Research Article Computational Mathematics and Computer Modeling with Applications

Received 13 December 2021
Accepted 31 December 2021

DOI: 10.52547/CMCMA.1.1.21
AMS Subject Classification: 15A18; 15A29; 65F18

Solving parameterized generalized inverse eigenvalue problems via Golub-Kahan bidiagonalization

Zeynab Dalvand[a] **and Mohammad Ebrahim Dastyar**[b]

In this study, we present two two-step methods to solve parameterized generalized inverse eigenvalue problems that appear in diverse areas of computation and engineering applications. At the first step, we transfer the inverse eigenvalue problem into a system of nonlinear equations by using of the Golub-Kahan bidiagonalization. At the second step, we use Newton's and Quasi-Newton's methods for the numerical solution of system of nonlinear equations. Finally, we present some numerical examples which show that our methods are applicable for solving the parameterized inverse eigenvalue problems.

Keywords: Parameterized generalized inverse eigenvalue problem; Golub-Kahan bidiagonalization; Nonlinear equations; Newton's method.

1. Introduction

In this paper, we need some notations which are described in the following sentences. We use the symbol $\mathbb{R}^{n\times n}$ to denote the set of all real $n \times n$ matrices. We apply I_n to show the $n \times n$ identity matrix, e_j to represent the j-th column vector of I_n and $I^j = (e_1, e_2, \ldots, e_j) \in \mathbb{R}^{n\times j}$. The symbols A^T, $\det(A)$ and $\kappa(A)$ denote the transpose, the determinant and the condition number of a matrix A, respectively. We show the set of all $n \times n$ upper-bidiagonal matrices by using $\mathcal{U}_n$. Let $\|.\|_2$ denotes the Euclidean norm for vectors and induced norms for matrices.
Let $A(c)$ and $B(c)$ be real $n \times n$ matrix-valued functions depending on $c = (c_1, c_2..., c_n)^T \in \mathbb{R}^n$ and F represents the scalar field of real $\mathbb{R}$. If we have a set of scalars $\Omega \in$ F, then the parameterized inverse eigenvalue problem (PIEP) is defined as follows:

Problem 1 *Assume that a set of real numbers $\{\lambda_1, \lambda_2, ..., \lambda_n\} \subseteq \Omega$ and $n \times n$ the matrices $A_i \in \mathbb{R}^{n\times n}(i = 0, 1, 2, \ldots, n)$ are given. Find a vector $c \in \mathbb{R}^n$ such that the matrix $A(c) = A_0 + \sum_{i=1}^{i=n} c_i A_i$ has the given eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$.*

In this paper we consider the following extended form of the PIEP, which is called the parameterized generalized inverse eigenvalue problem (PGIEP).

Problem 2 *Assume that a set of distinct real numbers $\{\lambda_1, \lambda_2, ..., \lambda_n\}$ and the matrices $A_i, B_i \in \mathbb{R}^{n\times n}(i = 0, 1, 2, \ldots, n)$ are given. Find vector $c \in \mathbb{R}^n$ such that the generalized eigenvalue problem $A(c)x = \lambda B(c)x$ has the prescribed eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$, where matrices $A(c)$ and $B(c)$ are affine in c as follow:*

$$A(c) = A_0 + \sum_{i=1}^{i=n} c_i A_i, \quad B(c) = B_0 + \sum_{i=1}^{i=n} c_i B_i, \tag{1}$$

[a] *Department of Applied Mathematics, Faculty of Mathematical Sciences, Shahid Beheshti University, Tehran, Iran. Email: z_dalvand@sbu.ac.ir, zeynab.dalvand137@gmail.com,*
[b] *Department of Mathematics, Faculty of Mathematical Sciences, Tarbiat Modares University, Tehran, Iran. Email: m.e.dastyar@modares.ac.ir*
* *Correspondence to: Z. Dalvand.*

A special type of Problems 2 is the algebraic inverse eigenvalue problem [49, 32, 43] which $A_0, A_2, ..., A_n$ are real symmetric matrices, $B_0 = I$ and $B_i = 0$ for $i = 1, ..., n$.
The additive inverse eigenvalue problems [14, 22] are included in the formulation of Problem 2 with the following matrices

$$A_0 = A, \quad A_i = e_i e_i^T, \ i = 1, 2, ..., n,$$

$$B_0 = I, \quad B_i = 0, \ i = 1, 2, ..., n,$$

in which A is a constant matrix. In [39], Yuan *et al.* considered a parameterized inverse eigenvalue problem which B_i for $i = 1, ..., n$, are bisymmetric matrices. Other cases of the PGIEP is presented by Du *et al.* [21].

PGIEP is one of the most frequent inverse eigenvalue problems that arises in variant areas of engineering applications, among which we can refer to vibrating string [42], structure design and applied mechanics [33, 6]. PGIEP also play an important role in the study of the nuclear spectroscopy and molecular spectroscopy [13]. In addition, these problems have numerous applications in numerical computing, e.g., inverse Sturm-Liouville problems [5, 44], the factor analysis [35] and the educational testing problems[3]. The existence of these problems in the design of control systems is given in reference [12].

In the recent decades, many researchers have focused their attention on specific categories of parameterized inverse eigenvalue problems [30, 14, 32]. There are a lot of articles on numerical methods and existence theory for different categories of parameterized inverse eigenvalue problems [47, 28, 26, 27, 18, 20, 19]. Using Brouwers fixed-point theorem, sufficient conditions for the solvability of Problem 1 and 2 are presented in [9], and Alexander reported sufficient conditions by employing topological degree for the solvability of this problem [4]. Sufficient conditions for the positive solution of the algebraic inverse eigenvalue problem are provided in [12]. In [46], the sufficient conditions for the solvability of algebraic inverse eigenvalue problems are presented. In [47], Xu investigated the necessary conditions for the solvability of algebraic inverse eigenvalue problems. In [29], Ji presented the sufficient conditions for guaranteeing the existence of a solution of the parameterized inverse eigenvalue problem and then Dai *et al.* expanded the conditions for the PGIEP [16]. Also, a number of sufficient conditions for the existence of the solution of the symmetric nonnegative inverse eigenvalue problem have been collected and compared in [34].More resources in this area can be seen in [38, 41, 45, 31].

Moreover, there are more numerical methods to find the solution for the parameterized generalized inverse eigenvalue problems [36, 17, 40]. One of the developed ideas is the use of iterative methods for solving the parameterized inverse eigenvalue problem. Most existing iterative methods for solving these problems, lead to a system of nonlinear equations. In some of these methods by assuming that the eigenvalues of the generalized eigenvalue problem $A(c)x = \lambda B(c)x$ are shown by $\lambda_1(c) \leq \lambda_2(c) \leq ... \leq \lambda_n(c)$ and the eigenvalues given in the problem are denoted $\lambda_1 < \lambda_2 < ... < \lambda_n$, we need to solve $F(c) = 0$, where

$$F(c) = \begin{pmatrix} \lambda_1(c) - \lambda_1 \\ \lambda_2(c) - \lambda_2 \\ \vdots \\ \lambda_n(c) - \lambda_n \end{pmatrix} = 0, \tag{2}$$

or

$$F(c) = \begin{pmatrix} \det(A(c) - \lambda_1 B(c)) \\ \det(A(c) - \lambda_2 B(c)) \\ \vdots \\ \det(A(c) - \lambda_n B(c)) \end{pmatrix} = 0. \tag{3}$$

The Newton's method is proposed for solving the system of nonlinear equations (2) by Dai and Lancaster [17], by expanding the ideas which were developed by Friedland in [24]. This method requires the computation of all eigenvalues and eigenvectors of the problem $A(c)x = \lambda B(c)x$ in each iteration of the Newton's method. Also in [40], Shu and Li introduced a homotopy method for solving (3). Different Newton-like methods are give for solving (2) in [7], such as the Cayley transform method and the inexact Cayley transform method. Since, the Cayley transform takes a lot arithmetic operations to produce an orthogonal matrix in each iteration, in [1], an algorithm based on matrix equations was presented for inverse eigenvalue problems, which it can refine orthogonality without the Cayley transform and reduces the operations of the Cayley transform in each iteration. In addition, Aishima improved this algorithm by the using of an optimization problem for the eigenvectors associated with multiple eigenvalues [2]. In 1996, Xu [48] provided a method based on the properties of the smallest singular value of matrices.

Most of the iterative methods, which provided to solve inverse eigenvalue problems are computationally complex, because they need to solve a eigenvalue problem in each iteration. In this paper, two numerical algorithms based on the bidiagonalization are proposed to avoid solving a eigenvalue problem in each iteration and decrease some computational complexity. The remainder of this paper is organized as follows. In Section 2, first a description of the bidiagonal factorization for a matrix-valued function is given and then two iterative methods are constructed for solving Problem 2. In Section 3, the numerical examples illustrate the numerical behavior of the proposed methods. We end the paper by a brief concluding in Section 4.

2. Main Results

In this section, we offer two iterative algorithms based on the Golub-Kahan bidiagonalization for solving Problem 2.

2.1. Bidiagonalization

In this subsection, we briefly recall the bidiagonal factorization. Bidiagonal factorization is a two-sided orthogonal factorization [25]. Let $A \in \mathbb{R}^{n\times n}$, the main idea of the bidiagonal factorization of the matrix A is computing the orthogonal matrices U_B $(n-by-n)$ and V_B $(n-by-n)$ such that

$$U_B^T A V_B = B, \tag{4}$$

in which B is upper bidiagonal matrix in the following form

$$B = \begin{pmatrix} \alpha_1 & \beta_1 & 0 & \dots & 0 \\ 0 & \alpha_2 & \beta_2 & & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & & \alpha_{n-1} & \beta_{n-1} \\ 0 & \dots & & & \alpha_n \end{pmatrix},$$

and $U_B = U_1 \dots U_n, \quad V_B = V_1 \dots V_{n-2}$, where each of V_i, U_i could be Householder or Givens matrices. Using these matrices, the matrix A is transformed in the following way:

$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \overset{U_1}{\rightarrow} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix} \overset{V_1}{\rightarrow} \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix} \overset{U_2}{\rightarrow}$$

$$\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} \overset{V_2}{\rightarrow} \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} \overset{U_3}{\rightarrow} \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$$

In general, U_i creates zeros elements in i-th column and V_i creates zeros elements in i-th row. Since matrices U_B and V_B are orthogonal matrices, Eq.(4) leads to

$$|\det(A)| = |\det(B)|.$$

2.2. Bidiagonal factorization for a matrix-valued function

Since matrices $A(c)$ and $B(c)$ are matrix-valued functions depending on several parameters, we can not directly use the bidiagonal factorization. So, in this subsection, we present a smooth bidiagonal factorization for a differentiable matrix-valued function of multiple parameters.
Let $Q(c) = (q_{kl}(c)) \in \mathbb{C}^{n\times n}$ be differentiable matrix defined on F, which F is a connected open subset of $\mathbb{C}^n$ such that

$$Q(c) = Q(c^{(0)}) + \sum_{i=1}^{n} \frac{\delta Q(c^{(0)})}{\delta c_i}(c_i - c_i^{(0)}) + o(\|c - c^{(0)}\|_2), \tag{5}$$

where

$$\frac{\delta Q(c^{(0)})}{\delta c_i} = \frac{\delta q_{kl}(c^{(0)})}{\delta c_i}|_{c=c^{(0)}} \in C^{n\times n}. \tag{6}$$

In the following theorem, we offer an existence result for bidiagonal factorization of the matrix-valued function $Q(c)$.

Theorem 1 *Let $Q(c) \in \mathbb{R}^{n\times n}$ be a differentiable matrix defined on a connected open subset $\mathbf{F} \subseteq \mathbb{R}^n$ such that at a given point $c^{(0)}$, rank $Q(c^{(0)}) \geq n-1$. Assume that there exist two orthogonal matrices U_T and V_T such that $U_T^T Q(c^{(0)}) V_T$ has a bidiagonal factorization*

$$U_T^T Q(c^{(0)}) V_T = T_0(c^{(0)}),$$

where

$$T_0 = \begin{pmatrix} T_{11} & T_{12} \\ 0 & t_{nn} \end{pmatrix}.$$

Then there exists a neighborhood of $c^{(0)}$ as $\mathbf{N}(c^{(0)}) \subseteq \mathbf{F}$, such that for all $c \in \mathbf{N}(c^{(0)})$ the matrix-valued function $Q(c)$ has a bidiagonal factorization as:

$$U_{T_c}^T Q(c) V_{T_c} = T(c), \quad for \quad c \in \mathbf{N}(c^{(0)}),$$

where

$$T(c) = \begin{pmatrix} T_{11}(c) & T_{12}(c) \\ 0 & t_{nn}(c) \end{pmatrix}.$$

In addition it holds that

$$t_{nn}(c) = t_{nn}(c^{(0)}) + e_n^T U_T^T \sum_{i=1}^{n} \frac{\delta Q(c^{(0)})}{\delta c_i} V_T (e_n - I_{n-1} T_{11}^{-1} T_{12})(c_i - c_i^{(0)}) + o(\|c - c^{(0)}\|_2). \tag{7}$$

Proof. By using Eq. (5) and

$$Q_d(c) := \sum_{i=1}^{n} \frac{\delta Q(c^{(0)})}{\delta c_i}(c_i - c_i^{(0)}) + o(\|c - c^{(0)}\|_2),$$

we obtain

$$Q(c) = Q(c^{(0)}) + Q_d(c),$$

and

$$U_T^T Q(c) V_T = U_T^T Q(c^{(0)}) V_T + U_T^T Q_d(c) V_T := T_0 + \widetilde{Q}(c),$$

where

$$\widetilde{Q}(c) = U_T^T \sum_{i=1}^{n} \frac{\delta Q(c^{(0)})}{\delta c_i} V_T (c_i - c_i^{(0)}) + o(\|c - c^{(0)}\|_2),$$

$$T_0 = \begin{pmatrix} T_{11} & T_{12} \\ 0 & t_{nn} \end{pmatrix}.$$

By considering

$$\widetilde{Q}(c) = \begin{pmatrix} \widetilde{Q}_{11}(c) & \widetilde{Q}_{12}(c) \\ \widetilde{Q}_{21}(c) & \widetilde{q}_{nn}(c) \end{pmatrix},$$

where $T_{11} \in \mathcal{U}_{n-1}$ and $\widetilde{Q}_{11}(c) \in C^{(n-1)\times(n-1)}$, we define the matrix-valued function $U_1(c)$ as

$$U_1(c) := \begin{pmatrix} I & -\widetilde{Q}_{21}(c)(T_{11} + \widetilde{Q}_{11}(c))^{-1} \\ 0 & 1 \end{pmatrix}.$$

Simple computations show that

$$U_1^T U_T^T Q(c) V_T^T = \begin{pmatrix} \widetilde{T}_{11}(c) & \widetilde{T}_{12}(c) \\ 0 & t_{nn}(c) \end{pmatrix},$$

where

$$\widetilde{T}_{11}(c) = T_{11}(c^{(0)}) + \widetilde{Q}_{11}(c), \quad \widetilde{T}_{12}(c) = T_{12}(c^{(0)}) + \widetilde{Q}_{12}(c)$$

and

$$\begin{aligned} t_{nn} &= t_{nn}(c^{(0)}) + \widetilde{q}_{nn}(c) - \widetilde{Q}_{21}(c)(T_{11}(c^{(0)}) + \widetilde{Q}_{11}(c))^{-1}(T_{12}(c^{(0)}) + \widetilde{Q}_{12}(c)) \\ &= t_{nn}(c^{(0)}) + e_n^T U_T^T \sum_{i=1}^{n} \frac{\delta Q(c^{(0)})}{\delta c_i} V_T (e_n - I_{n-1} T_{11}^{-1} T_{12})(c_i - c_i^{(0)}) + o(\|c - c^{(0)}\|_2). \end{aligned} \tag{8}$$

Now the proof is finished. □

2.3. Two new algorithms to solve Problem 2

In this subsection, we present two algorithms based on the bidiagonalization for solving special categories of the parameterized generalized inverse eigenvalue problems.
At first, we introduce a system of nonlinear equations which is equivalent to the PGIEP. For this purpose, we compute the bidiagonal factorization of $A(c) - \lambda_i B(c) (i = 1, \ldots, n)$ by using Householder matrices as follows:

$$U_i^T(c)(A(c) - \lambda_i B(c)) V_i(c) = T_i(c), \quad i = 1, \ldots, n, \tag{9}$$

where $U_i^T(c)$ and $V_i(c)$ are orthogonal matrices, and

$$T_i = \begin{pmatrix} T_{11}^{(i)} & T_{12}^{(i)} \\ 0 & t_{nn}^{(i)} \end{pmatrix}.$$

The determinant of $T_i(c)$ is equal to the determinant of the left hand side matrix in Eq.(9). Afterwards, if

$$t_{nn}^{(i)}(c) = 0, \quad i = 1, 2, \ldots, n,$$

then the generalized eigenvalue problem $A(c)x = \lambda B(c)x$ has the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. So, for solving the parameterized inverse eigenvalue problem, we create the following system of nonlinear equations:

$$F(c) = \begin{pmatrix} t_{nn}^{(1)}(c) \\ t_{nn}^{(2)}(c) \\ \cdots \\ t_{nn}^{(n)}(c) \end{pmatrix} = 0. \tag{10}$$

Now we use the Newton method to solve the system of the nonlinear equations (10). Suppose flow iterate $c^{(k)}$ be sufficiently close to a solution of the nonlinear system (10), then one step of Newton method for the solution of (10) is as follow:

$$J_F(c^{(k)})(c^{(k+1)} - c^{(k)}) = -F(c^{(k)}), \tag{11}$$

where $J_F(c^{(k)})$ is the Jacobian matrix of the nonlinear function (10) by following form

$$J_F(c^{(k)}) = \begin{pmatrix} \frac{\partial t_{nn}^{(1)}(c)}{\partial c_1} & \cdots & \frac{\partial t_{nn}^{(1)}(c)}{\partial c_n} \\ \frac{\partial t_{nn}^{(2)}(c)}{\partial c_1} & \cdots & \frac{\partial t_{nn}^{(2)}(c)}{\partial c_1} \\ \cdots & & \cdots \\ \frac{\partial t_{nn}^{(n)}(c)}{\partial c_1} & \cdots & \frac{\partial t_{nn}^{(n)}(c)}{\partial c_1} \end{pmatrix}, \tag{12}$$

By replacing $(A(c) - \lambda_i B(c))$ instead of $Q(c)$ in Eq. (8), it follows from Theorem 1 that we can conclude that $t_{nn}^{(i)}$ is calculated as

$$t_{nn}^{(i)}(c) = t_{nn}^{(i)}(c^{(0)}) + e_n^T (U^i)^T (\sum_{j=1}^{n} (A_j - \lambda_i B_j))(V^i)(e_n^T - I_{n-1} {T_{11}^i}^{-1} T_{12}^i)(c_j - c_j^{(0)}) \tag{13}$$

$$+ o(\|c - c^{(0)}\|_2), \tag{14}$$

where $(A_j - \lambda_i B_j) = \frac{\partial (A(c^{(k)}) - \lambda_i B(c^{(k)}))}{\partial c_j}$. On the other hand, using Taylor series, we can get

$$t_{nn}^{(i)}(c) = t_{nn}^{(i)}(c^{(k)}) + \sum_{j=1}^{n} \frac{\partial t_{nn}^{(i)}(c^{(k)})}{\partial c_j}(c_j - c_j^{(k)}) + O(\|c - c^{(0)}\|_2^2) \tag{15}$$

Using Eq. (13) and (15), and similar to [16], we obtain $\frac{\partial t_{nn}^{(i)}(c^{(k)})}{\partial c_j}$ as

$$\frac{\partial t_{nn}^{(i)}(c^{(k)})}{\partial c_j} = e_n^T {U^i}^T (c^{(k)})(A_j - \lambda_i B_j) V^i(c^{(k)})(e_n^T - I_{n-1} {T_{11}^i}^{-1} T_{12}^i). \tag{16}$$

From the above discussion, a new method for solving the Problem 2 can be resumed in the following algorithm.

Algorithm 1 *(The algorithm for finding a solution of the Problem 2)*
Choose an initial guess $c^{(0)}$ and, for $k = 0, 1, 2, \ldots$ until $\|F(c^{(k)})\|_2$ is small enough;

1. *Compute $A(c^{(k)}) - \lambda_i B(c^{(k)})(i = 1, \ldots, n)$ and $A_j - \lambda_i B_j (i = 1, \ldots, n, j = 1, \ldots, n,)$;*

2. *Compute bidiagonal factorization of $A(c^{(k)}) - \lambda_i B(c^{(k)})$:*

$$U^{(i)^T}(c^{(k)})(A(c^{(k)}) - \lambda_i B(c^{(k)}))V^{(i)}(c^{(k)}) = T_i(c^{(k)}), \quad i = 1, \ldots, n;$$

3. *Compute $F(c^{(k)})$and $J_F(c^{(k)})$ using (10) and (16);*

4. *Find $c^{(k+1)}$ by solving the Newton equation (11);*

The computational cost of each step of Algorithm 1 is as follows: Step 1 is calculated using $n^4 + 3n^3$ flops, the bidiagonal factorization of $A(c^{(k)}) - \lambda_i B(c^{(k)})$ needs $\frac{4}{3}n^3$ flops (see, e.g.,[25]), so that Step 2 needs$\frac{4}{3}n^4$ flops; Step 3 requires approximately n^4 flops; and Step 4 requires $\frac{2}{3}n^3$ flops. In summary, the total cost of Algorithm 1 is about $\frac{1}{3}(10n^4 + 11n^3)$ flops. Note that the algorithms proposed in [16] and [17] require approximately $\frac{1}{3}(8n^4 + 11n^3)$ and $\frac{1}{3}(2n^4 + 35n^3)$ flops per iteration, respectively. However, some of our numerical tests showed that Algorithm 1 took generally less iterations than algorithms proposed in [16].

Since in Algorithm 1 the Jacobian matrix $J_F(c)$ and its inverse should be calculated at each iteration, we now use the Quasi-Newton method to avoids the direct computation of the Jacobian matrix. Burden and Faires developed the Quasi-Newton methods for approximating the Jacobian matrix at each iteration [11]. One of Quasi-Newton method types is Broyden's method [10]. Broyden's iterative procedure is defined as

$$c^{(k+1)} = c^{(k)} - H_k^{-1} F(c^{(k)}) \tag{17}$$

where the matrix H_k is defined as

$$H_k = H_{k-1} + \frac{y_k - H_{k-1}s_k}{\|s_k\|_2^2} s_k^T, \tag{18}$$

in which $y_k = F(c_{k-1}) - F(c_{k-1})$, $s_i = c^{(k)} - c^{(k-1)}$. Using the Sherman-Morrison formula [37], the matrix H_k^{-1} can be calculated easily as

$$\begin{aligned} H_k^{-1} &= (H_{k-1} + \frac{y_k - H_{k-1}s_k}{\|s_k\|_2^2})^{-1} \\ &= H_{k-1}^{-1} - \frac{H_{k-1}^{-1}(H_{k-1} + \frac{y_k - H_{k-1}s_k^t}{\|s_k\|_2^2}s_k)H_{k-1}^{-1}}{1 + s_k^t H_{k-1}^{-1}(\frac{y_k - H_{k-1}s_k}{\|s_k\|_2^2})} \\ &= H_{k-1}^{-1} - \frac{(H_{k-1}^{-1}y_k - s_k)s_k^t H_{k-1}^{-1}}{\|s_k\|_2^2 + s_k^t H_{k-1}^{-1}y_k - \|s_k\|_2^2}. \end{aligned}$$

Therefore, an approximation of the inverse matrix H_k is calculated in each iteration using

$$H_k^{-1} = H_{k-1}^{-1} + \frac{(s_k - H_{k-1}^{-1}y_k)s_k^t H_{k-1}^{-1}}{s_k^t H_{k-1}^{-1}y_k}. \tag{19}$$

Now, by using the Broyden's method we propose a new algorithm to solve Problem 2.

Algorithm 2 *(The algorithm for finding a solution of the Problem 2).*
Choose an initial guess $c^{(0)}$ and, for $k = 0, 1, 2, \ldots$ until $\|F(c^{(k)})\|_2$ is small enough;

1. *Compute $A(c^{(k)}) - \lambda_i B(c^{(k)})(i = 1, \ldots, n)$ and $A_j - \lambda_i B_j (i = 1, \ldots, n, j = 1, \ldots, n,)$;*

2. *Compute bidiagonal factorization of $A(c^{(k)}) - \lambda_i B(c^{(k)})$:*

$$U^{(i)^T}(c^{(k)})(A(c^{(k)}) - \lambda_i B(c^{(k)}))V^{(i)}(c^{(k)}) = T_i(c^{(k)}), \quad i = 1, \ldots, n;$$

3. *Compute $F(c^{(k)})$ using (10)*

4. *Compute*

$$H_k^{-1} = \begin{cases} H_{k-1}^{-1} + \frac{(s_k - H_{k-1}^{-1}y_k)s_k^t H_{k-1}^{-1}}{s_k^t H_{k-1}^{-1}y_k}, & \text{for } k = 1, 2, \ldots., \\ J_F^{-1}(c^{(0)}), & \text{for } k = 0. \end{cases} \tag{20}$$

7. *Compute $c^{(k+1)}$ using Broyden's iterative (17).*

3. COMPUTED EXAMPLES

We have tested both algorithms on several kinds of parameterized inverse eigenvalue problems. In this section, we present five examples of our experiments with Algorithms 1 and 2 to illustrate the efficiency of these algorithms. Also, we provide a comparison between Algorithms 1 and 2 and the algorithms contained in [16, 17, 15, 48]. The iterations were stopped when the norm $\|f(c^{(k)})\|_2$ was less than 10^{-9}. All computations were performed using MATLAB (version 8.5) installed on a computer with Intel i7 CPU 2100 MHz and 8 GB RAM.

Example 1 *[16] In this example, we consider a PGIEP with $n = 2, \lambda_1 = -1, \lambda_2 = 3$,*

$$A_1 = \begin{pmatrix} 1.25 & 1 \\ 1 & 1.25 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1.9 & 0.7 \\ 0.7 & 1.7 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 0.575 & -0.1 \\ -0.775 & 4 \end{pmatrix},$$

and

$$B_0 = \begin{pmatrix} -0.25 & 0 \\ 0 & 0.75 \end{pmatrix}, \quad B_1 = \begin{pmatrix} 0.3 & 0.3 \\ 0.3 & 0.6 \end{pmatrix}, \quad B_2 = \begin{pmatrix} -0.475 & 0 \\ -0.225 & 0 \end{pmatrix}.$$

By applying Algorithm 1 and Algorithm 4.1 in [16], with the different initial vectors $c^{(0)} = (-0.5, 0.5)^T$, $c^{(0)} = (-5, 2)^T$ and $c^{(0)} = (-1.5, 0.5)^T$, we obtain the solutions $c^* = (-0.4530, 0.3612)^T$, $c^* = (-5.1223, 1.8123)^T$ and $c^* = (-2.3724, 0.6855)^T$ of the PGIEP, respectively. It is known that there can be as many as $n!$ different solutions [23] and, in practice, one may wish to search among these for a solution which is optimal in some sense. Figure 1 shows the numerical results obtained from Algorithm 1 with the different initial vectors. The results are illustrated in Table 1.

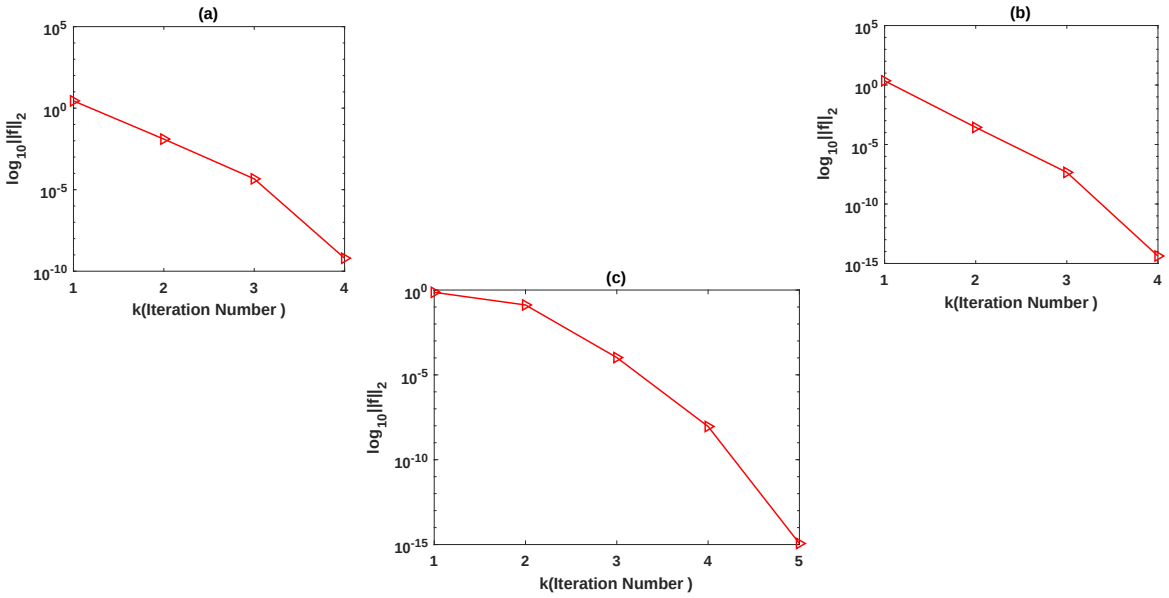

Figure 1. The results obtained for Example 1, (a): initial vector $c^{(0)} = (-0.5, 0.5)^T$, (b): initial vector $c^{(0)} = (-5, 2)^T$ and (c): initial vector $c^{(0)} = (-1.5, 0.5)^T$.

Table 1. Numerical results for Example 1.

	Ite	Algorithm 1		Algorithm 4.1 in [16]	
$c^{(0)}$	k	$\|c^* - c^{(k)}\|$	$\|f(c^{(k)})\|$	$\|c^* - c^{(k)}\|$	$\|f(c^{(k)})\|$
$(-0.5, 0.5)^T$	0	0.146E+00	0.611E+00	0.146E+00	0.611E+00
	1	0.142E-01	0.109E-01	0.494E-02	0.109E-01
	2	0.102E-05	0.750E-05	0.175E-05	0.750E-05
	3	0.201E-13	0.209E-11	0.474E-12	0.209E-11
$(-5, 2)^T$	0	0.224+00	0.226E+01	0.224E+00	0.121E+01
	1	0.403E-01	0.843E-01	0.489E-01	0.637E-02
	2	0.253E-04	0.145E-03	0.231E-03	0.292E-04
	3	0.133E-09	0.302E-09	0.539E-08	0.680E-09
$(-1.5, 0.5)^T$	0	0.891E+00	0.730+00	0.891E+00	0.801E+00
	1	0.237E-01	0.109E-01	0.264E+00	0.159E+00
	2	0.221E-03	0.104E-03	0.682E-01	0.329E-01
	3	0.192E-07	0.905E-08	0.512E-02	0.228E-02
	4	0.144E-14	0.115E-14	0.297E-04	0.131E-04
	5			0.101E-08	0.448E-09

Example 2 *[17] As the second example, we have a parameterized inverse eigenvalue problem in which* $n = 5$,

$$A_0 = \text{diag}(9, 11, 10, 8, 14), \quad B_0 = \text{diag}(11, 13, 15, 11, 10), \quad A_1 = B_1,$$

$$A_2 = \begin{pmatrix} 0 & 2 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 \end{pmatrix},$$

$$A_3 = \begin{pmatrix} 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 3 & 0 & 0 & 0 & -1 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix}, \quad B_3 = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix},$$

$$A_4 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \quad B_4 = \begin{pmatrix} 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix},$$

Table 2. Numerical results for Example 2.

	lte	Algorithm 1			Algorithm 4.1 in [16]			Algorithm 2.1 in [17]	
	k	$\|c^* - c^{(k)}\|$	$\|f(c^{(k)})\|$	K_κ	$\|c^* - c^{(k)}\|$	$\|f(c^{(k)})\|$	K_κ	$\|c^* - c^{(k)}\|$	$\|\lambda(c^{(k)}) - \lambda^*\|$
(a)	0	0.335E+00	0.817E+00	0.300E+02	0.335E+00	0.146E+01	0.300E+02	0.335E+00	0.034E-03
	1	0.101E-01	0.149E-01	0.019E+04	0.623E-01	0.248E+00	0.154E+03	0.744E-01	0.997E-04
	2	0.432E-04	0.447E-04	0.289E+05	0.520E-02	0.240E-01	0.204 E+04	0.450E-04	0.105E-05
	3	0.369E-09	0.452E-09	0.508E+09	0.247E-04	0.129E-04	0.261E+09	0.278E-08	0.497E-09
	4	0.290E-15	0.939E-15	0.080E+16	0.139E-08	0.359E-09	0.796E+12	0.171E-13	0.221E-13
	5				0.100E-13	0.327E-13	0.008E+16		
(b)	0	0.335E+00	0.246E+01	0.221E+02	0.335E+00	0.195E+01	0.221E+02	0.335E+00	0.485E-01
	1	0.393E+00	0.334E+00	0.202E+03	0.121E+01	0.378E+00	0.154E+03	0.676E-01	0.117E-02
	2	0.184E-01	0.195E+01	0.001E+07	0.191E+00	0. 846E-01	0.204E+04	0.694E-03	0.126E-04
	3	0.370E-04	0.684E-04	0.037E+09	0.365E-02	0.173E-02	0.261E+09	0.568E-07	0.108E-08
	4	0.263E-09	0. 389E-09	0.700E+12	0.258E-05	0.738E-06	0.008E+12	0.152E-12	0.710E-12
	5	0.179E-13	0.133E-13	0.244E+16	0.876E-12	0.279E-12	0.008E+16		
(c)	0	0.741E+00	0.158E+00	0.973E+01	-	-	-	0.741E+00	0.158E+00
	1	0.115E+00	0.361E-02	0.079E+03				0.160E+01	0.192E-01
	2	0.220E-02	0.363E-04	0.051E +04				0.115E+00	0.361E-02
	3	0.525E-06	0.105E-07	0.051E+09				0.220E-02	0.363E-04
	4	0.174E-13	0.730E-15	0.345E+16				0.525E-06	0.105E-07
	5							0.108E-12	0.845E-15

$$A_5 = B_5 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$A(c) = A_0 + \sum_{i=1}^{n} c_i A_i, \quad B(c) = B_0 + \sum_{i=1}^{n} c_i B_i.$$

The eigenvalues are given by

$$\lambda^* = (0.43278721102, 0.66366274839, 0.94385900467, 1.10928454002, 1.49235323254)^T.$$

Algorithm 1, considering that the starting vectors

$$c^{(0)} = \begin{cases} (1.25, 1.15, 1.05, 0.9, 0.85)^T, & Case(a), \\ (1.15, 1.15, 1.05, .075, 1.05)^T, & Case(b), \\ (1.1, 1.2, 1.3, 1.4, 1.5)^T, & Case(c), \end{cases}$$

converges to the solution $c^{(*)} = (1, 1, 1, 1, 1)^T$. *The numerical results for Algorithm 1, Algorithm* 4.1 *in [16] and Algorithm* 2.1 *in [17] are displayed in Table 2, where* $K_\kappa = \min_{1 \le i \le n} \kappa(A(c^{(k)}) - \lambda_i B(c^{(k)}))$, *and "lte." represents the iteration number, and "-" denotes the corresponding algorithm fails to converge, respectively. As recent research has shown that the bidiagonal factorization works best for ill-conditioned matrix [8], our implementation showed that Algorithm 1 performs better in these cases. Also, it is considerable that in [17], the system of nonlinear equations* $\lambda(c) - \lambda = 0$ *is solved and all eigenvalues and eigenvectors are calculated in each step of the algorithm.*

Example 3 *In this example, we study a PGIEP with* $n = 5$ *and*

$$\lambda_1 = \delta, \quad \lambda_2 = 1 - \delta, \quad \lambda_3 = 2, \quad \lambda_4 = 3, \quad \lambda_5 = 4,$$

where δ is a constant. Let $\delta = 0.454$ and

$$A_0 = \begin{pmatrix} 30 & 8.4 & -0.4 & 0 & 0 \\ 5.64 & 33.56 & 7.28 & -0.16 & 0 \\ -0.09 & 7.61 & 13.75 & 2.72 & -0.048 \\ 0 & -0.12 & 3.85 & 4.89 & 0.624 \\ 0 & 0 & -0.03 & 0.9625 & 2.154 \end{pmatrix},$$

$$B_0 = \begin{pmatrix} 13.007181 & 3.997188 & 0 & 0 & 0 \\ 2.498594 & 15.007181 & 3.997636 & 0 & 0 \\ 0 & 3.997636 & 5.007181 & 1.799363 & 0 \\ 0 & 0 & 2.498938 & 0.507181 & 0.600012 \\ 0 & 0 & 0 & 0.900024 & -0.892819 \end{pmatrix},$$

$$A_1 = \begin{pmatrix} 15 & 5 & 0 & 0 & 0 \\ 2.64 & 0.88 & 0 & 0 & 0 \\ 0.09 & 0.03 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -0.5015 & -0.005 & 0 & 0 & 0 \end{pmatrix}, \quad B_1 = I,$$

$$A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 16 & 4 & 0 & 0 \\ 0 & 3.64 & 0.91 & 0 & 0 \\ 0 & 0.04 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$A_3 = \begin{pmatrix} 0 & 0 & 0.1 & 0.025 & 0 \\ 0 & 0 & 0.02 & 0.05 & 0 \\ 0 & 0 & 6 & 1.5 & 0 \\ 0 & 0 & 1.88 & 0.47 & 0 \\ 0 & 0 & 0.01 & 0.0025 & 0 \end{pmatrix}, \quad B_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$A_4 = \begin{pmatrix} 0 & 0 & 0 & -0.1 & -0.02 \\ 0 & 0 & 0 & 0.02 & 0.004 \\ 0 & 0 & 0 & 0.01 & 0.002 \\ 0 & 0 & 0 & 2 & 0.4 \\ 0 & 0 & 0 & 0.48 & 0.094 \end{pmatrix}, \quad B_4 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$A_5 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0.15 \\ 0 & 0 & 0 & 0 & -0.05 \\ 0 & 0 & 0 & 0 & 0.01 \\ 0 & 0 & 0 & 0 & 0.01 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad B_5 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0.2 & 0 \end{pmatrix}.$$

Algorithm 1, with the starting vector $c^{(0)} = (-4.5, -1, -2, -6, -12)^T$, converges to a solution

$$c^{(*)} = (-4.6114, -1.0023, -2.0008, -6.2871, -11.7904)^T.$$

The numerical results are displayed in Table 3. Moreover, the numerical results by Algorithm 1 and Algorithm 4.1 described in [16] are observed in Figure 2.

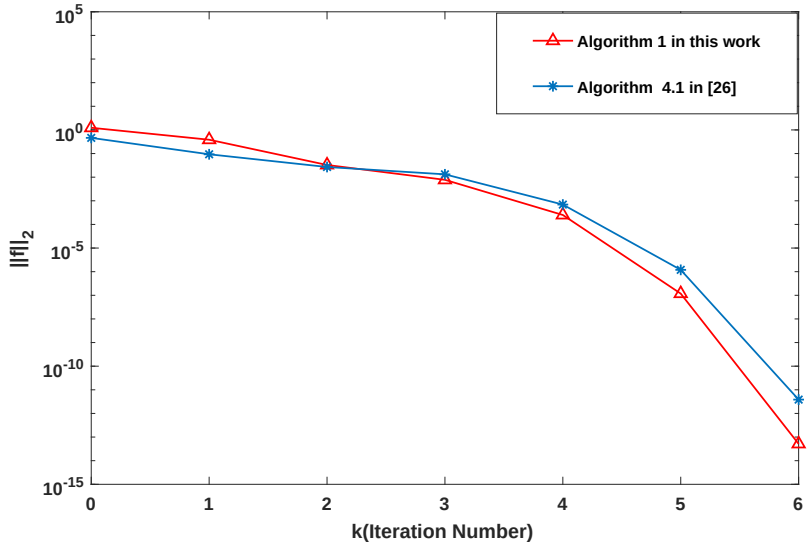

Figure 2. The results obtained for Example 3.

Table 3. Numerical results for Example 3.

Iteration	Algorithm 1			Algorithm 4.1 in [16]		Algorithm 3.1 in [15]	
k	$\|c^{(k+1)} - c^{(k)}\|$	$\|c^* - c^{(k)}\|$	$\|f(c^{(k)})\|$	$\|c^* - c^{(k)}\|$	$\|f(c^{(k)})\|$	$\|c^* - c^{(k)}\|$	$\|f(c^{(k)})\|$
0	0.142E+00	0.372E+00	0.256E+00	0.37E+00	0.46E+00	0.37E+00	0.45E+00
1	0.282E+00	0.195E+00	0.416E-01	0.51E+00	0.93E-01	0.45E+00	0.92E-01
2	0.771E-01	0.878E-01	0.280E-03	0.18E+00	0.27E-01	0.16E+00	0.23E-01
3	0.117E-01	0.117E-01	0.189E-05	0.11E-01	0.13E-01	0.92E-02	0.11E-01
4	0.821E-05	0.822E-04	0.169E-07	0.97E-03	0.69E-03	0.67E-03	0.46E-03
5	0.803E-07	0.803E-07	0.104E-11	0.16E-05	0.12E-05	0.70E-06	0.53E-06
6	0.219E-13	0.219E-13	0.513E-13	0.33E-10	0.39E-11	0.40E-13	0.76E-12

Example 4 *Consider* $n = 3$, $\lambda = \{1, 2, 3\}$ *and*

$$A_0 = \begin{pmatrix} 0.66 & -0.42 & -0.34 \\ 2.94 & 0.33 & 4.09 \\ 0.1 & 0.48 & 2.96 \end{pmatrix}, \quad B_0 = I,$$

$$A_1 = \begin{pmatrix} 1 & 0.1 & 0.02 \\ 0.1 & 0 & 0.01 \\ 0.02 & 0.03 & 1 \end{pmatrix}, \quad B_1 = 0,$$

$$A_2 = \begin{pmatrix} 0 & 0.01 & 0 \\ 0 & 1 & 0 \\ 0.5 & 0.01 & 0 \end{pmatrix}, \quad B_2 = 0,$$

$$A_3 = \begin{pmatrix} 0 & 0 & 0.01 \\ 0 & 1 & 0.01 \\ 0 & 0.6 & 1 \end{pmatrix}, \quad B_3 = 0.$$

On the assumption that the starting vector is $c^{(0)} = (0.8, 2.2, -1.8)^T$, *Algorithm 2 converges to a solution*

$$c^{(*)} = (1.0757, 4.1263, -2.1138)^T,$$

and in [47] by considering the starting vector $c^{(0)} = (0.8, -0.5, -1.8)^T$, *Algorithm* 2.1 *converges the same solution. The results are given in Table 4 and the numerical results for this example are illustrated on Figure 3. Our observation in this example and other examples indicated that Algorithm 2 is efficient in problems which the Jacobian matrix* (12) *is singular.*

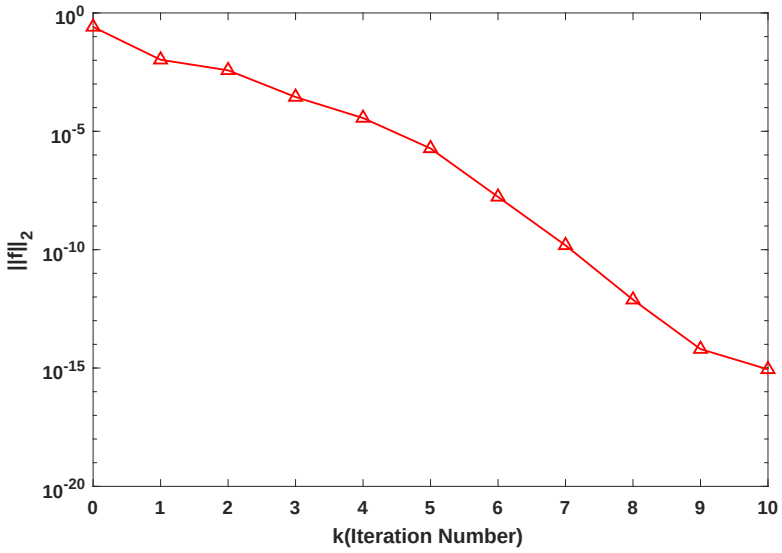

Figure 3. The results obtained for Example 4.

Table 4. Numerical results for Example 4.

Ite	Algorithm 2		Algorithm 2.1 in [47]
k	$\|c^{(k+1)} - c^{(k)}\|$	$\|f(c^{(k)})\|$	$\|f(c^{(k)})\|$
0	0.132E+01	0.256E+00	0.88831112E+00
1	0.293E-01	0.105E-01	0.3411482853E+00
2	0.610E-02	0.377E-02	0.6993713739E+00
3	0.144E-02	0.280E-03	0.2497254940E+00
4	0.106E-03	0.363E-04	0.8748208974E+00
5	0.674E-05	0.189E-05	0.1599815846E-01
6	0.607E-07	0.169E-07	0.6326446527E-03
7	0.503E-09	0.152E-09	0.1314660311E-05
8	0.245E-11	0.767E-12	0.2657927133E-10
9	0.110E-13	0.630E-14	0.7655396217E-15
10	0.294E-14	0.876E-15	

Example 5 *This example is an additive inverse eigenvalue problem with* $n = 5$, $\lambda = \{0, 1, 2, 3, 4\}$ *and*

$$A_0 = \begin{pmatrix} 2 & -0.08 & 0 & 0 & 0 \\ -0.03 & 2 & -0.08 & 0 & 0 \\ 0 & -0.03 & 2 & -0.08 & 0 \\ 0 & 0 & -0.03 & 2 & -0.08 \\ 0 & 0 & 0 & -0.03 & 2 \end{pmatrix}, \tag{21}$$

and $A_i = r_i e_i{}^T$ *for* $i = 1, 2, 3, 4, 5$, *where*

$$R = \sum_{i=1}^{5} r_i e_i{}^T = \begin{pmatrix} 1 & 0 & 0.01 & -0.02 & 0.03 \\ -0.03 & 1 & 0 & 0.01 & -0.02 \\ 0.02 & -0.03 & 1 & 0 & 0.01 \\ -0.01 & 0.02 & -0.03 & 1 & 1 \\ 0 & -0.01 & 0.02 & -0.03 & 1 \end{pmatrix}$$

Supposing that the starting vector $c^{(0)} = (-2, -1, 0, 1, 2)^T$, *Algorithm 1 converges to a solution*

$$c^{(*)} = (-2.0024, -0.9979, 0.0024, 1.0027, 1.9952)^T,$$

and with the starting vector $c^{(0)} = (-2, -1, 0, 1, 2)^T$, *Algorithm* 2.1 *in [48] converges to the same solution. We report the obtained results in Table 5. It could be mentioned that in [48], the smallest singular values are used and the system of nonlinear*

Table 5. Numerical results for Example 5.

Iteration	Algorithm 2		Algorithm 2.1 in [48]
k	$\|c^{(k+1)} - c^{(k)}\|$	$\|f(c^{(k)})\|$	$\|f(c^{(k)})\|$
0	$0.946E-03$	$0.555E-02$	$0.4625594388E-02$
1	$0.242E-05$	$0.826E-05$	$0.3079993599E-06$
2	$0.180E-10$	$0.707E-10$	$0.1767117906E-14$
3	$0.497E-16$	$0.185E-14$	

equations

$$f(c) = \begin{pmatrix} f_1(c) \\ f_2(c) \\ \vdots \\ f_n(c) \end{pmatrix} = 0$$

is solved, where

$$f_i(c) = \sigma_{min}(A(c) - \lambda_i B(c)).$$

In addition, in each iteration of the method presented in [48], we should obtain the smallest singular values. The numerical results are displayed in Figure 4.

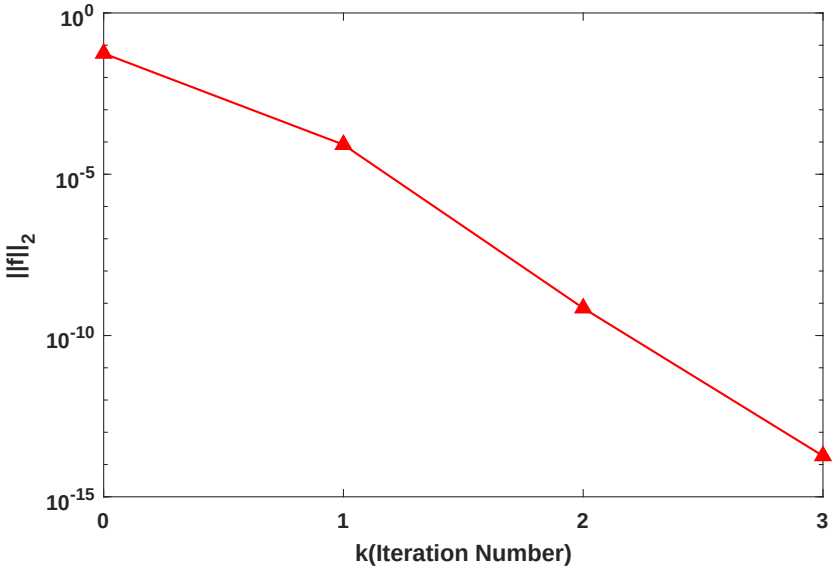

Figure 4. The results obtained for Example 5.

Example 6 *Let us consider an additive inverse eigenvalue problem. here* $n = 8$ *and*

$$A(c) = A_0 + \sum_{i=1}^{8} c_i A_i, \quad B(c) = B_0 + \sum_{i=1}^{8} c_i B_i,$$

where

$$A0 = \begin{pmatrix} 0 & 4 & 1 & 3 & -1 & 1 & 1 & 7 & -1 & 1 \\ 4 & 0 & 1 & 2 & -1 & 2 & 1 & 6 & -1 & 2 \\ 1 & 1 & 0 & 1 & -1 & 3 & 1 & 5 & -1 & 3 \\ 3 & 2 & 1 & 0 & -1 & 4 & 1 & 4 & -1 & 4 \\ -1 & -1 & -1 & -1 & 0 & 5 & 1 & 3 & -1 & 5 \\ 1 & 2 & 3 & 4 & 5 & 0 & 1 & 2 & -1 & 6 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & -1 & 7 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & -1 & 8 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 9 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 \end{pmatrix},$$

Table 6. Numerical results for Example 6.

Iteration	Algorithm 2		
k	$\|c^{(k+1)} - c^{(k)}\|$	$\|c^* - c^{(k)}\|$	$\|f(c^{(k)})\|$
0	$0.200E+02$	$0.5678E+01$	$0.628E+02$
1	$0.223E+00$	$0.242E-01$	$0.726E+01$
2	$0.944E-02$	$0.933E-02$	$0.917E+00$
3	$0.108E-02$	$0.388E-03$	$0.223E-01$
4	$0.822E-05$	$0.144E-04$	$0.904E-02$
5	$0.152E-07$	$0.835E-06$	$0.571E-03$
6	$0.848E-08$	$0.144E-07$	$0.759E-04$
7	$0.848E-09$	$0.908E-09$	$0.113E-05$
8	$0.373E-11$	$0.608E-10$	$0.846E-07$
9	$0.402E-12$	$0.225E-12$	$0.574E-09$
10	$0.128E-13$	$0.128E-13$	$0.211E-11$

and

$$A_i = e_i e_i^T, \ \ i = 1, 2, ..., 8,$$

$$B_0 = I, \quad B_i = 0, \ \ i = 1, 2, ..., 8.$$

In this example, the eigenvalues are given as follow:

$$\lambda^* = \{8.2933, 20.5063, 29.5033, 39.5916, 49.8555, 83.9949, 103.0154, 121.7343, 141.4475, 162.6978\}$$

Supposing that

$$c^{(0)} = (11.90788, 19.70552, 30.54550, 40.06266, 51.58714, 64.70213, 70.17068, 81.2121, 90.9911, 101.0523)^T$$

and by performing Algorithm 2, we have

$$c^{(*)} = (10, 20, 30, 40, 50, 60, 70, 80, 90, 100)^T.$$

Table 6 displays the values of the residual, for Algorithm 2.

Example 7 *As the final example, consider an additive inverse eigenvalue problem. Where the matrices $A(c), B(c)$ are mentioned in Example 7 with the following parameters:*

$$A0 = \begin{pmatrix}
0 & 4 & -1 & 1 & 1 & 5 & -1 & 1 & 1 & 9 & -1 & 1 & 1 & 13 & -1 & 1 & 1 & 17 & -1 & 1 \\
4 & 0 & -1 & 2 & 1 & 4 & -1 & 2 & 1 & 8 & -1 & 2 & 1 & 12 & -1 & 2 & 1 & 16 & -1 & 2 \\
-1 & -1 & 0 & 3 & 1 & 3 & -1 & 3 & 1 & 7 & -1 & 3 & 1 & 11 & -1 & 3 & 1 & 15 & -1 & 3 \\
1 & 2 & 3 & 0 & 1 & 2 & -1 & 4 & 1 & 6 & -1 & 4 & 1 & 10 & -1 & 4 & 1 & 14 & -1 & 4 \\
1 & 1 & 1 & 1 & 0 & 1 & -1 & 5 & 1 & 5 & -1 & 5 & 1 & 9 & -1 & 5 & 1 & 13 & -1 & 5 \\
5 & 4 & 3 & 2 & 1 & 0 & -1 & 6 & 1 & 4 & -1 & 6 & 1 & 8 & -1 & 6 & 1 & 12 & -1 & 6 \\
-1 & -1 & -1 & -1 & -1 & -1 & 0 & 7 & 1 & 3 & -1 & 7 & 1 & 7 & -1 & 7 & 1 & 11 & -1 & 7 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 0 & 1 & 2 & -1 & 8 & 1 & 6 & -1 & 8 & 1 & 10 & -1 & 8 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & -1 & 9 & 1 & 5 & -1 & 9 & 1 & 9 & -1 & 9 \\
9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & -1 & 10 & 1 & 4 & -1 & 10 & 1 & 8 & -1 & 10 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 11 & 1 & 3 & -1 & 11 & 1 & 7 & -1 & 11 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 0 & 1 & 2 & -1 & 12 & 1 & 6 & -1 & 12 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & -1 & 13 & 1 & 5 & -1 & 13 \\
13 & 12 & 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & -1 & 14 & 1 & 4 & -1 & 14 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 15 & 1 & 3 & -1 & 15 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 0 & 1 & 2 & -1 & 16 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & -1 & 17 \\
17 & 16 & 15 & 14 & 13 & 12 & 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & -1 & 18 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 19 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 0
\end{pmatrix},$$

$$A_i = e_i e_i^T, \ \ i = 1, 2, ..., 20,$$

$$B_0 = I, \quad B_i = 0, \ \ i = 1, 2, ..., 20,$$

Table 7. Numerical results for Example 7.

Iteration	Algorithm 2		
k	$\|c^{(k+1)} - c^{(k)}\|$	$\|c^* - c^{(k)}\|$	$\|f(c^{(k)})\|$
0	$0.145E+03$	$0.1161E+03$	$0.897E+03$
1	$0.366E+02$	$0.294E+02$	$0.921E+02$
2	$0.111E+01$	$0.721E+01$	$0.910E+02$
3	$0.530E+00$	$0.270E-01$	$0.604E+01$
4	$0.164E-01$	$0.165E-02$	$0.803E+00$
5	$0.263E-02$	$0.165E-03$	$0.658E-01$
6	$0.127E-02$	$0.113E-04$	$0.196E-02$
7	$0.544E-03$	$0.608E-05$	$0.562E-03$
8	$0.134E-04$	$0.769E-06$	$0.184E-04$
9	$0.341E-05$	$0.813E-07$	$0.491E-05$
10	$0.184E-06$	$0.493E-07$	$0.256E-06$
11	$1.54E-07$	$0.321E-08$	$0.826E-07$
12	$1.90E-9$	$0.127E-9$	$0.750E-08$
13	$5.02E-11$	$0.673E-11$	$0.748E-09$
14	$6.14E-12$	$0.614E-12$	$0.119E-10$

$$\lambda^* = \{5.4340, 15.2403, 29.1990, 47.3533, 82.9628, 95.5328, 105.9533, 112.0969, 116.5968, 127.7562,$$
$$135.2594, 137.3847, 147.3474, 148.9224, 157.8651, 160.3074, 171.7052, 181.2133, 199.7340, 252.1357\}.$$

The starting vector is as follows:

$$c^{(0)} = (11.9, 20.5, 30.1, 41.5, 50.7, 60.1, 71, 81, 90.7, 100.5,$$
$$111.05, 121, 131, 140.15, 151, 160.9, 171.5, 180.5, 190.5, 201)^T,$$

and using Algorithm 2, we get

$$c^{(*)} = (10, 20, 30, 40, 50, 60, 70, 80, 90, 100,$$
$$110, 120, 130, 140, 150, 160, 170, 180, 190, 200)^T.$$

The numerical results and the values of the residual for Algorithm 2 are shown in Table 7.

These examples demonstrate that in Algorithm 1 quadratic convergence indeed occurs in practice, when jacobin is nonsingular. Also Algorithm 2 is very effective when the jacobian matrix is singular.

4. Conclusions

Several problems in structure design, applied mechanics and the design of control systems can be modeled by Problem 2. In the present paper, we have proposed an iterative method for solving parameterized inverse eigenvalue problems. In this method, by using the bidiagonalization, we have transformed the PGIEP into a system of nonlinear equation. Then, we have applied Newton's method to solve the system of created nonlinear equations, and we have applied the Quasi-Newton's in cases where the Jacobi matrix was nonsingular. The use of bidiagonalization eliminates solving a generalized eigenvalue problem in each iteration of the iterative method and reduces the computational complexity. The numerical results indicate the convergence of the method and its effect for solving Problem 2.

Acknowledgments

The authors would like to express their heartfelt thanks to anonymous referees for their useful comments and constructive suggestions that substantially improved the quality and presentation of this article.

References

1. K. Aishima. A quadratically convergent algorithm based on matrix equations for inverse eigenvalue problems. *Linear Algebra Appl.*, 542:310–333, 2018.

2. K. Aishima. A quadratically convergent algorithm for inverse eigenvalue problems with multiple eigenvalues. *Linear Algebra Appl.*, 549:30–52, 2018.
3. S. Al-Homidan. Semidefinite programming for the educational testing problem. *Cent. Eur. J. Oper. Res.*, 16(3):239–249, 2008.
4. J. Alexander. The additive inverse eigenvalue problem and topological degree. *Proc. Am. Math. Soc.*, 70:5–7, 1978.
5. A. Andrew. Solving inverse sturm-liouville problems: theory and practice. *ANZIAM J.*, 58:124–136, 2017.
6. H. Bahai, K. Farahani, and M. Djoudi. Eigenvalue inverse formulation for optimising vibratory behaviour of truss and continuous structures. *Comput. Struct.*, 80:2397–2403, 2002.
7. Z. J. Bai, R. H.Chan, and B. Morini. An inexact cayley transform method for inverse eigenvalue problems. *Inverse Probl.*, 20(5):1675–1689., 2004.
8. J. L. Barlow. More accurate bidiagonal reduction for computing the singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 23(3):761–798, 2002.
9. F. Biegler-König. Sufficient conditions for the solubility of inverse eigenvalue problems. *Linear Algebra Appl.*, 40:89–100, 1981.
10. C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Math. Comput.*, 19(92):577–593, 1965.
11. R. Burden and J. Faires. *Numerical analysis*. Brooks-Cole, 9th edition, 2010.
12. C. Byrnes and X. Wang. The additive inverse eigenvalue problem for linear perturbations. *SIAM J. Matrix Anal. Appl.*, 14:113–117, 1993.
13. J. Cerny. *Nuclear Spectroscopy and Reactions 40-C*. Elsevier. New York, 2012.
14. M. Chu and G. Golub. *Inverse eigenvalue problems: theory, algorithms, and applications*, volume 13. Oxford University Press, 2005.
15. H. Dai. An algorithm for symmetric generalized inverse eigenvalue problems. *Linear Algebra Appl.*, 296:79–98, 1999.
16. H. Dai, Z. Bai, and Y. Wei. On the solvability condition and numerical algorithm for the parameterized generalized inverse eigenvalue problem. *SIAM J. Matrix Anal. Appl.*, 36:707–726, 2015.
17. H. Dai and P. Lancaster. Newton's method for a generalized inverse eigenvalue problem. *Numer. Linear Algebra Appl.*, 4:1–21, 1997.
18. Z. Dalvand and M. Hajarian. Solving generalized inverse eigenvalue problems via l-bfgs-b method. *Inverse Probl. Sci. Eng.*, 28(12):1719–1746, 2020.
19. Z. Dalvand and M. Hajarian. Newton-like and inexact newton-like methods for a parameterized generalized inverse eigenvalue problem. *Math. Methods Appl. Sci.*, 44(6):4217–4234, 2021.
20. Z. Dalvand, M. Hajarian, and J. E. Roman. An extension of the cayley transform method for a parameterized generalized inverse eigenvalue problem. *Numer. Linear Algebra Appl.*, 27(6):e2327, 2020.
21. K. Du, Y. Huang, and Y. Wang. On two generalized inverse eigenvalue problems for hessenberg-upper triangular pencils and their application to the study of gmres convergence. *Linear Algebra Appl.*, 553:16–36, 2018.
22. S. Friedland. Inverse eigenvalue problems. *Linear Algebra Appl.*, 17:15–51, 1977.
23. S. Friedland. Inverse eigenvalue problems. *Linear Algebra Appl.*, 17(1):15–51, 1977.
24. S. Friedland, J. Nocedal, and M. Overto. The formulation and analysis of numerical methods for inverse eigenvalue problems. *SIAM J. Numer. Anal.*, 24:634–667, 1987.
25. G. Gulob and C. V. Loan. *Matrix Computations*. Johns Hopkins University Press, 4th edition, 2010.
26. M. Hajarian. An efficient algorithm based on lanczos type of bcr to solve constrained quadratic inverse eigenvalue problems. *J. Comput. Appl. Math.*, 346:418–431, 2019.
27. M. Hajarian. Solving constrained quadratic inverse eigenvalue problem via conjugate direction method. *Comput. Math. with Appl.*, 76(10):2384–2401, 2019.
28. M. Hajarian and H. Abbas. Least squares solutions of quadratic inverse eigenvalue problem with partially bisymmetric matrices under prescribed submatrix constraints. *Comput. Math. with Appl.*, 76(6):1458–1475, 2018.
29. X. Ji. On matrix inverse eigenvalue problems. *Inverse Probl.*, 14(2):275–285, 1998.
30. V. Kublanovskaya. An approach to solving inverse eigenvalue problems for matrices. *J. Math. Sci.*, 114(6):1808–1819, 2003.
31. L. Li. Some sufficient conditions for the solvability of inverse eigenvalue problems. *Linear Algebra Appl.*, 148:225–236, 1991.
32. L. Luoluo. Sufficient conditions for the solvability of an algebraic inverse eigenvalue problem. *Linear Algebra Appl.*, 221:117–129, 1995.
33. L. Majkut. Eigenvalue based inverse model of beam for structural modification and diagnostics: examples of using. *Lat. Am. J. Solids Struct.*, 7:437–456, 2010.
34. C. Marijuán, M. Pisonero, and L. Ricardo. A map of sufficient conditions for the symmetric nonnegative inverse eigenvalue problem. *Linear Algebra Appl.*, 530:344–365, 2017.
35. S. Mulaik. Fundamentals of common factor analysis. *The Wiley Handbook of Psychometric Testing: A Multidisciplinary Reference on Survey, Scale and Test Development*, 1:209–251, 2018.
36. S. Qi and H. Dai. Numerical methods for solving generalized inverse eigenvalue problem. *Trans. Nanjing Univ. Aeronaut. Astronaut.*, (31(6)):15–19, 1999.
37. C. Remani. *Numerical Methods for Solving Systems of Nonlinear Equations*. Lakehead University, Thunder Bay, Ontario, Canada, 2013.
38. O. Rojo and R. Soto. New conditions for the additive inverse eigenvalue problem for matrices. *Comput. Math. with Appl.*, 23:41–46, 1992.
39. S.F.Yuan, Q. Wang, and Z. Xiong. Linear parameterized inverse eigenvalue problem of bisymmetric matrices. *Linear Algebra Appl.*, 7(439):1990–2007, 2013.
40. L. Shu, W. Bo, and H. Ji-zhong. Homotopy solution of the inverse generalized eigenvalue problems in structural dynamics. *Appl. Math. Mech.*, 25(5):580–586, 2004.
41. J. Sun and Q. Ye. The unsolvability of inverse algebraic eigenvalue problems almost everywhere. *J. Comput. Math.*, 4:212–226, 1986.
42. W. Thomson. *Theory of vibration with applications*. CrC Press, 2018.
43. N. Wagner. Inverse eigenvalue problems in structural dynamics. *In PAMM: Proceedings in Applied Mathematics and Mechanics*, 6(1):339–340, 2006.
44. Y. Wei and H. Dai. An inverse eigenvalue problem for the finite element model of a vibrating rod. *J. Comput. Appl. Math.*, 300:172–182, 2016.

45. W.Ma. A backward error for the symmetric generalized inverse eigenvalue problem. *Linear Algebra Appl.*, 464:90–99, 2015.
46. S. Xu. On the necessary conditions for the solvability of algebraic inverse eigenvalue problems. *J. Comput. Math.*, 10:93–97, 1992.
47. S. Xu. On the sufficient conditions for the solvability of algebraic inverse eigenvalue problems. *J. Comput. Math.*, 10:17–80, 1992.
48. S. Xu. A smallest singular value method for solving inverse eigenvalue problems. *J. Comput. Math.*, 1:23–31, 1996.
49. Y. Zhang. On the general algebraic inverse eigenvalue problems. *J. Comput. Math.*, 22(4):567–580, 2004.

