Statistische Methoden der Datenanalyse WS 2017/18

Prof. Dr. Ulrich Landgraf

Aufgabenblatt 9 vom 17.01.2018

Aufgabe 1 (Nichtlineare Parameterschätzung) (5 Punkte)

Wir behandeln das gleiche Problem wie bei dem Aufgabenblatt 8, nämlich die Analyse des Tones einer offenen Orgelpfeife. Wir wollen aber jetzt auch die Frequenz der Pfeife aus den Daten ermitteln, denn es könnte ja sein, dass die Pfeife etwas verstimmt ist und der Grundton gar nicht genau 440 Herz beträgt; dann hätten wir in der letzten Aufgabe wahrscheinlich nicht die korrekten Amplitudenverhältnisse ermittelt.

Wir verwenden wieder die gleichen Daten in der Datei `http://hep.uni-freiburg.de/tl_files/home/wwwherten/statistik/Aufg8.data`.

Lesen Sie zunächst die Daten wieder ein. Sie können natürlich dazu die gleiche Routine verwenden wie beim letzten Aufgabenblatt. Stellen Sie sie ebenfalls wieder grafisch mit der Klasse TGraph dar.

Wir werden nun die Klasse `TMinuit` verwenden. Dadurch, dass diese von der alten FORTRAN-Version des MINUIT-Algorithmus abgeleitet ist und auch interaktiv außerhalb von ROOT verwendet werden kann, hat diese Klasse ein etwas seltsames und auch etwas inkonsistentes Interface.

Als allererstest deklariert man (wie üblich) eine Instanz der Klasse TMinuit. Als Argument des Konstruktors wird dabei die Zahl der zu fittenden Parameter angegeben, also zum Bespiel `new TMinuit(5)`.

Der Benutzer muss eine Funktion schreiben, welche für jeden Satz der zu bestimmenden Parameter einen Wert an MINUIT zurück gibt. MINUIT variiert dann die Parameter so lange, bis er den Satz gefunden hat, bei dem der Funktionswert minimal wird. Bei der Methode der kleinsten Fehlerquadrate muss diese Funktion also die Variable X^2 berechnen.

Früher in Fortran musste die Funktion immer `fcn` heißen, in ROOT kann man sie aber beliebig nennen und den Name mit der Methode `SetFCN(Name)` an ROOT übergeben. Damit MINUIT die Routine richtig aufrufen kann, muss man sich aber genau an die Parameter halten:

```
void fcn(Int_t &npar, Double_t *derivative, Double_t &f,
                                            Double_t *par, Int_t flag)
```

Dabei sind für Sie nur die Argumente `&f` und `par` wichtig. `par` ist ein Array, in dem die Parameter übergeben werden, die MINUIT gerade testen will. Sie werden in der Funktion `fcn` mit `par[0], par[1]` etc. verwendet. Und `f` ist die Variable, in der Sie den Funktionswert zurückgeben müssen (daher das Zeichen `&` vor dem Variablennamen). Sie müssen also am Ende der Funktion `fcn` der Rückgabevariable mit `f= ...;` einen Wert zuweisen. Beachten Sie aber auch, dass Sie in der Funktion `fcn` einen Zugriff auf die eingelesenen Daten brauchen. Daher müssen Sie die entsprechenden Felder „global“ erklären, d.h. die Deklarationen ganz oben im Code *außerhalb* der {..}-Klammern plazieren!

Bitte wenden!

Nach dem Befehl `SetFCN` im Hauptprogramm müssen Sie MINUIT noch mitteilen, wie es die Fehler der Parameter bestimmen muss und die Parameter erklären, die es variieren soll, bevor die Minimalisierung starten kann. Mit der Methode `Command("SET ERR 1")` wird MINUIT mitgeteilt, dass der Parameter gerade eine Standardabweichung vom optimalen Wert entfert ist, wenn der Wert der Funktion `fcn` um Eins größer ist als der Minimalwert. Danach verwenden Sie die Methode `mmparm(i,Name,start,step,ll,ul,errorflag)` um die Parameter einzeln zu erklären. Dabei ist `i` der Index des Parameters (von 0 bis $n-1$), `Name` eine Bezeichnung des Parameters für die Ausgabe (z.B. `"Par1"`), `start` ein Wert für den Parameter, bei dem MINUIT starten soll und `step` die Schrittweite, mit der MINUIT am Anfang den Parameter variieren soll; später passt das Programm die Schrittweite selbständig an. Die nächsten beiden Parameter `ll` und `ul` sind untere und obere Schranken, die MINUIT beim Variieren nicht überschreiten soll; in der Regel sind sie nicht nötig und wir setzen Sie daher auf den Wert 0.0. `errorflag` schließlich ist ein Rückgabevariable für einen Fehlercode, die wir zwar deklarieren müssen aber nicht weiter beachten werden.

Nun, endlich, kann man mit einem weiteren Kommando die Minimalisierung starten. Das erfolgt wieder durch Aufrufen der Methode `Command`, aber diesmal mit dem Argument `MIGRAD` für die MINUIT Gradientenmethodem also `Command("MIGRAD")`. Das reicht bereits aus, um die Ergebnisse auf dem Schirm zu sehen.

Wenn man die Ergebnisse aber wieder in ROOT benutzen will, kann man sie mit der Methode `mnpout` erhalten, die 7 Argumente besitzt: `mnpout(i,name,value,error,ll,ul,internal)`. `i` ist wieder der Index der Variable, `name` der bei der Deklaration angegebene Name (als TString), `value` und `error` geben den Ergebniswert mit seinem Fehler aus, `ll` und `ul` die untere und obere Grenze für den Parameter und `internal` ist ein Integer, der eine interne Nummer der Variable zurückgibt, die uns nicht weiter interessieren muss.

Die Kovarianzmatrix erhält man mit einer anderen Methode, nämlich mit `mnemat`. Dazu erklärt man die Matrix als `Double_t Cov[4][4]`. Weil in C++ die Arrays mit einem Zeiger übergeben werden, aber `Cov` durch den doppelten Index kein Zeiger mehr ist, muss man die Routine mit einem „Cast“ aufrufen: `mnemat((Double_t *)Cov,4)`, wobei das zweite Argument der Routine die Größe der quadratischen Matrix mitteilt.

Leider ist die Dokumentation von TMinuit in ROOT nicht sehr ausführlich; deshalb habe ich Ihnen hier genau beschrieben, wie man vorgehen muss. Als weitere Information gibt es ein Beispielprogramm mit Namen `Ifit.C`, das Sie im Unterordner tutorial/fit des ROOT-Verzeichnisses finden. Außerdem wird immer auf die alte MINUIT-Beschreibung von 1994 verwiesen `https://root.cern.ch/sites/d35c7d8c.web.cern.ch/files/minuit.pdf`, die sehr vollständig ist, aber alle Beispiele in FORTRAN-Code enthält.

Daneben gibt es in ROOT auch andere Fit-Routinen, auch eine „modernere“ Version namens `TMinuit2`, welche aber nur für Histogramme funktioniert und keinen direkten Zugang auf die Minimalisierungsroutine `fcn` erlaubt. Daher ist sie zwar sehr einfach mit `histo->Fit()` zu aktivieren, aber nicht so universell zu verwenden wie `TMinuit`, denn Sie können nicht programmieren, *was* minimalisiert werden soll.

Als letzte Aufgabe zeichnen Sie bitte die von Ihnen bestimmte Funktion wieder mit der Option `SAME` in das Diagramm der Messpunkte ein.