Techniques and software for data analysis
PHY 312

There are many programs available for analysis and presentation of data. I recommend using Genplot, which can be downloaded for free from www.genplot.com. It can be used for fitting and other analysis tasks and for making presentation-quality plots. Online documentation for Genplot is available at http://paros.princeton.edu/genplot.html. Another common program for data analysis that some of you may already be familiar with is Matlab. Princeton has a site license for it. Mathematica can also be used, but it is not very efficient dealing with large datasets. There are also various menu-driven programs, such as Excel and Origin that can perform a limited set of tasks, but inevitably prove insufficient.

Below I will describe some typical data manipulation and analysis tasks and provide commands for Genplot and Matlab as a starting point.

Create fake data set: Gaussian noise with a standard deviation of 1, with *x* ranging from 0 to 100 in steps of 0.01:

GENPLOT
```
     create y=gnoise(x) –from 0 -to 100 –by 0.01
```

MATLAB
```
     x=(0:0.01:100);
     y=randn(length(x),1);
```

Plot data and analyze their statistical distribution:

GENPLOT
```
     plot
     eval @ave(y)
     eval @std(y)
     eval @std(y)/sqrt(npt)     – this gives the standard deviation of the average
     trans hist 0.1             - transforms data into a histogram that gives the
                                number of points in bins with a width of 0.1
     plot –hist                 – makes a plot of the histogram.
```

MATLAB
```
     plot(x,y)
     mean(y)
     std(y)
     std(y)/sqrt(length(y))
     [nbin,ybin]=hist(y,100);        - creates a histogram with 100 bins
     bar(ybin,nbin)                  -makes a histogram plot
```

We can fit the distribution to a Gaussian. Here are the steps needed to fit the data.

GENPLOT

Go to non-linear fitting mode:

```
    nlsfit
```

Define the fitting function with parameters *a*, *x0*, *s*

```
    define f(x)=a*exp(-(x-x0)^2/(2*s))
    fun f
```

Define initial values for the variables

```
    setvar a 100
    setvar x0 0
    setvar s 1
```

Specify which parameters you want to vary during the fit. You can also specify the partial derivatives of the fitting function with respect to those parameters or let the program calculate them numerically with "/" option.

```
    vary a /
    vary x0 /
    vary s /
```

If you want to remove one of the parameters from the list being varied use `remove` command

`fit` - this executes the fit command. It returns fit values of parameters and their 1-sigma errors as well as other statistical data, such as χ^2

`ov –fit –lt 1` -this will overlay the resulting fit over the data using a solid line

MATLAB

`fun=fittype('a*exp(-(x-x0)^2/(2*s))')` – defines a fitting function with *x* as an independent variable and *a, s, x0* as fitting parameters.

```
    [res,gof,out]=fit(ybin',nbin',fun,'startpoint',[100 1 0],
    'algorithm', 'gauss-newton')
```

- fitting with starting points of 100, 1, 0 for *a, s, x0* (variables in alphabetical order).

`gof` and `out` contain various statistical parameters:

```
    J=out.Jacobian
    sqrt(diag(inv(J'*J)))
    gof.sse/gof.dfe
```

`sqrt(diag(inv(J'*J)))` – returns 1-sigma errors of fit parameters

`gof.sse/gof.dfe` - returns reduced χ^2

```
    hold on
    plot(res)
```

`hold on` - overlays next plot on top of previous one

As you can see the distribution is overall well described by a Gaussian, but there are deviations in each bin from the fit. What do you think is the distribution of the number of counts in each bin of the histogram?

In the fit so far we assumed that the uncertainty is the same for each bin and did not apply any relative weighting in the sum to calculate χ^2. A more correct procedure is to use weighs corresponding to the uncertainty in each data point.

GENPLOT

```
     weight 1/(max(1,y))
```
-this sets the weight for each point, so that $\chi^2 = \sum w \times (y - f(x))^2$

$w = 1/\sigma^2 = 1/(\sqrt{y})^2 = 1/y$. When y = 0, still have uncertainty of 1, approximating Poisson distribution.

```
     fit                   
```
Now the fit is done using weighting.

```
     plot –erry sqrt(y)
```
- Plot data with errorbars

```
     ov –fit –lt 1
```
- overlay fit

MATLAB

```
     [res,gof,out]=fit(ybin',nbin',fun,'startpoint',[100 1 0],
     'algorithm', 'gauss-newton','weight',1./max(1,nbin))
     gof.sse/gof.dfe
     hold off
     plot(res,ybin,nbin)
```
–plot data, fit

```
     hold on
     errorbar(ybin,nbin,sqrt(nbin))
```
-add errorbars

Look at the quantity called reduced χ^2, which is equal to χ^2/N and should be close to 1 when proper weighting is used. The fit also gives estimates for the uncertainty in each parameter. You can check that this uncertainty is reasonable by generating several random datasets, converting them to histograms and doing the fit again. The distribution of fit parameters should itself be a Gaussian with a width given by 1-sigma error.

Data saving and retrival:

GENPLOT

```
     write test.dat
```
-this writes data in a two-column ASCII format

```
     read test.dat
```
- if you have more columns you can use `read –col 2 3` to read 2nd and 3rd columns as x and y

MATLAB

```
     data=[ybin’,nbin’]
```
– makes a data matrix, “’” transposes rows into columns

```
     save -ascii 'test.dat' 'data'
```
– writes a two-column data file

```
     load ‘test.dat’
     ybin1= test(:,1)’
     nbin1= test(:,2)’
```

Data averaging and filtering:

GENPLOT

```
     create y=gnoise(x) –from 0 -to 100 –points 1000
     arch noise
```
- saves data in a curve called “noise” for later use

```
     plot
     trans average 10      -  averages 10 adjacent points into a single point
     ov –color 2           -  Noise is reduced by averaging, overlay in red color
     retr noise
     trans fft_filter 1/(1+exp(3*(f$-1))- this applies a  low-pass filter using
                                          Fast Fourier Transform.
     ov –color 3                          - Another way of reducing noise
```

MATLAB

```
     x=(1:1000)/10;
     y=randn(length(x),1);
     plot(x,y)
     hold on
     xave=sum(reshape(x,10,100))/10;      - average 10 points into 1
     yave=sum(reshape(y,10,100))/10;
     plot(xave,yave,’color’,’r’)
     xdec=decimate(x,10);                 -“decimate” data, similar to average
     ydec=decimate(y,10);                    but with additional filtering
     plot(xdec,ydec,’color’,’g’)

     f=inline('1./(1.+exp(3.*(x-1)))')    - commands for Fourier filtering
     ffty=fftshift(fft(y));
     filffty=ffty’.*f(abs(-length(ffty)/2:length(ffty)/2-1)/
     (x(end)-x(1)));
     fily=ifft(ifftshift(filffty));
     plot(x,fily,’color’,’c’)
```

The filtering function `1/(1+exp(3*(f-1))` in frequency space is motivated by the Fermi-Dirac distribution. It gives flat response for frequencies below 1 Hz and exponential attenuation for frequencies above 1 Hz. The width of the transition region is 1/3 Hz.

Fourier Transforms

GENPLOT

```
create y=0.5*sin(2*Pi*x)+gnoise(x) –from 0 -to 100 –points 1000
     plot                                 - the sine wave is barely visible in the noise
     trans fft –magn       -this gives the fft magnitude transform (sum in quadrature
                           of real and imaginary components)
     plot                  - now you can clarly see the peak in frequency space.
```

MATLAB

```
     x=(1:1000)/10
     y=0.5*sin(2*pi*x)+randn(1000,1)'
     hold off
     plot(x,y)
     ffty=abs(fft(y))
     fftx=(1:length(x)/2)/(x(end)-x(1));
     plot(fftx,ffty(1:length(ffty)/2))
```

Note that the Fourier transform extends to 5 Hz. This is the result of the Nyquist theorem, for data with a sampling rate of 10 Hz (1000 points over an interval of 100 sec), the frequency range is equal to 10 Hz/2 (in Matlab the data above 5 Hz are a mirror image).

Digital lock-in amplifier

GENPLOT

```
create y=0.1*sin(2*Pi*x)+gnoise(x) –from 0 -to 100 –points 1000
```

Now the signal is even smaller

```
     let y 2*y*sin(2*Pi*x)
     trans fft_filter 1/(1+exp(10*(f$-0.5))
```

- fft filter with cut off at 0.5 Hz

```
     plot
     eval @ave(y)
```

This number gives the amplitude of the sine wave.

MATLAB

```
     x=(1:1000)/10
     y=0.1*sin(2*pi*x)+randn(1000,1)'
     yl=2*y.*sin(2*pi*x);
     [b,a] = butter(5,0.5*(x(2)-x(1))*2);
```

- Butterworth filter of 5th order with cut-off at 0.5 Hz

```
     yfil=filter(b,a,yl);
     plot(x,yfil)
     mean(yfil)
```

What do you think is the uncertainty in the estimate of this amplitude?

Making nice plots.

You can use menus on top of Genplot and Matlab windows to make labels for plots and change their appearance. Plots can be saved to Windows metafiles or EPS files.