SPD Collaboration meeting
25 October 2023

New algorithms of the primary vertex reconstruction in SpdRoot

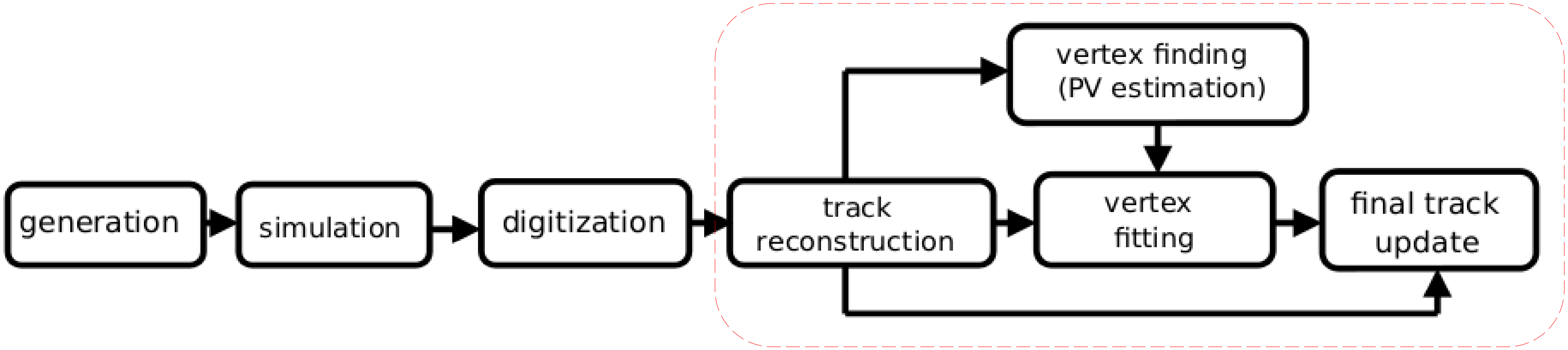
V. Andreev (LPI, Moscow)

Primary vertex reconstruction algorithm

Primary vertex reconstruction algorithm consists of two parts:

1. Initial approximation of primary vertex (vertex finding).

2. Fitting procedure for primary vertex (vertex fitting).



3. The current primary vertex reconstruction algorithm was introduced in SPDroot in 2019 and it's performance was checked with the next procedure:

   a) comparison with MC vertex position;

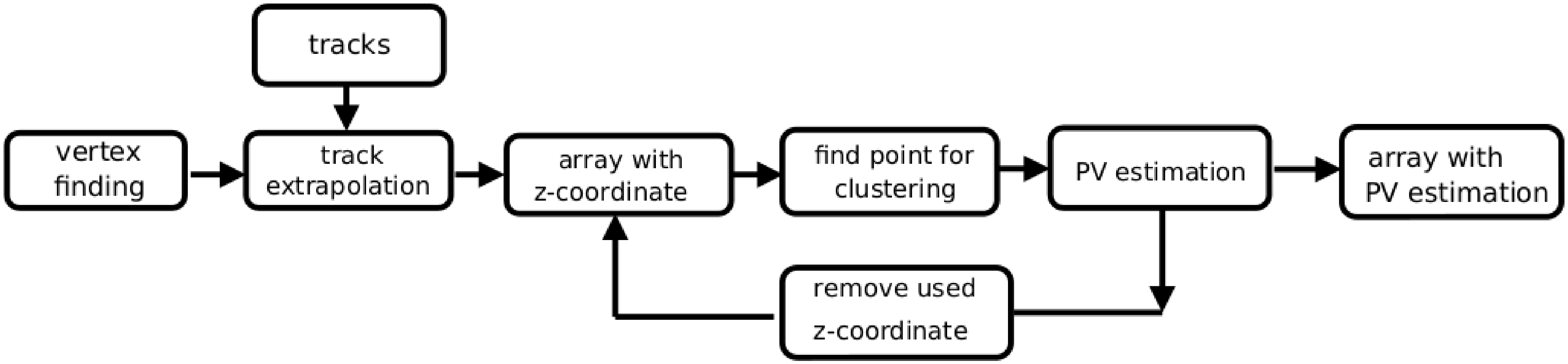
   b) comparison with the others primary vertex reconstruction algorithms used in High Energy Physics.

4. Current reconstruction algorithm shows the good performance.

5. But current algorithm has some weak points which will be presented below.

Primary vertex finding algorithm

Block schema of initial approximation for primary vertex (vertex finding)



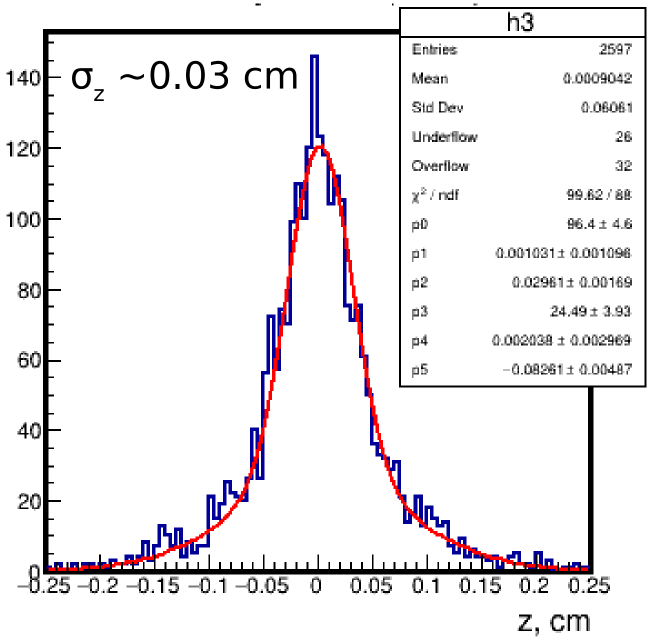
In SPD the next procedure for primary vertex finding is realized in assumption, that beam distribution in transverse XY-plane has Gaussian shape with $\sigma_{x,y}$ = 0.1 cm.

Vertex finding algorithm includes the next steps:

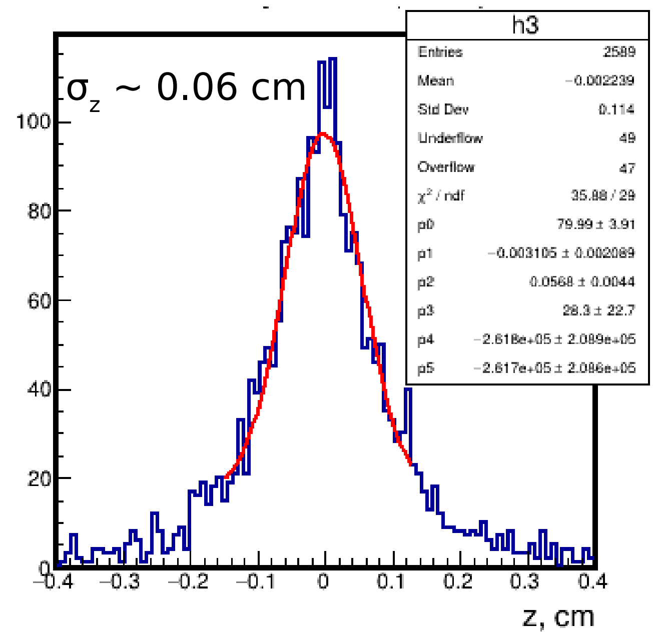
- select good tracks;
- extrapolate tracks to the beam axis;
- estimate z-coordinate for point of the closest approach to the beam axis and form array of z-points;
- apply 1-D clustering algorithm for estimation of initial primary vertex position;
- do this procedure several times;
- used vertex with maximum number of track as primary vertex candidate.

Primary vertex finding (precision)

no PV smearing



PV smearing



On the plots you see the difference in z-coordinate between the generated primary vertex position and the vertex position which is obtained with vertex finding procedure for Minimum Bias events (MAPS) and for two cases:

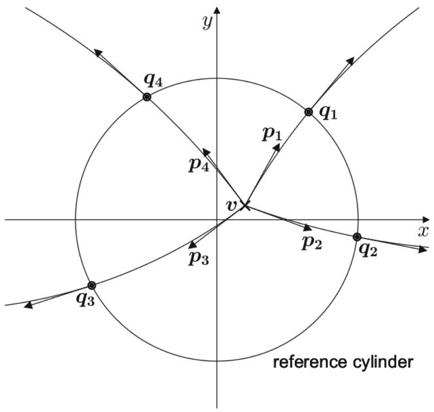
- constant generated PV position at x = y = z = 0.0 (left plot);

- PV position are distributed with Gaussian function with σ_z = 30 cm and $\sigma_{x,y}$ = 0.1 cm (right).

This vertex finding algorithm provides estimation of primary vertex position at the level ~0.6 mm

Vertex fitting algorithm

The Kalman filter algorithm is used for primary vertex fitting procedure. This algorithm was formulated as an extended Kalman filter (R. Fruhwirth, Application of Kalman filtering to track and vertex fitting, Nucl. Instr. Meth. A 262 (1987) 444.).



Usually the track parameters $\mathbf{q_i}$ are known on the some reference surface with corresponding covariance matrices $\mathbf{V_i}$, i = 1, . . . , n.

The vertex position $\boldsymbol{v}$ and momentum vectors $\boldsymbol{p_i}$ of all tracks at this vertex are the parameters which should be estimated.

The track parameters $\boldsymbol{q_i} = \boldsymbol{h_i}(\boldsymbol{v}, \boldsymbol{p_i})$ are nonlinear functions of the vertex and track parameters $\boldsymbol{v}$ and $\boldsymbol{p_i}$.

The initial information about the vertex position $\boldsymbol{v_0}$, and its covariance matrix $\boldsymbol{C_0}$ are given by some procedure as was described above.

The first-order Taylor expansion of $\boldsymbol{q_i} = \boldsymbol{h_i}(\boldsymbol{v}, \boldsymbol{p_i})$ at a some expansion point $\boldsymbol{e_0} = (\boldsymbol{v_0}, \boldsymbol{p_{i,0}})$ will gives the following approximate in linear model:

$$\boldsymbol{q}_i \approx \boldsymbol{A}_i \boldsymbol{v} + \boldsymbol{B}_i \boldsymbol{p}_i + \boldsymbol{c}_i, \ i = 1 \ldots, n,$$

$$\boldsymbol{A}_i = \left.\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{v}}\right|_{\boldsymbol{e}_0}, \quad \boldsymbol{B}_i = \left.\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{p}_i}\right|_{\boldsymbol{e}_0}$$

$$\boldsymbol{c}_i = \boldsymbol{h}_i(\boldsymbol{v}_0, \boldsymbol{p}_{i,0}) - \boldsymbol{A}_i \boldsymbol{v}_0 - \boldsymbol{B}_i \boldsymbol{p}_{i,0}.$$

$$\boldsymbol{r}_i = \boldsymbol{q}_i - \boldsymbol{h}_i(\boldsymbol{v}_i, \boldsymbol{p}_i),$$

$$\chi_i^2 = \boldsymbol{r}_i^{\mathsf{T}} \boldsymbol{G}_i \boldsymbol{r}_i + (\boldsymbol{v}_i - \boldsymbol{v}_{i-1})^{\mathsf{T}} \boldsymbol{C}_{i-1}^{-1} (\boldsymbol{v}_i - \boldsymbol{v}_{i-1}).$$

Kalman filter equations for updating vertex and track parameters with covariance matrix will look like these:

vertex $\Rightarrow$ $$\boldsymbol{v}_i = \boldsymbol{C}_i \left[\boldsymbol{C}_{i-1}^{-1} \boldsymbol{v}_{i-1} + \boldsymbol{A}_i^{\mathsf{T}} \boldsymbol{G}_i^{\mathrm{B}} (\boldsymbol{q}_i - \boldsymbol{c}_i) \right],$$

momentum $\Rightarrow$ $$\boldsymbol{p}_i = \boldsymbol{W}_i \boldsymbol{B}_i^{\mathsf{T}} \boldsymbol{G}_i (\boldsymbol{q}_i - \boldsymbol{c}_i - \boldsymbol{A}_i \boldsymbol{v}_i),$$

covariance $\Rightarrow$ $$\mathrm{Var}[\boldsymbol{v}_i] = \boldsymbol{C}_i = \left(\boldsymbol{C}_{i-1}^{-1} + \boldsymbol{A}_i^{\mathsf{T}} \boldsymbol{G}_i^{\mathrm{B}} \boldsymbol{A}_i \right)^{-1},$$

$$\mathrm{Var}[\boldsymbol{p}_i] = \boldsymbol{W}_i + \boldsymbol{W}_i \boldsymbol{B}_i^{\mathsf{T}} \boldsymbol{G}_i \boldsymbol{A}_i \boldsymbol{C}_i \boldsymbol{A}_i^{\mathsf{T}} \boldsymbol{G}_i \boldsymbol{B}_i \boldsymbol{W}_i,$$

$$\mathrm{Cov}[\boldsymbol{v}_i, \boldsymbol{p}_i] = -\boldsymbol{C}_i \boldsymbol{A}_i^{\mathsf{T}} \boldsymbol{G}_i \boldsymbol{B}_i \boldsymbol{W}_i.$$

$$\boldsymbol{W}_i = (\boldsymbol{B}_i^{\mathsf{T}} \boldsymbol{G}_i \boldsymbol{B}_i)^{-1}$$

$$\boldsymbol{G}_i^{\mathrm{B}} = \boldsymbol{G}_i - \boldsymbol{G}_i \boldsymbol{B}_i \boldsymbol{W}_i \boldsymbol{B}_i^{\mathsf{T}} \boldsymbol{G}_i.$$

Possible simplification of Kalman algorithm

The disadvantage of the basic method is that it requires too many calculations. This is especially important for high multiplicity events.

For speeding up the calculations the following simplifications are usually applied:

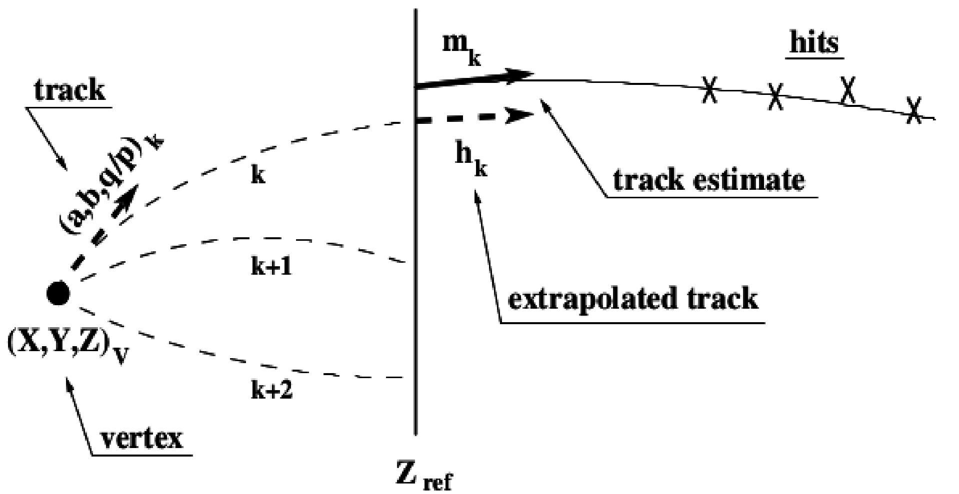
- neglect of the magnetic field in the vertex region when tracks are considered as straight lines;
- fixation of the track directions and momenta neglecting uncertainties of these parameters;
- use of initial track parameters for linearization at each iteration;

Current vertex fitting algorithm

A special feature of the current algorithm is the next:

- track is extrapolated to the some virtual plane z_{ref} (z_{ref} is determined from clustering algorithm)
- then track parameters are estimated on this virtual plane;
- use the second order curve for description of track trajectory;
- and finally track parameters are linearized in the vicinity of this point.

This approach makes it possible to fit the vertex without including the track parameters into the vertex state vector and to simplify the calculations.



$\boldsymbol{r} = (x_v, y_v, z_v)^T$, $\boldsymbol{C_v}$ — the vertex position and its covariance matrix;

$\boldsymbol{t_k} = (a_k, b_k, (q/p)_k)^T$, $\boldsymbol{C^{tk}}$ — the directions and the inverse momentum of the k-th track, originating from the vertex $\boldsymbol{r}$, and covariance matrix for these parameters; measurement;

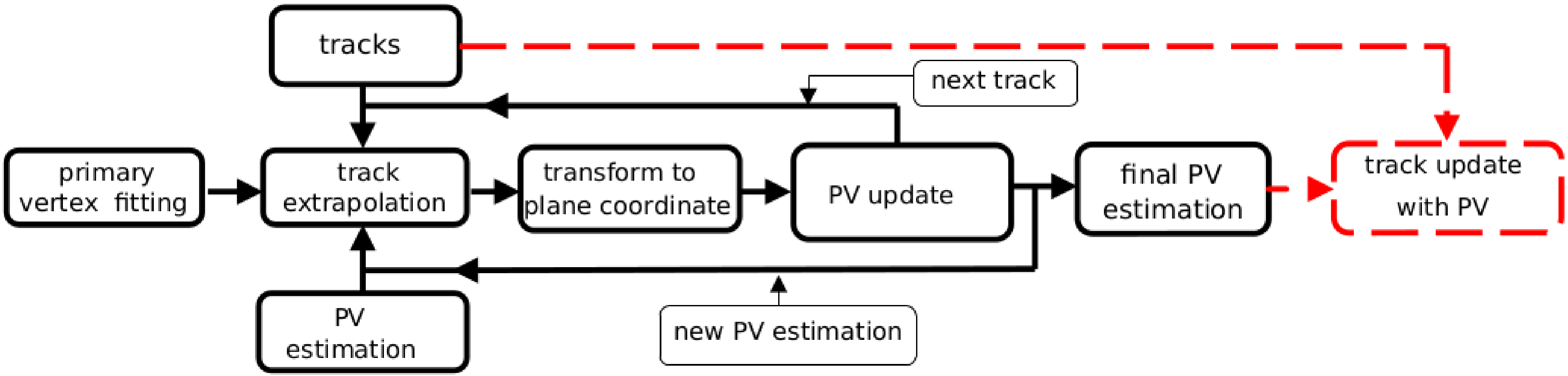
$\boldsymbol{h_k}(r, a_k, b_k, (q/p)_k)$ — parameters of the k-th track, extrapolated from $\boldsymbol{z_v}$ to $\boldsymbol{z_{ref}}$;

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

$\boldsymbol{V_k}$ — the covariance matrix of the k-th track estimate;

Each track estimation $\boldsymbol{m_k}$ is considered as measurement of the corresponding track $\boldsymbol{t_k}$.

Current vertex fitting algorithm



General schema for current primary vertex fitting algorithm

The current primary vertex fitting algorithm has the some “weak” points:

- does not include track with theta angle +-3^0 around the vertical plane in fitting procedure;
- use the second order curve for track trajectory description;
- track parameters are considered constant during fitting procedure;
- need additional step for final update track parameters at primary vertex (red box).

Cross check with others reconstruction algorithm

The cross check for the current SPD primary vertex reconstruction algorithm was done on the base of comparison with the others vertex reconstruction algorithms which used in high energy physics.

The next algorithms were considered:

- classical Kalman vertex filter (KVF);

- trimmed Kalman filter (TVF);

- adaptive vertex filter (AVF);

- primary vertex reconstruction from KFParticle package (KFP);

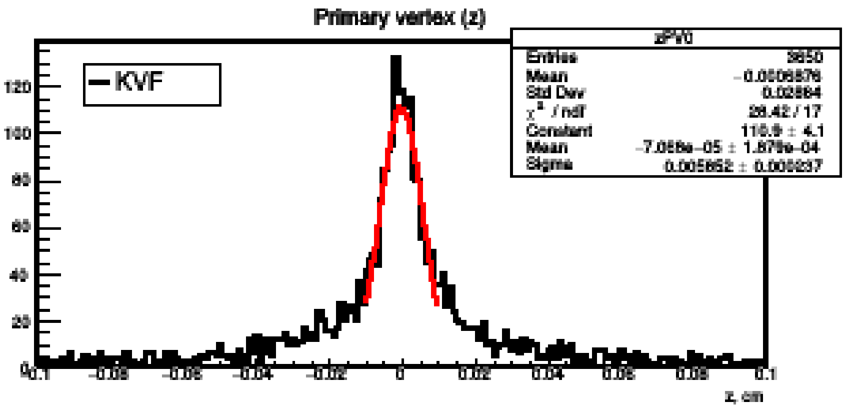
- current SPD vertex reconstruction algorithm.

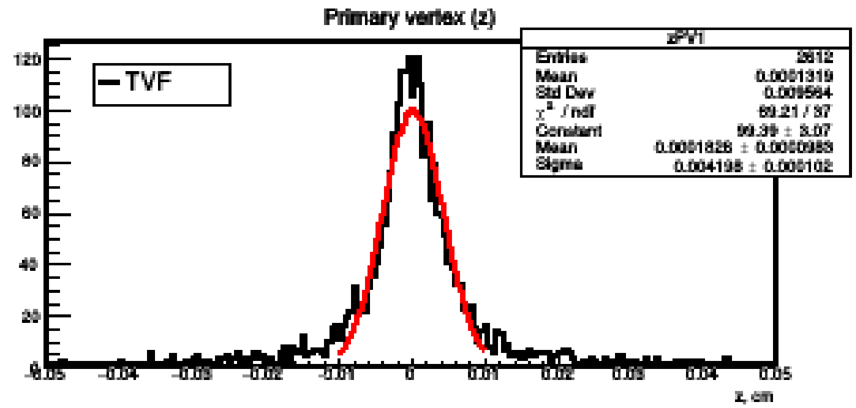
The KVF, TVF and AVF primary vertex reconstruction algorithms are realized in the special RAVE package (Reconstruction in an Abstract, Versatile Environment) which was developed for vertex reconstruction in CMS experiment.

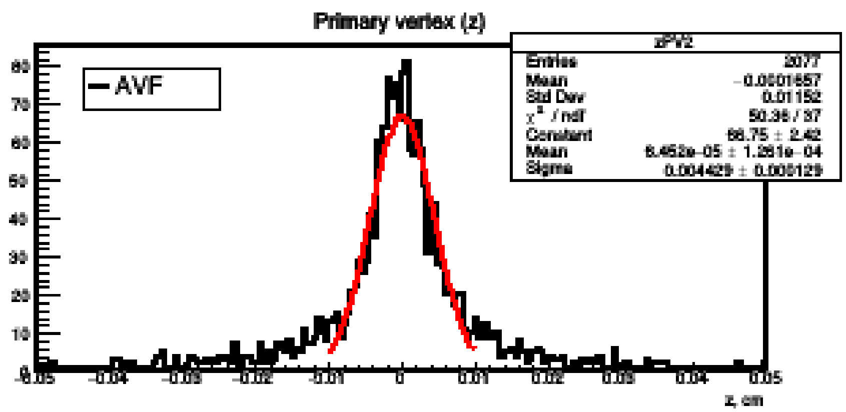
It is necessary to emphasize that the considered algorithms in RAVE use the different parameters for track representation and also various algorithms for finding initial approximation of the primary vertex with comparison of SPD reconstruction.
Also these algorithms (KVF, TVF, AVF, KFP) are realized in assumption of the constant magnetic field and for this reason the comparison was done for the constant magnetic field also for SPD primary vertex reconstruction.

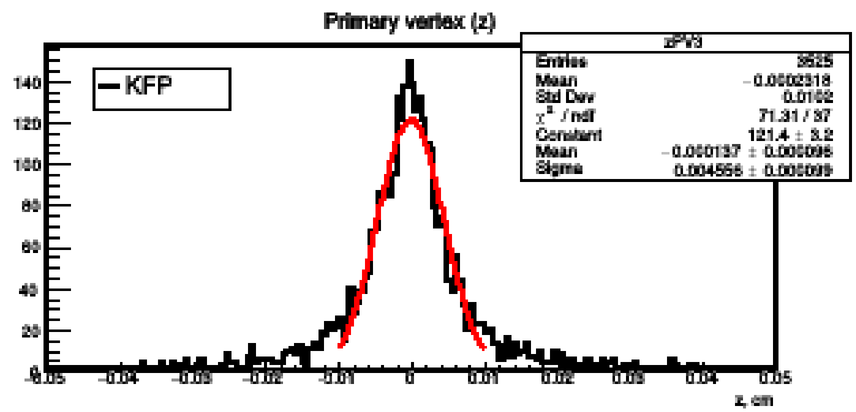
The special interface was written in order to use data from SPD in RAVE package.

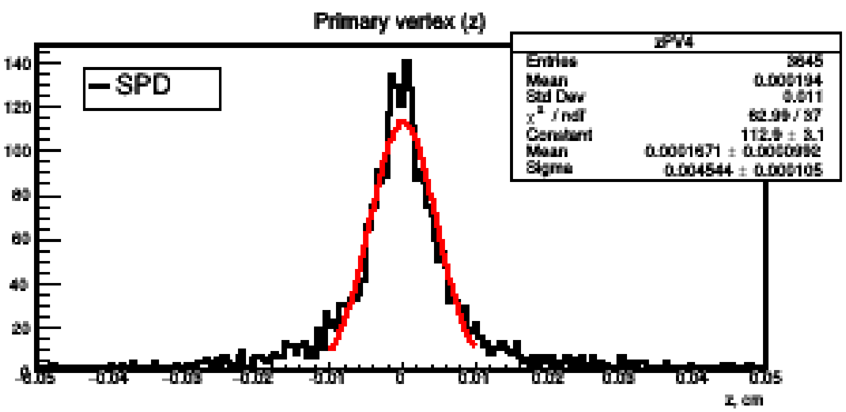
Comparison of different algorithms (MB, MAPS+DSSD)











Current primary vertex reconstruction algorithm shows the good agreement with TVF and KFP algorithms.

AVF algorithm demonstrates a little worse results and KVF shows more worse results

New algorithm for vertex fitting

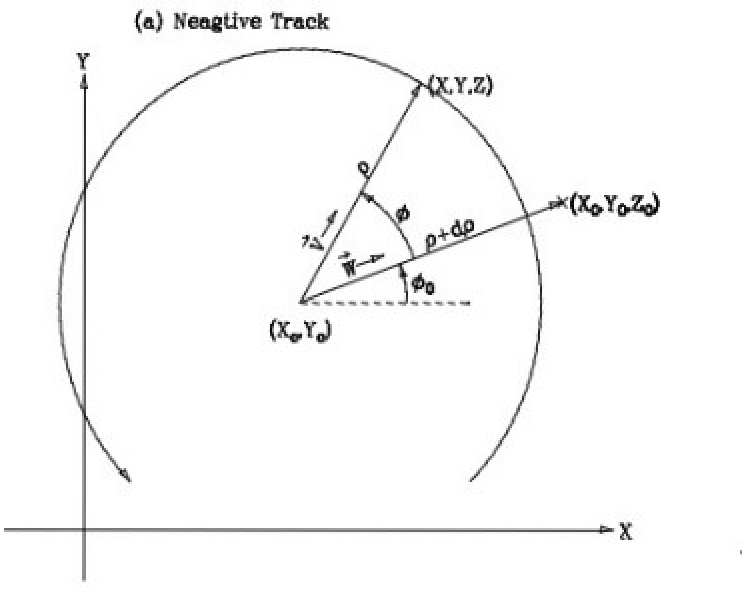
New primary vertex reconstruction algorithm will try to remove the “weak” points of the current algorithm.

The next set parameters are used for track description in new algorithm:

1) 6 global parameters => ***x,y,z*** – coordinates and momentum of track $\boldsymbol{p_x}$ **,** $\boldsymbol{p_y}$ **,** $\boldsymbol{p_z}$ at this point (now in SPDroot these track parameters with covariance matrix are known at the first measured point => GetFirstState());

2) 5 local parameters (helix) which usually used for description of the helix in constant magnetic field. These are the same helix track parameters as used in KEK or BES3 experiments:

$$\begin{cases} x = x_0 + d_\rho \cos\phi_0 + \frac{\alpha}{\kappa}(\cos\phi_0 - \cos(\phi_0 + \phi)) \\ y = y_0 + d_\rho \sin\phi_0 + \frac{\alpha}{\kappa}(\sin\phi_0 - \sin(\phi_0 + \phi)) \\ z = z_0 + d_z - \frac{\alpha}{\kappa}\tan\lambda \cdot \phi, \end{cases}$$



where $x_0 = (x_0 , y_0 , z_0)^T$ is an arbitrarily chosen reference point. If the reference point is fixed, the helix is determined by a 5-component parameters vector h = $(d\rho , \varphi_0 , \kappa, d_z , \tan \lambda)^T$, where dρ is the distance of the helix from the reference point in the xy plane, φ_0 is the azimuthal angle to the reference point with respect to the helix center, $k = Q/p_T$, d_z is the distance of the helix from the reference point in the z direction, and tan λ is the dip angle.

New algorithm for vertex fitting

If we know 6 global track parameters at some x point of helix then the track helix parameters can be calculated

$$\mathrm{h} = \begin{pmatrix} \tilde{d}_\rho \\ \tilde{\phi}_0 \\ \tilde{\kappa} \\ \tilde{d}_z \\ \tilde{\lambda} \end{pmatrix} = \begin{pmatrix} -\dfrac{T - p_\perp}{a} \\ \tan^{-1}\left[-\dfrac{p_x + ay}{p_y - ax}\right] \\ \dfrac{Q}{p_\perp} \\ z - \dfrac{p_z}{a}\sin^{-1} J \\ \dfrac{p_z}{p_\perp} \end{pmatrix},$$

$$p_\perp = \sqrt{p_x^2 + p_y^2},$$

$$T = \sqrt{(p_x + ay)^2 + (p_y - ax)^2},$$

$$J = \sin \rho s_\perp = \frac{p_{0x}p_y - p_{0y}p_x}{p_\perp^2} = \frac{p_y}{p_\perp}\cdot\frac{p_x + ay}{T} - \frac{p_x}{p_\perp}\cdot\frac{p_y - ax}{T} = \frac{a}{Tp_\perp}(xp_x + yp_y).$$

TMatrixD m_a(5,3); $\boldsymbol{A_i = \partial h_i / \partial v}$

```
m_a(0,0) = 0. + (py - a*x)/T;
m_a(0,1) = 0. - (px + a*y)/T;
m_a(1,0) = 0. - a*(px + a*y)/T/T;
m_a(1,1) = 0. - a*(py - a*x)/T/T;
m_a(3,0) = 0. - (pz/T)*(px + a*y)/T;
m_a(3,1) = 0. - (pz/T)*(py - a*x)/T;
m_a(3,2) = 1.;
```

TMatrixD m_b(5, 3); $\boldsymbol{B_i = \partial h_i / \partial p_i}$

```
m_b(0,0) =      (px/pxy-(px+a*y)/T)/a;
m_b(0,1) =      (py/pxy-(py-a*x)/T)/a;
m_b(1,0) = 0. - (py-a*x)/T/T;
m_b(1,1) = 0. + (px+a*y)/T/T;
m_b(2,0) = 0. - charge*px/pxy/pxy/pxy;
m_b(2,1) = 0. - charge*py/pxy/pxy/pxy;
m_b(3,0) = 0. + (pz/a)*(py/pxy/pxy-(py-a*x)/T/T);
m_b(3,1) = 0. - (pz/a)*(px/pxy/pxy-(px+a*y)/T/T);
m_b(3,2) = 0. - asin(J)/a;
m_b(4,0) = 0. - (px/pxy)*(pz/pxy)/pxy;
m_b(4,1) = 0. - (py/pxy)*(pz/pxy)/pxy;
m_b(4,2) =      1./pxy;
```

New vertex fitting procedure

General fitting procedure:

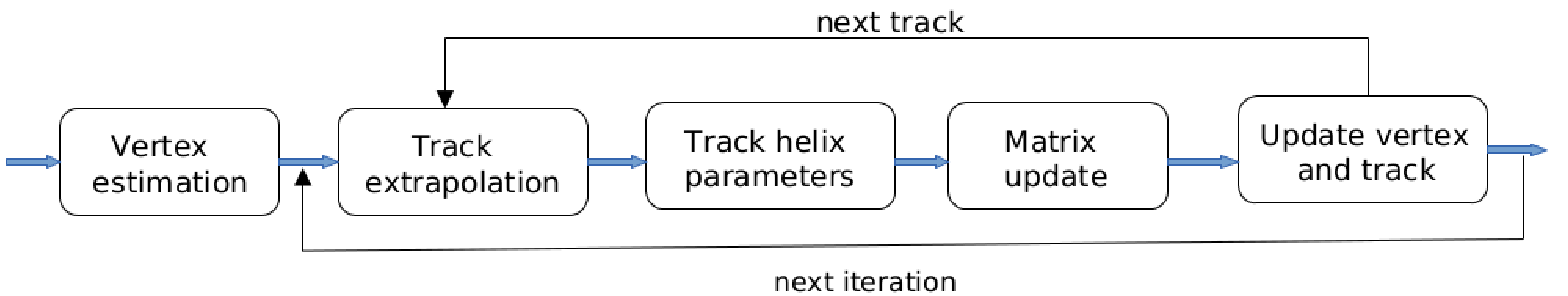
1. determine the preliminary position of the primary vertex (vertex finding);

2. extrapolate track to this vertex position using Runge-Kutta-Nyström method;

3. transform track to the local helix parameters in the area of this vertex;

4. apply linearization of track parameters => calculate all necessary matrix;

5. update vertex position and track parameters and corresponding covariance matrix
   using Kalman filter equations;

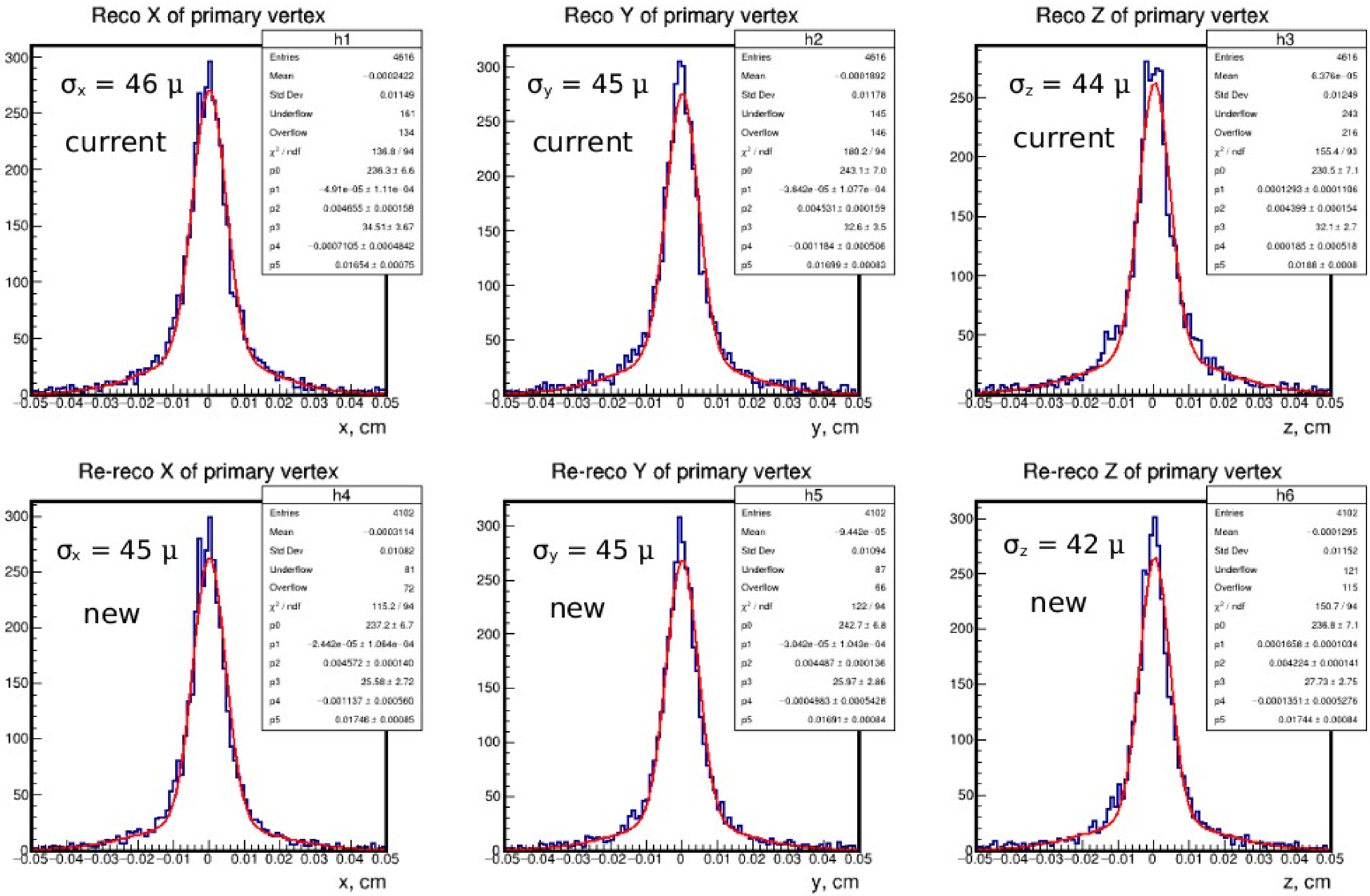


6. do loop over all selected tracks;

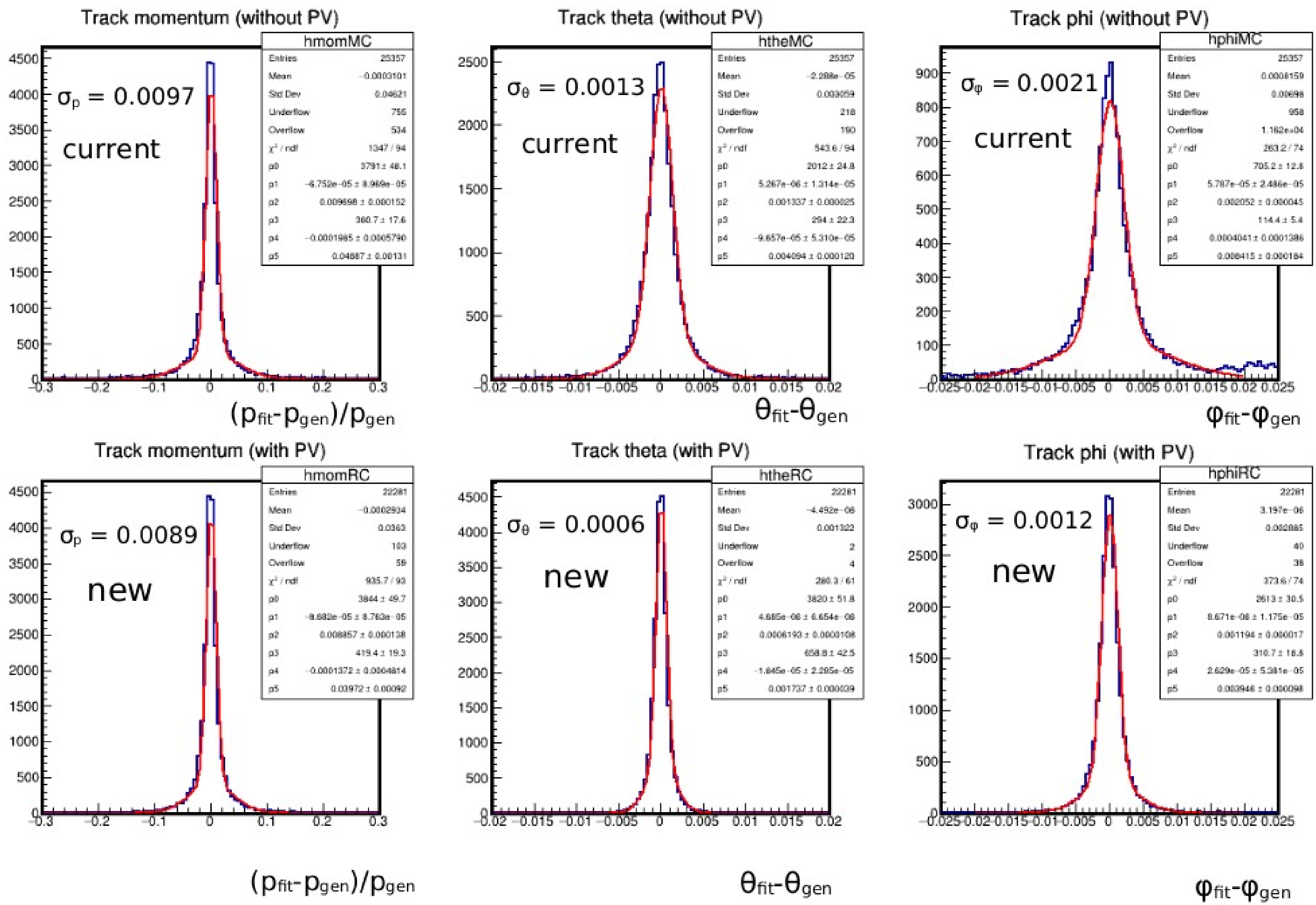
7. do this procedure several time (iteration)

8. finally the primary vertex position and tracks parameters which connected with this vertex
   and corresponding covariance matrix are determined

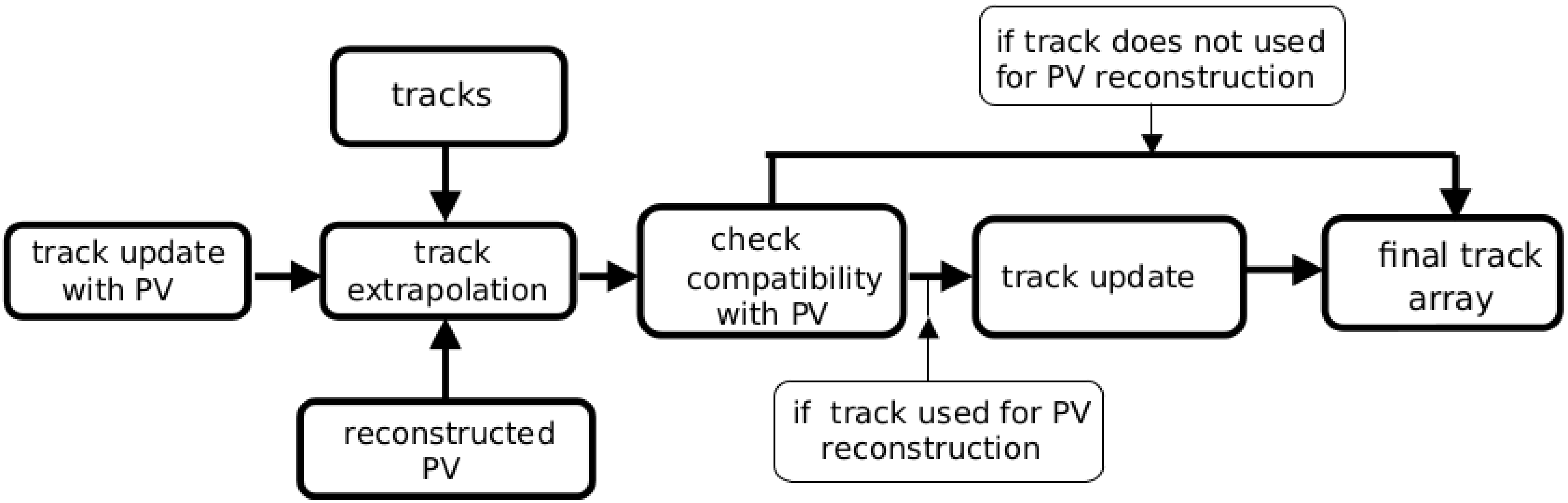
Primary vertex precision



Track parameters precision



Update track parameters at primary vertex for current algorithm



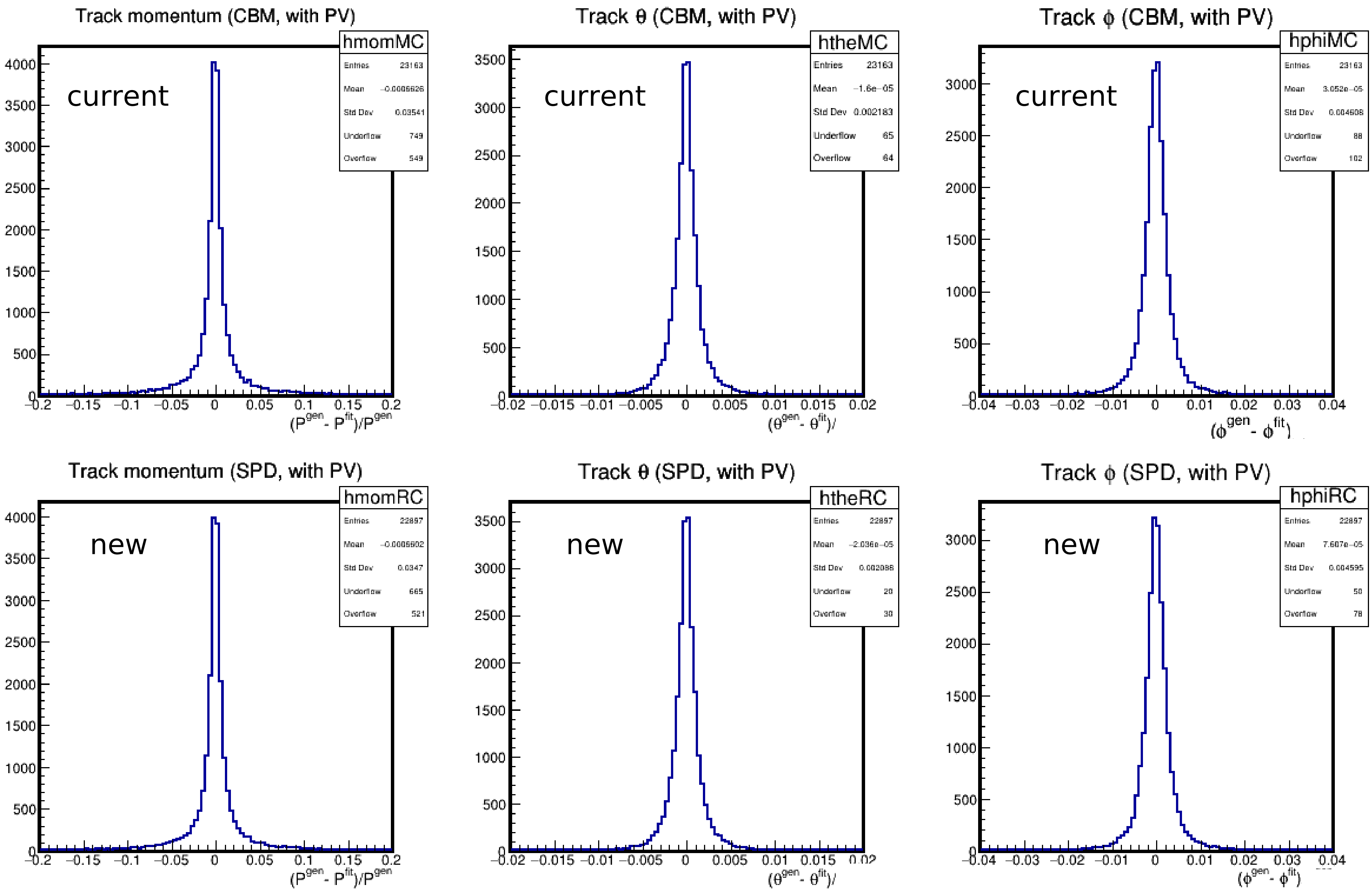
1. track is extrapolated to the reconstructed primary vertex and the track parameters (position, momentum and covariance) are defined at point of closest approach;

2. if track was used for the primary vertex fitting procedure then PV is considered as additional hit point for this track and track parameters are updated with Kalman filter procedure;

3. put these new track parameters in FinalState – function (GetFinalState());

4. if track was not used for the fitting procedure – nothing to do with this track.

Track parameters precision



Current primary reconstruction algorithm with additional step for updating track parameters and new primary vertex algorithm show the very similar results

Comparison of two primary vertex fitting algorithms

	current	new
1. general selection of track	yes	yes
2. additional selection $(\theta - \pi/2) < \Delta\theta$	yes	no
3. initial track parameters	6 global	6 global
4. track extrapolation to	plane (XY)	to space point
5. track extrapolation method	Runge-Kutta	Runge-Kutta
6. local track parameters	5 on plane	5 helix
7. local track description	~curve 2-d order	exact helix
8. track linearization	yes	yes
9. update vertex at each step	yes	yes
10. track selection ($\chi 2$) at each step	yes	yes
11. update track parameters	no	yes
12. can be used in nonuniform field	yes	yes
13. additional step for track parameters update	yes	no
14. performance	similar	similar

Summary

1. Current primary vertex reconstruction algorithm was introduced in 2019 in SPDroot on the base of CBM reconstruction algorithm.

2. Current vertex reconstruction algorithm shows good reconstruction precision.

3. But current algorithm has some weak points: does not use tracks at $\sim 90^0$, no updating track parameters at primary vertex.

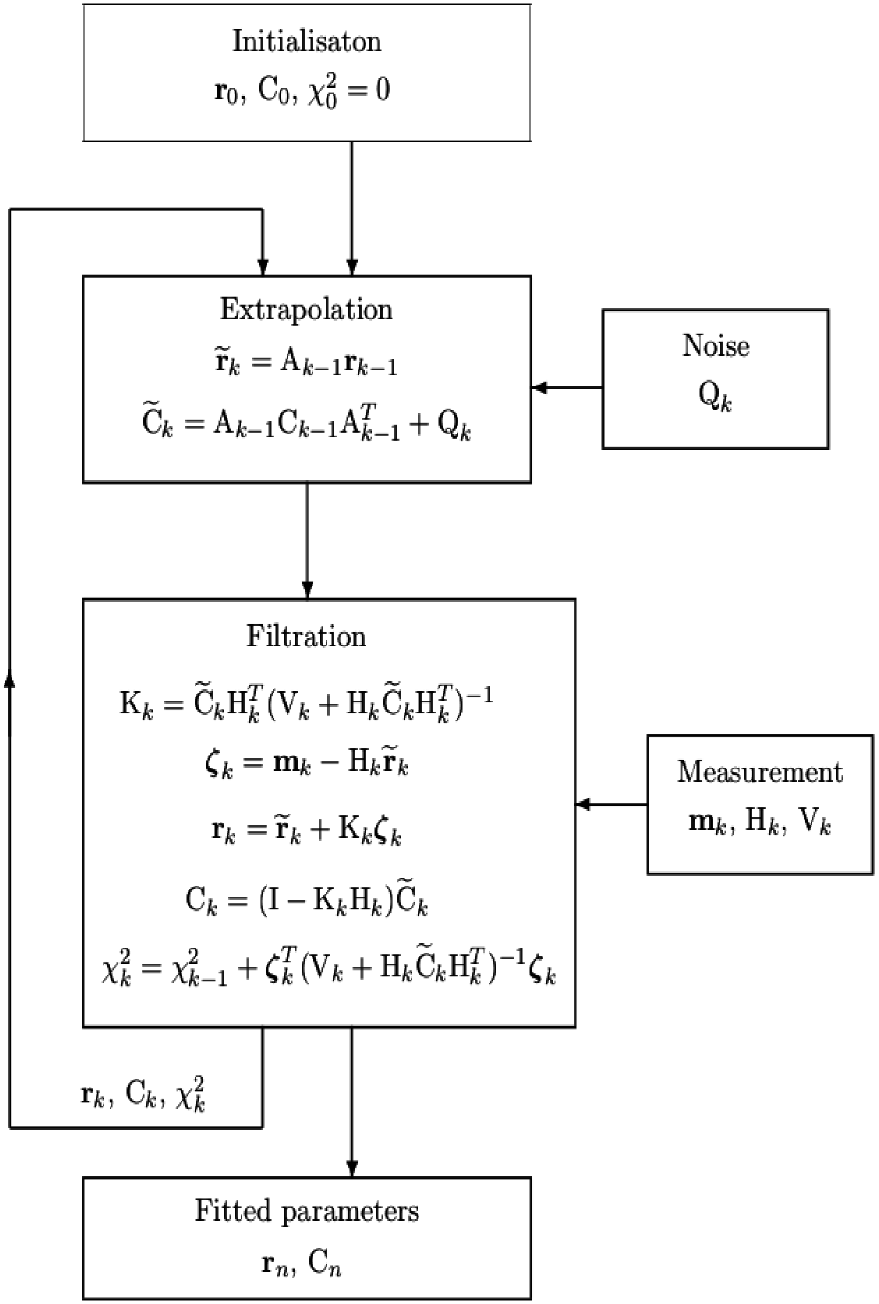
4. The new primary vertex reconstruction algorithm is proposed.

5. Also the new procedure for updating track parameters at primary vertex is added for current primary vertex algorithm.

6. Both reconstruction algorithm show good and compatible results.

7. It looks reasonable to have both vertex reconstruction algorithms in SPDroot software.

Kalman filter algorithm



state vector $\boldsymbol{r^t}$ – vector real numbers that represents the unknown quantities to be estimated

Extrapolation – changes current estimation of vector $\boldsymbol{r_k}$ upon transfer from (k-1)-th measurement to the k-th measurement

$\boldsymbol{r_k^t = A_k\, r_{k-1}^t + v_k}$, $\boldsymbol{A_k}$ – is a known linear operator, called **etrapolator;** v_k – process noise between (k-1) and k - measurement

Filtration – the measurement information is incorporated into the estimator and its covariance matrix

measurement $\mathbf{m_k}$ – a known (measured) quantity with linearly depends on state vector $\mathbf{m = H \cdot r^t + \eta}$
H – is a (known) linearly operator represented as a matrix, called **model of measurement; η –** measurement error

Steps extrapolation and filtration sequentially repeat n times, for each measurement m_k , k = 1,...n. After the filtration of the last measurement $\boldsymbol{m_n}$, the obtained estimator r_n is the desired best estimator with the covariance matrix $\boldsymbol{C_n}$.

In practice, the transport equation and the measurement model are often nonlinear. To solve the nonlinear fit problem, one should linearise all the equations before applying the fitting algorithm, but the algorithm itself does not change.