Lecture 4: Plotting

Math 98, Fall 2023

Agenda

- Plotting Basics (`MOOCplotting.m`)
  - Plotting Vectors
  - Plotting Multiple Curves
  - Multiple Figures and `hold`
  - Title, Axis Labels, and Legend
  - Axis Limits
- Exercise: `cosineplotting.m`
- Advanced Plotting (`plotmisc.m`)
- Exercise: `heart.m`
- 3-D Plots
- Exercise: `SinCosPlot.m`
- Scatter Plots

Plotting: Plotting Vectors

You learned how to plot a single vector:

```
>> a = (-10:10).^2;
>> plot(a)
```

where the values of a will be the "y-values" of your graph and the indices of the elements of a will be the "x-values".

You also learned how to plot an array of x and y values together.

```
>> t = -10:10; b = t.^2;
>> plot(t, b);
```

Where the two vectors must be the same length. (Try and see the resulting error message if they aren't).

Plotting: Plotting Multiple Curves

Then you can plot two curves on the same figure:

```
>> x1 = 0:0.1:2*pi; y1 = sin(x1);
>> x2 = pi/2:0.1:3*pi; y2 = cos(x2);
>> plot(x1, y1, x2, y2);
```

You can also change the style and color of the lines:

```
>> plot(x1, y1, 'r', x2, y2, 'k:')
```

You can see all the plot options by typing `help plot` in the Command Window.

Plotting: Multiple Figures and `hold`

You can also specify the figure you want to the subsequent commands to plot on:

```
>> figure(1)
>> plot(t, b, 'm--O')
```

And plot multiple series on the same plot with `hold on`

```
>> figure(2)
>> plot(x1, y1, 'r'); hold on;
>> plot(x2, y2, 'k:')
```

Note that if you then type in `hold off`

```
>> hold off;
>> plot(x2, sin(x2))
```

Everything that was on that figure is overwritten.

Plotting: Title, Axis Labels, and Legend

You also learned how to add a title to the plot

```
>> figure(3)
>> plot(x1, y1, x2, y2);
>> title('A Sine Plot and a Cosine Plot')
```

and x-y labels and a legend:

```
>> xlabel('The argument of sine and cosine')
>> ylabel('The value of the sine or cosine')
>> legend('sine', 'cosine')
```

MATLAB automatically associates the legend labels to the series plotted in order. 'sine' gets associated with (x1, y1) and 'cosine' with (x2, y2).

The code from the MOOC video is available as `MOOCplotting.m`

Plotting: Axis Limits

You can also change the axis of the plots:

```
>> axis([-2, 12, -1.5, 15])
```

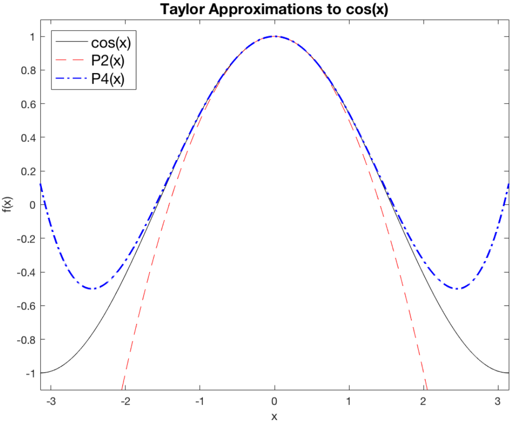
To remember the arguments of `axis`, you can type in `help axis` to the Command Window to see it works like `axis([XMIN XMAX YMIN YMAX])`.

I also like to change the x-limits and y-limits individually

```
>> xlim([-2, 12]); ylim([-1.5, 15]);
```

Exercise: `cosineplotting.m`

Say we want a visual comparison of $\cos(x)$ with its Taylor series approximations. Your exercise is to create the following plot:



STOP HERE AND TRY IT!

Solution (I): `cosineplotting.m`

Say we want a visual comparison of $\cos(x)$ with its Taylor series approximations. We can start out with

```
>> xs = -5:5;
>> plot(xs,cos(xs))
```

This doesn't look great because Matlab only plotted the 11 points $[-5, -4, \ldots, 4, 5]$ and then used linear interpolation. Try making the divisons finer to get a smoother curve:

```
>> xs = -5:0.01:5;
>> plot(xs,cos(xs))
```

MATLAB only knows how to plot straight lines!

Solution (II): cosineplotting.m

One way to plot multiple lines together is to use hold on.

```
>> hold on
>> f = @(x)(1-x.^2/2);
>> plot(xs,f(xs));
>> g = @(x)(1-x.^2/2 + x.^4/24);
>> plot(xs,g(xs));
```

Not bad, but we probably want to zoom in a little farther.

```
>> ylim([-1.1, 1.1]);
>> xlim([-pi, pi]);
```

Solution (III): cosineplotting.m

Finally, we add a title, labels, and a legend.

```
>> xlabel('x');
>> ylabel('f(x)');
>> legend('cos(x)','P2(x)','P4(x)','location','northwest');
>> title('Taylor Approximations to cos(x)', 'FontSize',14);
```

A few other commands can alter the line width, color, and style. We can use `cla` (Clear Axis) to reset the axes or `clf` (Clear Figure) to clear the entire figure.

```
>> plot(xs, cos(xs), 'k'); hold on
>> plot(xs, f(xs), 'r--');
>> plot(xs, g(xs), 'b-.','LineWidth',1);
```

and we're done! The script that generates this is also on my homepage.

Advanced Plotting (`plotmisc.m`)

- If you want multiple figures open at once, `figure` creates a new figure.
  - `figure(10)` would open up Figure 10.
- `close` closes the current figure. `close all` closes all figures.
- `loglog(xs,ys)` plots on a log-log scale.
- `semilogx(xs,ys)` and `semilogy(xs,ys)` make linear-logarithmic plots.
- `scatter(xs,ys)` makes a scatter plot instead of a line plot.
- `subplot(m,n,p)` is for putting multiple plots in a single figure. Adds a plot to the p-th position an $m \times n$ grid (counting across each row).
- `set(gcf,'position',[a b L W])` changes the location and size of the figure window. It places the lower left corner of an L-by-W figure window at (a, b).

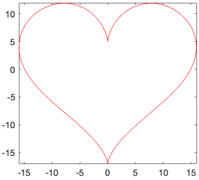
Exercise: heart.m

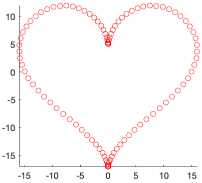
Plot the parametric curve given by the relations

$$x = 16\sin^3(\theta)$$
$$y = 13\cos(\theta) - 5\cos(2\theta) - 2\cos(3\theta) - \cos(4\theta)$$

as θ ranges from 0 to 2π. (Remember `linspace`?)
Create a single plot with two subplots. Solid line on the left and scatter plot on the right.

- What do the commands `axis equal` and `axis tight` do?





STOP HERE AND TRY IT!

3-D Plots

- `plot3(x,y,z)` plots lines in 3-D space.
  Example: A helix.

```
>> t = 0:(pi/50):10*pi;
>> plot3(sin(t),cos(t),t);
```

- `surf(X,Y,Z)` and `mesh(X,Y,Z)` make a solid surface and a mesh, respectively, in 3-D.
- There are a number of ways to control the camera position. `view(AZ,EL)` controls the rotation around the z-axis and the vertical elevation. `view(3)` is the default 3-D view and `view(2)` = `view(0,90)` gives a direct overhead view.
- Another option is the pair of commands `campos` and `camtarget`, setting the "camera" position and target.

Exercise: `SinCosPlot.m`

Make a 3-D plot of the function $f(x, y) = 2\sin(x)\cos(y)$ on the interval $[0, 2\pi] \times [0, 2\pi]$.

Scatter Plots

Instead of `plot` or `plot3`, try `scatter` and `scatter3`.

```
>> x = -5:0.1:5;
>> subplot(1, 2, 1)
>> plot(x, sin(x))
>> subplot(1, 2, 2)
>> scatter(x, sin(x))
```