World Applied Sciences Journal 13 (7): 1764-1768, 2011
ISSN 1818-4952

New Method for Computing the Inertia of Symmetric Matrix Without Computing the Eigenvalues

[1]H. Nikfarjam, [2]M. Kardel and [3]M. Ghamgosar

[1,2]Department of Mathematics, Islamic Azad University, Zabol Branch, Zabol, Iran
[3]Environmental Research, Institute of ACECR, Iran

Abstract: If A is a symmetric matrix (complex Hermitian) then the Sylvester law of inertia provides us with diagonal pivoting factorization for compute the inertia of A. This factorization requires $n^3/6$ flops,when A is a large and sparse matrix, this factorization is not useful [1-3]. In this paper we develop an algorithm based on Krylov subspace method for computing the exact inertia of a real symmetric (complex Hermitian) matrix without computing the eigenvalues which requires only m^2n flops. The implementation of the final algorithm has been tested by numerical examples, the results show that the algorithm converges fast and works accurately.

Key words: Krylov subspace · Exact inertia · Symmetric · Tridiagonal form

INTRODUCTION

Inertia of a complex Hermitian or a real symmetric matrix is defined as the number of positive, negative and zero eigenvalues of the matrix. It is well-known that the system of differential equations $\dot{x}(t) = Ax(t)$ is asymptotically stable (that is, $x(t) \to 0$ *as* $t \to \infty$) if and only if all eigenvalues of A have negative real parts. Determination of the stability of second-order differential equation arising in vibration and structural analysis is one of the most important issues in engineering. K.V. Fernando describe an algorithm in floating point arithmetic to compute the exact inertia of a real symmetric tridiagonal matrix [4]. There are reliable algorithm to transform real symmetric matrices and complex Hermitian matrices to the real symmetric tridiagonal format. Our main task in this paper is using Arnoldi, weighted Arnoldi and block Arnoldi methods to develop an efficient algorithm for computing the inertia of symmetric matrices, not necessarily tridiagonal matrix.

Definition

Definition 2.1: An equilibrium solution of the system $\dot{x}(t) = Ax(t), x(0) = x_0$ is the vector x_e satisfying:

$$Ax_e = 0$$

Clearly $x_e = 0$ is an equilibrium solution and it is the unique equilibrium if and only if A is nonsingular.

Definition 2.2: An equilibrium solution x_e is said to be stable, if for every $\varepsilon > 0$, there exist a real number $\delta > 0$, such that $\|x(t) - x_e\| < \varepsilon$ whenever $\|x_0 - x_e\| < \delta$.

Definition 2.3: An equilibrium solution x_e is asymptotically stable if it is stable and there exist a $\delta > 0$ such that $\|x(t) - x_e\| \to 0$ as $t \to \infty$, whenever $\|x_0 - x_e\| < \delta$.

Definition 2.4: The system $\dot{x}(t) = Ax(t), x(0) = x_0$ is asymptotically stable if the equilibrium solution $x_e = 0$ is asymptotically stable.

Definition 2.5: The inertia of a matrix order n, denoted by $In(A)$, is a 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 $In(A) = (0, n, 0)$.

Theorems

A necessary and sufficient condition for the equilibrium solution $x_e = 0$ of the homogeneous system $\dot{x}(t) = Ax(t), x(0) = x_0$ to be stable is that all the eigenvalues of the matrix A have negative real parts Proof in [5].

(The Sylvester Law of Inertia): Let A be a Hermitian matrix and P be a nonsingular matrix. Then $In(A) = In(PAP^*)$ Proof in [6].

Corresponding Author: H. Nikfarjam, Department of Mathematics, Islamic Azad University, Zabol Branch, Zabol, Iran. E-mail: nikfarjamt@yahoo.com.

Remark 3.1: Using the Sylvester law of inertia, the inertia of a given Hermitian matrix A can be computed in terms of the diagonal matrix D associated with its triangular factorization $A = LDL^*$, where L is a nonsingular lower triangular matrix and D is diagonal matrix with p positive, q negative and r zero diagonal entries ($p + q + r = n$). Then by the Sylvester law of inertia,

$$In(A) = (p,q,r)$$

Inertia of a symmetric tridiagonal matrix:

Let

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix}$$

And

$z_i = \beta_i^2 \quad (i = 1,2,..., n-1)$

We make the UDU^T factorization of a symmetric triangular matrix T. We use v to denote the number of negative diagonal elements of D, which according to the Sylvester inertia theorem gives the number of negative eigenvalues of T. Similarly π is the number of positive elements of D and it indicates the number of positive eigenvalues of T. A simple algorithm exists for computing the diagonal elements of the matrix D. [7, 8].

Lemma 4.1: The diagonal elements of diagonal matrix D in UDU^T factorization of T are given by:

$$d_i = \alpha_i \quad for\ i = 1 \qquad (4.1)$$
$$d_i = \alpha_i - \frac{z_i - 1}{d_i - 1} \ for\ i = 2,3,...,n$$

K.V. Fernando by using this lemma and floating point arithmetic describe the following algorithm to compute the exact inertia of a real symmetric tridiagonal matrix [4].

Inertia of a Symmetric Matrix: The Arnoldi process is an orthogonal projection method in a Krylov subspace. This method also uses for non symmetric matrices; the result is a Hessenberg matrix. If the matrix is symmetric then we obtain a tri-diagonal matrix. In general we have three Arnoldi methods; they are Arnoldi, weighted Arnoldi and block Arnoldi. From combination of these algorithms by algorithm 1 we can obtain new algorithms for inertia problem.

Algorithm 1 (*inertia of asymmetric matrix*)
choose a vector v
$\tilde{v}_1 = v / \| v \|_D$
choosa a scalar τ (*shift*)
for j = 1,...,*n*
$w = \tilde{v}_j$
for I = 1,...,*j*
$\tilde{h}_{i,j} = (w, \tilde{v}_i)_D$
$w = w - \tilde{h}_{i,j}\tilde{v}_i$
end{*for*}
$\tilde{h}_{j+1,j} = \| w \|_D$ *if* $\tilde{h}_{j+1,j} = 0$ *stop*
$\tilde{v}_{j+1} = w / \tilde{h}_{j+1,j}$
end{*for*}
for $i = 1,...,n$
$\alpha_i = \tilde{h}_{i,j}$ *and* $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$
fot $i = 1,...,n-1$
$z_i = \tilde{h}_{i,i+1}^2$ *and* $z = (z_1, z_2, ..., z_{n-1})$
$(v, \pi, \delta) =$ inertia (α, z, τ)

Remark 5.2: Let the matrix $\tilde{H}_m \in \mathbb{R}^{m \times m}$ be the Hessenberg matrix whose nonzero entries as the scalars $\tilde{h}_{i,j}$ constructed by the weighted Arnoldi process. Let us define the matrix $\bar{\tilde{H}}_m \in \mathbb{R}^{(m+1) \times m}$ by

$$\bar{\tilde{H}}_m = \begin{pmatrix} \tilde{H}_m \\ \tilde{h}_{m+1,m} e_m^T \end{pmatrix}$$

It is known that the matrices built by the Arnoldi process satisfy the following relations

$\tilde{V}_m^T D \tilde{V}_m = I_m, A\tilde{V}_m = \tilde{V}_{m+1}\bar{\tilde{H}}_m$ and $\bar{\tilde{H}}_m = \tilde{V}_m^T D A \tilde{V}_m$ [9].

Numerical Test 1:
At first let A be a 10 × 10 matrix as:

$$A = \begin{bmatrix} -1 & -.21 & -1.2 & 1.5 & -.13 & -1.42 & & & & \\ -.21 & -1 & -.21 & -1.2 & 1.5 & -.13 & -1.42 & & & \\ -1.2 & -.21 & -1 & -.21 & -1.2 & 1.5 & -.13 & -1.42 & & \\ 1.5 & -1.2 & -.21 & -1 & -.21 & -1.2 & 1.5 & -.13 & -1.42 & \\ -.13 & 1.5 & -1.2 & -.21 & -1 & -.21 & -1.2 & 1.5 & -.13 & -1.42 \\ -1.42 & -.13 & 1.5 & -1.2 & -.21 & -1 & -.21 & -1.2 & 1.5 & -.13 \\ & -1.42 & -.13 & 1.5 & -1.2 & -.21 & -1 & -.21 & -1.2 & 1.5 \\ & & -1.42 & -.13 & 1.5 & -1.2 & -.21 & -1 & -.21 & -1.2 \\ & & & -1.42 & -.13 & 1.5 & -1.2 & -.21 & -1 & -.21 \\ & & & & -1.42 & -.13 & 1.5 & -1.2 & -.21 & -1 \end{bmatrix}_{10 \times 10}$$

Table 1: Shows implementation of algorithm 3 with different value of n

n	Error	Shift interval (τ)	$In_0(A)$	situation	time
10	3.9792	[0,3.8]	(8,2,0)	exact	0.001231
16	4.4023	[0,1.9]	(12,4,0)	exact	0.002358
32	4.4863	[0,1.14]	(24,8,0)	exact	0.009025
64	4.564	[0,. 04]	(45,19,0)	exact	0.035277
128	4.7443	[0,.02]	(90,38,0)	exact	0.150856
256	4.7743	[10e-13,.02]	(178,78,0)	fail	0.771581

We apply algorithm 1 to find the inertia of A and then 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.

As the results show although by increasing the dimension of the matrix the error also increases. The other good point in this algorithm is that when $\tau = 0$ then $In\ (A)$ can be computed very accurately. Note that when n=256 then $\tau = 10E - 13$. Recall that for any τ belong in shift interval the value of $In(A)$ can be computed, but the most important point is that when $\tau = 0$, $In\ (A)$ must be computed (zero is in the shift interval).

Now we modify the algorithm 1 in the way that when n is large, works accurate. Our idea is in Arnoldi method instead of using an initial vector we use a block of vectors, in other words a matrix. In this way the error of similarization decreases. We must also use Arnoldi or Weighted Arnoldi methods in new algorithm to have a tri-diagonal form.

Algorithm 2: (block krylov subspace method)

Choose an unitary matrix V_1 *of dimension* $n \times r$
for $j = 1,..., m$ *do*
for $i = 1,...,j$ *do*
$H_{i,j} = V^T_i AV_j$

$$W_j = AV_j - \sum_{k=1}^{j} V_k H_{k,j}$$

compute the QR decomposition $W_j = V_{j+1}H_{j+1,j}$
end do
end do
choose a vector v
$\tilde{v}_1 = v/\|v\|_D$
choose a scalar τ (*shift*)
for $j = 1,..., n$
$w = H\tilde{v}_j$
for $i = 1,...,j$

$$\tilde{h}_{i,j} = (w.\tilde{v}_i)_D$$

$$w = w - \tilde{h}_{i,j}\tilde{v}_i$$

end {*for*}

$$\tilde{h}_{j+1,j} = \|w\|_D \ \ if \ \ \tilde{h}_{j+1,j} = 0 \ stop$$

$$\tilde{v}_{j+1} = w/\tilde{h}_{j+1,j}$$

end {*for*}
for $i = 1,..., n$

$$a_i = \tilde{h}_{i,i} \ and \ a = (a_1, a_2,..., a_n)$$

for $i = 1,...,n-1$

$$z_i = \tilde{h}^2_{i,i+1} \ and \ z = (z_1, z_2,..., a_{n-1})$$

(v, π, δ) = inertia (α, z, τ)
end

Remark 5.3: After implementation of the above block algorithm we have:

$U_m = [V_1, V_2,..., V_m]$
$H_m = (H_{i,j})$
E_m = *matrix of the last r columns of* $I_{n,r}$

$$AU_m = U_m H_m + V_{m+1}H_{m+1,m}\ E^T_m \ (see[9])$$

There are two parameters in this algorithm that they have important role in accuracy and speed of this algorithm for computing the inertia of matrix.

Numerical Test 2: In this test we set n=512, which is the dimension of A, and use block Krylov subspace method to compute the value of In(A). The results are shown in Table 2.

Table 2: Shows implementation of algorithm 2 for n=512 with different value of *r*, *m*

r	m	error	Shift interval (τ)	$In_0(A)$	situation	time
1	512	246.57	[.11,.15]	(336,176,0)	fail	6.878495893
2	256	83.644	[.09,.095]	(344,168,0)	fail	4.082044034
4	128	20.791	[.023,.025]	(354,158,0)	fail	3.107915265
8	64	1.177	[10E-04,3.0E-3]	(355,157,0)	fail	2.685232904
16	32	6.27E-07	[10E-09,3.0E-3]	(356,156,0)	fail	2.581601718
32	16	4.52E-09	[10E-12,3.0e-3]	(356,156,0)	fail	2.567660821
128	4	1.30E-09	[10E-13,3.0e-3]	(357,155,0)	fail	2.522569521
256	2	3.67E-09	[0,3.00e-3]	(358,154,0)	exact	2.501321558

Table 3: Shows implementation of algorithm 1 and block Krylov subspace method for different value of *n*

	Algorithm1					Block Krylov Subspace Method				
n	Error	Shift interval(τ)	$In_0(A)$	situation	time	Error	Shift interval (τ)	$In_0(A)$	situation	time
10	3.9792	[0,3.8]	(8,2,0)	exact	0.0012311	5.26E-13	[0,3.8]	(8,2,0)	exact	0.0004675
16	4.4023	[0,1.9]	(12,4,0)	exact	0.0023582	1.20E-12	[0,1.9]	(12,4,0)	exact	0.0016276
32	4.4863	[0,1.14]	(24,8,0)	exact	0.0090254	8.95E-11	[0,1.14]	(24,8,0)	exact	0.0056336
64	4.564	[0,.04]	(45,19,0)	exact	0.0352767	4.34E-11	[0,.04]	(45,19,0)	exact	0.0151446
128	4.7443	[0,.02]	(90,38,0)	exact	0.1508556	9.41E-10	[0,.02]	(90,38,0)	exact	0.0637092
256	4.7743	[10e-13,.02]	(178,78,0)	fail	0.7715805	1.21E-10	[0,.02]	(179,77,0)	exact	0.3424205
512	4.7933	[10e-12,.003]	(356,158,0)	fail	5.3165374	3.67E-09	[0,.003]	(358,156,0)	exact	2.5013216
1024	4.8505	[10e-11,.00001]	(715,309,0)	fail	38.603636	1.84E-08	[0,.00001]	(717,307,0)	exact	18.726558

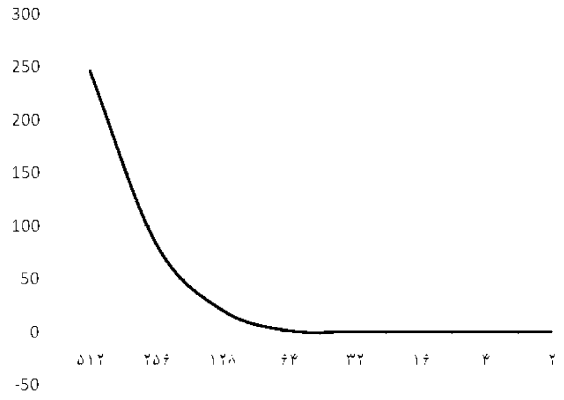

Fig. 1: Shows the decreasing of orthogonalization error when *m* decreases and n=512

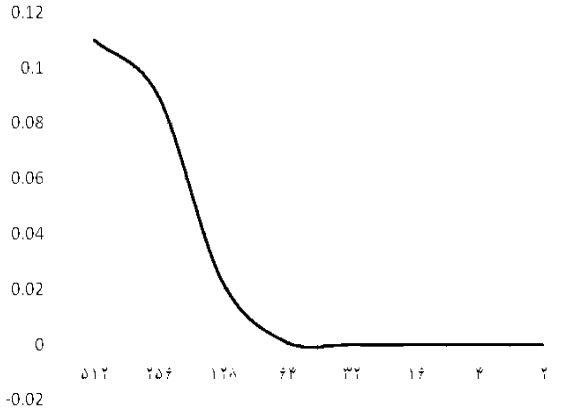

Fig. 2: Shows the decreasing of minimum shift for computing *In(A)* when *m* decreases

As the results show when m decreases or r increases the error decreases and minimum shift for computing the exact value of *In(A)* tends to zero. (Fig. 1 and Fig 2). Thus for computing $In(A_{512\times512})$ by algorithm 4 it is sufficient to have m=2 and r=256. In table 4 the results show that when higher dimensions used the model works well.

Numerical Test 3: Let A is the same matrix that used in numerical test 1 and we increase its dimension orderly. We apply algorithm 1 and block Krylov subspace method to find the exact inertia of *A* with different value of n. the result has been shown in Table 3.

CONCLUSIONS

As the results show algorithm 1 for large dimensions takes a lot of time to do the job, but block Krylov subspace method works fast and very accurate. Not that in algorithm 2 we select m=2 and r=n/2 for any value of n. for example when n=1024 for computing the inertia of A with algorithm 4, it is sufficient m=2 and r=512 and it is a computation remarkable point in this algorithm. Since weighted Arnoldi process requires $2mN_{nz}+\frac{5}{2}m^2n$ flops and block Arnoldi process requires $2mN_{nz} + 2m^2n$ that N_{nz} is the number of nonzero elements of the matrix *A*, thus the total number of operations for block Krylov subspace

method is approximately $8N_{nz}+18n$ that with comparison diagonal pivoting factorization, block Krylov subspace method is a robust algorithm for computing the inertia of a large and sparse symmetric matrices.

REFERENCES

1. Bunch, J.R., 1971. Analysis of the diagonal pivoting method, SIAM J. Numer. Anal., 8 (1971) 656-680.
2. Bunch, J.R. and L. Kaufman, 1977. Some stable method for inertia calculating and solving symmetric linear system, Math. Comp., 31: 162-179.
3. Bunch, J.R., L. Kaufman and B. Parlett, 1976. Decomposition of a symmetric matrix, Numer. Math., 27: 95-109.
4. Fernando, K.V., 2007. Computing of inertia and inclusions of eigenvalues of tridiagonal matrices, Linear Algebra and Its Applications, 422: 77-97.
5. Datta, B.N., 2000. Stability and inertia, Linear Algebra and Its Applications, 302-303: 563-600.
6. Cain, B.E., 1980. Inertia theory, Linear Algebra Appl., 30: 211-240.
7. Fernando, K.V., 1996. Computing an eigenvector of a tridiagonal when the eigenvalue is known, Z.Angew, Math. Mech., 76(Suppl,1): 299-302.
8. Fernando, K.V., 1998. Accurately counting singular values of bidiagonal matrices and eigenvalues of skew-symmetric tridiagonal matrices, SIAM J.Matrix Anal, APPL., 20: 373-399.
9. Y.SAAD, 1981. Krylov subspace method for solving large unsymmetric linear system, Math. Comput., 37: 105-126.