Lecture 13:
Graphics and Visualisation

Overview

- Graphics & Visualisation
- 2D plotting
  1. Plots for one or multiple sets of data, logarithmic scale plots
  2. Axis control & Annotation
  3. Other forms of 2D plots: charts, histograms

Graphics and visualisation of data in Matlab

- There will be many occasions where you will use a computer, not to perform calculations on data, but rather just to visualise some data.

- Often the appropriate visualisation will be easy to choose, but in more complex cases a basic understanding of the human visual system is important for designing a successful visualisation of the data.

Simple 2D plotting that you already familiar with

```matlab
>> plot(x, y); % Plots the vector y versus the vector x.
>> plot(y); % Plots the elements of vector y versus their indices.
% If y is complex this is equivalent to plot(real(y), imag(y)).
>> plot(x, y, s); % Plots the vector y versus the vector x using markers to represent the data points. The markers used are controlled by the character string s. s is made up of up to 3 (or 4) characters: - a character denoting a colour (red, yellow, blue, etc.). % - a character denoting a symbol to use at each coordinate: . (point), o (circle), * star, v (triangle), etc.
% - a character denoting the line style between coordinates: - (solid), : (dotted), -. (dashdot), -- (dashed).
```
Plotting several data sets on one set of axes

`>> plot(X1, Y1, X2, Y2, X3, Y3, ...);`

Plots Y1 versus X1, Y2 versus X2, Y3 versus X3, etc.

`>> plot(X1, Y1, S1, X2, Y2, S2, ...);`

As above, but with plotting specification strings for each data set.

- Example:

  `>> plot(X, Y, 'y--', X, Y, 'go');`

  Plots the data twice: first as a yellow dashed line, then with green circles at the data points.

Plotting in logarithmic scales

`>> semilogx(x, y);`  % The same as plot(x, y) but with a log (base 10) x-axis scale.

`>> semilogy(x, y);`  % The same as plot(x, y) but with a log (base 10) y-axis scale.

`>> loglog(x, y);`    % The same as plot(x, y) but with a log scale on both axes.

- Be on the lookout for occasions when logarithmic plots are more appropriate for your data.

Cases when you might want to use logarithmic plots:

- When there is a very large range of magnitudes in your data.
- When the ratio of data values relative to each other are more important than their absolute magnitudes.
- When you are seeking to show that the relationship between two variables follows some power law. A plot of a power function should produce a straight line on log axes.

Example

- For example, assume we have the following function: 
  \[ y = a \cdot x^3 \]

  - Taking the log of both sides produces: 
    \[ \log(y) = \log(a) + 3 \cdot \log(x) \]
    which is a straight line with slope of 3 and a y intercept of \( \log(a) \).
**Axis control**

- Matlab generally does a fairly good job of automatically scaling axes to suit the data.
- However, you can set the axis range yourself with the `axis` function:
  ```matlab
  >> axis([xmin, xmax, ymin, ymax]);
  ```
- Note that the argument supplied to `axis` is one vector.
- Cases when you might want to set the axes manually:
  - When you want to ensure that the y-axis range starts at zero and not at the minimum value in the array of y values.
  - When you are generating two or more plots side by side for comparison and you want to ensure that both plots have the same axis ranges.

**More on the `axis` command**

- The `axis` command can do other useful things. For example,
  ```matlab
  >> axis equal; % Makes axis increments equal on both axes.
  % This preserves the geometry of shapes and prevents them from appearing distorted.
  >> axis off; % Removes the axes from the plot.
  ```
- Type `help axis` at the Matlab command prompt to learn more about the `axis` command.

**Graph annotation**

- You should always label all axes in a plot and annotate the plot with a title.
- The `title` function is used to generate a plot title.
  ```matlab
  >> title('The plot title');
  ```
- The `xlabel` and `ylabel` functions are used to label the x and y axes respectively:
  ```matlab
  >> xlabel('X axis label');
  >> ylabel('Y axis label');
  ```
- Display the units of measurements used on each axis.
- You can place text at a specified location with the `text` function.
  ```matlab
  >> text(3, 4, 'The point (3,4) is very interesting');
  ```

**The `legend` function**

- The `legend` function is used to provide legends for each data set that have been displayed simultaneously on one set of axes via the `plot(X1, Y1, X2, Y2, X3, Y3, ...)` function.
- For example:
  ```matlab
  >> legend('Label 1', 'Label 2', ..., 'Label N');
  % Adds a legend for each data set in the plot.
  ```
- Type `help legend` at the Matlab command prompt to learn more about the many options available with this function.
### Creating and switching between figure windows

- **The `figure` function is used to control different figure (or plot) windows.**

  ```matlab
  >> figure;        % Creates a new figure window.
  % This becomes the "current" figure.
  >> figure(N);     % Makes figure N the current figure window.
  % This function creates a new figure window
  % called N if N does not already exist.
  >> delete(N);     % Deletes figure N.
  ```

### Multiple plots within the one figure window

- You can divide a figure window into a grid and place subplots within individual grid cells.

- **The `subplot` command is used to break a figure window into a grid:**

  ```matlab
  >> subplot(NoOfRows, NoOfCols, CellNo)
  - This function divides the figure window into NoOfRows rows and
    NoOfCols columns.
  - The cells are numbered from left to right, top to bottom.
  - After breaking the figure window into cells, the `subplot` function
    makes the cell numbered CellNo the "active" cell.
  - The active cell "receives" all subsequent plotting commands.
  ```

### Example

- For example, imagine generating two plots one above the other:

  ```matlab
  >> subplot(2, 1, 1), plot(x1, y1), title('Plot 1');
  % The first call to subplot breaks the figure window into
  % 2x1 cells, making the top one the active cell.
  >> subplot(2, 1, 2), semilogy(x2, y2), title('Plot 2');
  % A second call to subplot is needed to plot to the
  % second cell.
  ```

### Another example

  ```matlab
  >> xx = 0 : 0.1 : 10;
  >> subplot(3, 1, 1), plot(xx, sin(xx)), xlabel('x'), ylabel('sin(x)');
  >> subplot(3, 1, 2), plot(xx, cos(xx)), xlabel('x'), ylabel('cos(x)');
  ```
Other forms of 2D plots

- You can plot 2D data in many different forms:

  >> bar(x, y); % Creates a bar graph. Values in array x
               % are used to label each bar.
               % Values in array y are used to determine
               % the height of each bar.

  >> pie(x);    % Creates a pie graph. The values in
               % array x are converted into percentages
               % so that their sum is 100%.

  >> hist(x, nbins); % Plots a histogram. The values in
                     % array x are divided into nbins bins.
                     % The number of data values that fall
                     % in each bin are then plotted.
                     % This function can also return the bin
                     % frequencies.