Universidade de Santiago de Compostela
Departamento de Física de Partículas
Laboratorio Carmen Fernández

New Contributions to the Momentum Reconstruction Methods and First Analysis with Proton-Proton Elastic Collisions in the HADES Experiment

Memoria presentada por:
PABLO CABANELAS EIRAS
para optar ó
Grao de Licenciado en Física
Setembro 2005

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

JUAN A. GARZON HEYDT, PROFESOR TITULAR DO DEPARTAMENTO DE FISICA DE PARTICULAS DA UNIVERSIDADE DE SANTIAGO DE COMPOSTELA,

CERTIFICA:

que a memoria titulada:

New Contributions to the Momentum Reconstruction Methods and First Analysis with Proton-Proton Elastic Collisions in the HADES Experiment

foi realizada por **D. PABLO CABANELAS EIRAS** no **Laboratorio Carmen Fernández** do Departamento de Física de Partículas da Universidade de Santiago de Compostela baixo a dirección do Prof. **Dr. Juan A. Garzón (U. Santiago de Compostela)** e do **Dr. Manuel Sánchez García (CHUS)** e constitue o traballo de **tesina** que presenta para optar ó **Grao de Licenciado en Física.**

Santiago de Compostela,
1 de Setembro de 2005

Juan A. Garzón Heydt
Departamento de Física de Partículas

Agradecementos

Nunca resulta sinxelo escribir unha páxina de agradecementos. Detrás de calquera traballo non hai unha soa persoa que o asina, senón que, dun xeito ou doutro, sempre hai unha boa morea de xente por detrás. Por iso agardo non esquecerme polo menos dos mais importantes.

En primeiro lugar, debo darlle as gracias ó profesor Juan Antonio Garzón (Hans) por darme a oportunidade de incorporarme ó seu grupo de traballo.

En segundo lugar, quero agradecerlle ós meus pais o seu esforzo en darme unha carreira e en facerme coma hoxe son. Tamén á miña irmá, que sofre ó meu lado o día a día, e ó resto da familia que sempre me apoiou, en especial a miña avoa.

Unha mención especial merecen os meus compañeiros de chollo que dende o principio me axudaron: Miguel, por mostrarme os comenzos (o mais difícil!); Héctor, polas súas boas ideas e porque sempre ten resposta (que listo é..., e guapo!); Diego, que foi o primeiro en ve-lo "tridente" en HADES; David, cos trucos de LaTeX e ROOT (e pola compañía); e por suposto, Manuel, sinxelamente por todo. Todo é del.

Quero darlle as gracias a tódolos amigos que dalgún xeito me axudaron durante a miña vida en Santiago, en especial a dous: a Diego, polo soporte informático, por aguantarme, polas parrafadas diarias, polo café en Pachín (bueno, agora xa non), as cañas, caipirinhas, Tapia...; e a Xosé Díaz Díaz, porque coñecerte e vivir contigo foi das mellores cousas que me ocorreron na vida. Sen o teu exemplo algunhas veces non sairía adiante.

I would also like to thank all the HADES (and non-HADES) people who has help me during my stays at GSI: Alex, Gosia, Isaac, Noe, Jochen, Stefano, Sasha...

E bueno, non penses que me esquecía de ti, "irundiña". Non é que esté moi seguro se sen ti sería todo mais fácil ou mais difícil, pero do que si estou seguro é de que non quero probar. Gracias polo teu apoio, Miriña.

Contents

List of Figures

List of Tables

Introduction

The work presented here is embedded in the international HADES Collaboration. This Collaboration is composed of 19 institutions from 9 European countries, with almost two hundred of scientists workig together. HADES, High Acceptance Di-Electron Spectrometer, is the name of the full detector system. It consists on a variety of instruments working in harmony in only one spectrometer. The HADES central headquarters is placed at the GSI (Gessellschaft für SchwerIonen forschung) institute in Darmstadt, Germany. The spectrometer and the needed particle accelerator are also there.

HADES is designed for the study of lepton pair emission in relativistic heavy ion collisions, di-lepton production in elementary reactions and experiments aimed at studying the structure of hadrons. For this purpose, the detector has a geometrical acceptance of almost 50% for electron pairs and a mass resolution of 0.8% (sigma) in the vector mesons ρ and ω region (771 and 782 MeV/c^2 respectively). It is also needed to assign very precise directions and momenta to the particles in order to determine the characteristics of the decaying particle.

The spectrometer is composed of six sectors, symmetrically distributed around the beam axis with nearly full azimuthal coverage. On each sector, from inner to outer, it is composed by the following devices:

- START and VETO diamond detectors providing the trigger signals
- A Ring-Imaging Cherenkov detector (RICH) which allows electrons to be selected
- Multiwire Drift Chambers (MDCs) consisting in two modules before the magnetic field and another two after, in order to determine the position and direction of electrons with high accuracy
- A Superconducting Toroidal Magnet, providing the magnetic feild
- Time-Of-Flight Walls (TOF and TOFino) able to distinguish particle types by using the fact that lighter particles are faster than heavier ones

- A SHOWER detector which can separate leptons from hadrons

In addition to the detectors themselves, highly advanced electronics and software analysis programs are used to understand and merge the information from the various subsystems. The whole event reconstruction software is explained in this work with detail.

One of the parts of such analysis software is the momentum reconstruction. They are a set of algorithms developed in function of spectrometer setup available. Here we present some work related with two of those algorithms, the so called "Kick Plane" and "Reference Trajectories".

Kick Plane algorithm is used either when only the MDCs which are before the magnet are available, reaching a resolution about 8%, or when we have also the first MDC after the magnet, with a resolution of 5%. It is based in the idea that the deflection of a particle in a magnetic field is proportional both to the field strength and the distance traveled in the field. Then we can replace the original field by another one compressed in space but increased in strength. In the limit, all deflection takes place in a surfce: the Kick Plane. The relation between deflection and momentum is given by constants (kick plane parameters) depending only on the points of the kick plane. Some tests of this method were performed, like the dependency of such parameters with magnetic field or detector's alignment. Some improvements were also tested using the real data of the so called Nov02 HADES bematime.

Reference Trajectories method is used when the full spectrometer setup is available. It is based in the use of a Montecarlo simulation (GEANT packages) for builting a table of all possible tracks. We store the information given by detectors per each track. When we want to know the momentum of a real (or simulated) track, we just need to look for in the table the more similar track. We can reach with this method resolution in momentum reconstruction around 2%.

Reference Trajectories method was never tested in real data. The data sets chosen to do it were the so called Sep03 and Jan04 beamtimes, corresponding to *p*-*p* collisions at 2.2 GeV. Approximately half of these reactions are elastic scattering of protons, which have a well known kinematics. Then, we can know in advance the momentum of an elastic scattered proton just by measuring the angle of the interaction, and then comparing it with the momentum given by the algorithm. We just need to perform a good elastic events selection and apply some corrections like an energy loss correction. These data represent a good tool not only for checking momentum reconstruction, but also for another methods like alignment or acceptance.

Chapters 1 and 2 give us a description of the HADES spectrometer as well as the software structure.

In Chapter 3, a description of the KickPlane Method is given. We can see also tests and improvements of such method and some results in both simulated and real data.

Chapter 4 deals with the Reference Trajectories Method, its description and its aplication to proton-proton elastic scattering real data. Some theory of such collisions and comparisons with simulated data are also shown.

Another studies like the movement of charged particles within the magnetic field are also treated in this work and explained in Appendixes A, B and C.

Chapter 1

The HADES experiment

Investigation on the properties of nuclear matter at high temperatures and densities conditions is essential to understand processes like, for example, those giving birth to the Universe in the Big Bang and its later evolution, since at those moments the medium was one of high temperature and density. This line of investigation contributes also to obtain the equation of state of nuclear matter which is not only important in Nuclear Physics, but also to understand physical processes taking place during the latest stages of stars evolution.

HADES[1] contributes to that line of investigation. The focus of HADES is the study of in medium modifications to the properties of the vector mesons. Calculations based on QCD and some hadronic models, predict detectable changes in the width and mass of hadrons produced in dense nuclear medium. From the point of view of QCD such modifications could be a signal of the so called *chiral symmetry restoration*, which is a non perturbative phenomenon. HADES' main goal is to provide experimental insight for the study of QCD on the non perturbative regime and possibly see a signal of the expected chiral symmetry restoration.

[1]High Acceptance DiElectron Spectrometer

1.1 The HADES physics

The basic idea to observe the characteristics of vector mesons in dense nuclear matter is to produce them in heavy ion collisions and then analyze the different variables in the collision's final state. The different particle species have been studied (pions, kaons, protons...). HADES [Col94], [ea04a], [HAD] is focused on the study of lepton pairs produced in the decays of vector mesons inside the hot, dense medium. However, the spectrometer is also able to detect and study the properties of hadrons and this is an important task.

During the initial stage of an ion collision at 1 or 2 AGeV, a compression phase is created where density reaches values of up to 3 times that of normal nuclear matter. This compression phase lasts for about 10 fm/c and is followed by an expansion phase. During the expansion, meson scattering, absortion and reemision equilibrate the various hadronic species. After a few tens of fm/c the expansion makes inelastic reactions between constituents impossible, hence the hadrochemical composition is frozen. This is the so called *freeze out point* (See fig. 1.1).

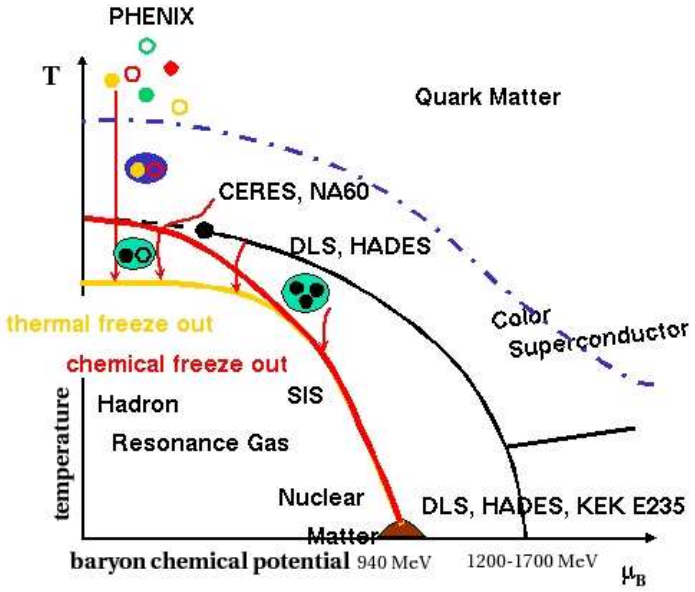


Figure 1.1: *The freeze-out points in the QCD phase diagram*

What we want to study is the high density phase. For that purpose light

vector mesons are a well suited probe. Their lifetimes (see table 1.1) are short enough for them to have a significant chance of decaying in the same dense medium where they were created. When they decay, they may do so in two leptons. Since leptons do not experiment *strong* interaction, when they leave the interaction zone they retain memory about how they were produced. Hence they carry information about the properties of the vector mesons in the dense medium. If their masses or widths have changed due to a partial restoration of chiral simetry we should be able to tell by looking at the lepton pairs' invariant masses. Predominant dilepton sources for m $e^+e^- > 500MeV/c^2$ are direct or Dalitz decays of the light mesons ρ, ω and ϕ produced in these collisions.

The main problem is the low branching ratio for the dilepton channel in the vector meson decays. This needs to be compensated with high statistics, which translates into a need for a high acceptance spectrometer and high beam intensities. Another problem is the presence of several background sources, like pion Dalitz decays, wich also produce leptons and lepton pairs, so that the vector meson signal is sitting on a continous background.

Vector mesons are hadrons composed by a quark and an anti-quark. ρ, ω and ϕ mesons have diferent properties concerning lifetimes, widths, lepton pair branching ratios.... It seems that ρ meson is the more suitable to be used as a probe since it has a large chance of decaying in the dense zone but nevertheless, all three of them contribute and will be studied.

Meson	Mass $(\frac{MeV}{c^2})$	Width $(\frac{MeV}{c^2})$	$c\tau$ (fm)	Dominant channel	e^+e^- branching ratio
ρ	775.8	150	1.3	$\pi\pi$	4.67×10^{-5}
ω	782.6	8.49	23.4	$\pi^+\pi^-\pi^0$	7.14×10^{-5}
ϕ	1019.5	4.26	44.4	K^+K^-	2.98×10^{-4}

Table 1.1: *Light vector mesons life times [PDG]*

1.1.1 Previous experiments

In the ultra-relativistic energy regime pioneering work have been performed by the CERES at CERN[2], HELIOS1-3 and NA38-50-51 collaborations. All of them find more or less pronounced enhacements of the dilepton yield as compared to an appropiate superposition of leptons originating from hadrons decaying after freeze-out. In the energy regime of HADES, the only existent data are from the original DLS experiment at BEVALAC, Berkley.

[2]Laboratoire Européen pour la Physique des Particules

An excess of up to a factor 7 was found in the intermediate mass region, but with limited mass resolution and acceptance which translated in rether low statistics.

HADES aims to improve on the resolution and statistics from previous experiments to be able to put the different models to test, since the HADES detector is specifically designed to provide an excellent mass resolution ($\delta m/m \approx 1\%$).

1.2 The acelerator

The accelerator machine providing the beam for HADES is located at the GSI[3] Institute in Darmstadt, Germany [GSI].

The accelerator complex consists of 4 major structures: a linear accelerator (UNILAC) injecting ions into a 60 meter diameter Synchrotron (SIS), from where the beam can be extracted to the Fragment Separator (FRS), to the Electron Storage Rings (ESR) or to the experimental areas.

The UNILAC was built in 1975 as a Wideroe-Alvarez linear acelerator and was upgraded in 1999 to become a high current injector for the SIS. This new high current injector, called HSI, provides an increase in the beam intensities filling the synchrotron up to its space charge limit for all ions. Two ion sources feed the HSI. After stripping and charge state separation, the beam from the HSI is matched to the Alvarez accelerator, which accelerates the ion beams, without any significant particle loss, up to a few AMeV.

The SIS is a synchotron with a circunference of 216 meters consisting in 24 bending magnets and 36 magnetic lenses. Before entering the SIS, ions from the UNILAC interact with a carbon foil achieving high ionization states. The acceleration takes place in two resonance cavities diametrically opposed, where ions see a potential of 15 kV. The operation frequency ranges from 800KHz to 5.6MHz. The vacuum in the beam line is lower than 10^{-11} Torr.

1.3 The HADES spectrometer

HADES [Col94], [HAD] is a second generation experiment in high resolution dilepton spectroscopy. It intends to precisely measure the invariant mass of lepton pairs produced by the decay of vector mesons in heavy ion collisions. That goal puts a number of requirements on the design of HADES, shaping it.

[3]Gessellschaft für SchwerIonen forschung

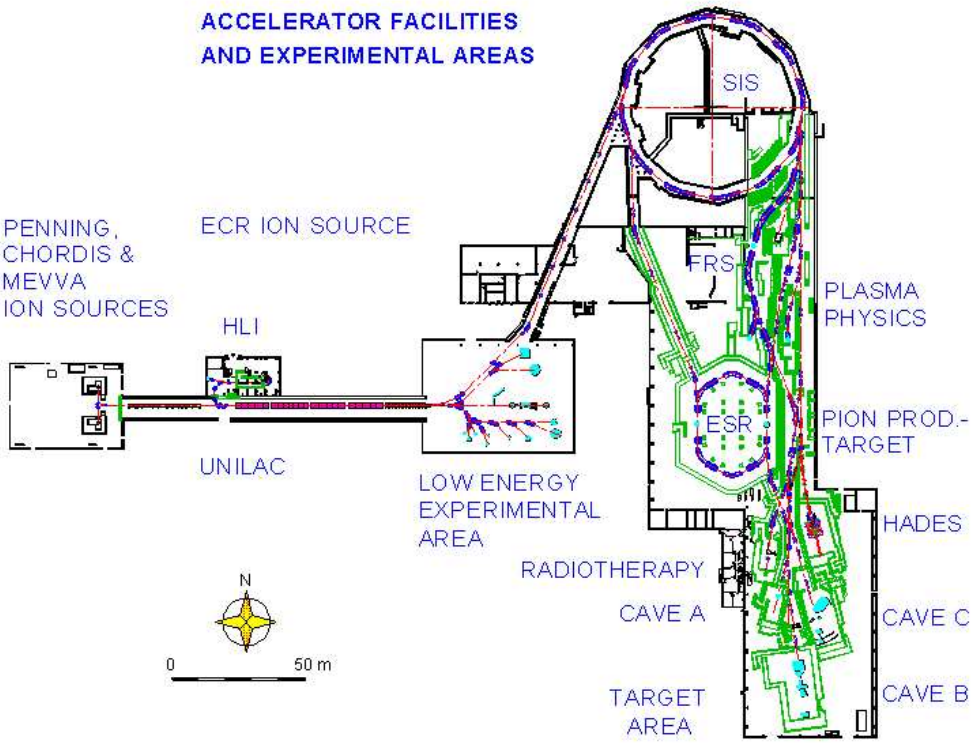


Figure 1.2: *GSI's accelerator complex*

A quick overview of the spectrometer will be offered in this section before entering into a more detailed description of the different sub detectors.

The design of HADES is governed by the high-multiplicity enviroment of heavy ion collisions, the production cross sections below threshold, and the small branching ratio for the dilepton decay channel due to the electromagnetic coupling constant. Only one of $10^5 - 10^6$ central Au+Au collisions will produce an e^+e^- pair from a meson decay. For this reason, the key features of the new instrument are:

- *Large acceptance* in order to maximize the probability of detecting a pair once it is produced. The acceptace of HADES is $\epsilon_{pair} = 40\%$.
- *High count rates* need to be supported. The goal is to be able to operate with beam intensities as high as 10^8 particles per second.
- A *trigger system* able to downscale the amount of raw data by several orders of magnitude. This trigger scheme in HADES is made of three stages, ideally the joint rejection power would be in the order of 10^4.

- A *high resolution for invariant mass* reconstruction, in the order of the ω width in the mass region of the ω, that is $\frac{\Delta M_{inv}}{M_{inv}} \approx 1\%$.
- A *signal to background ratio* larger than unity for invariant masses up to $M \approx 1GeV/c^2$.
- *Sufficient granurality* to minimize ambiguities and provide redundant tracking information.

1.3.1 Overview of the spectometer

Fig. 1.3 shows an schematic view of the HADES spectrometer. It shows how it is divided azimuthally in six identical sectors, each covering polar angles $18^o < \theta < 85^o$, with practically full azimuthal coverage, besides the shadow regions introduced by the coils and detector frames. This gives an acceptance for lepton pairs of 40%. The detector systems in HADES are, from inner to outer:

- Start and Veto detectors sitting before and after the magnet respectively. These are two fast diamond detectors providing a 'start' signal for TDCs and vetoing events with no interaction in the target.
- A Ring Imaging Cherenkov (RICH) detector, placed around the target. This detector works in the threshold mode and provides lepton identification.
- A magnetic spectrometer composed of two sets of two Multi-wire Drift Chambers (MDCs) separated by a superconducting magnet made up of six coils shaping a toroidal field embedded in the space between the chambers. MDCs determine the track's direction and position before and after the magnet, and the magnet provides the deflectin between them.
- A Time Of Flight (TOF) plastic scintillator wall made of thin scintillator strips for large polar angles. For low polar angles (below 45^o) four big plastic scintillators (TOFINO) cover the area providing the time of flight determination in low multiplicity reactions.
- A SHOWER detector for lower polar angles, made of three gaseous chambers separated by lead converters, which is able to separate electromagnetic and hadronic showers.

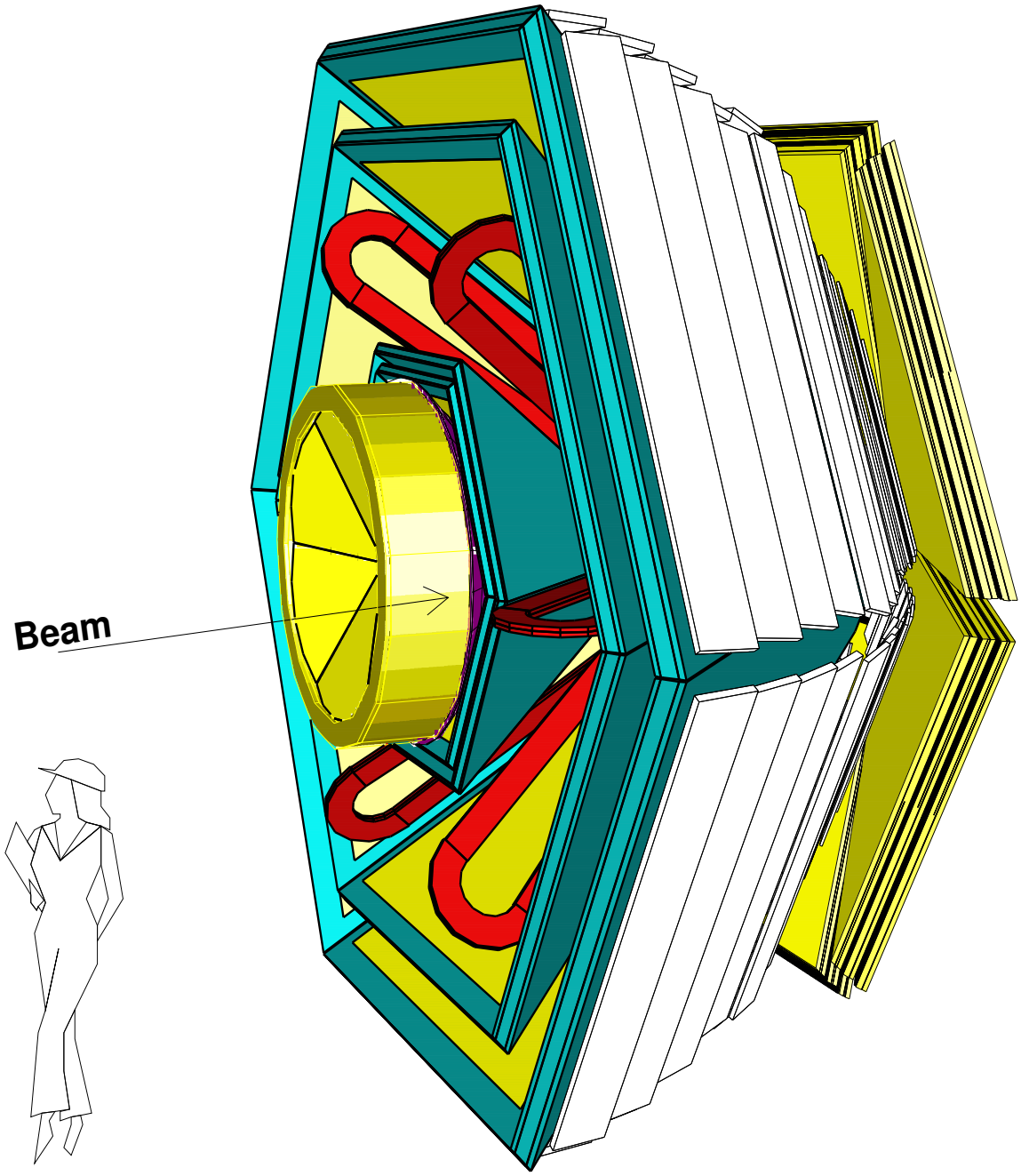


Figure 1.3: *3D view of the HADES detector. The hexagonal symmetry is apparent*

1.3.2 The START and VETO detectors

Figure 1.4: *Start detector*

START and VETO are two identical 8-strip diamond detectors of octagonal shape and are placed 75 cm downstream respectively 75 cm upstream of the HADES target. The downstream detector shall veto all particles with no reaction with target nuclei to provide a start signal with a rate of more than 10^7 particles/s. The widths of the strips are optimized such that a coincidence of one start strip with 3 veto strips is sufficient for a veto efficiency of 96.5%. The outer dimensions of the detectors are 25 mm and 15 mm matching the beam spot in this position. To keep multiple scattering and secondary reactions low the detectors have a thickness of 100 μm.

Both diamond detectors are synthesized using a Chemical Vapor Deposition (CVD) technique, wich allows the diamond to grow in an enviroment under control.

The detector is capable of a time resolution, including electronics, of up to 29ps and tolerates rates of more than 10^8 particles per second for a single detector channel. The detectors are radiation resistant and can be constructed in very thin layers.

1.3.3 The RICH detector

The Ring Imaging CHerenkov detector (RICH) is a device for electron and positron identification in hadron and heavy ion induced nuclear reactions and at $10^5 s^{-1}$ interaction rate. The detector is placed in the field-free region of the spectrometer around a segmented target to assure full polar angle acceptance. A schematic view of the RICH is shown in fig. 1.6. Leptons with momenta $100 MeV/c < p_e < 1500 MeV/c$ will produce Cherenkov radiation, when velocity is larger than that of the light in the medium, when passing through the gaseous C_4F_{10}-radiator, while all hadrons in the same momentum region have velocities far below the threshold ($\gamma_{th} \approx 18$ for C_4F_{10}). The emitted Cherenkov radiation is reflected by a segmented spherical mirror with

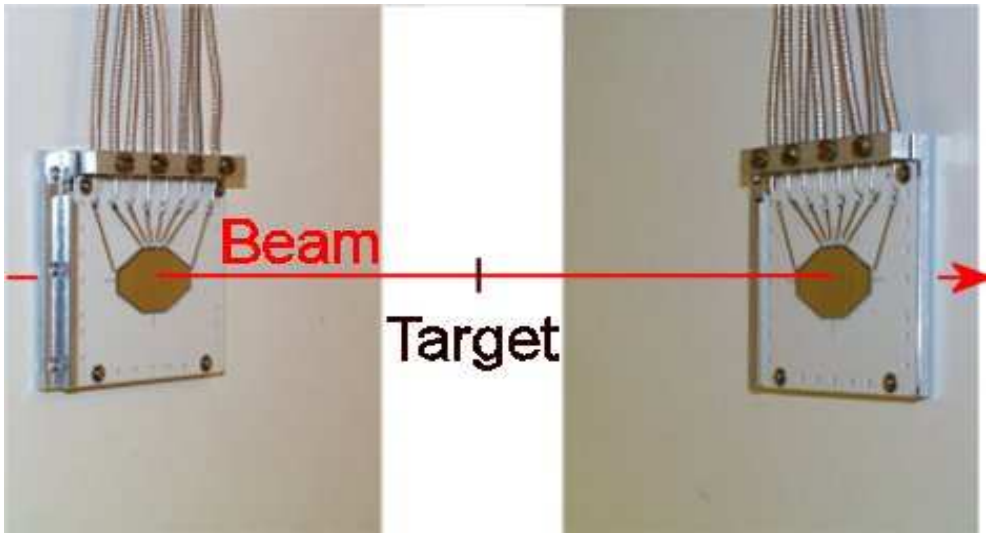


Figure 1.5: *Start and Veto detectors*

Vacuum Ultra-Violet (VUV) wavelength region and focused onto a position sensitive photon detector with CaF_2 entrance window to form rings of almost constant radius.

The VUV mirror has a radius of curvature $R = 871mm$ and a diameter $D = 1.50m$. For technical reasons it is segmented in 6 sectors with 3 panels each. To minimize multiple scattering and photon conversion the panels are made from a low Z material (pure carbon, $Z = 6$). They are machined to a thickness $d = 2mm$, polished, and coated with a thin $Al + MgF_2$ layer. The average reflectivity obtained is $R \sim 80\%$.

To cope with the high event rates the photon detector consists of 6 multiwire proportional chambers with cathode pad readout and a solid CsI photon converter evaporated onto the pads. Reflective layers of CsI have been proved to allow efficient VUV-photon conversion in gaseous detectors. In the wavelength region accessible with the CsI converter, the C_4F_{10}-radiator shows no relevant emission of scintillation light despite an energy deposition $\Delta E \simeq 350MeV$ per central collision. Great emphasis was put on the proper choice of the radiator container material ($d = 0.4mm$ carbon fiber composite) and the mirror substrate to guarantee a low mass surface density.

1.3.4 The magnetic spectrometer: MDC and ILSE

HADES has a magnetic spectrometer consisting of a superconducting toroidal magnet (ILSE) and 24 multi-wire drift chambers (MDCs) in 6 sectors

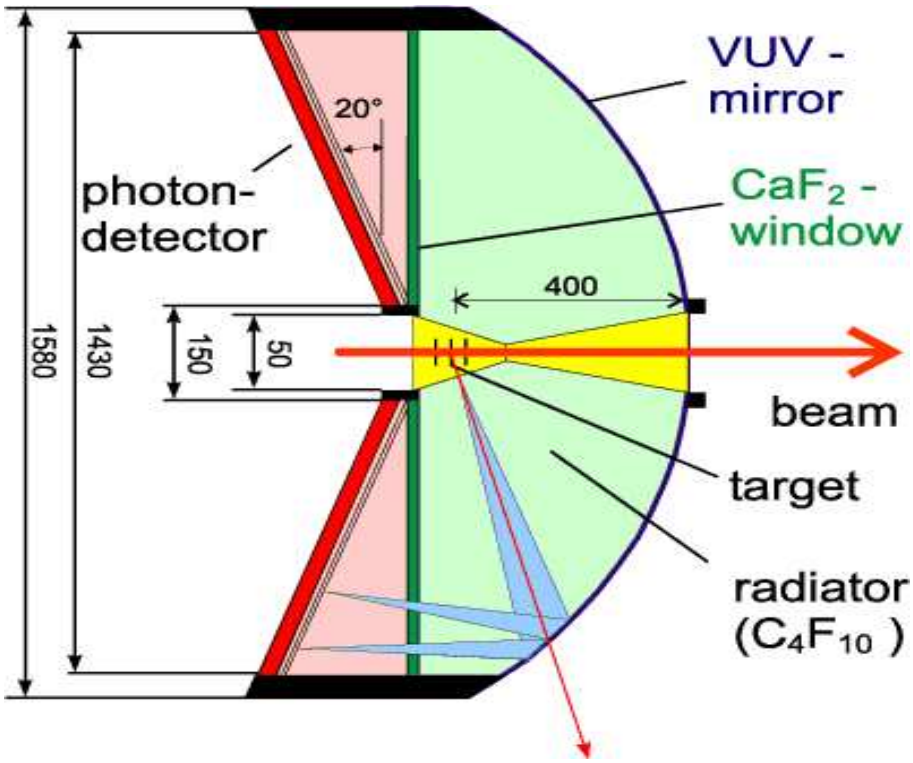


Figure 1.6: *Side view of the RICH detector*

(see fig. 1.7).

The magnet consists of 6 superconducting coils separately mounted in 80mm thick boxes, producing an inhomogeneous magnetic field which reaches a maximum value of 3T near the coils, but produces a maximum field around 1.5T in the acceptance region. The momentum kick ranges from 40 to 120 MeV at full field.

The 4 drift chambers in each sector are divided in two groups, with two chambers before the magnet and another two after. The toroidal shape of the field allows to confine it in the region between the two groups of chambers. The frames of the chambers are situated in the shadow region defined by the magnet's coils so that the acceptance is not further reduced. The size of the dritf cell in the chambers ranges from 2.5 to 7 mm, keeping a granularity high enough to deal with occupancies in the order of $0.6cm^{-1}$ (in the low polar region). Each chamber has 6 layers of sense wires with respective orientations $+40^o, -20^o, 0^o, 0^o, +20^o, -40^o$ providing enough redundancy in the measurements. This wire orientation is chosen to optimize the position resolution in the direction of the momentum kick, thus maximizing the res-

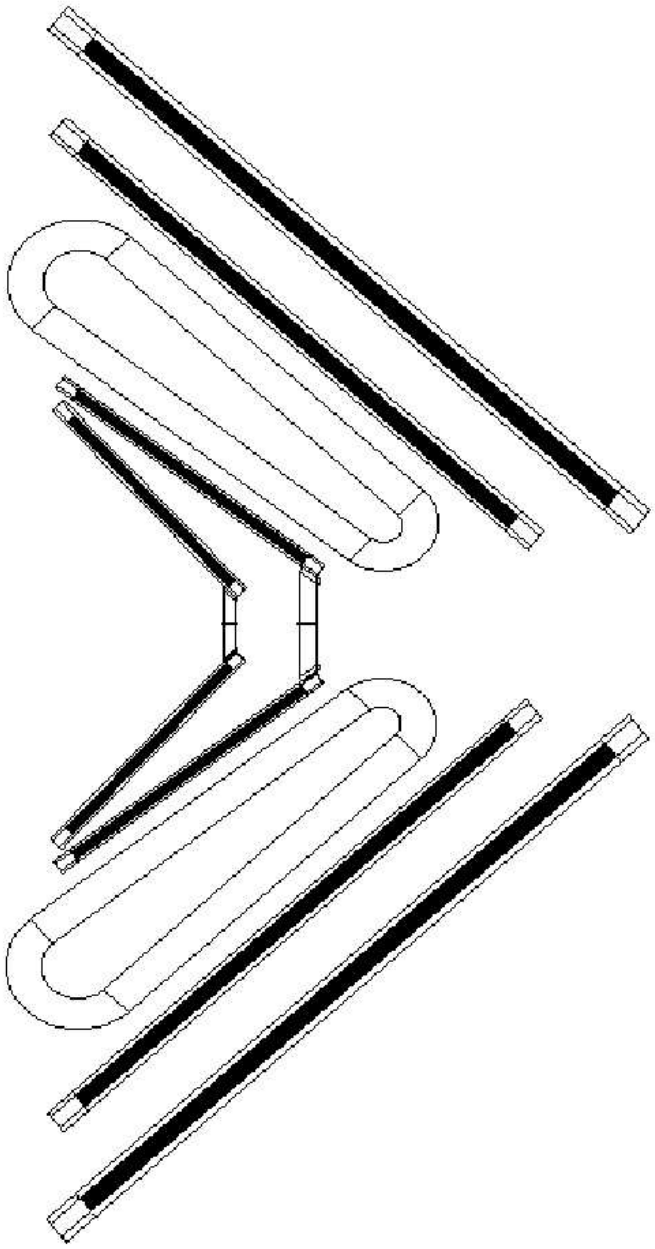


Figure 1.7: *Transversal cut showing two opposing sectors of the HADES magnetic spectrometer*

olution in momentum. Besides, the two 0° layers are displaced by half a cell in the direction perpendiclular to the wires. Fig. 1.8 shows an schematic

view of the wires configuration inside the chamber.

It is also necessary to minimize the effect of multiple scattering. That effect is dominant over the position resolution for momenta below 0.4GeV. To minimize it, low mass materials have been chosen in the manufacturing of the chambers: the Golden Tungsten sense wires have a diameter of $20\mu m$ and the field and cathode wires are $12\mu m$ thick; the gas mixture used is $He - iC_4H_{10}$.

Drift velocity for the mixture saturates around $4cm/\mu s$ for the used voltages, managing a nearly linear relationship between dritf time and track position in most of the cell. The average resolution for a single hit is around $80\mu m$ over more than 80% of the cell.

In order to handle the very fast data acquisition of the Drift Chambers, an specific TDC has been designed. Those TDCs are based in technology of 0.6μ and they are able to work at 25 MHz.

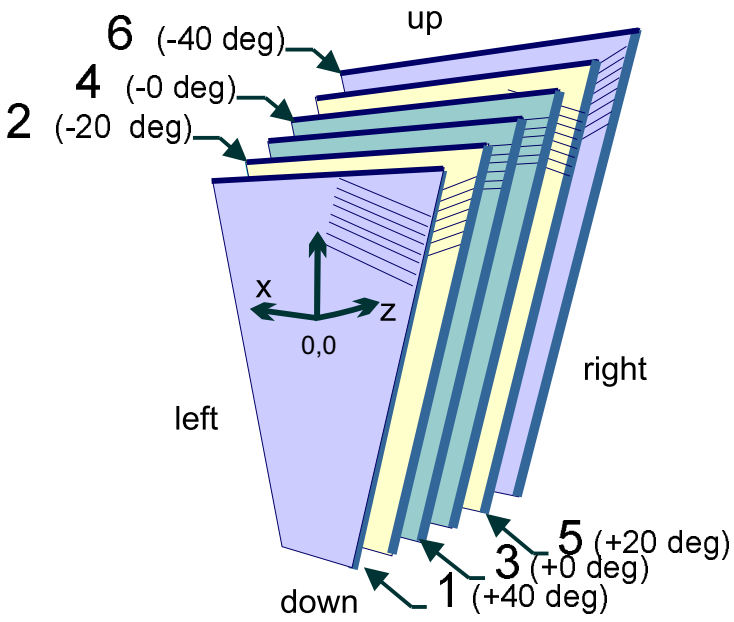


Figure 1.8: *Configuration of the six sense wire layers in an HADES drift chamber. The two central layers have parallel sense wires, but stagged by half a pitch*

1.3.5 TOF (Time-Of-Flight) detectors

TOF and TOFino detectors cover the complete spectrometer acceptance, meassuring the time-of-flight of the charged paticles (relative to a given start

detector time). The TOF main tasks are:

- Measure the charged particle multiplicity, in order to trigger on the centrality of the colission (first-level trigger).
- Perform electron/hadron identification and participate in the second-level trigger.
- Characterize the full event in the analisys phase, identifying all charged particles using the time-of-flight information.

The TOF detector is made of plastic scintillator rods (BC408) read out at both ends by EMI 9133B photo-multipliers. This double reading allows the reconstruction with a good accurancy of the time-of-flight from the target ($\sigma \simeq 100 - 150ps$) and the hit position along the rod itself ($\sigma \simeq 1.5 - 2.3cm$). Eight rods are included in each carbon set, and eight carbon sets are assembled in each sector, covering the polar angles between 85º and 45º. The pads profile is a $2.0 \times 2.0cm^2$ rectangle for the inner four carbon sets and $3.0 \times 3.0cm^2$ for those at larger polar angle.

The TOFino detector is made of four scintillating planes covering the low polar angles up to an angle close to 45º. Only one photo-multiplier is used in the case of the TOFino. The substitution of the TOFino detector by a RPC (Resistive Plate Chamber) wall is now under development, in order to improve dramatically the granularity, to deal with heavy ion reactions

From the multipiers signals not only a time is obtained, but also an amplitude. TDC and ADC are integrated in the same VME motherboard that hosts 32 channels and implements zero suppression and a fast VME block transfer. For the second trigger unit, two TDC and two ADC values per scintillator strip and event must be first calibrated; then position and time-of-flight information must be calculated and corrected. This task is performed in an array of six digital signal processors (SHARC) containig 32 floating point computing units.

1.3.6 The SHOWER detector

Due to the larger momenta of particles at low polar angles, the lepton identification based only on time-of-flight measurements is not enough. Consecuently the SHOWER detector is placed in the external part of the spectrometer, at low polar angles (behind the TOFino detector), with the main task of improving the lepton/hadron discrimination.

The SHOWER detector is divided in 6 sectors forming a trapezoid shape around the beam line. Each sector has 3 wire chambers (preshower and two

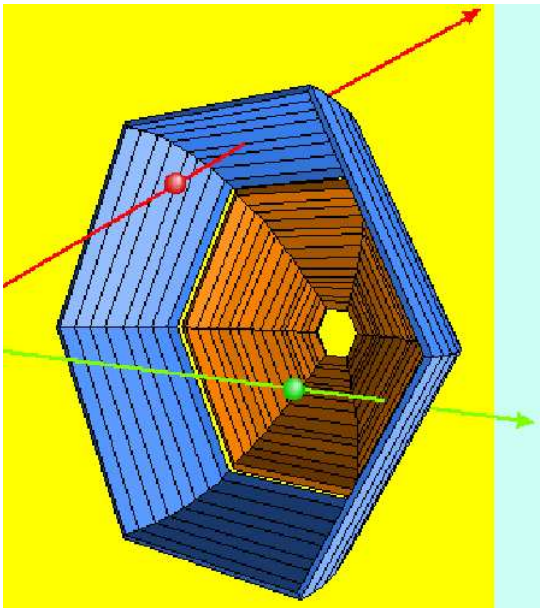


Figure 1.9: *3D view of TOF and TOFino detectors. The hexagonal symmetry is apparent*

postshower) read by pads and two lead converters 1.2cm wide, as it is shown in Fig. 1.10. The width of the lead converters is optimized to maximize the probability of producing an electromagnetic shower for leptons in the HADES energy regime, while keeping to tolerable levels the chances of an hadronic one.

Pads are organized in rows and columns. There are 32 pad rows and a number of columns ranging from 20 to 32, making up for a total of 942 cells. The pad's height ranges from 3cm to 4.5cm. Each pad covers an integer number of drift cells. These pads are aligned in each sector with respect to the target such that there is a one-to-one correspondence between pads in a particular row and column in all three detectors. The shapes of the pads have been optimized to fulfil two conditions:

- to minimize pad double hit probability
- to maintain reasonable area for shower integration

The basic idea of lepton identification is to compare multiplicity of charged particles before and after a lead converter. This is implemented via charge measurement in a mode called 'Self Quenching Streamer' (SQS) with readout of the charge at the pads of detectors. The main advantage of operating in

SQS is that induced charge is nearly independent of the particle energy loss and the particle mass. A simple comparison of the charge before and after the lead converters allows to discriminate between leptons with electromagnetic shower and hadrons wich do not produce such shower.

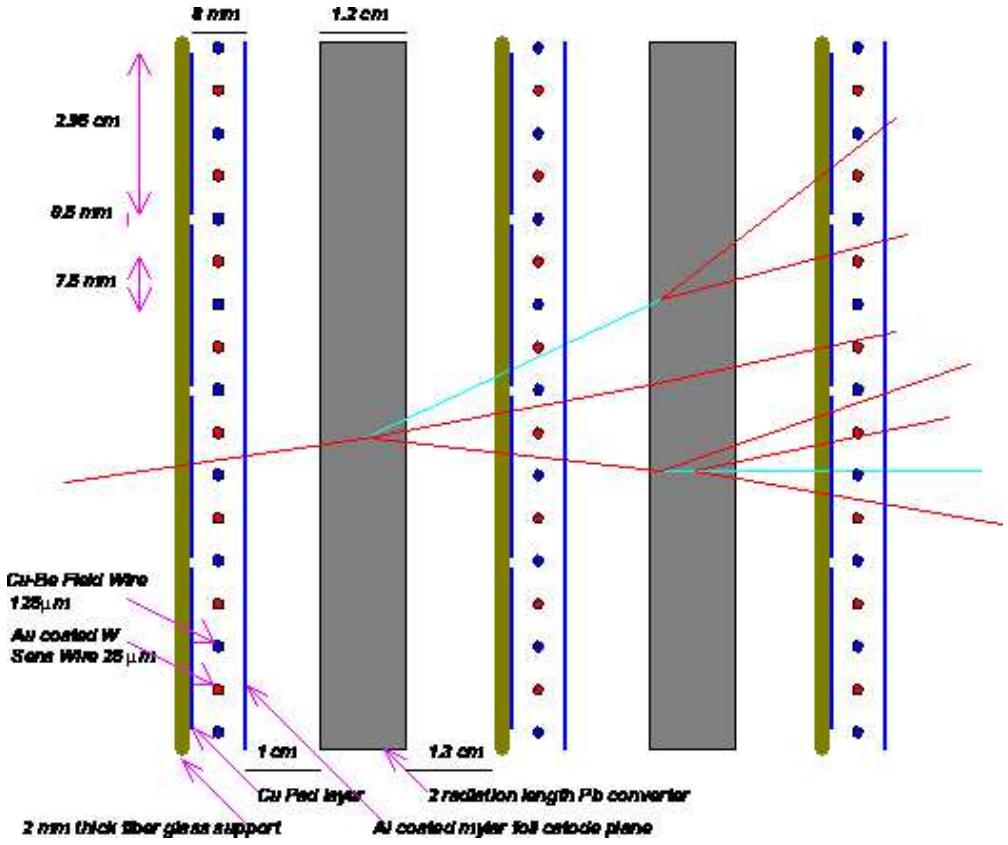


Figure 1.10: *Side cut of the shower detector*

1.3.7 The trigger scheme

In experiments conducted with the HADES detector, event rates up to 10^6 Hz must be handled. This requires a trigger system able to reduce the event rate by up to a factor of 10000 by preselecting interesting events with relevant signatures. This selection of relevant dilepton events is performed in a three level trigger system.

The first level trigger (LVL1) selects the 10% most central heavy ion collisions via the multiplicity of charged particles measured in the META

scintillator array. A positive first level trigger initiates the readout of all detectors at a rate of up to 10^5 Hz (reduction factor of 10).

The second level trigger (LVL2) consists of two stages. In a first step a search for electron signatures is performed by looking for ring images from Cherenkov light on the RICH pad plane. Moreover, clusters with the signature of an electromagnetic shower on the three planes of the META shower detector as well as particles with appropriate time of flight in the scintillator detector are selected.The resulting position information from this first stage is correlated between the inner RICH and the outer META, taking into account the bending in the magnet field between the two detectors and thereby applying a selection on the particle momenta. This task is performed in the Matching Unit (MU). Two such valid candidates with a minimum opening angle are considered to be a valid dilepton candidate with a sufficiently high invariant mass which initiates a positive second level trigger signal. The MU must provide a second level trigger decision in less than $10\mu s$ on average. The second level trigger reduces the event rate by about a factor of 100 with a latency time of about 15 events ($150\mu s$).

The third level trigger (LVL3) performs a consistency check of the potential electron candidates determined in the second level trigger evaluating the hit pattern of wires from the MDC modules. The electron hit positions determined in the Second Level Trigger both for the RICH and the META detector define regions of interest in the MDC modules. To discard events with uncorrelated hits in RICH and META, the pattern information of hit wires from the MDCs without any drift times is used. The regions of interest must be determined from a simple approach which assumes a single kick plane in the magnet and basic logical correlations must be evaluated for the corresponding wire pattern. The third level trigger gains a reduction factor of 10.

The communication between the different detectors is performed via a dedicated trigger bus between a central unit (CTU) for each trigger level and several detector trigger units (DTU). The bus distributes the three level trigger decisions, Event -ID's, trigger codes and detector busy- and error conditions. The CTU distributes the trigger decisions, generates event IDs and handles event types as well as busy and error conditions. The DTUs are responsible for the handling of the incoming HADES trigger bus signals and control the various readout components via local readout system trigger buses.

With the three trigger levels active about 100 events per second (4 MB/s) would finally be written to tape.

1.3.8 The Data Acquisition system (DAQ)

The acquisition system for HADES is based on an ATM network connecting the VME crates, used for the detector readout and second level trigger, with a central event builder CPU. The data taping speed in the event builder is 5 MB/s, corresponding to up to 2000 events per second on a C+C collision or around 100 events per second for the heavier systems.

The DAQ system uses a two pipe architecture. The first pipe of the readout system is filled with data sampled on each valid first level trigger. In the first level pipe a trigger tag is distributed for each subevent. Trigger tags are used to simplify identification of faulty modules with errors in the data transmission. The data remains stored until a decision of second level trigger is available. After a positive decision, the event data are written to the second level pipe. A negative decision initiates the removal of the event from the first level pipe without storing it. The second level pipe is implemented in the RAM memory of the VME bus system, being directly mapped in their address space for direct access by a fast data transport interface to the event builder. The CPU controllers for the different sector subsystem are accessible via Ethernet for configuration, diagnostic and error handling purposes.

Chapter 2

The HADES software

As we said previously, the main goal of the HADES experiment is the study of in medium modifications of vector meson properties, through their decay in lepton pairs. The HADES spectrometer has been built in order to achieve that goal, but we also need software enabling us to make use of the machine and transform the low level information provided by the detectors into some high level data, with higher level physical content. We can divide the software used in HADES in four large working areas:

- Data acquisition and monitoring
- Simulation
- Event reconstruction
- Physics analysis

In this chapter we will treat first the event reconstruction software; later, we will treat briefly the analysis strategy, and finally we will concern ourselves with the momentum reconstruction software, one of the parts of event reconstruction.

2.1 The HYDRA event reconstruction software

HYDRA[1] [SG99], [SG03] is the *framework* for the event reconstruction program. We can say a framework is a set of rules, interfaces and services put at the disposal of programers, who can extend it to perform a set of tasks.

In the case of HYDRA, the main goal is the processing of events recorded in the spectrometer. That is, we read input data and those data are processed by a set of algorithms wich depend on parameters and need access to the data in some structured way. As a result, new, elaborated data, are produced.

The reconstruction proceeds in steps. Each algorithm reads some input data, maybe the output of another algorithm, and takes it to a new level of elaboration. In that sense we can speak of *data levels*, wich correspond to the different levels of elaboration.

HYDRA is written in C++, an Object Oriented Programming language. Object oriented represents a new paradigm for software modeling, including the design, evaluation and implementation and enforces the code modularity, reusability and readability. These features allow the management and maintenance of large and complex programs, made by distant collaborators. The concept of the object, a significative representation of a real or abstract entity, wich is the element of a class composed of a set of related entities, is a semantic aproximation, closer to our minds organization.

The three fundamental characteristics of this paradigm are *encapsulation*, *polymorphism* and *inheritance*:

The encapsulation is related with the concept of object and it is refered to that the code is divided in closed entities with a full semantic sense (classes). Basically, classes tell us how to create an object. They are represented as a set of data members and a set of functions or methods. Encapsulating the implementation of these elements, being *private* (not accessible to non-class methods), allows the modification of the internal code and the correction of errors without any impact in another parts of the code.

The inheritance is used to specialize and generalize concepts, reducing the code volume and systematizing the data organization within the code. We say that a daughter class inherits from a parent class when the first one implements all methods (functions) of the parent and contains all its data members.

[1] Hades sYstem for Data Reduction and Analysis

The polymorphism allows the use of identical semantics applied to the functions of different classes which inherit from a common parent. In other words, we do not need to know if we are treating with parent or daughters classes, we can have different behaviors but using always the same interface.

HYDRA software is based in ROOT, an Object Oriented Data Analysis Framework [BR97], [ROO]. ROOT has been developed at CERN and is specially designed for its use in High Energy Physics experiments. It is made by a set of frameworks and a C++ interpreter (CINT is the C++ interpreter used by ROOT). Main features ROOT has are:

- Provides an user interface
- Provides several systems for storing objects into output files
- Makes graphics and histograms
- Generates documentation automatically from code coments

As disadvantage, ROOT interpreter is not fully compatible with C++. It imposes some restrictions in the software that users can develope in HYDRA.

2.1.1 System overview

HYDRA touches 5 main areas:

- Data Input; either raw or partially reconstructed
- Data Output in a suitable form for further analysis
- Management of algorithm's parameters, like geometry, calibrations...
- Data structure
- Managemet of algorithms

Besides that, we also need a component to coordinate the different subsystems. Each subsystem is defined by a set of classes wich are explained brieflly in the following paragraphs.

The fundamental HADES class

`Hades` is the main and central class in HYDRA. It encapsulates the whole reconstruction program and is the responsible for activating the different subsystems in the framework, provide access to them and coordinate them.

`Hades` class has several significant components like a data source where to read event data from, objects storing the tasks to perform for each event type (`HTaskSet` objects), an `HEvent` object where to store the information being procesed, a `HSpectrometer` object, created during initalization, wich contains information about detector's structure, a database where to read parameters from and outputs like files and ROOT trees.

There are two ways to acces that object. By the one hand we can use the `gHades` global pointer. It is a simple global variable but it has the problem that users can instantiate the Hades class twice by mistake. By the other hand we can use the Hades::instance() method. It gives back a pointer to the current Hades object with the first advantage that the instantiation is always done once, without user intervention.

Classes to contain data: data structures

Since the reconstruction process is done event per event, the basic data structure is the **event**. It can be either real, simulated or a calibration event, and it can contain both the original data coming from the spectrometer (called *raw* data) and more elaborated data which is the result from the reconstruction process.

We only need to store one event in memory an any given time. The event structure is made accessible through the `HEvent` interface. Each `HEvent` object represents all data relative to one event. Hades class is the class in charge of keeping one `HEvent` object in memory at all times, that is the current event.

The `HRecEvent` class implements the `HEvent` interface for physical events wich can be totally or partially reconstructed, and `HPartialEvent` objects give us information about parts of an event under reconstruction. As we can suspect, `HRecEvent` and `HPartialEvent` classes inherit both from `HEvent` class.

Conceptually, an `HEvent` is nothing else but a container for categories, providing access to any category in the event. A **category** is an object container. Main property of categories is that all objects within a category have to instantiate exactly the same class.

There are several possible implementations for a category depending on how the data objects are internally stored. However, all categories de-

rive from the `HCategory` class, which is abstract. `HCategory` defines the interface that must be implemented by the different categories implementations. Such interface provides functions to access the stored data objects. This access can be either random or sequential (invoking the HCategory::makeIterator() method). Data objects can be either stored in a matrix-like structure (`HMatrixCategory`) or in a simple linear structure (`HLinearCategory`).

As an example, in order to access one category we would write something like:

```
HEvent *event = Hades::instance()->getCurrentEvent()
HCategory *cat = event->getCategory(catKickTrack)
```

In this case, `catKickTrack` contains data objects from one of the momentum reconstruction algorithms.

Data input and output

As we know, HADES is an spectrometer composed of a set of different kind of detectors, giving each detector different kinds of information, that is, different data sources. In order to deal with those different data sources wich are possible, a generic interface called `HDataSource` is defined. That interface is used by the `Hades` class to request any particular implementation for event data. Different implementations of `HDataSource` are realized through daughter classes.

When having data from the Data Acquisition System, `HLdSource` is the base class for those data sources, reading data in the format produced by the HADES DAQ. This class reads raw data and place it within the event structure. This process is known as *unpacking* and is realized by unpackers within a `HLdSource`, objects of a class inheriting from `HLdUnpack`. Each detector has its own unpackers.

If we treat with partially reconstructed data, the data source is a ROOT file holding an event tree. The trees are usually generated by the reconstruction program itself holding completely or partially reconstructed events. Now, the `HRootSource` class is able to dynamically select if the whole event from the input file should be read, or just part of it.

After event reconstruction, we need a method to put the data from the data containers into output files. This output system is based on the ROOT Input/Output methods, with fuctions to store objects in the so called ROOT files. ROOT's TTree facility is used. This structure is specialized in storing multiple objects of the same class in a friendly way for further physics analysis in the so called trees. Mainly, they are objects storing arrays of other objects.

Each tree is organizated in branches; for each data member of the object's class a branch is created.

Reconstruction parameters management

In order to do the event reconstruction, several numeric parameters are needed: calibration, alignment... Each of these sets of numbers is represented by a subclass of `HParSet` class known as parameter container. The interface class responsible for managing parameter sets in HYDRA is the `HRuntimeDb` class. It implemets a runtime database, which has to manage parameter input and output. Each container in the runtime database is identified by a name: `MdcSetup, ShowerCalPar, KickPlane2Meta...` The method `HRuntimeDb::getContainer()` can be used to retrieve a pointer to a container given its name.

Each set of parameters can have different versions since they can change with time and may come from different sources, each source being implemented as a subclass of `HParIo` class. There are three kinds of sources, each source with its corresponding subclass. They are:

- `HParOraIo`: Interfaces to a database based in ORACLE (version 10.1 nowadays) residing at the central cluster at GSI. We need network conection to use it but we are sure that we are getting the canonical set of parameters at any given time
- `HParAsciiFileIo`: Allows storage and retrieval of parameters from ASCII files. We only can store one version of the parameters per file
- `HParRootFileIo`: Implements storage and retrieval of parameters from binary ROOT files. They can contain different versions of parameters per file but it is not easy to modify them in ROOT files

When we have a set of events taken in a time range where parameters have not changed, we define a *run*. Each run has an unique identifier. The parameter management system allows us to choose either the avaliable or more suitable parameters versions for any given run.

Algorithms management

We need a system which allows us to select wich algorithms to use for event reconstruction as well as the way they are combined. For this purpose, it is called *task* to a process executed event per event, and each task is implemented as subclasses of the abstract class `HTask`. This interface has

a special function, called `HTask::next()`, to execute each task and also to return a pointer to the next task to be executed.

There are two kind of predefined tasks in HYDRA, the `HReconstructor` and the `HTaskSet`. The first is the one subclassed by developers to implement new algorithms in HYDRA. This class implements the `next()` method and defines a new function, `execute()`, to be overridden by subclasses. This function is the one which gets called by the framework for every event. `HTaskSet` allows to group several tasks together, adding tasks (`HTaskSet::add()`) or connecting tasks (`HTaskSet::connect()`).

The `Hades` class has several predefined task sets: for simulations, for real data, for calibration etc.

HYDRA's initalization

We will see now how the different systems interact during the program's initialization. This initialization is doing from settings given by the user in a ROOT macro. This macro is written in C++ and interpreted in runtime by CINT. This settings instantiate classes and call their functions.

The basic settings given by the user are:

- Spectrometer configuration; that is, what detectors are going to be used and even which parts of them
- Data base initialization, giving inputs, up to a maximum of two, which are going to be used for reading reconstruction parameters and also what output to use for parameters
- The source where the data will be read from
- The list of tasks to be performed
- The output file to use, if any, and which part of the event structure should go to output
- Optionally, the type of event and the types of categories used to store the data

An example can be seen in listing 1.

2.2 HGeant simulation package

HGEANT [GEA93] is a simulation package for HADES written in FORTRAN and built upon the GEANT-3 program from CERN. The purpose of

Listing 1 Example of initilization macro

```
//
//Setup spectrometer, in this case the Mdc Detector
HMdcDetector *mdc = new HMdcDetector;
Int_t module[4] = {1,1,1,1};
gHades->getSetup()->addDetector(mdc);
//
//Set the data source from a root file
TString inFile = ''myData.root'';
HRootSource *data = new HRootSource;
data->addFile(inFile.Data());
//
//Set runtime databse to read from Oracle and from a ROOT file
HRuntimeDb *rtdb;
rtdb->setFirstInput(new HParOraIo);
TString parFile = ''myPar.root'';
HParRootFileIo *input = new HParRootFileIo;
input->open((Text_t*)parFile.Data(),''READ'');
rtdb->setSecondInput(new HParRootFileIo);
//
//Setup task list:  KickPlane reconstuction
HTask *kickTask = HKickTaskSet::make(''lowres'',''simulation'');
//Adding task
HTaskSet *task = gHades->getTaskSet(''simulation'');
task->add(kickTask);
//
//Set output file
gHades->setOutputFile(''test.root'');
//
//Instruct the framework to initialize
gHades->init();
//
//The output is set up
gHades->makeTree();
```

HGeant is to simulate the detector response of the HADES spectrometer to the passage of charged particles.

HGeant does not simulate any ion collision. For that purpose, we need external programs, called event generators, like PLUTO or UrQMD. HGeant reads information given by event generators and propagate such events through

the HADES spectrometer. But we are also allowed to generate these *tracks* by hand, using the C++ iterpreter embedded in HGeant, either in an interactive session or with a macro.

HGeant provides a way to read the geometry describing the spectrometer, including both the geometrical shapes and material composition of detectors. It makes that we can simulate the different physical effects affecting particles passing through matter, like energy loss, multiple scattering, secondary particles production, bremsstrahlung emission, etc.

HGeant mostly records information about the particle transversing the different detectors and uses the HYDRA facilities to write its output. Then, we can read directly in our analysis the files generated by the simulation. It also makes possible to compare the output of the full event reconstruction in HYDRA with the original HGeant input and study the differences to evaluate the reconstruction's quality.

2.3 Analysis strategy: the DSTs

When a set of files with raw data, taken during an specific beamtime, is ready to be processed, (all neccesary algorithms and parameters are available) starts the physics analysis itself. A process called Data Summary Tape, DST [HAD]. Each beamtime has its own DSTs, divided in *generations*. It is expected to have better parameters and improved algorithms per each DST generation.

DSTs start with files containig raw data, *hld* files, stored in a tape robot, and end giving plots with the physical information. The execution of the programs is controlled via shell scripts using the batch farm at GSI. Figure 2.1 shows how the DST analysis is processed. The DST_batch&&.sh scripts calls a second script : hydra_batch.sh; this first script transport the hld files from the tape robot and get them ready to be processed. The second script executes the analysis of the hld files on the batch machines and if the analysis is completed successfully, processes the diagnostic of the analyzed file and places the diagnostic plots directly on the web.

The analysis is processed via analysis macros that contains the necessary parameters and the task list that one wants to run. The diagnosis plots are needed to check in a reasonable fast way the quality of the analysed DST files. The diagnosis macros produce control histograms for all the detector hit level (inlcuding a mass spectrum).

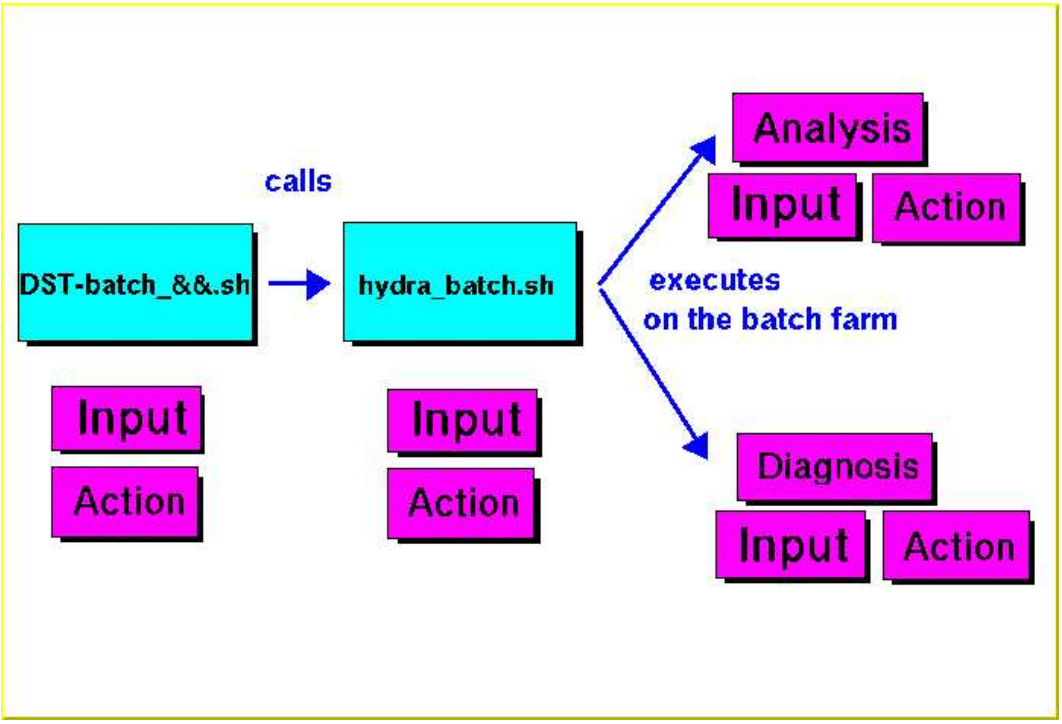


Figure 2.1: *Schematic view of the DST analysis process*

2.4 Momentum reconstruction algorithms

In this section we will deal briefly with the HADES' problem of obtaining the momentum of particles traversing the spectrometer. We saw before that HADES main goal is to obtain the invariant mass of dilepton pairs coming out of heavy ion collisions. For this purpose a reliable momentum reconstruction is needed, since a dilepton's invariant mass directly depends upon the momentum of the participating leptons through the deffinition

$$M = \sqrt{(E_1 + E_2)^2 - |\mathbf{p}_1 + \mathbf{p}_2|^2}$$

where $(E_i, \mathbf{p}_i)$ is the quadrimomentum vector for lepton i.

The information on particle momentum has many other applications like trigger, PID[2], etc, and it is obtained from the particle's movement in the magnetic field, which requires measuring the direction before and after the magnet. The main problem is that our magnetic field, provided by the toroidal superconducting magnet, is inhomogeneous, making more difficult our task.

[2]Particle IDentification

There are in HADES four momentum reconstruction algorithms avaliable up to now, which are the so called "KickPlane", "Spline Fit", "Runge-Kutta method" and "Reference Trayectories". We will describe them briefly in the next sections.

2.4.1 Kick Plane algorithm

This algorithm [SG03] is based on the deflection of a charged particle passing through a magnetic field. This deflection is proportional both to the field strength and the distance travelled by the particle in the field. Therefore one can replace the original field by another one compressed in space, but increased in strenght, such that the effect stays the same. Taking this process to the limit, the field is compressed into a 2 dimensional sheet, the *kick plane*, where all of the deflection takes place. We say that magnetic field gives the particle a *momentum kick* in the kick plane.

In our case, momentum and deflection are related by

$$p = \frac{p_k}{2\sin(\xi/2)} = \frac{A}{\sin(\xi/2)} + B + C\sin(\xi/2)$$

where p_k is the momentum kick, ξ is the track deflection in the magnetic field and A, B and C are the so called Kick Plane parameters.

The crucial point is that these parameters depend essentially on the magnet properties alone. It makes that we can parameterize them like functions of the polar and azimuthal angles that particles have in the kick plane. Once we have them, we are able to calculate momentum from particle's deflection.

This method is valid even when only inner drift chambers are avaliable, since we can meassure particle's deflection from the track in these chambers plus a point in the external META detectors. The consecuence is a worse final resolution.

The details of this parameterization, as well as some tests and improvements in this method, will be presented in the next chapter.

2.4.2 Spline Fit algorithm

This method [Rus03] is based on *spline* curves. Such curves allow us to define "trayectories" in our scenarios. We only need specify a set of points in order to obtain a curve connecting them. In the case of HADES, we asume as splines the trayectories of our particles through the spectrometer, starting from meassured points in the four MDC's (or three in case of only three are avaliable). A cuadratic curve between both pairs of chambers, inner and

outer, and a cubic curve between second and third MDC (that is, in the magnetic field region) were chosen.

Using these splines, the momentum reconstruction itself starts from equation of movement in a magnetic field

$$k(z)\left(-B_z\frac{dx}{dx}-B_y\frac{dy}{dz}\frac{dx}{dz}+B_z\left[1+\left(\frac{dy}{dz}\right)^2\right]\right)=p\frac{d^2y}{dz^2}$$

where

$$k(z)=\sqrt{1+\left(\frac{dx}{dz}\right)^2+\left(\frac{dy}{dz}\right)^2}$$

Then, it is assumed the solution

$$Y_{calculated}=C_1+C_2(z_i-z_0)+\frac{1}{P}Y_i$$

on each measured point i of the trayectory and minimizated the quality factor *qSpline* to get the momentum, imposing continuity on the solutions:

$$\chi^2=(Y_{calculated}-Y_{spline})^2=qSpline$$

The main problem of this method is that we are assuming curves close to the real trayectory but not the real one. Since we have some residual magnetic field in the chambers' region, as we can see in fig. 2.2, and we have a very inhomogeneous field, it is quite difficult estimate curves close to the real ones. It also makes that this method is very sensitive to missalignments on drift chambers.

2.4.3 Runge-Kutta method algorithm

This is the newest momentum reconstruction algorithm in HADES. It is based, as the name suggests, on the numeric Runge-Kutta method [Iva04]. This method consists in evaluate and minimize some derivatives in order to resolve the Equation of movement of a charged particle through a magnetic field. In this case, our equation is

$$m\frac{d^2\mathbf{x}}{dt^2}=q\frac{d\mathbf{x}}{dt}\times\mathbf{B}\quad\text{or}\quad\frac{d^2\mathbf{x}}{ds^2}=\left(\frac{kq}{p}\right)\left(\frac{d\mathbf{x}}{ds}\right)\times\mathbf{B}(\mathbf{x})$$

It starts from an initial track parameters vector $\mathbf{p}=(x,y,u_x,u_y,p)$ at the first MDC and calculate the optimal vector $\mathbf{f}(\mathbf{p})$, which is solution of the equation of movement, by minimizing the functional

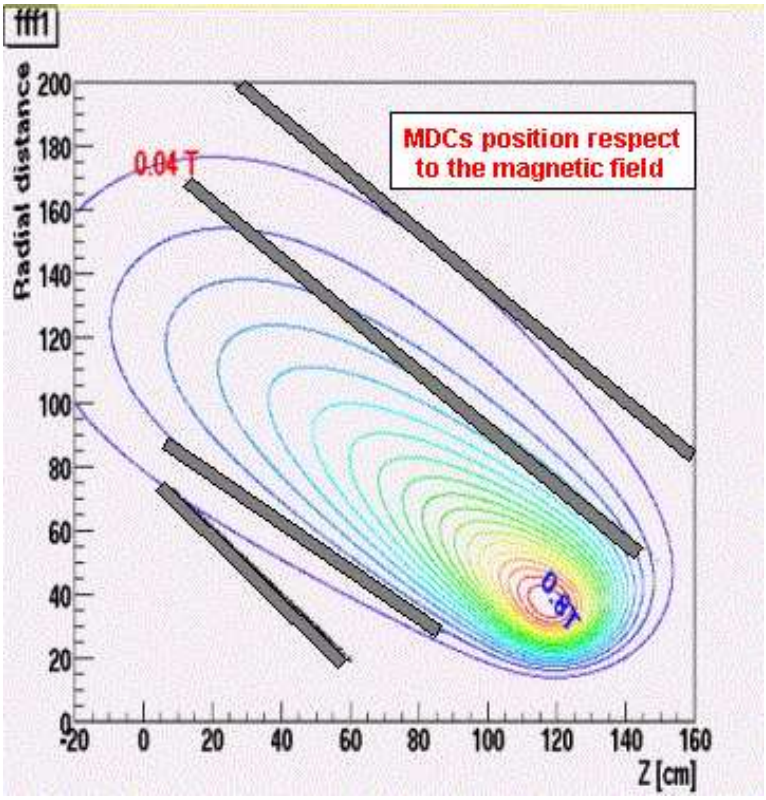


Figure 2.2: *Side view of magnetic field at center of sector. We can see the influence on drift chambers region*

$$\mathbf{Q(p)} = (\mathbf{m} - \mathbf{f(p)})^{\mathbf{T}}\mathbf{W}(\mathbf{m} - \mathbf{f(p)})$$

where $\boldsymbol{m}$ is the measurement vector and $\boldsymbol{W}$ is the weight matrix, using the Least Square Method.

This method has mainly two advantages; by the one hand, we get an "exact" solution of the equation of movement with a rather simple and short code, and, by the other hand, we get a χ^2 giving the quality of the track fitting, which can be used in further procedures like alignment or background rejection.

But it also have some disadvantages, like: rather CPU-consuming, multiple scattering not included for simplicity (up to now) and it needs an initial momentum value with some arbitrary accuracy.

2.4.4 Reference Trayectories algorithm

This algorithm is based on the use of a Monte Carlo simulation (HGeant package is used in this case), in order to built a set of all possible tracks passing through the spectrometer [SG03]. Per each track of the set we store the information given by detectors. In fact, we only store a five component vector: $\mathbf{p} = (1/p, \rho, z, \theta, \phi)$. When we want to know the momentum of a real track, it is enough to serach for the more similar one in the set of simulated tracks, and later make a linear interpolation between neighbours. This method is extensively presented in chapter 5, as well as its application to real data.

Chapter 3

Tests on KickPlane momentum reconstruction method

As we introduced in the previous chapter, the Kick Plane Method is one of the HADES momentum reconstruction algorithms. It was designed and written by M.Sanchez [SG03] and it is based on the deflection of a charged particle within a magnetic field.

Kick Plane has three different implementations depending on which different spectrometer's setups are available:

a. only inner drift chambers and none of the two outer ones are available

b. the presence of the third drift chamber

c. all four chambers are available

as well as META detectors in the three cases.

The first implementation is the so called *Low Resolution*-Kick Plane. We do not have any outer chamber, so we use the information about track directions given by the two inner ones and the position given by the META for getting the momentum from deflection. Since META detectors (Tof and Shower) are not specially designed for position measurements, we achieve the lower possible resolution for this algorithm.

The second possible setup is the presence of the third chamber, the first of the outer ones, as well as inner ones and META. In this case we have the so called *High Resolution*-Kick Plane. As we can suspect, we can achieve a better resolution in momentum reconstruction starting from deflection because we are using the tracking information of three drift chambers.

The last case occurs when the four chambers are available. Now we have the *Full Resolution*-Kick Plane. We are using all possible information given

by detectors but, due to Kick Plane is not a fitting procedure, it is not able to use redundant information, so we are losing resolution in comparison with the high resolution case. That is the reason why this case will not be discussed in this study.

The way to calculate the momentum is the same for the three cases. Such procedure as well as some behaviour tests in the method will be explained in the following sections.

3.1 Obtaining momentum

Assuming there is no electric field in the magnetic region, the motion of a single charged particle in the field is governed by the Lorentz force. Then the motion is given by

$$m\frac{d^2\mathbf{r}}{dt^2} = q\frac{d\mathbf{r}}{dt} \times \mathbf{B} \tag{3.1}$$

and defining $d\lambda^2 = |d\mathbf{r}|^2 = dx^2 + dy^2 + dz^2$, (3.1) can be written as

$$\frac{d^2\mathbf{r}}{d\lambda^2} = \frac{q}{p}\frac{d\mathbf{r}}{d\lambda} \times \mathbf{B} \tag{3.2}$$

Given the geometrical interpretation of $\frac{d^2\mathbf{r}}{d\lambda^2}$ the infinitesimal deflection of the track can be written as

$$d\xi = \left|\frac{d^2\mathbf{r}}{dt^2}\right| d\lambda = \frac{q}{p}B\sin\alpha d\lambda \tag{3.3}$$

where α is the angle between $\frac{d\mathbf{r}}{dt}$ and $\mathbf{B}$. If deflection is additive, the total deflection of a path between λ_1 and λ_2 is

$$\Delta\xi = \frac{q}{p}\int_{\lambda_1}^{\lambda_2} B\sin\alpha d\lambda \tag{3.4}$$

In the case of HADES, and most other spectrometers, the field is confined to some spatial area. Then, it can be expressed as

$$\mathbf{B} = \begin{cases} \mathbf{B}(\lambda) & \lambda_1 < \lambda < \lambda_2 \\ 0 & otherwise \end{cases}$$

Since Lorentz force is perpendicular to the momentum, $|\mathbf{p}|$ remains constant and only the momentum direction changes. Then, as we can see in Fig. 3.1, we can express the momentum variation as

$$\|p_1\| = \|p_2\| \Rightarrow p_k = \Delta p = 2p \sin\left(\frac{\xi}{2}\right) \tag{3.5}$$

and, keeping in mind that $\sin(\xi) \approx \xi$ for small angles, in differential form

$$dp_k = pd\xi \tag{3.6}$$

Then, using (3.3) and (3.6) we get

$$dp_k = qB\sin(\alpha)d\lambda \tag{3.7}$$

Now we can integrate for each particular track. In the case of HADES, $\sin(\alpha) = 1$, and also deflection takes place in one plane. Then

$$p_k = q\int_{\lambda_1}^{\lambda_2} Bd\lambda = K(\lambda_1 - \lambda_2) \tag{3.8}$$

where K is a constant, different for each track, and $\lambda_1 - \lambda_2$ is the track's path length inside the field, which depends essentially on the tracks deflection (since the larger the deflection the larger the track length). So we can make the ansatz $\lambda_1 - \lambda_2 = F(\xi) = f(\sin(\xi/2))$.

Performing a Taylor expansion around $\sin(\xi/2) = 0$ we get

$$f = a + b\sin(\xi/2) + c\sin^2(\xi/2) + O(\sin^3(\xi/2)) \tag{3.9}$$

Then, we get that momentum and deflection are related by

$$p = \frac{p_k}{2\sin(\xi/2)} = \frac{A}{\sin(\xi/2)} + B + C\sin(\xi/2) \tag{3.10}$$

We need now to realize that A, B and C depend essentially on the magnet properties alone, that is, on the spatial region being traversed by the particle. This means that we can use this equation for arbitrary tracks where A, B and C are functions of the polar and azimuthal angles.

Then, if we parameterizate A, B and C as functions of the angles, we are allowed to calculate momentum from particle deflection using Eq. (3.10). We will see now the details of such parameterization.

3.1.1 Kick surface parameterization

We advanced in the previous chapter that the easiest way to calculate the deflection was replacing the original field by another compressed in space but increased in strenght. In the limit, the field is compressed into a 2 dimensional sheet where all the deflection takes place. Then, it is enough to

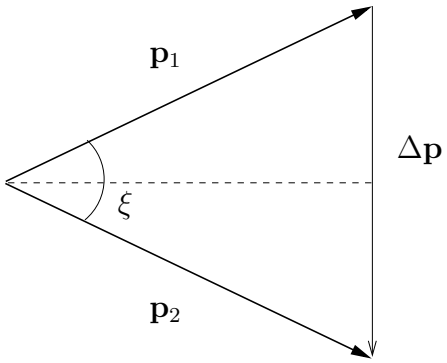


Figure 3.1: *Track deflection*

know the particle's direction before and after the point where deflection took place. Outside the magnetic field region, trayectories can be approximated by straight lines. We say that each track has two *segments*, or pieces of straight lines, one before the magnet and another one after. Such segments give us the directions we need.

We expect for two segments belonging to the same track to meet at one point. That is, since movement takes place in one plane, the corresponding straight lines should cross. If we make a 3D plot with the coordinates of all such points, we discover that they lay in a surface; this is the so called *kick surface*.

By construction of the kick surface, the intersection of any segment before the magnet with the surface gives us a point wich also belong to the segment after the magnet. Then, we do not need two segments any more in order to get a particle's deflection. It is enough with one point of the second segment.

Since that, given any track entering in the field region with a certain direction it will always hit the same point in the kick surface, if we think about the parameters which we have introduced previously, it is easy to realize that A, B and C can be parameterizated as functions of positions on the kick surface.

There is only left to calculate the kick surface. In practice, both straight lines do not cross exactly at the same point, so we need to calculate the points of closest approach between them. It is done with a LSM[1]approach using the information given by the HGeant package after running a simulation with only electrons and positrons passing through the spectrometer.

At this point we are obligated to distinguish between low resolution and high resolution cases. In the low resolution kick plane we only have the two

[1]Least Square Method

inner MDCs and the META detectors. The straight line before the magnet, lets call it inner segment, is perfectly defined by the two MDCs. The straight line after the magnet, lets call it outer segment, is constructed with the point on the fourth MDC (the closest one to META) and the direction vector at that point, both given by HGeant. Crossing all corresponding inner and outer segments we obtain the shape that we can see in Fig. 3.2 (a). Even more, since negatively charged particles are bending by the magnetic field in a different way from positively charged ones[2], we have to distinguish also between positive (like e^+) and negative (like e^-) particles. The parameterization of each surface is performed using the MINUIT package from CERN, wich comes integrated in ROOT. By direct inspection we can estimate the function to be used in this low resolution case. The surface looks like a plane but with some curvature which can be approximated by a cuadratic term in x, giving a parameterization function $y = a + bx^2 + cz$. The surface for negative and positive particles are only slightly different. So we can parameterizate both and construct the average between both values. We have to keep in mind that such surfaces can change if spectrometer's geometry does. In table 3.1 we can see the obtained values in the parameterization for the ideal detector position and the ones when the outer MDCs are moved 5 cm downstream in Z_{LAB} direction[3] (in mm).

		a	b	c
ideal position	average	1460 ± 2	$-(3.70 \pm 0.04) \cdot 10^{-4}$	-0.866 ± 0.05
outer MDCs displaced 5 cm	e^-	1486 ± 2	$-(4.32 \pm 0.05) \cdot 10^{-4}$	-0.900 ± 0.05
	e^+	1481 ± 2	$-(4.33 \pm 0.05) \cdot 10^{-4}$	-0.894 ± 0.05
	average	1483 ± 3	$-(4.32 \pm 0.07) \cdot 10^{-4}$	-0.897 ± 0.07

Table 3.1: *Parameters values for low resolution kick surface*

In the high resolution case, we have the third MDC available. Now we construct the surface with the inner segment and the straight line given by the point in the third MDC and the direction vector at this point. Since in this MDC we have some residual magnetic field (see Fig. 2.2) and the particles are still being bent such vector corresponds to the trayectory's tangent vector at this point. Then we can suspect that we are going to obtain a quite different surface from the previous case. This surface is shown in Fig. 3.2 (b). It looks like three conected planes within a V profile. The model used to parameterizate this surface is

[2]See Apendix ??

[3]Note that 5 cm is a big displacement in term of HADES' geometry

$$y = \begin{cases} a_1 z + a_2 + a_3|x| & z \leq z_1 \\ a_4 z + (a_1 - a_4)z_1 + a_2 + a_3|x| & z > z_1 \text{ and } z \leq z_2 \\ a_5 z + (a_4 - a_5)z_2 + (a_1 - a_4)z_1 + a_2 & z > z_2 \end{cases}$$

where $z_1 = 400\,mm$ and $z_2 = 1100\,mm$ are calculated by direct inspection. Following the same procedure as the previous case, the fitting yields parameters of table 3.2

	a_1	a_2	a_3	a_4	a_5
ideal geometry	-0.655	1295	0.125	-0.820	-1.119
displaced MDCs (e^-)	-0.662	1311	0.0736	-0.817	-1.413
displaced MDCs (e^+)	-0.666	1312	0.0680	-0.808	-1.481
displaced MDCs (*average*)	-0.664	1311	0.0708	-0.813	-1.447

Table 3.2: *Parameters values for high resolution kick surface*

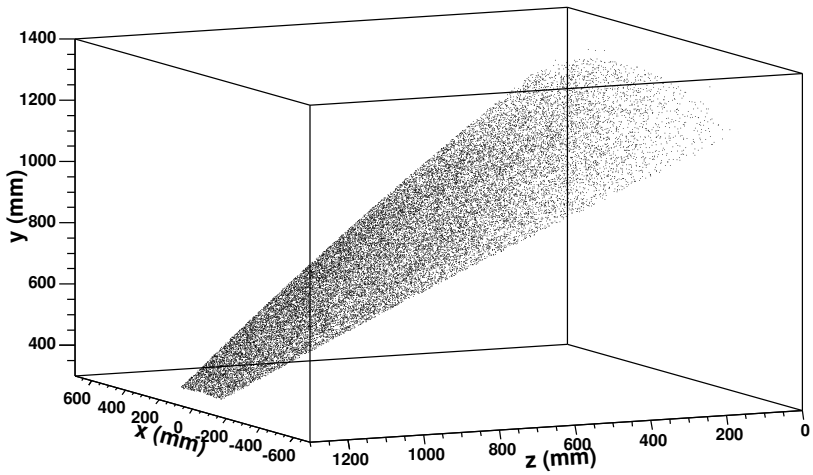
From tables 3.1 and 3.2 we can see that the effect that geometry changes and the particles' charge have on the kicksurfaces is bigger for the high resolution case. It makes think about a new parameterization, at least for this last case, per each new detector geometry in order to get a better deflection measure resolution.

Since our detector is simetric, the parameterization is always only performed for half a sector and extrapolated later to the whole spectrometer.

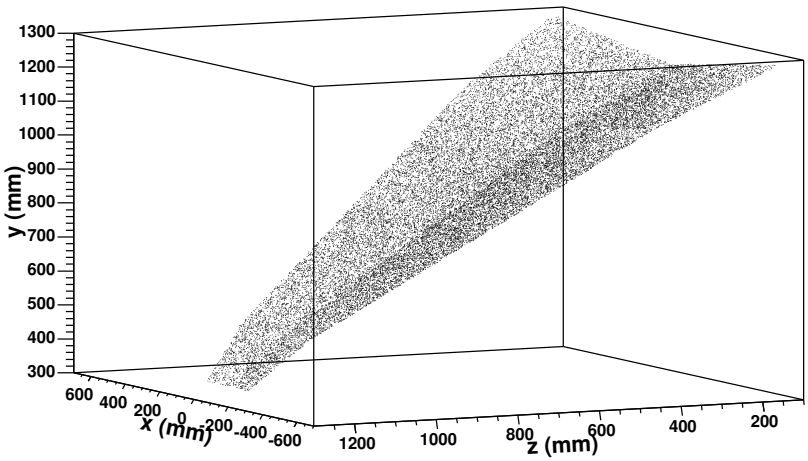
3.1.2 Kick Plane parameters' parameterization

The goal now is to analyze more deeply the form of the functions A, B and C in eq. 3.10. As noted in section 3.1.1 and since our magnetic field is not constant, is natural to think that A, B and C must depend on the geometrical region of the magnetic field being traversed. In other words, the existence of the kick surface suggest that A, B and C can be parameterized as functions over the kick surface itself.

For any given track we can compute A, B and C if we know the track's momentum and the deflection it suffered. We use HGeant, which is able to provide us with that information. For each track we record p, ξ and also a point on the kick surface. Such a point is given by the three cartesian coordinates, but since the kick surface is a real 2-dimension surface, we only need two coordinates to specify the point. This two coordinates are the well known polar and azimuthal angles from polar coordinates, θ and ϕ. Then, it



(a) Meta surface



(b) MDC3 surface

Figure 3.2: *The two different kick surfaces in the LAB system: low resolution kick plane surface (a), where we can see the cilindric shape, and high resolution kick plane surfce (b),where the 3-V-plane shape is apparent.*

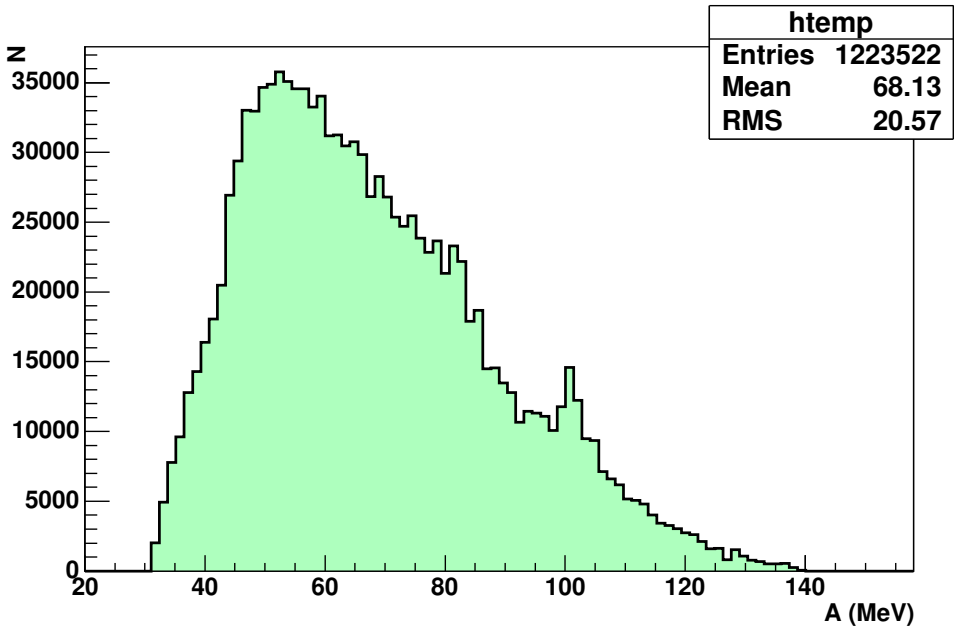


Figure 3.3: *Spectrum of the magnet's momentum kick: A integrated over θ, ϕ and p for maximum magnetic field intensity*

would be enough to shoot through HGeant a sample of particles with uniform distribution in (θ, ϕ) and covering the momentum range for the real particles. For this last purpose we use an uniform distribution in $1/p$ plus another one constant for high p.

After that we store all neccesary information and minimize, using a LSM fit, the functional

$$Q^2(A, B, C) = \sum_{i=1}^{N} w_i \left(p_i - \frac{A}{\sin\left(\frac{\xi_i}{2}\right)} - B - C \sin\left(\frac{\xi_i}{2}\right) \right)^2 \qquad (3.11)$$

We simply build a table in θ and ϕ, choosing a bin size of half a degree, both in θ and ϕ. We need al least three simulated tracks per cell and for each cell in the table we store the (p, ξ) of all tracks falling into the cell and compute 3.11. So, N is the number of tracks in a given cell, w_i is a constant weight for track i and p_i and ξ_i are its momentum and deflection respectively. In order to not use too much memory storing several tracks per cell, the fit is performed following this incremental fit: every time a new track is read the fit parameters are updated for the cell where the track falls. We can cope with any number of tracks per cell.

Once we have parameterizated A, B and C and store them per each bin we are allowed to calculate momentum. Given a real track, from the meassured inner segment we obtain the (θ, ϕ) position in the kick surface. So we know the bin where the track hits this surface. From the meassured outer direction, either with only META or with the third MDC, we obtain the deflection. Introducing this deflection in 3.10 we obtain the particle's momentum.

If we have a look at 3.10, we can easily realize that A has essentially the physical meaning of momentum kick of the magnet while B and C can be interpreted as higher order corrections in ξ. As we can see in Fig. 3.3 the mean magnet's momentum kick, at maximum field intensity, is around 70 MeV and it ranges from 30 to 140 MeV. Particles with low momenta, under (or close to) the minimum momentum kick of the field region they hit, suffer loops on their tracks and can not be well reproduced by the kick plane algorithm. We can use these numbers to apply momentum cuts in the parameterization in order to remove those particles which we are not interested in.

We can also see how the different parameters are going to depend on (θ, ϕ) (see Fig. 3.4) and how they are sensible to different regions in $\sin \xi$. In fact, A is responsible for the behaviour at the smaller deflection angles, while C dominates for the larger ones. We will see more clearly this dependences in the next section.

The parameterization procedure is the same for low and high resolution kick planes, only changing the kick surfaces, and it is performed both for pisitive charged particles and negative charged particles separately.

3.2 Dependence of kick plane parameters with magnetic field

We have described previously how the parameterization is performed. Such parameterization costs some time and memory consumption. It makes necessary the study of the behaviour of the kick plane parameters A, B and C in order to reduce the work spent on its parameterization. Due to their physical meaning, the most important factor which we should keep in mind is the magnetic field strength. During HADES' life, there was taken date with several field intensities. For example, in the so called "Nov02" beam time, we had a magnetic field of 72.15%. Per each different magnetic field configuration, a new generation of parameters was merged. We can think that we only need one parameterization, for example, with the full magnetic

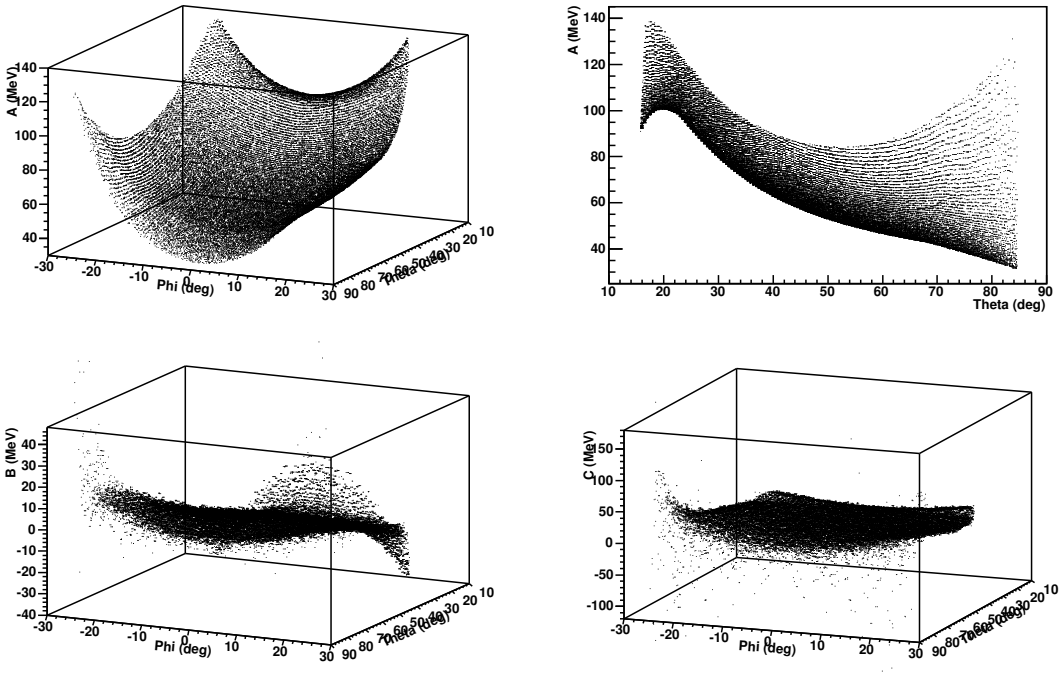


Figure 3.4: *Momentum kick A as a function of θ and ϕ, up left, and its proyection into the (A, θ) plane, up right. Down, left and right, parameters B and C as a function of θ and ϕ*

field strength, and later extend it with scale factors to other intensities. But analysis show how this task is neither necessary nor convenient.

In order to obtain the right behaviour of A, B and C with the magnetic field, 10 parameterizations were performed with different magnetic field strength ranging from 10% to 100% of full field. On each parameterization, A, B and C were calculated and tabulated following the steps explained in section 3.1.2, and they were plotted as a function of magnetic field. The results are shown in Fig. 3.5, 3.6 and 3.7. There are represented the fitted values of A, B and C in nine different regions of phase space (θ, ϕ) on the kick surface.

Since A is essentially the momentum kick, is logical to think that A is linear with magnetic field; the more field strength the more momentum kick. In fact, if we have a look at Fig. 3.5 we can see how parameter A fits perfetcly to a linear function on each region of the kick surface. But B and C are quite different. They are like corrections of higher order and their behaviour is not clear. They even have very different behaviours depending

on the position on the kick surface. For example, B contribution is higher downwards while C's is upwards. In both cases, contributions are higher near the kick surface's edges, where the momentum kick is bigger (see Fig. 3.6 and 3.7). The behaviours for low resolution parameters and high resolution parameters are the same.

It makes very difficult any kind of parameterization in order to know a priori the values of B and C for a given magnetic field strength. We need a full kick plane parameterization per each one.

3.3 Dependency of kick plane parameters with chambers' position

In the previous section we have concluded that kick plane parameterization is quite sensitive to the magnetic field. We would also like to know how sensitive it is to the different spectrometer's geometry. In principle we can think that our parameters can change as a function mostly of drift chambers' position. In such a case we need a different set of parameters per each new detector geometry.

If we spend some time looking at the parameterization procedure, we can easily realize that such procedure depends only on magnet properties. Once we have parameterizated and tabulated A, B and C as a function of positions over the kick surfaces, we do not care anymore about where the chambers were. Any ideal, simulated or real chambers' position give us the correct momentum measure since the chambers only provide us the incomming and outgoing directions in order to get the deflection.

In order to show this more clearly, three sets of kick plane parameters were generated. The first set was generated using the HADES' ideal geometry in the HGeant package. The other ones were generated with the outer MDCs displaced 2.5 cm and 5 cm downstream respectively in the Z direction in the laboratory system. With the three sets was procesed the same file of nov02 simulated data. The result is shown in Figs. 3.8, 3.9 and 3.10. They are represented the different momentum reconstruction resolutions (since we know a priori the momentum of the particles). We can see how resolutions, σ of the gaussian fit of the histograms, do not practically change: $\sigma \sim 3\%$ in high resolution and $\sigma \sim 10\%$ in low resolution kick plane. Then, we can generate the parameters once and use them later with any geometry (of course, keeping allways the same magnetic field).

We have to take into account also the effects of the so called *misalignment*; that is, the difference between the real position of the detectors and

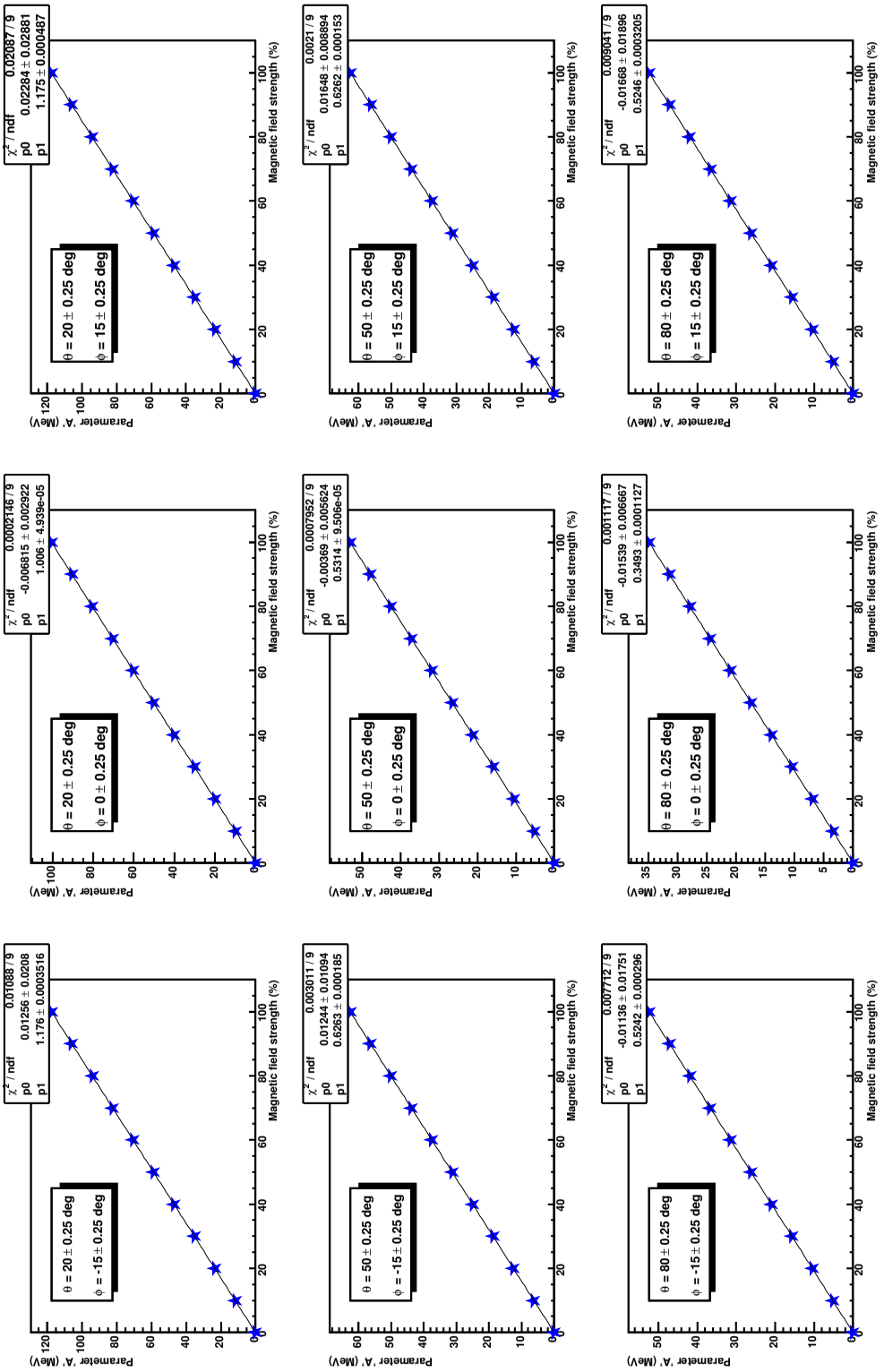


Figure 3.5: *Parameter A as a function of magnetic field in nine different regions of kick surfaces in bins of half a degree in* (θ, ϕ)

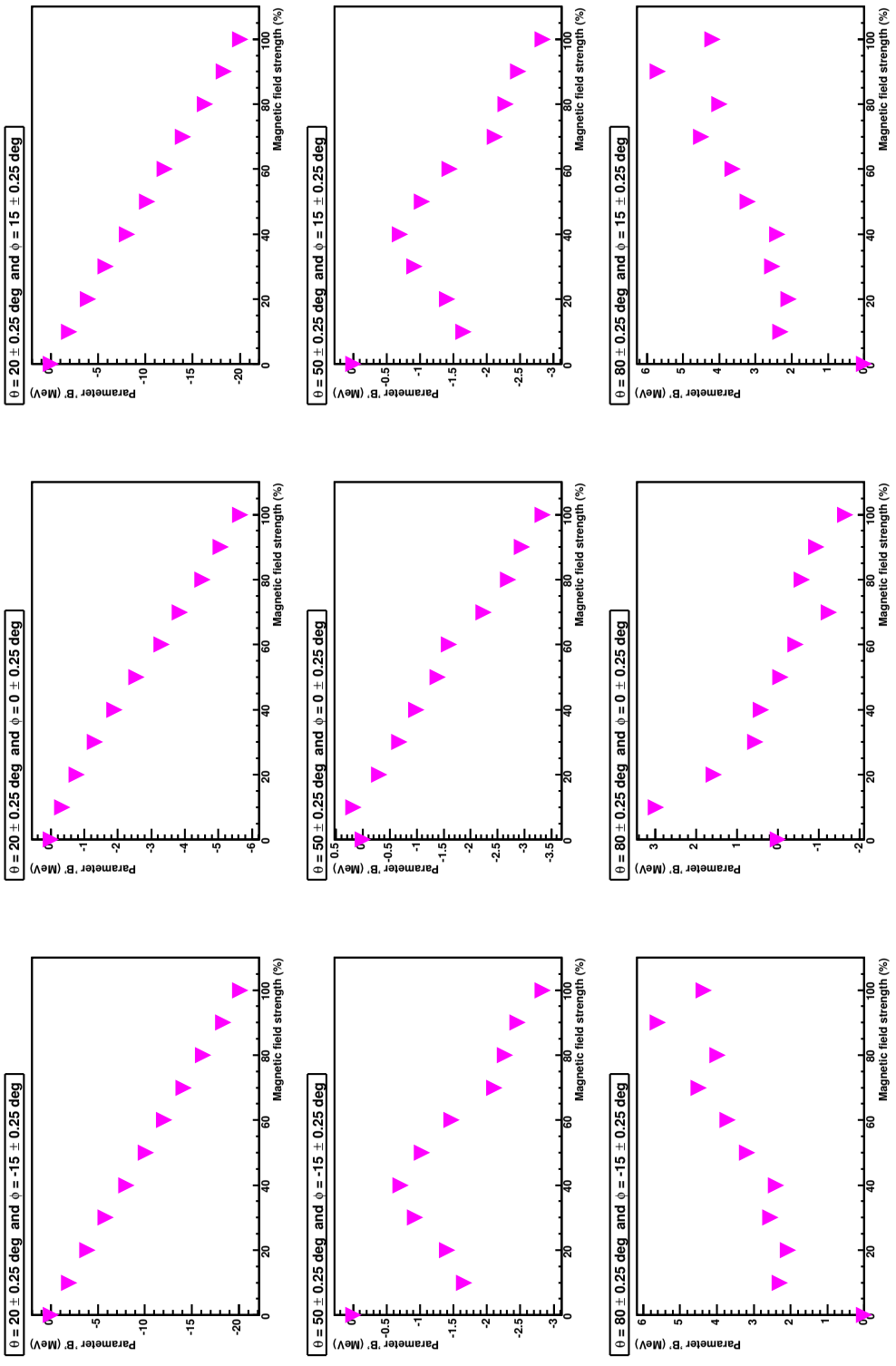


Figure 3.6: *Parameter B as a function of magnetic field in nine different regions of kick surfaces in bins of half a degree in (θ, ϕ)*

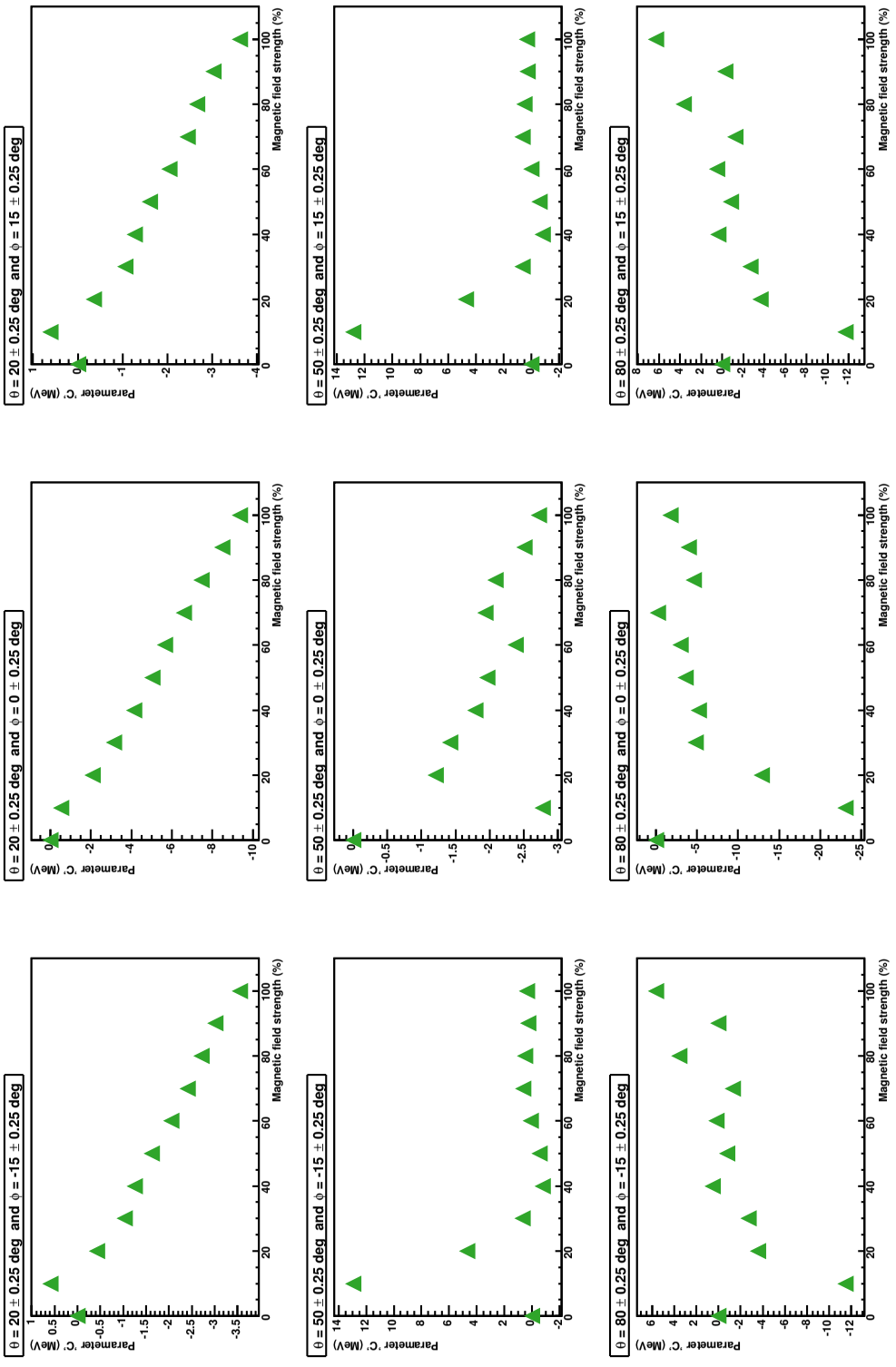
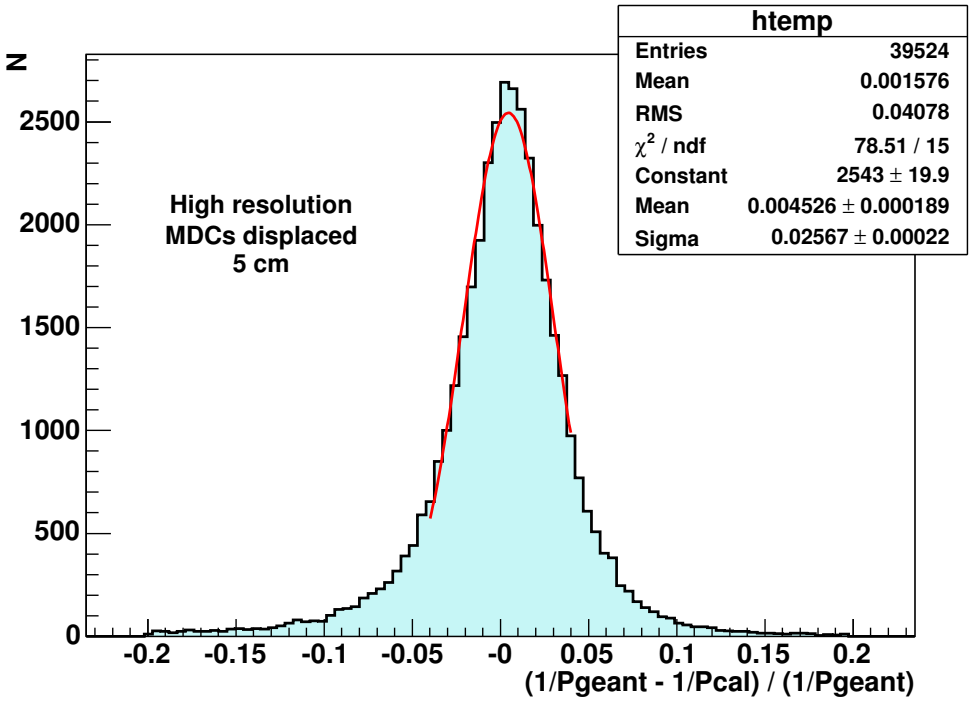
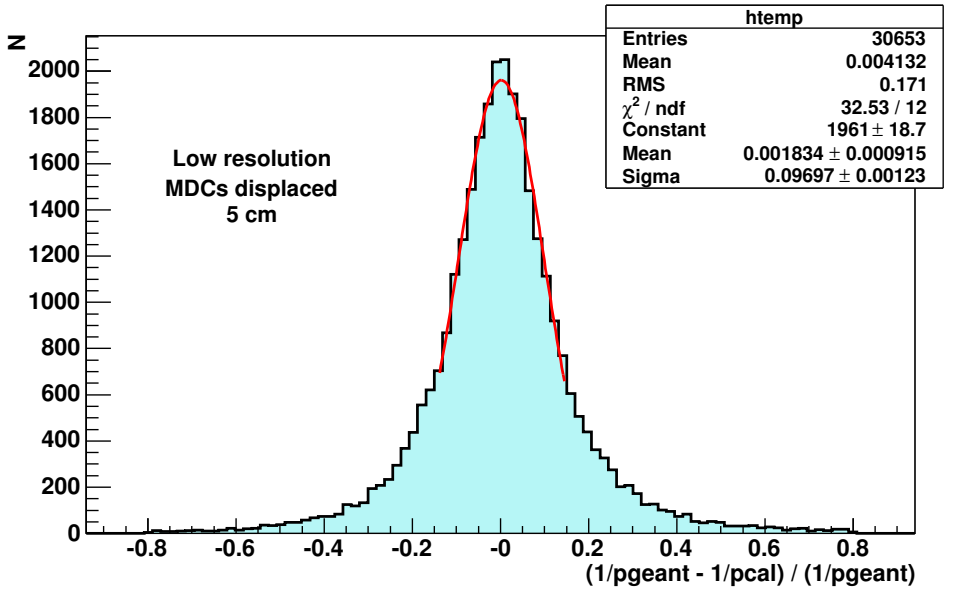


Figure 3.7: *Parameter C as a function of magnetic field in nine different regions of kick surfaces in bins of half a degree in (θ, ϕ)*

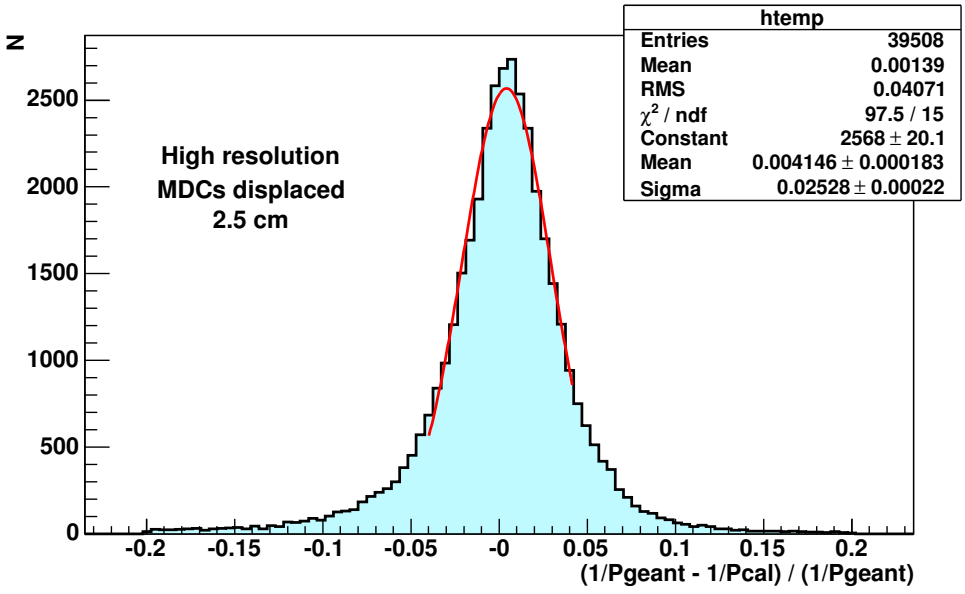


(a)

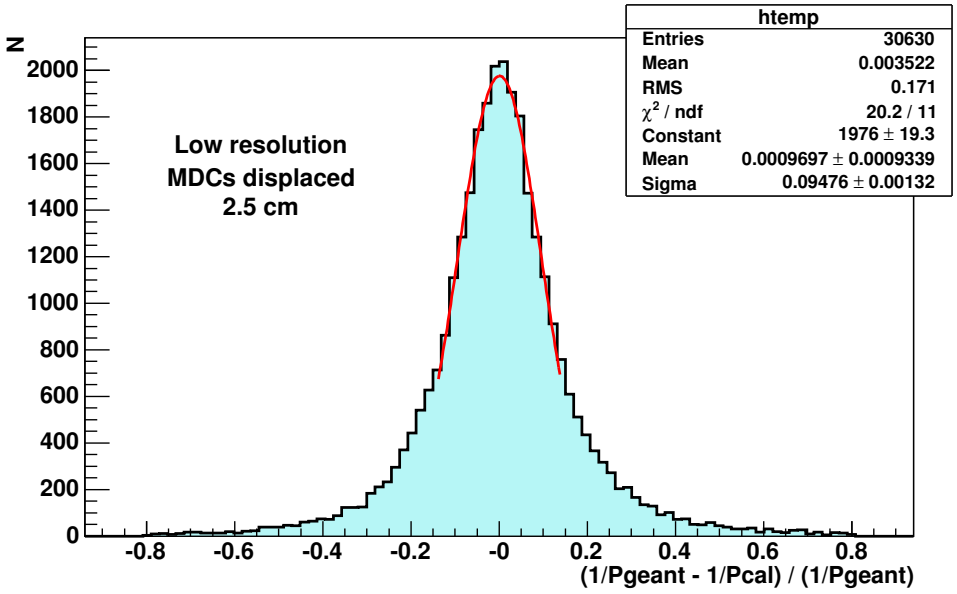


(b)

Figure 3.8: *Momentum reconstruction residuals for high resolution kick plane (a) and low resolution kick plane (b) when outer MDCs are displaced 5 cm downstream from their original position. The difference in resolution between the two kick plane methods is clear*

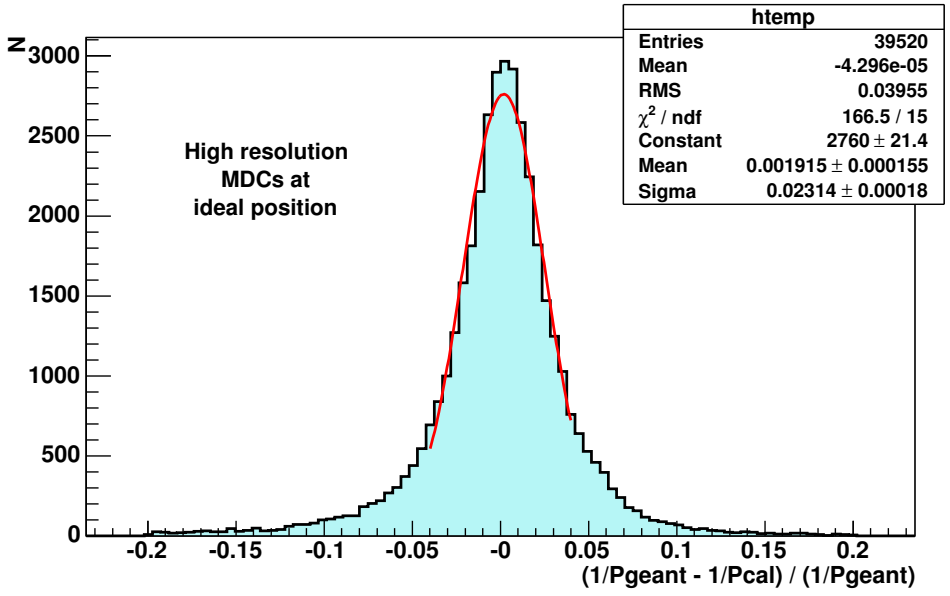


(a)

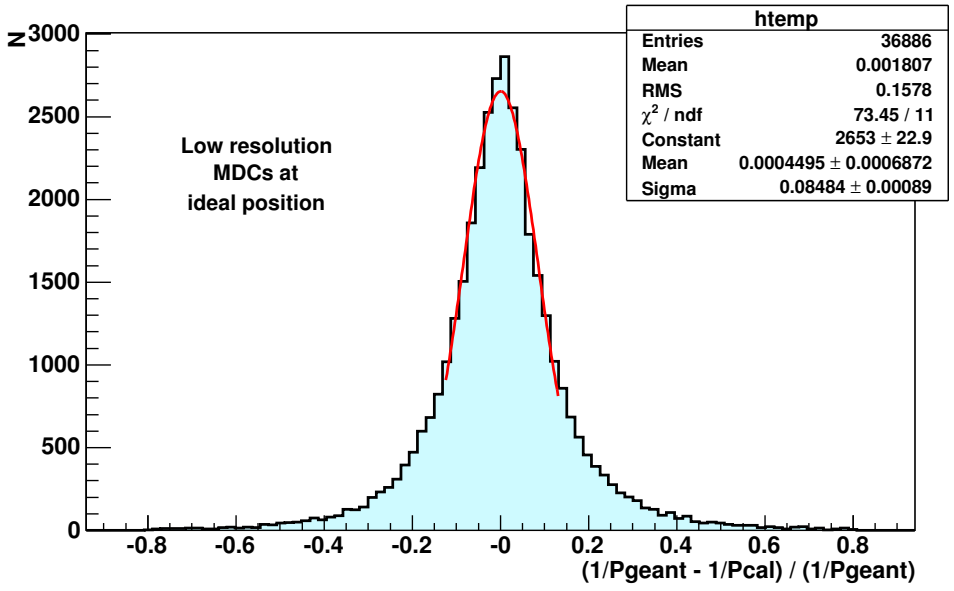


(b)

Figure 3.9: *Same picture as 3.8 but here MDCs are shifted 2.5 cm downstream from the original position*



(a)



(b)

Figure 3.10: *Same picture as 3.8 and 3.9 but MDCs are placed at the ideal position*

the position included in the parameters used along the track reconstruction methods. The HADES alignment has been performed analyzing the hits obtained from the drift chambers [AP02], but the results obtained up to now, due to problems in the MDCs setup, do not report the required accuracy, affecting the momentum and mass reconstruction procedures.

We can see the influence of this effect in the kick plane method in Figs. 3.11, 3.12 and 3.13. A new like-nov02 beamtime simulation was run but in this case with a "simulated" misalignment: the third MDC was shifted 1 cm in the Z_{MDC} direction and the resulting data were processed using parameters for ideal configuration. The results show clearly how big can this effect be, not only in momentum reconstruction where we have now a resolution of $\sigma \sim 10\%$ in the high resolution case (Fig. 3.11), but also in the subsequent mass spectra (Fig. 3.12). Due to the different particles' treatment by magnetic field, this effect becomes greater for positive charged particles, mainly protons and also π^+ (Fig. 3.13).

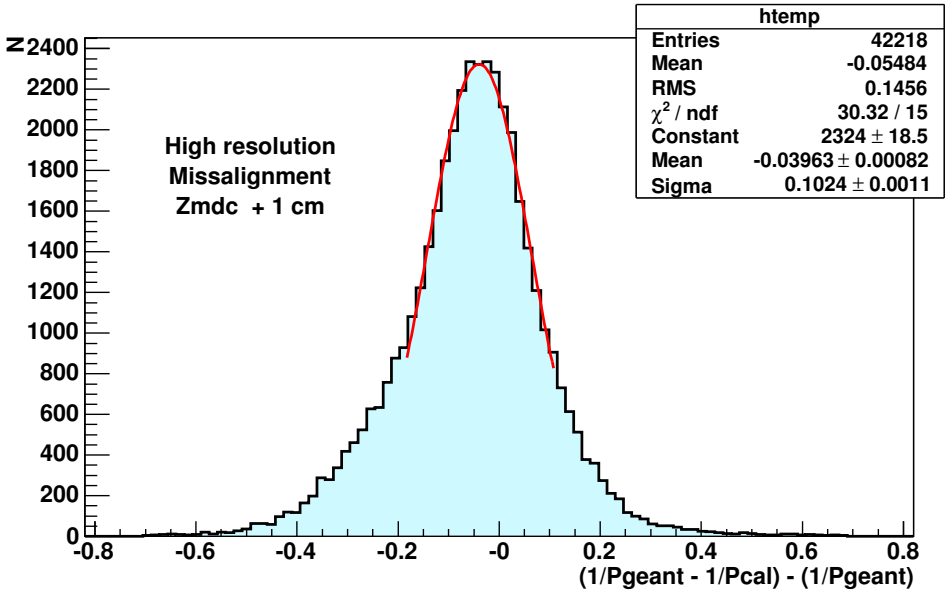
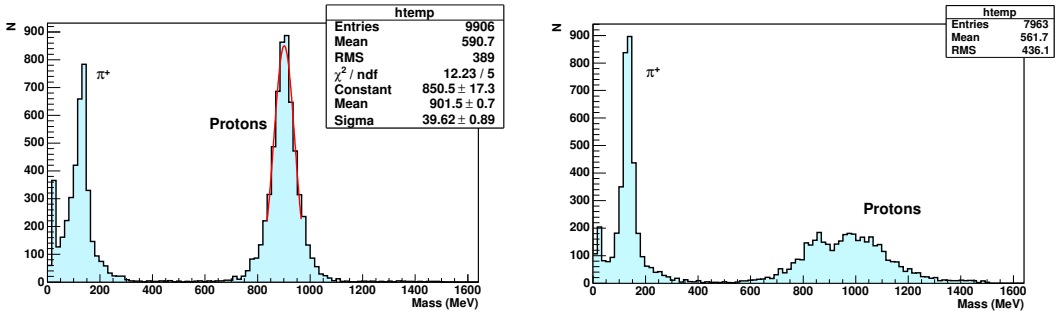
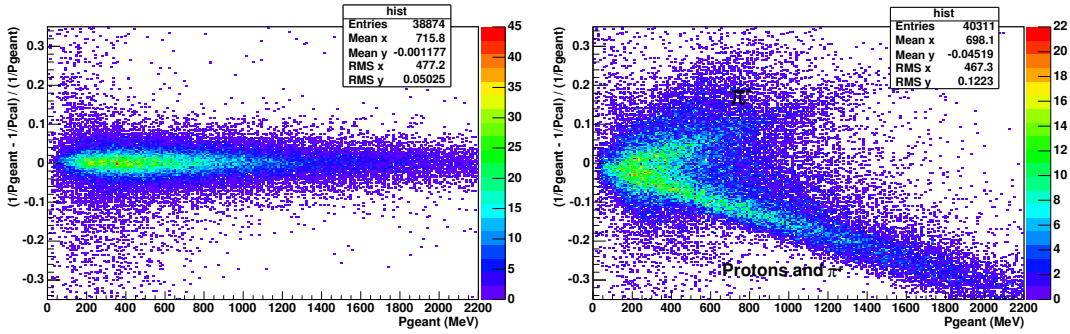


Figure 3.11: *Momentum residuals for high resolution kick plane with the simulated missalignmet. We can see the worst resolution comparing with the ideal one (Fig. 3.10)*



(a) (b)

Figure 3.12: *Mass spectra using momentum given by high resolution kick plane in the correct case (a) and in the case where the missalignment is present (b). Proton mass peak in (a) is placed around 900 MeV due to momentum reconstruction is not corrected by energy loss (we will treat this correction in the next chapter).*



(a) (b)

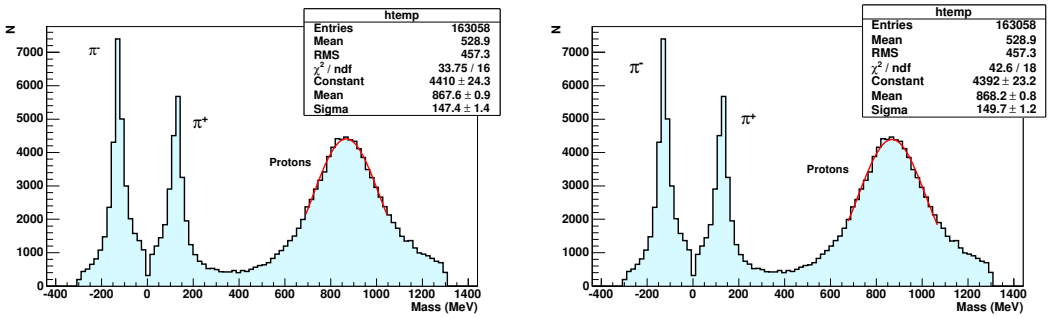
Figure 3.13: *Momentum residuals in function of momentum in the two cases: the correct one (a) and the one with missalignment (b). We can see the differentation between positive charged and negative charged particles in case (b)*

3.4 Dependency of kick plane with target position

One of the problems in HADES is that thin targets have to be used, so that leptons produced in a collision are not reabsorbed in the target. On the other hand, thick targets have the advantage of providing higher reaction rates. The solution is to build segmented targets; that is, several thin targets

in a row. It allows for the leptons to escape while keeping high reaction rates. We have also our segmented targets in different positions from one beamtime to other. It makes necessary to know the influence of target position in the kick plane efficiency.

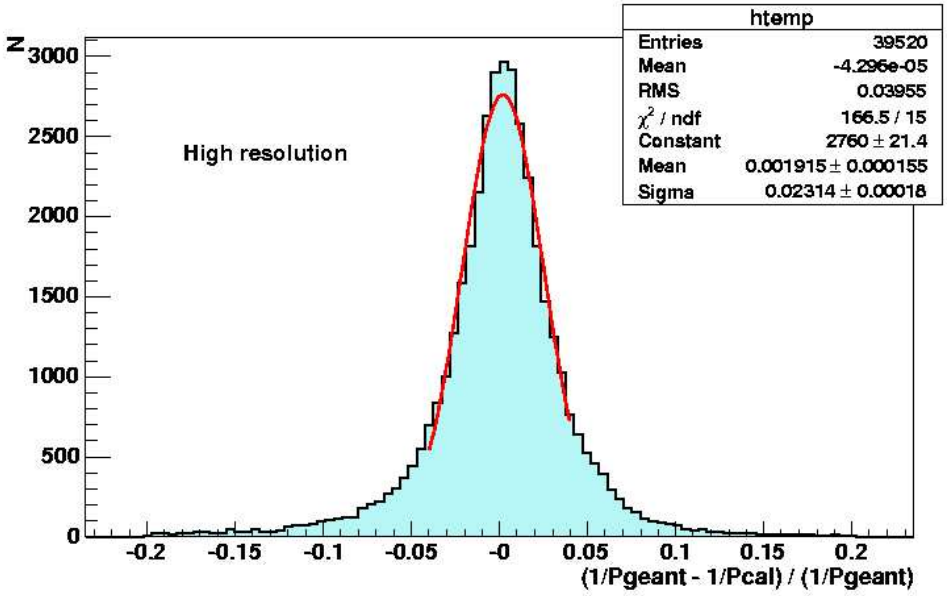
This influence was tested for one DST of Nov02 beamtime data. In this beamtime, the segmented target were two small discs of carbon (diameter $8mm$, lenght $5mm$ and density $2.15g/cm^3$) placed in the beam line and separated $20mm$ to each other. They were generated two sets of kick plane parameters, one per each target position. One new task was implemented in the kick plane code: select the correct set of parameters to use per each event after the information given by the HADES' vertex reconstruction procedure. The method tells us then the momentum of the particle and the piece of target where it cames from. This solution worked correctly but did not produce better results in mass plots (see Fig. 3.14).



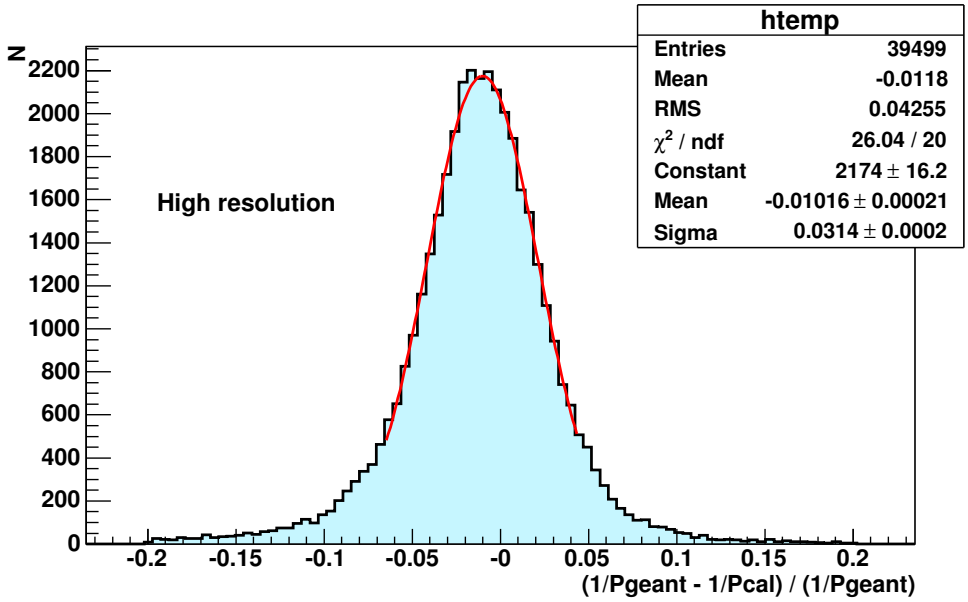
(a) (b)

Figure 3.14: *Mass spectra for Nov02 data procesed with only one set of kick plane parameters (a) and with two sets, one per each piece of segmented target (b). In both cases mass is calculated starting from the low resolution kick plane momentum value*

At the same time, a test on simulated like-Nov02 data was done. A set of events comming from one piece of segmented target was processed using all parameters corresponding to the other piece of target. The results show how the final momentum reconstruction resolution did not hardly change (see Fig. 3.15). Actually, target position affects the momentum reconstruction itself but this effect is hidden by the other uncertainty causes in HADES: energy loss, multiple scattering at RICH, tracking systematics, alignment, etc. So, the final decision was to use only one set of kick plane parameters per each beamtime.



(a)



(b)

Figure 3.15: *Momentum reconstruction residuals for high resolution kick plane in the correct case (a) and in the case where target is displaced around 20 mm respect to the position indicated by parameters*

Chapter 4

Reference Trajectories method and elastic proton-proton analysis

The main idea of the "Reference trajectories" algorithm [SG03] is to obtain a full track model by numerical means. It is designed in order to obtain the maximum resolution, using the information given by the four drift chambers. The method parts from a big and realistic simulation using HGeant package for getting a five dimensional vector which define completly a track. Those parameters are tabulated and momentum of a real track is obtained looking at the table. Such procedure is explained in more detail in following sections.

The method was developed by R. Schicker [Col94] and implemented by M. Sánchez [SG03], and was never tested with real data. The data chosen to do it were the beamtimes Sep03 and Jan04, where protons at 2.2 AGeV hit a liquid hydrogen target. In these collisions, some protons are scattered in elastic reactions. Since elastic reactions have a well known kinematics, we can know a priori several tracks' properties, like momentum. We can compare then the expected values from kinematics and the reconstructed values. An analysis of elastic proton-proton collisions is shown in this chapter.

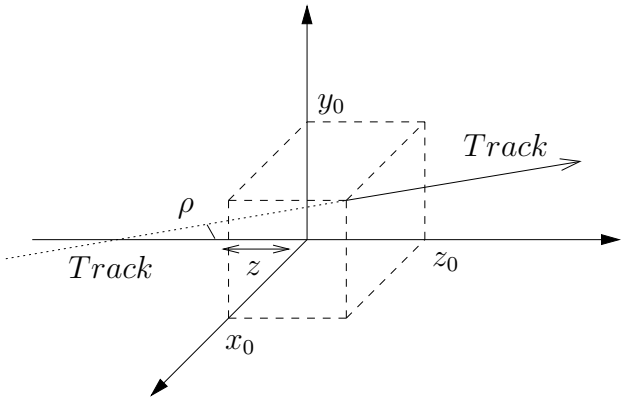


Figure 4.1:

4.1 Reference Trajectories algorithm: fitting procedure

A track is completely defined by 5 parameters, thus it can be represented by a five dimensional vector with components $\mathbf{p} = (1/p, \rho, z, \theta, \phi)$, living in a space P. The first parameter is the inverse of momentum, which can be expected to behave in Gaussian way (while p does not). θ and ϕ are the well known polar and azimuthal angles of the track with respect to the z axis, and ρ and z give information on the track's origin (see Fig. 4.1).

All the information about the tracks that we get from detectors are the (x, y) coordinates on the four MDCs; that is, a 8-dimensional vector x_m living in a space X. Then, there must be a function relating the two vector spaces:

$$\mathbf{F} : P \longrightarrow X \Rightarrow \mathbf{x} = \mathbf{F}(\mathbf{p})$$

Given a known $\mathbf{x}_m$ meassured vector, if the function $\mathbf{F}(\mathbf{p})$ were known, using the LSM method, we can get an estimation of $\mathbf{p}$ just minimizing the functional

$$Q^2 = (\mathbf{F}(\mathbf{p}) - \mathbf{x}_m)^T W (\mathbf{F}(\mathbf{p}) - \mathbf{x}_m) \tag{4.1}$$

where $\mathbf{x}_m$ has two contributions, $\mathbf{x}_m = \mathbf{x} + \mathbf{x}_r$, and $\mathbf{x}_r$ corresponds to the random measurement errors while W is the inverse of its covariance matrix.

In order to compute Eq. 4.1, since $\mathbf{F}$ in unknown, we approximate it by a linear one admiting a Taylor expansion around any point $\mathbf{p}_0$ close to $\mathbf{p}$. Thus,

$$\mathbf{F}(\mathbf{p}) \simeq \mathbf{F}(\mathbf{p}_0) + A \cdot (\mathbf{p} - \mathbf{p}_0) + \mathcal{O}\left((\mathbf{p} - \mathbf{p}_0)^2\right) \tag{4.2}$$

where A is the matrix with elements $A(i,j) = \left.\frac{\partial F_i(\mathbf{p})}{\partial p_j}\right|_{p=p_0}$. With this approximation and after some math work, we arrive at

$$\mathbf{p}_e = \mathbf{p}_0 + (A^T W A)^{-1} A^T W \cdot (\mathbf{x}_m - F(\mathbf{p}_0)) \tag{4.3}$$

being $\mathbf{p}_e$ the solution of Eq. 4.1, that is, $\left.\frac{\partial Q^2}{\partial \mathbf{p}}\right|_{p=p_0} = 0$

The problem now is how to obtain $\mathbf{p}_0$ and compute $\mathbf{F}(\mathbf{p}_0)$. Once we have them, it is easy to devise an iterative algorithm to estimate the track parameters, starting from an initial estimation.

The solution is to know in advance the value of $\mathbf{F}(\mathbf{p})$ for any given $\mathbf{p}$. For this purpose, we tabulate $\mathbf{F}$ formig a five-dimensional grid in the parameter space P. For each entry in this table, or grid vertex, a set of particles is shot and $\mathbf{F}$ is computed using the HGeant simulation package. We are shooting about few hundreds of particles per grid entry and we need two such grids, one for positively charged particles and another one for negatively charged ones. The final result is about 20 Mb of data stored in two grids.

The limits of the grid are defined by the geometrical acceptance of HADES and the physics to do. The granularity has been chosen reaching a compromise between resolution and memory consumption: the more fine grained the binning the better the resolution of the fit routine and the larger the memory consumption. The final grid chosen was:

- The range for $1/p$ is set to $1/p \in \left[\frac{1}{3000}, \frac{1}{100}\right]$ in units of MeV^{-1} with 18 bins inside the range. 100 MeV is the minimum because it is essentially the momentum kick, and 3000 MeV is the maximum momenta that can achieve some particles in collisions like proton-proton at 2 AGeV.

- ρ an z make a virtual cilinder around all possible target positions. Then, ρ is set from -30mm up to 30mm, with 5 bins, and z is set from -60mm up to 60mm, with 15 bins.

- θ range is defined by the angular coverage by the spectrometer; it goes from 18º to 85º but the grid was set from 14º up to 90º in order to increase the acceptance window. Since the main variation in momentum kick happens when moving along θ, the binning has to be relatively fine grained. The grid was set at 20 bins in θ.

- ϕ should cover from 180º to -180º but due to spectrometer's sector symmetry, we only need ϕ covering half a sector. The grid in ϕ is set from 90º up to 122º with 12 bins.

Once we have built the grid, we are able to calculate the momentum for any real or simulated track: we get the position in the four MDCs and calculate the parameters ρ, z, θ and ϕ of for that track. We also get a first estimation of the track's momentum, for example from the KickPlane algorithm. Then, a better estimation of momentum can be iteratively obtained by

$$\mathbf{p}_e^{k+1} = \mathbf{p}_e^k + (A^T W A)^{-1} A^T W \cdot (\mathbf{x}_m - \mathbf{F}(\mathbf{p}_e^k)) \tag{4.4}$$

In order to compute the derivative matrix, we make use of one method wich allows to efficiently compute the derivatives of a tabulated function (like $\mathbf{F}(\mathbf{p})$ is). It is the so called Savitzky-Golay filters. They are independent of the function values, and we can calculate all of them in advance and reuse them each time a partial derivative needs to be computed.

4.2 Proton-Proton collisions

The investigation of dielectron production in elementary reactions attracts a lot of attention nowadays. Since there is no consistent theoretical description of the various experimental findings until now, it is the goal of the HADES collaboration to establish clear experimental signals for modified in-medium spectral functions of mesons and in particular of the light vector mesons ρ and ω in pp and dp collisions at beam energy of E=1.25 GeV and ω production in pp reactions at E=3.5 GeV [Col04].

But the starting point of the HADES physics program were pp collisions at beam energy of 2.2 GeV [Spa05]. There are three important goals in these reactions: study of elastic scattering, which is a good tool for checking the reconstruction program, study of η production and study of π^0 production, which are good tools for HADES acceptance corrections. The work presented here is mainly focused in the analysis of elastic scattering of two protons.

Elastic Proton-Proton collisions are very well known reactions. We are going to present here some needed kinematics for the further analysis presented in next sections. We start from a target particle, which is a proton, hitting by a projectile, which is another proton, with a incident energy of 2.2 GeV. The total cross section of such a process is $47mb$ [Col04], where $43mb$ is the cross section for 2 charged hadrons production and $4mb$ for 4 hadrons production. The total reaction types are shown in table 4.1.

Table 4.1 shows how half of reactions in these collisions are elastic scattering of protons. The most important feature for this study is the fact that we can calculate the expected final momentum of that scattered proton just

Reaction channel	**Cross section** $\sigma[mb]$
$p+p$	18.0
$p+p+\pi^0$	3.5
$p+n+\pi^+$	15.0
$p+p+\pi^0+\pi^0$	0.6
$p+p+\pi^+ +\pi^-$	2.5
$p+n+\pi^+ +\pi^0$	3.5
$p+p+\pi^0+\pi^0+\pi^0$	0.13
$p+p+\pi^+ +\pi^0+\pi^-$	0.4
$p+n+\pi^+ +\pi^+ +\pi^-$	0.4
$p+p+\eta$	0.12
$p+p+\omega$	0.06
$p+p+\rho$	0.06

Table 4.1: *Cross sections for* pp *reaction channels at beam energy of 2.2 GeV [Col04], [Prz03]*

measuring its polar angle after the collision. Let's see how [Kop95], [Mar84], [Fre88], [Woo03], [McC99], [GPS02].

Lets supose a proton of mass m and momentum P_{inc} moving along the z axis, and suffering a head-on collision in the plane yz with another proton at rest in the Laboratory system (see Fig. 4.2 (a)). Lets also take always the speed of light $c = 1$. Then, the initial 4-momentum is:

$$p^{\mu} = ([m\gamma + m], 0, 0, p_{inc} = m\gamma \mathbf{v}_{inc}) \tag{4.5}$$

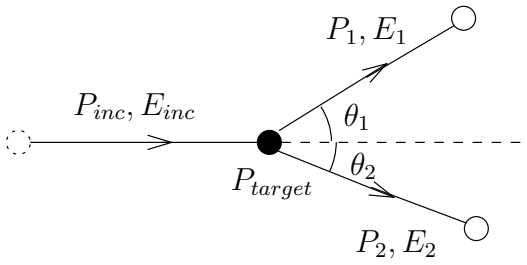
But this reaction can be also seen from the so called "center-of-mass" frame, where the total momentum is zero (actually, this frame should be called "center-of-momentum" frame). Fig 4.2 illustrates the relations of the incident and scattered spatial momentum vectors in both systems (lets call them LAB and CM). Both frames are related by the usual Lorentz transformations. Primes on the vectors denote CM values while unprimed vectors are in LAB system. Then, the total momentum in CM system is:

$$([m\gamma_1^{'} + m\gamma_2^{'}], 0, 0, 0) \tag{4.6}$$

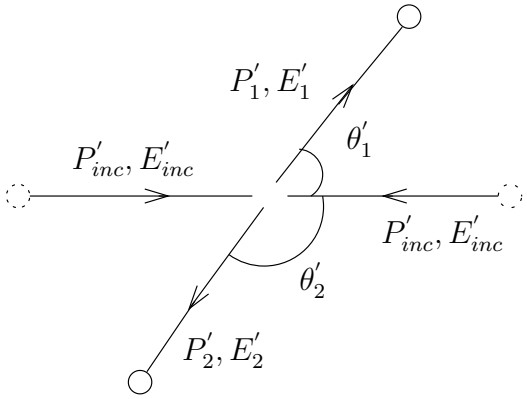
since by definition $\mathbf{p}_1^{'} + \mathbf{p}_2^{'} = 0$.

The Lorentz factor of the two particle system, γ, in LAB frame is, then,

$$\gamma = \frac{E_{inc} + m}{M}$$



(a) Laboratory Frame



(b) Center-of-Mass Frame

Figure 4.2: *Schematic view of an elastic collision between two protons in both coordinate system*

where M is the invariant mass of the two particle system. So,

$$\gamma = \frac{E_{inc} + m}{\sqrt{2mE_{inc} + 2m^2}} = \frac{T_{inc} + 2m}{\sqrt{2mT_{inc} + 4m^2}} \tag{4.7}$$

T is the kinetic energy of the particle.

Applying the Lorentz transformations, the components $p_{inc}^{\mu'}$ in CM are given by:

$$\begin{aligned} p_{inc}^{'} &= p_{inc}^{3'} = \gamma(p_{inc} - \beta E_{inc}) \\ E_{inc}^{'} &= p_{inc}^{0'} = \gamma(E_{inc} - \beta p_{inc}) \end{aligned} \tag{4.8}$$

After the collision, $\mathbf{p}_1^{'}$ and $\mathbf{p}_2^{'}$ are no longer along the z axis, but since the

collision is elastic (the masses remain unchanged), and due to the momentum conservation law, the components of, for example, $\mathbf{p}_1^{'}$ in CM are:

$$p_1^{2^{'}} = p_1^{'} \sin\theta_1^{'}, \quad p_1^{3^{'}} = p_1^{'} \cos\theta_1^{'}, \quad p_1^{0^{'}} = p_{inc}^{3^{'}} = E_{inc}^{'} \tag{4.9}$$

The transformation back to the LAB system is the same Lorentz transformation but with relative velocity $-\beta$. Hence, the components of $\mathbf{p}_1$ are:

$$\begin{aligned} p_1^2 &= p_1^{2^{'}} = p_{inc}^{'} \sin\theta_1^{'} \\ p_1^3 &= \gamma(p_1^{2^{'}} - \beta p_1^{0^{'}}) = \gamma(p_{inc}^{'} \cos\theta_1^{'} + \beta E_{inc}^{'}) \\ p_1^0 &= \gamma(p_1^{0^{'}} - \beta p_1^{3^{'}}) = \gamma(E_{inc}^{'} + \beta p_{inc}^{'} \cos\theta_1^{'}) \end{aligned} \tag{4.10}$$

Then, replacing $E_{inc}^{'}$ and $p_{inc}^{'}$ from Eqs. 4.8, we obtain, after a little simplification, an expression for the energy of the scattered proton in terms of the incident properties:

$$E_1 = E_{inc} - \gamma^2 \beta (p_{inc} - \beta E_{inc})(1 - \cos\theta_1^{'}) \tag{4.11}$$

γ is given by Eq. 4.7 and β can be written as:

$$\beta = \sqrt{1 - \frac{1}{\gamma^2}} = \frac{\mathbf{p}_{inc}}{T_{inc} + 2m}$$

Then,

$$\gamma^2 \beta (p_{inc} - \beta E_{inc}) = \frac{m(T_{inc}^2 + 2mT_{inc})}{2mT_{inc} + 4m^2} \tag{4.12}$$

Now, after some algebraic manipulation, Eq. 4.11 can be rewritten as:

$$\frac{T_1}{T_{inc}} = 1 - \frac{1 - T_{inc}/2m}{2 + T_{inc}/m}(1 - \cos\theta_1^{'}) \tag{4.13}$$

The problem now is that what we can measure is the angle of the scattered proton in the LAB system, θ_1. So, we need a relation between θ_1 and $\theta_1^{'}$. But it easy to realize that (upper index means component, subindex means particle):

$$\tan\theta_1 = \frac{p_1^2}{p_1^3} = \frac{\sin\theta_1^{'}}{\gamma_{CM}(1 + \cos\theta_1^{'})} \tag{4.14}$$

where

$$\gamma_{CM} = \sqrt{\frac{\gamma_{inc}+1}{2}}$$

Finally, after some math work with Eqs. 4.13 and 4.14, we arrive at:

$$\frac{T_1}{T_{inc}} = \frac{2\cos^2\theta_1}{(\gamma+1)-(\gamma-1)\cos^2\theta_1} \tag{4.15}$$

and

$$p_1 = \sqrt{T_1^2 + 2mT_1} \tag{4.16}$$

which allows to calculate the kinetic energy and the momentum of the scattered particle from the incident energy (T_{inc} and γ) and the measured scattering angle θ_1.

Following the same procedure for the other scattered proton, we can arrive at the following useful expression:

$$\tan\theta_1 \cdot \tan\theta_2 = \frac{2}{1+\gamma_{inc}} = \frac{1}{\gamma_{CM}^2} \tag{4.17}$$

and the opening angle between the two scattered protons is:

$$\phi = \theta_1 + \theta_2 = 2\arctan\left(\frac{2}{1+\gamma_{inc}}\right)^{1/2} \tag{4.18}$$

4.3 Experimental setup

The SEP03 and JAN04 data sets correspond to $p+p$ collisions at 2.2 AGeV [HAD]. The experimental setup available at the moment of the data taking consisted of the RICH detector for all sectors, four sectors equiped with the four MDCs[1], two sectors equiped with only three MDCs (the outer one was disabled), as well as TOF, TOFino and SHOWER detectors also for all sectors. The magnetic field current was set up to 3195 A, which means 92.32% of full magnetic field.

The incident beam were protons acelerated up to 2.2 GeV of kinetic energy, which means momentum of 3 GeV/c per proton. The target was a LH_2 target[2] with $2 \cdot 10^{23} atoms/cm^2$ within a cilinder aluminium pipe of $5mm$ diameter and $50mm$ length, placed, in z direction, from $-37.5mm$ to $+12.5mm$ respect to the HADES main coordinate system.

[1] This means four sectors with the full resolution momentum reconstruction setup; that is, we are only allowed to use Reference Trajectories in that four sectors

[2] Liquid hydrogen target

In order to check the momentum reconstruction algorithms in these data, two files, one from the Sep03 DST and another one from the Jan04 DST, were taken and reprocessed with the Reference Trajectories algorithm (for further comparision, they were also processed with the low-resolution KickPlane). The file processed from Sep03, be03276011901_or_dst.root, had about 700000 events, and the one from Jan04, be04046184749_dstgen1.root, had about 300000 events. The alignment parameters were slightly different in both cases, as well as the START detector presence only in the Jan04 setup. This causes a different quality of the data taken in Sep03 and Jan04.

We have also to keep in mind the different releases of the reconstruction software for both runs, mainly with the so called "tracking" software which remained with systematic errors in the polar angle reconstruction in the Jan04 DST. In the Sep03 DST there was not any META alignment, so it was set to the ideal geometry in order to use the KickPlane method.

4.4 Selecting elastic events

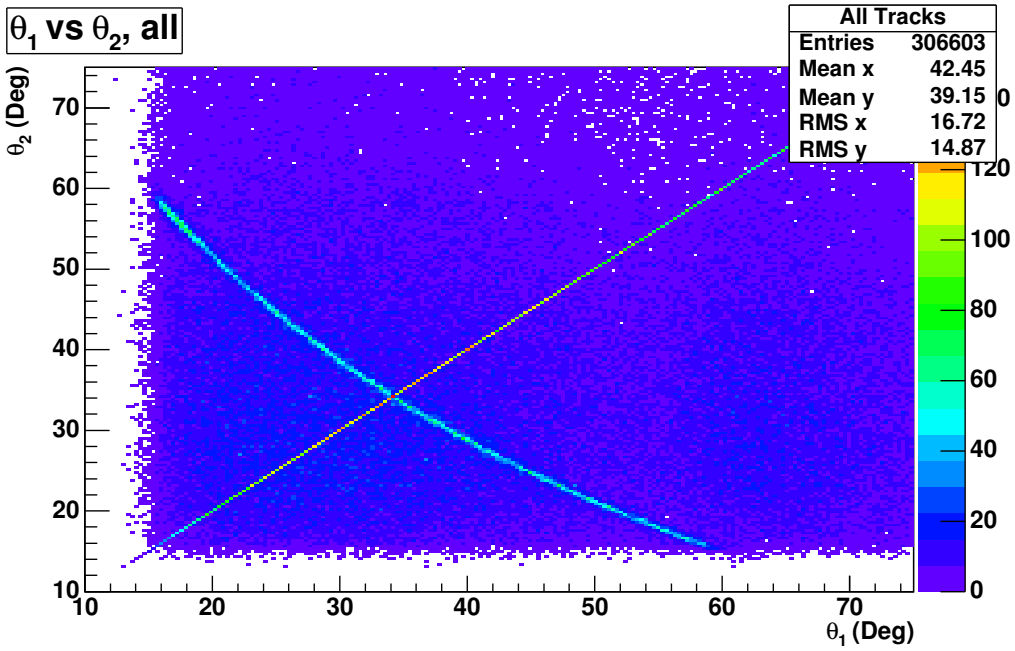
As we have seen in section 4.2, in our proton-proton collisions, several kind of reactions can be produced. It makes necessary to select those reactions which we are interested on. This task is performed applying some cuts in the events read with the help of a ROOT macro. The selection cuts applied are:

1. Events with only two reconstructed tracks
2. Both tracks should be positive
3. Both tracks hitting in opposite sectors
4. Coplanarity: cut in azimuthal angles. We have chosen $Abs(\phi_{proton1} - \phi_{proton2}) < 1.5^{o}$
5. Cut in polar angles: $\tan\theta_{proton1} \cdot \tan\theta_{proton2} = \frac{1}{\gamma_{CM}^2}$ between 0.4525 and 0.4725 (elastic kinematics condition)

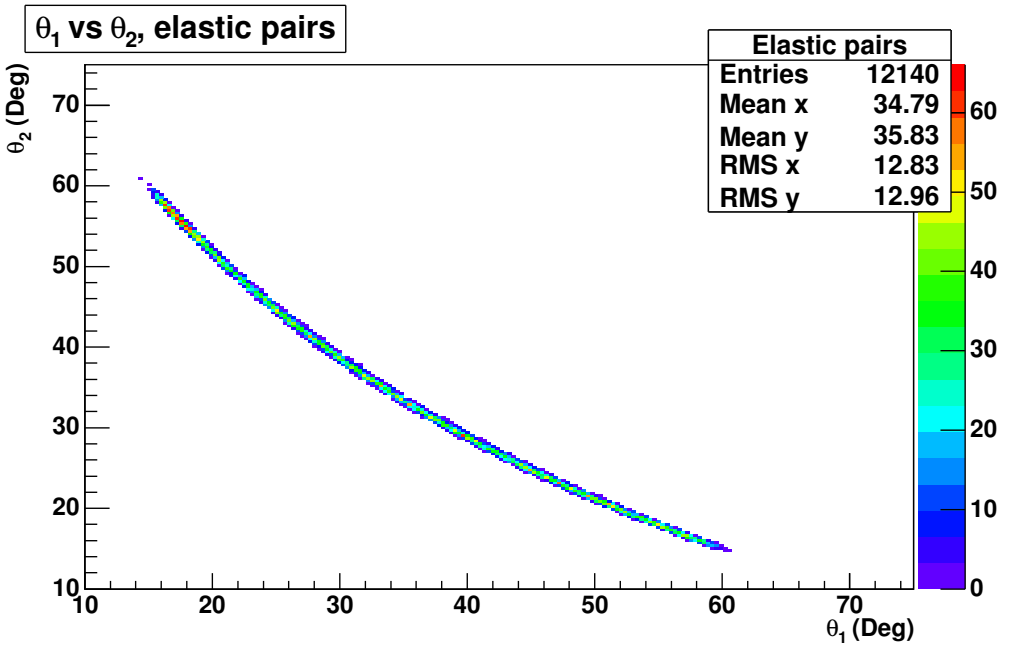
After the cuts, about 5% of well reconstructed events[3] are selected as very good elastic events.

In Fig. 4.3 we can see a plot of $\theta_{proton1}$ versus $\theta_{proton2}$ before and after elastic events selection. In Fig. 4.4 we can see the openning angle $\theta_{proton1} + \theta_{proton2}$ between the two elastic protons.

[3]Well reconstructed events are those events which were well reconstructed by the momentum reconstruction algorithm (since we are not taking into account, for example, events going through sectors with only three chambers). They represent about 33% of total events in the files.



(a) All events



(b) Elastic events

Figure 4.3: *Both polar angles, $\theta_{proton1}$ and $\theta_{proton2}$ before (a) and after (b) aplying elastic selection. From (b), it is clear that we are only selecting elastic events. In (a) we have all possible events: elastic, non-elastic, and double hits*

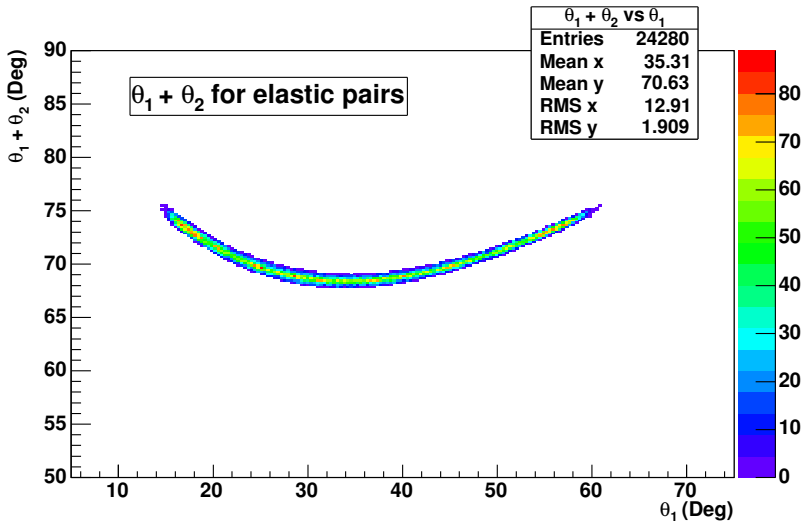


Figure 4.4: *Openning angle $\theta_1 + \theta_2$ of the two elastic protons as a function of θ_1*

4.5 Energy loss correction

Charged particles lose energy when traversing matter primarily by ionization and excitation[4]. The mean rate of energy loss is given by the well known Bethe-Bloch equation, which basically is a function of the charge z and β of the incident particle [ea04b].

When a particle leaves the interation zone it has a given momentum p_{target}. This particle then suffers energy loss mainly in the target itself (we have $250mm^3$ of LH_2 target), in the aluminium pipe and in the RICH's carbon mirror. So when it reaches the MDCs and magnetic field region, its momentum is $p_{mdc} < p_{target}$ (energy loss inside the MDCs is negligible). Since the track deflection in the field is obviously proportional to p_{mdc}, what we measure is p_{mdc} and we need to correct it for the energy loss to reobtain p_{target}.

Typical energy loss curves for protons on different materials are shown in Fig. 4.5. We can see how, in our momentum regime (see section 4.6), protons suffer energy loss in the range of a few MeV, depending on their momentum, which means a variation in momentum between 2% and 0.1%. This variation can be not negligible.

What we are really interested in, is in such variation in momentum. Using the HGeant simulation package, an estimation of this effect can be achieved.

[4]See Appendix B for details

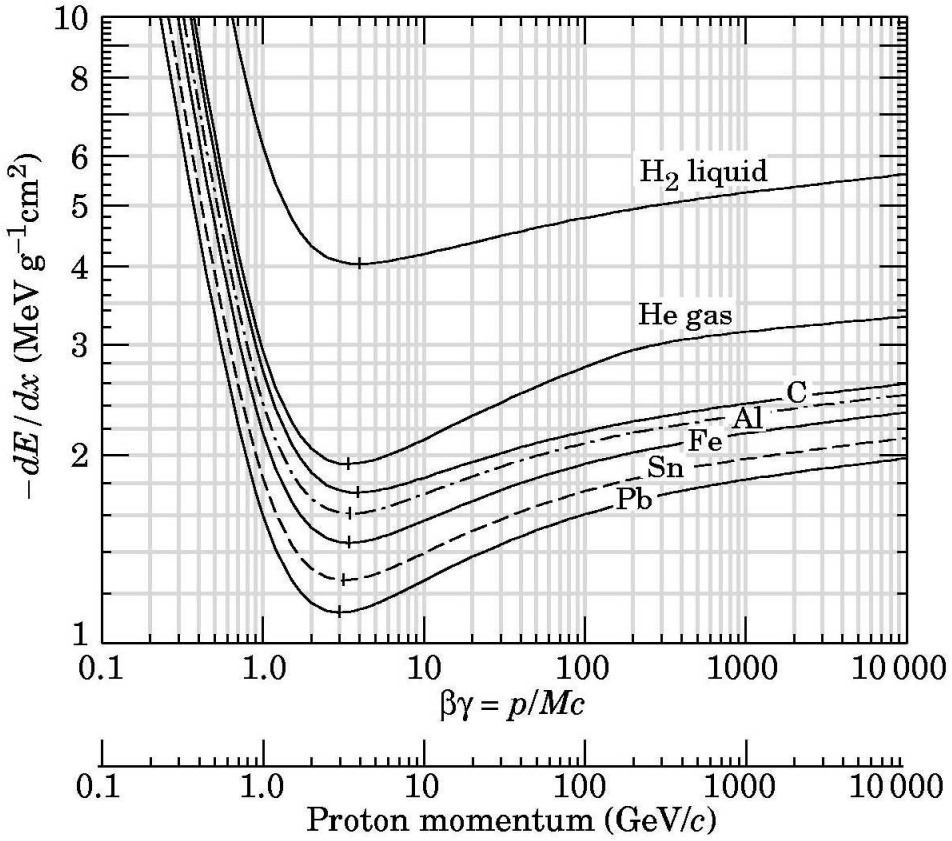


Figure 4.5: *Energy loss for protons in various materials*

Starting from a realistic simulation (in fact, from the Sep03 Simulation DST), we can get easily from HGeant the momentum of any particle in the interaction point (p_{target}) and in the first MDC (p_{mdc}). Then, we can plot the difference between those two quantities as a function of p_{mdc} and fit such dependency to a phenomenological model. After that, we can make use of this fit in order to correct our momentum measurements.

The model used [SG03], derivated by direct inspection of the data, is:

$$p_{target} - p_{mdc} = p_0 + e^{p_1 + p_2 \cdot p_{mdc}}$$

where p_0, p_1 and p_2 are the model parameters. The fit and the values of this parameters are shown in Fig. 4.6

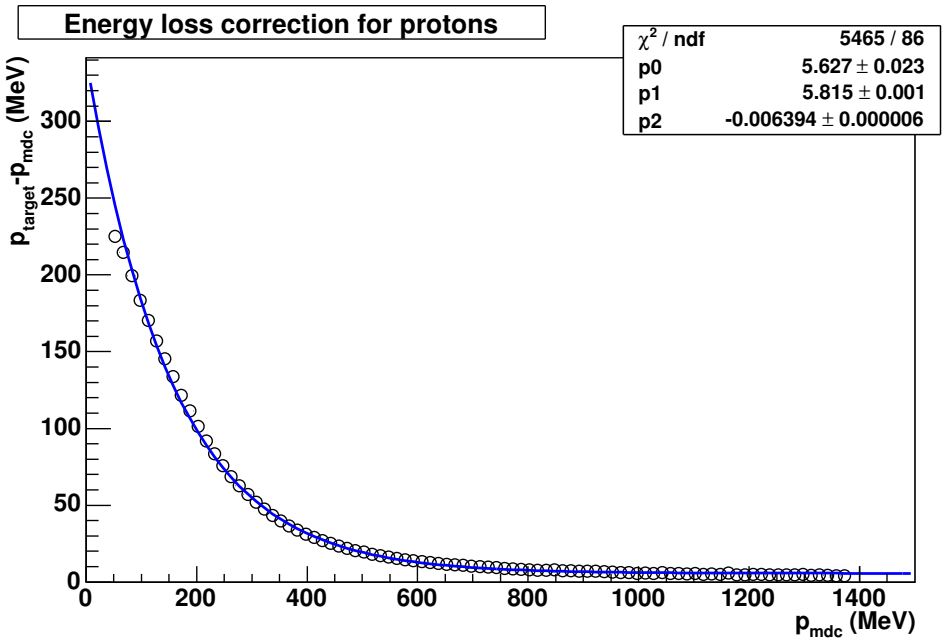


Figure 4.6: *Variation in the tracks's momentum due to energy loss meassured between the target poin (p_{target}) and the first MDC (p_{mdc}). Error bars are purely statistical and they are of the same order as the markers in the picture*

4.6 Results on real data

Due to several reasons, Sep03 data set has a better quality than Jan04 data set. For that reason, the analysis presented here is focused almost completely in the Sep03 *pp*-collisions data set. The pictures shown in this pages correspond to real data analyzed with the Reference Trajectories algorithm presented in section 4.1.

The first check we can do with these data are the tracking and alignment procedures [Fn05], [AP02]. As we have seen in section 4.2 we know in advance, for example, the quantity $\tan\theta_1 \cdot \tan\theta_2$ which is equal to $\frac{2}{1+\gamma_{incident}}$ or $\frac{1}{\gamma_{CM}^2}$. By kinematics, this value should correspond[5] to 0.461. Fig. 4.7 shows this value calculated for our real data. We can see the very good agreement between the calculated and the theoretical value.

Using this data, the best way to get an idea about how good our momentum reconstruction method is, is to compare the calculated momentum with the kinematically expected value from Eqs. 4.15 and 4.16. Figs. 4.8

[5] $T_{incident} = 2.2GeV$ means $\gamma_{incident} = 3.34$

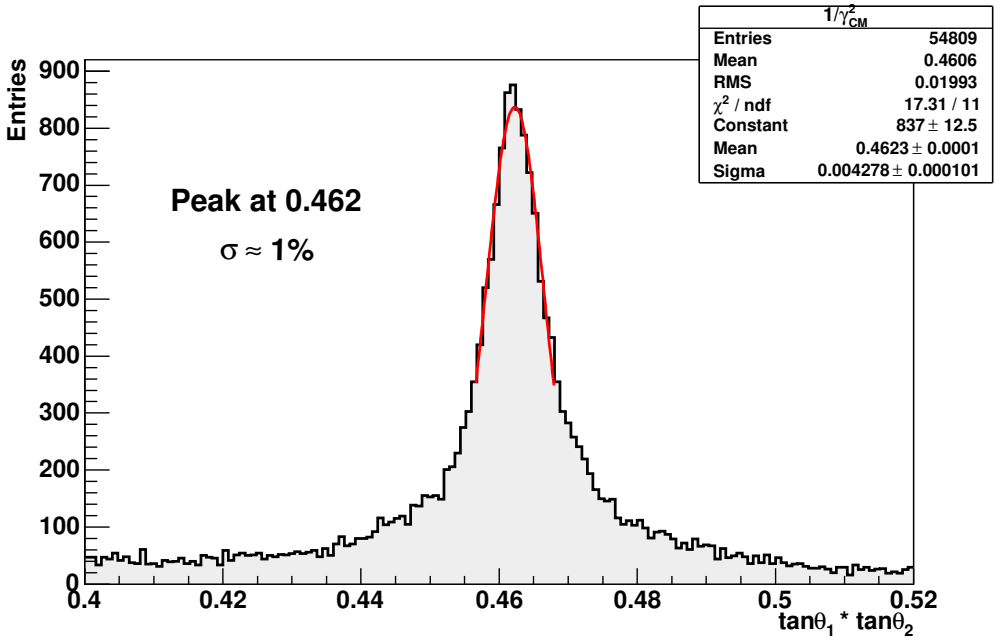


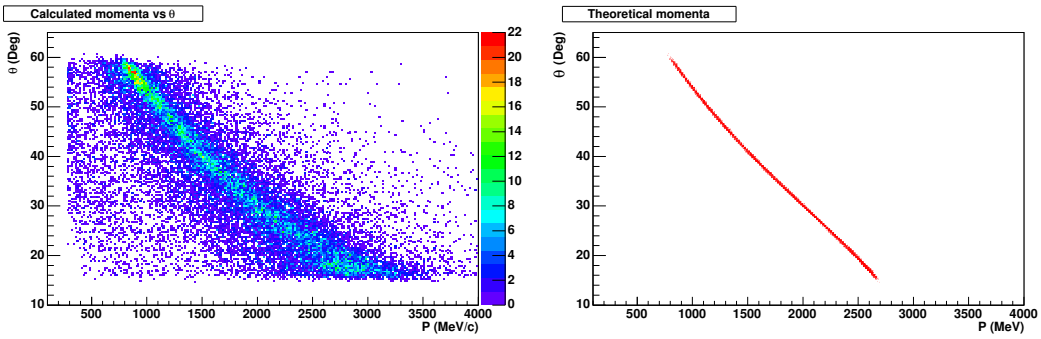
Figure 4.7: *Calculated value of* $\frac{2}{1+\gamma_{incident}}$ *as a function of the two outgoing protons' polar angle. The gaussian fit yields a value of 0.462, with* $\sigma = 0.004$*, while the kinematical one is 0.461*

and 4.9 show this comparison. The momentum range for our elastic protons goes from 700 MeV up to 3 GeV. Lower momenta correspond to higher polar angles, and higher momenta, to lower polar angles, where the resolution of the method is worse than for low momenta. The most important factor which contributes to this worst resolution is the fact that our spectrometer is optimizated for momenta from 500 to 1000 MeV, region where we can found the vector mesons decay. Then, for high momenta, we lose quality of data.

Another effect we can observe in this pictures is the polar angle acceptance in elastic collisions. We know from kinematics that the openning angle between the two scattered protons[6] is around to 70º. Then, if we put a cut in opposite sectors, since the polar acceptance of the spectrometer covers from 18º to 85º, the maximum polar angle of the measured elastic proton should be around 60º, and the minimum around 15º.

Making use of both momenta, the calculated and the theoretical ones, we can estimate a resolution of the full momentum reconstruction procedure. Fig. 4.10 shows such momentum residuals. The resolution achieved

[6]See Eq. 4.18 and Fig. 4.4



(a) (b)

Figure 4.8: *Calculated (a) and theoretical (b) momentum of elastic protons as a function of polar angle θ*

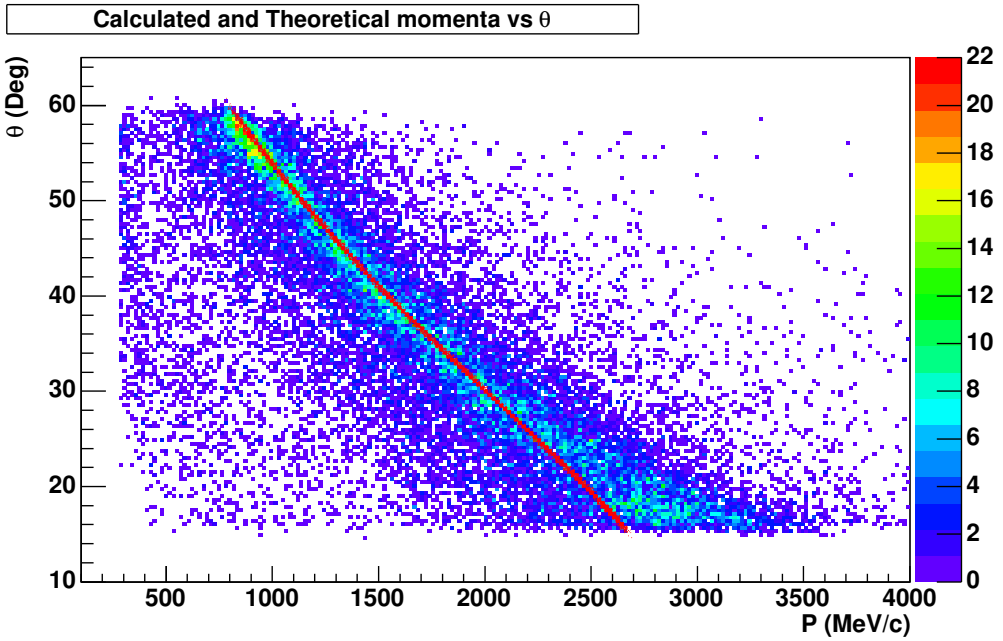


Figure 4.9: *Superposition of pictures 4.8 (a) and (b). The agreement between calculated and theoretical results is apparent, except at very high momenta*

is around 10%. This number is far from the 1% resolution in momentum proposed by the Hades Collaboration, but we must realize that, on the one side, Hades spectrometer is not optimized for this kind of data and the qual-

ity of the data is not the best; on the other hand, this 10% is not only the resolution of the momentum reconstruction method itself, but also of the full event reconstruction. We have to convolute the uncertainty on detectors' calibration, on detectors' hit finders, on segments and tracks reconstruction, on alignment, on beam energy, etc.

Fig. 4.11 shows momentum residuals as a function of polar angle θ (a) and reconstructed momentum (b). Despite of possible systematics errors in tracks' reconstruction, which produces deviations at very low and very high θ, the most important effect in the momentum reconstruction is the value of the momentum itself, as it was explained before.

By the way, since we can know a priori how our systematic errors are (see for instance Fig. 4.11), we could correct those errors during the momentum reconstruction procedure, achieving so a better final resolution than 10%. This task is proposed for further analysis and requires some extra work.

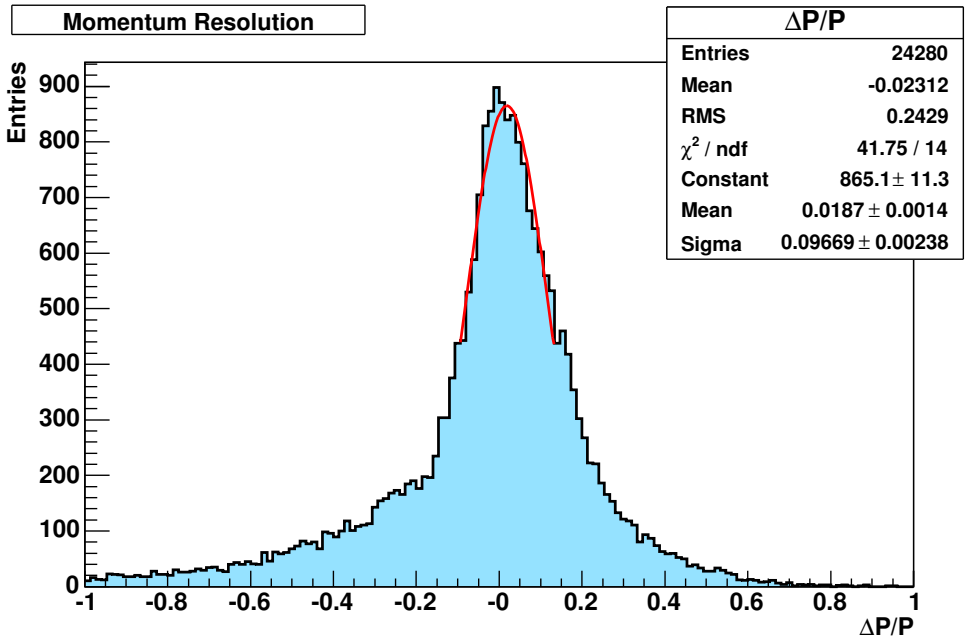
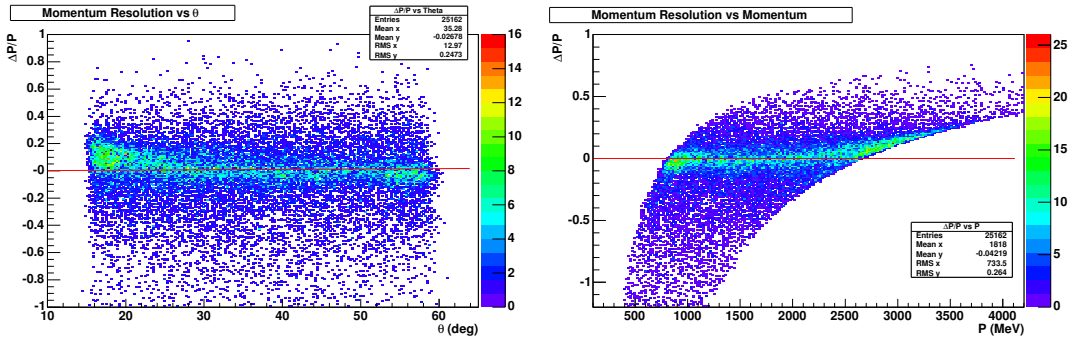


Figure 4.10: *Momentum residuals for the Sep03 pp data.* $\Delta P/P$ *means* $\frac{1/P_{theo}-1/P_{cal}}{1/P_{theo}}$. *The fit to a gaussian yields a resolution around 10%.*

The last pictures of this analysis show the invariant mass of the elastic *pp* pair, Fig. 4.12 (a), and also the missing mass, Fig. 4.12 (b). For these calculations we need to construct the quadrimomentum vectors of both protons in the selected elastic pair, $\mathbf{p_i} = (\vec{p}_i, E_i)$, $i = (1, 2)$. Invariant mass is just the



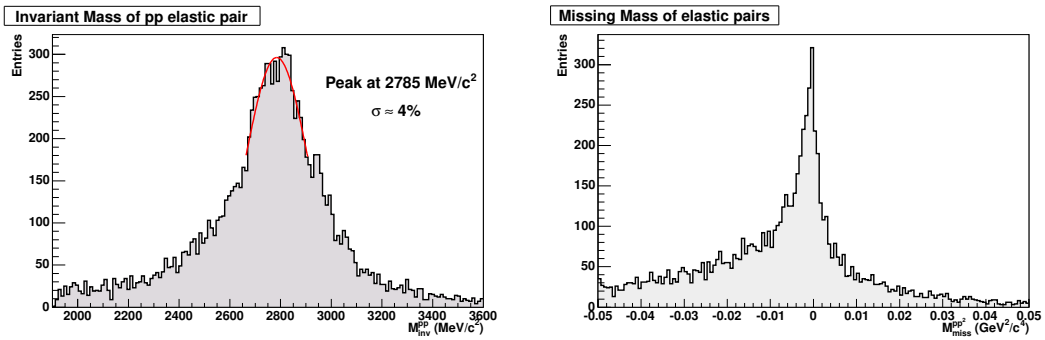
(a) (b)

Figure 4.11: *Momentum residuals as a function of θ (a) and p (b). In (b) we can see the momentum range and acceptance of this elastic pp collisions*

module of the quadrivector sum of both cuadrivectors. From kinematics, this value should be $2767 MeV/c^2$, while the calculated one is around $2785 MeV/c^2$ with $\sigma \approx 4\%$. For the missing mass, we substract the quadrivectors initial minus final. Then,

$$M^{pp}_{missing} = \sqrt{(E_{ini} - E_{p_1} - E_{p_2})^2 - (p_{ini} - p_{p_1} - p_{p_2})^2}$$

For the elastic pair only, the missing mass should be compatible with zero.



(a) (b)

Figure 4.12: *Invariant (a) and Missing (b) mass for the elastic pp pairs of the Sep03 data*

Figs. 4.13 and 4.14 show the momentum vs. polar angle θ (same picture as 4.9) for the Low-resolution Kick Plane method with Sep03 data and for the Reference Trajectories method with the Jan04 data respectively. In the first case, the worst resolution of this method is evident, besides the no-good-alignment of META detectors. In the second case, we can easy realize the worst quality of the Jan04 data in comparisson with the Sep03 data, as well as low statistics in the file procesed.

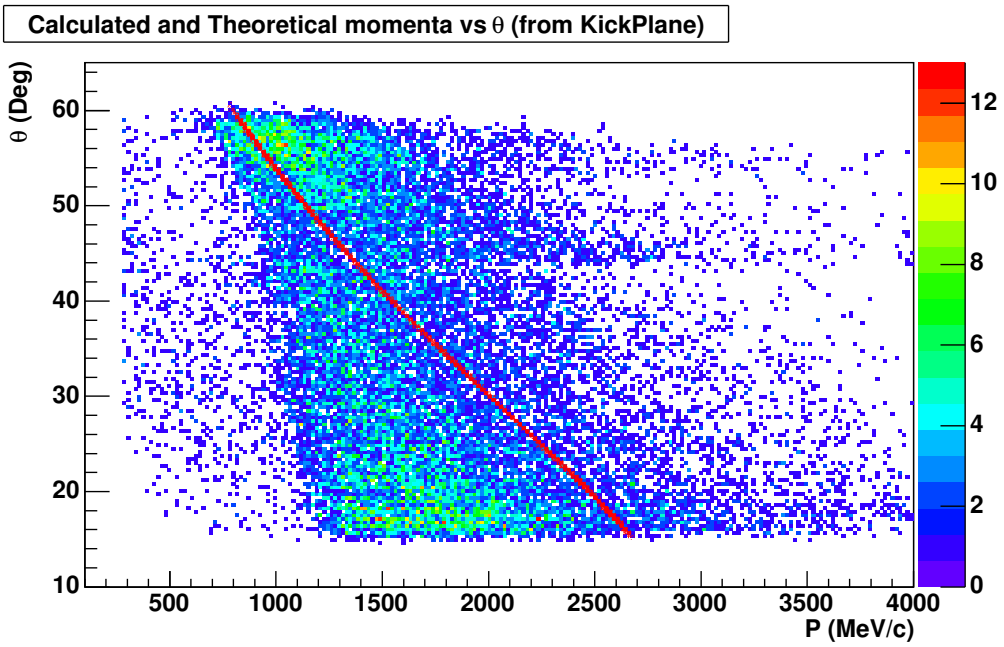


Figure 4.13: *Calculated and theoretical momenta of Sep03 data with low-resolution Kick Plane*

4.7 Comparison with simulated data

To compare our results with results on simulated data is always a good way to check our work and to get explanations of some effects. In this case, the results were briefly compared with two complementary simulations that we will see next.

The first important simulation of *pp* collisions in the HADES spectrometer was performed by W.Przygoda [Prz03] using one event generator called Pluto. Pluto++ [HAD] is a collection of C++ classes, added up to the framework of a simulation package for hadronic-physics reactions. It is launched

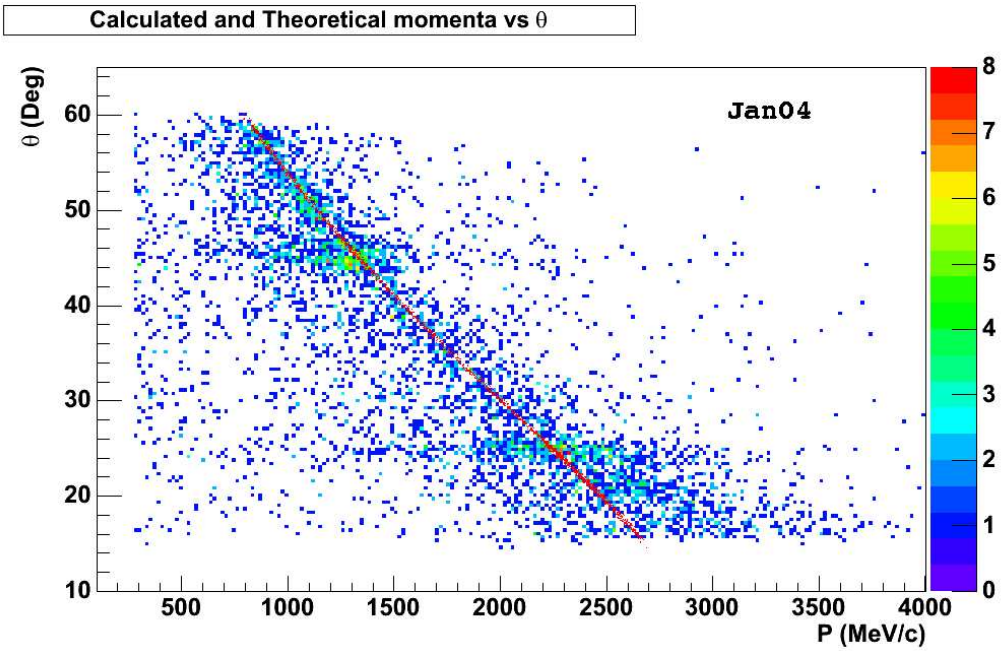


Figure 4.14: *Calculated and theoretical momenta of Jan04 data with the Reference Trajectories method*

interactively from within the ROOT environment, and makes use of ROOT and CLHEP-library resources. The output may be analyzed on line, or further processed with GEANT. In this case was analized on line with a tool based on reference trajectories by R.Schicker.

In Fig. 4.15 we can see the polar angles of the two elastic scattered protons after the elastic events selection. Only about 10% of events were selected. The agreement with Fig. 4.3 (b) is good but the wider distribution of elastic events in the real data gives us an idea about the event reconstruction procedure in HADES. It must be improved. Fig. 4.16 shows a comparison between the momentum of the two elastic protons in the case of Pluto and in the real data and Fig. 4.17 shows the protons' momentum as a function of the polar angle θ. Again, in this case, the agreement with Fig. 4.9 is very good.

The other comparison was made with HGeant simulation package. One file of the so called SimDST Sep03 was taken and reprocessed using the Reference Trajectories algorithm. That file, pp_3gev_10s_pp_001_11.root, contains 250000 events. In this case we get a resolution of the method around 2%, which is only the resolution of the algorithm, and this is what we expect

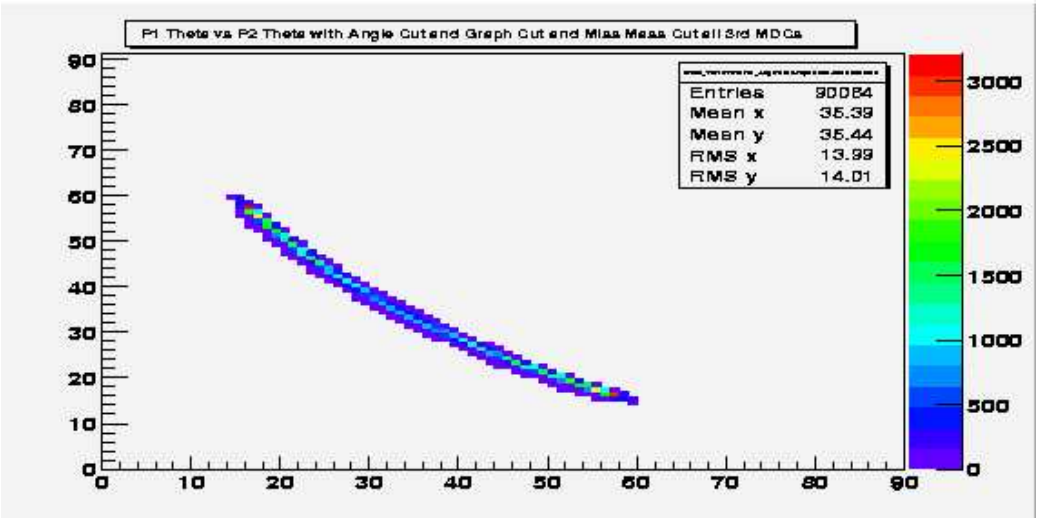
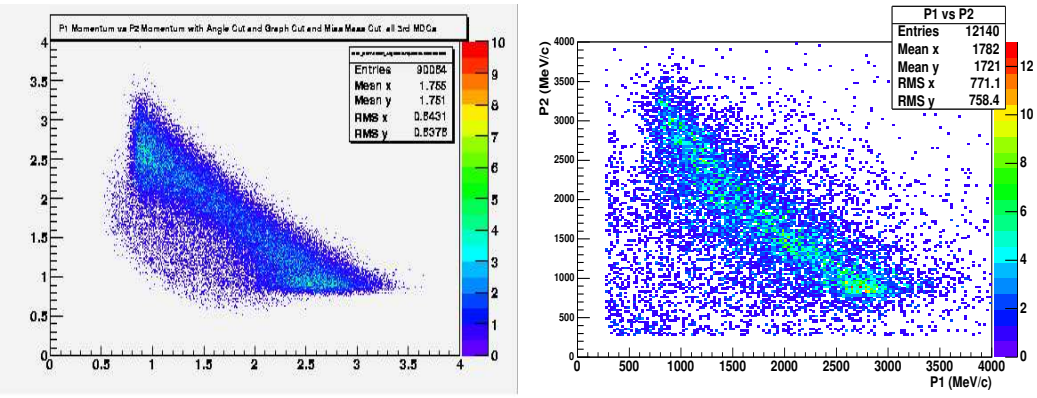


Figure 4.15: *Elastic events in Pluto:* θ_1 *vs* θ_2

for this momentum reconstruction method. Fig. 4.18 shows the comparison between the theoretical momentum from kinematics and the calculated one. The agreement (excluded some background events) is quite good with no systematic effects.



(a) (b)

Figure 4.16: *$p_{proton1}$ vs $p_{proton2}$ in the Pluto simulation case (a) and in the Sep03 data case (b)*

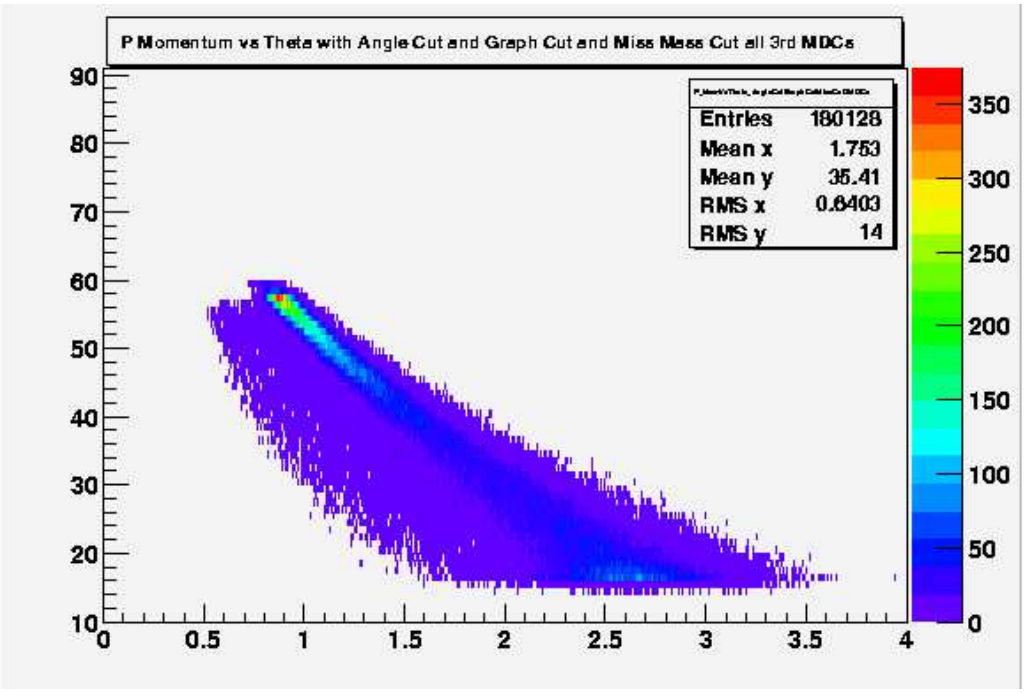
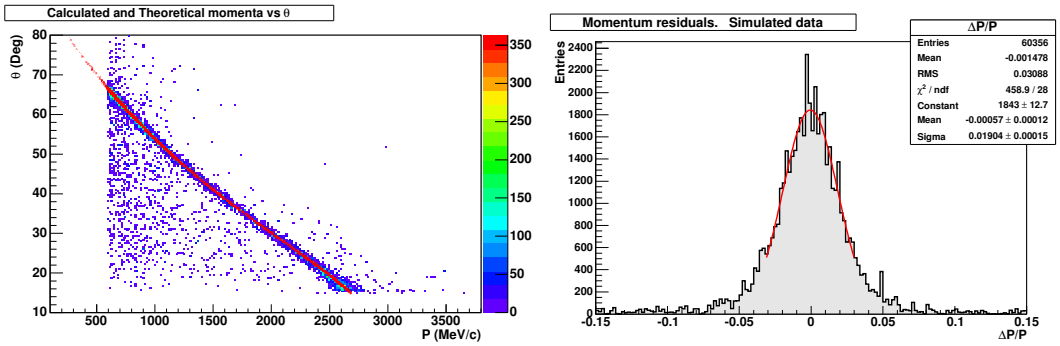


Figure 4.17: *Momentum of the elastic protons as a function of the polar angle θ in the Pluto simulation*



(a) (b)

Figure 4.18: *(a) Calculated and theoretical momenta in the HGeant simulation. The simulated data values are under the red curve (theoretical momentum) in this picture due to the good resolution. (b) Momentum residuals for this case. The resolution is* $\sigma \approx 2\%$

Conclusions

In this work, several tasks in order to a better understanding of momentum reconstruction methods in the HADES experiment were performed, both in simulated and real data.

In a first half, the so called "Kick Plane" method is explained. An extensive analysis of the Kick Plane parameterization was performed, both in the so called "low resolution" Kick Plane (where we only use the two innner chambers) and in the so called "high resolution" Kick Plane (where three MDCs are used).

The kick surfaces were calculated for different setups in the spectrometer, using the HGeant simulation packages. Results show the change in these surfaces when we change the detector's geometry. Then, a new kicksurface parameterization on each beam time would be useful. The change is bigger in the high resolution case.

The influence of the HADES magnetic field in the Kick Plane parameters was studied. The behaviour of the three main parameters, A, B and C, is different on each case. While A is completely linear with magnetic field, B and C are not. It makes imposible to parameterizate the full parameters set as a function of magnetic field.

The dependence of Kick Plane parameters with geometry was also studied. In a first step, the dependency with chamber's position was studied, showing the results how the resolution of the method do not change, keeping it around 3% in the high resolution case and around 10% in the low resolution case. In a second step, the so called misalignment was introduced, showing how a small misalignment produces big errors not only in momentum resolution, but also in mass spectra resolution. Simulated data were used in both steps.

A theoretical improvement was introduced in the code. It was adapted to deal with several target positions and parameters sets, but results shown a very small dependency of the Kick Plane with target position which made not useful the solution tested. It was tested both in simulated and in real data, getting in this case mass spectra in the so called Nov02 beamtime.

In a second half, a test of the "Reference Trajectories" method was presented. This method was tested in *proton-proton* collisions at 2.2 GeV of the Sep03 and Jan04 runs, after the appropiate code corrections. A precise selection of elastic events coming from the *p-p* interaction was performed. Using kinematics, the momentum of the elastic scattered protons is known in advance. The momentum was also calculated with the algorithm and compared it then with the calculated using kinematics. An energy loss correction was aplied in the results comming from the algorithm. The comparison allowed to estimate a number in the momentum reconstruction procedure, reaching the 10% in these two sets of data. Correcting systematic errors, a further improvement of the method can be developed, reaching so a better resolution . Another quantities like invariant mass or missing mass of the elastic pair were also calculated.

Results also show the different quality of the both *p-p* data sets, being much better in the Sep03 case. These results were compared also with simulated data.

Appendix A

Tracks' deflection under the HADES magnetic field

The good understanding of particles' deflection under our spectrometer's magnetic field is always needed in order to improve not only detectors' design[1] but also all the event reconstruction software and therefore the momentum reconstruction methods.

For this purpose, a simulation in the HGeant package was performed. Actually, since the magnetic field is sensitive to the charge of the particles, two different simulations were performed, one for positively charged particles and another one for negatively charged ones; positrons in the first case and electrons in the second one. The features of these simulatiuons were:

- Only electrons and positrons going through the spectrometer in HGeant
- One million events with e^- and one million events with e^+
- One lepton per event
- Full magnetic field strength
- Uniform distribution of particles shot in $\cos\theta$ and in ϕ
- $1/p$ momentum distribution (see Fig. A.1)
- Any secondary particles
- Target point at HADES main frame $(0, 0, 0)$

[1] Like the new RPC wall

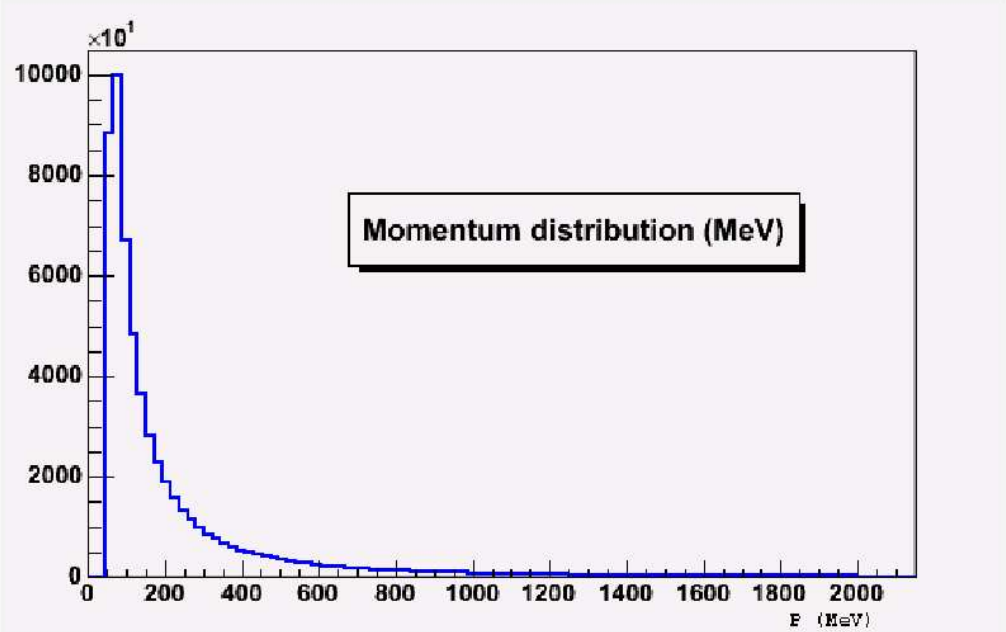


Figure A.1: *$1/p$ momentum distribution of leptons*

A.1 The HADES magnetic field

As we introduced in section 1.3.4, the HADES magnetic field [KKS95], [Bre99] is produced by a toroidal superconducting magnet composed of six coils, each coil in the edge of a sector forming the hexagonal symmetry. The coils consist in two straight sections connected via two arcs. The straight exit section covers a range of 22º to 82º. The "active" range on which particle tracking takes place covers a range of 18º to 85º. A cross section of a coil is displayed in Fig. A.2. In Fig. A.3 we can see the details of the full magnetic field system.

The current/coil amounts to 485000 $A \cdot turns$ wich means a maximum current value around 3500 A about 140 turns. The magnetic field reaches its maximum at the inner part of the arc connecting end-points of the straight sections. This peak amounts up to 3.75 T. Countour plots of the magnetic field at different azimuthal angles are shown in Fig. A.4. The high field region is shown in detail in Fig. A.5

The magnetic field support also has a cryogenic system consisting in a liquid Helium and liquid Nitrogen cooling circuit. Operating temperatures of the magnet range typically from 20º to 30º.

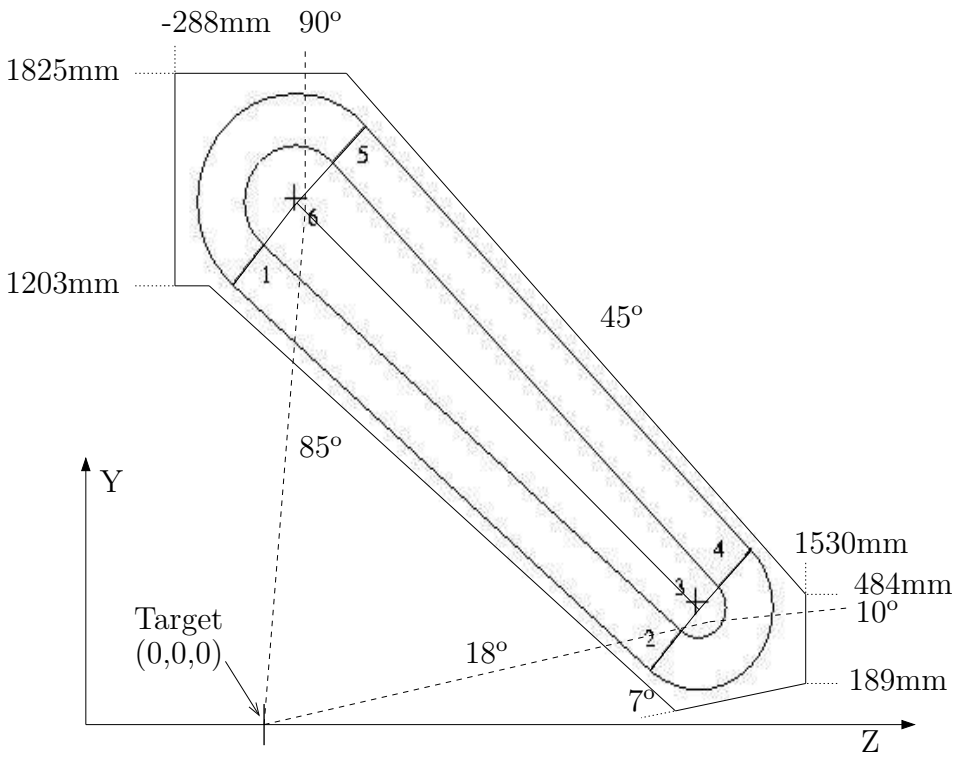
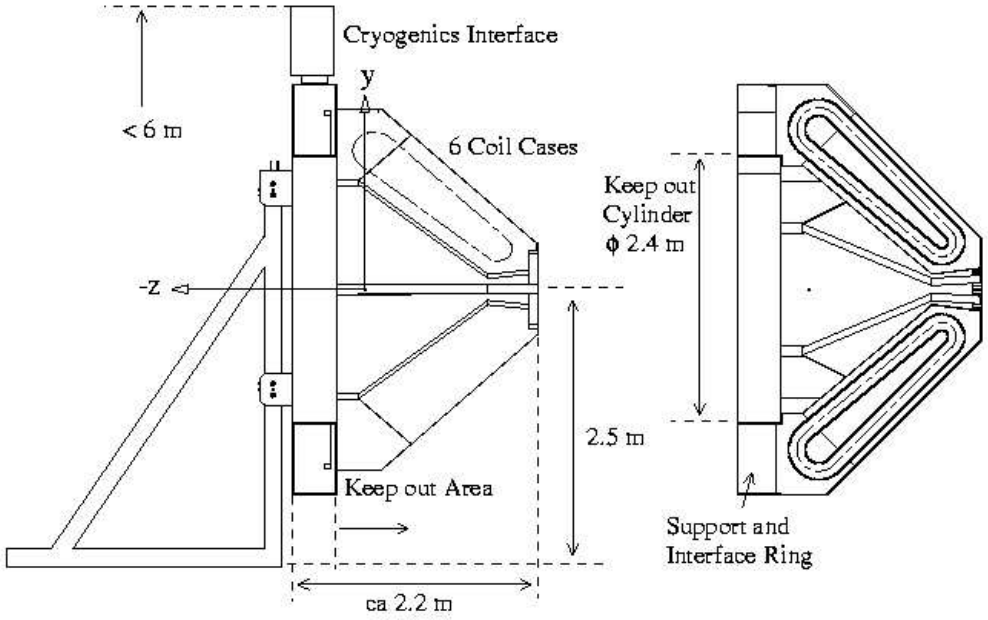


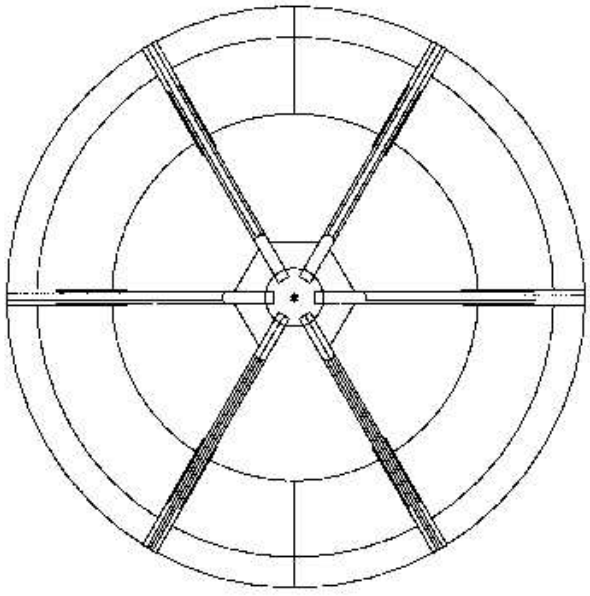
Figure A.2: *Shape of the superconducting coil; each coil is placed at azimuthal angles of* -90^o, -30^o, 30^o, 90^o, 150^o *and* 210^o.

A.2 Azimuthal deflection

The particles' deflection under the HADES' magnetic field is not a completly well studied effect. Our toroidal field produces both polar and azimuthal deflection of particles, in different direction depending on the charge of the particle. The polar deflection has been studied during these years by the HADES Colaboration. It is well understood how negatively charged particles are bending upwards by the magnetic field; that is, they go to the upper polar angles. On the other hand, positively charged particles are bending downwards, in the direction of the HADES beam axis (see Fig. A.6). Nevertheless, the azimuthal deflection is not completly understood yet. The small study presented here is focused in such deflection. Lets take the electron and the positron as an example of negatively charged particle and positively charged particle respectively. Lets also define $\Delta\eta$ as the total deflected angle, that is, the angle forming by the mometum vectors before and after the magnetic field (Fig. A.7)



(a)



(b)

Figure A.3: *(a) Vertical (left) and horizontal (right) cut through the magnet including schematically coil cases, support ring and front support hexagon. (b) Schematic front view of the torus*

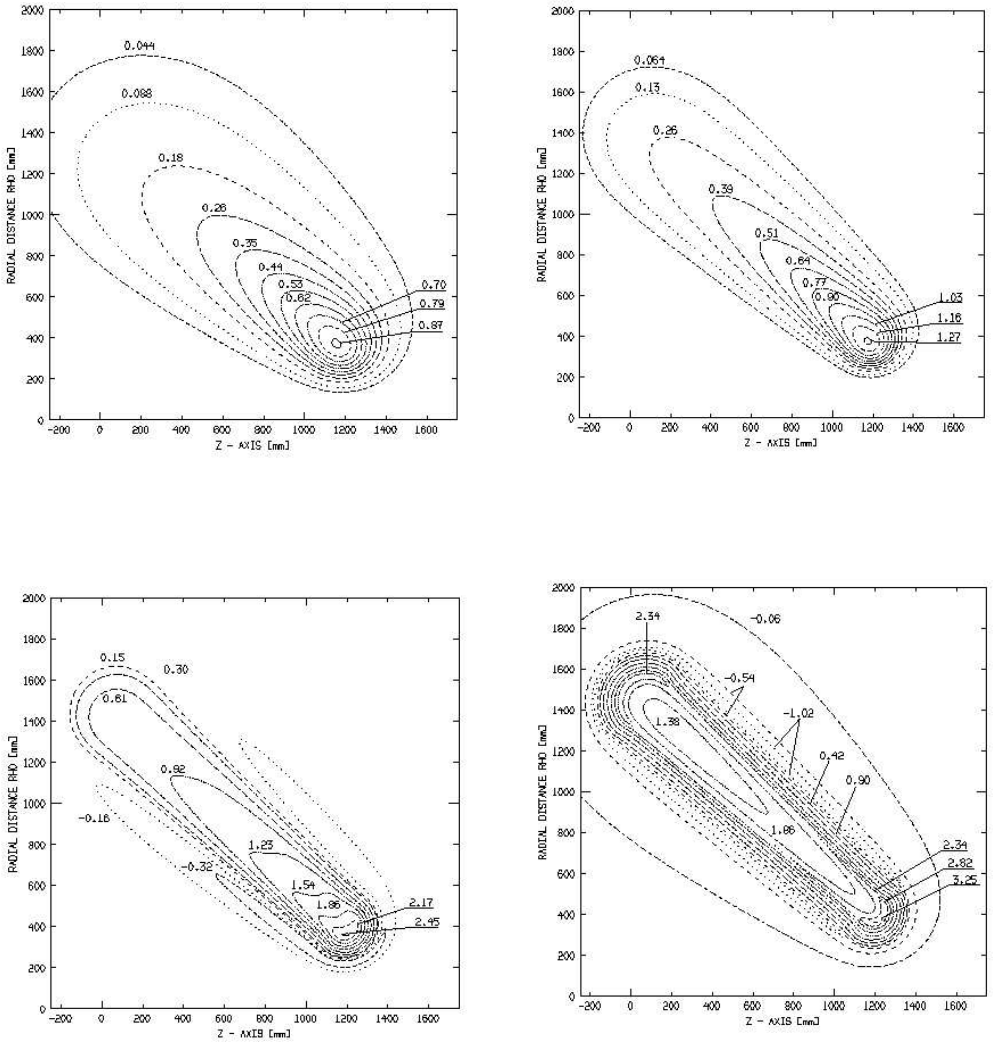


Figure A.4: *Main magnetic field component in between two sectors at azimuthal angles of* 0^{o} *(top, left),* 15^{o} *(top, right),* 25^{o} *(bottom, left) and* 30^{o} *(bottom, right). The field strength is specified in Tesla.*

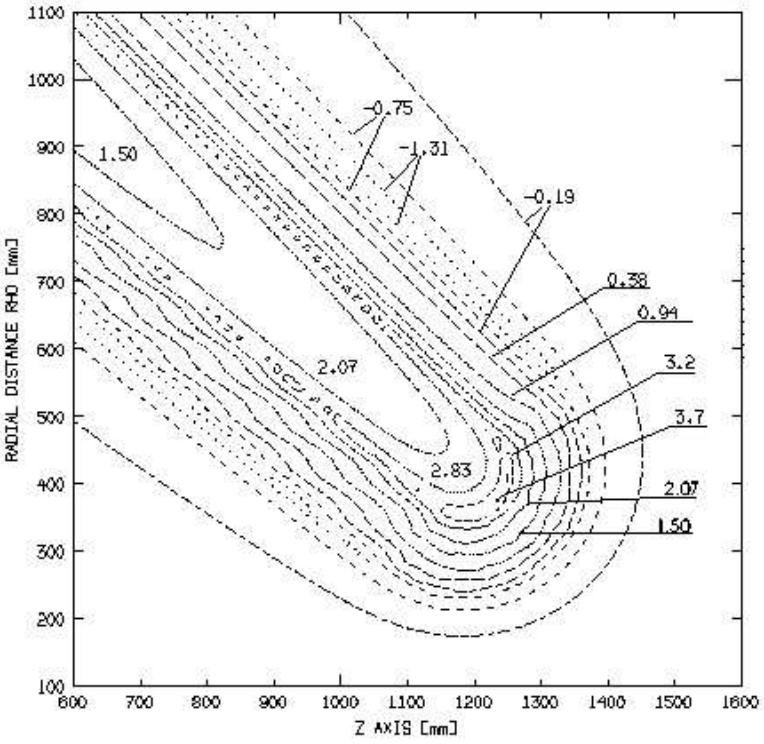


Figure A.5: *Field map in the midplane of the coil. The high field region in shown*

A.2.1 Negatively charged particles

As well as electrons are bending upwards in the polar direction, it is logical to think that they are going to be bent away from the center of the sectors in the azimuthal direction. In fact, pictures shown here tell us how it is so. In Fig. A.8 we can see the azimuthal momentum kick given by the field. It is clear that particles passing by the field region near the coils suffer a bigger kick than those passing by the center of the sectors. Then, the closer they are to the edges of the field, the bigger the change in the azimuthal angle (see also Fig. A.9 and Fig. A.10). Of course it is also clear that the lower momenta the bigger deflection.

A.2.2 Positively charged particles

On the other hand, positively charged particles should be bent in the opposite diretion than negatively charged ones. Then, in azimuthal direction, they are bent towards the center of the sector. Although these particles have a shorter path inside the field region, they cross exactly the high field regions.

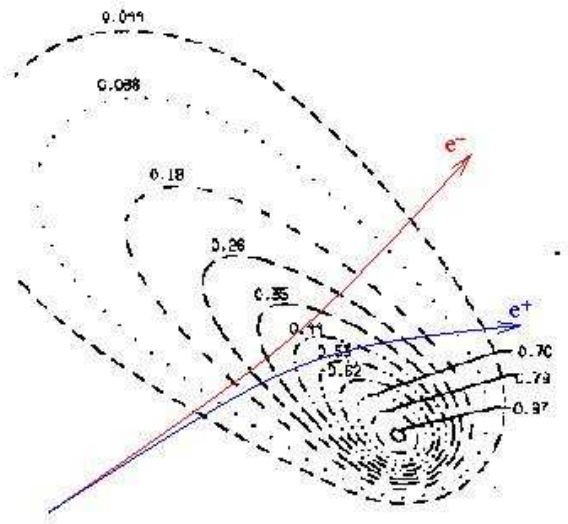


Figure A.6: *Polar deflection in a magnetic field transversal cut at the center of a sector*

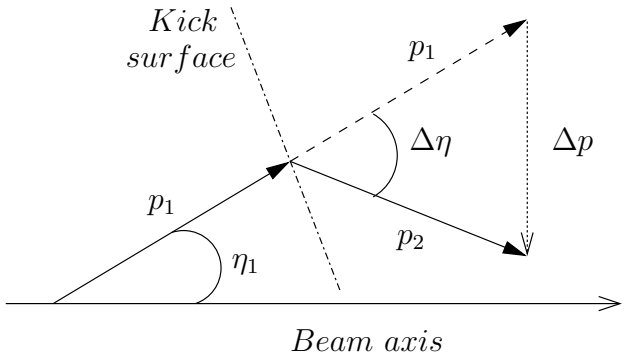


Figure A.7: *Definition of the total $\Delta\eta$ deflection*

Then, the kick suffered by positrons is, in general, bigger than the one suffered by electrons. It is easy to check these effects having a look at Figs. A.11, A.12 and A.13. It is clear from them how positively charged and negatively charged particles have a completly oppposite behaviour under the magnetic field.

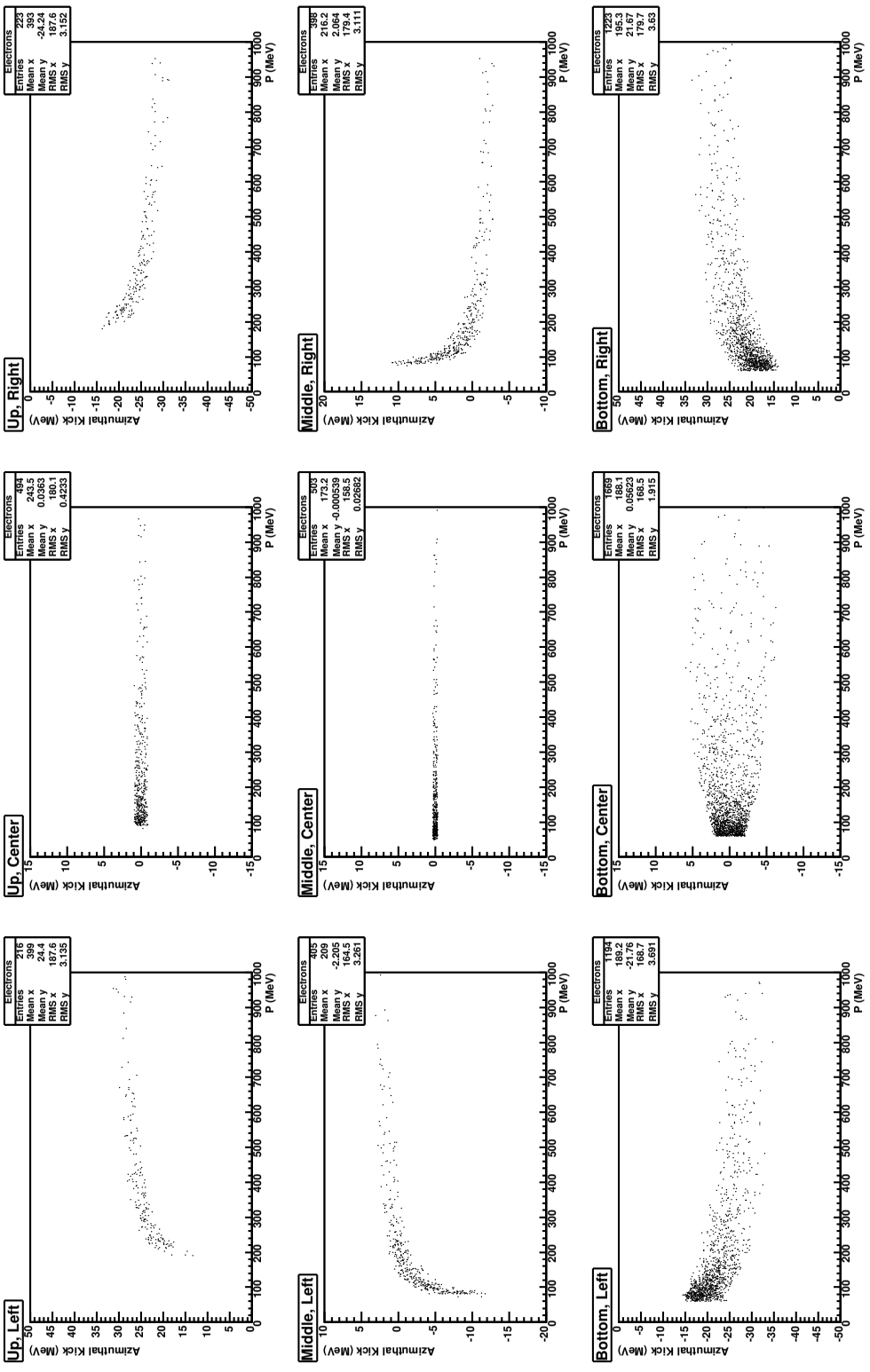


Figure A.8: *Azimuthal momentum kick for electrons at nine different regions on the kick surface, from top of the surface to the bottom, and from left side (negative ϕ) to right side (positive ϕ)*

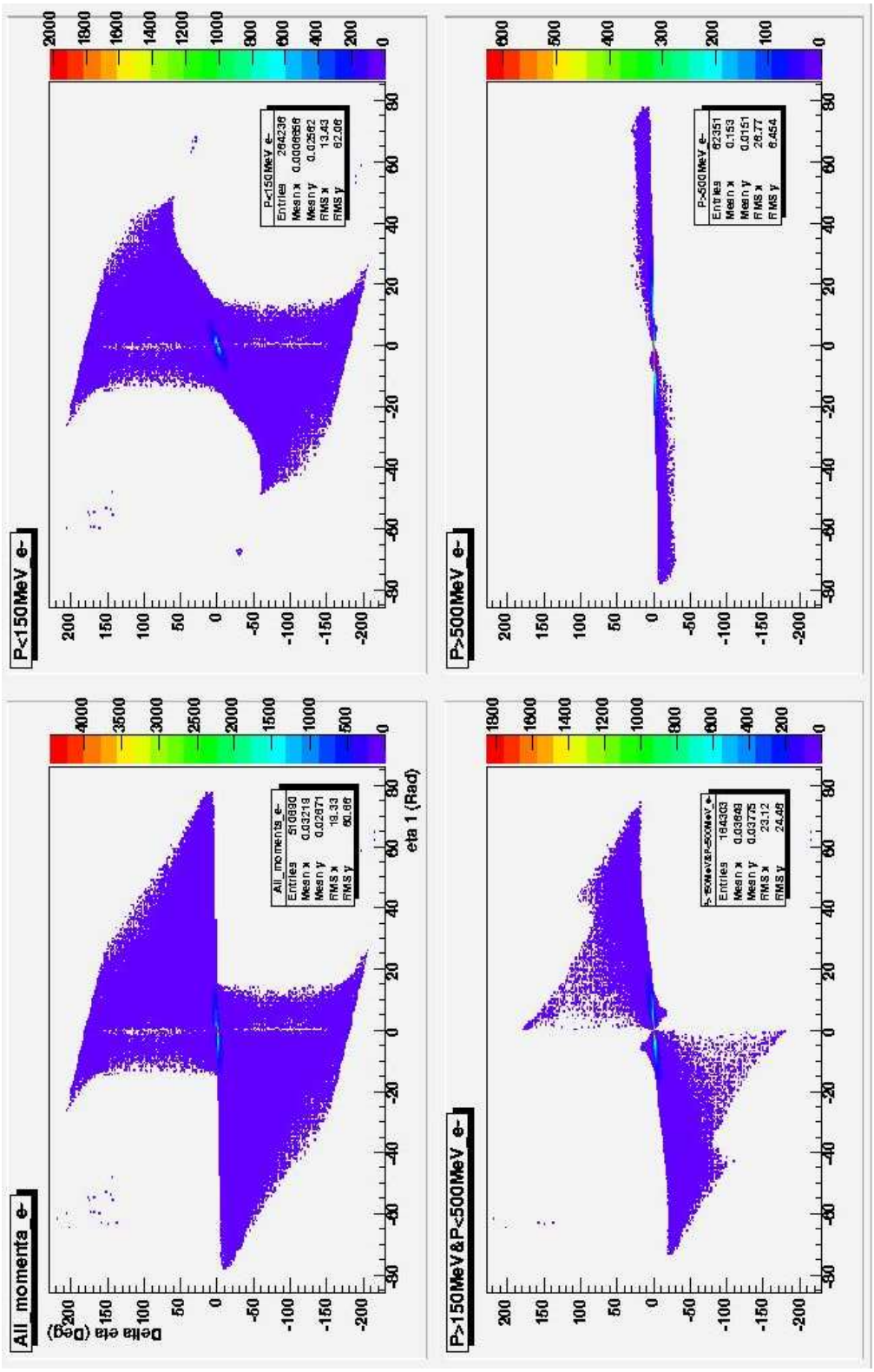


Figure A.9: *Total deflection $\Delta\eta$ for electrons as a function of incident angle η_1 in different momentum regions*

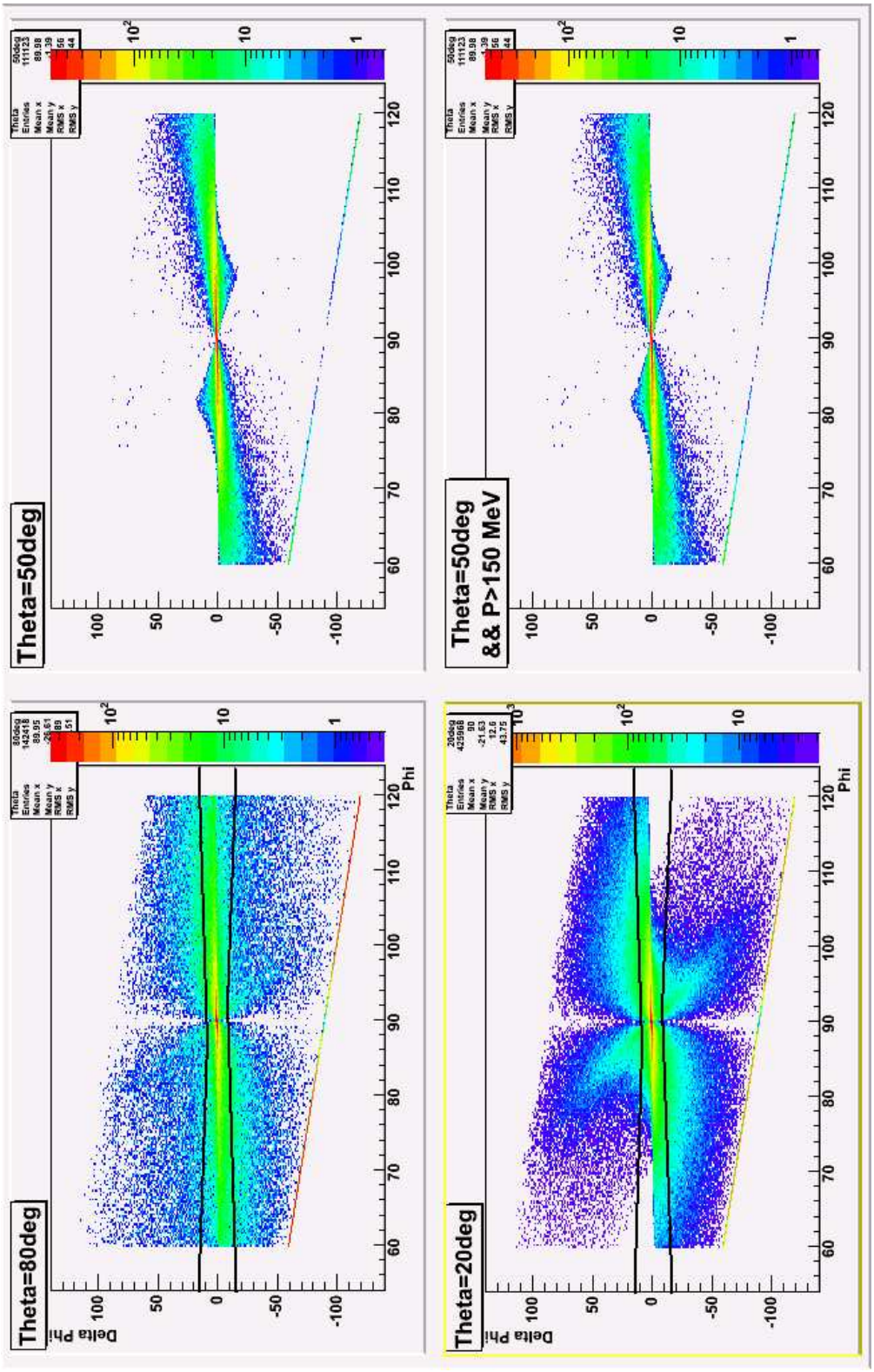


Figure A.10: *Change in azimuthal angle as a funtion of azimuthal incident angle for electrons in three different polar regions. The black lines correspond to the cut in the Matching Unit Second Level Trigger System*

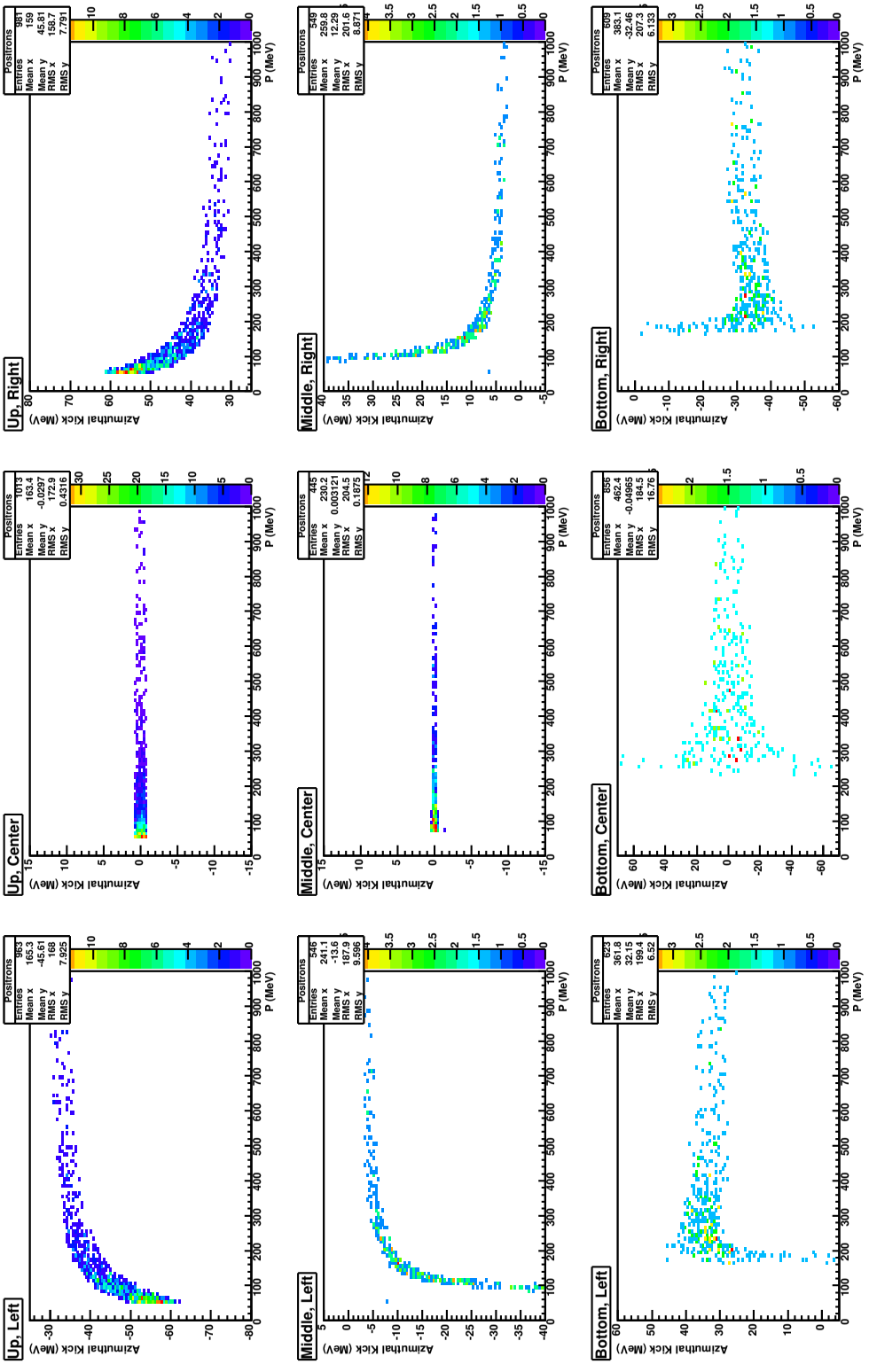


Figure A.11: *Azimuthal momentum kick for positrons at nine different regions on the kick surface, from top of the surface to the bottom, and from left side (negative ϕ) to right side (positive ϕ). It is clear the different behaviour of the momentum kick between this picture and Fig. A.8*

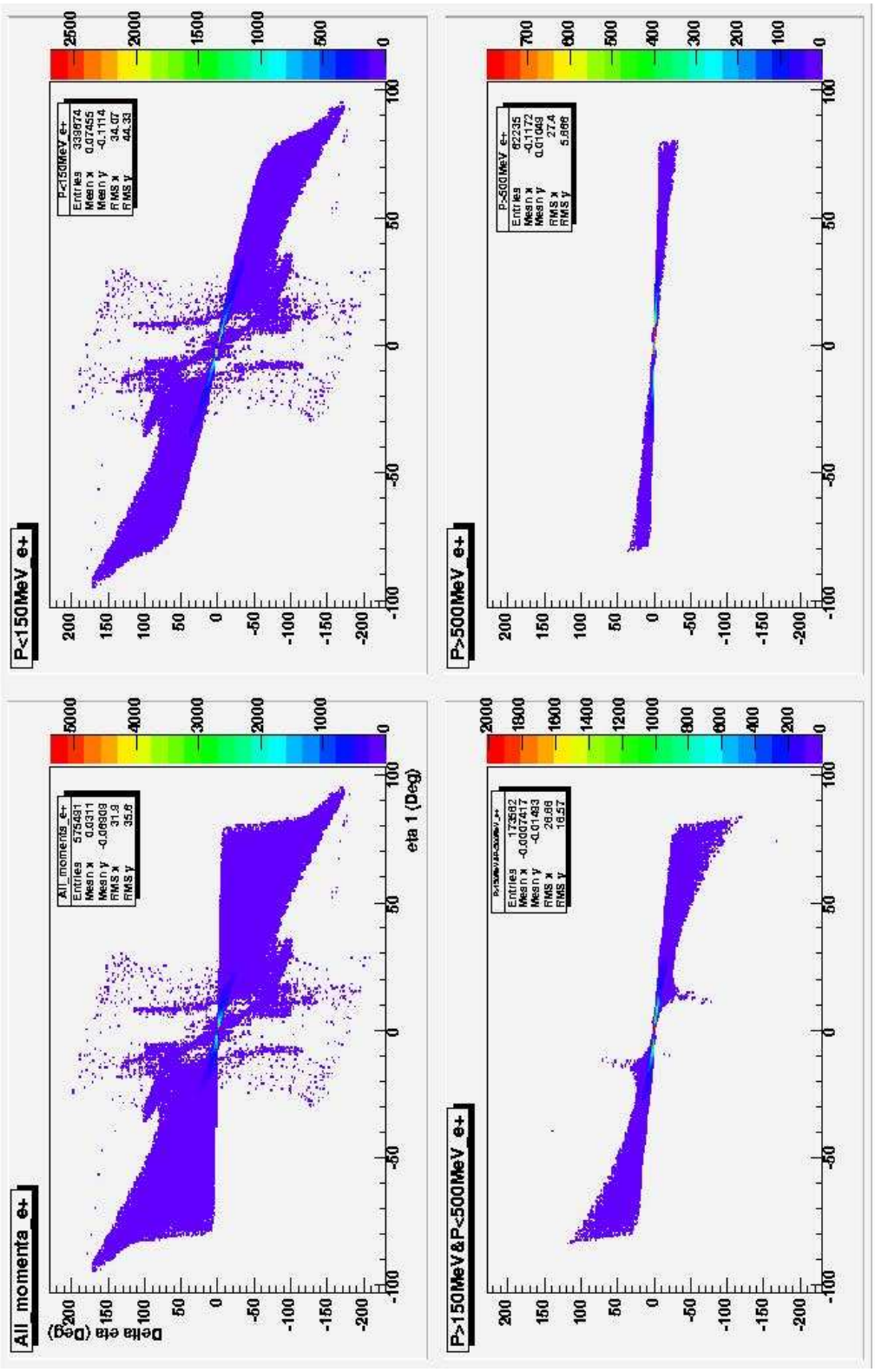


Figure A.12: *Total deflection $\Delta\eta$ for positrons as a function of incident angle η_1 in different momentum regions*

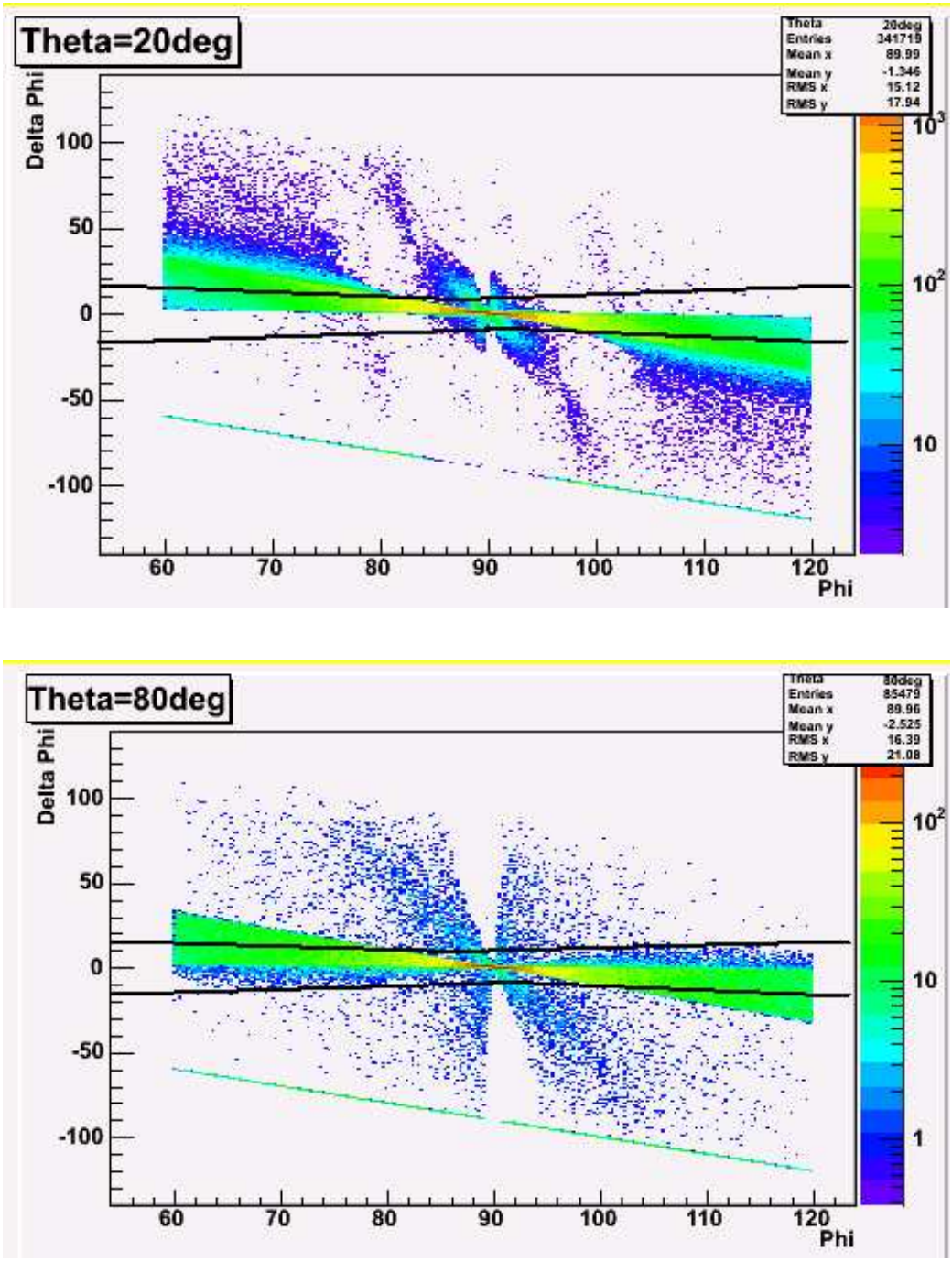


Figure A.13: *Change in azimuthal angle as a funtion of azimuthal incident angle for positrons in two different polar regions. The black lines correspond again to the cut in the Matching Unit Second Level Trigger System*

A.3 Path through the HADES spectrometer

We already know how the effect of our magnetic field over the two different kind of detectable particles is. It is an interesting check to see the path of electrons and positrons along the spectrometer. These paths are shown in Figs. A.14 and A.16 (a) in the case of electrons and in Figs. A.15 and A.16 (b) in the positrons case.

Figs. A.14 and A.15 show the distribution of electrons and positrons respectively in the four drift chambers. Before the magnetic field, both particles follow the same path, but after cross it, positrons are focused to the center of the sector while electrons are defocused away from it. Although it is the opposite effect, the final state is the same: mostly of particles are near the center of the sectors. It is easy to realize why in the case of positrons, but in the case of electrons, what is happening is that those electrons which are near the magnetic coils are throwing away from the MDCs acceptance, while those which are close to the center of the sector are only slightly bent, so they remain there.

Fig. A.16 shows the distribution of both particles at the SHOWER plane. We have much more positrons than electrons because electrons are bending upwards in the polar direction, so most of them fall in the TOF region.

An important consequence of this effect is the fact that, if we put the logical condition of hit in the META detectors, we are losing those particles that, due to the magnetic field effects, are bent away from the spectrometer acceptance, hitting for example in the MDCs' frames, or reaching angles bigger than 85º or lower than 18º. Fig. A.17 shows the momentum distribution of all tracks and also those tracks which had a hit in any MDC but not in the META detectors. We are losing about 33% of electrons and about 26% of positrons, all of them being very low momenta particles.

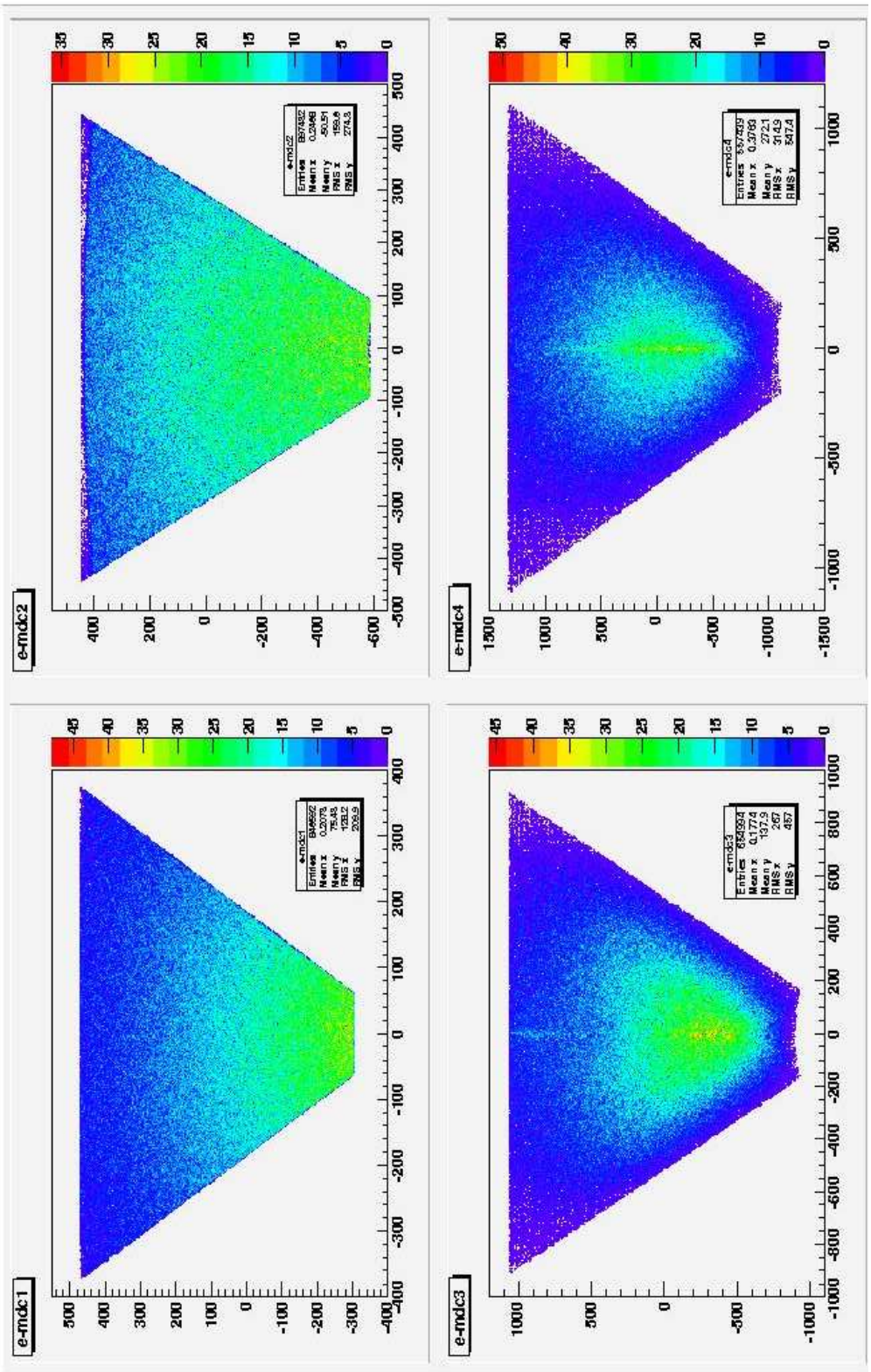


Figure A.14: *Distribution of electrons in the four drift chambers*

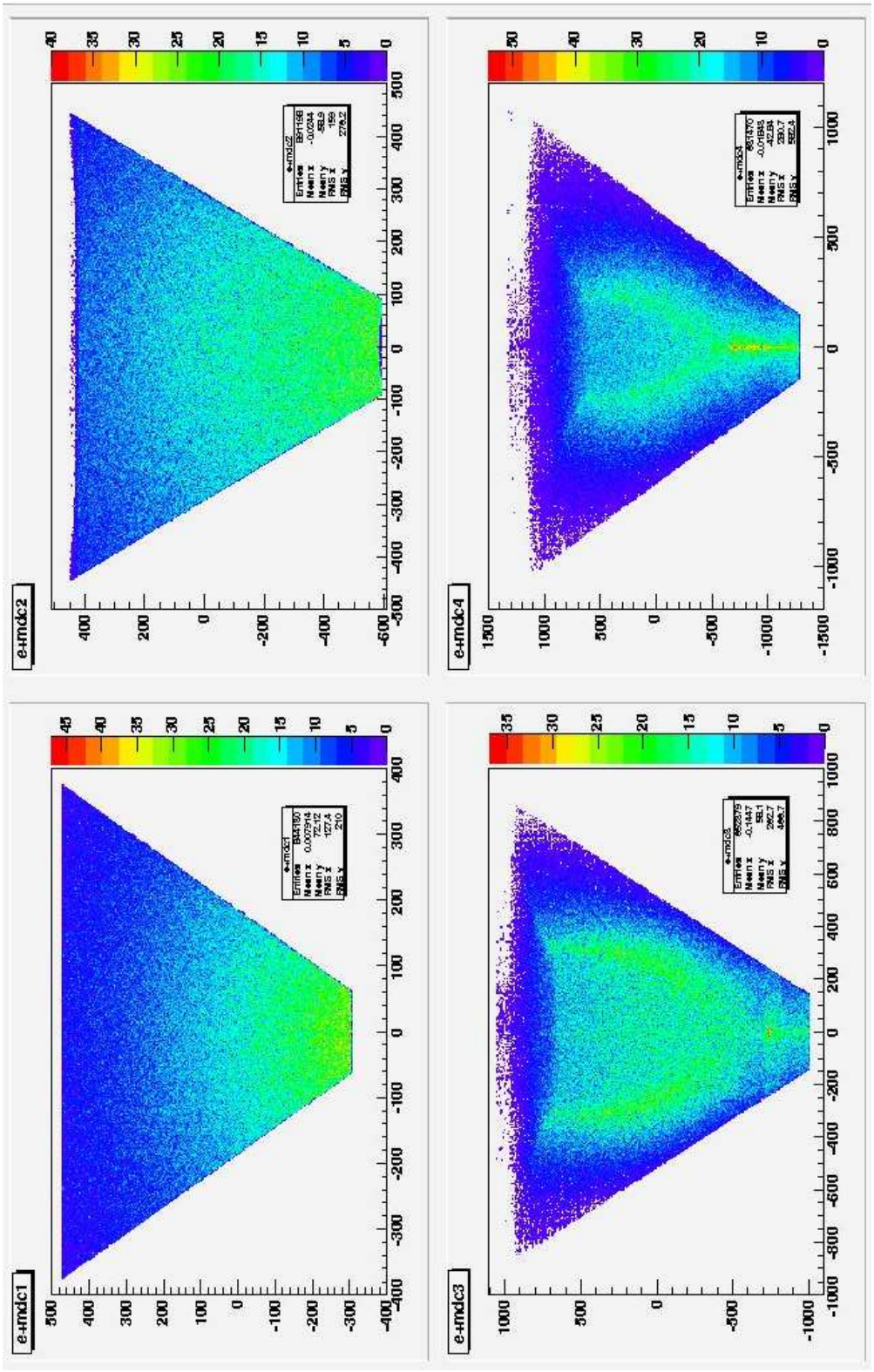
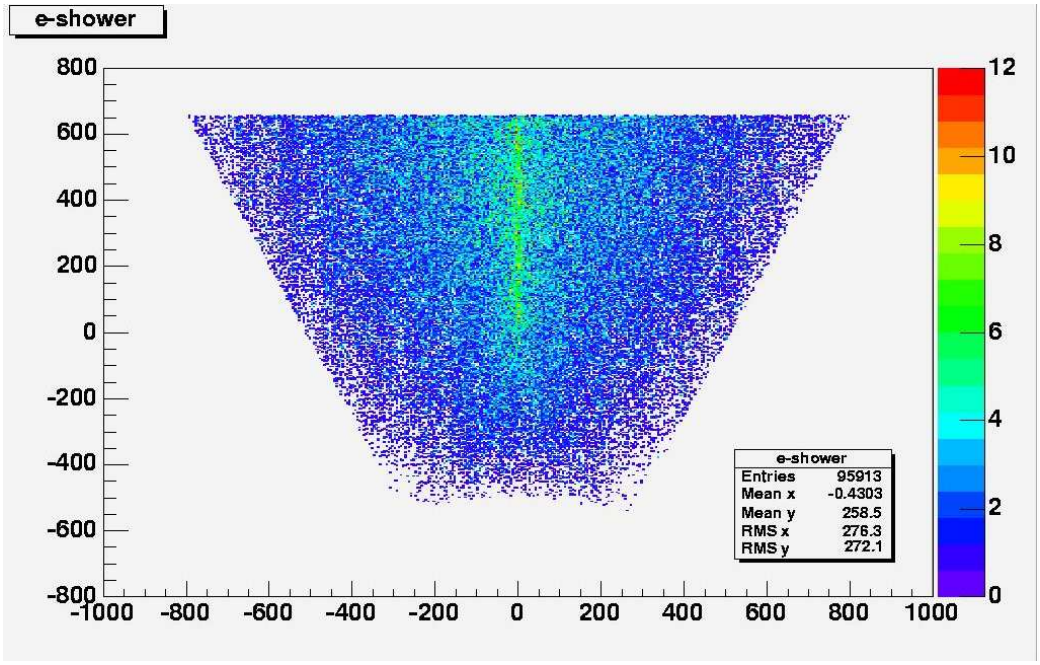
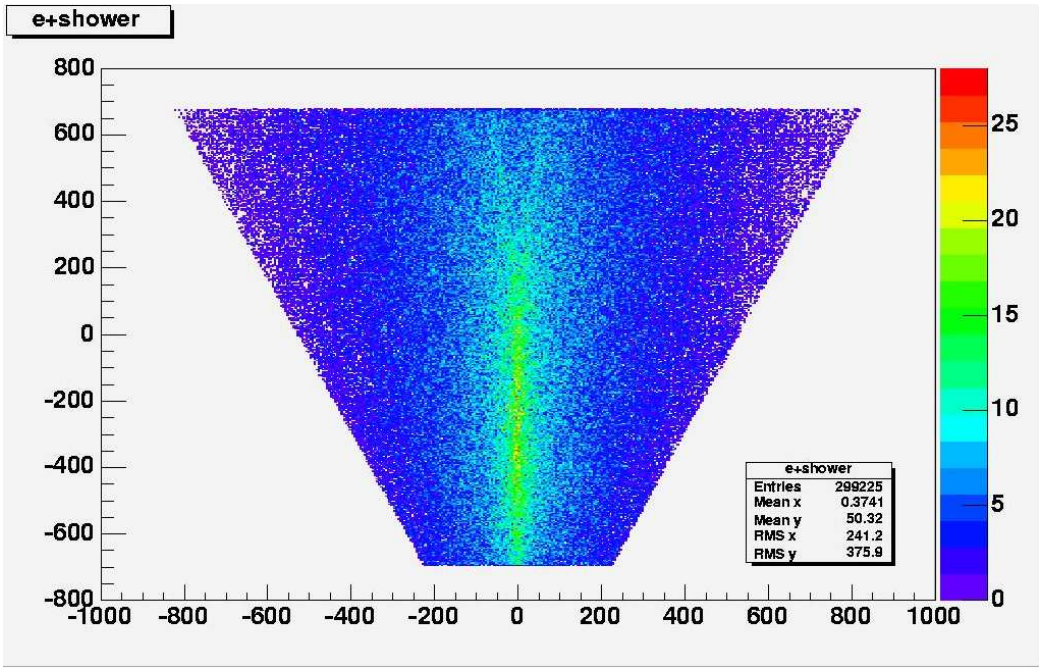


Figure A.15: *Distribution of positrons in the four drift chambers*

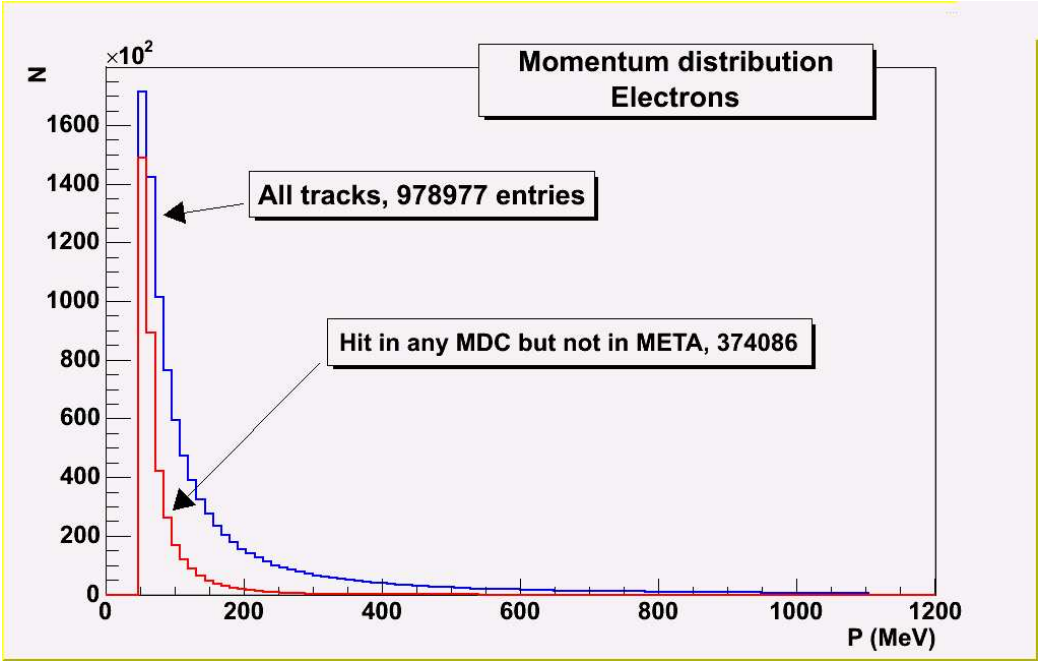


(a) Electrons

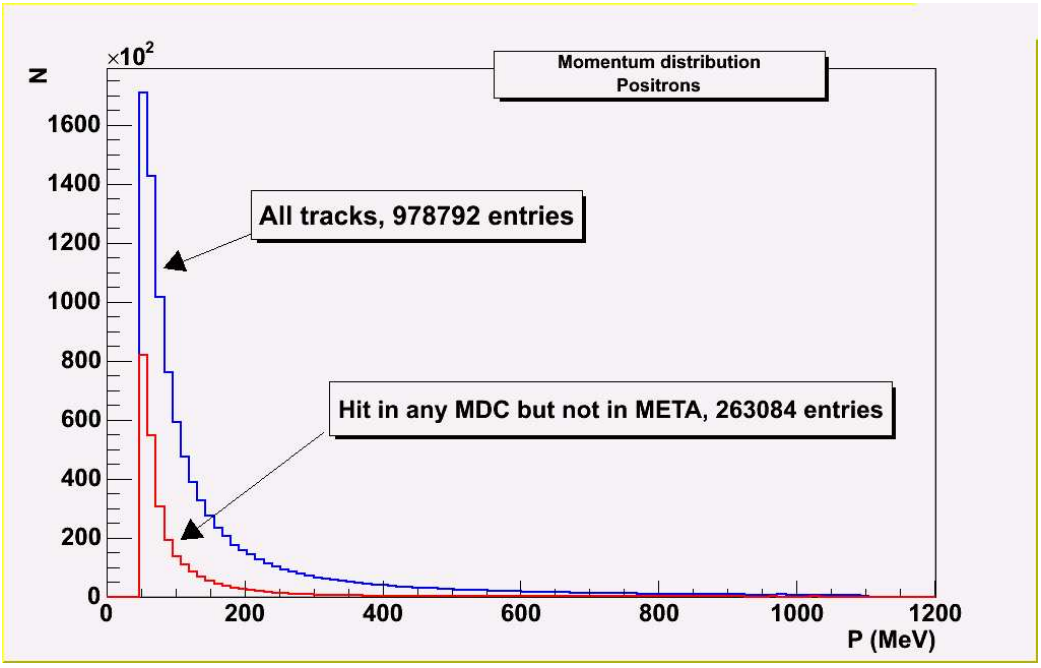


(b) Positrons

Figure A.16: *Distribution of electrons (a) and positrons (b) at the SHOWER plane*



(a) Electrons



(b) Positrons

Figure A.17: *Momentum distribution for electrons (a) and positrons (b) separating all tracks and those with hit in any MDC but not in META detectors*

Appendix B

Energy loss of protons in HADES

Relativistic charged particles lose energy in matter primarily by ionization and atomic excitation. The mean rate of such energy loss (or stopping power) is given by the well known Bethe-Bloch equation [ea04b], [PDG]:

$$-\frac{dE}{dx} = Kz^2\frac{Z}{A}\frac{1}{\beta^2}\left[\frac{1}{2}\ln\frac{2m_ec^2\beta^2\gamma^2T_{max}}{I^2} - \beta^2 - \frac{\delta}{2}\right]$$

which is basically a function of the charge z and the velocity β of the incident particle and the atomic number over mass number ratio of the medium, Z. The heavier the medium the bigger the energy loss. On the other hand, the slower the particle the bigger also the energy loss. The different symbols in this equation are defined in Table B.1.

T_{max} is the maximum kinetic energy which can be transfered to a free electron of the medium in a single collision. For a particle with mass M and momentum $p = M\beta\gamma c$, T_{max} is given by

$$T_{max} = \frac{2m_ec^2\beta^2\gamma^2}{1 + 2\gamma m_e/M + (m_e/M)^2}$$

The Bethe-Bloch equation may be integrated to find the total (or partial) "continuous slowing-down approximation" range R for a particle which loses energy only through ionization and atomic excitation. Since for a given medium, dE/dx depends only on β, R/M is a function of E/M. R/M as a function of $\beta\gamma = p/Mc$ is shown for a variety of materials in Fig. B.1.

In the HADES Sep03 and Jan04 beamtimes cases, protons were the incident particles. We need to know in advance the energy loss of those protons in order to correct the calculated momentum values; that is, we are interested

Symbol	*Definition*
E	Energy of the incident particle
x	Mass depth
$m_e c^2$	Electron mass in MeV
z	Charge of the incident particle
Z	Atomic number of medium
A	Atomic mass of medium
K	$4\pi N_A r_e^2 m_e c^2$
N_A	Avogadro's number
r_e	Classical electron radius
I	Mean excitation energy
δ	Density effect correction to ionization energy loss
β	Velocity of the incident particle
γ	Lorentz factor of the incident particle
T_{max}	Maximum T transferred to an electron in a collision

Table B.1: *Symbols in the Bethe-Bloch formula*

in the variation in momentum induced by energy loss. The "momentum loss" can be triviallly obtained from the energy loss, since that

$$\frac{dp}{dx} = \frac{dp}{dE}\frac{dE}{dx} = \sqrt{1 + \left(\frac{m}{p}\right)^2}\frac{dE}{dx}$$

We can see how the variation in momentum due to energy loss, for a given particle, depends only on the particle's momentum, once we know the Stopping Power.

Our protons lose energy (or momentum) mainly in the target itself, after the primary interaction. As we have explained in section 4.3 our target consists in liquid hydrogen within a very thin aluminium pipe, containing $10^{26} atoms/cm^3$. But protons not only lose energy in the LH_2 or in the Al, but also in the RICH radiator gas, C_4F_{10}, RICH VUV mirror, which is 2mm thickness of pure Carbon, and inside the Drift Chambers. This energy loss becomes more important in the case of slow anf light particles, like pions.

In our *pp* collisions, we produce fast protons. Their momentum goes from 500 MeV up to 3 GeV. This means a not too big energy loss. Taking into account all the formulae and Fig. B.1 we can estimate the amount of lost momentum inside the target. For example, a 500 MeV proton would suffer:

$$\frac{dE}{dx}\left[\frac{MeV \cdot cm^2}{g}\right] \approx 30 MeV \left[\frac{MeV \cdot cm^2}{g}\right]$$

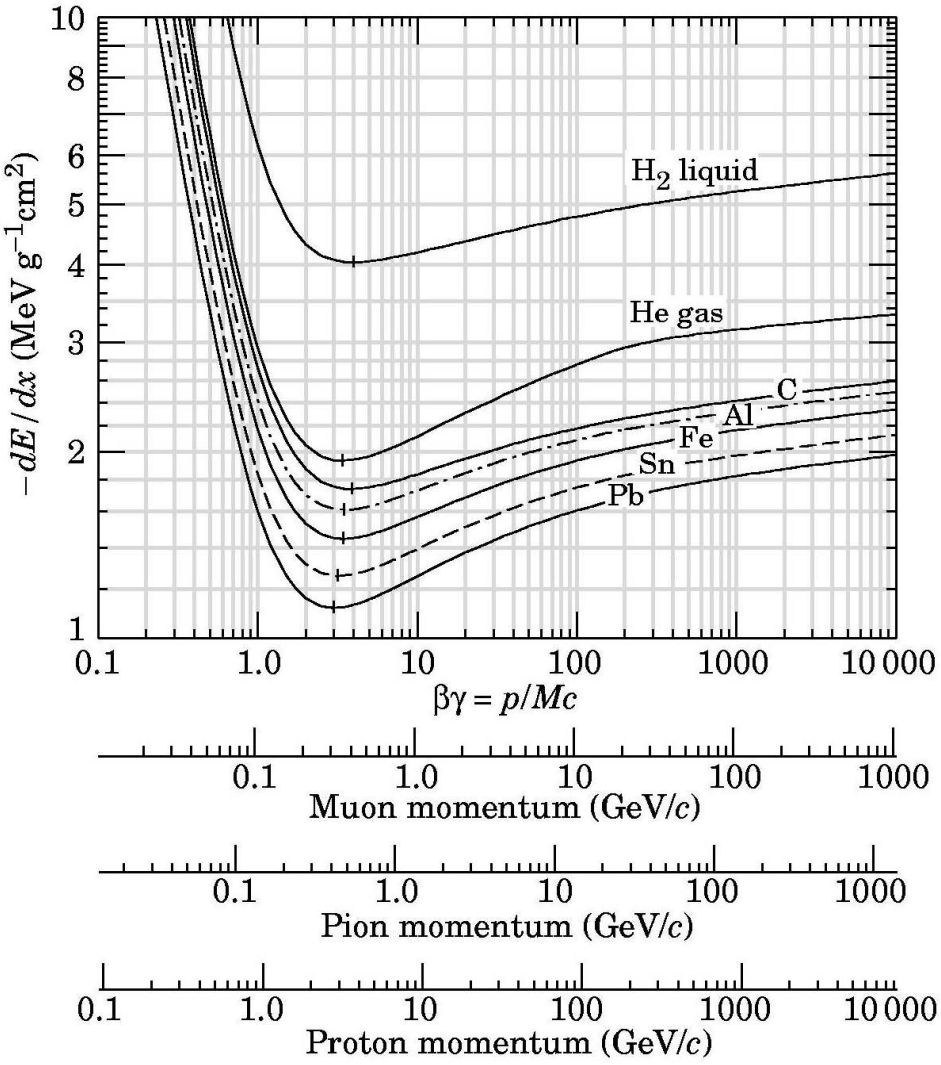


Figure B.1: *Mean rate of Energy loss in variuos materials for different particles and momenta*

(from Fig. B.1). Density of $10^{26} atoms/cm^3$ of LH_2 means $1.6 \cdot 10^{-1} g/cm^3$, since $m_H = 1.672 \cdot 10^{-27} kg$. If the proton is produced at the beginning of the target, it travels at least 5cm of material. Using this,

$$\frac{dE}{dx}[MeV] = 30MeV \left[\frac{MeV \cdot cm^2}{g}\right] \cdot 0.16 g/cm^3 \cdot 5cm \approx 24MeV$$

Then,

$$\frac{dp}{dx} = \sqrt{1 + \left(\frac{938MeV/c^2}{500MeV/c}\right)^2} \cdot 24MeV \approx 50MeV/c$$

which means 10% of the original value. This is not exactly the maximum momentum loss for our protons because is also depends on the direction of the produced proton. The track length inside the target changes with the polar angle. A high polar track can travel more than 5cm inside the LH_2, so the energy loss could be bigger.

On the other hand, the same proton with 2000 MeV would lose about 4 MeV, while a proton with 3 GeV would lose less than 3 MeV, meanning less than 1% of the original momentum value, which becomes negligible.

Appendix C

Example of a ROOT analysis macro

C.1 Header file

```
#ifndef prueba_h
#define prueba_h

#include <TROOT.h>
#include <TChain.h>
#include <TFile.h>

   const Int_t kMaxHRtMdcTrk = 1;
   const Int_t kMaxHRtMdcTrk_fData = 48;
   const Int_t kMaxTracks = 1;
   const Int_t kMaxHKickTrack = 1;
   const Int_t kMaxHKickTrack_fData = 10;
   const Int_t kMaxHKickTrackB = 1;
   const Int_t kMaxHKickTrackB_fData = 10;

class prueba {
public :
   TTree          *fChain;   //!pointer to the analyzed TTree or TChain
   Int_t           fCurrent; //!current Tree number in a TChain

   // Declaration of leave types

   HMatrixCategory *HRtMdcTrk.;
   UInt_t          HRtMdcTrk_HCategory_fUniqueID;
   UInt_t          HRtMdcTrk_HCategory_fBits;
   Short_t         HRtMdcTrk_HCategory_fCat;
   Int_t           HRtMdcTrk_HCategory_fBranchingLevel;
   Int_t           HRtMdcTrk_fNDataObjs;
   Int_t           HRtMdcTrk_fData_;
   UInt_t          HRtMdcTrk_fData_fUniqueID[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   UInt_t          HRtMdcTrk_fData_fBits[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Float_t         HRtMdcTrk_fData_z[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Float_t         HRtMdcTrk_fData_r[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Float_t         HRtMdcTrk_fData_p[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
```

```
   Float_t         HRtMdcTrk_fData_theta[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Float_t         HRtMdcTrk_fData_phi[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Int_t           HRtMdcTrk_fData_charge[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Float_t         HRtMdcTrk_fData_chi2[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   UInt_t          HRtMdcTrk_fData_cov_fUniqueID[kMaxHRtMdcTrk_fData];
//[HRtMdcTrk.fData_]
   UInt_t          HRtMdcTrk_fData_cov_fBits[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Int_t           HRtMdcTrk_fData_cov_size[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Int_t           HRtMdcTrk_fData_cov_dim[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Float_t         HRtMdcTrk_fData_cov_data[kMaxHRtMdcTrk_fData][15];
//[HRtMdcTrk.fData_]
   Int_t           HRtMdcTrk_fData_flag[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Int_t           HRtMdcTrk_fData_sector[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Int_t           HRtMdcTrk_fData_segIndex[kMaxHRtMdcTrk_fData][2];   //[HRtMdcTrk.fData_]
   UInt_t          HRtMdcTrk_fData_fOuterPos_fUniqueID[kMaxHRtMdcTrk_fData];
//[HRtMdcTrk.fData_]
   UInt_t          HRtMdcTrk_fData_fOuterPos_fBits[kMaxHRtMdcTrk_fData];
//[HRtMdcTrk.fData_]
   Double_t        HRtMdcTrk_fData_fOuterPos_x[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Double_t        HRtMdcTrk_fData_fOuterPos_y[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Double_t        HRtMdcTrk_fData_fOuterPos_z[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   UInt_t          HRtMdcTrk_fData_fOuterDir_fUniqueID[kMaxHRtMdcTrk_fData];
//[HRtMdcTrk.fData_]
   UInt_t          HRtMdcTrk_fData_fOuterDir_fBits[kMaxHRtMdcTrk_fData];
//[HRtMdcTrk.fData_]
   Double_t        HRtMdcTrk_fData_fOuterDir_x[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Double_t        HRtMdcTrk_fData_fOuterDir_y[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Double_t        HRtMdcTrk_fData_fOuterDir_z[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Float_t         HRtMdcTrk_fData_length[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Float_t         HRtMdcTrk_fData_dKick[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Float_t         HRtMdcTrk_fData_d[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Float_t         HRtMdcTrk_fData_dPhi[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Float_t         HRtMdcTrk_fData_leverArm[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   Int_t           HRtMdcTrk_fData_fitResult[kMaxHRtMdcTrk_fData];   //[HRtMdcTrk.fData_]
   HPartialEvent   *Tracks.;
   UInt_t          Tracks_HEvent_fUniqueID;
   UInt_t          Tracks_HEvent_fBits;
   Int_t           Tracks_fRecLevel;
   Short_t         Tracks_fBaseCategory;
   HMatrixCategory *HKickTrack.;
   UInt_t          HKickTrack_HCategory_fUniqueID;
   UInt_t          HKickTrack_HCategory_fBits;
   Short_t         HKickTrack_HCategory_fCat;
   Int_t           HKickTrack_HCategory_fBranchingLevel;
   Int_t           HKickTrack_fNDataObjs;
   Int_t           HKickTrack_fData_;
   UInt_t          HKickTrack_fData_fUniqueID[kMaxHKickTrack_fData];
//[HKickTrack.fData_]
   UInt_t          HKickTrack_fData_fBits[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Float_t         HKickTrack_fData_pTof[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Float_t         HKickTrack_fData_errPTof[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Char_t          HKickTrack_fData_pid[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Char_t          HKickTrack_fData_quality[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Float_t         HKickTrack_fData_z[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Float_t         HKickTrack_fData_r[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Float_t         HKickTrack_fData_p[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Float_t         HKickTrack_fData_theta[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Float_t         HKickTrack_fData_phi[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Float_t         HKickTrack_fData_mass[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Int_t           HKickTrack_fData_charge[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   UInt_t          HKickTrack_fData_cov_fUniqueID[kMaxHKickTrack_fData];
//[HKickTrack.fData_]
```

```
   UInt_t          HKickTrack_fData_cov_fBits[kMaxHKickTrack_fData];
//[HKickTrack.fData_]
   Int_t           HKickTrack_fData_cov_size[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Int_t           HKickTrack_fData_cov_dim[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Float_t         HKickTrack_fData_cov_data[kMaxHKickTrack_fData][21];
//[HKickTrack.fData_]
   Char_t          HKickTrack_fData_sector[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Int_t           HKickTrack_fData_system[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Float_t         HKickTrack_fData_tof[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Float_t         HKickTrack_fData_metaeloss[kMaxHKickTrack_fData];
//[HKickTrack.fData_]
   Float_t         HKickTrack_fData_beta[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Float_t         HKickTrack_fData_showerSum10[kMaxHKickTrack_fData];
//[HKickTrack.fData_]
   Float_t         HKickTrack_fData_showerSum20[kMaxHKickTrack_fData];
//[HKickTrack.fData_]
   Short_t         HKickTrack_fData_segmentId[kMaxHKickTrack_fData];
//[HKickTrack.fData_]
   Short_t         HKickTrack_fData_ringId[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Short_t         HKickTrack_fData_outerHitId[kMaxHKickTrack_fData];
//[HKickTrack.fData_]
   Float_t         HKickTrack_fData_pull[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   Float_t         HKickTrack_fData_flag[kMaxHKickTrack_fData];   //[HKickTrack.fData_]
   HMatrixCategory *HKickTrackB.;
   UInt_t          HKickTrackB_HCategory_fUniqueID;
   UInt_t          HKickTrackB_HCategory_fBits;
   Short_t         HKickTrackB_HCategory_fCat;
   Int_t           HKickTrackB_HCategory_fBranchingLevel;
   Int_t           HKickTrackB_fNDataObjs;
   Int_t           HKickTrackB_fData_;
   UInt_t          HKickTrackB_fData_fUniqueID[kMaxHKickTrackB_fData];
//[HKickTrackB.fData_]
   UInt_t          HKickTrackB_fData_fBits[kMaxHKickTrackB_fData];   //[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_z[kMaxHKickTrackB_fData];   //[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_r[kMaxHKickTrackB_fData];   //[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_theta[kMaxHKickTrackB_fData];   //[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_phi[kMaxHKickTrackB_fData];   //[HKickTrackB.fData_]
   Short_t         HKickTrackB_fData_tofHitInd[kMaxHKickTrackB_fData];
//[HKickTrackB.fData_]
   Short_t         HKickTrackB_fData_showerHitInd[kMaxHKickTrackB_fData];
//[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_p[kMaxHKickTrackB_fData];   //[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_beta[kMaxHKickTrackB_fData];   //[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_mass2[kMaxHKickTrackB_fData];   //[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_tof[kMaxHKickTrackB_fData];   //[HKickTrackB.fData_]
   Char_t          HKickTrackB_fData_polarity[kMaxHKickTrackB_fData];
//[HKickTrackB.fData_]
   Char_t          HKickTrackB_fData_sector[kMaxHKickTrackB_fData];
//[HKickTrackB.fData_]
   UInt_t          HKickTrackB_fData_cov_fUniqueID[kMaxHKickTrackB_fData];
//[HKickTrackB.fData_]
   UInt_t          HKickTrackB_fData_cov_fBits[kMaxHKickTrackB_fData];
//[HKickTrackB.fData_]
   Int_t           HKickTrackB_fData_cov_size[kMaxHKickTrackB_fData];
//[HKickTrackB.fData_]
   Int_t           HKickTrackB_fData_cov_dim[kMaxHKickTrackB_fData];
//[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_cov_data[kMaxHKickTrackB_fData][21];
//[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_metaEloss[kMaxHKickTrackB_fData];
//[HKickTrackB.fData_]
   Char_t          HKickTrackB_fData_quality[kMaxHKickTrackB_fData];
```

```
//[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_pTof[kMaxHKickTrackB_fData];   //[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_errPTof[kMaxHKickTrackB_fData];
//[HKickTrackB.fData_]
   Char_t          HKickTrackB_fData_pid[kMaxHKickTrackB_fData];   //[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_pull[kMaxHKickTrackB_fData];   //[HKickTrackB.fData_]
   Int_t           HKickTrackB_fData_outherHitId[kMaxHKickTrackB_fData];
//[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_showerSum10[kMaxHKickTrackB_fData];
//[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_showerSum20[kMaxHKickTrackB_fData];
//[HKickTrackB.fData_]
   Float_t         HKickTrackB_fData_flag[kMaxHKickTrackB_fData];   //[HKickTrackB.fData_]

   // List of branches

   TBranch        *b_HRtMdcTrk_HCategory_fUniqueID;   //!
   TBranch        *b_HRtMdcTrk_HCategory_fBits;   //!
   TBranch        *b_HRtMdcTrk_HCategory_fCat;   //!
   TBranch        *b_HRtMdcTrk_HCategory_fBranchingLevel;   //!
   TBranch        *b_HRtMdcTrk_fNDataObjs;   //!
   TBranch        *b_HRtMdcTrk_fData_;   //!
   TBranch        *b_HRtMdcTrk_fData_fUniqueID;   //!
   TBranch        *b_HRtMdcTrk_fData_fBits;   //!
   TBranch        *b_HRtMdcTrk_fData_z;   //!
   TBranch        *b_HRtMdcTrk_fData_r;   //!
   TBranch        *b_HRtMdcTrk_fData_p;   //!
   TBranch        *b_HRtMdcTrk_fData_theta;   //!
   TBranch        *b_HRtMdcTrk_fData_phi;   //!
   TBranch        *b_HRtMdcTrk_fData_charge;   //!
   TBranch        *b_HRtMdcTrk_fData_chi2;   //!
   TBranch        *b_HRtMdcTrk_fData_cov_fUniqueID;   //!
   TBranch        *b_HRtMdcTrk_fData_cov_fBits;   //!
   TBranch        *b_HRtMdcTrk_fData_cov_size;   //!
   TBranch        *b_HRtMdcTrk_fData_cov_dim;   //!
   TBranch        *b_HRtMdcTrk_fData_cov_data;   //!
   TBranch        *b_HRtMdcTrk_fData_flag;   //!
   TBranch        *b_HRtMdcTrk_fData_sector;   //!
   TBranch        *b_HRtMdcTrk_fData_segIndex;   //!
   TBranch        *b_HRtMdcTrk_fData_fOuterPos_fUniqueID;   //!
   TBranch        *b_HRtMdcTrk_fData_fOuterPos_fBits;   //!
   TBranch        *b_HRtMdcTrk_fData_fOuterPos_x;   //!
   TBranch        *b_HRtMdcTrk_fData_fOuterPos_y;   //!
   TBranch        *b_HRtMdcTrk_fData_fOuterPos_z;   //!
   TBranch        *b_HRtMdcTrk_fData_fOuterDir_fUniqueID;   //!
   TBranch        *b_HRtMdcTrk_fData_fOuterDir_fBits;   //!
   TBranch        *b_HRtMdcTrk_fData_fOuterDir_x;   //!
   TBranch        *b_HRtMdcTrk_fData_fOuterDir_y;   //!
   TBranch        *b_HRtMdcTrk_fData_fOuterDir_z;   //!
   TBranch        *b_HRtMdcTrk_fData_length;   //!
   TBranch        *b_HRtMdcTrk_fData_dKick;   //!
   TBranch        *b_HRtMdcTrk_fData_d;   //!
   TBranch        *b_HRtMdcTrk_fData_dPhi;   //!
   TBranch        *b_HRtMdcTrk_fData_leverArm;   //!
   TBranch        *b_HRtMdcTrk_fData_fitResult;   //!
   TBranch        *b_HKickTrack_HCategory_fUniqueID;   //!
   TBranch        *b_HKickTrack_HCategory_fBits;   //!
   TBranch        *b_HKickTrack_HCategory_fCat;   //!
   TBranch        *b_HKickTrack_HCategory_fBranchingLevel;   //!
   TBranch        *b_HKickTrack_fNDataObjs;   //!
   TBranch        *b_HKickTrack_fData_;   //!
   TBranch        *b_HKickTrack_fData_fUniqueID;   //!
```

```
TBranch        *b_HKickTrack_fData_fBits;   //!
TBranch        *b_HKickTrack_fData_pTof;   //!
TBranch        *b_HKickTrack_fData_errPTof;   //!
TBranch        *b_HKickTrack_fData_pid;   //!
TBranch        *b_HKickTrack_fData_quality;   //!
TBranch        *b_HKickTrack_fData_z;   //!
TBranch        *b_HKickTrack_fData_r;   //!
TBranch        *b_HKickTrack_fData_p;   //!
TBranch        *b_HKickTrack_fData_theta;   //!
TBranch        *b_HKickTrack_fData_phi;   //!
TBranch        *b_HKickTrack_fData_mass;   //!
TBranch        *b_HKickTrack_fData_charge;   //!
TBranch        *b_HKickTrack_fData_cov_fUniqueID;   //!
TBranch        *b_HKickTrack_fData_cov_fBits;   //!
TBranch        *b_HKickTrack_fData_cov_size;   //!
TBranch        *b_HKickTrack_fData_cov_dim;   //!
TBranch        *b_HKickTrack_fData_cov_data;   //!
TBranch        *b_HKickTrack_fData_sector;   //!
TBranch        *b_HKickTrack_fData_system;   //!
TBranch        *b_HKickTrack_fData_tof;   //!
TBranch        *b_HKickTrack_fData_metaeloss;   //!
TBranch        *b_HKickTrack_fData_beta;   //!
TBranch        *b_HKickTrack_fData_showerSum10;   //!
TBranch        *b_HKickTrack_fData_showerSum20;   //!
TBranch        *b_HKickTrack_fData_segmentId;   //!
TBranch        *b_HKickTrack_fData_ringId;   //!
TBranch        *b_HKickTrack_fData_outerHitId;   //!
TBranch        *b_HKickTrack_fData_pull;   //!
TBranch        *b_HKickTrack_fData_flag;   //!
TBranch        *b_HKickTrackB_HCategory_fUniqueID;   //!
TBranch        *b_HKickTrackB_HCategory_fBits;   //!
TBranch        *b_HKickTrackB_HCategory_fCat;   //!
TBranch        *b_HKickTrackB_HCategory_fBranchingLevel;   //!
TBranch        *b_HKickTrackB_fNDataObjs;   //!
TBranch        *b_HKickTrackB_fData_;   //!
TBranch        *b_HKickTrackB_fData_fUniqueID;   //!
TBranch        *b_HKickTrackB_fData_fBits;   //!
TBranch        *b_HKickTrackB_fData_z;   //!
TBranch        *b_HKickTrackB_fData_r;   //!
TBranch        *b_HKickTrackB_fData_theta;   //!
TBranch        *b_HKickTrackB_fData_phi;   //!
TBranch        *b_HKickTrackB_fData_tofHitInd;   //!
TBranch        *b_HKickTrackB_fData_showerHitInd;   //!
TBranch        *b_HKickTrackB_fData_p;   //!
TBranch        *b_HKickTrackB_fData_beta;   //!
TBranch        *b_HKickTrackB_fData_mass2;   //!
TBranch        *b_HKickTrackB_fData_tof;   //!
TBranch        *b_HKickTrackB_fData_polarity;   //!
TBranch        *b_HKickTrackB_fData_sector;   //!
TBranch        *b_HKickTrackB_fData_cov_fUniqueID;   //!
TBranch        *b_HKickTrackB_fData_cov_fBits;   //!
TBranch        *b_HKickTrackB_fData_cov_size;   //!
TBranch        *b_HKickTrackB_fData_cov_dim;   //!
TBranch        *b_HKickTrackB_fData_cov_data;   //!
TBranch        *b_HKickTrackB_fData_metaEloss;   //!
TBranch        *b_HKickTrackB_fData_quality;   //!
TBranch        *b_HKickTrackB_fData_pTof;   //!
TBranch        *b_HKickTrackB_fData_errPTof;   //!
TBranch        *b_HKickTrackB_fData_pid;   //!
TBranch        *b_HKickTrackB_fData_pull;   //!
TBranch        *b_HKickTrackB_fData_outherHitId;   //!
TBranch        *b_HKickTrackB_fData_showerSum10;   //!
```

```
   TBranch        *b_HKickTrackB_fData_showerSum20;   //!
   TBranch        *b_HKickTrackB_fData_flag;   //!

   prueba(TTree *tree=0);
   ~prueba();
   Int_t  Cut(Int_t entry);
   Int_t  GetEntry(Int_t entry);
   Int_t  LoadTree(Int_t entry);
   void   Init(TTree *tree);
   void   Loop();
   Bool_t Notify();
   void   Show(Int_t entry = -1);
};

#endif

#ifdef prueba_cxx
prueba::prueba(TTree *tree)
{
// if parameter tree is not specified (or zero), connect the file
// used to generate this class and read the Tree.
   if (tree == 0) {
      TFile *f = (TFile*)gROOT->GetListOfFiles()->FindObject("File_to_be_procesed.root");
      if (!f) {
         f = new TFile("File_to_be_procesed.root");
      }
      tree = (TTree*)gDirectory->Get("T");

   }
   Init(tree);
}

prueba::~prueba()
{
   if (!fChain) return;
   delete fChain->GetCurrentFile();
}

Int_t prueba::GetEntry(Int_t entry)
{
// Read contents of entry.
   if (!fChain) return 0;
   return fChain->GetEntry(entry);
}
Int_t prueba::LoadTree(Int_t entry)
{
// Set the environment to read one entry
   if (!fChain) return -5;
   Int_t centry = fChain->LoadTree(entry);
   if (centry < 0) return centry;
   if (fChain->IsA() != TChain::Class()) return centry;
   TChain *chain = (TChain*)fChain;
   if (chain->GetTreeNumber() != fCurrent) {
      fCurrent = chain->GetTreeNumber();
      Notify();
   }
   return centry;
}

void prueba::Init(TTree *tree)
{
```

```
   // Set branch addresses
   if (tree == 0) return;
   fChain = tree;
   fCurrent = -1;
   fChain->SetMakeClass(1);


fChain->SetBranchAddress("HRtMdcTrk.HCategory.fUniqueID",&HRtMdcTrk_HCategory_fUniqueID);
   fChain->SetBranchAddress("HRtMdcTrk.HCategory.fBits",&HRtMdcTrk_HCategory_fBits);
   fChain->SetBranchAddress("HRtMdcTrk.HCategory.fCat",&HRtMdcTrk_HCategory_fCat);

fChain->SetBranchAddress("HRtMdcTrk.HCategory.fBranchingLevel",&HRtMdcTrk_HCategory_fBranchingLevel);
   fChain->SetBranchAddress("HRtMdcTrk.fNDataObjs",&HRtMdcTrk_fNDataObjs);
   fChain->SetBranchAddress("HRtMdcTrk.fData",&HRtMdcTrk_fData_);
   fChain->SetBranchAddress("HRtMdcTrk.fData.fUniqueID",HRtMdcTrk_fData_fUniqueID);
   fChain->SetBranchAddress("HRtMdcTrk.fData.fBits",HRtMdcTrk_fData_fBits);
   fChain->SetBranchAddress("HRtMdcTrk.fData.z",HRtMdcTrk_fData_z);
   fChain->SetBranchAddress("HRtMdcTrk.fData.r",HRtMdcTrk_fData_r);
   fChain->SetBranchAddress("HRtMdcTrk.fData.p",HRtMdcTrk_fData_p);
   fChain->SetBranchAddress("HRtMdcTrk.fData.theta",HRtMdcTrk_fData_theta);
   fChain->SetBranchAddress("HRtMdcTrk.fData.phi",HRtMdcTrk_fData_phi);
   fChain->SetBranchAddress("HRtMdcTrk.fData.charge",HRtMdcTrk_fData_charge);
   fChain->SetBranchAddress("HRtMdcTrk.fData.chi2",HRtMdcTrk_fData_chi2);
   fChain->SetBranchAddress("HRtMdcTrk.fData.cov.fUniqueID",HRtMdcTrk_fData_cov_fUniqueID);
   fChain->SetBranchAddress("HRtMdcTrk.fData.cov.fBits",HRtMdcTrk_fData_cov_fBits);
   fChain->SetBranchAddress("HRtMdcTrk.fData.cov.size",HRtMdcTrk_fData_cov_size);
   fChain->SetBranchAddress("HRtMdcTrk.fData.cov.dim",HRtMdcTrk_fData_cov_dim);
   fChain->SetBranchAddress("HRtMdcTrk.fData.cov.data[15]",HRtMdcTrk_fData_cov_data);
   fChain->SetBranchAddress("HRtMdcTrk.fData.flag",HRtMdcTrk_fData_flag);
   fChain->SetBranchAddress("HRtMdcTrk.fData.sector",HRtMdcTrk_fData_sector);
   fChain->SetBranchAddress("HRtMdcTrk.fData.segIndex[2]",HRtMdcTrk_fData_segIndex);

fChain->SetBranchAddress("HRtMdcTrk.fData.fOuterPos.fUniqueID",HRtMdcTrk_fData_fOuterPos_fUniqueID);

fChain->SetBranchAddress("HRtMdcTrk.fData.fOuterPos.fBits",HRtMdcTrk_fData_fOuterPos_fBits);
   fChain->SetBranchAddress("HRtMdcTrk.fData.fOuterPos.x",HRtMdcTrk_fData_fOuterPos_x);
   fChain->SetBranchAddress("HRtMdcTrk.fData.fOuterPos.y",HRtMdcTrk_fData_fOuterPos_y);
   fChain->SetBranchAddress("HRtMdcTrk.fData.fOuterPos.z",HRtMdcTrk_fData_fOuterPos_z);

fChain->SetBranchAddress("HRtMdcTrk.fData.fOuterDir.fUniqueID",HRtMdcTrk_fData_fOuterDir_fUniqueID);

fChain->SetBranchAddress("HRtMdcTrk.fData.fOuterDir.fBits",HRtMdcTrk_fData_fOuterDir_fBits);
   fChain->SetBranchAddress("HRtMdcTrk.fData.fOuterDir.x",HRtMdcTrk_fData_fOuterDir_x);
   fChain->SetBranchAddress("HRtMdcTrk.fData.fOuterDir.y",HRtMdcTrk_fData_fOuterDir_y);
   fChain->SetBranchAddress("HRtMdcTrk.fData.fOuterDir.z",HRtMdcTrk_fData_fOuterDir_z);
   fChain->SetBranchAddress("HRtMdcTrk.fData.length",HRtMdcTrk_fData_length);
   fChain->SetBranchAddress("HRtMdcTrk.fData.dKick",HRtMdcTrk_fData_dKick);
   fChain->SetBranchAddress("HRtMdcTrk.fData.d",HRtMdcTrk_fData_d);
   fChain->SetBranchAddress("HRtMdcTrk.fData.dPhi",HRtMdcTrk_fData_dPhi);
   fChain->SetBranchAddress("HRtMdcTrk.fData.leverArm",HRtMdcTrk_fData_leverArm);
   fChain->SetBranchAddress("HRtMdcTrk.fData.fitResult",HRtMdcTrk_fData_fitResult);

fChain->SetBranchAddress("HKickTrack.HCategory.fUniqueID",&HKickTrack_HCategory_fUniqueID);
   fChain->SetBranchAddress("HKickTrack.HCategory.fBits",&HKickTrack_HCategory_fBits);
   fChain->SetBranchAddress("HKickTrack.HCategory.fCat",&HKickTrack_HCategory_fCat);

fChain->SetBranchAddress("HKickTrack.HCategory.fBranchingLevel",&HKickTrack_HCategory_fBranchingLevel);
   fChain->SetBranchAddress("HKickTrack.fNDataObjs",&HKickTrack_fNDataObjs);
   fChain->SetBranchAddress("HKickTrack.fData",&HKickTrack_fData_);
   fChain->SetBranchAddress("HKickTrack.fData.fUniqueID",HKickTrack_fData_fUniqueID);
   fChain->SetBranchAddress("HKickTrack.fData.fBits",HKickTrack_fData_fBits);
   fChain->SetBranchAddress("HKickTrack.fData.pTof",HKickTrack_fData_pTof);
```

```
   fChain->SetBranchAddress("HKickTrack.fData.errPTof",HKickTrack_fData_errPTof);
   fChain->SetBranchAddress("HKickTrack.fData.pid",HKickTrack_fData_pid);
   fChain->SetBranchAddress("HKickTrack.fData.quality",HKickTrack_fData_quality);
   fChain->SetBranchAddress("HKickTrack.fData.z",HKickTrack_fData_z);
   fChain->SetBranchAddress("HKickTrack.fData.r",HKickTrack_fData_r);
   fChain->SetBranchAddress("HKickTrack.fData.p",HKickTrack_fData_p);
   fChain->SetBranchAddress("HKickTrack.fData.theta",HKickTrack_fData_theta);
   fChain->SetBranchAddress("HKickTrack.fData.phi",HKickTrack_fData_phi);
   fChain->SetBranchAddress("HKickTrack.fData.mass",HKickTrack_fData_mass);
   fChain->SetBranchAddress("HKickTrack.fData.charge",HKickTrack_fData_charge);

fChain->SetBranchAddress("HKickTrack.fData.cov.fUniqueID",HKickTrack_fData_cov_fUniqueID);
   fChain->SetBranchAddress("HKickTrack.fData.cov.fBits",HKickTrack_fData_cov_fBits);
   fChain->SetBranchAddress("HKickTrack.fData.cov.size",HKickTrack_fData_cov_size);
   fChain->SetBranchAddress("HKickTrack.fData.cov.dim",HKickTrack_fData_cov_dim);
   fChain->SetBranchAddress("HKickTrack.fData.cov.data[21]",HKickTrack_fData_cov_data);
   fChain->SetBranchAddress("HKickTrack.fData.sector",HKickTrack_fData_sector);
   fChain->SetBranchAddress("HKickTrack.fData.system",HKickTrack_fData_system);
   fChain->SetBranchAddress("HKickTrack.fData.tof",HKickTrack_fData_tof);
   fChain->SetBranchAddress("HKickTrack.fData.metaeloss",HKickTrack_fData_metaeloss);
   fChain->SetBranchAddress("HKickTrack.fData.beta",HKickTrack_fData_beta);
   fChain->SetBranchAddress("HKickTrack.fData.showerSum10",HKickTrack_fData_showerSum10);
   fChain->SetBranchAddress("HKickTrack.fData.showerSum20",HKickTrack_fData_showerSum20);
   fChain->SetBranchAddress("HKickTrack.fData.segmentId",HKickTrack_fData_segmentId);
   fChain->SetBranchAddress("HKickTrack.fData.ringId",HKickTrack_fData_ringId);
   fChain->SetBranchAddress("HKickTrack.fData.outerHitId",HKickTrack_fData_outerHitId);
   fChain->SetBranchAddress("HKickTrack.fData.pull",HKickTrack_fData_pull);
   fChain->SetBranchAddress("HKickTrack.fData.flag",HKickTrack_fData_flag);

fChain->SetBranchAddress("HKickTrackB.HCategory.fUniqueID",&HKickTrackB_HCategory_fUniqueID);
   fChain->SetBranchAddress("HKickTrackB.HCategory.fBits",&HKickTrackB_HCategory_fBits);
   fChain->SetBranchAddress("HKickTrackB.HCategory.fCat",&HKickTrackB_HCategory_fCat);

fChain->SetBranchAddress("HKickTrackB.HCategory.fBranchingLevel",&HKickTrackB_HCategory_fBranchingLevel);
   fChain->SetBranchAddress("HKickTrackB.fNDataObjs",&HKickTrackB_fNDataObjs);
   fChain->SetBranchAddress("HKickTrackB.fData",&HKickTrackB_fData_);
   fChain->SetBranchAddress("HKickTrackB.fData.fUniqueID",HKickTrackB_fData_fUniqueID);
   fChain->SetBranchAddress("HKickTrackB.fData.fBits",HKickTrackB_fData_fBits);
   fChain->SetBranchAddress("HKickTrackB.fData.z",HKickTrackB_fData_z);
   fChain->SetBranchAddress("HKickTrackB.fData.r",HKickTrackB_fData_r);
   fChain->SetBranchAddress("HKickTrackB.fData.theta",HKickTrackB_fData_theta);
   fChain->SetBranchAddress("HKickTrackB.fData.phi",HKickTrackB_fData_phi);
   fChain->SetBranchAddress("HKickTrackB.fData.tofHitInd",HKickTrackB_fData_tofHitInd);

fChain->SetBranchAddress("HKickTrackB.fData.showerHitInd",HKickTrackB_fData_showerHitInd);
   fChain->SetBranchAddress("HKickTrackB.fData.p",HKickTrackB_fData_p);
   fChain->SetBranchAddress("HKickTrackB.fData.beta",HKickTrackB_fData_beta);
   fChain->SetBranchAddress("HKickTrackB.fData.mass2",HKickTrackB_fData_mass2);
   fChain->SetBranchAddress("HKickTrackB.fData.tof",HKickTrackB_fData_tof);
   fChain->SetBranchAddress("HKickTrackB.fData.polarity",HKickTrackB_fData_polarity);
   fChain->SetBranchAddress("HKickTrackB.fData.sector",HKickTrackB_fData_sector);

fChain->SetBranchAddress("HKickTrackB.fData.cov.fUniqueID",HKickTrackB_fData_cov_fUniqueID);
   fChain->SetBranchAddress("HKickTrackB.fData.cov.fBits",HKickTrackB_fData_cov_fBits);
   fChain->SetBranchAddress("HKickTrackB.fData.cov.size",HKickTrackB_fData_cov_size);
   fChain->SetBranchAddress("HKickTrackB.fData.cov.dim",HKickTrackB_fData_cov_dim);
   fChain->SetBranchAddress("HKickTrackB.fData.cov.data[21]",HKickTrackB_fData_cov_data);
   fChain->SetBranchAddress("HKickTrackB.fData.metaEloss",HKickTrackB_fData_metaEloss);
   fChain->SetBranchAddress("HKickTrackB.fData.quality",HKickTrackB_fData_quality);
   fChain->SetBranchAddress("HKickTrackB.fData.pTof",HKickTrackB_fData_pTof);
   fChain->SetBranchAddress("HKickTrackB.fData.errPTof",HKickTrackB_fData_errPTof);
   fChain->SetBranchAddress("HKickTrackB.fData.pid",HKickTrackB_fData_pid);
```

```
   fChain->SetBranchAddress("HKickTrackB.fData.pull",HKickTrackB_fData_pull);
   fChain->SetBranchAddress("HKickTrackB.fData.outherHitId",HKickTrackB_fData_outherHitId);
   fChain->SetBranchAddress("HKickTrackB.fData.showerSum10",HKickTrackB_fData_showerSum10);
   fChain->SetBranchAddress("HKickTrackB.fData.showerSum20",HKickTrackB_fData_showerSum20);
   fChain->SetBranchAddress("HKickTrackB.fData.flag",HKickTrackB_fData_flag);
   Notify();
}

Bool_t prueba::Notify()
{
   // The Notify() function is called when a new file is opened.

   // Get branch pointers
   b_HRtMdcTrk_HCategory_fUniqueID = fChain->GetBranch("HRtMdcTrk.HCategory.fUniqueID");
   b_HRtMdcTrk_HCategory_fBits = fChain->GetBranch("HRtMdcTrk.HCategory.fBits");
   b_HRtMdcTrk_HCategory_fCat = fChain->GetBranch("HRtMdcTrk.HCategory.fCat");
   b_HRtMdcTrk_HCategory_fBranchingLevel =
fChain->GetBranch("HRtMdcTrk.HCategory.fBranchingLevel");
   b_HRtMdcTrk_fNDataObjs = fChain->GetBranch("HRtMdcTrk.fNDataObjs");
   b_HRtMdcTrk_fData_ = fChain->GetBranch("HRtMdcTrk.fData");
   b_HRtMdcTrk_fData_fUniqueID = fChain->GetBranch("HRtMdcTrk.fData.fUniqueID");
   b_HRtMdcTrk_fData_fBits = fChain->GetBranch("HRtMdcTrk.fData.fBits");
   b_HRtMdcTrk_fData_z = fChain->GetBranch("HRtMdcTrk.fData.z");
   b_HRtMdcTrk_fData_r = fChain->GetBranch("HRtMdcTrk.fData.r");
   b_HRtMdcTrk_fData_p = fChain->GetBranch("HRtMdcTrk.fData.p");
   b_HRtMdcTrk_fData_theta = fChain->GetBranch("HRtMdcTrk.fData.theta");
   b_HRtMdcTrk_fData_phi = fChain->GetBranch("HRtMdcTrk.fData.phi");
   b_HRtMdcTrk_fData_charge = fChain->GetBranch("HRtMdcTrk.fData.charge");
   b_HRtMdcTrk_fData_chi2 = fChain->GetBranch("HRtMdcTrk.fData.chi2");
   b_HRtMdcTrk_fData_cov_fUniqueID = fChain->GetBranch("HRtMdcTrk.fData.cov.fUniqueID");
   b_HRtMdcTrk_fData_cov_fBits = fChain->GetBranch("HRtMdcTrk.fData.cov.fBits");
   b_HRtMdcTrk_fData_cov_size = fChain->GetBranch("HRtMdcTrk.fData.cov.size");
   b_HRtMdcTrk_fData_cov_dim = fChain->GetBranch("HRtMdcTrk.fData.cov.dim");
   b_HRtMdcTrk_fData_cov_data = fChain->GetBranch("HRtMdcTrk.fData.cov.data[15]");
   b_HRtMdcTrk_fData_flag = fChain->GetBranch("HRtMdcTrk.fData.flag");
   b_HRtMdcTrk_fData_sector = fChain->GetBranch("HRtMdcTrk.fData.sector");
   b_HRtMdcTrk_fData_segIndex = fChain->GetBranch("HRtMdcTrk.fData.segIndex[2]");
   b_HRtMdcTrk_fData_fOuterPos_fUniqueID =
fChain->GetBranch("HRtMdcTrk.fData.fOuterPos.fUniqueID");
   b_HRtMdcTrk_fData_fOuterPos_fBits =
fChain->GetBranch("HRtMdcTrk.fData.fOuterPos.fBits");
   b_HRtMdcTrk_fData_fOuterPos_x = fChain->GetBranch("HRtMdcTrk.fData.fOuterPos.x");
   b_HRtMdcTrk_fData_fOuterPos_y = fChain->GetBranch("HRtMdcTrk.fData.fOuterPos.y");
   b_HRtMdcTrk_fData_fOuterPos_z = fChain->GetBranch("HRtMdcTrk.fData.fOuterPos.z");
   b_HRtMdcTrk_fData_fOuterDir_fUniqueID =
fChain->GetBranch("HRtMdcTrk.fData.fOuterDir.fUniqueID");
   b_HRtMdcTrk_fData_fOuterDir_fBits =
fChain->GetBranch("HRtMdcTrk.fData.fOuterDir.fBits");
   b_HRtMdcTrk_fData_fOuterDir_x = fChain->GetBranch("HRtMdcTrk.fData.fOuterDir.x");
   b_HRtMdcTrk_fData_fOuterDir_y = fChain->GetBranch("HRtMdcTrk.fData.fOuterDir.y");
   b_HRtMdcTrk_fData_fOuterDir_z = fChain->GetBranch("HRtMdcTrk.fData.fOuterDir.z");
   b_HRtMdcTrk_fData_length = fChain->GetBranch("HRtMdcTrk.fData.length");
   b_HRtMdcTrk_fData_dKick = fChain->GetBranch("HRtMdcTrk.fData.dKick");
   b_HRtMdcTrk_fData_d = fChain->GetBranch("HRtMdcTrk.fData.d");
   b_HRtMdcTrk_fData_dPhi = fChain->GetBranch("HRtMdcTrk.fData.dPhi");
   b_HRtMdcTrk_fData_leverArm = fChain->GetBranch("HRtMdcTrk.fData.leverArm");
   b_HRtMdcTrk_fData_fitResult = fChain->GetBranch("HRtMdcTrk.fData.fitResult");
   b_HKickTrack_HCategory_fUniqueID = fChain->GetBranch("HKickTrack.HCategory.fUniqueID");
   b_HKickTrack_HCategory_fBits = fChain->GetBranch("HKickTrack.HCategory.fBits");
   b_HKickTrack_HCategory_fCat = fChain->GetBranch("HKickTrack.HCategory.fCat");
   b_HKickTrack_HCategory_fBranchingLevel =
fChain->GetBranch("HKickTrack.HCategory.fBranchingLevel");
```

```
   b_HKickTrack_fNDataObjs = fChain->GetBranch("HKickTrack.fNDataObjs");
   b_HKickTrack_fData_ = fChain->GetBranch("HKickTrack.fData");
   b_HKickTrack_fData_fUniqueID = fChain->GetBranch("HKickTrack.fData.fUniqueID");
   b_HKickTrack_fData_fBits = fChain->GetBranch("HKickTrack.fData.fBits");
   b_HKickTrack_fData_pTof = fChain->GetBranch("HKickTrack.fData.pTof");
   b_HKickTrack_fData_errPTof = fChain->GetBranch("HKickTrack.fData.errPTof");
   b_HKickTrack_fData_pid = fChain->GetBranch("HKickTrack.fData.pid");
   b_HKickTrack_fData_quality = fChain->GetBranch("HKickTrack.fData.quality");
   b_HKickTrack_fData_z = fChain->GetBranch("HKickTrack.fData.z");
   b_HKickTrack_fData_r = fChain->GetBranch("HKickTrack.fData.r");
   b_HKickTrack_fData_p = fChain->GetBranch("HKickTrack.fData.p");
   b_HKickTrack_fData_theta = fChain->GetBranch("HKickTrack.fData.theta");
   b_HKickTrack_fData_phi = fChain->GetBranch("HKickTrack.fData.phi");
   b_HKickTrack_fData_mass = fChain->GetBranch("HKickTrack.fData.mass");
   b_HKickTrack_fData_charge = fChain->GetBranch("HKickTrack.fData.charge");
   b_HKickTrack_fData_cov_fUniqueID = fChain->GetBranch("HKickTrack.fData.cov.fUniqueID");
   b_HKickTrack_fData_cov_fBits = fChain->GetBranch("HKickTrack.fData.cov.fBits");
   b_HKickTrack_fData_cov_size = fChain->GetBranch("HKickTrack.fData.cov.size");
   b_HKickTrack_fData_cov_dim = fChain->GetBranch("HKickTrack.fData.cov.dim");
   b_HKickTrack_fData_cov_data = fChain->GetBranch("HKickTrack.fData.cov.data[21]");
   b_HKickTrack_fData_sector = fChain->GetBranch("HKickTrack.fData.sector");
   b_HKickTrack_fData_system = fChain->GetBranch("HKickTrack.fData.system");
   b_HKickTrack_fData_tof = fChain->GetBranch("HKickTrack.fData.tof");
   b_HKickTrack_fData_metaeloss = fChain->GetBranch("HKickTrack.fData.metaeloss");
   b_HKickTrack_fData_beta = fChain->GetBranch("HKickTrack.fData.beta");
   b_HKickTrack_fData_showerSum10 = fChain->GetBranch("HKickTrack.fData.showerSum10");
   b_HKickTrack_fData_showerSum20 = fChain->GetBranch("HKickTrack.fData.showerSum20");
   b_HKickTrack_fData_segmentId = fChain->GetBranch("HKickTrack.fData.segmentId");
   b_HKickTrack_fData_ringId = fChain->GetBranch("HKickTrack.fData.ringId");
   b_HKickTrack_fData_outerHitId = fChain->GetBranch("HKickTrack.fData.outerHitId");
   b_HKickTrack_fData_pull = fChain->GetBranch("HKickTrack.fData.pull");
   b_HKickTrack_fData_flag = fChain->GetBranch("HKickTrack.fData.flag");
   b_HKickTrackB_HCategory_fUniqueID =
fChain->GetBranch("HKickTrackB.HCategory.fUniqueID");
   b_HKickTrackB_HCategory_fBits = fChain->GetBranch("HKickTrackB.HCategory.fBits");
   b_HKickTrackB_HCategory_fCat = fChain->GetBranch("HKickTrackB.HCategory.fCat");
   b_HKickTrackB_HCategory_fBranchingLevel =
fChain->GetBranch("HKickTrackB.HCategory.fBranchingLevel");
   b_HKickTrackB_fNDataObjs = fChain->GetBranch("HKickTrackB.fNDataObjs");
   b_HKickTrackB_fData_ = fChain->GetBranch("HKickTrackB.fData");
   b_HKickTrackB_fData_fUniqueID = fChain->GetBranch("HKickTrackB.fData.fUniqueID");
   b_HKickTrackB_fData_fBits = fChain->GetBranch("HKickTrackB.fData.fBits");
   b_HKickTrackB_fData_z = fChain->GetBranch("HKickTrackB.fData.z");
   b_HKickTrackB_fData_r = fChain->GetBranch("HKickTrackB.fData.r");
   b_HKickTrackB_fData_theta = fChain->GetBranch("HKickTrackB.fData.theta");
   b_HKickTrackB_fData_phi = fChain->GetBranch("HKickTrackB.fData.phi");
   b_HKickTrackB_fData_tofHitInd = fChain->GetBranch("HKickTrackB.fData.tofHitInd");
   b_HKickTrackB_fData_showerHitInd = fChain->GetBranch("HKickTrackB.fData.showerHitInd");
   b_HKickTrackB_fData_p = fChain->GetBranch("HKickTrackB.fData.p");
   b_HKickTrackB_fData_beta = fChain->GetBranch("HKickTrackB.fData.beta");
   b_HKickTrackB_fData_mass2 = fChain->GetBranch("HKickTrackB.fData.mass2");
   b_HKickTrackB_fData_tof = fChain->GetBranch("HKickTrackB.fData.tof");
   b_HKickTrackB_fData_polarity = fChain->GetBranch("HKickTrackB.fData.polarity");
   b_HKickTrackB_fData_sector = fChain->GetBranch("HKickTrackB.fData.sector");
   b_HKickTrackB_fData_cov_fUniqueID =
fChain->GetBranch("HKickTrackB.fData.cov.fUniqueID");
   b_HKickTrackB_fData_cov_fBits = fChain->GetBranch("HKickTrackB.fData.cov.fBits");
   b_HKickTrackB_fData_cov_size = fChain->GetBranch("HKickTrackB.fData.cov.size");
   b_HKickTrackB_fData_cov_dim = fChain->GetBranch("HKickTrackB.fData.cov.dim");
   b_HKickTrackB_fData_cov_data = fChain->GetBranch("HKickTrackB.fData.cov.data[21]");
   b_HKickTrackB_fData_metaEloss = fChain->GetBranch("HKickTrackB.fData.metaEloss");
   b_HKickTrackB_fData_quality = fChain->GetBranch("HKickTrackB.fData.quality");
```

```
   b_HKickTrackB_fData_pTof = fChain->GetBranch("HKickTrackB.fData.pTof");
   b_HKickTrackB_fData_errPTof = fChain->GetBranch("HKickTrackB.fData.errPTof");
   b_HKickTrackB_fData_pid = fChain->GetBranch("HKickTrackB.fData.pid");
   b_HKickTrackB_fData_pull = fChain->GetBranch("HKickTrackB.fData.pull");
   b_HKickTrackB_fData_outherHitId = fChain->GetBranch("HKickTrackB.fData.outherHitId");
   b_HKickTrackB_fData_showerSum10 = fChain->GetBranch("HKickTrackB.fData.showerSum10");
   b_HKickTrackB_fData_showerSum20 = fChain->GetBranch("HKickTrackB.fData.showerSum20");
   b_HKickTrackB_fData_flag = fChain->GetBranch("HKickTrackB.fData.flag");

   return kTRUE;
}

void prueba::Show(Int_t entry)
{
// Print contents of entry.
// If entry is not specified, print current entry
   if (!fChain) return;
   fChain->Show(entry);
}
Int_t prueba::Cut(Int_t entry)
{
// This function may be called from Loop.
// returns  1 if entry is accepted.
// returns -1 otherwise.
   return 1;
}
#endif // #ifdef prueba_cxx
```

C.2 Main file

```
#define prueba_cxx
#include "prueba.h"
#include <TH2.h>
#include <TStyle.h>
#include <TCanvas.h>
#include <TCutG.h>
#include <TMath.h>
#include <TVector3.h>
#include <TLorentzVector.h>

void prueba::Loop()
{

   if (fChain == 0) return;

 // Defining histograms
   TH2F *Momentum   = new TH2F("P vs #theta","Momentum vs #theta",220,100,4000,150,10,65);
   TH2F *Momentumk   = new TH2F("P (kick) vs #theta","Momentum vs
#theta",220,100,4000,150,10,65);
   TH2F *MomTeo   = new TH2F("Pteo","Theoretical momenta",220,100,4000,150,10,65);
   TH2F *MomTeok   = new TH2F("Pteo (kick)","Theoretical momenta",220,100,4000,150,10,65);
   TH2F *histP1P2   = new TH2F("P1 vs P2","P1 (MeV) vs P2 (MeV)",150,0,4000,150,0,4000);
   TH2F *Thetas12all  = new TH2F("All Tracks","#theta_{1} vs #theta_{2},
all",200,10,75,200,10,75);
   TH2F *Thetas12elastic  = new TH2F("Elastic pairs","#theta_{1} vs #theta_{2}, elastic
pairs",200,10,75,200,10,75);
```

```
   TH2F *T1plusT2 = new TH2F("#theta_{1} + #theta_{2} vs #theta_{1}","#theta_{1} + #theta_{2}
vs #theta_{1}",200,5,75,200,50,90);
   TH1F *Sector1  = new TH1F("Sector 1","Sector 1",50,-1,6);
   TH1F *Sector2  = new TH1F("Sector 2","Sector 2",50,-1,6);
   TH1F *histPhi1  = new TH1F("phi1","phi1",150,0,360);
   TH1F *histPhi2  = new TH1F("phi2","phi2",150,0,360);
   TH1F *cop  = new TH1F("cop","Coplanar cut",200,170,190);
   TH1F *missmassAll = new TH1F("missmassAll","All Missing Mass",150,-.05,.05);
   TH1F *missmassVeto = new TH1F("missmassVeto","Missing Mass with veto on pp
pair",150,-.05,.05);
   TH1F *missmass = new TH1F("missmass","Missing Mass of elastic pair",150,-.05,.05);
   TH1F *gamma = new TH1F("1/#gamma_{CM}^{2}","gamma",150,0.4,0.52);
   TH1F *invmass = new TH1F("InvMass","Invariant Mass",150,1900,3600);
   TH1F *DeltaP = new TH1F("#DeltaP/P","#DeltaP/P",150,-1,1);
   TH2F *DeltaPvsP = new TH2F("#DeltaP/P vs P","#DeltaP/P vs
Theta",200,100,4200,200,-1.2,1);
   TH2F *DeltaPvsTheta = new TH2F("#DeltaP/P vs Theta","#DeltaP/P vs
Theta",200,10,65,200,-1,1);

 // Defining variables
   Float_t deg2rad=180/TMath::Pi();
   Int_t sector1, sector2, sector1k, sector2k;
   Double_t theta1, theta2, phi1, phi2;
   Float_t T0=2200., T1, T2, pteo, p1, p2, Dp1, Dp2;
   Float_t InvMass, MissMass2, m_prot=.938272, m_protMeV=938.272;
   Double_t pbeam=2.994, pbeamMeV=2994;
   Float_t pteo1, pteo2;
   Int_t charge1, charge2, charge1k, charge2k;
   Float_t p1k, p2k, pteo1k, pteo2k, theta1k, theta2k, T1k, T2k;  // KickPlane

 // Defining some 3D-vectors and Lorentz-vectors
   TVector3 P1, P2;
   TLorentzVector sum_f;
   TVector3 Pbeam(0.,0.,pbeamMeV);
   TLorentzVector lva(Pbeam,sqrt(Pbeam*Pbeam+m_protMeV*m_protMeV));
   TLorentzVector lvb(0.,0.,0.,m_protMeV);
   TLorentzVector sum_i = lva + lvb;
   TLorentzVector diff;

   TFile *fcut = new TFile("micorte.root");    // Getting the graphical cut from a root file
   TCutG *cutg = (TCutG *)fcut->Get("CUTG");   // not used up to now


 // Main loop
   Int_t nentries = Int_t(fChain->GetEntriesFast());
   Int_t nbytes = 0, nb = 0;
   for (Int_t jentry=0; jentry<nentries;jentry++) {
      Int_t ientry = LoadTree(jentry);
      if (ientry < 0) break;
      nb = b_HRtMdcTrk_fData_->GetEntry(ientry);   nbytes += nb;  // Getting entries
      b_HRtMdcTrk_fData_->GetEntry(ientry);                       // of needed branches
      b_HRtMdcTrk_fNDataObjs->GetEntry(ientry);  // Reft Tracks

      b_HKickTrack_fData_->GetEntry(ientry);    // KickPlane
      b_HKickTrack_fNDataObjs->GetEntry(ientry);


 // Calculating from Reference Trayectories
      if (HRtMdcTrk_fNDataObjs) {

      // Getting momemtum of both tracks from the Reference Trayectories output
  p1 = HRtMdcTrk_fData_p[0];  // (MeV)
```

```
        p1 = p1 / .9232;                                          // magnetic field
correction
        p1 = p1 + 5.6275 + TMath::Exp( 5.81456 - 0.006394*p1 );  // eloss correction
  p2 = HRtMdcTrk_fData_p[1];  // (MeV)
        p2 = p2 / .9232;                                          // magnetic field
correction
        p2 = p2 + 5.6275 + TMath::Exp( 5.81456 - 0.006394*p2 );  // eloss correction

  theta1 = HRtMdcTrk_fData_theta[0];   // rads
  theta2 = HRtMdcTrk_fData_theta[1];   // rads

  phi1 = HRtMdcTrk_fData_phi[0];   // rads
  phi2 = HRtMdcTrk_fData_phi[1];   // rads

        sector1 = HRtMdcTrk_fData_sector[0];
        sector2 = HRtMdcTrk_fData_sector[1];

      // Conversion of Phi azimuthal angle to LAB system
        if (sector1 == 0)  phi1 = phi1;
        if (sector2 == 0)  phi2 = phi2;
        if (sector1 == 1)  phi1 = phi1 + 60/deg2rad;
        if (sector2 == 1)  phi2 = phi2 + 60/deg2rad;
        if (sector1 == 2)  phi1 = phi1 + 120/deg2rad;
        if (sector2 == 2)  phi2 = phi2 + 120/deg2rad;
        if (sector1 == 3)  phi1 = phi1 + 180/deg2rad;
        if (sector2 == 3)  phi2 = phi2 + 180/deg2rad;
        if (sector1 == 4)  phi1 = phi1 + 240/deg2rad;
        if (sector2 == 4)  phi2 = phi2 + 240/deg2rad;
        if (sector1 == 5)  phi1 = phi1 - 60/deg2rad;
        if (sector2 == 5)  phi2 = phi2 - 60/deg2rad;

        cop->Fill(abs(phi1*deg2rad-phi2*deg2rad));    // Coplanarity

      // Calculating Invariant Mass of pp pair and pp-Missing Mass
        P1.SetMagThetaPhi(p1,theta1,phi1);
        P2.SetMagThetaPhi(p2,theta2,phi2);

        TLorentzVector lv1(P1,sqrt(P1*P1+m_protMeV*m_protMeV));    // Cuadrimomentum vector
for proton one
        TLorentzVector lv2(P2,sqrt(P2*P2+m_protMeV*m_protMeV));    // Cuadrimomentum vector
for proton two

        sum_f = lv1 + lv2;      // Cuadrimomentum vector of pp pair

        diff = sum_i - sum_f;   // Initial Cuadrimomentum vector minus Cuadrimomentum vector
of pp pair

        MissMass2 = diff.M2();  // Missing Mass (MeV^2/c^4)
        missmassAll->Fill(MissMass2/1000000);    // (GeV^2/c^4)

        charge1 = HRtMdcTrk_fData_charge[0];
        charge2 = HRtMdcTrk_fData_charge[1];

        Thetas12all->Fill(theta1*deg2rad,theta2*deg2rad);


        if (HRtMdcTrk_fNDataObjs == 2) {      // selecting objects with two tracks per event

   if (charge1>0 && charge2>0) {         // both positive tracks

          if (abs(sector1-sector2)==3) {     // both tracks hitting in opposite sectors
```

```
     histPhi1->Fill(phi1*deg2rad);
     histPhi2->Fill(phi2*deg2rad);
           gamma->Fill(TMath::Tan(theta1)*TMath::Tan(theta2));

     if (TMath::Tan(theta1)*TMath::Tan(theta2)>.4525 &&
TMath::Tan(theta1)*TMath::Tan(theta2)<.4725) {   // Cut in gamma value

            Momentum->Fill(p1,theta1*deg2rad);
            Momentum->Fill(p2,theta2*deg2rad);

      T1 = T0 * (2. * TMath::Cos(theta1) * TMath::Cos(theta1)) / (4.34 - 2.34 *
TMath::Cos(theta1) * TMath::Cos(theta1));   // Theoretical value for first proton's kinetic
energy
      T2 = T0 * (2. * TMath::Cos(theta2) * TMath::Cos(theta2)) / (4.34 - 2.34 *
TMath::Cos(theta2) * TMath::Cos(theta2));   // Theoretical value for second proton's kinetic
energy

      pteo1 = sqrt( T1 * T1 + 2. * m_prot * 1000 * T1);   // Theoretical value for first
proton's momentum (MeV)
      pteo2 = sqrt( T2 * T2 + 2. * m_prot * 1000 * T2);   // Theoretical value for first
proton's momentum (MeV)

            MomTeo->Fill(pteo1,theta1*deg2rad);
            Thetas12elastic->Fill(theta1*deg2rad,theta2*deg2rad);
            T1plusT2->Fill(theta1*deg2rad,theta1*deg2rad+theta2*deg2rad);
            T1plusT2->Fill(theta2*deg2rad,theta1*deg2rad+theta2*deg2rad);

      histP1P2->Fill(p1,p2);

            missmass->Fill(MissMass2/1000000);    // (GeV^2/c^4)

            InvMass = sum_f.M2();   // Invariant mass of the pair (MeV^2/c^4)
            invmass->Fill(sqrt(InvMass));   // (MeV/c^2)

      Sector1->Fill(sector1);
      Sector2->Fill(sector2);

            Dp1 = pteo1 * ((1/pteo1)-(1/p1));   // Calculating momentum resolution
            Dp2 = pteo2 * ((1/pteo2)-(1/p2));
            DeltaP->Fill(Dp1);
            DeltaP->Fill(Dp2);
            DeltaPvsP->Fill(p1,Dp1);
            DeltaPvsP->Fill(p2,Dp2);
            DeltaPvsTheta->Fill(theta1*deg2rad,Dp1);
            DeltaPvsTheta->Fill(theta2*deg2rad,Dp2);

      //  Testing values "on-line"
         if(jentry%100==0) {
           cout << "j = " << jentry << endl;
           cout << "p1 = " << p1 << "  p2 = " << p2 << " MeV" << endl;
           cout << "MissingMass = " << MissMass2/1000000 << " GeV^2/c^4" << endl;
           cout << "InvariantMass = " << sqrt(InvMass) << " MeV/c^2" << endl << endl; }

           }   // gamma cut

          } // opposite sectors

      // if (Cut(ientry) < 0) continue;

         }  // positive tracks

        }  // two tracks
```

```
       else  missmassVeto->Fill(MissMass2/1000000);   // veto on elastic pair; searching
pi0?

     }  // Track objects
 // End of Reference Trayectories


 // Calculating from KickPlane
     if (HKickTrack_fNDataObjs == 2) {    // Two tracks per event

        p1k = HKickTrack_fData_p[0];  // (MeV)
        p1k = p1k + 5.6275 + TMath::Exp( 5.81456 - 0.006394*p1k );  // eloss correction
  p2k = HKickTrack_fData_p[1];  // (MeV)
        p2k = p2k + 5.6275 + TMath::Exp( 5.81456 - 0.006394*p2k );  // eloss correction

        charge1k = HKickTrack_fData_charge[0];
        charge2k = HKickTrack_fData_charge[1];

        sector1k = HKickTrack_fData_sector[0];
        sector2k = HKickTrack_fData_sector[1];

        theta1k = HKickTrack_fData_theta[0];   // rads
  theta2k = HKickTrack_fData_theta[1];   // rads

        if (charge1k>0 && charge2k>0) {         // both positive tracks

          if (abs(sector1k-sector2k)==3) {     // both tracks hitting in opposite sectors

      if (TMath::Tan(theta1k)*TMath::Tan(theta2k)>.4525 &&
TMath::Tan(theta1k)*TMath::Tan(theta2k)<.4725) {   // Cut in gamma value

             T1k = T0 * (2. * TMath::Cos(theta1k) * TMath::Cos(theta1k)) / (4.34 - 2.34 *
TMath::Cos(theta1k) * TMath::Cos(theta1k));   // Theoretical value for first proton's
kinetic energy
       T2k = T0 * (2. * TMath::Cos(theta2k) * TMath::Cos(theta2k)) / (4.34 - 2.34 *
TMath::Cos(theta2k) * TMath::Cos(theta2k));   // Theoretical value for second proton's
kinetic energy

       pteo1k = sqrt( T1k * T1k + 2. * m_prot * 1000 * T1k);   // Theoretical value for
first proton's momentum (MeV)
       pteo2k = sqrt( T2k * T2k + 2. * m_prot * 1000 * T2k);   // Theoretical value for
first proton's momentum (MeV)
             Momentumk->Fill(p1k,theta1k*deg2rad);
             Momentumk->Fill(p2k,theta2k*deg2rad);

             MomTeok->Fill(pteo1k,theta1k*deg2rad);

           }  // gamma cut
         }  // opposite sectors
       }  // positive tracks
     }  // Two tracks (KickPlane)
  // End of KickPlane

   }  // main loop


 // Drawing histograms

  TCanvas *THE12all = new TCanvas();
  Thetas12all->Draw("colz");
  Thetas12all->GetXaxis()->SetTitle("#theta_{1} (Deg)");
  Thetas12all->GetYaxis()->SetTitle("#theta_{2} (Deg)");
```

```
TCanvas *PHI1 = new TCanvas();
histPhi1->Draw();

TCanvas *PHI2 = new TCanvas();
histPhi2->Draw();

TCanvas *MISSMASS = new TCanvas();
missmassAll->Draw();
missmass->SetLineColor(kBlue);
missmass->Draw();
missmass->SetStats(kFALSE);
missmass->SetTitle("Missing Mass of elastic pair");
missmass->GetXaxis()->SetTitle("M_{miss}^{pp}^{2} (GeV^{2}/c^{4})");
missmass->GetYaxis()->SetTitle("Entries");
missmassVeto->SetLineColor(kRed);
missmassVeto->Draw("same");

TCanvas *P1P2 = new TCanvas();
histP1P2->Draw("colz");
histP1P2->GetXaxis()->SetTitle("P1 (MeV/c)");
histP1P2->GetYaxis()->SetTitle("P2 (MeV/c)");
histP1P2->SetTitle("");

TCanvas *THE12el = new TCanvas();
Thetas12elastic->Draw("colz");
Thetas12elastic->GetXaxis()->SetTitle("#theta_{1} (Deg)");
Thetas12elastic->GetYaxis()->SetTitle("#theta_{2} (Deg)");

TCanvas *COP = new TCanvas();
cop->Draw();

TCanvas *MAIN = new TCanvas();
Momentum->Draw("colz");
Momentum->SetStats(kFALSE);
Momentum->SetTitle("Calculated and Theoretical momenta vs #theta");
Momentum->GetXaxis()->SetTitle("P (MeV/c)");
Momentum->GetYaxis()->SetTitle("#theta (Deg)");
MomTeo->SetMarkerColor(kRed);
MomTeo->Draw("same");

TCanvas *mini1 = new TCanvas();
Momentum->Draw("colz");
Momentum->SetStats(kFALSE);
Momentum->SetTitle("Calculated and Theoretical momenta vs #theta");
Momentum->GetXaxis()->SetTitle("P (MeV/c)");
Momentum->GetYaxis()->SetTitle("#theta (Deg)");

TCanvas *mini2 = new TCanvas();
MomTeo->SetMarkerColor(kRed);
MomTeo->Draw();

TCanvas *t1PLUSt2 = new TCanvas();
T1plusT2->Draw("colz");
T1plusT2->SetTitle("#theta_{1} + #theta_{2} for elastic pairs");
T1plusT2->GetXaxis()->SetTitle("#theta_{1} (Deg)");
T1plusT2->GetYaxis()->SetTitle("#theta_{1} + #theta_{2} (Deg)");

TCanvas *MAINkickplane = new TCanvas();
Momentumk->Draw("colz");
Momentumk->SetStats(kFALSE);
Momentumk->SetTitle("Calculated and Theoretical momenta vs #theta (from KickPlane)");
```

```
  Momentumk->GetXaxis()->SetTitle("P (MeV/c)");
  Momentumk->GetYaxis()->SetTitle("#theta (Deg)");
  MomTeok->SetMarkerColor(kRed);
  MomTeok->Draw("same");

  TCanvas *GAMMA = new TCanvas();
  gamma->Draw();
  gamma->SetTitle("");
  gamma->GetXaxis()->SetTitle("tan#theta_{1} * tan#theta_{2}");
  gamma->GetYaxis()->SetTitle("Entries");

  TCanvas *INVMASS = new TCanvas();
  invmass->Draw();
  invmass->SetStats(kFALSE);
  invmass->SetTitle("Invariant Mass of pp elastic pair");
  invmass->GetXaxis()->SetTitle("M_{inv}^{pp} (MeV/c^{2})");
  invmass->GetYaxis()->SetTitle("Entries");

  TCanvas *SEC = new TCanvas();
  Sector2->SetLineColor(kRed);
  Sector2->Draw();
  Sector1->SetLineColor(kBlue);
  Sector1->Draw("same");

  TCanvas *DELTAP = new TCanvas();
  DeltaP->Draw();
  DeltaP->SetTitle("Momentum Resolution ");
  DeltaP->GetXaxis()->SetTitle("#DeltaP/P");
  DeltaP->GetYaxis()->SetTitle("Entries");

  TCanvas *DELTAPvsP = new TCanvas();
  DeltaPvsP->Draw("colz");
  DeltaPvsP->SetTitle("Momentum Resolution vs Momentum ");
  DeltaPvsP->GetXaxis()->SetTitle("P (MeV)");
  DeltaPvsP->GetYaxis()->SetTitle("#DeltaP/P");

  TCanvas *DELTAPvsTHETA = new TCanvas();
  DeltaPvsTheta->Draw("colz");
  DeltaPvsTheta->SetTitle("Momentum Resolution vs #theta ");
  DeltaPvsTheta->GetXaxis()->SetTitle("#theta (deg)");
  DeltaPvsTheta->GetYaxis()->SetTitle("#DeltaP/P");

}
```

Bibliography

[AP02] H. Álvarez Pol. *On the Multiwire Drift Chambers alignment of the HADES dilepton spectrometer*. PhD thesis, USC, 2002.

[BR97] R. Brun and F. Rademakers. Root-an object oriented data analysis framework. *Nucl. Instr. and Meth. A*, 389:81, 1997.

[Bre99] T. Bretz. *Magnetfeldeigenschaften des Spektrometers HADES*. Diploma Thesis, T.U. München, 1999.

[Col94] HADES Collaboration. Proposal for a high acceptance dielectron spectrometer. *GSI Scientific Report*, 1994.

[Col04] HADES Collaboration. Dielectron production in pp, dp and AA collisions. *GSI Scientific Report*, 2004.

[ea04a] P. Salabura et al. Study of e^+, e^- production in elementary and nuclear collisions near production threshold with hades. *Nucl. Phys.*, 53:49, 2004.

[ea04b] S. Eidelman et al. Passage of particles through matter. *Phys. Lett. B*, 592:1, 2004.

[Fn05] N. Fariña. *Perfomance Analysis and Improvements of the Santiago Tracking using Simulated and Real Data in the HADES Experiment*. Diploma Thesis, USC, 2005.

[Fre88] A.P. French. *Relatividad Especial*. Reverté, Barcelona, 1988.

[GEA93] Geant-detector description and simulation tool. CERN, 1993.

[GPS02] H. Goldstein, C. Poole, and J. Safko. *Classical Mechanics*. Addison-Wesley, San Francisco, third edition, 2002.

[GSI] `http://www.gsi.de`.

[HAD] `http://www-hades.gsi.de`.

[Iva04] A. Ivashkin. Momentum reconstruction with runge-kutta method and it's application to *pp*-scattering. *Internal HADES Report, GSI*, 2004.

[KKS95] W. Koënig, U. Kopf, and D. Schüll. Specification of the hades superconducting torus system. *GSI Scientific Report*, 1995.

[Kop95] G.I. Kopilov. *Simplemente Cinemática*. Platón, Alicante, 1995.

[Mar84] J.B. Marion. *Dinámica Clásica de las partículas y sistemas*. Reverté, Barcelona, seventh edition, 1984.

[McC99] W.D. McComb. *Dynamics and Relativity*. Oxford University Press, New York, 1999.

[PDG] `http://pdg.lbl.gov`.

[Prz03] W. Przygoda. Brief report on proton-proton collisions simulation. *Internal HADES Report, GSI*, 2003.

[ROO] `http://root.cern.ch`.

[Rus03] A. Rustamov. Momentum reconstruction with spline in hades. *Internal HADES Report, GSI*, 2003.

[SG99] M. Sánchez García. *Diseño y Programación orientado a Objetos de la reconstrucción de Sucesos en el Experimento HADES de Colisiones Núcleo-Núcleo*. Diploma Thesis, USC, 1999.

[SG03] M. Sánchez García. *Momentum Reconstruction and Pion Production Analysis in the HADES Spectrometer at GSI*. PhD thesis, USC, 2003.

[Spa05] S. Spataro. *pp* collisions at 2.2 gev with hades. *International Meeting on Nuclear Physics, Bormio*, 2005.

[Woo03] N.M.J. Woodhouse. *Special Relativity*. Springer-Verlag, London, 2003.