Some ideas towards a 4D vertexing in Billoir and KF formalism

Valentina Cairo, PF, Ariel Schwartzmann, Lorenzo Santi.

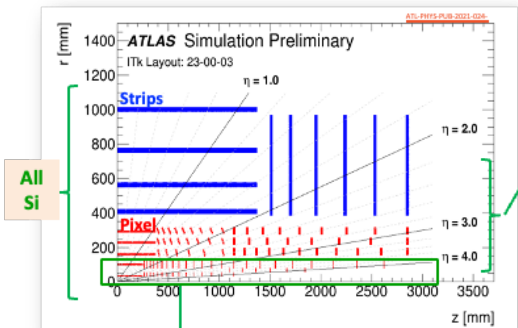
4th April 2023

Alignment Dev Meeting

Introduction

- Ongoing project in ATLAS since ~2021

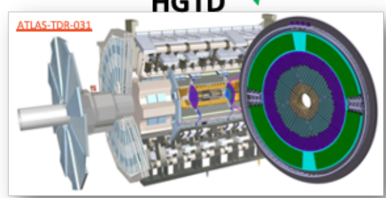
ITk



Extractable & Replaceable
(Radiation hardness up to ~10-15 MGy)

Increased η coverage (from 2.4 to 4.0)

HGTD



Extractable & Replaceable
(Radiation hardness up to ~2 MGy)

Average time resolution per hit (start and end of operational lifetime) $2.4 < \|\eta\| < 4.0$	≈ 35 ps (start), ≈ 70 ps (end)
Average time resolution per track (start and end of operational lifetime)	≈ 30 ps (start), ≈ 50 ps (end)

The question we are addressing:

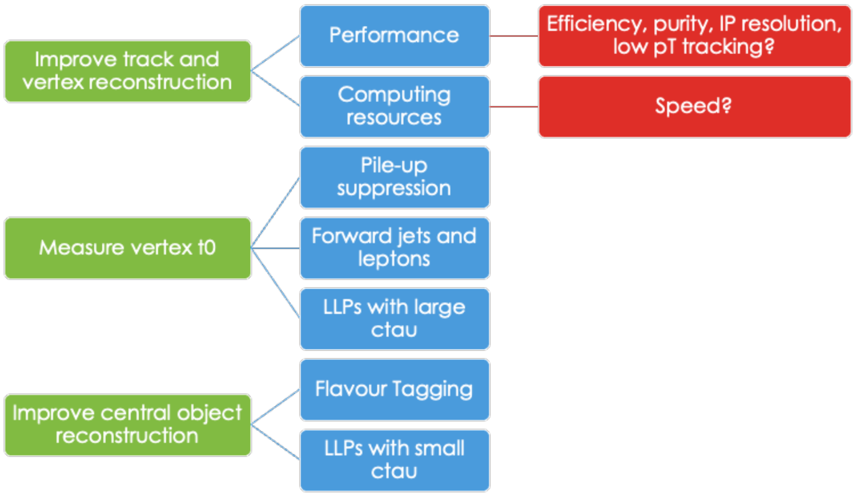
Can we maximize the ATLAS physics potential beyond Run 4 by extending the timing coverage to the full η acceptance?

Features (Order-of-magnitude):

Ultra-fast timing resolution:
O(10) ps
Precise longitudinal information:
O(10) μm

Introduction

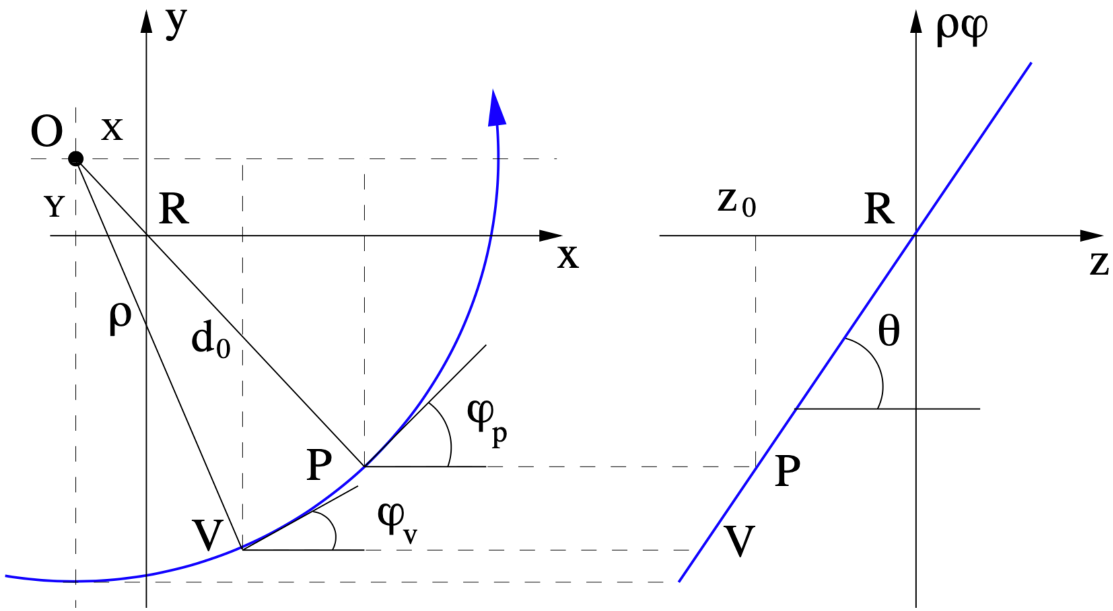
- More details in dedicated Upgrade Physics agenda in January
- Updates this coming Thursday
- Impact in ATLAS spans over several aspects
  ——————————>



- State of the art:
  - **Vertex t0 resolution** has been demonstrated
  - Impact on **Flavour Tagging** has been assessed
    - Both aspects are being extended to the ACTS realm and will include also more in-depth Tracking & Vertexing studies

      Today's talk!
  - Potential for big signal acceptance increase in **delayed photon analysis** also demonstrated
  - Dedicated work for **pile-up rejection** has started

Refresher - Track parameters in the perigee representation

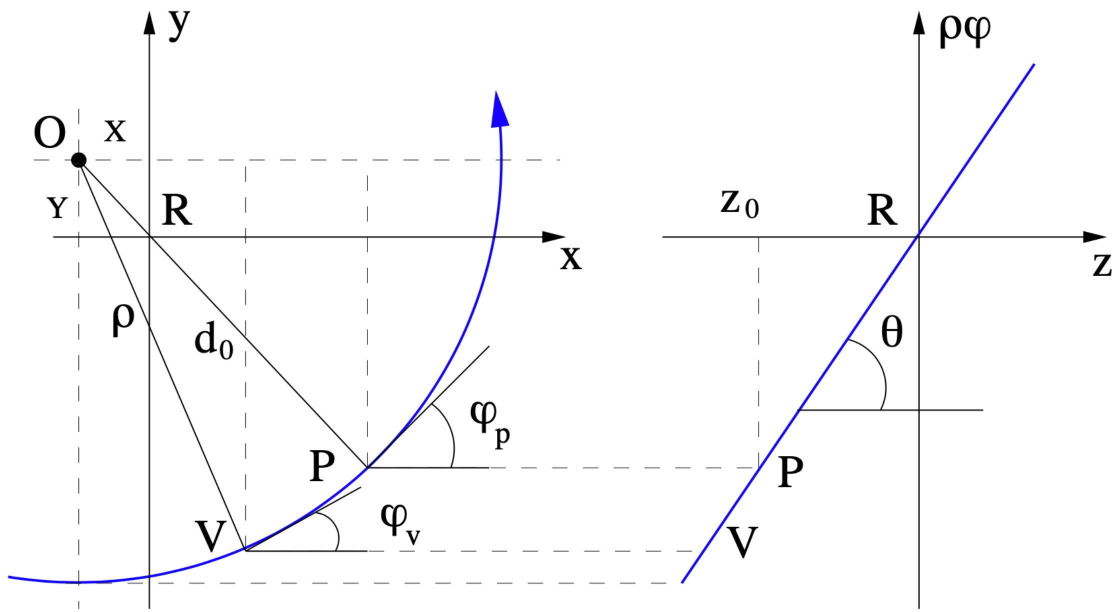


$$\rho = \frac{sin(\theta)}{\frac{q}{p}B_z}.$$

- Track parameters at the point of closest approach P to a reference point R
- d0: signed transverse IP
- z0: longitudinal IP
- ϕ_P: azimuthal angle of trajectory at P
- θ_P: polar angle of trajectory at P
- q/p: ratio of charge over momentum magnitude -> curvature

Equations for a generic point on the trajectory

- A generic point on the trajectory V can be expressed as a function of the reference point coordinates and the perigee track parameters



$$
\begin{aligned}
x_V &= x_R + d_0 \cos\left(\phi_p + \frac{\pi}{2}\right) + \rho\left[\cos\left(\phi_V + \frac{\pi}{2}\right) - \cos\left(\phi_p + \frac{\pi}{2}\right)\right] \\
y_V &= y_R + d_0 \sin\left(\phi_p + \frac{\pi}{2}\right) + \rho\left[\sin\left(\phi_V + \frac{\pi}{2}\right) - \sin\left(\phi_p + \frac{\pi}{2}\right)\right] \\
z_V &= z_R + z_0 - \frac{\rho}{tan(\theta)}\left[\phi_V - \phi_p\right]
\end{aligned}
\tag{5.32}
$$

Vertexing problem

- Vertex fit: find the "intersection" of a set of N tracks
- Actually, the vertex fit doesn't care if the tracks intersect or not:
  - Find the location in space "closest" to a set of N tracks
- All the 3D fitting was nicely implemented by Bastian in ACTS.

- How to incorporate time in this fit?
- Current track representation limitation (I'll use ACTS as example)
  - ACTS tracks have 6 parameters but only a "3D" surface representation, i.e. the perigee representation is determined by a 3D-vector, i.e. a line. No time reference, time is always wrt a global time at 0.
  - Propagator only to 3D-surfaces. Therefore:
    - Time propagation is just : $\Delta_t = s/\beta$, where s is the arc-length from A to B, and β is the velocity. If we would propagate to a 4D point (like a surface with a time measurement or a 4D vertex location), we need to subtract the time of the reference and obtain a t_0 time relative to the reference (same as previous point)
    - Do we need to define a "point of closest approach" in 4d-space? What metric, euclidean?
      - Let's try a simpler solution first

4D reference point

- Our idea to extrapolate to a “real” 4D point:
  - Go to the Point of closest approach (PCA) **spatially** and compute Δ_t
  - Global track time is $t_0 + \Delta_t + t_R$, where t_R is the 4D track reference time, i.e. if $t_R = 0$ (as it is currently), then t0 is the global track fit time at the reference point.
  - If we would keep the spatial reference point, but just change time component, then $t_0' = t_0 + t_R - t_R^{'}$,
- Defining the PCA spatially is justified if time-resolution is large compared to transverse IP resolution
  - 1ns for a particle at $\beta \sim 1$ is ~300mm
  - Current resolution in the transverse plane is O(10um), means that we would need O(0.1ps - 0.03mm) time resolution to match spatial
    - **We can avoid worrying of space-time PCA and propagate spatial PCA and compute Δt (right?)**

Ingredients

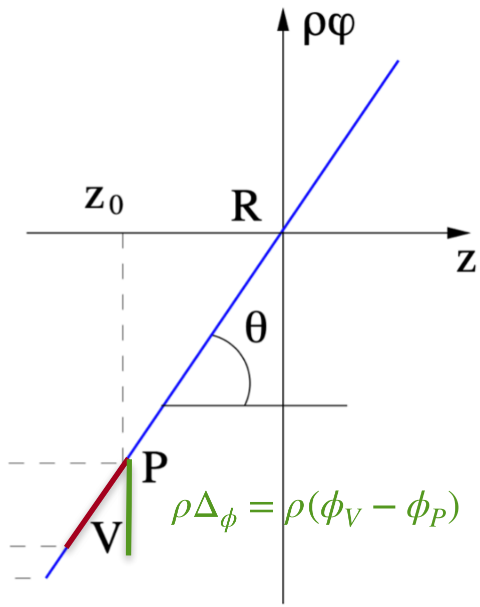
- Two type of vertexing:
  - **Simple** (derivatives track parameters wrt vertex position) and **Full** (derivatives track parameters wrt 3-momentum).
- Today I'll discuss only **simple** vertex fitting
  - Only care about the position Jacobian $A = \dfrac{\partial(d_0, z_0, \phi_p, \theta, q/p, t_0)}{\partial(x_V, y_V, z_V, t_V)}$
- We basically need to invert the equations shown at slide 3 and have the track parameters as function of the vertex (new reference) position

$$
\begin{aligned}
d_0 &= \rho + \mathbf{sgn}(d_0 - \rho)\sqrt{\left(x_V - x_R - \rho\cos\left(\phi_V + \frac{\pi}{2}\right)\right) + \left(y_V - y_R - \rho\sin\left(\phi_V + \frac{\pi}{2}\right)\right)} \\
\phi_P &= \mathtt{arctan}\left(\frac{y_V - y_R - \rho\sin\left(\phi_V + \frac{\pi}{2}\right)}{x_V - x_R - \rho\cos\left(\phi_V + \frac{\pi}{2}\right)}\right) \qquad (5.33) \\
z_0 &= z_R + z_V + \frac{\rho}{tan(\theta)}\left[\phi_V - \phi_p\left(x_V, y_V, \phi_V, \theta, q/p\right)\right] \\
\left(\frac{q}{p}\right)_P &= \left(\frac{q}{p}\right)_V \\
\theta_P &= \theta_V
\end{aligned}
$$

4D-Vertexing - Ingredients

- What about time?
  - “Approximated” and 4d-fit
  - We extend the extrapolation to the beam line with an extrapolation to a **beam-plane** where the plane is z-t
  - The time of a track with respect to the 4D reference point R, will be the t-coordinate of the PCA in the bending plane (similar to z0)
- **4d fit**
  - $t_0^V = t_0 + t_R + \text{sgnt} \cdot s(x_V, y_V, z_V)/\beta = t_0 + t_R + \dfrac{\rho\Delta_\phi}{\beta \sin\theta}$
    - The sgnt is a sign which depends if the propagation to the vertex location is forward (-) or backward (+) wrt the current track 3D reference point

      [Really need to crosscheck this sign when going in the $\rho\Delta_\phi$ plane]
- **Approximated 4D-fit**
  - **Neglect arc-length effect on time $\Delta_t = s/\beta \to 0$**
    - No dependence on the vertex time from the vertex location => **we expect that the vertex time is just the weighted mean of the track time**
    - Nothing new, just inserted in a “fit formalism”



Approximated simple 4D Vertexing

- $\partial t_0 / \partial t_V = 1,\ \partial t_0 / \partial k_V = 0,$ with k=x,y,z
- That's the last row of the position jacobian:

$$A = \frac{\partial(d_0, z_0, \phi_P, \theta_P, q/p)}{\partial(x_V, y_V, z_V)} = \begin{pmatrix} -h\frac{X}{S} & -h\frac{Y}{S} & 0 \\ \frac{\rho}{\tan\theta}\frac{Y}{S^2} & -\frac{\rho}{\tan\theta}\frac{X}{S^2} & 1 \\ -\frac{Y}{S^2} & \frac{X}{S^2} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \longrightarrow \quad A_{4D}^{approx} = \begin{pmatrix} -h\frac{X}{S} & -h\frac{Y}{S} & 0 & 0 \\ \frac{\rho}{\tan\theta}\frac{Y}{S^2} & -\frac{\rho}{\tan\theta}\frac{X}{S^2} & 1 & 0 \\ -\frac{Y}{S^2} & \frac{X}{S^2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- We expect just the weighted average, no effect on vertex location
- Advantage:
  - It's inserted directly the Billoir / KF vtx-fit formalism.

Approximated simple 4D Vertexing

- Implemented logic in ACTS and tested using the Billoir unit test
  - We generated 4D vertex positions, space and time
  - We generate N tracks around those vertex locations with different resolutions of track parameters and diagonal covariance matrix
    - For simplicity I fix the time resolution to 100ps for all tracks (so time should be the simple average)
  - CHECK
    - NOMINAL: Is the original 3D vertex fit without time
    - APPROXIMATED SIMPLE: is the 4D vertex fit in the approximated case $\Delta_t \to 0$
- **The vertex location is independent on the time fit [very small 4th significant digit corrections]**
- **The vertex time is the average of the track time [expected]**

NOMINAL

```
True Vertex: 0.0350348, 0.0941408, 15.3922
Fitted Vertex: -0.0607083
-0.0778377
  15.3543
```

```
Fitting nTracks: 7
True Vertex: 0.00193901, 0.0908309, -11.8037
Fitted Vertex:   0.111397
-0.0203807
  -11.7717
```

```
Fitting nTracks: 3
True Vertex: 0.0660289, -0.0190278, -2.10198
Fitted Vertex:  -0.046738
-0.0749018
  -2.11771
```

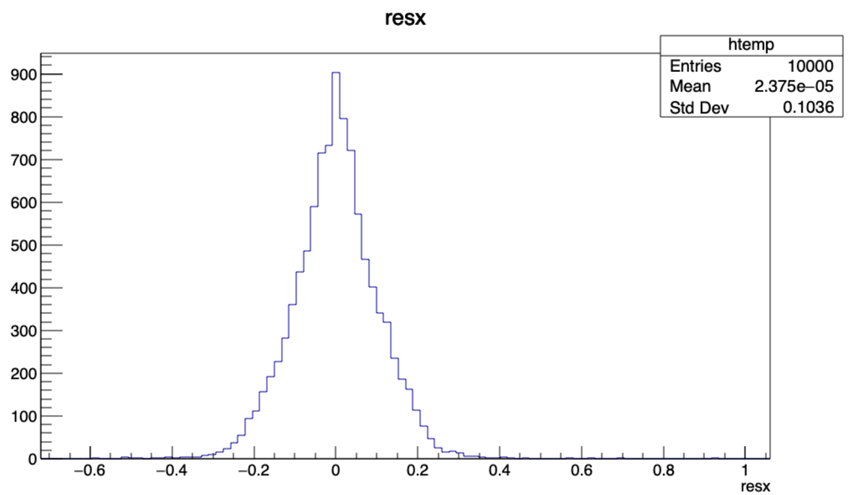
APPROXIMATED SIMPLE

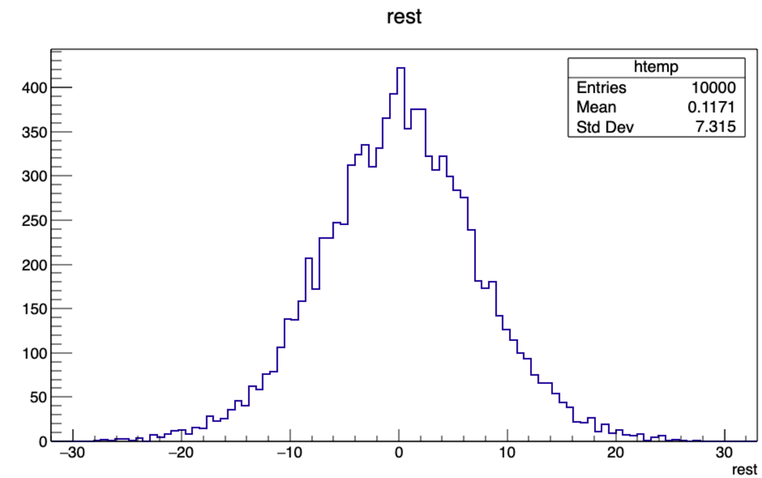
```
True Vertex: 0.0350348, 0.0941408, 15.3922, -76.2096
Fitted Vertex 4Pos: -0.0607083 -0.0778377   15.3543   -79.8125
```

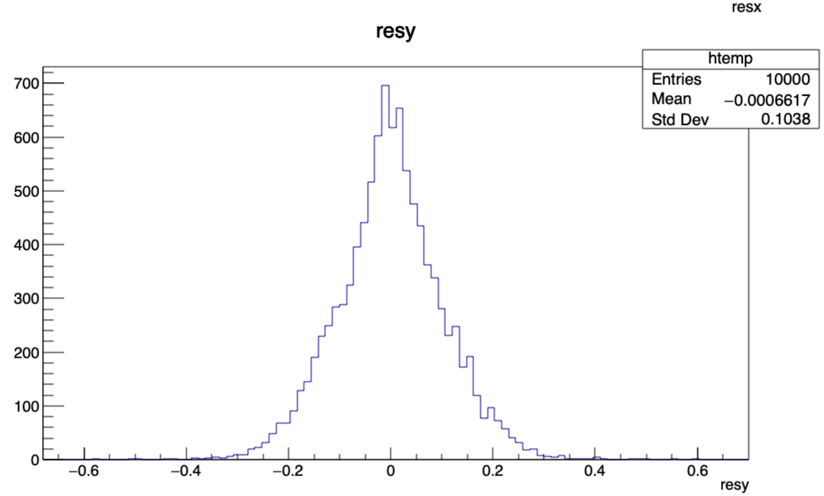
```
True Vertex: 0.00193901, 0.0908309, -11.8037, -79.8651
Fitted Vertex 4Pos:   0.111817
-0.0209129
  -11.7719
  -98.0354
```

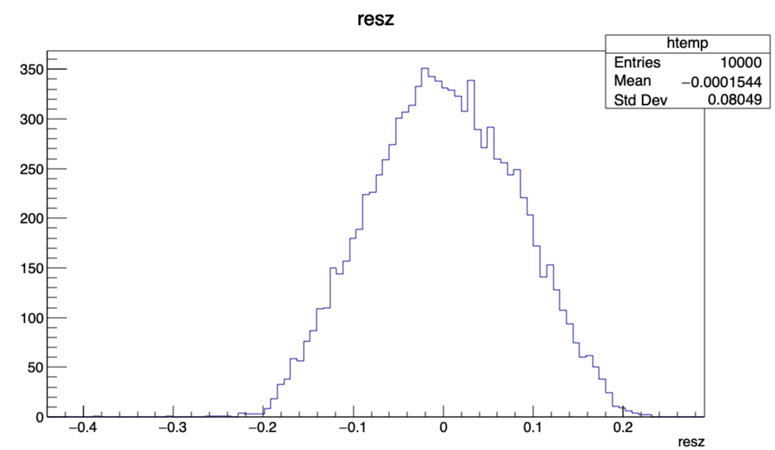
```
True Vertex: 0.0660289, -0.0190278, -2.10198, 123.407
Fitted Vertex 4Pos: -0.0467366
-0.0749053
  -2.11771
  125.604
```

Approximated simple 4D Vertexing - UnitTest









- X-Y-Z / T correlations at 0

Simple 4D vertex

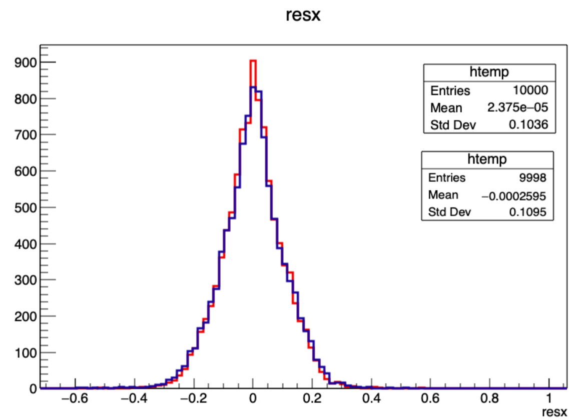
- In the simple 4D vertex fit we assume $\dfrac{\partial(d_0, z_0, \phi_P, \theta_P, q/p)}{\partial(\phi_V, \theta_V, q/p)} = 0$
- We just need to compute the $\dfrac{\partial(t_0)}{\partial(x_V, y_V, z_V, t_V)}$
- Remind: $t_0 = t_V - t_R - \dfrac{\rho\Delta_\phi}{\beta \sin\theta}$ is very similar to $z_0 = z_V - z_R + \dfrac{\rho\Delta_\phi}{\tan\theta}$ and the derivatives are already computed! So:
- $\dfrac{\partial t_0}{\partial t_V} = 1$
- $\dfrac{\partial t_0}{\partial x_V} = -\dfrac{1}{\beta\cos\theta}\dfrac{\partial z0}{\partial x_V} \quad \dfrac{\partial t_0}{\partial y_V} = -\dfrac{1}{\beta\cos\theta}\dfrac{\partial z0}{\partial y_V} \quad \dfrac{\partial t_0}{\partial z_V} = 0$
- The full Jacobian for the Simple 4D Vertex fit becomes (modulo wrong sign in last row first 2 elements)

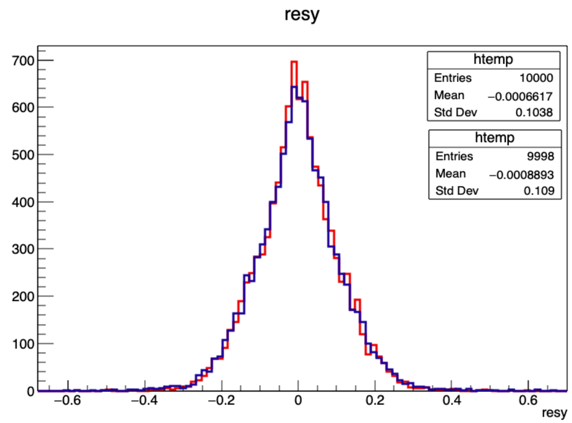
$$A = \frac{\partial(d_0, z_0, \phi_P, \theta_P, q/p)}{\partial(x_V, y_V, z_V)} = \begin{pmatrix} -h\frac{X}{S} & -h\frac{Y}{S} & 0 \\ \frac{\rho}{\tan\theta}\frac{Y}{S^2} & -\frac{\rho}{\tan\theta}\frac{X}{S^2} & 1 \\ -\frac{Y}{S^2} & \frac{X}{S^2} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

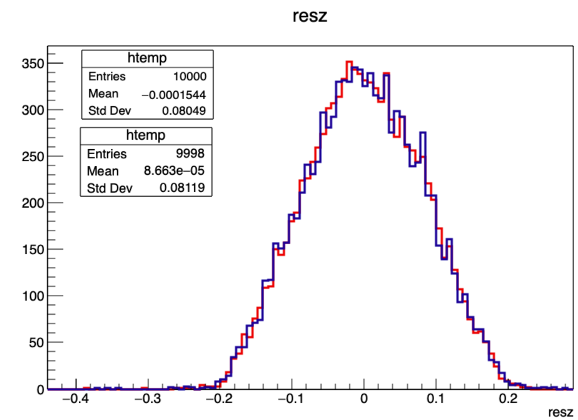
$$\boxed{A_{4D} = \begin{pmatrix} -h\frac{X}{S} & -h\frac{Y}{S} & 0 & 0 \\ \frac{\rho}{\tan\theta}\frac{Y}{S^2} & -\frac{\rho}{\tan\theta}\frac{X}{S^2} & 1 & 0 \\ -\frac{Y}{S^2} & \frac{X}{S^2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{\rho}{\beta\sin\theta}\frac{Y}{S^2} & +\frac{\rho}{\beta\sin\theta}\frac{X}{S^2} & 0 & 1 \end{pmatrix}}$$

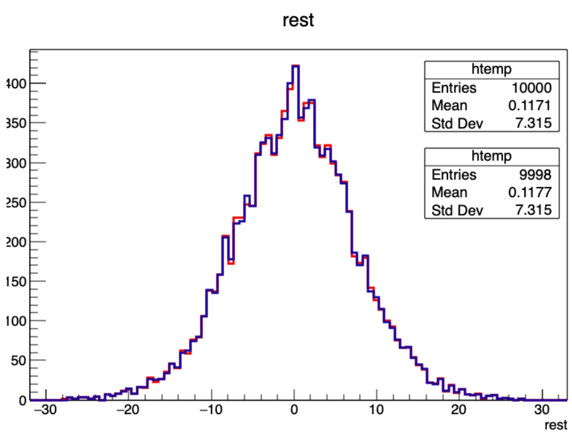
Simple 4D vertex

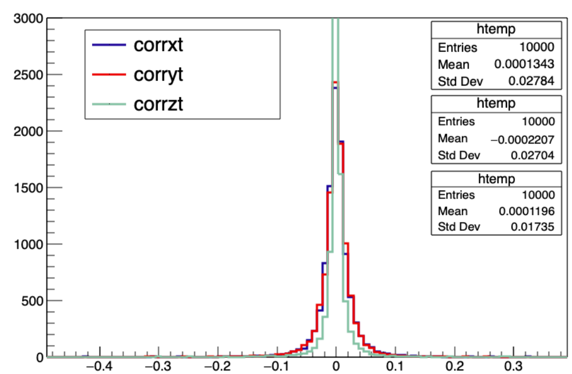
- Tested with a sigma_t ~ 1mm











Some words about the implementation

- Added time information to the FullBilloirVertexFitter.ipp
- Added proper reference to t0 for each of Billoir fit iteration to keep track of the deltaV correction to the reference point
- Assumed pion mass for the beta computation (doesn’t really matter)
- To be fixed:
  - Weight matrix doesn’t have to be cast to a 5x5 matrix in the case of time fit.
  - Currently weight matrix with time is wrong => Fixed after the meeting
- These changes are necessary even if a different Linearizer is used
  - Such as a numerical linearizer.

- Additionally if we want a 6D track, we should think about a 4D reference point (in ou opinion)

Derivation of the full jacobians

Here is the derivation of the jacobian terms.

$$\frac{\partial t_0}{\partial x_V} = -\left[\frac{\rho}{\beta}\frac{Y}{S^2 \sin\theta}\right]$$

$$\frac{\partial t_0}{\partial y_V} = -\left[-\frac{\rho}{\beta}\frac{X}{S^2 \sin\theta}\right]$$

$$\frac{\partial t_0}{\partial z_V} = 0$$

$$\frac{\partial t_0}{\partial t_V} = 1$$

$$\frac{\partial t_0}{\partial \phi_V} = -\left[\frac{\rho}{\beta \sin\theta}\left(1 - \frac{\rho Q}{S^2}\right)\right]$$

$$\frac{\partial t_0}{\partial \theta_V} = -\left[\frac{\rho}{\beta}\left(\Delta\phi + \frac{\rho R}{S^2 \tan^2\theta} + \frac{\Delta\phi}{\cos\theta}\right)\right]$$

$$\frac{\partial t_0}{\partial (q/p)} = -\left[\frac{\rho}{(q/p)\beta\cos\theta}\left(\Delta\phi - \frac{\rho R}{S^2}\right) + \frac{\rho\Delta\phi}{\beta(q/p)\sin\theta}(1-\beta^2)\right]$$

Summary and to-do

- The vertexing code in ACTS is really nicely developed and clear. Kudos++
- Last time we checked ACTS vertexing with time we found that such fit was not supported.
  - The developers solved the issue by removing time from the fit, basically. (which is fine)
- We tried to tackle the problem and tried to write the Jacobian matrix for the vertex fit
  - We need other experts to cross-check if it's correct
- We extended the ACTS unit-tests to check the basic case where we neglect the correction to the time of the vertex due to the extrapolation position
  - It gives the expected results and should be good enough for expected short-term time measurement time sensitivity (it's nothing fancy, just trivial case)
- We showed that the spatial derivatives are simple and similar to dz0/dV
- We finished the math and have a version for review after adding the dependence on the track momenta (Full4DBilloirVertexFit)

- We are testing this in a simple scenario (unit tests, for example)
  - In particular we want to check when time sensitivity starts to "play a role" on determining the vertex position (as function of N-Tracks and measurement resolution)