Journal of Computational and Applied Mathematics 233 (2010) 2994–3004

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Fast enclosure for all eigenvalues in generalized eigenvalue problems

Shinya Miyajima

Faculty of Engineering, Gifu University, 501-1193, Japan

ARTICLE INFO

Article history:
Received 3 September 2008

MSC:
65F15
65G20
65G50

Keywords:
Generalized eigenvalue problems
Guaranteed enclosure
Non-Hermitian case

ABSTRACT

A fast method for enclosing all eigenvalues in generalized eigenvalue problems $Ax = \lambda Bx$ is proposed. Firstly a theorem for enclosing all eigenvalues, which is applicable even if A is not Hermitian and/or B is not Hermitian positive definite, is presented. Next a theorem for accelerating the enclosure is presented. The proposed method is established based on these theorems. Numerical examples show the performance and property of the proposed method. As an application of the proposed method, an efficient method for enclosing all eigenvalues in polynomial eigenvalue problems is also sketched.

1. Introduction

In this paper, we are concerned with the accuracy of numerically computed eigenvalues in the generalized eigenvalue problems

$$Ax = \lambda Bx, \quad A, B \in \mathbb{C}^{n\times n},\ \lambda \in \mathbb{C},\ x \in \mathbb{C}^n \tag{1}$$

where λ is an eigenvalue and x is an eigenvector corresponding to λ. We assume that B is nonsingular. The problems (1) arise in many applications of scientific computations, e.g. stationary analysis of circuits, image processing, structure analysis and so forth.

There are several methods for calculating guaranteed enclosures of eigenvalues in (1), e.g. [1–8]. On enclosing eigenvalues in the case that A is Hermitian and B is Hermitian positive definite, see [1–4,7,8]. The case that A is Hermitian and B is Hermitian positive definite is important (see e.g. [9]). On the other hand, the case that A is not Hermitian and/or B is not Hermitian positive definite is also important. For instance, this case arises in the finite element analysis of Maxwell's equation and forward kinematics for the Stewart platform of robotics [10]. *A few specified eigenvalues and eigenvectors* can be enclosed by applying the method in [5,6] even if A is not Hermitian and/or B is not Hermitian positive definite. Especially in [6], methods are presented for computing enclosure of multiple eigenvalues and a basis for a corresponding invariant subspace. Excellent overviews on perturbation theory for matrix eigenvalues can be found in [9,11,12].

In this paper, we present a theorem for enclosing *all eigenvalues* in (1). This theorem is applicable even if A is not Hermitian and/or B is not Hermitian positive definite. Moreover we present a theorem for accelerating the enclosure, which is based on a priori error estimation (e.g. [13,14]) of floating point arithmetic. Based on these theorems, we propose a fast method of enclosing *all eigenvalues*. The proposed method supplies a rigorous error bound ε such that all eigenvalues are included in the set

$$\bigcup_{i=1}^{n} \left\{ z \in \mathbb{C} \,\middle|\, |z - \tilde{\lambda}_i| \le \varepsilon \right\}$$

E-mail address: miyajima@gifu-u.ac.jp.

0377-0427/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2009.11.048

where $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n$ denote approximate eigenvalues, when all approximate eigenvalues and eigenvectors are given. Moreover the proposed method takes into account the presence of underflow in floating point arithmetic.

As an application of the proposed method, we also sketch an efficient method of enclosing all eigenvalues in polynomial eigenvalue problems

$$(\lambda^m A_m + \cdots + \lambda A_1 + A_0)x = 0, \quad A_0, \ldots, A_m \in \mathbb{C}^{n\times n},\ \lambda \in \mathbb{C},\ x \in \mathbb{C}^n \tag{2}$$

where λ is an eigenvalue and x is an eigenvector corresponding to λ. We assume that A_m is nonsingular. The problems (2) with $m = 2$ arise in e.g. acoustic systems, electrical circuit simulation and structural mechanics.

2. Main theorem

In this section, we present a theorem for enclosing all eigenvalues in (1).

Throughout this paper, we assume that as a result of numerical computation, we have a diagonal matrix $\tilde{D} \in \mathbb{C}^{n\times n}$ and a matrix $\tilde{X} \in \mathbb{C}^{n\times n}$ such that $A\tilde{X} \approx B\tilde{X}\tilde{D}$. Then it holds approximately for all $i \in \{1, \ldots, n\}$ that $A\tilde{x}^{(i)} \approx \tilde{\lambda}_i B\tilde{x}^{(i)}$, where $\tilde{\lambda}_i$ and $\tilde{x}^{(i)}$ denote the (i, i) element of $\tilde{D}$ and the ith column of $\tilde{X}$, respectively. Moreover in this paper, I denotes the $n \times n$ identity matrix.

Firstly we cite Lemma 1 with respect to nonsingularity of $I \pm F$ and an upper bound of $\|(I \pm F)^{-1}\|_p (1 \le p \le \infty)$ for $F \in \mathbb{C}^{n\times n}$.

Lemma 1 (*E.g. Golub et al. [15]*). *For $F \in \mathbb{C}^{n\times n}$, if $\|F\|_p < 1$ $(1 \le p \le \infty)$, then $I \pm F$ is nonsingular and*

$$\|(I \pm F)^{-1}\|_p \le \frac{1}{1 - \|F\|_p}. \tag{3}$$

Remark 1. In [15], it is written that the results in Lemma 1 follow for real matrices. On the other hand, the results also follow for complex matrices. Similar can be said for Lemma 4.

We will use Lemma 1 in the proof of the presented theorem. The presented theorem is a modification of the Bauer–Fike theorem [16] suited for enclosing all eigenvalues in (1).

Theorem 1. *Let Y be an arbitrary $n \times n$ complex matrix. Let also $n \times n$ complex matrices R_1 and R_2 be defined as follows:*

$$R_1 := Y(A\tilde{X} - B\tilde{X}\tilde{D}), \qquad R_2 := YB\tilde{X} - I.$$

If $\|R_2\|_\infty < 1$, then B, $\tilde{X}$ and Y are nonsingular, and it follows that

$$\min_{1\le i\le n} |\lambda - \tilde{\lambda}_i| \le \varepsilon, \quad \varepsilon := \frac{\|R_1\|_\infty}{1 - \|R_2\|_\infty}. \tag{4}$$

Proof. $\|R_2\|_\infty < 1$ and Lemma 1 imply that B, $\tilde{X}$ and Y are nonsingular. From nonsingularity of B, (1) is equivalent to the standard eigenvalue problems $B^{-1}Ax = \lambda x$ so that $B^{-1}A - \lambda I$ is singular.

If λ coincides with one of the diagonal elements of $\tilde{D}$, then $\min_{1\le i\le n} |\lambda - \tilde{\lambda}_i| = 0$ so that (4) holds. Thus it is sufficient to consider only the case that λ does not coincide with any diagonal element of $\tilde{D}$. In this case, $\tilde{D} - \lambda I$ is nonsingular.

Noticing $\tilde{D} - \lambda I$ and $\tilde{X}$ be nonsingular, and $B^{-1}A - \lambda I$ be singular, it holds that

$$\begin{aligned}(\tilde{D} - \lambda I)^{-1}\tilde{X}^{-1}(B^{-1}A - \lambda I)\tilde{X} &= (\tilde{D} - \lambda I)^{-1}\tilde{X}^{-1}(\tilde{X}(\tilde{D} - \lambda I) - \tilde{X}(\tilde{D} - \lambda I) + (B^{-1}A - \lambda I)\tilde{X}) \\ &= I - (\tilde{D} - \lambda I)^{-1}\tilde{X}^{-1}B^{-1}(B\tilde{X}\tilde{D} - A\tilde{X}).\end{aligned} \tag{5}$$

From singularity of (5) and contraposition of Lemma 1, we have

$$\begin{aligned}1 &\le \|(\tilde{D} - \lambda I)^{-1}\tilde{X}^{-1}B^{-1}(B\tilde{X}\tilde{D} - A\tilde{X})\|_\infty \\ &\le \|(\tilde{D} - \lambda I)^{-1}\|_\infty \|\tilde{X}^{-1}B^{-1}(B\tilde{X}\tilde{D} - A\tilde{X})\|_\infty \\ &= \frac{1}{\min_{1\le i\le n} |\lambda - \tilde{\lambda}_i|} \|\tilde{X}^{-1}B^{-1}(A\tilde{X} - B\tilde{X}\tilde{D})\|_\infty.\end{aligned}$$

This and nonsingularity of Y yield

$$\begin{aligned}\min_{1\le i\le n} |\lambda - \tilde{\lambda}_i| &\le \|\tilde{X}^{-1}B^{-1}(A\tilde{X} - B\tilde{X}\tilde{D})\|_\infty \\ &= \|(I + (YB\tilde{X} - I))^{-1}Y(A\tilde{X} - B\tilde{X}\tilde{D})\|_\infty \\ &\le \|(I + R_2)^{-1}\|_\infty \|R_1\|_\infty.\end{aligned} \tag{6}$$

From $\|R_2\|_\infty < 1$, (3) and (6), we obtain (4). □

Remark 2. In practical use of Theorem 1, Y is computed such that $Y \approx (B\tilde{X})^{-1}$. See Algorithm 1 in Section 4 for detail.

Remark 3. Theorem 1 essentially means that all eigenvalues are included in the set

$$\bigcup_{i=1}^{n} \left\{ z \in \mathbb{C} \,\middle|\, |z - \tilde{\lambda}_i| \le \varepsilon \right\}.$$

3. A theorem for accelerating the enclosure

Let ε and R_2 be defined as in Theorem 1. In the proposed method, the rigorous upper bound of ε is computed for enclosing all eigenvalues. To compute the rigorous upper bound of ε, we need to compute the rigorous upper bound of $\|R_2\|_\infty$. Hence in this section, we present a theorem in regard to accelerating the computation of the rigorous upper bound of $\|R_2\|_\infty$.

Throughout this paper, $\mathrm{fl}(\cdot)$ denotes a result of floating point computations, where all operations inside parentheses are executed by ordinary floating point arithmetic fulfilling rounding mode instruction, especially $\mathrm{fl}_\square(\cdot)$ in rounding-to-nearest, $\mathrm{fl}_\triangle(\cdot)$ in rounding-upward and $\mathrm{fl}_\triangledown(\cdot)$ in rounding-downward. Moreover let $e := (1, \ldots, 1)^T \in \mathbb{R}^n$. Let also real numbers $\mathbf{u}$ and $\underline{\mathbf{u}}$ be defined as unit roundoff and underflow unit (especially, $\mathbf{u} = 2^{-53}$ and $\underline{\mathbf{u}} = 2^{-1074}$ in IEEE 754 double precision), respectively. Then γ_n denotes $\gamma_n := n\mathbf{u}/(1 - n\mathbf{u})$.

For $F_c \in \mathbb{C}^{n\times n}$ and $F_r \in \mathbb{R}^{n\times n}$ with $F_r \ge 0$, the notation $\langle F_c, F_r \rangle$ denotes a matrix interval whose center and radius are F_c and F_r, respectively. Additionally for $n \times n$ real matrices $\underline{F}$ and $\overline{F}$ with $\underline{F} \le \overline{F}$, the notation $[\underline{F}, \overline{F}]$ denotes a matrix interval whose lower and upper bounds are $\underline{F}$ and $\overline{F}$, respectively.

Firstly we cite Lemmas 2–4 regarding rounding errors in complex floating point addition, subtraction, multiplication and summation.

Lemma 2 (*E.g. Higham [14]*). *For $x, y \in \mathbb{C}$, floating point addition and subtraction according to IEEE 754 satisfy*

$$\mathrm{fl}_\square(x \pm y) = (x \pm y)(1 + \delta), \quad |\delta| \le \mathbf{u},$$

also in the presence of underflow.

Lemma 3 (*Brent et al. [13]*). *For $x, y \in \mathbb{C}$, if $\mathbf{u} \le 2^{-5}$, floating point multiplication according to IEEE 754 satisfy*

$$\mathrm{fl}_\square(xy) = xy(1 + \delta) + \eta, \quad |\delta| \le \sqrt{5}\mathbf{u}, \ |\eta| \le 4\underline{\mathbf{u}} \tag{7}$$

also in the presence of underflow.

Remark 4. The error term η in (7) is devised by the author. By adding this term, Lemma 3 holds also in the presence of underflow.

Lemma 4 (*E.g. Higham [14]*). *For $x = (x_1, \ldots, x_n)^T \in \mathbb{C}^n$, the following inequality holds including the presence of underflow:*

$$\left| \mathrm{fl}_\square \left(\sum_{i=1}^{n} x_i \right) - \sum_{i=1}^{n} x_i \right| \le \gamma_{n-1} \sum_{i=1}^{n} |x_i|.$$

Next we present Lemma 5 with regard to rounding errors in complex floating point dot product.

Lemma 5. *For $x, y \in \mathbb{C}^n$, if $\mathbf{u} \le 2^{-5}$, it follows that*

$$|\mathrm{fl}_\square(x^T y) - x^T y| \le \gamma'_{n-1} |x|^T |y| + 4n(1 + \gamma_{n-1})\underline{\mathbf{u}},$$

where $\gamma'_{n-1} := (\sqrt{5}\mathbf{u} + \gamma_{n-1}(1 + \sqrt{5}\mathbf{u}))$, also in the presence of underflow.

Proof. For $x = (x_1, \ldots, x_n)^T$ and $y = (y_1, \ldots, y_n)^T$, we have

$$\begin{aligned} \mathrm{fl}_\square(x^T y) - x^T y &= \mathrm{fl}_\square \left(\sum_{i=1}^{n} x_i y_i \right) - \sum_{i=1}^{n} x_i y_i \\ &= \sum_{i=1}^{n} (\mathrm{fl}_\square(x_i y_i) - x_i y_i) + \mathrm{fl}_\square \left(\sum_{i=1}^{n} x_i y_i \right) - \sum_{i=1}^{n} \mathrm{fl}_\square(x_i y_i). \end{aligned}$$

This and Lemmas 3 and 4 yield

$$\begin{aligned}
|\mathrm{fl}_\square(x^Ty) - x^Ty| &\le \sum_{i=1}^n |\mathrm{fl}_\square(x_iy_i) - x_iy_i| + \left|\mathrm{fl}_\square\left(\sum_{i=1}^n x_iy_i\right) - \sum_{i=1}^n \mathrm{fl}_\square(x_iy_i)\right| \\
&\le \sum_{i=1}^n (\sqrt{5}\mathbf{u}|x_i||y_i| + 4\underline{\mathbf{u}}) + \gamma_{n-1}\sum_{i=1}^n |\mathrm{fl}_\square(x_iy_i)| \\
&\le \sqrt{5}\mathbf{u}|x|^T|y| + 4n\underline{\mathbf{u}} + \gamma_{n-1}\sum_{i=1}^n ((1+\sqrt{5}\mathbf{u})|x_i||y_i| + 4\underline{\mathbf{u}}) \\
&= \sqrt{5}\mathbf{u}|x|^T|y| + 4n\underline{\mathbf{u}} + \gamma_{n-1}(1+\sqrt{5}\mathbf{u})|x|^T|y| + 4n\gamma_{n-1}\underline{\mathbf{u}} \\
&= \gamma'_{n-1}|x|^T|y| + 4n(1+\gamma_{n-1})\underline{\mathbf{u}}. \quad \square
\end{aligned}$$

Finally we present Theorem 2 with respect to an upper bound of $\|FG - I\|_\infty$ for $n \times n$ complex matrices F and G.

Theorem 2. *Let γ'_{n-1} be defined as in Lemma 5. For $n \times n$ complex matrices F and G, if $\mathbf{u} \le 2^{-5}$, the following inequality holds including the presence of underflow:*

$$\|FG - I\|_\infty \le \|\mathrm{fl}_\square(FG - I)\|_\infty + \gamma'_{n-1}\||F||G|e\|_\infty + \mathbf{u}(\|\mathrm{fl}_\square(FG)\|_\infty + 1) + 4n^2(1+\gamma_{n-1})\underline{\mathbf{u}}.$$

Proof. For F and G, we have

$$FG - I = \mathrm{fl}_\square(FG - I) + FG - \mathrm{fl}_\square(FG) + (\mathrm{fl}_\square(FG) - I) - \mathrm{fl}_\square(FG - I).$$

From this and Lemmas 2 and 5, it follows that

$$\begin{aligned}
|FG - I| &\le |\mathrm{fl}_\square(FG - I)| + |FG - \mathrm{fl}_\square(FG)| + |(\mathrm{fl}_\square(FG) - I) - \mathrm{fl}_\square(FG - I)| \\
&\le |\mathrm{fl}_\square(FG - I)| + \gamma'_{n-1}|F||G| + 4n(1+\gamma_{n-1})ee^T\underline{\mathbf{u}} + \mathbf{u}|\mathrm{fl}_\square(FG) - I| \\
&\le |\mathrm{fl}_\square(FG - I)| + \gamma'_{n-1}|F||G| + \mathbf{u}(|\mathrm{fl}_\square(FG)| + I) + 4n(1+\gamma_{n-1})ee^T\underline{\mathbf{u}}.
\end{aligned}$$

Hence we obtain the desired result. $\square$

From Theorem 2, we obtain Corollary 1.

Corollary 1. *Let Y and R_2 be defined as in Theorem 1. Let $n \times n$ real matrices Y_r and Y_i be the real and the imaginary part of Y, respectively. Let also γ'_{n-1} be defined as in Lemma 5. Moreover let $Z_c \in \mathbb{C}^{n\times n}$ and $Z_r \in \mathbb{R}^{n\times n}$ satisfy $Z_r \ge 0$ and $B\tilde{X} \in \langle Z_c, Z_r\rangle$. If $\mathbf{u} \le 2^{-5}$, it follows that*

$$\begin{aligned}
\|R_2\|_\infty &\le \|\mathrm{fl}_\square(YZ_c - I)\|_\infty + \|(|Y_r| + |Y_i|)Z_re\|_\infty + \gamma'_{n-1}\||Y||Z_c|e\|_\infty \\
&\quad + \mathbf{u}(\|\mathrm{fl}_\square(YZ_c)\|_\infty + 1) + 4n^2(1+\gamma_{n-1})\underline{\mathbf{u}},
\end{aligned} \tag{8}$$

also in the presence of underflow.

Proof. The result follows from

$$\begin{aligned}
\|R_2\|_\infty &\le \|Y\langle Z_c, Z_r\rangle - I\|_\infty = \|\langle YZ_c - I, (|Y_r| + |Y_i|)Z_r\rangle\|_\infty \\
&\le \|YZ_c - I\|_\infty + \|(|Y_r| + |Y_i|)Z_r\|_\infty
\end{aligned}$$

(see e.g. [17] about the equality) and Theorem 2. $\square$

Remark 5. Z_c and Z_r in Corollary 1 can be obtained via rounding mode controlled computation. See Algorithms 2 and 3 in Section 4 for detail.

From Corollary 1, if Z_c and Z_r have already been obtained, we need to execute complex matrix multiplication YZ_c only once in rounding-to-nearest for calculating the rigorous upper bound of $\|R_2\|_\infty$. Note that $\mathrm{fl}_\square(YZ_c)$ can be utilized for computing both of $\|\mathrm{fl}_\square(YZ_c - I)\|_\infty$ and $\|\mathrm{fl}_\square(YZ_c)\|_\infty$. The computational cost for $\mathrm{fl}_\square(YZ_c)$ is $8n^3$ flops. The computational cost for the other parts in (8) is $\mathcal{O}(n^2)$ flops.

4. The proposed method

Based on Sections 2 and 3, in this section, we propose a method for enclosing all eigenvalues in (1). Algorithm 1 displays the steps of the proposed method.

Algorithm 1. Let R_1 and R_2 be defined as in Theorem 1. Let also Z_c and Z_r be defined as in Corollary 1. This algorithm computes an rigorous error bound ε such that

$$\lambda \in \bigcup_{i=1}^{n} \left\{ z \in \mathbb{C} \,\middle|\, |z - \tilde{\lambda}_i| \le \varepsilon \right\}$$

for all λ on the assumptions that $\tilde{D}$ and $\tilde{X}$ are given, and $\mathbf{u} \le 2^{-5}$.

Step1 Compute Z_c and Z_r (see Algorithms 2 and 3).
Step2 Compute $Y \in \mathbb{C}^{n\times n}$ such that $Y \approx Z_c^{-1}$.
Step3 Compute ε_2, a rigorous upper bound of $\|R_2\|_\infty$ based on (8).
Step4 If $1 \le \varepsilon_2$, this algorithm fails. Terminate. Otherwise go to Step 5.
Step5 Compute $\varepsilon_1 := \|R_1\|_\infty$ reusing Z_c and Z_r (see Procedure 2).
Step6 Compute ε such that $\varepsilon = \mathrm{fl}_\triangle(\varepsilon_1/\mathrm{fl}_\triangledown(1-\varepsilon_2))$.

Note that nonsingularities of B, $\tilde{X}$ and Y are verified in Step 4.

Z_c and Z_r can be computed using Algorithms 2 and 3 for B and $\tilde{X}$, and the matrices which Algorithm 2 returns, respectively. In Algorithms 2–7, and Procedures 1 and 2, steps are expressed MATLAB-like.

Algorithm 2 (*Oishi [18]*). For $n \times n$ complex matrices F and G, this algorithm computes $n \times n$ real matrices $\underline{H_r}$, $\overline{H_r}$, $\underline{H_i}$ and $\overline{H_i}$ which satisfy $\underline{H_r} \le \overline{H_r}$, $\underline{H_i} \le \overline{H_i}$ and $FG \in [\underline{H_r}, \overline{H_r}] + \sqrt{-1}[\underline{H_i}, \overline{H_i}]$. Computational cost of this algorithm is $16n^3$ flops if F and G are dense.

$[\underline{H_r}, \overline{H_r}, \underline{H_i}, \overline{H_i}] = \mathtt{cprod}(F, G)$
$F_r = \mathtt{real}(F); \quad F_i = \mathtt{imag}(F);$
$G_r = \mathtt{real}(G); \quad G_i = \mathtt{imag}(G);$
$\underline{H_r} = \mathrm{fl}_\triangledown(F_r G_r + (-F_r)G_i);$
$\overline{H_i} = \mathrm{fl}_\triangledown(F_r G_i + F_i G_r);$
$\overline{H_r} = \mathrm{fl}_\triangle(F_r G_r + (-F_r)G_i);$
$\overline{H_i} = \mathrm{fl}_\triangle(F_r G_i + F_i G_r);$

In Algorithms 2 and 7, $\mathtt{real}(F)$ and $\mathtt{imag}(F)$ are operations to return the real and the imaginary part of a complex matrix F, respectively.

Algorithm 3. For $n \times n$ real matrices $\underline{H_r}$, $\overline{H_r}$, $\underline{H_i}$ and $\overline{H_i}$ with $\underline{H_r} \le \overline{H_r}$ and $\underline{H_i} \le \overline{H_i}$, this algorithm computes $H_c \in \mathbb{C}^{n\times n}$ and $H_r \in \mathbb{R}^{n\times n}$ which satisfy $H_r \ge 0$ and $[\underline{H_r}, \overline{H_r}] + \sqrt{-1}[\underline{H_i}, \overline{H_i}] \subseteq \langle H_c, H_r \rangle$. Computational cost of this algorithm is $\mathcal{O}(n^2)$ flops.

$[H_c, H_r] = \mathtt{ccr}(\underline{H_r}, \overline{H_r}, \underline{H_i}, \overline{H_i})$
$R_c = \mathrm{fl}_\triangle(\underline{H_r} + 0.5(\overline{H_r} - \underline{H_r}));$
$I_c = \mathrm{fl}_\triangle(\underline{H_i} + 0.5(\overline{H_i} - \underline{H_i}));$
$R_r = \mathrm{fl}_\triangle(\overline{R_c} - \underline{H_r});$
$I_r = \mathrm{fl}_\triangle(I_c - \underline{H_i});$
$H_c = R_c + \sqrt{-1}I_c;$ % floating point computations are not executed
$H_r = \mathrm{fl}_\triangle(|R_r + \sqrt{-1}I_r|);$

We present Algorithm 4 with respect to concrete implementation of Step 5 in Algorithm 1.

Algorithm 4. For $n \times n$ real matrices J_r and J_i, let $J := J_r + \sqrt{-1}J_i$. For $J, \underline{H_c} \in \mathbb{C}^{n\times n}$, and $H_r \in \mathbb{R}^{n\times n}$ with $H_r \ge 0$, this algorithm computes $n \times n$ real matrices $\underline{K_r}$, $\overline{K_r}$, $\underline{K_i}$ and $\overline{K_i}$ which satisfy $\underline{K_r} \le \overline{K_r}$, $\underline{K_i} \le \overline{K_i}$ and $J\langle H_c, H_r \rangle \subseteq [\underline{K_r}, \overline{K_r}] + \sqrt{-1}[\underline{K_i}, \overline{K_i}]$. Computational cost of this algorithm is $18n^3$ flops if J, H_c and H_r are dense.

$[\underline{K_r}, \overline{K_r}, \underline{K_i}, \overline{K_i}] = \mathtt{ccrprod}(J, H_c, H_r)$
$[\underline{M_r}, \overline{M_r}, \underline{M_i}, \overline{M_i}] = \mathtt{cprod}(J, H_c);$
$R = \mathrm{fl}_\triangle((|J_r| + |J_i|)H_r);$ % see e.g. [17]
$\overline{K_r} = \mathrm{fl}_\triangle(\overline{M_r} + R);$
$\overline{K_i} = \mathrm{fl}_\triangle(\overline{M_i} + R);$

$\underline{K_r} = \mathrm{fl}_\nabla(\underline{M_r} - R)$;
$\underline{K_i} = \mathrm{fl}_\nabla(\underline{M_i} - R)$;

We use Algorithms 3 and 4 at Step 5 in Algorithm 1. To show the computational efficiencies of Algorithms 3 and 4, we introduce Algorithms 5–7.

Algorithm 5 (*Oishi [18]*). For $n \times n$ real matrices $\underline{F}$ and $\overline{F}$ with $\underline{F} \le \overline{F}$, this algorithm computes $n \times n$ real matrices F_c and F_r which satisfy $F_r \ge 0$ and $[\underline{F}, \overline{F}] \subseteq \langle F_c, F_r \rangle$. Computational cost of this algorithm is $\mathcal{O}(n^2)$ flops.

`function` $[F_c, F_r] = \mathtt{cr}(\underline{F}, \overline{F})$
$F_c = \mathrm{fl}_\triangle(\underline{F} + 0.5(\overline{F} - \underline{F}))$;
$F_r = \mathrm{fl}_\triangle(F_c - \underline{F})$;

Algorithm 6 (*Oishi [18]*). For $n \times n$ real matrices F, G_c and G_r with $G_r \ge 0$, this algorithm computes $n \times n$ real matrices $\underline{H}$ and $\overline{H}$ which satisfy $\underline{H} \le \overline{H}$ and $F\langle G_c, G_r \rangle \subseteq [\underline{H}, \overline{H}]$. Computational cost of this algorithm is $6n^3$ flops if F, G_c and G_r are dense.

`function` $[\underline{H}, \overline{H}] = \mathtt{iprod}(F, G_c, G_r)$
$R = \mathrm{fl}_\triangle(|F|G_r)$;
$\overline{H} = \mathrm{fl}_\triangle(FG_c + R)$;
$\underline{H} = \mathrm{fl}_\nabla(FG_c - R)$;

Algorithm 7. For $J \in \mathbb{C}^{n\times n}$ and $n \times n$ real matrices $\underline{H_r}$, $\overline{H_r}$, $\underline{H_i}$ and $\overline{H_i}$ with $\underline{H_r} \le \overline{H_r}$ and $\underline{H_i} \le \overline{H_i}$, this algorithm computes $n \times n$ real matrices $\underline{K_r}$, $\overline{K_r}$, $\underline{K_i}$ and $\overline{K_i}$ which satisfy $\underline{K_r} \le \overline{K_r}$, $\underline{K_i} \le \overline{K_i}$ and $J([\underline{H_r}, \overline{H_r}] + \sqrt{-1}[\underline{H_i}, \overline{H_i}]) \subseteq [\underline{K_r}, \overline{K_r}] + \sqrt{-1}[\underline{K_i}, \overline{K_i}]$. Computational cost of this algorithm is $24n^3$ flops if J, $\underline{H_r}$, $\overline{H_r}$, $\underline{H_i}$ and $\overline{H_i}$ are dense.

`function` $[\underline{K_r}, \overline{K_r}, \underline{K_i}, \overline{K_i}] = \mathtt{ciprod}(J, \underline{H_r}, \overline{H_r}, \underline{H_i}, \overline{H_i})$
$J_r = \mathtt{real}(J)$; $J_i = \mathtt{imag}(J)$;
$[H_{rr}, H_{rc}] = \mathtt{cr}(\underline{H_r}, \overline{H_r})$;
$[H_{ir}, H_{ic}] = \mathtt{cr}(\underline{H_i}, \overline{H_i})$;
$[\underline{M_{rr}}, \overline{M_{rr}}] = \mathtt{iprod}(J_r, H_{rr}, H_{rc})$;
$[\underline{M_{ir}}, \overline{M_{ir}}] = \mathtt{iprod}(J_i, H_{rr}, H_{rc})$;
$[\underline{M_{ri}}, \overline{M_{ri}}] = \mathtt{iprod}(J_r, H_{ir}, H_{ic})$;
$[\underline{M_{ii}}, \overline{M_{ii}}] = \mathtt{iprod}(J_i, H_{ir}, H_{ic})$;
$\underline{K_r} = \mathrm{fl}_\nabla(\underline{M_{rr}} - \overline{M_{ii}})$;
$\underline{K_i} = \mathrm{fl}_\nabla(\underline{M_{ir}} + \underline{M_{ri}})$;
$\overline{K_r} = \mathrm{fl}_\triangle(\overline{M_{rr}} - \underline{M_{ii}})$;
$\overline{K_i} = \mathrm{fl}_\triangle(\overline{M_{ir}} + \overline{M_{ri}})$;

To compute ε_1 in Algorithm 1, we need to compute the rigorous enclosure of R_1 in Theorem 1. In the case that we compute the rigorous enclosure of R_1 without using Algorithms 3 and 4, Procedure 1, which uses Algorithm 7 instead of Algorithms 3 and 4, can be considered. This procedure requires $n \times n$ real matrices $\underline{Z_r}$, $\overline{Z_r}$, $\underline{Z_i}$ and $\overline{Z_i}$ which satisfy $\underline{Z_r} \le \overline{Z_r}$, $\underline{Z_i} \le \overline{Z_i}$ and $B\tilde{X} \in [\underline{Z_r}, \overline{Z_r}] + \sqrt{-1}[\underline{Z_i}, \overline{Z_i}]$. These matrices can be computed by using Algorithm 2 for B and $\tilde{X}$. Namely these matrices can be obtained in the process of computing Z_c and Z_r. From this and the fact that Z_c and Z_r have already been computed before executing Procedure 1, it can be seen that $\underline{Z_r}$, $\overline{Z_r}$, $\underline{Z_i}$ and $\overline{Z_i}$ have already been obtained before executing Procedure 1.

Procedure 1. *Let Y and R_1 be defined as in Theorem 1. Let also $n \times n$ real matrices $\underline{Z_r}$, $\overline{Z_r}$, $\underline{Z_i}$ and $\overline{Z_i}$ satisfy $\underline{Z_r} \le \overline{Z_r}$, $\underline{Z_i} \le \overline{Z_i}$ and $B\tilde{X} \in [\underline{Z_r}, \overline{Z_r}] + \sqrt{-1}[\underline{Z_i}, \overline{Z_i}]$. For A, $\tilde{X}$, $\tilde{D}$, Y, $\underline{Z_r}$, $\overline{Z_r}$, $\underline{Z_i}$ and $\overline{Z_i}$, this procedure computes $n \times n$ real matrices $\underline{W_r}$, $\overline{W_r}$, $\underline{W_i}$ and $\overline{W_i}$ which satisfy $\underline{W_r} \le \overline{W_r}$, $\underline{W_i} \le \overline{W_i}$ and $R_1 \in [\underline{W_r}, \overline{W_r}] + \sqrt{-1}[\underline{W_i}, \overline{W_i}]$ using Algorithm 7 instead of Algorithms 3 and 4.*

`function` $[\underline{W_r}, \overline{W_r}, \underline{W_i}, \overline{W_i}] =$ `encR1ci` $(A, \tilde{X}, \tilde{D}, Y, \underline{Z_r}, \overline{Z_r}, \underline{Z_i}, \overline{Z_i})$
$[\underline{C_r}, \overline{C_r}, \underline{C_i}, \overline{C_i}] =$ `cprod` $(A, \tilde{X})$;
$[\underline{V_r}, \overline{V_r}, \underline{V_i}, \overline{V_i}] =$ `ciprod'` $(\underline{Z_r}, \overline{Z_r}, \underline{Z_i}, \overline{Z_i}, \tilde{D})$; % $\mathcal{O}(n^2)$ `flops`
$\underline{U_r} = \mathrm{fl}_\nabla(\underline{C_r} - \overline{V_r})$;
$\underline{U_i} = \mathrm{fl}_\nabla(\underline{C_i} - \overline{V_i})$;
$\overline{U_r} = \mathrm{fl}_\triangle(\overline{C_r} - \underline{V_r})$;
$\overline{U_i} = \mathrm{fl}_\triangle(\overline{C_i} - \underline{V_i})$;
$[\underline{W_r}, \overline{W_r}, \underline{W_i}, \overline{W_i}] =$ `ciprod` $(Y, \underline{U_r}, \overline{U_r}, \underline{U_i}, \overline{U_i})$;

In Procedure 1, the function `ciprod'` denotes an analogous of the function `ciprod` and returns $n \times n$ real matrices $\underline{K_r}, \overline{K_r}, \underline{K_i}$ and $\overline{K_i}$ which satisfy $\underline{K_r} \le \overline{K_r}, \underline{K_i} \le \overline{K_i}$ and $([\underline{H_r}, \overline{H_r}] + \sqrt{-1}[\underline{H_i}, \overline{H_i}])J \subseteq [\underline{K_r}, \overline{K_r}] + \sqrt{-1}[\underline{K_i}, \overline{K_i}]$ for $n \times n$ real matrices $\underline{H_r}, \overline{H_r}, \underline{H_i}$ and $\overline{H_i}$ with $\underline{H_r} \le \overline{H_r}$ and $\underline{H_i} \le \overline{H_i}$, and $J \in \mathbb{C}^{n\times n}$.

In the case that we compute the rigorous enclosure of R_1 using Algorithms 3 and 4, Procedure 2 can be considered. Note that Procedure 2 requires Z_c and Z_r, and these matrices have already been obtained before executing Procedure 2.

Procedure 2. *Let Y and R_1 be defined as in Theorem 1. Let also Z_c and Z_r be defined as in Corollary 1. For $A, \tilde{X}, \tilde{D}, Y, Z_c$ and Z_r, this procedure computes $n \times n$ real matrices $\underline{W_r}, \overline{W_r}, \underline{W_i}$ and $\overline{W_i}$ which satisfy $\underline{W_r} \le \overline{W_r}, \underline{W_i} \le \overline{W_i}$ and $R_1 \in [\underline{W_r}, \overline{W_r}] + \sqrt{-1}[\underline{W_i}, \overline{W_i}]$ using Algorithms 3 and 4.*

`function` $[\underline{W_r}, \overline{W_r}, \underline{W_i}, \overline{W_i}] =$ `encR1ccr` $(A, \tilde{X}, \tilde{D}, Y, Z_c, Z_r)$
$[\underline{C_r}, \overline{C_r}, \underline{C_i}, \overline{C_i}] =$ `cprod` $(A, \tilde{X})$;
$[\underline{V_r}, \overline{V_r}, \underline{V_i}, \overline{V_i}] =$ `ccrprod'` $(Z_c, Z_r, \tilde{D})$; % $\mathcal{O}(n^2)$ `flops`
$\underline{U_r} = \mathrm{fl}_\nabla(\underline{C_r} - \overline{V_r})$;
$\underline{U_i} = \mathrm{fl}_\nabla(\underline{C_i} - \overline{V_i})$;
$\overline{U_r} = \mathrm{fl}_\Delta(\overline{C_r} - \underline{V_r})$;
$\overline{U_i} = \mathrm{fl}_\Delta(\overline{C_i} - \underline{V_i})$;
$[U_c, U_r] =$ `ccr` $(\underline{U_r}, \overline{U_r}, \underline{U_i}, \overline{U_i})$;
$[\underline{W_r}, \overline{W_r}, \underline{W_i}, \overline{W_i}] =$ `ccrprod` (Y, U_c, U_r);

In Procedure 2, the function `ccrprod'` denotes an analogous of the function `ccrprod` and returns $n \times n$ real matrices $\underline{K_r}, \overline{K_r}, \underline{K_i}$ and $\overline{K_i}$ which satisfy $\underline{K_r} \le \overline{K_r}, \underline{K_i} \le \overline{K_i}$ and $\langle H_c, H_r\rangle J \subseteq [\underline{K_r}, \overline{K_r}] + \sqrt{-1}[\underline{K_i}, \overline{K_i}]$ for $n \times n$ complex matrices H_c and J, and $H_r \in \mathbb{R}^{n\times n}$ with $H_r \ge 0$.

Computational costs of Procedures 1 and 2 are $40n^3$ flops and $34n^3$ flops, respectively. From this, it can be seen that the computation of the rigorous enclosure for R_1 can be accelerated by using Algorithms 3 and 4 rather than using Algorithm 7. On the other hand, Procedure 1 supplies narrower enclosure than that by Procedure 2, since Algorithm 3 overestimates the radii of matrix intervals.

Finally we describe the computational costs of Steps 1, 3, 5 and 6 in Algorithm 1. From the discussions in this section, Steps 1 and 5 require $16n^3$ and $34n^3$ flops, respectively. From Section 3, Step 3 requires $8n^3$ flops. Step 6 requires 2 flops.

Remark 6. If one needs only eigenvalues, i.e., eigenvectors are not calculated, then the computational cost may significantly be reduced. However the proposed method also need to calculate eigenvectors for enclosing eigenvalues. We will show the difference of computing time with or without calculating eigenvectors by numerical examples in Section 5.

5. Numerical examples

In this section, we report some numerical results to show the property of Algorithm 1 and performance of our implementation. We used a computer with Intel Xeon 2.66 GHz Dual CPU, 4.00 GB RAM, Windows Vista OS and MATLAB 7.5 with ATLAS and IEEE 754 double precision for all computations.

We used the MATLAB functions `eig` and `inv` to obtain all approximate eigenvalues and eigenvectors, and to execute Step 2 in Algorithm 1, respectively. Then Step 2 requires $8n^3$ flops. From this and Section 4, computational cost of Algorithm 1 becomes $66n^3$ flops.

Let ε be defined as in Algorithm 1. We introduce the method in [3] for comparison. In [3], it is written that their method is applicable if A is Hermitian and B is Hermitian positive definite. On the other hand, their method can be expanded to be applicable even if A is not Hermitian and/or B is not Hermitian positive definite. Then the expanded method (hereafter we call this method as the method in [3] simply) supplies approximate eigenvalues $\tilde{\mu}_1, \ldots, \tilde{\mu}_n$ and rigorous error bounds $\zeta_1, \ldots, \zeta_n$ such that

$$\lambda \in \bigcup_{i=1}^{n} \{z \in \mathbb{C} \,||z - \tilde{\mu}_i| \le \zeta_i\}$$

for all λ when $\tilde{X}$ is given. For fairness, we used Algorithms 3 and 4 rather than Algorithm 7 in the implementation of the method in [3]. Then the computational cost of the method in [3] becomes $148n^3$ flops. To see the mean values of $\zeta_1, \ldots, \zeta_n$, mean ζ_i denotes

$$\text{mean } \zeta_i := \mathrm{fl}_\square\left(\sum_{i=1}^{n} \zeta_i \Big/ n\right).$$

Let $t_\lambda, t_{\lambda x}, t_\varepsilon$ and t_ζ be the computing time (sec) for calculating $\tilde{D}$, calculating $\tilde{D}$ and $\tilde{X}$, Algorithm 1 and the method in [3], respectively. Moreover define $\kappa(Q) := \|Q\|_2\|Q^{-1}\|_2$ for a nonsingular matrix Q.

Table 1
Obtained error bounds and computing times (s) in Example 1.

n	ε	min ζ_i	mean ζ_i	max ζ_i	t_λ	$t_{\lambda x}$	t_ε	t_ζ
500	6.80e−08	1.02e−06	2.01e−06	6.45e−06	7.16	12.3	3.99	8.38
1000	2.62e−07	6.34e−06	1.60e−05	5.67e−05	61.3	100	29.7	63.6
1500	3.25e−06	6.51e−05	1.67e−04	6.11e−04	197	318	100	215
2000	1.92e−06	7.76e−05	2.08e−04	8.44e−04	512	821	236	507

Table 2
Obtained error bounds in Example 2.

cond	ε	min ζ_i	mean ζ_i	max ζ_i
1e+04	1.15e−06	2.04e−05	7.30e−05	2.58e−04
1e+06	1.02e−04	1.69e−03	8.10e−03	1.80e−02
1e+08	2.42e−02	4.35e−01	1.95e−00	4.92e−00
1e+10	1.14e−00	1.86e+01	1.06e+02	1.73e+02
1e+12	8.39e+01	1.47e+03	9.60e+03	1.92e+04

5.1. Example 1

In this example, we observe the sizes of error bounds and computing times for large n when $\kappa(A)$ and $\kappa(B)$ are small. Consider the case that $n \times n$ complex matrices A and B are generated by

```
A = randn(n) + i*randn(n);
B = randn(n) + i*randn(n);
```

on MATLAB. Here the function `randn` generates a random real matrix whose elements are uniformly distributed in $[-1, 1]$. Algorithm 1 verified that B is nonsingular. For various n, Table 1 displays ε, $\min_{1\le i\le n}\zeta_i$, mean ζ_i, $\max_{1\le i\le n}\zeta_i$, t_λ, $t_{\lambda x}$, t_ε and t_ζ.

From Table 1, we can confirm that Algorithm 1 supplied smaller error bounds than those by the method in [3] in this example. Moreover it can be seen that t_ε was approximately a half of t_ζ. Namely Algorithm 1 was approximately twice faster than the method in [3] in this example. This identifies the fact that computational costs of Algorithm 1 and the method in [3] are $66n^3$ flops and $148n^3$ flops, respectively. We can also confirm that t_ε was smaller than t_λ. Namely Algorithm 1 was faster than the computation of $\tilde{D}$ in this example.

One may be interested in enclosing *a few specified* eigenvalues. For instance, consider the case of $n = 1000$ and enclosing six eigenvalues in this example. In this case, we can apply the INTLAB [19] function `VerifyEig`. `VerifyEig` is designed to enclose one eigenvalue and eigenvector when $\tilde{\lambda}_i$ and $\tilde{x}^{(i)}$ are given, and can be significantly accelerated if more than one eigenvalue is to be included. When we applied `VerifyEig` for $(\tilde{\lambda}_j, \tilde{x}^{(j)})$, $j \in \{1, 2, 3, 998, 999, 1000\}$, the obtained error bounds for $\tilde{\lambda}_j$ were approximately 10^3 times as small as ε. The computing time for `VerifyEig` was 89.8 s for all j. As shown in Table 1, Algorithm 1 required 29.7 s to enclose *all eigenvalues*. From these it can be seen that Algorithm 1 is faster than `VerifyEig` although `VerifyEig` supplies smaller error bounds than ε in this case.

5.2. Example 2

In this example, we observe how the sizes of error bounds change when $\kappa(A)$ increases. Consider the case that 500×500 complex matrices A and B are generated by the following MATLAB code:

```
cond10 = log10(cond); % cond: anticipated condition number of A
D = diag(logspace(0,cond10,500));
[U,S,V] = svd(randn(500) + i*randn(500));
A = U*D*V';
B = randn(500) + i*randn(500);
```

We used the MATLAB function `svd` for generating a random (approximately) unitary matrix. Then it holds approximately that $\kappa(A) \approx$ `cond`. Algorithm 1 verified that B is nonsingular. For various `cond`, Table 2 displays ε, $\min_{1\le i\le n}\zeta_i$, mean ζ_i and $\max_{1\le i\le n}\zeta_i$. Tendencies with respect to computing times were similar to those in Example 1.

From Table 2, we can confirm that error bounds increase as $\kappa(A)$ increases. Moreover it can be seen that Algorithm 1 supplied smaller error bounds than those by the method in [3] also in this example.

In the case of `cond` $=$ 1e + 08, we applied `VerifyEig` for $(\tilde{\lambda}_j, \tilde{x}^{(j)}), j \in \{1, 2, 3, 498, 499, 500\}$. Then the obtained error bounds for $\tilde{\lambda}_j$ were approximately 10^2 times as small as ε. The computing time for `VerifyEig` was 13.4 s for all j. Against this, Algorithm 1 required 3.98 s to enclose *all eigenvalues*. From these we can confirm the similar tendencies to those in Example 1 regarding the relation between `VerifyEig` and Algorithm 1.

Table 3
Obtained error bounds in Example 3.

cond	ε	min ζ_i	mean ζ_i	max ζ_i
1e+04	7.64e−07	3.96e−07	1.15e−05	6.63e−05
1e+06	3.55e−05	3.32e−07	3.51e−04	2.83e−03
1e+08	1.39e−03	3.27e−07	1.36e−02	1.16e−01
1e+10	3.11e−01	3.50e−07	8.38e−01	2.23e+01
1e+12	–	3.21e−07	4.50e+01	5.28e+02

Table 4
Obtained error bounds and computing times (s) in Example 4.

matrices	ε	min ζ_i	mean ζ_i	max ζ_i	t_λ	$t_{\lambda x}$	t_ε	t_ζ
BFW62	5.49e−08	1.02e−08	3.90e−08	1.06e−07	0.00789	0.0107	0.0125	0.0238
BFW398	1.43e−05	8.77e−07	7.19e−06	3.83e−05	1.15	1.97	2.08	4.04
BFW782	9.75e−04	4.13e−06	4.18e−05	3.50e−04	9.33	17.6	14.3	28.6
DGW961	1.45e−02	1.58e−06	2.73e−03	2.97e−02	37.3	55.6	27.0	57.7
LUND	3.53e−07	3.40e−07	4.57e−06	1.22e−05	0.0192	0.0353	0.0962	0.204
RBS480	3.89e−09	1.39e−08	2.37e−07	9.57e−07	2.36	4.29	3.54	6.93

5.3. Example 3

In this example, we observe how the sizes of error bounds change when $\kappa(B)$ increases. Consider the case that 500×500 complex matrices A and B are generated by the following MATLAB code:

```
A = randn(500) + i*randn(500);
cond10 = log10(cond); % cond: anticipated condition number of B
D = diag(logspace(0,cond10,500));
[U,S,V] = svd(randn(500) + i*randn(500));
B = U*D*V';
```

Then it holds approximately that $\kappa(B) \approx$ cond. Algorithm 1 verified that B is nonsingular except the case that cond $= 1e + 12$. For various cond, Table 3 displays the similar quantities to Table 2. In Table 3, the notation "–" means that ε_2 became larger than 1 so that Algorithm 1 failed, where ε_2 is defined as in Algorithm 1. Tendencies with respect to computing times were similar to those in Example 1.

From Table 3, we can confirm that min ζ_i scarcely change even though $\kappa(B)$ increase. Moreover we can confirm that Algorithm 1 failed when cond $= 1e + 12$, although the method in [3] succeeded. From these it can be seen that the method in [3] is robuster than Algorithm 1 for B with large $\kappa(B)$.

In the case of cond $= 1e+08$, we applied VerifyEig for $(\tilde{\lambda}_j, \tilde{x}^{(j)}), j \in \{1, 2, 3, 498, 499, 500\}$. Then enclosure succeeded for $\tilde{\lambda}_{498}$, $\tilde{\lambda}_{499}$ and $\tilde{\lambda}_{500}$, and failed for $\tilde{\lambda}_1$, $\tilde{\lambda}_2$ and $\tilde{\lambda}_3$. The obtained error bounds for $\tilde{\lambda}_{498}$, $\tilde{\lambda}_{499}$ and $\tilde{\lambda}_{500}$ were approximately 10^{15} times as small as ε. The computing time for VerifyEig was 18.0 s for all j. Against this, Algorithm 1 required 3.98 s to enclose *all eigenvalues*. From these it can be seen that Algorithm 1 is faster and robuster than VerifyEig although VerifyEig supplied much smaller error bounds for $\tilde{\lambda}_{498}$, $\tilde{\lambda}_{499}$ and $\tilde{\lambda}_{500}$ than ε in this case.

5.4. Example 4

In this example, we observe the sizes of error bounds and computing times for matrices in Matrix Market [10]. Algorithm 1 verified that B is nonsingular. For various matrices, Table 4 displays the similar quantities to Table 1.

From Table 4, we can confirm that Algorithm 1 supplied comparable error bounds to those by the method in [3] in this example. Moreover it can be seen that the relations between t_ε and t_ζ were similar to those in Example 1.

For DGW961, we applied VerifyEig for $(\tilde{\lambda}_j, \tilde{x}^{(j)}), j \in \{1, 2, 3, 959, 960, 961\}$. Then enclosure succeeded for $\tilde{\lambda}_1$, $\tilde{\lambda}_2$ and $\tilde{\lambda}_3$, and failed for $\tilde{\lambda}_{959}$, $\tilde{\lambda}_{960}$ and $\tilde{\lambda}_{961}$. The obtained error bounds for $\tilde{\lambda}_1$, $\tilde{\lambda}_2$ and $\tilde{\lambda}_3$ were approximately 10^3 times as small as ε. The computing time for VerifyEig was 105 s for all j. As shown in Table 4, Algorithm 1 required 27.0 s to enclose *all eigenvalues*. From these we can confirm the similar tendencies to those in Example 3 as regards the relation between VerifyEig and Algorithm 1.

5.5. Example 5

In this example, we observe the property of Algorithm 1 and the method in [3] when there exist multiple eigenvalues. Consider the case that A and B are defined as follows:

$$A = \begin{pmatrix} -30 & 6 & 9 \\ -30 & 6 & 9 \\ -170 & 34 & 51 \end{pmatrix}, \qquad B = \begin{pmatrix} 2 & -1 & 5 \\ 1 & 0 & 2 \\ 1 & 5 & -4 \end{pmatrix}.$$

Table 5
Approximate eigenvalues and obtained error bounds in Example 5.

$\tilde{\lambda}_1$	$\tilde{\lambda}_2$	$\tilde{\lambda}_3$	ε
−2.08e−13	1.00e−00	6.38e−14	8.32e−12

$\tilde{\mu}_1$	$\tilde{\mu}_2$	$\tilde{\mu}_3$	ζ_1	ζ_2	ζ_3
4.42e−13	1.00e−00	−1.59e−14	2.39e−11	1.64e−11	9.01e−12

In this case, $\lambda \in \{0, 1\}$ and the algebraic multiplicity of $\lambda = 0$ is two. The geometric multiplicity of $\lambda = 0$ is also two. Algorithm 1 verified that B is nonsingular. Table 5 displays ε, $\tilde{\lambda}_i$, $\tilde{\mu}_i$ and ζ_i for $i \in \{1, 2, 3\}$.

From Table 5, we can confirm that Algorithm 1 and the method in [3] could enclose all eigenvalues even if there exist multiple eigenvalues, although these methods cannot check whether there exist multiple eigenvalues or closely clustered eigenvalues. Moreover it can be seen that Algorithm 1 supplied a smaller error bound than those by the method in [3] in this example.

We applied `VerifyEig` for $(\tilde{\lambda}_j, \tilde{x}^{(j)})$, $j \in \{1, 2, 3\}$. Then enclosure succeeded for $\tilde{\lambda}_2$, and failed for $\tilde{\lambda}_1$ and $\tilde{\lambda}_3$. Namely enclosure of $\lambda = 0$ failed. The obtained error bound for $\tilde{\lambda}_2$ was approximately equal to ε.

6. Application to polynomial eigenvalue problems

As an application of the proposed method, in this section, we sketch an efficient method of enclosing all eigenvalues in the polynomial eigenvalue problems (2).

The problems (2) are equivalent to the following generalized eigenvalue problems

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{B}\mathbf{x}, \quad \mathbf{A}, \mathbf{B} \in \mathbb{C}^{mn\times mn},\ \lambda \in \mathbb{C},\ \mathbf{x} \in \mathbb{C}^{mn} \tag{9}$$

where

$$\mathbf{A} := \begin{pmatrix} 0 & I & 0 & \cdots & 0 \\ 0 & 0 & I & \ddots & \vdots \\ \vdots & & & \ddots & 0 \\ & & & & I \\ -A_0 & -A_1 & -A_2 & \cdots & -A_{m-1} \end{pmatrix}, \quad \mathbf{B} := \begin{pmatrix} I & & & & \\ & I & & & \\ & & \ddots & & \\ & & & I & \\ & & & & A_m \end{pmatrix},$$

$$\mathbf{x} := (x^T\ \lambda x^T\ \cdots\ \lambda^{m-1}x^T)^T.$$

From the assumption that A_m is nonsingular, $\mathbf{B}$ is also nonsingular. Therefore all eigenvalues in (2) can be enclosed by applying the proposed method to (9).

Assume that as a result of numerical computation, we have $\tilde{\Lambda} \in \mathbb{C}^{mn}$ and $\tilde{\mathcal{X}} \in \mathbb{C}^{n\times mn}$ such that

$$(\tilde{\lambda}_k^m A_m + \cdots + \tilde{\lambda}_k A_1 + A_0)\tilde{x}^{(k)} \approx 0, \quad k = 1, \ldots, mn$$

where $\tilde{\lambda}_k$ and $\tilde{x}^{(k)}$ denote the kth element of $\tilde{\Lambda}$ and the kth column of $\tilde{\mathcal{X}}$, respectively. Moreover let $mn \times mn$ complex matrices $\tilde{\mathbf{D}}$ and $\tilde{\mathbf{X}}$ be defined as follows:

$$\tilde{\mathbf{D}} := \mathrm{diag}(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_{mn}), \quad \tilde{\mathbf{X}} := \begin{pmatrix} \tilde{x}^{(1)} & \cdots & \tilde{x}^{(mn)} \\ \mathrm{fl}_\square(\tilde{\lambda}_1\tilde{x}^{(1)}) & \cdots & \mathrm{fl}_\square(\tilde{\lambda}_{mn}\tilde{x}^{(mn)}) \\ \vdots & & \vdots \\ \mathrm{fl}_\square(\tilde{\lambda}_1^{m-1}\tilde{x}^{(1)}) & \cdots & \mathrm{fl}_\square(\tilde{\lambda}_{mn}^{m-1}\tilde{x}^{(mn)}) \end{pmatrix}.$$

Then it holds approximately that $\mathbf{A}\tilde{\mathbf{X}} \approx \mathbf{B}\tilde{\mathbf{X}}\tilde{\mathbf{D}}$. Therefore by giving $\mathbf{A}$, $\mathbf{B}$, $\tilde{\mathbf{D}}$ and $\tilde{\mathbf{X}}$ to Algorithm 1, all eigenvalues in (2) can be enclosed. Note that we can reduce the computational cost of Algorithm 1 by utilizing the sparsity of $\mathbf{A}$ and $\mathbf{B}$.

7. Conclusion

In this paper, a theorem for enclosing all eigenvalues in generalized eigenvalue problems (1) was presented. This theorem is applicable even if A is not Hermitian and/or B is not Hermitian positive definite. Moreover a theorem for accelerating the enclosure was presented. Based on these theorems, a fast method of enclosing all eigenvalues was proposed. As an application of the proposed method, an efficient method of enclosing all eigenvalues in polynomial eigenvalue problems (2) was also sketched.

By modifying the proposed method slightly, enclosure for all eigenvalues in (1) where A and/or B are complex interval matrices is also possible.

Acknowledgements

The author wish to thank Prof. Takeshi Ogita at the Tokyo Woman's Christian University, Dr. Katsuhisa Ozaki at the Waseda University, and Prof. Michael Plum at the University of Karlsruhe for fruitful discussions. The author also wish to thank Prof. Christian Wieners at the University of Karlsruhe for suggesting the residual term R_1 in Theorem 1. This research was partially supported by Grant-in-Aid for Young Scientists (B) (19760055, 2007–2010) from the Ministry of Education, Science, Sports and Culture of Japan.

References

[1] H. Behnke, Inclusion of eigenvalues of general eigenvalue problems for matrices, in: U. Kulisch, H.J. Stetter (Eds.), Scientific Computation with Automatic Result Verification, Computing 6 (Suppl) (1988), 69–78.
[2] H. Behnke, The calculation of guaranteed bounds for eigenvalues using complementary variational principles, Computing 47 (1991) 11–27.
[3] K. Maruyama, T. Ogita, Y. Nakaya, S. Oishi, Numerical inclusion method for all eigenvalues of real symmetric definite generalized eigenvalue problem, IEICE Trans. J87–A(8) (2004) 1111–1119 (in Japanese).
[4] S. Miyajima, T. Ogita, S.M. Rump, S. Oishi, Fast verification for all eigenpairs in symmetric positive definite generalized eigenvalue problems, Reliab. Comput. (in press).
[5] S.M. Rump, Guaranteed inclusions for the complex generalized eigenproblem, Computing 42 (1989) 225–238.
[6] S.M. Rump, Computational error bounds for multiple or nearly multiple eigenvalues, Linear Algebra Appl. 324 (2001) 209–226.
[7] Y. Watanabe, N. Yamamoto, M.T. Nakao, Verification methods of generalized eigenvalue problems and its applications, Trans. JSIAM 9 (3) (1999) 137–150 (in Japanese).
[8] N. Yamamoto, A simple method for error bounds of eigenvalues of symmetric matrices, Linear Algebra Appl. 324 (2001) 227–234.
[9] B.N. Parlett, The Symmetric Eigenvalue Problem, in: Classics in Applied Mathematics, vol. 20, SIAM Publications, Philadelphia, 1997.
[10] Matrix Market (Last change: 10 May 2007), http://math.nist.gov/MatrixMarket/.
[11] S.W. Stewart, J.G. Sun, Matrix Perturbation Theory, Academic Press, New York, 1990.
[12] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.
[13] R. Brent, C. Percival, P. Zimmermann, Error bounds on complex floating-point multiplication, Math. Comp. 76 (259) (2007) 1469–1481.
[14] N.J. Higham, Accuracy and Stability of Numerical Algorithms, second ed., SIAM Publications, Philadelphia, 2002.
[15] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., The Johns Hopkins University Press, Baltimore and London, 1996.
[16] F. Bauer, C. Fike, Norms and exclusion theorems, Numer. Math. 2 (1960) 137–141.
[17] H. Arndt, On the interval systems $[x] = [A][x] + [b]$ and the powers of interval matrices in complex interval arithmetics, Reliab. Comput. 13 (2007) 245–259.
[18] S. Oishi, Fast enclosure of matrix eigenvalues and singular values via rounding mode controlled computation, Linear Algebra Appl. 324 (2001) 133–146.
[19] S.M. Rump, INTLAB - INTerval LABoratory, in: T. Csendes (Ed.), Developments in Reliable Computing, Kluwer Academic Publishers, Dordrecht, 1999, pp. 77–107.