MATH.1020 First-Year Seminar in Mathematics
Some Useful MATLAB Commands

Please note that in MATLAB, everything that follows the % symbol is a comment. You do not have to type the comments in the sample commands below.

A. 2D Graphing

The basic MATLAB graphing command is the *plot* command. You follow a 3-step procedure:

1. generate some x values;
2. calculate the corresponding y values; and
3. plot the points.

Try these 3 MATLAB commands:

```
x = linspace(0, 2*pi, 100); %Creates a vector of 100 numbers evenly spaced between 0 and 2π.

y = sin(x); %Creates a vector with the values of the sine function at each of the entries
            %in the x vector

plot(x, y)  %This plots the (x, y) points connected with straight line segments
```

Note that the semicolon at the end of a command suppresses the output, so you don't see all the calculated values.

To plot more than one graph on the same set of axes, you can try something like this:

```
z = cos(x);

plot(x, y, '-', x, z, ':') %Plots the (x, y) points connected with a solid line ('-') and the
                           %(x, z) points connected with a dotted line (':')
```

You can add a title and axis labels and you can label your curves using the following commands. DO NOT close the figure window containing the plot.

```
title( 'Graphs of sine and cosine functions' )  %This creates a title at the top of the graph.

xlabel('x')    %Puts a label under the horizontal axis

ylabel('y')    %Puts a label next to the vertical axis

legend('y = sin(x)', 'y = cos(x)')  %Creates a legend indicating which graph is which
```

If you don't like where MATLAB places the legend box, you can use the mouse to drag the box wherever you want it.

In addition to solid lines and dotted lines, you can generate dashdot graphs (`'.-'`) or dashed graphs (`'--'`). **You can control the color of the graph** by adding a letter after the line type (**b**lue, **g**reen, **r**ed, **c**yan, **m**agenta, **y**ellow, or blac**k**). For example, to plot the sine graph in red using a dashdot line and the cosine graph in black using a dashed line, type

```
plot(x, y, '.-r', x, z, '--k')
```

OVER

B. 3D Graphing

Here is an example of how to graph the equation $z = x^2 - y^2$ over the square $-2 \le x \le 2,\ -2 \le y \le 2$.

```
x=linspace(-2,2,100);
y=x;
[X,Y]=meshgrid(x,y);
Z=X.^2-Y.^2;
mesh(x,y,Z)
```

If you click on the icon to the right of the hand icon, you can rotate the graph in three dimensions.

C. Matrix Algebra and Systems of Linear Equations

MATLAB is very good at calculations involving matrices; the MAT in MATLAB stands for matrix. Here are some examples involving systems of linear equations.

1. Here is one way to solve the system

$$\begin{cases} x + 2y &= 0 \\ 3x + 4y &= 2 \end{cases}$$

```
A=[1,2;3,4] % This generates the matrix of coefficients of the unknowns x and y
b=[0;2] %This generates the vector containing the right-hand sides of the equations
A\b %This solves the system Ax=b
```

Notice that rows are separated by semicolons in the definitions of A and b.

2. MATLAB can solve overdetermined systems (systems with more equations than unknowns). Try solving the overdetermined system

$$\begin{cases} x + 2y &= 0 \\ 3x + 4y &= 2 \\ 5x + 6y &= 3 \end{cases}$$

using these commands:

```
A=[1,2;3,4;5,6]
b=[0;2;3]
A\b
```

MATLAB is calculating the "least-squares" solution: the pair (x, y) for which the residual $[x + 2y - 0]^2 + [3x + 4y - 2]^2 + [5x - 6y - 3]^2$ is a minimum.

3. MATLAB can also generate a solution to an underdetermined system (systems with more unknowns than equations). Try solving the underdetermined system

$$\begin{cases} x + 2y + 3z &= 0 \\ 4x + 5y + 6z &= 6 \end{cases}$$

using these commands:

```
A=[1,2,3;4,5,6]
b=[0;6]
A\b
```

D. Solving Nonlinear Equations

The MATLAB command *fzero* generates an approximate solution to an equation $f(x) = 0$. For example, to find an approximate solution of the equation $\cos(x) - x = 0$ near $x = 1$, try the command

```
fzero('cos(x)-x',1)
```

E. Interpolation

The MATLAB command *polyfit* generates a polynomial that best fits (in the sense of least squares) a given set of data points. For example, to find the linear function that best fits the data points $(0, 1), (1, 2), (2, 3), (3, 3), (4, 6), (5, 8)$, try the commands

```
x = [0 1 2 3 4 5];
y = [1 2 3 3 6 8];
p = polyfit(x,y,1) % The 1 tells MATLAB you want a polynomial of degree 1
% The first element of p is the coefficient of x, and the second is the constant term
```

To see a graph of the data points, marked with a + sign, and the line that best fits the data, try the commands

```
xp = linspace(-1,6,20);
yp = polyval(p,xp); % This command evaluates the polynomial p at the points in xp
plot(x,y,'+',xp,yp)
```

To find the quadratic that best fits these data points and plot the points and quadratic, try

```
p2 = polyfit(x,y,2)
yp2 = polyval(p2,xp)
plot(x,y,'+',xp,yp2)
```