Script: Anpassen von Funktionen an Messdaten

24. Januar 2012

Funktionsanpassung mit der χ^2-Methode

Zusammenfassung

Der Vergleich von Modellen mit Messungen gehört zu den Standardaufgaben in der Experimentalphysik. Im einfachsten Fall stellt ein Modell eine Funktion dar, die Vorhersagen für Messdaten liefert und die üblicherweise von Modellparametern abhängt. Neben der Überprüfung, ob das Modell die Daten beschreibt, gehört die Bestimmung der Modellparameter zu den typischen Aufgaben. Zur Funktionsanpassung wird häufig die Methode der kleinsten Quadrate verwendet, mit der sich sehr elegant auch korrelierte Unsicherheiten oder Fehler in Abszissenrichtung zusätzlich zu Fehlern in Ordinatenrichtung behandeln lassen. Dieses Script gibt einen kurzen Abriss der Methode und enthält praktische Hinweise zur Funktionsanpassung mit Hilfe numerischer Methoden auf dem Computer.

Prof. Dr. Günter Quast
Homepage: `http://www-ekp.physik.uni-karlsruhe.de/~quast`
E-Mail: `G.Quast@kit.edu`

Inhaltsverzeichnis

1 Anpassung von Funktionen an Datenpunkte

Zum Vergleich von Messdaten mit theoretischen Modellen oder zur Bestimmung von Parametern werden Funktionen an Messdaten angepasst. Zu den gegebenen N Messwerten (x_1, y_i), $i = 1, \ldots, N$ wird also eine Funktion f gesucht, deren Funktionswerte $f(x_i)$ an den Stützstellen x_i möglichst nahe bei den Werten y_i liegt. Oft wird eine bestimmte Form der Funktion vorgegeben, z. B. ein Polynom, eine Exponentialfunktion o. Ä., die durch die theoretische Erwartung vorgegeben ist, und die eine Anzahl von K freien Parametern p_j enthält mit $j = 1, \ldots, K; K < N$. Abbildung 1 zeigt das Beispiel einer Parabel, die an mit Fehlern in Richtung der Ordinaten-Achse behaftete Messpunkte angepasst wurde.

Zum Auffinden der besten Werte der Parameter wird ein geeignetes Abstandsmass benötigt. Fasst man die auf den Stützstellen x_i definierten Funkionen als Elemente eines Vektorraums auf, also $\vec{y} = (y_1, \ldots, y_N)$ und $\vec{f} = (f(x_1; p_1, \ldots, p_K), \ldots, f(x_1; p_1, \ldots, p_K))$, so bietet das Skalarprodukt dieser Vektoren ein solches Abstandsmass, dass von den Elementen p_j des Parametervektors $\boldsymbol{p}$ abhängt: $d(\boldsymbol{p})^2 = (\vec{y} - \vec{f}(\boldsymbol{p})) \cdot (\vec{y} - \vec{f}(\boldsymbol{p}))$.

Für den so definierten Abstand von zwei Funktionen besteht die Aufgabe also darin, die besten Werte der Elemente des Parametervektors $\boldsymbol{p}$ zu finden. Dies gelingt in einfachen Fällen analytisch, d. h. durch die Bestimmung der Nullstellen der ersten Ableitungen von $d(\boldsymbol{p})^2$ nach den Parametern p_j. Meist wird die Minimierung jedoch mit Hilfe von numerischen Optimierungsmethoden vorgenommen, wie sie in gängigen Programmen wie *gnuplot* (http://www.gnuplot.info/), *Origin* (nur für Windows, akutelle Version lizenzpflichtig), *qtiplot* (http://wiki.ubuntuusers.de/qtiplot, unter Linux frei verfügbar, Bedienkonzept dem von *Origin* nachempfunden) und *Root* (im Quellcode verfügbar, http://root.cern.ch) implementiert sind. Auf einige dieser Programme wird in Abschnitt 6 etwas näher eingegangen.

Für statistische Daten, wie sie auch Messungen mit Messfehlern darstellen, muss das Abstandsmass natürlich die Messfehler berücksichtigen: Messpunkte mit großen Fehlern dürfen weiter von der anzupassenden Funktion entfernt sein als solche mit kleinen Unsicherheiten.

2 χ^2-Methode zur Funktionsanpassung

Häufig wird zur Funktionsanpassung an Datenpunkte mit Fehlern die „Methode der kleinsten Quadrate“ verwendet, d. h. die Minimierung der Summe der quadratischen Abweichungen der Datenpunkte von der Fit-Funktion normiert auf die jeweiligen Messfehler. Als Abstandsmass verwendet man in diesem Fall ein Skalarprodukt mit Gewichten, die dem Inversen der quadrierten Messfehler σ_i der Messungen entsprechen:

$$S = \left(\vec{y} - \vec{f}(\boldsymbol{p})\right) \cdot \left(\vec{y} - \vec{f}(\boldsymbol{p})\right) = \sum_{i=1}^{N} (y_i - f(x_i; \boldsymbol{p}))\ w_i\ (y_i - f(x_i; \boldsymbol{p}))\ , \text{ mit } w_i = \frac{1}{{\sigma_i}^2}\,.$$

Man nennt S auch die „gewichtete Summe von Residuenquadraten“(engl. „weighted sum of squared rediduals, WSSR“). Für Gauß-förmig um den wahren Wert verteilte Messfehler folgt S einer χ^2-Verteilung mit einer Anzahl von Freiheitsgraden, die durch die Zahl der Messwerte reduziert um die Zahl der anzupassenden Parameter, also $n_f = N - K$, gegeben ist, $\text{pdf}(S) = \chi^2(S; N - K)$. Daher hat sich für diese Methode auch der Name „χ^2-Methode“ etabliert.

Geschrieben mit den Messfehlern σ_i und etwas umgeformt ergibt sich

$$\chi^2(\boldsymbol{p}) = \sum_{i=1}^{N} \left(\frac{y_i - f(x_i; \boldsymbol{p})}{\sigma_i}\right)^2 . \tag{1}$$

Anmerkung 1: Falls $f(x_i, \boldsymbol{p})$ den Erwartungswert einer Messung am Punkt x_i beschreibt und die Messwerte einer Gaußverteilung mit Breite σ_i um diesen Erwartungswert folgen, so handelt es sich bei den einzelnen Summanden um die Quadrate von sogenannten „reduzierten Zufallsvariablen“, die einer Standard-Normalverteilung

folgen, d. h. einer Gaußverteilung mit Mittelwert Null und einer Standardabweichung von Eins. Der Zusammenhang mit der χ^2-Verteilung wird dadurch unmittelbar klar: die χ^2-Verteilung mit n_f Freiheitsgraden beschreibt ja gerade die Verteilung der Summe der Quadrate von n_f standardnormalverteilten Zufallszahlen.

Anmerkung 2: Für um den Erwartungswert $f(x_i)$ Gauß-förmig verteilte Messwerte y_i ist die χ^2-Methode äquivalent zum Likelihood-Verfahren, das die vollständige Kenntnis der Verteilungsdichte der Messfehler voraussetzt. Die χ^2-Methode benötigt dagegen nur den Erwartungswert und die Standardabweichung der Messfehlerverteilung und ist daher allgemeiner anwendbar.

Es lässt sich zeigen, dass die χ^2-Methode eine optimale und unverzerrte Schätzung der Parameter liefert, wenn die Parameter die Koeffizienten einer Linearkombination von Funktionen sind (siehe Abschnitt 4 zur analytischen Lösung solcher Probleme). Als Voraussetzung muss lediglich sichergestellt sein, dass die Varianzen der Messfehlerverteilungen existieren.

Unsicherheiten der Datenpunkte bzgl. der Abszissenachse, d. h. Fehler der x_i, können mit der χ^2-Methode recht elegant durch Iteration berücksichtigt werden:

- zunächst erfolgt eine Anpassung ohne Abszissenfehler,
- im zweiten Schritt werden diese dann mit Hilfe der ersten Ableitungen $f'(x_i)$ der im ersten Schritt angepassten Funktion f in entsprechende Fehler der Ordinate umgerechnet und quadratisch zu den betreffenden Fehlern der Ordinate addiert: ${\sigma_i}^2 = {\sigma_{y_i}}^2 + (f'(x_i) \cdot \sigma_{x_i})^2$. Die Größe χ^2 wird jetzt mit den neuen Fehlern σ_i berechnet und minimiert.
- Ein dritter Schritt, der der Vorgehensweise beim zweiten Schritt entspricht, dient zur Verbesserung des Ergebnisses und zur Fehlerkontrolle - der Wert von χ^2 am Minimum darf sich vom zweiten zum dritten Schritt nicht signifikant ändern, ansonsten muss nochmals iteriert werden.

Eine geometrische Interpretation dieser Vorgehensweise zeigt Abbildung 2. Die anzupassende Funktion wird durch die Tangente im Punkt (x_i,y_i) angenähert und das Quadrat des auf den Messfehler normierte Abstands der Funktion von den Datenpunkten, $\chi^2 = d^2 / \left(\cos^2(\alpha)\,{\sigma_y}^2 + \sin^2(\alpha)\,{\sigma_x}^2\right)$ minimiert. Mit $d = \cos(\alpha)\,(y_i - f(x_i))$ und $\tan\alpha = f'(x_i)$ folgt die oben angegebene Formel. Wenn die Tangente keine gute Näherung der anzupassenden Funkion über den Bereich der Messfehler von x_i ist, wird dieses Verfahren ungenau.

2.1 Behandlung von korrelierten Fehlern

Unsicherheiten von Messwerten sind typischerweise einerseits bestimmt durch unabhängige Fehler jeder Einzelmessung, wie z.B. Ablesefehler, statistische Fehler usw.. Andererseits gibt es systematische Fehler, die alle Messwerte in gleicher Weise betreffen, also zwischen diesen „korreliert“ sind. Die Beschreibung solcher korrelierter Fehler geschieht mit Hilfe der sogenannten Kovarianz-Matrix C, einer symmetrischen Matrix, deren Dimension der Anzahl der Messungen entspricht. Die Diagonalelemente enthalten die Varianzen, d. h. den quadrierten Gesamtfehler der Messwerte, $C_{ii} = {\sigma_i^t}^2$, die Nebendiagonalelemente enthalten die gemeinsamen Fehlerkomponenten, $C_{ij} = \sigma_i^g \cdot \sigma_j^g$ der Messungen mit den Indizes i und j.

Wenn die Messfehler korreliert sind, wird χ^2 mit Hilfe der Kovarianzmatrix der Messfehler $\mathbf{C}$ ausgedrückt. Fasst man die Differenzen $y_i - f(x_i;\boldsymbol{p})$ zum sogenannten Residuenvektor mit den Komponenten zusamen, $\Delta_i(\boldsymbol{p}) = y_i - \mathrm{f}(x_i;\boldsymbol{p})$, so ergibt sich

$$\chi^2(\boldsymbol{p}) = \vec{\Delta}(\boldsymbol{p})^T\, \mathbf{C}^{-1}\, \vec{\Delta}(\boldsymbol{p})\,.$$

Hierbei ist C^{-1} die Inverse der Kovarianzmatrix.

Korrelierte Fehler der Messpunkte in Richtung der Abszisse werden durch eine Kovarianzmatrix mit den Elementen C_{ij}^x beschrieben und mit Hilfe der 1. Ableitung zu den Kovarianzmatrixelementen C_{ij}^y der Ordinatenfehler addiert, d. h. die Elemente der gesamten Kovarianzmatrix $\mathbf{C}$ ergeben sich zu

$$C_{ij} \;=\; C_{ij}^y \;+\; C_{ij}^x \cdot f'(x_i) \cdot f'(x_j)\,.$$

Mit dieser neuen Kovarianzmatrix wird nun die Anpassung wiederholt und ein dritter Schritt zur Kontrolle der numerischen Fehler durchgeführt.

Insgesamt ergibt sich also mit dem Vektor der ersten Ableitungen $\vec{f}'$ der allgemeinste Ausdruck zur Berücksichtigung von korrelierten Fehlern der Messpunkte in Abszissen- und Ordinatenrichtung mit Kovarianzmatrizen $\mathbf{C}^x$ bzw. $\mathbf{C}^y$:

$$\chi^2(\boldsymbol{p}) = \vec{\Delta}(\boldsymbol{p})^T \left(\mathbf{C}^y + \vec{f}'^T \mathbf{C}^x \vec{f}'\right)^{-1} \vec{\Delta}(\boldsymbol{p})\,. \tag{2}$$

Dieses allgemeine Verfahren ist im Programm *RooFiLab* implementiert.

2.2 Bestimmung der Fehler der Parameter

Die Fehler der Parameter hängen mit den zweiten Ableitungen von $\chi^2(\boldsymbol{p})$ nach den Parametern p_j zusammen, die bei den Werten $\hat{p}_i$ der Parameter am Minimum, dem best-fit-Punkt, ausgewertet werden. Die Krümmungen am Minimum legen die Elemente der Inversen der Kovarianzmatrix V_{ij} der Parameter fest:

$${V_{ij}}^{-1} = \frac{1}{2} \left.\frac{\partial^2 \chi(\boldsymbol{p})^2}{\partial p_i\, \partial dp_j}\right|_{\hat{p}_i \hat{p}_j}\,. \tag{3}$$

Werden mehrere Parameter angepasst, so sind deren Fehler häufig korreliert, selbst wenn die Fehler der Messdaten unkorreliert sind; dies ist unerwünscht, weil Ergebnisse nur unter Angabe aller mit ihnen korrelierter Parameter verwendbar sind. Durch geeignete Parametrisierung kann aber oft die Korrelation verringert werden. Zum Beispiel ist es bei der Anpassung von Geraden an Messpunkte viel sinnvoller, statt der gewohnten Darstellung $f(x) = ax + b$ eine Parametrisierung in den transformierten Variablen $y - \bar{y}$ und $x - \bar{x}$ vorzunehmen, wobei $\bar{x}$ und $\bar{y}$ die Mittelwerte der x- bzw. y-Werte bedeuten. Es ergibt sich dann als Geradengleichung $f(x) = a\,(x - \bar{x}) + b'$

Werden keine Fehler der Datenpunkte angegeben, so werden in der χ^2-Summe alle Gewichte, d. h. die Fehler σ_i der Messwerte, auf Eins gesetzt und eine Anpassung mit gleichem Gewicht aller Messpunkte durchgeführt. In diesem Fall sind die von manchen Programmen dennoch ausgegebenen Fehler der Parameter bedeutungslos.

Dennoch kann bei unbekannten Fehlern der Datenpunkte eine Abschätzung der Fehler der Parameter über den Wert von χ^2 am Minimum gewonnen werden, dessen Erwartungswert $\left\langle \chi^2(\hat{\boldsymbol{p}}) \right\rangle = N - K$ ist. Anders ausgedrückt, $\chi^2/n_f = \frac{\left\langle \chi^2(\hat{\boldsymbol{p}}) \right\rangle}{N-K}$ hat den Wert Eins. Werden für die Messpunkte in etwa gleiche Fehler vermutet, so führt man zunächst eine Anpassung mit $\sigma_i = 1$ durch und erhält χ^2/n_f. Die von der Anpassung gelieferten Fehler können nun in einem zweiten Schritt entsprechend skaliert werden. Bei diesem Verfahren werden die Fluktuationen der Messwerte um die angepasste Kurve benutzt, um Aussagen über die statistischen Fehler der Datenpunkte zu erhalten, die mitunter auf andere Art nur schwer zu bestimmen sind. Allerdings entspricht das dabei angenommen Fehlermodell einer gleichmäßigen Skalierung aller Messwertfehler, $\sigma_i = \sqrt{\chi^2/n_f}$, eine Annahme, die in der Praxis selten gerechtfertigt ist. Außerdem verliert man so die Möglichkeit, aus dem Wert von χ^2 am Minimum eine Aussage über die Übereinstimmung der Messdaten mit dem gewählten Modell zu gewinnen, wie im folgenden Kapitel beschrieben, da eine wiederholte Anpassung mit so skalierten Messwertfehlern per Konstruktion $\chi^2/n_f = 1$ liefert. Bei einigen Programmpaketen, die nicht primär von Physikern genutzt werden, ist dieses Vorgehen als Standard voreingestellt.

2.3 χ^2 als Testgröße für die Qualität einer Anpassung

Da bei Gauß-förmiger Fehlerverteilung der Datenpunkte die Werte von χ^2 am Minimum einer χ^2-Verteilung folgen (s. Abbildung 3), kann dieser Wert als Test für die Güte der Beschreibung der Daten durch die Funktion benutzt werden. Dazu integriert man die χ^2-Verteilung vom beobachteten Wert bis ∞, und erhält so eine

Aussage darüber, mit welcher Wahrscheinlichkeit ein schlechterer Wert χ_{min} von χ^2 am Minimum erwartet würde als tatsächlich beobachtet. Dies wird oft als χ^2-Wahrscheinlichkeit bezeichnet:

$$\chi^2{}_{\rm prob} = \int_{\chi_{min}}^{\infty} \chi^2(s;n_f)\,{\rm d}s \;=\; 1. - \int_0^{\chi_{min}} \chi^2(s;n_f)\,{\rm d}s \;\;{}^{1}. \tag{4}$$

Die in *ROOT* verfügbare Funktion `Double_t Prob(Double_t chi2, Int_t ndf)` liefert diese wichtige Testgröße $\chi^2{}_{\rm prob}$. Für ein korrektes Modell ist sie im Intervall $[0,1]$ gleichverteilt, d. h. z. B. dass in 5 % der Fälle auch bei korrektem Modell eine χ^2-Wahrscheinlichkeit von 0.05 oder kleiner beobachtet wird.

Anschaulich leichter zu handhaben ist der auf die Zahl der Freiheitsgrade normierte Wert χ^2/n_f mit einem Erwartungswert der Verteilung von Eins, wie in Abbildung 4 gezeigt. Mit wachsender Zahl der Freiheitsgrade wird die Streuung der Verteilung um den Wert 1. kleiner, die Breite ist $\frac{2}{\sqrt{n_f}}$. Bei 20 Freiheitsgraden wir nur mit einer Wahrscheinlichkeit von 10 % ein Wert von χ^2/n_f größer als 1.5 erwartet.

3 Konstruktion von Kovarianzmatrizen

Die Kovarianzmatrix C ist eine quadratische und symmetrische Matrix, deren Dimension die Anzahl der Messwerte N hat. Die Diagonalelemente der Kovarianzmatrix sind durch den Gesamtfehler der Messwerte y_i gegeben:

$$C_{ii} = {\sigma_i^{(t)}}^2$$

Die Produkte der korrelierten Fehlerkomponenten $\sigma_i^{(g)}$ und $\sigma_j^{(g)}$ der Messwerte y_i und y_j bilden die Nebendiagonalelemente:

$$C_{ij} = \sigma_i^{(g)}\,\sigma_j^{(g)}$$

Sind zum Beispiel alle Messwerte von einem gemeinsamen Fehler $\sigma^{(g)}$ betroffen, so gilt $C_{ij} = {\sigma^{(g)}}^2$ für alle i,j. Es können auch Gruppen von Messungen von gemeinsamen Fehlern $\sigma^{(g_K)}$ betroffen sein; dann stehen die Quadrate dieser Fehler jeweils in den zu den Blockmatrizen der Gruppe k gehörenden Nebendiagonalelementen. Im allgemeinsten Fall müssen die korrelierten Fehleranteile $\sigma_i^{(g)}$ und σ_j^{g} nicht gleich sein. Das ist zum Beispiel dann der Fall, wenn die Messfehler durch einen relativen Anteil der Messwerte gegeben sind, also beispielsweise ein korrelierter Fehler von 1 % des jeweiligen Messwertes vorliegt.

Bei der Konstruktion der Kovarianzmatrix beginnt man mit den unkorrelierten Fehlern ${\sigma^{(u)}}_i$ der Messwerte und setzt deren Quadrate auf die Diagonale. Solche unkorrelierten Fehler sind häufig die statistischen Fehler einer Messung. Die korrelierten Fehler ${\sigma^{g}}_i$ und ${\sigma^{g}}_j$, häufig von systematischen Fehlern herrührend, werden quadratisch zum jeweiligen Diagonalelement addiert und auch auf der Nebendiagonalen eingetragen:

$$\begin{aligned}
C_{ii} &= {\sigma_i^{(t)}}^2 = {\sigma_i^{(u)}}^2 + {\sigma_i^{(g)}}^2 \\
C_{jj} &= {\sigma_j^{(t)}}^2 = {\sigma_j^{(u)}}^2 + {\sigma_j^{(g)}}^2 \\
C_{ij} &= \sigma_i^{(g)}\sigma_j^{(g)} \\
C_{ji} &= C_{ij}
\end{aligned}$$

Wenn es mehrere korrelierte Fehlerkomponenten gibt, so erhält man die Gesamtfehler-Kovarianzmatrix durch Addition aller so berechneten Kovarianzmatrizen. Dies entspricht der quadratischen Addition von Einzelfehlerbeiträgen – Kovarianzmatrixelemente sind quadratische Formen!

[1] Der 2. Ausdruck ist numerisch einfacher zu berechnen.

3.1 Korrelationsmatrizen

Häufig verwendet man statt der Kovarianzmatix die sogenannte Korrelationsmatrix Cor mit den Elementen

$$Cor_{ij} = \frac{C_{ij}}{\sqrt{C_{ii}\,C_{jj}}} = \frac{C_{ij}}{\sigma_i\,\sigma_j}\,.$$

Alle Diagonalelemente der Korrelationsmatrix sind 1, und für die Nebendiagonalelemente gilt $-1 < Cor_{ij} < 1$. Ist Cor_{ij} Eins, so sind die Messungen y_i und y_j voll korreliert, für $Cor_{ij} = -1$, spricht man von vollständiger Antikorrelation. Wegen des eingeschränkten Wertebereichs der Matrixelemente sind Korrelationsmatrizen anschaulicher und leichter zu bewerten als Kovarianzmatrizen. Bei Kenntnis der Korrelationsmatrix müssen auch die Gesamtfehler der Messwerte bekannt sein, um die Kovarianzmatrix z. B. für die Verwendung in Parameteranpassungen zu konstruieren:

$$C_{ij} = \frac{\underbrace{\sigma_i \cdot \sigma_j}}{\sqrt{C_{ii}\cdot C_{jj}}} \cdot Cor_{ij}$$

4 Analytische Lösung für lineare Probleme

Wenn die Parameter nur linear in der gewichteten Summe S der Residuenquadrate auftreten, lässt sich das Minimum bzgl. des Parametervektors $\boldsymbol{p}$ analytisch bestimmen. Schreibt man die anzupassende Funktion f als Linearkombination von K Funktionen F_j mit $f(x_i) = \sum_{j=1}^{K} p_j F_j(x_i)$ und führt die $N \times K$-Matrix A mit N Zeilen und K Spalten mit den Koeffizienten $A_{ij} := F_j(x_i)$ ein, so vereinfacht sich der Residuenvektor zu $\vec{\Delta}(\boldsymbol{p}) = \vec{y} - A\boldsymbol{p}$. Für S ergibt sich also mit $W = C^{-1}$:

$$S(\boldsymbol{p}) = (\vec{y} - A\boldsymbol{p})^T W (\vec{y} - A\boldsymbol{p}). \tag{5}$$

Das Minimum findet man durch Nullstellenbestimmung der ersten Ableitung, $\left.\frac{\mathrm{d}S}{\mathrm{d}\boldsymbol{p}}\right|_{\hat{\boldsymbol{p}}} = 0$, und Auflösen nach dem gesuchten Parametervektor $\hat{\boldsymbol{p}}$. Die Lösung ist

$$\hat{\boldsymbol{p}} = (A^T W A)^{-1} A^T W \vec{y}. \tag{6}$$

Die Schätzwerte für die Parameter ergeben sich also durch Linearkombination der Messwerte mit Koeffizienten, in die die Kovarianzmatrixelemente der Messungen und die Funktionswerte $F_j(x_i)$ eingehen.

Die Kovarianzmatrix der Parameter erhält man durch Fehlerfortpflanzung der Kovarianzmatrix C der Messfehler. Mit der Abkürzung $B := (A^T W A)^{-1} A^T W$ gilt $\hat{\boldsymbol{p}} = B\vec{y}$; damit ergibt sich die Kovarianzmatrix der Parameter zu $V_{\hat{\boldsymbol{p}}} = B^T C B$, also nach einigen Vereinfachungen

$$V_{\hat{\boldsymbol{p}}} = (A^T W A)^{-1} = (A^T C^{-1} A)^{-1}\,. \tag{7}$$

Alternativ hätte man natürlich, wie oben schon beschrieben, die mit $\frac{1}{2}$ multiplizierte Inverse der Matrix der zweiten Ableitungen von S nach den Parametern bilden können, mit identischem Ergebnis.

4.1 Lineare Regression

Aus dem hier erhaltenen allgemeinen Ergebnis lassen sich die bekannten Formeln für die lineare Regression bei unkorrelierten Messfehlern gewinnen. Für die Anpassung einer Geraden $f(x) = p_1 + p_2\,x$ gilt

$$A = \begin{pmatrix} 1 & x_1 \\ \ldots & \ldots \\ 1 & x_N \end{pmatrix}, \quad W = \begin{pmatrix} \frac{1}{\sigma_1^2} & 0 & \ldots & 0 & 0 \\ 0 & \ldots & \frac{1}{\sigma_i^2} & \ldots & 0 \\ 0 & 0 & \ldots & 0 & \frac{1}{\sigma_N^2} \end{pmatrix}.$$

Man erhält durch Einsetzen in Gleichungen 6 und 7 mit den Abkürzungen

$$S_1 = \sum_{i=1}^{N} \frac{1}{\sigma_i^2}\,, \qquad S_x = \sum_{i=1}^{N} \frac{x_i}{\sigma_i^2} = \overline{x}\, S_1\,, \qquad S_y = \sum_{i=1}^{N} \frac{y_i}{\sigma_i^2} = \overline{y}\, S_1\,,$$
$$S_{xx} = \sum_{i=1}^{N} \frac{x_i^2}{\sigma_i^2} = \overline{x^2}\, S_1\,, \qquad S_{xy} = \sum_{i=1}^{N} \frac{x_i\, y_i}{\sigma_i^2} = \overline{xy}\, S_1\,, \qquad D = S_1 S_{xx} - S_x^2$$

als Lösung

$$\hat{p}_1 = \frac{S_{xx} S_y - S_x S_{xy}}{D}\,, \qquad {\sigma_{p_1}}^2 = \frac{S_{xx}}{D}\,,$$
$$\hat{p}_2 = \frac{S_1 S_{xy} - S_x S_y}{D}\,, \qquad {\sigma_{p_2}}^2 = \frac{S_1}{D}\,, \qquad V_{12} = \frac{-S_x}{D}\,.$$

Das Kovarianzmatrixelement V_{12} verschwindet, wenn $S_x = 0$ gilt, der Erwartungswert $\overline{x}$ der Abszissenwerte also Null ist. Dies kann man durch geeignete Parametrisierung der Geradengleichung erreichen, wenn man $x' = x - \overline{x}$ setzt, d. h. $f'(x) = p'_1 + p'_2\,(x - \overline{x})$. Jetzt erhält man die einfacheren, unkorrelierten Lösungen

$$\hat{p}'_1 = \frac{S_y}{S_1} = \overline{y}\,, \qquad {\sigma_{p'_1}}^2 = \frac{1}{S_1}\,,$$
$$\hat{p}'_2 = \frac{S_{x'y}}{S_{x'x'}} = \frac{\overline{x'y}}{\overline{x'^2}}\,, \qquad {\sigma_{p'_2}}^2 = \frac{1}{S_{x'x'}}\,.$$

Für die Weiterverwendung sind unkorrelierte Ergebnisse von großem Vorteil, so dass man zur linearen Regression immer dieses letztgenannte Verfahren anwenden sollte.

Die hier abgeleiteten Formeln finden in zahlreichen Computerprogrammen und auch in Taschenrechnern Verwendung und sind Bestandteil mancher Praktikumsanleitung. Oft werden Fehler der Messwerte nicht berücksichtigt, d. h. $\sigma_i = 1$ für alle i und damit $S_1 = N$. Das hier beschriebene Verfahren mit Berücksichtigung von Messfehlern wird in der Literatur üblicherweise als "gewichtete lineare Regression" bezeichnet.

5 Likelihood-Methode

Bei typischen Problemen in der Physik, bei denen Zählraten oder Häufigkeitsverteilungen gemessen werden, folgt die Verteilung der Messfehler einer Poisson-Verteilung, d. h. die Wahrscheinlichkeit n Ereignisse zu beobachten, wenn μ erwartet wurden, ist gegeben durch $P(n;\mu) = \frac{\mu^n}{n!}\mathrm{e}^{-\mu}$. Für große n nähert sich diese Verteilung einer Gaußverteilung mit Mittelwert μ und Breite $\sqrt{\mu}$ an.

In solchen Fällen kann man also ebenfalls die χ^2-Methode einsetzen:

$$S(\vec{n};\boldsymbol{p}) = \sum_{i=1}^{N} \frac{\left(n_i - \mu_i(\boldsymbol{p})\right)^2}{\mu_i(\boldsymbol{p})}\,.$$

Der Einfachheit halber setzt man für die Fehlerquadrate im Nenner oft die aus der Beobachtung gewonnen Werte n_i ein; dann jedoch erhält man eine stark verzerrte Anpassung, denn eine Fluktuation zu kleineren Werten führt zu einem kleineren Fehler, und das Gewicht in der Anpassung wird größer; in der Konsequenz wird die Anpassung also in Richtung der zu kleineren Werten fluktuierten Messungen verzerrt. Wenn es für einzelne Messungen i zu einer Beobachtungen von null Ereignissen kommt, kann dieser Messpunkt überhaupt nicht verwendet werden und muss weggelassen werden – obwohl auch eine solche Beobachtung Information enthält! Dieses Problem kann man durch Iteration vermeiden:
– in einem ersten Schritt wird eine Anpassung mit den beobachteten Fehlern durchgeführt,
– im zweiten Schritt werden die Fehlerquadrate durch die im ersten Schritt gewonnenen Werte für $\mu_i(\boldsymbol{p})$ ersetzt.

Insbesondere bei sehr kleinen Zählraten ist es aber nötig, die genaue Form der Poisson-Verteilung zu berücksichtigen. Dies gelingt mit der Likelihood-Methode, bei der man die Wahrscheinlichkeiten, in der Messung i den Wert n_i zu beobachten, gemäß der Poissonverteilung berechnet und die so erhaltenen Werte aller Messungen multipliziert. Man erhält dann die vom Parametervektor abhängige Likelihood-Funktion $\mathcal{L} = \prod_{i=1}^{N} \mathrm{P}\left(n_i; \mu_i(\boldsymbol{p})\right)$. Gemäß dem Likelihood-Prinzip liefert die Maximierung der Likelihood-Funktion bzgl. der Parameter $\boldsymbol{p}$ eine Schätzung für die gesuchten Parameter.

Man verwendet meist den Logarithmus der Likelihood, lässt konstante, d. h. nicht vom Parametervektor abhängige Terme weg und erhält die „negative Log-Likelihood Funktion“ für das Problem,

$$-\log \mathcal{L}(\vec{n}; \boldsymbol{p}) \;=\; \sum_{i=1}^{N} -n_i \cdot \ln(\mu_i(\boldsymbol{p})) \;+\; \mu(\boldsymbol{p})\,, \tag{8}$$

die man bzgl. der Parameter minimiert. Für Gaußverteilungen entspricht $-\log \mathcal{L}$ bis auf einen Faktor $\frac{1}{2}$ der χ^2-Größe, $-\log \mathcal{L} = \frac{1}{2}\chi^2$. Die Bestimmung der Fehler der Parameter erfolgt durch Analyse der zweiten Ableitungen am Minimum,

$$V_{ij}{}^{-1} = \left.\frac{\partial^2 \ln \mathcal{L}(\vec{n}; \boldsymbol{p})}{\partial p_i \, \partial dp_j}\right|_{\hat{p}_i \hat{p}_j}\,. \tag{9}$$

$-\log \mathcal{L}$ ist fast immer eine nichtlineare Funktion der Parameter, und daher werden zur Lösung des Minimierungsproblems numerische Methoden benötigt.

6 Programme zur Funktionsanpassung

Dank der Verbreitung von Computern können heute vollständige Minimierungsverfahren inklusive eines χ^2-Tests und einer Untersuchung der Korrelationen der angepassten Parameter auch für nicht-lineare Probleme durchgeführt werden. Die Transformation von Parametern zum Erzwingen eines linearen Zusammenhangs zwischen Abszissen- und Ordinaten-Werten ist nicht mehr notwendig [2]

Numerische Minimierungsverfahren nutzen verschiedene, oft mehrschrittige Algorithmen zur Suche nach einem Minimum einer skalaren Funktion im K-dimensionalen Parameterraum. Solche Verfahren funktionieren sowohl für lineare als auch für nichtlineare Probleme, sind aber bei linearen Problemen deutlich ineffizienter als das oben besprochene analytischen Lösungsverfahren. Nichtlineare Problemstellungen sind allerdings eher die Regel; auch ein zunächst lineares Problem kann durch Erweiterungen zur besseren Modellierung der Daten sehr schnell zu einem nichtlinearen werden. Bei nichtlinearen Problemen gibt es in der Regel mehr als ein Minimum, und ein Algorithmus findet nicht notwendigerweise das globale Minimum. Welches Minimum gefunden wird, hängt stark von den Startwerten ab, die solche Algorithmen grundsätzlich benötigen. Daneben verlangen einige Algorithmen auch die Angabe einer anfänglichen Schrittweite zur Suche nach einem geeigneten Startpunkt für eine zweite Stufe von effizienteren Algorithmen, die in der Nähe des vermuteten Minimums die ersten Ableitungen nach den Parametern nutzen, um die Konvergenz zu beschleunigen. Optional erlaubt z. B. das in *ROOT* zur Minimierung werwendete Paket *MINUIT* zur Steigerung der numerischen Effizienz, die Ableitungen der χ^2-Funktion nach den Parametern in Form vom Programmcode zu spezifizieren. Normalerweise werden die benötigten Ableitungen sowie die zweiten Ableitungen zur Konstruktion der Fehlermatrix numerisch bestimmt. Bei komplexen anzupassenden Funktionen kann es sogar notwendig werden, die Genauigkeit der Funktionsauswertung anzugeben,

[2] Dieses immer noch oft praktizierte Verfahren ist bei großen Messfehlern sogar falsch, denn durch die Transformation der Messdaten werden auch die Fehlerverteilungen verzerrt, und die Annahme von Gauß-förmigen Fehlern bei der linearen Regression ist nicht mehr gerechtfertigt.

um numerisches Rauschen von einer tatsächlichen Änderung der χ^2-Funktion zu unterscheiden. Werden keine Angaben gemacht, so verwenden fast alle Programme vernünftige Standart-Werte, die in vielen Fällen zu guten Ergebnissen führen.

Es existieren eine Reihe von Programmen, die eine Funktionsanpassung mit numerischer Minimierung von $S(\boldsymbol{p})$ erlauben und starke Unterschiede in Bezug auf ihre Eigenschaften und Möglichkeiten aufweisen. In den folgenden Abschnitten wird kurz auf drei Programme bzw. Softwarepakte eingegangen, die auf allen üblichen Platformen als offener Quellcode verfügbar sind.

6.1 Funktionsanpassung mit *qtiplot*

Das Programm *qtiplot* (http://wiki.ubuntuusers.de/qtiplot) ist in einigen Linux-Distributionen enthalten und frei verfügbar. Die Bedienung erfolgt über die grafische Oberfläche, die der Funktionalität des teuren *Origin* entspricht. Daten werden in Tabellenform eingegeben, wie man es aus Tabellenkalkulationen kennt, ein Datenimport aus ASCII-Dateien ist ebenfalls möglich, deren Format in Abbildung 5 zusammen mit einer typischen grafischen Darstellung des Ergebnisses gezeigt ist.

Nach dem Starten des Programms werden über den Menü-Punkt File/Import/Import ASCII die Beispieldaten eingelesen und als Tabelle dargestellt. Die dritte Spalte muss nun mit der rechten Maustaste angeklickt werden, um im Kontext-Menü set as / Y Error anzuwählen. Als Standard für Anpassungen ist in *qtiplot* eine ungewichtete χ^2–Methode eingestellt; zur korrekten Berücksichtigung von Fehlern der Eingabedaten müssen daher zunächst einige Optionen eingestellt werden. Die für die Anpassung vorgesehenen Felder in der Tabelle müssen mit der Maus markiert werden, dann den Menüpunkt Analysis/Fit Wizard ... wählen und die anzupassende Funktion festlegen. Im zweiten Reiter dieses Menüs können nun die Fit-Optionen eingestellt werden – zur Berücksichtigung der in der dritten Spalte der Tabelle angegebenen Fehler die Option Weighting: instrumental auswählen und dann die Aktion Fit zum Ausführen der Anpassung anklicken. Das Ergebnis der Anpassung einer Parabel an die Daten aus dem obigen Beispiel sieht dann wie folgt aus:

```
Polynomial Fit of dataset: fitexample-qti_2, using function: a0+a1*x+a2*x^2
Weighting Method: Instrumental, using error bars dataset: fitexample-qti_3
From x = 5,000e-02 to x = 4,600e+00
a0 = 3,155e-01 +/- 4,578e-02
a1 = 1,932e-01 +/- 5,905e-02
a2 = -1,652e-02 +/- 1,276e-02
--------------------------------------------------------------------------------
Chi^2/doF = 9,646e-01
R^2 = 0,864
Adjusted R^2 = 0,819
RMSE (Root Mean Squared Error) = 0,982
RSS (Residual Sum of Squares) = 9,65
```

Die Angabe `RSS` ist der χ^2-Wert der Anpassung, `RMSE` ist die Wurzel aus `{chi^2/doF}`, dem Wert von χ^2 dividiert durch die Zahl der Freiheitsgrade. Vorsicht: in den Standardeinstellungen werden die Fehler der Parameter mit diesem Wert skaliert, d./,h. es wird angenommen, das die angepasste Funktion die Daten genau beschreibt, χ^2/n_f also exakt Eins ist, und alle Fehler der Eingabedaten werden mit dem gleichen Faktor skaliert. Dieses Verfahren wird auch angewandt, wenn keine Fehler angegeben werden. In diesem Fall sind die ausgegebenen Parameterfehler mit größter Vorsicht zu behandeln!

qtiplot enthält eine ganze Reihe weiterer Möglichkeiten zur Darstellung und Analyse von Messdaten. Es sei an dieser Stelle auf die Online-Hilfe verwiesen.

6.2 Funktionsanpassung mit *gnuplot*

Das Programm *gnuplot* gibt es für Linux und Windows. Seine Hauptanwendung ist die Visualisierung von Daten und Funktionen, es beinhaltet aber auch die Möglichkeit, Funktionen an fehlerbehaftete Messdaten anzupassen.

Zur Anpassung einer Parabel an die in der Datei `fitexample.dat` im Format `" x y Fehler "` gespeicherten Messungen genügt in *gnuplot* die folgende einfache Befehlssequenz, die man auf der Kommandozeile nach dem Aufruf des Programms eingibt:

```
gnuplot> f(x) = a*x*x + m * x + b
gnuplot> fit f(x) 'fitexample.dat' using 1:2:3 via a,m,b
gnuplot> plot 'fitexample.dat' using 1:2:3 with errorbars ,f(x)
```

Man erhält damit die in Abbildung 6 gezeigte Grafik und folgende Ausgabe auf der Textkonsole:

```
degrees of freedom    (FIT_NDF)                       : 10
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.982122
variance of residuals (reduced chisquare) = WSSR/ndf   : 0.964564
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = -0.0165198       +/- 0.01253      (75.87%)
m               = 0.193232         +/- 0.058        (30.01%)
b               = 0.315463         +/- 0.04496      (14.25%)
correlation matrix of the fit parameters:
               a      m      b
a               1.000
m              -0.956  1.000
b               0.742 -0.855  1.000
```

Angezeigt werden die Zahl der Freiheitsgrade, n_f, der Anpassung, sowie der Wert von χ^2/n_f am Minimum (`WSSR`/ndf). Auch die Korrelationsmatrix der Parameter wird mit ausgegeben. Im obigen Beispiel sind die Korrelationen sehr groß - eine bessere Parametrisierung sollte gewählt werden – etwa eine Verschiebung der x-Werte gemäß $x' = x - \overline{x}$.

6.3 Funktionsanpassung mit *ROOT*

Das Programmpaket *ROOT* ist ein mächtiges Software-Framework zur Datenanalyse für wissenschaftliche Problemstellungen. *ROOT* gibt es als vollständigen Quellcode oder vorcompiliert für Linux und Windows. *ROOT* kann über eine Makro-Sprache in C++-Syntax oder über ein Python-Interface interaktiv benutzt werden, eigener C++ code kann aber auch mit den Bibliotheken von *ROOT* gelinkt und so ein ausführbares Programm mit effizientem Maschinen-Code erzeugt werden.

ROOT enthält zwei Basisklassen, `TGraph` und `TH1`, die Methoden zur Anpassung von Funktionen bereit stellen. `TH1` ist einen Klasse zur Darstellung und Bearbeitung von Häufigkeitsverteilungen und daher nur in speziellen Fällen anwendbar. Die Klasse `TGraphErrors` stellt Messungen als fehlerbehaftete Datenpunkte (x_i, y_i) dar und erlaubt es, Anpassungen mit Fehlern in Ordinaten- und Abszissenrichtung durchzuführen, allerdings ohne Korrelationen der Messfehler zu berücksichtigen. `TGraphErrors` bietet eine Anzahl von Methoden, Daten einzulesen und darzustellen und nutzt die Basisklasse `TVirtualFitter` zur Ausführung von Anpassungen. Komplexere Probleme lassen sich durch eigenen Programmcode bewältigen, der die Minimierung von beliebigen Nutzer-definierten Funktionen $S[\boldsymbol{p})$ mit Hilfe der durch zahlreiche Optionen konfigurierbaren Basisklasse `TVirtualFitter` durchführt.

ROOT-Macros in C++ können interaktiv vom eingebauten Interpreter CINT ausgewertet, aber auch mit Hilfe des System-Compilers übersetzt und dann im Interpreter als schneller Maschinencode ausgeführt werden. Für Details sei auf die Dokumentation von *Root*[3] verwiesen. Der Konstruktor der Klasse `TGraphErrors` erlaubt die Übernahme von Daten aus einer Datei oder aus entsprechend initialisierten C-Arrays, wie im folgenden Beispiel gezeigt. Ist dieses *ROOT*-Macro in der Datei `TGraphFit.C` gespeichert, so lässt es sich nach dem Aufruf von *ROOT* direkt von der *ROOT*-Shell aus aufrufen:

```
root [0] .x TGraphFit.C
```

Das Beispiel demonstriert die Anpassung eines Polynoms dritten Grades an Daten mit Fehlern in x- und y-Richtung.

```
void TGraphFit() {
   //Draw a graph with error bars and fit a function to it.
   //Originial source Rene Brun    modified: Gunter Quast

  //set global options
  gStyle->SetOptFit(111);  //superimpose fit results

  // make nice Canvas
  TCanvas *c1 = new TCanvas("c1","Daten",200,10,700,500);
  c1->SetGrid();

  //define some data points
  const Int_t n = 10;
  Float_t x[n]  = {-0.22, 0.1, 0.25, 0.35, 0.5, 0.61, 0.7, 0.85, 0.89, 1.1};
  Float_t y[n]  = {0.7, 2.9, 5.6, 7.4, 9., 9.6, 8.7, 6.3, 4.5, 1.1};
  Float_t ey[n] = {.8,.7,.6,.5,.4,.4,.5,.6,.7,.8};
  Float_t ex[n] = {.05,.1,.07,.07,.04,.05,.06,.07,.08,.05};
  //  Float_t ex[n] = {.0,.0,.0,.0,.0,.0,.0,.0,.0,.0};  for test

  // copy data to TGraphErros object
  TGraphErrors *gr = new TGraphErrors(n,x,y,ex,ey);
  gr->SetTitle("TGraphErrors mit Fit");
  gr->Draw("AP");

  // now perform a fit(with errors in x and y!)
   gr->Fit("pol3");
   c1->Update();
```

Dieses Makro erzeugt Textausgaben im Konsolenfenster und die Grafik in einem separaten Grafikfenster, das interaktive Möglichkeiten zur Bearbeitung und zum Export der Grafik anbietet. Auch der Export als Makro in C++-Syntax ist möglich - auf diese Art kann man Beispiele für die Anwendung der Methoden der unterschiedlichen Grafik-Klassen für die Darstellung von Rahmen, Achsen, Text usw. erhalten.

6.4 Funktionsanpassung mit *RooFiLab*

Obwohl im Prinzip einfach, stellt die Formulierung eines Problems in C++ und die komplexe Umgebung des *ROOT*-Frameworks eine relativ hohe Hürde dar. Eine Vereinfachung und Erweiterung der Funktionalität bringt das Programm *RooFiLab*, dass dem Anwender in den meisten Fällen die Erstellung von eigenem C++-Code erspart.

[3] `http://root.cern.ch/`

Basierend auf *ROOT* stellt das in Karlsruhe entwickelte Programm *RooFiLab* eine Erweiterung zur Durchführung von Parameter-Anpassungen auf der Basis der χ^2-Methode zur Verfügung. Die Anwendung für den Einsteiger wird durch eine strukturierte grafische Oberfläche erleichtert.

In den in *ROOT* direkt oder in Programmpaketen wie z. B. *gnuplot* verfügbaren Anpassungsmethoden werden Kovarianzmatrizen nicht berücksichtigt. *RooFiLab* erweitert *ROOT* um diese Funktionalität und stellt eine vereinfachte, strukturierte grafische Benutzeroberfläche zur Verfügung, mit deren Hilfe Funktionsanpassungen an Messwerte unter Berücksichtigung korrelierter Fehler der Ordinate und der Abszisse möglich werden. In allgemeinen Fall können kompliziertere Fehlermodelle durch das Einlesen von explizit angegebenen Kovarianzmatrizen in der Anpassung verwendet werden. Für den einfachen Fall gemeinsamer absoluter oder relativer Fehler aller Messwerte enthält das Programm eine vereinfachte Eingabemöglichkeit. Ein Beispiel ist in Abbildung 7 gezeigt.

Eine hohe Flexibilität bei der Definition der anzupassenden Funktion wird einerseits durch die Interpreter-Funktion von *ROOT* erreicht, die die Eingabe von Funktionen in einer einfachen Formelsprache erlaubt. Daneben können aber auch komplexere, als C- oder C++-Code implementierte Funktionen zur Laufzeit des Programms eingebunden werden.

Die Elemente der grafischen Oberfläche (siehe Abbildung 8) des Programms *RooFiLab* und die Steuerung über die Eingabedatei sind im Handbuch `RooFiLab.pdf` im Verzeichnis `RooFiLab/doc` beschrieben. Ein knapper Überblick wird im Folgenden gegeben.

6.4.1 Installation

RooFiLab wird komplett installiert in einer virtuellen Maschine auf Ubuntu-Basis [4] bereit gestellt. Das gepackte Festplattenabbild kann mit Hilfe des frei verfügbaren Virtualisierers *VirtualBox* auf gängigen Linux-Distributionen, Windows-Versionen ab Windows XP und Macintosh-Computern mit Betriebssystem Mac OS X ausgeführt werden (siehe URL http://www.virtualbox.org sowie gesonderte Anleitung und Hilfe-Datei zur Virtuellen Maschine, `http://www-ekp.physik.uni-karlsruhe.de/~quast`).

Der Programm-Code von *RooFiLab* findet sich unter der gleichen URL in der gepackten Datei `RooFiLab.tar.gz`. Unter Linux ist damit die Installation mit Hilfe der im Unterverzeichnis `RooFiLab` enthaltenen Quelldateien möglich. Die Datei `Makefile` enthält alle notwendigen Anweisungen zur Erzeugung der ausführbaren Datei durch einfaches Aufrufen von `make`. Dazu muss eine *ROOT*-Installation vorhanden und initialisiert sein, d. h. die Umgebungsvariable `PATH` muss den Pfad zu den ausführbaren *ROOT*-Dateien und die Umgebungsvariable `LD_LIBRARY_PATH` den Pfad zu den *ROOT*-Bibliotheken enthalten.

6.4.2 Bedienung des Programms *RooFiLab*

RooFiLab bietet dem Anwender eine in zwei Programmfester untergliederte Oberfläche: Im rechten (Haupt-) Fenster erfolgt die Steuerung des Programms, während in dem anderen Fenster die entsprechende Grafik dargestellt wird. Die Steuerung ist in vier Shutter untergliedert. Die Eingaben bzw. Aktionen der jeweiligen Shutter,

- Einlesen der Daten und Definition der Fit-Parameter,
- Festlegen sinnvoller Anfangswerte sowie „Fit-by-Eye“,
- Durchführen der Anpassung, ggf. mit Fixierung einzelner Fit-Parameter,
- Festlegen von Optionen und Bearbeitung der Ausgabe-Grafik

sollten nacheinander ausgeführt werden.

[4] `http://www-ekp.physik.uni-karlsruhe.de/~quast/VMroot`

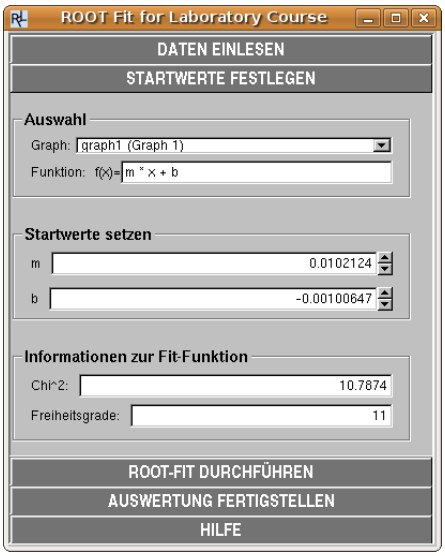

Abbildung 8: Die graphische Oberfläche von *RooFiLab*.

Zur Ausführungszeit stehen einige Funktionen von *ROOT* zur Verfügung, insbesondere für die grafische Darstellung, die interaktive Manipulation und den Export der erstellten Grafiken. Dies wird durch Öffnen der Kontext-Menüs von *ROOT* durch Rechtsklicks in die entsprechenden Komponenten der Grafik und mit Hilfe der Toolbar im oberen Graphikfenster ermöglicht.

Neben der Nutzung der Steuerelemente der grafischen Oberfläche können Anpassungen auch automatisiert durch die Angabe von Steueroptionen in der Eingabedatei ausgeführt werden, in der auch die Datenpunkte definiert werden. Die zunächst in einer interaktiven Anpassung ermittelten Eingabeoptionen können so in der Eingabedatei archiviert und für wiederholte, automatisierte Anpassungen verwendet werden.

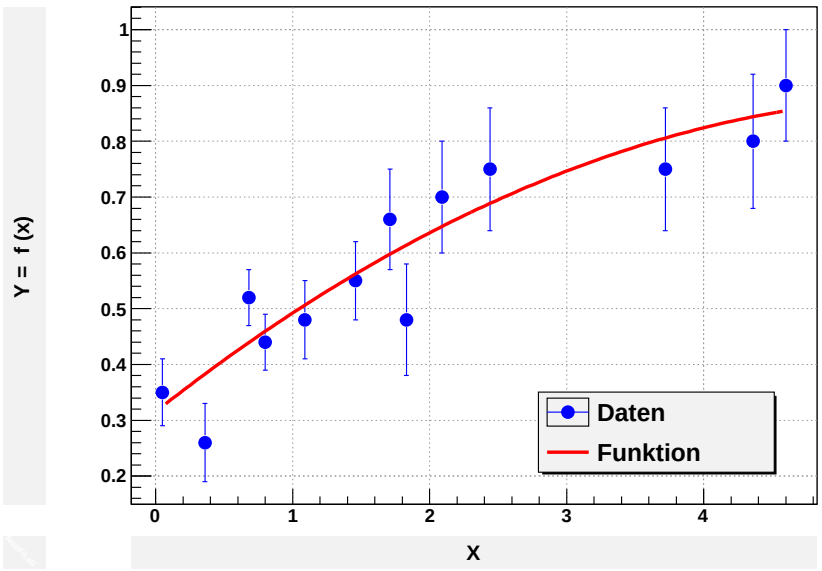

Abbildung 1: Beispiel der Anpassung einer Funktion (hier $f(x) = ax^2 + bx + c$) an Messpunkte (x_i, y_i).

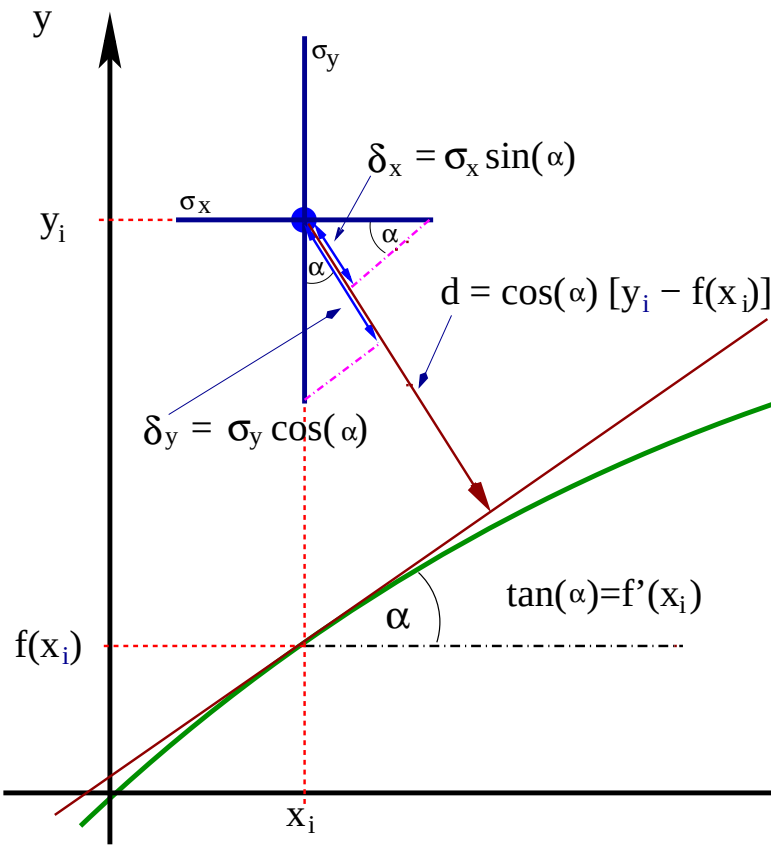

Abbildung 2: Illustration des Abstandsmaßes für einen Punkt mit Fehlern in Richtung der Ordinaten- und Abszissenachse. Das im Text beschriebene Verfahren entspricht der Minimierung des normierten Abstands der Messpunkte $(x_i,\, y_i)$ von der Tangente durch den Punkt $(x_i,\, f(x_i))$

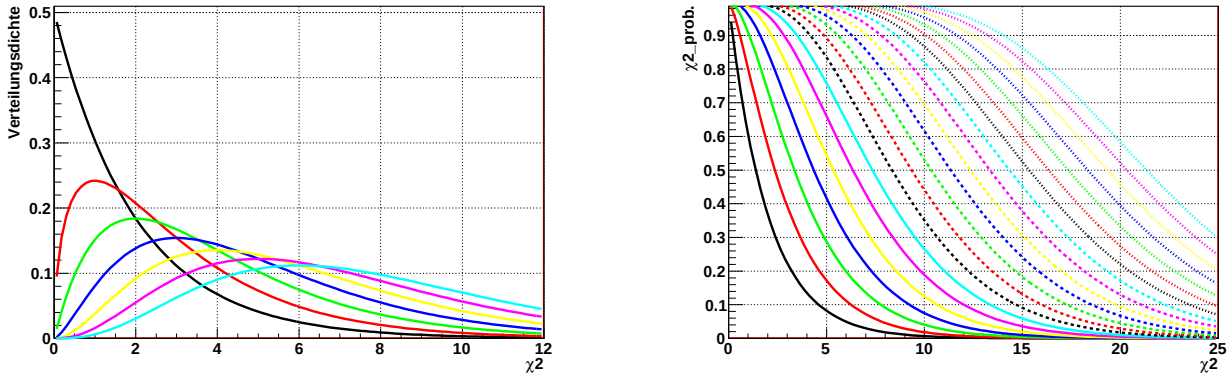

Abbildung 3: χ^2-Verteilung (links) für 2 – 8 Freiheitsgrade und χ^2-Wahrscheinlichkeit für 2 – 22 Freiheitsgrade (rechts). Der Erwartungswert der χ^2-Verteilung ist n_f und ihre Standardabweichung ist $\sqrt{2n_f}$. Für eine sehr große Anzahl von Freiheitsgraden geht die Verteilung in eine Gauß-Verteilung über. Durch Integration der Verteilungsdichte erhält man die im Text definierte χ^2-Wahrscheinlichkeit als Testgröße für die Güte einer Anpassung.

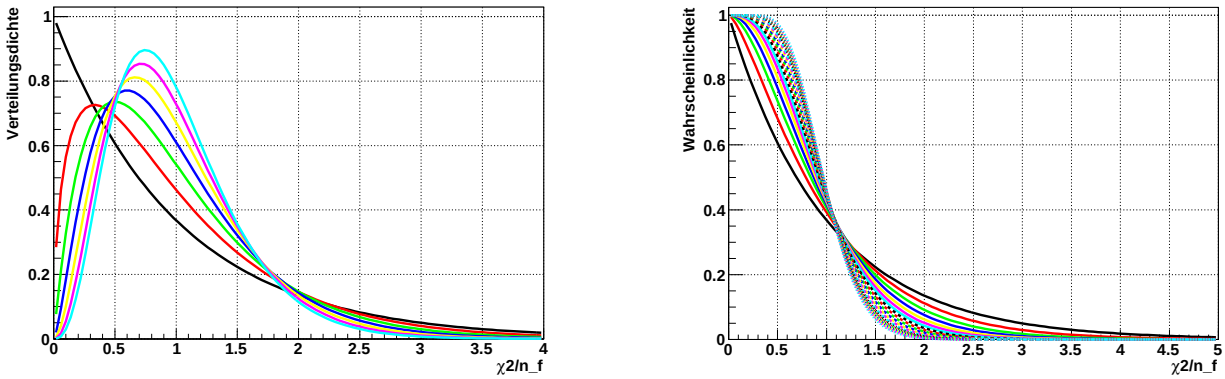

Abbildung 4: Verteilungsdichte von χ^2/n_f (links) für 2 – 8 Freiheitsgrade und die entsprechende Wahrscheinlichkeit für 2 – 22 Freiheitsgrade, einen größeren Wert als den auf der x-Achse angegebenen zu finden (rechts).

# x	y	dy
0,05	0,35	0,06
0,36	0,26	0,07
0,68	0,52	0,05
0,80	0,44	0,05
1,09	0,48	0,07
1,46	0,55	0,07
1,71	0,66	0,09
1,83	0,48	0,10
2,44	0,75	0,11
2,09	0,70	0,10
3,72	0,75	0,11
4,36	0,80	0,12
4,60	0,90	0,10

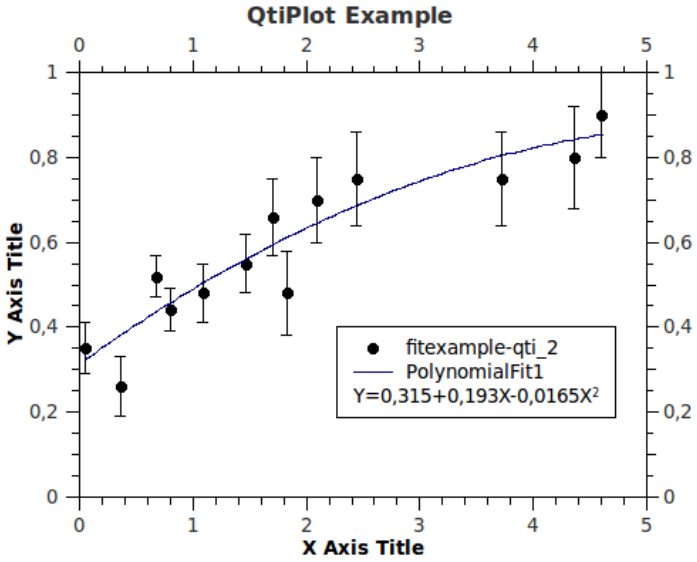

Abbildung 5: Beispiel für die Anpassung einer Parabel an Daten (links) mit *qtiplot*. Die Daten sind die gleichen wie in Abbildung 1, bei der Anpassung und grafische Ausgabe mit *RooFiLab* durchgeführt wurden.

```
# ---------- data ------
# x       y       ey
 .05     0.35    0.06
0.36     0.26    0.07
0.68     0.52    0.05
0.80     0.44    0.05
1.09     0.48    0.07
1.46     0.55    0.07
1.71     0.66    0.09
1.83     0.48    0.1
2.44     0.75    0.11
2.09     0.70    0.1
3.72     0.75    0.11
4.36     0.80    0.12
4.60     0.90    0.1
```

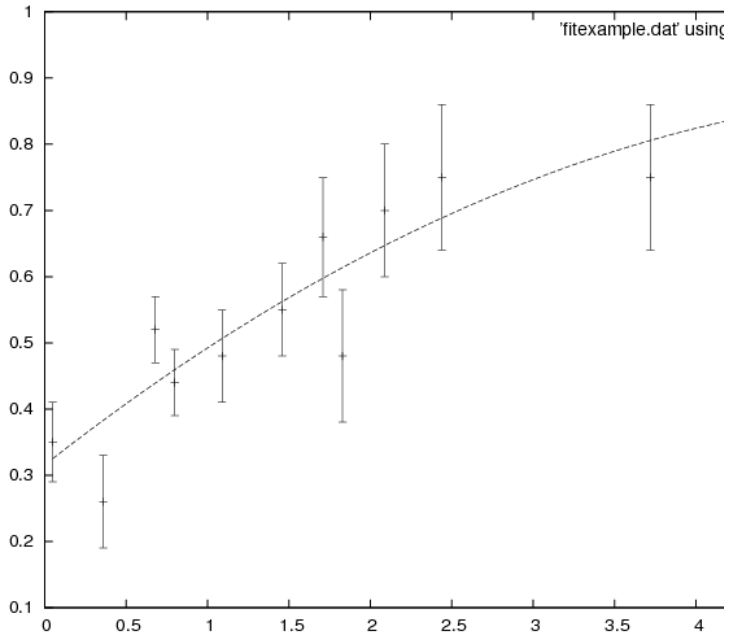

Abbildung 6: Beispiel für die Anpassung einer Parabel an Daten (links) mit *gnuplot*. Die Daten sind die gleichen wie in Abbildung 1, bei der Anpassung und grafische Ausgabe mit *RooFiLab* durchgeführt wurden.

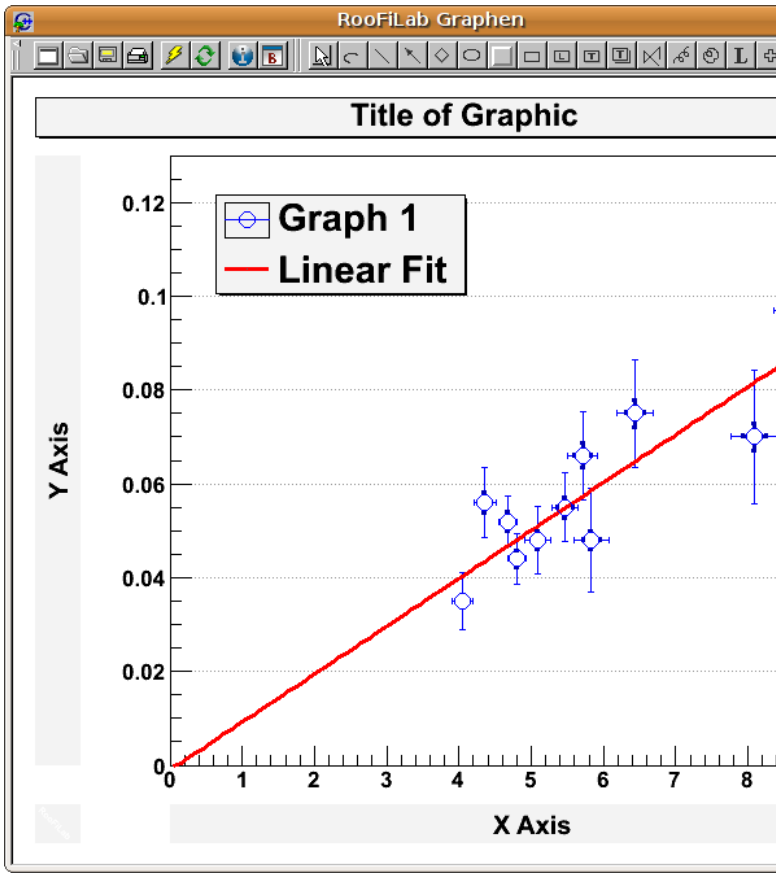

Abbildung 7: Beispiel einer Geradenanpassung mit (korrelierten) systematischen Fehlern in x- und y-Richtung.

Literatur

[1] G. Bohm, G. Zech, *Einführung in Statistik und Messdatenanalyse für Physiker*, DESY eBook, `http://www-library.desy.de/preparch/books/vstatmp.pdf`

[2] V. Blobel u. E. Lohrmann, *Statitische Methoden der Datenanalyse*, Teubner

[3] R. Barlow, *Statistics*, Wiley

[4] G. Cowen, *Statistical data analysis*, Oxford Science Publications

[5] S. Brandt, *Datenanalyse*, Spektrum akademischer Verlag

[6] W.H. Press, *Numerical Recipes in C++*, Cambridge University Press

[7] Data Analysis Framework ROOT; *Root Users Guide*, `http://root.cern.ch`
ROOT beginners guide: *Diving into Root*, D. Piparo, G. Quast,
`http://www-ekp.physik.uni-karlsruhe.de/~quast`

[8] Programm *qtiplot*, Home Page `http://soft.proindependent.com/qtiplot.html`, Information zu freien Version unter Linux siehe `http://wiki.ubuntuusers.de/qtiplot`

[9] Programm *gnuplot*, Home Page `http://www.gnuplot.info`

[10] Programm *RooFiLab*, Home Page des Autors:
`http://tmnet.cli.data-cmr.net/index.php/studium/programme/roofilab`

[11] Home Page G. Quast, dieses und weitere Skripte, Virtuelle Maschine, RooFiLab
`http://www-ekp.physik.uni-karlsruhe.de/~quast`

A Beispiele mit *RooFiLab*

Die folgenden Unterkapitel enthalten einfache Beispiele, die die Verwendung von *RooFiLab* illustrieren und als Basis für eigene Anwendungen dienen können.

A.1 Geradenanpassung an Messdaten mit Fehlern in x und y

Dieses Eingabedatei für *RooFiLab* enthält zahlreiche Kommentarzeilen und dokumentiert damit an einem einfachen Beispiel die verfügbaren Optionen. Durch die Zeile `#! dofit = true` wird eine automatisierte Anpassung mit in den entsprechenden Zeilen mit der Zeichenfolge `#!` gesetzten Optionen ausgeführt.

Interessant an diesem Beispiel ist es, die übliche Geradengleichung `m*x+b` durch `m*(x-5)+b` zu ersetzen - dies führt zu deutlich reduzierten Korrelationen der angepassten Parameter.

```
# straight-line fit to data with errors in x and y, incl. simple correlations
# ============================================================================
#! staterrors = xy
#! systerrors = 0.02 0.04 rel rel

#! fit = "m*x+b" "m,b" "roofilab.fit"
### fit = "m*(x-5.)+b" "m,b" "roofilab.fit" # führt zu kleineren Korrelationen
#! initialvalues = 0.015 0

### Befehl zum Ausf\"uhren des Fits
#! dofit = true

#! secondgraph = syst

#! title = "Anpassung an Daten mit korrelierten Fehlern"
#! graphlegend = "Daten" bottom right
#! functionlegend = "Funktion" bottom right
#! xaxis = "X-Werte"
#! yaxis = "Y-werte bzw. f(x)"

#! markersettings = 1.5 4 24
#! functionsettings = 1 3 2
#! grid = y
#! logscale = 0
#! savegraphic = "roofilab.eps"

# =================Eingabe der Daten ==================================
# values in up to four columns separated by whitespaces
#       (except for linebreaks or linefeeds)
# x       y      ex      ey
4.05 0.035 0.12 0.006
4.36 0.056 0.13 0.007
4.68 0.052 0.09 0.005
4.80 0.044 0.09 0.005
5.09 0.048 0.14 0.007
5.46 0.055 0.14 0.007
5.71 0.066 0.17 0.009
5.83 0.048 0.21 0.011
6.44 0.075 0.22 0.011
8.09 0.070 0.28 0.014
8.72 0.097 0.32 0.016
9.36 0.080 0.37 0.018
9.60 0.120 0.39 0.020
```

A.2 Einfache Mittelung von korrelierten Messungen

Auch die Mittelung von Messdaten entspricht formal einer χ^2-Anpassung. Dazu wird als Funktion einfach eine Konstante gewählt. Die im Beispiel gezeigten Messungen sind vier individuelle Messungen der Masse des Z-Bosons am Beschleuniger LEP des CERN. Der allen Messungen gemeinsame Fehler von 1.7 MeV rührt von Unsicherheiten der Schwerpunktesenergie des Beschleunigers her. Dieser Fehler wird in der Zeile
`#! systerrors = 0 0.0017 abs abs` spezifiziert.

```
# Mesurements of Z-Mass by AELPH, DELPHI, L3 and OPAL
# ---------------------------------------------------

# graphics options
#! markersettings = 1.5 4 24
#! functionsettings = 1 3 3
#! grid = y
# logscale = 0
# savegraphic = "roofilab.eps"
# saverfl = "data.rfl"

# plot lables
#! title = "averaging measurements"
#! xaxis = "n"
#! yaxis = "Mass of Z boson"
#! graphlegend = "Z mass measurements" bottom right
#! functionlegend = "average Z mass" bottom right

# fit control
#! fit = "m" "m" "mittelung.fit"
#! initialvalues =  91.2
#! dofit = true

#! staterrors = y # control-command
#! systerrors = 0 0.0017 abs abs
# the data, LEP electroweak working group, CERN 2000
1 91.1893 0.0031
2 91.1863 0.0028
3 91.1894 0.0030
4 91.1853 0.0029
```

A.3 Polynom-Anpassung an Datenpunkte mit Poisson-Fehlern

In diesem Beispiel wird die Anpassung eines Polynoms vierten Grades an Datenpunkte mit unkorrelierten Fehlern gezeigt. Diese Aufgabe entspricht der Anpassung einer Winkelverteilung an Daten eines Streuexperiments, bei dem (recht seltene) Ereignisse unter verschiedenen Winkeln gemessenen werden. Die Fehler sind daher durch die statistischen Fehler dominiert, die für jeden Messpunkt mit n_i beobachteten Ereignissen durch $\sqrt{(n_i)}$ gegeben sind. Obwohl die Fehler nicht Gauß-förimg sind, liefert eine χ^2-Anpassung oft akzeptable Ergebnisse.

Bei sehr kleiner Zählstatistik sollte allerdings als Fehlerverteilung die Poisson-Verteilung zu Grunde gelegt werden. Dies ist mit einer Log-Likelihoodanpassung in *ROOT* möglich. Für die im Bin i eines Histogramms beobachtete Zahl an Ereignissen wird die Likelihood $\mathcal{L}_i = (\mathrm{Poisson}(\mathrm{n_i}, \mu(\mathrm{x_i}; \boldsymbol{p}))$ berechnet, n_i Ereignisse zu beobachten, wenn $\mu(x_i, \boldsymbol{p})$ erwartet wurden. Die Gesamtlikelihood ergibt sich als Produkt über alle Bins, dessen negativer Logarithmus $-\ln(\mathcal{L}) = -\log(\prod \mathcal{L}_i)$, bzgl. der Parameter $\boldsymbol{p}$ minimiert wird. In *RooFiLab* kann diese Option mit der Zeile `#! fitmethod = likelihood` gewählt werden; in diesem Fall werden die angegebenen statistischen Fehler ignoriert bzw. können auch weggelassen werden. Aus technischen Gründen (wegen der Verwendung der Histogramm-Klasse TH1) müssen die x-Werte äquidistant gewählt werden.

```
############################################################
#     example: fit of an angular distribution
############################################################

# plot commands
#! title = "angular distribution "
#! xaxis = "cos(theta)"
#! yaxis = "number of events"
#! graphlegend ="observed rate " top left
#! functionlegend ="fitted cos(theta) distribution " top left
#! markersettings = 1.5 2 5
#! functionsettings = 1 3 3
# fit control
#! fit = "a4*x^4+a3*x^3+a2*x^2+a1*x+a0" "a0,a1,a2,a3,a4" "v_vs_cost.fit"
#! dofit = true

# fitmethod = likelihood # uncomment to perform a Log Likelihood fit

# definition of data
#! staterrors = y
# cost   N    sqrt(N)
-0.9    81.    9.0
-0.7    50.    7.1
-0.5    35.    5.9
-0.3    27.    5.2
-0.1    26.    5.1
 0.1    60.    7.7
 0.3   106.    10.3
 0.5   189.    13.7
 0.7   318.    17.8
 0.9   520.    22.8
```

A.4 Mittelung von korrelierten Messungen mit Kovarianz-Matrix

Als Beispiel für die Berücksichtigung von Korrelationen werden hier acht Messungen der Masse des W-Bosons miteinander kombiniert. Die Messungen der vier LEP-Experimente in jeweils zwei Endzuständen haben unterschiedliche, teilweise zwischen allen Messungen oder nur innerhalb der entsprechenden Endzustände korrelierte Unsicherheiten. Es ist deshalb notwendig, die komplette 8×8-Kovarianz-Matrix zu spezifizieren (s.u.), die in 4×4-Blockmatrizen zerfällt. Das geschieht in der Zeile

```
 #! covmatrices = 0 wmass.cov .
```

```
# Mesurements of W-Mass by AELPH, DELPHI, L3 and OPAL
# ---------------------------------------------------
# ### example of fit with covariance matrix#
# --- graphics options
#! markersettings = 1.5 4 24
#! functionsettings = 1 3 3
#! grid = y
# logscale = 0
# savegraphic = "graph.eps"
# plot lables
#! title = "averaging measurements"
#! xaxis = "n"
#! yaxis = "Mass of W boson"
#! graphlegend = "W mass measurements" top right
#! functionlegend = "average W mass" top right
# --- fit control
#! fit = "m" "m" "Wmittelung.fit"
#! initialvalues =  80.5
#! dofit = true
# --- the data (LEP electroweak working group, CERN 2006)
#! staterrors = 0
#! systerrors = 0 0 abs abs
#! covmatrices = 0 wmass.cov
1 80.429  0.059 # qqlv ALEPH
2 80.340  0.076 # qqlv DELPHI
3 80.213  0.071 # qqlv L3
4 80.449  0.062 # qqlv OPAL
5 80.475  0.082 # qqqq ALEPH
6 80.310  0.102 # qqqq DELPHI
7 80.323  0.091 # qqqq L3
8 80.353  0.081 # qqqq OPAL
```

```
//file wmass.cov
  0.003481 0.000316 0.000316 0.000316 0.000383 0.000383 0.000383 0.000383
  0.000316 0.005776 0.000316 0.000316 0.000383 0.000383 0.000383 0.000383
  0.000316 0.000316 0.005041 0.000316 0.000383 0.000383 0.000383 0.000383
  0.000316 0.000316 0.000316 0.003844 0.000383 0.000383 0.000383 0.000383
  0.000383 0.000383 0.000383 0.000383 0.006724 0.001741 0.001741 0.001741
  0.000383 0.000383 0.000383 0.000383 0.001741 0.010404 0.001741 0.001741
  0.000383 0.000383 0.000383 0.000383 0.001741 0.001741 0.008281 0.001741
  0.000383 0.000383 0.000383 0.000383 0.001741 0.001741 0.001741 0.006561
```