
LIP
LABORATÓRIO DE INSTRUMENTAÇÃO
E FÍSICA EXPERIMENTAL DE PARTÍCULAS
partículas e tecnologia
[ROOT Intermediate Tutorial]

Tutorial instructions

- All the instructions needed to complete this tutorial can be found in the indico page file: ROOT_Intermediate_Tutorial_2023.pdf

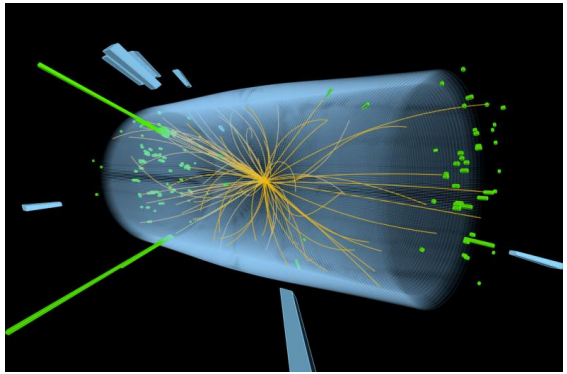
- Once inside the LIP machines, access the input file in folder /home/share/ and name zjet_unrec.root. Alternatively, download the input file with this command:

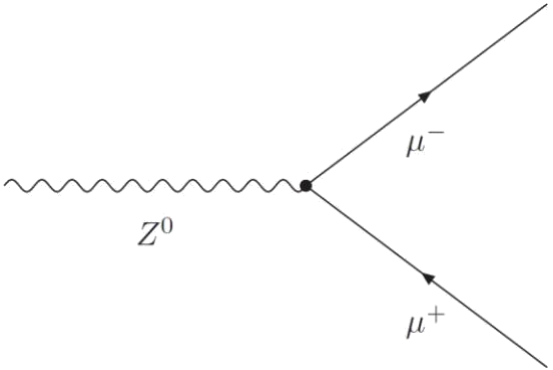
wget "https://indico.lip.pt/event/1226/contributions/4222/attachments/3500/5443/zjet_unrec.root"

Input file

- The file contains simulated events for the decay of the Z-boson
- In this exercise, we are interested in the decay:

$$Z \rightarrow \mu^+\mu^-$$





Input file

- The final objective of this tutorial is for you to draw the invariant mass spectrum of the resulting muon pairs

$$IM = \sqrt{E_T^2 - (p_{x_T}^2 + p_{y_T}^2 + p_{z_T}^2)}$$

Checking the input file

- Once you have the file downloaded, you can see its contents using the following commands (just as in the first tutorial):
    - .ls
    - Tree->Print()
    - Tree->Scan()

Intermediate tutorial

- Create a new file (with whatever name you want) with a function with the same name as the file (so that you can run it as a standard ROOT macro)
- Declare the file object that we are going use to access our input file
- Declare the tree object as in previous tutorials
- Declare a histogram object, with the limits, number of bins, and title you think best for the Z mass

Intermediate tutorial

```
5  void ROOT_Intermediate_part1(){
6
7      // read input file
8      TFile *fInput = new TFile("zjet_unrec.root", "read");
9
10     // load input tree
11     TTree *t = (TTree*) fInput->Get("Tdata");
12
13     // declare TH1 object for Z boson mass
14     TH1D *h = new TH1D("Zmass", "Z boson mass", 100, 0., 150);
15
```

Intermediate tutorial

- Declare the variables that we will use to access the particles of each event, saved in the tree (as done in the first ROOT tutorial)
- Link these variables to the tree with the SetBranchAddress function

Intermediate tutorial

```
12
13      // declare the variables to assess each event
14      vector<int> *id = 0;
15      vector<double> *m = 0;
16      vector<double> *px = 0;
17      vector<double> *py = 0;
18      vector<double> *pz = 0;
19      vector<double> *E = 0;
20
21      // link these variables to the tree
22      t->SetBranchAddress("id", &id);
23      t->SetBranchAddress("px", &px);
24      t->SetBranchAddress("py", &py);
25      t->SetBranchAddress("pz", &pz);
26      t->SetBranchAddress("m", &m);
27      t->SetBranchAddress("En", &E);
```

Intermediate tutorial

- Loop over the entries in tree with a for loop, for example (to get the number of entries in the tree: t–>GetEntries() and to get the values of entry i loaded on the linked variables make t–>GetEntry(i))
- Loop over the particles in each event
- When you find the muons (there should be exactly two per event), calculate their invariant mass and fill a histogram with it

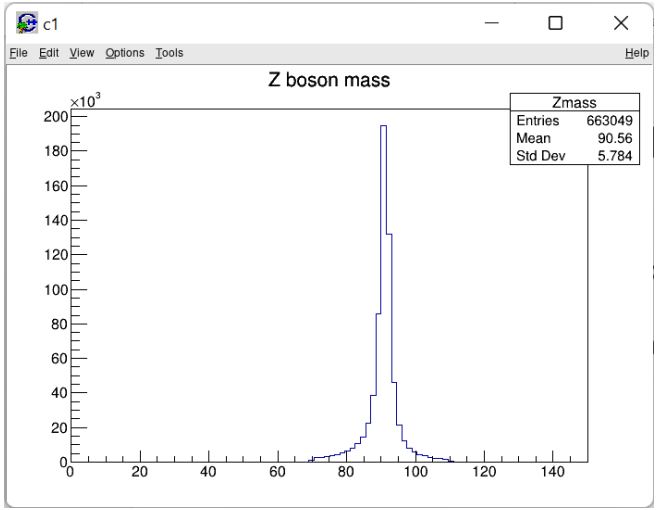
Intermediate tutorial

$$IM = \sqrt{E_T^2 - (p_{x_T}^2 + p_{y_T}^2 + p_{z_T}^2)}$$

```
55      int NEvents = t->GetEntries();
56      for(int i = 0; i <= NEvents; i++){
57
58          t->GetEntry(i);
59
60          // loop through every particle and find muon and antimuon
61          int i_mu_p = 0, i_mu_n = 0;
62          int flag_mu_p = 0, flag_mu_n = 0;
63
64          for(int iObj = 0; iObj <= (*id).size(); iObj++){
65              //muon
66              if((*id)[iObj]==13){
67                  i_mu_p = iObj;
68                  flag_mu_p++;
69              }
70              //antimuon
71              if((*id)[iObj]==-13){
72                  i_mu_n = iObj;
73                  flag_mu_n++;
74              }
75
76          }
77
78          // check if we found both and calculate invariant mass
79          if(flag_mu_p && flag_mu_n){
80
81              // calculate invariant mass
82              double Et = (*E)[i_mu_p] + (*E)[i_mu_n];
83              double pxt = (*px)[i_mu_p] + (*px)[i_mu_n];
84              double pyt = (*py)[i_mu_p] + (*py)[i_mu_n];
85              double pzt = (*pz)[i_mu_p] + (*pz)[i_mu_n];
86
87              double invar_mass = sqrt(Et*Et - (pxt*pxt + pyt*pyt + pzt*pzt));
```

Intermediate tutorial

- Verify that the spectrum makes sense and then save the events that pass your selection in a new tree, in a new file. You can choose what variables you want to save but as a starting point the four vector (momentum plus energy) and the mass should be enough.



Intermediate tutorial

- To do this you should declare a file, more or less in the same place where you had declared the input one and the same for the tree
- You must declare new output variables (i.e. px, etc.) and link them to the new tree with the Branch function
- Every time an event passes you should fill the output variables with the values you want to save and then use t–>Fill(), in order to save them to your output tree

Intermediate tutorial

```
34      // create output file
35      TFile *fOutput = new TFile("zjet_filtered.root", "recreate");
36
37      // create output tree
38      TTree *tOutput = new TTree("Tdata_output", "Tdata_output");
39
40      // declare the variables for output tree
41      double m_output = 0;
42      double px_output = 0;
43      double py_output = 0;
44      double pz_output = 0;
45      double E_output = 0;
46
47      // link these variables to the tree
48      tOutput->Branch("px", &px_output);
49      tOutput->Branch("py", &py_output);
50      tOutput->Branch("pz", &pz_output);
51      tOutput->Branch("m", &m_output);
52      tOutput->Branch("En", &E_output);
```

```
90          // save event to new tree
91          E_output = Et;
92          px_output = pxt;
93          py_output = pyt;
94          pz_output = pzt;
95          m_output = invar_mass;
96          tOutput->Fill();
97
```

```
107     // save new tree to file
108     tOutput->Write();
109     fOutput->Close();
```

Intermediate tutorial

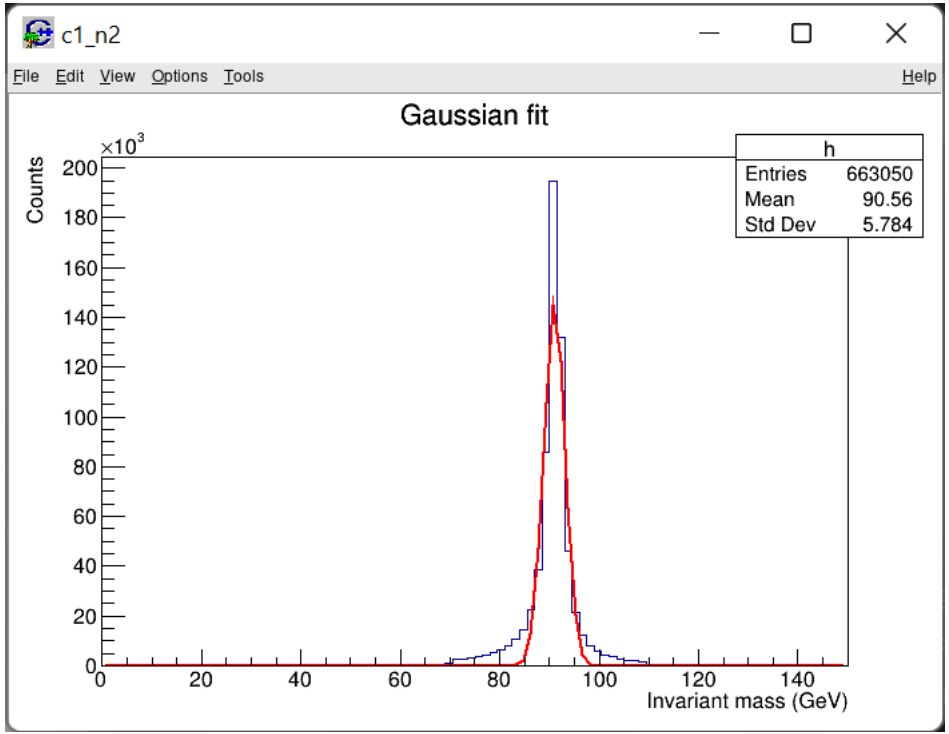
- Write a new ROOT macro that opens your data file.

- Draw the *Z* boson mass peak and fit it using a Gaussian function. The fitted value of the Gaussian's mean should approximately correspond to the *Z* boson measured mass. How good is the agreement?

Intermediate tutorial

```
12  void ROOT_intermediate_part2(){
13
14      // read input file
15      TFile *fInput = new TFile("zjet_filtered.root", "read");
16
17      // read tree
18      TTree *t = (TTree*)fInput->Get("Tdata_output");
19
20      // declare variables to read tree
21      double invar_mass = 0.;
22      t->SetBranchAddress("m", &invar_mass);
23
24      // TH1 object to see Z boson's mass spectrum
25      TH1D *h = new TH1D("h", "Z boson's mass spectrum; Invariant mass (GeV); Counts", 100, 0., 150.);
27      // go through tree
28      int NEvents = t->GetEntries();
29      for(int i = 0; i <= NEvents; i++){
30          t->GetEntry(i);
31          h->Fill(invar_mass);
32      }
33
38      // fit gaussian
39      new TCanvas();
40      h->SetTitle("Gaussian fit");
41      h->Fit("gaus");
42      h->DrawClone();
```

Intermediate tutorial



Intermediate tutorial

- In this case, because we are not taking into account detector resolution effects, the *Z* boson mass peak should be better described by a Breit-Wigner function. This function is not predefined in ROOT, therefore you need to define it yourself. Once you have defined your own TF1 function you can use it to fit the mass peak as before. Does your estimation the *Z* boson mass gets closer to the measured value? What about the goodness of fit, is it improved?

$$f(E) = \frac{k}{\left(E^2 - M^2\right)^2 + M^2\Gamma^2}$$

$$k = \frac{2\sqrt{2}M\Gamma\gamma}{\pi\sqrt{M^2+\gamma}} \qquad \text{with} \qquad \gamma = \sqrt{M^2\left(M^2+\Gamma^2\right)}\,.$$

Intermediate tutorial

```
3   Double_t fitFunc(Double_t* x, Double_t* par)
4   {
5     Double_t arg1 = 14.0/22.0; // 2 over pi
6     Double_t arg2 = par[1]*par[1]*par[2]*par[2]; //Gamma=par[1]  M=par[2]
7     Double_t arg3 = ((x[0]*x[0]) - (par[2]*par[2]))*((x[0]*x[0]) - (par[2]*par[2]));
8     Double_t arg4 = x[0]*x[0]*x[0]*x[0]*((par[1]*par[1])/(par[2]*par[2]));
9     return par[0]*arg1*arg2/(arg3 + arg4);
10  }
```

```
44      // fit Breit-Wigner function
45      new TCanvas();
46      h->SetTitle("Breit-Wigner fit");
47      TF1 *func = new TF1("fitFunc",fitFunc,50,140,3);
48      func->SetParameter(0,1.0);
49      func->SetParameter(2,5.0);
50      func->SetParameter(1,95.0);
51      h->Fit("fitFunc");
52      h->DrawClone();
```

$$f(E) = \frac{k}{\left(E^2 - M^2\right)^2 + M^2\Gamma^2}$$

$$k = \frac{2\sqrt{2}M\Gamma\gamma}{\pi\sqrt{M^2+\gamma}} \qquad \text{with} \qquad \gamma = \sqrt{M^2\left(M^2+\Gamma^2\right)}\,.$$

Intermediate tutorial