Validated computation tool for the Perron-Frobenius eigenvalues

Nagatou, Kaori
Faculty of Mathematics, Kyushu University

Ishii, Yutaka
PRESTO, Japan Science and Technology Agency

出版情報：Kyushu University Preprint Series in Mathematics, 2008. 九州大学大学院数理学研究院
バージョン：
権利関係：

Validated computation tool for the Perron-Frobenius eigenvalues

Kaori Nagatou[†,‡] and Yutaka Ishii[†],

[†]Faculty of Mathematics, Kyushu University, Japan
[‡]PRESTO, Japan Science and Technology Agency

Abstract

A matrix with non-negative entries has a special eigenvalue, the so called Perron-Frobenius eigenvalue, which plays an important role in several fields of science [1]. In this paper we present a numerical tool to compute rigorous upper and lower bounds for the Perron-Frobenius eigenvalue of non-negative matrices. The idea is to express a non-negative matrix in terms of a directed graph, and make use of R. Tarjan's algorithm [5] which finds all strongly connected components of a directed graph very efficiently. This enables us to decompose the original matrix into irreducible components (possibly of small size), and then to enclose the aimed Perron-Frobenius eigenvalue. We also show a numerical example which demonstrates the efficiency of our tool.

1 Introduction

Calculating eigenvalues of a large matrix plays an important role in several fields of science, e.g. fluid dynamics, engineering, physics, chemistry and economics [1]. Among others, non-negative matrices appear in several mathematical models for Markov process, graph theory and dynamical systems. Matrices of this kind are known to have a special eigenvalue, the so called Perron–Frobenius eigenvalue and it often corresponds to an important invariant which characterizes the model.

Although there are several numerical tools to compute eigenvalues of matrices, they may potentially produce big numerical errors of computation in the case of large and sparse matrices which often appear in practical situation. Thus, some validated computation is needed for such class of matrices.

The purpose of this paper is to present an effective numerical tool to compute the Perron–Frobenius eigenvalue for general non-negative matrices with guaranteed accuracy. The idea is to express a non-negative matrix in terms of a directed graph, and make use of R. Tarjan's algorithm [5] which finds all strongly connected components of a directed graph very efficiently. We moreover develope a toolbox on MATLAB [3] which will be open to the public in near future.

2 A search of graph and Tarjan's Algorithm

2.1 A search and strong connectivity

Let $G = (V, E)$ be a graph consisting of a set of verticies V and a set of edges E. When we treat a directed graph, (v, w) represents an edge; $v \in V$ is the

tail and $w \in V$ is the *head* of the *edge*. For each vertex $v \in V$ the list of all verticies w which satisfy $(v, w) \in E$ is called the *adjacency list* for vertex v. The set of adjacency lists for the vertices of G is called the *adjacency structure*. The adjacency structure can be expressed as a square matix as follows. Let N be the number of verticies in G and supposed that they are numbered as $\{v_1, v_2, \cdots, v_N\}$. We define an $N \times N$ matrix A by

$$\begin{aligned} a_{ij} &= 1 \text{ if there is an edge between } v_i \text{ and } v_j \\ a_{ij} &= 0 \text{ (otherwise).} \end{aligned}$$

Then this matrix expresses the adjacency structure and is called the *adjacency matrix*. As we can see from the definition the adjacency matrix for a directed graph is in general asymmetric.

A *search* of a graph is to explore all verticies and edges. The basic two search methods for a graph are the *breadth-first search (BFS)* and the *depth-first search (DFS)*, and here we focus on the DFS which is the basis of Tarjan's algorithm. The DFS algorithm is a search method in which an edge emanating from the vertex most recently reached which still has unexplored edges are always chosen. This rule could be executed using an adjacency structure.

The following program ([5]) carries out the DFS of a connected graph, starting at vertex s. The program gives an number for each vertex of the graph due to the reaching order during the search. In this algorithm each edge is examined twice and each vertex is numbered only once.

DFS Algorithm [5]

```
BEGIN
  INTEGER i;
  PROCEDURE DFS(v, u)
    BEGIN
      NUMBER(v):= i := i+1;
      FOR w in the adjacency list of v DO
        BEGIN
          IF w is not yet numbered THEN
            BEGIN
              construct arc (v, w)
              DFS(w, v);
            END
          ELSE IF NUMBER(w) < NUMBER(v) and w ≠ u
            THEN construct arc (v, w);
        END;
    END;
  i:= 0;
  DFS(s,0);
END
```

Now we consider a directed graph $G = (V, E)$. When there exist both a path from v to w and a path from w to v for each pair of verticies $v,\ w$ in G, the graph G is said to be *strongly connected*.

We say that two verticies v and w are equivalent if there is a closed path from v to v which contains w. Let V_i $(i = 1, \cdots, n)$ be the distinct equivalence classes under this equivalence relation and for i $(1 \leq i \leq n)$ we define the subgraphs G_i of G as follows:

$$G_i = (V_i, E_i), \quad E_i = \{(v, w) \in E \mid v, w \in V_i\}.$$

Then it is known that each G_i is strongly connected and no G_i is a proper subgraph of a strongly connected subgraph of G ([5]). We call the subgraphs G_i the *strongly connected components (SCC)* of G.

2.2 Tarjan's algorithm

It is known that the division into the SCCs of a directed graph is uniquely determined, and the DFS algorithm is an efficient way to achieve it. Especially Tarjan's algorithm [5] is well known as a very efficient one which requires only $O(|V|, |E|)$.

The following program is an implementation of Tarjan's algorithm which was proposed in 1972 [5]. LOWLINK(v) in the program means the smallest vertex which is in the same SCC as v, and v could be the root of the SCC if and only if LOWLINK(v) = v (Lemma 12 in [5]). Tarjan proved that this algorithm worked correctly on any directed graphs.

Tarjan's Algorithm [5]

```
BEGIN
  INTEGER i;
  PROCEDURE STRONGCONNECT(v)
    BEGIN
      LOWLINK(v) := NUMBER(v):= i := i+1;
      put v on stack of points;
      FOR w in the adjacency list of v DO
        BEGIN
          IF w is not yet numbered THEN
            BEGIN
              STRONGCONNECT(w);
              LOWLINK(v) := min(LOWLINK(v), LOWLINK(w);)
            END
          ELSE IF NUMBER(w) < NUMBER(v) DO
            BEGIN
              if w is on stack of points THEN
                LOWLINK(v) := min(LOWLINK(v), NUMBER(w));
            END
        END

      If(LOWLINK(v) =NUMBER(v)) THEN
        BEGIN
          start new strongly connected component;
          WHILE w on top of point stack satisfies
                NUMBER(w) ≥ NUMBER(v) DO
            delete w from point stack and put w in current component;
        END
    END
  i:= 0;
  empty stack of points;
  FOR w a vertex DO
    IF w is not yet numbered THEN STRONGCONNECT(w);
  END
```

3 The toolbox on MATLAB

In this section we explain about the MATLAB function which we have developed by implementing Tarjan's algorithm. See the Code 1. The function *scc* finds all SCCs of a directed graph, and after executing it the member of each SCC corresponds to the elements of each row of a matrix G. The function *list* in the program expresses the adjacency structure of a graph which should be prepared by the users. (`list(v)` is a vector consisting of the adjacency list for a vertex v.)

Code 1: MATLAB code for Tarjan's Algorithm

```
 % Function to find strongly connected components
function [G,N,L,c,k,sccI,stack]=scc(v,G,N,L,c,k,sccI,stack)
UNDEFINED=0;
c=c+1; k=k+1;
N(v)=c; % Visit Number for the vertex v
L(v)=c; % LOWLINK(v)
stack(k)=v;
for w=list(v) % w is in the adjacency list of v
    if N(w)==UNDEFINED
        [G,N,L,c,k,sccI,stack]=scc(w,G,N,L,c,k,sccI,stack);
        L(v)=min(L(v),L(w));
    elseif N(w)<N(v)
       for z=stack
           if w==z
               L(v)=min(L(v),N(w));
           end
       end
    end
end
if L(v)==N(v)
    sccI=sccI+1
    j=1; m=1;
    for t=stack
        if (t>0 & N(t)>=N(v))
            G(sccI,j)=t; %SCC component
            j=j+1;
            stack(m)=-1;
        end
        m=m+1;
    end
end
```

An example of a main function to use *scc* is seen in Code 2. Here, `M` stands for the number of verticies and `sccI` finally reserves the number of SCCs.

Code 2: An example of a main function

```
global M
M=2048
UNDEFINED=0; % Flug for "not yet numbered"
c=0; % Counter for visiting vertex
k=0; % Counter for stack
sccI=0; % Counter for SCC
G=zeros(M,M); % Each row corresponds to a SCC
for v=1:M
    N(v)=UNDEFINED;
    L(v)=UNDEFINED;
    stack(v)=UNDEFINED;
end
v=1 % start from this vertex
    % (Choose any vertex as a starting vertex.)

[G,N,L,c,k,sccI,stack]=scc(v,G,N,L,c,k,sccI,stack);

for v=1:M
    if(N(v)==UNDEFINED)
       [G,N,L,c,k,sccI,stack]=scc(v,G,N,L,c,k,sccI,stack);
    end
end
```

4 Validated computations of the Perron–Frobenius eigenvalues

4.1 Perron–Frobenius Theorem

A square matrix is said to be a permutation matrix if the all elements are either 0 or 1 and each row and column contains exactly only one element 1.

Now we transform a $N \times N$ square matrix A to a block upper triangle matrix of the form:

$$P^t A P = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ \vdots & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \cdots & A_{mm} \end{pmatrix} \tag{4.1}$$

using a permutation matrix P. Since the permutation matrix P is an orthogonal matrix, this transformation is a similarity transformation.

A matrix $A = (a_{ij})$ $(1 \leq i, j \leq n)$ is said to be *reducible* if it can be transformed to the form

$$P^t AP = \begin{pmatrix} A_{11} & A_{12} \\ O & A_{22} \end{pmatrix} \tag{4.2}$$

where A_{kk} $(k = 1, 2)$ are square matrices, by using a orthogonal matrix P. In case of $n = 1$, A is called reducible when $A = O$ (the zero matrix). A matrix is said to be *irreducible* if it is not reducible.

The irreduciblity of a matrix is closely related to a strong connectivity of a graph. Given a non-negative matrix A we create a directed graph $G = (V, E)$ by $V = \{1, \cdots, n\}$ and $(i, j) \in E$ if and only if $a_{ij} \neq 0$. Then one can see that G is strongly connected if and only if A is irreducible. Decomposing G into the SCCs $G_1, \cdots, G_m$ produces a decomposition of V into $V_1, \cdots, V_m$. By reordering them appropriately we can obtain a block upper triangle matrix of the form (4.1), where the matrices A_{ii} $(i = 1, \cdots, m)$ are irreducible and they are called the *irreducible components* of A.

Let $\lambda_1, \lambda_2, \cdots, \lambda_n$ be the eigenvalues of A. We define the *spectral radius* of A by

$$\rho(A) \equiv \lim_{n \to \infty} \|A^n\|^{\frac{1}{n}}, \tag{4.3}$$

where $\|\cdot\|$ is any norm. Since $\|\cdot\|$ is equivalent to the supremum norm, we see

$$\rho(A) = \max_{1 \leq i \leq n} |\lambda_i|. \tag{4.4}$$

An eigenvalue λ is said to be *geometrically simple* if the dimension of the corresponding eigenspace is one, and it is said to be *algebraically simple* if it is a simple root of the characteristic equation. An algebraic simplicity leads a geometric simplicity, but the converse is not.

The following theorem claims the existence of a special eigenvalue for an

irreducible non-negative matrix.

Theorem 4.1 (Perron–Frobenius Theorem) [2]

Let A be a $n \times n$ irreducible non-negative matrix. Then, A has an eigenvalue $\lambda_{\rm PF}(A) > 0$ called the Perron–Frobenius eigenvalue which has the property that $|\lambda| \leq \lambda_{\rm PF}(A)$ for any eigenvalue λ of A. Moreover, (i) it is simple both algebraically and geometrically, (ii) there exists a corresponding eigenvector $\boldsymbol{v}_{\rm PF}(A)$ whose components are all positive.

Below we show Theorem 4.1 via Brouwer's fixed point theorem:

Theorem 4.2 (Brouwer's Fixed Point Theorem) [7]

Let U be a compact and convex subset in $\mathbf{R}^n$ and F be a continuous function on U into itself. Then F has a fixed point $\mathbf{x}$ (i.e. $F(\mathbf{x}) = \mathbf{x}$) in U.

Proof of Theorem 4.1 Thanks to the equality (4.4), by letting $\lambda_{\rm PF}(A) = \rho(A)$ it is enough to show the claims (i) and (ii). The following proof is mainly due to [6].

Since A is non-negative, for $\boldsymbol{x} = (x_i)_{i=1}^n \geq 0$ with $\boldsymbol{x} \neq \mathbf{0}$ we get $A\boldsymbol{x} \geq \mathbf{0}$. Writing A as

$$A = (\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_n), \quad \boldsymbol{a}_i \in \mathbf{R}^n \ (i = 1, \cdots, n),$$

we have

$$A\boldsymbol{x} = x_1\boldsymbol{a}_1 + x_2\boldsymbol{a}_2 + \cdots + x_n\boldsymbol{a}_n.$$

The irreducibility of A implies $\boldsymbol{a}_1 \neq \mathbf{0}$, $\boldsymbol{a}_2 \neq \mathbf{0}, \cdots, \boldsymbol{a}_n \neq \mathbf{0}$, hence $A\boldsymbol{x} = \mathbf{0}$ holds if and only if $x_1 = x_2 = \cdots = x_n = 0$.

Here we define $S \subset \mathbf{R}^n$ and a map $T : S \to \mathbf{R}^n$ as

$$S = \left\{ \boldsymbol{x} \in \mathbf{R}^n \ : \ x_i \geq 0, \ \sum_{i=1}^n x_i = 1 \right\}, \tag{4.5}$$

$$T\boldsymbol{x} = \frac{1}{\theta(\boldsymbol{x})} A\boldsymbol{x}, \quad \theta(\boldsymbol{x}) = \sum_{i=1}^n \sum_{j=1}^n a_{ij} x_j > 0. \tag{4.6}$$

Then S is a compact convex set in $\mathbf{R}^n$, and T is a continuous function on S into itself. Therefore by Brouwer's fixed point theorem there exists a point $\boldsymbol{x}_0 \in S$ such that $T\boldsymbol{x}_0 = \boldsymbol{x}_0$. Setting $\lambda = \theta(\boldsymbol{x}_0) > 0$, we have by (4.6)

$$A\boldsymbol{x}_0 = \lambda \boldsymbol{x}_0. \tag{4.7}$$

This shows that $\lambda > 0$ is an eigenvalue of A and $\boldsymbol{x}_0$ is an eigenvector corresponding to λ.

Next we show that $\boldsymbol{x}_0$ is a positive vector. If $\boldsymbol{x}_0$ contains zero elements then by a suitable permutation matrix P we can write as

$$P\boldsymbol{x}_0 = \begin{pmatrix} \boldsymbol{\xi} \\ \boldsymbol{0} \end{pmatrix} \quad (\boldsymbol{\xi} \text{ is a positive vector}).$$

Then by writing

$$PAP^t = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

(4.7) and $P^tP = I$ imply

$$(PAP^t)P\boldsymbol{x}_0 = \lambda P\boldsymbol{x}_0$$

and we have

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{\xi} \\ \boldsymbol{0} \end{pmatrix} = \lambda \begin{pmatrix} \boldsymbol{\xi} \\ \boldsymbol{0} \end{pmatrix}.$$

This means that $A_{21}\boldsymbol{\xi} = \boldsymbol{0}$, i.e. $A_{21} = O$ and it is inconsistent with the irreducibility of A. Therefore $\boldsymbol{x}_0 > \boldsymbol{0}$ is assured.

If A is irreducible then A^t is also irreducible, so by applying the above argument to A^t it is derived that there exists an eigenvalue $\lambda_1 > 0$ and the corresponding eigenvector $\boldsymbol{x}_1$ which satisfy

$$A^t\boldsymbol{x}_1 = \lambda_1\boldsymbol{x}_1. \tag{4.8}$$

Here let B be a matrix such that $O \leq B \leq A$, β be an any eigenvalue of B and $\boldsymbol{y} = (y_i)_{i=1}^n$ be the corresponding eigenvector to β. By setting

$$|\boldsymbol{y}| = (|y_i|)_{i=1}^n,$$

$B\boldsymbol{y} = \beta\boldsymbol{y}$ implies $|\beta||\boldsymbol{y}| \leq B|\boldsymbol{y}| \leq A|\boldsymbol{y}|$ and we have

$$|\beta|\boldsymbol{x}_1^t|\boldsymbol{y}| \leq \boldsymbol{x}_1^tB|\boldsymbol{y}| \leq \boldsymbol{x}_1^tA|\boldsymbol{y}| = (A^t\boldsymbol{x}_1)^t|\boldsymbol{y}| = \lambda_1\boldsymbol{x}_1^t|\boldsymbol{y}|. \tag{4.9}$$

Since $\boldsymbol{x}_1^t|\boldsymbol{y}| > 0$ holds, we have $|\beta| \leq \lambda_1$ and by the fact that β is an any eigenvalue of B one obtains

$$\rho(B) \leq \lambda_1. \tag{4.10}$$

Especially by choosing $B = A$

$$\rho(A) \leq \lambda_1 \tag{4.11}$$

is derived. λ_1 is an eigenvalue of A^t, and since the eigenvalues of A and A^t are identical, λ_1 is also an eigenvalue of A. Consequently

$$\lambda_1 \leq \rho(A) \tag{4.12}$$

holds and by (4.11) and (4.12) we have

$$\rho(A) = \lambda_1$$

which proves that $\lambda_{\mathrm{PF}}(A) = \rho(A)$ is the eigenvalue of A.

By the way, since (4.10) leads $\rho(B) \leq \rho(A)$, we have

$$O \leq B \leq A \Rightarrow \rho(B) \leq \rho(A). \tag{4.13}$$

Here let's consider the case of $\rho(B) = \rho(A)$. Since β in (4.9) is an any eigenvalue of B, especially by taking $|\beta| = \rho(B)$ and by recalling that $\lambda_1 = \rho(A)$, all inequalities in (4.9) could be replaced by equalities and we have

$$\rho(B)\boldsymbol{x}_1^t|\boldsymbol{y}| = \boldsymbol{x}_1^t B|\boldsymbol{y}| = \boldsymbol{x}_1^t A|\boldsymbol{y}|.$$

Therefore

$$\boldsymbol{x}_1^t(\rho(B)|\boldsymbol{y}| - B|\boldsymbol{y}|) = \boldsymbol{0} \ \text{ and } \ \boldsymbol{x}_1^t(B|\boldsymbol{y}| - A|\boldsymbol{y}|) = \boldsymbol{0}$$

hold. Furthermore by the positivity of $\boldsymbol{x}_1$ and by the inequalities

$$\rho(B)|\boldsymbol{y}| - B|\boldsymbol{y}| \leq \boldsymbol{0} \ \text{ and } \ B|\boldsymbol{y}| - A|\boldsymbol{y}| \leq \boldsymbol{0},$$

we obtain

$$\rho(B)|\boldsymbol{y}| - B|\boldsymbol{y}| = \boldsymbol{0} \ \text{ and } \ B|\boldsymbol{y}| - A|\boldsymbol{y}| = \boldsymbol{0}. \tag{4.14}$$

Finally we have

$$A|\boldsymbol{y}| = \rho(B)|\boldsymbol{y}| = \rho(A)|\boldsymbol{y}|.$$

By the similar argument that $\boldsymbol{x}_0$ was proved as positive, we can prove that $|\boldsymbol{y}| > 0$. Hence the second equality in (4.14) and the relation $B \leq A$ leads $B = A$. Therefore it is proved that $\rho(B) = \rho(A)$ holds if and only if $A = B$.

Here we lastly prove the simplicity of $\lambda_{\mathrm{PF}}(A) = \rho(A)$. We construct a $(n-1) \times (n-1)$ matrix $\widetilde{A}_{ii}$ by eliminating the i-th and j-th row from A. Let I_n be a $n \times n$ identity matrix and we define

$$\begin{aligned} f(\lambda) &= \det(\lambda I_n - A), \\ g_i(\lambda) &= \det(\lambda I_{n-1} - \widetilde{A}_{ii}). \end{aligned}$$

Then we have

$$f'(\lambda) = \sum_{i=1}^{n} g_i(\lambda). \tag{4.15}$$

Let $n \times n$ matrix B be a matrix constructed by replacing i-th and j-th row in A by zero vector. Then $A \geq B_i$ and $A \neq B_i$ hold. Hence by the equality condition in the right hand side of (4.13) we have $\rho(B_i) < \lambda_{\mathrm{PF}}(A)$. Since the eigenvalues of B_i consists of zero and the eigenvalues of A, the roots of

$$g_i(\lambda) = \lambda^{n-1} + \cdots = 0$$

i.e. the eigenvalues of A_{ii} exist in the open disc $|z| < \lambda_{\mathrm{PF}}(A)$. Therefore in case of $\lambda \geq \lambda_{\mathrm{PF}}(A)$ we have $g_i(\lambda) > 0$, and by (4.15)

$$f'(\lambda_{\mathrm{PF}}(A)) = \sum_{i=1}^{n} g_i(\lambda_{\mathrm{PF}}(A)) > 0$$

holds. This assures that $\lambda_{\mathrm{PF}}(A)$ is a simple root of the characteristic equaiton $f(\lambda) = 0$, that is $\lambda_{\mathrm{PF}}(A)$ is algebraic simple. This also implies the geometric simplicity. ∎

Even when A is not irreducible, one can find an eigenvalue $\lambda_{\mathrm{PF}}(A)$ which we again call the *Perron–Frobenius eigenvalue* (PF-eigenvalue) of A with the property that $|\lambda| \leq \lambda_{\mathrm{PF}}(A)$ for any eigenvalue λ of A. In this case, both the algebraic and geometric simplicity does not necessarily holds. Moreover, one can only say that the corresponding eigenvector has non-negative entries.

Let A be a non-negative matrix and let $A_1, A_2, \cdots, A_m$ be its iredducible components. Let $\chi_B(t)$ be the characteristic equation of a matrix B. Then we have

$$\chi_A(t) = \chi_{A_1}(t)\chi_{A_2}(t)\cdots\chi_{A_m}(t).$$

Consequently we have the following

Lemma 4.1 [2]

Let A be a non-negative matrix A and let $A_1, A_2, \cdots, A_m$ be its irreducible components. Then we have

$$\lambda_{\mathrm{PF}}(A) = \max_{1\leq i\leq m} \lambda_{\mathrm{PF}}(A_i). \tag{4.16}$$

Thus, the PF-eigenvalue of a (possibly huge and sparse) non-negative matrix A, which is difficult to compute directly, can be computed via the PF-eigenvalues of the (possibly small) irreducible components A_i of A, which is relatively easier to compute.

4.2 Enclosure of Perron–Frobenius eigenvalue

In order to compute an interval which rigorously contains the PF-eigenvalue of a non-negative matrix A, we make use of the following result which is derived from Theorem 4.1:

Theorem 4.2

Let A be a $n \times n$ non-negative matrix. Then for any positive vector $\boldsymbol{x} = (x_i)_{i=1}^n$ $(x_i > 0)$ we have

$$\min_{1\leq i\leq n} \frac{(A\boldsymbol{x})_i}{x_i} \leq \lambda_{\mathrm{PF}}(A) \leq \max_{1\leq i\leq n} \frac{(A\boldsymbol{x})_i}{x_i}. \tag{4.17}$$

If we choose an approximate eigenvector as a positive vector in Theorem 4.2 then the sharp enclosure for PF-eigenvalue is expected. One of a possible

way to compute such approximate eigenvector is a power method, but it is not efficient if the matrix A is not irreducible. Moreover in case that the matrix A is large a power method needs a high cost in actual computations. In our method, we decompose the matrix into irreducible components and after enclosing PF-eigenvalue of each irreducible matrix we obtain the PF-eigenvalue of the original matrix. In the step of decomposing into irreducible components we made use of Tarjan's algorithm.

The algorithm to enclose the PF-eigenvalue of a non-negative matrix is summarized as in Algorithm 1.

Algorithm 1

1. **Making a graph G from a non-negative matrix A**
   We create a directed graph $G = (V, E)$ by $V = \{1, \cdots, n\}$ and $(i, j) \in E$ if and only if $a_{ij} \neq 0$.
2. **Finding all SCCs of G**
   By making use of Tarjan's algorithm we find all SCCs $G_1, G_2, \cdots, G_m$ of G.
3. **Finding all irreducible components of A**
   From G_i computed in Step 2 we construct an irreducible component A_i.
4. **Enclosing the PF-eigenvalue of each A_i**
   After calculating an approximate eigenvector of A_i, we enclose the PF-eigenvalue $\lambda_{\mathrm{PF}}(A_i)$ by (4.17).
5. **Enclosure of the PF-eigenvalue of A**
   By taking the maximum of $\lambda_{\mathrm{PF}}(A_i)$ we finally enclose the PF-eigenvalue $\lambda_{\mathrm{PF}}(A)$ of A as in (4.16).

5 A numerical example

The computations were carried out on the DELL Precision Workstation (Intel Pentium4 2.4GHz) using MATLAB (Ver. 7.0.1) [3].

We consider the following $n \times n$ asymmetric tridiagonal matrix A.

$$A = \begin{pmatrix} 1 & 0 & & & \\ 1 & 0 & 1 & & \\ & 1 & 0 & \ddots & \\ & \ddots & \ddots & 0 & 1 \\ & & & 0 & 1 \end{pmatrix} \tag{5.1}$$

This matrix has two eigenvalues (one is positive and the other is negative) which have the maximum absolute value. This means that the power method for computing an approximate eigenvector could often fail. In fact in case

of $n = 1024$, even 10000 times iterations could only produce a rather 'bad' approximate eigenvector which gave us the following very rough enclosure

$$\lambda_{\mathrm{PF}}(A) \in [1.0,\ 1.999999998409]$$

with 51.203 seconds execution time.

On the other hand, by using Tarjan's algorithm the matrix A is decomposed into 3 irreducible components and the power method worked efficiently for such irreducible matrices. Finally we obtained the enclosure of PF-eigenvalue as

$$[1.9978632479, 2.000000000000003].$$

The execution time was 42.515 seconds including the finding all SCCs.

6 Conclusion and future works

We produced a numerical verification tool for computing the PF-eigenvalue of a non-negative matrix and confirmed its efficiency. Moreover we can prove that a matrix is irreducible if the program results in that the number of SCC for corresponding graph is 1. In general it is rather difficult to judge the irreducibility of matrices, so this tool is also useful to achieve it.

So far this tool is only fit to a full matrix. Our future work is to modify this tool for sparse matrices (graphs) and also for larger matrices. We are also planning to make use of a parallel computing because enclosing each PF-eigenvalue for irreducible components could be parallelized.

Acknowledgement.

This work was supported by Kyushu University Interdisciplinary Programs in Education and Projects in Research Development (No. 18425).

References

[1] Chatelin, F.: Eigenvalues of Matrices, Wiley, Chichester (1993).

[2] Lind, D., Marcus, B.: An Introduction to Symbolic Dynamics and Coding, Cambridge University Press, Cambridge (1995).

[3] MATLAB: `http://www.mathworks.com/`

[4] Rump, S. M.: Perron-Frobenius Theory for Complex Matrices, Linear Algebra and its Applications (LAA) **363**, pp. 251–273 (2003).

[5] Tarjan, R.: Depth-first search and linear graph algorithms, SIAM J. Comput. **1**, no. 2, pp. 146–160 (1972).

[6] Yamamoto, T.: Suuchi-Kaiseki-Nyumon (in Japanese), Science-sya (2003).

[7] Zeidler, E: Nonlinear Functional Analysis and its Applications I, Springer, New York (1986).