Tracking in Test-beam data analysis on micro-ITS3 with target (July 2021)

Shyam Kumar, Arianna Torres, Francesco Barile, Giuseppe Bruno, Gianfranco Tassielli, Mihail Bogdan Blidaru, Lukas Lautner, Alperen Yuncu, et al.

INFN Bari, Italy

Previous presentations

https://indico.cern.ch/event/1207616/contributions/5079140/attachments/2521089/4335005/Tracking_BeamTestData_Shyam.pdf

https://indico.cern.ch/event/1196877/contributions/5036272/attachments/2503083/4300248/WP3_meeting_06_09_2022.pdf

Analysis Steps

- Status of Tracking:
  - Event selection: done
  - Track finding: done
  - Track fitting using Global Chi2 fitting (Ignoring MS) and the Kalman filter: done
  - Distance of Closest Approach between Beam and Produced Tracks: done
  - Distance in the transverse plane w.r.t. to the Beam position at Z=0: done
  - Event display package for both cases Chi2 fitting and Kalman filter: done

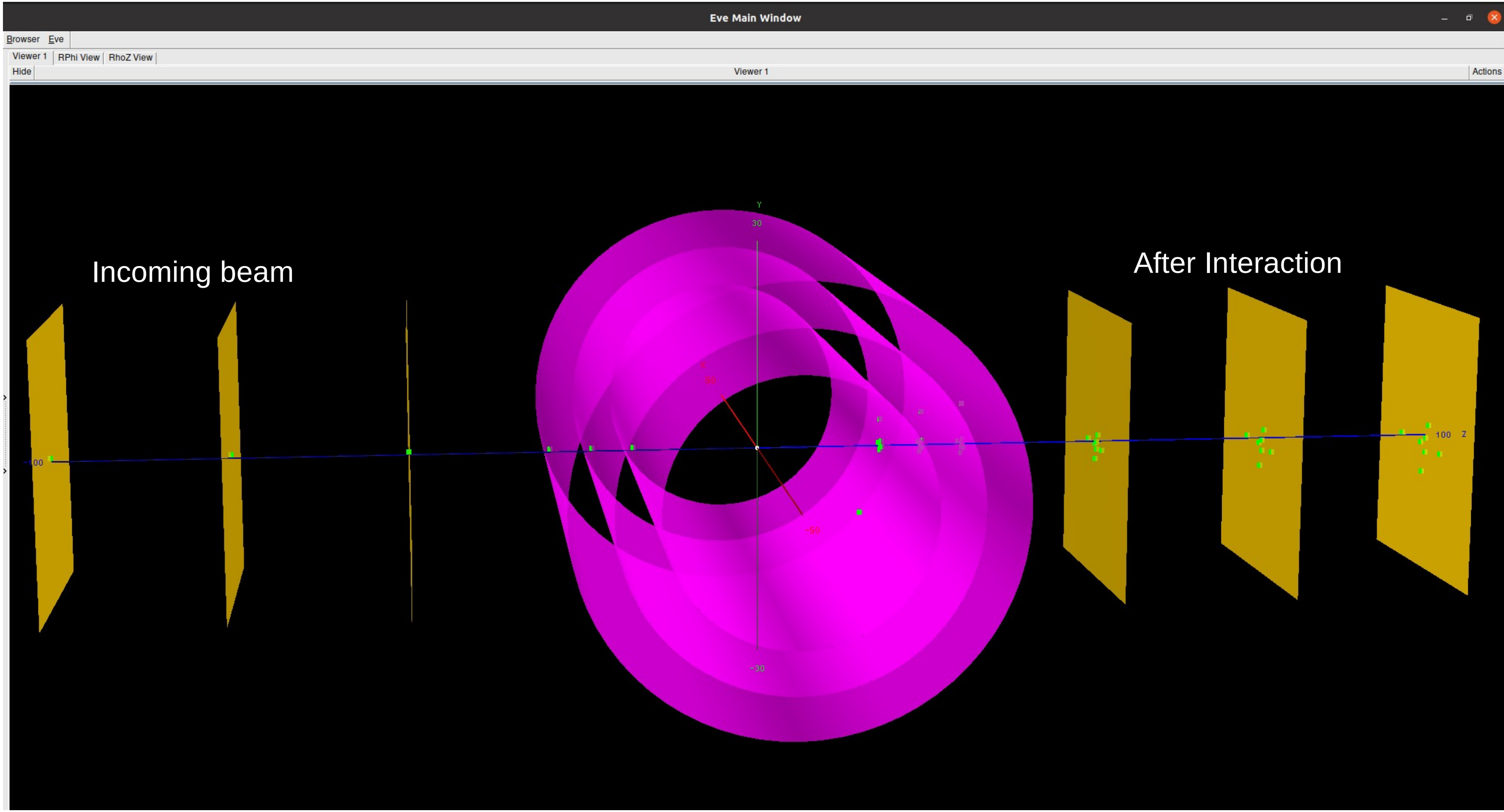
Reference for Kalman filter:

1. R. Fruhwirth, APPLICATION OF KALHAN FILTERING TO TRACK FITTING IN THE DELPHI DETECTOR
https://inspirehep.net/files/52ffb2cc6aed04254988586bb79e1532

2. R. Fruhwirth, Application of Kalman filtering to track and vertex fitting
https://www.sciencedirect.com/science/article/abs/pii/0168900287908874?via%3Dihub

3. Belikov, Yu (Dubna, JINR) ; Safarík, K (CERN) ; Batyunya, B (Dubna, JINR), Kalman Filtering Application for Track Recognition and Reconstruction in ALICE Tracking System
http://cds.cern.ch/record/689414/files/INT-1997-24.pdf

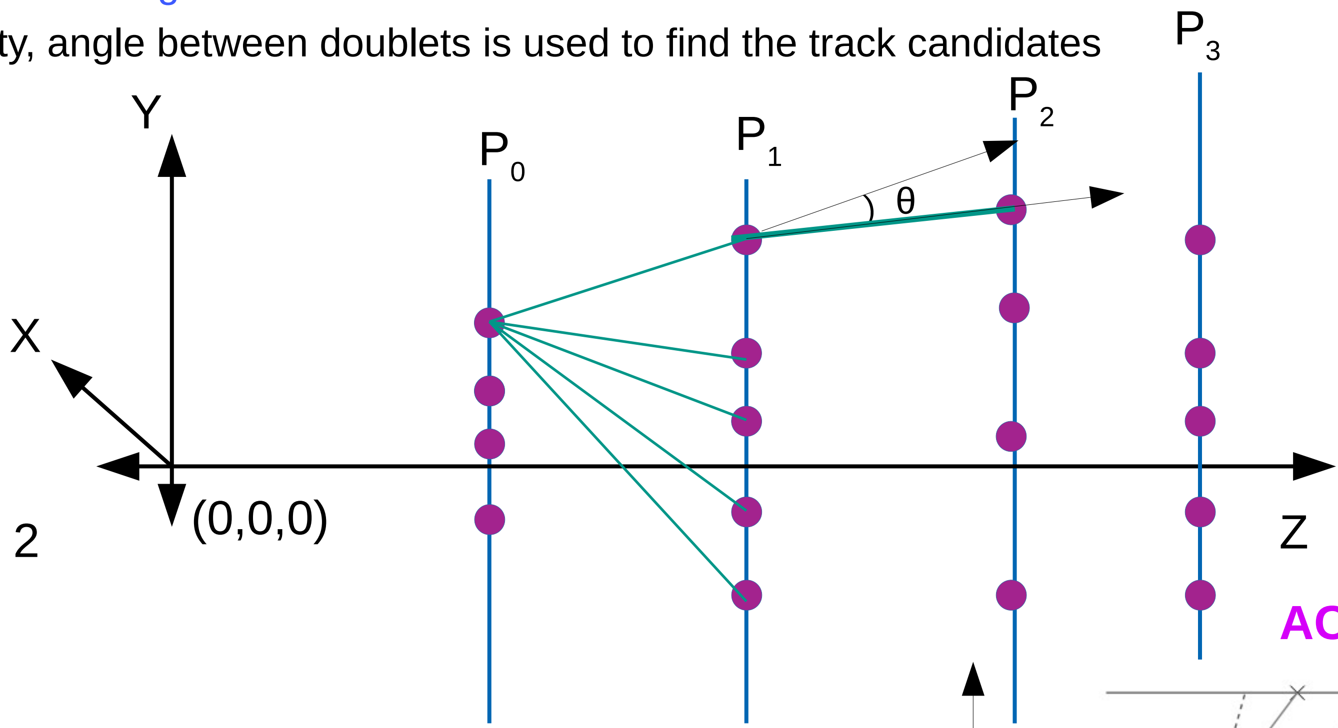
Event display (3D-view)



Thanks Gianfranco for the help in implementing

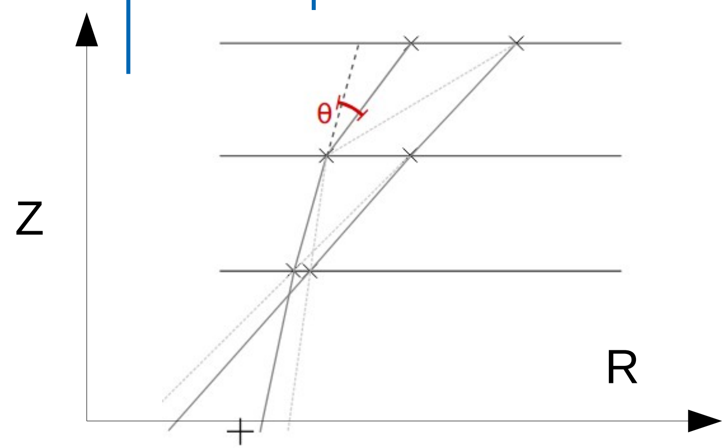
MS: Multiple Scattering

- Low track density, angle between doublets is used to find the track candidates



Tracking:
Number of hits > 2
Chi2/ndf

ACTS



Main Idea: Hits belong to a track should be compatile with in some sigma 's of MS angle (multiple of $\sigma_{\theta MS}$)

Fig2. ACTS on the link below

https://iopscience.iop.org/article/10.1088/1742-6596/898/4/042011/pdf

Track Fitting (Global Chi2 Minimization)

No Mangetic field: Track model (straight line) in three dimensions

Vector equation of a line in 3D:

$$\vec{r} = \vec{r}_0 + \vec{a}$$

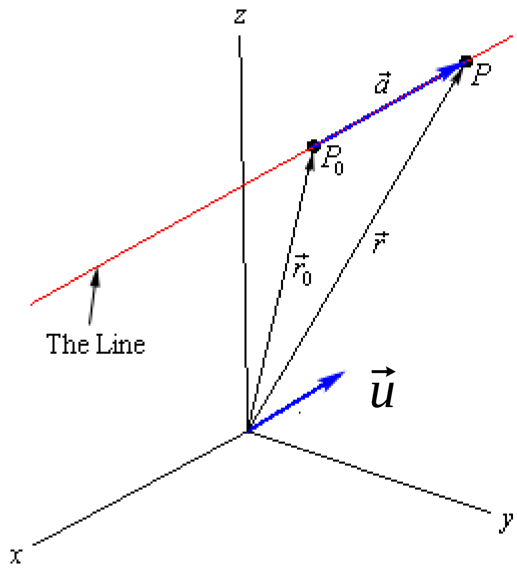
If u is the unit vector along the line and t is parameter:

$$\vec{r} = \vec{r}_0 + t\,\vec{u}$$

$$(x, y, z) = (x_0, y_0, z_0) + t\,(a, b, c)$$

$$\boxed{\begin{aligned} x &= x_0 + t\,a \\ y &= y_0 + t\,b \\ z &= z_0 + t\,c \end{aligned}}$$

$$t = \frac{(x - x_0)}{a} = \frac{(y - y_0)}{b} = \frac{(z - z_0)}{c}$$



To define a line in 3D, we should know a point (x_0,y_0,z_0) on the line and unit vector (a,b,c) in the direction of line
Therefore 6 parameters to minimize!!!

https://tutorial.math.lamar.edu/classes/calciii/eqnsoflines.aspx

```
x = p[0] + p[1]*t;
y = p[2] + p[3]*t;
z = p[4] + p[5]*t;
```

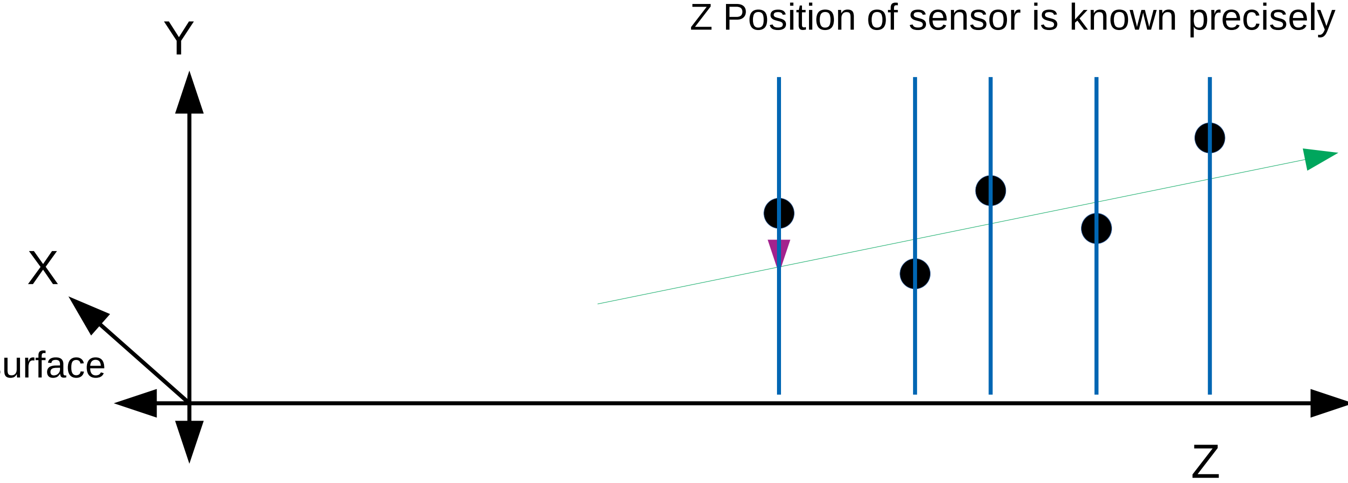
```
x = p[0] + p[1]*t;
y = p[2] + p[3]*t;
z =  t;
```

Our case

P[4] = 0 and p[5] = 1; hit points on the surface

(Chi2 Minimization) Ignoring uncertainity due to MS

Z Position of sensor is known precisely



Ndf: 2 (number of points)-4

Each plane has 2 degrees of freedom (x,y) and lines has 4 independent parameters (constraint)

$$\sigma_x = \sigma_y = 5\,\mu m$$

$$\chi^2 = \sum_{i=1}^{n} \frac{dx_i^2}{\sigma_{x_i}^2} + \frac{dy_i^2}{\sigma_{y_i}^2}$$

```
XYZVector xp(x,y,z);
double ti = z;
XYZVector u(p[0]+p[1]*ti,p[2]+p[3]*ti,z);
double d2 = ((xp-u)).Mag2();
double dx = (xp-u).X(); double dy = (xp-u).Y();
double chi2 = dx*dx/(xerr*xerr)+dy*dy/(yerr*yerr);
return chi2;
```

Corryvreken also using the similar method as given the link of the class

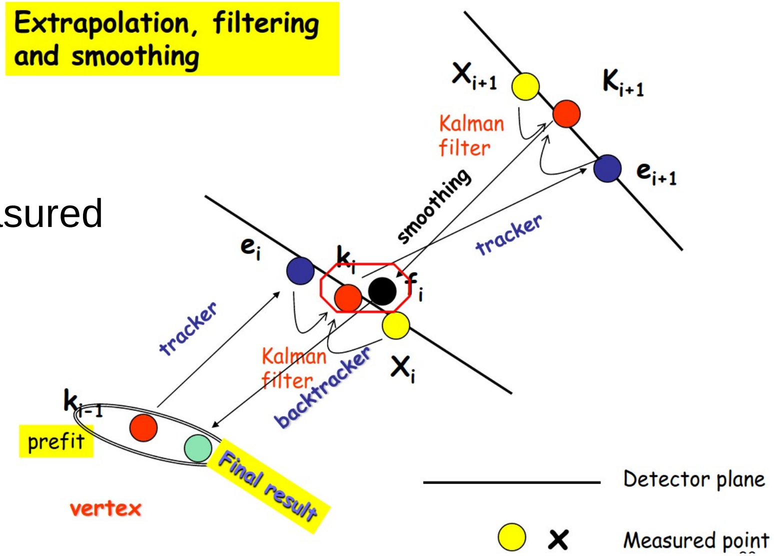
https://gitlab.cern.ch/corryvreckan/corryvreckan/-/blob/master/src/objects/StraightLineTrack.cpp

https://github.com/Simple-Shyam/Phd-work/blob/master/HFPPT/distance.pdf

Track Fitting (Kalman filter)

Trajectory is affected by multiple scattering

1. Extrapolation: Extrapolation (e_i) on the next plane with M.S. effect
2. Filtering: Weighted average of extrapolated value (e_i) and measured value (x_i), known as Kalman filtered value (k_i)
3. Smoothing: Filtering in backward direction



For Multiple scattering we need to know momentum of particle (PYTHIA Simulation)

Kalman filter properly take care of multiple scattering

Class used is AliExternalTrackParam

https://agenda.infn.it/event/1096/contributions/6159/attachments/4504/4980/Rotondi_3.pdf

Thanks to Ruben

More details see back up

Track Fitting (Chi2 vs Kalman filter)

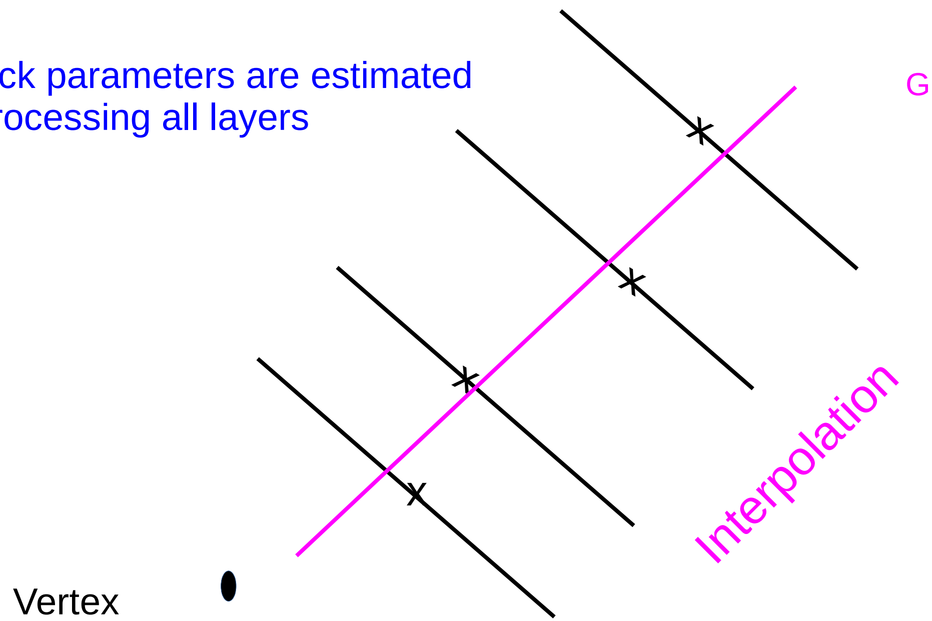
Global Chi2 fitting

- Chi2 fitting is easier for few layers for large number of points covariance matrix is difficult to invert e.g. 200 points it should be 200x200 matrix
- It is very complicated to handle multiple scattering which is correlated among the planes: non-diagonal large matrix
- We get final parameters and errors to extrapolate to any point

Kalman filter

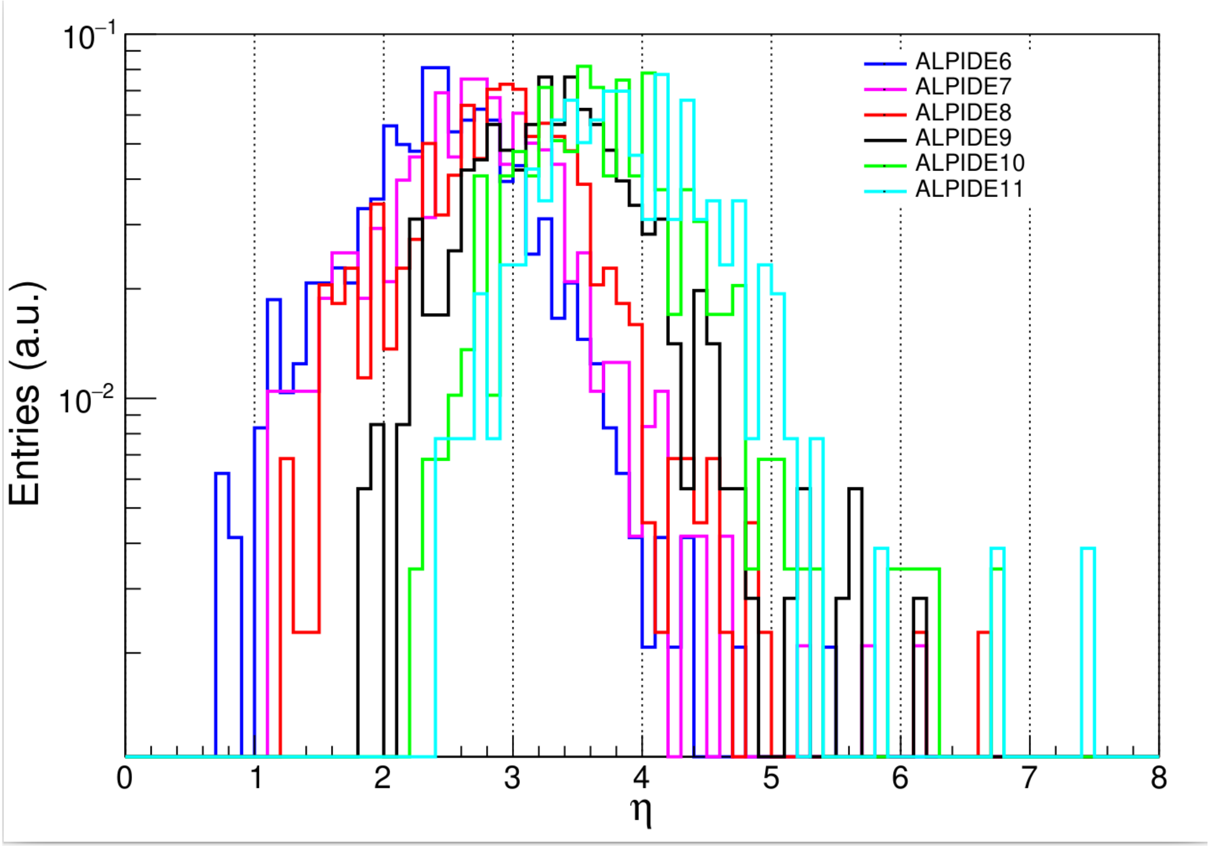
- Kalman filter works as a local fitter for each step we improve information and final parameters are extracted after processing all layers. We always have 5x5 covariance matrix with 5 track parameters
- We do point by point so every step we have extra thetaMS (easier to handle as a Gaussian noise)
- We need to performed forward/backward step, extrapolation of track at long distance increase the uncertainity in extrapolation

Kalman track parameters are estimated processing all layers



Global line fit (we have parameter to interpolate and extrpolate)

Eta Maps of Hits



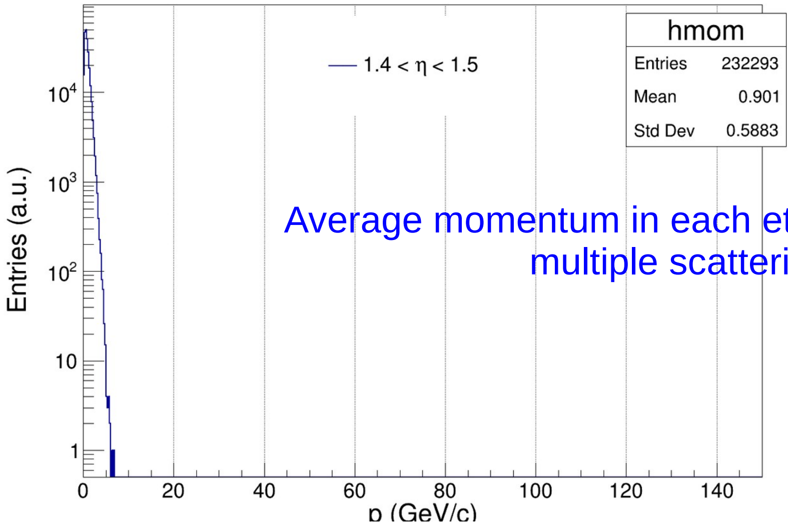
Average momentum in each eta bin is required to treat M.S. properly

Eta maps extracted using Reconstructed Hits assuming vertex at origin

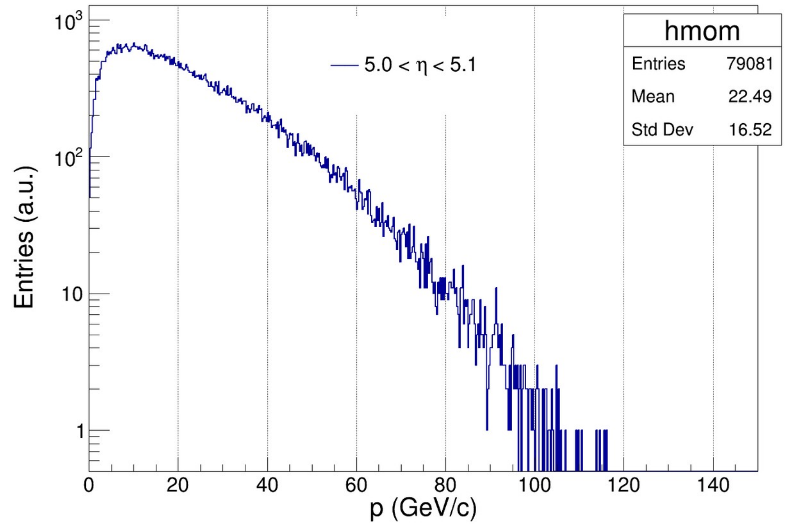
Proper treatement of M.S. requires the knowlege of average momentum in each eta bin: PYTHIA Simulation (Angantyr model) thanks to F. Colamaria

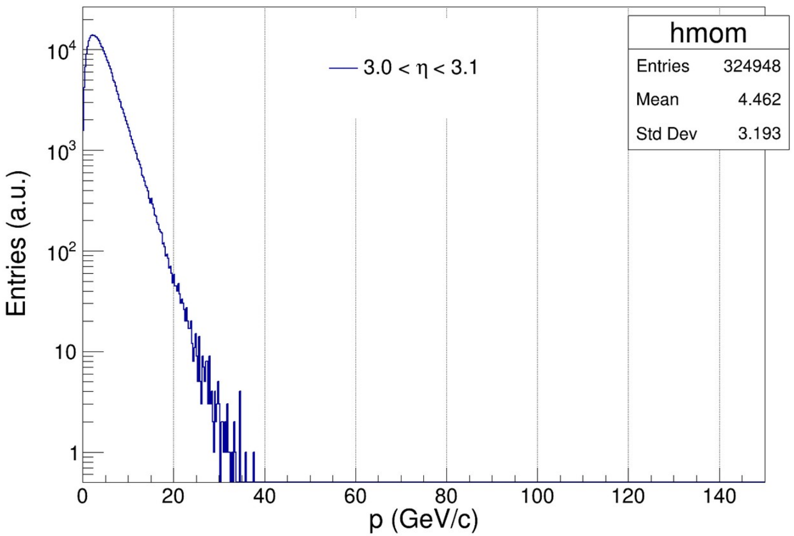
PYTHIA Simulation (Angantyr model)

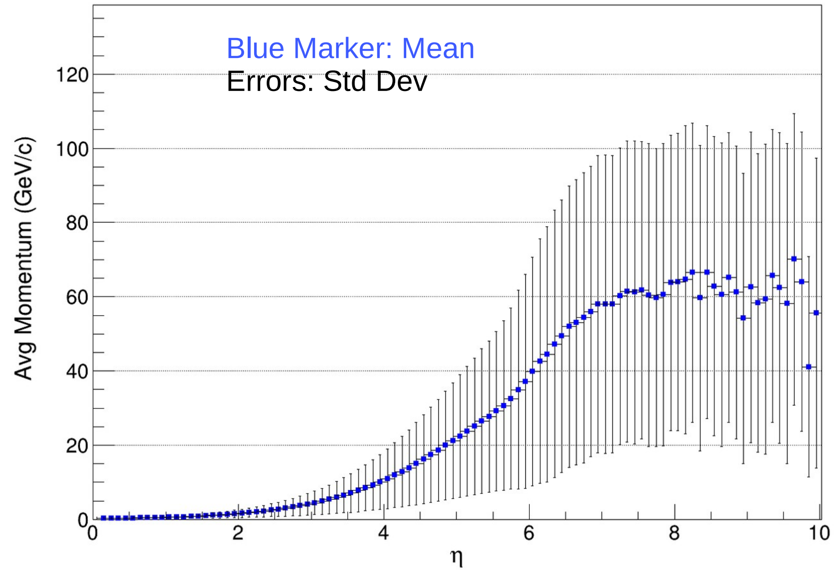
120 GeV proton on Copper Nucleus



Average momentum in each eta bin required for multiple scattering







Track Fitting Method (ALICE)

Tracking frame is local frame

X local is normal to the sensor

Y local is on the sensor

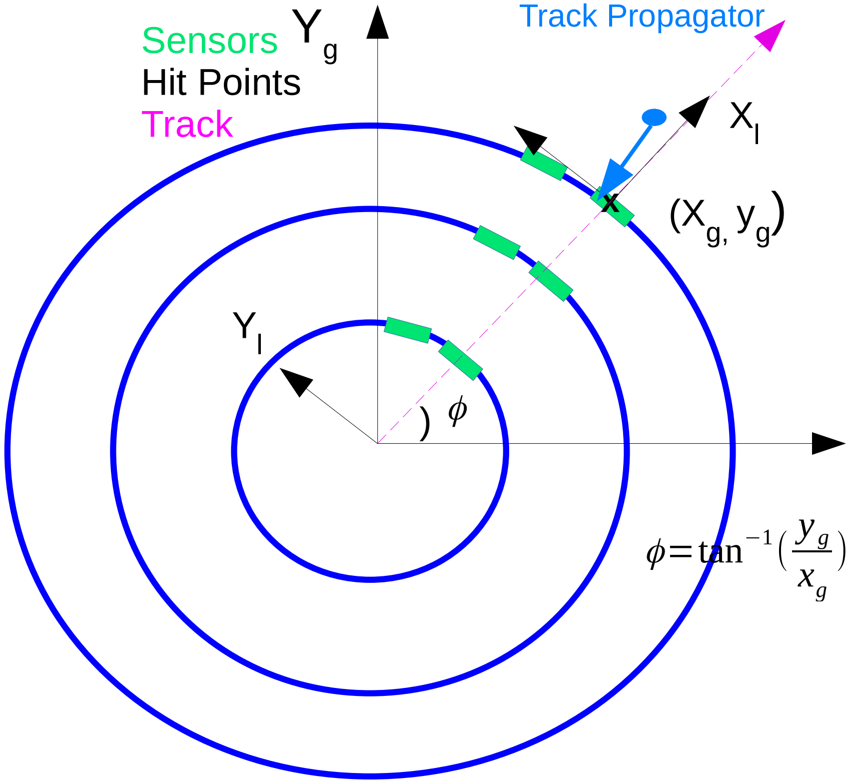
In this coordinate system:

Track Parameters (Local):

$$(y_l, z_l, \sin\phi, \tan\lambda, q/p_T)$$

- ylocal and zlocal will describes the sensor surface and uncertainties are σ(rφ) and on zlocal is σ(z) respectively (x_l becomes radius)

```
root [0] TVector2 a(2.,3.);
root [1] a.Print();
TVector2 A 2D physics vector (x,y)=(2.000000,3.000000)
(rho,phi)=(3.605551,56.309932)
root [2] TVector2 rot_a = a.Rotate(-a.Phi());
root [3] rot_a.Print()
TVector2 A 2D physics vector (x,y)=(3.605551,-0.000000)
(rho,phi)=(3.605551,360.000000)
```



Two things required: Extrapolation with MS and Measured Points

Common Steps:

- Track Model initialize with the last point x+Margin (0.1)
- Convert last point to local coordinate system rotated by phi.
- Extrapolate track to the last layer
- Rotate Track to local coordinate system
- Update extrapolation and measurement (weighted average of position and errors
- Multiple scattering correction

Repeat above steps for each layer up to layer 0 and then extrapolate to the vertex

Track Extrapolation

Bool_t PropagateTo(Double_t x, Double_t b);

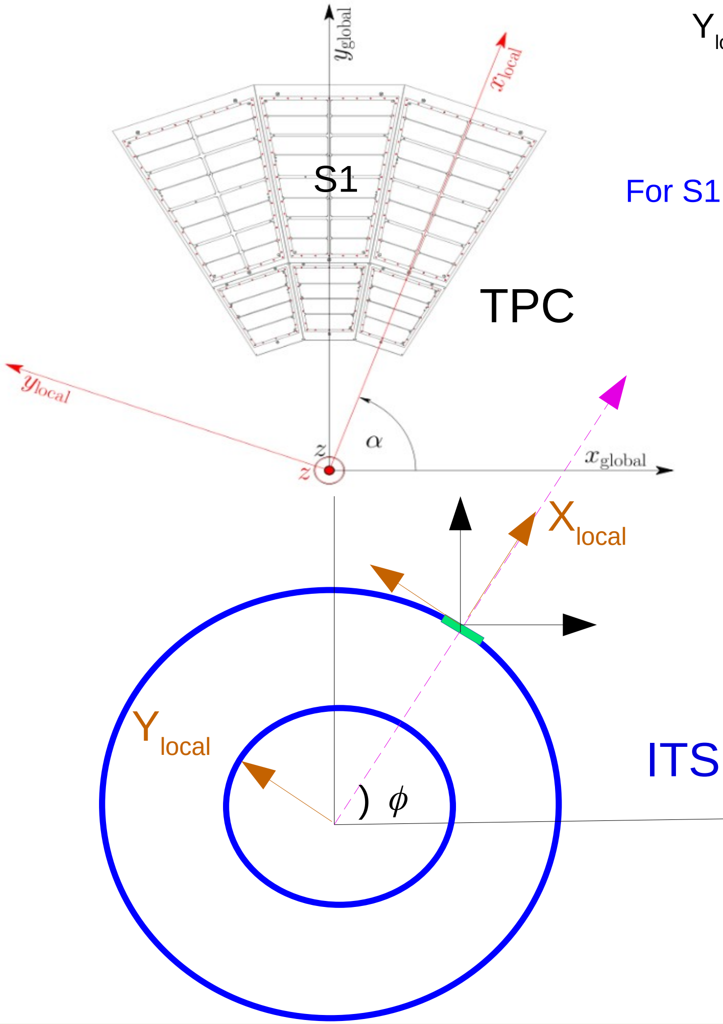
Bool_t CorrectForMeanMaterialdEdx(Double_t **xOverX0Si**, Double_t 0, Double_t mPion, Bool_t anglecorr=kTRUE);

At the vertex x_l = 0 and y_l is DCA_{xy}

Track Fitting (Kalman Filter)

Track Parameters (Local): $\left(y_l, z_l, \sin\phi, \tan\lambda, q/p_T\right)$

TPC single sector is a planar detector



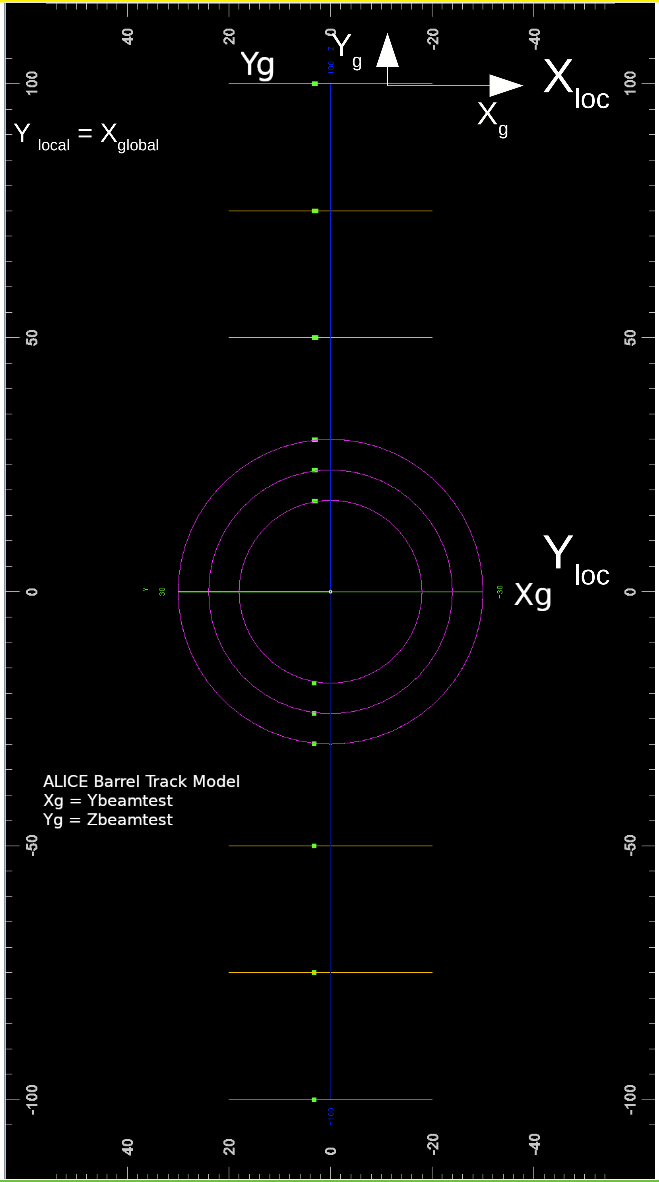
Y_{local} and Z_{local} describes sensor

For S1 alpha should be 90^0 (Our case of planes)

$X_{local} = Y_{global}$

$Y_{local} = X_{global}$

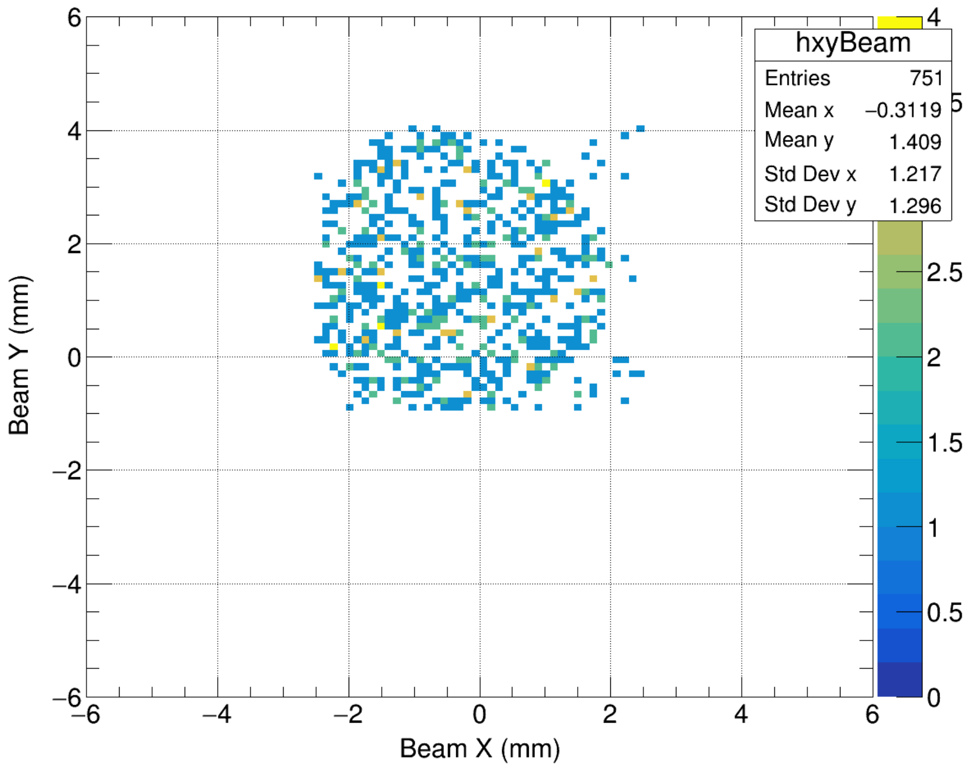
With B = 0



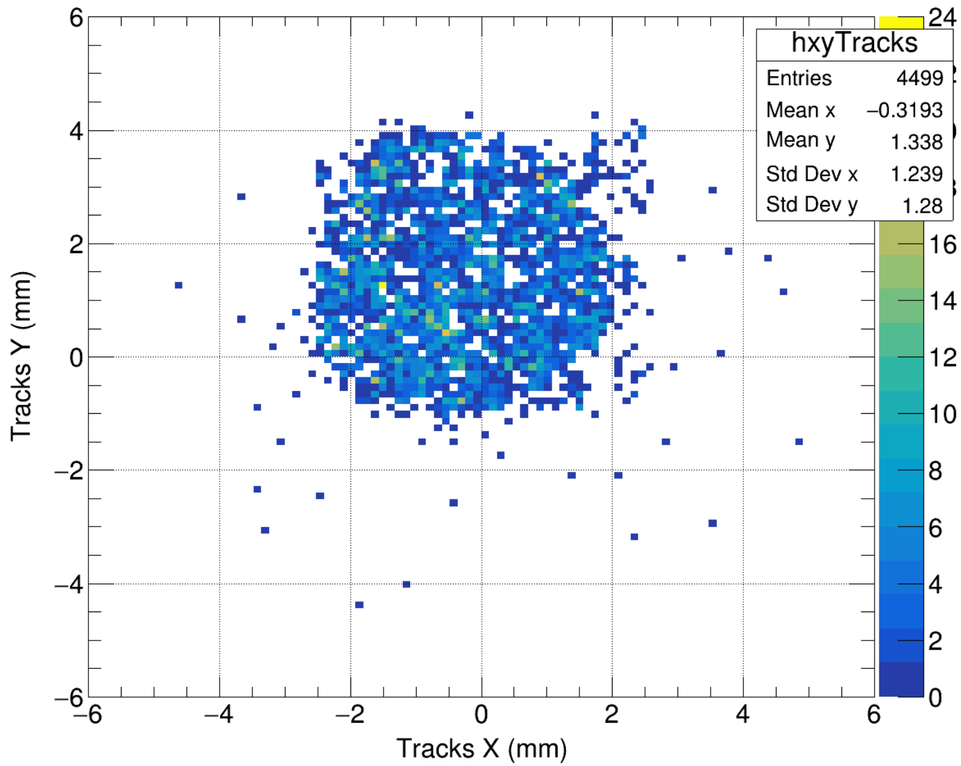
Thanks Ruben

Fitting Beam and Tracks with Kalman filter

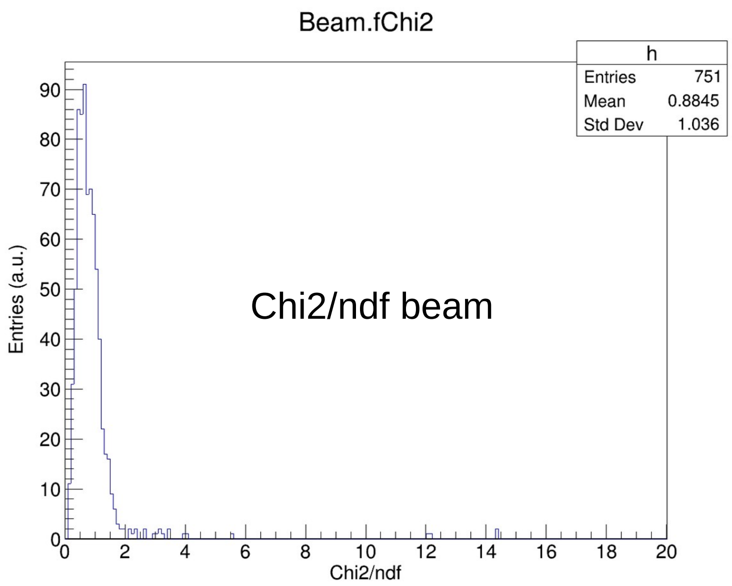
Extrapolation of Beam at origin

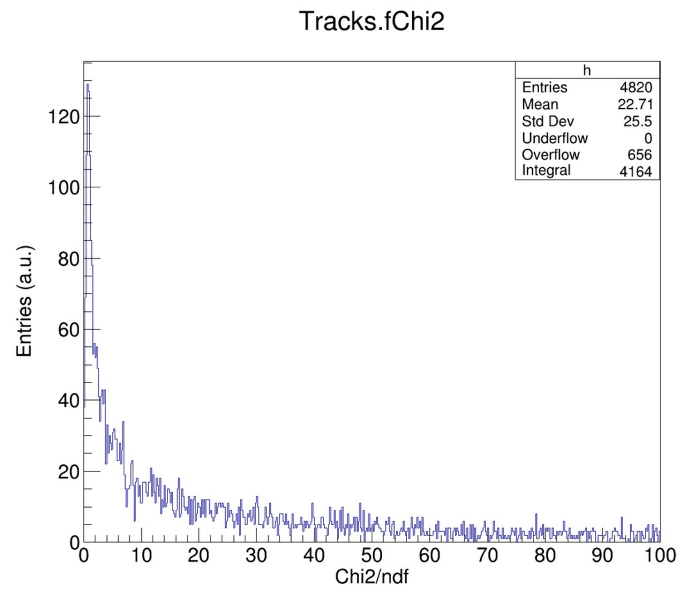


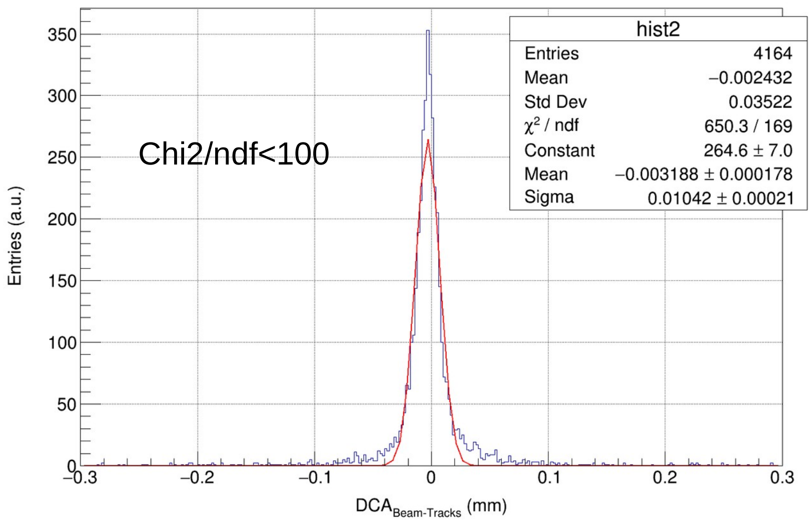
Extrapolation of Tracks at origin

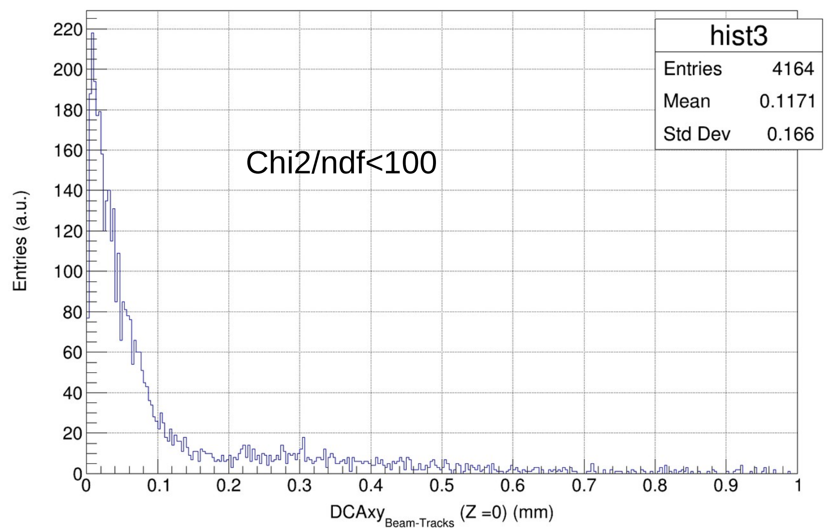


Track Fitting Results (Global Chi2 Fitting)

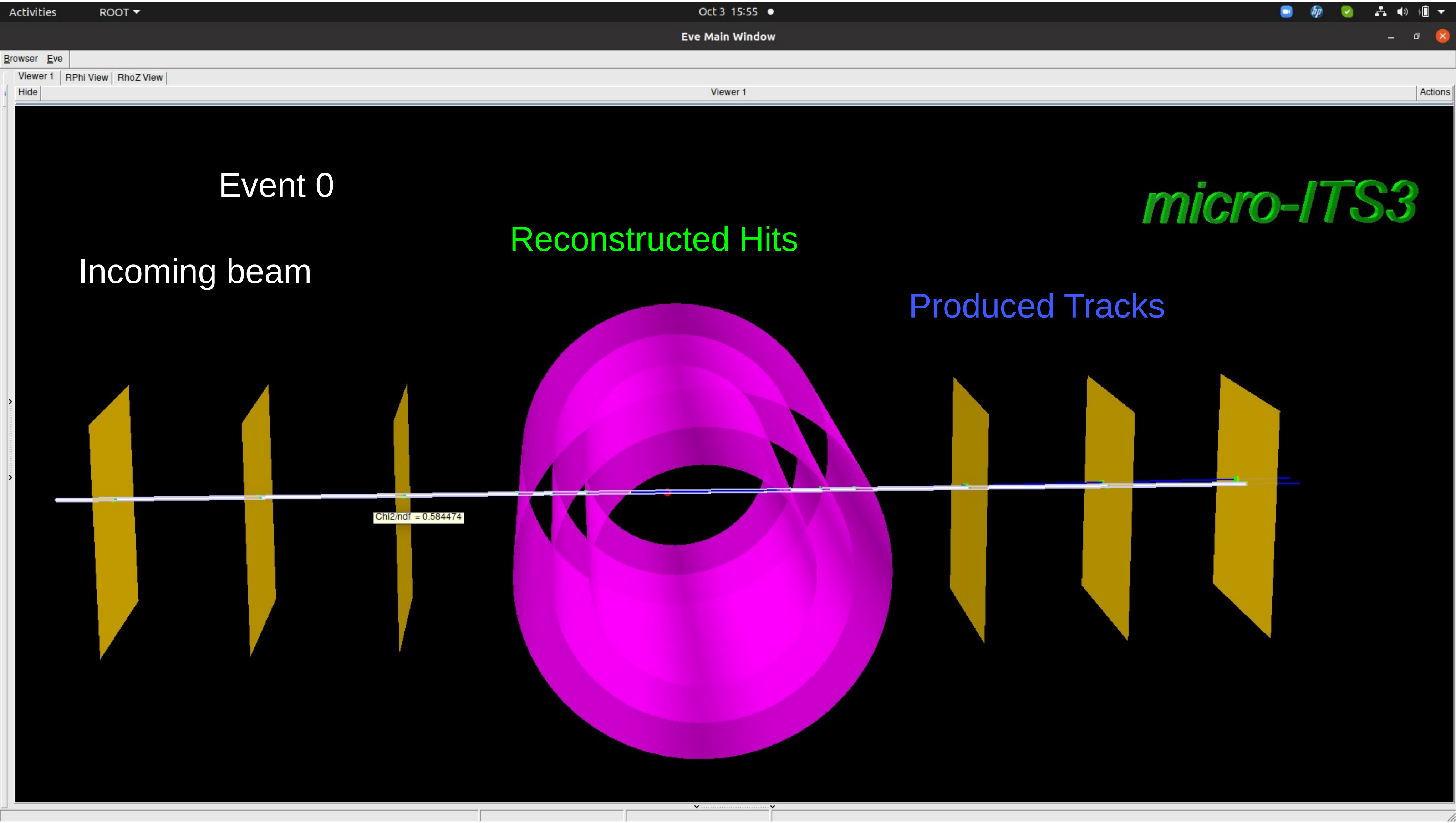




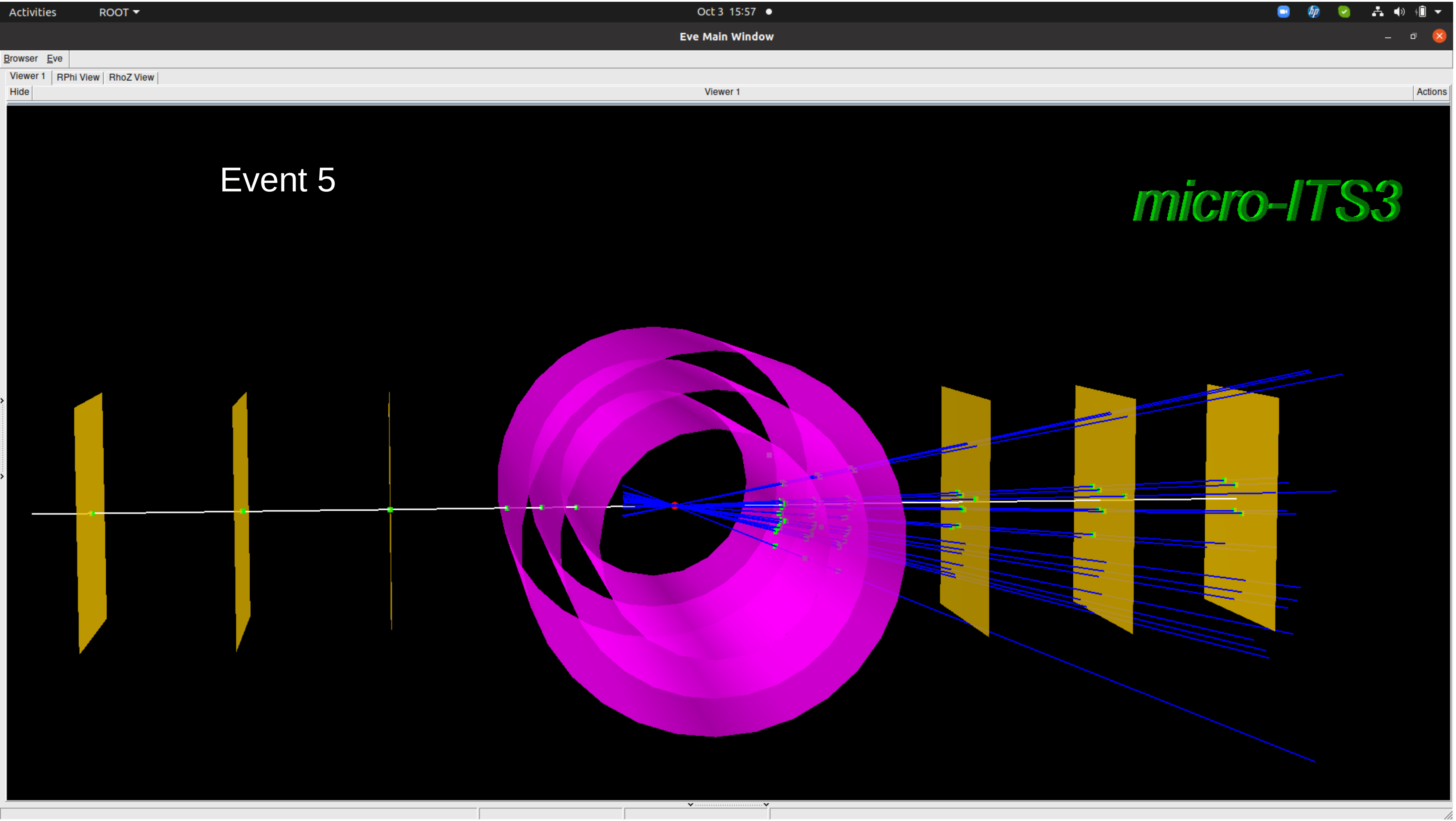




Event display



Event display



Definition of Chi2 (Kalman filter)

$$\chi^2 = (Y_{meas} - Y_{fit})^T W^{-1} (Y_{meas} - Y_{fit})$$

If $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ then

$$A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

fP: filtered value of track
p: measurement

In our case

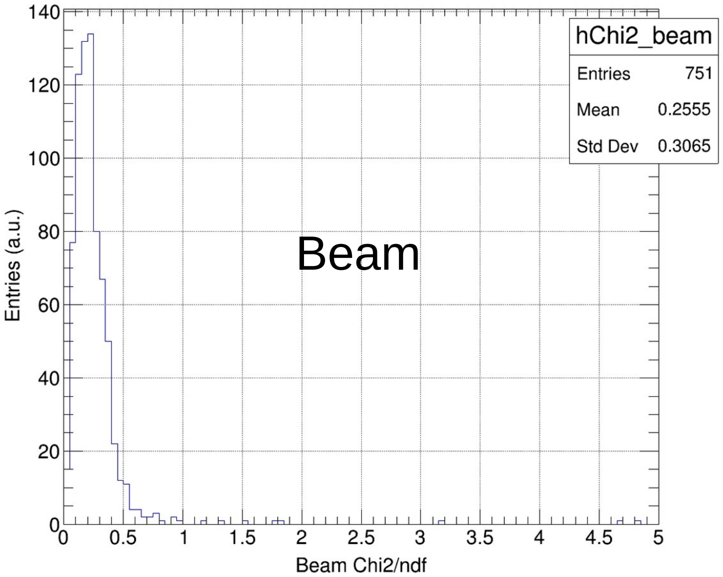
$$\chi^2 = (fP[0] - p[0], fP[1] - p[1]) \begin{pmatrix} s_{dd} & s_{dz} \\ S_{dz} & s_{zz} \end{pmatrix}^{-1} \begin{pmatrix} fP[0] - p[0] \\ fP[1] - p[1] \end{pmatrix}$$

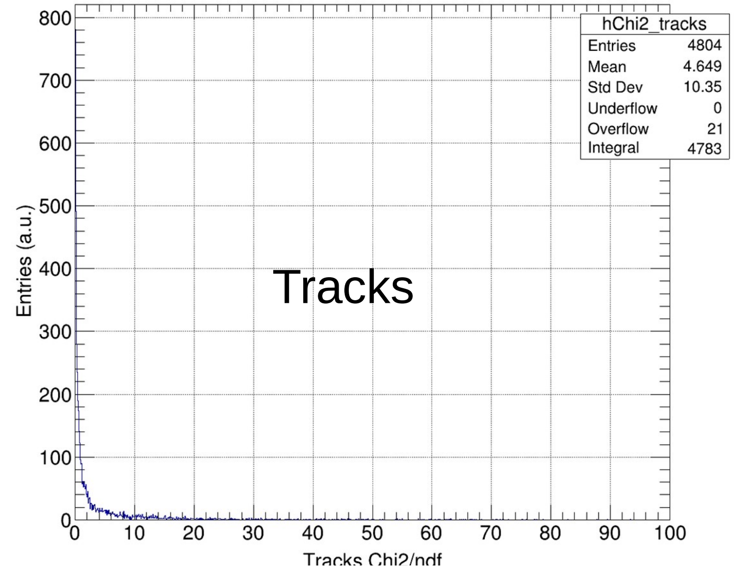
```
Double_t
AliExternalTrackParam::GetPredictedChi2(const Double_t p[2],const Double_t cov[3]) const {
  //------------------------------------------------------------------
  // Estimate the chi2 of the space point "p" with the cov. matrix "cov"
  //------------------------------------------------------------------
  Double_t sdd = fC[0] + cov[0];
  Double_t sdz = fC[1] + cov[1];
  Double_t szz = fC[2] + cov[2];
  Double_t det = sdd*szz - sdz*sdz;

  if (TMath::Abs(det) < kAlmost0) return kVeryBig;

  Double_t d = fP[0] - p[0];
  Double_t z = fP[1] - p[1];

  return (d*szz*d - 2*d*sdz*z + z*sdd*z)/det;
}
```

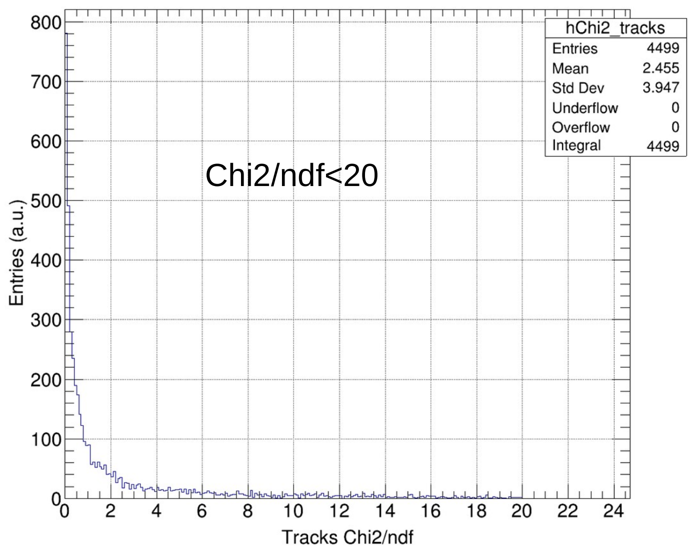
$$\boxed{\chi^2 = (d, z) \frac{1}{s_{dd} * s_{zz} - sdz * sdz} \begin{pmatrix} s_{zz} & -s_{dz} \\ -S_{dz} & s_{dd} \end{pmatrix} \begin{pmatrix} d \\ z \end{pmatrix}}$$

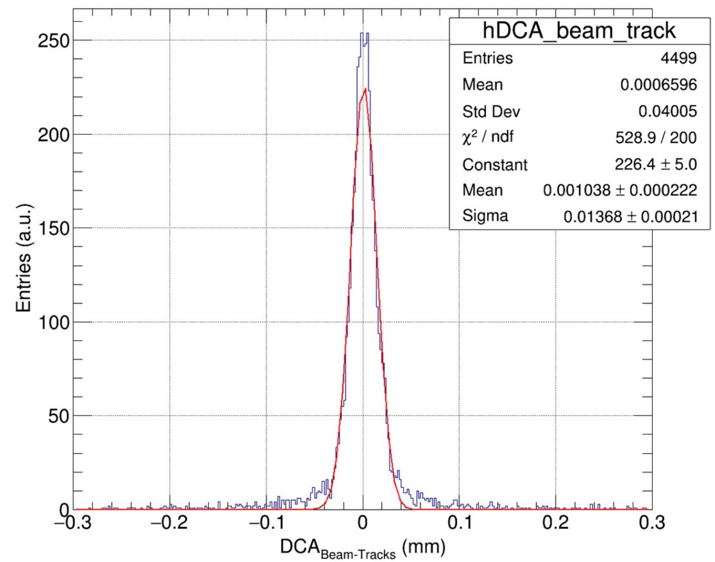
Ndf: 2 (number of points)-4

Track Fitting Results (Kalman Filter)

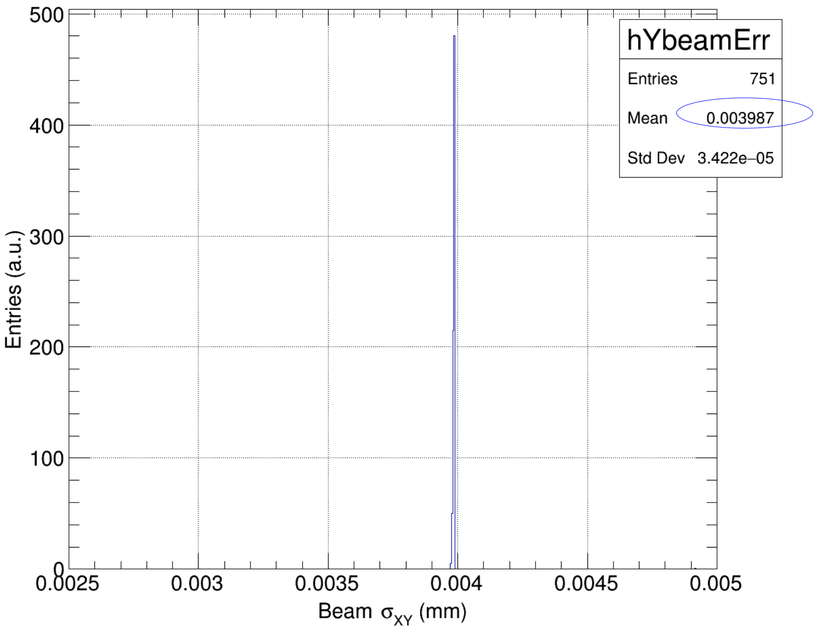


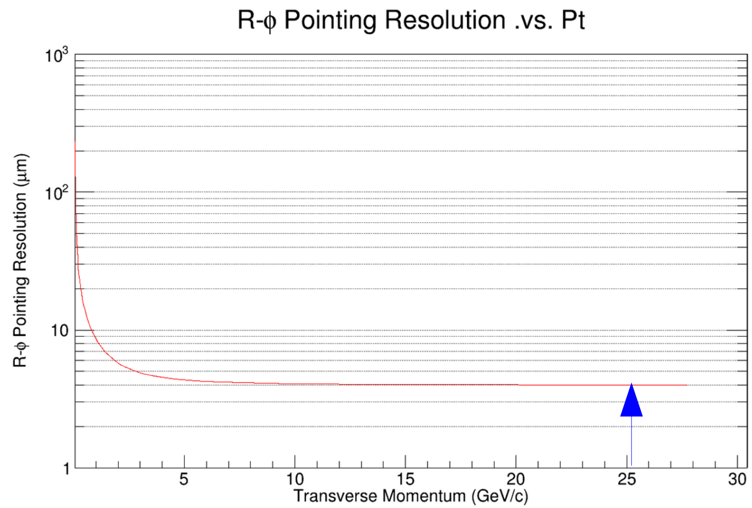






```
Detector BeamTest: "Detector"
  Name              r [cm]           X0       phi & z res [um] layerEff
0. vertex           0.00          0.0000        -     -        -
1. VTX0             1.80          0.0005        5     5      1.00
2. VTX1             2.40          0.0005        5     5      1.00
3. VTX2             3.00          0.0005        5     5      1.00
4. VTX3             5.00          0.0005        5     5      1.00
5. VTX4             7.50          0.0005        5     5      1.00
6. VTX5            10.00          0.0005        5     5      1.00
```

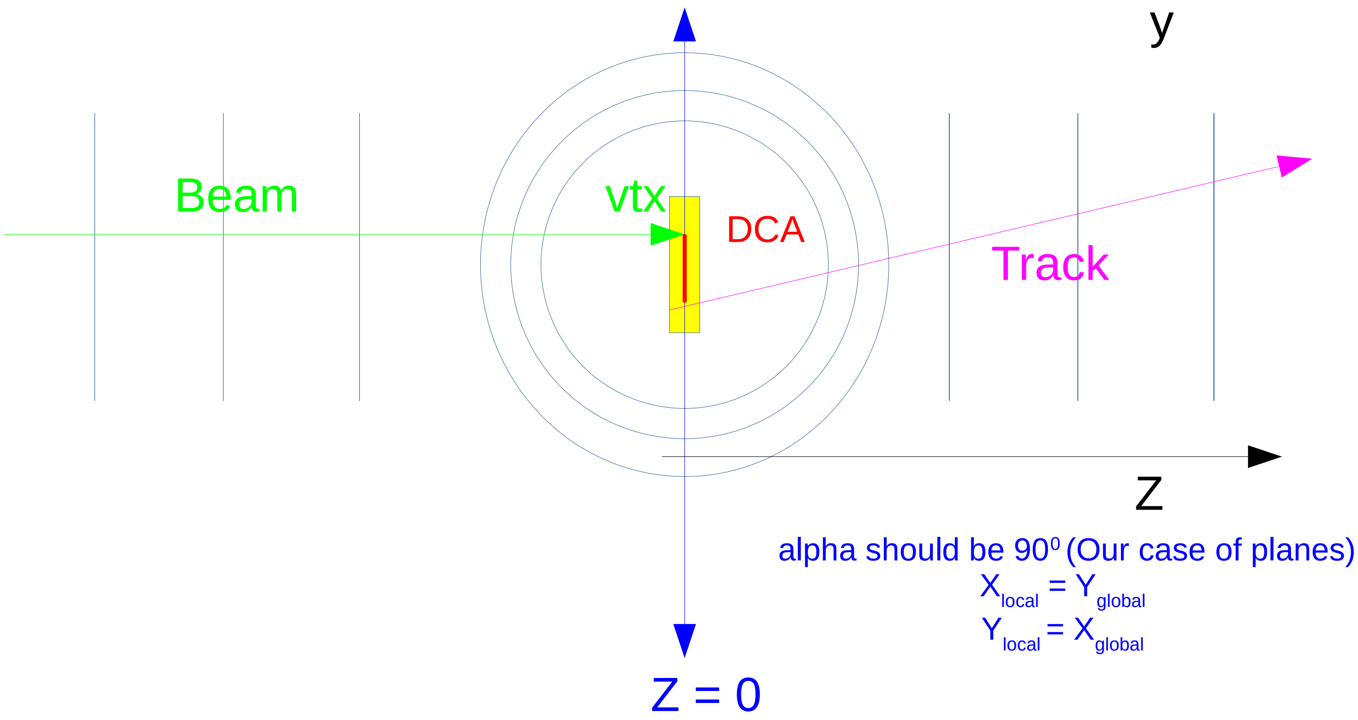




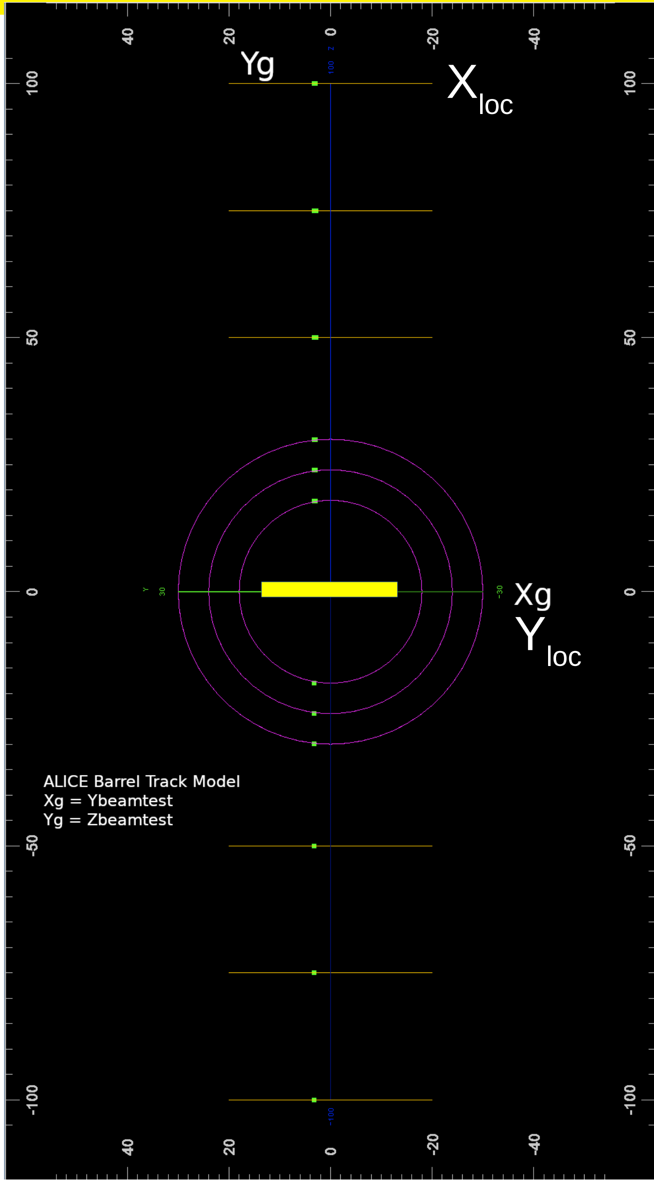
It comes 5 μm if we use equal distance

- Extrapolation of beam at the origin and error on DCAxy is compatible with fast simulation (~4 μm)

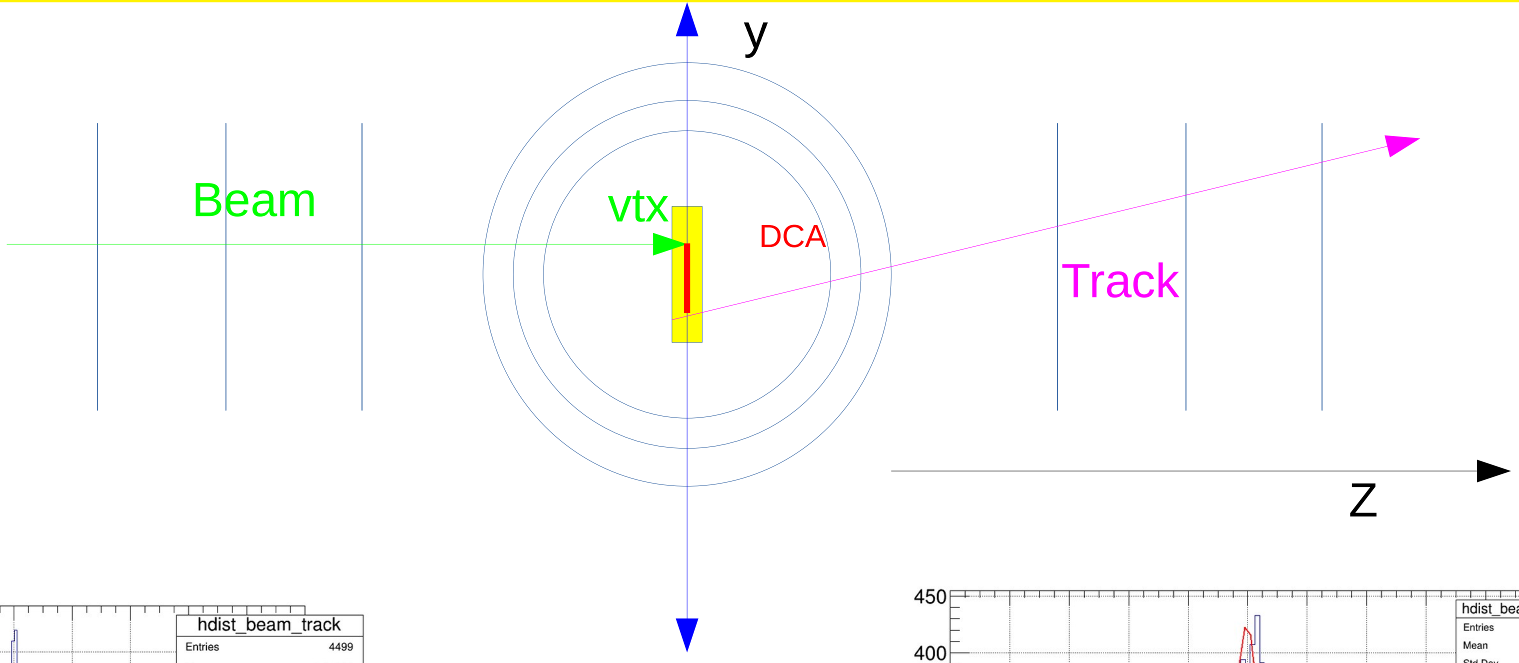
Track Fitting Results (Kalman Filter)

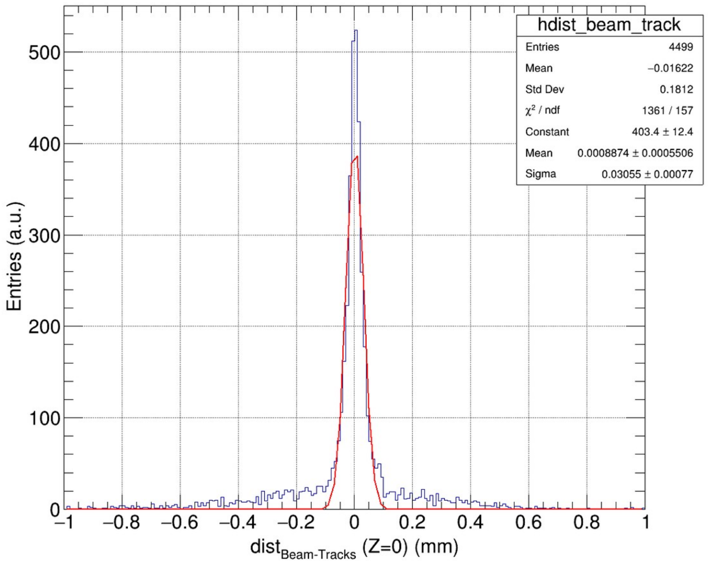


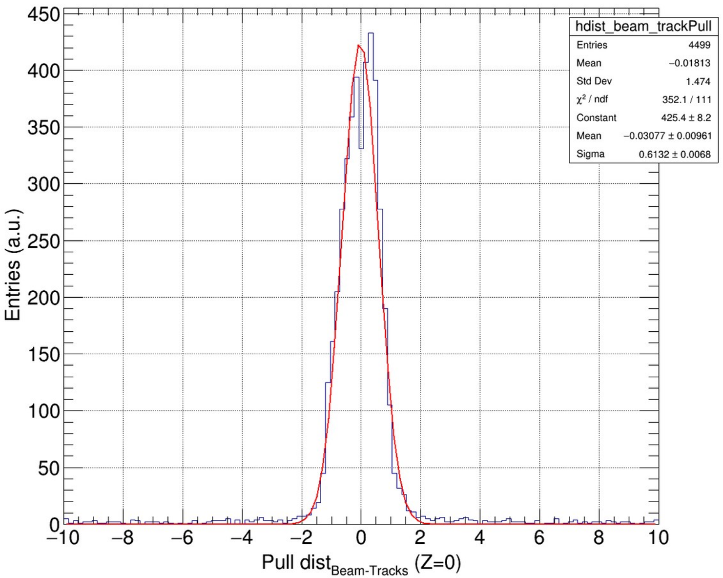

```
Bool_t AliExternalTrackParam::PropagateToDCA(const AliVVertex *vtx,
Double_t b, Double_t maxd, Double_t dz[2], Double_t covar[3]) {
  //
  // Propagate this track to the DCA to vertex "vtx",
  // if the (rough) transverse impact parameter is not bigger then "maxd".
  //            Magnetic field is "b" (kG).
  //
  // a) The track gets extapolated to the DCA to the vertex.
  // b) The impact parameters and their covariance matrix are calculated.
  //
  //    In the case of success, the returned value is kTRUE
  //    (otherwise, it's kFALSE)
  //
```



Track Fitting Results (Kalman Filter)

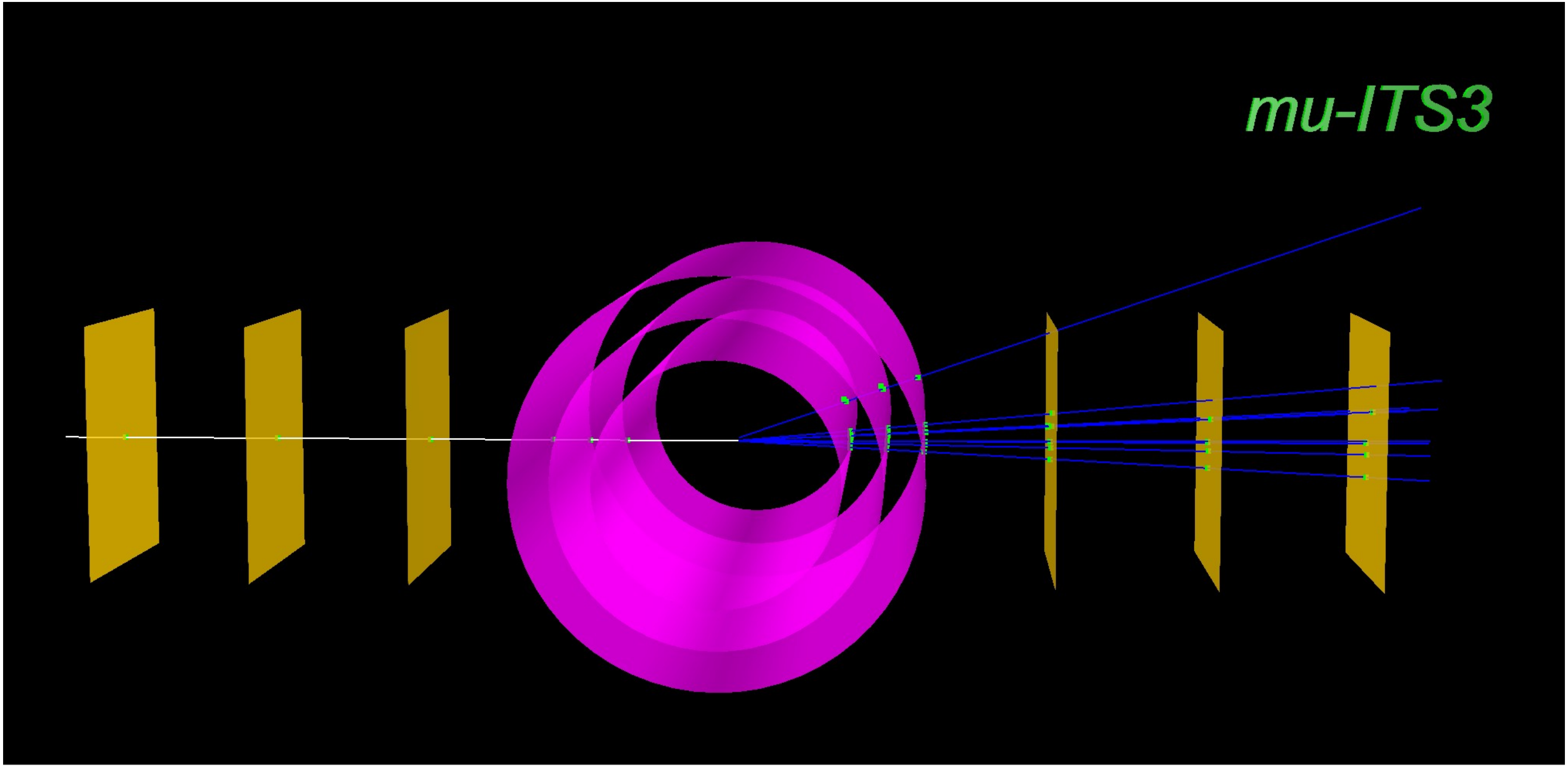






Track Fitting Results (Kalman Filter)

Event display



Summary & Future Plan

- Track finding is fixed and working fine.
- Track fitting is done using Global Chi2 fitting and Kalman filter method
- DCA between beam and tracks are evlauted using two methods
- Distance between beam and tracks and also pull distribution evaluated at z =0 and will visualize in event display

Detector Simulation:

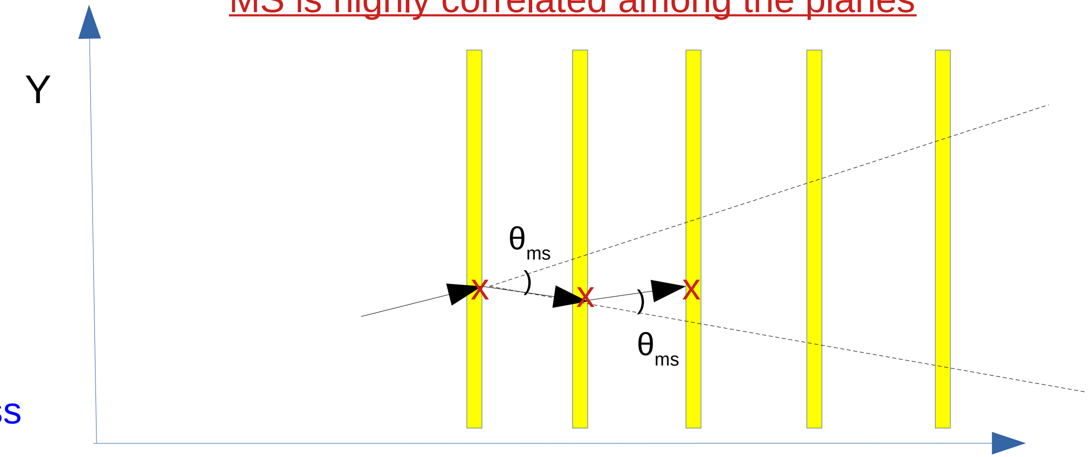
Simulation: Red Points we get during detector simulation considering energy loss effect and multiple scattering

Digitization: Smear these red point by pixel resolution (spatial resolution)

Reconstruction: Fit the point after digitization

In Beam test data, the points already include energy loss effect, multiple scattering, and spatial resolution

MS is highly correlated among the planes



Multiple scattering is highly correlated among the planes (Covariance matrix is non-diagonal)

Evaluate Residual (R) and Full covariance matrix for proper treatment of multiple scattering in the fit

$$\boxed{\chi^2 = R^T W^{-1} R}$$

Let's understand lines in 3D for fitting them

Vector equation of a line in 3D:

$$\vec{r} = \vec{r}_0 + \vec{a}$$

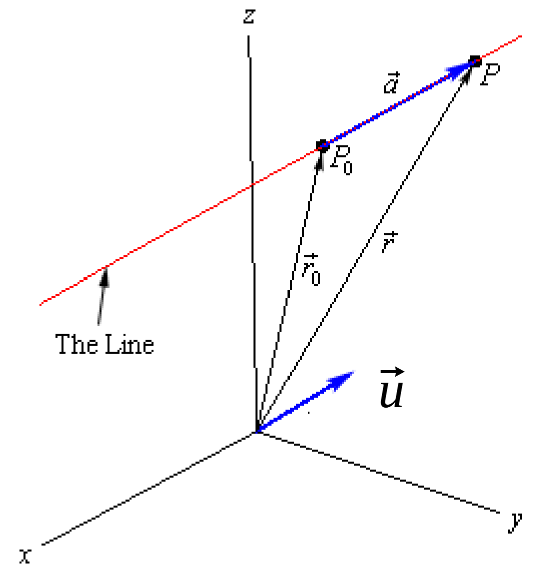
If u is the unit vector along the line and t is parameter:

$$\vec{r} = \vec{r}_0 + t\,\vec{u}$$

$$(x, y, z) = (x_0, y_0, z_0) + t\,(a, b, c)$$

$$x = x_0 + t\,a$$
$$y = y_0 + t\,b$$
$$z = z_0 + t\,c$$

$$t = \frac{(x - x_0)}{a} = \frac{(y - y_0)}{b} = \frac{(z - z_0)}{c}$$



To define a line in 3D, we should know a point (x_0,y_0,z_0) on the line and unit vector (a,b,c) in the direction of line
Therefore 6 parameters to minimize!!!

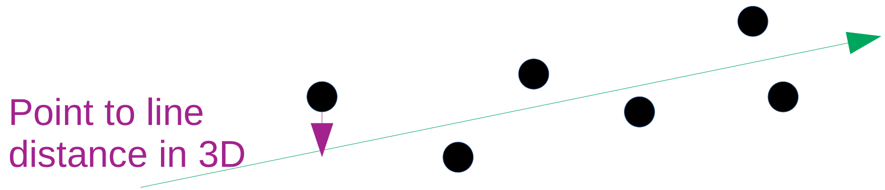
https://tutorial.math.lamar.edu/classes/calciii/eqnsoflines.aspx

Fitting Lines in 3D (General)

General line in 3D with 6 parameters:

$$x = p_0 + t\, p_1$$
$$y = p_2 + t\, p_3$$
$$z = p_4 + t\, p_5$$

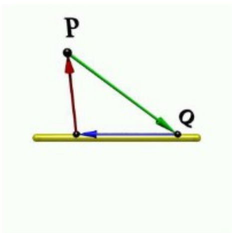
We have given hit points (x_0, y_0, z_0), (x_1, y_1, z_1).....
Minimize of Chi2 to best estimate the parameters



DISTANCE POINT-LINE (3D). If P is a point in space and L is the line $\vec{r}(t) = Q + t\vec{u}$, then

$$d(P, L) = \frac{|(\vec{PQ}) \times \vec{u}|}{|\vec{u}|}$$

is the distance between P and the line L. Proof: the area divided by base length is height of parallelogram.



Here we are minimizing distance correspond to a general point

```
// define the parametric line equation
void line(double t, const double *p, double &x, double &y, double &z) {
          // a parametric line is define from 6 parameters but 4 are independent
          // x0,y0,z0,z1,y1,z1 which are the coordinates of two points on the line
          x = p[0] + p[1]*t;
          y = p[2] + p[3]*t;
          z = p[4] + p[5]*t;
}
```

```
 // where ux is direction of line and x0 is a point in the line (like t = 0)
 XYZVector xp(x,y,z);
 XYZVector x0(p[0], p[2], p[4])
// distance line point is D= | (xp-x0) cross  ux |
```

https://github.com/Simple-Shyam/Phd-work/blob/master/HFPPT/distance.pdf
https://root.cern.ch/doc/master/line3Dfit_8C.html

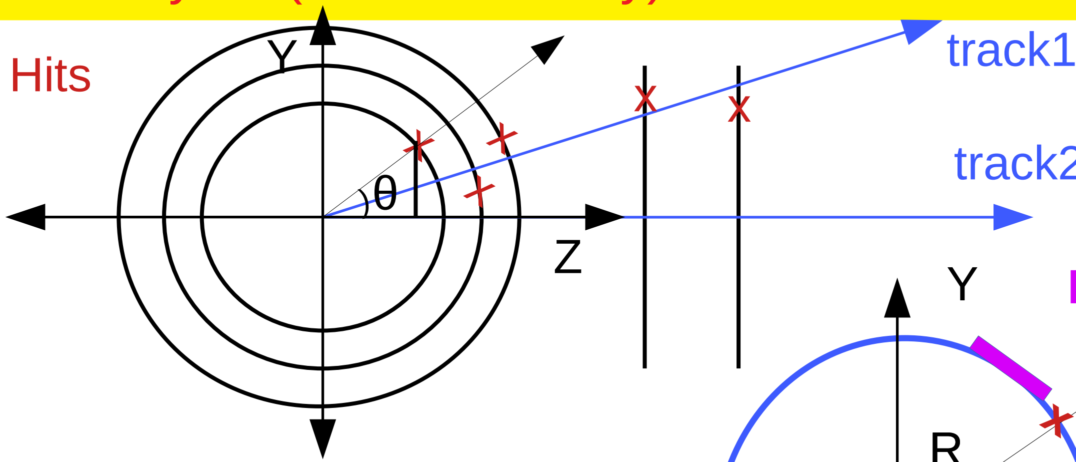
ALPIDE Layers (See Geometry)

ALPIDE Planes:

For Planes

$$\chi^2 = \frac{dx^2}{\sigma_x^2} + \frac{dy^2}{\sigma_y^2}$$

$$\sigma_x = \sigma_y = 5\,\mu m$$



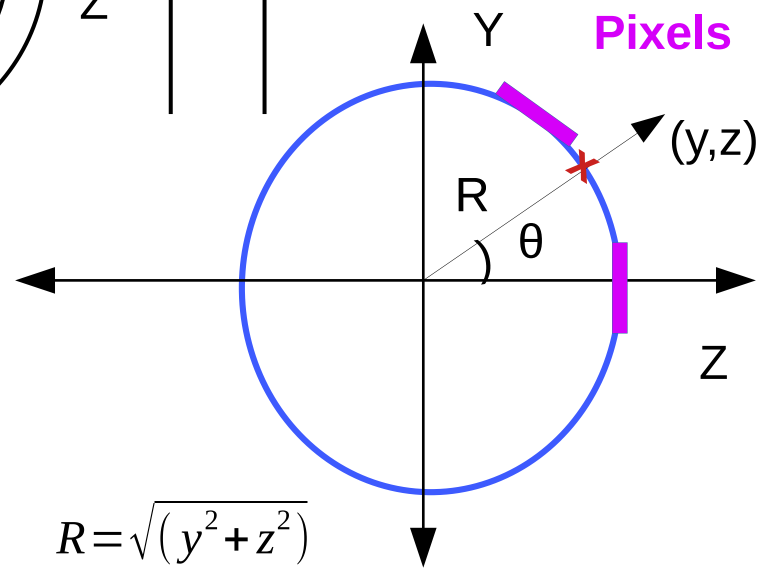
ALPIDE Cylinderical:

σ_x will be the same

$$\chi^2 = \frac{dx^2}{\sigma_x^2} + \frac{dy^2}{\sigma_{y'}^2}$$

For Cylinderical Layers:

$$\sigma_x = \sigma_{R\theta} = 5\,\mu m$$



$$R = \sqrt{(y^2 + z^2)}$$

$$\theta = \tan^{-1}\left(\frac{y}{z}\right)$$

$$\sigma_y' = \sigma_{R\theta} \cos\theta$$

$$z = R\cos\theta \qquad y = R\sin\theta$$

$$dy = R\cos\theta \;\; d\theta$$

$$dy = R\cos\theta \;\; d\theta = d(R\theta)\cos\theta$$

Sensor frame will be helpful

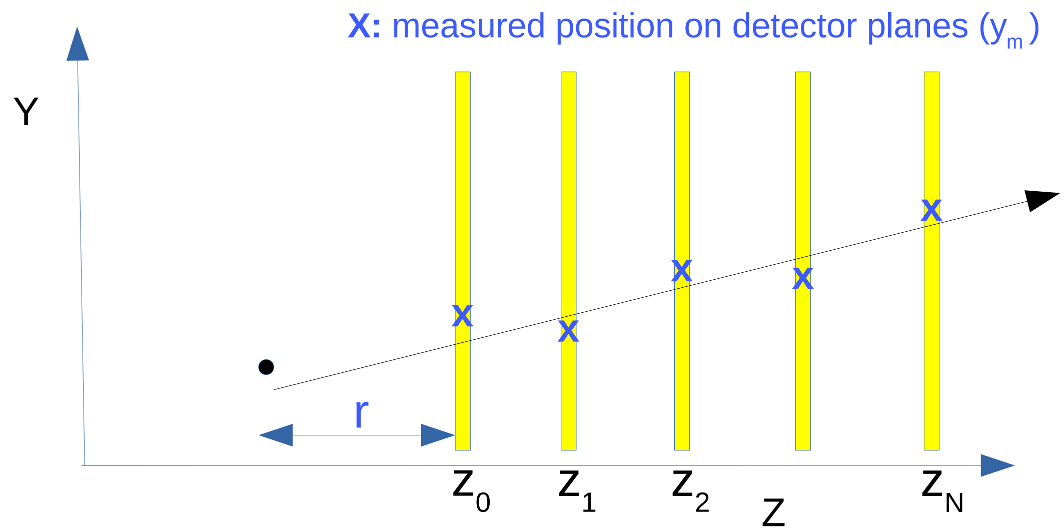
Straight Line Fit (With Multiple Scattering)

$$y = a + b z$$

If points on planes are uncorrelated

Minimize the quantity below (works for Spatial resolutions):

$$\chi^2 = \sum_{i=0}^{N} \frac{(y_m - y_i)^2}{\sigma_i^2} = \sum_{i=0}^{N} \frac{(y_m - a - bz_i)^2}{\sigma_i^2}$$



Multiple Scattering between planes are highly correlated, then quantity to be Minimized:

$$\chi^2 = (Y - Ap)^T (V_{SR} + V_{MS})^{-1} (Y - Ap)$$

$$\chi^2 = \sum_{i,j=0}^{N} \frac{(y_{m_i} - y_i)(y_{m_j} - y_j)}{\sigma_{ij}}$$

$$V_{SR} = \begin{pmatrix} \sigma_0^2 & \ldots\ldots \\ 0 & \sigma_1^2 \ldots\ldots \\ \ldots\ldots & \ldots\ldots\ldots \\ 0 & 0\ldots\ldots\ \sigma_N^2 \end{pmatrix} \qquad V_{MS} = \begin{pmatrix} 0 & \sigma_{01\ldots\ldots} \\ 0 & \sigma_1^2 \ldots\ldots \\ \ldots\ldots & \ldots\ldots\ldots \\ 0 & 0\ldots\ldots\ \sigma_N^2 \end{pmatrix}$$

For 100 points:
100x100 matrix difficult to Inverse (Chi2 fitting)
For Kalman filter 100 matrix of 5x5 dimensions

MS matrix is non-diagonal

Local Coordinate ALICE

Tracking frame is Sensor local frame

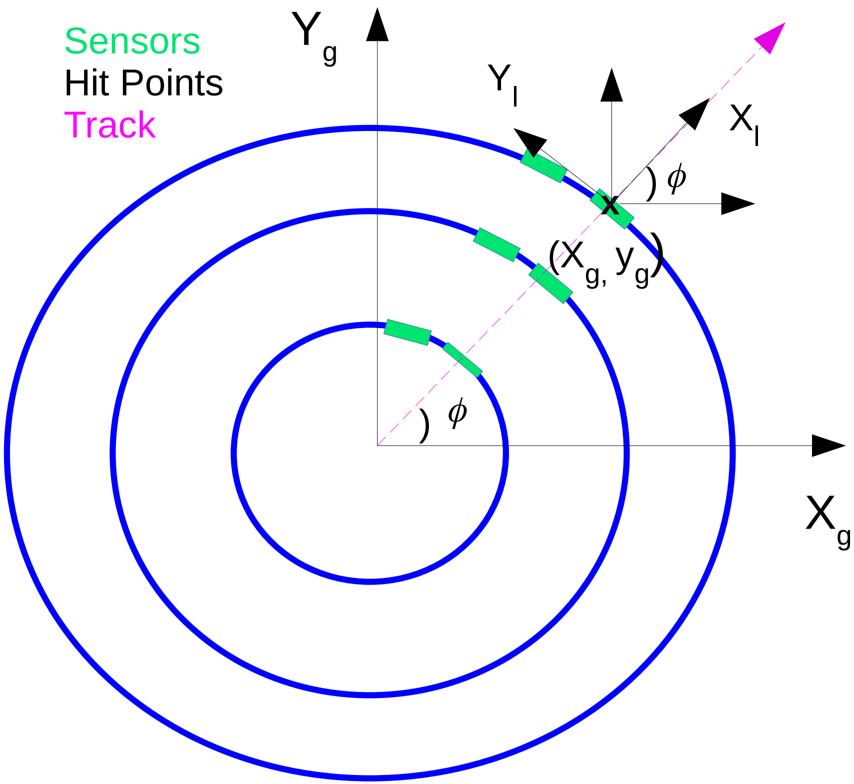
$$\phi = \tan^{-1}\left(\frac{y_g}{x_g}\right)$$

X local is normal to the sensor

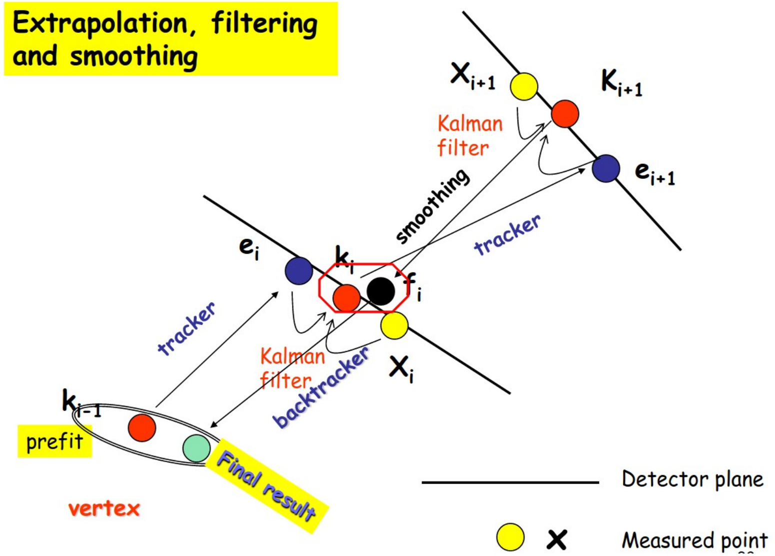
Y local is on the sensor

In this coordinate system: ylocal and zlocal will describes the sensor

Uncertainties in ylocal is σ(rφ) and on zlocal is σ(z)
x_l is describing the radius



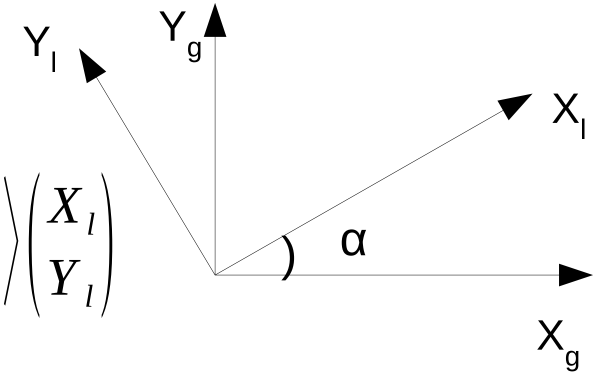
Track Parameters (Local): $\left(y_l, z_l, \sin\phi, \tan\lambda, q/p_T\right)$



Kalmn Filter Method:

1. Extrapolation: Extrapolation (e_i) on the next plane with M.S. effect
2. Filtering: Weighted average of extrapolated value (e_i) and measured value (x_i), known as Kalman filtered value (k_i)
3. Smoothing: Estimation of parameters to extrapolate

Track Parameters (Local): $\left(y\,,z\,,\sin\phi,\tan\lambda\,,q/p_T\right)$



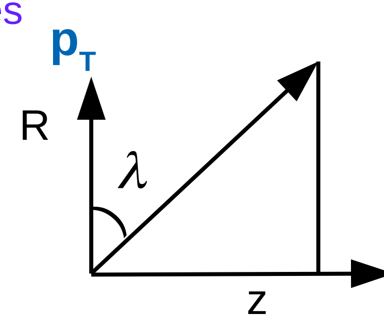
On cylindrical surface:

$$x^2+y^2=R^2$$

$$\begin{pmatrix} X_g \\ Y_g \end{pmatrix} = \left\langle \begin{matrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{matrix} \right\rangle \begin{pmatrix} X_l \\ Y_l \end{pmatrix}$$

Local coordinates rotated by α w.r.t. global coordinates

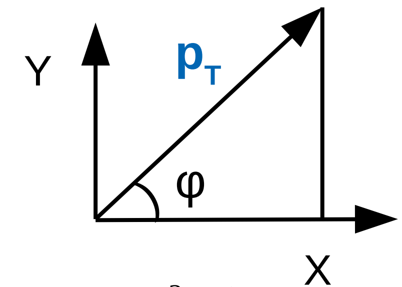
$$\frac{\sigma_{1/p_t}}{(1/p_t)} = \frac{1/p_t^2 * \sigma_{pt}}{(1/p_t)} = \frac{\sigma_{pt}}{p_t} \qquad \sigma^2_{DCA_{r\phi}} = \sigma^2_y \qquad \sigma^2_{DCA_z} = \sigma^2_z$$



Parameter Covariance

Track Parameters

$$\begin{pmatrix} y \\ z \\ \sin\phi \\ \tan\lambda \\ 1/p_T \end{pmatrix} \qquad W = \begin{pmatrix} \boxed{\sigma^2_y} & \sigma_{yz} & \sigma_{y\sin\phi} & \sigma_{y\tan\lambda} & \sigma_{y.1/p_T} \\ \sigma_{zy} & \boxed{\sigma^2_z} & \sigma_{z\sin\phi} & \sigma_{z\tan\lambda} & \sigma_{z.1/p_T} \\ \sigma_{\sin\phi\, y} & \sigma_{\sin\phi\, z} & \sigma^2_{\sin\phi} & \sigma_{\sin\phi\tan\lambda} & \sigma_{\sin\phi.1/p_T} \\ \sigma_{\tan\lambda\, y} & \sigma_{\tan\lambda\, z} & \sigma_{\tan\lambda\sin\phi} & \sigma^2_{\tan\lambda} & \sigma_{\tan\lambda.1/p_T} \\ \sigma_{1/p_T.y} & \sigma_{1/p_T.z} & \sigma_{1/p_T.\sin\phi} & \sigma_{1/p_T.\tan\lambda} & \boxed{\sigma^2_{1/p_T}} \end{pmatrix}$$



Parameter Covariance matrix = 5X5 matrix; 5(5+1)/2 = 15 independent entries

$$\left(\sigma^2_y, \sigma_{yz}, \sigma_{ysin\phi}, \sigma_{yTan\lambda}, \sigma_{y.1/p_T}, \sigma^2_z, \sigma_{zsin\phi}, \sigma_{ztan\lambda}, \sigma_{z.1/p_T}, \sigma^2_{\sin\phi}, \sigma_{\sin\phi\tan\lambda}\, \sigma_{\sin\phi.1/p_T}, \sigma^2_{\tan\lambda}, \sigma_{\tan\lambda.1/p_T}, \sigma^2_{1/p_T}\right)$$

Definition of Chi2

$$\chi^2 = (Y_{meas} - Y_{fit})^T W^{-1} (Y_{meas} - Y_{fit})$$

In our case

```
Double_t
AliExternalTrackParam::GetPredictedChi2(const Double_t p[2],const Double_t cov[3]) const {
  //-----------------------------------------------------------------
  // Estimate the chi2 of the space point "p" with the cov. matrix "cov"
  //-----------------------------------------------------------------
  Double_t sdd = fC[0] + cov[0];
  Double_t sdz = fC[1] + cov[1];
  Double_t szz = fC[2] + cov[2];
  Double_t det = sdd*szz - sdz*sdz;

  if (TMath::Abs(det) < kAlmost0) return kVeryBig;

  Double_t d = fP[0] - p[0];
  Double_t z = fP[1] - p[1];

  return (d*szz*d - 2*d*sdz*z + z*sdd*z)/det;
}
```

Chi2 with 5 parameters

```
TMatrixDSym c(5);
  c(0,0)=GetSigmaY2();
  c(1,0)=GetSigmaZY();   c(1,1)=GetSigmaZ2();
  c(2,0)=GetSigmaSnpY(); c(2,1)=GetSigmaSnpZ(); c(2,2)=GetSigmaSnp2();
  c(3,0)=GetSigmaTglY(); c(3,1)=GetSigmaTglZ(); c(3,2)=GetSigmaTglSnp(); c(3,3)=GetSigmaTgl2();
  c(4,0)=GetSigma1PtY(); c(4,1)=GetSigma1PtZ(); c(4,2)=GetSigma1PtSnp(); c(4,3)=GetSigma1PtTgl(); c(4,4)=GetSigma1Pt2();

  c(0,0)+=t->GetSigmaY2();
  c(1,0)+=t->GetSigmaZY();  c(1,1)+=t->GetSigmaZ2();
  c(2,0)+=t->GetSigmaSnpY();c(2,1)+=t->GetSigmaSnpZ();c(2,2)+=t->GetSigmaSnp2();
  c(3,0)+=t->GetSigmaTglY();c(3,1)+=t->GetSigmaTglZ();c(3,2)+=t->GetSigmaTglSnp();c(3,3)+=t->GetSigmaTgl2();
  c(4,0)+=t->GetSigma1PtY();c(4,1)+=t->GetSigma1PtZ();c(4,2)+=t->GetSigma1PtSnp();c(4,3)+=t->GetSigma1PtTgl();c(4,4)+=t->GetSigma1Pt2();
  c(0,1)=c(1,0);
  c(0,2)=c(2,0); c(1,2)=c(2,1);
  c(0,3)=c(3,0); c(1,3)=c(3,1); c(2,3)=c(3,2);
  c(0,4)=c(4,0); c(1,4)=c(4,1); c(2,4)=c(4,2); c(3,4)=c(4,3);

c.Invert();
if (!c.IsValid()) return kVeryBig;


Double_t res[5] = {
  GetY()   - t->GetY(),
  GetZ()   - t->GetZ(),
  GetSnp() - t->GetSnp(),
  GetTgl() - t->GetTgl(),
  GetSigned1Pt() - t->GetSigned1Pt()
};

Double_t chi2=0.;
for (Int_t i = 0; i < 5; i++)
  for (Int_t j = 0; j < 5; j++) chi2 += res[i]*res[j]*c(i,j);

return chi2;
```

Kalman Filter

$$\chi^2 = \frac{(x_p - \mu)^2}{\sigma_p^2} + \frac{(x_m - \mu)^2}{\sigma_m^2}$$

$$\frac{\partial \chi^2}{\partial \mu} = 0$$ Chi2 minimization

$$\mu = \frac{\frac{x_p}{\sigma_p^2} + \frac{x_m}{\sigma_m^2}}{\frac{1}{\sigma_p^2} + \frac{1}{\sigma_m^2}} = x_p \frac{\sigma_m^2}{\sigma_p^2 + \sigma_m^2} + x_m \frac{\sigma_p^2}{\sigma_p^2 + \sigma_m^2}$$

$$\sigma(\mu)^2 = \frac{1}{\frac{1}{\sigma_p^2} + \frac{1}{\sigma_m^2}} = \frac{\sigma_p^2 \sigma_m^2}{\sigma_p^2 + \sigma_m^2}$$

If ($\sigma_m >> \sigma_p$) $\mu \approx x_p$

If ($\sigma_p >> \sigma_m$) $\mu \approx x_m$

$$\mu = x_m + \frac{\sigma_m^2}{\sigma_p^2 + \sigma_m^2} (x_p - x_m)$$

$$\mu = x_m + K (x_p - x_m)$$

$$\sigma(\mu)^2 = \frac{\sigma_p^2 \sigma_m^2 + \sigma_m^4 - \sigma_m^4}{\sigma_p^2 + \sigma_m^2} = \sigma_m^2 - \frac{\sigma_m^4}{\sigma_p^2 + \sigma_m^2} = \sigma_m^2 (1 - K)$$

In ALICE

$$\mu_y = \frac{y_p \sigma_m^2 + y_m \sigma_p^2}{\sigma_p^2 + \sigma_m^2}$$

$$\mu_z = \frac{z_p \sigma_m^2 + z_m \sigma_p^2}{\sigma_p^2 + \sigma_m^2}$$

y_p, z_p from the track model

y_m, z_m from the measurement

K- Kalman gain factor

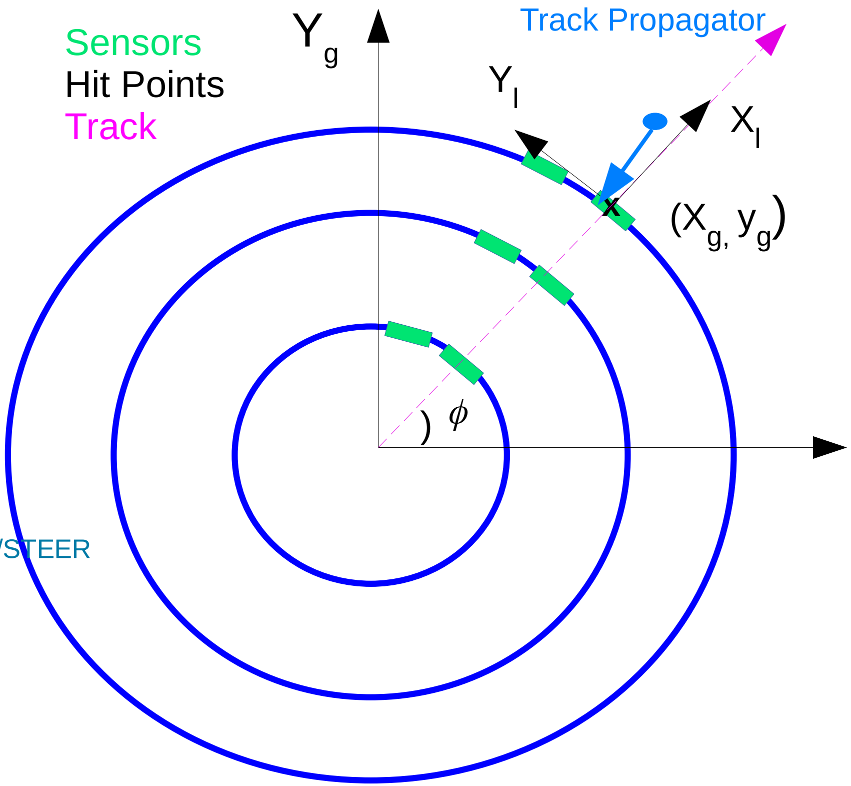
Measurement is corrected by K Factor

Two things required: Extrapolation with MS and Measured Points

Common Steps:

- Track Model initialize with the last point x+Margin (0.1)
- Convert last point to local coordinate system rotated by phi.
- Extrapolate track to the last layer
- Rotate Track to local coordinate system
- Update extraolation and measurement (weight average of position and errors
- Multiple scattering correction

Repeat above steps for each layer up to layer 1 and then extrapolate to the vertex



https://github.com/alisw/AliRoot/blob/master/STEER/STEERBase/AliExternalTrackParam.h

Track Extrapolation

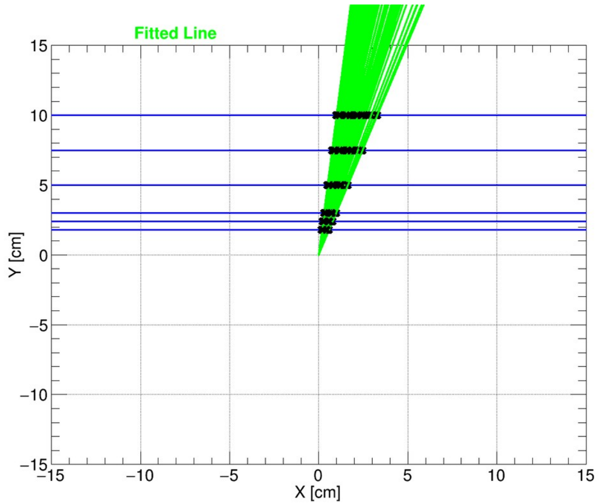
Bool_t PropagateTo(Double_t x, Double_t b);

Bool_t CorrectForMeanMaterialdEdx(Double_t **xOverX0Si**, Double_t 0,
Double_t mPion, Bool_t anglecorr=kTRUE);

At the vertex x_l = 0 and y_l is DCA_{xy}

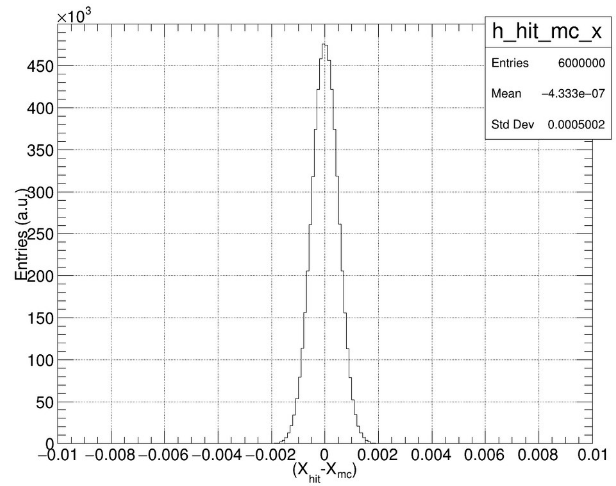
Fast Simulation (Expected DCAxy)

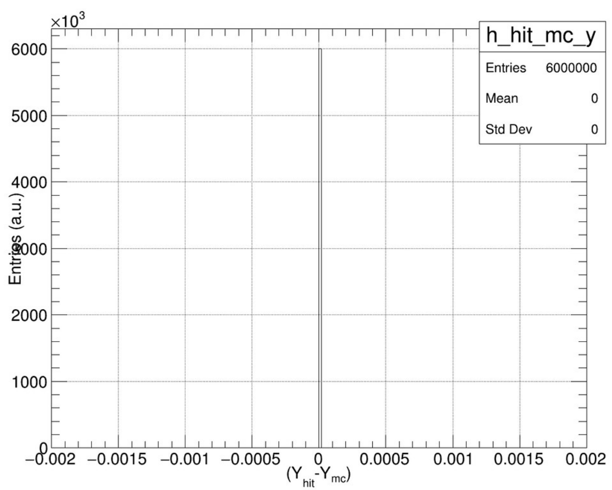
Chi2 Minimization (Ignoring Multiple Scattering)

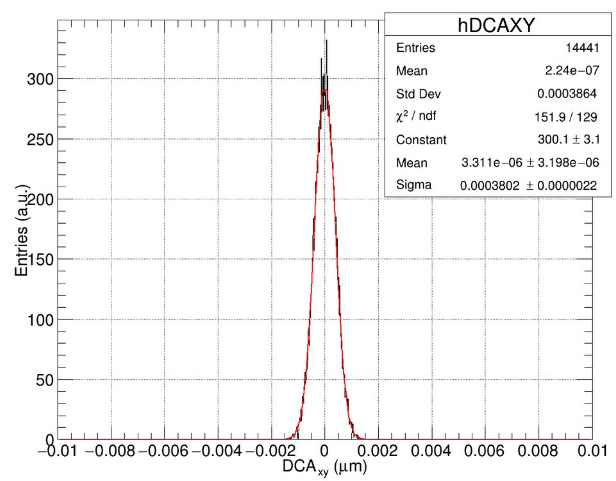


$$y = a_0 + a_1\, x$$

$$d = \frac{a_0}{\sqrt{1+a_1^2}}$$







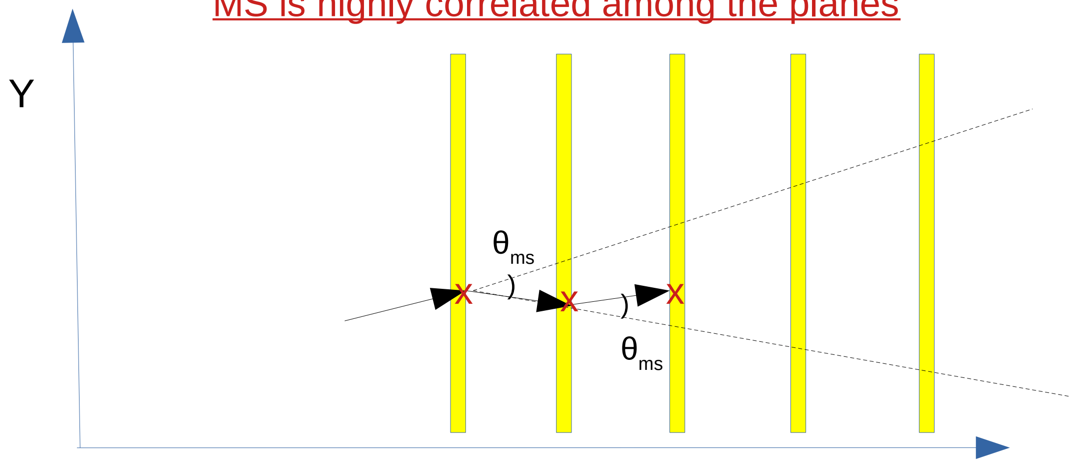
Simulation: Red Points we get during detector simulation with energy loss and Multiple scattering

Digitization: Smear these red point by pixel resolution (spatial resolution)

Reconstruction: Fit the point after digitization

In the Beam test data, the point are already include above effects so we can directly fit the points

MS is highly correlated among the planes



$$C_{ij} = \langle (y_{hit} - y_{true})_i * (y_{hit} - y_{true})_j \rangle$$

arXiv:1805.12014 [physics.ins-det]

Matrix by Werner Reigler (equal spacing)

$$\mathbf{C}_y = \mathbf{M} = \frac{\sigma_\alpha^2 L^2}{N^2} \begin{pmatrix} \frac{N^2\sigma_0^2}{\sigma_\alpha^2 L^2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & \cdots \\ 0 & 2 & 5 & 8 & 11 & 14 & 17 & 20 & \cdots \\ 0 & 3 & 8 & 14 & 20 & 26 & 32 & 38 & \cdots \\ 0 & 4 & 11 & 20 & 30 & 40 & 50 & 60 & \cdots \\ 0 & 5 & 14 & 26 & 40 & 55 & 70 & 85 & \cdots \\ 0 & 6 & 17 & 32 & 50 & 70 & 91 & 112 & \cdots \\ 0 & 7 & 20 & 38 & 60 & 85 & 112 & 140 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

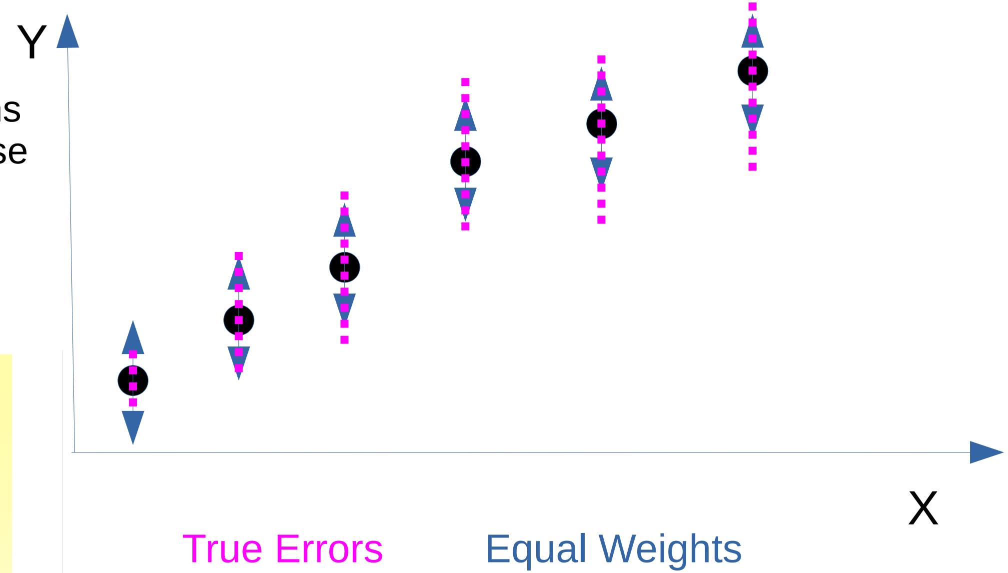
Matrix by me statistically (equal spacing): Multiple scattering

```
6x6 matrix is as follows

     |      0    |      1    |      2    |      3    |      4    |
----------------------------------------------------------------------
   0 |          0          0          0          0          0
   1 |          0          1      2.001          3      3.999
   2 |          0      2.001      5.003      8.001         11
   3 |          0          3      8.001         14      19.99
   4 |          0      3.999         11      19.99      29.98
   5 |          0      4.997      13.99      25.98      39.97

     |      5    |
----------------------------------------------------------------------
   0 |          0
   1 |      4.997
   2 |      13.99
   3 |      25.98
   4 |      39.97
   5 |      54.95
```

Global Chi2 fitting with MS (Line Fit)

The previous case if we use only spatial resolution means we are giving the equal weights to each points in this case

In reality points have different errors: Chi2 is overestimated



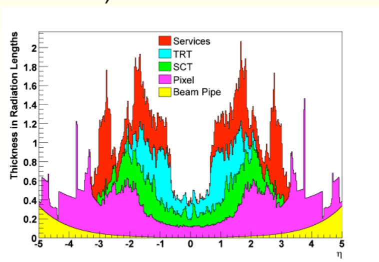
P 10

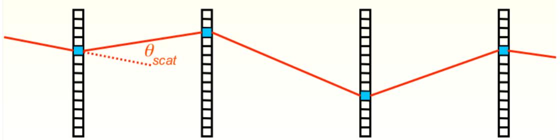
Scattering Angle Formulation

- The scattering angles and the energy losses become additional fit parameters, along with the track parameters at the vertex (i.e. impact parameter, direction and momentum).
- The χ^2 function to be minimized w.r.t. the fit parameters is:

$$\chi^2 = \sum_{hits}\frac{\Delta y^2}{\sigma_{hit}^2} + \sum_{scatters}\frac{\theta_{scat}^2}{\sigma_{scat}^2} + \sum_{Eloss}\frac{(\Delta E - \overline{\Delta E})^2}{\sigma_{Eloss}^2}$$

where Δy are the residuals, and σ_{scat} and $\overline{\Delta E}$ can be estimated from the material properties (done by the MultipleScatteringUpdator and EnergyLossUpdator tools)





CHEP07 conference 5 September 2007, T. Cornelissen

Minimize Full covariance matrix for proper treatment of multiple scattering in the fit

Straight Line Fit

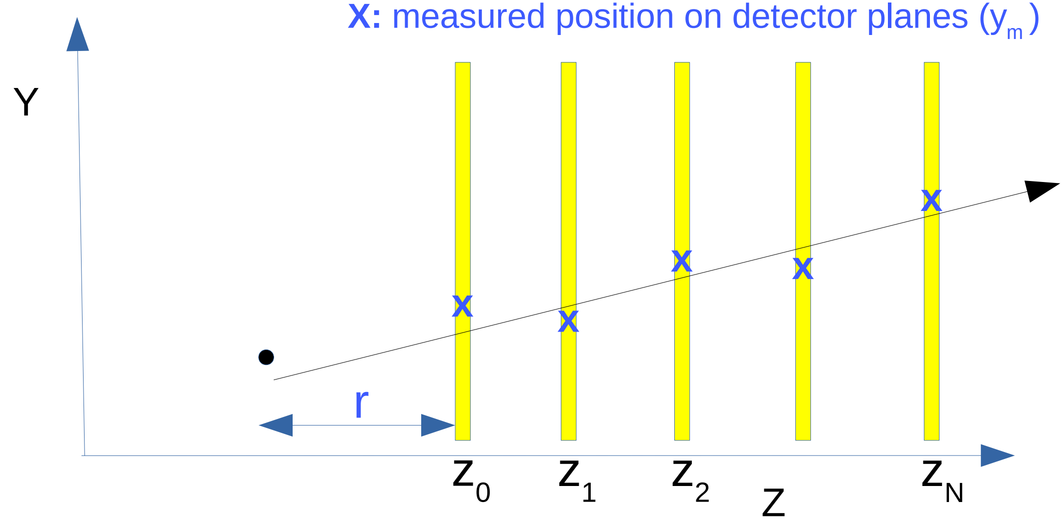
General line

$$y = a + bz$$

$$\chi^2 = \sum_{i=0}^{N} \frac{(y_m - y_i)^2}{\sigma_i^2} = \sum_{i=0}^{N} \frac{(y_m - a - bz_i)^2}{\sigma_i^2}$$

$$\partial \chi^2 / \delta a = 0 \qquad \partial \chi^2 / \delta b = 0$$

X: measured position on detector planes (y_m)



Chi2 minimization: returns best a ,b and σ_a , σ_b, σ_{ab} , σ_{ba}

$$S_1 = \sum_{i=0}^{N} \frac{1}{\sigma_i^2} \qquad S_y = \sum_{i=0}^{N} \frac{y_i}{\sigma_i^2} \qquad S_z = \sum_{i=0}^{N} \frac{z_i}{\sigma_i^2} \qquad S_{yz} = \sum_{i=0}^{N} \frac{y_i z_i}{\sigma_i^2} \qquad S_{zz} = \sum_{i=0}^{N} \frac{z_i^2}{\sigma_i^2}$$

$$\Delta = S_1 S_{zz} - S_z^2$$

$$a = (S_y S_{zz} - S_z S_{zy}) / \Delta$$

$$b = (S_1 S_{zy} - S_z S_y) / \Delta$$

$$\sigma_a^2 = S_{zz} / \Delta$$

$$\sigma_b^2 = S_1 / \Delta$$

$$\sigma_{ab} = \sigma_{ba} = -S_z / \Delta$$

Rewrite $\Rightarrow$

Covariance Matrix of Parameters

$$\begin{pmatrix} \sigma_a^2 & \sigma_{ab} \\ \sigma_{ba} & \sigma_b^2 \end{pmatrix} = \frac{1}{\Delta} \begin{pmatrix} S_{zz} & -S_z \\ -S_z & S_1 \end{pmatrix}$$

http://www.le.infn.it/lhcschool/talks/Ragusa.pdf
http://www.foo.be/docs-free/Numerical_Recipe_In_C/c15-2.pdf

Covariance matrix entries affected by multiple scattering

$$
\begin{array}{c|c}
 & \begin{array}{ccccc} 1/p & \lambda & \phi & y_{\perp} & z_{\perp} \end{array} \\
\begin{array}{c} 1/p \\ \lambda \\ \phi \\ y_{\perp} \\ z_{\perp} \end{array} &
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & <\theta_p^2> & 0 & 0 & -\dfrac{<\theta_p^2> \ \mathrm{d}l}{2} \\
0 & 0 & \dfrac{<\theta_p^2>}{\cos^2\lambda} & \dfrac{<\theta_p^2> \ \mathrm{d}l}{(2\cos\lambda)} & 0 \\
0 & 0 & \dfrac{<\theta_p^2> \ \mathrm{d}l}{(2\cos\lambda)} & \dfrac{<\theta_p^2> \ (\mathrm{d}l)^2}{3} & \\
0 & -\dfrac{<\theta_p^2> \ \mathrm{d}l}{2} & 0 & 0 & \dfrac{<\theta_p^2> \ (\mathrm{d}l)^2}{3}
\end{pmatrix}
\end{array}
$$

https://agenda.infn.it/event/1096/contributions/6159/attachments/4504/4980/Rotondi_3.pdf