Scientific Research Publishing

Journal of Applied Mathematics and Physics, 2017, 5, 680-692
http://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

An Efficient Algorithm for the Numerical Computation of the Complex Eigenpair of a Matrix

R. O. Akinola*, K. Musa, I. A. Nyam, S. Y. Kutchin, K. V. Joshua

Department of Mathematics, Faculty of Natural Sciences, University of Jos, Jos, Nigeria
Email: *roakinola@gmail.com

How to cite this paper: Akinola, R.O., Musa, K., Nyam, I.A., Kutchin, S.Y. and Joshua, K.V. (2017) An Efficient Algorithm for the Numerical Computation of the Complex Eigenpair of a Matrix. *Journal of Applied Mathematics and Physics*, **5**, 680-692.
https://doi.org/10.4236/jamp.2017.53057

Received: December 28, 2016
Accepted: March 21, 2017
Published: March 24, 2017

Abstract

In computing the desired complex eigenpair of a matrix, we show that by adding Ruhe's normalization to the matrix pencil, we obtain a square nonlinear system of equations. In this work, we show that the corresponding Jacobian is non-singular at the root and that with an appropriately chosen initial guesses, Ruhe's normalization with a fixed complex vector not only converges quadratically but also faster than the earlier Algorithms for the numerical computation of the complex eigenpair of a matrix. The mathematical tools used in this work are Newton and Gauss-Newton's methods.

Keywords

Quadratic Convergence, Newton's Method

1. Introduction

In [1], Akinola and Spence considered the problem of computing the eigenpair $(\boldsymbol{x},\lambda)$ from the generalized complex eigenvalue problem:

$$\boldsymbol{Dx} = \lambda \boldsymbol{Px}, \tag{1}$$

where $\boldsymbol{x} \in \mathcal{C}^n, \boldsymbol{x} \neq 0, \lambda \in \mathcal{C}, \boldsymbol{D}$ is a large real $n \times n$ non-symmetric matrix and $\boldsymbol{P}$ a real symmetric positive definite matrix. After adding the normalization [2]

$$\boldsymbol{x}^H \boldsymbol{Px} = 1, \tag{2}$$

to (1), they obtained a combined system of equations of the form $F(\boldsymbol{u}) = 0$, where $\boldsymbol{u} = [\boldsymbol{x}^H, \lambda]$, given as

$$F(\boldsymbol{u}) = \begin{bmatrix} (\boldsymbol{D} - \lambda \boldsymbol{P})\boldsymbol{x} \\ -\frac{1}{2}x^H \boldsymbol{Px} + \frac{1}{2} \end{bmatrix} = 0. \tag{3}$$

In trying to solve the nonlinear system (3), two drawbacks were encountered. The first one is that if $\boldsymbol{x}$ from $(\boldsymbol{x},\lambda)$ solves (3), then so does $\exp^{i\theta}\boldsymbol{x}$ for any $\theta\in[0,2\pi)$, which means that $\boldsymbol{x}$ has no unique solution. Secondly, $\overline{\boldsymbol{x}}$ in $\boldsymbol{x}^H=\overline{\boldsymbol{x}}^{\mathrm{T}}$ is not differentiable since $\overline{\boldsymbol{x}}$ does not satisfy the Cauchy-Riemann [3] equations which implies that (3) cannot be differentiated and the standard Newton's method cannot be applied. The author then proposed that the above drawbacks could be overcome at least for the $\boldsymbol{P}=I$ case. Before the works of Akinola, Ruhe [4] and Tisseur [5] added the differentiable normalizations

$$\boldsymbol{c}^H\boldsymbol{x}=1, \tag{4}$$

and

$$\tau\boldsymbol{e}_s^{\mathrm{T}}\boldsymbol{x}=\tau, \tag{5}$$

where $\boldsymbol{c}$ is a fixed complex vector and $\tau=\max\left(\|\boldsymbol{D}\|,\|\boldsymbol{P}\|\right)$ for some fixed *s*. Adding each of the two normalization to (1), Ruhe and Tisseur then obtained the following combined system of equations;

$$F(\boldsymbol{u})=\begin{bmatrix}(\boldsymbol{D}-\lambda\boldsymbol{P})\boldsymbol{x}\\ \boldsymbol{c}^H\boldsymbol{x}-1\end{bmatrix}=0, \tag{6}$$

and

$$F(\boldsymbol{u})=\begin{bmatrix}(\boldsymbol{D}-\lambda\boldsymbol{P})\boldsymbol{x}\\ \tau\boldsymbol{e}_s^{\mathrm{T}}\boldsymbol{x}-\tau\end{bmatrix}=0, \tag{7}$$

which have the corresponding Jacobians

$$F_u(\boldsymbol{u})=\begin{bmatrix}(\boldsymbol{D}-\lambda\boldsymbol{P}) & -\boldsymbol{P}\boldsymbol{x}\\ \boldsymbol{c}^H & 0\end{bmatrix}, \tag{8}$$

and

$$F_u(\boldsymbol{u})=\begin{bmatrix}(\boldsymbol{D}-\lambda\boldsymbol{P}) & -\boldsymbol{P}\boldsymbol{x}\\ \tau\boldsymbol{e}_s^{\mathrm{T}} & 0\end{bmatrix}. \tag{9}$$

In this paper, we show that the square Jacobian given by (8) is nonsingular at the root using the ABCD lemma if the eigenvalue of interest is algebraically simple. The major distinction between the two-norm normalization and Ruhe's normalization is that the two-norm normalization is a natural normalization which makes the choice of $\boldsymbol{c}$ free. The Jacobian (9) above was shown to be singular in [5] at the root if and only if λ^* is a finite multiple eigenvalue of the pencil $(\boldsymbol{D},\boldsymbol{P})$.

In this paper, we compare the numerical performance of the algorithm (Algorithm 1) based on Ruhe's normalization (*i.e.,* an application of Newton's method on (6) using the Jacobian (8)) with previous algorithms developed by Akinola *et al.,* in [1] [6] and [7]. All three algorithms: Algorithm 2 as discussed in [1], Algorithm 3 as described in [6], Algorithm 4 as presented in [7] were based on the natural two-norm normalization for the eigenvector. We show that with the same starting guesses, and a carefully chosen fixed complex vector $\boldsymbol{c}$ that the algorithm based on Ruhe's normalization converges faster than the other three.

The plan of this paper is as follows: in Section 2, we used Keller's ABCD Lemma [8] to show that the Jacobian (8) is nonsingular at the root in Theorem 2.1 and we present the four Algorithms. In Section 3, we compare the performance of the four algorithms on three numerical examples. Eigenvalues are used in differential equations in studying stability and in complex biological systems in determining eigenvector centrality (see also [9] [10]).

2. Methodology

In this section, we proof the main result in this paper which states the condition under which the Jacobian matrix (8) (for $\boldsymbol{P}=\boldsymbol{I}$) is non-singular at the root, that is $\boldsymbol{x}^*=\left(\boldsymbol{x}^*,\lambda^*\right)$. This is then followed by a presentation of Algorithm 1, which is actually Newton's method for solving (6). The remaining algorithms have been discussed extensively in [1] [6] and [7]. For the sake of avoiding self plagiarism, we refer the interested reader to those articles.

Algorithm 1 involves solving an $(n+1)$ by $(n+1)$ square system of equations using LU factorisation and does not involve splitting the eigenvalue and eigenvector into real and imaginary parts.

Algorithm 2 involves splitting the eigenpair into real and imaginary parts to obtain an under-determined non linear system of equations. This results in solving a $(2n+1)$ real under-determined linear system of equations for $(2n+2)$ real unknowns using Gauss-Newton method [11]. This is solved using QR factorisation.

Algorithm 3 also involves splitting the eigenpair into real and imaginary parts but with the help of an added equation we obtained a square $(2n+2)$ by $(2n+2)$ system of linear equations which is solved using LU factorisation.

Algorithm 4 is closely related to Algorithm 1 in the sense that both uses complex arithmetic. While Algorithm 1 used a fixed complex vector which does not change throughout the computation, Algorithm 4 uses the natural two-norm normalization which ensures that the eigenvector is updated at each stage of the computation.

Theorem 2.1. *Let* $\left(\boldsymbol{D}-\lambda^*\boldsymbol{I}\right)$ *be an* n *by* n *matrix,* $\boldsymbol{x}^*,\boldsymbol{c}\in C^n$. *Let*

$$\boldsymbol{M}=\begin{bmatrix}\left(\boldsymbol{D}-\lambda^*\boldsymbol{I}\right) & -\boldsymbol{x}^* \\ \boldsymbol{c}^H & 0\end{bmatrix}, \tag{10}$$

be an $(n+1)$ *by* $(n+1)$ *matrix. If* $\boldsymbol{D}-\lambda^*\boldsymbol{I}$ *is singular and* $\operatorname{rank}\left(\boldsymbol{D}-\lambda^*\boldsymbol{I}\right)=n-1$, *then* $\boldsymbol{M}$ *is nonsingular if and only if* $\psi^H\boldsymbol{x}^*\neq 0$, *for all* $\psi\in\mathcal{N}\left(\boldsymbol{D}-\lambda^*\boldsymbol{I}\right)^H\setminus\{0\}$ *and* $\boldsymbol{c}^H\phi\neq 0$, *for all* $\phi\in\mathcal{N}\left(\boldsymbol{D}-\lambda^*\boldsymbol{I}\right)\setminus\{0\}$. *Where* $\mathcal{N}\left(\boldsymbol{D}-\lambda^*\boldsymbol{I}\right)$ *is the nullspace of* $\boldsymbol{D}-\lambda^*\boldsymbol{I}$.

Proof: Let $\boldsymbol{M}$ be nonsingular. Assume $\boldsymbol{D}-\lambda^*\boldsymbol{I}$ is singular and $\boldsymbol{c}^H\phi=0$, we want to show by contradiction that $\boldsymbol{c}^H\phi\neq 0$. We multiply $\boldsymbol{M}$ from the right by the nonzero vector $[\phi,0]^H$ to yield

$$\begin{bmatrix}\left(\boldsymbol{D}-\lambda^*\boldsymbol{I}\right) & -\boldsymbol{x}^* \\ \boldsymbol{c}^H & 0\end{bmatrix}\begin{bmatrix}\phi \\ 0\end{bmatrix}=\begin{bmatrix}\left(\boldsymbol{D}-\lambda^*\boldsymbol{I}\right)\phi \\ \boldsymbol{c}^H\phi\end{bmatrix}=\begin{bmatrix}0 \\ 0\end{bmatrix}. \tag{11}$$

This shows that we have multiplied the nonsingular matrix $\boldsymbol{M}$ by a nonzero vector to obtain the zero vector, this implies that $\boldsymbol{M}$ is singular, a contradiction, hence $\boldsymbol{c}^H\phi \neq 0$. Similarly, let $\boldsymbol{\psi}^H\boldsymbol{x}^* = 0$, multiply $\boldsymbol{M}$ from the left by the nonzero vector $[\psi, 0]^H$ to obtain

$$[\psi \quad 0]^H \begin{bmatrix} (\boldsymbol{D}-\lambda^*\boldsymbol{I}) & -\boldsymbol{x}^* \\ \boldsymbol{c}^H & 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\psi}^H(\boldsymbol{D}-\lambda^*\boldsymbol{I}) & -\boldsymbol{\psi}^H\boldsymbol{x}^* \end{bmatrix} = \begin{bmatrix} 0^H & 0 \end{bmatrix}.$$

This shows that $\boldsymbol{M}$ is singular, contradicting the nonsingularity of $\boldsymbol{M}$, therefore, $\boldsymbol{\psi}^H\boldsymbol{x}^* \neq 0$.

Conversely, let $\boldsymbol{D}-\lambda^*\boldsymbol{I}$ be singular of $\operatorname{rank}(\boldsymbol{D}-\lambda^*\boldsymbol{I}) = n-1$, and assume $\boldsymbol{\psi}^H\boldsymbol{x}^* \neq 0$ and $\boldsymbol{c}^H\phi \neq 0$. We want to show that $\boldsymbol{M}$ is nonsingular. If we can show that the vector $[\boldsymbol{p}, q]^H$ is zero in

$$\begin{bmatrix} (\boldsymbol{D}-\lambda^*\boldsymbol{I}) & -\boldsymbol{x}^* \\ \boldsymbol{c}^H & 0 \end{bmatrix}\begin{bmatrix} \boldsymbol{p} \\ q \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ 0 \end{bmatrix},$$

then $\boldsymbol{M}$ is nonsingular. After expanding the above equation, we obtain

$$(\boldsymbol{D}-\lambda^*\boldsymbol{I})\boldsymbol{p} - q\boldsymbol{x}^* = 0 \tag{12}$$

$$\boldsymbol{c}^H\boldsymbol{p} = 0. \tag{13}$$

By using the fact that $\psi^H(\boldsymbol{D}-\lambda^*\boldsymbol{I}) = 0^H$ in $\psi^H(\boldsymbol{D}-\lambda^*\boldsymbol{I})\boldsymbol{p} - q(\boldsymbol{\psi}^H\boldsymbol{x}^*) = 0$, we have $q(\boldsymbol{\psi}^H\boldsymbol{x}^*) = 0$. But by assumption, $\boldsymbol{\psi}^H\boldsymbol{x}^* \neq 0$, hence $q = 0$. With this value of q, we are left with $(\boldsymbol{D}-\lambda^*\boldsymbol{I})\boldsymbol{p} = 0$ in (12) and because $\boldsymbol{D}-\lambda^*\boldsymbol{I}$ is singular, this implies that $\boldsymbol{p} = \alpha\phi$. After substituting the value of $\boldsymbol{p}$ into (13), we have $\alpha\boldsymbol{c}^H\phi = 0$ from which $\alpha = 0$ is immediate since $\boldsymbol{c}^H\phi \neq 0$. Therefore, $\boldsymbol{p} = 0$ and $\boldsymbol{M}$ is nonsingular.

Next, we present Algorithm 1 for computing the complex eigenpair of $\boldsymbol{D}$ using complex arithmetic. This is the main contribution to knowledge in this paper.

Algorithm 1 Eigenpair Computation using Newton's method

```
Input: $\boldsymbol{D}, \boldsymbol{x}^{(0)} = [\boldsymbol{c}, \lambda^{(0)}]^T, k_{\max}$ and tol.
1: for $k = 0, 1, 2, \cdots,$ until convergence do
2:     Compute the LU factorisation of
           $\begin{bmatrix} (\boldsymbol{D}-\lambda^{(k)}\boldsymbol{I}) & -\boldsymbol{x}^{(k)} \\ \boldsymbol{c}^H & 0 \end{bmatrix}.$
3:     Form
           $\boldsymbol{d}^{(k)} = -\begin{bmatrix} (\boldsymbol{D}-\lambda^{(k)}\boldsymbol{I})\boldsymbol{x}^{(k)} \\ \boldsymbol{c}^H\boldsymbol{x}^{(k)} - 1 \end{bmatrix}.$
4:     Solve the lower triangular system $L\boldsymbol{y}^{(k)} = \boldsymbol{d}^{(k)}$ for $\boldsymbol{y}^{(k)}$.
5:     Solve the upper triangular system $U\Delta\boldsymbol{v}^{(k)} = \boldsymbol{y}^{(k)}$ for $\Delta\boldsymbol{v}^{(k)}$.
6:     Update $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \Delta\boldsymbol{x}^{(k)}$.
7: end for
Out for: $\boldsymbol{v}^{(k_{\max})}$.
```

Stop Algorithm 1 as soon as $\left\|\Delta\boldsymbol{v}^{(k)}\right\| \leq \text{tol}$.

Next, we present Algorithm 4 for computing the complex eigenpair of $\boldsymbol{D}$ using complex arithmetic.

Algorithm 2 Eigenpair Computation using Gauss-Newton's method [1]

```
Input: $\boldsymbol{D}$, $\boldsymbol{v}^{(0)}=\left[\boldsymbol{x}_1^{(0)},\boldsymbol{x}_2^{(0)},\alpha^{(0)},\beta^{(0)}\right]^{\mathrm{T}}$, $k_{\max}$ and tol.
  1: for $k=0,1,2,\cdots,$ until convergence do
  2:     Find the reduced QR factorisation of $\boldsymbol{F}_v\left(\boldsymbol{v}^{(k)}\right)^{\mathrm{T}}=\boldsymbol{QR}$.
  3:     Solve $\boldsymbol{R}^{\mathrm{T}}\boldsymbol{g}^{(k)}=-\boldsymbol{F}\left(\boldsymbol{v}^{(k)}\right)$ for $\boldsymbol{g}^{(k)}$.
  4:     Compute $\Delta\boldsymbol{v}^{(k)}=\boldsymbol{Qg}^{(k)}$ for $\Delta\boldsymbol{v}^{(k)}$.
  5:     Update $\boldsymbol{v}^{(k+1)}=\boldsymbol{v}^{(k)}+\Delta\boldsymbol{v}^{(k)}$.
  6: end for
  Out for: $\boldsymbol{v}^{(k_{\max})}$.
```

Algorithm 3 Eigenpair Computation using Newton's method [6]

```
Input: $\boldsymbol{D},\boldsymbol{w}^{(0)}=\left[\boldsymbol{x}_1^{(0)},\boldsymbol{x}_2^{(0)}\right],\boldsymbol{v}^{(0)}=\left[\boldsymbol{w}^{(0)},\alpha^{(0)},\beta^{(0)}\right]^{\mathrm{T}},k_{\max}$ and tol.
  1: for $k=0,1,2,\cdots$ until convergence do
  2:     Compute the LU factorisation of
             $\begin{bmatrix}\boldsymbol{M} & -\boldsymbol{w} & \boldsymbol{Jw}\\ -\boldsymbol{w}^{\mathrm{T}} & 0 & 0\\ (\boldsymbol{Jw})^{\mathrm{T}} & 0 & 0\end{bmatrix}.$
  3:     Form
             $\boldsymbol{d}^{(k)}=\begin{bmatrix}-\boldsymbol{Mw}\\ \frac{1}{2}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}-1\right)\\ 0\end{bmatrix}.$
  4:     Solve the lower triangular system $L\boldsymbol{c}^{(k)}=\boldsymbol{d}^{(k)}$ for $\boldsymbol{c}^{(k)}$.
  5:     Solve the upper triangular system $U\Delta\boldsymbol{v}^{(k)}=\boldsymbol{c}^{(k)}$ for $\Delta\boldsymbol{v}^{(k)}$.
  6:     Update $\boldsymbol{v}^{(k+1)}=\boldsymbol{v}^{(k)}+\Delta\boldsymbol{v}^{(k)}$.
  7: end for
  Out for: $\boldsymbol{v}^{(k_{\max})}=\left[\boldsymbol{w}^{(k_{\max})},\alpha^{(k_{\max})},\beta^{(k_{\max})}\right]^{\mathrm{T}}$.
```

Algorithm 4 Eigenpair Computation using Newton's method in complex arithmetic [7]

```
Input: $\boldsymbol{D},\boldsymbol{v}^{(0)}=\left[\boldsymbol{x}_1^{(0)},\lambda^{(0)}\right]^{\mathrm{T}},k_{\max}$ and tol.
  1: for $k=0,1,2,\cdots,$ until convergence do
  2:     Compute the LU factorisation of $\begin{bmatrix}\boldsymbol{D}-\lambda^{(k)}\boldsymbol{I} & -\boldsymbol{x}^{(k)}\\ -\left(\boldsymbol{x}^{(k)}\right)^{H} & 0\end{bmatrix}.$
  3:     Form $\boldsymbol{d}^{(k)}=-\begin{bmatrix}\left(\boldsymbol{D}-\lambda^{(k)}\boldsymbol{I}\right)\boldsymbol{x}^{(k)}\\ -\frac{1}{2}\boldsymbol{x}^{(k)H}\boldsymbol{x}^{(k)}+\frac{1}{2}\end{bmatrix}.$
  4:     Solve the lower triangular system $L\boldsymbol{y}^{(k)}=\boldsymbol{d}^{(k)}$ for $\boldsymbol{y}^{(k)}$.
  5:     Solve the upper triangular system $U\Delta\boldsymbol{v}^{(k)}=\boldsymbol{y}^{(k)}$ for $\Delta\boldsymbol{v}^{(k)}$.
  6:     Update $\boldsymbol{v}^{(k+1)}=\boldsymbol{v}^{(k)}+\Delta\boldsymbol{v}^{(k)}$.
  7: end for
  Out for: $\boldsymbol{v}^{(k_{\max})}$.
```

3. Numerical Experiments

In this section, we compare the performance of the algorithm (Algorithm 1) obtained by adding Ruhe's normalization with three other algorithms (Algorithm

2, Algorithm 3 and Algorithm 4) which were presented in the last section on three numerical examples. Throughout this section $\boldsymbol{w}^{(k)} = \left[\boldsymbol{x}_1^{(k)T}, \boldsymbol{x}_2^{(k)T}\right]$ and $\boldsymbol{\lambda}^{(k)} = \left[\alpha^{(k)}, \beta^{(k)}\right]$.

Example 3.1.

Consider the matrix

$$D = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

We compared the performance of the four algorithms on the two by two matrix and the results are as presented in Table 1, Table 2, Table 3 and Table 4 respectively. In all four algorithms we used the same initial guesses $\alpha^{(0)} = 6.0\times10^{-3}$, $\beta^{(0)} = 9.9\times10^{-1}i$, $\boldsymbol{x}^{(0)} = \begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix}i,$ and $\boldsymbol{c} = \mathcal{N}(A+iI)$. It was observed that

Table 1. Values of $\alpha^{(k)} + i\beta^{(k)}$ for the two by two matrix using Algorithm 1.

k	$\alpha^{(k)} + i\beta^{(k)}$	$\lVert \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)} \rVert$	$\lvert \boldsymbol{\lambda}^{(k+1)} - \boldsymbol{\lambda}^{(k)} \rvert$	$\lVert \Delta \boldsymbol{v}^{(k)} \rVert$	$\lVert \boldsymbol{F}(\boldsymbol{v}^{(k)}) \rVert$
0	6.00000e−03+9.90000e−01i	1.7e+02	2.0e+00	1.7e+02	2.7e+00
1	1.41739e+00+2.39290e+00i	1.7e+02	3.7e+00	1.7e+02	3.4e+02
2	7.08322e−14−1.00000e+00i	3.2e+02	1.4e−13	3.2e+02	6.3e+02
3	4.90140e−16−1.00000e+00i	2.2e−11	5.4e−16	2.2e−11	4.4e−11

Table 2. Values of $\alpha^{(k)} + i\beta^{(k)}$ for the two by two matrix using Algorithm 2. Quadratic convergence is shown in columns 4, 6 and 7 for $k = 3, k = 4$ and $k = 5$.

k	$\alpha^{(k)}$	$\beta^{(k)}$	$\lVert \boldsymbol{w}^{(k+1)} - \boldsymbol{w}^{(k)} \rVert$	$\lvert \boldsymbol{\lambda}^{(k+1)} - \boldsymbol{\lambda}^{(k)} \rvert$	$\lVert \Delta \boldsymbol{v}^{(k)} \rVert$	$\lVert \boldsymbol{F}(\boldsymbol{v}^{(k)}) \rVert$
0	6.00000e−03	0.99000	1.1e+00	1.8e−02	1.1e+00	2.1e+00
1	3.09120e−03	1.00505	4.2e−01	4.3e−03	4.2e−01	6.4e−01
2	8.65482e−04	1.00141	8.2e−02	1.5e−03	8.2e−02	8.9e−02
3	6.54625e−05	1.00011	3.4e−03	1.2e−04	3.4e−03	3.4e−03
4	2.19153e−07	1.00000	5.6e−06	4.2e−07	5.7e−06	5.7e−06
5	1.23636e−12	1.00000	1.6e−11	2.4e−12	1.6e−11	1.6e−11
6	4.04413e−18	1.00000	7.9e−17	1.0e−18	7.9e−17	1.6e−16

Table 3. Values of $\alpha^{(k)}$ and $\beta^{(k)}$ for the two by two matrix using Algorithm 3. Quadratic convergence is shown in columns 4, 6 and 7 for $k = 3, 4$ and $k = 5$.

k	$\alpha^{(k)}$	$\beta^{(k)}$	$\lVert \boldsymbol{w}^{(k+1)} - \boldsymbol{w}^{(k)} \rVert$	$\lvert \boldsymbol{\lambda}^{(k+1)} - \boldsymbol{\lambda}^{(k)} \rvert$	$\lVert \Delta \boldsymbol{v}^{(k)} \rVert$	$\lVert \boldsymbol{F}(\boldsymbol{v}^{(k)}) \rVert$
0	6.00000e−03	0.99000	1.1e+00	1.8e−02	1.1e+00	2.1e+00
1	3.09120e−03	1.00505	4.2e−01	4.3e−03	4.2e−01	6.4e−01
2	8.65482e−04	1.00141	8.2e−02	1.5e−03	8.2e−02	8.9e−02
3	6.54625e−05	1.00011	3.4e−03	1.2e−04	3.4e−03	3.4e−03
4	2.19153e−07	1.00000	5.6e−06	4.2e−07	5.7e−06	5.7e−06
5	1.23636e−12	1.00000	1.6e−11	2.4e−12	1.6e−11	1.6e−11
6	4.41777e−18	1.00000	0.0e+00	0.0e+00	0.0e+00	0.0e+00

Table 4. Values of $\alpha^{(k)} + i\beta^{(k)}$ for the two by two matrix using Algorithm 4. Quadratic convergence is shown in columns 3, 5 and 6 for $k = 3,4$ and $k = 5$.

k	$\alpha^{(k)} + i\beta^{(k)}$	$\left\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\right\|$	$\left\|\boldsymbol{\lambda}^{(k+1)} - \boldsymbol{\lambda}^{(k)}\right\|$	$\left\|\Delta\boldsymbol{v}^{(k)}\right\|$	$\left\|\boldsymbol{F}\left(\boldsymbol{v}^{(k)}\right)\right\|$
0	6.00000e−03+9.90000e−01i	1.1e+00	1.8e−02	1.1e+00	2.1e+00
1	−3.09120e−03+1.00505e+00i	4.2e−01	4.3e−03	4.2e−01	6.4e−01
2	−8.65482e−04+1.00141e+00i	8.2e−02	1.5e−03	8.2e−02	8.9e−02
3	−6.54625e−05+1.00011e+00i	3.4e−03	1.2e−04	3.4e−03	3.4e−03
4	−2.19153e−07+1.00000e+00i	5.6e−06	4.2e−07	5.7e−06	5.7e−06
5	−1.23633e−12+1.00000e+00i	1.6e−11	2.4e−12	1.6e−11	1.6e−11
6	1.68283e−17+1.00000e+00i	0.0e+00	8.4e−18	7.6e−17	1.5e−16

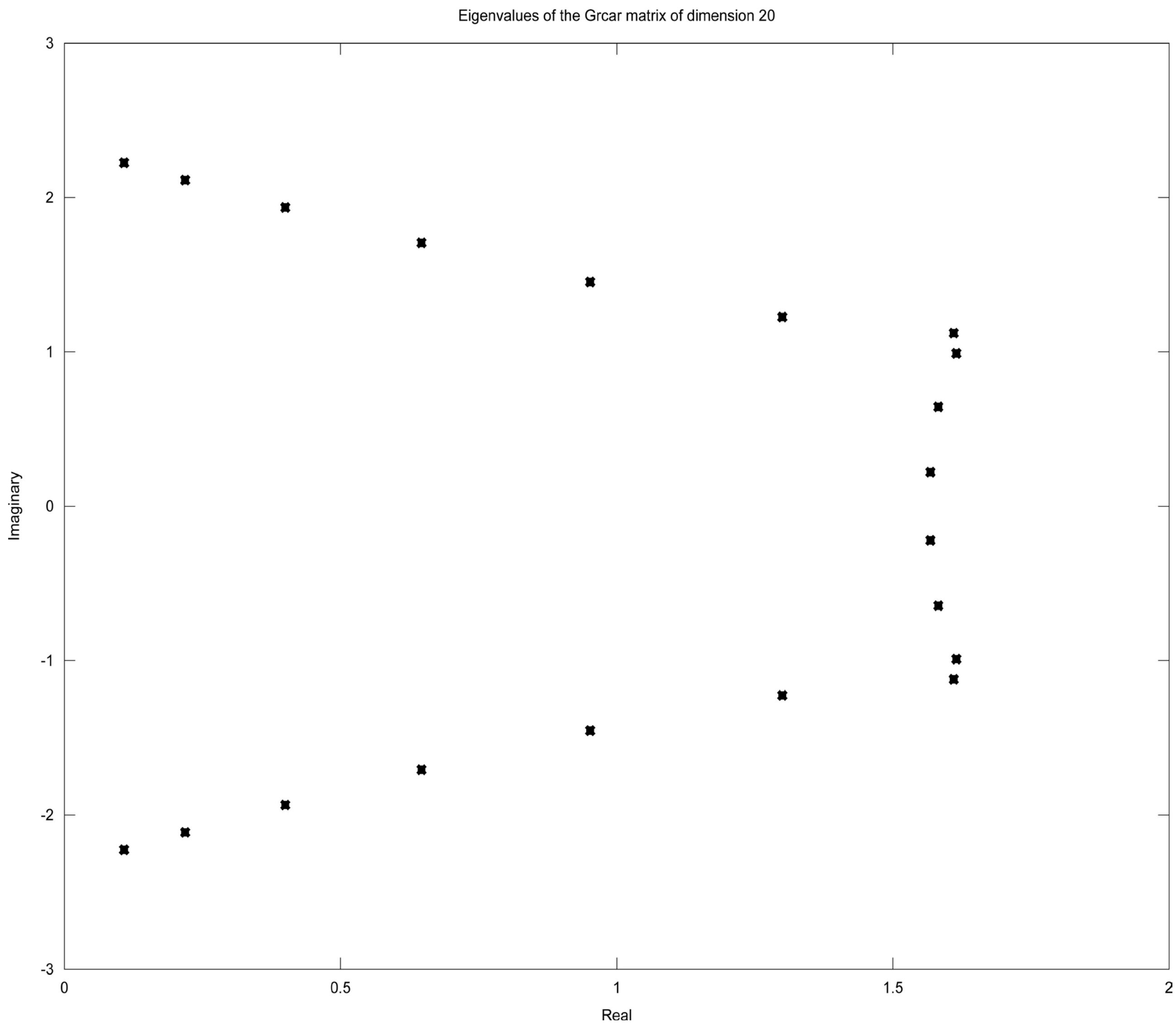

Figure 1. Distribution of the complex eigenvalues of the 20 by 20 grcar matrix. The x-axis is the real axis while the y-axis is the imaginary axis.

Algorithm 1 converged after only four iterations while it took seven iterates for the other three to converge to the eigenvalue $\lambda^* = i$.

Example 3.2.

The grcar matrix [12] [13] is a non symmetric matrix with sensitive eigenvalues and defined by

$$\boldsymbol{D}(i,j)=\begin{cases}-1, & \text{if} \quad i=j+1\\ 1, & \text{if} \quad i\le j \quad \text{and} \quad j\le i+k\\ 0, & \text{Otherwise.}\end{cases}$$

Figure 1 shows the distribution of the complex eigenvalues of the twenty by twenty grcar matrix on the real and imaginary axis. All the four algorithms discussed in the last section converged to the eigenvalue $\lambda^* = 1.58207 + 6.43690\times10^{-1} i$ after 12 iterations with the same starting guesses as shown in Table 5, Table 6, Table 7 and Table 8. However, unlike the first example in which Algorithm 1 converged faster than the other three, that was not the case, maybe due to the sensitivity of its eigenvalues.

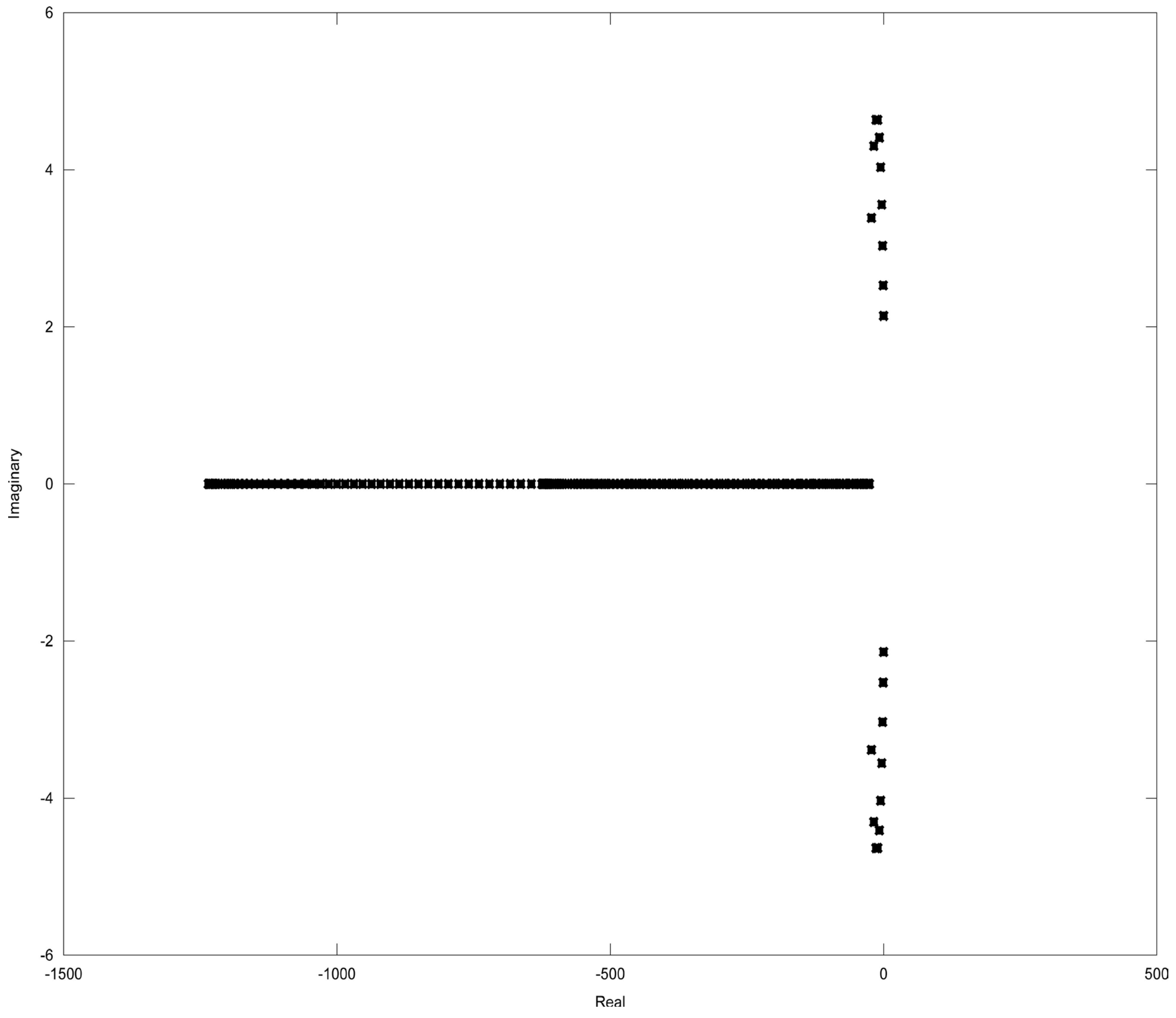

Figure 2. Distribution of the complex eigenvalues of the 200 by 200 bmw 200.mtx matrix. The x-axis is the real axis while the y-axis is the imaginary axis.

Example 3.3.

Consider the 200 by 200 matrix $\boldsymbol{D}$ bwm200.mtx from the matrix market library [14]. It is the discretised Jacobian of the Brusselator wave model for a chemical reaction. The resulting eigenvalue problem with $P = \boldsymbol{I}$ was also studied in [9] and we are interested in finding the rightmost eigenvalue of $\boldsymbol{D}$ which is closest to the imaginary axis and its corresponding eigenvector. **Figure 2** shows the distribution of the complex eigenvalues of the matrix.

For this example, in all four algorithms we take $\alpha^{(0)} = 0.0,\ \beta^{(0)} = 2.5$ in line with [9] and took $\boldsymbol{x}_1^{(0)} = 1/2\|1\|$, $\boldsymbol{x}_2^{(0)} = \frac{\sqrt{3}}{2}1/\|1\|$ and $\boldsymbol{c} = 1 + \frac{1}{2\|1\|}i$, where 1 is

Table 5. Values of $\alpha^{(k)} + i\beta^{(k)}$ for the twenty by twenty grcar matrix using Algorithm 1. Quadratic convergence is shown in columns 3 and 5 for $k = 10$ and 11.

k	$\alpha^{(k)} + i\beta^{(k)}$	$\left\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\right\|$	$\left\|\boldsymbol{\lambda}^{(k+1)} - \boldsymbol{\lambda}^{(k)}\right\|$	$\left\|\Delta\boldsymbol{v}^{(k)}\right\|$	$\left\|\boldsymbol{F}\left(\boldsymbol{v}^{(k)}\right)\right\|$
0	9.00000e−02+2.00000e−01i	1.2e+00	1.3e+00	1.8e+00	2.2e+00
1	1.08132e+00+1.08186e+00i	1.5e+00	2.6e+00	3.0e+00	1.6e+00
2	3.72785e+00+9.63132e−01i	9.1e−01	1.3e+00	1.6e+00	4.0e+00
3	2.47494e+00+7.62445e−01i	5.7e−01	7.6e−01	9.4e−01	1.2e+00
4	1.71950e+00+7.19775e−01i	7.6e−01	4.6e−01	8.9e−01	4.3e−01
5	1.26075e+00+7.25835e−01i	3.7e−01	3.1e−01	4.8e−01	3.5e−01
6	1.56634e+00+7.38078e−01i	3.1e−01	1.6e−01	3.5e−01	1.1e−01
7	1.62480e+00+5.86198e−01i	1.1e−01	6.7e−02	1.3e−01	5.1e−02
8	1.58877e+00+6.42286e−01i	1.9e−02	6.8e−03	2.0e−02	7.7e−03
9	1.58214e+00+6.43859e−01i	4.3e−04	1.8e−04	4.6e−04	1.3e−04
10	1.58207e+00+6.43690e−01i	2.1e−07	7.1e−08	2.2e−07	7.8e−08
11	1.58207e+00+6.43690e−01i	4.6e−14	1.9e−14	4.9e−14	1.5e−14

Table 6. Values of $\alpha^{(k)} + i\beta^{(k)}$ for the twenty by twenty grcar matrix using Algorithm 2. Quadratic convergence is shown in columns 4, 5 and 6 for $k = 9,10$ and 11.

k	$\alpha^{(k)}$	$\beta^{(k)}$	$\left\|\boldsymbol{w}^{(k+1)} - \boldsymbol{w}^{(k)}\right\|$	$\left\|\boldsymbol{\lambda}^{(k+1)} - \boldsymbol{\lambda}^{(k)}\right\|$	$\left\|\Delta\boldsymbol{v}^{(k)}\right\|$	$\left\|\boldsymbol{F}\left(\boldsymbol{v}^{(k)}\right)\right\|$
0	9.00000e−02	0.20000	7.6e−01	1.4e+00	1.6e+00	2.0e+00
1	1.49489e+00	0.11791	1.1e+00	1.1e+00	1.5e+00	1.1e+00
2	2.03853e+00	1.09037	5.5e−01	5.1e−01	7.5e−01	1.3e+00
3	1.70500e+00	0.69852	4.6e−01	3.2e−01	5.6e−01	3.2e−01
4	1.50162e+00	0.45319	5.5e−01	3.8e−01	6.7e−01	1.8e−01
5	1.84080e+00	0.63082	3.1e−01	1.9e−01	3.6e−01	2.6e−01
6	1.66945e+00	0.55208	1.5e−01	1.3e−01	2.0e−01	7.6e−02
7	1.55071e+00	0.60360	7.0e−02	5.4e−02	8.9e−02	2.3e−02
8	1.58816e+00	0.64225	7.9e−03	6.2e−03	1.0e−02	4.5e−03
9	1.58207e+00	0.64358	1.4e−04	1.1e−04	1.8e−04	5.9e−05
10	1.58207e+00	0.64369	3.0e−08	2.1e−08	3.7e−08	1.8e−08
11	1.58207e+00	0.64369	2.0e−15	1.6e−15	2.5e−15	7.8e−16

Table 7. Values of $\alpha^{(k)} + i\beta^{(k)}$ for the twenty by twenty grcar matrix using Algorithm 3. Quadratic convergence is shown in columns 4, 5 and 6 for $k = 9,10$ and 11.

k	$\alpha^{(k)}$	$\beta^{(k)}$	$\left\|\mathbf{w}^{(k+1)} - \mathbf{w}^{(k)}\right\|$	$\left\|\boldsymbol{\lambda}^{(k+1)} - \boldsymbol{\lambda}^{(k)}\right\|$	$\left\|\Delta\mathbf{v}^{(k)}\right\|$	$\left\|\mathbf{F}\left(\mathbf{v}^{(k)}\right)\right\|$
0	9.00000e−02	0.20000	7.6e−01	1.4e+00	1.6e+00	2.0e+00
1	1.49489e+00	0.11791	1.1e+00	1.1e+00	1.5e+00	1.1e+00
2	2.03853e+00	1.09037	5.5e−01	5.1e−01	7.5e−01	1.3e+00
3	1.70500e+00	0.69852	4.6e−01	3.2e−01	5.6e−01	3.2e−01
4	1.50162e+00	0.45319	5.5e−01	3.8e−01	6.7e−01	1.8e−01
5	1.84080e+00	0.63082	3.1e−01	1.9e−01	3.6e−01	2.6e−01
6	1.66945e+00	0.55208	1.5e−01	1.3e−01	2.0e−01	7.6e−02
7	1.55071e+00	0.60360	7.0e−02	5.4e−02	8.9e−02	2.3e−02
8	1.58816e+00	0.64225	7.9e−03	6.2e−03	1.0e−02	4.5e−03
9	1.58207e+00	0.64358	1.4e−04	1.1e−04	1.8e−04	5.9e−05
10	1.58207e+00	0.64369	3.0e−08	2.1e−08	3.7e−08	1.8e−08
11	1.58207e+00	0.64369	2.0e−15	1.6e−15	2.5e−15	8.3e−16

Table 8. Values of $\alpha^{(k)} + i\beta^{(k)}$ for the twenty by twenty grcar matrix using Algorithm 4. Quadratic convergence is shown in columns 4, 5 and 6 for $k = 9,10$ and 11.

k	$\alpha^{(k)} + i\beta^{(k)}$	$\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|$	$\left\|\boldsymbol{\lambda}^{(k+1)} - \boldsymbol{\lambda}^{(k)}\right\|$	$\left\|\Delta\mathbf{v}^{(k)}\right\|$	$\left\|\mathbf{F}\left(\mathbf{v}^{(k)}\right)\right\|$
0	9.00000e−02+2.00000e−01i	7.6e−01	1.4e+00	1.6e+00	2.0e+00
1	1.49489e+00+1.17906e−01i	1.1e+00	1.1e+00	1.5e+00	1.1e+00
2	2.03853e+00+1.09037e+00i	5.5e−01	5.1e−01	7.5e−01	1.3e+00
3	1.70500e+00+6.98519e−01i	4.6e−01	3.2e−01	5.6e−01	3.2e−01
4	1.50162e+00+4.53192e−01i	5.5e−01	3.8e−01	6.7e−01	1.8e−01
5	1.84080e+00+6.30823e−01i	3.1e−01	1.9e−01	3.6e−01	2.6e−01
6	1.66945e+00+5.52083e−01i	1.5e−01	1.3e−01	2.0e−01	7.6e−02
7	1.55071e+00+6.03599e−01i	7.0e−02	5.4e−02	8.9e−02	2.3e−02
8	1.58816e+00+6.42251e−01i	7.9e−03	6.2e−03	1.0e−02	4.5e−03
9	1.58207e+00+6.43581e−01i	1.4e−04	1.1e−04	1.8e−04	5.9e−05
10	1.58207e+00+6.43690e−01i	3.0e−08	2.1e−08	3.7e−08	1.8e−08
11	1.58207e+00+6.43690e−01i	2.0e−15	1.6e−15	2.5e−15	7.9e−16

the vector of all ones. Results of numerical experiments are as tabulated in Tables 9-12 respectively. We observed that while it took Algorithm 1 with a fixed complex vector six iterations to converge to the desired eigenvalue $1.81999 \times 10^{-5} + 2.13950i$ as shown in Table 9, it took eight, ten and eight iterates for Algorithm 2 (Table 10), Algorithm 3 (Table 11) and Algorithm 4 (Table 12) respectively to achieve the same result. This shows that Algorithm 1 converged faster than the others.

As shown in Table 1 and Table 9, we observed that an application of Algorithm 1 showed faster convergence to the eigenvalue of interest with a close enough initial guess than the previous algorithms already discussed in [1] [6] and [7] viz-a-viz: Algorithm 2, Algorithm 3 and Algorithm 4 respectively.

Table 9. Values of $\alpha^{(k)} + i\beta^{(k)}$ for the 200 by 200 matrix of Example 3.3 using Algorithm 1. Columns 4 and 5 shows that the results converged quadratically for $k = 1,2,3$ and 4.

k	$\alpha^{(k)} + i\beta^{(k)}$	$\left\|x^{(k+1)} - x^{(k)}\right\|$	$\left\|\lambda^{(k+1)} - \lambda^{(k)}\right\|$	$\left\|\Delta v^{(k)}\right\|$	$\left\|F\left(v^{(k)}\right)\right\|$
0	0.00000e+00+2.50000e+00i	1.0e+00	5.5e−02	1.0e+00	3.8e+01
1	−5.34905e−02+2.48607e+00i	4.2e−02	3.7e−01	3.8e−01	5.5e−02
2	−2.93885e−03+2.11634e+00i	4.0e−03	2.3e−02	2.4e−02	1.6e−02
3	1.47186e−04+2.13954e+00i	4.5e−05	1.3e−04	1.4e−04	9.5e−05
4	1.82101e−05+2.13950e+00i	2.3e−09	1.1e−08	1.1e−08	6.1e−09
5	1.81999e−05+2.13950e+00i	7.9e−15	1.2e−14	1.5e−14	1.8e−14

Table 10. Values of $\alpha^{(k)}$ and $\beta^{(k)}$ for the 200 by 200 matrix of Example 3.3 using Algorithm 2. Columns 6 and 7 show that the results converged quadratically for $k = 3,4,5,6$ and 7.

k	$\alpha^{(k)}$	$\beta^{(k)}$	$\left\|w^{(k+1)} - w^{(k)}\right\|$	$\left\|\lambda^{(k+1)} - \lambda^{(k)}\right\|$	$\left\|\Delta v^{(k)}\right\|$	$\left\|F\left(v^{(k)}\right)\right\|$
0	0.00000e+00	2.50000	3.8e+00	7.8e−01	3.9e+00	3.6e+01
1	2.34253e−01	1.75371	1.8e+00	2.2e−01	1.8e+00	7.8e+00
2	1.18745e−01	1.94460	8.1e−01	1.4e−01	8.2e−01	1.7e+00
3	4.47044e−02	2.06484	2.5e−01	7.0e−02	2.6e−01	3.4e−01
4	8.82702e−03	2.12479	3.1e−02	1.7e−02	3.5e−02	3.7e−02
5	2.48114e−04	2.13905	4.8e−04	5.2e−04	7.1e−04	7.1e−04
6	1.80714e−05	2.13950	1.2e−07	2.5e−07	2.8e−07	2.8e−07
7	1.81999e−05	2.13950	2.1e−14	2.9e−14	3.6e−14	6.0e−14

Table 11. Values of $\alpha^{(k)}$ and $\beta^{(k)}$ for the 200 by 200 matrix of Example 3.3 using Algorithm 3. Columns 6 and 7 show that the results converged quadratically for $k = 3,4,5,6$ and 7.

k	$\alpha^{(k)}$	$\beta^{(k)}$	$\left\|w^{(k+1)} - w^{(k)}\right\|$	$\left\|\lambda^{(k+1)} - \lambda^{(k)}\right\|$	$\left\|\Delta v^{(k)}\right\|$	$\left\|F\left(v^{(k)}\right)\right\|$
0	0.00000e+00	2.50000	3.8e+00	7.8e−01	3.9e+00	3.6e+01
1	2.34253e−01	1.75371	1.8e+00	2.2e−01	1.8e+00	7.8e+00
2	1.18745e−01	1.94460	8.1e−01	1.4e−01	8.2e−01	1.7e+00
3	4.47044e−02	2.06484	2.5e−01	7.0e−02	2.6e−01	3.4e−01
4	8.82702e−03	2.12479	3.1e−02	1.7e−02	3.5e−02	3.7e−02
5	2.48114e−04	2.13905	4.8e−04	5.2e−04	7.1e−04	7.1e−04
6	1.80714e−05	2.13950	1.2e−07	2.5e−07	2.8e−07	2.8e−07
7	1.81999e−05	2.13950	1.1e−14	9.2e−14	9.2e−14	6.3e−14
8	1.81999e−05	2.13950	1.7e−14	6.8e−14	7.0e−14	5.6e−14
9	1.81999e−05	2.13950	1.4e−14	6.8e−15	1.5e−14	5.1e−14

Table 12. Values of $\alpha^{(k)}$ and $\beta^{(k)}$ for the 200 by 200 matrix of Example 3.3 using Algorithm 4. Columns 5 and 6 show that the results converged quadratically for $k = 3, 4, 5, 6$ and 7.

k	$\alpha^{(k)} + i\beta^{(k)}$	$\left\|x^{(k+1)} - x^{(k)}\right\|$	$\left\|\lambda^{(k+1)} - \lambda^{(k)}\right\|$	$\left\|\Delta v^{(k)}\right\|$	$\left\|F\left(v^{(k)}\right)\right\|$
0	0.00000e+00+2.50000e+00i	3.8e+00	7.8e−01	3.9e+00	3.6e+01
1	2.34253e−01+1.75371e+00i	1.8e+00	2.2e−01	1.8e+00	7.8e+00
2	1.18745e−01+1.94460e+00i	8.1e−01	1.4e−01	8.2e−01	1.7e+00
3	4.47044e−02+2.06484e+00i	2.5e−01	7.0e−02	2.6e−01	3.4e−01
4	8.82702e−03+2.12479e+00i	3.1e−02	1.7e−02	3.5e−02	3.7e−02
5	2.48114e−04+2.13905e+00i	4.8e−04	5.2e−04	7.1e−04	7.1e−04
6	1.80714e−05+2.13950e+00i	1.2e−07	2.5e−07	2.8e−07	2.8e−07
7	1.81999e−05+2.13950e+00i	1.1e−14	3.7e−14	3.8e−14	6.3e−14

4. Conclusion

In this paper, we have shown using the ABCD Lemma that the Jacobian obtained from adding Ruhe's normalization to the matrix pencil is non-singular at the root. With a proper choice of the fixed complex vector and an initial guess close to the eigenvalue of interest, we recommend the use of Algorithm 1 for the numerical computation of the desired complex eigenpair of a matrix because of its faster convergence.

Acknowledgements

The authors acknowledge valuable suggestions of an anonymous referee which helped in improving the final version of this paper. The main part of this work was done when the first author was a Ph.D. student at the University of Bath and duly acknowledge financial support in the form of a studentship.

References

[1] Akinola, R.O. and Spence, A. (2014) Two-Norm Normalization for the Matrix Pencil: Inverse Iteration with a Complex Shift. *International Journal of Innovation in Science and Mathematics*, **2**, 435-439.

[2] Stewarti, G.W. (2001) Matrix Algorithms, Volume II: Eigensystems. SIAM, Philadelphia.

[3] Kreyszig, E. (1999) Advanced Engineering Mathematics. John Wiley & Sons Inc., New York.

[4] Ruhe, A. (1973) Algorithms for the Nonlinear Eigenvalue Problem. *SIAM Journal on Numerical Analysis*, **10**, 674-689. https://doi.org/10.1137/0710059

[5] Tisseur, F. (2001) Newton's Method in Floating Point Arithmetic and Iterative Refinement of Generalized Eigenvalue Problems. *SIAM Journal on Matrix Analysis and Applications*, **22**, 1038-1057. https://doi.org/10.1137/S0895479899359837

[6] Akinola, R.O. and Spence, A. (2015) Numerical Computation of the Complex Eigenvalues of a Matrix by Solving a Square System of Equations. *Journal of Natural Sciences Research*, **5**, 144-156.

[7] Akinola, R.O. (2015) Computing the Complex Eigenpair of a Large Sparse Matrix in

Complex Arithmetic. *International Journal of Pure & Engineering Mathematics,* **3**, 137-158.

[8] Keller, H.B. (1977) Numerical Solution of Bifurcation and Nonlinear Eigenvalue Problems. In: Rabinowitz, P., Ed., *Applications of Bifurcation Theory*, Academic Press, New York, 359-384.

[9] Parlett, B.N. and Saad, Y. (1987) Complex Shift and Invert Strategies for Real Matrices. *Linear Algebra and Its Applications*, **88**-**89**, 575-595. https://doi.org/10.1016/0024-3795(87)90126-1

[10] Meerbergen, K. and Roose, D. (1996) Matrix Transformations for Computing Rightmost Eigenvalues of Large Sparse Non-Symmetric Eigenvalue Problems. *IMA Journal of Numerical Analysis*, **16**, 297-346. https://doi.org/10.1093/imanum/16.3.297

[11] Deuflhard, P. (2004) Newton Methods for Nonlinear Problems. Springer, Heidelberg, 174-175.

[12] Grcar, J. (1989) Operator Coefficient Methods for Linear Equations. Technical Report SAND89-8691, Sandia National Laboratories, Albuquerque, New Mexico, Appendix 2.

[13] Nachtigal, N.M., Reichel, L. and Trefethen, L. (1992) A Hybrid GMRES Algorithm for Nonsymmetric Linear Systems. *SIAM Journal on Matrix Analysis and Applications*, **13**, 796-825. https://doi.org/10.1137/0613050

[14] Boisvert, B., Pozo, R., Remington, K., Miller, B. and Lipman, R. Matrix Market. http://math.nist.gov/MatrixMarket/