

## MATH 313 Spring 2009, Homework , Computer-based problems

In this course, we will be using MATLAB for computer-based matrix computations. MATLAB is a programming language/environment that provides easy access to a number of standard libraries: LAPACK, EISPACK, BLAS. These libraries are written in Fortran, and there exist a number of standard interfaces to other languages as well. However, one usually spends a bit of effort interacting with the libraries, which obscures the underlying linear algebra. MATLAB permits us to avoid the formalities and instead focus on the mathematics.

There are a number of ways to run MATLAB, and you can choose whichever suits you best: as a student, you can buy a student license relatively cheaply; you can use it on the SAS computer labs; or you can use the freely available program GNU Octave which is essentially identical. (If you need help with getting Octave installed, I can help. In the places where they differ, Octave is a slightly better language. When I refer to “Matlab” I mean either MATLAB or Octave. I’ll assume that you’ve never used Matlab, but we’ll quickly ramp up your Matlab skills.

The first command you should be aware of is

```
>> help
```

which is useful for finding out about commands and their arguments. You should be aware that nearly everything in Matlab is self-documenting, so you can figure out much of the language yourself easily.

Everything in Matlab is an vector or a matrix (I’m slightly cheating here. There are more general objects as well, but we won’t encounter them in this course.) Vectors can be named almost anything except certain reserved words. To make a vector come into existence, simply give it a value:

```
>> x=10
```

creates a variable  $x$  and assigns the value 10 to it. You can also make vectors pretty easily:

```
>> y=1:.3:25
```

makes a row vector whose entries (from left to right) start at 1, increment by 0.3 and are no larger than 25. Here’s how to specify elements of a matrix:

```
>> a=[1 2 3; 4 5 6; 7 8 9; 10 11 12]
```

This is a 4x3 matrix with those specified values.

**Problem 1** How might I enter the vector  $y$  manually? What if I wanted it to be a column vector instead?

Matlab is a useful calculator. Most of the usual operations you might want are available:

```
>> cos(((sqrt(3)+1)/7)^3)
```

What's nice about Matlab is that many of these also work for matrices. You can avoid writing loops in Matlab by working with entire arrays of numbers. For instance, here's how one might compute the squares of the first 10 odd numbers:

```
>> (2.*(1:10)-1).^2
```

Notice two things with the above expression: namely the array multiplication is written (`.*`) and the array exponentiation is written (`.^`). This is to avoid confusion with matrix multiplication (`*`) and powers (`^`). Many other functions are also “vectorized” which means that they work on elements of a matrix or vector, such as sine or cosine.

**Problem 2** Without writing any Matlab loops, show how one might compute the first eight triangular numbers  $\frac{n(n+1)}{2}$ .

One of the big strengths of Matlab is easy access to visualization. It's pretty easy to plot things in 2 or 3 dimensions... For instance,

```
>> x=0:0.1:2*pi;
>> figure
>> plot(x,sin(x))
>> title('A sine wave');
>> xlabel('Angle (radians)');
>> ylabel('Value of sine');
```

should give you a sine wave with some axis labels. The “plot” command is a parametric plotting function, so you can also do something like

```
>> t=0:.1:10;
>> x=sin(t.^2);
>> y=cos(t);
>> figure
>> plot(x,y,'r');
```

And 3d plotting is easy too:

```
>> t=0:.1:6*pi;
>> x=sin(t);
>> y=cos(t);
>> z=t;
>> plot3(x,y,z,'+');
```

should plot points on a helix.

**Problem 3** Suppose a rock of mass 1 kg is thrown at an angle of  $38^\circ$  with the horizon and an initial speed of 2 m/s. By hand, write a differential equation describing the motion of this rock. Solve the differential equation, and plot its path in Matlab.

If you've done a bit of work in Matlab, sometimes you want to save your commands so that you don't have to type them all in again. The way to do this is to list the commands you want executed one after another in a plain text file

with extension “.m”, called a “script”. You can (and should) insert comments with a percent sign to clarify what you’ve written. MATLAB has a convenient editor for writing scripts. You can start it by typing

```
>> edit
```

and then cutting and pasting commands into the window. You can later execute the commands in a script by typing its filename (without the “.m”) at the Matlab prompt.

**Problem 4** Copy all of the commands you’ve used thus far (for the other problems) into a script file. Add some comments to your script, save it, and print it out (along with any figures you’ve produced) and hand it in to me with your other homework problems.

**In the future assignments as well, please put all of the Matlab commands you used into a script, print it out, and turn it in with the problems from the book. Don’t forget to print out any figures you have made as well!**