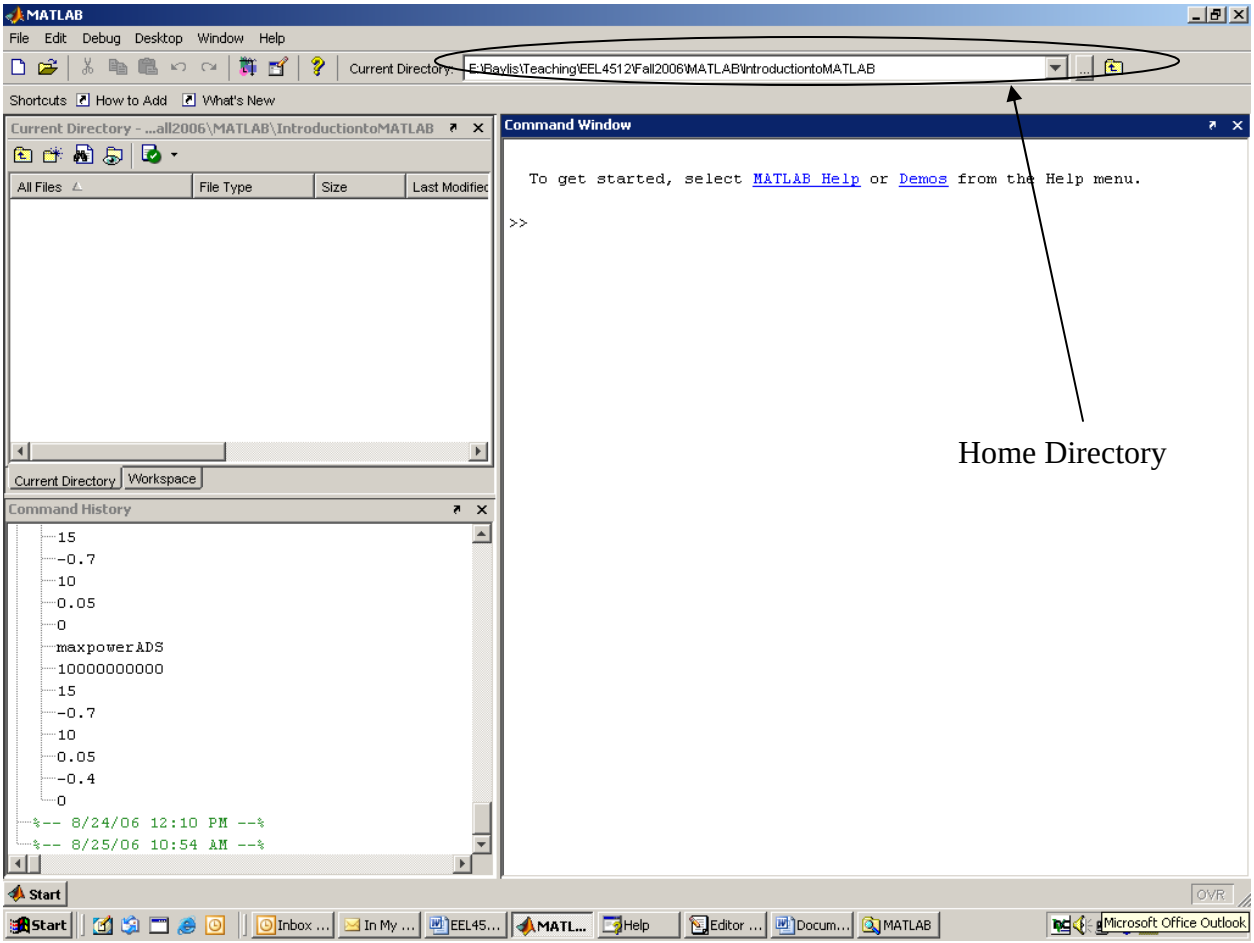
EEL 4512 – INTRODUCTION TO COMMUNICATION SYSTEMS

INTRODUCTION TO MATLAB

This handout will introduce some basic procedures and capabilities of MATLAB. Work along with the handout and this will give you a primer into using the software.

After starting MATLAB, the following window will appear:



The command window (on the right) is where execution of commands, assignments of variables, and other operations are performed. The command history shows the history of the commands. The upper left window shows the files in the present home directory (the home directory is shown in the upper right). Any functions called by the program should be in this directory.

Let's begin by performing some simple mathematical operations in the command window. The first operation will be the simple addition of two numbers:

We will first perform the addition of two numbers. Type "x = 1", followed by Enter, in the command window. Then type "y = 1", followed by Enter in the command window. Finally the sum is

computed by typing “x+y” and pressing enter. The sum appears below the command. The command window should look like this when finished:

```
>> x = 1

x =

     1

>> y = 1

y =

     1

>> x + y

ans =

     2

>>
```

Next, we will learn how to define functions and perform operations on functions in MATLAB. Following the order of commands below, we will define x(t) = t, y(t) = t + 1 and then perform addition and multiplication operations. Finally, the plot of the addition and multiplication is performed.

The first step is to define the values of the independent variable t. The following establishes t from -5 to 5 in steps of 0.01:

```
>> t=[-5:0.01:5]
```

When you press Enter, you will see a large amount of numbers printed on the screen. These are the values of t. Now type the same command in the screen again, but with a semicolon following. Upon pressing Enter, you will see that the numbers are not printed. The semicolon causes the value of the computation you typed to not be printed on the screen; however, the operation is equally valid.

```
>> t=[-5:0.01:5];
```

Now you can define x(t) and y(t):

```
>> x=t;
```

>> y=t+1;

For the addition, define m(t) = x(t) + y(t) and n(t) = x(t)y(t):

>> m=x+y;

>> n=x.*y;

Notice the period "." After the x in the multiplication operation. This is necessary because x and y are matrices in MATLAB, so a simple "x*y" would cause MATLAB to try to compute the matrix multiplication of the column vectors x and y. Obviously, this is impossible because the inner matrix dimensions do not agree. What we desire is a component by component multiplication, which this dot allows to be performed.

Now let's plot x(t), y(t), m(t), and n(t) in a single plot box with four subplots. This can be performed by using the built-in "subplot" function in MATLAB. To learn about any function type "help", followed by the function. An explanation will appear in the command window explaining the function, along with the necessary input arguments and the output(s) of the function:

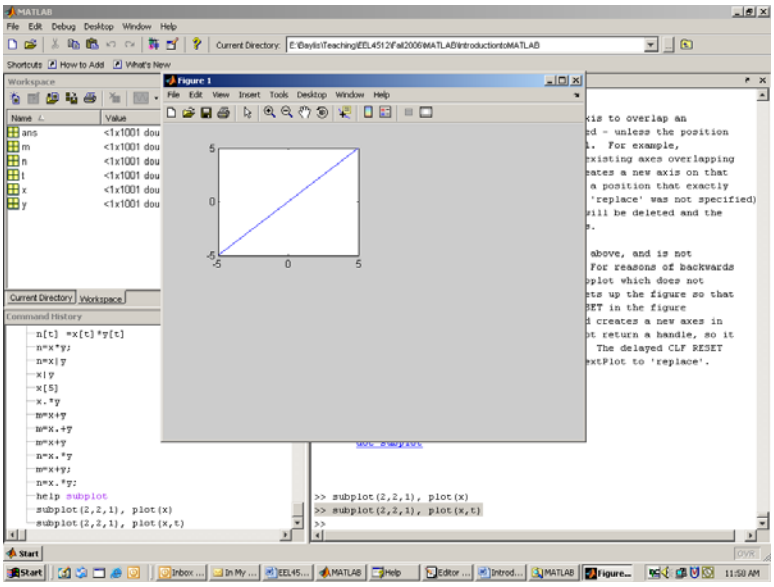
```
>> help subplot
 SUBPLOT Create axes in tiled positions.
    H = SUBPLOT(m,n,p), or SUBPLOT(mnp), breaks the Figure window
    into an m-by-n matrix of small axes, selects the p-th axes for
    for the current plot, and returns the axis handle.  The axes
    are counted along the top row of the Figure window, then the
    second row, etc.  For example,

        SUBPLOT(2,1,1), PLOT(income)
        SUBPLOT(2,1,2), PLOT(outgo)

    plots income on the top half of the window and outgo on the
    bottom half. If the CurrentAxes is nested in a uipanel the
    panel is used as the parent for the subplot instead of the
    current figure.

    SUBPLOT(m,n,p), if the axis already exists, makes it current.
    SUBPLOT(m,n,p,'replace'), if the axis already exists, deletes it and
    creates a new axis.
    SUBPLOT(m,n,p,'align') places the axes so that the plot boxes
    are aligned instead of preventing the labels and ticks from
    overlapping.
    SUBPLOT(m,n,P), where P is a vector, specifies an axes position
    that covers all the subplot positions listed in P.
    SUBPLOT(H), where H is an axis handle, is another way of making
    an axis current for subsequent plotting commands.

    SUBPLOT('position',[left bottom width height]) creates an
    axis at the specified position in normalized coordinates (in
    in the range from 0.0 to 1.0).

    SUBPLOT(m,n,p, PROP1, VALUE1, PROP2, VALUE2, ...) sets the
    specified property-value pairs on the subplot axis. To add the
    subplot to a specific figure pass the figure handle as the
    value for the 'Parent' property.
```

```
If a SUBPLOT specification causes a new axis to overlap an
existing axis, the existing axis is deleted - unless the position
of the new and existing axis are identical.  For example,
the statement SUBPLOT(1,2,1) deletes all existing axes overlapping
the left side of the Figure window and creates a new axis on that
side - unless there is an axes there with a position that exactly
matches the position of the new axes (and 'replace' was not specified),
in which case all other overlapping axes will be deleted and the
matching axes will become the current axes.

SUBPLOT(111) is an exception to the rules above, and is not
identical in behavior to SUBPLOT(1,1,1).  For reasons of backwards
compatibility, it is a special case of subplot which does not
immediately create an axes, but instead sets up the figure so that
the next graphics command executes CLF RESET in the figure
(deleting all children of the figure), and creates a new axes in
the default position.  This syntax does not return a handle, so it
is an error to specify a return argument.  The delayed CLF RESET
is accomplished by setting the figure's NextPlot to 'replace'.

See also  gca, gcf, axes, figure, uipanel


Reference page in Help browser
   doc subplot
```

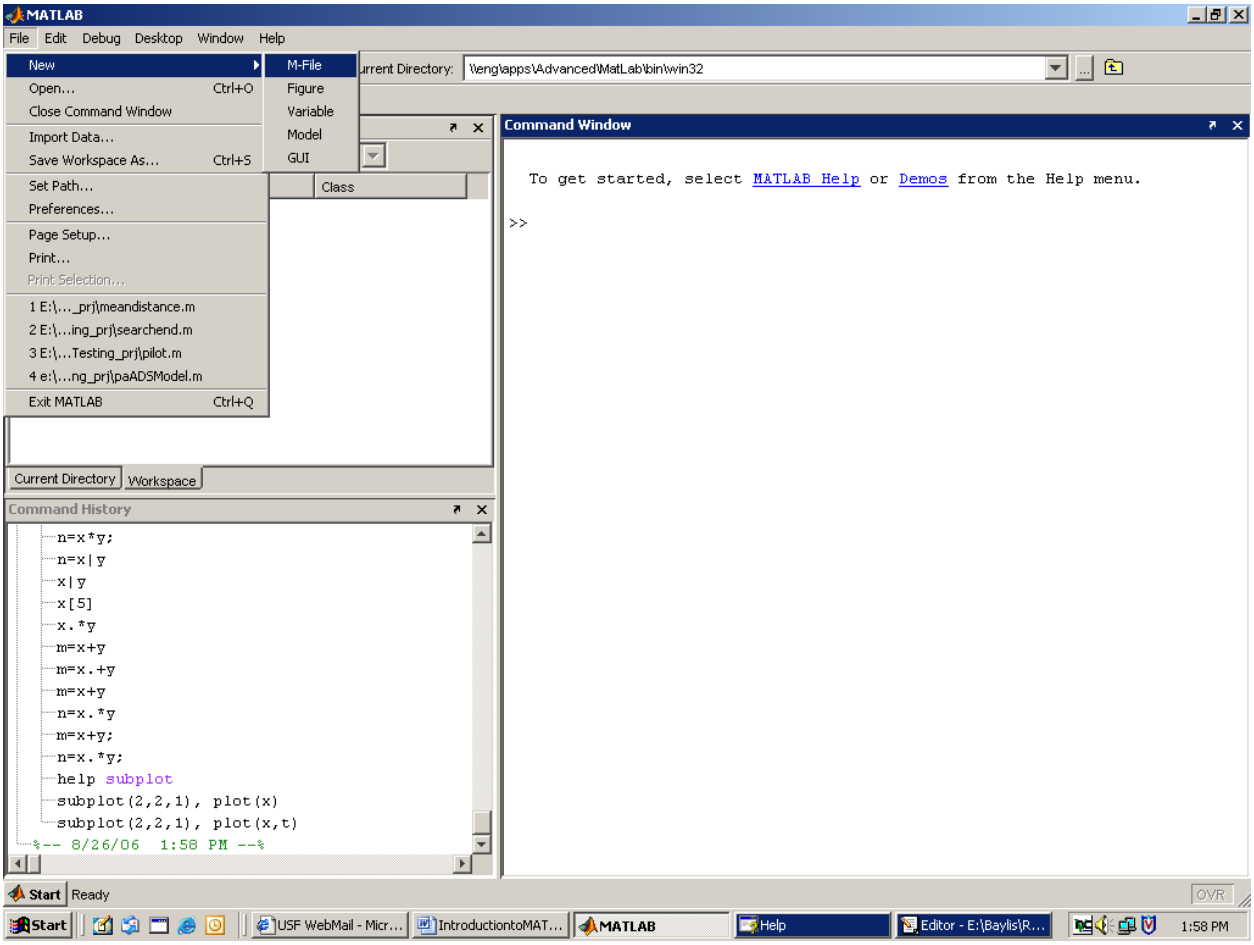
Thus “subplot(2,2,1), plot (t,x)” will form a plot box for the function that is a 2 x 2 array of subplots. The plot to be used for the plot at present will be the first plot (the first row is counted first, followed by the second row). x will be plotted on the vertical axis with t on the horizontal axis. The command line of this is as follows:

>> subplot(2,2,1), plot(t,x)

The plot box pops up and appears as follows:



The next step is the development of a program, or "M-file," to perform all of these tasks. The commands are the same as those used in the command window, but the M-file can be repeatedly called and thus this is more convenient for operations with a larger number of commands. To create a new M-file, go to File → New → M-file.

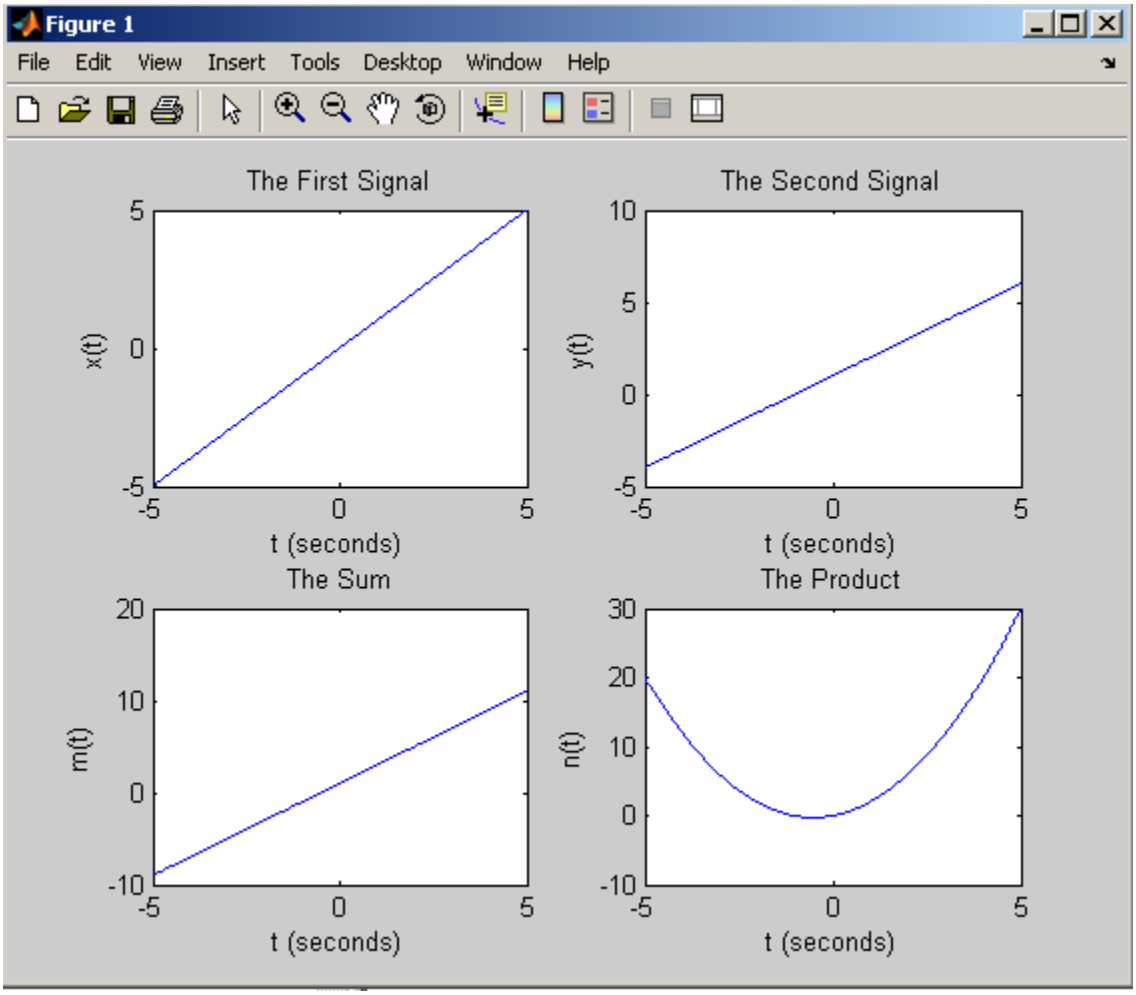


A new window will open. You can begin typing code into this window. We would like to create a program that defines the functions x(t) and y(t), then calculates the sum (m(t)) and product (n(t)) of the two functions and plots all four functions. The m-file code appears as follows:

```
function sumproduct()
t=[-5:0.01:5];    %You write comments like this, with a percent sign.
x=t;
y=t+1;
m=x+y;
n=x.*y;

%Plot the first input function, x(t):
figure                    %This command opens a new graph or figure.
subplot(2,2,1), plot(t,x);          %This sets up the first plot.
title('The First Signal');          %This titles the graph.
xlabel('t (seconds)');          %xlabel labels the horizontal axis.
ylabel('x(t)')                  %ylabel labels the vertical axis.

%Plot the second input function, y(t):
subplot(2,2,2), plot(t,y);      %Form plot (xaxis, yaxis)
title('The Second Signal');
xlabel('t (seconds)');
ylabel('y(t)');

%Plot the sum, m(t) = x(t)+y(t):
subplot(2,2,3), plot(t,m);
title('The Sum');
xlabel('t (seconds)');
ylabel('m(t)');

%Plot the product, n(t)=x(t)y(t):
subplot(2,2,4), plot(t,n);
title('The Product');
xlabel('t (seconds)');
ylabel('n(t)');
```

After you have typed in this code, go to File → Save and type in the file name sumproduct (this will save the file as “sumproduct.m”).

Return to the main MATLAB window and type “sumproduct” at the prompt. The function will execute and should result in a plot that appears as follows:



If you would like to repeat the previous operation performed in the main MATLAB window, then simply press the upward arrow key. This re-enters the previous command-line statement. Thus you can run this program again by pressing the up arrow followed by Enter.