

# **From Discrete Nahm Equations to a New Algorithm for Principal Component Analysis in Data Science**

**Kinji Kimura**

University of Fukui

# Outline and Field-Specific Notation

## Outline

- ▶ Discrete Integrable System
- ▶ Our Aim
- ▶ Principal Component Analysis
- ▶ The QQDS Method, Interpreted as a Discrete Integrable System
- ▶ Numerical Experiments

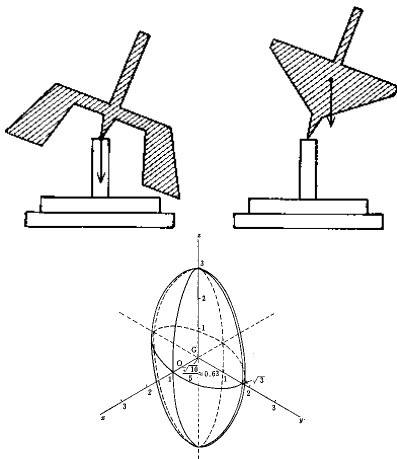
## Field-Specific Notation

- ▶ For example, the three-term recurrence relation

$$H(n)u_{n+1} + G(n)u_n^2 + F(n)u_{n-1} = 0, (n = 1, 2, \dots)$$

is written as follows.

$$H^n u^{n+1} + G^n (u^n)^2 + F^n u^{n-1} = 0, (n = 1, 2, \dots)$$



## Differential Equations Describing the Motion of Spinning Tops (Euler-Poisson Equation)

$$A \frac{d\omega_1}{dt} = (B - C) \omega_2 \omega_3 + z_0 \gamma_2 - y_0 \gamma_3$$

$$B \frac{d\omega_2}{dt} = (C - A) \omega_3 \omega_1 + x_0 \gamma_3 - z_0 \gamma_1$$

$$C \frac{d\omega_3}{dt} = (A - B) \omega_1 \omega_2 + y_0 \gamma_1 - x_0 \gamma_2$$

$$\frac{d\gamma_1}{dt} = \omega_3 \gamma_2 - \omega_2 \gamma_3$$

$$\frac{d\gamma_2}{dt} = \omega_1 \gamma_3 - \omega_3 \gamma_1$$

$$\frac{d\gamma_3}{dt} = \omega_2 \gamma_1 - \omega_1 \gamma_2$$

Sufficient conditions for integrable cases:  
 (For Euler)  $x_0 = y_0 = z_0 = 0$ , (For Lagrange)  $A = B, x_0 = y_0 = 0$ , (For Kovalevskaya)  $A = B = 2C, z_0 = 0$

In these cases, the top can rotate indefinitely.

## The Kahan-Hirota-Kimura Difference Scheme(1)

The following scheme preserves the property of indefinite rotation.

### Discrete Euler Top (Hirota and Kimura, 2000)

$$A(\omega_1^{n+1} - \omega_1^n) / \delta = (B - C)(\omega_2^{n+1} \omega_3^n + \omega_2^n \omega_3^{n+1}) / 2$$

$$B(\omega_2^{n+1} - \omega_2^n) / \delta = (C - A)(\omega_3^{n+1} \omega_1^n + \omega_3^n \omega_1^{n+1}) / 2$$

$$C(\omega_3^{n+1} - \omega_3^n) / \delta = (A - B)(\omega_1^{n+1} \omega_2^n + \omega_1^n \omega_2^{n+1}) / 2$$

### Discrete Lagrange Top (Kimura and Hirota, 2000)

$$A(\omega_1^{n+1} - \omega_1^n) / \delta = (A - C)(\omega_2^{n+1} \omega_3^n + \omega_2^n \omega_3^{n+1}) / 2 + z_0(\gamma_2^{n+1} + \gamma_2^n) / 2$$

$$A(\omega_2^{n+1} - \omega_2^n) / \delta = (C - A)(\omega_3^{n+1} \omega_1^n + \omega_3^n \omega_1^{n+1}) / 2 - z_0(\gamma_1^{n+1} + \gamma_1^n) / 2$$

$$(\omega_3^{n+1} - \omega_3^n) / \delta = 0$$

$$(\gamma_1^{n+1} - \gamma_1^n) / \delta = (\omega_3^{n+1} \gamma_2^n + \omega_3^n \gamma_2^{n+1}) / 2 - (\omega_2^{n+1} \gamma_3^n + \omega_2^n \gamma_3^{n+1}) / 2$$

$$(\gamma_2^{n+1} - \gamma_2^n) / \delta = (\omega_1^{n+1} \gamma_3^n + \omega_1^n \gamma_3^{n+1}) / 2 - (\omega_3^{n+1} \gamma_1^n + \omega_3^n \gamma_1^{n+1}) / 2$$

$$(\gamma_3^{n+1} - \gamma_3^n) / \delta = (\omega_2^{n+1} \gamma_1^n + \omega_2^n \gamma_1^{n+1}) / 2 - (\omega_1^{n+1} \gamma_2^n + \omega_1^n \gamma_2^{n+1}) / 2$$

## The Kahan-Hirota-Kimura Difference Scheme(2)

The following scheme preserves the property of indefinite rotation.

Discrete Euler Top (Hirota and Kimura, 2000)

$$A(\omega_1^{n+1} - \omega_1^n) / \delta = (B - C)(\omega_2^{n+1} \omega_3^n + \omega_2^n \omega_3^{n+1}) / 2$$

$$B(\omega_2^{n+1} - \omega_2^n) / \delta = (C - A)(\omega_3^{n+1} \omega_1^n + \omega_3^n \omega_1^{n+1}) / 2$$

$$C(\omega_3^{n+1} - \omega_3^n) / \delta = (A - B)(\omega_1^{n+1} \omega_2^n + \omega_1^n \omega_2^{n+1}) / 2$$

Lax Pair Representation Using Quaternions

The discrete Euler top can be written as

$$A^n = \mathbf{i}z_1^n + \mathbf{j}z_2^n + \mathbf{k}z_3^n, B^n = -1 + \mathbf{i}\alpha_1 z_1^n + \mathbf{j}\alpha_2 z_2^n + \mathbf{k}\alpha_3 z_3^n, A^{n+1}B^n = B^{n+1}A^n.$$

Written componentwise,  $A^{n+1}B^n = B^{n+1}A^n$  gives:

$$z_1^{n+1} - z_1^n = (\alpha_3 - \alpha_2)(z_2^{n+1} z_3^n + z_3^{n+1} z_2^n),$$

$$z_2^{n+1} - z_2^n = (\alpha_1 - \alpha_3)(z_3^{n+1} z_1^n + z_1^{n+1} z_3^n),$$

$$z_3^{n+1} - z_3^n = (\alpha_2 - \alpha_1)(z_1^{n+1} z_2^n + z_2^{n+1} z_1^n).$$

## Discrete Matrix Version of Euler's Top

### Discrete Matrix Version of Euler's Top

Let  $X_1^n = X_1(n\delta)$ ,  $X_2^n = X_2(n\delta)$ ,  $X_3^n = X_3(n\delta)$ . The discrete matrix version of Euler's top is

$$\begin{cases} (X_1^{n+1} - X_1^n) / \delta = X_2^{n+1} X_3^n + X_3^{n+1} X_2^n \\ (X_2^{n+1} - X_2^n) / \delta = -(X_3^{n+1} X_1^n + X_1^{n+1} X_3^n) \\ (X_3^{n+1} - X_3^n) / \delta = X_1^{n+1} X_2^n + X_2^{n+1} X_1^n \end{cases}$$

## Lax Pair Representation for Discrete Matrix Version of Euler's Top

$\zeta$  is the spectral parameter. Let  $X_1^n, X_2^n, X_3^n$  be  $K \times K$  matrices, and consider the following dynamical system:

$$F^{n+1}(\zeta) G^n(\zeta) = G^{n+1}(\zeta) F^n(\zeta)$$

$$F^n(\zeta) = \begin{bmatrix} O & O & -X_2^n & X_3^n \\ O & O & -X_3^n & -X_2^n \\ X_2^n & X_3^n & O & O \\ -X_3^n & X_2^n & O & O \end{bmatrix} + \zeta \begin{bmatrix} O & 2X_1^n & O & O \\ -2X_1^n & O & O & O \\ O & O & O & -2X_1^n \\ O & O & 2X_1^n & O \end{bmatrix} \\ + \zeta^2 \begin{bmatrix} O & O & -X_2^n & -X_3^n \\ O & O & X_3^n & -X_2^n \\ X_2^n & -X_3^n & O & O \\ X_3^n & X_2^n & O & O \end{bmatrix},$$

$$G^n(\zeta) = \begin{bmatrix} I & -\delta X_1^n & O & O \\ \delta X_1^n & I & O & O \\ O & O & I & \delta X_1^n \\ O & O & -\delta X_1^n & I \end{bmatrix} - \zeta \delta \begin{bmatrix} O & O & -X_2^n & -X_3^n \\ O & O & X_3^n & -X_2^n \\ X_2^n & -X_3^n & O & O \\ X_3^n & X_2^n & O & O \end{bmatrix},$$

which gives the Lax pair of the discrete matrix version of Euler's top.

## Discrete Nahm Equations

Using the following reduction,

$$X_1^n = \begin{bmatrix} O & T_1^n \\ T_1^n & O \end{bmatrix}$$

$$X_2^n = \begin{bmatrix} O & -T_2^n \\ T_2^n & O \end{bmatrix}$$

$$X_3^n = \begin{bmatrix} T_3^n & O \\ O & -T_3^n \end{bmatrix}$$

we obtain the following discrete Nahm equations:

$$\begin{cases} (T_1^{n+1} - T_1^n) / \delta = T_2^{n+1} T_3^n - T_3^{n+1} T_2^n \\ (T_2^{n+1} - T_2^n) / \delta = T_3^{n+1} T_1^n - T_1^{n+1} T_3^n \\ (T_3^{n+1} - T_3^n) / \delta = T_1^{n+1} T_2^n - T_2^{n+1} T_1^n \end{cases}$$

The Nahm equations arise in differential geometry and gauge theory, and their successful discretization suggests the possibility of constructing a discrete differential geometry or a discrete-time gauge theory.

# Reduction from the Discrete Nahm Equations to the Discrete Toda Equations with Periodic Boundary Conditions(1)

Using the following reductions,

$$T_1^n = -\frac{1}{\sqrt{2}} \begin{bmatrix} 0 & a_1^n & & & c_K^n \\ c_1^n & 0 & a_2^n & & \\ & c_2^n & 0 & & \\ & & \ddots & \ddots & a_{K-1}^n \\ a_K^n & & & c_{K-1}^n & 0 \end{bmatrix},$$

$$T_2^n = \frac{i}{\sqrt{2}} \begin{bmatrix} 0 & a_1^n & & & -c_K^n \\ -c_1^n & 0 & a_2^n & & \\ & -c_2^n & 0 & & \\ & & \ddots & \ddots & a_{K-1}^n \\ a_K^n & & & -c_{K-1}^n & 0 \end{bmatrix},$$

$$T_3^n = -i \times \text{diag.} (b_1^n, b_2^n, \dots, b_K^n),$$

## Reduction from the Discrete Nahm Equations to the Discrete Toda Equations with Periodic Boundary Conditions(2)

we obtain

$$\left\{ \begin{array}{l} (b_k^{n+1} - b_k^n) / \delta = c_{k-1}^{n+1} a_{k-1}^n - a_k^{n+1} c_k^n \\ (a_k^{n+1} - a_k^n) / \delta = b_k^{n+1} a_k^n - a_k^{n+1} b_{k+1}^n \\ (c_k^{n+1} - c_k^n) / \delta = c_k^{n+1} b_k^n - b_{k+1}^{n+1} c_k^n \\ a_{K+1}^n = a_1^n, \quad b_{K+1}^n = b_1^n, \quad c_{K+1}^n = c_1^n \end{array} \right.$$

Let  $f_k^n = \frac{1}{2\delta} + \frac{1}{2}b_k^n$ ,  $g_k^n = -\frac{1}{2}a_k^n c_k^n$  and  $I_k^n = \frac{1}{\delta^2 f_k^n}$ ,  $V_k^n = \frac{g_k^n}{\delta^2 f_k^n f_{k+1}^n}$ . Then we obtain the discrete Toda equations with periodic boundary conditions:

$$\left\{ \begin{array}{l} (I_k^{n+1} - I_k^n) / \delta = V_{k-1}^{n+1} - V_k^n \quad \text{for } k = 1, \dots, K \\ (V_k^{n+1} - V_k^n) / \delta = V_k^{n+1} I_k^n - V_k^n I_{k+1}^{n+1} \quad \text{for } k = 1, \dots, K \\ I_{K+1}^n = I_1^n \\ V_{K+1}^n = V_1^n \end{array} \right.$$

The discrete Toda equations with periodic boundary conditions play an important role in data science.

# Our Aim(1)

## Background

- ▶ Principal Component Analysis (PCA) is effective in data science, so it is important to accelerate PCA.
- ▶ PCA is essentially equivalent to a partial singular value decomposition. In the PCA using the statistical computing software R, singular value decomposition is performed by upper bidiagonalization followed by the QR method applied to the upper bidiagonal matrix.
- ▶ However, **with the QR method, all singular values and singular vectors are computed**. Therefore, when the matrix dimension is large, the computation time becomes substantial.
- ▶ **The combination of bisection and inverse iteration is not always suitable because it can perform poorly for certain matrices.**

## Our Aim(2)

The glued Kimura matrix  $gB$

$$B = \text{bidiag.} \begin{bmatrix} 1 & \cdots & \cdots & \cdots & \cdots & 1 \\ 9 & 8 & \cdots & 1 & \cdots & 8 & 9 \end{bmatrix}, \quad gB = \begin{bmatrix} B & \delta & & \\ & B & \delta & \\ & & \ddots & \ddots \\ & & & B \end{bmatrix}$$

$\delta \equiv 10.0^{-8}$ . Note that  $\delta$ , not  $\delta I$ .

Inserting  $\delta$  at each junction of  $B$  and  $B \rightarrow gB$  has a cluster of singular values.

A method for observing how the inverse iteration method fails

- ▶ Create a symmetric tridiagonal matrix  $F'$  of size  $1564 \times 1564$  whose eigenvalues are the 782 singular values of the  $782 \times 782$  matrix  $gB$  and their negatives.
- ▶ For  $F'$ , we compute 184 eigenvalues and eigenvectors using Intel MKL's combined function of bisection and inverse iteration (DSTEVX).

## Our Aim(3)

### Purpose

- ▶ Develop a solution method that is as reliable as the QR method while computing only selected singular values and singular vectors.
- ▶ The target computational complexity is as follows.

For an  $n \times n$  upper bidiagonal matrix, when computing  $w$  singular values and their associated singular vectors in descending order,  
 Inverse iteration method:  $O(w^2n)$  = Proposed method:  $O(w^2n)$   
 $<$  QR method:  $O(n^3)$

- ▶ Unlike the combination of bisection and inverse iteration, the new algorithm has no known input matrices on which it performs poorly.

# About PCA(1)

## Principal Component Analysis(1)

- ▶ At the checkout, customers' data are collected.



purchase      condition

|          |  |  |  |                 |  |  |  |
|----------|--|--|--|-----------------|--|--|--|
| ┌───┐    |  |  |  | ┌───┐           |  |  |  |
| purchase |  |  |  | condition       |  |  |  |
| magazine |  |  |  | weather is rain |  |  |  |
| umbrella |  |  |  | twenties        |  |  |  |
| pudding  |  |  |  | thirties        |  |  |  |
| ...      |  |  |  | ...             |  |  |  |

customer1

1 0 0 ... 0 1 0 ...

customer2

0 1 0 ... 1 0 0 ...

## About PCA(2)

### Principal Component Analysis(2)

- Represent the customers' data as a matrix  $A = (a_{i,j}) \in \mathbb{R}^{m \times n}$ .

$$A = \begin{matrix} & \begin{matrix} \text{magazine} \\ \text{umbrella} \\ \text{pudding} \\ \vdots \\ \text{weather is rain} \\ \text{twenties} \\ \vdots \end{matrix} & \end{matrix} \begin{pmatrix} \text{customer1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \text{customer2} & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ \text{customer3} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 1 \\ & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & \vdots & & & & \vdots & & & \ddots \end{pmatrix}$$

## About PCA(3) – Normalization in PCA

### Principal Component Analysis(3)

- ▶ Normalize the matrix  $A$  to obtain  $C = (A - M)S$ .

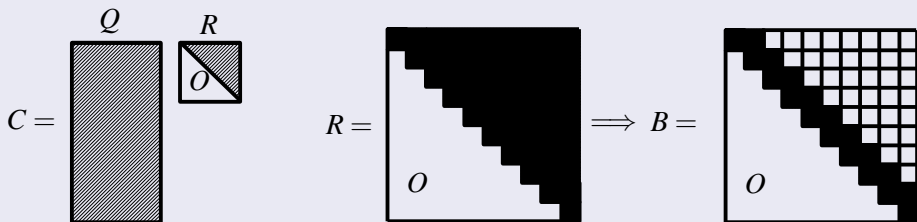
$$M = \begin{bmatrix} \frac{1}{m} (\sum_{i=1}^m a_{i,1}) & \cdots & \frac{1}{m} (\sum_{i=1}^m a_{i,n}) \\ \frac{1}{m} (\sum_{i=1}^m a_{i,1}) & \cdots & \frac{1}{m} (\sum_{i=1}^m a_{i,n}) \\ \vdots & \vdots & \vdots \\ \frac{1}{m} (\sum_{i=1}^m a_{i,1}) & \cdots & \frac{1}{m} (\sum_{i=1}^m a_{i,n}) \end{bmatrix}$$

$$S = \text{diag.} \left[ \frac{1}{\sqrt{\sum_{i=1}^m ((A-M)_{i,1})^2}}, \cdots, \frac{1}{\sqrt{\sum_{i=1}^m ((A-M)_{i,n})^2}} \right]$$

- ▶ PCA is realized by partial singular value decomposition of  $C$ .
- ▶ Only the  $w$  largest singular values and their associated right singular vectors are required.

## About PCA(4) – Direct Method

### Upper bidiagonal reduction



### Calculation of PCA ( $w$ is the number of singular values needed for PCA)

$$B\tilde{v}_i^{(2)} = \sigma_i \tilde{u}_i^{(2)} \quad (i = 1, \dots, w), \quad B^\top \tilde{u}_i^{(2)} = \sigma_i \tilde{v}_i^{(2)} \quad (i = 1, \dots, w),$$

$$\tilde{U}^{(2)} := [\tilde{u}_1^{(2)}, \tilde{u}_2^{(2)}, \dots, \tilde{u}_w^{(2)}] \in \mathbb{R}^{n \times w}, \quad \tilde{V}^{(2)} := [\tilde{v}_1^{(2)}, \tilde{v}_2^{(2)}, \dots, \tilde{v}_w^{(2)}] \in \mathbb{R}^{n \times w},$$

$$\Sigma := \text{diag.}(\sigma_1, \sigma_2, \dots, \sigma_w) \in \mathbb{R}^{w \times w}, \quad \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_w > 0$$

Singular values of  $C$ :  $\Sigma$ , Left singular vectors of  $C$ :  $U = Q\tilde{U}^{(1)}\tilde{U}^{(2)}$ ,

Right singular vectors of  $C$ :  $V = \tilde{V}^{(1)}\tilde{V}^{(2)}$

## Converting Large Singular Values into Small Ones (1)

Conversion from a singular value problem for  $B$  to an eigenvalue problem for a symmetric tridiagonal matrix  $F'$

Let  $B$  be an  $n \times n$  upper bidiagonal matrix,

$$B = \text{bidiag.} \begin{bmatrix} \beta_1 & \cdots & \beta_{n-1} \\ \alpha_1 & \cdots & \cdots & \alpha_n \end{bmatrix} \quad \text{with singular values } \sigma_1, \dots, \sigma_n$$

Define the  $2n \times 2n$  symmetric tridiagonal matrix  $F'$

$$F' = \begin{bmatrix} 0 & \alpha_1 & & & O \\ \alpha_1 & \ddots & \beta_1 & & \\ & \beta_1 & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_{n-1} \\ & & & \beta_{n-1} & \ddots & \alpha_n \\ O & & & & \alpha_n & 0 \end{bmatrix} \quad \text{whose eigenvalues are } \pm\sigma_1, \dots, \pm\sigma_n$$

Singular vectors of  $B$  can be computed from eigenvectors of  $F'$ .

## Converting Large Singular Values into Small Ones (2)

Making  $F'$  positive definite and converting back to a singular value problem

Let  $\mu$  be an upper bound of the largest singular value of  $B$ , and let  $v = \sqrt{\mu}$ . Then

$$v^2 I - F' = \begin{bmatrix} v^2 & -\alpha_1 & & & & O \\ -\alpha_1 & \ddots & -\beta_1 & & & \\ & -\beta_1 & \ddots & \ddots & & \\ & & \ddots & \ddots & -\beta_{n-1} & \\ & & & -\beta_{n-1} & \ddots & -\alpha_n \\ O & & & & -\alpha_n & v^2 \end{bmatrix}$$

is symmetric and positive definite, so we can apply a modified Cholesky decomposition:  $v^2 I - F' = (L)^\top L$ . ( $L$  is a  $2n \times 2n$  bidiagonal matrix.)

## Recurrence formulas for the modified Cholesky decomposition

$$\mathbf{v}^2 \mathbf{I} - \mathbf{F}' = (\mathbf{L})^\top \mathbf{L}, \quad \mathbf{L} = \begin{bmatrix} s_1 & & & & \\ t_1 & s_2 & & & \\ & \ddots & \ddots & & \\ & & t_{2n-1} & s_{2n} & \end{bmatrix}$$

is equivalent to the following recurrence relations.

$$\begin{aligned} s_{2n} &= \mathbf{v} \\ t_{2n-1} &= -\frac{\alpha_n}{s_{2n}} \\ s_{2n-1} &= \sqrt{\mathbf{v} - t_{2n-1}} \sqrt{\mathbf{v} + t_{2n-1}} \\ t_{2n-2} &= -\frac{\beta_{n-1}}{s_{2n-1}} \\ s_{2n-2} &= \sqrt{\mathbf{v} - t_{2n-2}} \sqrt{\mathbf{v} + t_{2n-2}} \end{aligned}$$

$$\begin{aligned} t_{2n-3} &= -\frac{\alpha_{n-1}}{s_{2n-2}} \\ s_{2n-3} &= \sqrt{\mathbf{v} - t_{2n-3}} \sqrt{\mathbf{v} + t_{2n-3}} \\ t_{2n-4} &= -\frac{\beta_{n-2}}{s_{2n-3}} \\ s_{2n-4} &= \sqrt{\mathbf{v} - t_{2n-4}} \sqrt{\mathbf{v} + t_{2n-4}} \\ &\vdots \end{aligned}$$

Since  $-\mathbf{v} < t_k < \mathbf{v}$  for  $k = 1, \dots, 2n-1$ ,  $t_k$  does not diverge to  $\pm\infty$ .

### Relationship between the singular values of $L$ and $B$

Singular values of  $L$  :  $\sigma'_1 \geq \sigma'_2 \geq \cdots \geq \sigma'_{2n-1} \geq \sigma'_{2n}$

Singular values of  $B$  :  $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{n-1} \geq \sigma_n$

They are related as follows:

$$\sigma_\ell = \left( v - \sigma'_{2n+1-\ell} \right) \left( v + \sigma'_{2n+1-\ell} \right), \quad \ell = 1, \dots, n$$

The problem of computing the largest singular values in order is converted into that of computing the smallest singular values in order.

Relation between the right singular vector  $X$  of  $L$  and the left and right singular vectors  $\tilde{U}^{(2)}$ ,  $\tilde{V}^{(2)}$  of  $B$

$$B\tilde{v}_i^{(2)} = \sigma_i \tilde{u}_i^{(2)} \quad (i = 1, \dots, w), \quad B^\top \tilde{u}_i^{(2)} = \sigma_i \tilde{v}_i^{(2)} \quad (i = 1, \dots, w),$$

$$\tilde{U}^{(2)} := [\tilde{u}_1^{(2)}, \tilde{u}_2^{(2)}, \dots, \tilde{u}_w^{(2)}] \in \mathbb{R}^{n \times w}, \quad \tilde{V}^{(2)} := [\tilde{v}_1^{(2)}, \tilde{v}_2^{(2)}, \dots, \tilde{v}_w^{(2)}] \in \mathbb{R}^{n \times w}$$

$$X = \begin{bmatrix} \cdots & \tilde{v}_w^{(2)}(1) & \cdots & \tilde{v}_2^{(2)}(1) & \tilde{v}_1^{(2)}(1) \\ \cdots & \tilde{u}_w^{(2)}(1) & \cdots & \tilde{u}_2^{(2)}(1) & \tilde{u}_1^{(2)}(1) \\ \cdots & \tilde{v}_w^{(2)}(2) & \cdots & \tilde{v}_2^{(2)}(2) & \tilde{v}_1^{(2)}(2) \\ \cdots & \tilde{u}_w^{(2)}(2) & \cdots & \tilde{u}_2^{(2)}(2) & \tilde{u}_1^{(2)}(2) \\ \cdots & \vdots & \cdots & \vdots & \vdots \\ \cdots & \tilde{v}_w^{(2)}(n-1) & \cdots & \tilde{v}_2^{(2)}(n-1) & \tilde{v}_1^{(2)}(n-1) \\ \cdots & \tilde{u}_w^{(2)}(n-1) & \cdots & \tilde{u}_2^{(2)}(n-1) & \tilde{u}_1^{(2)}(n-1) \\ \cdots & \tilde{v}_w^{(2)}(n) & \cdots & \tilde{v}_2^{(2)}(n) & \tilde{v}_1^{(2)}(n) \\ \cdots & \tilde{u}_w^{(2)}(n) & \cdots & \tilde{u}_2^{(2)}(n) & \tilde{u}_1^{(2)}(n) \end{bmatrix} \quad (\ell): \text{the } \ell \text{ component.}$$

The left singular vector of  $L$  is not needed for this purpose.

## Key Point

### Orthogonal qd algorithm with shift (OQDS)<sup>[1]</sup>

- ▶ This method is an algorithm derived from the discrete Toda equations with periodic boundary conditions.
- ▶ If the matrix is not split in the middle of the computation, singular values and their associated singular vectors can be computed in order from the smallest to the largest.
- ▶ Singular values are computed with high accuracy, and the right singular vectors are highly orthogonal.

[1] Urs von Matt, The Orthogonal qd-Algorithm, SIAM Journal on Scientific Computing, Vol. 18, Iss. 4, pp. 1163–1186, 1997

### Aim of this research

Remove the assumption

“the matrix is not split in the middle of the computation.”

Even if the matrix is split, we want to compute singular values and singular vectors in order from the smallest to the largest.

## The OQDS method

Hereafter, we regard the  $2n \times 2n$  matrix  $L$  as a  $p \times p$  matrix, where  $p = 2n$ . Let  $\hat{L}^{(0)} \equiv L$ . For each  $i$ , let  $\hat{L}^{(i)}$  be a  $p \times p$  lower bidiagonal matrix,

$$\hat{L}^{(i)} = \begin{bmatrix} a_1^{(i)} & & & O \\ b_1^{(i)} & a_2^{(i)} & & \\ & \ddots & \ddots & \\ O & & b_{p-1}^{(i)} & a_p^{(i)} \end{bmatrix}$$

Let  $r^{(i)}$  be a shift satisfying  $r^{(i)} \leq \sigma_{\min}(\hat{L}^{(i)})$ , let  $\tilde{G}^{(i)}$  be a product of  $2p \times 2p$  generalized Givens rotations, and let  $\hat{G}^{(i)}$  be a product of  $p \times p$  Givens rotations.

Let  $z^{(0)} = 0$  and  $z^{(i+1)} = \sqrt{(z^{(i)})^2 + (r^{(i)})^2}$ . Then the OQDS method is described by the following relations:

$$\begin{bmatrix} I & O \\ O & (\hat{G}^{(i)})^\top \end{bmatrix} \tilde{G}^{(i)} \begin{bmatrix} \hat{L}^{(i)} \\ z^{(i)} I \end{bmatrix} \hat{G}^{(i)} = \begin{bmatrix} \hat{L}^{(i+1)} \\ z^{(i+1)} I \end{bmatrix}$$

$$(\hat{G}^{(i)})^\top \left( (\hat{L}^{(i)})^\top \hat{L}^{(i)} - (r^{(i)})^2 I \right) \hat{G}^{(i)} = (\hat{L}^{(i+1)})^\top \hat{L}^{(i+1)}$$

## Moving Zero Singular Values to the Lower Block

### Key technique

If the matrix splits into two blocks, zero singular values cannot be moved from the upper block to the lower block.

In the usual OQDS method, the two blocks are treated as separate matrices. However, to compute singular values and their singular vectors in order from the smallest, the matrix must be treated as a whole.

### Redefining the Givens Rotation

$$\begin{bmatrix} I & & & \\ & \cos \theta & \sin \theta & \\ & & I & \\ & -\sin \theta & \cos \theta & \\ & & & I \end{bmatrix} \begin{bmatrix} \vdots \\ x \\ \vdots \\ y \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \sqrt{x^2 + y^2} \\ \vdots \\ 0 \\ \vdots \end{bmatrix}, \quad \begin{cases} \cos \theta = \frac{x}{\sqrt{x^2 + y^2}} \\ \sin \theta = \frac{y}{\sqrt{x^2 + y^2}} \end{cases}$$

When  $x = 0$  and  $y = 0$ , the conventional definition gives  $\cos \theta = 1$  and  $\sin \theta = 0$ . The new definition gives  $\cos \theta = 0$  and  $\sin \theta = 1$ .

## OQD method without shift (left: conventional, right: modified)

```

1: Set  $\rho_1^{(i)} = \alpha_1^{(i)}$ 
2: if  $\rho_1^{(i)} \leq \sqrt{\varepsilon} z^{(i)}$  then
3:   Set  $\rho_1^{(i)} = 0$ 
4: end if
5: for  $k := 1, 2, \dots, p-1$  do
6:   Set  $\gamma_k^{(i)} = \sqrt{(\rho_k^{(i)})^2 + (\beta_k^{(i)})^2}$ 
7:   if  $\gamma_k^{(i)} = 0$  then
8:     Set  $\zeta_k^{(i)} = 0, \rho_{k+1}^{(i)} = \alpha_{k+1}^{(i)}$ 
9:   else
10:    Set  $\zeta_k^{(i)} = (\beta_k^{(i)} / \gamma_k^{(i)}) \alpha_{k+1}^{(i)}$ 
11:    Set  $\rho_{k+1}^{(i)} = (\rho_k^{(i)} / \gamma_k^{(i)}) \alpha_{k+1}^{(i)}$ 
12:    if  $\rho_{k+1}^{(i)} \leq \sqrt{\varepsilon} z^{(i)}$  then
13:      Set  $\rho_{k+1}^{(i)} = 0$ 
14:    end if
15:  end if
16: end for
17: Set  $\gamma_p^{(i)} = \rho_p^{(i)}$ 

```

```

1: Set  $\rho_1^{(i)} = \alpha_1^{(i)}$ 
2: if  $\rho_1^{(i)} \leq \sqrt{\varepsilon} z^{(i)}$  then
3:   Set  $\rho_1^{(i)} = 0$ 
4: end if
5: for  $k := 1, 2, \dots, p-1$  do
6:   Set  $\gamma_k^{(i)} = \sqrt{(\rho_k^{(i)})^2 + (\beta_k^{(i)})^2}$ 
7:   if  $\gamma_k^{(i)} = 0$  then
8:     Set  $\zeta_k^{(i)} = \alpha_{k+1}^{(i)}, \rho_{k+1}^{(i)} = 0$ 
9:   else
10:    Set  $\zeta_k^{(i)} = (\beta_k^{(i)} / \gamma_k^{(i)}) \alpha_{k+1}^{(i)}$ 
11:    Set  $\rho_{k+1}^{(i)} = (\rho_k^{(i)} / \gamma_k^{(i)}) \alpha_{k+1}^{(i)}$ 
12:    if  $\rho_{k+1}^{(i)} \leq \sqrt{\varepsilon} z^{(i)}$  then
13:      Set  $\rho_{k+1}^{(i)} = 0$ 
14:    end if
15:  end if
16: end for
17: Set  $\gamma_p^{(i)} = \rho_p^{(i)}$ 

```

## Modifying the Implementation to Compute Singular Values in Ascending Order Even When a Split Occurs

Use the smallest singular value as the shift

1. If  $\beta_1^{(i)} = 0$ , then  $\alpha_1^{(i)} - r^{(i)} = 0$  is allowed.
2. If  $\beta_{k+1}^{(i)} = 0$  for  $k = 1, \dots, p-2$ , then  $\left(\rho_k^{(i)} / \gamma_k^{(i)}\right) \alpha_{k+1}^{(i)} - r^{(i)} = 0$  is allowed.

With these improvements, the entire matrix can be treated as a single matrix, even if a split occurs during the computation.

This feature is not needed in the usual QQDS iteration.

It is needed, however, to ensure that singular values are computed sequentially from the smallest one.

The next two slides present the algorithm before and after these modifications.

## Original OQDS method

```

1: Set  $temp = \alpha_1^{(i)} - r^{(i)}$ 
2: if  $temp > 0$  then
3:   Set  $\rho_1^{(i)} = \sqrt{temp} \times \sqrt{\alpha_1^{(i)} + r^{(i)}}$ 
4: else
5:   Fail. Retry with a smaller value of  $r^{(i)}$ .
6: end if
7: for  $k := 1, 2, \dots, p-1$  do
8:   Set  $\gamma_k^{(i)} = \sqrt{(\rho_k^{(i)})^2 + (\beta_k^{(i)})^2}$ 
9:   Set  $\zeta_k^{(i)} = (\beta_k^{(i)} / \gamma_k^{(i)}) \alpha_{k+1}^{(i)}$ 
10:  Set  $temp = (\rho_k^{(i)} / \gamma_k^{(i)}) \alpha_{k+1}^{(i)} - r^{(i)}$ 
11:  if  $temp > 0$  or  $(k = p-1 \text{ and } temp = 0)$  then
12:    Set  $\rho_{k+1}^{(i)} = \sqrt{temp} \times \sqrt{(\rho_k^{(i)} / \gamma_k^{(i)}) \alpha_{k+1}^{(i)} + r^{(i)}}$ 
13:  else
14:    Fail. Retry with a smaller value of  $r^{(i)}$ .
15:  end if
16: end for
17: Set  $\gamma_p^{(i)} = \rho_p^{(i)}$ 

```

## Modified OQDS method

```

1: Set  $\beta_p^{(i)} = 0$ 
2: Set  $temp = \alpha_1^{(i)} - r^{(i)}$ 
3: if  $temp > 0$  or  $(\beta_1^{(i)} = 0 \text{ and } temp = 0)$  then
4:   Set  $\rho_1^{(i)} = \sqrt{temp} \times \sqrt{\alpha_1^{(i)} + r^{(i)}}$ 
5: else
6:   Fail. Retry with a smaller value of  $r^{(i)}$ .
7: end if
8: for  $k := 1, 2, \dots, p-1$  do
9:   Set  $\gamma_k^{(i)} = \sqrt{(\rho_k^{(i)})^2 + (\beta_k^{(i)})^2}$ 
10:  if  $\gamma_k^{(i)} = 0$  then
11:    Set  $\zeta_k^{(i)} = 0, \rho_{k+1}^{(i)} = \alpha_{k+1}^{(i)}$ 
12:  else
13:    Set  $\zeta_k^{(i)} = (\beta_k^{(i)} / \gamma_k^{(i)}) \alpha_{k+1}^{(i)}$ 
14:    Set  $temp = (\rho_k^{(i)} / \gamma_k^{(i)}) \alpha_{k+1}^{(i)} - r^{(i)}$ 
15:    if  $temp > 0$  or  $(\beta_{k+1}^{(i)} = 0 \text{ and } temp = 0)$  then
16:      Set  $\rho_{k+1}^{(i)} = \sqrt{temp} \times \sqrt{(\rho_k^{(i)} / \gamma_k^{(i)}) \alpha_{k+1}^{(i)} + r^{(i)}}$ 
17:    else
18:      Fail. Retry with a smaller value of  $r^{(i)}$ .
19:    end if
20:  end if
21: end for

```

## Numerical Experiments

### Experimental Environment

CPU: Intel(R) Core(TM) i9-10920X 3.50 GHz, 12 core, 24 thread, MEM: 256GB

OS: Rocky Linux v9.8, COMPILER: GFortran, LIB: Intel MKL

### Evaluation Criteria

The number of singular values required for principal component analysis is  $w = 5$ . Let  $\sigma_i$  ( $i = 1, \dots, w$ ) be the singular values of  $B$ , with  $\sigma_{\max} = \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_w$ ,  $\hat{\Sigma}_w = \text{diag}(\sigma_1, \dots, \sigma_w)$ , and let  $\hat{U}_w$  and  $\hat{V}_w$  be the  $n \times w$  matrices of left and right singular vectors of  $B$ , respectively.

(1) elapsed time, (2)  $\|B\hat{V}_w - \hat{U}_w\hat{\Sigma}_w\|_F + \|B^\top\hat{U}_w - \hat{V}_w\hat{\Sigma}_w\|_F$ , (3)  $\|\hat{U}_w^\top\hat{U}_w - I_w\|_F$ , and (4)  $\|\hat{V}_w^\top\hat{V}_w - I_w\|_F$

### Methods Compared

- ▶ Demmel–Kahan QR method
- ▶ Bisection method + inverse iteration
- ▶ Proposed method using the OQDS method

## Test Matrices

- ▶ TYPE-I: an upper bidiagonal matrix  $B$  obtained by reducing a square matrix generated from uniform random numbers in  $[0, 1]$ 
  - ▶ The largest singular value is overwhelmingly dominant; setting  $w = 5$  is therefore not representative of the intended PCA setting
  - ▶ Matrix sizes: 10,000, 15,000, 20,000, 25,000, 30,000
- ▶ TYPE-II: an upper bidiagonal matrix  $B$  generated from uniform random numbers in  $[0, 1]$ 
  - ▶ The five largest singular values are all simple.
  - ▶ Matrix sizes: 1,000, 10,000, 100,000, 1,000,000, 10,000,000
- ▶ TYPE-III: an upper bidiagonal matrix  $B$  whose diagonal and superdiagonal entries are all 1
  - ▶ A matrix with clustered singular values  $\sigma_k = 2 \cos \frac{2k-1}{4n+2} \pi, k = 1, \dots, n$
  - ▶ Matrix sizes: 1,000, 10,000, 100,000, 1,000,000, 10,000,000
- ▶ TYPE-IV: the glued Kimura matrix
  - ▶ It has numerically multiple singular values in double-precision arithmetic.
  - ▶ Matrix sizes: 1,700, 17,000, 170,000, 1,700,000, 17,000,000

## Numerical Experiments TYPE-I (1)

TYPE-I: an upper bidiagonal matrix  $B$  obtained by reducing a square matrix generated from uniform random numbers in  $[0, 1]$

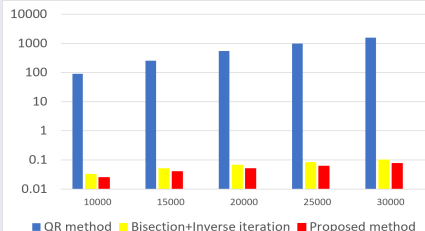
### Elapsed Time Comparison

All three methods use parallel implementations; unit: seconds.

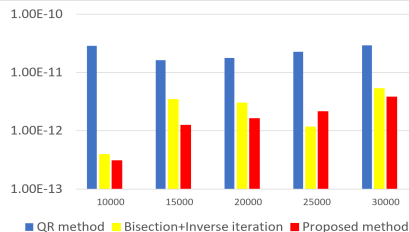
| Dimension | QR method | Bisection<br>+ Inverse iteration | Proposed method |
|-----------|-----------|----------------------------------|-----------------|
| 10,000    | 90.034    | 0.032                            | 0.025           |
| 15,000    | 255.334   | 0.050                            | 0.040           |
| 20,000    | 542.813   | 0.066                            | 0.050           |
| 25,000    | 978.654   | 0.082                            | 0.061           |
| 30,000    | 1582.566  | 0.099                            | 0.076           |

# Numerical Experiments TYPE-I (2)

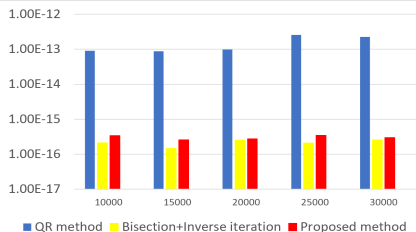
Elapsed time (sec.)



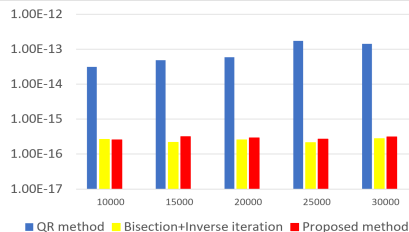
$\|B\hat{V}_w - \hat{U}_w\hat{\Sigma}_w\|_F + \|B^\top\hat{U}_w - \hat{V}_w\hat{\Sigma}_w\|_F$



$\|\hat{U}_w^\top \hat{U}_w - I_w\|_F$



$\|\hat{V}_w^\top \hat{V}_w - I_w\|_F$



## Numerical Experiments TYPE-II (1)

TYPE-II: an upper bidiagonal matrix  $B$  generated from uniform random numbers in  $[0, 1]$

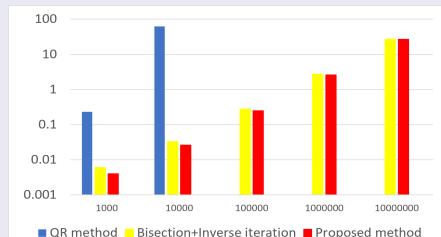
### Elapsed Time Comparison

All three methods use parallel implementations; unit: seconds.

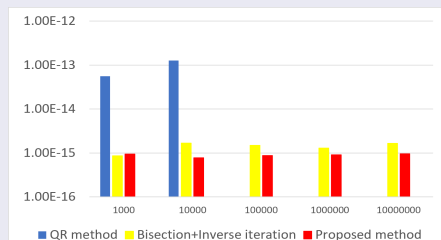
| Dimension  | QR method   | Bisection<br>+ Inverse iteration | Proposed method |
|------------|-------------|----------------------------------|-----------------|
| 1,000      | 0.232       | 0.006                            | 0.004           |
| 10,000     | 62.535      | 0.033                            | 0.026           |
| 100,000    | Out of mem. | 0.278                            | 0.249           |
| 1,000,000  | Out of mem. | 2.751                            | 2.598           |
| 10,000,000 | Out of mem. | 26.836                           | 26.820          |

# Numerical Experiments TYPE-II (2)

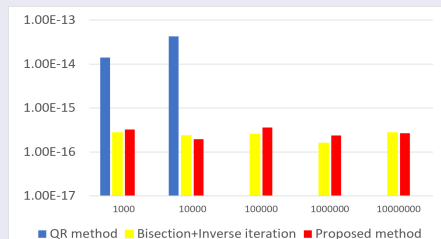
Elapsed time (sec.)



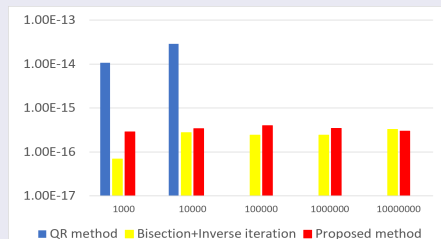
$\|B\hat{V}_w - \hat{U}_w\hat{\Sigma}_w\|_F + \|B^\top\hat{U}_w - \hat{V}_w\hat{\Sigma}_w\|_F$



$\|\hat{U}_w^\top \hat{U}_w - I_w\|_F$



$\|\hat{V}_w^\top \hat{V}_w - I_w\|_F$



## Numerical Experiments TYPE-III (1)

TYPE-III: an upper bidiagonal matrix  $B$  whose diagonal and superdiagonal entries are all 1

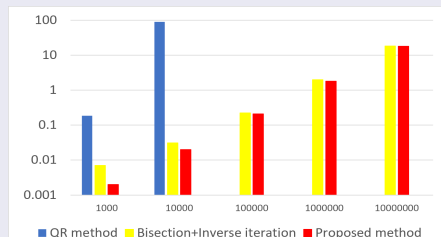
### Elapsed Time Comparison

All three methods use parallel implementations; unit: seconds.

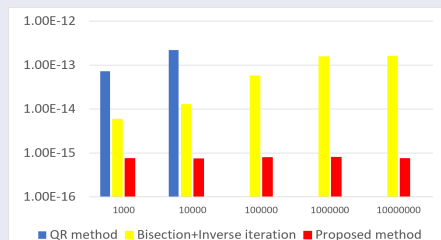
| Dimension  | QR method   | Bisection<br>+ Inverse iteration | Proposed method |
|------------|-------------|----------------------------------|-----------------|
| 1,000      | 0.185       | 0.007                            | 0.002           |
| 10,000     | 90.352      | 0.031                            | 0.020           |
| 100,000    | Out of mem. | 0.224                            | 0.209           |
| 1,000,000  | Out of mem. | 1.992                            | 1.815           |
| 10,000,000 | Out of mem. | 18.278                           | 18.133          |

# Numerical Experiments TYPE-III (2)

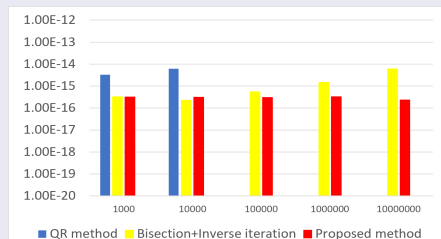
Elapsed time (sec.)



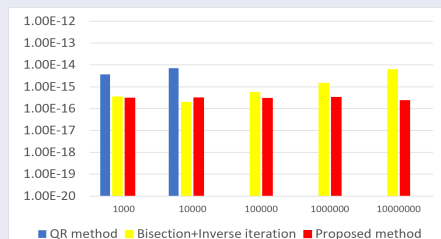
$\|B\hat{V}_w - \hat{U}_w\hat{\Sigma}_w\|_F + \|B^\top\hat{U}_w - \hat{V}_w\hat{\Sigma}_w\|_F$



$\|\hat{U}_w^\top \hat{U}_w - I_w\|_F$



$\|\hat{V}_w^\top \hat{V}_w - I_w\|_F$



## Numerical Experiments TYPE-IV (1)

TYPE-IV: the glued Kimura matrix

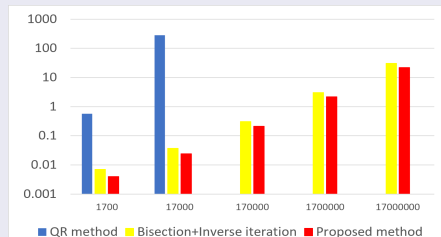
### Elapsed Time Comparison

All three methods use parallel implementations; unit: seconds.

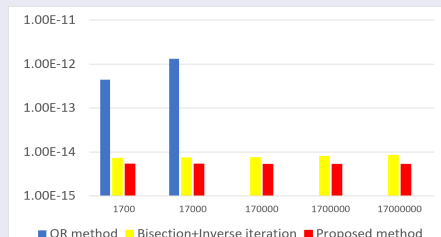
| Dimension  | QR method   | Bisection<br>+ Inverse iteration | Proposed method |
|------------|-------------|----------------------------------|-----------------|
| 1,700      | 0.564       | 0.007                            | 0.004           |
| 17,000     | 281.378     | 0.037                            | 0.024           |
| 170,000    | Out of mem. | 0.309                            | 0.211           |
| 1,700,000  | Out of mem. | 3.046                            | 2.172           |
| 17,000,000 | Out of mem. | 30.808                           | 22.117          |

# Numerical Experiments TYPE-IV (2)

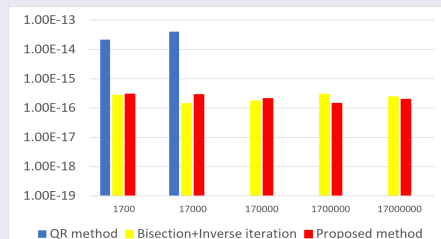
Elapsed time (sec.)



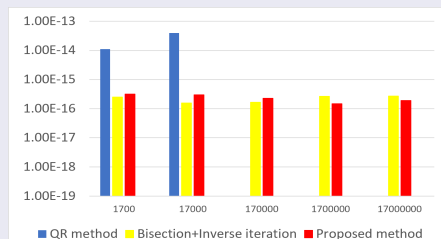
$\|B\hat{V}_w - \hat{U}_w\hat{\Sigma}_w\|_F + \|B^\top \hat{U}_w - \hat{V}_w\hat{\Sigma}_w\|_F$



$\|\hat{U}_w^\top \hat{U}_w - I_w\|_F$



$\|\hat{V}_w^\top \hat{V}_w - I_w\|_F$



# Summary

## Summary

- ▶ We developed a method that computes only selected singular values and singular vectors.
- ▶ It achieves the same level of accuracy as the combination of bisection and inverse iteration.
- ▶ The target computational complexity has been achieved, as follows.

For an  $n \times n$  upper bidiagonal matrix, when computing  $w$  singular values and their associated singular vectors in descending order,

$$\begin{aligned} \text{Inverse iteration method: } O(w^2n) &= \text{Proposed method: } O(w^2n) \\ &< \text{QR method: } O(n^3) \end{aligned}$$

- ▶ Unlike the combination of bisection and inverse iteration, our new algorithm has no known input matrices on which it performs poorly.