MATH.2720 Introduction to Programming with MATLAB
User-Defined Functions (aka Custom Functions)

A. User-Defined Functions

A MATLAB function file is similar to a script file in that it contains MATLAB code. Most function files, however, require input and produce output, like a mathematical function. The first word on the first line of a function file must be `function` This is followed by the name(s) of the output variables, an `=` sign, the name of the function, and the name(s) of the input variables in parentheses.

Here is an example, taken from section 3.3 of the text.

In the File menu on the MATLAB toolbar, click on *New*, then click on *Function.* A box will open in the Editor window with the skeleton of a function file.

Edit the file so it contains the following lines.

```
function distance = LightningDistance( seconds )
% LightningDistance: Estimates distance of lightning strike based on
%   seconds between observer seeing lightning strike and hearing thunder.
%   Input:  seconds -- Measured time between lightning strike and thunder.
%   Output: distance -- Distance to lightning strike, in kilometers.
speedOfSound = 340.29;    % Speed of sound in meters/sec at sea level
distance = (speedOfSound * seconds) / 1000;   % / 1000 yields km
end
```

Save the file using the file name *LightningDistance.* In the command window, type the command

```
>>help LightningDistance
```

Now type the command

```
distKm = LightningDistance(10)
```

B. Scope of Variables within Function Files

Variables used within a function file are known only within the function, not in other files or in the command window. Similarly, variables defined in a script file or in the command window are not known by any functions. One way to make a variable known everywhere is to use the `global` command. Here is a simple example.

Create a function file containing the following lines

```
function yMax = max_height(v0)
%max_height calculates how high a ball rises given its initial velocity
%Input: v0 -- initial velocity of ball in m/s
%Output: yMax -- maximum height of ball
global g
ymax = v0.^2/(2*g);
end
```

In the command window, type the commands

```
>>global g
>>g = 9.81
>>ymax = max_height(40)
```

C. Anonymous Functions

It is possible to define a function within a script file, a function file, or the command window *without* creating a separate function file. Functions defined in this way are called *anonymous functions*. Here is an example of a script file in which an anonymous function is defined.

```
circleArea = @ (r) pi*r^2;
r = input('Enter the radius of a circle: ');
area = circleArea(r);
fprintf('The area of your circle is %6.2f\n',area)
```

The first line of the script file defines a function named `circlArea` which has one input argument (`r`) and one output (the area of a circle of radius `r`.)

The command defining an anonymous function has the form
 `function_name = @ (list of input arguments) function_formula`.

D. Function Functions

Some MATLAB functions require the name of another function as an input argument. For example, the built-in MATLAB function `fzero` finds a root of a function, but you have to tell `fzero` what function you want a root of. Try this example. First create the function file `f.m` defining the function given by $f(x) = x - \cos(x)$:

```
function y = f(x)
y = x - cos(x);
end
```

In the command window, type the command

```
fzero(@f, 1)
```

This will calculate a root of the function `f` near $x = 1$. Notice that the first input argument to `fzero` is the `@` symbol followed directly by the function file name. (Exception to the rule: If your function has been defined as an anonymous function, you don't need the `@` symbol before the function name.)

E. Subfunctions (aka Local Functions)

It is possible to define more than one function within a single function file. If you do this, the first function defined in the file is called the *primary function* and the others are called *subfunctions*. Here is an example of a file containing multiple function definitions.

```
function [circ, area] = CircleMeasurements(radius)
%CircleMeasurements calculates the circumference and area of a circle of
%given radius
%Input: radius -- radius of circle
%Outputs: circ = circumference of circle
%         area = area of circle
circ = Circumf(radius);
area = CircleArea(radius);

    function perim = Circumf(r)
        perim = 2*pi*r;
    end

    function area_of_circle = CircleArea(r)
        area_of_circle = pi*r^2;
    end

end
```

Try executing this command in the command window:

```
[c, a] = circle_measurements(1)
```

Practice Problems

1. Write a function file that takes the height of a person in inches as input and produces the corresponding height in centimeters as output. (1 inch = 2.54 centimeters.) Test your function with the input 65 inches.
2. Write a function file that takes the length and width of a rectangle as inputs and produces the area and perimeter of the rectangle as outputs. Test your function using dimensions of 3 and 2.
3. Modify the script file example in the section on anonymous functions to calculate both circumference and area.
4. Use `fzero` to find a root of the function given by $f(x) = \tan(x) - x$ near $x = 4$.
5. Write a function file that calculates the volume and surface area of a sphere given the radius of the sphere. Use subfunctions to calculate volume and surface area.

Answers to Practice Problems

1. 165.1

2. 6

4. 4.4934