Lecture 5 MATLAB programming (3)

Dr. Tony Cahill
Objectives

• User-defined Functions
A Simple Example: factorial

• Write a program to calculate the factorial:

\[ N! = \begin{cases} \ 1, & \text{if } N=0 \\ \ N \times (N-1) \times (N-2) \times \cdots \times 3 \times 2 \times 1, & \text{if } N>0 \end{cases} \]

```matlab
n=input('Please enter N:');
n_factorial=1;
for ii=1:n
    n_factorial=n_factorial*ii;
end
disp ([ num2str(n), '!=', num2str(n_factorial) ]);```

A Simple Example: factorial

• What if I want to automatically calculate 0!, 1!, 2!, 3!, 4! and 5!?

Using nested loop:

```matlab
for n=0:5
    n_factorial=1;
    for ii=1:n
        n_factorial=n_factorial*ii;
    end
    disp ([ num2str(n),'!=',num2str(n_factorial) ]);
end
```
A Simple Example: factorial

• We can develop a general MATLAB function that calculates $N!$, given $N$.

List of `factorial.m`

```matlab
function y = factorial( n )
  \%FACTORIAL Calculate the factorial of n
  \% Function FACTORIAL calculates the factorial of n:
  \% n!=1*2*...*(n-1)*n
  \%
  \% Written by: Qi Ying (Jan 2011)
  \% CVEN 302
  y=1;
  for ii=1:n  \% calculate N!
    y=y*ii;
  end
end
```
A Simple Example: factorial

• Calling the user-defined function:

```matlab
List of test1.m
% demonstrates how to call the user
% defined function 'factorial'
for ii=0:5
    n_factorial=factorial(ii);
    fprintf ('%d!=%d
',ii,n_factorial);
end

>> test1
0!=1
1!=1
2!=2
3!=6
4!=24
5!=120
```
A Simple Example: factorial

• Some definitions:

```matlab
function y = factorial( n )

%FACTORIAL Calculate the factorial of n
% Function FACTORIAL calculates the factorial of n:
% n!=1*2*...*(n-1)*n

% Written by: Qi Ying (Jan 2011)
% CVEN 302

  y=1;
  for ii=1:n
    y=y*ii;
  end
end
```

1. The M-file is called factorial.m.
2. n is called the **input argument**
3. y is called the **output argument**
4. The first comment line is called the **H1 comment line**, which is searchable by the ‘lookfor’ command
5. The remaining comment lines until the first blank line or executable line is displayed by the ‘help’ command
6. Loop variable `ii` is a ‘**local**’ variable, only visible inside the function.
General form of a MATLAB function

```matlab
function [outarg1, outarg2, ...] = fname (inarg1, inarg2, ...)
%H1 comment line
%   other comment lines
%
% more comment lines

(executable code)
...
end
```
function [ mean, std_dev ] = stat1( x )
%STAT1 Calculate the mean and standard deviation of x.
    sum_x=0; sum_x2=0;
    n=length(x);  % number of data points in x
% loop over each data point in x to calculate sum(x) and sum(x^2)
    for ii=1:n
        sum_x=sum_x+x(ii);
        sum_x2=sum_x2+x(ii)^2;
    end
% return the mean and standard deviation
    mean=sum_x/n;
    std_dev=sqrt((n*sum_x2-sum_x^2)/(n*(n-1)));
end % end function avg1
Another example: statistics

List of test3.m

```matlab
x=[1 2 3 4 5]
[m1,s1]=stat1(x);
fprintf('mean=%f, std=%f\n',m1,s1);
% generate some random numbers
x=rand(1,5)
[m2,s2]=stat1(x);
fprintf('mean=%f, std=%f\n',m2,s2);
```

Results

```matlab
>> test3
x =
    1   2   3   4   5
mean=3.000000, std=1.581139
x =
   0.1419   0.4218   0.9157   0.7922   0.9595
mean=0.646217, std=0.352429
```
Variable Passing in MATLAB

• Remember: MATLAB communicate with the functions using a ‘pass-by-value’ scheme. See the following example:

List of func1.m

```matlab
function y = func1( x )
    fprintf('In the beginning of func1, x=%f
',x);
    for ii=1:3
        x=x+1;
        fprintf('In func1: x=%f
',x);
    end
end
```

List of test2.m

```matlab
x=5;
fprintf('Before func1, in the main program x=%f
',x);
func1(x);
fprintf('After func1, in the main program x=%f
',x);
```
Variable Passing in MATLAB

>> test2

Before func1, in the main program x=5.000000
In the beginning of func1, x=5.000000
In func1: x=6.000000
In func1: x=7.000000
In func1: x=8.000000
After func1, in the main program x=5.000000

Output from func1

Pass-by-value is important for data isolation

The value of x in the main program (test2.m) is passed into the function func1 to the first input argument, x.

However, the argument x in the function is isolated from the main program. No changes of the argument will be visible to the main program.
In-class Exercise

• Write a MATLAB function that returns the minimum and maximum values in a 2D array
One solution

function [ r ] = minmax( a )
% MINMAX finds the minimum and maximum values in an array

n=numel(a); % number of total elements in an array
% r(1) saves the minimum value
% r(2) saves the maximum value
r(1)=realmax;   % maximum possible float number -> r(1)
r(2)=realmin;   % minimum possible float number -> r(2)
for ii=1:n
    if a(ii)>r(2)    % if we find a larger number
        r(2)=a(ii);
    end
    if a(ii)<r(1)    % if we find a smaller number
        r(1)=a(ii);
    end
end
end % end of function minmax