3 Unkorrelierte Fehler, bei denen die Standardabweichungen σ_i nicht bekannt, aber alle gleich σ sind: Methode der kleinsten Fehlerquadrate (non-linear least squares)

3.1 Allgemeine Behandlung

Definition des mittleren quadratischen Fehlers (mean square error). In diesem Fall ist σ ein weiterer Fitparameter. Die Stichprobenverteilung in Gl. (1) wird damit

$$\mathrm{pdf}(\{y_i\}|\{x_i\},\theta,\sigma) = \frac{1}{(2\pi)^{N/2}\sigma^N}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f(x_i;\theta))^2\right),$$

Wir definieren den mittleren quadratischen Fehler

$$\boxed{Q(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i;\theta))^2.}$$

Bemerkung 36. *Bei bekannten Wertepaaren (x_i, y_i) ist der mittlere quadratische Fehler eine Funktion der M unbekannten Parameter θ.*

Beispiel 8. *Ist das Modell der Messung $y_i = \mu + \epsilon_i$, dann ist der mittlere quadratische Fehler quadratisch im Parameter μ, nämlich*

$$Q(\mu) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)^2.$$

Graphisch beschreibt diese Funktion eine nach oben geöffnete Parabel. Damit ist klar, dass ein eindeutiges Minimum von $Q(\mu)$ existiert.

Beispiel 9. *Ist das Modell der Messung $y_i = ax_i + b + \epsilon_i$, dann ist die Funktion $Q(a, b)$ eine quadratische Form der beiden Parameter a und b. Die Konturlinien dieser Funktion sind Ellipsen und die Funktion besitzt ein eindeutiges Minimum.*

Bemerkung 37. *Ist das Modell der Messung nichtlinear in den Parametern θ, dann besitzt die $Q(\theta)$-Funktion nicht notwendigerweise ein eindeutiges Minimum.*

3.1.1 Priorverteilung der Parameter θ und σ

Indifferenzprinzip. Auch hier verwenden wir eine flache Priorverteilung (siehe 2.1.1) der Form

$$\mathrm{prob}(\theta) = \mathrm{pdf}(\theta)d\theta = \mathrm{const.} \times d\theta$$

für alle Parameter θ.

Die Priorverteilung für σ kann dementsprechend gemäss

$$\text{prob}(\sigma) = \text{pdf}(\sigma)d\sigma = \begin{cases} \text{const.} & \text{für } \sigma \geq 0 \\ 0 & \text{für } \sigma < 0 \end{cases}$$

gewählt werden. Diese Verteilung beinhaltet automatisch die Randbedingung, dass σ niemals negativ sein kann.

Bemerkung 38. *Diese Form liegt den traditionellen Methoden der Gaussschen Fehlerrechnung zugrunde und wird auch heute noch von den meisten numerischen Implementierungen implizit verwendet.*

Jeffrey's Priorverteilung. Eine modernere Form der Priorverteilung für den Parameter σ wurde von Jeffreys vorgeschlagen:

$$\text{prob}(\sigma) = \text{pdf}(\sigma)d\sigma = \begin{cases} \text{const.} \times \frac{d\sigma}{\sigma} & \text{für } \sigma \geq 0 \\ 0 & \text{für } \sigma < 0 \end{cases}$$

Bemerkung 39. *Zur klaren Kennzeichnung der Veränderung an der orthodoxen Methode, die durch die Wahl von Jeffreys' Priorverteilung verursacht wird, verwenden wir die rote Farbe des Textes.*

Bemerkung 40. *Der Vorteil von Jeffrey's Priorverteilung gegenüber der konstanten Verteilung ist ihre Skaleninvarianz.*

Bemerkung 41. *Jeffreys' Prior kann als konstante Priorverteilung auf einer logarithmischen Skala angesehen werden, denn const.* $\times\, d(\ln\sigma) =$ *const.* $\times\, d\sigma/\sigma$.

3.1.2 Posteriorverteilung für die Parameter

Bei konstanten Priorverteilungen für die θ und konstanter Priorverteilung (Jeffreys Prior) für σ ist die Posteriorverteilung, die man aus der Anwendung des Bayesschen Theorems (siehe 2.1.2) gewinnt

$$\text{pdf}(\theta, \sigma|\{(x_i, y_i)\}) \propto \frac{1}{\sigma^{N+1}} \exp\left(-\frac{NQ(\theta)}{2\sigma^2}\right).$$

Zur Abschätzung der Parameter θ und σ suchen wir das Maximum dieser Verteilung bezüglich der Parameter. Mathematisch äquivalent, aber rechnerisch angenehmer ist es, das Miminum des negativen Logarithmus der Posteriorverteilung zu bestimmen. Der negative Logarithmus dieser Verteilung ist

$$L(\theta, \sigma) = (N+1)\ln\sigma + \frac{NQ(\theta)}{2\sigma^2}. \tag{4}$$

Am Minimum $(\hat{\theta}, \hat{\sigma})$ gilt

$$\left.\frac{\partial L(\theta,\sigma)}{\partial \theta_i}\right|_{\hat{\theta},\hat{\sigma}} = \frac{N}{2\hat{\sigma}^2}\left.\frac{\partial Q(\theta)}{\partial \theta_i}\right|_{\hat{\theta}} = 0 \Rightarrow \left.\frac{\partial Q(\theta)}{\partial \theta_i}\right|_{\hat{\theta}} = 0,$$

$$\left.\frac{\partial L(\theta,\sigma)}{\partial \sigma}\right|_{\hat{\theta},\hat{\sigma}} = \frac{(N+1)}{\hat{\sigma}} - \frac{NQ(\hat{\theta})}{\hat{\sigma}^3} = 0 \Rightarrow \hat{\sigma} = \sqrt{\frac{NQ(\hat{\theta})}{N+1}}.$$

Den Wert $\hat{\theta}$ gewinnt man (meist) durch (numerisches) Minimieren von $Q(\theta)$ gemäss der ersten Gleichung.[4] Den Wert $\hat{\theta}$ nennt man den Schätzwert der kleinsten Fehlerquadrate ('non-linear least squares estimator'). Nach der Minimierung von $Q(\theta)$ sind $\hat{\theta}$ und $Q(\hat{\theta})$ bekannt, und damit gemäss der zweiten Gleichung auch $\hat{\sigma}$.

Bemerkung 42. *Besitzt die $Q(\theta)$-Funktion ein eindeutiges Minimum $Q(\hat{\theta})$ für gewisse Parameterwerte $\hat{\theta}$, so ist die Posteriorverteilung der Parameter an der Stelle $\theta = \hat{\theta}$ und $\sigma = \hat{\sigma}$ maximal.*

Python: die Funktion `scipy.optimize.curve_fit`. Das Minimieren der Funktion $Q(\theta)$ wird im Rahmen der Python Funktion `scipy.optimize.curve_fit` durchgeführt. Zu diesem Zweck wird die Funktion `scipy.optimize.leastsq` aufgerufen. Diese Funktion wiederum benutzt eine Fortran-Implementierung des Levenberg-Marquardt Algorithmus, die von Burton, Hillstrom und Moré am Argonne National Laboratory im Rahmen von Minipack entwickelt wurde. Minipack ist eine Bibliothek von mathematischen Subroutinen, die zum Beispiel bestimmte Minimierungsprobleme numerisch lösen können. Die Funktion `curve_fit` erhält als Eingabeparameter die Fitfunktion $f(x_i;\theta)$, die (x_i, y_i)-Werte aus der Messung, sowie Startwerte der Parameter für die Minimierung. Hier folgt ein Beispielcode für die Anwendung:

```
import numpy as np
from scipy.optimize import curve_fit

# define the fitting function with two parameters
def fit_func(x,p0,p1):
    return np.sqrt( 2 * (x - p1)/p0 )

# least mean square minimization
result = curve_fit(fit_func,xdata,ydata,[p0start,p1start],full_output=True)
```

[4] Ein gebräuchlicher Algorithmus ist der Levenberg-Marquardt Algorithmus, der eine Weiterentwickung des Gauss-Newton Verfahrens ist. Dieser Algorithmus kommt zum Beispiel bei der Python Funktion `scipy.optimize.curve_fit` zum Einsatz.

```
# extract the two parameter estimates
p0hat = result[0][0]
p1hat = result[0][1]

# determine the estimate for the sigma-parameter
residuals = result[2]['fvec']
NQmin = residuals.dot(residuals)
# Orthodox
sigmahat = np.sqrt(NQmin/N)
# Jeffreys Prior
sigmahat = np.sqrt(NQmin/(N+1))
```

3.2 Spezialfall 1: konstante Fitfunktion

Nehmen wir

$$f(x_i, \mu) = \mu,$$

dann ist

$$Q(\mu) = \mathrm{Var}(y_i) + (\mu - \langle y_i \rangle)^2,$$

wobei

$$\langle y_i \rangle = \frac{1}{N} \sum_{i=1}^{N} y_i$$

der ungewichtete empirische Mittelwert der Messwerte ist. Entsprechend ist

$$\mathrm{Var}(y_i) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \langle y_i \rangle)^2$$

die ungewichtete empirische Varianz der Messwerte. Damit findet man

$$\hat{\mu} = \langle y_i \rangle$$

und $Q(\hat{\mu}) = \mathrm{Var}(y_i)$, so dass

$$\hat{\sigma} = \sqrt{\frac{N}{N+1} \mathrm{Var}(y_i)}.$$

In Python berechnet man den ungewichteteten empirischen Mittelwert und die Varianz wie unten gezeigt.

```
from numpy import mean, var

Meany = mean(data[:])
vary = var(data[:])
```

3.3 Spezialfall 2: lineare Fitfunktion

Nehmen wir

$$f(x_i, \alpha, \beta) = \alpha(x_i - \langle x_i \rangle) + \beta,$$

wobei

$$\langle x_i \rangle = \frac{1}{N} \sum_{i=1}^{N} x_i,$$

dann ist

$$\begin{aligned} Q(\alpha, \beta) &= \frac{1}{N} \sum_{i=1}^{N} (y_i - \alpha(x_i - \langle x_i \rangle) - \beta)^2 \\ &= \mathrm{Var}(x_i) \left[\alpha - \frac{\mathrm{Cov}(x_i, y_i)}{\mathrm{Var}(x_i)} \right]^2 + (\beta - \langle y_i \rangle)^2 + \mathrm{Var}(y_i)(1 - \rho^2). \end{aligned}$$

Damit ist

$$\begin{aligned} \hat{\alpha} &= \frac{\mathrm{Cov}(x_i, y_i)}{\mathrm{Var}(x_i)} \\ \hat{\beta} &= \langle y_i \rangle \\ \hat{\sigma} &= \sqrt{\frac{N \mathrm{Var}(y_i)(1 - \rho^2)}{N + 1}}. \end{aligned}$$

3.4 Genauigkeit der Schätzwerte

3.4.1 Genauigkeit aus der Posteriorverteilung der Parameter

Wir bleiben bei dem in 2.4 eingeführten Prinzip, den Fehler mit der Konturfläche im Parameterraum zu beschreiben, bei der die Posteriorverteilung vom Maximum um den Faktor $e^{-1/2}$ abgenommen hat. Das entspricht der Bedingung

$$\frac{\exp(-L(\theta, \sigma))}{\exp(-L(\hat{\theta}, \hat{\sigma}))} = \exp\left(-\frac{1}{2}\right) \Rightarrow \boxed{2[L(\theta, \sigma) - L(\hat{\theta}, \hat{\sigma})] = 1.}$$

Variante I: graphische Bestimmung; geeignet bei einem Parameter. Für den Fall, dass die Fitfunktion nur einen Parameter θ enthält lässt sich die Lösung dieser Gleichung wiederum graphisch in der Ebene der Parameter θ und σ darstellen (siehe 2.4).

Variante II: Näherungsweise Bestimmung mit Hilfe der Kovarianzmatrix. Wir starten von der Posteriorverteilung für die Parameter θ und σ

$$\mathrm{pdf}(\theta, \sigma|\{(x_i, y_i)\}) \propto \frac{1}{\sigma^{N+1}} \exp\left(-\frac{NQ(\theta)}{2\sigma^2}\right).$$

Wir erhalten eine Verteilungsfunktion für die Parameter θ alleine, indem wir σ ausintegrieren (marginalisieren). Das ergibt

$$\mathrm{pdf}(\theta|\{(x_i, y_i)\}) = \int_0^\infty d\sigma \mathrm{pdf}(\theta, \sigma|\{(x_i, y_i)\}) \propto \left(\frac{Q(\hat{\theta})}{Q(\theta)}\right)^{(N-1+1)/2}$$

Wir schreiben diese Funktion in Analogie zum bisherigen in der Form

$$\mathrm{pdf}(\theta|\{(x_i, y_i)\}) = e^{-L(\theta)},$$

mit

$$L(\theta) = \frac{N-1+1}{2} \ln \frac{Q(\theta)}{Q(\hat{\theta})}.$$

Insbesondere ist dann

$$\frac{\partial L(\theta)}{\partial \theta_i} = \frac{N-1+1}{2} \frac{1}{Q(\theta)} \frac{\partial Q(\theta)}{\partial \theta_i},$$

so dass die Posteriorverteilung für θ bei $\hat{\theta}$ maximal ist und die Ableitung von L dort verschwindet. Die zweite Ableitung von L bei $\hat{\theta}$ ist dann

$$\left.\frac{\partial^2 L}{\partial \theta_i \theta_j}\right|_{\hat{\theta}} = \frac{N-1+1}{2} \frac{1}{Q(\hat{\theta})} \left.\frac{\partial^2 Q(\theta)}{\partial \theta_i \partial \theta_j}\right|_{\hat{\theta}}.$$

Wir fragen wieder nach den näherungsweise ellipsoiden Konturlinien der Posteriorverteilung und entwickeln zu diesem Zweck

$$1 = 2(L(\theta) - L(\hat{\theta})) \approx \frac{N-1+1}{2} \frac{1}{Q(\hat{\theta})} \sum_{i,j} \underbrace{\left.\frac{\partial^2 Q(\theta)}{\partial \theta_i \partial \theta_j}\right|_{\hat{\theta}}}_{H_{i,j}} (\theta_i - \hat{\theta}_i)(\theta_j - \hat{\theta}_j)$$

Die gesuchte Kovarianzmatrix K erfüllt die Gleichung

$$\frac{N-1+1}{2} \frac{1}{Q(\hat{\theta})} H = \frac{1}{2} K^{-1},$$

so dass

$$\boxed{K = \frac{Q(\hat{\theta})}{N - 1{+}1} H^{-1}.}$$

Python: Kovarianzmatrix. In Python ist die Kovarianzmatrix ein Rückgabeparameter der Funktion `curve_fit`.

```
covMatrix =  result[1]
```

Zur Bestimmung der Genauigkeit der Abschätzung von $\hat{\sigma}$ wählen wir einen analogen Weg. Wir integrieren die Parameter θ aus der Posteriorverteilung aus und erhalten so eine Verteilung für den Parameter σ alleine. Zu diesem Zweck entwickeln wir

$$Q(\theta) \approx Q(\hat{\theta}) + \frac{1}{2} \sum_{i,j} H_{ij} (\theta_i - \hat{\theta}_i)(\theta_j - \hat{\theta}_j).$$

Damit ergibt sich die Posteriorverteilung näherungsweise zu

$$\mathrm{pdf}(\theta, \sigma | \{(x_i, y_i)\}) \propto \frac{1}{\sigma^{N+1}} \exp \left[-\frac{N}{2\sigma^2} \left(Q(\hat{\theta}) + \frac{1}{2} \sum_{i,j} H_{ij} (\theta_i - \hat{\theta}_i)(\theta_j - \hat{\theta}_j) \right) \right].$$

Das ist eine multidimensionale Normalverteilung in den θ, die analytisch integriert werden kann. Es ergibt sich Diese Näherung wird für $N \to \infty$ exakt, so dass die nachfolgenden Ausdrücke vor allem für grosse N gültig sind. Aus der genäherten Posteriorverteilung lassen sich die Parameter θ analytisch ausintegrieren. Es ergibt sich

$$\mathrm{pdf}(\sigma | \{(x_i, y_i)\}) \propto \frac{1}{\sigma^{N-M+1}} \exp \left(-\frac{N Q_{\min}}{2\sigma^2} \right), \tag{5}$$

so dass

$$L(\sigma) = (N - M{+}1) \ln \sigma + \frac{N Q_{\min}}{2\sigma^2}.$$

Das Minimum dieser Funktion ist bei

$$\hat{\sigma}^2 = \frac{N Q_{\min}}{(N - M{+}1)}. \tag{6}$$

Der Ausdruck

$$\hat{\sigma} = \sqrt{\frac{N Q_{\min}}{N - M}}$$

ist der Schätzwert der Standardabweichung der Daten in der orthodoxen Statistik.

Die Genauigkeit dieses Schätzwertes ergibt sich näherungsweise aus der Entwicklung von $L(\sigma)$.

$$L(\sigma) \approx L_{\min} + \frac{1}{2}\frac{2(N-M{+}1)^2}{NQ_{\min}}(\sigma-\hat{\sigma})^2,$$

so dass

$$H = \frac{2(N-M{+}1)^2}{NQ_{\min}}$$

und

$$K = \frac{NQ_{\min}}{2(N-M{+}1)^2}.$$

Damit haben wir das Ergebnis

$$\boxed{\sigma = \sqrt{\frac{NQ_{\min}}{N-M{+}1}}\left(1 \pm \sqrt{\frac{1}{2(N-M{+}1)}}\right).}$$

Python: Schätzwert für σ und seine Ungenauigkeit.

```
N = len(data[:,0]) # number of data points
sigma_est = sqrt( NQmin/(N-M) )
stderr_sigma = sigma_est*sqrt( 1./2./(N-M) )
print('sigma = (%1.3f +/-%1.3f) ms'% (1e3*sigma_est,1e3*stderr_sigma))
```

Python: Code zum least-mean-square fit. Wir sind jetzt in der Lage, den kompletten Python Code-Ausschnitt zum least-mean-square fit zusammen zu stellen:

```
import numpy as np
from scipy.optimize import curve_fit

# Fitfunktion definieren:
def fit_func(x,p0,p1):
    return np.sqrt( 2 * (x - p1)/p0 )

# Least-mean-square Minimierung, optimale Parameter und NQmin:
pstart = [p0start,p1start] # Startwerte f. Parameter
result = curve_fit(fit_func,xdata,ydata,pstart,full_output=True)

p0hat = result[0][0]
p1hat = result[0][1]
residuals = result[2]['fvec']
NQmin = residuals.dot(residuals)

# Schaetzwert fuer sigma und seine Ungenauigkeit:
N = len(xdata) # number of data points
M = len(pstart) # number of fit parameters
sigma_est = sqrt( NQmin/(N-M) )
stderr_sigma = sigma_est*sqrt( 1./2./(N-M) )

# Kovarianzmatrix und Standardfehler:
covMatrix = result[1]
stderr_p0 = np.sqrt(covMatrix[0,0])
stderr_p1 = np.sqrt(covMatrix[1,1])
rho = covMatrix[0,1]/stderr_p0/stderr_p1

# Ausgabe des Ergebnisses:
print('p0 = %1.4f +/-%1.4f'% (p0hat, stderr_p0))
print('p1 = %1.4f +/-%1.4f'% (p1hat, stderr_p1))
print('Correlation coefficient = %1.4f'% (rho,))
print('sigma = %1.3f +/-%1.3f'% (sigma_est,stderr_sigma))
```