Dokumentation

12. Februar 2010

RooFiLab

Root Fit for Laboratory courses

Thomas Müller
Thomas.Mueller@student.kit.edu

Inhaltsverzeichnis

1 Überblick über die Funktionalität von *R*ooFiLab

Ausgehend vom Datenanalysewerkzeug und objektorientierten Framework *R*OOT[1] stellt das Programm *R*ooFiLab eine Erweiterung zur Durchführung von Parameter-Anpassungen („Fits“) zur Verfügung. Dabei wurde insbesondere auf typische Anforderungen aus den Anfänger- und Fortgeschrittenenpraktika Wert gelegt.

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, einer symmetrischen Matrix, deren Dimension der Anzahl der Messungen entspricht; die Diagonalelemente enthalten die Varianzen, d.h. den quadrierten Gesamtfehler der Messwerte, $\sigma_i^t \cdot \sigma_j^t$, die Nebendiagonalelemente enthalten die gemeinsamen Fehlerkomponenten, $\sigma_i^g \cdot \sigma_j^g$ der Messungen mit den Indizes i und j.

In den über die grafische Benutzerschnittstelle von *R*OOT verfügbaren Anpassungsmethoden oder ähnlichen Programmpaketen wie z. B. *g*nuplot werden Kovarianz-Matrizen nicht berücksichtigt. Das vorliegende Programmpaket erweitert *R*OOT um diese Funktionalität und stellt eine vereinfachte 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.

Eine hohe Flexibilität bei der Definition der anzupassenden Funktion wird einerseits durch die Interpreter-Funktion von *R*OOT 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.

*R*ooFiLab 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.

Zur Ausführungszeit stehen einige Funktionen von *R*OOT zur Verfügung, insbesondere für die grafische Darstellung, die interaktive Maniplulation und den Export der erstellten Grafiken. Dies wird durch Öffnen der Kontext-Menüs von *R*OOT durch Rechtsklicks in die entsprechenden Komponenten der Grafik und mit Hilfe der Toolbar im oberen Graphikfenster ermöglicht (zur näheren Erläuterung s. 7.2).

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.

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

2 Installation

Diese Version von *RooFiLab* wird komplett installiert in einer virtuellen Maschine auf Ubuntu-Basis[2] bereit gestellt.

Unter Linux ist 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.

2.1 Hinweis zur Programmsprache

Es gibt zwei Sprachversionen von RooFiLab für die graphische Oberfläche: Deutsch und Englisch. Alles andere (z.B. Befehle in der `rfl`-Datei und Ausgabe der Fit-Ergebnisse in der Fit-Logdatei) gibt es standardmäßig auf englisch. Vor dem Kompilieren kann man die Sprache wählen, indem man die Datei `language/Language.h` in der gewünschten Weise verändert. Dazu braucht man nur die Zeile mit der richtigen Sprache auskommentieren und die andere kommentieren. Nach solch einer Einstellung muss das Programm neu kompiliert werden, damit die Änderungen übernommen werden.

3 Die grafische Oberfläche - Shutter 1: Daten

3.1 Verwalten von Graphen mit den zugehörigen Fehlern

3.1.1 Messdaten aus einer Datei einlesen

Üblicherweise werden Messdaten aus einer Textdatei (Endung *.rfl* empfohlen) in das Programm importiert. Dabei können zwei bis vier durch Tabs oder Leerzeichen getrennte Spalten verarbeitet werden. Standardmäßig werden nur die ersten beiden Spalten eingelesen, die die Messwerte für Abszisse und Ordinate in dieser Reihenfolge enthalten. Über den Button *Graph hinzufügen* kann dieser Import erfolgen. Eine andere Möglichkeit besteht darin, RooFiLab schon mit *rfl*-Dateien als Parameter zu starten.

```
./RooFiLab <rfl-Datei Graph 1> <rfl-Datei Graph 2> ...
```

Will man nun auch unkorrelierte (oft statistische) Fehler laden, muss man einen Steuerbefehl (beginnend mit `#!`) die *rfl*-Datei über den Messwerten aufnehmen:

```
#! staterrors = <Format>
```

Hier kann der Platzhalter `<Format>` vier mögliche Werte annehmen: `0`, `x`, `y` oder `xy`. Diese Werte versteht das Programm als Konstanten und müssen tatsächlich so eingegeben werden. Also z.B. keine Leerzeichen zwischen x und y in `xy` verwenden. Die Einstellungen sind selbsterklärend, weil sie genau die statistischen Fehler mit einlesen, die angegeben wurden. `0` ist die Standardeinstellung. Bei `x` und `y` wird jeweils die dritte Spalte mit eingelesen.

Alle fehlerhaften Zeilen und zusätzliche Spalten in den Messwerten werden vom Programm ignoriert, wobei bei fehlerhaften Zeilen eine Warnmeldung erscheint.

Hinweise:

- Die Benutzung von eigenen Kovarianzmatrizen schließt alle anderen Fehlereingaben aus. Will man also später Kovarianzmatrizen einlesen können, müssen statistische deaktiviert werden.

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

- *RooFiLab* sortiert die Werte beim Import nach den Abszissenwerten, falls sie noch nicht sortiert waren. Wenn man eine eigene Kovarianzmatrix einlesen will, müssen die Werte schon in der *rfl*-Datei sortiert vorliegen, da sonst die Zuordnung der Kovarianzmatrixelemente nicht möglich ist.

3.1.2 Eingabe von Systematischen Fehlern

Unter systematischen Fehlern versteht *RooFiLab* korrelierte Fehler und bietet für diese ein spezielles vereinfachtes Fehlermodell an, welches in Praktikumskurse oft sinnvolle Verwendung findet. Es gibt die Möglichkeit, absolute oder relative korrelierte Fehler auf die Messwerte anzunehmen, wozu der Bereich *Systematische Fehler* im ersten Shutter dient.

Im Fall von absoluten systematischen Fehlern bekommen alle Messwerte den selben korrelierten Fehler $\Delta q \geqslant 0$. Im Fall von relativen systematischen Fehlern bekommen alle Messwerte (x_i, y_i) den selben korrelierten relativen Fehleranteil $0 \leqslant \Delta q_{\mathrm{rel}} \leqslant 1$, wobei z.B. für die Ordinate $\Delta q_y = y_i \cdot \Delta q_{y_{\mathrm{rel}}}$ den absoluten Fehler darstellt, mit dem intern weitergerechnet wird. Die Umschaltung des Wertebereichs erfolgt im Programm mit der Umschaltung der Art der systematischen Fehler. Zur Weiterverarbeitung dieser Fehlerwerte siehe Kapitel 5.1.2.

Hinweise:

- Wenn schon eine Kovarianzmatrix eingelesen wurde, können keine anderen Fehler mehr verarbeitet werden und die entsprechenden Felder im Programm werden solange deaktiviert, bis die Kovarianzmatrizen wieder gelöscht werden.
- Zu große systematische Fehler gegenüber den statistischen Fehlern führen zu Problemen beim Fit.

3.1.3 Einlesen von benutzerdefinierten Kovarianzmatrizen

Ganz allgemein kann *RooFiLab* auch benutzerdefinierte Kovarianzmatrizen verarbeiten, wobei die enthaltenen Fehler als Gesamtfehler auf die Messwerte angesehen werden, was alle anderen Fehler ausschließt. Zum (mathematischen) Aufbau von Kovarianzmatrizen siehe Kapitel 5.1.3. Zum Importieren gibt es zwei Buttons (*X-Kovarianzmatrix* und *Y-Kovarianzmatrix*), wobei man eine Kovarianzmatrix für jede Koordinatenrichtung einlesen kann. Über den Button *Matrizen löschen* werden alle benutzerdefinierten Matrizen aus dem Programm entfernt, was zu Folge hat, dass dem Programm danach keine Fehler auf die Messwerte bekannt sind.

RooFiLab erwartet eine quadratische und symmetrische (wird bisher noch nicht vom programm überprüft) Kovarianzmatrix mit einer Dimension, die der Anzahl der Messwerte entspricht in einer Textdatei (Endung *.cov* empfohlen). Spalten müssen dabei wieder durch Whitespaces (Tab oder Leerzeichen) und Reihen per Zeilenumbruch getrennt werden. Im Fall von fehlerhaften Dateien erscheint eine Warnmeldung.

Hinweise:

- Kovarianzmatrizen können nur eingelesen werden, wenn keine anderen Fehler betrachtet werden. Wenn die entsprechenden Buttons also deaktiviert sind, müssen zur Aktivierung die systematischen Fehler auf Null gesetzt werden und statistische Fehler in der *rfl*-Datei deaktiviert werden.
- Es ist nicht möglich, für eine Koordinatenrichtung benutzerdefinierte Kovarianzmatrizen und für die andere das vereinfachtes Fehlermodell zu verwenden.
- Bestehen schon benutzerdefinierte Kovarianzmatrizen im Programm, werden diese beim erneuten Einlesen überschrieben.
- Es ist möglich, für eine Koordinatenrichtung eine Kovarianzmatrix und für die andere eine Korrelationsmatrix einzulesen.

3.1.4 Einlesen von benutzerdefinierten Korrelationsmatrizen

Ähnlich wie im vorangegangenen Abschnitt lassen sich auch (für Menschen besser lesbare) Korrelationsmatrizen anstatt von Kovarianzmatizen einlesen. Das Vorgehen ist analog. Allerdings muss beachtet werden, dass eine Korrelationsmatrix keine tatsächlichen Fehlerwerte enthält. Deshalb müssen die Gesamtfehler in einer extra Spalte der Korrelationsmatrix vorangestellt werden:

```
<Ges.-Fehler 1> <Cor 1,1> ... <Cor 1,n>
...
<Ges.-Fehler n> <Cor n,1> ... <Cor n,n>
```

Hinweise:

- Korrelationsmatrizen können nur eingelesen werden, wenn keine anderen Fehler betrachtet werden. Wenn die entsprechenden Buttons also deaktiviert sind, müssen zur Aktivierung die systematischen Fehler auf Null gesetzt werden und statistische Fehler in der *rfl*-Datei deaktiviert werden.
- Es ist nicht möglich, für eine Koordinatenrichtung benutzerdefinierte Korrelationsmatrix und für die andere das vereinfachtes Fehlermodell zu verwenden.
- Bestehen schon benutzerdefinierte Korrelationsmatrizen im Programm, werden diese beim erneuten Einlesen überschrieben.
- Es ist möglich, für eine Koordinatenrichtung eine Korrelationsmatrix und für die andere eine Kovarianzmatrix einzulesen.

3.2 Vorbereitungen für den Fit

3.2.1 Spezifizieren der Funktion

Um einen Fit durchführen zu können, muss das Programm eine Funktion mit mindestens einem freien Parameter kennen. Die Eingabe einer Funktion erfolgt im Bereich des ersten Shutters unter *Vorbereitung für Fit*. Das Programm erwartet eine gültige *ROOT*-Syntanx[3] entsprechend der *ROOT*-Klasse `TFormula` mit einer die Bedienung erleichternden Ausnahme: *RooFiLab* kann mit alphanumerischen Parametern umgehen und erwartet nicht zwingend die Nummerierungsschreibweise in eckigen Klammern von *ROOT*.

Für das Spezifizieren der Funktion gibt es zwei Eingabefelder: *Funktion* und *Parameter*. Im ersten wird die Funktion in Abhängigkeit von der Variablen x und den Parametern eingegeben. Im zweiten Feld wird eine Parameterliste erwartet, damit *RooFiLab* die Parameter in *ROOT*-Syntax übersetzen kann. Dazu müssen die Parameternamen mit Komma, Semikolon oder Leerzeichen bzw. Tab getrennt eingegeben werden.

Beispiel:

Die beiden Eingabefelder *Funktion* und *Parameter* könnten für eine Exponentialfunktion folgendermaßen aussehen:

```
a * exp( -x / x0 )
a, x0
```

Bei Initialisieren der Funktion werden alle Parameter mit dem Wert 1.0 vorbelegt und die Funktion gezeichnet. Im Fall von unkorrekter Syntax erscheint eine Warnmeldung.

Hinweise:

[3] siehe `http://root.cern.ch/root/html/TF1.html` und `http://root.cern.ch/root/html/TFormula.html`

- Die alphanumerischen *Roofilab*-Parameter müssen mit einem Buchstaben beginnen und können danach eine beliebige Buchstaben- und Zahlenkombination enthalten.
- Um die Verarbeitung der Eingabedaten dieser der Felder *Funktion* und *Parameter* zu veranlassen, muss mindestens ein Feld mit `Enter` bestätigt werden.
- Wir eine Funktion geändert, müssen ebenfalls beide Felder geändert werden, weil z.B. ein neuer Parameter hinzugekommen sein könnte. Sind keine Änderungen in einem der beiden Felder dazu nötig, so muss dieses trotzdem mit `Enter` bestätigt werden.
- Die Standard-*ROOT*-Syntax für Parameter ist natürlich auch möglich, wobei das *Parameter*-Feld auch dann nicht leer bleiben darf. Man könnte z.B. ein Leerzeichen eingeben oder Namen für die Parameter wählen.
- Shutter 2 (*Startwerte festlegen*) und 3 (*ROOT-Fit durchführen*) werden erst nach erfolgreich initialisierter Funktion freigeschaltet.

3.2.2 Bemerkungen zur Formelsyntax (in *ROOT*)

ROOT kennt die üblichen einfachen Funktionen:

```
sin, cos, tan, exp, asin, acos, atan, sinh, cosh, tanh, asinh, acosh, atanh, log, sqrt
```

Diese Basisfunktionen können mit mathematischen Operationen verknüpft oder beliebig verkettet werden. Zum Potenzieren kennt der Formelinterpreter von *ROOT* die beiden Varianten `<Basis>^<Exponent>` oder `<Basis>**<Exponent>` , wobei die erste Möglichkeit auf der deutschen Tastatur unter *ROOT* unter Umständen nicht zu Verfügung steht.

Neben den ober erwähnten einfachen Basisfunktionen stehen außerdem die Funktionen aus der Klasse `TMath`[4] zur Verfügung, die mit `TMath::<Funktion>(<Argumente>)` verwendet werden werden können. Über die Steueroption

```
#! loadfunction = <Datei mit Funktion in C/C++-Code>
```

in der Eingabedatei kann auch eine beliebige, vom Anwender in C/C++ implementierte Funktion geladen werden, die dann über ihren Funktionsnamen im Eingabefeld für den Formelinterpreter verfügbar ist. Die Optionen sind so engestellt, dass die Funktion mit dem System-Compiler übersetzt und dann als Bibliothek dynamisch geladen wird, statt sie im *ROOT*-Interpreter auszuführen; auf diese Weise lassen sich auch komplexe numerische Berechnungen effizient ausführen.

3.2.3 Fit-Log

Im dem Eingabefeld `Fit-Logdatei` kann man einen Dateinamen für die Logdatei angeben, in der die einzelnen Schritte einer jeden Anpassung nachvollzogen werden können. In diese Datei werden sowohl die jeweiligen Startwerte als auch die Ergebnisse jedes Fits abgelegt bzw. an vorhandene Einträge angehängt. Nähere Informationen zu dieser Datei finden sich in Kapitel 5.2.8.

4 Die grafische Oberfläche - Shutter 2: Startwerte

Der zweite Shutter ermöglicht die Einstellungen der Startparameter für den Fit.

Hinweise:

[4] siehe `http://root.cern.ch/root/html/TMath.html`

- Der zweite und dritte Shutter werden nur freigeschaltet, wenn der aktuell ausgewählte Graph eine gültige Fit-Funktion enthält.
- Das Feld *Funktion* kann nicht editiert werden. Dazu muss der erste Shutter benutzt werden.

4.1 Einstellen der Startwerte

In dem Bereich *Startwerte setzen* kann man mit Reglern jeden Parameter mit einem sinnvollen Wert versehen, der dann als Startwert für den Fit benutzt wird. Wird ein Wert verändert, so wir auch die Funktion im Graphen-Fenster direkt neu gezeichnet.

Hinweis:

- Es ist möglich, Zahlen mit der üblichen wissenschaftlichen Darstellung im C/C++-Stil, z.B. `4.2e-5`, einzugeben.

4.2 Verwendung für „Fit-by-Eye“

Um das Fitten manuell zu testen, gibt es neben den Einstellmöglichkeiten für die Parameter auch ein Ausgabefeld für das χ^2, welches ein Maß für den Abstand der Funktion von den Messwerten darstellt, und eines für die Anzahl der Fit-*Freiheitsgrade* mit $ndf = N_{\text{Messwerte}} - N_{\text{freie Fitparameter}}$. Im Idealfall gilt $\frac{\chi^2}{ndf} \approx 1$. So ist es möglich, sich ein Bild über den Verlauf der χ^2-Funktion und ihrer Minima zu machen.

5 Die grafische Oberfläche - Shutter 3: Fit

Der dritter Shutter dient ausschließlich zum Starten der Fit-Routine über den entsprechenden Button. Alle Eingabefelder sind nicht editierbar, sondern dienen nur der Ausgabe. Die Eingabe ist nicht deaktiviert, um das Markieren und Kopieren der Inhalte per Tastenkombination `Strg + C` zu erlauben.

Hinweis:

- Der zweite und dritte Shutter werden nur freigeschaltet, wenn der aktuell ausgewählte Graph eine gültige Fit-Funktion enthält.

5.1 Mathematische Grundlagen

Um ein Verständnis von dem Ablauf der Fit-Routine, dem Kernstück des Programms, zu erhalten, werden diesem Kapitel einige mathematische Formalitäten vorangestellt, die wichtig für einen sinnvollen Umgang mit dem Programm sind.

5.1.1 χ^2-Methode zur Funktionsanpassung

RooFiLab verwendet zur Anpassung von Funktionen an Messdaten die Methode der kleinsten Quadrate, d.h. die quadratische Abweichung der Datenpunkte von der Fit-Funktion normiert auf die jeweiligen Messfehler, auch χ^2-Methode genannt.

Allgemein mit Hilfe der Kovarianzmatrix $\mathbf{C}$ geschrieben ergibt sich

$$\chi^2(p_1, p_2, \ldots) = \mathbf{\Delta}(p_1, p_2, \ldots)^T \, C^{-1} \, \mathbf{\Delta}(p_1, p_2, \ldots) \tag{1}$$

mit dem Residuenvektor

$$\Delta_i(p_1, p_2, \ldots) = y_i - \mathrm{f}(x_i; p_1, p_2, \ldots) \tag{2}$$

und der inversen Kovarianzmatrix C^{-1}.

Die Minimierung von χ^2 erfolgt numerisch mit Hilfe des in *ROOT* integrierten Programmpakets MINUIT.

Unsicherheiten der Datenpunkte bzgl. der Abszissenachse werden durch Iteration berücksichtigt: zunächst erfolgt eine Anpassung ohne Abszissenfehler, im zweiten Schritt werden diese mit Hilfe der ersten Ableitung der im ersten Schritt angepassten Funktion in entsprechende Fehler der Ordinate umgerechnet und quadratisch zu den betreffenden Kovarianzmatrixelementen C^y_{ij} addiert, d.h. die Elemente der gesamten Kovarianzmatrix **C** ergeben sich zu

$$C_{i,j} \;=\; C^y_{i,j} \;+\; C^x_{i,j} \cdot f'(x_i) \cdot f'(x_j)\,. \tag{3}$$

Mit dieser neuen Kovarianzmatrix wird schließlich die Anpassung wiederholt. 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.

5.1.2 Aufbau der Kovarianzmatrizen im Programm

Statistische Fehler σ_q sieht *RooFiLab* immer als unkorrelierte Fehler an. Das bedeutet, dass es keinen statistischen Fehler gibt, der mehr als einen Messwert gleichzeitig betrifft. Deshalb muss man die statistischen Fehler quadriert auf die Diagonale der (statistischen) Kovarianzmatrix schreiben:

$$C^{\mathrm{stat}}_{i,j} = \sigma^2_{q_i}\,\delta_{ij} \tag{4}$$

Unter korrelierten Fehlern versteht *RooFiLab* systematische Fehler Δq. Hier soll nur auf das vereinfachte Fehlermodell mit gleichen absoluten oder relativen systematischen Fehlern auf alle Messwerte eingegangen werden. Der allgemeine Fall wird im nächsten Abschnitt (5.1.3) angesprochen.

Im Fall von einem absoluten Fehler Δq auf alle Messwerte wird dieser quadriert an jeder Position der systematischen Kovarianzmatrix eingetragen, weil man annimmt, dass alle Messwerte zum einen den gleichen systematischen Fehler haben (Diagonale) und auch untereinander die gleiche Fehlerabhängigkeit oder Korrelation aufweisen. Man setzt voraus, dass man, wenn man einen Fehler kennt, auch weiß, dass der nächste Messwert den gleichen Fehler hat, weil man ihn beispielsweise mit dem selben Messgerät auf die gleiche Weise gemessen hat.

$$C^{\mathrm{syst}}_{i,j} = (\Delta q)^2 \tag{5}$$

Liegt aus dem Experiment nur ein relativer systematischer Fehler Δq_{rel} auf alle Messwerte vor, so muss dieser zuerst mit dem jeweiligen Messwert q_i skaliert werden, um wieder alle absoluten Fehler als Kovarianzmatrixelemente zu erhalten:

$$C^{\mathrm{syst}}_{i,j} = (\Delta q_{\mathrm{rel}})^2 \cdot q_i\; q_j \tag{6}$$

Eine Kovarianzmatrix für die Gesamtfehler erhält man nun einfach als Summe der beiden Kovarianzmatrizen, da diese schon quadratische Elemente enthalten:

$$C^{\mathrm{ges}} = C^{\mathrm{stat}} + C^{\mathrm{syst}} \tag{7}$$

Alle diese Berechnungen führt *RooFiLab* bei passenden Eingaben automatisch aus, sodass sich der Benutzer nicht darum kümmern muss.

5.1.3 Anleitung zum Aufbau von benutzerdefinierten Kovarianzmatrizen

Da *RooFiLab* auch benutzerdefinierte Kovarianzmatrizen verarbeiten kann, folgt hier eine kurze Einführung in die Aufstellung von Kovarianzmatrizen, die sich nicht wesentlich von den oben beschriebenen Vorgängen unterscheidet.

Ein Kovarianzmatrix C ist eine quadratische und symmetrische Matrix, deren Dimension die Anzahl der Messwerte n hat. Jede Spalte bzw. Zeile mit dem Index i steht dabei für den Messwert q_i.

$$C \in \mathbb{R}^{n \times n} \quad \text{mit} \quad C^T = C \tag{8}$$

Alle unkorrelierten Fehler gehören auf die Diagonale der Matrix an die entsprechende Stelle des Messwerts. Wenn jeder Messwert q_i einen unkorrelierten Fehler σ_{q_i} hat, sieht eben die zugehörige Kovarianzmatrix folgendermaßen aus:

$$C_{i,j}^{\text{stat}} = \sigma_{q_i}^2 \, \delta_{ij} \tag{9}$$

Bei den korrelierten Fehlern wollen wir etwas allgemeiner als im letzten Abschnitt bleiben: Wir nehmen an, wir könnten einen systematischen Fehler Δq_i für die Messwerte q_i mit $i \in [a, b]$ und $1 \leqslant a \leqslant b \leqslant n$, also nur für einen (geordnete) Untermenge der Messwerte. Dann ergibt sich die zugehörige Kovarianzmatrix folgendermaßen:

$$C_{i,j}^{\text{syst}} = \begin{cases} \Delta q_i \cdot \Delta q_j & \text{falls } i, j \in [a, b] \\ 0 & \text{sonst,} \end{cases} \tag{10}$$

, d. h. eine Blockmatrix mit den quadrierten gemeinsamen Fehlern. Dies lässt sich natürlich auch auf den Fall anwenden, dass $a = 1$ und $b = n$ gilt, wenn wir alle Fehler auf alle Messwerte mit der gleichen Korrelation kennen. Oft hat man jedoch mehrere Messwerte in unterschiedlichen Messbereichen eines Messgeräts mit unterschiedlichen korrelierten Fehlern gemessen. Dann erhält man für jeden Messbereich solch eine Blockmatrix innerhalb der Kovarianzmatrix.

Die Gesamtfehler-Kovarianzmatrix erhält man durch Addition aller so berechneten Kovarianzmatrizen. Wenn Messungen von mehreren Fehlern betroffen sind, so werden einfach die entsprechenden Elemente der Kovarianzmatizen der Felherbeiträge addiert. Dies entspricht der quadratischen Addition von Einzelfehlerbeiträgen – Kovarianzmatrixelemente sind quadratische Formen!

5.1.4 Korrelationsmatrizen

RooFiLab kann auch mit Korrelationsmatrizen anstatt von Kovarianzmatizen umgehen. Deshalb soll hier noch kurz erklärt werden, wie diese beiden Matrizen auseinander folgen.

Korrelationsmatrizen sind für Menschen leichter lesbar, da sie nicht die abstrakten Fehlerwerte in quadrierter Form enthalten, sondern nur ein Maß für den Korrelationsgrad der Fehler zweier Messwerte darstellen. Aus diesem Grund müssen die Gesamtfehler auf jeden Messwert (die quadriert auf der Diagonale einer Kovarianzmatrix stehen) gesondert mitgeliefert werden.

$$Cor_{i,j} = \frac{Cov_{i,j}}{\sqrt{Cov_{j,j} \cdot Cov_{j,j}}} \quad \Leftrightarrow \quad Cov_{i,j} = \underbrace{\sqrt{Cov_{j,j} \cdot Cov_{j,j}}}_{\sigma_{q_i}^{\text{ges}} \cdot \sigma_{q_j}^{\text{ges}}} \cdot Cor_{i,j} \tag{11}$$

Anhand der Korrelationsmatrix lässt sich auch leicht überprüfen, ob eine solche Matrix korrekt ist, da Korrelationsmatrizen wie Kovarianzmatizen symmetrisch sein müssen. Außerdem liegen alle Werte c einer Korrelationsmatrix im Intervall $-1 \leqslant c \leqslant 1$. und alle Diagonalelemente sind eins.

5.2 Fit per Root-Funktionalität

RooFiLab ist nur für die Aufstellung der χ^2-Funktion, siehe Gleichung (1), zuständig. Die Minimierung dieser Funktion in Abhängigkeit bzgl. der Fitparameter und die Berechnung der Fehler auf die Parameter erledigt das in *ROOT* intgegrierte Programmpaket *MINUIT* mittels verschiedener numerischer Verfahren. *RooFiLab* definiert eine von *MINUIT* verwendedte χ^2-Funktion, die korrelierte Fehler der Ordinate ermöglicht und mittels eines Iterationsverfahrens auch Fehler der Abszissen-Werte berücksichtigen kann. Durch Ausführen von Anpassungen ohne Berücksichtigung von systematischen Fehlern zusätzlich zu Anpassungen mit den gesamten Fehlern kann in *RooFiLab* auch der Anteil systematischer Fehler am Gesamtfehler quantifiziert werden.

5.2.1 Fit ohne Fehler auf die Messwerte

Werden keine Fehler in das Programm importiert, also sowohl keine statistischen oder systematischen Fehler als auch keine benutzerdefinierten Kovarianzmatrizen, so gibt es nur eine sinnvolle Gewichtung des Residuenvektors, siehe Gleichung (2), in der χ^2-Funktion: Alle müssen gleich gewichtet sein, weil es keine bessere Information gibt. Das wird einfach mit einer Einheits-Kovarianzmatrix erreicht.

RooFiLab führt nun diese Fit-Routine einmal mit der Einheits-Kovarianzmatrix durch und ermittelt dabei nur die Werte der Parameter. Fehler auf die Parameter zu bestimmen wäre sinnlos, weil diese von den willkürlich gewählten Gewichten (= 1) auf der Diagonale der Kovarianzmatrix abhängen. Insofern werden bei einem Fit ohne Eingabefehler auch keine Fehler auf die Parameter ausgegeben.

5.2.2 Fit mit statistischen Fehlern auf die Messwerte

Wenn allerdings statistische Fehler eingegeben wurden, so wird obige Einheitsmatrix durch die statistische Kovarianzmatrix C^{stat} aus Gleichung (4) ersetzt und ein Fit mit dieser Diagonalmatrix durchgeführt. Als Ergebnis für die Parameter kann man nun auch einen statistischen Fehler erhalten.

Hinweis:

- Gibt es nur statistische Fehler auf die Abszissenwerte, so wird zuerst ein Fit ohne Fehler durchgeführt (siehe Abschnitt 5.2.1), um eine Funktion zum Projizieren der Abszissenfehler zu erhalten (siehe Abschnitt 5.1.1). Danach wird die Einheits-Kovarianzmatrix durch die Kovarianzmatrix der projizierten Abszissen-Kovarianzmatrix ersetzt.

5.2.3 Fit mit systematischen Fehlern auf die Messwerte

Wenn man nur korrelierte Fehler auf die Messwerte annimmt, kann man sich leicht klarmachen, dass die Kovarianzmatrix C^{syst} aus Gleichung (5) singulär wird, sodass eine Invertierung mathematisch nicht möglich ist. Deshalb wird in diesem Fall die Ausführung der Fit-Routine Probleme ergeben. Dieser Fit sollte vermieden werden.

Hinweis:

- Invertierungsprobleme können auch auftreten, wenn die statistischen Fehler gegenüber den systematischen vernachlässigbar klein werden (siehe Abschnitt 5.2.4).

5.2.4 Fit mit statistischen und systematischen Fehlern auf die Messwerte

Wenn sowohl statistische als auch systematische Fehler auf die Messwerte eingegeben wurden, führt *RooFiLab* die Fit-Routine zweimal hintereinander aus, wobei nur die Kovarianzmatrix in der χ^2-Funktion, siehe

Gleichung (1), ersetzt wird: Der erste Fit wird mit der statistischen Kovarianzmatrix C^{stat} aus Gleichung (4) und der zweite mit der Gesamtfehler-Kovarianzmatrix C^{ges} aus Gleichung (7) berechnet.

Als Parameterwerte werden dann die Ergebnisse des zweiten Fits ausgegeben. Für die Parameterfehler entstehen zwei Sätze von Fehlerwerten: $\sigma_{p_i}^{\text{stat}}$ und $\sigma_{p_i}^{\text{ges}}$. Als statistische Parameterfehler werden die des ersten Fits, also $\sigma_{p_i}^{\text{stat}}$, ausgegeben:

$$\sigma_{p_i} = \sigma_{p_i}^{\text{stat}} \tag{12}$$

Die systematischen Parameterfehler ergeben sich dann aus der quadratischen Differenz der beiden Fehler:

$$\Delta p_i \equiv \sigma_{p_i}^{\text{syst}} = \sqrt{\left(\sigma_{p_i}^{\text{ges}}\right)^2 - \left(\sigma_{p_i}^{\text{stat}}\right)^2} \tag{13}$$

Hinweis:

- Falls die korrelierten Fehler gegenüber den unkorrelierten zu groß sind, kann es Invertierungsprobleme mit der Gesamtfehler-Kovarianzmatrix oder unerwartete Ergebnisse auf die Fitparameter geben.

5.2.5 Fit mit benutzerdefinierten Kovarianzmatrizen/Korrelationsmatrizen

Die letzte und allgemeinste Möglichkeit, Fits mit *RooFiLab* durchzuführen, ist die Nutzung eigener Kovarianzmatrizen. Hier werden dann nur die in den Kovarianzmatrizen enthaltenen Fehler in den Fit einbezogen. Dementsprechende ergibt die Fit-Routine auch nur einen Fehler auf die Fitparameter, die dann als Gesamtfehler gekennzeichnet werden. Das Projizieren der Abszissen-Kovarianzmatrix geschieht analog zu dem Vorgehen in den anderen Fit-Methoden (siehe Abschnitt 5.1.1).

Hinweis:

- Gibt es nur eine Abszissen-Kovarianzmatrix, so wird zuerst ein Fit ohne Fehler durchgeführt (siehe Abschnitt 5.2.1), um eine Funktion zum Projizieren der Abszissen-Kovarianzmatrix zu erhalten (siehe Abschnitt 5.1.1). Danach wird die Einheits-Kovarianzmatrix durch die Kovarianzmatrix der projizierten Abszissen-Kovarianzmatrix ersetzt.

5.2.6 Likelihood-Fit mit poissonverteilten Messwerten

Eine letzte Fit-Möglichkeit rundet das Angebot ab. Wenn man an gebinnte Daten einer Zählstatistik eine Funktion anpassen will, bietet sich der Likelihood-Fit an, wo die Messwerte nicht als gaußverteilt mit der Breite der jeweiligen Messfehler angesehen werden, sondern eine Poisson-Verteilung als Statistik für die Ordinatenwerte N_i mit dem Fehler $\sqrt{N_i}$ ansetzen kann. Diese Fit-Möglichkeit erreicht man unter dem dritten Shutter in der Auswahlbox *Fit-Methode*.

Bei einem solchen Fit ist die Einteilung der Abszissenachse in Bins entscheidend für die Qualität des Ergebnisses. Der Einfachheit halber ist das Programm auf äquidistante Abszissenwerte optimiert, was bedeutet, dass es für jeden Messwert einen Bin gibt. Im Idealfall landet jeder Abszissenwert somit in der Mitte eines eigenen Bins.

Hinweise:

- Wir ein Likelihood-Fit ausgewählt, so werden alle eingegebenen Fehler auf die Messwerte ignoriert und durch ein Poisson-Verteilung in der Likelihood-Funktion ersetzt. Daher sind auch keine Abszissenfehler verarbeitbar. Der Fehler auf die Fitparameter sind als statistische Fehler zu interpretieren.
- Der Fit liefert grundlegend falsche Ergebnisse, wenn die Abszissenwerte nicht äquidistant vorliegen. Es ist also sinnvoll, die nicht äquidistante Daten vorher in Bins zu sortieren und sie dann in Einheiten von Bins in *RooFiLab* einzulesen.

5.2.7 Festhalten von Parametern

Manchmal kann es erforderlich sein, einzelne Fitparameter während eines Fits auf einem bestimmten Wert festzuhalten. Dazu dienen die „Fix“-Checkboxen neben den Parameterfeldern auf dem dritten Shutter. Sinnvoll wird dies zum Beispiel dann, wenn man keine guten Startwerte für alle Parameter kennt und der Fit immer wieder in ein lokales Minimum der χ^2-Funktion springt. Ein andere Fall ergibt sich, wenn mindestens zwei Parameter so stark korreliert sind, dass die Fit-Routine sie nicht sinnvoll trennen kann.

Hinweise:

- Wir ein Parameter festgehalten, liefert die Fit-Routine keine korrekten Fehler. Es ist also zu empfehlen, den letzten Fit zu einem Problem immer ohne festgehaltene Parameter durchzuführen.
- Ist mindestens ein Fitparameter festgehalten, so erfolgt keine Ausgabe der Fit-Ergebnisse in der Fit-Logdatei, um Fehlern vorzubeugen.
- Es können selbstverständlicherweise nicht alle Parameter gleichzeitig festgehalten werden. Wird dies trotzdem versucht, liefert das Programm Fehler oder stürzt ab.

5.2.8 Ausgabe der Fit-Ergebnisse in der Fit-Logdatei

Jede erfolgreich beendete Fit-Routine erzeugt einen Eintrag in der (im ersten Shutter ausgewählten) Fit-Logdatei. Dabei steht die Information zu einem Fit immer zwischen zwei Reihen aus Gleichheitszeichen und wird an die bestehenden Einträge angehängt. Jeder Eintrag kann wieder in sechs Bereiche untergliedert werden:

1. Unter *RooFiLab-Fit* wird die Funktion und die eingegebene Parameterliste aufgeführt, was helfen soll, mehrere Fits in einer Datei unterscheiden zu können.
2. Unter *Startwerte für den Fit* sind die verwendeten Startwerte einzusehen.
3. Unter *Fit-Probleme* werden eventuelle Probleme während des Fits gespeichert. In der aktuellen Version sind das nur die Ausgaben, die auch die Fehlermeldungen enthielten.
4. Der Bereich *Fit-Ergebnisse für die Parameter* enthält die Fitparameter-Ergebnisse in der ausführlichsten Form in einer Art Matrixdarstellung. Jede Zeile enthält einen Parameter. Die zweite Spalte enthält die Ergebnisse mit den eventuellen Fehlern und die letzte Spalte die Korrelationsmatrix der Fitparameter. Die Werte hier liegen immer zwischen -1 und 1 und sollten bei idealer Parametrisierung der Fit-Funktion in der Nähe von 0 liegen. Liegen sie in der Nähe von ± 1, lohnt es sich, über eine geeignetere Parametrisierung nachzudenken, so dass die Werte der einzelnen Fitparameter nicht so stark voneinander abhängen.
5. Im Abschnitt *Formatierte Fit-Ergebnisse (Text)* werden die Fit-Ergebnisse in ausführlicher textueller Form wiedergegeben.
6. Zuletzt unter *Formatierte Fit-Ergebnisse (Latex)* werden die Fit-Ergebnisse in ausführlicher textueller Form und in Latex-Code wiedergegeben.

Hinweis:

- Die Werte in den letzten beiden Bereichen der textuellen Ausgabe werden automatisch (oft sinnvoll) gerundet. Fall die Anzahl der Stellen nicht optimal ist, muss auf eigentlichen Ergebnisse im Bereich darüber zurückgegriffen werden.

6 Die grafische Oberfläche - Shutter 4: Fertigstellen

Der vierte Shutter ermöglicht sämtliche „gestalterischen“ Einstellungen am Graphen vorzunehmen. Als Ergänzung sei hier schon auf die Toolbar im Graphen-Fenster hingewiesen, die ebenfalls Zeichenoptionen ermöglicht. Allerdings sollten diese direkt vor dem Abspeichern der Graphik getätigt werden, da bei einem eventuellen Neuzeichnen der Graphen durch *RooFiLab* alle Änderungen verloren gehen. Genauso funktioniert es mit den Einstellungen, die per Kontextmenüs im Graphen-Fenster verändert werden können.

Hier sollen nun kurz alle Funktionen von *RooFiLab* nach vier Bereichen untergliedert besprochen werden.

Hinweise:

- Der vierte Shutter wird nur freigeschaltet, wenn es initialisierte Graphen gibt.
- Veränderungen der Einstellungen im vierten Shutter ziehen (bisher) alle ein Neuzeichnen des Graphen nach sich. Das bedeutet dann, dass alle Änderung an der Abbildung, die nicht mit Funktionen von *RooFiLab* vorgenommen wurden, dabei verloren gehen.

6.1 Zusätzliche Fehlerbalken einzeichnen

Im ersten Bereich kann man einstellen, dass zusätzliche (innere) Fehler in dem Graphen eingezeichnet werden, wenn der Graph zwei Sorten von Fehlern enthält. Dabei wird dann ein Zusätzlicher Graph erzeugt, der etwas breitere und dunklere Fehlerbalken enthält. Bei der (graphischen) Interpretation ist zu beachten, dass die Gesamtfehler ja eine quadratische Summe darstellen, sodass bei kleinen „zweiten“ Fehlern diese etwas größer erscheinen. Standardmäßig werden keine zusätzlichen Fehlerbalken eingezeichnet.

Hinweise:

- Diese Optionen sind nur aktiviert, falls der aktuell ausgewählte Graph statistische und systematische Fehler auf die Messwerte enthält.
- Der zusätzliche Graph wird im Graphen-Fenster in einer Ebene über dem ursprünglichen Graphen gezeichnet. Will man also per Rechtsklick auf den ursprünglichen Graphen zugreifen, sollte man auf die äußeren Enden der größeren Fehlerbalken klicken.
- In der aktuellen Version gibt es noch Probleme bei der automatischen Speicherung einer Abbildung mit zusätzlichen Fehlerbalken. Falls dies nicht korrekt erfolgt, muss hier das Speichern manuell vorgenommen werden.

6.2 Beschriftung anpassen

Im zweiten Bereich kann die Beschriftung des Graphen festgelegt werden. Die Eingabefelder sind selbsterklärend.

6.3 Zeichenoptionen für Marker und Funktion setzen

Im dritten Bereich können die Zeichenoptionen für Marker (oben) und für die Fit-Funktion (unten) festgelegt werden. Die Bedienelemente sind selbsterklärend und können nicht falsch angewandt werden.

Hinweis:

- Die Auswahldialoge stellen nicht alle Möglichkeiten von *ROOT* dar. Die Einstellungen, die über die Klassen `TAttMarker`[5] und `TAttLine`[6], benutzbar sind, können in *RooFiLab* alle per Steuerbefehl (siehe Abschnitt 7.1) erreicht werden.

[5] `http://root.cern.ch/root/html/TAttMarker.html`
[6] `http://root.cern.ch/root/html/TAttLine.html`

6.3.1 Funktionsrange anpassen

Automatisch erstreckt sich der Definitionsbereich einer eingezeichneten Fit-Funktion nur über den jeweiligen zugrunde liegenden Datensatz der Messwerte. Will man dies ändern, kann man hier unter *Fkt.-Range* den Standardwert „*Graph*“ modifizieren. Mit „*Multigraph*“ erzeugt man das Zeichnen der Funktion über die gesamte Breite der Abszissenachse hinweg. Es gibt auch eine manuelle Einstellmöglichkeit, unter der dann auch die beiden Eingabefelder aktiviert werden.

6.4 Einstellungen zum Koordinatensystem

Zuerst gibt es hier die Möglichkeiten, das Grid und die logarithmische Achsenskalierung für jede Koordinatenachse getrennt ein- und auszuschalten. Standardmäßig ist das Grid komplett ein- und die logarithmische Achsenskalierung komplett ausgeschaltet.

Für den Zeichenbereich gibt es eine Auswahlbox mit folgenden vier Einstellungen:

- Im *automatisch*-Modus wird der Graph bezüglich seiner χ^2-Funktion skaliert. Das bedeutet, dass die Optimierung des Achsenranges[7] von *ROOT* nur dann aufgerufen wird, wenn man aufgrund des aktuellen Wertes der χ^2-Funktion darauf schließen kann, wenn sich die Fit-Funktion nahe bei den Messwerten befindet (siehe Abschnitt 5.1.1).

  $$\frac{\chi^2}{ndf} < 2 \tag{14}$$

  Ist diese Bedingung nicht erfüllt, wird eine manuelle Einstellung des Ranges vorgenommen (siehe Einstellung *Alles*). Gibt es noch keine Fit-Funktion, so wird immer die automatische Optimierung per *ROOT*-Funktionalität vorgenommen.

- Mit der Einstellung *optimieren* schaltet man auf ständige Optimierung per *ROOT*-Funktionalität ohne das Überprüfen der χ^2-Bedingung.

- Mit der Einstellung *alles zeigen* erreicht man eine Festlegung des Ordinaten-Ranges, bei der einfach das Minimum und das Maximum von Graphen und Fit-Funktionen als Grenzen angenommen wird. Das kann zu unschönen Ergebnissen im Fall von größeren Fehlerbalken führen.

- Unter *manuell* kann man den Zeichenbereich in Abszissen- als auch in der Ordinatenrichtung frei wählen. Dazu werden (nur hier) die entsprechenden vier Eingabefelder aktiviert.

6.5 Graphik speichern

Zuletzt gibt es unter dem vierten Shutter noch die Möglichkeit, den Inhalt des Graphen-Fensters (Abbildung) zu speichern. Es öffnet sich ein Datei-Speichern-Dialog, in dem ein Paar Dateiformate voreingestellt sind. Allerdings sind alle von *ROOT* unterstützten Formate auch hier möglich. Dazu gehören folgende Formate[8]:

`ps, eps, pdf, svg, gif, xpm, png, jpg, tiff, C, root, xml`

Hinweise:

- Der Dateiname sollte die Dateiendung immer schon enthalten, auch wenn das gewünschte Format schon in der Drop-Down-List ausgewählt ist, da ansonsten die Seiteneinstellungen im Dokument nicht richtig gesetzt werden.

[7] Aufruf der Methode `TAxis::UnZoom()`, siehe `http://root.cern.ch/root/html/TAxis.html#TAxis:UnZoom`
[8] siehe `http://root.cern.ch/root/html/TPad.html#TPad:SaveAs`

- Wurden in der Graphik viele Änderungen per Toolbar oder Kontextmenüs vorgenommen, wird empfohlen, die Abbildung als *ROOT*-Macro mit der Endung `.C` abzuspeichern. Im Idealfall lässt sich dieses Macro dann von *ROOT* per `root <Macroname>.C` ausführen. Oft müssen jedoch noch die Variablennamen in C-konformer Syntax umbenannt werden. Dieses Macro lässt sich dann leicht bearbeiten, wobei dann aber keine *RooFiLab*-Funktionalität mehr nutzbar ist.

7 Erweiterte Bedienung

7.1 Automatisierte Programmsteuerung mit erweiterter rfl-Datei

Für den häufigen Gebrauch kann es lästig werden, alle Einstellungen immer der Maus vornehmen zu müssen. Dafür schafft *RooFiLab* Abhilfe, indem es ermöglicht, die rfl-Dateien mit Befehlszeilen (beginnend mit `#!`) auszuwerten. Auch Kommentarzeilen sind mit `#` am Anfang der Zeile möglich.

Im Folgenden sollen nun alle zur Verfügung stehenden Befehle kurz erklärt werden. Da sie jedoch immer die selbe Aufgabe erledigen, wie sie auch per graphischer Oberfläche zu erreichen ist, werden hier nur die wichtigsten Informationen bezüglich der rfl-Datei genannt. Alles Weitere kann in den entsprechenden Abschnitten der Kapitel 3 bis 6 nachgelesen werden.

- `#! addgraph = "<rfl-Datei>"`

  Funktion: Einlesen einer weiteren Graphen-Datei
  Hinweis: Es wird empfohlen, diesen Befehl als erstes in einer `rfl`-Datei zu verwenden. Allerdings muss ein Graph (mit Werten) in jeder `rfl`-Datei enthalten sein. Globale Einstellungen wie Achsenbeschriftungen sollten dann im letzten so hinzugefügten Graphen eingefügt werden, da sie ansonsten von später hinzugefügten Graphen wieder verändert werden könnten.

- `#! staterrors = <Fehlerspalten>`

  Mögliche Werte für `<Fehlerspalten>`: `0`, `x`, `y` oder `xy`; Standard: `0`
  Funktion: Import von statistischen (unkorrelierten) Fehlern aus der Messwertedatei beim Einlesen.
  Hinweise: Unkorrelierte Fehler können nur auf diese Weise im Programm verarbeitet werden. Diese Fehlerwerte können dann im Programm auch nicht mehr verändert werden. Bei den Werten `x` und `y` werden die drei ersten Spalten aus der rfl-Datei eingelesen. Diese Zeile liefert eine Programm-Fehlerausgabe und wird ignoriert, wenn schon vorher Kovarianzmatrizen eingelesen wurden.

- `#! systerrors = <Abszissenfehler> <Ordinatenfehler> <Art Absz.-F.> <Art Ord.-F.>`

  Mögliche Werte für `<Art Fehler>`: `abs` oder `rel`; Standard: `0.0 0.0 abs abs`
  Funktion: Festlegung von systematischen (korrelierten) Fehlern für alle Messwerte.
  Hinweis: Im Fall von relativen Fehlern sollte der Wert zwischen 0 und 1 liegen. Diese Zeile liefert eine Programm-Fehlerausgabe und wird ignoriert, wenn schon vorher Kovarianzmatrizen eingelesen wurden.

- `#! covmatrices = <Absz.-Kovarianzmatrix-Datei> <Ord.-Kovarianzmatrix-Datei>`

  Funktion: Auswählen der Kovarianzmatrix-Dateien.
  Hinweis: Diese Zeile liefert eine Programm-Fehlerausgabe und wird ignoriert, wenn schon vorher andere Fehler spezifiziert wurden.

- `#! cormatrices = <Absz.-Korrelationsmatrix-Datei> <Ord.-Korrelationsmatrix-Datei>`

  Funktion: Auswählen der Korrelationsmatrix-Dateien.
  Hinweise: Diese Zeile liefert eine Programm-Fehlerausgabe und wird ignoriert, wenn schon vorher andere Fehler spezifiziert wurden. Die Datei enthält eine Spalte für die Gesamtfehler gefolgt von der Korrelationsmatrix

- `#! loadfunction = <C-Code-Datei>`

  Funktion: Eigene Funktionen als C/C++-Code importieren.

- `#! fit = ‘‘<Fit-Funktion>’’ ‘‘<Parameterliste>’’ ‘‘<Fit-Logdatei>’’`

  Funktion: Anlegen einer Fit-Funktion
  Hinweise: Die Fit-Funktion muss in Abhängigkeit von der Variablen x definiert werden. Die Fit-Logdatei wird relativ zur rfl-Datei angelegt.

- `#! initialvalues = <Parameterwert 1> <Parameterwert 2> ...`

  Standard: `1.0 1.0 ...`
  Funktion: Festlegen der Startparameter für den (ersten) Fit.

- `#! fitmethod = <Fit-Methode>`

  Mögliche Werte für *<Fit-Methode>*: `chisquare` oder `likelihood`; Standard: `chisquare`
  Funktion: Umstellung von χ^2- auf Likelihood-Fit.

- `#! dofit = <Automatischer Fit>`

  Mögliche Werte für *<Automatischer Fit>*: `true` oder `false`; Standard: `false`
  Funktion: Automatisches Aufrufen der Fit-Routine beim Laden der rfl-Datei.

- `#! secondgraph = <Zusätzliche Fehlerbalken>`

  Mögliche Werte für *<Zusätzliche Fehlerbalken>*: `0`, `stat` oder `syst`; Standard: `0`
  Funktion: Zusätzliche Fehlerbalken auswählen
  Hinweis: Diese Zeile wird ignoriert, falls es keine unterschiedlichen Fehlerarten im Graphen gibt.

- `#! title = ‘‘<Titel>’’`

  Funktion: Titel der Abbildung festlegen.

- `#! xaxis = ‘‘<Beschriftung Abszissenachse>’’`

  Funktion: Beschriftung der Abszissenachse in der Abbildung festlegen.

- `#! yaxis = ‘‘<Beschriftung Ordinatenachse>’’`

  Funktion: Beschriftung der Ordinatenachse in der Abbildung festlegen.

- `#! graphlegend = ‘‘<Legende>’’ <Vertikale Position> <Horizontale Position>`

  Mögliche Werte für *<Vertikale Position>*: `top` oder `bottom`; Standard: `top`
  Mögliche Werte für *<Horizontale Position>*: `left` oder `right`; Standard: `left`
  Funktion: Legende für den Graphen festlegen.
  Hinweis: Die Legende kann auch leer gelassen werden (**‘‘’’**), sodass kein Legendeneintrag erzeugt wird.

- `#! functionlegend = ‘‘<Legende>’’ <Vertikale Position> <Horizontale Position>`

  Mögliche Werte für *<Vertikale Position>*: `top` oder `bottom`; Standard: `top`
  Mögliche Werte für *<Horizontale Position>*: `left` oder `right`; Standard: `left`
  Funktion: Legende für die Funktion festlegen.
  Hinweis: Die Legende kann auch leer gelassen werden (**‘‘’’**), sodass kein Legendeneintrag erzeugt wird.

- `#! markersettings = <Größe> <Farbe> <Stil>`

  Funktion: Aussehen der Marker einstellen.
  Hinweis: Alle Parameter hier sind Zahlenwerte. Für Mehr Information siehe *ROOT*-Klasse `TAttMarker`[9]

- `#! functionsettings = <Stil> <Breite> <Farbe>`

  Funktion: Aussehen der Fit-Funktion einstellen.
  Hinweis: Alle Parameter hier sind Zahlenwerte. Für Mehr Information siehe *ROOT*-Klasse `TAttLine`[10]

- `#! functionrange = <Voreinstellung> <Startwert> <Endwert>`

  Funktion: Range (oder Definitionsbereich) der Fit-Funktion einstellen.
  Mögliche Werte für *<Voreinstellung>*: `graph`, `multigraph` und `man`
  Hinweis: Die letzten beiden Parameter sind Zahlenwerte und werden nur ausgewertet, falls die Voreinstellung `man` gewählt wurde.

- `#! grid = <Grid>`

  Mögliche Werte für *<Grid>*: `0`, `x`, `y` oder `xy`; Standard: `xy`
  Funktion: Grid ein- bzw. ausschalten.

- `#! logscale = <Logarithmische Achsenskalierung>`

  Mögliche Werte für *<Logarithmische Achsenskalierung>*: `0`, `x`, `y` oder `xy`; Standard: `0`
  Funktion: Logarithmische Achsenskalierung ein- bzw. ausschalten.

- `#! xyrange = <Voreinst.> <Startw. Absz.> <Endw. Absz.> <Startw. Ord.> <Endw. Ord.>`

  Funktion: Range (oder Definitionsbereich) der Fit-Funktion einstellen.
  Mögliche Werte für *<Voreinstellung>*: `auto`, `opt`, `all` und `man`
  Hinweis: Die letzten vier Parameter sind Zahlenwerte und werden nur ausgewertet, falls die Voreinstellung `man` gewählt wurde.

- `#! savegraphic = "<Datei>"`

  Funktion: Abbildung speichern.
  Hinweis: Die Bilddatei wird relativ zur rfl-Datei angelegt.

7.2 Hilfen zur Nutzung weiterer *ROOT*-Funktionalität

Weil *RooFiLab* auf *ROOT* basiert, ist es natürlich uneingeschränkt möglich, alle Funktionalität von *ROOT* auch hier zu nutzen, sofern es die *ROOT*-Syntax erlaubt. Dazu soll zuerst auf die *ROOT*-Dokumentation[11] hingewiesen werden. Nichtsdestotrotz werden hier einige wichtige Funktionen erläutert, die oft eine praktische Anwendung finden könnten.

[9] `http://root.cern.ch/root/html/TAttMarker.html`
[10] `http://root.cern.ch/root/html/TAttLine.html`
[11] User's Guide: `http://root.cern.ch/drupal/content/users-guide`
Dokumentation der Klassen: `http://root.cern.ch/root/html/ClassIndex.html`

7.2.1 Benutzung der Toolbar

Am oberen Rand des Graphen-Fensters findet man die Toolbar mit der man sich unter anderem zeichnerisch betätigen kann (Buttons rechts). Die per ToolTipText erklärten Buttons sind selbsterklärend.

Hinweis:

- Alle neu einzeichneten Objekte werden in *RooFiLab* nicht extra abgespeichert. Das hat zur Folge, dass diese Objekte und auch andere Veränderungen verloren gehen, wenn ein Graph von *RooFiLab* neugezeichnet wird. Das erfolgt in der aktuellen Version bei jeder Veränderung des Aussehens vom Graphen (hauptsächlich über den vierten Shutter). Es wird also empfohlen, zuerst alle Einstellungen in *RooFiLab* vorzunehmen und erst danach *ROOT*-Funktionalität anzuwenden.

7.2.2 Attribute eines Graphikobjekts ändern

Alle Objekte in der Abbildung sind per Maus verschiebbar und lassen sich in ihrer Größe ändern. Weitere Eigenschaften lassen sich per Kontextmenüeinträge `Set...Attributes` vornehmen. Hier öffnet sich dann ein neues Fenster mit den gewünschten Einstellmöglichkeiten. Per Kontextmenüeintrag `Delete` lässt sich ein Objekt auch ganz löschen. Der sich öffnende Dialog erfordert keine Eingabe sondern nur eine Bestätigung.

Genauso lassen sich in einem `TPaveText`-Objekt, z.B. der Legenden-Box, einzelne Einträge mit dem Kontextmenüeintrag `DeleteEntry` für die jeweilige Zeile entfernen. Den Text kann man zeilenweise mit `SetEntryLabel` editieren, wobei der ganze Eintrag leider immer neu eingegeben werden muss.

Hinweis:

- Es ist empfohlen, alle mit *RooFiLab* möglichen Einstellungen auch dort vorzunehmen, weil sie dann korrekt für den Programablauf gespeichert werden.

7.2.3 Einstellungen zum Koordinatensystem

Den Achsenbereich kann man verkleinern, indem man im Bereich der Achsenbeschriftung (Zahlen) die gewünschte Spanne bei gedrückter (linker) Maustaste markiert. Vergrößern lässt sich der Zeichenbereich mit dem Kontextmenüeintrag `SetRangeUser` unter den Zahlen. Im Fall von logarithmischer Achsenskalierung könnten die Kontextmenüeinträge `SetMoreLogLabels` und `SetNoExponent` hilfreich sein.

7.2.4 Ausführen eigener Befehle

Per Toolbar-Button `Browser` lässt sich der `TBrowser` von *ROOT* öffnen. Ab Version 5.25 verfügt *ROOT* über einen neuen Browser, der sich nur umständlich im alten über `file -> New Browser` starten lässt. Dieser aber ermöglicht es in einem Unterfenster, eigene Befehle auszuführen.

A Anhang - Beispiele

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

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

Dieses etwas lange Beispiel enthält zahlreiche Kommentarzeilen und dokumentiert damit an einem einfachen Beispiel alle verfügbaren Optionen. Durch die Zeile `#! dofit = true` wird eine automatisierte Anpassung mit in den entsprechenden Zeilen mit der Zeichenfolge `#!` gesetzten Optionen ausgeführt. Durch Abschalten der automatischen Aufsführung, d.h. entfernen des `!`-Zeichens in der Zeile `#! dofit = true`, kann das Beispiel auch interaktiv als Übung verwendet werden, um die Funktionen von *R*ooFiLab auszuprobieren.

```
# 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"
#! initialvalues = 0.015 0
#! dofit = true

#! secondgraph = syst

#! title = "Title of Graphic"
#! graphlegend = "Graph 1" bottom right
#! xaxis = "X Axis"
#! yaxis = "Y Axis"

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

# ================================================================================
# 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 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" 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 Mittelung von korrelierten Messungen mit Kovarianz-Matrix

In diesem Beispiel werden 8 Messungen der W-Bosonmasse 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" 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
```

A.4 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 in diesem Beispiel 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 $-\log(\mathcal{L}) = -\log(\prod \mathcal{L}_i)$, 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.5 Anpassung einer Bereit-Wigner-Verteilung

Die Datenpunkte entsprechen den gemittelten Messungen des hadronischen Wirkungsquerschnitts als Funktion der Schwerpunktsenergie in der Nähe der Z-Resonanz, die von den Experimenten ALEPH, DELPHI, L3 und OPAL am Elektron-Positron-Beschleuniger LEP am CERN in den Jahren von 1990-1995 durchgeführt wurden. Die Fit-Funktion ist eine modifizierte Breit-Wigner-Verteilung mit einer von der Schwerpunktsenergie abhängigen Breite. Neben den unkorrelierten Fehlern der Wirkungsquerschnitte werden hier gemeinsame systematische Fehler der Messungen der Wirkungsquerschnitte (relativ) und der Schwerpunktsenergie (absolut) berücksichtigt.[12]

```
# approximate average mesurements of hadronic cross sections
#   by AELPH, DELPHI, L3 and OPAL around the Z resonance
# ------------------------------------------------------------

# graphics options
#! markersettings = 0.2 4 24
#! functionsettings = 7 1 3
#! grid = y

# plot lables
#! title = "Cross section measurements"
#! xaxis = "Energy [GeV]"
#! yaxis = "cross section [nb]"
#! graphlegend = "Breit-Wigner" top left

# fit control commands
# BW with s-dependent width
#! fit = "s0*x^2*G^2/((x^2-M^2)^2+(x^4*G^2/M^2))" "s0,M,G" "BreitWigner.fit"
#! initialvalues =  41. 91.2 2.5

#! dofit = true

# LEP average sig_had, approx. radiative corrections applied
#! staterrors = xy
#! systerrors =  0.0017 0.0007 abs rel
# CMenergy  sig0_h  E err sig err
   88.387   6.803  0.005  0.036
   89.437  13.965  0.0015 0.013
   90.223  26.113  0.005  0.075
   91.238  41.364  0.003  0.010
   92.059  27.535  0.005  0.088
   93.004  13.362  0.0015 0.015
   93.916   7.302  0.005  0.045

# official results (Particle Data Group):
# s0=41.540+/-0.037nb  M=91.1875+/-0.0021GeV/c^2  G=2.4952+/-0.0023 GeV
```

[12] Dieses einfache Fehlermodell entspricht nur grob der Wirklichkeit, insbesondere wurde eine durch die Methode der Energiekalibration bedingte Antikorrelation der Energiewerte unter- und oberhalb vom Pol der Z-Resonanz vernachlässigt. Die Messdaten wurden näherungsweise auf radiative Effekte korrigiert, deren Berechnung die Kenntnis der Z-Masse voraussetzt.

A.6 Anpassung an Einschwingvorgang eines harmonischen Oszillators

Dieses Beispiel demonstriert die Anpassung von 8 Parametern an eine Funktion, die einem exponentiell gedämpften Abfall einer Schwingung mit Eigenfrequenz `omega0` und dem exponentiell ansteigenden Anteil der erzwungenen Schwingung der Frequenz `omega` entspricht. Die Datenpunkte entsprechen (noch) keinem realen Experiment, sondern sind künstlich. Die Anpassung einer solchen großen Zahl von Parametern wird mit hoher Wahrscheinlichkeit in einem Neben-Minimum enden - es ist daher wichtig, gute Startwerte zu setzen und einige der anzupassenden Parameter zunächst fest zu halten, bevor man in der Nähe des Minimums dann alle Parmeter in der Anpassung frei lässt. Dies kann mit *R*ooFiLab leicht interaktiv durchgeführt werden.

```
############################################################
#     example: forced harmonic oscillator
############################################################

# plot commands
#! title = "Forced Oscillation"
#! xaxis = "time [t]"
#! yaxis = "Amplitude"
#! graphlegend =" Amplitude" top right
#! markersettings = 1.5 2 5
#! functionsettings = 1 3 3
# fit control
#! fit = "A0*exp(-x/tau0)*cos(omega0*x+phi0)+A1*(1-exp(-x/tau))*cos(omega*x+phi)" "A0,tau0,omega0,phi
#                 A0 tau0 omega0  phi0  A1 tau omega    phi
#! initialvalues = 1  6    6.28    0.    1  2   3.14     0.
# dofit = true

# definition of data
#! staterrors = xy
# t          A    dt   dA
  0         1.    0.01  0.1
  0.25      0.0   0.2  0.05
  0.5      -0.9   0.1  0.09
  0.75      0.0 0.2  0.05
  1.        0.3   0.1  0.06
  1.25      0.0   0.2  0.05
  1.5      -0.6   0.1  0.06
  1.75      0.0   0.2  0.05
  2.        1.1   0.1  0.11
  3.       -0.4   0.1  0.054
  4.        1.1   0.1  0.1
  5.       -0.6   0.1  0.08
  6.        1.0   0.1  0.1
  7.       -0.7   0.1  0.08
  8.        1.0   0.1  0.1
  9.       -0.8   0.1  0.09
 10.        1.    0.1  0.01
 13.       -1.    0.1  0.1
 16.        1.    0.1  0.1
 19.       -1.    0.1  0.1
 22.        1.    0.1  0.1
```

A.7 Anpassung eines Doppelspalt-Interferenzmusters

Dieses Beispiel demonstriert die Anpassung einer als C-Code implementierten Funktion an Daten, die dem an einem Doppelspalt gemessenen Interferenzmuster entsprechen. Die Zeile

```
#! loadfunction = IDoubleSlit.C
```

lädt den in der angegebenen Datei enthaltenen C-Code zur Berechnung der Intensitätsverteilung des Interferenzmusters an einem Doppelspalt.

Die Anpassung kann nicht automatisiert durchgeführt werden, weil die Parameterliste die Wellenzahl und den Spaltabstand enthält, die beide das Interferenzmuster in gleicher Weise beeinflussen. Vor der Anpassung muss man sich also entscheiden, ob die Wellenzahl `k` oder Spaltbreite und Spaltabstand, `b`,`g`, angepasst werden sollen. Die jeweils anderen Parameter müssen dann im Tab "Root-Fit Ausführen" fixiert werden.

```
#     example: fit interference pattern from double slit
#! title = "Double Slit"
#! xaxis = "alpha [rad]"
#! yaxis = "intensity"
#! graphlegend =" " top left
#! markersettings = 1.5 2 5
#! functionsettings = 1 3 3
#! loadfunction = IDoubleSlit.C
#! fit = "I*IDoubleSlit(x,b,g,k)" "I,b,g,k" "doubleslit2.fit"
#                  I    b     g      k
#! initialvalues = 1. 15e-6 45e-6 12.57e6 # 500 nm wavelength
#! staterrors = xy
# alpha   I     dalhpa dI
-0.040    0.01  0.001  0.1
-0.036    0.05  0.001  0.05
-0.030    0.03  0.001  0.1
-0.024    0.2   0.001  0.05
-0.018    0.04  0.001  0.1
-0.012    0.8   0.001  0.05
-0.008    0.06  0.001  0.1
-0.004    0.3   0.001  0.05
 0.003    0.5   0.001  0.05
 0.008    0.04  0.001  0.1
 0.012    0.8   0.001  0.05
 0.018    0.05  0.001  0.1
 0.024    0.2   0.001  0.05
 0.030    0.04  0.001  0.1
 0.036    0.05  0.001  0.05
 0.040    0.01  0.001  0.1

//file IDoubleSlit.C
Double_t IDoubleSlit(Double_t x, Double_t b=15e-6, Double_t g=45e-6, Double_t k= 12.57e6)
// calculate intensity at double slit as a function of the angle
{ Double_t delta= k/2.* sin(x);
  Double_t gamma= delta * b;
  delta=delta*g;
  Double_t I2=cos(delta);
  Double_t I1 = ( abs(gamma) < 1e-4 ) ? 1.-gamma*gamma/6. : sin(gamma)/gamma;
  return I1*I1 * I2*I2; }
```