Laboratorio II – 1° modulo
Lezione 9

Analisi dei dati e rappresentazione grafica con ROOT

Indice

- Rappresentazione grafica dei dati con ROOT
  - Istanziare un istogramma nella memoria dinamica
  - Visualizzazione grafica degli istogrammi, funzioni, ecc…
- Analisi dei dati
  - Estrazione dei parametri di un modello mediante l'operazione di regressione (i.e. fit)
  - La classe `TF1` di ROOT
  - Cosa fare se il fit non converge
- Esercizi
- Approfondimenti

Istogrammi e memoria dinamica

Qualunque tipo di variabile, classi incluse, può essere istanziata nella memoria dinamica del calcolatore mediante l'istruzione `new`. Vediamo il caso di alcune classi di ROOT:

```
TH1D* myHisto = new TH1D("myHisto","Titolo istogramma",
100,-10,10); // Creazione di un istogramma

...

myHisto->Fill(0.4); // Fill di un istogramma

myHisto->Draw(); // Draw di un istogramma

myHisto->Fit("gaus"); // Fit di un istogramma
```

ecc…

Ovviamente quanto detto vale per qualunque tipo di classe, sia creata da voi che di ROOT (*cfr. slide 26 Lezione 5*)

Visualizzazione degli istogrammi

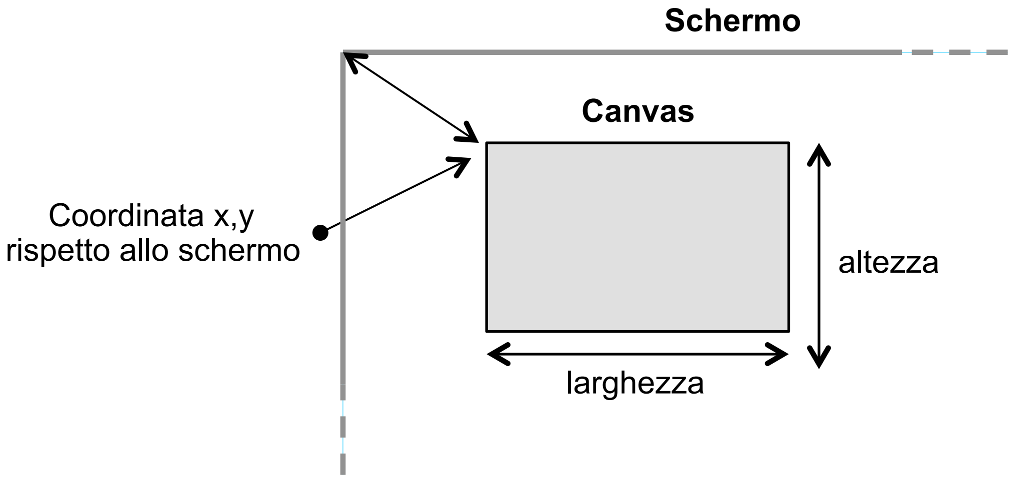
In ROOT è anche possibile visualizzare a schermo istogrammi, funzioni, grafici, etc… Le classi ed i metodi di ROOT da utilizzare sono i seguenti:

TCanvas: genera la finestra in cui verrà disegnato l'istogramma (funzione, grafico, etc...)

```
#include <TCanvas.h>

...

TCanvas* myCanv = new TCanvas("myCanv","Titolo canvas",
0,0,700,500); // TCanvas("nome", "titolo", x, y, larghezza,
altezza)
```



Visualizzazione degli istogrammi

Esempio:

```
TCanvas* myCanv = new TCanvas("myCanv","Titolo canvas",
0,0,700,500);

TH1D* myHisto = new TH1D("myHisto","Titolo istogramma",
100,-10,10);

...

myCanv->cd(); // Ci si "sposta" nel canvas

myHisto->Draw();

myCanv->Modified();
                    // Esegue il "refresh" del canvas
myCanv->Update();

...
```

Visualizzazione degli istogrammi

`TApplication`: gestisce l'accesso alla finestra del canvas mediante il mouse

```
#include <TApplication.h>

...

TApplication* myApp = new TApplication("myApp", NULL, NULL); // Si possono passare
dei parametri ma noi non li utilizziamo
```

Esempio:

```
...

int main()

{

    TApplication* myApp = new TApplication("myApp", NULL, NULL);

    TCanvas* myCanv = new TCanvas("myCanv","Titolo canvas",0,0,700,500);

    TH1D* myHisto = new TH1D("myHisto","Titolo istogramma",100,-10,10);

    // Inserite qui il vostro codice

    myApp->Run();

    return 0;

}
```

Con il metodo `Run()` della classe `TApplication` viene lanciata l'esecuzione del codice ROOT che gestisce le finestre. **N.B.**: dovete istanziare una ed una sola variabile `TApplication`. Anche se istanziate più canvas (cioè più finestre) `myApp` è in grado di gestirli tutti

Analisi dei dati

Cosa significa eseguire una regressione (i.e. fit)?

- Si esegue una regressione (o fit) quando si vuole trovare il **miglior adattamento** di un modello ad un set di dati affetto da delle incertezze

- Generalmente il **modello**, $f(x_1 \ldots x_n; \theta_1 \ldots \theta_m)$, è una **funzione parametrica** (θ_j), con la quale si intende descrivere l'andamento dei **dati sperimentali**

- I dati sperimentali possono legare una osservabile ad un'altra, e.g. **grafico** posizione (y_i) – tempo (x_i) di un grave, oppure sono la registrazione dell'occorrenza (y_i) di una certa osservabile, e.g. **istogramma** delle altezze (x_i) di un campione di persone

- L'operazione di fit consiste nel **massimizzare** (o minimizzare) un certo **funzionale** (F). Questa operazione di **ottimizzazione** viene eseguita in **funzione dei parametri** del modello che vengono cosi ottimizzati per avere il miglio adattamento ai dati

$$F(\overrightarrow{x}; \overrightarrow{\theta}) \Rightarrow \nabla_{\overrightarrow{\theta}} F(\overrightarrow{x}; \overrightarrow{\theta}) = 0 \quad \text{i.e.} \quad \frac{\partial F(\overrightarrow{x}; \overrightarrow{\theta})}{\partial \theta_i} = 0$$

$\vec{x}$ vettore dei dati (i.e. misure), $\vec{\theta}$ vettore dei parametri del modello

Il metodo dei minimi quadrati

Tipicamente i funzionali, cioè i metodi di fit, sono di due tipi:

- **Minimi quadrati** (problema di minimizzazione di una certa "distanza" del modello dai dati)
- **Massima verosimiglianza** (problema di massimizzazione della probabilità, descritta dal modello, di ottenere i dati)

Il funzionale dei minimi quadrati e`:

Che espressione assume σ_i nel caso di un istogramma?

$$\mathrm{Min}^2(\overrightarrow{\theta}) = \sum_{i=1}^{n} \left(\frac{y_i - f(x_i; \overrightarrow{\theta})}{\sigma_i} \right)^2$$

n = numero di misure / conteggi
y_i = misura-i / conteggio-i
$f()$ = modello con m parametri θ_j
σ_i = errore sulla misura / conteggio

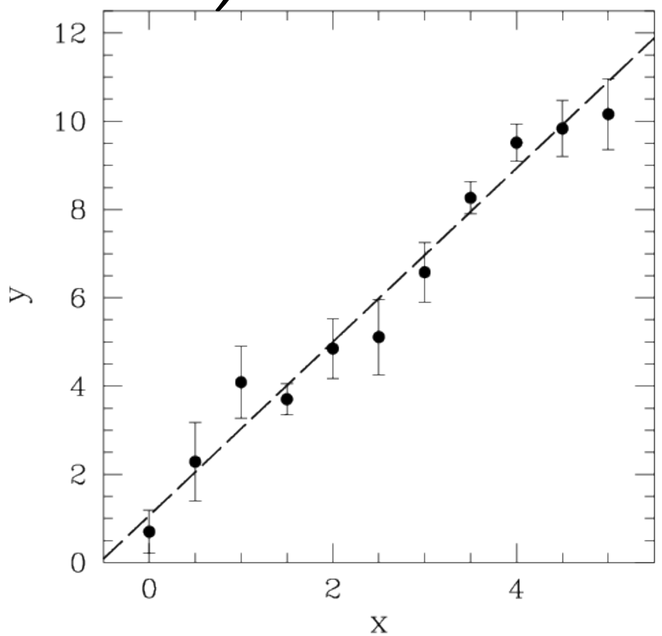
Perché proprio questa relazione matematica? → **Teorema di Gauss-Markov**: per modelli lineari nei parametri il metodo dei minimi quadrati fornisce degli stimatori lineari, *unbiased*, a varianza minima, cioè per i quali l'incertezza nella stima dei θ_j è più piccola possibile (nella classe degli stimatori lineari e *unbiased*)

Il metodo dei minimi quadrati

- Prendiamo per esempio un modello lineare: `y = ax + b`, e.g. l'andamento della velocità di un grave in caduta libera, nel vuoto, in un campo gravitazionale in funzione del tempo
- Il funzionale dei minimi quadrati avrà espressone:

$$\mathrm{Min}^2(a,b) = \sum_{i=1}^{n} \left(\frac{y_i - (ax_i + b)}{\sigma_i} \right)^2$$

Dove le x_i rappresentano i diversi tempi a cui sono state effettuate le misure di velocità y_i. Le σ_i sono le incertezze associate alle diverse misure di velocità y_i

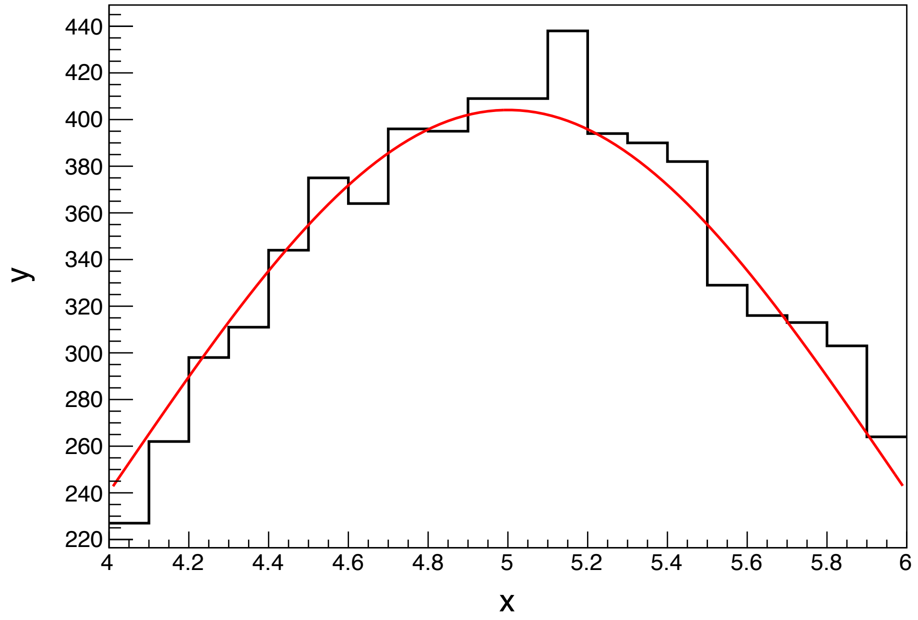


Il metodo dei minimi quadrati

- Prendiamo per esempio un istogramma di una misura affetta da da una incertezza avente distribuzione Normale
- Il funzionale dei minimi quadrati avrà espressone:

$$\mathrm{Min}^2(A,\mu,\sigma)=\sum_{i=1}^{n}\left(\frac{y_i - A\cdot e^{-\left(\frac{x_i-\mu}{\sigma}\right)^2}}{\sqrt{y_i}}\right)^2$$

Dove le x_i rappresentano i centri dei diversi bin. Le y_i sono i conteggi per bin. Le incertezze sui conteggi, σ_i, sono semplicemente date dalla radice quadrata dei conteggi medesimi, i.e. $\sqrt{y_i}$



Il metodo dei minimi quadrati

L'operazione di regressione (i.e. fit) consta in realtà di tre passaggi distinti:

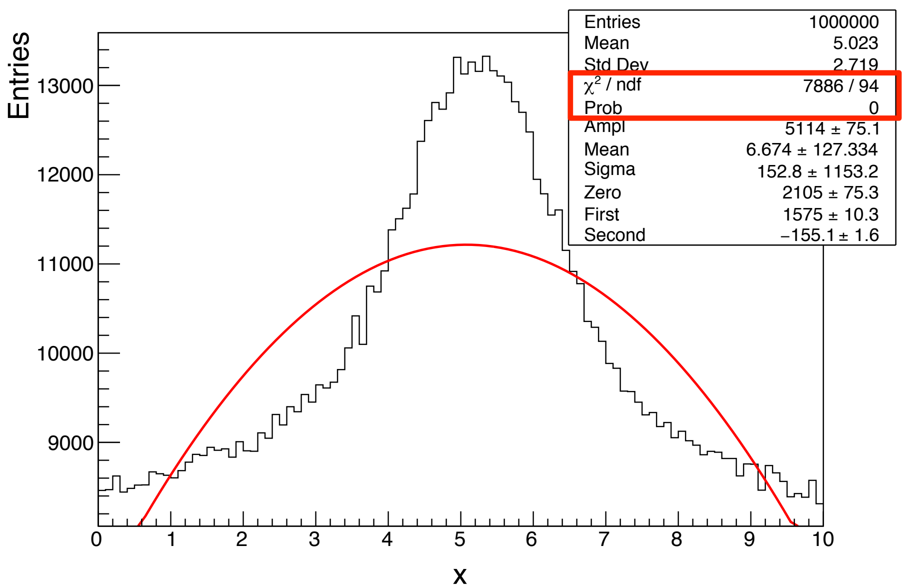
- ✓ Problema di ottimizzazione del modello rispetto ai dati (e.g. minimizzazione minimi quadrati) → miglior stima dei parametri
- ✓ Quantificare la qualità della regressione → effettuare un test statistico che sia in grado di dire quale sia la probabilità che il modello descriva i dati
- ❑ Quantificare l'incertezza sulla stima dei parametri

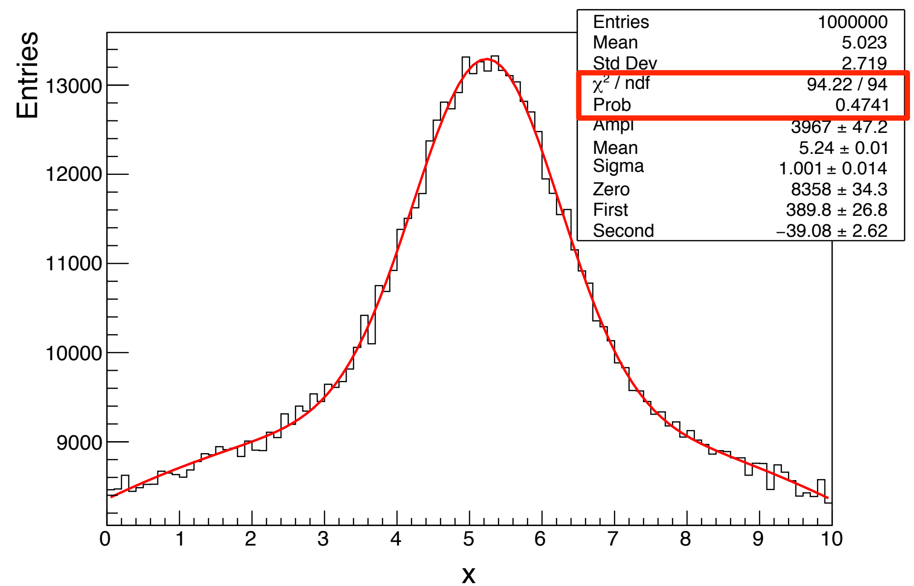
Nel caso dei **minimi quadrati**, e sotto l'ipotesi di incertezze aventi distribuzione di probabilità Normale, il funzionale ha **distribuzione di probabilità χ^2_{n-m}**, `n` numero di misure, `m` numero dei parametri del modello (per una dimostrazione si veda "*Statistical Methods in Data Analysis*," W.J. Metzger, capitolo 3.11)

$$\mathrm{Min}^2(\vec{\theta}) = \sum_{i=1}^{n} \left(\frac{y_i - f(x_i; \vec{\theta})}{\sigma_i} \right)^2 \sim \chi^2_{n-m}$$

Verificare la bontà di un fit

Esempio di "cattiva" (sinistra) e "buona" (destra) descrizione dell'andamento dei dati da parte del modello





- Con metodo dei minimi quadrati è quindi possibile effettuare il **test del Chi-2** per capire se il modello fornisce una "buona" (o "cattiva") descrizione dei dati
- Il test ci fornisce uno strumento per giudicare se le differenze tra i dati sperimentali ed il modello sono compatibili con le fluttuazioni statistiche

Torniamo a ROOT e vediamo un esempio di regressione

Funzioni predefinite in ROOT

In ROOT ci sono alcune **funzioni pre-definite** pronte per essere usate nel fit:

- **gaus** → Normale con 3 parametri: `f(x) = p0 * exp(-0.5*((x-p1)/p2)`^{2}`))`
- **expo** → Esponenziale con 2 parametri: `f(x) = exp(p0+p1*x)`
- **polN** → Polinomio di grado N: `f(x) = p0 + p1*x + p2*x`^{2}` +...`

Il loro utilizzo è immediato, non è nemmeno necessario inizializzare i parametri perché lo fa ROOT in automatico, e.g. `myHisto->Fit("`**`gaus`**`")`

Funzioni definite dal programmatore

E` inoltre possibile definire una qualsivoglia funzione di fit utilizzando la classe **TF1** (funzioni di una variabile)

- Ci sono due modi per definire un oggetto della classe `TF1` da passare come argomento al metodo `Fit`:
  - Scrivendo esplicitamente la funzione all'atto della dichiarazione della `TF1`:

```
TF1* myFun = new TF1("myFun","[0]*x+[1]",0,10);
```

  - Passando al costruttore della `TF1` una funzione C++:

```
double retta (double* x, double* par)
{
   return par[0]*x[0]+par[1];
}
...
TF1* myFun = new TF1("myFun",retta,0,10,2);
```

Range della funzione

Numero di parametri

Inizializzazione dei parametri

- Inizializzazione dei parametri:
```
myFun->SetParameter(0,0.5);
myFun->SetParameter(1,3.3);
```

E` fondamentale fornire all'algoritmo di fit dei valori iniziali per "aiutarlo" nella ricerca dei valori ottimali dei parametri

- Impostazione del nome dei parametri:
```
myFun->SetParName(0,"a");
myFun->SetParName(1,"b");
```

- Eseguire il fit: `myHisto->Fit("myFun");`

- Accedere ai risultati del Fit:
```
double chi2 = myFun->GetChisquare();
...
double p0 = myFun->GetParameter(0);
double e0 = myFun->GetParError(0);
...
```

Le opzioni del metodo `Fit`

In ROOT il metodo di `Fit` è definito come segue:

```
void Fit(TF1* f1, Option_t* option = "", Option_t* goption = "", double xmin = 0, double xmax = 0)
```

`option`:

- `"R"` → il fit è eseguito solo nell'intervallo in cui è definita la `TF1`
- `"L"` → il fit è eseguito massimizzando la Likelihood (il default è il metodo dei minimi quadrati)
- etc... (**consultare la documentazione**)

`goption` (graphic option) → per impostare la grafica dell'istogramma (di default il metodo `Fit` richiama un `Draw` dell'istogramma e della `TF1` risultante)

`xmin, xmax` → impostazione dell'intervallo in cui restringere il fit

Vediamo un esempio

```
...
int main()
{
  int nBins      = 100;
  double xMin    = 0;
  double xMax    = 10;
  int N          = 10000;
  double mean    = 5;
  double sigma = 1;

  // Stampa i parametri del fit sull'istogramma,
  // bisogna includere #include <TStyle.h>
  gStyle->SetOptFit(1112);

  TApplication* myApp = new TApplication("myApp", NULL, NULL);
  TCanvas* myCanv = new TCanvas("myCanv","myCanv",0,0,700,500);
  TH1D* myHisto = new TH1D("myHisto","myHisto",nBins,xMin,xMax);

  TF1* myFun = new TF1("myFun",myGauss,0,10,3);
  myFun->SetParameter(0,1);
  myFun->SetParameter(1,mean);
  myFun->SetParameter(2,sigma);

  myFun->SetParName(0,"Ampl");
  myFun->SetParName(1,"Mean");
  myFun->SetParName(2,"Sigma");
```

```
double myGauss (double* x, double* par)
{
  return par[0] * exp(-0.5*((x[0]-par[1])*(x[0]-par[1])/
                           (par[2]*par[2])));
}
```

Come inizializzare il numero di bin: al fine di utilizzare l'approssimazione Normale della statistica binomiale del bin è necessario che il numero di conteggi per bin sia maggiore di **9**

Come inizializzare i parametri: prima di eseguire il fit leggete e plottate i dati, in base alla loro forma/distribuzione inizializzate di conseguenza i parametri del modello (e.g. se dovete fittare una Normale guardate dove sono centrati i dati e la larghezza della distribuzione, se dovete fittare un polinomio di grado 2 e la concavità dei dati è verso il basso, inizializzate il coefficiente di x^2 ad un valore negativo, etc…)

Vediamo un esempio

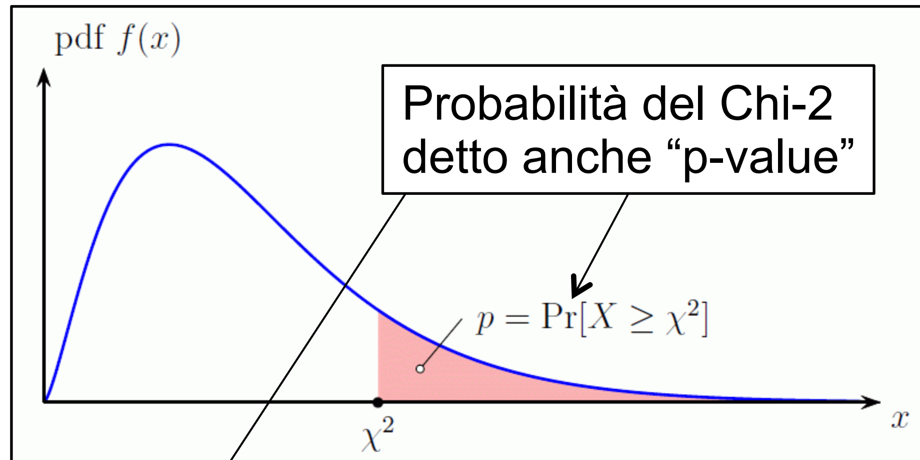
```
...
 for (unsigned int i = 0; i < N; i++)
    {
      myHisto->Fill(rand_TAC_gaus(mean, sigma));
    }

  myCanv->cd();
  myHisto->Draw();
  myHisto->Fit("myFun");

  myCanv->Modified();
  myCanv->Update();

  std::cout << "Reduced Chi-2: ";
  std::cout << myFun->GetChisquare()/myFun->GetNDF() << std::endl;
  std::cout << "p-value: " << myFun->GetProb() << std::endl;
  std::cout << "Number of entries: ";
  std::cout << nBins / (xMax - xMin) * myFun->Integral(xMin, xMax);
  std::cout << std::endl;

  myApp->Run();
  return 0;
}
```



Probabilità del Chi-2 detto anche “p-value”

Numero di gradi di libertà del Chi-2

Numero di eventi

Cosa fare se il fit non converge

Prima ancora di guardare al risultato del fit è necessario guardare l'output di **MIGRAD** a schermo. MIGRAD è il programma di ROOT che effettua la procedura di minimizzazione

```
FCN=63.2621 FROM MIGRAD    STATUS=CONVERGED      215 CALLS         216 TOTAL
                    EDM=3.00633e-08    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                   STEP         FIRST
 NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
  1  Ampl         4.04111e+02   5.04127e+00   1.58692e-02   5.16625e-05
  2  Mean         5.00040e+00   9.87541e-03   3.86068e-05  -1.03630e-03
  3  Sigma        9.81257e-01   7.30460e-03   2.29747e-05   3.85261e-02
```

Solo quando si è sicuri che MIGRAD ha prodotto un risultato credibile ha senso guardare il plot, Chi-2, p-value, etc …

Se lo STATUS non è CONVERGED, potrebbe essere necessario:

- cambiare i valori di inizio dei parametri
- evitare che i parametri assumano valori troppo grandi o troppo piccoli moltiplicandoli per dei valori opportuni (e.g. se un parametro (θ) ha un valore nell'intorno di 10^{-9}, scrivere il modello come $\theta \cdot 10^{-9}$, cosi θ varierà nell'intorno delle unità)
- "guidare" il fit fissando uno o più parametri e lasciando liberi gli altri (e.g. se il modello ha due parametri (θ_1, θ_2) si può lanciare il fit fissando θ_1 ad un valore ragionevole (metodo **FixParameter(…)** di TF1) e lasciando θ_2 libero; se il fit converge si esegue un secondo fit immediatamente dopo il primo, in maniera tale che θ_2 abbia come valore di partenza quello trovato dal primo fit, lasciando liberi sia θ_1 che θ_2 (metodo **ReleaseParameter(…)** di TF1)

Esercizi

Esercizio 1: Scaricate il file `data1.txt` e realizzate un programma che legga e fitti i dati in esso contenuti. In particolare il programma deve:

- Ricevere in input il nome del file di dati
- Implementare una funzione che legge il file di dati e li salva in un istogramma (`100` bin, `0 < x < 10`). Prototipo della funzione:
  - `bool ReadData(char* fileName, TH1D* myHisto); // return true se la lettura e` andata a buon fine, altrimenti false`
- Implementare mediate `TF1` la funzione di fit: somma di una Normale e di un polinomio del second'ordine
- Fittare i dati
- Scrivere a schermo il Chi-2, il numero di gradi di libertà, e il p-value computati da ROOT
- Scrivere a schermo il Chi-2 e il numero di gradi di libertà **computati da voi** (mediante una opportuna funzione) e verificare che coincidano con quelli computati da ROOT. Prototipo della funzione:
  - `void ComputeChi2(TH1D* myHisto, TF1* myFun, double& chi2, double& NDF);`

Esercizi

<u>Esercizio 2:</u> Scaricate il file `data2.txt` e, come nell'esercizio 1, realizzate un programma che legga e fitti i dati in esso contenuti. Svolgere i medesimi punti dell'esercizio 1 con l'unica differenza che in questo caso non vi viene fornito nessun suggerimento ne` sul range del fit, ne` sulla funzione con cui fittare i dati

Commento: in linea di principio potreste fittare qualunque cosa con un polinomio di grado sufficientemente elevato, ma fatevi guidare dal vostro senso fisico … e dal rasoio di Occam: "*Pluralitas non est ponenda sine necessitate*" (i.e. a parità di fattori la spiegazione più semplice è da preferire)

Approfondimenti

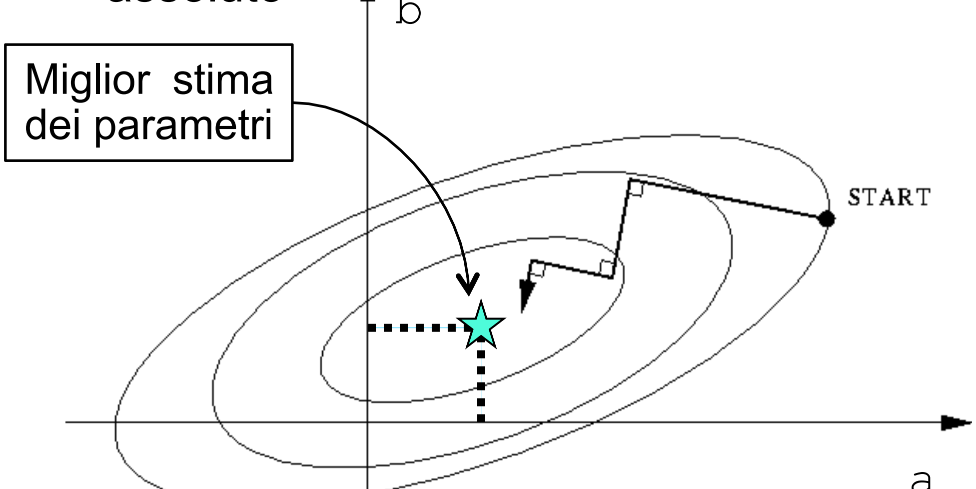
- Come funziona il processo di minimizzazione
- Referenze

Come funziona il processo di minimizzazione

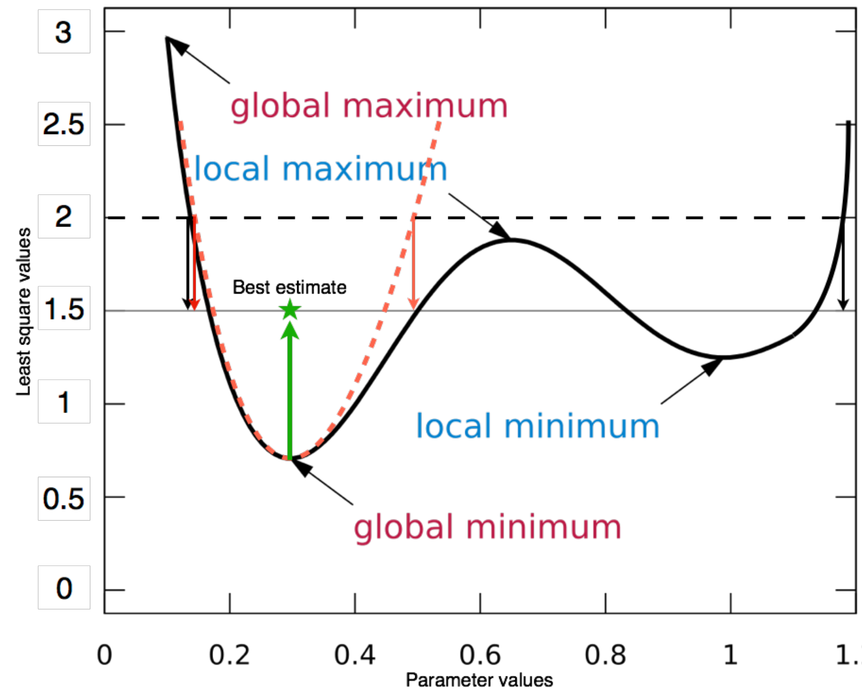
MIGRAD si basa sul computo della derivata prima in maniera tale da seguire la "discesa più ripida" (*steepest descent*) verso il minimo

- Come qualunque problema di ottimizzazione si possono incontrare le seguenti patologie: minimi secondari, minimi non ben definiti (i.e. derivata seconda nulla), complicato dominio di esistenza dei parametri
- Il problema è quindi non banale e anche se MIGRAD è un algoritmo robusto è necessario avere sempre un occhio critico

Come fare per evitare di incappare in minimi secondari? → Una tecnica molto comune è quella di lanciare il fit con diversi valori iniziali dei parametri del modello. Il fit per il quale il valore di Min² è più piccolo è quello che ha raggiunto il minimo assoluto



Curve di livello di Min² nello spazio dei parametri (e.g. a e b del modello lineare $y = ax+b$)



Referenze

Il pacchetto di ROOT per l'ottimizzazione si chiama `MINUIT`. Il programma di `MINUIT` che ricerca il minimo si chiama `MIGRAD`, mentre i programmi che calcolano gli errori si chiamano `HESSE` e `MINOS`

Per un approfondimento si veda:

- "*MINUIT - A system for function minimization and analysis of the parameter errors and correlations*" F. James and M. Roos, Computer Physics Communications **10** (1975) 343-367

- "*Interpretation of the shape of the likelihood function around its minimum*" F. James, Computer Physics Communications **20** (1980) 29-35

- "*MINUIT - Function minimization and error analysis - Reference Manual*" F. James

Come si può vedere da questo link non esistono molti software per risolvere problemi di ottimizzazione: https://en.wikipedia.org/wiki/List_of_optimization_software, uno di questi è il pacchetto `MINUIT` di ROOT