Christopher Czaban : czaban@th.physik.uni-frankfurt.de

Mathias Neuman : neuman@th.physik.uni-frankfurt.de

Final Project: χ^2 Data Fitting

1 Introduction

The goal of this project is to write a program that fits a function to a given set of data points depending on a set of parameters. When fitting a function to data points the usual approach is to define a merit function that measures the agreement between the data and the model given a particular choice of parameters. Conventionally a merit function is designed in such a way that its minimum represents close agreement. In order to find this minimum the parameters of the fit model have to be adjusted properly which is a task of minimization.

Suppose we want to fit a function with M adjustable parameters to N data points (x_i, y_i), $i = 1 \dots N$, i.e. a polynom of degree $M-1$. A simple example for such a problem is the Least-Squares-Fit for which the function to be minimized is given by

$$\mathcal{M} = \sum_{i=1}^{N} \left(y_i - y(x_i; a_1 \dots a_M)\right)^2 , \tag{1}$$

where $y(x_i; a_1 \dots a_M)$ is the function we want to fit to the data, depending on the adjustable parameters a_i. The function $y(x_i; a_1 \dots a_M)$ e.g. could be given by

$$y(x) = a_0 + a_1 x + a_2 x + \dots \quad . \tag{2}$$

Since data typically never exactly fits a model one is not purely interested in the best fit parameters (which minimize the merit function) but also wants to estimate the uncertainties in the determination of these parameters. Lets assume that each data point to be fitted has a known error σ_i (i.e. from the uncertainty when measuring this value in an experiment). To incorporate this error each term in the sum of the right side of equation (1) is divided by σ_i. Thus the following equation is obtained,

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - y(x_i; a_1 \dots a_M)}{\sigma_i} \right)^2 , \tag{3}$$

called the "chi-square".

In order to minimize the χ^2-function we take the derative with respect to the M parameters a_k which yields

$$0 = \sum_{i=1}^{N} \left(\frac{y_i - y(x_i)}{\sigma_i^2} \right) \left(\frac{\partial y(x_i; \dots a_k \dots)}{\partial a_k} \right) , \qquad k = 1, \dots, M . \tag{4}$$

In the following this equation has to be adapted for each particular problem.

2 Linear Regression

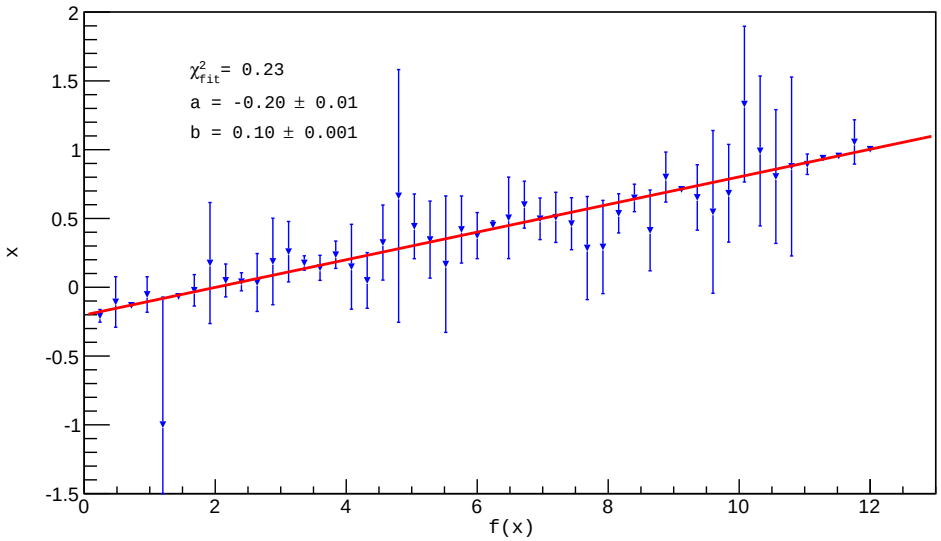

Figure 1: Linear Regression for $f(x) = a + b \cdot x$

For now let us consider the simple case of linear regression, which is simply fitting Data to a straight line. Thus the model function reads

$$y(x) = y(x; a, b) = a + bx \; . \tag{5}$$

In this case the minimization of χ^2 is straightforward and does not require any kind of sophisticated algorithms. At its minimum the derivatives of $\chi^2(a, b)$ with respect to the parameters a and b vanish:

$$0 = \frac{\partial \chi^2}{\partial a} \; , \qquad 0 = \frac{\partial \chi^2}{\partial b} \; . \tag{6}$$

In order to obtain the error estimates on the parameters a and b consider error propagation which is given by

$$\sigma_a = \sum_{i=1}^{N} \sigma_i \left(\frac{\partial a}{\partial y_i} \right)^2 \; , \qquad \sigma_b = \sum_{i=1}^{N} \sigma_i \left(\frac{\partial b}{\partial y_i} \right)^2 \; . \tag{7}$$

2.1 Tasks

- Given these equations implement a χ^2-fitting procedure that fits equation (5) to a set of data points in order to obtain a, b as well as the error estimates. Afterwards use the fitted parameters to compute the reduced chi-square value χ^2/ν where ν represents the degrees of freedom and is given by $\nu = N - M$ (number of data points minus number of free parameters). This value is known as reduced χ^2, where $\chi^2 = 1$ indicates a good fit.
- Fit the function to the set of data points provided here: `http://th.physik.uni-frankfurt.de/~mwagner/mcwagner.html`.
- Plot the data points as well as the fitted function.
- The parameters and the χ^2/ν value should be either printed in the terminal as output from the program or included in the plot.

3 General Linear Least Squares Problems

Most physical applications are more involved when it comes to fitting data and can not be handled with linear regression. The more general form of a model function has to be considered,

$$y(x; a_1 \dots a_M) = \sum_{k=1}^{M} a_k X_k(x) \ , \tag{8}$$

with M basis functions X_i, $i = 1, \dots, M$. Each X_i could be an arbitrary function of x, but for the sake of simplicity one could just assume $X_k(x) = x^k$, meaning we are fitting a polynome of the form $f(x) = \sum_{k=0}^{M} = a_k x^i$.
The corresponding χ^2-function to be minimised then reads

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - \sum_{k=1}^{M} a_k X_k(x_i)}{\sigma_i} \right)^2 \ . \tag{9}$$

This time, in order to minimize χ^2, we have to use a more sophisticated algorithm. One possibility is to solve the problem by reformulating it in terms of normal equations:
Specializing equation (4) to the case of (8) we obtain

$$0 = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[y_i - \sum_{j=1}^{M} a_j X_j(x_i) \right] X_k(x_i) \ , \qquad k = 1, \dots, M \ . \tag{10}$$

Since in the subsequent discussion the above problem will be reformulated as a matrix equation it is reasonable to introduce some notations. Let A be a $N \times M$ matrix whose components are constructed of the M basis functions $X_j(x_i)$, evaluated at the N positions x_i, divided by the standard deviations x_i:

$$A_{ij} = \frac{X_j(x_i)}{\sigma_i} \ . \tag{11}$$

Let us also define a vector $\vec{b}$ of length N by

$$b_i = \frac{y_i}{\sigma_i}, \tag{12}$$

which represents the result vector of the matrix equation. The solution vector is given by $\vec{a}$, accordingly a_i and contains the parameters to be fitted. The following rearrangement leads to the matrix equation.

$$0 = \sum_{i=1}^{N} \frac{y_i X_k(x_i)}{\sigma_i^2} - \sum_{j=1}^{M} \sum_{i=1}^{N} \frac{X_j(x_i) X_k(x_i)}{\sigma_i^2} a_j \ , \tag{13}$$

$$\Rightarrow \quad \underbrace{\sum_{i=1}^{N} \frac{y_i X_k(x_i)}{\sigma_i^2}}_{\beta_k} = \sum_{j=1}^{M} \underbrace{\sum_{i=1}^{N} \frac{X_j(x_i) X_k(x_i)}{\sigma_i^2}}_{\alpha_{kj}} a_j \ . \tag{14}$$

Finally with

$$(\beta) = A^T \vec{b} \, , \qquad (\alpha) = A^T A \tag{15}$$

one obtains the normal equations

$$(A^T A) \vec{a} = A^T \vec{b} \, . \tag{16}$$

Note that the dimension of $(A^T A)$ is $M \times M$. This equation can be solved for the vector $\vec{a}$ which contains the fit parameters a_k in (8). Thus the χ^2 minimization has been reformulated into a problem of solving a system of linear equations which can be accomplished e.g. by methods like Gauss-Jordan or LU decomposition [1]. At this point we are not done, since we still have to compute the uncertainties for the fit parameters.

[1] Numerous descriptions of the algorithm including examples for implementations in the programming language C can be found on the internet.

3.1 Computation of Error Estimates

Principally we use error propagation again (c.f. equation (7)):

$$\sigma^2(a_j) = \sum_{i=1}^{N} \sigma_i^2 \left(\frac{\partial a_j}{\partial y_i} \right)^2 . \tag{17}$$

This time the evaluation of equation (17) is a little bit more complicated. In order to be able to take the derivatives of a_j with respect to y_i we have to invert equation (16)

$$\vec{a} = (A^T A)^{-1} A^T \vec{b} \, . \tag{18}$$

Using equations (13) - (16) and denoting $(A^T A)^{-1} = C$ we can express the above equation in the following way:

$$a_j = \sum_{k=1}^{M} C_{jk} \left[\sum_{i=1}^{N} \frac{y_i X_k(x_i)}{\sigma_i^2} \right] . \tag{19}$$

Having derived this expression the derivatives of a_j in equation (17) can be computed. Further evaluation will reduce equation (17) to

$$\sigma^2(a_j) = C_{jj}. \tag{20}$$

This means the diagonal elements of the covariance matrix C are the variances (squared uncertainties) of the fit parameters a_j.
In summary in order to calculate the errors for the fit parameters the matrix $A^T A$ has to be inverted (e.g. Gauss-Jordan, LU decomposition) to obtain the covariance matrix. The variances are given by the diagonal of this matrix. $\sigma^2(a_j) = C_{jj}$.

The final task will be to implement this procedure for a simple physical application, namely the static quark antiquark potential.

4 The Static Quark Antiquark Potential

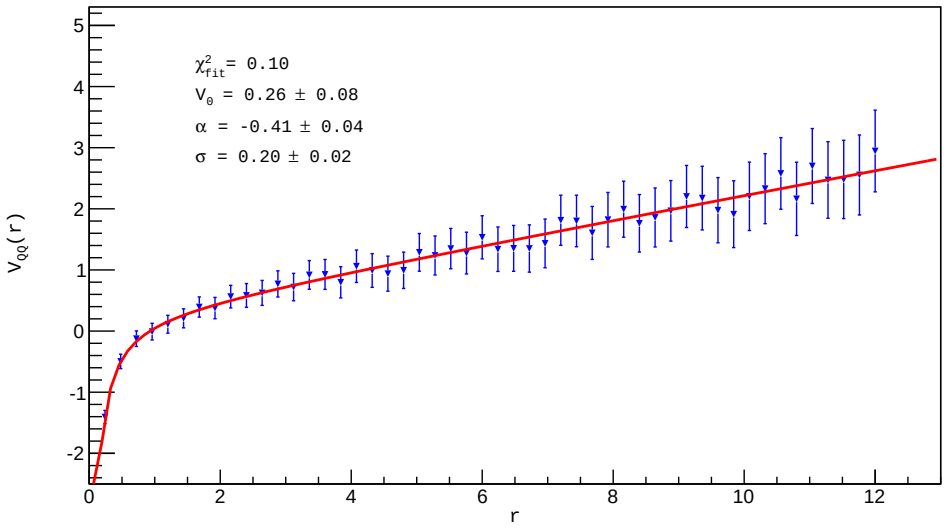

Figure 2: Static $\bar{q}q$ Potential: χ^2 minimizing fit for $V_{\bar{q}q(r)} = V_0 + \alpha/r + \sigma \cdot r$

One of the most characteristic phenomena in Quantumchromodynamics is confinement. This means that quarks and gluons can not be isolated and do only occur in bound states. In sophisticated computer simulations it can be shown that the potential between two static quarks is roughly a linear function of the distance. Contrary to gravitation or electromagnetism the force between the two quarks remains constant. However, since not only the strong force is involved in this process but also the coulomb force, at shorter separations the latter dominates and the potential is best described by a coulomb part. In summary the static potential can be parametrized by the following equation:

$$V_{\bar{q}q}(r) = V_0 + \frac{\alpha}{r} + \sigma r \ , \tag{21}$$

where the second term on the right side represents the so called Lüscher term ($\propto 1/r$) and the third term ($\propto r$) stems from the strong interaction.

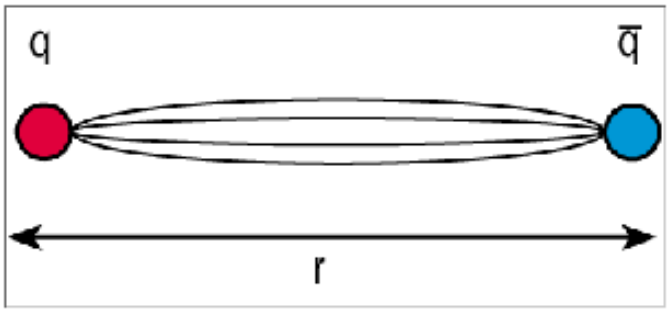

Figure 3: source: http://cerncourier.com/

4.1 Tasks

- On the basis of the theoretical foundations explained in the previous section apply a χ^2-minimizing fit of equation (21) to the provided data points (`http://th.physik.uni-frankfurt.de/~mwagner/mcwagner.html`).
- Compute the covariance matrix in order to obtain the error estimates on the fitted parameters V_0, α and σ.
- Don't forget to compute the χ^2/ν value!
- Plot the data points as well as the fitted function.
- The parameters and the χ^2/ν value should be either printed in the terminal when running your program or included in the plot.

5 General Hints for the Tasks

Static $\bar{q}q$ Potential

- The χ^2 minimization in this case is based on solving a matrix equation. For this reason the first step could be to separately implement a procedure to solve a matrix equation. As already mentioned above this can be e.g. Gauss-Jordan or LU decomposition. Another essential part that should be included in a stable algorithm for solving systems of linear equations is pivoting.
- The next big step is to invert the matrix A^TA what can also be done with the above mentioned algorithms.
- When the above tasks are accomplished the main work is done and one can think about how to incorporate the procedures into the fitting program.
- You should also think about how to construct the matrices A, A^T and A^TA (c.f. equation (11)). For example you can directly construct A and A^T by filling them with the data points from the provided file by e.g. looping over the rows of the file with a *while*-loop or a for-loop. Afterwards you can construct A^TA by a simple matrix multiplication.