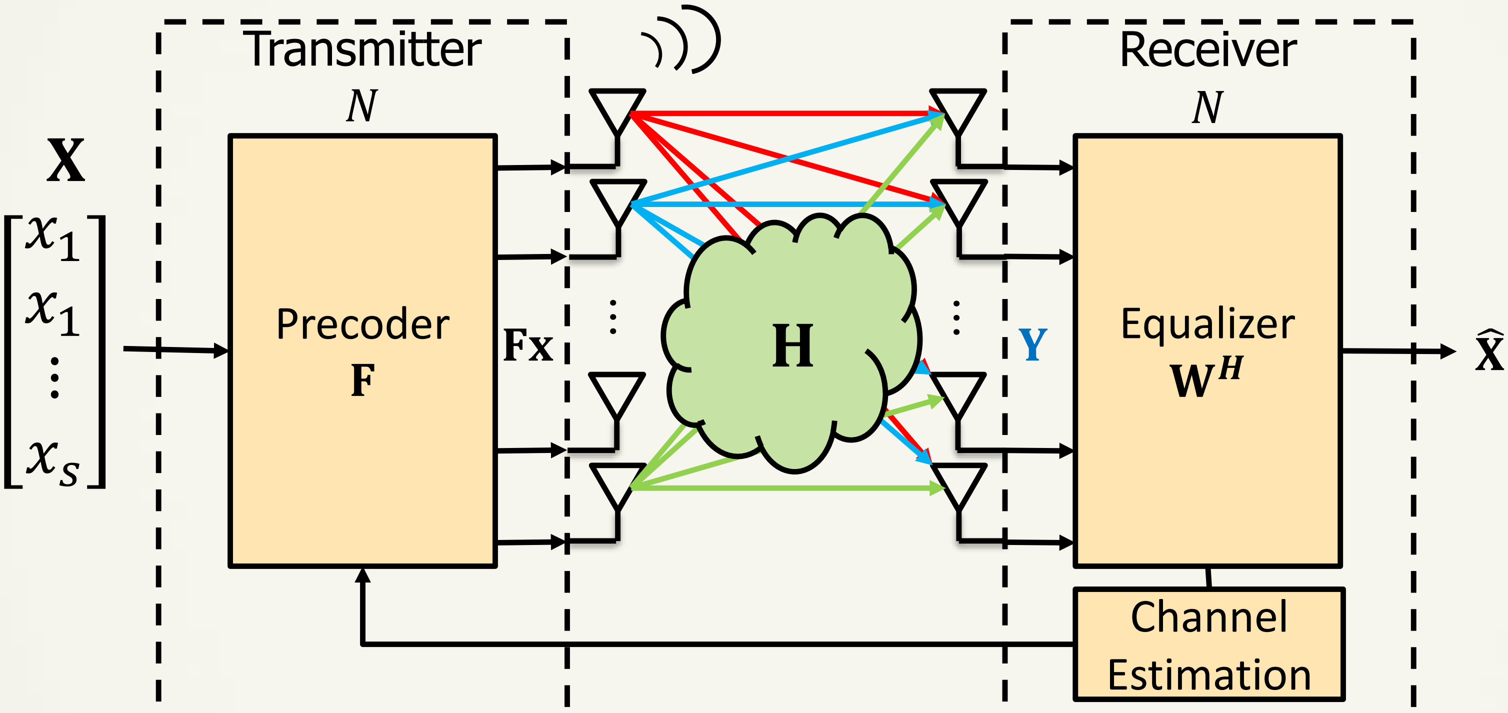
Design and Implementation of Matrix Decomposition

Professor : Pei-Yun Tsai
Department of Electrical Engineering,
National Central University

Applications in Communication Systems

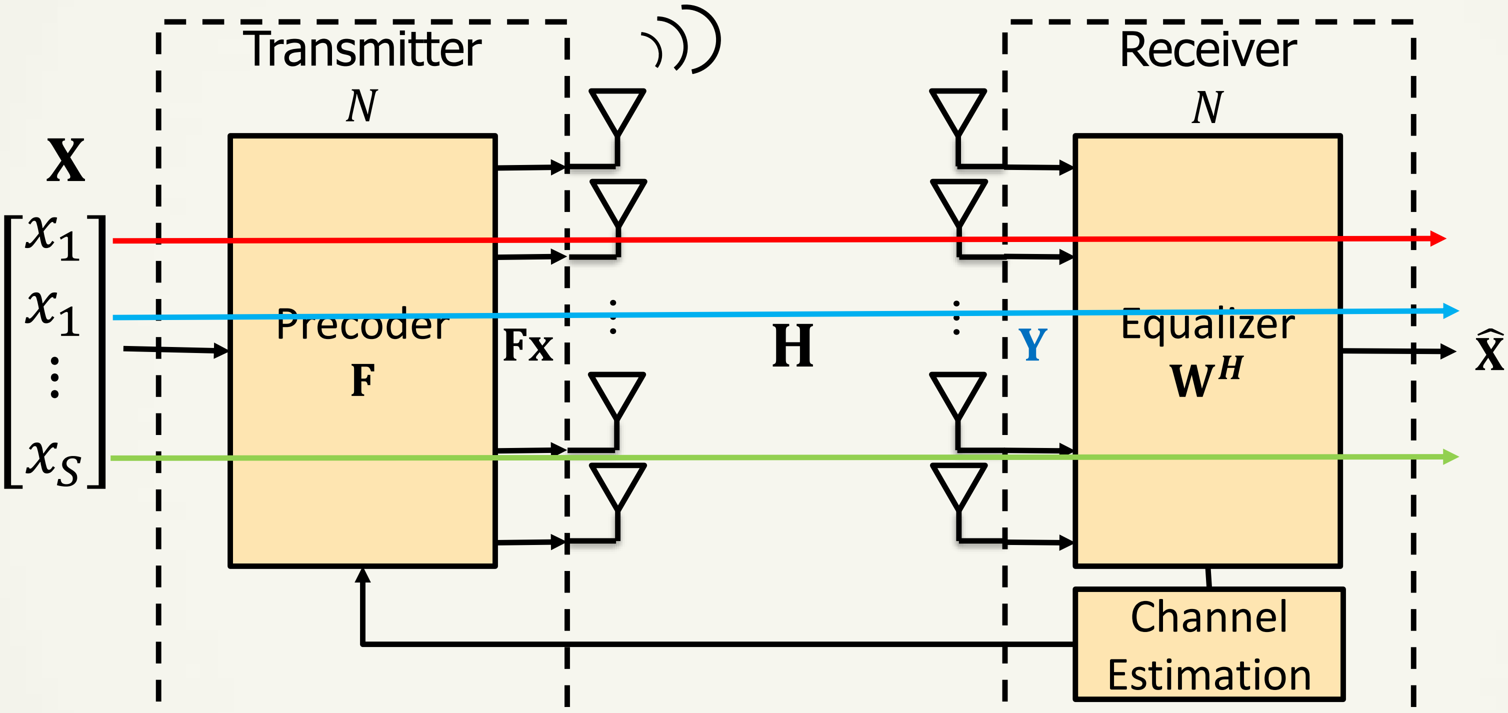


MIMO Precoding System

$$\begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots & h_{1,N} \\ h_{2,1} & h_{2,2} & h_{2,3} & \cdots & h_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{N,1} & h_{N,2} & h_{N,3} & \cdots & h_{N,N} \end{bmatrix}_{N \times N}$$

$$\mathbf{Y} = \mathbf{H} \cdot \mathbf{Fx} + \mathbf{n}$$

Channel Decomposition

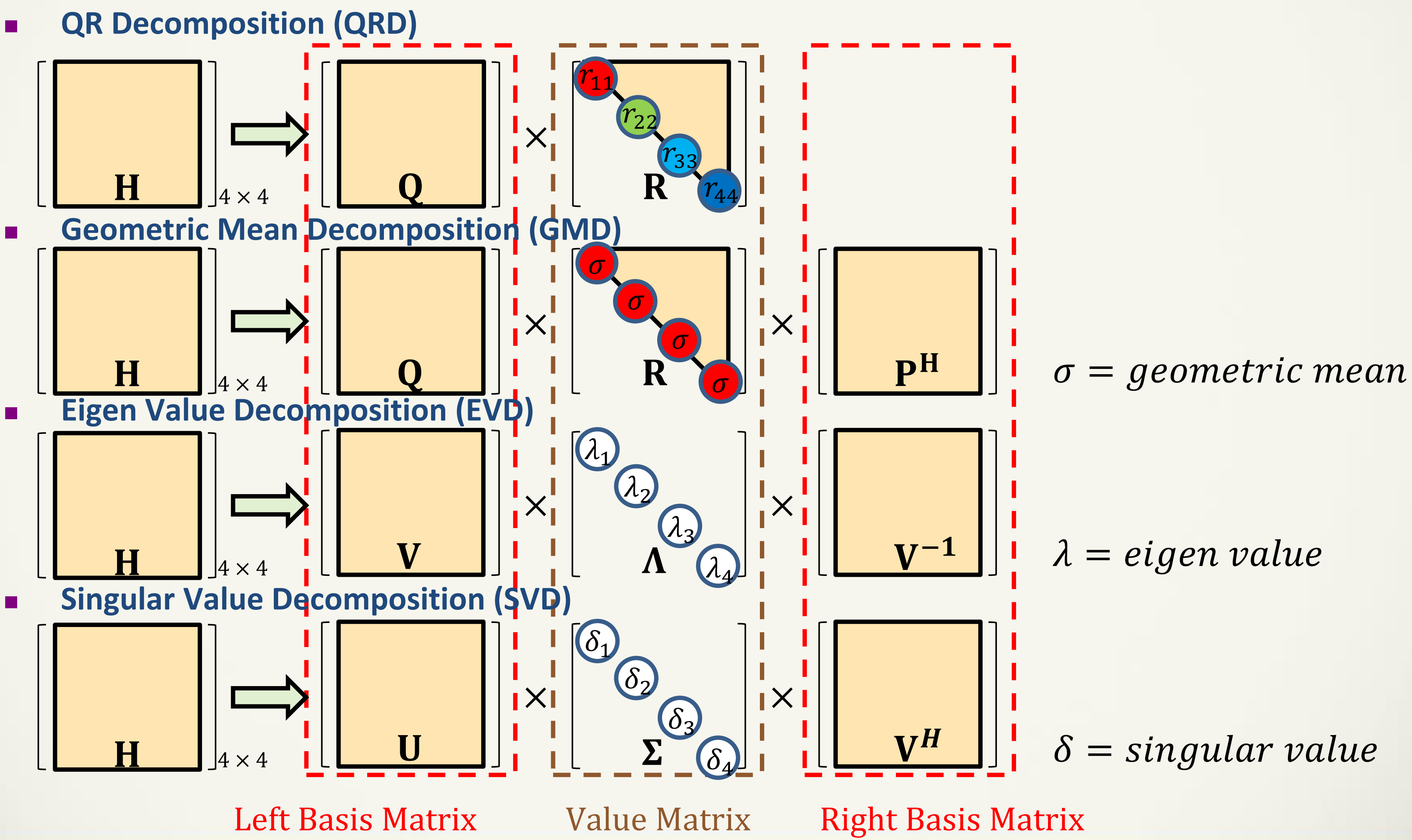


MIMO Precoding System

$$\mathbf{W}^H\mathbf{H}\mathbf{F} = \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_S \end{bmatrix}_{S \times S}$$

$$\begin{aligned} \hat{\mathbf{X}} &= \mathbf{W}^H\mathbf{Y} \\ &= \mathbf{W}^H\mathbf{H}\mathbf{F}\mathbf{x} + \mathbf{W}^H\mathbf{n} \\ &= \mathbf{\Sigma}\mathbf{x} + \mathbf{W}^H\mathbf{n} \end{aligned}$$

Matrix Decomposition



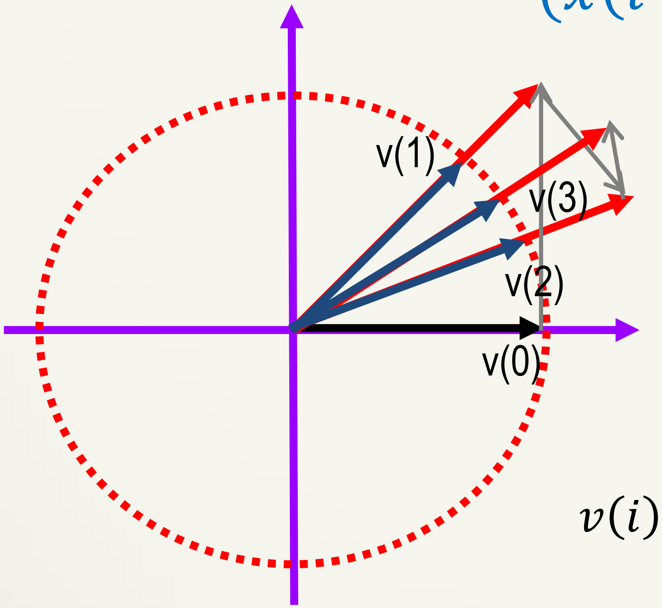
Methods

- Coordinate rotation digital computer (CORDIC)
- Matrix decomposition
  - QR decomposition
  - Eigenvalue decomposition
  - Singular value decomposition

CORDIC (1/2) (Rotation)

- CORDIC (coordinate rotation digital computer)
  - Linear Mode
  - Circular Mode
  - Hyperbolic Mode

$$(x(i+1)+jy(i+1)) = e^{\,j\theta_i}(x(i)+jy(i))$$



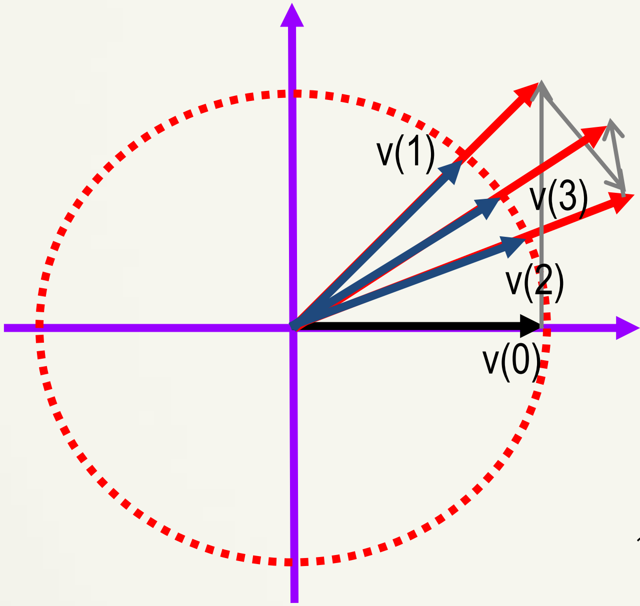
$$\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

$$= \cos(\theta_i) \begin{bmatrix} 1 & -\tan(\theta_i) \\ \tan(\theta_i) & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

$$v(i) = \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

CORDIC (2/2) (Rotation)

- In circular mode,
  - Angle of the ith micro rotation is defined as $\theta_i = tan^{-1}(\frac{1}{2^i})$



$$\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \cos(\theta_i) \begin{bmatrix} 1 & -\tan(\theta_i) \\ \tan(\theta_i) & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

$$= \cos(\theta_i) \begin{bmatrix} 1 & -2^{-i} \\ 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

$$v(i) = \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

CORDIC Arctangent Function (1/2) (Vectoring)

- Initialization

$$x(0) = x, y(0) = y, \theta'(0) = 0$$

- Direction

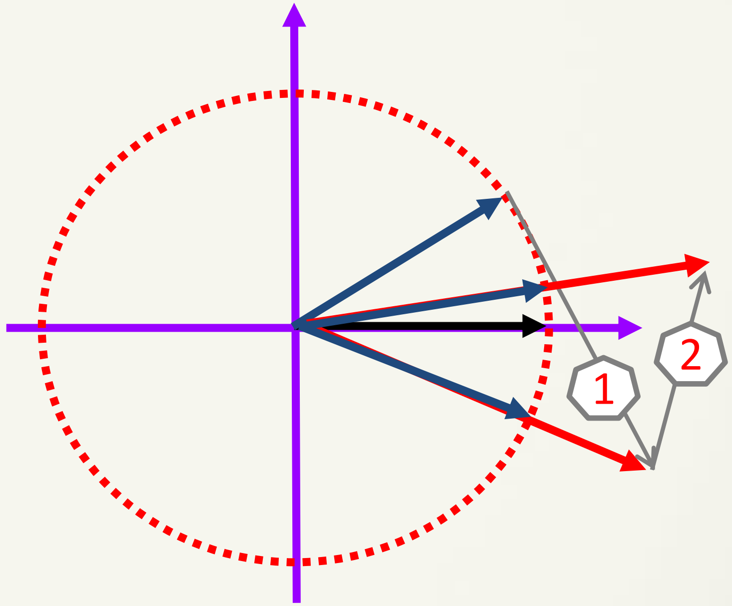
$$\mu_i = -sign(y(i))$$

- Micro rotation

$$\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} 1 & -\mu_i 2^{-i} \\ \mu_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

- Angle accumulation

$$\theta'(i+1) = \theta'(i) - \mu_i \theta_i$$



CORDIC Arctangent Function (2/2) (Vectoring)

- To restore the original vector length, we need scaling factor

$$S(\infty) = \prod_{i=0}^{\infty} \sqrt{(1 + 2^{-2i}} = 1.6468$$

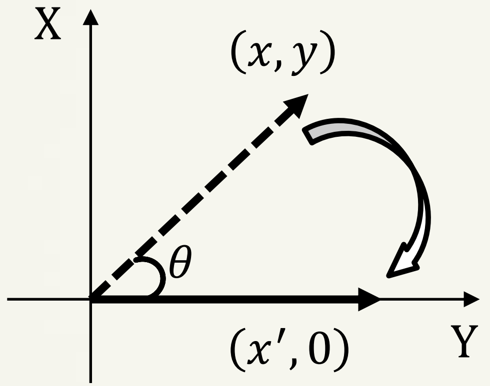
- The converge range of CORDIC algorithm is

$$\theta_{MAX} = \sum_{i=0}^{\infty} tan^{-1}\left(2^{-i}\right) = 1.7433 \ldots (\sim 99°)$$

- Hence, initial mapping of the vectors in the second and the third quadrant is required.

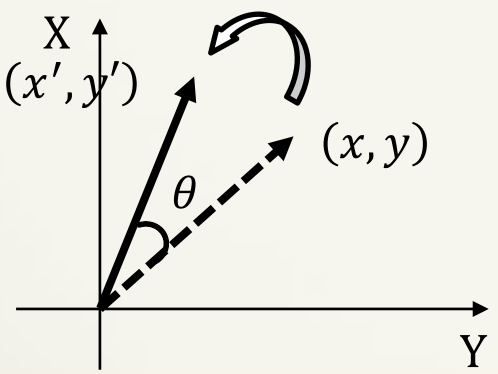
Givens Rotation for Real Numbers

- Givens rotation for two real numbers
  - **Vectoring (Nullification, Real to Zero (R2Z))**



$$\begin{bmatrix} x' \\ 0 \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \quad \left(\theta = \tan^{-1}\frac{y}{x}\right)$$

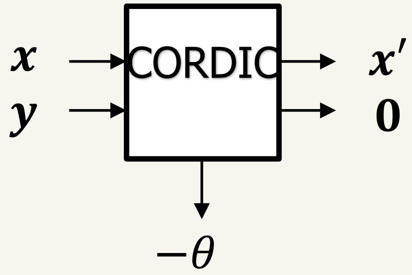
  - **Rotation**



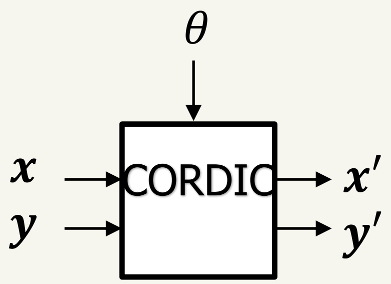
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \quad (\theta\ is\ given)$$

CORDIC for Real Numbers

- Vectoring



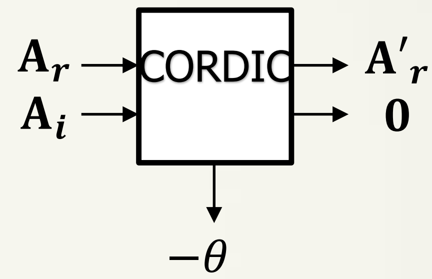
- Rotation



Givens Rotation for One Complex Number

- Givens rotation for one complex number

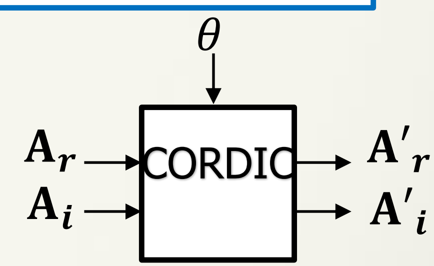
$$e^{-i\theta}\cdot(\mathbf{A}_r+j\mathbf{A}_i) = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}\cdot\begin{bmatrix}\mathbf{A}_r\\ \mathbf{A}_i\end{bmatrix}$$



  - Vectoring (Nullification, Complex to Real (C2R))

$$\mathbf{A}'_r = e^{-i\theta}\cdot(\mathbf{A}_r+j\mathbf{A}_i) = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}\cdot\begin{bmatrix}\mathbf{A}_r\\ \mathbf{A}_i\end{bmatrix} \quad \left(\theta = \tan^{-1}\frac{\mathbf{A}_i}{\mathbf{A}_r}\right)$$

  - Rotation of one complex number



$$\mathbf{A}'_r + j\mathbf{A}'_i = e^{i\theta}\cdot(\mathbf{A}_r+j\mathbf{A}_i) = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}\cdot\begin{bmatrix}\mathbf{A}_r\\ \mathbf{A}_i\end{bmatrix} \quad (\theta\ is\ given)$$

Givens Rotation for Two Complex Numbers (1/2)

- Givens rotation for two complex numbers
  - Vectoring (Nullification, Complex to Real (C2R) and Real to Zero (R2Z))

$$\mathbf{Gu} = \begin{bmatrix} cos\theta_z & sin\theta_z \\ -sin\theta_z & cos\theta_z \end{bmatrix} \begin{bmatrix} e^{-j\theta_x} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & e^{-j\theta_y} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A}_r + j\mathbf{A}_i \\ \mathbf{B}_r + j\mathbf{B}_i \end{bmatrix} = \begin{bmatrix} \mathbf{A'}_r \\ \mathbf{0} \end{bmatrix}$$

(R2Z)

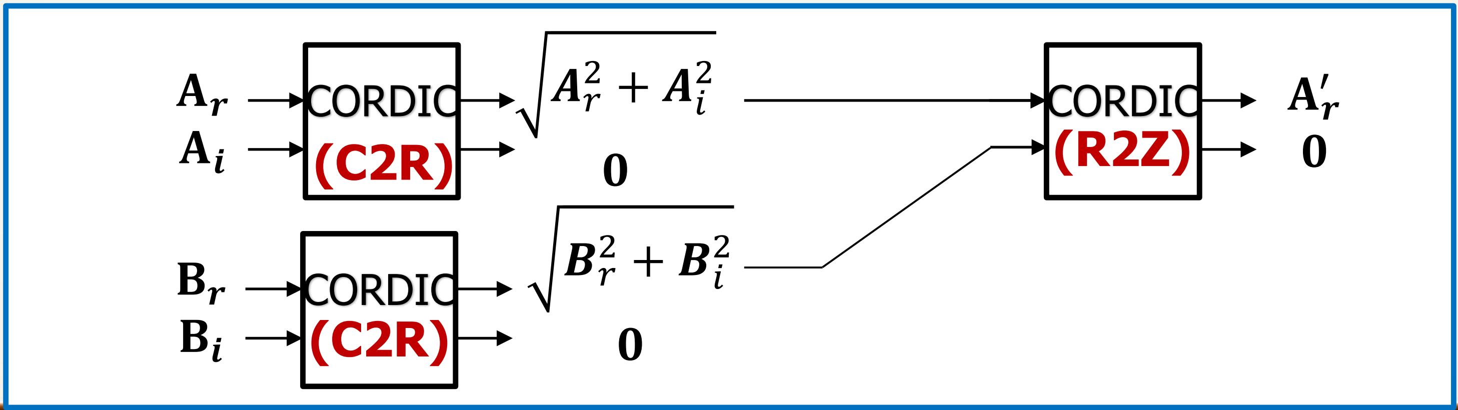
$$\theta_z = \tan^{-1} \frac{\sqrt{\boldsymbol{B}_r^2 + \boldsymbol{B}_i^2}}{\sqrt{\boldsymbol{A}_r^2 + \boldsymbol{A}_i^2}}$$

(C2R)

$$\theta_x = \tan^{-1} \frac{\mathbf{A}_i}{\mathbf{A}_r}$$

(C2R)

$$\theta_y = \tan^{-1} \frac{\mathbf{B}_i}{\mathbf{B}_r}$$

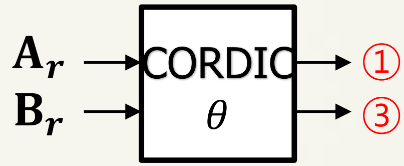


Givens Rotation for Two Complex Numbers (2/2)

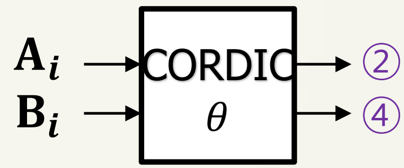
- **Rotation of two complex numbers**

$$\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A}_r + j\mathbf{A}_i \\ \mathbf{B}_r + j\mathbf{B}_i \end{bmatrix} = \begin{bmatrix} \underbrace{(\mathbf{A}_r cos\theta - \mathbf{B}_r sin\theta)}_{①} + j\underbrace{(\mathbf{A}_i cos\theta - \mathbf{B}_i sin\theta)}_{②} \\ \underbrace{(\mathbf{A}_r sin\theta + \mathbf{B}_r cos\theta)}_{③} + j\underbrace{(\mathbf{A}_i sin\theta + \mathbf{B}_i cos\theta)}_{④} \end{bmatrix}$$

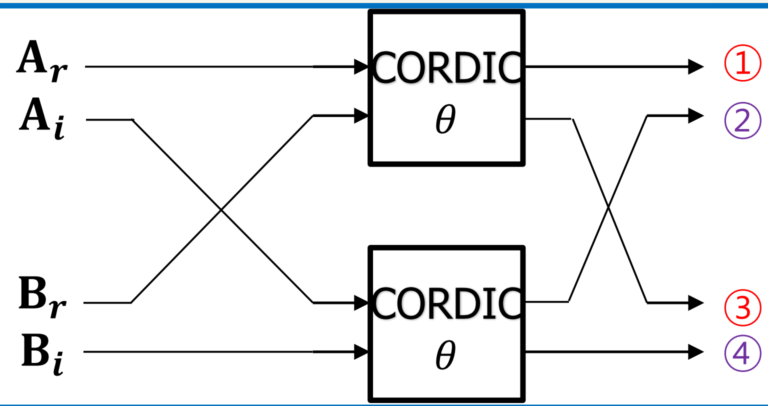
①、③ $\Rightarrow \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A}_r \\ \mathbf{B}_r \end{bmatrix} \longrightarrow$



②、④ $\Rightarrow \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A}_i \\ \mathbf{B}_i \end{bmatrix} \longrightarrow$



$\therefore \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A}_r + j\mathbf{A}_i \\ \mathbf{B}_r + j\mathbf{B}_i \end{bmatrix} =$



QR Decomposition

QR Decomposition Algorithm

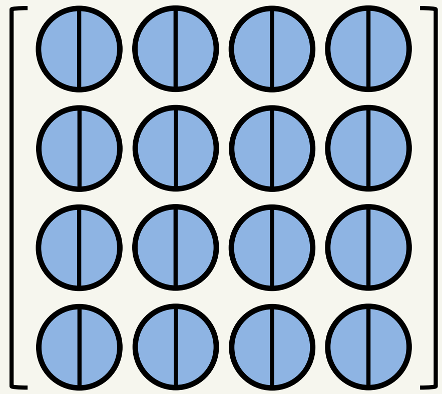
- Gram Schmidt algorithm
  - Modified Gram Schmidt algorithm
- Givens rotation
- Householder transformation

- With QR decomposition, $\mathbf{H} = \mathbf{QR}$
  - Q is a unitary matrix
  - R is an upper triangular matrix

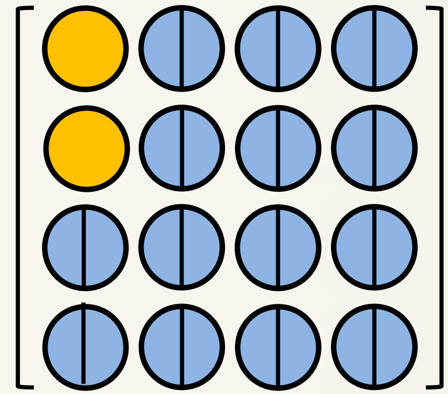
Givens Rotation for QR Decomposition (1/2)

$r_2 \cdot r_1 \cdot \mathbf{H}$

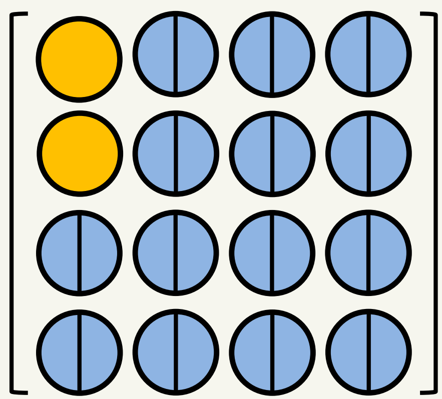
$$\begin{bmatrix} e^{j\theta_1} & 0 & 0 & 0 \\ 0 & e^{j\theta_2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$



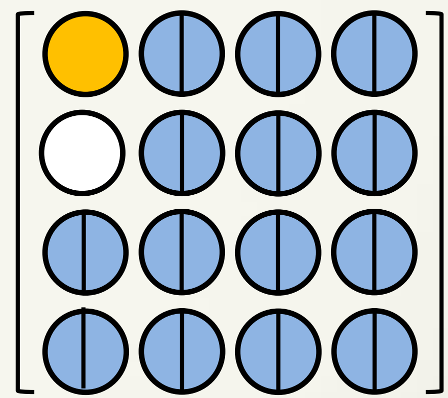
$=$

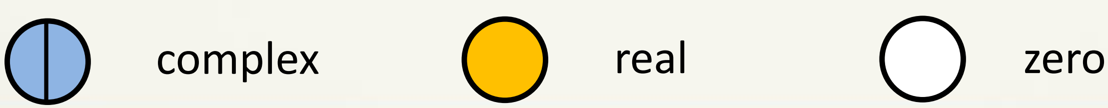


$$\begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$



$=$

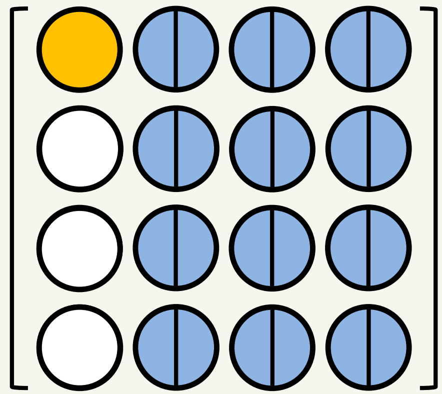




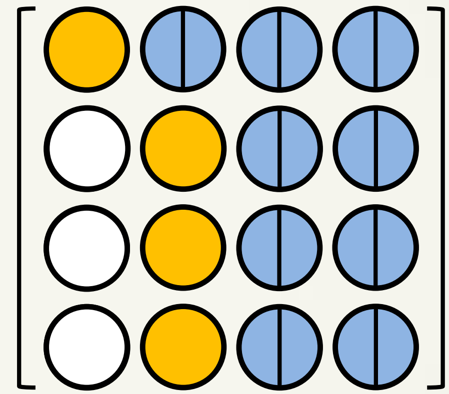
Givens Rotation for QR Decomposition (2/2)

$$\overbrace{r_N \cdot \cdots \; r_5 \cdot r_4 \cdot r_3 \cdot r_2 \cdot r_1 \cdot}^{\mathbf{Q}^H} \; \mathbf{H}$$

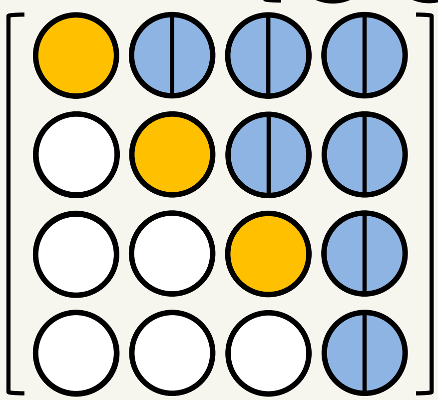
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{j\theta_8} & 0 & 0 \\ 0 & 0 & e^{j\theta_9} & 0 \\ 0 & 0 & 0 & e^{j\theta_{10}} \end{bmatrix} \cdot$$



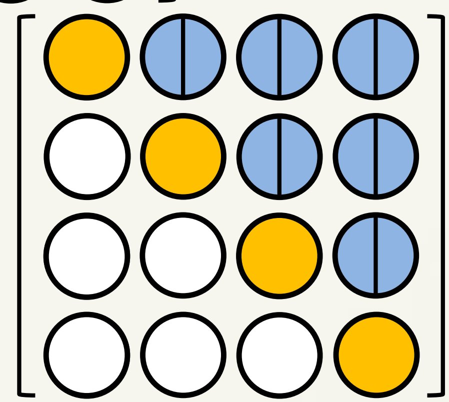
$=$



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{j\theta_N} \end{bmatrix} \cdot$$



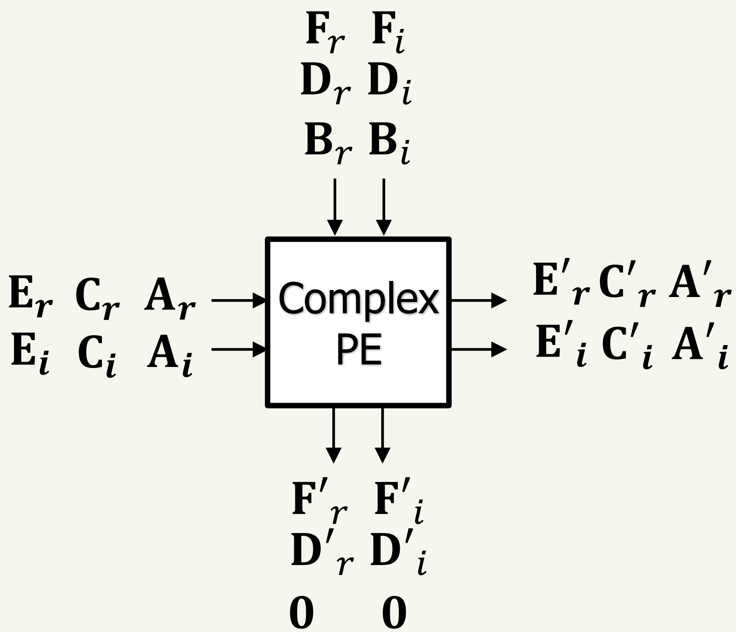
$=$

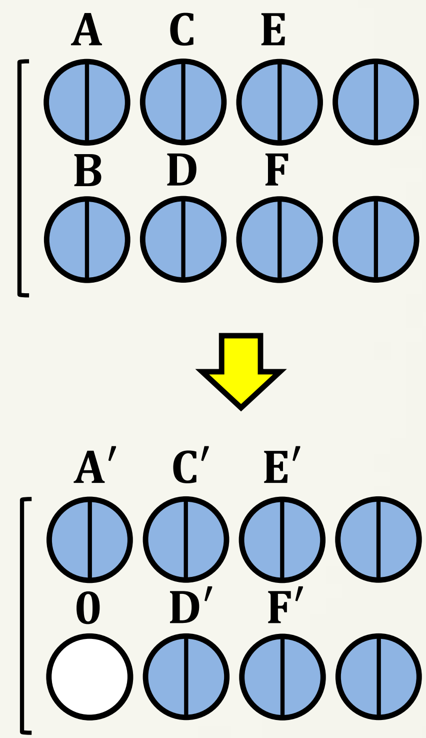


$= \mathbf{R}$

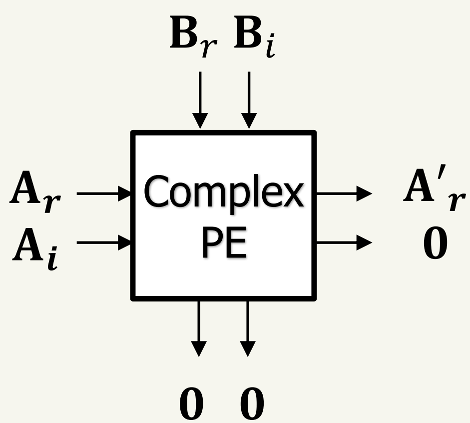
$$\mathbf{Q} = \mathbf{I} \cdot r_1^H \cdot r_2^H \cdot r_3^H \cdots r_N^H$$

Architecture Design
Complex Processing Element (Complex PE)

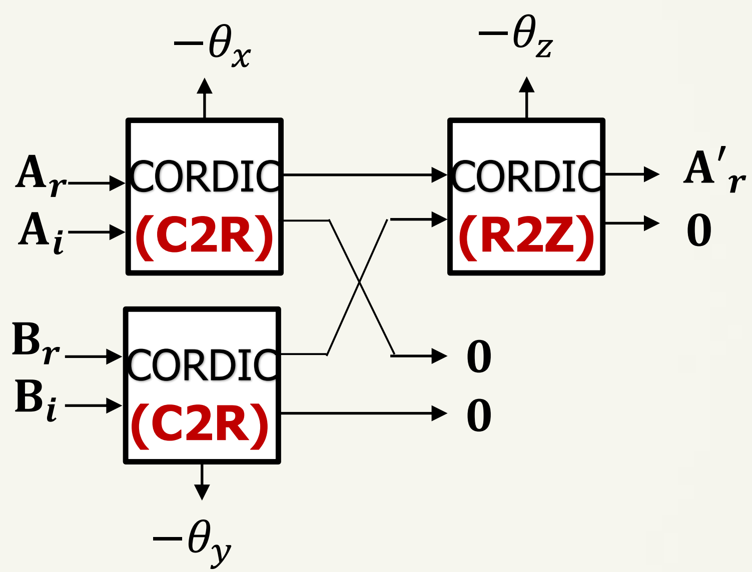




Complex PE Vectoring Mode

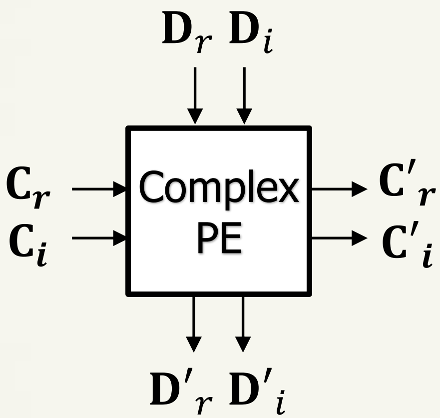


[1]

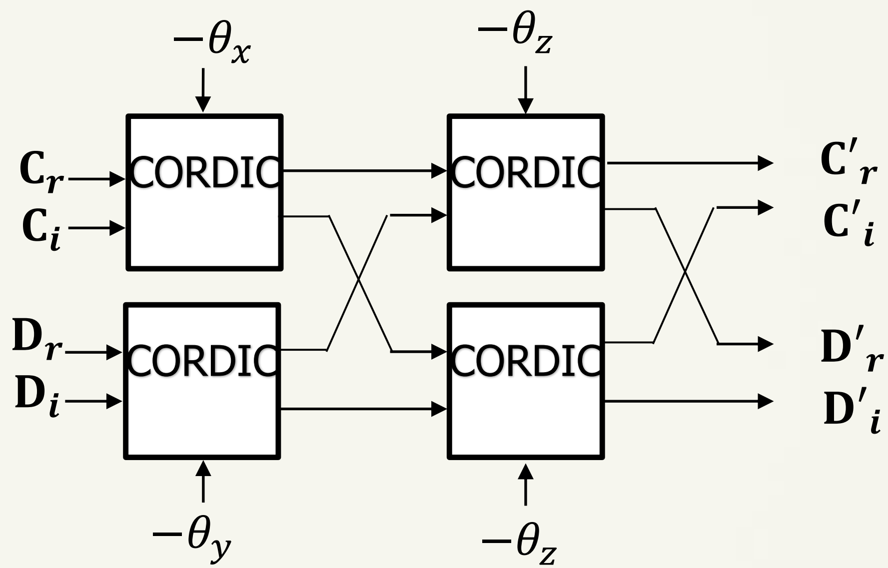


$$\begin{bmatrix} cos\theta_z & sin\theta_z \\ -sin\theta_z & cos\theta_z \end{bmatrix}\begin{bmatrix} e^{-j\theta_x} & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & e^{-j\theta_y} \end{bmatrix}\cdot\begin{bmatrix} \mathbf{A}_r + j\mathbf{A}_i \\ \mathbf{B}_r + j\mathbf{B}_i \end{bmatrix}=\begin{bmatrix} \mathbf{A'}_r \\ \mathbf{0} \end{bmatrix}$$

Complex PE Rotation Mode

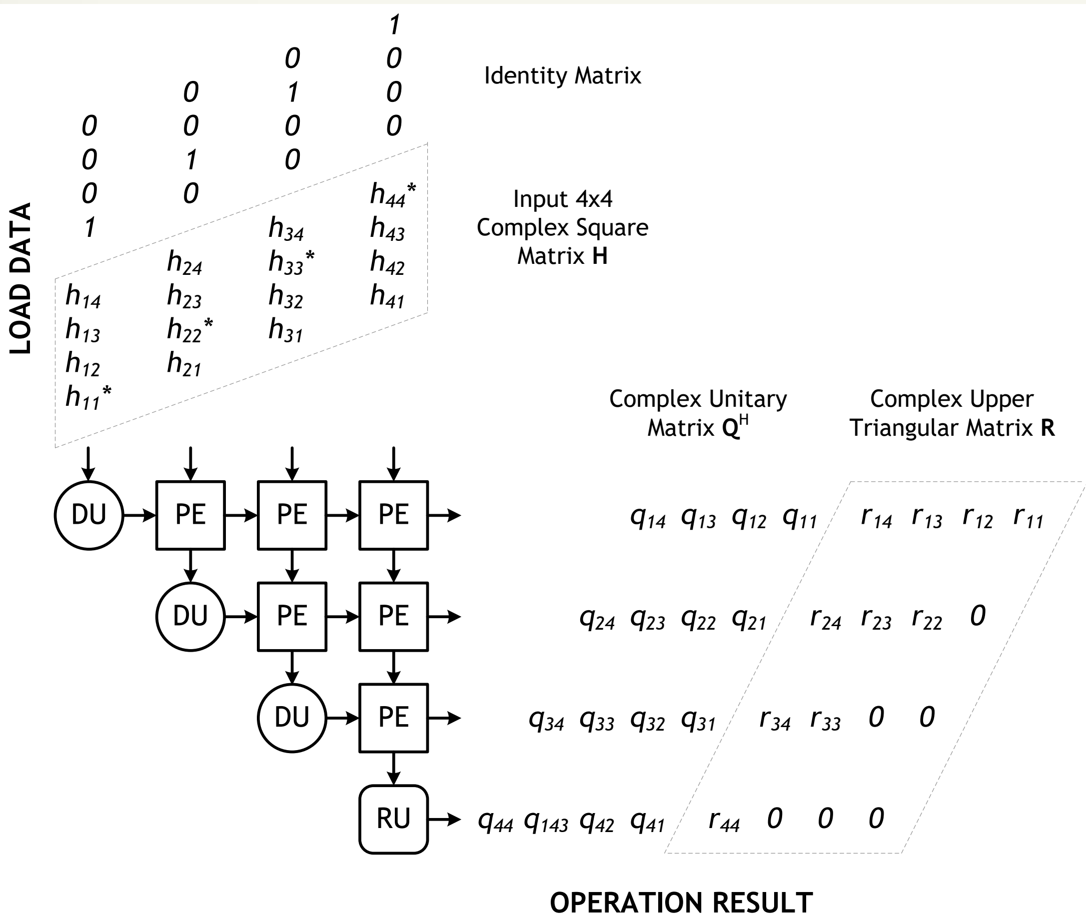


[1]

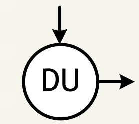


$$\begin{bmatrix} cos\theta_z & sin\theta_z \\ -sin\theta_z & cos\theta_z \end{bmatrix}\begin{bmatrix} e^{-j\theta_x} & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & e^{-j\theta_y} \end{bmatrix}\cdot\begin{bmatrix} \mathbf{C}_r + j\mathbf{C}_i \\ \mathbf{D}_r + j\mathbf{D}_i \end{bmatrix}=\begin{bmatrix} \mathbf{C'}_r + j\mathbf{C'}_i \\ \mathbf{D'}_r + j\mathbf{D'}_i \end{bmatrix}$$

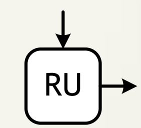
Conventional Systolic Array for QR Decomposition (1/2)



- The input matrix is loaded in the skewed triangular shape.
- Input identity matrix, we can obtain the unitary matrix.
- DU (Delay Unit)



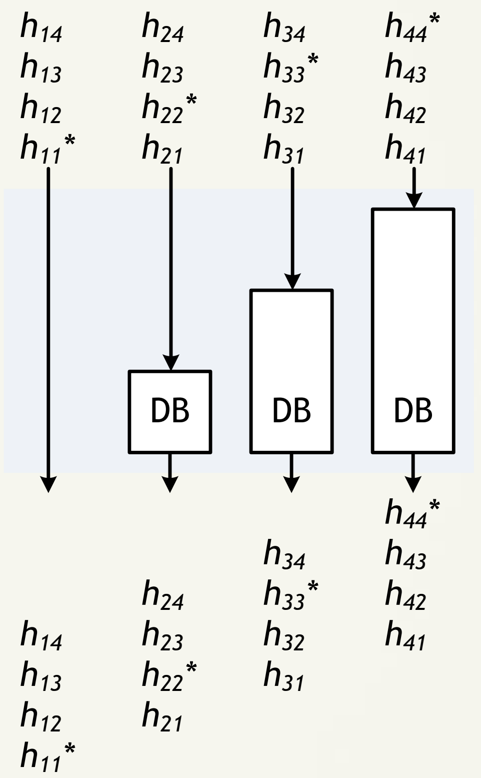
  - Reads data, delays them, and then passes to the output.
- RU (Rotation Unit)



  - Eliminate complex lowest diagonal element of matrix by additional rotation.

Conventional Systolic Array for QR Decomposition (2/2)

- In the conventional design, four input signal streams enter into the systolic array in skewed form. Hence, delay buffers are required.
- Delays data for period equal to PE operation time.
- If quantity of pipeline stage in CORDIC is large, a lot of delay buffers are introduced to hardware at input.



Eigen Value Decomposition

Eigen Value Decomposition

- For Hermitian symmetric matrix $\mathbf{A} = \mathbf{H}^{\mathrm{H}}\mathbf{H}$
  - $\mathbf{A} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{-1}$

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \lambda_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \lambda_S \end{bmatrix} \qquad \mathbf{V} = [\mathbf{v_1} \quad \mathbf{v_2} \quad \cdots \quad \mathbf{v}_S]$$

  - λ_s is the eigenvalue and $\mathbf{v}_s$ is the eigen vector

Eigen Value Decomposition Algorithm [3]

- Power iteration
- Inverse power iteration
- Iterative QR algorithm (with shift)

Power Method (1/2)

Algorithm: Power Method

Given Hermitian symmetric matrix $\mathbf{A} \in \mathbb{C}^{\mathrm{N}\times\mathrm{N}}$

1. $\mathbf{A}(1) = \mathbf{A}$
2. **For** $s = 1{:}\,S$
3. $\quad \mathbf{q}_s^{(0)} = [\mathbf{1}\ \mathbf{0}\ \ldots\mathbf{0}]^{\boldsymbol{T}}$
4. $\quad$ **For** $i = 1{:}\,I_1$
5. $\quad\quad \mathbf{z}_s^{(i)} = \mathbf{A}(s)\mathbf{q}_s^{(i-1)}$
6. $\quad\quad \mathbf{q}_s^{(i)} = \mathbf{z}_s^{(i)} / \left\|\mathbf{z}_s^{(i)}\right\|_2$
7. $\quad$ **End**
8. $\quad \lambda_s = \mathbf{q}_s^{(I_1)\mathbf{H}}\mathbf{A}(s)\mathbf{q}_s^{(I_1)}$
9. $\quad \mathbf{A}(s+1) = \left(\mathbf{I} - \mathbf{q}_s^{(I_1)}\mathbf{q}_s^{(I_1)\boldsymbol{H}}\right)\mathbf{A}(s)$
10. **End**
11. $\boldsymbol{\Lambda} = [\mathbf{q}_1^{(I_1)} \quad \ldots \quad \mathbf{q}_S^{(I_1)}]$,

Eigenvalue

Deflation

Power Method (2/2)

- Power method has linear convergence rate
  - Slow
  - Especially when two eigenvalues are close
- Deflation is helpful to find the remaining eigenvectors
  - Eigenvector corresponding to the largest eigenvalue is obtained first.

Inverse Power Iteration (1/2)

Algorithm: Inverse Power Method

Given Hermitian symmetric matrix $\mathbf{A} \in \mathbb{C}^{\mathrm{N}\times\mathrm{N}}$

1. $\mathbf{A}(1) = \mathbf{A}$
2. For $s = 1:S$
3. $\quad \mathbf{q}_s^{(0)} = [\mathbf{1}\ \mathbf{0}\ \ldots \mathbf{0}]^{T}$
4. $\quad$ For $i = 1:I_1$
5. $\quad\quad \mathbf{z}_s^{(i)} = (\mathbf{A}(s) - \lambda\mathbf{I})^{-1}\mathbf{q}_s^{(i-1)}$
6. $\quad\quad \mathbf{q}_s^{(i)} = \mathbf{z}_s^{(i)} / \left\|\mathbf{z}_s^{(i)}\right\|_2$
7. $\quad$ End
8. $\quad \lambda_s = \mathbf{q}_s^{(I_1)\mathbf{H}}\mathbf{A}(s)\mathbf{q}_s^{(I_1)}$
9. $\quad \mathbf{A}(s+1) = \left(\mathbf{I} - \mathbf{q}_s^{(I_1)}\mathbf{q}_s^{(I_1)H}\right)\mathbf{A}(s)$
10. End
11. $\mathbf{\Lambda} = [\mathbf{q}_1^{(I_1)} \quad \ldots \quad \mathbf{q}_S^{(I_1)}]$,

Shift λ

Eigenvalue

Deflation

Inverse Power Method (2/2)

- Given that $(\mathbf{q}_k, \lambda_k)$ is eigen pair of matrix $\mathbf{A}$,

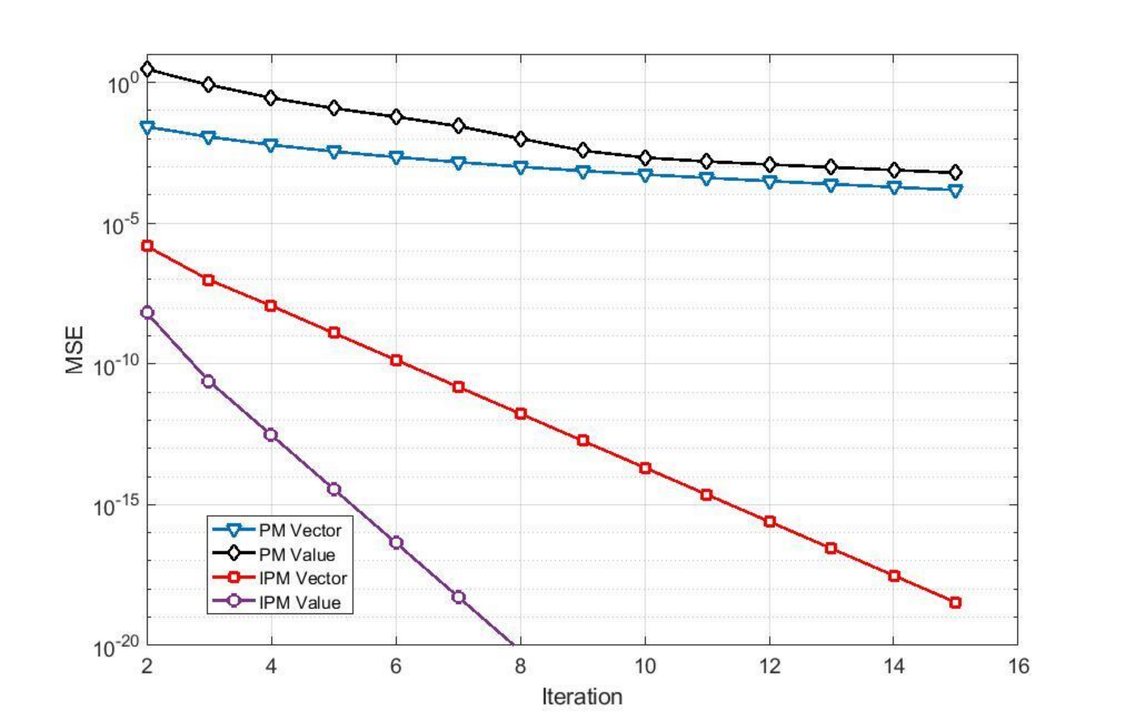
$$\mathbf{q}_s^{(i)} = \sum_{k=1}^{K} a_k^{(i)} \mathbf{q}_k \qquad \mathbf{A}\mathbf{q}_k = \lambda_k \mathbf{q}_k$$

$$(\mathbf{A} - \lambda)^{-\mathbf{1}} \mathbf{q}_s^{(i-1)} = \sum_{k=1}^{K} \frac{a_k^{(i-1)}}{\lambda_k - \lambda} \mathbf{q}_k$$

- Thus, if λ is very close to an eigenvalue λ_s of A, the inverse iteration will make $\mathbf{q}_s^{(i)}$ close to the direction of $\mathbf{q}_s$.

Comparison

- Use 4x4 complex channel matrix to generate Hermitian symmetric matrix A
- Inverse power method uses true eigenvalue rounding to 10^{-1} as the shift



Iterative QR Algorithm

Algorithm: Iterative QR

Given Hermitian symmetric matrix $\mathbf{C} \in \mathbb{C}^{N \times N}$

// First phase

1. $[\mathbf{U}_{EVD}^{(\mathbf{0})}, \mathbf{A}^{(0)}]$=HessenbergReduction(C)

// Second phase

i =0,

1. while (!converged)
2. $\mathbf{T}^{(i)} = \mathbf{A}^{(i)} - \mu_i \mathbf{I}$
3. $\left[\mathbf{Q}^{(i)}, \mathbf{R}^{(i)}\right] = QRD(\mathbf{T}^{(i)})$
4. $\mathbf{T}^{(i+1)} = \mathbf{R}^{(i)} \mathbf{Q}^{(i)}$
5. $\mathbf{A}^{(i+1)} = \mathbf{T}^{(i+1)} + \mu_i \mathbf{I}$
6. $\mathbf{U}_{EVD}^{(\boldsymbol{i}+\mathbf{1})} = \mathbf{U}_{EVD}^{(\boldsymbol{i})} \mathbf{Q}^{(i)}$
7. $i = \mathrm{i} + 1$

End

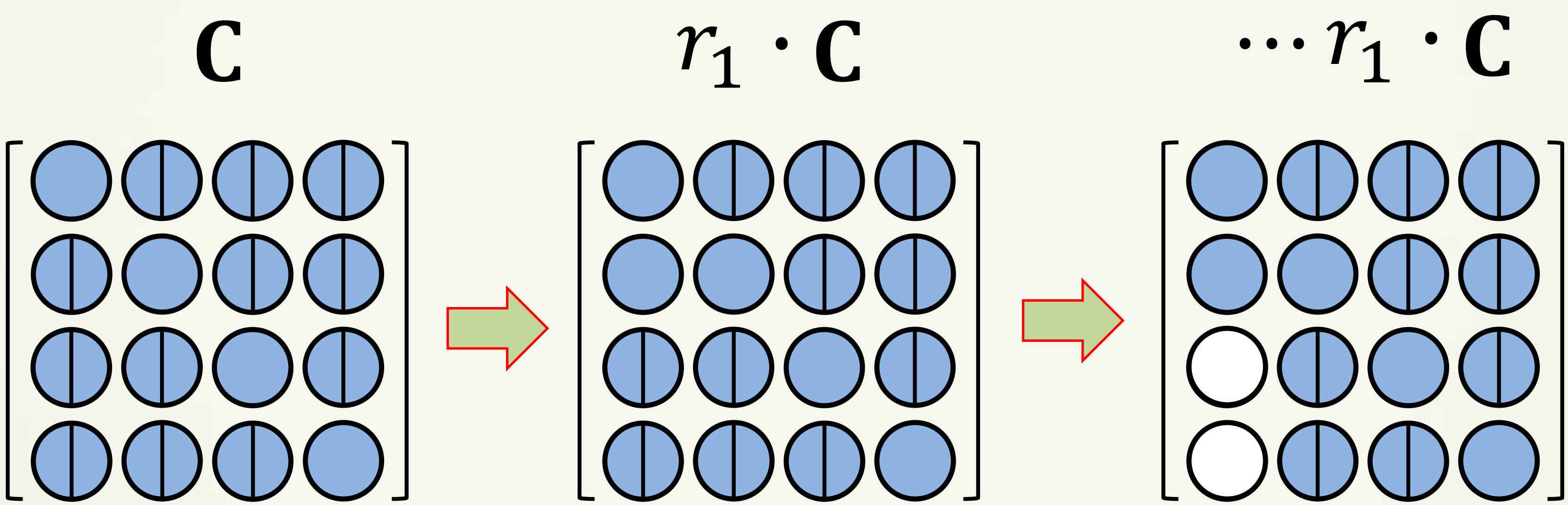
Phase 1: Triangularization

Phase 2: Iterative QR

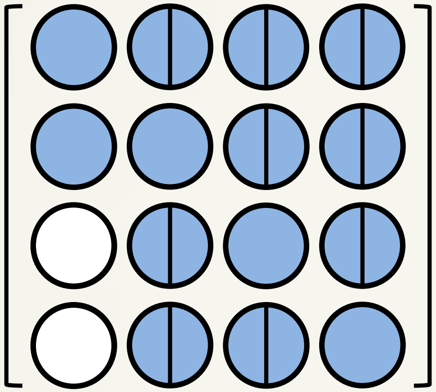
Phase 2: Shift to accelerate convergence

Hessenberg Reduction (1/3)

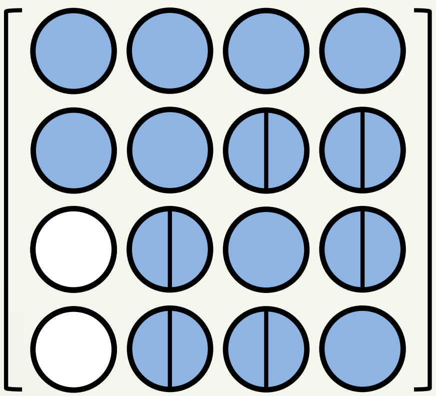
- Hessenberg reduction (tridiagonalization) helps to reduce computation complexity in the iterative procedure



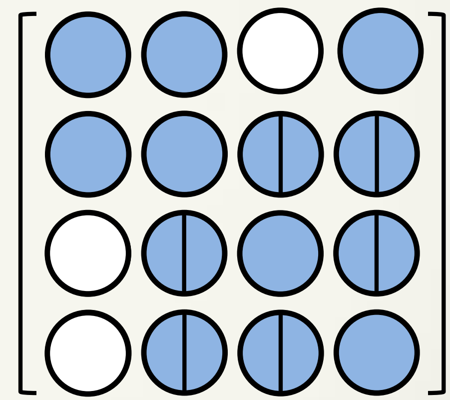
Hessenberg Reduction (2/3)



$$\cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-j\theta_1} & 0 & 0 \\ 0 & 0 & e^{-j\theta_2} & 0 \\ 0 & 0 & 0 & e^{-j\theta_3} \end{bmatrix}$$

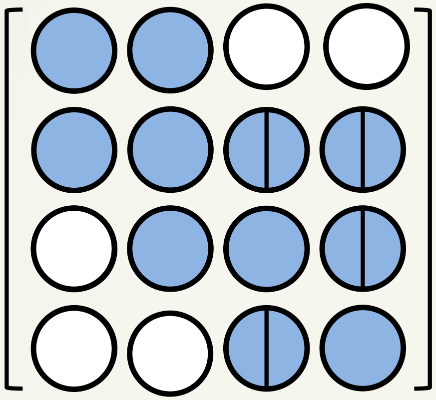


$$\cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta_4 & -sin\theta_4 & 0 \\ 0 & sin\theta_4 & cos\theta_4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

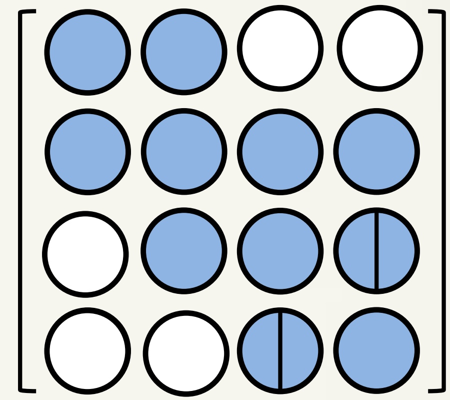


$$\vdots$$

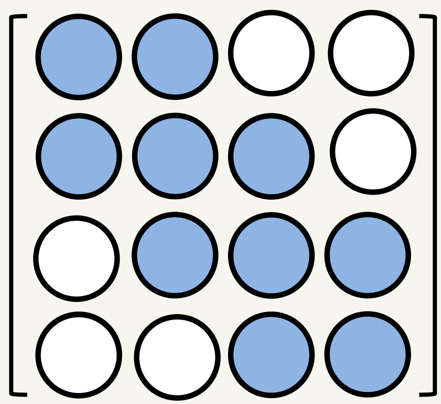
Hessenberg Reduction (3/3)



$$\cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{-j\theta_3} & 0 \\ 0 & 0 & 0 & e^{-j\theta_4} \end{bmatrix} =$$



$$\vdots$$



$$\overbrace{r_6 \cdot \quad \cdots \quad r_1}^{\mathbf{V}^{\mathrm{H}}} \cdot \ \mathbf{C} \ \overbrace{\cdot\, c_1 \ \cdot c_2 \ \cdot c_3 \ \cdot c_4 \ \cdot c_5 \ \cdot c_6}^{\mathbf{V}} = \mathbf{A}^{(0)}$$

Eigen Value Decomposition Based on Iterative QR Algorithm

$$\mathbf{T}^{(0)} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{-1}$$

$$\xrightarrow{QR} = \mathbf{R}^{(0)} \Rightarrow \mathbf{R}^{(0)} \cdot \mathbf{Q}^{(0)} = \mathbf{T}^{(1)}\left(= \mathbf{Q}^{(0)H} \cdot \mathbf{T}^{(0)} \cdot \mathbf{Q}^{(0)}\right)$$

$$\mathbf{T}^{(1)} \xrightarrow{QR} = \mathbf{R}^{(1)} \Rightarrow \mathbf{R}^{(1)} \cdot \mathbf{Q}^{(1)} = \mathbf{T}^{(2)}\left(= \mathbf{Q}^{(1)H} \cdot \mathbf{T}^{(1)} \cdot \mathbf{Q}^{(1)}\right)$$

$$\mathbf{T}^{(2)} \xrightarrow{QR} = \mathbf{R}^{(2)} \Rightarrow \mathbf{R}^{(2)} \cdot \mathbf{Q}^{(2)} = \mathbf{T}^{(3)}\left(= \mathbf{Q}^{(2)H} \cdot \mathbf{T}^{(2)} \cdot \mathbf{Q}^{(2)}\right)$$

$$\mathbf{T}^{(3)} \xrightarrow{QR} = \mathbf{R}^{(3)} \Rightarrow \mathbf{R}^{(3)} \cdot \mathbf{Q}^{(3)} = \mathbf{T}^{(4)}\left(= \mathbf{Q}^{(3)H} \cdot \mathbf{T}^{(3)} \cdot \mathbf{Q}^{(3)}\right)$$

$$\vdots$$

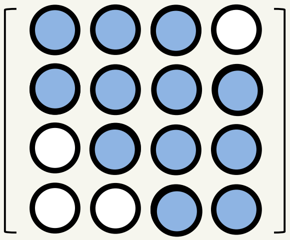
$$\mathbf{T}^{(\infty)} = \cancel{\mathbf{Q}^{(\infty)H} \cdot \cdots \cdot \mathbf{Q}^{(1)H} \cdot \mathbf{Q}^{(0)H}} \cdot \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{-1} \cancel{\cdot \mathbf{Q}^{(0)} \cdot \cdots \cdot \mathbf{Q}^{(\infty)}}$$

$$= \boldsymbol{\Lambda}$$

QR and RQ Operation

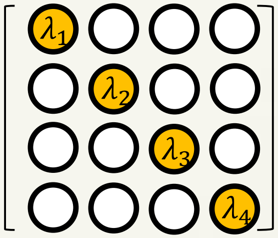
- Row operation (left multiplication) and column operation (right multiplication)

$$\underbrace{\mathbf{Q}^{(2)H}\,\mathbf{Q}^{(1)H}\,\mathbf{Q}^{(0)H}}_{\mathbf{Q^H}}\;\underset{(\mathbf{QR})}{\mathbf{T}^{(\mathbf{0})}}\cdot\underbrace{\mathbf{Q}^{(0)}\cdot\mathbf{Q}^{(1)}\cdot\mathbf{Q}^{(2)}}_{\mathbf{Q}}$$



$= \mathbf{R}$

$\Rightarrow \cdots$



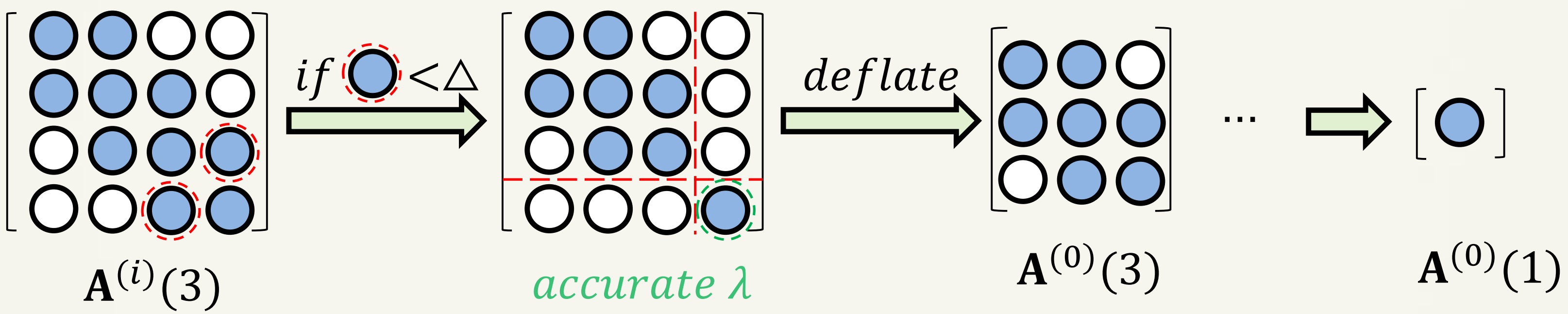
$\mathbf{\Lambda}$

Shift

- In iterative QR algorithm, shift μ_i can be
  - Zero shift
    - $\mu_i = 0$
    - Slow convergence
  - Rayleigh quotient shift
    - $\mu_i = \mathbf{A}_{N,N}^{(i)}$, the lower right corner entry of $\mathbf{A}^{(i)}$
    - Medium convergence rate
  - Wilkinson shift
    - Eeigenvalue of the lower right 2x2 submatrix close to $\mathbf{A}_{N,N}^{(i)}$
    - Fast convergence

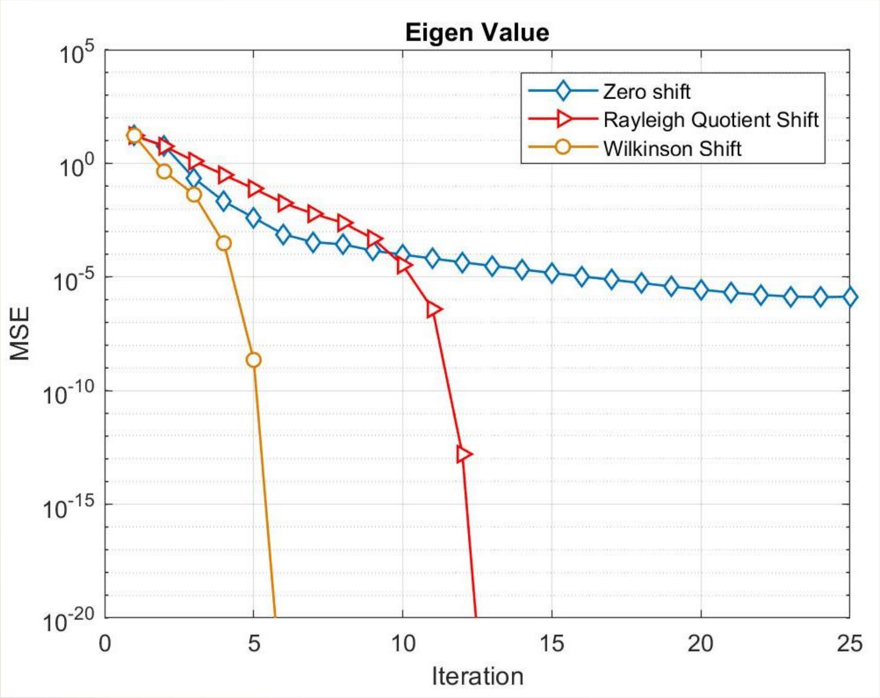
Deflation

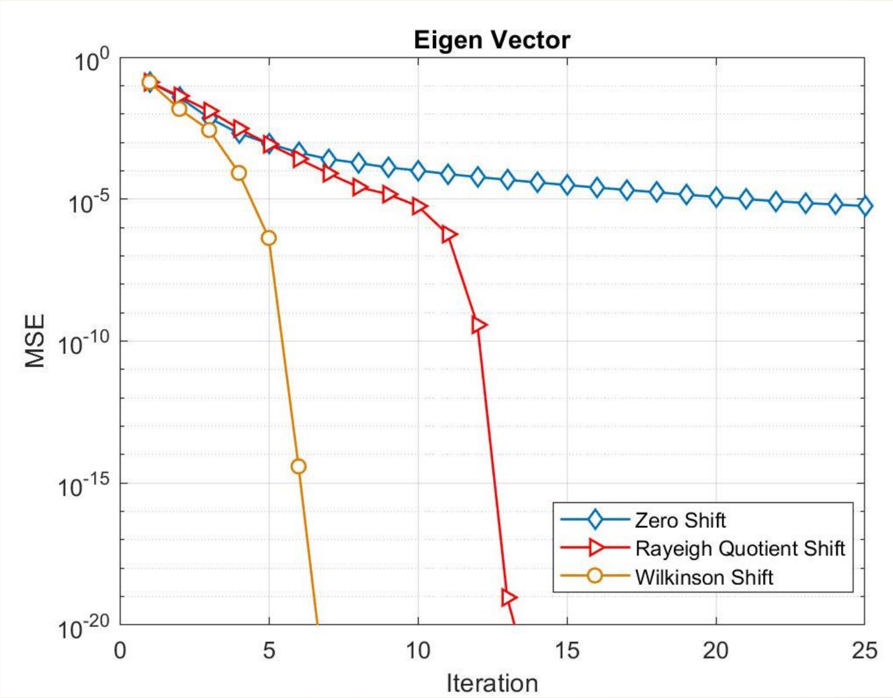
- Deflate to accelerate convergence



Results

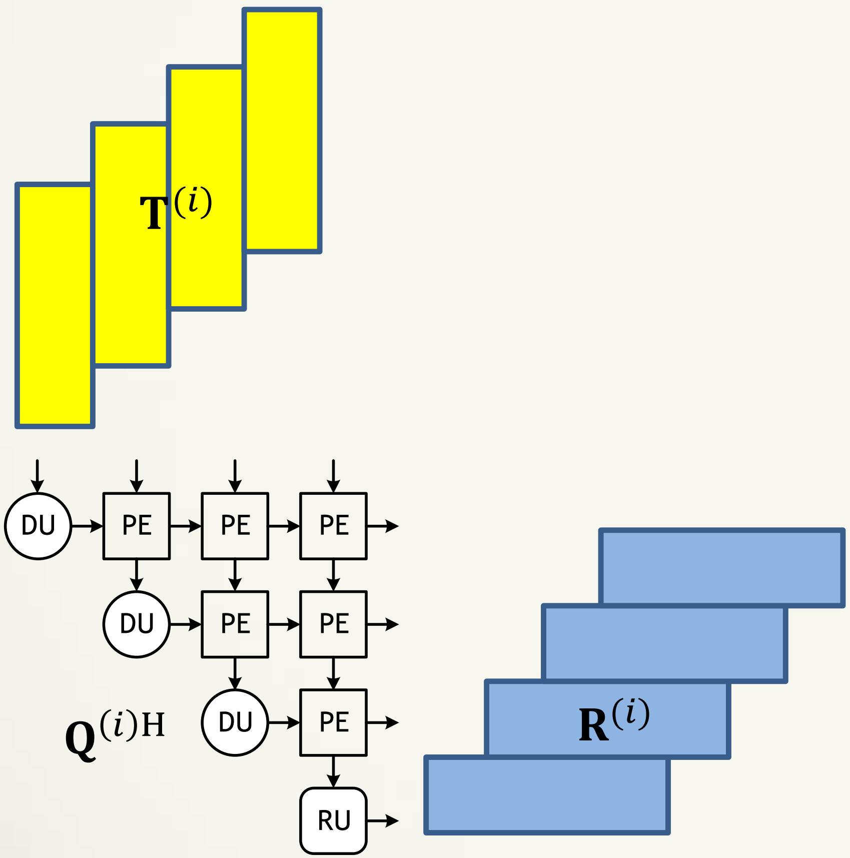
- Use 4x4 complex channel matrix to generate Hermitian Symmetric matrix A
- With deflation

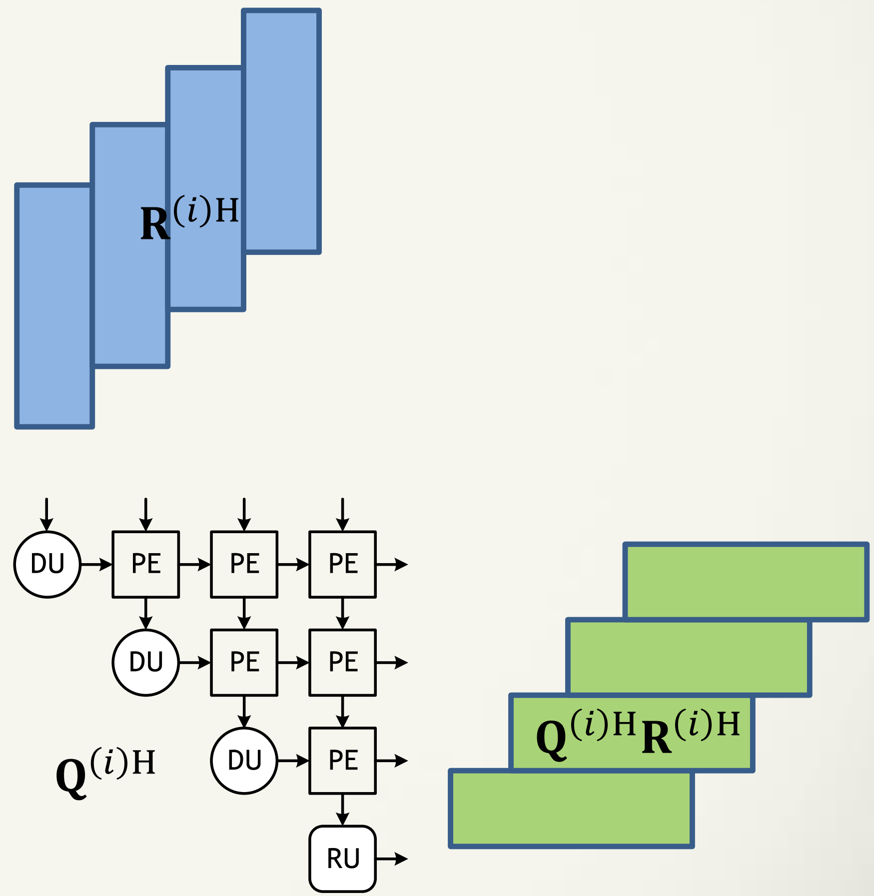




Architecture

- Based on QR systolic array





Singular Value Decomposition

Singular Value Decomposition

- Jacobi algorithm
  - Real Two-sided Jacobi algorithm [2]
  - Complex Two-sided Jacobi algorithm [4]

- Two phase algorithm
  - Golub-Kahan algorithm [3]

Singular Value Decomposition

- For real/complex matrix $\mathbf{A} \in \mathbb{C}^{\mathrm{M}\times\mathrm{N}}$
  - $\mathbf{A} = \mathbf{U\Sigma V}^{\mathrm{H}}$
    - $\mathbf{\Sigma} \in \mathbb{R}^{\mathrm{M}\times\mathrm{N}}$ is a real non-negative diagonal matrix with entries, known as singular values. Usually, they are arranged in descending order.

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & & & & 0 \\ 0 & \sigma_1 & 0 & & & 0 \\ & 0 & \ddots & 0 & & 0 \\ 0 & & 0 & \sigma_M & 0 & 0 \end{bmatrix}$$

    - $\mathbf{U} \in \mathbb{C}^{\mathrm{M}\times\mathrm{M}}$ is a real/complex unitary matrix. The column of $\mathbf{U}$ is called left singular vector
    - $\mathbf{V} \in \mathbb{C}^{\mathrm{N}\times\mathrm{N}}$ is a real/complex unitary matrix. The column of $\mathbf{V}$ is called right singular vector

Real Two-Sided Jacobi Algorithm (1/2)

- **Diagonalization**
- **For symmetric real matrix $\mathbf{A} \in \mathbb{R}^{2\times 2}$**

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \qquad \mathbf{J} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

$$\mathbf{B} = \mathbf{J}^T\mathbf{A}\mathbf{J} = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}^T \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$$

If $b_{1,2} = b_{2,1} = 0 = cs\left(a_{1,1} - a_{2,2}\right) + a_{1,2}(c^2 - s^2)$

we have

$$\boxed{\rho = \frac{a_{2,2} - a_{1,1}}{2a_{1,2}} \qquad tan\theta = \frac{sgn(\rho)}{|\rho| + \sqrt{1+\rho^2}}}$$

Real Two-Sided Jacobi Algorithm (2/2)

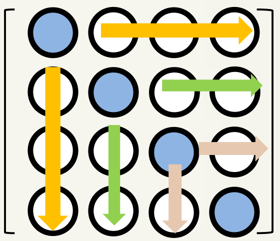
Algorithm: Real Two-Sided Jacobi

Given real symmetric matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$

1. **for** $sweep = 1{:}S$
2. **for** $p = 1{:}N-1$
3. **for** $q = p{:}N$
4. $\rho = \frac{a_{q,q} - a_{p,p}}{2a_{p,q}}$
5. $\theta = \tan^{-1}\left(\frac{sgn(\rho)}{|\rho| + \sqrt{1+\rho^2}}\right),$
6. $\mathbf{J} = \mathbf{I}_N,\ \mathbf{J}_{p,p} = \mathbf{J}_{q,q} = cos\theta,$
7. $\mathbf{J}_{p,q} = -\mathbf{J}_{q,p} = sin\theta$
8. $\mathbf{A} = \mathbf{J}^T\mathbf{A}\mathbf{J}$
9. $\mathbf{V} = \mathbf{V}\mathbf{J}$
10. **end**
11. **end**

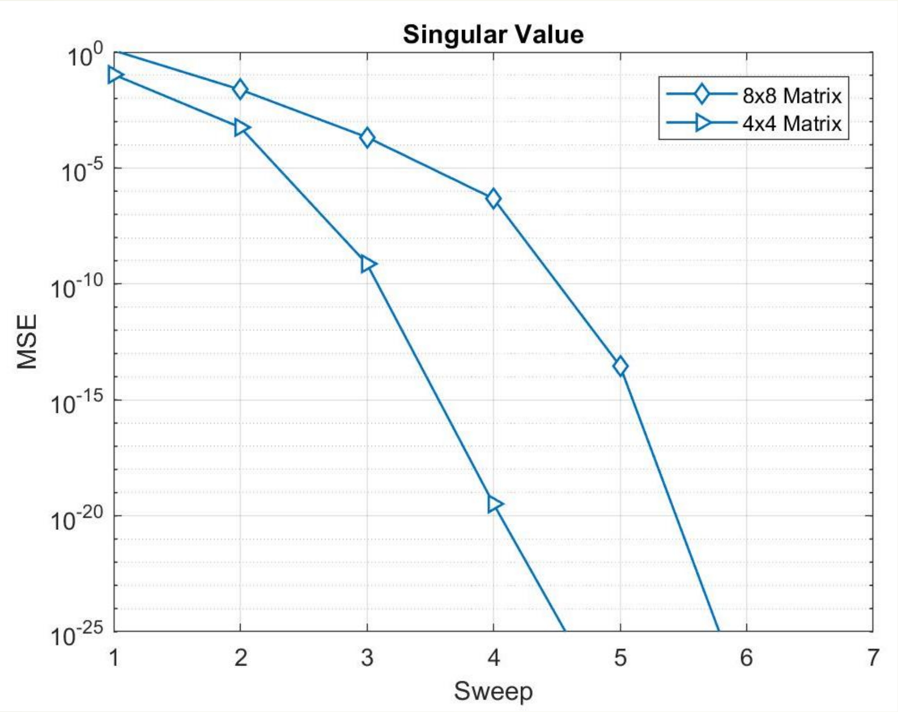
end

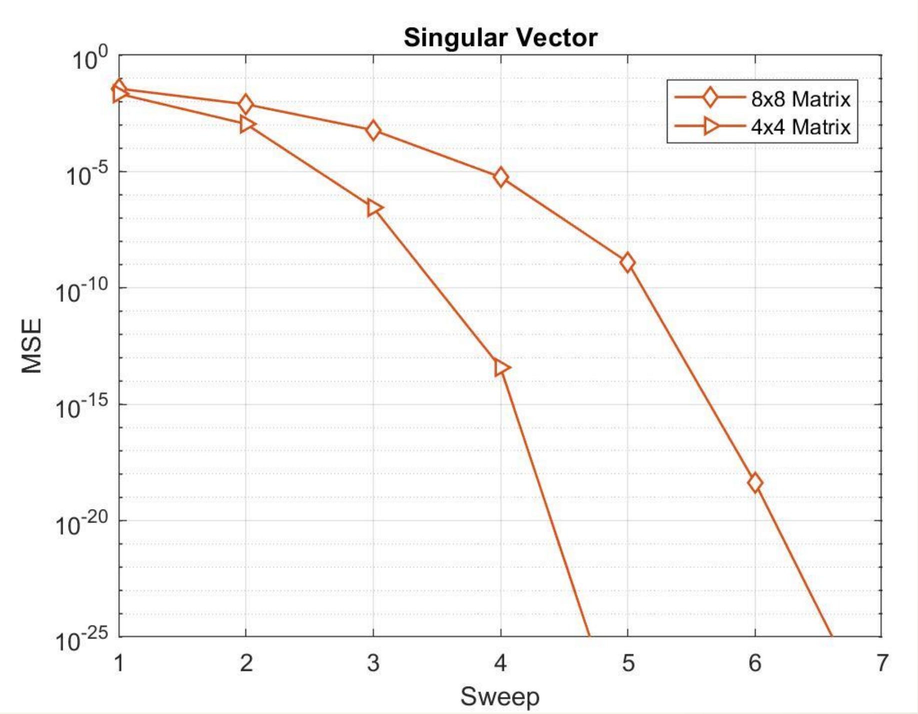
One sweep



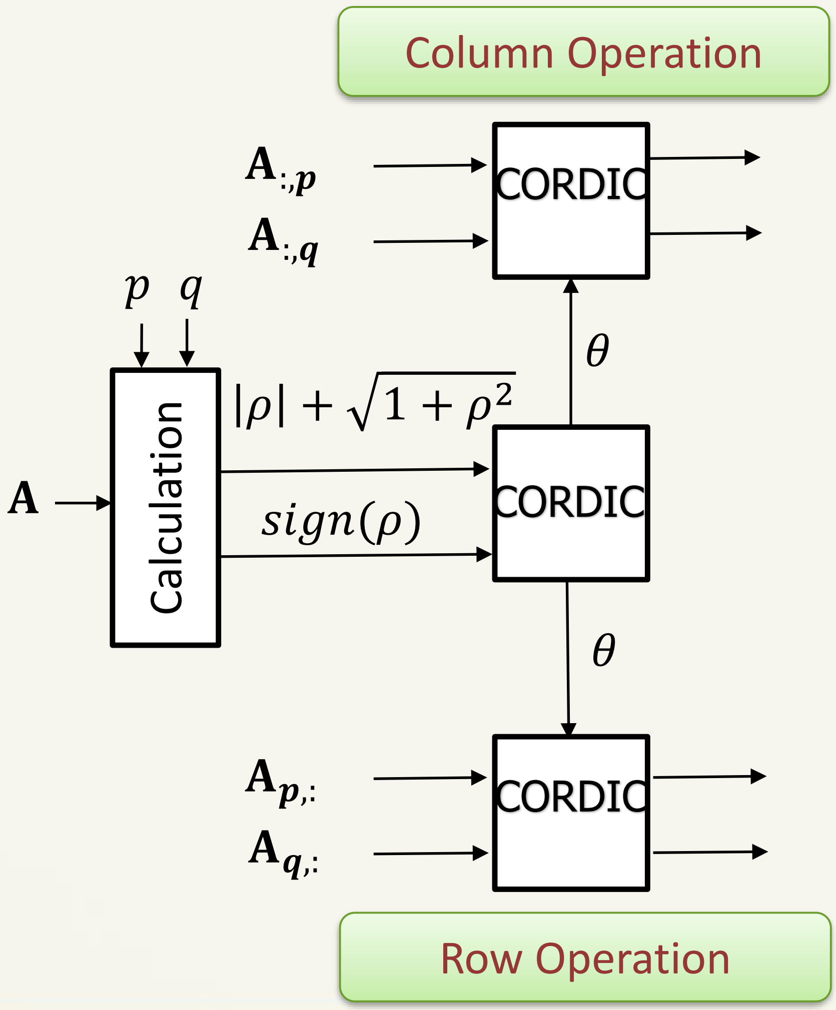
Performance

- Fast convergence but large complexity for each sweep.





Operations of Real Two-Sided Jacobi Algorithm



Complex Two-Sided Jacobi Algorithm (1/3)

- Two-step two-sided rotation [4]
  - Step 1: upper-triangularization

$$\mathbf{C}=\begin{bmatrix} a_{1,1}e^{j\theta_{1,1}} & a_{1,2}e^{j\theta_{1,2}} \\ a_{2,1}e^{j\theta_{2,1}} & a_{2,2}e^{j\theta_{2,2}} \end{bmatrix}$$

$$\begin{bmatrix} a_{1,1}e^{j\theta_{1,1}} & a_{1,2}e^{j\theta_{1,2}} \\ a_{2,1}e^{j\theta_{2,1}} & a_{2,2}e^{j\theta_{2,2}} \end{bmatrix}\cdot\begin{bmatrix} e^{j\gamma_1} & 0 \\ 0 & e^{j\varepsilon_1} \end{bmatrix}\begin{bmatrix} cos\varphi & sin\varphi \\ -sin\varphi & cos\varphi \end{bmatrix}=\begin{bmatrix} \bar{a}_{1,1}e^{j\bar{\theta}_{1,1}} & \bar{a}_{1,2}e^{j\bar{\theta}_{1,2}} \\ 0 & \bar{a}_{2,2} \end{bmatrix}$$

$$\mathbf{C}\cdot\mathbf{D}(\mathbf{C}_{2,1},\mathbf{C}_{2,2})\mathbf{G}(\mathbf{C}_{2,1},\mathbf{C}_{2,2})=\bar{\mathbf{C}}$$

$$\gamma_1=-\theta_{2,1},\varepsilon_1=-\theta_{2,2} \qquad \varphi=tan^{-1}(\frac{a_{2,1}}{a_{2,2}})$$

$$\begin{bmatrix} e^{j\alpha_1} & 0 \\ 0 & e^{j\beta_1} \end{bmatrix}\begin{bmatrix} \bar{a}_{1,1}e^{j\bar{\theta}_{1,1}} & \bar{a}_{1,2}e^{j\bar{\theta}_{1,2}} \\ 0 & \bar{a}_{2,2} \end{bmatrix}\begin{bmatrix} e^{j\alpha_2} & 0 \\ 0 & e^{j\beta_2} \end{bmatrix}=\begin{bmatrix} r_{1,1} & r_{1,2} \\ 0 & r_{2,2} \end{bmatrix}$$

$$\mathbf{D}_l(\bar{\mathbf{C}}_{1,1},\bar{\mathbf{C}}_{1,2})\bar{\mathbf{C}}\mathbf{D}_r(\bar{\mathbf{C}}_{1,1},\bar{\mathbf{C}}_{1,2})=\mathbf{R}$$

$$\alpha_1=-\frac{\bar{\theta}_{1,1}+\bar{\theta}_{1,2}}{2} \qquad \beta_1=\alpha_2=-\beta_2=\frac{\bar{\theta}_{1,2}-\bar{\theta}_{1,1}}{2}$$

Complex Two-Sided Jacobi Algorithm (2/3)

- Step 2: diagonalization

$$\mathbf{R} = \begin{bmatrix} r_{1,1} & r_{1,2} \\ 0 & r_{2,2} \end{bmatrix}$$

$$\begin{bmatrix} cos\theta_1 & sin\theta_1 \\ -sin\theta_1 & cos\theta_1 \end{bmatrix} \cdot \begin{bmatrix} r_{1,1} & r_{1,2} \\ 0 & r_{2,2} \end{bmatrix} \cdot \begin{bmatrix} cos\theta_2 & -sin\theta_2 \\ sin\theta_2 & cos\theta_2 \end{bmatrix} = \begin{bmatrix} \delta_1 & 0 \\ 0 & \delta_2 \end{bmatrix}$$

$$\mathbf{G}_l(\mathbf{R})\mathbf{R}\mathbf{G}_r(\mathbf{R}) = \mathbf{\Sigma}$$

$$\boxed{\theta_1 = \frac{1}{2} \cdot \left(\tan^{-1}\left(\frac{r_{1,2}}{r_{1,1} - r_{2,2}} \right) + \tan^{-1}\left(\frac{-r_{1,2}}{r_{1,1} + r_{2,2}} \right) \right)}$$

$$\boxed{\theta_2 = \frac{1}{2} \cdot \left(\tan^{-1}\left(\frac{r_{1,2}}{r_{1,1} - r_{2,2}} \right) - \tan^{-1}\left(\frac{-r_{1,2}}{r_{1,1} + r_{2,2}} \right) \right)}$$

Complex Two-Sided Jacobi Algorithm (3/3)

Algorithm: Complex Two-Sided Jacobi

Given Complex matrix $\mathbf{C} \in \mathbb{C}^{N \times N}$

1. for $sweep = 1: S$
2. for $p = 1: N-1$
3. for $q = p: N$
4. $\bar{\mathbf{C}} = \mathbf{C} \cdot \widetilde{\mathbf{D}}(\mathbf{C}_{q,p}, \mathbf{C}_{q,q})\tilde{\mathbf{G}}(\mathbf{C}_{q,p}, \mathbf{C}_{q,q})$
5. $\mathbf{V} = \mathbf{V}\widetilde{\mathbf{D}}(\mathbf{C}_{q,p}, \mathbf{C}_{q,q})\tilde{\mathbf{G}}(\mathbf{C}_{q,p}, \mathbf{C}_{q,q})$
6. $\mathbf{R} = \widetilde{\mathbf{D}}_l(\bar{\mathbf{C}}_{p,p}, \bar{\mathbf{C}}_{p,q})\bar{\mathbf{C}}\widetilde{\mathbf{D}}_r(\bar{\mathbf{C}}_{p,p}, \bar{\mathbf{C}}_{p,q})$
7. $\mathbf{V} = \mathbf{V}\widetilde{\mathbf{D}}_r(\bar{\mathbf{C}}_{p,p}, \bar{\mathbf{C}}_{p,q}), \mathbf{U}^H = \widetilde{\mathbf{D}}_l(\bar{\mathbf{C}}_{p,p}, \bar{\mathbf{C}}_{p,q})\mathbf{U}^H$
8. $\mathbf{C} = \tilde{\mathbf{G}}_l(\mathbf{R}_{[p,q],[p,q]})\mathbf{R}\tilde{\mathbf{G}}_r(\mathbf{R}_{[p,q],[p,q]})$
9. $\mathbf{V} = \mathbf{V}\tilde{\mathbf{G}}_r(\mathbf{R}_{[p,q],[p,q]}), \mathbf{U}^H = \tilde{\mathbf{G}}_l(\mathbf{R}_{[p,q],[p,q]})\mathbf{U}^H$
10. end
11. end

end

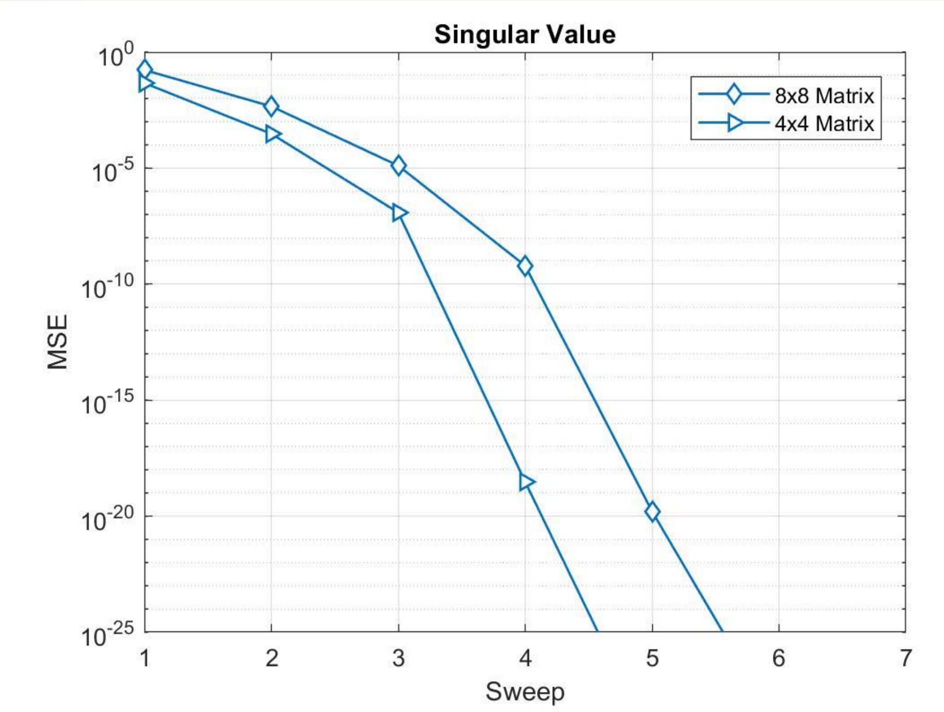
Extend the dimension of rotation matrixes

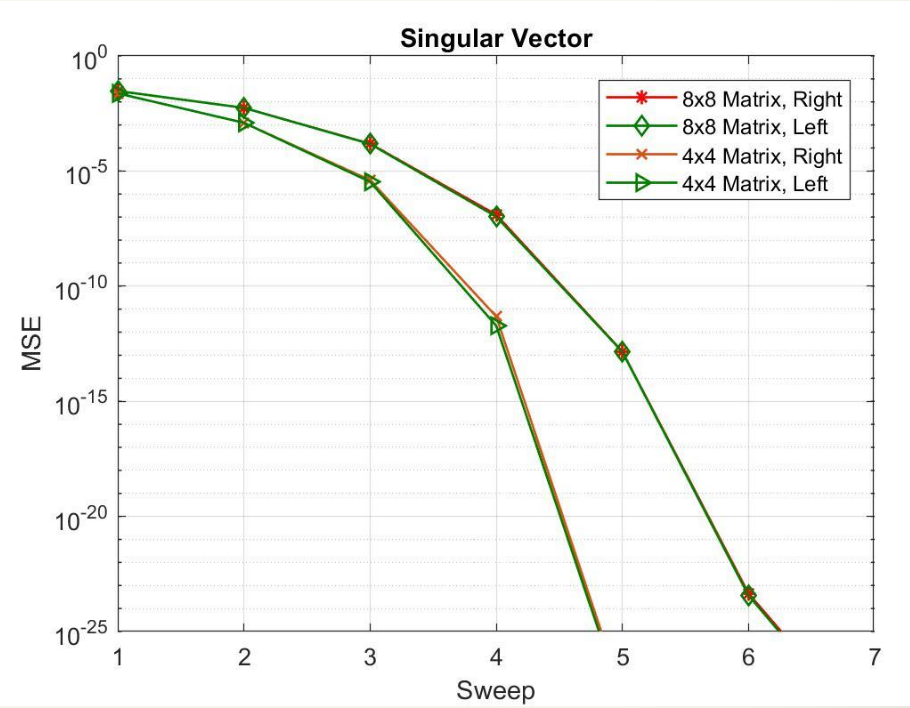
$$\widetilde{\mathbf{D}} = \mathbf{I}_{\boldsymbol{N}}$$

$$\widetilde{\mathbf{D}}_{[\boldsymbol{p},\boldsymbol{q}],[\boldsymbol{p},\boldsymbol{q}]}(\cdot,\cdot) = \mathbf{D}(\cdot,\cdot)$$

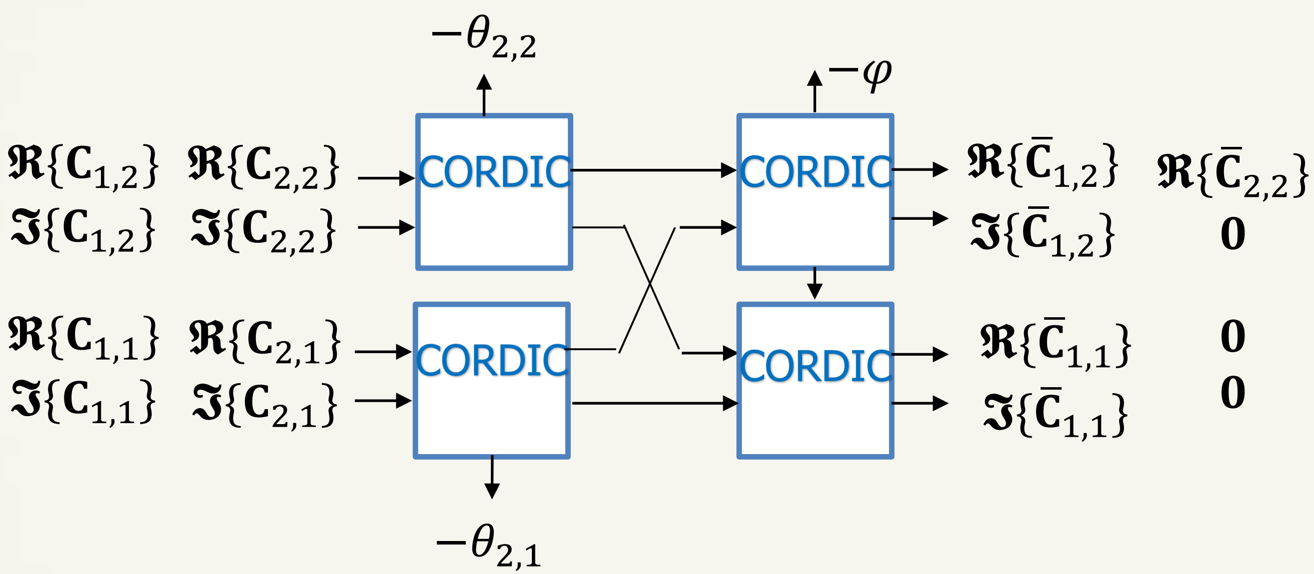
Performance

- Fast convergence but large complexity for each sweep.



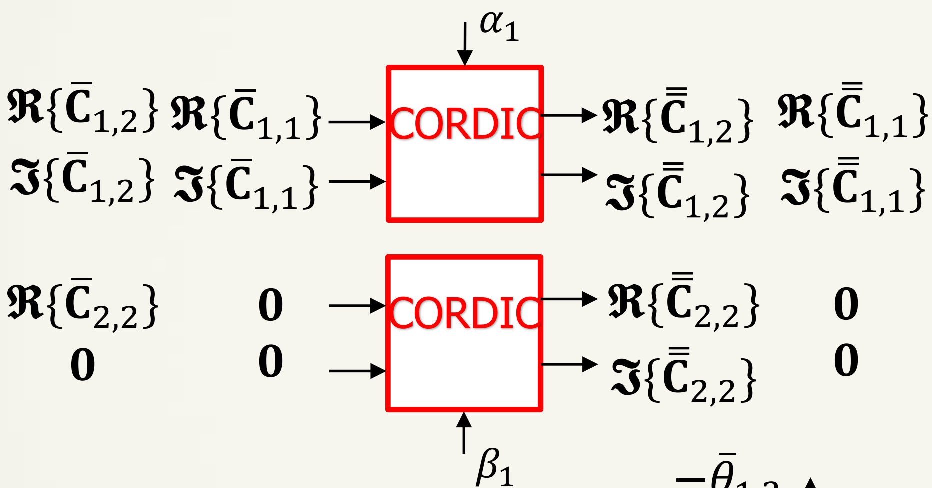


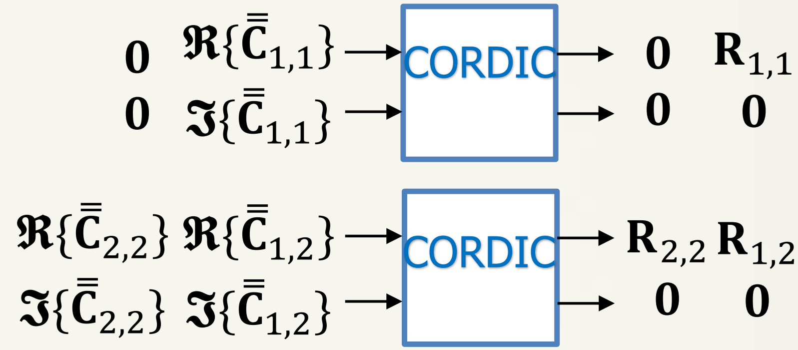
Operations of Complex Two-Sided Jacobi Algorithm (1/3)

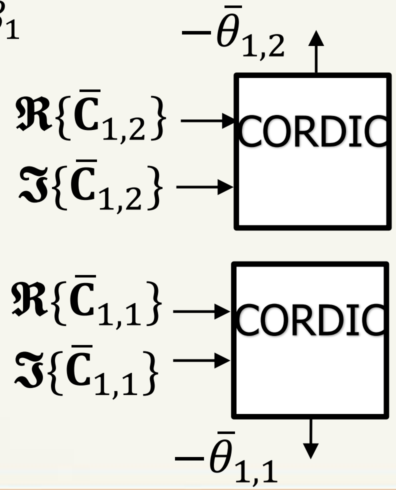


$$\mathbf{C} \cdot \mathbf{D}(\mathbf{C}_{2,1}, \mathbf{C}_{2,2})\mathbf{G}(\mathbf{C}_{2,1}, \mathbf{C}_{2,2}) = \bar{\mathbf{C}}$$

Operations of Complex Two-Sided Jacobi Algorithm (2/3)

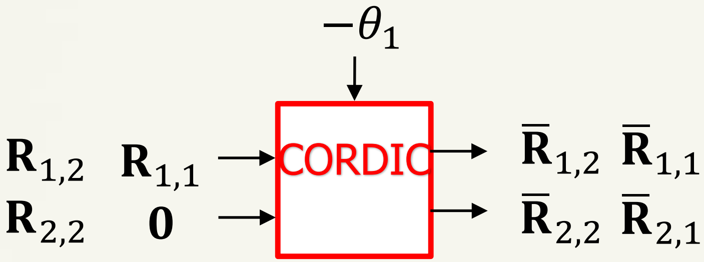


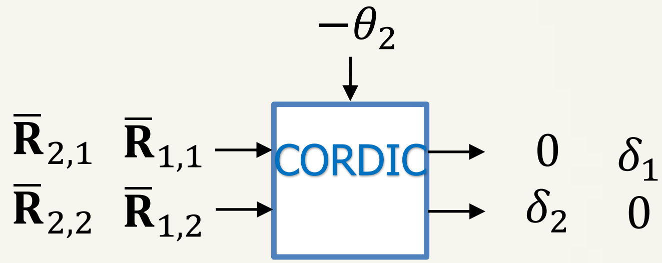




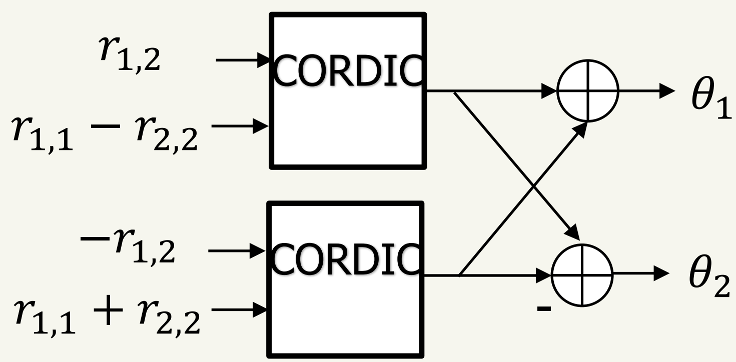
$$\mathbf{D}_l(\bar{\mathbf{C}}_{1,1}, \bar{\mathbf{C}}_{1,2})\bar{\mathbf{C}}\mathbf{D}_r(\bar{\mathbf{C}}_{1,1}, \bar{\mathbf{C}}_{1,2}) = \mathbf{R}$$

Operations of Complex Two-Sided Jacobi Algorithm (3/3)



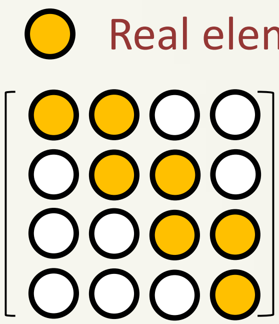


$$\mathbf{G}_l(\mathbf{R})\mathbf{R}\mathbf{G}_r(\mathbf{R}) = \mathbf{\Sigma}$$

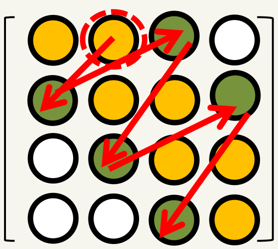


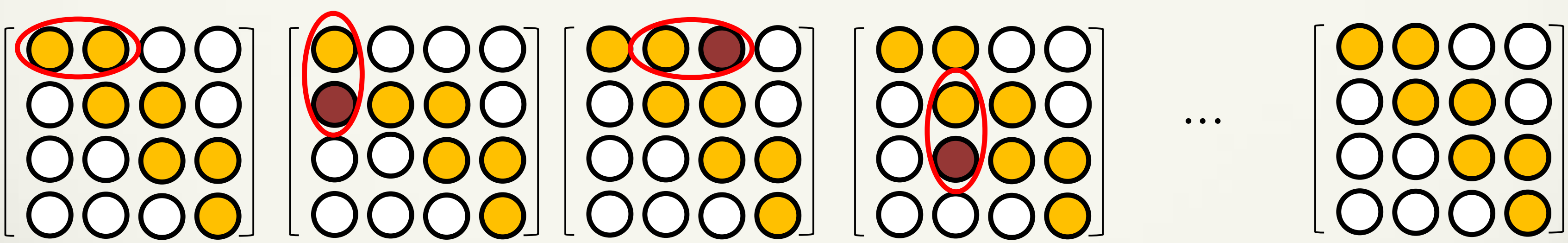
Two-Phase Algorithm

- Bidiagonalization
  - $\mathbf{H} = \mathbf{QBP}^{\mathrm{H}}$
  - $\mathbf{B}$ is an upper bidiagonal matrix with real elements



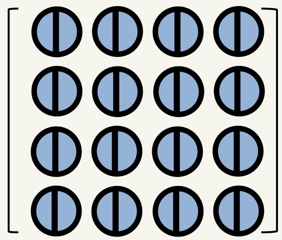
- Implicit QR algorithm (Golub-Kahan Algorithm)
  - Diagonalization
    - Chasing
    - Eliminating the bulge



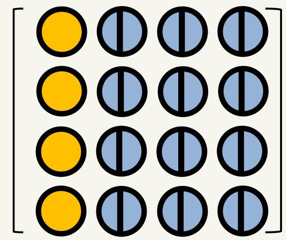


Bidiagonalization (1/2)

$$\begin{bmatrix} e^{-j\theta_1} & 0 & 0 & 0 \\ 0 & e^{-j\theta_2} & 0 & 0 \\ 0 & 0 & e^{-j\theta_3} & 0 \\ 0 & 0 & 0 & e^{-j\theta_4} \end{bmatrix}$$

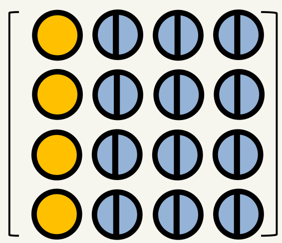


$=$

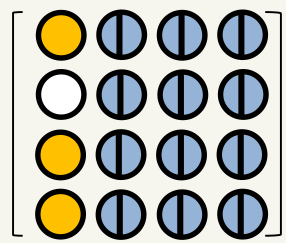


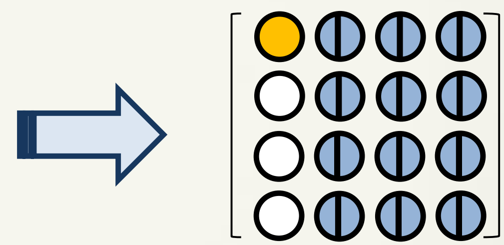
 Complex element  Real element

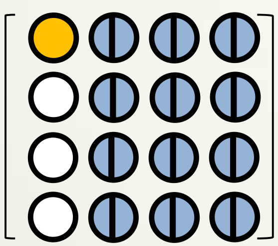
$$\begin{bmatrix} cos\theta_5 & sin\theta_5 & 0 & 0 \\ -sin\theta_5 & cos\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$=$

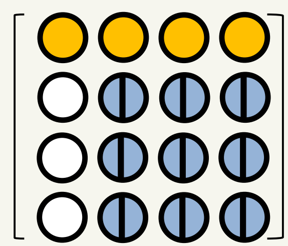




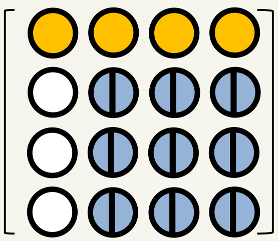


$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-j\theta_8} & 0 & 0 \\ 0 & 0 & e^{-j\theta_9} & 0 \\ 0 & 0 & 0 & e^{-j\theta_{10}} \end{bmatrix}$$

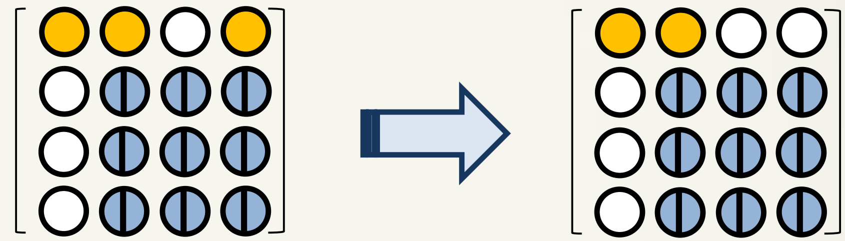
$=$



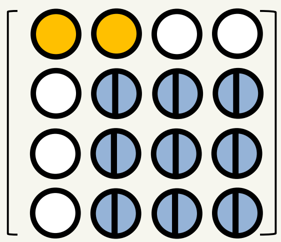
Bidiagonalization (2/2)



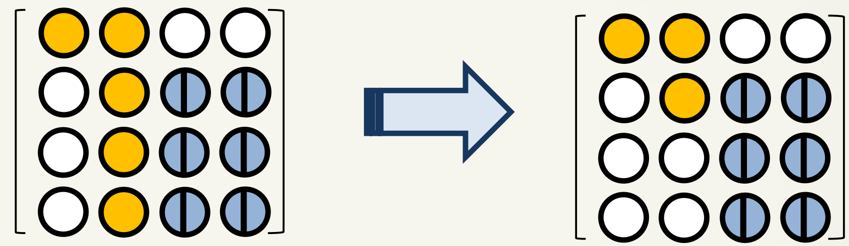
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta_{11} & -sin\theta_{11} & 0 \\ 0 & sin\theta_{11} & cos\theta_{11} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

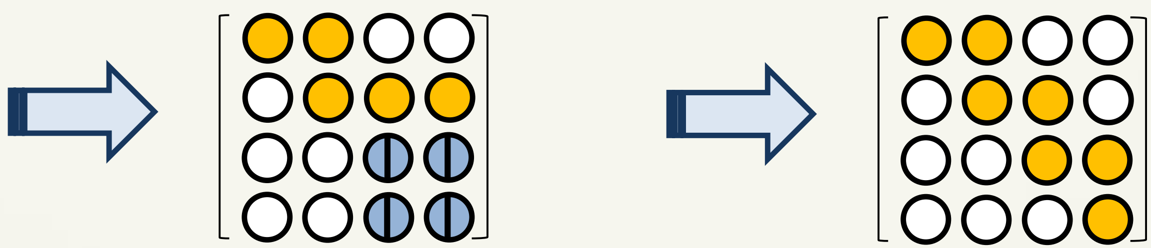


$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-j\theta_{12}} & 0 & 0 \\ 0 & 0 & e^{-j\theta_{13}} & 0 \\ 0 & 0 & 0 & e^{-j\theta_{14}} \end{bmatrix}$$



$$=$$





Golub-Kahan SVD Algorithm

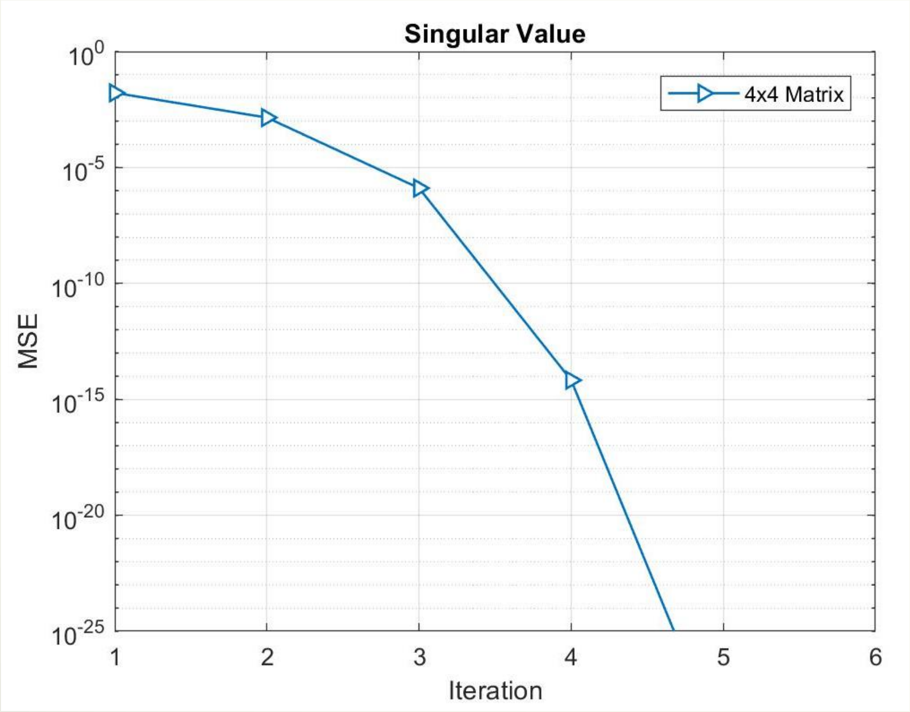
Algorithm: Golub-Kahan Algorithm

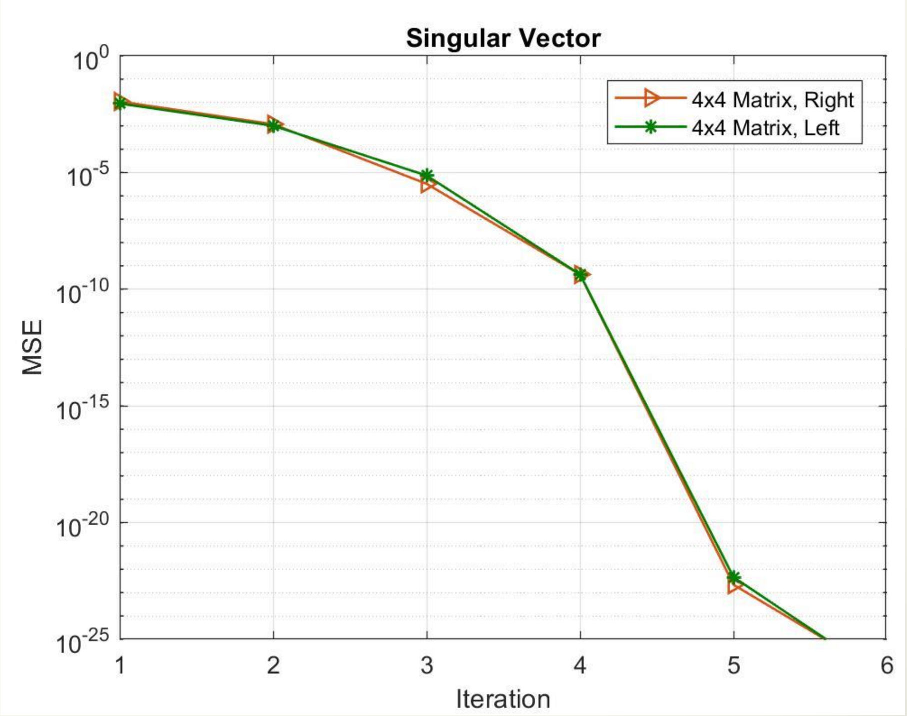
Given bidiagonal matrix $\mathbf{B} \in \mathbb{R}^{N \times N}$ with nonzero bidiagonal elements

While $(N > 1)$
    $\mathbf{T} = \mathbf{B}^T\mathbf{B}$, μ is the eigenvalue of $\mathbf{T}_{N-1:N,N-1:N}$ close to $\mathbf{T}_{N,N}$
    for $l = 1 : N-1$
        if $(l == 1)$ // implicit shifted QR
            $\alpha = \left(\mathbf{B}_{1,1}\right)^2 - \mu, \quad \beta = \mathbf{B}_{1,1}\mathbf{B}_{1,2}$
        else
            $\alpha = \mathbf{B}_{l-1,l}, \quad \beta = \mathbf{B}_{l-1,l+1}$
        end
        $\theta = tan^{-1}\left(\frac{\beta}{\alpha}\right)$, $\mathbf{B}_{:,l:l+1} = \mathbf{B}_{:,l:l+1}\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$
        $\alpha = \mathbf{B}_{l,l}, \quad \beta = \mathbf{B}_{l+1,l}$
        $\theta = tan^{-1}\left(\frac{\beta}{\alpha}\right)$, $\mathbf{B}_{l:l+1,:} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}\mathbf{B}_{l:l+1,:}$
    end
    if $|\mathbf{B}_{N-1,N}| < \delta$ // deflate
        $\mathbf{B} = \mathbf{B}_{1:N-1,1:N-1}$, $N = N-1$
    end
end

Performance of Golub-Kahan Algorithm

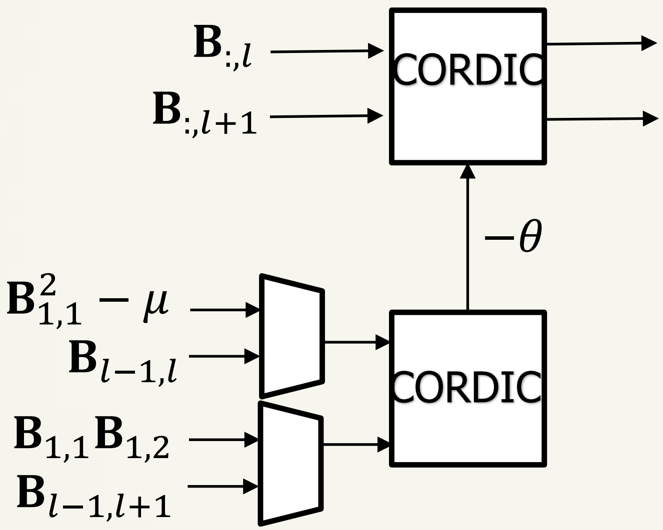
- The complexity of iteration is low because of sparse real elements. However, the complexity of bidiagonalization is high.
- Wilkinson shift can accelerate convergence.



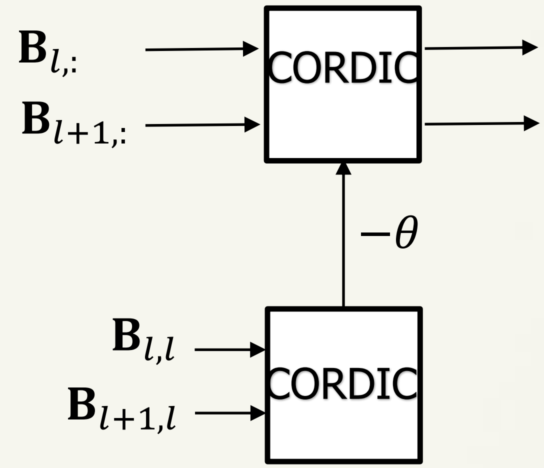


Operations of GK SVD

Column Operation



Row Operation



Summary

- Usually, iterative algorithms are required for EVD and SVD
- Shift and deflation is helpful to accelerate convergence.
- CORDIC is effective to handle Givens rotation for matrix decomposition.
- Tradeoff is required between performance and complexity.

Reference

[1] Zheng-Yu Huang and Pei-Yun Tsai, "Efficient Implementation of QR Decomposition for Gigabit MIMO-OFDM Systems," IEEE Transactions on Circuits and Systems I: Regular paper, vol. 58, pp. 2531-2542, Oct. 2011.

[2] R. Brent, F. Luk, and C. Van Loan, "Computation of the singular value decomposition using mesh-connected processors," Dept. Comput. Science, Cornell Univ., Ithaca, NY, USA Tech. Rep., 1983. [Online]. Available: http://hdl.handle.net/1813/6367

[3] G. H. Golub and C. F. Van Loan, *Matrix Computation*, 4th ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 2013.

[4] C.-H. Yang, C.-W. Chou, C.-S. Hsu, and C.-E. Chen, "A systolic array based GTD processor with a parallel algorithm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 4, pp. 1099–1108, Apr. 2015.