# Numerical Methods for IEP

Chong Li

Department of Mathematics
Zhejiang University
Hangzhou, 310027, P R China
cli@zju.edu.cn

# Outline

# Outline

# Outline

# Introduction

## Formulation of the IEP

Suppose that

- $\{A_i\}_{i=1}^n$ is a sequence of real symmetric $n \times n$ matrices;
- $\mathbf{c} = (c_1, c_2, \cdots, c_n)^T \in \mathbb{R}^n$;
- $A(\mathbf{c}) = \sum_{i=1}^n c_i A_i$;
- $\lambda_1(\mathbf{c}) \leq \lambda_2(\mathbf{c}) \leq \ldots \leq \lambda_n(\mathbf{c})$ are the eigenvalues of the matrix $A(\mathbf{c})$.

Then the inverse eigenvalue problem (IEP) considered here is as follows.

**Definition**

For $n$ given real numbers $\{\lambda_i^*\}_{i=1}^n$ with $\lambda_1^* \leq \lambda_2^* \leq \cdots \leq \lambda_n^*$, find a vector $\mathbf{c}^* \in \mathbb{R}^n$ such that

$$\lambda_i(\mathbf{c}^*) = \lambda_i^*, \quad 1 \leq i \leq n. \tag{1}$$

$\mathbf{c}^*$ is called a solution of the IEP (1).

# Introduction

## Formulation of the IEP

Suppose that

- $\{A_i\}_{i=1}^n$ is a sequence of real symmetric $n \times n$ matrices;
- $\mathbf{c} = (c_1, c_2, \cdots, c_n)^T \in \mathbb{R}^n$;
- $A(\mathbf{c}) = \sum_{i=1}^n c_i A_i$;
- $\lambda_1(\mathbf{c}) \leq \lambda_2(\mathbf{c}) \leq \ldots \leq \lambda_n(\mathbf{c})$ are the eigenvalues of the matrix $A(\mathbf{c})$.

Then the inverse eigenvalue problem (IEP) considered here is as follows.

### Definition

For $n$ given real numbers $\{\lambda_i^*\}_{i=1}^n$ with $\lambda_1^* \leq \lambda_2^* \leq \cdots \leq \lambda_n^*$, find a vector $\mathbf{c}^* \in \mathbb{R}^n$ such that

$$\lambda_i(\mathbf{c}^*) = \lambda_i^*, \quad 1 \leq i \leq n. \tag{1}$$

$\mathbf{c}^*$ is called a solution of the IEP (1).

# Introduction

Applications of inverse eigenvalue problems:

- the pole assignment problem [C. I. Byrnes, Three Decades of Mathematics Systems Theory, 1989];
- the inverse Sturm-Liouville problem [H. Petzeltova, Commentat. Math. Univ. Carol. 21(1980)];
- applied geophysics [R. L. Parker and K. A. Whaler, J. Geophys. Res. 86(1981)];
- applied physics [N. Li, Linear Algebra Appl. 266(1997)];
- the vibrating string [M. T. Chu and G. H. Golub, Acta Numer., 11(2002)]
- nuclear spectroscopy [P. J. Brussard and P. W. Glaudemans, Elsevier, New York, 1977].

## Equivalent description of the IEP

Define the function $f : \mathbb{R}^n \to \mathbb{R}^n$ by

$$f(\mathbf{c}) = (\lambda_1(\mathbf{c}) - \lambda_1^*, \ldots, \lambda_n(\mathbf{c}) - \lambda_n^*)^T \quad \text{for each } \mathbf{c} \in \mathbb{R}^n. \tag{2}$$

Then the inverse eigenvalue problem is equivalent to the following problem.

**Problem**

*Solve the nonlinear operator equation*

$$\mathbf{f}(\mathbf{c}) = \mathbf{0}, \tag{3}$$

*where* $\mathbf{f}$ *is given by* (2).

## Equivalent description of the IEP

Define the function $f : \mathbb{R}^n \to \mathbb{R}^n$ by

$$f(\mathbf{c}) = (\lambda_1(\mathbf{c}) - \lambda_1^*, \ldots, \lambda_n(\mathbf{c}) - \lambda_n^*)^T \quad \text{for each } \mathbf{c} \in \mathbb{R}^n. \tag{2}$$

Then the inverse eigenvalue problem is equivalent to the following problem.

### Problem

*Solve the nonlinear operator equation*

$$\mathbf{f}(\mathbf{c}) = \mathbf{0}, \tag{3}$$

*where* $\mathbf{f}$ *is given by* (2).

# Outline

# Previous Work on the IEP

Suppose that

- $\boldsymbol{\lambda}^* \equiv (\lambda_1^*, \ldots, \lambda_n^*)^T$;
- $\{\mathbf{q}_i(\mathbf{c})\}_{i=1}^n$ are the normalized eigenvectors corresponding to the eigenvalues $\{\lambda_i(\mathbf{c})\}_{i=1}^n$;
- $J(\mathbf{c})$ is the matrix given by

$$[J(\mathbf{c})]_{ij} = q_i(\mathbf{c})^T A_j q_i(\mathbf{c}), \quad 1 \le i, j \le n; \tag{4}$$

(Then $J(\mathbf{c})$ is the Jacobian matrix of $\mathbf{f}$ at $\mathbf{c}$ and moreover

$$J(\mathbf{c})\mathbf{c} = (\lambda_1(\mathbf{c}), \ldots, \lambda_n(\mathbf{c}))^T,$$

which gives $\mathbf{f}(\mathbf{c}) = J(\mathbf{c})\mathbf{c} - \boldsymbol{\lambda}^*$ [R. H. Chan, H. L. Chung and X. F. Xu, BIT, 43(2003)].)

- $\{\mathbf{c}^k\}$ is the sequence generated by the methods for the IEP.

# Previous Work on the IEP

Suppose that

- $\boldsymbol{\lambda}^* \equiv (\lambda_1^*, \ldots, \lambda_n^*)^T$;
- $\{\mathbf{q}_i(\mathbf{c})\}_{i=1}^n$ are the normalized eigenvectors corresponding to the eigenvalues $\{\lambda_i(\mathbf{c})\}_{i=1}^n$;
- $J(\mathbf{c})$ is the matrix given by

$$[J(\mathbf{c})]_{ij} = q_i(\mathbf{c})^T A_j q_i(\mathbf{c}), \quad 1 \leq i, j \leq n; \tag{4}$$

(Then $J(\mathbf{c})$ is the Jacobian matrix of $\mathbf{f}$ at $\mathbf{c}$ and moreover

$$J(\mathbf{c})\mathbf{c} = (\lambda_1(\mathbf{c}), \ldots, \lambda_n(\mathbf{c}))^T,$$

which gives $\mathbf{f}(\mathbf{c}) = J(\mathbf{c})\mathbf{c} - \boldsymbol{\lambda}^*$ [R. H. Chan, H. L. Chung and X. F. Xu, BIT, 43(2003)].)

- $\{\mathbf{c}^k\}$ is the sequence generated by the methods for the IEP.

# Previous Work on the IEP

Suppose that

- $\boldsymbol{\lambda}^* \equiv (\lambda_1^*, \ldots, \lambda_n^*)^T$;
- $\{\mathbf{q}_i(\mathbf{c})\}_{i=1}^n$ are the normalized eigenvectors corresponding to the eigenvalues $\{\lambda_i(\mathbf{c})\}_{i=1}^n$;
- $J(\mathbf{c})$ is the matrix given by

$$[J(\mathbf{c})]_{ij} = q_i(\mathbf{c})^T A_j q_i(\mathbf{c}), \quad 1 \leq i, j \leq n; \tag{4}$$

(Then $J(\mathbf{c})$ is the Jacobian matrix of $\mathbf{f}$ at $\mathbf{c}$ and moreover

$$J(\mathbf{c})\mathbf{c} = (\lambda_1(\mathbf{c}), \ldots, \lambda_n(\mathbf{c}))^T,$$

which gives $\mathbf{f}(\mathbf{c}) = J(\mathbf{c})\mathbf{c} - \boldsymbol{\lambda}^*$ [R. H. Chan, H. L. Chung and X. F. Xu, BIT, 43(2003)].)

- $\{\mathbf{c}^k\}$ is the sequence generated by the methods for the IEP.

# Previous Work on the IEP

Suppose that

- $\boldsymbol{\lambda}^* \equiv (\lambda_1^*, \ldots, \lambda_n^*)^T$;
- $\{\mathbf{q}_i(\mathbf{c})\}_{i=1}^n$ are the normalized eigenvectors corresponding to the eigenvalues $\{\lambda_i(\mathbf{c})\}_{i=1}^n$;
- $J(\mathbf{c})$ is the matrix given by

$$[J(\mathbf{c})]_{ij} = q_i(\mathbf{c})^T A_j q_i(\mathbf{c}), \quad 1 \leq i, j \leq n; \tag{4}$$

(Then $J(\mathbf{c})$ is the Jacobian matrix of $\mathbf{f}$ at $\mathbf{c}$ and moreover

$$J(\mathbf{c})\mathbf{c} = (\lambda_1(\mathbf{c}), \ldots, \lambda_n(\mathbf{c}))^T,$$

which gives $\mathbf{f}(\mathbf{c}) = J(\mathbf{c})\mathbf{c} - \boldsymbol{\lambda}^*$ [R. H. Chan, H. L. Chung and X. F. Xu, BIT, 43(2003)].)

- $\{\mathbf{c}^k\}$ is the sequence generated by the methods for the IEP.

# Previous Work on the IEP

Previous work on the IEP:

- Algorithm NM: Newton's method for the IEP
  For $k = 0, 1, \ldots$ until convergence do:
  (a) Compute the eigen-decomposition of $A(\mathbf{c}^k)$:

$$\mathbf{q}_i(\mathbf{c}^k)^T A(\mathbf{c}^k)\mathbf{q}_i(\mathbf{c}^k) = \lambda_i(\mathbf{c}^k), \quad 1 \le i \le n.$$

  (b) Form the Jacobian matrix $J(\mathbf{c}^k)$:

$$[J(\mathbf{c}^k)]_{ij} = \mathbf{q}_i(\mathbf{c}^k)^T A_j \mathbf{q}_i(\mathbf{c}^k), \quad 1 \le i, j \le n.$$

  (c) Solve $\mathbf{c}^{k+1}$ from the Jacobian equation:

$$J(\mathbf{c}^k)\mathbf{c}^{k+1} = \boldsymbol{\lambda}^*.$$

The quadratic convergence rate of this method was showed by W. N. Kublanovskaja in [Zap. Naucn. Sem. Leningrad. Otdel. Mat. Inst., in V. A. Steklova Akad. Nauk SSSR, 1970].

# Previous Work on the IEP

Previous work on the IEP:

- Algorithm NM: Newton's method for the IEP
  For $k = 0, 1, \ldots$ until convergence do:
  **(a)** Compute the eigen-decomposition of $A(\mathbf{c}^k)$:

$$\mathbf{q}_i(\mathbf{c}^k)^T A(\mathbf{c}^k)\mathbf{q}_i(\mathbf{c}^k) = \lambda_i(\mathbf{c}^k), \quad 1 \le i \le n.$$

  **(b)** Form the Jacobian matrix $J(\mathbf{c}^k)$:

$$[J(\mathbf{c}^k)]_{ij} = \mathbf{q}_i(\mathbf{c}^k)^T A_j \mathbf{q}_i(\mathbf{c}^k), \quad 1 \le i, j \le n.$$

  **(c)** Solve $\mathbf{c}^{k+1}$ from the Jacobian equation:

$$J(\mathbf{c}^k)\mathbf{c}^{k+1} = \boldsymbol{\lambda}^*.$$

The quadratic convergence rate of this method was showed by W. N. Kublanovskaja in [Zap. Naucn. Sem. Leningrad. Otdel. Mat. Inst., in V. A. Steklova Akad. Nauk SSSR, 1970].

# Previous Work on the IEP

Previous work on the IEP:

- Algorithm NM: Newton's method for the IEP
  For $k = 0, 1, \ldots$ until convergence do:
  **(a)** Compute the eigen-decomposition of $A(\mathbf{c}^k)$:

$$\mathbf{q}_i(\mathbf{c}^k)^T A(\mathbf{c}^k)\mathbf{q}_i(\mathbf{c}^k) = \lambda_i(\mathbf{c}^k), \quad 1 \le i \le n.$$

  **(b)** Form the Jacobian matrix $J(\mathbf{c}^k)$:

$$[J(\mathbf{c}^k)]_{ij} = \mathbf{q}_i(\mathbf{c}^k)^T A_j \mathbf{q}_i(\mathbf{c}^k), \quad 1 \le i, j \le n.$$

  **(c)** Solve $\mathbf{c}^{k+1}$ from the Jacobian equation:

$$J(\mathbf{c}^k)\mathbf{c}^{k+1} = \boldsymbol{\lambda}^*.$$

The quadratic convergence rate of this method was showed by W. N. Kublanovskaja in [Zap. Naucn. Sem. Leningrad. Otdel. Mat. Inst., in V. A. Steklova Akad. Nauk SSSR, 1970].

# Previous Work on the IEP

Disvantage of Algorithm NM: the computation of the eigen-decomposition. (This seems inefficient when the problem size $n$ is large.)

Example with large $n$: large discrete inverse Sturm-Liouville problems ($n$ is the number of grid points) [M. T. Chu and G. H. Golub, Acta Numer., 11(2002)]

To overcome this drawback, different Newton-like methods have been proposed and studied. Consider the following two Newton-like methods [S. Friedland, J. Nocedal and M. L. Overton, SIAM. J. Numer. Anal., 24(1987)].

# Previous Work on the IEP

Disvantage of Algorithm NM: the computation of the eigen-decomposition. (This seems inefficient when the problem size $n$ is large.)

Example with large $n$: large discrete inverse Sturm-Liouville problems ($n$ is the number of grid points) [M. T. Chu and G. H. Golub, Acta Numer., 11(2002)]

To overcome this drawback, different Newton-like methods have been proposed and studied. Consider the following two Newton-like methods [S. Friedland, J. Nocedal and M. L. Overton, SIAM. J. Numer. Anal., 24(1987)].

# Previous Work on the IEP

Disvantage of Algorithm NM: the computation of the eigen-decomposition. (This seems inefficient when the problem size $n$ is large.)

Example with large $n$: large discrete inverse Sturm-Liouville problems ($n$ is the number of grid points) [M. T. Chu and G. H. Golub, Acta Numer., 11(2002)]

To overcome this drawback, different Newton-like methods have been proposed and studied. Consider the following two Newton-like methods [S. Friedland, J. Nocedal and M. L. Overton, SIAM. J. Numer. Anal., 24(1987)].

# Previous Work on the IEP

- Algorithm NLM: Newton-like method

  1 Given $\mathbf{c}^0$, iterate Newton's method once to obtain $\mathbf{c}^1$. Let

  $$\mathbf{p}_i^0 = \mathbf{q}_i(\mathbf{c}^0), \quad 1 \le i \le n.$$

  2 For $k = 1, 2 \ldots$ until convergence do:

  **(a)** Compute $\mathbf{v}_i^k$ by the one-step inverse power method:

  $$(A(\mathbf{c}^k) - \lambda_i^* I)\mathbf{v}_i^k = \mathbf{p}_i^{k-1}, \quad 1 \le i \le n.$$

  **(b)** Normalize $\mathbf{v}_i^k$ to obtain an approximate eigenvector $\mathbf{p}_i^k$ of $A(\mathbf{c}^k)$:

  $$\mathbf{p}_i^k = \frac{\mathbf{v}_i^k}{\| \mathbf{v}_i^k \|}, \quad 1 \le i \le n.$$

# Previous Work on the IEP

- **(c)** Form the approximate Jacobian matrix $J_k$:

$$[J_k]_{ij} = (\mathbf{p}_i^k)^T A_j \mathbf{p}_i^k, \quad 1 \le i, j \le n.$$

**(d)** Solve $\mathbf{c}^{k+1}$ from the approximate Jacobian equation:

$$J_k \mathbf{c}^{k+1} = \boldsymbol{\lambda}^*.$$

This Newton-like method find approximate eigenvectors by the one-step inverse power method. Quadratic convergence property was considered in [S. Friedland, J. Nocedal and M. L. Overton, SIAM. J. Numer. Anal., 24(1987)] but there was a gap in its proof. R. H. Chan *et al*. proved the quadratic convergence rate in [R. H. Chan, X. F. Xu and H. M. Zhou, SIAM J. Numer. Anal., 36(1999)].

# Previous Work on the IEP

- **(c)** Form the approximate Jacobian matrix $J_k$:

$$[J_k]_{ij} = (\mathbf{p}_i^k)^T A_j \mathbf{p}_i^k, \quad 1 \le i, j \le n.$$

**(d)** Solve $\mathbf{c}^{k+1}$ from the approximate Jacobian equation:

$$J_k \mathbf{c}^{k+1} = \boldsymbol{\lambda}^*.$$

This Newton-like method find approximate eigenvectors by the one-step inverse power method. Quadratic convergence property was considered in [S. Friedland, J. Nocedal and M. L. Overton, SIAM. J. Numer. Anal., 24(1987)] but there was a gap in its proof. R. H. Chan *et al*. proved the quadratic convergence rate in [R. H. Chan, X. F. Xu and H. M. Zhou, SIAM J. Numer. Anal., 36(1999)].

# Previous Work on the IEP

- Algorithm CTM: Cayley transform method

  1 Given $\mathbf{c}^0$, compute the orthonormal eigenvectors $\mathbf{q}_i(\mathbf{c}^0)_{i=1}^n$ of $A(\mathbf{c}^0)$. Let $P_0 = [\mathbf{p}_1^0, \ldots, \mathbf{p}_n^0] = [\mathbf{q}_1(\mathbf{c}^0), \ldots, \mathbf{q}_n(\mathbf{c}^0)]$.

  2 For $k = 0, 1, \ldots$ until convergence, do:

  (1) Do the steps (c) and (d) in Algorithm NLM.

  (2) Form the skew-symmetric matrix $Y_k$:

$$[Y_k]_{ij} = \frac{(\mathbf{p}_i^k)^T A(\mathbf{c}^{k+1})\mathbf{p}_j^k}{\lambda_j^* - \lambda_i^*}, \quad i, j = 1, 2, \ldots, n \quad \text{and} \quad i \neq j.$$

  (3) Compute $P_{k+1} = [\mathbf{p}_1^{k+1}, \ldots, \mathbf{p}_n^{k+1}] = [\mathbf{v}_1^{k+1}, \ldots, \mathbf{v}_n^{k+1}]^T$ by solving

$$(I + \tfrac{1}{2} Y_k)\mathbf{v}_j^{k+1} = \mathbf{h}_j^k, \quad i = 1, 2, \ldots, n,$$

  where $\mathbf{h}_j^k$ is the $j$th column of $H_k = (I - \frac{1}{2} Y_k)P_k^T$.

This algorithm forms approximate eigenvectors by applying matrix exponentials and Cayley transforms. It also converges quadratically [S. Friedland, J. Nocedal and M. L. Overton, SIAM. J. Numer. Anal., 24(1987)].

# Previous Work on the IEP

- Algorithm CTM: Cayley transform method

  1 Given $\mathbf{c}^0$, compute the orthonormal eigenvectors $\mathbf{q}_i(\mathbf{c}^0)_{i=1}^n$ of $A(\mathbf{c}^0)$. Let $P_0 = [\mathbf{p}_1^0, \ldots, \mathbf{p}_n^0] = [\mathbf{q}_1(\mathbf{c}^0), \ldots, \mathbf{q}_n(\mathbf{c}^0)]$.

  2 For $k = 0, 1, \ldots$ until convergence, do:

  (1) Do the steps (c) and (d) in Algorithm NLM.

  (2) Form the skew-symmetric matrix $Y_k$:

$$[Y_k]_{ij} = \frac{(\mathbf{p}_i^k)^T A(\mathbf{c}^{k+1})\mathbf{p}_j^k}{\lambda_j^* - \lambda_i^*}, \quad i, j = 1, 2, \ldots, n \quad \text{and} \quad i \neq j.$$

  (3) Compute $P_{k+1} = [\mathbf{p}_1^{k+1}, \ldots, \mathbf{p}_n^{k+1}] = [\mathbf{v}_1^{k+1}, \ldots, \mathbf{v}_n^{k+1}]^T$ by solving

$$(I + \tfrac{1}{2} Y_k)\mathbf{v}_j^{k+1} = \mathbf{h}_j^k, \quad i = 1, 2, \ldots, n,$$

  where $\mathbf{h}_j^k$ is the $j$th column of $H_k = (I - \frac{1}{2} Y_k)P_k^T$.

This algorithm forms approximate eigenvectors by applying matrix exponentials and Cayley transforms. It also converges quadratically [S. Friedland, J. Nocedal and M. L. Overton, SIAM. J. Numer. Anal., 24(1987)].

# Previous Work on the IEP

Disadvantage of Algorithms NLM and CTM: the exact solutions of $n+1$ linear systems.

Although iterative methods (the inner iterations) can reduce the cost, they may over-solve the systems in the sense that the last few inner iterations before convergence may not improve the convergence of the outer iteration [R. H. Chan, H. L. Chung and S. F. Xu, BIT, 43(2003)].

To alleviate the over-solving problem, Chan *et al*. and Bai *et al*. proposed respectively in [R. H. Chan, H. L. Chung and S. F. Xu, BIT, 43(2003)] and [Z. J. Bai, R. H. Chan and B. Morini, Inverse Problems, 20(2004)] the following two inexact methods.

# Previous Work on the IEP

Disadvantage of Algorithms NLM and CTM: the exact solutions of $n+1$ linear systems.

Although iterative methods (the inner iterations) can reduce the cost, they may over-solve the systems in the sense that the last few inner iterations before convergence may not improve the convergence of the outer iteration [R. H. Chan, H. L. Chung and S. F. Xu, BIT, 43(2003)].

To alleviate the over-solving problem, Chan *et al*. and Bai *et al*. proposed respectively in [R. H. Chan, H. L. Chung and S. F. Xu, BIT, 43(2003)] and [Z. J. Bai, R. H. Chan and B. Morini, Inverse Problems, 20(2004)] the following two inexact methods.

# Previous Work on the IEP

Disadvantage of Algorithms NLM and CTM: the exact solutions of $n+1$ linear systems.

Although iterative methods (the inner iterations) can reduce the cost, they may over-solve the systems in the sense that the last few inner iterations before convergence may not improve the convergence of the outer iteration [R. H. Chan, H. L. Chung and S. F. Xu, BIT, 43(2003)].

To alleviate the over-solving problem, Chan *et al*. and Bai *et al*. proposed respectively in [R. H. Chan, H. L. Chung and S. F. Xu, BIT, 43(2003)] and [Z. J. Bai, R. H. Chan and B. Morini, Inverse Problems, 20(2004)] the following two inexact methods.

# Previous Work on the IEP

- Algorithm INLM: Inexact Newton-like method

  1. Do the first step in Algorithm NLM.
  2. For $k = 1, 2, \ldots$ until convergence, do:
     - **(a)** Solve $\mathbf{v}_i^k$ inexactly by the one-step inverse power method

$$(A(\mathbf{c}^k) - \lambda_i^* I)\mathbf{v}_i^k = \mathbf{p}_i^{k-1} + \mathbf{t}_i^k, \quad 1 \le i \le n.$$

       until the residual $\mathbf{t}_i^k$ satisfies $\| \mathbf{t}_i^k \| \le \frac{1}{4}, \quad 1 \le i \le n.$
     - **(b)** Normalize $\mathbf{v}_i^k$ to obtain an approximation eigenvector $\mathbf{p}_i^k$ of $A(\mathbf{c}^k)$:

$$\mathbf{p}_i^k = \frac{\mathbf{v}_i^k}{\| \mathbf{v}_i^k \|}, \quad 1 \le i \le n.$$

# Previous Work on the IEP

- **(c)** Form the approximation Jacobian matrix $J_k$ by

$$[J_k]_{ij} = (\mathbf{p}_i^k)^T A_j \mathbf{p}_i^k, \quad 1 \le i, j \le n.$$

**(d)** Solve $\mathbf{c}^{k+1}$ inexactly from the approximate Jacobian equation:

$$J_k \mathbf{c}^{k+1} = \boldsymbol{\lambda}^* + \mathbf{r}_k$$

until the residual $\mathbf{r}_k$ satisfies

$$\| \mathbf{r}_k \| \le \left( \max \frac{1}{\| \mathbf{v}_i^k \|} \right)^{\beta}, \quad 1 < \beta \le 2.$$

# Previous Work on the IEP

A local convergence of Algorithm INLM was presented in [R. H. Chan, H. L. Chung and S. F. Xu, BIT, 43 (2003).]

## Theorem

*Let the given eigenvalues* $\{\lambda_i^*\}_{i=1}^n$ *be distinct and the Jacobian* $J(\mathbf{c}^*)$ *be nonsingular. Then the inexact Newton-like method is locally convergent with convergence rate* $\beta$*. More precisely, if* $\|\mathbf{c}^0 - \mathbf{c}^*\| \leq \delta$*, then* $\mathbf{c}^k$ *converges to* $\mathbf{c}^*$ *with*

$$\|\mathbf{c}^{k+1} - \mathbf{c}^*\| \leq \alpha \|\mathbf{c}^k - \mathbf{c}^*\|^\beta, k \geq 1.$$

# Previous Work on the IEP

- Algorithm ICTM: The inexact Cayley transform method
  1. Do the first step in Algorithm CTM. Let $P_0 = [\mathbf{p}_1^0, \ldots, \mathbf{p}_n^0]$ and $\boldsymbol{\rho}^0 = (\lambda_1(\mathbf{c}^0), \ldots, \lambda_n(\mathbf{c}^0))^T$.
  2. For $k = 0, 1, \ldots$ until convergence, do:
     (a) Form the approximate Jacobian matrix $J_k$ as follows:

$$[J_k]_{ij} = (\mathbf{p}_i^k)^T A_j \mathbf{p}_i^k, \quad i, j = 1, 2, \ldots, n.$$

     (b) Solve $\mathbf{c}^{k+1}$ inexactly from the approximate Jacobian equation

$$J_k \mathbf{c}^{k+1} = \boldsymbol{\lambda}^* + \mathbf{r}^k, \tag{5}$$

     until the residual $\mathbf{r}^k$ satisfies

$$\|\mathbf{r}^k\| \leq \frac{\|\boldsymbol{\rho}^k - \boldsymbol{\lambda}^*\|^\beta}{\|\boldsymbol{\lambda}^*\|^\beta}, \quad \beta \in (1, 2].$$

# Previous Work on the IEP

- **(c)** Form the skew-symmetric matrix $Y_k$:

$$[Y_k]_{ij} = \frac{(\mathbf{p}_i^k)^T A(\mathbf{c}^{k+1})\mathbf{p}_j^k}{\lambda_j^* - \lambda_i^*}, \quad i, j = 1, 2, \ldots, n \quad \textit{and} \quad i \neq j.$$

**(d)** Compute $P_{k+1} = [\mathbf{p}_1^{k+1}, \ldots, \mathbf{p}_n^{k+1}] = [\mathbf{v}_1^{k+1}, \ldots, \mathbf{v}_n^{k+1}]^T$ by solving

$$(I + \tfrac{1}{2} Y_k)\mathbf{v}_j^{k+1} = \mathbf{h}_j^k, \quad i = 1, 2, \ldots, n, \tag{6}$$

where $\mathbf{h}_j^k$ is the $j$th column of $H_k = (I - \frac{1}{2} Y_k)P_k^T$.

(e) Compute $\boldsymbol{\rho}^{k+1} = (\rho_1^{k+1}, \ldots, \rho_n^{k+1})^T$ by

$$\rho_i^{k+1} = (\mathbf{p}_i^{k+1})^T A(\mathbf{c}^{k+1})\mathbf{p}_i^{k+1}, \quad i = 1, 2, \ldots, n.$$

# Previous Work on the IEP

A local convergence of Algorithm ICTM was presented in [Z. J. Bai, R. H. Chan and B. Morini, Inverse Problems, 20(2004)].

**Theorem**

*Let the given eigenvalues* $\{\lambda_i^*\}_{i=1}^n$ *be distinct and the Jacobian* $J(\mathbf{c}^*)$ *be nonsingular. If* $\|\mathbf{c}^0 - \mathbf{c}^*\| \leq \delta$, *then the inexact Cayley transform method is locally convergent with convergence rate* $\beta$.

# Previous Work on the IEP

In each outer iteration of Algorithms INLM and ICTM, it requires to solve an approximate Jacobian equation by iterative methods. If the residual controls are small, it will need a number of iterations to solve the approximate Jacobian equation. On the other hand, solving this system will become an unstable problem in the case when the matrix $J_k$ has a bad condition.

# Previous Work on the IEP

References:


**(1)** Z. J. Bai, R. H. Chan, B. Morini, An inexact Cayley transform method for inverse eigenvalue problems, Inverse Problems, 20 (2004), 1675-1689.

**(2)** P. J. Brussard, P. W. Glaudemans, Shell Model Applications in Nuclear Spectroscopy, Elsevier, New York, 1977.

**(3)** C. I. Byrnes, Pole placement by output feedback, Three Decades of Mathematics Systems Theory (Lecture Notes in Control and Inform. Sci. vol 135), 1989, 31-78.

**(4)** R. H. Chan, H. L. Chung and S. F. Xu, *The inexact Newton-like method for inverse eigenvalue problem*, BIT, 43 (2003), 7-20.

**(5)** R. H. Chan, S. F. Xu and H. M. Zhou, *On the convergence of a quasi-Newton method for inverse eigenvalue problem*, SIAM J. Numer. Anal., 36 (1999), 436-441.

**(6)** M. T. Chu, G. H. Golub, Structured inverse eigenvalue problems, Acta Numer., 11 (2002), 1-71.

# Previous Work on the IEP


**(7)** S. Friedland, J. Nocedal and M. L. Overton, *The formulation and analysis of numerical methods for inverse eigenvalue problems*, SIAM. J. Numer. Anal., 24 (1987), 634-667.

**(8)** W. N. Kublanovskaja, On an approach to the solution of the inverse eigenvalue problem, Zap. Naucn. Sem. Leningrad. Otdel. Mat. Inst., in V. A. Steklova Akad. Nauk SSSR, (1970), 138-149.

**(9)** N. Li, A matrix inverse eigenvalue problem and its application, Linear Algebra Appl. 266 (1997), 143-152.

**(10)** R. L. Parker, K. A. Whaler, Numerical methods for establishing solutions to the inverse problem of electromagnetic induction, J. Geophys. Res. 86 (1981), 9574-9584.

**(11)** H. Petzeltova, Remark on Newton–Moser type method, Commentat. Math. Univ. Carol. 21 (1980), 719–725.

**(12)** S. F. Xu, *An introduction to inverse algebric eigenvalue problems, Peking University Press*, 1998.

# Moser's Method and Ulm's Method

Suppose that

- $X$ and $Y$ are Banach spaces;
- $\Omega \subset X$ is an open, convex and nonempty subset;
- $g : \Omega \subset X \to Y$ is an operator;
- $g'$ is the Fréchet derivative of $g$;
- $x_0 \in \Omega$ and $B_0 \in \mathcal{L}(Y, X)$.

# Moser's Method and Ulm's Method

Suppose that

- $X$ and $Y$ are Banach spaces;
- $\Omega \subset X$ is an open, convex and nonempty subset;
- $g : \Omega \subset X \to Y$ is an operator;
- $g'$ is the Fréchet derivative of $g$;
- $x_0 \in \Omega$ and $B_0 \in \mathcal{L}(Y, X)$.

# Moser's Method and Ulm's Method

Suppose that

- $X$ and $Y$ are Banach spaces;
- $\Omega \subset X$ is an open, convex and nonempty subset;
- $g : \Omega \subset X \to Y$ is an operator;
- $g'$ is the Fréchet derivative of $g$;
- $x_0 \in \Omega$ and $B_0 \in \mathcal{L}(Y, X)$.

# Moser's Method and Ulm's Method

Newton's method:

$$g'(x_k)(x_{k+1} - x_k) = g(x_k).$$

Moser's method :

$$x_{k+1} = x_k - B_k g(x_k)$$

$$B_{k+1} = 2B_k - B_k g'(x_k) B_k$$

Moser's method avoids solving equations by a sequence of changes of variables. It can be shown that the root-convergence rate of moser's method is $\frac{1+\sqrt{5}}{2}$. [J. Moser, Herman Weil Lectures, Annals of Mathematics Studies, 77, Prinston Univ. Press].

# Moser's Method and Ulm's Method

Newton's method:

$$g'(x_k)(x_{k+1} - x_k) = g(x_k).$$

Moser's method :

$$x_{k+1} = x_k - B_k g(x_k)$$

$$B_{k+1} = 2B_k - B_k g'(x_k) B_k$$

Moser's method avoids solving equations by a sequence of changes of variables. It can be shown that the root-convergence rate of moser's method is $\frac{1+\sqrt{5}}{2}$. [J. Moser, Herman Weil Lectures, Annals of Mathematics Studies, 77, Prinston Univ. Press].

# Moser's Method and Ulm's Method

Newton's method:

$$g'(x_k)(x_{k+1} - x_k) = g(x_k).$$

Moser's method :

$$x_{k+1} = x_k - B_k g(x_k)$$

$$B_{k+1} = 2B_k - B_k g'(x_k) B_k$$

Moser's method avoids solving equations by a sequence of changes of variables. It can be shown that the root-convergence rate of moser's method is $\frac{1+\sqrt{5}}{2}$. [J. Moser, Herman Weil Lectures, Annals of Mathematics Studies, 77, Prinston Univ. Press].

# Moser's Method and Ulm's Method

Ulm's method

$$x_{k+1} = x_k - B_k g(x_k)$$

$$B_{k+1} = 2B_k - B_k g'(x_{k+1}) B_k$$

It can be shown that Ulm's method converges quadratically. [S. Ulm, Izv. Akad. Nauk. Est. SSR].

Our idea:

$$g(x_k) \longrightarrow \mathbf{f}(\mathbf{c}^k) = J(\mathbf{c}^k)\mathbf{c}^k - \boldsymbol{\lambda}^* \longrightarrow J_k \mathbf{c}^k - \boldsymbol{\lambda}^*$$

$$g'(x_{k+1}) \longrightarrow J(\mathbf{c}^{k+1}) \longrightarrow J_{k+1}$$

# Moser's Method and Ulm's Method

Ulm's method

$$x_{k+1} = x_k - B_k g(x_k)$$

$$B_{k+1} = 2B_k - B_k g'(x_{k+1}) B_k$$

It can be shown that Ulm's method converges quadratically. [S. Ulm, Izv. Akad. Nauk. Est. SSR].

Our idea:

$$g(x_k) \longrightarrow \mathbf{f}(\mathbf{c}^k) = J(\mathbf{c}^k)\mathbf{c}^k - \boldsymbol{\lambda}^* \longrightarrow J_k \mathbf{c}^k - \boldsymbol{\lambda}^*$$

$$g'(x_{k+1}) \longrightarrow J(\mathbf{c}^{k+1}) \longrightarrow J_{k+1}$$

# Moser's Method and Ulm's Method

Ulm's method

$$x_{k+1} = x_k - B_k g(x_k)$$

$$B_{k+1} = 2B_k - B_k g'(x_{k+1}) B_k$$

It can be shown that Ulm's method converges quadratically. [S. Ulm, Izv. Akad. Nauk. Est. SSR].

Our idea:

$$g(x_k) \longrightarrow \mathbf{f}(\mathbf{c}^k) = J(\mathbf{c}^k)\mathbf{c}^k - \boldsymbol{\lambda}^* \longrightarrow J_k \mathbf{c}^k - \boldsymbol{\lambda}^*$$

$$g'(x_{k+1}) \longrightarrow J(\mathbf{c}^{k+1}) \longrightarrow J_{k+1}$$

# Moser's Method and Ulm's Method

Ulm's method

$$x_{k+1} = x_k - B_k g(x_k)$$

$$B_{k+1} = 2B_k - B_k g'(x_{k+1}) B_k$$

It can be shown that Ulm's method converges quadratically. [S. Ulm, Izv. Akad. Nauk. Est. SSR].

Our idea:

$$g(x_k) \longrightarrow \mathbf{f}(\mathbf{c}^k) = J(\mathbf{c}^k)\mathbf{c}^k - \boldsymbol{\lambda}^* \longrightarrow J_k \mathbf{c}^k - \boldsymbol{\lambda}^*$$

$$g'(x_{k+1}) \longrightarrow J(\mathbf{c}^{k+1}) \longrightarrow J_{k+1}$$

# Moser's Method and Ulm's Method

References:


**(1)** J. A. Ezquerro, M. A. Hernández, The Ulm method under mild differentiability conditions. Numer. Math. 109 (2008), pp. 193–207.

**(2)** J. M. Gutiérrez, M. A. Hernández and N. Romero, *A note on a modification of Moser's method*, J. Complexity, 24 (2008), pp. 185–197.

**(3)** J. Moser, *Stable and random motions in dynamical systems with special emphasis on celestian mechancis*, Herman Weil Lectures, Annals of Mathematics Studies, 77, Prinston Univ. Press, Prinston, New Jersey, 1973.

**(4)** S. Ulm, *On iterative methods with successive approximation of the inverse operator*, Izv. Akad. Nauk. Est. SSR, 16(1967), pp. 403-411.

1 Introduction

2 Ulm-like Method
- Previous Work
- Our Theoretical Results
- Numerical Experiments

3 Multiple Eigenvalues
- Previous Work
- Our Theoretic Result
- Numerical Experiments

# Outline

# Our Theoretical Results

- Algorithm 1

1 Given $\mathbf{c}^0$, compute the eigen-decomposition of $A(\mathbf{c}^0)$. Let $J_0 = J(\mathbf{c}^0)$ and $\mathbf{p}_i^0 = \mathbf{q}_i(\mathbf{c}^0)$ for $i = 1, 2, \ldots, n$. Let $B_0$ be a matrix satisfying $\|I - B_0 J_0\| \leq \mu$ where $\mu$ is a positive constant. Compute $\mathbf{c}^1$ by

$$\mathbf{c}^1 = B_0 \boldsymbol{\lambda}^*.$$

2 For $k = 1, 2 \ldots$ until convergence, do:

(1) Do the steps (a),(b) and (c) in Algorithm INLM.
(2) Compute the matrix $B_k$ by

$$B_k = 2B_{k-1} - B_{k-1} J_k B_{k-1}.$$

(3) Compute $\mathbf{c}^k$ by

$$\mathbf{c}^{k+1} = \mathbf{c}^k - B_k(J_k \mathbf{c}^k - \boldsymbol{\lambda}^*). \tag{7}$$

# Our Theoretical Results

- Algorithm 2

1 Given $\mathbf{c}^0$, compute the eigen-decomposition of $A(\mathbf{c}^0)$. Let $J_0 = J(\mathbf{c}^0)$ and $P_0 = [\mathbf{p}_1^0, \ldots, \mathbf{p}_n^0] = [\mathbf{q}_1(\mathbf{c}^0), \ldots, \mathbf{q}_n(\mathbf{c}^0)]$. Let $B_0$ be a matrix satisfying $\|I - B_0 J_0\| \leq \mu$ where $\mu$ is a positive constant. Compute $\mathbf{c}^1$ by

$$\mathbf{c}^1 = B_0 \boldsymbol{\lambda}^*.$$

2 For $k = 0, 1, 2, \ldots$ until convergence, do:

(1) Do the steps (c), (d) and (a) in Algorithm ICTM.
(2) Compute the matrix $B_{k+1}$ by

$$B_{k+1} = 2B_k - B_k J_{k+1} B_k.$$

(3) Compute $\mathbf{c}^{k+1}$ by

$$\mathbf{c}^{k+1} = \mathbf{c}^k - B_k(J_k \mathbf{c}^k - \boldsymbol{\lambda}^*). \tag{8}$$

# Our Theoretical Results

Note that, in Algorithms INLM and ICTM, we need to solve the approximation Jacobian equation by iterative methods:

$$J_k \mathbf{c}^{k+1} = \boldsymbol{\lambda}^* + \mathbf{r}_k.$$

Our method avoids solving the equation by

$$B_k = 2B_{k-1} - B_{k-1} J_k B_{k-1}$$

and

$$\mathbf{c}^{k+1} = \mathbf{c}^k - B_k(J_k \mathbf{c}^k - \boldsymbol{\lambda}^*).$$

It is clear to see that Algorithms ICTM, INLM and Algorithms 1, 2 have the same complexity of $O(n^4)$. However, Algorithms 1 and 2 reduce the difficulty by avoiding the solvation of the (approximate) Jacobian equations and hence they seems more stable than Algorithms ICTM and INLM.

# Our Theoretical Results

Note that, in Algorithms INLM and ICTM, we need to solve the approximation Jacobian equation by iterative methods:

$$J_k \mathbf{c}^{k+1} = \boldsymbol{\lambda}^* + \mathbf{r}_k.$$

Our method avoids solving the equation by

$$B_k = 2B_{k-1} - B_{k-1} J_k B_{k-1}$$

and

$$\mathbf{c}^{k+1} = \mathbf{c}^k - B_k(J_k \mathbf{c}^k - \boldsymbol{\lambda}^*).$$

It is clear to see that Algorithms ICTM, INLM and Algorithms 1, 2 have the same complexity of $O(n^4)$. However, Algorithms 1 and 2 reduce the difficulty by avoiding the solvation of the (approximate) Jacobian equations and hence they seems more stable than Algorithms ICTM and INLM.

# Our Theoretical Results

Note that, in Algorithms INLM and ICTM, we need to solve the approximation Jacobian equation by iterative methods:

$$J_k \mathbf{c}^{k+1} = \boldsymbol{\lambda}^* + \mathbf{r}_k.$$

Our method avoids solving the equation by

$$B_k = 2B_{k-1} - B_{k-1} J_k B_{k-1}$$

and

$$\mathbf{c}^{k+1} = \mathbf{c}^k - B_k(J_k \mathbf{c}^k - \boldsymbol{\lambda}^*).$$

It is clear to see that Algorithms ICTM, INLM and Algorithms 1, 2 have the same complexity of $O(n^4)$. However, Algorithms 1 and 2 reduce the difficulty by avoiding the solvation of the (approximate) Jacobian equations and hence they seems more stable than Algorithms ICTM and INLM.

# Our Theoretical Results

Note that, in Algorithms INLM and ICTM, we need to solve the approximation Jacobian equation by iterative methods:

$$J_k \mathbf{c}^{k+1} = \boldsymbol{\lambda}^* + \mathbf{r}_k.$$

Our method avoids solving the equation by

$$B_k = 2B_{k-1} - B_{k-1} J_k B_{k-1}$$

and

$$\mathbf{c}^{k+1} = \mathbf{c}^k - B_k(J_k \mathbf{c}^k - \boldsymbol{\lambda}^*).$$

It is clear to see that Algorithms ICTM, INLM and Algorithms 1, 2 have the same complexity of $O(n^4)$. However, Algorithms 1 and 2 reduce the difficulty by avoiding the solvation of the (approximate) Jacobian equations and hence they seems more stable than Algorithms ICTM and INLM.

# Our Theoretical Results

The following example is taken from [Z.J. Bai, X.Q. Jin, Recent Advances in Scientific Computing and Matrix Analysis, International Press, 2011], where the approximate Jacobian equation is increasingly ill-conditioned.

## Example

Given $B = I + VV^T$ , where

$$V = \begin{pmatrix} 1 & -1 & -3 & -5 & -6 \\ 1 & 1 & -2 & -5 & -17 \\ 1 & -1 & -1 & 5 & 18 \\ 1 & 1 & 1 & 2 & 0 \\ 1 & -1 & 2 & 0 & 1 \\ 1 & 1 & 3 & 0 & -1 \\ 2.5 & .2 & .3 & .5 & .6 \\ 2 & -.2 & .3 & .5 & .8 \end{pmatrix}_{8\times 5}$$

Define $A_k = b_{kk} e_k e_k^T + \sum_{j=1}^{k-1} b_{kj}(e_k e_j^T + e_j e_k^T)$ for each $k = 1, 2, \ldots, 8$.

# Our Theoretical Results

Suppose that

$$\mathbf{c}^* = (1.043890381645, 1.065644751834, 1.091344270553, 1.023155499528,$$
$$0.997448154933, 0.991139967277, 1.094291990723, 0.996548791312)^T.$$

Then,

$$\boldsymbol{\lambda}^* = (-1.292714668049, 0.754908489475, 1.294574985726, 2.361040489862,$$
$$8.801548359777, 17.222889574448, 35.134256281335, 783.036252731297)^T.$$

# Our Theoretical Results

If we choose

$$\mathbf{c^0} = \frac{\text{floor}(30 \times \cdot \mathbf{c}^*)}{30},$$

then the condition number $\kappa_2(J_k)$ is increasingly ill-conditioned and Algorithm INLM fails to converge.

Table: $\|\mathbf{c}^k - \mathbf{c}^*\|$ and $\kappa_2(J_k)$ of Algorithm INLM

| $k$ | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | | | $\kappa_2(J_k)$ |
|---|---|---|---|---|
| | $\beta = 1.6$ | $\beta = 1.8$ | $\beta = 2.0$ | |
| 0 | $5.69e-2$ | $5.69e-2$ | $5.69e-2$ | $6.26e+2$ |
| 1 | $1.60e-2$ | $1.60e-2$ | $1.60e-2$ | $2.02e+3$ |
| 2 | $7.51e-3$ | $7.51e-3$ | $7.51e-3$ | $1.26e+3$ |
| 3 | $5.23e-3$ | $5.23e-3$ | $5.23e-3$ | $4.85e+4$ |
| 4 | $8.51e-2$ | $8.51e-2$ | $8.51e-2$ | $1.42e+3$ |
| 5 | $7.97e-2$ | $7.97e-2$ | $7.97e-2$ | $3.19e+5$ |
| 6 | $2.78e+00$ | $2.78e+00$ | $2.78e+00$ | $7.46e+4$ |
| 7 | $2.78e+00$ | $2.78e+00$ | $2.78e+00$ | $5.57e+6$ |
| 8 | $2.78e+00$ | $2.78e+00$ | $2.78e+00$ | $2.17e+8$ |
| 9 | $2.78e+00$ | $2.78e+00$ | $2.78e+00$ | $1.21e+9$ |
| 10 | $2.78e+00$ | $2.78e+00$ | $2.78e+00$ | |

# Our Theoretical Results

The following theorem shows that, under suitable conditions, Algorithm 1 converges quadratically.

## Theorem

*Suppose that* $\{\lambda_i^*\}_{i=1}^n$ *are distinct and the Jacobian matrix* $J(\mathbf{c}^*)$ *is invertible. Suppose that* $\mu$ *is sufficiently small. There exists* $\delta > 0$ *such that, if* $\mathbf{c}^0 \in \mathbf{B}(\mathbf{c}^*, \delta)$*, then the sequence* $\{\mathbf{c}^k\}$ *generated by Algorithm 1 with initial point* $\mathbf{c}^0$ *converges to* $\mathbf{c}^*$*, and moreover, for* $k = 0, 1, \ldots,$

$$\|\mathbf{c}^k - \mathbf{c}^*\| \leq \tau_1 q^{2^k}$$

$$\|I - B_k J_k\| \leq \tau_2 q^{2^k}$$

*where* $\tau_1$*,* $\tau_2$ *and* $0 < q < 1$ *are constants.*

# Our Theoretical Results

The following theorem shows that, under suitable conditions, Algorithm 2 converges quadratically.

## Theorem

*Suppose that* $\{\lambda_i^*\}_{i=1}^n$ *are distinct and that the Jacobian* $J(\mathbf{c}^*)$ *is invertible. Suppose that* $\mu \leq \delta$ *where* $\delta$ *is a positive constant. If* $\mathbf{c}^0 \in \mathbf{B}(\mathbf{c}^*, \delta)$*, then the sequence* $\{\mathbf{c}^k\}$ *generated by Algorithm 2 with initial point* $\mathbf{c}^0$ *converges to* $\mathbf{c}^*$*, and moreover*

$$\|\mathbf{c}^k - \mathbf{c}^*\| \leq \tau q^{2^k},$$

$$\|I - B_k J_k\| \leq \tau q^{2^k}$$

*hold for each* $k = 0, 1, \ldots$ *where* $\tau$*,* $0 < q < 1$ *are constants.*

1 Introduction

2 Ulm-like Method
- Previous Work
- Our Theoretical Results
- Numerical Experiments

3 Multiple Eigenvalues
- Previous Work
- Our Theoretic Result
- Numerical Experiments

# Outline

# Comparisons of Algorithms 1 and INLM

We illustrate the convergence performance of Algorithm 1 on two examples. For comparison, Algorithm INLM is also tested. In particular, an example for which Algorithm 1 converges but not Algorithm INLM is provided.

## Example

We use Toeplitz matrices as our $A_i$: $A_1 = I$,

$$A_2 = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & \ddots & \vdots \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 & 1 \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}, \cdots, A_n = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & \ddots & \ddots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \ddots & \ddots & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

# Comparisons of Algorithms 1 and INLM

We illustrate the convergence performance of Algorithm 1 on two examples. For comparison, Algorithm INLM is also tested. In particular, an example for which Algorithm 1 converges but not Algorithm INLM is provided.

## Example

We use Toeplitz matrices as our $A_i$: $A_1 = I$,

$$A_2 = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & \ddots & \vdots \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 & 1 \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}, \cdots, A_n = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & \ddots & \ddots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \ddots & \ddots & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

# Comparisons of Algorithms 1 and INLM

We constructed ten 120-by-120 test problems where $\mathbf{c}^*$ is formed with entries randomly chosen between 0 and 10. The exact solutions $\mathbf{c}^*$ are chosen randomly. Then we computed the eigenvalues $\{\lambda_i^*\}_{i=1}^n$ of $A(\mathbf{c}^*)$ as the prescribed eigenvalues. All the concerned linear systems are solved iteratively by the QMR method using the matlab-provided QMR function. The initial guess $\mathbf{c}^0$ is formed by chopping the components of $\mathbf{c}^*$ to three decimal places. For all algorithms, the stopping tolerance for the outer iterations is $10^{-10}$.

# Comparisons of Algorithms 1 and INLM

The following table illustrates the convergence performances of Algorithms 1 and INLM, where "ite." represents the averaged total numbers of outer iterations on ten test problems. Here we take $B_0 = J(\mathbf{c}^0)^{-1}$ for Algorithm 1.

Table: $\|\mathbf{c}^k - \mathbf{c}^*\|$ and ite. of Algorithms 1 and INLM

| $k$ | Algorithm INLM | | | | Algorithm 1 |
|---|---|---|---|---|---|
| | $\beta$=1.5 | $\beta$=1.6 | $\beta$=1.8 | $\beta$=2 | |
| 0 | $6.94e-3$ | $6.94e-3$ | $6.94e-3$ | $6.94e-3$ | $6.94e-3$ |
| 1 | $8.00e-4$ | $8.00e-4$ | $8.00e-4$ | $8.00e-4$ | $8.00e-4$ |
| 2 | $8.58e-7$ | $8.48e-7$ | $8.40e-7$ | $8.40e-7$ | $1.10e-6$ |
| 3 | $5.74e-11$ | $1.07e-11$ | $9.12e-12$ | $9.60e-12$ | $2.21e-11$ |
| 4 | $4.60e-13$ | 0.00 | 0.00 | 0.00 | 0.00 |
| 5 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ite. | 3.1 | 3.0 | 3.0 | 3.0 | 3.0 |

# Comparisons of Algorithms 1 and INLM

The following table illustrates the convergence performances of Algorithms 1 and INLM, where "ite." represents the averaged total numbers of outer iterations on ten test problems. Here we take $B_0 = J(\mathbf{c}^0)^{-1}$ for Algorithm 1.

Table: $\|\mathbf{c}^k - \mathbf{c}^*\|$ and ite. of Algorithms 1 and INLM

| $k$ | Algorithm INLM | | | | Algorithm 1 |
|---|---|---|---|---|---|
| | $\beta$=1.5 | $\beta$=1.6 | $\beta$=1.8 | $\beta$=2 | |
| 0 | $6.94e-3$ | $6.94e-3$ | $6.94e-3$ | $6.94e-3$ | $6.94e-3$ |
| 1 | $8.00e-4$ | $8.00e-4$ | $8.00e-4$ | $8.00e-4$ | $8.00e-4$ |
| 2 | $8.58e-7$ | $8.48e-7$ | $8.40e-7$ | $8.40e-7$ | $1.10e-6$ |
| 3 | $5.74e-11$ | $1.07e-11$ | $9.12e-12$ | $9.60e-12$ | $2.21e-11$ |
| 4 | $4.60e-13$ | 0.00 | 0.00 | 0.00 | 0.00 |
| 5 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ite. | 3.1 | 3.0 | 3.0 | 3.0 | 3.0 |

# Comparisons of Algorithms 1 and INLM

Below we give the convergence performance of Algorithm 1 for different initial guess $B_0$. Recall that $B_0$ satisfies $\|I - B_0 J(\mathbf{c}^0)\| \leq \mu$. The following table gives the averaged total numbers of outer iterations of Algorithm 1 with different $\mu$.

Table: $\|\mathbf{c}^k - \mathbf{c}^*\|$ and ite. of Algorithm 1 with different $\mu$

| $k$ | $\mu = 1e-1$ | $\mu = 1e-2$ | $\mu = 1e-3$ | $\mu = 1e-4$ | $\mu = 0$ |
|---|---|---|---|---|---|
| 0 | $6.94e-3$ | $6.94e-3$ | $6.94e-3$ | $6.94e-3$ | $6.94e-3$ |
| 1 | $8.00e-4$ | $8.00e-4$ | $8.00e-4$ | $8.00e-4$ | $8.00e-4$ |
| 2 | $4.31e-5$ | $1.46e-6$ | $1.08e-6$ | $1.10e-6$ | $1.10e-6$ |
| 3 | $2.14e-8$ | $3.21e-11$ | $2.22e-11$ | $2.53e-11$ | $2.21e-11$ |
| 4 | $9.89e-12$ | 0.00 | 0.00 | 0.00 | 0.00 |
| 5 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ite. | 4.0 | 3.0 | 3.0 | 3.0 | 3.0 |

# Comparisons of Algorithms 1 and INLM

Below we give the convergence performance of Algorithm 1 for different initial guess $B_0$. Recall that $B_0$ satisfies $\|I - B_0 J(\mathbf{c}^0)\| \leq \mu$. The following table gives the averaged total numbers of outer iterations of Algorithm 1 with different $\mu$.

Table: $\|\mathbf{c}^k - \mathbf{c}^*\|$ and ite. of Algorithm 1 with different $\mu$

| $k$ | $\mu = 1e-1$ | $\mu = 1e-2$ | $\mu = 1e-3$ | $\mu = 1e-4$ | $\mu = 0$ |
|---|---|---|---|---|---|
| 0 | $6.94e-3$ | $6.94e-3$ | $6.94e-3$ | $6.94e-3$ | $6.94e-3$ |
| 1 | $8.00e-4$ | $8.00e-4$ | $8.00e-4$ | $8.00e-4$ | $8.00e-4$ |
| 2 | $4.31e-5$ | $1.46e-6$ | $1.08e-6$ | $1.10e-6$ | $1.10e-6$ |
| 3 | $2.14e-8$ | $3.21e-11$ | $2.22e-11$ | $2.53e-11$ | $2.21e-11$ |
| 4 | $9.89e-12$ | 0.00 | 0.00 | 0.00 | 0.00 |
| 5 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ite. | 4.0 | 3.0 | 3.0 | 3.0 | 3.0 |

# Comparisons of Algorithms 1 and INLM

Below we consider an example of non-Toeplitz matrices, which illustrates that even if the condition numbers are not very large, Algorithm INLM fails to converge; while Algorithm 1 still converges.

## Example

This is an inverse problem with $n = 6$. Define $A_0 := \mathbf{0}$, $A_1 := \frac{1}{m_1}\mathbf{e}_1\mathbf{e}_1^T$,

$$A_k := \left(\frac{1}{\sqrt{m_1}}\mathbf{e}_1 - \frac{1}{\sqrt{m_k}}\mathbf{e}_k\right)\left(\frac{1}{\sqrt{m_1}}\mathbf{e}_1 - \frac{1}{\sqrt{m_k}}\mathbf{e}_k\right)^T, \quad k = 2, 3, \ldots, 6,$$

where $m_1 = 2$, $m_2 = m_3 = m_4 = m_5 = m_6 = 0.2$. Now suppose that

$$\boldsymbol{\lambda}^* = (-310.25, -249.22, -28.08, 113.31, 218.74, 487.96)^T.$$

Then,

$$\mathbf{c}^* = (-83.48, -53.83, 89.13, 40.83, -47.79, 21.51)^T.$$

# Comparisons of Algorithms 1 and INLM

Below we consider an example of non-Toeplitz matrices, which illustrates that even if the condition numbers are not very large, Algorithm INLM fails to converge; while Algorithm 1 still converges.

## Example

This is an inverse problem with $n = 6$. Define $A_0 := \mathbf{0}, \quad A_1 := \frac{1}{m_1}\mathbf{e}_1\mathbf{e}_1^T,$

$$A_k := \left(\frac{1}{\sqrt{m_1}}\mathbf{e}_1 - \frac{1}{\sqrt{m_k}}\mathbf{e}_k\right)\left(\frac{1}{\sqrt{m_1}}\mathbf{e}_1 - \frac{1}{\sqrt{m_k}}\mathbf{e}_k\right)^T, \quad k = 2, 3, \ldots, 6,$$

where $m_1 = 2$, $m_2 = m_3 = m_4 = m_5 = m_6 = 0.2$. Now suppose that

$$\boldsymbol{\lambda}^* = (-310.25, -249.22, -28.08, 113.31, 218.74, 487.96)^T.$$

Then,

$$\mathbf{c}^* = (-83.48, -53.83, 89.13, 40.83, -47.79, 21.51)^T.$$

# Comparisons of Algorithms 1 and INLM

We adopt the following four initial points:

- (a) $\mathbf{c}^0 = (-77.96, -62.09, 96.54, 40.11, -44.33, 20.79)^T$;
- (b) $\mathbf{c}^0 = (-76.86, -63.46, 95.29, 41.39, -42.24, 17.38)^T$;
- (c) $\mathbf{c}^0 = (-78.58, -65.98, 97.84, 43.48, -49.27, 23.67)^T$;
- (d) $\mathbf{c}^0 = (-85.48, -67.29, 80.29, 35.39, -45.45, 23.48)^T$.

# Comparisons of Algorithms 1 and INLM

Here we take $B_0 = J(\mathbf{c}^0)^{-1}$ for Algorithm 1. For both algorithms, the stopping tolerance for the outer (Newton) iterations is $10^{-10}$. The following four tables display the error of $\|\mathbf{c}^k - \mathbf{c}^*\|$ and the condition numbers $\kappa_2(J_k)$ of $J_k$ for the above four initial points $\mathbf{c}^0$, where "ite." represents the number of outer iterations, and "*" denotes the corresponding algorithm fails to converge, respectively.

# Comparisons of Algorithms 1 and INLM

Table: $\|\mathbf{c}^k - \mathbf{c}^*\|$, ite. and $\kappa_2(J_k)$ of Algorithms 1 and INLM

| ini. | $k$ | Algorithm INLM | | | | | Algorithm 1 | |
|---|---|---|---|---|---|---|---|---|
| | | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | | | | $\kappa_2(J_k)$ | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | $\kappa_2(J_k)$ |
| | | $\beta = 1.5$ | $\beta = 1.6$ | $\beta = 1.8$ | $\beta = 2.0$ | | | |
| (a) | 0 | $1.29e+1$ | $1.29e+1$ | $1.29e+1$ | $1.29e+1$ | $1.64e+1$ | $1.29e+1$ | $1.64e+1$ |
| | 1 | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.64e+1$ | $1.10e+0$ | $1.64e+1$ |
| | 2 | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.64e+1$ | $1.49e-1$ | $1.64e+1$ |
| | 3 | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.64e+1$ | $6.70e-3$ | $1.64e+1$ |
| | 4 | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.64e+1$ | $2.43e-5$ | $1.64e+1$ |
| | 5 | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.64e+1$ | $4.70e-10$ | $1.64e+1$ |
| | 6 | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.64e+1$ | $1.20e-13$ | |
| | 7 | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.64e+1$ | 0.00 | |
| | 8 | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.64e+1$ | 0.00 | |
| | 9 | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.64e+1$ | 0.00 | |
| | 10 | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.10e+0$ | $1.64e+1$ | 0.00 | |
| | ite. | * | * | * | * | | 6 | |

Table: $\|\mathbf{c}^k - \mathbf{c}^*\|$, ite. and $\kappa_2(J_k)$ of Algorithms 1 and INLM

| ini. | $k$ | Algorithm INLM | | | | | Algorithm 1 | |
|---|---|---|---|---|---|---|---|---|
| | | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | | | | $\kappa_2(J_k)$ | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | $\kappa_2(J_k)$ |
| | | $\beta = 1.5$ | $\beta = 1.6$ | $\beta = 1.8$ | $\beta = 2.0$ | | | |
| (b) | 0 | $1.49e+1$ | $1.49e+1$ | $1.49e+1$ | $1.49e+1$ | $1.63e+1$ | $1.49e+1$ | $1.63e+1$ |
| | 1 | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.64e+1$ | $1.39e+0$ | $1.64e+1$ |
| | 2 | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.64e+1$ | $2.02e-1$ | $1.64e+1$ |
| | 3 | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.64e+1$ | $1.10e-2$ | $1.64e+1$ |
| | 4 | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.64e+1$ | $6.05e-5$ | $1.64e+1$ |
| | 5 | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.64e+1$ | $2.79e-9$ | $1.64e+1$ |
| | 6 | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.64e+1$ | $8.00e-14$ | $1.64e+1$ |
| | 7 | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.64e+1$ | 0.00 | |
| | 8 | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.64e+1$ | 0.00 | |
| | 9 | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.64e+1$ | 0.00 | |
| | 10 | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.39e+0$ | $1.64e+1$ | 0.00 | |
| | ite. | $*$ | $*$ | $*$ | $*$ | | 6 | |

# Comparisons of Algorithms 1 and INLM

Table: $\|\mathbf{c}^k - \mathbf{c}^*\|$, ite. and $\kappa_2(J_k)$ of Algorithms 1 and INLM

| ini. | $k$ | Algorithm INLM | | | | | Algorithm 1 | |
|---|---|---|---|---|---|---|---|---|
| | | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | | | | $\kappa_2(J_k)$ | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | $\kappa_2(J_k)$ |
| | | $\beta = 1.5$ | $\beta = 1.6$ | $\beta = 1.8$ | $\beta = 2.0$ | | | |
| (c) | 0 | $1.62e+1$ | $1.62e+1$ | $1.62e+1$ | $1.62e+1$ | $1.64e+1$ | $1.62e+1$ | $1.64e+1$ |
| | 1 | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $1.64e+1$ | $8.97e-1$ | $1.64e+1$ |
| | 2 | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $1.64e+1$ | $1.11e-1$ | $1.64e+1$ |
| | 3 | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $1.64e+1$ | $3.98e-3$ | $1.64e+1$ |
| | 4 | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $1.64e+1$ | $8.96e-6$ | $1.64e+1$ |
| | 5 | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $1.64e+1$ | $6.63e-11$ | |
| | 6 | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $1.64e+1$ | 0.00 | |
| | 8 | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $1.64e+1$ | 0.00 | |
| | 9 | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $1.64e+1$ | 0.00 | |
| | 10 | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $8.97e-1$ | $1.64e+1$ | 0.00 | |
| | ite. | $*$ | $*$ | $*$ | $*$ | | 5 | |

# Comparisons of Algorithms 1 and INLM

Table: $\|\mathbf{c}^k - \mathbf{c}^*\|$, ite. and $\kappa_2(J_k)$ of Algorithms 1 and INLM

| ini. | $k$ | Algorithm INLM | | | | | Algorithm 1 | |
|---|---|---|---|---|---|---|---|---|
| | | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | | | | $\kappa_2(J_k)$ | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | $\kappa_2(J_k)$ |
| | | $\beta = 1.5$ | $\beta = 1.6$ | $\beta = 1.8$ | $\beta = 2.0$ | | | |
| (d) | 0 | $1.74e+1$ | $1.74e+1$ | $1.74e+1$ | $1.74e+1$ | $1.63e+1$ | $1.74e+1$ | $1.63e+1$ |
| | 1 | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.64e+1$ | $1.33e+0$ | $1.64e+1$ |
| | 2 | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.64e+1$ | $1.90e-1$ | $1.64e+1$ |
| | 3 | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.64e+1$ | $9.93e-3$ | $1.64e+1$ |
| | 4 | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.64e+1$ | $5.02e-5$ | $1.64e+1$ |
| | 5 | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.64e+1$ | $1.94e-9$ | $1.64e+1$ |
| | 6 | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.64e+1$ | $8.00e-14$ | |
| | 7 | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.64e+1$ | 0.00 | |
| | 8 | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.64e+1$ | 0.00 | |
| | 9 | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.64e+1$ | 0.00 | |
| | 10 | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.33e+0$ | $1.64e+1$ | 0.00 | |
| | ite. | * | * | * | * | | 6 | |

# Comparisons of Algorithms 2 and ICTM

We illustrate the convergence performance of Algorithm 2 on two examples. For comparison, Algorithm ICTM is also tested. In the first example, we consider the inverse Toeplitz eigenvalue problem.

## Example

We use Toeplitz matrices as our $\{A_i\}_{i=1}^n$: $A_1 = I$,

$$A_2 = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & \ddots & \vdots \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 & 1 \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}, \ldots, A_n = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & \ddots & \ddots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \ddots & \ddots & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

# Comparisons of Algorithms 2 and ICTM

We consider three problem sizes: $n = 100$, 200, and 300. For each value of $n$, we constructed ten n-by-n test problems. For each test problem, we first generate $\mathbf{c}^*$ with entries randomly chosen between 0 and 10. Then we compute the eigenvalues $\{\lambda_i^*\}_{i=1}^n$ of $A(\mathbf{c}^*)$ as the prescribed eigenvalues. $\mathbf{c}^0$ is formed by chopping the components of $\mathbf{c}^*$ to four decimal places for $n = 100$ and five decimal places for $n = 200$, 300. For both algorithms, the stopping tolerance for the outer iterations is $10^{-10}$.

# Comparisons of Algorithms 2 and ICTM

The following table gives the averaged values of $\|\mathbf{c}^k - \mathbf{c}^*\|$ and the total numbers of outer iterations averaged over the ten test matrices, where "ite." is the same as in the previous table. Here we take $B_0 = J(\mathbf{c}^0)^{-1}$ for Algorithm 2.

# Comparisons of Algorithms 2 and ICTM

Table: Convergence performances of Algorithms 2 and ICTM

| $n$ | $k$ | Algorithm ICTM | | | | Algorithm 2 |
|---|---|---|---|---|---|---|
| | | $\beta$=1.5 | $\beta$=1.6 | $\beta$=1.8 | $\beta$=2 | |
| 100 | 0 | $5.27e-4$ | $5.27e-4$ | $5.27e-4$ | $5.27e-4$ | $5.27e-4$ |
| | 1 | $4.46e-6$ | $4.46e-6$ | $4.44e-6$ | $4.44e-6$ | $4.44e-6$ |
| | 2 | $1.44e-8$ | $1.44e-8$ | $1.44e-8$ | $1.44e-8$ | $3.41e-8$ |
| | 3 | $4.21e-13$ | $4.19e-13$ | $4.28e-13$ | $4.31e-13$ | $1.90e-11$ |
| | 4 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| | ite. | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 200 | 0 | $7.84e-5$ | $7.84e-5$ | $7.84e-5$ | $7.84e-5$ | $7.84e-5$ |
| | 1 | $3.26e-7$ | $3.26e-7$ | $3.26e-7$ | $3.26e-7$ | $3.26e-7$ |
| | 2 | $8.83e-10$ | $8.85e-10$ | $8.84e-10$ | $8.84e-10$ | $2.60e-9$ |
| | 3 | $1.53e-14$ | $1.64e-14$ | $1.45e-14$ | $1.64e-14$ | $3.65e-13$ |
| | 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | ite. | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 300 | 0 | $9.03e-5$ | $9.03e-5$ | $9.03e-5$ | $9.03e-5$ | $9.03e-5$ |
| | 1 | $4.64e-7$ | $4.58e-7$ | $4.58e-7$ | $4.58e-7$ | $4.58e-7$ |
| | 2 | $8.59e-10$ | $8.59e-10$ | $8.59e-6$ | $8.59e-10$ | $2.12e-9$ |
| | 3 | $3.70e-14$ | $3.77e-14$ | $3.83e-14$ | $3.67e-14$ | $2.51e-12$ |
| | 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | ite. | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |

# Comparisons of Algorithms 2 and ICTM

Below we give the convergence performance of Algorithm 2 for different initial guess $B_0$.

Table: Convergence performances of Algorithm 2 with different $\mu$

| $n$ | $k$ | $\mu = 1e-1$ | $\mu = 1e-2$ | $\mu = 1e-3$ | $\mu = 1e-4$ | $\mu = 0$ |
|---|---|---|---|---|---|---|
| 100 | 0 | $5.27e-4$ | $5.27e-4$ | $5.27e-4$ | $5.27e-4$ | $5.27e-4$ |
| | 1 | $5.22e-5$ | $6.32e-6$ | $4.38e-6$ | $4.43e-6$ | $4.44e-6$ |
| | 2 | $5.49e-7$ | $3.14e-8$ | $3.35e-8$ | $3.40e-8$ | $3.41e-8$ |
| | 3 | $1.46e-10$ | $1.33e-11$ | $1.82e-11$ | $1.89e-11$ | $1.90e-11$ |
| | 4 | $4.96e-15$ | 0.00 | 0.00 | 0.00 | 0.00 |
| | 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | ite. | 3.8 | 3.0 | 3.0 | 3.0 | 3.0 |
| 200 | 0 | $7.84e-5$ | $7.84e-5$ | $7.84e-5$ | $7.84e-5$ | $7.84e-5$ |
| | 1 | $7.82e-6$ | $8.30e-7$ | $3.30e-7$ | $3.25e-7$ | $3.26e-7$ |
| | 2 | $7.87e-8$ | $2.28e-9$ | $2.55e-9$ | $2.59e-9$ | $2.60e-9$ |
| | 3 | $1.41e-11$ | $3.05e-13$ | $3.53e-13$ | $3.63e-13$ | $3.65e-13$ |
| | 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | ite. | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 300 | 0 | $9.03e-5$ | $9.03e-5$ | $9.03e-5$ | $9.03e-5$ | $9.03e-5$ |
| | 1 | $8.96e-6$ | $9.30e-7$ | $4.49e-7$ | $4.56e-7$ | $4.58e-7$ |
| | 2 | $8.92e-8$ | $2.28e-9$ | $2.11e-9$ | $2.11e-9$ | $2.12e-9$ |
| | 3 | $2.47e-11$ | $2.44e-12$ | $2.5055e-12$ | $2.51e-12$ | $2.51e-12$ |
| | 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | ite. | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |

# Comparisons of Algorithms 2 and ICTM

The second one is a non-Toeplitz example, where the matrices $\{J_k\}$ are large though both Algorithms 2 and ICTM converge.

## Example

Given $B = I + VV^T$ , where

$$V = \begin{pmatrix} 1 & -1 & -3 & -5 & -6 \\ 1 & 1 & -2 & -5 & -17 \\ 1 & -1 & -1 & 5 & 18 \\ 1 & 1 & 1 & 2 & 0 \\ 1 & -1 & 2 & 0 & 1 \\ 1 & 1 & 3 & 0 & -1 \\ 2.5 & .2 & .3 & .5 & .6 \\ 2 & -.2 & .3 & .5 & .8 \end{pmatrix}_{8\times 5}$$

Define $A_k = b_{kk} e_k e_k^T + \sum_{j=1}^{k-1} b_{kj}(e_k e_j^T + e_j e_k^T)$ for each $k = 1, 2, \ldots, 8$.

# Comparisons of Algorithms 2 and ICTM

Suppose that

$$\mathbf{c}^* = (1.043890381645, 1.065644751834, 1.091344270553, 1.023155499528,$$
$$0.997448154933, 0.991139967277, 1.094291990723, 0.996548791312)^T.$$

Then,

$$\boldsymbol{\lambda}^* = (-1.292714668049, 0.754908489475, 1.294574985726, 2.361040489862,$$
$$8.801548359777, 17.222889574448, 35.134256281335, 783.036252731297)^T.$$

# Comparisons of Algorithms 2 and ICTM

$\mathbf{c^0}$ is chosen by

$$\mathbf{c^0} = \frac{\text{floor}(\phi \times 10^{\psi} \cdot \mathbf{c}^*)}{\phi \times 10^{\psi}}.$$

We consider the following four cases:

1. (a)$\phi = 5$ and $\psi = 1$;
2. (b)$\phi = 3$ and $\psi = 2$;
3. (c)$\phi = 1$ and $\psi = 2$;
4. (d)$\phi = 1$ and $\psi = 3$.

Here we take $B_0 = J(\mathbf{c}^0)^{-1}$ for Algorithm 2. For both algorithms, the stopping tolerance for the outer (Newton) iterations is $10^{-10}$.

# Comparisons of Algorithms 2 and ICTM

The following two tables display the error of $\|\mathbf{c}^k - \mathbf{c}^*\|$ and the condition numbers $\kappa_2(J_k)$ of $J_k$ for the above four initial points $\mathbf{c}^0$, where "ite." represents the number of outer iterations.

Table: $\|\mathbf{c}^k - \mathbf{c}^*\|$, ite. and $\kappa_2(J_k)$ of Algorithms 2 and ICTM

| ini. | $k$ | Algorithm ICTM | | | | | Algorithm 2 | |
|---|---|---|---|---|---|---|---|---|
| | | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | | | | $\kappa_2(J_k)$ | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | $\kappa_2(J_k)$ |
| | | $\beta = 1.5$ | $\beta = 1.6$ | $\beta = 1.8$ | $\beta = 2.0$ | | | |
| (a) | 0 | $3.31e-2$ | $3.31e-2$ | $3.31e-2$ | $3.31e-2$ | $1.42e+3$ | $3.31e-2$ | $1.42e+3$ |
| | 1 | $2.78e-3$ | $2.78e-3$ | $2.78e-3$ | $2.78e-3$ | $1.54e+3$ | $2.78e-3$ | $1.54e+3$ |
| | 2 | $7.06e-5$ | $7.06e-5$ | $7.06e-5$ | $7.06e-5$ | $1.51e+3$ | $4.02e-5$ | $1.51e+3$ |
| | 3 | $1.85e-8$ | $1.85e-8$ | $1.85e-8$ | $1.85e-8$ | $1.51e+3$ | $1.53e-8$ | $1.51e+3$ |
| | 4 | $3.00e-14$ | $3.00e-14$ | $3.00e-14$ | $3.00e-14$ | | $4.00e-14$ | |
| | ite. | 4 | 4 | 4 | 4 | | 4 | |
| (b) | 0 | $5.53e-3$ | $5.53e-3$ | $5.53e-3$ | $5.53e-3$ | $1.61e+3$ | $5.53e-3$ | $1.63e+1$ |
| | 1 | $4.65e-4$ | $4.65e-4$ | $4.65e-4$ | $4.65e-4$ | $1.51e+3$ | $4.65e-4$ | $1.51e+3$ |
| | 2 | $4.90e-7$ | $4.90e-7$ | $4.90e-7$ | $4.90e-7$ | $1.51e+3$ | $2.75e-6$ | $1.51e+3$ |
| | 3 | $1.33e-12$ | $1.33e-12$ | $1.30e-12$ | $1.32e-12$ | | $9.51e-11$ | |
| | ite. | 3 | 3 | 3 | 3 | | 3 | |

# Comparisons of Algorithms 2 and ICTM

The following two tables display the error of $\|\mathbf{c}^k - \mathbf{c}^*\|$ and the condition numbers $\kappa_2(J_k)$ of $J_k$ for the above four initial points $\mathbf{c}^0$, where "ite." represents the number of outer iterations.

Table: $\|\mathbf{c}^k - \mathbf{c}^*\|$, ite. and $\kappa_2(J_k)$ of Algorithms 2 and ICTM

| ini. | $k$ | Algorithm ICTM | | | | | Algorithm 2 | |
|---|---|---|---|---|---|---|---|---|
| | | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | | | | $\kappa_2(J_k)$ | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | $\kappa_2(J_k)$ |
| | | $\beta = 1.5$ | $\beta = 1.6$ | $\beta = 1.8$ | $\beta = 2.0$ | | | |
| (a) | 0 | $3.31e-2$ | $3.31e-2$ | $3.31e-2$ | $3.31e-2$ | $1.42e+3$ | $3.31e-2$ | $1.42e+3$ |
| | 1 | $2.78e-3$ | $2.78e-3$ | $2.78e-3$ | $2.78e-3$ | $1.54e+3$ | $2.78e-3$ | $1.54e+3$ |
| | 2 | $7.06e-5$ | $7.06e-5$ | $7.06e-5$ | $7.06e-5$ | $1.51e+3$ | $4.02e-5$ | $1.51e+3$ |
| | 3 | $1.85e-8$ | $1.85e-8$ | $1.85e-8$ | $1.85e-8$ | $1.51e+3$ | $1.53e-8$ | $1.51e+3$ |
| | 4 | $3.00e-14$ | $3.00e-14$ | $3.00e-14$ | $3.00e-14$ | | $4.00e-14$ | |
| | ite. | 4 | 4 | 4 | 4 | | 4 | |
| (b) | 0 | $5.53e-3$ | $5.53e-3$ | $5.53e-3$ | $5.53e-3$ | $1.61e+3$ | $5.53e-3$ | $1.63e+1$ |
| | 1 | $4.65e-4$ | $4.65e-4$ | $4.65e-4$ | $4.65e-4$ | $1.51e+3$ | $4.65e-4$ | $1.51e+3$ |
| | 2 | $4.90e-7$ | $4.90e-7$ | $4.90e-7$ | $4.90e-7$ | $1.51e+3$ | $2.75e-6$ | $1.51e+3$ |
| | 3 | $1.33e-12$ | $1.33e-12$ | $1.30e-12$ | $1.32e-12$ | | $9.51e-11$ | |
| | ite. | 3 | 3 | 3 | 3 | | 3 | |

# Comparisons of Algorithms 2 and ICTM

Table: $\|\mathbf{c}^k - \mathbf{c}^*\|$, ite. and $\kappa_2(J_k)$ of Algorithms 2 and ICTM

| ini. | $k$ | Algorithm ICTM | | | | | Algorithm 2 | |
|---|---|---|---|---|---|---|---|---|
| | | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | | | | $\kappa_2(J_k)$ | $\|\mathbf{c}^k - \mathbf{c}^*\|$ | $\kappa_2(J_k)$ |
| | | $\beta = 1.5$ | $\beta = 1.6$ | $\beta = 1.8$ | $\beta = 2.0$ | | | |
| (c) | 0 | $1.33e-2$ | $1.33e-2$ | $1.33e-2$ | $1.33e-2$ | $1.78e+3$ | $1.33e-2$ | $1.78e+3$ |
| | 1 | $8.81e-4$ | $8.81e-4$ | $8.81e-4$ | $8.81e-4$ | $1.52e+3$ | $8.97e-1$ | $1.52e+3$ |
| | 2 | $9.01e-6$ | $9.01e-6$ | $9.01e-6$ | $9.01e-6$ | $1.51e+3$ | $1.11e-1$ | $1.51e+3$ |
| | 3 | $2.58e-10$ | $2.58e-10$ | $2.58e-10$ | $2.58e-10$ | $1.51e+3$ | $3.98e-3$ | $1.51e+3$ |
| | 4 | $3.00e-14$ | $3.00e-14$ | $3.00e-14$ | $6.00e-14$ | | $3.00e-14$ | |
| | ite. | 4 | 4 | 4 | 4 | | 4 | |
| (d) | 0 | $1.40e-3$ | $1.40e-3$ | $1.40e-3$ | $1.40e-3$ | $1.51e+3$ | $1.40e-3$ | $1.51e+3$ |
| | 1 | $4.98e-6$ | $4.98e-6$ | $4.98e-6$ | $4.98e-6$ | $1.51e+3$ | $4.98e-6$ | $1.51e+3$ |
| | 2 | $1.71e-10$ | $1.71e-10$ | $1.72e-10$ | $1.72e-10$ | $1.51e+3$ | $3.56e-10$ | $1.51e+3$ |
| | 3 | $5.00e-14$ | $5.00e-14$ | $4.00e-14$ | $4.00e-14$ | | $4.00e-14$ | |
| | ite. | 3 | 3 | 3 | 3 | | 3 | |

1. Introduction

2. Ulm-like Method
   - Previous Work
   - Our Theoretical Results
   - Numerical Experiments

3. Multiple Eigenvalues
   - Previous Work
   - Our Theoretic Result
   - Numerical Experiments

# Outline

# Previous Work

When multiple eigenvalues are present, solving the IEP becomes much more complicate:

(i) the eigenvalue function are not differentiable around the solution;

(ii) the eigenvectors corresponding to the multiple eigenvalue cannot generally be defined as a continuous function of $c$ at the solution.

# Previous Work

When multiple eigenvalues are present, solving the IEP becomes much more complicate:

(i) the eigenvalue function are not differentiable around the solution;

(ii) the eigenvectors corresponding to the multiple eigenvalue cannot generally be defined as a continuous function of $c$ at the solution.

# Previous Work

When multiple eigenvalues are present, solving the IEP becomes much more complicate:

(i) the eigenvalue function are not differentiable around the solution;

(ii) the eigenvectors corresponding to the multiple eigenvalue cannot generally be defined as a continuous function of $c$ at the solution.

# Previous Work

Previous Work on the IEP with multiple eigenvalues:

- Algorithm NM: Newton's method for the IEP with multiple eigenvalues

  For $k = 0, 1, \ldots$ until convergence do:

  **(a)** Compute the eigen-decomposition of $A(\mathbf{c}^k)$:

  $$\mathbf{q}_i(\mathbf{c}^k)^T A(\mathbf{c}^k)\mathbf{q}_i(\mathbf{c}^k) = \lambda_i(\mathbf{c}^k), \quad 1 \le i \le n.$$

  **(b)** Form the Jacobian matrix $J(\mathbf{c}^k)$:

  $$[J(\mathbf{c}^k)]_{ij} = \mathbf{q}_i(\mathbf{c}^k)^T A_j \mathbf{q}_i(\mathbf{c}^k), \quad 1 \le i, j \le n.$$

  **(c)** Solve $\mathbf{c}^{k+1}$ from the Jacobian equation:

  $$J(\mathbf{c}^k)\mathbf{c}^{k+1} = \boldsymbol{\lambda}^*.$$

This algorithm is the same as the corresponding one for the IEP with distinct eigenvalues.

# Previous Work

Previous Work on the IEP with multiple eigenvalues:

- Algorithm NM: Newton's method for the IEP with multiple eigenvalues

  For $k = 0, 1, \ldots$ until convergence do:

  **(a)** Compute the eigen-decomposition of $A(\mathbf{c}^k)$:

  $$\mathbf{q}_i(\mathbf{c}^k)^T A(\mathbf{c}^k)\mathbf{q}_i(\mathbf{c}^k) = \lambda_i(\mathbf{c}^k), \quad 1 \le i \le n.$$

  **(b)** Form the Jacobian matrix $J(\mathbf{c}^k)$:

  $$[J(\mathbf{c}^k)]_{ij} = \mathbf{q}_i(\mathbf{c}^k)^T A_j \mathbf{q}_i(\mathbf{c}^k), \quad 1 \le i, j \le n.$$

  **(c)** Solve $\mathbf{c}^{k+1}$ from the Jacobian equation:

  $$J(\mathbf{c}^k)\mathbf{c}^{k+1} = \boldsymbol{\lambda}^*.$$

This algorithm is the same as the corresponding one for the IEP with distinct eigenvalues.

# Previous Work

Previous Work on the IEP with multiple eigenvalues:

- Algorithm NM: Newton's method for the IEP with multiple eigenvalues

  For $k = 0, 1, \ldots$ until convergence do:

  **(a)** Compute the eigen-decomposition of $A(\mathbf{c}^k)$:

  $$\mathbf{q}_i(\mathbf{c}^k)^T A(\mathbf{c}^k)\mathbf{q}_i(\mathbf{c}^k) = \lambda_i(\mathbf{c}^k), \quad 1 \le i \le n.$$

  **(b)** Form the Jacobian matrix $J(\mathbf{c}^k)$:

  $$[J(\mathbf{c}^k)]_{ij} = \mathbf{q}_i(\mathbf{c}^k)^T A_j \mathbf{q}_i(\mathbf{c}^k), \quad 1 \le i, j \le n.$$

  **(c)** Solve $\mathbf{c}^{k+1}$ from the Jacobian equation:

  $$J(\mathbf{c}^k)\mathbf{c}^{k+1} = \boldsymbol{\lambda}^*.$$

This algorithm is the same as the corresponding one for the IEP with distinct eigenvalues.

# Previous Work

A local convergence of Algorithm NM was presented in [S. Friedland, J. Nocedal and M. L. Overton, SIAM. J. Numer. Anal., 24(1987)].

**Theorem**

*Let $\{\lambda_i^*\}_{i=1}^n$ be given with $\lambda_1^* = \cdots = \lambda_t^* < \lambda_{t+1}^* < \cdots < \lambda_n^*$. There exists $c^*$ such that $\{\lambda_i^*\}_{i=1}^n$ are the eigenvalues of $A(c^*)$. If*

$$\limsup_{k\to\infty} \|J(c^k)^{-1}\| < \infty,$$

*then the iterates $\{c^k\}$ generated by Algorithm NM converge quadratically to $c^*$.*

# Previous Work

A local convergence of Algorithm NM was presented in [S. Friedland, J. Nocedal and M. L. Overton, SIAM. J. Numer. Anal., 24(1987)].

## Theorem

*Let* $\{\lambda_i^*\}_{i=1}^n$ *be given with* $\lambda_1^* = \cdots = \lambda_t^* < \lambda_{t+1}^* < \cdots < \lambda_n^*$. *There exists* $c^*$ *such that* $\{\lambda_i^*\}_{i=1}^n$ *are the eigenvalues of* $A(c^*)$. *If*

$$\limsup_{k\to\infty} \|J(c^k)^{-1}\| < \infty,$$

*then the iterates* $\{c^k\}$ *generated by Algorithm NM converge quadratically to* $c^*$.

# Previous Work

- Algorithm NLM: Newton-like method

  1 Given $\mathbf{c}^0$, iterate Newton's method once to obtain $\mathbf{c}^1$. Let

  $$\mathbf{p}_i^0 = \mathbf{q}_i(\mathbf{c}^0), \quad 1 \le i \le n.$$

  2 For $k = 1, 2 \ldots$ until convergence do:

  **(a)** Compute $\mathbf{v}_i^k$ by the one-step inverse power method:

  $$(A(\mathbf{c}^k) - \lambda_i^* I)\mathbf{v}_i^k = \mathbf{p}_i^{k-1}, \quad 1 \le i \le n.$$

  **(b)** Normalize $\mathbf{v}_i^k$ to obtain an approximate eigenvector $\mathbf{p}_i^k$ of $A(\mathbf{c}^k)$:

  $$\mathbf{p}_i^k = \frac{\mathbf{v}_i^k}{\| \mathbf{v}_i^k \|}, \quad 1 \le i \le n.$$

# Previous Work

- **(c)** Form the approximate Jacobian matrix $J_k$:

$$[J_k]_{ij} = (\mathbf{p}_i^k)^T A_j \mathbf{p}_i^k, \quad 1 \le i, j \le n.$$

**(d)** Solve $\mathbf{c}^{k+1}$ from the approximate Jacobian equation:

$$J_k \mathbf{c}^{k+1} = \boldsymbol{\lambda}^*.$$

This method is also the same as the corresponding one for solving the IEP with distinct eigenvalues.
Paper [S. Friedland, J. Nocedal and M. L. Overton, SIAM. J. Numer. Anal., 24(1987)] presented the convergence analysis of this method in multiple case. However, as Chan *et al*. pointed in [R. H. Chan, X. F. Xu and H. M. Zhou, SIAM J. Numer. Anal., 36(1999)], the proof given by Friedland *et al*. is incorrect.

# Previous Work

- **(c)** Form the approximate Jacobian matrix $J_k$:

$$[J_k]_{ij} = (\mathbf{p}_i^k)^T A_j \mathbf{p}_i^k, \quad 1 \le i, j \le n.$$

**(d)** Solve $\mathbf{c}^{k+1}$ from the approximate Jacobian equation:

$$J_k \mathbf{c}^{k+1} = \boldsymbol{\lambda}^*.$$

This method is also the same as the corresponding one for solving the IEP with distinct eigenvalues.

Paper [S. Friedland, J. Nocedal and M. L. Overton, SIAM. J. Numer. Anal., 24(1987)] presented the convergence analysis of this method in multiple case. However, as Chan *et al*. pointed in [R. H. Chan, X. F. Xu and H. M. Zhou, SIAM J. Numer. Anal., 36(1999)], the proof given by Friedland *et al*. is incorrect.

# Previous Work

- **(c)** Form the approximate Jacobian matrix $J_k$:

$$[J_k]_{ij} = (\mathbf{p}_i^k)^T A_j \mathbf{p}_i^k, \quad 1 \le i, j \le n.$$

**(d)** Solve $\mathbf{c}^{k+1}$ from the approximate Jacobian equation:

$$J_k \mathbf{c}^{k+1} = \boldsymbol{\lambda}^*.$$

This method is also the same as the corresponding one for solving the IEP with distinct eigenvalues.
Paper [S. Friedland, J. Nocedal and M. L. Overton, SIAM. J. Numer. Anal., 24(1987)] presented the convergence analysis of this method in multiple case. However, as Chan *et al*. pointed in [R. H. Chan, X. F. Xu and H. M. Zhou, SIAM J. Numer. Anal., 36(1999)], the proof given by Friedland *et al*. is incorrect.

# Previous Work

- Algorithm CTM: Cayley transform method

  1 Given $\mathbf{c}^0$, compute the orthonormal eigenvectors $\mathbf{q}_i(\mathbf{c}^0)_{i=1}^n$ of $A(\mathbf{c}^0)$. Let $P_0 = [\mathbf{p}_1^0, \ldots, \mathbf{p}_n^0] = [\mathbf{q}_1(\mathbf{c}^0), \ldots, \mathbf{q}_n(\mathbf{c}^0)]$.

  2 For $k = 0, 1, \ldots$ until convergence, do:

  **(1)** Do the steps (c) and (d) in Algorithm NLM.

  **(2)** Form the skew-symmetric matrix $Y_k$:

$$[Y_k]_{ij} = \begin{cases} 0, & \text{for each } i, j \in [1, t]; \\ \dfrac{(\mathbf{p}_i^k)^T A(\mathbf{c}^{k+1}) \mathbf{p}_j^k}{\lambda_j^* - \lambda_i^*}, & \text{for each } i, j \in [t+1, n] \quad \text{and} \quad i \neq j. \end{cases}$$

  **(3)** Compute $P_{k+1} = [\mathbf{p}_1^{k+1}, \ldots, \mathbf{p}_n^{k+1}] = [\mathbf{v}_1^{k+1}, \ldots, \mathbf{v}_n^{k+1}]^T$ by solving

$$(I + \tfrac{1}{2} Y_k)\mathbf{v}_j^{k+1} = \mathbf{h}_j^k, \quad i = 1, 2, \ldots, n,$$

  where $\mathbf{h}_j^k$ is the $j$th column of $H_k = (I - \frac{1}{2} Y_k) P_k^T$.

# Previous Work

A local convergence of Algorithm CTM was presented in [S. Friedland, J. Nocedal and M. L. Overton, SIAM. J. Numer. Anal., 24(1987)].

**Theorem**

*Let $\{\lambda_i^*\}_{i=1}^n$ be given with $\lambda_1^* = \cdots = \lambda_t^* < \lambda_{t+1}^* < \cdots < \lambda_n^*$. There exists $c^*$ such that $\{\lambda_i^*\}_{i=1}^n$ are the eigenvalues of $A(c^*)$. If*

$$\limsup_{k\to\infty} \|J_k^{-1}\| < \infty,$$

*then the iterates $\{c^k\}$ generated by Algorithm CTM converge to $c^*$ quadratically.*

# Previous Work

A local convergence of Algorithm CTM was presented in [S. Friedland, J. Nocedal and M. L. Overton, SIAM. J. Numer. Anal., 24(1987)].

## Theorem

*Let* $\{\lambda_i^*\}_{i=1}^n$ *be given with* $\lambda_1^* = \cdots = \lambda_t^* < \lambda_{t+1}^* < \cdots < \lambda_n^*$. *There exists* $c^*$ *such that* $\{\lambda_i^*\}_{i=1}^n$ *are the eigenvalues of* $A(c^*)$. *If*

$$\limsup_{k\to\infty} \|J_k^{-1}\| < \infty,$$

*then the iterates* $\{c^k\}$ *generated by Algorithm CTM converge to* $c^*$ *quadratically.*

# Previous Work

Note that the nonsingularity assumption needed for convergence was that the inverse of the involved Jacobian and/or the involved approximate Jacobian matrices at all iterations $c^k$ are bounded, and the bound must be independent of the initial point $c^0$.

For ensuring this nonsingularity assumption for Newton's method, Sun *et al*. developed in [D. F. Sun and J. Sun, SIAM J. Numer. Anal., 40 (2003)] a new approach, by using the tool of the strong semismoothness of the eigenvalue function for symmetric matrices, to study the convergence issue for the case with multiple eigenvalues.

# Previous Work

Note that the nonsingularity assumption needed for convergence was that the inverse of the involved Jacobian and/or the involved approximate Jacobian matrices at all iterations $c^k$ are bounded, and the bound must be independent of the initial point $c^0$.

For ensuring this nonsingularity assumption for Newton's method, Sun *et al*. developed in [D. F. Sun and J. Sun, SIAM J. Numer. Anal., 40 (2003)] a new approach, by using the tool of the strong semismoothness of the eigenvalue function for symmetric matrices, to study the convergence issue for the case with multiple eigenvalues.

# Our Theoretic Result

Suppose that

- $\Lambda(c) = diag(\lambda_1(\mathbf{c}), \ldots, \lambda_n(\mathbf{c}))$;
- $\mathcal{Q}(\mathbf{c}) = \{Q(\mathbf{c}) | Q(\mathbf{c})^T Q(\mathbf{c}) = I \text{ and } Q(\mathbf{c})^T A(\mathbf{c}) Q(\mathbf{c}) = \Lambda(\mathbf{c})\}$;
- $\partial_Q \mathbf{f}(c) = \{J(\mathbf{c}) | [J(\mathbf{c})]_{ij} = \mathbf{q}_i(\mathbf{c})^T A_j \mathbf{q}_i(\mathbf{c}) \text{ where } [\mathbf{q}_1(c), \ldots, \mathbf{q}_n(c)] \in \mathcal{Q}(c)\}$;
- $S = \{c \in \mathbb{R}^n | A(c) \text{ has distinct eigenvalues}\}$;
- $\partial_{Q|S} f(c) = \{J(c) \mid J(c) \text{ is the limit of } J(y^i) \in \partial_Q f(y_i) \text{ where } \mathbf{y}^i \in S \text{ and } \mathbf{y}^i \to \mathbf{c}\}$;

# Previous Work

A local convergence of Algorithm NM was presented in [D. F. Sun and J. Sun, SIAM J. Numer. Anal., 40 (2003)] where the nonsingularity assumption is in terms of the Jacobian matrices evaluated at the solution $\mathbf{c}^*$.

**Theorem**

*Let $\{\lambda_i^*\}_{i=1}^n$ be given with $\lambda_1^* = \cdots = \lambda_t^* < \lambda_{t+1}^* < \cdots < \lambda_n^*$. There exists $c^*$ such that $\{\lambda_i^*\}_{i=1}^n$ are the eigenvalues of $A(c^*)$.If (i) for each $k$, $c^k \in S$ and $c^* \in clS$ (ii) all $J(c) \in \partial_{Q|S}\mathbf{f}(c)$ are nonsingular, then there exists a neighborhood $N(c^*)$ of $c^*$ such that for any $c^0 \in N(c^*)$ Algorithm NM is well defined and the iterates $c^k$ converge to $c^*$ quadratically.*

# Previous Work

A local convergence of Algorithm NM was presented in [D. F. Sun and J. Sun, SIAM J. Numer. Anal., 40 (2003)] where the nonsingularity assumption is in terms of the Jacobian matrices evaluated at the solution $\mathbf{c}^*$.

## Theorem

*Let $\{\lambda_i^*\}_{i=1}^n$ be given with $\lambda_1^* = \cdots = \lambda_t^* < \lambda_{t+1}^* < \cdots < \lambda_n^*$. There exists $c^*$ such that $\{\lambda_i^*\}_{i=1}^n$ are the eigenvalues of $A(c^*)$.If (i) for each $k$, $c^k \in S$ and $c^* \in clS$ (ii) all $J(c) \in \partial_{Q|S}\mathbf{f}(c)$ are nonsingular, then there exists a neighborhood $N(c^*)$ of $c^*$ such that for any $c^0 \in N(c^*)$ Algorithm NM is well defined and the iterates $c^k$ converge to $c^*$ quadratically.*

# Previous Work

References:


**(1)** R. H. Chan, S. F. Xu and H. M. Zhou, *On the convergence of a quasi-Newton method for inverse eigenvalue problem*, SIAM J. Numer. Anal., 36 (1999), 436-441.

**(2)** S. Friedland, J. Nocedal and M. L. Overton, *The formulation and analysis of numerical methods for inverse eigenvalue problems*, SIAM. J. Numer. Anal., 24 (1987), 634-667.

**(3)** D. F. Sun and J. Sun, *Strong semismoothness of symmetric matrices and its application to inverse eigenvalue problems*, SIAM J. Numer. Anal., 40 (2003), 2352-2367.

1. Introduction

2. Ulm-like Method
   - Previous Work
   - Our Theoretical Results
   - Numerical Experiments

3. Multiple Eigenvalues
   - Previous Work
   - Our Theoretic Result
   - Numerical Experiments

# Outline

# Our Theoretic Result

- Algorithm ICTM: The inexact Cayley transform method

  1. Do the first step in Algorithm CTM. Let $P_0 = [\mathbf{p}_1^0, \ldots, \mathbf{p}_n^0]$ and $\boldsymbol{\rho}^0 = (\lambda_1(\mathbf{c}^0), \ldots, \lambda_n(\mathbf{c}^0))^T$.
  2. For $k = 0, 1, \ldots$ until convergence, do:
     **(a)** Form the approximate Jacobian matrix $J_k$ as follows:

$$[J_k]_{ij} = (\mathbf{p}_i^k)^T A_j \mathbf{p}_i^k, \quad i, j = 1, 2, \ldots, n.$$

     **(b)** Solve $\mathbf{c}^{k+1}$ inexactly from the approximate Jacobian equation

$$J_k \mathbf{c}^{k+1} = \boldsymbol{\lambda}^* + \mathbf{r}^k, \tag{9}$$

     until the residual $\mathbf{r}^k$ satisfies

$$\|\mathbf{r}^k\| \leq \theta_k \|\boldsymbol{\rho}^k - \boldsymbol{\lambda}^*\|^\beta, \quad \beta \in [1, 2] \quad \text{and} \quad \theta_k \geq 0.$$

# Our Theoretic Result

- **(c)** Form the skew-symmetric matrix $Y_k$:

$$[Y_k]_{ij} = \begin{cases} 0, & \text{for each } i,j \in [1,t]; \\ \dfrac{(\mathbf{p}_i^k)^T A(\mathbf{c}^{k+1})\mathbf{p}_j^k}{\lambda_j^* - \lambda_i^*}, & \text{for each } i,j \in [t+1,n] \quad \text{and} \quad i \neq j. \end{cases}$$

**(d)** Compute $P_{k+1} = [\mathbf{p}_1^{k+1}, \ldots, \mathbf{p}_n^{k+1}] = [\mathbf{v}_1^{k+1}, \ldots, \mathbf{v}_n^{k+1}]^T$ by solving

$$(I + \tfrac{1}{2} Y_k)\mathbf{v}_j^{k+1} = \mathbf{h}_j^k, \quad i = 1, 2, \ldots, n, \tag{10}$$

where $\mathbf{h}_j^k$ is the $j$th column of $H_k = (I - \frac{1}{2} Y_k)P_k^T$.

(e) Compute $\boldsymbol{\rho}^{k+1} = (\rho_1^{k+1}, \ldots, \rho_n^{k+1})^T$ by

$$\rho_i^{k+1} = (\mathbf{p}_i^{k+1})^T A(\mathbf{c}^{k+1})\mathbf{p}_i^{k+1}, \quad i = 1, 2, \ldots, n.$$

# Our Theoretic Result

**Theorem**

*Suppose that there exists* $\mathbf{c}^* \in clS$ *such that the matrix* $A(\mathbf{c}^*)$ *has eigenvalues given by* $\lambda_1^* = \lambda_2^* = \cdots = \lambda_t^* < \lambda_{t+1}^* < \cdots < \lambda_n^*$. *Suppose that all* $J(\mathbf{c}^*) \in \partial_{Q|S}\mathbf{f}(\mathbf{c}^*)$ *are nonsingular. Then there exists a number* $\delta > 0$ *such that, for each* $\mathbf{c}^0 \in \mathbf{B}(\mathbf{c}^*, \delta) \bigcap S$, *the sequence* $\{\mathbf{c}^k\}$ *generated by Algorithm ICTM converges to* $\mathbf{c}^*$. *More precisely,*

(i) *if* $\beta \in (1,\ 2]$, $\{\mathbf{c}^k\}$ *converges to* $\mathbf{c}^*$ *with convergence rate* $\beta$;

(ii) *if* $\beta = 1$ *and* $\lim_{k\to\infty} \theta_k = 0$, $\{\mathbf{c}^k\}$ *converges to* $\mathbf{c}^*$ *with superlinear convergence property.*

1. Introduction

2. Ulm-like Method
   - Previous Work
   - Our Theoretical Results
   - Numerical Experiments

3. Multiple Eigenvalues
   - Previous Work
   - Our Theoretic Result
   - Numerical Experiments

# Outline

# Numerical Examples

We illustrate the convergence performance of Algorithm ICTM on two examples. For comparison, Algorithm CTM is also tested. Our aim is, for the inverse eigenvalue problems with multiple eigenvalues, to illustrate the advantage of Algorithm ICTM over Algorithm CTM in terms of minimizing the oversolving problem and the overall computational complexity.

# Numerical Experiments

## Example

Given $B = (b_{ij})_{8\times 8} = I + VV^T$ , where

$$V = \begin{pmatrix} 1 & -1 & -3 & -5 & -6 \\ 1 & 1 & -2 & -5 & -17 \\ 1 & -1 & -1 & 5 & 18 \\ 1 & 1 & 1 & 2 & 0 \\ 1 & -1 & 2 & 0 & 1 \\ 1 & 1 & 3 & 0 & -1 \\ 2.5 & .2 & .3 & .5 & .6 \\ 2 & -.2 & .3 & .5 & .8 \end{pmatrix}_{8\times 5}.$$

Define $A_k = b_{kk}\mathbf{e}_k\mathbf{e}_k^T + \sum_{j=1}^{k-1} b_{kj}(\mathbf{e}_k\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_k^T)$ for each $k = 1, 2, \ldots, 8$, where $\mathbf{e}_k$ is the kth column of the identity matrix $I$.

Suppose that $\mathbf{c}^* = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1)^T$. Then

$$\boldsymbol{\lambda}^* = (1,\ 1,\ 1,\ 2.12075,\ 9.21887,\ 17.2814,\ 35.7082,\ 722.681)^T.$$

We constructed ten test problems where the initial guess $\mathbf{c^0}$ is generated via perturbing each entry of $\mathbf{c}^*$ uniformly distributed in an interval of $[-0.01,\ 0.01]$. All linear systems appearing in Algorithms ICTM and CTM were solved by the QMR method. We choose $\theta_k \equiv 2/3$ for $\beta \in (1,\ 2]$ and $\theta_k = \frac{2}{3(k+1)}$ for $\beta = 1$. The stopping tolerance for the outer (Newton) iterations is $10^{-13}$.

# Numerical Experiments

The following table illustrates the convergence performances of Algorithm ICTM for one of the tests with MILU preconditioner used.

Table: Convergence performance of Algorithm ICTM

| $k$ | $\beta = 1.0$ | $\beta = 1.1$ | $\beta = 1.3$ | $\beta = 1.5$ | $\beta = 1.7$ | $\beta = 1.9$ | $\beta = 2$ |
|---|---|---|---|---|---|---|---|
| 0 | $1.97e-2$ | $1.97e-2$ | $1.97e-2$ | $1.97e-2$ | $1.97e-2$ | $1.97e-2$ | $1.97e-2$ |
| 1 | $2.84e-2$ | $2.84e-2$ | $2.84e-2$ | $2.84e-2$ | $2.84e-2$ | $2.84e-2$ | $2.84e-2$ |
| 2 | $1.03e-3$ | $1.03e-3$ | $1.03e-3$ | $1.03e-3$ | $1.03e-3$ | $9.33e-4$ | $9.33e-4$ |
| 3 | $1.91e-5$ | $1.91e-5$ | $1.91e-5$ | $1.91e-5$ | $3.65e-6$ | $3.00e-6$ | $3.00e-6$ |
| 4 | $1.38e-7$ | $1.38e-7$ | $1.38e-7$ | $8.14e-9$ | $3.01e-11$ | $2.08e-11$ | $2.08e-11$ |
| 5 | $7.93e-10$ | $7.93e-10$ | $2.28e-11$ | $5.00e-14$ | $5.00e-14$ | $5.00e-14$ | $5.00e-14$ |
| 6 | $8.00e-14$ | $8.00e-14$ | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 7 | 0.00 | 0.00 | | | | | |

# Numerical Experiments

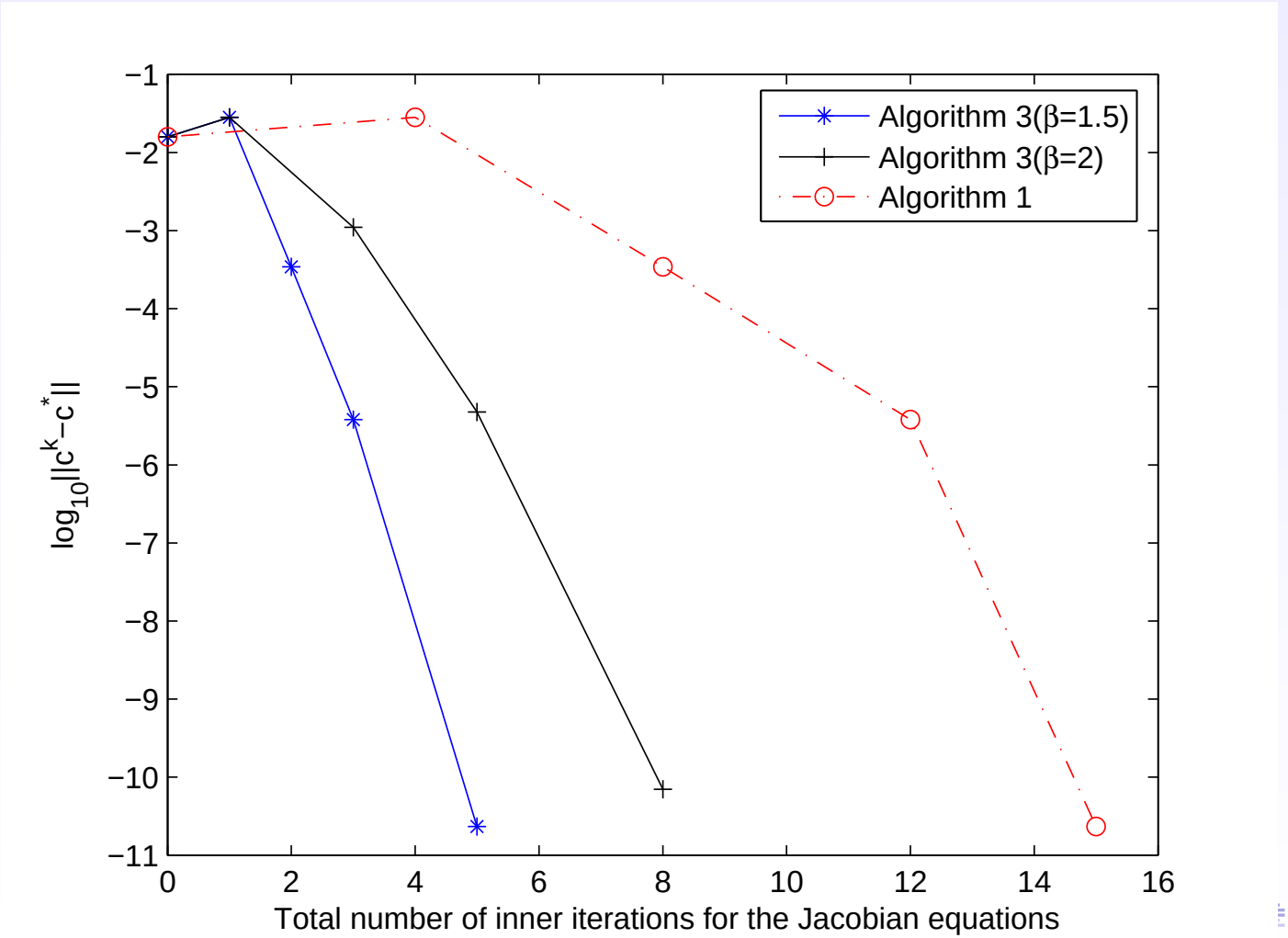
While the convergence performances averaged over the ten test problems of Algorithms ICTM and CTM are illustrated in the following table, where "$N_o$", "$N_i$" represent respectively the averaged total numbers of outer iterations and inner iterations required for solving the approximate Jacobian equations averaged over the ten test problems. In the table, "I" and "P" respectively mean no preconditioner or the MILU preconditioner is used.

Table: Averaged total numbers of outer and inner iterations

| | | Algorithm 1 | Algorithm 2 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | $\beta = 1.0$ | $\beta = 1.1$ | $\beta = 1.3$ | $\beta = 1.5$ | $\beta = 1.7$ | $\beta = 1.9$ | $\beta = 2$ |
| I | $N_o$ | 4.6 | 4.6 | 5.4 | 5.1 | 4.6 | 4.6 | 4.6 | 4.6 |
| | $N_i$ | 46.1 | 37.2 | 36.4 | 34.0 | 30.8 | 31.0 | 31.1 | 31.2 |
| P | $N_o$ | 4.5 | 5.8 | 5.8 | 4.7 | 4.5 | 4.5 | 4.5 | 4.5 |
| | $N_i$ | 17.1 | 7.1 | 6.7 | 6.0 | 7.2 | 8.0 | 8.8 | 9.0 |

# Numerical Experiments

To further illustrate the oversolving problem, we give the convergence history of Algorithms ICTM and CTM in the following figure for one of the tests.



# Numerical Experiments

## Example

We use Toeplitz matrices as our $\{A_i\}_{i=1}^n$:

$$A_1 = I, \quad A_2 = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & \ddots & \vdots \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 & 1 \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}, \quad \dots, \quad A_n = \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \dots \\ \vdots & \ddots & \ddots & \ddots \\ 0 & \dots & \ddots & \ddots \\ 1 & 0 & \dots & 0 \end{pmatrix}$$

Thus $A(\mathbf{c})$ is a symmetric Toeplitz matrix with the first column equal to $\mathbf{c}$.

# Numerical Experiments

We consider three problem sizes: $n = 50$, 100 and 150. For each value of n, we first generate a vector $\mathbf{c}$ such that there exist some integers $1 \leq k \leq n-1$ such that $|\lambda_{k+1}(\mathbf{c}) - \lambda_k(\mathbf{c})| < 5e-6$ where $\{\lambda_i(\mathbf{c})\}_{i=1}^n$ are the eigenvalues of matrix $A(\mathbf{c})$. Set

$$\lambda_i^* = \begin{cases} \lambda_k(\mathbf{c}), & i = k,\ k+1; \\ \lambda_i(\mathbf{c}), & \text{otherwise.} \end{cases}$$

Then we choose $\{\lambda_i^*\}_{i=1}^n$ as the prescribed eigenvalues. Since both Algorithm ICTM and Algorithm CTM are locally convergent, $\mathbf{c}^0$ is formed by chopping the components of $\mathbf{c}$ to three decimal places for $n = 50$, to four decimal places for $n = 100$ and to five decimal places for $n = 150$.

# Numerical Experiments

We present the following table where "$N_o$", "$N_i$" represent respectively the averaged total numbers of outer iterations and inner iterations required for solving the approximate Jacobian equations averaged over the ten test problems.

Table: Averaged total numbers of outer and inner iterations

| | | Algorithm 1 | Algorithm 2 | |
|---|---|---|---|---|
| | | | $\beta = 1.5$ | $\beta = 2$ |
| 50 | $N_o$ | 5.4 | 5.4 | 5.4 |
| | $N_i$ | 38.7 | 14.6 | 23.4 |
| 100 | $N_o$ | 5.3 | 5.3 | 5.3 |
| | $N_i$ | 56.3 | 26.0 | 39.1 |
| 150 | $N_o$ | 3.7 | 3.7 | 3.7 |
| | $N_i$ | 47.5 | 23.9 | 34.9 |

*Thank You*