Applied Mathematics, **2011, 2, 942-946**
doi:10.4236/am.2011.28129 Published Online August 2011 (http://www.SciRP.org/journal/am)

Analyzing the Stability of a *n-DOF* System with Viscous Damping

Hashem Saberi Najafi*, Amir Hosein Refahi Sheikhani
Department of Applied Mathematics, Faculty of Mathematical Sciences, University of Guilan, Rasht, Iran
E-mail: *hnajafi@guilan.ac.ir, ah_refahi@yahoo.com*
Received January 22, 2011; *revised May* 31, 2011; *accepted June* 7, 2011

Abstract

In this paper we introduce a numerically stable method for determining the stability of *n-DOF* system without computing eigenvalues. In this sense, at first we reduce the second-order system to a standard eigenvalue problem with symmetric tridiagonal form. Then we compute the exact inertia by using an algorithm based on floating point arithmetic [1]. Numerical tests report the effectiveness of these methods.

Keywords: *n-DOF* System, Inertia, Stability, Lanczos

1. Introduction

The matrix second-order system

$$\boldsymbol{M}\ddot{x}(t)+\boldsymbol{C}\dot{x}(t)+\boldsymbol{K}x(t)=0$$

with real coefficient matrices $\boldsymbol{M}$, $\boldsymbol{C}$ and $\boldsymbol{K}$ arises in a wide variety practical applications such as in the mechanical vibrations, and structural design analysis. Many important characteristic of physical and engineering systems, such as stability and inertia can often be determined only by knowing the nature and location of the eigenvalues. It is well known that the stability of a physical system modeled by a system of differential equation is determined just by knowing if the eigenvalues of the system matrix have all negative real parts. In many engineering applications, it may not be enough to determine if the system is stable [2]. A problem more general than the stability problem is the inertia problem. The inertia of a matrix (denoted by $In(\boldsymbol{A})$) is the triplet of the numbers of the eigenvalues of $\boldsymbol{A}$ with positive, negative and zero real parts. There are reliable algorithms to compute the inertia and stability of a *n-DOF* system with viscous damping. Some of these algorithms are numerically unstable and are primarily of theoretical interest [3]. Another group of these methods are not practical for system of large numbers of degrees of freedom. The following are the usual computational approaches for determining the inertia of a nonsymmetric matrix $\boldsymbol{A}$.

1) Compute the eigenvalues of $\boldsymbol{A}$ explicitly.
2) Compute the characteristic polynomial of $\boldsymbol{A}$ and then apply the well-known Routh-Hurwitz criterion.
3) Solve the Lyapunov equation

$$\boldsymbol{XA}+\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X}=-\boldsymbol{C}$$

The second approach is usually discarded as a numerical approach [2] and the last approach is counterproductive. Thus, the only viable way, from a numerical viewpoint, of determining the inertia of a matrix, is to compute explicitly its eigenvalues. Carlson and Datta described a computational method for determining the inertia of a nonsymmetric matrix [4,5]. The method is based on the implicit solution of a special Lyapunov equation. But this method is not practical for large and sparse matrices [2]. The paper is organized as follows. In Section 2, we introduce some important theorems and applications of inertia ans stability problem. Then we describe two new methods for computing the inertia of a large sparse nonsymmetric matrix in Sections 3 and 4. Finally, the conclusion are given in the last section.

2. Inertia and Stability

Theorem 2.1. A homogeneous system of differential equation with constant coefficients of the form

$$\dot{x}(t)=\boldsymbol{A}x(t) \tag{2.1}$$

is asymptotically stable if and only if all the eigenvalues of $\boldsymbol{A}$ have negative real parts.

Proof. see [2].

Definition 2.2. *A matrix* $\boldsymbol{A}$ *is called a stable matrix*

if all of the eigenvalues of A *have negative real parts.*

Knowing that the system (2.1) is asymptotically stable if and only if A is a stable matrix.

Definition 2.3. *The inertia of a matrix order* n*, is the triplet* $(\pi(A),\nu(A),\delta(A))$ *where* $\pi(A)$, $\nu(A)$ *and* $\delta(A)$ *are, respectively, the number of eigenvalues of* A *with positive, negative and zero real parts.*

Not that $\pi(A)+\nu(A)+\delta(A)=n$, and A is a stable matrix if and only if $In(A)=(0,n,0)$.

Definition 2.4. *The second order differential equations*

$$M\ddot{x}(t)+C\dot{x}(t)+Kx(t)=0 \tag{2.2}$$

is asymptotically stable(that is, $\mathbf{P}x(t)\mathbf{P}\to 0$ *as* $t\to 0$), if and only if all the eigenvalues of the quadratic pencil

$$p(\lambda)=\lambda^2 M+\lambda C+K \tag{2.3}$$

have negative real parts. Similarly, by the inertia of the quadratic pencil (2.3) is defined to be the triplet of the numbers of eigenvalues of $p(\lambda)$ with positive, negative, and zero real parts.

Remark 2.5. *The effective numerical methods for the quadratic eigenvalue problem are still not well developed, especially for large and sparse problems that arise in practical applications. In case of general damping, assuming that the solutions are of the form* $y=x\mathrm{e}^{\lambda t}$ *, where* x *is a constant vector, we will have the quadratic eigenvalue problem*

$$\left(\lambda^2 M+\lambda C+K\right)x=0 \tag{2.4}$$

that we can rewrite (2.3) as

$$\begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{pmatrix}\begin{pmatrix} x \\ \lambda x \end{pmatrix}=\lambda\begin{pmatrix} x \\ \lambda x \end{pmatrix}$$

or

$$Au=\lambda u \tag{2.5}$$

where

$$A=\begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{pmatrix} \quad u=\begin{pmatrix} x \\ \lambda x \end{pmatrix}$$

Thus we have reduced the quadratic eigenvalue problem (2.3) to the $2n\times 2$ standard eigenvalue problem (2.5), on the other hand, for determining the stability of (2.2), we can compute $In(A)$ in stead of $In(p(\lambda))$.

Example 2.6. *Consider the* 4*-story building of equal rigid and equal interstory stiffness. The floors and roof are represented by lumped masses* m_1 *to* m_4 *and* k_1 *to* k_4 *are equivalent spring constants of columns that act as springs in parallel with viscous damper* c_1 *to* c_4*. The equations of motion for the system, in matrix form, can be written as*

$$M\ddot{y}(t)+C\dot{y}(t)+Ky(t)=0 \tag{2.6}$$

where M , C , K are, respectively, mass matrix, damping matrix and stiffness matrix represented in follow:

$$M=diag(5.53,4.93,4.925,4.94)$$

$$C=\begin{pmatrix} 25.4091 & -19.5165 & 0 & 0 \\ -19.5165 & 39.9733 & -15.2036 & 0 \\ 0 & -15.2036 & 35.6427 & -15.1965 \\ 0 & 0 & -15.1965 & 35.4753 \end{pmatrix}$$

$$K=\begin{pmatrix} 39.3665 & -19.8015 & 0 & 0 \\ -19.8015 & 39.6122 & -19.8107 & 0 \\ 0 & -19.8107 & 45.2413 & -25.4306 \\ 0 & 0 & -25.4306 & 25.4306 \end{pmatrix}$$

We would like to study the stability (2.6) with determining the inertia of $p(\lambda)$ or $In(A)$ where

$$A=\begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{pmatrix}$$

$In(p(\lambda))=In(A)=(0,8,0)$ thus the system n*-DOF* (2.6) is stable.

Now by removing K_1 we have $In(p(\lambda))=In(A)=(0,7,1)$. On the other hand the structure loses its stability.

Remark 2.7. *According to the example* (2.6) *, in the case that the degree of freedom is small, the stability of* n*-DOF system and* $In(A)$ *can be routinely determine. However for system of large numbers of degrees of freedom the available methods can be costly. Specially that we would like to determine the* $In(A)$ *without computing the eigenvalues.*

3. Shifted Lanczos Process

In this Section we provide a stable numerically method for determination of a nonsymmetric matrix. Our scheme is first to reduce a given matrix A to a symmetric tridiagonal form with a Lanczos process, and then compute the exact inertia of a symmetric matrix by a floating point algorithm [5].

***Step* 1.** (*Lanczos process*): Given vectors v_1 and w_1 such that $v_1^{\mathrm{T}}w_1=1$, this process provide a tridiagonal

matrix T_n of the form

$$T_n = \begin{pmatrix} a_1 & b_1 & & & \\ d_1 & a_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & a_{n-1} & b_{n-1} \\ & & & d_{n-1} & a_n \end{pmatrix}_{n\times n}$$

and produces the basis $V_n = [v_1, ..., v_n]$ and $W_n = [w_1, ..., w_n]$, respectively, for the Krylov subspaces $K_n(A^T, v_1)$ and $K_n(A, w_1)$ which satisfies the relations:

$$AW_n = W_n T_n + d_{n+1} w_{n+1} e_n^{\mathrm{T}} \tag{3.1}$$

$$A^{\mathrm{T}} V_n = V_n T_n^{\mathrm{T}} + b_{n+1} v_{n+1} e_n^{\mathrm{T}} \tag{3.2}$$

$$V_n^T W_n = I_n$$

Thus

$$V_n^T A W_n = T_n$$

***Step* 2.** (*determining the exact inertia*):

Theorem 3.1. (*The Sylvester law of inertia*)

Let A be a Hermitian matrix and P be a non-singular matrix. Then $In(A) = In(PAP^{\mathrm{T}})$. The following algorithm by using the Sylvester law of inertia describes a floating point process to compute the exact inertia of a symmetric tridiagonal matrix [5]. In this algorithm $a = (a_1, \cdots, a_n)$, $z = (z_1, \cdots, z_{n-1})$ such that $z_i = b_i^2$ for $(i = 1, \cdots, n-1)$ and τ is a proper shift parameter.

Algorithm 1. (*inertia of a symmetric tridiagonal matrix*)

Function $(\pi, \nu, \delta) = inertia(a, z, \tau)$

$d^+ = inertia^+(a, z, \tau)$

$d^- = inertia^-(a, z, \tau)$

for $i = 1, \cdots, n$

if $\mathrm{sign}(d_i^+) = \mathrm{sign}(d_i^-)$ then

if $d_i^+ < 0$ then $\nu = \nu + 1$

else if $d_i^+ > 0$ then $\pi = \pi + 1$

end if

end if

end do

$\delta = n - (\nu + \pi)$

End.

Example 3.2. *Consider the nonsymmetric matrix A as the form*:

$$\begin{pmatrix} 5-n & 0.21 & 1.2 & 0 & .13 & 1.42 & 0 & \cdots & \cdots & 0 \\ 0.45 & 5-n+1 & & 1.2 & 0 & 0.13 & 1.42 & 0 & \ddots & 0 \\ 0.34 & 0.45 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\ 0 & 0.34 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 1.42 & 0 \\ 0.12 & 0 & \ddots & \ddots & (n/2-n+6) & \ddots & \ddots & \ddots & 0.13 & 1.42 \\ 0.11 & 0.12 & \ddots & \ddots & \ddots & n/2+3 & \ddots & \ddots & 0 & 0.13 \\ 0 & 0.11 & \ddots & \ddots & \ddots & \ddots & n/2+4 & \ddots & 1.2 & 0 \\ \vdots & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0.21 & 1.2 \\ \vdots & \vdots & 0 & 0.11 & 0.12 & 0 & 0.34 & 0.45 & n-1 & 0.21 \\ 0 & 0 & \cdots & 0 & 0.11 & 0.12 & 0 & 0.34 & 0.45 & n \end{pmatrix}_{2n\times 2n}$$

We apply Shifted Lanczos process to compute the exact inertia of A. This algorithm has been tested when the dimension of matrix A increases. The results are shown in **Table 1**.

In **Table 1** the column of error is the precision of transforming the matrix A to a tri-diagonal matrix. Note that if the error is small, then the inertia of A can be computed correctly. But if the error is not small, this dose not mean that the inertia of A cannot be computed, in this case by choosing a proper shift, the inertia of A will be computed. Shift intervals are seen in **Table 1**. The best case is when the shifted parameter is zero. In that case the amount of computations is less, that is why we have a column called $In_0(A)$ in **Table 1** to have more information. Also the results show that by increasing the dimension the matrix this method does not work very well.

4. Weighted Shifted Lanczos Method

According to the results shown in **Table 1**, we can see that the shifted Lanczos method computes the inertia accurately, when the matrix is not so large, but does not have an exact results when the dimension is large. The

Table 1. Implementation of Shifted Lanczos method for determining inertia of a *n-DOF* system with different values of degree of freedom.

$DOF(n)$	$\|V_n^{\mathrm{T}}AW_n - T_n\|$	*Shift interval*(τ)	$In_0(A)$	*Situation*	*Time*
8	$7.65E-013$	$[-0.5, 0.5]$	$(4,4,0)$	*exact*	0.0013
16	$1.21E-010$	$[-1.5, 1.5]$	$(6,10,0)$	*exact*	0.0026
32	$1.09E-007$	$[-9, 9]$	$(14,18,0)$	*exact*	0.0070
64	0.3317	$[-25, 25]$	$(30,34,0)$	*exact*	0.0235
128	608.34	$[66, 67]$	$(63,65,0)$	*fail*	0.0852
256	1591	$[129.5, 131]$	$(127,129,0)$	*fail*	0.3774
512	27351	$[60, 220]$	$(255,257,0)$	*fail*	2.2154

reason for the above is that, the non-hermitian Lanczos method is not an orthogonal projection comparing with the other Krylov subspace methods. Because in this method the matrices $\boldsymbol{V}_n$ and $\boldsymbol{W}_n$ are not orthogonal, but anyhow we have $\boldsymbol{W}_n^{\mathrm{T}}\boldsymbol{V}_n = \boldsymbol{I}_n$, and for this reason the method is an oblique projection method. In this section we have tried to decrease the error by making changes in the Lanczos algorithm to be able to develop an effective method for computing the exact inertia of a large sparse nonsymmetric matrices. Using (3.1) and (3.2) we have:

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{V}_n - \boldsymbol{V}_n\boldsymbol{T}_n^{\mathrm{T}} = b_{n+1}\boldsymbol{v}_{n+1}\boldsymbol{e}_n^{\mathrm{T}} \tag{4.1}$$

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{V}_n - \boldsymbol{V}_n\boldsymbol{T}_n^{\mathrm{T}} = b_{n+1}\boldsymbol{v}_{n+1}\boldsymbol{e}_n^{\mathrm{T}} \tag{4.2}$$

The right side of the above relations indicates the error of oblique projection in the Lanczos method. We multiply the both sides of (4.1) and (4.2) by a small scaler $\beta > 0$ with the hope that to prevent the increasing error in the Lanczos process. Thus we obtain

$$\beta(\boldsymbol{A}\boldsymbol{W}_n - \boldsymbol{W}_n\boldsymbol{T}_n) = \beta d_{n+1}\boldsymbol{w}_{n+1}\boldsymbol{e}_n^{\mathrm{T}} \tag{4.3}$$

$$\beta\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{V}_n - \boldsymbol{V}_n\boldsymbol{T}_n^{\mathrm{T}}\right) = \beta b_{n+1}\boldsymbol{v}_{n+1}\boldsymbol{e}_n^{\mathrm{T}} \tag{4.4}$$

Now let

$$\beta\boldsymbol{V}_n^{\mathrm{T}}\boldsymbol{W}_n = \boldsymbol{I}_n$$

Thus we have

$$\beta\boldsymbol{V}_n^{\mathrm{T}}\boldsymbol{A}\boldsymbol{W}_n = \boldsymbol{T}_n \tag{4.5}$$

Therefore for holding (4.5) we must construct a pair β orthogonal basis $\boldsymbol{V}_n$ and $\boldsymbol{W}_n$, respectively, for the two following Krylov subspaces

$$K_n\left(\boldsymbol{A}^{\mathrm{T}}, \boldsymbol{v}\right) = span\left\{v, \boldsymbol{A}^{\mathrm{T}}\boldsymbol{v}, \cdots, \left(\boldsymbol{A}^{\mathrm{T}}\right)^{n-1}\boldsymbol{v}\right\}$$

$$K_n\left(\boldsymbol{A}, \boldsymbol{w}\right) = span\left\{\boldsymbol{w}, \boldsymbol{A}\boldsymbol{w}, \cdots, \left(\boldsymbol{A}\right)^{n-1}\boldsymbol{w}\right\}$$

The following algorithm computes exact inertia of a nonsymmetric matrix $\boldsymbol{A}$, use transforming $\boldsymbol{A}$ to a tridiagonal form by the weighted shifted Lanczos process.

Algorithm 2. (*weighted shifted Lanczos process*)

Input a shift parameter τ

Choose two vectors $\boldsymbol{v}_1$ and $\boldsymbol{w}_1$ such that $(\boldsymbol{v}_1, \boldsymbol{w}_1) = 1/\beta$

set $b_1 = p_0 = q_0 = 0$.

For $j = 1, 2, \cdots, n$ do

$a_j = \beta\left(Aw_j, p_j\right)$

$r_{j+1} = Aw_j - a_j w_j - b_j w_{j-1}$

$s_{j+1} = A^{\mathrm{T}}v_j - r_j v_j - d_j v_{j-1}$

$d_{j+1} = \sqrt{\beta \geq \left|\left(r_{j+1}, s_{j+1}\right)\right|}$

$b_{j+1} = \beta\left(r_{j+1}, s_{j+1}\right)/d_{j+1}$

$v_{j+1} = s_{j+1}/b_{j+1}$

$w_{j+1} = v_{j+1}/d_{j+1}$

End for

For $i = 1, 2, \cdots, n-1$ do

$z_i = b_i^2$

End for

Set $a = \left(a_1, \cdots, a_n\right)$ and $z = \left(z_1, \cdots, z_{n-1}\right)$

$(\pi, \nu, \delta) = inertia\left(a, z, \tau\right)$

End.

Example 4.1. *Let A be the same matrix that used in Example* 1 *and we increase its dimension orderly. We apply algorithm* 3 *to find the exact inertia of* $\boldsymbol{A}$. *The results for different values of* n *are shown in* **Table 2**.

According to the **Table 2**, we can see that by decreasing the error of oblique projection, $In(\boldsymbol{A})$ can be computed accurately without consuming more time.

Example 4.2. *According to the results in* **Table 1** *and* **Table 2**, *we see that the Weighted Shifted Lanczos in comparison with Shifted Lanczos method works better. Now consider* $\boldsymbol{A}$ *is the same matrix that used in Example* 3.1. *We apply our two computational methods*

Table 2. Implementation of Weighted Shifted Lanczos method for determining inertia of a *n-DOF* system with different values of degree of freedom.

$DOF(n)$	$\|V_n^{\mathrm{T}}AW_n - T_n\|$	*Shift interval*(τ)	$In_0(A)$	*Situation*	*Time*
8	$3.84E-014$	$[-1.01, 1.01]$	$(4,4,0)$	*exact*	0.0035
16	$1.51E-012$	$[-4.5, 4.5]$	$(6,10,0)$	*exact*	0.0050
32	$2.11E-011$	$[-18, 18]$	$(14,18,0)$	*exact*	0.0077
64	$8.87E-011$	$[-31.5, 31.5]$	$(30,34,0)$	*exact*	0.0269
128	$4.19E-010$	$[-39, 39]$	$(62,62,0)$	*exact*	0.0927
256	$1.69E-009$	$[-50, 50]$	$(126,130,0)$	*exact*	0.3916
512	$9.23E-009$	$[-65, 65]$	$(254,258,0)$	*exact*	2.3921

Table 3. Implementation of Shifted Lanczos and Weighted shifted Lanczos methods for large values of degree of freedom.

$DOF(n)$	Shifted Lacczos method			Weighted Shifted Lacczos method		
	$In_0(A)$	situation	Time	$In_0(A)$	situation	Time
800	(397, 403, 0)	*fail*	7.65	(398, 402, 0)	*exact*	8.35
1024	(513, 511, 0)	*fail*	15.82	(510, 514, 0)	*exact*	17.02
1200	(597, 603, 0)	*fail*	22.735	(598, 602, 0)	*exact*	24.838
1400	(699, 701, 0)	*fail*	35.813	(698, 702, 0)	*exact*	37.77
1600	(801, 799, 0)	*fail*	55.296	(798, 802, 0)	*exact*	57.74
1800	(900, 900, 0)	*fail*	73.639	(898, 902, 0)	*exact*	78.761
2048	(1023, 1025, 0)	*fail*	114.49	(1022, 1026, 0)	*exact*	123.92

to compute the exact inertia of A *when the dimension of the matrix is large. Results are shown in* **Table 3**.

Table 3 shows that the weighted shifted Lanczos method works very well for large sparse matrices and in any case the exact inertia can be computed.

5. Comments and Conclusion

Two new iterative methods presented in this paper can compute $In(A)$ in the case that A is a nonsymmetric large sparse matrix. However the shifted Lanczos method may not be able to compute the exact inertia of a nonsymmetric large sparse matrices, but the results show that they do not have a big difference with the exact solutions. Therefore this method can be used for the application of many engineering problems like, vibration problems which needs to be aware of the behavior of the eigenvalues.

6. References

[1] K. V. Fernando, "Computation of Exact Inertia and Inclusions of Eigenvalues of Tridiagonal Matrices," *Linear Algebra and Its Applications*, Vol. 422, No. 1, 2007, pp. 71-99. doi:10.1016/j.laa.2006.09.008

[2] D. Carlson and B. N. Datta, "The Lyapunov Matrix Equation $SA + A^*S = S^*B^*BS$," *Linear Algebra and Its Applications*, Vol. 28, 1979, pp. 43-52. doi:10.1016/0024-3795(79)90117-4

[3] A. C. Antoulas and D. C.Sorensen, "Lyapunov, Lanczos and Inertia," *Linear Algebra and Its Applications*, Vol. 326, No. 1-3, 2001, pp. 137-150. doi:10.1016/S0024-3795(00)00288-3

[4] D. Carlson and B. N. Datta, "On the Effective Computation of the Inertia of a Nonhermitian Matrix," *Numerical Mathematics*, Vol. 33, No. 3, 1979, pp. 315-322. doi:10.1007/BF01398647

[5] B. N. Datta, "Stability and Inertia," *Linear Algebra and Its Applications*, Vol. 302-303, 2000, pp. 563-600. doi:10.1016/S0024-3795(99)00213-X

