Lecture 4

In last lecture, we looked at some basic MATLAB syntaxes

vector or array

```
>> x1 = 0

x1 =

     0

>> x2 = 0 ;
```

; → suppress output

```
>> x1 = 1 , x2 = 10 ;
```

; → suppress output

(x1 = 1 , x2 = 10) → Combine two variable declaration into one using comma

```
>> x1 = 1 , x2 = 10

x1 =

     1

x2 =

     10

>> a = 1 : 10
```

1 : 10 → to create vector between 1 & 10 with spacing 1

```
a =

  1   2   3   4   5   6   7   8   9   10

>> a = 1 : 2 : 10
```

1 : 2 : 10 → to create vector between 1 & 10 with spacing 2

```
a =

  1   3   5   7   9
```

```
>> a = 0.5 : (-0.1) : 0.1
a =
    0.5000    0.4000    0.3000    0.2000    0.1000
```

← You can have sequence of decreasing number too

```
>> a = x1 : x2
```

→ a = x1 : dx : x2

→ use variables to create vector (recall that we have set x1=1, x2=10)

```
a =
    1    2    3    4    ..    10

>> x1, x2
```

← check

```
x1 =
    1

x2 =
    10

>> a = [1  2  3]

a =
    1    2    3

>> a = [1, 2, 3]

a =
    1    2    3
```

same effect

```
>> size(a)
```

← compute size of vector or matrix

```
ans =
    1    3
```

← row vector has just 1 row

1 → # row

3 → # columns

```
>> a = [1; 2; 3; 4]
```

→ tell MATLAB to create new row

```
a =
```

1
2
3
4

```
>> size(a)

ans =

  4  1
```

(4) (1) ← column vector has just one column

```
>> b = 1:4

b =
      1   2   3   4

>> a = b'
```

' ← stands for transpose

⇓

row vector becomes column vector

column —"— row vector

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

- Array / matrix

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

A =

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

→ "," for next element in row

";" for next row

```
>> size(A)

ans =
      3   3

>> A = [1:3 ; 4:6 ; 7:9]

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

- Operations on vectors / matrices

+ , − are defined following usual principle
↓
elementwise operations

*, /, ^ — are defined only if these operations are valid between two variables

e.g. if a & b are row vectors

↳ a * b, a / b, a ^ 2 are not valid operations

size(a) = 1 × n
size(b) = 1 × n
→ n = 1

↳ a .* b, a ./ b, a .^ 2 are *, /, ^ operations applied to each element

Similarly

if a & b are column vectors

size(a): n × 1
size(b): n × 1

↳ a * b, a / b, a ^ 2 are not valid operations

↳ a .* b, a ./ b, a .^ 2 elementwise operations defined.

Row vector a, $size(a) = 1 \times n$ ⟹ $a * b$, $size(a*b) = 1 \times 1$

Column vector b, $size(b) = n \times 1$ ⟹ $b * a$, $size(b*a) = n \times n$

General rule

(i) treat row vector as matrix with just one row

(ii) treat column — // — one column

(iii) Let A be a matrix of size $n \times m$

B — // — $l \times k$

Then $A * B$ is defined only if $m = l$

(# columns of A should be equal to # rows of B)

$size(A*B) = n \times k$

$size(B*A) = l \times m$

x, y

$x * y = y * x$

$A * B \neq B * A$

With this rule, you can see multiplication of two rows or two columns is invalid

Example

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad B = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \end{bmatrix}$$

Then

$$A * B = \begin{bmatrix} [1,2,3] \times \begin{bmatrix} 0.1 \\ 0.4 \\ 0.7 \end{bmatrix} & [1,2,3] \times \begin{bmatrix} 0.2 \\ 0.5 \\ 0.8 \end{bmatrix} & [1,2,3] \times \begin{bmatrix} 0.3 \\ 0.6 \\ 0.9 \end{bmatrix} \\ [4,5,6] \times \begin{bmatrix} 0.1 \\ 0.4 \\ 0.7 \end{bmatrix} & [4,5,6] \times \begin{bmatrix} 0.2 \\ 0.5 \\ 0.8 \end{bmatrix} & [4,5,6] \times \begin{bmatrix} 0.3 \\ 0.6 \\ 0.9 \end{bmatrix} \\ [7,8,9] \times \begin{bmatrix} 0.1 \\ 0.4 \\ 0.7 \end{bmatrix} & [7,8,9] \times \begin{bmatrix} 0.2 \\ 0.5 \\ 0.8 \end{bmatrix} & [7,8,9] \times \begin{bmatrix} 0.3 \\ 0.6 \\ 0.9 \end{bmatrix} \end{bmatrix}$$

$size(A*B) = 3 \times 3$

$$= \begin{bmatrix} 3.0 & 3.6 & 4.2 \\ 6.6 & 8.1 & \_ \\ \_ & \_ & \_ \end{bmatrix}_{3 \times 3}$$

↳ From our general rule

- multiplication of row and column vector
- multiplication of column & row vectors

are valid.

e.g. $a = [1, 2, 3]$, $b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

Then (i) $a * b = 14$ ← scalar

(ii) $b * a = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}$ ← matrix

size(a), size(b)

In today's lecture

- plotting
- script file
- function file

Plotting:

```
>> t = 0 : 0.1 : 10 ;      <- "radian"
```

t = [0 , 0.1 , 0.2 , . . . , 10]

row vector →

```
>> yt = sin(t) ;
```

sin(t) = [sin(0), sin(0.1), sin(0.2), . . . , sin(10)]

cos( ) , tan , tanh

```
>> plot(t, yt)
```

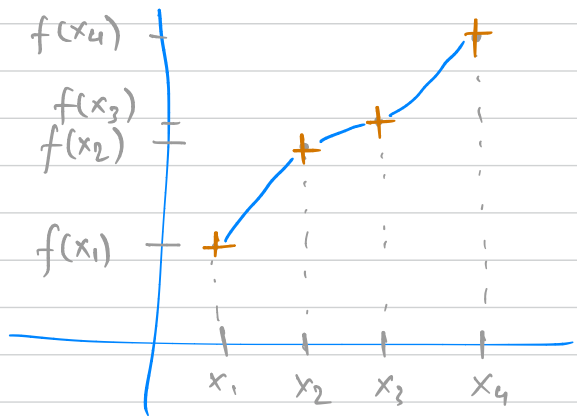
↑ x points

← function values at x points

Documentation

```
>> help plot
```

```
help sin
help size
```



Script file:

```
>> t = 0 : 0.1 : 10 ;      ← row vector
>> yt = sin(t) ;           ← row vector
>> zt = yt.^2 ;            ← row vector

>> plot(t, yt, 'r+:')      ← dotted line
>> hold on                 ← I want to add more plots to the same figure
>> plot(t, zt, 'g*-')      ← solid line
```

→ create a demoPlot.m file

demoPlot.m

```
t = 0:0.1:10;
yt = sin(t);
zt = yt.^2;
plot(t, yt, 'r+:')
hold on
plot(t, zt, 'g*-')
```

(← 20)

```
>> demoPlot ↵
```

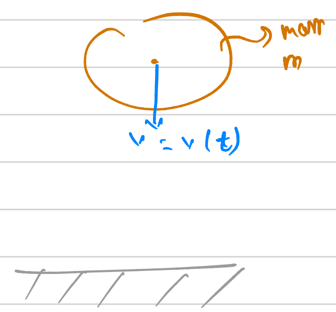
↳ it will put variables in your workspace

⇓

script files

Function files

Recall gravity example



$$\frac{dv}{dt} = g - \frac{C_d}{m} v^2 , \quad v(0) = 0$$

⇒ Exact solution

$$v(t) = \sqrt{\frac{gm}{C_d}} \tanh\left(t \sqrt{\frac{gC_d}{m}}\right)$$

Goal is to create a function in MATLAB that computes v given c_d.

gravityExample.m

```
function v = freefall(t, cd)
% freefall: compute velocity of free falling object assuming mass m=1 kg
% v = freefall(t, cd)
% input:
%     t = time (s)   vector of time
%     cd = drag coefficient (kg/m)
% Output:
%     v = downward velocity (m/s)
g = 9.81;          % gravity acceleration
m = 1;
a = sqrt(m*g/cd);
b = sqrt(g*cd/m);
v = a*tanh(b*t);
```