Least Squares Curve Fitting

HW 8: Wednesday, Oct. 31, 2018
DUE: Wednesday, Nov. 7, 2018
READ: Numerical Recipes in C++, Section 2.0-2.3, pages 37-56, solving matrices,
Section 15.0-15.4 pages 773-799 (linear least squares)
OPTIONAL: Section 15.5 on nonlinear least squares
OPTIONAL: Landau, Paez, and Bordeianu: section 8.7

1. **FINAL PROJECT OUTLINE DUE ON FRI. NOV. 16, 2018**: Please write a one page outline of your final project. This should include a title and a few paragraphs and equations outlining your project goals and how you expect to accomplish them. You should also include at least one reference. Your outline must be approved before turning in your final project report. You may start to work on the project before your outline is approved but you may need to change your project if the outline is not accepted. Also, if this is a joint project (with another course) you should state this in your outline and on your final report.

2. **FINAL PROJECT DUE ON WED. DEC 12, 2018** (during exams): Please read the previous handout. This will be graded out of 20 points total. Because of the tight deadlines for senior grades etc. there will be a 2 point/day penalty for late reports. If you have another exam on this day your project is due the next day you do not have an exam (please see me in advance). Please plan to work on your project well in advance. If you have registered S/U for this course, you do not have to turn in a final project or outline if you have turned in all homework and have an average of 8/10 or better.

PROBLEM (10 points):

The measured values of atmospheric carbon dioxide measured at La Jolla Pier, California are tabulated at `http://cdiac.ornl.gov/ftp/trends/co2/ljo.dat`. The parent site, `http://cdiac.ornl.gov/trends/co2/sio-keel.html` has more explanation of this data and other interesting tabulated data about global atmospheric changes. These values were measured in parts per million (ppm) and are the average value per month since about 1969. A graph of the raw data is given below in figure 1. There is an obvious seasonal variation in the data (i.e. with a period of 1 year, possibly from plant respiration). The goal for this homework is to remove this seasonal variation and extract the underlying trend.

Download the data file mentioned above and save in a disk file. The data file is a little difficult to read in from a program because it is formatted for people and not computers. Negative values (if any) indicate that no data was recorded that particular month. A sample program to read in the data file is given below. It uses two new autosizing container classes in the STL. string and vector. These have a little extra CPU overhead but are very convenient for this task. To verify that you have the correct data, there should be 467 points with a range of 321.49 to 388.91.

Write a program that reads in the data from a file and performs a linear least squares fit to this

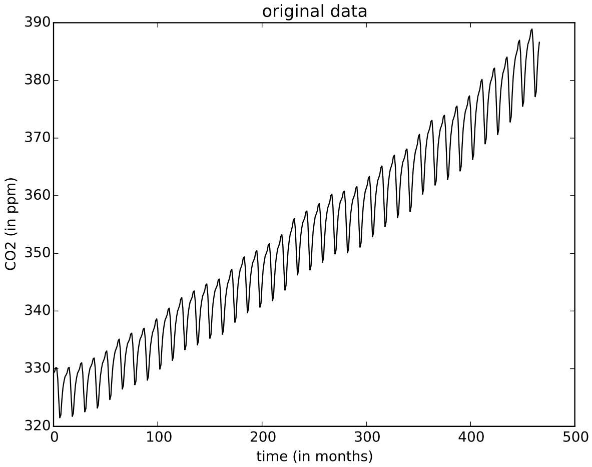

Figure 1: Input data to fit. CO_2 concentration at La Jolla Pier, California since 1969.

data assuming an error of 0.20% in each y value. The yearly variations are not well defined but can be approximated by a sine wave with a period of one year (12 months) and its first harmonic (with a period of 6 months). This form is rather arbitrary and may not fit well but should be adequate for recovering the underlying trend. We do not know the relative phase of either of these sine waves so you should include both a sin() and cos() function for each of these oscillating components. The underlying trend should be approximated as a quadratic in time. There should be a total of seven coefficients to fit (four sin/cos terms and three terms for the quadratic). Use the independent variable in units of months and the dependent variable in units of ppm of CO_2.

Calculate the reduced Chi-squared for your fit and plot the original data plus the fit and another curve without the seasonal variations. Compare the fit to the original data by plotting the residual or difference between fit and data. Is this a good fit? You should use a general purpose subroutine that performs Gauss-Elimination (you may use gaussj() from Numerical Recipes modified for arrayt or using Num. Rec. data classes). Tabulate your final best-fit parameters and the errors in these parameters (from the inverse matrix, returned by gaussj).

And finally, compare the reduced Chi-squared for the fit with and without the first harmonic (6 month) terms (i.e. a 7 parameter fit vs. a five parameter fit). Which is better?

OPTIONAL-1: Try one of the more sophisticated matrix solving routine in section 2.0-2.3 of Numerical Recipes, or in the armadillo or Eigen packages if available.

OPTIONAL-2: Try doing the fit in Matlab or Python and compare to your results in C++.'

Linear Least Squares Curve Fitting

The least squares fitting procedure follows that described in section 15.4 of Press[1] et al. If the raw data is represented by (t_i, y_i) and the error for each y_i is $\sigma_i (i = 1, 2, \cdots, N)$ then a "best fit" of a linear combination of fitting functions:

$$f(t, \vec{a}) = \sum_{k=1}^{m} a_k f_k(t) \tag{1}$$

is defined as minimizing the total reduced chi-square figure-of-merit:

$$\chi_r^2 = \frac{1}{N-m} \sum_{i=1}^{N} \left[\frac{y_i - f(t_i)}{\sigma_i} \right]^2 \tag{2}$$

where N is the number of data points and m is the number of fitting parameters. The quantity $N - m$ is the number of degrees of freedom. This problem reduces to solving the following matrix equation for the a_i coefficients:

$$F\vec{a} = \vec{b} \quad ; \quad \vec{a} = (a_0, a_1, a_2, \cdots, a_{m-1})^T \tag{3}$$

where:

$$F_{lk} = \sum_{i=1}^{N} \frac{f_l(t_i) f_k(t_i)}{\sigma_i^2} \qquad \text{and} \qquad b_l = \sum_{i=1}^{N} \frac{y_i f_l(t_i)}{\sigma_i^2} \tag{4}$$

Find the parameters $\vec{a}$ by solving the matrix equation. The errors in the parameters are the diagonal elements of F^{-1}, as;

$$\sigma_{a_k} = \sqrt{F_{kk}^{-1}} \tag{5}$$

Gauss-Jordan Elimination (without pivoting)

This method is not described very well in the text, so it is listed here. Gauss elimination is a simple method of solving the matrix equation:

$$A\vec{x} \;=\; \vec{b} \quad \text{or} \quad \sum_{j=1}^{N} a_{ij} x_j = b_i \;\; \text{for } i = 1, \ldots N$$

to find x given A and b. A is an N by N matrix and x and b are vectors of length N. Gauss elimination is a reasonably efficient and simple method but is not the most efficient. It can be summarized as:

```
for $i = 1, 2, 3, \ldots, N$ (loop over all rows)
    $b_i \leftarrow b_i / a_{ii}$
    $a_{ij} \leftarrow a_{ij} / a_{ii}$ for $j = N, \ldots, i$          (make 1 on diagonal)
    for all $j \neq i$ (loop over all other rows)
        $b_j \leftarrow b_j - a_{ji} b_i$
        $a_{jk} \leftarrow a_{jk} - a_{ji} a_{ik}$ for $k = N, \ldots, i$     (make 0 in this col. in all other rows)
    end $j$
end $i$
```

On exit the vector b has the solution for x and A is the unit matrix (i.e. it is destroyed). This assumes that there are no zero's on the diagonal and the matrix is not singular. If the diagonal elements are too small then there may also be some numerical round-off error problems. These can be fixed using pivoting as discussed in the ”Numerical Recipes”. Subroutine gaussj() in Numerical Recipes adds pivoting (good) and a clever trick to get the inverse matrix. gaussj() (or equivalent) should be used for this homework problem.

References

[1] W. H. Press, S. A, Teukolsky, W. T. Vetterling and B. P. Flannery *Numerical Recipes, The Art of Scientific Computing, 3rd edit.*, Camb. Univ. Press 2007.

[2] J. Orear, ”LeastSquares when both Variable have Uncertainties”, Amer. J. Phys. 50 (1982) p.912-916, and 52 (1984) p.278.

[3] M. Lybanon, ”A Better Least-Squares Method when both Variables have Uncertainties” Amer. J. Phys. 52 (1984) p.22-26, and 52 (1984) p.276.

Sample Code to Read the Data File

```
/* AEP 4380 Homework # 8a

Test program to read in CO2 data file from La Jolla Pier, California from
http://cdiac.ornl.gov/ftp/trends/co2/ljo.dat

run on core i7 with g++ 4.8.1 under Win7
E. Kirkland 26-oct-2018
*/

#include <cstdlib>
#include <cmath>

#include <iostream> //  stream IO
#include <fstream>  //  stream file IO
#include <iomanip>  //  to format the output
#include <string>// STD strings
#include <vector>// STD vector class
```

```
using namespace std;

int main()
{
int i, j, npts, year, t, nval;
double co2, ymin, ymax;

// use dynamically sized container classes
string cline;
vector<double> x, y;

ifstream fp; // input file stream

fp.open( "ljo.dat" );
if( fp.fail() ) { cout << "Can't open file."<< endl; exit(0); }

//--------- read data from file in complicted format -------
// skip first 16 lines

for( i=0; i<16; i++) getline( fp, cline ); // read a whole line

t = 0; // time in months
npts = 0; // number of data points
ymin = 1000.0;
ymax = -ymin;
for( i=0; i<70; i++) {
fp >> year;
cout <<  "year = " << year << endl;
if( 0 == i ) nval = 11; else nval = 12;  // line is short(?)
for( j=0; j<nval; j++) {
fp >> co2;
if( co2 > 0.0 ) {
x.push_back( t ); // use auto sizing because we don't
y.push_back(co2); // know how many elements there will be
if( y[npts] > ymax ) ymax = y[npts];  // x and y index like an array
if( y[npts] < ymin ) ymin = y[npts];  // could use co2 here also
npts++;
}
cout << "t= " << t << ", co2= " << co2 << ", npts= " << npts << endl;
t += 1;
}
if( year >= 2007 ) break; // end of file
getline( fp, cline ); // read rest of line
}

cout << "y.size() = " << y.size() << endl;  // example of size(); should be same as npts
cout << "Total number of points = " << npts << endl;
cout << "  with range " << ymin << " to " <<  ymax << endl;

return( EXIT_SUCCESS );

}  /* end main */
```