4-й семестр, lesson 6

Probability distribution functions and minimisation

- Repeat examples with fits from Lesson 5.
- Why we use TTree
- https://root.cern.ch/root/htmldoc/guides/users-guide/ROOTUsersGuideA4.pdf
- Simple variables and vectors
- Creation of Tree
- Looking for Tree content with TBrowser

Resonances in ($\pi^+ \pi^- \pi^0$) system

- Exercises:
- 1) set link for input Ntuple: ln -s /nfs/lfi.mipt.su/data/nikola/ves/run42/ntbeam_cher_r17_1_v15.root ntbeam_cher_r17_1_v15.root
- 2) copy file scp –p /nfs/lfi.mipt.su/data/nikola/ves/run42/example_5_edited.C . Look for this file and run it in root : .X example_5_edited.C ; histogram h_002 is created and wriyyen to output file ntuple.root
- Try to fit the distribution in h_002 by a Gauss function in mass range (0.65, 0.90) GeV with file from /nfs/lfi.mipt.su/data/nikola/ves/run42/example_6_edited.C
- needed edition: please set the starting values for 3 parameters in the Gauss

Exercises cont.

- Please compare the fitted curve with histogram.
- 3) In order to improve agreement between data and fit result, let's include a Background term (linear function). An example available in /nfs/lfi.mipt.su/data/nikola/ves/run42/example_7_edited.C
- Again, needed a starting point for 3+2 parameters (which might be very approximate)
- Run the fit and look for result. Sttill the χ^2 is bad, but the agreement between the data and the fit result is significantly better.

About Trees

- In case you want to store large quantities of same-class objects, ROOT has designed the TTree class for this purpose. A TTree can hold all kind of data, such as objects or arrays in addition to all the simple types.
- When using a TTree, we fill its branch buffers with leaf data, and the buffers are written to disk when it is full. Each object is not written individually, but rather collected and written a bunch at a time.
- The TTree takes advantage of compression and will produce a much smaller file than if the objects were written individually.
- The TTree is also used to optimize the data access. Each branch can be used independently from any other branch.
- File which contains Tree(s) is usually called NTUPLE.

Variables in Tree

- It can be simple variables or arrays with fixed length, p.ex.
- Int RunNumber;
- Int EventNumber;
- Double VertexXYZ[3];
- Or it can be vectors of fixed or variable length:
- TVector3 Vertex; // from class TVector3
- TlorentzVector Electron; // with components px, py, pz, E
- vector <double> * Px; // it is used if number of tracks
- vector <double> * Py; // in a vertex is not fixed
- vector <double> * Pz; //

Operations with Vectors

- Needed headers, for example #include <TTree.h>; #include <TLorentzVector.h>; #include <Vector3.h>;
- You can create and fill TVector3 vtx (vtxx , vtyy , vtzz); or Vector3 * vtx2 = new TVector3(vtxx , vtyy , vtzz);
- Filling of TlorentzVector Electron;
- Electron.SetPxPyPzE(value_px, value_py, value_pz, value_E);
- Get value from TLorenzVector: double px1 = Electro.px();
- Size of variable size vectors: int Ntrack = Px->size();
- Reading from variable size vector: double value= Px->at(N); //N<size
- Caution: an access with « -> » operator instead of « . » is needed if a pointer to object is declared

Creation of Tree

- 0) #include statements
- 1) declaration of objects:
- static TTree *t3ev;
- static int nrun_t3;
- static int nevt_t3;
- static double bs_m_t3;
- 2) declaration of Tree and Branches
- t3ev= new TTree("t3ev", " Bs ntuple ");
- t3ev->Branch("nrun_t3", &nrun_t3, "nrun_t3/I");
- t3ev->Branch("nevt_t3", &nevt_t3, "nevt_t3/I");
- t3ev->Branch("bs_m_t3", &bs_m_t3, "bs_m_t3/D");

Creation of Tree

- 3) Filling inside Loop over events:
- nrun_t3 = runNumber;
- nevt_t3 = eventNumber;
- bs_m_t3 = ((*VTX_mass)[j])/1000.;
- t3ev->Fill(); // fill information about event
- 4) After Loop over events:
- t3ev->AutoSave();
- TFile *_filewOut= TFile::Open("./nt_mumu_f0.root", "recreate");
- t3ev->Write();
- _filewOut->Close();

Looking for Tree

- With link to previously used file
- root -l ntbeam_cher_r17_1_v15.root
- TBrowser b
- Go to line below «ROOT files» and click on it
- Click to Tree name (h9024), you see a list of Branches
- You can do simple operations, p.ex. Click to «Neth2g», the program reads this tree and plot the histogram with number of eta->gamma gamma per event (0 or 1).
- You can do the same operation from command line: h9024->Draw("Neth2g");
- The 2nd parameter in Draw permits to apply selection: h9024->Draw("Neth2g" ,"Npi0==0 ") will select events without π^0
- h9024->Draw("Neth2g:Npi0", "", "box") produces 2-dimensional plot with option «box»

Exercise – produce new Tree

- Starting from previous example in /nfs/lfi.mipt.su/data/nikola/ves/run42/example_4_edited.C
- Produce new tree «omega» with 4 branches:
- Itape, Ispln, Ievtn, m(pi+pi-pi0)
- Useful comand : l9024->Print(), it helps to find the type of objects
- Please do not use the identifiers from input Tree for definition of new structure.
- Write new Tree to file.