Functions and for Looping

Rong Qin
Department of Mathematics

Overview

We will create m-files or script files.

M-files

We use m-files or script files when creating a complicated program or function in MATLAB. These allow us to execute several lines of command at once. The m-file is also *reusable* - the file can be called upon in the command window and can act just like a built-in function. It also *simplifies* the code by breaking down groups of code into separate files, which make the code easier to read. To begin an m-file, you can type ctrl+N or go to file → new → script.

Creating Functions

• The general form of the `function` statement is:

function output = functionname (input)
...
end

For example,

```
function y = myfunction(x)

% Input: x
% Output: y = x^2 + x + 1.

y = x^2 + x + 1;

end
```

• In order to run this function from the command line or another m-file you must save it as ****.m, where **** should be the name of your function. For example, in the function above, it would be "myfunction.m".

As typed, the above function only works for a single input. How could we change this function to work for a vector input?

• *Note*: Always put comments throughout an m-file to say what the code does so that it is clear to you when you come back to use it, or to someone else who may use it.

• *Do not forget to assign output!* Nothing whill happens when no output is assigned. For example;

```
function y = myfunction(x)

x^2 + x + 1;

end
```

• *Functions do not save variables back to the main workspace unless they are output.*
• The general form of the function with multiple outputs/inputs is

function [output1, output2] = functionname (input1, input2, input3)
...
end

Anonymous Function

An anonymous function lets us define a function in the middle of a MATLAB script or in the command line. They generally look like:

```
functionName = @(inputs)(output)
```

For example,

```
f1 = @(x)(x^2 + x + 1);
f2 = @(x,y)(x^3 - y);
f1(3)
f2(10,2)
```

In-Class Exercise

The square of the sum of the first ten natural numbers is,

$$(1 + 2 + ... + 10)^2 = 55^2 = 3025.$$

The sum of the squares of the first ten natural numbers is,

$$1^2 + 2^2 + ... + 10^2 = 385.$$

Hence, the difference between the square of the sum of the first ten natural numbers and the sum of the squares is $3025 - 385 = 2640$.

1. Write a function that takes an input n and for the first n natural numbers returns: the square of the sum, the sum of the squares, and the difference between the square of the sum and the sum of the squares.
2. Find the difference between the square of the sum of the first one hundred natural numbers and the sum of the squares.

FOR LOOP

A `for` loop executes commands repeatedly. The general form of the `for` statement is:

for varname = startvalue : increment : endvalue
...
end

If no increment value is given, then the default is 1. For example,

```
a = 0;   % Initialize a.
for i = 1:1:5    % Variable i takes on values 1, 2, 3, 4, and then 5.
    a = a + i;   % What does this do?
end
```

The context of a `for` loop should always be indented between the `for` and `end` lines of the statement for readability.

In-Class Exercise

1. Create a function called `vectorsum` that takes as input a vector of arbitrary length and sums up its entries.

   (a) Test your function with $u = \begin{pmatrix} 2 & 8 & 3 & 4 & 9 & 10 \end{pmatrix}$

   (b) `x=linspace(0,1,300)`

   You can check to see if your program is running correctly by using the MATLAB command

   ```
   >> sum(u)
   ```

2. Write a function call **mydot** that calculate the dot product of two vectors with arbitrary dimension. The function should input two vectors, u and v (you can assume u and v have the same dimension), and output their dot product, d. Test your program with the vectors $u = (1, 2, 3, 4)$ and $v = (2, 4, 6, 8)$. Recall that the command "`dot(u,v)`" also calculates the dot product, so this is a good way to check that your program is working correctly.

3. Write a function that takes three inputs, a, b, c and returns the two roots of a quadratic equation $ax^2 + bx + c = 0$.

Note: i and j are predefined symbols for the square root of -1. However, MATLAB does not prevent us from using these as variables and reassigning them. If we aren't using complex numbers in our code (none of the assignments in class require complex numbers), this does not usually lead to trouble but occasionally can cause confusion. The most common case is that if our code uses i or j without initializing them, we do not get an error, as with other variables, i.e., a + 1, and i + 1.