Tracking and Vertexing in 3D B-field

Norman Graf (SLAC)

HPS Collaboration Meeting, JLab
October 26, 2015

Track Extrapolation

- At the heart of both track and vertex fitting in the presence of a non-uniform magnetic field is the task of track extrapolation
  - Given a track representation at point a, how do we get to point b?
  - How do we know where point b is?
- In addition to trajectory propagation, we also need to account for physical effects which occur along the path, primarily
  - Scattering (affects track direction, account for in covariance matrix)
  - Energy loss (affects momentum (curvature), modify track state and covariance matrix)

Vertexing

- Requires well-fit tracks extrapolated to vertex position
  - Knowledge of fully-propagated covariance matrix is essential.
- Provides not only vertex position (x,y,z), but momentum as well (px,py,pz).
  - Imposing the vertex fit improves each track and also introduces a covariance term between that track and all others involved in the fit.
- Need to be able to impose constraints, e.g.
  - position (target and/or beam spot), direction (point back to beam-spot), mass (calibration with Mollers)

Z-Track Parameterization

- Specified by single parameter z, which is beam coordinate.
- Track Parameters: (x, y, dx/dz, dy/dz, q/p)
- Much more natural for HPS.
  - Position parameters (x,y) map directly onto tracker modules and Ecal.
  - Direction parameters (dx/dz, dy/dz) also map clearly on to HPS coordinate system.
  - Provides (inverse) momentum directly to the end user.
- Provides “natural” coordinates for vertexing
  - (x, y, z, px, py, pz)

Track Extrapolation

❖ x ❖ y	Position	Track Parameters
❖ tx (dx/dz) ❖ ty (dy/dz)	Direction	
❖ q/p	charge/momentum	
o (z)	reference location	

- z coordinate is a reference, other track parameters are fitted.
- Convenient for track extrapolation as target and detectors essentially referenced by z

Track Extrapolation

- Equation of motion for track parameters in magnetic field:

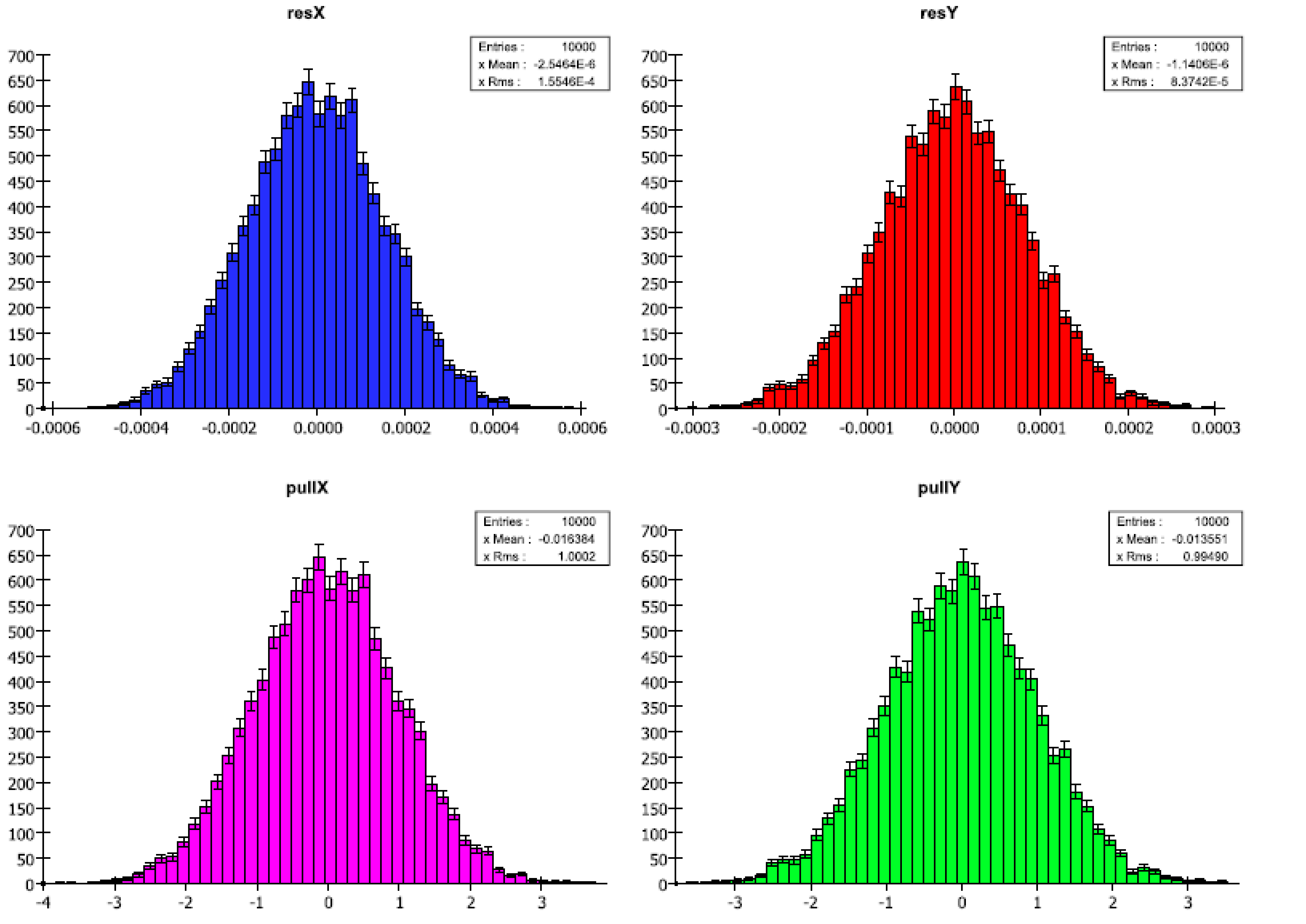
$$\frac{\mathrm{d}\mathbf{r}(z)}{\mathrm{d}z} = \begin{pmatrix} t_x \\ t_y \\ \kappa \cdot (q/p) \cdot \sqrt{1 + t_x^2 + t_y^2} \cdot \Big(t_x t_y \cdot B_x - (1 + t_x^2) \cdot B_y + t_y \cdot B_z \Big) \\ \kappa \cdot (q/p) \cdot \sqrt{1 + t_x^2 + t_y^2} \cdot \Big((1 + t_y^2) \cdot B_x - t_x t_y \cdot B_y - t_x \cdot B_z \Big) \\ 0 \end{pmatrix} (z)$$

- Solve this Cauchy problem using fourth-order Runge-Kutta (also used by GEANT)
  - Fourth-order method means that the precision of the method depends on the step size to the 5th power.
  - Runge-Kutta methods of any order exist but the fourth-order method is the optimal one with respect to CPU time consumption.

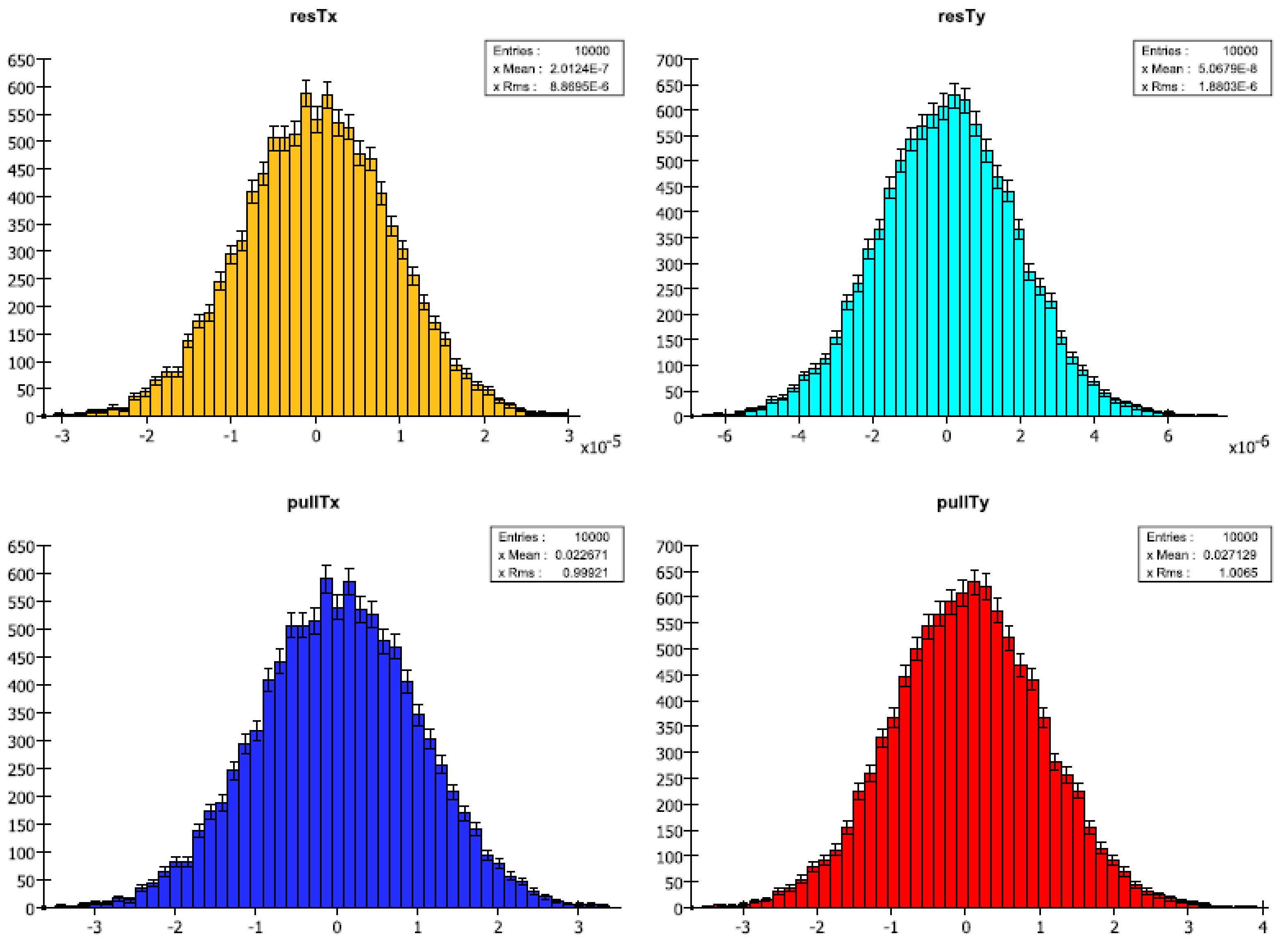
Track Extrapolation

- RK extrapolator implemented with a variable step size. Number of iterations is driven by the desired propagation precision.
- Test using HPS Phantom detector
  - Massless planes at fixed z
    - No scattering, no energy loss, no energy loss fluctuations
  - Smear SimTrackerHits with gaussian resolution
  - Fit resulting hits (no finding necessary)
  - Generate residuals, plot pulls
- Checks algorithm implementation in hps-java
  - slic propagates particles in field using GEANT (C++)
  - fitter propagates tracks in field using RK4 (Java)

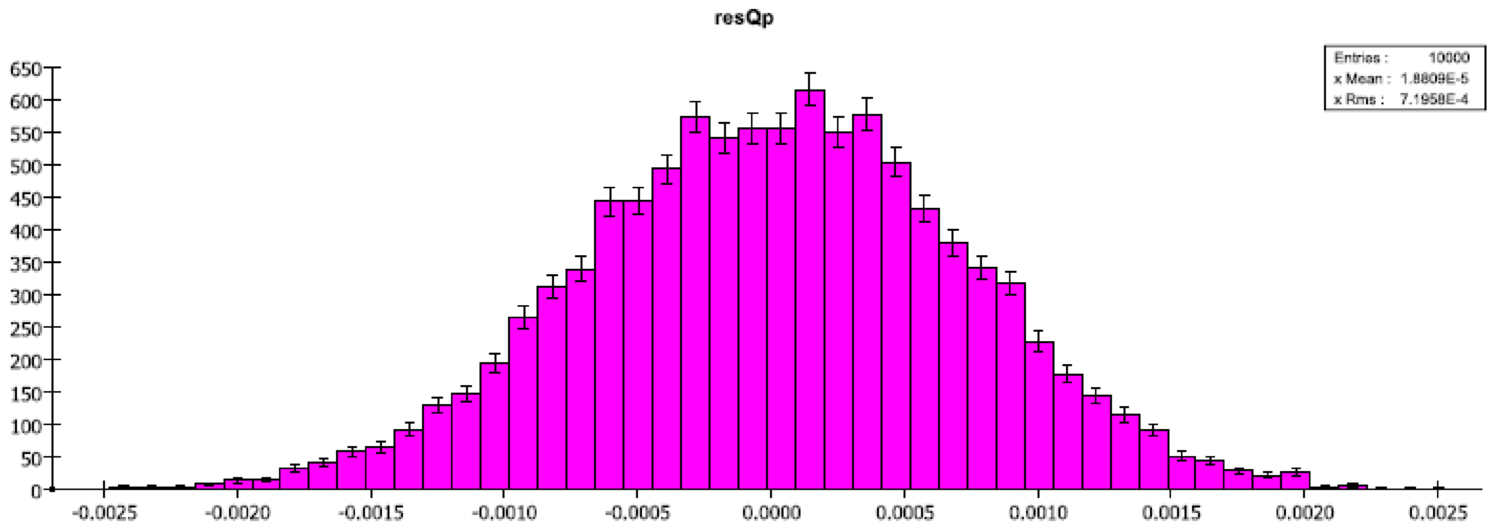
HPS Phantom Position Res & Pulls

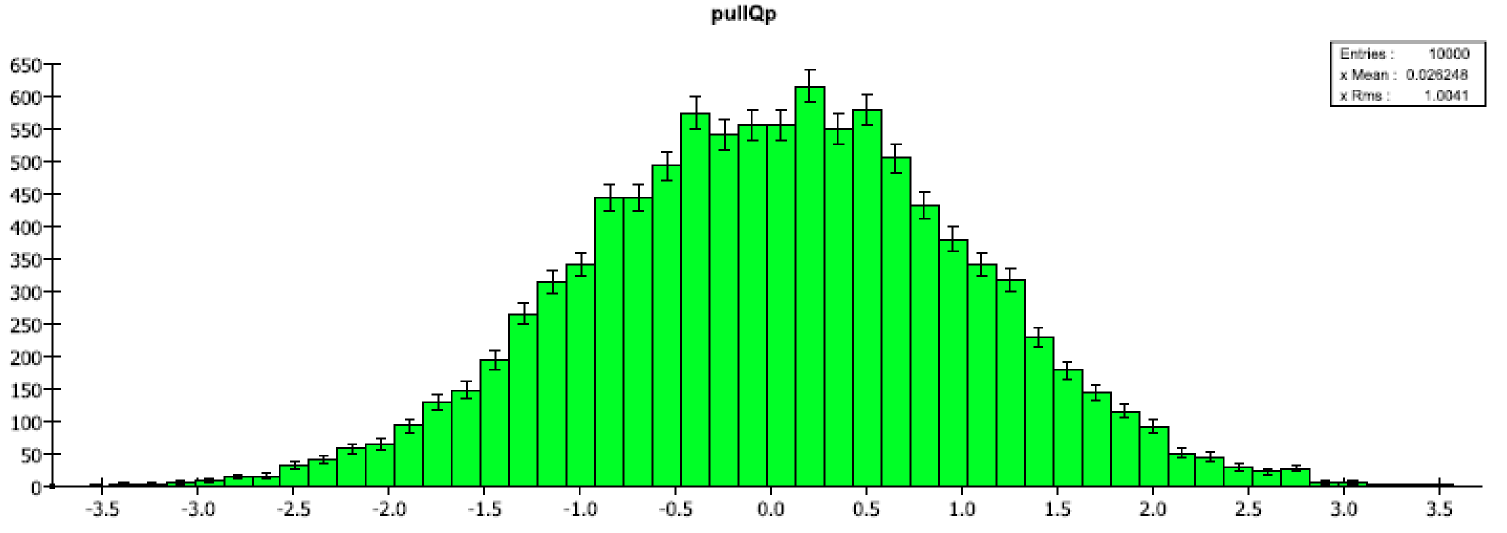


HPS Phantom Slope Res & Pulls

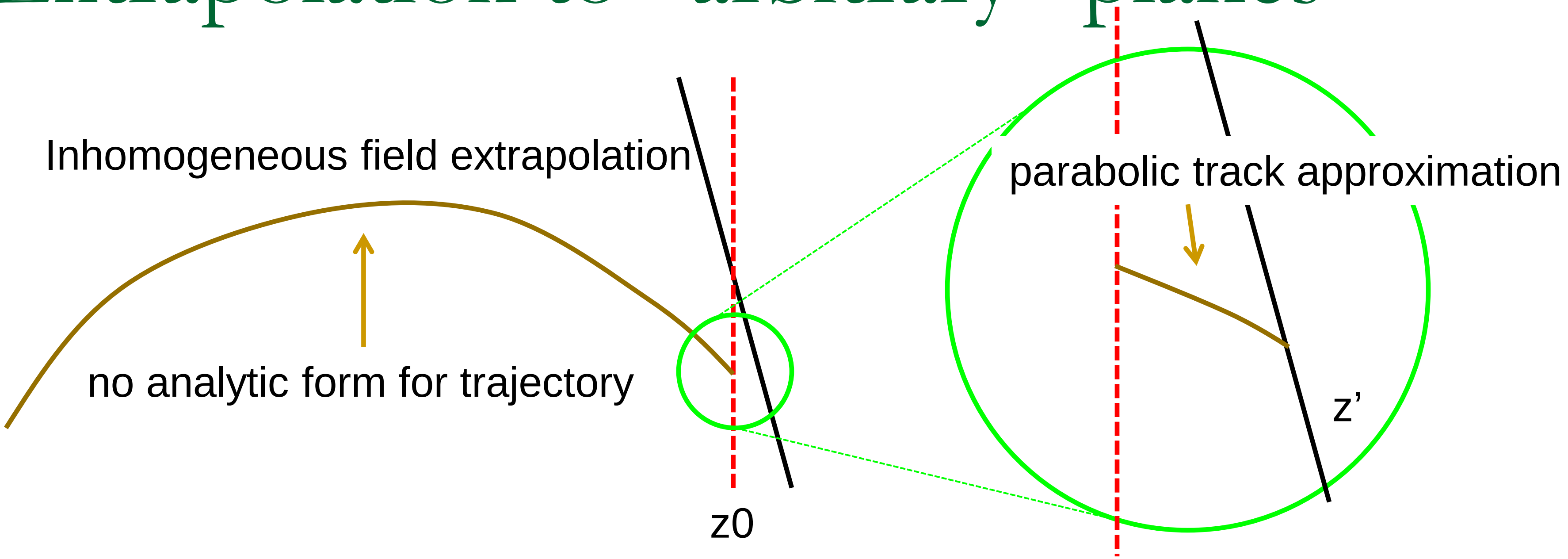


HPS Phantom q/p Res and Pull





Extrapolation to "arbitrary" planes



- Propagate to reference z position of the plane using RK and full field
- Assume field is constant enough in local neighborhood to calculate intersection of helix with plane
  - Approximating circle with parabola allows analytic calculation of intersection point.
- Propagate track to resulting z' position
  - iterate if necessary

HPS Detector

- Fitting and extrapolation code seems to be working (at least internally consistent)
- Analysis of HPS full simulation output produces pull distributions which are broad and not centered at zero, especially momentum
  - means are incorrectly reconstructed
  - uncertainties are incorrectly calculated
- Issues to be addressed for full detector reconstruction:
  - Energy loss and energy loss fluctuations
  - Coulomb scattering
  - …

Energy Loss

- Have been using standard Bethe-Bloch

$$\left(\frac{dE}{dx}\right)_{ionization} = -Kz^2 \frac{Z}{A}\frac{1}{\beta^2}\left[\frac{1}{2}\ln\frac{2m_e c^2 \beta^2 \gamma^2 T_{\max}}{I^2} - \beta^2 - \frac{\delta(\beta\lambda)}{2}\right],$$

- Implementing improved Bethe-Bloch for e^-, e^+ *

for electrons
$$\left(\frac{dE}{dx}\right)^{electrons}_{ionization} = -K\frac{Z}{A}\left[\ln\frac{2m_e c^2}{I^2} + 1.5\ln\gamma - 0.975\right],$$

for positrons
$$\left(\frac{dE}{dx}\right)^{positrons}_{ionization} = -K\frac{Z}{A}\left[\ln\frac{2m_e c^2}{I^2} + 2\ln\gamma - 1\right].$$

- Looking into Si-specific Eloss and dEloss (e.g. Bichsel)

* D.Stampfel et al. Track fitting with energy loss, Computer Physics Communications 79 (1994) 157-164

Coulomb Scattering

- Long history in HPS…
- Highland formula is a good approximation to the central Gaussian core, with scattering angle

$$\Theta_0 = \frac{13.6MeV}{\beta pc} z \sqrt{\frac{l}{X_0}} \left[1 + 0.038 \ln \frac{l}{X_0} \right]$$

- Implementing techniques to accommodate non-Gaussian large-angle scattering
  - e.g. Gaussian-mixture models

Current Vertexing

- Using Billoir implementation of Kalman Filter vertexing using perigee parameters.
  - Supposes tracks have been propagated to close proximity of the actual vertex position.
- Provide both beam-spot constrained and unconstrained vertices.
  - Need to generate a running beam-spot profile.
- For unconstrained vertex fit, would benefit by constraining vertex to point to beam-spot.
- Vertex fit provides improved track fit, which we currently neither use nor persist.
  - Add additional information into LCIO file to persist updated track fits as well as complete covariance matrix (track-vertex, track-track).

Kalman Vertex Fit (Physical parameters)

$\mathbf{r} = (x_v,\ y_v,\ z_v)^T$ — the vertex position;

$a_k, b_k, (q/p)_k$ — the directions and the inverse momentum of the k-th track, originating from the vertex $\mathbf{r}$;

$\mathbf{m}_k = (x_k, y_k, t_{xk}, t_{yk}, (q/p)_k)^T$ — the k-th track estimate, parametrized at a certain z_{ref};

V_k — the covariance matrix of the k-th track estimate;

$\mathbf{h}_k(\mathbf{r}, a_k, b_k, (q/p)_k)$ — parameters of the k-th track, extrapolated from z_v to z_{ref}:

$$\mathbf{h}_k(\mathbf{r}, a_k, b_k, (q/p)_k) = \begin{pmatrix} x_v + a_k \cdot (z_{ref} - z_v) + O((z_{ref} - z_v)^2) \\ y_v + b_k \cdot (z_{ref} - z_v) + O((z_{ref} - z_v)^2) \\ a_k \\ b_k \\ (q/p)_k \end{pmatrix}. \tag{7}$$

- Fitting follows the standard Kalman algorithm
  - Initial estimate (0,0,0), I•∞
  - Extrapolate track fits to current vertex position
  - Update vertex position
  - Iterate

Vertex Fit status

- Basic fitting functionality coded
- Implementation has been tested using phantom detector (no material effects)
- Pulls are normally distributed
  - both position and momentum
- Have not yet implemented the addition of constraints
- Not yet applied to full detector simulations
  - awaiting MCS and Eloss corrections to provide tracks with good pulls to use as input.

Summary

- Forward tracking (z) track parameter representation implemented for hps-java
- Track Extrapolation using adaptive step-size Runge-Kutta implemented and tested on Phantom detector
  - handles non-uniform magnetic field and HPS geometry
- Handling of energy loss and scattering required improvements for handling electrons.
  - Work in progress to understand and normalize pulls.
- Kalman Filter vertex fit utilizing z-track parameterizations implemented.
  - Constraints need to be implemented.
- Code needs to be documented and released.