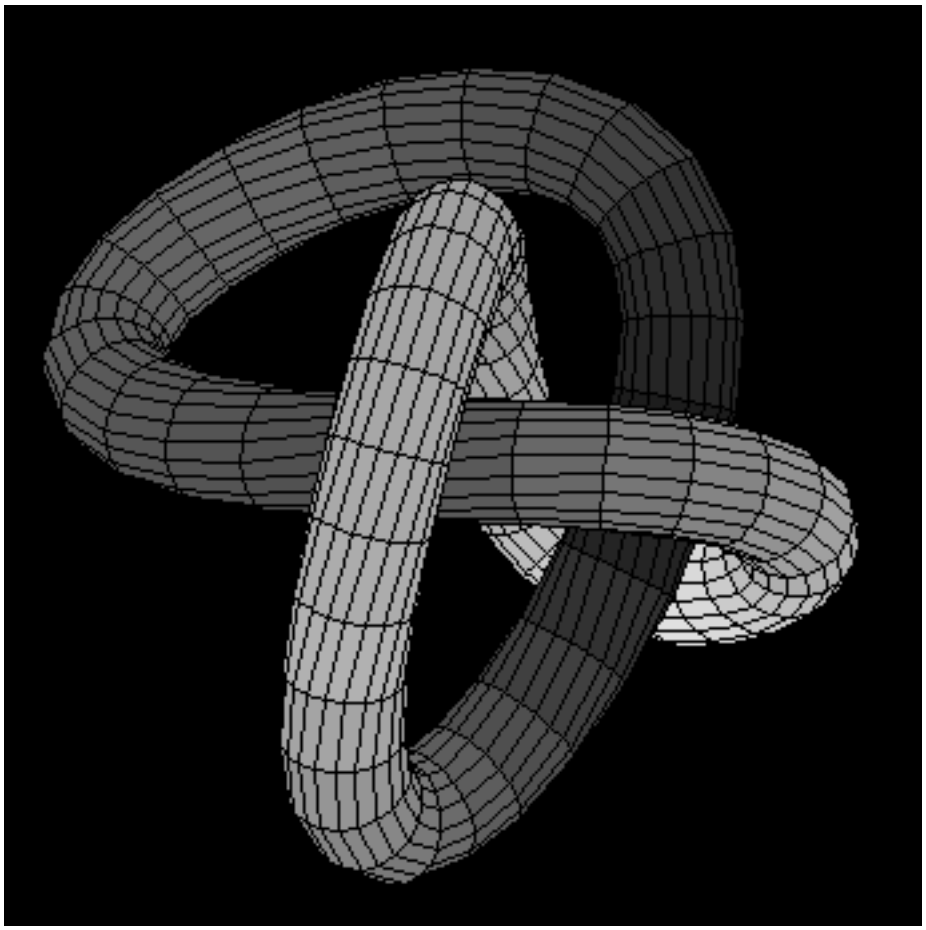
Revised: January 13, 2006

Math and MATLAB Workshop



Revised: January 13, 2006

Matlab Workshop

Introduction to Matlab

Matlab is the industry-standard technical computing environment with capabilities in numeric computation and visualization. It integrates the key requirements of a technical computing system: numeric computation, graphics and application-specific tools, and cross-platform capabilities. Matlab provides a versatile environment that has reached an enormous range of users in academia, industry, and government. Over one thousand universities around the world use Matlab for research and in their education curricula, while hundreds of companies design, model, simulate and analyze systems with these same tools. In addition to use in the Vibration and Control Laboratory, Matlab can be used on the PC and UNIX Networks throughout the Engineering Building.

In this workshop you will experiment with the most commonly used functions and features of Matlab used in the ME 451 lab. You will perform the basic mathematical procedures required of system and control analysis using both conventional methods and the numerical tools provided by Matlab.

Matlab Features

Command Window - The default window that opens when you start is named the Command window. Matlab commands can be entered on the command line and executed one at a time.

Scripts – Scripts are Matlab program files executed by the Matlab command interpreter. Any valid command for the command window can be included as a line in a Matlab script file. Although you could do all your work on the command line, it is not advisable, because, commands must be entered one line at a time. The preferred way of working with Matlab is to prepare and execute a script file in text format with the suffix "m". A script file contains a simple text of Matlab commands listed line by line. The commands are executed when the script file is executed by typing the filename into Matlab's command window. Normally these script files are developed by testing each of the script's commands in the command window and then adding each tested command to the script file. The sequence of tested commands can then be executed without repeatedly entering many commands on the command line while having to retype the entire line when you make a spelling error. You can use a script, write as many commands as you like then execute them all at once. Using scripts provides a text file that documents your work. If you use the command line, there is no permanent record of the commands you entered.

Never use the command line when you can use a script instead!

Selected Matlab Functions

% - Comments can be inserted on any Matlab line by proceeding them with the "%" character. The % character for documentation; Matlab does not evaluate any text on a line following the % character.

help – Information about any Matlab function can be displayed on the screed by entering

help < function_name>

Matlab has extensive help on all commands. To see some of Matlabs features, Find the command window with the ">>" showing and type the command *help demo.* Matlab will return text in the command window documenting the command *demo*. Information regarding plotting with Matlab can be found with the command *help plot*.

plot - Matlab has powerful plotting capabilities. You can plot many lines on one graph, add a title, axis labels, a grid...

ode23/ode45 - The Matlab command *ode23* and *ode45* integrate a set of an ordinary differential equations (ODE). You must create one m-file that defines the ODE and a second m-file to call the ODE using the *ode23* or *ode45* commands.

lsim - The *lsim* command is similar to the *ode23* command, but can only evaluate linear ODEs. However, when you are analyzing a linear ODE, lsim is much easier to implement than *ode23* or *ode45*.

step - The *step* command computes the unit step response of a system. The system can be defined by a transfer function or in state space formulation. This command is most commonly used in the 451 course.

conv - The *conv* function multiplies two polynomials algebraically. This function is useful when computing the response of two transfer functions in series.

series - The *series* command is a special controls application command. It computes the equivalent transfer function of two transfer functions in series. It does the same thing as the *conv* command but for both the numerator and denominator at once.

feedback - The *feedback* command is similar to the *series* command, except that it computes the equivalent transfer function of a feed forward and a feedback transfer function.

Procedure

Follow these procedures and answer the questions on the attached short form. Be sure to attach both the Matlab scripts and the output to the short form.

0. Start Matlab

Use either the "Matlab" entry in the "Programs" list in the "Start" menu or The "Matlab" icon on the desktop to start the Matlab program on the computer

1. Linear Algebra – Matrix and Vector Computations:

The multiplication of matrix and vector quantities always involves multiplication of a row vector and a column vector. Your TA will supply two vectors and two matrices.

1.0 Record the vectors and matrices supplied by your lab consultant on your short form report

Multiplication of Two Vectors

Define the two vectors **a** and **b**

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

The vector **a** is made up of the scalar quantities a_i. It is not possible to multiply two "column" vectors directly. There are two possible "row time column" products of **a** and **b**,

$$\mathbf{a}'\mathbf{b} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}' \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4 \text{ and}$$

$$\mathbf{b}'\mathbf{a} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}' \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = b_1a_1 + b_2a_2 + b_3a_3 + b_4a_4$$

The single quote operator (') on the vector is the "transpose" operator that switches to columns of a vector or matrix to rows. The following vector quantity notations are equivalent:
transpose (**a**) =**a**' = $\mathbf{a}^T$.

1.1 Perform the product of two vectors in your short form using only your pen or pencil. – no Matlab yet.

Multiplication of Two Matrices

Define the two vectors **A** and **B**

$$\mathbf{A}=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{21} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{bmatrix}$$

Each matrix is made up of multiple column vectors where each column vector is made up of the scalar quantities a_{ij} where the index *i* is the row and the index *j* is the column position of the scalar. As long as the number of rows of **A** equals the number of columns of **B** , the rows of **A** can multiply the columns of **B** It is not possible to multiply two "column" vectors directly. There are two possible "row time column" products of **a** and **b** ,

$$\mathbf{AB}=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{21} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{bmatrix}=\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

where $$c_{ij}=\begin{bmatrix} a_{i1} & a_{i2} & a_{i3} & a_{i4} \end{bmatrix}\begin{bmatrix} b_{1j} \\ b_{2j} \\ b_{3j} \\ b_{4j} \end{bmatrix}$$

Note that multiplying a matrix with *n* rows and *m* columns times a matrix with *m* rows and *p* columns results in a matrix with *n* rows and *p* columns, eg.

$$\mathbf{A_{n\times m}B_{m\times p}=C_{n\times p}}$$

1.2 Compute the product of two matrices in your short form using only your pen and pencil. – no Matlab yet.

1.3 Using Matlab to Multiply Vectors and Matrices

Enter the following commands in the command window. Enter one line at a time, and press return at the end of each line.

a=[1; 2; 3] %Defines a column vector named a

b=[4; 5; 6] %Define a column vector named b

*c=a'*b %Matrix multiplication of transpose(a) * b*

After the entry of each line, Matlab prints the result on the command screen. The "%" character allows comments and all text after the "%" character on each line is ignored. Note [1;2;3] defines a column vector, where [1,2,3] or [1 2 3] defines a row vector. You should have a command screen that looks something like the result below.

```
» a=[1; 2; 3] %Defines a column vector named a
a =
     1
     2
     3

» b=[4; 5; 6] %Define a column vector named b
b =
     4
     5
     6

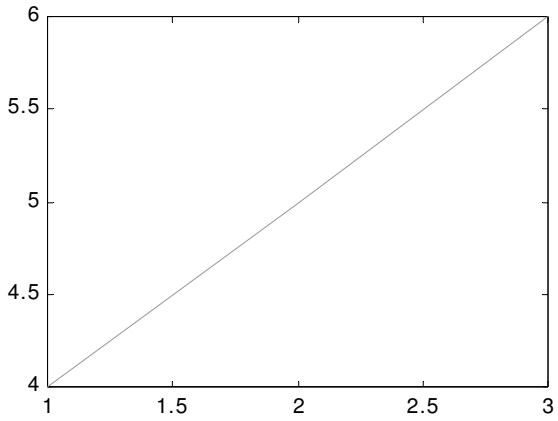
» c=a'*b %Matrix multiplication of transpose(a) * b
c =
    32
```

These Matlab commands define the two column vectors **a** and **b** then compute the product **c = a' b** .

1.3 Use Matlab to compute the product of the vectors and matrices provided by your lab consultant.

2. Plotting

The *plot* command allows easy plotting of data vectors. To learn more, type *help plot* in your Matlab command window. The command *plot(a,b,'-g')* plots the vector *b* (y axis) against *a* (x axis) using a solid green line. The result is the plot below.



2.0 Generate the plot of the vectors supplied by your lab consultant and attach to your short form

3. Scripts

Write a script file to perform the computations and plot you made in steps 1 and 2 above. The "*New m-file"* option is in the File menu. The script must be saved with a 'm' extension as in *test.m.* Save the m-file on the computer "desktop". Run your script by typing the new command *test.*

3.1 What do you see in the command window?

As a further modification to your script file, add a semicolon ";" to the end of every line in the script defining the vectors and matrices provided by your lab consultant. A semicolon at the end of any line keeps the output of that line from being displayed as the script is runs.

3.2 How does the Command window output change when semicolons are added?

Document your script so that the reader can understand its function. Add a comment preceded by a "%" character at the end of each line describing of what each line of the script does as above. Run your script again. It should still function normally. Use the Windows **Print** command to print your "test.m" file.

3.3 Attach a copy of your script file.

4. Transfer Functions

Transfer functions are defined as the ratio of the Laplace transform of an output time function to an input time function. For the differential equation,

$$\frac{d^2y}{dt^2} + 1.5\frac{dy}{dt} + 5y = 2\frac{du}{dt} + 4u$$

where $y = y(t)$ and $u = u(t)$, the input-output transfer function $G(s)$ is

$$\frac{Output}{Input} = \frac{Y(s)}{U(s)} = G(s) = \frac{N(s)}{D(s)} = \frac{2s+4}{s^2+1.5s+5}$$

A transfer function representation of a system's dynamics is often equivalent to the differential equation representation but allows easier analysis of system dynamic characteristics and response. Matlab represents such transfer functions using numerator and denominator polynomials. The Matlab function ***tf*** defines transfer functions. Find out more by entering the command *help tf*. The Matlab lines below first define a numerator polynomial "num", then a denominator polynomial "den" and finally the transfer function G. The tf command expects the "num" and "den" polynomials to be defined as row vectors of the polynomial coefficients, starting with the highest power. See below.

```
» num = [2 4]

num =
     2     4

» den = [1 1.5 5]

den =
    1.0000    1.5000    5.0000

» G=tf(num,den)

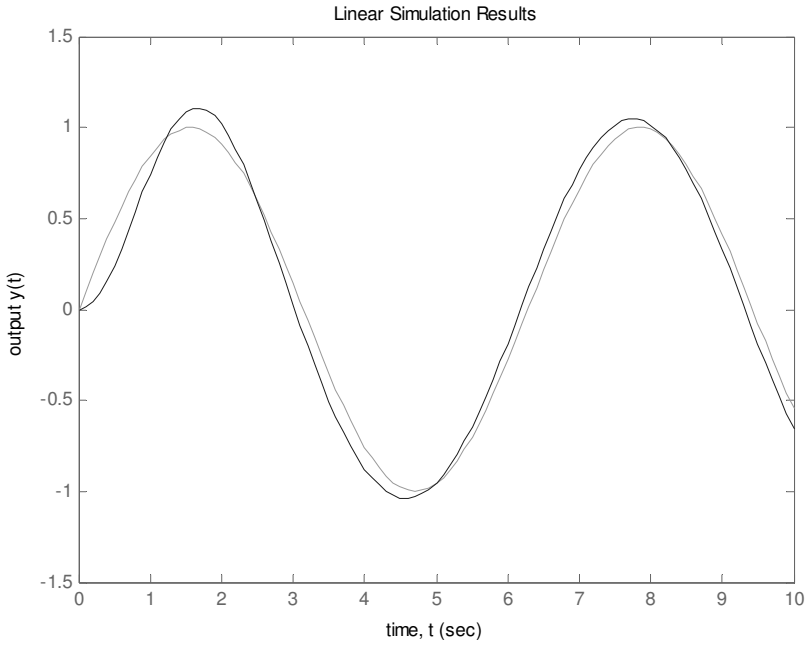
Transfer function:
    2 s + 4
---------------
s^2 + 1.5 s + 5
```

4.0 Use Matlab to define the transfer function T(S) provided by your lab consultant. Select and print that portion of your command window. Attach it to your short form report.

5. The lsim Command

Use the Matlab command *help lsim* to find information on this command. The script below defines a) a time vector consisting of $0 < t < 10$ seconds with a step size of 0.1 seconds, b) an input vector of values $u(t) = \sin(t)$ defined at each time in the vector t, and c) a transfer function g(s) which represents the system of interest. The lsim command simulates the response of the system, and plots both the input (in gray) and the output (in blue).

5.0 Create a script to plot the time solution of the linear ODE represented by the transfer function and input provided by your Lab Consultant.




```
%Simulate a system's dynamics
t=[0:0.1:10];        %define a time range 0-10 and increment 0.1
u=sin(t);            %define input function

%Define the transfer function
num = [ 2 4];        %The numerator (don't print it...)
den = [1 1.5 5];     %The denominator (don't print it either...)
G = tf(num,den)      %The transfer function (print this so I can
see it)

%Simulation and plot
lsim(G,u,t)          %With no output indicated, plot is automatic
xlabel('time, t')    %label x axis
ylabel('output y(t)')%label y axis
```

6. The step *Command*

The *step* command plots the unit step response of a transfer function. It is even easier to use than the *lsim* command but not so flexible. It only works for a step input. Use help step to get information on this command and then

6.1 Plot the step response of the transfer function provided by your lab consultant.

6.2 Plot the step response of the same transfer function to a step input defined as $\begin{cases} 0 & t = 0 \\ 20 & t > 0 \end{cases}$

Label the plots and attach the plots to your short form.

7. The "conv", "series" and "feedback" Commands

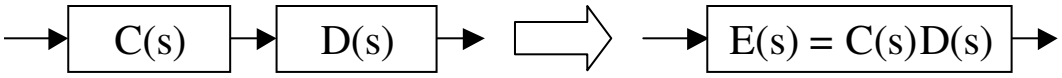
7.1 The "conv" Function Multiplies Two Polynomials

Computes the product of A and B using the conv function.

$$A = 5s^2 + 3s + 1 \quad B = 11s^2 + s$$

Use the command *help conv* to get more information on this command's syntax.

7.2 The "series" Function

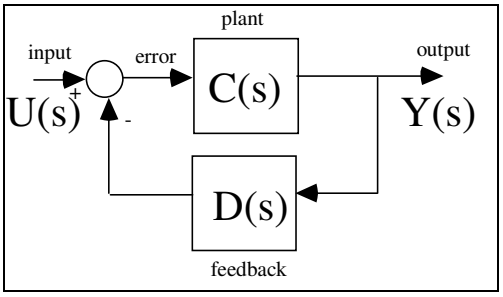


Attaches two transfer functions in series

$$C = \frac{5}{10s + 4} \quad D = \frac{2s + 1}{s^2 - 0.1s + 3}$$

Use the command *help series* to get more information on this command's syntax.

7.3 The "feedback" Function



Computes the equivalent transfer function F(s)=output/input of the system consisting of plant C connected with the feedback term D as shown in Figure 1. Use the command *help feedback* to get more information on this command's syntax.

7.0 Write a script file that

a) computes the product of A(s) and B(s) above, P(s) = A(s)B(s) using the *conv* function,

b) computes the series combination of the transfer functions C(s) and D(s) so that E(s) = C(s)D(s) using the *series* function, and

c) computes the feedback systems transfer function combination of the transfer functions C(s) and D(s) using the *feedback* function.

Print out this script file and command window output; attach them to your report.

Revised: January 13, 2006

Math and Matlab Workshop **Name:**_____

Date:_____

Short Form Report **Section / Group:**_____

i. Explain the advantage of using scripts over the command line.

1.0. Enter the vectors and matrices supplied by your TA in the spaces below

$\mathbf{d} = \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}$ $\mathbf{e} = \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}$ $\mathbf{D} = \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}$ $\mathbf{E} = \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}$

1.1 Compute the product $\mathbf{f} = \mathbf{d}'\mathbf{e} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$

1.2 Compute the product $\mathbf{F} = \mathbf{DE} = \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}$

1.3 Use Matlab to compute the results above. They should agree exactly. If not, check both your hand calculation and the Matlab commands. When they agree "select" your Matlab commands in the command window, print the selection and attach to this short form.

2.0 Plot the vector **d** vs. **e** using the plot command. The **d** values should appear on the y axis and the **e** values on the x axis. Attach your plot to this short form report.

3.1 What do you see in the command window?

3.2 How does the Command window output change when semicolons are added?

3.3 Attach a copy of your script file to this short form report.

4.0 Record the transfer function provided by your lab consultant. Use Matlab to define the transfer function T(s) provided by your lab consultant. Select and print that portion of your command window. Attach it to your short form report.

T(s) = _____

5.0 Create a script to plot the time solution of the linear ODE represented by the transfer function T(s) and input u(t) provided by your Lab Consultant. Attach a copy of the script and plot to this report. Record u(t) below.

u(t) =

6.1 Use the step command to plot the unit step response of the transfer function T(s). Write the command below, **label the plot** and attach the plot to this report.

6.2 Use the step command to plot the step response of the transfer function T(s) to the step function $\begin{cases} 0 & t = 0 \\ 20 & t > 0 \end{cases}$.

Write the command below, **label the plots** and attach the plot to this report.

7.0 Write a script file that

a) computes the product of A(s) and B(s) above, P(s) = A(s)B(s) using the *conv* function,

b) computes the series combination of the transfer functions C(s) and D(s) so that E(s) = C(s)D(s) using the *series* function, and

c) computes the feedback systems transfer function combination of the transfer functions C(s) and D(s) using the *feedback* function.

Print out this script file and command window output; attach them to your report.