LAR TO GAR AND TRACK RECONSTRUCTION:
FUTURE PLANS AND CURRENT STATUS

UNIVERSITY OF OXFORD

Presents:
Federico Battisti

LAR TO GAR SAMPLE: MOTIVATION AND CURRENT PROCEDURE

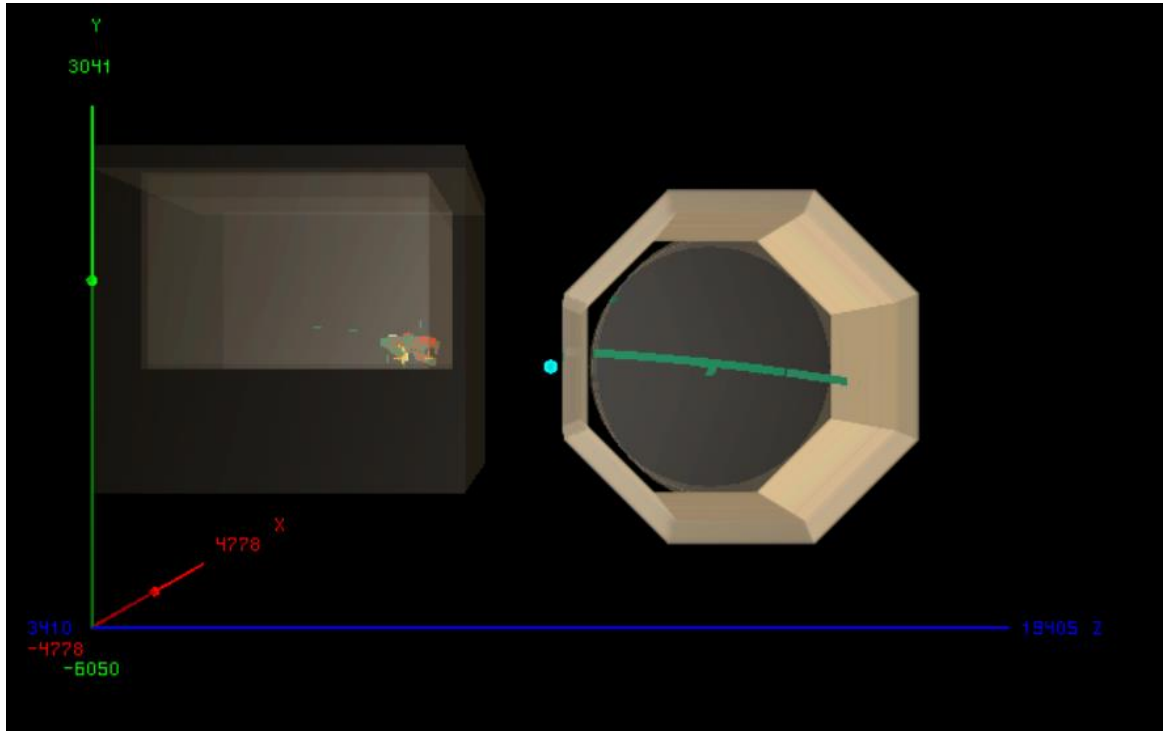
- An important role of ND-GAr will be to function as a muon spectrometer of ND-LAr: to evaluate its capabilities in that sense the LAr → GAr propagation of tracks needs to be very well studied

- The current ND LAr → GAr simulation chain:
  1. Simulate neutrino interactions with GENIE in a ND hall geometry file containing only the liquid Argon detector
  2. Propagate particles using edep-sim in a ND hall geometry file containing both ArgonCube and HPgTPC
  3. Convert edep-sim file to root file readable by GarSoft
  4. Follow the Garsoft reconstruction chain

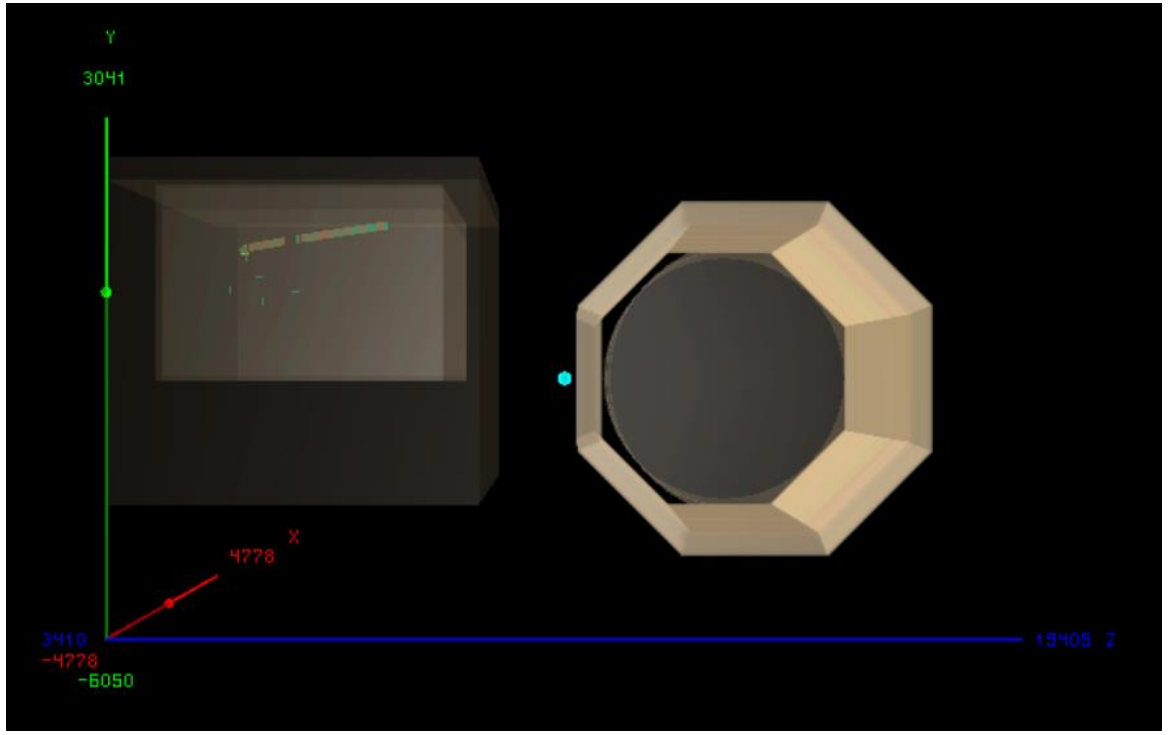
- Recently a wiki has been developed by Eldwan Brianne : https://cdcvs.fnal.gov/redmine/projects/dune-neardet-design/wiki/Run_edep-sim_samples_through_GArSoft

LAR TO GAR SAMPLE: AN EXAMPLE

- Two examples of $\nu_\mu(CC)$ interactions in ArgonCube one with a passing muon reaching NDGAr, the other without (both made with edep-disp)



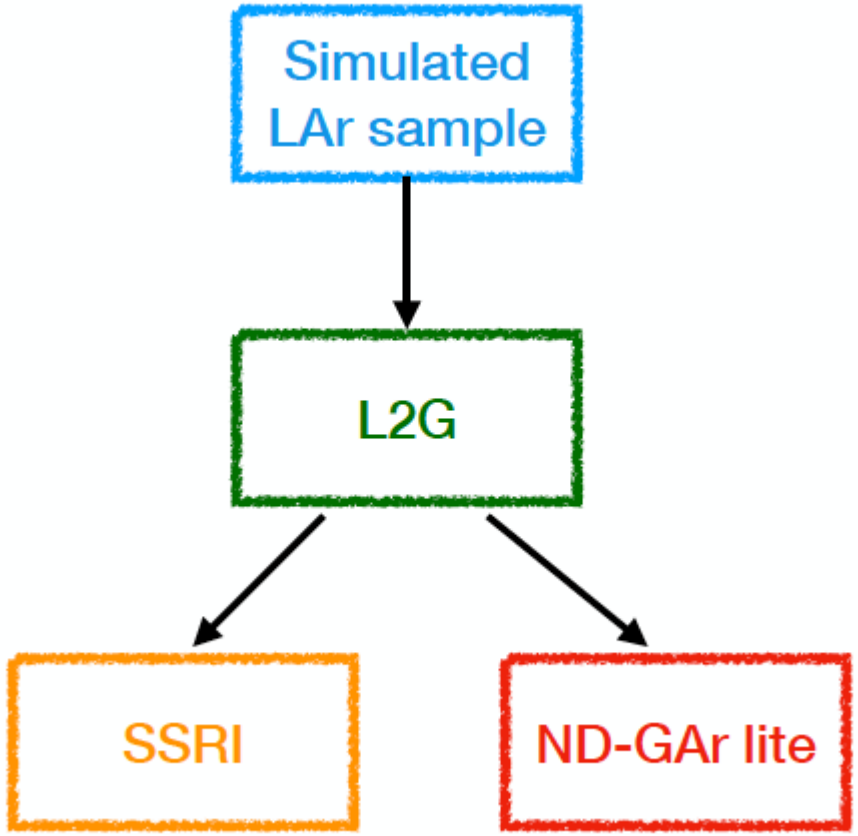
PASSING MUON



NON-PASSING MUON

LAR TO GAR SAMPLE: IMPROVEMENTS

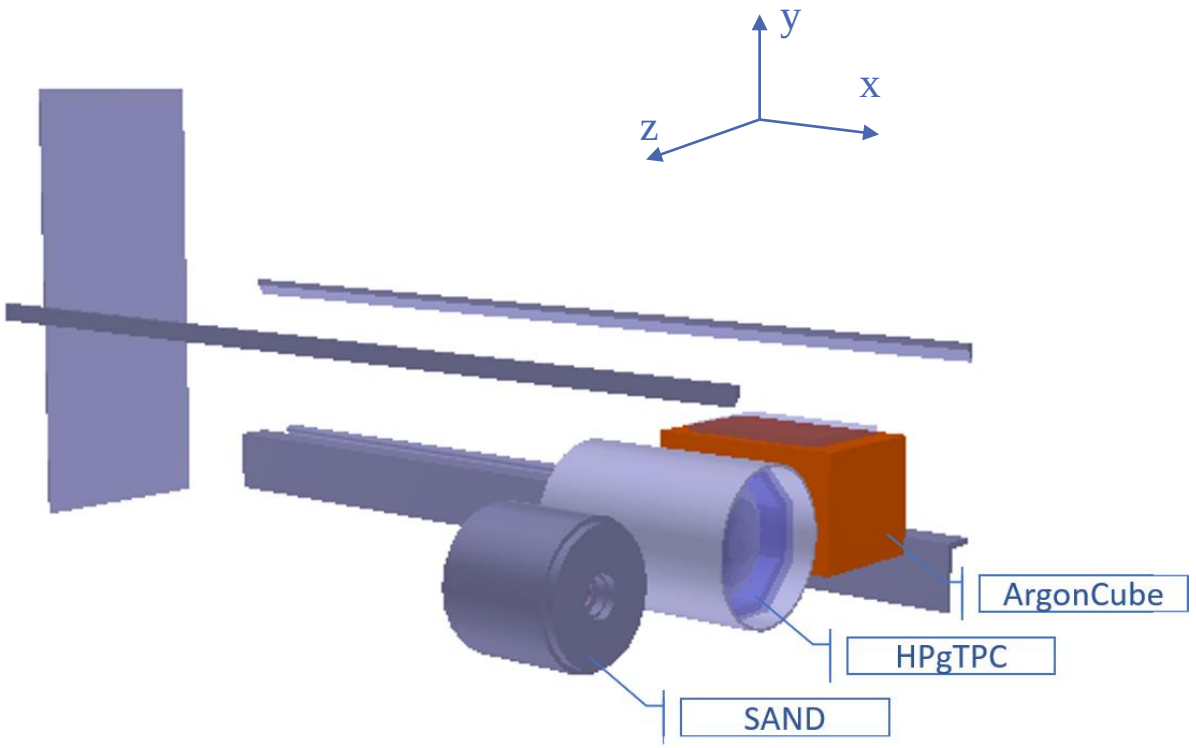
- Strong need for large LAr samples already propagated in edep-sim to test the chain
- L2G: interface that takes outgoing LAr particles and feeds them to edep-sim with any TMS detector could speed up the sample production (Currently starting to work on it with Eldwan towards TMS meeting)
- Constant B-field currently being used (Custom B-field is already implemented in edep-sim/GArSoft)
- Need to improve the track reconstruction and fitting



https://indico.fnal.gov/event/44562/contributions/200915/attachments/136745/170170/DUNE_ND_Meeting_28.10.20.pdf

TRACK FINDING AND FITTING IN GAR

- Track finding:
  1. Find hits (i.e. pulses found on a single channel)
  2. Group nearby hits in space and time and find their charge centroids found: TPC Clusters
  3. Find track segments (vector hits) and associate them with TPC Clusters
  4. Cluster vector hits together into track candidates



- Track fitting: Kalman filter is applied to the track candidates two times in order to compute the best estimates of the track parameters on both ends of the track (Developed by Thomas Junk and Leo Bellantoni: https://docs.dunescience.org/cgi-bin/private/ShowDocument?docid=13933)

KALMAN FILTER

- A Kalman filter is an iterative algorithm which uses a system's physical laws of motion, known control inputs and multiple sequential measurements to form an estimate of the system's varying quantities
- At each step of the iteration an estimate of the state of the system is produced as a weighted average of the system's predicted state and of the new measurement. The weights are calculated from the covariance.
- The extended Kalman filter expands the Kalman filter technique to non-linear systems
- The models for state transition and measurement can be written as:

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1}$$

$$z_k = h(x_k) + v_k$$

- Where f is the function of the previous state, x_{k-1} , and the control input, u_{k-1}, that provides the current state x_k. h is the measurement function that relates the current state, x_k, to the measurement z_k. $w_{k-1}$and v_k are Gaussian noises for the process model and the measurement model with covariance Q and R, respectively

EXTENDED KALMAN FILTER ALGORITHM

1. Make a priori predictions for the current step's state and covariance matrix using the a posteriori best estimate of the previous step (i.e. updated using measurement)

STATE VECTOR	$\hat{x}_k^- = f(\hat{x}_{k-1}^+, u_{k-1})$
COVARIANCE MATRIX	$P_k^- = F_{k-1} P_{k-1}^+ F_{k-1}^T + Q$

$$F_{k-1} = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1}^+, u_{k-1}}$$

JACOBIAN

$$H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_k^-}$$

CONVERSION MATRIX

$$Q$$

PROCESS NOISE COVARIANCE

Note: In the first iteration step we use step 0 estimates for the state vector and the covariance matrix (x_0, P_0), which can be made very roughly

EXTENDED KALMAN FILTER ALGORITHM

2. Calculate the measurement residual and the Kalman Gain

RESIDUAL $\tilde{y}_k = z_k - h(\hat{x}_k^-)$

KALMAN GAIN $K_k = P_k^- H_k^T \left(R + H_k P_k^- H_k^T\right)^{-1}$

R

MEASUREMENT NOISE COVARIANCE

3. Update the estimate

STATE VECTOR $\hat{x}_k^+ = \hat{x}_k^- + K_k \tilde{y}$

COVARIANCE MATRIX $P_k^+ = (1 - K_k H_k) P_k^-$

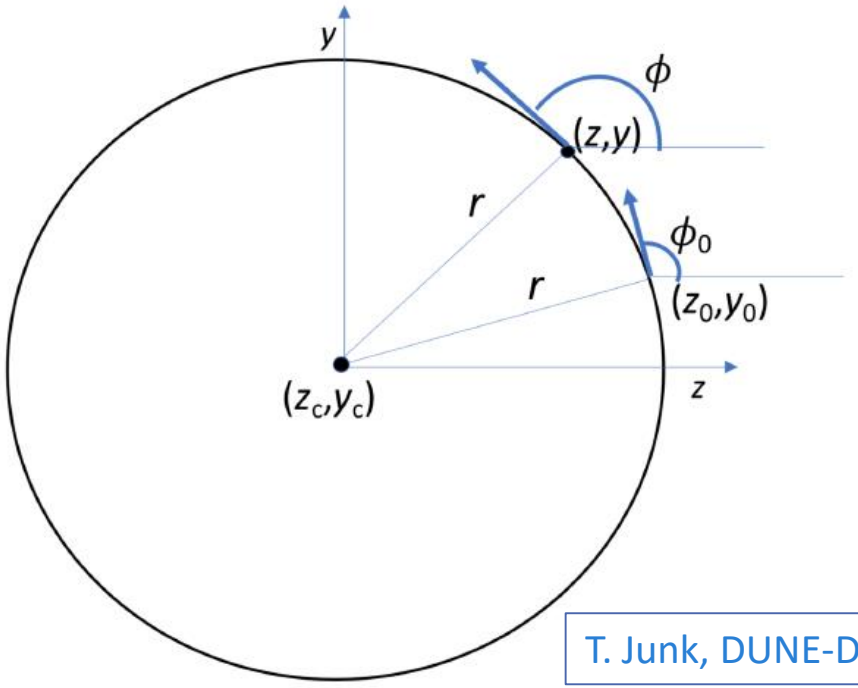
KALMAN FILTER APPLICATION

- We want to apply the extended Kalman filter to the motion of a charged particle in the magnetic field

$$\begin{cases} x = x_0 + r \tan\lambda\,(\phi - \phi_0) \\ y = y_C - r\cos\phi \\ z = z_C + r\sin\phi \end{cases}$$

TRACK PARAMETERS

We apply the Kalman filter to track candidates, consisting of groups of TPC clusters, which are identified and put together during the reconstruction process. Each step of the Kalman filter algorithm is identified by one of these TPC clusters



T. Junk, DUNE-Doc-13933

FIG. 1: Track parameter definitions in the (z, y) plane.

https://docs.dunescience.org/cgi-bin/private/ShowDocument?docid=13933

KALMAN FILTER APPLICATION: INITIAL ESTIMATES

- Before the Kalman filter algorithm can be applied, we need an initial estimate for the state vector, which in our case includes $y, z, 1/r, \phi$ and λ and the covariance matrix

STATE VECTOR

$$x_0^T = (y_0 \quad z_0 \quad 1/r_0 \quad \phi_0 \quad \lambda_0) = (0 \quad 0 \quad 0.1 \quad 0 \quad 0)$$

COVARIANCE MATRIX

$$P_0 = \begin{pmatrix} 1^2 & 0 & 0 & 0 & 0 \\ 0 & 1^2 & 0 & 0 & 0 \\ 0 & 0 & 0.5^2 & 0 & 0 \\ 0 & 0 & 0 & 0.5^2 & 0 \\ 0 & 0 & 0 & 0 & 0.5^2 \end{pmatrix}$$

- The estimated quantities from the state vector can be used to estimate the particle's momentum

$$\begin{cases} p_x = p_T \tan\lambda \\ p_y = p_T \sin\phi \\ p_z = p_T \cos\phi \end{cases}$$

$$p_T(\mathrm{GeV}/c) = 0.3 \times B(T) \times r(m)$$

KALMAN FILTER APPLICATION: PREDICTION AND MEASUREMENT

- From the equation of motion we obtain the prediction function for our state vector.

$$\hat{x}_k^- = f\begin{pmatrix} \hat{y}_{k-1}^+ \\ \hat{z}_{k-1}^+ \\ 1/\hat{r}_{k-1}^+ \\ \hat{\phi}_{k-1}^+ \\ \hat{\lambda}_{k-1}^+ \end{pmatrix} = \begin{pmatrix} \hat{y}_{k-1}^+ + \mathrm{dx_k} \times \cot\hat{\lambda}_{k-1} \times \sin\hat{\phi}_{k-1}^+ \\ \hat{z}_{k-1}^+ + \mathrm{dx_k} \times \cot\hat{\lambda}_{k-1} \times \cos\hat{\phi}_{k-1}^+ \\ 1/\hat{r}_{k-1}^+ \\ \hat{\phi}_{k-1}^+ + \mathrm{dx_k} \times \cot\hat{\lambda}_{k-1} \times \sin\hat{\phi}_{k-1}^+ \\ \hat{\lambda}_{k-1}^+ \end{pmatrix}$$

Note: the prediction model does not account for dE/dx energy loss

- The only measured quantities in our case are y and z, and are set at the center of the TPC cluster correspondent to the present step.

$$z_k = \begin{pmatrix} y_k^h \\ z_k^h \end{pmatrix}$$

KALMAN FILTER APPLICATION: STEP DETERMINATION

- Each algorithm step corresponds to a TPC cluster. The x coordinate is treated as independent and used to identify the step width dx. The step width is determined for each algorithm step, so that it minimizes:

$$\Delta = \left[\frac{\left(x_k^h - \hat{x}_k^-\right)}{\sigma_x}\right]^2 + \left[\frac{\left(y_k^h - \hat{y}_k^-\right)}{\sigma_y}\right]^2 + \left[\frac{\left(z_k^h - \hat{z}_k^-\right)}{\sigma_z}\right]^2$$

σ_x and $\sigma_y = \sigma_z$ are the errors associated with the TPC Clusters

- For each step we then have:

Note: despite being our free parameter, x is associated with an uncertainty; other options might be worth exploring

$$dx_k = \frac{\left(\frac{\cot\hat{\lambda}_{k-1}^+}{\sigma_{yz}^2}\left(\left(y_k^h - \hat{y}_{k-1}^+\right)\sin\hat{\phi}_{k-1}^+ + \left(z_k^h - \hat{z}_{k-1}^+\right)\cos\hat{\phi}_{k-1}^+\right)\right) + \frac{\left(x_k^h - \hat{x}_{k-1}^+\right)}{\sigma_x^2}}{\cot^2\hat{\lambda}_{k-1}^+ \Big/ \sigma_{yz}^2 + 1 \Big/ \sigma_x^2}$$

KALMAN FILTER APPLICATION: COVARIANCE MATRIX PREDICTION

- First step to make the prediction for the covariance matrix is to calculate the Jacobian

$$F_{k-1} = \frac{\partial f(\hat{x}_{k-1}^{+})}{\partial \hat{x}_{k-1}^{+}} = \begin{bmatrix} 1 & 0 & 0 & dx_k \cot\hat{\lambda}_{k-1}^{+} \cos\hat{\phi}_{k-1}^{+} & dx_k \sin\hat{\phi}_{k-1}^{+}(-1-\cot^2\hat{\lambda}_{k-1}^{+}) \\ 0 & 1 & 0 & -dx_k \cot\hat{\lambda}_{k-1}^{+} \sin\hat{\phi}_{k-1}^{+} & dx_k \cos\hat{\phi}_{k-1}^{+}(-1-\cot^2\hat{\lambda}_{k-1}^{+}) \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & dx_k \cot\hat{\lambda}_{k-1}^{+} & 1 & dx_k/\hat{r}_{k-1}^{+}(-1-\cot^2\hat{\lambda}_{k-1}^{+}) \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- The step uncertainty matrix is also needed

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{\Delta 1/r} & 0 & 0 \\ 0 & 0 & 0 & \sigma_{\Delta\phi} & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\Delta\lambda} \end{bmatrix}$$

- The prediction is then:

$$P_k^{-} = F_{k-1} P_{k-1} F_{k-1}^{T} + Q$$

KALMAN FILTER APPLICATION: EVALUATE THE RESIDUAL

- We now evaluate the residual and Kalman Gain

RESIDUAL

$$\tilde{y}_k = z_k - H(\hat{x}_k^-) = \begin{pmatrix} y_k^h - \hat{y}_k^- \\ z_k^h - \hat{z}_k^- \end{pmatrix}$$

KALMAN GAIN

$$K_k = P_k^- H (R + H P_k^- H^T)^{-1}$$

With:

$$R = \begin{pmatrix} \sigma_{yz}^2 & 0 \\ 0 & \sigma_{yz}^2 \end{pmatrix}$$

MEASUREMENT NOISE COVARIANCE

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

CONVERSION MATRIX

Note: The uncertainties in R are fixed, before the Kalman filter is applied, as external parameters: R is not updated.

KALMAN FILTER APPLICATION: PREDICTION UPDATE

- We are now finally able to update our estimates using both the a priori prediction and the measurement

STATE VECTOR	$\hat{x}_k^+ = \hat{x}_k^- + K_k \tilde{y}_k$
COVARIANCE MATRIX	$P_k^+ = (1 - H_k K_k) P_k^-$

KALMAN FILTER APPLICATION: χ^2

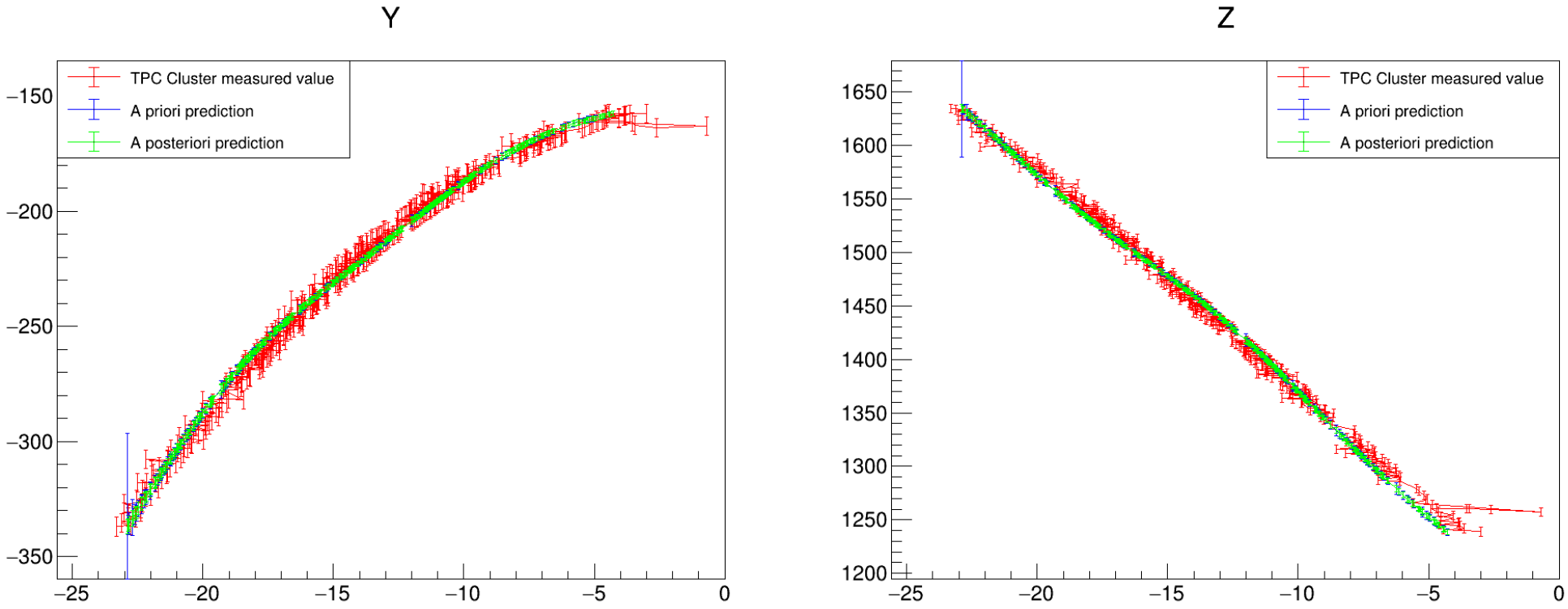
- The Kalman filter algorithm is applied to the track candidate both ways (i.e. From first TPC cluster to last and vice versa) and each time a χ^2 value is calculated

$$\chi^2 = \frac{\sum|\tilde{y}_k|^2}{\sum\left|y_k^{typ}\right|^2}$$

- Where y_k^{typ} is the residual typical value which depends on the particle trajectory position in the detector and the tracking planes (fixed values evaluated a priori)

KALMAN FILTER: PERFORMANCE EVALUATION

- Plots describing the evolution of the estimate for the measured quantities (y, z) as a function of the free parameter x (Information retrieved directly modifying the kalman filter module in garsoft tpctrackfit2_module.cc) for a muon from a particle gun pointing towards the center of the TPC. FORWARD FIT

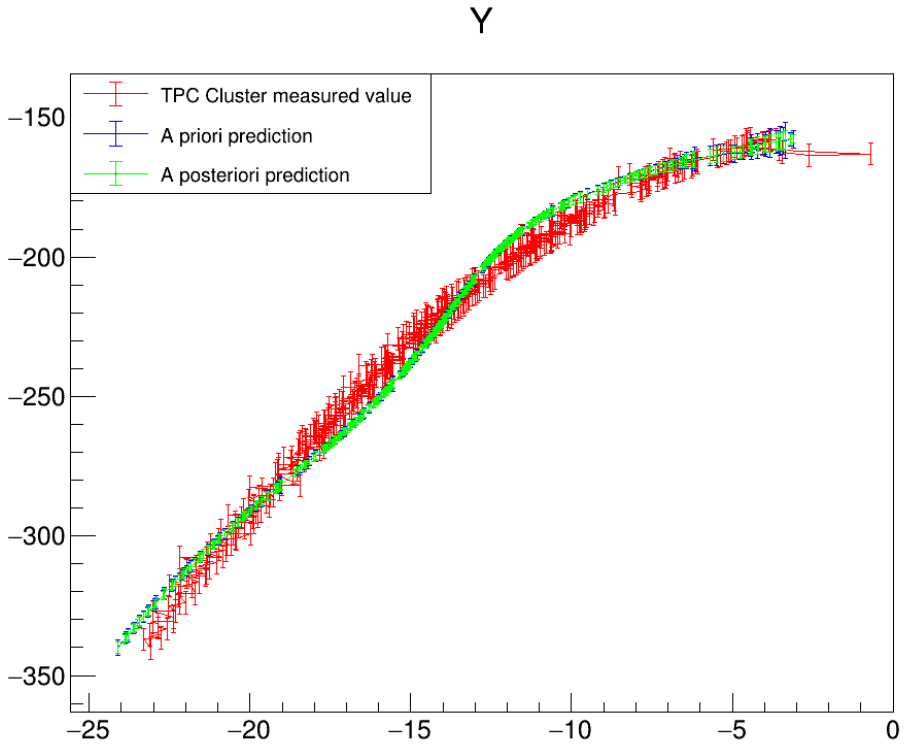


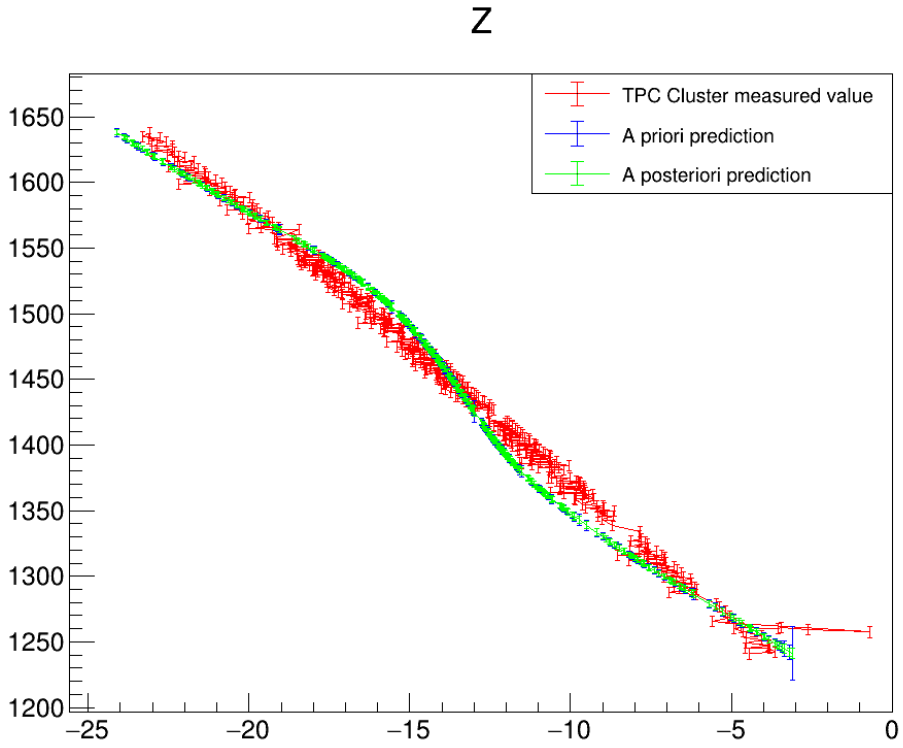
FITTER PROPAGATION DIRECTION

Note: the energy loss (dE/dx) of the particle along the track causes the particle to slow down and the track parameters (i.e the curvature) to change. The Kalman filter updates step by step its estimates using measurement and taking into account these changes: for this reason the Forward and Backward fits will produce different results, despite being performed with the same procedure

KALMAN FILTER: PERFORMANCE EVALUATION

- Plots describing the evolution of the estimate of the measured quantities (y, z) as a function of the free parameter x (Information retrieved directly modifying the kalman filter module in garsoft tpctrackfit2_module.cc) for a muon from a particle gun pointing towards the center of the TPC: BACKWARD FIT



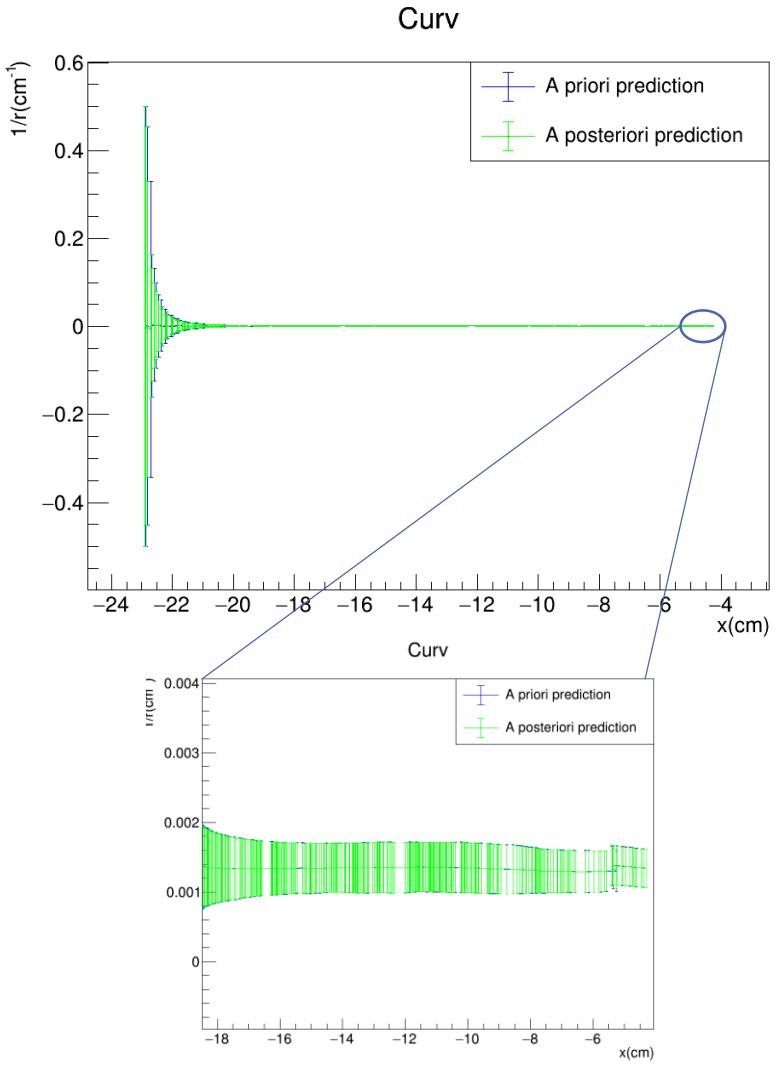


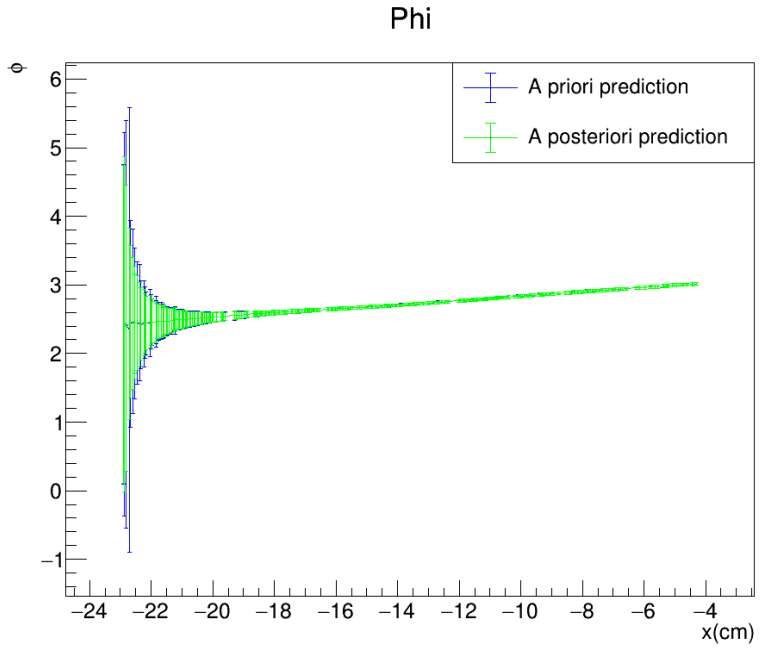
FITTER PROPAGATION DIRECTION

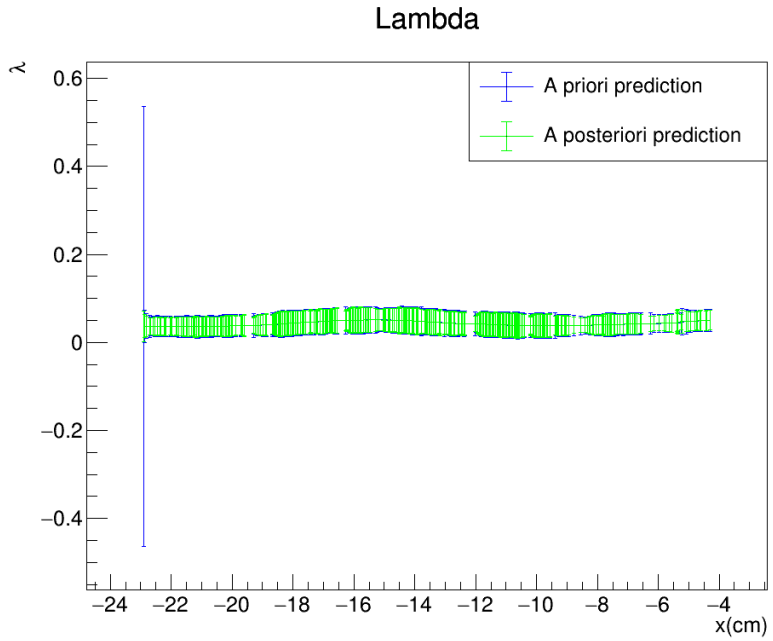
Note: at the moment the Kalman gain is calibrated in such a way to strongly favour the a priori prediction over the measurement; this might not be the ideal choice

KALMAN FILTER PERFORMANCE

- Plots describing the evolution of the estimate of the non measured quantities $(\frac{1}{r}, \phi, \lambda)$ as a function of the free parameter x:
  FORWARD FIT



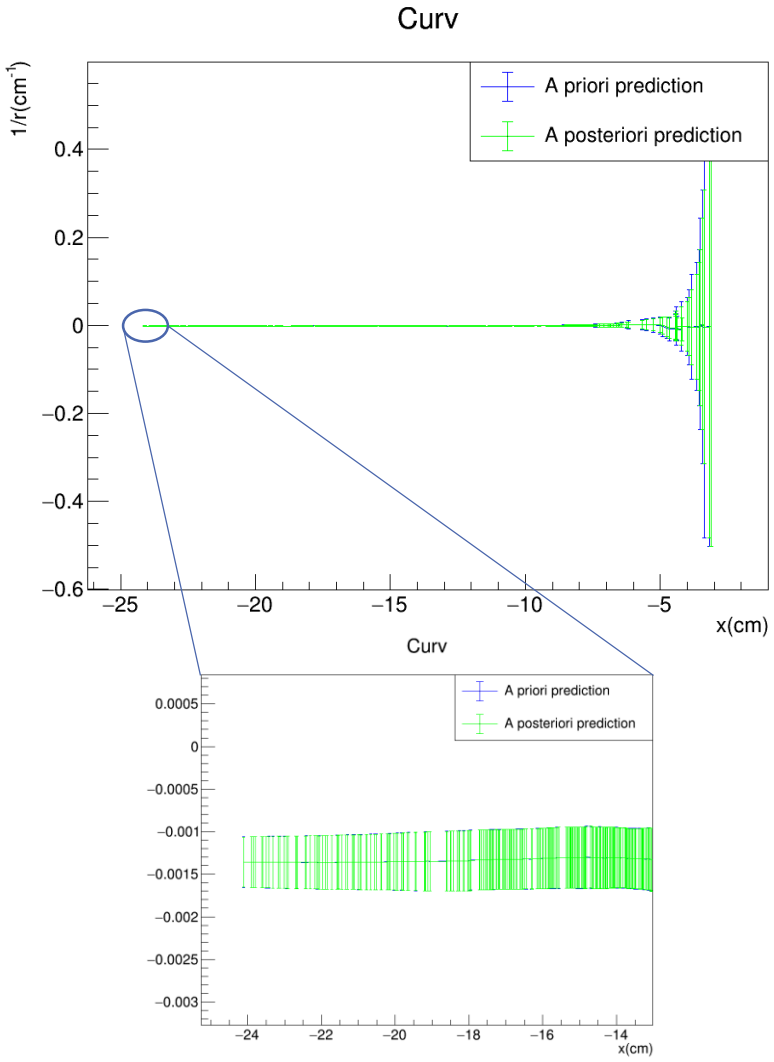


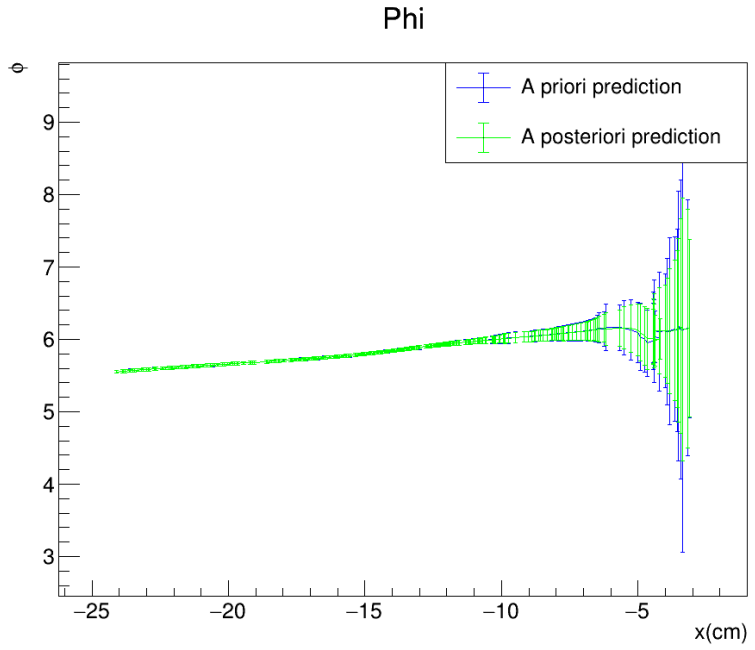


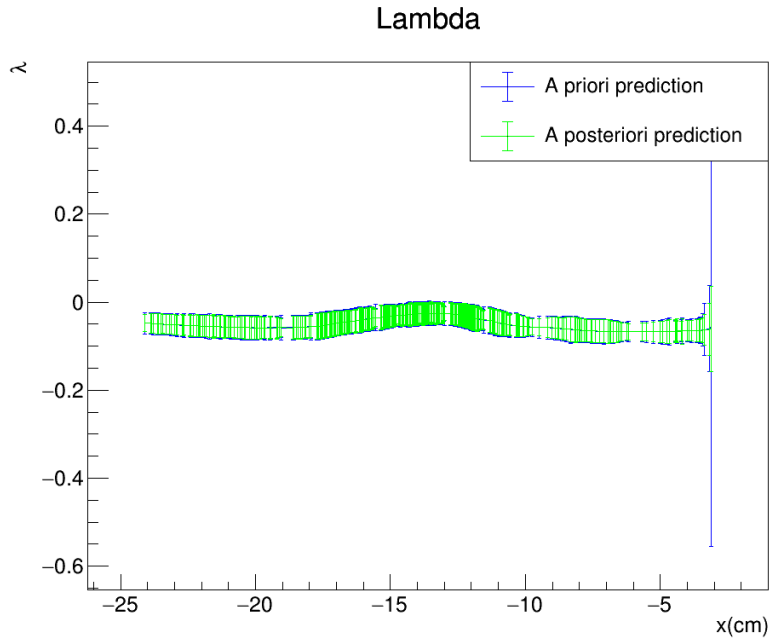
FITTER PROPAGATION DIRECTION

KALMAN FILTER PERFORMANCE

- Plots describing the evolution of the estimate of the non measured quantities $(\frac{1}{r}, \phi, \lambda)$ as a function of the free parameter x:
  BACKWARD FIT



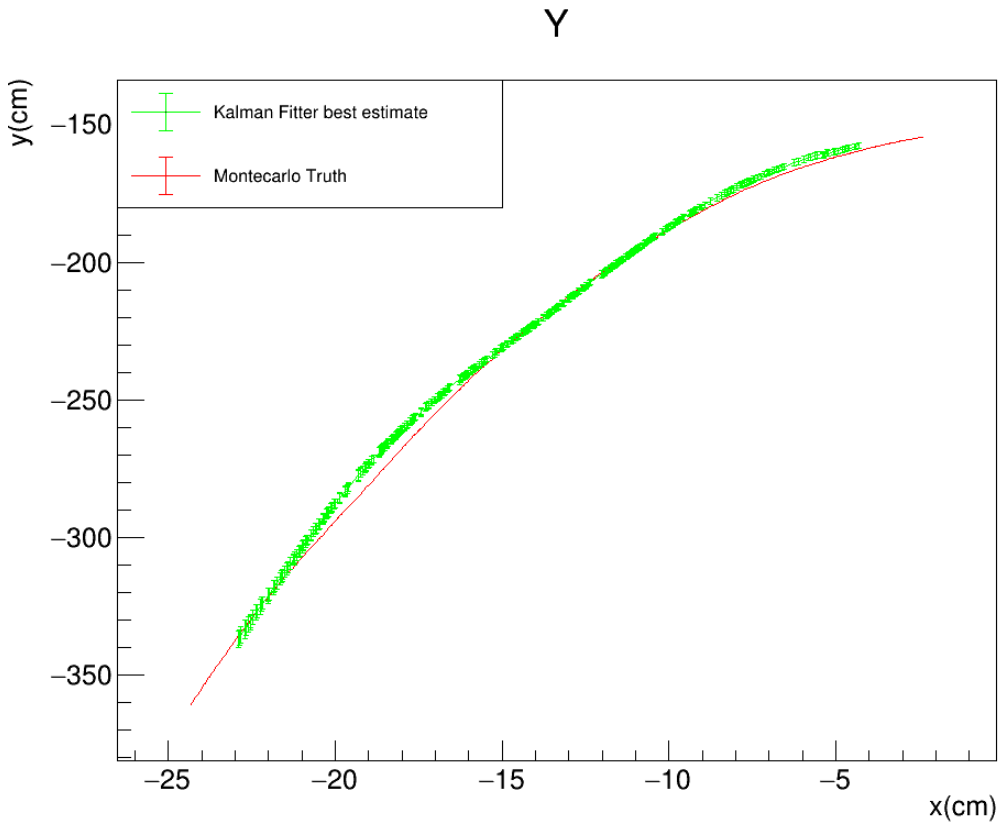


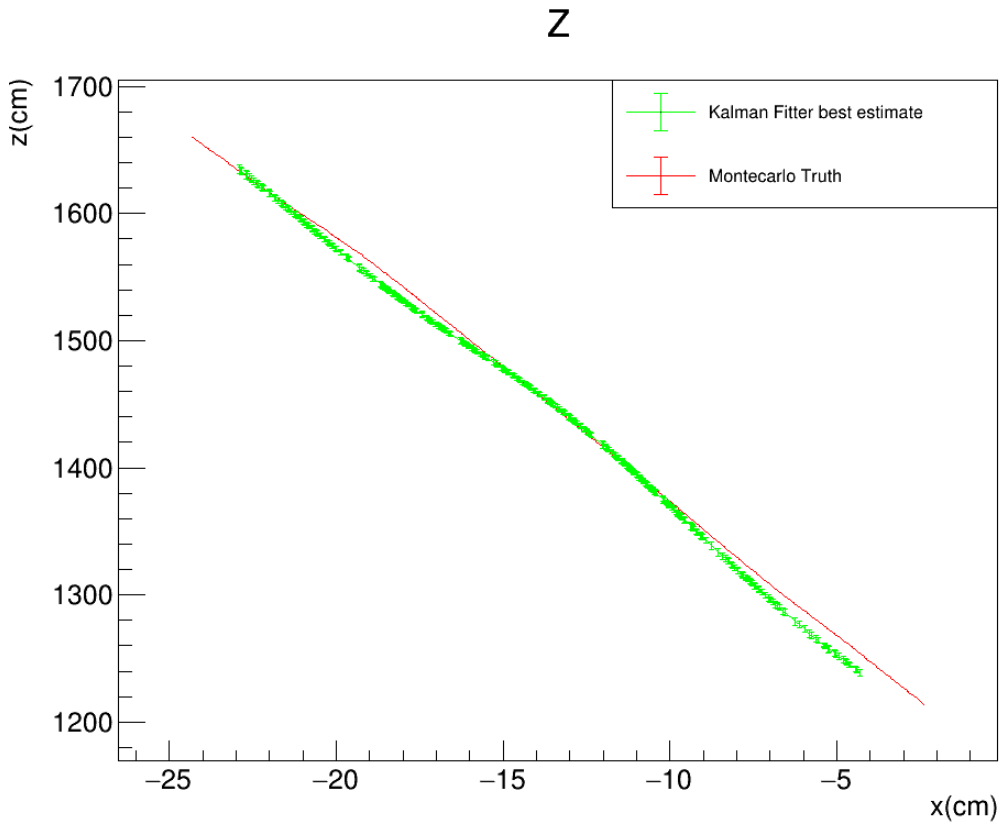


FITTER PROPAGATION DITECTION

KALMAN FILTER PERFORMANCE

- Plots comparing the best estimate from the Kalman fitter algorithm for the measured quantities (y, z) as a function of the free parameter x with the MC Truth values: FORWARD FIT

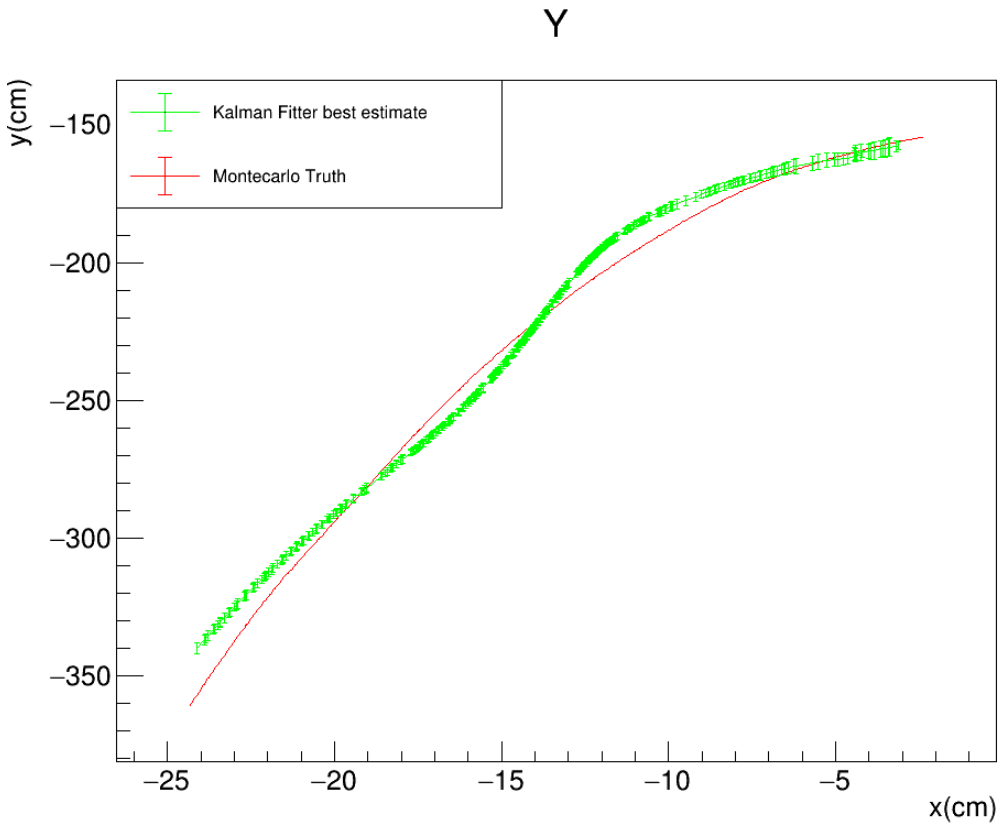


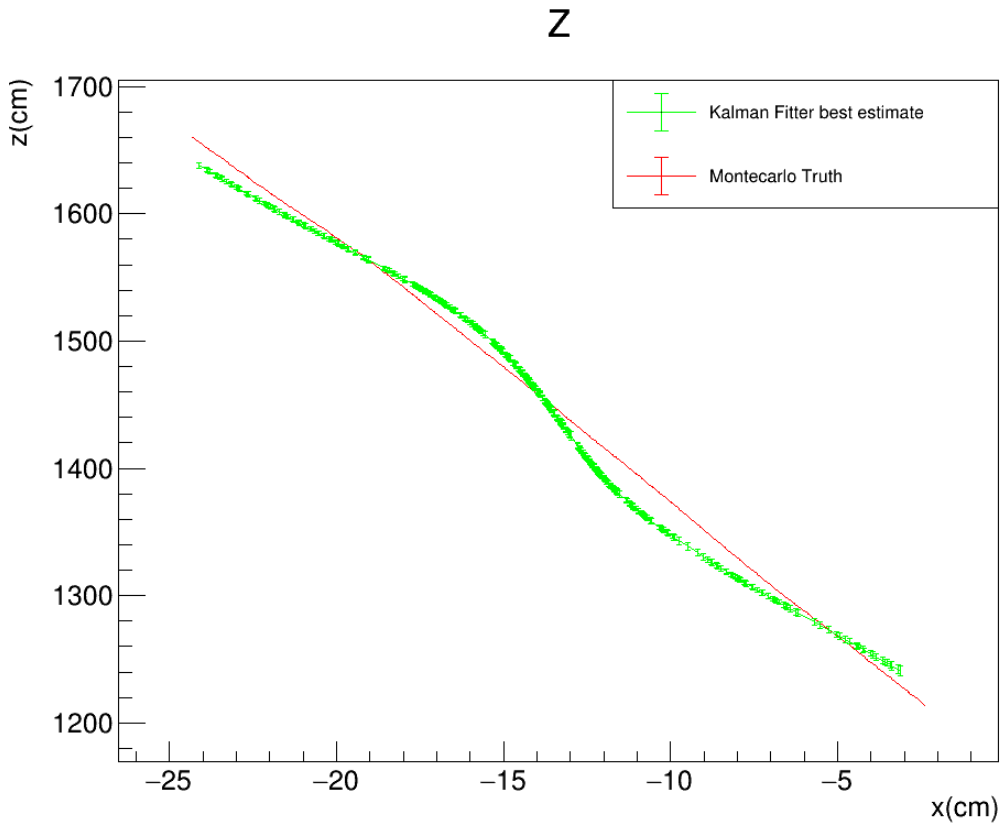


FITTER PROPAGATION DIRECTION

KALMAN FILTER PERFORMANCE

- Plots comparing the best estimate from the Kalman fitter algorithm for the measured quantities (y, z) as a function of the free parameter x with the MC Truth values: BACKWARD FIT

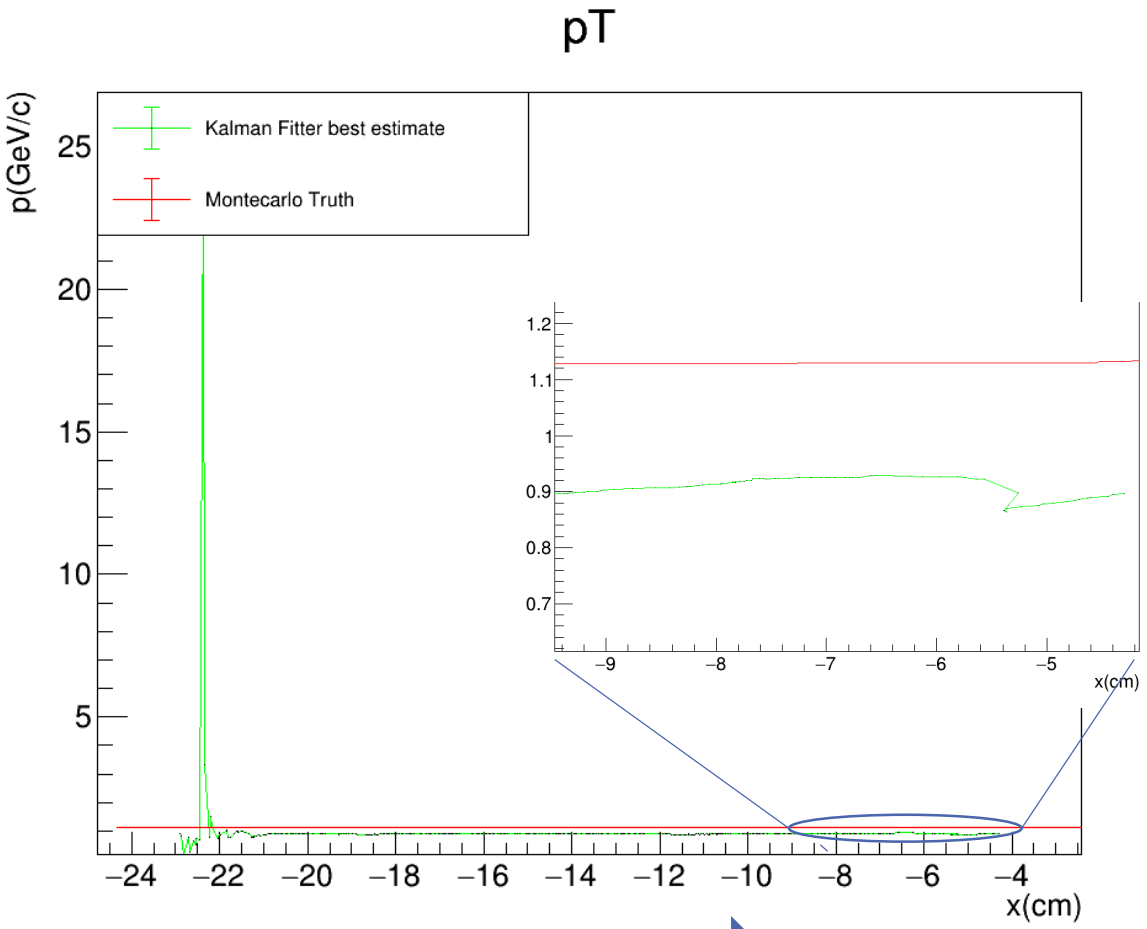




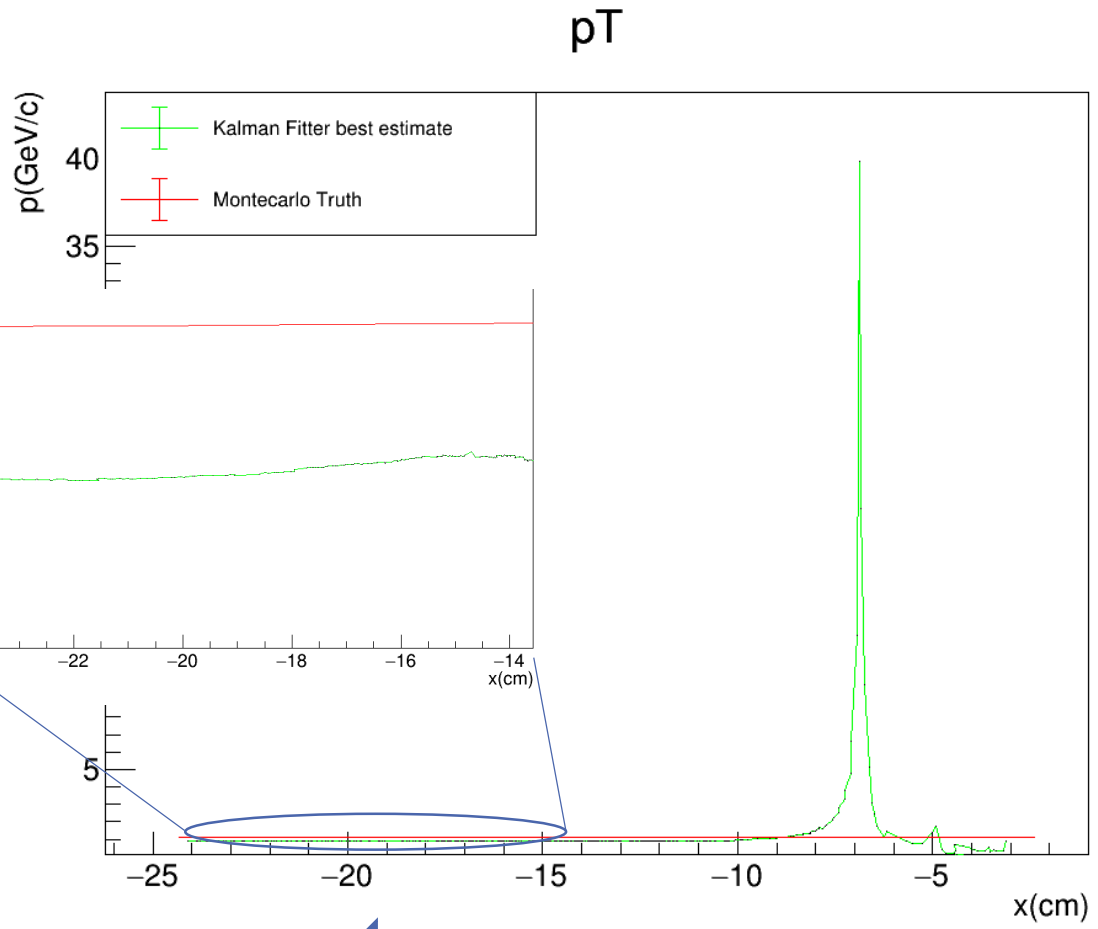
FITTER PROPAGATION DIRECTION

KALMAN FILTER PERFORMANCE

- Plots comparing the reconstructed tansverse momentum p_T from the Kalman fitter algorithm to the MC truth as a function of the free parameter x



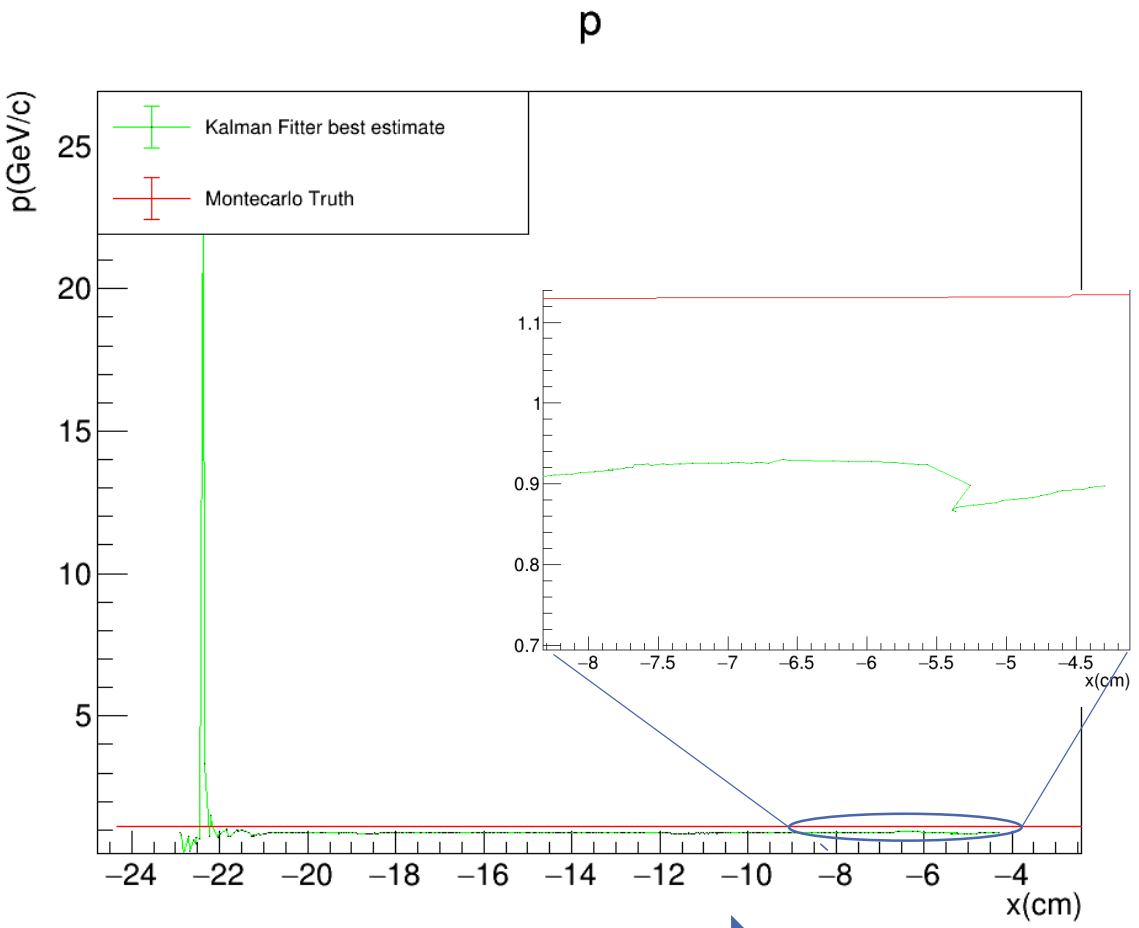
FORWARD FIT



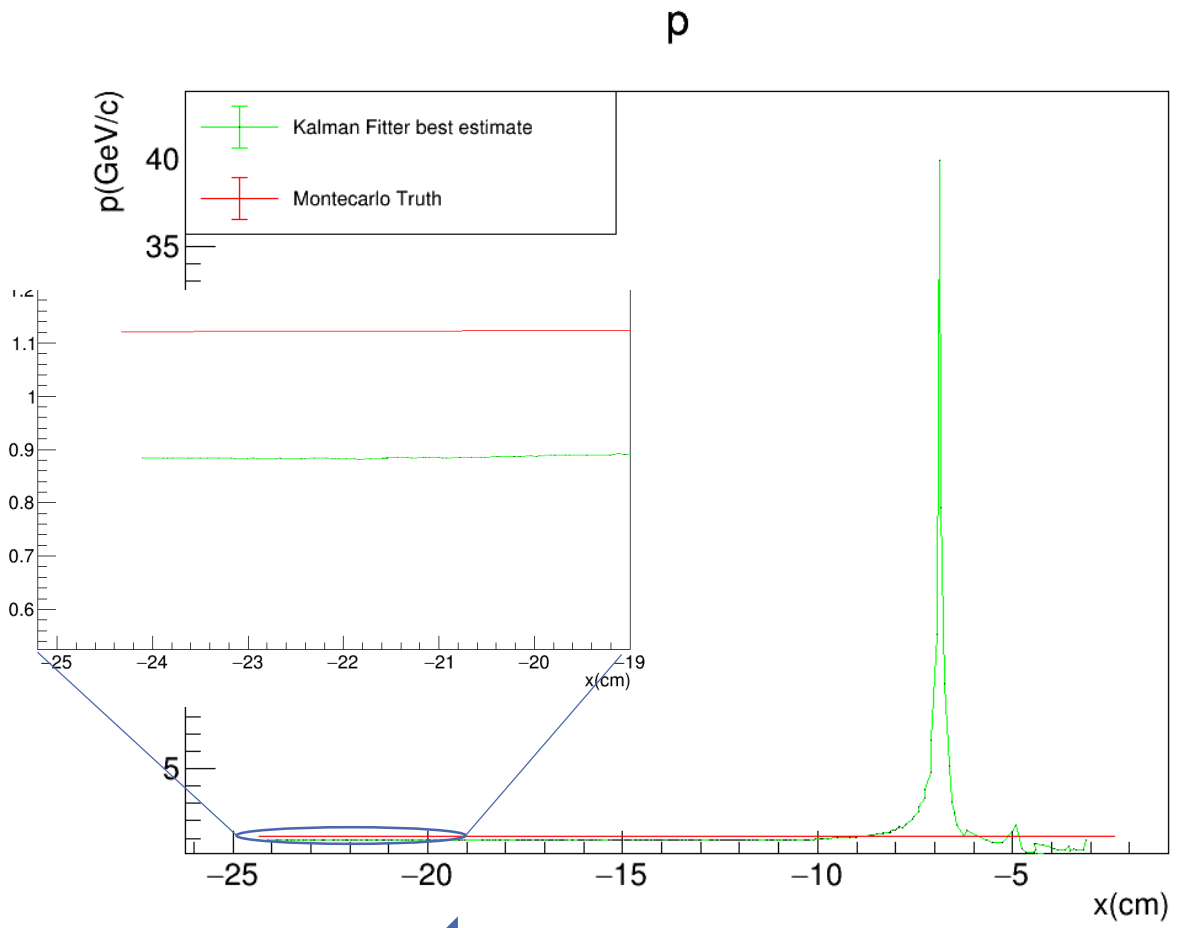
BACKWARD FIT

KALMAN FILTER PERFORMANCE

- Plots comparing the reconstructed total momentum p from the Kalman fitter algorithm to the MC truth as a function of the free parameter x



FORWARD FIT



BACKWARD FIT

SUMMARY AND FUTURE STEPS

- The $LAr \rightarrow GAr$ simulation chain is up and running, but much larger samples are needed
  - L2G: interface that takes outgoing LAr particles and feeds them to edep-sim with any TMS detector (currently starting to develop with Eldwan)

- In the Kalman Filter algorithm, at the moment, dE/dx (which is simulated in the MC propagation) is not taken into account in the prediction step and only accounted for by a posteriori measurement corrections
  - dE/dx inclusion in the prediction step could alleviate biases in the reconstruction

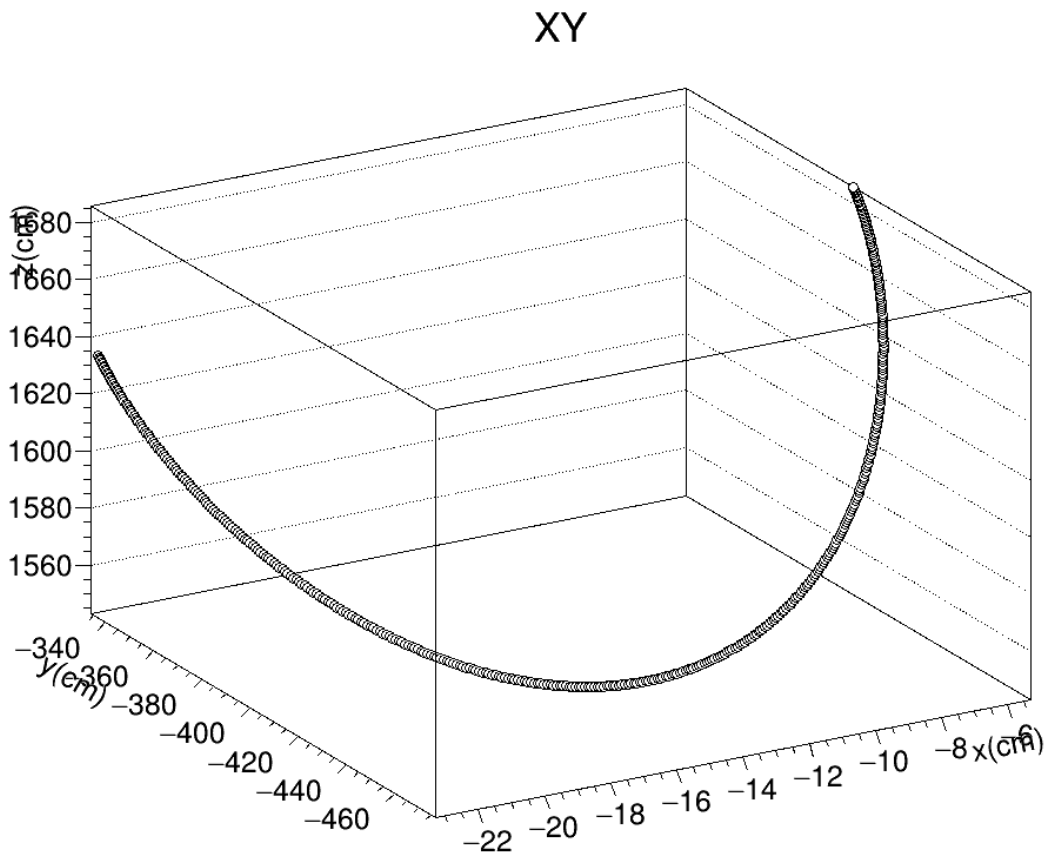
- Optimize the choice for the free parameter (currently drift coordinate) in the Kalman filter procedure

- Optimize the Kalman gain so that the measurements have more weight in the prediction

- In the $LAr \rightarrow GAr$ context Kalman filter and reconstruction in general need to be expanded to project tracks backwards to NDLAr and connect with the liquid Argon reconstruction information

KALMAN FILTER PERFORMANCE

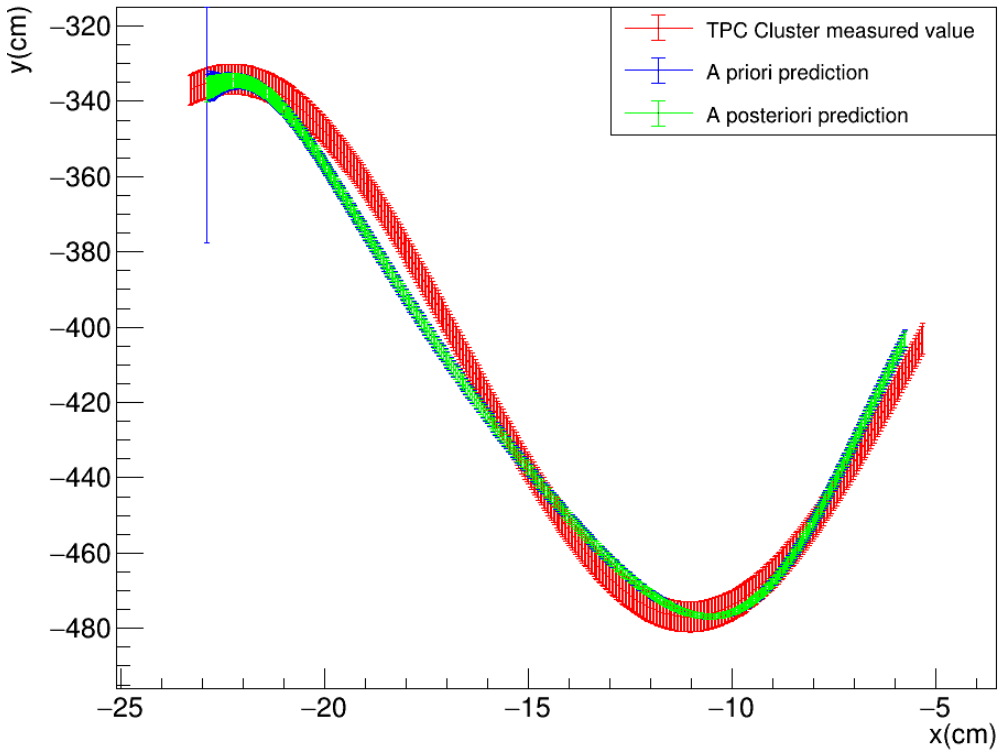
- Applied Kalman Filter to ideal measurements following perfectly an helix with initial coordinates: $x_i^T = (y_i \quad z_i \quad 1/r_i \quad \phi_i \quad \lambda_i) = (y_1^h \quad z_1^h \quad 0.014\ cm^{-1} \quad 6\ \text{rad} \quad -0.05\ \text{rad})$ and the free parameter $x_i = x_1^h$ and the step $dx = 0.04$ cm



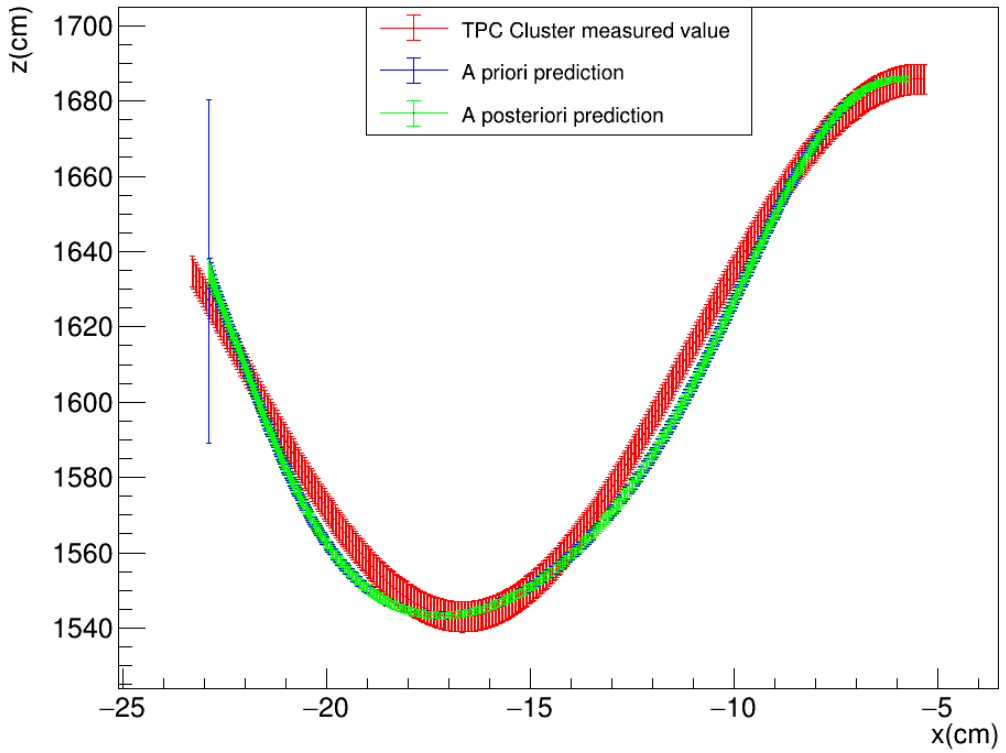
KALMAN FILTER PERFORMANCE

- Plots describing the evolution of the estimate of the measured quantities (y, z) as a function of the free parameter x for the perfect helix. Note that the the fake TPC points have on the x coordinate the fake measured value, while for the estimates we use the estimated x (i.e. $x_k = x_0 + k \times dx_k$)

Y



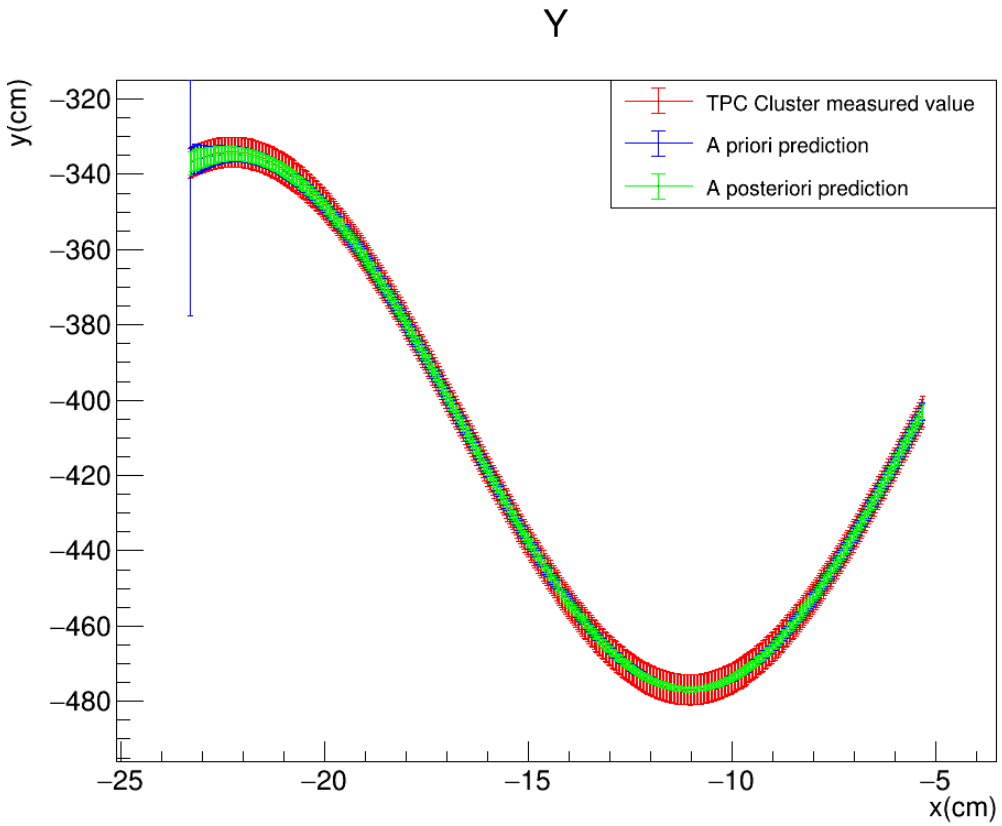
Z

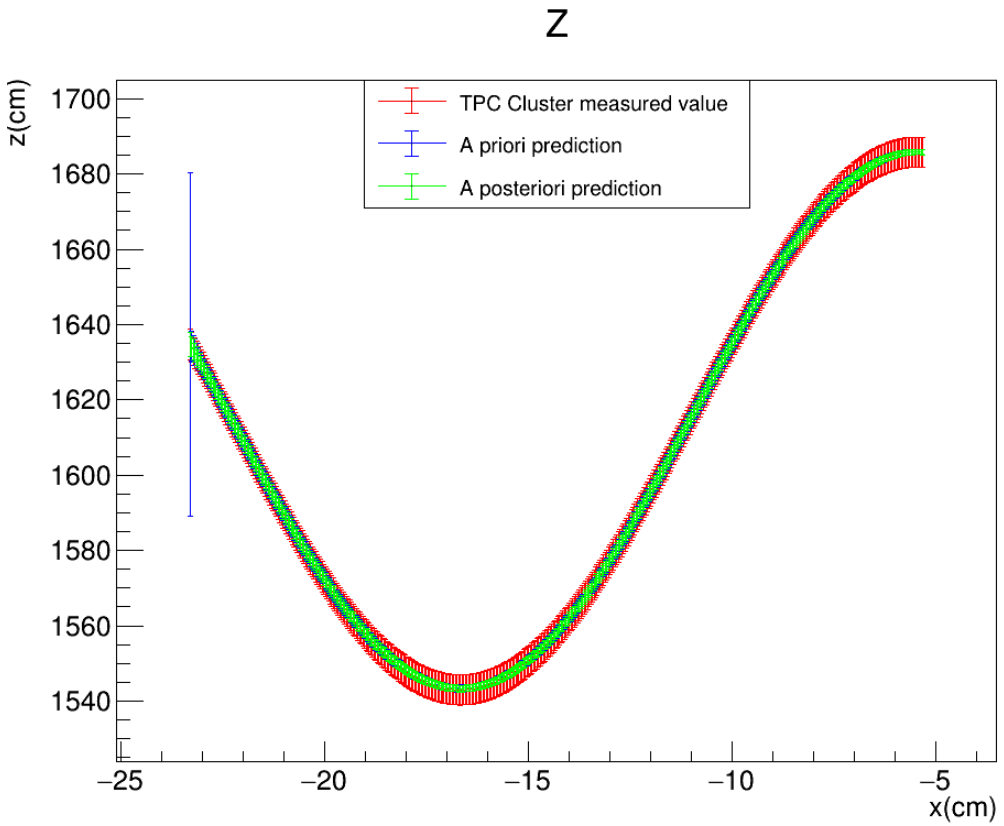


FITTER PROPAGATION DIRECTION

KALMAN FILTER PERFORMANCE

- Plots describing the evolution of the estimate of the measured quantities (y, z) as a function of the free parameter x for the perfect helix. This time both the fake TPC points have on the x coordinate the fake measured value

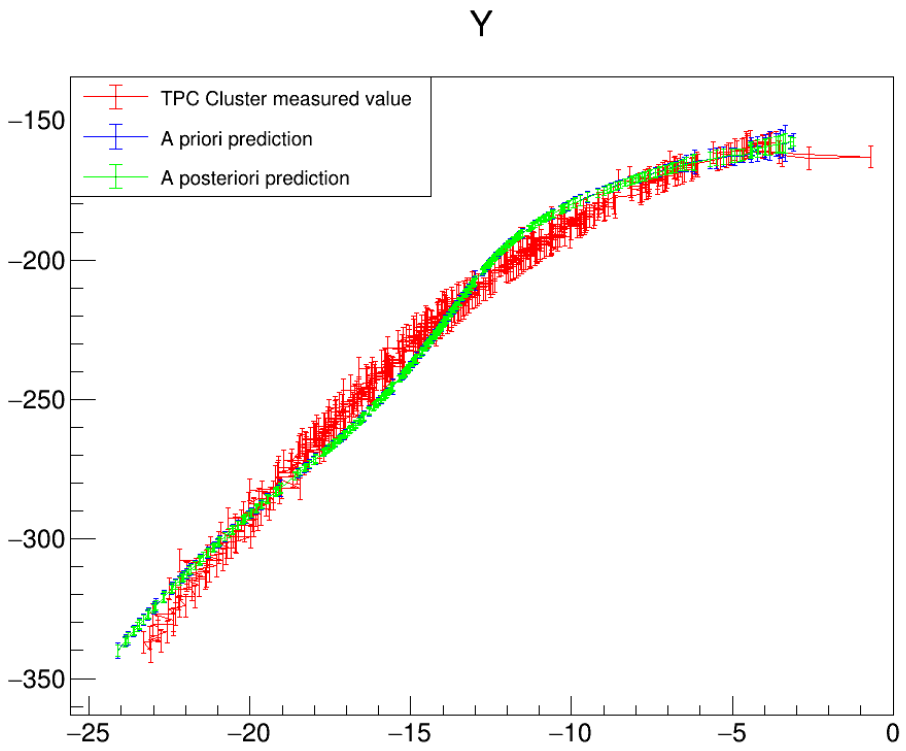


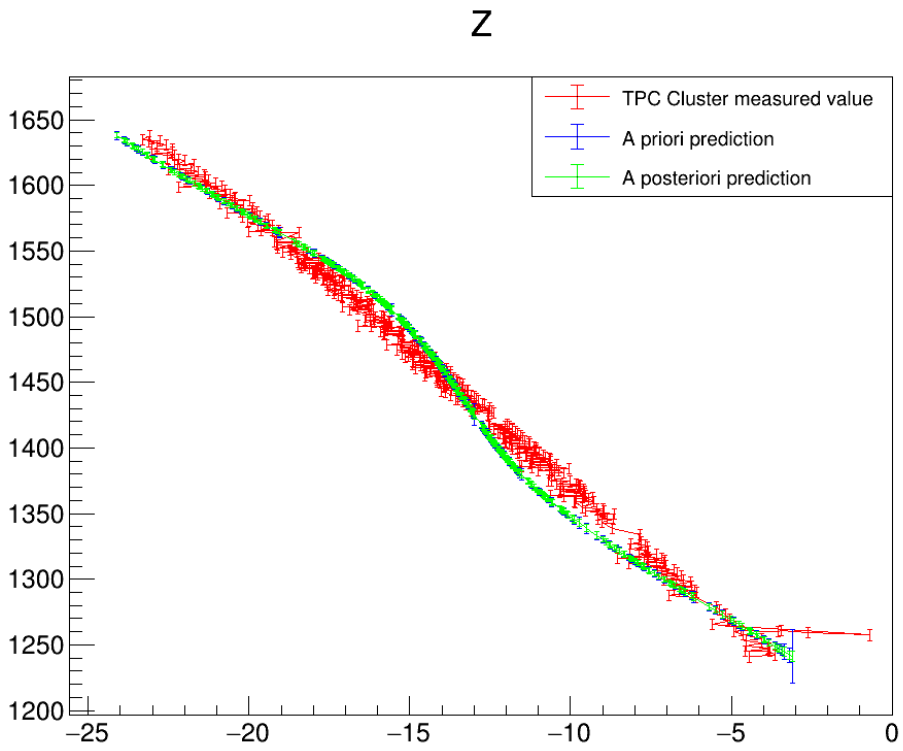


FITTER PROPAGATION DIRECTION

KALMAN FILTER: PERFORMANCE EVALUATION

- Going back to the anomalous backward fit plots, we first plot them like it was done originally (i.e. the fake TPC points have on the x coordinate the fake measured value, while for the estimates we use the estimated x)

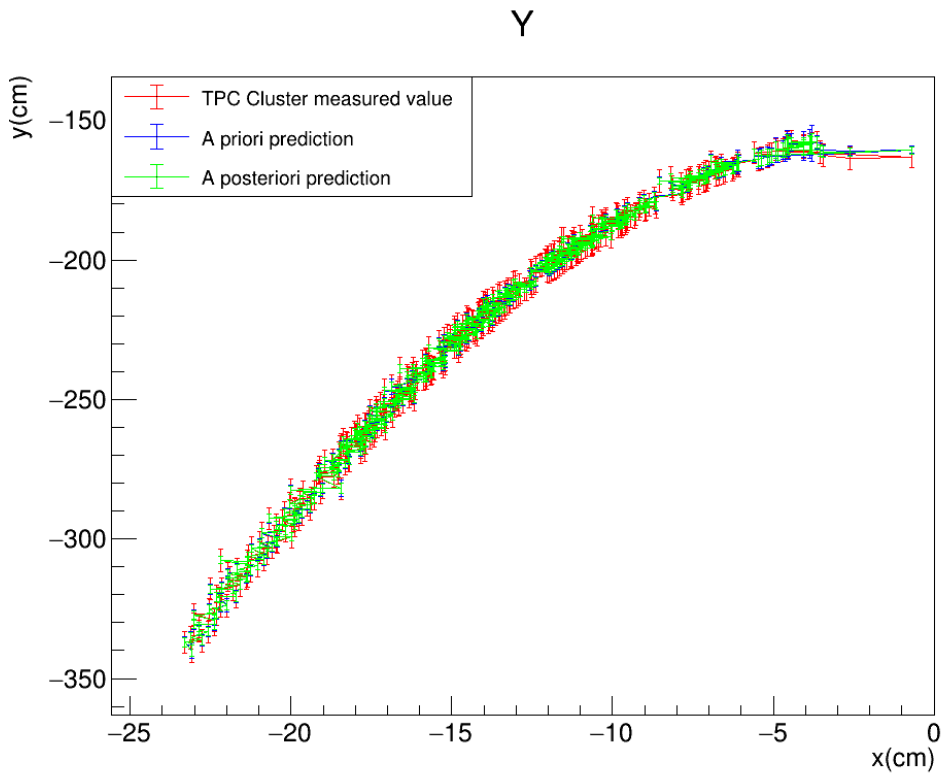


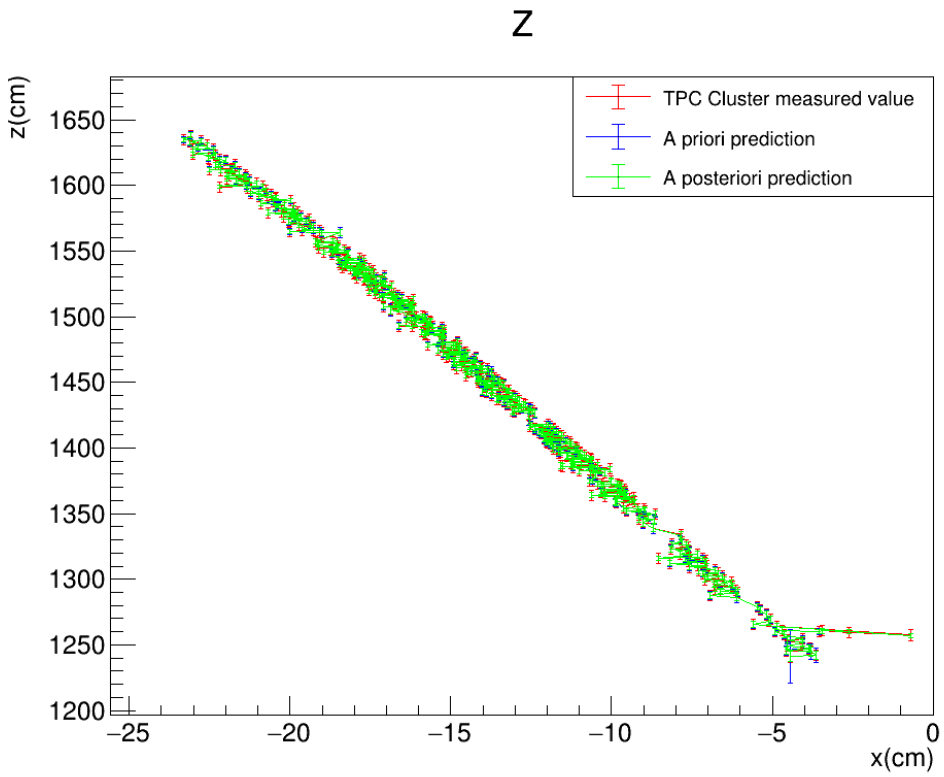


FITTER PROPAGATION DIRECTION

KALMAN FILTER: PERFORMANCE EVALUATION

- We now try using the measured value of x for everything





FITTER PROPAGATION DIRECTION

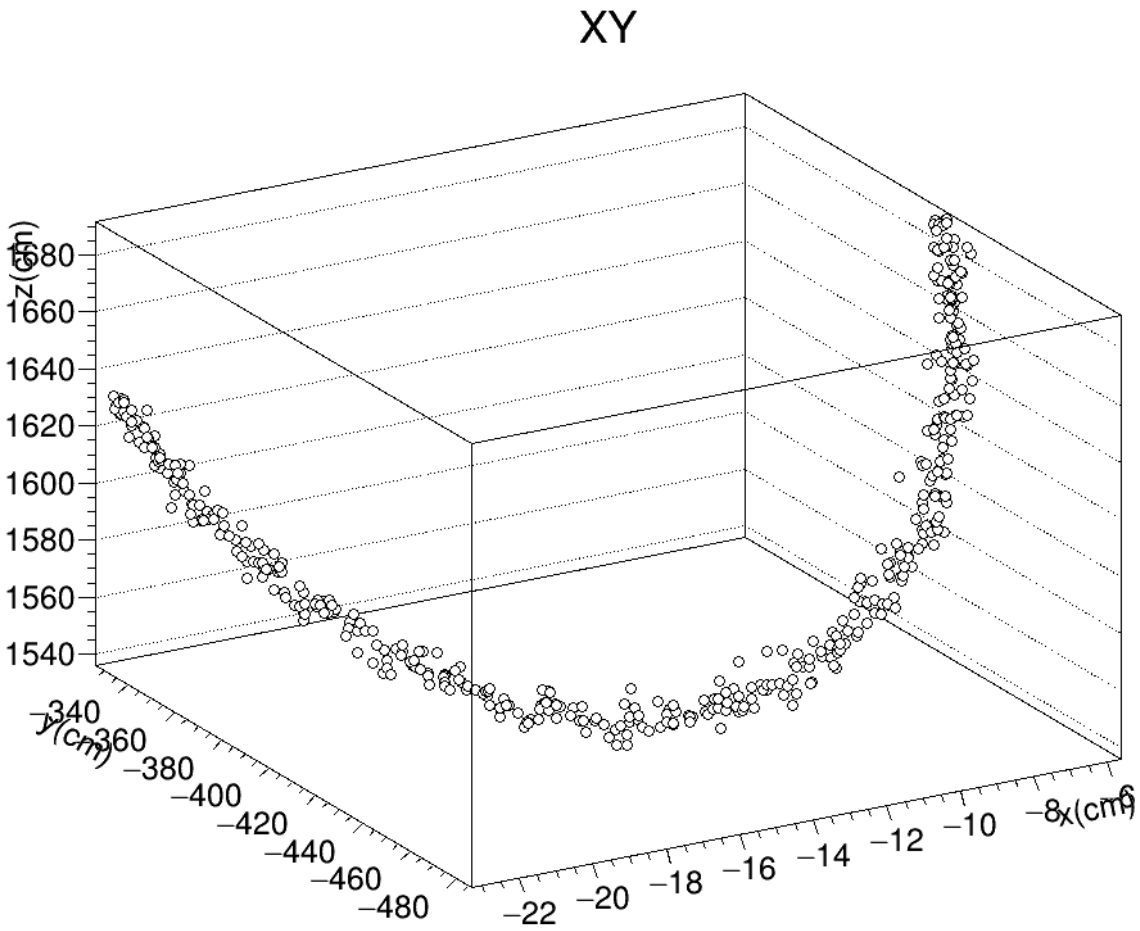
KALMAN FILTER PERFORMANCE

- Reset Toy MonteCarlo to have smeared helix points:
  same initial coordinates: $x_i^T =$
  $(y_i \quad z_i \quad 1/r_i \quad \phi_i \quad \lambda_i) =$
  $(y_1^h \quad z_1^h \quad 0.014\ cm^{-1} \quad 6\ \text{rad} \quad -0.05\ \text{rad})$

- The initial free parameter is $x_i = x_1^h$ with step $dx_k = Gauss(0.04\text{cm}, 0.01\text{cm})$

- After being evaluated for the current x_k as if we had a perfect helix, the y_k and z_k are smeared with a Gauss distibution: $Gauss(y_k, 4\text{cm})$

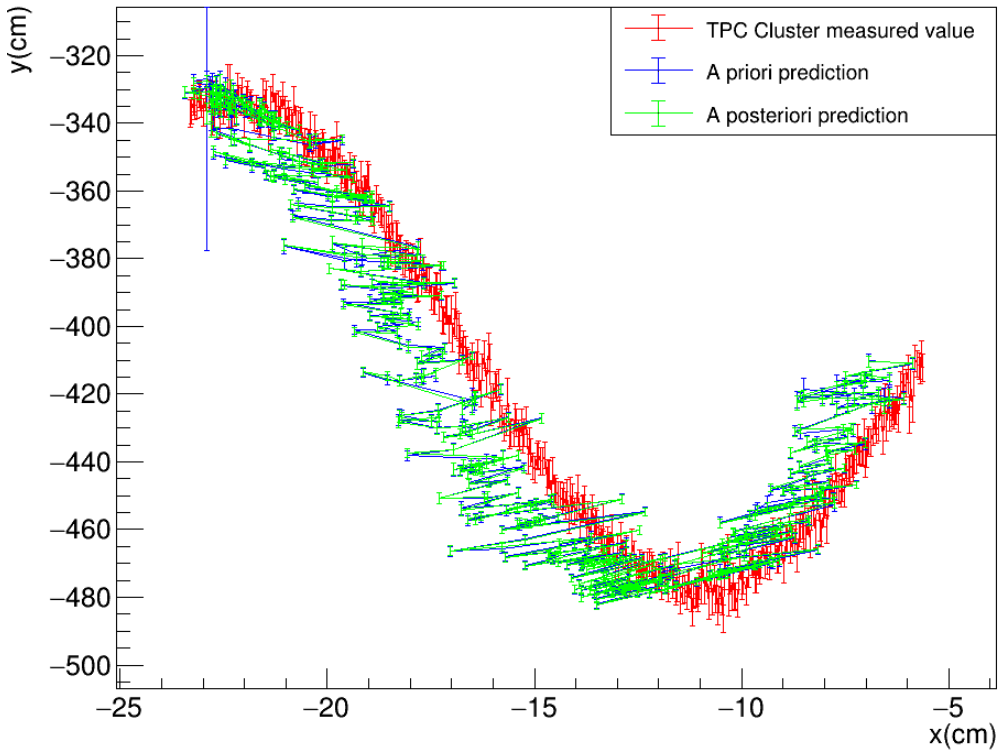
- The error matrix R is coherent with the smearing:
  $$R = \begin{pmatrix} \sigma_{yz}^2 & 0 \\ 0 & \sigma_{yz}^2 \end{pmatrix} = \begin{pmatrix} 16\text{cm}^2 & 0 \\ 0 & 16\text{cm}^2 \end{pmatrix}$$



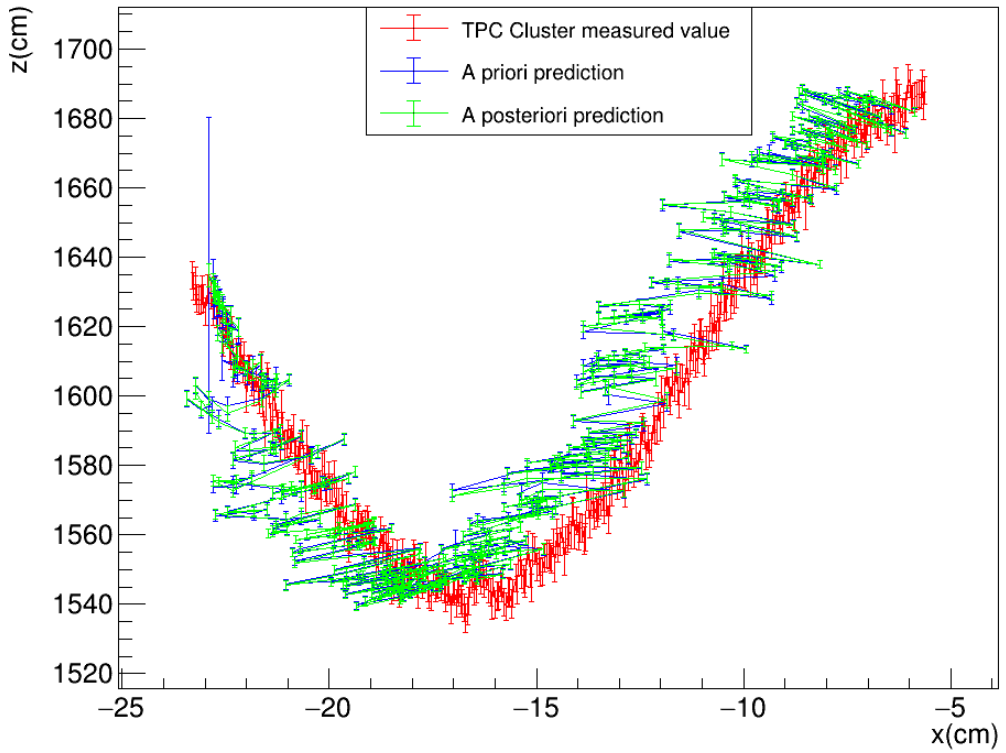
KALMAN FILTER PERFORMANCE

- Plots describing the evolution of the estimate of the measured quantities (y, z) as a function of the free parameter x for the perfect helix. Note that the the fake TPC points have on the x coordinate the fake measured value, while for the estimates we use the estimated x (i.e. $x_k = x_0 + k \times dx_k$)

Y



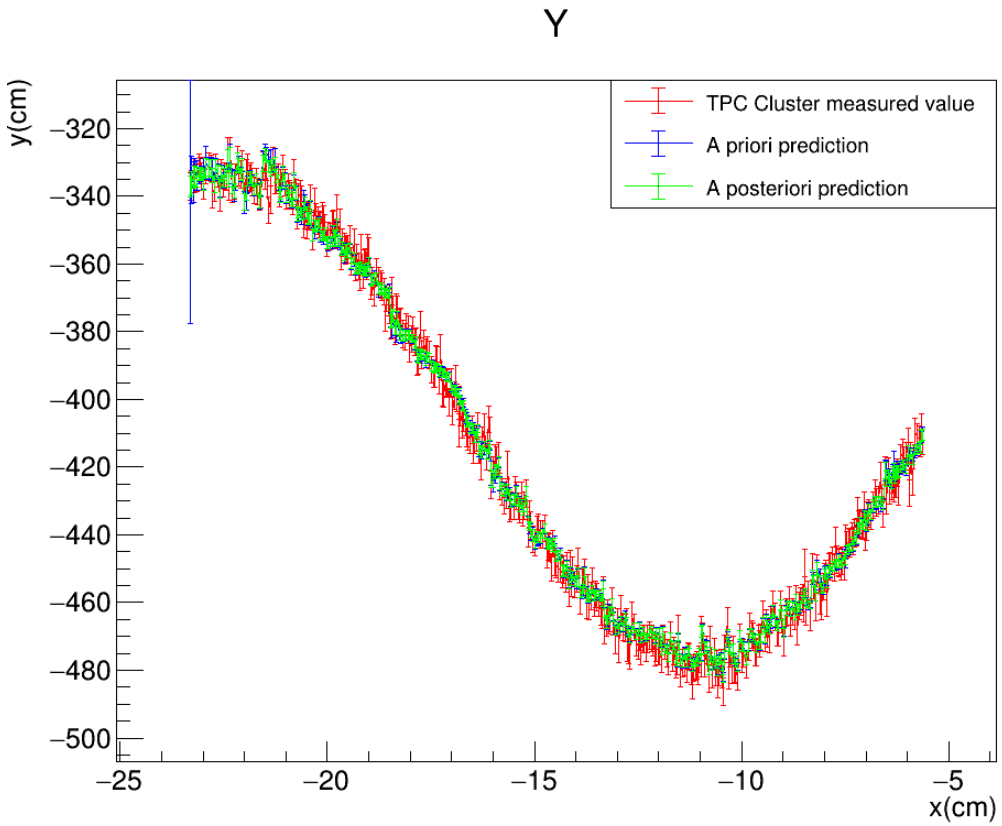
Z

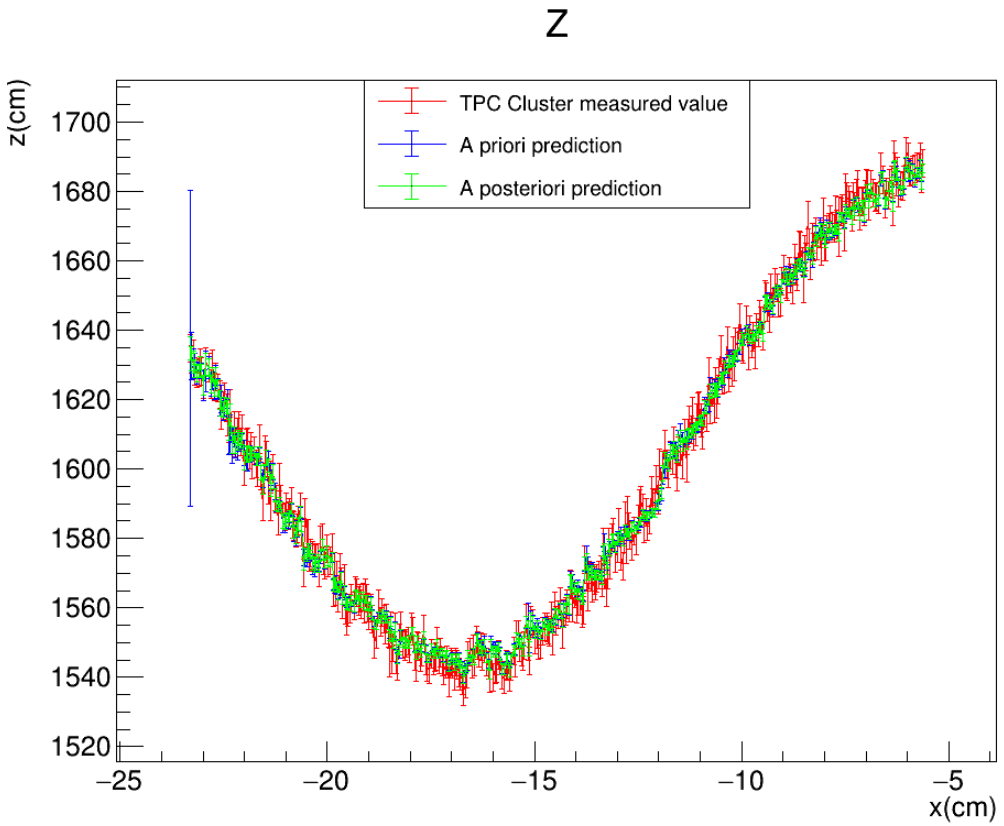


FITTER PROPAGATION DIRECTION

KALMAN FILTER PERFORMANCE

- Plots describing the evolution of the estimate of the measured quantities (y, z) as a function of the free parameter x for the perfect helix. This time both the fake TPC points have on the x coordinate the fake measured value





FITTER PROPAGATION DIRECTION

UNDERSTANDING STEP DETERMINATION

- The step width is determined for each algorithm step, so that it minimizes the distance between the current measurement and the a priori prediction:

$$\Delta = \left[\frac{\left(x_k^h - \hat{x}_k^-\right)}{\sigma_x}\right]^2 + \left[\frac{\left(y_k^h - \hat{y}_k^-\right)}{\sigma_{yz}}\right]^2 + \left[\frac{\left(z_k^h - \hat{z}_k^-\right)}{\sigma_{yz}}\right]^2$$

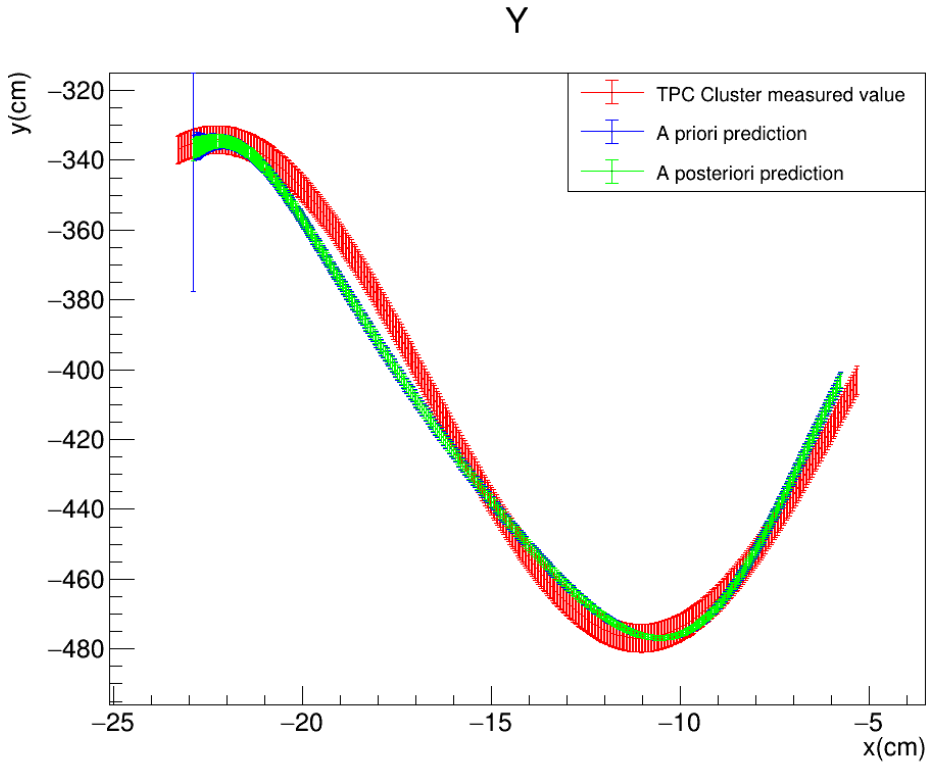
$\sigma_x = 1\text{cm}$ and $\sigma_{yz} = 2cm$ are arbitrary values, which do not coincide with the TPC values in the R matrix

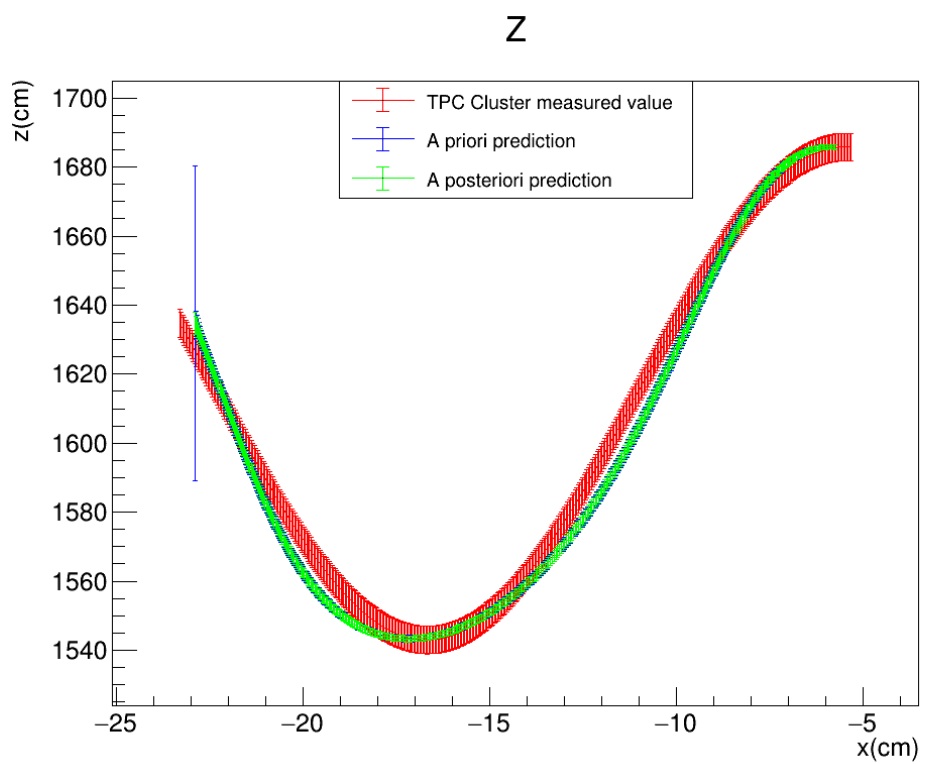
- The dx_k update formula can be devided into two parts: one that depends on the evolution of the x free parameter, and one on the measured quantities y and z

$$dx_k = \frac{\frac{\cot\hat{\lambda}_{k-1}^+}{\sigma_{yz}^2}\left(\left(y_k^h - \hat{y}_{k-1}^+\right)\sin\hat{\phi}_{k-1}^+ + \left(z_k^h - \hat{z}_{k-1}^+\right)\cos\hat{\phi}_{k-1}^+\right)}{\cot^2\hat{\lambda}_{k-1}^+ \Big/ \sigma_{yz}^2 + 1 \Big/ \sigma_x^2} + \frac{\frac{\left(x_k^h - \hat{x}_{k-1}^+\right)}{\sigma_x^2}}{\cot^2\hat{\lambda}_{k-1}^+ \Big/ \sigma_{yz}^2 + 1 \Big/ \sigma_x^2} = dx_{yz} + dx_x$$

UNDERSTANDING STEP DETERMINATION

- The values of σ_x and σ_{yz} determine how much the update is influenced by the measured values of x or of y and z respectivelly
- With the current sigma values dx_{yz} completely dominates, being often 2 or 3 orders of magnitude larger than dx_x: this gives us fairly accurate predictions for y and z, but completely wrong ones for x, becuase the fit can never recover from a bad initial estimate for the free parameter

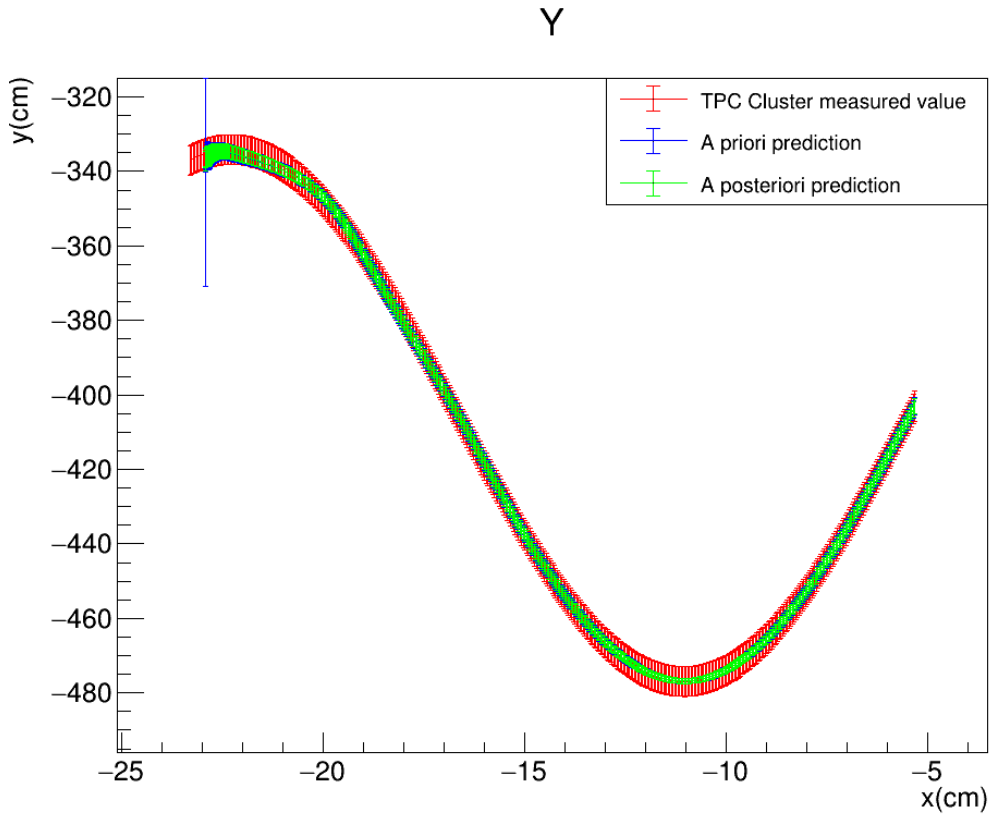


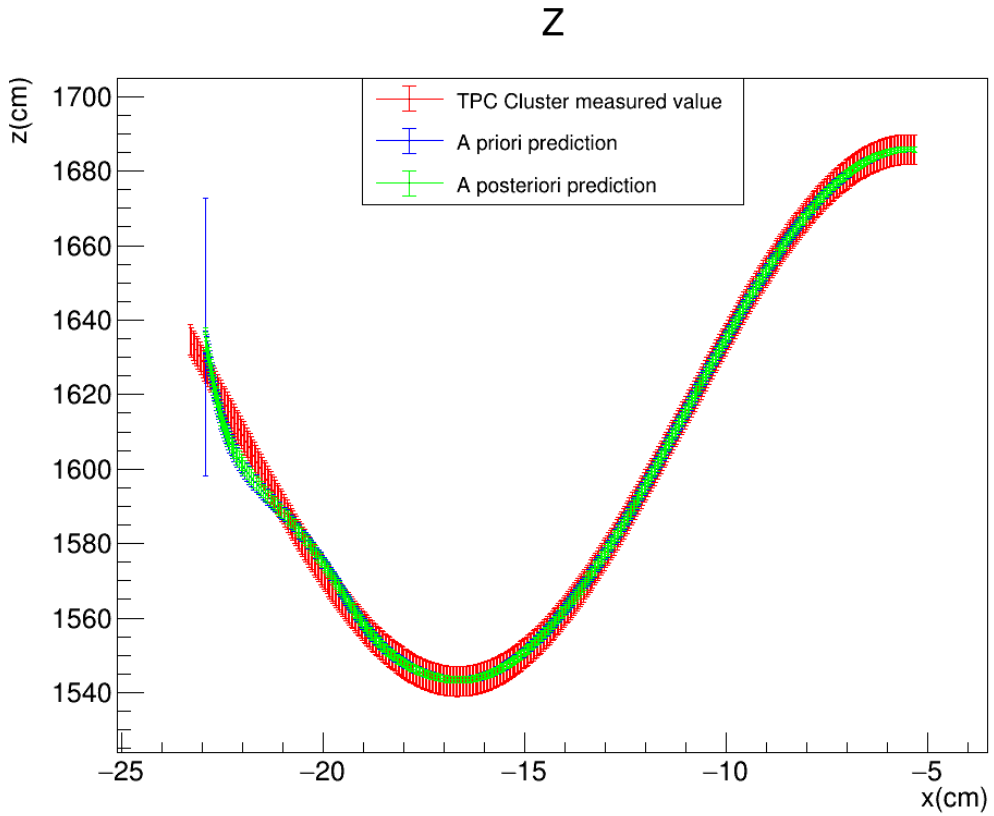


FITTER PROPAGATION DIRECTION

UNDERSTANDING STEP DETERMINATION

- In order for the x parameter to have more weight in the update of dx_k, changed the value of σ_{yz} from 1cm to 3cm
- The fit initially concentrates on fixing the x prediction until that becomes accurate, and then it focuses on yz, reaching similar levels of precision by the end

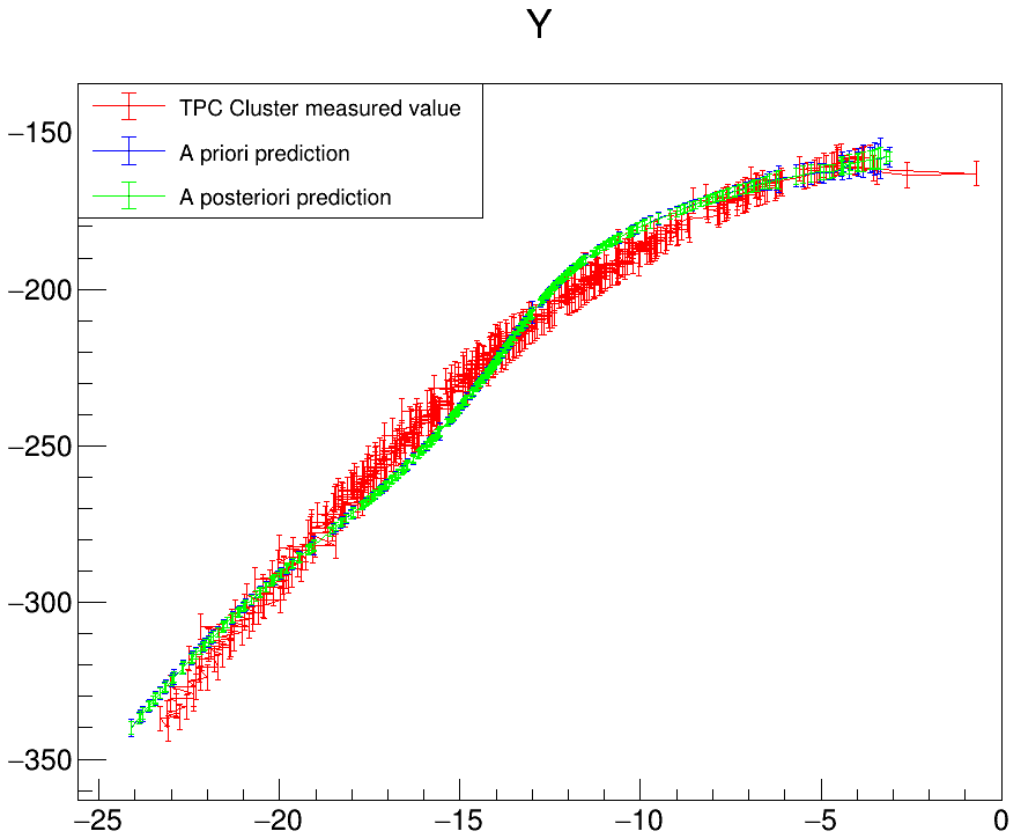




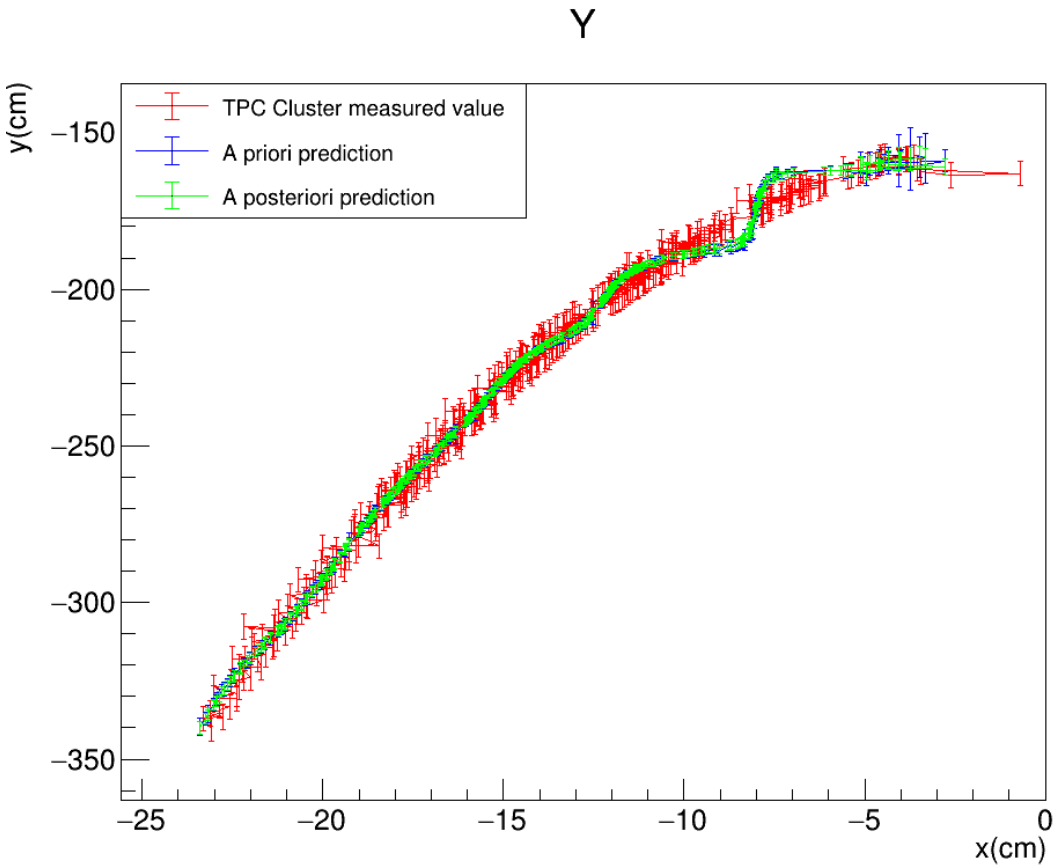
FITTER PROPAGATION DIRECTION

UNDERSTANDING STEP DETERMINATION

- Reapplying the fit with the new $\sigma_{yz} = 4$cm we see that the 3D predictions are more in line with the Montecarlo truth, past the first few steps



$\sigma_{yz} = 2cm$

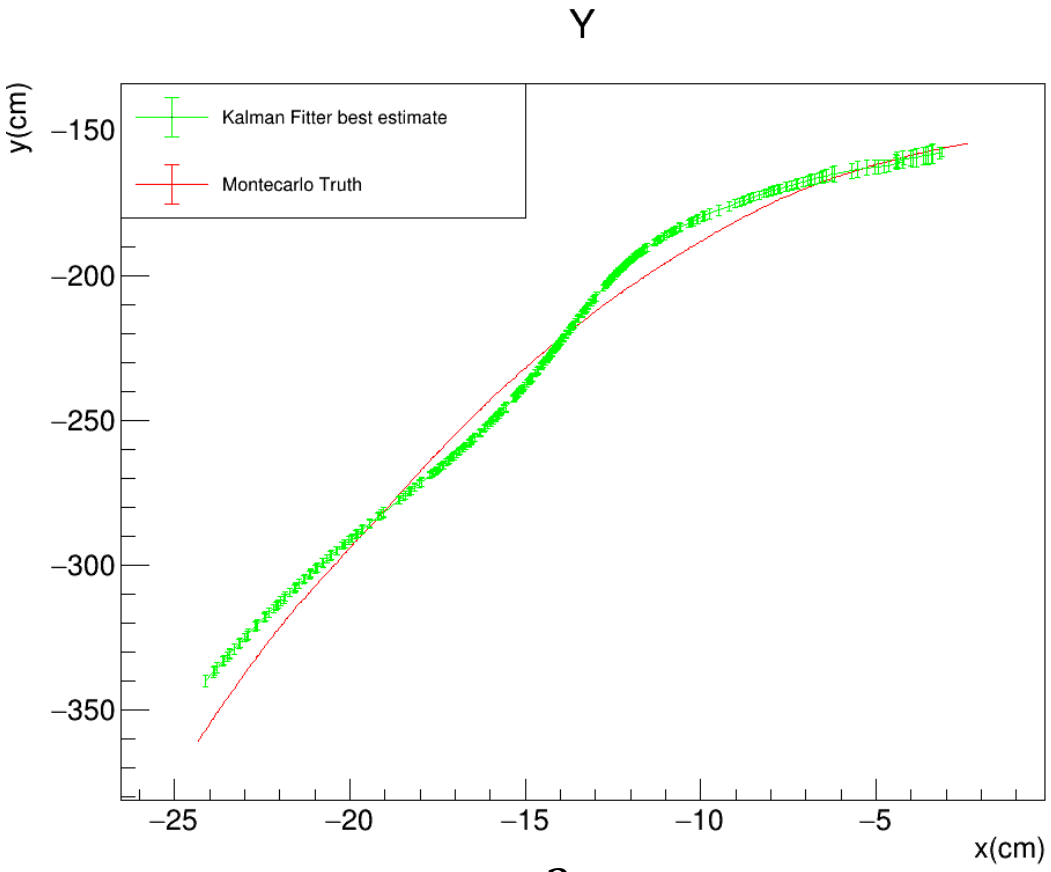


$\sigma_{yz} = 4cm$

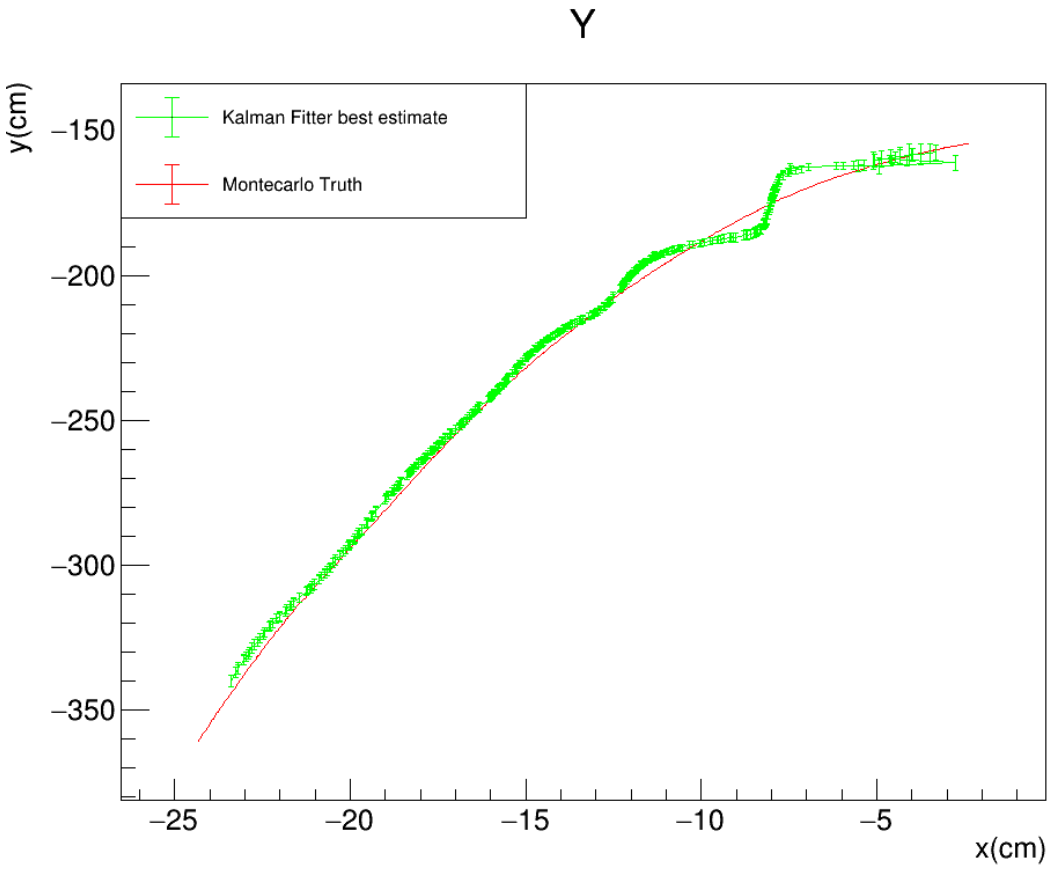
FITTER
OPAGATION
IRECTION

UNDERSTANDING STEP DETERMINATION

- Reapplying the fit with the new $\sigma_{yz} = 4$cm we see that the 3D predictions are more in line with the Montecarlo truth, past the first few steps



$\sigma_{yz} = 2cm$

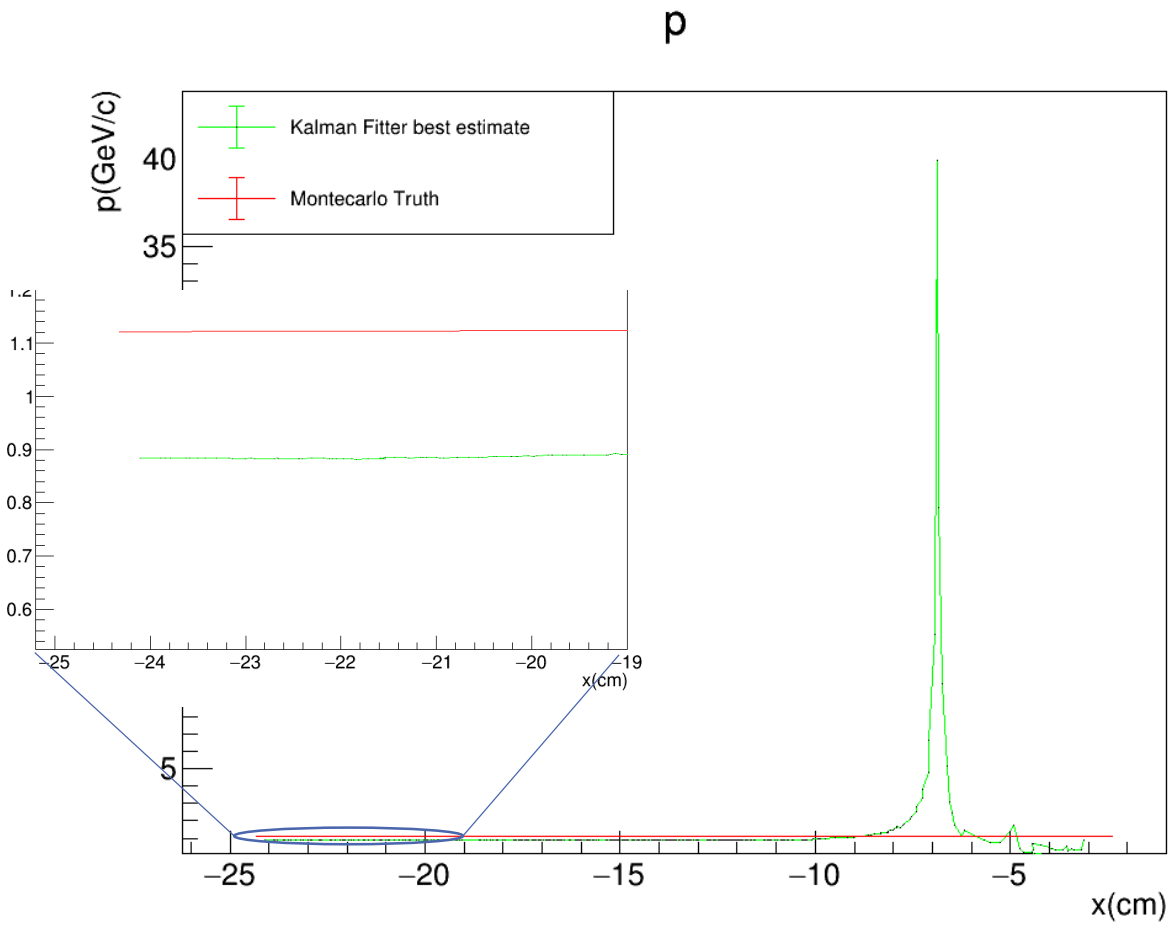


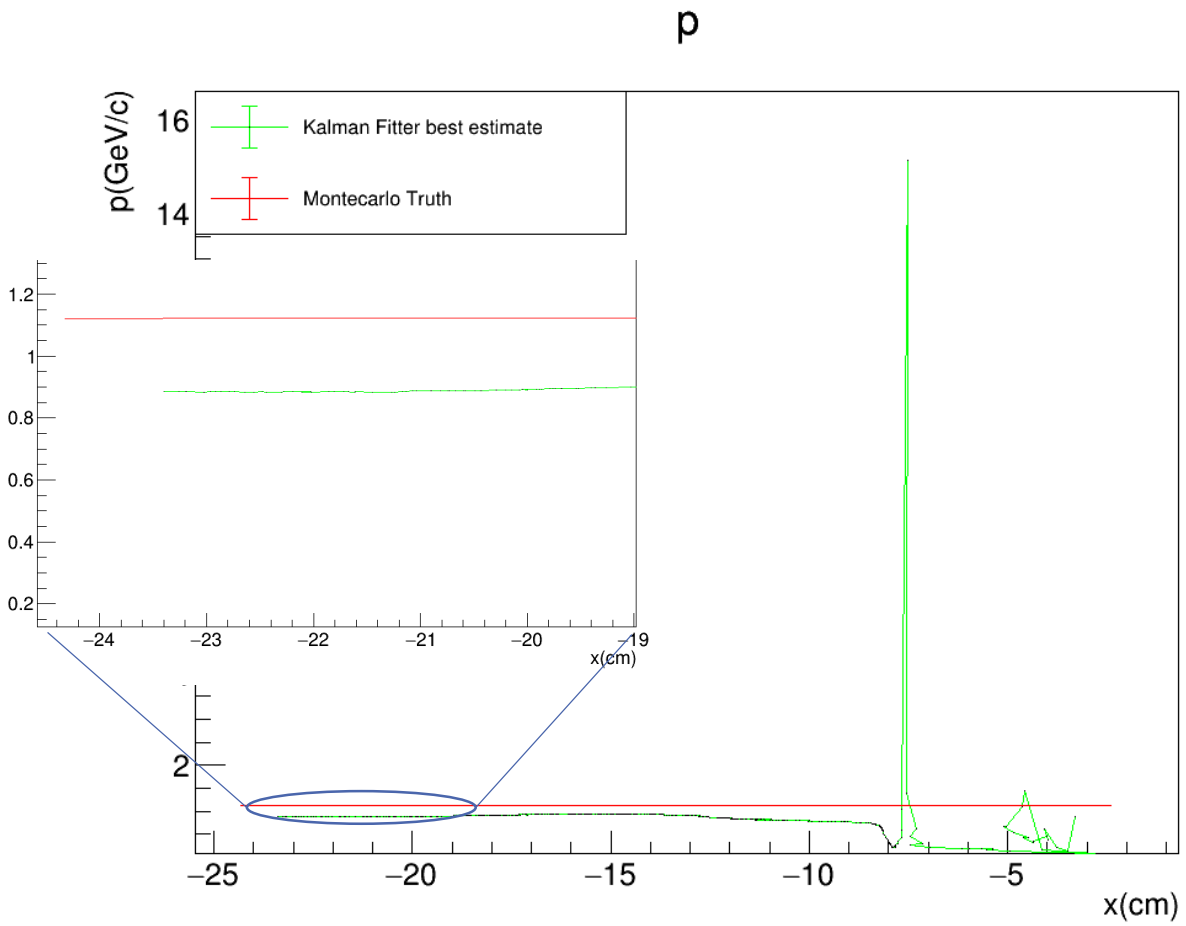
$\sigma_{yz} = 4cm$

FITTER PROPAGATION DIRECTION

UNDERSTANDING STEP DETERMINATION

- The momentum reconstruction performance remains roughly the same





BACKWARD FIT