Report CMEPDA

Giulio Cordova Matilde Carminati

January 12, 2023

Abstract

A software able to cut a dataframe and fit lineshapes with customizable parameters was written in order to perform the analysis of the differential cross sections as a function of transverse momentum p_T for the production of Υ(nS) (n = 1, 2, 3) states decaying into a pair of muons. Data corresponding to an integrated luminosity of $11.6\,\mathrm{fb}^{-1}$ in pp collisions at $\sqrt{s} = 8\,\mathrm{TeV}$ were collected with the CMS detector at the LHC and acquired by us via the CMS Open Data portal. This analysis selects events with dimuon rapidity $|y| < 1.2$ and dimuon invariant mass in the range $8.5 < p_T < 11.5\,\mathrm{GeV}$, although these values are customizable in the software.

1 Introduction

Hadronic production of S-wave bb mesons has been extensively studied for many years. At the CERN LHC, the CMS, ATLAS and LHCb Collaborations have published results on Υ(nS) (n = 1, 2, 3) production cross sections times dimuon branching fractions in pp collisions at $\sqrt{(s)} = 7$ TeV as a function of the Υ transverse momentum, rapidity and polarization.

In this report we present a calculation of the differential production cross sections of the three lowest-mass Υ(nS) states in pp collisions at $\sqrt{s} = 8\,\mathrm{TeV}$ up to $p_T = 100\,\mathrm{GeV}$ using a software created by the authors. We plot the p_T dependence of the Υ(nS) differential cross section times the branching fraction to $\mu^+\mu^-$ using the DoubleMuParked dataset from 2012 in NanoAOD format reduced on muons[4] from the CMS experiment, corresponding to an integrated luminosity of $11.6\,\mathrm{fb}^{-1}$.

The main focus in 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, some results may not result optimal.

2 Data Cleaning and Formatting

Event selection is done with a series of trigger on the presence of at least two high-energy muons in the event. More information about the Data taking and High-Level Trigger (HLT) can be found on the CERN open data platform [1][2].

This trigger selected dimuons in the invariant mass region spanning more than three orders of magnitude, from a few hundred MeV/c^2 to a few hundred GeV/c^2. The following kinematic requirements are also imposed to ensure accurate muon detection efficiency evaluation:

$$\begin{aligned} p_T(\mu) &> 3\,\mathrm{GeV} && \text{for } 1.4 < |\eta(\mu)| < 1.6, \\ p_T(\mu) &> 3.5\,\mathrm{GeV} && \text{for } 1.2 < |\eta(\mu)| < 1.4, \\ p_T(\mu) &> 4.5\,\mathrm{GeV} && \text{for } |\eta(\mu)| < 1.2. \end{aligned} \tag{1}$$

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-vectors. It resulted useful to define new columns in our dataframe, such as the pt, rapidity and mass of the dimuon four vector just created. This actionwas actually necessary in order to cut the dataframe while performing the fits for calculating the differnetial cross section. In fact, a cut around on the invariant mass is at once 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.

3 Differential Cross Section measurement methodology

The $\Upsilon(nS)$ differential cross section times dimuon branching fraction, integrated over either of the two $|y|$ ranges and in a given p_T bin of width Δp_T, is

$$\left.\frac{\mathrm{d}\sigma(\mathrm{pp}\to\Upsilon(n\,\mathrm{S}))}{\mathrm{d}p_{\mathrm{T}}}\right|_{|y|\text{ range}} \mathscr{B}\left(\Upsilon(n\,\mathrm{S})\to\mu^+\mu^-\right) = \frac{N^{\mathrm{fit}}_{\Upsilon(n\,\mathrm{S})}\left(p_{\mathrm{T}}\right)}{L\Delta p_{\mathrm{T}}\varepsilon_{\mu\mu}\left(p_{\mathrm{T}}\right)\mathscr{A}\left(p_{\mathrm{T}}\right)\varepsilon_{\mathrm{sg}}\varepsilon_{\mathrm{vp}}} \tag{2}$$

where $N^{fit}_{\Upsilon(nS)}$ is the fitted number of Υ(nS) events from the dimuon invariant mass distribution in a p_T bin for the selected $|y|$ range, $\varepsilon_{\mu\mu}$ is the dimuon efficiency, L is the integrated luminosity, $\mathscr{A}$ is the polarization-corrected acceptance, ε_{sg} is the efficiency of the seagull selection, and ε_{vp} is the efficiency of the dimuon vertex χ^2 probability requirement.

For this reason, we decided to implement in our software functionalities able to

- cut the dataframe on the trasverse momentum p_T and rapidity y;
- perform an extended maximum-likelihood estimation (MLE) fit using RooFit to determine the number of signal events associated with each normalized signal PDF.

For filtering the dataset, decalrative analysis was used in order to apply the cuts. This is possible because of the just-in-time compilation property of RDataFrame.

3.1 Fitting

The Υ(nS) lineshape for a given p_T bin is expressed by a probability density function (PDF) for the signal dimuon mass $M_{\mu\mu}$. For this project, it was decided to use a gaussian distribution

$$F(M_{\mu\mu};m,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(M_{\mu\mu}-m)^2}{\sigma^2}\right) \tag{3}$$

where m is the mass of the resonance and σ its width.

The background was modeled with a second degree polynomial, leading to define a model for the complete unnormalized PDF:

$$P(M_{\mu\mu};N_n,m_n,\sigma_n,a_0,a_1,a_2) = \sum_{n=1}^{3}\frac{N_n}{\sqrt{2\pi}\sigma_n}\exp\left(-\frac{(M_{\mu\mu}-m_n)^2}{\sigma_n^2}\right) + (a_0 + a_1 M_{\mu\mu} + a_2 M_{\mu\mu}^2)N_{back} \tag{4}$$

where the index n refers to the three Υ(nS) resonances and N_n to the total number of events yield under each curve. We measure them by performing an extended maximum-likelihood estimation (MLE) fit using RooFit to determine the number of signal events associated with each normalized signal PDF.

The data are binned in p_T, with bin edges at 2 GeV intervals between 12 and 40 GeV, then wider bins with edges at 43, 46, 50, 55, 60, 70, and 100 GeV. The cut to apply on the absolute value of rapidity $|y|$ is customizable, although a default value is set in our software at $|y| < 1.2$ for each p_T bin.

For the initialisation of the signal fit parameters, such as the Υ masses and widths, we take the values from the Particle Data Group. Furthermore, the initial parameters for the background function are chosen by computing a slope using the extreme points of the histogram, while the offset is set accordingly to the slope. The curvature is initialized to zero.

The plots in Fig. 1 show two examples of fitting the dimuon invariant mass distribution. The lower plots show the pull, $(N_{data} - N_{fit})/\sigma_{data}$, in each dimuon mass bin, where N_{data} is the observed number of events in the bin, N_{fit} is the integral of the fitted signal and background function in that bin, and the uncertainty σ data is the Poisson statistical uncertainty. As example, the results returned by RooFit for the fit in the two p_T bins, in comparison with to the initialized parameters, are also reported in the Table 1.

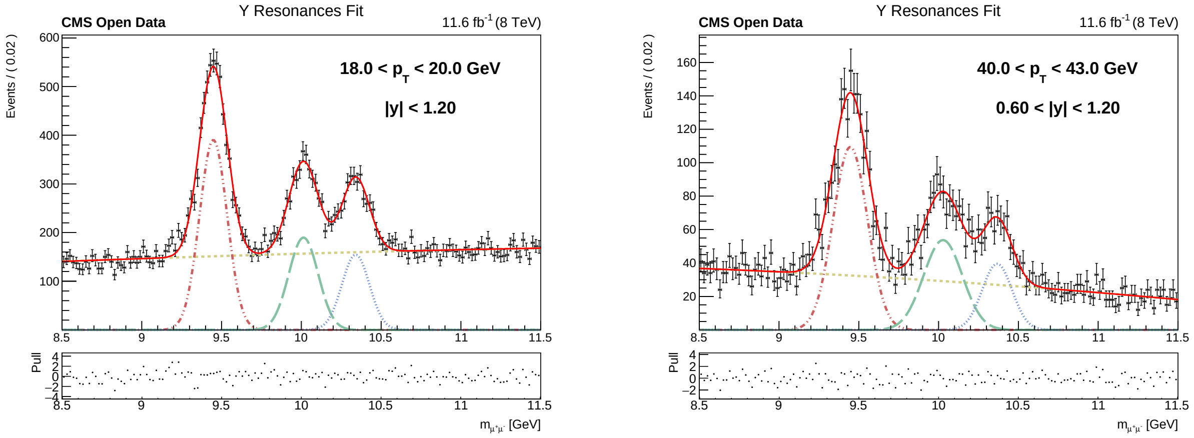

Figure 1: Results of the fits to the dimuon invariant mass distribution for events in two bins.The solid line is the result of the full fit. The dash-dotted line is the Υ(1S) signal fit, the long-dashed line is the Υ(2S) signal fit, and the dotted line is the Υ(3S) signal fit. The short-dashed line is the background contribution. The lower plots show the pull for each mass bin.

Floating Parameter	InitialValue	FinalValue ± Error
a0	50.473	$(1.0510 \pm 0.314) \cdot 10^3$
a1	10.403	5.3018 ± 13.4
a2	0.0	-2.1123 ± 0.796
mass1	9.45	$(9448.5 \pm 2.10) \cdot 10^{-3}$
mass2	10.01	$(10013 \pm 4.31) \cdot 10^{-3}$
mass3	10.35	$(10340 \pm 5.03) \cdot 10^{-3}$
nback	$10.063 \cdot 10^3$	$(23.372 \pm 0.206) \cdot 10^3$
nsig1	$13.208 \cdot 10^3$	4223.6 ± 99.6
nsig2	$9.7489 \cdot 10^3$	2189.4 ± 96.21
nsig3	$6.9186 \cdot 10^3$	1663.5 ± 89.3
sigma1	$54.0 \cdot 10^{-3}$	$(86.384 \pm 2.22) \cdot 10^{-3}$
sigma2	$32.0 \cdot 10^{-3}$	$(92.120 \pm 4.79) \cdot 10^{-3}$
sigma3	$20.0 \cdot 10^{-3}$	$(85.961 \pm 5.26) \cdot 10^{-3}$

(a) $18 < p_T < 20\,\text{GeV}$ and $|y| < 1.2$

Floating Parameter	InitialValue	FinalValue ± Error
a0	109.54	103.14 ± 4.65
a1	-8.0537	-8.5644 ± 0.376
a2	0.0	$(60.104e-02 \pm 31.0) \cdot 10^{-3}$
mass1	9.4500	$(9446.9 \pm 4.19) \cdot 10^{-3}$
mass2	10.010	$(10027 \pm 11.0) \cdot 10^{-3}$
mass3	10.350	$(10367 \pm 12.4) \cdot 10^{-3}$
nback	$2.2605 \cdot 10^3$	4306.4 ± 94.9
nsig1	$2.9669 \cdot 10^3$	1479.0 ± 55.8
nsig2	$2.1898 \cdot 10^3$	816.15 ± 63.8
nsig3	$1.5541 \cdot 10^3$	462.73 ± 58.4
sigma1	$54.0 \cdot 10^{-3}$	$(107.87 \pm 4.17) \cdot 10^{-3}$
sigma2	$32.0 \cdot 10^{-3}$	$(121.26 \pm 10.7) \cdot 10^{-3}$
sigma3	$20.0 \cdot 10^{-3}$	$(93.459 \pm 12.3) \cdot 10^{-3}$

(b) $40 < p_T < 43\,\text{GeV}$ and $0.6 < |y| < 1.2$

Table 1: Results returned by RooFit for the fit in the two bins, compared with to the initialized parameters. These parameters relies on the model PDF defined in Eq. (4). The initial value of the mass and the sigma are the same for every fit in each binning, while the initial parameters relative to the background polynomial depend on the p_t binning. The $Nsig_n$ (n=1,2,3) parameters are extracted in each fit for computing the differential cross sections.

3.2 Efficiency factors and acceptance

Since we did not have the Monte Carlo data to compute the efficiencies and the acceptance, we have taken the values reported in the article of reference[3]:

- $\varepsilon_{\mu\mu} = 0.75$
- $\varepsilon_{sg} = 0.5$
- $\varepsilon_{vp} = 0.99$

These values represent an average of the efficiencies. In the reality, this ones vary with the binning in p_T.

The values of acceptance for each p_T bin and Υ state can be found in the Tables A.9, A.12, A.15 of the appendix A of the article of reference [3].

4 Results

The measured Υ(nS) differential cross sections versus p_T are shown in Fig. 2 over the full rapidity range $|y| < 1.2$. The vertical bars on the points in Fig. 2 show the statistical uncertainties.

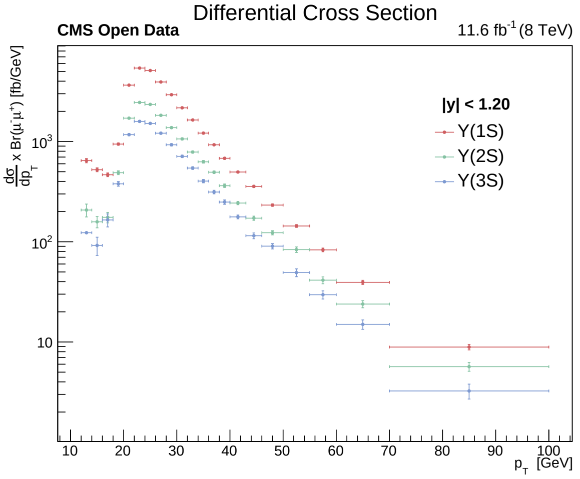

Figure 2: The Υ(nS) differential p_T cross sections times dimuon branching fractions for $|y| < 1.2$. The vertical bars show the total uncertainty, excluding the systematic uncertainty in the integrated luminosity. The horizontal bars show the bin widths.

Looking at Figure 2, we observe two different behaviour, at low and high energy. At high energy, the trend looks like an exponential, while at the low scale is not well-defined. Comparing our result with the CMS one[3], it is visible the effect of the two different behaviour does not exist. One of the possible explanation may rely on the fact that we have used average efficiency factors for all the bins. We also notice an evident difference in the y axis scale of a factor 10^2 for the bins at high energy.

A possible further development could be the fitting of the differential cross sections with exponential and power-law parametrisation such as

$$\left.\frac{d\sigma(\text{pp} \to \Upsilon(n\,\text{S}))}{dp_\text{T}}\right|_{|y|\ \text{range}} \mathscr{B}\left(\Upsilon(n\,\text{S}) \to \mu^+\mu^-\right) = \frac{A}{C + \left(\frac{p_\text{T}}{p_0}\right)^{\alpha}}, \tag{5}$$

in order to investigate the NRQCD + NLO predictions.

5 Software description

5.1 Functionalities of the YCrossFit library

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

5.1.1 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.

5.1.2 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 Equation (1). 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.

5.1.3 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.

5.1.4 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.

5.1.5 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.

5.1.6 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. 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 5.1.5. 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.

5.2 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.

5.3 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.

5.4 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.

5.5 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.

5.6 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...)
- Implementation of the calculation of efficiencies and acceptance with a Monte Carlo

References

[1] CMS Collaboration. */DoubleMuParked/Run2012B-22Jan2013-v1/AOD*. 2017. DOI: 10.7483/OPENDATA.CMS.YLIC.86ZZ. URL: http://opendata.cern.ch/record/6004.

[2] CMS Collaboration. */DoubleMuParked/Run2012C-22Jan2013-v1/AOD*. 2017. DOI: 10.7483/OPENDATA.CMS.M5AD.Y3V3. URL: http://opendata.cern.ch/record/6030.

[3] CMS Collaboration. “Measurements of the Upsilon(1S), Upsilon(2S), and Upsilon(3S) differential cross sections in pp collisions at sqrt(s) = 7 TeV”. In: *Physics Letters B* 749 (Oct. 2015), pp. 14–34. DOI: 10.1016/j.physletb.2015.07.037. URL: https://doi.org/10.1016%2Fj.physletb.2015.07.037.

[4] Stefan Wunsch. *DoubleMuParked dataset from 2012 in NanoAOD format reduced on muons*. 2019. DOI: 10.7483/OPENDATA.CMS.LVG5.QT81. URL: http://opendata.cern.ch/record/12341.