Subject: Introduction to Computing

Topic: Matlab @ Matrix Manipulation

Teacher: Dr. Ajeet Kumar

Basics of Matrix

- **Special Matrices**
  - Identity Matrix
  - Zero Matrix
  - One's Matrix
- **Important Matrix Operation**
  - diag
  - rot90
  - fliplr
  - flipud
  - tril
  - triu
  - reshape

Special Matrices

- **Identity Matrix:** A square matrix (n×n) is known as Identity matrix whose all diagonal elements are "1" and off-diagonal elements are "0".
- It is usually represented by notation I_n and satisfy following condition
- $AI_n = I_nA = A$ where A is $n \times n$ matrix and I_n is $n \times n$ identity matrix
- *Used frequently in matrix algebra wherever applicable*

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Special Matrices

Syntax:

- **eye => returns the scalar, 1.**

```
>> I =eye
   I =
        1
```

- **eye (m)=> creates identity matrix of (m×m)**

```
>> I =eye (2)
   I =
        1     0
        0     1
```

Special Matrices

Syntax:

- **eye(m,n)=> creates a matrix of m-by-n with ones on the main diagonal and zeros elsewhere**

```
>> I = eye(2,3)
     I =
          1     0     0
          0     1     0
```

- **eye(size(A)) => creates a identity matrix of size of matrix A**

```
>> A=[1 2; 3 4];
>> P= eye(size(A))
     P =
          1     0
          0     1
```

Special Matrices

- **Zero Matrix:** A matrix having all elements equal to “o” is called Zero Matrix. *Used frequently in matrix algebra wherever applicable*

$$I = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Special Matrices

Syntax:

- **zeros => returns the scalar, 0.**

```
>> I =zeros
   I =
       0
```

- **zeros(m)=> creates zero matrix of (m×m)**

```
>> I = zeros (2)
   I =
       0     0
       0     0
```

- **zeros (m,n)=> creates a matrix of m-by-n with all elements "0"**

```
>> I = zeros(2,3)
   I =
       0     0     0
       0     0     0
```

Special Matrices

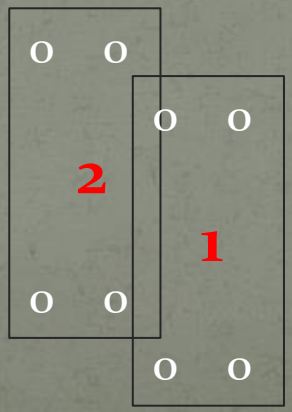
Syntax:

- **zeros (size(A)) => creates a zero matrix of the size of matrix "A"**

```
>> A=[1 2; 3 4];
>> P=zeros(size(A))
    P =
        0     0
        0     0
```

- **zeros(m,n,p) => creates 3-D matrix of having all elements zero**

```
>> A=zeros(2,2,2)
    A(:,:,1) =
        0     0
        0     0

    A(:,:,2) =
        0     0
        0     0
```



Special Matrices

- **One's Matrix:** A matrix having all elements equal to "1" is called one's Matrix. *Used frequently in matrix algebra wherever applicable*

$$I = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Special Matrices

Syntax:

- **ones => returns the scalar, 1.**

```
>> I =ones
     I =
          1
```

- **ones(m)=> creates a matrix of (m×m) whose all elements are "1"**

```
>> I = ones (2)
     I =
          1    1
          1    1
```

- **ones (m,n)=> creates a matrix of m×n whose all elements are "1"**

```
>> I = ones(2,3)
     I =
          1    1    1
          1    1    1
```

Special Matrices

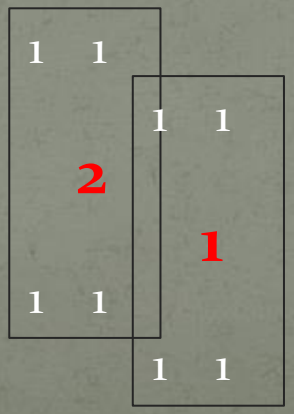
Syntax:

- **ones (size(A)) => creates a one's matrix of the size of matrix "A"**

```
>> A=[1 2; 3 4];
>> P=ones(size(A))
    P =
      1     1
      1     1
```

- **ones(m,n,p) => creates 3-D matrix of having all elements "1"**

```
>> A=ones(2,2,2)
   A(:,:,1) =
        1     1
        1     1

   A(:,:,2) =
        1     1
        1     1
```



Important Matrix Operations

- **diag**

diag (A)=> returns column vector extract of the elements on the diagonal of matrix "A"

```
>> A=[1 2 3; 4 5 6; 7 8 9];     % square matrix
>> p=diag(A)
    p =
         1
         5
         9
>> A=[1 2 3; 4 5 6];  % rectangular matrix
>> q=diag(A)
    q =
         1
         5
```

Important Matrix Operations

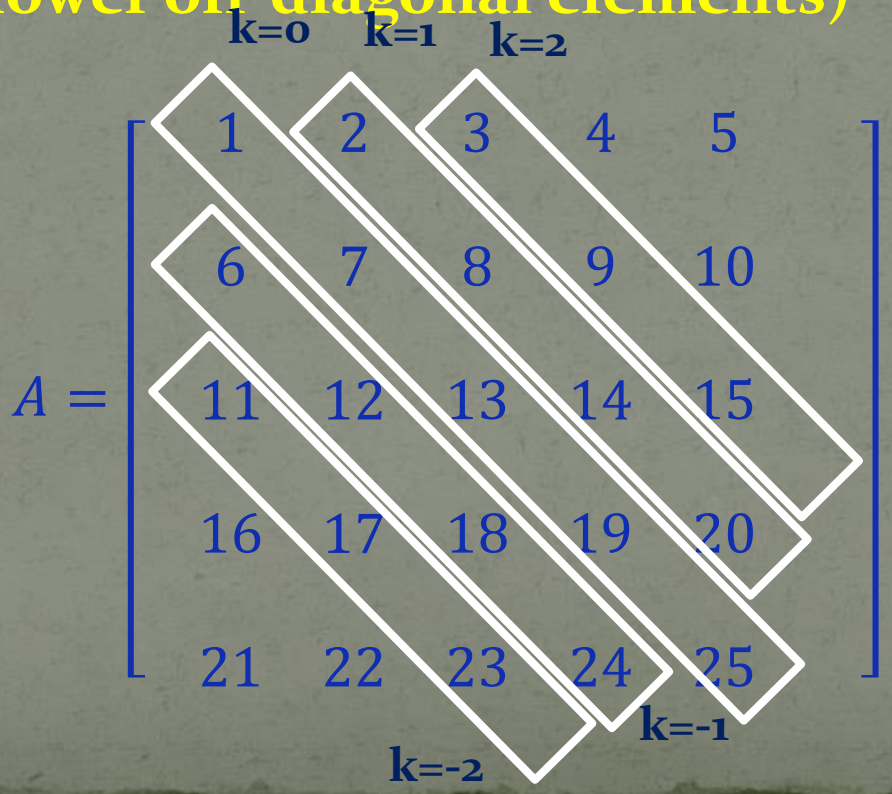
- **diag**

diag (A,k)=> returns a column vector of the elements on the k^{th} diagonal of A.

k=o (diagonal elements) [Default, discussed in previous slide]

k=+ve (k^{th} upper off-diagonal elements)

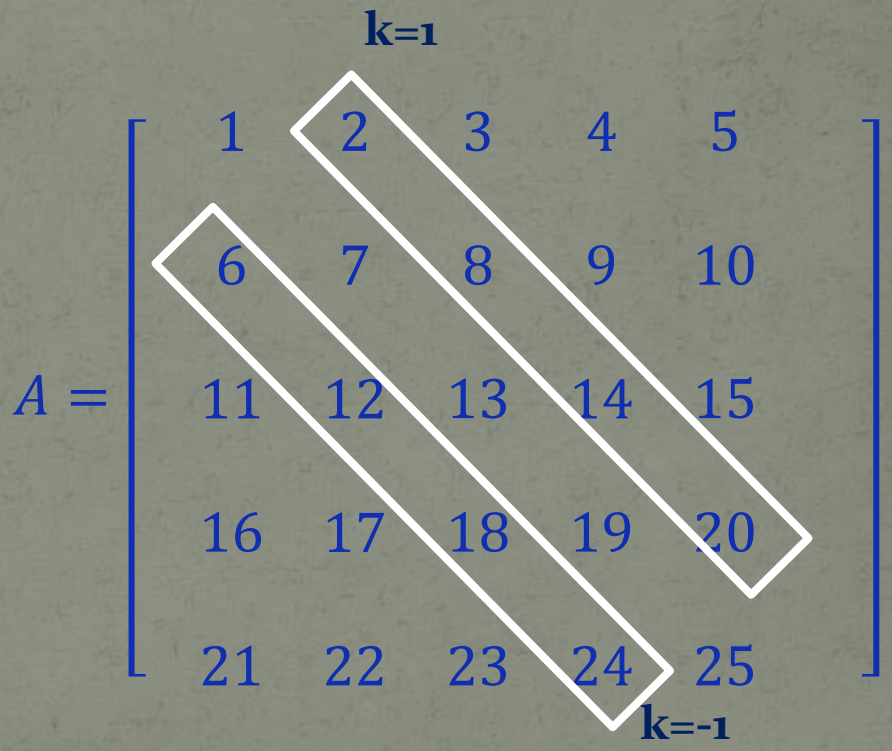
k=-ve (k^{th} lower off-diagonal elements)



Important Matrix Operations

- **diag**

```
>> A=[1 2 3 4 5; 6 7 8 9 10; 11 12 13 14 15; 16 17 18 19 20; 21 22 23 24 25];
>> u=diag(A,1)

u =
     2
     8
    14
    20
>> v=diag(A,-1)

v=

     6
    12
    18
    24
```



Important Matrix Operations

- **diag**

diag (v)=> create a square matrix whose diagonal elements are the elements of the vector "v" and rest of the elements are zero

```
>> v=[1 2 3];
>> P=diag(v)

P =

     1     0     0
     0     2     0
     0     0     3
```

Important Matrix Operations

- **rot90**

rot90 (A)=> rotates the matrix anti-clockwise by 90°

```
>> A=[1 2 3; 4 5 6; 7 8 9];
>> P=rot90(A)
```

```
P =
     3     6     9
     2     5     8
     1     4     7
```

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Important Matrix Operations

- **rot90**

rot90 (A,k)=> rotates the matrix by "k*90°" anti-clockwise for k =1,2,3,4 and clockwise for k=-1,-2,-3 and -4

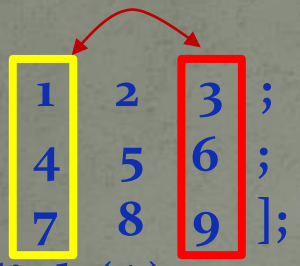
```
>> A=[1 2 3; 4 5 6; 7 8 9];
>> P=rot90(A,3)
P =
    7    4    1
    8    5    2
    9    6    3

>> Q=rot90(A,-3)
Q =

    3    6    9
    2    5    8
    1    4    7
```

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

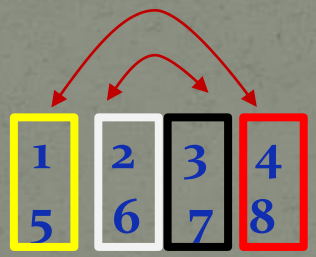
Important Matrix Operations

- **fliplr**

fliplr (A)=> flip the column's of the matrix A from left to right and right to left



```
>> A=[ 1 2 3 ;
       4 5 6 ;
       7 8 9 ];
>> P=fliplr(A)
P =
     3     2     1
     6     5     4
     9     8     7
```



```
>> A=[ 1 2 3 4 ;
       5 6 7 8 ];
>> Q=fliplr(A)
Q =
     4     3     2     1
     8     7     6     5
```

Important Matrix Operations

- **flipud**

flipud (A)=> flip the row's of the matrix A from top to bottom and bottom to top

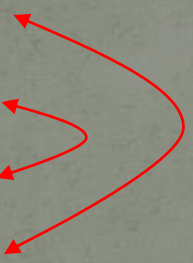
```
>> A=[ 1  2  3 ;
       4  5  6 ;
       7  8  9 ];

>> P=flipud(A)
P =
    7  8  9
    4  5  6
    1  2  3
```

Important Matrix Operations

- **flipud**

flipud (A)=> flip the row's of the matrix A from top to bottom and bottom to top

```
>> A=[1     2;
      3     4;
      5     6;
      7     8];
>> Q=flipud(A)

Q =
    7     8
    5     6
    3     4
    1     2
```



Important Matrix Operations

- **tril**

tril (A)=> returns the lower triangular portion of matrix A

```
>> A=[1 2 3; 4 5 6; 7 8 9]

>> P=tril(A)

P =

     1     0     0
     4     5     0
     7     8     9
```

Important Matrix Operations

- **tril**

tril (A,k)=> returns the elements on and below the k^{th} diagonal of A

```
>> A=[1 2 3; 4 5 6; 7 8 9]
>> P=tril(A,1)
P =
     1     2     0
     4     5     6
     7     8     9
```

k=1

```
>> Q=tril(A,-1)
Q =
     0     0     0
     4     0     0
     7     8     0
```

k=-1

Important Matrix Operations

- **triu**

triu (A)=> returns the upper triangular portion of matrix "A" including diagonal

```
>> A=[1 2 3;
      4 5 6;
      7 8 9]

>> P=triu(A)
P =
    1    2    3
    0    5    6
    0    0    9
```

Important Matrix Operations

- **triu**

triu (A,k)=> returns the elements on and above the k^{th} diagonal of A

```
>> A=[1 2 3; 4 5 6; 7 8 9]
>> P=triu(A,1)
    P =
        0     2     3
        0     0     6
        0     0     0
```

k=1

```
>> Q=triu(A,-1)
Q =
        1     2     3
        4     5     6
        0     8     9
```

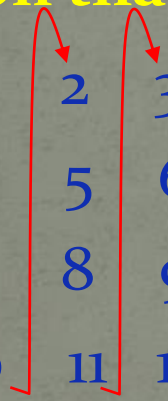
k=-1

Important Matrix Operations

- **reshape**

A is matrix of m-by-n

reshape (A, p,q)=> returns the matrix "A" of p-by-q with condition that m*n = p*q



```
>> A=[ 1   2   3;
       4   5   6;
       7   8   9;
       10  11  12];
>> P=reshape(A,2,6)
P =
    1    7    2    8    3    9
    4   10    5   11    6   12
```

Important Matrix Operations

- **reshape**

```
>> A=[1   2   3;
      4   5   6;
      7   8   9;
      10  11  12];
>> Q=reshape(A,6,2)
Q =
    1    8
    4   11
    7    3
   10    6
    2    9
    5   12
```