Vector Diagram Simulation Package: A Matlab Toolbox

Ohio Advanced EPR Laboratory
Robert McCarrick – 1/23/2012

Matlab Primer

This short Matlab primer is adapted from a lecture in Introduction to EPR Methods and Instrumentation at Miami University and gives a very basic overview to get the user started with the package. Really, the only thing needed to operate the Vector Diagram Simulation Package is the section on data entry in Matlab, but the other information might prove useful for someone just getting started and using Matlab.

Getting Help in Matlab

Fortunately, Matlab is a very popular piece of software and as such, there's no shortage of places to obtain help. I have found the following book to be extremely helpful in learning the ins and outs.

Mastering Matlab 7 ISBN: 0131430181

In addition, there is extensive online documentation from MathWorks, the creators of Matlab:

http://www.mathworks.com/access/helpdesk/help/helpdesk.html

There is a user forum where you can post questions:

http://www.mathworks.com/matlabcentral/newsreader/

If all else fails, just Google "Matlab bla bla bla". I have never been unable to figure something out that I wanted to get accomplished by going through these channels.

Introduction

Matlab is a technical computing platform that consists of the mathematical functions, programing tools and figure generators. Think of it as an operating system that runs within your main OS that has all of the math-based things that you need. The command line interface uses mostly the same commands as Linux/Unix/MacOS terminal windows.

The main Matlab Interface consists of:

Command Window – like a DOS prompt or terminal window for issuing commands.
Editor Window – a text file editor like Notepad that can be used to generate functions, programs, input files or any other text based files.
Command History Window – provides a list of previously issued commands with time stamps.
Current Directory Window– the current directory in which Matlab will save a file or retrieve a file without an entire path to the file (i.e. in this directory, Matlab will recognize test.txt without having to give c:/documents/user/desktop/test.txt).

Workspace Window – lists all of the variables, array, matrices, vectors, structures, etc. that are in the working memory and gives some info about them.

Each Window can be moved around, placed into a side tab or a top tab, and popped in or out of the Matlab control window.

Data Manipulation in Matlab

In Matlab, there are four main types of data: numbers, arrays, strings and structures.

Numbers – self explanatory, use the following method to store a number to a variable:

```
>> x = 3

x =

    3
```

Arrays – Arrays are vectors or matrices, they can be entered in the following way:

```
>> A = [1 2;3 4]

A =

    1     2
    3     4
```

Strings – Strings are pieces of data that Matlab will treat as text. Strings can be entered as follows:

```
>> samplestring = 'This is text in Matlab.'

samplestring =

This is text in Matlab.
```

Structures – Structures can be thought of as a package that contains data of various types. The format for structures is "nameofthestructure.field". You can put whatever type of data within each field that you want and can recall it by either using just the structure name (this will display all of the fields in the structure), or by referencing a particular field directly.

```
>> sample.Name = 'Rob McCarrick'

sample =

    Name: 'Rob McCarrick'
```

```
>> sample.Age = 34

sample =

    Name: 'Rob McCarrick'
     Age: 32

>> sample.favorite_matrix = [0 1/2;-1/2 0]

sample =

              Name: 'Rob McCarrick'
               Age: 32
    favorite_matrix: [2x2 double]

>> sample

sample =

              Name: 'Rob McCarrick'
               Age: 32
    favorite_matrix: [2x2 double]

>> sample.favorite_matrix

ans =

         0    0.5000
   -0.5000         0
```

Reading in Text Data

One of the things that you will often need to do is read in data. If you have a text file containing data, you can use a built in Matlab function called textread.

```
>> [xdata ydata] = textread('sample.txt','%f
%f','headerlines',1)
```

textread requires first, a filename, second the type of data in each column of the text file (in this case floating point numbers) and third, the number of headerlines containing labels and such that need to be skipped.

Plotting Data in Matlab

Once you have some data read into arrays, it would be nice to be able to plot it. Fortunately, this is really easy in Matlab using the plot function.

```
>> plot(xdata,ydata)
```

If you have more than one pieces of data you want to plot, just tack them on,

```
>> plot(xdata1,ydata1,xdata2,ydata2)
```

Now lets assume that you have some mathematical function that you want to plot (for example, lets assume that you want to generate a sine function). The first step is to generate an array that will serve as the x values for the function. The best way to do this is to use the linspace function in Matlab.

```
>> x = linspace(-10,10,201)
```

The first value is the start point, the second, the end point and the third is the number of points in between.

Now we can create an array that is the sine of each of the points in the first array.

```
>> y = sin(x)
```

Now we simply plot it.

```
>> plot(x,y)
```

Saving data in Matlab

Often, you will want to take data generated or manipulated in Matlab and save an ASCII file. This is done using the save command. The easiest way to do this is to create a single array containing all of the data and then use that for the text file.

```
[data] = [xdata,ydata]
save 'sample.txt' data  -ascii
```

Using the Editor to Issue Commands

Using the command line is fine for some things, but it can be pretty inconvenient when dealing with much larges sets of commands or scripts that you've written. The easiest way to create large sets of commands and scripts and keep a record of what you have done is to use the editor window. In the editor window, you can create a text file that can be executed in a line by line manner.

Shown here is a sample txt file (files saved in Matlab carry the ".m" extension.

```
%% A Sample M-file that Generates a Plot of the Sine
Function

clear all;  % this command clears all variable and such
```

```
%% Generate the Arrays

x = linspace(-10,10,201);   % Creates an array of x values
y = sin(x);  % creates an array of y values

%% Plot the Data

plot(x,y);  %  creates a plot of the sin function from the two
arrays

%% Save the Data
[data] = [x,y]  % creates a double array with the two x and
y vectors

save 'data.txt' data  -ascii % saves an ascii file of the two
arrays
```

You probably notice the extensive use of the "%" symbol. In Matlab, anything after a % sign in a line is ignored. This allows one to make comments that are useful when going back to something after a long period of time. In addition, recent versions of Matlab, you can use "%%" to separate that part of the file into a cell. The cell can be executed separately from the rest .

Once you have everything set and ready to execute, you can select all of the text and press the F9 key (in Windows). This will execute all of the commands. This way of issuing commands is convenient as it lets you save a file for each set of commands.