Topics in Particle Physics and Astroparticles II

Guilherme Soares

Laboratory of Instrumentation and Experimental Particle Physics

17 March 2022

Index

TLorentzVectors

One of the main TObject types that we have are **TLorentzVectors**.

It is a 4-vector for position (X, Y, Z, T) or energy (Px, Py, Pz, E), but there is no actual distinction when defining it (the only thing that matters is that we know what it is).

It can perform operations between 4-vectors automatically, and can give quantities of interest like mass easily (TLorentzVector : :M()).

It cannot be utilized as an automatic variable for plotting within ROOT (There is no TTree : :Draw(TLorentzVector) option, so we need to loop over all events and fill histograms manually.)

Tutorial 2 - TLorentz Vectors and Data Fitting

Available on the usual github page, the template "Template_RooFit.ipynb" reads *Skim4.root*, which is a file containing dimuon data.

Start by checking that there are 4 branches now, with all of them but "*event*" being TLorentzVectors.

Look at a few values for the branch "*event*". Why are the values like that ?

Plot the mass spectrum of dimuon events by utilizing the TLorentzVector : :M() function. Can you recognise any of the peaks ? Compare it to the mass spectrum for the *muonp_4* and *muonn_4* branches.

Check the TLorentzVector class and make use of its functions to create a new TLorentzVector that is the sum of muonP and muonN. Plot this new vector's mass and compare it to the dimuon spectrum.

As a last exercise, you can see that the binning on TH*F histograms doesn't need to be uniform. By giving the binning an array input, you can define the size of each individual bin equal to the value of each entry of the array.

Fitting a histogram with RooFit - Principles

There are several ways of utilizing ROOT to fit data. We will be utilizing RooFit to do so, but feel free to explore other methods, such as TH1's fitting functions, or using TFormula and TGraphs (this is how I did throughout my bachelors).

The template shows a gaussian fit being applied to the J/ψ peak with an associated exponential for the expected background.

RooFit has a plethora of specific classes that transform usual ROOT objects into a format that RooFit can work with. The primary ones are :

- **RooDataHist** / RooDataSet
- **RooRealVar**
- RooFit **Functions**

Since we will be fitting a histogram we will use a RooDataHist object that contains the histogram data to be fitted, and can be created through a TH1F histogram.

Fitting a histogram with RooFit - Creating the Dataset

So we start by creating said histogram.

In order to do this we start by defining our TH1F object with the limits of the peak, and proceed to fill it with the relevant data from our TLorentzVectors.

After this we can create the aforementioned **RooDataHist** object through RooDataHist("name", "title", RooArgList, *TH1F). Here RooArgList is a specific object that contains all the variables of our dataset. In our case we only want the **mass** of the particles.

In order to consider the **mass** as real variable we create the RooRealVar mass object, with RooRealVar("name", "title", minimum, maximum, "units"). Notice that we are creating this variable in a different way to all other RooRealVar objects in the template, as the other variables will be parameters of our fitting models, which require limits **and** initial estimations.

Fitting a histogram with RooFit - Creating the Fitting Model

There are 3 primary items :

- Signal
- Background
- Amount of events that belong to each one (Normalization)

RooFit already has a plethora of functions, such as RooExponential which we will use. However, we need to first define the variable of our data (mass that we already defined), but also define the free parameters of our functions as RooRealVar objects.

The exponential function only has one free parameter, since the normalizing factor is included in the amount of events : **lambda**, the exponential factor.

We define this through RooRealVar("name", "title", estimation, min, max)

As for the background function, we utilize RooExponential("name","title",real_var,parameter_var)

The signal gaussian function is done in a similar way, but we use RooGaussian and now we have 2 variables : **mean**, **sigma**.

Fitting a histogram with RooFit - Fitting the Data

The only thing left is to join our signal and background functions into a single model.

Here we also need the aforementioned amount of events predicted. Hence the last 2 RooRealVar objects n_signal and n_back.

We are now ready to create our final model "model" = RooAddPdf("name", "title", RooArgList(functions), RooArgList(amount of events))

We are now all set, and to fit the model to our data one simply uses

RooAddPdf : :fitTo(RooDataHist) $\rightarrow$ model.fitTo(dh)

The rest of the code is for plotting the different RooFit models, the RooDataHist and a nice legend to accompany it.

Unbinned Fits - Principles

As the name implies we do not bin data.

The advantage against the histogram fitting is that we **do not "compress" data** by binning. The downside is that we cannot perform a traditional fit on a histogram that has infinitely small bins.

We transform it into an optimization problem

We define a likelihood function $L(\vec{x}, \vec{\theta})$ that depends on the parameter space from our data points $\vec{x} = \{x_0, x_1, ..., x_n\}$ and also on the free parameters from our fitting models $\theta = \{\theta_0, \theta_1, ..., \theta_m\}$.

The likelihood function is usually written in the form of $L(\vec{x}, \vec{\theta}) = -log[P(\vec{x}, \vec{\theta})]$, and we then use iterative methods (Minuit2 in ROOT) to obtain the $\vec{\theta}$ that minimizes $\sum_{\text{dataset}} L(\vec{x}, \vec{\theta})$.

Unbinned Fits - Creating the Dataset

The process of fitting the dataset with RooFit remains very similar to when we fit the histogram. However, now we utilize a **RooDataSet** object in order to store our data.

By following the template, you can see that initializing the RooDataSet object is slightly more complicated. In our case, we will do so by first defining an empty RooDataSet, that has a RooRealVar "mass", and we will then fill it one entry at a time. Notice that our method allows us to perform the mass cuts a priori to only include the relevant range.

While you can always just load the whole dataset into RooDataSet, unlike for the TH1F objects, values outside of the specified range for the variable are counted as being on the limit, damaging the quality of the data for the fit, unless posterior cuts are made. Additionally, the computing time is significantly higher.

With the RooDataSet created, the rest of the process is the same, and we can easily copy the code from our previous block in order to perform the fit.

As you can see, plotting a RooDataSet yields a RooDataHist-like plot.

Project

Select a peak other than the J/ψ.

Perform **two different fits** with relevant functions in order to determine the mass of the peaks.

Leave a little text explaining why the functions were chosen (no complex answers needed).

Give an estimate to the invariant mass of the peak with statistical and systematic errors

Perform a yield measurement in transverse momentum ? ? ?