ECE-210-B Homework #3

Plotting

Cat Van West, Spring 2024

Contents

Matlab without plots is boring. Matlab with *bad* plots is worse. Matlab with *good* plots is… sublime. Since Fred's been complaining about people's general plotting skills, I decided to spend some time on it explicitly this year. Plotting well is as much an art as a science, so I'll leave you with a few guidelines and loose requirements. Get creative!

Here's what each plot should have (even if not stated):

- a title (for subplots too!)
- axis labels (if none given, make some up label as x/y/z, or, if you're really brave, leave blank)
- a legend, if multiple datasets are overlaid on the same plot
- sane axis ticks and tick labels (set yourself if matlab's autogeneration is not sufficient)
- a sane color scheme (usually matlab does decently well here)
- a sane aspect ratio (default is usually fine but occasionally terrible)

These guidelines apply to subsequent assignments as well. Make me pretty plots. ;)

1 The Assignment

The following instructions are negotiable; if you don't follow them exactly but make a beautiful and readable plot anyway, I'm willing to be persuaded. I don't claim to know what the Fontaine wishes.

Two Dimensions Create the following plots. Don't forget to call `figure` before each to ensure you don't overwrite previous plots.

1. Sample a sine wave at 50 points evenly spaced between zero and 2π. Use `plot` to plot this wave in green.
2. Sample a cosine wave at the same 50 points and create a plot with both waves superimposed (either use `hold` or pass MATLABplot a matrix – it can be done in one call!). Use `legend` to say which wave is which, `xlim` & `ylim` to set (reasonable) axis limits, and `xticks` & `xticklabels` to place labeled ticks at 0, $\pi/2$, π, $3\pi/2$, & 2π. Give the plot a terrible title (bonus points if you make me laugh).
3. Re-plot the sine and cosine waves in two different subplots using `stem` (to indicate they're discrete-time signals). Randomly change the amplitude of one of them (see `rand` or `randn` to generate random numbers). Compute the amplitude ratio of the two waves in decibels and use `sprintf` to embed that information in the title of one of the subplots.

   Set up the axes, ticks, titles, legends, colors, etc. of these plots as you see fit. Turn the grid on for at least one of them.
4. Find a color image on the internet and download it. Use `imread` to read it into MATLAB, and show it with `imshow`. Cool.
5. Examine the data that `imread` returned. Either use `whos` or the workspace inspector – just get an idea of what type (integer/floating point) and what size/shape the data is. Using this info, create *another* image (either mathematically or by modifying the returned data) and plot that using `imshow`. Check out the docs for these functions for more info on how they work.

Three Dimensions Create the following plots. Basically the same instructions, but now we have an extra dimension to deal with!

1. Plot the surface $z = \exp(-x^2 - y^2)$ over the square $0 \leqslant x \leqslant 1,\ 0 \leqslant y \leqslant 1$. Create this plot three times, in three different subplots, using `plot3` (a strange choice here), `scatter3`, `surf`, and `mesh`. Adapt the code in the lecture notes for this – the surface plotted there is very similar.
2. Plot the parametric conical helix $x = t\cos 27t$, $y = t\sin 27t$, $z = t$, for t from 0 to 1. Use `pbaspect` and/or `daspect` to set the aspect ratio of the plot to 1:1:1, so that the cone looks nice. Set `grid on` to give some background. Use `view` to set the camera position to a nice view of the cone.
3. Read the code in the lecture notes for creating the MATLAB logo and run it! This won't be collected (though you can include it if you wish) – it's just for your amusement. It also uses a different plotting approach (object-oriented rather than imperative), and as such provides another potentially useful perspective.

A Creating a General Plot

This is mostly a catalog of how *I* do it, and as such should be taken with a grain of salt.

I begin plots by calling `figure`. This creates a new plotting window, ensuring I don't overwrite a plot I already made. I then set up subplots, if necessary – I treat each call to `subplot` like the original `figure` call, so that nothing gets overwritten in the subplots either. If I need to plot overlapping data, I call `hold on`, which prevents subsequent calls to plotting functions from overwriting the plot window completely.

After that comes data – this roughly establishes what sort of axis scaling I'll need and lets me see where the plot needs work. `plot`, `plot3`, `imshow`, `hist`, etc. – you get the idea. Once the data's in place, I'll title the plot, label the axes, and add whatever legend I need, so I remember what the data on the plot represents.

Finally, I'll fiddle with the axis ticks, axis tick labels, plot aspect ratio, camera positioning, background grid, and so on – all those little details that bring a plot from workable to truly readable. At the very end, if necessary, I'll export the plot to an image using `exportgraphics` or something similar.

Not all of these steps are necessary for every plot, but if you want to produce something polished, you'll need many of them. Plus Fred will be happy. Also your research partners and whoever's reading your paper.

B Other Things To Check Out

- Mathworks has a useful intro to plotting – nothing fancy, but covers the basics – at mathworks.com/help/matlab/learn_matlab/plots.html.
- The code in the lecture notes for the MATLAB logo was taken from Mathworks' docs: mathworks.com/help/matlab/visualize/creating-the-matlab-logo.html. I recommend looking at this page – it gives a beautiful live-script walkthrough of the process of creating the plot.
- The `sprintf` docs are at mathworks.com/help/matlab/ref/sprintf.html. The format is very similar to C's `printf` (as you probably guessed).