Lecture 2

Programming with MATLAB

Ye Ding (丁烨)

Email: y.ding@sjtu.edu.cn

School of Mechanical Engineering

Shanghai Jiao Tong University

Programming with MATLAB

- **Reference for Programming**

1. **Timothy Sauer, Numerical analysis (2nd ed.), Pearson Education, 2012. Appendix B**
2. **MATLAB, Help Documentation**

Programming with MATLAB

M-files

MATLAB programs are usually written into files called M-files

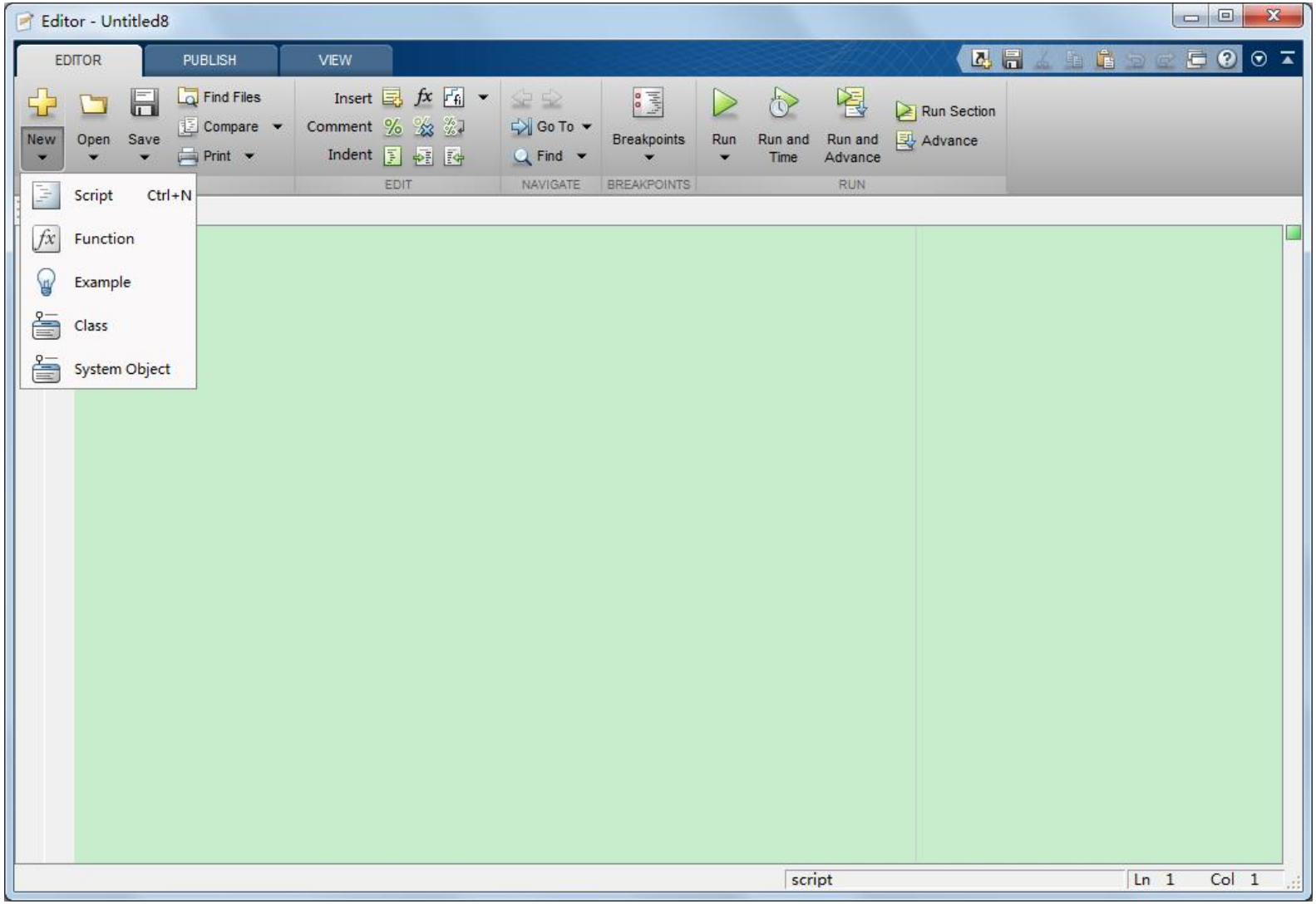
- **M-file Script**
- **M-file Function**

Elements of a program:

- Data, Input & Output
- Files (main script, functions)
- Variables & Operations
- Flow Control
- Explanatory Text and Comments
- Algorithm (accuracy & efficiency)

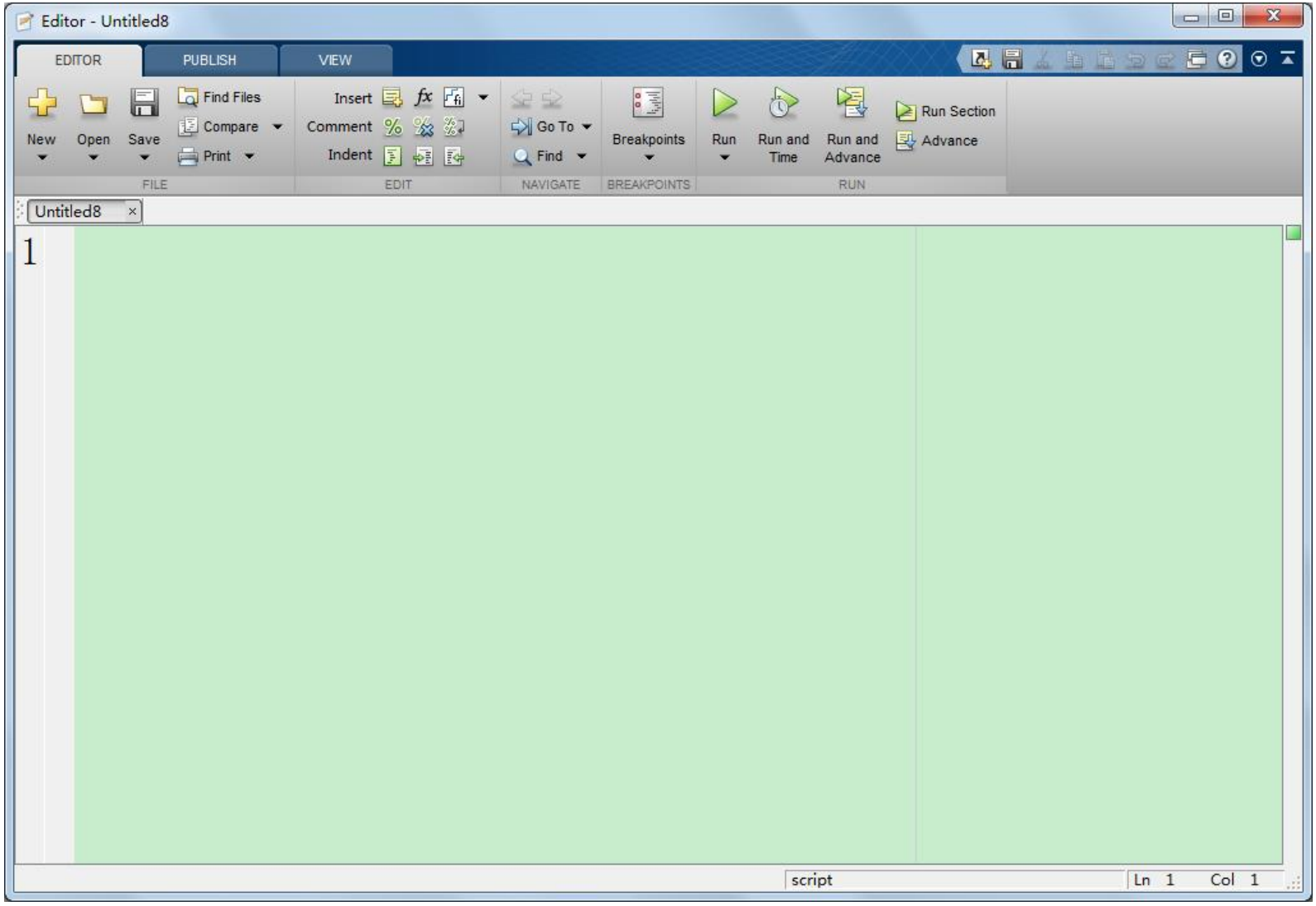
Programming with MATLAB

M-file Script



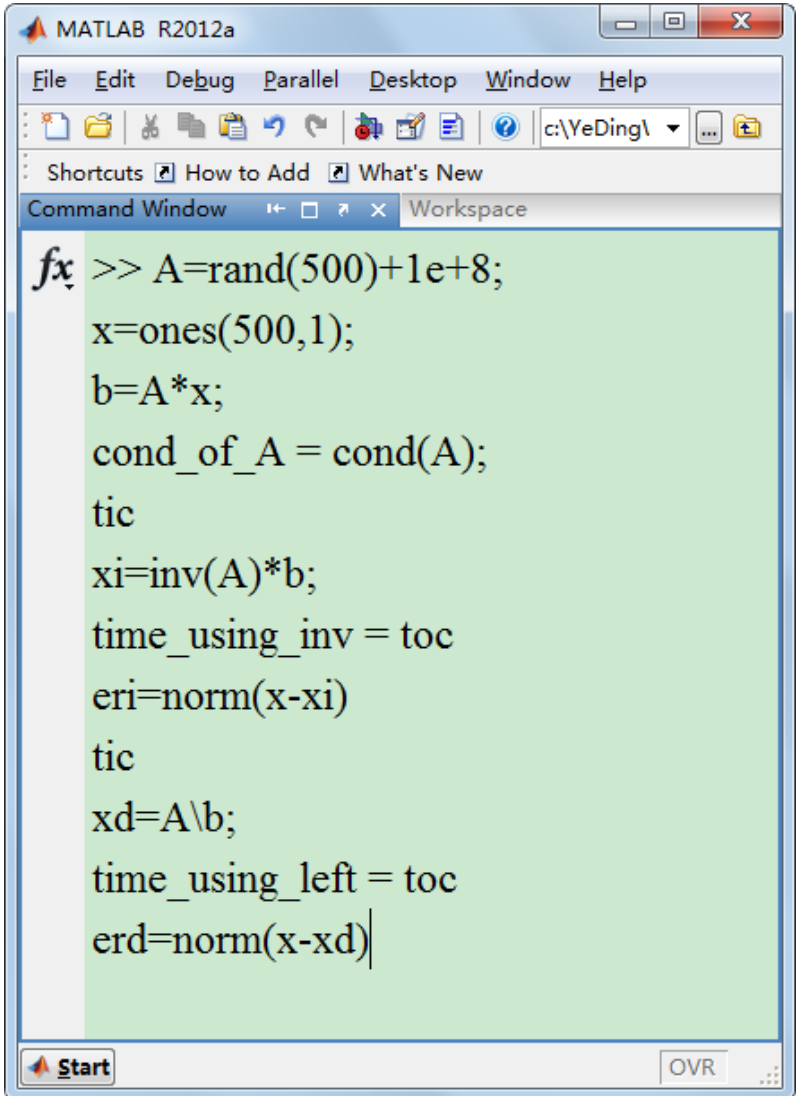
Programming with MATLAB

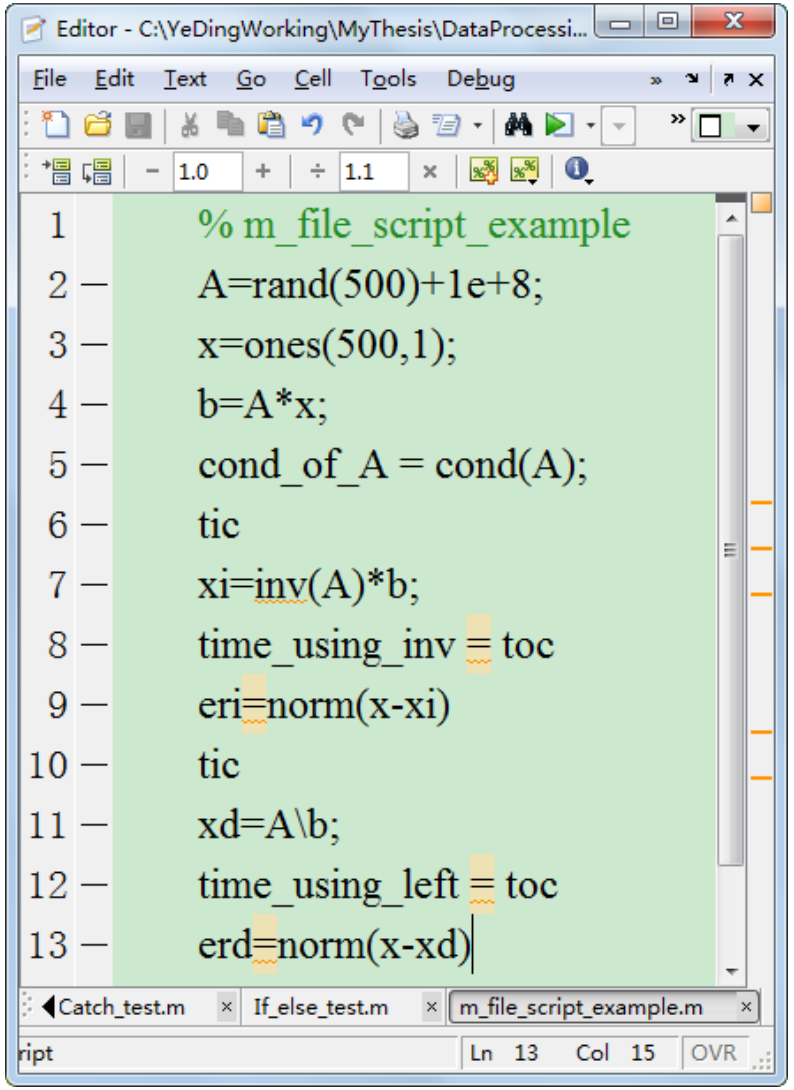
M-file Script



Programming with MATLAB

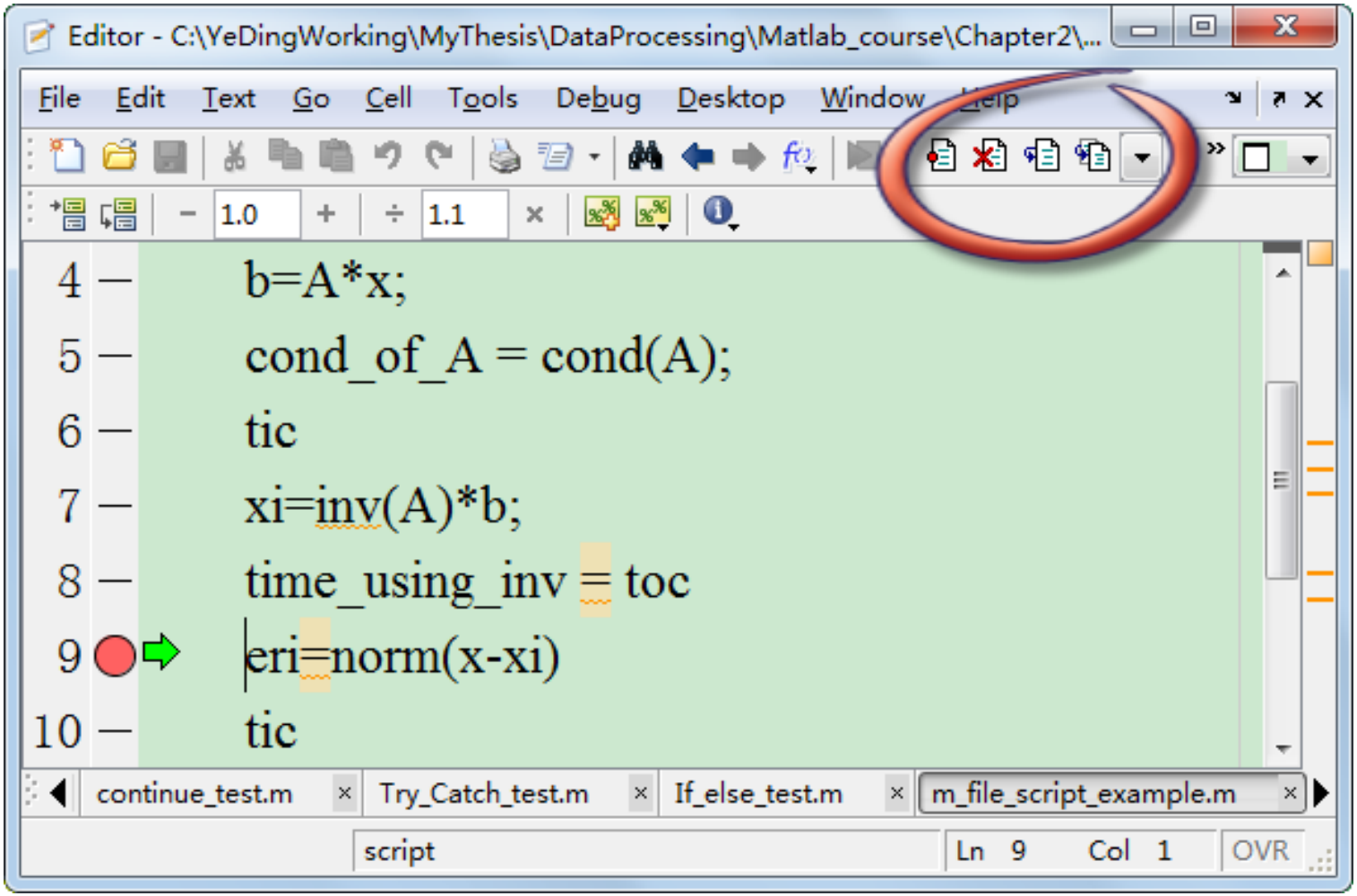
M-file Script: Example





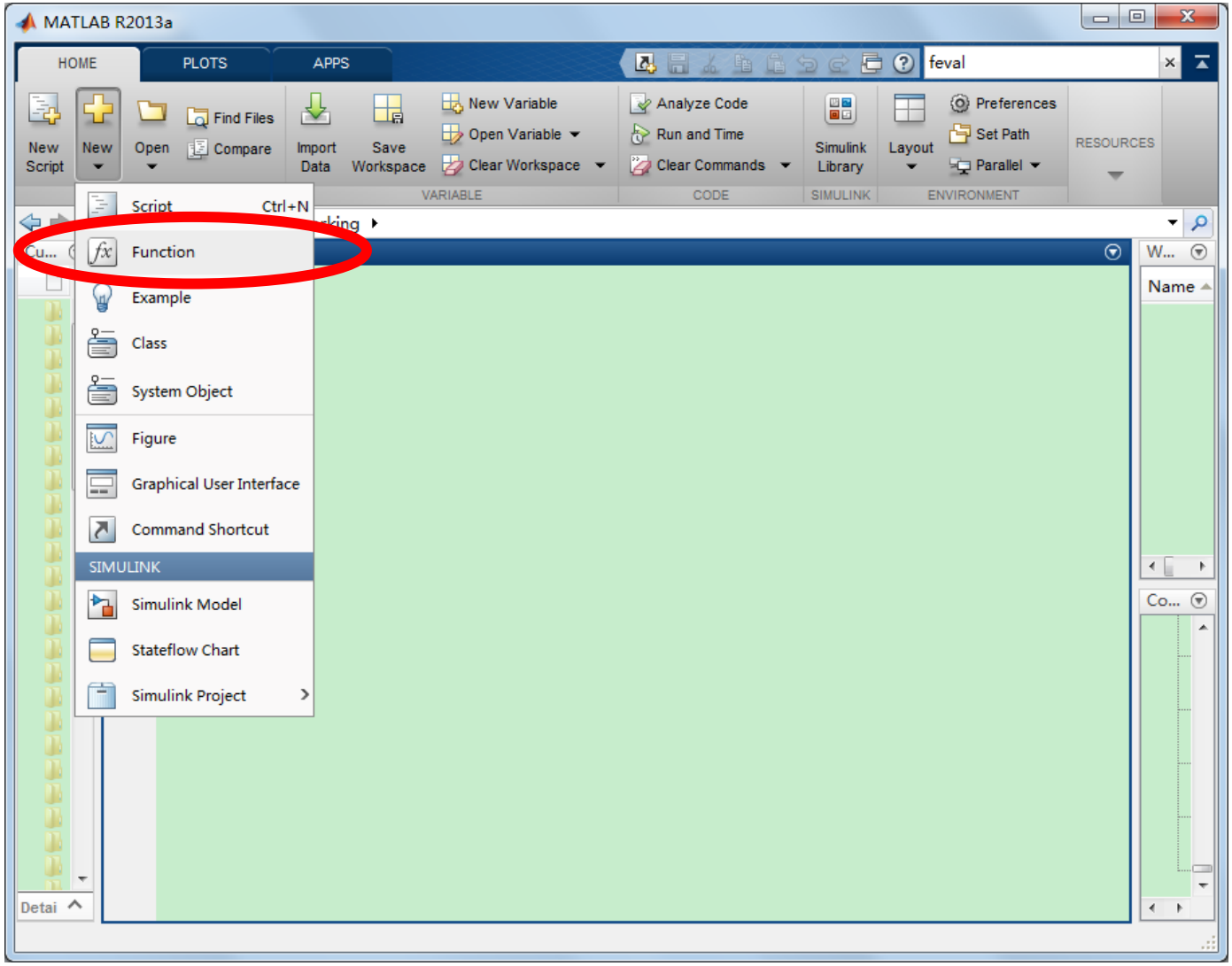
Programming with MATLAB

- **M-file Script: Debug tool**



Programming with MATLAB

M-file Function



Programming with MATLAB

M-file Function

MATLAB allows users to define their own functions by constructing an M-file in the M-file Editor/Debugger. The first line of a function has the form:

```
function output_parameters = function_name (input_parameters)
the function body
```

Three useful commands:

- ✓ help function_name
- ✓ type function_name
- ✓ edit function_name

Programming with MATLAB

M-file Function: Example 1

```
function y = fliplr(x)
%FLIPLR Flip matrix in left/right direction.
%   FLIPLR(X) returns X with row preserved
and columns flipped
%   in the left/right direction.
%
%   X = 1 2 3     becomes  3 2 1
%       4 5 6              6 5 4
%
if ~ismatrix(x)
  error(message('MATLAB:fliplr:SizeX'));
end
y = x(:,end:-1:1);
```

Programming with MATLAB

M-file Function: Example 2

```
function y = m_function_demo(x)
% Definition of a simple function
y=2*x.^3-3*x+1;
```

```
>> y = m_function_demo(2)
y =
    11
>> y = m_function_demo([1:1:3])
y =
     0    11    46
```

Programming with MATLAB

M-file Function: Example 3

```
function y = m_function_demo_a(x,a)
% Definition of a simple function
y=a*x.^3-3*x+1;
```

```
>> y = m_function_demo_a (2,2)
y =
    11
>> y = m_function_demo_a ([1:1:3],2)
y =
     0    11    46
```

Programming with MATLAB

M-file Function: Example 4

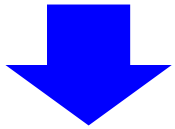
```
function [y,y2] = m_function_demo_a_y2(x,a)
% Definition of a simple function
y=a*x.^3-3*x+1;
y2 = y.*y;
```

```
>> [y,y2] = m_function_demo_a_y2 (2,2)
y =
    11
y2 =
   121
```

Programming with MATLAB

Anonymous Function (匿名函数)

```
function y = m_function_demo(x)
% Definition of a simple function
y=2*x.^3-3*x+1;
```



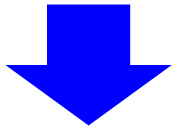
```
>> y = @(x) 2*x.^3-3*x+1;
>> y_list = y([0:0.01:1])
```

Computational Efficiency?

Programming with MATLAB

- **Anonymous Function: Application**

$$(e^y + x^y)^{\frac{1}{y}} - x^2 y = 0$$



```
>> y = @(x) fzero(@(y) (exp(y) + x^y)^(1/y) - x^2 * y,x);
>> y1 = y(1)
y1 =
   2.777942350124938
```

Programming with MATLAB

- **Nested Function (嵌套函数)**

```
function r = NestedFunctionDemo(a)
b = a + 1;
    function Nested_1
        c = b + 1;
        function Nested_2
            d = c + 1;
        end
        Nested_2
    end
Nested_1
r = d;
end
```

Programming with MATLAB

- **Input & Output**

Opening and Closing Files

fid = fopen ('file' , 'permission')

Opens the file for the given permission type.

fclose (fid)

Closes the identifier fid file if it is open.
Returns 0 if the process has been performed successfully and -1 otherwise.

Programming with MATLAB

- **Input & Output**

Reading and Writing Formatted ASCII Text Files

fprintf(fid, 'format' , A,. . .)

Writes the specified items in A (which in general is an array) in the file identifier fid (previously opened) with the format specified in 'format'

[A, count] = fscanf(fid, 'format' , size)

Reads the data from the file identifier fid with the specified size and format, and writes them to a matrix A of dimension size and whose number of elements is count.

Programming with MATLAB

- **Input & Output**

Reading and Writing Formatted ASCII Text Files

```
% Open_Close_test
x = 0:.1:1;
y= [x; exp(x)];
fid=fopen('exponen.txt','w');
fprintf(fid,'%6.2f  %12.8f\n', y);
fclose(fid)
```

Programming with MATLAB

- **Input & Output**

Reading and Writing Binary Files

fwrite(fid, A, precision)

Writes the specified items in A (which in general is an array) in the file identifier fid (previously opened) with the specified accuracy.

[A, count] = fread(fid, size, precision)

Reads the data from the file identifier fid with the dimension specified in size and precision given by precision, and writes them to a matrix A of dimension size and whose total number of elements is count.

Programming with MATLAB

- **Input & Output**

Reading and Writing Binary Files

```
%  Open_Close_test_fwrite.m
x = 0:.1:1;
y= [x; exp(x)];
fid=fopen('exponen.bin','w');
fwrite(fid,y, 'double');
fclose(fid)
```

Programming with MATLAB

- **Input & Output**

Direct Input:

input ('string')
Displays the string on the screen and waits for a key press to continue.

load('filename.mat')
Reads all variables from the file filename.mat.

C = textscan(fid, 'format', N)
Read formatted data from text file or string.

Programming with MATLAB

Relational operators

The relational operators in MATLAB are

==	equal
<=	less than or equal
>=	greater than or equal
~=	not equal
<	less than
>	greater than

Programming with MATLAB

Relational operators

Examples:

```
>> A = magic(3); A == 7
ans =
     0     0     0
     0     0     1
     0     0     0
```

```
>> A = 2:8; A~=9
ans =
     1     1     1     1     1     1     1
```

Programming with MATLAB

Logical operators

The logical operators in MATLAB are

&	and
|	or
~	not
xor	exclusive

Programming with MATLAB

Logical operators

Examples:

```
>> A = 2:7;P =(A>3) &(A<6)
P =
     0     0     1     1     0     0
```

```
>> A = 2:7;P =(A<3) | (A>6)
P =
     1     0     0     0     0     1
```

Programming with MATLAB

- **Logical operators**

  Examples:

```
>> A = [0 0 pi 2]; B = [0 -2.4 0 1];
>> C = xor(A,B)
C =
     0     1     1     0
```

Programming with MATLAB

Logical functions

```
>>A=[1 2 3; 4 5 6; 7 8 0];    find(A>=5)'
ans=   3     5     6     8
>> [i,j]=find(A>=5); [i,j]
ans=   3     1
       2     2
       3     2
       2     3
>> all(A>=5)      % if all the element of one column is bigger or equal to 5, it is 1
ans= 0     0     0
>> any(A>=5)      % if one element of the column is bigger or equal to 5, it is 1
ans= 1     1     1
```

Programming with MATLAB

- **Logical functions**

```
>> a = [1  2  nan  inf  nan];
>> c = sqrt(a)
c =
   1.0000   1.4142      NaN      Inf      NaN
>> g = isnan(a)
g =
    0    0    1    0    1
>> h = isinf(a)
h =
    0    0    0    1    0
```

Programming with MATLAB

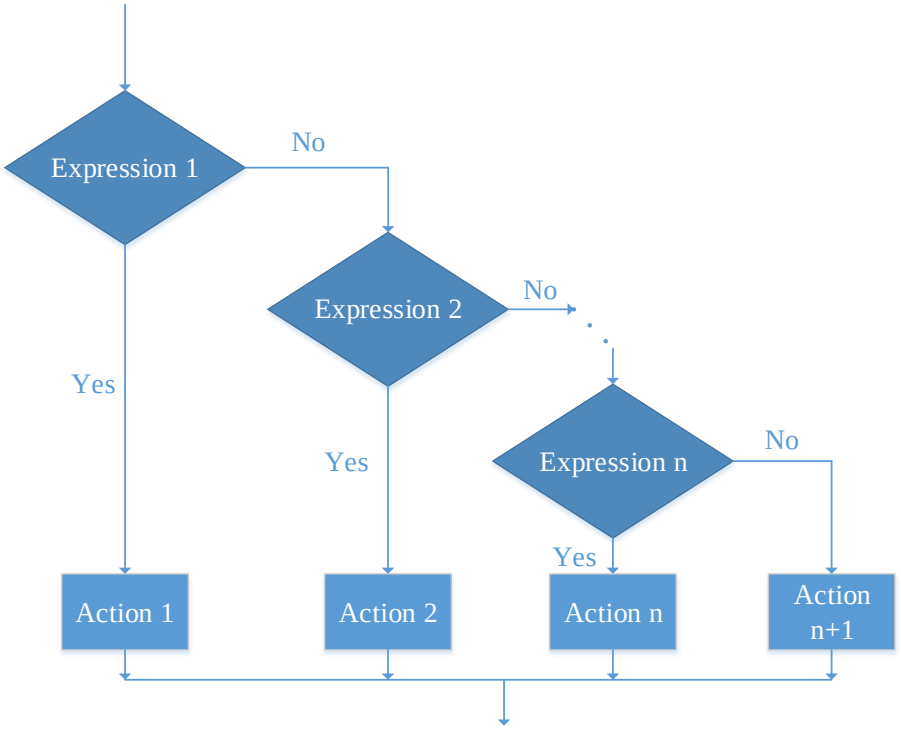
- **Flow Control**
  - Sequential Structure
  - Selective Structure (If elseif, switch)
  - Cycle Structure (for, while)
  - Try and Catch Structure

Programming with MATLAB

Selective Structure

The most useful selective structures in MATLAB are

if elseif and *switch*.




```
if (expression)
            commands ...
elseif (expression)
            commands ...
else
            commands ...
end
```

Programming with MATLAB

Selective Structure - Example

- To calculate the area of a triangle:

```
A=input('Input triangle side lengths:');
if A(1)+A(2)>A(3) & A(1)+A(3)>A(2) &
A(2)+A(3)>A(1)
  p=(A(1)+A(2)+A(3))/2;
  s=sqrt(p*(p-A(1))*(p-A(2))*(p-A(3)));
  disp(s)
else
  disp('It is NOT a triangle!')
end
```

Programming with MATLAB

Selective Structure - Example

- To test a number:

```
if isnan(x)
    disp('Not a Number')
elseif isinf(x)
    disp('Plus or minus infinity')
else
    disp('A "regular" floating point number')
end
```

Programming with MATLAB

Selective Structure - switch

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

Programming with MATLAB

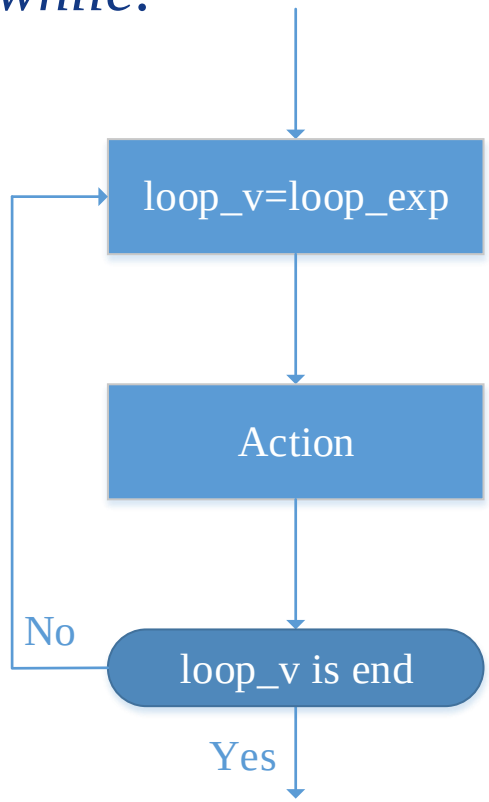
Selective Structure - Example

```
function y = norm_switch(x,p)
switch p
        case 1
                y = sum(abs(x));
        case 2
                y = sqrt(x'*x);
        case inf
                y = max(abs(x));
        otherwise
                error('p must be 1, 2 or inf.')
end
```

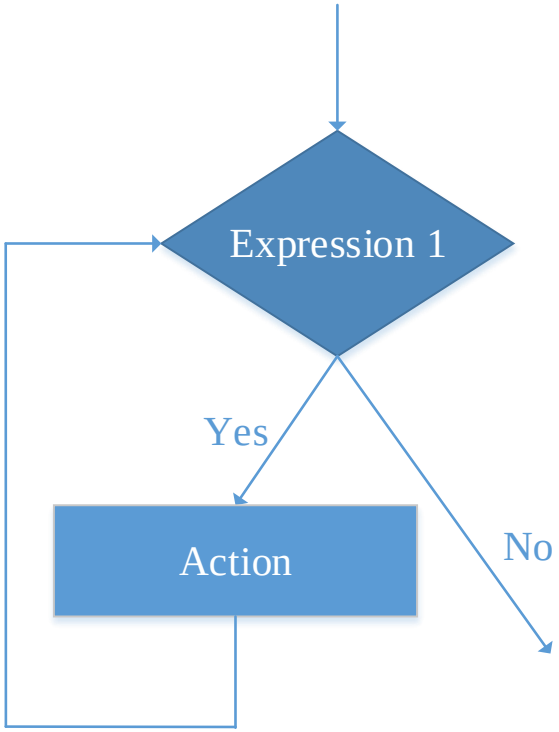
Programming with MATLAB

Cycle Structure

for and *while*.



(a) for



(b) while

MATLAB does not have a "repeat-until" loop

Programming with MATLAB

Cycle Structure: Motivation

```
1  n = 15;
2  y = zeros(n,1);
3  y(1) = 1;
4  z = input('Enter z = ');
5  for i = 1:n-1
6      y(i+1) = 2*y(i)/3 + z/(3*y(i)^2);
7  end
8  figure(1)
9  plot(1:n,y,'-ro')
10 title('Iterative method for cube roots')
11 xlabel('Iteration number')
12 ylabel('Approximate cube root')
```

Programming with MATLAB

Cycle Structure

```
>> s=0;
    for i=1:100
        s=s+i;
    end

 s=   5050
```

```
>> s=0;  i=1;
        while i<=100
            s=s+i;
            i=i+1;
        end
```

```
>> s=0; m=0;
    while s<1000
            m=m+1;      s=s+m;
    end
s=1035;         m=45;
```

Programming with MATLAB

Cycle Structure – Example 1

```
>> for  i=1:10;
       for j=1:10;
           if i==j
              A(i,j)=1;
           else
             A(i,j)=0;
           end
        end
   end
```

```
A=
     1     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0     0
     0     0     0     1     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     0     0     1
```

Computational Efficiency?

Programming with MATLAB

- **Cycle Structure – Example 2.1**

```
tic;
nSize = 10000;
for n=1:nSize
    x(n)=sin(n*pi/nSize);
end
Time1 = toc
```

VS

```
tic;
m = 1:nSize;
y = sin(m*pi/nSize);
Time2 = toc
```

Computational Efficiency?

Programming with MATLAB

Cycle Structure – Example 2.2

```
tic;
nSize = 100000;
s = 0;
for n = 1:nSize
        s = s + 1/n^2;
end
Time1 = toc
```

Vs

```
tic;
nSize = 100000;
n = 1:nSize;
s = sum(1./n.^2);
Time2 = toc
```

Computational Efficiency?

Programming with MATLAB

- **Cycle Structure – Example 3**

```
function for_test
n = 30000;
tic;
for k = 1:n
    a(k) = 1;
end
time1 = toc
```

```
tic;
b = zeros(1,n);
for k = 1:n
    b(k) = 1;
end
time2 = toc
```

time1/time2 = ???

Programming with MATLAB

- **Local and Global Variables**

```
function for_test
global time1
n = 30000;
tic;
for k = 1:n
    a(k) = 1;
end
time1 = toc
```

```
global time2
tic;
b = zeros(1,n);
for k = 1:n
    b(k) = 1;
end
time2 = toc
```

time1/time2 = ???

Programming with MATLAB

Cycle Structure – Example 4. Anonymous Function

```
y = @(x) fzero(@(y) (exp(y) + x^y)^(1/y) - x^2 * y,x);
x_list = linspace(1,10,100);
x_list_len = length(x_list);
y_list = zeros(1,x_list_len);
for k = 1 : x_list_len
    y_list(k) = y(x_list(k));
end
plot(x_list,y_list,'ro-.')
```

Programming with MATLAB

Cycle Structure – Example 5. Visit Speed Test

```
function TimeRatio = VisitSpeedTest()
n = 1e+5; a = [1,2;3,4]; b = {1,2;3,4};
tic;
for k = 1 : n
    c = a(1);
end
time1 = toc
tic;
for k = 1 : n
    c = b{1};
end
time2 = toc
TimeRatio = time2/time1
```

Programming with MATLAB

- **Cycle Structure – break and continue**

"break" allows loops to stop when certain conditions are met.

```
x = 1;
while (1==1)
    x = x+1;
    if x>10
        break;
    end
end
```

Programming with MATLAB

- **Cycle Structure – break and continue**

"continue" statement passes control to the next iteration in a for loop or while loop

```
x = 1:10;
sum_x =0;
num_x = length(x);
for k = 1:num_x
    if x(k) == 7
        continue;
    end
    sum_x = sum_x+x(k);
end
```

Programming with MATLAB

Try and Catch Structure

The instructions between try and catch are executed until an error occurs. The instruction *lasterr* is used to show the cause of the error.

```
try,
    instruction
    ...,
    instruction
catch,
    instruction
    ...,
    instruction
end
```

Programming with MATLAB

- **Application: Function Handle**

```
function y = fd_derivative(fun,x,h)
% fd_derivative(FUN,X,H) is a finite difference
% approximation to the derivative of function FUN at X
% with difference parameter H. H defaults to SQRT(EPS).
if nargin < 3
    h = sqrt(eps);
    if nargin < 2
        x = 0;
    end
end
y = (fun(x+h) - fun(x))/h;
```

Programming with MATLAB

- **Application: Fibonacci number**

$$f_n = f_{n-1} + f_{n-2} \quad with\ f_1 = 1,\ f_2 = 2$$

```
function f = fibonacci(n)
% FIBONACCI generates
% the first n Fibonacci
% numbers
f = zeros(n,1);
f(1) = 1;
f(2) = 2;
for k = 3:n
    f(k) = f(k-1) + f(k-2);
end
```

```
function f = fibnum(n)
% FIBNUM Fibonacci number.
% FIBNUM(n) generates the
% nth Fibonacci number.
if n <= 1
   f = 1;
else
   f = fibnum(n-1) + fibnum(n-2);
end
```

A recursive function

Thank You !