A New Method for assigning ,the eigenvalues sign in equations 
\[ (Ax = \lambda x) \text{ and } (Ax = \lambda Bx) \]

Hashem Saberi Najafi
Guilan University
Computer Center
P. O. Box 1914 , Rasht
IRAN

Maryam Shams Solary
Guilan University
Department of Mathematics , Faculty of Sciences
P. O. Box 1914 , Rasht
IRAN

Abstract: The inertia of an \( n \times n \) complex matrix \( A \), is defined to be an integer triple \( \text{In}(A) = (\pi(A), \nu(A), \delta(A)) \) where \( \pi(A) \) is the number of eigenvalues of \( A \) with positive real parts, \( \nu(A) \) is the number of eigenvalues with negative real parts and \( \delta(A) \) is the number of eigenvalues with zero real parts. We are interested in computing the Inertia for large unsymmetric generalized eigenproblem \( (A,B) \) for equation \( A\varphi = \lambda B\varphi \) Where \( A \) and \( B \) are \( n \times n \) large matrices.

For standard eigenvalues problem let \( B = \text{Identity matrix} \).

An obvious approach for determine Inertia of pair \( (A,B) \), is to transform this to a standard eigenproblem by inverting either \( A \) or \( B \).

In this paper we show that the eigenvalues sign can be computed by assigning the interval that including all the eigenvalues and this method is compared by results in Matlab.

Key–Words: Sign, Inertia, Arnoldi algorithm, Generalized eigenproblem, Shift, Interval, Block matrix

1 Introduction

The problem of finding eigenvalues arises in a wide variety of practical applications. It arises in almost all branches of science and engineering.

We explain Arnoldi algorithm and shift-and-invert Arnoldi algorithm for computing the eigenvalues of equation \( Ax = \lambda x \) \( Ax = \lambda Bx \) and then we compute inertia for them. For computing the Inertia of \( (A,B) \), we gain the first shift by block Hessenberg form of \( A,B \) because we want to have \( M = (A - \sigma B)^{-1} B \).

we will discuss the inverse of a block matrix [7] for computing matrix \( M \) and so we gain

1- The number of eigenvalues of \( (A,B) \) with positive real parts
2- The number of eigenvalues of \( (A,B) \) with negative real parts
3- The number of eigenvalues that may have been zero real parts

by Gereshgorin theorem and end by Arnoldi algorithm gain eigenvalues for third part that we do not know their nature .

We then report several numerical examples and compare this technique with Matlab functions Eig.m and Sptarn. m.

2 Arnoldi Method

This method was developed by Arnoldi [9], in 1951.
Starts with an initial vector and after \( m \) steps produces an \( n \times m \) Hessenberg matrix \( H_m \) and an orthogonal \( n \times m \) matrix \( V_m \) such that

\[ V_m^H AV_m \simeq H_m \quad m \leq n \]

So the eigenvalues of \( H_m \) is an approximation of the eigenvalues of \( A \). One of the strongest points in the Arnoldi method compared to the other methods is that if one needs \( m \) eigenvalues of a \( n \times n \) matrix it is sufficient to use the algorithm \( m \) times, [8,9].

Definition 1 The approximate eigenvalues \( \lambda_i^{(m)} \) provided by the projection process onto \( K_m \) are the eigenvalues of the Hessenberg matrix \( H_m \) is called Ritz eigenvalues . The Ritz approximate eigenvector associated with \( \lambda_i^{(m)} \) is defined by \( u_i^{(m)} = V_m y_i^{(m)} \) where \( y_i^{(m)} \) is an eigenvector associated with the eigenvalue \( \lambda_i^{(m)} \).
3 The shift- and- invert Arnoldi method

If the matrix A or B is invertible for some shift , the eigenproblem
$$A\varphi_i = \lambda_i B \varphi_i$$
can be transformed into the standard eigenproblem
$$A\varphi_i = \lambda_i B \varphi_i \quad (1)$$
$$(A - \sigma B)\varphi_i = \lambda_i B \varphi_i - \sigma B \varphi_i \quad (2)$$
$$(A - \sigma B)\varphi_i = (\lambda_i - \sigma)B \varphi_i \Rightarrow 1/(\lambda_i - \sigma)\varphi_i = \varphi_i \quad (3)$$
$$C\varphi_i = \theta_i \varphi_i \quad (4)$$
Where
$$\theta_i = 1/(\lambda_i - \sigma)$$

It is easy to verify that $(\lambda_i, \varphi_i)$ is an eigenpair of problem (1) if and only if $(\theta_i, \varphi_i)$ is an eigenpair of the matrix C. Therefore, the shift- and- invert Arnoldi method for the eigenproblem (1) is mathematically equivalent to the standard Arnoldi method for the transformed eigenproblem (2).

It starts with a given unit length vector $v_1$ (usually chosen randomly) and builds up an orthonormal basis $V_m$ for the Krylov subspace $K_m(C, v_1)$ by means of the Gram- Schmidt orthogonalization process. In finite precision, reorthogonalization is performed whenever same sever cancellation occurs [4,9].

Then the approximate eigenpairs for the transformed eigenproblem (2) can be extracted from $K_m(C, v_1)$. The approximate solutions for problem (1) can be recovered from these approximate eigenpairs. The shift- and- invert Arnoldi process can be written in matrix form
$$(A - \sigma B)^{-1}BV_m = V_mH_m + h_{m+1,m}v_{m+1}e_m^t \quad (6)$$
$$A - \sigma B)^{-1}BV_m = V_mH_m \quad (7)$$

Where $e_m$ is the $m^{th}$ coordinate vector of dimension $m, V_{m+1} = (V_m, v_{m+1}) = (v_1, v_2, \ldots, v_{m+1})$ is an $n \times (m + 1)$ matrix whose columns form an orthonormal basis of the dimensional Krylov subspace $K_{m+1}(C, v_1)$, and $H_m$ is the $(m + 1) \times m$ upper Hessenberg matrix that is same as $H_m$ except for an additional row in which the only nonzero entry is $h_{m+1,m}$ in the position $(m+1,m)$.

Suppose that$(\theta_i, \tilde{\varphi}_i), \ i = 1, 2, \ldots, m$ are the eigenpairs of the matrix $H_m$.
$$H_m\tilde{\varphi}_i = \tilde{\theta}_i \tilde{\varphi}_i \quad (8)$$

$$\lambda_i = \sigma + 1/\theta_i \text{ and } \tilde{\varphi}_i = v_m\tilde{\varphi}_i \quad (9)$$

and when the shift- and- invert Arnoldi method uses $(\lambda_i, \tilde{\varphi}_i)$ to approximate the eigenpairs $(\lambda_i, \varphi_i)$ of the problem (1) the $\lambda_i$ and $\tilde{\varphi}_i$ are called the Ritz values and the Ritz vectors of A with respect to $Km(C, v_1)$ and we call the upper process Shift-and-Invert Algorithm. For details refer to [2,9].

4 A new method for computing the eigenvalues sign of equation

$$(A\varphi = \lambda B\varphi)$$

In the last section, we described that the shift- and- invert Arnoldi method for the eigenproblem (1) is mathematically equivalent to the standard Arnoldi method for the transformed eigenproblem (2).

We have

$$A\varphi_i = \lambda_i B \varphi_i \rightarrow (A - \sigma B)^{-1}B \varphi_i = \theta_i \varphi_i$$

where $\sigma$ is a shift. In matlab function sptrn that gain eigenvalues for equation (1) select shift $\sigma$ randomly in interval $[Lb, Ub]$ that want to find eigenvalues.

We show a new method for select shift $\sigma$.

We gain first shift with helping $(n,m-1), (n,n)$ entry of matrix $(\tilde{A})^{-1}\tilde{B}$ that $\tilde{A}, \tilde{B}$ are Hessenberg form of matrices A,B and gain with Householder transformations and implicit shift[7,8].

Theorem 2 If $A, D$ were invertible so we have
$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} + A^{-1}B^{-1}CA^{-1} & -A^{-1}B^{-1} \\ -s^{-1}CA^{-1} & s^{-1} \end{pmatrix}$$
$$S = (D - CA^{-1}B)$$

proof: see [7].

We do not need all of entries of matrix$(\tilde{A})^{-1}\tilde{B}$ we decompose matrix $A^{-1} and\tilde{B} 2-\text{block} 2-\text{block}$ matrix that diagonal blocks are $(n - 2, n - 2)$ and $(2, 2)$ and gain alone entries of matrix $2 \times 2$ on the diameter.

$$\text{Inverse}(\text{Hessenberg}(A)) = \text{Inverse}(\tilde{A}) =$$

$$\begin{pmatrix} (A^{-1})_{(2\times 2)} & (B^{-1})_{(2\times 2)} \\ (0 \times (n-2)) & (C^{-1})_{(2\times 2)} \end{pmatrix}^{-1} =$$

$$\begin{pmatrix} A^{-1}_{(n-2)\times (n-2)} & B^{-1}_{(n-2)\times 2} \\ (0 \times 2) & C^{-1}_{2\times 2} \end{pmatrix}$$
Hessenberg(B) = $\hat{B} = \begin{pmatrix} A_2^{(n-2)\times(n-2)} & B_2^{(n-2)\times2} \\ 0_{2\times(n-2)} & C_2^{2\times2} \end{pmatrix}$

$(\hat{A})^{-1}\hat{B} = \begin{pmatrix} A_1^{-(n-2)\times(n-2)} & -A_1^{-(n-2)\times2}B_1C_1^{-1} \\ 0_{2\times(n-2)} & C_1^{2\times2} \end{pmatrix}$

$(A_2^{(n-2)\times(n-2)} & B_2^{(n-2)\times2} \\ 0_{2\times(n-2)} & C_2^{2\times2}) = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$

for gain (n,n)th and (n,n-1)th entry of the current matrix gain $D = C_1^{-1}C_2$ that is a $2 \times 2$ matrix.

It is easy that finding inverse of $2 \times 2$ matrix $C_1$, without losing any generalities of the problem we take

$\sigma = D_{2,2} + D_{2,1}$

as an initial approximation of shift so in the last row of $(\hat{A})^{-1}\hat{B}$ matrix.

According to the deflation process [5,8], $\sigma$ is an approximation of eigenvalue $(A\varphi = \lambda B\varphi)$ and have

$(A - \sigma B)\varphi_i = \theta_i B\varphi_i$

$\theta_i = 1/(\lambda_i - \sigma)$

For computing

$M = (A - \sigma B)^{-1}$

we can use, block matrix method [6].

In this method the matrix divided in $2 - block \times 2 - block$ matrix and by applying LDV decomposition, the inverse is computed. In fact by using this decomposition instead of finding $M^{-1}$ we compute the inverse of LDV which is a much easier and faster process in finding the inverse of M.

Now we want to find a region that all of eigenvalues of $(A - \sigma B)^{-1}B$ situated there.

**Theorem 3** Let $A = (a_{ij})_{n \times n}$. Define

$r_i = \sum_{i=1}^{n} |a_{ij}|, \quad i = 1, \ldots, n$

then each eigenvalue $\lambda$ of A satisfies at least one of the following inequalities:

$|\lambda - a_{ii}| \leq r_i, \quad i = 1, 2, \ldots, n$

In other words, all the eigenvalues of A can be found in the union of disks

$\{z : |z - a_{ii}| \leq r_i, i = 1, 2, \ldots, n\}$

proof: see [8].

With using above theorem we can see that

$r_{pi} = \text{real}(A(i, i)) - r_i$

and

$r_{ni} = \text{real}(A(i, i)) + r_i$

can determine nature and location of the eigenvalues. For example, if $r_{pi} > 0$ for $1 \leq i \leq n$ agreement definition $r_{pi}$ then ith eigenvalue situated in positive part of real numbers.

if $r_{ni} < 0$ for $1 \leq i \leq n$ agreement definition $r_{ni}$ then ith eigenvalue situated in negative part of real numbers.

else eigenvalue may be zero but we do not sure about its nature we must examine more.

5 **Comparisons of numerical tests**

We have tested the results of the above algorithm for different matrix and compare them with results of the functions eig.m and sptarn.m in Matlab.

For this work using Matlab 7.04 on a personal computer PENTIUM 4 with the machine precision $\epsilon \approx 2.22 \times 10^{-16}$.

5.1 **Numerical Experiments**

Algorithms are tested for various matrices using Matlab software, where $\pi(M)_1, \pi(M)_2, \pi(M)_3$ are the number of eigenvalues of $(A,B)$ with positive real parts, for Inertia shift-and-invert algorithm (above algorithm), eig algorithm and sptarn algorithm $\nu(M)_1, \nu(M)_2, \nu(M)_3$ are the number of eigenvalues with negative real parts, for above algorithm, eig algorithm and sptarn algorithm $\delta(M)_1, \delta(M)_2, \delta(M)_3$ are the number of eigenvalues with zero real parts, for above algorithm, eig algorithm and sptarn algorithm $per_1, per_2, per_3$ are percent eigenvalues namely the number of eigenvalues that finds with above algorithms divided to rank matrix multiply 100.

5.2 **Matlab function Sptarn. m**

In this function the Arnoldi algorithm with spectral transformation is used. $[xv, lmb, iresult] = \text{sptarn}(A, B, Lb, Ub)$ finds eigenvalues of the equation $(A - \lambda B)x$ in the interval $[Lb, Ub]$. A,B are $n \times n$ matrices, and $[Lb, Ub]$ are lower and upper bounds for eigenvalues to be sought, for finding all of eigenvalues, we find $[Lb, Ub]$ by helping Gerseshgorin theorem for searching all of eigenvalues.
In the complex case, the real parts of \( \text{Im} \) are compared to \( L_b \) and \( U_b \). \( xv \) are eigenvectors, ordered so that norm \( (A \times xv - B \times xv \times \text{diag}(\text{Im}b)) \) is small. \( \text{Im} \) is the sorted eigenvalues. If \( \text{result} \geq 0 \) the algorithm succeeded and all eigenvalues in the intervals have been found. If \( \text{result} < 0 \) the algorithm is not successful, there may be more eigenvalues, try with a smaller interval. Normally the algorithm stops earlier when enough eigenvalues have converged.

The shift is chosen at a random point in the interval \([L_b, U_b]\) when both bounds are finite. The number of stops of the Arnoldi run depends on how many eigenvalues there are in the interval. After a stop, the algorithm restarts to find more schur vectors in orthogonal complement to all those already found. When no eigenvalues are found in \( L_b \leq \text{Im}b \leq U_b \), the algorithm stops. If it fails again check whether the pencil may be singular.

### Example 1:

(Has been taken from [1] Dielectric channel waveguide problems arise in many integrated circuit applications. Discretization of the governing Helmholtz equation for the magnetic field \( H \)

\[
\nabla^2 H_x + k^2 n^2(x, y) H_x = \beta^2 H_x \\
\n\nabla^2 H_y + k^2 n^2(x, y) H_y = \beta^2 H_y
\]

By finite difference leads to an unsymmetric matrix eigenvalue problem of the form

\[
\begin{pmatrix}
  C_{11} & C_{12} \\
  C_{21} & C_{22}
\end{pmatrix}
\begin{pmatrix}
  H_x \\
  H_y
\end{pmatrix} = \beta^2
\begin{pmatrix}
  B_{11} & B_{12} \\
  B_{21} & B_{22}
\end{pmatrix}
\begin{pmatrix}
  H_x \\
  H_y
\end{pmatrix}
\]

where \( C_{11} \) and \( C_{22} \) are five- or tridiagonal matrices, \( C_{12} \) and \( C_{21} \) are (tri-) diagonal matrices, \( B_{11} \) and \( B_{22} \) are nonsingular diagonal matrices. The problem has been tested for finding sign all of eigenvalues.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( \text{px} )</th>
<th>( \text{px2} )</th>
<th>( \text{px3} )</th>
<th>( \text{px1} )</th>
<th>( \text{px} )</th>
<th>( \text{px2} )</th>
<th>( \text{px3} )</th>
<th>( \text{px1} )</th>
<th>( \text{px} )</th>
<th>( \text{px2} )</th>
<th>( \text{px3} )</th>
<th>( \text{px1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>300</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>400</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>500</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>600</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>700</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>800</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>900</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( n )</th>
<th>( \text{px} )</th>
<th>( \text{px2} )</th>
<th>( \text{px3} )</th>
<th>( \text{px1} )</th>
<th>( \text{px} )</th>
<th>( \text{px2} )</th>
<th>( \text{px3} )</th>
<th>( \text{px1} )</th>
<th>( \text{px} )</th>
<th>( \text{px2} )</th>
<th>( \text{px3} )</th>
<th>( \text{px1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>150</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>250</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>300</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>350</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>400</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>450</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>500</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

we can see algorithm (Inertia Shift-and-Invert) determined sign for all the eigenvalues but function eig.m works only for 6 eigenvalues and sptarn.m function do not give acceptable digits.
Example 2: (Has been taken from gallery.m in Matlab) Clement–Tridiagonal matrix with zero diagonal entries

\[ A = \textit{gallery}('\text{clement}', n, \text{sym}) \]

returns an \( n \times n \) tridiagonal matrix with zeros on its main diagonal and known eigenvalues. It is singular if order \( n \) is odd. About 64 percent of the entries of the inverse are zero.

The eigenvalues include plus and minus the numbers \( n-1, n-3, n-5, \ldots \), as well as (for odd \( n \)) a final eigenvalue of 1 or 0. Argument \( \text{sym} \) determines whether the Clement matrix is symmetric. For \( \text{sym} = 0 \) (the default) the matrix is nonsymmetric, while for \( \text{sym} = 1 \), it is symmetric.

Table 2 shows that Inertia shift-and-Invert Algorithm (above algorithm) and function \textit{eig.m} have results similar for all of eigenvalues but function \textit{sptarn.m} only give a part of sign all of eigenvalues.

Example 3

\[ A = \text{delsq(numgrid('C',30))} \] is a symmetric positive definite matrix of size 632. The Gereshgorin disks for matrix \( A \) show that eigenvalues situated in the interval \([0, 8]\) see[8].

\[ \text{In}(A) = (\pi(A)_1, \nu(A)_1, \delta(A)_1) = (632, 0, 0) \]

by Inertia shift-and-invert algorithm,

\[ \text{In}(A) = (\pi(A)_2, \nu(A)_2, \delta(A)_2) = (6, 0, 0) \]

by function \textit{eig.m} and

\[ \text{In}(A) = (\pi(A)_3, \nu(A)_3, \delta(A)_3) = (52, 0, 0) \]

by function \textit{sptarn.m}.

We see \( \text{per}_1 = 100, \text{per}_2 = 95, \text{per}_3 = 8.23 \) namely Inertia shift-and-invert algorithm (above algorithm) work better than other algorithms.

The results of this examples are plotted as Figures 1-2. The broken lines are the results of function \textit{sptarn.m} and connected lines are the results of Inertia Shift-and-Invert and circle-lines are the results of function \textit{eig.m}.

We can see almost for all the cases Inertia shift-and-Invert Algorithm works better or similar to function \textit{eig.m} but function \textit{sptarn.m} needs very times and do not give acceptable results.

For example 1 (fig 1) we can see \( \text{Time(}\text{Inertia Shift -and -Invert}) = 52.433 \) s and give nature for all the eigenvalues \( \text{Time(}\text{eig.m}) = 29.273 \) s for nature only 6...
6 Conclusion

One of the most topics in the eigenvalues problem is to determine nature and location of the eigenvalues because this matter is many important characteristics of physical and engineering systems, such as stability. So this encouraged us into gaining new method for this problem.

We have already mentioned, solving Inertia for the generalized eigenvalue problems $A\varphi_i = \lambda_i B \varphi_i$ For this problem we select first shift by block Hessenberg form of A, B and use Block shift- and- invert Arnoldi method see [7].

We gain a regions that positive, negative and zero eigenvalues are situated these and compare this technique with Matlab functions sptarn.m and eig.m.

It is observed that Algorithm sign eigenvalues by Shift-and-Invert was much more efficient than Algorithm functions sptarn.m and eig.m for nearly all the cases and this algorithm finds nature almost all the eigenvalues by less time than other Algorithms. Also we see that function sptarn.m fails for some large matrices but new algorithm and function eig.m gives more acceptable results.

References:


