1. **Setup & the Basic Fitting Interface**

This section will introduce how to fit a linear equation to data using MATLAB’s Basic Fitting Interface and how to evaluate if the linear fit is justified.

**Initial Setup:** The default colors for MATLAB plots often do not provide enough contrast for the residual plots. Run the script ***FixColors.m*** (i.e., copy the script into your current folder and type *FixColors* at the command prompt) before trying these fitting examples to set up a higher contrast default color pallet. This script needs to be rerun anytime a new MATLAB session is started.

***Load Data***: The data for the examples in this lab are available in the MATLAB data file *EMP1.mat*. Copy that file to the current MATLAB working directory. Then at the command prompt type
 >> ***load EMP1***
to load the variables into your workspace.

**Figure 1:** Spring Experiment Setup. Weights used to add force to the spring.

**Fitting a linear equation to spring data:**  This first example is shown in Figure 1. It examines an experiment where the force on a hanging spring is increased using weights and the resulting spring lengths are measured.

**Independent variable:** Force – this variable in the workspace contains a series of different weights used to load the spring. The load is in lbf.

**Dependent variable:** L – this variable in the workspace contains the change in length from the resp position of the spring in inches.

**Fit a Linear Equation:** Follow the steps show in Figure 2.

Steps for using the Basic Fitting Interface

1. Plot the data using the normal plot command:

e.g., >> plot(x, y, ‘o’)

1. On the Figure window open basic interface:
***Tools*** menu 🡺 ***Basic fitting***
This will open the dialogue box at right
2. Under ***Plot fits*** select: ***linear***
3. Select: ***Show equation***

This is will likely be preselected.

Adjust significant digits if needed
For these problems 3 is
appropriate.

 **Figure 2:** Basic Fitting Interface and the initial steps to fit a line to the data (MATLAB R2020a). The A and B labels are for the following section on plotting residuals.
An alternative Figure 2, for MATLAB R2019a and earlier, is available online.

A

B

****This should result in a plot similar to Figure 3.

**Figure 3**: Example of a linear fit line with data. This is for the spring data fitted as outlined in Figure 2.

**Plotting Residuals**

1. Nomenclature Review:

 vectors individual values

 🡺 predictor (independent variable) vector  🡺 an individual predictor value

 🡺 response (dependent variable) vector  🡺 an individual response

 🡺 predicted values for each value in *x* (the fits)  🡺 the predicted value at *xi*

1. Key issue**: residuals**
	1. Definition: *Residual = actual y - predicted y at the same x location*

i.e., What is left after model pattern is subtracted from the data

If the model is complete residuals should not have any identifiable pattern

* 1. Nomenclature $r\_{i}=y\_{i}-\hat{y}\_{i}$ residual for an individual point

$r=y- \hat{y}$ vector of residuals for all points

* 1. Plotting residuals:
* Residuals vs. fits (the predicted values of y) is the most common plot
* Residuals vs. independent variable (x) is also common.
This is what MATLAB’s Basic Fitting Interface does. For problems with a single independent variable (one x variable) and a monotonic function, plotting residuals vs. the x variable is essentially equivalent to plotting residuals vs. fits.
* Usually plotted as a scatter plot of points. However, a bar graph is sometimes used in MATLAB for clarity. Run the FixColors function before plotting to improve the color contrast
	1. To plot In Basic Fitting Interface:

Check ***Plot Residuals*** (***A*** in Figure 2)

Select ***Scatter or Bar plot*** from drop down me (***B*** in Figure 2)

 🡺 Look for patterns in the residuals, particularly signs of curvature.

**II. Fitting Linear Equations Example: Deflection of a Cantilever Beam (d vs. P.)**

d

P

This section covers a linear fit example; the relationship between deflection, d, and a point force, P, applied to the end of a cantilever beam as shown in Figure 4. The data is stored in the variables d and P. Where d is the deflection in centimeters, and P is magnitude of the force in kilonewtons.

**Figure 4:** Diagram of deflected cantilever beam. P is the applied force (kN) and d is the size of the deflection (cm)

1. **Model:** The general model being fit in all the examples in this lab is y = mx + b plus error. For this particular case the model is:

d = m\*P + b plus error.

1. **Setup:** To set up this problem plot the two variables in MATLAB. We are plotting d vs. P, therefore d is the y-variable and P is the x variable). This is experimental data that is being plotted so it must be plotted as individual points (not a line!). The plot command must reflect these two aspects of the nature of the data (i.e., use plot(P, d, ‘p’)).
2. **Adding a Fit:** In the figure window with this plot select the “tools” menu and access the “Basic Fitting” GUI interface. On the interface select the “linear” option. The ***Show equations*** option should also be selected. Adjust the significant digits as necessary (but avoid excessive digits).
3. **Predict:** The Basic Fitting interface automatically shows the fitted line.
4. **Plot:** To plots are needed to show and evaluate the fit. 1) The y vs x plot of both the data and the fitted line (this part is complete automatically when using the Basic fitting interface in MATLAB), and 2) A residual plot that shows the residuals as a function of the independent variable, x, the dependent variable, y or the fitted values, $\hat{y}$. The residuals are also sometimes plotted verses order or time. The Basic fitting interface will automatically plot the residuals versus the independent variable (P in this case).

To plot residuals click the “Plot residuals” check box. Notice you have a couple of drop down menus for switching to a scatterplot and for making the residual plot a separate figure window (try these options out. Residual plots are traditionally scatter plots (however in the fitting interface the bar plots are sometimes easier to read. Final plots need to include proper axis labeling; this usually easier to do if you separate the residual plot.

1. **Cantilever Beam:** Includethe resulting plots in your lab due next week and answer the following questions based on the plots.
2. Does a linear fit look appropriate to this data? Justify your answer based on the nature of the residual plot pattern (it should be scatter without a specific pattern). In addition, the xy graph can also be considered.
3. What are the units on the slope (the m parameter)?

**This and the other two grayed boxes are your assignment due next week in one document.**

**III. Linear Equations - Three Fitting Problems**

Consider the following three problems (details listed below the table). For each problem carry out the five steps used in the previous examples (list ***model***, ***setup***, ***fit***, ***predict***, ***plot***). Plotting includes both the xy plot with data and fitted line and the residual plot. Based on these steps evaluate if a linear model is justified. Complete Table 1 as you proceed.

**Table 1:** Summary of three problems

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem** | **Fitted Linear Equation** | **Is it Linear?** | **Evidence** |
| a) Wind Power Generation |  |  |  |
| b) Level Gage Calibration  |  |  |  |
| c) Flowmeter Calibration |  |  |  |

1. Wind Power Generation (*Power* vs. *Wind*): The amount of power produced by a wind turbine is a function of the wind speed. Data collected from tests on a specific wind turbine are included in the EMP1.mat file. The wind speeds in miles per hour are stored in the variable *Wind.* The corresponding power generation values in Megawatts (MW) are stored in the variable *Power*. Treat *Wind* as the independent variable (x) and *Power* as the dependent variable (y). This problem is adapted from a problem in Montgomery, D. C. and Runger, G. C., **Applied Statistics and Probability for Engineers**, Third Edition, Wiley, (2003).
2. LEGO Level Gage Calibration (*Height* vs. *Reading*): The liquid level in a small laboratory tank is measured using a pressure sensor at the bottom of the tank that is connected to a computer via a LEGO® RCX® brick. A raw value between 0 and 1023 is recorded by the computer (this is a 10 bit range in binary). It is desired to find an equation that will calculate the level in the tank based on this raw reading. In the data file *Reading* is the raw value on the computer and Height is the height measured in the tank in cm. For this calibration curve it is desired to predict the height based on the reading, so *Height* is the dependent variable in this case.

1. LEGO Flowmeter Calibration (*Flow* vs. *Read*): The flowrate through a small flow meter in this same laboratory setup as above results in a raw value on the computer of 0 to 1023 (the variable name is *Read*). For comparison the flow through the meter has been recorded in ml/second (this variable is named *Flow*). Estimate the linear calibration curve and evaluate if the linear model is appropriate. For this calibration it is desired to predict flow rate from reading. Flow rate is therefore the dependent variable.

**2. Three fitting problems:** Add the following to your weekly lab document

1) A copy of Table 1 filled,
 2) xy plot of the data and fit on a properly formatted graph for a chosen example
 3) the residual plot for that same data.
You may choose any of the three problems (Wind, LEGO Level or LEGO Flowmeter). Clearly identify the problem chosen.

**IV. Introduction to Function Discovery**

A couple of the previous examples are not quite linear and a different model with some curvature is needed. Two possible alternative models are:

1. an exponential model ( $y=be^{mx}$ or $y=b10^{mx}$). The exponential model has the two common forms. Any data that will fit one will be fit equally well by the other.
2. a power model ($y=bx^{m}$ ). For example a very common form of this model would be the squared model where y is proportional to x2.

These are both still two-parameter models (i.e., two values in the equation are fitted from the data) but they are capable of modeling some common cases of curvature. Table 2 summaries three two-parameter models (the linear model plus these two new models).

**Table 2:** Basic Two-Parameter Models

|  |  |  |
| --- | --- | --- |
| **Mathematics Expressions** | **To test Likely Fit** | **To actually fit** |
| **Model type** | **Equation** | **Linearized form** | **Plot type****(MATLAB command)** | **Fit** |
| linear  |  |  | linear (plot) |  y vs. x |
| exponential |  |  | semilog (semilogy) |  ln(y) vs. x |
|  |  | log10(y) vs. x |
| power  |  |  | log-log (loglog) | log10(y) vs. log10(x) |

Next lab how to fit and assess these models will be evaluated. As a start the likeliness that one of these models will fit a specific curved data set can be checked using plots. Notice each model in Table 2 has an associated graph that should be linear if the data follows the respective model. To determine which model is best, the three plots are created. If the data looks scatted about a straight line on the linear plot then the model is likely fit by a linear model, if it looks scatted about a straight line on the semilog plot then it is likely fit by an exponential model, or if it looks scatted about a straight line on the log-log plot then it is likely fit by a power model. If two plots are basically linear then the plot and model with fewer log transformations is generally chosen.

**3. Function Discovery Graphs:** Complete the following and add to your Lab document.

Write a script that prepares and labels all three of these plots (linear, semilogy, and log-log) for the Flowmeter data (data set c above). This may be done:

* Three separate figures. To keep new plots from overwriting previous plots either the figure command needs to be issued between plots, or the script will need to be published.
* Or one figure with the graphs side-by-side (i.e., one figure with one row of three plots). This will require using the subplot function.

Examine the plots and write out which model (linear, exponential or power) you conclude is the best to use for this data set and why. Remember that the linear model was looked at in the previous section. Do not try and fit any of these alternative models or create residual plots at this time (you do not have the techniques to fit the two alternative models yet).

**Appendix, FYI: Applying Matrix Algebra to Linear Regression in MATLAB (**not required**)**

**Linear Regression** using a matrix approach**:** E.G. Another Spring Problem

1. **Model:** For a simplified spring expansion problem: L = m\*Force + b

Set up in Matrix Algebra form:





Y is a column vector of the dependent (response) variable, in this case length change (L)

X is the matrix of independent variables (constant and Force)

 with one column for each coefficient (same number of Columns as β has rows)

 and one row for each observation (same number of rows as Y)

β is the coefficient vector (a column vector)

ε is a column vector of errors (same number of rows as Y)

1. **Setup:** Create the X & Y matrices

 from scratch

 >> X = [1,0;1,0.47;1,1.15;,1,1.64]

 >> Y = [0, 2.5, 5.9, 8.2]’

or you can use the vectors available in the EMP1.mat file using the following MATLAB code. (Force is a column vector of the forces; L is a column vector of the lengths)

 >> X = [ones(size(Force)), Force]

 >> Y = L

If the variables were row vectors you would need to transpose them to column create vectors before carrying out the above steps.

1. **Fit:** We can find the best fit coefficient vector (β) in MATLAB by using left division of the matrices.

 >> B=X\Y

Notes: - notice we are using left division. This is essential - this is not usual linear algebra nomenclature

What is the resulting linear equation?

1. **Predict:** Now calculate the fitted values using matrix multiplication.

 >> Yhat = X\*B

1. **Plot:**  plot the data and the resulting model on a graph to check and present result

>**>** plot(X(:, 2), Y, ‘p’, X(:,2), Yhat)

Add appropriate axis labels and curve legend

A residuals plot is also appropriate. We can also calculate the residuals using matrix algebra: >> res = Y - X\*B

 Then plot the results >> plot(Yhat, res, ‘s’, Yhat,res\*0)