Research Article

Katsuhisa Ozaki* and Takeshi Terao

Test matrices with specified eigenpairs for the symmetric positive definite generalized eigenvalue problem

https://doi.org/10.1515/spma-2025-0038
received December 22, 2024; accepted June 02, 2025

Abstract: The generalized eigenvalue problem is a significant topic with numerous applications in scientific and technological computing. Therefore, verifying the reliability of the results produced by numerical solvers is crucial. This article presents a method for generating test problems for generalized eigenvalue problems such that the exact eigenvalues and eigenvectors are known in advance, thereby enabling precise error monitoring. The proposed method generates a matrix pair that can be represented exactly in floating-point numbers. Ozaki and Ogita previously proposed a method for generating test problems for standard eigenvalue problems. We extend their approach to generalized eigenvalue problems, proposing a method that allows users to specify target eigenvalues and a condition number of the matrix defining the generalized inner product in advance. Furthermore, iterative refinement methods to approach the expected eigenvalues as closely as possible for the user are also discussed. Numerical examples are presented to evaluate the accuracy of the computed results and to assess potential overestimation in the error bounds produced by a method for given approximate eigenvalues.

Keywords: test matrix, generalized eigenvalue problems, rounding errors, floating-point arithmetic

MSC 2020: 65G50, 15A18, 65F15

1 Introduction

In this study, we address the generalized eigenvalue problem for matrices $A, B \in \mathbb{R}^{n\times n}$ defined by

$$Ax^{(i)} = \lambda_i Bx^{(i)}, \quad 1 \le i \le n,$$

where A is a symmetric matrix, B is a symmetric positive definite matrix, λ_i is an eigenvalue, and $x^{(i)}$ is the eigenvector corresponding to λ_i. We express the eigenvalue decomposition as

$$AX = BXD, \quad X^TBX = I, \tag{1}$$

where $D = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a diagonal matrix of eigenvalues, $X = [x^{(1)}, x^{(2)}, \ldots, x^{(n)}]$ is a matrix of eigenvectors, and I is the identity matrix. Solving this problem is crucial for scientific applications.

We rely on numerical methods to approximate these eigenpairs in large-scale generalized eigenvalue problems. Such numerical methods are prone to rounding errors because of finite precision arithmetic, necessitating thorough accuracy checks of the results. If we let $\hat{D}$ and $\hat{X}$ represent the approximations of

* **Corresponding author: Katsuhisa Ozaki,** Department of Mathematical Sciences, Shibaura Institute of Technology, Saitama, Japan, e-mail: ozaki@sic.shibaura-it.ac.jp
Takeshi Terao: Research Institute for Information Technology, Kyushu University, Fukuoka, Japan, e-mail: terao.takeshi.412@m.kyushu-u.ac.jp

Open Access. © 2025 the author(s), published by De Gruyter. This work is licensed under the Creative Commons Attribution 4.0 International License.

D and X in the decomposition, then the accuracy of the computed results is typically evaluated using the residuals and B-orthogonality as follows:

$$|A\hat{X} - B\hat{X}\hat{D}|, \quad |\hat{X}^T B\hat{X} - I|.$$

For a matrix A, we use the notation $|A|$ to denote the matrix whose entries are the absolute values of the corresponding entries of A. The same notation is applied to vectors as well. However, these measures may sometimes be insufficient. As detailed in [5], for an approximate eigenvalue $\hat{\lambda}_k$ and the corresponding eigenvector $\hat{x}^{(k)}$, the error is not solely dependent on the residuals but can be bounded as

$$\min_{1\le j\le n}|\lambda_j - \hat{\lambda}_k| \le \sqrt{||B^{-1}||_2}\,\frac{||A\hat{x}^{(k)} - \hat{\lambda}_k B\hat{x}^{(k)}||_2}{\sqrt{(\hat{x}^{(k)})^T B\hat{x}^{(k)}}}.$$

This expression highlights the complexity of error analysis of numerical solutions to the generalized eigenvalue problem.

To generate a test set for the problem defined in (1), one might consider presetting floating-point matrices X, B, and D and computing $A := BXDX^{-1}$. However, such an approach encounters several challenges:

- Even if X is a floating-point matrix, some elements of X^{-1} may not be representable as floating-point numbers.
- Even if all elements of X^{-1} are floating-point numbers, it is still possible that some elements of A may not be representable in floating-point format.
- Even if all elements of A can be represented as floating-point numbers, the exact matrix A is often unattainable due to the rounding errors inherent to finite-precision arithmetic. Furthermore, using rational arithmetic or high-precision computations to obtain A is impractically slow for large values of n.

If the exact matrix A is not obtained and we have $A + \Delta A$ through numerical computation, then let $\hat{\lambda}$ represent an eigenvalue of (1) with $A + \Delta A$. According to [7], the deviation of the eigenvalues is bounded as

$$|\lambda_i - \hat{\lambda}_i| \le \frac{||\Delta A||_2}{\lambda_{\min}(B)},$$

where A, $A + \Delta A$, and B are symmetric positive definite matrices, and λ_i and $\hat{\lambda}_i$ are sorted in increasing order. Hence, even if $||\Delta A||_2$ is close to the unit roundoff, the bound of $|\lambda_i - \hat{\lambda}_i|$ still depends on the minimum eigenvalue of the matrix B.

The main contribution of this study is a method for the generation of test problems for generalized eigenvalue problems as defined in (1), aimed at assessing the accuracy of numerical results. While test problems with exact solutions have been previously discussed for linear systems [4,9] and standard eigenvalue problems [10], test problems for generalized eigenvalue problems have not been extensively explored. The goal is to generate matrices A and B in (1) that are representable as floating-point numbers, with D and X being exactly known. This allows for the rigorous evaluation of errors in eigenvectors and eigenvalues:

$$|x^{(i)} - \hat{x}^{(i)}| \quad \text{and} \quad |\lambda_i - \hat{\lambda}_i|. \tag{2}$$

Here, we assume that the norms of $x^{(i)}$ and $\hat{x}^{(i)}$ are normalized, the inner product of $x^{(i)}$ and $\hat{x}^{(i)}$ is positive, and that $\hat{\lambda}_i$ represents an approximate eigenvalue. Note that there are no multiple eigenvalues for the comparison of the exact and approximated eigenvectors in (2). This framework also facilitates monitoring the convergence behavior of iterative methods used in numerical computations. In addition, the study addresses verified numerical computations [12], which provide an error bound for an approximate solution obtained using a numerical method. A critical aspect of this process is evaluating how overestimated the obtained error bound is. The numerical results are strongly influenced by the condition number of the matrix B. Typical routines for dense matrices in LAPACK [1], such as sygv, sygvd, and sygvx, begin by applying a Cholesky decomposition to the matrix B. Users specify the expected eigenvalues and the expected condition number α of B, and the proposed method generates matrices A and B, represented in floating-point numbers such that $||B^{-1}||||B|| \approx \alpha$. Moreover, exact matrices D and X can be specified, where the diagonal elements in D are

approximations of the expected eigenvalues from (1). We present two numerical examples to assess the accuracy of MATLAB routines [1] and the overestimation of the upper bounds of errors in verified numerical computations [2,4].

The structure of the remainder of this article is as follows: Section 2 introduces the notation, reviews previous studies, discusses a theorem used in this article, and outlines relevant orthogonal matrices. Section 3 proposes a method for generating test problems for the generalized eigenvalue problem. Section 4 presents an iterative refinement of the proposed method. Finally, numerical examples using the proposed methods verify the accuracy of the results obtained by MATLAB and evaluate the overestimation of error bounds for an approximate solution.

2 Notation, previous study and its extension, and orthogonal matrices

This section introduces the notation and lemmas related to floating-point arithmetic. Let $\mathbb{F}$ denote the set of binary floating-point numbers as defined by the IEEE 754 standard [3]. Typically, $\mathbb{F}$ is the set of binary32 or binary64. Computed results in floating-point arithmetic for rounding to nearest ties to even, rounding toward negative, and rounding toward positive, all in accordance with IEEE 754 [3], are, respectively, denoted as $\mathrm{fl}(\cdot)$, $\mathrm{fl}_\nabla(\cdot)$, and $\mathrm{fl}_\Delta(\cdot)$. We assume that operations using $\mathrm{fl}(\cdot)$ neither overflow nor underflow. Next, u is defined as the unit roundoff, for example, in binary64, $u = 2^{-53}$. The representation ufp (unit in the first place) is defined as follows:

$$\mathrm{ufp}(a) \coloneqq \begin{cases} 0, & a = 0 \\ 2^{\lfloor \log_2 |a| \rfloor}, & \text{otherwise} \end{cases}, \quad a \in \mathbb{R}.$$

Additionally, we define the function npt (next power of two) as

$$\mathrm{npt}(a) \coloneqq \begin{cases} 0, & a = 0 \\ 2^{\lceil \log_2 |a| \rceil}, & \text{otherwise} \end{cases}, \quad a \in \mathbb{R}.$$

We also introduce two lemmas for use herein. One lemma follows directly from the definition of binary floating-point numbers under the IEEE 754 standard.

Lemma 1. *For any real number a, if a belongs to $u2^k\mathbb{Z}$ and $|a| \le 2^k$ for some integer $k \in \mathbb{Z}$, then a is representable in $\mathbb{F}$. Here, we assume that neither overflow nor underflow occurs when a is rounded to a nearest floating-point number.*

The other lemma was developed by Minamihata et al. [6].

Lemma 2. *For $a, \sigma \in \mathbb{F}$, where $\sigma \coloneqq 0.75 \cdot 2^k \cdot 2^\ell$, and $2^\ell \ge |a|$, in which $k \in \mathbb{N}$ and $\ell \in \mathbb{Z}$, the result $a' \coloneqq \mathrm{fl}((\sigma + a) - \sigma)$ satisfies $|a'| \le 2^{-k} \cdot \mathrm{ufp}(\sigma)$ and $a' \in u \cdot \mathrm{ufp}(\sigma)\mathbb{Z}$.*

In this article, we allow the products and sums in each dot product involved in matrix multiplication to be evaluated in an arbitrary order, and the use of fused multiply-add (FMA) operations is also permitted. Accordingly, we do not employ divide-and-conquer approaches such as the Strassen [13] or Winograd [15] algorithms. Given $Y \in \mathbb{F}^{m\times n}$ and $Z \in \mathbb{F}^{n\times p}$, the matrix multiplication routine satisfies the inequality $\mathrm{fl}_\nabla(YZ) \le \mathrm{fl}_\Delta(YZ)$.

We review an error-free transformation of matrix multiplication described in [10] that was employed to generate test matrices for standard eigenvalue problems. Consider matrices $M^{(1)} \in \mathbb{F}^{m\times n}$, $M^{(2)} \in \mathbb{F}^{n\times p}$, and $M^{(3)} \in \mathbb{F}^{p\times q}$. Ozaki and Ogita proposed a method that produces a matrix $M^{(2)'} \in \mathbb{F}^{n\times p}$ satisfying

$$\begin{aligned} \mathrm{fl}(M^{(2)'}M^{(3)}) &= M^{(2)'}M^{(3)} \in \mathbb{F}^{n\times q}, \\ \mathrm{fl}(M^{(1)}(M^{(2)'}M^{(3)})) &= M^{(1)}M^{(2)'}M^{(3)} \in \mathbb{F}^{m\times q} \end{aligned} \tag{3}$$

that serves as an approximation to $M^{(2)}$. The details of this process are summarized in the following theorem from [10]. Note that we can discuss the following similarly:

$$\begin{aligned} \mathrm{fl}(M^{(1)}M^{(2)'}) &= M^{(1)}M^{(2)'} \in \mathbb{F}^{m\times p}, \\ \mathrm{fl}((M^{(1)}M^{(2)'})M^{(3)}) &= M^{(1)}M^{(2)'}M^{(3)} \in \mathbb{F}^{m\times q}. \end{aligned}$$

Theorem 1. *Consider matrices $M^{(1)} \in \mathbb{F}^{m\times n}$, $M^{(2)} \in \mathbb{F}^{n\times p}$, and $M^{(3)} \in \mathbb{F}^{p\times q}$. Define ϕ_{ij} for non-zero $m_{ij}^{(3)}$ and ψ_{ij} for non-zero $m_{ij}^{(1)}$ as follows*:

$$m_{ij}^{(3)} \in \phi_{ij}\mathbb{Z}, \quad m_{ij}^{(3)} \notin 2\phi_{ij}\mathbb{Z}, \quad m_{ij}^{(1)} \in \psi_{ij}\mathbb{Z}, \quad m_{ij}^{(1)} \notin 2\psi_{ij}\mathbb{Z},$$

where ϕ_{ij} and ψ_{ij} are the powers of two. For $m_{ij}^{(3)} = 0$, set $\phi_{ij} := 0$, and for $m_{ij}^{(1)} = 0$, set $\psi_{ij} := 0$. Define the constants β, γ, θ, and ω as follows:

$$\beta := \max_{\substack{i,j,k \\ \phi_{kj}\neq 0}} \frac{\phi_{ij}}{\phi_{kj}}, \quad \gamma := \max_{\substack{i,j \\ \phi_{ij}\neq 0}} \frac{\mathrm{ufp}(m_{ij}^{(3)})}{\phi_{ij}}, \quad \theta := \max_{\substack{i,j,k \\ \psi_{ik}\neq 0}} \frac{\psi_{ij}}{\psi_{ik}}, \quad \omega := \max_{\substack{i,j \\ \psi_{ij}\neq 0}} \frac{\mathrm{ufp}(m_{ij}^{(1)})}{\psi_{ij}}. \tag{4}$$

Let constants n_1, n_2, and n_3 be defined as the maximum number of non-zero elements in a column of $M^{(1)}$, a row of $M^{(2)}$, and a row of $M^{(3)}$, respectively. Define σ by

$$\sigma := 12\mathrm{ufp}(\alpha)\beta\gamma\theta\omega \in \mathbb{F},$$

where α satisfies:

$$\mathbb{F} \ni \alpha \geq n_1 \cdot n' \max_{i,j} |m_{ij}^{(2)}|, \quad n' := \min(n_2, n_3).$$

The matrix $M^{(2)'}$ is computed as

$$m_{ij}^{(2)'} := \mathrm{fl}((\sigma + m_{ij}^{(2)}) - \sigma), \quad \forall i, j.$$

If the condition $4n_1 \cdot n' \cdot u\beta\gamma\theta\omega \leq 1$ is met, then (3) *is satisfied.*

We present another theorem for achieving error-free matrix multiplication.

Theorem 2. *For $M^{(1)} \in \mathbb{F}^{m\times n}$, $M^{(2)} \in \mathbb{F}^{n\times p}$, and $M^{(3)} \in \mathbb{F}^{p\times q}$, vectors θ for $m_{ij}^{(1)} \neq 0$ and ϕ for $m_{ij}^{(3)} \neq 0$ are defined as follows*:

$$m_{ij}^{(1)} \in \theta_i\mathbb{Z}, \quad m_{ij}^{(1)} \notin 2\theta_i\mathbb{Z} \quad \textit{for } \forall j, \quad k_i^{(1)} = \mathrm{npt}(\alpha_i^{(1)}), \quad \alpha_i^{(1)} = \max_j |m_{ij}^{(1)}|, \tag{5}$$

$$m_{ij}^{(3)} \in \phi_j\mathbb{Z}, \quad m_{ij}^{(3)} \notin 2\phi_j\mathbb{Z} \quad \textit{for } \forall i, \quad k_j^{(3)} = \mathrm{npt}(\alpha_j^{(3)}), \quad \alpha_j^{(3)} = \max_i |m_{ij}^{(3)}|, \tag{6}$$

where θ_i and ϕ_j are power of two, and it is assumed that there is no zero row vector in $M^{(1)}$ and no zero column vector in $M^{(3)}$. Let n_1, n_2, and n_3 be defined as in Theorem 1. *A constant σ is defined by*

$$\begin{aligned} &\sigma := 0.75k_0 \max_i \frac{k_i^{(1)}}{\theta_i} k^{(2)} \max_j \frac{k_j^{(3)}}{\phi_j} \in \mathbb{F}, \quad k_0 = \mathrm{npt}(n_1 n'), \quad n' = \min(n_2, n_3), \\ &k^{(2)} = \mathrm{npt}(\alpha^{(2)}), \quad \alpha^{(2)} = \max_{i,j} |m_{ij}^{(2)}|. \end{aligned} \tag{7}$$

If matrix $M^{(2)'}$ is computed as

$$m_{ij}^{(2)'} \coloneqq \mathrm{fl}((\sigma + m_{ij}^{(2)}) - \sigma), \quad \forall i, j,$$

then (3) *is satisfied.*

Proof. If there is a zero row vector in $M^{(1)}$ or zero column vector in $M^{(2)}$ or $M^{(3)}$, the corresponding computations involve no rounding errors. Therefore, we assume that there is no zero row vector in $M^{(1)}$ or no zero column vector in $M^{(3)}$ hereinafter.

From Lemma 2 with

$$2^k \coloneqq k_0 \max_i \left(\frac{k_i^{(1)}}{\theta_i} \right) \left(\max_j \frac{k_j^{(3)}}{\phi_j} \right), \quad 2^\ell \coloneqq k^{(2)},$$

we have

$$\begin{aligned} & m_{ij}^{(2)'} \in u \cdot \mathrm{ufp}(\sigma)\mathbb{Z}, \\ & |m_{ij}^{(2)'}| \le \frac{\mathrm{ufp}(\sigma)}{k_0} \Big/ \left(\max_i \frac{k_i^{(1)}}{\theta_i} \right) \Big/ \left(\max_j \frac{k_j^{(3)}}{\phi_j} \right) \le \frac{\mathrm{ufp}(\sigma)}{k_0} \cdot \frac{\theta_i}{k_i^{(1)}} \cdot \frac{\phi_j}{k_j^{(3)}}. \end{aligned} \tag{8}$$

From this, (5), and (6), we can obtain

$$\sum_{\ell=1}^{p} m_{k\ell}^{(2)'} m_{\ell j}^{(3)} \in u \cdot \mathrm{ufp}(\sigma)\phi_j \mathbb{Z}, \quad \sum_{k=1}^{n} m_{ik}^{(1)} \sum_{\ell=1}^{p} m_{k\ell}^{(2)'} m_{\ell j}^{(3)} \in u \cdot \mathrm{ufp}(\sigma)\theta_i \phi_j \mathbb{Z}. \tag{9}$$

Using (5)–(8), we can derive

$$\begin{aligned} \left| \sum_{\ell=1}^{p} m_{k\ell}^{(2)'} m_{\ell j}^{(3)} \right| &\le \sum_{\ell=1}^{p} (|m_{k\ell}^{(2)'}| \cdot |m_{\ell j}^{(3)}|) \\ &\le n' \frac{\mathrm{ufp}(\sigma)}{k_0} \cdot \frac{\theta_i}{k_i^{(1)}} \cdot \frac{\phi_j}{k_j^{(3)}} k_j^{(3)} \\ &\le n' \frac{\mathrm{ufp}(\sigma)}{k_0} \cdot \frac{\theta_i}{k_i^{(1)}} \phi_j \le \mathrm{ufp}(\sigma)\phi_j \end{aligned} \tag{10}$$

and

$$\begin{aligned} \left| \sum_{k=1}^{n} m_{ik}^{(1)} \sum_{\ell=1}^{p} m_{k\ell}^{(2)'} m_{\ell j}^{(3)} \right| &\le \sum_{k=1}^{n} |m_{ik}^{(1)}| \sum_{\ell=1}^{p} (|m_{k\ell}^{(2)'}| \cdot |m_{\ell j}^{(3)}|) \\ &\le n' \sum_{k=1}^{n} |m_{ik}^{(1)}| \frac{\mathrm{ufp}(\sigma)}{k_0} \cdot \frac{\theta_i}{k_i^{(1)}} \cdot \frac{\phi_j}{k_j^{(3)}} k_j^{(3)} \\ &\le n n' k_i^{(1)} \frac{\mathrm{ufp}(\sigma)}{k_0} \cdot \frac{\theta_i}{k_i^{(1)}} \phi_j \le \mathrm{ufp}(\sigma)\theta_i\phi_j. \end{aligned} \tag{11}$$

From (9)–(11), Lemma 1 guarantees (3). Note that the last term in (10) and (11) is an upper bound of $|M^{(2)'}| \cdot |M^{(3)}|$ and $|M^{(1)}| \cdot |M^{(2)'}| \cdot |M^{(3)}|$, respectively. Hence, all intermediate results are exactly representable in floating-point arithmetic, and no rounding errors arise during the computation. □

Next, we will show how Theorems 1 and 2 differ. Let

$$M^{(1)} \coloneqq \begin{pmatrix} 1.5 & 1 \\ 0.5 & 1.5 \end{pmatrix}, \quad M^{(2)} \coloneqq \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \quad M^{(3)} \coloneqq \begin{pmatrix} 1.5 & 0.5 \\ 1 & 1.5 \end{pmatrix}.$$

For Theorem 1,

$$\sigma = 768 \text{ from } \beta = \gamma = \theta = \omega = 2, \quad \alpha = 8.$$

On the other hand, for Theorem 2,

$$\sigma = 48 \text{ from } k_0 = k_2 = 2, \quad \max_i \frac{k_i^{(1)}}{\theta_i} = \max_j \frac{k_j^{(3)}}{\phi_j} = 4.$$

Thus, we obtained a smaller σ from Theorem 2.

To generate a test problem for standard eigenvalue problems using a symmetric matrix, we set $M^{(2)}$ as a diagonal matrix and $M^{(3)}$ as an orthogonal matrix, and define $M^{(1)} \coloneqq M^{(3)T}$. Ozaki and Ogita [10] utilized Hadamard matrices, a type of a weighing matrix, for both $M^{(1)}$ and $M^{(3)}$ with scaling. These matrices are advantageous because they allow us to set $\beta = \gamma = \theta = \omega \coloneqq 1$ in (4) in Theorem 1 and

$$\max_i \frac{k_i^{(1)}}{\theta_i} = \max_j \frac{k_j^{(3)}}{\phi_j} = 1$$

in (7) in Theorem 2. Furthermore, if an $n \times n$ Hadamard matrix is available, an $2n \times 2n$ Hadamard matrix can easily be generated from it [14]. Here, we define H as a Hadamard matrix and H' as an orthogonal matrix scaled from H. Specifically, for a matrix dimension n, $H' \coloneqq H/\sqrt{n}$. In this study, we refer to H' as a scaled Hadamard matrix. Note the following:

- Hadamard matrices do not exist for certain values of n, and
- if the dimension of H' is not a power of four, then $H' \notin \mathbb{F}^{n\times n}$.

3 Test matrices of generalized eigenvalue problem

We here consider generating test matrices for generalized eigenvalue problems. First, we assume that B is constructed from a nonsingular matrix $R \in \mathbb{F}^{n\times n}$ such that $B = R^TR$. From (1), we can derive

$$AX = BXD \Leftrightarrow AX = R^TRXD \quad \Leftrightarrow R^{-T}AR^{-1}RX = RXD.$$

Let us define

$$A \coloneqq R^TQDQ^TR, \quad Y \coloneqq RX,$$

where Q is an orthogonal matrix. This allows the standard eigenvalue decomposition

$$AX = BXD \Leftrightarrow QDQ^TY = YD.$$

Therefore, D contains the eigenvalues in its diagonal elements, and $X = R^{-1}Y$ is the matrix of eigenvectors (where we set $Y \coloneqq Q$). In summary, generating the test matrices requires the following setup:

$$\begin{cases} S \coloneqq R^TQ \\ A \coloneqq \mathrm{SDS}^T \\ B \coloneqq R^TR \\ X \coloneqq R^{-1}Q \end{cases}. \tag{12}$$

We focus on configuring R and $D \in \mathbb{F}^{n\times n}$ such that

$$\mathrm{SDS}^T, \quad R^TR \in \mathbb{F}^{n\times n}.$$

A nonsingular matrix R is generated as follows:

$$R \coloneqq \sqrt{n}EQ = EH. \tag{13}$$

Here, $E \in \mathbb{F}^{n\times n}$ is a diagonal matrix with elements $e_{ii} = 2^{p_i}$ for $p_i \in \mathbb{Z}$. The orthogonal matrix Q can be generated by a scaled Hadamard matrix. We can control the condition number of B by $\max e_{ii}$ and $\min e_{ii}$. It is possible to use a Householder matrix as an orthogonal matrix. While this would be an advantageous in terms of a lower computational cost for matrix multiplication, this article does not use a Householder matrix due to a strong limitation on the condition number of A or B.

Theorem 3. *Let R be as in* (13). *If*

$$2u \cdot \sum_{i=1}^{n} e_{ii}^2 \le \min_i e_{ii}^2 \tag{14}$$

holds, then $R^T R \in \mathbb{F}^{n\times n}$.

Proof. From the definition of R in (13), we have $R^T R = H^T E^2 H$. Because

$$\left(\min_i e_{ii}^2\right)\mathbb{Z} \ni |H^T E^2 H|_{ij} \le (|H^T| \cdot E^2 \cdot |H|)_{ij} \le \sum_{i=1}^{n} e_{ii}^2,$$

if (14) is satisfied, from Lemma 1, $H^T E^2 H \in \mathbb{F}^{n\times n}$. □

Next, we discuss how to set D such that $\mathrm{SDS}^T \in \mathbb{F}^{n\times n}$.

Theorem 4. *Let $S \coloneqq R^T Q$, $R \coloneqq EH$ in* (12) *and* (13), *$Q \coloneqq H'$. Further, let $\lambda_1, \ldots, \lambda_n \in \mathbb{F}$. Define $\sigma \in \mathbb{F}$ by*

$$\sigma \coloneqq 0.75\left(\frac{\mathrm{npt}(\sum_{i=1}^{n} e_{ii})}{\min_i e_{ii}}\right)^2 \cdot \mathrm{npt}(n) \cdot \mathrm{npt}\left(\max \frac{\lambda_i}{n}\right). \tag{15}$$

The diagonal elements in $D'(=D/n)$ are computed as

$$d'_{ii} \coloneqq \mathrm{fl}\left(\left(\sigma + \frac{\lambda_i}{n}\right) - \sigma\right).$$

Let

$$M^{(1)} = M^{(3)} \coloneqq H^T EH, \quad M^{(2)} \coloneqq \mathrm{diag}(d', d', \ldots, d').$$

If

$$2u \cdot \sum_{i=1}^{n} e_{ii} \le \min_i e_{ii}, \tag{16}$$

then $\mathrm{fl}(M^{(1)}(M^{(2)}M^{(3)})) = M^{(1)}M^{(2)}M^{(3)}$.

Proof. If (16) is satisfied, then $M^{(1)}, M^{(3)} \in \mathbb{F}^{n\times n}$ from Lemma 1 and that

$$\left(\min_i e_{ii}\right)\mathbb{Z} \ni |H^T EH|_{ij} \le (|H^T| \cdot E \cdot |H|)_{ij} = \sum_{i=1}^{n} e_{ii}.$$

From this, we obtain

$$\max_i \frac{k_i^{(1)}}{\theta_i}, \quad \max_i \frac{k_i^{(3)}}{\phi_i} \le \frac{\mathrm{npt}\left(\sum_{i=1}^{n} e_{ii}\right)}{\min_i e_{ii}},$$

for $M^{(1)}$ and $M^{(3)}$. Finally, if we define the constant σ as in (15) and apply Theorem 2, the proof is complete. □

Note that from (12), $S = H^T EH'$ and

$$A = (H^T EH')D((H')^T EH) = (H^T EH)D'(H^T EH) \in \mathbb{F}^{n\times n}.$$

These indicate that λ_i in Theorem 4 is an expected eigenvalue and nd_{ii} is a generalized eigenvalue. If $\lambda_i = \lambda_j$ holds for $i \ne j$, then $d_{ii} = d_{jj}$, allowing us to generate a test problem with multiple eigenvalues, as desired by the user. The MATLAB code is provided in the Appendix for reference.

The main purpose of the proposed method is to evaluate the accuracy of the computed eigenpairs. We have

Table 1: Size of σ' relative to the original σ: $\log_2(\sigma/\sigma')$ for $n = 1{,}024$ (binary64)

$cnd \backslash e_{\text{ratio}}$	10^1	10^2	10^3	10^4	10^5
10^1	12	12	12	11	13
10^2	12	12	12	14	14
10^3	11	11	10	12	13
10^4	13	13	12	14	15
10^5	11	11	11	13	13

$$X \coloneqq R^{-1}Q = H^{-1}E^{-1}Q = \frac{1}{n}H^TE^{-1}H' = \frac{1}{n\sqrt{n}}H^TE^{-1}H.$$

From a discussion similar to that for Theorem 3, if $2u \cdot \sum_{i=1}^{n} e_{ii} \le \min e_{ii}$ is satisfied,

$$\mathrm{fl}(H^T(E^{-1}H)) = H^TE^{-1}H \in \mathbb{F}^{n\times n}.$$

The user gives a vector $r \in \mathbb{F}^n$ whose elements are anticipated eigenvalues of the symmetric positive definite matrix B ($r_i > 0$ for all i). Let $\alpha \coloneqq \max r_i / \min r_i$; this represents the expected condition number of the matrix B. We produce $r' \in \mathbb{F}^n$ with integers k_i such that

$$r_i' = 2^{2k_i}, \quad k_i = \{\eta \in \mathbb{Z} \mid \min|r_i - 2^{2\eta}|\}.$$

Then, we have

$$\frac{1}{2} \le \frac{r_i}{r_i'} \le 2.$$

We set $E \coloneqq \mathrm{diag}(\sqrt{r_1'}, \sqrt{r_2'}, \ldots, \sqrt{r_n'})$, where $\sqrt{r_i'} \in \mathbb{F}$ for all i. Hence, if $R^TR = \mathrm{fl}(R^TR)$, then

$$\frac{1}{2}\alpha \le \|B^{-1}\|\|B\| \le 2\alpha$$

holds. Therefore, the actual condition number of B is not so far from the expected B-condition number.

4 Iterative refinement and numerical examples

In the previous section, we introduced how to generate test matrices with specified eigenvalues. If we set a smaller σ in Theorem 2, the difference $|nd_{ii} - \lambda_i|$ becomes smaller. The important point is whether rounding errors occur in the generation of A and B. Hereafter, all examples are computed using MATLAB R2021b on Windows 10 with a Core i7-8665U.

For $Y, Z \in \mathbb{F}^{n\times n}$, we consider the matrix multiplication YZ. If $\mathrm{fl}_\nabla(YZ) = \mathrm{fl}_\Delta(YZ)$, then $\mathrm{fl}(YZ) = YZ$. Therefore, if we verify that no rounding error occurs in computing A and B, we can perform the update $\sigma \coloneqq 0.5\sigma$. These computations are repeated until a rounding error occurs in the generation of A or B.

Let σ' be the last value of σ updated by iterative refinements. Assume that the expected eigenvalue λ_i is uniformly distributed and

$$cnd \coloneqq \frac{\max|\lambda_i|}{\min|\lambda_i|}, \quad e_{\text{ratio}} \approx \frac{\max_i e_i}{\min_i e_i}.$$

Table 1 shows $\log_2(\sigma/\sigma')$, which indicates how well the iterative refinement method worked. In all cases shown in the table, $\log_2(\sigma/\sigma') \ge 10$, indicating that d_{ii} retains more than 10 bits in the significand relative to the original value.

We set a vector e to control the condition number of the matrix B. If $\max_i e_i / \min_i e_i$ is large, we obtain $d_{ii} = 0$, which is not desired if a user expects a non-singular matrix. Tables 2 and 3 show the maximum values

Table 2: Comparison of upper bound of the B-condition number (binary64)

n\cnd	10^1	10^3	10^5	10^7	10^9	10^{11}
1,024	10^{13}	10^{13}	10^{13}	10^{12}	10^8	10^4
2,048	10^{12}	10^{12}	10^{12}	10^{11}	10^7	10^3
4,096	10^{12}	10^{12}	10^{12}	10^{11}	10^7	10^2
8,192	10^{12}	10^{10}	10^{12}	10^8	10^6	10^6

of e_{ratio} and cnd_A from fixed cnd_A and cnd_B, respectively, to ensure that no rounding error occurs during generation. Here, *cnd* and cnd_B are finite. Note that we applied the iterative refinement method for these examples.

Next, we compare the difference between the expected eigenvalues λ provided by the user and the eigenvalues λ' obtained by the proposed method. Assume that

$$\lambda_1 < \lambda_2 < \cdots < \lambda_n, \quad \lambda_1' < \lambda_2' < \cdots < \lambda_n',$$

and that λ is arithmetically distributed. The relative difference is measured as $|\lambda_i - \lambda_i'|/|\lambda_i|$, and the results are shown in Figures 1 and 2. From these results, it can be observed that as the condition number increases, the overall gap tends to widen, and the gap tends to be larger for eigenvalues with smaller absolute values.

5 Application of the proposed method

In this section, we present numerical examples using test matrices generated by the proposed method.

5.1 Evaluation of MATLAB's built-in function

MATLAB's built-in function eig produces a vector $\hat{d}$ with approximate eigenvalues and the corresponding eigenvector matrix $\hat{X}$ such that

$$[\hat{X}, \hat{d}] = \text{eig}(A, B, \text{'vector'}, \text{algorithm}),$$

where we can select "chol" or "qz" as the algorithm for the eigenvalue problem. We can generate $A, B \in \mathbb{F}^{n\times n}$ such that the exact eigenvalue nd_i and the corresponding eigenvector $x^{(i)}$ are known. Then, we can check the accuracy of the computed eigenvalues as

$$\frac{|\hat{d}_{ii} - nd_i|}{|nd_i|}, \quad d_i \neq 0, \tag{17}$$

and that of computed eigenvectors as

Table 3: Comparison of upper bound of *cnd* (binary64)

n\e_{ratio}	10^1	10^3	10^5	10^7	10^9	10^{11}
1,024	10^{12}	10^{11}	10^{10}	10^9	10^8	10^7
2,048	10^{12}	10^{11}	10^{10}	10^9	10^8	10^7
4,096	10^{11}	10^{10}	10^{10}	10^9	10^7	10^7
8,192	10^{11}	10^{10}	10^9	10^8	10^7	10^6

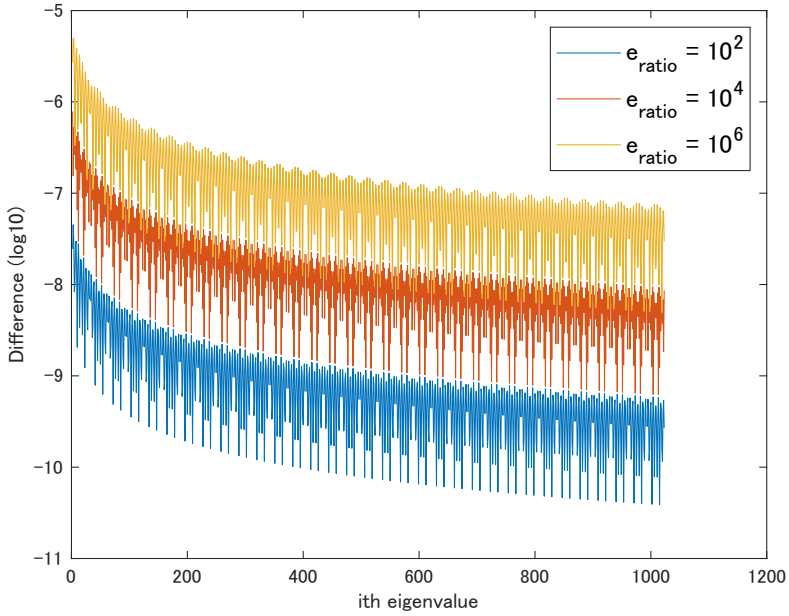

Figure 1: Difference between expected and exact eigenvalues for $cnd = 10^2$ and three values of e_{ratio} ($n = 1{,}024$).

$$|\hat{x}^{(i)} - x^{(i)}|, \tag{18}$$

where $x^{(i)}$ denotes the ith column vector of matrix X, $\hat{x}^{(i)}$, and $x^{(i)}$ are normalized, and the sign of $\hat{x}^{(i)}$ was adjusted to ensure that the inner product between $\hat{x}^{(i)}$ and $x^{(i)}$ is zero.

Figures 3 and 4 indicate (17) and (18), respectively, for $n = 1{,}024$, $nd_i \approx i$, and $\text{cnd}_B = 10^2, 10^9$. The computational accuracy deteriorates according to the condition number of B, and the smaller the absolute value of an eigenvalue, the worse the relative error tends to be. Additionally, the differences in accuracy due to variations in algorithms can also be observed from the figures.

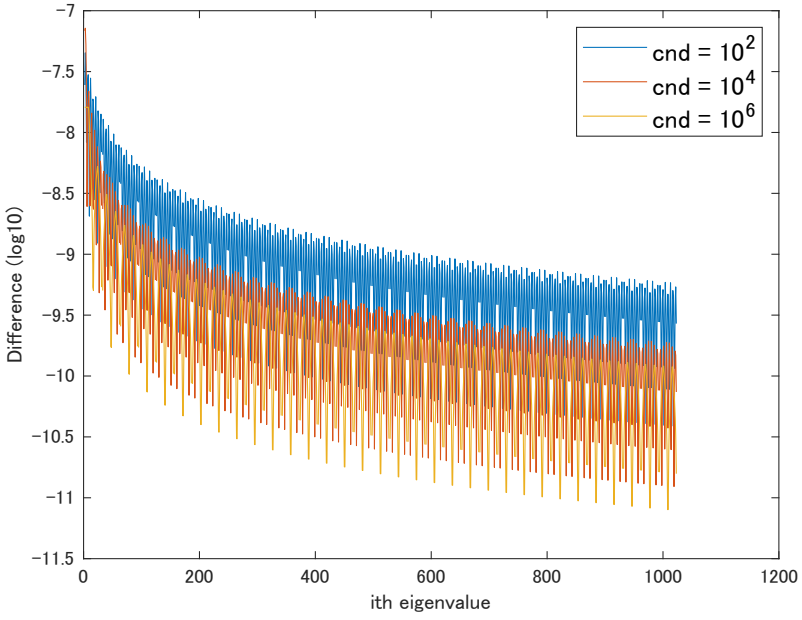

Figure 2: Difference between expected and exact eigenvalues for $e_{\text{ratio}} = 10^2$ and three values of cnd ($n = 1{,}024$).

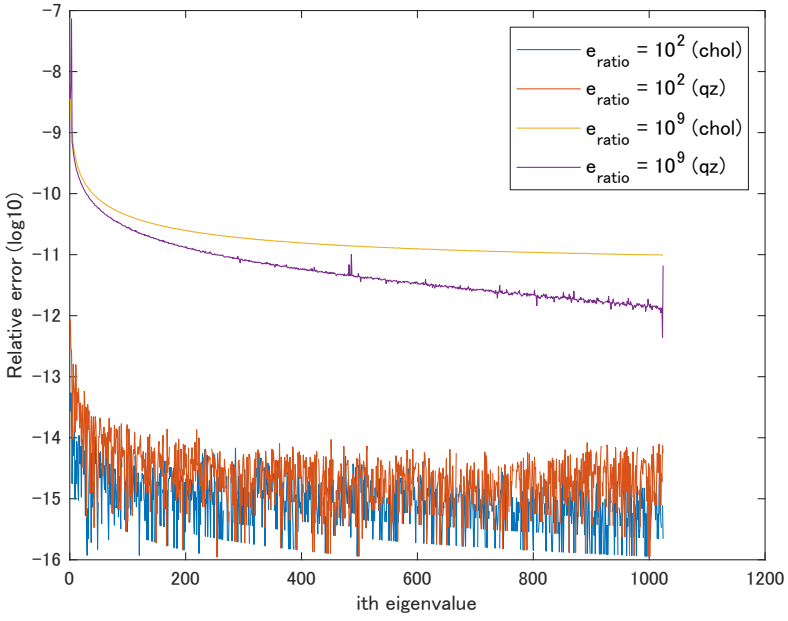

Figure 3: Relative errors for the computed eigenvalues (17).

5.2 Checking overestimation of the verified numerical computations

We next review the verification method in [2,4]. For the generalized eigenvalue problem (1), define $\hat{D}$ and $\hat{X}$ as the computed results of D and X, respectively. Let

$$R = \hat{X}^T(A\hat{X} - B\hat{X}\hat{D}), \quad G = \hat{X}^T B\hat{X} - I.$$

Eigenvalues are included in the following union:

$$\bigcup_{i=1}^{n}[\tilde{\lambda}_i - r_i, \tilde{\lambda}_i + r_i], \quad |R|e + \frac{\|R\|_\infty}{1 - \|G\|_\infty}|G|e \le r,$$

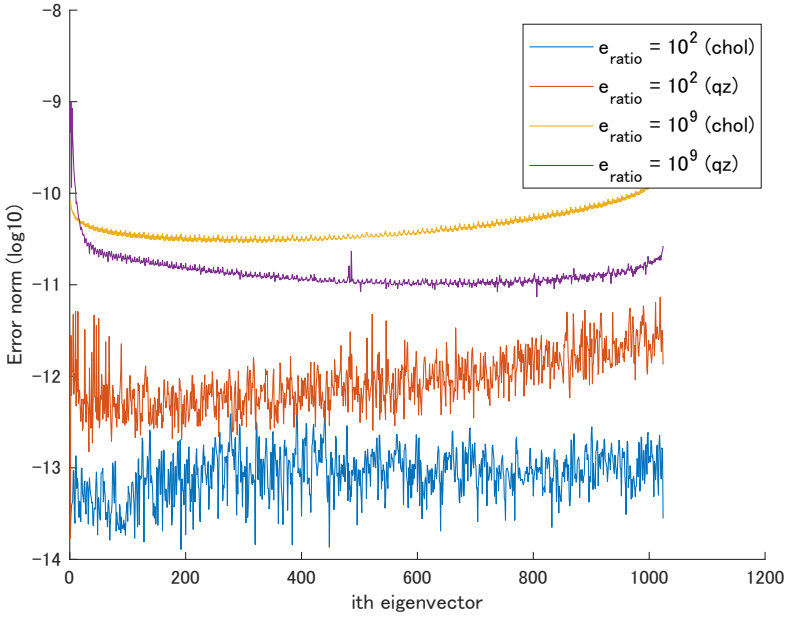

Figure 4: Relative errors for the computed eigenvalues (18).

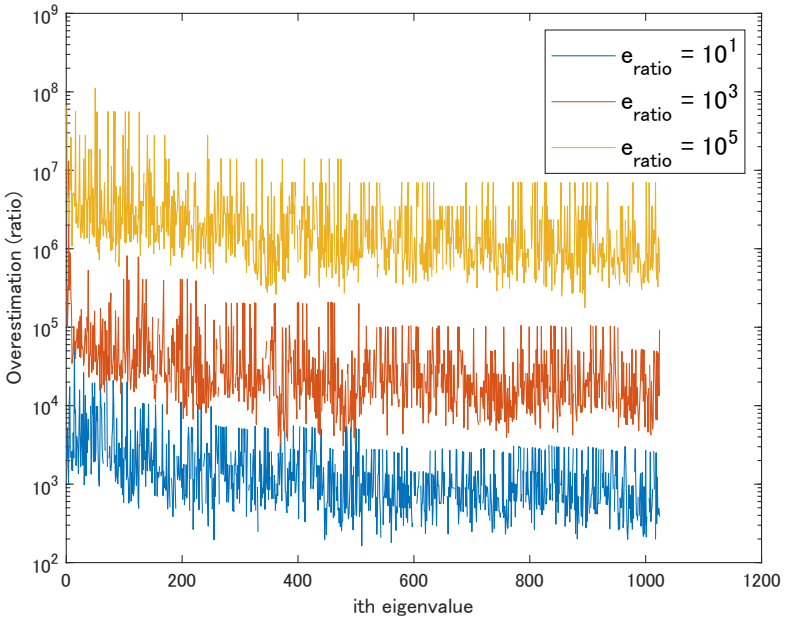

Figure 5: Ratio of overestimation of the error bound.

where $e = (1, 1, \ldots, 1)^T$ and $\|G\|_\infty < 1$ is assumed. If $[\tilde{\lambda}_i - r_i, \tilde{\lambda}_i + r_i]$ is isolated, r_i is the error bound for $\tilde{\lambda}_i$ (see [2,4] for details). The vector r can be computed using only floating-point arithmetic, where [8] gives an example for interval arithmetic and [11] gives a method for interval matrix multiplication. If we generate a test set for the generalized eigenvalue problem, we can determine the exact eigenvalues.

We obtain approximate eigenvalues and eigenvector matrix by the MATLAB function eig, and generate the exact eigenvalues d_{ii} by the proposed method. We checked the overestimation of r_i by comparing

$$\frac{r_i}{|\hat{d}_{ii} - d_{ii}|}. \tag{19}$$

Figure 5 shows the quantity in (19) for $n = 1{,}024$. We set $(1, 2, 3, \ldots, 1{,}024)^T$ as expected eigenvalue and $\mathrm{cnd}_B = 10^1, 10^3, 10^5$. Figure 5 indicates that the overestimation ratio is strongly related to cnd_B.

6 Conclusion

In this article, we proposed a method for generating generalized eigenvalue problems such that the exact eigenvalues and eigenvectors are known. All elements of the matrices A and B are represented exactly using floating-point numbers. Therefore, the proposed method is useful for verifying the accuracy of numerical computations. Using this method, we can monitor the accuracy differences between algorithms and assess the extent of overestimation in verified numerical computations.

Acknowledgments: We would like to express our gratitude to the two reviewers for their careful reading and valuable comments.

Funding information: This work was supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI Grant Number 23H03410.

Author contributions: All algorithms, analyses, numerical experiments, and manuscript preparation were conducted collaboratively, with all authors contributing equally.

Conflict of interest: The authors declare no conflict of interest.

Data availability statement: The code used in this study is provided in the appendix and is freely available for use.

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, et al., *LAPACK Users' Guide*, SIAM, 1999.

[2] T. Hoshi, T. Ogita, K. Ozaki, and T. Terao, *An a posteriori verification method for generalized real-symmetric eigenvalue problems in large-scale electronic state calculations*, J. Comput. Appl. Math. **376** (2020), 112830, DOI: https://doi.org/10.1016/j.cam.2020.112830.

[3] IEEE Standard for Floating-Point Arithmetic. ANSI/IEEE STD. 754-2008, 2008.

[4] S. Miyajima, T. Ogita, and S. Oishi, *A method of generating linear systems with an arbitrarily ill-conditioned matrix and an arbitrary solution*, Proceeding of International Symposium on Nonlinear Theory and its Applications, Paper presented at International Symposium on Nonlinear Theory and its Applications, Bruges, Belgium, 2005, Oct. 18–21, pp. 741–744.

[5] S. Miyajima, T. Ogita, S. M. Rump, and S. Oishi, *Numerical enclosure for each eigenvalue in generalized eigenvalue problem*, Rel. Comput. **14** (2010), 24–45.

[6] A. Minamihata, K. Ozaki, T. Ogita, and S. Oishi, *Improved extraction scheme for accurate floating-point summation*, Proceedings of The 35th JSST Annual Conference: International Conference on Simulation Technology, Paper presented at The 35th JSST Annual Conference (Kyoto, Japan), 2016, Oct. 27–29, Kyoto.

[7] Y. Nakatsukasa, *Algorithms and perturbation theory for matrix eigenvalue problems and the singular value decomposition*, PhD thesis, University of California, Davis, CA, 2011.

[8] A. Neumaier, *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, 1990, DOI: https://doi.org/10.1017/CBO9780511526473.

[9] K. Ozaki and T. Ogita, *Generation of linear systems with specified solutions for numerical experiments*, Rel. Comput. **25** (2017), 148–167.

[10] K. Ozaki and T. Ogita, *Generation of test matrices with specified eigenvalues using floating-point arithmetic*, Numer. Algor. **90** (2022), 1–22, DOI: https://doi.org/10.1007/s11075-021-01186-7.

[11] S.M. Rump, *Fast and parallel interval arithmetic*, BIT Numer. Math. **39** (1999), 539–560, DOI: https://doi.org/10.1023/A:1022374804152.

[12] S. M. Rump, *Verification methods: rigorous results using floating-point arithmetic*, Acta Numer. **19** (2010), 287–449, DOI: https://doi.org/10.1017/S096249291000005X.

[13] V. Strassen, *Gaussian elimination is not optimal*, Numer. Math. **13** (1969), 354–356, DOI: https://doi.org/10.1007/BF02165411.

[14] J. J. Sylvester, *Thoughts on inverse orthogonal matrices, simultaneous signsuccessions, and tessellated pavements in two or more colours, with applications to Newton's rule, ornamental tile-work, and the theory of numbers*, Lond. Edinb. Dubl. Phil. Mag. **34** (1867), 461–475, DOI: https://doi.org/10.1080/14786446708639914.

[15] S. Winograd, *On multiplication of* 2×2 *matrices*, Linear Algebra Appl. **4** (1971), 381–388, DOI: https://doi.org/10.1016/0024-3795(71)90009-7.

Appendix

Here, we present an example of MATLAB code.

```
01:function [A,B,v1,v2,X] = test_geig(d,e)
02:% Generating a test problem:
03:%     A X = B X diag(v1+v2)
04:% d    : expected eigenvalue
05:% e    : diag(E) in (12)
06:% v1+v2: exact eigenvalue
07:%    X : very accurate or exact eigenvector matrix
08:
09:%% set constants
10:n = length(d);
11:n2 = 2n^extpow2(n);
12:u = 2-^53; % for binary64
13:Q = hadamard(n);
14:%% Matrix B
15:if 2*u*sum(e)2 > min(e)^2^
16:   error('generation of B may be failed')
17:end
18:B = (Q.*e') * (e.*Q);
19:% Matrix A
20:d2 = d / n;
21:sigma = 0.75*n2*2nextpow2(max(abs(d2))) * (sum(e)/min(e))^2^;
22:v = (sigma + d2) - sigma;
23:if min(abs(v)) == 0
24:   error('zero element')
25:end
26:A = Q.*e'*Q.*v'*(Q.*e')*Q;
27:% Eigenvalues
28:[v1,v2] = TwoProduct(v,n);
29:%% Eigenvector matrix
20:X = (Q .* (1./e)' * Q) / n;
30:end
```

The matrix computations on lines 18 and 26 involve products with the Hadamard matrix, allowing for simplification [10]. Additionally, iterative refinement of sigma on line 21 brings it closer to the target eigenvalue.