A Study on Parallelization of Successive Rotation Based Joint Diagonalization

Xiu-Lin Wang, Xiao-Feng Gong, Qiu-Hua Lin
Dalian University of Technology, China

19^{th} DSP, August, 2014

Content

1. Introduction

Joint diagonalization

JD seeks unloading matrix $\boldsymbol{B}$ so that $\boldsymbol{B}\boldsymbol{C}_k\boldsymbol{B}^H$ are diagonal

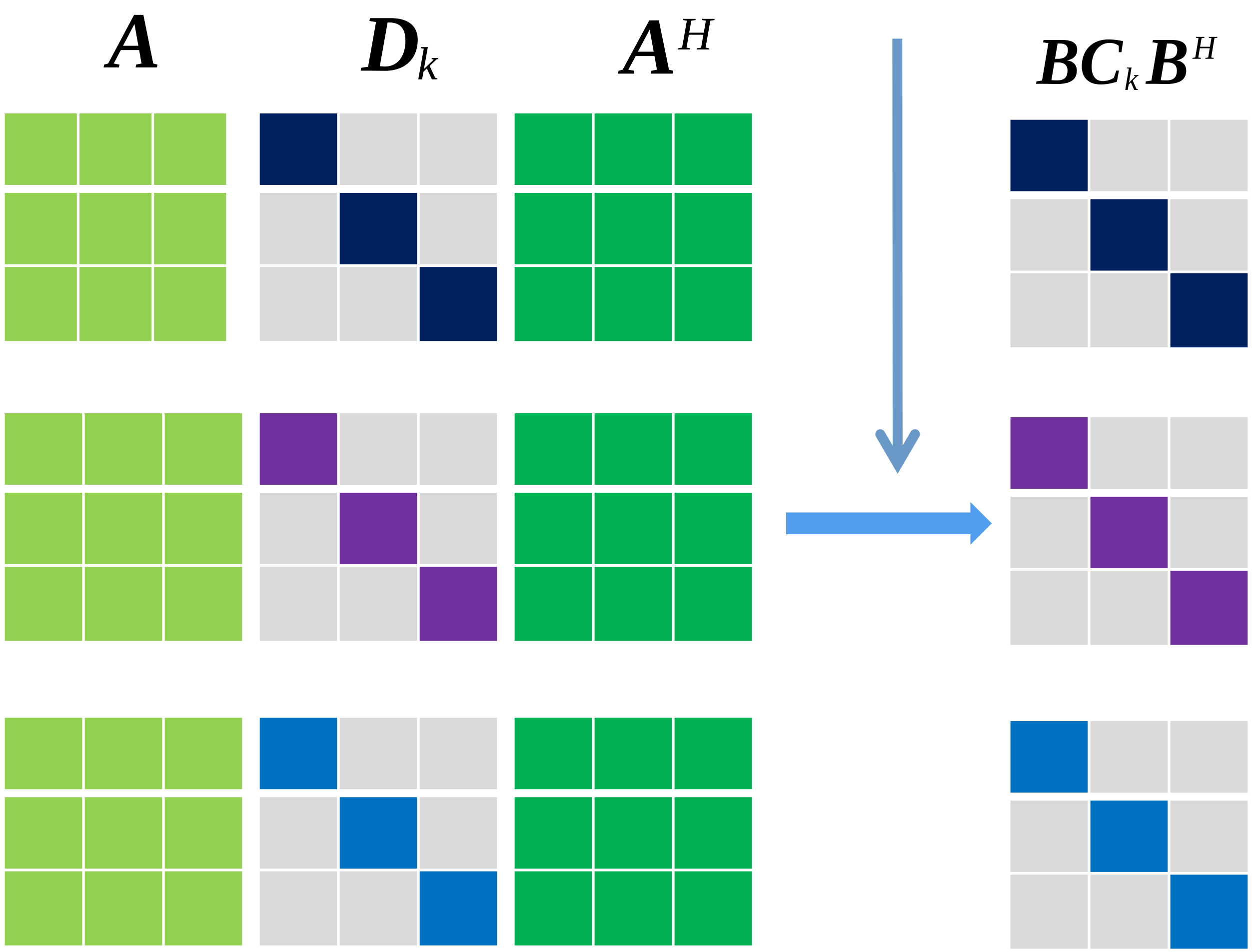
Target matrices

$$\boldsymbol{C}_1 = \boldsymbol{A}\boldsymbol{D}_1\boldsymbol{A}^H$$

$$\boldsymbol{C}_2 = \boldsymbol{A}\boldsymbol{D}_2\boldsymbol{A}^H$$

……

$$\boldsymbol{C}_k = \boldsymbol{A}\boldsymbol{D}_k\boldsymbol{A}^H$$



1. Introduction

Joint Diagonalization(JD) has been widely applied

- Array processing (X. F. Gong, 2012)
- Tensor decomposition (L. Delathauwer, 2008; X. F. Gong, 2013)
- Speech signal processing (D. T. Pham, 2003)
- Blind source separation (J. F. Cardoso, 1993; A. Mesloub, 2014)

Slicewise form: $\boldsymbol{\Gamma}(:,:,k) = \boldsymbol{A} \times \boldsymbol{D}_k \times \boldsymbol{B}^T$

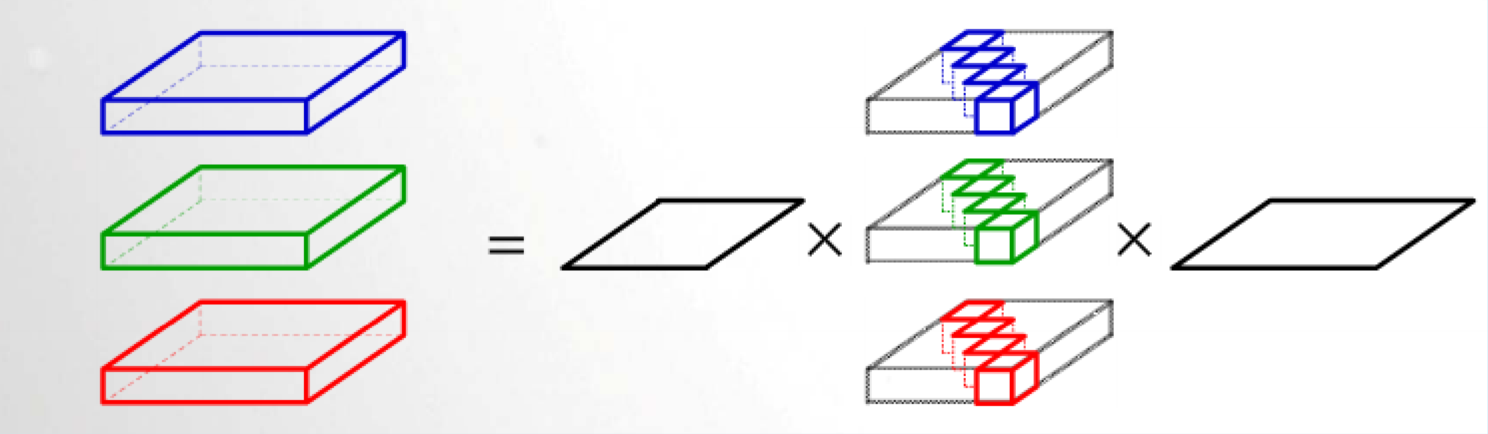


Fig.1 Visualization of models for CPD

1. Introduction

Joint Diagonalization(JD) has been widely applied

- Array processing (X. F. Gong, 2012)
- Tensor decomposition (L. Delathauwer, 2008; X. F. Gong, 2013)
- Speech signal processing (D. T. Pham, 2003)
- Blind source separation (J. F. Cardoso, 1993; A. Mesloub, 2014)

Slicewise form: $\boldsymbol{\Gamma}(:,:,k) = \boldsymbol{A} \times \boldsymbol{D}_k \times \boldsymbol{B}^T$

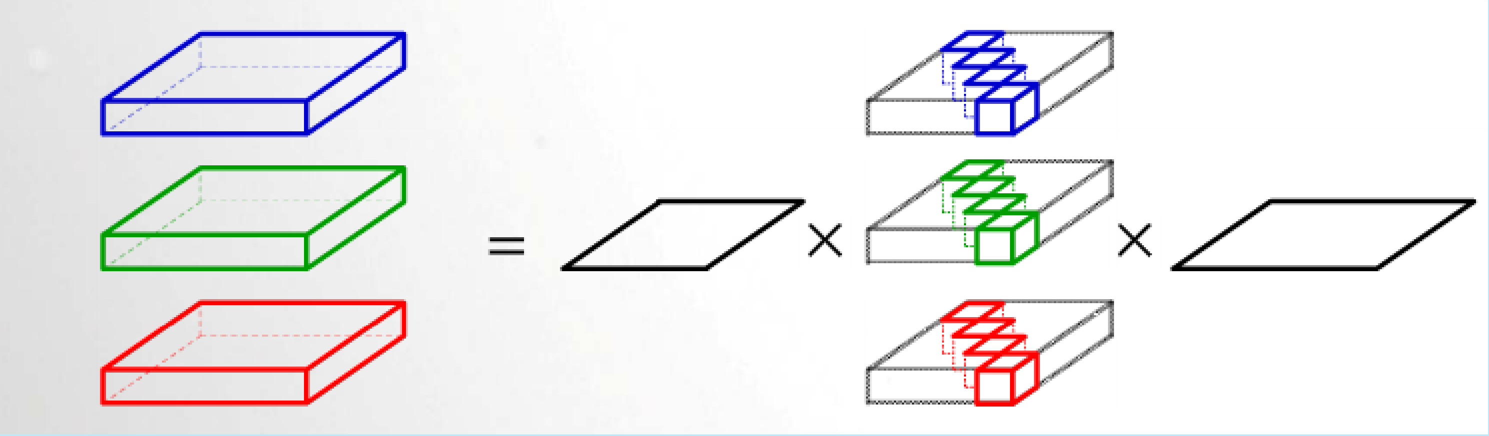


Fig.1 Visualization of models for CPD

1. Introduction

Joint Diagonalization(JD) has been widely applied

- Array processing (X. F. Gong, 2012)
- Tensor decomposition (L. Delathauwer, 2008; X. F. Gong, 2013)
- Speech signal processing (D. T. Pham, 2003)
- Blind source separation (J. F. Cardoso, 1993; A. Mesloub, 2014)

Slicewise form: $\boldsymbol{\Gamma}(:,:,k) = \boldsymbol{A} \times \boldsymbol{D}_k \times \boldsymbol{B}^T$

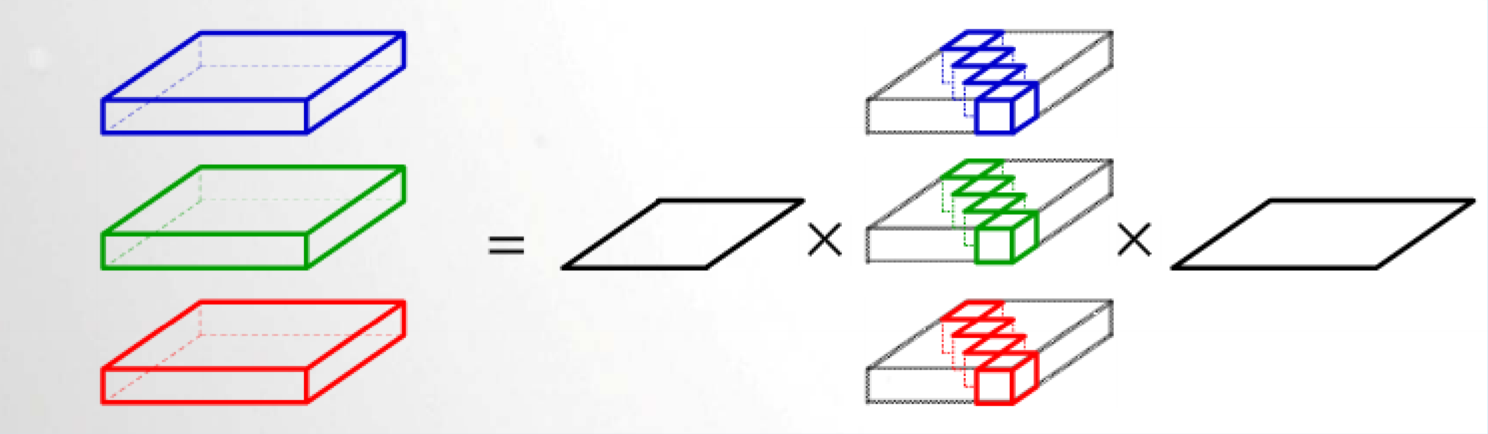


Fig.1 Visualization of models for CPD

1. Introduction

Joint Diagonalization(JD) has been widely applied

- Array processing (X. F. Gong, 2012)
- Tensor decomposition (L. Delathauwer, 2008; X. F. Gong, 2013)
- Speech signal processing (D. T. Pham, 2003)
- Blind source separation (J. F. Cardoso, 1993; A. Mesloub, 2014)

Slicewise form: $\boldsymbol{\Gamma}(:,:,k) = \boldsymbol{A} \times \boldsymbol{D}_k \times \boldsymbol{B}^T$

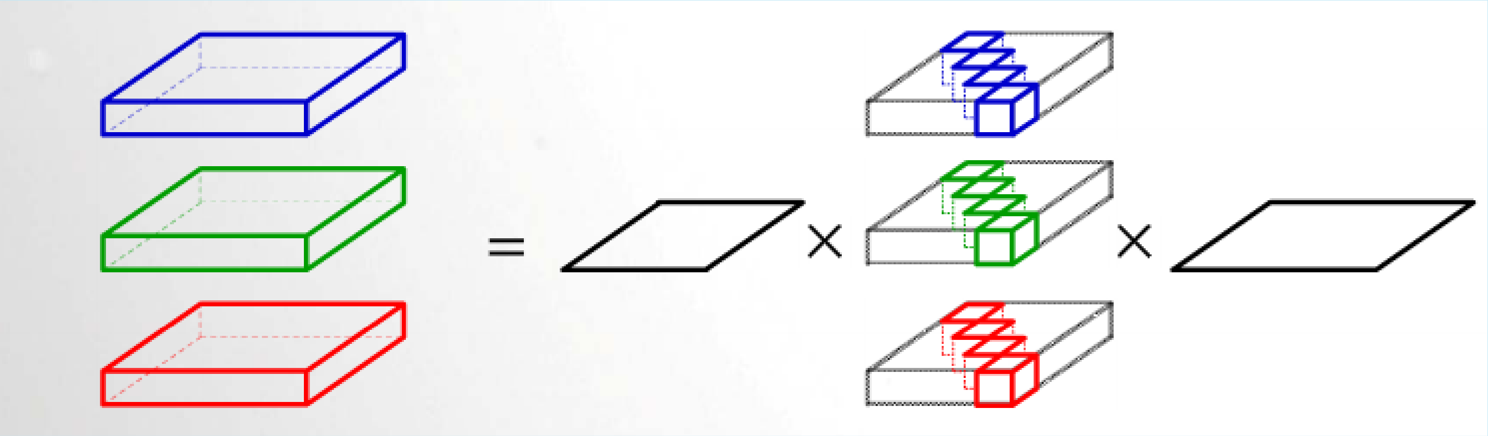


Fig.1 Visualization of models for CPD

1. Introduction

Joint Diagonalization(JD) has been widely applied

- Array processing (X. F. Gong, 2012)
- Tensor decomposition (L. Delathauwer, 2008; X. F. Gong, 2013)
- Speech signal processing (D. T. Pham, 2003)
- Blind source separation (J. F. Cardoso, 1993; A. Mesloub, 2014)

Slicewise form: $\Gamma(:,:,k) = \boldsymbol{A} \times \boldsymbol{D}_k \times \boldsymbol{B}^T$

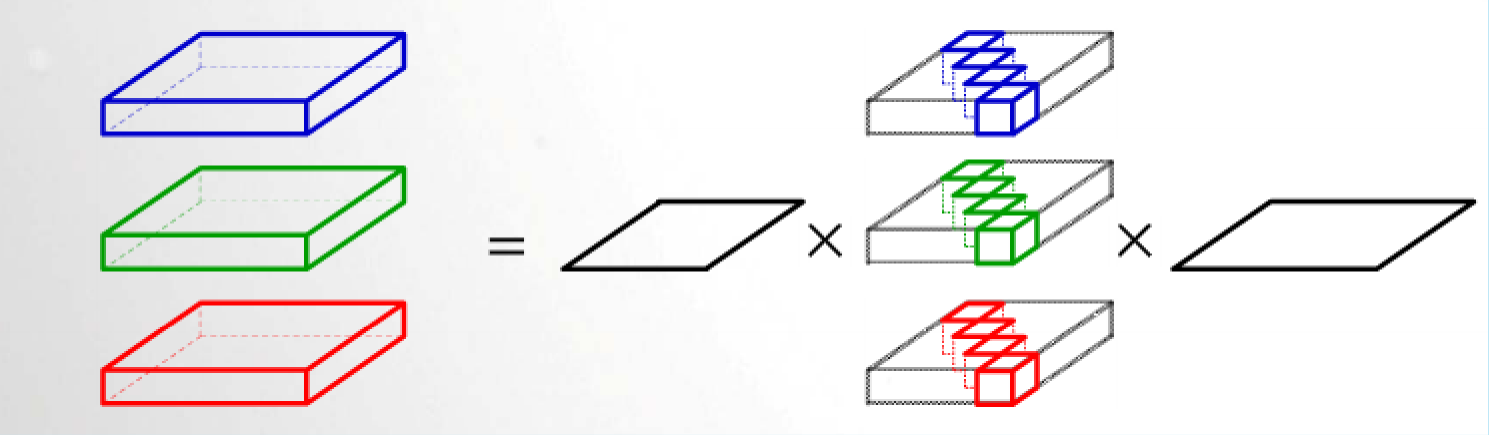


Fig.1 Visualization of models for CPD

1. Introduction

Joint Diagonalization(JD) has been widely applied

- Array processing (X. F. Gong, 2012)
- Tensor decomposition (L. Delathauwer, 2008; X. F. Gong, 2013)
- Speech signal processing (D. T. Pham, 2003)
- Blind source separation (J. F. Cardoso, 1993; A. Mesloub, 2014)

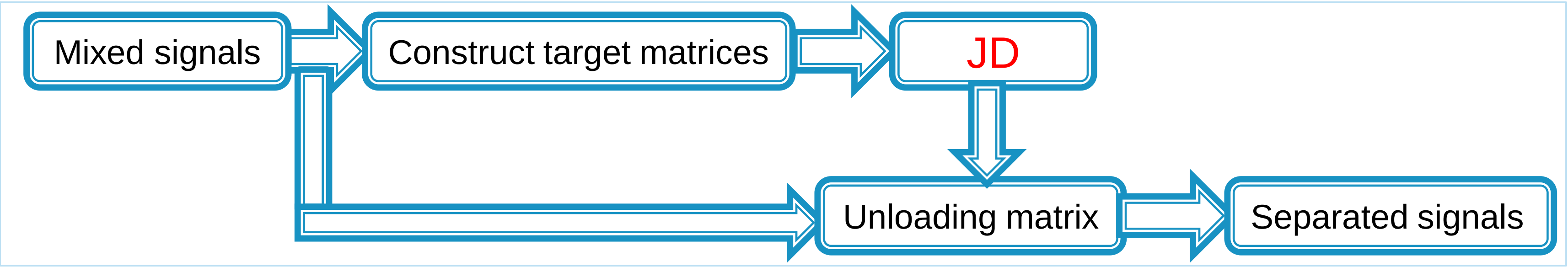
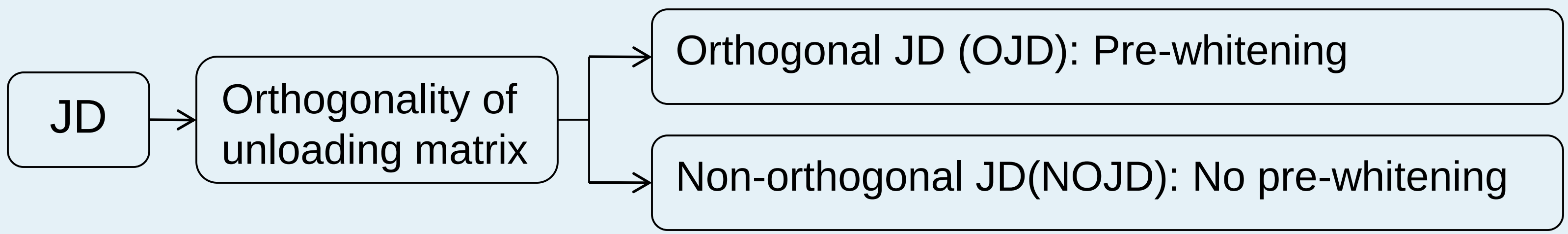


Fig.2 Blind source separation algorithm's block diagram based JD

1. Introduction

Joint diagonalization



- Several criteria applied to JD problems
  - minimization of off-norm
  - weighted least squares
  - information theory

- Optimization strategies
  - some specific optimization like as Newton, Gauss iteration…
  - algebraic strategies like as Jacobi-type or successive rotation

1. Introduction

Joint diagonalization

JD → Orthogonality of unloading matrix
- → Orthogonal JD (OJD): Pre-whitening
- → Non-orthogonal JD(NOJD): No pre-whitening

- Several criteria applied to JD problems
  - minimization of off-norm
  - weighted least squares
  - information theory

- Optimization strategies
  - some specific optimization like as Newton, Gauss iteration…
  - algebraic strategies like as Jacobi-type or successive rotation

1. Introduction

Joint diagonalization

JD → Orthogonality of unloading matrix
→ Orthogonal JD (OJD): Pre-whitening
→ Non-orthogonal JD(NOJD): No pre-whitening

- Several criteria applied to JD problems
  - minimization of off-norm
  - weighted least squares
  - information theory

- Optimization strategies
  - some specific optimization like as Newton, Gauss iteration…
  - algebraic strategies like as Jacobi-type or successive rotation

1. Introduction

Joint diagonalization

JD → Orthogonality of unloading matrix
- → Orthogonal JD (OJD): Pre-whitening
- → Non-orthogonal JD(NOJD): No pre-whitening

- Several criteria applied to JD problems
  - minimization of off-norm
  - weighted least squares
  - information theory

- Optimization strategies
  - some specific optimization like as Newton, Gauss iteration…
  - algebraic strategies like as Jacobi-type or successive rotation

2. The Successive Rotation

Cost function: $\rho = \sum_{k=1}^{K} \mathrm{off}(\boldsymbol{B}\boldsymbol{C}_k\boldsymbol{B}^H)$

- Niter: Maximal sweep number
- Tol: Stopping threshold

while k<Niter && err>Tol

$\boldsymbol{B} = \boldsymbol{I}$

for i = 1:N-1 — Sweep

for j = i +1:N

Obtain $\boldsymbol{T}_{(i,j)}$, $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)}\boldsymbol{C}_{k,old}\boldsymbol{T}_{(i,j)}^H$ — Rotation

$\boldsymbol{B}_{new} = \boldsymbol{T}_{(i,j)}\boldsymbol{B}_{old}$ to minimize ρ

end

end

k = k + 1

end

2. The Successive Rotation

Cost function: $\rho = \sum_{k=1}^{K} \mathrm{off}\,(\boldsymbol{B}\boldsymbol{C}_k\boldsymbol{B}^H)$

- Niter: Maximal sweep number
- Tol: Stopping threshold

while k<Niter && err>Tol

$\boldsymbol{B} = \boldsymbol{I}$

Sweep

for i = 1:N-1

for j = i +1:N

Rotation

Obtain $\boldsymbol{T}_{(i,j)}$, $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)}\boldsymbol{C}_{k,old}\boldsymbol{T}_{(i,j)}^H$

$\boldsymbol{B}_{new} = \boldsymbol{T}_{(i,j)}\boldsymbol{B}_{old}$ to minimize ρ

end

end

k = k + 1

end

2. The Successive Rotation

Cost function: $\rho = \sum_{k=1}^{K} \mathrm{off}(\boldsymbol{B}\boldsymbol{C}_k\boldsymbol{B}^H)$

- Niter: Maximal sweep number
- Tol: Stopping threshold

```
while k<Niter && err>Tol
```

$\boldsymbol{B} = \boldsymbol{I}$

for i = 1:N-1 Sweep

for j = i +1:N

Obtain $\boldsymbol{T}_{(i,j)}$, $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)}\boldsymbol{C}_{k,old}\boldsymbol{T}_{(i,j)}^H$ Rotation

$\boldsymbol{B}_{new} = \boldsymbol{T}_{(i,j)}\boldsymbol{B}_{old}$ to minimize ρ

end

end

k = k + 1

end

2. The Successive Rotation

Cost function: $\rho = \sum_{k=1}^{K} \mathrm{off}\,(\boldsymbol{B}\boldsymbol{C}_k\boldsymbol{B}^H)$

while k<Niter && err>Tol

$\boldsymbol{B} = \boldsymbol{I}$

for i = 1:N-1 — Sweep

for j = i +1:N

Obtain $\boldsymbol{T}_{(i,j)}$, $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)}\boldsymbol{C}_{k,old}\boldsymbol{T}_{(i,j)}^H$ — Rotation

$\boldsymbol{B}_{new} = \boldsymbol{T}_{(i,j)}\boldsymbol{B}_{old}$ to minimize ρ

end

end

k = k + 1

end

- Niter: Maximal sweep number
- Tol: Stopping threshold

2. The Successive Rotation

Cost function: $\rho = \sum_{k=1}^{K} \mathrm{off}(\boldsymbol{B}\boldsymbol{C}_k\boldsymbol{B}^H)$

```
while k<Niter && err>Tol
        B = I
        for i = 1:N-1                                   Sweep
            for j = i +1:N
                Obtain T_(i,j) , C_k,new = T_(i,j) C_k,old T^H_(i,j)     Rotation
                B_new = T_(i,j) B_old   to minimize ρ
            end
        end
        k = k + 1
end
```

- Niter: Maximal sweep number
- Tol: Stopping threshold

where $\boldsymbol{B} = \boldsymbol{I}$, $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)}\boldsymbol{C}_{k,old}\boldsymbol{T}_{(i,j)}^H$, $\boldsymbol{B}_{new} = \boldsymbol{T}_{(i,j)}\boldsymbol{B}_{old}$ to minimize ρ

2. The Successive Rotation

Cost function: $\rho = \sum_{k=1}^{K} \mathrm{off}(\boldsymbol{B}\boldsymbol{C}_k\boldsymbol{B}^H)$

while k<Niter && err>Tol

- Niter: Maximal sweep number
- Tol: Stopping threshold

$\boldsymbol{B} = \boldsymbol{I}$

for i = 1:N-1 — Sweep

for j = i +1:N

Obtain $\boldsymbol{T}_{(i,j)}$, $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)}\boldsymbol{C}_{k,old}\boldsymbol{T}_{(i,j)}^H$ — Rotation

- Problem: The time consumed in the above process is in quadratic relationship with the dimensionality of target matrices N
- Solution: So we consider the parallelization of these successive rotations to address this problem.

end

2. The Successive Rotation

Rotation $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)} \boldsymbol{C}_{k,old} \boldsymbol{T}_{(i,j)}^{H}$

In general, the elementary rotation matrix:

$$\boldsymbol{T}_{(i,j)} = \begin{bmatrix} \boldsymbol{I} & & & & \\ & \alpha_{ii} & & \alpha_{ij} & \\ & & \boldsymbol{I} & & \\ & \alpha_{ji} & & \alpha_{jj} & \\ & & & & \boldsymbol{I} \end{bmatrix}$$

$\boldsymbol{T}_{(i,j)}$ only impacts the ith and jth row and column of $\boldsymbol{C}_{k,old}$

- JADE: Joint Approximate Diagonalization of Eigenmatrices by J.-F. Cardoso,1993
- CJDi: Complex Joint Diagonalization by A. Mesloub, 2014

2. The Successive Rotation

Rotation $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)} \boldsymbol{C}_{k,old} \boldsymbol{T}_{(i,j)}^{H}$

In general, the elementary rotation matrix:

$$\boldsymbol{T}_{(i,j)} = \begin{bmatrix} \boldsymbol{I} & & & & \\ & \alpha_{ii} & & \alpha_{ij} & \\ & & \boldsymbol{I} & & \\ & \alpha_{ji} & & \alpha_{jj} & \\ & & & & \boldsymbol{I} \end{bmatrix}$$

$\boldsymbol{T}_{(i,j)}$ only impacts the ith and jth row and column of $\boldsymbol{C}_{k,old}$

- JADE: Joint Approximate Diagonalization of Eigenmatrices by J.-F. Cardoso,1993
- CJDi: Complex Joint Diagonalization by A. Mesloub, 2014

2. The Successive Rotation

Rotation $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)} \boldsymbol{C}_{k,old} \boldsymbol{T}_{(i,j)}^{H}$

In general, the elementary rotation matrix:

$$\boldsymbol{T}_{(i,j)} = \begin{bmatrix} \boldsymbol{I} & & & & \\ & \alpha_{ii} & & \alpha_{ij} & \\ & & \boldsymbol{I} & & \\ & \alpha_{ji} & & \alpha_{jj} & \\ & & & & \boldsymbol{I} \end{bmatrix}$$

$\boldsymbol{T}_{(i,j)}$ only impacts the ith and jth row and column of $\boldsymbol{C}_{k,old}$

- JADE: Joint Approximate Diagonalization of Eigenmatrices by J.-F. Cardoso,1993
- CJDi: Complex Joint Diagonalization by A. Mesloub, 2014

2. The Successive Rotation

Rotation $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)} \boldsymbol{C}_{k,old} \boldsymbol{T}_{(i,j)}^{H}$

But in LUCJD, the elementary rotation matrix:

$$\boldsymbol{T}_{(i,j)} = \begin{bmatrix} \boldsymbol{I} & & & & \\ & 1 & & & \\ & & \boldsymbol{I} & & \\ & \alpha_{ij} & & 1 & \\ & & & & \boldsymbol{I} \end{bmatrix}$$

$\boldsymbol{T}_{(i,j)}$ only impacts the ith row and column of $\boldsymbol{C}_{k,old}$

LUCJD：

- LU decomposition for Complex Joint Diagonalization by K. Wang, LVA/ICA2012
- To solve the complex non-orthogonal joint diagonalization problem via LU decomposition and successive rotation

2. The Successive Rotation

Rotation $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)} \boldsymbol{C}_{k,old} \boldsymbol{T}_{(i,j)}^{H}$

But in LUCJD, the elementary rotation matrix:

$$\boldsymbol{T}_{(i,j)} = \begin{bmatrix} \boldsymbol{I} & & & & \\ & 1 & & & \\ & & \boldsymbol{I} & & \\ & \alpha_{ij} & & 1 & \\ & & & & \boldsymbol{I} \end{bmatrix}$$

$\boldsymbol{T}_{(i,j)}$ only impacts the ith row and column of $\boldsymbol{C}_{k,old}$

***LUCJD*：**

- LU decomposition for Complex Joint Diagonalization by K. Wang, LVA/ICA2012
- To solve the complex non-orthogonal joint diagonalization problem via LU decomposition and successive rotation

2. The Successive Rotation

Rotation $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)} \boldsymbol{C}_{k,old} \boldsymbol{T}_{(i,j)}^{H}$

But in LUCJD, the elementary rotation matrix:

$$\boldsymbol{T}_{(i,j)} = \begin{bmatrix} \boldsymbol{I} & & & & \\ & 1 & & & \\ & & \boldsymbol{I} & & \\ & \alpha_{ij} & & 1 & \\ & & & & \boldsymbol{I} \end{bmatrix}$$

$\boldsymbol{T}_{(i,j)}$ only impacts the ith row and column of $\boldsymbol{C}_{k,old}$

LUCJD：

- LU decomposition for Complex Joint Diagonalization by K. Wang, LVA/ICA2012
- To solve the complex non-orthogonal joint diagonalization problem via LU decomposition and successive rotation

2. The Successive Rotation

Rotation $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)} \boldsymbol{C}_{k,old} \boldsymbol{T}_{(i,j)}^{H}$

But in LUCJD, the elementary rotation matrix:

$$\boldsymbol{T}_{(i,j)} = \begin{bmatrix} \boldsymbol{I} & & & & \\ & 1 & & & \\ & & \boldsymbol{I} & & \\ & \alpha_{ij} & & 1 & \\ & & & & \boldsymbol{I} \end{bmatrix}$$

$\boldsymbol{T}_{(i,j)}$ only impacts the ith row and column of $\boldsymbol{C}_{k,old}$

LUCJD:

- LU decomposition for Complex Joint Diagonalization by K. Wang, LVA/ICA2012
- To solve the complex non-orthogonal joint diagonalization problem via LU decomposition and successive rotation

2. The Successive Rotation

Rotation $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)} \boldsymbol{C}_{k,old} \boldsymbol{T}_{(i,j)}^{H}$

But in LUCJD, the elementary rotation matrix:

$$\boldsymbol{T}_{(i,j)} = \begin{bmatrix} \boldsymbol{I} & & & & \\ & 1 & & & \\ & & \boldsymbol{I} & & \\ & \alpha_{ij} & & 1 & \\ & & & & \boldsymbol{I} \end{bmatrix}$$

$\boldsymbol{T}_{(i,j)}$ only impacts the ith row and column of $\boldsymbol{C}_{k,old}$

LUCJD:

- LU decomposition for Complex Joint Diagonalization by K. Wang, LVA/ICA2012
- To solve the complex non-orthogonal joint diagonalization problem via LU decomposition and successive rotation

2. The Successive Rotation

Rotation $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)} \boldsymbol{C}_{k,old} \boldsymbol{T}_{(i,j)}^{H}$

But in LUCJD, the elementary rotation matrix:

$$\boldsymbol{T}_{(i,j)} = \begin{bmatrix} \boldsymbol{I} & & & & \\ & 1 & & & \\ & & \boldsymbol{I} & & \\ & \alpha_{ij} & & 1 & \\ & & & & \boldsymbol{I} \end{bmatrix}$$

$\boldsymbol{T}_{(i,j)}$ only impacts the ith row and column of $\boldsymbol{C}_{k,old}$

In this paper, we consider the parellelization of LUCJD

- To solve the complex non-orthogonal joint diagonalization problem via LU decomposition and successive rotation

3. The proposed algorithm

Parallelization of LUCJD

- The key to an efficient parallelization is the segmentation of entire index pairs into multiple subsets
- Then those optimal elementary rotations could be calculated at one shot
- Noting that the index pairs in one subset are non-conflicting

We develop the following *3* parallelization schemes:

- *Row-wise parallelization*
- *Column-wise parallelization*
- *Diagonal-wise parallelization*

3. The proposed algorithm

Parallelization of LUCJD

- The key to an efficient parallelization is the segmentation of entire index pairs into multiple subsets
- Then those optimal elementary rotations could be calculated at one shot
- Noting that the index pairs in one subset are non-conflicting

We develop the following *3* parallelization schemes:

- *Row-wise parallelization*
- *Column-wise parallelization*
- *Diagonal-wise parallelization*

3. The proposed algorithm

Parallelization of LUCJD

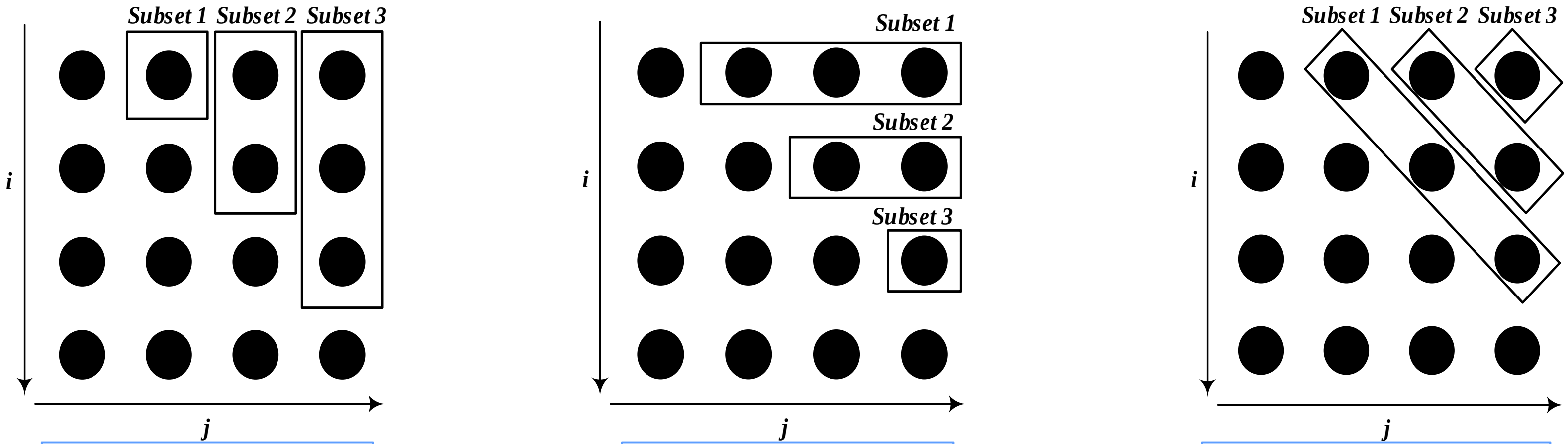
- The key to an efficient parallelization is the segmentation of entire index pairs into multiple subsets
- Then those optimal elementary rotations could be calculated at one shot
- Noting that the index pairs in one subset are non-conflicting

We develop the following *3* parallelization schemes:

- *Row-wise parallelization*
- *Column-wise parallelization*
- *Diagonal-wise parallelization*

3. The proposed algorithm

Rotation $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)}\boldsymbol{C}_{k,old}\boldsymbol{T}^{H}_{(i,j)}$



(a) Column-wise (b) Row-wise (c) Diagonal-wise

Fig.3 Subset segmentation for parallelization of LUCJD

3. The proposed algorithm

Rotation $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)} \boldsymbol{C}_{k,old} \boldsymbol{T}_{(i,j)}^{H}$

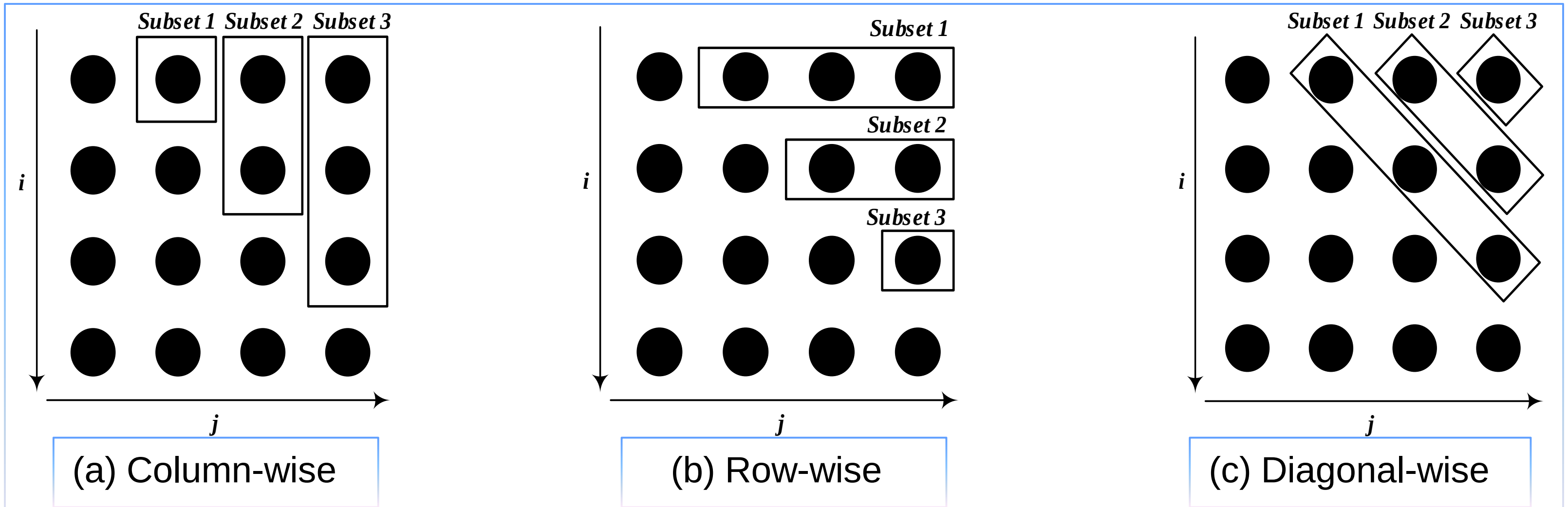


Fig.3 Subset segmentation for parallelization of LUCJD

3. The proposed algorithm

Rotation $$\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)}\boldsymbol{C}_{k,old}\boldsymbol{T}_{(i,j)}^{H}$$

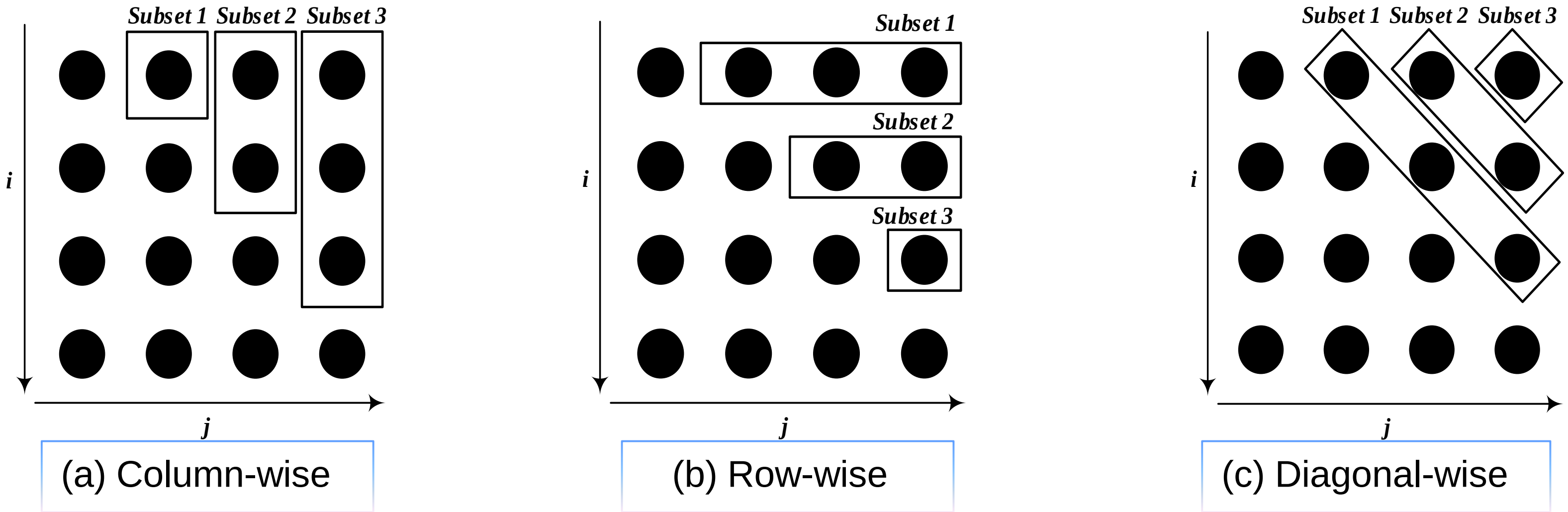
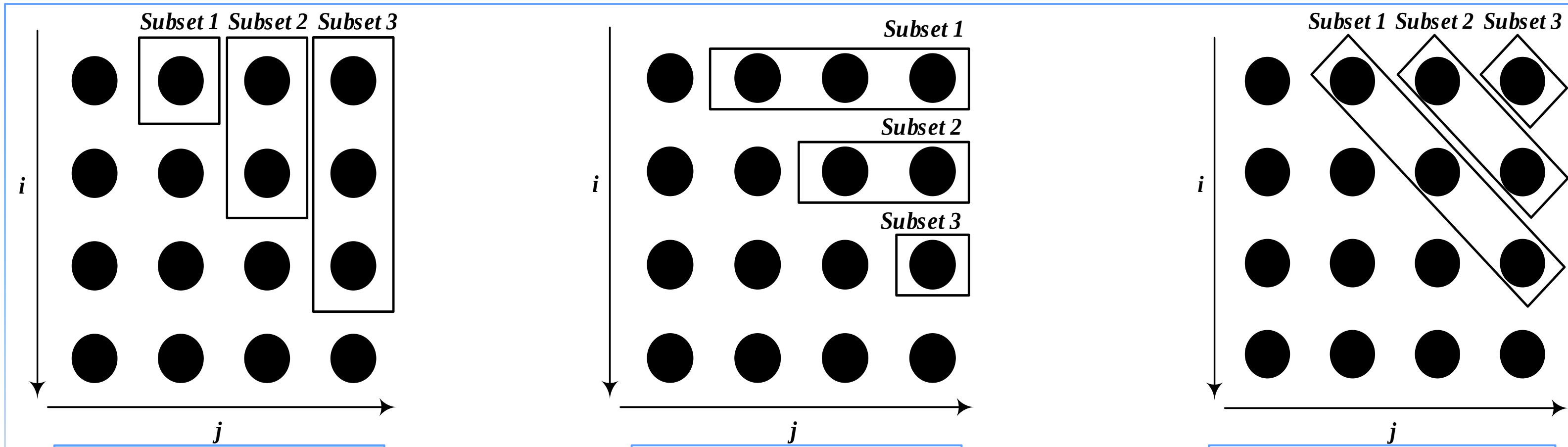


Fig.3 Subset segmentation for parallelization of LUCJD

3. The proposed algorithm

Rotation $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)} \boldsymbol{C}_{k,old} \boldsymbol{T}_{(i,j)}^{H}$



(a) Column-wise (b) Row-wise (c) Diagonal-wise

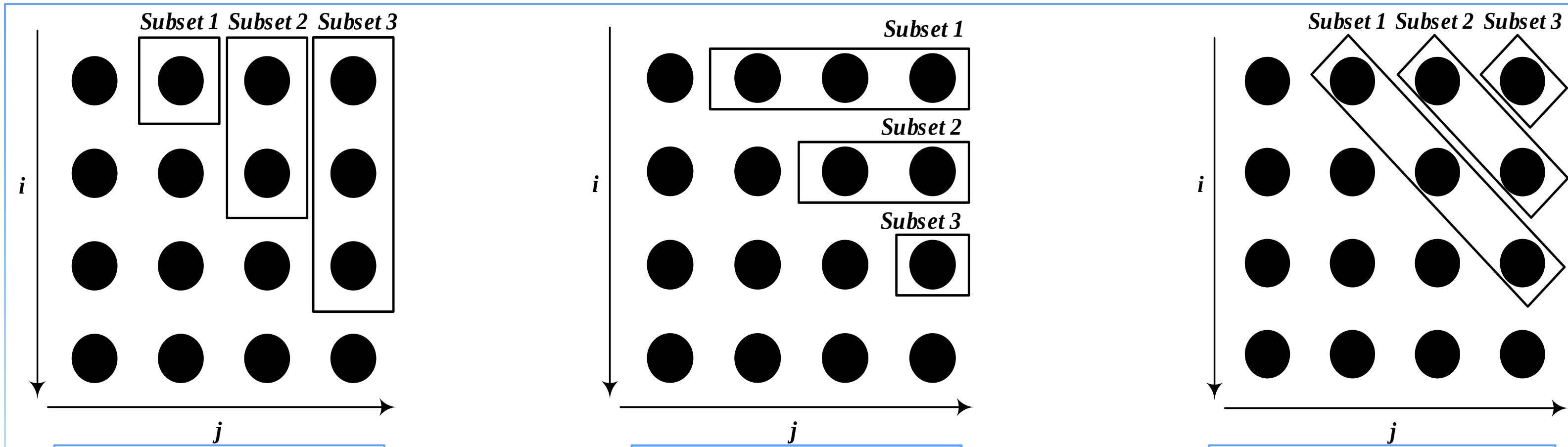
Fig.3 Subset segmentation for parallelization of LUCJD

$$\Omega_j^C \triangleq \{(i,j) \mid i = 1,2,\ldots, j-1\}$$

$$j = 2,3,\ldots,N$$

3. The proposed algorithm

Rotation $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)}\boldsymbol{C}_{k,old}\boldsymbol{T}_{(i,j)}^{H}$



(a) Column-wise

(b) Row-wise

(c) Diagonal-wise

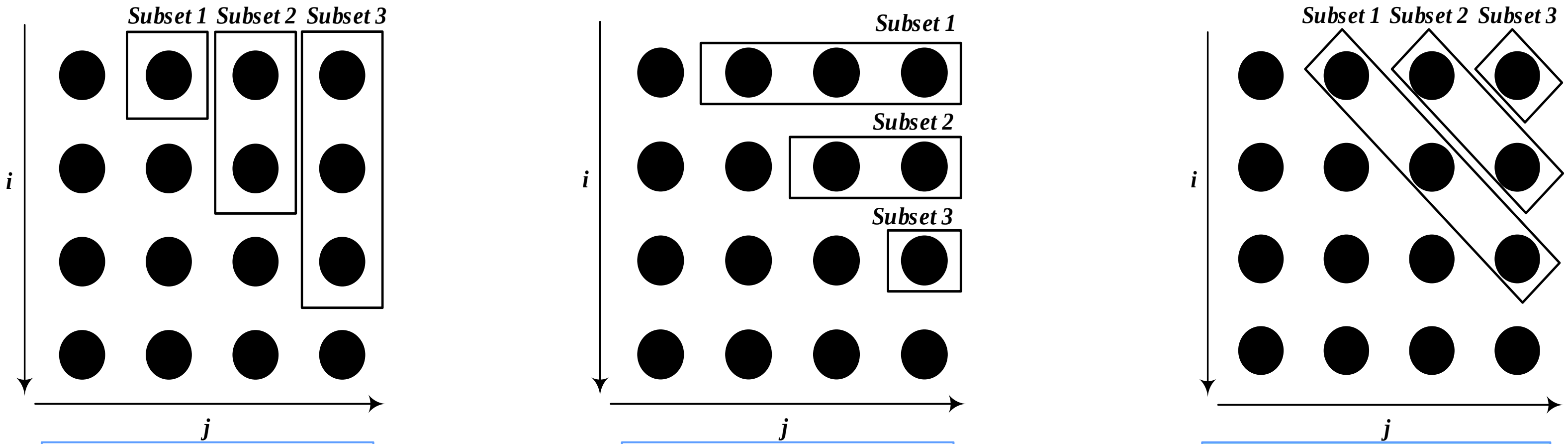
Fig.3 Subset segmentation for parallelization of LUCJD

$$\Omega_j^C \triangleq \{(i,j) \mid i = 1,2,\ldots,j-1\}$$
$$j = 2,3,\ldots,N$$

$$\Omega_i^R \triangleq \{(i,j) \mid j = i+1,\ldots,N\}$$
$$i = 1,2,\ldots,N-1$$

3. The proposed algorithm

Rotation $\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,j)} \boldsymbol{C}_{k,old} \boldsymbol{T}_{(i,j)}^{H}$



(a) Column-wise

(b) Row-wise

(c) Diagonal-wise

Fig.3 Subset segmentation for parallelization of LUCJD

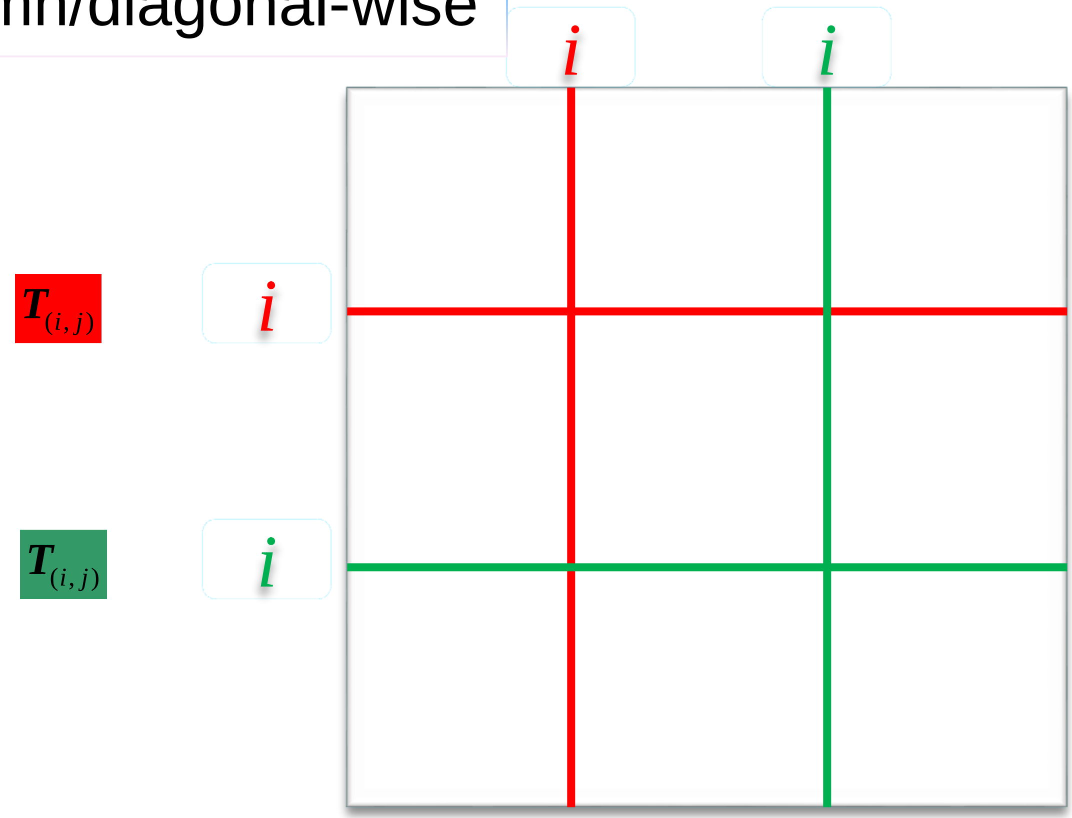
$$\Omega_j^C \triangleq \{(i,j) \mid i = 1,2,\ldots,j-1\}$$
$$j = 2,3,\ldots,N$$

$$\Omega_i^R \triangleq \{(i,j) \mid j = i+1,\ldots,N\}$$
$$i = 1,2,\ldots,N-1$$

$$\Omega_j^D \triangleq \{(i,i+j) \mid i = 1,2,\ldots,N-j\}$$
$$j = 1,2,\ldots N-1$$

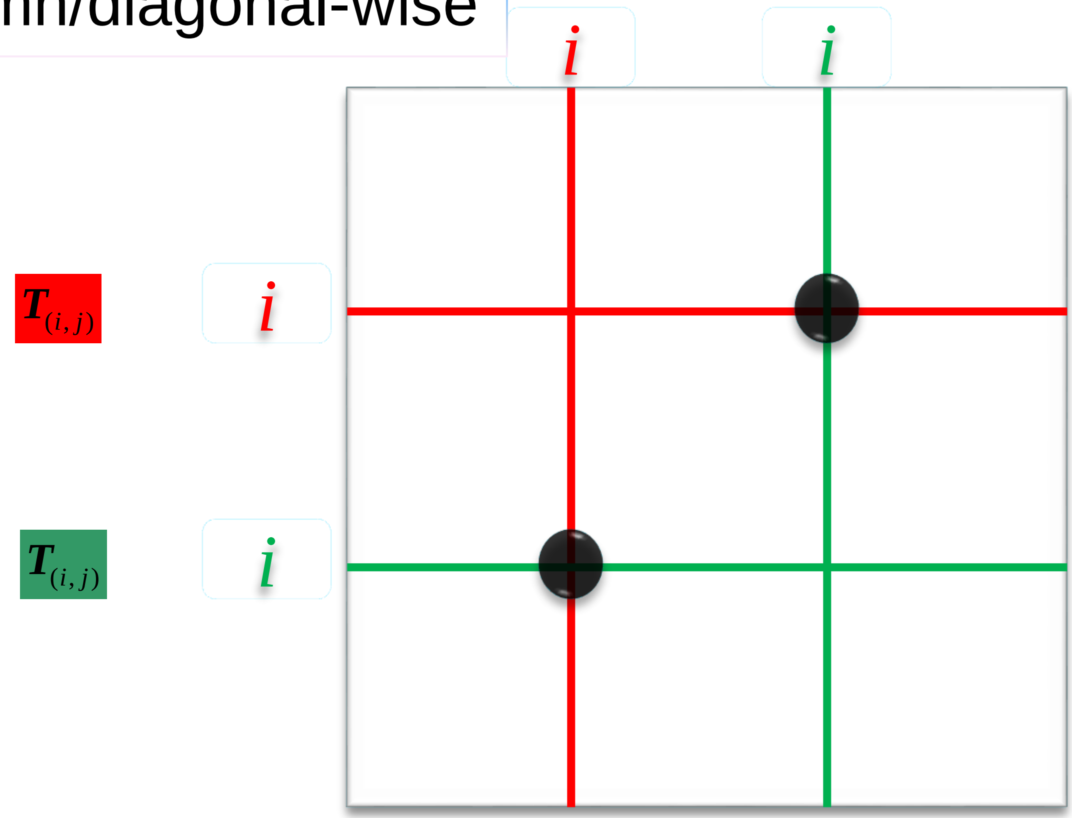
3. The proposed algorithm

Column/diagonal-wise



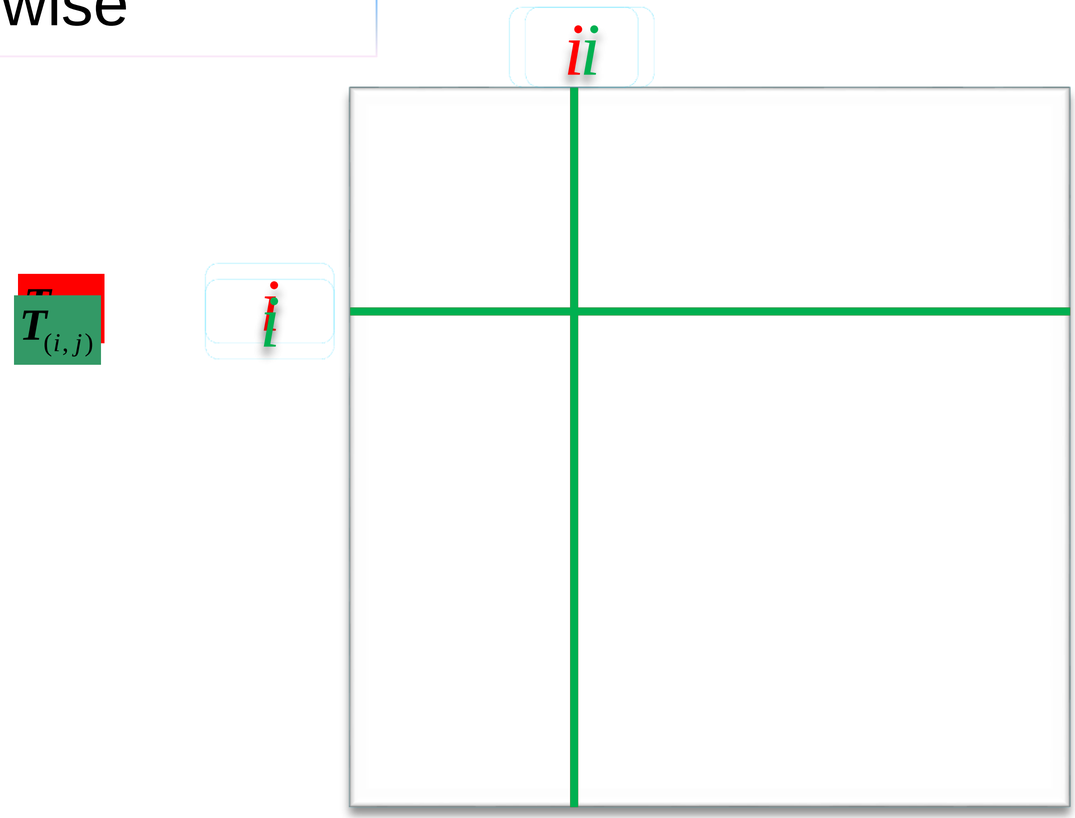
3. The proposed algorithm

Column/diagonal-wise



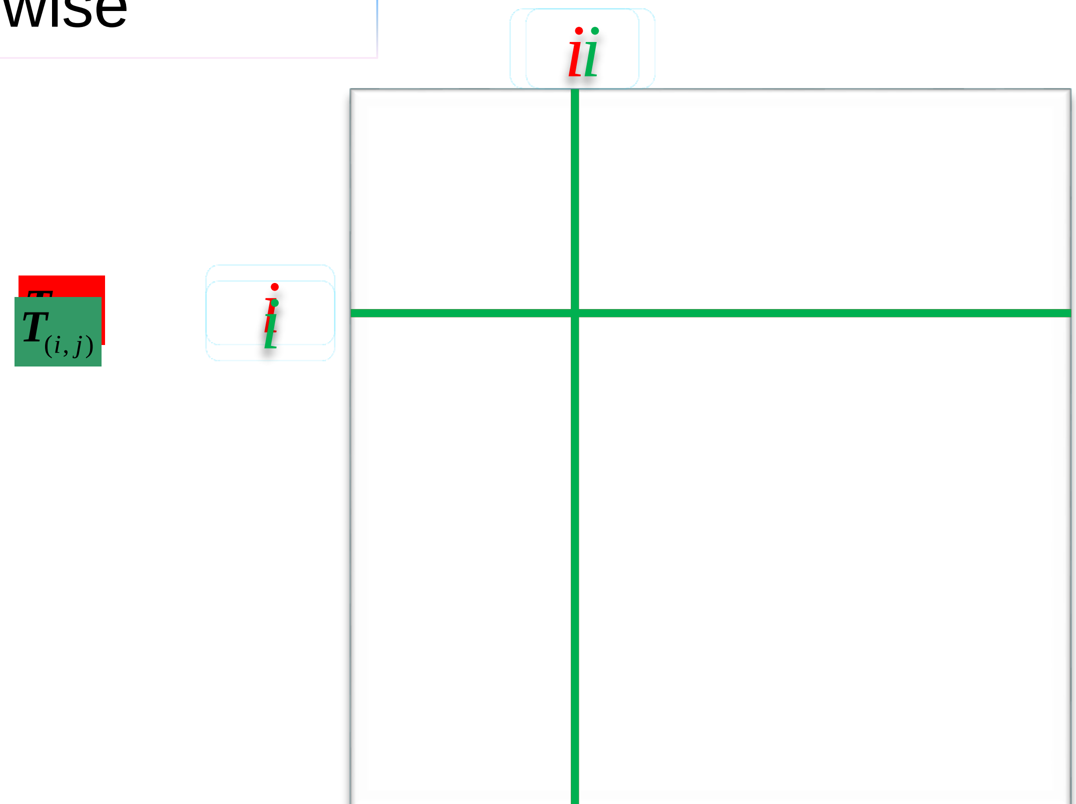
3. The proposed algorithm

Row-wise



3. The proposed algorithm

Row-wise



The subset elements of row-wise scheme are not non-conflicting comparing with other two schemes which might result in performance loss,
this will be shown later.

3. The proposed algorithm

Table 1 summarization of row-wise of LUCJD

- Niter: Maximal sweep number
- Tol: Stopping threshold

while k<Niter && err>Tol

$\boldsymbol{B} = \boldsymbol{I}$

for i = 1:N-1 Sweep

Obtain $\boldsymbol{T}_{(i,i+1:N)}$, Rotation

$\boldsymbol{C}_{k,new} = \boldsymbol{T}_{(i,i+1:N)} \boldsymbol{C}_{k,old} \boldsymbol{T}_{(i,i+1:N)}^H \quad \boldsymbol{B}_{new} = \boldsymbol{T}_{(i,i+1:N)} \boldsymbol{B}_{old}$

to minimize ρ

end

k = k + 1

end

4. Simulation results

- The target matrices are generated as: $\boldsymbol{C}_k = P_s \boldsymbol{A}\boldsymbol{D}_k \boldsymbol{A}^H + P_n \boldsymbol{N}_k$
- Signal-to-noise ratio(SNR): $SNR = 10\log_{10}(P_s / P_n)$
- Performance index(PI): to evaluate the JD quality

$$PI(G) = \frac{1}{2N(N-1)}\left[\sum_{i=1}^{N}\left(\sum_{j=1}^{N}\frac{|g_{ij}|}{\max_k |g_{ik}|}-1\right)+\sum_{i=1}^{N}\left(\sum_{j=1}^{N}\frac{|g_{ji}|}{\max_k |g_{ki}|}-1\right)\right]$$

We also consider the TPO parallelized strategy

(Tournament Player's Ordering, by A. Holobar, *EUROCON2003*)

- *Simulation 1-Convergence Pattern*
- *Simulation 2-Execution Time*
- *Simulation 3-Joint Diagonalization Quality*

4. Simulation results

- The target matrices are generated as: $\boldsymbol{C}_k = P_s \boldsymbol{A}\boldsymbol{D}_k \boldsymbol{A}^H + P_n \boldsymbol{N}_k$
- Signal-to-noise ratio(SNR): $SNR = 10\log_{10}(P_s / P_n)$
- Performance index(PI): to evaluate the JD quality

$$PI(G) = \frac{1}{2N(N-1)}\left[\sum_{i=1}^{N}\left(\sum_{j=1}^{N}\frac{|g_{ij}|}{\max_k |g_{ik}|}-1\right)+\sum_{i=1}^{N}\left(\sum_{j=1}^{N}\frac{|g_{ji}|}{\max_k |g_{ki}|}-1\right)\right]$$

We also consider the TPO parallelized strategy

(Tournament Player's Ordering, by A. Holobar, *EUROCON2003*)

- *Simulation 1-Convergence Pattern*
- *Simulation 2-Execution Time*
- *Simulation 3-Joint Diagonalization Quality*

4. Simulation results

- The target matrices are generated as: $\boldsymbol{C}_k = P_s \boldsymbol{A}\boldsymbol{D}_k \boldsymbol{A}^H + P_n \boldsymbol{N}_k$
- Signal-to-noise ratio(SNR): $SNR = 10\log_{10}(P_s / P_n)$
- Performance index(PI): to evaluate the JD quality

$$PI(G) = \frac{1}{2N(N-1)}\left[\sum_{i=1}^{N}\left(\sum_{j=1}^{N}\frac{|g_{ij}|}{\max_k |g_{ik}|}-1\right)+\sum_{i=1}^{N}\left(\sum_{j=1}^{N}\frac{|g_{ji}|}{\max_k |g_{ki}|}-1\right)\right]$$

We also consider the TPO parallelized strategy

(Tournament Player's Ordering, by A. Holobar, *EUROCON2003*)

- *Simulation 1-Convergence Pattern*
- *Simulation 2-Execution Time*
- *Simulation 3-Joint Diagonalization Quality*

4. Simulation results

- The target matrices are generated as: $\boldsymbol{C}_k = P_s \boldsymbol{A}\boldsymbol{D}_k \boldsymbol{A}^H + P_n \boldsymbol{N}_k$
- Signal-to-noise ratio(SNR): $SNR = 10\log_{10}(P_s / P_n)$
- Performance index(PI): to evaluate the JD quality

$$PI(G) = \frac{1}{2N(N-1)}\left[\sum_{i=1}^{N}\left(\sum_{j=1}^{N}\frac{|g_{ij}|}{\max_k |g_{ik}|}-1\right)+\sum_{i=1}^{N}\left(\sum_{j=1}^{N}\frac{|g_{ji}|}{\max_k |g_{ki}|}-1\right)\right]$$

We also consider the TPO parallelized strategy
(Tournament Player's Ordering, by A. Holobar, *EUROCON2003*)

- *Simulation 1-Convergence Pattern*
- *Simulation 2-Execution Time*
- *Simulation 3-Joint Diagonalization Quality*

4. Simulation Results

Simulation 1-Convergence Pattern

- Matrix numbers: K = 10
- Matrix dimensionality: N = 10
- 5 independent runs
- Signal to Noise Ratio: SNR = 20dB

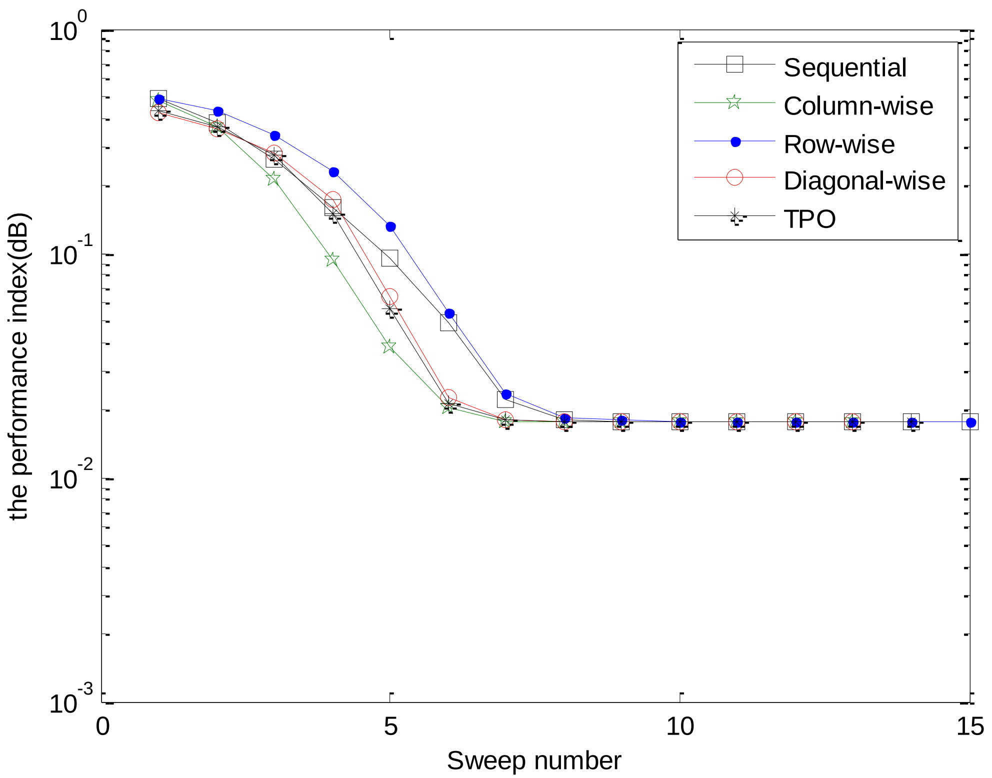


Fig.4 PI versus number of iterations

4. Simulation Results

Simulation 1-Convergence Pattern

- Matrix numbers: K = 10
- Matrix dimensionality: N = 10
- 5 independent runs
- Signal to Noise Ratio: SNR = 20dB

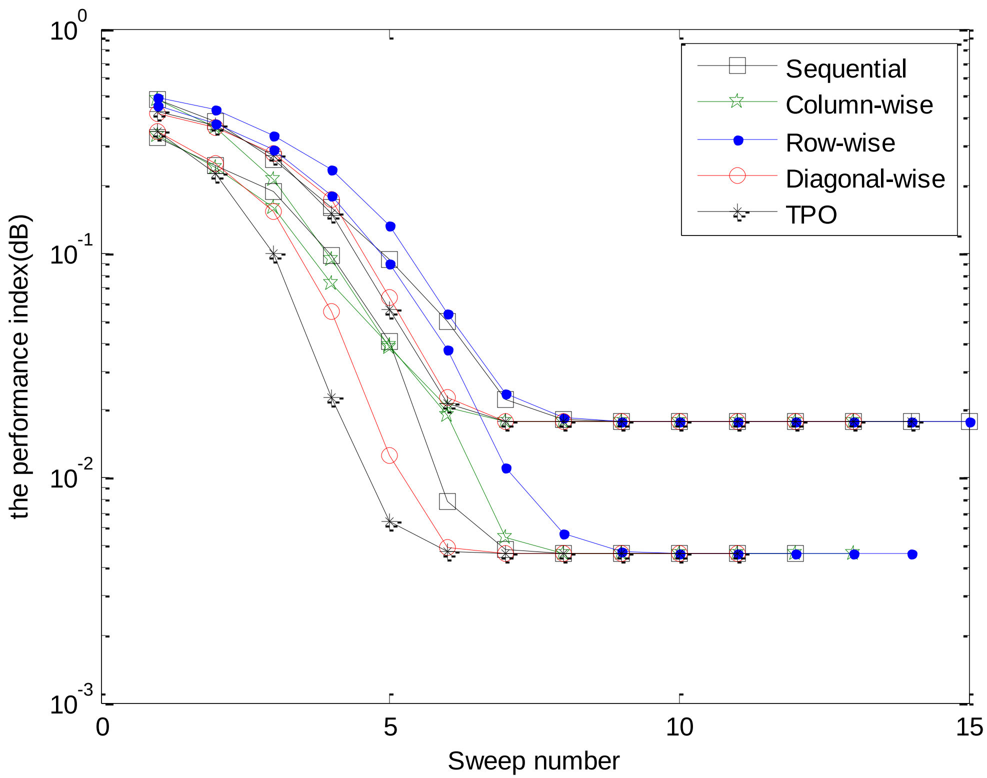


Fig.4 PI versus number of iterations

4. Simulation Results

Simulation 1-Convergence Pattern

- Matrix numbers: K = 10
- Matrix dimensionality: N = 10
- 5 independent runs
- Signal to Noise Ratio: SNR = 20dB

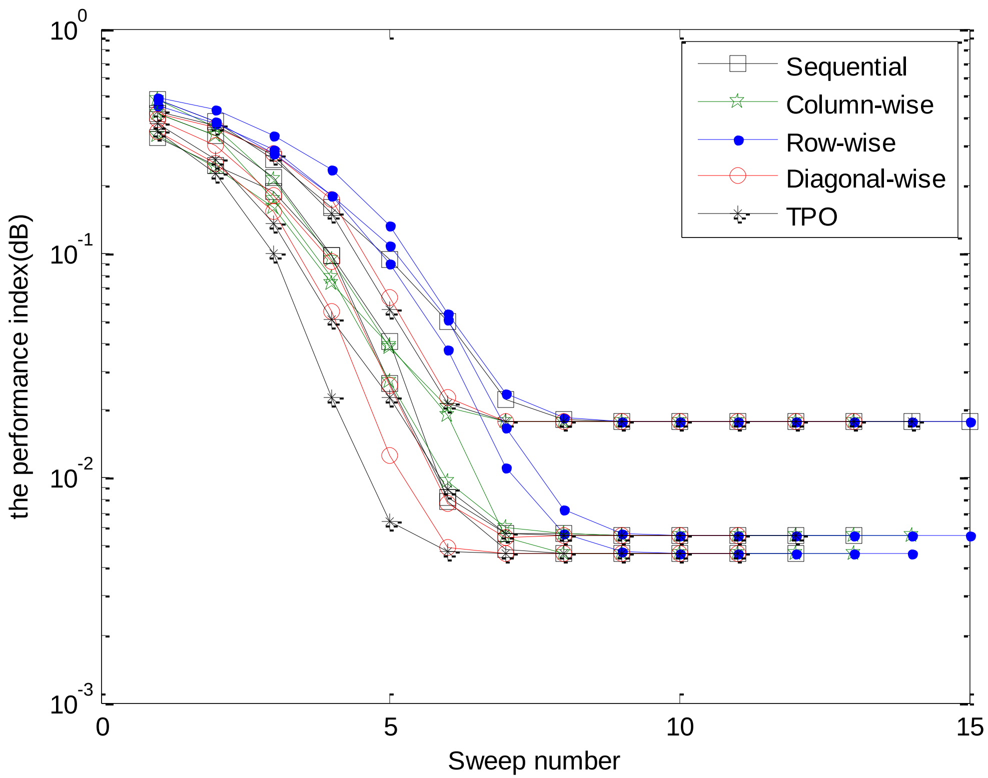


Fig.4 PI versus number of iterations

4. Simulation Results

Simulation 1-Convergence Pattern

- Matrix numbers: K = 10
- Matrix dimensionality: N = 10
- 5 independent runs
- Signal to Noise Ratio: SNR = 20dB

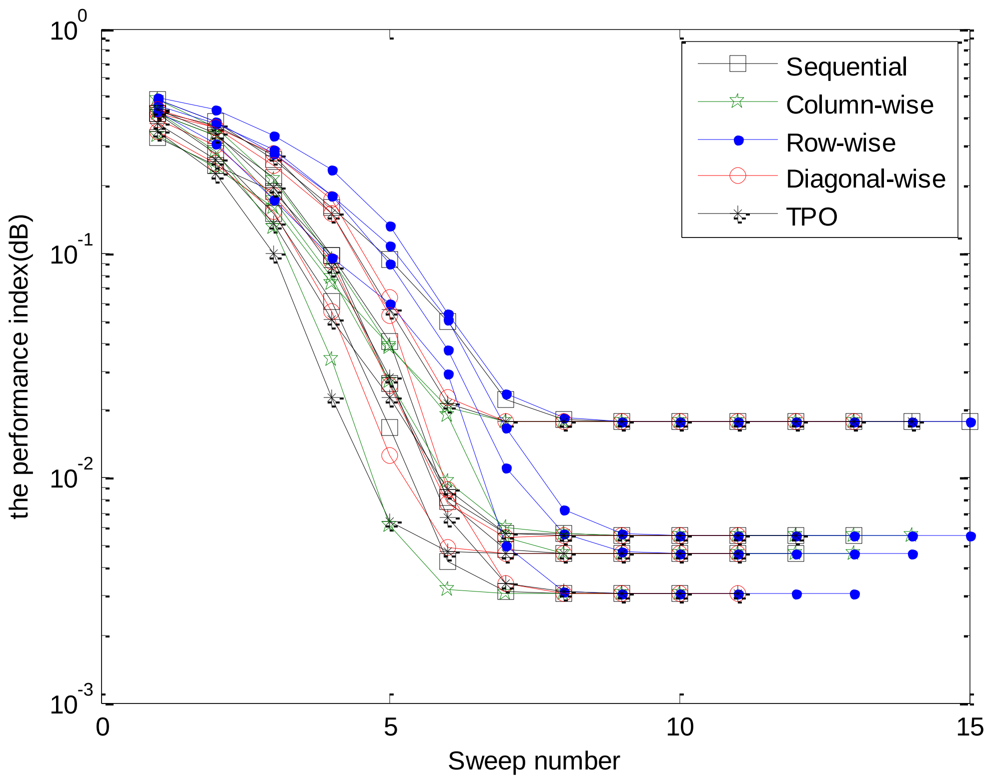


Fig.4 PI versus number of iterations

4. Simulation Results

Simulation 1-Convergence Pattern

- Matrix numbers: K = 10
- Matrix dimensionality: N = 10
- 5 independent runs
- Signal to Noise Ratio: SNR = 20dB

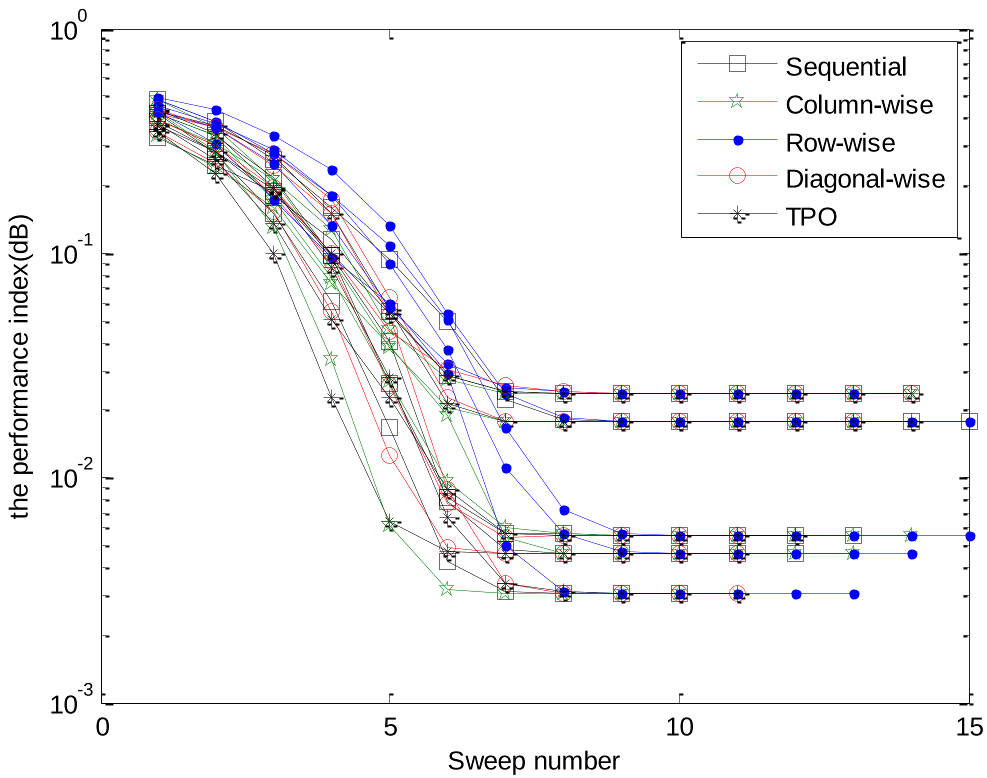


Fig.4 PI versus number of iterations

- Similar iteration number
- Little impact on the convergence

4. Simulation Results

Simulation 1-Convergence Pattern

- Matrix numbers: K = 10
- Matrix dimensionality: N = 10
- 5 independent runs
- Noise free

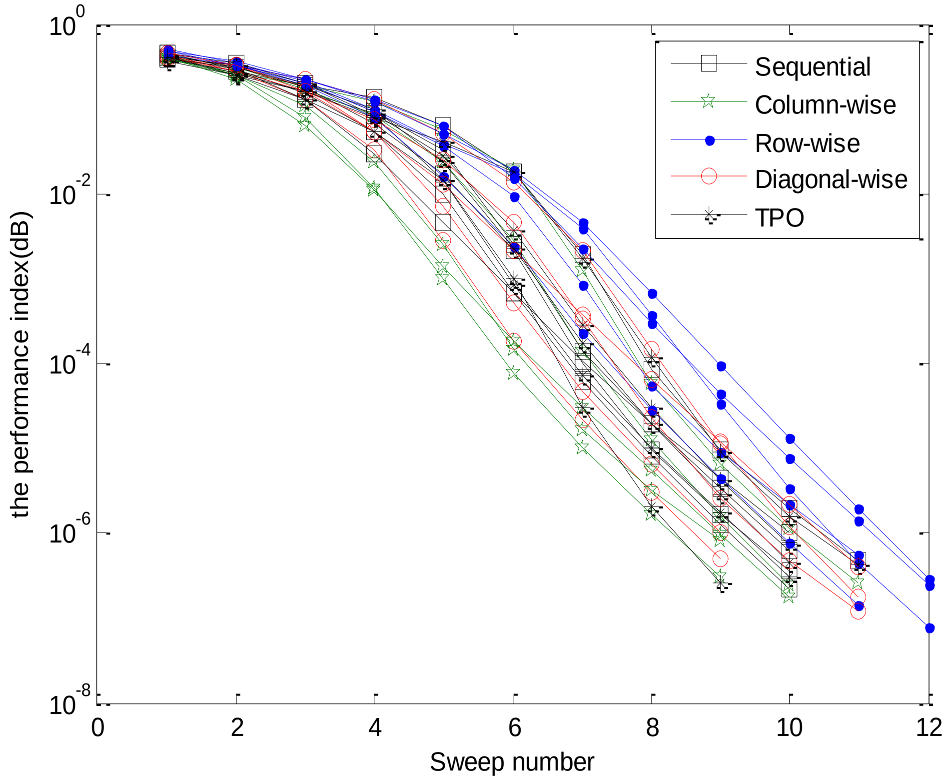


Fig.5 PI versus number of iterations

4. Simulation Results

Simulation 2-Execution Time

- Matrix numbers: K = 20
- Signal to Noise Ratio: SNR = 20dB
- 100 Monte Carlo runs

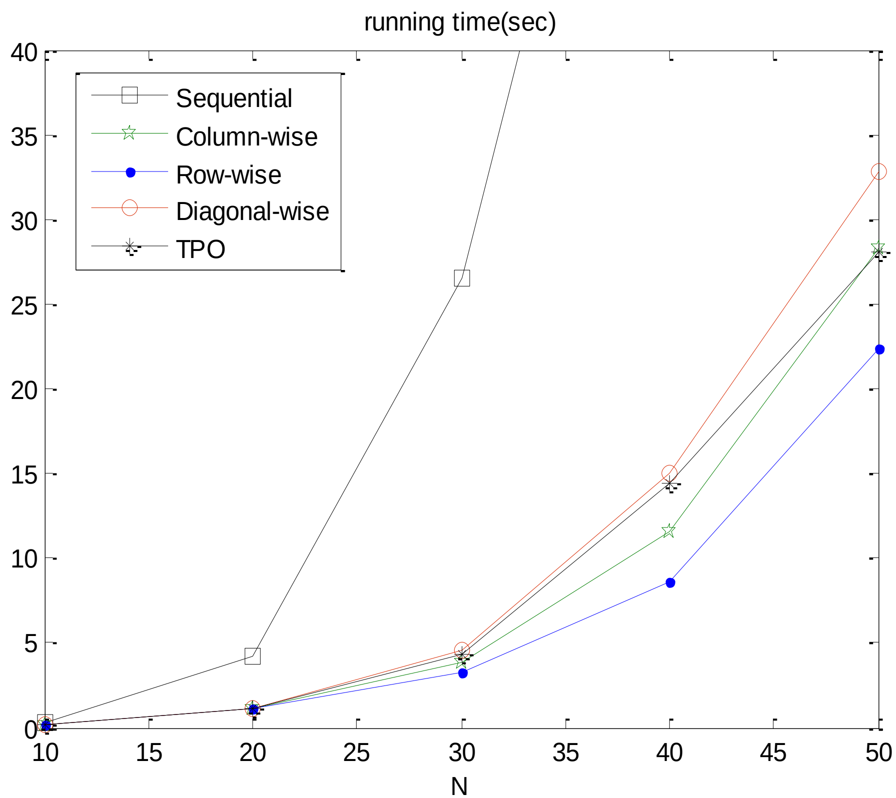


Fig.6 Average running time versus dimensionality

- Largely reduce execution time
- More significant as N increases
- Row>Column>TPO>Diagonal >sequential

4. Simulation Results

Simulation 3-Joint Diagonalization Quality

- Matrix numbers: K = 20
- Matrix dimensionality: N = 30
- 100 Monte Carlo runs

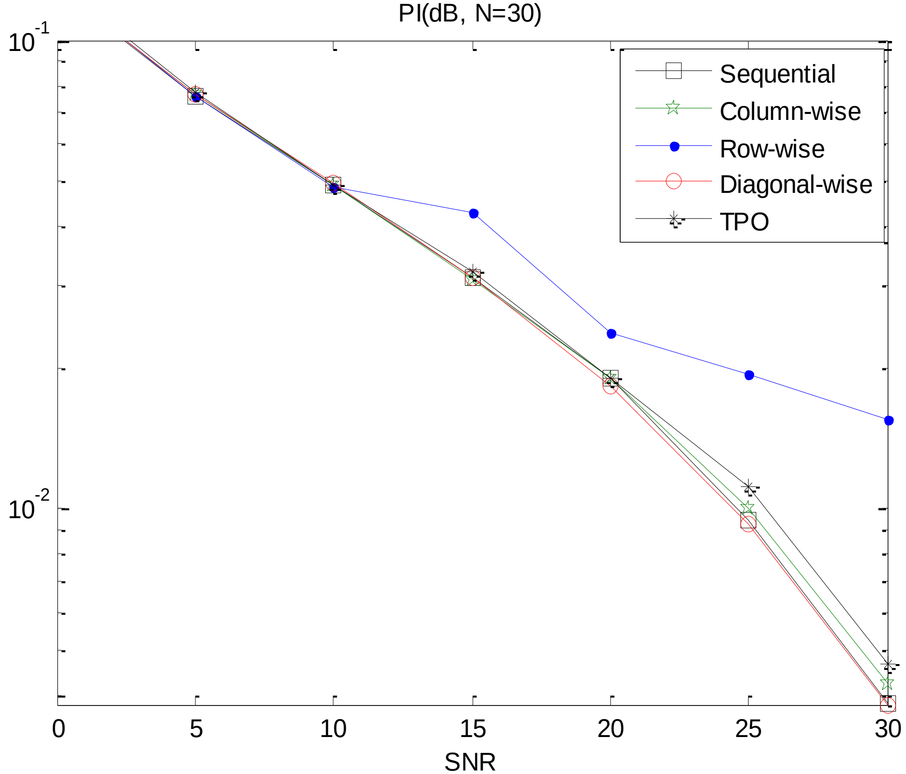


Fig.7 Performance index versus SNR

- equal performance for low SNR
- Row-wise's performance loss
- Parallelization schemes perform well

5. Conclusion

- Considers the parallelization of JD problems
  - Joint diagonalization has been widely applied.
  - This paper addresses the parallelization of JD with successive rotation.
  - Introduces 3 parallelized schemes.

- Behavior of the parallelized schemes
  - Largely reduce the running time without losing the JD quality
  - Row-wise scheme has slight performance loss

5. Conclusion

- Considers the parallelization of JD problems
  - Joint diagonalization has been widely applied.
  - This paper addresses the parallelization of JD with successive rotation.
  - Introduces 3 parallelized schemes.

- Behavior of the parallelized schemes
  - Largely reduce the running time without losing the JD quality
  - Row-wise scheme has slight performance loss

5. Conclusion

- Considers the parallelization of JD problems
  - Joint diagonalization has been widely applied.
  - This paper addresses the parallelization of JD with successive rotation.
  - Introduces 3 parallelized schemes.

- Behavior of the parallelized schemes
  - Largely reduce the running time without losing the JD quality
  - Row-wise scheme has slight performance loss

谢谢！

Thank you！