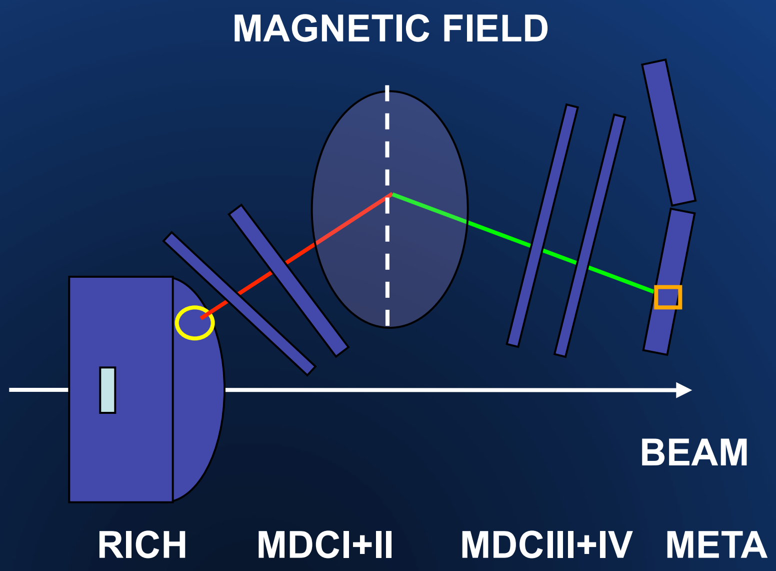
Tracking tutorial

Jochen Markert

Correlation of detector hits



① The track segments of **inner** and **outer** MDCs are matched on Cluster level

② **Outer segments** are matched with **META hits**

③ **Inner segments** are matched to **RICH hits**

Geometry of the Tracking System

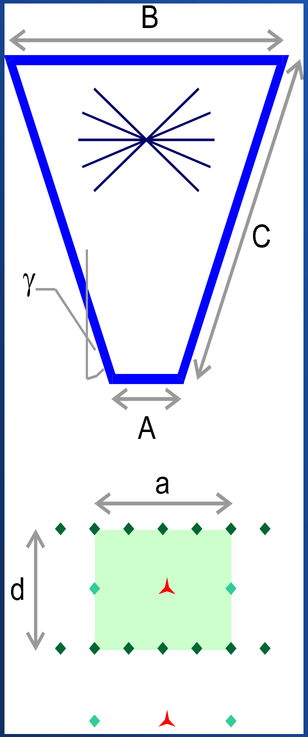
	A [mm]	B [mm]	C [mm]	a [mm]	γ [degree]	R [mm]	d [mm]
I	139,21	767,38	839,19	5	21,98	543,83	5
II	205,00	905,00	1049,27	6	19,49	705,19	5
III	310,43	1804,80	2139,05	12	20,44	1347,69	8
IV	345,46	2224,05	2689,04	14	20,44	1641,68	10

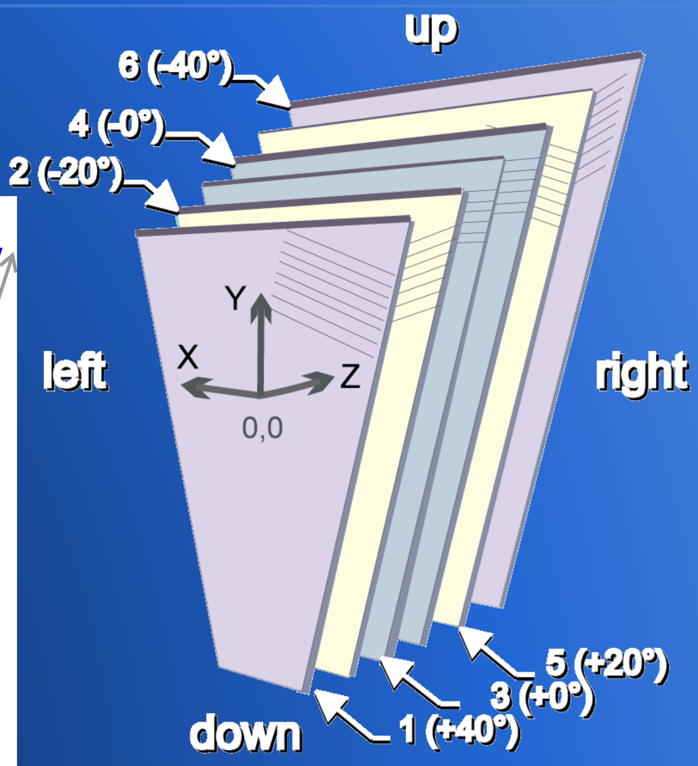
① 24 conceptually identical modules in 4 different geometries

② 6 drift cell layers in each module orientation optimised with respect to resolution in direction of the kick angle

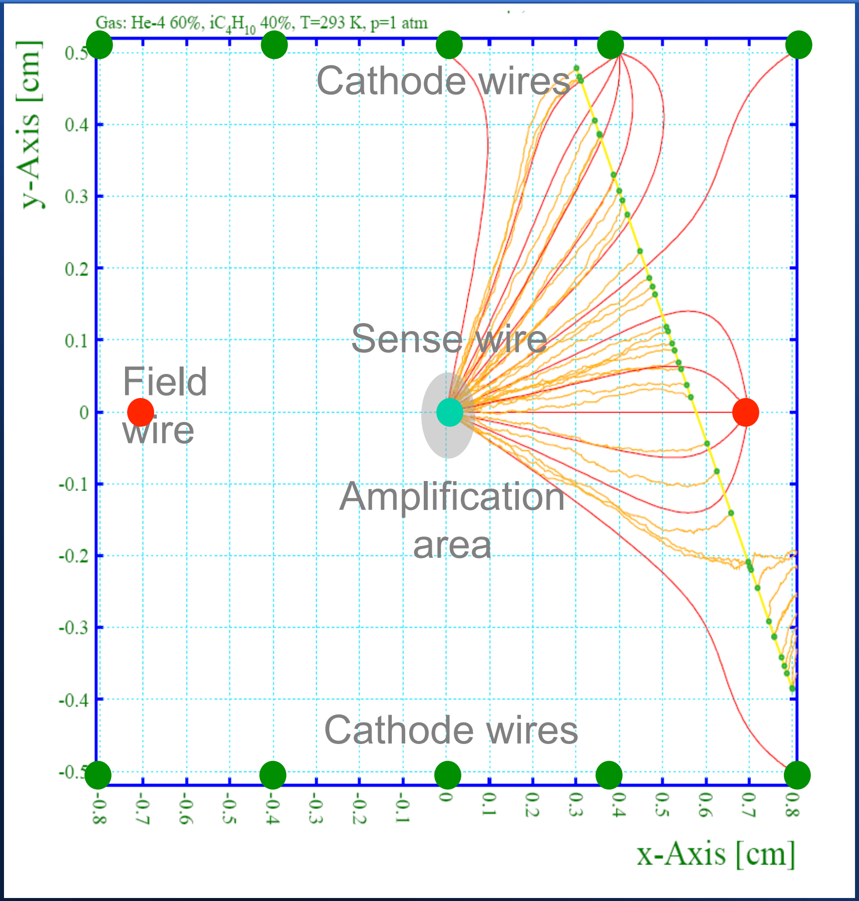
③ ≈ 190 cells per layer sufficient granularity (max multiplicity = 0.6 hits/cm along y)

④ 26 200 cells in total



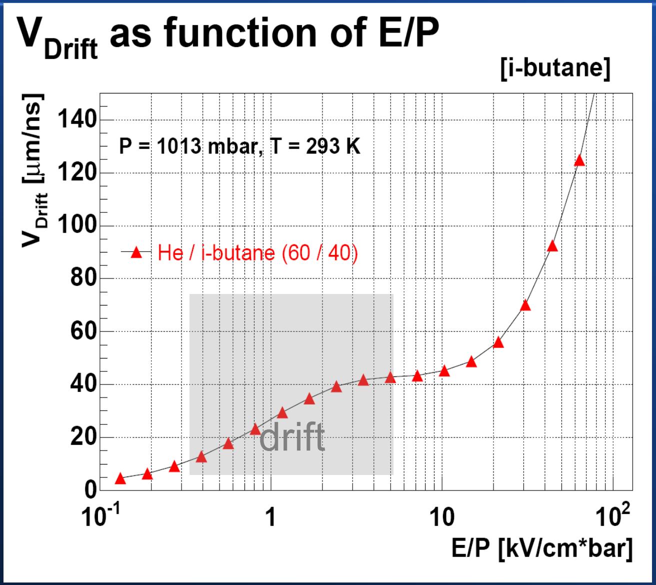


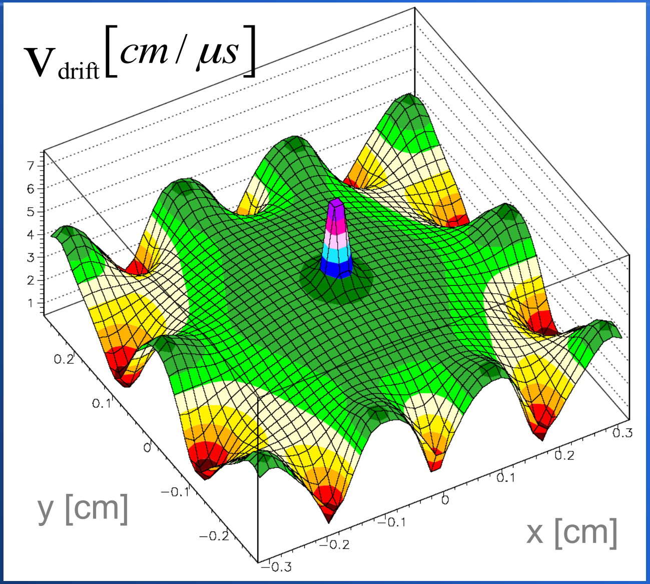
The MDC drift cell



① dimension of the drift cell der 5x5 - 10x14 mm^2
② Gasmixture He/i-Butan , Ar/CO2
③ Simulation of the cell using **GARFIELD** - geometry, field, drift
④ **MAGBOLZ** - gas properties
⑤ **HEED** - primary ionization

Simulation with GARFIELD





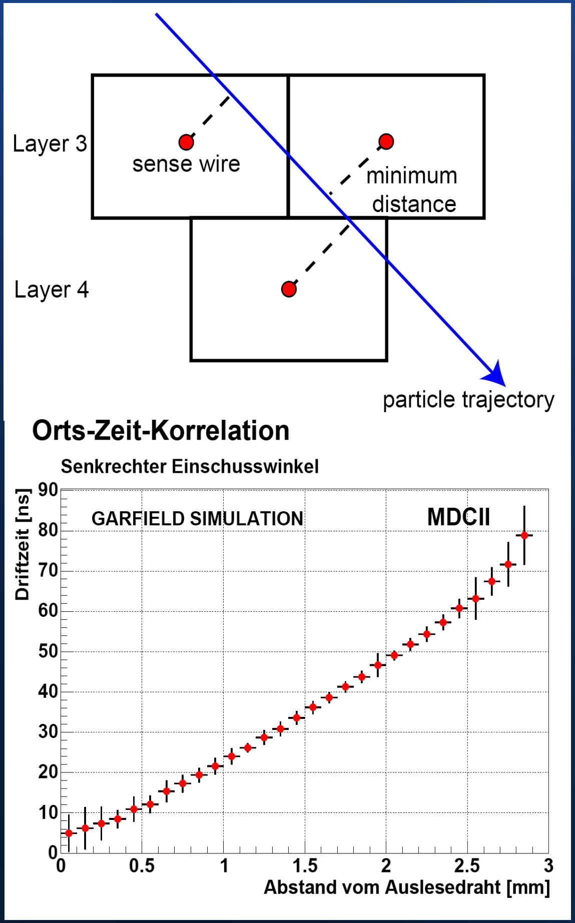
Simulation:

① Inhomogeneous electric field

② V_{Drift} depends on electric field

③ $\Rightarrow$ inhomogenous V_{Drift}

drift time $\longrightarrow$ distance from wire

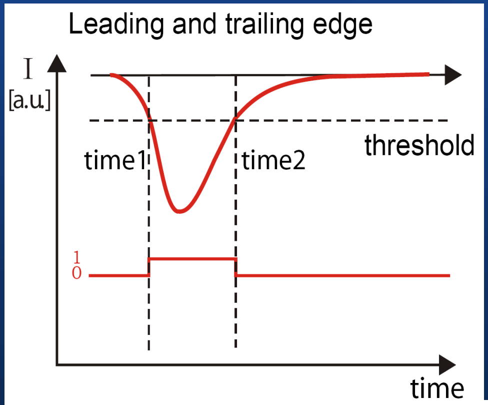


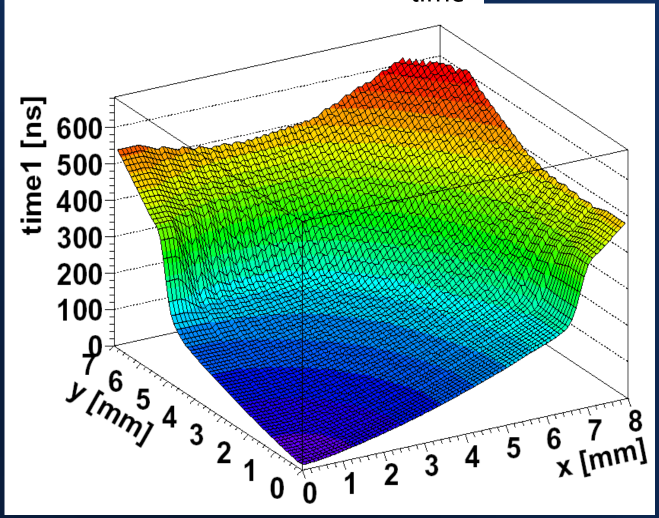
① The track reconstruction needs space coordinates, but MDC measures drift times

$\Downarrow$

② Correlation between drit time and distance from sense wire has to be precisely known

xt- Korrelation





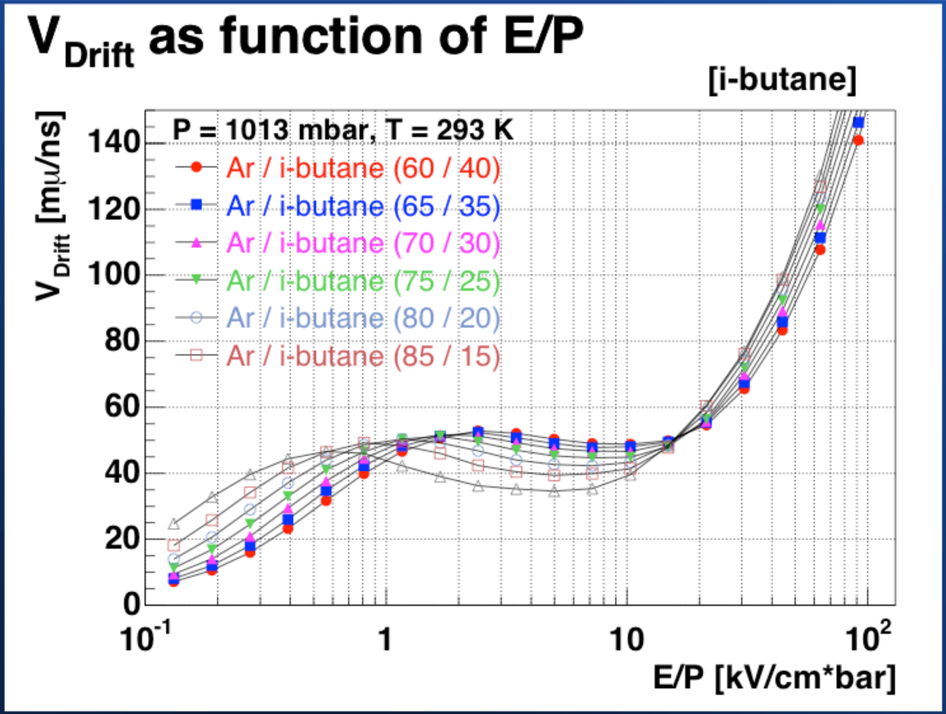
I. **2-dimensional drift cell model**:

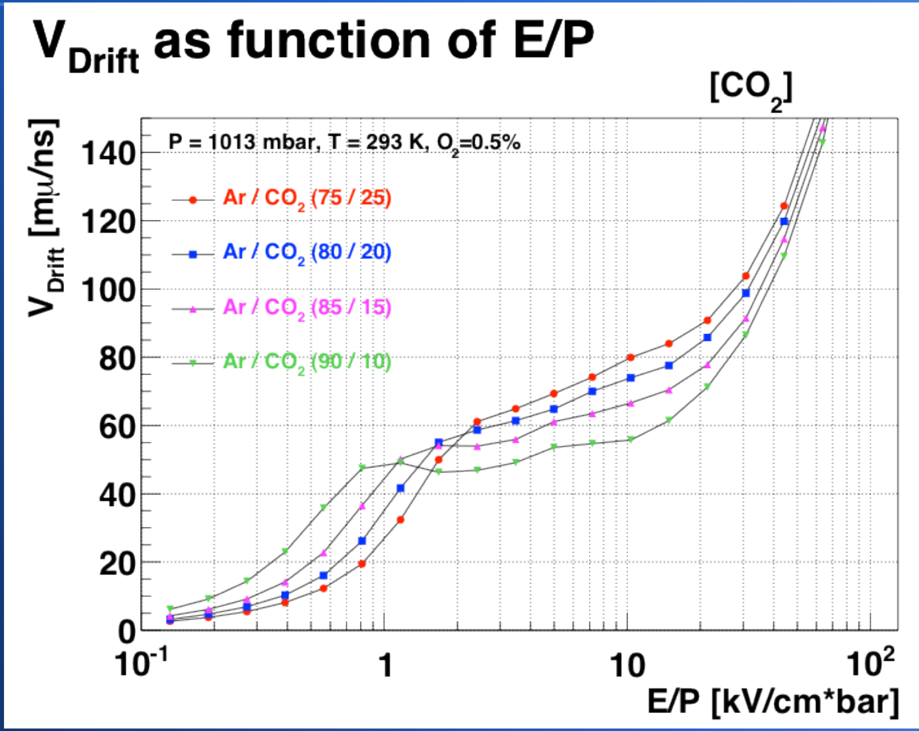
   ① Simulation of drift signals using GARFIELD

   ② Parametrization using impact angle of the particle track into the drift cell and minimum distance from wire

II. Implementation into track reconstruction and GEANT - Simulation

Drift Velocity vs. Electric Field



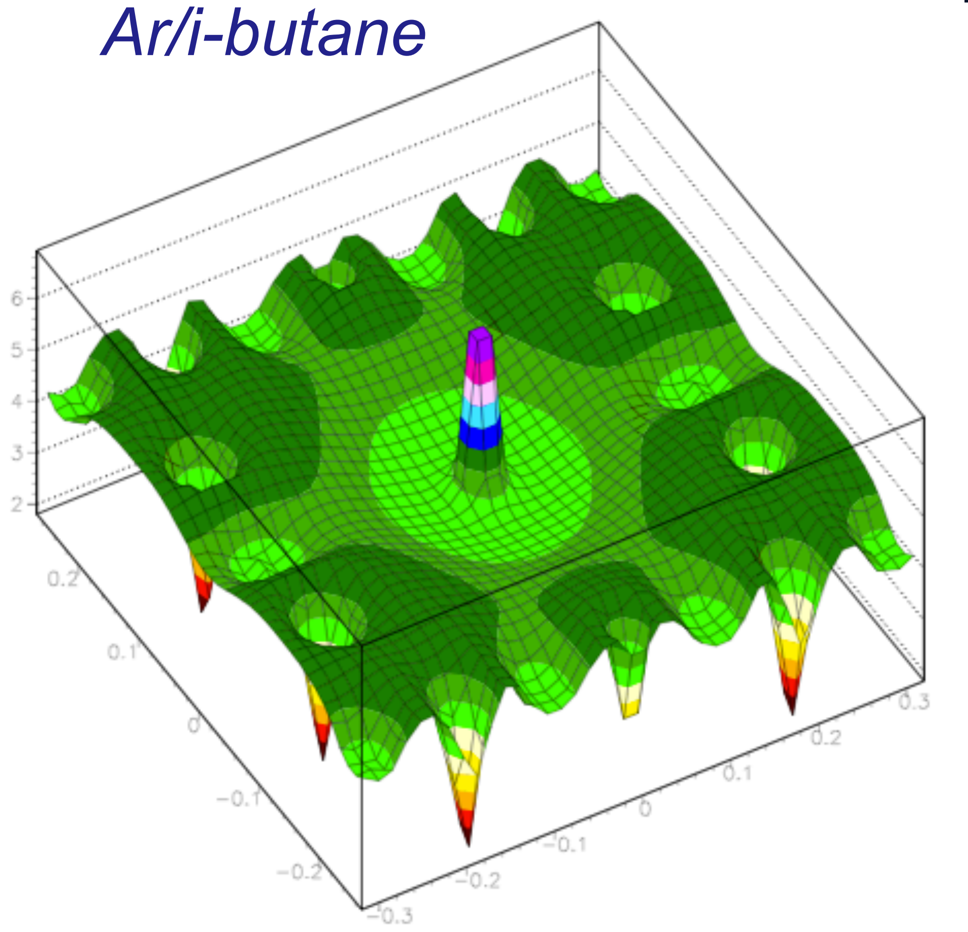


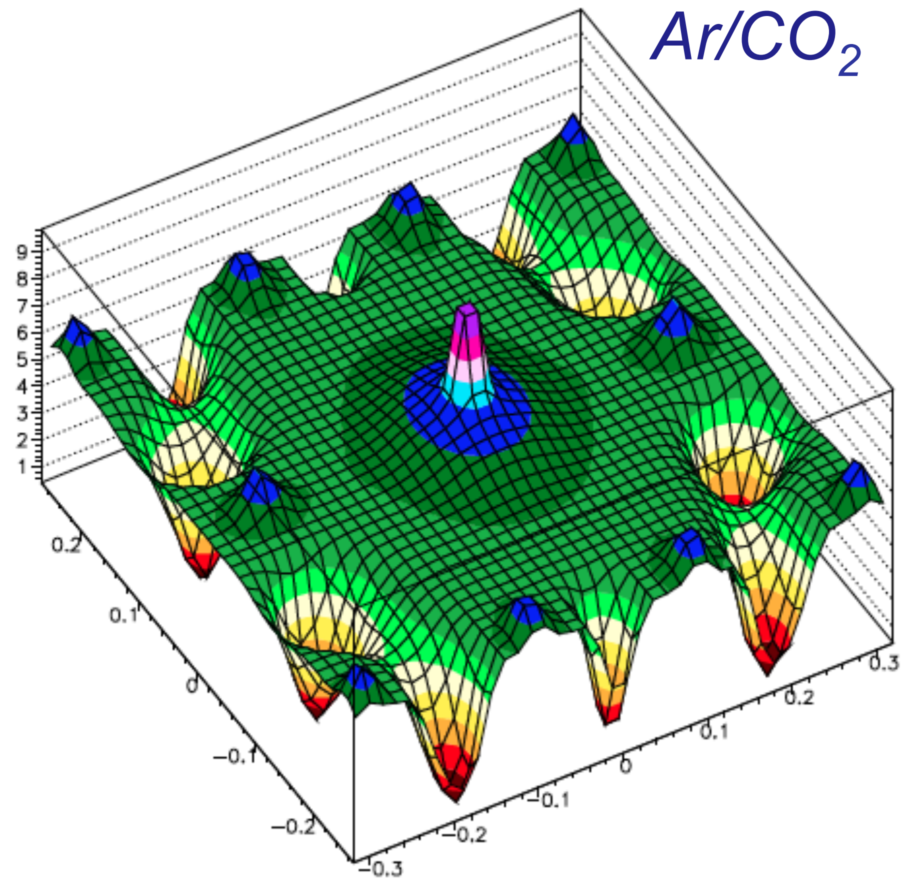
① Drift velocity of Ar/CO_2 higher compared to Ar/i-butane

② No real plateau for Ar/CO_2

③ Drift velocity very low at low electric field values for Ar/CO_2

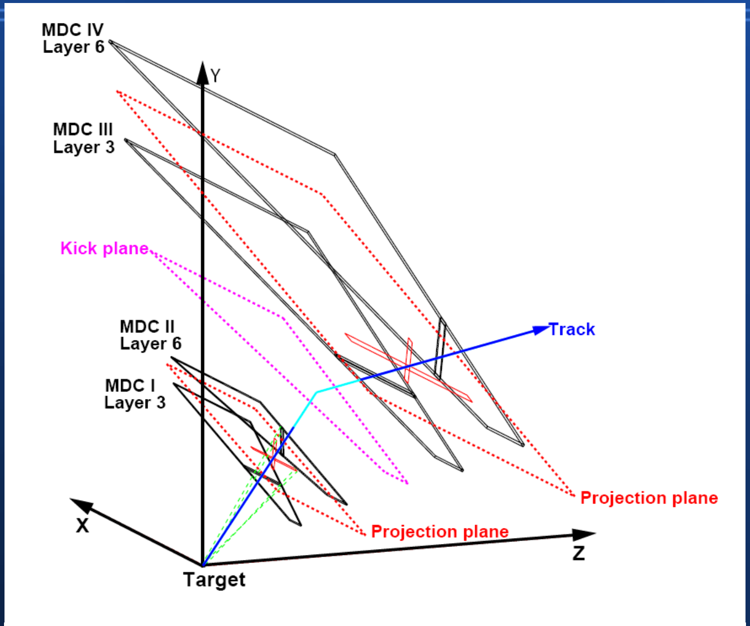
Drift Velocity Contours for MDCI

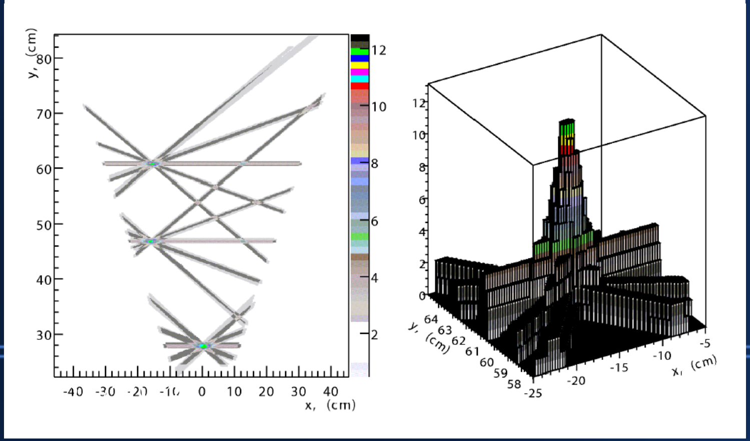




- Strong inhomogeneous drift velocity contour for Ar/CO_2

Tracking in MDCs

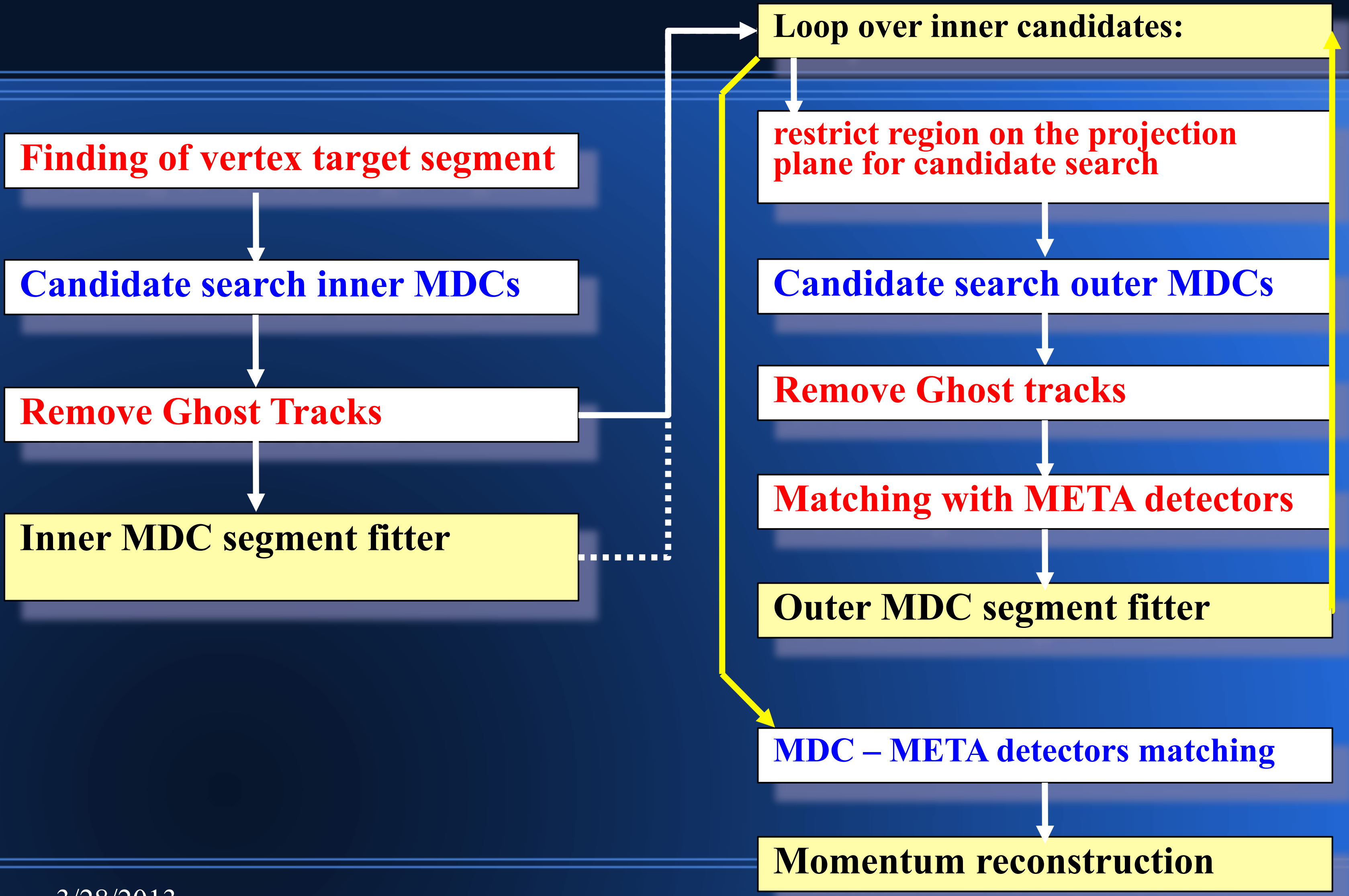




1. Target segment reconstruction
2. projection of drift cells with respect to the target provides Cluster in inner segment
3. projection of outer drift cells with respect to hit point of inner segment on the kickplane defines Cluster in outer segment
4. Fitting of inner and outer Cluster by a straight line model function using the measured drift times (Least square minimization of the functional below)

$$F = \sum_i \frac{(t^i_{drift} + t_{off} - t^i_{TDC})^2}{(\Delta^i_{TDC})^2} \cdot w_i$$

Real life: Current Scheme of the Candidate Search

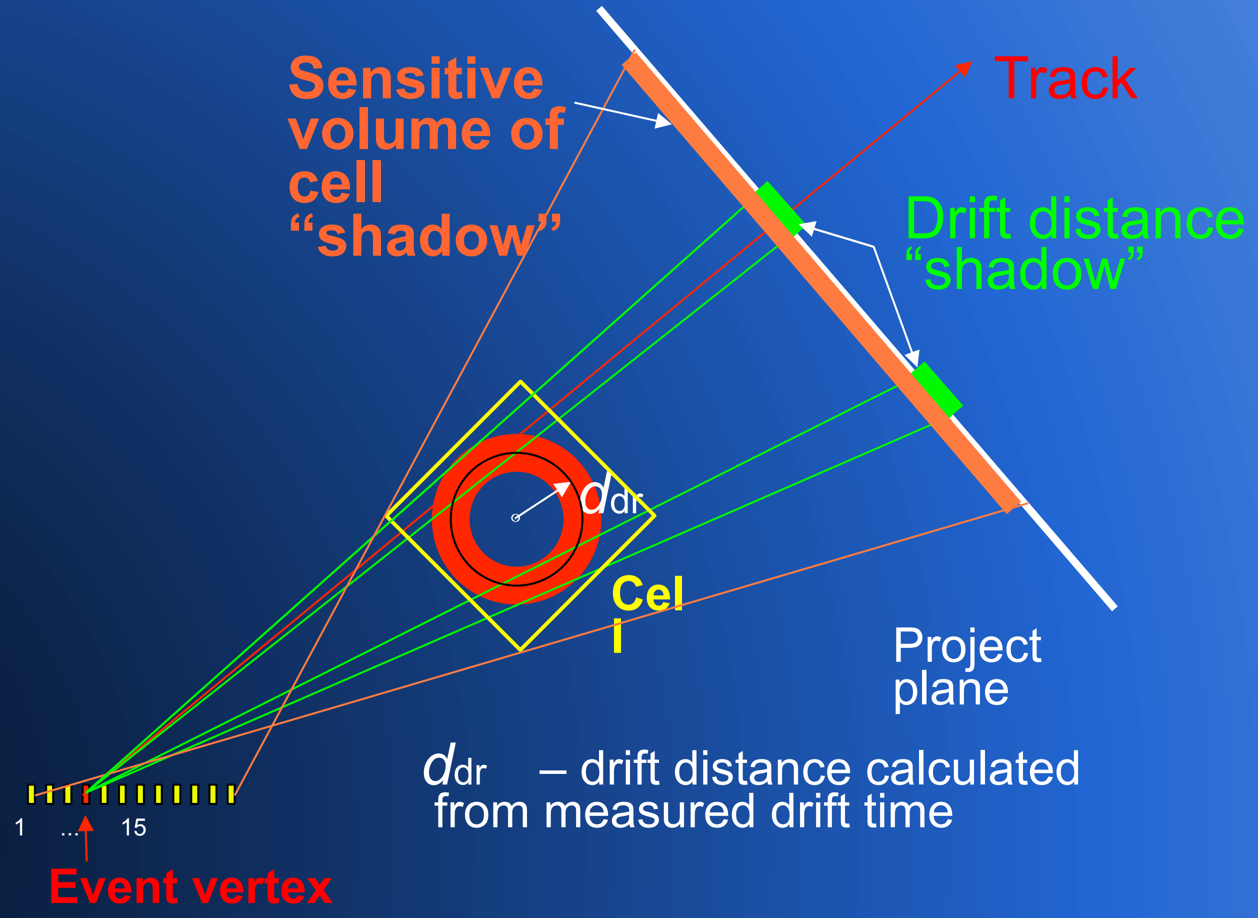


Finding of vertex target segment
Candidate search inner MDCs
Remove Ghost Tracks
Inner MDC segment fitter
Loop over inner candidates:
restrict region on the projection plane for candidate search
Candidate search outer MDCs
Remove Ghost tracks
Matching with META detectors
Outer MDC segment fitter
MDC – META detectors matching
Momentum reconstruction

Scheme of using Drift Time in the Candidate Search

① Using the drift time to shrink the projected drift cell volume increases the contrast of the candidate search

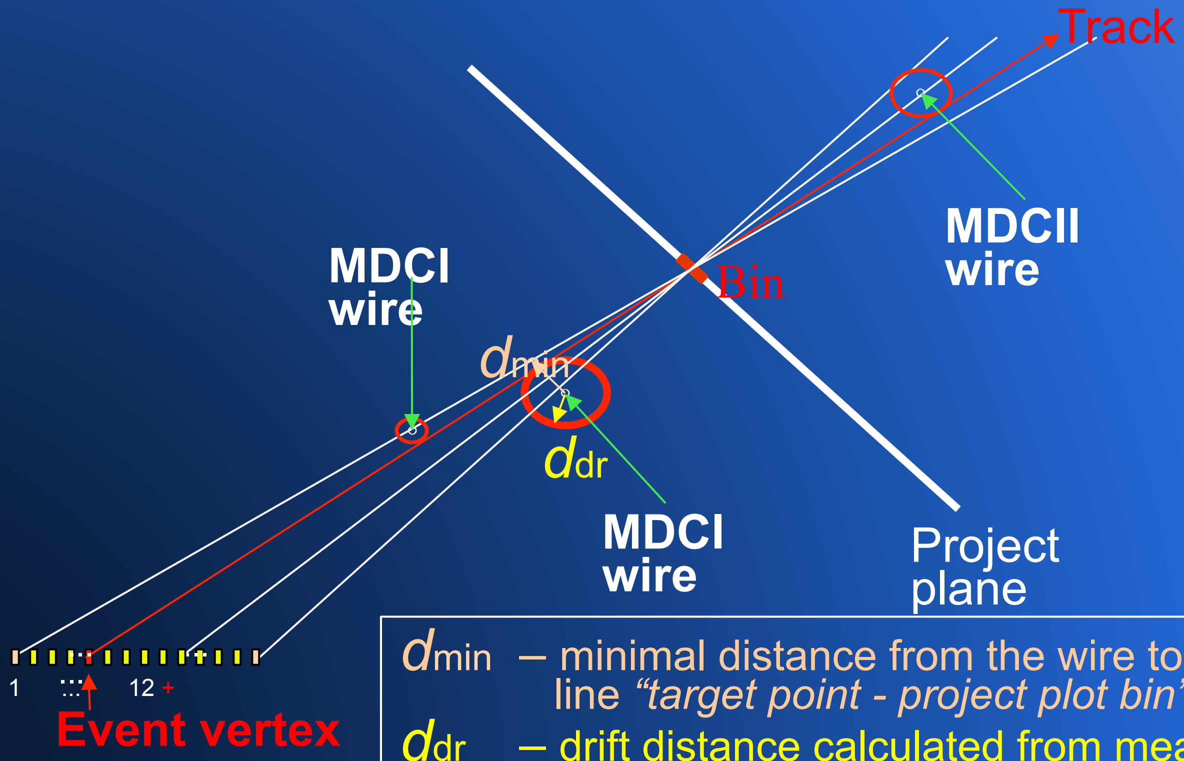
② Increasing the number of bins on the projection plane improves further but increases the number of ghost tracks too



d_{dr} – drift distance calculated from measured drift time

Target Segment reconstruction of the Event Vertex

① **Strategy:** try to find the target segment to shrink down the projection volumes and improved the initial values for tracking



d_{min} – minimal distance from the wire to the line *"target point - project plot bin"*

d_{dr} – drift distance calculated from measured drift time

Cut: $|d_{dr} – d_{min}| < \Delta d_{cut}$

Inner MDCs: Rejection of Ghost Tracks

Ghost cluster signatures:

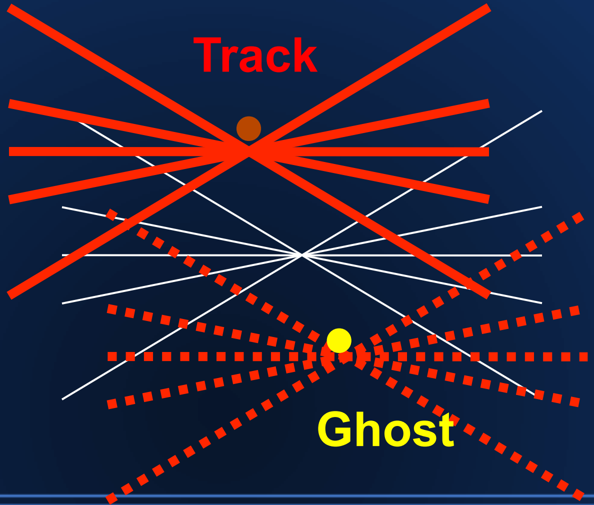
① smaller average spike amplitude

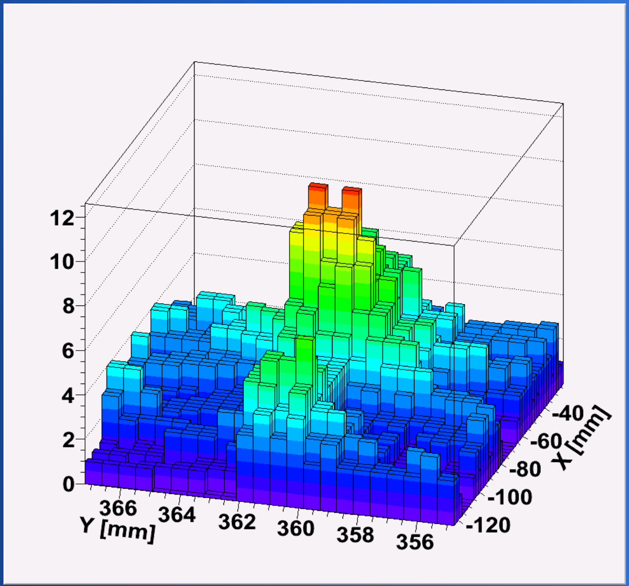
② smaller average number of unique wires (contributing to this cluster only)

③ larger average number of wires participating in real clusters

④ smaller average cluster size (number of bins on the top of spike)

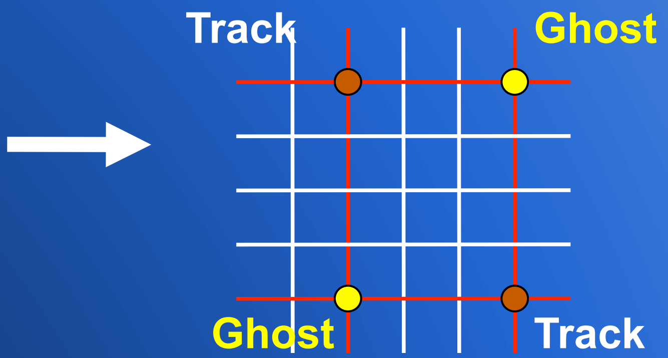
① Find and remove of clusters which have practically **identical set** of wires

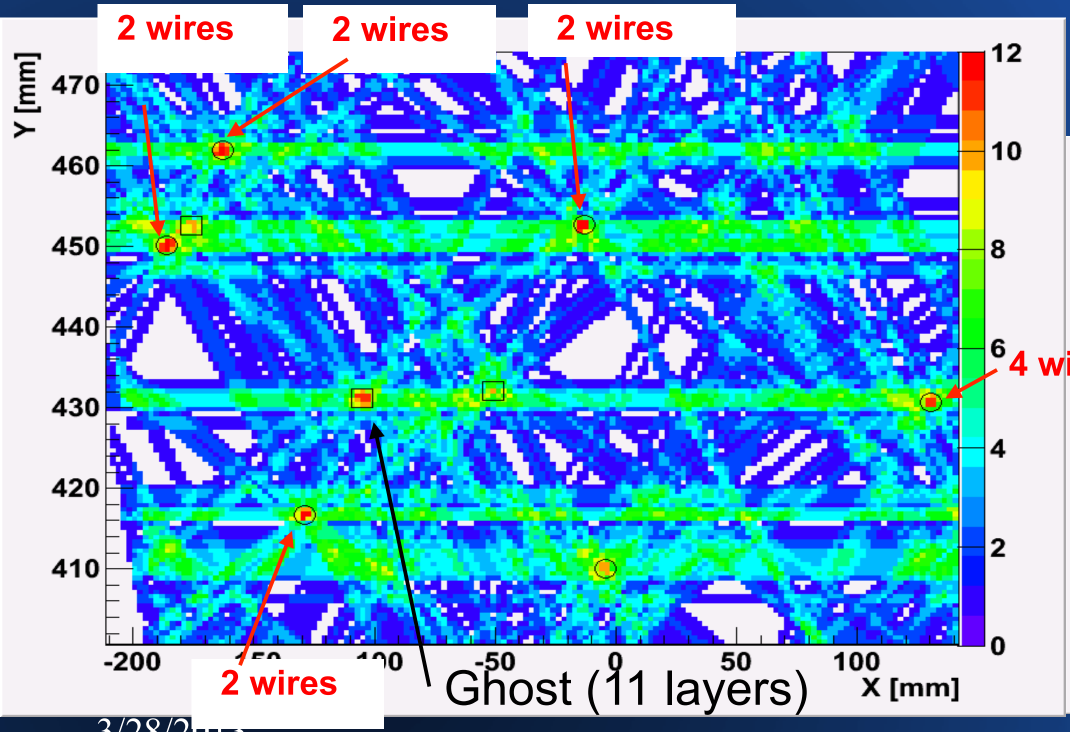


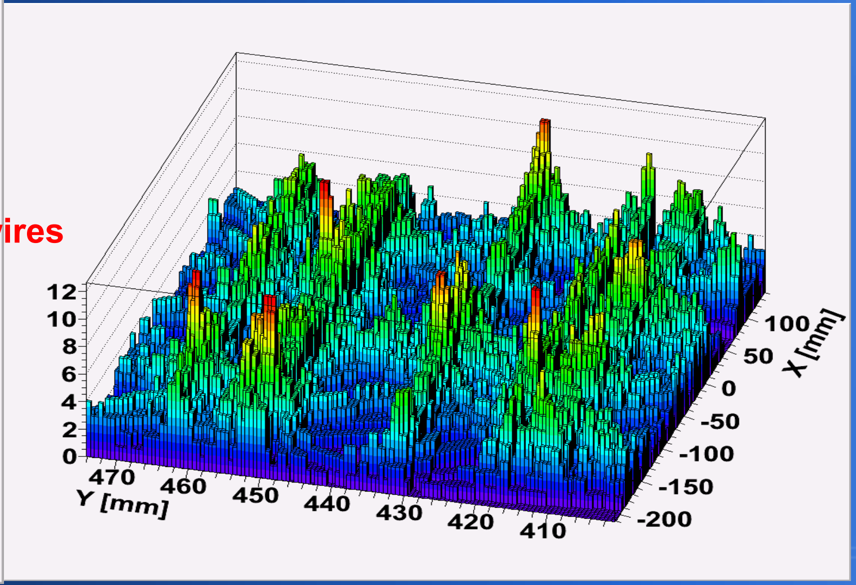


Inner MDCs: Rejection of Ghost tracks

② Removing of clusters **combined from different track** wires. Complex algorithm (~10 parameters)



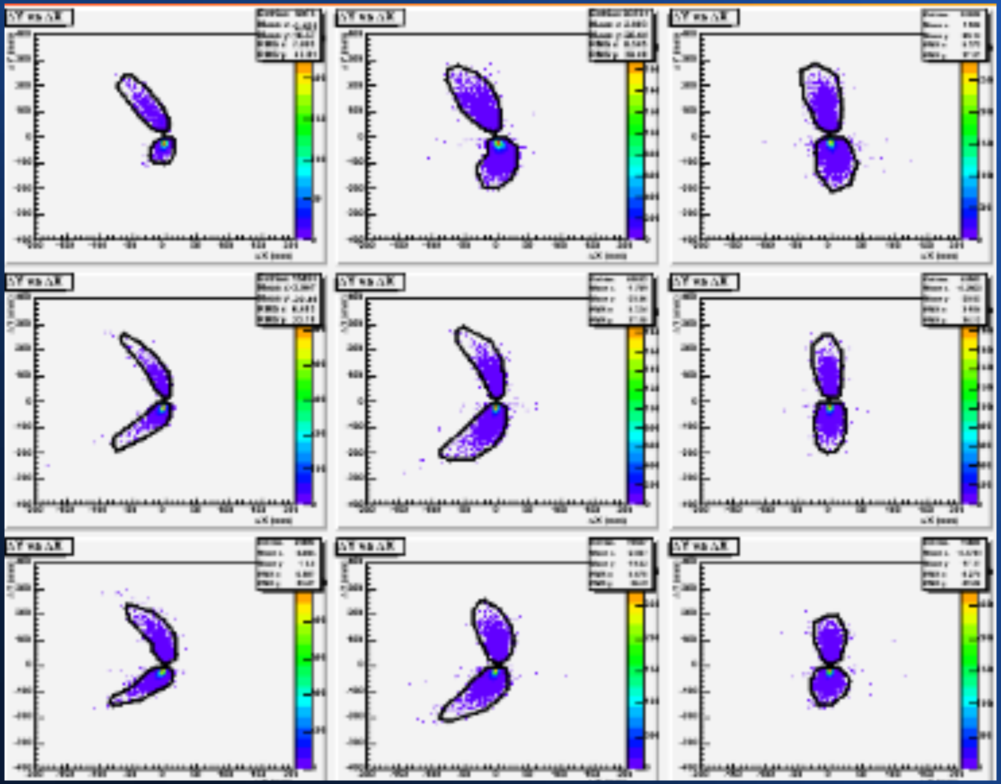




Outer MDCs : Reducing combinatory of possible candidates for one inner segment

Phi ➔

Theta ➔



Strategy:

① restrict projections to range of physical tracks

② Cuts applied on the Δx,Δy coordinates on the projection plane in front and behind the field region

③ Purity:89%

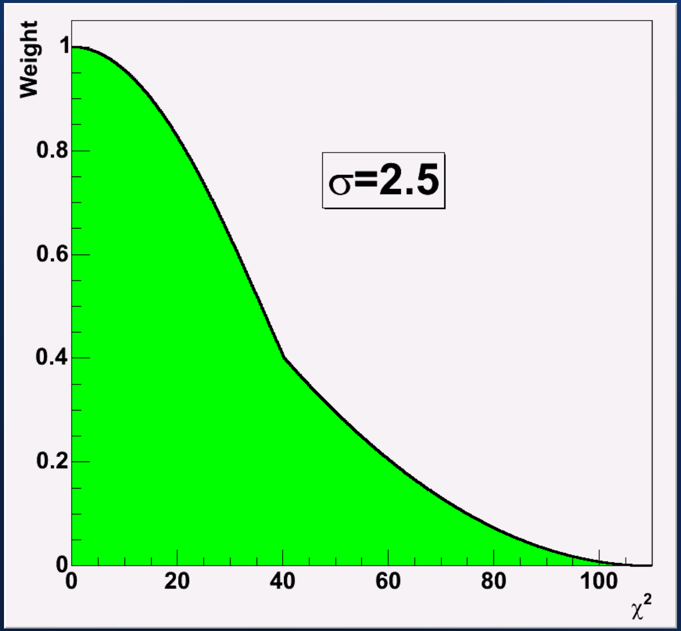
④ 0.26% lost tracks

⑤ 87.2% removed ghosts tracks

⑥ 6x6 bins in phi and theta

The Segment fitter

$$F = \sum_i \frac{(t^i_{drift} + t_{off} - t^i_{TDC})^2}{(\Delta^i_{TDC})^2} \cdot w_i$$



① Pre-fit procedure to find optimal start position for fit to avoid being trapped in local minima

② Straight line fit to wires of a pair of MDCs

③ Distance of closest approach of the line to the wire is converted to drift time

④ Drift time – distance correlation from GARFIELD calculations

⑤ The functional is evaluated in time space

⑥ Global offset parameter is used (time of flight etc…)

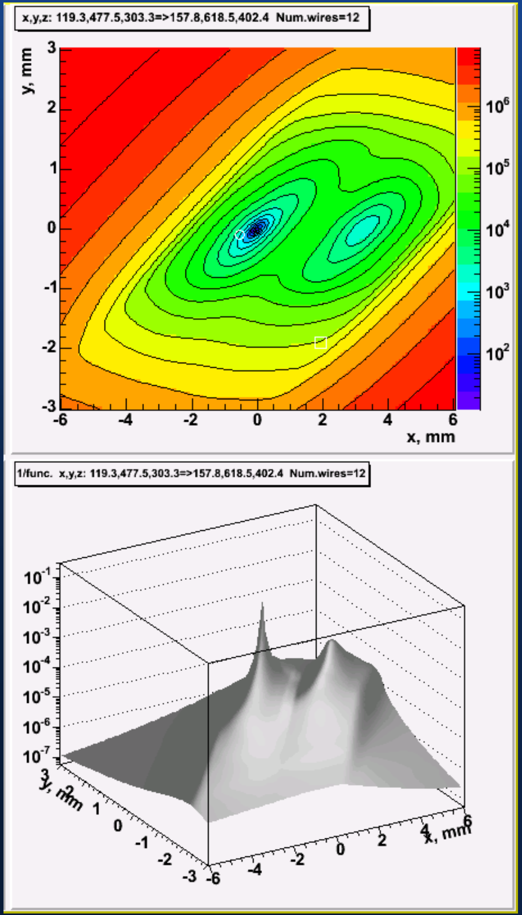
⑦ Tukey weights used to minimize impact of wrong wires in the fit

⑧ χ^2 minimization

⑨ resolution better 150 mu

Avoiding local minima

Fit has not reached minimum



① standard gradient down hill fits can get easily trapped into a local minima

② due to wire orientation many local minima shows up

③ Example: when starting with initial parameters pointing to the global vertex the probability is enhanced to find a local minima close to the starting value ➔ off-vertex tracks get systematically shifted

④ Solution: run a pre fit to test possible combinations of wires for the best initial value

Correlation of detector hits

① HMetaMatchF task matches the detector hits of RICH,MDC and META

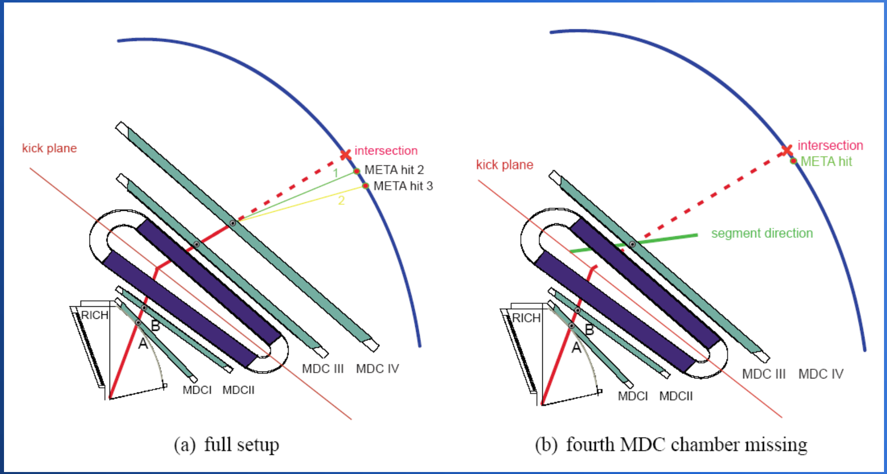
② The matched objects provide input to the momentum reconstruction algorithms

③ Rings in the RICH and track segments in the inner MDCs are spatial correlated inside a matching window given by

$$\Delta\theta = \theta_{rich} - \theta_{mdc}$$

$$qPhi = \frac{(\varphi_{rich} - \varphi_{mdc} - \varphi_{offcet}) * \sin(\theta_{mdc})}{\sigma_{rich}}$$

④ Outer segments are matched with META hits using straight lines (neglecting the curvature of the track in magnetic field). From the difference of the propagated hit position of the segment on the META and the measured Hits position a quality factor (normalized by the errors) is calculated.

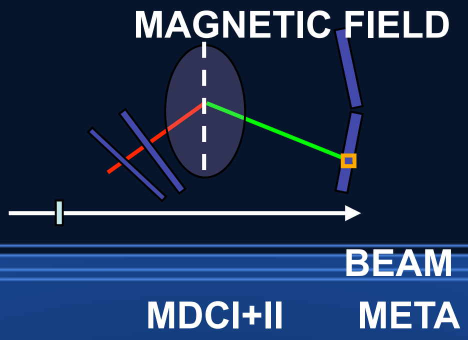


(a) full setup (b) fourth MDC chamber missing

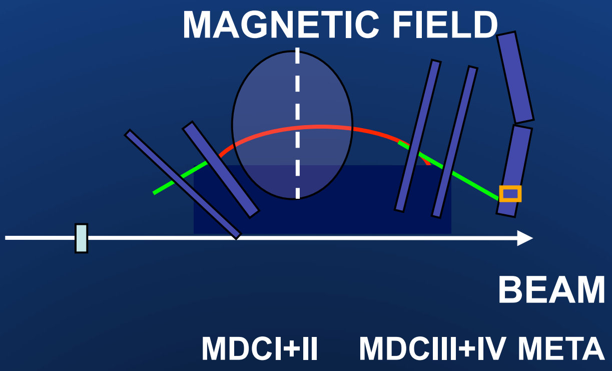
$$qSh = \sqrt{\left(\frac{xSh - xInt - x_{offset}}{\sigma_x}\right)^2 + \left(\frac{ySh - yInt - y_{offset}}{\sigma_x}\right)^2}$$

$$qTof = \sqrt{\left(\frac{xTof - xInt - x_{offset}}{\sigma_x}\right)^2 + \left(\frac{yTof - yInt - y_{offset}}{\sigma_y}\right)^2}$$

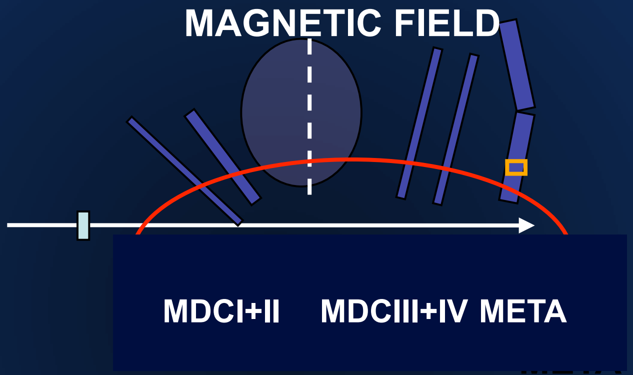
Momentum reconstruction



Kickplane



Spline

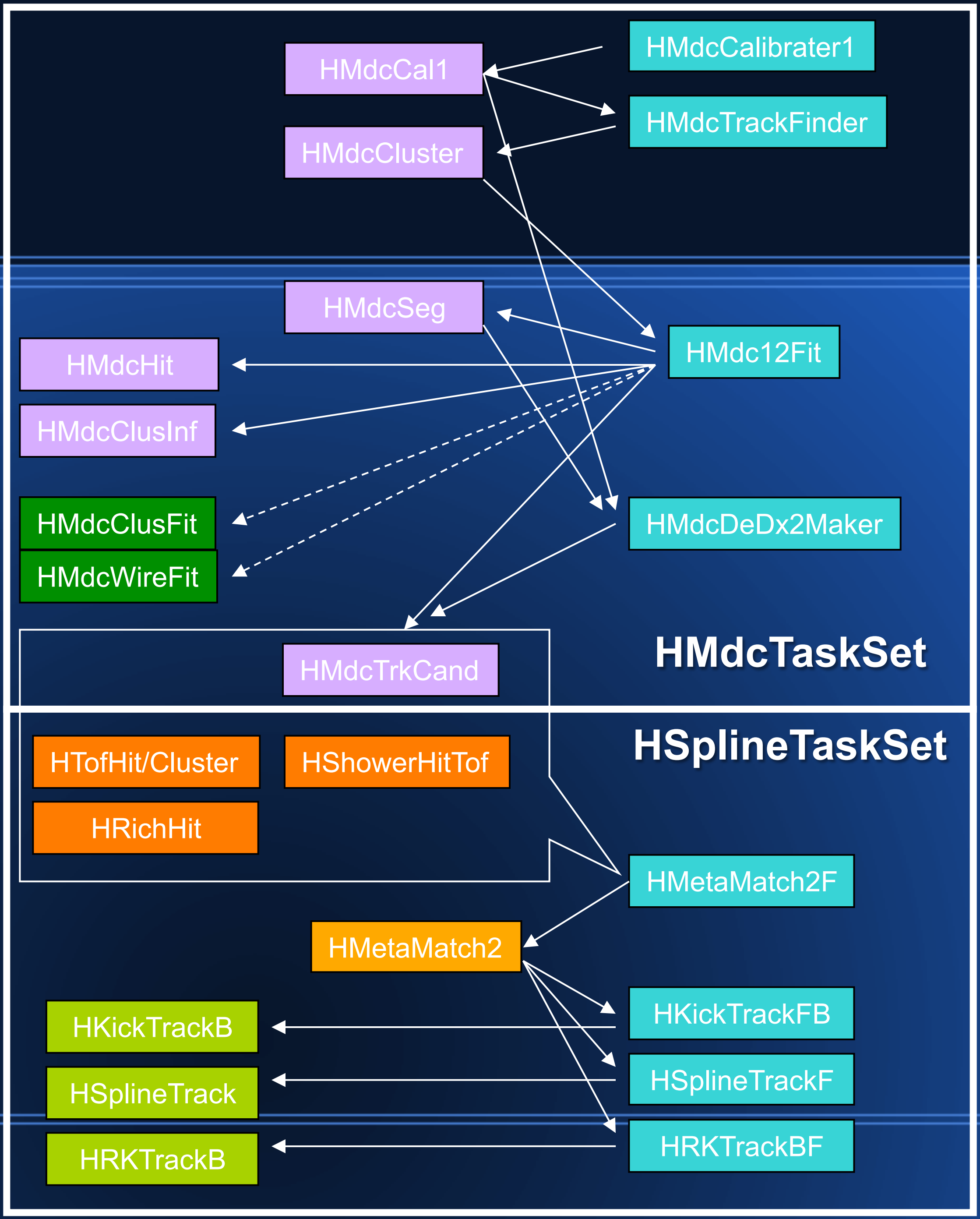


Runge Kutta

I. KickPlane:
   ① use inner segment + META hit to define the deflection angle and momentum
   ② low resolution, very fast

II. Spline :
   ① use 5th order spline to emulate a track in the field region
   ② use inner and outer segments
   ③ the segment directions are not touched by the algorithm
   ④ good resolution, fast

III. Runge Kutta:
   ① propagates particle through magnetic field
   ② uses inner and outer segments
   ③ modifies the segments direction and position
   ④ best resolution, computing intensive



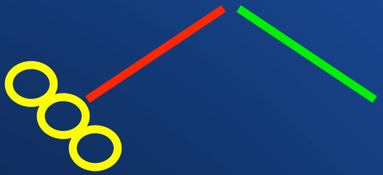
Tracking scheme

① Tracking Tasks and the involved data structures

② From `HMetaMatch2` objects the `HParticleCand` objects will be filled later

HParticleCandFiller

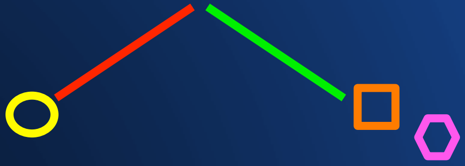
HMetaMatch



	TofClst	TofHit1	TofHit2	RpcClst
1	index, matchQA, RKmatchQA, indRK	…	…	…
2	…	…	…	…
3	…	…	…	…

① Input from `HMdcTrkCand`
② Broad matching with RICH and META
③ RICH sorted by best ΔΦ

HParticleCand



① Best matched RICH in ΔΦ and ΔΘ
② Best matched META hit
③ Prefers RPC+TOF over SHOWER
④ SHOWER hit is always copied

HParticleTrackSorter

I. After the tracking procedure (hit reconstruction, Hit matching, momentum reconstruction still tracks are inside the sample of candidates which share the same detector hits

II. To get rid of such candidates a cleaning procedure is applied

   ① All track candidates are inspected and flagged according certain properties

   ② Two select functions (selectLeptons,selectHadrons) are performed to apply conditions on the candidates

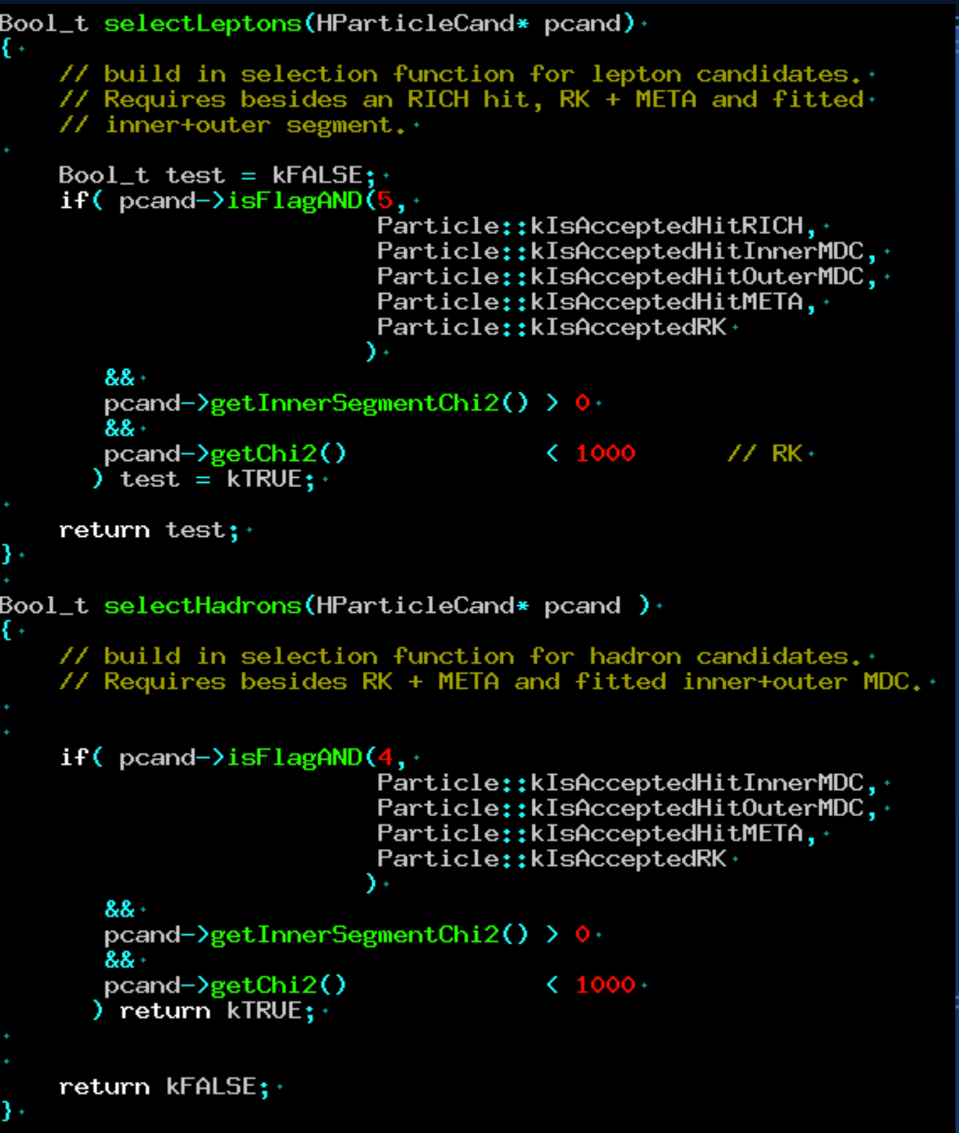
   ③ Candidates which have passed the select functions are inserted in a list which is sorted by track quality (Standard : RK*RKMETAQA)

   ④ Starting from the best candidate a list of used detector hits is build and each following candidate which reuses an hit is discarded. Which detector hits should be ignored in the double hit rejection can be configured. Rich hits are ignored.

   ⑤ Candidates which survives the selection are marked with kIsUsed

   ⑥ Candidates selected by the selectLepton function are marked in addition as kIsLepton

HParticleTrackSorter



```
Bool_t selectLeptons(HParticleCand* pcand)
{
    // build in selection function for lepton candidates.
    // Requires besides an RICH hit, RK + META and fitted
    // inner+outer segment.

    Bool_t test = kFALSE;
    if( pcand->isFlagAND(5,
                         Particle::kIsAcceptedHitRICH,
                         Particle::kIsAcceptedHitInnerMDC,
                         Particle::kIsAcceptedHitOuterMDC,
                         Particle::kIsAcceptedHitMETA,
                         Particle::kIsAcceptedRK
                        )
       &&
       pcand->getInnerSegmentChi2() > 0
       &&
       pcand->getChi2()             < 1000      // RK
      ) test = kTRUE;

    return test;
}

Bool_t selectHadrons(HParticleCand* pcand )
{
    // build in selection function for hadron candidates.
    // Requires besides RK + META and fitted inner+outer MDC.


    if( pcand->isFlagAND(4,
                         Particle::kIsAcceptedHitInnerMDC,
                         Particle::kIsAcceptedHitOuterMDC,
                         Particle::kIsAcceptedHitMETA,
                         Particle::kIsAcceptedRK
                        )
       &&
       pcand->getInnerSegmentChi2() > 0
       &&
       pcand->getChi2()             < 1000
      ) return kTRUE;


    return kFALSE;
}
```

① Selection functions have to be provided by the user

② Selection takes place in the user macro. Flags written during DST production can be reset

③ The selection function has to used with care: For analysis of rare probes the selection has to remove enough background to avoid random matches. On the other hand don't select to strong to allow for more refined cuts later

Retrieving data containers

All data containers (Catgegories) can be retieved from the current Event:

```
HCategory* mycat = gHades->getCurrentEvent()->getCategory(catNumber)
```

① catNumbers are defined in xxxdef.h of each hydra library (typical detector or tracking lib) they belong to

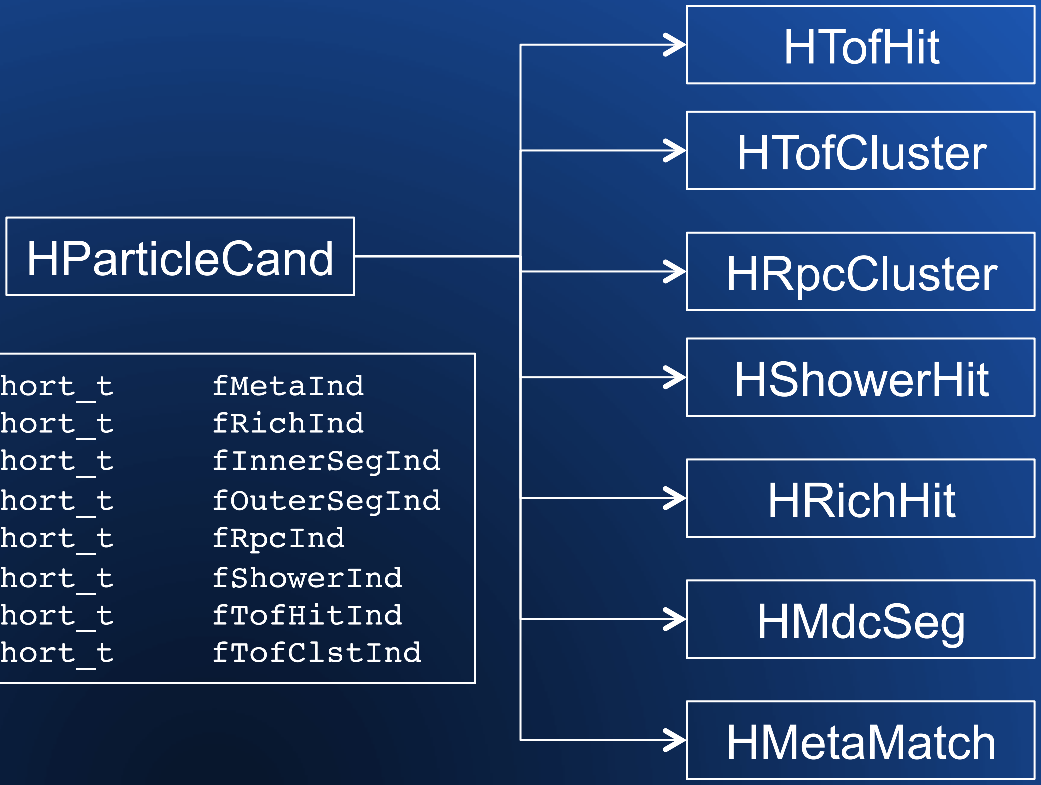
② `HCategory` is basically a `TClonesArray` of objects of the same type.

③ Each object can be retrieved via the index in the array.

④ Links from objects to other objects inside the same or other categories are stored via indices.

⑤ Since data structures in tracking are complex it is needed to retrieve the full tree of objects starting from the highest level object down:
`HMdcSeg` → `HMdcCal1`

Retrieve Hit information form HParticleCand



```
Short_t      fMetaInd
Short_t      fRichInd
Short_t      fInnerSegInd
Short_t      fOuterSegInd
Short_t      fRpcInd
Short_t      fShowerInd
Short_t      fTofHitInd
Short_t      fTofClstInd
```

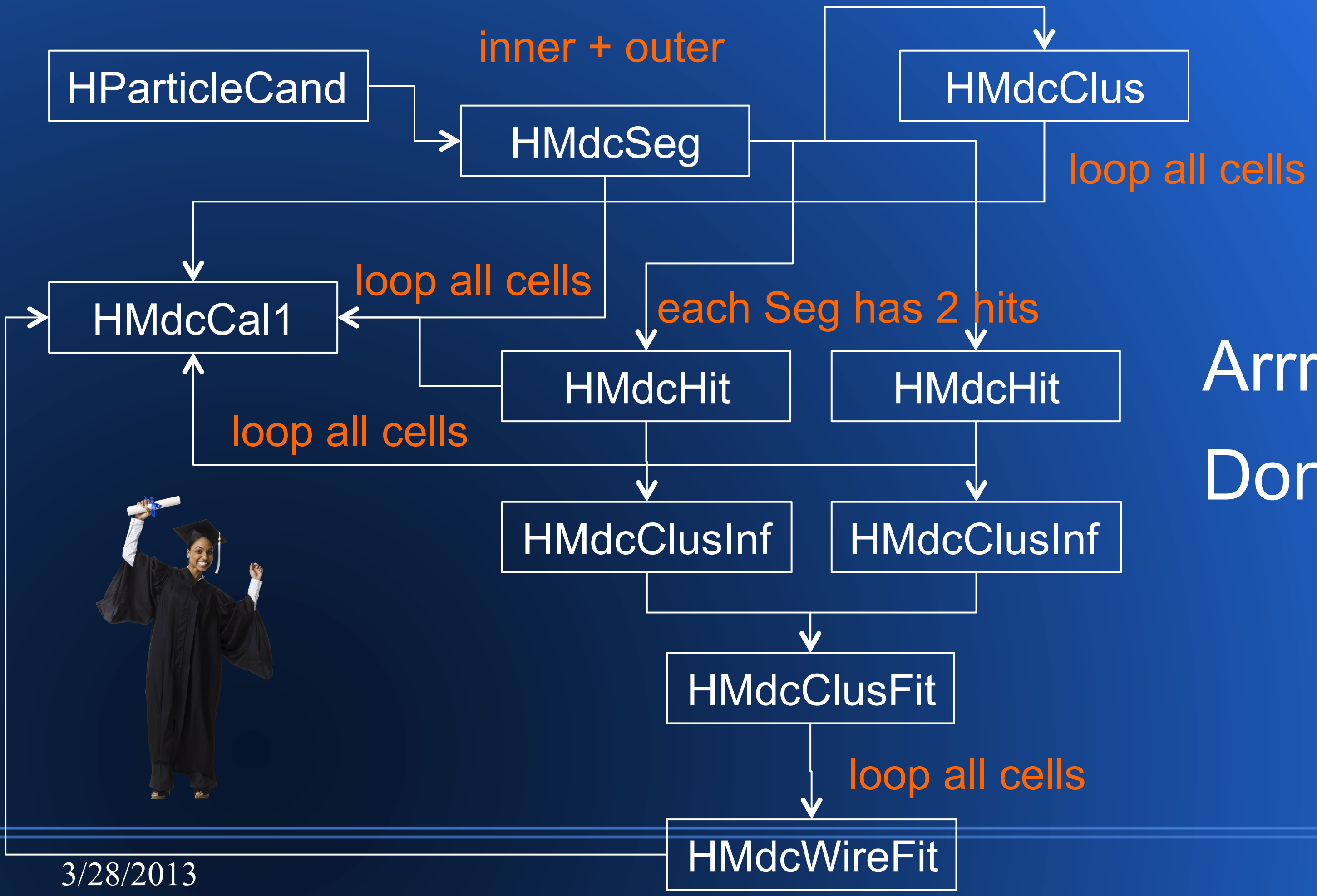
The easy part:

① direct access to the hit objects via stored index

② indices are always -1 if no hit was contributing

③ Hint for META: take the hit which was has been used to provide beta

Retrieving Tracking information



Arrrgh ... but ...

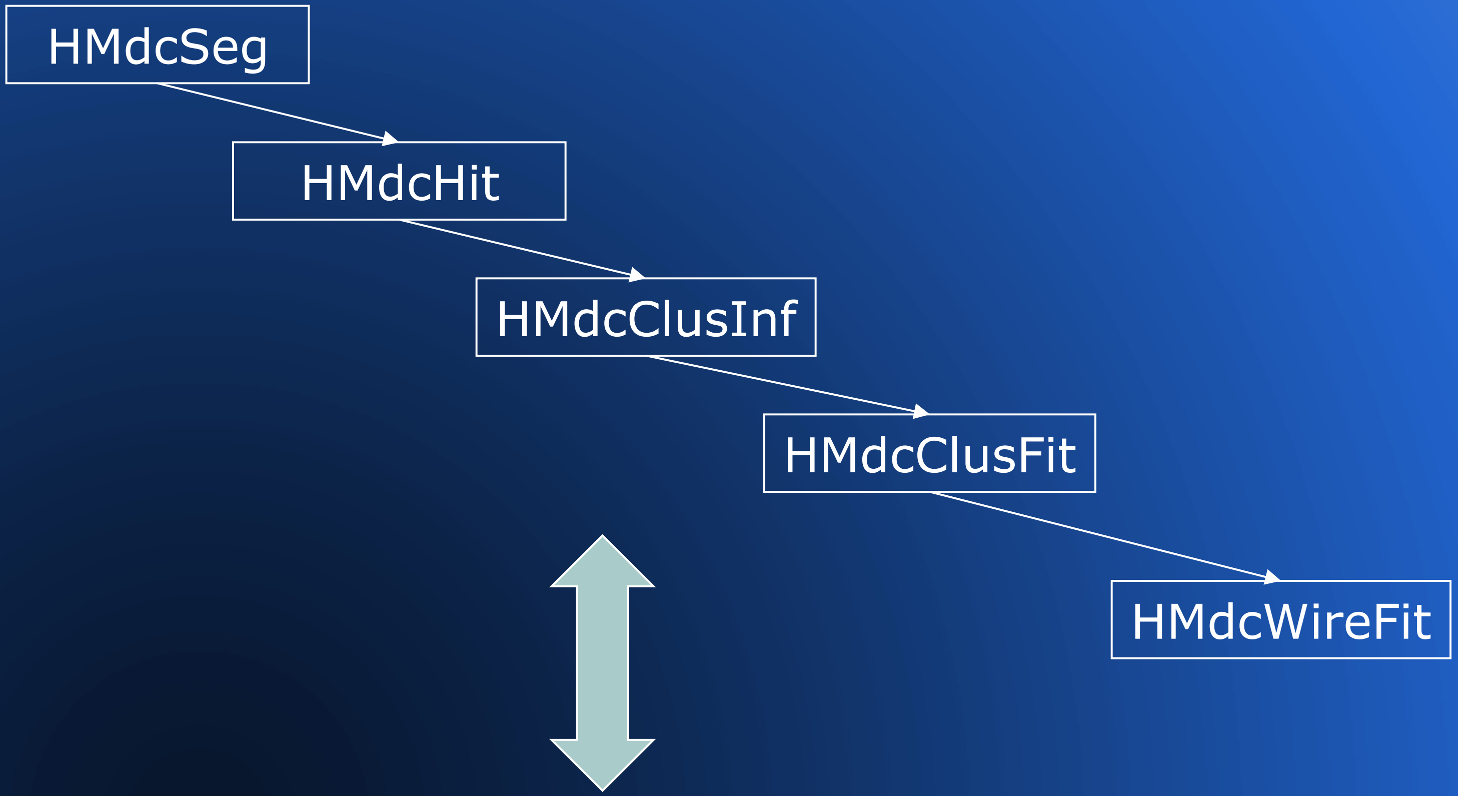
Don't give up !

Helper functions

```
static TObjArray*   HParticleTool::getMdcCal1Cluster(Int_t segind)
static TObjArray*   HParticleTool::getMdcCal1Seg(Int_t segind)
static HMdcClus*    HParticleTool::getMdcClus(Int_t segind)
static HMdcClusFit* HParticleTool::getMdcClusFit(Int_t segind)
static HMdcClusInf* HParticleTool::getMdcClusInf(Int_t segind, Int_t nhit = 0)
static HMdcHit*     HParticleTool::getMdcHit(Int_t segind, Int_t nhit = 0)
static HMdcSeg*     HParticleTool::getMdcSeg(Int_t segind)
static HMdcTrkCand* HParticleTool::getMdcTrkCand(Int_t metaind)
static TObjArray*   HParticleTool::getMdcWireFitSeg(Int_t segind)
static HMetaMatch2* HParticleTool::getMetaMatch(Int_t metaind)
static HRichHit*    HParticleTool::getRichHit(Int_t richind)
static HRpcCluster* HParticleTool::getRpcCluster(Int_t rpcind)
static HShowerHit*  HParticleTool::getShowerHit(Int_t showerind)
static HTofCluster* HParticleTool::getTofCluster(Int_t tofind)
static HTofHit*     HParticleTool::getTofHit(Int_t tofind)
```

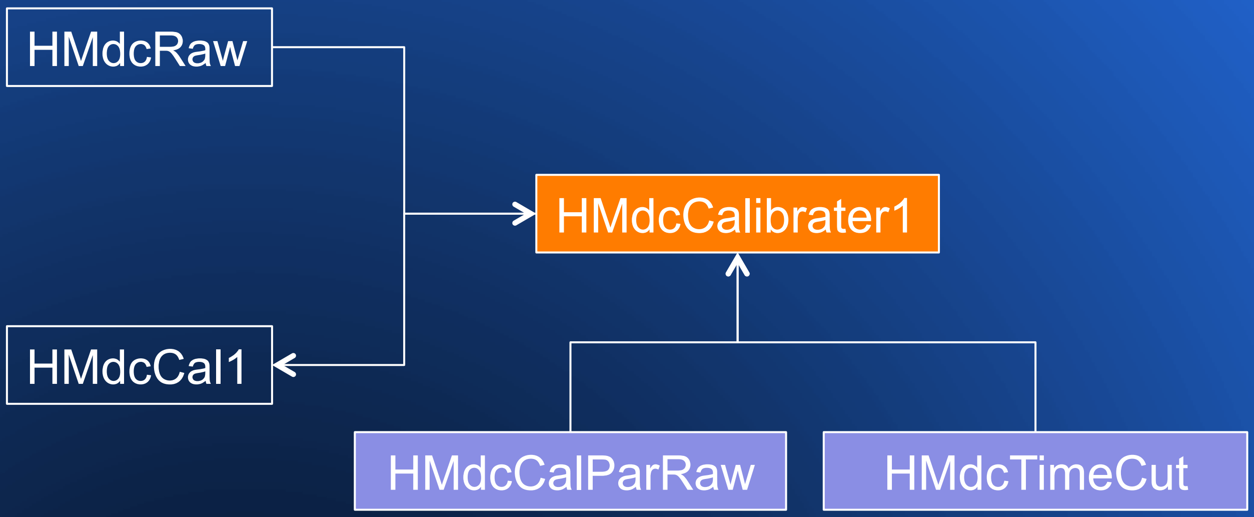
Example SIM ROOT file with all Categories can be found in:
/hera/hades/user/kempter/tracking_tutorial/

Things which are obvious how to use ...



```
TObjArray*   HParticleTool::getMdcWireFitSeg(Int_t segind)
```

HMdcCalibrater1



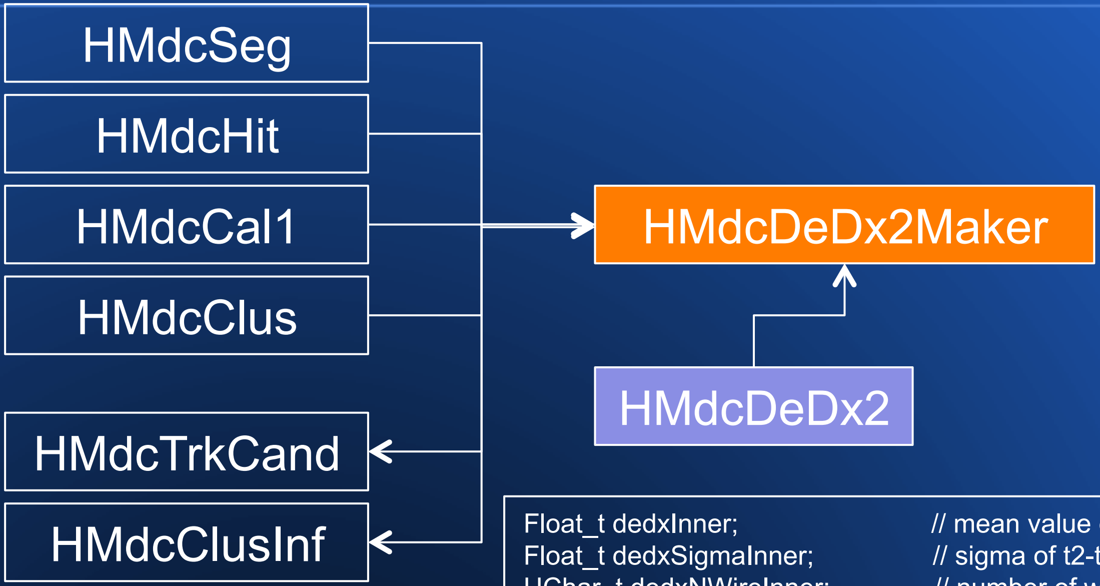
```
Int_t nHits;    // number of hits in this cell
Float_t time1;  // drift time of first hit  [ns]
Float_t time2;  // trailing edge of signal [ns]
Int_t sector;   // sector number [0-5]
Int_t module;   // module number [0-3]
Int_t layer;    // layer number  [0-5]
Int_t cell;     // cell number in wire plane
```

① Calibrate tdc measurements to drift times: `t1=offset-slope*raw`

② Apply noise cuts for each MDC module

③ Change coordinates form Electronics to MDC: `sector,module,motherboard,tdc` → `sector, module,layer,cell`

HMdcDeDx2Maker

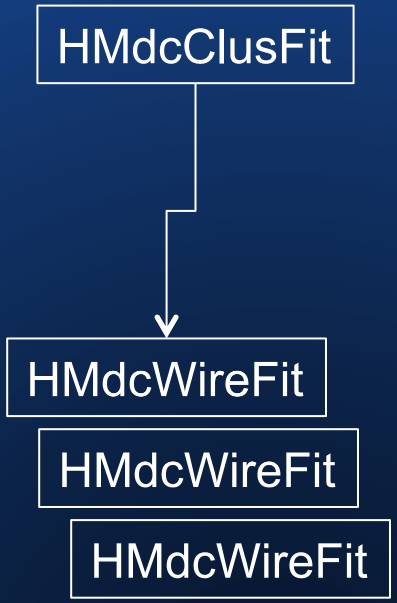


① Calibrate `HMdcCal1` data to dEdx using impact angle into the drift cell and distance from sense wire

② Reads list of cells from tracking classes

```
Float_t dedxInner;                 // mean value of t2-t1 for inner segment
Float_t dedxSigmaInner;            // sigma of t2-t1 distribution in inner segment
UChar_t dedxNWireInner;            // number of wires in inner segment before truncated mean procedure
UChar_t dedxNWireCutInner;         // number of wires in inner segment cutted by truncated mean procedure
Float_t dedxOuter;                 // mean value of t2-t1 for outer segment
Float_t dedxSigmaOuter;            // sigma of t2-t1 distribution in outer segment
UChar_t dedxNWireOuter;            // number of wires in outer segment before truncated mean procedure
UChar_t dedxNWireCutOuter;         // number of wires in outer segment cutted by truncated mean procedure
Float_t dedxCombined;              // mean value of t2-t1 for inner+outer segment
Float_t dedxSigmaCombined;         // sigma of t2-t1 distribution in inner+outer segment
UChar_t dedxNWireCombined;         // number of wires in inner+outer segment before truncated mean procedure
UChar_t dedxNWireCutCombined; // number of wires in inner+outer segment cutted by truncated mean proced.
Float_t dedx[4];                   // mean value of t2-t1 per module
Float_t dedxSigma[4];              // sigma of t2-t1 distribution per module
UChar_t dedxNWire[4];              // number of wires per module before truncated mean procedure
UChar_t dedxNWireCut[4];           // number of wires per module cutted by truncated mean procedure
```

Tracking data containers extended

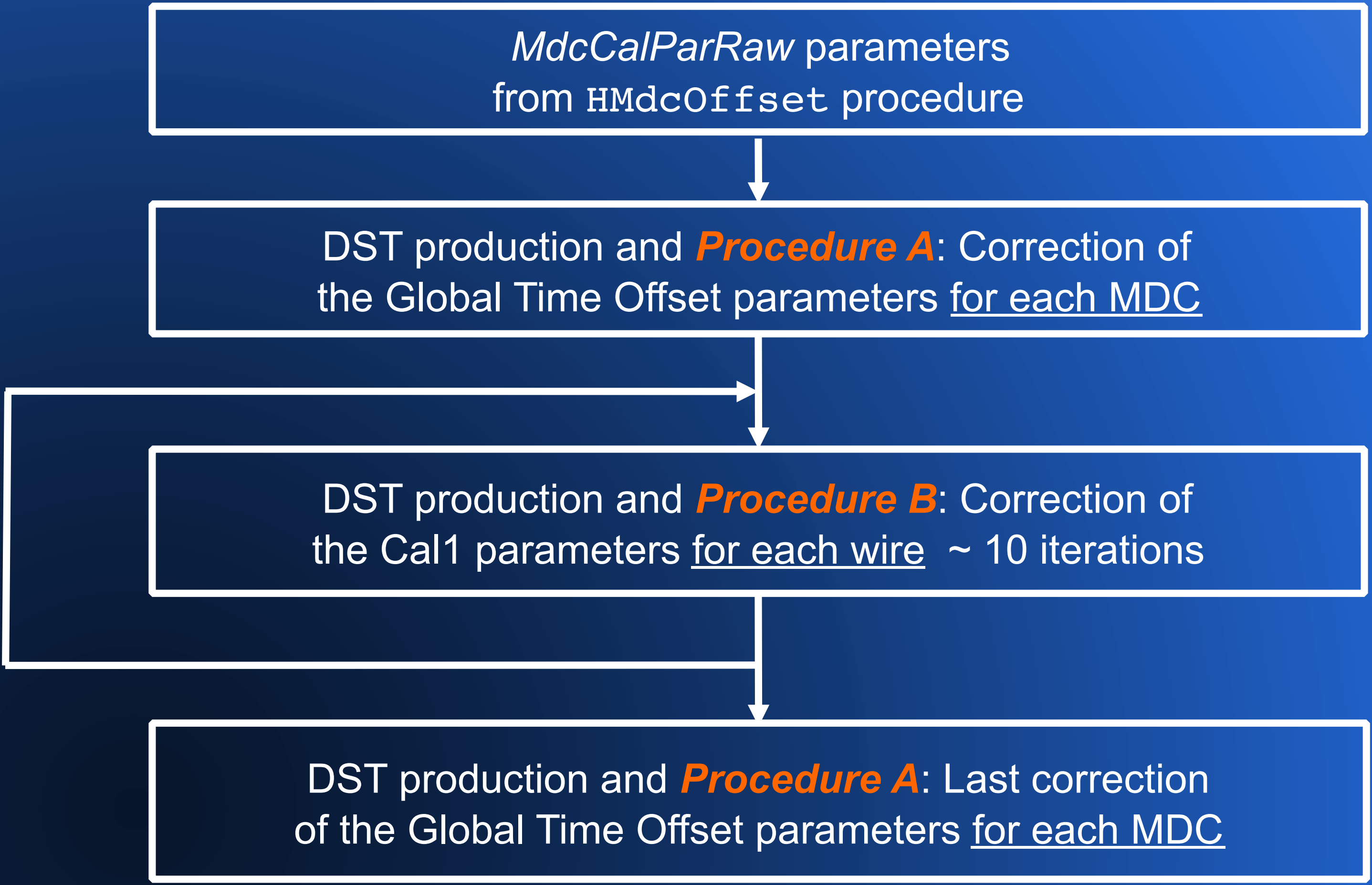


```
Float_t functional;  // value of functional
Float_t timeOffMdc1; // time offset for each MDC in sec. [ns]
Float_t timeOffMdc2; // if timeOff==-1000. no wires in this mod.
Float_t timeOffMdc3; //
Float_t timeOffMdc4; //
Short_t numOfWires;  // Num. of wires with weight > weight_min
Char_t  numOfLayers; // Num. of layers with wire weight > weight_min
Float_t x1;          // Track parameters: [mm]
Float_t y1;          // Track is line (x1,y1,z1) - (x2,y2,z2)
Float_t z1;          // in sector coor. system.
Float_t x2;          //
Float_t y2;          //
Float_t z2;          //
Short_t numIter;     // number of iterations
Int_t   indf;        // index of the first HMdcWireFit obj.
Int_t   indl;        // index of the last HMdcWireFit obj.
```

```
Char_t  timeNum;       // =1 or 2 (time1 or time2 from HMdcCal1)
Float_t tdcTimeCal1;   // drift time from HMdcCal1
Float_t tdcTime;       // tdcTimeCal1 minus time of wire offset
Float_t dev;           // fullTime-tdcTime
Float_t weight;        // wire weight. =0 if wire was excluded from fit
Float_t driftTime;     // calculated drift time
Float_t fullTime;      // driftTime + time offset
Float_t minDist;       // min.distance from track to wire with SIGN.
                       // >0 - track pass OVER wire (in mdc coor.sys.)
Float_t alpha;         // impact angle in cell system
Bool_t  isInCell;      // =kFALSE if track don't cross cell
Float_t tdcTimeErr;    // error of drift time.
Bool_t  isUsedInFit;   // kTRUE if wire was used in fit
```

This categories have to be explicitly switched on!

MdcCal1 Calibration



MdcCalParRaw parameters
from `HMdcOffset` procedure

DST production and ***Procedure A***: Correction of
the Global Time Offset parameters for each MDC

DST production and ***Procedure B***: Correction of
the Cal1 parameters for each wire ~ 10 iterations

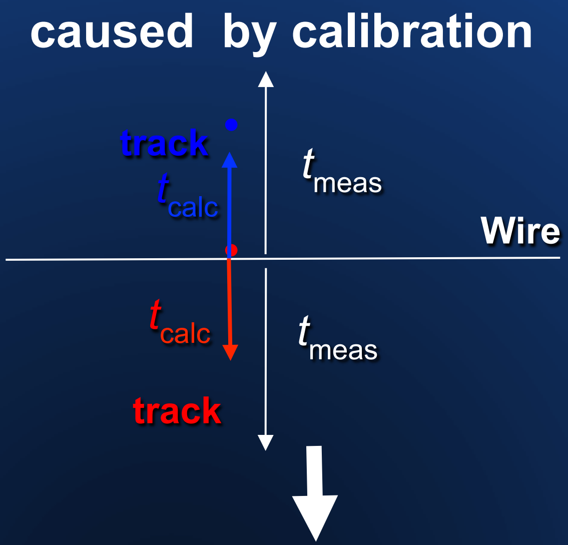
DST production and ***Procedure A***: Last correction
of the Global Time Offset parameters for each MDC

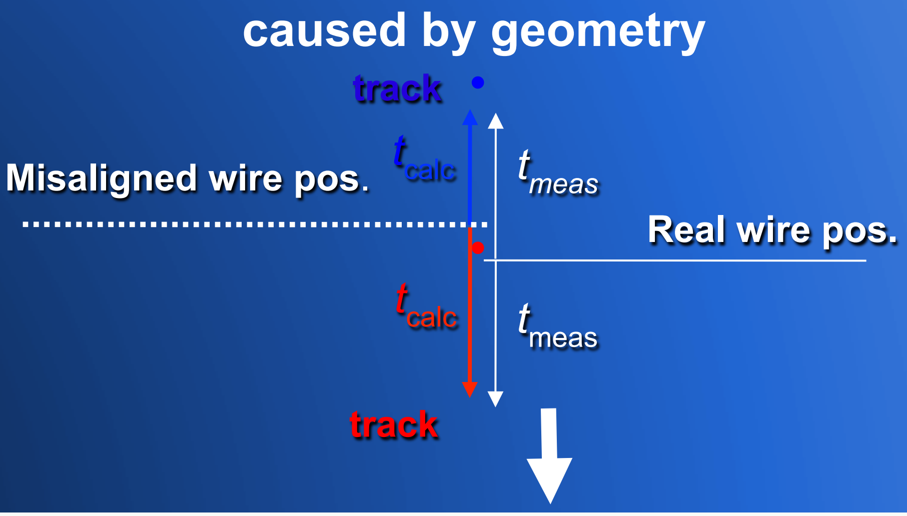
Time Calibration and Geometry

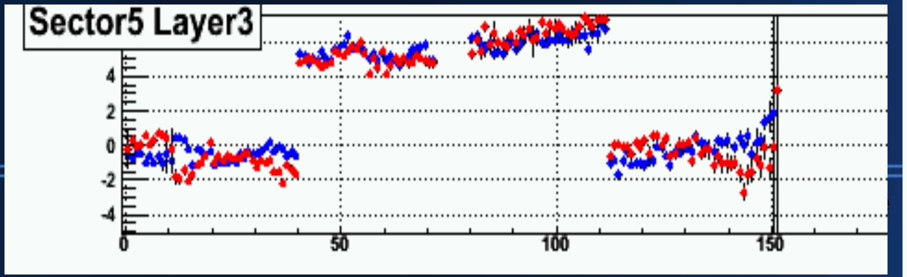
① Time deviation Δt = $t_{calculated} - t_{measured}$ [ns]

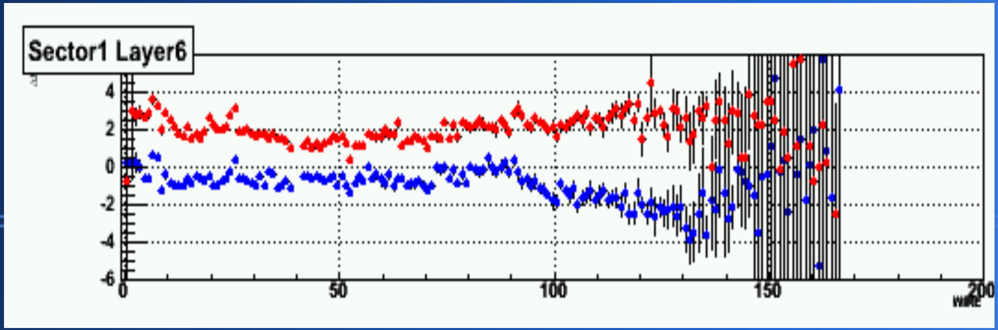
② Tracks passed left and right side of the wire

Investigation by dependence Δt vs wire





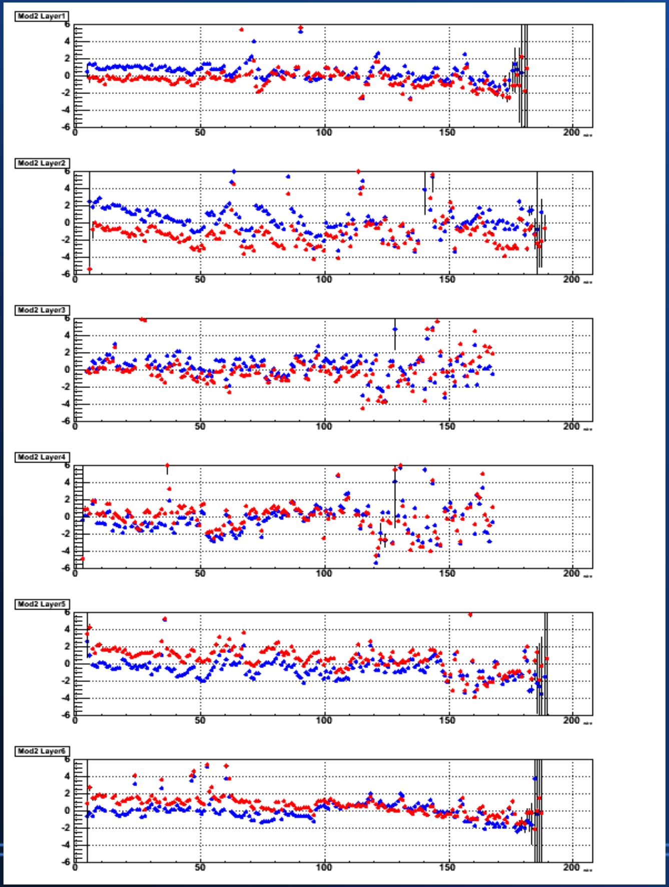




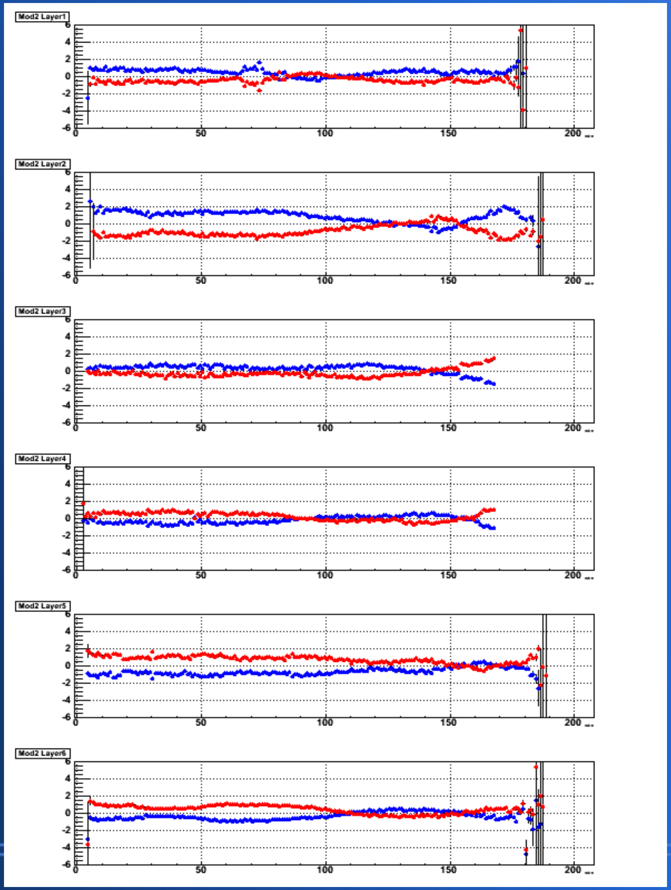
Comparison Δt before and after calibration

MDCII

Before



After


Mod2 Layer1
Mod2 Layer2
Mod2 Layer3
Mod2 Layer4
Mod2 Layer5
Mod2 Layer6