Tutto quello che avreste voluto sapere sui fit* (*ma non avete mai osato chiedere) (parte I)

Laboratorio di Metodi Computazionali e Statistici (2022/2023)

R. Cardinale, F. Parodi, S. Passaggio

November 24, 2022

Fit

- Scopo di questa lezione/esercitazione è raggiungere una maggiore comprensione dello strumento del fit sulla scorta di quanto visto a lezione sui minimi quadrati
- Gli esempi saranno basati su ROOT ma i concetti sottostanti sono generali.

Metodo dei minimi quadrati

- Il metodo dei minimi quadrati permette di determinare la funzione parametrica f che meglio approssima una seria di dati sperimentali $(x_i, y_i, \sigma(y_i))$ minimizzando la funzione $\chi^2(\boldsymbol{\beta})$

$$\chi^2(\boldsymbol{\beta}) = \sum_{i=1}^{N} \left(\frac{y_i - f(x_i|\boldsymbol{\beta})}{\sigma(y_i)} \right)^2$$

  rispetto al vettore di parametri $\boldsymbol{\beta}$ $(\beta_1, \cdots, \beta_M)$. Il vettore $\hat{\boldsymbol{\beta}}$ è quello che minimizza il χ^2
- La minimizzazione della funzione di χ^2 avviene:
  - algebricamente ricavando una formula analitica, nel caso in cui la funzione sia lineare nei parametri e l'errore sulle coordinate x trascurabile;
  - numericamente in tutti gli altri casi.

Minimizzazione numerica

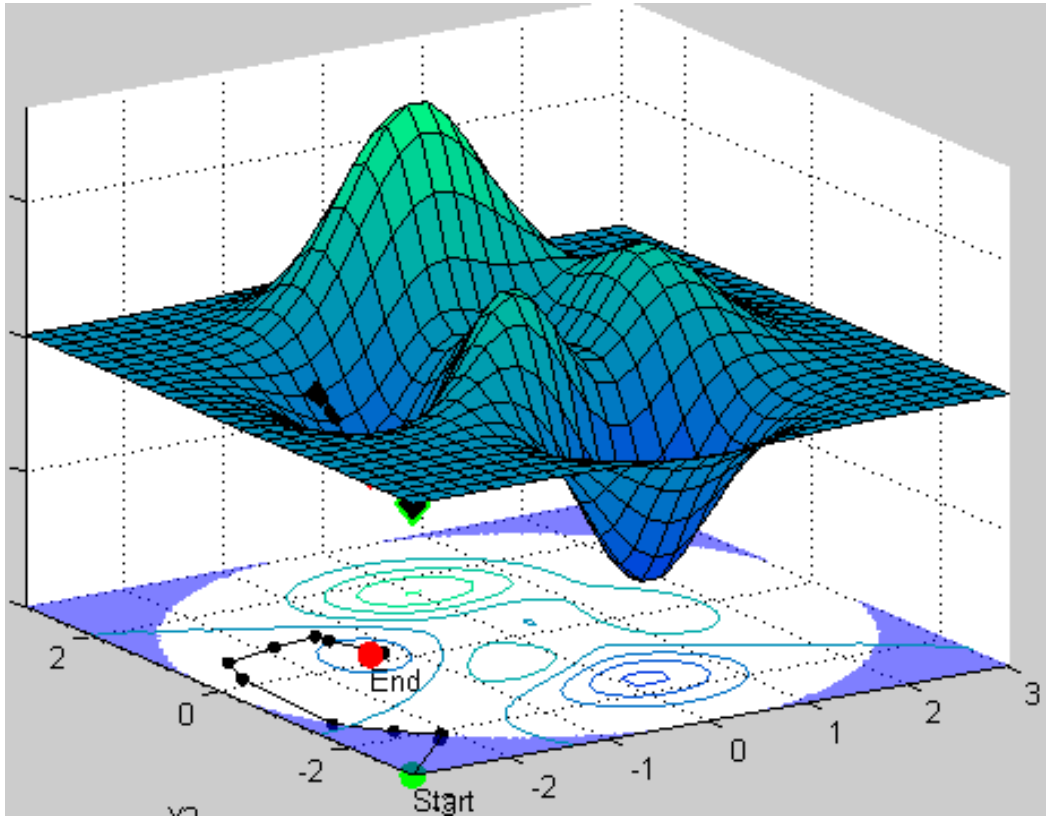
- Per la minimizzazione numerica del χ^2 useremo (ed in parte avete già usato) gli strumenti che ROOT fornisce.
  - In particolare le classi TGraph, TGraphErrors e tutte le classi di istogrammi sono dotate di un metodo

    ```
    void Fit(const char* fun);
    ```

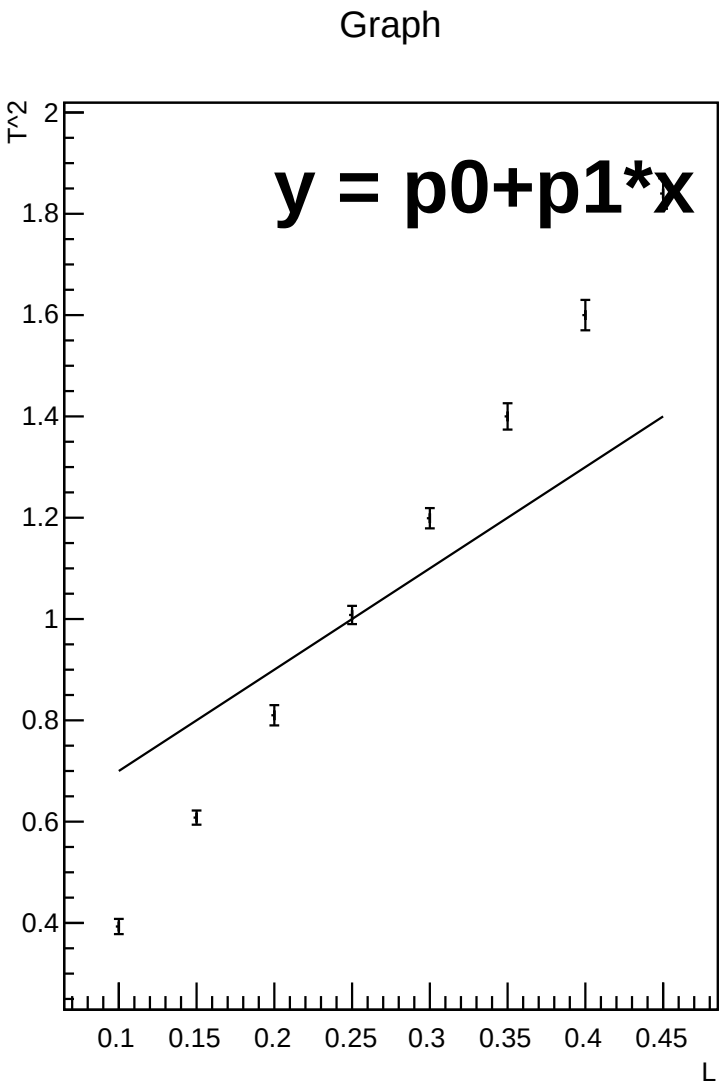
    dove fun è il nome della TF1 che si vuole utilizzare funzione di fit.
  - Elemento chiave di questa minimizzazione è il valore iniziale dei parametri
    - Occorre anche avere una stima ragionevole dei valori dei parametri che sono presenti nella funzione.
    - Infatti i programmi di minimizzazione hanno bisogno di un punto di partenza (punto nello spazio dei parametri).
    - Se questo punto di partenza è troppo distante dal minimo fisico la minimizzazione può selezionare un minimo relativo non fisico.

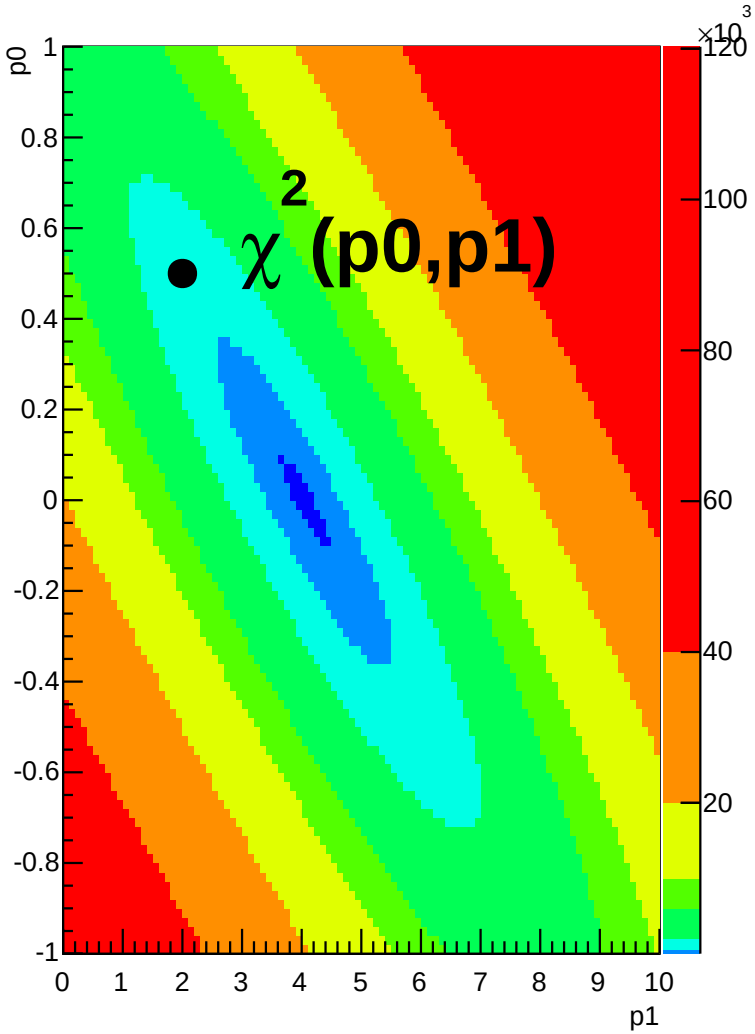
Inizializzazione dei parametri

- La visualizzazione della funzione di χ^2 (per esempio nello spazio di due parametri) rende chiaro come un punto di partenza errato possa risultare nell'individuazione di un minimo locale (senza significato fisico) e non globale come invece desiderato.

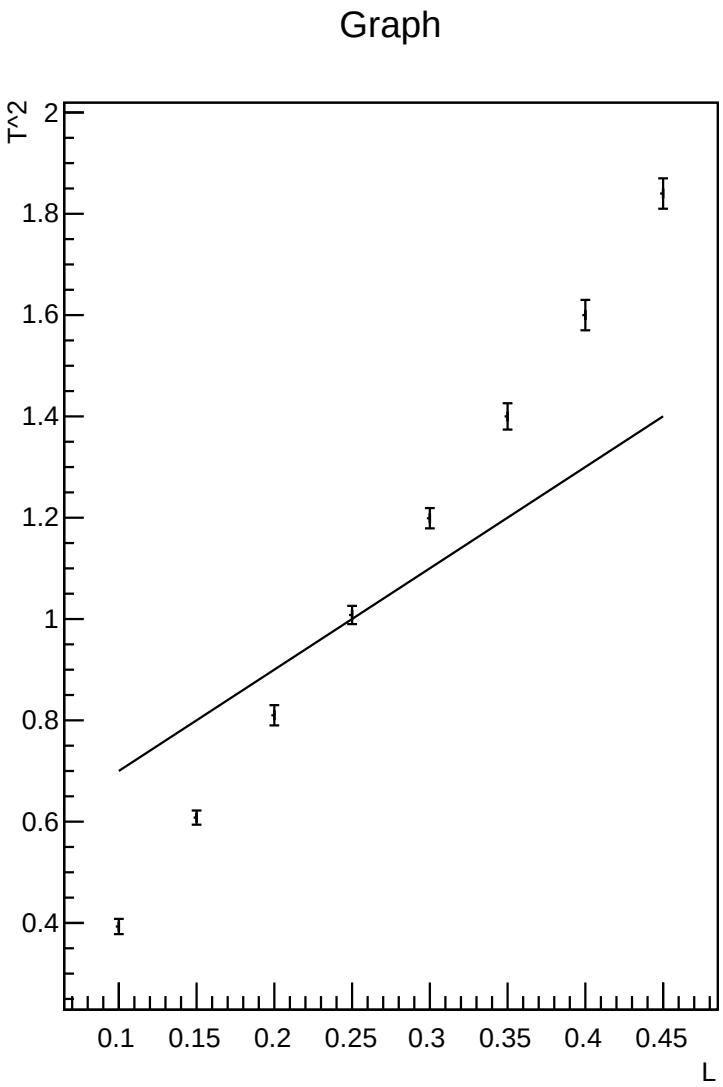


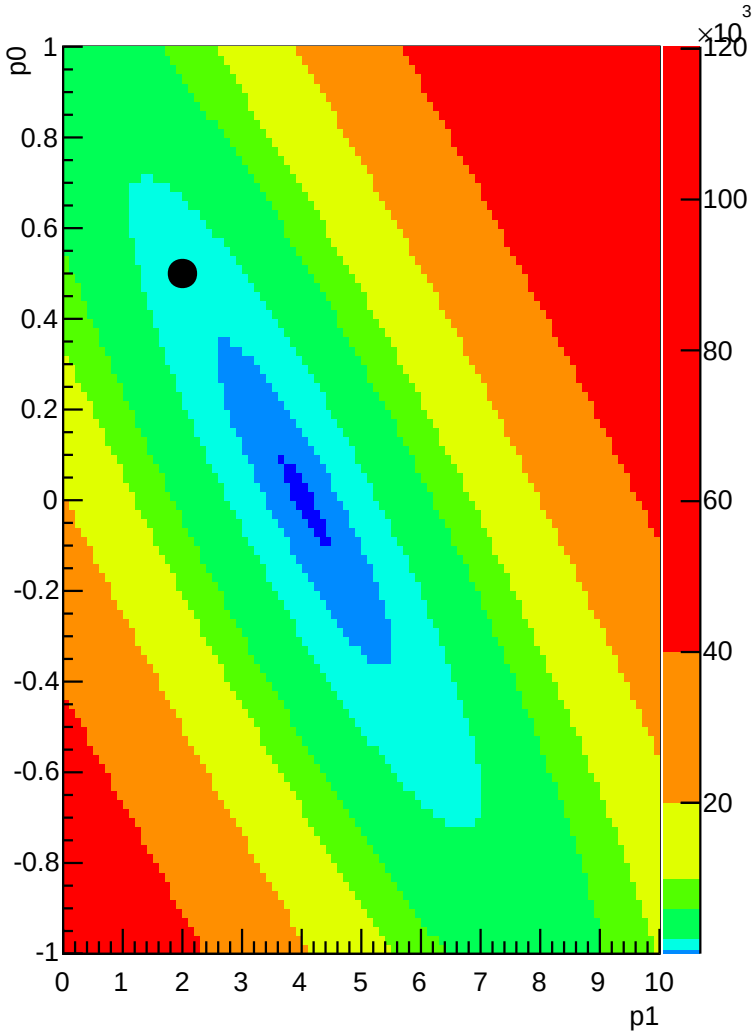
Fit di una retta



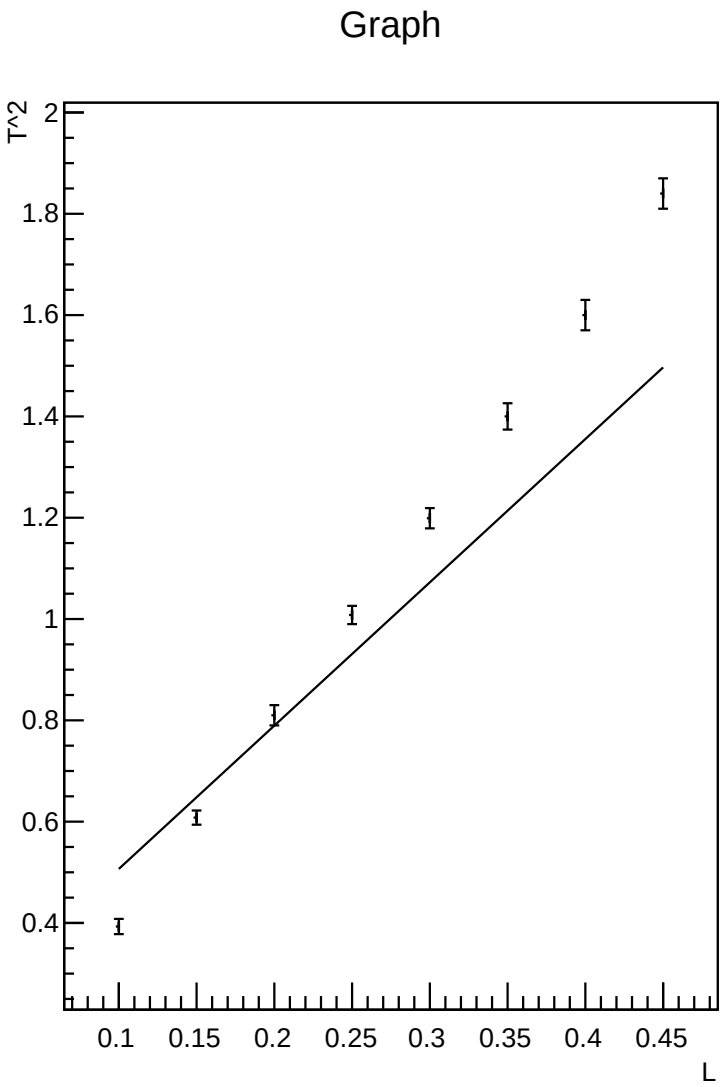


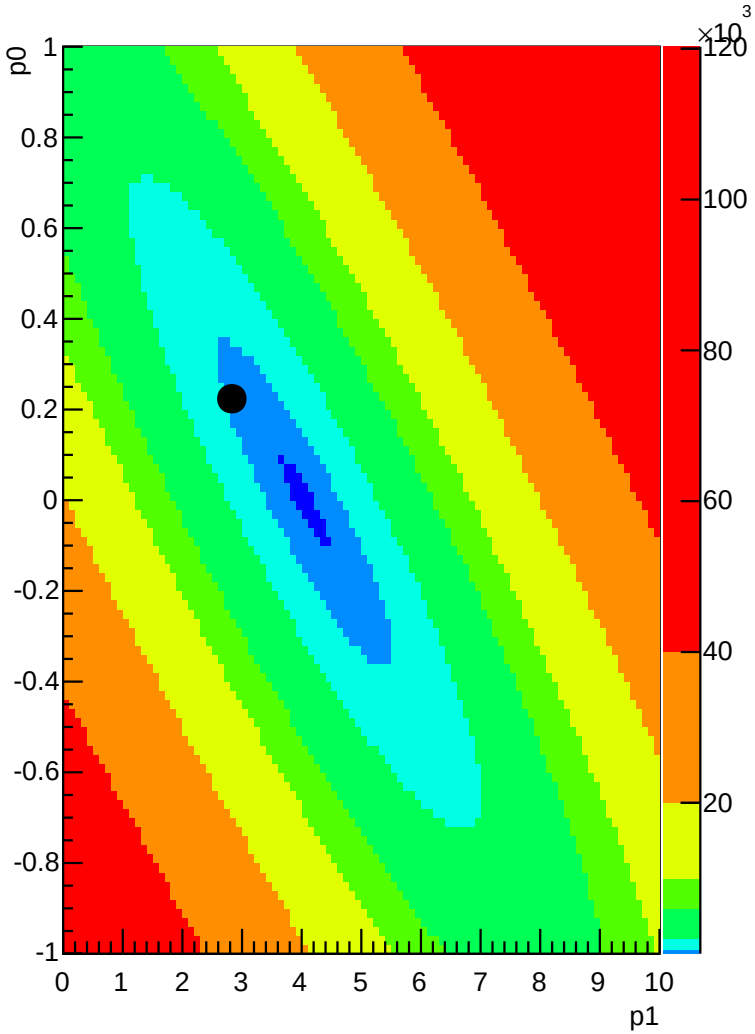
Fit di una retta



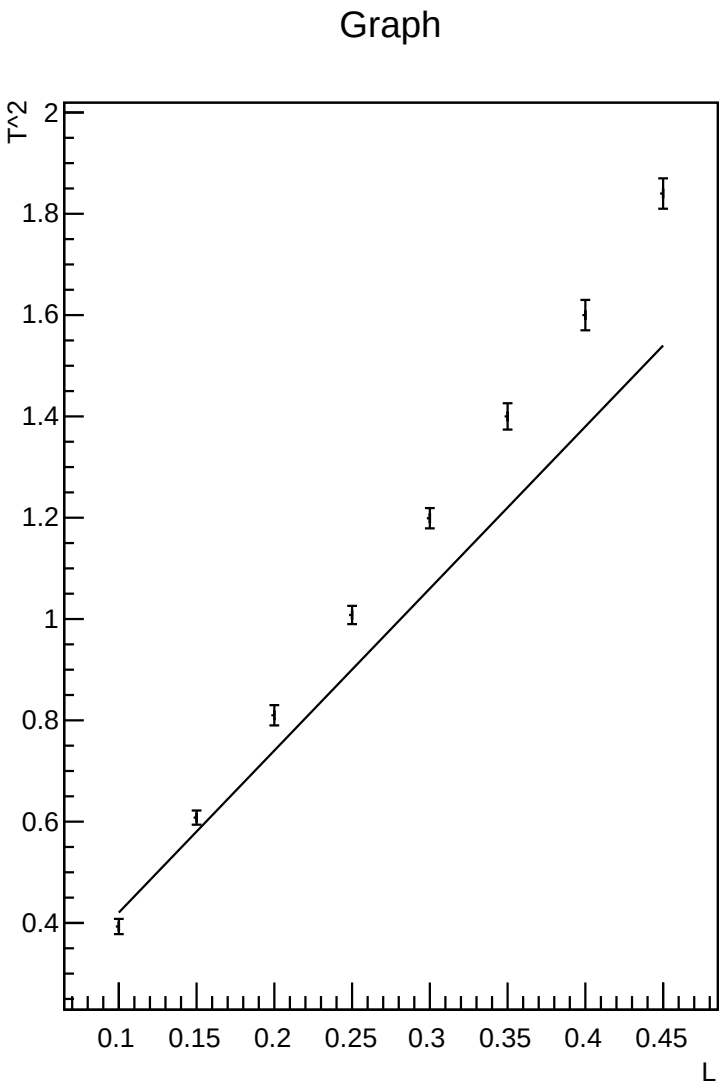


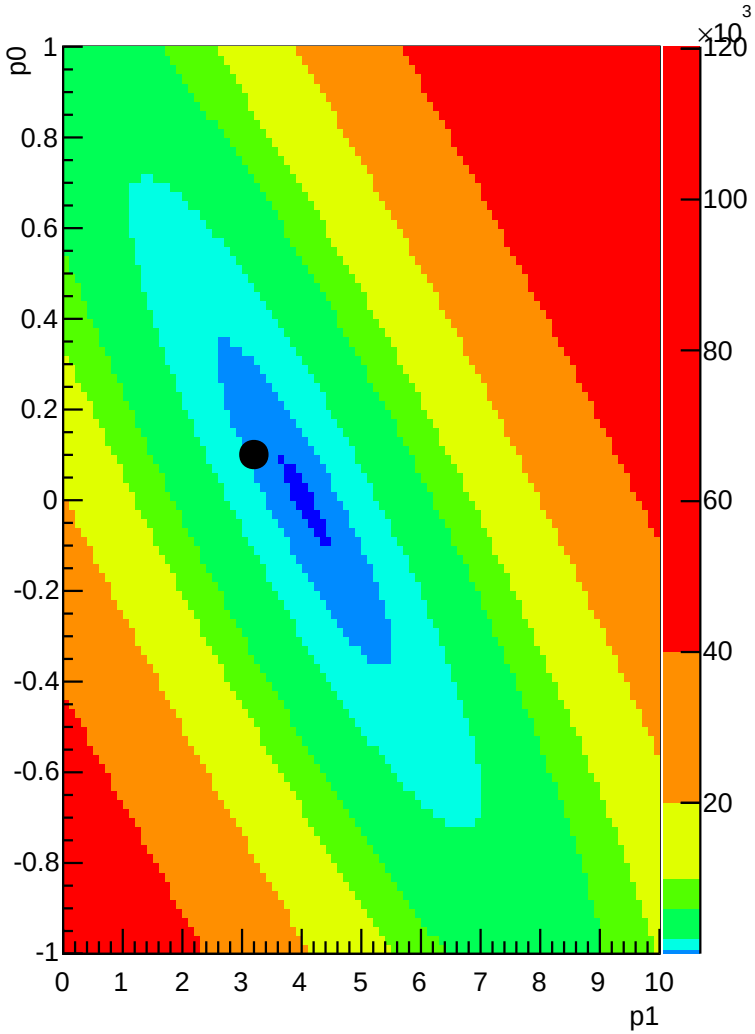
Fit di una retta



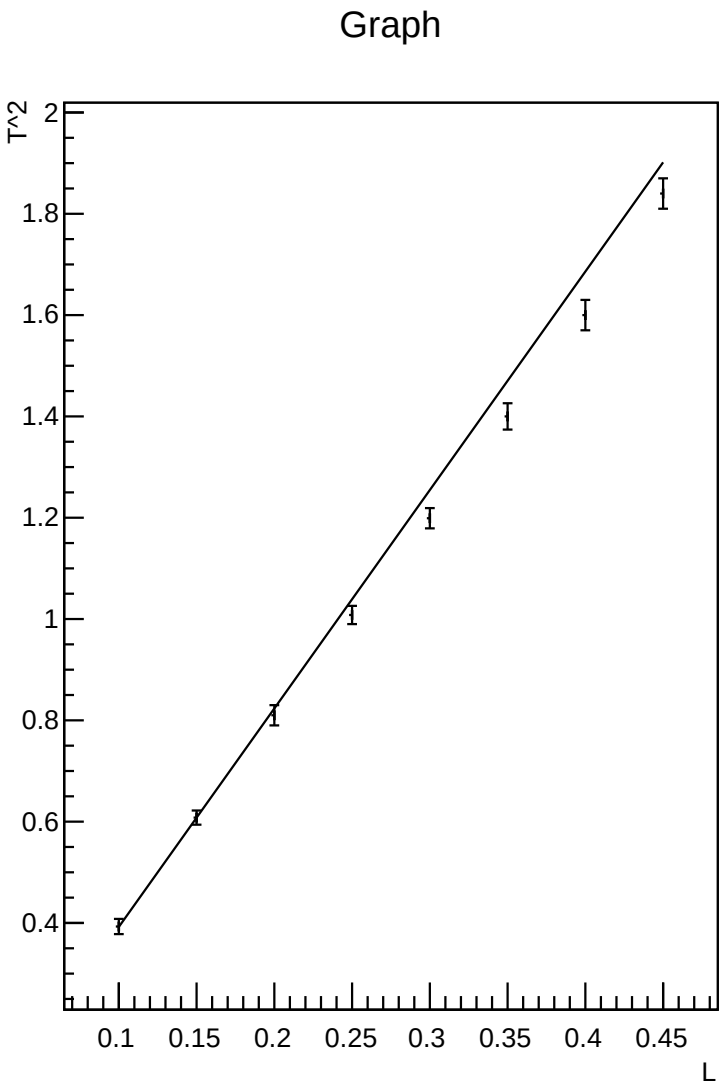


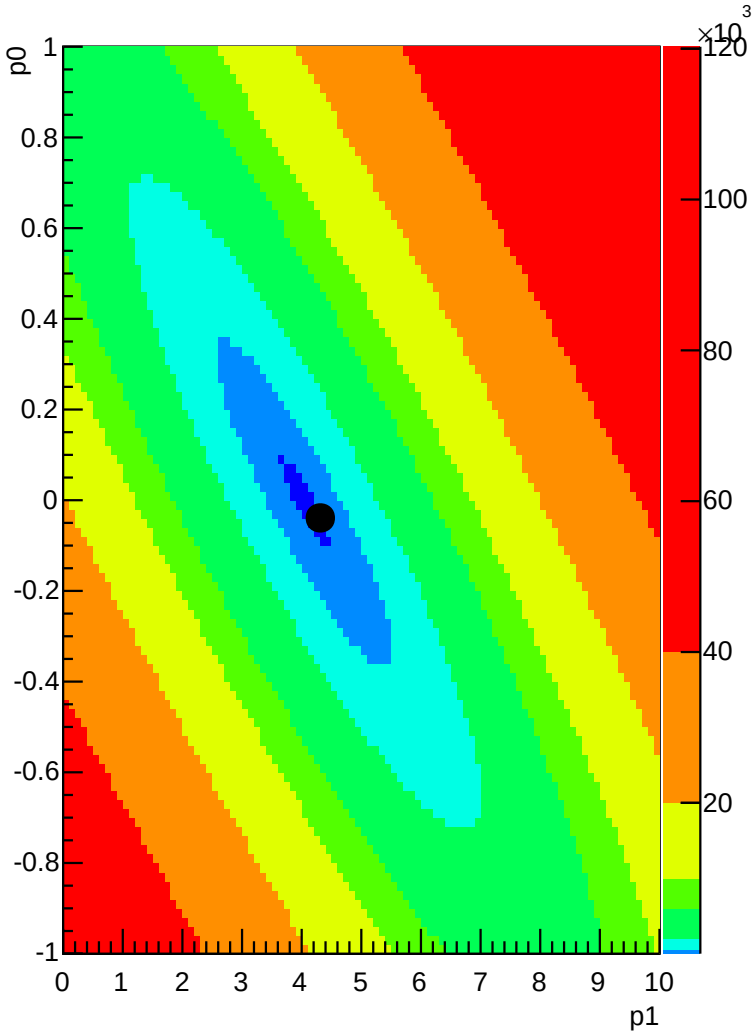
Fit di una retta



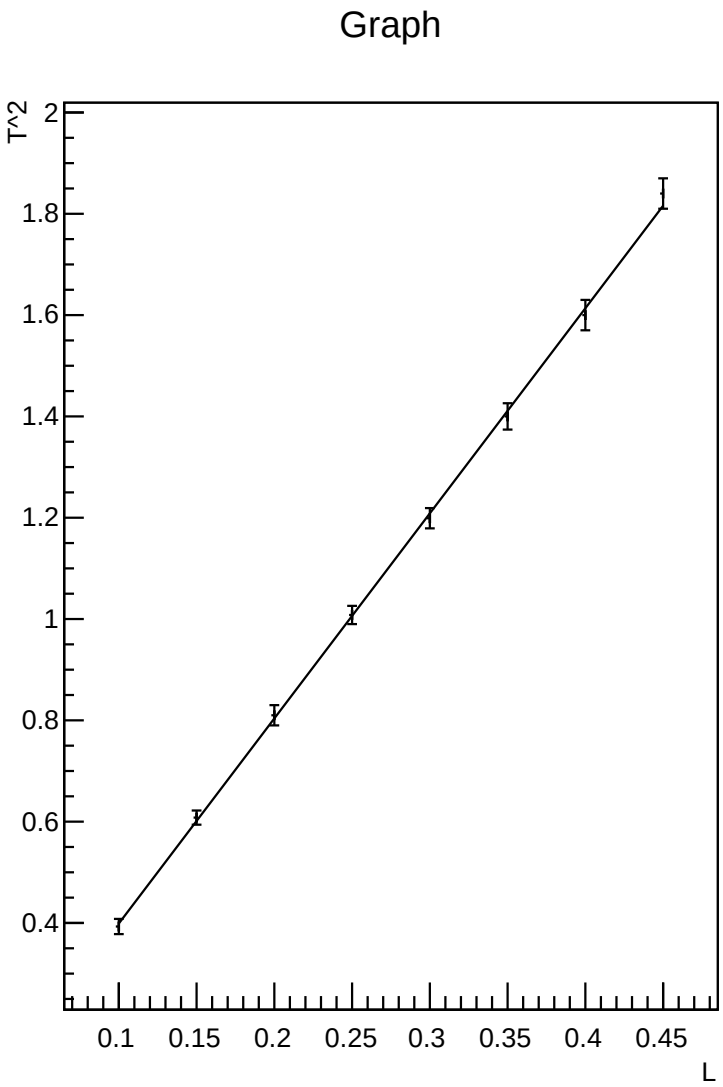


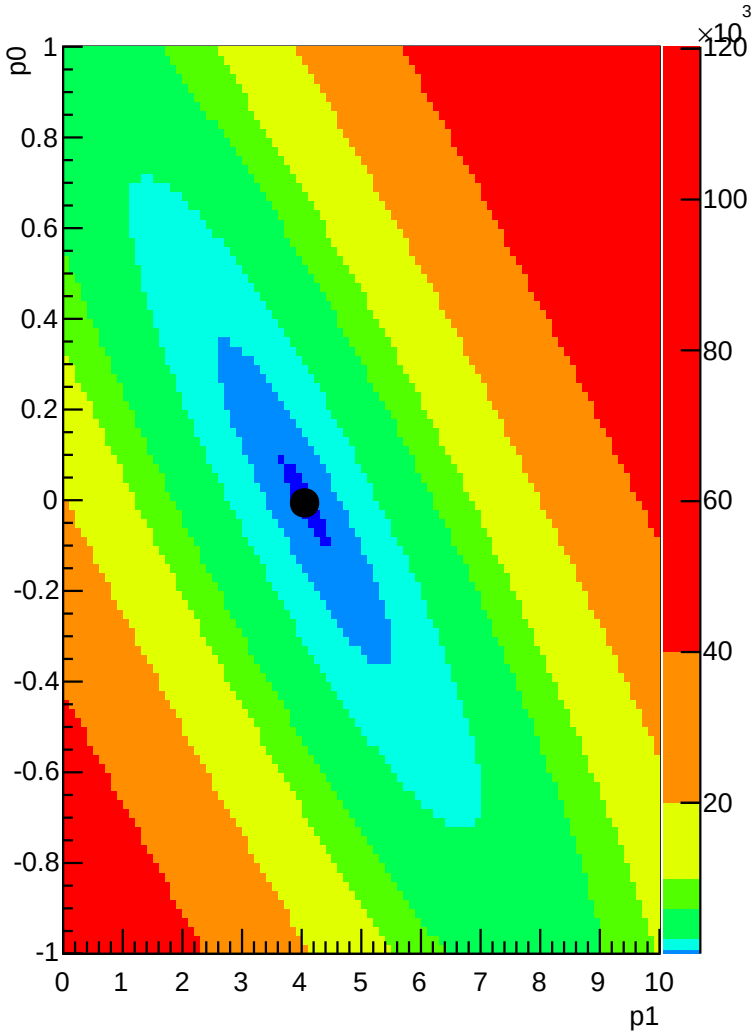
Fit di una retta



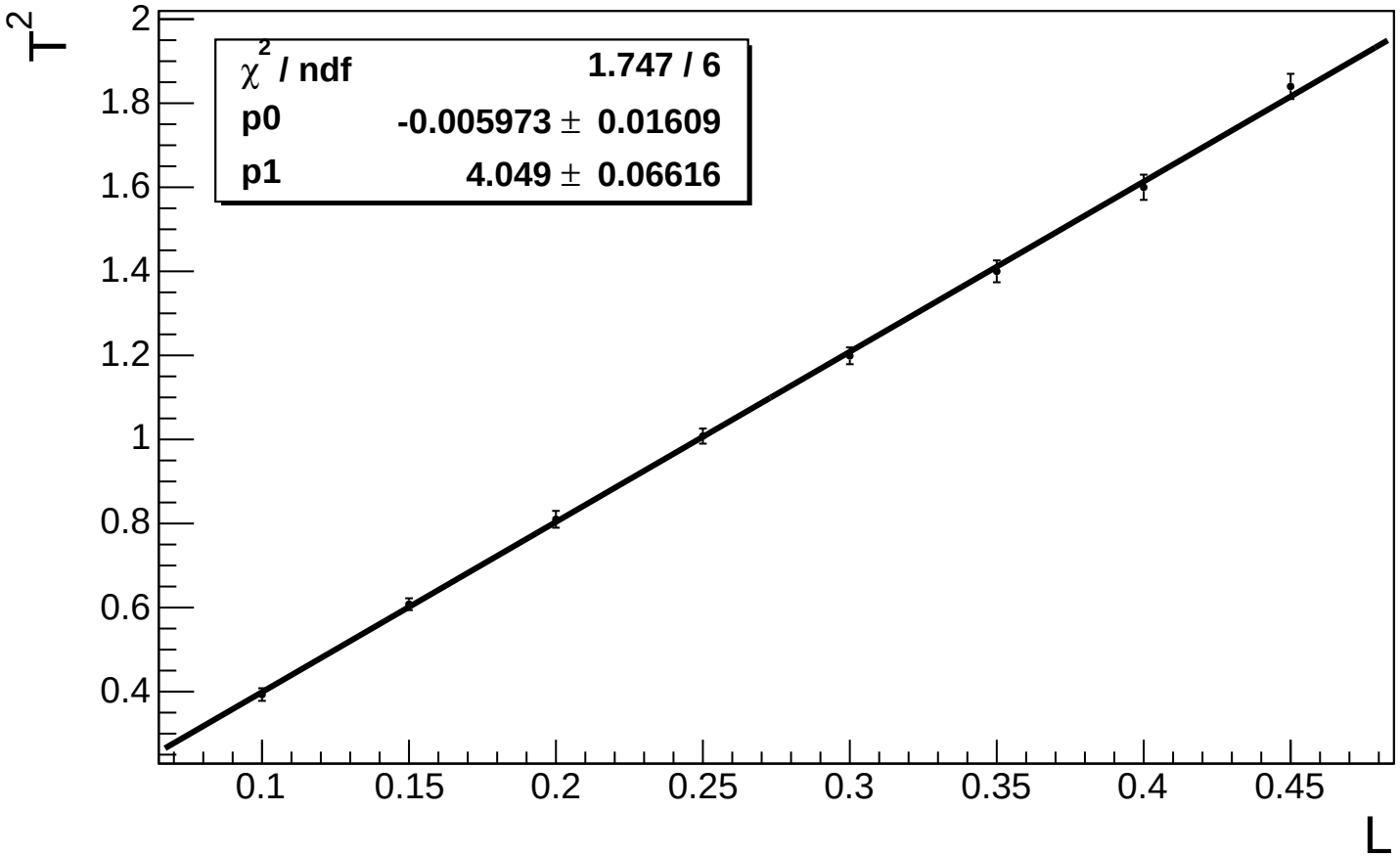


Fit di una retta





Esempio




```
FCN=1.74719 FROM MIGRAD    STATUS=CONVERGED      42 CALLS          43 TOTAL
                             EDM=9.34034e-09    STRATEGY= 1      ERROR MATRIX ACCURATE
EXT  PARAMETER                                        STEP              FIRST
NO.    NAME       VALUE                ERROR          SIZE           DERIVATIVE
  1      p0      -5.97342e-03   1.60871e-02       5.54236e-06         -1.99465e-02
  2      p1       4.04874e+00   6.61605e-02       2.27938e-05         -4.45864e-03
```

Fit con definizione autonoma del χ^2

- In molti casi è utile poter definire autonomamente la funzione (χ^2) da minimizzare. Le motivazioni possono essere molteplici
  - il fit è complesso, ad esempio si vuole descrivere i dati con più funzioni contemporaneamente: fit "simultanei"
  - i dati da fittare sono correlati
- Nel seguito mostreremo, per semplicità, gli strumenti di minimizzazione per il caso lineare.

Minuit

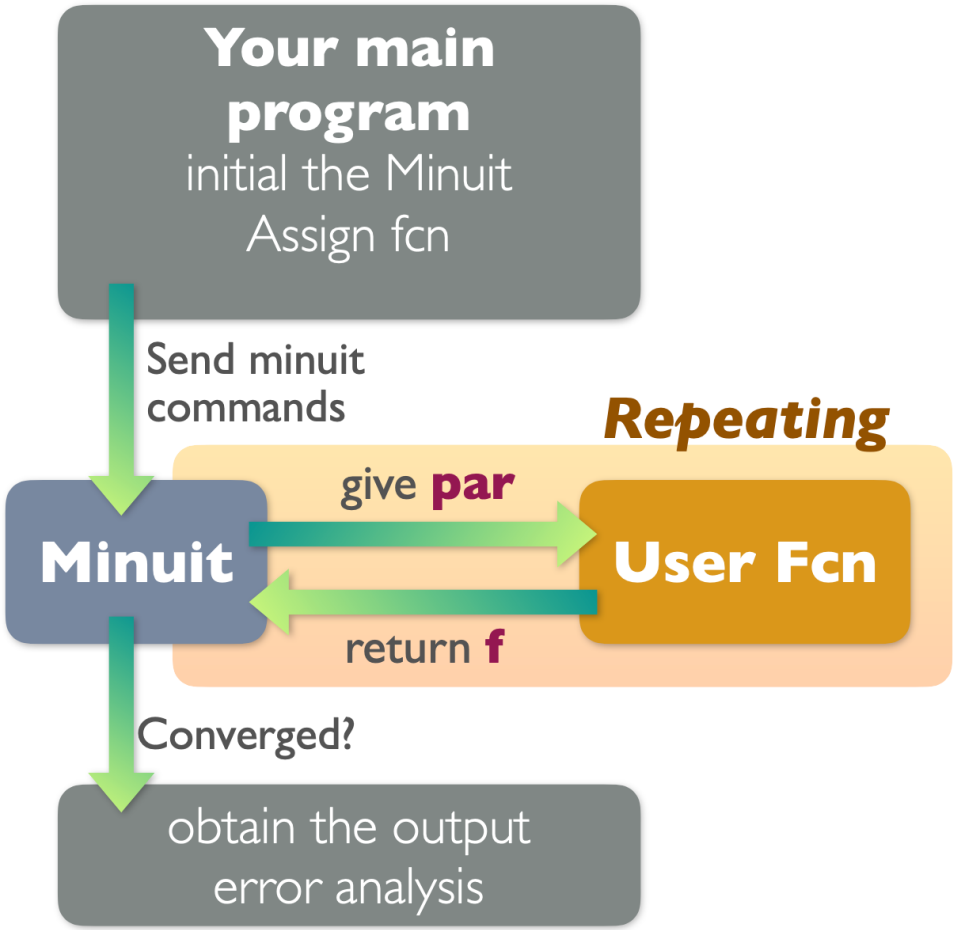
- In questa lezione tratteremo il pacchetto MINUIT
- Minuit è concepito come uno strumento per trovare il minimo di una funzione multiparametrica ed analizzarne la forma intorno al minimo stesso.
- L'applicazione principale è l'inferenza statistica. Lavorando con funzioni di χ^2 o log-likehood, si determinanto, data una funzione parametrica supposta descrivere dati, i parametri e le loro incertezze, comprese le correlazioni tra i parametri.
- Scritto inizialmente in fortran (CERNLIB) è stato tradotto in C++ e inglobato in ROOT
- La classe ROOT che implementa MINUIT è TMinuit
- Anche se non è instanziato esplicitamente, TMinuit viene instanziato implicitamente dal metodo Fit. Il puntatore

  `gMinuit`

  è un puntatore globale all'ultimo oggetto TMinuit istanziato.

Minuit workflow

- Il programma principale deve inizializzare la classe TMinuit e fornire la funzione da minimizzare
- Si definiscono i parametri (variabili o costanti)
- Si danno comandi a Minuit che procede alla minimizzazione chiamando la funzione fornita.



Algoritmi per la ricerca del minimo/calcolo degli errori

Algoritmi più comuni:

- **MIGRAD**: itera il calcolo del gradiente della funzione (direzione della variazione) fino a trovare il minimo. Il calcolo della matrice di errore viene aggiornato ad ogni passo (e determina la lunghezza del passo).
- **HESSE** : calcola la matrice di errore dalle derivate seconde nel minimo. Fornisce errori simmetrici (assumendo che la funziona sia parabolica in un intorno del minimo)
- **MINOS** : calcola gli errori cercando esplicitamente i punti in cui $\Delta\chi^2 = 1$ o $\Delta(-ln\mathcal{L}) = 0.5$. Gli errori possono essere asimmetrici se il minimo non è parabolico. Preciso ma time-consuming.

Il comando di minimizzazione è inviato dal metodo `Command` di TMinuit:

```
int Command (const char *command)
```

La sequenza di chiamate è MIGRAD/HESSE/MINOS.

Definizione χ^2 (C++)

```
1  namespace data{
2    vector<double>  x, y, ex, ey;
3  }
4
5  double fun(const double *x,const double *par){
6    return par[0]*(*x)+par[1];
7  }
8
9  void fcn(int &npar,double *gin,double &f,double *par,int iflag){
10   f = 0.0;
11   for (int i=0;i<data::x.size();i++){
12     f += pow((data::y[i]-fun(&data::x[i],par) )/data::ey[i], 2);
13   }
14 }
```

- L'utente deve sempre fornire una funzione con questo prototipo
- Significato dei parametri in input/output:
  - `npar` numero di parametri (input)
  - `gin` vettore delle derivate prime della funzione nei parametri (opzionale, output)
  - `f` valore della funzione (output)
  - `par` vettori di parametri (costanti o variabili) (input)
  - `iflag` intero che indica lo stadio di minimizzazione (input)

Per iniziare è sufficiente definire `f` sulla base di `par`.

Definizione χ^2 (C++)

```
1
2  void fitlin(){
3    ifstream file("pendolo.dat");
4    double x,y,ex,ey;
5    while (file >> x >> y >> ex >> ey){
6      data::x.push_back(x); data::y.push_back(y); data::ex.push_back(ex); data
           ::ey.push_back(ey);
7    }
8
9    // Define the minimization problem
10
11   // Minimize
12
13
14   // Get result
15
16 }
```

Interfaccia Minuit in Python (iminuit)

- Un'alternativa è l'interfaccia Python a Minuit, indipendente da ROOT. Per installare il modulo (se non fosse presente)

  ```
  python -m pip install iminuit
  ```

  Per chi ha python2 consigliamo:

  ```
  python -m pip install iminuit==1.3.8
  ```

  per chi ha python3 invece:

  ```
  python3 -m pip install iminuit
  ```

- Se non avete pip installatelo seguendo le istruzione che trovate a https://pip.pypa.io/en/stable/installing/.
  pip è un installatore automatico per Python che installa i pacchetti compatibili con la vostra versione di python.
- Manuali e reference guide di iminuit:
  https://iminuit.readthedocs.io/en/stable

Interfaccia Minuit in Python (iminuit)

Funzionamento del modulo Minuit in iminuit

- La funzione da minimizzare può essere definita in due modi:
  - Parametri espliciti

    ```
    def fcn(a, b, c): ...
    Minuit(fcn,...)
    ```

    potete passare i parametri con assegnazione per nome (nome=...)
  - Numpy array

    ```
    def fcn(x): ...
    Minuit(fcn,...)
    ```

    `x` deve essere un numpy array.
- Tutti gli altri parametri di Minuit sono *default*:
  - `errordef` definisce $\Delta\chi^2$ per il calcolo degli errori (default 1)
  - `print_level` definisce il livello di output (default 0)

Definizione χ^2(iminuit)

```
1  from   iminuit import Minuit
2  import numpy as np
3
4  def f(x,a,b):
5      return a*x+b
6
7  def chi2(par):
8      val = 0
9      for i in range(0,len(x)):
10         val = val + ((y[i]-f(x[i],par[0],par[1]))/ey[i])**2
11     return val
12
13 x  = np.array([]); y  = np.array([]); ex = np.array([]); ey = np.
       array([])
14 for line in open("pendolo.dat"):
15     dt  = line.split()
16     if len(dt)!=4:
17         continue
18     x   = np.append(x,  float(dt[0])); y   = np.append(y,  float(dt
           [1]))
19     ex  = np.append(ex, float(dt[2])); ey  = np.append(ey, float(dt
           [3]))
```

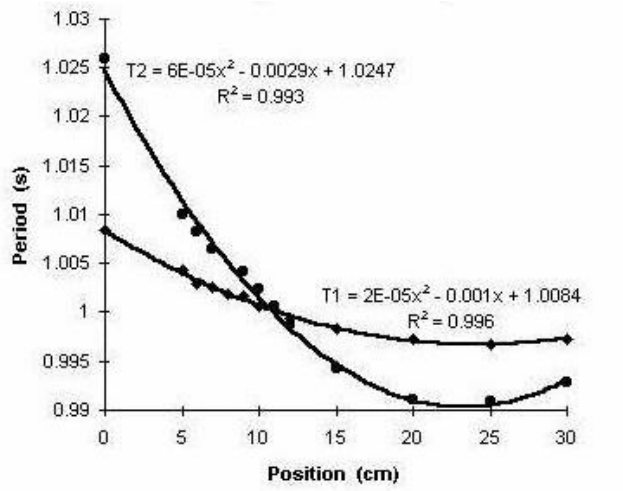
Definizione χ^2(iminuit)

```
1  from   iminuit import Minuit
2  import numpy as np
3
4  def f(x,a,b):
5      return a*x+b
6  def chi2(a,b):
7      val = 0
8      for i in range(0,len(x)):
9          val = val + ((y[i]-f(x[i],a,b))/ey[i])**2
10     return val
11 x  = np.array([]); y  = np.array([]); ex = np.array([]); ey = np.
       array([])
12 for line in open("pendolo.dat"):
13     dt  = line.split()
14     if len(dt)!=4:
15         continue
16     x   = np.append(x,  float(dt[0])); y   = np.append(y,  float(dt
           [1]))
17     ex  = np.append(ex, float(dt[2])); ey  = np.append(ey, float(dt
           [3]))
```

Fit simultanei

La possibilità di costruire autonomamente la funzione di χ^2 consente di implentare strategie complesse di fit che permettano di estrarre direttamente la grandezza di interesse.

- Ad esempio una delle esperienza del primo anno consisteva nel determinare g con il pendolo di Kater:
  - si fittavano le due curve (periodo vs distanza dal perno): il punto di intersezione fornisce il periodo di isocronia.
  - problemi: propagazione complessa dell'errore sul punto di intersezione. Approssimazione (arbitraria): prendo come errore sul punto di isocronia l'errore medio sui periodi.



- Soluzione: di esegue un fit simultaneo delle due serie di dati con le funzioni:

$$f_2(x) = a_2(x - x_0)^2 + b_2(x - x_0) + T_0$$
$$f_1(x) = a_1(x - x_0)^2 + b_1(x - x_0) + T_0$$

Il valore fittato del parametro T_0 fornisce direttamente, senza approssimazioni, il periodo di isocronia con il suo errore.

Esercizio

Dall'esame del 15/01/2020:

Si vuole determinare il periodo di isocronia di un pendolo di Kater. Si sono registrati i periodi, per ciascuno dei due perni, in funzione della distanza della massa mobile.
Si vuole eseguire un file simultaneo ai due grafici usando come le funzioni:

$$f_1(x) = \alpha_1(x - x_0)^2 + \alpha_2(x - x_0) + T$$
$$f_2(x) = \beta_1(x - x_0)^2 + \beta_2(x - x_0) + T$$

Sfruttando la traccia `skel3.cpp` *si esegua il fit e si stampi il valore di* T *con il suo errore.*

Appendice

Definizione χ^2 (Python)

```
1 def func(x,a,b):
2     return a*x+b
3
4 def fcn(npar, gin, f, par,iflag):
5     chi2 = 0.0
6     for i in range(0,len(x)):
7         chi2 += ((y[i]-func(x[i],par[0],par[1]))/ey[i])**2
8     f.value = chi2
9     #f[0]   = chi2
```

Definizione χ^2 (Python)

```
from   ROOT    import *
from   numpy   import *
from   ctypes  import *
#... chi2 ...
# Acquisizione dati
x  = array([]); y  = array([]); ex = array([]); ey = array([])
for line in open("pendolo.dat"):
    dt  = line.split()
    if len(dt)!=4:
        continue
    x   = append(x,  float(dt[0])); y   = append(y,  float(dt[1]))
    ex  = append(ex, float(dt[2])); ey  = append(ey, float(dt[3]))

# Minuit
minuit = TMinuit(2)
minuit.SetFCN(fcn);
minuit.DefineParameter(0,'par0',4,0.01,0.,0.)
minuit.DefineParameter(1,'par1',0,0.01,0.,0.)
minuit.Command("MIGRAD")

# Risultati
a  = c_double(0.0); b  = c_double(0.0)
ea = c_double(0.0); eb = c_double(0.0)
minuit.GetParameter(0,a,ea);
minuit.GetParameter(1,b,eb);
print("a = %f +- %f, b = %f +- %f"%(a.value,ea.value,b.value,eb.value))
```

Parentesi: uso di `ctypes`

- `ctypes` è un modulo python per gestire variabili di tipo C (o C++)
- la dichiarazione

```
import ctypes
x = ctypes.c_double(2.0)
```

  crea un double (analogamente per alti tipi)
- per accedere al valore e/o modificarlo si usa il metodo `value`

```
x.value = 3
```

- questa classe è fondamentale quando si vuole chiamare una funzione C/C++ che utilizza referenze