Department of Physics and Astronomy
University of Heidelberg

Bachelor Thesis in Physics
submitted by

Sebastian Hornung

born in Speyer (Germany)

2014

New criteria for distinguishing hadrons containing beauty from hadrons with charm in proton-proton collisions with $\sqrt{s} = 7\,\text{TeV}$ at ALICE

This Bachelor thesis has been carried out by Sebastian Hornung at the
Physikalisches Institut in Heidelberg
under the supervision of
Dr. rer. nat., Priv.-Doz. Silvia Masciocchi

Zusammenfassung

Im Zuge dieser Arbeit wurden Kriterien zur Unterscheidung von Hadronen, welche Beauty- beziehungsweise Charm-Quarks enthalten, mit Hilfe von Monte Carlo Simulationen von pp Kollisionen mit $\sqrt{s} = 7\,\mathrm{TeV}$ untersucht. Diese Kriterien waren die invariante Masse der rekonstruierten Beauty-Hadronen, ihre Lebensdauer und die Anzahl der Teilchen, die zu den entsprechenden rekonstruierten Sekundärvertices beitragen, sowie der Stoßparameter der Tochterelektronen. Die Analyse zeigt, dass mit Hilfe der Masse eine reine Auswahl von Hadronen mit Beauty-Quarks mit einer Effizienz von $(13.4 \pm 0.1)\,\%$ erzielt werden kann. Die Lebensdauer kann nicht alleine zur Unterscheidung von Hadronen mit Charm- oder Beauty-Quarks genutzt werden, da Hadronen mit Charm die Auswahl zu stark verunreinigen. Die Auswahl mit Hilfe der anderen Kriterien ist durch Hadronen mit Charm-Quarks weniger stark verunreinigt. Für eine gute Auswahl müssen die drei zuletzt genannten Kriterien kombiniert werden. Ein erster Versuch der Rekonstruktion der Sekundärvertices und die Anwendung der beschriebenen Kriterien zeigte das große Potential dieses Ansatzes.

Abstract

In this thesis criteria to distinguish hadrons with beauty or charm quarks were investigated using Monte Carlo simulations of pp collisions at $\sqrt{s} = 7\,\mathrm{TeV}$. These criteria were the invariant mass of the b hadrons, their lifetime, the impact parameter of the daughter electrons and the number of particles contributing to the reconstructed secondary vertex (prongs). The analysis showed that the mass can be used to create a pure data-set of beauty hadron vertices with an efficiency of $(13.4 \pm 0.1)\,\%$. The lifetime criterion cannot be used alone, because of the high contamination of the selection by hadrons containing charm. The selection with the number of prongs and the impact parameter criteria suffer less from contamination by hadrons containing charm quarks. Therefore the impact parameter, the number of prongs and the lifetime have to be combined to be used in further analyses. A first test of reconstruction and selection showed that the analysed approach is really promising.

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 24. September 2014,

Contents

1. Theoretical background

1.1. The Standard Model of Particle Physics and the Quark-Gluon Plasma

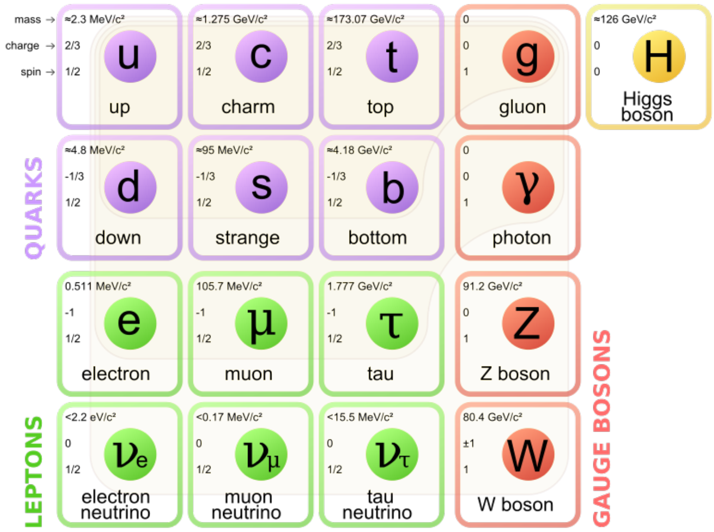


Figure 1.1.: A schematic representation of the Standard Model of Particle Physics containing three generations of matter particles, the gauge bosons and the Higgs boson (taken from [1]).

The basic theory of particle physics is the Standard Model. It contains three types of particles: quarks, leptons and bosons. The quarks and the leptons are organised in three generations, see Figure 1.1. The gauge bosons are mediating particles for the basic forces: electromagnetic, weak and strong force. Gravity is not yet introduced into the Standard Model. But since gravity is only a very weak force compared to the others, it has only minor influence on most of the particle physics topics. The strong force acts only on quarks and gluons because they carry the strong charge, called colour. There are three colours, red, blue and green, which can be added up to a colourless state. The theory describing the strong force is called QuantumChromoDynamics (QCD). Since we do not observe colour-charged particles in nature, the principle of confinement was introduced which claims that quarks have to form colourless bound states (hadrons). Quark-antiquark states are called mesons and 3-quark states are called baryons.

Further QCD calculations predicted an additional phase in the QCD phase diagram [Figure 1.2] in which the quarks and gluons form a de-confined medium called Quark-Gluon Plasma (QGP).

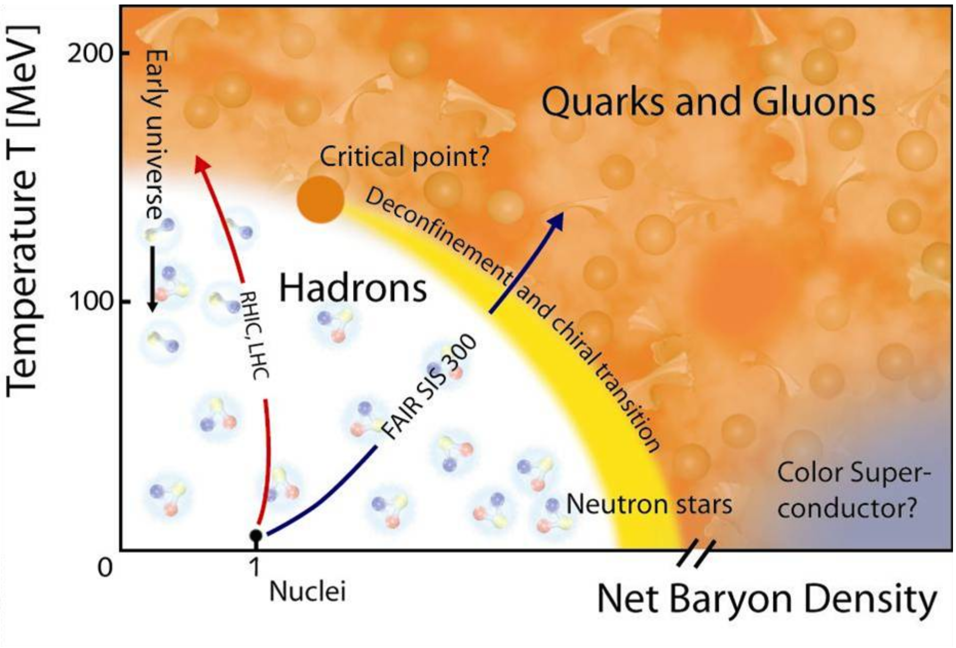


Figure 1.2.: The QCD phase diagram in terms of temperature and density [2]. The red line indicates the regimes of RHIC and LHC, while the blue one shows the regime of the future accelerator (SIS 300) at FAIR.

To form this special matter phase the energy density has to be increased far in excess of the values for ordinary nuclear matter. Two possibilities to reach this goal are by increasing the temperature or the density of the medium. Therefore it is difficult to create and study the QGP, but detailed information is important to test QCD. In addition, astrophysicists expect that the early universe had been in the state of a very hot QGP within the first (≈10) microseconds and that QGP is present in the core of neutron stars [3].

To study the QGP physicists collide heavy ions (e.g. gold or lead) with an energy density close to $1\,\mathrm{GeV/fm^3}$ [4] or higher to create a so-called fireball which develops into a QGP. QGP can be studied with the so-called soft and hard probes. The analysis approach via soft probes uses particles created via soft processes, which are interactions with low and intermediate transverse momentum transfer, to deduce information about the global properties (e.g. energy density, temperature and collective dynamics). More detailed information about the soft probe analyses can be found in reference [5] and [6].

Another direction of investigation is to analyse the energy loss of particles going through the QGP. But since there is not enough time to shoot a particle beam through it, the hard probes are used as a replacement. These are particles, which are formed by initial hard scattering processes with high transfer of transverse momentum and therefore exist before the formation of the QGP. Typical hard probes are jets, heavy quarkonia (like J/Ψ) and open heavy quarks (beauty and charm quarks).

1.2. Open heavy quarks

The (bare) masses of beauty ($m_b \approx 4.19\,\mathrm{GeV/c^2}$) and charm ($m_c \approx 1.29\,\mathrm{GeV/c^2}$) quarks are significantly larger than the QCD scale parameter $\lambda_{QCD} \approx 0.2\,\mathrm{GeV}$. Therefore their production can be described theoretically via perturbative QCD (pQCD) over the full range of momenta while gluon and light quarks can only be treated perturbatively at high transverse momenta [4]. Theoretical predictions achieve reasonable accuracy because of this unique feature of the heavy quark production. The heavy quarks are formed by initial hard scattering processes, which are in Leading Order (LO) gluon fusion and quark-antiquark annihilation (Figure 1.3), and Next-to-Leading Order (NLO) processes such as gluon splitting and flavour excitation get important [4] (Figure 1.4).

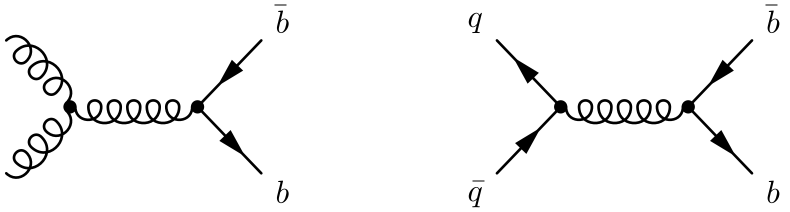


Figure 1.3.: Examples for leading order Feynman diagrams. The left diagram shows gluon fusion and right one quark-antiquark-annihilation.

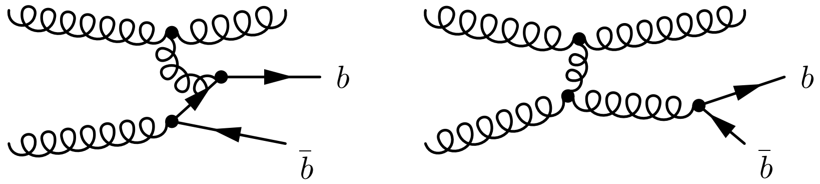


Figure 1.4.: Examples for next-to-leading order Feynman diagrams. The left diagram shows flavour excitation and the right one gluon splitting.

The open heavy flavour quarks quickly hadronise and form relatively long lived particles, see Table 1.1. These hadrons are mostly D mesons for the charm quark and B mesons for the beauty quark as well as Λ_b and Λ_c, respectively. For short hand notation these hadrons containing beauty quarks or charm quarks are called b or c hadrons, respectively, in this thesis. By measuring the yield of b hadrons in pp-collisions the production cross-section can be evaluated and compared to pQCD-model calculations. In addition, these measurements provide important reference values for measurements in Pb-Pb collisions. As these hard probes are quite rare, it is crucial to have good methods to detect and distinguish them.

Table 1.1.: Properties of hadrons carrying open heavy flavour quarks. Given are mass, lifetime, decay length $c\tau$, the dominating decay mode and its branching ratio [7]. The related antiparticles are not mentioned explicitly.

Particle	**Mass** [MeV/c^2]	**Lifetime** [fs]	$\boldsymbol{c\tau}$ [µm]	**Decay mode**	**B.R.** [%]
$\mathbf{D}^+$	1869.62 ± 0.15	1040 ± 7	312 ± 2	e^+anything	16.07 ± 0.30
				μ^+anything	17.6 ± 3.2
				$K^- 2\pi^+$	9.13 ± 0.19
$\mathbf{D^0}$	1864.86 ± 0.13	410.1 ± 1.5	123 ± 1	e^+anything	6.49 ± 0.11
				μ^+anything	6.7 ± 0.6
				$K^-\pi^+$	3.88 ± 0.05
$\mathbf{D}_s^+$	1968.50 ± 0.32	500 ± 7	150 ± 2	e^+anything	6.5 ± 0.4
				$K^+K^-\pi^-$	5.49 ± 0.27
$\boldsymbol{\Lambda}_c^+$	2286.46 ± 0.14	200 ± 6	60 ± 2	$p\,K^-\pi^+$	5.0 ± 1.3
$\mathbf{B}^+$	5279.26 ± 0.17	1641 ± 8	492 ± 2	$l^+\nu_l$ anything	10.99 ± 0.28
$\mathbf{B}^0$	5279.58 ± 0.17	1519 ± 7	455 ± 2	$l^+\nu_l$ anything	10.33 ± 0.28
$\mathbf{B}_s^0$	5366.77 ± 0.24	1516 ± 11	441 ± 8	D_s^+anything	93 ± 25
				$l^+\nu_l$ anything	9.5 ± 2.7
$\mathbf{B}_c^+$	6274.5 ± 1.8	452 ± 33	136 ± 10	$J/\Psi\, l^+\nu_l$ anything	$(5.2^{+2.4}_{-2.1}) \cdot 10^{-3}$
$\boldsymbol{\Lambda}_b^0$	5619.4 ± 0.6	1429 ± 24	428 ± 7	$\Lambda_c^+ l^- \bar{\nu}_l$ anything	9.8 ± 2.2

To analyse open heavy quarks electrons and muons are commonly used, because they have some special features. First of all they can only be created via electromagnetic interactions, like quark-antiquark annihilations and photon conversions, or via weak interactions. Therefore they are not part of the initial collision material. In addition, these leptons are the most common long-lived decay products of the hadrons carrying heavy flavour after pions and kaons. Furthermore, the branching ratios for semi-electronic decays are rather high (see Table 1.1) compared to most of the exclusive hadrons decays. Since the b hadrons have large lifetimes and decay at a point displaced from the collision point, an additional very important feature can be used to select electrons from these hadrons, the "impact parameter".

The transverse impact parameter d_t is the distance of closest approach (DCA) to the primary vertex in the plane perpendicular to the beam direction, see Figure 1.5. The sign of d_t indicates whether the prolonged particle trajectory misses the primary vertex on the left or on the right side and takes its curvature into account. Since b and c hadrons have much bigger lifetimes than for example neutral pions, which are a big source of electron background via Dalitz decays, their daughter electrons have a broader d_t-distribution.

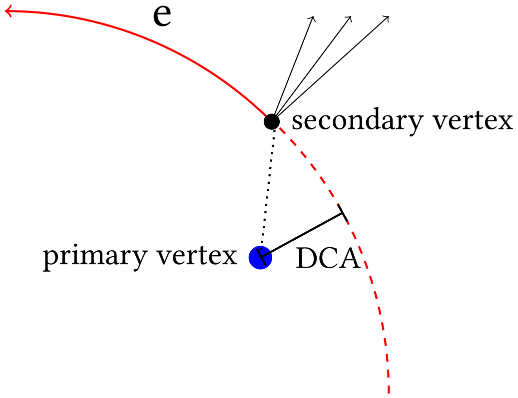


Figure 1.5.: Definition of the Impact Parameter which is the distance of closest approach (DCA). The sign of the impact parameter is given by the side on which the particle misses the primary vertex and its curvature.

1.3. Recent approach in ALICE

One recent way of analysing the beauty yield with the ALICE detector is to determine the yield of electrons from semileptonic decays of hadrons carrying beauty. This analysis starts with selecting charged particle tracks fulfilling some electron identification criteria based on the information from the Time Projection Chamber (TPC) and the Time-of-Flight detector (TOF). A possible choice for these criteria is the following. The first step is to reject all candidates which are not within 3σ around the time-of-flight expected for an electron because more than 99% of the electron signals are expected to be within this range. To improve the purity of the sample, which is still contaminated by pions, an additional cut on the deviation from the expected energy loss in the TPC is introduced. To obtain a sample which is as pure as possible the usual cut is to request a deviation between 0 and 3σ [Figure 1.6]. Sometimes additional cuts based on the Transition Radiation Detector (TRD) are used to improve the selection for transverse momenta above $4\,\mathrm{GeV}/c$.

For the further analysis, a technique called "cocktail subtraction" method is used. The electron sample contains electrons not only from beauty hadron decays but also from decays of other hadrons (e.g. charm hadrons, π^0, ,...) and from photon conversions in the detector material. This background electrons are described by the so-called "cocktail". The p_T-distributions of the cocktail electrons is evaluated by simulating the decays of the background sources using ALICE measurements of their spectra. To reduce the abundance of these background electrons a cut on the transverse impact parameter of the electron is applied. The efficiencies of this selection are calculated using Monte Carlo simulations where the p_T-distributions of the background hadron have been re-weighted to reproduce the measurements used to create the cocktail. Since this method depends on data coming from other measurements it is limited to the transverse momentum regimes covered by these. Therefore this approach has, for example, its difficulties below 1 GeV/c because D meson were measured only above this value. This type of analysis is statistical and does not offer any chance to really distinguish hadrons carrying beauty or charm on a candidate by candidate basis. More detailed information about this analysis approach can be found in [8] and [9] .

1.4. Possible new selection criteria

The aim of this thesis was to check additional criteria for the determination of the b hadron yield. Therefore characteristic properties of the b hadrons were studied to check if they could be useful to discriminate between b and c hadrons. Most of the b hadrons have a larger lifetime than the c hadrons, therefore a displaced decay vertex is more common. If these secondary vertices can be reconstructed by identifying the decay products of the hadrons, the mass and the decay length could be determined. The decay length ($L = \beta\gamma c\tau$) is the distance between the secondary and the primary vertex and is proportional to the lifetime of the particle. The pure distance between the primary and the secondary vertices cannot be used as a separation criteria because of its additional dependence on the particle momentum. Therefore a momentum independent quantity has to be used. The obvious first guess to divide the decay length by the momentum results in a mass dependent quantity, $\frac{L}{p} = \frac{\gamma\beta c\tau}{\gamma m\beta c} = \frac{\tau}{m}$. Since the real mass is not known when analysing real data the b and c hadron mass has to be reconstructed like for the mass criteria, see subsection 3.3.2.

Even though not all decay products of b and c hadron decays are measurable, these two quantities could possibly be good criteria for distinguishing them, see Table 1.1. Having a higher mass also results in a higher energy content at the same momentum and therefore heavier particles typically have more decay daughters in their final state. In addition, the probability to have only very few particles created by the decay of very heavy particles get smaller with increasing initial mass. Therefore the distribution of the number of b hadron daughter particles is expected to have its mean at higher values than the distribution for the c hadrons.

The new approach to analyse the b yield starts with reconstructing the secondary vertices and uses the following criteria to select the ones belonging to hadrons containing beauty quarks:

- number of prongs contributing to the reconstructed secondary vertex
- lifetime of the hadron which decayed at the reconstructed vertex
- reconstructed invariant mass

The first step of checking if these criteria could be used to improve the discrimination between hadrons containing beauty or charm quarks, was the analysis using MC simulations.

1.5. Estimate of b hadron and resulting electron yield

The expected yield of a certain particle is a good criterion to decide if it is useful to study it. Therefore an estimate of the yield of b hadrons in pp-collisions was carried out. For this calculation the goal luminosity of the LHC at ALICE, $\mathcal{L} = 10^{30}\ \mathrm{cm}^{-2}\mathrm{s}^{-1}$, and the beauty production cross-section per unit-rapidity, $\frac{\mathrm{d}\sigma_{\mathrm{b\bar{b}}}}{\mathrm{dy}} = 42.3 \pm 3.5(\mathrm{stat.})^{+12.3}_{-11.9}(\mathrm{syst.})^{+1.1}_{-1.7}(\mathrm{extr.})$ µb [8], were used.

Rapidity (y) is an alternative way to express the velocity of a particle and was introduced to restore simple adding of velocities even in the relativistic regime. It is defined by the velocity (v) and the speed of light (c) or by the energy (E) and the absolute value of the momentum (p):

$$y := \tanh\left(\frac{v}{\mathrm{c}}\right) \equiv \frac{1}{2}\ln\left(\frac{E+p}{E-p}\right)$$

In experimental particle physics rapidity is often given relative to the beam axis changing $|\vec{p}|$ into the momentum along the beam axis (p_{z}). In high energy physics the mass of the particles is often negligible compared to the momentum. Therefore the concept of the rapidity gives rise to a new quantity, the pseudorapidity η, which is linked to the polar angle θ:

$$\eta = \frac{1}{2}\ln\left(\frac{p+p_{\mathrm{z}}}{p-p_{\mathrm{z}}}\right) \equiv -\ln\left[\tan\left(\frac{\theta}{2}\right)\right]$$

In particle physics, the pseudorapidity is commonly used instead of the polar angle.
For this estimate, a runtime of $\tau = 10\,\text{months}$ with $50\,\%$ stable beam was assumed. To convert the resulting number of b hadrons into an electron yield, the beauty to electron branching ratio, $\text{BR}_{\text{H}_\text{b}\to\text{e}} + \text{BR}_{\text{H}_\text{b}\to\text{H}_\text{c}\to\text{e}} = (20.5 \pm 0.7)\,\%$ [8], was used. The rapidity covered by ALICE and used for analysis is $|y| \leq 0.8$.

$$\text{integrated luminosity: } L = 0.5 \int\limits_0^\tau \mathcal{L}\text{d}t \approx 6.32 \times 10^7\,\mu\text{b}^{-1}$$

$$\text{expected number of b hadron: } N_{\text{H}_\text{b}} = \int\limits_{|y|} \frac{\text{d}\sigma_{\text{b}\bar{\text{b}}}}{\text{d}y}\, L\, \text{d}y \approx 4.28 \times 10^9$$

$$\text{expected number of electrons: } N_\text{e} = \text{BR}_{\text{H}_\text{b}\to\text{e}} \cdot N_{\text{H}_\text{b}} \approx 8.77 \times 10^8$$

To determine the number of collisions these calculations were repeated with the total inelastic cross section determined with ALICE, $\sigma_{\text{inelastic}} = 73.2^{+2.0}_{-4.6} \pm 2.6\text{mb}$ [10]. The resulting number of collisions for this integrated luminosity is $\text{N}_{\text{coll.}} = 4.63 \times 10^{12}$. Therefore "detectable" b hadrons are created only in about every thousandth collision, $\frac{\text{N}_{\text{H}_\text{b}}}{\text{N}_{\text{coll.}}} = 9.25 \times 10^{-4}$.

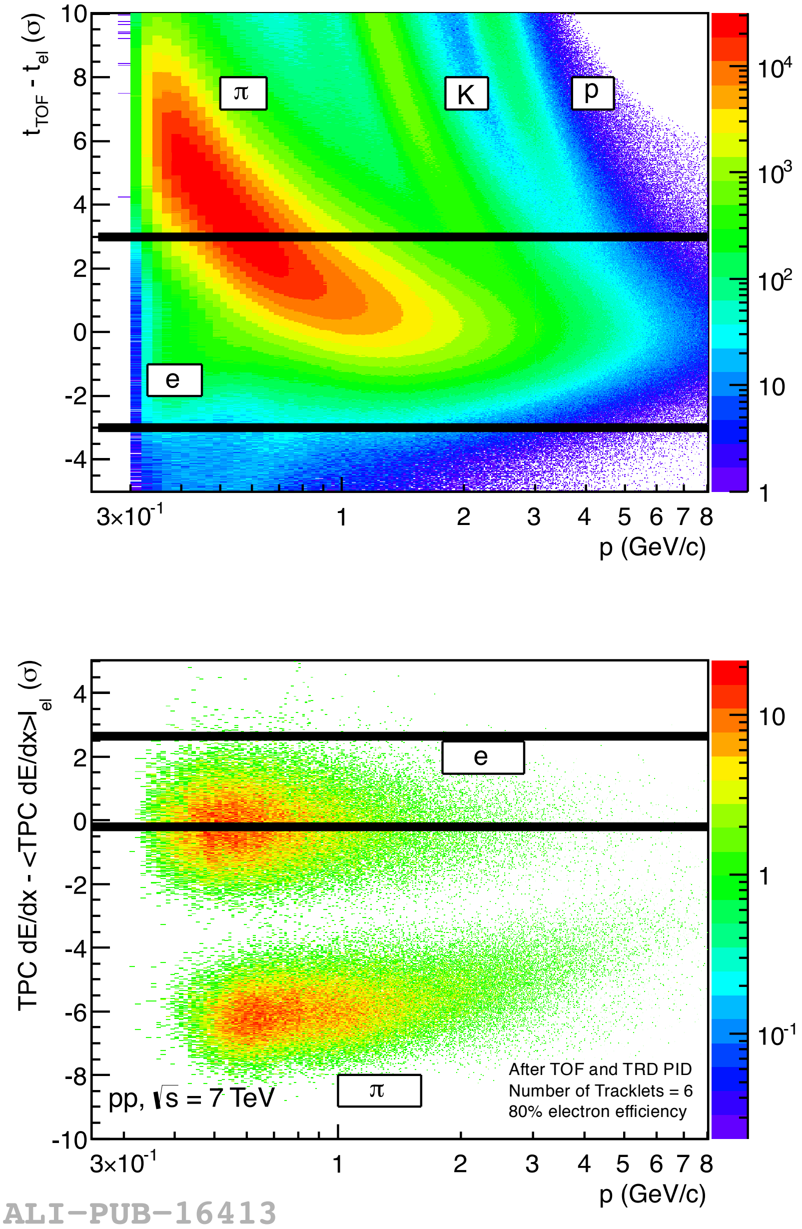


Figure 1.6.: The upper panel shows the deviation of the measured TOF time from the expected electron time-of-flight as a function of the particle momentum. As an additional information, the particle type of the different signal sources is given. The cuts used for the electron identification are shown as black lines. After a second cut using the TRD a distribution of the energy-loss deviation in the TPC as a function of the particle momentum was obtained. It is shown in the lower panel indicating very clearly why the additional cut on the deviation to the expected TPC value is needed to reduce the pion contamination. These pictures are taken from [9].

2. ALICE Detector

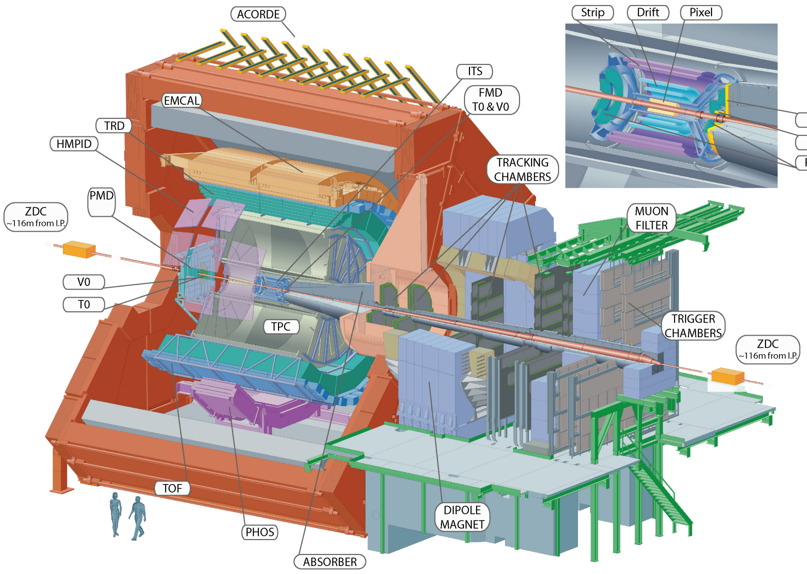


Figure 2.1.: The ALICE detector is built up of 18 different detectors which are indicated by their acronyms. The central detectors (left side) are mounted inside the large solenoid magnet which had already been part of the L3 detector at LEP. The Muon spectrometer with its dipole magnet is placed in forward direction (right side). An additional blow-up of the innermost region, showing the Inner Tracking System, the forward trigger and the multiplicity detectors, was added on the top right position. This picture is taken from [11].

ALICE is one of the four main experiments at the Large Hadron Collider (LHC) at CERN. It was primarily designed for data taking at heavy ion collisions as is indicated by the fact that the name is an acronym for "A Large Ion Collider Experiment".
The ALICE detector has an overall size of 16 x 16 x 26 m^3 and a total weight of about $10\,000\,\mathrm{t}$ and was optimized to cope with the high particle densities expected from heavy-ion collisions. It can be divided into two parts: the central-barrel detectors and the MUON spectrometer. The MUON spectrometer covers the range of $-4 < \eta < -2.5$ and is used for measurements of muons, quarkonia and light vector mesons. The very heart of ALICE are the central-barrel detectors which are covering the midrapidity region ($|\eta| < 0.9$). They

are embedded in the L3 solenoid magnet providing $\mathrm{B} = 0.5\,\mathrm{T}$. These detectors are the Inner Tracking System (ITS), the Time Projection Chamber (TPC), the Transition Radiation Detector (TRD), the Time Of Flight detector (TOF), the Photon Spectrometer (PHOS), the Electromagnetic Calorimeter (EMCal) and High Momentum Particle Identification Detector (HMPID) [12]. Inside the central region there are three additional detectors, the Forward Multiplicity Detector (FMD), V0 and T0. These detectors are used to determine the number of particles produced in the collision and their spatial distribution. T0 is also used to measure the time when the collision has taken place very precisely.
The following sections will give more detail information about the subdetectors used in this analysis.

2.1. Inner Tracking System (ITS)

The ITS is the innermost detector system of the *central barrel* and is built up of six cylindrical layers with three different types of silicon detectors. At radii of $3.9\,\mathrm{cm}$ and $7.6\,\mathrm{cm}$, the two innermost layers are mounted, the Silicon Pixel Detectors (SPD). These provide a very good spatial resolution in the transverse plane ($12\,\mu\mathrm{m}$) and in the beam direction ($100\,\mu\mathrm{m}$) [9]. The SPD layers are of central importance for the reconstruction of primary and secondary vertices and for determining the deviation of tracks from the primary vertex. The last point is essential for the current way of analysing heavy flavour hadron decays. The intermediate and outer layers consist of Silicon Drift Detectors (SDD) and double-sided Silicon Strip Detectors (SSD), respectively. These two parts are also capable of providing particle identification information via deposited energy. The whole ITS extends to a radius of $43\,\mathrm{cm}$ and provides spatial resolution information (tracking) for charged particles near the beam pipe [9]. This contributes to the high momentum and angular resolution of particle trajectories at ALICE.

2.2. Time Projection Chamber (TPC)

The main tracking detector of ALICE is the TPC. It provides momentum and particle identification informations. The TPC is a cylindrical drift detector with a length of $5\,\mathrm{m}$, a diameter of $5.6\,\mathrm{m}$ and is filled with Ne (85.5%), CO_2 (9.5%) and N_2 (4.8%). It is divided into two drift regions by a central high-voltage electrode. Charged particles, passing through it, ionize the gas molecules and free electrons. These electrons drift towards the end plates of the TPC which are equipped with multi-wire proportional chambers [9]. The coordinates transverse to the beam direction are given by the signal position at the end-cap while the third dimension is reconstructed via the drift time. The TPC is one of the main detectors for particle identification which is determined by the energy loss per unit length $\left(\frac{\mathrm{dE}}{\mathrm{dx}}\right)$ given by the collected charge.

2.3. Time of Flight detector (TOF)

The TOF is a gas detector consisting of Multigap Resistive Plate Chamber (MRPCs), a type of detector developed to meet the requirements of time resolution and number of read-out channels [11]. It contains about 157000 individual cells covering an area of $160\,\mathrm{m}^2$ at a radius of $3.7\,\mathrm{m}$ [13]. The TOF array is an important detector for particle identification because it measures the flight time of the individual particles between the collision and the point were it is mounted. The moment of collision is determined by the T0 signal, if it is available [9]. Using the determined time and the momentum of a certain particle, one can determine its mass and therefore identify the particle.

3. Feasibility studies

3.1. Monte Carlo samples used for this analysis

Table 3.1.: Overview of the Monte Carlo simulation samples used for this analysis

Sample name	**Generator**	**Collision energy**	**extra information**
LHC10f6a	Pythia	7 TeV	Minimum Bias
LHC10f6	Phojet	7 TeV	Minimum Bias
LHC10f7a_d	Pythia	7 TeV	Enhanced heavy flavour

Phojet [14] and Pythia [15] are particle generators needed to simulate the particle produced in the collision. In the minimum bias samples the charm and beauty production cross sections are more or less reproduced. To produce the enhanced sample, a c-$\bar{c}$ or b-$\bar{b}$ pair was requested in every event. Therefore the statistics of heavy flavour hadrons are enriched. This special data-set has to be used carefully because this enhancement of heavy flavour results in wrong production cross sections. Hence, some features from the minimum bias Pythia samples are not reproduced, e.g. the branching ratios.

3.2. Pre-analysis

Before using the new criteria for an analysis purpose, one has to check whether they are expected to create pure enough samples in real data. For this purpose the selection criteria were tested with Monte Carlo (MC) simulations of proton-proton collisions at $\sqrt{s} = 7\,\text{TeV}$ fitting to data taken with ALICE during the 2010 LHC run.

But since there are several possible MC particle generators, it has to be assessed which are working well for this analysis. Therefore the branching ratios of the semi-electronic decays of b hadrons and the total number of daughter particles of b and c hadrons were checked in Pythia and Phojet samples (Figure 3.1 and Figure 3.2). In this thesis the particles which are created during the total decay chain are called prongs.

Table 3.2.: Branching ratios (BR) for semi-electronic decay of B mesons and Λ_b. Given are the PDG values [7], the values calculated with Pythia and Phojet and the associated deviations from the PDG value in units of the combined error. The related antiparticles are included in the numbers of the respective particle.

Particle	$\mathbf{BR_{PDG}}$ [%]	$\mathbf{BR_{Pythia}}$ [%]	**deviation**	$\mathbf{BR_{Phojet}}$ [%]	**deviation**
$\mathbf{B^+}$	10.99 ± 0.28	10.89 ± 0.06	0.35	12.6 ± 0.3	4.0
$\mathbf{B^0}$	10.33 ± 0.28	11.03 ± 0.04	2.5	12.8 ± 0.4	6.2
$\mathbf{B^0_s}$	9.5 ± 2.7	10.99 ± 0.09	0.6	13.8 ± 1.0	1.5
$\mathbf{B^+_c}$	–	12.5 ± 0.8	–	16.7 ± 17.9	–
$\mathbf{\Lambda^0_b}$	9.8 ± 2.2	10.8 ± 0.1	0.5	12.9 ± 1.6	1.1

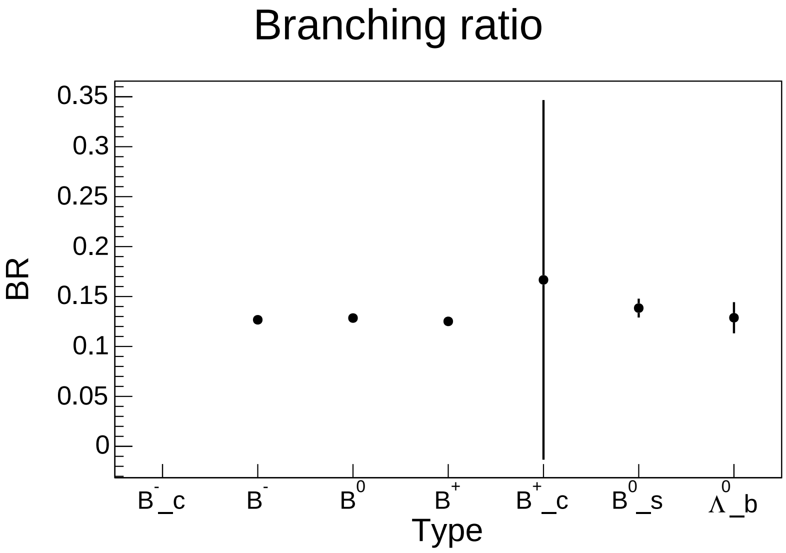
These two criteria showed that the particle generator Phojet should not be used for the further analysis. One reason is that Phojet does not reproduce the expected branching ratios of semi-electronic decays of b hadrons. The branching ratios determined with Phojet differ from the PDG-values much more than the ones determined with Pythia. Most of the Pythia results differ by less than 3σ. Since the PDG does not offer a value for the theoretical branching ratio of semi-electronic decays of $B_c^{+/-}$, it cannot be compared to the values determined with Pythia and Phojet. But since the $B_c^{+/-}$ is a very rare particle (Figure 3.3) only a small effect on the analysis is expected and therefore it was excluded from the analysis.

Since higher numbers of prongs are more likely for heavier particles, the number of prongs originating from b hadrons should extend to higher values and at some point higher numbers of prongs should be more frequently belonging to b hadrons. In addition, the mean number of total prongs from c hadrons should be lower than the one for b hadrons. Since these features are only observable in the simulations with Pythia, this is an important argument against Phojet.

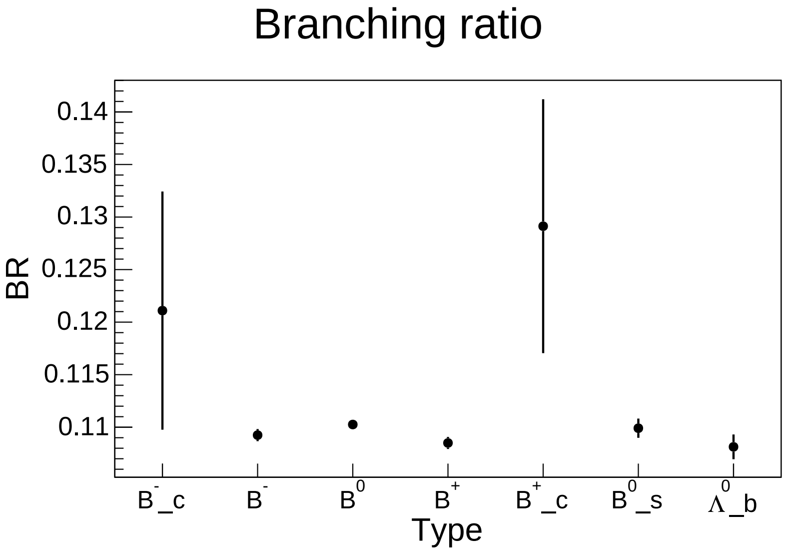
Therefore, Pythia6 and Geant3 [16] were used for the further analysis. Geant is a detector simulator which contains the full information about the material and the geometry of the whole detector and is used to simulate the interactions of the produced particles with the detector material.

Another basic information is about unexpected features in the pseudorapidity (η) and the transverse momentum (p_T) distribution for B mesons and their electron daughters because this could lead to the need of spatial or momentum dependent treatment.

Figure A.1, which can be found in the appendix, illustrates that the spatial distributions within the acceptance of the ALICE detector are almost flat. Therefore the only spatial constraint is the acceptance of the detector. The momentum distribution of the electrons and b hadrons also show the expected distribution (Figure A.2).



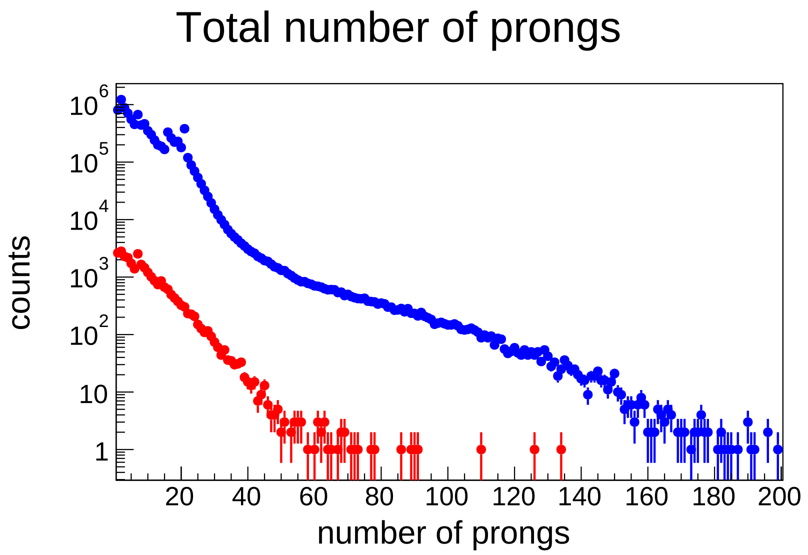
(a) Branching ratios for semi-electronic decays with Phojet



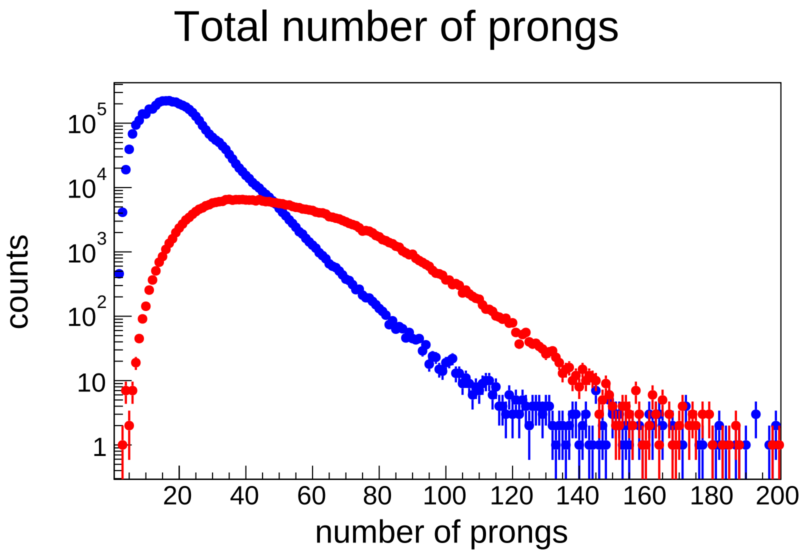
(b) Branching ratios for semi-electronic decays with Pythia

Figure 3.1.: Branching ratios for the semi-electronic decay of the different B meson types and Λ_b. The upper plot shows the results using Phojet, the lower one the Pythia results.

An additional basic information is the kinematic correlation between the daughter electrons and the b or c hadron itself, see appendix Figure A.3 and Figure A.4. If there had been no correlation, it would be impossible to use these electrons to start the reconstruction. The plots for the b hadrons were less clear because they are much rarer than c hadrons, therefore the histograms taken from the enhanced sample is shown in this thesis (Table 3.1). For the



(a) Number of prongs from b (red) and c (blue) hadrons including particles from Phojet and Geant



(b) Number of prongs from b (red) and c (blue) hadrons including particles from Pythia and Geant

Figure 3.2.: These plots are showing the abundance of the number of particles created during b or c hadron decay chain. Here every particle which is created during the whole decay chain of the respective hadron but not via secondary reaction with the detector material is taken into account. The entries belonging to b hadrons are shown in red and the ones belonging to c hadrons in blue. The upper plot shows the results using Phojet samples, the lower one the results using Pythia samples.

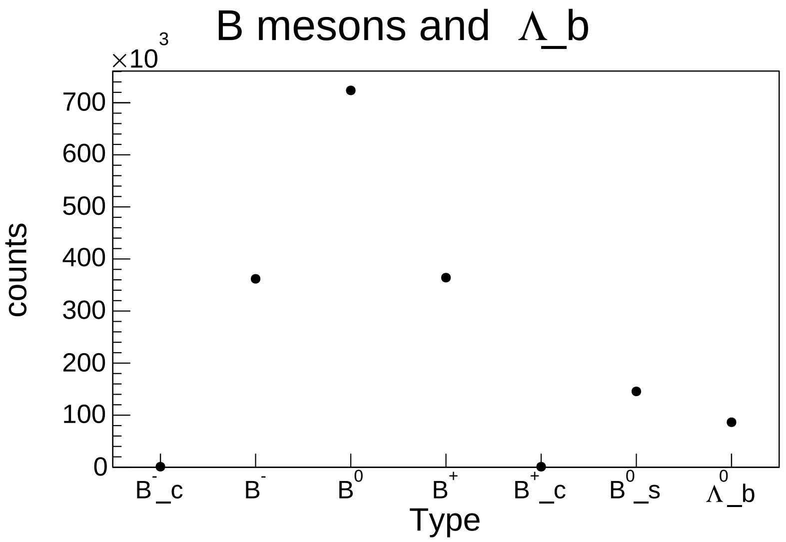


Figure 3.3.: The yield of the different b hadron types within the simulation with Pythia. The numbers for the neutral hadrons and their anti-particle are summed up to one entry. The most common B meson type are the B^0, the B^+ and their antiparticle. The rarest are the $B_c^{+/-}$.

further analysis all types of b and c hadrons, except of the B_c^+, were taken into account, if not stated differently.

3.3. Analysis of the new selection criteria

Since the analysis should provide general information about the feasibility of the proposed method, the b and c hadrons were taken into account independently from there decay type, i.e. the hadrons do not have to decay semi-electronically. This analysis only deals with the discrimination of b hadrons from c hadrons, it does not take any other background source into account. For simplicity the whole analysis was done p_T-integrated. The feasibility analysis was split into three parts characterized by the information taken into account. The first step of the analysis was based on the full information about the particles given by Pythia and Geant. Since not all daughter particles are detectable, the next step was to analyse how the detectability changes the information given. For this purpose some criteria for detectability were introduced, see subsection 3.3.2.

However, real data does not contain all the information known in the simulation (MC-Truth and Geant) and not all particles which are theoretically measurable are really reconstructed. Therefore the results of this feasibility checking step are not enough to show that the new selection criteria are useful. Hence, the last part uses the information provided by the track reconstruction algorithm included in AliROOT, the knowledge about the particle identity and the information which particles belong to b or c hadrons, respectively. The tracks were reconstructed mostly using information from hits in the ITS and ionization clusters in the TPC.

3.3.1. Part 1 - Based on the full information given by Pythia and Geant

The first part of this analysis is technically the simplest because all the particles which originate from b or c hadrons and belong to the particles called "primary particles" in Pythia were taken into account. In Pythia the "primary particles" are defined as physical primary particles, which are emerging from the collision, and all products from strong and weak decays within $c\tau < 3.9\,\mathrm{cm}$.

C hadrons created by b hadron decays were discarded for the c hadron analysis. This was done by a checking function, see Code A.1, lines 1733 - 1750. In this part of the analysis only the checks for the number of prongs and the decay length were made because the mass reconstruction would reveal only the literature values of the different b and c hadrons which would not be very interesting.

As seen in this first check, the number of prongs could be used as a separation criterion but it will possibly not lead to high purity. The selection will maybe suffer from contamination by c mesons with unexpected high number of prongs (figure 3.2(b)).

The lifetime of the particle should follow a exponential distribution according to the decay law. Therefore the literature lifetime should be found again by looking at the point where $\frac{1}{e}$ of the hadrons at the beginning are left. The respective values taken from Figure 3.4 are $\tau_{\mathrm{c}} \approx 410\,\mathrm{fs}$ and $\tau_{\mathrm{b}} \approx 1400\,\mathrm{fs}$ and fit well to the literature values of the c and b hadrons, respectively.

Since in hadronic collisions many more c hadrons than b hadrons are created and therefore the c hadrons are much more abundant even at higher lifetime values than the b hadrons the lifetime cannot be used to distinguish these hadrons very efficiently, see Figure 3.4.

3.3.2. Part 2 - Based on measurable particles and their properties given by Pythia and Geant

Since not all prongs of b and c hadrons are measurable this step of the analysis gives a first hint whether the criteria are useful for real data analysis without wasting too much time to

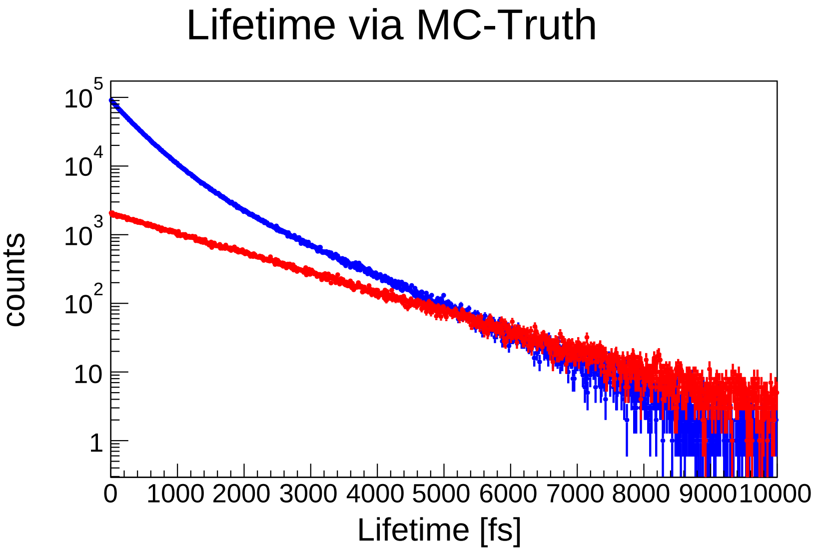


Figure 3.4.: This plot shows the distribution of the lifetime values for c hadrons (blue) and b hadrons (red) calculated with the Pythia and Geant information. This plot indicates that the lifetime criterion should not be used as a standalone criterion. Since the crossing point of the distribution vary with the ratio of b and c hadrons, the life time can be helpful when combining criteria.

implement the more difficult third part of this analysis. Therefore the analysis step of part 1 had been repeated with a pre-selection of detectable particles via a recursive function, which checks if the daughters of a given particle are measurable (Code A.1, lines 1473 - 1501).

The most fundamental criterion for measurability was that the particle has to appear in the region of $|\eta| \leq 0.8$, because the central barrel of ALICE covers only the region between $-0.9 \leq \eta \leq 0.9$ and the outermost region provides track information of lower quality. Therefore the outermost regions were not used for the analysis. The second criterion was set by the acceptance in the momentum space. Therefore particles with a high enough transverse momentum had been taken into account, $0.1\,\mathrm{GeV/c} \leq \mathrm{p_T} \leq 50\,\mathrm{GeV/c}$. The last step of selecting the measurable particles was to choose only charged particles which are interesting for the reconstruction of vertices (vertexing). These were electrons, muons, charged kaons, charged pions and protons.

The number of measurable prongs is much lower than the total number of prongs, but the distributions show that this criterion is much purer, if only the measurable prongs have been used, see Figure 3.5.

The first check for the mass criterion was implemented by reconstructing the four-momentum vector of the b or c hadron via its measurable prongs. This was done by checking if the daugh-

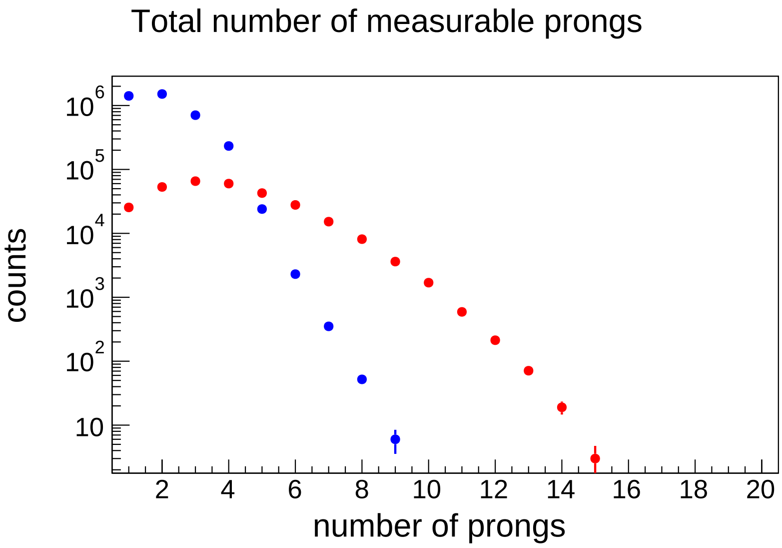


Figure 3.5.: This plot is showing the distribution of the number of measurable prongs for b hadrons (red) and c hadrons (blue). Since not all prongs are measurable the numbers do not extend to values as high as for the total number of prongs, but by cutting back to the measurable prongs the distribution became better for selection.

ters of a given particle belong to the "primary particles" of Pythia and are detectable. If these two conditions were fulfilled the four-momentum vector of this daughter was added to the four-momentum container belonging to the respective particle. If a prong was not detectable the function was restarted with this prong to reconstruct its four-momentum and later add it to the four-momentum container of the respective mother particle. The invariant mass is given by the norm of the four-momentum vector. Since we have to reconstruct the decay vertex of the hadrons to access the information about the new selection criteria in real data, all four-momentum vectors containing only values from one prong were neglected. The explicit implementation can be found in the appendix, Code A.1, lines 1547 - 1616. The resulting distributions of the invariant mass of the b and c hadrons show that the mass could offer the chance to create a totally pure sample of b hadrons (Figure 3.6).

In this part the lifetime was calculated using the distance between the production point of the first measurable direct daughter of the respective hadron and the primary vertex. The mass and the momentum had been taken from Pythia and Geant. Since the distribution for the lifetime has not changed by much in this step, it will not be discussed in this section.

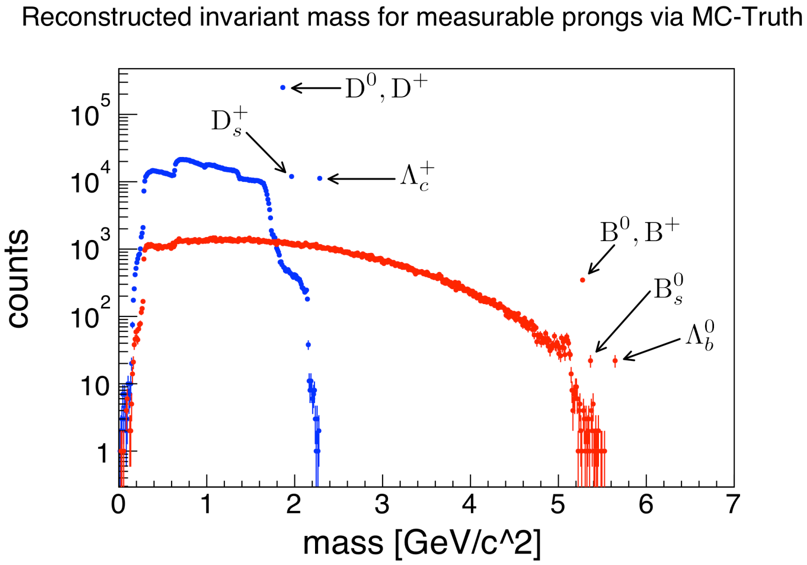


Figure 3.6.: The distribution of the mass values reconstructed with the measurable prongs using momentum and particle identity information taken from Pythia and Geant. The c hadrons are shown in blue and the b hadrons in red. The peaks for the fully reconstructed c and b hadrons are labelled with the respective particles.

3.3.3. Part 3 - Based on reconstructed tracks

For this part the information gained by the reconstruction with AliROOT, the so-called ESD track, had to be linked to the MC-Truth and Geant information to keep the knowledge of particle identity and which particles descend from b or c hadrons. To achieve this goal an array was filled with the MC number of the particle, the MC number of the ancestor b or c hadron and the generation for every measurable particle belonging to b and c hadrons with a recursive function during the MC analysis part (Code A.1, line 1473 - 1501). In this context a generation is defined as the number of decays needed to get from the respective hadron to the particle, e.g. if the B0 decays semi-electronically the resulting electron belongs to the first generation.

If a track is found for a particle from MC-Truth or Geant, AliROOT assigns the MC number to the track as an additional label. Therefore these labels had been compared to the array information to restore the link to the MC information. But since the quality of the different tracks vary, some limiting conditions were introduced to leave tracks with very low reconstruction quality out of the analysis. These conditions were set similar to the ones chosen in [9]:

- The tracks had to be refitted in ITS and TPC to receive a higher track quality.

- The minimal number of charge clusters used for tracking in TPC was set to 100 to obtain only tracks with enough data points to be sure about the track quality.
- The maximum χ^2 for each cluster was set to 4 to prevent fake tracks from close-by clusters created by more than one particle.
- Kink daughters were discarded to reduce double tracks, e.g. for electrons suffering from bremsstrahlung.
- The maximum distance of closest approach (DCA) to the primary vertex was limited to 1 cm in the transversal plane and 2 cm along the beam direction. This is useful to reduce contributions from background tracks (e.g. cosmic rays) and non-primary tracks.
- For electron tracks, the minimal number of hits in ITS was set to 4. In addition, only electron tracks having hits in both innermost layers of ITS (SPD) had been used to reduce contributions from photon conversions in the detector layers.

To test the efficiency of all the separation criteria, the impact parameter of electrons descending from b or c hadrons divided by the combination of the uncertainty of the track and the primary vertex was added. In this case the impact parameter was evaluated in all three spatial directions, that means the impact parameter d in this analysis is defined as $\mathrm{d} = \sqrt{\mathrm{d}_\mathrm{t}^2 + \mathrm{d}_\mathrm{z}^2}$ with the impact parameter in the transverse plane (d_t) and the distance in beam direction (d_z).

Figure 3.7 shows that the impact parameter cannot be used as a standalone criterion. However, it can be used to pre-select b hadron candidates by cutting away the regions where the c hadrons dominate. The main goal of using this criteria is to reduce the number of reconstructed vertices not belonging to b or c hadrons at all and to lower the contribution by c hadrons. The efficiency, which means the number of b hadrons selected with a cut using a certain criterion against the total number of b hadrons with measurable daughter, is shown in Table 3.3. The error on this efficiency is determined by $\Delta_\mathrm{eff} = \frac{\sqrt{\Delta_{\mathrm{N_b}}^2 + \mathrm{N_c^2}}}{\mathrm{N_{meas.}}}$ with the error of the number of selected b hadrons ($\Delta_{\mathrm{N_b}}$), the contamination by c hadrons ($\mathrm{N_c}$) and the total number of b hadrons with measurable prongs ($\mathrm{N_{meas.}}$). For the impact parameter the number of b hadrons with measurable electron daughter instead of measurable daughters was used. Two possible cuts were chosen which do not have errors higher than their efficiency and are still so efficient that enough hadrons remain.

The number of prongs only changed a little compared to the results from part 2 and offers an additional nearly pure criterion to combine with the others (Figure 3.8 and Table 3.3).

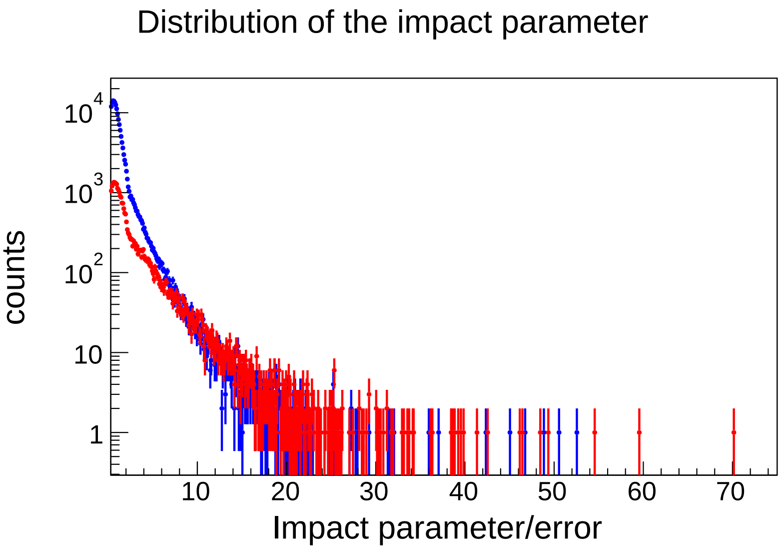


Figure 3.7.: The distribution of the impact parameter of the electrons descending from b hadrons (red) and c hadrons (blue) calculated with KF package, see [17]. This plot indicates that the impact parameter can only be used to pre-select b hadron candidates by cutting away the region where most of the electrons belong to c hadrons.

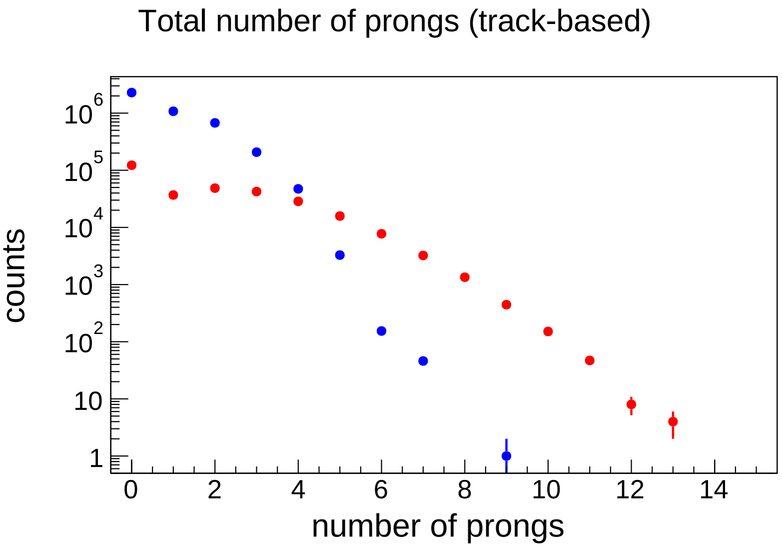


Figure 3.8.: This plot is showing the distribution of number of tracks contributing to the reconstructed b hadron (red) and c hadron (blue) vertices. Since not all measurable prongs are really detected the numbers are not equal to values for the part 2 but the possibility of distinguishing b and c hadrons has not changed by much.

Table 3.3.: The efficiency of the selection criteria. Given are two examples for a lower cut and the respective efficiency. The efficiency is defined by the number of b hadrons selected divided by the total number of b hadrons with measurable prongs. The error contains the error on the summation of the selected b hadrons and the c hadron contamination. The values given for the number of prongs and the mass criteria are the cuts which are the most efficient cut which still has not a too high error and the most efficient pure cut. For the lifetime the most efficient cut is given for the ratio of b and c hadrons given by Pythia and for a pre-selected sample containing as many b hadrons as c hadrons. The efficiencies given for the impact parameter belong to the cut values which are the most efficient with not too high errors and are given with respect to the total number of b hadrons with measurable electron daughter.

Criteria	**cut value**	**efficiency**$[\%]$	**cut value**	**efficiency**$[\%]$
impact parameter	10σ	3.0 ± 2.3	12σ	1.9 ± 1.3
number of prongs	6	4.26 ± 0.08	9	0.215 ± 0.008
lifetime	$1300\,\mathrm{fs}$	30 ± 132	$1300\,\mathrm{fs}$	30 ± 19
invariant mass	$2.3\,\mathrm{GeV/c^2}$	16.7 ± 0.5	$2.48\,\mathrm{GeV/c^2}$	13.44 ± 0.07

The mass for the hadrons was reconstructed in a similar way as in the previous part. The principle of the reconstruction function is the same but since the four-momenta given by the reconstructed track were used instead of the one given by Pythia and Geant some modifications had to be made. First of all only measurable daughters with a corresponding ESD track were used to fill the four-momentum container. The particle identity and assumption for the track mass were taken from Pythia and Geant. In addition, the number of prongs with corresponding ESD tracks were evaluated with this function. Since we want to reconstruct the respective hadrons with real data more easily, the AliRoot KF (Kalman filter) package [17], was used. This package reconstructs vertices or particles by adding up daughter particle tracks. Therefore not only the four-momentum is reconstructed, but the whole mother particle properties (e.g. invariant mass and decay length). The daughter particle tracks were either created using an existing ESD track and the information about the particle identity or by reconstructing it with its daughters. The KF package is a very powerful tool and provides many possibilities to access the information needed for the further analysis more easily, e.g. the position of the vertices or the number of prongs. The detailed implementation is given in the appendix, Code A.1, lines 1618 - 1731.

The distribution of the reconstructed mass show that this criterion can be used to create a total pure selection (Figure 3.9) and still having a good efficiency (Table 3.3). The two c hadrons with masses higher than the literature value are caused by electron tracks created e.g. by photon conversion. These tracks were not identified properly using the TPC and TOF and therefore the special electron track cuts had not discarded them.

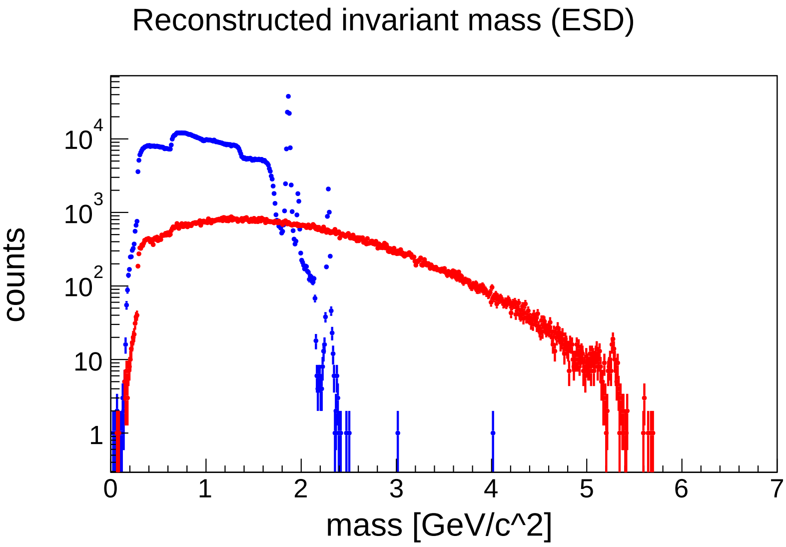


Figure 3.9.: The distribution of the mass values reconstructed with the ESD track information. The c hadron values (blue) show a clear cut at about $2.5\,\mathrm{GeV/c^2}$, while b hadrons with a mass up to $5.4\,\mathrm{GeV/c^2}$ have been found.

The lifetime was determined using the distance between the reconstructed secondary vertex and the primary vertex calculated with the KF package. The momentum and the mass were taken from the reconstructed hadrons, too. Since fewer hadrons with high masses are reconstructed, the number of b hadrons in the high lifetime region is lower. Therefore the lifetime cannot be used without any pre-selection, which had lowered the ratio between b and c hadrons, any more, see appendix Figure A.5. Therefore the lifetime criterion was checked by normalizing the data from b and c hadrons to the same total number. Figure 3.10 indicates that the lifetime can be useful to improve selections which had already lowered the number of c hadrons contained in the sample. The efficiency of the lifetime criterion was evaluated for a cut near to the crossing point (at 1300 fs) shown in the normalized plot using the un-normalized and the normalized data.

3.4. Correlations between the criteria

Since the selection criteria have to be combined to efficiently select and distinguish b and c hadrons, it is important to know how they are connected with each other. The respective

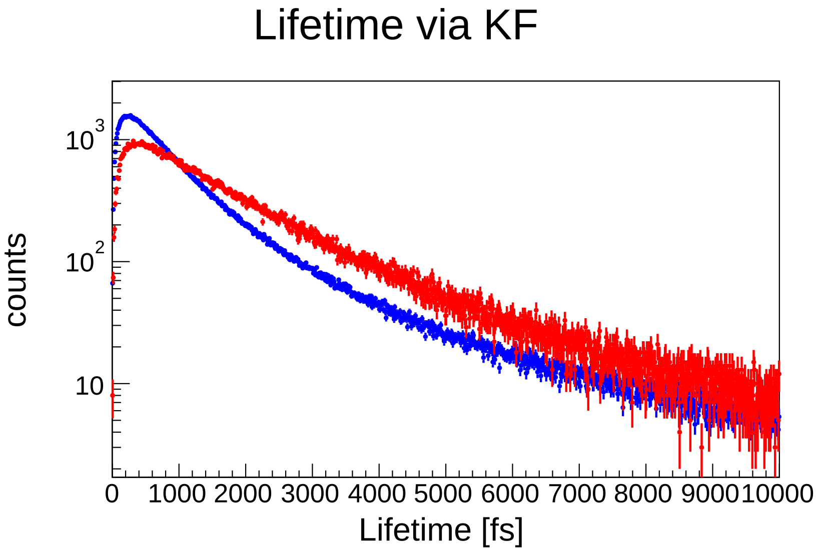


Figure 3.10.: This plot shows the distribution of the lifetime values for c hadrons (blue) and b hadrons (red) calculated with the information taken from the hadrons reconstructed with the KF package normalized to the same number of entries. This plot indicates that the lifetime criterion can be used, if a pre-selection had lowered the ratio of c and b hadrons.

plots showing the correlations of the analysed criteria can be found in the appendix (Figure A.2 ff.). These plots indicate that the cut in the minimum impact parameter has to be chosen as small as possible to prevent losing too many candidates. Therefore it would be good to use the impact parameter only to reduce the number of non-b and non-c hadron candidates.

The impact parameter seems to be slightly linked to the number of prongs because a high number of prongs leads to a smaller maximum impact parameter. A higher number of prongs leads to a distribution of the rest-energy of the hadron over more particles. Therefore the daughter particles cannot deviate from the mother direction by much and therefore lower impact parameter values are expected. The correlation of mass and the number of prongs is caused by the fact that a higher number of prongs leads to a more complete reconstruction and therefore to a higher minimum mass. The distance of the reconstructed vertices from the primary vertex is linked to the impact parameter of the electron. Therefore the lifetime and the impact parameter are expected to be correlated. Since the mass was used to evaluate the lifetime, these two criteria should be correlated, too.

4. First approach for data-based analysis

As we have seen in the previous chapter the new criteria can be used to distinguish b and c hadrons. To access them the secondary vertices have to be reconstructed at least partially. In this chapter a first attempt to reconstruct theses vertices via particle tracks which are near to each other will be shown. After the reconstruction, the selection criteria are employed to reduce the contamination from non-b hadron vertices. The quality of this analysis approach is checked with the information provided by the MC-Truth. For this part the minimum bias data-set was used.

4.1. Reconstruction of b hadron candidates

The reconstruction starts by selecting electrons with a deviation from the primary vertex of at least 4σ to reduce the number of tracks which are belonging to non-b and non-c hadrons. Since many c hadrons are included in this selection the efficiency of this cut has a high error. Hence we have to use the new criteria to improve this selection. For the reconstruction the same track quality cuts as in subsection 3.3.3 were used. The electrons were identified as described in section 1.3 and the impact parameter was determined using the KF package (Code A.1, lines 1263 - 1279).

The next step was to check which particle track is the one nearest to the respective electron track and creates a vertex having a maximum distance in transverse direction of $3.9\,\mathrm{cm}$ and $10\,\mathrm{cm}$ in beam direction (Code A.1, lines 1280- 1323). These transversal cuts were employed because a b hadron is not expected to reach the ITS and the cut in the beam direction was needed to reduce the effect of fake vertices by particles created when collision particles interact with the detector material.

The last step of vertex reconstruction was to add all the particles not exceeding a deviation of 3σ from the respective reconstructed vertex. The particles were identified with the information provided by the TPC (Code A.1, lines 1290 - 1297).

The first step of improving the selection was to neglect all vertices which have less than two contributors because for reconstructing a vertex at least two particles are needed. To distinguish b hadron vertices from other vertices, it was asked for a mass of at least $2.3\,\mathrm{GeV/c^2}$. To remove fake vertices a maximum mass of $6.3\,\mathrm{GeV/c^2}$ had been introduced. To further reduce the number of fake vertices the minimum value for the lifetime was set to $900\,\mathrm{fs}$.
The quality of the electron and vertex selection were checked by comparing with the information given by the MC-Truth and Geant. The respective functions for the quality check of the electron and vertex selection are shown in the appendix, Code A.1, lines 1390 - 1408 and lines 1410 - 1471. The electron sample was checked before reducing non-b hadron contributions using the number of prongs, the lifetime and the mass criteria. The vertices were check after these selection cuts.

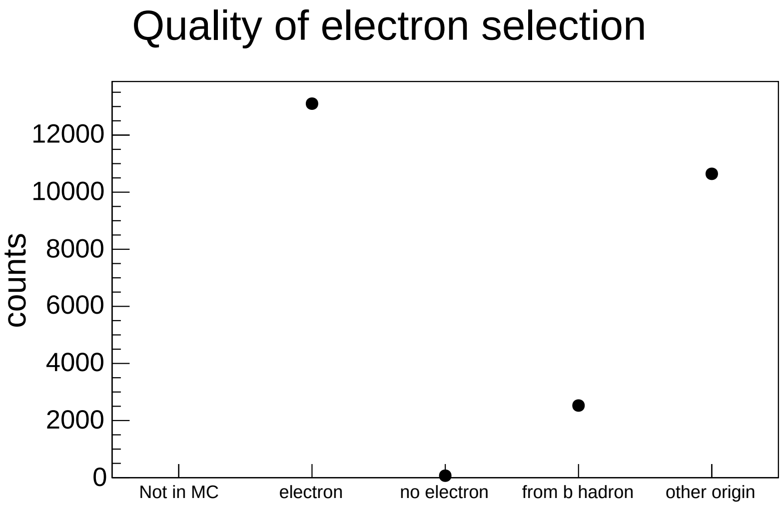


Figure 4.1.: The results of the quality check of the electron selection. The first bin shows how many electrons are not contained in the MC-Truth and therefore cannot be linked to certain collision products. The next two bins indicate how many electron candidates, which can be found in the MC-Truth, are really electrons. The last two bins reveal the origin of the real MC-Truth electrons.

Figure 4.1 shows that most of the selected electron candidates can be found in the MC-Truth and can therefore be linked to collision products. The percentage of the electrons belonging to b hadrons is lower than the one with other origin (like c hadrons or pions) because of the very loose impact parameter cut and the fact that the other selection criteria were not yet used.
Since not all electron tracks have other particle tracks in their vicinity, with which they create

a vertex being close enough to the primary vertex, fewer vertices than electrons are reconstructed. These vertices still contain many fake vertices created accidentally by particles not descending from the same particle. Therefore the additional cuts on the lifetime and the mass were important not only to reduce the share of vertices belonging to other particles than b hadrons but also to reduce the contamination by fake vertices. After introducing these cuts about 19.8 % of all vertices are left for the analysis of the quality of the classification.

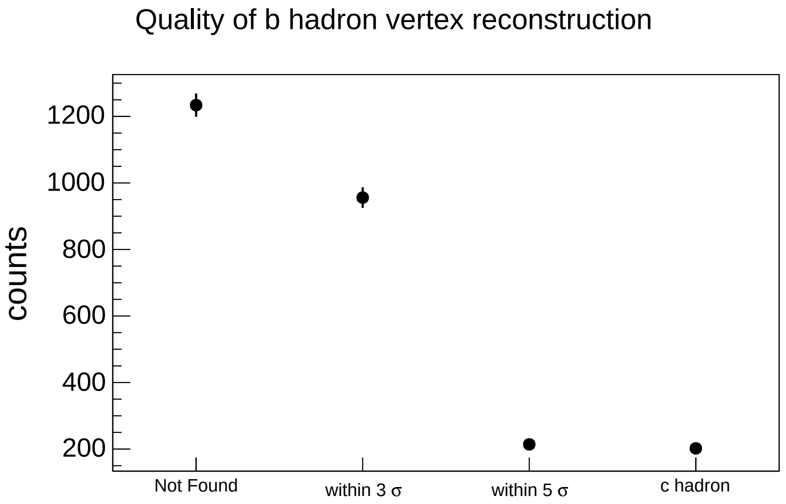


Figure 4.2.: The results of the quality check of the vertex selection. The 5σ-bin shows only the vertices which cannot be linked to a MC-Truth vertex within 3σ, but within 5σ. Within 5σ, most of the vertices cannot be linked to any vertex contained in the MC-Truth. But nearly the same amount of vertices are really belonging to b hadrons and have been found in MC at least once.

Even though the criteria for identifying reconstructed vertices with vertices known from the simulation and the vertex reconstruction method used in this part were only very simple approaches, the result shows a relatively good quality. About 44.9 % of the resulting vertex sample really belong to b hadrons, but about the same number of the vertices (47.4 %) cannot be linked to any vertex contained in the MC-Truth within 5σ. The high amount of fake tracks is caused by the very simple vertexing and will be lowered by improvements. The c hadron contamination which survived the mass cut is caused by the fact that the maximum c hadron mass, which can be reconstructed, is a little bit increased because of the error on the track energies and momenta. A higher minimum mass cut would result in a lower of efficiency and therefore was not chosen.

The reconstruction could be improved by trying to create the first guess for the vertex not only with one track nearest by the electron but trying to find, for example, two tracks which create the same vertex with the electron. A future goal could be to find an electron independent way of reconstructing secondary vertices because this would lead to a higher number of possible b hadron vertices since only about $20.5\ \%$ of the b hadrons produce electrons during their decay chain.

5. Summary and outlook

New criteria to distinguish b and c hadrons were investigated using MC simulations of pp collisions at $\sqrt{s} = 7\,\mathrm{TeV}$. These criteria were the mass, the lifetime and the number of particles contributing to a partially reconstructed secondary vertex. In addition, the impact parameter of electrons created by semi-electronic decays of b hadrons was taken into account. The analysis showed that the mass and the number of prongs can be used to create pure samples. Allowing c hadron contamination results in a higher efficiency without having significantly larger errors. The mass used as an almost pure criterion has an efficiency of $(16.7 \pm 0.5)\,\%$ and the number of prongs reaches $(4.26 \pm 0.08)\,\%$ efficiency. Tests have shown that vertices with many contributing tracks are very rare if one uses the reconstructing method described in chapter 4. Therefore the cut on the number of prongs cannot be very strong. The selections using the other criteria are less efficient and suffer from contamination by c hadrons. Hence the impact parameter can only be used to pre-select b and c hadrons. The lifetime criterion can only be used in combination with others. It can either be used after a pre-selection has reduced the number of c hadrons to about the number of b hadrons or can be used to create such a pre-selection.

The next step could be a p_T-dependent analysis instead of a p_T-integrated one and to check the influence of other background particles on the quality of the criteria. Starting from the knowledge about the feasibility of the criteria, further tests could maybe improve the way of reconstructing the secondary vertices or find the best combination of cut values for using these criteria in a mixture. An additional aspect to investigate could be trying to find more elaborated ways of reconstructing secondary vertices. Maybe kaons or pions can be used to develop additional electron independent approaches. Furthermore an efficient way of identifying reconstructed vertices with vertices contained in the MC-Truth has to be developed to get clearer information about the quality of the reconstruction.

The results from these investigations could be very useful for the determination of the b hadron yield in pp and in p-Pb. The reconstruction method could be eventually developed to study the beauty production in heavy ion collisions and its modifications due to interactions with a strongly interacting, de-confined medium, the Quark-Gluon Plasma. This would help both towards more precise test of pertubative QCD and to study the characteristics of the QGP.

A. Appendix

A.1. The analysis task code

```
void AliAnalysisTaskBeauty::UserExec(Option_t *) {
    // Main loop; Called for each event; Checking for MC
    Bool_t MC=0;
    AliMCEventHandler * mcH = dynamic_cast<AliMCEventHandler*>((↩
        AliAnalysisManager::GetAnalysisManager())->GetMCtruthEventHandler());
    if(!mcH){ } else { MC=1; }
    //__________________ ESD ____________________________________________
    // Create pointer to reconstructed event
    AliVEvent *event = InputEvent();
    if (!event) { Printf("ERROR: Could not retrieve event"); return; }
    // Get ESD event
    AliESDEvent* esd = dynamic_cast<AliESDEvent*>(event);
    if (!esd) { AliError("Cannot get the ESD event"); return; }
    Int_t ntracks = esd->GetNumberOfTracks();
    TObject* eventhandler = AliAnalysisManager::GetAnalysisManager()->↩
        GetInputEventHandler();
    if(!((AliESDInputHandler*)eventhandler)) Printf("ESD inputhandler not ↩
        available \n");
    fESDpid = ((AliESDInputHandler*)(AliAnalysisManager::GetAnalysisManager()->↩
        GetInputEventHandler()))->GetESDpid();
    if (!fESDpid){ Printf("ERROR: fESDpid not available"); return; }
    AliKFParticle::SetField(esd->GetMagneticField());
    // Primary Vertex and it's posititon
    const AliESDVertex *prima = esd->GetPrimaryVertex();
    const AliKFVertex primVtx( *(esd->GetPrimaryVertex()));
    Double_t primares2 = TMath::Power( prima->GetXRes(),2)+TMath::Power(prima->↩
        GetYRes(),2)+TMath::Power(prima->GetZRes(),2);
    if(MC) AnalysiswithMCTruth(mcH, esd, fTrackCuts, fTrackCutsElec, ntracks, ↩
        primVtx, primares2);
    AnalysiswithoutMCTruth(mcH, MC, esd, fTrackCuts, fTrackCutsElec, ntracks, ↩
        primVtx, primares2);
    // Information for this iteration of the UserExec in the container
    PostData(1, fOutput);
}
//______________________________________________________________________
void AliAnalysisTaskBeauty::AnalysiswithMCTruth(AliMCEventHandler* mcH, ↩
    AliESDEvent* esd, AliESDtrackCuts* fTrCuts, AliESDtrackCuts* fTrCutsElec, ↩
    Int_t ntracks, const AliKFVertex primVtx, Double_t primares2) {
    Double_t Bacceptance=0.9;
    // Create pointer to MC event
```

```
AliMCEvent *mcEvent = mcH->MCEvent();
if (!mcEvent) { Printf("ERROR: Could not retrieve MC event"); return; }
AliStack* stack = mcEvent->Stack();
if( !stack ) { Printf( "Stack not available"); return; }
TParticle* mcpart = 0x0;
Int_t nNumberOfMCPrimaries = stack->GetNprimary();
Double_t DecayLength=0;
Double_t DecayLengthXY=0;
TLorentzVector MassRecVec= TLorentzVector();
TNtuple* Bmespart = new TNtuple("mespart", "# of measurable particles of one↩
     b hadron", "MClabel:B number:Generation");
TNtuple* Dmespart = new TNtuple("mespart", "# of measurable particles of one↩
     b hardon", "MClabel:B number:Generation");
Int_t nProng[2]; // Counter for prongs and measurable prongs
Int_t nBmes = 0; // Counter for number of b hardon in this event
Int_t nDmes = 0; // Counter for number of c hardon in this event
Int_t gen = 0;  // Counter for decay generation

for (Int_t iCurrentLabelStack = 0; iCurrentLabelStack < nNumberOfMCPrimaries↩
    ; iCurrentLabelStack++) {
   mcpart = stack->Particle( iCurrentLabelStack );
   if (!mcpart) { printf("Stack loop %d - MC TParticle pointer to current ↩
       stack particle = 0x0 ! Skip ...\n", iCurrentLabelStack ); continue; }
   Int_t pdg = mcpart->GetPdgCode();
   Int_t abspdg = TMath::Abs(pdg);
   // Filling the b hardon histogram
   switch (pdg){
      case  521: fHistBm->Fill(1); break;
      case -521: fHistBm->Fill(-1); break;
      case  511: fHistBm->Fill(0); break;
      case -511: fHistBm->Fill(0); break;
      case  541: fHistBm->Fill(2); break;
      case -541: fHistBm->Fill(-2); break;
      case  531: fHistBm->Fill(3); break;
      case -531: fHistBm->Fill(3); break;
      case 5122: fHistBm->Fill(4); break;
      case -5122: fHistBm->Fill(4); break;
      default: break;
   }
   // B, B_s, B_c, lambda_b for prong analysis
   if ( abspdg==521 || abspdg==511 || abspdg==531 || abspdg==541 || abspdg↩
       ==5122 ) {
      fHistBEta->Fill(mcpart->Eta());
      fHistBpt->Fill(mcpart->Pt());
      nBmes++;
      MassRecVec= TLorentzVector();
      Int_t fchild = mcpart->GetFirstDaughter();
      Double_t absEta=TMath::Abs(mcpart->Eta());
      Int_t children = 0;
      if (fchild > 0) {
         TParticle *decaypoint = stack->Particle(fchild);
         if(abspdg!=541 && absEta <= Bacceptance){ // B_c not used in this ↩
             analysis and acceptance
```

```
                    MCReconstructMass(stack, iCurrentLabelStack, nNumberOfMCPrimaries, &MassRecVec );
                    if (MassRecVec != TLorentzVector()) fHistmeasMassB->Fill(MassRecVec.M());
                    DecayLength = TMath::Sqrt(TMath::Power(decaypoint->Vx()-mcpart->Vx(),2)+TMath::Power(decaypoint->Vy()-mcpart->Vy(),2)+TMath::Power(decaypoint->Vz()-mcpart->Vz(),2));
                    fDecayLengthMCB->Fill(DecayLength);
                    fLifeTimeMCB->Fill(DecayLength*0.33e5*(mcpart->GetMass()/mcpart->P()));
                    fDecayLengthMCBT->Fill(TMath::Sqrt(TMath::Power(decaypoint->Vx()-mcpart->Vx(),2)+TMath::Power(decaypoint->Vy()-mcpart->Vy(),2)));
                    nProng={0};
                    MeasurableDecays(stack, iCurrentLabelStack, nNumberOfMCPrimaries, Bmespart, nBmes, gen, nProng);
                    MeasurableDecays2(stack, iCurrentLabelStack, nNumberOfMCPrimaries, fHistBProngs, MassRecVec, fLifeTimeMeasB);
                    fHistNProngB->Fill(nProng[0]);
                    fHistmeNProngB->Fill(nProng[1]);
                }
                children = mcpart->GetNDaughters();
                for (Int_t loopchild = fchild; loopchild <= (fchild + children); loopchild++){
                    TParticle *looppart = stack->Particle(loopchild);
                    Int_t pdgchild = looppart->GetPdgCode();
                    if (TMath::Abs(pdgchild) == 11) {
                        if (abspdg!=541){
                            fHisteEta->Fill(looppart->Eta());
                            fHistept->Fill(looppart->Pt());
                            fHistCorPt->Fill(mcpart->P(), looppart->P());
                            fHistCorEta->Fill(mcpart->Eta(), looppart->Eta());
                        }
                        switch (pdg){
                            case  521: fHistBme->Fill(1); break;
                            case -521: fHistBme->Fill(-1); break;
                            case  511: fHistBme->Fill(0); break;
                            case -511: fHistBme->Fill(0); break;
                            case  541: fHistBme->Fill(2); break;
                            case -541: fHistBme->Fill(-2); break;
                            case  531: fHistBme->Fill(3); break;
                            case -531: fHistBme->Fill(3); break;
                            case 5122: fHistBme->Fill(4); break;
                            case -5122: fHistBme->Fill(4); break;
                            default: break;
                        }
                    }
                }
            }
        }
        // D, D_s, lambda_c for prong analysis
        if( abspdg == 411 || abspdg == 421 || abspdg == 431 || abspdg == 4122 ){
```

```
            Bool_t check = 0;
            check = checkmother(stack, mcpart);
            if (check == false){
                nDmes++;
                MassRecVec= TLorentzVector();
                Int_t fchild = mcpart->GetFirstDaughter();
                Int_t children=0;
                Double_t absEta=TMath::Abs(mcpart->Eta());
                if (fchild > 0) {
                    if (absEta<=Bacceptance){
                        MCReconstructMass(stack, iCurrentLabelStack, ↩
                            nNumberOfMCPrimaries, &MassRecVec );
                        if (MassRecVec != TLorentzVector()) fHistmeasMassD->Fill(↩
                            MassRecVec.M());
                        TParticle *decaypoint = stack->Particle(fchild);
                        DecayLength = TMath::Sqrt(TMath::Power(decaypoint->Vx()-↩
                            mcpart->Vx(),2)+TMath::Power(decaypoint->Vy()-mcpart->↩
                            Vy(),2)+TMath::Power(decaypoint->Vz()-mcpart->Vz(),2));
                        fDecayLengthMCD->Fill(DecayLength);
                        fLifeTimeMCD->Fill(DecayLength*0.33e5*(mcpart->GetMass()/↩
                            mcpart->P()));
                        fDecayLengthMCDT->Fill(TMath::Sqrt(TMath::Power(decaypoint↩
                            ->Vx()-mcpart->Vx(),2)+TMath::Power(decaypoint->Vy()-↩
                            mcpart->Vy(),2)));
                        nProng = {0};
                        MeasurableDecays(stack, iCurrentLabelStack, ↩
                            nNumberOfMCPrimaries, Dmespart, nDmes, gen, nProng);
                        MeasurableDecays2(stack, iCurrentLabelStack, ↩
                            nNumberOfMCPrimaries, fHistDProngs, MassRecVec, ↩
                            fLifeTimeMeasD);
                        fHistNProngD->Fill(nProng[0]);
                        fHistmeNProngD->Fill(nProng[1]);
                    }
                    children = mcpart->GetNDaughters();
                    for (Int_t loopchild = fchild; loopchild <= (fchild + children)↩
                        ; loopchild++){
                        TParticle *looppart = stack->Particle(loopchild);
                        Int_t pdgchild = looppart->GetPdgCode();
                        if (TMath::Abs(pdgchild) == 11) {
                            fHistCorPtD->Fill( mcpart->P(), looppart->P() );
                            fHistCorEtaD->Fill(mcpart->Eta(), looppart->Eta());
                        }
                    }
                }
            }
        }
    }
    // List for the hadrons
    Int_t* flistB = new Int_t[nBmes]();
    Int_t* flistD = new Int_t[nDmes]();
    Int_t fillB = 0;
    Int_t fillD = 0;
```

```
for (Int_t iCurrentLabelStack1 = 0; iCurrentLabelStack1 < nNumberOfMCPrimaries; iCurrentLabelStack1++) {
    mcpart = stack->Particle( iCurrentLabelStack1 );
    if (!mcpart) { printf("Stack loop %d - MC TParticle pointer to current stack particle = 0x0 ! Skip ...\n", iCurrentLabelStack1 ); continue; }
    Int_t abspdg = TMath::Abs(mcpart->GetPdgCode());
    Double_t absEta = TMath::Abs(mcpart->Eta());
    if( absEta<=Bacceptance){
        if ( abspdg == 521 || abspdg == 511 || abspdg == 531 || abspdg==5122 ) {
            flistB[fillB]= iCurrentLabelStack1;
            fillB++; continue;
        }
        if( abspdg == 411 || abspdg == 421 || abspdg == 431 || abspdg == 4122){
            Bool_t check = checkmother(stack, mcpart);
            if (check == false){
                flistD[fillD]= iCurrentLabelStack1;
                fillD++;
            }
        }
    }
}
nBmes=fillB;
nDmes=fillD;
//__________________ ESD ______________________________________________
Float_t DCA[2] = {}; // DCA[0] is radial
Float_t errDCA[3] = {}; // errDCA[0] radial error, errDCA[2] in z direction
Double_t elecres2 = 0;
Int_t* NESDProngB = new Int_t[nBmes]();
Int_t* NESDProngD = new Int_t[nDmes]();
Int_t* fGProngB = new Int_t[nBmes]();
Int_t* fGProngD = new Int_t[nDmes]();
Double_t* ImPar = new Double_t[nBmes]();
TLorentzVector* VecB = new TLorentzVector[nBmes]();
TLorentzVector* VecD = new TLorentzVector[nDmes]();
TLorentzVector* VecBall = new TLorentzVector[nBmes]();
TLorentzVector* VecDall = new TLorentzVector[nDmes]();
AliKFParticle *DecayB1 = new AliKFParticle[nBmes]();
AliKFParticle *DecayD1 = new AliKFParticle[nDmes]();
AliKFParticle *DecayB = new AliKFParticle[nBmes]();
AliKFParticle *DecayD = new AliKFParticle[nDmes]();

for(Int_t i = 0; i < ntracks; i++) {
    AliESDtrack* esdtrack = esd->GetTrack(i); // Pointer to reconstructed track
    if(!esdtrack) {
        AliError(Form("ERROR: Could not retrieve esdtrack %d",i)); continue; }
    Bool_t goodESD = fTrCuts->AcceptTrack(esdtrack);
    Bool_t goodElec = 1;
    if (goodESD){
        // Read the label
        Int_t lab=TMath::Abs(esdtrack->GetLabel());
        // Searching for electrons
```

```
if (TMath::Abs(fESDpid->NumberOfSigmasTOF(esdtrack, AliPID::kElectron))<3. && fESDpid->NumberOfSigmasTPC(esdtrack,AliPID::kElectron)>0. && fESDpid->NumberOfSigmasTPC(esdtrack,AliPID::kElectron)<3 ){
    goodElec = fTrCutsElec->AcceptTrack(esdtrack);
    if (goodElec){
        esdtrack->GetImpactParameters(DCA, errDCA);
        fDCAe->Fill(esdtrack->P(), TMath::Sqrt(TMath::Power(DCA[0],2)+TMath::Power(DCA[1],2)));
        AliKFParticle Elec (*esdtrack, stack->Particle(lab)->GetPdgCode());
        fDCAeKF->Fill(Elec.GetP(), Elec.GetDistanceFromVertex(primVtx));
    }
}
// Checking for MC-Truth measurables
for (Int_t ilab=0; ilab < Bmespart->GetEntries(); ilab++) {
    Bmespart->GetEntry(ilab);
    Float_t* row_cont = Bmespart->GetArgs();
    if (lab == row_cont [0]){ // Measurable b hadron daughter
        Int_t pdgB = stack->Particle(lab)->GetPdgCode();
        Int_t abspdgB = TMath::Abs(pdgB);
        Double_t DummyImPar = 0;
        if (abspdgB == 11){
            goodElec = fTrCutsElec->AcceptTrack(esdtrack);
            if (!goodElec) continue;
            esdtrack->GetImpactParameters(DCA, errDCA);
            fDCAeESDB->Fill(TMath::Sqrt(TMath::Power(DCA[0],2)+TMath::Power(DCA[1],2)));
            elecres2 = TMath::Power(errDCA[0],2)+TMath::Power(errDCA[2],2);
            fDCAeESDBerr->Fill( TMath::Sqrt(TMath::Power(DCA[0],2)+TMath::Power(DCA[1],2))/TMath::Sqrt(primares2+elecres2));
            AliKFParticle ElecB (*esdtrack, pdgB);
            fDCAeKFB->Fill(ElecB.GetDistanceFromVertex(primVtx));
            fDCAeKFBT->Fill(ElecB.GetDistanceFromVertexXY(primVtx));
            DummyImPar = ElecB.GetDeviationFromVertex(primVtx);
            fDCAeKFBerr->Fill(DummyImPar);
            fDCAeKFBTerr->Fill(ElecB.GetDeviationFromVertexXY(primVtx) );
        }
        for (Int_t imes=0; imes<nBmes; imes++){
            if(imes+1 == row_cont[1]){ // Which b hardon
                if(DummyImPar!= 0 ){ ImPar[imes]= DummyImPar; }
                if (row_cont[2]==1){ // Only direct prongs
                    Int_t PDG = stack->Particle(lab)->GetPdgCode();
                    if (PDG == 11 || PDG == -11 || PDG==13 || PDG==-13) goodElec = fTrCutsElec->AcceptTrack(esdtrack);
                    if (!goodElec) continue;
                    VecB[imes] += TLorentzVector(stack->Particle(lab)->Px(), stack->Particle(lab)->Py(),stack->Particle(lab)->Pz(),stack->Particle(lab)->Energy());
                    DecayB1[imes].AddDaughter(AliKFParticle(*esdtrack, PDG));
                    fGProngB[imes]++;
```

```
                        }
                        NESDProngB[imes]++; break;
                    }
                }
                if (row_cont[2]==1){
                    switch (abspdgB) {
                        case 11: fHistESDdprongB->Fill(1); break;
                        case 13: fHistESDdprongB->Fill(2); break;
                        case 211: fHistESDdprongB->Fill(3); break;
                        case 321: fHistESDdprongB->Fill(4); break;
                        case 2212: fHistESDdprongB->Fill(5); break;
                        default: break;
                    }
                } break;
            }
        }
        // Checking for MC-Truth D measurables
        for (Int_t ilab=0; ilab < Dmespart->GetEntries(); ilab++) {
            Dmespart->GetEntry(ilab);
            Float_t* row_cont = Dmespart->GetArgs();
            if (lab == row_cont [0]){
                Int_t pdgD = stack->Particle(lab)->GetPdgCode();
                Int_t abspdgD = TMath::Abs(pdgD);
                if (abspdgD == 11){
                    goodElec = fTrCutsElec->AcceptTrack(esdtrack);
                    if (!goodElec) continue;
                    esdtrack->GetImpactParameters(DCA, errDCA);
                    fDCAeESDD->Fill( TMath::Sqrt(TMath::Power(DCA[0],2)+TMath::↩
                        Power(DCA[1],2)) );
                    elecres2 = TMath::Power(errDCA[0],2)+TMath::Power(errDCA↩
                        [2],2);
                    fDCAeESDDerr->Fill( TMath::Sqrt(TMath::Power(DCA[0],2)+↩
                        TMath::Power(DCA[1],2))/TMath::Sqrt(primares2+elecres2)↩
                        );
                    AliKFParticle ElecD (*esdtrack, pdgD);
                    fDCAeKFD->Fill(ElecD.GetDistanceFromVertex(primVtx));
                    fDCAeKFDT->Fill(ElecD.GetDistanceFromVertexXY(primVtx));
                    fDCAeKFDerr->Fill( ElecD.GetDeviationFromVertex(primVtx) );
                    fDCAeKFDTerr->Fill(ElecD.GetDeviationFromVertexXY(primVtx) );
                }
                for (Int_t imes=0; imes<nDmes; imes++){
                    if(imes+1 == row_cont[1]){
                        if (row_cont[2]==1){ // Only direct prongs
                            Int_t PDG = stack->Particle(lab)->GetPdgCode();
                            if (PDG == 11 || PDG == -11|| PDG==13 || PDG==-13) ↩
                                goodElec = fTrCutsElec->AcceptTrack(esdtrack);
                            if (!goodElec) continue;
                            VecD[imes] += TLorentzVector(stack->Particle(lab)->↩
                                Px(), stack->Particle(lab)->Py(),stack->Particle↩
                                (lab)->Pz(),stack->Particle(lab)->Energy());
                            DecayD1[imes].AddDaughter(AliKFParticle(*esdtrack, ↩
                                PDG));
                            fGProngD[imes]++;
```

```
                        }
                        NESDProngD[imes]++; break;
                    }
                }
                if (row_cont[2]==1){
                    switch (abspdgD) {
                        case 11: fHistESDdprongD->Fill(1); break;
                        case 13: fHistESDdprongD->Fill(2); break;
                        case 211: fHistESDdprongD->Fill(3); break;
                        case 321: fHistESDdprongD->Fill(4); break;
                        case 2212: fHistESDdprongD->Fill(5); break;
                        default: break;
                    }
                } break;
            }
        }
        esdtrack->GetImpactParameters(DCA, errDCA);
        elecres2 = TMath::Power(errDCA[0],2)+TMath::Power(errDCA[2],2);
        fHistImPar->Fill(esdtrack->P() , TMath::Sqrt(TMath::Power(DCA[0],2)+↩
            TMath::Power(DCA[1],2))/TMath::Sqrt(primares2+elecres2));
    }
}
// Full mass reconstruction via ESD and KF
KFReconstructMass(stack, esd, fTrCuts, fTrCutsElec, flistB, nBmes, DecayB, ↩
    VecBall );
KFReconstructMass(stack, esd, fTrCuts, fTrCutsElec, flistD, nDmes, DecayD, ↩
    VecDall );
Double_t Mesres2=0;
Double_t MesresT2=0;
Double_t mass=0;
Double_t momentum=0;
Double_t transmom=0;
Double_t Lifetime=0;
for (Int_t imes=0; imes<nBmes; imes++){
    if(fGProngB[imes]>=2){
        fHistKFmassB1Gen->Fill(DecayB1[imes].GetMass());
        fHistinvmassB->Fill(VecB[imes].M());
    }
    fHistNESDprongB->Fill(NESDProngB[imes]);
    mass=VecBall[imes].M();
    momentum = DecayB[imes].GetMomentum();
    transmom = DecayB[imes].GetPt();
    if (mass != 0 ) fHistKFmassB->Fill(mass);
    Mesres2 = TMath::Power(DecayB[imes].GetErrX(),2)+TMath::Power(DecayB[imes↩
        ].GetErrY(),2)+TMath::Power(DecayB[imes].GetErrZ(),2);
    MesresT2 = TMath::Power(DecayB[imes].GetErrX(),2)+TMath::Power(DecayB[↩
        imes].GetErrY(),2);
    DecayLength = TMath::Sqrt(TMath::Power(DecayB[imes].X()-primVtx.X(),2)+ ↩
        TMath::Power(DecayB[imes].Y()-primVtx.Y(),2)+ TMath::Power(DecayB[↩
        imes].Z()-primVtx.Z(),2));
    if(DecayLength>0){
        fHistKFdistB->Fill(DecayLength);
        fKFdistBerr->Fill(DecayLength/TMath::Sqrt(primares2+Mesres2));
```

```
            if(mass!=0 && momentum!=0){
                Lifetime=DecayLength*0.33e5*mass/momentum;
                fLifeTimeKFB->Fill(Lifetime);
                fCorLifPt->Fill(transmom, Lifetime);
                fCorLifM->Fill(Lifetime,mass);
                fCorLifNPro->Fill(Lifetime,NESDProngB[imes]);
                fCorLifImPar->Fill(Lifetime,ImPar[imes]);
            }
        }
        DecayLengthXY = TMath::Sqrt( TMath::Power(DecayB[imes].X()-primVtx.X(),2)↩
            + TMath::Power(DecayB[imes].Y()-primVtx.Y(),2));
        if(DecayLengthXY>0){
            fHistKFdistBT->Fill(DecayLengthXY);
            fKFdistBerrT->Fill(DecayLengthXY/TMath::Sqrt(primares2+MesresT2));
        }
        if (VecBall[imes]!= TLorentzVector()){
            ftotalinvmassB->Fill(mass);
            fCorMNPro->Fill(NESDProngB[imes], mass);
            fCorMImPar->Fill(mass, ImPar[imes]);
            fCorMPt->Fill(transmom, mass);
        }
        if (NESDProngB[imes]!=0) fCorNProImPar->Fill(NESDProngB[imes], ImPar[imes↩
            ]);
        if (transmom!=0){
            fCorImPt->Fill(transmom, ImPar[imes]);
            fCorProPt->Fill(transmom, NESDProngB[imes]);
        }
    }

    for (Int_t imes=0; imes<nDmes; imes++) {
        if(fGProngD[imes]>=2){
            fHistKFmassD1Gen->Fill(DecayD1[imes].GetMass());
            fHistinvmassD->Fill(VecD[imes].M());
        }
        fHistNESDprongD->Fill(NESDProngD[imes]);
        mass=DecayD[imes].GetMass();
        momentum = DecayD[imes].GetMomentum();
        if (mass != 0 ) fHistKFmassD->Fill(mass);
        Mesres2 = TMath::Power(DecayD[imes].GetErrX(),2)+TMath::Power(DecayD[imes↩
            ].GetErrY(),2)+TMath::Power(DecayD[imes].GetErrZ(),2);
        MesresT2 = TMath::Power(DecayD[imes].GetErrX(),2)+TMath::Power(DecayD[↩
            imes].GetErrY(),2);
        DecayLength = TMath::Sqrt( TMath::Power(DecayD[imes].X()-primVtx.X(),2)+ ↩
            TMath::Power(DecayD[imes].Y()-primVtx.Y(),2)+ TMath::Power(DecayD[↩
            imes].Z()-primVtx.Z(),2));
        if(DecayLength>0){
            fHistKFdistD->Fill(DecayLength);
            fKFdistDerr->Fill(DecayLength/TMath::Sqrt(primares2+Mesres2));
                if(mass!=0 && momentum!=0) fLifeTimeKFD->Fill(DecayLength*0.33e5*↩
                    mass/momentum);
            }
        DecayLengthXY = TMath::Sqrt( TMath::Power(DecayD[imes].X()-primVtx.X(),2)↩
            + TMath::Power(DecayD[imes].Y()-primVtx.Y(),2));
```

```
        if(DecayLengthXY>0){
            fHistKFdistDT->Fill(DecayLengthXY);
            fKFdistDerrT->Fill(DecayLengthXY/TMath::Sqrt(primares2+Mesres2) );
        }
        if (VecDall[imes]!= TLorentzVector()) ftotalinvmassD->Fill(VecDall[imes].↩
            M());
    }
    delete Bmespart;
    delete Dmespart;
    delete[] NESDProngB;
    delete[] NESDProngD;
    delete[] fGProngB;
    delete[] fGProngD;
    delete[] ImPar;
    delete[] VecB;
    delete[] VecD;
    delete[] VecBall;
    delete[] VecDall;
    delete[] DecayB1;
    delete[] DecayB;
    delete[] DecayD1;
    delete[] DecayD;
    delete[] flistB;
    delete[] flistD;
}
//_________________________________________________________________________
void AliAnalysisTaskBeauty::AnalysiswithoutMCTruth(AliMCEventHandler* mcH, ↩
    Bool_t MC, AliESDEvent *esd, AliESDtrackCuts *fTrCuts, AliESDtrackCuts* ↩
    fTrCutsElec, Int_t ntracks, const AliKFVertex primVtx, Double_t primares2) {
    // Values for selection criteria cut
    const Double_t Imparcut = 4;
    const Double_t masscut = 2.3;
    const Int_t prongcut = 2;
    const Double_t taucut = 900;
    const Int_t arraysize = 100;
    Double_t DecayLength=0;
    AliKFParticle *Belec = new AliKFParticle[arraysize]();
    Int_t ESDtoelec[arraysize]= {};
    Double_t ClosestToElec[arraysize][3] = {{0}};
    Int_t nelec = 0;
    AliKFParticle part = AliKFParticle();
    Double_t DCAtoElec; // Dummy for DCA to elec (3D, unit: primary vert. ↩
        resoultion)
    Int_t PID=0;
    Double_t distPVXY=0;
    Double_t distPVZ=0;
    // Find electron exceeding the impact parameter cut
    for(Int_t i = 0; i < ntracks; i++) {
        AliESDtrack* esdtrack = esd->GetTrack(i);
        if(!esdtrack) { AliError(Form("ERROR: Could not retrieve esdtrack %d",i))↩
            ; continue; }
        Bool_t goodElec = fTrCutsElec->AcceptTrack(esdtrack);
        if (goodElec){
```

```
            // Searching for electrons
            if (TMath::Abs(fESDpid->NumberOfSigmasTOF(esdtrack, AliPID::kElectron↩
                ))<3. && fESDpid->NumberOfSigmasTPC(esdtrack,AliPID::kElectron)↩
                >=0. && fESDpid->NumberOfSigmasTPC(esdtrack,AliPID::kElectron)<3 )↩
                {
                AliKFParticle Elec (*esdtrack, 11);
                if( TMath::Abs(Elec.GetDeviationFromVertex(primVtx)) >= Imparcut){
                    Belec[nelec]= Elec;
                    ESDtoelec[nelec]= i;
                    ClosestToElec[nelec][1]=100;
                    nelec++;
                }
            }
        }
    }
    if (nelec!=0){
        // Find closest track to the electron
        for(Int_t i = 0; i < ntracks; i++) {
            AliESDtrack* esdtrack = esd->GetTrack(i);
            if(!esdtrack) { AliError(Form("ERROR: Could not retrieve esdtrack %d"↩
                ,i)); continue; }
            Bool_t goodESD = fTrCuts->AcceptTrack(esdtrack);
            Bool_t goodElec = 1;
            if (goodESD){
                for (Int_t ielec=0; ielec<nelec; ielec++){
                    if (ESDtoelec[ielec] == i) continue; // Skipping the electron ↩
                        itself
                    if (TMath::Abs(fESDpid->NumberOfSigmasTOF(esdtrack, AliPID::↩
                        kElectron))<3. && fESDpid->NumberOfSigmasTPC(esdtrack,↩
                        AliPID::kElectron)>=0. && fESDpid->NumberOfSigmasTPC(↩
                        esdtrack,AliPID::kElectron)<3 ) { PID = 11; goodElec = ↩
                        fTrCutsElec->AcceptTrack(esdtrack);}
                    else if (TMath::Abs(fESDpid->NumberOfSigmasTPC(esdtrack, AliPID↩
                        ::kKaon))<3.){
                        if (esdtrack->P() <= 0.7){ PID = 321;
                        } else PID = 211;
                    } else if (TMath::Abs(fESDpid->NumberOfSigmasTPC(esdtrack, ↩
                        AliPID::kProton))<3.){
                        if (esdtrack->P() <= 1.1){ PID = 2212;
                        } else PID = 211;
                    } else { PID = 211; }
                    if (!goodElec) continue;
                    part = AliKFParticle (*esdtrack, PID);
                    DCAtoElec = part.GetDeviationFromParticle(Belec[ielec]);
                    if (DCAtoElec < ClosestToElec[ielec][1]){
                        part += Belec[ielec];
                        distPVXY=TMath::Sqrt(TMath::Power(part.X()-primVtx.X(),2)+ ↩
                            TMath::Power(part.Y()-primVtx.Y(),2));
                        distPVZ=TMath::Abs(part.Z()-primVtx.Z());
                        if (distPVXY<3.9 && distPVZ<10){
                            ClosestToElec[ielec][0]= i;
                            ClosestToElec[ielec][1]= DCAtoElec;
                            ClosestToElec[ielec][2]= PID;
```

```
                }
            }
        }
    }
}
AliKFVertex *BVertex = new AliKFVertex[nelec]();
Double_t *Vertres2 = new Double_t[nelec]();
for (Int_t ielec=0; ielec<nelec; ielec++){
    if (ClosestToElec[ielec][1] == 100) continue;
    PID = ClosestToElec[ielec][2];
    part = AliKFParticle(*(esd->GetTrack( ClosestToElec[ielec][0])), PID);
    BVertex[ielec] += Belec[ielec];
    BVertex[ielec] += part;
    Vertres2[ielec] = TMath::Power(BVertex[ielec].GetErrX(),2)+TMath::↩
        Power(BVertex[ielec].GetErrY(),2)+TMath::Power(BVertex[ielec].↩
        GetErrZ(),2);
}
// Add Tracks close to Vertex
for(Int_t i = 0; i < ntracks; i++) {
    AliESDtrack* esdtrack = esd->GetTrack(i);
    if(!esdtrack) { AliError(Form("ERROR: Could not retrieve esdtrack %d"↩
        ,i)); continue; }
    Bool_t goodESD = fTrCuts->AcceptTrack(esdtrack);
    Bool_t goodElec = 1;
    if (goodESD){
        for (Int_t ielec=0; ielec<nelec; ielec++){
            if (ESDtoelec[ielec] == i || ClosestToElec[ielec][0]== i|| ↩
                ClosestToElec[ielec][1] == 100) continue; // Skipping the ↩
                electron itself and closest and no-closest found
            if (TMath::Abs(fESDpid->NumberOfSigmasTOF(esdtrack, AliPID::↩
                kElectron))<3. && fESDpid->NumberOfSigmasTPC(esdtrack,↩
                AliPID::kElectron)>=0. && fESDpid->NumberOfSigmasTPC(↩
                esdtrack,AliPID::kElectron)<3 ) { PID = 11; goodElec = ↩
                fTrCutsElec->AcceptTrack(esdtrack);}
            else if (TMath::Abs(fESDpid->NumberOfSigmasTPC(esdtrack, AliPID↩
                ::kKaon))<3.){
                if (esdtrack->P() <= 0.7){ PID = 321;
                } else PID = 211;
            } else if (TMath::Abs(fESDpid->NumberOfSigmasTPC(esdtrack, ↩
                AliPID::kProton))<3.){
                if (esdtrack->P() <= 1.1){ PID = 2212;
                } else PID = 211;
            } else { PID = 211; }
            if (!goodElec) continue;
            part = AliKFParticle (*esdtrack, PID);
            DCAtoElec = part.GetDeviationFromVertex(BVertex[ielec]);
            if (DCAtoElec <= 3 ){
                BVertex[ielec] += part;
            }
        }
    }
}
// Results
```

```
        Double_t Mesres2 = 0;
        for (Int_t ielec=0; ielec<nelec; ielec++){
            Double_t mass=BVertex[ielec].GetMass();
            Double_t momentum=BVertex[ielec].GetMomentum();
            Double_t lifetime=0;
            if (BVertex[ielec].GetNContributors()>1){
                Mesres2 = TMath::Power(BVertex[ielec].GetErrX(),2)+TMath::Power(↩
                    BVertex[ielec].GetErrY(),2)+TMath::Power(BVertex[ielec].↩
                    GetErrZ(),2);
                DecayLength = TMath::Sqrt( TMath::Power(BVertex[ielec].X()-primVtx↩
                    .X(),2)+ TMath::Power(BVertex[ielec].Y()-primVtx.Y(),2)+ TMath↩
                    ::Power(BVertex[ielec].Z()-primVtx.Z(),2));
                fexMCDecayLength->Fill(DecayLength/TMath::Sqrt(primares2+Mesres2));
            }
            if (BVertex[ielec].GetNContributors()>1){
                if(mass!=0 && momentum!=0){
                    lifetime = DecayLength*0.33e5*mass/momentum;
                    fexMCLifetime->Fill(lifetime);
                }
                fexMCMass->Fill(mass);
                fexMCNProngs->Fill(BVertex[ielec].GetNContributors());
                if(mass!=0 && momentum!=0) lifetime = DecayLength*0.33e5*mass/↩
                    momentum;
                if(mass<masscut || mass>6.3) BVertex[ielec] = AliKFVertex();
                if(BVertex[ielec].GetNContributors()< prongcut) BVertex[ielec] = ↩
                    AliKFVertex();
                if(lifetime < taucut) BVertex[ielec] = AliKFVertex();
            }
        }
        if (MC){
            // Create pointer to MC event
            AliMCEvent *mcEvent = mcH->MCEvent();
            if (!mcEvent) { Printf("ERROR: Could not retrieve MC event"); return; }
            // Set up a stack for use in quality check
            AliStack* stack = mcEvent->Stack();
            if( !stack ) { Printf( "Stack not available"); return; }
            checkelectron(stack, esd, nelec, ESDtoelec);
            checkBvertex(stack, nelec, BVertex);
        }
        delete[] BVertex;
        delete[] Vertres2;
    }
    delete[] Belec;
}
//______________________________________________________________________
void AliAnalysisTaskBeauty::checkelectron(AliStack* stack, AliESDEvent *esd, ↩
    Int_t maxelec, Int_t ESDelec[]) {
    TParticle* mcpart=0x0;
    Int_t eleclabel=0;

    for (Int_t ielec=0; ielec<maxelec; ielec++){
        AliESDtrack* esdtrack = esd->GetTrack(ESDelec[ielec]);
        eleclabel = TMath::Abs(esdtrack->GetLabel());
```

```
        if (!eleclabel){ fexMCeSelec->Fill(0);
        } else{
            mcpart = stack->Particle( eleclabel );
            Int_t abspdg = TMath::Abs(mcpart->GetPdgCode());
            if (abspdg==11){ fexMCeSelec->Fill(1);
            } else{ fexMCeSelec->Fill(2); }
            Bool_t Bmoth = checkmother(stack, mcpart);
            if (Bmoth){ fexMCeSelec->Fill(3);
            } else{ fexMCeSelec->Fill(4); }
        }
    }
}
//_____________________________________________________________________
void AliAnalysisTaskBeauty::checkBvertex(AliStack* stack, Int_t maxelec, ↩
    AliKFVertex Vertex[]) {
    TVector3* RecVertex = new TVector3 [maxelec]();
    TVector3* ErrRecVertex = new TVector3 [maxelec]();
    const Int_t maxB = 500;
    Int_t countB = 0;
    Int_t countD = 0;
    TVector3 MCVert[maxB] = {};
    TVector3 MCD[maxB] = {};
    Int_t* Found = new Int_t[maxelec]();
    for (Int_t ielec=0; ielec<maxelec; ielec++){
        if (Vertex[ielec].GetNContributors()>1){
            RecVertex[ielec].SetXYZ(Vertex[ielec].GetX(), Vertex[ielec].GetY(), ↩
                Vertex[ielec].GetZ());
             ErrRecVertex[ielec].SetXYZ(Vertex[ielec].GetErrX(), Vertex[ielec].↩
                 GetErrY(), Vertex[ielec].GetErrZ());
        }
    }
    TParticle* mcpart = 0x0;
    Int_t nNumberOfMCPrimaries = stack->GetNprimary();
    for (Int_t iCurrentLabelStack1 = 0; iCurrentLabelStack1 < ↩
        nNumberOfMCPrimaries; iCurrentLabelStack1++) {
        mcpart = stack->Particle( iCurrentLabelStack1 );
        if (!mcpart) { printf("Stack loop %d - MC TParticle pointer to current ↩
            stack particle = 0x0 ! Skip ...\n", iCurrentLabelStack1 ); continue; ↩
            }
        Int_t abspdg = TMath::Abs(mcpart->GetPdgCode());
        if ( abspdg == 521 || abspdg == 511 || abspdg == 531 || abspdg == 541 || ↩
            abspdg==5122 ) {
            Int_t fdaughter = mcpart->GetFirstDaughter();
            if (fdaughter>0){
                TParticle *fdau = stack->Particle(fdaughter);
                if (countB < maxB){
                    MCVert[countB].SetXYZ(fdau->Vx(), fdau->Vy(), fdau->Vz());
                    countB++;
                }
            }
        }
        if ( abspdg == 421 || abspdg == 411 || abspdg == 431 || abspdg==4122 ) {
            Int_t fdaughter = mcpart->GetFirstDaughter();
```

```
            if (fdaughter > 0){
                if (!checkmother(stack,mcpart)){
                    TParticle *fdau = stack->Particle(fdaughter);
                    if (countD < maxB){
                        MCD[countD].SetXYZ(fdau->Vx(), fdau->Vy(), fdau->Vz());
                        countD++;
                    }
                }
            }
        }
    }
    for (Int_t ielec=0; ielec<maxelec; ielec++){
        if (Vertex[ielec].GetNContributors()>1){
            for (Int_t iBmes=0; iBmes <= countB; iBmes++){
                if( (MCVert[iBmes]-RecVertex[ielec]).X()<= 3* ErrRecVertex[ielec].↩
                    X() && (MCVert[iBmes]-RecVertex[ielec]).Y()<= 3* ErrRecVertex[↩
                    ielec].Y() && (MCVert[iBmes]-RecVertex[ielec]).Z()<= 3* ↩
                    ErrRecVertex[ielec].Z() ){ Found[ielec]=1;}
                else if( (MCVert[iBmes]-RecVertex[ielec]).X()<= 5* ErrRecVertex[↩
                    ielec].X() && (MCVert[iBmes]-RecVertex[ielec]).Y()<= 5* ↩
                    ErrRecVertex[ielec].Y() && (MCVert[iBmes]-RecVertex[ielec]).Z↩
                    ()<= 5* ErrRecVertex[ielec].Z() ){ Found[ielec]=2; }
            }
            for (Int_t iDmes=0; iDmes <= countD; iDmes++){
                if( Found[ielec]==0 && (MCD[iDmes]-RecVertex[ielec]).X()<= 3* ↩
                    ErrRecVertex[ielec].X() && (MCD[iDmes]-RecVertex[ielec]).Y()<=↩
                     3* ErrRecVertex[ielec].Y() && (MCD[iDmes]-RecVertex[ielec]).Z↩
                    ()<= 3* ErrRecVertex[ielec].Z() ){
                    if (!Found[ielec]){ Found[ielec]=3; }
                }
            }
            fexMCBVert->Fill(Found[ielec]);
        }
    }
    delete[] RecVertex;
    delete[] ErrRecVertex;
    delete[] Found;
}
//___________________________________________________________________
void AliAnalysisTaskBeauty::MeasurableDecays( AliStack *sta, Int_t iptc, Int_t ↩
    nPrime, TNtuple* fmpart, Int_t Nmes, Int_t ge ,Int_t* npro ) {
    TParticle * part = sta->Particle(iptc);
    Int_t fdaughter = part->GetFirstDaughter();
    Int_t children = 0;

    if (fdaughter > 0){
        children = part->GetNDaughters();
        npro[0] += children;
        ge++;

        for ( Int_t loopdaughter = fdaughter; loopdaughter < (fdaughter+children)↩
            ; loopdaughter++){
            if (loopdaughter <= nPrime) {
```

```
                TParticle * looppart = sta->Particle(loopdaughter);
                Int_t abslooppdg = TMath::Abs(looppart->GetPdgCode());
                // Choice via pdg number
                if ( abslooppdg==11 || abslooppdg==13 || abslooppdg== 211 || abslooppdg==321 || abslooppdg==2212 ){
                    // Checking for electrons, muons, charged pions, charged kaons, protons
                    if (looppart->Pt()>= 0.1 && looppart->Pt()<=50 ){
                        if (TMath::Abs(looppart->Eta()) <= 0.8 ){
                            fmpart->Fill(loopdaughter, Nmes, ge);
                            npro[1]++;
                        }
                    }
                }
                MeasurableDecays(sta, loopdaughter, nPrime, fmpart, Nmes, ge, npro);
            }
        } ge--;
    }
}
//______________________________________________________________________
void AliAnalysisTaskBeauty::MeasurableDecays2( AliStack *sta, Int_t iptc, Int_t nPrime, TH1D *histprong, TLorentzVector MassVec, TH1D *fLifeTime ) {
    TParticle *part = sta->Particle(iptc);
    Int_t fdaughter = part->GetFirstDaughter();
    Int_t children = 0;
    Bool_t taudone = 0;
    Double_t DecayLength=0;
    Double_t mass = part->GetMass();
    Double_t momentum = part->P();
    if (MassVec != TLorentzVector()){
        mass = MassVec.M();
        momentum = MassVec.P();
    }
    if (fdaughter >0){
        children = part->GetNDaughters();
        for ( Int_t loopdaughter = fdaughter; loopdaughter < (fdaughter+children); loopdaughter++){
            if (loopdaughter <= nPrime) {
                TParticle * looppart = sta->Particle(loopdaughter);
                Int_t abslooppdg = TMath::Abs(looppart->GetPdgCode());
                if (looppart->Pt()>= 0.1 && looppart->Pt()<=50 ){
                    if (TMath::Abs(looppart->Eta()) <= 0.8 ){
                        if (!taudone){
                            taudone=1;
                            DecayLength = TMath::Sqrt(TMath::Power(looppart->Vx()-part->Vx(),2)+TMath::Power(looppart->Vy()-part->Vy(),2)+TMath::Power(looppart->Vz()-part->Vz(),2));
                            fLifeTime->Fill(DecayLength * mass/momentum *0.33e5);
                        }
                        // Checking for electrons, muons, charged pions, charged kaons, photons, protons and D mesons
                        switch (abslooppdg) {
                            case 11: histprong->Fill(1); break;
```

```
                            case 13:  histprong->Fill(2); break;
                            case 211: histprong->Fill(3); break;
                            case 321: histprong->Fill(4); break;
                            case 2212: histprong->Fill(5); break;
                            case 411: histprong->Fill(6); break;
                            case 421: histprong->Fill(6); break;
                            case 431: histprong->Fill(6); break;
                            default: break;
                        }
                    }
                }
            }
        }
    }
}
//______________________________________________________________________
void AliAnalysisTaskBeauty::MCReconstructMass(AliStack *st, Int_t iptc, Int_t nPrime, TLorentzVector *Vec ) {
    TParticle * part = st->Particle(iptc);
    Int_t fdaughter = part->GetFirstDaughter();
    Int_t children = 0;
    Int_t nadd = 0;
    TLorentzVector AddVec=TLorentzVector();

    if (fdaughter>0){
        children = part->GetNDaughters();
        for ( Int_t loopdaughter = fdaughter; loopdaughter < (fdaughter+children); loopdaughter++){
            if (loopdaughter < nPrime) {
                TParticle * looppart = st->Particle(loopdaughter);
                Int_t abslooppdg = TMath::Abs(looppart->GetPdgCode());
                // Checking for electrons, muons, charged pions, charged kaons, protons
                if(abslooppdg==11 || abslooppdg==13 || abslooppdg==211 || abslooppdg==321 || abslooppdg==2212){
                    if(looppart->Pt()>= 0.1 && looppart->Pt()<=50 && TMath::Abs(looppart->Eta())<=0.8){
                        *Vec += TLorentzVector(looppart->Px(),looppart->Py(),looppart->Pz(),looppart->Energy());
                        nadd++;
                    } else{
                        AddVec = TLorentzVector();
                        MCReconstructMass2(st,loopdaughter,nPrime, &AddVec);
                        if (AddVec != TLorentzVector()){
                            *Vec += AddVec;
                            nadd++;
                        }
                    }
                } else{
                    AddVec = TLorentzVector();
                    MCReconstructMass2(st,loopdaughter,nPrime, &AddVec);
                    if (AddVec != TLorentzVector()){
                        *Vec += AddVec;
```

```
                        nadd++;
                    }
                }
            }
        }
    }
    if (nadd<2) *Vec = TLorentzVector();
}
//_______________________________________________________________________
void AliAnalysisTaskBeauty::MCReconstructMass2(AliStack *st, Int_t iptc, Int_t ↩
    nPrime, TLorentzVector *Vec ) {
    TParticle * part = st->Particle(iptc);
    Int_t fdaughter = part->GetFirstDaughter();
    Int_t children = 0;
    TLorentzVector AddVec=TLorentzVector();

    if (fdaughter >0){
        children = part->GetNDaughters();
        for ( Int_t loopdaughter = fdaughter; loopdaughter < (fdaughter+children)↩
            ; loopdaughter++){
            if (loopdaughter < nPrime) {
                TParticle * looppart = st->Particle(loopdaughter);
                Int_t abslooppdg = TMath::Abs(looppart->GetPdgCode());
                // Checking for electrons, muons, charged pions, charged kaons, ↩
                    protons
                if(abslooppdg==11 || abslooppdg==13 || abslooppdg==211 || ↩
                    abslooppdg==321 || abslooppdg==2212){
                    if(looppart->Pt()>= 0.1 && looppart->Pt()<=50 && TMath::Abs(↩
                        looppart->Eta())<=0.8){
                        *Vec += TLorentzVector(looppart->Px(),looppart->Py(),↩
                            looppart->Pz(),looppart->Energy());
                    } else {
                        AddVec = TLorentzVector();
                        MCReconstructMass2(st,loopdaughter,nPrime, &AddVec);
                        *Vec += AddVec;
                    }
                } else{
                    AddVec = TLorentzVector();
                    MCReconstructMass2(st,loopdaughter,nPrime, &AddVec);
                    *Vec += AddVec;
                }
            }
        }
    }
}
//_______________________________________________________________________
void AliAnalysisTaskBeauty::KFReconstructMass(AliStack *st, AliESDEvent *esd, ↩
    AliESDtrackCuts *fCuts, AliESDtrackCuts* fTrCutsElec, Int_t flist[],Int_t ↩
    maxM, AliKFParticle Decay[], TLorentzVector Vecall[] ) {
    Int_t nESD = esd->GetNumberOfTracks();
    Int_t NPrim = st->GetNprimary();
    for (Int_t ipart=0; ipart<maxM; ipart++) {
        Int_t nadd=0;
```

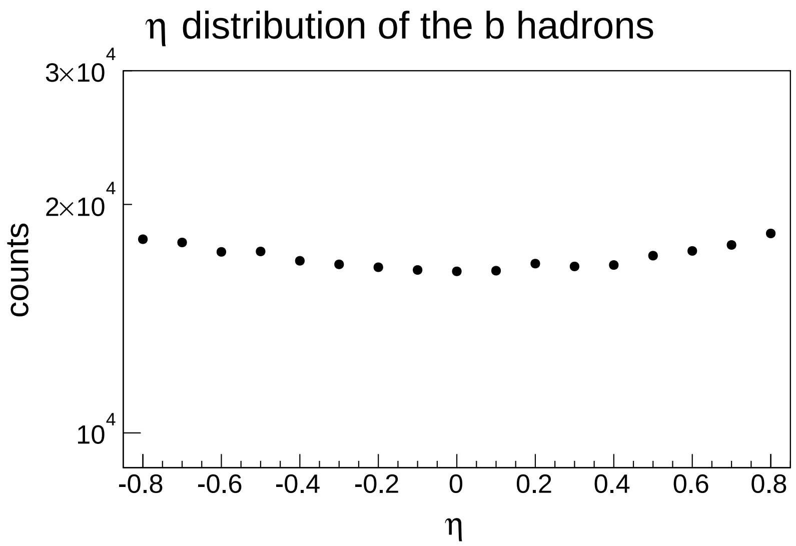
```
if(flist[ipart]>0){
   TParticle *part = st->Particle (flist[ipart]);
   Int_t ndau = part->GetNDaughters();
   Int_t firstdaughter = part->GetFirstDaughter();
   if (firstdaughter>0){
       for (Int_t idau=firstdaughter; idau < (firstdaughter+ndau); idau↩
           ++) {
          if (idau<NPrim){
              TParticle *daughter = st->Particle(idau);
              Int_t abspdg = TMath::Abs(daughter->GetPdgCode());
              if ( TMath::Abs(daughter->Eta()) <= 0.8 && daughter->Pt()>=↩
                   0.1 && (abspdg==11 || abspdg==13 || abspdg== 211 || ↩
                  abspdg==321 || abspdg==2212) ) {
                 Bool_t InESD=false;
                 for(Int_t i = 0; i < nESD; i++) {
                     AliESDtrack* esdtrack = esd->GetTrack(i);
                     if(!esdtrack) { AliError(Form("ERROR: Could not ↩
                        retrieve esdtrack %d",i)); continue; }
                     Bool_t goodESD = fCuts->AcceptTrack(esdtrack);
                     Bool_t goodelec = 1;
                     Int_t lab=TMath::Abs(esdtrack->GetLabel());
                     if (goodESD && lab == idau ) {
                        // Add measurable Daughters
                      Int_t PDG = st->Particle(lab)->GetPdgCode();
                      if (PDG == 11 || PDG == -11 || PDG==13 || PDG==-13 ↩
                          ) goodelec = fTrCutsElec->AcceptTrack(esdtrack)↩
                          ;
                      if (!goodelec) continue;
                      Decay[ipart].AddDaughter(AliKFParticle(*esdtrack, ↩
                          PDG));
                        Vecall[ipart]+=TLorentzVector(esdtrack->Px(), ↩
                            esdtrack->Py(),esdtrack->Pz(), TMath::Sqrt(↩
                            TMath::Power(esdtrack->P(),2)+TMath::Power(↩
                            daughter->GetMass(),2)));
                        nadd++;
                        InESD=true; break;
                     }
                 }
                 if (!InESD){
                     TLorentzVector AddVec = TLorentzVector();
                     AliKFParticle ADDPAR = KFReconstructMass2(st, esd, ↩
                        fCuts, fTrCutsElec, idau, &AddVec);
                     Decay[ipart].AddDaughter(ADDPAR);
                     nadd++;
                     Vecall[ipart] += AddVec;
                     if (ADDPAR.X()== 0 && ADDPAR.GetE()== 0 && ADDPAR.↩
                        GetPt()==0){
                        nadd--;
                     }
                 }
              } else {
                 TLorentzVector AddVec = TLorentzVector();
```

```
              AliKFParticle ADDPAR = KFReconstructMass2(st, esd, fCuts, fTrCutsElec, idau, &AddVec);
              Decay[ipart].AddDaughter(ADDPAR);
              Vecall[ipart] += AddVec;
              nadd++;
              if (ADDPAR.X()== 0 && ADDPAR.GetE()== 0 && ADDPAR.GetPt()==0){
                nadd--;
              }
            }
          }
        }
      }
    }
    if (nadd<2){
      Decay[ipart]= AliKFParticle();
      Vecall[ipart] = TLorentzVector();
    }
  }
}
//_____________________________________________________________________
AliKFParticle AliAnalysisTaskBeauty::KFReconstructMass2(AliStack *st, AliESDEvent *esd, AliESDtrackCuts *fCuts, AliESDtrackCuts* fTrCutsElec, Int_t fpart, TLorentzVector* Vec ) {
  Int_t nESD = esd->GetNumberOfTracks();
  Int_t NPrim = st->GetNprimary();
  AliKFParticle KFmother= AliKFParticle();

  TParticle *mother = st->Particle(fpart);
  Int_t ndau = mother->GetNDaughters();
  Int_t firstdaughter = mother->GetFirstDaughter();

  if (firstdaughter>0){
   for (Int_t idau=firstdaughter; idau < (firstdaughter+ndau); idau++){
    if (idau<NPrim){
     TParticle *daughter = st->Particle(idau);
     Int_t abspdg = TMath::Abs(daughter->GetPdgCode());
     if ( TMath::Abs(daughter->Eta()) <= 0.8 && daughter->Pt()>= 0.1 && ( abspdg==11 || abspdg==13 || abspdg== 211 || abspdg==321 || abspdg==2212 ) ){
      Bool_t InESD=false;
      for(Int_t i = 0; i < nESD; i++) {
       AliESDtrack* esdtrack = esd->GetTrack(i);
       if(!esdtrack) { AliError(Form("ERROR: Could not retrieve esdtrack %d",i)); continue; }
       Bool_t goodESD = fCuts->AcceptTrack(esdtrack);
       Bool_t goodelec = 1;
       if (goodESD){
        Int_t lab=TMath::Abs(esdtrack->GetLabel());
        if ( lab == idau ) {
         // Add measurable Daughters
         Int_t PDG = st->Particle(lab)->GetPdgCode();
```

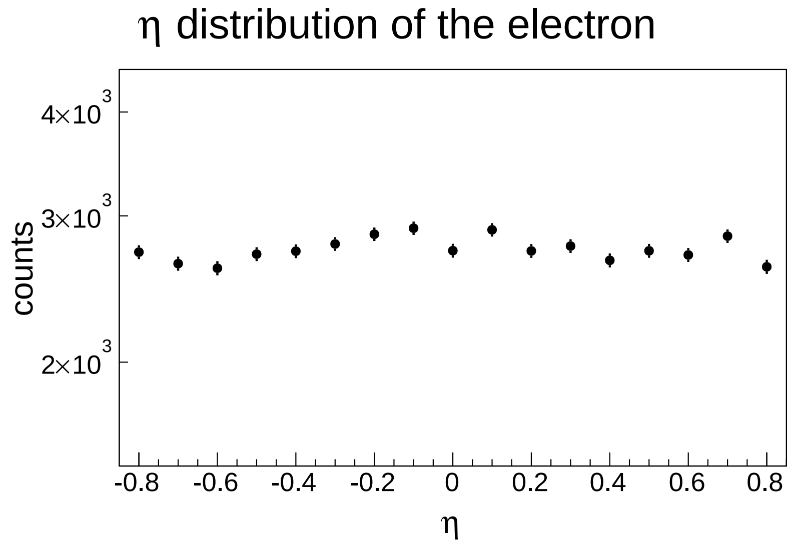
```
         if (PDG == 11 || PDG == -11 || PDG==13 || PDG==-13) goodelec = ↩
           fTrCutsElec->AcceptTrack(esdtrack);
         if (!goodelec) continue;
         KFmother.AddDaughter(AliKFParticle(*esdtrack,PDG));
         *Vec += TLorentzVector(esdtrack->Px(), esdtrack->Py(),esdtrack->↩
           Pz(), TMath::Sqrt(TMath::Power(esdtrack->P(),2)+TMath::Power↩
           (daughter->GetMass(),2)));
         InESD = true; break;
        }
       }
      }
      if (!InESD){
       TLorentzVector AddVec = TLorentzVector();
       KFmother.AddDaughter(KFReconstructMass2(st, esd, fCuts, fTrCutsElec,↩
          idau, &AddVec));
       *Vec += AddVec;
      }
     }
     else{
      TLorentzVector AddVec = TLorentzVector();
      KFmother.AddDaughter(KFReconstructMass2(st, esd, fCuts, fTrCutsElec, ↩
        idau, &AddVec));
      *Vec += AddVec;
     }
    }
   }
  }
  return KFmother;
}
//________________________________________
Bool_t AliAnalysisTaskBeauty::checkmother(AliStack *st, TParticle *part) {
  Bool_t check = false;
  Int_t nmoth = part->GetFirstMother();
  if (nmoth <! 0) return 0;
  TParticle *mother = st->Particle(part->GetFirstMother());
  Int_t absmotherpdg = TMath::Abs(mother->GetPdgCode());
  if ( absmotherpdg == 511 || absmotherpdg == 521 || absmotherpdg == 541 || ↩
    absmotherpdg == 531 || absmotherpdg==5122) {
    check = true;
    return check;
  }
  else if (absmotherpdg == 1 || absmotherpdg == 2 || absmotherpdg == 3 || ↩
    absmotherpdg == 4 || absmotherpdg == 5 || absmotherpdg == 6 || ↩
    absmotherpdg == 7 || absmotherpdg == 8){
    return 0;
  }
  else {
    check = checkmother(st, mother);
  }
  return check;
}
```

Code A.1: Code of the analysis tasks performed for this thesis (AliAnalysisTaskBeauty.cxx).

A.2. Additional plots

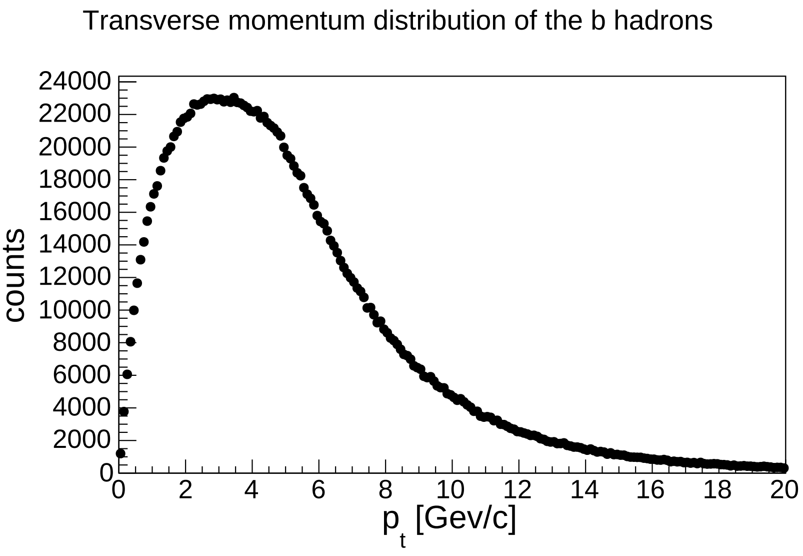


(a) The pseudorapidity distribution of the b hadrons simulated with Pythia.

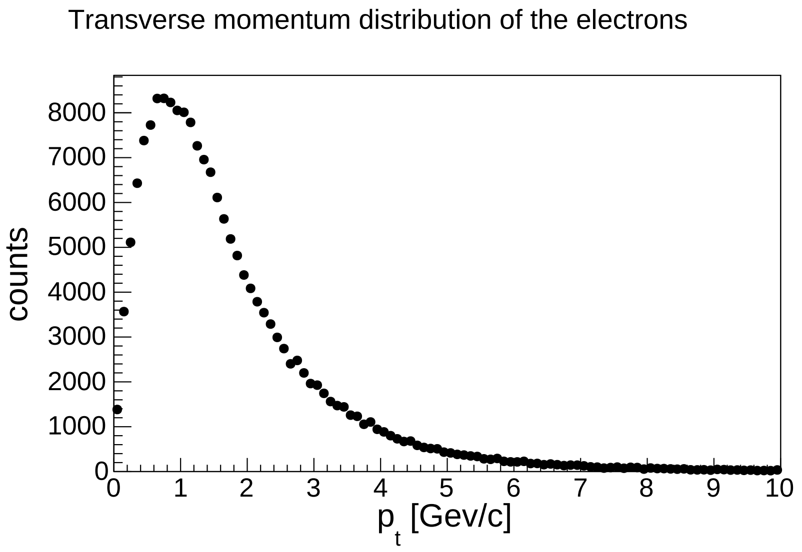


(b) The pseudorapidity distribution of electrons coming directly from b hadrons simulated with Pythia.

Figure A.1.: The pseudorapidity distribution of the b hadrons and direct electron daughters simulated with Pythia. The diagrams show that the spatial distribution of b hadrons and their direct electron daughter is fairly flat within the acceptance of the ALICE detector.

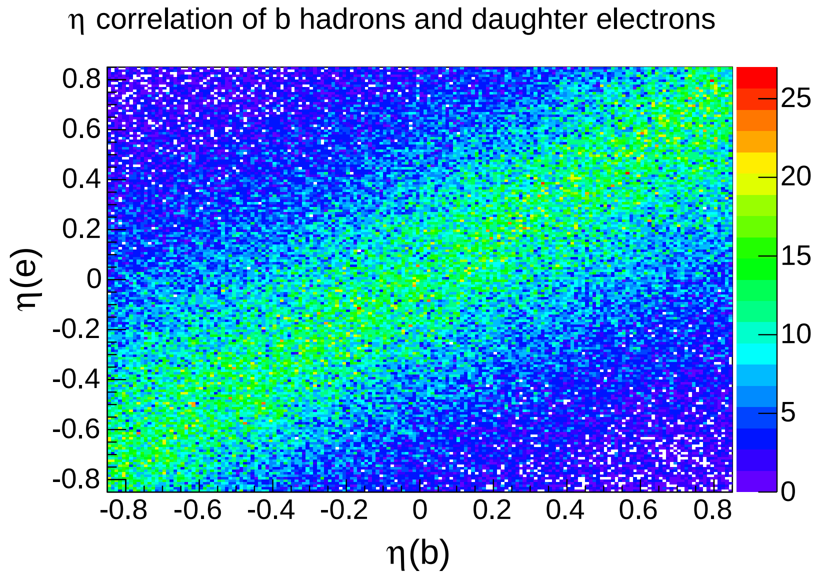


(a) The transverse momentum distribution of the b hadrons simulated with Pythia.

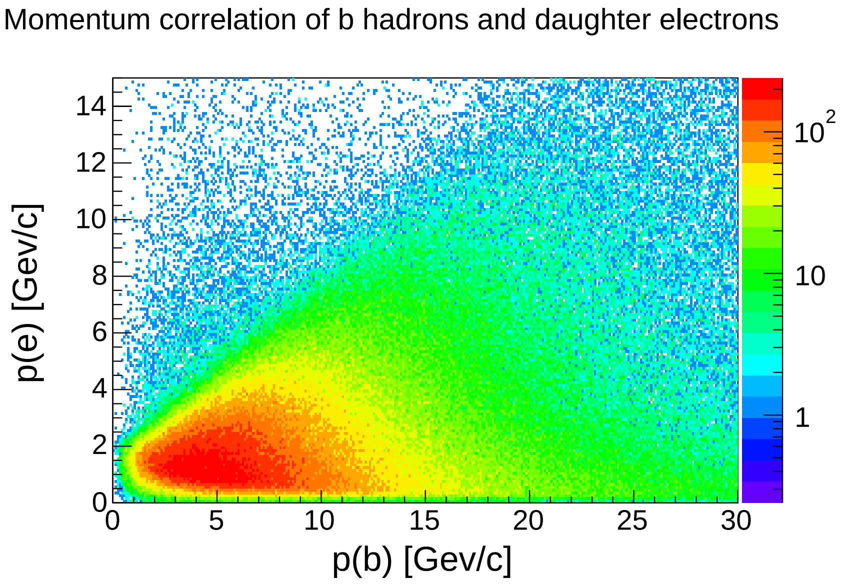


(b) The transverse momentum distribution of electrons coming directly from b hadrons simulated with Pythia.

Figure A.2.: The transverse momentum distribution of the b hadrons and direct electron daughters simulated with Pythia.

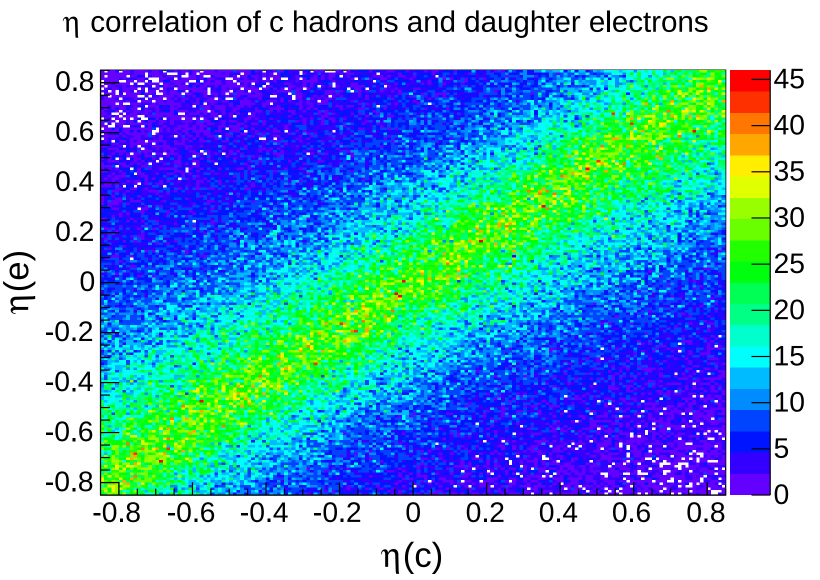


(a) The pseudorapidity correlation of the b hadrons and their direct electron daughters taken from the heavy flavour enhanced sample.

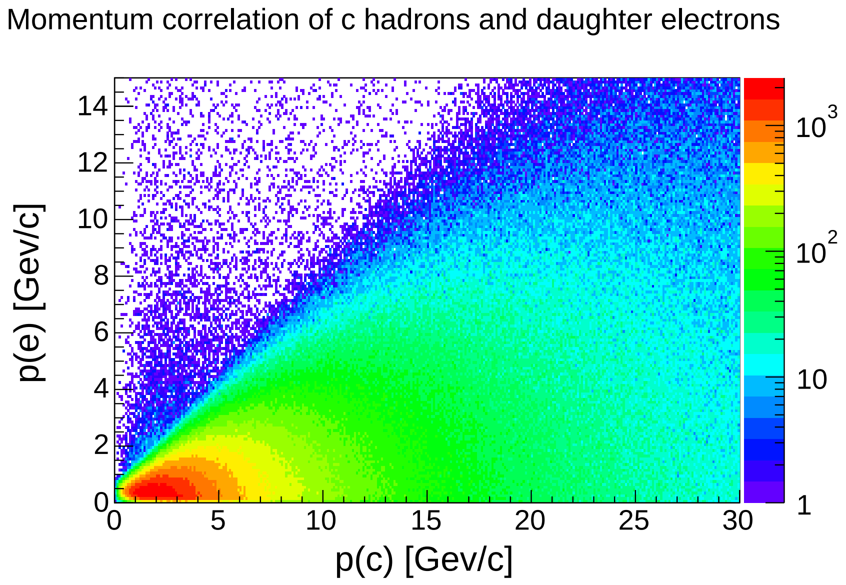


(b) The momentum correlation of the b hadrons and their direct electron daughters taken from the heavy flavour enhanced sample.

Figure A.3.: Correlations of b hadrons and their direct electron daughters taken from the heavy flavour enhanced sample.



(a) The pseudorapidity correlation of the c hadrons and their direct electron daughters simulated with Pythia.



(b) The momentum correlation of the c hadrons and their direct electron daughters simulated with Pythia.

Figure A.4.: Correlations of c hadrons and their direct electron daughters with Pythia.

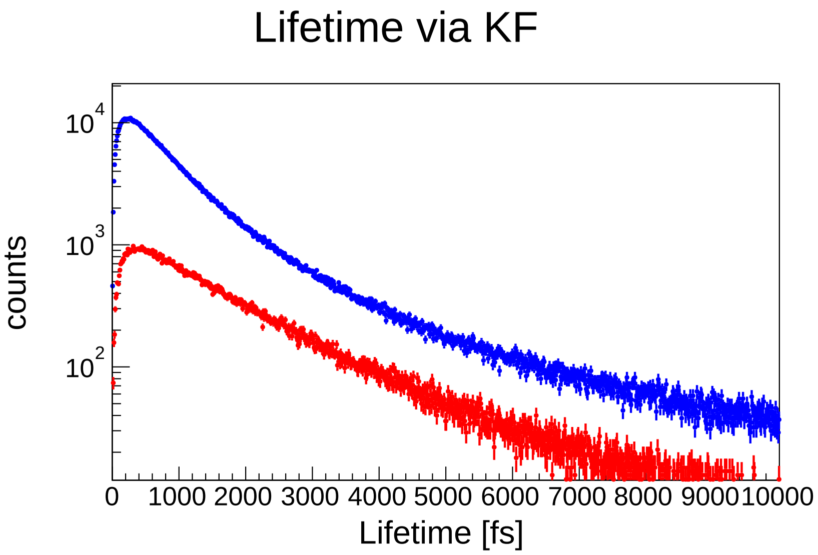
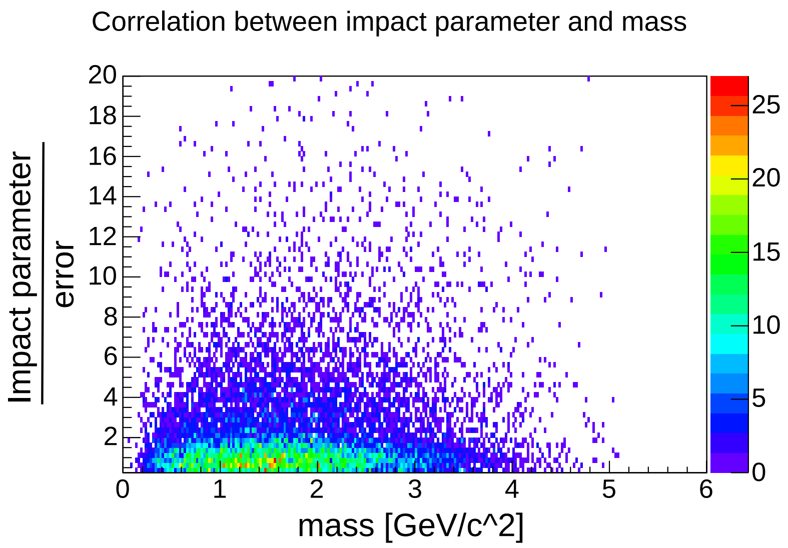
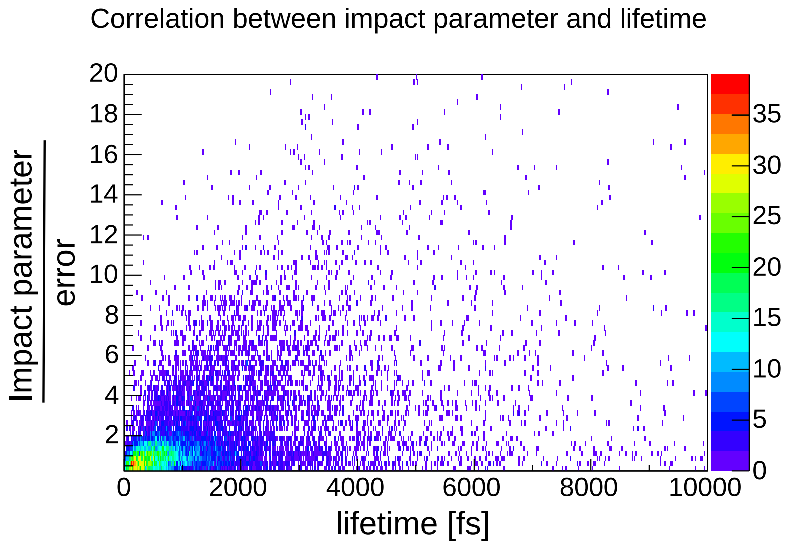


Figure A.5.: This unnormalized lifetime plot shows the distribution of the lifetime values for c hadrons (blue) and b hadrons (red) calculated using the information given by the reconstructed tracks. These plot shows why the lifetime is not a good standalone criterion to distinguish b and c hadrons without any pre-selection, but it can be used in combination with other criteria.

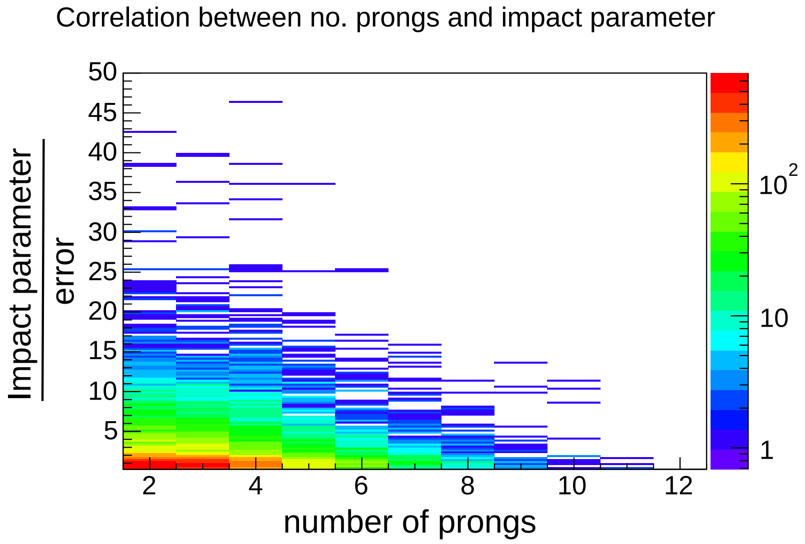


(a) Correlation between the impact parameter of the descending electron and the mass of the reconstructed b hadron.

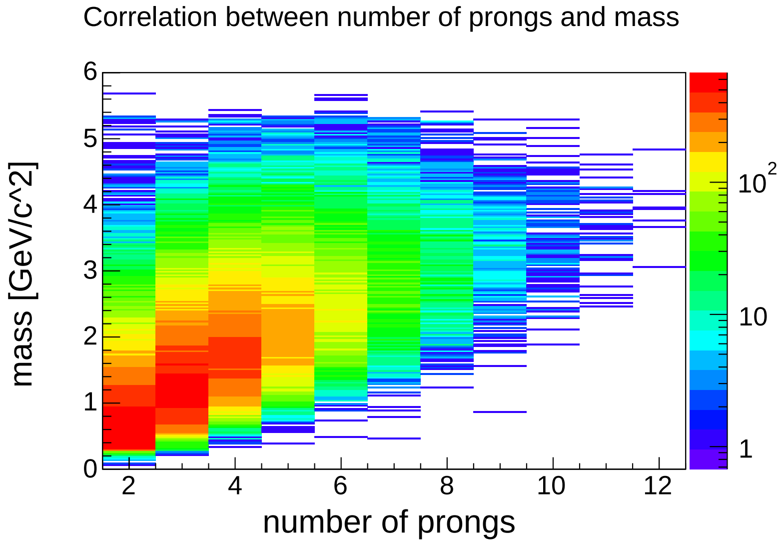


(b) Correlation between the impact parameter of the descending electron and the lifetime of the reconstructed b hadron.

Figure A.6.: Correlation plot belonging to the analysed criteria (Part 1).

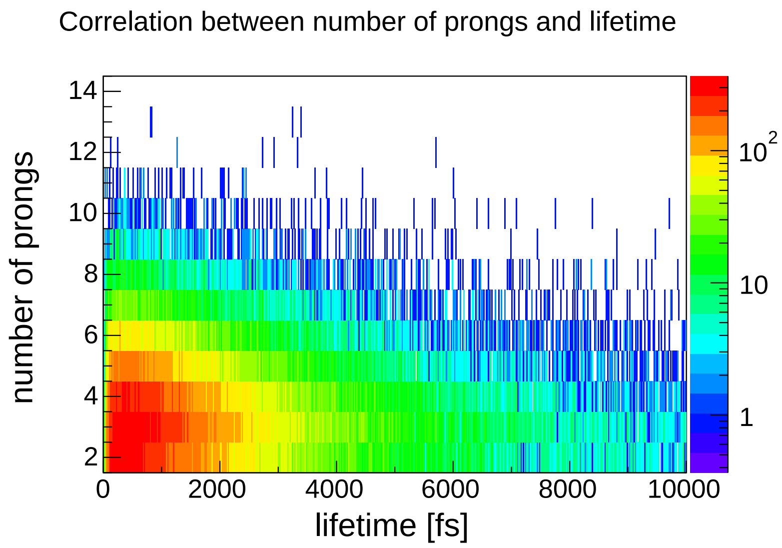


(a) Correlation between the impact parameter of the descending electron and the number of contributors to the reconstructed b hadron.

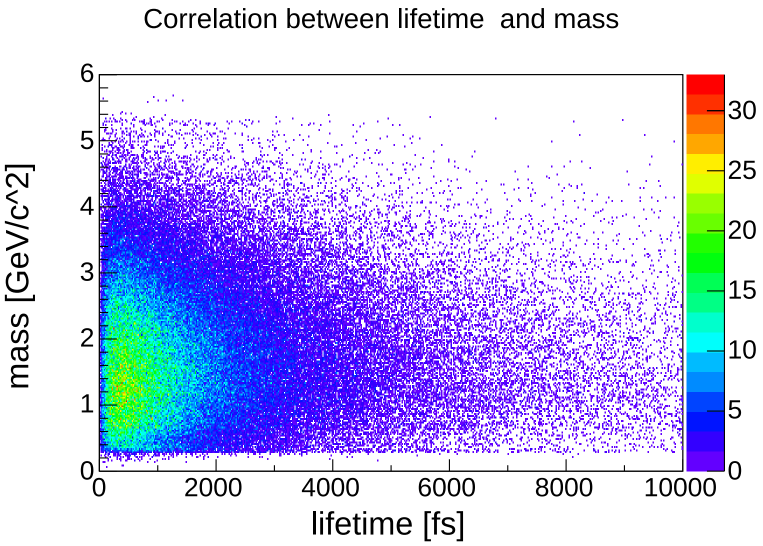


(b) Correlation between the mass of the reconstructed b hadron and the number of contributors to it.

Figure A.7.: Correlation plot belonging to the analysed criteria (Part 2).



(a) Correlation between the lifetime of the reconstructed b hadron and the number of contributors to it.



(b) Correlation between the mass and the lifetime of the reconstructed b hadron.

Figure A.8.: Correlation plot belonging to the analysed criteria (Part 3).

Bibliography

[1] URL: http://en.wikipedia.org/wiki/Standard_Model.

[2] URL: http://www.teilchen.at/images/b389/phasendiagram.jpg.

[3] A. Andronic. „The study of quark-gluon matter in high-energy nucleus-nucleus collisions“. In: *American Institute of Physics Conference Series*. Ed. by L. Trache and P. Gina Isar. Vol. 1498. American Institute of Physics Conference Series. Nov. 2012, pp. 125–133. DOI: 10.1063/1.4768487. arXiv: 1210.8126 [nucl-ex].

[4] R. Averbeck. „Heavy-flavor production in heavy-ion collisions and implications for the properties of hot QCD matter“. In: *Progress in Particle and Nuclear Physics* 70 (May 2013), pp. 159–209. DOI: 10.1016/j.ppnp.2013.01.001.

[5] R. Vernet (for the ALICE Collaboration). „Soft Probes of the Quark-Gluon Plasma with ALICE at LHC“. In: *ArXiv e-prints* (June 2009). arXiv: 0906.1171 [nucl-ex].

[6] A. Andronic. „An overview of the experimental study of quark-gluon matter in high-energy nucleus-nucleus collisions“. In: *ArXiv e-prints* (July 2014). arXiv: 1407.5003 [nucl-ex].

[7] URL: http://pdg.lbl.gov/.

[8] The ALICE Collaboration. „Measurement of electrons from beauty hadron decays in pp collisions at sqrt{s} = 7 TeV“. In: *ArXiv e-prints* (Aug. 2012). arXiv: 1208.1902 [hep-ex].

[9] The ALICE Collaboration. „Measurement of electrons from semileptonic heavy-flavor hadron decays in pp collisions at s=7TeV“. In: *Physical Review D* 86.11, 112007 (Dec. 2012), p. 112007. DOI: 10.1103/PhysRevD.86.112007. arXiv: 1205.5423 [hep-ex].

[10] V. A. Petrov and A. Prokudin. „Three Pomerons versus D0 and TOTEM data“. In: *Physical Review D* 87.3, 036003 (Feb. 2013), p. 036003. DOI: 10.1103/PhysRevD.87.036003. arXiv: 1212.1924 [hep-ph].

[11] J. Schukraft. „Heavy-ion physics with the ALICE experiment at the CERN Large Hadron Collider“. In: *Royal Society of London Philosophical Transactions Series A* 370 (Feb. 2012), pp. 917–932. DOI: `10.1098/rsta.2011.0469`. arXiv: `1109.4291 [hep-ex]`.

[12] The ALICE Collaboration. „Performance of the ALICE Experiment at the CERN LHC“. In: *ArXiv e-prints* (Feb. 2014). arXiv: `1402.4476 [nucl-ex]`.

[13] URL: `http://aliceinfo.cern.ch/Public/en/Chapter2/Page3-subdetectors-en.html`.

[14] R. Engel, J. Ranft, and S. Roesler. „Hard diffraction in hadron-hadron interactions and in photoproduction“. In: *Physical Review D* 52 (Aug. 1995), pp. 1459–1468. DOI: `10.1103/PhysRevD.52.1459`. eprint: `hep-ph/9502319`.

[15] T. Sjöstrand, S. Mrenna, and P. Skands. „PYTHIA 6.4 physics and manual“. In: *Journal of High Energy Physics* 5, 026 (May 2006), p. 26. DOI: `10.1088/1126-6708/2006/05/026`. eprint: `hep-ph/0603175`.

[16] R. Brun et al. „GEANT Detector Description and Simulation Tool“. CERN Program Library Long Writeup W5013. 1994.

[17] I. Kisel, I. Kulakov, and M. Zyzak. „Standalone First Level Event Selection Package for the CBM Experiment“. In: *IEEE Transactions on Nuclear Science* 60 (Oct. 2013), pp. 3703–3708. DOI: `10.1109/TNS.2013.2265276`.

List of Figures