Laboratorio II – 1° modulo
Lezione 12

Esercizi

Indice

- **Esercizio 1:** Implementazione di una classe per l'analisi dati
- **Esercizio 2:** Confronto tra due metodi per il fit di un istogramma con un numero "basso" di conteggi (minimi quadrati e massima verosimiglianza)
- **Esercizio 3:** … esercizio orbite pianeti …
- **Esercizio 4:** … esercizio Mandelbrot …

Esercizio 1

Si vuole implementare una classe “Analyzer” per l’analisi dati che sia in grado di eseguire le seguenti operazioni:

- Lettura dei dati da un file, riempimento di opportuni container per i valori letti, calcolo dei valori minimo e massimo della `x`, riempimento di un opportuno oggetto per la visualizzazione grafica (`TH1D` nel caso di esperimento di conteggio, `TGraphErrors` nel caso di misure)
- Calcolo della media campionaria, della deviazione standard campionaria e dell’errore sulla media (con la possibilità di usare la media pesata e il relativo errore quando, oltre ai valori numerici, vengono forniti gli errori ad essi associati)
- Fit dell’istogramma o del grafico (dovrà essere lasciata la possibilità, a chi utilizza la classe, di scegliere la funzione e il range)

Esercizio 1

- Calcolo del Chi-2, del numero di gradi di libertà e del p-value associati al fit dell'istogramma o del grafico
- Calcolo compatibilità tra due misure ($M_1 \pm \sigma_1$ e $M_2 \pm \sigma_2$) in base a un test Gaussiano oppure t-Student (selezionabili tramite un parametro)
- Calcolo della curva di livello per $\Delta Min^2 = delta$ della funzione utilizzata per il fit

Realizzare in seguito un `main` che istanzi un oggetto di tipo `Analyzer` ed esegua tutte le operazioni descritte sopra, stampando a schermo i risultati opportuni

Su e-learning trovate i file `data1.txt` (lezione 9 – esercizio 1) e `ese01Fit.txt` (lezione 11 – esercizio 1) per testare il codice nel caso di istogramma e grafico rispettivamente, oltre che al file dei prototipi della classe, `Analyzer.h`

Esercizio 2

Si vuole realizzare un programma per confrontare il risultato del fit di un istogramma contente pochi conteggi utilizzando i metodi dei minimi quadrati e della massima verosimiglianza

Il file `datiS.txt` contiene `N` numeri casuali campionati da una pdf uniforme: `pdf(x) = 0.1` per $0 \leq x \leq 10$, `0` altrimenti

Lo si legge e si riempiono – con gli stessi dati – due istogrammi `TH1D` (che avranno contenuto identico). Si usa poi il metodo `Fit()` per interpolare gli istogrammi con una funzione costante, usando per il primo istogramma il metodo dei minimi quadrati (opzione "`C`") e per il secondo istogramma quello della Likelihood (opzione "`L`"). Si confrontano i risultati ottenuti nei due casi con il risultato atteso. Si verifica quindi come è calcolato il Chi-2 associato ai due fit usando le formule descritte nella traccia indicata qui di seguito

Esercizio 2

1. Definire due istogrammi (e.g. `TH1D* histoC` e `TH1D* histoL`) entrambi di `10` bin e di estremi `0` e `10`. Riempirli con le `N` estrazioni lette dal file `datiS.txt` usando il metodo `Fill()`. Nel ciclo di lettura del file calcolare `N`. Stampare quindi a schermo:
   - il numero `N` di estrazioni effettuate dalla `pdf(x)` uniforme
   - il contenuto k_i di ciascun bin dell'istogramma (dove $i = 0, 1, 2 \ldots 9$)
   - l'errore e_i che ROOT ha associato a ciascun bin dell'istogramma
   - un commento su quale sia la $pdf(k_i)$ che descrive la variabile aleatoria k_i, e su come abbia fatto ROOT ad associare un errore a ciascun k_i
   - il valore di aspettazione $\mu = E[k_i]$ per la variabile random k_i e la sua deviazione standard $\sigma = Var[k_i]$ determinati conoscendo la $pdf(k_i)$ e il numero `N` di estrazioni (ovviamente tutti i bin hanno lo stesso valore di aspettazione e la stessa varianza e deviazione standard, ricordate che i vostri istogrammi non sono una pdf perché non sono normalizzati per avere area 1)

Esercizio 2

2. Fittare entrambi gli istogrammi con una funzione costante $k_i = K$ definita nell'intervallo `[0, 10)`:
   - fittare `TH1D* histoC` usando il metodo dei minimi quadrati, stampare a schermo il valore di `K` così stimato (sia K_C), il suo errore e il relativo Chi-2
   - fittare `TH1D* histoL` con il metodo della Likelihood, stampare a schermo il valore di `K` così stimato (sia K_L), il suo errore e il relativo Chi-2
3. Confrontare (quantitativamente) il valore ottenuto per la costante `K` nei due casi (K_C e K_L) con il valore atteso $\mu = E[k_i]$ calcolato al punto 1

   Inserire un commento che spieghi che metodo è stato usato per fare il confronto e quale è il suo esito
4. Implementare una funzione `void chiquadro(TH1D* histo, double K)` che usando i valori di k_i contenuti nell'istogramma, e il valore stimato della costante `K`, determini il valore del Chi-2 associato ai due fit considerando le seguenti due definizioni:

Esercizio 2

$$\chi_C = \sum_{i=1}^{N} \frac{(k_i - K)^2}{e^2{}_i}$$ solo per bin non vuoti

$$\chi_L = \sum_{i=1}^{N} \frac{(k_i - K)^2}{e^2{}_i}$$ sia per bin vuoti che per bin non vuoti, nel caso di bin vuoti porre $e^2_i = 1$

Nel primo caso fare la somma solo sui bin a contenuto non nullo ($k_i > 0$). Questo primo metodo è la funzione che ROOT ha minimizzato per stimare `K`. Come si calcolano i gradi di libertà? Nel secondo caso mettere al denominatore $e^2_i = 1$ se il bin ha contenuto `0` ma contare tutti i bin. Questo secondo metodo è associato a quello di Baker-Cousins, usato da ROOT nel fit col metodo Likelihood, in cui si tiene conto dei bin a contenuto nullo. Stampare a schermo i due Chi-2 e i corrispondenti Chi-2 ridotti. Inserire un commento sul confronto fra risultati dei fit e risultati del calcolo, e su come e perché devono essere valutati i gradi di libertà per i due Chi-2

5. Ripetere l'esecuzione del programma leggendo il file `datiL.txt`. Cosa è cambiato? Secondo voi perché?

Esercizio 3

Dati 3 corpi celesti descritti da:

- masse *m1, m2, m3,*
- posizioni iniziali (vettori nel piano dei 3 corpi) *r1, r2, r3*
- velocità iniziali (vettori nel piano) *v1, v2, v3,*

calcolare, usando la legge di gravitazione di Newton (vettoriale), le orbite dei tre corpi e rappresentarle nel piano.

Risolvere le equazioni del moto per via numerica utilizzando le seguenti espressioni (vettoriali):

$$v(t+\Delta t) = a\,\Delta t + v(t)$$

$$r(t+\Delta t) = v\,\Delta t + r(t)$$

In pratica, dopo aver impostato le condizioni iniziali dei 3 corpi celesti (massa, posizione e velocità), aggiornare ad ogni step temporale Δt la velocità e la posizione di ciascuno.

Scegliere la lunghezza dello step temporale Δt in modo da ottenere soluzioni numeriche sufficientemente precise e, allo stesso tempo, ottimizzare il tempo computazionale.

continua …

Esercizio 3

Gestire i calcoli utilizzando due classi:

a) una classe vettore in 2D (**`vett2d`**) che memorizzi le coordinate x e y dei vettori, e che consenta di fare su di essi tutti i calcoli necessari (+, -, =, mod(), etc.)

b) una classe **`planet`** che memorizzi massa, posizione, velocità del corpo celeste, e che consenta di fare con esse i calcoli necessari.

   In particolare, implementare un metodo DoStep che riceva in input i puntatori degli altri due corpi celesti del sistema e lo step temporale:

```
void DoStep(planet *pl1, planet *pl2, double dt);
```

   Questo metodo deve:

   - creare i vettori **`a1`** e **`a2`**, che corrispondono alle accelerazioni del corpo celeste in oggetto per la presenza degli altri due corpi celesti (pl1 e pl2).
   - calcolare l'accelerazione totale: **`a=a1+a2`**
   - aggiornare i vettori velocità e posizione del corpo celeste in oggetto con le formule riportate nella slide precedente **`a=a1+a2`**

Utilizzare gli header files .h delle due classi postati su e-learning, contenenti più dettagli e commenti utili per la realizzazione delle classi

continua …

Esercizio 3

Creare un main in cui si definiscano tre corpi celesti (meglio crearli sulla memoria dinamica tramite **`new`**)

Si definiscano anche 3 oggetti `TGraph` per rappresentare le orbite dei corpi celesti in un `TCanvas`

Impostare un ciclo **`for`** in cui, per ciascun corpo celeste:

- chiamate il metodo `DoStep`
- inserite le coordinate nel `TGraph` corrispondente.

Si suggerisce di partire dal calcolo di un sistema noto (Sole, Terra, Marte), per cui è ragionevole usare uno step Δt = 2 ore.

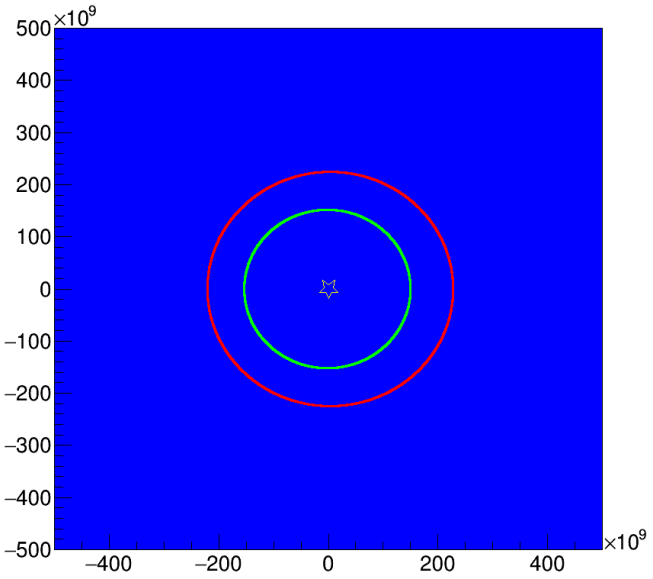
Sole:	r=0	v=0	m=2e+30kg
Terra:	r=150e+9m	v=30e+3m/s	m=5.97e+24kg
Marte:	r=228e+9m	v=24e+3m/s	m=6.42e+23kg

Utilizzare i metodi `Modified()` e `Update()` per eseguire il refresh del Canvas ogni N iterazioni (es: N=30) ed ottenere un'animazione dei moti dei pianeti.

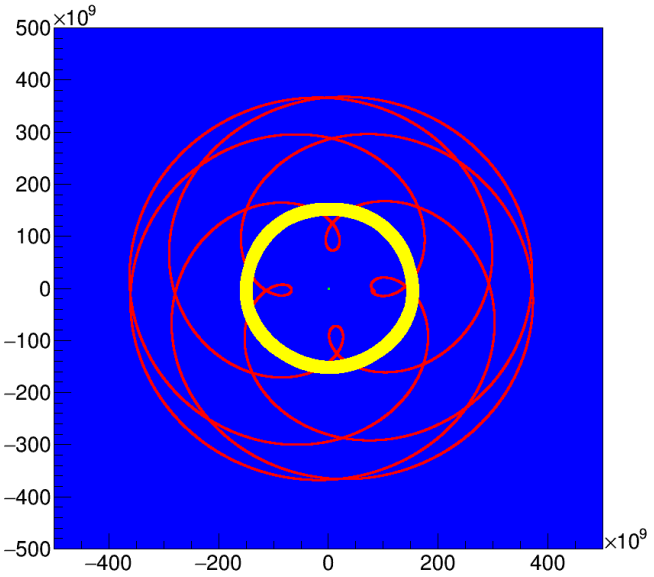
Esercizio 3

Modificare infine il codice del main per dare all'utente la possibilità di scegliere in quale sistema di riferimento (SdR) vuole rappresentare le orbite dei corpi celesti.

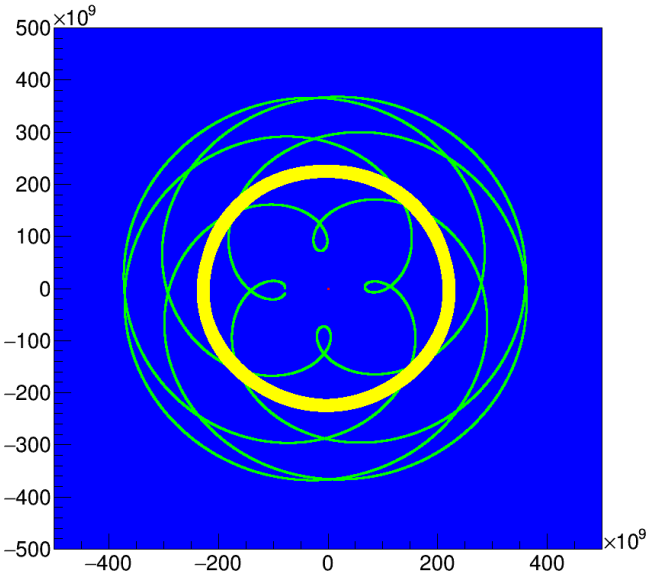
SdR: Sole



SdR: Terra



SdR: Marte



N.B.: Per tenere fissato il range degli assi durante l'esecuzione del programma è necessario invocare i seguenti metodi nella sequenza:

`SetRangeUser` *(applicato a entrambi gli assi),* `Modified()`, `SetLimits` *(applicato a entrambi gli assi),* `Modified()`, `Update()`.

Esercizio 4

Scrivere un codice per rappresentare la struttura frattale dell'insieme di Mandelbrot.

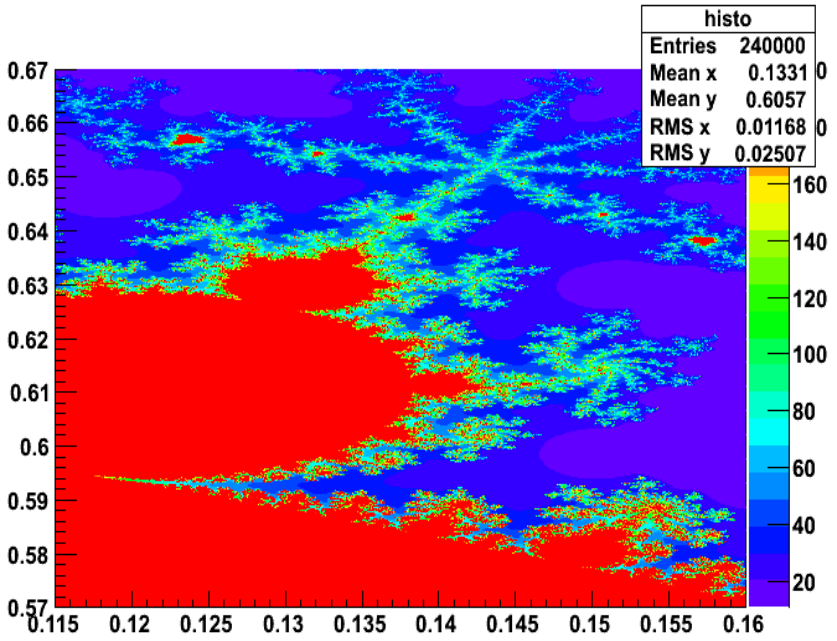
Si consideri il piano dei numeri complessi nel range [-2, 1] x [-1,1]

Ad ogni punto *c* si associ un numero *N* che quantifichi la rapidità della divergenza della successione $z_{n+1} = z_n^2 + c$, con $z_0 = 0$:

→ per ogni *c*, valutiamo a quale iterazione *n* si verifica la condizione:

$$|z_n| > 2$$

Rappresentare graficamente i valori di *N* ottenuti campionando una fitta griglia di numeri complessi nella porzione di piano [-2, 1] x [-1,1], utilizzando un istogramma 2D (consultare le slides seguenti per maggiori dettagli sulla definizione di istogrammi TH2F di Root)



Esercizio 4

- Scegliere un numero massimo di iterazioni M = ~100
- Definire un TH2F nel range [-2, 1] x [-1,1], in modo che i bin siano quadratini di lato 0.005.
- In un doppio ciclo **for**, consideriamo, di volta in volta, ciascun punto al centro dei bin del TH2F:
  - `h2 → GetXaxis() → GetBinCenter(i)` *restituisce il centro del'i-esimo bin sull'asse X (analogo per l'asse Y)*
- Inizializzato $z_0 = 0$, calcolare i valori della successione

  $$z_{n+1} = z_n^2 + c$$ *(usate la classe dei numeri complessi!)*

  e interrompere il ciclo quando si verifica la condizione $|z_n| > 2$
- Incrementare il bin corrispondente con un "peso" pari al numero *n* in cui il ciclo si è interrotto. È possibile usare il metodo:

  ```
  Fill(double x, double y, double w);
  ```

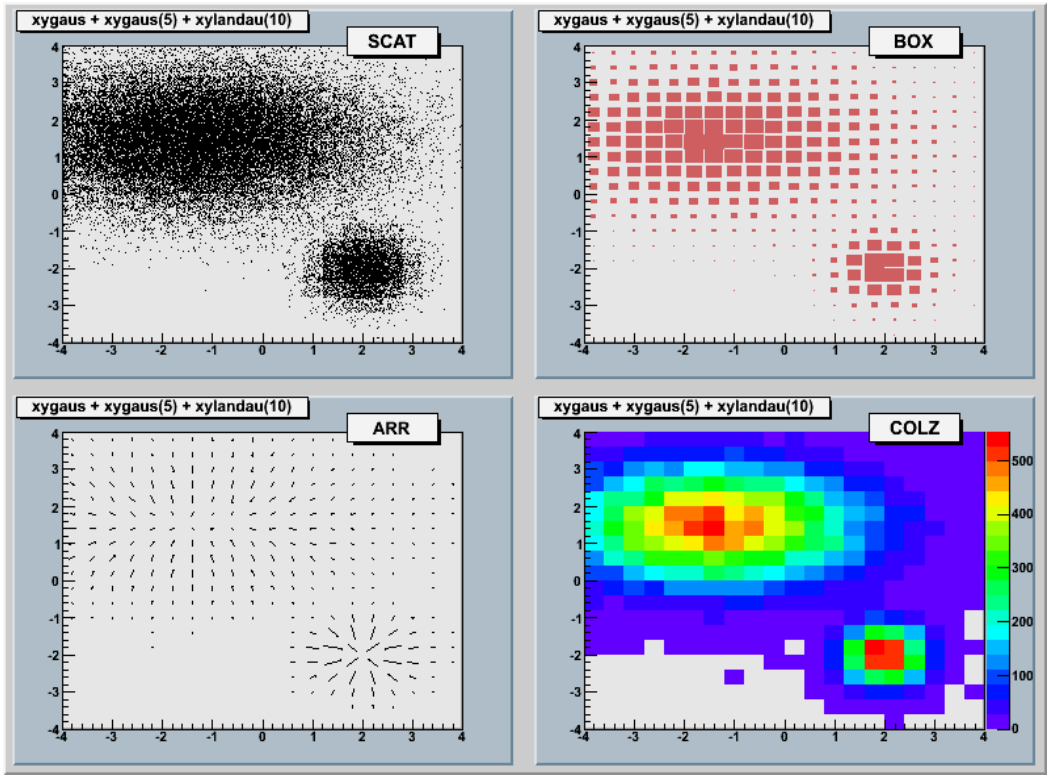
  in cui w è il valore con cui si vuole incrementare il bin.

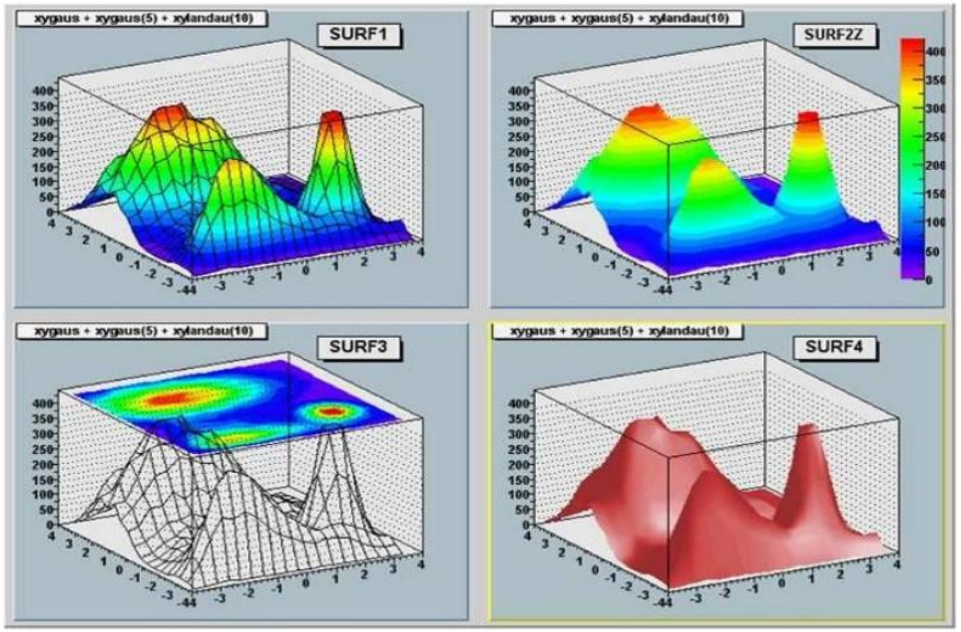
Istogrammi 2D (TH2)

```
#include "TH2.h"
```

In un **TH2** ci sono **due variabili indipendenti** (rappresentate nel piano XY) → ogni bin è un un **rettangolo**!

I conteggi contenuti in ciascun bin sono rappresentati sull'asse z





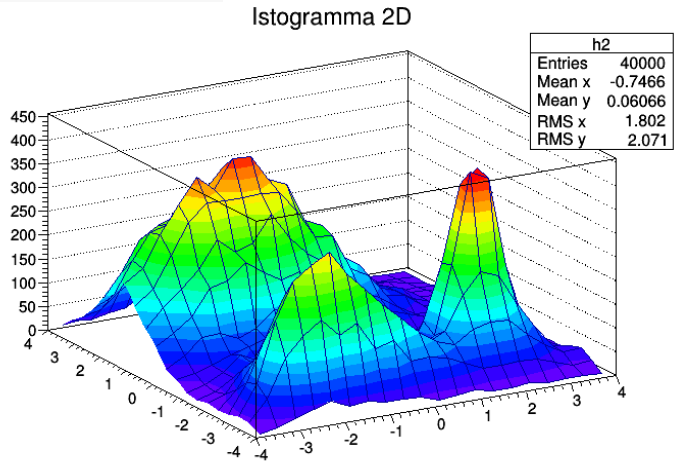
Definire un TH2F

Costruttori: ce ne sono di diversi (come per i TH1F), ad esempio:

- `TH2F(const char* name, const char* title, int nbinsx, double xlow, double xup, int nbinsy, double ylow, double yup)`
- `TH2F(const char* name, const char* title, int nbinsx, const double* xbins, int nbinsy, const double* ybins)`

Per **riempire** un istogramma, posso usare il metodo **`Fill(x,y)`**, al quale passo le coordinate del punto in corrispondenza del quale voglio incrementare il bin di un'unità.

```
TH2F *h2 = new TH2F("h2","Istogramma 2D",20,-4,4,20,-4,4);
ifstream in("DatiTH2F.txt");
double xrandom, yrandom;
while (1)       {
        in >> xrandom>> yrandom;
        if (in.eof()) break;
        h2->Fill(xrandom, yrandom);
}
h2->Draw("SURF1");
```



Disegnare un TH2F

- Esistono diverse opzioni grafiche per rappresentare un TH2F:
  - Densità di punti diversa a seconda del contenuto del bin (“`SCAT`”)
  - Colori diversi a seconda del contenuto di ciascun bin (“`COLZ`”)
  - Contenuto del bin rappresentato sull'asse *z* (“`LEGO`”)
  - … http://root.cern.ch/root/html/THistPainter.html
- Per usarle, basta specificarle come argomenti del metodo Draw:
  - `h2->Draw("option");`