UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

FACULTAD DE FÍSICA

Departamento de Física de Partículas

Nuevos algoritmos para la reconstrucción de trazas en las cámaras de deriva MDC del espectrómetro HADES

Memoria presentada por:
Teresa Kurtukian Nieto
como trabajo de investigación tutelado, dentro del programa de doctorado Física de Partículas y Dinámica no Lineal

Supervisor:
J. Garzón Heydt

23 de Junio 2003

Contents

List of Figures

List of Tables

Chapter 1

Introducción: el espectrómetro HADES

HADES (High Acceptance Di-Electron Spectrometer) [1, 2] es el acrónimo que describe a una gran colaboración de más de 150 científicos de 9 países europeos. El espectrómetro HADES está instalado en el GSI (Gesellschaft für Schwerionenforschung) un centro para la investigación de iones pesados localizado en Darmstadt, Alemania.

HADES pretende investigar las propiedades de los mesones vectoriales ρ, ϕ, y ω, en medios de alta densidad bariónica. La producción de los mesones dentro de la materia nuclear se consigue mediante la colisión de haces de iones pesados, piones y protones.

Para observar experimentalmente los cambios en las propiedades de los mesones vectoriales es necesario estudiar un tipo de decaimiento cuyos productos no cambian debido a la interacción fuerte de los estados finales. Este canal corresponde al decaimiento de dileptones (pares e^+e^-). En este decaimiento la masa vectorial puede reconstruirse como era durante la fase densa, ya que el vector 4-momento de los leptones no cambia debido a la interacción fuerte. La probabilidad de que ocurra este decaimiento es muy baja y el número de dileptones procedentes de otras fuentes (decaimiento Dalitz de π^0, bremsstrahlung y conversión de fotones externos) pueden obscurecer la señal.

Para poder seleccionar los dileptones y medir su masa invariante, se requiere construir un espectrómetro capaz de proporcionar una alta aceptancia y resolución en la reconstrución de masas. En base a estos re-

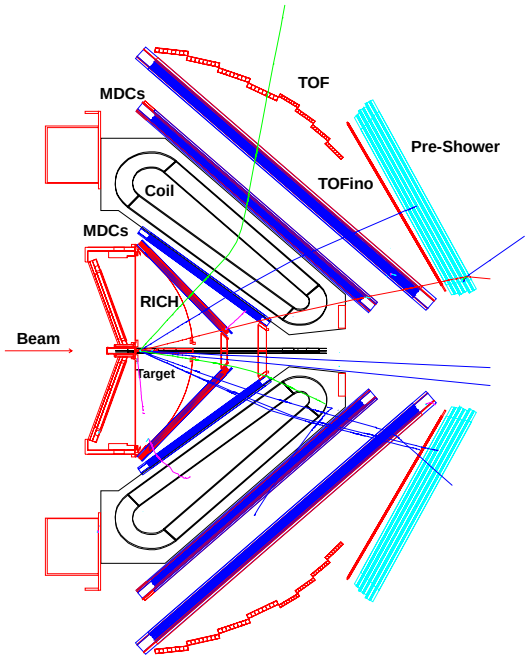

Figure 1.1: *El espectrómetro HADES.*

querimientos se construyó HADES con simetría hexagonal alrededor del eje del haz y organizado en seis sectores cubriendo un ángulo polar entre 18 y 85 grados y prácticamente todo el rango azimutal, solo limitado por los marcos de los detectores y el imán.

Como puede verse en la figura 1.1, el espectrómetro está compuesto por los siguientes detectores:

- START: compuesto por dos detectores idénticos de diamante localizados antes y después del blanco, denominados Start y Veto, encargados de abrir las ventanas de adquisición en otros detectores.

- Un contador umbral de anillos producidos por luz Cherenkov (RICH). Realiza la identificación de los leptones, siendo insensible al paso de los hadrones, incluso de los piones de mayor velocidad (hasta 3 GeV).

- Cuatro cámaras de deriva multihilos (MDC), localizadas dos antes y dos después del imán, encargadas de la determinación de las trayectorias de las partículas cargadas, para la caracterización de los eventos via distribuciones angulares y de momentos.

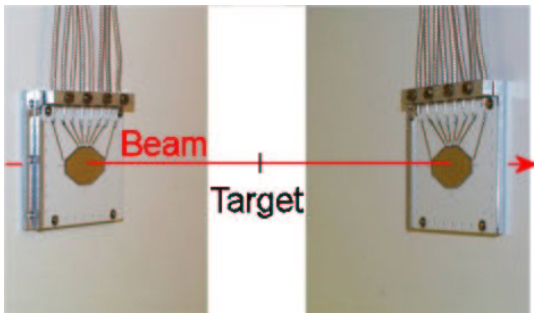

Figure 1.2: *START y VETO*

Figure 1.3: *Ring Imaging Cherenkov (RICH)*

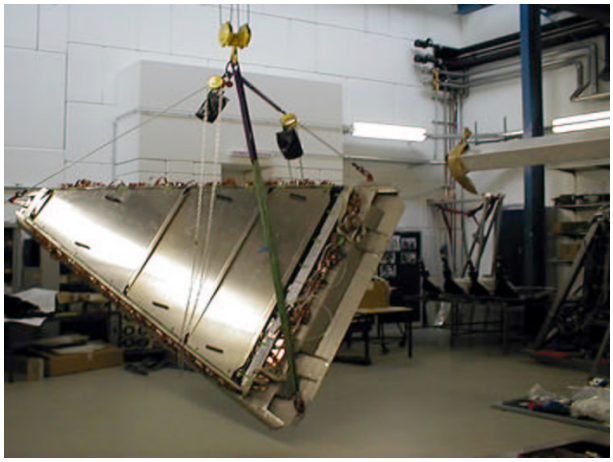

Figure 1.4: *Cámara de deriva multihilo MDC*

Figure 1.5: *IMAN Superconductor*

- Un imán superconductor que genera un campo toroidal en la zona limitada espacialmente entre las camaras de deriva, que proporciona un cambio de momento de las partículas cargadas del orden de 100 MeV/c , lo cual permite obtener una resolución en la determinación del momento cercana al 1% .

- META (Multiplicity/Electron Trigder Array), diseñado para seleccionar las colisiones centrales(primer nivel de trigger) y para realizar una rápida identificación de los leptones, que junto con la información del RICH, distingue las trazas leptónicas válidas (segundo nivel de trigger). El META esta formado por los siguientes detectores:

  - TOF(time of flight)/TOFINO: matríz de centelleadores utilizado para medir el tiempo de vuelo de las partículas cargadas en todo el ángulo polar del espectrómetro.
  - Pre-Shower: para bajos ángulos polares, donde el tiempo de vuelo no es efectivo en la identificación de leptones, se utiliza un detector de cascadas electromagnéticas, basado en tres camaras de hilos y electrodos que recogen la señal inducida, y dos convertidores de plomo localizados entre las cámaras. La diferencia entre la carga depositada en las cámaras antes y después de los

convertidores en el desarrollo de la cascada electromagnética permite diferenciar e identificar los leptones.

Del conjunto de detectores existentes en el espectrómetro HADES, las cámaras de deriva juegan un rol fundamental, ya que estas permiten reconstruir las trayectorias de las partículas cargadas, lo que permite, con la ayuda del iman toroidal superconductor, la determinación del momento de las partículas.

Los algoritmos de reconstrucción de trazas son usualmente conocidos como *trackers* o *tracking software*. El Grupo Experimental de Nucleos y Partículas (GENP) de la Universidad de Santiago de Compostela, ha desarrollado un algoritmo de reconstrucción basado en el ajuste analítico por el método de mínimos cuadrados de las trazas de las partículas que atraviesan cada cámara de deriva (MDC).

La tarea de reconstrucción de trazas se divide en cuatro pasos. En los dos primeros se reconstruyen impactos (señales en cada plano de hilos) en cada uno de los cuatro módulos y se obtienen segmentos rectos en los dos módulos internos(MDC I y II) y en los dos módulos externos(MDC III y IV). En el tercer paso se empalman los dos segmentos de traza y en el cuarto paso se ajusta la traza completa.

Para reconstruir *hits* (4,5 o 6 impactos ajustados a una recta), se utiliza un algoritmo de ajuste por mínimos cuadrados, en forma analítica. Partiendo de las coordenadas de los impactos candidatos en cada plano de hilos, expresadas en el sistema de referencia local del plano, se obtienen los parámetros del *hit*, expresados en el sistema de referencia local de la cámara. Estos parámetros son: las coordenadas x_0, y_0 correspondientes al centro de la cámara (z = 0), las pendientes x', y', y la matriz de error de los parámetros, así como el χ^2 del ajuste. El algoritmo busca primero *hits* con 6 impactos. Con los impactos restantes se construyen *hits* con 5 impactos y si se requiere con 4 impactos.

Para reconstruir *segmentos*, antes y después del campo magnético, los *hits* en el primer módulo son extrapolados y transformados al sistema de referencia del segundo módulo. Se verifica la compatibilidad para todas las combinaciones de *hits* en ambos módulos. El criterio de compatibilidad asegura que la diferencia entre las coordenadas que definen el *hit* en ambos módulos se encuentra dentro de ciertos valores máximos, teniendo en cuenta la correlación entre las variables. Luego, cada par de *hits* compatibles se ajustan a una recta. Los parámetros de los segmentos son: r, z, θ, ϕ, la matríz de error de los parámetros y el χ^2 del ajuste.

Tras esta introducción, el texto de la presente memoria se escribirá en lengua inglesa a fin de que pueda servir como documentación para

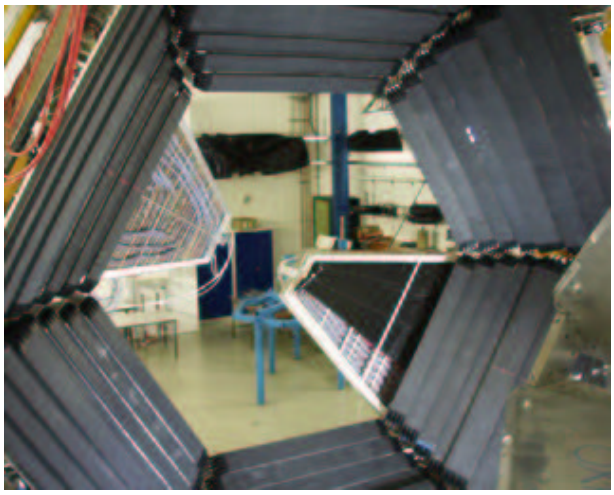

Figure 1.6: *Time of Flight (TOF)*

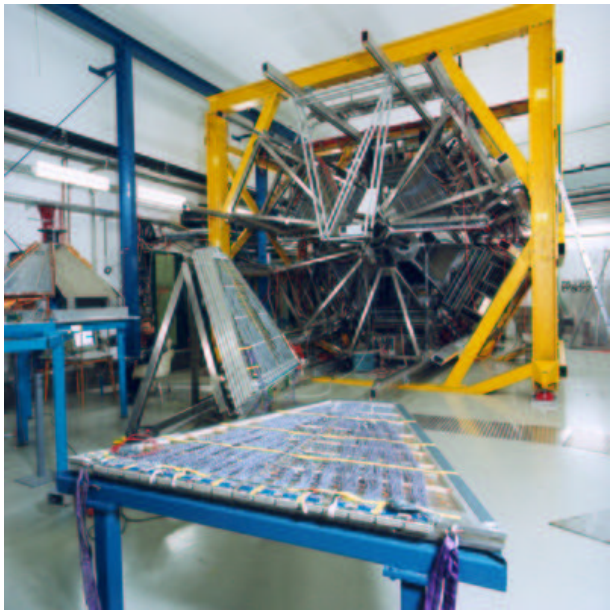

Figure 1.7: *Shower*

los miembros de la colaboración internacional HADES. Asi pues, en el capítulo 2, se expone la teoría de funcionamiento de las cámaras de deriva, en el capítulo 3 se hace una descripción detallada de las cámaras de deriva multihilo del espectrómetro, en el capítulo 4 se describe el algoritmo de reconstrución de trazas y se muestran algunos resultados obtenidos con el mismo y finalmente en el capítulo 5 se muestran las conclusiones de este trabajo y las recomendaciones para futuras mejoras del programa de reconstruccón de trazas.

Chapter 2

Theory of Drift Chamber Operation

2.1 Ionization

When a charged particle traverses a gas chamber it transfers energy to the gas, by ionizing molecules and atoms of the gas. The mean rate of ionization loss of a charged particle, as given by the Bethe-Bloch formula [3]

$$\frac{dE}{dx} = \frac{4\pi N_o Z^2 e^4}{mv^2}\frac{Z}{A}[ln\frac{2mv^2}{I(1-\beta^2)} - \beta^2] \tag{2.1}$$

The energy loss to the gas results in the formation of ion pairs (positive ions and negative electrons) in the medium. First the incident particle collides with gas molecules and creates drift electrons, the so called primary ionization. Those electrons produced in this process that have a high enough energy will create futher inonization themselves; this process is known as secondary ionization. This secondary ionization occurs when the electron has acquired an energy greater than the ionization energy.

2.2 Cluster Size Distribution

One of the factors affecting the space resolution of drift chambers is the non-continuous nature of the distribution of ionization along the particle track, expressed as the cluster-size distribution [4]. As an incident particle traverses the chamber, the ionization of the gas is not a continuous distribution. Instead, this ionization is deposited in lumps, or ionization clusters, each produced as a result of individual primary ionizations. The probability distribution of the number of electrons ionized, either directly

or indirectly, by a particle on its trajectory is known as cluster-size distribution. The cluster size, ie, the total number of ions liberated close to the primary ionization event, may vary from one to many electrons, and is roughly proportional to the energy transferred in the primary ionization, and also depends on the ionization energy required to liberate an ion pair. More precisely, it is necessary to determine the spectrum of energy loss $F(E)dE$, and the probability $p(E,k)$ for each energy E of producing exactly k ionization electrons. The cluster size distribution $P(k)$ can then be found by integrating over the energy [5]:

$$P(k) = \int F(E)p(E,k)dE \tag{2.2}$$

2.3 Drift of Electrons

Once ionization of the gas has occurred, and electrons and ions have been produced, we are interested in the drift properties of these particles. On average, both electrons and ions drift with a constant drift velocity v_d, in the direction of the electric field E. The electron will travel an average distance λ, the mean free path, before it collides with a gas molecule. After an electron collides, it scatters isotropically, due to its small angular momentum. The electron is then accelerated by the field until it collides with another gas molecule. The actual drift velocity is much smaller than the instantaneous velocity between collisions, and is given by:

$$v_d = \mu E = \frac{e}{m} E_\tau \tag{2.3}$$

The scalar mobility μ is defined as the ratio of drift velocity to electric field in the absence of a magnetic field and is proportional to τ , the mean time between collisions [5]. The energy acquired is then transferred, in the next collision, to recoil or excitation. The drift velocity is also affected by the effective fractional energy loss per collision. A full account of these effects is not given here. However, Blum and Rolandi [5] discuss this in further detail. The E field, and thus the drift velocity, is relatively constant for most of the drift distance, but close to the wire, it increases rapidly, so that the drift velocity becomes large very close to the wire.

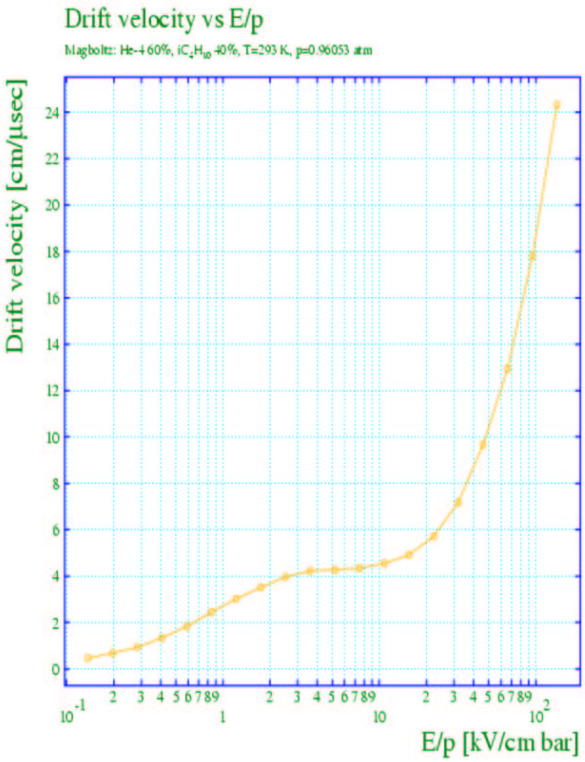

Figure 2.1: *The Drift Velocity*

2.4 Drift of Ions

The drift properties of ions differ from those of electrons because they possess a significantly larger mass and interact differently with the gas. Electrons are more rapidly accelerated in an electric field than ions, and lose very little energy in elastic collisions with gas atoms. In typical drift chambers, electrons reach energies much greater than thermal motions, and their mobility is a function of the energy loss associated with inelastic excitation of gas molecules [5].

Ions in similar fields acquire a similar amount of energy as do electrons, but much of this energy is lost in the next collision. Additionally, the ion momentum is less randomized in each collision, thus less field energy is stored in random motion, and the random energy of ions is mostly thermal. The result is that the effect of diffusion on ions is orders of magnitude smaller than in electrons.

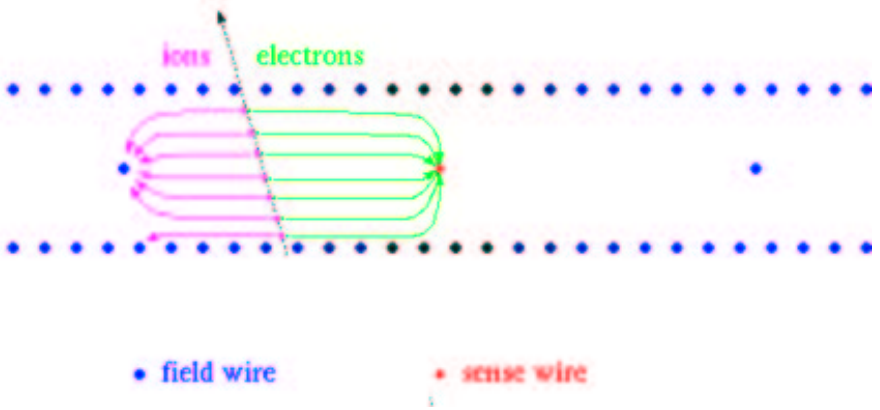

Figure 2.2: *Drift cell.*

2.5 Drift Chambers

The drift chamber is an instrument used in the detection of charged particles by means of observing ionization in the chamber medium (usually taken to be gaseous) induced by the particles as they traverse the chamber, interacting with the molecules within. For this purpose thin wires are fixed in a volume filled with a special gas, in a way that the wires form cells:

When a particle traverses the chamber, it collides with gas molecules and liberates "drift" electrons; this is known as the primary ionization.At the high-voltage planes, an electric field is created in the drift chamber that is perpendicular to the wire plane. This field is constant through most of the drift area, but near the thin wires it becomes closer to a purely radial (1/r) field. The liberated electrons drift along these field lines with a nearly constant velocity. Once a drift electron enters the region of increasing field, it eventually acquires enough energy to create secondary ionizations in the gas. The process repeats with the secondary electrons, which creates an avalanche effect in the region of the wire. The separation of the positive and negative charges of this avalanche creates a signal in the sense wire.

The time that elapses between the primary ionization event and the creation of a signal is roughly proportional to the distance traveled by the drift electron , that is, the distance measured along the field lines from the location of the primary ionization to the wire. By analysis of the distances, perpendicular to the plane of the wires, from the wires to the trajectory of the incident particle for several wires, the angle of the trajectory and the location at which it crossed the wire plane can be determined.

The length of the path traveled by the drift electron is not the same

as this perpendicular distance, however, and an important part of the data analysis in all experiments using drift chambers lies in determining this perpendicular distance from the time measured. To do this, a drift-chamber simulation program called GARFIELD can be used to simulate the drift-time properties of the chamber.

GARFIELD is a computer program written by Rob Veenhof at CERN for detailed simulation of two-dimensional drift chambers [7]. It can be used to calculate field maps, x(t) relations, arrival-time distributions, and signals. It includes an interface to the program MAGBOLTZ, written by Steve Biagi, which computes gas properties in arbitrary mixtures of commonly used gases.

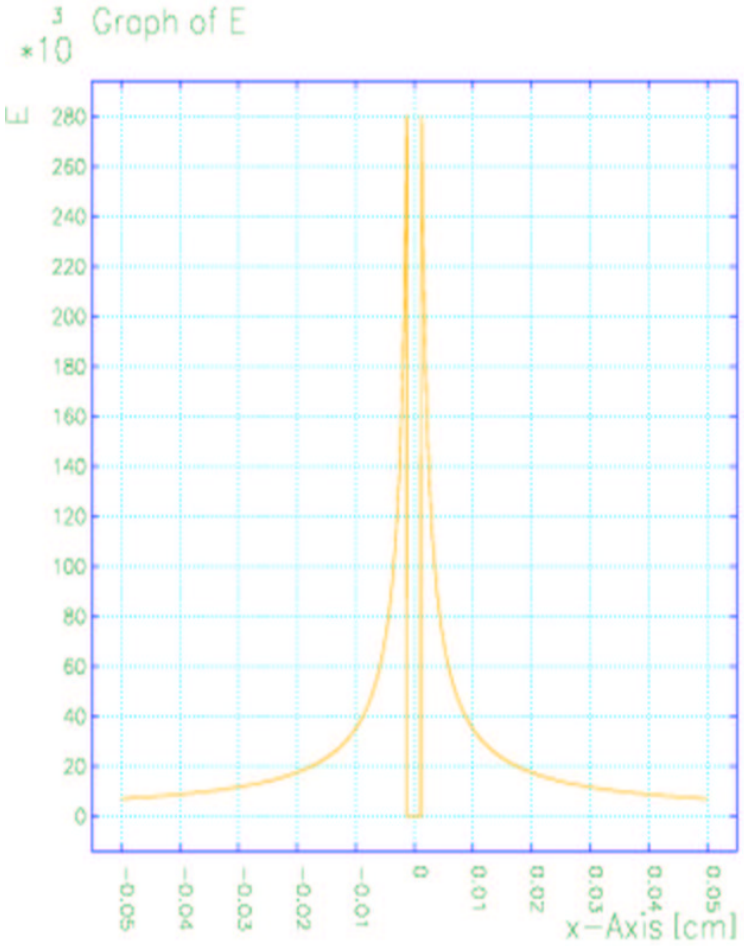

Figure 2.3: *The magnitude of the electric field, as a function of the perpendicular distance from the wire plane, along an axis through the wire*

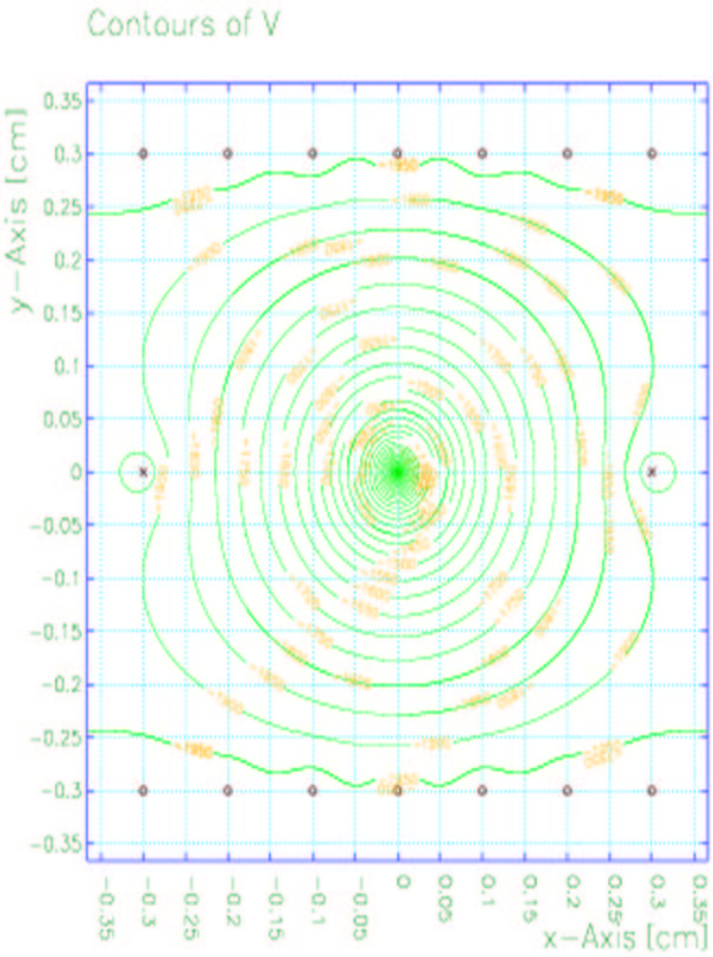

Figure 2.4: *The contour of the voltage*

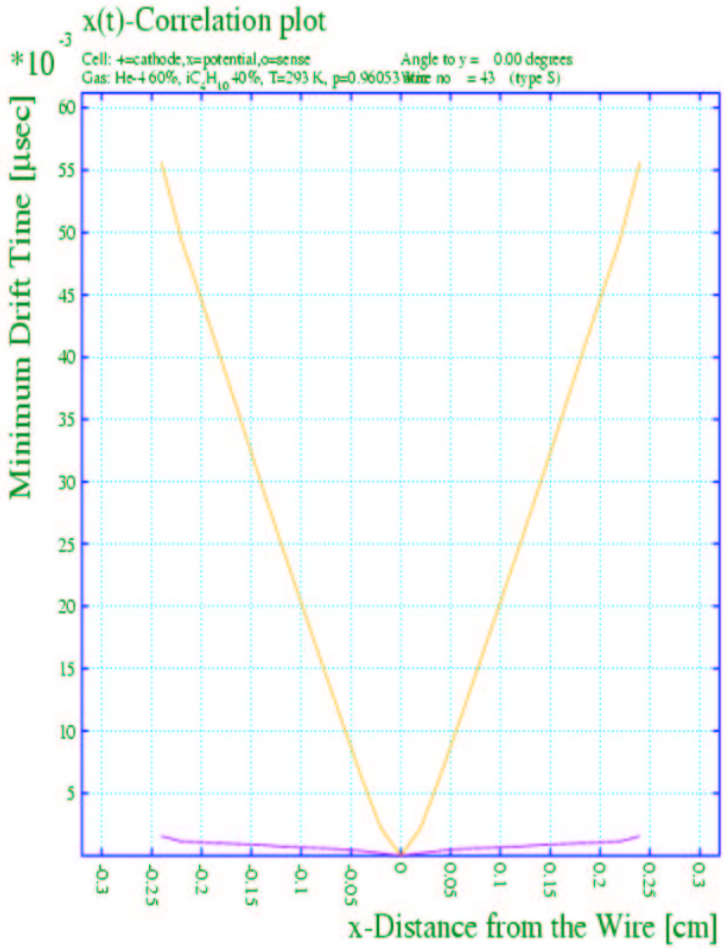

Figure 2.5: *x-t correlation plot*

Chapter 3

HADES Multiwire Drift Chambers

3.1 Chambers specifications

The HADES tracking system consists of four drift chambers (divided in 6 modules, one per sector), two before and two behind the field area of the superconducting magnet.They are refered in this text with the Latin numbers I, II, III and IV. The module sizes range from 88 $cm \times 80\ cm$ for the inner one to 280 $cm \times 230\ cm$ for the outer.

Plane	sense wires (#)	cell size ($x \times z$) mm^2	active area m^2	inner size $cm \times cm$
MDC I	1006	5×5	0.35	$76(12) \times 79$
MDC II	1104	6×5	0.53	$86(20) \times 100$
MDC III	1098	12×8	2.21	$185(30) \times 206$
MDC IV	1159	14×10	3.21	$220(32) \times 255$
Totals	4367		6.3	
Grand total	26202		37.8	

Table 3.1: *Basic properties of the HADES MDCs*

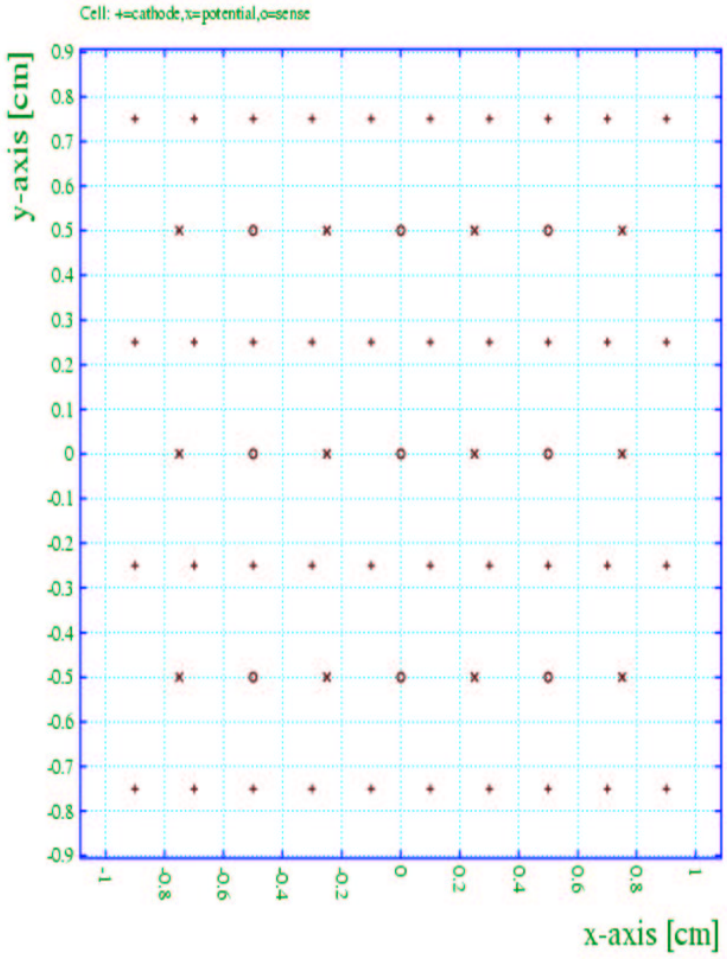

Figure 3.1: *Drift cell example*

Each module consists of six independent sense and field wire planes surrounded by seven cathode planes which determine the drift cells. The sense wires are grounded, while the cathode and field wires have a positive voltage. The approximate cell configuration is shown in figure 3.1

The sense wires of the different wire planes follow different directions, the so-called butterfly geometry [8]. The two inner sense wires are parallel to the parallel frames of the module, while the outers are tilted 20^o and 40^o degrees respectively, as shown in figure 3.2.

This disposition emphasizes the drift chamber resolution in the direction where the magnetic field deflects the particles and eliminates more

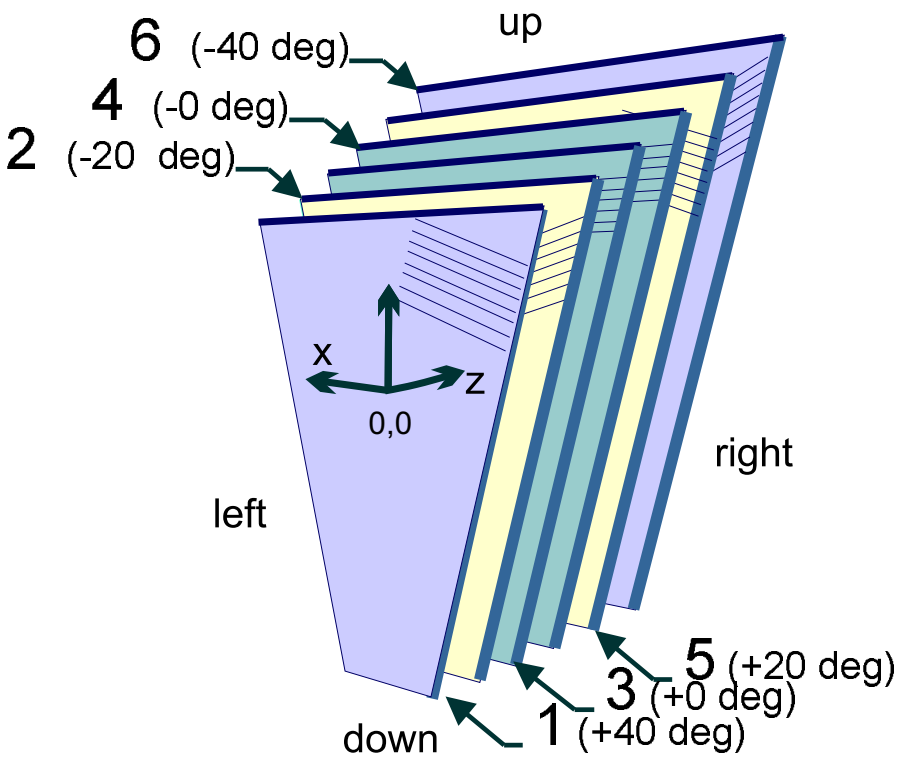

Figure 3.2: *Configuration of the six sense wire layers in a HADES multiwire drift chamber.*

efficiently identification errors. For a fired wire one can assign the ionization at any of both sides of the wires. The identification errors come from the assignment of wrong sides in the fired wires when the hits are constructed. The cathode wires are disposed perpendicular to the (parallel) top and bottom frames.

3.2 Coordinate systems

The coordinate systems used for the description of the MDCs are :

- The *Laboratory Reference System*, notated (X, Y, Z), is defined unequivocally as that having Z axis pointing in the beam direction and sense, Y axis pointing in the direction of the gravity but contrary sense, and X axis settling down a right-handed system.

- The *Module Reference System* is notated as $(X_{MDC}, Y_{MDC}, Z_{MDC})$ and is module dependent. Its origin is attached to the physical center point of the drift chamber module, i.e., the physical center point is precisely defined in each module as the cross point of the fourth cathode plane in the module and a straight line normal to the plane and crossing the target. The axes X_{MDC} and Y_{MDC} lie inside the fourth cathode plane, while Z_{MDC} is normal to this plane pointing in the opposite sense to the target.

3.3 Gas mixture

The requested features for the drift modules include the use of low mass gases which present a large radiation length to keep the multiple scattering along the spectrometer low. The contribution of the multiple scattering to the momentum resolution results crucial, being of the same order as the intrinsic resolution of the drift modules for electrons in all the studied momentum range. Multiple scattering dominates at low ($< 0.4\ GeV/c^2$) dilepton invariant mass [9]. The most interesting mixture is that made of Helium and Isobutane.

The mixture containing a 60% Helium and 40% Isobutane was chosen as optimal choice, providing enough primary ionization under stable operation at moderate gains. The total gain is approximately 10^5 and the drift velocity is $4cm/\mu s$ in most of the drift path [10, 11].

For a mixture 60% Helium and 40% Isobutane, there are about 35 clusters/cm path length. The ionization electrons drift toward the wire, reaching this point at different times. The drift in the cells follows the paths given by the electrostatic configuration; the shorter paths correspond to the minimum trajectory to sense wire distance.

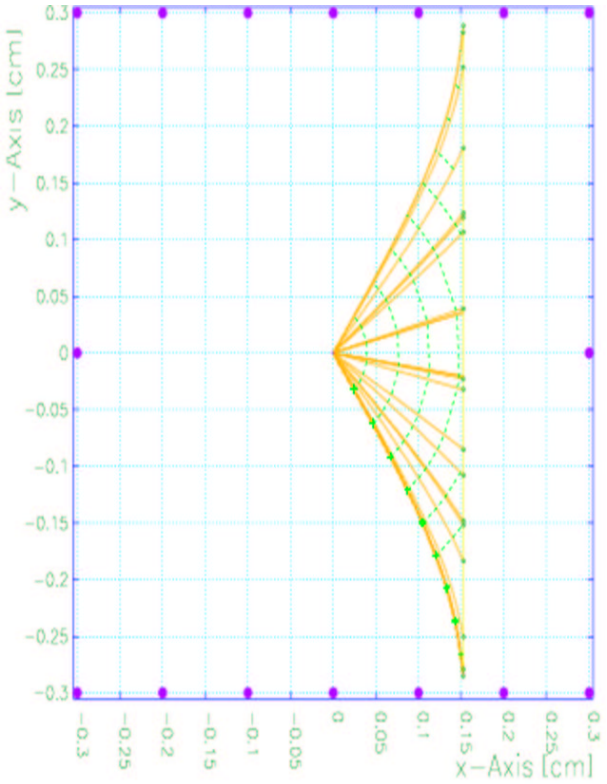

Figure 3.3: *Electron drift lines.*

The number of clusters per path length is large enough to work with the hypothesis that there are ionization electrons in a narrow interval from the closer point of the trajectory to the wire. Then, the signal gives us a measurement of the distance to the point of the trajectory which passes closer to the wire.

3.4 Position resolution

To obtain the drift distance, the time of an external reference, the Start detector, is used. The leading and trailing edges of the pulse over the threshold define the start time for two different channels in the TDC, counting up to the arriving of a stop signal. The drift time is given by the difference between the first (leading edge) signal and the stop signal in the TDC, which is a delayed line from the Start detector. But this time contains also an offset and should be calibrated from TDC channels to time. Once one knows the time, the distance can be obtained. The details are explained in the next section.

The position resolution of the modules was measured in several tests and by different methods [10, 11].

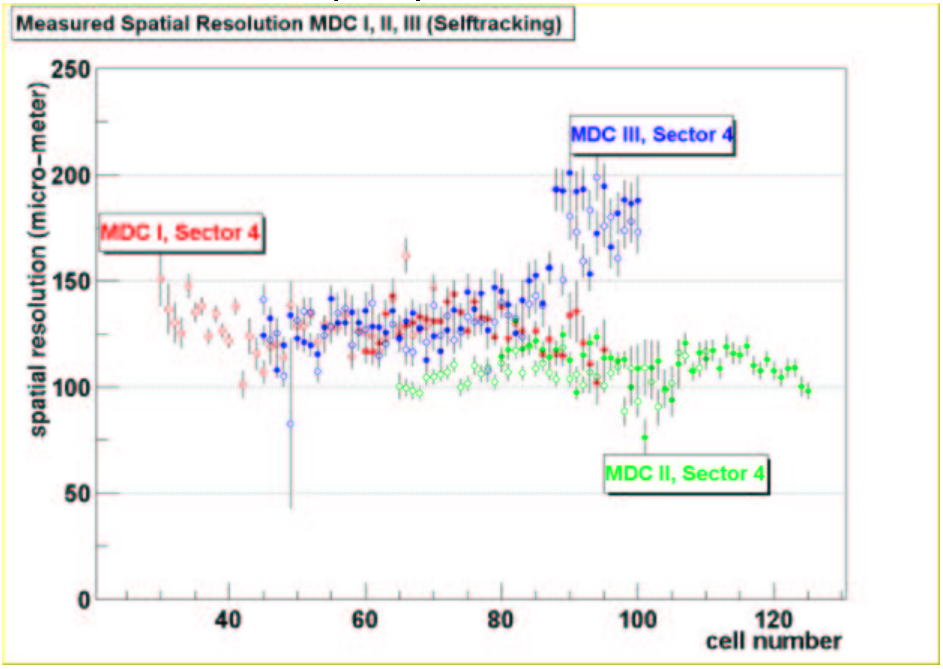

Figure 3.4: *Spatial resoluction of HADES MDCs I,II,III.*

3.5 Bilinear Interpolation from a Table of GARFIELD Data

In order to make use of this method to determine x position from measured drift times, it is necessary to construct a table of values consisting of the positions corresponding to several incident angles and drift times. This data can be quickly calculated from GARFIELD for different parameters.

The basic method of bilinear interpolation on a table of values is discussed in Numerical Recipes [12]. The method is as follows: first, a grid of the four tabulated x values surrounding the desired value is found. If we seek the x for a given t_1 and θ_1, we first find j and k such that

$$t[j] <= t_1 <= t[j+1] \tag{3.1}$$

$$\theta[k] <= \theta_1 <= \theta[k+1] \tag{3.2}$$

This is done using a bisection algorithm which will locate the right value in about $log_2 n$ tries, where n is the number of data points in the table

xdata. The four points surrounding the desired value are then given by:

$$x_1 = xdata[j][k] \tag{3.3}$$

$$x_2 = xdata[j+1][k] \tag{3.4}$$

$$x_3 = xdata[j+1][k+1] \tag{3.5}$$

$$x_4 = xdata[j][k+1] \tag{3.6}$$

The simplest way to interpolate in two dimensions is then to define

$$u \equiv \frac{(t_1 - t[j])}{t[j+1] - t[j]} \tag{3.7}$$

$$v \equiv \frac{(\theta_1 - \theta[j])}{\theta[j+1] - \theta[j]} \tag{3.8}$$

We then approximate that

$$x(t_1, \theta_1) = (1-u)(1-v)x_1 + u(1-v)x_2 + u\ v\ x_3 + (1-u)v\ x_4 \tag{3.9}$$

This method results in a very smooth function x(t), that is always between the values given in the table, and varies continuously between grid squares. A more accurate method, that of interpolating a polynomial of order 3 or 4 using the 3 or 4 values in each dimension in the table that are closest to the desired value can also be used, but much calculation time is lost and this sort of accuracy is probably not necessary.

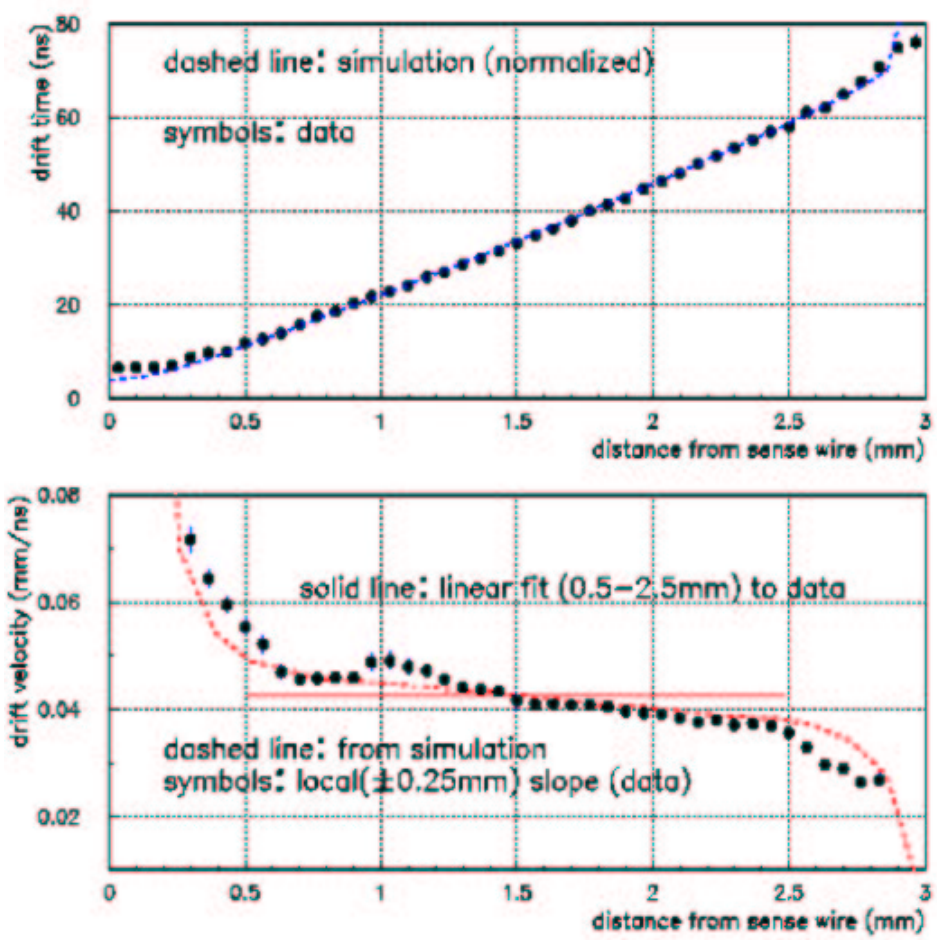

Figure 3.5: *Drift Velocity and time-distance relation for MDC prototype.*

Chapter 4

MDC tracking algorithm

4.1 HADES software

The HADES basical software package for the data simulation, reconstruction and analysis is called HYDRA (HADES System for Data Reduction and Analysis) [13].

HYDRA is an application based in ROOT [14], an Object Oriented Data Analysis Framework. ROOT is designed for its use in High Energy Physics experiments, providing an user interface, data organization and streamer, histogramming, automatic documentation and many other interesting features. For simulation, a HADES Simulation package called HGeant has been developed, based on Geant 3.21 [15].

The HYDRA classes can be categorized in few groups. There is a fundamental class called HADES which encapsulates the whole reconstruction program, providing methods to control the different tasks to be done. The HADES class contains: the data source where the event's data are read from, the tasks set, the information about the spectrometer structure, the output files, the output tree and the parameters database. There are classes for the storage of the data in different levels of refinement (data levels) as well as classes to manage the input/output of the data. An important group of classes contains and manages the parameters (parameter containers), including a version management for parameters which can be modified. Finally, there are classes to perform tasks; the tasks are the representation of algorithms or groups of algorithms.

4.2 Track reconstruction

The trajectory of the tracks is approximated, before and after the magnetic field zone, using straight lines. The straight lines are obtained from the two estimates in the drift modules. This approximation results in several interesting benefits:

- The determination of the deflection angle in the magnetic field zone is not affected by the angular straggling in the space between the target and the drift modules.
- The scheme improves the close pairs separation, mainly coming from conversion of photons in leptonic pairs.
- Each straight piece can be independently extrapolated to other detectors in charge of particle recognition as the inner RICH or outer TOF or Pre-Shower.

The tracking method developed in Santiago to the HADES Collaboration has two main steps

- Recontruction of hits candidates in each MDC independently
- Reconstruction of segments starting from pairs of 2 hits in the same leverarm

This two steps are included in the HYDRA program by means of a library called libMdcTrackS.so which includes the reconstructors HMdcHitF and HMdcSegmentF.

In the following sections the code developed is concisely described, including a brief description of the calibration data levels and the parameter containers.

4.3 Data levels

A set of software data levels has been developed to structure the information from the modules and simplify data comparison and conversion between successive calibration steps.

The information registered in the drift modules is written in a fixed format called Raw MDC subevent, which contains the hardware information where the signal was generated and the channel contents in the corresponding TDC channel. This information is "unpacked" to a more friendly format for further calibration. After the unpacking, a software data level

denoted as **Raw** is filled. This level still contains the hardware address and the TDC raw data in channels, now structured inside the classes scheme of analysis software package HYDRA.

The next data level is called **Cal1** and is obtained from Raw through a calibration procedure. The first calibration step translates the hardware addresses to module addresses. The information from the TDCs is transformed to drift times using a linear function. The parameters in the linear correspondence are the TDC offset and gain, individually determined and stored as parameters. The gain is determined from the internal calibration procedure in the TDC. The offset is introduced to remove from the measured time the part that does not correspond to drift time,i.e., a part of the time of flight of the particle. A global offset is removed from the time reported by the TDC which corresponds to the time of the fastest particles which passes the cell closest to the wire. These particles define the sharp edge of the time distribution, being all the other drift times larger (or including in the TDC time information a larger part of the time of flight). Up to this point, there is no identification of the particles producing the signal or about their momenta so there is no way to refine the method discriminating tracks. For instance, if the offset is determined by light particles ionizing the gas close to the wire, then the offset should be increased for heavier particles. As heavy particles with lower velocities will cross the modules later than lighters, the common offset will not remove for these particles the complete time of flight. This can be corrected in the hit finder as will be shown later.

The next data level called **Cal2** stores the distance to the sense wire. The drift distance can be obtained from the drift time once it is known the drift velocity along the ionization electrons path toward the sense wire. As a first approximation, the drift velocity can be taken as constant (for a MDC II with a mixture 60% Helium and 40% Isobutane at a voltage of $-2\ kV$ in cathode and field wires, it was determined a mean drift velocity $v_d = 0.042\ mm/ns$, constant in around 80% of the cell [10, 11]). In a more sophisticate approximation, the transformation from drift time to drift distance is done by the bilinear interpolation of the GARDIELD xt-plot data, that takes into account the dependence of the drift velocity with the track angle and with the distance to the sense wire.

An estimation of the geometrical parameters of the track in its path through the module can be obtained from the previous levels. This information is stored in the so-called **Hit** level.

From two Hits or directly from the calibrated information in two modules it is possible to construct the next data level, called **Segment**. A

segment contains the best estimate to the straight line path in a "lever arm" before or after the magnetic field area.

Aditionally to this data levels the *Santiago Tracking Software* includes two internal data levels:

- Cal3: is a calibrated coordinate of the impact in a MDC plane in the local reference system.
- HitAux: is a hit in a MDC. Basically stores the output of the hit-finder fit function.

4.4 Parameter containers

The parameters are stored in parameter containers which are accessible in the runtime database via their names. The *Santiago tracking* uses the following parameter containers:

- Geometry containers: this containers are the official geometry parameter containers for HYDRA
  - MdcGeomStruct: container for parameters decribing the geometrical tree of an Mdc (sectors, modules, layers with the number of cells in each layer. This information is needed to create the data tree and to create the parameter containers for the Mdc.
  - MdcGeomPar: container for the basic MDC geometry parameters
  - MdcLayerGeomPar: container for the Mdc specific geometry parameters of the layers.
  - SpecGeomPar: parameter container for the geometry of the cave, the sector(s) and the target(s)
- Reconstructors containers: this parameter containers are specifics for the *Santiago tracking*
  - MdcHitFPar: container needed by the class HMdcHitF, to store parameters for the reconstruction of mdc data in one module. The parameters included are:
    * Minimum number of impacts in a hit. The minimum value of this parameter is 4.[1]

[1]It is not recomended to use 4 planes in the fit because allways it is posible to fit a 4-parameter hit with 4 planes

    - Road to incorporate impacts to a hit candidate.
    - Maximum χ^2 per degree of freedom to accept the hit.
    - Maximum number of common impacts between two hits.
    - Maximum slope of the hit projected in the XZ plane.
    - Minimum slope of the hit projected in the XZ plane.
    - Maximum slope of the hit projected in the YZ plane.
    - Minimum slope of the hit projected in the YZ plane.
    - Maximum distance to z axis.
    - Sigma of layer
  - MdcSegmentFPar : stores the parameters needed by the HMdcSegmentF class, the reconstructor of one lever arm of the MDC system. The parameters included are:
    - Correlation factor between Δx-ΔxDir of hits.
    - Correlation factor between Δy-ΔyDir of hits.
    - Parameter of the four-dimensional contour line.

4.5 Track finder and fitting

The hit finder and fitting takes as input the distances to the Cal2 data level, which contains basically the distance from the wire to the closest point of the ionizing particle. The Cal2 data is converted to an *impact coordinate* in a plane. For each sense wire plane, the relevant *impact coordinate* is the distance to a reference wire in the plane. Due to the left-right ambiguity of the drift chamber wire planes, each Cal2 will produce two *impact coordinate* or *impacts* in the plane, symmetrically disposed with respect to the sense wire. This is the entry point for the combinatorials, taken an *impact coordinate* for each plane involved in the calculations. The algorithm first searches for all the Hits with six impacts and with the remaining impacts, searches for Hits with five impacts. The procedure for six impacts is the following:

- Each combination of four impacts in four different layers defines a straight line. Due to the better straight line geometrical definition and the convenience of interpolate instead of extrapolate, impacts from the external planes are used in the fit. The obtained straight line is tested to be between adequate limits in position and slope, to avoid impossible combinations. For the inner MDCs it is posible to test if the straight line points approximately to the target area, looking at the distance from the track to the z-axis (Zdist in the MdcHitFPar container).

- The straight line is extrapolated to the two remaining layers and their cross points determined. A road (Road parameter in MdcHitFPar) or interval in the plane is scanned around each cross point. If there are impacts in a given road around the line in these two layers, a straight line is fitted with the six impacts. The Hit must fulfill a maximum chi-square test before continuing.

- The Hit is tested versus the other posible Hits found sharing some impacts. If they share more impacts than the maximum allowed, only the one with the best χ^2 is kept.

- Once all Hits with six impacts have been found, the used impacts are flagged. To assure that they will not be used later, these impacts and their symmetrical partners are deleted from the list of available impacts for the construction of Hits with five impacts.

- If five impact Hits are searched for, a new procedure begins for the remaining Hits. The combinations of four impacts are extrapolated to a fifth layer. The line is fitted and kept if the fifth impact is found within the road, the obtained χ^2 is good enough and there are less common impacts with other five impact Hits than the maximum allowed. All combinations of five layers are scanned and the used impacts are flagged as used; they and their symmetric impacts are deleted from the list of available impacts for next steps, if any.

4.6 Slope correction to the tracking software

The fit performed inside the Santiago tracking sofware to obtain the track estimates requires the distances to the fired wires in each sense wire plane. The transformation from Cal2 to a coordinate in plane cannot be exactly done without the information of the slopes of the track. Once the Hit is found and a first fit has been performed, a correction in slope must be applied to their impacts, improving then the accuracy of the fit parameters and the quality of the fit in successive steps. [16].The individual track offset can also be determined and the impact coordinates corrected for this offset. To determine the offset in the Santiago tracking scheme, the fit function is called for the final set of obtained Hits, with different offsets in each call.

The distance obtained in each wire after calibration is converted to a coordinate in the wire plane. But, if the track crosses the cell inclined an angle α, the closest point between the track and the wire does not lie in the plane. From figure 4.1 it is immediate to obtain the expression for the

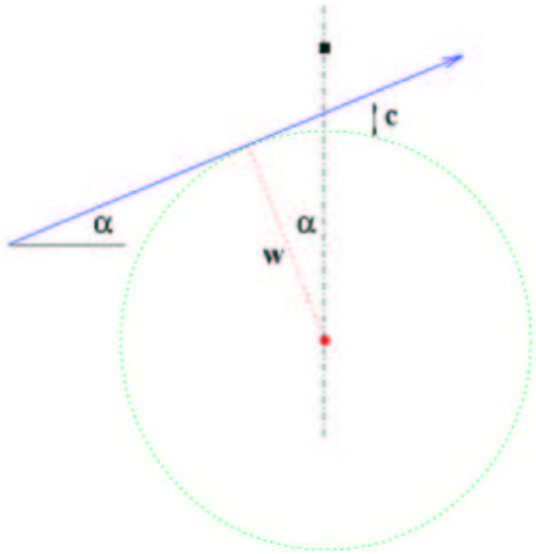

Figure 4.1: *geometrical correction due to the track inclination in the cell.*

correction c of the track impact coordinate in the plane

$$c = \frac{w}{\cos \alpha} - w \tag{4.1}$$

where w is the original impact coordinate. The correction is always positive, increasing slightly the impact coordinate for all cells. To apply this correction it is necessary to know previously the angle α. An estimation of the angle is obtained from a previous track fit, where the transformation is performed without any correction. This iterative approximation converges very fast (one or two steps) for most tracks.

4.7 Fit algorithm

The algorithm is based on the analytical least squares fit of calibrated data Cal3 to a straight line (*hit*) that passes through 4,5 or 6 wire planes .

Starting from the coordinates of candidate *impacts* in each wire plane in the plane's local reference system, we obtain the parameters of *hits* in the chamber's local reference system. These parameters are: the coordinates (x,y) corresponding to the chamber's centre (z = 0), the projected slopes (x', y'), and the covariance matrix of the parameters, as well as the χ^2 of the fit.

The fit function starts from the coordinates in the plane of each one of *impacts* . A local reference system in each wire plane is defined, being the measured coordinate t , perpendicular to the wires. The coordinate direction is defined as an angle α with the negative 'y' axis, measured unclockwise, as it is in figure 1.

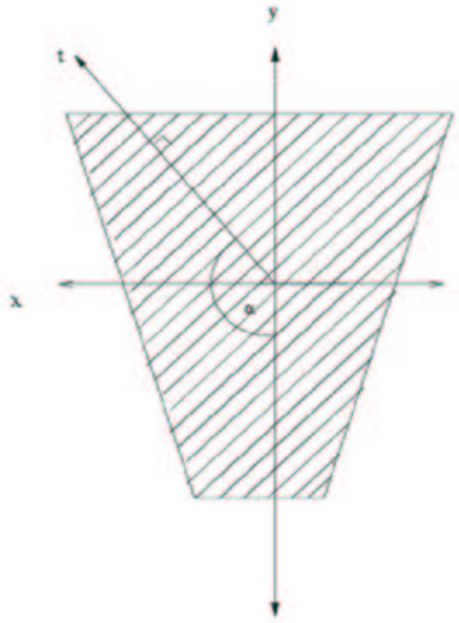

Figure 4.2: Definition of t coordinate

In this way, given the point (x_i, y_i) in the plane i , the coordinate t_i in the local system is given by:

$$t_i = x_i \sin(\alpha_i) - y_i \cos(\alpha_i) \tag{4.2}$$

In the other hand, if the point (x_i, y_i) is the intersection of the straight line to which the hit fits with the plane i then:

$$x_i = x_0 + x'(z_i - z_0) \tag{4.3}$$
$$y_i = y_0 + y'(z_i - z_0) \tag{4.4}$$

Using these equations the coordinate can be written in the plane i as a function of the parameters of the straight line as follow:

$$t_i = \sin(\alpha_i)[x_0 + x'(z_i - z_0)] - \cos(\alpha_i)[y_0 + y'(z_i - z_0)] \tag{4.5}$$

The algorithm starts from four *impacts* in four different wire planes, so that we have 4 equations with 4 variables:

$$t_1 = \sin(\alpha_1)[x_0 + x'(z_1 - z_0)] - \cos(\alpha_1)[y_0 + y'(z_1 - z_0)] \quad (4.6)$$
$$t_2 = \sin(\alpha_2)[x_0 + x'(z_2 - z_0)] - \cos(\alpha_2)[y_0 + y'(z_2 - z_0)] \quad (4.7)$$
$$t_3 = \sin(\alpha_3)[x_0 + x'(z_3 - z_0)] - \cos(\alpha_3)[y_0 + y'(z_3 - z_0)] \quad (4.8)$$
$$t_4 = \sin(\alpha_4)[x_0 + x'(z_4 - z_0)] - \cos(\alpha_4)[y_0 + y'(z_4 - z_0)] \quad (4.9)$$

This can be written in matrix form as:

$$T = M.P$$

$$\begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix} = \begin{pmatrix} \sin(\alpha_1) & -\cos(\alpha_1) & \sin(\alpha_1)(z_1 - z_0) & -\cos(\alpha_1)(z_1 - z_0) \\ \sin(\alpha_2) & -\cos(\alpha_2) & \sin(\alpha_2)(z_2 - z_0) & -\cos(\alpha_2)(z_2 - z_0) \\ \sin(\alpha_3) & -\cos(\alpha_3) & \sin(\alpha_3)(z_3 - z_0) & -\cos(\alpha_3)(z_3 - z_0) \\ \sin(\alpha_4) & -\cos(\alpha_4) & \sin(\alpha_4)(z_4 - z_0) & -\cos(\alpha_4)(z_4 - z_0) \end{pmatrix} \begin{pmatrix} x_0 \\ x' \\ y_0 \\ y' \end{pmatrix} \quad (4.10)$$

The parameters to be determined are: x_0, x', y_0, y'. So we need to find:

$$P = M^{-1}.T \quad (4.11)$$

Let be T_i the coordinate measured in the plane i, with deviation σ_i. We have to find the parameters of the straight line that minimize χ^2, defined as

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{t_i - T_i}{\sigma_i} \right)^2 \quad (4.12)$$

being N the number of planes of the chamber to be fitted. For a given z_0,

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{\sin(\alpha_i)[\hat{x_0} + \hat{x}'(z_i - z_0) - \cos(\alpha_i)[\hat{y}_0 + \hat{y}'(z_i - z_0)] - T_i}{\sigma_i} \right)^2 \quad (4.13)$$

The conditions of minimum are:

$$\begin{aligned}
\frac{\partial \chi^2}{\partial \hat{x}_0} &= 0 \\
\frac{\partial \chi^2}{\partial \hat{x}'} &= 0 \\
\frac{\partial \chi^2}{\partial \hat{y}_0} &= 0 \\
\frac{\partial \chi^2}{\partial \hat{y}'} &= 0
\end{aligned} \tag{4.14}$$

and then we have:

$$\sum_{i=1}^{N} \frac{\sin\alpha_i \left\{[\hat{x}_0 + \hat{x}'(z_i - z_0)]\sin\alpha_i - [\hat{y}_0 + \hat{y}'(z_i - z_0)]\cos\alpha_i - T_i\right\}}{\sigma_i} = 0$$

$$\sum_{i=1}^{N} \frac{(z_i - z_0)\sin\alpha_i \left\{[\hat{x}_0 + \hat{x}'(z_i - z_0)]\sin\alpha_i - [\hat{y}_0 + \hat{y}'(z_i - z_0)]\cos\alpha_i - T_i\right\}}{\sigma_i} = 0$$

$$\sum_{i=1}^{N} -\frac{\cos\alpha_i \left\{[\hat{x}_0 + \hat{x}'(z_i - z_0)]\sin\alpha_i - [\hat{y}_0 + \hat{y}'(z_i - z_0)]\cos\alpha_i - T_i\right\}}{\sigma_i} = 0$$

$$\sum_{i=1}^{N} \frac{-(z_i - z_o)\cos\alpha_i \left\{[\hat{x}_0 + \hat{x}'(z_i - z_0)]\sin\alpha_i - [\hat{y}_0 + \hat{y}'(z_i - z_0)]\cos\alpha_i - T_i\right\}}{\sigma_i} = 0$$

Let us define $S_i \equiv \frac{\sin(\alpha_i)}{\sigma_i}$, $C_i \equiv \frac{\cos(\alpha_i)}{\sigma_i}$ and $Z_i \equiv z_i - z_0$, then:

$$\sum_{i=1}^{N} \left(S_i^2\right)\hat{x}_0 + \sum_{i=1}^{N} \left(S_i^2 Z_i\right)\hat{x}' - \sum_{i=1}^{N} \left(S_i C_i\right)\hat{y}_0 - \sum_{i=1}^{N} \left(S_i C_i Z_i\right)\hat{y}' = \sum_{i=1}^{N} S_i T_i$$

$$\sum_{i=1}^{N} \left(Z_i S_i^2\right)\hat{x}_0 + \sum_{i=1}^{N} \left(Z_i^2 S_i^2\right)\hat{x}' - \sum_{i=1}^{N} \left(Z_i S_i C_i\right)\hat{y}_0 - \sum_{i=1}^{N} \left(Z_i^2 S_i C_i\right)\hat{y}' = \sum_{i=1}^{N} Z_i S_i T_i$$

$$\sum_{i=1}^{N} \left(-C_i S_i\right)\hat{x}_0 - \sum_{i=1}^{N} \left(C_i S_i Z_i\right)\hat{x}' + \sum_{i=1}^{N} \left(C_i^2\right)\hat{y}_0 + \sum_{i=1}^{N} \left(C_i^2 Z_i\right)\hat{y}' = \sum_{i=1}^{N} -C_i T_i$$

$$\sum_{i=1}^{N} \left(-Z_i C_i S_i\right)\hat{x}_0 - \sum_{i=1}^{N} \left(Z_i^2 C_i S_i\right)\hat{x}' + \sum_{i=1}^{N} \left(Z_i C_i^2\right) y_o + \sum_{i=1}^{N} \left(Z_i^2 C_i^2\right)\hat{y}' = \sum_{i=1}^{N} -Z_i C_i T_i$$

Or in matrix form,

$$M'.P = T'$$

$$\begin{pmatrix} \sum_{i=1}^{N} S_i^2 & \sum_{i=1}^{N} S_i^2 Z_i & \sum_{i=1}^{N} -C_i S_i & \sum_{i=1}^{N} -C_i S_i Z_i \\ \sum_{i=1}^{N} S_i^2 Z_i & \sum_{i=1}^{N} S_i^2 Z_i^2 & \sum_{i=1}^{N} -C_i S_i Z_i & \sum_{i=1}^{N} -C_i S_i Z_i^2 \\ \sum_{i=1}^{N} -C_i S_i & \sum_{i=1}^{N} -C_i S_i Z_i & \sum_{i=1}^{N} C_i^2 & \sum_{i=1}^{N} C_i^2 Z_i \\ \sum_{i=1}^{N} -C_i S_i Z_i & \sum_{i=1}^{N} -C_i S_i Z_i^2 & \sum_{i=1}^{N} C_i^2 Z_i & \sum_{i=1}^{N} C_i^2 Z_i^2 \end{pmatrix} \begin{pmatrix} x_0 \\ x' \\ y_0 \\ y' \end{pmatrix} = \begin{pmatrix} t_1{}' \\ t_2{}' \\ t_3{}' \\ t_4{}' \end{pmatrix} \tag{4.15}$$

The error matrix (M'), is matrix of the second derivative of the chisquare function with respect to its free parameters, that is

$$M' = 2 \begin{pmatrix} \sum_{i=1}^{N} S_i^2 & \sum_{i=1}^{N} S_i^2 Z_i & \sum_{i=1}^{N} -C_i S_i & \sum_{i=1}^{N} -C_i S_i Z_i \\ \sum_{i=1}^{N} S_i^2 Z_i & \sum_{i=1}^{N} S_i^2 Z_i^2 & \sum_{i=1}^{N} -C_i S_i Z_i & \sum_{i=1}^{N} -C_i S_i Z_i^2 \\ \sum_{i=1}^{N} -C_i S_i & \sum_{i=1}^{N} -C_i S_i Z_i & \sum_{i=1}^{N} C_i^2 & \sum_{i=1}^{N} C_i^2 Z_i \\ \sum_{i=1}^{N} -C_i S_i Z_i & \sum_{i=1}^{N} -C_i S_i Z_i^2 & \sum_{i=1}^{N} C_i^2 Z_i & \sum_{i=1}^{N} C_i^2 Z_i^2 \end{pmatrix} \tag{4.16}$$

Then, the covariance matrix is given by

$$cov = \frac{1}{2} M'^{-1}$$

The calculated covariance matrix for MDCI in a 6 layers fit, asuming σ_i=σ=80 μ m is the following:

$$\begin{pmatrix} 0.00735371 & 1.05613e\text{-}07 & -7.82983e\text{-}09 & -0.000186562 \\ & 0.00140806 & -7.4119e\text{-}05 & -9.16475e\text{-}09 \\ & & 4.88838e\text{-}05 & 8.03865e\text{-}10 \\ & & & 2.64125e\text{-}05 \end{pmatrix}$$

From this matrix it is posible to get the errors in the x_0, y_0 parameters:

$$\sigma_{x_0} = 85.75\ \mu\text{m}\ \ \sigma_{y_0} = 37.52\ \mu\text{m}$$

4.8 Fit Algorithm including time offset

Drift times obtained from TDC times include several systematic effects. One of them is the time of flight of the particle between the target and the chamber. As this effect is almost the same for all the measured coordinates in the same chamber, it is possible to introduce a common time offset t_{off} in the fit as a new free parameter.

Taken into account the time offset, the coordinate t_i can be written as follows:

$$t_i = \sin\alpha_i[x_0 + x'(z_i - z_0)] - \cos\alpha_i[y_0 + y'(z_i - z_0) + v_d * t_{off}]$$

where v_d, the drift velocity, can be taken constant.

In this case the error matrix M' is given by:

$$M' = 2 \begin{pmatrix} \sum_{i=1}^{N} S_i^2 & \sum_{i=1}^{N} S_i^2 Z_i & \sum_{i=1}^{N} -C_i S_i & \sum_{i=1}^{N} -C_i S_i Z_i & \sum_{i=1}^{N} S_i v_d \\ \sum_{i=1}^{N} S_i^2 Z_i & \sum_{i=1}^{N} S_i^2 Z_i^2 & \sum_{i=1}^{N} -C_i S_i Z_i & \sum_{i=1}^{N} -C_i S_i Z_i^2 & \sum_{i=1}^{N} -C_i v_d \\ \sum_{i=1}^{N} -C_i S_i & \sum_{i=1}^{N} -C_i S_i Z_i & \sum_{i=1}^{N} C_i^2 & \sum_{i=1}^{N} C_i^2 Z_i & \sum_{i=1}^{N} S_i Z_i v_d \\ \sum_{i=1}^{N} -C_i S_i Z_i & \sum_{i=1}^{N} -C_i S_i Z_i^2 & \sum_{i=1}^{N} C_i^2 Z_i & \sum_{i=1}^{N} C_i^2 Z_i^2 & \sum_{i=1}^{N} -C_i Z_i v_d \\ \sum_{i=1}^{N} S_i v_d & \sum_{i=1}^{N} -C_i v_d & \sum_{i=1}^{N} S_i Z_i v_d & \sum_{i=1}^{N} -C_i Z_i v_d & \sum_{i=1}^{N} v_d^2 \end{pmatrix}$$

and then, the covariance matrix is $cov = \frac{1}{2}M^{-1}$ As an example, the covariance matrix in the MDCI for a 6 layers fit, asuming σ_i=σ=80 μ m is:

$$\begin{pmatrix} 0.032539 & 0.00523798 & -0.00027573 & -0.000621646 & -0.283419 \\ & 0.0024974 & -0.000131462 & -9.04952e\text{-}05 & -0.0589437 \\ & & 5.19023e\text{-}05 & 4.764e\text{-}06 & 0.00310281 \\ & & & 3.39287e\text{-}05 & 0.00489617 \\ & & & & 3.18942 \end{pmatrix}$$

The corresponding errors in the x_0, y_0 parameters are:

$$\sigma_{x_0} = 180\ \mu\text{m}\ \ \sigma_{y_0} = 50\ \mu\text{m}$$

4.9 Fit algorithm including the time of flight

The time of flight of a particle coming from the target and going through a MDC chamber is sligthly different for each layer. The time of flight of each layers can be writen in terms of the time of flight of the particle in the physical center of the chamber, and the distance from the target to the layer.

If the remaining systematic effects could be estimated and corrected, the time of flight of the particle at each layer can be introduced in the LSM fit instead of the common time offset. Using the same formalism as before, the covariance matrix results be the same as the one obtained with the time offset algorithm, but there is a factor $(1 + \frac{(k-1)\delta z)}{L})$ multiplying the Time t_{off}. Here k is the number of the layer, δz is the distance between layers, and L is the distance from the target to the physical center of the MDC.

For instance, the covariance matrix in the MDCI for a 6 layers fit, asuming σ_i=σ=80 μ m is:

$$\begin{pmatrix} 0.03 & 0.005 & \text{-2e-04} & \text{-6e-04} & \text{-0.27} \\ & 0.002 & \text{-1e-04} & \text{-8e-05} & \text{-0.05} \\ & & \text{5.e-05} & \text{4.e-06} & 0.002 \\ & & & \text{3.e-05} & 0.005 \\ & & & & 3.1 \end{pmatrix}$$

As can be seen there is no significative improvement when comparing with the results obtained in section 4.6.

4.10 Performance of the tracking

The performance of the tracking algorithm in the reconstruction of dileptons has been studied using a GEANT simulation of electrons and positrons with an uniform momentum distribution. The data includes smearing, time offset and 100% efficiency of MDCs.

The results using both methods, 4 parameter fit (x_0, y_0, x', y') and the 5 parameter fit $(x_0, x_0, x'.y', t_{off})$, are sumarized in the table 4.1. Also in the following pictures can be seen the residuals obtained with the 5-parameters fit for the positrons sample in MDCI.

Particle	MDC	N. of params fitted	σX	$\bar{x}$	σY	$\bar{y}$
e^-	I	4	120	1.4	61	-5.8
	I	5	111	0.4	59	-5.7
	II	4	129	2.3	63	3.1
	II	5	118	1.7	62	3.1
	III	4	145	3.5	72	11.6
	III	5	126	5.7	65	10.7
	IV	4	153	0.8	60	1.9
	IV	5	157	5.4	72	-2.5
e^+	I	4	120	1.4	62	-6.5
	I	5	123	1.6	61	-5.1
	II	4	130	-2.8	63	-0.9
	II	5	118	-0.9	61	-0.4
	III	4	144	1.7	73	-15
	III	5	134	-0.3	62	-13
	IV	4	153	-1.	69	-6
	IV	5	147	-4.8	67	-4.2

Table 4.1: *Sumary of residuals with 4-parameter and 5-parameter fit algorithms in all MDCs for* e^+/e^-

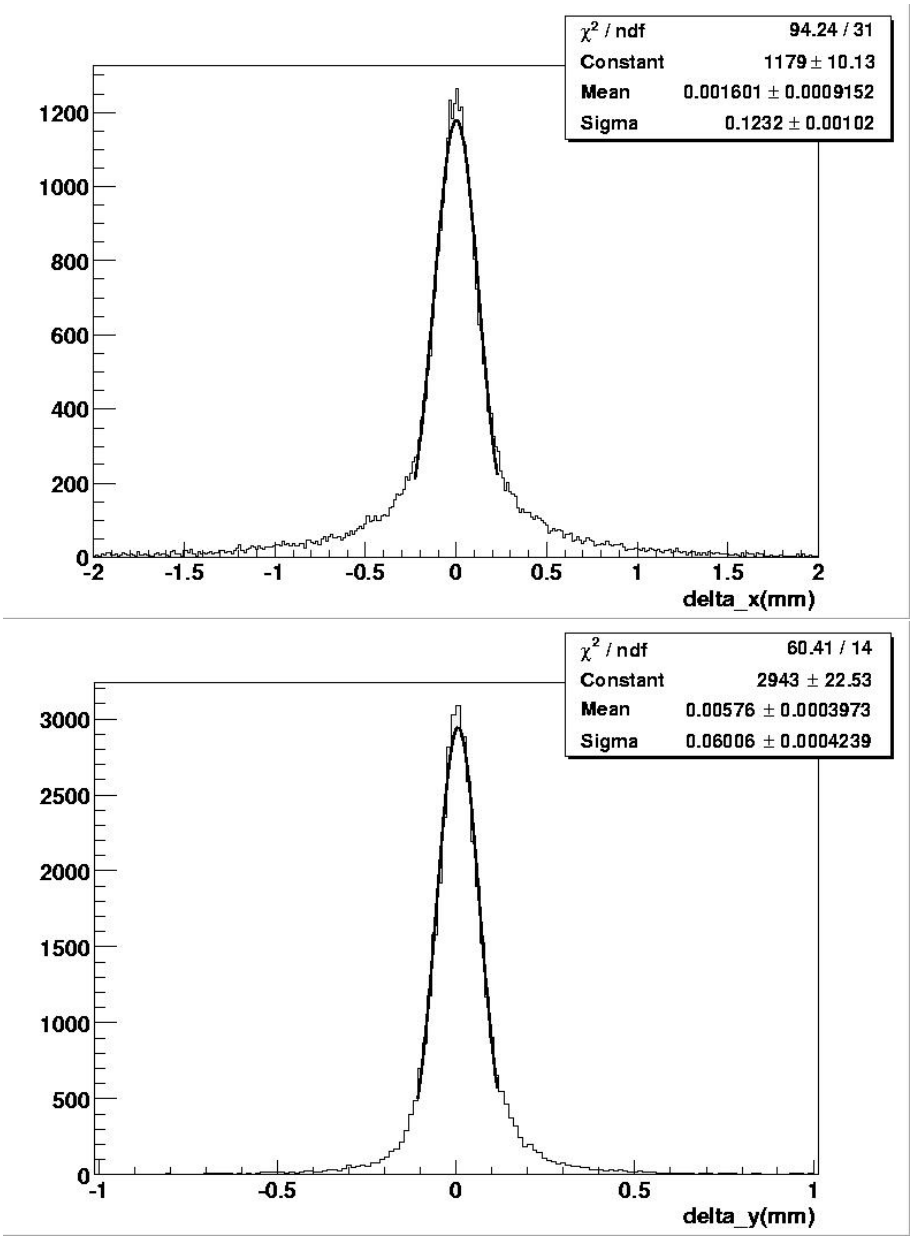

Figure 4.3: Residuals in X and Y for e^+ in MDCI

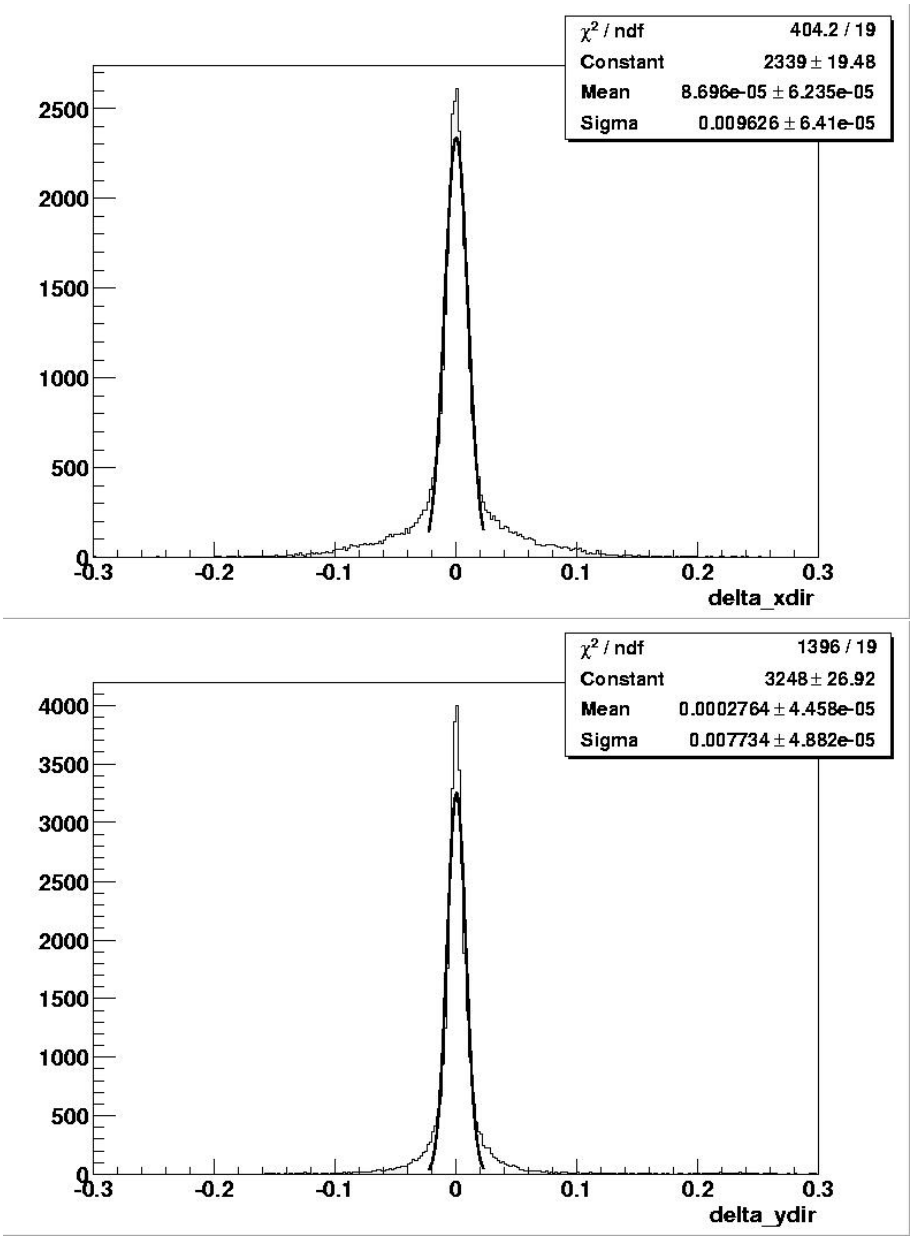

Figure 4.4: Residuals in XDir and YDir for e^+ in MDCI

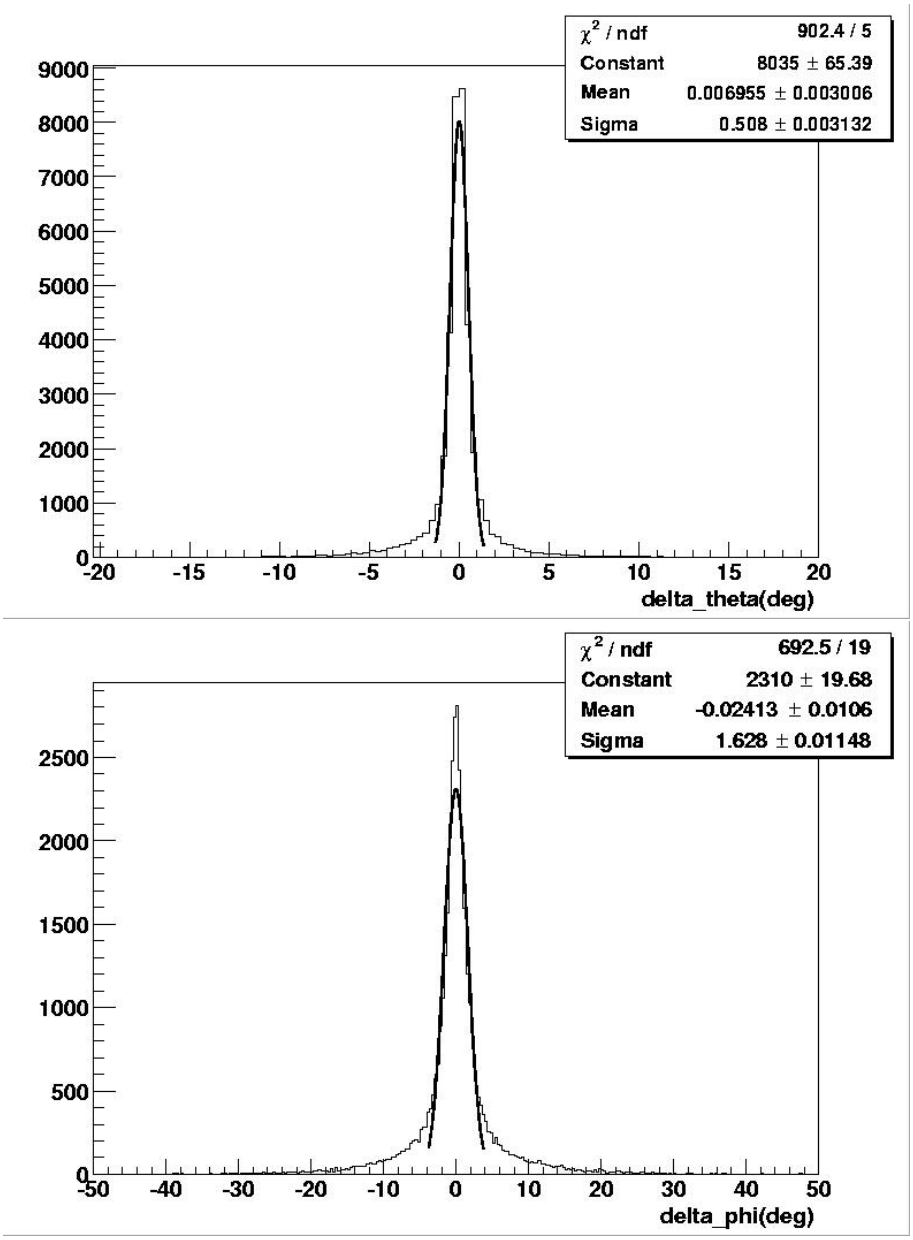

Figure 4.5: Residuals in Theta and Phi for e^+ in MDCI

4.11 Santiago Tracking Quality Control Program (QCP)

It has been developed a set of C++ classes that can be used to perform the Santiago tracking quality assestment [17] This QCP program allows the user to make 2D-plots and resume plots of the residuals and the χ^2 with differnt cuts in momentum and angle (θ and ϕ) in order to study posible systematic errors.

The following plots shows some examples of the pictures that can be obtained with the QCP. This data correspond to a GEANT sumulation of protons and π^+ in half a sector, with an uniform distribution in $cos\theta$, so that it will be the same ammount of particles in the whole chamber. The data includes smearing,TOF, noise and wireOffset.

The table 4.2 shows the summary of the fit algorithms for different settings.

Particle	Params fitted	TOF	Smearing	σX	$\bar{x}$	σY	$\bar{y}$
p	4	no	no	111	0	42	0
	4	yes	no	175	19	92	13
	5	yes	no	230	3	80	10
	4	yes	yes	255	12	104	15
	5	yes	yes	432	10	132	17
π^+	4	no	no	105	0	54	0
	4	yes	no	164	15	70	17
	5	yes	no	202	6	71	15
	4	yes	yes	205	16	78	17
	5	yes	yes	313	13	112	14

Table 4.2: *Sumary of residuals with 4-parameter and 5-parameter fit algorithms in MDCI for p and π+*

The following pictures are examples of the outputs that can be obtained with the QCP; this data corresponds to the 5-parameter fit.

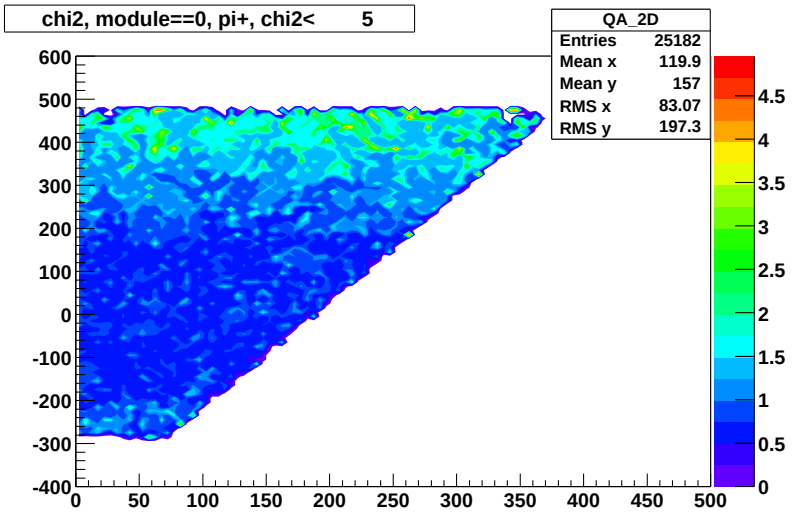

Figure 4.6: 2D plot of χ^2 distribution in MDCI

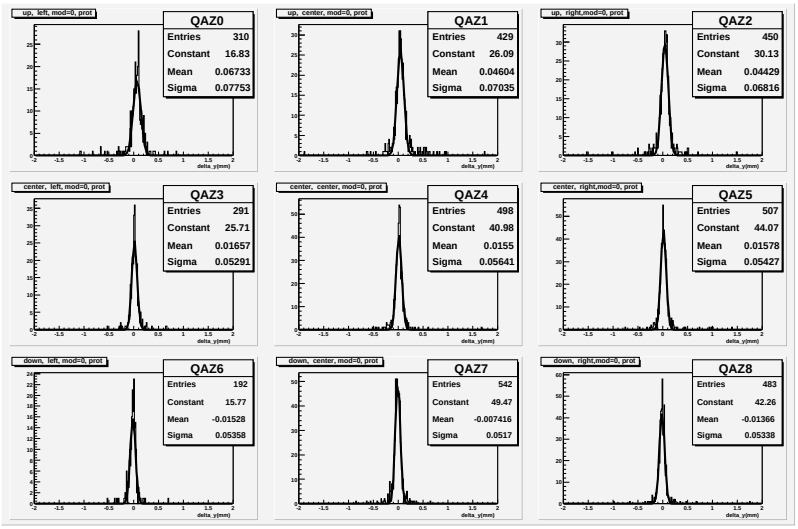

Figure 4.7: Residuals in Y in diferents zones of MDCI

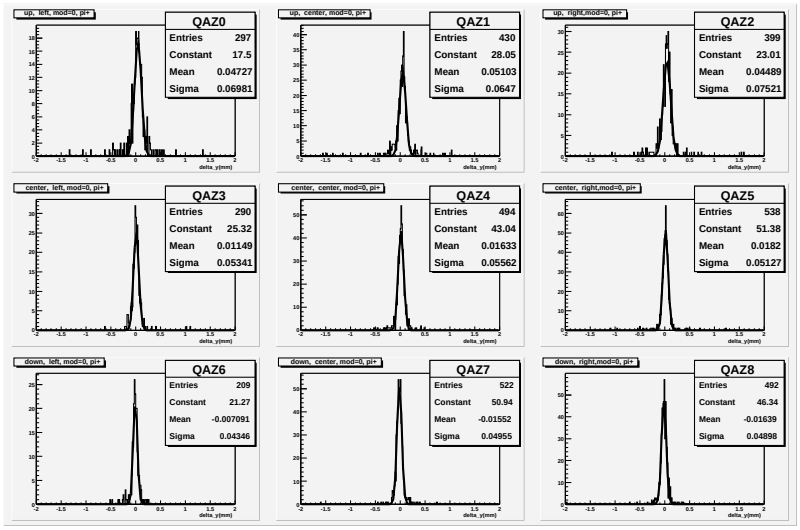

Figure 4.8: ΔY for fast π^+ in MDCI

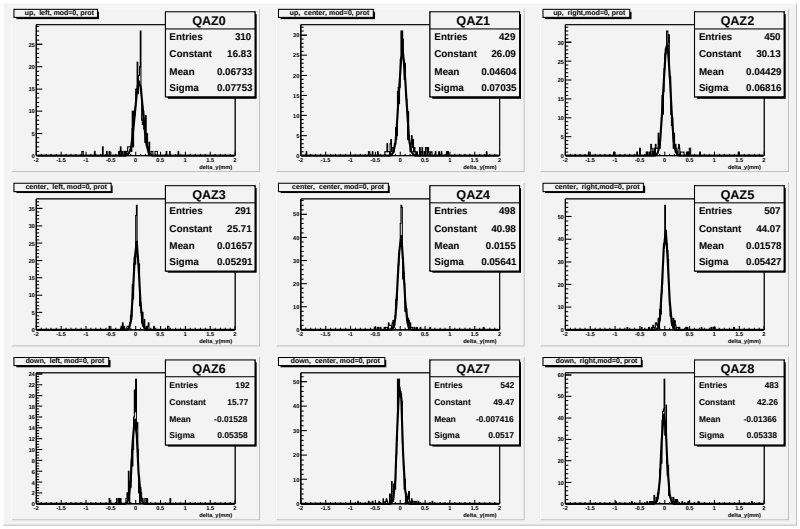

Figure 4.9: ΔY for fast protons in MDCI

Conclusions

The problem of finding an efficient and accurate method to reconstruct particle tracks starting from perpendicular drift distances from measured drift times remains an important one in experiments using drift chambers. The objective of the present work has been the study and developing of track reconstruction algorithms for the HADES MDC chambers.

As a consequence of this study it was needed to review the calibration methods of drift time and drift distance and the use of the Garfield program.

Two methos of track reconstructions has been present, both based on the analytical fit of the tracks by Least Squares Method. The first one allows to fit the tracks starting from 4-parameters (x_0, y_0, x', y') which corresponds to the coordinates and slopes of the hit in the physical center of each MDC chamber. The second one introduces a fifth parameter, the time offset (t_{off}), an aditional time included in the time registered by the TDCs, due to the diference in time of flight of the particles.

The structure of the code and the parameter containers has been adapted to the official ones to be compatible with the last version of HYDRA.

The methods presented in this work allows to get resolutions in MDC chambers under the desing values. The *Santiago tracking algorithm* may be very useful for the HADES tracking system, not only for the resolution that can be reach in the reconstruction of tracks but also because it is flexible to reconstruct tracks coming from target or not, allowing to reconstruct secondary tracks and specially in cosmic ray meararements for aligment. Also the methods presented above, with slight modifications, can easily be used with other drift chambers.

With further study, more efficient methods may be found that are

based on the algorithms discussed in this work. Specifically we suggest to introduce an iterative process in cal2 calibration including the 5-parameter fit algorithm in order to take off the time offset introduced in Cal1 data level, because in the present work we have used an average drift velocity in the 5-parameter fit, but this aproximation do not improve the results when comparing with the 4-parameter fit. Probably an iterative process of fit and recalibration of Cal2 data will produce better results.

Bibliography

[1] H. Neumann *et al*; Acta P.P.B **44** (1994) 195

[2] P. Salabura. Acta P. P. B **27** (1996) 421.

[3] Donald H. Perkins. *Introduction to High Energy Physics.* Addison-Wesley, Reading, Massachusetts, Second edition, 1982.

[4] Hansjorg Fischle, Joachim Heintze, and Bernhard Schmidt. *Experimental determination of ionization cluster size distributions in counting gases.* Nuclear Instruments and Methods in Physics Research, A301:202-214, 1991.

[5] W. Blum, L. Rolandi. *Particle Detection with Drift Chambers*, Springer Verlag,New York, 1993

[6] Groom *et al.* , Particle Data Group. *Review of Particle Physycs*, Eur. Phys. J. C **15** 1-878 (2000)

[7] Rob Veenhof. Garfield, a drift-chamber simulation program: User's guide.CERN

[8] F. Bruyant, J.M. Lesceux, R. J. Plano Nucl. Ins. Meth. **176** (1980) 409-413

[9] C. Garabatos *et al.* Nucl. Ins. Meth. A **412** (1998) 38-46

[10] C. Müntz, Nuclear Physics B **78**, (1999), 139-144

[11] C. Müntz; *Results from MDC Prototype-1 Analysis - Test Beam April'97*, MDC Internal report, October 1997

[12] William H. Press, Brian Flannery, Saul Teukolsky, and William Vetterling.
Numerical Recipes in C. Cambridge University Press, New York, second edition, 1992.

[13] M. Sanchez; *Diseño y programación orientados a objetos de la reconstrucción de sucesos en el experimento HADES de colisiones nucleo-nucleo*, Tesina, Univ. Santiago de Compostela, 1999

[14] R. Brun and Fons Rademakers; *ROOT - An Object Oriented Data Analysis Framework*, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996

Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86

See also http://root.cern.ch/

[15] *GEANT, Detector Description and Simulation Tool*, http://wwwinfo.cern.ch/asd/geant/index.html

[16] H. Alvarez P. PhD Thesis. Univesidad de Santiago de Compostela, 2002

[17] Diego Gonzalez. Santiago tracking quality control. Not published.

Appendix A

A.1 Santiago tracking macro for simulation

```
{
//********************** Configuration area ********************************
Int_t mdcMods[4] = {1,1,1,1};       //Bitmap of active modules
Int_t sectors[6] = {1,1,1,1,1,1};   //Bitmap of active sectors
TStrinf inDir    = "myDir"
TString inFile   = "input.root";  //Input file
TString outFile  = "output.root";  //Output file
//A maximum of two parameter files is accepted. The string "oracle" can be used
//instead of a parameter file name for initialization from Oracle.
TString parFile1 = "oracle";
TString parFile2 = "mdcSPar.par";
//****************** Advanced configuration ******************************
Int_t splitLevel = 2;

//************************************************************************
//** Macro code                          ****************************
//** You should not need to change this ****************************
//************************************************************************
HLocation aLoc; //Used to conf. the data source and segment finders
aLoc.set(2,0,0);

HMdcDetector *mdc=new HMdcDetector;
for (Int_t sector=0;sector<6;sector++)
  if (sectors[sector] == 1) mdc->setModules(sector,mdcMods);
gHades->getSetup()->addDetector(mdc);
```

```
//Set data source
HDataSource *dataSource=0;
if (inFile.Contains(".root")) {         //Root input file
  HRootSource *datos=new HRootSource;
  datos->setDirectory(inDir.Data());
  datos->addFile(inFile.Data());
  dataSource = datos;
} else if (inFile.Contains(".hld")) {   //List mode data file
  HldFileSource *datos = new HldFileSource;
  datos->setDirectory(inDir.Data());
  datos->addFile(inFile.Data());
  dataSource = datos;
} else cerr << "Input data suffix not known" << endl;
gHades->setDataSource(datos);

//Set split level
gHades->setSplitLevel(splitLevel);

//Set runtime database
HRuntimeDb* rtdb=gHades->getRuntimeDb();
if (parFile1=="oracle") {
  HParOraIo *ora = new HParOraIo;
  ora->open();
  if (!ora->check()) {
    printf("No connection to Oracle available\n");
    return -1;
  }
  rtdb->setFirstInput(ora);
} else if (parFile1.Contains(".root")) {
  HParRootFileIo *inroot = new HParRootFileIo;
  inroot->open(parFile1.Data());
  rtdb->setFirstInput(inroot);
} else {
  HParAsciiFileIo *input1 = new HParAsciiFileIo;
  input1->open(parFile1.Data(),"in");
  rtdb->setFirstInput(input1);
}

if (parFile2!="") {
  HParAsciiFileIo *input2=new HParAsciiFileIo;
```

```
  input2->open(parFile2.Data(),"in");
  rtdb->setSecondInput(input2);
}

//Set tasks
HTaskSet *simulTasks=gHades->getTaskSet("simulation");
HMdcDigitizer *digitizer=0; Text_t digiCad[]="mdc.digi";
HMdcHitFSim *finder=0;      Text_t findCad[]="mdc.hitf";
HMdcSegmentF *segFinder=0;  Text_t segFindCad[]="mdc.segf";
HMdcHitComp *comp=0;
HMdcCalibrater2Sim *calibrater=0; Text_t calibCad[]="mdc.calib";
Text_t buffer[255];
HReconstructor *antTask=0;

//Add digitizer
digitizer=new HMdcDigitizer(digiCad,"Mdc digitizer");
simulTasks->connect(digitizer);
antTask=digitizer;

//Add calibrater2
calibrater = new HMdcCalibrater2Sim(calibCad,"Mdc calibrater");
simulTasks->connect(calibrater,antTask);
antTask=calibrater;

for (Int_t sector = 0; sector<6; sector++) { //For each sector
  if (sectors[sector] == 1) { //Sector is activated
    aLoc[0]=sector;

    for (Int_t i=0;i<4;i++) { //Add Finders
      if (mdcMods[i] != 0) {
sprintf(buffer,"%s%i.%i",findCad,sector,i);
finder=new HMdcHitFSim(buffer,"Mdc Hit finder");
aLoc[1]=i;
finder->setLoc(aLoc);
simulTasks->connect(finder,antTask);
antTask=finder;

  sprintf(buffer,"mdc.hitcomp%i.%i",sector,i);
  comp=new HMdcHitComp(buffer,"Mdc Hit Comp");
  aLoc[1]=i;
```

```
  comp->setLoc(aLoc);
  simulTasks->connect(comp,antTask);
  antTask=comp;
      }
    }
    sprintf(buffer,"%s%i",segFindCad,sector);
    segFinder=new HMdcSegmentF(buffer,"Mdc Segment finder");
    aLoc[1]=0;
    segFinder->setLoc(aLoc);
    simulTasks->connect(segFinder,antTask);
    antTask=segFinder;
  }
}

//Initialize the system
if (!gHades->init()) {
  printf("Error during initialization\n");
  rtdb->closeFirstInput();
  rtdb->closeSecondInput();
  return 1;
}
//Configure output
gHades->setOutputFile(outFile.Data(),"RECREATE","Test",1);
gHades->makeTree();
}
```

A.2 Santiago tracking macro for real data

```
{
//Standard macro to analyze real events
//************************ Configuration area **********************
Int_t mdcMods[4] = {1,1,1,1};             //Bitmap of active modules
Int_t sectors[6] = {1,1,1,1,1,1};         //Bitmap of active sectors
Bool_t doSegmentMatching = kFALSE;        //Add segment matcher (MDC 1,2 needed)
TString inputDir = "Dir";      //Input Directory
TString inFile   = "inputFile.hld";         //Input file
TString outFile  = "outputFile.root";           //Output file
//A maximum of two parameter files is accepted. The string "oracle" can be used
//instead of a parameter file name.
TString parFile1 = "oracle";
```

```
TString parFile2 = "mdcSPar.par";  //Tracking parameters
//****************** Advanced configuration **************************
Int_t splitLevel = 2;

//*********************************************************************
//** Macro code                          ******************************
//** You should not need to change this ******************************
//*********************************************************************
HLocation aLoc; //Used to conf. the data source and segment finders
aLoc.set(2,0,0);

HMdcDetector *mdc=new HMdcDetector;
for (Int_t sector=0;sector<6;sector++)
  if (sectors[sector] == 1) mdc->setModules(sector,mdcMods);
gHades->getSetup()->addDetector(mdc);

//Set data source
HDataSource *dataSource=0;
if (inFile.Contains(".root")) {         //Root input file
  HRootSource *datos=new HRootSource;
  datos->setDirectory(inputDir.Data());
  datos->addFile(inFile.Data());
  dataSource = datos;
} else if (inFile.Contains(".hld")) {   //List mode data file
  HldFileSource *source = new HldFileSource;
  source->addUnpacker(new HMdcUnpacker(513,kFALSE));
  gHades->setDataSource(source);
  source->setDirectory(inputDir.Data());
  source->addFile(inFile.Data());
  dataSource = source;
} else cerr << "Input data suffix not known" << endl;
gHades->setDataSource(source);

//Set split level
gHades->setSplitLevel(splitLevel);

//Set runtime database: FIXME
HRuntimeDb* rtdb=gHades->getRuntimeDb();
if (parFile1=="oracle") {
  HParOraIo *ora = new HParOraIo;
```

```
  ora->open();
  if (!ora->check()) {
    printf("No connection to Oracle available\n");
    return -1;
  }
  rtdb->setFirstInput(ora);
} else if (parFile1.Contains(".root")) {
  HParRootFileIo *inroot = new HParRootFileIo;
  inroot->open(parFile1.Data());
  rtdb->setFirstInput(inroot);
} else {
  HParAsciiFileIo *input1 = new HParAsciiFileIo;
  input1->open(parFile1.Data(),"in");
  rtdb->setFirstInput(input1);
}

if (parFile2!="") {
  HParAsciiFileIo *input2=new HParAsciiFileIo;
  input2->open(parFile2.Data(),"in");
  rtdb->setSecondInput(input2);
}

//Set tasks
HTaskSet *realTasks=gHades->getTaskSet("real");
HMdcCalibrater1 *calibrater1=0; Text_t calib1Cad[]="mdc.calib1";
HMdcCalibrater2 *calibrater2=0; Text_t calib2Cad[]="mdc.calib2";
HMdcHitFSim *finder=0;      Text_t findCad[]="mdc.hitf";
HMdcSegmentF *segFinder=0;  Text_t segFindCad[]="mdc.segf";
Text_t buffer[255];
HReconstructor *antTask=0;

//Add calibrater1
calibrater1 = new HMdcCalibrater1(calib1Cad,"Mdc calibrater");
realTasks->connect(calibrater1,antTask);
antTask=calibrater1;

//Add calibrater2
calibrater2 = new HMdcCalibrater2(calib2Cad,"Mdc calibrater");
realTasks->connect(calibrater2,antTask);
antTask=calibrater2;
```

```
for (Int_t sector = 0; sector<6; sector++) { //For each sector
  if (sectors[sector] == 1) { //Sector is activated
    aLoc[0]=sector;

    for (Int_t i=0;i<2;i++) { //Add Hit Finders
      if (mdcMods[i] != 0) {
sprintf(buffer,"%s%i.%i",findCad,sector,i);
finder=new HMdcHitF(buffer,"Mdc Hit finder");
aLoc[1]=i;
finder->setLoc(aLoc);
realTasks->connect(finder,antTask);
antTask=finder;
      }
    }
    if (doSegmentMatching) { //Add segment finders
      sprintf(buffer,"%s%i",segFindCad,sector);
      segFinder=new HMdcSegmentF(buffer,"Mdc Segment finder");
      aLoc[1]=0;
      segFinder->setLoc(aLoc);
      realTasks->connect(segFinder,antTask);
      antTask=segFinder;
    }
  }
}

//Initialize the system
if (!gHades->init()) {
  printf("Error during initialization\n");
  rtdb->closeFirstInput();
  rtdb->closeSecondInput();
  return 1;
}
//Configure output
gHades->setOutputFile(outFile.Data(),"RECREATE","Test",1);
gHades->makeTree();
}
```