



AISF - COMITATO LOCALE DI PERUGIA

---

# GRAFICI PARTE 2

## IMPORTATIONE E SOVRAPPOSIZIONE GRAFICI

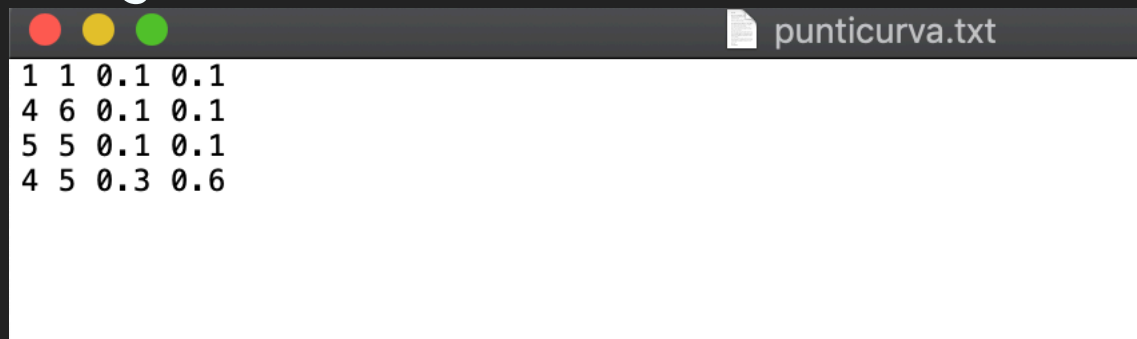
- ▶ Recap della lezione
- ▶ Root è un pacchetto software fornito dal Cern, contenente una serie di funzioni raggruppate in Classi (TGraph, TGraphErrors, TCanvas, .....)
- ▶ Ogni classe ha una serie di funzioni corrispondenti chiamate in termini informatici: Metodi della classe.
- ▶ se vogliamo importare direttamente da file .txt i dati(eventualmente anche con errori) possiamo usare nel caso di `TgraphErrors` un particolare costruttore(passando il parametro `TString`). Per istogrammi vedremo il metodo di importazione

## ► inserimento dati per TGraphErrors direttamente da file.txt

- Notiamo la nuova sintassi (\*g = new) per dichiarare un oggetto. Da ricordare solo il fatto che così facendo si definisce un oggetto dinamico. Il risultato è un'efficienza maggiore per il codice ma "praticamente" non cambia nulla. \*\*Differenza sintattica si usa -> e non "." per chiamare i metodi.

```
{  
    TString nomefile = "/Users/David/Desktop/punticurva.txt"; //percorso  
    TGraphErrors *g = new TGraphErrors(nomefile);  
    g->GetXaxis()->SetTitle("x");  
    g->GetYaxis()->SetTitle("f(x)");  
    g->GetXaxis()->CenterTitle();  
    g->GetYaxis()->CenterTitle();  
  
    g->SetMarkerColor(4); //Markers...  
    g->SetMarkerStyle(20);  
    g->SetTitle("titolo");  
    g->Draw("ap");  
}
```

- si può passare direttamente il "path" del file(txt) contenente i dati
- il grafico viene creato esattamente allo stesso modo delle macros precedenti



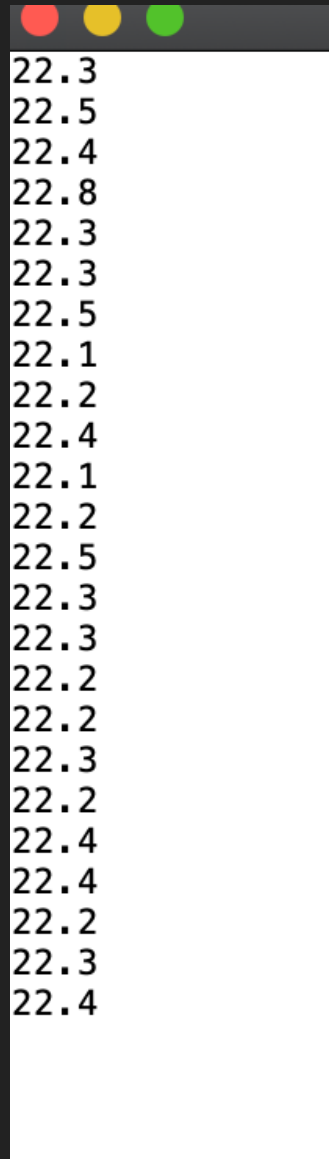
punticurva.txt

1	1	0.1	0.1
4	6	0.1	0.1
5	5	0.1	0.1
4	5	0.3	0.6

scrivo i punti in 4 colonne-> nell'ordine( x ,y ,errx ,erry)

## ► inserimento dati per TH1F direttamente da file.txt

Nel file.txt inserire i dati per l'istogramma in colonna o in riga ma senza virgole o altra punteggiatura di separazione--> il metodo di importazione non funziona



```
22.3
22.5
22.4
22.8
22.3
22.3
22.5
22.1
22.2
22.4
22.1
22.2
22.5
22.3
22.3
22.2
22.2
22.3
22.2
22.4
22.4
22.2
22.3
22.4
```

il breve codice in allegato (**IstoImport.cpp**) dunque permette di importare i dati per istogramma senza chiamare "N" volte il comando Fill  
ma "riempiendo" l'oggetto istogramma tramite un semplice ciclo while

## ▶ \*\*\*\*\*NB\*\*\*\*\*

- ▶ come lanciare più volte la stessa macro?
- ▶ una soluzione "rozza" è quella di uscire e rientrare su Root in modo da terminare la sessione e non avere più in memoria gli oggetti creati alla prima chiamata della macro

```
David — root.exe ◀ root — 80x24
MacBook-Pro-di-David:~ David$
MacBook-Pro-di-David:~ David$ root
-----
| Welcome to ROOT 6.12/06                               http://root.cern.ch |
|                                                         (c) 1995-2017, The ROOT Team |
| Built for macosx64                                     |
| From tag v6-12-06, 9 February 2018                     |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.' |
-----

root [0] .x IstoImport.cpp
inserte nbin, xmin e xmax
100
22
23
Info in <TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [1] .x IstoImport.cpp
/Users/David/root/macros/IstoImport.cpp:3:12: error: redefinition of 'nbin'
    double nbin,xmin,xmax;
           ^
/Users/David/root/macros/IstoImport.cpp:3:12: note: previous definition is here
    double nbin,xmin,xmax;
           ^
root [2]
```

- ▶ primo lancio(inserendo i parametri)
- ▶ oggetti restano in memoria.
- ▶ Al secondo lancio ERRORE
- ▶ gli oggetti che creo(2' volta) risultano già esistere

▶ ogni volta uscire da Root per rieseguire può essere frustrante

## ▶ \*\*\*\*\*NB\*\*\*\*\*

- ▶ Esiste un modo per ovviare a tale procedura modificando la macro.
- ▶ Basta chiamare una funzione all'interno della macro

```
2
3
4 //devono essere oggetti dinamici|
5 void g(int n){
6   TH1F *hist5 = new TH1F("Isto", "Titolo",n,22,23);
7   //0.70-0.80 intervallo dell asse x
8
9   hist5->Fill(22.3);
10  hist5->Fill(22.5);
11  hist5->Fill(22.4);
12  hist5->Fill(22.8);
13  hist5->Fill(22.3);
14  hist5->Fill(22.3);
15  hist5->Fill(22.5);
16  hist5->Fill(22.1);
17  hist5->Fill(22.2);
18  hist5->Fill(22.4);
19  hist5->Fill(22.1);
20  hist5->Fill(22.2);
21  hist5->Fill(22.5);
22  hist5->Fill(22.3);
23  hist5->Fill(22.3);
24  hist5->Fill(22.3);
25  hist5->Fill(22.2);
26  hist5->Fill(22.2);
27  hist5->Fill(22.3);
```

- ▶ la funzione g(void) prende in input il numero di bin
- ▶ al termine dell'esecuzione "cancella" tutti gli oggetti creati
- ▶ posso richiamare la funzione(anche cambiando parametri)

chiamo più volte la macros della stessa sessione  
senza uscire e rientrare ogni volta da Root

Vediamo esempio di esecuzione

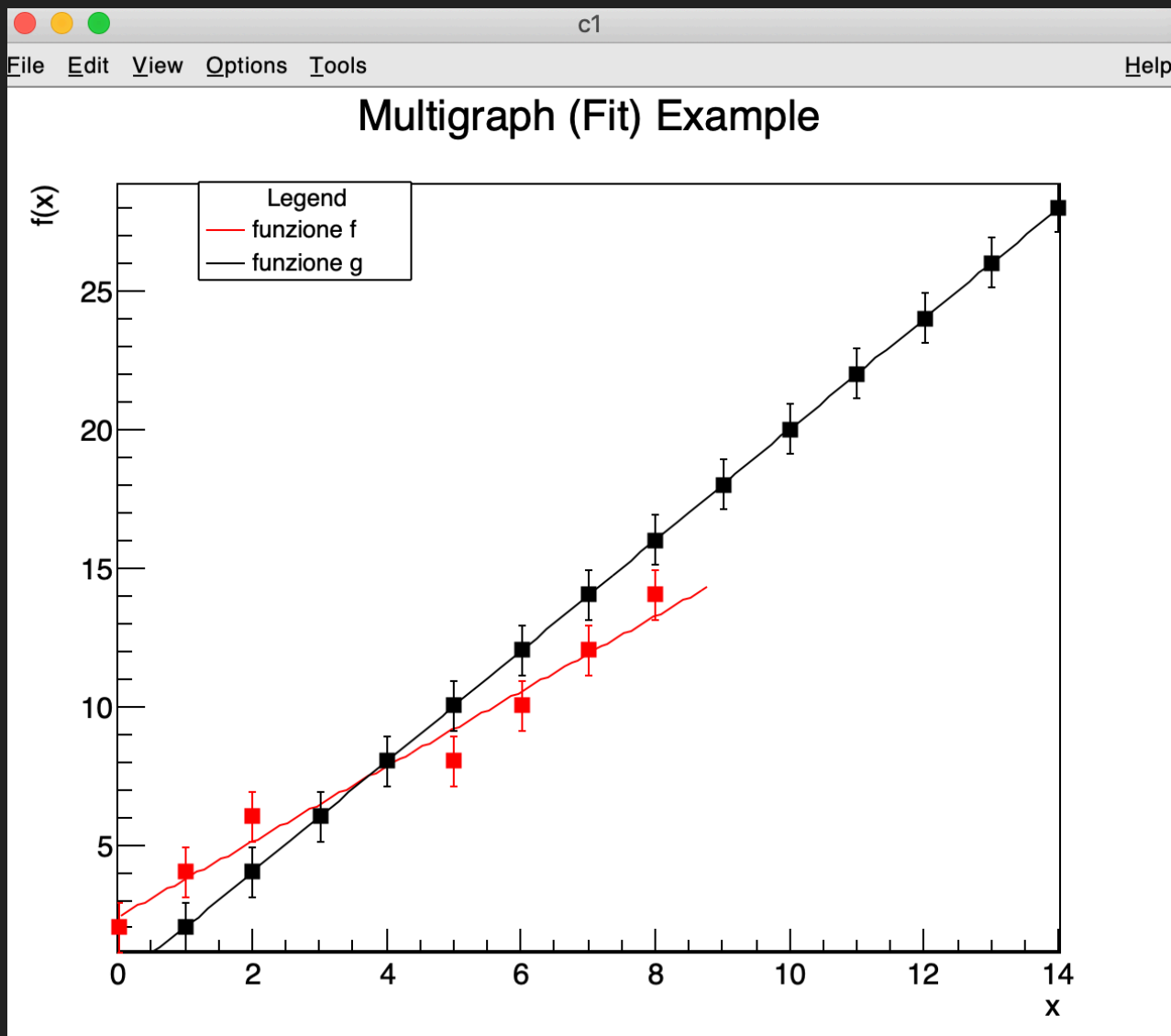
## ► Sovrapposizione grafici

- Esiste la nuova classe TMultigraph

si creano singolarmente i grafici, si esegue il fit dei singoli dati tramite il metodo Fit(".....") e poi si aggiungono i risultati al multigraph tramite il metodo **Add**.

- Vediamo in dettaglio la macros
- Multigraph.cpp in allegato

- Il risultato sarà.....



► Il grafico (multifit) finale con legenda

```
root [13]
root [13] g()
*****
FIT funzione G-->
FCN=1.4087e-12 FROM MIGRAD STATUS=CONVERGED 36 CALLS 37 TOTAL
EDM=2.81741e-12 STRATEGY= 1 ERROR MATRIX ACCURATE
EXT PARAMETER
NO. NAME VALUE ERROR STEP SIZE FIRST DERIVATIVE
1 p0 1.00001e-02 5.08718e-01 1.17478e-04 -8.16595e-06
2 p1 2.00000e+00 5.96841e-02 1.37828e-05 -8.36476e-05
*****
FIT funzione F-->
FCN=4.02641 FROM MIGRAD STATUS=CONVERGED 31 CALLS 32 TOTAL
EDM=7.84104e-07 STRATEGY= 1 ERROR MATRIX ACCURATE
EXT PARAMETER
NO. NAME VALUE ERROR STEP SIZE FIRST DERIVATIVE
1 p0 2.38551e+00 5.94257e-01 3.72461e-04 -1.33030e-03
2 p1 1.35916e+00 1.17326e-01 7.35374e-05 -1.54767e-02
Info in <TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [14]
```

► i risultati del Fit stampati su Terminale