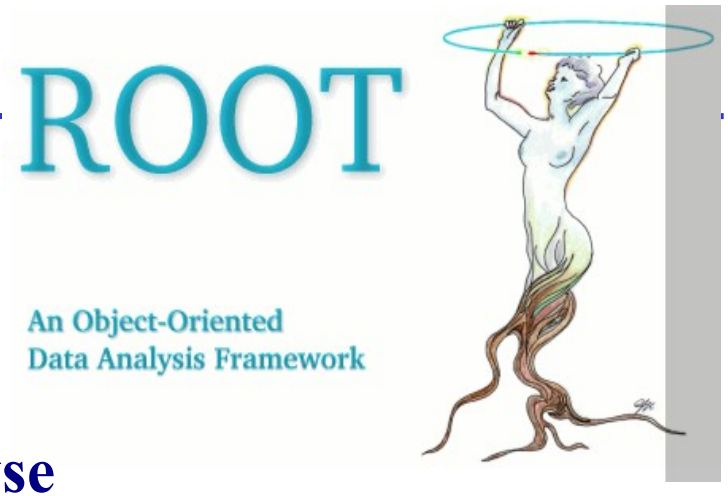
Datenanaufnahme, Speicherung, Visualisierung und Analyse:

das ***ROOT***-Framework

ROOT:

Vorläufer: Hbook, PAW aus der Teilchenhpysik
langjährige Entwicklungsarbeit, großer Nutzerkreis
siehe http://root.cern.ch



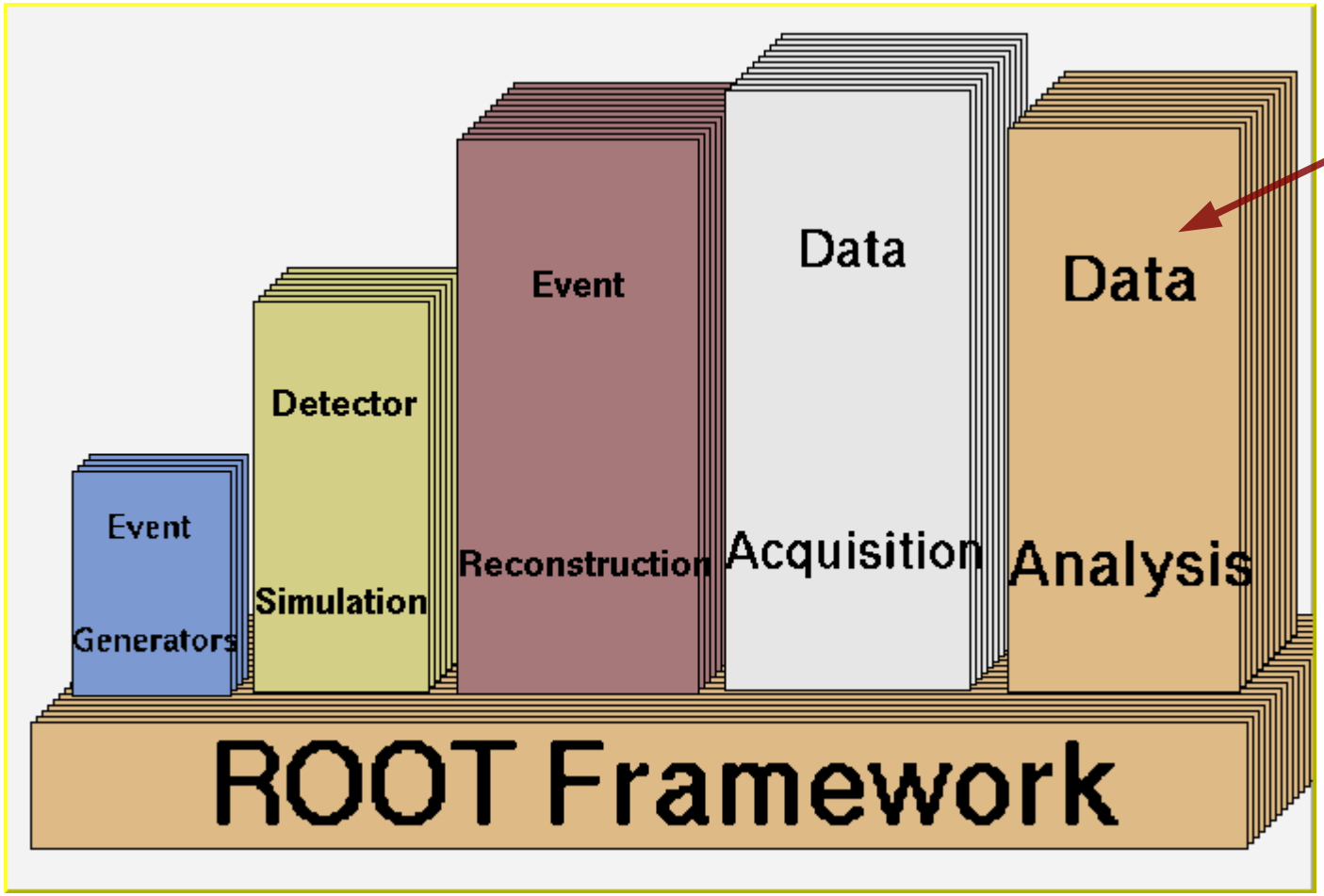
Root ist ein „Framework“ zur Datenanalyse

- Kommandozeileninterpreter mit c++ Syntax, Python-Interface
- Compilierung von Skripten, dynamische Einbindung d. erzeugten Bibliotheken
- Grafische Benutzeroberfläche, Ein- und Ausgabe-Funktionen für Daten & Grafiken
- Anpassung von Funktionen an Daten (incl. versch. Verfahren zur numerischen Optimierung)
- Histogramme (Häfugikeitsverteilungen) und „nTuples“ („Vektoren“ aus Messwerten)
- Unterstützung für „Monte-Carlo“-Simulationen (basierend auf Zufallszahlen)
- Ständige Erweiterung um neue, mächtige Pakete (Neuronale Netze und andere multivariate Analyseverfahren, Berechnung von Konfidenzintervallen u. v. a.)

- **ROOT ist Open Source !!!**

Entwicklungsumgebung
für DatenanalyseProjekte !

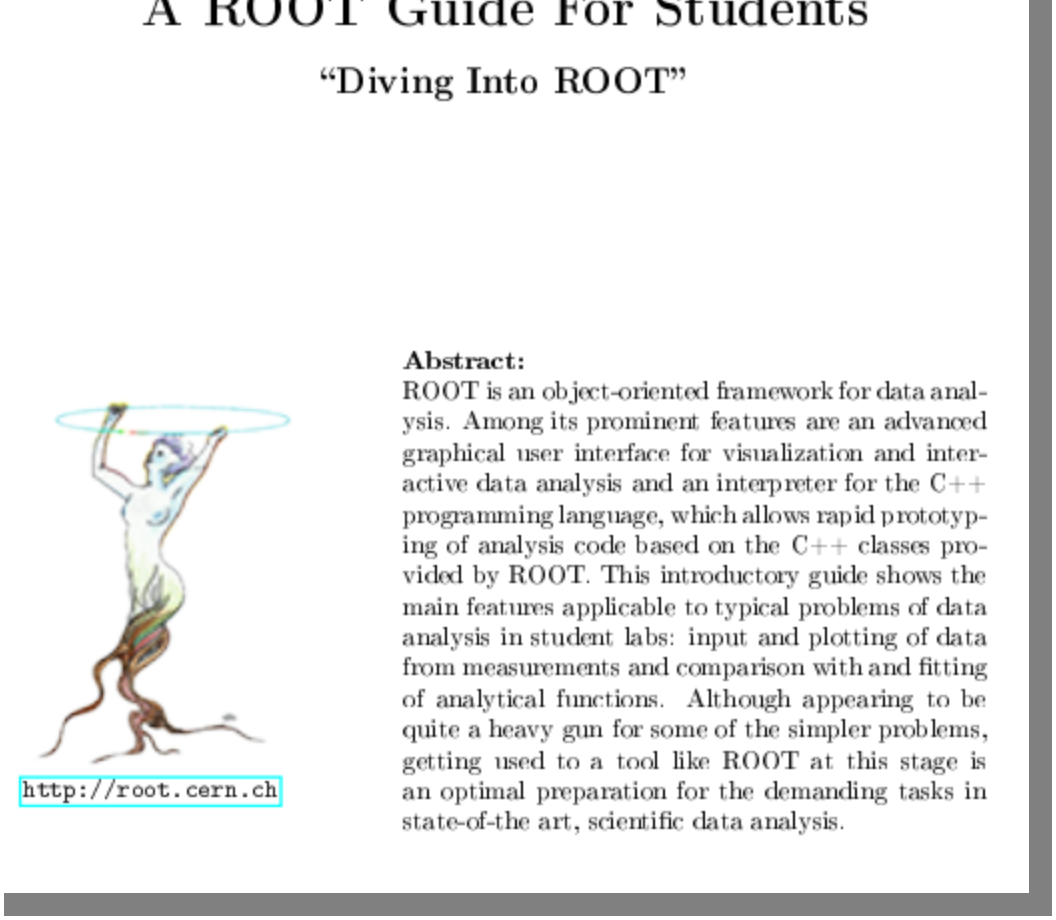
ROOT Framework



Werde hier hauptsächlich diesen Aspekt behandeln.

Diving into ROOT: neues Skript zur Einführung

http://www-ekp.physik.uni-karlsruhe.de/~quast/Skripte/



A ROOT Guide For Students

"Diving Into ROOT"

http://root.cern.ch

Abstract:
ROOT is an object-oriented framework for data analysis. Among its prominent features are an advanced graphical user interface for visualization and interactive data analysis and an interpreter for the C++ programming language, which allows rapid prototyping of analysis code based on the C++ classes provided by ROOT. This introductory guide shows the main features applicable to typical problems of data analysis in student labs: input and plotting of data from measurements and comparison with and fitting of analytical functions. Although appearing to be quite a heavy gun for some of the simpler problems, getting used to a tool like ROOT at this stage is an optimal preparation for the demanding tasks in state-of-the art, scientific data analysis.

als html und pdf;
die Datei ***divingROOT.zip***
enthält auch die Code-Beispiele

Inhalt

- Root Basics
- Root Macros
- Graphs
- File I/O und N-tuples
- Histograms
- Functions and Parameter Estimation
- Appendix

Die ersten Schritte

- Root von http://root.cern.ch herunterladen
  (für verschiedene Linux-Distributionen oder Windows-Version,
  bzw. aus Quellcode selbst übersetzen)
- Evtl. Root-Dokumentation (im html-Format als komprimiertes tar-Archiv) herunter laden
- Root installieren (d.h. Archiv auspacken unter Linux!)
  es wird ein C++-Compiler benötigt
  gcc unter Linux, Visual C++ unter Windows, auch Mac OS X unterstützt
- Kleines Script in jeder login-shell ausführen (evtl. schon in login-script, z.B. .bashrc, erledigt):

```
#!/bin/bash
# setuproot.sh
# set execution PATH etc. for root
export ROOTSYS=/cern/root
export PATH=${PATH}:$ROOTSYS/bin
# include ROOT libraries in linker path
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$ROOTSYS/lib
```

```
>source setuproot.sh
```

- Fertig !

ROOT:

Root-Sitzung:

```
unix-prompt> root
  ****************************************
  *                                      *
  *        W E L C O M E  to  R O O T    *
  *                                      *
  *   Version   3.02/07   18 January 2002   *
  *                                      *
  *  You are welcome to visit our Web site  *
  *          http://root.cern.ch         *
  *                                      *
  ****************************************

Compiled for linux with thread support.

CINT/ROOT C/C++ Interpreter version 5.15.25, Jan 6 2002
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] TF1 meinF("sin/x)/x","sin(x)/x",0,10)
root [1] meinF.Draw()
<TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [2] .q
```

Nützliche ROOT-Kommandos

1d-Histogramme:

Histogramm erzeugen

```
TH1F *h1=new TH1F
  ("Title","name",nbin,x_min,x_max);
```

Histogramm füllen

```
h1->Fill(x,weight=1);
```

Eintrag im i-ten Bin setzen

```
h1->SetBinContent(i,x);
```

Eintrag im i-ten Bin auslesen

```
h1->GetBinContent(i);
```

Fehler im i-ten Bin setzten

```
h1->SetBinError(i,err);
```

Histogramm zeichnen

```
h1->Draw();
```

Histrogramm leeren

```
h1->Reset();
```

Histogramminhalt in Feld schreiben

```
Float_t* content=h1->GetArray();
```

Maximum auf der y-Achse setzen

```
h1->SetMaximum(ymax);
```

Mnimum auf der y-Achse setzen

```
h1->SetMimimum(ymin);
```

ROOT:

2d-Histogramme:

Histogramm erzeugen

füllen
zeichnen
Optionen

Profile erzeugen

Funktionen:

Funktion erzeugen
(nur im Macro so einfach)
parametrisierte Fkt. erzeugen,
par [0],[1],[2] ...
Parameter setzen
Histogramm mit Ftk. Fitten
auf Def. Ber. von Fkt. fitten
auf Unterbereich fitten
verdefinierte Fkt.

```
TH2F *h2=new TH1("Title","name",
 nbin_x,x_min,x_max,nbin_y,y_min,y_max);

h2->Fill(x,y,weight=1);

h2->Draw();

    Draw("SAME") ,   DRAW("BOX")

Tprofile* tp=new Tprofile("title",

 TF1* f1=new("name",
       "sin(x)/x",xmin,xmax)

 TF1* f1=new TF1("name",

       "[0]*sin([1]*x)/x+[2]",xmin,xmax)

 Double_t par[3]; f2->SetParameter(par);

 h1->fit("f1");

 h1->fit("f1","R");

 h1->fit("f1","","",x1,x2);

 gaus,expo,polN(N=1,2,...),landau
```

ROOT:

Zufallsgeneratoren:

Voreingestellten Generator ersetzen
schneller Generator in ROOT
vordef. Verteilungen

```
delete gRandom; gRandom=new MyRandom();

Klasse Trandom3(seed=0);

gaus(E=0,σ=1),uniform(x1,x2),poisson(E)
binomial(n,p),landau(E=0,σ=1)
```

Histogramm mit n Zufallszahlen füllen
mit. Zuf.Zahlen gem. TH1F* f1 füllen

```
h1->FillRandom("gaus",n);

h1->FillRandom("f1",n);
```

Globale Einstellungen

Fitparameter in StatistikBox anzeigen
Parameteranzeige in Statistk-Box
Pause einbauen

```
gStyle->setOptFit(11...1);

gStyle->SetOptStat(11...1);

gSystem->Sleep(t);
```

ROOT:

Ntuple:

tuple erzeugen

Ntuple füllen
Var. zeichnen
in ein Histogramm projezieren
Var. mit Schnitten zeichenen
2 Var. Zeichnen

Dateioperationen

Datei erzeugen
Histogramm schreiben
Datei schließen
Histrogram lesen

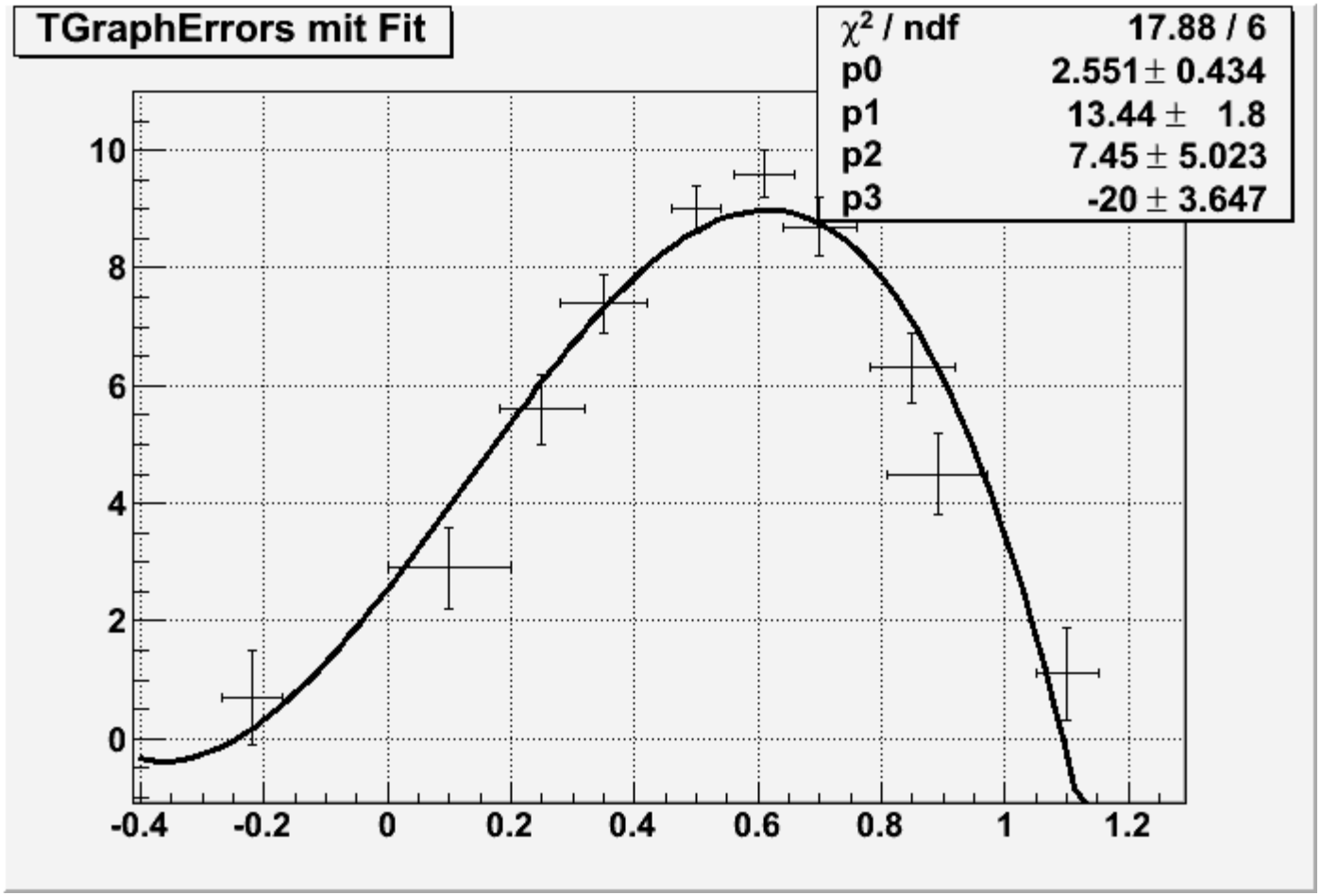
Mehrere Histogramme/Seite

„Canvas“ erzeugen
untereilen
Histogramme zeichnen

```
Tntuple* nt =new Tntuple("name",
  "title","x1:x2:...xn")

nt->Fill(x1,x2,...,xn);

nt->Draw("x1");

nt->Project("h1","x1");

nt->Draw("x1","x2<3");
Tfile* file=new Tfile("file.root");

file->cd(); h1->Write();

file->Close();

TFIle *f("file.root");

Tcanvas* c1 =new Tcanvas("name","titel",
             xpos,ypos,xsiz,ysiz);

c1->Divide(2,3); c1->cd();

h1->Draw();...;hn->Draw(); c1->Update();
```

Ein praktisches Beispiel:

Darstellung von Datenpunkten und Funktionsanpassung



Beispiel TgraphFit.C

- Darstellung von Datenpunkten mit Fehlern in x- und y-Richtung
- Anpassung einer Funktion (Polynom dritten Grades)

Ein praktisches Beispiel

Darstellung von Datenpunkten und Funktionsanpassung

```
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};

 // 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();
}
```

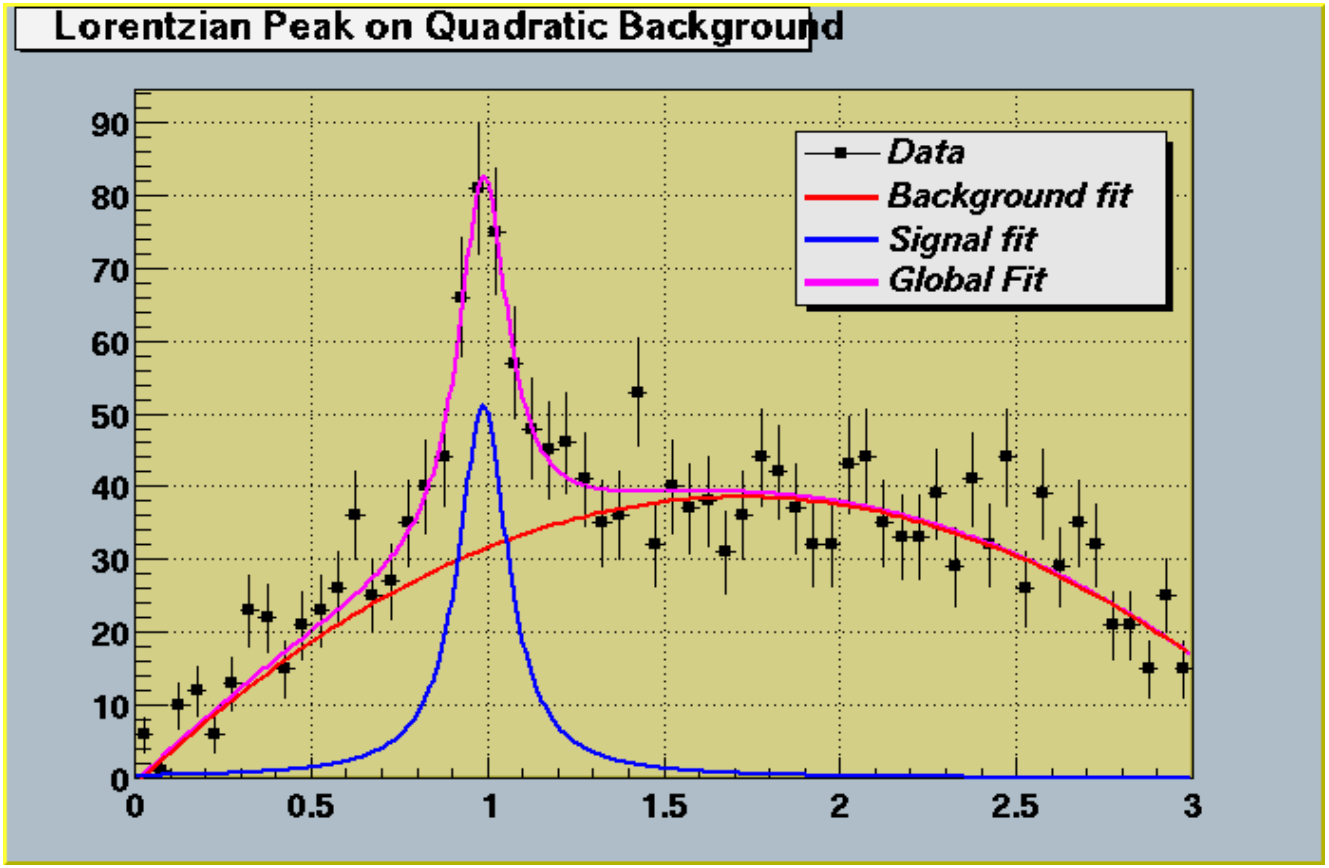
mehr code ist in Root
nicht nötig;

Was
`gr->Fit(''pol3'')`
macht, ist Thema (u.a.) dieser Vorlesung !

ROOT:

Es gibt noch eine ganze Menge weitere Beispiele
- und täglich werden es mehr.

Es lohnt sich, (einige) Beispiele in $ROOTSYS/tutorials zu studieren.



/root/tutorials/FittingDemo.c

Eine Root-Anwendung: RooFiLab

RooFiLab ist eine Root-basierte Anwendung
zur Anpassung von Modellen (=Funktionen) an Messdaten:

- Dateneingabe als ASCII-Datei
- Definition der Fit-Funktion als Formel
- Setzen von Startwerten und graphische Darstellung
- Durchführen der Anpassung
- Nachbearbeiten der graphischen Ausgabe

Graphische Oberfläche zur geführten Anpassung sowie
Automatisierung durch einfache Anweisungen in der Eingabedatei

Bezugsquelle:
http://www-ekp.physik.uni-karlsruhe.de/~quast/RooFiLab/RooFiLab.tar.gz
Quell-Code, Dokumentation, Makefile, Beispiele

Homepage des Autors:

auch in virtueller Maschine zum Kurs enthalten !

`http://tmnet.cli.data-cmr.net/index.php/studium/programme/roofilab`