

PHY480: Activities 11

Online handouts: Pendulum power spectrum graphs, listings of `ode_test_class.cpp` and `classes`, `multifit_test.cpp` and `multimin_test.cpp`.

In this Activities, we'll take a look at nonlinear least-squares fitting with GSL.

Your goals for today:

- Take a quick look at some pendulum power spectra.
- Try out and extend the adaptive ode code with classes.
- Try out nonlinear minimization on a demonstration problem.
- Try out nonlinear least-squares fitting on a demonstration problem.

Please work in pairs (more or less). The instructors will bounce around and answer questions.

Pendulum Power Spectra [Activities 10 Follow-up]

Look at the handout with the three rows of plots showing the time dependence of a pendulum on the left and the corresponding power spectrum on the right plotted in terms of frequency = 1/period.

1. *How can you most accurately determine the single frequency in the first row from the graph on the left?*

From 0 to 150 there are almost exactly 23 peaks \Rightarrow frequency = $\frac{23}{150} = 0.153 \text{ s}^{-1}$

This lines up exactly with the plot to the right

2. Identify (and write down) two of the frequencies in the second row *from the plot on the left*. Do they agree with the plot on the right?

The largest amplitude peak has a frequency of $\frac{1}{25} = 0.04 \text{ s}^{-1}$

The smallest also has a frequency of 0.04 s^{-1} This can be seen in the plot to the right

3. *What is the characteristic of the power spectrum in the third row that is consistent with a chaotic signal?*

There are no obvious frequency peaks.

Multidimensional Minimization with GSL Routines

The basic problem is to find a minimum of the scalar function $f(\text{xvec})$ [scalar means that the result of evaluating the function is just a number] where $\text{xvec} = (x_0, x_1, \dots, x_N)$ is an N-dimensional vector. This is a much harder problem for $N > 1$ than for $N=1$. The GSL library offers several algorithms; the best one to use depends on the problem to be solved. Therefore, it is useful to be able to switch between different choices to compare results. GSL makes this easy. Here we'll try a sample problem. For more details, see the online GSL manual under "Multidimensional Minimization" (linked on 6810 page).

The routines used here find *local* minima only, and only one at a time. They have no way of determining whether a minima is a global one or not (we'll return to this point later).

1. Look at the test code "multimin_test.cpp" and compare it to the online GSL documentation under "Multidimensional Minimization" (there is also a handout copy). Identify the steps in the minimization process. *What classes might you introduce for this code?*

It might be beneficial to combine the function and vectors into a class. One could also add a minimize method.

2. Compile and link the program with "make_multimin_test" and run it. Your results may differ from what is in the comments of the code. *Adjust the program so that your answer is good to 10^{-6} accuracy. What is it that is found to that accuracy? (E.g., is it the positions of the minimum or something else?)*

The tolerance seems to affect the accuracy of the positions of the mins

3. Modify the function minimized so it is a function of three variables (e.g., x, y, z), namely a three-dimensional paraboloid with your choice of minimum. Change the code so that the dimension of xvec is a parameter. (Be sure to change ALL of the relevant functions.) *Is the minimum still found?*

yes the minimum is still found.

4. *What algorithm is used initially to do the minimization? Modify the code to try steepest_descent and then one of the other algorithms. Which is "best" for this problem?*

Currently the Fletcher-Reeves conjugate gradient algorithm. I will try steepest descent and the BFGS algorithm. The steepest descent was the worst (didn't find it after 100 iterations) the Fletcher-Reeves was in between (took 18 iterations) and the BFGS was the best (only 3 iterations).

Nonlinear Least-Squares Fitting with GSL Routines

Using the nonlinear least-squares fitting routines from GSL is similar to using the GSL minimizer (and involves a special case of minimization). Your main job is to figure out from the documentation (see 6810 page) and this example from GSL (only slightly modified from what you will find in the documentation) how to use the routines. Here we briefly explore the sample problem, which also illustrates how to create a "noisy" (i.e., realistic) test case ("pseudo-data").

The model is that of a weighted exponential with a constant background:

$$y = A e^{-\lambda t} + b$$

You'll generate pseudo-data at a series of time steps t_i (in practice, t_i is taken to be 0, 1, 2, ...). Noise is added to each y_i in the form of a random number distributed like a gaussian with a given standard deviation σ_i . We'll revisit the GSL functions that generate this random distribution in the next Activities; for now just accept that it works.

1. Take a look at the multifit_test.cpp code (there is a printout), run it (compile and link with make_multifit_test) and figure out the flow of the program. *What is the fitting ("objective") function to be minimized? What role does σ_i play?*

The function to be minimized is $y = A e^{-\lambda t} + b$ and σ_i is the standard deviation of the random numbers that create the noise

2. *What is the "Jacobian" for? (Check the online documentation for the answer!)*

It is used to linearize around an initial guess

3. *Modify the code so that you can plot both the initial data and the fit curve using gnuplot. Look up how to add error bars for the data in gnuplot. [Hint: Try "help set style", "help errors", and "help plot errorbars" for some clues.] Did it work??.* *yes it did*
4. The covariance matrix of the best-fit parameters can be used to extract information about how good the fit is (see notes). *How is it used in the program to estimate the uncertainties in the fit parameters? How do these uncertainties scale with the magnitude of the noise? Do they scale with the number of time-steps used in the fit?* (I.e., if you change the "amplitude" of the gaussian noise, how do the errors in the fit parameters change?)

The Covariance matrix stores the error for A, lambda, & b using gsl routines. It doesn't seem to change with magnitude of error. And it also doesn't change with time steps, I wonder if there is something wrong with the code.

GSL Adaptive ODE Solver Revisited

In Activities 10, we took a quick look at `ode_test.cpp`, which implemented the GSL adaptive differential equation solver on the Van der Pol oscillator. Here we look at a rewrite that uses classes.

1. The details of `ode_test_class.cpp` and the `Ode` and `Rhs` classes are described in detail in the Activities 11 notes. Look at the printout while you go through the notes. *What questions do you have?*

None

2. Add two additional calls to `evolve_and_print` so that all three initial conditions from Activities 10, $[x_0=1.0, v_0=0.0]$, $[x_0=0.1, v_0=0.0]$, and $[x_0=-1.5, v_0=2.0]$, are generated with the same run. *Did the three output files get generated? (You can try plotting to check correctness.)*

yes there were three

3. Add another instance of the `Rhs_VdP` class called `vdp_rhs_2` with $\mu=3$ and generate results for the same initial conditions. *Plot these. Is it still an isolated attractor?*

yes there is

4. [Bonus. Come back to this if you finish everything else.] Add another class `Rhs_Pendulum` that implements the pendulum differential equation with natural frequency ω_0 and a driving force with amplitude f_{ext} and frequency ω_{ext} . (You will want to modify or replace `evolve_and_print`.)