SPD Collaboration Meeting
5-8 November 2024

Status of primary vertex reconstruction in SpdRoot

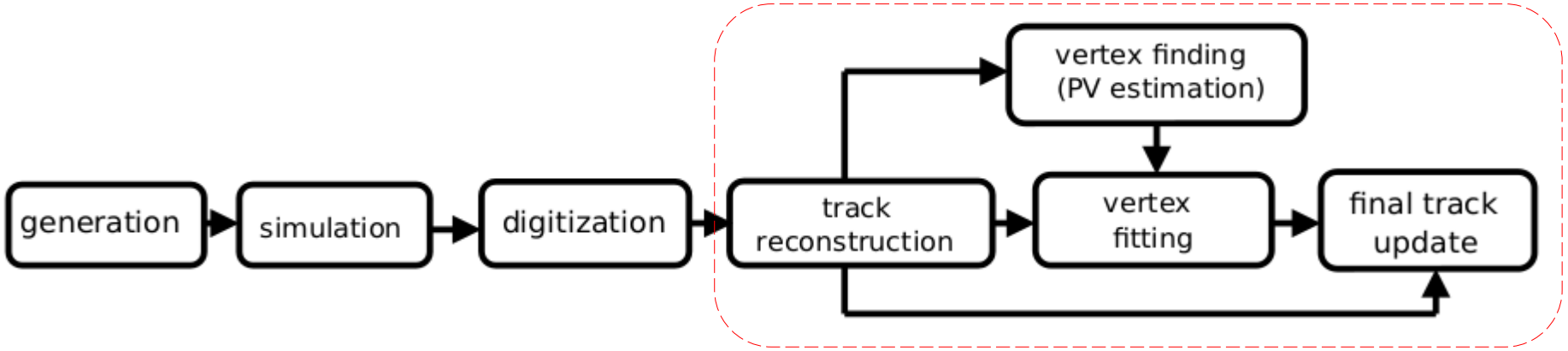
V. Andreev (LPI, Moscow)

Introduction

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 (on the base of CBM algorithm) was introduced in SPDroot 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. The new vertex reconstruction algorithm on the base of KFParticle is now introduced in SPDroot.

Primary vertex finding

Primary vertex finding procedure is very important part of the vertex reconstruction algorithm.

The main specific feature of SPD experiment is the very broad primary vertex distribution in z-direction (σ_z = 30. cm). In SPD the vertex finding procedure is realized in assumption, that the beam distribution in transverse XY-plane is relatively small and has Gaussian shape with $\sigma_{x,y}$ = 0.1 cm.

The next algorithms are realized for initial approximation of the primary vertex:

a) 1-D clustering procedure:

- select good fitted tracks;

- extrapolate tracks to the beam axis (x=y=0) and assign to each track a weight which is proportional to inverse track error at the point of closest approach to the beam axis;

- remove tracks which are faraway from the beam axis;

- construct 1-st estimation of z-vertex position from the selected tracks with taking into account the track weight;

- find the “best” and “worst” tracks:
  a) “worst” track – track with maximum distance from z-vertex position;
  b) “best” track – track with maximum weight (or minimum error) inside some range around z-vertex position;

- produce 2-nd estimation of z-vertex position without “worst” track;

- do next iteration;

- final vertex estimation => construct cluster around “best” track using only tracks inside some range around z-position of the “best” track with taking into track weight.

Primary vertex finding (2)

b) 3-D clustering procedure on the base of KFParticle package.

Vertex construction procedure in KFParticle package from n-daughter particles:

- determine point of closest approach for two first daughter particles;
- extrapolate daughter particle to this point;
- update position of this point using standard Kalman filter equations;
- track parameters are remained unchanging during this procedure.

Procedure for primary vertex finding algorithm:

- select good fitted tracks;

- extrapolate tracks to the beam axis (using Runge-Kutta algorithm) and construct KFParticle track parameters in the point of closest approach to the z-axis;

- remove tracks which are faraway from the beam axis;

- construct vertex from tracks using KFParticle procedure (1-st step, put Bz value at x=y=z=0.0);

- find " worst" track with maximum chi2 to the constructed vertex and larger than some threshold value;

- remove " worst" track;

- do next iteration until all "worst" tracks will be removed or only 2 tracks are remained;

- construct the final primary vertex estimation using remaining KFParticle tracks.

Primary vertex finding (3)

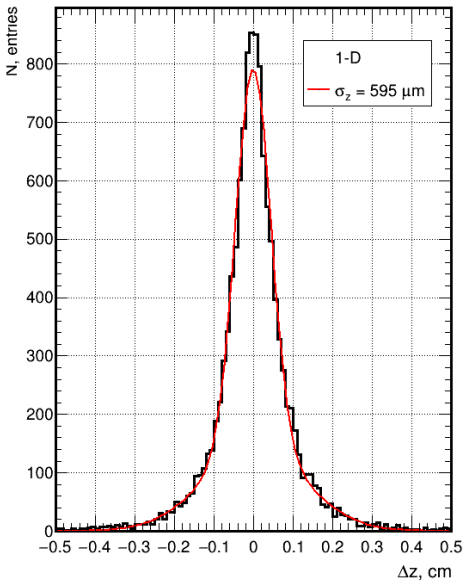
Minimum Bias events (with primary and secondary tracks) is simulated for vertex detector with 1 layer of MicroMegas. And then two vertex finding procedure was applied:

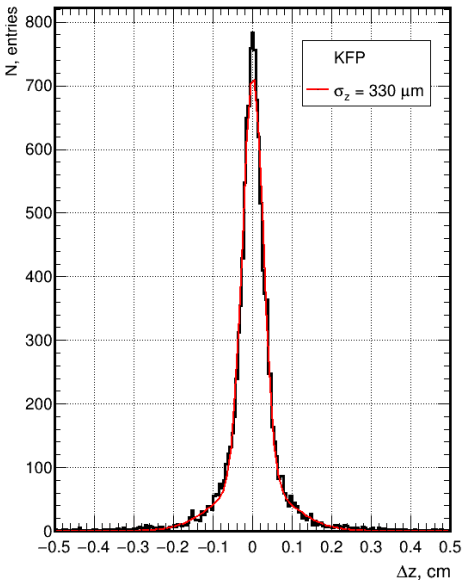
a) 1-D clustering procedure;

b) 3-D clustering procedure on the base of KFParticle package.

Fit is done by two Gaussian functions and sigma is determined as mean value of two sigmas with corresponding weights.

σ_z = 595 μm





σ_z = 330 μm

Plots show the difference of z-coordinate between the generated primary vertex position and vertex position which is provided by the vertex finding methods (1-D and 3-D on the base of KFParticle package).

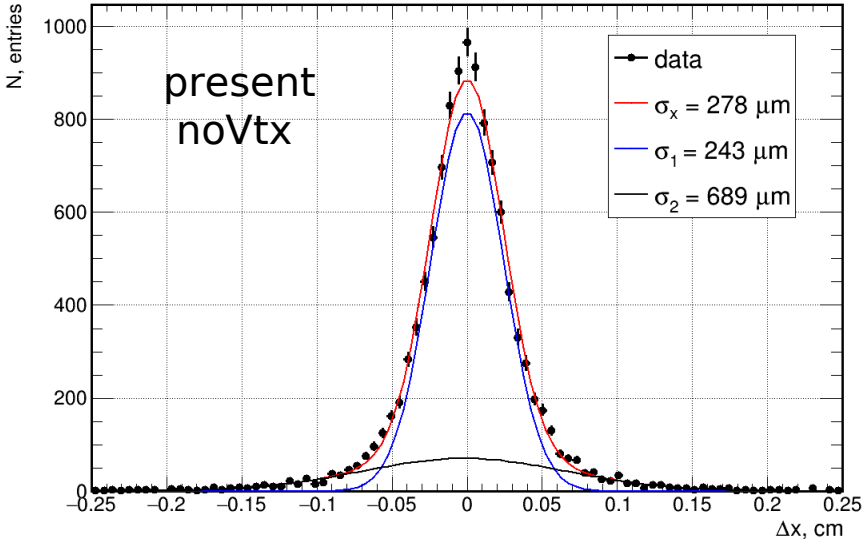
3-D algorithm on the base of KFParticle package shows the better precision (~1.8 times) for vertex finding with comparison of 1-D method.

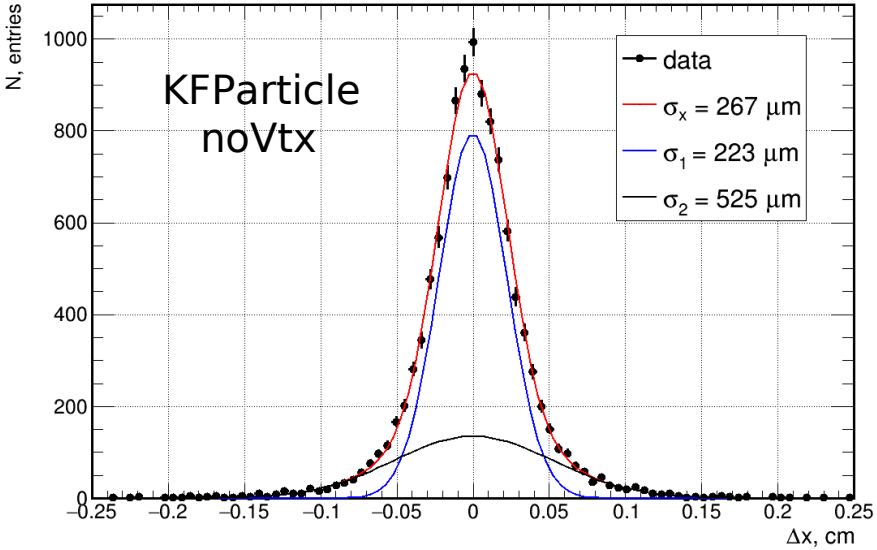
Primary vertex fitting algorithms

The vertex fitting procedure uses the vertex position which is defined in vertex finding algorithm as the 1-st approximation. Fitting algorithms use the Kalman filter method.
The next 2 algorithms are now available:

a) fitting algorithm on the base of CBM experiment method (or present);

b) new primary vertex fitting algorithm which uses the KFParticle package.

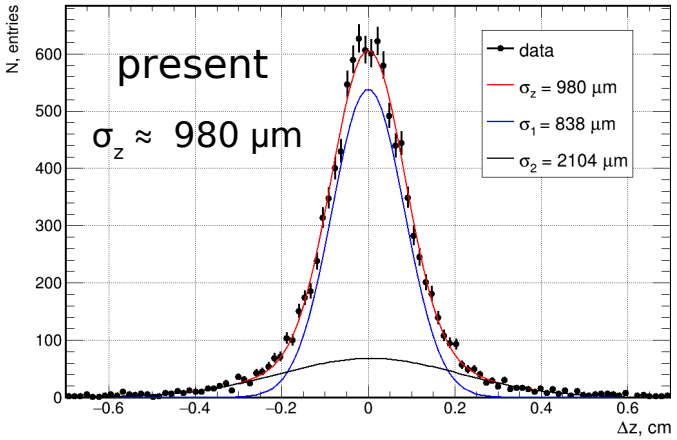




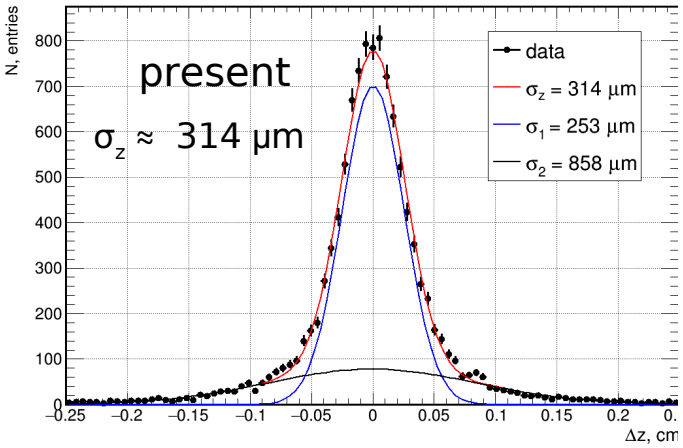
10000 Minimum Bias events were simulated with Pythia8. Primary vertex was smeared with Gaussian function ($\sigma_{x,y}$ = 0.1 cm and σ_z = 30 cm). For reconstruction procedure “good” reconstructed tracks are selected: fitted track with chi2/ndf < 12 and with number of hits > 8 (in vertex and straw detectors).

Different vertex detector options were also considered: without vertex detector, with 1 and 3 layers of MicroMegas detector, DSSD and MAPS detectors. Reconstruction precision Is determined as the difference between reconstructed and generated value. Fit is done by two Gaussian functions and final sigma is determined as mean value of two sigmas with corresponding weights.

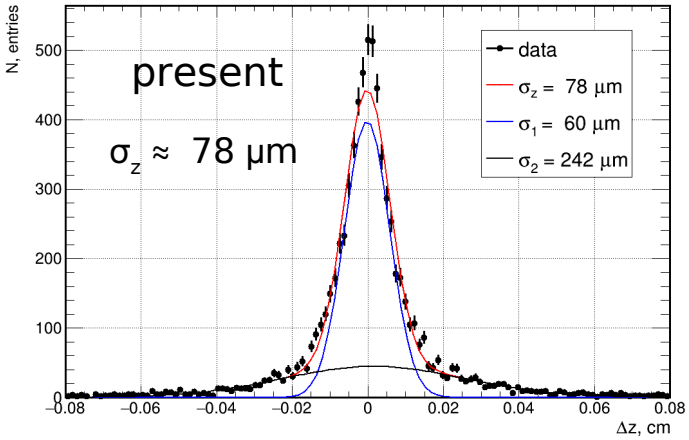
Primary vertex reconstruction



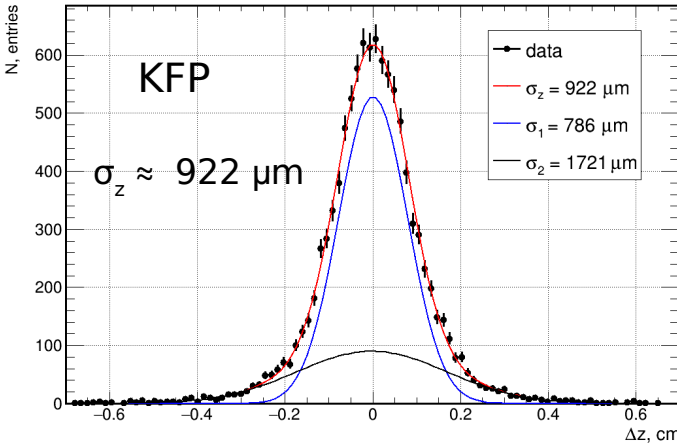
noVtx

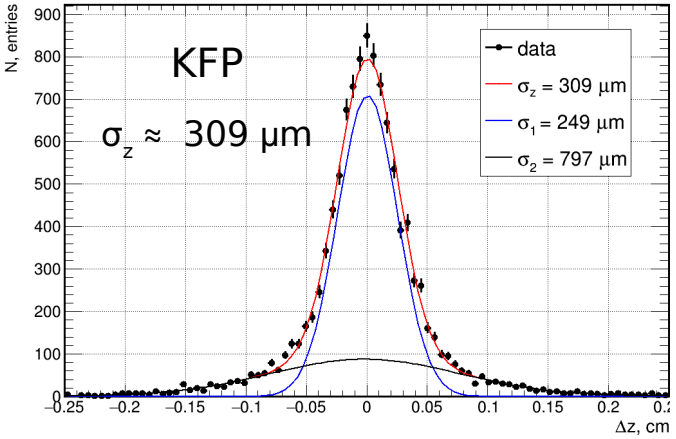


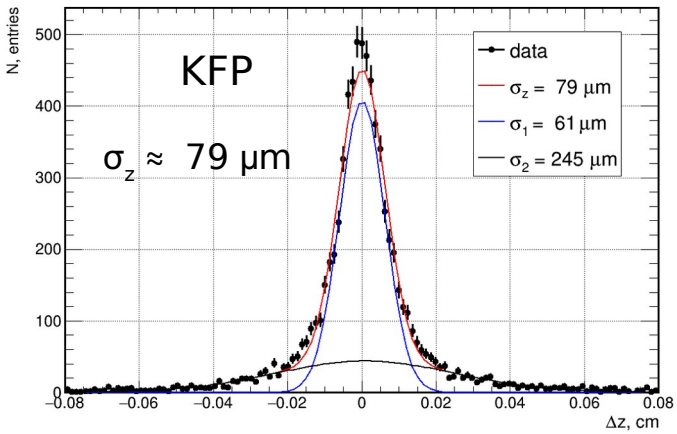
MM, 1 layer



MAPS







Both algorithms (present and KFParticle) show the similar primary vertex reconstruction precision for all considered option of vertex detector.

New option for primary vertex reconstruction

The next options for primary vertex reconstruction procedure are proposed:

1) present class **SpdRCVerticesFinder** will stay the same (only minor corrections are added) and can be used with the standard commands in reco macro:

```
SpdRCVerticesFinder* vtxs_finder = new SpdRCVerticesFinder();     <= init PV finder class
Run→AddTask(vtxs_finder);                                         <= run
```

2) new class **SpdRCVertexFinder** is added which contains different vertex finding algorithms;

3) new primary vertex reconstruction class **SpdRCPrimVertexFinder** is also added which contains possibility to select vertex finding and fitting algorithms with the next set of commands in reco macro:

```
SpdRCPrimVertexFinder* vtx_fitter = new SpdRCPrimVertexFinder();  <= init new PV class

SpdRCVertexFinder*  vtx_RCfinder = vtx_fitter→RCFinder();    <= init vertex finder class
SpdRCTrackFitterGF* vtx_RCfitter  = vtx_fitter→RCFitter();     <= init fitter class

//vtx_fitter->SetPVFinderMethod(1);                 // 1 - 1D  (default); others - 3D;
//vtx_fitter→SetPVFitterMethod(1);                  // 0 - CBM (default); >=1 – KFParticle;


Run→AddTask(vtx_fitter);                            // run
```

Using SpdRCPrimVertexFinder for re-reconstruction of PV

Below you see the example of using the new primary vertex reconstruction procedure in user analyzer (from directory **spdroot/macro/examples/K0decay**):

PrimVertexAnalyzer.C – macro with example how to use the primary vertex finder and vertex fitting procedure in user analyzer for re-reconstruction of primary vertex:

a) select tracks;
b) init primary vertex finder and fitter classes:

```
SpdRCVertexFinder fVertexFinder;
fVertexFinder.Init();

SpdRCPrimVertexFinder fVertexFitter;
fVertexFitter.Init();
```

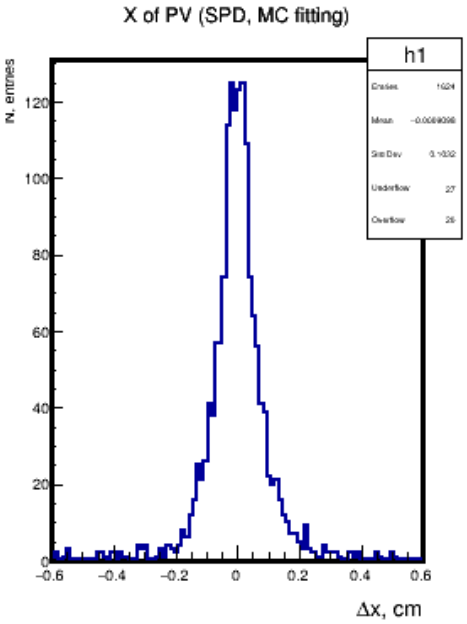
c) run primary vertex finder (select method):

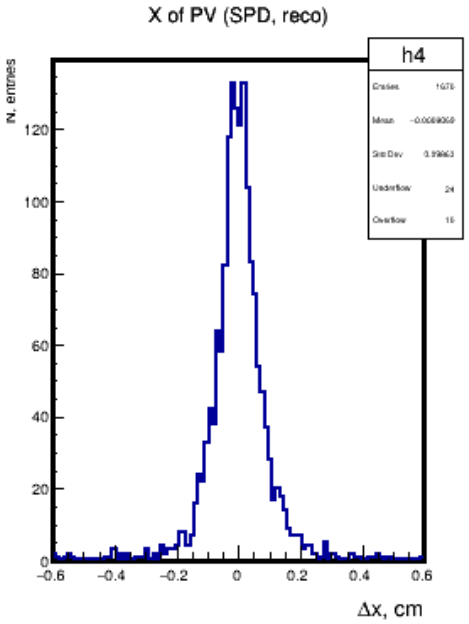
```
TVector3 pVtx;
if ( !fVertexFinder.FindPrimVertexCand1(fSelTrk,pVtx) ) continue;  // 1-D finder
if ( !fVertexFinder.FindPrimVertexCand3(fSelTrk,pVtx) ) continue;  // 3-D finder
```

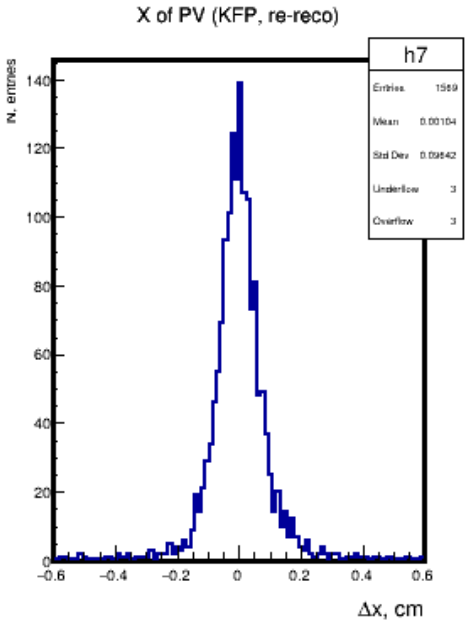
d) run primary vertex fitter (select method):

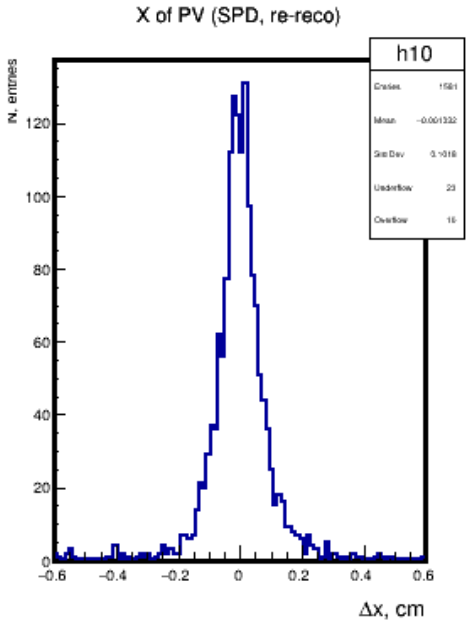
```
if ( !fVertexFitter.SpdFitPrimaryVertex(fSelTrk,pVtx) ) continue;      // CBM base
if ( !fVertexFitter.KFpFitPrimaryVertex(fSelTrk,pVtx) ) continue;      // KFParticle base

TVector3 rVtx = fVertexFitter.GetVertex();            // get vertex information

GetVertex(),
GetVx(),  GetVy(), GetVz(),
GetNTracks(), GetChi2(), GetNDF(),  GetChi2overNDF()
const TMatrixDSym& cov = fVertexFitter.GetCov();
```

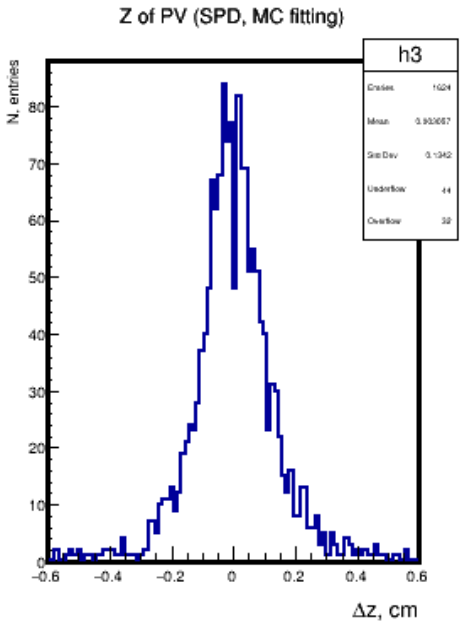
Primary vertex re-reconstruction

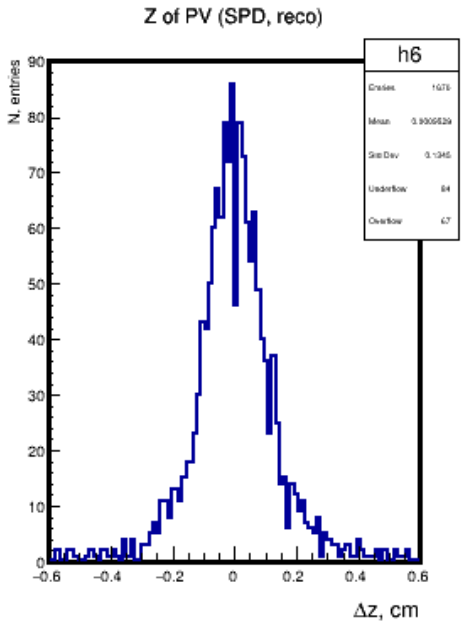


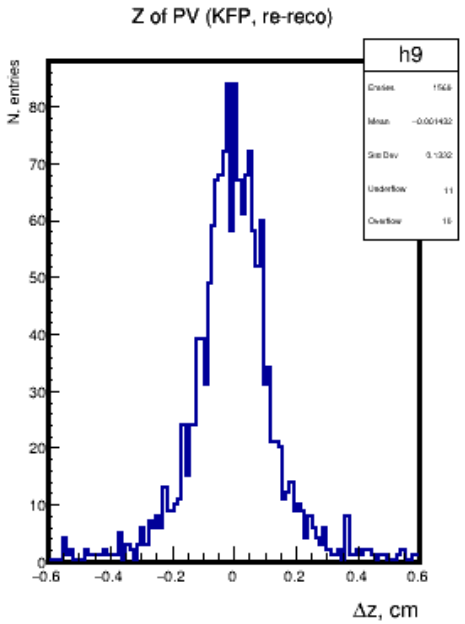


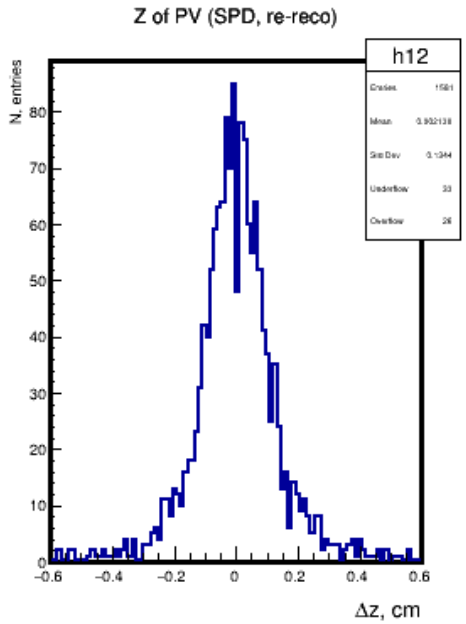












Output histos running of **PrimVertexAnalyzer.C** for MB without vertex detector

Using KFParticle for re-reconstruction of PV

PrimVertexKFpAnalyzer.C – macro with example how to use KFParticle in analyzer for re-reconstruction of primary vertex:

a) select tracks;
b) init primary vertex finder class:

```
SpdRCVertexFinder fVertexFinder;
fVertexFinder.Init();
```

c) run primary vertex finder (select method):

```
TVector3 pVtx;
if ( !fVertexFinder.FindPrimVertexCand1(fSelTrk,pVtx) ) continue;     // 1-D finder
if ( !fVertexFinder.FindPrimVertexCand3(fSelTrk,pVtx) ) continue;     // 3-D finder
```

d) extrapolate track to position of PV estimation and form KFParticle sample;

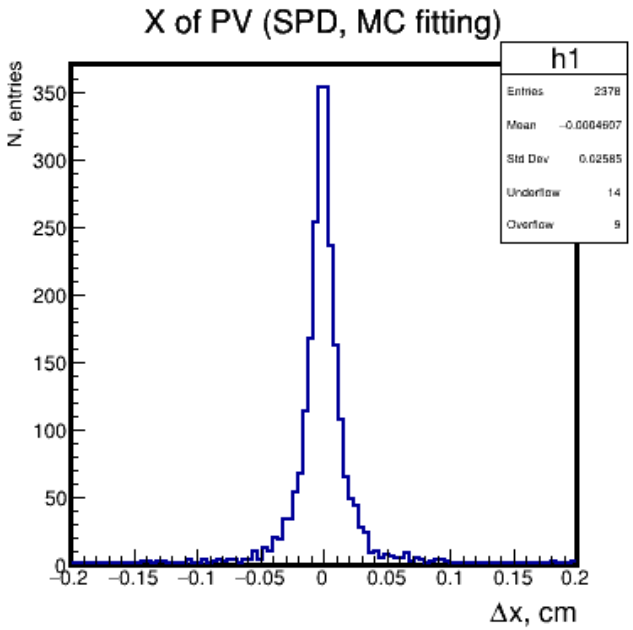
e) run reconstruction of primary vertex with KFParticle:

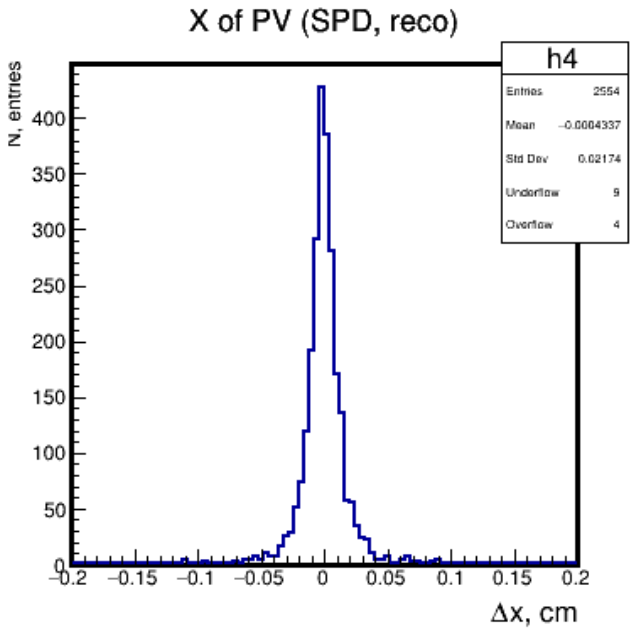
```
KFParticle primVtx;
Int_t nDaughters = fKFparticles.size();
const KFParticle* vDaughters[nDaughters];

for (Int_t i=0; i< fKFparticles.size(); i++) vDaughters[i] = &fKFparticles[i];

primVtx.Construct(vDaughters,nDaughters,0,-1);
```

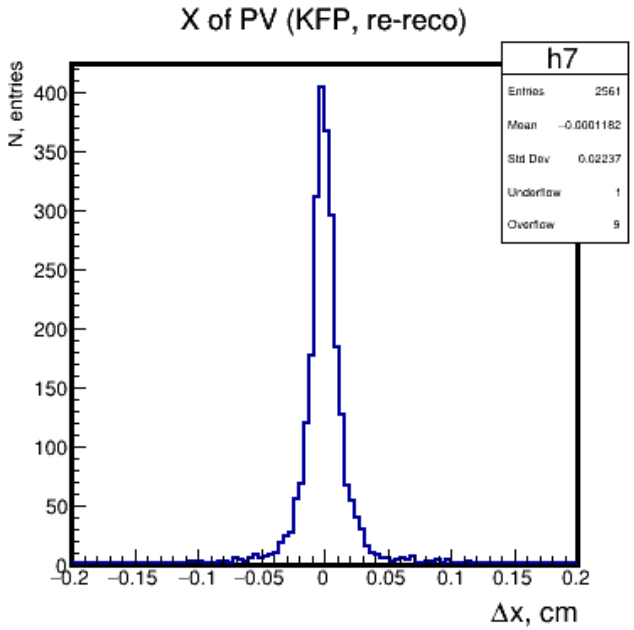
f) improve primary vertex by removing “bad” track:

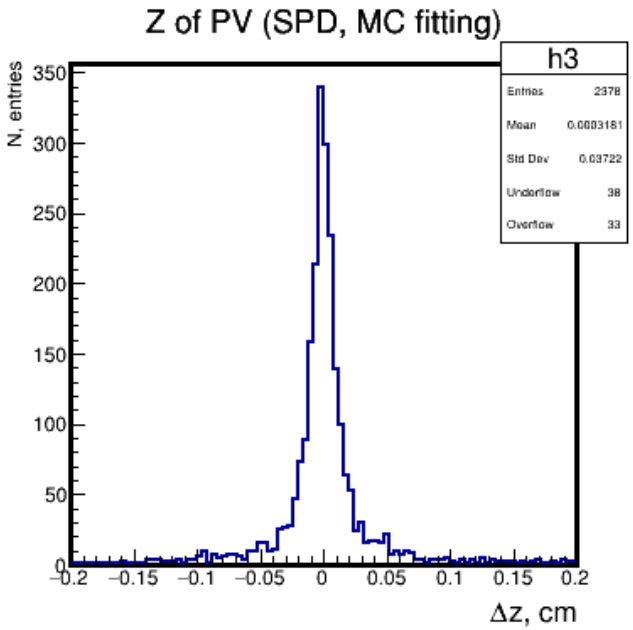
```
chi2 = fKFparticles[it].GetDeviationFromVertex(primVtx);        // check chi2
fKFparticles[it].SubtractFromVertex(primVtx);                   // remove “bad” track
```

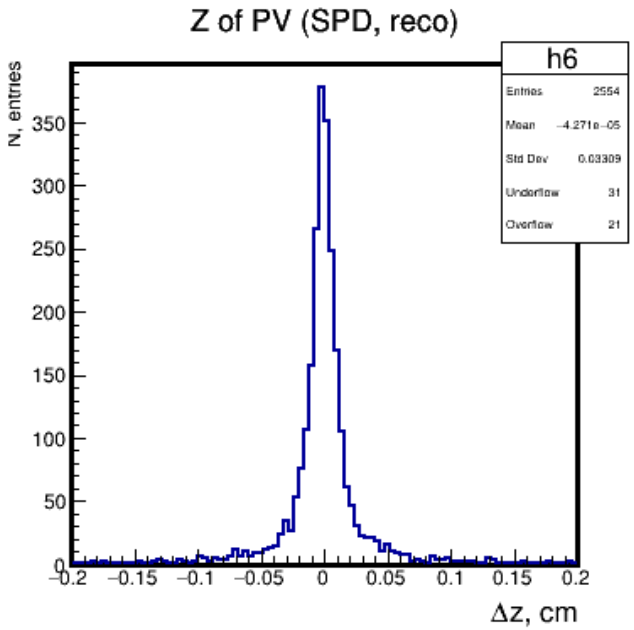
Primary vertex re-reconstruction (KFParticle)

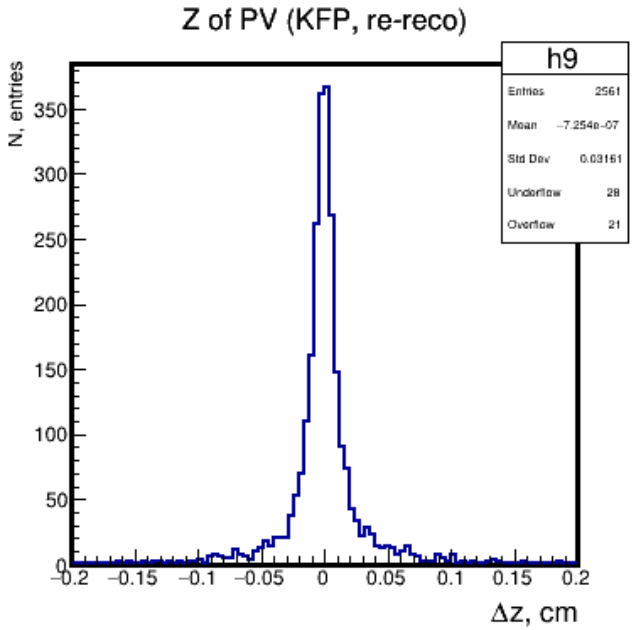












Output histos running of **PrimVertexKFpAnalyzer.C** for MB and with MAPS vertex detector.

Summary

1. New classes **SpdRCVertexFinder** and **SpdRCPrimVertexFinder** are added in SPDroot **(development branch).**

2. It includes two primary vertex finding algorithms and also two primary vertex fitting algorithms.

3. These algorithms (finding and fitting) can be used with the standard way in running job.

4. These algorithms (primary vertex reconstruction) can be also used in user analyzer for re-reconstruction of primary vertex (if it is necessary).