Laboratorio di Meccanica e Termodinamica

Modulo 2 (ROOT) AA 2021/22

Scritto del 06/09/2022

Si scriva la parte rilevante e autoconsistente del codice di una macro di ROOT in cui:

1. Si definisce un istogramma unidimensionale con 1000 bin, in un range da 0. a 10.;

2. In un ciclo si riempie l'istogramma con 10^7 occorrenze di una variabile casuale estratta da una distribuzione gaussiana con media μ=5 e σ=1;

3. Al termine del ciclo si esegue il fit dell'istogramma con una funzione gaussiana esplicitamente definita attraverso un TF1 (tre parametri liberi: ampiezza,media e deviazione standard) e si stampano a schermo:

    a. la media e la RMS dell'istogramma risultante con le rispettive incertezze.

    b. Il valore dei tre parametri della funzione dopo il fit, con rispettive incertezze, e Il χ2 ridotto del fit.

```
#include "TH1.h"
#include "TF1.h"
#include "TRandom.h"
using namespace std;

void esempio()
{
    TH1F *h1 = new TH1F("h1", "Istogramma unidimensionale", 1000, 0., 10.);
    for (int i{0}; i < 1000000; ++i)
    {
        Float_t x = gRandom->Gaus(5., 1.);
        h1->Fill(x);
    }

    TF1 *fit = new TF1("gaussFit", "gaus(0)");

    // h1->Draw();
    cout << "media istogramma = " << h1->GetMean() << " +/- " << h1->GetMeanError() << '\n';
    cout << "dev. standard istogramma = " << h1->GetRMS() << " +/- " << h1->GetRMSError() << '\n';

    h1->Fit("gaussFit", "Q");
    cout << "ampiezza fit = " << fit->GetParameter(0) << " +/- " << fit->GetParError(0) << '\n';
    cout << "media fit = " << fit->GetParameter(1) << " +/- " << fit->GetParError(1) << '\n';
    cout << "dev. standard fit = " << fit->GetParameter(2) << " +/- " << fit->GetParError(2) <<
'\n';
    cout << "chi quadro fit = " << fit->GetChisquare() / fit->GetNDF() << '\n';
}
```

Laboratorio di Meccanica e Termodinamica

Modulo 2 (ROOT) AA 2017/18

Scritto del 07/02/2019

Si scriva la parte rilevante e autoconsistente del codice di una macro di ROOT in cui:

1. Si definisce un grafico (Classe TGraphErrors) atto a rappresentare le 5 coppie di misure (x,y) che dovranno essere opportunamente inserite in forma di arrays:

   valori in x: (1,2,3,4,5)

   valori in y: (3,9,19,33,51)

   errori in y: (0.1,0.4,0.9,1.6,2.5)

2. Si disegna il grafico;

3. Si esegue sul grafico un fit secondo una dipendenza funzionale del tipo:

   $f(x)=A*x^2+B$

   Dove A e B sono parametri liberi del fit.

```
// scritto 07/02/19

#include "TGraphErrors.h"
#include "TF1.h"
#include "TCanvas.h"

const int n{5};
double x_val[n]{1, 2, 3, 4, 5};
double y_val[n]{3, 9, 19, 33, 51};
double y_err[n]{0.1, 0.4, 0.9, 1.6, 2.5};

void macro()
{
    TGraphErrors *graph = new TGraphErrors(n, x_val, y_val, 0, y_err);

    TF1 *fit = new TF1("f", "[0] * x ^ 2 + [1]");
    fit->SetParameters(0.5, 0.5);
    graph->Fit(fit);

    TCanvas *c = new TCanvas();
    graph->Draw();
    fit->Draw("SAME");
}


// se non voglio che stampi a schermo i parametri, Fit(fit, "Q")

// perché funziona anche se non inizializzo i parametri?
```

Laboratorio di Meccanica e Termodinamica

Modulo 2 (ROOT) AA 2017/18

Scritto del 08/06/2018

Si scriva la parte rilevante ed autoconsistente del codice di una macro di ROOT in cui:

1. Si definisce un istogramma undimensionale (Classe TH1D) con 1000 bin, in un range da -1. a 1.;
2. In un ciclo lo si riempe con 10^6 occorrenze di una variabile casuale, estratte da una distribuzione gaussiana con media 0 e deviazione standard 1 (attraverso l'opportuno metodo della classe TRandom);
3. Al termine del ciclo si stampano a schermo:
    a. la media e la RMS dell'istogramma, con i loro errori,
    b. il contenuto dei bin di underflow e overflow (occorrenze fuori range). Dire anche se ci si aspetta che questi ultimi (underflow e overflow) siano diversi da 0, motivando la risposta.

```
// scritto 08/06/18

void macro()
{
    TH1D *h1 = new TH1D("histo", "Istogramma unidimensionale", 1000, -1., 1.);

    for (int i{0}; i < 1'000'000; ++i)
    {
        Float_t x = gRandom->Gaus(0., 1.);
        h1->Fill(x);
    }

    cout << "mean = " << h1->GetMean() << " +/- " << h1->GetMeanError() << '\n';
    cout << "dev. standard = " << h1->GetRMS() << " +/- " << h1->GetRMSError() << '\n';
    cout << "undeflow " << h1->GetBinContent(0) << '\n';
    cout << "overflow " << h1->GetBinContent(1001) << '\n';
}
```

Laboratorio di Meccanica e Termodinamica

Modulo 2 (ROOT) AA 2018/19

Scritto del 10/06/2019

Si scriva la parte rilevante e autoconsistente del codice di una macro di ROOT in cui:

1. Si definisce un grafico atto a rappresentare le 7 coppie di misure (x,y) che dovranno essere opportunamente inserite in forma di arrays:

   valori in x: (-0.75,-0.5,-0.25,0.,0.25,0.5,0.75)

   valori in y: (3.1, 0.1 ,-3.2, -0.1, 2.9, 0.1,-3.2)

   errori in y: (0.2,0.1,0.2,0.1,0.2,0.1,0.2)

2. Si disegna il grafico con l'opzione per visualizzare anche le barre di errore;

3. Si esegue sul grafico un fit secondo una dipendenza funzionale del tipo:

   $f(x)=A*\sin(B*x)$

   Dove A e B sono parametri liberi del fit. A tal fine istanziare un oggetto funzione TF1 in modo opportuno.

4. Si stampano a schermo i valori dei due parametri e le relative incertezze, e il valore del $\chi 2$ ridotto.

```
// scritto 10/06/19

#include "TGraphErrors.h"
#include "TF1.h"
using namespace std;

const int n{7};

double x_val[n]{-0.75, -0.5, -0.25, 0., 0.25, 0.5, 0.75};
double y_val[n]{3.1, 0.1, -3.2, -0.1, 2.9, 0.1, -3.2};
double y_err[n]{0.2, 0.1, 0.2, 0.1, 0.2, 0.1, 0.2};

void iris()
{
    TGraphErrors *graph = new TGraphErrors(n, x_val, y_val, 0, y_err);
    graph->Draw();

    TF1 *fit = new TF1("f", "[0]*sin([1]*x)", -0.75, 0.75);
    fit->SetParLimits(0, 1, 3);
    fit->SetParLimits(1, 4, 7);
    graph->Fit(fit, "Q");

    cout << "A = " << fit->GetParameter(0) << " +/- " << fit->GetParError(0) << '\n';
    cout << "B = " << fit->GetParameter(1) << " +/- " << fit->GetParError(1) << '\n';
    cout << "reduced chi-squared = " << fit->GetChisquare() / fit->GetNDF() << '\n';
}
```

Laboratorio di Meccanica e Termodinamica

Modulo 2 (ROOT) AA 2021/22

Scritto del 12/07/2022

Si scriva la parte rilevante ed autoconsistente del codice di una macro di ROOT in cui:

1. Si definisce un grafico (Classe TGraphErrors) atto a rappresentare le 5 coppie di misure (x,y) che dovranno essere opportunamente inserite in forma di arrays:

   valori in x: (1,4,9,16,25)

   valori in y: (10,20,30,40,50)

   errori in x: (0.1,0.4,0.9,1.6,2.5)

   errori in y: (0.1,0.2,0.3,0.4,0.5)

2. Si disegna il grafico, inserendo come titoli degli assi “misure in x” e “misure in y”

3. Si esegue sul grafico un fit secondo una dipendenza funzionale del tipo:

   f(x)=A+B*sqrt(x).

   Dove A e B sono i parametri liberi del fit. Stampare a schermo i valori di A e B esito del fit, con rispettive incertezze, e il $\chi 2$ ridotto.

```
// scritto 12/07/22

const Int_t n{5};
double x_vals[n] = {1, 4, 9, 16, 25};
double y_vals[n] = {10, 20, 30, 40, 50};
double x_err[n] = {0.1, 0.4, 0.9, 1.6, 2.5};
double y_err[n] = {0.1, 0.2, 0.3, 0.4, 0.5};

void dahlia()
{
    TGraphErrors *graph = new TGraphErrors(n, x_vals, y_vals, x_err, y_err);

    TF1 *fit = new TF1("f", "[0] + [1]*sqrt(x)");
    // fit->SetParameters(1, 1);
    graph->Fit(fit, "Q");

    // TF1 *fitCorr = graph->GetFunction("f");

    TCanvas *c = new TCanvas();
    graph->SetTitle(";misure in x;misure in y");
    graph->SetMarkerStyle(8);
    graph->Draw("A, P");
    fit->Draw("SAME");
    // fitCorr->Draw();

    cout << "A = " << fit->GetParameter(0) << " +/- " << fit->GetParError(0) << '\n';
    cout << "B = " << fit->GetParameter(1) << " +/- " << fit->GetParError(1) << '\n';
    cout << "reduced chi-squared = " << fit->GetChisquare() / fit->GetNDF() << '\n';
    // cout << "A (corr) = " << fitCorr->GetParameter(0) << " +/- " << fitCorr->GetParError(0)
<< '\n';
    // cout << "B (corr) = " << fitCorr->GetParameter(1) << " +/- " << fitCorr->GetParError(1)
<< '\n';
}

// qui sembra funzionare anche senza settare i parametri

// viene un chi quadro dell'ordine di 10e-10, è normale?

// N.B. il SetTitle è fatto in modo che il grafico non abbia titolo
// se lo avessi voluto, graph->SetTitle("Grafico 1;misure in x;misure in y");
```

Laboratorio di Meccanica e Termodinamica

Modulo 2 (ROOT) AA 2020/21

Scritto del 23/06/2021

Si scriva la parte rilevante e autoconsistente del codice di una macro di ROOT in cui:

2. Si definisce un grafico (Classe TGraphErrors) atto a rappresentare le 10 coppie di misure (x,y) e l'incertezza sui valori in y che sono, nell'ordine (x,y,dy), salvati nelle tre colonne del file di testo denominato dati.txt come segue:

1	6	0.6
2	17	1.7
3	34	3.4
4	57	5.7
5	86	8.6
6	121	12.1
7	162	16.2
8	209	20.9
9	262	26.2
10	321.	32.1

3. Si disegna il grafico, inserendo le label degli assi;

4. Si esegue sul grafico un fit secondo una dipendenza funzionale del tipo:

   $f(x)=A*x^2+B*x+C$

   Dove A, B e C sono parametri liberi del fit. Si stampi a schermo il valore dei parametri dopo il fit con le relative incertezze, e il $\chi 2$ ridotto.

```
// scritto 23/06/21

void iris()
{
    TGraphErrors *graph = new TGraphErrors("dati.txt", "%lg %lg %lg");
    graph->SetTitle(";misure in x;misure in y");

    TF1 *fit = new TF1("f", "[0]*x^2 + [1]*x + [2]");
    graph->Fit(fit, "Q");

    TCanvas *c = new TCanvas();
    graph->SetMarkerStyle(8);
    graph->Draw("A, P");
    fit->Draw("SAME");

    cout << "A = " << fit->GetParameter(0) << " +/- " << fit->GetParError(0) << '\n';
    cout << "B = " << fit->GetParameter(1) << " +/- " << fit->GetParError(1) << '\n';
    cout << "C = " << fit->GetParameter(2) << " +/- " << fit->GetParError(2) << '\n';
    cout << "reduced chi-square = " << fit->GetChisquare() / fit->GetNDF() << '\n';
}

// errori molto alti sui parametri B e C, e chi quadro ridotto nullo
```

Laboratorio di Meccanica e Termodinamica

Modulo 2 (ROOT) AA 2021/22

Scritto del 23/06/2022

Si scriva la parte rilevante e autoconsistente del codice di una macro di ROOT in cui:

1. Si definiscono due istogrammi undimensionali con 100 bin, in un range da 0. a 5.;
2. In un ciclo si riempono gli istogrammi con 10^6 occorrenze:
    a. di una variabile casuale estratta da una distribuzione gaussiana con media 2.5 e deviazione standard 0.25 (primo istogramma);
    b. di una variabile casuale estratta una distribuzione uniforme fra 0 e 5 (secondo istogramma);
3. Al termine del ciclo si effettua la somma dei due istogrammi, e si stampano a schermo la media e la RMS dell'istogramma risultante con le rispettive incertezze, Il numero di ingressi totali, il numero di underflow e di overflow.

```
// scritto 23/06/22

const Int_t n{100};

void anemone()
{
    TH1F *h1 = new TH1F("h1", "Histogram 1", n, 0., 5.);
    TH1F *h2 = new TH1F("h2", "Histogram 2", n, 0., 5.);

    for (int i{0}; i < 1e6; ++i)
    {
        Float_t x = gRandom->Gaus(2.5, 0.25);
        h1->Fill(x);

        Float_t y = gRandom->Uniform(0., 5.);
        h2->Fill(y);
    }

    TH1F *h3 = new TH1F("h1", "Histogram 3", n, 0., 5.);
    h3->Add(h1, h2);

    cout << "mean = " << h3->GetMean() << " +/- " << h3->GetMeanError() << '\n';
    cout << "st. deviation = " << h3->GetRMS() << " +/- " << h3->GetRMSError() << '\n';
    cout << "number of entries = " << h3->GetEntries() << '\n';
    cout << "underflow = " << h3->GetBinContent(0) << '\n';
    cout << "overflow = " << h3->GetBinContent(n + 1) << '\n';
}

// non so se sia giusto che vengano 0 undeflow e 0 overflow, ma aumentando
// la dev. st. di h1 a 0.75 vengono 6e3 over e underflow

// è giusto inizializzare così h3?
```

Laboratorio di Meccanica e Termodinamica

Modulo 2 (ROOT) AA 2020/21

Scritto del 27/01/2022

Si scriva la parte rilevante ed autoconsistente del codice di una macro di ROOT in cui:

1. Si definiscono tre istogrammi unidimensionali con 100 bin, in un range da 0. a 10.;
2. In un ciclo si riempiono gli istogrammi con 10^6 occorrenze:
    a. di una variabile casuale estratta da una distribuzione gaussiana con media 5. e deviazione standard 0.5 (primo istogramma);
    b. di una variabile casuale estratta una distribuzione uniforme fra 0 e 10 (secondo istogramma);
    c. di una variabile casuale estratta una distribuzione esponenziale decrescente con media 1 (terzo istogramma);
3. Al termine del ciclo si effettua la somma dei tre istogrammi, e si stampano a schermo la media e la RMS dell'istogramma risultante con le rispettive incertezze, Il numero di ingressi totali, il numero di underflow e di overflow, il massimo contenuto dei bin (maximum bin content) e l'indice del bin corrispondente al massimo contenuto.

```
// scritto 27/01/2022

const Int_t n{100};

void camellia()
{
    gRandom->SetSeed(69);
    TH1F *h1 = new TH1F("histo1", "Istogramma 1", n, 0., 10);
    TH1F *h2 = new TH1F("histo2", "Istogramma 2", n, 0., 10);
    TH1F *h3 = new TH1F("histo3", "Istogramma 3", n, 0., 10);

    for (int i{0}; i < 1'000'000; ++i)
    {
        Float_t x1 = gRandom->Gaus(5., 0.5);
        h1->Fill(x1);
        Float_t x2 = gRandom->Uniform(0., 10.);
        h2->Fill(x2);
        Float_t x3 = gRandom->Exp(1);
        h3->Fill(x3);
    }

    TH1F *hSum = new TH1F(*h1);
    hSum->Add(h1, h2, 1, 1);
    hSum->Add(h3, 1);
    hSum->Draw();

    cout << "mean = " << hSum->GetMean() << " +/- " << hSum->GetMeanError() << '\n';
    cout << "st. dev. = " << hSum->GetRMS() << " +/- " << hSum->GetRMSError() << '\n';
    cout << "total entries = " << hSum->GetEntries() << '\n';
    cout << "underflow = " << hSum->GetBinContent(0) << '\n';
    cout << "overflow = " << hSum->GetBinContent(n + 1) << '\n';
    cout << "maximum bin content = " << hSum->GetBinContent(hSum->GetMaximumBin()) << "
entries, it's the " << hSum->GetMaximumBin() << "th bin\n";
}

// se disegno hSum, il titolo e il nome saranno quelli dell'istogramma 1

// !! SetSeed deve essere dentro la macro, altrimenti non funziona
```