Report on polynomial model least-square fitting

Dominik Brandstetter (1605547)
Department of Physics, Faculty of Natural Sciences, Graz University
(Dated: November 17, 2020)

As part of the exercise "Methods of Modeling and Simulation" (20W PHM.204UB), I present a report on the results of polynomial least-square fits to different data series as well as the code to do so. The calculation includes linear and square models, however, the code itself is capable of least-square fitting any linear sum of nearly arbitrary functions. The fits will be examined by various fitting criteria like the variance-covariance matrix and χ^2 hypothesis tests.

1. Introduction

Least-square fitting is arguably one of the most powerful tools in analyzing experimental data. It postulates a model function $f(x)$ defining the relationship between the measured data points and the scale they were measured on (e.g. time, position, ...). Predicting the observed data points by usage of this model allows us to characterise the model with a cost function χ^2 defined by the square of weighted differences between observation and prediction. This gives us a framework to improve the model by minimizing this cost function.

In section 2 we will briefly discuss the theory of least-square fitting and further develop the ansatz by restriction to linear models. We will also take a look at how to assess the model we calculated by error analysis. This includes the variance-covariance matrix as well as goodness-of-fit hypothesis tests.

After briefly discussing the simulation procedure in section 3 the results will be documented in section 4 and discussed in section 5. The code and the plots can be found in the appendix B and A respectively.

2. Theory

2.1. Least-Square Fitting

We start with N data points $\{y_i\}$ for which we assume a functional dependency on some variable x

$$y_i = f(x_i)$$

We further assume $\{y_i\}$ to be subject of individual Gaussian-shaped measurement errors $\{\sigma_i\}$. The goal now is, given a certain model for $f(x)$ with M degrees of freedom, to find the one fitting our data best. This naturally leads to the least-square model[1]

$$\chi^2 = \sum_i^N \left(\frac{y_i - f(x_i)}{\sigma_i} \right)^2 \tag{1}$$

Minimizing the sum over the square differences between what we observed y_i and what we predict $f(x_i)$ weighted by the standard deviation σ_i^2 will yield the best possible set of parameters $\{a_j\}$ for our model.

2.2. Linear Problems and Design Matrix A

We did not yet make any restrictions on our model $f(x_i)$, however, to further investigate (1) we will assume a linear model, i.e. linear dependence on the parameter set $\{a_j\}$. This means we can split $f(x_i)$ into a sum of functions $X_j(x_i)$

$$f(x_i) = \sum_j^M X_j(x_i)\, a_j$$

Finding the minimum of χ^2 requires us to derive in respect to a_j

$$\begin{aligned} &\frac{\partial \chi^2}{\partial a_m} \stackrel{!}{=} 0 \\ &\Rightarrow \frac{y_i - f(x_i)}{\sigma_i^2} X_m(x_i) \stackrel{!}{=} 0 \\ &\Rightarrow \frac{1}{\sigma_i} y_i \stackrel{!}{=} \frac{1}{\sigma_i} \sum_j^M a_j X_j(x_i) \end{aligned} \tag{2}$$

which we can easily identify as the following matrix equation

$$\vec{y}_w = A\vec{a}; \quad (\vec{y}_w)_i = \frac{1}{\sigma_i} y_i \tag{3}$$

A is a $N \times M$ matrix we call "design matrix" or sometimes "data matrix". The solution of (3) will then be our optimised parameter set $\vec{a}$. However, in general, A will not be a square matrix and as such not invertible. Fortunately, the generalised inverse $A^{-g} = \left(A^T A\right)^{-1} A^T$ (also called "pseudoinverse") will suffice as

$$\begin{aligned} \vec{a} &= \left(A^T A\right)^{-1} A^T \vec{y}_w \\ A^T A\vec{a} &= A^T \vec{y}_w \\ A\vec{a} &= \vec{y}_w \end{aligned}$$

2.3. Model Analysis

2.3.1. Variance-Covariance Matrix

The variance-covariance matrix C_a has the variance of each parameter on its diagonal and the correlation between them on the off-diagonal elements. It can be, without proof[1], calculated directly from the design matrix A

$$C_a = \left(AA^T\right)^{-1} \tag{4}$$

C_a is symmetric therefore symmetric, as is to be expected. Normalising the covariance matrix gives us R, the correlation matrix

$$(R)_{ij} = \frac{\sigma_{ij}^2}{\sigma_{ii}\sigma_{jj}} \tag{5}$$

It is, of course, also symmetric and has only 1 on its diagonal since every value correlates completely with itself.

2.3.2. Hypothesis Test

To quantify how good our model fits the data we can perform a hypothesis test. In our case we will test against the χ^2-distribution $f(x, f)$, an asymmetric distribution with an additional parameter g, the degrees of freedom. We state as null hypothesis H_0, that our sum of weighted square differences χ^2, equation (1), follows this distribution. Thus, by integrating over the χ^2-distribution up to our χ^2

$$\alpha = \int_0^{\chi^2} f(x, g)\, dx \tag{6}$$

we get the likelihood α with which we would have expected a weighted sum of differences smaller or equal χ^2 if the model fits the data. $p = 1 - \alpha$ value is therefore the percentage we are willing to reject the null hypothesis, even though it was correct. The higher we choose p, the lower the threshold χ^2_{test} our sum of differences can't cross or we would reject the model. Therefore p is a common criterion to assess the significance and a $p > 0.05$ is widely accepted to be a significant result.

3. Simulation

The fitting code, see appendix B, was entirely written in Python 3.8.5 utilising the following packages: NumPy 1.19.1 to handle vector-shaped data and its "linalg" module to perform matrix opertions, SciPy 1.5.2 for the χ^2-distribution and MatPlotLib 3.3.1 for plotting. The code was used to fit three sets of data (example 1, 2 and 3) provided by the exercise sheet. Please refer to it for more details on the data sets.

4. Results

Each subsection contains the parameter results for the fits, the variance-covariance matrix and the correlation matrix and the results from the hypothesis test. For the plots please see appendix A.

4.1. Example 1

TABLE I: Parameter results for example 1.

Parameter	Value x
Linear Term x_1	1.01 ± 0.06
Constant Term x_0	0.8 ± 0.4

$$C_a = \begin{pmatrix} 0.132 & -0.02 \\ -0.02 & 0.04 \end{pmatrix}; \quad R = \begin{pmatrix} 1 & -0.889 \\ -0.889 & 1 \end{pmatrix}$$

$$\chi^2 = 10.32; \quad \chi^2_{\alpha=0.95} = 14.07; \quad p = 0.171$$

4.2. Example 2

4.2.1. Quadratic Fit

TABLE II: Parameter results for quadratic fit for example 2.

Parameter	Value x
Second Order Term x_2	0.84 ± 0.19
Linear Term x_1	0 ± 2
Constant Term x_0	6 ± 3

$$C_a = \begin{pmatrix} 11.43 & -4.79 & 0.42 \\ -4.79 & 3.06 & -0.32 \\ 0.42 & -0.32 & 0.04 \end{pmatrix}$$

$$R = \begin{pmatrix} 1 & -0.81 & 0.66 \\ -0.81 & 1 & -0.96 \\ 0.66 & -0.96 & 1 \end{pmatrix}$$

$$\chi^2 = 3.59; \quad \chi^2_{\alpha=0.95} = 14.07; \quad p = 0.825$$

4.2.2. Linear Fit

TABLE III: Parameter results for linear fit for example 2.

Parameter	Value x
Linear Term x_1	7.56 ± 0.47
Constant Term x_0	-4.56 ± 2.53

$$C_a = \begin{pmatrix} 6.39 & -1.01 \\ -1.01 & 0.23 \end{pmatrix}; \quad R = \begin{pmatrix} 1 & -0.84 \\ -0.84 & 1 \end{pmatrix}$$

$$\chi^2 = 23.84; \quad \chi^2_{\alpha=0.95} = 15.50; \quad p = 0.002$$

4.3. Example 3

4.3.1. Weighted

TABLE IV: Parameter results for weighted example 3.

Parameter	Value x
Linear Term x_1	10.8 ± 0.8
Constant Term x_0	0.01004 ± 0.00018

$$C_a = \begin{pmatrix} 3.21E-8 & -7.93E-5 \\ -7.93E-5 & 5.70E-1 \end{pmatrix}; \quad R = \begin{pmatrix} 1 & -0.59 \\ -0.59 & 1 \end{pmatrix}$$

$$\chi^2 = 0.28; \quad \chi^2_{\alpha=0.95} = 15.51; \quad p = 1.000$$

4.3.2. Unweighted

TABLE V: Parameter results for unweighted example 3.

Parameter	Value x
Linear Term x_1	11.2
Constant Term x_0	0.01

5. Discussion

5.1. Example 1

As is displayed in FIG.1, a linear fit seems the right approach to this data set. The data points follow a clear trend, however, with relatively large deviations. This intuition was confirmed by the $p = 0.171$ value of the linear fit. It lies well above the 0.05 threshold, which is a common criterion for significance, however, compared to the p values for other data sets and their fits, this value is comparatively low.

5.2. Example 2

In FIG.2 we compared the results for a linear and a quadratic fit for this data set. From the p values it is clear, that the quadratic approximation is a much better fit for the data set than the linear one. However, we have very large uncertainty and the confidence interval, therefore, opens up quite a bit and the predictability for large x is limited.

5.3. Example 3

For example 3 in FIG.3 we see the influence of the uncertainty in the observed data y_i on the fit. The fit without any uncertainty fits the later data points better, however, it neglects the fact that we have less confidence in their values. Without uncertainty we cannot give any p value, however, from the p value for the weighted fit we can have a very high degree of certainty in the linearity of our data.

[1] A. K. Steiner. *Methods of Modeling and Simulation*, 2020.

Appendix A

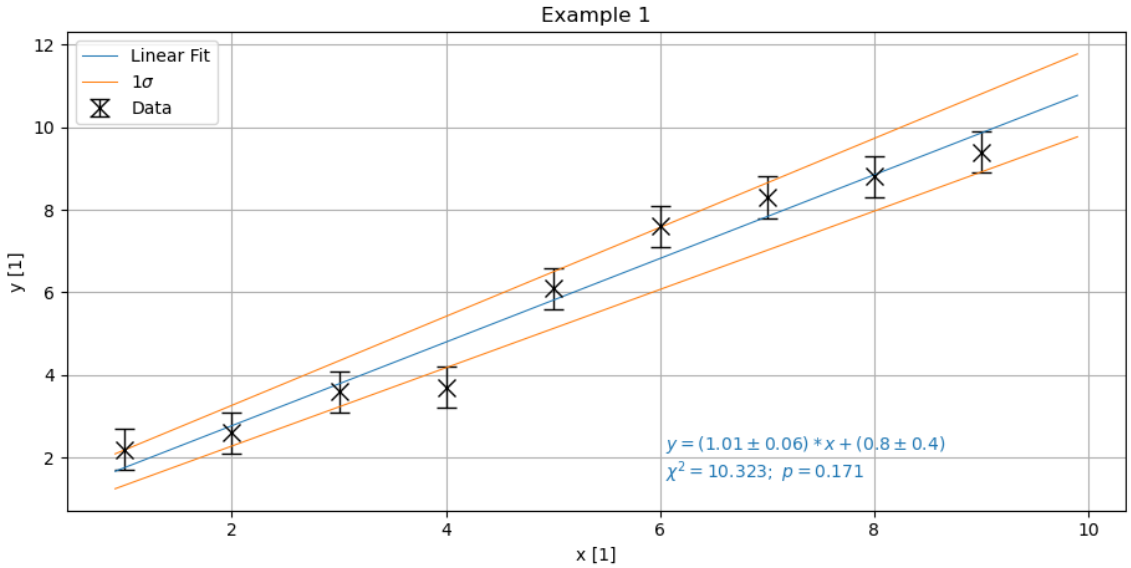

FIG. 1: The data from the example 1 displayed in black, including the uncertainty in form of error bars with one standard deviation σ. The optimal fit determined and its parameters as well as the uncertainty of said parameters with one standard deviation in blue. Additionally in blue, the χ^2 value of the fit as well as its significance expressed as a p value. The lines in orange are the confidence interval for the fit for 1σ.

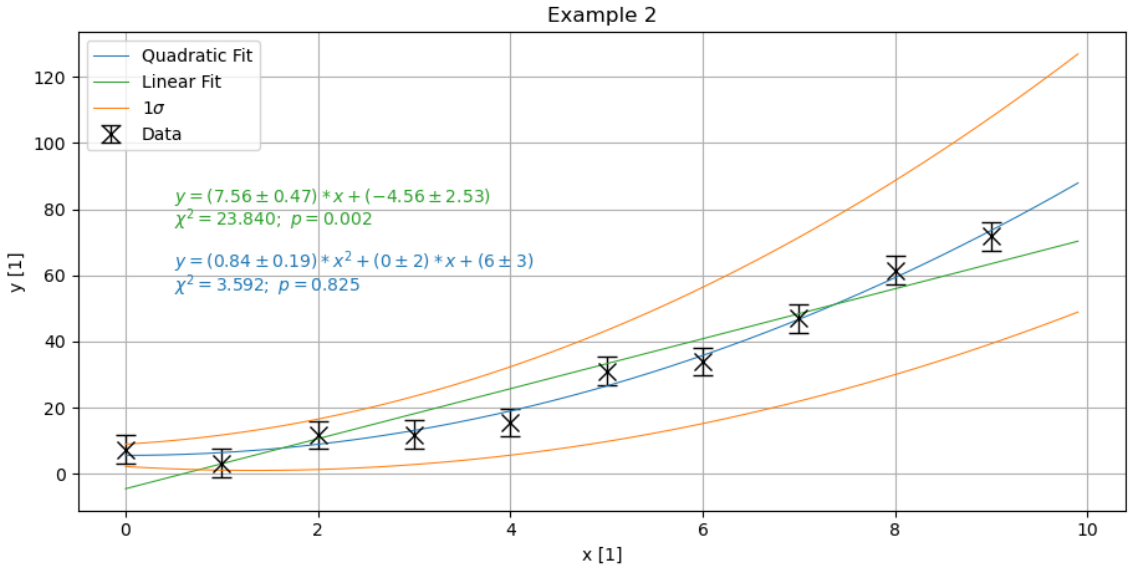

FIG. 2: The data from the example 2 displayed in black, including the uncertainty in form of error bars with one standard deviation σ. The optimal quadratic fit determined and its parameters as well as the uncertainty of said parameters with one standard deviation in blue. Additionally in blue, the χ^2 value of the fit as well as its significance expressed as a p value. The lines in orange are the confidence interval for the fit for 1σ. The optimal linear fit determined and its parameters as well as the uncertainty of said parameters with one standard deviation in green. Additionally in green, the χ^2 value of the fit as well as its significance expressed as a p value.

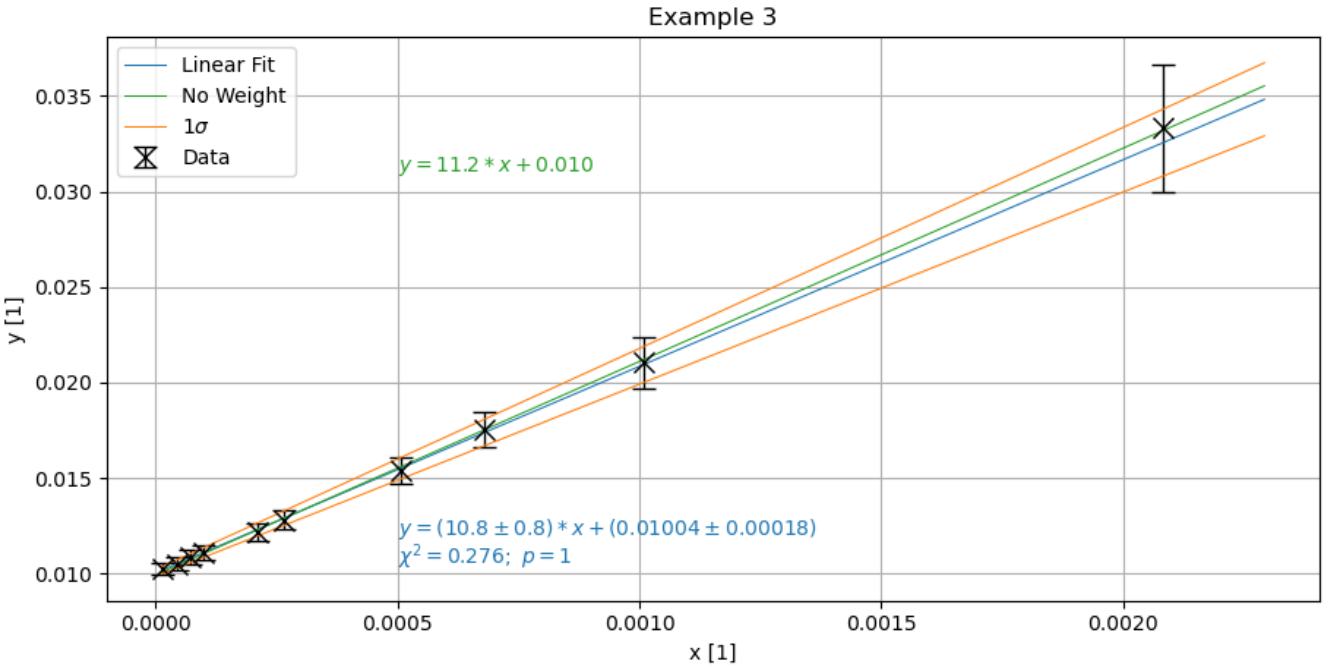

FIG. 3: The data from the example 3 displayed in black, including the uncertainty in form of error bars with one standard deviation σ. The optimal fit determined and its parameters as well as the uncertainty of said parameters with one standard deviation in blue. Additionally in blue, the χ^2 value of the fit as well as its significance expressed as a p value. The lines in orange are the confidence interval for the fit for 1σ. A linear fit with the assumption of no uncertainty in the data is displayed in green.

Appendix B

```
import numpy as np
from scipy import stats


def polynomial_fit(x, y, sigma, order=1):

    funcs = get_polynomial_funcs(order=order)

    return fit(np.array(x), np.array(y), sigma, funcs)


def get_polynomial_funcs(order=1):

    return [eval('lambda x: x**{}'.format(power))
            for power in range(order + 1)]


def fit(x, y, sigma, funcs):

    # Design Matrix A
    A = get_design_matrix(x, sigma, funcs)
    N, M = A.shape

    # Generalized Inverse Matrix
    A_inv = np.linalg.inv((A.T.dot(A))).dot(A.T)

    # Covariance and Correlation Matrices
    C = np.linalg.inv(A.T.dot(A))
    R = np.ones((len(funcs), len(funcs)))
    for i in range(len(funcs)):
        for j in range(len(funcs)):
            R[i, j] = C[i, j] / np.sqrt(C[i, i] * C[j, j])

    if sigma == 0:
        parameter = A_inv.dot(y)
        p_val, C, R = [None, None, None]

    else:
        # Optimal Parameters
        y_w = np.array(y) / sigma
        parameter = A_inv.dot(y_w)

        # Chi^2
        p_val = get_p_val(x, y, sigma, parameter, funcs)

    return parameter, p_val, C, R


def get_design_matrix(x, sigma, funcs):

    if sigma == 0:
        sigma = 1

    if isinstance(sigma, float) or isinstance(sigma, int):
        sigma = sigma * np.ones((len(x), 1))
```

```
    else:
        sigma = np.reshape(sigma, (len(sigma), 1))

    x, funcs = map(np.array, (x, funcs))

    A = np.zeros((len(x), len(funcs)))

    for col, func in enumerate(funcs):
        A[:, col] = func(x)

    A = A * (1 / sigma)

    return A


def get_p_val(x, y, sigma, parameter, funcs, alpha=0.95):

    y_mod = eval_result(x, parameter, funcs)

    chi2 = np.sum(((y - y_mod) / sigma)**2)
    deg_of_freedom = len(x) - len(funcs)
    p_val = stats.chi2.sf(chi2, df=deg_of_freedom)
    chi2_test = stats.chi2.ppf(q=alpha, df=deg_of_freedom)

    return p_val, chi2, chi2_test


def eval_result(x, parameter, funcs):

    y = []

    for p, func in zip(parameter, funcs):
        y.append(p * func(x))

    return np.sum(np.array(y), axis=0)
```