Ports from MATLAB:

The MATLAB program used for doing engineering calculations and programming provides facilities for generating plots. Getting these plots back into a Word document in a suitable manner is the issue. As with other applications used to generate graphs, it helps to make the figure as close to what you want as possible before you port it.

The most basic MATLAB command for creating a graph is "plot". It can be used to plot one or more sets of values on the vertical axis. Figure C-11 shows a plot of the capacitor Voltage and Current data from Table 1 of Chapter 4. The figure was saved as a .jpg file and inserted into this Word document. Vectors v, a, and t represented the values of Voltage, Current, and Time respectively. These were plotted using the commands:

```
>> v=[0 1.97 3.16 3.88 4.32 4.59 4.85 4.94];
>> t=[0 .1 .2 .3 .4 .5 .7 .9];
>> a=[.1 .061 .037 .022 .013 .008 .003 .001];
>> plot(t,v,'ko-',t,a,'ks--');
```

The text strings included in the plot command make both sets of data plotted in black (k), and designates round (o) or square (s) markers and solid (-) or dashed (--) lines. The graph shown does not yet have labels or grid lines. There are commands that can be used to do those things. (The MATLAB "help" documentation is very helpful; use it.) The most serious problem, however, is that both dependent variables use the same scale, and that cannot produce a satisfactory graph.

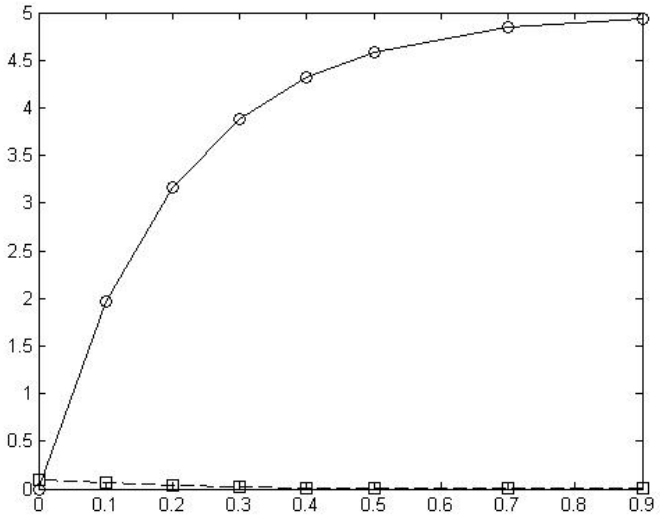


Figure C-11 MATLAB plot of two variables using the 'plot' function

There is a plot command that provides two axes, 'plotyy'. That command does not allow such easy specification of markers and lines, however. One must accept defaults or use separate (and more complicated) commands later to set the colors and other attributes. The use of the command plotyy as follows produces Figure C-12.

```
>> [AX,H1,H2]=plotyy(t,v,t,a,'plot'); % default axes are blue, green.
```

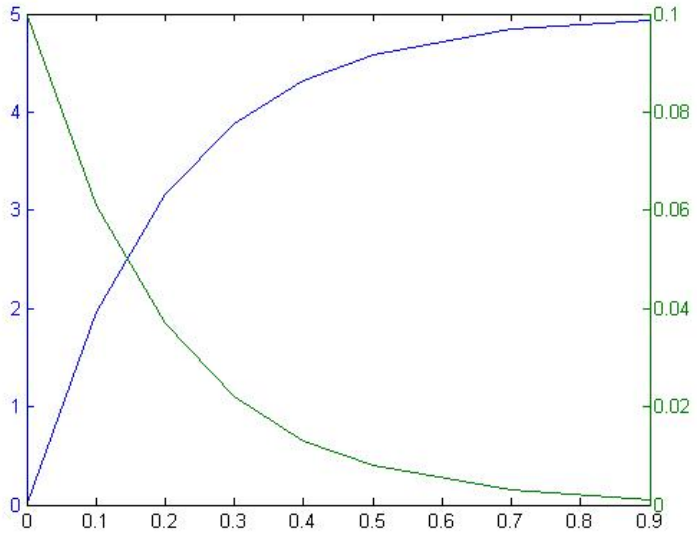


Figure C-12 MATLAB plot of two variables using 'plotyy' command

The command without the "`[AX,H1,H2]=`" would do the same thing, but returning the axis pointer AX and data pointers H1 and H2 allows subsequent commands to improve the figure. In this case, the commands executed subsequently were:

```
>> set(H1,'Color','k');
>> set(H2,'Color','k');
>> set(H1,'LineStyle','-');
>> set(H2,'LineStyle','--');
>> set(H1,'MarkerFaceColor','k');
>> set(get(AX(1),'Ylabel'),'Color','k');
>> set(get(AX(2),'Ylabel'),'Color','k');
>> grid on;
```

These produced the graph shown in Figure C-13 below. This still has problems. Markers need to be specified and the vertical axes and labels need to be colored black.

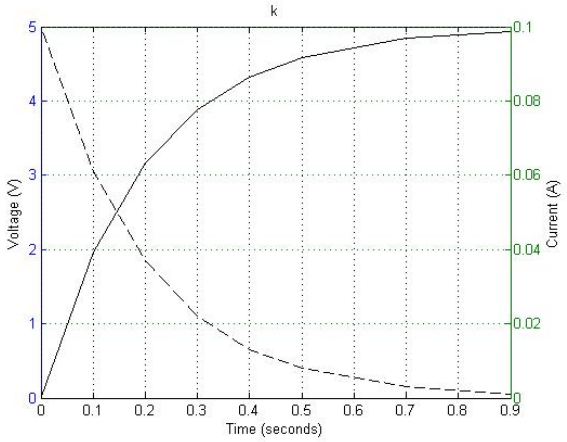


Figure C-13 MATLAB modified 'plotyy' figure

MATLAB provides a second way of editing figures, in the figure window, once they are produced. In the "Views" menu of that window, the Properties Editor can be invoked, which allows direct access to the various attributes of axes, the figure background, and various other things. Using the tools therein, the figure above was edited to the form shown in Figure C-14 (on next page).

It is also possible to build up a plot in MATLAB line by line, adding and editing the properties of axes as you go. See the MATLAB documentation of how to do plots with multiple axes. That is probably the best way to do more complex graphs, such as one having multiple variables, with some being linear and some being logarithmic. Simpler log plots can use the 'loglog', 'semilogx' or 'semilogy' commands. MATLAB also supports three dimensional plots. These topics are beyond the scope of what this manual can address. Our main concern here is how to get those figures back into a Word document. Figured C-11 to C-13 were ported by saving them as .jpg files within MATLAB, then inserting them into the Word document as pixelated graphics. That does not necessarily produce the best graphic, but it is simple, and adequate in many cases.

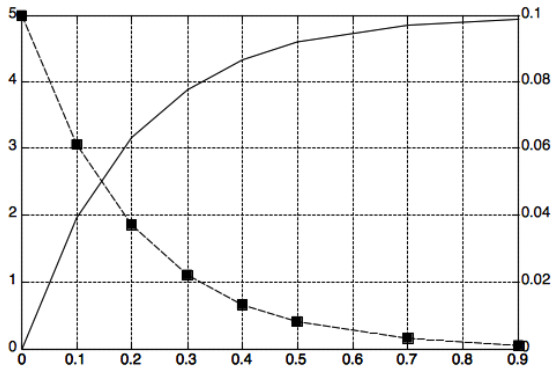


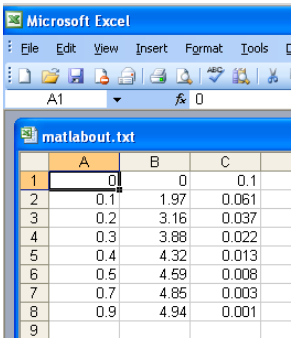
Figure C-14 MATLAB graph after editing in the figure editing window.

Figure C-14 was saved from MATLAB as an ".emf" fomat file, which stands for "Enhanced Metafile". The file opened in LibreOffice and from there was pasted into Word, but remains pixelated rather than a line graphic. (It may be that one of the other formats that preserve line features, .ai or .eps, can be used. Or perhaps more recent updates to MATLAB provide a save format that can be transitioned to Word as line art. What you don't want to do is transfer by screen shot; that preserves a grey background.)

Another strategy for getting good graphics is to have MATLAB print out the data to be graphed into a comma (or tab) delimited text file that can then be opened in Excel. Then Excel can be used to produce a satisfactory graph which can be pasted into Word as line art, avoiding the problems with pixelated graphics. For example, in the MATLAB context of the capacitor data above, one could include in a script (or type to the command window) the following commands:

```
>> fp=fopen('capdataout.txt','w');
>> for i=1:8, fprintf(fp,'%f, %f, %f\n',t(i),v(i),a(i)),end;
>> fclose(fp);
```

This produces a comma delimited file. When opening the file with Excel, it is necessary to specify that it is comma delimited text. Once opened, it appears as seen in Figure C-15 below. (The columns were reformatted to fixed point style.) Now Excel, with all of its graphing tools, can be used to produce the same graph seen earlier as Figure 5-1 of Chapter 5, which can be copied in as line art.



	A	B	C
1	0	0	0.1
2	0.1	1.97	0.061
3	0.2	3.16	0.037
4	0.3	3.88	0.022
5	0.4	4.32	0.013
6	0.5	4.59	0.008
7	0.7	4.85	0.003
8	0.9	4.94	0.001
9			

Figure C-15 Comma delimited text opened by Excel