ISSN (e): 2250 – 3005 || Vol, 04 || Issue, 9 || September – 2014 ||
International Journal of Computational Engineering Research (IJCER)

Iterative Determinant Method for Solving Eigenvalue Problems

Owus M. Ibearugbulem[1], Osasona, E. S. and Maduh, U. J.[2]
[1,2] Civil Engineering Department, Federal University of Technology, Owerri, Nigeria

ABSTRACT:
This paper presents iterative determinant method for solving eigenvalue problems. A matlab program that operates iterative to evaluate the determinant of the problem was written. In the program, a trial eigenvalue is used in the program to compute the determinant. A small value (iterator) is added to trial eigenvalue to obtain a new trial eigenvalue, which is used to compute new determinant. The trial eigenvalue in the sequence that made the determinants to move from negative values to positive value or move from positive values to negative value becomes required eigenvalue. Some engineering problems were used to validate the eigensolver. Some of the data from the eigensolver and the corresponding exact data are 2.468 and 2.4678; 9.876 and 9.875; 60.001 and 60.00; 12.37 and 12.3695. Close look at

Key word: *eigensolver, iterative, determinant, eigenvalue, matlab program, iterator*

I. INTRODUCTION

According to Ibearugbulem et al (2013), the stiffness matrix [k], the geometric matrix [kg] and the mass (inertia) matrix [ki] are formulated using the assumed shape function. This shape function is usually assumed to approximate the deformed shape of the continuum. If the shape function assumed is the exact one, it means the solution will converge to the exact solution. The inertia matrix and the geometric matrix formulated using the assumed shape functions are called consistent mass matrix and consistent geometric matrix respectively (Paz, 1980 and Geradin, 1980). In the work of Ibearugbulem et al (2013), the dynamic equation in structural dynamic as:

$$[k] - \lambda[ki] = 0 \quad (1)$$

For static stability problem, the eigenvalue equation is given as:

$$[k] - Nc[kg] = 0 \quad (2)$$

Where K is the material stiffness matrix, k_i is the matrix of inertia, k_g is the geometric matrix λ is natural frequency parameter and N_c is the critical buckling load parameter. Of note here is that both k_g and k_i are consistent matrices (that is they are non-diagonal matrices). The difficulty posed by this type of eigenvalue problem led many researchers to transform the consistent matrices to diagonal matrices as:

$$[k] - \lambda A[I] = 0 \quad (3)$$

$$[k] - Nc\, A[I] = 0 \quad (4)$$

Here, A[I] is the transformed diagonal inertia or geometric matrix. In dynamics, the diagonal inertia matrix is often called lumped mass matrix. Many methods are adopted so far by researchers for solutions of eigenvalue problems of equation (3) and (4). According to Ibearugbulem et al, (2013), the methods include Jacobi method, Polynomial method, Iterative method and Householder's method were used by (Greenstadt, 1960; Ortega, 1967; and James, Smith and Wolford, 1977. Others are Power method, Inverse iterative (Wilkinson, 1965), Lanczos method (lanczos, 1950), Arnoldi method (Arnoldi, 1951; Demmel, 1997; Bai et al, 2000;Chatelin, 1993; and Trefethenand Bau, 1997), Davidson method, Jacobi-Davidson method (Hochstenback and Notay, 2004; and Sleijpen and Vander Vorst, 1996), Minimum residual method, generalized minimum residual method were used by Barrett et al, (1994), Multilevel preconditioned iterative eigensolvers (Arbenz and Geus, 2005), Block inverse-free preconditioned Krylov subspace method (Quillen and Ye, 2010), Inner-outer iterative method (Freitag, 2007), and adaptive inverse iteration method (Chen, Xu and Zou, 2010), Matrix iterative-inversion (Ibearugbulem et al, 2013). Sadly, of all these methods, only polynomial and iterative-inversion methods can handle the problems of equations (1) and (2). Other methods can only be used for equation (3) and (4). However, polynomial method also becomes very difficult to use when the size of the matrix exceeds 3x3. In the same way, iterative-inversion method is also difficult when the matrix size is large and the speed of computation is very slow.

The essence of this paper is to present a method that can be used in solving eigenvalue problems of equations (1), (2), (3), and (4) for any size of matrix with high speed and good accuracy

II. ITERATIVE DETERMINANT

In this method, a trial eigenvalue λ_0 or Nc_0 of default value of zero shall be substituted into the eigenvalue equation to obtain eigenvalue matrix as:

$$[k]- \lambda[ki] = [kk]_0 \quad (5)$$

The determinant of the eigenvalue matrix, $[kk]_0$ shall then be computed and the value kept. The value of this initial determinant, dt0 may be positive or negative. Now the value of the default eigenvalue shall be increased by adding iterator (say 1, 0.1, 0.01, 0.001 etc. as desired) to it to obtain new eigenvalue, λ_1. This shall be substituted into the eigenvalue equation to obtain $[kk]_1$. The determinant, dt_1 of $[kk]_1$ shall be computed. If dt_0 is negative and dt1 is also negative, then the actual eigenvalue has not been obtained and the iterator has to be added to λ_1 to obtain λ_2. This process has to be continued until we reach a stage where $\lambda_{n-1} > 0$ and $\lambda_n < 0$ or $\lambda_{n-1} < 0$ and $\lambda_n > 0$. Note here that the bigger the iterator, the faster the computation and less the accuracy, the smaller the iterator, the slower the computation and more the accuracy. To ease the use of this method, a general matlab program was written. The user is at will to modify the iterator, f and sub iterators ff (i) in the program, where is 1, 2, 3 … n and n is the size of the square matrix involved. For instance, if the order of the expected eigenvalue is 100, iterator and sub iterators of 1 or 0.1 can be used. In this case the expected accuracy shall be of the order of iterator, 1 or 0.1. In the same way, if the order of expected eigenvalue is 1, iterator of 0.001 can be used. The choice of iterator is guided by the speed and accuracy of the computation. In all, for high accuracy, iterators of 0.001 and below are advisable but the speed may be very slow. Thus, the level of accuracy and speed shall guide you in using your choice iterator in this program. The user is also at liberty to choose the range of eigenvalues to compute for by adjusting "for r = 1:3" to say "for r = 1:5" if the size of the matrix is 5x5. The stiffness matrix "prs" and the geometric matrix or inertia matrix "pri" in the program are to be entered by the user. The iterating count ceiling, "k" in the program can also be raised from 6260 to any higher value to suit the user. Number of sub iterators "ff(i) as we have them in the program is 7. The user is at liberty also to introduce more sub iterators say ff(8), ff(9) etc. as matches the size of the matrix. Some engineering problems were used to ascertain the adequacy of the method and the program.

III. NUMERICAL EIGENVALUE PROBLEMS

The program will be used to test the following problems.

1. $\left|\begin{bmatrix} 2 & 0 & 1 \\ -1 & 4 & -1 \\ -1 & 2 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right| = 0$ (Stroud, 1982)

2. $\left|\begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.2 \\ 0.1 & 0.2 & 0.3 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right| = 0$ (James, Smith and Wolford, 1977)

3. $\left|\begin{bmatrix} 204.8 & -102.4 & 25.6 \\ -102.4 & 63.2 & -18.8 \\ 25.6 & -18.8 & 7.2 \end{bmatrix} - \lambda \begin{bmatrix} 4.8762 & -2.438 & 0.0762 \\ -2.438 & 2.419048 & -0.1381 \\ 0.0762 & -0.1381 & 0.0762 \end{bmatrix}\right| = 0$

4. $\left|\begin{bmatrix} 7.2 & -25.6 & -1.2 \\ -25.6 & 204.8 & 25.6 \\ -1.2 & 25.6 & 7.2 \end{bmatrix} - \lambda \begin{bmatrix} 0.07619 & -0.076 & 0.02381 \\ -0.07619 & 4.8762 & 0.07619 \\ 0.02381 & 0.0762 & 0.07619 \end{bmatrix}\right| = 0$

5. $\left|\begin{bmatrix} 204.8 & -102.4 & 25.6 \\ -102.4 & 63.2 & -18.8 \\ 25.6 & -18.8 & 7.2 \end{bmatrix} - \lambda \begin{bmatrix} 0.406349 & 0.0635 & -0.00635 \\ 0.0635 & 0.2063 & -0.01587 \\ -0.00635 & -0.016 & 0.001587 \end{bmatrix}\right| = 0$

6. $\begin{vmatrix} 126.4 & 18.8 & 18.8 & -102.4 & 0 & -102.4 \\ 18.8 & 14.4 & -1.2 & -25.6 & -25.6 & 0 \\ 18.8 & -1.2 & 14.4 & 0 & 25.6 & -25.6 \\ -102.4 & -25.6 & 0 & 204.8 & 0 & 0 \\ 0 & -25.6 & 25.6 & 0 & 204.8 & 0 \\ -102.4 & 0 & -25.6 & 0 & 0 & 204.8 \end{vmatrix} - \lambda \begin{vmatrix} 2.419048 & 0.138095 & 0.138095 & -2.4381 & 0 \\ 0.138095 & 0.15238 & 0.02381 & -0.07619 & -0.07619 \\ 0.138095 & 0.02381 & 0.15238 & 0 & 0.07619 \\ -2.4381 & -0.07619 & 0 & 4.87619 & 0 \\ 0 & -0.07619 & 0.07619 & 0 & 4.87619 \\ -2.4381 & 0 & -0.07619 & 0 & 0 \end{vmatrix} = 0$

IV. RESULT AND DISCUSSION

Table 1 shows result data from eigenvalue problems herein. Eigenvalues obtained from the program were compared with the exact eigenvalues. The exact eigenvalues were obtained by trial and error means using Microsoft excel worksheet. Any eigenvalue that made the determinant of the eigenvalue matrix [kk] to become zero is the exact eigenvalue. The data from the program compared very well with the values from Microsoft excel worksheet with high degree of accuracy. As said earlier, it was observed that with small iterators say 0.01, we obtained more accurate eigenvalues with slow computing speed. We also confirmed that with big iterators say 1.0 we obtained less accurate eigenvalues with fast computing speed.

Table 1: Result data from the Eigenvalue Problems

	Eigenvalues									
	From iterative Determinant method					**Exact Eigenvalues**				
Problems	1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th
1	1	2	3			1	2	3		
2	0.0308	0.0644	0.5049			0.0308	0.0644	0.5049		
3	2.468	23.392	109.143			2.4678	23.3912	109.1422		
4	9.876	60.001	170.128			9.875124	60	170.1276		
5	12.37	494.266				12.3695	494.2658			
6	15.1	27	42.1	83.1	133.8	15.075832	26.95929142	42	83.07804	133.72546

V. APPENDIX A (MATLAB PROGRAM)

```
nn = input('enter size of matrix');nn = nn*1;
p=0;m=1;k=6260;j=1;f=0.1;ff(1)=0.1;ff(2)=0.01;ff(3)=0.001;ff(4)=0.001;ff(5)=0.001;ff(6)=0.001;ff(7)=0.001;
for r = 1 : 3
while p < k
prs = [43.2 0   0   ;0  6.400093    0   ;0  0   6.4];
pri = [0.54824  0   0   ;0  0.01268 0   ;0  0   0.01268];
a = prs - p*pri;
for x = 1:nn
for y = 1:nn
c(x,y)= a(x,y) ;
end
end
d = det(c);
if (m < 1.1); t1 = d;end
m = m + 1;
if (j >nn); break; end
if ((d >= 0)&& (t1 <= 0));py(j) = p;j = j + 1;t1 = d;end
if ((d <= 0)&& (t1 >= 0));py(j) = p;j = j + 1;t1 = d; end
p = p + f;
end
p = p-f; f =ff(r);m=1;
end
```

REFERENCES

[1] Arnoldi, W. E. (1951). The principle of minimized iteration in the solution of the matrix eigenvalue problem, Quarterly of Applied Mathematics, vol. 9, pp. 17– 29.

[2] Arbenz, P. andGeus, R. (2005). Multilevel preconditioned iterative eigensolvers for maxwell eigenvalue problems. Applied numerical mathematics.Vol. 54 , issue 2 .pp 107 – 121

[3] Bai, Z. Demmel, J., Dongarra, J. Ruhe, A., and van der Vorst, H. (2000). Templates for the Solution of Algebraic Eigenvalue Problems - A Practical Guide, SIAM, Philadelphia, PA,

[4] Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C.and van der Vorst, H. A. (1994). Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition, SIAM, Philadelphia, PA,

[5] Chatelin, F. (1993).Eigenvalues of matrices,JohnWiley&SonsLtd,Chichester,West Sussex, England, Originally published in two separate volumes by Masson, Paris: Valeurspropres de matrices (1988) and Exercises de valeurspropres de matrices (1989).

[6] Demmel, J. W. (1997). Applied Numerical Linear Algebra, SIAM, Philadelphia, PA A.

[7] Freitag, M. (2007).Inner-outer Iterative Methods for Eigenvalue Problems - Convergence and Preconditioning.PhD Thesis submitted to University of Bath
[8] Fullard, K. (1980). The Modal Method for Transient Response and its Application to Seismic Analysis. In J. Donea (Ed.), Advanced Structural Dynamics (pp.43-70)
[9] Geradin, M. (1980).Variational Methods of Structural Dynamics and their Finite Element Implementation. In J. Donea (Ed.), Advanced Structural Dynamics (pp.1-42)
[10] (Greenstadt, J. (1960). Mathematical Methods for Digital Computers. Vol. 1, ed. A. Ralston and H. S. Wilf (New York: John Wiley & sons). Pp. 84-91.
[11] Hochstenbach, M. E. andNotay, Y. (2004). The Jacobi-Davidson method, GAMM Mitt.,
[12] Ibearugbulem, O. M., Ettu, L. O., Ezeh, J.C., and Anya, U.C. (2013).Application of Matrix Iterative – Inversion in Solving Eigenvalue Problems in Structural Engineering. International Journal of Computational Engineering Research, vol.3. pp: 17-22
[13] James, M. L., Smith, G. M. and Wolford, J. C. (1977). Applied Numerical Methods for Digital Computation: with FORTRAN and CSMP. Harper and Row Publishers, New York.
[14] Key, S. W. (1980). Transient Response by Time Integration: Review of Implicit and Explicit Operators. In J. Donea (Ed.), Advanced Structural Dynamics.
[15] Key, S. W. and Krieg, R. D. (1972) Transient Shell Response by numerical Time Integration.2nd US-JAPAN Seminar on Matrix Methods in Structural Analysis. Berkeley, California.
[16] Lanczos, C. (1950). An iterative method for the solution of the eigenvalue problem of linear differential and integral operators, Journal of Research of the National Bureau of Standards, vol. 45 , pp. 255–282.
[17] [Ortega, J. (1967). Mathematical Methods for Digital Computers. Vol. 2, ed. A. Ralston and H. S. Wilf (New York: John Wiley & sons). Pp. 94-115.