LAB 1.1
Vectors, Functions, and Plots In MATLAB

In this book > will indicate commands to be entered in the command window. You do not actually type the command prompt > .

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 an $m \times 1$ matrix. To enter a row vector in MATLAB, type the following at the prompt (>) 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
```

and

```
> y = linspace(0,1,11)
```

Basic Formatting

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

```
> format compact
```

To make Matlab display more digits, enter

```
> format long
```

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

Plotting Data

Consider the data in Table 1.1. We can enter this data into Matlab with the following commands entered in the command window:

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.

```
> 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,'.')
```

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,40)
> 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 Inline 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 inline function, which is a way to define a function in the command window. The following is a typical inline function:

```
> f = inline('2*x.^2 - 3*x + 1','x')
```

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

```
> 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

1.1.1 Find a table of data in an engineering textbook or engineering website. Input it as vectors and plot it. 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 where it came from.

1.2.1 Make an inline function g(x) = x + cos(x^5). Plot it using vectors x = -5:.1:5; and y = g(x);. What is wrong with this graph? Find a way to make it look more like the graph of g(x) should. Turn in both plots.

LAB 1.2
MATLAB Programs

In Matlab, programs may be written and saved in files with a suffix .m called M-files. There are two types of M-file programs: functions and scripts.

Function Programs

Begin by clicking on the new document icon in the top left of the Matlab window (it looks like an empty sheet of paper).

In the document window type the following:

```
function y = myfunc(x)
   y = 2*x.^2 - 3*x + 1;
```

Save this file as: myfunc.m in your working directory. This file can now be used in the command window just like any predefined Matlab function; in the command window enter:

```
> x = -2:.1:2; ..... Produces a vector of x values.
> y = myfunc(x); ..... Produces a vector of y values.
> plot(x,y)
```

Note that the fact we used x and y in both the function program and in the command window was just a coincidence. In fact, it is the name of the file myfunc.m that actually mattered, not what anything in it was called. We could just as well have made the function

```
function nonsense = yourfunc(inputvector)
nonsense = 2*inputvector.^2 - 3*inputvector + 1;
```

Look back at the program. All function programs are like this one, the essential elements are:

- Begin with the word function.
- There is an input and an output.
- The output, name of the function and the input must appear in the first line.
- The body of the program must assign a value to the output.

Functions can have multiple inputs, which are separated by commas. For example:

```
function y = myfunc2d(x,p)
y = 2*x.^p - 3*x + 1;
```

Functions can have multiple outputs, which are collected into a vector. Open a new document and type:

```
function [x2 x3 x4] = mypowers(x)
x2 = x.^2;
x3 = x.^3;
x4 = x.^4;
```

Save this file as mypowers.m. In the command window, we can use the results of the program to make graphs:

```
> x = -1:.1:1
> [x2 x3 x4] = mypowers(x);
> plot(x,x,'black',x,x2,'blue',x,x3,'green',x,x4,'red')
```

Script Programs

Matlab uses a second type of program that differs from a function program in several ways, namely:

- There are no inputs and outputs.
- A script program may use and change variables in the current workspace (the variables used by the command window.)

Below is a script program that accomplishes the same thing as the function program plus the commands in the previous section:

```
x2 = x.^2;
x3 = x.^3;
x4 = x.^4;
plot(x,x,'black',x,x2,'blue',x,x3,'green',x,x4,'red')
```

Type this program into a new document and save it as mygraphs.m. In the command window enter:

```
> x = -1:.1:1;
> mygraphs
```

Note that the program used the variable x in its calculations, even though x was defined in the command window, not in the program.

Many people use script programs for routine calculations that would require typing more than one command in the command window. They do this because correcting mistakes is easier in a program than in the command window.

Program Comments

For programs that have more than a couple of lines it is important to include comments. Comments allow other people to know what your program does and they also remind yourself what your program does if you set it aside and come back to it later. It is best to include comments not only at the top of a program, but also with each section. In Matlab anything that comes in a line after a % is a comment.

For a function program, the comments should at least give the purpose, inputs, and outputs. A properly commented version of the function with which we started this section is:

```
function y = myfunc(x)
% Computes the function 2x^2 -3x +1
% Input: x -- a number or vector;
%            for a vector the computation is elementwise
% Output: y -- a number or vector of the same size as x
y = 2*x.^2 - 3*x + 1;
```

For a script program it is often helpful to include the name of the program at the beginning. For example:

```
% mygraphs
% plots the graphs of x, x^2, x^3, and x^4
% on the interval [-1,1]

% fix the domain and evaluation points
x = -1:.1:1;

% calculate powers
% x1 is just x
x2 = x.^2;
x3 = x.^3;
x4 = x.^4;

% plot each of the graphs
plot(x,x,'black',x,x2,'blue',x,x3,'green',x,x4,'red')
```

The Matlab command help prints the first block of comments from a file. If we save the above as mygraphs.m and then do
> help mygraphs
it will print into the command window:
mygraphs
plots the graphs of x, x^2, x^3, and x^4
on the interval [-1,1]

Exercises

1.2.1 Write a function program for the function $x^2 e^{-x}$, using entry-wise operations (such as .* and .^). To get e^x use exp(x). Include adequate comments in the program. Plot the function on $[-5, 5]$. Show the program and the graph to the Lab Instructor.

1.2.2 Write a script program that graphs the functions sin x, sin 2x, sin 3x, sin 4x, sin 5x and sin 6x on the interval $[0, 2\pi]$ on one plot. (π is pi in Matlab.) Include comments in the program. Show the program and the graph to the Lab Instructor.

LAB 1.3

Discretization Error

Introduction

Below an example is given. A derivative of a function is evaluated by using the values of the function only. To accomplish this Taylor series expansion is required. Since Taylor series is infinite some terms are left out which introduces an error.

Example Consider the problem of approximating the derivative $f'(x_0)$ of a given smooth function $f(x)$ at the point $x = x_0$. For instance, let $f(x) = \sin(x)$ be defined on the real line $-\infty < x < \infty$, and set $x_0 = 1.2$. Thus, $f(x_0) = \sin(1.2) \approx 0.932 \ldots$.

Further, consider a situation where $f(x)$ may be evaluated at any point x near x_0, but $f'(x_0)$ may not bedirectly available or is computationally expensive to evaluate. Thus, we seek ways to approximate $f'(x_0)$ by evaluating f at x near x_0.
A simple algorithm may be constructed using Taylor's series. This fundamental theorem is given on the preceding page. For some small, positive value h that we will choose in a moment, write

$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \frac{h^3}{6}f'''(x_0) + \frac{h^4}{24}f''''(x_0) + \cdots .$$

Then

$$f'(x_0) = \frac{f(x_0+h) - f(x_0)}{h} - \left(\frac{h}{2}f''(x_0) + \frac{h^2}{6}f'''(x_0) + \frac{h^3}{24}f''''(x_0) + \cdots\right).$$

Our *algorithm* for approximating $f'(x_0)$ is to calculate

$$\frac{f(x_0+h) - f(x_0)}{h}.$$

The obtained approximation has the *discretization error*

$$\left|f'(x_0) - \frac{f(x_0+h) - f(x_0)}{h}\right| = \left|\frac{h}{2}f''(x_0) + \frac{h^2}{6}f'''(x_0) + \frac{h^3}{24}f''''(x_0) + \cdots\right|.$$

If we know $f''(x_0)$, and it is nonzero, then for h small we can estimate the discretization error by

$$\left|f'(x_0) - \frac{f(x_0+h) - f(x_0)}{h}\right| \approx \frac{h}{2}\left|f''(x_0)\right|.$$

The MATLAB Script below gives the plot for log-log of the absolute error versus h.

```
x0 = 1.2;
f0 = sin(x0);
fp = cos(x0);
i = -20:0.5:0;
h = 10.^i;
err = abs (fp - (sin(x0+h) - f0)./h );
d_err = f0/2*h;
loglog (h,err,'-*');
hold on
loglog (h,d_err,'r-.');
xlabel('h')
ylabel('Absolute error')
```

Exercises

1.3.1 Approximate the derivative of the function $f(x)=e^{-2x}$ evaluated at x_0 =0.5(refer to the above example and observe the difference and similarities).

1.3.2 Carry out derivation and calculation analogous to the above example by using the expression

$$\frac{f(x_0+h) - f(x_0-h)}{2h}$$

For approximating the firs derivative $f'(x_0)$.