ENME202 Matlab

CUSTOM FUNCTIONS

Syntax for a custom function with single input & output values:

```
function return_variable = function_name(input_variables)
  % return_variable -->  value returned by the function
  % function_name   -->  name of the function
  % input_variable  -->  argument passed to the function
  %
  % function code goes here...
  %
end
```

The function must be saved as a separate m-file with the same name as the function_name itself. For example, the following function must be saved as "timestwo.m":

```
function z = timestwo(y)
  z = y*2;
end
```

Function overview:

A function is a construct that takes one or more inputs (arguments), and returns either nothing or a single entity, which can be a value (of any type) or an array of multiple values (also of any type).

Each function must be saved in a separate m-file!

Let's look at the above function declaration line in detail:

```
function z = timestwo(y)
```

This is the function declaration. This particular function takes a single argument, i.e. a value that is passed to the function and assigned to the variable name y within the function. A single value is returned by the function. This value is whatever is stored in the variable name z. That is, whatever value has been assigned to z at the end of the function is the value that will be returned to the calling code.

The code between the "function..." line and the final "end" keyword can be any valid Matlab code. Generally this code is used to describe how the output (z) is computed from the input (y). When the function code has been evaluated, a numerical value for the output MUST have been calculated from the input value. Otherwise, Matlab will give a warning, and you will get no output from the function.

Variable scope:

The global workspace holds variables declared in our main code (as well as variables declared directly in the command window).

Variables defined in a function (like y and z above) DO NOT EXIST IN THE WORKSPACE. You can NEVER directly access these private variables from the global workspace. We say that the variables are "local" to the function (and that the "variable scope" is limited to the function).

Functions cannot see anything in the main workspace, and the workspace cannot see any variables contained inside of functions. The only way to get values into a function is through the function inputs, and the only way to get values out of a function is by having the function return a value back to the main code.

As a result, the names used for the input and output variables in a function are completely arbitrary. The workspace never needs to know the names of these variables. We could rewrite the above function code as:

```
function bar = timestwo(foo)
  bar = foo*2;
end
```

and it will work identically to the version at the top.

Here are some additional examples of functions that implement the quadratic formula.

To use each one individually, copy the text from "function" through the last "end" in each case, and paste into a separate m-file in the Matlab editor, saving it with a name that matches the function, i.e. quadform1.m, quadform2.m, quadform3.m etc.

quadform1.m :

```
function r = quadform1(a,b,c)
  % Re-implementation of quadform.m as a function.
  % Three coefficients of quadratic polynomial as inputs (a,b,c).
  % One output (r), which will be a vertical array of the two roots.

  x1 = (-b+sqrt(b^2-4*a*c))/(2*a);
  x2 = (-b-sqrt(b^2-4*a*c))/(2*a);

  r = [x1; x2];

end
```

quadform2.m :

```
function r = quadform2(p)
  % Another implementation, here designed to take only
  % one input (a 3-element array) to more closely match
  % the operation of the built-in "roots" function

  a = p(1);
  b = p(2);
  c = p(3);

  x1 = (-b+sqrt(b^2-4*a*c))/(2*a);
  x2 = (-b-sqrt(b^2-4*a*c))/(2*a);

  r = [x1; x2];

end
```

quadform3.m :

```
function [x1,x2] = quadform3(a,b,c)
  % Yet another implementation designed to output
  % the two roots separately, instead of together
  % in a single array as in the previous two examples
  %
  % This function has two separate outputs, similar to
  % the max/min commands we have used.  The first will be
  % x1, the second will be x2

  x1 = (-b+sqrt(b^2-4*a*c))/(2*a);
  x2 = (-b-sqrt(b^2-4*a*c))/(2*a);

end
```

Function handles:

At the start of this topic, we said that functions must be defined in a separate m-file. This is not quite true. Starting with Matlab v7, the concept of "function handles" is supports, allowing simple functions to be defined within any m-file.

Here is an example:

```
f = @(x,y) (x.^2 - y.^2);
```

The @(x,y) syntax indicates the local variables defined within the function to hold values passed to the function (two in the above case, but there could be none, or many). The expression in parentheses at the end of the line defines the value that will be returned by the function. So:

```
f(-1,9)          % returns (-1)^2 - (9)^2 = -80

f([1 2],[3 4])   % returns [1 2].^2 - [3 4].^2 = [1 4] - [9 16] = [-8 -12]
```