Objectives

This session is a tutorial designed to a very quick overview of some of the numerical skills that you'll need to get started. Throughout the tutorial, the instructors will roam the class and help you work through the tasks.

This session will familiarize you with the following concepts.

1. MATLAB as a calculator
2. basic matrix manipulation
3. data storage: arrays, indexing
4. matrix multiplication vs element-by-element multiplication
5. basic plotting of functions
6. numbers vs characters and strings
7. overlaying plots
8. use some built-in functions; take a mean, a std-dev

1 Running, and Interface Layout

Double-click the MATLAB icon to start the program. It takes a while to load...

2 Toolboxes, Documentation and Tutorials

You can obtain information on some command by typing `doc plot`, for example. This will bring up a separate window with information and (usually) examples.

3 Basic Calculations

MATLAB is first and foremost a calculator. Built-in functions like sine, cosine and logarithms are all accessible by typing the name of the function and passing an argument enclosed in parentheses. Try typing `sin(2*pi)`. π is a pre-defined constant.

Perform the following operations in the command windows:

1. $(1.27 \times 10^{-7}) \times \pi =?$
2. $\cot(\pi/7) =?$
3. $e^{-3} =?$
4. $\ln(2.2) =?$
5. $\log(10^{3.2}) =?$
6. $\tan^{-1}(\pi/17) =?$

4 Matrix Algebra

MATLAB is short for "Matrix Laboratory". We will now explore some of its matrix manipulating features. Input the two matrices

$$A = \begin{pmatrix} 1 & 2 \\ -1 & 2 \end{pmatrix} \qquad b = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

using the following commands.

```
>> A = [ 1 2; -1 2]

A =

     1     2
    -1     2

>> b = [2; 1]

b =

     2
     1
```

The variables `A` and `b` now stand for these two matrices since we have "assigned values" to them using the equals sign, which you should interpret as the assignment operator. This means expressions like `x=x+1` are legitimate, even if it looks strange as a math expression. Until we overwrite or clear these variables, `A` and `b` now stand for these matrices. The first matrix A is a 2×2 matrix, where the first number denotes the number of rows and the second the number of columns. The second matrix b is then a 2×1 column matrix. The use of the semicolon in the assignment statement forces the start of a new row.

Simple algebraic operations: use * to denote multiplication, and $\wedge$ to denote exponentiation. You can obtain the transpose of a matrix by designating 'prime'. Try typing `b` and then `b'`.

Now perform the following operations. Some of the expressions are invalid mathematical operations. If this is the case, denote this.

1. Ab and bA
2. AA and A^2
3. bb and bb' and $b'b$
5. $|A|$, the determinate of A
6. $|b|$, the determinate of b

5 Array Indexing

Each element of an array is designated by integers that "point" to locations in the array. In MATLAB indices start with 1, not 0. Two-dimensional arrays are addressed as (row,column).

For the following section, input the following arrays. You'll need to use the format shown above.

$$c = (1, 2, 3, 4, 5) \qquad d = \begin{pmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \end{pmatrix} \tag{5.1}$$

Execute the following commands.

1 Single entries.
`c(1)`
`c(3)`
`c(end)`
2 A sub-array of values
`c(1:end)` and `c(:)`
`c(3:end)`
`c(2:4)`

```
d(1,1)
d(2,end)
d(4, 2)
d(7)
d(1:end, :)
d(2:3, 2:4)
d(2:4, 2:4)
```

Question:
What does the `:` operator do?

6 Element-by-element Operations

Clear the variables using the command:

```
>> clear A b c d
```

Note, you can clear all of the variables at once using `clear all`. Be careful though! You could lose all of your work.

Define the following variables.

$$a = (1, 2, 3, 4) \qquad b = (2, 2, 2, 2) \qquad c = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

Execute the following commands.

1 Compare
`a*a` with `a.*a` and `a.^2`
`a*b` `a.*b`
`c*c` `c.*c`
`2*a` with `2.*a`

Question: What is the difference between the `*` and `.*` operators?

2 built-in functions
`tan(a)`, `tan(c)`
`exp(a)`, `exp(c)`
`log(a)`, `log(c)`
`sqrt(a)`, `sqrt(c)`

Question: What is common in output of all of the previous commands?

Execute the following commands.

1 Dimensions and size
`size(a)`, `size(b)`, `size(c)`
`length(a)`, `length(b)`, `length(c)`

2 Maximum and Minimum
`max(a)`, `max(b)`, `max(c)`
`min(a)`, `max(b)`, `min(c)`

3 statistics
`mean(a)`, `mean(c)`
`mean(c, 1)`, `mean(c,2)`
(there are commands for `mode` and standard deviation `std` as well)

Question: Explain what the commands are doing.

7 Plotting

7.1 Plotting Data

The following data represents the time t it takes to fall a given distance d. Plot the following data and label the axis. First, cut and paste the data into the screen.

```
>> t = [0 0.100 0.200 0.300 0.400 0.500 0.600 0.700 0.800 0.900 1.000]
>> d = [ 0 -0.014 0.189 0.406 0.854 1.055 1.758 2.221 3.110 3.882 5.045]
```

Another way to generate an array of equally spaced elements is by using the colon operator. We could also have done

```
>> t = 0.00:0.1:1.00;
```

where the semicolon suppresses output to the screen. This creates a row array, using the `startvalue:delta:endvalue` syntax.

To make the plot, the command is easy:

```
>> plot(t,d)
```

Notice MATLAB automatically connects the dots. If we don't want to do that, try these commands, including the ones to add labels:

```
plot(t,y,'bo ')
title('Distance vs time', 'fontsize',18);
xlabel('t', 'fontsize',18);
ylabel('d', 'fontsize',18);
```

In the `plot` command, the `b` stands for blue, the `o` says small circles instead of dots, and the blank space tells MATLAB not to connect the dots.

By typing `doc plot`, a very detailed help page will come up explaining all of this in more detail.

Using the basic fitting tools found under `tools` → `basic fitting`, fit the data with

1. a linear fit
2. a quadratic fit

For each, investigate the residual plot using the `Plot residuals` feature in the basic fitting window.

Question:
Which fit curve do you think does a better job and why do you think this?

7.2 Plotting a mathematical functions with overlays

Clear the figure using `clf` or start a new figure using the command `figure`. To plot a mathematical function, you need two arrays to represent the domain (x-axis values) and the range (y-axis values). If we want to plot $f(x) = \sin(x)$ over one single period, you might try the following commands.

```
>> x = 0:0.1:2*pi
>> y = sin(x)
>> plot(x,y)
>> title('f(x) = sin(x)', 'fontsize',18);
>> xlabel('x', 'fontsize',18);
>> ylabel('y', 'fontsize',18);
```

Now, overlay a plot of $g(x) = \cos(x)$. You'll need the `hold on` command to keep MATLAB from clearing the figure. You can also add a legend using the command:

```
>> legend('sin(x)' , 'cos(x)')
```

The following commands do the trick.

```
>> x = 0:0.1:2*pi;
>> y1 = sin(x);
>> y2 = cos(x);
```

```
>> plot(x,y1, 'r')
>> hold on
>> plot(x,y2, 'b')
>> title('f(x) and g(x)', 'fontsize',18);
>> xlabel('x', 'fontsize',18);
>> ylabel('y', 'fontsize',18);

>> xlim([0,2*pi]);
>> ylim([-1.2, 1.2]);

>> legend('f(x) = sin(x)', 'g(x) = cos(x)')
```

There are also commands that change the domain and range of the plot.

8 Random Numbers

As you probably noticed already, the sequence of commands needed to produce even the most simple plots quite long and tedious. In the this section, you will be required to put several commands together to tell a story, we recommend writing a script to complete the tasks.

8.1 Uniform Distribution

In this section, we will explore some basic fact about statistics using MATLAB as the tool to generate and explore random numbers. Choose a random number between -1 and 1 using the (pseudo) random number generator. The command to use is:

```
>> random('Uniform', -1, 1)
```

There are two comments about the first entry to this function.

- The first entry tells the function named `random` how to distribute the random numbers. If we want to make all of the numbers equally likely to be anywhere between -1 and 1, we need to select the "`Uniform`" distribution, as we have done in this example.
- The single quotes around `Uniform` is an indication to MATLAB that the sequence of characters `Uniform` is not a variable name, but rather a new type of data called a string. Strings are a grouping of characters like `'a'` or `'b'`. The computer will interpret the variable name `a` differently than character `'a'`.

Now imagine that this random number represents the outcome of an experiment with a single trial. Clearly, a single trial doesn't provide us with much information and the obvious thing to do is to perform many identically prepared trials. Let's pretend that we have repeated each measurement 5 times. We can do this by making an array of "results". Add two more arguments to the `random` function and store them in the variable name `trials`:

```
>> trials = random('Uniform', -1, 1, 1, 5)

trials =

    0.7386    0.1594    0.0997   -0.7101    0.7061
```

This produced a 1×5 array of random numbers between -1 and 1. Okay, let's visualize the data using the `plot` command.

```
>> plot(trials,'o')
```

This generates a scatter plot. The x-axis is the index number denoting where in the sequence of measurements it occurred, but it isn't very interesting. We are likely more interested in how often a certain number shows up. This will require a histogram, which is simple to make with the following command.

```
>> hist(trials)
```

Of course, we only have 5 trials.

Questions:

1. What happens to the distribution as the number of trials (N) progresses from $N = 5, 100, 1000$ and 1E6? You might consider putting placing more than one plot on the figure using the `subplot` command. For last two values of N, change the number of bins in the histogram to 30 using:

   ```
   >> hist(trials, 30)
   ```

2. For each $N = 5, 100, 1000$ and 1E6, compute the mean and standard deviation. How does it change as N increases?

8.2 Gaussian Distributions

A very important distribution that we will discuss in considerable detail later in the course is the Gaussian Distribution. What is a Gaussian curve you might ask? A quick Google search gave me the following info. Mathematically, the Gaussian function is

$$f(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{8.1}$$

and its graph is given in figure 1.
The mean, or average, is given by μ and the standard deviation is given by σ. The standard deviation is a parameter that characterizes the width; the larger σ is, the wider the

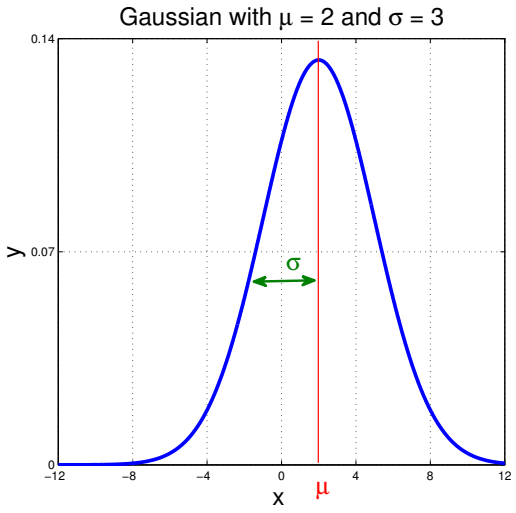

Figure 1: Gaussian distribution with $\mu = 2$ and $\sigma = 3$.

distribution is. The strange factors in front of the eq. (8.1) ensure the area under the curve is unity.

$$\int_{-\infty}^{\infty} f(x;\mu,\sigma)\; dx = 1\,, \tag{8.2}$$

Repeat the exercise of the previous section using a gaussian distribution with mean $\mu = 2$ and $\sigma = 3$. In MATLAB you might use the commands:

```
>> random('Normal', 2, 3,1,5)

ans =

   -1.0717    3.3159    1.2737    4.1638   -2.6045
```

Questions:

1. What happens to the distribution as the number of trials (N) progresses from $N = 5, 100, 1000$ and 1E6? You will need to adjust the binning when N gets large.
2. For each $N = 5, 100, 1000$ and 1E6, compute the mean and standard deviation. How does it change as N increases?