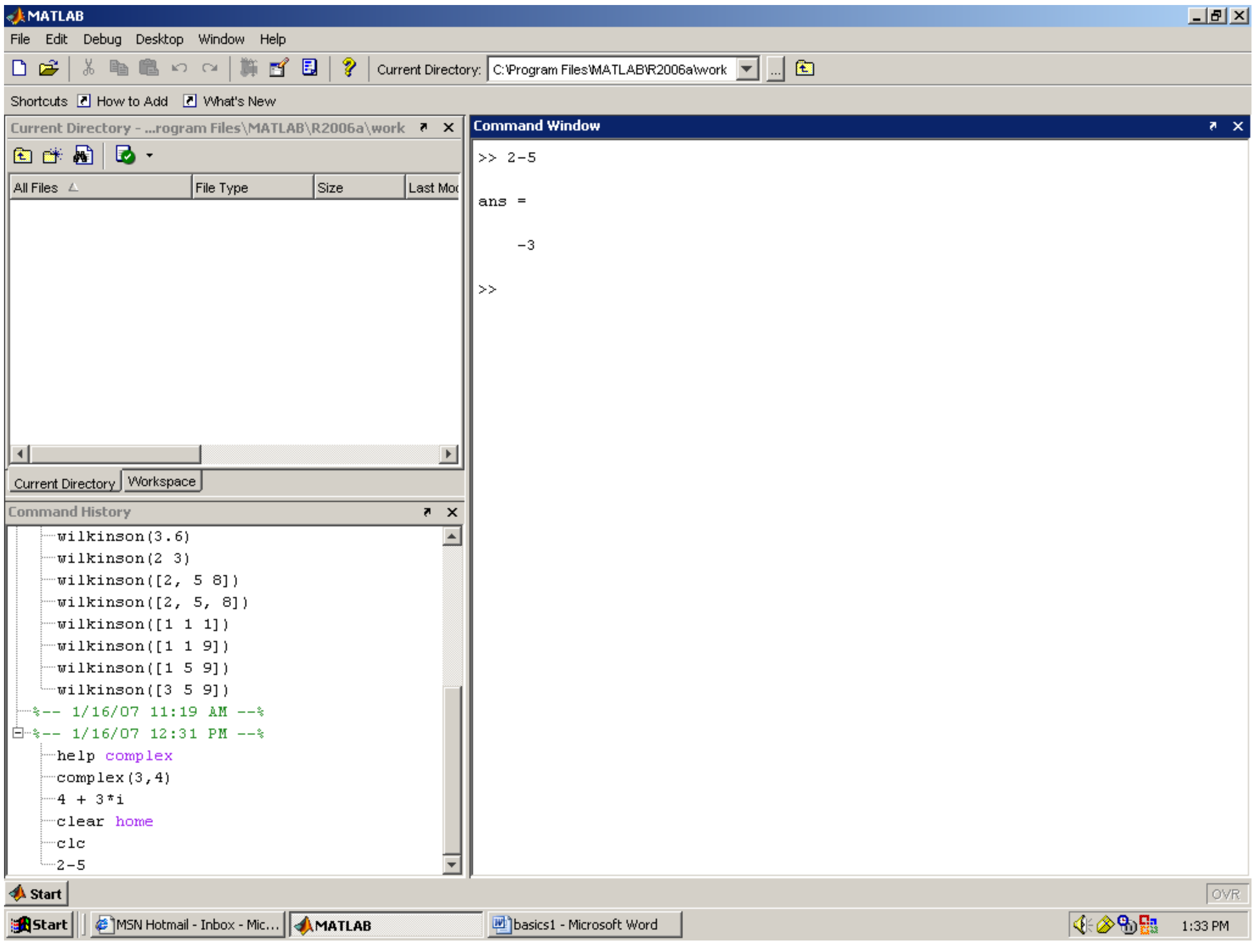
MATH 495.3 (CRN 13695)

Lab 1: Basics for Linear Algebra and Matlab

Below is a screen similar to what you should see when you open Matlab. The command window is the large box to the right containing the calculation 2-5 = -3. Any m-files (programs) you've created and used previously will appear in the upper left-hand box. Previous commands entered into the command window will appear in the command history box in the lower left.


MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\Program Files\MATLAB\R2006a\work
Shortcuts
How to Add
What's New
Current Directory - ...rogram Files\MATLAB\R2006a\work
All Files
File Type
Size
Last Moc
Current Directory
Workspace
Command History
Command Window
Start
OVR
MSN Hotmail - Inbox - Mic...
MATLAB
basics1 - Microsoft Word
1:33 PM

```
>> 2-5

ans =

    -3

>>
```

```
wilkinson(3.6)
wilkinson(2 3)
wilkinson([2, 5 8])
wilkinson([2, 5, 8])
wilkinson([1 1 1])
wilkinson([1 1 9])
wilkinson([1 5 9])
wilkinson([3 5 9])
%-- 1/16/07 11:19 AM --%
%-- 1/16/07 12:31 PM --%
help complex
complex(3,4)
4 + 3*i
clear home
clc
2-5
```

Basic Operations

We'll begin working just with the command window, which allows you to do computations as well as access programs you've written. Type your computations/commands after the prompt, shown below:

```
>>
```

The simple operations of +,-, *, /, ^ work as you would expect. For example, type

```
>> 2 - 5
ans =
          -3
```

The answer should appear below unless you end the line with a semi-colon, in which case, Matlab will evaluate the line, but will not display the result. Try it.
You can also store values to variables to use in later computations:

```
>> a = 5
a =
        5
>> b = -2
b =
        -2
>> x = 3;
>> y = a*x + b
y =
        13
```

To recall the value of a particular variable, just type the variable:

```
>> x
x =
        3
```

To clear the value of x, type

```
>> clear x
```

You also have the option of overwriting the value of a variable. Note that if you overwrite a value in an equation, the value of the output will not be updated. For example, if you update >>b=4; and then ask Matlab for the value of y, you'll get the old value as that is the last known value Matlab had of y.

```
>> y
y =
        13
```

This is fixed by asking Matlab to re-evaluate the y with the new value of b.

```
>>y = a*x+b
y =
        19
```

Complex Numbers
The complex number 4 + 3i can be entered into Matlab just as any other operation:

```
>>-4 + 3*i
ans =
        -4.0000 + 3.0000i
```

You can also enter a complex number with the “COMPLEX” command:

```
>> complex(-4,3)
```

```
ans =
          4.0000 + 3.0000i
```

m-files
If you have a lot of complicated calculations or repeated calculations with different values, you will probably want to create an m-file (a Matlab program). This is a separate file (and window) you can work in and store a series of calculations, then ask Matlab to evaluate them from the command window. The values of variables in the command window and the m-file are linked. To see this, open a new m-file (from the file menu) and save it as program1.m in the current directory. The directory you're working in is in a box at the top of the page – be sure the m-files you're saving (and trying to access) are in the same directory. Type y = a*x+b in the m-file and save it. (Note: you must save the m-file every time you make any changes in order for Matlab to recognize them.) Now click in the command window to activate it (alt + tab also works) and type

```
>>program1
```

You should see the appropriate value of y appear in the command window.

When writing your code in an m-file, it is a very good idea to put comments in your code. This will help you and others understand the code you have written (for example, if you decide to use some code you have written six months from now, comments in your code will really help). A comment is created by typing a "%" -- everything on the rest of the line will be ignored by the interpreter. You can be sure that something is commented if it turns green. In your m-file, add a comment to the line y = a*x+b that describes what you are doing (of course, a comment for this line of code is really unnecessary, but it is good practice). It is good practice to begin each m-file with a comment describing the main objective of the file.

Vectors
There are a few different ways to enter vectors into Matlab. (Also see **Arrays**.) For a 1x3 row vector A, type

```
>> A = [4  3  0.6];
or
>> A = [4, 3, 0.6];
```

Notice each entry in a row is separated by a space or comma. Be sure to use square brackets.
To enter the 3x1 column vector B, type

```
>> B = [5  -3  2.7]';
>> B = [5, -3, 2.7]';
>> B = [5; -3; 2.7];
```

Here, we've typed a row vector and then the apostrophe symbol at the end to indicate the transpose. Recall the transpose of a matrix switches the first row to the first column, the second row to the second column, etc. (We'll talk work with this more later.) Another option is using the semicolon (instead of a comma) to indicate the start of a new row.

Matrices
For a 2x3 matrix C, follow the same rules: a comma to separate entries in a row and a semi-colon to start a new row.

```
>>C = [4  0  -5; 2  6.3  1]
C =

    4.0000         0   -5.0000
    2.0000    6.3000    1.0000
```

Matrix operations, +, -, *, ^ will all work as you would write them on paper as long as the dimensions are correct. (If they're not, you'll get an error from Matlab.)

Recall, addition and subtraction of matrices require matrices to have the same dimension and the operations are done element by element:

$$A = \begin{bmatrix} 2 & -5 & 1 \\ 4 & 0 & -3 \end{bmatrix}, \quad B = \begin{bmatrix} 7 & 3 & 8.5 \\ -3 & -6 & 9 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 2+7 & -5+3 & 1+8.5 \\ 4+(-3) & 0+(-6) & -3+9 \end{bmatrix} = \begin{bmatrix} 9 & -2 & 9.5 \\ 1 & -6 & 6 \end{bmatrix}$$

For scalar multiplication of a matrix, each element of the matrix is multiplied by the scalar:

```
>>7*A
ans =
          14   -35   7
          28    0    -21
```

For completeness, we include matrix multiplication below, where the number of columns in the first matrix has to be the same as the number of rows in the second matrix. We multiply corresponding elements in the first row (of the first matrix) by the elements in the first column (of the second matrix) and add them together – place the result in the first row and first column of the resulting matrix. Next multiply corresponding elements in the first row (of the first matrix) by the elements in the second column (of the second matrix) and add them together – placing the result in the first row and second column of the resulting matrix. Continue with the first row until all the columns of the second matrix have been multiplied. Then proceed to the second row of the first matrix and multiply (in the same manner) all the columns of the second matrix, and so on.

$$C = \begin{bmatrix} 4 & 0 & -5 \\ 2 & 6.3 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 4 & 2 \\ 2 & -5 \\ 0.5 & 1 \end{bmatrix}$$

$$C * D = \begin{bmatrix} 4*4+0*2+(-5)*0.5 & 4*2+0*(-5)+(-5)*1 \\ 2*4+6.3*2+1*0.5 & 2*2+6.3*(-5)+1*1 \end{bmatrix} = \begin{bmatrix} 1\ .5 & 3\ 3 \\ 2\ .1 & -2\ .5 \end{bmatrix}$$

In Matlab, we enter the above matrices and calculation as follows:

```
>> C = [4  0  -5; 2  6.3  1]
C =
    4.0000         0    -5.0000
    2.0000    6.3000    1.0000

>> D = [4 2; 2 -5; 0.5 1]
D =
    4.0000    2.0000
    2.0000   -5.0000
    0.5000    1.0000

>> C*D
ans =
   13.5000    3.0000
   21.1000  -26.5000
```

To identify or reference a particular element, row, or column of a matrix, we use the following notation (considering the matrix D above):

```
>> D(3,2)
ans =
        1.0000
>> D(2,:)
ans =
        2.0000     -5.0000
>>D(:,2)
ans =
        2.0000
        -5.0000
        1.0000
```

There is another type of multiplication with vectors and matrices. It may be necessary to multiply them component-wise. This can be done by placing a "." in front of the operation. For example:

```
>> a = [1,2,3,4];
>> b = [6,2,4,5];
>> a.*b
ans =
     6     4    12    20
```

If you tried this without the "." you would get an error, unless you use the transpose (which is still a fundamentally different result):

```
>> a*b
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

```
>> a*b'
ans =
   42                    % as it should be
```

This component-wise process can be done with multiplication ".*", division "./" and exponentiation ".^".

Arrays

Arrays are simply matrix vectors whose elements follow a particular sequence. A short-hand way of writing an array is as follows:

```
>>a = (-1:1:2)             % notice open parentheses instead of brackets
a =
    -1  0  1  2
```

This gives an array starting from -1, increasing by a step of 1, until the endpoint 2 is reached. Experiment with this using positive and negative steps and using decimal endpoints.
You can also use 'linspace' to create an array that specifies the starting value (0), ending value (10), and number of points you want between them (5).

```
>>b = linspace(0,10,5)
b =
          0  2.5000  5.0000  7.5000  10.0000
```

Mathematical operations with arrays ALWAYS require that the arrays be of the same length and +, -, *, / all are done on an element-by-element basis.
To specify a particular element in array, use similar notation as with matrices (above).

```
>>b(2)
ans =
          2.5000
>>a(2:4)
ans =
          0  1  2
```

There are some specific vectors and arrays in MATLAB. The following creates a 5 by 5 dimensional array of 1's.

```
>> y = ones(5, 5)

ans =
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
```

Also, “zeros(n,m)” creates an n by m matrix of zeros, and “eye(n)” creates the nxn Identity matrix.

Help

Everybody will get stuck in Matlab, regardless of how long you’ve been using it. When you get to this point or you keep getting an error that you can’t figure out, you have a few options: 1) Try the pull down Help menu in Matlab. 2) Use the online help. 3) You can simply type the word help to get the help menu in the command window. If you type a topic after the word help, Matlab gives you help on the topic - if it exists as you’ve typed it. For example,

```
>>help complex
```

Option 4) is to ask me or your neighbor. I’m here to help and I’m happy to do it.

Assignment 1 – due: Friday, January 31
First, create an m-file called assignment1_yourname.m that contains every question on the assignment. Start each new question with a comment so that I can easily see them. After completing each question, please run your m-file (or you can just copy the code needed only to answer the current question and paste it in the command window) to get the output. Copy the output under each question, and comment the copied output. Hint: You can comment a block of code by highlighting the block and pressing ctrl+r and uncomment with ctrl+t.

For example, the first part of your m-file should look like this:

```
% Assignment 1 – Simon Tavener

% Q1
%a)
x=5;
y=-2;
2*x^2+5*y-1

% ans =
%
%     39

%b)
```

Please e-mail your lab to me by the beginning of the next lab period. tavener@math.colostate.edu

Q1: Define x = 5 and y = -2 and find (a) $2x^2+5y-1$ and (b) $2*(x^2+5y)-1$

Q2: Find (a) A+B, (b) A-B, and (c) 3A-2B for the matrices:

$$A=\begin{bmatrix}-1 & 9 & 4\\ 2 & -3 & -6\\ 0 & 5 & 7\end{bmatrix},\qquad B=\begin{bmatrix}-4 & 9 & 2\\ 3 & -5 & 7\\ 8 & 1 & -6\end{bmatrix}$$

Q3: This question explores the concept of variables.

(a) In the command window (not your m-file), type “x=3;” and hit enter. Now type “2*x+4” in the command window. Now type ans. What is the value of ans? Try typing “4*ans”. Now what is ans? Answer these questions as comments in your m-file.
(b) In the m-file, type “x=4;”. On the next line, type “x = x+1”;. After running these lines of code, what is the value of x? Try it, and record your answer with a short explanation in the m-file as a comment.
(c) Type “z = rand(3,2)” and record the output as a comment. Also add a short comment describing what is happening. Try some different inputs, such as rand(4,1), if you need to, or type “help rand” in the command window.

Q4. In MATLAB, strings are simply character arrays (or vectors). This question explores the concept of strings. Define the string s as ‘abcd’ by typing “s = ‘abcd’ ;” in the m-file.
(a) What is the value of s(3)?
(b) Also define string t as ‘efgh’. What is z = [s,t]?

(c) You can convert numbers to strings using the command num2str. What is the result of y = [s, num2str(65), t]? What happens if you don't use the num2str command?

Q5. Type "x = rand(1,10)" and "y = x <= 0.5" on separate lines. Now type "x(y)". Explain all of the results as a comment.

Q6. You need to know how to create a nice looking plot in MATLAB. As an introduction, try the following:

```
x = linspace(0, 2*pi, 100);
y = cos(x);
plot(x, y)
title('cosine function')
xlabel('x')
ylabel('y')
```

(a) What happens if you type "`plot(x, y, 'r-')`"? What about "`plot(x, y, '-', 'LineWidth', 3, 'Color', [0.5 1.0 0.5])`"? Describe the results as a comment. Use the help menu and search for LineSpec and ColorSpec if you need to.

(b) Create a plot that is labeled Figure 5 that includes the sine function and the cosine function defined on the domain from –π to π. Make the sine function a green dotted line and the cosine function a thicker red line (solid line). Title the graph and label the axes appropriately. Use the help menu to find out how to label it *figure* 5 and how to *hold on* to multiple plots on the same figure. Save the figure and send it to my e-mail as well.