MATLAB

Creating and Executing a Function File

Instructor: Dr. G. L. Prajapati
IET-DAVV

Constructing Circles

- Create a script file to construct a unit circle.

```
>>theta=linspace(0,2*pi, 100);
>> x=cos(theta);
>>y=sin(theta);
>>plot(x, y);
>>axis('equal');
>>title('Circle of Unit Radius')
```

Constructing Circles

- Create a script file to construct a circle of any radius. Take radius from user.
- **>> r =input('Enter the radius of the Circle: ')**

Functions

Three forms of the use of a function in Matlab are:

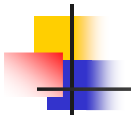
```
>> VAR = function_name(arg1,arg2, …);
>> [VAR1,VAR2,...] = function_name(arg1,arg2, …);
>> function_name(arg1,arg2, …);
```

A Matlab function like a mathematical function is a rule where given a certain input or inputs, the rule tells you how to compute the output value or how to produce an effect (e.g. the plot function produces a figure).

The inputs are called the "arguments" to the function.

Functions - User Defined

Users can add to Matlab's built-in functions by "programming" a function.
A program in Matlab is either in the form of a script or a function.



Why define your own Functions?

- **Modular Programming**
  Break complicated tasks up into pieces(functions).
- **Functions can "call" other functions.**
  This means you don't have to re-write the code for the function again and again.
- **Variables in Functions are "local".**
  All variables in the function are " local " by default. That means that if you have a variable in your workspace with the same name as the variable in a function, then assigning a value to the variable in the function has no affect on the variable in the workspace. That is, a function cannot accidentally change (destroy) the data in your workspace.

Programming Steps

1. Problem Definition
2. Refine, Generalize, Decompose the problem definition
   (i.e., identify sub-problems, I/O, etc.)
3. Develop Algorithm
   (processing steps to solve problem)
4. Write the "Program" (Code)
   (instruction sequence to be carried out by the computer)
5. Test and Debug the Code
6. Run Code

(Simple) Function - Structure

```
function [output_variables]  = function_name(input_variables);
```

% The line above is called the function definition line.
% Simple functions have only one output variable.
% A line that begins with a "%" symbol is called a comment.
% Comments are ignored by Matlab but necessary for humans.

{ Body of the function }

Note: One statement in the body of the function should be of the form,

output_variable = expression;

Functions - An Example

Problem Definition

Write a function that construct a circle of given radius.

```
function [x,y] = mycircle(r);
theta=linspace(0,2*pi, 100);
x=r*cos(theta);
y=r*sin(theta);
plot(x, y);
axis('equal');
title('Circle of Radius r')
```

Save the file with the name of function As "mycircle.m".

Functions - An Example

Executing a function file

```
>> R=5;
>> [x,y] =mycircle(r);
```

```
>> [cx,cy] =mycircle(2.7);
```

```
>> mycircle(2);
```

```
>> mycircle(2*3-1);
```

Function – Definition Line

Function Definition Line	*File Name*
`function [rho,H,F] = motion(x,y,t);`	motion.m
`function [theta] = angleTH(x,y);`	angleTH.m
`function theta = THETA(x,y,z);`	THETA.m
`function [] = circle(r);`	circle.m
`function circle(r);`	circle.m

Functions - An Example

```
function [n] = factorial (k)
% The function [n] = factorial(k) calculates and
% returns the value of k factorial. If k is negative,
% an error message is returned.
if (k < 0)  n = 'Error, negative argument';
elseif k<2  n=1;
else
   n = 1;
   for j = [2:k]   n = n * j;   end
end
```

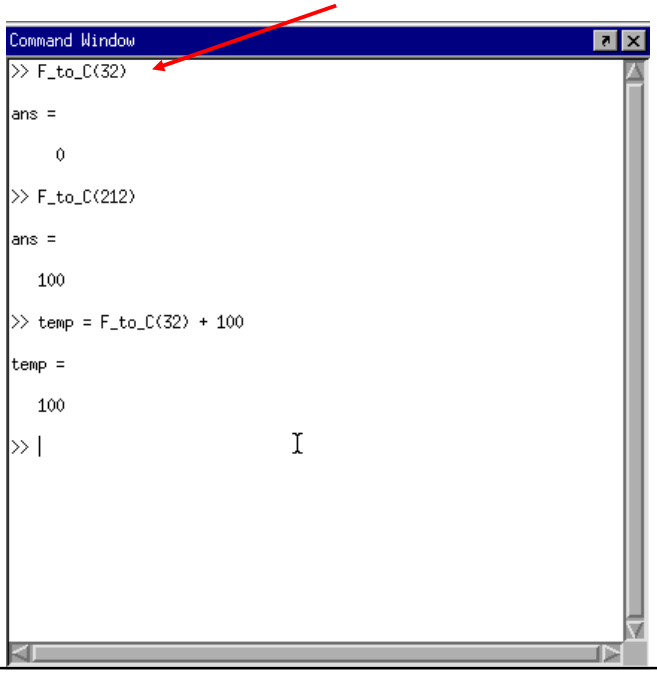
Function Example

4. Write the for converting Fahrenheit to Celsius

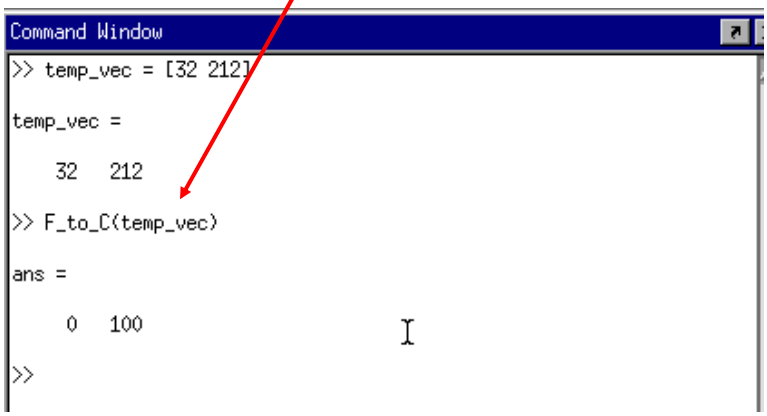
```
function celsius = F_to_C(fahr)
% This function converts Fahrenheit to Celsius.
celsius = (fahr -32)*5/9;
```

Click Save to name the file "F_to_C.m".

Execute(call) the function in by typing " F_to_C(temperature)"



Input is a vector [32 212]



Documentation

Improved documentation for the F_to_C function.

```
function celsius = F_to_C(fahr);
% function celsius = F_to_C(fahr)
% This function converts Fahrenheit to Celsius.
% input fahr can be a scalar or vector of temps in degree F
% output celsius is a scalar or vector of temps in  degree C
% Programmer: Tom Gambill
% Date: 1/30/01
celsius = (fahr -32)*5/9;
```

Anonymous Functions

Anonymous functions enable you to create a simple function without needing to create an M-file for it. You can construct an anonymous function either at the MATLAB command line or from within another function or script. The syntax for creating an anonymous function from an expression is

```
fhandle = @(arglist) expr
```

where `arglist` is a comma-separated list of input arguments to be passed to the function, and `expr` is any single, valid MATLAB expression.

Anonymous Functions (continued)

To create a simple function called `sq` to calculate the square of a number, type

```
>>sq = @(x) x.^2;
```

To improve readability, you may enclose the expression in parentheses, as `sq = @(x) (x.^2);`. To execute the function, type the name of the function handle, followed by any input arguments enclosed in parentheses. For example,

```
>>sq([5,7])
ans =
   25      49
```

Anonymous Functions (continued)

You might think that this particular anonymous function will not save you any work because typing `sq([5,7])` **requires nine keystrokes, one more than is required to type** `[5,7].^2`**.**

Here, however, the anonymous function protects you from forgetting to type the period (.) required for array exponentiation.

Anonymous functions are useful, however, for more complicated functions involving numerous keystrokes.

Anonymous Functions (continued)

You can pass the handle of an anonymous function to other functions. For example, to find the minimum of the polynomial $4x^2 - 50x + 5$ over the interval [–10, 10], you type

```
>>poly1 = @(x) 4*x.^2 - 50*x + 5;
>>fminbnd(poly1, -10, 10)
ans =
    6.2500
```

If you are not going to use that polynomial again, you can omit the handle definition line and type instead

```
>>fminbnd(@(x) 4*x.^2 - 50*x + 5, -10, 10)
```

Multiple Input Arguments

You can create anonymous functions having more than one input. For example, to define the function $\sqrt{(x^2 + y^2)}$, type

```
>>sqrtsum = @(x,y) sqrt(x.^2 + y.^2);
```

Then type

```
>>sqrtsum(3, 4)
ans =
    5
```

As another example, consider the function defining a plane, $z = Ax + By$. The scalar variables A and B must be assigned values before you create the function handle. For example,

```
>>A = 6; B = 4:
>>plane = @(x,y) A*x + B*y;
>>z = plane(2,8)
z =
  44
```

Calling One Function within Another

One anonymous function can call another to implement function composition. Consider the function 5 sin(x^3). It is composed of the functions $g(y) = 5\sin(y)$ and $f(x) = x^3$. In the following session the function whose handle is `h` calls the functions whose handles are `f` and `g`.

```
>>f = @(x) x.^3;
>>g = @(x) 5*sin(x);
>>h = @(x) g(f(x));
>>h(2)
ans =
    4.9468
```

Nested Function

```
function [ dream_time ] = inception( time )
%INCEPTION Calculates the amount of dream time occured
%   during a given amount of real time
%
%   [ dream_time ] = inception ( time )
%       Converts real-time minutes to dream-time minutes

dream_time=time*12;
deeper;
    function deeper
        dream_time=dream_time*12;
        deeper;
        function deeper
            dream_time=dream_time*12;
            deeper;
            function deeper
                dream_time=dream_time*12;
            end
        end
    end
end
```

Nested Function

```
function [ avg, med ] = calc_stats( in )
%CALC_STATS Calculates the average and median of the input vector.
%
%   [ avg, med ] = calc_stats( in )
%       Calculates the average and median of in.

    parse_input(in);
    N = length(in);
    avg=calc_avg(N);
    med=calc_med(N);

    function [avg] = calc_avg(N)
        avg = sum(in) / N; % Calculate average
    end

    function [med] = calc_med(N)
        in = sort(in);
        if rem(in,2)==1 % Odd number of elements
            med = in( (N+1)/2 ); % median is middle value
        else % Even number of elements
            med = (in(N/2) + in(N/2+1)) / 2; % median is average of 2 middle values
        end
    end
end
```

Conclusions

A Matlab function is a sequence of Matlab statements that are entered in a file. The first line of the function called the function definition line and has a specific format. A function is executed whenever it occurs in a Matlab expression that is executing in the Matlab environment.

Matlab function filenames end with the .m extension and the filename must be the same as the function name.

You create your own function by following a six step design procedure.

Functions provide modular design.

Exercise [Write a script file]

- Plot the expressions $y= \sqrt{1-x^2}$ and $y= -\sqrt{1-x^2}$ on the interval [-1, 1]. The two graphs should form the unit circle $x^2 + y^2 =1$. Look up the axis command in help to see how to get the picture to actually look like a circle, rather than an ellipse.

Matlab Script File

```
clear all
clc
domf = linspace(-1, 1, 60);
domg = domf;
f =@(x)(1-x.^2).^(1/2);
g = @(x)((1-x.^2).^(1/2)).*-1;
clf reset
figure(1)
PLOT1 = plot(domf,f(domf));
hold on
PLOT2 = plot(domg,g(domg) );
hold off
```

- title({'Graph of the Circle x^2 + y^2 = 1','by','Foster,Antony'},'Color','red','FontName','mathematica','FontWeight','bold','FontSize',22)
- legend('sqrt(1-x^2)','-sqrt(1-x^2)','Location','NorthEastOutside')
- set(PLOT1,'Color','black','LineWidth',3,'LineStyle',':')
- set(PLOT2,'Color','black','LineWidth',3,'LineStyle','-')

- set(gca,'XTick',[-1:1/2:1],'XMinorTick','on','XTickLabel', {'-1','-1/2','0','1/2','1'},'FontName','mathematica','FontWeight','bold')
- set(gca,'YTick',[-1:1/2:1],'YTickLabel',{'-1','-1/2','0','1/2','1'}, 'YMinorTick','on','FontName','mathematica','FontWeight','bold')
- xlabel('X-Axis','Color','red','FontName','mathematica','FontWeight', 'bold','FontSize',16)
- ylabel('Y-Axis','Color','red','FontName','mathematica','FontWeight', 'bold','FontSize',16)
- axis square equal
- axis([-1 1 -1 1])

Output Graph

Graph of the Circle $x^2 + y^2 = 1$
by
Foster,Antony