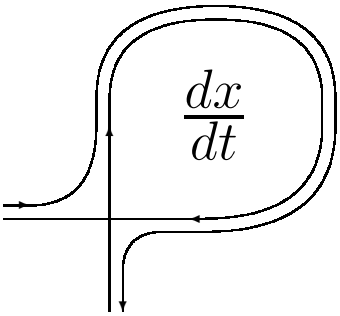

DIFFERENTIAL EQUATIONS
AND
CONTROL PROCESSES
N 2, 2011
Electronic Journal,
reg. N ΦC77-39410 at 15.04.2010
ISSN 1817-2172

http://www.math.spbu.ru/diffjournal
e-mail: jodiff@mail.ru

New Numerical Method for Analyzing BIBO Stability of the Continuous Time Linear System

H.S.Najafi and A.H.Refahi
Department of Mathematics,Faculty of Sciences
University of Guilan,P.O.Box 1841,Rasht,Iran
hnajafi@guilan.ac.ir,ah_refahi@yahoo.com

Abstract

In this paper we prove a new theorem for computing inertia of symmetric large sparse matrix. Based on this theorem, we develop an robust algorithm for determining the inertia and BIBO stability of a continuous-time linear system. The most important property of these methods is determining the inertia and BIBO stability without computing the poles of transfer function.

Mathematics Subject Classification:65F30; 65F50

Keywords: Krylov subspace, stability, exact inertia, symmetric, tridiagonal form.

1 Introduction

A classical approach to determine the stability and inertia is to find the characteristic polynomial of the of the state matrix A followed by application of the Routh-Hurwitz criterion in the continuous-time case and the Schur-Cohn criterion in the discrete-time case. These approaches are historically important and were developed at a time when numerically finding the eigenvalues of a matrix was a difficult problem. In view of the above statement, it is clear that the best way to numerically check the stability and inertia is to explicitly compute all the eigenvalues. However, by computing the eigenvalues, one gets more than stability and inertia. Furthermore if the eigenvalues of A are very ill-conditioned the eigenvalue problem may be misleading. 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(see[1-3]). K.V.Fernando describe an algorithm in floating point arithmetic to compute the exact inertia of a real symmetric tridiagonal matrix (see[6]). Our main task in this paper is using Lanczos, weighted Arnoldi and block Arnoldi processes to develop an efficient algorithm for determining the inertia and BIBO stability of symmetric state matrix, not necessarily tridiagonal matrix.

2 Stability of a Continuous-Time System

Definition 2.1 *An equilibrium state of the system*

$$\dot{x}(t) = Ax(t), \qquad x(0) = x_0, \tag{2-1}$$

is the vector x_e satisfying

$$Ax_e = 0.$$

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

Definition 2.2 *An equilibrium state x_e is asymptotically stable if for initial state, the state vector $x(t)$ approaches x_e as time increases.*

The system (2.1) is asymptotically stable if and only if the equilibrium state $x_e = 0$ is asymptotically stable. Thus the system (2.1) is asymptotically stable if and only if $x(t) \to 0$ as $t \to \infty$.

Theorem 2.3 *The system* (2.1) *is asymptotically stable if and only if all the eigenvalues of A have negative real parts.*

Proof. see[5].

Definition 2.4 *A matrix A is called a stable matrix if all of the eigenvalues of A have negative real parts.*

2.1 Bounded-Input Bounded-Output Stability

The continuous-time linear system:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) \end{aligned} \tag{2-2}$$

is said to be bounded-input bounded output ($BIBO$) stable if for any bounded input, the output is also bounded.
The ($BIBO$) stability is governed by the poles of the transfer function $G(s) = C(sI - A)^{-1}B$.

Theorem 2.5 *The system* (2.2) *is ($BIBO$) stable if and only if every pole of $G(s)$ has a negative real part.*

Remark 2.6 *Since every pole of $G(s)$ is also an eigenvalue of A, an asymptotically stable system is also ($BIBO$) stable.However, the converse is not true.*

2.2 Inertia

Definition 2.7 *The inertia of a matrix order n,denoted by $In(A)$,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.*

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

Theorem 2.8 *(*The Sylvester law of inertia*)*
Let A be a Hermitian matrix and P be a nonsingular matrix. Then $In(A) = In(PAP^T)$.

Proof. see[4].

Remark 2.9 *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, $In(A) = (p, q, r)$.*

Let

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

And

$$Z_i = \beta_i^2 \qquad (i = 1, \cdots, n-1)$$

K.V.Fernando makes the UDU^T factorization of a shifted symmetric triangular matrix $T-\tau I$. He uses ν to denote the number of negative diagonal elements of D, which according to the Sylvester inertia theorem gives the number of negative eigenvalues of $T-\tau I$. Similarly, π is the number of positive elements of D and it indicates the number of positive eigenvalues of $T-\tau I$. A simple algorithm exists for computing the diagonal elements of the matrix D. (See[7,8]).

3 Inertia and BIBO Stability of a Continuous-Time System

There are reliable algorithms to transform real symmetric matrices and complex Hermitian matrices to the real symmetric tri-diagonal format. On of the most important methods for doing this work,in the case that matrix is large and sparse, are the Krylov subspace methods

such as Hermitian Lanczos and Block Arnoldi process. From combination of these algorithms by algorithm described in [4] we can obtain new algorithms for determination the inertia of a large sparse matrices without computing the eigenvalues.

Remark 3.1 *Let A is a symmetric matrix of order n and the matrix $H_n \in R^{n\times n}$ be the tri-diagonal matrix whose nonzero entries as the scalars $t_{i,j}$ constructed by the Lanczos process. Let us define the matrix $\bar{H}_n \in R^{(n+1)\times n}$ by*

$$\bar{H}_n = \begin{pmatrix} H_n \\ h_{n+1,n}e_n^T \end{pmatrix}$$

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

$$V_n^T V_n = I_n,$$

$$AV_n = V_{n+1}\bar{H}_n,$$

$$H_n = V_n^T A V_n.$$

Now by using the Sylvester law of inertia (Theorem 2.8), we have

$$In(A) = In(V_n^T A V_n) = In(H_n).$$

Remark3.1 suggests the following algorithm for computing the inertia and determination the BIBO stability of a continuous-time linear system.

Algorithm 1 *(Lanczos Krylov subspace method)*
choose vector v_1 of norm 1 and set $\beta_1 = 0, v_0 = 0$
choose scalar τ (shift parameter)
for $j = 1, \ldots, n$ do
$w_j = Av_j - \beta_j v_{j-1}$
$\alpha_j = (w_j, v_j)$
$w_j = w_j - \alpha_j v_j$
$\beta_{j+1} = \|w_j\|_2$
$v_{j+1} = w_j/\beta_{j+1}$
end for
$\alpha = (\alpha_1, \ldots, \alpha_n)$
for $i = 1, \ldots, n-1$ do
$z_i = \beta_{i+1}^2$ and $z = (z_1, z_2, \ldots, z_{n-1})$
$(\pi, \nu, \delta) = inertia(\alpha, z, \tau)$ see([3])
End

Example 3.2 *Consider the symmetric matrix A as the form:*

Table 1: Shows implementation of Lanczos Krylov subspace method with different value of n

n	$\|V_n^T A V_n - T_n\|$	$In_0(A)$	*Situation*	*BIBO stability*	*Time*
10	3.311	$(0, 10, 0)$	*exact*	*yes*	0.0006
16	3.971	$(0, 16, 0)$	*exact*	*yes*	0.0014
32	4.98	$(1, 31, 0)$	*fail*	*fail*	0.0029
64	8.22	$(2, 62, 0)$	*fail*	*fail*	0.0063
128	13.16	$(6, 122, 0)$	*fail*	*fail*	0.0209
256	13.65	$(9, 247, 0)$	*fail*	*fail*	0.1223
400	14.19	$(13, 387, 0)$	*fail*	*fail*	0.4314
512	14.38	$(13, 499, 0)$	*fail*	*fail*	0.9658

$$A = \begin{pmatrix} -7.33 & -.21 & -1.2 & 1.5 & -.13 & -1.42 & 0 & \dots & \dots & 0 \\ -.21 & -7.33 & -.21 & -1.2 & 1.5 & -.13 & 1.42 & 0 & \ddots & 0 \\ -1.2 & -.21 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\ 1.5 & -1.2 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & -1.42 & 0 \\ -.13 & 1.5 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & -.13 & -1.42 \\ -1.42 & -.13 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 1.5 & -.13 \\ 0 & -1.42 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & -1.2 & 1.5 \\ \vdots & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & -.21 & -1.2 \\ \vdots & \vdots & 0 & -1.42 & -.13 & 1.5 & -1.2 & -.21 & -7.33 & -.21 \\ 0 & 0 & \dots & 0 & -1.42 & -.13 & 1.5 & -1.2 & -.21 & -7.33 \end{pmatrix}_{n\times n}$$

We apply Lanczos Krylov subspace method 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 (τ) the inertia of A will be computed. The best case is when the shifted parameter is zero. In this case the amount of computation is less. Thus is why we have a column called $In_0(A)$ in table 1 to have more information. As the results show for any n the value of $In_0(A)$ is fail. Now we prove the main theorem of the paper.

Theorem 3.3 *Assume that the basis* $V_n = [v_1, ..., v_m]$ *and* $\tilde{V}_n = [\tilde{v}_1, ..., \tilde{v}_m]$ *are constructed by Arnoldi and weighted Arnoldi process respectively, such that*

$$V_n^T V_n = I_n \qquad H_n = V_n^T A V_n \qquad \tilde{V}_n^T D \tilde{V}_n = I_n \qquad \tilde{H}_n = \tilde{V}_n^T D A \tilde{V}_n$$

where

$$AV_n = V_{n+1}\bar{H}_n \qquad A\tilde{V}_n = \tilde{V}_{n+1}\bar{\tilde{H}}_n \tag{3-1}$$

and

$$\bar{H}_n = \begin{pmatrix} H_n \\ h_{n+1,n}e_n^T \end{pmatrix} \qquad \bar{\tilde{H}}_n = \begin{pmatrix} \tilde{H}_n \\ \tilde{h}_{n+1,n}e_n^T \end{pmatrix}.$$

If v_{n+1} *and* $\tilde{v}_{n+1}$ *are linearly dependent then*

$$In(A) = In(H_n) = in(\tilde{H}_n) \quad .$$

Proof. *As* V_j *and* $\tilde{V}_j$ *are two basis of the Krylov subspace* $K_j(A, v)$ *for all* $j \in \{1, 2, ..., m\}$*, we can express* $\tilde{V}_n$ *in terms of* V_n *as*

$$\tilde{V}_n = V_n U_n \tag{3-2}$$

where U_n *is a* $m \times m$ *upper triangular matrix.*

If we multiply (3.2) *on the left by* V_n^T*, we obtain*

$$U_n = V_n^T \tilde{V}_n,$$

and if we multiply (3.2) *on the left by* $\tilde{V}_n^T D$*, we get*

$$U_n^{-1} = \tilde{V}_n^T D V_n.$$

Using the relations (3.1) *and* (3.2)*, we obtain*

$$AV_nU_n = V_{n+1}U_{n+1}\bar{\tilde{H}}_n \qquad V_{n+1}\bar{H}_nU_n = V_{n+1}U_{n+1}\bar{\tilde{H}}_n.$$

As V_{n+1} *is orthonormal, we get*

$$\bar{H}_nU_n = U_{n+1}\bar{\tilde{H}}_n$$

Multiplying the last equality on the left by the matrix U_{n+1}^{-1}*, we obtain*

$$\bar{\tilde{H}}_n = U_{n+1}^{-1}\bar{H}_nU_n. \tag{3-3}$$

Now we denote by $z_{i,j}(1 \le i, j \le n)$ *the entries of the matrix* U_n^{-1}*. We split the matrix* U_{n+1}^{-1} *as*

$$U_{n+1}^{-1} = \begin{pmatrix} U_n^{-1} & z_{n+1} \\ 0 \dots 0 & z_{n+1,n+1} \end{pmatrix} = \begin{pmatrix} \hat{U_{n+1}^{-1}} \\ 0 \dots 0 \qquad z_{n+1,n+1} \end{pmatrix}$$

where $z_{n+1} = (z_{1,n+1}, z_{2,n+1}, \dots, z_{n,n+1})^T$*. So, the matrix* $\hat{U_{n+1}^{-1}}$ *is obtained from the matrix*U_{n+1}^{-1} *by removing its last row.*

From the relation (3.2)*, we can write*

$$\tilde{H}_n = \hat{U_{n+1}^{-1}} \tilde{H}_n U_n,$$

then, we have the relations

$$\begin{aligned} \tilde{H}_n &= \begin{pmatrix} U_n^{-1} & z_{n+1} \end{pmatrix} \begin{pmatrix} H_n \\ h_{n+1,n} e_n^T \end{pmatrix} U_n, \\ &= U_n^{-1} H_n U_n + h_{n+1,n} z_{n+1} e_n^T U_n. \end{aligned}$$

As the matrix U_n is upper triangular, the matrix product $e_n^T U_n$ is equal to $u_{n,n} e_n^T$. Therefore, we obtain

$$\tilde{H}_n = U_n^{-1} H_n U_n + h_{n+1,n} z_{n+1} e_n^T,$$

where $z_{n+1} \in R^n$ is obtained from the column $n+1$ of the matrix U_{n+1}^{-1} by deleting its last component. As v_{n+1} and $\tilde{v}_{n+1}$ are linearly dependent, $z_{n+1} = 0$. Therefor the matrix H_n and $\tilde{H}_n$ are similar and we have $In(A) = In(H_n) = In(\tilde{H}_n)$. ■

Theorem3.3 suggests the following method for computing the inertia and determination the BIBO stability of a continuous-time linear system.

Algorithm 2 *(Weighted Krylov method)*
choose a vector $d = (d_1, ..., d_n)$ and set $D = diag(d)$
choose a vector v and set $\hat{v}_1 = v/\|v\|_D$
choose scalar τ (shift parametr)
for $j = 1, \ldots, n$ do
$w = A\hat{v}_j$
for $i = 1, \ldots, j$ do
$\hat{h}_{i,j} = (w, \hat{v}_i)_D$
$w = w - \hat{h}_{i,j}\hat{v}_i$
endfor
$\hat{h}_{j+1,j} = \|w\|_D$, if $\hat{h}_{j+1,j} = 0$ stop
$\hat{v}_{j+1} = w/\hat{h}_{j+1,j}$
end for
for $i = 1, \ldots, n$ do
$\alpha_i = \hat{h}_{i,i}$ and $\alpha = (\alpha_1, \ldots, \alpha_n)$
for $i = 1, \ldots, n-1$ do
$z_i = \hat{h}_{i,i+1}^2$ and $z = (z_1, z_2, \ldots, z_{n-1})$
$(\pi, \nu, \delta) = inertia(\alpha, z, \tau)$ see([3])
End

Example 3.4 *Let A is the same matrix that used in numerical Example* 3.2 *and we increase its dimension orderly. We apply Weighted Krylov method to find the exact inertia of A. The result has been shown in Table* 3.

Table 2: Shows implementation of Weighted Krylov method with different value of n

n	$\|V_n^T A V_n - T_n\|$	$In_0(A)$	$Situation$	$BIBOstability$	$Time$
10	1.66	$(0, 10, 0)$	$exact$	yes	0.0011
16	1.89	$(0, 16, 0)$	$exact$	yes	0.0025
32	2.21	$(0, 32, 0)$	$exact$	yes	0.0094
64	2.90	$(0, 64, 0)$	$exact$	yes	0.0376
128	4.43	$(0, 128, 0)$	$exact$	yes	0.1418
256	5.13	$(0, 256, 0)$	$exact$	yes	0.7419
400	5.45	$(0, 400, 0)$	$exact$	yes	1.598
512	8.89	$(13, 499, 0)$	$fail$	$fail$	2.965

As the results show although by increasing the dimension of the matrix the error also increases but still the result completely better than the results of Lanczos Krylov subspace method. The other good point in Weighted Krylov subspace method 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 3 in the way that when n is large, works accurate. Our scheme for doing this work is to use Block Arnoldi process instead of Arnoldi process. In this way the error of similarization decreases. We must also use Arnoldi or Weighted Arnoldi process in new algorithm to have a tri-diagonal form.

Algorithm 3 (*Block Krylov method*)
Choose an unitary matrix V_1 of dimension $n \times r$
for $j = 1, \ldots, m$ do
for $i = 1, \ldots, j$ do
$H_{i,j} = V_i^T A V_j$
$W_j = AV_j - \sum_{k=1}^{j} V_k H_{k,j}$
Compute the QR decomposition $W_j = V_{j+1} H_{j+1,j}$ end for end for.
choose a vector v and scalar τ(shift parameter) and set $\hat{v_1} = \frac{v}{\|v\|_D}$
for $j = 1, \ldots, n$ do
$w = H\hat{v_j}$
for $i = 1, \ldots, j$ do
$\hat{h}_{i,j} = (w.\hat{v_i})_D \quad w = w - \hat{h}_{i,j}\hat{v_i}$ end for
$\hat{h}_{j+1,j} = \|w\|_D$ if $\hat{h}_{j+1,j} = 0$ stop
$\hat{v}_{j+1} = \frac{w}{\hat{h}_{j+1,j}}$ end for
for $i = 1, \ldots, n$ do
$\alpha_i = \hat{h}_{i,i}$ and $\alpha = (\alpha_1, \ldots, \alpha_n)$
for $i = 1, \ldots, n-1$ do
$z_i = \hat{h}_{i,i+1}^2$ and $z = (z_1, z_2, \ldots, z_{n-1})$
$(\pi, \nu, \delta) = inertia(\alpha, z, \tau)$ see([3])
End

Example 3.5 *In this test we set* $n = 512$*, which is the dimension of* A*, and use Block*

Table 3: Shows implementation of Block Krylov subspace method for $n = 512$ with different value of r, m

r	m	$error$	$In_0(A)$	$Situation$	$BIBO \quad stability$	$Time$
1	512	94.43	$(10, 500, 2)$	$fail$	$fail$	6.8784
2	256	46.91	$(7, 503, 2)$	$fail$	$fail$	4.0820
4	128	13.11	$(3, 508, 1)$	$fail$	$fail$	3.1079
8	64	0.09	$(2, 510, 0)$	$fail$	$fail$	2.6852
16	32	5.32E−06	$(0, 512, 0)$	$exact$	yes	2.5816
32	16	3.12E−08	$(0, 512, 0)$	$exact$	yes	2.5676
64	8	4.52E−11	$(0, 512, 0)$	$exact$	yes	2.5408
128	4	1.30E−11	$(0, 512, 0)$	$exact$	yes	2.5225
256	2	3.67E−12	$(0, 512, 0)$	$exact$	yes	2.5013

Table 4: Implementation of Block Krylov and Weighted Block Krylov methods for large values of n

n	Block Krylov method			Weighted Block Krylov method			
	$In_0(A)$	situation	Time	$In_0(A)$	situation	Time	$BIBO \quad stability$
512	$(0, 512, 0)$	$exact$	2.50	$(12, 4, 0)$	$exact$	2.11	yes
800	$(0, 800, 0)$	$exact$	9.34	$(0, 800, 0)$	$exact$	8.358	yes
1000	$(0, 1000, 0)$	$exact$	23.37	$(0, 1000, 0)$	$exact$	19.51	yes
1200	$(0, 1200, 0)$	$exact$	39.09	$(0, 1200, 0)$	$exact$	31.862	yes
1400	$(11, 1385, 4)$	$fail$	54.62	$(0, 1400, 0)$	$exact$	44.286	yes
1800	$(19, 1778, 3)$	$fail$	82.19	$(0, 1800, 0)$	$exact$	66.342	yes
2000	$(23, 1976, 1)$	$fail$	101.33	$(0, 2000, 0)$	$exact$	81.654	yes

Krylov method to compute $In(A)$. The results are shown in Table 3.

As the results show when m decreases or r increases the error decreases. 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.

Example 3.6 *Let A is the same matrix that used in Example* 3.2 *and we increase its dimension orderly. We apply Block Krylov and Weighted Block Krylov methods to find the exact inertia and determination BIBO stability of A with different value of n. the result has been shown in Table* 4.

4 COMMENTS AND CONCLUSION

1. Two new methods presented in this paper can compute $In_(A)$ in the case that A is a symmetric large sparse matrix. Mean while, other methods such as diagonal pivoting factorization do not have this capability.

2. As the results show algorithm 2 for large dimensions takes a lot of time to do the job,

but Weighted Block Krylov method works fast and very accurate. Not that in algorithm 3 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 3, it is sufficient $m = 2$ and $r = 512$ and it is a computation remarkable point in this algorithm. Since weighted Arnoldi process requires $2mN_{nz}+5/2m^2n$ 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 method,that requires $n^3/6$ flops, Block Krylov subspace method is a robust algorithm for computing the inertia of a large and sparse symmetric matrices.

Acknowledgment

Authors would like to thank the referees for hosting this paper.

References

[1] J.R.Bunch, *Analysis of the diagonal pivoting method*, SIAM J.Numer.Anal.8 (1971) 656-680.

[2] J.R.Bunch, L.Kaufman, *Some stable method for inertia calculating and solving symmetric linear system*, Math.Comp.31 (1977) 162-179.

[3] J.R.Bunch, L.Kaufman, B.Parlett, *Decomposition of a symmetric matrix*, Numer. Math. 27(1976) 95-109.

[4] B.E.Cain, *Inertia theory*, Linear Algebra Appl. 30 (1980) 211-240.

[5] B.N.Datta, *Stability and inertia*, Linear Algebra and Its Applications, 302-303 (2000) 563-600.

[6] K.V. Fernando, *Computing of inertia and inclusions of eigenvalues of tridiagonal matrices*, Linear Algebra and Its Applications, 422(2007) 77-97.

[7] K.V. Fernando, *Computing an eigenvector of a tridiagonal when the eigenvalue is known*, Z.Angew, Math. Mech, 76 (suppl,1) (1996) 299-302.

[8] K.V. Fernando, *Accurately counting singular values of bidiagonal matrices and eigenvalues of skew-symmetric tridiagonal matrices*, SIAM J.Matrix Anal, APPL,20 (1998) 373-399.

[9] Y.SAAD, *Krylov subspace method for solving large unsymmetric linear system*, Math. Comput,37 (1981) 105-126.