Christopher Lum
lum@u.washington.edu

Intermediate Matlab Tutorial

Introduction

This document is designed to act as a mini homework assignment to familiarize the reader with more Matlab functions and operations. Every step will not be explained complete and only periodic checks will ensure that the reader does not become lost. The reader should have already read through the Beginner's Matlab Tutorial document or have an introductory level of experience with the program. For any questions or concerns, please contact

Christopher Lum
lum@u.washington.edu

Problem Statement

For this problem, let's consider the standard mass/spring/damper problem as shown in Figure 1. To avoid confusion, we will denote the horizontal distance as *z(t).*

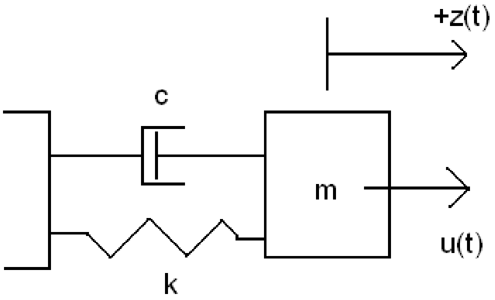


Figure 1: Spring mass damper

Recall that the equations of motion can be obtained by applying Newton's second law. This yields a second order, linear, differential equation of the form

$$\ddot{z}(t) = \frac{1}{m}u(t) - \frac{k}{m}z(t) - \frac{c}{m}\dot{z}(t) \qquad \textbf{Equation 1}$$

By choosing the state vector as $\bar{x} = \begin{pmatrix} z & \dot{z} \end{pmatrix}^T$ and the output as $y = z$, the state space representation of this system can be written as

$$\dot{\bar{x}} = A\bar{x} + Bu \qquad \textbf{Equation 2}$$
$$y = C\bar{x} + Du$$

$$\text{where} \quad A = \begin{pmatrix} 0 & 1 \\ -k/m & -c/m \end{pmatrix} \quad B = \begin{pmatrix} 0 \\ 1/m \end{pmatrix} \quad C = \begin{pmatrix} 1 & 0 \end{pmatrix} \quad D = 0$$

Analyzing with Matlab

For this problem, let k = 1, m = 1, c = 1.

1. Start a new m-file and enter these matrices.

Functions:	*None*
Note:	*Enter matrices using square brackets. Use spaces or commas to separate elements in the same row and use the semicolon to start a new row*

2. To check the internal stability of the system, calculate the eigenvalues of the A matrix. Also obtain the eigenvectors.

Functions:	*eig*
Note:	*Remember, to obtain help about a function type "help eig" in the Command Window.* *You should obtain eigenvalues of* $\lambda = -0.5 \pm 0.866i$

3. Verify that the eigenvectors and eigenvalues obtained satisfy the definition

$$(A - \lambda_i I)v_i = 0 \qquad \textbf{Equation 3}$$

Functions:	*eye*
Note:	*Remember, you can reference individual elements of a matrix using the parentheses (ie* A_{12} *is obtained using A(1,2)).* *Also, you can reference entire rows or columns by using the : symbol which stands for "all rows" or "all columns" (ie the entire 2nd column of the A matrix is A(:,2))*

4. Calculate the characteristic equation for this system. Recall that the characteristic equation is given by

$$\det(\lambda I - A) = 0$$ **Equation 4**

Functions:	*none*
Note:	*Matlab requires an extra toolkit to perform symbolic manipulations so you will have to do this by hand.*

5. Write a function which solves the quadratic equation to obtain the roots of the characteristic equation. You should obtain the same results as part 2.

Functions:	*sqrt*
Note:	*See "Writing a Custom Matlab Function" section for more information regarding writing your own function.*

6. Perform a similarity transformation on the A matrix to diagonalize it and place the eigenvalues on the diagonals. Recall that a similarity transform is defined as

$$\tilde{A} = T^{-1}AT$$ **Equation 5**

where T = Matrix of eigenvectors of A

Functions:	*inv*
Note:	*Remember that Matlab treats most variables as matrices, so the * operation is actually matrix multiplication.*

7. Compute the controllability matrix. Recall that this is given by

$$P_c = \begin{pmatrix} B & AB & \ldots & A^{n-1}B \end{pmatrix}$$ **Equation 6**

Check the rank of the controllability matrix to see if the system is controllable or not.

To double check the results from the rank function, calculate the determinant of P_c and verify that it is not close to zero.

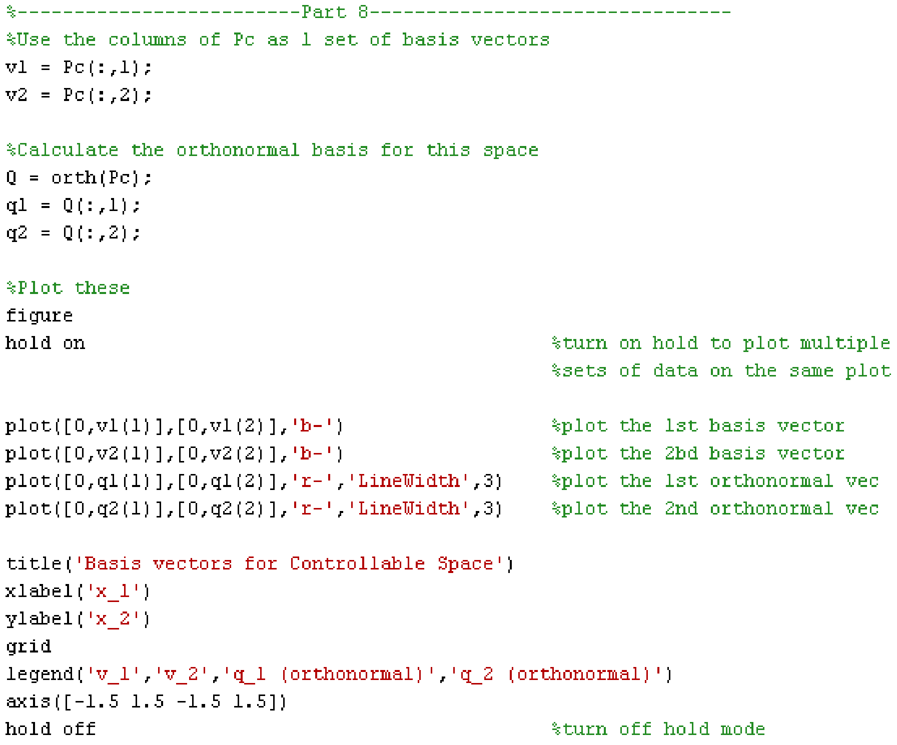
Functions:	*rank, det*
Note:	*You should see that the rank of the P_c matrix is 2*

8. Given that the P_c matrix is full rank, its columns form a basis for the controllable subspace. Use the two columns of the P_c matrix as two separate basis vectors and plot these vectors as blue lines.

Also given that the basis vectors are not unique, find a set of orthonormal basis vectors which spans the same space. Plot these as thick, red lines.

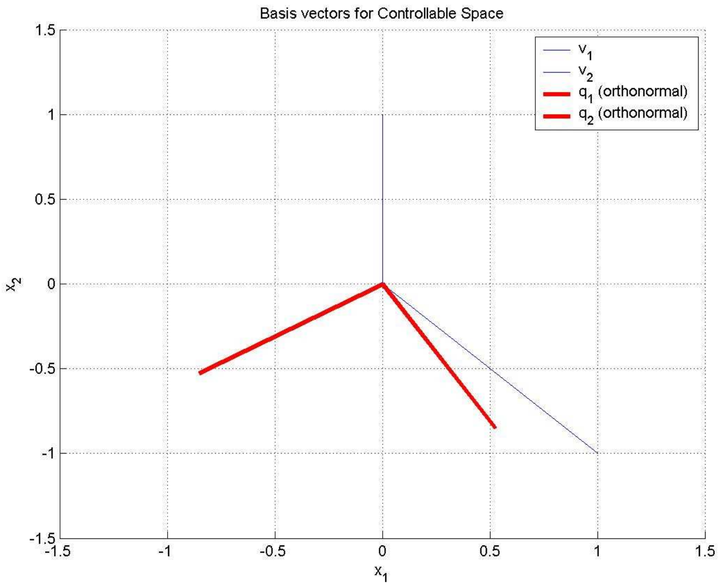
Add a title to the plot, label the axis, add a grid, and add a legend to the plot. Also make the x and y axes run from -1.5 to 1.5.

<table>
<tr><td><i>Functions:</i></td><td><i>orth, figure, plot, title, grid, xlabel, ylabel, axis, legend</i></td></tr>
<tr><td><i>Note:</i></td><td><i>When plotting more than 1 set of data, it is sometimes useful to use the “hold on” feature. A sample code is shown below</i>



```
%-------------------------Part 8---------------------------------
%Use the columns of Pc as 1 set of basis vectors
v1 = Pc(:,1);
v2 = Pc(:,2);

%Calculate the orthonormal basis for this space
Q = orth(Pc);
q1 = Q(:,1);
q2 = Q(:,2);

%Plot these
figure
hold on                                         %turn on hold to plot multiple
                                                %sets of data on the same plot

plot([0,v1(1)],[0,v1(2)],'b-')                  %plot the 1st basis vector
plot([0,v2(1)],[0,v2(2)],'b-')                  %plot the 2bd basis vector
plot([0,q1(1)],[0,q1(2)],'r-','LineWidth',3)    %plot the 1st orthonormal vec
plot([0,q2(1)],[0,q2(2)],'r-','LineWidth',3)    %plot the 2nd orthonormal vec

title('Basis vectors for Controllable Space')
xlabel('x_1')
ylabel('x_2')
grid
legend('v_1','v_2','q_1 (orthonormal)','q_2 (orthonormal)')
axis([-1.5 1.5 -1.5 1.5])
hold off                                        %turn off hold mode
```

<i>Your plot should look like this</i></td></tr>
</table>

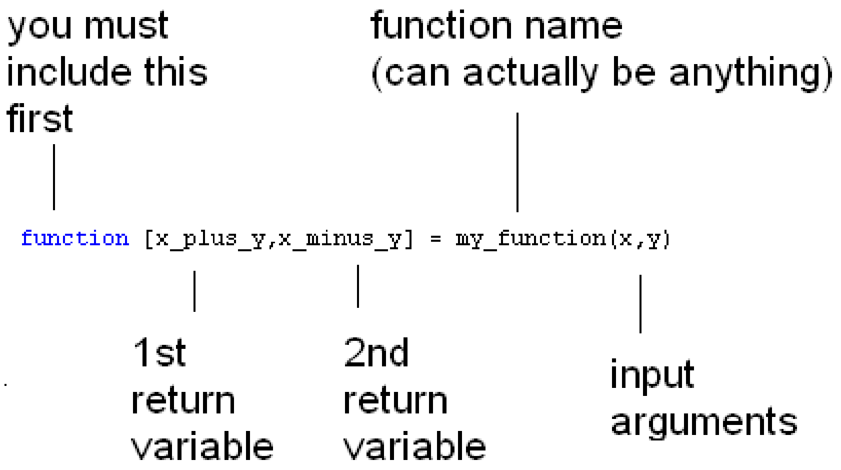
	Basis vectors for Controllable Space v_1 v_2 q_1 (orthonormal) q_2 (orthonormal) x_1 x_2 1.5 1 0.5 0 -0.5 -1 -1.5 -1.5 -1 -0.5 0 0.5 1 1.5

Writing a Custom Matlab Function

1. Start a new m-file
2. The first line of the new m-file must have the form

   function [“returned variables”] = “function name”(“input arguments”)

   For example, a function which takes two number and then returns the sum and the difference of the two numbers would have the first line similar to Figure 2.

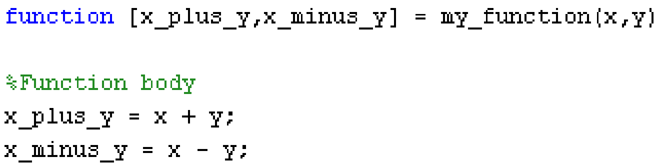


```
function [x_plus_y,x_minus_y] = my_function(x,y)
```

you must include this first
function name (can actually be anything)
1st return variable
2nd return variable
input arguments

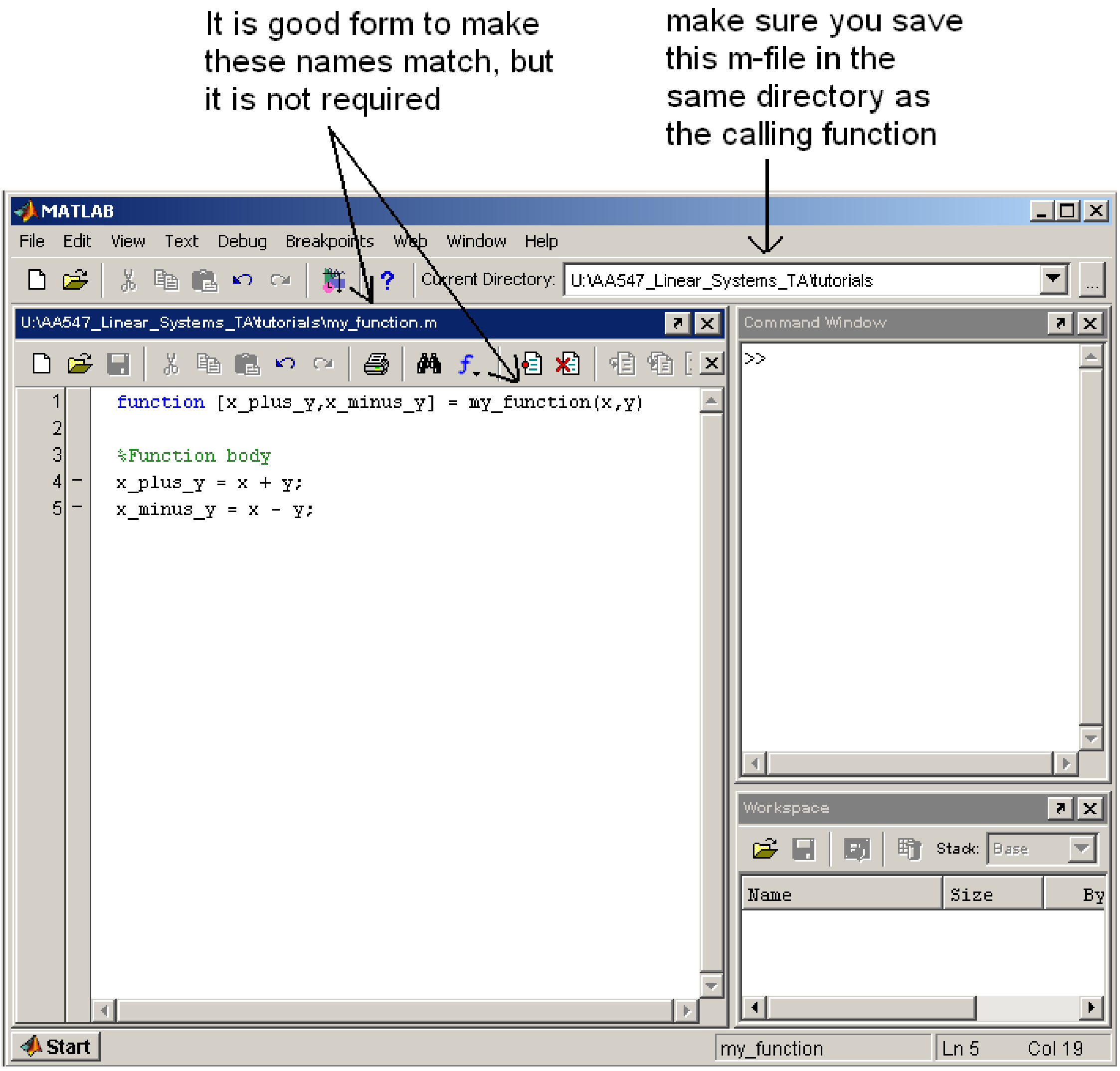
Figure 2: 1st line of MATLAB function

3. Type in the function body anywhere underneath the function header. Be sure that the returned variables are assigned somewhere in the function body.



```
function [x_plus_y,x_minus_y] = my_function(x,y)

%Function body
x_plus_y = x + y;
x_minus_y = x - y;
```

4. Save the m-file. Be sure to save this file in the same directory where the calling function is located. Also, be sure to name the file the same as the name of the function as shown in Figure 3.



It is good form to make these names match, but it is not required

make sure you save this m-file in the same directory as the calling function

MATLAB
File Edit View Text Debug Breakpoints Web Window Help
U:\AA547_Linear_Systems_TA\tutorials\my_function.m
Command Window
>>
Workspace
Stack: Base
Name
Size
By
Start
my_function
Ln 5
Col 19

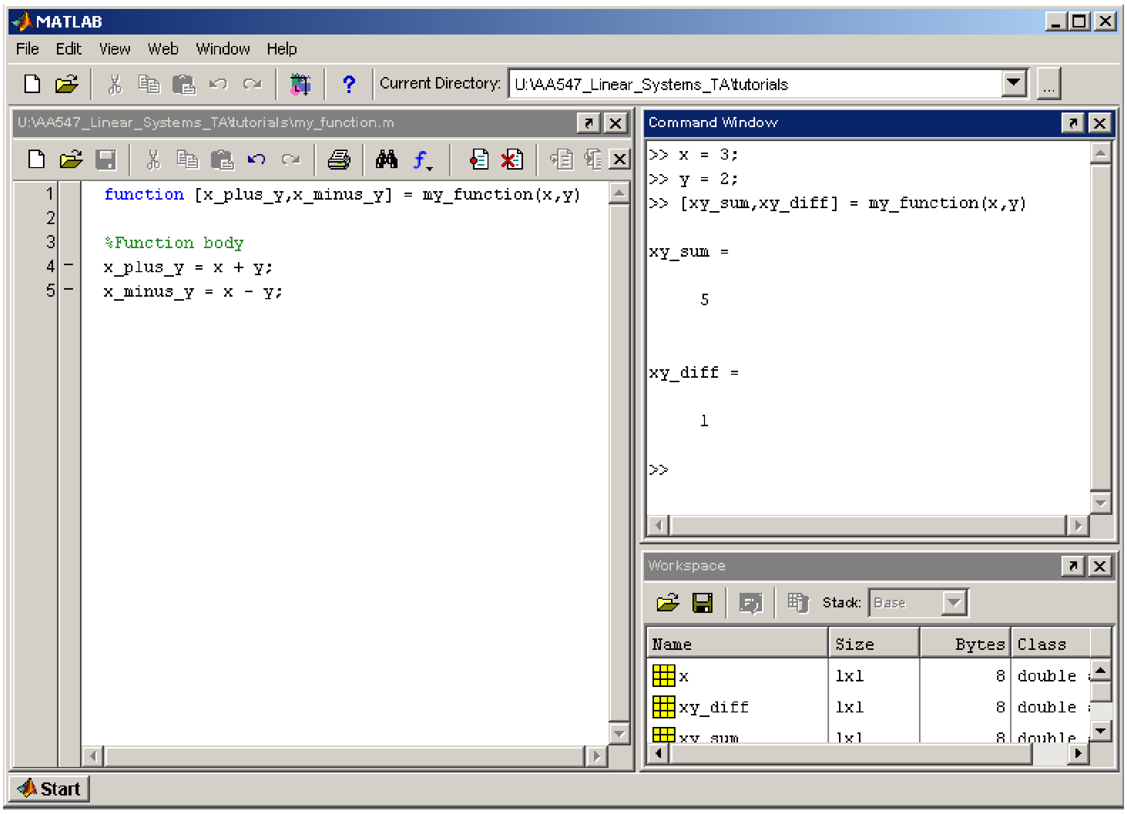
Current Directory: U:\AA547_Linear_Systems_TA\tutorials

```
function [x_plus_y,x_minus_y] = my_function(x,y)

%Function body
x_plus_y = x + y;
x_minus_y = x - y;
```

Figure 3: Saving the m-file

Note that it is good form to make the two names match, but it is not required. The name of the function in the m-file can be anything since Matlab only uses the name of the actual m-file when calling the function.

5. You can now call your custom function from another m-file in the same directory or from the Command Window (assuming that the Current Directory is the same place where you saved your custom function).



Current Directory: U:\AA547_Linear_Systems_TA\tutorials

U:\AA547_Linear_Systems_TA\tutorials\my_function.m

```
function [x_plus_y,x_minus_y] = my_function(x,y)

%Function body
x_plus_y = x + y;
x_minus_y = x - y;
```

```
>> x = 3;
>> y = 2;
>> [xy_sum,xy_diff] = my_function(x,y)

xy_sum =

     5


xy_diff =

     1

>>
```

Stack: Base

Name	Size	Bytes	Class
x	1x1	8	double
xy_diff	1x1	8	double
xy_sum	1x1	8	double

Figure 4: Calling your custom function

Version History:	09/14/04: Created:	
	11/23/05: Updated:	made this format match other to-do documents and removed references to AA547.
	12/01/05: Updated:	changed headers to match how-to template
	12/09/05: Updated:	Made changes to layout and added footer.
	09/15/09: Updated:	Added second order differential equation.
	09/25/10: Updated:	Minor changes