MATLAB Tutorial
Applied Mathematics I, Fall 1999

The following excercises have been checked in the MATLAB versions 5 for the PC only. Students wishing to purchase the softward for a personal computer should check out the new Stdent Version being marketed by the Mathworks, Inc.

From the command line type "matlab" to load, wait for the prompt ">>".

Matlab vectorizes operations where possible. Commands are usually followed by a semicolon, which suppresses screen printouts. Try the following (ret stands for hitting the return key):

```
A=[1,2,3,4,5] ret
A=[1;2;3;4;5]; ret
A
A=[1 2 3 4 5] ret
A=[1
2
3
4
5] ret
```

This shows various ways row and column vectors can be entered.
A=[1 2 3 4;2 3 4 1; 3 4 1 2; 4 1 2 3] ; ret
B=[1 -1 0 1]; ret
C=[1;2;1;1];
Let's try matrix multiplication. Try
F=A*C ret
This works, since C is a column vector.
A*B ret
gives an error, as does C*A, but
G=B*A ret
works.

Adjoint:
E=C' ret
i.e. E=B.

Solve the linear equation Ax=F:
x=A\F ret
That is A\F is the solution to Ax=F, hence x = C.
The same result can be obtained using the inverse matrix:
x=inv(A)*B ret
Solve the equation xA=G:
x=G/A ret

Array operations:

The ".*" multiplication multiplies array componentwise:
x=[1 2 3]; ret
y=[4 5 6];
z=x.*y ret
The result, [4 10 18], is multiplication by component.
Similarly for division:

z=x./y ret
Try
z=x.\y ret
z=x.^2 ret
Note that componentwise exponentiation is done with ".^".

Array operations apply to matrices as well:
C=[1 2 3
4 5 6] ret
D=C.^2 ret

Addition and subtraction:
A=[1 0;2 3]; ret
B=[1 -1; 4 5];ret
C=A + B ret
That is, the ordinary "+" symbol is used.

Function values can also be conveniently stored as vectors:
x=-2:.2:1; ret
This creates a row vector consisting of [-2 -1.8 -1.6 ….. .8 1]. To verify this
x ret
y=exp(x);
This is a row vector with the values [.1353 ….. 2.718].

Plotting:
help plot ret
will get you some information. A simple example is
x = -2:.01:2; ret
y=1./(1.+x.^2); ret
plot(x,y) ret
To add some info:
xlabel('values of x') ret
ylabel('values of y') ret
title('My favorite function') ret

M-Files:

MATLAB makes use of M-files and FUNCTION files to store lists of commands and routines for creating functions.Think of M-files as simply lists of commands as would
be entered sequentially at the command line, stored as a file "name.m' where "name" is then executable.
Typing
name ret
executes all commands in the m-file. To create an m-file that will plot the
previous function, we could simply list the commands used above in a file "favfunc.m" say, then execute. To use a slightly different way of creating the function, let favfunc.m be given by

for i=1:401
x(i) = -2 + (i-1)(.01);*
y(i)= 1/(1.+x(i)^2);
end
plot(x,y)

Now typing
favfunc ret
will give us the plot.

We could also have created two files a function file called say ffunc.m, and a M-file favfunc.m:
ffunc.m:

```
function y=ffunc(x)
y=1 ./(1.+x.^2);
```

favfunc.m:

```
x=-2:.01:2;
y=ffunc(x);
plot(x,y)
```

Typing favfunc ret at the command line will produce the plot.

ODEs:

Consider thelinear forst-order differential equation

x dy/dx+2y = exp(-x). (*)

The general solution (check this) is

y=[(-1-x)exp(-x)+C]/x^2.

Write this as x^2y+(x+1)exp(-x) = z(x,y) = constant. The following commands will create a contour map of the integral curves in the interval .1 < x < 2:

```
x=.1:.1:2;
y=x;
[X,Y]= meshgrid(x,y);
Z=X.^2.*Y+(X+1).*exp(-X);
contour(X,Y,Z,20)
```

This will produce 20 integral curves. Not that the meshgrid command takes to axis partitions and creates a 2D grid. dThe MATLAB package contains several routines for solving the IVP for ODEs. We shall be making use of explicit codes in order to see how they work. However we shall look at our example to see how MATLAB does it. Suppose we want to solve (*) with y(1)= 1. Create the M-file “ourode.m:

```
function ydot=ourode(x,y)
ydot=-(2.*y)./x+exp(-x)./x;
```

At the command line (or in another M-file), type

```
x0=1;
xf=2.;
y0=1.;
xspan=[x0 xf];
[x,y]=ode23(‘ourode’,xspan,y0);
plot(x,y)
```