Physics 263: MATLAB Cheatsheet III

This is the third collection of basic information and techniques for using MATLAB.

1. Symbolic Partial Derivatives

This is a supplement to the discussion from “MATLAB Cheatsheet I”. Any symbol other than the differentiation variable in a function being differentiated using \texttt{diff} is assumed to be constant. So, for example, if we want to take

\[
\frac{\partial f(x,y)}{\partial x} \quad \text{where} \quad f(x,y) = 2xy + x^2 + y^3,
\]

\[
\gg \text{syms } x \ y
\]
\[
\gg f = 2\times y + x^2 + y^3
\]
\[
\gg \text{diff}(f,x)
\]

where we have assigned the symbolic expression for our function to \texttt{f}. This enables us to easily take other derivatives. For example, we can find in turn: \( f_y, f_{xx}, \) and \( f_{xy} \) using

\[
\gg \text{diff}(f,y)
\]
\[
\gg \text{diff}(f,x,2) \quad \% \text{ the "2" means 2nd derivative}
\]
\[
\gg \text{diff(diff(f,y),x)} \quad \% \text{ a mixed partial derivative}
\]

and so on. Sometimes after taking a derivative the answer will be hard to read. A more readable form is obtained using \texttt{pretty}. For example,

\[
\gg f = x/(x^2+y^2);
\]
\[
\gg \text{diff}(f,y)
\]

\[
\text{ans } =
\]
\[
-2/(x^2+y^2)^2*x*y
\]

\[
\gg \text{pretty(ans)}
\]

\[
\begin{array}{c}
x y \\
-2 \quad \text{-----------}
\end{array}
\]
\[
\begin{array}{cccc}
2 & 2 & 2 \\
(x & + & y)
\end{array}
\]

2. Three-Dimensional and Contour Plots I

There are many ways to make three-dimensional plots in MATLAB, some of which were illustrated in the \texttt{eqheat.m} example from 1094 Session 1. Here we’ll give a sampling of one type of 3D plot and one type of contour plot.
a. Making a 3D surface or mesh plot. Suppose we want to plot the function \( f(x, y) = x^3 - y^3 - 2xy + 2 \) with \( x \) and \( y \) both ranging from \(-1\) to \(+1\). We make the \( x-y \) grid using the function meshgrid:

\[
\begin{align*}
\text{>> } & [X, Y] = \text{meshgrid(linspace(-1,1,20),linspace(-1,1,20));} \\
\text{with 20 points in each direction or} \\
\text{>> } & [X, Y] = \text{meshgrid(-1:.1:1, -1:.1:1);}
\end{align*}
\]

to have a grid spacing of 0.1. We want to use a reasonable number of points, since the number of points will affect the final appearance of the grid surface, but we can always change this later.

We calculate \( f \) for each of the \((x, y)\) values in \([X,Y]\) with \([\text{we'll use } Z \text{ for } f(x, y)]\):

\[
\text{>> } Z = X.^3 - Y.^3 - 2*X.*Y + 2; \ 	ext{note the use of } .^{'},s
\]

Finally, we plot it and add a colorbar (a translation between colors and the value of \( Z \)):

\[
\text{>> surf(X,Y,Z) \ 	ext{ also try mesh(X,Y,Z)} \\
\text{>> colorbar}
\]

b. Making a filled contour plot. To make a contour plot of the same \( f(x, y) \) over the same range, we set up \( X \), \( Y \), and \( Z \) the same way. If we want 10 filled contour levels, use:

\[
\text{>> contourf(X,Y,Z,10)}
\]

For a similar contour plot without the color filling, use contour instead of contourf.

3. Finding the Slope of a Straight Line on a Log-Log Plot

On a MATLAB log-log plot (where the logarithms are base 10 instead of natural logarithms), a straight line plot with slope \( a \) and intercept \( b \), namely

\[
(\log_{10} y) = a(\log_{10} x) + b,
\]

means that \( y \) has the power law behavior (make sure you can derive this!)

\[
y = 10^b x^a.
\]

If you’ve generated a set of points that seem to lie on a straight line on a log-log plot and you want to fit a line to them and determine the slope, we can use polyfit. Suppose the data we’ve generated is in the vectors \( x \) and \( y \). Then

\[
\text{>> my_fit = polyfit(log10(x),log10(y),1)} \ 	ext{ % the 1 means a linear fit} \\
\text{>> a = my_fit(1)} \ 	ext{ % the first element of my_fit is the slope} \\
\text{>> b = my_fit(2)} \ 	ext{ % the second element of my_fit is the intercept}
\]

will give you the slope \( a \) and intercept \( b \). We can add the fit line to the original plot with

\[
\text{>> fit_line = 10^b * x.^a; \ % reconstruct the power law} \\
\text{>> loglog(x,y,x,fit_line) \ % plot both on the same log-log plot}
\]

Note: Be sure to include in \( x \) and \( y \) only the region of the plot you want to fit.