Journal of Mathematical Sciences: Advances and Applications
Volume 7, Number 2, 2011, Pages 105-114

NEW NUMERICAL METHOD FOR DETERMINING INERTIA AND STABILITY OF NONSYMMETRIC LARGE SPARSE MATRIX

H. SABERI NAJAFI and A. H. REFAHI SHEIKHANI

Department of Applied Mathematics
Faculty of Mathematical Sciences
University of Guilan
Rasht
Iran
e-mail: hnajafi@guilan.ac.ir

Abstract

In this paper, we present a numerically stable method for determining the exact inertia of a non-symmetric large sparse matrices without computing eigenvalues. For doing this scheme, at first, we reduce a non-symmetric matrix to a symmetric tridiagonal form in a finite number of steps with a new algorithm based on the Krylov subspace method. Then, we compute the exact inertia by using an algorithm based on floating point arithmetic. Numerical tests report the effectiveness of these methods.

1. Introduction

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 modelled by a system of differential equation is determined just by knowing, if the eigenvalues of the system matrix have

2010 Mathematics Subject Classification: 65L15, 65F50.

Keywords and phrases: Lanczos, exact inertia, non-symmetric, tridiagonal form.

Received February 19, 2011

all negative real parts. In many engineering applications, it may not be enough to determine if the system is stable, see [2]. A problem more general than the stability problem is the inertia problem. The inertia of a matrix is the triplet of the numbers of the eigenvalues of A with positive, negative, and zero real parts. There are reliable algorithms to compute the inertia of a non-symmetric matrix. Some of these algorithms are numerically unstable and are primarily of theoretical interest, see [1]. Another group of these methods are not practical for large and sparse matrices. The following are the usual computational approaches for determining the inertia of a non-symmetric matrix A:

(1) Compute the eigenvalues of A explicitly.

(2) Compute the characteristic polynomial of A and then apply the well-known Routh-Hurwitz criterion.

(3) Solve the Lyapunov equation

$$XA + A^T X = -C.$$

The second approach is usually discarded as a numerical approach, see [2]. 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 non-symmetric matrix, see [3, 4]. The method is based on the implicit solution of a special Lyapunov equation. But, this method is not practical for large and sparse matrices, see [2].

The paper is organized as follows. In Section 2, we recall some fundamental results of the inertia and stability. Then, we describe two new methods for determining the inertia of a non-symmetric matrix in Sections 3, 4. Moreover, several numerical examples are presented to illustrate the efficiency of the proposed methods in Sections 3, 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*

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

is asymptotically stable, if and only if all the eigenvalues of A have negative real parts.

Proof. See [2].

Definition 2.2. A matrix 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, denoted by $\mathrm{In}(A)$, 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.

Note that $\pi(A) + \nu(A) + \delta(A) = n$, and A is a stable matrix, if and only if $\mathrm{In}(A) = (0, n, 0)$.

3. Shifted Lanczos Process

In this section, we provide a stable numerically method for determination of a non-symmetric 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, see [5].

Step 1 (Lanczos process)**.** Given vectors v_1 and w_1 such that $v_1^T w_1 = 1$, 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 $V_n = [v_1, \ldots, v_n]$ and $W_n = [w_1, \ldots, w_n]$, respectively, for the Krylov subspaces $K_n(A^T, v_1)$ and $K_n(A, w_1)$, which satisfies the relations:

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

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

$$V_n^T W_n = I_n.$$

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 nonsingular matrix. Then* $\mathrm{In}(A) = \mathrm{In}(PAP^T)$.

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, see [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 $(\pi, \nu, \delta) = \mathrm{inertia}\,(a, z, \tau)$

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

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

for $i = 1, \ldots, n$

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 non-symmetric matrix A as the form:

$$\begin{pmatrix} 5-n & .21 & 1.2 & 0 & .13 & 1.42 & 0 & \cdots & \cdots & 0 \\ .45 & 5-n+1 & .21 & 1.2 & 0 & .13 & 1.42 & 0 & \ddots & 0 \\ .34 & .45 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\ 0 & .34 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 1.42 & 0 \\ .12 & 0 & \ddots & \ddots & (n/2-n+6) & \ddots & \ddots & \ddots & .13 & 1.42 \\ .11 & .12 & \ddots & \ddots & \ddots & n/2+3 & \ddots & \ddots & 0 & .13 \\ 0 & .11 & \ddots & \ddots & \ddots & \ddots & n/2+4 & \ddots & 1.2 & 0 \\ \vdots & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & .21 & 1.2 \\ \vdots & \vdots & 0 & .11 & .12 & 0 & .34 & .45 & n-1 & .21 \\ 0 & 0 & \cdots & 0 & .11 & .12 & 0 & .34 & .45 & n \end{pmatrix}_{n\times n}.$$

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.

Table 1. Implementation of shifted Lanczos method for determining $\mathrm{In}(A)$ with different values of n

n	$\|V_n^T AW_n - T_n\|$	Shift interval (τ)	$\mathrm{In}_0(A)$	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

In Table 1, the column of error is the precision of transforming the matrix A to a tridiagonal 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 does 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 $\mathrm{In}_0(A)$ in Table 1 to have more information. Also, the results show that by increasing the dimension of 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 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 V_n and W_n are not orthogonal, but anyhow, we have $W_n^T V_n = I_n$, 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 non-symmetric matrices. Using (3.1) and (3.2), we have

$$AW_n - W_n T_n = d_{n+1} w_{n+1} e_n^T, \tag{4.1}$$

$$A^T V_n - V_n T_n^T = b_{n+1} v_{n+1} e_n^T. \tag{4.2}$$

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

$$\beta(AW_n - W_nT_n) = \beta d_{n+1}w_{n+1}e_n^T, \tag{4.3}$$

$$\beta(A^TV_n - V_nT_n^T) = \beta b_{n+1}v_{n+1}e_n^T. \tag{4.4}$$

Now let

$$\beta V_n^T W_n = I_n.$$

Thus, we have

$$\beta V_n^T AW_n = T_n. \tag{4.5}$$

Therefore, for holding (4.5), we must construct a pair β orthogonal basis V_n and W_n, respectively, for the two following Krylov subspaces:

$$K_n(A^T, v) = \text{span}\{v, A^Tv, \ldots, (A^T)^{n-1}v\},$$

$$K_n(A, w) = \text{span}\{w, Aw, \ldots, (A)^{n-1}w\}.$$

The following algorithm computes exact inertia of a non-symmetric matrix A, use transforming 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 v_1 and w_1 such that $(v_1, w_1) = 1/\beta$;

set $b_1 = p_0 = q_0 = 0$.

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

$a_j = \beta(Aw_j, p_j)$

$r_{j+1} = Aw_j - a_jw_j - b_jw_{j-1}$

$s_{j+1} = A^Tv_j - r_jv_j - d_jv_{j-1}$

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

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

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

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

End for

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

$z_i = b_i^2$

End for

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

$(\pi, \nu, \delta) = \text{inertia}(a, z, \tau)$

End.

Example 4.1. Let A be the same matrix that used in Example 3.2 and we increase its dimension orderly. We apply Algorithm 2 to find the exact inertia of A. The results for different values of n are shown in Table 2.

Table 2. Implementation of weighted shifted Lanczos method for determining $\text{In}(A)$ with different values of n

n	$\|V_n^T A W_n - T_n\|$	Shift interval (τ)	$\text{In}_0(A)$	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	$2.11E-009$	[– 18, 18]	(14, 18, 0)	exact	0.0077
64	0.0019	[– 31.5, 31.5]	(30, 34, 0)	exact	0.0269
128	14.3	[– 39, 39]	(62, 62, 0)	exact	0.0927
256	34.72	[– 50, 50]	(126, 130, 0)	exact	0.3916
512	81.116	[– 65, 65]	(254, 258, 0)	exact	2.3921

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

Example 4.2. According to the results in Tables 1 and 2, we see that the weighted shifted Lanczos in comparison with shifted Lanczos method works better. Now, consider A is the same matrix that used in Example 3.2. We apply our two computational methods 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.

Table 3. Implementation of shifted Lanczos and weighted shifted Lanczos methods for large values of n

n	Shifted Lanczos method			Weighted shifted Lanczos method		
	$\mathrm{In}_0(A)$	Situation	Time	$\mathrm{In}_0(A)$	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	17.028
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

5. Comments and Conclusion

(1) Two new iterative methods presented in this paper can compute $\mathrm{In}(A)$ in the case that A is a non-symmetric large sparse matrix.

(2) However, the shifted Lanczos method may not be able to compute the exact inertia of a non-symmetric 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 behaviour of the eigenvalues.

References

[1] A. C. Antoulas and D. C. Sorensen, Lyapunov, Lanczos and inertia, Linear Algebra Appl. 326 (2001), 137-150.

[2] D. Carlson and B. N. Datta, The Lyapunov matrix equation $SA + A^*S = S^*B^*BS$, Linear Algebra Appl. 28 (1979a), 43-52.

[3] D. Carlson and B. N. Datta, On the effective computation of the inertia of a non-Hermitian matrix, Numer. Math. 33 (1979b), 315-322.

[4] B. N. Datta, Stability and inertia, Linear Algebra Appl. 302-303 (2000), 563-600.

[5] K. V. Fernando, Computation of exact inertia and inclusions of eigenvalues of tridiagonal matrices, Linear Algebra Appl. 422 (2007), 71-99.

■