AYDIN
ADNAN MENDERES UNIVERSITY
FACULTY OF ENGINEERING
Department of Electrical and Electronics Engineering

EE214 – Computer Aided Engineering Applications

Prof. Dr. Yılmaz KALKAN

EE214 – Computer Aided Engineering Applications

Lecture 3

MATRIX OPERATIONS

Topics

- **Mathematical Operations with Matrices**
  - **Addition and Subtraction**
  - **Multiplication**
  - **Determinants**
  - **Matrix Inverse**

Mathematical Operations with Matrices

Addition and Subtraction

If we have two matrices *A* and *B*, each of the order (*m*×*n*), then

$$A \pm B = \begin{bmatrix} a_{11} \pm b_{11} & a_{12} \pm b_{12} & \cdots & a_{1n} \pm b_{1n} \\ a_{21} \pm b_{21} & a_{22} \pm b_{22} & & \\ \cdots & & \cdots & \\ a_{m1} \pm b_{m1} & & & a_{mn} \pm b_{mn} \end{bmatrix} \rightarrow (m \times n)$$

The MATLAB expression for matrix addition or subtraction is

W = A + B or W = A – B

Multiplication

If we have an (*m*×*k*) matrix *A* and a (*k*×*n*) matrix *B*, then

$$C = AB = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & & \vdots \\ \vdots & & \ddots & \\ c_{m1} & \cdots & & c_{mn} \end{bmatrix} \rightarrow (m \times n)$$

where

$$c_{lp} = \sum_{j=1}^{k} a_{lj} b_{jp}$$

For the conditions stated, the MATLAB expression for matrix multiplication is

W = A*B

Note: The product of two matrices is defined only when the adjacent integers of their respective orders are equal; *k* in this case. In other words,

$$(m\times k)(k\times n) \rightarrow (m\times n)$$

$$c_{lp} = \sum_{j=1}^{k} a_{lj} b_{jp}$$

where the notation indicates that we have summed the *k* terms.

Matrix Transpose

The transpose of a matrix is obtained as C'

Where

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & & \vdots \\ \vdots & & \ddots & \\ c_{m1} & \cdots & & c_{mn} \end{bmatrix} \rightarrow (m \times n) \qquad C^{\mathrm{T}} = \begin{bmatrix} c_{11} & c_{21} & \cdots & c_{m1} \\ c_{12} & c_{22} & & \vdots \\ \vdots & & \ddots & \\ c_{1n} & \cdots & & c_{mn} \end{bmatrix} \rightarrow (n \times m)$$

It can be shown that if *C* = *AB*, then its transpose is

$$C' = (AB)' = B'A'$$

If *A* is an identity matrix (*A* = *I*) and *m* = *n*, then

$$C = IB = BI = B$$

Example –

We shall multiply the following two matrices and show numerically that the transpose can be obtained either of the two ways indicated previously.

$$A = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \end{bmatrix} \qquad B = \begin{bmatrix} 11 & 12 \\ 21 & 22 \\ 31 & 32 \end{bmatrix}$$

The script is

```
A = [11, 12, 13; 21, 22, 23];
B = [11, 12; 21, 22; 31, 32];
C = A*B
Ctran1 = C'
Ctran2 = B'*A'
```

Upon execution, we obtain

```
C =
         776         812
        1406        1472
Ctran1 =
         776        1406
         812        1472
Ctran2 =
         776        1406
         812        1472
```

```
A = [11, 12, 13; 21, 22, 23];
B = [11, 12; 21, 22; 31, 32];
C = A*B
Ctran1 = C'
Ctran2 = B'*A'
```

Thus, matrix multiplication performs the summation of the series at each combination of values for *x* and *y*.

We now consider three special cases of this general matrix multiplication:

1. **The product of a row and a column vector**
2. **The product of a column and a row vector**
3. **The product of a row vector and a matrix**

Product of a Row Vector and Column Vector

Let a be the row vector

$$a = [a_1\ a_2\ \ldots\ a_k] \rightarrow (1\times k)$$

and b be the column vector

$$b = [b_1\ b_2\ \ldots\ b_k]' \rightarrow (k\times 1)$$

Then the product $d = ab$ is the scalar

$$C = AB = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & & \vdots \\ \vdots & & \ddots & \\ c_{m1} & \cdots & & c_{mn} \end{bmatrix}$$

$$c_{lp} = \sum_{j=1}^{k} a_{lj} b_{jp}$$

$(1\times k)(k\times 1)$

$l = 1$ and $p = 1$

$$d = ab = \begin{bmatrix} a_1 & a_2 & \ldots & a_k \end{bmatrix} \begin{bmatrix} b_1 \\ b_1 \\ \ldots \\ b_k \end{bmatrix} = \left[\sum_{j=1}^{k} a_j b_j \right] = \sum_{j=1}^{k} a_j b_j \rightarrow (1\times 1)$$

This is called the dot product of two vectors.

The MATLAB notation for this operation is either

```
d = a*b
```

or

```
d = dot(a, b)
```

Product of a Column and Row Vector

Let *a* be an (*m*×1) column vector and *b* a (1×*n*) row vector. Then the product *H* = *ba* is

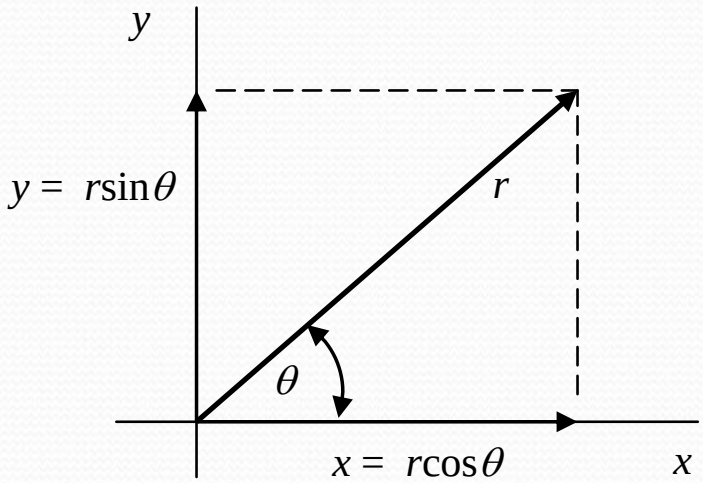
$$H = ba = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_m \end{bmatrix} \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix} = \begin{bmatrix} a_1 b_1 & a_1 b_2 & \dots & a_1 b_n \\ a_2 b_1 & a_2 b_2 & & \dots \\ \dots & & \dots & \\ a_m b_1 & & \dots & a_m b_n \end{bmatrix} \rightarrow (m \times n)$$

The elements of *H*, which are $h_{ij} = b_i a_j$, are the individual products of all the combinations of the elements of *b* and *a*.

$$c_{lp} = \sum_{j=1}^{k} a_{lj} b_{jp}$$

l* = 1, 2, …, *m
p* = 1, 2, …, *n

Example - Polar to Cartesian Coordinates



Consider a vector of radial values

$$r = [r_1\ r_2\ \ldots\ r_m]$$

and a vector of angular values

$$\theta = [\theta_1\ \theta_2\ \ldots\ \theta_n]$$

then

$$X = r' * \cos(\theta) = \begin{bmatrix} r_1 \\ r_2 \\ \ldots \\ r_m \end{bmatrix} \begin{bmatrix} \cos\theta_1 & \cos\theta_2 & \ldots & \cos\theta_n \end{bmatrix}$$

$$= \begin{bmatrix} r_1\cos\theta_1 & r_1\cos\theta_2 & \ldots & r_1\cos\theta_n \\ r_2\cos\theta_1 & r_2\cos\theta_2 & & \\ \ldots & & \ldots & \\ r_m\cos\theta_1 & & \ldots & r_m\cos\theta_n \end{bmatrix}$$

and

$$Y = r' * \sin(\theta) = \begin{bmatrix} r_1 \\ r_2 \\ \dots \\ r_m \end{bmatrix} \begin{bmatrix} \sin\theta_1 & \sin\theta_2 & \dots & \sin\theta_n \end{bmatrix}$$

$$= \begin{bmatrix} r_1\sin\theta_1 & r_1\sin\theta_2 & \dots & r_1\sin\theta_n \\ r_2\sin\theta_1 & r_2\sin\theta_2 & & \\ \dots & & \dots & \\ r_m\sin\theta_1 & & \dots & r_m\sin\theta_n \end{bmatrix}$$

A typical MATLAB program would look like

```
r = [0:0.05:1]';                % (21×1)
phi = 0:pi/20:2*pi;             % (1×41)
x = r*cos(phi);                 % (21×41)
y = r*sin(phi);                 % (21×41)
```

Product of a Row Vector and a Matrix

Let *B* be an (*m*×*n*) matrix and *a* a (1×*m*) row vector. Then, the product *g* = *aB* is

$$g = aB = \begin{bmatrix} a_1 & a_2 & \dots & a_m \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & & \\ \dots & & \dots & \\ b_{m1} & & \dots & b_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{j=1}^{m} a_j b_{j1} & \sum_{j=1}^{m} a_j b_{j2} & \dots & \sum_{j=1}^{m} a_j b_{jn} \end{bmatrix} \rightarrow (1 \times n)$$

$$c_{lp} = \sum_{j=1}^{k} a_{lj} b_{jp}$$

***l* = 1**
p* = 1, 2, …, *n

Matrix Exponential

- The matrix operation $\mathbf{A}^2$ means $\mathbf{A} \times \mathbf{A}$, where $\mathbf{A}$ must be a square matrix.
- The operator ^ is used for matrix exponentiation, so, for example, if

```
a =
     1     2
     3     4
```

the statement

```
a ^ 2
```

results in

```
ans =
     7    10
    15    22
```

Try a.^2

in command window to see the difference

Determinants

A determinant of an array *A* of order (*n*×*n*) is represented symbolically as

$$|A| = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & & \dots \\ \dots & & \dots & \\ a_{n1} & \dots & & a_{nn} \end{vmatrix}$$

For *n* = 2:

$$|A| = a_{11}a_{22} - a_{12}a_{21}$$

For *n* = 3:

$$|A| = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33}$$

The MATLAB expression for the determinant is

`det(a)`

Example –

If *A* is defined as

$$A = \begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix}$$

then the determinant of *A* is obtained from

```
A = [1, 3; 4, 2];
d = det(A)
```

which, upon execution, gives

```
d =
   -10
```

Matrix Inverse

The inverse of a square matrix *A* is an operation such that

$$A^{-1}A = AA^{-1} = I$$

provided that *A* is not singular, that is, its determinant is not equal to zero ($|A| \neq 0$).

The expression for obtaining the inverse of matrix *A* is either

```
inv(A)
```

or

```
A^-1
```

Note: $1/A \neq A^{-1}$; $1/A$ will cause the system to respond with an error message. However, $1./A$ is valid, but it is not the inverse.

Example - Inverse of Matrix

We shall determine the inverse of the `magic` matrix and the product of the inverse with the original matrix. The program is

```
invM = inv(magic(3))
IdentMat = invM*magic(3)
```

Its execution gives

```
invM =
    0.1472   -0.1444    0.0639
   -0.0611    0.0222    0.1056
   -0.0194    0.1889   -0.1028
IdentMat =
    1.0000         0   -0.0000
         0    1.0000         0
         0    0.0000    1.0000
```