Appendix C: `fit` & `plot` Example

This appendix provides a longer example using `fit` and `plot` to analyze and display data. The example data set is a recapitulation of work done in your last PHYS 200 lab on a.c. circuits and resonance: the current flowing in a LRC circuit is measured as a function of the frequency f of the a.c. voltage source. You may recall[1] that because of frequency dependent reactances, a series LRC circuit has a current, I, that depends on frequency. You can follow along with this example if you grab a copy of the data file. At the *linux* `%` prompt type:

```
% cp /usr/local/physics/help/LRC.dat .
```

and a copy of the datafile should appear in your current directory (which is called "."). Similarly you can get a copy of the actual commands used in the below example by:

```
% cp /usr/local/physics/help/LRC.com .
```

Theoretically the current as a function of frequency should be given by:

$$I = \frac{V}{\sqrt{R^2 + \left(\frac{1}{2\pi f C} - 2\pi f L\right)^2}} \tag{C.1}$$

Start the program `fit` by typing to the *linux* `%` prompt:

```
% fit
*
```

[1] or see Electrical Measurements Review particularly Fig. 3.1 on page 80

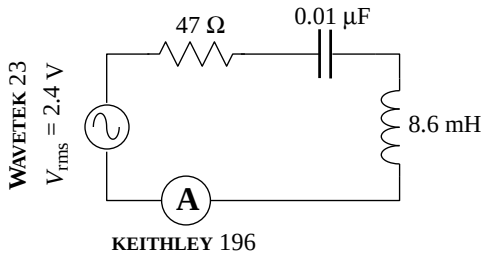


Figure C.1: This series LRC circuit was constructed. A WAVETEK 23 function generator provided a GPIB-controlled sine wave; a KEITHLEY 196 reported the resulting current via GPIB. The program `LRC.f` controlled the data collection.

The program responds with its own prompt: `*` . We need to tell `fit` the name of the datafile we want to analyze and then read in that data:

```
* set file=LRC.dat
* read
freq(Hz) I(A) dI(A) by default this line ignored!
READ Done.
```

The first line of the file (`freq(Hz)`...) is simply echoed back as a way of confirming that you are reading the intended file. It should be noted the file `LRC.dat` was written in the default form, with x values in the first column (`XCOL=1`) , y values in the second column (`YCOL=2`), and y-error values in the third column (`YECOL=3`). Next we need to tell `fit` what function should match the data.

```
* set f(x)=k1/sqrt(k2^2+(1/(2*pi*k3*x)-2*pi*k4*x)^2)
```

where $V = k_1$, $R = k_2$, $C = k_3$, $L = k_4$ and $f = x$. For nonlinear fits we must always give starting parameter estimates. In this case that's easy as I measured the component values when I built the circuit:

```
* set k1=2.4 k2=47 k3=1e-8 k4=.0086
```

Next I command a `fit`: please adjust the parameters k_1–k_4 to achieve the smallest possible reduced χ^2:

```
* fit
Enter list of Ks to vary, e.g.  K1-K3,K5 k1-k4
FIT finished with change in chi-square= 1.5258789E-04
6 iterations used
REDUCED chi-squared= 3.790897 chi-squared= 363.9261
K1= 2.262721 K2= 190.7750 K3= 0.9930065E-08
K4= 0.8230284E-02
Display covariance/curvature matrices?  No, Screen, File [N,S,F]
```

The result is pretty much as expected: reduced χ^2 is high, but in the usually acceptable range, the parameters are close to my original guess, with the exception of $k_2 = R$. On reflection even R is OK as the inductor itself has resistance (77.8 Ω) and according to Fig. 3.1 on page 80, the function generator also has an internal 50 Ω resistor. Thus: $47 + 77.8 + 50 = 167.8$ Ω should have been expected. Currently `fit` is asking if I want to look at the covariance matrix, but I'll delay replying No, Screen, or File until I look at the fitted curve.

I start another terminal to display, using the program `plot`, what the fitted curve looks like: at the *linux* `%` prompt I type:

```
% plot
```

A new blue terminal (labeled plot.exe) pops up with its own * prompt. Exactly as with `fit` I must tell the program the name of the datafile and then read in the data:

```
* set file LRC.dat
* read
freq(Hz) I(A) dI(A) by default this line ignored!
READ Done.
```

The x-data (frequency) ranges from 10000 to 30000; y-data (current) ranges from .002 to .012. I could set by hand those ranges (e.g., `XMIN=10000`), but its much easier to have the program make what it thinks are reasonable choices using the command:

```
* scale
```

I could now draw a border and display the datapoints, but first I tell `plot` the proper axes labels:

```
* set xlabel='Frequency (Hz)' ylabel="RMS Current (A)" title=Resonance
* border
* dpoint
```

`Plot` is to use exactly the same function as `fit`:

```
* set f(x)=k1/sqrt(k2^2+(1/(2*pi*k3*x)-2*pi*k4*x)^2)
```

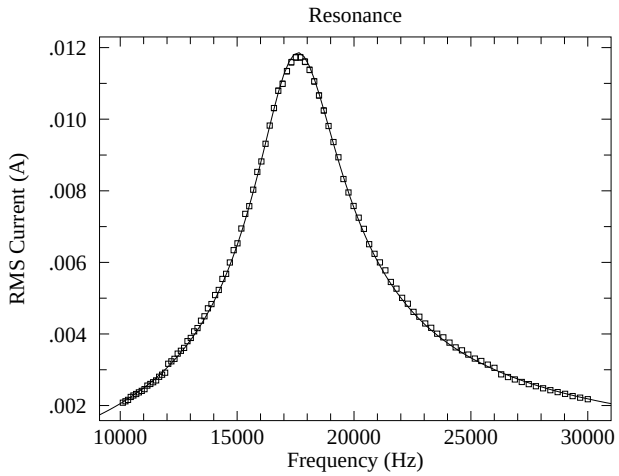
and I can copy&paste[2] from the `fit` screen the values `fit` found for those parameters:

```
* set K1= 2.262721 K2= 190.7750 K3= 0.9930065E-08
* set K4= 0.8230284E-02
```

Then I can have `plot` display that function curve and print a hardcopy:

```
* fcurve
* pcopy
```

[2]Warning repeated: *Ctrl* C kills not copies; highlight and middle-mouse for copy



If I go back to the `fit` screen and type `s` in answer to the question about matrices:

```
Display covariance/curvature matrices? No, Screen, File [N,S,F] s

 Results of a fit to the function
 f(x)=K1/SQRT(K2^2+(1/(2*PI*K3*X)-2*PI*K4*X)^2)
 at 15:15 on 13-MAY-2009
 found a REDUCED chi-square of    3.790897     and a chi-squared of    363.9261
 using the following parameter estimates
 K1=   2.262721         K2=   190.7750           K3=  0.9930065E-08
 K4=  0.8230284E-02
 and 100 data points at row   2 in the file:
 LRC.dat

 Below are displayed the covariance matrix = inverse curvature matrix and the
 curvature matrix = .5 * Hessian = .5 * matrix of 2nd partials of chi-square
 see Numerical Recipes 15.5-15.6 for a discussion of these matrices

 Matrix form:           K1 K2 K3 K4
                    K1
                    K2
                    K3
                    K4

 COVARIANCE MATRIX:
  -.55E+01  -.46E+03  0.24E-07  -.20E-01
  -.46E+03  -.39E+05  0.20E-05  -.17E+01
  0.24E-07  0.20E-05  -.11E-15  0.87E-10
  -.20E-01  -.17E+01  0.87E-10  -.72E-04

 CURVATURE MATRIX:
  0.37E+06  -.12E+04  0.34E+14  -.34E+08
  -.12E+04  0.83E+01  -.54E+11  0.63E+05
  0.34E+14  -.54E+11  0.53E+23  0.56E+17
  -.34E+08  0.63E+05  0.56E+17  0.76E+11
```

The error results are a total disaster! The usual rule is that the error in a parameter is given by the square root of the diagonal element of the covariance matrix. Thus (supposedly):

$$\delta k_1 = \sqrt{-.55 \times 10^{1}} \tag{C.2}$$
$$\delta k_2 = \sqrt{-.39 \times 10^{5}} \tag{C.3}$$
$$\delta k_3 = \sqrt{-.11 \times 10^{-15}} \tag{C.4}$$
$$\delta k_4 = \sqrt{-.72 \times 10^{-4}} \tag{C.5}$$

Clearly this is some sort of nonsense as the errors cannot be imaginary. Further note that in every case $|\delta k_i| \sim k_i$.

A hint about what's going on is gained by making a different starting guess for the values of the parameters:

```
* set k1=240 k2=.47 k3=1e-7 k4=.00086
```

Fit results show the same reduced χ^2, but with radically different parameters:

```
REDUCED chi-squared=   3.790896      chi-squared=   363.9260
K1=  0.2419768         K2=   20.40159           K3=  0.9285606E-07
K4=  0.8801518E-03
```

There is a curve of (L, R, C, V) values that produces exactly the same resonance curve. Substituting

$$L \rightarrow \alpha L \tag{C.6}$$
$$C \rightarrow C/\alpha \tag{C.7}$$
$$R \rightarrow \alpha R \tag{C.8}$$
$$V \rightarrow \alpha V \tag{C.9}$$
$$\tag{C.10}$$

into Eq. C.1 results in an unchanged function (the α cancels out).

We can rewrite our function with just three parameters:

$$I = \frac{I_0}{\sqrt{1 + Q^2 \left(\frac{f_0}{f} - \frac{f}{f_0}\right)^2}} \tag{C.11}$$

where:

$$I_0 = V/R \tag{C.12}$$
$$Q = \frac{\sqrt{L/C}}{R} \tag{C.13}$$
$$f_0 = \frac{1}{2\pi\sqrt{LC}} \tag{C.14}$$
$$\tag{C.15}$$

This reparametrization of the function is both required and easier, in the sense that initial guesses for the parameters can be immediately obtained from the raw data: $I_0 \sim .012$ A is the current at resonance, $f_0 \sim 17600$ Hz is the resonant frequency, and the quality factor Q is $f_0/\Delta f$, where $\Delta f \sim 19500 - 15800 = 3700$ is the full width at $I = I_0/\sqrt{2} \sim .0085$ A.

```
* set f(x)=k1/sqrt(1+k2^2*(k3/x-x/k3)^2)
* set k1=.012 k2=4.7 k3=17.6e3
* fit
Enter list of Ks to vary, e.g. K1-K3,K5 k1-k3
 FIT finished with change in Ks implying 4 significant figures
   2 iterations used
 REDUCED chi-squared=   3.751816      chi-squared=   363.9261
 K1=  0.1186070E-01     K2=   4.772110           K3=   17605.02
Display covariance/curvature matrices? No, Screen, File [N,S,F] s

 Results of a fit to the function
 f(x)=K1/SQRT(1+K2^2*(K3/X-X/K3)^2)
 at 16:05 on 13-MAY-2009
 found a REDUCED chi-square of    3.751816     and a chi-squared of    363.9261
 using the following parameter estimates
 K1=  0.1186070E-01     K2=   4.772110           K3=   17605.02
 and 100 data points at row   2 in the file:
 LRC.dat

 Below are displayed the covariance matrix = inverse curvature matrix and the
 curvature matrix = .5 * Hessian = .5 * matrix of 2nd partials of chi-square
 see Numerical Recipes 15.5-15.6 for a discussion of these matrices

 Matrix form:           K1 K2 K3
                     K1
                     K2
                     K3

 COVARIANCE MATRIX:
  0.52E-09  0.25E-06  -.68E-06
  0.25E-06  0.14E-03  -.77E-03
  -.68E-06  -.77E-03  0.16E+02

 CURVATURE MATRIX:
  0.13E+11  -.24E+08  -.60E+03
  -.24E+08  0.52E+05  0.15E+01
  -.60E+03  0.15E+01  0.63E-01
```

We now have reasonable results for errors:

$$\delta k_1 = \sqrt{.52 \times 10^{-9}} = 2.3 \times 10^{-5} \text{ A} \tag{C.16}$$

$$\delta k_2 = \sqrt{.14 \times 10^{-3}} = .012 \tag{C.17}$$

$$\delta k_3 = \sqrt{.16 \times 10^{2}} = 4 \text{ Hz} \tag{C.18}$$

Another way of estimating errors is to bootstrap:

```
    * boots
```

The results:

```
 MEANS
 1.187386E-02    4.77916        17605.1
 STANDARD DEVIATIONS
 2.891049E-05   1.747892E-02    8.34518
```

are similar to those above. With the largish reduced χ^2 one might want to enlarge y-error values until reduced $\chi^2 \approx 1$...`fit` calls that a `fudge`. Note that once the y-errors have been modified the only way to return to the unmodified data is to re-read the file.

```
* fudge
* fit
Enter list of Ks to vary, e.g. K1-K3,K5 k1-k3
 FIT finished with change in chi-square=  9.8335266E-02
   1 iterations used
 REDUCED chi-squared=  0.9858394      chi-squared=   95.62643
 K1=  0.1186207E-01     K2=   4.772794           K3=   17605.02
Display covariance/curvature matrices? No, Screen, File [N,S,F] s

 Results of a fit to the function
 f(x)=K1/SQRT(1+K2^2*(K3/X-X/K3)^2)
 at 16:15 on 13-MAY-2009
 found a REDUCED chi-square of   0.9858394     and a chi-squared of    95.62643
 using the following parameter estimates
 K1=  0.1186207E-01     K2=   4.772794           K3=   17605.02
 and 100 data points at row   2 in the file:
 LRC.dat

 Below are displayed the covariance matrix = inverse curvature matrix and the
 curvature matrix = .5 * Hessian = .5 * matrix of 2nd partials of chi-square
 see Numerical Recipes 15.5-15.6 for a discussion of these matrices

 Matrix form:           K1 K2 K3
                     K1
                     K2
                     K3

 COVARIANCE MATRIX:
  0.20E-08  0.94E-06  -.26E-05
  0.94E-06  0.51E-03  -.29E-02
  -.26E-05  -.29E-02  0.60E+02
```

Once the error bars have been expanded (`fudge`) this new `fit` gives a new covariance matrix from which fudged errors can be determined:

$$\delta k_1 = \sqrt{.20 \times 10^{-8}} = 4.5 \times 10^{-5} \text{ A} \quad \text{(C.19)}$$

$$\delta k_2 = \sqrt{.51 \times 10^{-3}} = .023 \quad \text{(C.20)}$$

$$\delta k_3 = \sqrt{.60 \times 10^{2}} = 7.7 \text{ Hz} \quad \text{(C.21)}$$

What is the source of the uncomfortably large reduced χ^2? If you look at the plot you'll notice that there are glitches at the two frequencies where $I = .003$ A — that is where the ammeter has automatically switched scales. A scale switch can result in different systematic error and it certainly results in a different voltage burden[3]. If we just fit to the data with $I > .003$ A we can avoid those effects. This data lies between rows 22 and 88 in the file `LRC.dat`; we can selectively have `fit` read this data:

```
* set row=22 npoint=67
* read
* print
```

Note that when you `read` all 21 rows of unwanted stuff in the file is echoed to your screen. The command `print` displays the data that is inside the program, allowing eyeballs to confirm that `fit`'s current data has $I > .003$ A. The `fit` result is:

```
REDUCED chi-squared=  0.6370068      chi-squared=   40.76844
K1=  0.1174801E-01     K2=   4.633543           K3=   17598.74
```

One can achieve almost the same thing by telling `fit` to limit its analysis to a subset block of the data it holds. When the data was originally read in, it was stored in adjacent cells labeled 1,2,3,...,`npoint`. We can tell `fit` to analyze any subset block of that data by changing which cell holds the "first" data point (`ibegin`) and the length of the run of wanted data.

```
* set ibegin=21 npoint=67
```

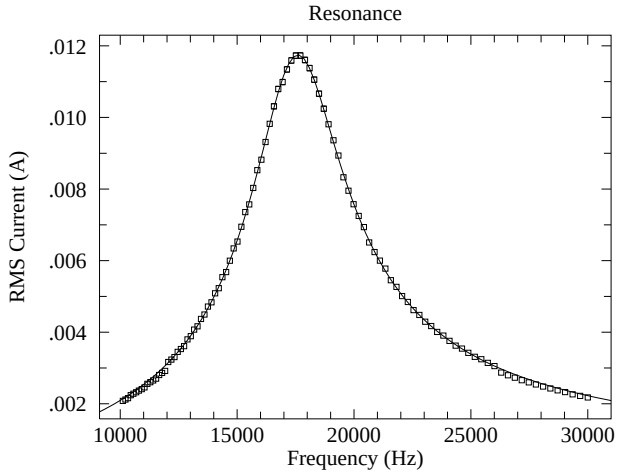
In this case we should not re-`read` the data, as the intent is to use the data already held in `fit`. (Earlier in this example changed the y-errors when we commanded `fudge`, so here we really do need to re-read the unmodified data rather than just use a subset of the modified data.)

We can use `plot` to display this new fitted curve with the entire dataset. All we need to do is change the function, set the newly found parameters k_1-k_3 via copy&paste from fit, and then redraw the plot:

```
* clear
* set f(x)=k1/sqrt(1+k2^2*(k3/x-x/k3)^2)
* set K1=  0.1174801E-01     K2=   4.633543           K3=   17598.74
```

[3] see page 83

```
* border
* dpoint
* fcurve
```



Mathematica

The program *Mathematica* can also find these nonlinear fits; of course it uses a different set of commands and data structures. You can follow along with this example if you grab a copy of the (slightly modified[4]) data file. At the *linux* `%` prompt type:

```
% cp /usr/local/physics/help/LRCm.dat .
```

and a copy of the datafile should appear in your current directory (which is called "`.`").

There are two version of *Mathematica* a command line version (`math`) and a browser version[5] (`mathematica`). The commands are the same in the two versions; the main difference is an absence of user friendly features—like arrow-key line editing, on-line help, drop-down menus—in the kernel version (`math`). I every much prefer the no-frills `math` version, but it is an acquired taste.

In a terminal, `cd` to the directory that contains the file `LRCm.dat`. Start *Mathematica* by typing to the *linux* `%` prompt:

```
% mathematica
```

or, if you're hard-core:

```
% math
```

[4] the first row of text has been removed

[5] i.e., with a Graphical User Interface—GUI, pronounced "gooey"—with menus and line editing

The file `LRC.com` that you previously copied also contains the *Mathematica* commands discussed below. Note that in `mathematica` to execute a command you need to simultaneously hit Shift and Enter, whereas in `math` it's just the usual Enter.

In *Mathematica* data is stored in set notation: `xy`=$\{\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_N, y_N\}\}$ with the errors stored in a different set (which must of course be in the same order): `ey`=$\{\delta y_1, \delta y_2, \ldots, \delta y_N\}$ To read the data into *Mathematica* in this format we need to do some rearrangement:

```
xyey=ReadList["LRCm.dat",{{Number,Number},Number}]
xy=Part[xyey,All,1]
ey=Part[xyey,All,2]
```

The following line must be typed in exactly (including case) as shown:

```
nlm = NonlinearModelFit[xy, k1/Sqrt[1+k2^2*(k3/x-x/k3)^2],{{k1,.012},{k2,4.7},
      {k3,17600}},{x},Weights->1/ey^2,VarianceEstimatorFunction->(1 &)]
```

I hope the parts make some sense to you: notice particularly how the initial parameter guesses have been entered (e.g., {`k1,.012`}). The result should look at bit like:

$$\texttt{FittedModel}[\frac{.0118607}{\sqrt{1 + 22.7729 \ll 1 \gg^2}}]$$

The symbol `nlm` now contains all the information about the fit; proper requests can extract that information:

```
In[5]:= nlm["ParameterConfidenceIntervalTable"]
Out[5]=      Estimate    Standard Error   Confidence Interval
                                          0.0118153
        k1   0.0118607   0.000022857      0.011906

                                          4.74902
        k2   4.77209     0.0116262        4.79517

                                          17597.1
        k3   17605.      3.98307          17612.9

In[6]:= nlm["CovarianceMatrix"]
                    -10             -7              -7              -7
Out[6]= {{5.2244 10    , 2.46067 10   , -7.06763 10   }, {2.46067 10   , 0.000135169,

                                 -7
>    -0.00078076}, {-7.06763 10   , -0.00078076, 15.8649}}

In[7]:= nlm["ANOVATable"]
Out[7]=                      DF    SS            MS
```

```
Model                3     1.88937 10    629790.

Error                97    363.926       3.75182

                                    6
Uncorrected Total    100   1.88974 10

Corrected Total      99    377421.
```

`ParameterConfidenceIntervalTable` gives us the fitted parameters with usual errors. As with `fit`, these estimates are the square root of the diagonal elements of the `CovarianceMatrix`. The quality of the fit is available in the `ANOVATable`: the `Error` row, the `SS` column is the χ^2 and the `MS` column is the reduced χ^2.

Mathematica needs some help to do plots with error bars. Additionally the error bars and the data need to be put together in exactly the right way:

```
Needs["ErrorBarPlots`"]
bar=Map[ErrorBar,ey]
xyEB=Transpose[{xy,bar}]
ErrorListPlot[xyEB]
Show[ErrorListPlot[xyEB], Plot[nlm[x], {x, 10000, 30000}], Frame->True]
```