L15: Data visualization

Plotting with Matlab

Lucas A. J. Bastien

E7 Spring 2017, University of California at Berkeley

February 22, 2017

Version: release

Announcements

Lab 05 is due on February 24 at 12 pm (noon)

Today: (Chapter 11)

- Review of last Friday's written feedback
- How to plot data in Matlab?
- How to properly format a plot in Matlab?

Friday: (Chapter 7)

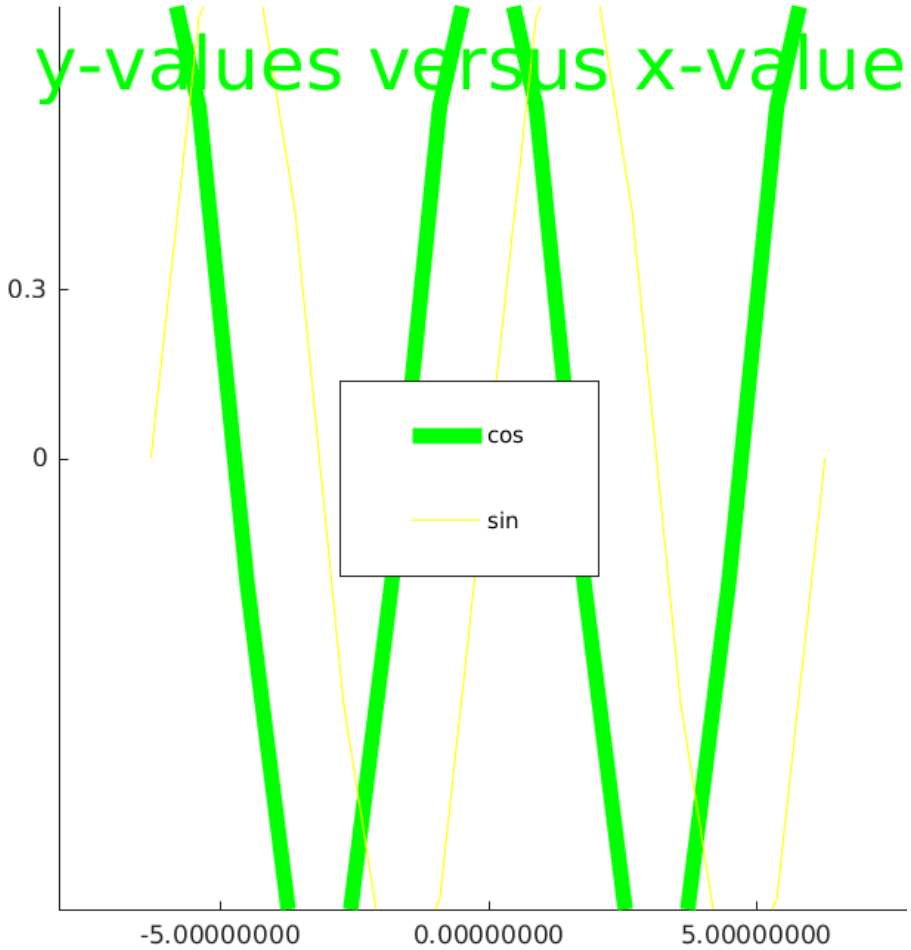
- Complexity of algorithms

Midterm:

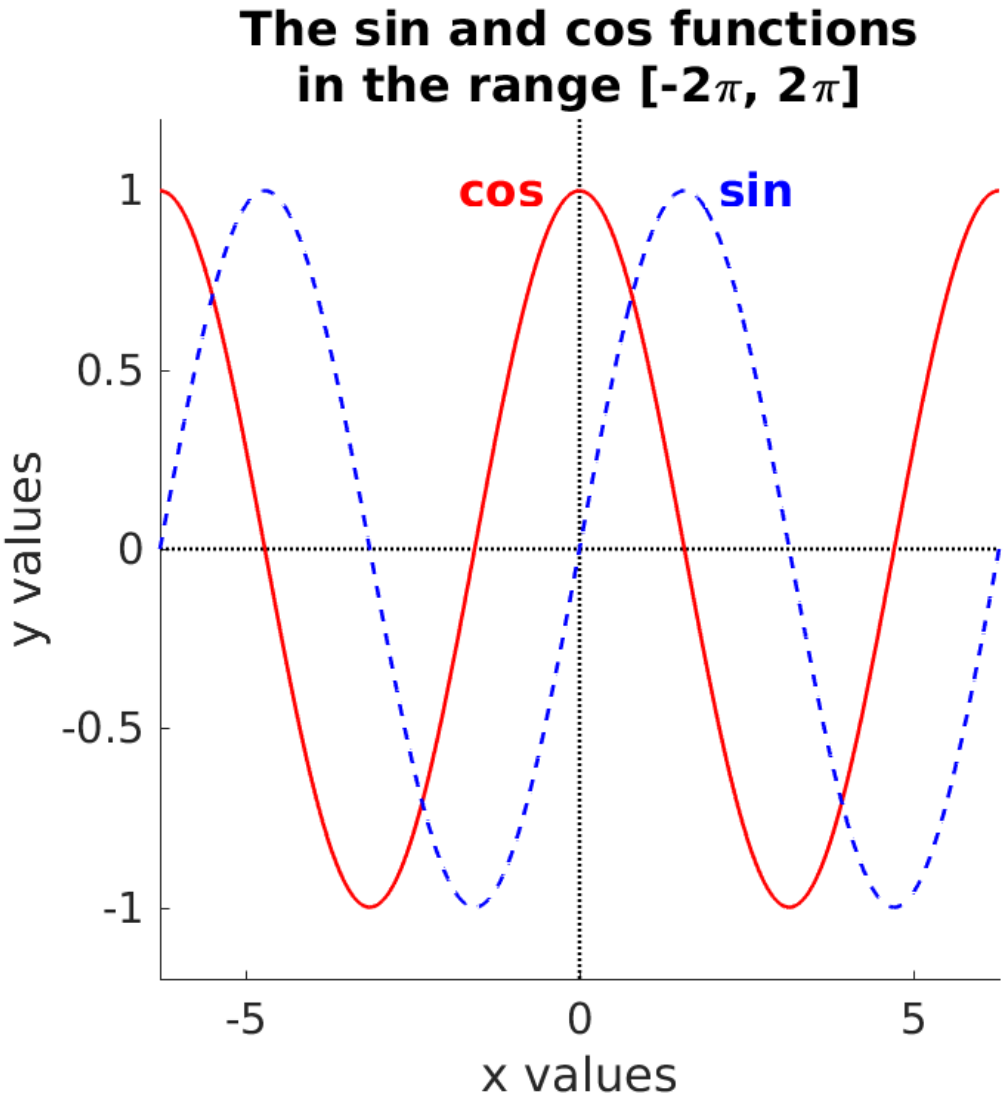
- March 1st 2017 from 2pm to 3pm in 155 Dwinelle Hall
- One sheet of notes (11 inches $\times$ 8.5 inches, double-sided, typed or hand-written) is allowed
- No electronic device (no calculator)
- You must bring your student ID (Cal 1 Card)
- Topics: Lectures 01 through 14

Plot examples

Not a great plot:



A better plot:



The good plot (non-exhaustive) checklist

- Is each written element easy to read? Adjust the font color and font size of the title, axis labels, tick labels, legend, and of any other annotation
- Are the axes properly labeled? Include units if relevant
- Are the axis ticks properly set? Check the spacing and format of the ticks and their labels (*e.g.*, do not use too many significant digits)
- Are the axis limits properly set? Do not cut graphic elements but avoid unused white space
- Is each graphic element (*e.g.*, lines) properly labeled? Do no leave any guess work for your audience, but avoid redundant information. In most cases, directly labelling the lines is better than using a legend box
- Do not rely solely on colors to distinguish graphic elements (*e.g.*, also use different line types, widths, labels, markers, etc.). Your plot should be unambiguous when converted to grayscale

The plot command

```
>> plot(x, y)
>> plot(x, y, s)
```

s is optional. It can be a combination of: a color and/or a line type and/or a marker type.

- color: `'r'` for red, `'b'` for blue, ...
- line type: `'-'` for solid line, `':'` for dotted line, ...
- marker type: `'o'` for circles, `'s'` for square, ...

Open Matlab and try the following:

```
>> help plot % Get list of colors, line types, marker types
>> plot(1:10, 1:10) % Default is blue line
>> plot(1:10, 1:10, 'r') % Red line
>> plot(1:10, 1:10, 'o') % Use circles, no line
>> plot(1:10, 1:10, '--') % Dashed line
>> plot(1:10, 1:10, ':s') % Dotted line and squares
>> plot(1:10, 1:10, 'k-.^') % Black dash-dot line and triangles
```

The plot command: more options

```
>> plot(x, y)
>> plot(x, y, s)
>> plot(x, y, option1, value1, option2, value2, ...)
>> plot(x, y, s, option1, value1, option2, value2, ...)
```

Useful options:

- 'LineWidth' (value is a number)
- 'MarkerSize' (value is a number)
- 'MarkerEdgeColor' and 'MarkerFaceColor' (value is a color)

Open Matlab and try the following:

```
>> % Thicker line
>> plot(1:10, 1:10, 'LineWidth', 3)
>> % Big red circles
>> plot(1:10, 1:10, 'ro', 'MarkerSize', 15)
>> % Green dotted line and red and blue squares
>> plot(1:10, 1:10, 'g:s', 'MarkerSize', 15, ...
>>      'MarkerEdgeColor', 'r', 'MarkerFaceColor', 'b')
```

Adjust the axes and the ticks

Adjust the limits of the axes:

```
>> % To adjust the limits of the axes, use:
>> axis([xmin, xmax, ymin, ymax])
>> % Or:
>> xlim([xmin, xmax])
>> ylim([ymin, ymax])
```

Make the axes aspect ratio 1:1 (so that one unit on the x-axis has the same length as one unit on the y-axis)

```
>> axis equal
```

Open Matlab and try the following:

```
>> plot(1:10, 1:10)
>> axis([-10, 10, -20, 40])
>> xlim([0, 15])
>> ylim([0, 20])
>> axis equal
```

Adjust the axes and the ticks (continued)

```
>> % To adjust the location of the ticks:
>> set(gca, 'XTick', myArray1)
>> set(gca, 'YTick', myArray2)
>> % One can use additional options:
>> set(gca, 'XTick', myArray1, option1, value1, ...)
>> set(gca, 'YTick', myArray2, option1, value1, ...)
```

`myArray1` and `myArray2` must be column or row vectors. Useful options:

- `'FontSize'` (value is a number)
- `'FontWeight'` (value is one of `'bold'`, `'light'`, `'demi'`)

Open Matlab and try the following:

```
>> plot(1:10, 1:10)
>> set(gca, 'XTick', 1:2:10)
>> set(gca, 'XTick', 1:0.5:10, 'FontSize', 14)
>> set(gca, 'YTick', [1, 3, 10], 'FontWeight', 'bold')
```

Title and axis labels

```
>> title(s)
>> xlabel(s)
>> ylabel(s)
>> % One can use additional options:
>> title(s, option1, value1, option2, value2, ...)
>> xlabel(s, option1, value1, option2, value2, ...)
>> ylabel(s, option1, value1, option2, value2, ...)
```

s is a character string. Useful options are 'FontSize' and 'FontWeight'

Open Matlab and try the following:

```
>> plot(1:10, 1:10)
>> xlabel('x values')
>> ylabel('y values', 'FontSize', 14)
>> title('Great title', 'FontWeight', 'bold', ...
>>       'FontSize', 20)
```

Add text to a plot

```
>> text(x, y, s)
>> text(x, y, s, option1, value1, option2, value2, ...)
```

x and y are the x- and y-location of the text. s is the character string to be displayed. Useful options are:

- 'FontSize' (value is a number)
- 'FontWeight' (value is one of 'bold', 'light', 'demi')
- 'Color' (value is a color)

Use _ to write subscripts and ^ to write superscripts. Use {} if the subscript or superscript has several characters

Open Matlab and try the following:

```
>> plot(1:10, 1:10)
>> text(2, 8, 'Hello')
>> text(6, 5, 'e^{2x}, CO_2', 'Color', 'r', ...
>>       'FontSize', 16) % Try out _ and ^
```

Log scales

```
>> semilogx(x, y) % Plot with x axis on a log scale
>> semilogy(x, y) % Plot with y axis on a log scale
>> loglog(x, y) % Plot with both axes on a log scale
```

Open Matlab and try the following:

```
>> plot(1:50, log(1:50))
>> semilogx(1:50, log(1:50))
```

Other way to set log-scale axes:

```
>> set(gca, 'XScale', 'log')
>> set(gca, 'YScale', 'log')
```

Subplots and hold on/off

Subplots:

```
>> % Create m * n subplots, arranged as m rows and n
>> % columns, and select subplot number i
>> subplot(m, n, i)
```

Multiple plots in the same window:

```
>> % Prevent the plot window to be erased when something
>> % new is plotted:
>> hold on
>> % Make the plot window to be erased each time something
>> % new is plotted:
>> hold off
```

Open Matlab and try the following:

```
>> hold off; plot(1:10, 1:10, 'b'); plot(1:10, 11:20, 'r')
>> hold on;  plot(1:10, 1:10, 'b'); plot(1:10, 11:20, 'r')
```

Legend box

Use the function `legend` to add a legend box to a plot

```
>> % Pass the labels as separate input arguments
>> legend(label1, label2, label3, ...)
>> % Or as a single cell array
>> legend({label1, label2, label3, ...})
>> % One can use additional options:
>> legend({label1, label2, ...}, option1, value1, ...)
```

- `label1`, `label2`, `label3`, ... are character strings: the labels of the lines (or sets of points), in the same order as they were plotted
- There must be one label for each line (or set of points) that has been plotted on the figure so far
- A useful option is `'Location'`, which is used to set the location of the legend box. It can take the values `'North'`, `'South'`, `'East'`, `'West'`, `'NorthOutside'`, `'SouthOutside'`, `'EastOutside'`, and `'WestOutside'`

Legend box: example

Open Matlab and try the following:

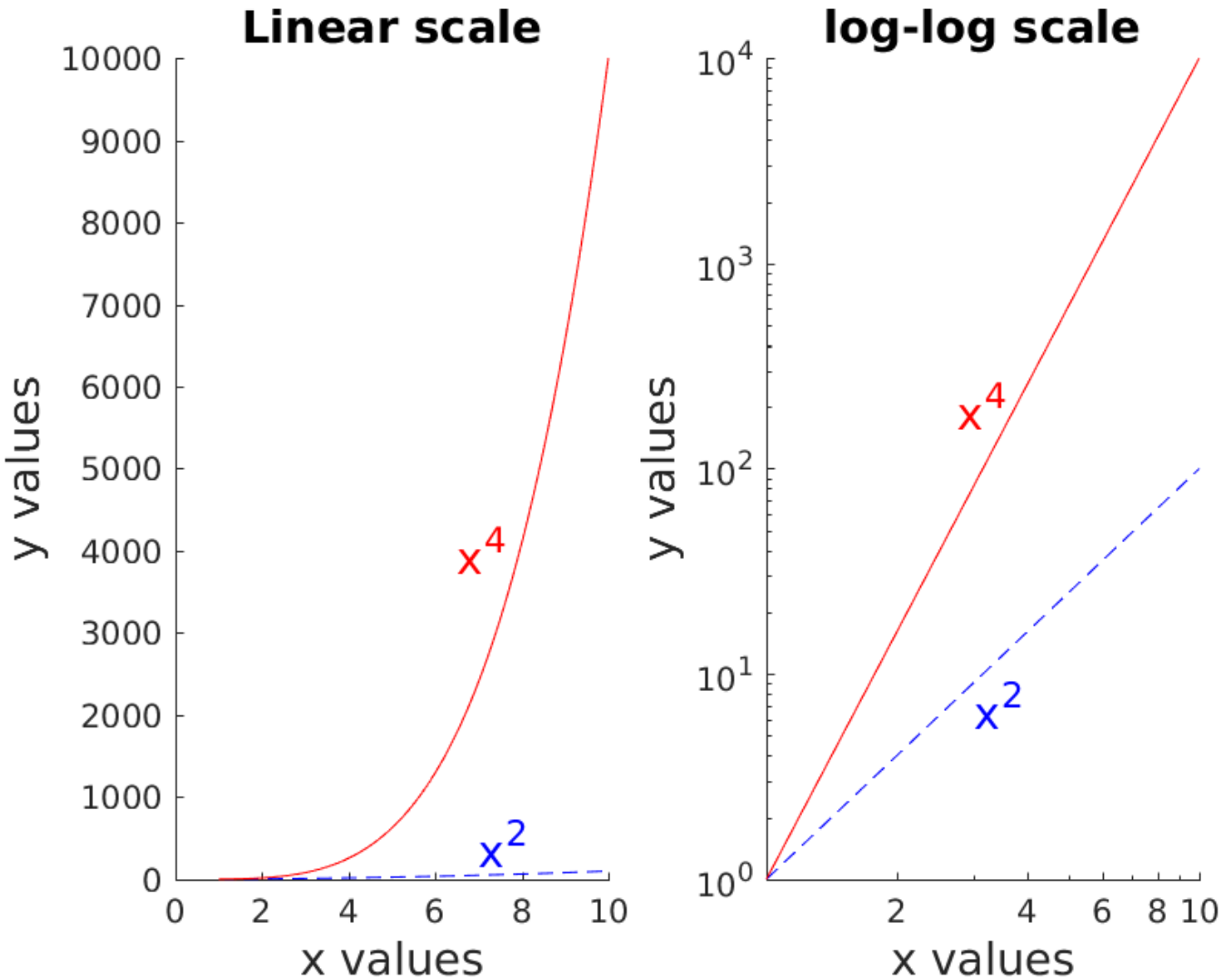
```
>> close all % Close all figures that are currently open
>> x = -pi:0.1:pi;
>> plot(x, cos(x), 'r-') % Cosine as a red line
>> hold on
>> plot(x, sin(x), 'go') % Sine as green circles
>> legend('The cosine function', 'The sine function', ...
>>        'Location', 'NorthWest')
>> plot(x, -cos(x).^2) % The legend box will not show this line
```

One can plot several lines in one call to the function `plot`:

```
>> close all
>> x = -pi:0.1:pi;
>> plot(x, cos(x), 'r-', x, sin(x), 'go')
>> legend('The cosine function', 'The sine function', ...
>>        'Location', 'NorthWest')
```

Full example of 2D plot

See script called `plotting_example.m` (posted on bCourses) that was used to create the following plot:



Introduction to the meshgrid function

Consider two vectors:

```
>> x = [1, 7, 4]; % x-locations
>> y = [0, 2]; % y-locations
```

The coordinates of all the points that can be obtained by combining these x- and y-locations are:

```
  (1, 0), (7, 0), (4, 0)
  (1, 2), (7, 2), (4, 2)
```

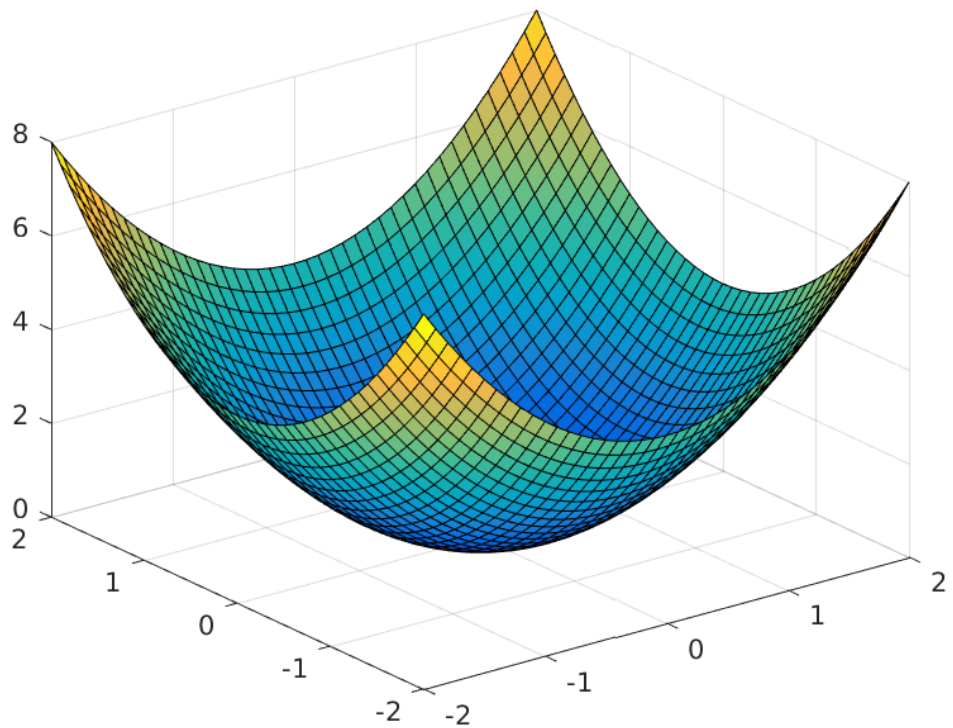
The two 2×3 matrices of all these x- and y-values can be generated using meshgrid:

```
>> [xg, yg] = meshgrid(x,y)
xg =
     1     7     4
     1     7     4
yg =
     0     0     0
     2     2     2
```

Use of the meshgrid function for surf plots

Explain with an example: $f(z) = x^2 + y^2$

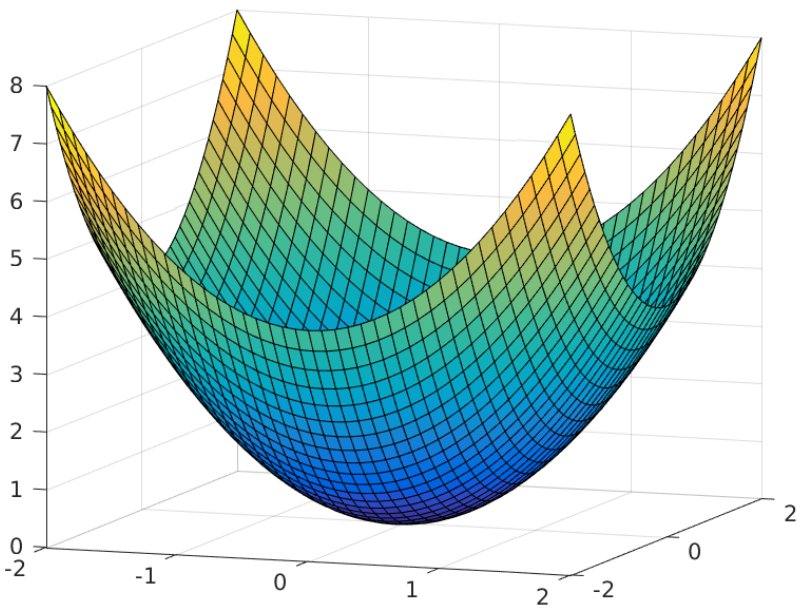
```
>> x = -2:0.1:2;
>> y = -2:0.1:2;
>> [xg, yg] = meshgrid(x, y); % Create x- and -y grids
>> z = xg.^2 + yg.^2;         % Calculate z values
>> surf(x, y, z)              % Surface plot
>> surf(xg, yg, z)            % Should create the same plot
```

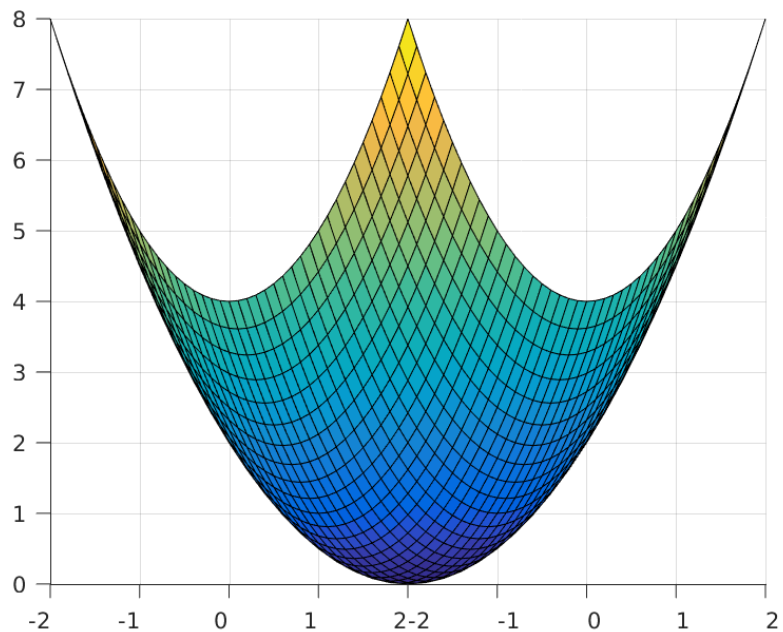


Surf plots: point of view

```
>> % Change the point of view using:
>> view(az, el)
>> % Experiment using the previous example:
>> view(20, 10) % See plot on the left
>> view(45, 0)  % See plot on the right
```

`az` is the viewing angle in degrees (rotates around the vertical) and `el` is the elevation

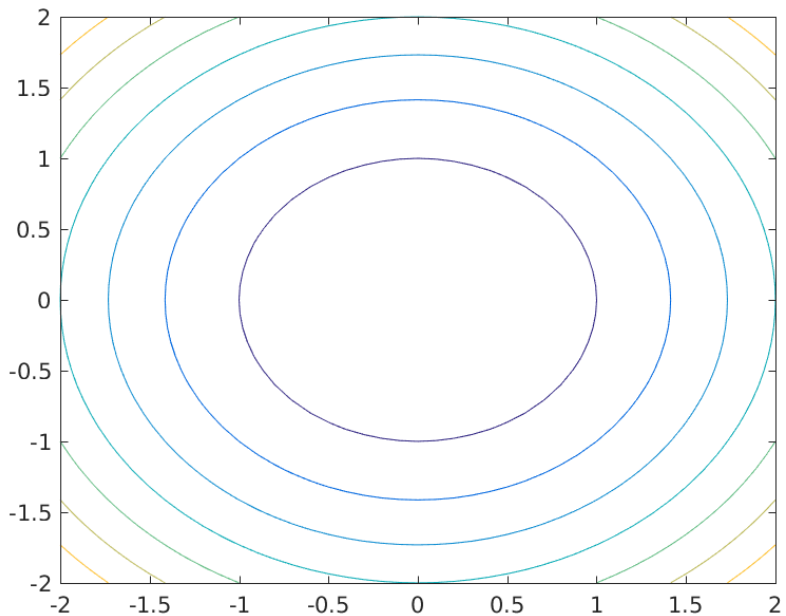


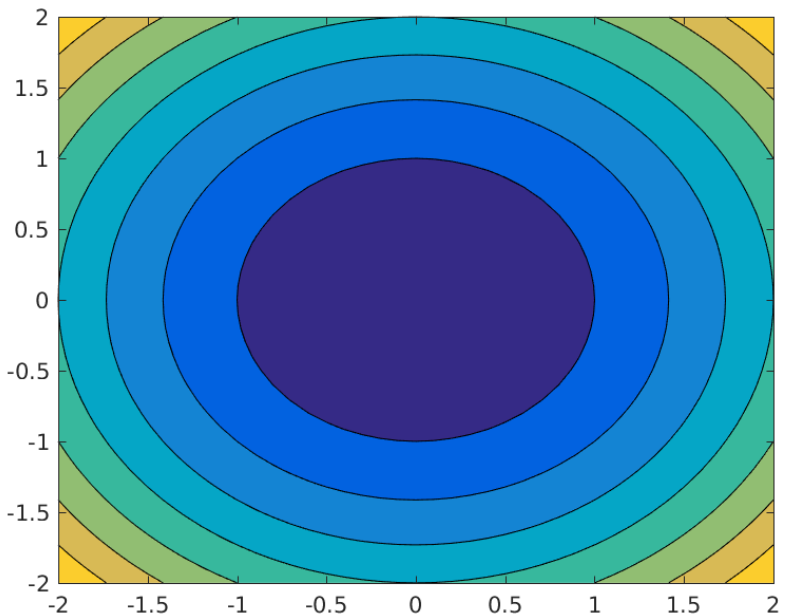


Use of the meshgrid function for contour/contourf plots

Explain with an example: $f(z) = x^2 + y^2$

```
>> x = -2:0.1:2;
>> y = -2:0.1:2;
>> [xg, yg] = meshgrid(x, y); % Create x- and -y grids
>> z = xg.^2 + yg.^2;         % Calculate z values
>> contour(x, y, z)           % Contour plot (left)
>> contourf(x, y, z)          % Filled contour plot (right)
>> % contour(xg, yg, z) and contourf(xg, yg, z) should
>> % create the same plots, respectively
```





Contourf plots: color scale and color bar

```
>> % Set range of colorscale
>> caxis([zmin, zmax])
>> % Add a colorbar
>> cb = colorbar()
>> cb = colorbar(option1, value1, option2, value2, ...);
>> % Set x label (horizontal colorbar):
>> xlabel(cb, 'Color bar label');
>> % Set x label (vertical colorbar):
>> ylabel(cb, 'Color bar label');
```

A useful option for the function `colorbar` is `'Location'`, which is used to set the location of the colorbar. It can take the values `'North'`, `'South'`, `'East'`, `'West'`, `'NorthOutside'`, `'SouthOutside'`, `'EastOutside'`, and `'WestOutside'`

Contourf plots: color scale and color bar (example)

Using the same x, y, and, z values as before:

```
>> contourf(x, y, z)
>> caxis([0, 15])
>> cb = colorbar('Location', 'WestOutside');
>> ylabel(cb, 'One color bar', 'FontSize', 14);
>> cb = colorbar('Location', 'NorthOutside');
>> xlabel(cb, 'Another color bar', 'FontSize', 14);
```