PHY 480

Session 11

Pendulum Power Spectra:

1. One can determine the frequency by counting how long it takes to the frequency plot to complete n cycles (any n you like). Then simply divide the number of cycles by this time.

2. You can determine the lowest frequency by counting the number of zero crossings. I count 12 zero crossings over a time length of 50. Now, 12/50 = 0.24. You then divide by 2 (because each cycle is 2 zero-crossings) to get 0.12 which is pretty much exactly the most powerful frequency seen on the right-hand side graph. I'm not sure how to find other frequencies without doing a Fourier transform.

3. A chaotic signal will have a "broad" power signal. That is, instead of discrete spikes, one sees a near continuum of non-zero frequencies.

Multi-dimension Minimization Using GSL Routines:

1. One could create a class that takes in the function (discretely) and returns as its attributes the gradients of f.

2. The code already gives both the position and value of the minimum to an accuracy of 10^(-6).

3. Yes! See 3d_parabaloid.pdf for the result. I choose the following paraboloid:
f(x,y,z) = 10.0 * sqr(x - dp[0]) + 20.0 * sqr(y - dp[1]) + 15 * sqr(z - dp[2]) + 30.0
Centered at dp = [1,2,3].

4. Steepest descent only gets you to (x_min,y_min,z_min,f(x_min,y_min,z_min) = (1.0182673 1.9999906 3.0001328, 30.0033372) which should be (1,2,3,30). Using the vector Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm gets the exact minimum in 13 iterations, same as the Fletcher-Reeves conjugate gradient algorithm used in question 3.

Non-linear Least-Squares Fitting with GSL Routines:

1. The objective function being minimized is half the sum of the squares of the residuals which are defined as the difference of predicted and observed values divided by the error (sigma). Here, the predicted values are given by the function Ae^(-t)+b. Sigma is the error (one standard deviation) of the data y which is randomly generated from a Gaussian distribution to simulate experimental error. We divide the residuals by sigma to turn the objective function into a weighted sum where the terms with higher error matter less and vice verse.

2. The Jacobian is used to calculate the gradient vector which is used in the Taylor expansion of the model function which is supposed to approximate the objective function. Multiplying the Jacobian by a vector of differentials generalizes the chain rule for the gradient to a vector function.

3. It worked! See multi_fit_plot.pdf.

4. The uncertainty (sigma) in the i^th fit parameter is obtained by taking the square root of the (i,i)^th component of the covariance matrix. This is done in an inline function called “err”. I’m getting all zeros for the covariance matrix (perhaps something wrong with the code or my fault?) but I suspect that the uncertainties would vary proportionally with the error in the data as well as with the number of time steps.

GSL Adaptive ODE Solver Revisited:

1. No questions.

2. Yes. See the output files and the plots: plot_mu_2_x0_0.1_v0_0 and plot_mu_2_x0_1_v0_0.

3. Yes there still appears to be an isolated attractor. See plot1, plot2, and plot3.