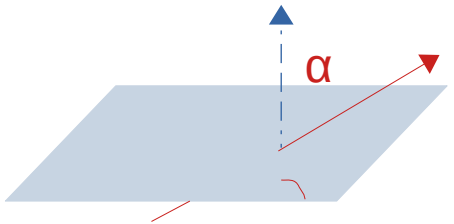
Second Order Kalman filtering

Xiaocong Ai

July 26, 2021

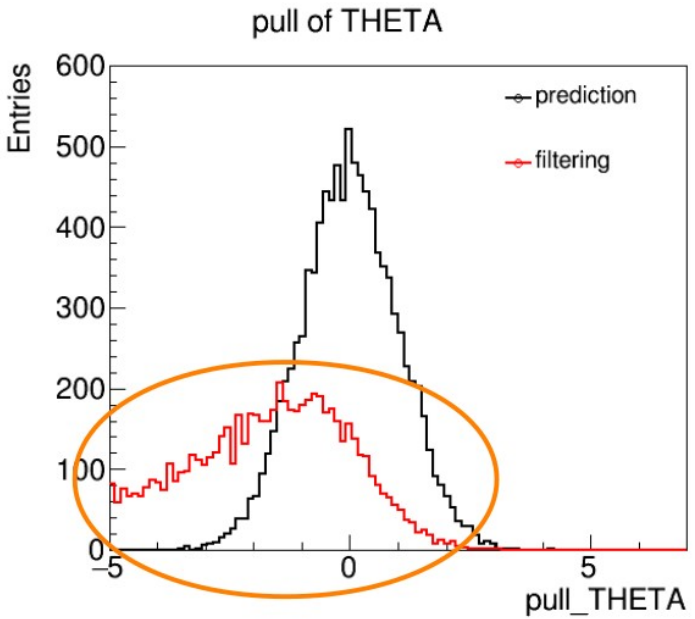
The non-linearity in tracking

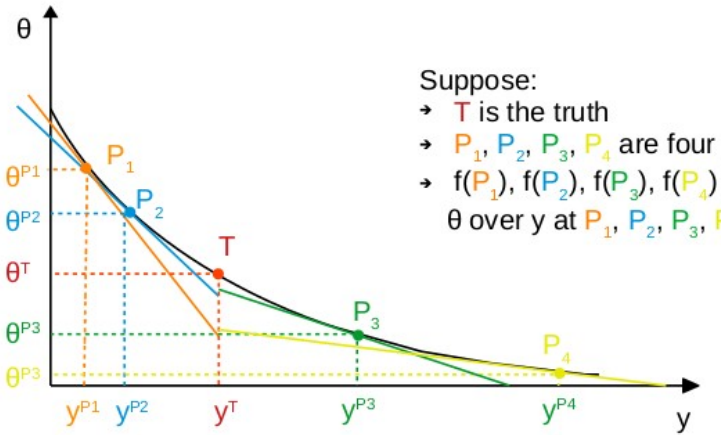


- One example of non-linearity: the dependence of the intersected local position on the track direction is non-linear when α>>0
- Other non-linearity happens during: cartesian coordinate ↔ polar/cylindrical/perigee coordinate etc.

Impact on the Kalman filtering

See details here





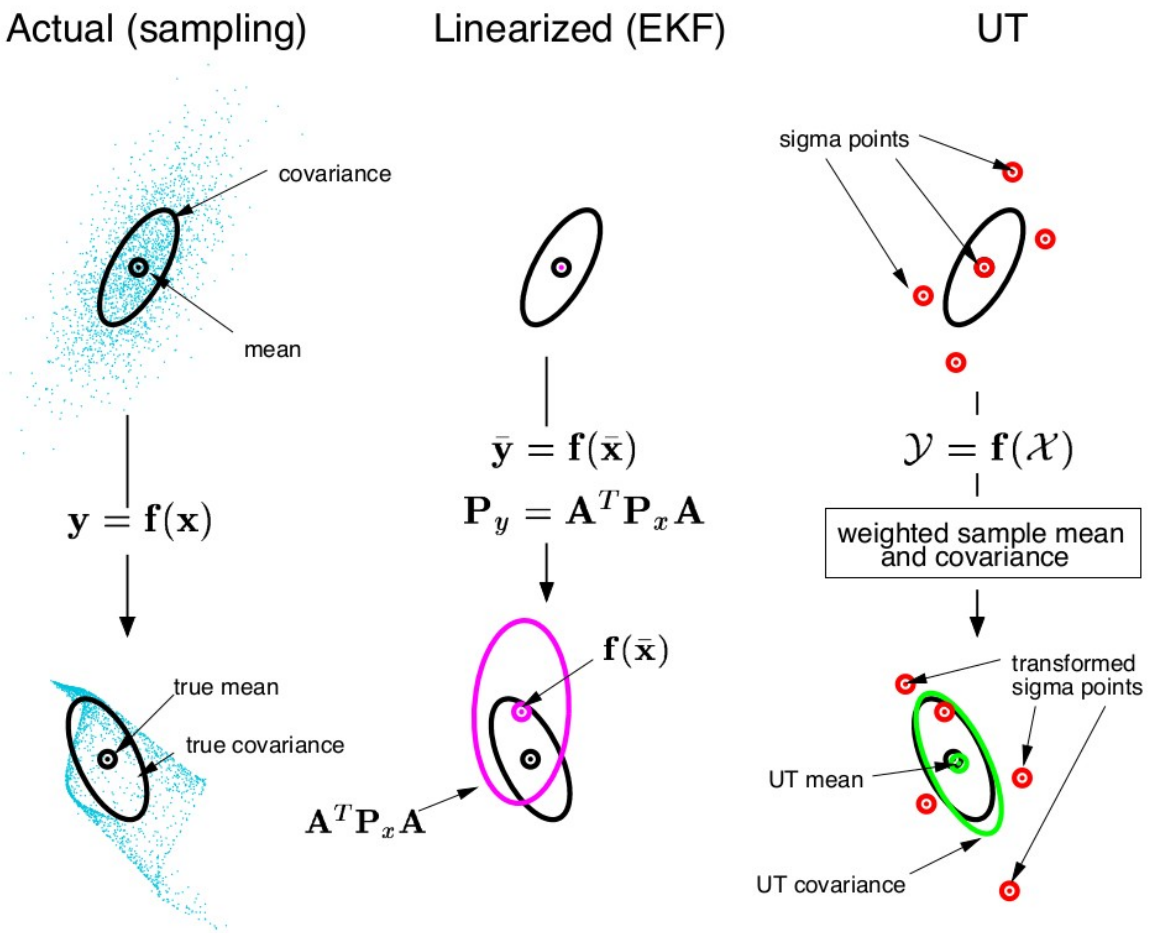
Suppose:
- T is the truth
- P_1, P_2, P_3, P_4 are four possible prediction
- $f(P_1)$, $f(P_2)$, $f(P_3)$, $f(P_4)$ are the derivative o θ over y at P_1, P_2, P_3, P_4, respectively.

When we use information at prediction to estimate the value of θ at truth, we will get:
- $\theta^{P1} + f(P_1)*(y^{P1}-y^T) < \theta^T$
- $\theta^{P2} + f(P_2)*(y^{P2}-y^T) < \theta^T$
- $\theta^{P3} + f(P_3)*(y^{P3}-y^T) < \theta^T$
- $\theta^{P4} + f(P_4)*(y^{P4}-y^T) < \theta^T$

$$K_k = C_k^{k-1} H_k^T (V_k + H_k C_k^{k-1} H_k^T)^{-1}$$
$$x_k = x_k^{k-1} + K_k \left(m_k - h_k(x_k^{k-1})\right)$$
$$C_k = (1 - K_k H_k)\, C_k^{k-1}$$

i.e. the estimated value of θ will always be smaller than θ^T !

The idea of Unscented Kalman Filter (UKF)



KF for non-linearity system

Idea: Jacobian→ Parameters evaluation at sigma points

See ref

The n-dimensional random variable $\mathbf{x}$ with mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}_{xx}$ is approximated by $2n+1$ weighted points given by

$$\begin{aligned} \boldsymbol{\mathcal{X}}_0 &= \bar{\mathbf{x}} & W_0 &= \kappa/(n+\kappa) \\ \boldsymbol{\mathcal{X}}_i &= \bar{\mathbf{x}} + \left(\sqrt{(n+\kappa)\mathbf{P}_{xx}}\right)_i & W_i &= 1/2(n+\kappa) \\ \boldsymbol{\mathcal{X}}_{i+n} &= \bar{\mathbf{x}} - \left(\sqrt{(n+\kappa)\mathbf{P}_{xx}}\right)_i & W_{i+n} &= 1/2(n+\kappa) \end{aligned} \tag{12}$$

where $\kappa \in \Re$, $\left(\sqrt{(n+\kappa)\mathbf{P}_{xx}}\right)_i$ is the ith row or column of the matrix square root of $(n+\kappa)\mathbf{P}_{xx}$ and W_i is the weight which is associated with the ith point. The transformation procedure is as follows:

1. Instantiate each point through the function to yield the set of transformed sigma points,

$$\boldsymbol{\mathcal{Y}}_i = \mathbf{f}\left[\boldsymbol{\mathcal{X}}_i\right].$$

2. The mean is given by the weighted average of the transformed points,

$$\bar{\mathbf{y}} = \sum_{i=0}^{2n} W_i \boldsymbol{\mathcal{Y}}_i. \tag{13}$$

3. The covariance is the weighted outer product of the transformed points,

$$\mathbf{P}_{yy} = \sum_{i=0}^{2n} W_i \left\{\boldsymbol{\mathcal{Y}}_i - \bar{\mathbf{y}}\right\}\left\{\boldsymbol{\mathcal{Y}}_i - \bar{\mathbf{y}}\right\}^T. \tag{14}$$

The implementation in ACTS

```
struct CorrectedBoundToFreeTransformer {
  /// The parameter to tune the weight
  ActsScalar kappa = 2;

/// Get the non-linearity corrected free parameters and its covariance
std::optional<std::tuple<FreeVector, FreeSymMatrix, FreeToBoundMatrix>>
operator()(const BoundVector& boundParams,
           const BoundSymMatrix& boundCovariance, const Surface& surface,
           const GeometryContext& geoContext) {
  size_t sampleSize = 2 * eBoundSize + 1;
  std::vector<std::pair<BoundVector, ActsScalar>> sampledBoundParams;
  sampledBoundParams.reserve(sampleSize);
```

```
truct CorrectedFreeToBoundTransformer {
  /// The parameter to tune the weight
  ActsScalar kappa = 4;

  /// Get the non-linearity corrected bound parameters and its covariance
  std::optional<std::tuple<BoundVector, BoundSymMatrix, BoundToFreeMatrix>>
  operator()(const FreeVector& freeParams, const FreeSymMatrix& freeCovariance,
             const FreeVector& freeDerivatives, const Surface& surface,
             const GeometryContext& geoContext,
             NavigationDirection navDir = forward) {
   size_t sampleSize = 2 * eFreeSize + 1;
   std::vector<std::pair<FreeVector, ActsScalar>> sampledFreeParams;
   sampledFreeParams.reserve(sampleSize);
```

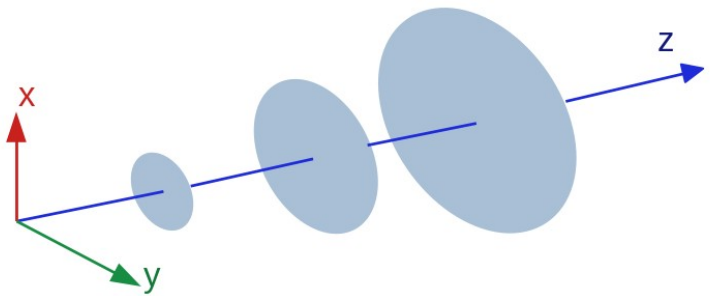
```
std::vector<FreeVector> transformedFreeParams;
// Loop over the sample points of the free parameters to get the weighted
// bound parameters
for (const auto& [params, weight] : sampledBoundParams) {
  // Transform the bound to free
  auto fp =
      detail::transformBoundToFreeParameters(surface, geoContext, params);
  transformedFreeParams.push_back(fp);
  fpMean += weight * fp;
}
```

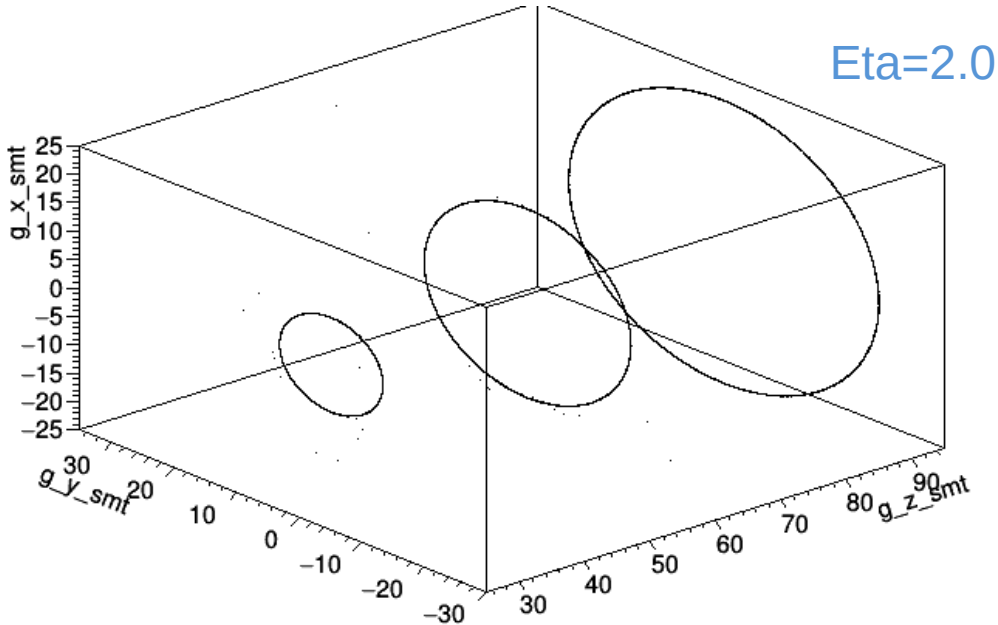
Get the transformed parameters for the sampled parameters

```
std::vector<BoundVector> transformedBoundParams;
// Loop over the sample points of the free parameters to get the weighted
// bound parameters
for (const auto& [params, weight] : sampledFreeParams) {

  auto correctedFreeParams = params;
  // Reintersect to get the corrected free params
  auto intersection =
      surface.intersect(geoContext, params.segment<3>(eFreePos0),
                        navDir * params.segment<3>(eFreeDir0), true);
  // The new position and time?
  correctedFreeParams.segment<3>(eFreePos0) =
      intersection.intersection.position;

  // Transform the free to bound
  auto result = detail::transformFreeToBoundParameters(correctedFreeParams,
                                                       surface, geoContext);
  if (not result.ok()) {
    continue;
  }
  auto bp = result.value();
  transformedBoundParams.push_back(bp);
  bpMean = bpMean + weight * bp;
}
```

Test with Telescope detector with disc surface

No material
Zero B filed

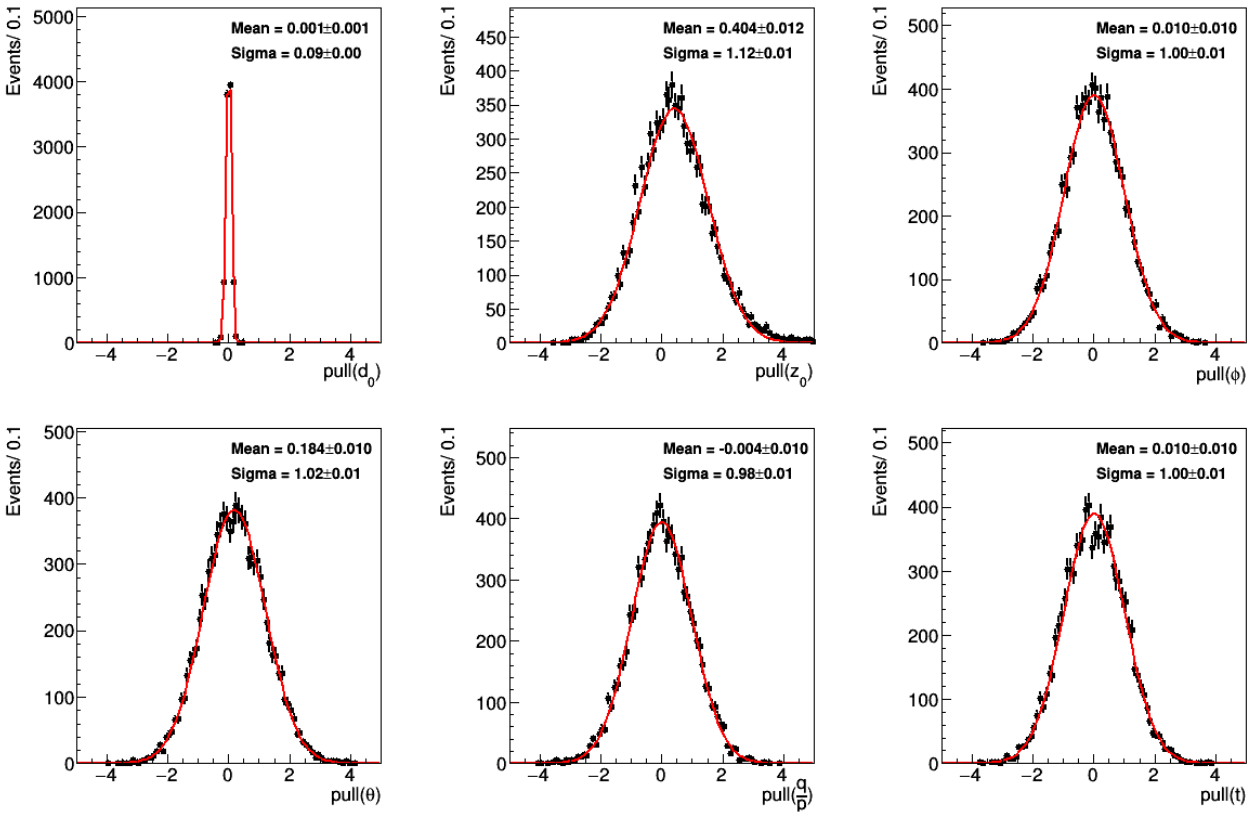




hitSmearingCfg.sigmaLoc0 = 25_um;
hitSmearingCfg.sigmaLoc1 = 0.1_rad ;

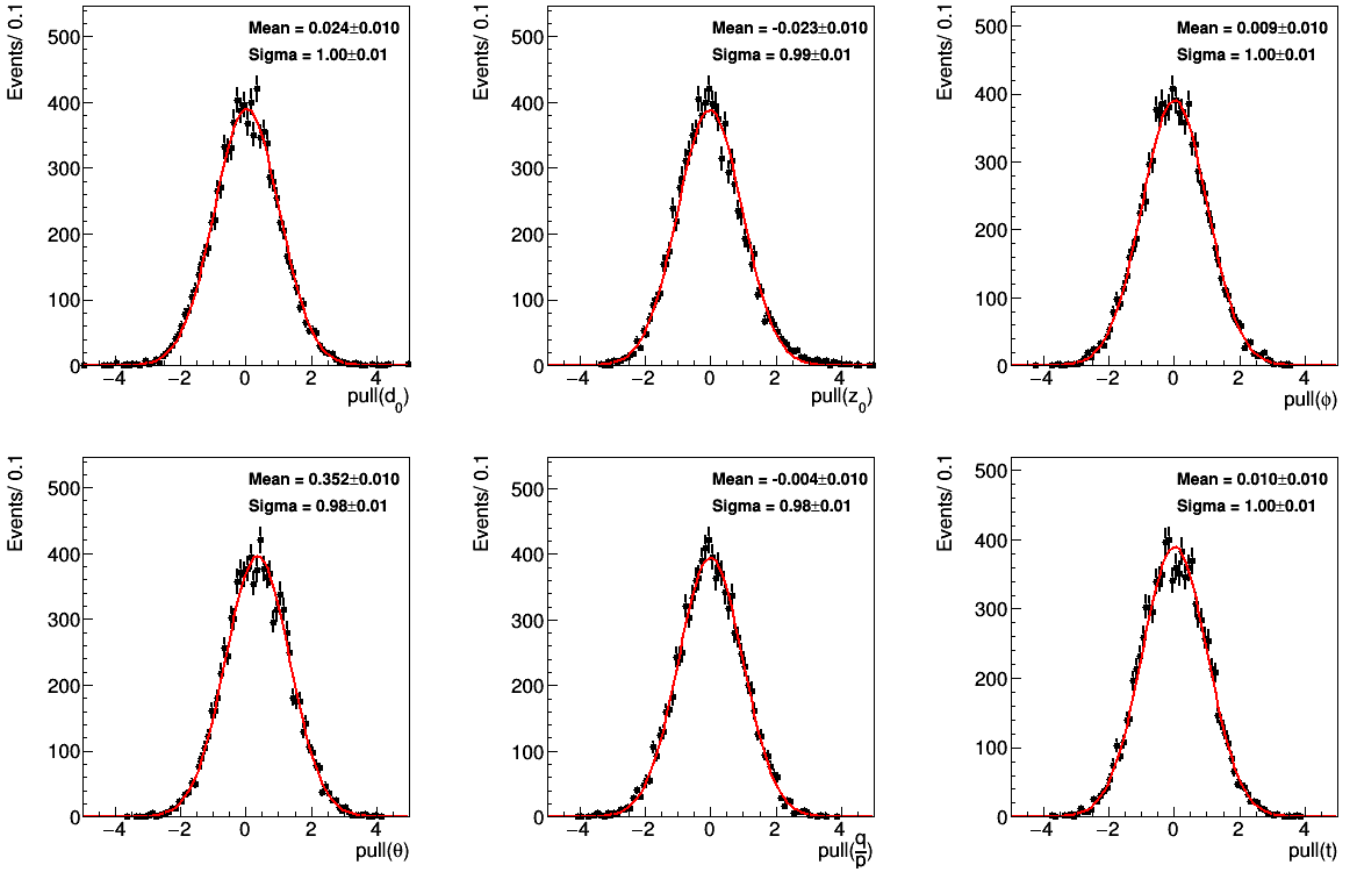
Fitted perigee params with disc telescope

No non-linearity correction



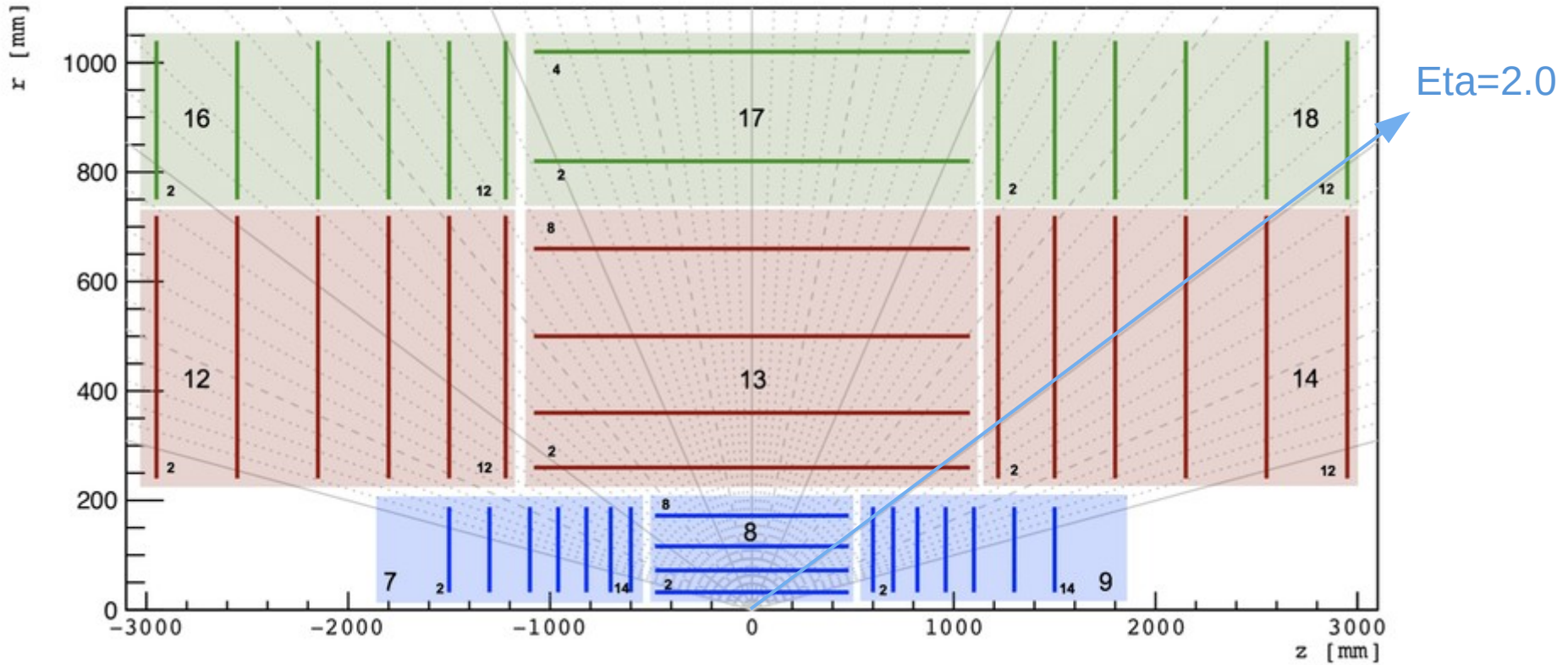
Fitted perigee params with disc telescope

With non-linearity correction



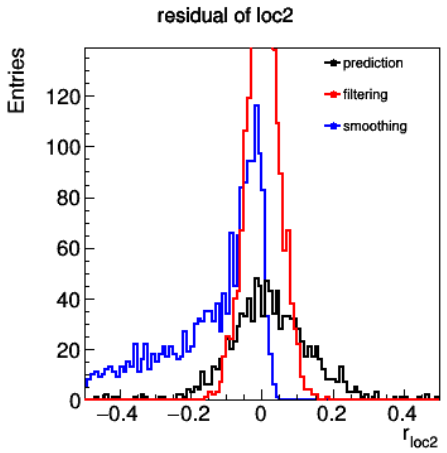
Test with generic detector

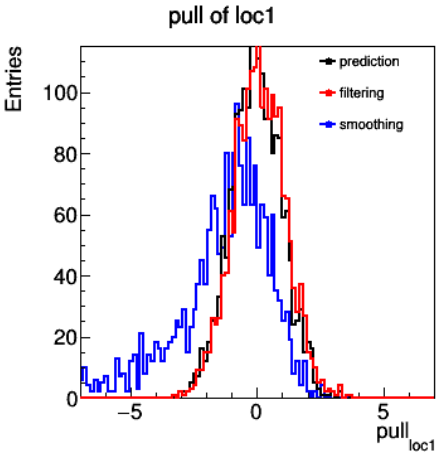
No material
Zero B filed

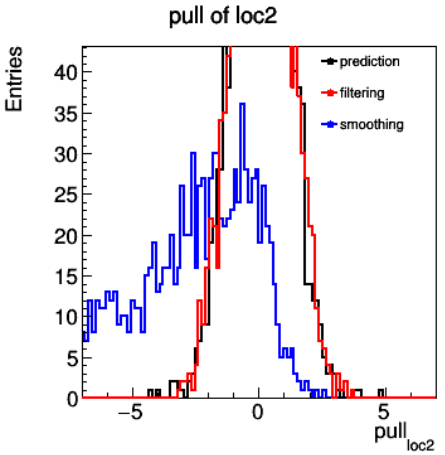


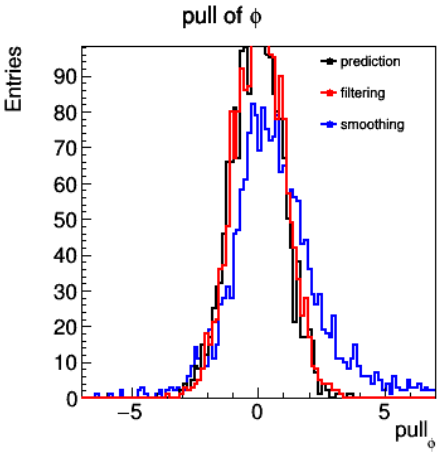
Bound Parameters at volume8-layer2

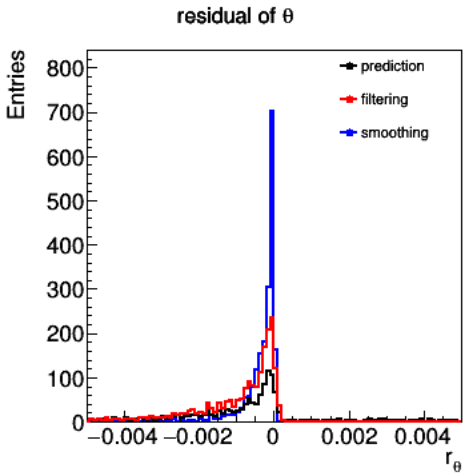
No non-linearity correction

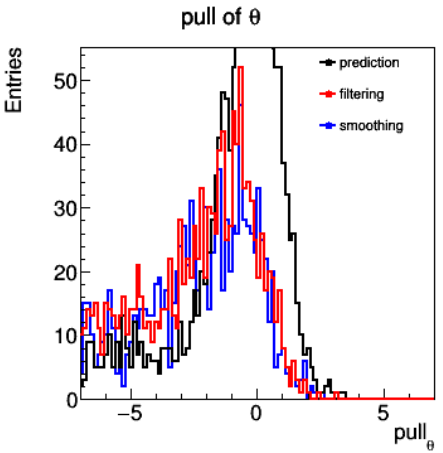


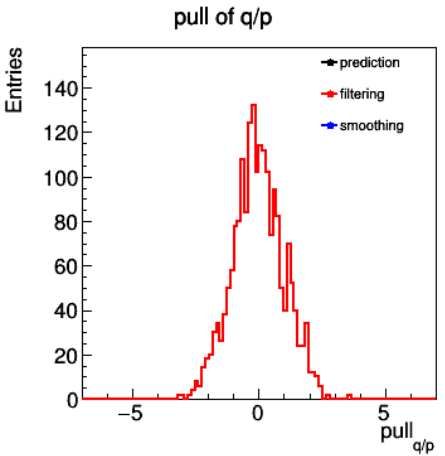


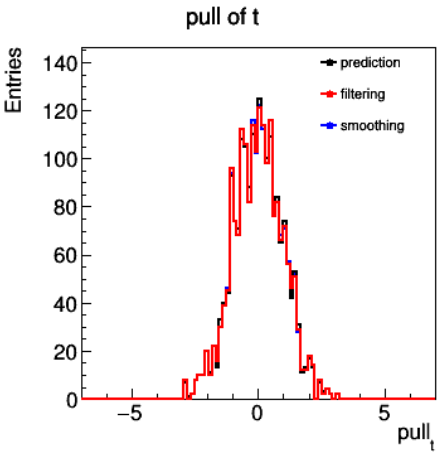






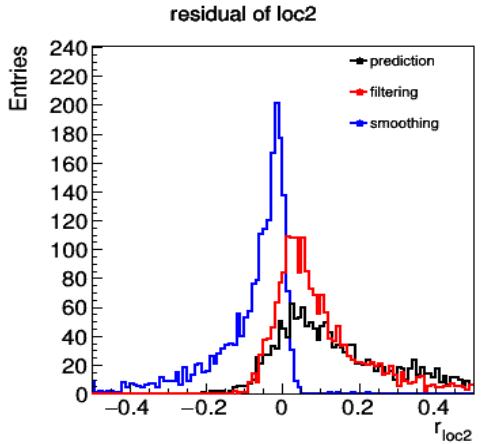


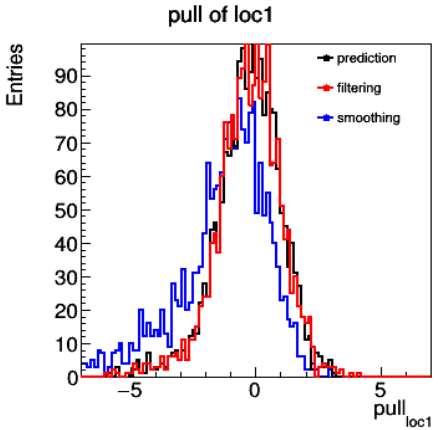


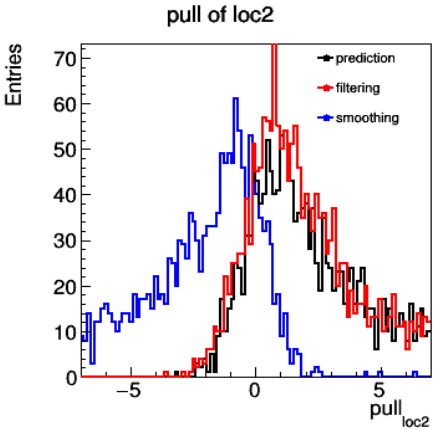


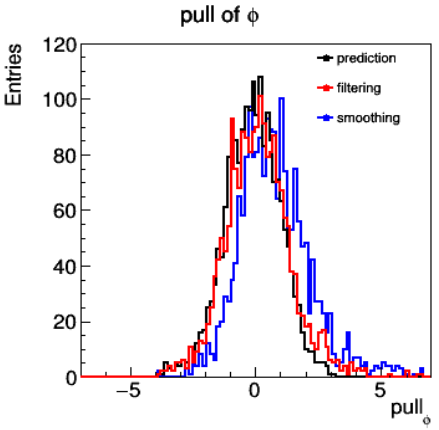
Bound Parameters at volume8-layer4

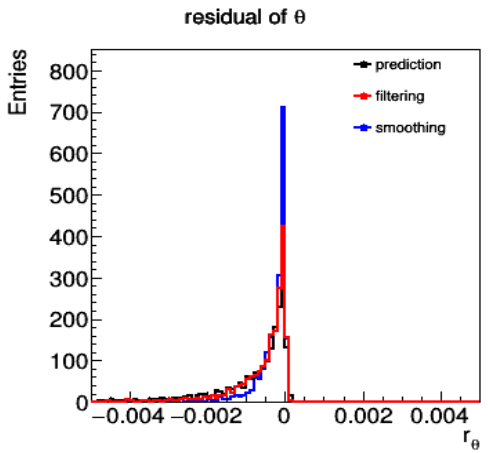
No non-linearity correction

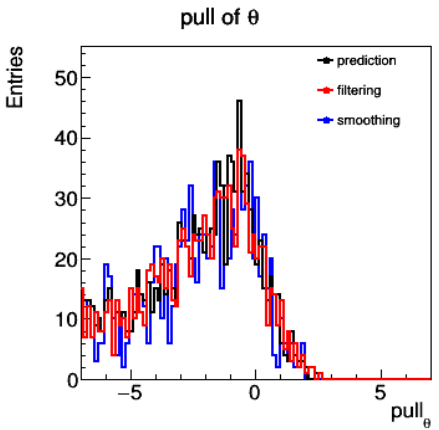


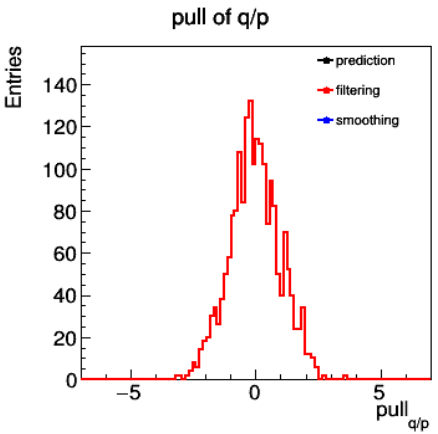


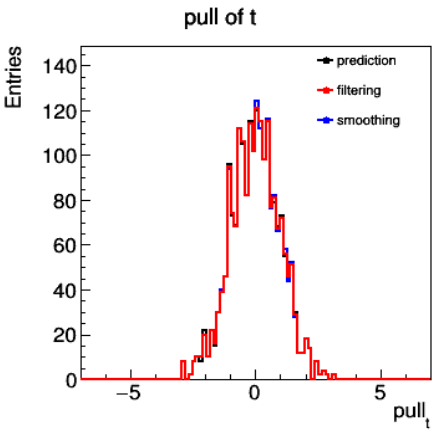






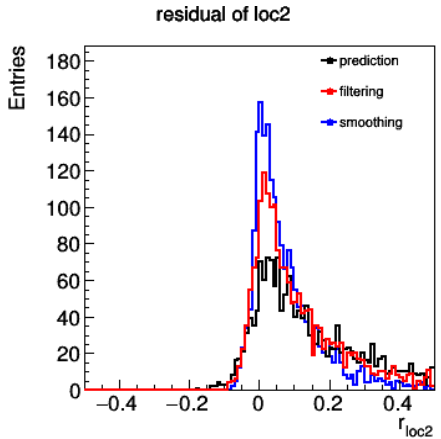


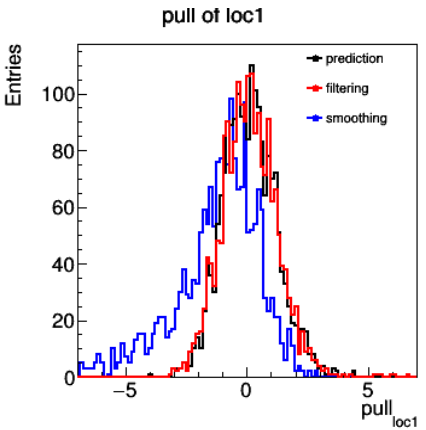


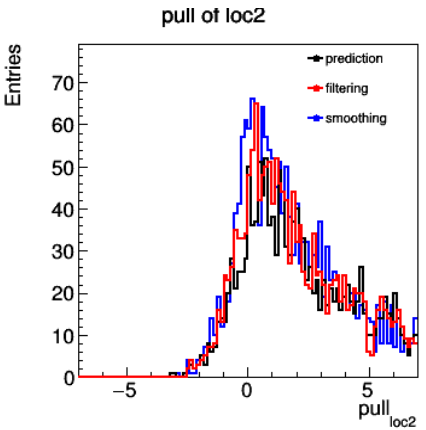


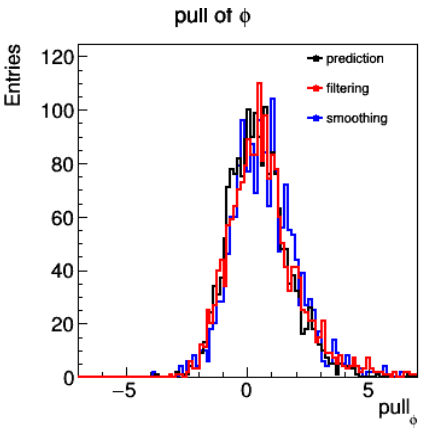
Bound Parameters at volume8-layer6

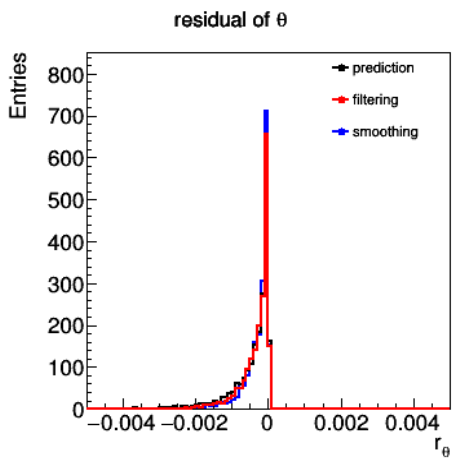
No non-linearity correction

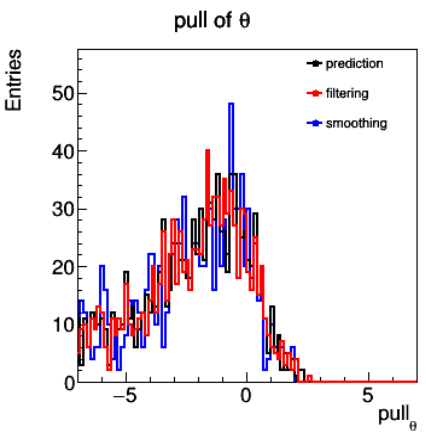


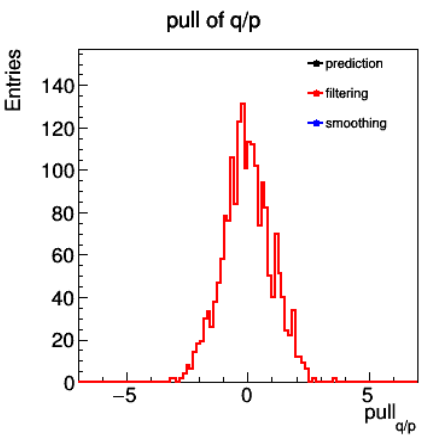


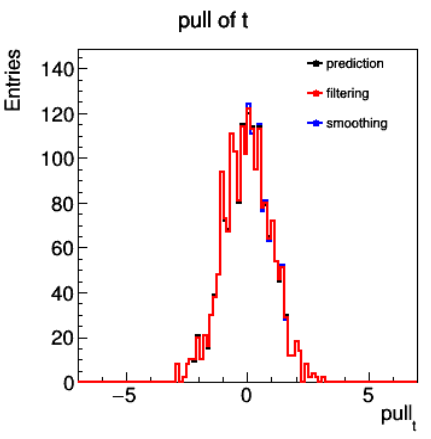






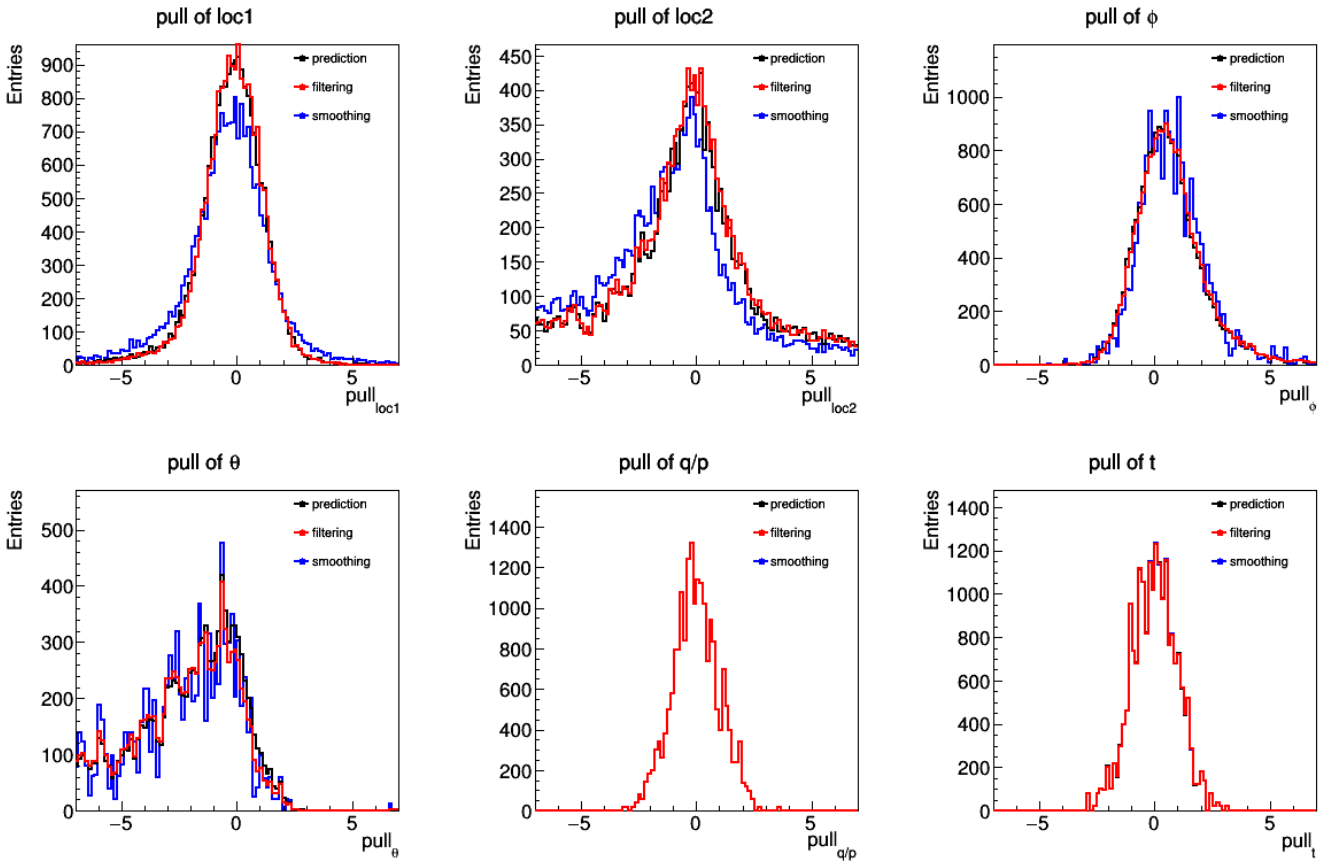






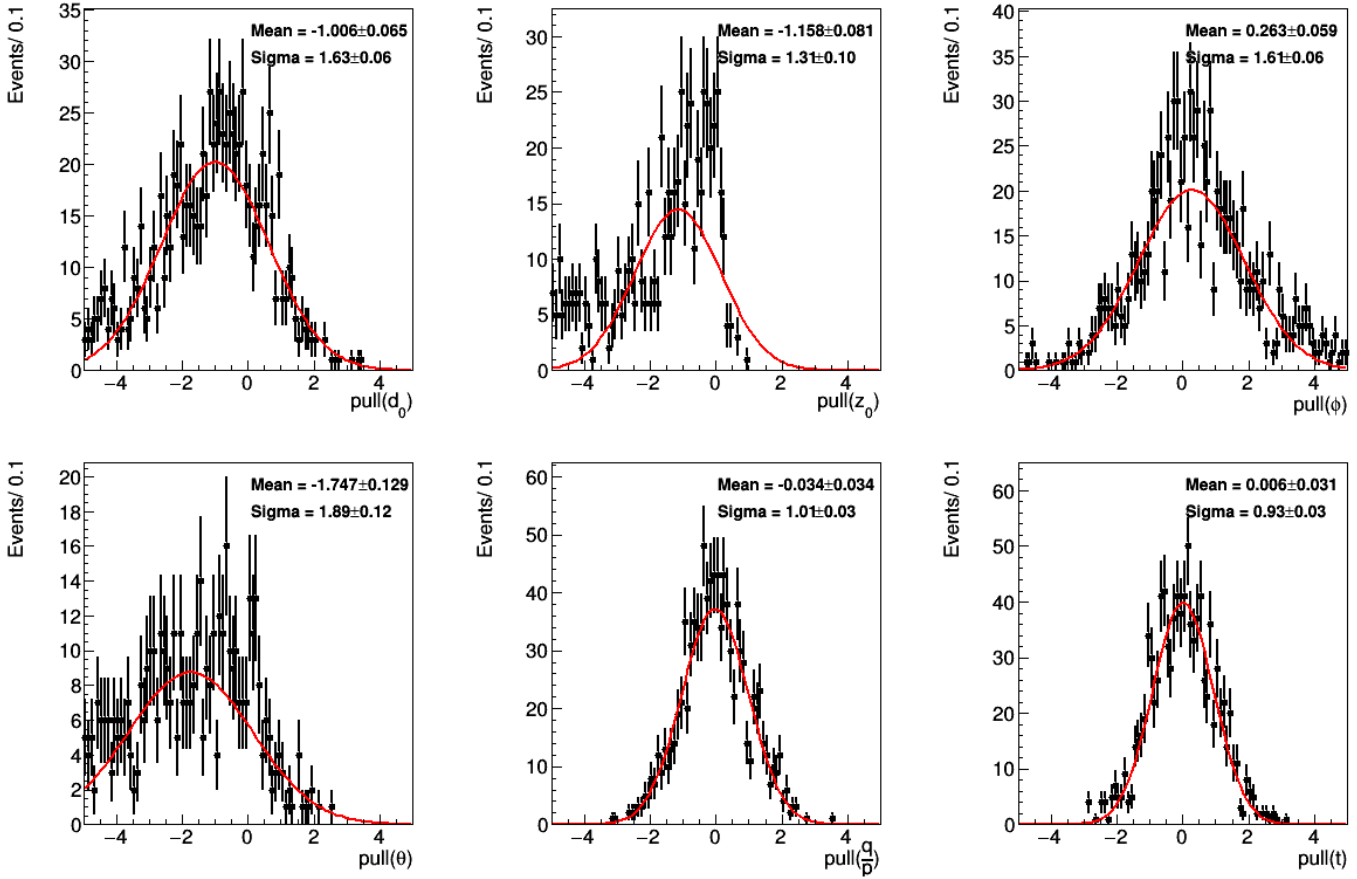
Bound Parameters at all surfaces

No non-linearity correction



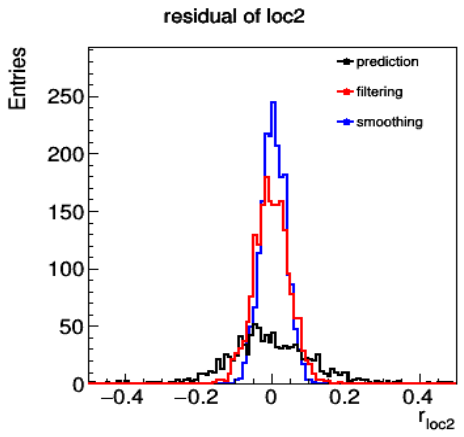
Fitted perigee parameters

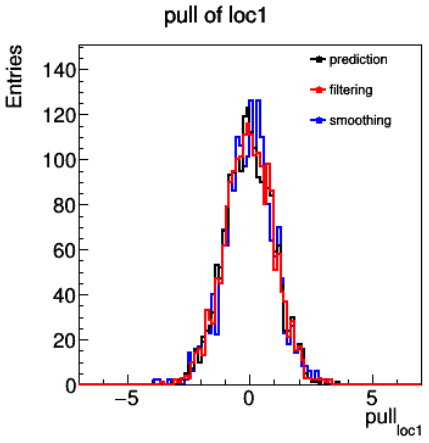
No non-linearity correction

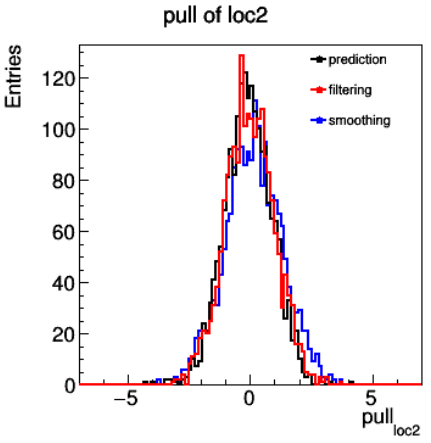


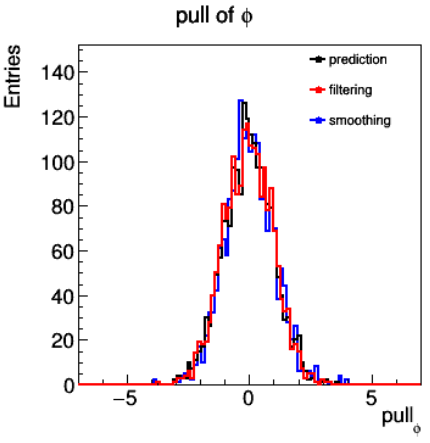
Bound Parameters at volume8-layer2

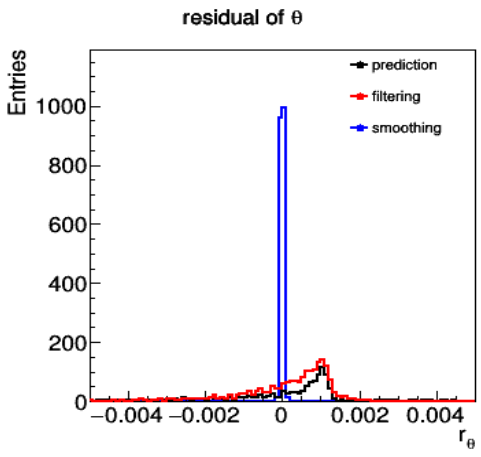
With non-linearity correction

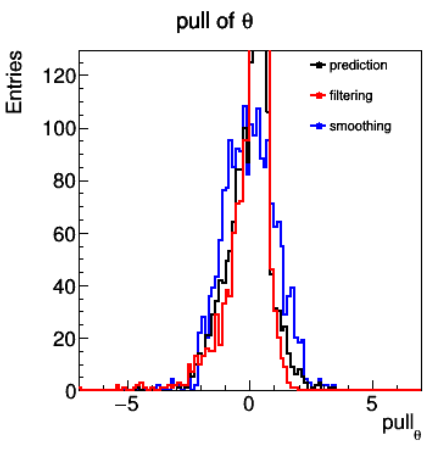


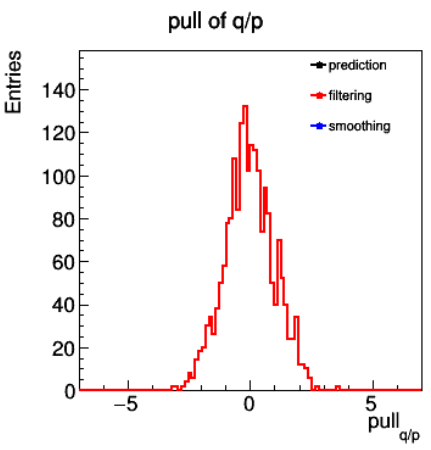


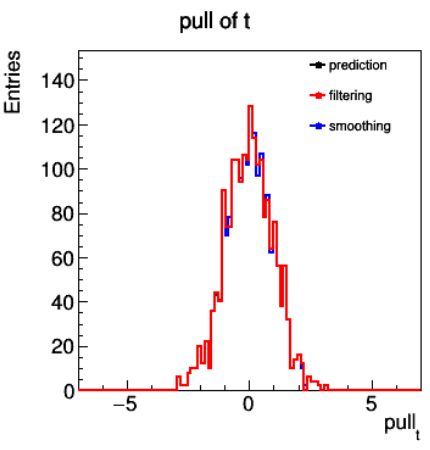






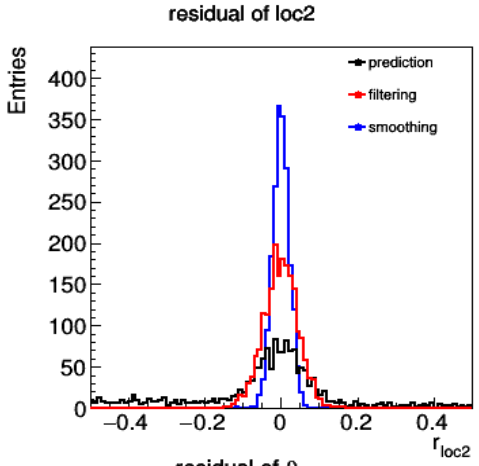


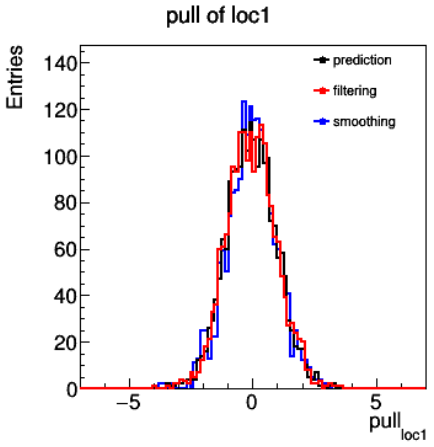


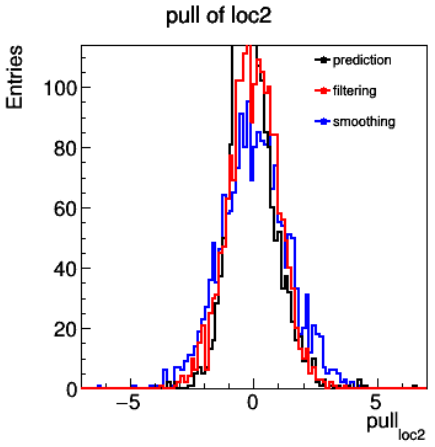


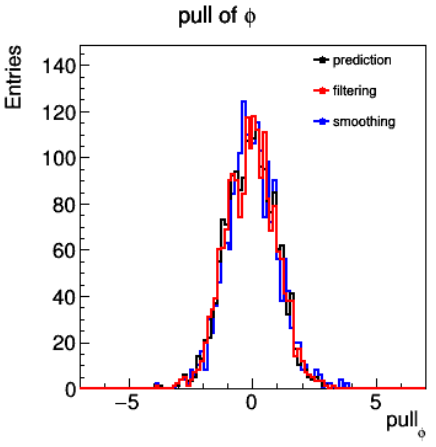
Bound Parameters at volume8-layer4

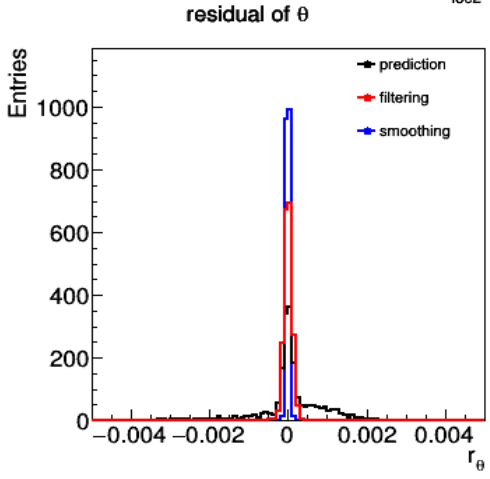
With non-linearity correction

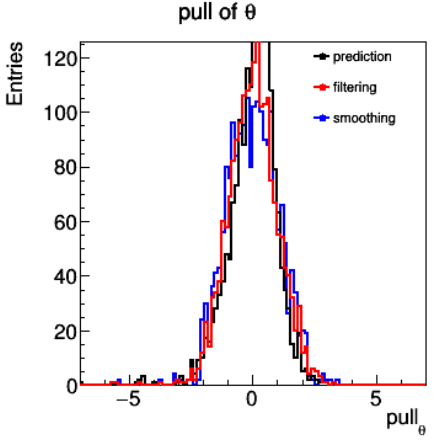


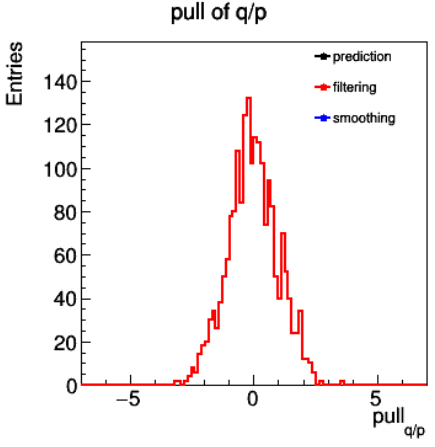


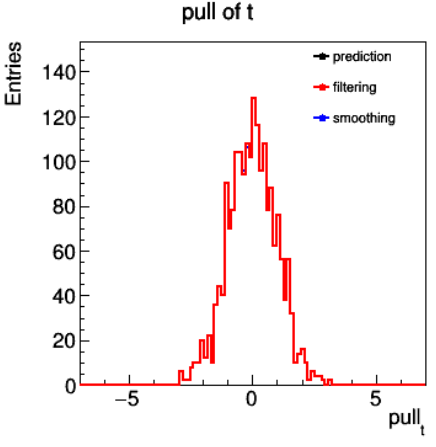






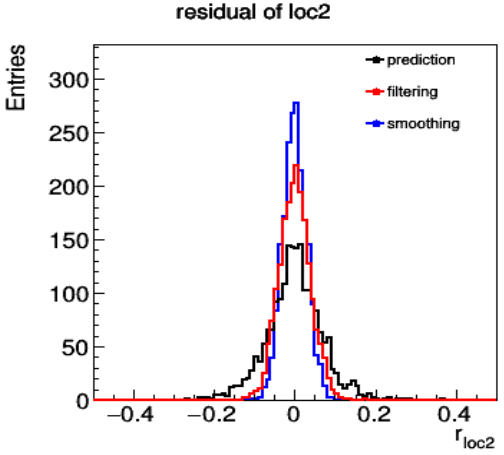


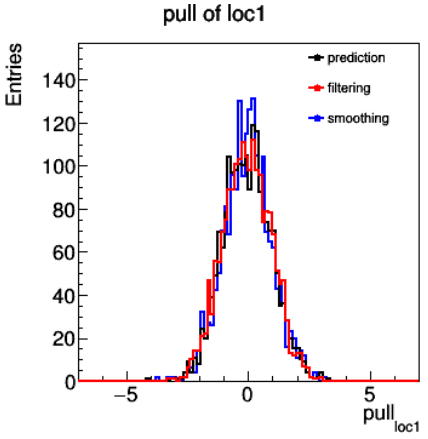


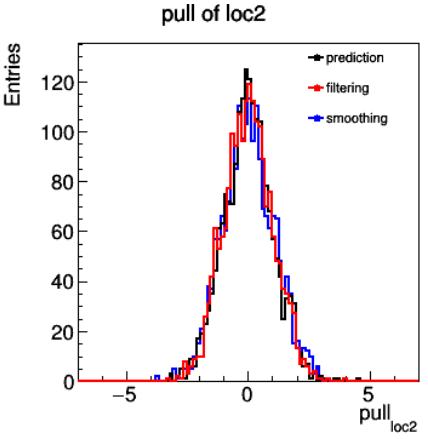


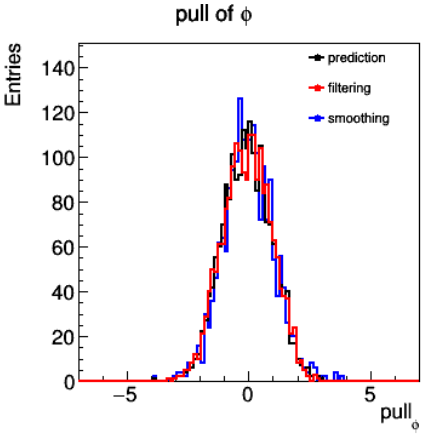
Bound Parameters at volume8-layer6

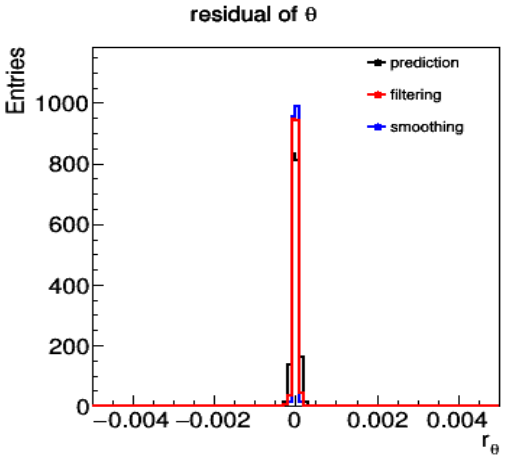
With non-linearity correction

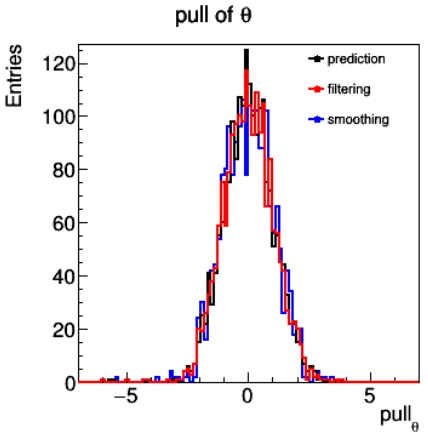


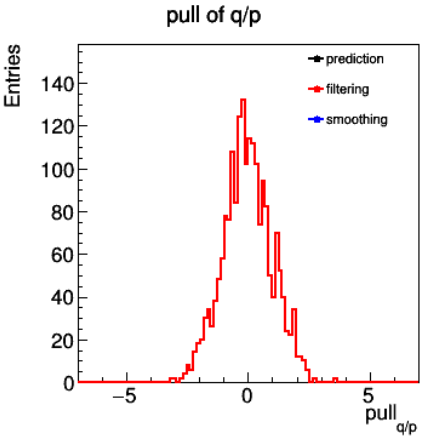


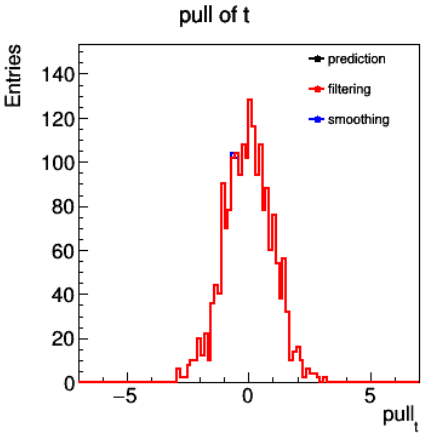






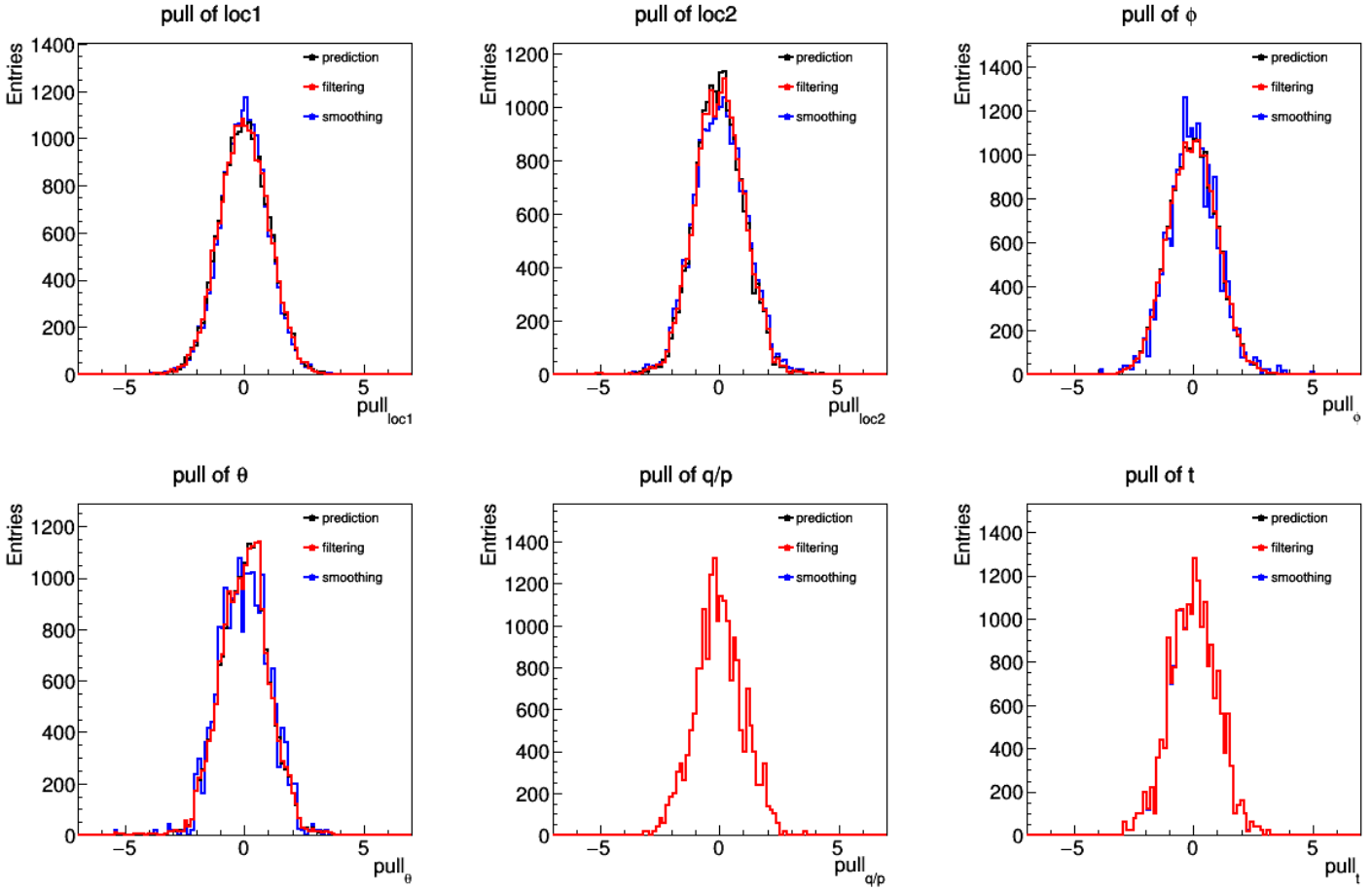






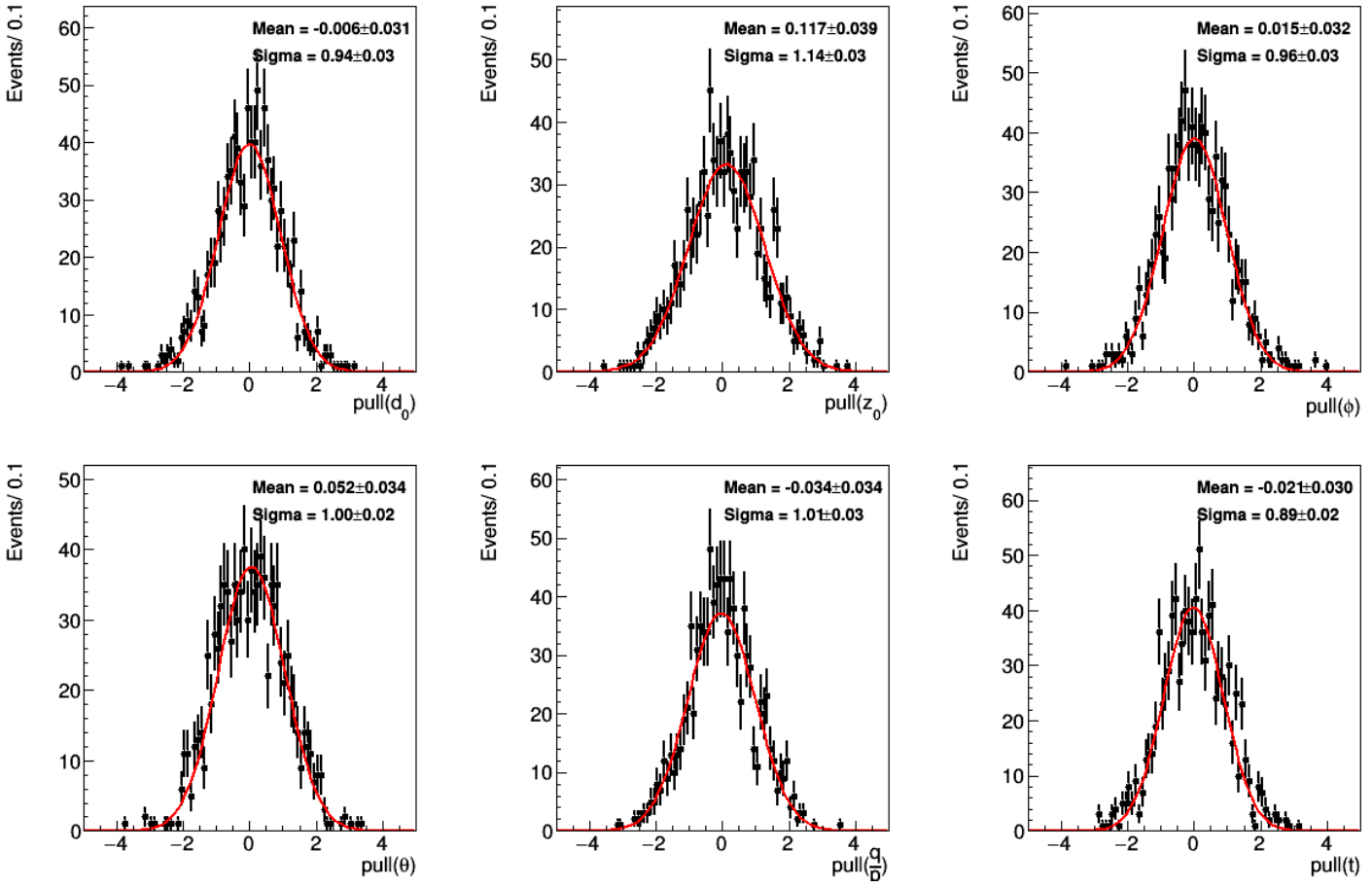
Bound Parameters at all surfaces

With non-linearity correction



Fitted perigee parameters

With non-linearity correction



Next-to-do

- Test with material effects and magnetic field
- Test lots of material + very good measurement precision
  - The non-linearity becomes comparable to the measurement error and non-negligible
  - Track parameters error decreases more rapidly than its actual error
  - Already observed issue with BGV detector
- Time test