Inserting Plots and Figures into your \LaTeX \document

Josh Walawender\textsuperscript{1}
January 21, 2001

1. Plotting in IDL

In this class you will be using IDL and \LaTeX\ extensively in your lab writeups. For the first lab it was sufficient to simply have the basics of making plots and inserting them into your writeups, however you can make your writeups and plots much nicer by delving a little more deeply into the capabilities of IDL and \LaTeX\ . This handout will show you a bit more about both of these programs. Of course, the only way to become a true expert in either program is to experiment yourself.

I should mention that all the information in this handout is available from other sources\textsuperscript{2}, I have simply summarized the commands and techniques which I have found useful in the past.

1.1. The “plot” Command

First, we’ll start out by examining the plot command in IDL. plot simply creates a plot of the data which are input. We can call plot with several keywords and optional inputs and these are what we’ll explore here. Fig. 1 is the plot generated by the following command: (Note that the $ symbol tells IDL that the command is continuing on the next line)\textsuperscript{3}

\begin{verbatim}
IDL> plot, xdata, ydata, psym=7, $ xrange=[0,10], yrange=[0,20], /xstyle, /ystyle, $ xtitle='x-axis label (units)', $ ytitle='y-axis label (units)', $ title='Title for entire plot'
\end{verbatim}

Now we can look at the properties of fig. 1 and see what commands created them. First we should note that by prepending a / onto an IDL keyword we are setting that keyword’s value to one – many keywords simply look to see if it has a value or not, a sort of true/false setting.

\texttt{xdata} and \texttt{ydata}: the data arrays which contain the data to be plotted on the x and y axis.

\textsuperscript{1}with small modifications by Carl Heiles 21 Jan 2001

\textsuperscript{2}Most notably the IDL on-line help and \LaTeX\ by Leslie Lamport (a well worn copy is floating around the 705 Lab somewhere).

\textsuperscript{3}Before we go into the details of the keywords in the following command I want to make a quick note about IDL. You can use shortcuts for some keywords in IDL, for example /xstyle can be replaced with /xs and \texttt{psym} with \texttt{ps}. You can play around to find these timesavers.
**psym=4**: sets the data symbol. By setting this to 7 we select the xs to represent the data. You can select other numbers to use other symbols (i.e. 3=dots, 4=squares, 5=triangles, 6=squares, 7=xs, 10=histogram style, etc.). When psym is set to a negative number, then the data points are connected by lines. A **stylistic note:** it is nice to represent your data by some sort of symbol so that the person looking at the graph can tell how often you sampled your data — lines are many times reserved to represent theoretical models or fits to the data.

**(xy)range:** setting these equal to a two element array causes the plot ranges to be set to the values in those arrays. The first element in the array is the minimum and the second is the maximum.

**(xy)style:** these override the plot command’s automatic setting of the x and y ranges. IDL seems to always set the ranges too high so that there is a lot of space surrounding your plot. Setting these will set the ranges so that the maximum and minimum range of the data is also the maximum and minimum range of the plot. These also will force IDL to use the exact values that you set in (xy)range.

**(xy)title:** set this to a string which becomes the title for that axis or for the plot itself. A **stylistic note:** always indicate the units on your axis.
1.2. The “oplot” command

We can also use the oplot command to overlay other plots on top of ones we have already made. For example, if we wanted to place a dashed line representing the theoretical model through our data (see fig. 2) we would use the following command:

IDL> oplot, xdata, ytheory, linestyle=2

Fig. 2.— An example plot with an additional plot overlayed.

**linestyle**: sets the linestyle. By setting this to 2 we selected a dashed line. Other numbers represent, for example a dotted line or a dash-dot line, etc. (linestyle also works in plot)

**ytheory**: the array of theoretical values calculated for y.

1.3. Plotting Error Bars

We can add also error bars to our plot using the oploterr command. In our example we can plot the y-axis uncertainty (see fig. 3) using the following command:

IDL> oploterr, xdata, ydata, yerrors

**yerrors**: the array of errors on the y measurements.

There are other ways to plot error bars. See, for example, errplot and ploterr.
Fig. 3.— An example plot with the error bars plotted.

1.4. The “xyouts” command and Plot Legends

You can add legends to your plots by using oplot and xyouts. The xyouts command places the contents of a string into your plot at the given coordinates. For example, we could add a legend to the plot that we’ve been working on. We’d use the following commands to make figure 4.

IDL> oplot, [6.8], [6.1], ps=7
IDL> xyouts, 7.0, 6.0, ’data points’
IDL> oplot, [6.5,6.9],[5.5,5.5], lines=2
IDL> xyouts, 7.0, 5.4, ’theory’
IDL> oplot, [6.8,6.8],[4.6,5.1]
IDL> xyouts, 7.0, 4.8, ’error bars’

The first line oplots the x symbol, the second line places the text ’data points’ next to it at position (x,y)=(7.6), the third line plots the dashed line, and so on. I chose the locations for the legend elements by simply playing around with the numbers until they looked good.

xyouts, 7.0, 6.0, ’data points’: the first number is the x location to begin the string and the second number is the y location to begin the string.
1.5. The !P.MULTI Variable

The !P.MULTI variable in IDL allows you to place multiple plots into the same window. Because !P.MULTI is an IDL system variable it is prepended by a ! - this means that each time you start IDL it is set to a specific value, just like !pi, !tdor, and !radeg.

To place multiple plots into one window, set !P.MULTI equal to an array with the first element zero (if you want to see what this variable does see the IDL online help), the second element equal to the number of plots across the window and the third element equal to the number of plots down the window. Then give your plotting commands. For example, if I wanted to place six plots in one window with two across and three down I'd do the following:

IDL> !P.MULTI=[0,2,3]
IDL> plot, x1, y1
IDL> plot, x2, y2
IDL> plot, x3, y3
IDL> plot, x4, y4
IDL> plot, x5, y5
IDL> plot, x6, y6
IDL> !P.MULTI=[0,1,1]

Always remember to set !P.MULTI back to [0,1,1] when you're done otherwise your next
window will have places for six plots. You can just type 
\texttt{!P.MULTI=0}.

2. Turning Your Plots into Postscript Files

2.1. Two procedures we provide: "openplotps.pro" and "closeps.pro"

We provide these to allow you to easily change sizes and change between portrait and landscape mode. To use them:

\texttt{IDL> openplotps}
\texttt{IDL> [your plotting commands go here]}
\texttt{IDL> closeps}

\texttt{openplotps} opens the plot's postscript file and will prompt you for the name. \texttt{closeps} closes the postscript file; if you forget this, the whole procedure won't work. Your plotting commands are the same ones you use to generate the plot on the screen; to make the above plots, this whole sequence would look like

\texttt{IDL> openplotps}
\texttt{IDL> plot, xdata, ydata, psym=7, $}
\texttt{xrange=[0,10], yrange=[0,20], /xstyle, /ystyle, $}
\texttt{xtitle='x-axis label (units)\', ytitle='y-axis label (units)\', $}
\texttt{title='Title for entire plot\',}
\texttt{IDL> oplot, xdata, ytheory, linestyle=2}
\texttt{IDL> oploterr, xdata, ydata, yerrors}
\texttt{IDL> closeps}

\texttt{openplotps} has some keywords that you can set. These include the $x$ and $y$ sizes and whether to use landscape orientation; you can also specify the postscript file name with a keyword. To see more information about \texttt{openplotps} (or any properly-documented procedure not provided by IDL, type the following at the IDL prompt:
\texttt{IDL> doc\_library, 'openplotps'}

2.2. The "set\_plot" and "device" Commands

These are the IDL-provided versions of the above; if you use them and want to do anything other than the defaults, you have to write some code yourself.

3. Placing Plots Into Your \LaTeX{} Documents

You should already be able to place plots into your \LaTeX{} documents, but I am going to present a slightly different set of commands that will do the same thing. I like this set because
they are a bit less cryptic and they allow you to place two figures side-by-side in your document. First, I'll show you the commands then I'll point out some of the highlights.

\begin{figure}![ht]
\begin{center}
\epsscale{.6}
\plotone{ex3.ps}
\end{center}
\caption{An example plot with the error bars plotted.}\label{figure}
\end{figure}

Now I'll break down the set of commands and show you what each one does:

\begin{itemize}
  \item \texttt{\begin{figure}![ht]}: this tells \LaTeX{} that you are beginning a figure. The \texttt{![ht]} tells \LaTeX{} that you want the figure placed at this point in your document. \LaTeX{} can't always place the figures exactly where you want them, but it tries to get them close.
  \item \texttt{\begin{center}}: this line centers whatever follows it.
  \item \texttt{\epsscale{.6}}: this line controls the overall scaling of the figure. Smaller numbers shrink the image and larger ones expand it, maintaining the aspect ratio. Scaling the image can be useful in getting \LaTeX{} to place figures where you want them. It is easier to place them in the correct location if they are smaller. This actually works very well, I've fixed documents in which the figures were scattered all over the place simply by reducing them all by 10%.
  \item \texttt{\plotone{ex3.ps}}: this indicates the file that you want to be placed into the document.
  \item \texttt{\end{center}}: ends the centering.
  \item \texttt{\caption{An example plot with the error bars plotted.}}\texttt{\label{figure}}: the first part of this command places a caption at the bottom of the figure. The second part of the command assigns a label to the figure, explained in the next section.
  \item \texttt{\end{figure}}: ends the figure and returns you to normal text mode.
\end{itemize}

\section{Labels in \LaTeX{}}

When you are writing your lab reports you will want to direct the reader to a specific figure, equation, or section. You do this by referring to figures and tables \textit{by number}; for example, "See Figure 2". \LaTeX{} numbers all figures sequentially as it encounters the \texttt{\begin{figure}} command, so this seems easy.

But it's not so easy. During the process of writing you might not know which figure is going to be Figure 3: you might interchange two figures during the process of writing, or add another; then, referring to figures by number, you would have to change the numbers each time you changed the ordering of the figures.

There's an easy way around this: in your \texttt{.tex} file, refer to the figures by \textit{reference}. This means
each figure gets a reference name, given by the \texttt{\textbackslash label\{xxx\}} command (which, as an example, gives the figure the reference name \texttt{xxx}.) Now \textit{in your tex file} you refer to that figure by reference, for example you type \texttt{See Figure \textbackslash ref\{xxx\}}; then, \textit{on the final printed page}, it puts “See Figure 2” or “See Figure 3” or whatever, depending on what sequence the number the figure actually has.

The \texttt{\textbackslash label\{ \}} command works for equations, tables, sections, and subsections, too.