Music 270a: Matlab Tutorial 1

Tamara Smyth, trsmyth@ucsd.edu
Department of Music,
University of California, San Diego (UCSD)

January 7, 2013

Getting Started

If using “Windows”, double click on the Matlab icon to start Matlab. Several windows will open but bring just the “Command Window” to the foreground for now. We will begin by typing commands at the >> prompt.

Getting Help

Thankfully, Matlab has extensive “Help” documentation that is built in and easy to access. Type “help” at the command prompt to see a list of all the help topics in Matlab. To read about any particular topic simply type “help < topic name >” at the command prompt. For example, if you want to get an idea of the audio support in Matlab, simply type

```
>> help audio
```

If you want any help on how to call a function, you simply type in “help” and the < function name > at the command prompt. For example

```
>> help sound
>> help cos
>> help plot
>> help help
```

If you want to complete a certain task and you’re wondering whether or not there’s a built in Matlab function that will do it for you, you may be able to find what you’re looking for using “lookfor”. For example, what Matlab function could you use to make a sine wave?

```
>> lookfor sine
```

Numerical Computation

Matlab can be used as a calculator. For example, inputing "3+3" at the command prompt, Matlab will yield the following:

```
>> 3+3

ans =

6
```

Try using operators for multiplication (" * "), division (" / ") and raising to an exponent (" ˆ "). You can type "help ops" for a list of all the operators in Matlab. Make sure you are aware of operator precedence. The following will yield different answers:

```
>> 8 * 7 / 4 + 3

ans =

17

>> 8 * 7 /(4 + 3)

ans =

8
```

Built-in Values

Some well-known constants are built in to Matlab.

```
>> pi
>> eps
>> i
```

Basic Functions

Be sure when using these function that you remember the parentheses. Try the following:

```
>> sqrt(25)
>> cos(pi)
>> sin(0)
>> cos(sin(pi))
```

Variables

Rather than using hard coded numbers, it makes sense to use variables. Variables are useful because they can be used in an expression instead of the value they hold, making the expression more meaningful for the programmer. They can be given any name you wish (as long as it isn't already in use by Matlab).

```
>> x = 3*2.0

ans =

6

>> y = x + 2

ans =

8
```

Try the following:

```
>> x <return>
>> who <return>
>> whos <return>
>> clear all <return>
>> whos <return>
```

Vectors and Matrices

A vector is a sequence of numbers. Vectors will be particularly important to us because that is how sound is represented, as a sequence of numbers. Try the following to see how the comma and semicolon change the output:

```
>> x = [1 3 7 15]
>> x = [1, 3, 7, 15]
>> x = [1; 3; 7; 15]
>> x'
```

In the last example, we see the transpose operator. We will see this more later, but for now you can use this to turn row vectors into column vectors and vice versa.

Some other useful ways to create vectors:

```
>> 1:5
>> z = 1:1/2:5
>> ones(1, 5)
>> zeros(1, 5)
```

You can determine the length and size of a vector using the following commands:

```
>>length(x)
>>size(x)
```

Accessing values within a vector:

```
>> x(3)
>> x(1:3)
>> x(find(x==7))
```

You can concatenate two (or more) vectors as follows:

```
>> z = [1:3, 4:6]
```

Vector Operations

Operations applied to variables can also be applied to vectors however some care must be taken. Create the following two vectors:

```
>> a = [5 6 7]
>> b = [1 2 3]
```

Try adding the vectors.

Now try multiplying the two vectors. You will get an error since Matlab is trying to do Matrix multiplication and is expecting the number of rows in **a** to equal the number of columns in **b**. To perform multiplication element by element, you must use a dot with the multiplication operator, ".*". This is also true for the division and exponent operator.

Scripting and creating M-files

It won't be long before your programs are too long and you get tired of entering everything on the command line. Fire up your favourite text editor (or use the built-in Matlab editor) and write a sequence of Matlab commands (remember to use ";" at the end of each line to suppress the output). Save the file with whatever name you wish < mfilename >, but make sure to include a ".m" extension, that is, < mfilename.m >. You can call your script from the command prompt with the name of the file (without the .m extension).

```
>> mfilename
```