FAST VERIFICATION FOR THE PERRON PAIR OF AN IRREDUCIBLE NONNEGATIVE MATRIX

SHINYA MIYAJIMA

Abstract. Fast algorithms are proposed for calculating error bounds for a numerically computed Perron root and vector of an irreducible nonnegative matrix. Emphasis is put on the computational efficiency of these algorithms. Error bounds for the root and vector are based on the Collatz–Wielandt theorem, and estimating a solution of a linear system whose coefficient matrix is an $M$-matrix, respectively. We introduce a technique for obtaining better error bounds. Numerical results show properties of the algorithms.

Key words. Perron pair, Irreducible nonnegative matrix, $M$-matrix, Verified computation.

AMS subject classifications. 15A18, 15B48, 65F15, 65G20.

1. Introduction. For $v \in \mathbb{R}^n$ and $M \in \mathbb{R}^{n \times n}$, let $v_i$ and $M_{ij}$ be the $i$th component and $(i,j)$ element of $v$ and $M$, respectively. For $v, w \in \mathbb{R}^n$ and $M, N \in \mathbb{R}^{n \times n}$, let $v \leq w$ and $M \leq N$ denote $v_i \leq w_i$ for all $i$ and $M_{ij} \leq N_{ij}$ for all $i, j$, respectively. We say $M$ is nonnegative if $M \geq 0$. We also say $A \in \mathbb{R}^{n \times n}$ is reducible if and only if for some permutation matrix $P$, the matrix $P^T A P$ is block upper triangular. If $A$ is not reducible, we say $A$ is irreducible. The following, called the Perron–Frobenius theorem, states a well-known property of irreducible nonnegative matrices.

THEOREM 1 (Frobenius [1, 2]). Let $A \in \mathbb{R}^{n \times n}$ be irreducible and nonnegative, $\rho(A)$ be the spectral radius of $A$, and $\lambda_i$, $i = 1, \ldots, n$ be eigenvalues of $A$ such that $|\lambda_1| \geq \cdots \geq |\lambda_n|$. Then, the following are true:

(a) $\lambda_1 = \rho(A) > 0$.
(b) There exists an eigenvector $x^*$ corresponding to $\lambda_1$ such that $x^* > 0$.
(c) If $A \geq B \geq 0$ for $B \in \mathbb{R}^{n \times n}$, then $\lambda_1 \geq \rho(B)$ holds, with equality holding if and only if $A = B$.
(d) The algebraic multiplicity of $\lambda_1$ is one.

The positive eigenvalue $\lambda_1$ and corresponding positive eigenvector $x^*$ are called the Perron root and vector, respectively, and we call the pair $(\lambda_1, x^*)$ the Perron pair. In this paper, we are concerned with the accuracy of a numerically computed Perron pair of a nonnegative irreducible matrix.

Perron pairs have many applications. For example, the Perron pair is required in finite-state Markov chains such as branching processes [3] and Markov rewards processes with exponential utility [4]. In information theory, Stanczak and Boche [5] studied the relationship between Kullback–Leibler distance and a Perron root and used it to develop power control algorithms for providing a desired quality of service. Mathematical analysis with respect to a Perron root and vector has been extensively studied (see, e.g., [6, 7, 8]).
The work presented herein addresses the problem of verified computations for \( \lambda_1 \) and \( x^* \), specifically, computing rigorous bounds for \( \lambda_1 \) and \( x^* \) based on approximations. There are many verification methods for eigenvalues and eigenvectors. For all eigenvalues, see [9, 10, 11, 12, 13, 14, 15]. For all eigenpairs, see [16, 17, 18]. For a few specified eigenvalues, see, e.g., [19, 20, 21, 22, 23, 24, 25]. For a few specified eigenvectors, see [26]. For computing bounds for \( \lambda_1 \) and \( x^* \), we can apply verification methods for a few specified eigenpairs (e.g., [27, 28, 29, 30, 31, 32, 33, 34]), which require \( O(n^3) \) operations in general. Only Nagato and Ishii [23] have proposed a verification algorithm designed specifically for the Perron root of a nonnegative matrix. On the other hand, it does not compute a bound for the Perron vector.

The purpose of the present paper is to propose two verification algorithms for the Perron pair of an irreducible nonnegative matrix. Emphasis is put on the computational efficiency of these algorithms. Let \( \tilde{\lambda} \) and \( \tilde{x} \) be numerical results for \( \lambda_1 \) and \( x^* \), respectively. We show that a subvector of \( x^* - \tilde{x} \) can be regarded as the solution of a linear system with \( M \)-matrix (see Section 4). Verification algorithms for the solution of a linear system with \( M \)-matrix have already been proposed (e.g., [35, 36, 37, 38]). However, herein is reported the first attempt at a verification algorithm for an eigenvector that exploits the \( M \)-matrix property. We introduce a technique for improving the accuracy of \( \tilde{\lambda} \) and \( \tilde{x} \), reducing error bounds (see Sections 3, 5, and 6).

This paper is organized as follows: Section 2 introduces the notation and two necessary theorems used in this paper. Section 3 discusses methods for computing \( \tilde{\lambda} \) and \( \tilde{x} \) and improving their accuracy. Section 4 establishes a theory for computing error bounds and proposes the first algorithm. Section 5 proposes a second algorithm that uses the improved approximations and gives better error bounds. Section 6 reports numerical results. Finally, Section 7 highlights future work.

2. Preliminaries. For \( M \in \mathbb{R}^{n \times n} \), let \( \rho(M) \) be the spectral radius of \( M \), and \( |M| := (|M|_{ij}) \). Denote the principal submatrix of \( M \) consisting of rows and columns in \( \mu \subseteq \{1, \ldots, n\} \) by \( M[\mu] \). Analogously, let \( x[\mu] \) be the subvector of \( x \in \mathbb{R}^n \) which consists of components in \( \mu \). For \( v \in \mathbb{R}^n \), let \( \min(v) := \min_i(v_i) \), \( \max(v) := \max_i(v_i) \), and \( |v| := (|v_1|, \ldots, |v_n|) \). Let \( \mathbb{R}_{\geq 0} := [0, \infty) \), \( \mathbb{R}_{> 0} := (0, \infty) \), \( \mathbb{R}_{\geq 0}^n := \{v \in \mathbb{R}^n : v \geq 0\} \), \( \mathbb{R}_{> 0}^n := \{v \in \mathbb{R}^n : v > 0\} \), \( \mathbb{R}_{\geq 0}^{n \times n} := \{M \in \mathbb{R}^{n \times n} : M \geq 0\} \), and \( \mathbb{F} \) be the set of all floating-point real numbers. Let \( I \) be the identity matrix with compatible size, \( \circ \) and \( ./ \) be element-wise multiplication and division, respectively, and \( \mathbb{I} := [1, \ldots, 1]^T \) with proper dimension. For \( c \in \mathbb{R}^n \) and \( r \in \mathbb{R}_{\geq 0}^n \), let \( (c, r) \) denote the real interval vector whose midpoint and radius are \( c \) and \( r \), respectively. We write \( \mathbb{f}() \), \( \mathbb{fl}_\Delta() \), and \( \mathbb{fl}_\Omega() \) to denote results of floating-point computations, where all the operations inside the parentheses are executed using the to-the-nearest, toward \(+\infty\), and toward \(-\infty\) rounding modes, respectively. Let \( \text{realmin} \) be the smallest positive normalized floating-point number (specifically \( \text{realmin} = 2^{-1022} \)) in IEEE 754 double precision. A real square matrix \( A \) is called a \( Z \)-matrix if \( A_{ij} \leq 0 \) for all \( i \neq j \). It is clear that any \( Z \)-matrix can be written as \( \mu I - B \) with \( B \geq 0 \). A \( Z \)-matrix \( \mu I - B \) is called an \( M \)-matrix if \( \mu > \rho(B) \). We say \( A \) has the “\( M \)-property” if \( A \) is an \( M \)-matrix.

In the proof of Theorem 5, we will apply Theorem 2, which is called the Collatz–Wielandt theorem.

**Theorem 2** (Collatz [39] and Wielandt [40]). Let \( A \in \mathbb{R}_{\geq 0}^{n \times n} \) and \( x \in \mathbb{R}_{> 0}^n \). It then follows that \( \min((Ax)./x) \leq \rho(A) \leq \max((Ax)./x) \).

In Section 4, we will use the following properties of an \( M \)-matrix.

**Theorem 3** (Fiedler and Pták [41]). For a \( Z \)-matrix \( A \), the following are equivalent:
1. $A$ is an $M$-matrix.
2. $A$ is nonsingular and $A^{-1} \geq 0$.
3. There exists $v \in \mathbb{R}^n_{\geq 0}$ satisfying $Av > 0$.

3. Computing an approximation and improving its accuracy. Let $\lambda_1$, $x^*$, $\hat{\lambda}$, and $\hat{x}$ be as in Section 1. We are interested in verified bounds for $(\lambda_1, x^*)$ based on $(\hat{\lambda}, \hat{x})$; however, we want to add some remarks on how to obtain $(\hat{\lambda}, \hat{x})$. In Section 3.1, we thus discuss methods for computing $(\hat{\lambda}, \hat{x})$ and present a procedure based on the discussion. Section 3.2 introduces a method for improving its accuracy.

3.1. Computing an approximation. Let $\lambda_i$ be as in Theorem 1, i.e., $|\lambda_1| \geq \cdots \geq |\lambda_n|$. If the cyclic index of $A$ is one, then $|\lambda_2|/\rho(A) < 1$ follows, so that we can compute $(\hat{\lambda}, \hat{x})$ via the power method $x^{(k+1)} = Ax^{(k)}$, $k = 0, 1, \ldots$, where $x^{(0)} \in \mathbb{R}_{\geq 0}^n$ is an initial guess. On the other hand, if the cyclic index of $A$ is two or more, then $|\lambda_2|/\rho(A) = 1$, which means the power method does not work. One may consider that the issue can be overcome by introducing a shift. However, introducing the shift is not generally working.

Therefore, using the power method only seems to be problematic. A stable way for computing eigenvalues and eigenvectors in MATLAB is to call the routine eig. On the other hand, eig computes all eigenvalues and eigenvectors, requiring large CPU times when $n$ is large, though we need only the Perron pair. In 2007, an iterative algorithm for computing $(\hat{\lambda}, \hat{x})$ has been proposed in [42]. Although it requires $O(n^3)$ operations per iteration, it is reported in [42, Section 3.5] that the iteration converges even in the problem where the power method does not converge. However, there is a case where the iteration does not converge (see Example 5 in Section 6).

We thus propose the following procedure:

1. Execute the power method. If it converged, then terminate.
2. Execute the iteration in [42]. If it converged, then terminate.
3. Call eig.

Let $\hat{x} \in \mathbb{R}^n$ be a numerical result for an eigenvector corresponding to $\rho(A)$. Then, $\hat{x} < 0$ may occur. In such a case, we must put $\hat{x} = -\hat{x}$.

3.2. Improving the accuracy. Let $k \in \{1, \ldots, n\}$ satisfy $\hat{x}_k = \max(\hat{x})$, and $x^* \in \mathbb{R}_{\geq 0}^n$ be the Perron vector of $A$ such that $x^*_k = \hat{x}_k$. Hereafter, in this paper, let $e \in \mathbb{R}^n$ be the $k$th column of $I$ and $U \in \mathbb{R}^{(n-1) \times n}$ be $I$ without the $k$th row. It is possible to improve the accuracy of $\hat{\lambda}$ and $\hat{x}$ via a Newton method applied to a nonlinear system derived from $(\rho(A)I - A)x^* = 0$. This technique is a subset of [34, Algorithm 2.1]. Since $x^*_k = \hat{x}_k$, we correct the unknown quantities $UX^*$ and $\rho(A)$ into $y \in \mathbb{R}^n$, i.e., we consider finding $y$ with $y_k = \rho(A)$ and $Uy = UX^*$. Let $\tilde{y} \in \mathbb{R}_{\geq 0}^n$ satisfy $U\tilde{y} = UX$ and $\tilde{y}_k = \hat{\lambda}$, and $r := A\tilde{x} - \hat{\lambda}\hat{x}$. Suppose that $\tilde{x}e^T - (A - \hat{\lambda}I)U^TU$ is nonsingular. From the analysis in [34], a correction term for $\tilde{y}$ obtained by the Newton method is $(\tilde{x}e^T - (A - \hat{\lambda}I)U^TU)^{-1}r$, which is the solution $z^* \in \mathbb{R}^n$ to the linear system

\begin{equation}
(\tilde{x}e^T - (A - \hat{\lambda}I)U^TU)z^* = r.
\end{equation}

We cannot assert that $\tilde{x}e^T - (A - \hat{\lambda}I)U^TU$ is nonsingular. As Theorem 4 implies, however, we can expect nonsingularity if $\hat{\lambda}$ and $\hat{x}$ are not too far from $\rho(A)$ and $x^*$, respectively.

**Theorem 4.** Let $A \in \mathbb{R}_{\geq 0}^{n \times n}$ be irreducible and $x^*$ be a Perron vector of $A$. Then, $x^*e^T - (A - \rho(A)I)U^TU$ is nonsingular for any $k \in \{1, \ldots, n\}$. 

Proof. Let \( C := A - \rho(A)I \) and \( C_{i,j} \) be the \((i,j)\) cofactor of \( C \). From Theorem 1 (d), we have rank\((C) = n - 1\), so that rank\((\text{adj}(C)) = 1\). Therefore, there exist \( \alpha \in \mathbb{R} \setminus \{0\} \) and a Perron vector \( v \in \mathbb{R}^n_{>0} \) of \( A^T \) such that \( \text{adj}(C) = x^*(av)^T \), which gives \( \det(x^*e^T - CU^TU) = (-1)^{n-1}\sum_{i=1}^n x_i^2C_{i,j} = (-1)^{n-1}\alpha x_i^2(v^T x^*) \neq 0 \).

We numerically solve (1) and obtain an approximate solution \( z \in \mathbb{R}^n \). Then, we can expect \( \tilde{\lambda} + z_k \) and \( \tilde{x} + U^T U z \) are more accurate than \( \tilde{\lambda} \) and \( \tilde{x} \), respectively. In order to obtain effective \( z \), it is necessary to compute \( r \) in extended precision.

4. Verification theory. Let \( \tilde{\lambda} \) and \( \tilde{x} \) be as in Section 1. We present Theorem 5, which gives an error bound for \( \tilde{\lambda} \).

**Theorem 5.** Let \( A \in \mathbb{R}^{n \times n}_{\geq 0} \), \( \tilde{x} \in \mathbb{R}^n_{>0} \), \( \tilde{\lambda} \in \mathbb{R}_{>0} \), \( r := A\tilde{x} - \lambda\tilde{x} \), and \( \varepsilon := \max\{|r_i|/|\tilde{x}_i|\} \). It then follows that \(|\tilde{\lambda} - \rho(A)| \leq \varepsilon\).

Proof. Define \( y := (A\tilde{x})/\tilde{x} \). Then, \( r_i = (y_i - \tilde{\lambda})\tilde{x}_i \), so that \( |y_i - \tilde{\lambda}| \leq \max\{|r_j|/|\tilde{x}_j|\} = \varepsilon \) for all \( i \). Theorem 2 moreover shows \( \min(y) \leq \rho(A) \leq \max(y) \). Hence, \(|\tilde{\lambda} - \rho(A)| \leq \max\{|y_j - \tilde{\lambda}|\} \leq \varepsilon \).

**Remark 6.** Let \( \lambda := \max(y) \), \( \Delta := \min(y) \), \( \omega := (\lambda + \Delta)/2 \), \( \delta := (\lambda - \Delta)/2 \), \( \lambda_\Delta := \text{fl}_\Delta((\lambda + \Delta)/2) \), \( \lambda_\varphi := \text{fl}_\varphi((\lambda + \Delta)/2) \), \( \delta_\Delta := \text{fl}_\Delta(\lambda_\Delta - \Delta) \), and \( \delta_\varphi := \text{fl}_\varphi(\lambda - \lambda_\varphi) \). From Theorem 2, we have \( \rho(A) \in (\lambda_\Delta, \delta_\Delta) \) and \( \rho(A) \in (\lambda_\varphi, \delta_\varphi) \), so that

\[
\rho(A) \in (\lambda^*, \delta^*), \quad \text{where } \lambda^* := \begin{cases} \lambda_\Delta & \text{if } \delta_\Delta \leq \delta_\varphi, \\ \lambda_\varphi & \text{otherwise} \end{cases} \quad \text{and } \delta^* := \min(\delta_\Delta, \delta_\varphi).
\]

This is an alternative way for enclosing \( \rho(A) \). Moreover, \( \delta^* \leq \varepsilon \) follows if \( \Delta/2 \leq \lambda - 2\lambda \); otherwise the enclosure is wide. Its proof given in the following is due to the reviewer. If the assumption is true, then Sterbenz’ lemma implies \( \tilde{\lambda} - \Delta \in \mathbb{F} \) and \( \delta = 0 \). We distinguish two cases.

1. If \( \tilde{\lambda} + \Delta \in \mathbb{F} \), then \( \lambda^* = \lambda_\Delta = \lambda_\varphi = \omega \in \mathbb{F} \), and \( \delta^* = \delta_\Delta = \delta_\varphi = \delta \) implies

\[
\varepsilon - \delta^* = \max_i |y_i - \tilde{\lambda}| = \delta = \max(\omega - \lambda, \lambda - \omega) = \delta \geq 0.
\]

2. If \( \tilde{\lambda} + \Delta \notin \mathbb{F} \), then \( \omega \) is the midpoint of two adjacent floating-point numbers. Denote by \( \eta \) the difference to its neighbors. Then, \( \lambda_\Delta = \omega + \eta \) and \( \lambda_\varphi = \omega - \eta \).

(a) If \( \tilde{\lambda} \geq \omega \), then \( \lambda \in \mathbb{F} \) implies \( \lambda \geq \lambda_\Delta \), \( \delta_\Delta = \lambda_\Delta - \Delta = \delta + \eta \in \mathbb{F} \), and

\[
\varepsilon - \delta^* \geq \varepsilon - \delta_\Delta = \max(\tilde{\lambda} - \lambda_\Delta - \Delta - \eta = \tilde{\lambda} - (\Delta + \delta + \eta)) = \lambda - \lambda_\Delta \geq 0.
\]

(b) If \( \tilde{\lambda} < \omega \), then, similarly, \( \lambda \leq \lambda_\varphi \), \( \delta_\varphi = \lambda - \lambda_\varphi = \delta + \eta \in \mathbb{F} \), and \( \varepsilon - \delta^* = \varepsilon - \delta_\varphi = \tilde{\lambda} - (\Delta + \delta + \eta) = \lambda - \lambda_\varphi \geq 0 \).

In Algorithm 11, therefore, we compute \( \delta^* \) as an error bound.

One may consider that computing \( \varepsilon \) is unnecessary. When we use an accurate dot product algorithm (e.g., the INTLAB [43] routine \texttt{Dot}.), however, there is a case where a numerically obtained \( \varepsilon \) is smaller than a numerically obtained \( \delta^* \). This is because \( \varepsilon \) contains the residual \( r \), and \( |r_i| \), \( i = 1, \ldots, n \) are small quantities computed from relatively large quantities. Therefore, calculating \( r \) with the accurate dot product algorithm gives smaller \( |r_i| \). In contrast, the quantities \( \lambda \) and \( \tilde{\lambda} \) are not small in general, so that calculating them with the accurate algorithm is not advantageous. Although \( \delta^* \) is a small quantity computed from relatively large quantities, we can compute it after the calculations of \( \lambda \) and \( \tilde{\lambda} \) are completed.
calculations of $\Delta$ and $\overline{\Delta}$ are completed, the computation of $\delta^*$ involves only few operations. Consequently, it is difficult to make the accurate dot product algorithm advantageous in the calculation of $\delta^*$. Let $\hat{\varepsilon}$ be $\varepsilon$ obtained via the accurate algorithm. We thus design Algorithm 12 such that the minimum of $\delta^*$ and $\hat{\varepsilon}$ is adopted as an error bound.

We next consider computing an error bound for $\hat{x}$. Let $k \in \{1, \ldots, n\}$ satisfy $\hat{x}_k = \max(\hat{x})$, $x^* \in \mathbb{R}_{>0}^n$ be a Perron vector of $A$ such that $x^*_k = \hat{x}_k$, and $\mu := \{1, \ldots, n\} \setminus \{k\}$. Since $U^T U$ coincides with $I$ except $(U^T U)_{kk} = 0$, we have $x^* = \hat{x}_k e + U^T x^*$ and $\hat{x} = \hat{x}_k e + U^T \hat{x}$, so that $x^* - \hat{x} = U^T U(x^* - \hat{x})$. Thus, we estimate an upper bound for $|U(x^* - \hat{x})|$ = $|(x^* - \hat{x})|\). Thus, we estimate an upper bound for $|U(x^* - \hat{x})|$ = $|(x^* - \hat{x})|\).

For the estimation, we give a representation for $(x^* - \hat{x})[\mu]$. Note that $U^T U(x^* - \hat{x}) = x^* - \hat{x}$ and $A[\mu] = UAU^T$. Using $IU = U$ and $Ax^* = \rho(A)x^*$ implies

$$
(\rho(A)I - A[\mu])U(x^* - \hat{x}) = \rho(A)U(x^* - \hat{x}) - UA(x^* - \hat{x}) = U(A\hat{x} - \rho(A)\hat{x}).
$$

Therefore, if $\rho(A)I - A[\mu]$ is nonsingular, then we obtain the representation

$$
(x^* - \hat{x})[\mu] = U(x^* - \hat{x}) = (\rho(A)I - A[\mu])^{-1}U(A\hat{x} - \rho(A)\hat{x}).
$$

In order to show nonsingularity, we prove that $\rho(A)I - A[\mu]$ is an $M$-matrix (see Theorem 3). The $M$-property of $\rho(A)I - A[\mu]$ enables us to calculate the upper bound for $|(x^* - \hat{x})[\mu]|$ at low computational cost (see Lemma 8).

**Lemma 7.** Let $k \in \{1, \ldots, n\}$, $\mu := \{1, \ldots, n\} \setminus \{k\}$, and $A \in \mathbb{R}^{n \times n}_{>0}$ be irreducible. Then, $\rho(A)I - A[\mu]$ is an $M$-matrix.

**Proof.** It is obvious that $\rho(A)I - A[\mu]$ is a Z-matrix. Since $A$ is irreducible, moreover, we have $\rho(A[\mu]) < \rho(A)$, which completes the proof. \qed

Let $\hat{\lambda} \in \mathbb{R}_{>0}$ satisfy $\hat{\lambda} \leq \rho(A)$. If we have $v \in \mathbb{R}^{n-1}_{>0}$ such that $(\hat{\lambda} I - A[\mu])v > 0$, then the upper bound for $|(x^* - \hat{x})[\mu]|$ can be computed efficiently. Lemma 8 is an analogue of the results in [35, 36, 37, 38].

**Lemma 8.** Let $A \in \mathbb{R}^{n \times n}_{\geq 0}$ be irreducible, $\hat{\lambda} \in \mathbb{R}_{>0}$, $\hat{x} \in \mathbb{R}^n_{>0}$, $v, w \in \mathbb{R}^{n-1}_{>0}$, and $k \in \{1, \ldots, n\}$. Define $r := A\hat{x} - \hat{\lambda}\hat{x}$, and $\mu := \{1, \ldots, n\} \setminus \{k\}$. Let also $x^* \in \mathbb{R}^n_{\geq 0}$ be a Perron vector of $A$ such that $x^*_k = \hat{x}_k$. Suppose $\hat{\lambda} \in \mathbb{R}_{>0}$ and $\varepsilon \in \mathbb{R}_{\geq 0}$ satisfy $\hat{\lambda} \leq \rho(A)$ and $\varepsilon \geq |\hat{\lambda} - \rho(A)|$, respectively, and $w \leq (\hat{\lambda} I - A[\mu])v$. Define $s := (|r| + \varepsilon\hat{x})[\mu]$, and $t := (s + \max(s/w))A[\mu]/\hat{\lambda}$. It then follows that $|(x^* - \hat{x})[\mu]| \leq t$.

**Proof.** Let $D := \rho(A)I$ and $E := A[\mu]$. Theorem 3 and Lemma 7 show

$$
|(D - E)^{-1}| = (D - E)^{-1} = D^{-1}(I + E(D - E)^{-1}).
$$

From $\varepsilon \geq |\hat{\lambda} - \rho(A)|$, it follows that

$$
|U(A\hat{x} - \rho(A)\hat{x})| = U|r + (\hat{\lambda} - \rho(A))\hat{x}| \leq U(|r| + |\hat{\lambda} - \rho(A)||\hat{x}|) \leq U(|r| + \varepsilon\hat{x}) = s.
$$

From $\hat{\lambda} \leq \rho(A)$ and $w \leq (\hat{\lambda} I - E)v$, we obtain

$$
(D - E)^{-1}w \leq (D - E)^{-1}(\hat{\lambda} I - E)v \leq (D - E)^{-1}(D - E)v = v.
$$
The relations (2) to (5) and $D^{-1} \leq (1/\Lambda)I$ prove

$$
|(x^* - \hat{x})[\mu]| \leq |(D - E)^{-1}|U(A\hat{x} - \rho(A)\hat{x})| \leq D^{-1}(I + E(D - E)^{-1})s
$$

$$
= D^{-1}(s + E(D - E)^{-1}(s/w \circ w))
$$

$$
\leq D^{-1}(s + \max(s/w)E(D - E)^{-1}w) \leq D^{-1}(s + \max(s/w)Ev) \leq t. \quad \Box
$$

Lemma 8 gives an error bound for $\hat{x}$.

**Theorem 9.** Let $\hat{x}$, $x^*$, and $t$ be as in Lemma 8, and $u := U^T t$. If the assumptions in Lemma 8 are true, then $|x^* - \hat{x}| \leq u$.

**Proof.** As mentioned above, $x^* - \hat{x} = U^T U(x^* - \hat{x})$ follows, so that $|x^* - \hat{x}| = U^T|(x^* - \hat{x})[\mu]|$. This and Lemma 8 prove the result. \( \Box \)

Algorithm 11 computes $\lambda^*$ and $\delta^*$ in Remark 6 and an upper bound on $u$. To obtain the upper bounds on $u$, we need to compute $v$ and $w$ in Lemma 8. We can obtain $v$ by the following procedure:

1. Numerically solve the linear system $(\Lambda I - A[\mu])v^* = b$, where $b \in \mathbb{R}^{n-1}$, and $v^*$ is the solution, via an iterative method. Let $v$ be the numerical solution.
2. Update $v$ such that $v_i = \max(v_i, \text{realmin})$, $i = 1, \ldots, n-1$.

If $v$ is not far from $v^*$, then it approximately follows that

$$
(\rho(A)I - A[\mu])v \geq (\Lambda I - A[\mu])v \approx b > 0.
$$

The choice of $b$ is arbitrary, and a natural one is $b = 1$. In Algorithm 11, we adopt the natural choice. If $\Lambda I - A[\mu]$ is well-conditioned, then the iterative method stops after few iterations, so that $v$ can be obtained efficiently.

If we have $v$, then we can obtain $w$ by the following procedure:

1. Compute a lower bound on $(\Lambda I - A[\mu])v$. Let $w$ be the result.
2. Check $w > 0$.

We then have $w \leq (\Lambda I - A[\mu])v$. If $v$ is not far from $v^*$, then we can expect $w > 0$ from (6). However, a case exists that the iterative method gives inaccurate $v$, so that $w > 0$ does not follow. In such a case, we compute $v$ again via a direct method, update $w$, and check $w > 0$ once more.

**Remark 10.** If the leading eigenvalues are close together, the case $\Lambda < \rho(A[\mu])$ may arise. If so, $\Lambda I - A[\mu]$ is not an $M$-matrix, so there does not exist $v \in \mathbb{R}_{>0}^{n-1}$ which satisfies $(\Lambda I - A[\mu])v > 0$ (see Theorem 3). Consequently, Algorithm 11 does not work. A similar remark applies to Algorithm 12.

Before computing $\lambda^*$, $\delta^*$, and an upper bound on $u$, we need to verify that $A$ is irreducible. If $A > 0$, then the irreducibility is obvious. Otherwise, the irreducibility can be verified by some standard graph algorithm (e.g., [44]).

Based on the discussion above, we propose Algorithm 11.

**Algorithm 11.** Let $\hat{x} \in \mathbb{R}_0^n$ be as in Section 1, $v, w \in \mathbb{R}_{>0}^{n-1}$, and $\Lambda$, $\lambda^*$, and $\delta^*$ be as in Remark 6. Let also $k \in \{1, \ldots, n\}$ satisfy $\hat{x}_k = \max(\hat{x})$, and $x^* \in \mathbb{R}_{>0}^n$ be the Perron vector of $A$ such that $x_k^* = \hat{x}_k$. Define $r := A\hat{x} - \lambda^*\hat{x}$, $\mu := \{1, \ldots, n\} \setminus \{k\}$, $s := (|r| + \delta^*\hat{x})[\mu]$, $t := (s + \max(s/w)A[\mu]v)/\Lambda$, and $u := U^T t$. This
algorithm computes $\lambda^* \in \mathbb{R}_{>0}$, $\delta^* \in \mathbb{R}_{>0}$, and $\pi \in \mathbb{R}_{>0}^n$ such that $\delta^* \geq |\lambda^* - \rho(A)|$ and $\pi \geq |\hat{x} - x^*|$. The irreducibility of $A$ is moreover proved if successful.

**Step 1.** If $A > 0$, then go to Step 3. Otherwise, go to Step 2.

**Step 2.** Verify the irreducibility of $A$ via a standard graph algorithm. If it cannot be verified, then terminate with failure.

**Step 3.** Calculate $\lambda^*$ and $\delta^*$.

**Step 4.** Compute $v$ by numerically solving $(\lambda I - A[\mu])v^* = 1$ via an iterative method. Update $v$ such that $v_i = \max(v_i, \text{realmin})$, $i = 1, \ldots, n - 1$.

**Step 5.** Compute a lower bound on $(\lambda I - A[\mu])v$. Let $w$ be the result. If $w > 0$ cannot be verified, then compute $v$ again via a direct method, update $w$, and check $w > 0$ once more. If $w > 0$ cannot be verified again, then the computation of $\pi$ fails. Terminate.

**Step 6.** Compute an upper bound on $u$. Let $\pi$ be the result. Terminate.

If $\rho(A[\mu]) < \Delta$, then the Neumann series gives

$$v^* = (\lambda I - A[\mu])^{-1}1 = \frac{1}{\Delta} \left( I + \frac{1}{\Delta} A[\mu] + \frac{1}{\Delta^2} (A[\mu])^2 + \cdots \right) 1.$$  

We thus use $\text{fl}((1 I + A[\mu] I/\Delta)/\Delta)$ as an initial guess of the iterative method in Step 4.

5. A technique for obtaining better error bounds. We can obtain better error bounds by using the improved approximation mentioned in Section 3.2. Let $\hat{\lambda}$, $\hat{x}$, $k$, $x^*$, $r$, and $z$ be as in Section 3.2 and $\epsilon$ and $u$ be as in Theorems 5 and 9, respectively. As mentioned in Section 3.2, we can expect $\hat{\lambda} + z_k$ and $\hat{x} + U^T U z$ to be more accurate than $\hat{\lambda}$ and $\hat{x}$, respectively. We store $z$ independently from $\hat{\lambda}$ and $\hat{x}$. Then, $\lambda + z_k$ and $\hat{x} + U^T U z$ are stored as if in multiple-precision floating-point numbers. Let $\hat{r}$, $\hat{\epsilon}$, and $\hat{u}$ be $r$, $\epsilon$, and $u$, respectively, where $\lambda$ and $\hat{x}$ are replaced by $\lambda + z_k$ and $\hat{x} + U^T U z$, respectively. We compute an upper bound on $|\hat{r}|$ executing an accurate dot product algorithm. In the first stage, we obtain $z$ via an iterative method to reduce computational cost. However, a case exists for which the iterative method gives inaccurate $z$. This case occurs when the coefficient matrix of (1) is ill-conditioned. Consequently, we cannot verify $\min(|r| - |\hat{r}|) \geq 0$. In such a case, we compute $z$ again via a direct method.

If $\lambda + z_k > 0$ and $\hat{x} + U^T U z > 0$, then Theorems 5 and 9 applied to $\lambda := \lambda + z_k$ and $\hat{x} := \hat{x} + U^T U z$ give $|\lambda + z_k - \rho(A)| \leq \epsilon$ and $|x^* - (\hat{x} + U^T U z)| \leq \hat{u}$. Let $\epsilon$ and $u$ be upper bounds on $\epsilon$ and $u$, respectively. Define $\lambda_1 := \text{fl}_C(\lambda + z_k - \pi)$, $\lambda_S := \text{fl}_C(\lambda + z_k + \pi)$, $x_I := \text{fl}_C(\hat{x} + U^T U z - \pi)$, $x_S := \text{fl}_C(\hat{x} + U^T U z + \pi)$, $\lambda_C := \text{fl}_C((\lambda + z_k)/2)$, $\epsilon_C := \text{fl}_C(\lambda - \lambda_I)$, $x_C := \text{fl}_C((x_I + x_S)/2)$, and $u_C := \text{fl}_C(x_C - x_I)$. Then, $|\lambda_C - \rho(A)| \leq \epsilon_C$ and $|x_C - x^*| \leq u_C$ follows. Since $\lambda_I \leq \rho(A)$, we update $\Delta$ in Remark 6 such that $\Delta = \lambda_I$ if $\lambda_I \geq \Delta$, which occurs when $\pi$ is small (cf. Remark 6).

We summarize our strategy in Algorithm 12.

**Algorithm 12.** Let $\lambda$, $x$, $k$, $x^*$, $r$, and $z$ be as in Section 3.2, $\Delta$, $\lambda^*$, and $\delta^*$ be as in Remark 6, and $\epsilon$ and $u$ be as in Theorems 5 and 9, respectively. Let also $\hat{r}$, $\hat{\epsilon}$, and $\hat{u}$ be $r$, $\epsilon$, and $u$, respectively, where $\lambda$ and $\hat{x}$ are replaced by $\lambda + z_k$ and $\hat{x} + U^T U z$, respectively. This algorithm computes $\lambda_C \in \mathbb{R}_{>0}$, $\epsilon_C \in \mathbb{R}_{>0}$, $x_C \in \mathbb{R}_{>0}^n$, and $u_C \in \mathbb{R}_{>0}^{\geq 0}$ such that $|\lambda_C - \rho(A)| \leq \epsilon_C$ and $|x_C - x^*| \leq u_C$. The irreducibility of $A$ is moreover proved if successful.

**Steps 1 and 2.** Similar to those in Algorithm 11.
6. Numerical results. We used a computer with Intel Core 1.51 GHz CPU, 16.0 GB RAM, and MATLAB R2012a with the Intel Math Kernel Library and IEEE 754 double precision. We denote the compared algorithms as follows:

M1: combination of the procedure in Section 3.1 and Algorithm 11,
M2: combination of the procedure and Algorithm 12, and
V: combination of the procedure and INTLAB routine verifyeig.

The comparison to verifyeig is not fair because it is applicable to general real or complex, point or interval matrices, computes inner inclusions, as well as bounds for invariant subspaces. However, lacking an algorithm tailored specifically to the Perron pair, we use verifyeig. In Step 2 of M1 and M2, we called tarjan.m in https://jp.mathworks.com/matlabcentral/fileexchange/50707-tarjan-e, which executes the algorithm in [44]. In M1 and M2, we adopted the MATLAB routine bicgstab as the iterative method. We used \((I + A)/\lambda\) as the initial guess of bicgstab (cf. (7)). For A sparse, we used numerically computed incomplete LU factors of A as a preconditioner in bicgstab. In M2, the INTLAB routine Dot_ was invoked as the accurate dot product algorithm. See http://web.cc.iwate-u.ac.jp/~miyajima/PP.zip for details of the implementations, where INTLAB codes of M1, M2, and V (denoted by M1.m, M2.m, and V.m) are uploaded. Let \((\lambda, z)\) and \((\hat{\lambda}, u)\) contain the Perron root and vector, respectively. To assess qualities of enclosures, define the relative radius for the root (RRR) and relative radius for the vector (RRV) as \(\varepsilon/|\lambda|\) and \(\|u\|_2/|\hat{\lambda}|\), respectively. In some problems, M1 and/or M2 failed to compute \(u\). The reason for the failure is that \(w > 0\) could not be verified even in the second stage. In Examples 1 to 5, the routine eig within the procedure in Section 3.1 was invoked once (see Example 5).

Example 1. We observe the RRR, RRV, and CPU times of the algorithms when \(A > 0\). Consider the cases where A is generated by the MATLAB code \(A = gallery(\text{’cauchy’}, 1:n, 2*(1:n))\). In this case, \(A > 0\). The ‘cauchy’ matrix is nonsymmetric. Table 1 displays the RRR, RRV, and CPU times (s) of the algorithms. This table shows that M1 and M2 were faster than V, and the RRR and RRV by M2 were smaller than those by V.

Remark 13. When we used the ‘lotkin’ and ‘minij’ matrices instead, we observed tendencies similar to those in Table 1.

Example 2. We observe properties of the algorithms when \(\min_{i,j} A_{ij} = 0\) and \(A\) is irreducible. Consider the cases where \(A\) is generated by
S. Miyajima

Table 1
RRR, RRV, and CPU times (s) for the ‘cauchy’ matrix.

| n  | M1     | M2     | V     | M1     | M2     | V     |
|----|--------|--------|-------|--------|--------|-------|
|    | RRR    | RRV    | RRV   | Time   | Time   | Time  |
| 500| 4.6e–14| 3.0e–16| 5.6e–15| 0.019  | 0.15   | 0.19  |
|    | 3.3e–12| 3.2e–16| 1.7e–14|        |        |       |
| 1000| 9.3e–14| 2.9e–16| 9.4e–15| 0.086  | 0.36   | 0.46  |
|     | 9.2e–12| 3.2e–16| 3.2e–14|        |        |       |
| 3000| 2.9e–13| 2.7e–16| 2.2e–14| 0.49   | 3.0    | 10    |
|     | 4.7e–11| 3.2e–16| 9.0e–14|        |        |       |
| 6000| 5.9e–13| 2.7e–16| 4.0e–14| 2.3    | 13     | 91    |
|     | 1.3e–10| 3.2e–16| 1.8e–13|        |        |       |

A = gallery('circul',[0;1;zeros(n-2,1)]);
A = gallery('toeppen',n,2,1,0,1,2);
A = gallery('tridiag',ones(n-1,1),zeros(n,1),[2;ones(n-2,1)]);
A = gallery('frank',n,K);

with K being a parameter. The ‘circul,’ ‘tridiag,’ and ‘frank’ matrices are nonsymmetric, whereas the ‘toeppen’ matrix is symmetric. Tables 2, 3, and 4 report the RRR, RRV, and CPU times (s) for the ‘circul,’ ‘toeppen,’ and ‘tridiag’ matrices. Let \( \kappa \) be an approximate condition number of \( \lambda I - A[\mu] \) obtained by the routine cond. Table 5 lists \( \kappa \), and the RRR, RRV, and CPU times (s) of the algorithms for the ‘frank’ matrix. We see from Table 5 that \( M_1 \) and \( M_2 \) failed to enclose the Perron vector in many cases and were slow. The reason for the failure is that a large \( \kappa \) caused an inaccurate \( v \), so that \( w > 0 \) could not be verified. The large \( \kappa \) also caused the slow execution of \( M_1 \) and \( M_2 \) because bicgstab required many iterations. Even when they succeeded, the RRV was large because \( \|A[\mu]v/\lambda\|_2 \) was large. In fact, \( \|A[\mu]v/\lambda\|_2 \) was \( O(10^4) \) and \( O(10^8) \) when \( n = 60 \) and \( K = 0 \) and \( n = 100 \) and \( K = 0 \), respectively.

Table 2
RRR, RRV, and CPU times (s) for the ‘circul’ matrix.

| n  | M1     | M2     | V     | M1     | M2     | V     |
|----|--------|--------|-------|--------|--------|-------|
|    | RRR    | RRV    | RRV   | Time   | Time   | Time  |
| 500| 7.9e–12| 2.2e–16| 3.3e–16| 0.13   | 0.15   | 0.16  |
|    | 4.6e–9 | 4.3e–16| 3.3e–16|        |        |       |
| 1000| 2.9e–11| 2.2e–16| 3.3e–16| 0.63   | 0.66   | 0.96  |
|     | 3.3e–8 | 4.4e–16| 3.3e–16|        |        |       |
| 2000| 2.0e–12| 2.2e–16| 3.3e–16| 3.3    | 4.3    | 6.4   |
|     | 4.7e–9 | 4.4e–16| 3.3e–16|        |        |       |
| 5000| 4.2e–12| 2.2e–16| 3.3e–16| 44     | 49     | 92    |
|     | 2.4e–8 | 1.7e–15| 3.3e–16|        |        |       |
Verified Perron pair of an irreducible nonnegative matrix

Table 3

| $n$  | M1     | M2     | V     | M1     | M2     | V     |
|------|--------|--------|-------|--------|--------|-------|
|      | RRR    | RRV    | RRV   | Time   | Time   | Time  |
| 500  | 2.1e–11| 3.0e–16| 3.0e–16| 0.10   | 0.11   | 0.11  |
|      | 3.1e–7 | 9.4e–15| 6.8e–13|        |        |       |
| 1000 | 5.9e–16| 3.0e–16| 1.5e–16| 0.42   | 0.42   | 0.60  |
|      | 3.0e–11| 2.7e–16| 2.7e–12|        |        |       |
| 2000 | 7.4e–16| 3.0e–16| 1.5e–16| 1.9    | 2.0    | 4.0   |
|      | 1.7e–10| 2.7e–16| 1.1e–11|        |        |       |
| 5000 | 6.5e–12| 3.0e–16| 3.0e–16| 25     | 26     | 63    |
|      | 9.3e–6 | 6.7e–13| 6.8e–11|        |        |       |

Table 4

| $n$  | M1     | M2     | V     | M1     | M2     | V     |
|------|--------|--------|-------|--------|--------|-------|
|      | RRR    | RRV    | RRV   | Time   | Time   | Time  |
| 500  | 5.6e–16| 2.2e–16| 2.2e–16| 0.12   | 0.13   | 0.13  |
|      | 8.5e–11| 2.7e–16| 2.2e–12|        |        |       |
| 1000 | 3.6e–13| 2.2e–16| 1.1e–16| 0.49   | 0.51   | 0.72  |
|      | 2.5e–7 | 2.7e–16| 8.7e–12|        |        |       |
| 2000 | 1.3e–15| 2.2e–16| 1.1e–16| 2.8    | 2.7    | 4.6   |
|      | 3.7e–9 | 1.5e–13| 3.5e–11|        |        |       |
| 5000 | 1.1e–14| 2.2e–16| 1.1e–16| 37     | 38     | 74    |
|      | 1.9e–7 | 5.7e–16| 2.2e–10|        |        |       |

Example 3. We consider the case where eigenvalues are closely clustered around $\rho(A)$. We generated $A$ by the code $A = \begin{bmatrix} 2 & e & 1; e & 2 & 1; e & e & 1 \end{bmatrix}$; with $e$ being a parameter. Then, two eigenvalues are closely clustered around $\rho(A)$ for $e$ close to 0. Let $\lambda_i$ be defined as in Theorem 1, i.e., $|\lambda_1| \geq \cdots \geq |\lambda_n|$, and $\hat{\lambda}_1$ and $\hat{\lambda}_2$ denote numerical results for $\lambda_1$ and $\lambda_2$, respectively. Table 6 displays $\text{fl}(|\hat{\lambda}_1 - \hat{\lambda}_2|)$, and the RRR, RRV, and CPU times (s) for various $e$. We see that $M1$ and $M2$ are not robust when eigenvalues are closely clustered around $\rho(A)$.

Example 4. We observe properties of the algorithms when $\text{min}(\tilde{x})/\|\tilde{x}\|_2$ is small. Consider the case where $A$ is generated by $n = 10; A = \text{gallery}('circul', [0;1; zeros(n-2,1)]); A(n,n) = s$; with $s$ being a parameter. As far as we tested, $\text{min}(\tilde{x})/\|\tilde{x}\|_2$ decreased as $s$ increased. Table 7 reports $\text{fl}(\text{min}(\tilde{x})/\|\tilde{x}\|_2)$, and the RRR, RRV, and CPU times (s) for various $s$. We see that $M1$ and $M2$ worked well even when $\text{min}(\tilde{x})/\|\tilde{x}\|_2$ is small.

Example 5. Let $\Lambda$ be as in Remark 6, and $\rho \in F$ be a lower bound for $\rho(A)$ obtained by $V$. We observe the properties when $\Lambda$ is not a tight lower bound. Consider the case where $A$ is generated by the INTLAB code $A = \text{band(a(rand(n),p,p)};$. Then, $\Lambda$ tends to be not tight when $n$ is large, or $p$ is small. The nontight
Table 5

| n     | K     | κ    | M1  | M2  | V    | M1  | M2  | V    |
|-------|-------|------|-----|-----|------|-----|-----|------|
|       | RRR   | RRV  | Time| Time| Time |
| 60    | 0     | 4.9e+6 | 2.7e–11 | 2.7e–16 | 2.7e–16 | 0.036 | 0.049 | 0.010 |
| 60    | 1     | 3.3e+14 | 4.2e–15 | 2.7e–16 | 2.7e–16 | 0.050 | 0.055 | 0.011 |
| 100   | 0     | 4.2e+10 | 2.5e–14 | 3.1e–16 | 1.6e–16 | 0.091 | 0.11  | 0.012 |
| 100   | 1     | 1.7e+17 | 7.5e–15 | 3.1e–16 | 1.6e–16 | 0.10  | 0.12  | 0.011 |
| 200   | 0     | 4.6e+18 | 2.1e–14 | 3.0e–16 | 3.0e–16 | 0.32  | 0.34  | 0.022 |
| 200   | 1     | 1.6e+18 | 1.7e–14 | 3.0e–16 | 3.0e–16 | 0.32  | 0.35  | 0.022 |
| 500   | 0     | 6.3e+19 | 1.0e+0  | 4.0e–16 | 2.4e–16 | 1.9   | 2.1   | 0.39  |
| 500   | 1     | 4.4e+18 | 5.0e–12 | 2.4e–16 | 2.4e–16 | 1.9   | 2.0   | 0.17  |

Table 6

| e     | fl(|λ1 - λ2|) | M1  | M2  | V    | M1  | M2  | V    |
|-------|---------------|-----|-----|------|-----|-----|------|
|       | RRR           | RRV | Time| Time| Time |
| 1.0e–8 | 4.0e–8 | 1.1e–15 | 4.4e–16 | 2.2e–16 | 0.0054 | 0.010 | 0.0051 |
| 1.0e–12 | 4.0e–12 | 1.1e–12 | 4.4e–16 | 8.7e–15 | 0.0050 | 0.012 | 0.0080 |
| 1.0e–15 | 4.4e–15 | 6.7e–16 | 4.4e–16 | 2.2e–16 | 0.0042 | 0.011 | 0.0070 |

Å is caused by an inaccurate \( \hat{x} \). Table 8 reports fl(\( \rho - \Lambda \)), and the RRR, RRV, and CPU times (s) for two choices of \( n \) and \( p \). Only when \( n = 1000 \) and \( p = 1 \), the iteration in [42] did not converge, so that eig is called. It can be seen that M1 and M2 do not work when \( \Lambda \) is nontight, i.e., \( \hat{x} \) is inaccurate.

7. Future work. As mentioned in Section 3.2, a generalization of the technique in the section is presented in [34, Algorithm 2.1]. One may consider that [34, Algorithm 2.1] can be utilized when eigenvalues are closely clustered around \( \rho(A) \). We denote the number of the clustered eigenvalues by \( \ell \in \{2, \ldots, n\} \). Then, [34, Algorithm 2.1] gives an accurate approximation \( \hat{X} \in \mathbb{C}^{n \times \ell} \) to a basis of invariant subspaces corresponding to the \( \ell \) eigenvalues. Therefore, span(\( \hat{X} \)) contains an accurate approximation to the Perron vector. However, it is not clear how to construct the approximate Perron vector from \( \hat{X} \). The construction is our future work.
Verified Perron pair of an irreducible nonnegative matrix

**Table 7**

RRR, RRV, and CPU times (s), and \( \text{fl}(\min(\tilde{x})/\|\tilde{x}\|_2) \) in Example 4.

| s   | fl(min(\tilde{x})/\|\tilde{x}\|_2) | M1  | M2  | V   | M1  | M2  | V   |
|-----|----------------------------------|-----|-----|-----|-----|-----|-----|
|     |                                  | RRR | RRV | RRV | Time | Time | Time |
| 100 | 1.0e–18                          | 1.4e–16 | 2.8e–16 | 1.4e–16 | 0.0035 | 0.011 | 0.0054 |
|     |                                  | 1.7e–18 | 3.5e–18 | 1.7e–18 |
| 200 | 2.0e–21                          | 1.4e–16 | 2.8e–16 | 1.4e–16 | 0.0029 | 0.0085 | 0.0053 |
|     |                                  | 8.7e–19 | 1.7e–18 | 8.7e–19 |
| 300 | 5.1e–23                          | 1.9e–16 | 3.8e–16 | 1.9e–16 | 0.0029 | 0.0095 | 0.0054 |
|     |                                  | 8.7e–19 | 8.7e–19 | 4.3e–19 |
| 400 | 3.8e–24                          | 1.4e–16 | 2.8e–16 | 1.4e–16 | 0.0025 | 0.0091 | 0.0055 |
|     |                                  | 4.3e–19 | 8.7e–19 | 4.3e–19 |
| 500 | 5.1e–25                          | 2.3e–16 | 2.3e–16 | 1.1e–16 | 0.0038 | 0.0093 | 0.0059 |
|     |                                  | 8.7e–19 | 8.7e–19 | 4.3e–19 |

**Table 8**

RRR, RRV, and CPU times (s), and \( \text{fl}(\rho – \lambda) \) in Example 5.

| n   | p     | fl(\rho - \lambda) | M1  | M2  | V   | M1  | M2  | V   |
|-----|-------|---------------------|-----|-----|-----|-----|-----|-----|
|     |       |                     | RRR | RRV | RRV | Time | Time | Time |
| 500 | 5     | 2.7e–15             | 1.1e–15 | 3.1e–16 | 4.6e–16 | 0.15 | 0.17 | 0.13 |
|     |       |                     | 2.6e–12 | 3.0e–16 | 3.1e–15 |
| 500 | 1     | 2.7e–15             | 8.4e–14 | 4.3e–16 | 2.1e–16 | 0.16 | 0.18 | 0.21 |
|     |       |                     | 1.2e–10 | 2.5e–16 | 3.1e–16 |
| 1000| 5     | 4.4e–15             | 1.3e–15 | 3.0e–16 | 6.0e–16 | 0.42 | 0.45 | 0.68 |
|     |       |                     | 2.5e–13 | 2.9e–16 | 6.2e–15 |
| 1000| 1     | 2.0e+0              | 1.0e+0 | 7.8e–1 | 2.0e–16 | 2.1 | 2.2 | 2.5 |
|     |       |                     | failed | failed | 5.1e–16 |

Lemma 8 and Theorem 9 are more complicated than Theorem 2. Establishing a simpler theory for the verification of \( x^* \) is also our future work.

As observed in Example 5, Algorithms 11 and 12 are sensitive to the accuracy of \( \tilde{x} \). Hence, developing a robust approximation method is also our future work.

**Acknowledgment.** The author sincerely thanks the reviewer for the comments, which significantly improved the first version. In particular, adding with the proof of \( \delta^* \leq \varepsilon \) in Remark 6, the concise proofs of Theorems 4 and 5, (2), and Lemma 7 presented in this paper are also due to the reviewer. The author especially acknowledges for giving them.

**REFERENCES**

[1] G. Frobenius. Über Matrizen aus positiven Elementen, 1. Sitzungsber. Königl. Preuss. Akad. Wiss., 471–476, 1908.
[2] G. Frobenius. Über Matrizen aus positiven Elementen, 2. Sitzungsber. Königl. Preuss. Akad. Wiss., 514–518, 1909.
A. Neumaier. A simple derivation of the Hansen-Bliek-Rohn-Ning-Kearfott enclosure for linear interval equations. Reliab. Comput., 217:259–267, 2008.

S.M. Rump. Guaranteed inclusions for the complex generalized eigenproblem. Computing, 42:225–238, 1989.

S.M. Rump. Computational error bounds for multiple or nearly multiple eigenvalues. Linear Algebra Appl., 324:209–226, 2001.

A. Neumaier. A simple derivation of the Hansen-Bliek-Rohn-Ning-Kearfott enclosure for linear interval equations. Reliab. Comput., 5:131–136, 1999.
[36] S.M. Rump. Accurate solution of dense linear systems, Part II: Algorithms using directed rounding. *J. Comput. Appl. Math.*, 242:185–212, 2013.

[37] A. Minamihata, K. Sekine, T. Ogita, and S. Oishi. Fast verified solutions of sparse linear systems with $H$-matrices. *Reliab. Comput.*, 15:127–141, 2013.

[38] A. Minamihata, K. Sekine, T. Ogita, S.M. Rump, and S. Oishi. Improved error bounds for linear systems with $H$-matrices. *NOLTA, IEICE*, 6(3):377–382, 2015.

[39] L. Collatz. Einschliessungssatz fuer die charakteristischen Zahlen von Matrizen. *Math. Z.*, 48:221–226, 1942–43.

[40] H. Wielandt. Unzerlegbare, nicht negative Matrizen. *Math. Z.*, 52:642–648, 1950.

[41] M. Fiedler and V. Pták. On matrices with non-positive off-diagonal elements and positive principal minors. *Czechoslovak Math. J.*, 12:382–400, 1962.

[42] P. Chanchana. An algorithm for computing the Perron root of a nonnegative irreducible matrix. *Ph.D thesis, North Carolina State University*, 2007. https://repository.lib.ncsu.edu/handle/1840.16/3756

[43] S.M. Rump. INTLAB - INTerval LABoratory. In: T. Csendes (editor), *Developments in Reliable Computing*, Kluwer Academic Publishers, Dordrecht, 77–104, 1999.

[44] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.