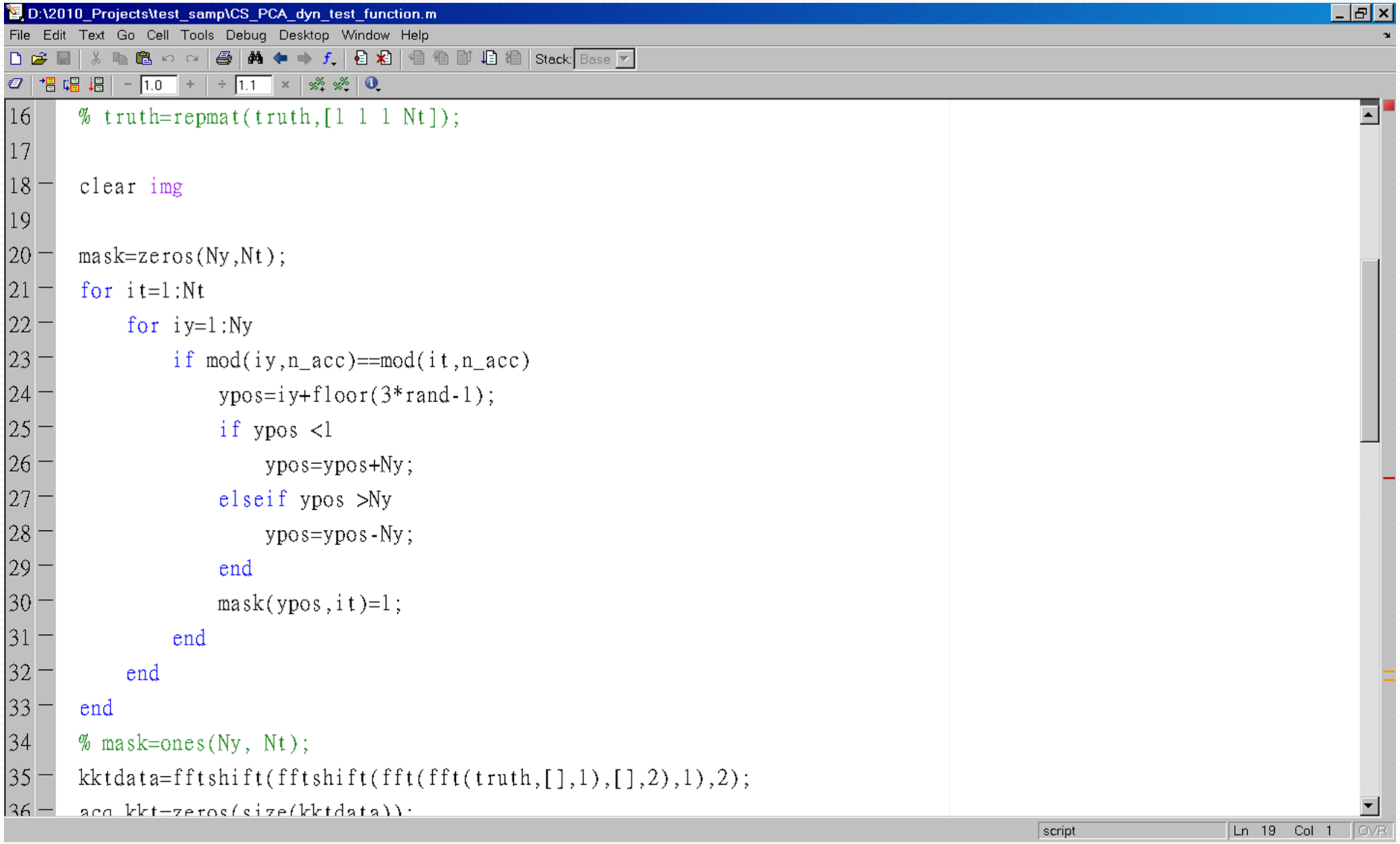
Introduction to MATLAB

資訊工程學系

趙梓程

Shape of a Matlab Script


D:\2010_Projects\test_samp\CS_PCA_dyn_test_function.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1
script
Ln 19 Col 1
OVR

```
% truth=repmat(truth,[1 1 1 Nt]);

clear img

mask=zeros(Ny,Nt);
for it=1:Nt
    for iy=1:Ny
        if mod(iy,n_acc)==mod(it,n_acc)
            ypos=iy+floor(3*rand-1);
            if ypos <1
                ypos=ypos+Ny;
            elseif ypos >Ny
                ypos=ypos-Ny;
            end
            mask(ypos,it)=1;
        end
    end
end
% mask=ones(Ny, Nt);
kktdata=fftshift(fftshift(fft(fft(truth,[],1),[],2),1),2);
acq_kkt=zeros(size(kktdata));
```

Before going further

- CTRL+C : to terminate a process
- Too many histories on the command window
  - clc
- To close bunches of pop-up windows
  - close all

- To clear the memory
  - clear all
- disp : to display the given variable
- % : comment
- fprintf(1,'….',variables)

  = printf(….) in C

Start being acquainted with Matlab

- Variable, Matrix , Array and Structure
- Program basics
- Other useful functions : plot, image, movie
- Input/Output

Variable

Variable declaration & initiation

- No need to declare a variable before using

- Ex: A = 8; B=10.0; C=19.2 + 3.7i; d=f(8);

- Variables are simultaneously declared and initialized.

- Default data type is double unless specified
  - a = int8(30)

Default declared constants

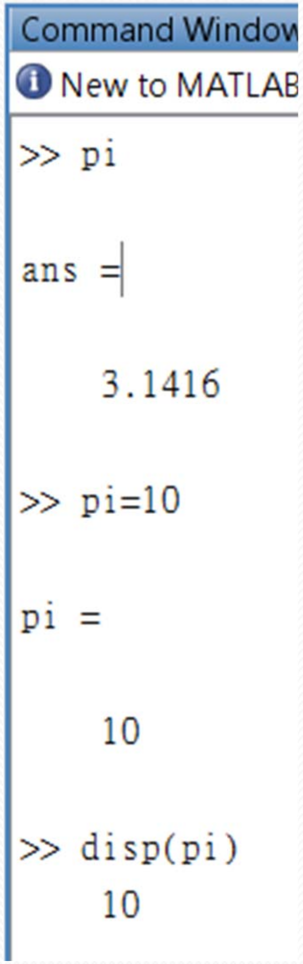
- 1i, 1j, i, j : the imaginary unit
- pi : Ratio of circle's circumference to its diameter
- eps: machine epsilon
- inf : infinity
- nan : not a number

Special declared constants/functions

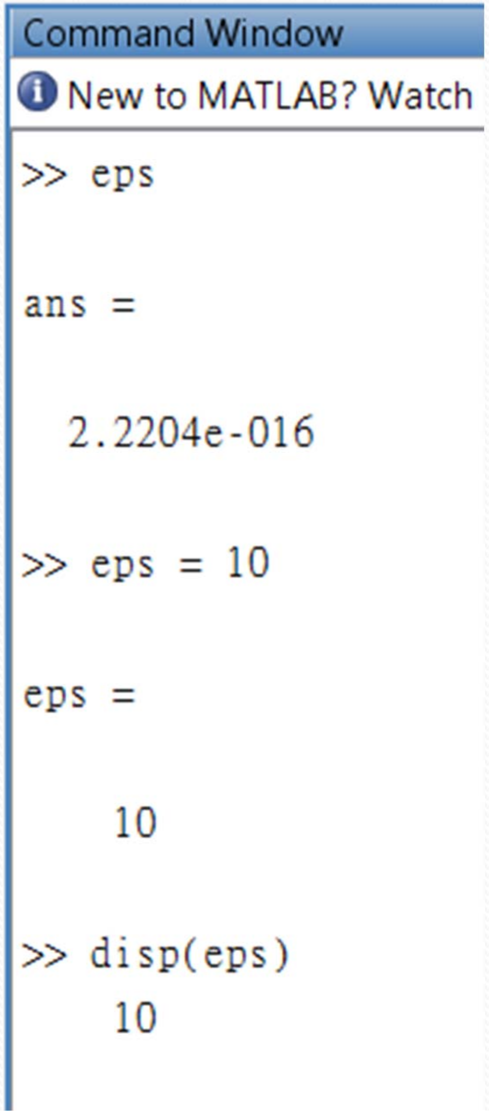
- nargin (can only be used in a function)
  - Number of input variables of a function
- nargout (can only be used in a function)
  - Number of output variables of a function
- realmax
  - Maximum normalized
- realmin
  - Minimm normalized scalar

- *These constants are declared, but they can still be redefined to whatever you want though not recommended.*

Redefined the pre-defined constants



```
>> pi

ans =

    3.1416

>> pi=10

pi =

    10

>> disp(pi)
    10
```



```
>> eps

ans =

  2.2204e-016

>> eps = 10

eps =

    10

>> disp(eps)
    10
```

Scalar Arithmetic

- Basic operations:
  - + , - , * , / , ^

- RECHECK the data type before doing arithmetic.

  - Ex: a = int16(8) ; b=9.2;
  - c = a + b = 17; class(c) = int16

Mathematical functions

- Trigonometric
  - sin, cos, tan,….
  - asin, acos, atan,…
  - sinh, cosh,…
  - sind, cosd, tand,….
  - asind, acosd, atand,…

- Exponential Function
  - ^ : power
  - exp: exponential
  - log, log10,log2

Mathematical functions

- Rounding (round to neighbor integer)
  - fix :toward 0
  - floor :toward -inf
  - ceil : toward +inf
  - round :toward nearest integer
  - mod: modulus
  - rem: remainder

- Complex Function
  - abs
  - real
  - imag
  - conj
  - angle

Matrix and Array

Something about matrices

- The size of an array will be automatically defined when any of the element of the array is specified
- A(1,2,3) : represent the element
- Index starts with 1.
  (similar to Fortran but not C)

Something about matrices

- Generate matrices/arrays
  - Manually List (or using a loop)
  - Using embedded function

- Array management and array arithmetic

Manually List

- A_0 = [1,3,2,4] – (1x4 matrix)
- A_1 = [1;2;3;4] – (4x1 matrix)
- A_2 = 2:0.1:10; (first : increment : last)
- B = [[1,2,3,4];[5,6,7,8];[9 10 11 12]] – (3x4 matrix)
- C = [[1:10];[3:12];[9:18]] – (3x2 matrix)
- Only valid for 1D or 2D matrices

Manually List for Higher Dimension Matrices

- Specify each element. No need to declare before use
  - D(1,1,1)=0; D(1,1,2)=3,…

- Construct a lower dim matrix then expand to higher dim
  - D=[[1 1 3];[2 4 6]];

    D(2,3,3)=10;

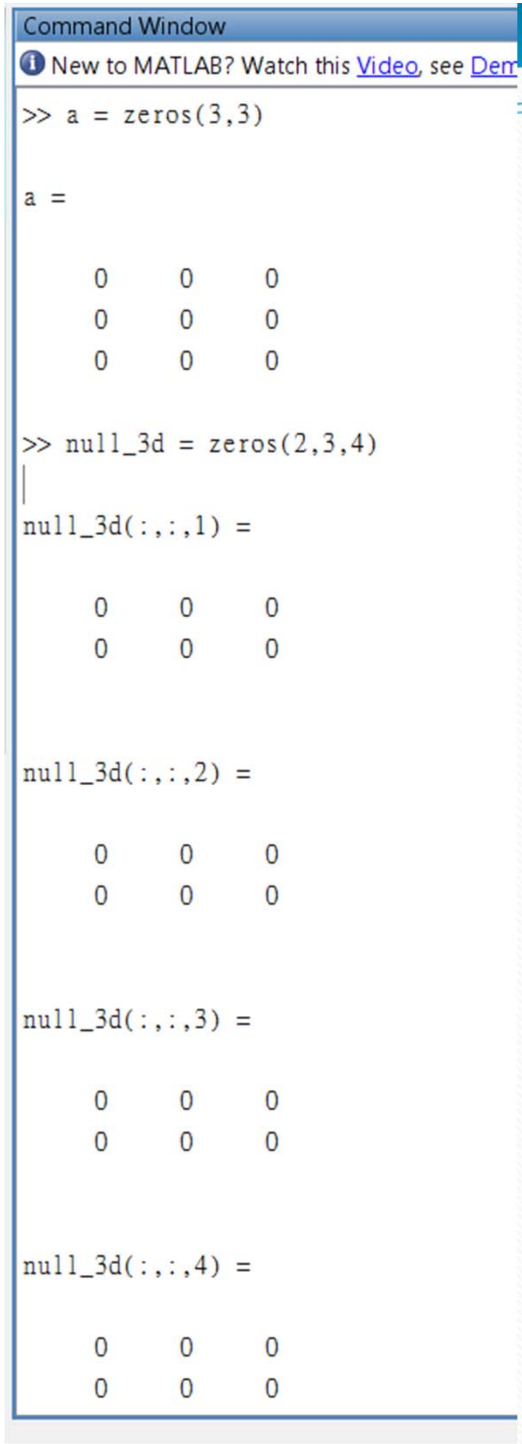
  - E_1=[[1 1 3];[2 4 6]];
    E_2=[[2 3 5];[1 2 8]];
    E(:,:,1)=E_1;
    E(:,:,2)=E_2;

Special Functions that create arrays

- zeros, ones, rand
  - null_2d = zeros(3,3)
  - null_3d = zeros(2,3,4)
  - one_3d = ones(2,2,3)
  - rand_3d = rand(2,4,3)
- eye : unit matrix

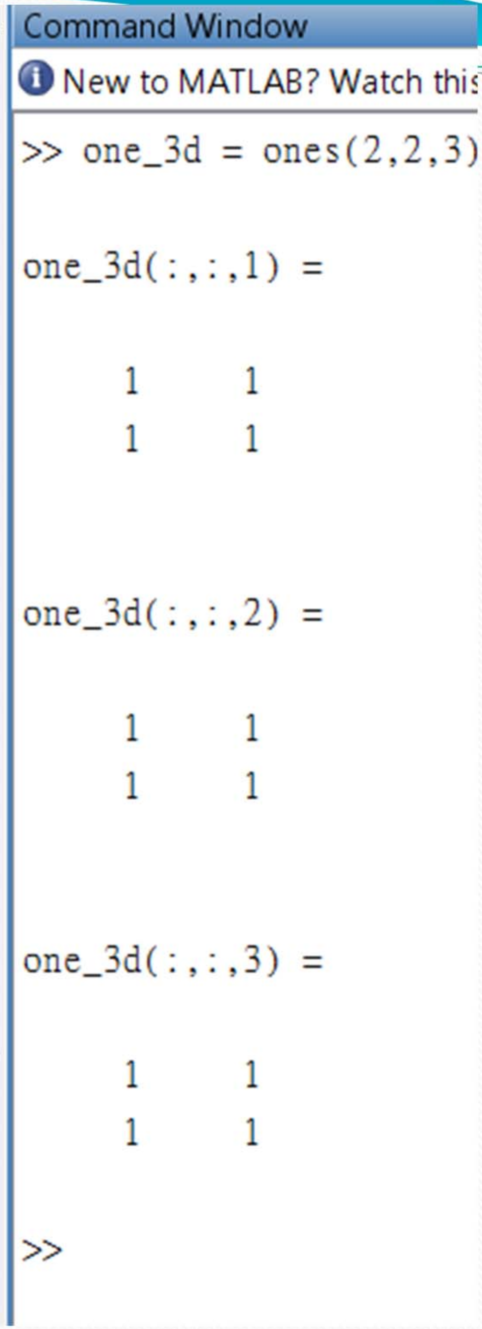
- linspace, logspace
- y = linspace(a,b,n)
- y = logspace(a,b,n)

- Meshgrid
- [X,Y] = meshgrid(x,y)
- [X,Y,Z] = meshgrid(x,y,z)



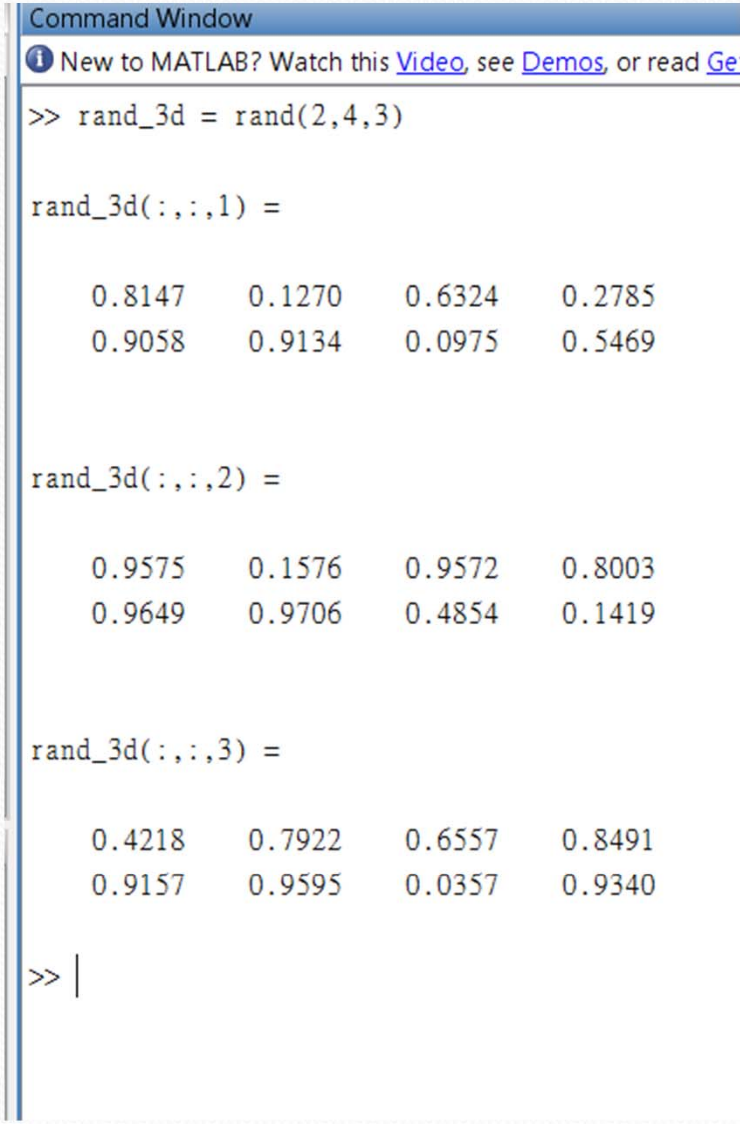
New to MATLAB? Watch this Video, see Dem

```
>> a = zeros(3,3)

a =

     0     0     0
     0     0     0
     0     0     0

>> null_3d = zeros(2,3,4)

null_3d(:,:,1) =

     0     0     0
     0     0     0


null_3d(:,:,2) =

     0     0     0
     0     0     0


null_3d(:,:,3) =

     0     0     0
     0     0     0


null_3d(:,:,4) =

     0     0     0
     0     0     0
```



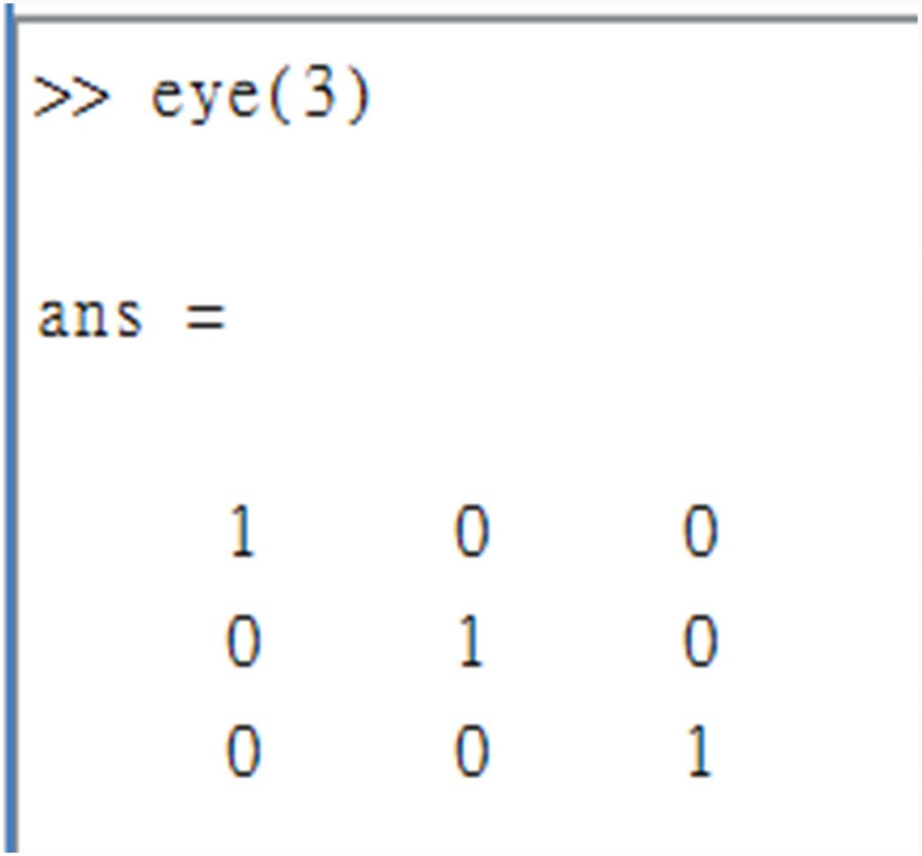
New to MATLAB? Watch this

```
>> one_3d = ones(2,2,3)

one_3d(:,:,1) =

     1     1
     1     1


one_3d(:,:,2) =

     1     1
     1     1


one_3d(:,:,3) =

     1     1
     1     1

>>
```

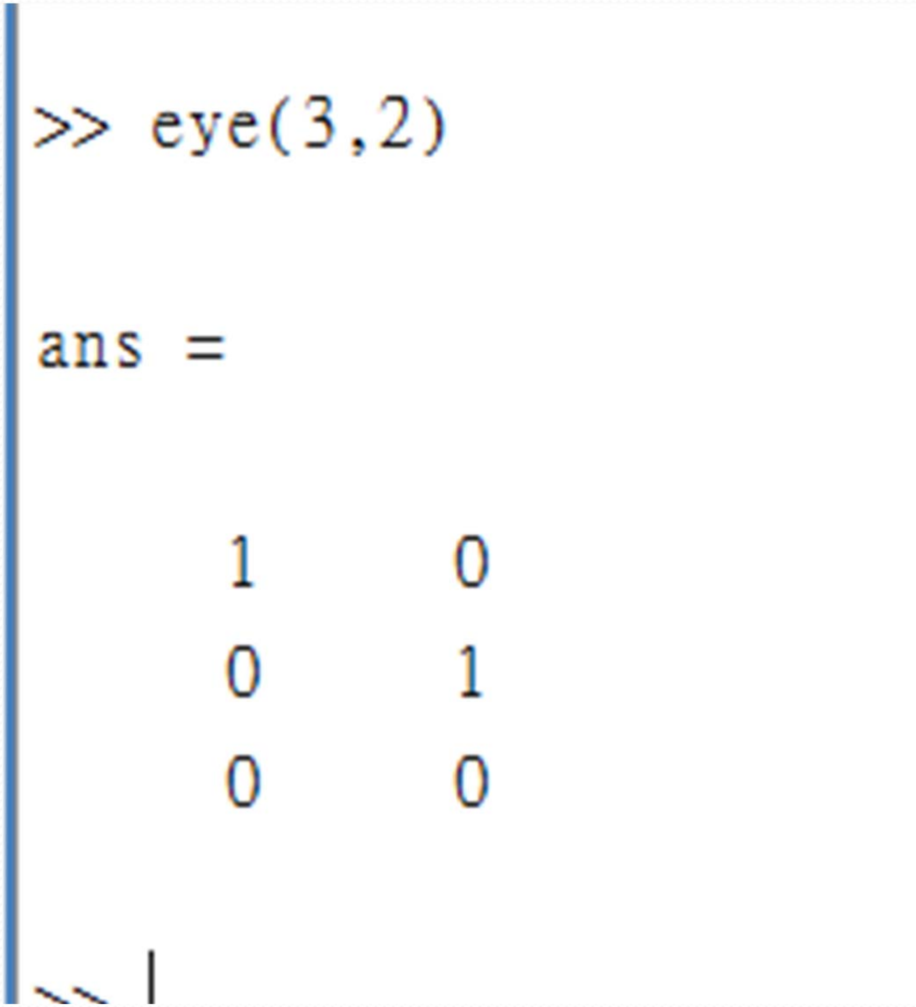


New to MATLAB? Watch this Video, see Demos, or read Ge

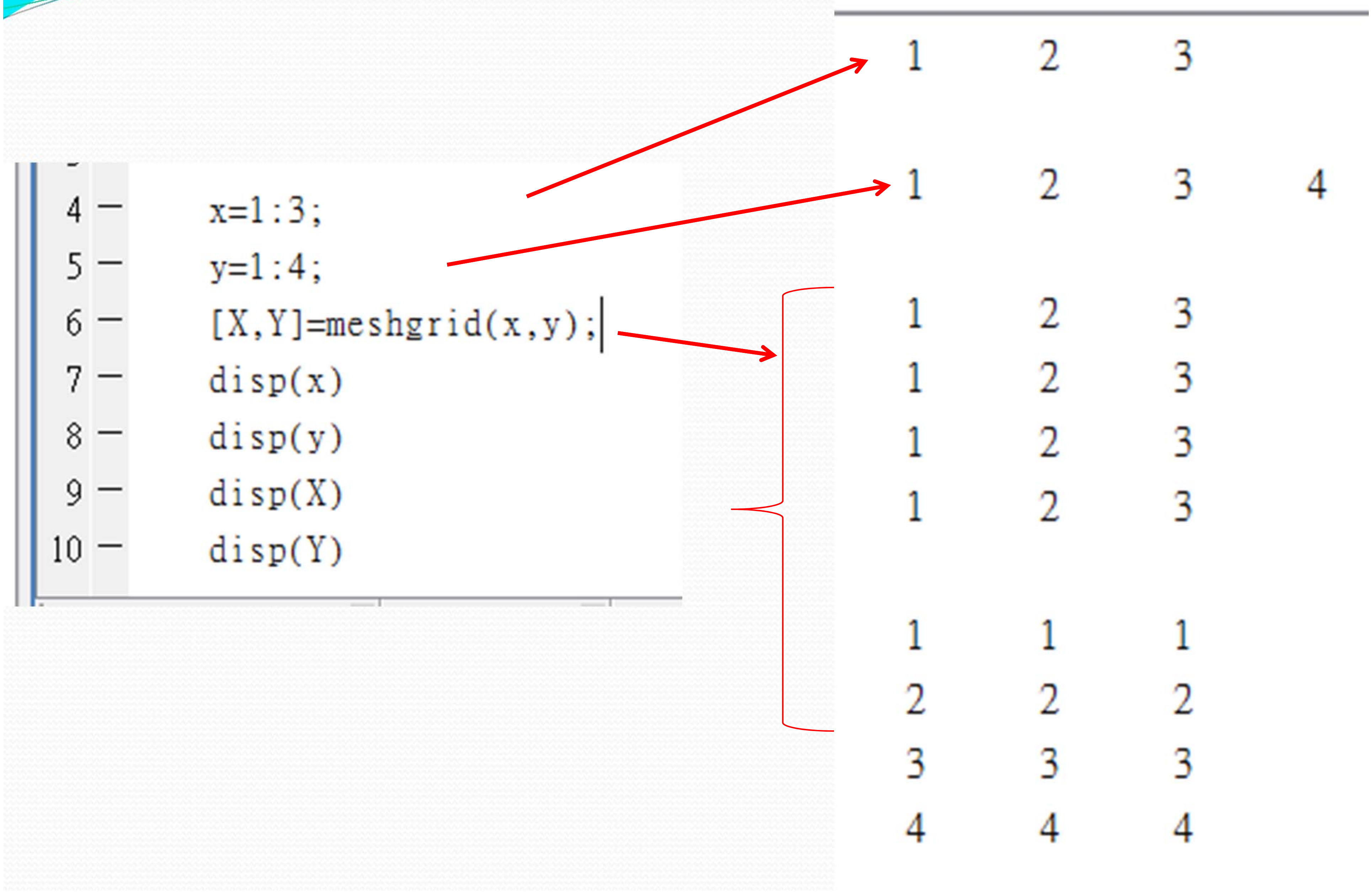
```
>> rand_3d = rand(2,4,3)

rand_3d(:,:,1) =

    0.8147    0.1270    0.6324    0.2785
    0.9058    0.9134    0.0975    0.5469


rand_3d(:,:,2) =

    0.9575    0.1576    0.9572    0.8003
    0.9649    0.9706    0.4854    0.1419


rand_3d(:,:,3) =

    0.4218    0.7922    0.6557    0.8491
    0.9157    0.9595    0.0357    0.9340

>>
```



```
>> eye(3)


ans =


     1     0     0
     0     1     0
     0     0     1
```

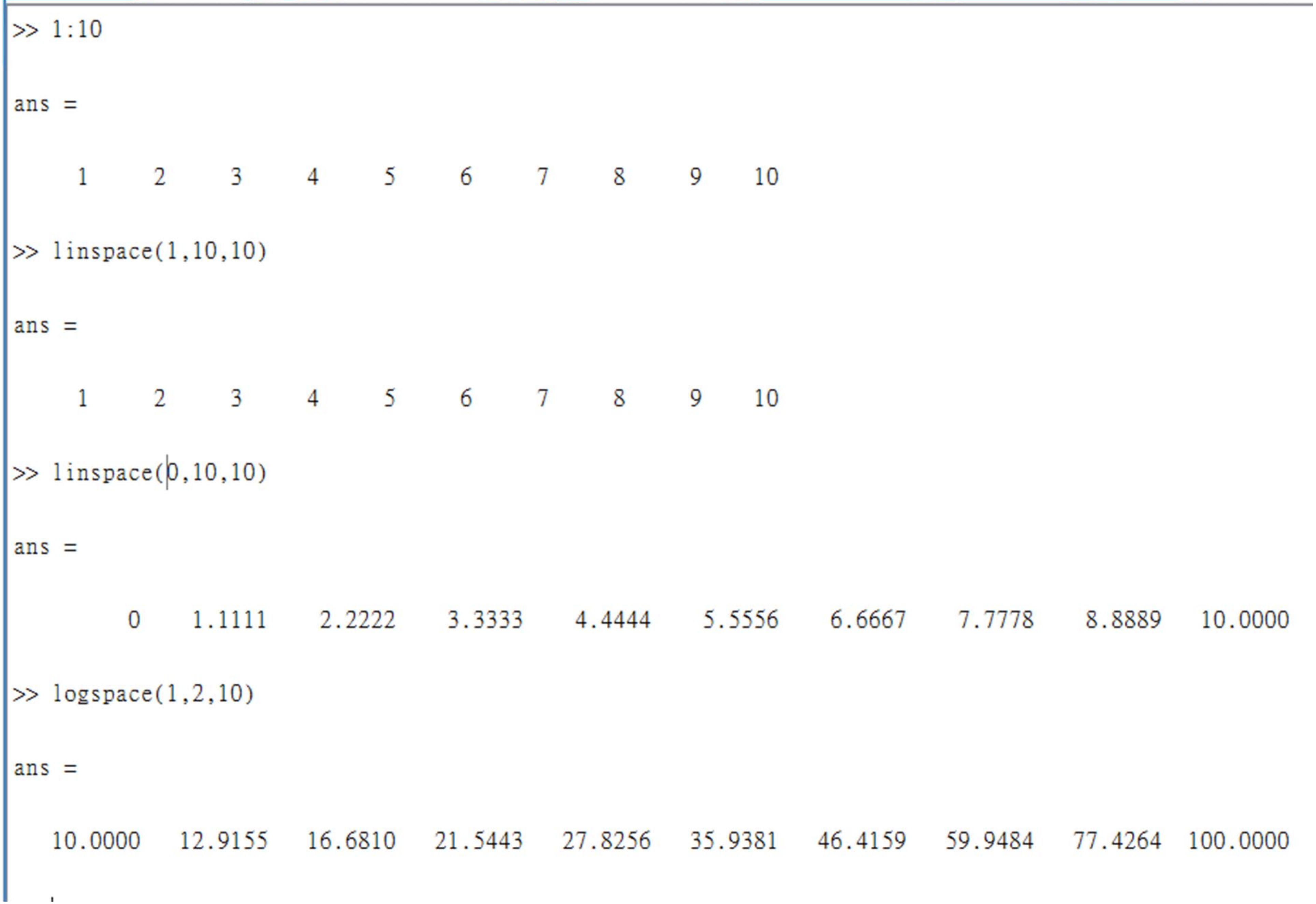


```
>> eye(3,2)


ans =


     1     0
     0     1
     0     0


>>
```



```
x=1:3;
y=1:4;
[X,Y]=meshgrid(x,y);
disp(x)
disp(y)
disp(X)
disp(Y)
```

```
     1     2     3

     1     2     3     4

     1     2     3
     1     2     3
     1     2     3
     1     2     3

     1     1     1
     2     2     2
     3     3     3
     4     4     4
```

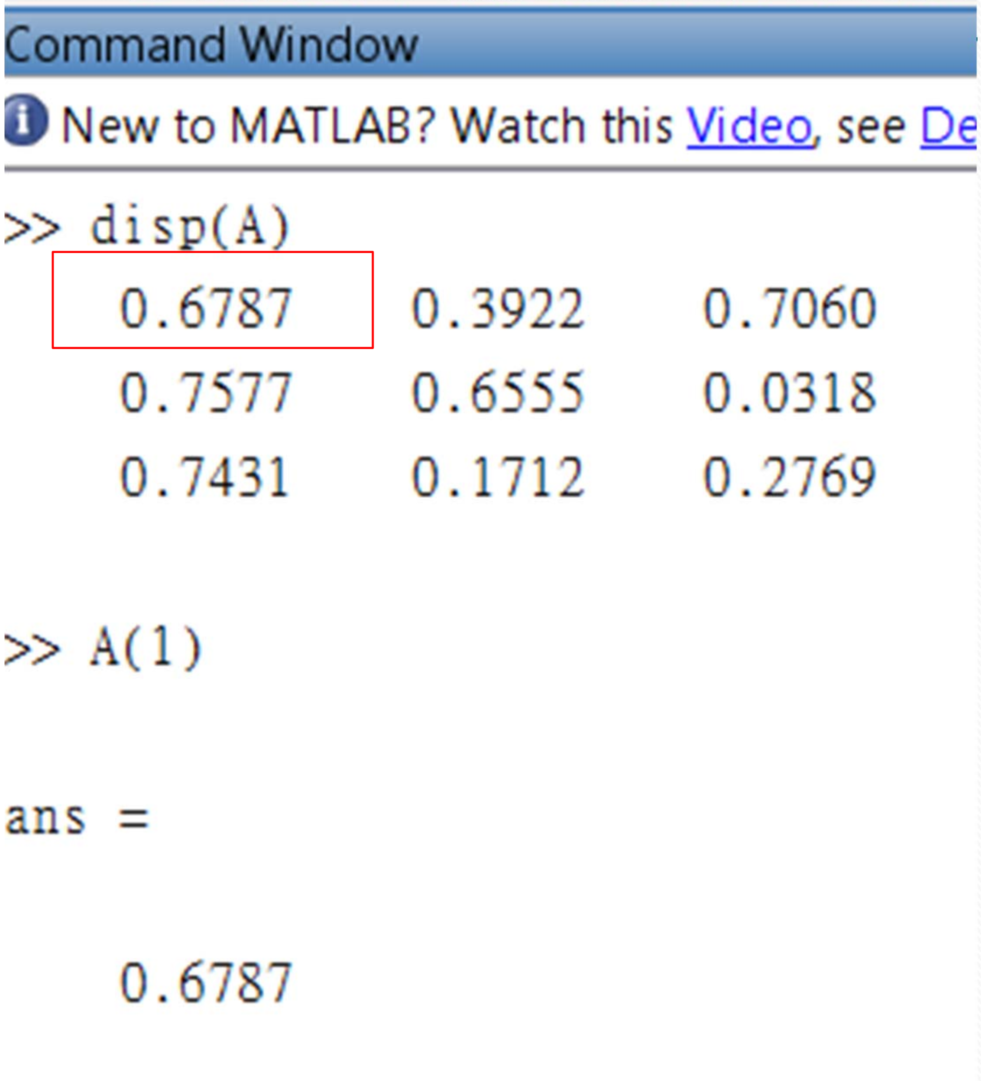


```
>> 1:10

ans =

     1     2     3     4     5     6     7     8     9    10

>> linspace(1,10,10)

ans =

     1     2     3     4     5     6     7     8     9    10

>> linspace(0,10,10)

ans =

         0    1.1111    2.2222    3.3333    4.4444    5.5556    6.6667    7.7778    8.8889   10.0000

>> logspace(1,2,10)

ans =

   10.0000   12.9155   16.6810   21.5443   27.8256   35.9381   46.4159   59.9484   77.4264  100.0000
```

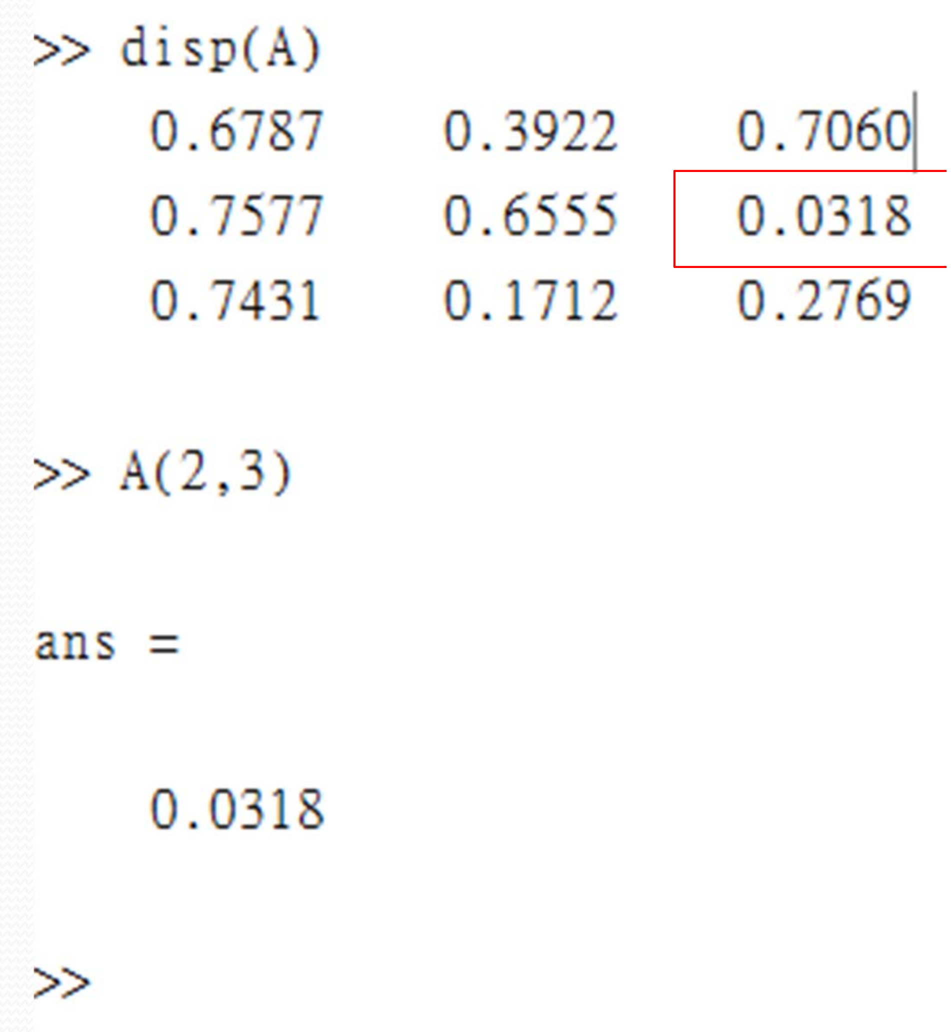
Array manipulation

- Extract the array elements
- A(i) : extract the i-th element of the array
- A(3,1,2) : extract the assigned element
- A(:,:,2)   : extract the 2D array at the 2nd layer
- A(2:end,2:3,2)

Array manipulation

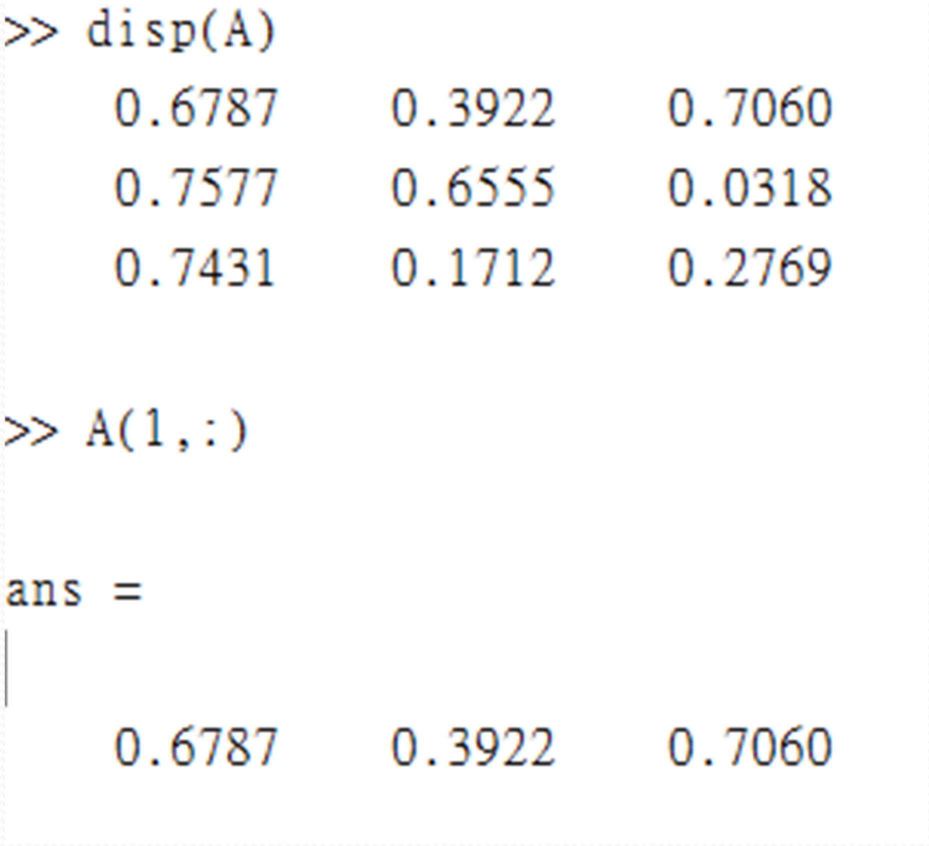


```
>> disp(A)
    0.6787    0.3922    0.7060
    0.7577    0.6555    0.0318
    0.7431    0.1712    0.2769

>> A(1)

ans =

    0.6787
```

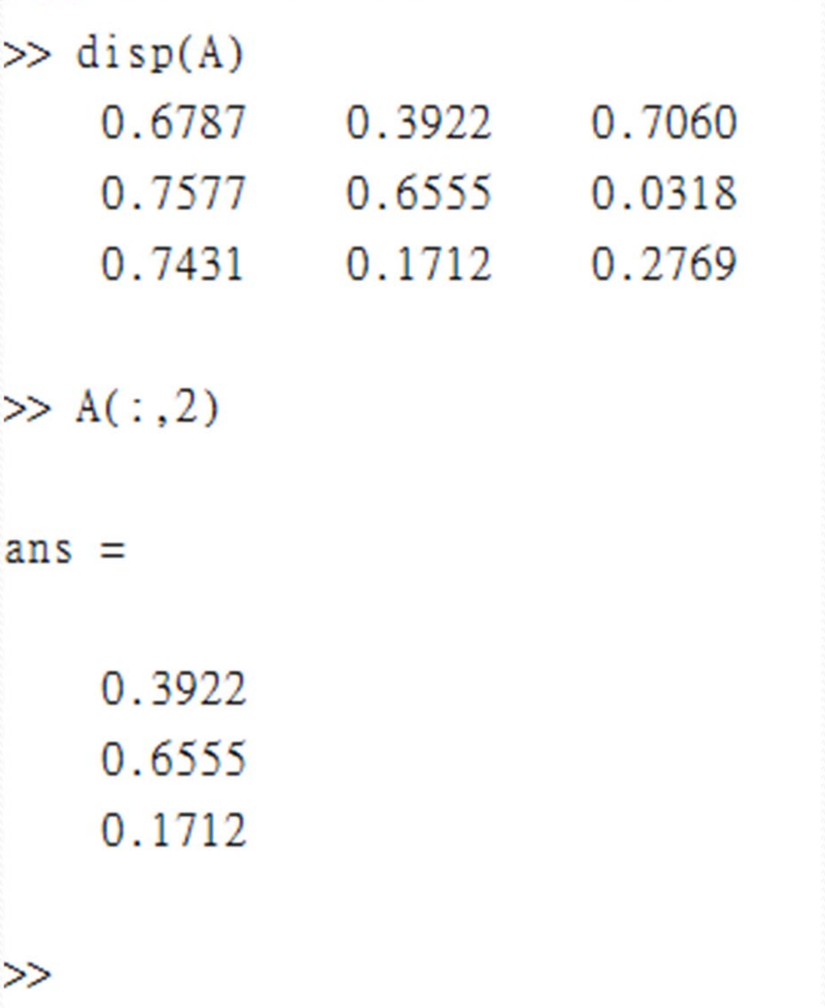


```
>> disp(A)
    0.6787    0.3922    0.7060
    0.7577    0.6555    0.0318
    0.7431    0.1712    0.2769

>> A(2,3)

ans =

    0.0318

>>
```

Array manipulation



```
>> disp(A)
    0.6787    0.3922    0.7060
    0.7577    0.6555    0.0318
    0.7431    0.1712    0.2769


>> A(1,:)


ans =


    0.6787    0.3922    0.7060
```



```
>> disp(A)
    0.6787    0.3922    0.7060
    0.7577    0.6555    0.0318
    0.7431    0.1712    0.2769


>> A(:,2)


ans =


    0.3922
    0.6555
    0.1712


>>
```

Array manipulation

- Rearrange the arrays
  - A(:) : cascade the multi-dimension array to be 1-D vector
  - Permutation: permute(A,[2 1])
  - Reshape : reshape(A,[a,b,c])
    - Note: the total number of elements before and after reshape must be identical

Addition/subtraction/ multiplication/division/power

- Elementwise arithmetic (Note the ‘dot’)
- Dimension of the arrays must be identical
- C_1 = A+B;
- C_2 = A-B;
- C_3 = A.*B;
- C_4 = A./B;
- C_5 = A.^2;

Basic program knowledge

If you know how to use C/C++, or Fortran, you may use similar way to construct a MATLAB script

Basic program knowledge

- creating an m-script
- Function/subroutine
- Function handle
- Debug mode

- Flow Countrol
  - While loop
  - For loop
  - If…else
  - Switch….

creating an m-script

- “FILE”->”NEW”->”Script”
- Copy what you wrote in the command history (of course only those correctly executed)
- Save the m-file to the designated folder
- Click F5 or type the filename on the command window

Function / Subroutines

- Like Fortran, MATLAB passes arguments by address. No pointer like declaration is used during functional call in MATLAB.

- Basic subroutine usage:

```
function a = my_fun(x,y,z)
        a=x+y+z….;
 return
```

Subroutine usage

- Input arguments shall be at right hand side
  - a = my_fun(x,y,z)
- Output arguments shall be at left hand side
  - a = my_fun(x,y,z)
- Remember to include a “return” or an “end” at the end
- The input and output arguments can be either kind including variable, array/matrix, and structure (cell array)

Multiple inputs, outputs and functions

- function [a,b,c] = my_fun2(x,y,z);

```
a =  x;
 b = y*z;
 c =  [x^2, y^2, z^2];


  return
```

Note:
3 inputs
3 outputs

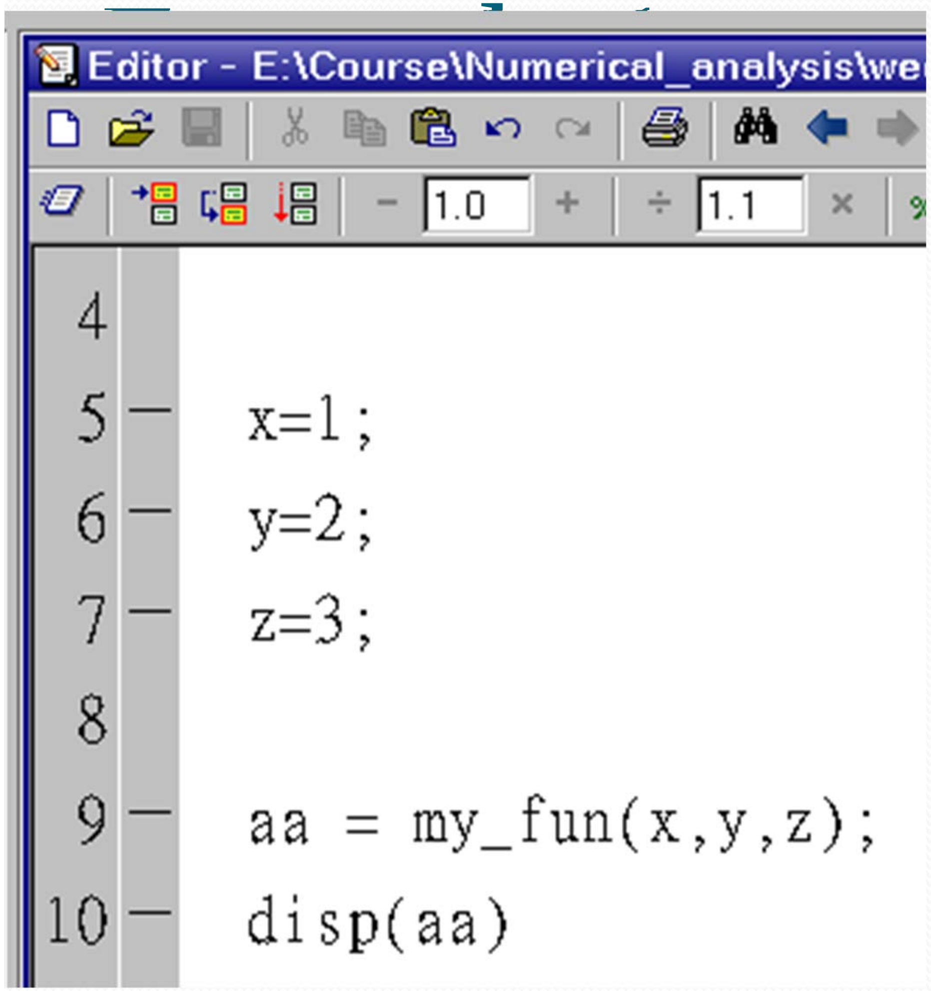
- function [a,b,c,d] = my_fun3(x,y,z)

Note:
3 inputs
4 outputs

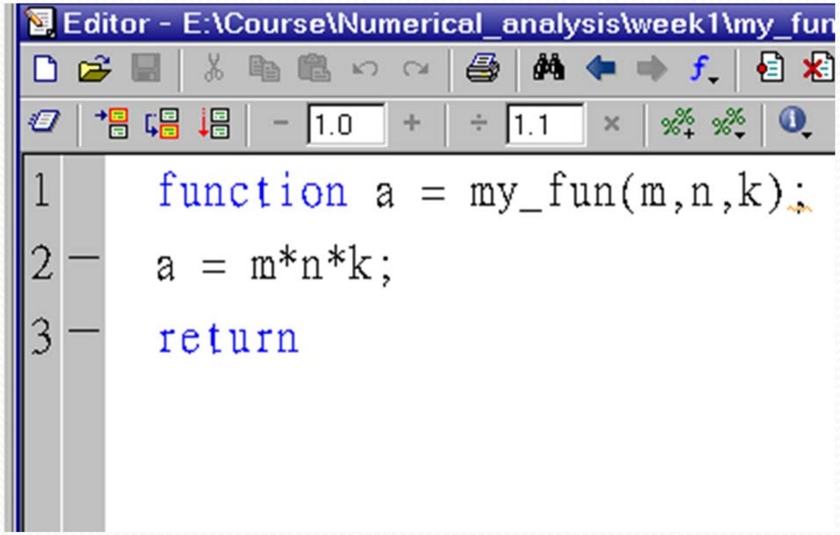
```
m = level_2_func(y)
a = m*x;
b = [a,a,a];
c =….
d = …

function mm = level_2_func (yy)
  m = y*y;
  return
return
```

Note:
1 input
1 output

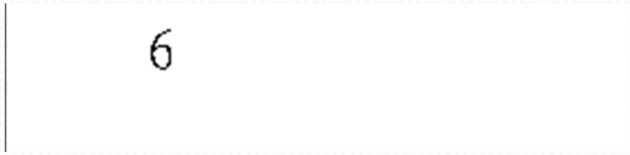
Example: 3 inputs , 1 output



```

x=1;
y=2;
z=3;

aa = my_fun(x,y,z);
disp(aa)
```



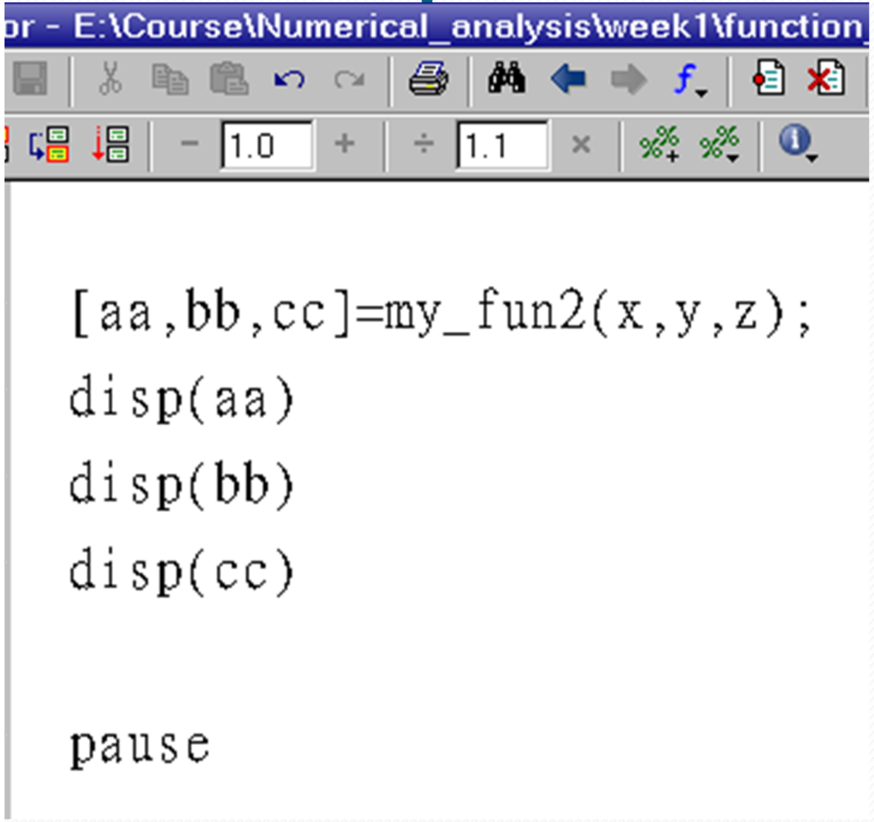
```
function a = my_fun(m,n,k);
a = m*n*k;
return
```



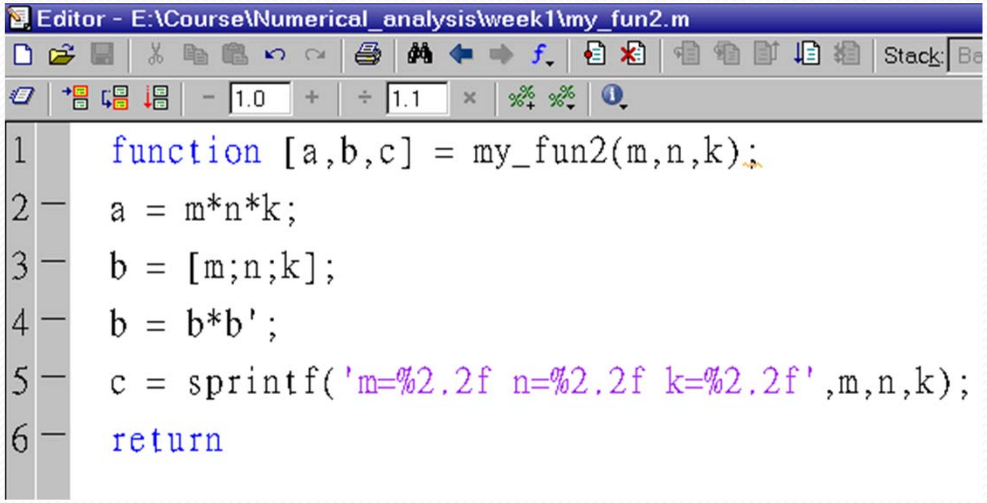
```
6
```

Example:
Input: three numbers ;Output a:a number, b:a matrix, c:a string

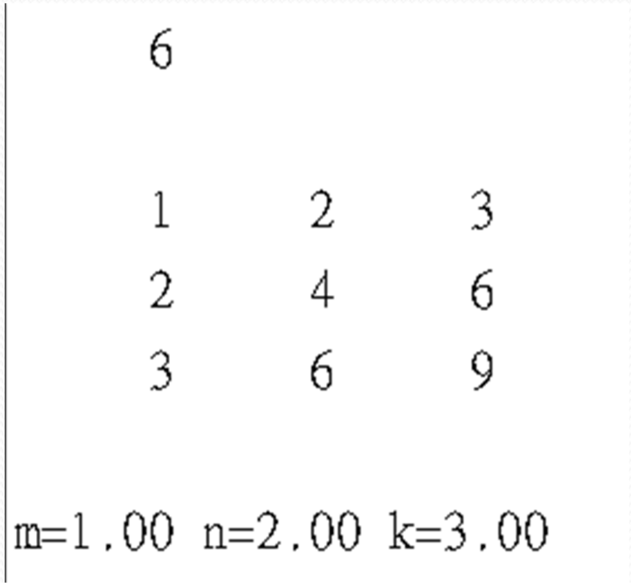
Example 2



```
[aa,bb,cc]=my_fun2(x,y,z);
disp(aa)
disp(bb)
disp(cc)


pause
```



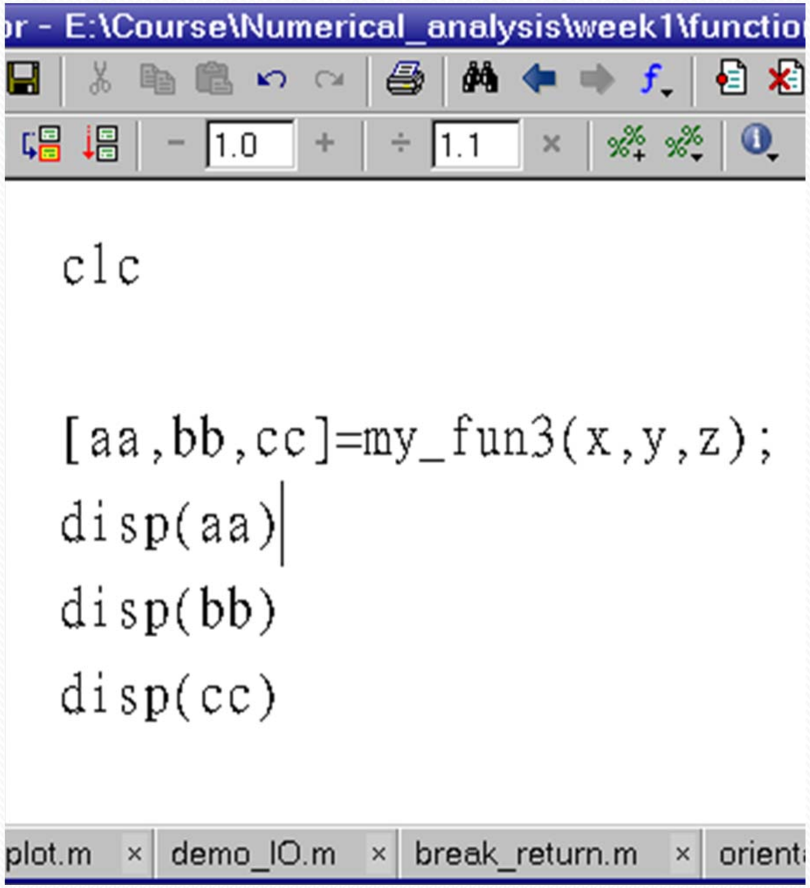
```
function [a,b,c] = my_fun2(m,n,k);
a = m*n*k;
b = [m;n;k];
b = b*b';
c = sprintf('m=%2.2f n=%2.2f k=%2.2f',m,n,k);
return
```



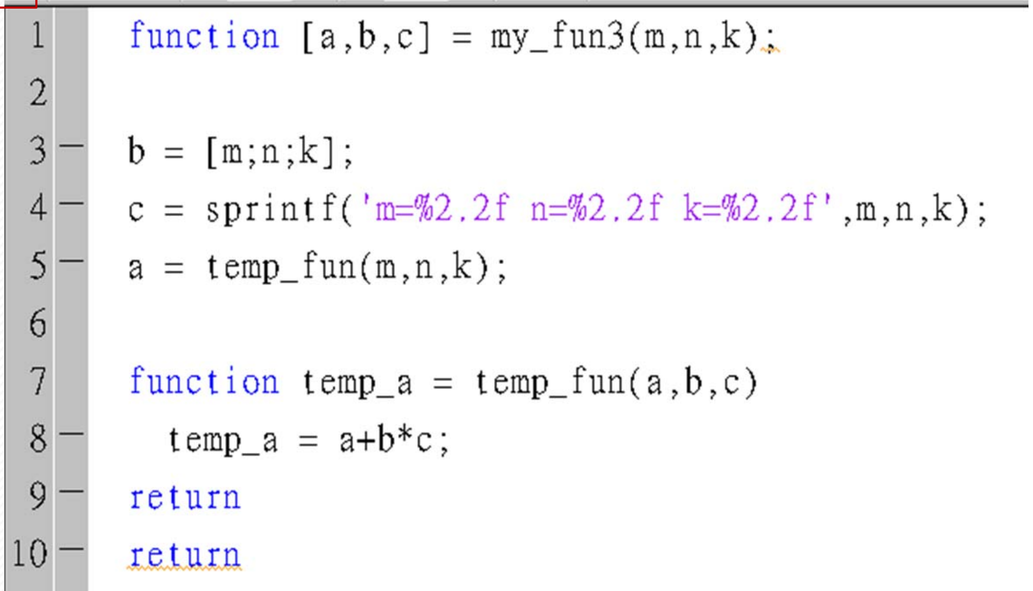
```
     6


     1     2     3
     2     4     6
     3     6     9


m=1.00 n=2.00 k=3.00
```

Example:
a two level nested function

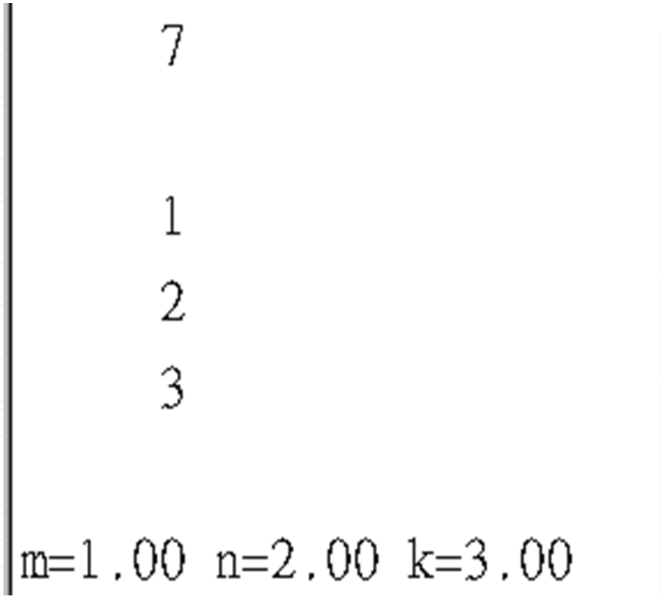
Example 3



```
clc


[aa,bb,cc]=my_fun3(x,y,z);
disp(aa)
disp(bb)
disp(cc)
```



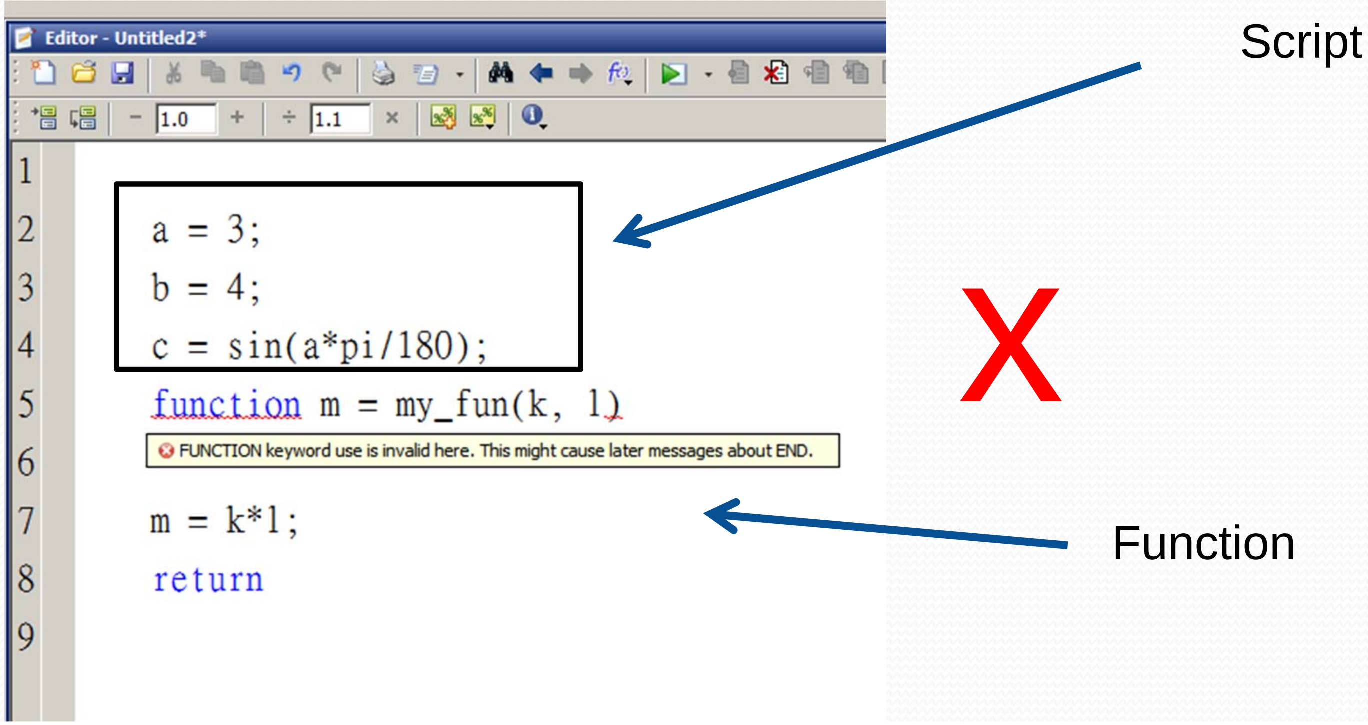
```
function [a,b,c] = my_fun3(m,n,k);

b = [m;n;k];
c = sprintf('m=%2.2f n=%2.2f k=%2.2f',m,n,k);
a = temp_fun(m,n,k);

function temp_a = temp_fun(a,b,c)
  temp_a = a+b*c;
return
return
```



```
     7


     1
     2
     3


m=1.00 n=2.00 k=3.00
```

About subroutine

- Nested functions are allowed
- Script cannot be mixed with function definition



```

a = 3;
b = 4;
c = sin(a*pi/180);
function m = my_fun(k, l)

m = k*l;
return
```

FUNCTION keyword use is invalid here. This might cause later messages about END.

Script

X

Function

About subroutine

- The filename of a function is suggested to be the same as the name of the function.

- In script, the program only recognizes the name of the m-file contains the function
  - always use the filename to call the function

Debug functions

- keyboard
  - enter debug mode
  - the program will pause at the command
  - you may check all the variables of the subroutine where "keyboard" is placed
- return
  - carry on the rest of the program
- dbquit
  - to quit the debug mode

While loop

- While expression
  - statements
- end

FOR Loop

- Start with
  - for ii = 1:5

- End with
  - end

Example of for loop

```
a = 10;


for ii = 1:10
    disp(a + ii);
end
```

Example of for loop

```
a = 10;

for ii = 1:2:10
    disp(a + ii);
end
```

Example of for loop

```
a = 10;

for ii = [3,5,8, 11, 200]
    disp(a + ii);
end
```

To terminate the loop

- Break:
  - only go out of the loop where the program is currently running

- Return
  - back to the main function

Example

```
for ii = 1:3
    for kk = 1:3
        txt = sprintf('ii=%d  kk=%d\n',ii, kk)
        disp(txt);
        if(kk ==2)
            break/ return
        end
    end
end
```

break V.S. return

- Break
- ii=1 kk=1
- ii=1 kk=2
- ii=2 kk=1
- ii=2 kk=2
- ii=3 kk=1
- ii=3 kk=2

- Return
- ii=1 kk=1
- ii=1 kk=2

If…elseif…else

- if expression1
- statements1
- elseif expression2
- statements2
- else
- statements3
- end

Do not use “else if”
if you don’t know what it
means

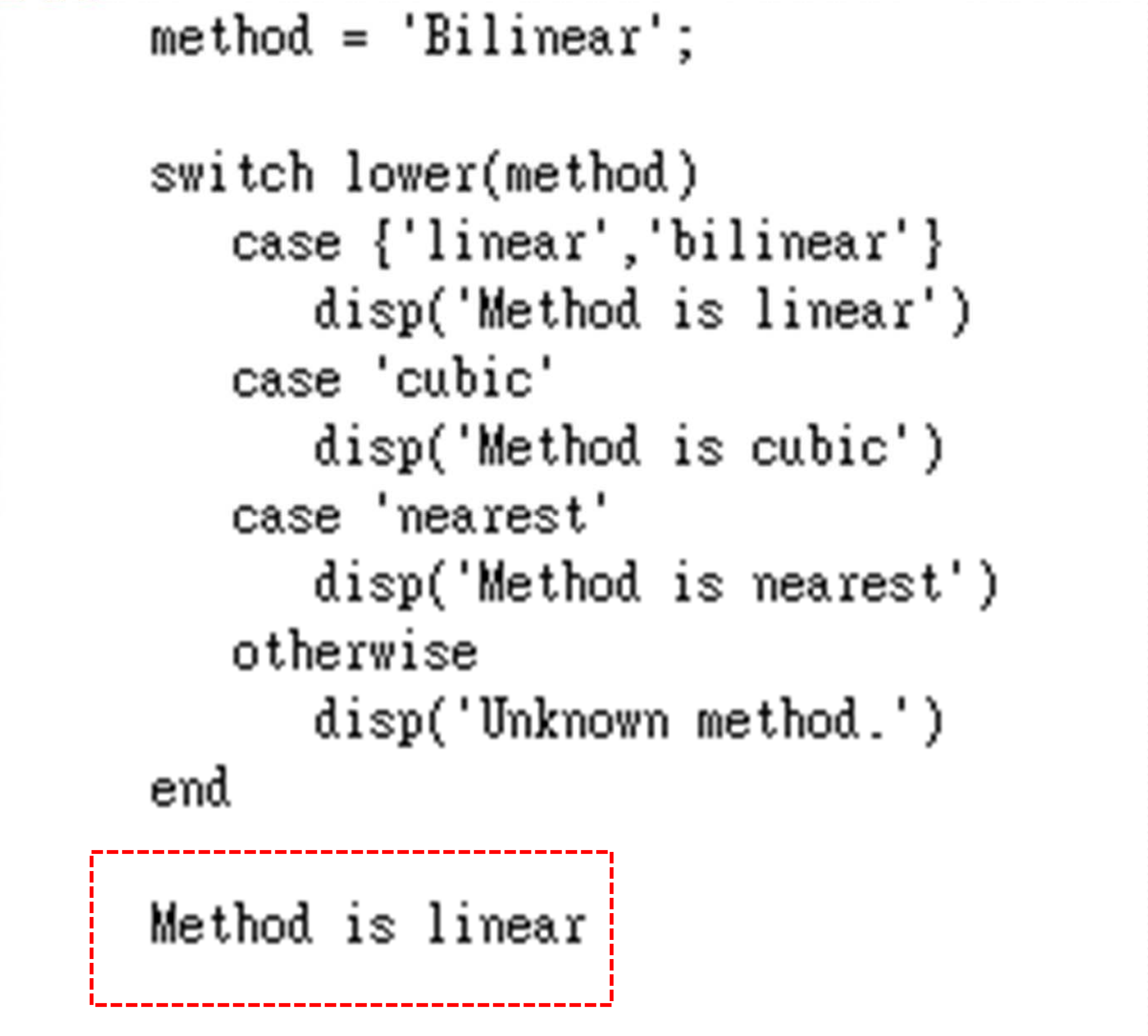
Logic Express

- Not : ~
- And : & (&& is reserved for short-circuit and)
- OR : | (|| is reserved for short-circuit or)

switch

```
switch switch_expr
 case case_expr
    statement, ..., statement
  case {case_expr1, case_expr2, case_expr3, ...}
    statement, ..., statement
  otherwise
    statement, ..., statement
end
```

switch



```
method = 'Bilinear';

switch lower(method)
    case {'linear','bilinear'}
        disp('Method is linear')
    case 'cubic'
        disp('Method is cubic')
    case 'nearest'
        disp('Method is nearest')
    otherwise
        disp('Unknown method.')
end
```

```
Method is linear
```

Plot function

Figure object management

- figure :
  - Create a new figure object for plot or image
  - ex: figure(1), figure(100)

- subplot:
  - separate figure into several subplots
  - subplot(2,2,1), subplot(2,2,4)

- hold on;
  - Keep the former plots while the new one is generated
  - ‘hold off’ to disable the ‘hold on’ command

plot

- **plot**函數：
- **plot(x,y)**：

The size of x must match the size of y

if x and y are both in vector form

“plot” draws dots based on each x, y pair with connecting lines. Find out more detail by yourself in the help manual.

Other plots

- **plot**：linear scale on x and y axes
- **loglog**：dual log scale on x and y axes
- **semilogx**：x-log scale, y- linear scale
- **semilogy**：x-linear scale, y-log scale
- **plotyy**：two different linear y range in the same plot
  - Remember to give 4 vectors

Plot options

- **title(‘title’)**：Set the title for the plot
- **xlabel(‘x_axis_name’)**：label of x axis
- **ylabel(‘y_axis_name’)**：label of y axis
- **grid on/off**：show/hide the grids
- **text(x, y, ‘text-at-x,y’)**：the text shown in the plot
- **gtext(‘text’)**：place text at where the cursor points
- **axis([xmin, xmax, ymin, ymax])**：the bound of the axes

3D plots

- meshgrid：對z=f(x,y)函數，產生完整的xy平面圖形點集（以繪出立體圖） mesh、surf、surfl、surfc、contour、contour3、contourf：先計算z值後再用這些函數繪出

3D plots

- **clf %** 清除圖形視窗
- **[x,y]=meshgrid(-4.0:0.2:4.0, -4.0:0.2:4.0); %** 產生**xy**平面上的格子點
- **z=0.5*(-20*x.^2+x)+0.5*(-15*y.^2+5*y);**
- **figure(1);**
- **surfl(x,y,z); %** 畫出三維圖形曲面上的點
- **axis([-4 4 -4 4 -400 0]) %** 設定圖軸範圍，**-4<=x<=4, -4<=y<=4, -400<=z<=0**
- **xlabel('x-axis'); ylabel('y-axis'); zlabel('z-axis');**
- **figure(2);**
- **contour3(x,y,z,15); %** 畫出曲面的立體輪廓線圖 **(15**層**)**
- **axis([-4 4 -4 4 -400 0])**
- **xlabel('x-axis'); ylabel('y-axis'); zlabel('z-axis');**
- **figure(3);**
- **contourf(x,y,z,10) %** 畫出二維填滿的輪廓線圖 **(10**層**)**
- **xlabel('x-axis'); ylabel('y-axis');**

Animation

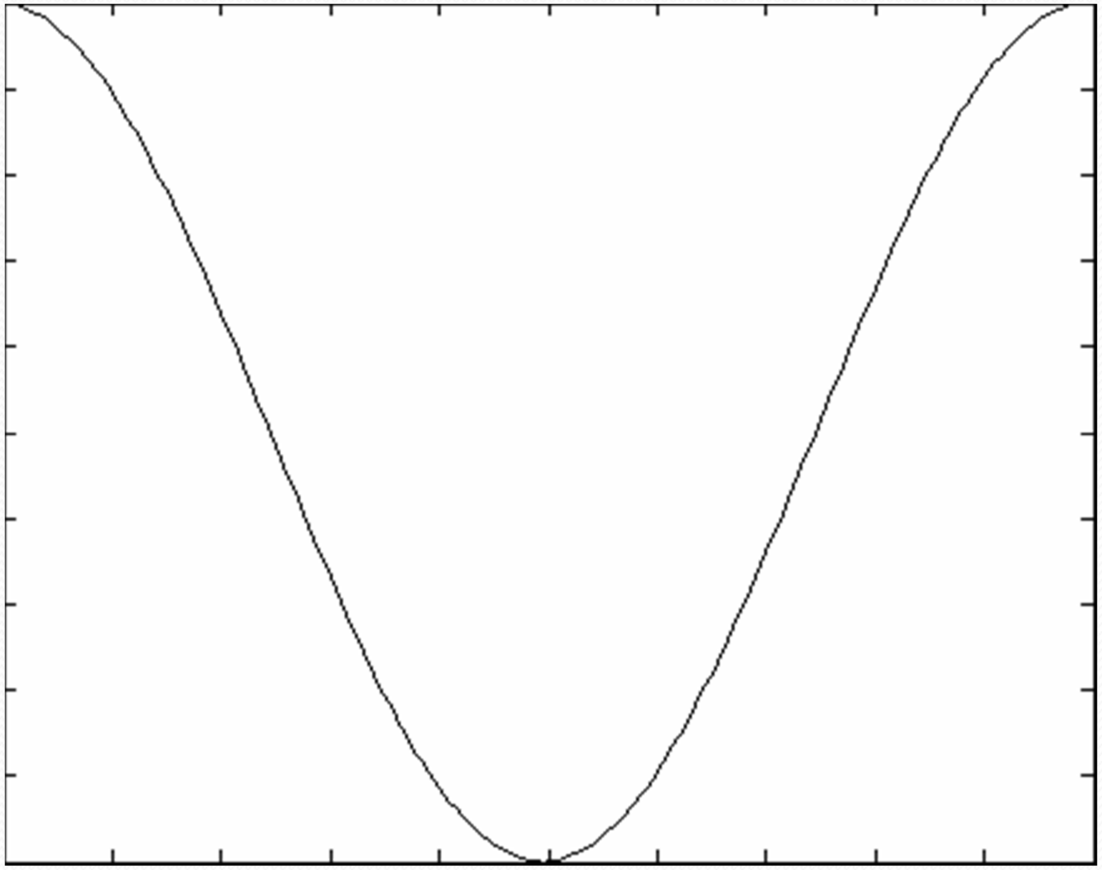
- To make a fancier and understandable demostration

- Commands
  - %% to covert a figure object to ONE frame
  - M = getframe;
  - %% to display the frames in the array M
  - Movie(M)

Example

```
x=1:200;
Nx = 200;
for t =1:100
    y = cos(x./Nx*2*pi + t/Nx*2*pi);
    plot(x,y,'b');
    M(t)=getframe;
end
movie(M)
```

Example



Input/output

Check Help

fopen, fread, fwrite,
fprintf, fscanf,
save, load

Brief Summary

- Matlab is a useful tool during program development stage
- The code can be easily translated to C or Fortran
- Make good use of the help manual