Lecture 11

Lecture 11: MATLAB Programs

- **Subfunctions**
  - So far, we have put every function in a separate M-file.

    Example

    rectarea.m (Main script)

```
% This program calculates & prints the area of a rectangle

% call a fn to prompt the user & read the length and width
[length, width] = readlenwid;
% call a fn to calculate and print the area
printrectarea(length, width)
```

    readlenwid.m

```
function [l, w] = readlenwid
% This function prompts the user for the length and width
l = input('Please enter the length: ' );
w = input('Please enter the width: ' );
```

Lecture 11: MATLAB Programs

- **Subfunctions (cont)**

printrectarea.m

```
function printrectarea(len, wid)
% This function calculates & prints the rectangle area

% it calls another function to calculate the area
area = calcrectarea(len,wid);
fprintf('For a rectangle with a length of %.2f\n', len)
fprintf('and a width of %.2f, the area is %.2f\n', wid, area)
```

calcrectarea.m

```
function area = calcrectarea(len, wid)
% This function calculates the rectangle area
area = len * wid;
```

Lecture 11: MATLAB Programs

- **Subfunctions**
  - it is possible to have more than one function in a given M-file. For example, if one function calls another, the first (calling) function would be the primary function, and the function that is called is a subfunction. These functions both could be stored in the same M-file. The name of the M-file is the same as the name of the primary function.

printrectarea.m

Primary function

```
function printrectarea(len, wid)
% This function prints the rectangle area
% it calls a subfunction to calculate the area
area = calcrectarea(len,wid);
fprintf('For a rectangle with a length of %.2f\n', len)
fprintf('and a width of %.2f, the area is %.2f\n', wid, area)
```

Sub-function

```
function area = calcrectarea(len, wid)
% This function calculates the rectangle area
area = len * wid;
```

Can the sub-function be called by functions or scripts other than the primary function?

Lecture 11: Advanced Functions

➢ Anonymous Functions

- Anonymous functions are simple one-line functions that are called using their function handle.
- The advantage of an anonymous function is that it does not have to be stored in an M-file.
- Anonymous functions can be created in the Command Window or in the script.
- The general form of an anonymous function is:

fnhandle = @ (arg1, arg2, …) equation

Lecture 11: Advanced Functions

- **Anonymous Functions**
  - The general form of an anonymous function is:

    fnhandle = @ (arg1, arg2, …) equation

    Similar to the definition of variables

  - Example:

    To calculate and return the area of a circle

```
>> cirarea = @(radius) pi * radius .^2;
>> cirarea(4)
ans =
      50.2655
>> cirarea(1:4)
ans =
      3.1416   12.5664   28.2743   50.2655
```

The anonymous functions are called in the same way as the built-in functions and other user-defined functions stored in M-files.

Lecture 11: Advanced Functions

- **Anonymous Functions**
  - Example:

    For the equation $f(x) = x^2 + 5x + 3$ we would use

    ```
    >> fx = @(x) x.^2 + 5*x + 3;
    ```

    to evaluate the function at x = 2, we would use

    ```
    >> fx(2)
    ans =
        17
    ```

Lecture 11: Advanced Functions

- **Anonymous Functions**
  - Example:

    For the equation $f(x,y) = x^2 + 5xy + 3y - 2$

```
>> fxy = @(x,y) x.^2 + 5*x.*y + 3*y -2
>> fxy(1,2)
ans =
    15
```

Can we change the order of x and y in the parentheses?

What if we define the function handle as fxy = @(x) x.^2 + 5*x.*y + 3*y -2 ?

Lecture 11: Advanced Functions

- **Anonymous Functions**
  - Example:

    For the equation $f(x,y) = x^2 + 5xy + 3y - 2$

```
>> fxy = @(x,y) x.^2 + 5*x.*y + 3*y -2

>> xx = [1  2  3];
>> yy = [4  5  6];
>> fxy(xx,yy)
ans =
     31   67   115
>> fxy(yy, xx)
ans =
   37   79   133
```

Lecture 11: Advanced Functions

- **Anonymous Functions**
  - Unlike functions stored in M-files, if no argument is passed to an anonymous function, the parentheses must still be in the function definition and in the function call.

  - Example: return a random real number as well as a call to this function.

```
>> rtran = @ () rand;
>> rtran()
0.95
```

Can we just type the name to call the function? What will happen?

Lecture 11: Advanced Functions

➢ Inline Functions

- Inline functions are essentially the same as anonymous functions, but are defined with a different syntax:
- The general form of an inline function is:

  fnhandle = inline('equation', 'arg1', 'arg2', …)

Anonymous function
fnhandle = @ (arg1, arg2, …) equation

- Example:

  For the equation $f(x) = x^2 + 5x + 3$ we would use

  ```
  >> fx = inline('x.^2 + 5*x + 3', 'x');
  ```

  to evaluate the function at x = 2, we would use

  ```
  >> fx(2)
  ans =
  17
  ```

For inline functions, the equations and input arguments should be put in quotation marks

Lecture 11: Advanced Functions

- **Inline Functions**
  - Example:

    For the equation $f(x) = x^2 + 5x + 3$ we would use

```
>> fx = inline('x.^2 + 5*x + 3', 'x');
>> x = [1   2   3];
>> y = fx(x) + fx(x+1)
y =
    26    44    66
```

Two function handles: anonymous functions and inline functions

Lecture 11: Advanced Functions

- **Function handle can be passed to a function program**

  Example: given a vector $v = (v_1, v_2 \ldots)$, and a function $f(x) = x^2 + 5x + 3$, calculate $y = \sum_i f(v_i)$

```
clc
clear
v = [1 2 3 4]';
f = @(x) x.^2 + 5*x + 3;
y = usef(v,f);
fprintf('\nThe value of y is: %.2f\n\n',y)
```

```
function y = usef(v,f)
n = length(v);
sm = 0
for i = 1:n
    sm = sm +f(v(i));
end
y = sm;
```

Lecture 11: Advanced Functions

➢Summary

- Subfunctions
- Function handle: anonymous functions and inline functions

➢Homework on Canvas