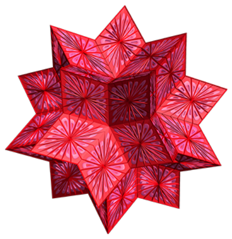
Mathematics Computer Laboratory - Math 1200 - Version 14
Lab 5 - Mathematical Modeling©



Due

You should only turn in exercises in this lab with its title and your name in `Title` and `Subtitle` font, respectively. Edit your document and use appropriate margins to get a polished and neat document.

1. The objective of this lab is to discuss syntax for data transformation, data fitting, and development of appropriate exponential and power models.

Data collection is an integral part of most scientific studies. Data analysis can lead to discovery of well-defined relationships between different variables. Then a mathematical model can be developed based on such relationships. Model interpretation and forecasting, in real-world terms, usually leads to a deeper understanding and more precise future studies. This cycle will continue until a sufficiently accurate mathematical model is developed. This method has led humans to discovery of almost all basic scientific laws that we know today.

2. Starting with a set of data points, let's use Mathematica to find its "best" fit among different classes of functions.

   (a) `xdata={0.5, 1, 1.4, 2, 2.5, 3, 3.6, 4, 4.8}` These are the x or domain values.
   Note: It is convenient to use a descriptive list name.

   (b) `ydata={0.2, 1, 3.5, 8, 15.5, 28, 43.1, 64, 112.5}` These are the corresponding y or range values.
   Note: You might get a warning message about the similarity of names used for the two lists.

   (c) `xydata=Transpose[{xdata, ydata}]` Now we have matched the corresponding data values.

   (d) `dataplot=ListPlot[xydata]` This gives the graph of the data points.

   (e) The command `Fit` will yield the function which "best" fits the data among all functions of a given type.

   `f1=Fit[xydata, {1,x}, x]` You should get $-29.612 + 23.7854\,x$.

   The list {`1,x`} specifies that the function type is linear: $f(x) = c_1 \cdot 1 + c_2 \cdot x$. The last `x` indicates that x is the independent variable.

   (f) `f1graph=Plot[f1, {x, 0, 5}]` This gives the graph of the fit function on the domain $[0, 5]$.

   (g) To see how well this function fits the data, view the graph of both on the same coordinate system. The `Show` command displays graphs on the same coordinate system.

   ```
   Show[dataplot, f1graph]
   ```

   The "best" linear function does not seem to fit the data well!

2. Continued.

(h) Let's find the "best" fit quadratic function.

`f2=Fit[xydata, {1, x, x^2}, x]` You should get $11.235-19.3367\,x+8.26701\,x^2$.

This is the function which "best" fits the data among all functions of the form $f(x)=c_1\cdot 1+c_2\cdot x+c_3\cdot x^2$.

(i) `f2graph=Plot[f2, {x, 0, 5}]`

(j) `Show[dataplot, f1graph, f2graph]` The graph of data points and both fit functions are displayed on the same coordinate system. It is clear that the quadratic function is a better fit.

The general form of the `Fit` command is

```
Fit[List of data points, List of functions, List of independent variables]
```

This command gives the best "least square fit" of the desired function form.

(k) Let's try a function of the form $f(x)=c_1\cdot 1+c_2\cdot x+c_3\cdot e^x$

`f3=Fit[xydata, {1, x, E^x}, x]` You should get $-7.42772+0.744476\,E^x+6.51318\,x$.

(l) `f3graph=Plot[f3, {x, 0, 5}]`

(m) `Show[dataplot, f2graph, f3graph]` From this comparison, one may conclude that the function `f3` is as good a fit as `f2`.

Although it might be possible to list an arbitrary collection of functions and get an accurate fit, such a modeling method has two major draw backs.

(a) The mathematical model may not reveal the true nature of the relationship between different variables and therefore provide no new insights.

(b) The mathematical model may give an inaccurate forecast.

The data in the last part was generated using a cubic relationship between x and y variables including small variations to account for data collection inaccuracies. None of the fit functions showed this behavior and all would be very poor predictor of value of y outside the x interval $[0.1, 4.8]$. Therefore, it is important to choose functions which represent the relationship between variables and also find the smallest number of functions for which there is an accurate fit.

We will now develop an indicator for recognizing exponential, $y=a\,e^{kx}$, and power, $y=a\,x^b$, relationships in data; in either case, taking the logarithm of both sides reveals a linear relationship in transformed data.

$y=a\,e^{kx}\iff$

$\ln y=\ln(a\,e^{kx})\iff$

$\ln y=\ln a+\ln e^{kx}=\ln a+kx\iff$

$Y=kx+A$ where $Y=\ln y$ and $A=\ln a$

$y=a\,x^b\iff$

$\ln y=\ln(a\,x^b)\iff$

$\ln y=\ln a+\ln x^b=\ln a+b\ln x\iff$

$Y=bX+A$ where $Y=\ln y$, $X=\ln x$ and $A=\ln a$

Notice that each of the above steps are reversible. Therefore, an exponential model $y = a\,e^{kx}$ is correct if and only if the plot of $\ln y$ versus x is a straight line. While a power model $y = a\,x^b$ is correct if and only if the plot of $\ln y$ versus $\ln x$ is a straight line. Hence the "straightness" of the plot of certain transformations of data is an indicator of the category of model that is most like the actual data.

3. Now let's find the model which best fit our `xydata`. Form the transformed data sets $(\ln y,\, x)$ and $(\ln y,\, \ln x)$. Then check for their "straightness".

(a) `lnydata=N[Log[ydata]]`
`Log` function applied to a list gives the `Log` or natural logarithm of each point of that list. As usual, the command `N` gives the decimal approximation. You could not do this if zero or negative numbers were in the `ydata`.

(b) `xlnydata=Transpose[{xdata, lnydata}]` This gives data points of the form $(x,\, \ln y)$.

(c) `logplot=ListPlot[xlnydata, Joined->True]`
It is customary to call this type of plot a Log Plot. Joining the transformed data points with straight lines is a visual help in determining straightness.

(d) `lnxdata=N[Log[xdata]]` This gives the natural logarithm of x values.

(e) `lnxlnydata=Transpose[{lnxdata, lnydata}]` This gives data points of the form $(\ln x,\, \ln y)$.

(f) `loglogplot=ListPlot[lnxlnydata, Joined->True]` This type of plot is called a Log-Log Plot.

(g) To compare the "straightness" of these two models, let's display the two graphs of transformed data on the same coordinate system.

`Show[logplot, loglogplot]` Notice that the `loglogplot` looks straighter.

(h) Now find the straight line which best fit the `lnxlnydata`.

`Fit[lnxlnydata, {1,u}, u]` You should get `0.208145 + 2.81877 u`.

Therefore $\ln y = 0.208145 + 2.81877 \ln x$. Exponentiating both sides yields:

$$\begin{aligned}
&\ln y = 0.208145 + 2.81877 \ln x \\
&e^{\ln y} = e^{0.208145 + 2.81877 \ln x} \\
&y = e^{0.208145}\, e^{2.81877 \ln x} = e^{0.208145}\, \left(e^{\ln x}\right)^{2.81877} \\
&y = e^{0.208145}\, x^{2.81877}
\end{aligned}$$

Using Mathematica:

`E^(0.208145) x^(2.81877)` Thus, approximately, $y = 1.23\,x^{2.82}$.

```
y=1.23 x^(2.82)
```

(i) `modelplot=Plot[y, {x, 0, 5}]`

(j) `Show[dataplot, modelplot]` This graph indicates that $y = 1.23\,x^{2.82}$ is a very good fit.

3. Continued.

(k) `Clear[xdata, ydata, xydata, dataplot, f1, f1graph, f2, f2graph, f3, f3graph];`
`Clear[lnydata, xlnydata, logplot, lnxdata, lnxlnydata, loglogplot, y, modelplot]`

4. We will now discuss another modeling example.

(a) `xdata={1.1, 2, 3.1, 4.1, 4.9, 6}`

(b) `ydata={0.34, 0.46, 0.61, 0.83, 1.12, 1.51}`

(c) `xydata=Transpose[{xdata, ydata}]`

(d) `dataplot=ListPlot[xydata]`

(e) `lnydata=Log[ydata]`

(f) `xlnydata=Transpose[{xdata, lnydata}]`

(g) `logplot=ListPlot[xlnydata, Joined->True]` The transformed data $(x, \ln y)$ seem pretty straight.

(h) `lnxdata=N[Log[xdata]]`

(i) `lnxlnydata=Transpose[{lnxdata, lnydata}]`

(j) `loglogplot=ListPlot[lnxlnydata, Joined->True]` The transformed data $(\ln x, \ln y)$ does not seem as straight.

(k) `Show[logplot, loglogplot]` The logplot looks more straight.

(l) `Fit[xlnydata, {1,u}, u]` You should get $-1.4111 + 0.30453\,\mathtt{u}$.

Therefore $\ln y = -1.4111 + 0.30453x$. Exponentiating both sides yields:

$$\begin{aligned}
&\ln y = -1.4111 + 0.30453x \\
&e^{\ln y} = e^{-1.4111+0.30453x} \\
&y = e^{-1.4111}\, e^{0.30453x}
\end{aligned}$$

Using Mathematica:

`E^(-1.4111) E^(0.30453  x)` Thus, approximately, $y = 0.24\, e^{0.3x}$.

`y=0.24 E^(0.3 x)`

(m) `modelplot=Plot[y, {x, 0.5,6.5}]`

(n) `Show[dataplot, modelplot]` This graph indicates that $y = 0.24\, e^{0.3x}$ is a very good fit.

(o) `Clear[xdata, ydata, xydata, dataplot, lnydata, xlnydata, logplot]`
`Clear[lnxdata, lnxlnydata, loglogplot, y, modelplot]`

In each of the following exercises, determine whether the exponential model or the power model better fits the data using the straightness indicator. Then develop the chosen model. Follow the steps in parts 3 and 4. Add appropriate **comments** (type your comments in text cells) to explain your work.

A. Let d be the average distance, in millions of miles, of planets in our solar system from the Sun and let T years be the corresponding period (time for one complete revolution around the Sun) of each planet.

Planet	d	T
Mercury	35.96	0.24084
Venus	67.20	0.6152
Earth	92.90	1.0000
Mars	141.6	1.8808
Jupiter	483.3	11.8620
Saturn	886.2	29.4580
Uranus	1783	84
Neptune	2793	164.8

B. Let L represent the length of a simple pendulum in centimeters and let period T be the corresponding time, in seconds, in which the pendulum completes one swing back and forth.

Note: Assuming a pendulum is not allowed to swing too far from the vertical, its period T is independent of the angle it swings through.

L	5	10	15	20	25	30	35
T	0.45	0.63	0.78	0.89	1.00	1.10	1.19

C. Let P represent the U.S. population size, in millions, at census years 1790 to 1890.

Year	1790	1800	1810	1820	1830	1840	1850	1860	1870	1880	1890
P	3.929	5.308	7.240	9.638	12.866	17.069	23.192	31.443	38.558	50.156	62.948

Note: You may want to start with setting year 1780 as time zero. Then the dates will transform to times 10 to 110. In Mathematica you can do this by `time=year−1780` where `year` is the list of the years: `year={1790, 1800 , ... }`.