Lecture 1

Vectors, Functions, and Plots in MATLAB

In these notes

≫
≫

will indicate commands to be entered at the MATLAB prompt ≫ in the command window. You do not type the symbol ≫.

Entering vectors

In MATLAB, the basic objects are matrices, i.e. arrays of numbers. Vectors can be thought of as special matrices. A row vector is recorded as a $1 \times n$ matrix and a column vector is recorded as a $m \times 1$ matrix. To enter a row vector in Matlab, type the following in the command window:

```
≫ v = [0 1 2 3]
```

and press enter. MATLAB will print out the row vector. To enter a column vector type

```
≫ u = [9; 10; 11; 12; 13]
```

You can access an entry in a vector with

```
≫ u(2)
```

and change the value of that entry with

```
≫ u(2)=47
```

You can extract a *slice* out of a vector with

```
≫ u(2:4)
```

You can change a row vector into a column vector, and vice versa, easily in Matlab using

```
≫ w = v'
```

(This is called *transposing* the vector and we call ' the transpose operator.) There are also useful shortcuts to make vectors such as

```
≫ x = -1:.1:1
≫ y = linspace(0,1,11)
```

Basic Formatting

To make MATLAB put fewer blank lines in its output, enter

```
≫ format compact
≫ pi
≫ x
```

To make MATLAB display more digits, enter

```
≫ format long
≫ pi
```

Note that this does not change the number of digits MATLAB is using in its calculations; it only changes what is displayed.

Plotting Data

Consider the data in Table 1.1.[1] We can enter this data into MATLAB with the following commands entered

T (C°)	5	20	30	50	55
μ	0.08	0.015	0.009	0.006	0.0055

Table 1.1: Viscosity of a liquid as a function of temperature.

in the command window:

```
≫ x = [ 5   20   30   50   55 ]
≫ y = [ 0.08   0.015   0.009   0.006   0.0055]
```

Entering the name of the variable retrieves its current values. For instance

```
≫ x
≫ y
```

We can plot data in the form of vectors using the plot command:

```
≫ plot(x,y)
```

This will produce a graph with the data points connected by lines. If you would prefer that the data points be represented by symbols you can do so. For instance

```
≫ plot(x,y,'*')
≫ plot(x,y,'o')
≫ plot(x,y,'.')
```

[1] Adapted from Ayyup &McCuen 1996, p.174.

Data as a Representation of a Function

A major theme in this course is that often we are interested in a certain function $y = f(x)$, but the only information we have about this function is a discrete set of data $\{(x_i, y_i)\}$. Plotting the data, as we did above, can be thought of envisioning the function using just the data. We will find later that we can also do other things with the function, like differentiating and integrating, just using the available data. Numerical methods, the topic of this course, means doing mathematics by computer. Since a computer can only store a finite amount of information, we will almost always be working with a finite, discrete set of values of the function (data), rather than a formula for the function.

Built-in Functions

If we wish to deal with formulas for functions, MATLAB contains a number of built-in functions, including all the usual functions, such as `sin( )`, `exp( )`, etc.. The meaning of most of these is clear. The dependent variable (input) always goes in parentheses in MATLAB. For instance

```
≫ sin(pi)
```

should return the value of $\sin \pi$, which is of course 0 and

```
≫ exp(0)
```

will return e^0, which is 1. More importantly, the built-in functions can operate not only on single numbers but on vectors. For example

```
≫ x = linspace(0,2*pi,41)
≫ y = sin(x)
≫ plot(x,y)
```

will return a plot of $\sin x$ on the interval $[0, 2\pi]$

Some of the built-in functions in MATLAB include: `cos( )`, `tan( )`, `sinh( )`, `cosh( )`, `log( )` (natural logarithm), `log10( )` (log base 10), `asin( )` (inverse sine), `acos( )`, `atan( )`. To find out more about a function, use the `help` command; try

```
≫ help plot
```

User-Defined Anonymous Functions

If we wish to deal with a function that is a combination of the built-in functions, MATLAB has a couple of ways for the user to define functions. One that we will use a lot is the anonymous function, which is a way to define a function in the command window. The following is a typical anonymous function:

```
≫ f = @(x) 2*x.^2 - 3*x + 1
```

This produces the function $f(x) = 2x^2 - 3x + 1$. To obtain a single value of this function enter

```
≫ y = f(2.23572)
```

Just as for built-in functions, the function f as we defined it can operate not only on single numbers but on vectors. Try the following:

```
≫ x = -2:.2:2
≫ y = f(x)
```

This is an example of *vectorization*, i.e. putting several numbers into a vector and treating the vector all at once, rather than one component at a time, and is one of the strengths of MATLAB. The reason $f(x)$ works when x is a vector is because we represented x^2 by `x.^2`. The `.` turns the exponent operator `^` into entry-wise exponentiation, so that `[-2 -1.8 -1.6].^2` means $[(-2)^2, (-1.8)^2, (-1.6)^2]$ and yields `[4 3.24 2.56]`. In contrast, `[-2 -1.8 -1.6]^2` means the matrix product $[-2, -1.8, -1.6][-2, -1.8, -1.6]$ and yields only an error. The `.` is needed in `.^`, `.*`, and `./`. It is not needed when you `*` or `/` by a scalar or for `+`.

The results can be plotted using the `plot` command, just as for data:

```
≫ plot(x,y)
```

Notice that before plotting the function, we in effect converted it into data. Plotting on any machine always requires this step.

Exercises

1.1 Recall that `.*`, `./`, `.^` are component-wise operations. Make row vectors $\mathbf{a} = 0 : 1 : 3$ and $\mathbf{b} = [-1 \;\; 0 \;\; 1 \;\; 2]$. Try the following commands and report the answer (or error) they produce. Are any of the results surprising?

(a) `a.*b`, (b) `a*b`, (c) `a*b'`, (d) `a+3*b`,

(e) `a./b`, (f) `2*b./a`, (g) `a.^3`, (h) `a.^b`,

1.2 Find a table of data in an engineering or science textbook or website. Input it as vectors and plot it, *using symbols at the data points.*

Use the insert icon to label the axes and add a title to your graph. Turn in the graph. Indicate what the data is and properly reference where it came from.

1.3 Find a *non-linear* function formula in an engineering or science textbook or website. Make an anonymous function that produces that function. Plot it with a *smooth curve* on a physically relevant domain.

Label the axes and add a title to your graph. Turn in the graph and include the Matlab command for the anonymous function. Indicate what the function means and properly reference where it came from.