Report CMEPDA

Giulio Cordova and Matilde Carminati

January 10, 2023

1 Introduction

The goal of this project was to develop a tool for fitting the mass of upsilon states and create a plot of the differential cross sections. In particular, the analysis is conducted keeping this article as a guideline, despite the fact that we had to use a different dataset, namely the DoubleMuParked dataset from 2012 in NanoAOD format reduced on muons from the CMS experiment.

The main focus of this project was to perform a data analysis with computationally efficient code and a software organization such that people reviewing or reusing the software could understand it easily. For this reason, we did not complain when the obtained fit results were not optimal. They were good enough to test the library :-)

2 Program functionalities and implementation

In this section a brief description of the program functionalities and its implementation is provided. This is not a documentation page, so there will not be details. For a more accurate description of the code, please refer to the documentation page.

We can summarize the workflow of the program in the following steps:

- set flag and options
- read data (and wisely store it, if not already present locally)
- apply some user-defined cuts (optional)
- create an histogram of the invariant mass
- resolve and fit the upsilon states
- calculate and plot the differential cross sections in p_T

2.1 Flags and Options

A function handling command line arguments as flags or options was necessary. For example, an option [`--mode`] was implemented for deciding whether to just fit the dataset, or to plot the differential cross section in p_T. Another flag was necessary to select the PDF used to perform the fit [`--fitFunction`], or another one to show the output of MINUIT during the MLE fit [`--verbose`]. A comprehensive list of the flags and their functionalities can be seen in the documentation or just by using the flag `--help`.

The acquisition of these parameters from the command line was implemented with the help of the library getopt.

In the file optionParse.C are also defined some functions in order to handle out of boundaries values or incorrect types.

2.2 Data reading (and writing)

We used a CMS open dataset in this analysis. The reading of the online dataset is quite a long task (up to 25 minutes, depending on the internet connection). For this reason, it was decided to store the data in an RDataFrame and save it a root file with a Snapshot().

It was decided to implement a function that checks for the existence of the data.root file in the folder Data. If they are not present, the function creates them.

Before storing the data in the file, it was decided to modify the dataframe in order to discard uninteresting events (to reduce the size of the file) and to define useful variables in the upcoming tasks.

First, single muons were kept that satisfied the kinematics constrains defined in the Equation 1 of the article of reference. The dataset were then filtered for keeping only at least two muons of opposite charge, and a PtEtaPhiMass four-vector was computed as the sum of the two muons four-vector. New columns were defined, such as the pt, rapidity, beta and mass of the dimuon four vector just created. Lastly, a cut around on the invariant mass is performed, to limit the dataset in upsilon region (between 8.5 and 11.5 GeV).

TheRDataFrame container resulted pretty helpful in this task because of its useful properties, such as declarative analysis, multi-threading and other low-level optimisations that allow users to exploit all the resources available on their machines completely transparently.

2.3 Cuts

It is possible to choose custom cuts on the transverse momentum and the rapidity of the dimuon state, by adding the appropriate options while calling the program from the terminal. This is useful for performing a fit with custom cuts. This feature is also used in the differential cross section calculation, because it loops over the fit with different values of p_T. For filtering the dataset, decalrative analysis was used in order to apply the cuts. This is possible because of the just-in-time compilation.

2.4 Spectrum plot

A preliminary histogram of the invariant mass of the dimuon is created by using the method Histo1D of RDataFrame. The histogram is plotted in a canvas, which is then saved in the folder Plots. If the folder Plots does not exist, the program automatically creates it. The saved plot could be useful for checking the consistence of the performed fit. The histogram is then saved in a TH1 container since it is easier to store it in a RooDataHist.

2.5 Fit plot

For fitting the histogram created in the previous step, we used the package RooFit, a toolkit for modeling the expected distribution of events by performing an unbinned maximum likelihood fit. The advantage of this library is to define a model fit PDF with different components whose fit results are easily separately accessible. In particular we found useful the result that indicate how many events are counted under a certain component as we needed for the calculation of the differential cross section. If the fit result does not satisfy certain conditions on the convergence and estimated distance to minimum, the program exit with a fit error code and the result is not displayed. The data and the fitted function are plotted in a canvas that is then saved in the folder Plots. The canvas is shown interactively thanks to a TApplication, if the flag [`--muteCanvas`] is not used.

2.6 Differential cross section

The differential cross section in p_T is calculated with the same binning and parameters (efficiencies, etc.) of the article of reference. The acceptance was considered equal to one, because we did not have the Monte Carlo data for estimating it in each p_T bin. To calculate the differential cross section, we need the number of events, which are the area under each shape of the resonances, for each bin in p_T. In order to get this quantity, we loop over the bin edges, cutting the dataframe according to them, and performing a fit as defined in Section 2.5. The area under each resonance is used to calculate the differential cross section, which is then saved in a structure with its associated uncertainty. This structure is finally used to plot a multigraph of the differential cross sections of the three Υ resonances.

2.7 Coding Style Options

The styling of the code files is formatted and checked using the library clang-format, using the guidelines provided by the ROOT official page.

3 Shared Library Implementation

This project was built with CMake, "an open-source, cross-platform family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice." We found this platform really useful for keeping an organised project and implementing our shared library *YCrossFit*, combining our code with the ROOT package.

4 Testing

CMake is also useful for defining unit testing for our library, using the command `ctest`, that automatically build targets for the tests. We chose to implement tests only on the functions created by us, instead of testing the functionalities of the ROOT library.

Test0 This test handles the reading of the command arguments and flags. In this test one define some variables, call the processArgs() and sees if the definition stands, then one modify the arguments and check if the options are evolved according to the made changes

Test1 Here is tested the online reading of the data and the behavior in case the Data folder or the Data file is missing. In the test, the folder Data is deleted and the function df_set() is called. This function should handle the creation of the folder Data and the downloading and saving of the data. Once finished this first step, we check if the Data folder exists and if it contains the file data.root. Next, we keep the folder Data and we eliminate the file data.root. The function df_set() is called again and after it finished, one check if the data is successfully recreated.

Test2 In this test the fit results are controlled. First off, one defines a model with a similar shape to the one expected and calls the function fitRoo() passing this model as an argument instead of the real data. The test checks if the fit converged by looking at the fitStatus and also check if the returned parameters are inside 5 sigma of the initial value.

Test3 This one tests the function SavePlot() which handles the saving of a canvas with a specific filename. If the folder Plots does not exist, it creates it.

Test4 This test is useful to check if the printing of the custom cuts on the canvas work. It compares the strings returned by the function formatYString() or formatPtString() with the expected ones.

5 Documentation

In order to write the documentation, we relied on the Doxygen tool. The special comments for the documentation are written only in the header files, for avoiding a difficult read of the source code. In the header file, each function is declared and its functionality is described in this special code above the function declaration. The source code is also well commented in order to guarantee a deeper understanding of the code.

A doxyfile was written for generating the documentation in html format starting from the special comments in the source code. For a better understanding, a website mainpage was also created in order to explain the project with examples on the functionalities. It is more or less an "hands-on" guide for the library usage.

A continuous integration for the documentation is implemented by using the GitHub action doxygenize developed by langroodi. The output files are saved in a different branch than the code for the correct creation of the gh-page for the documentation.

6 Future Developments

Our code is not perfect. We tried implementing a library with as many useful functionalities as possible as we could in these two months of work. There are many possible ways of improving the code. Some are summarized in this list below:

- Implementing environmental variables for accessing data and plots. Our code download the data again if they are not placed in the right relative path.
- Implementing some concurrential/parallel programming for some functions, e.g. the multiple fits over the bins for the computation of the differential cross section
- Implementing an installation, making possible running the program from anywhere in the terminal (including downloading the requirements)
- Implementation of more PDFs to fit the data (crystall ball, etc...)