Vertexing update

Franco Bedeschi, INFN

FCC Physics Performance

March, 2024

OUTLINE

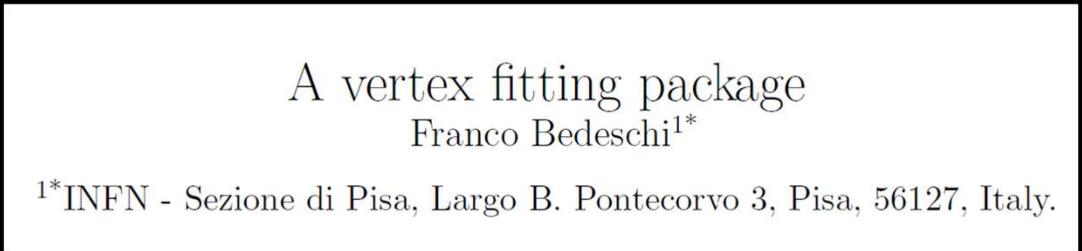
- Vertex fitting
  - Functionality
  - Examples
- Adding neutrals
  - Examples

Introduction

- Basic features of vertex fitting package
  - Fully contained in 3 classes (included in DELPHES):
    - TrkUtil, VertexFit, VertexMore
    - Only dependencies are the ROOT libraries
  - Functionality:
    - Vertex fit from list of track parameters and covariance matrices
      - Tracks can be added or removed incrementally from fit
    - Can include external vertex constraint (e.g. beam spot)
    - Re-calculation of track parameters and momenta after fit with associated error matrices
    - Mass constraints can be applied to improve resolution
    - Charged vertex can be treated as track to be used to fit chain decays
    - NOW CAN ALSO USE NEUTRAL VERTICES IN CHAIN DECAYS

Vertex fit note

- Note available describing methods and use



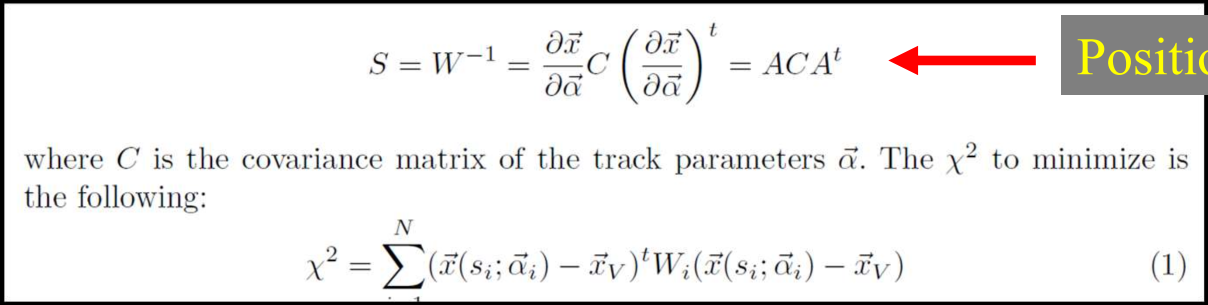
A vertex fitting package

Franco Bedeschi[1*]

[1*] INFN - Sezione di Pisa, Largo B. Pontecorvo 3, Pisa, 56127, Italy.

Vertex fit (no steering)

- Used to get first estimate of vertex position

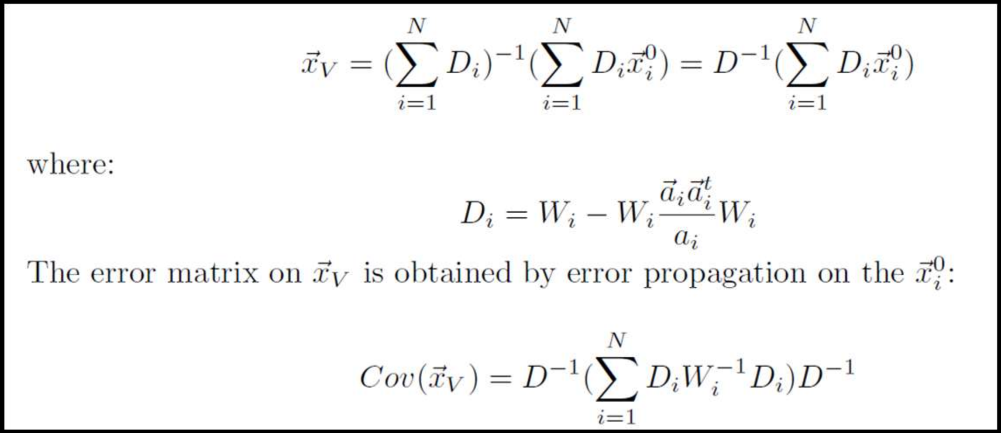


$$S = W^{-1} = \frac{\partial\vec{x}}{\partial\vec{\alpha}} C \left(\frac{\partial\vec{x}}{\partial\vec{\alpha}}\right)^t = ACA^t$$

where C is the covariance matrix of the track parameters $\vec{\alpha}$. The χ^2 to minimize is the following:

$$\chi^2 = \sum^{N} (\vec{x}(s_i;\vec{\alpha}_i) - \vec{x}_V)^t W_i (\vec{x}(s_i;\vec{\alpha}_i) - \vec{x}_V) \qquad (1)$$

Position error



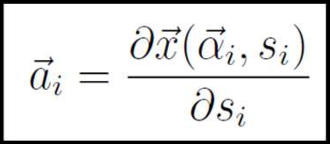
$$\vec{x}_V = (\sum_{i=1}^{N} D_i)^{-1} (\sum_{i=1}^{N} D_i \vec{x}_i^0) = D^{-1} (\sum_{i=1}^{N} D_i \vec{x}_i^0)$$

where:

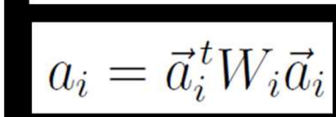
$$D_i = W_i - W_i \frac{\vec{a}_i \vec{a}_i^t}{a_i} W_i$$

The error matrix on $\vec{x}_V$ is obtained by error propagation on the $\vec{x}_i^0$:

$$Cov(\vec{x}_V) = D^{-1} (\sum_{i=1}^{N} D_i W_i^{-1} D_i) D^{-1}$$



$$\vec{a}_i = \frac{\partial\vec{x}(\vec{\alpha}_i, s_i)}{\partial s_i}$$



$$a_i = \vec{a}_i^t W_i \vec{a}_i$$

Vertex Fit (w/ steering)

- Used to get full vertex information

$$\chi^2 = \sum_{i=1}^{N} (\vec{\alpha}_i - \vec{\alpha}_i^{\,0})^t C_i^{-1} (\vec{\alpha}_i - \vec{\alpha}_i^{\,0}) + 2(\vec{x}(s_i, \vec{\alpha}_i) - \vec{x}_V)^t \vec{\lambda}_i\}$$

  - Solution very similar to no steering version

$$\vec{x}_V = \left(\sum_{i=1}^{N} D_i\right)^{-1} \left(\sum_{i=1}^{N} D_i(\vec{x}_i^{\,0} + A_i\,\delta\vec{\alpha}_i^{\,0})\right) = D^{-1} \left(\sum_{i=1}^{N} D_i(\vec{x}_i^{\,0} + \boxed{A_i\,\delta\vec{\alpha}_i^{\,0})}\right)$$

$$\begin{aligned} C_V = Cov(\vec{x}_V) &= D^{-1} \left(\textstyle\sum_{ij} D_i A_i < \delta\vec{\alpha}_i^{\,0} \delta\vec{\alpha}_j^{\,0t} > A_i^t D_j\right) D^{-1} \\ &= D^{-1} \left(\textstyle\sum_i D_i W_i^{-1} D_i\right) D^{-1} \end{aligned}$$

Vertex Fit (w/ steering)

- Get updated parameters and their errors

$$\begin{aligned}\vec{\alpha}_i &= \vec{\alpha}_i^0 - C_i\, A_i^t \vec{\lambda}_i \\ &= \vec{\alpha}_i^0 - C_i\, A_i^t D_i \sum_{k=1}^{N} (I\delta_{ik} - D^{-1} D_k)(\vec{x}_k^0 + A_k\, \delta\vec{\alpha}_k^0)\end{aligned}$$

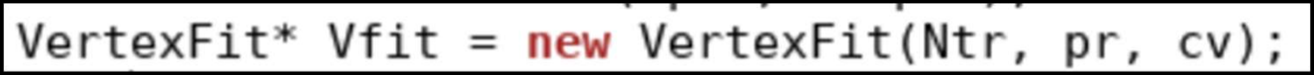
$$M_k^i = \frac{\partial \vec{\alpha}_i}{\partial \vec{\alpha}_k^0}$$

$$< \delta\vec{\alpha}_i \delta\vec{\alpha}_j^t > = \sum_{k=1}^{N} M_k^i < \delta\vec{\alpha}_i^0 \delta\vec{\alpha}_j^{0\,t} > M_k^{j\,t} = \sum_{k=1}^{N} M_k^i C_k (M_k^j)^t$$

- From this derive momenta at vertex, vertex track parameters and their covariance matices

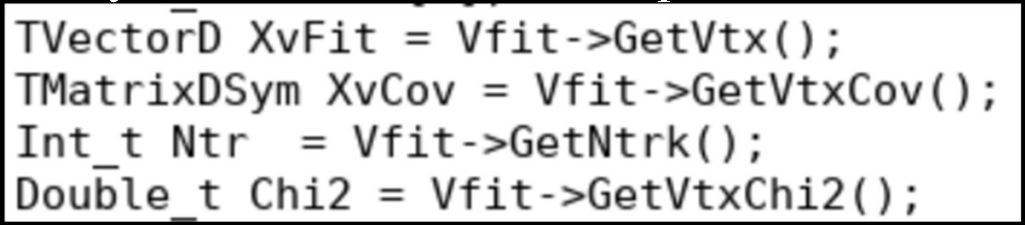
Example code

- ❖ Basic vertexing from list of tracks
  - ➢ pr = list of track parameters
  - ➢ cv = list of associated covariance matrices

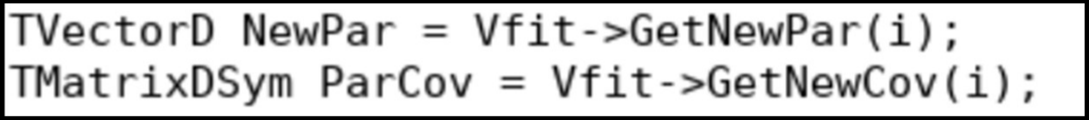


```
VertexFit* Vfit = new VertexFit(Ntr, pr, cv);
```

  - ➢ Many info available from Vtx pointer

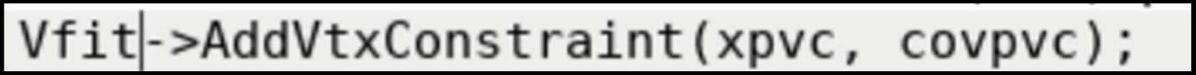


```
TVectorD XvFit = Vfit->GetVtx();
TMatrixDSym XvCov = Vfit->GetVtxCov();
Int_t Ntr  = Vfit->GetNtrk();
Double_t Chi2 = Vfit->GetVtxChi2();
```



```
TVectorD NewPar = Vfit->GetNewPar(i);
TMatrixDSym ParCov = Vfit->GetNewCov(i);
```

  - ➢ Add exernal vertex constraint (useful for primary vertex find)

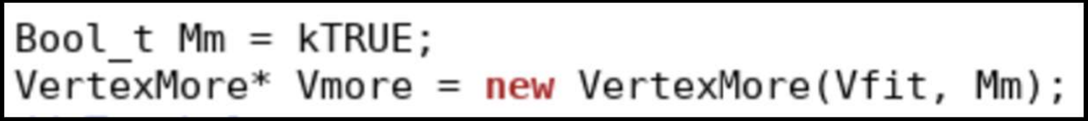


```
Vfit->AddVtxConstraint(xpvc, covpvc);
```

Additional info

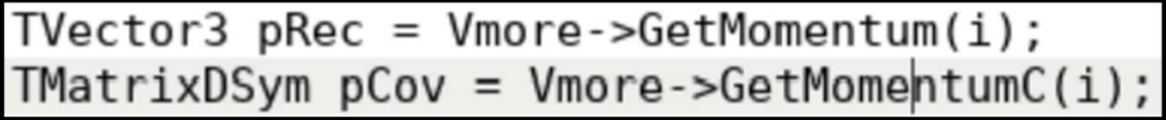
❖ Additional functionality provided by VertexMore

- ➢ Pass vertex pointer to Vertex more (select mm or meters)



```
Bool_t Mm = kTRUE;
VertexMore* Vmore = new VertexMore(Vfit, Mm);
```

- ➢ Extract additional information (e.g. track momentum):

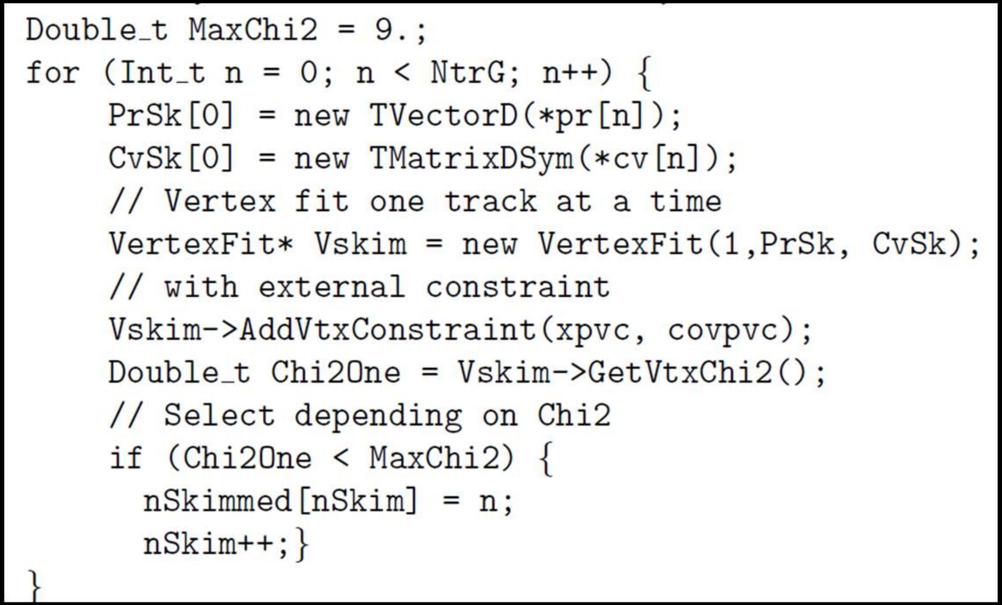


```
TVector3 pRec = Vmore->GetMomentum(i);
TMatrixDSym pCov = Vmore->GetMomentumC(i);
```

- ➢ Additional info Vertex total momentum and error matrix, vertex track parameters

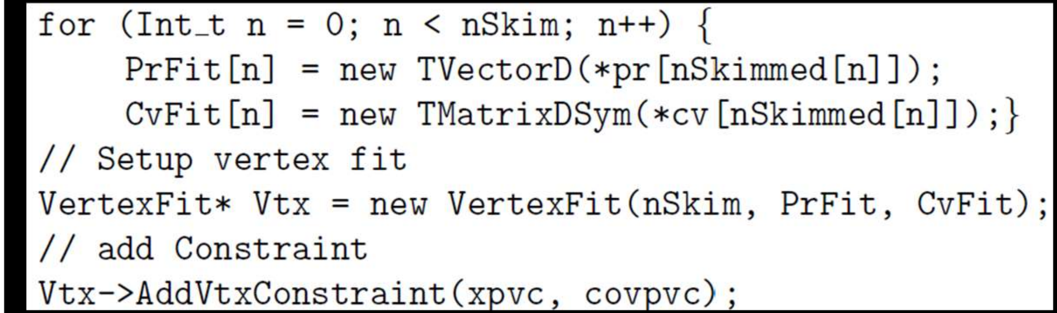
Primary vertex finder (Z→$b\bar{b}$)

- First skim: compare with external estimate



```
Double_t MaxChi2 = 9.;
for (Int_t n = 0; n < NtrG; n++) {
     PrSk[0] = new TVectorD(*pr[n]);
     CvSk[0] = new TMatrixDSym(*cv[n]);
     // Vertex fit one track at a time
     VertexFit* Vskim = new VertexFit(1,PrSk, CvSk);
     // with external constraint
     Vskim->AddVtxConstraint(xpvc, covpvc);
     Double_t Chi2One = Vskim->GetVtxChi2();
     // Select depending on Chi2
     if (Chi2One < MaxChi2) {
       nSkimmed[nSkim] = n;
       nSkim++;}
}
```

- Second skim: remove large Chi2 tracks
  - Fit 1st skim

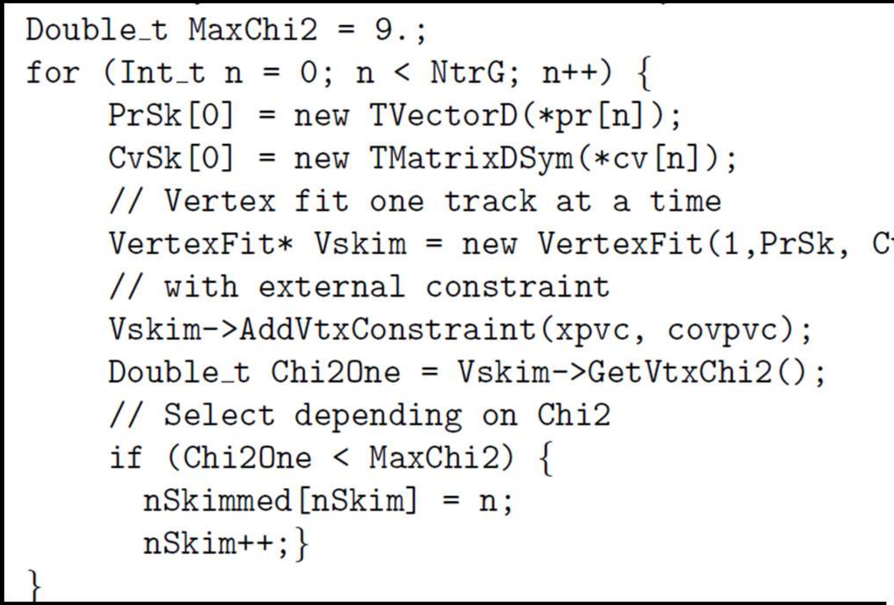


```
for (Int_t n = 0; n < nSkim; n++) {
     PrFit[n] = new TVectorD(*pr[nSkimmed[n]]);
     CvFit[n] = new TMatrixDSym(*cv[nSkimmed[n]]);}
// Setup vertex fit
VertexFit* Vtx = new VertexFit(nSkim, PrFit, CvFit);
// add Constraint
Vtx->AddVtxConstraint(xpvc, covpvc);
```

  - Remove large Chi2 tracks

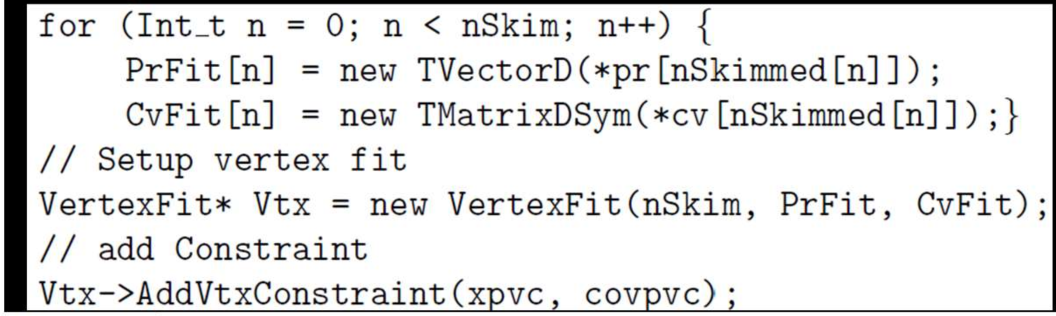
Primary vertex finder (Z→$b\bar{b}$)

- First skim: compare with external estimate

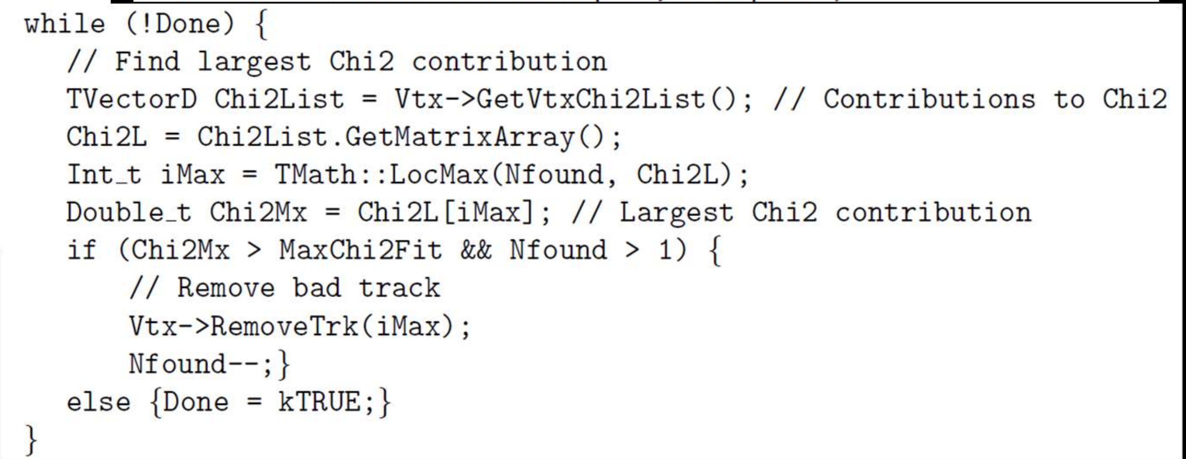


```
Double_t MaxChi2 = 9.;
for (Int_t n = 0; n < NtrG; n++) {
     PrSk[0] = new TVectorD(*pr[n]);
     CvSk[0] = new TMatrixDSym(*cv[n]);
     // Vertex fit one track at a time
     VertexFit* Vskim = new VertexFit(1,PrSk, CvSk);
     // with external constraint
     Vskim->AddVtxConstraint(xpvc, covpvc);
     Double_t Chi2One = Vskim->GetVtxChi2();
     // Select depending on Chi2
     if (Chi2One < MaxChi2) {
       nSkimmed[nSkim] = n;
       nSkim++;}
}
```

- Second skim: remove large Chi2 tracks
  - Fit 1st skim



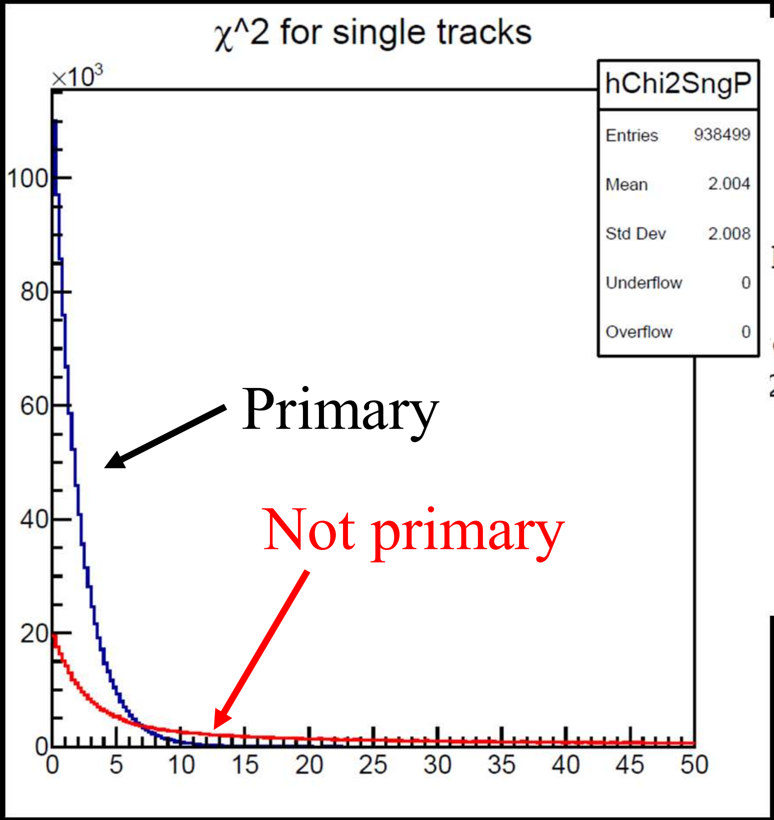
```
for (Int_t n = 0; n < nSkim; n++) {
     PrFit[n] = new TVectorD(*pr[nSkimmed[n]]);
     CvFit[n] = new TMatrixDSym(*cv[nSkimmed[n]]);}
// Setup vertex fit
VertexFit* Vtx = new VertexFit(nSkim, PrFit, CvFit);
// add Constraint
Vtx->AddVtxConstraint(xpvc, covpvc);
```



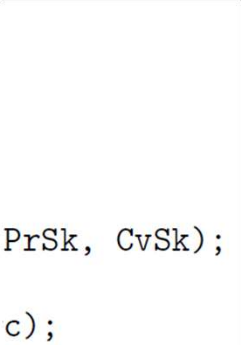
```
while (!Done) {
   // Find largest Chi2 contribution
   TVectorD Chi2List = Vtx->GetVtxChi2List(); // Contributions to Chi2
   Chi2L = Chi2List.GetMatrixArray();
   Int_t iMax = TMath::LocMax(Nfound, Chi2L);
   Double_t Chi2Mx = Chi2L[iMax]; // Largest Chi2 contribution
   if (Chi2Mx > MaxChi2Fit && Nfound > 1) {
       // Remove bad track
       Vtx->RemoveTrk(iMax);
       Nfound--;}
   else {Done = kTRUE;}
}
```

Primary vertex finder (Z→$b\bar{b}$)

- First skim: compare with external estimate

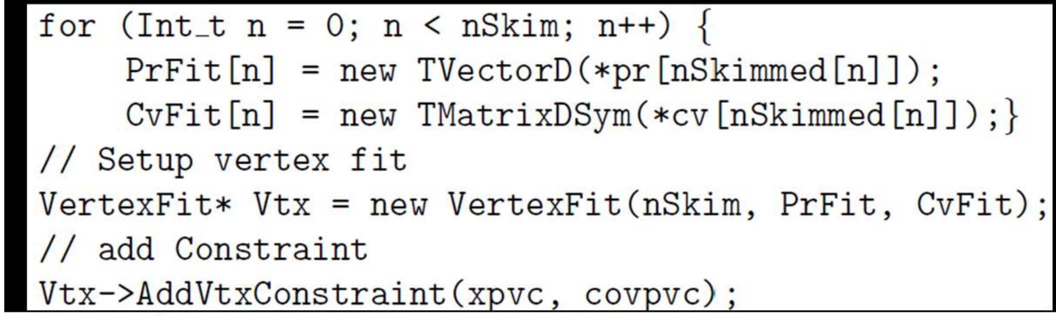


- Second skim: remove large Chi2 tracks
  - Fit 1st skim

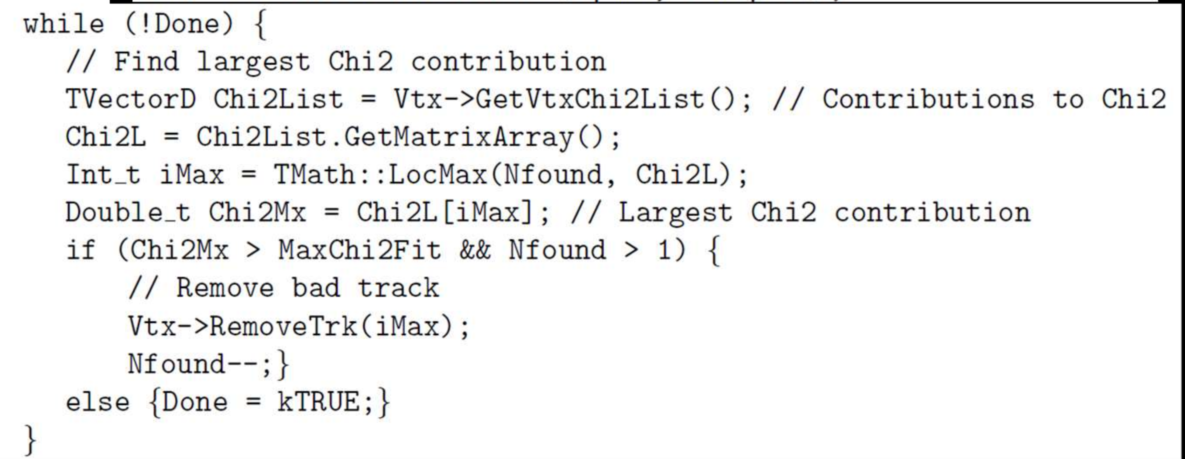


```
PrSk, CvSk);

c);
2();
```



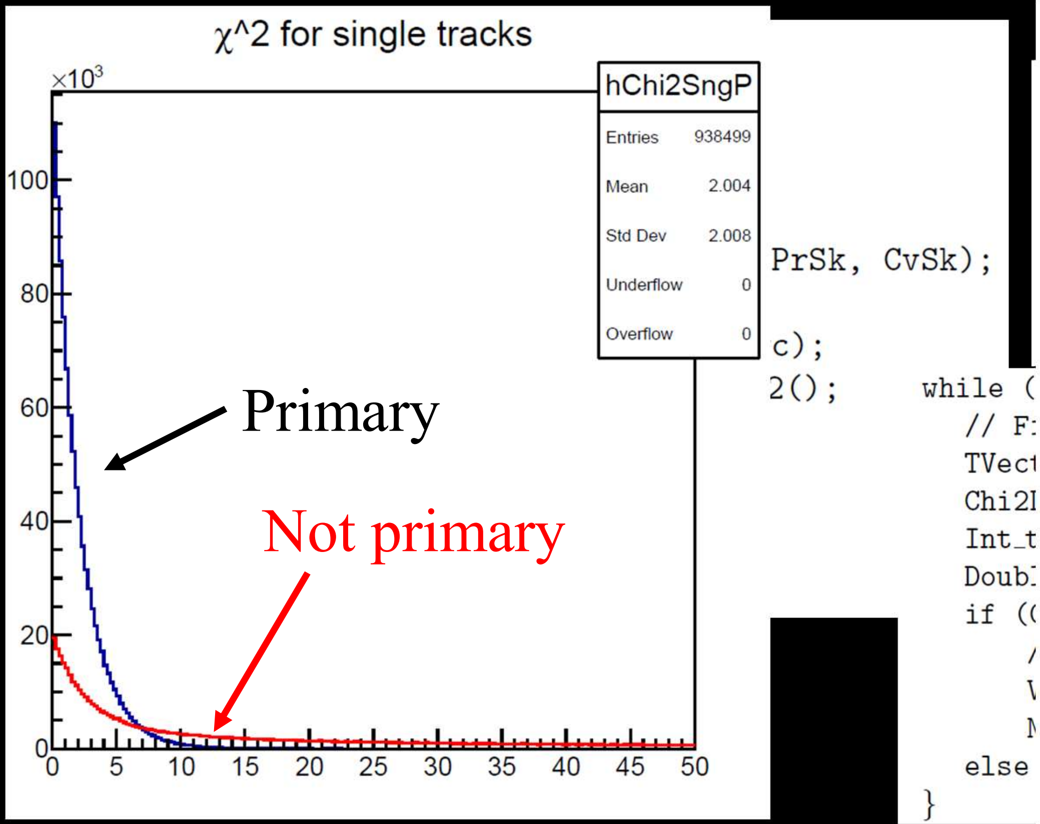
```
for (Int_t n = 0; n < nSkim; n++) {
     PrFit[n] = new TVectorD(*pr[nSkimmed[n]]);
     CvFit[n] = new TMatrixDSym(*cv[nSkimmed[n]]);}
// Setup vertex fit
VertexFit* Vtx = new VertexFit(nSkim, PrFit, CvFit);
// add Constraint
Vtx->AddVtxConstraint(xpvc, covpvc);
```



```
while (!Done) {
   // Find largest Chi2 contribution
   TVectorD Chi2List = Vtx->GetVtxChi2List(); // Contributions to Chi2
   Chi2L = Chi2List.GetMatrixArray();
   Int_t iMax = TMath::LocMax(Nfound, Chi2L);
   Double_t Chi2Mx = Chi2L[iMax]; // Largest Chi2 contribution
   if (Chi2Mx > MaxChi2Fit && Nfound > 1) {
       // Remove bad track
       Vtx->RemoveTrk(iMax);
       Nfound--;}
   else {Done = kTRUE;}
}
```

Primary vertex finder (Z→$b\bar{b}$)

❖ First skim: compare with external estimate

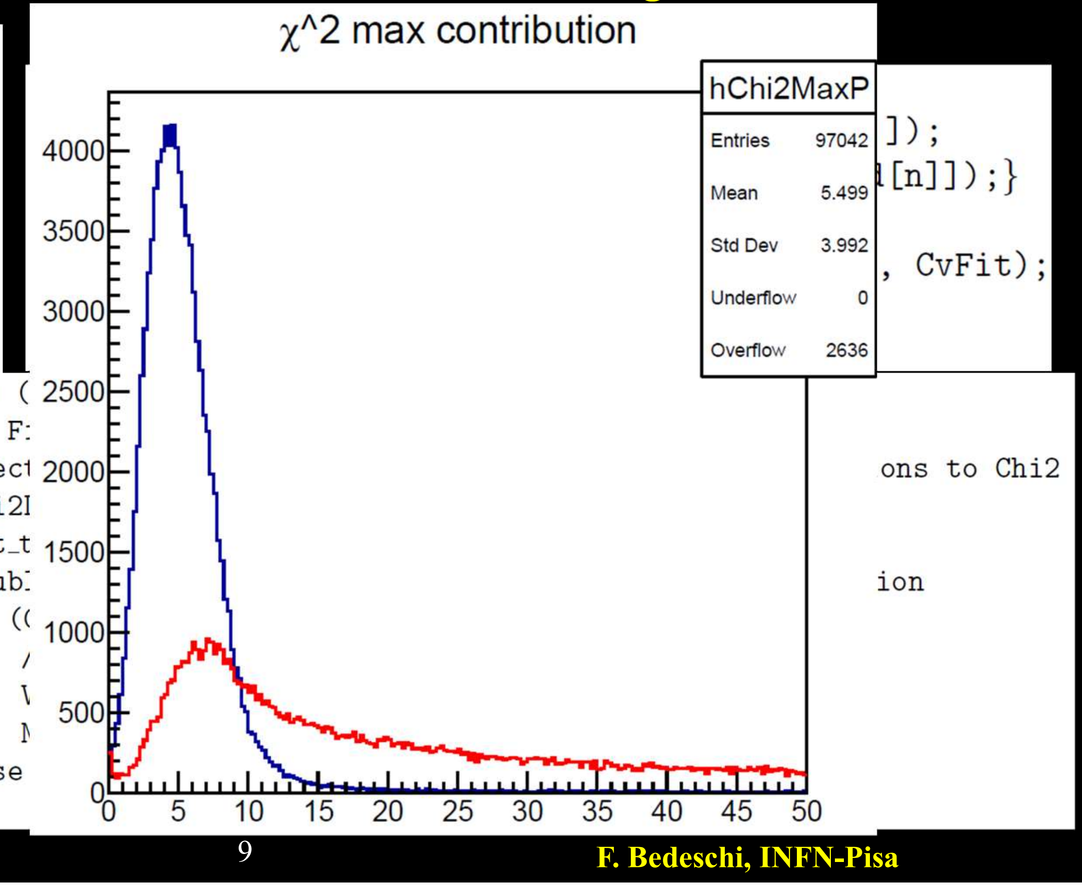


Primary

Not primary

```
PrSk, CvSk);
c);
2();      while (
            // F:
            TVect
            Chi2I
            Int_t
            Doubl
            if ((
                /
                \
                N
            else
          }
```

❖ Second skim: remove large Chi2 tracks



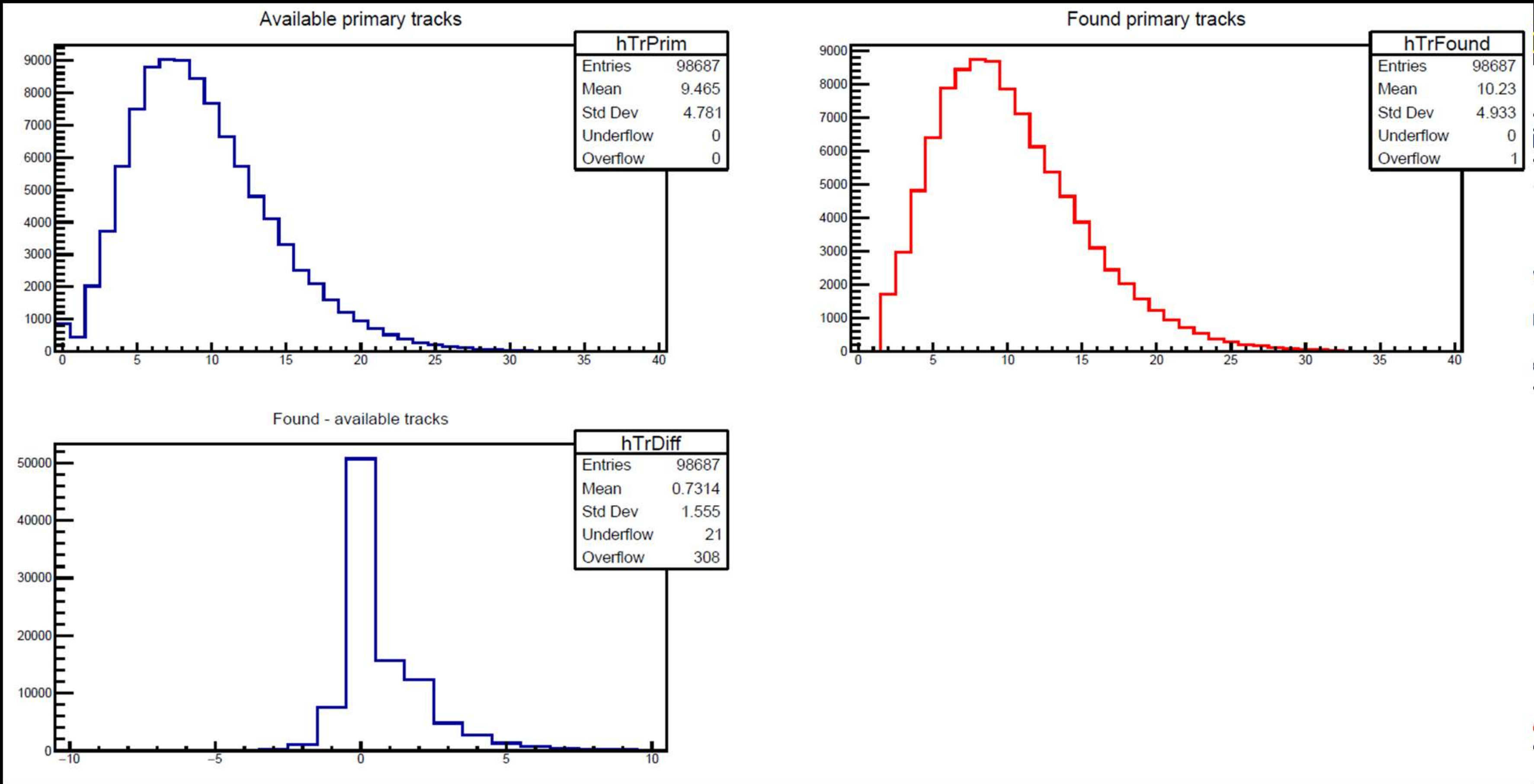
```
]);
l[n]]);}
, CvFit);

ons to Chi2

ion
```

Primary vertex finder (Z→$b\bar{b}$)

i2 tracks



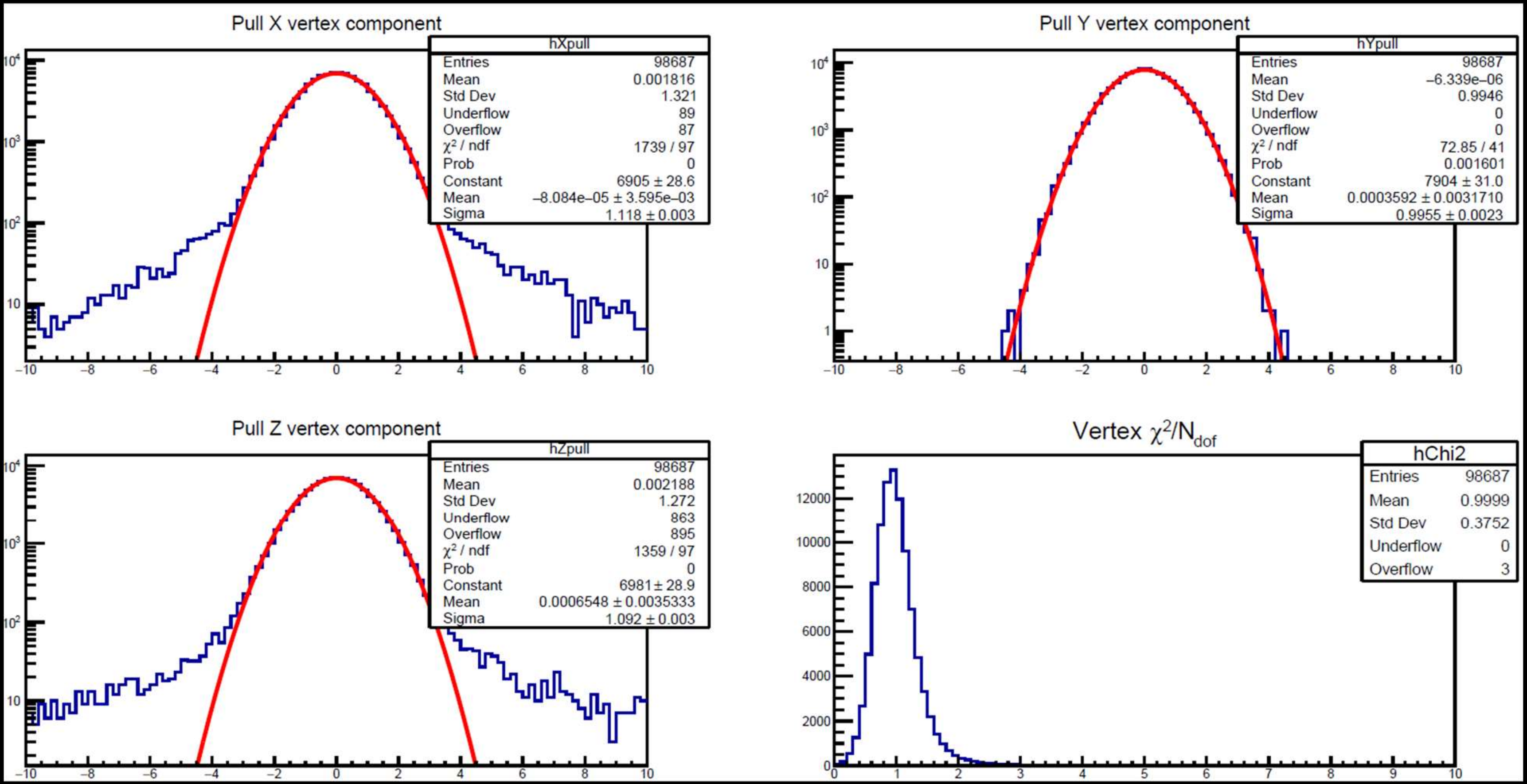
```
]);
d[n]]);}

, CvFit);
```

```
ons to Chi2

ion
```

Primary vertex finder pulls



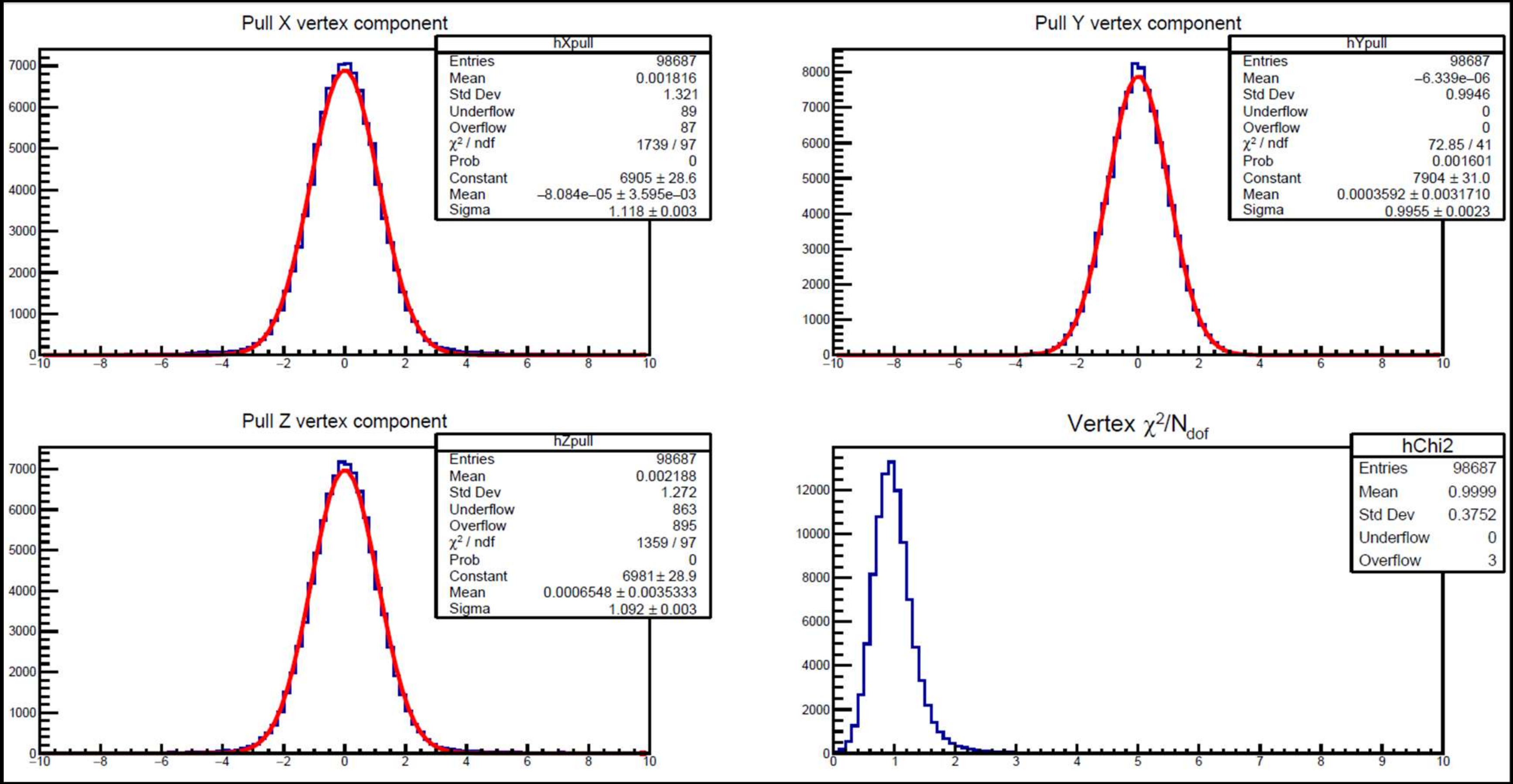
hXpull	
Entries	98687
Mean	0.001816
Std Dev	1.321
Underflow	89
Overflow	87
χ^2 / ndf	1739 / 97
Prob	0
Constant	6905 ± 28.6
Mean	−8.084e−05 ± 3.595e−03
Sigma	1.118 ± 0.003

hYpull	
Entries	98687
Mean	−6.339e−06
Std Dev	0.9946
Underflow	0
Overflow	0
χ^2 / ndf	72.85 / 41
Prob	0.001601
Constant	7904 ± 31.0
Mean	0.0003592 ± 0.0031710
Sigma	0.9955 ± 0.0023

hZpull	
Entries	98687
Mean	0.002188
Std Dev	1.272
Underflow	863
Overflow	895
χ^2 / ndf	1359 / 97
Prob	0
Constant	6981 ± 28.9
Mean	0.0006548 ± 0.0035333
Sigma	1.092 ± 0.003

hChi2	
Entries	98687
Mean	0.9999
Std Dev	0.3752
Underflow	0
Overflow	3

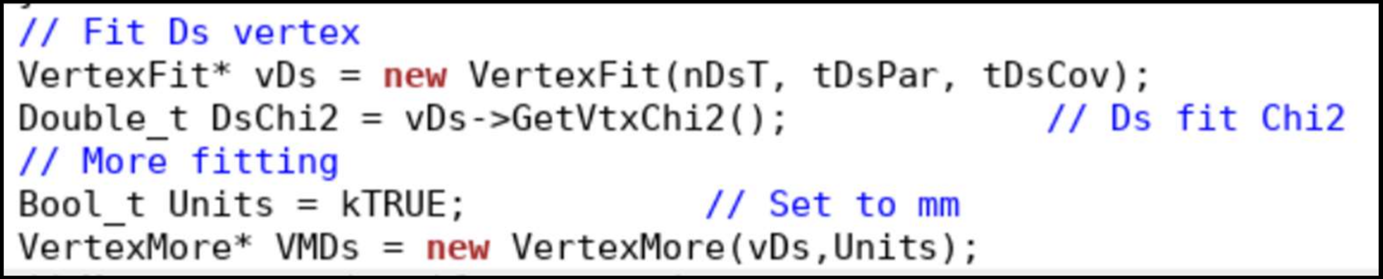
Primary vertex finder pulls



Example Bs→Ds π

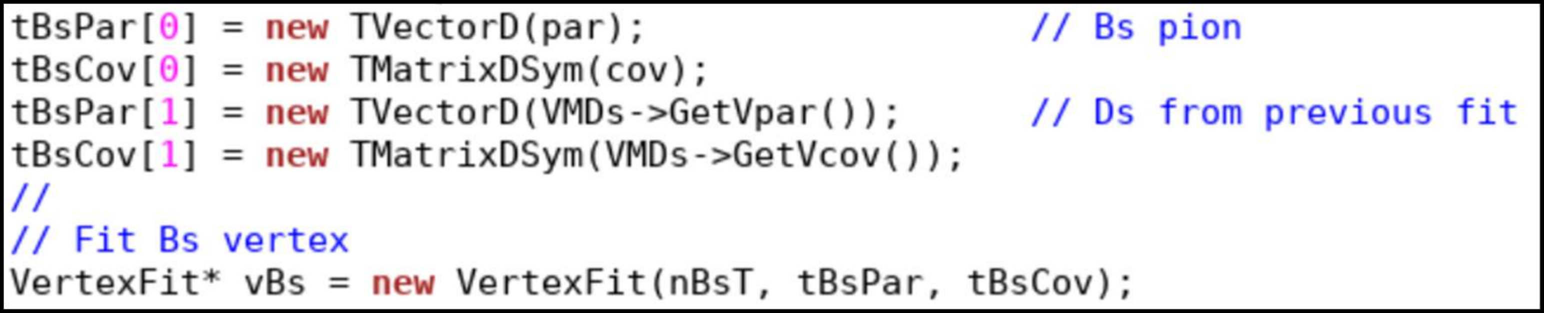
❖ Use Ds vertex track and pion track to vertex Bs

- Fit Ds → K K π



```
// Fit Ds vertex
VertexFit* vDs = new VertexFit(nDsT, tDsPar, tDsCov);
Double_t DsChi2 = vDs->GetVtxChi2();            // Ds fit Chi2
// More fitting
Bool_t Units = kTRUE;           // Set to mm
VertexMore* VMDs = new VertexMore(vDs,Units);
```

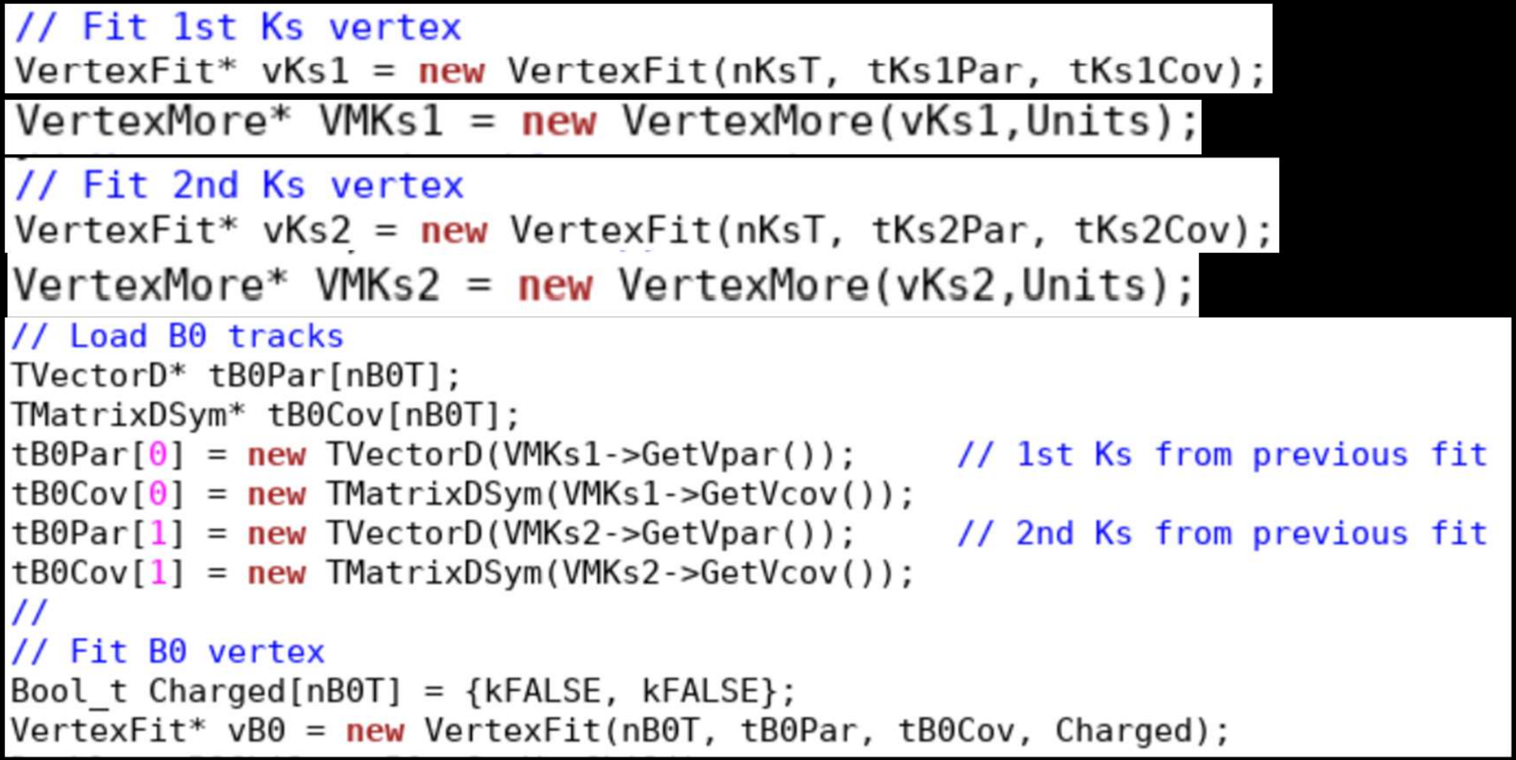
- Fit Bs



```
tBsPar[0] = new TVectorD(par);                  // Bs pion
tBsCov[0] = new TMatrixDSym(cov);
tBsPar[1] = new TVectorD(VMDs->GetVpar());      // Ds from previous fit
tBsCov[1] = new TMatrixDSym(VMDs->GetVcov());
//
// Fit Bs vertex
VertexFit* vBs = new VertexFit(nBsT, tBsPar, tBsCov);
```

Handling neutrals

- Tell VertexFit which tracks are neutral with additional Bool array (T = charged, F = neutral) ---- Example B_0 → KsKs



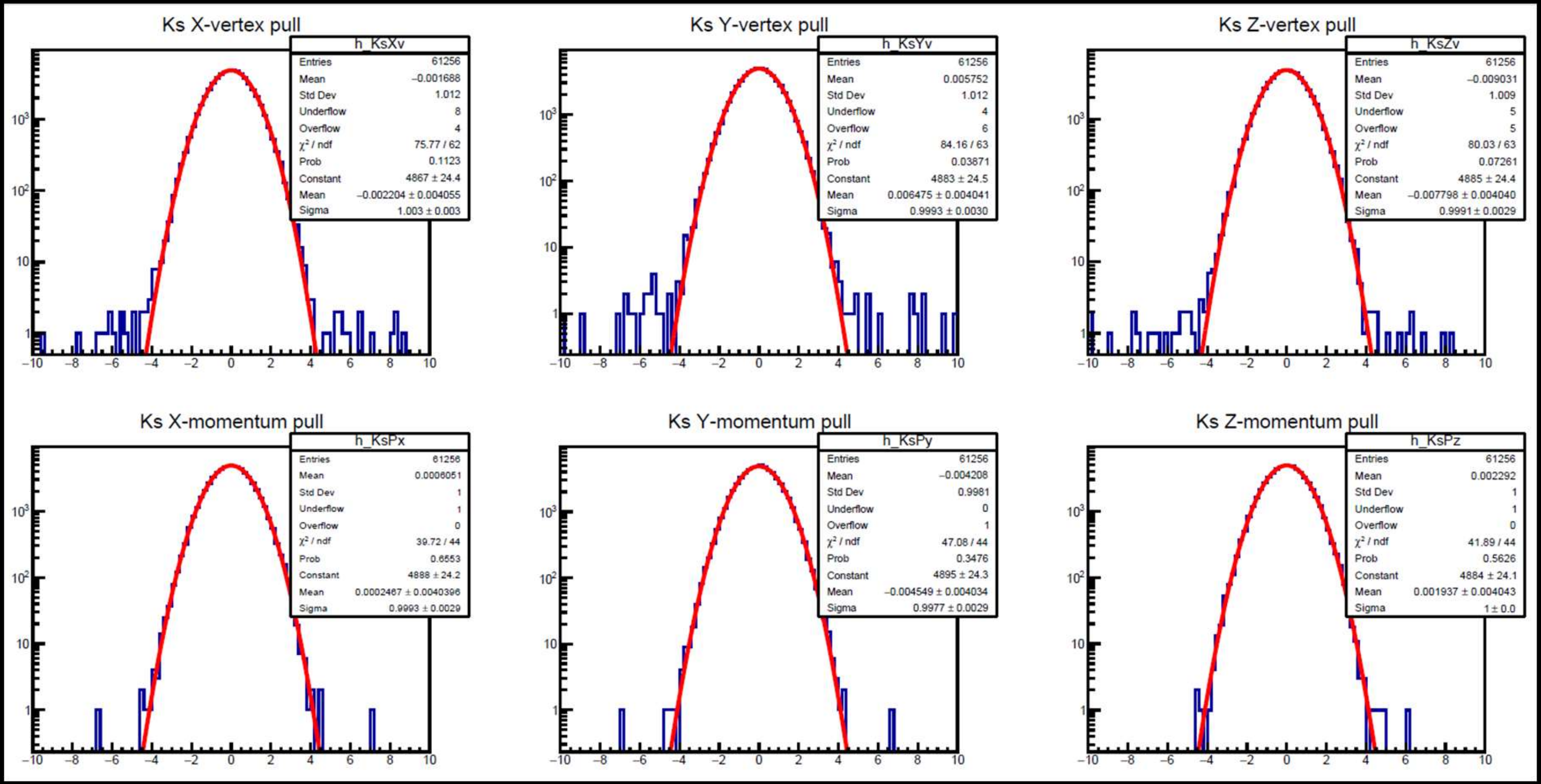
```
// Fit 1st Ks vertex
VertexFit* vKs1 = new VertexFit(nKsT, tKs1Par, tKs1Cov);
VertexMore* VMKs1 = new VertexMore(vKs1,Units);
// Fit 2nd Ks vertex
VertexFit* vKs2 = new VertexFit(nKsT, tKs2Par, tKs2Cov);
VertexMore* VMKs2 = new VertexMore(vKs2,Units);
// Load B0 tracks
TVectorD* tB0Par[nB0T];
TMatrixDSym* tB0Cov[nB0T];
tB0Par[0] = new TVectorD(VMKs1->GetVpar());      // 1st Ks from previous fit
tB0Cov[0] = new TMatrixDSym(VMKs1->GetVcov());
tB0Par[1] = new TVectorD(VMKs2->GetVpar());      // 2nd Ks from previous fit
tB0Cov[1] = new TMatrixDSym(VMKs2->GetVcov());
//
// Fit B0 vertex
Bool_t Charged[nB0T] = {kFALSE, kFALSE};
VertexFit* vB0 = new VertexFit(nB0T, tB0Par, tB0Cov, Charged);
```

$B_0 \rightarrow$ Ks Ks results

- Some problems previously seen now resolved

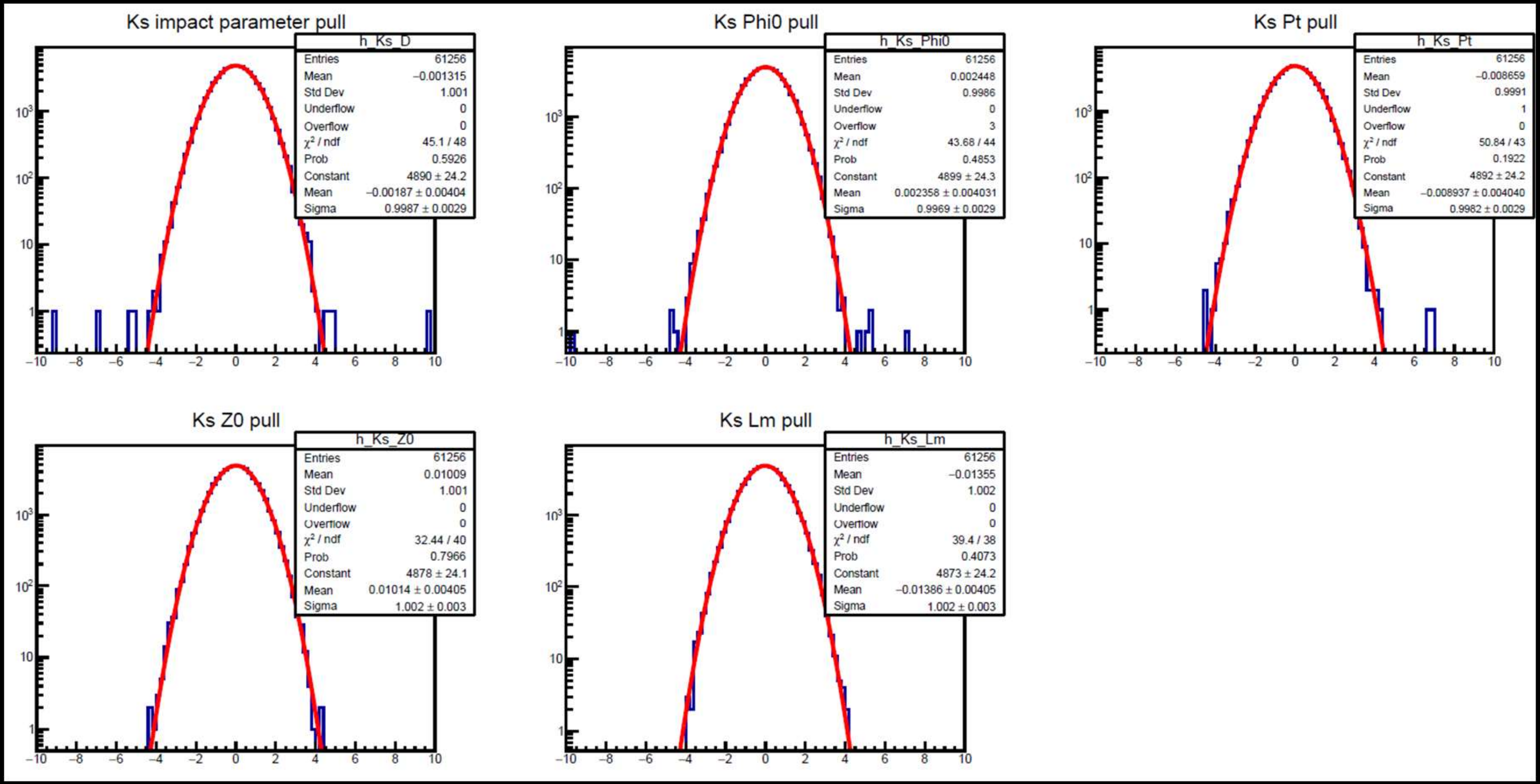
B_0 → Ks Ks results

- ❖ Ks vertex:



B_0 → Ks Ks results

- Ks parameters:



Ks impact parameter pull

h_Ks_D	
Entries	61256
Mean	−0.001315
Std Dev	1.001
Underflow	0
Overflow	0
χ^2 / ndf	45.1 / 48
Prob	0.5926
Constant	4890 ± 24.2
Mean	−0.00187 ± 0.00404
Sigma	0.9987 ± 0.0029

Ks Phi0 pull

h_Ks_Phi0	
Entries	61256
Mean	0.002448
Std Dev	0.9986
Underflow	0
Overflow	3
χ^2 / ndf	43.68 / 44
Prob	0.4853
Constant	4899 ± 24.3
Mean	0.002358 ± 0.004031
Sigma	0.9969 ± 0.0029

Ks Pt pull

h_Ks_Pt	
Entries	61256
Mean	−0.008659
Std Dev	0.9991
Underflow	1
Overflow	0
χ^2 / ndf	50.84 / 43
Prob	0.1922
Constant	4892 ± 24.2
Mean	−0.008937 ± 0.004040
Sigma	0.9982 ± 0.0029

Ks Z0 pull

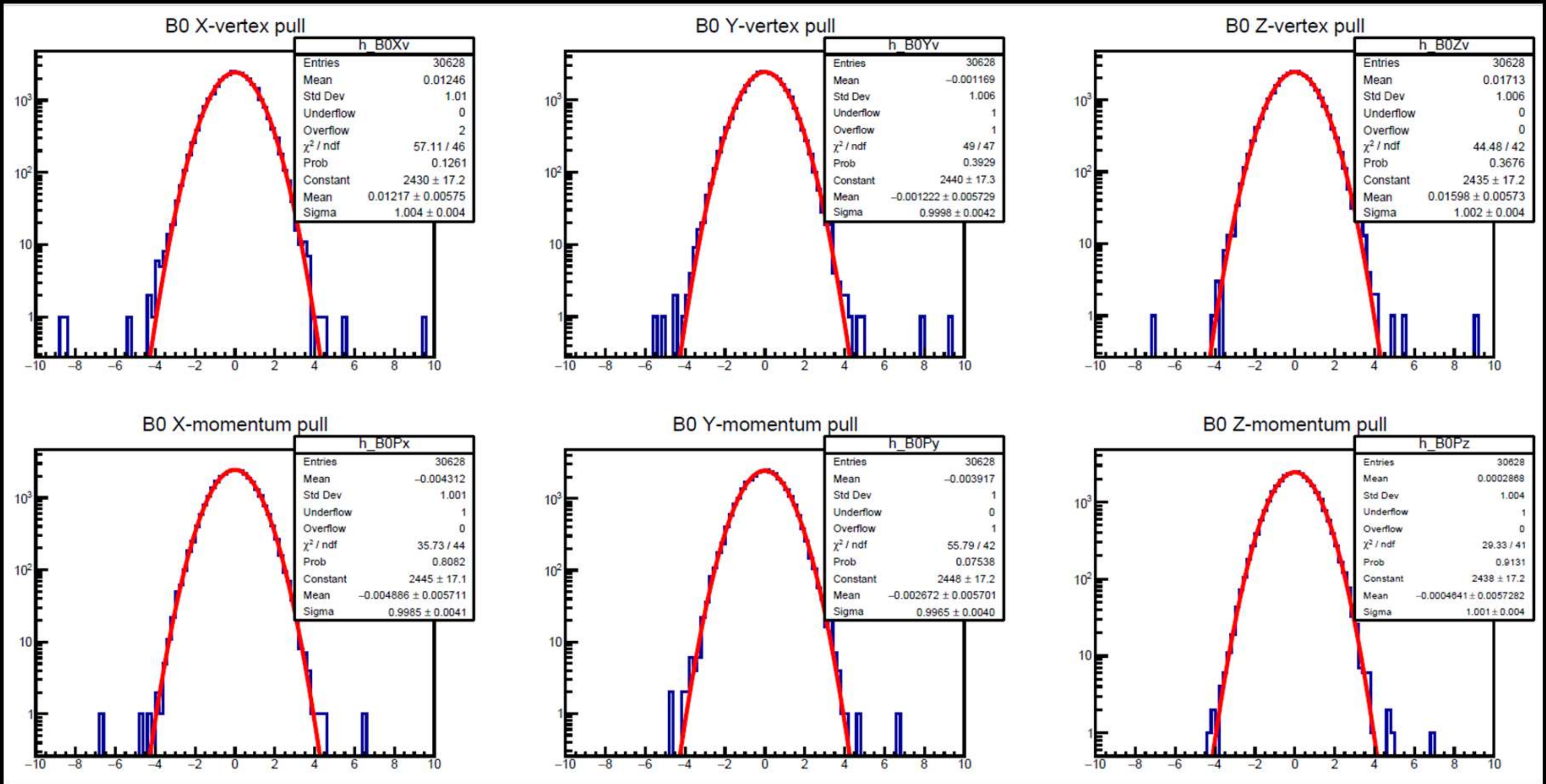
h_Ks_Z0	
Entries	61256
Mean	0.01009
Std Dev	1.001
Underflow	0
Overflow	0
χ^2 / ndf	32.44 / 40
Prob	0.7966
Constant	4878 ± 24.1
Mean	0.01014 ± 0.00405
Sigma	1.002 ± 0.003

Ks Lm pull

h_Ks_Lm	
Entries	61256
Mean	−0.01355
Std Dev	1.002
Underflow	0
Overflow	0
χ^2 / ndf	39.4 / 38
Prob	0.4073
Constant	4873 ± 24.2
Mean	−0.01386 ± 0.00405
Sigma	1.002 ± 0.003

B_0 → Ks Ks results

- ❖ B_0 Vertex:



h_B0Xv	
Entries	30628
Mean	0.01246
Std Dev	1.01
Underflow	0
Overflow	2
χ^2 / ndf	57.11 / 46
Prob	0.1261
Constant	2430 ± 17.2
Mean	0.01217 ± 0.00575
Sigma	1.004 ± 0.004

h_B0Yv	
Entries	30628
Mean	–0.001169
Std Dev	1.006
Underflow	1
Overflow	1
χ^2 / ndf	49 / 47
Prob	0.3929
Constant	2440 ± 17.3
Mean	–0.001222 ± 0.005729
Sigma	0.9998 ± 0.0042

h_B0Zv	
Entries	30628
Mean	0.01713
Std Dev	1.006
Underflow	0
Overflow	0
χ^2 / ndf	44.48 / 42
Prob	0.3676
Constant	2435 ± 17.2
Mean	0.01598 ± 0.00573
Sigma	1.002 ± 0.004

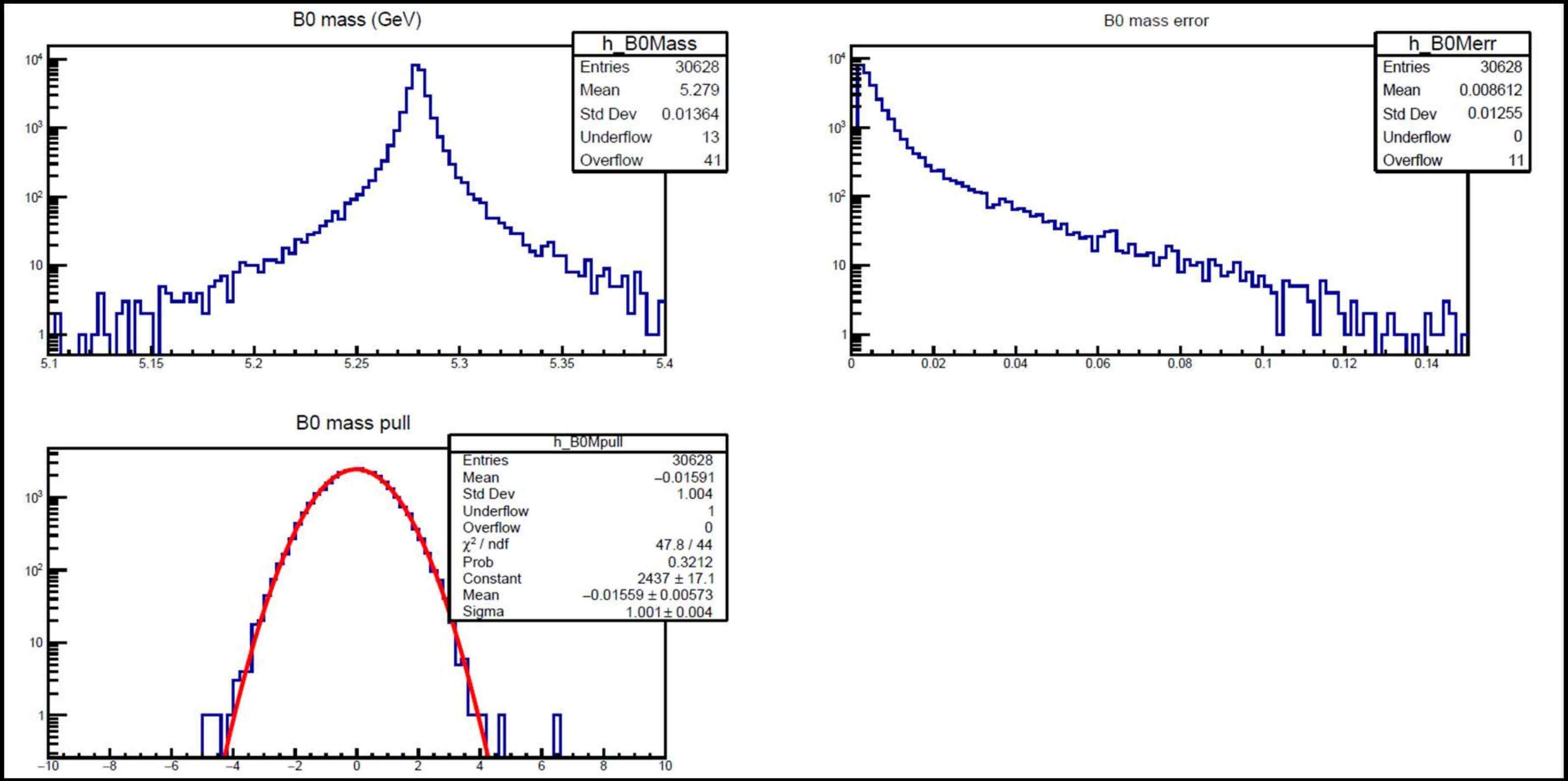
h_B0Px	
Entries	30628
Mean	–0.004312
Std Dev	1.001
Underflow	1
Overflow	0
χ^2 / ndf	35.73 / 44
Prob	0.8082
Constant	2445 ± 17.1
Mean	–0.004886 ± 0.005711
Sigma	0.9985 ± 0.0041

h_B0Py	
Entries	30628
Mean	–0.003917
Std Dev	1
Underflow	0
Overflow	1
χ^2 / ndf	55.79 / 42
Prob	0.07538
Constant	2448 ± 17.2
Mean	–0.002672 ± 0.005701
Sigma	0.9965 ± 0.0040

h_B0Pz	
Entries	30628
Mean	0.0002868
Std Dev	1.004
Underflow	1
Overflow	0
χ^2 / ndf	29.33 / 41
Prob	0.9131
Constant	2438 ± 17.2
Mean	–0.0004641 ± 0.0057282
Sigma	1.001 ± 0.004

B_0 → Ks Ks results

- B_0 mass:



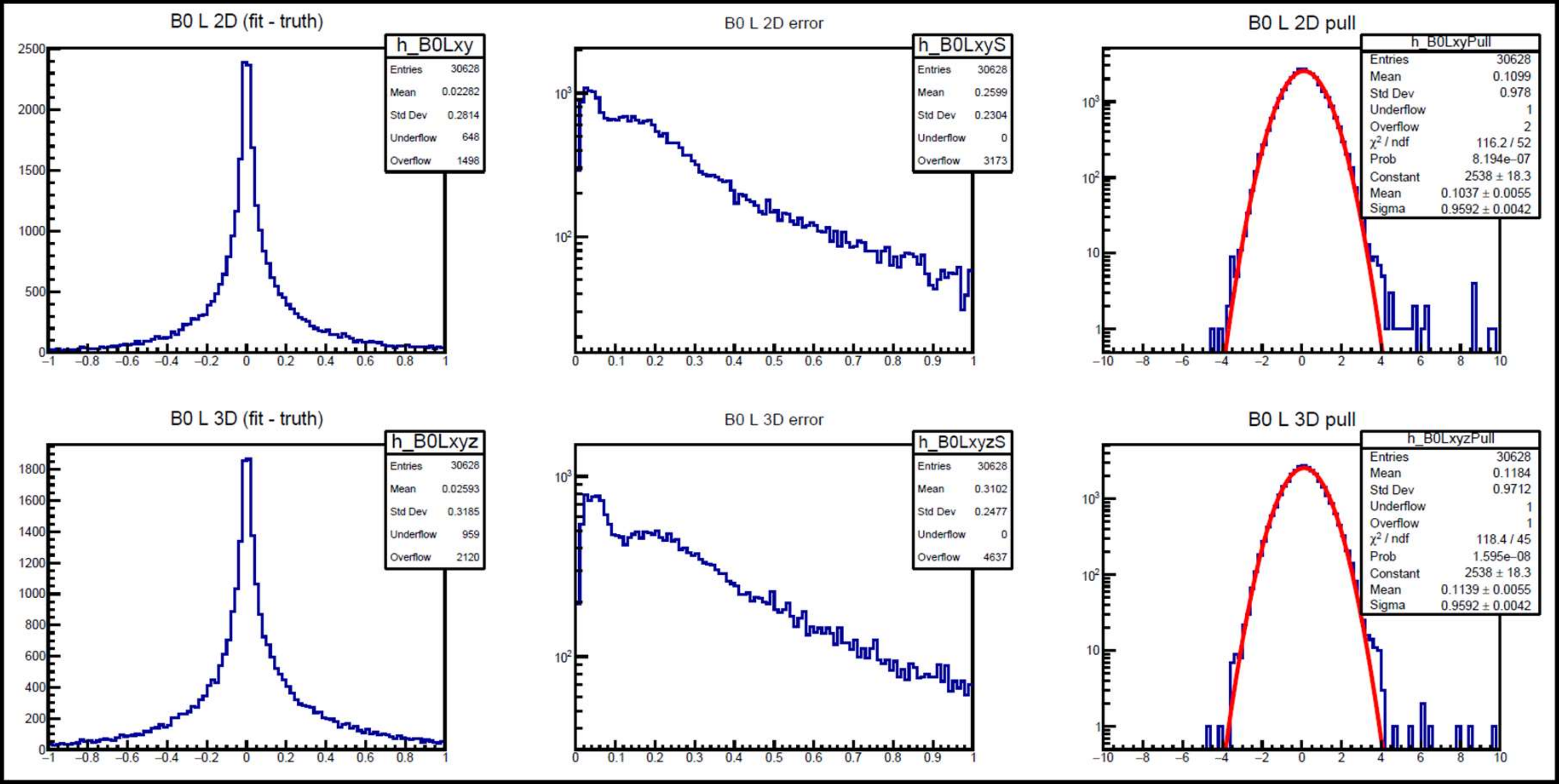
```
h_B0Mass
Entries      30628
Mean         5.279
Std Dev    0.01364
Underflow       13
Overflow        41
```

```
h_B0Merr
Entries      30628
Mean      0.008612
Std Dev    0.01255
Underflow        0
Overflow        11
```

```
h_B0Mpull
Entries                30628
Mean                -0.01591
Std Dev                1.004
Underflow                  1
Overflow                   0
χ² / ndf           47.8 / 44
Prob                  0.3212
Constant         2437 ± 17.1
Mean      -0.01559 ± 0.00573
Sigma          1.001 ± 0.004
```

$B_0 \rightarrow$ Ks Ks results

- B_0 flight:



Summary

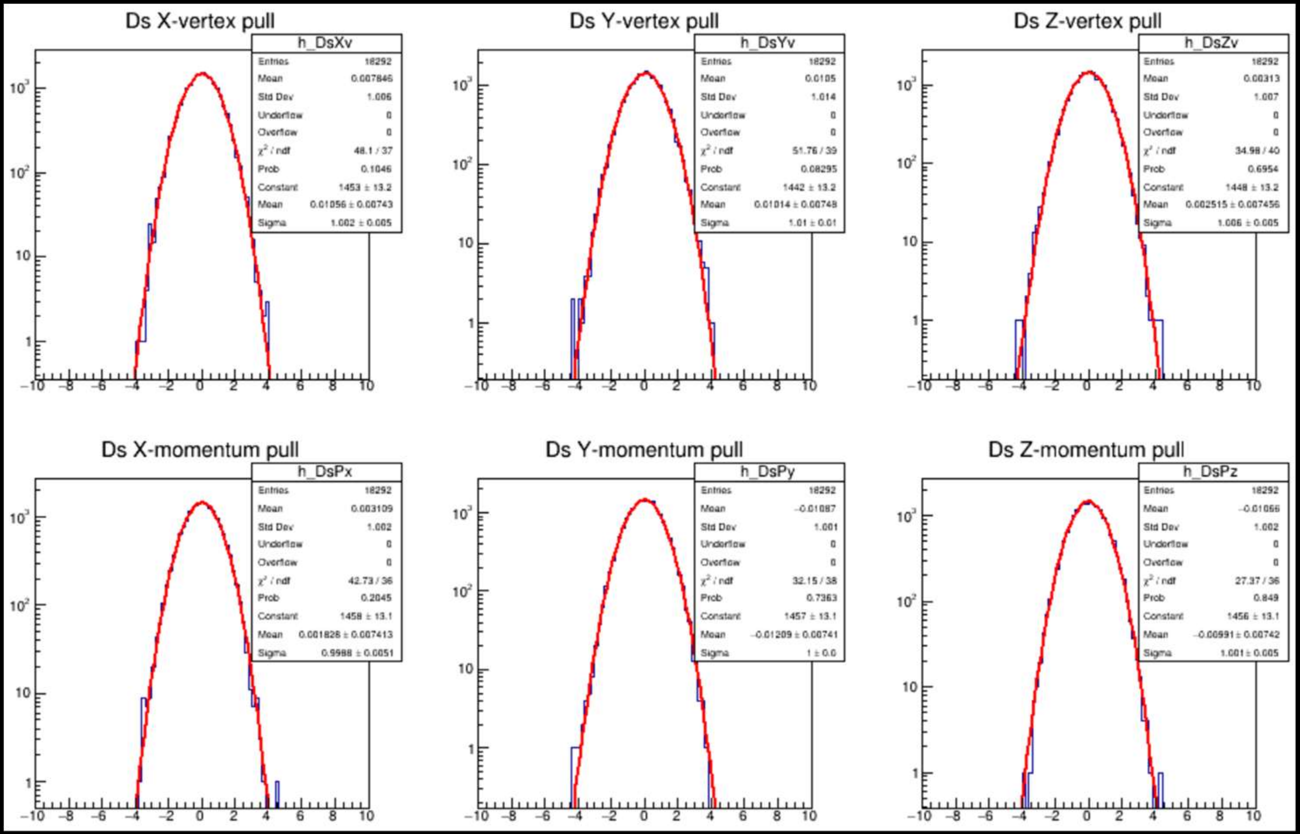
- Complete standalone vertexing package now allows fitting of any decay chain
  - Not connected to DELPHES
  - Can use with any track reconstruction based on helices/lines
    - Easily expandable to additional track parameterizations
- Method and usage fully documented in FCC note

- WARNING:
  - B field is set inside VertexMore at present. Should find a better way to pass it. It's a single line, but one must be careful if using this code with $B \neq 2$.

Additional slides

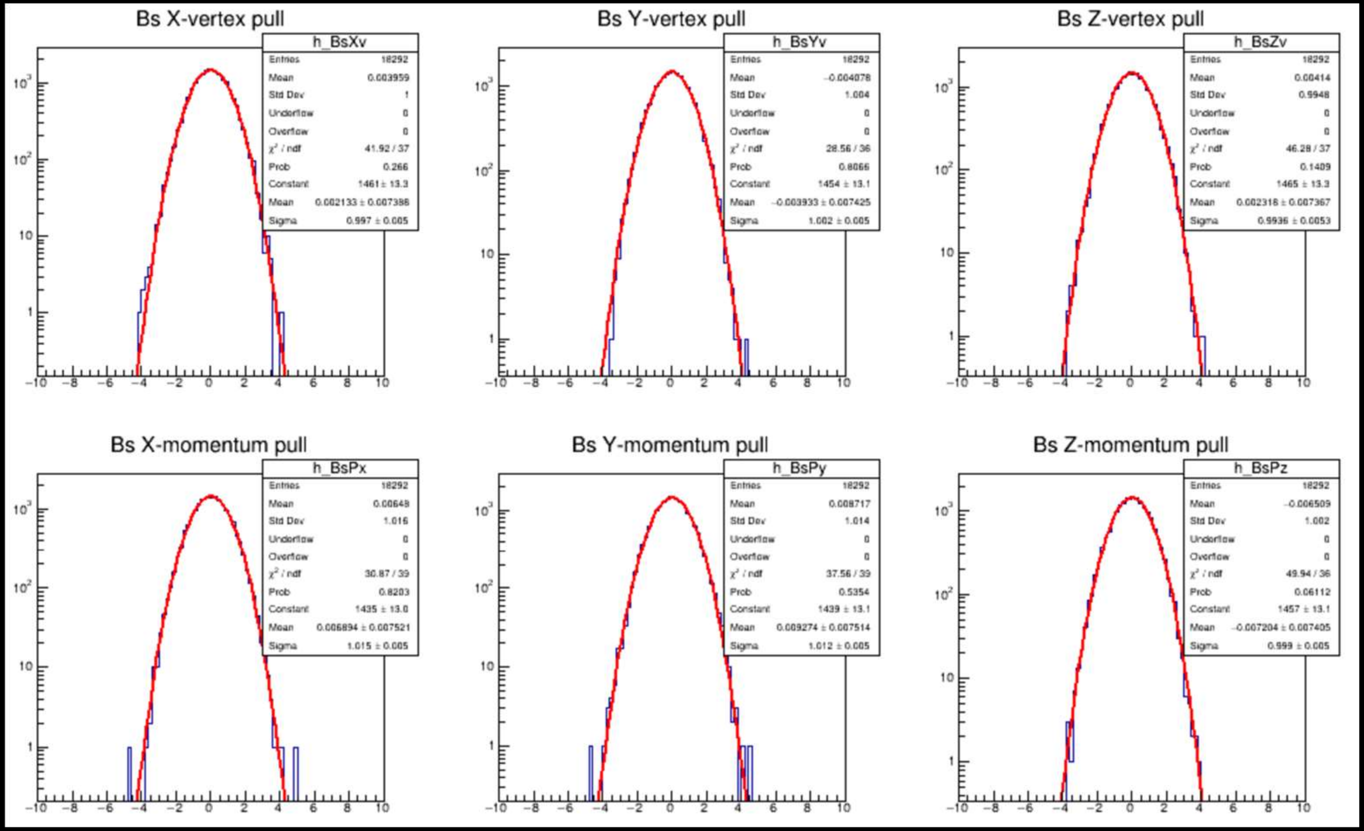
Bs → Ds π results

- Ds:



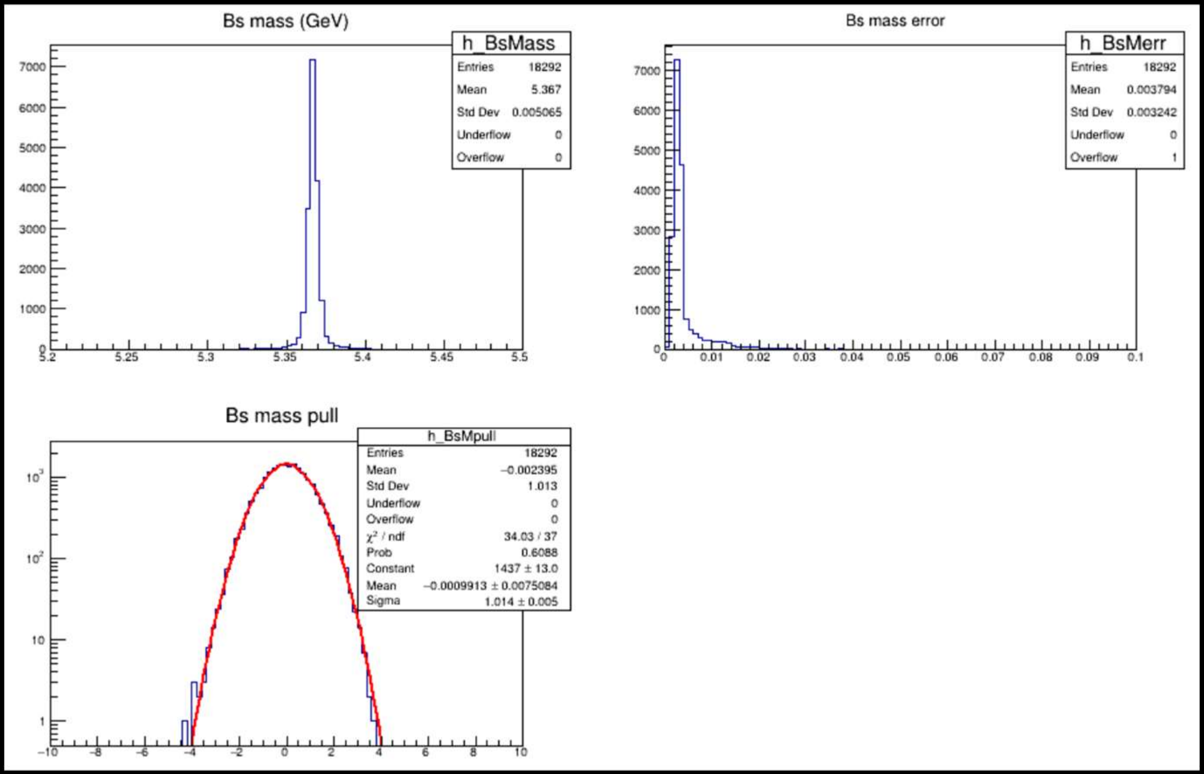
Bs → Ds π results

- ❖ Ds:
- ❖ Bs:



Bs → Ds π results

- Ds:
- Bs:
- Bs mass:


Bs mass (GeV)
h_BsMass
Entries 18292
Mean 5.367
Std Dev 0.005065
Underflow 0
Overflow 0
Bs mass error
h_BsMerr
Entries 18292
Mean 0.003794
Std Dev 0.003242
Underflow 0
Overflow 1
Bs mass pull
h_BsMpull
Entries 18292
Mean −0.002395
Std Dev 1.013
Underflow 0
Overflow 0
χ² / ndf 34.03 / 37
Prob 0.6088
Constant 1437 ± 13.0
Mean −0.0009913 ± 0.0075084
Sigma 1.014 ± 0.005
10²
10³