



TOOLS FOR INTEGRATED NA61/SHINE ANALYSIS

Silvestro di Luise
ETHZ

NA61/Shine Analysis/Calibration/Software Meeting
Katowice, Jan. 2015

PROJECT

**Build a set of Common Tools
to be shared in the NA61/Shine Analysis**

**T2K/CR
HI**

....

PROJECT GOAL

Deal efficiently with:

Multi Dimensional Phase Space Based Analysis

Multiple Dataset in 1 or 2 D

Multicomponent Models in 1 or 2 D

Multistage Fit-to-Data procedures

Tools must be framework independent:

Standalone

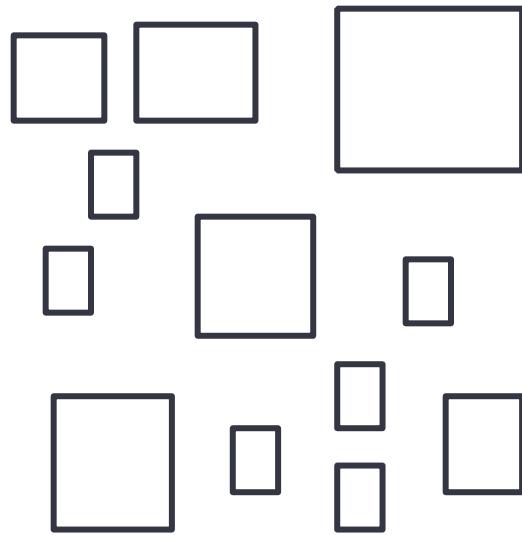
Transparent to data-format/structure

Portable

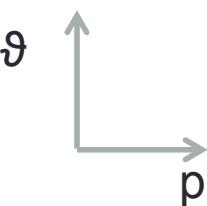
PROJECT DESCRIPTION OF THE MAIN LIBRARIES

Phase Space Map & Folders

TSPPhaseSpaceMap

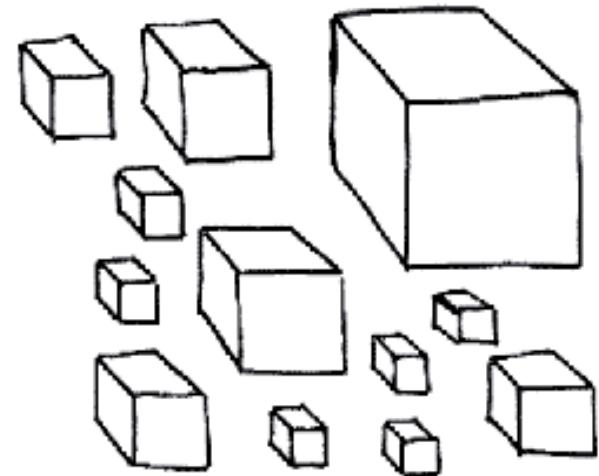


2D



12 volumes

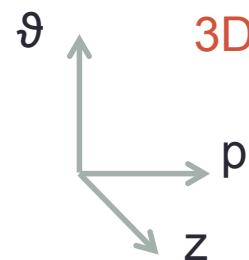
$$\times \quad - =$$



1D



1 volume

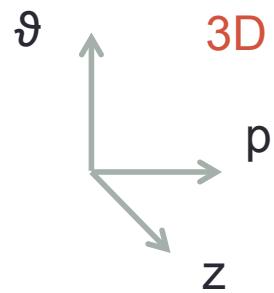
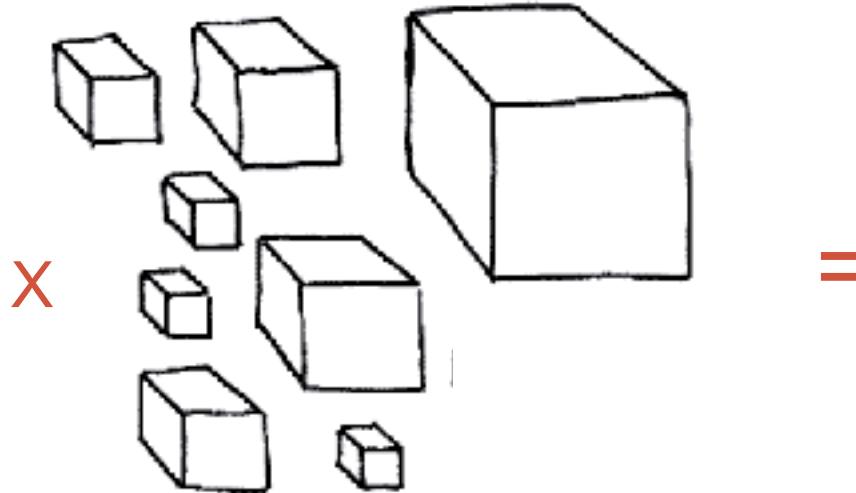
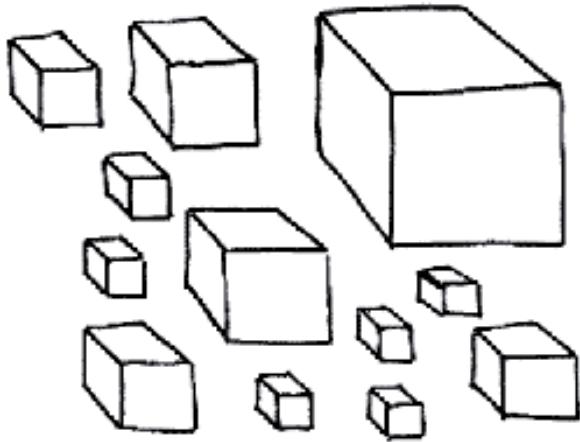


12 volumes

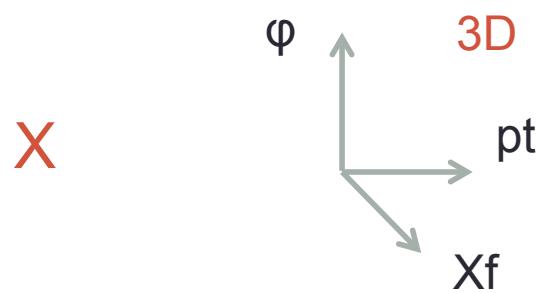
You can multiply two maps
....of the any dimensions

TSPPhaseSpaceMap

You can multiply two maps
....of the any dimensions



12 volumes

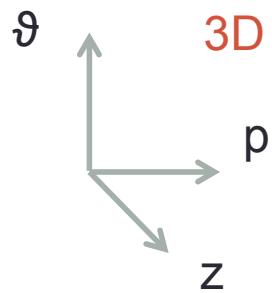
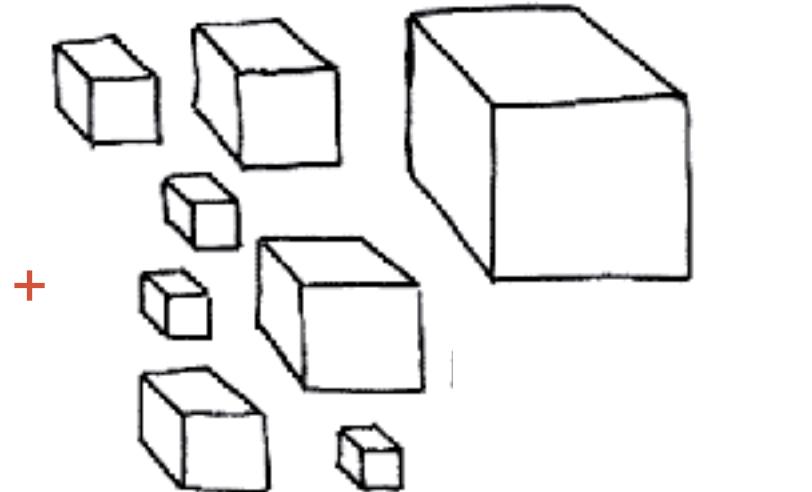
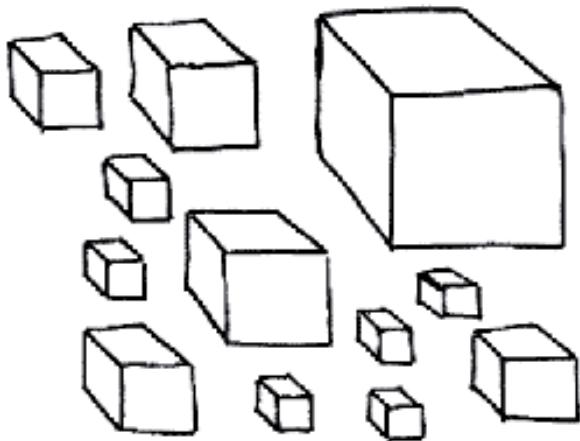


8 volumes

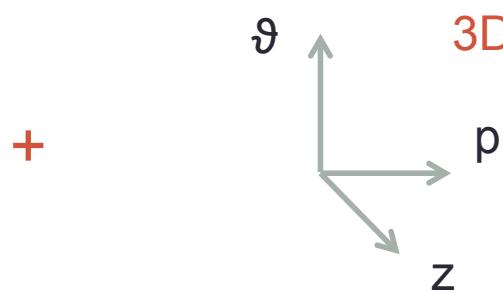
96 volumes

TSPPhaseSpaceMap

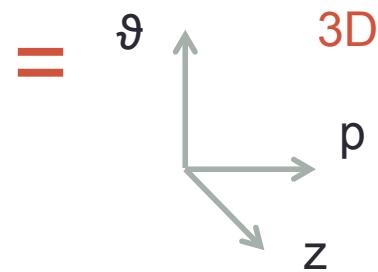
You can Add two maps
....of the same dimension



12 volumes



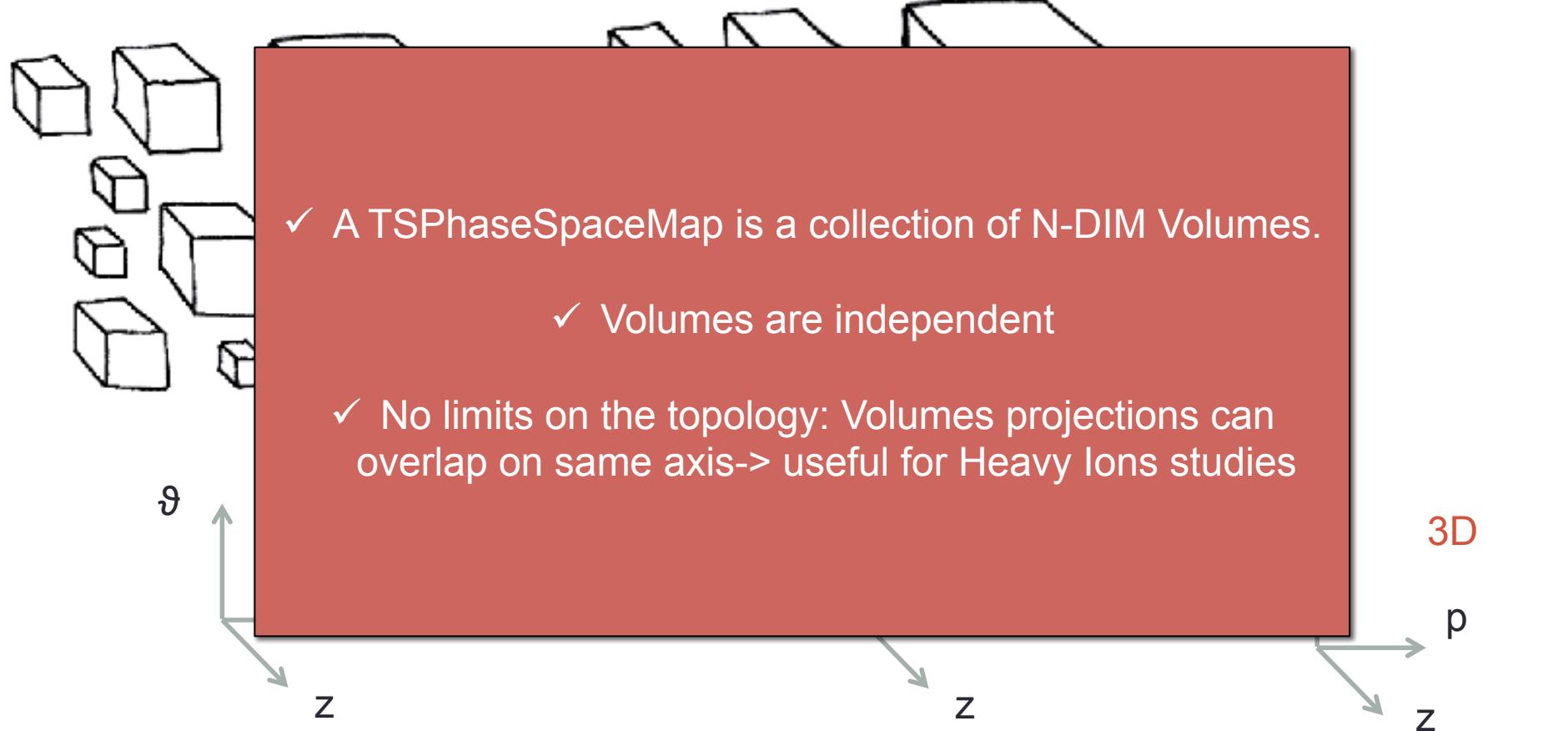
8 volumes



20 volumes

TSPhaseSpaceMap

You can Add two maps
....of the same dimension



12 volumes

8 volumes

20 volumes

TSPhaseSpaceMap

```
CC" --- Phase Space ---");
```

```
TSPhaseSpaceMap mapKin3D("mapKin3D", "map Theta-Mom-Z");
```

```
mapKin3D.SetVariable(Theta);  
mapKin3D.SetVariable(Mom);
```

```
mapKin3D.SetBinning("Theta:Mom", "mymap.theta_mom.txt");
```

```
mapKin3D.SetVariable(ZTarg);
```

```
mapKin3D.SetBinning(ZTarg, 3, ZTarg.GetMin(), ZTarg.GetMax());
```

```
CC" Kinematics_and_Observable_variables ")  
//  
  
TSVariable Theta("Theta", "#theta", "polar angle", "mrad");  
TSVariable Mom("Mom", "p_{tot}", "momentum", "GeV/c");  
TSVariable ZTarg("ZTarg", "z_{targ}", "z-target", "cm");  
  
TSVariable dEdx("dEdx", "dE/dx", "dE/dx", "arb. units");  
TSVariable ToF("ToF", "m^{2}", "ToF m^{2}", "GeV^{2}/c^{4}");  
TSVariable NPoints("NPoints", "Num. of Points", "Num. of Points");  
  
Theta.SetRange(0., 400.);  
Mom.SetRange(0.1, 31.);  
ZTarg.SetRange(0., 90.);  
  
dEdx.SetRange(0.8, 1.8);  
ToF.SetRange(-0.5, 2);  
NPoints.SetRange(1., 300.);  
  
TSArgList KinVarList(Theta, Mom, ZTarg);  
  
KinVarList.PrintVariables();
```

TSPPhaseSpaceMap

```
CC" --- Phase_Space ---");
```

```
TSPhaseSpaceMap mapKin3D("mapKin3D","map Theta-Mom-Z");
```

```
mapKin3D.SetVariable(Theta);
mapKin3D.SetVariable(Mom);
```

```
mapKin3D.SetBinning("Theta:Mom","mymap.theta_mom.txt");
```

```
mapKin3D.SetVariable(ZTarg);
```

```
mapKin3D.SetBinning(ZTarg,3,ZTarg.GetMin(),ZTarg.GetMax());
```

```
mapKin3D.Print();
```

```
mapKin3D.GetHistogramFld()->Write();
```

```
//Create two folders with the same map, one for positive one for negative tracks
TSPPhaseSpaceFld mapKin3D_Pos("map_PosTracks","q>0",mapKin3D.Title(),mapKin3D);
TSPPhaseSpaceFld mapKin3D_Neg("map_NegTracks","q<0",mapKin3D.Title(),mapKin3D);
```



```
GNU nano 2.0.6
File: mymap.theta_mom.txt

var1 offset 50. 20 x 10.
var2 offset 1. 3 x 2.

var1 offset 5. 20 x 5.
var2 offset 10. 5 x 2.

var1 offset 300. 2 x 50.
var2 offset 8. 3 x 3.
```

TSPPhaseSpaceMap

In a block maps (rows) are multiplied
Different blocks are Added

```
GNU nano 2.0.6
var1 offset 50. 20 x 10.
var2 offset 1. 3 x 2.

var1 offset 5. 20 x 5.
var2 offset 10. 5 x 2.

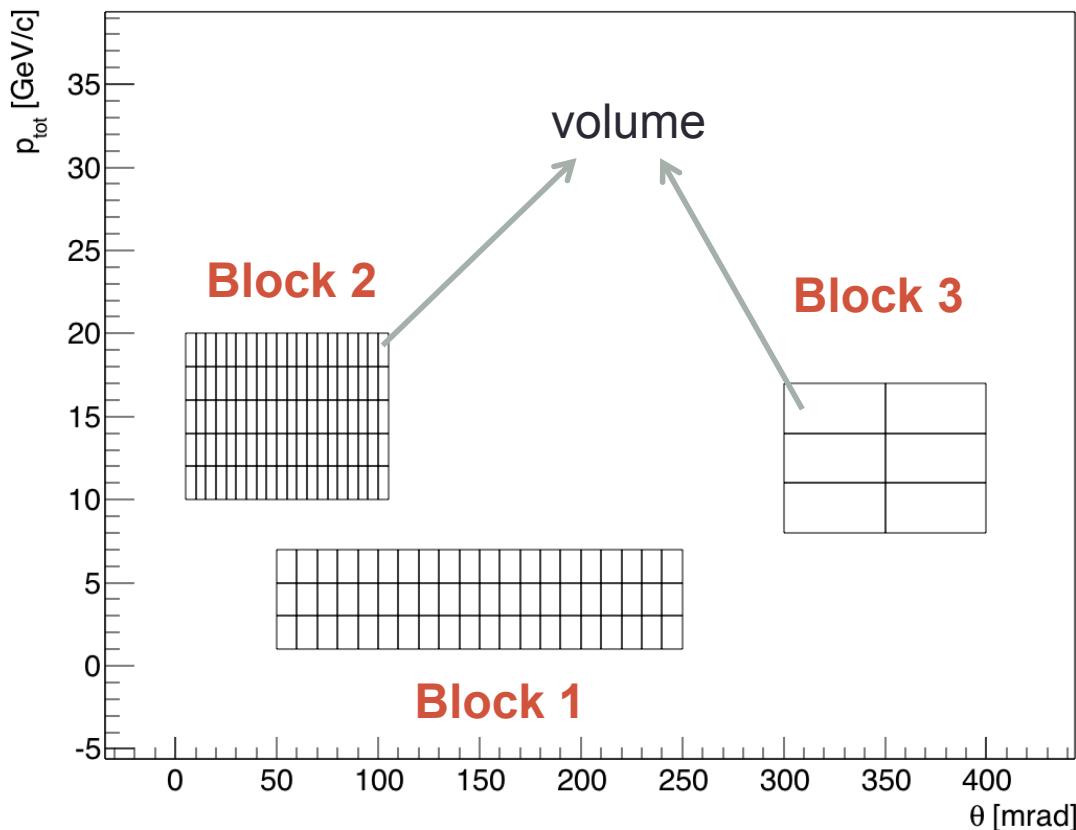
var1 offset 300. 2 x 50.
var2 offset 8. 3 x 3.
```

Block 1

Block 2

Block 3

map Theta-Mom-Z



TSPhaseSpaceMap

```

CC" --- Phase_Space ---");
TSPHaseSpaceMap mapKin2D("mapKin2D", "map Theta-Mom");
mapKin2D.SetVariable(Theta);
mapKin2D.SetVariable(Mom);

mapKin2D.SetBinning("Theta:Mom", "mymap.theta_mom.txt");

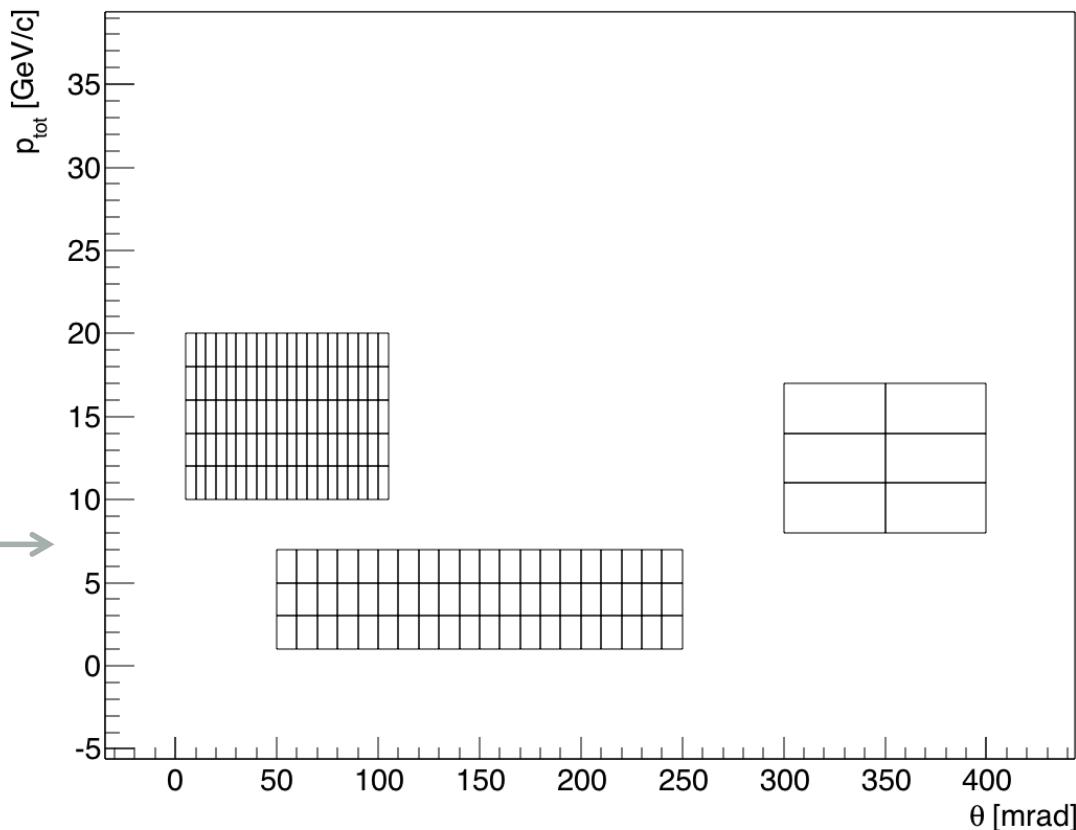
TSPHaseSpaceMap mapKin3D("mapKin3D", "map Theta-Mom-Z");
mapKin3D.SetVariable(Theta);
mapKin3D.SetVariable(Mom);

mapKin3D.SetBinning("Theta:Mom", "mymap.theta_mom.txt");
mapKin3D.SetVariable(ZTarg);
mapKin3D.SetBinning(ZTarg, 3, ZTarg.GetMin(), ZTarg.GetMi
mapKin3D.Print();           —————→
mapKin3D.GetHistogramFld()->Write();

//Create two folders with the same map, one for posit
TSPHaseSpaceFld mapKin3D_Pos("map_PosTracks", "q>0", ma
TSPHaseSpaceFld mapKin3D_Neg("map_NegTracks", "q<0", ma

```

map Theta-Mom-Z



TSPPhaseSpaceMap

```
CC" --- Phase_Space ---");

TSPPhaseSpaceMap mapKin2D("mapKin2D", "map Theta-Mom");
mapKin2D.SetVariable(Theta);
mapKin2D.SetVariable(Mom);

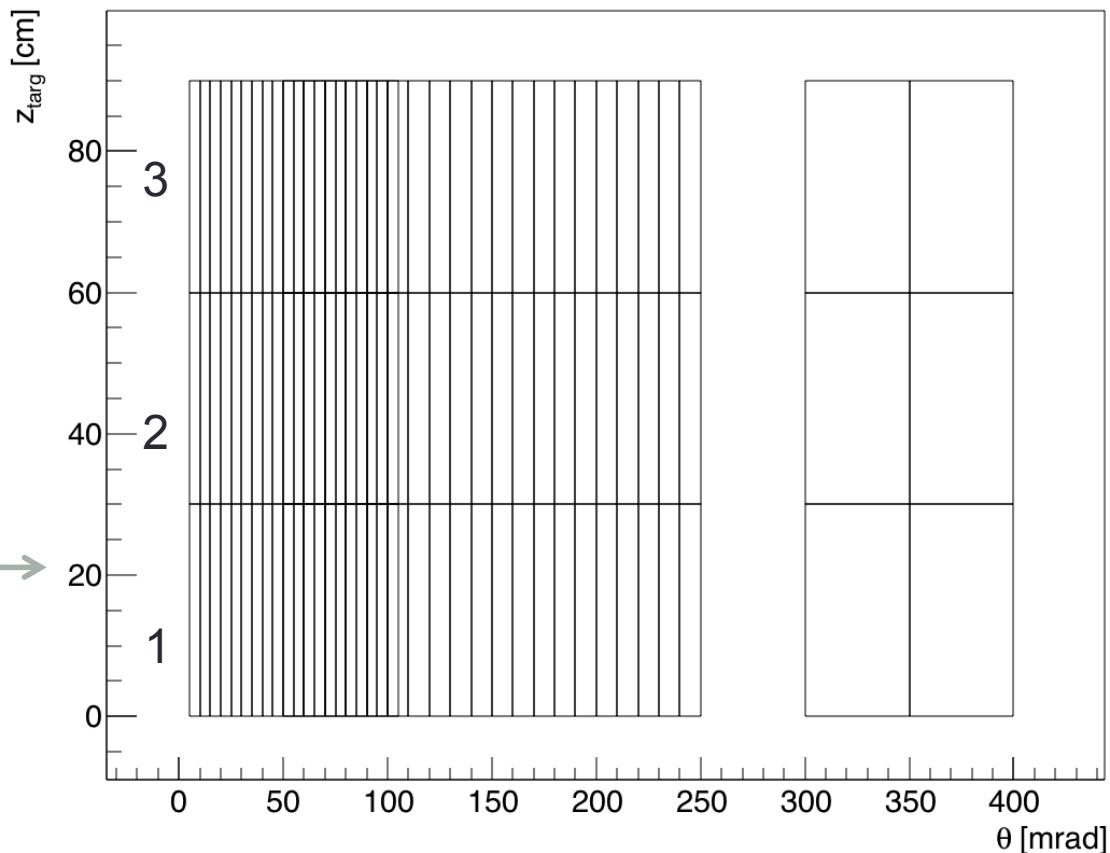
mapKin2D.SetBinning("Theta:Mom", "mymap.theta_mom.txt");

TSPPhaseSpaceMap mapKin3D("mapKin3D", "map Theta-Mom-Z");
mapKin3D.SetVariable(Theta);
mapKin3D.SetVariable(Mom);

mapKin3D.SetBinning("Theta:Mom", "mymap.theta_mom.txt");
mapKin3D.SetVariable(ZTarg);
mapKin3D.SetBinning(ZTarg, 3, ZTarg.GetMin(), ZTarg.GetMax());
mapKin3D.Print();
mapKin3D.GetHistogramFld()->Write();

//Create two folders with the same map, one for positive q
TSPPhaseSpaceFld mapKin3D_Pos("map_PosTracks", "q>0", 1);
TSPPhaseSpaceFld mapKin3D_Neg("map_NegTracks", "q<0", 1);
```

map Theta-Mom-Z



TSPPhaseSpaceMap

```
CC" --- Phase_Space ---");

TSPPhaseSpaceMap mapKin2D("mapKin2D", "map Theta-Mom");

mapKin2D.SetVariable(Theta);
mapKin2D.SetVariable(Mom);

mapKin2D.SetBinning("Theta:Mom", "mymap.theta_mom.txt");

TSPPhaseSpaceMap mapKin3D("mapKin3D", "map Theta-Mom-Z");

mapKin3D.SetVariable(Theta);
mapKin3D.SetVariable(Mom);

mapKin3D.SetBinning("Theta:Mom", "mymap.theta_mom.txt");

mapKin3D.SetVariable(ZTarg);

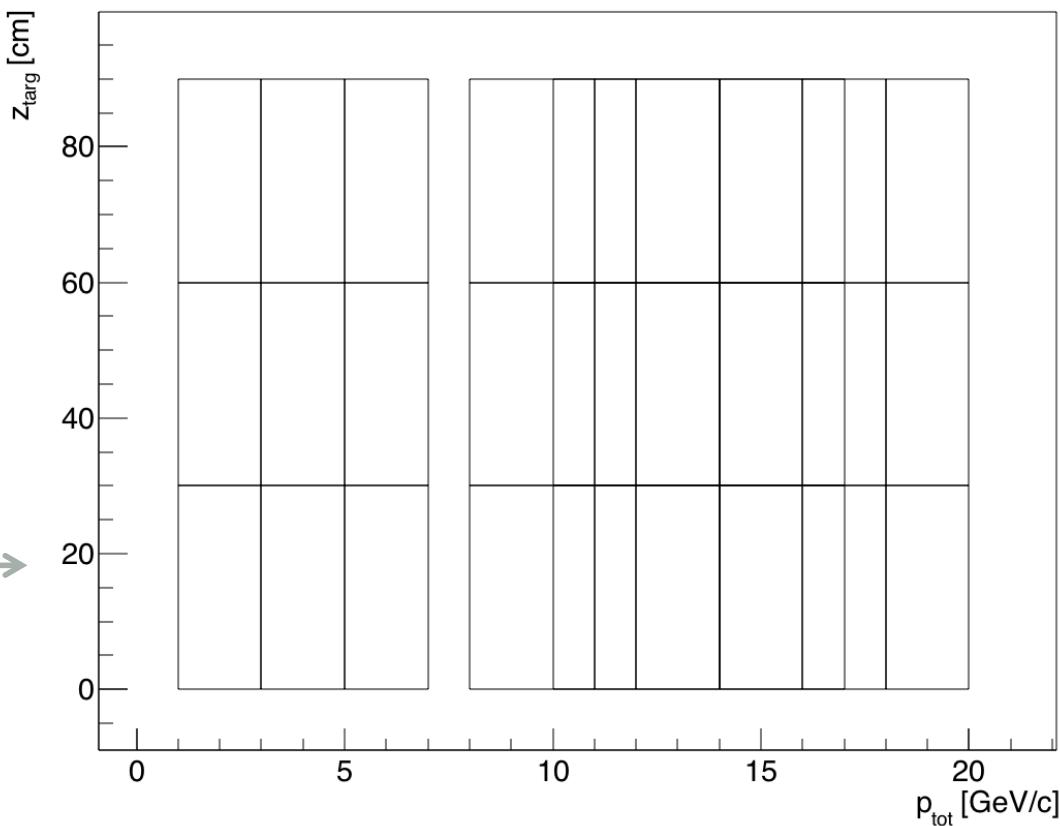
mapKin3D.SetBinning(ZTarg, 3, ZTarg.GetMin(), ZTarg.GetMax)

mapKin3D.Print();           —————→

mapKin3D.GetHistogramFld()->Write();

//Create two folders with the same map, one for positiv
TSPPhaseSpaceFld mapKin3D_Pos("map_PosTracks", "q>0", mapK
TSPPhaseSpaceFld mapKin3D_Neg("map_NegTracks", "q<0", mapK
```

map Theta-Mom-Z



TSPhaseSpaceMap

```

CC" --- Phase Space ---");

TSPHASESPACEMAP mapKin2D("mapKin2D", "map Theta-Mom");
mapKin2D.SetVariable(Theta);
mapKin2D.SetVariable(Mom);

mapKin2D.SetBinning("Theta:Mom", "mymap.theta_mom.txt");

TSPHASESPACEMAP mapKin3D("mapKin3D", "map Theta-Mom-Z");
mapKin3D.SetVariable(Theta);
mapKin3D.SetVariable(Mom);

mapKin3D.SetBinning("Theta:Mom", "mymap.theta_mom.txt");

mapKin3D.SetVariable(ZTarg);
mapKin3D.SetBinning(ZTarg, 3, ZTarg.GetMin(), ZTarg.GetMax());
mapKin3D.Print();

mapKin3D.GetHistogramFld()->Write();

```

```

//Create two folders with the same map, one for positive one for negative
TSPHASESPACEFLD mapKin3D_Pos("map_PosTracks", "q>0", mapKin3D.Title());
TSPHASESPACEFLD mapKin3D_Neg("map_NegTracks", "q<0", mapKin3D.Title());

```

```

===== PhaseSpaceMap mapKin3D map Theta-Mom-Z ======
Dimension 3
Axis Variables
1 Theta
2 Mom
3 ZTarg

Number of Volumes: 498
vol: 1 #theta:[50,60] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 2 #theta:[60,70] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 3 #theta:[70,80] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 4 #theta:[80,90] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 5 #theta:[90,100] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 6 #theta:[100,110] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 7 #theta:[110,120] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 8 #theta:[120,130] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 9 #theta:[130,140] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 10 #theta:[140,150] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 11 #theta:[150,160] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 12 #theta:[160,170] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 13 #theta:[170,180] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 14 #theta:[180,190] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 15 #theta:[190,200] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 16 #theta:[200,210] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 17 #theta:[210,220] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 18 #theta:[220,230] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 19 #theta:[230,240] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 20 #theta:[240,250] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 21 #theta:[50,60] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 22 #theta:[50,60] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 23 #theta:[60,70] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 24 #theta:[60,70] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 25 #theta:[70,80] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 26 #theta:[70,80] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 27 #theta:[80,90] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 28 #theta:[80,90] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 29 #theta:[90,100] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 30 #theta:[90,100] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 31 #theta:[100,110] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 32 #theta:[100,110] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 33 #theta:[110,120] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 34 #theta:[110,120] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 35 #theta:[120,130] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 36 #theta:[120,130] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 37 #theta:[130,140] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm

```

TSPhaseSpaceMap

```

CC" --- Phase Space ---");

TSPHASESPACEMAP mapKin2D("mapKin2D", "map Theta-Mom");
mapKin2D.SetVariable(Theta);
mapKin2D.SetVariable(Mom);
mapKin2D.SetBinning(ZTarg, 3, ZTarg.GetMin(), ZTarg.GetMax());
mapKin2D.Print();
mapKin2D.GetHistogramFld()->Write();

//Create two folders with the same map, one for positive one for negative
TSPHASESPACEFLD mapKin3D_Pos("map_PosTracks", "q>0", mapKin3D.Title());
TSPHASESPACEFLD mapKin3D_Neg("map_NegTracks", "q<0", mapKin3D.Title());

```

```

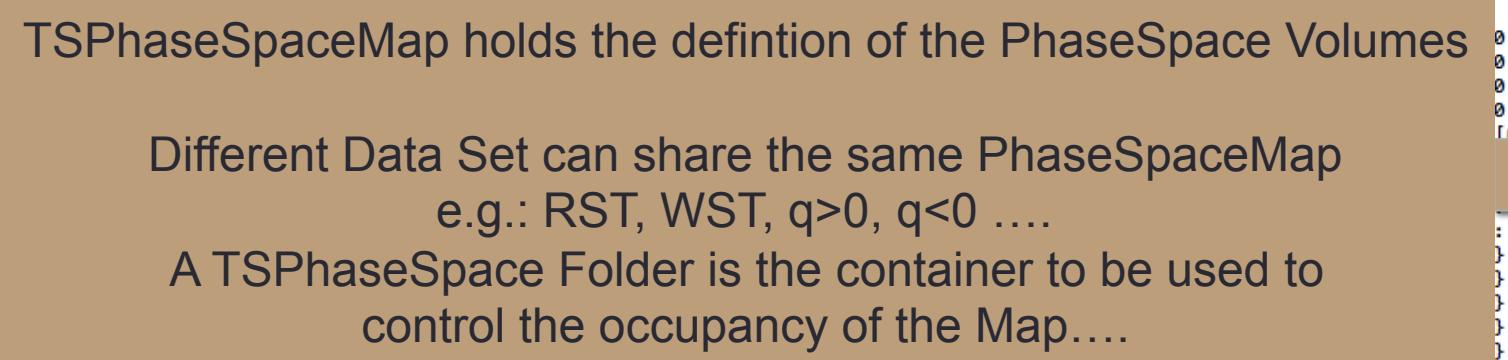
===== PhaseSpaceMap mapKin3D map Theta-Mom-Z ======
Dimension 3
Axis Variables
1 Theta
2 Mom
3 ZTarg

```

TSPHASESPACEMAP holds the definition of the PhaseSpace Volumes

Different Data Set can share the same PhaseSpaceMap
e.g.: RST, WST, $q>0$, $q<0$

A TSPHASESPACE Folder is the container to be used to control the occupancy of the Map....



```

vol: 15 #theta:[190,200] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 16 #theta:[200,210] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 17 #theta:[210,220] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 18 #theta:[220,230] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 19 #theta:[230,240] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 20 #theta:[240,250] mrad p_{tot}:[1,3] GeV/c z_{targ}:[0,30] cm
vol: 21 #theta:[50,60] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 22 #theta:[50,60] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 23 #theta:[60,70] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 24 #theta:[60,70] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 25 #theta:[70,80] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 26 #theta:[70,80] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 27 #theta:[80,90] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 28 #theta:[80,90] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 29 #theta:[90,100] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 30 #theta:[90,100] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 31 #theta:[100,110] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 32 #theta:[100,110] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 33 #theta:[110,120] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 34 #theta:[110,120] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 35 #theta:[120,130] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm
vol: 36 #theta:[120,130] mrad p_{tot}:[5,7] GeV/c z_{targ}:[0,30] cm
vol: 37 #theta:[130,140] mrad p_{tot}:[3,5] GeV/c z_{targ}:[0,30] cm

```

A Map Folder can be filled ...example later

TSPhaseSpaceFld

```
CC" --- Phase_Space ---");
```

```
TSPhaseSpaceMap mapKin2D("mapKin2D", "map_Theta-Mom");
```

```
mapKin2D.SetVariable(Theta);
mapKin2D.SetVariable(Mom);
```

```
mapKin2D.SetBinning("Theta:Mom", "mymap.theta_mom.txt");
```

```
TSPhaseSpaceMap mapKin3D("mapKin3D", "map_Theta-Mom-Z");
```

```
mapKin3D.SetVariable(Theta);
mapKin3D.SetVariable(Mom);
```

```
mapKin3D.SetBinning("Theta:Mom", "mymap.theta_mom.txt");
```

```
mapKin3D.SetVariable(ZTarg);
```

```
mapKin3D.SetBinning(ZTarg, 3, ZTarg.GetMin(), ZTarg.GetMax());
```

```
mapKin3D.Print();
```

```
mapKin3D.GetHistogramFld()->Write();
```

```
//Create two folders with the same map, one for positive one for negative tracks
```

```
TSPhaseSpaceFld mapKin3D_Pos("map_PosTracks", "q>0", mapKin3D.Title(), mapKin3D);
```

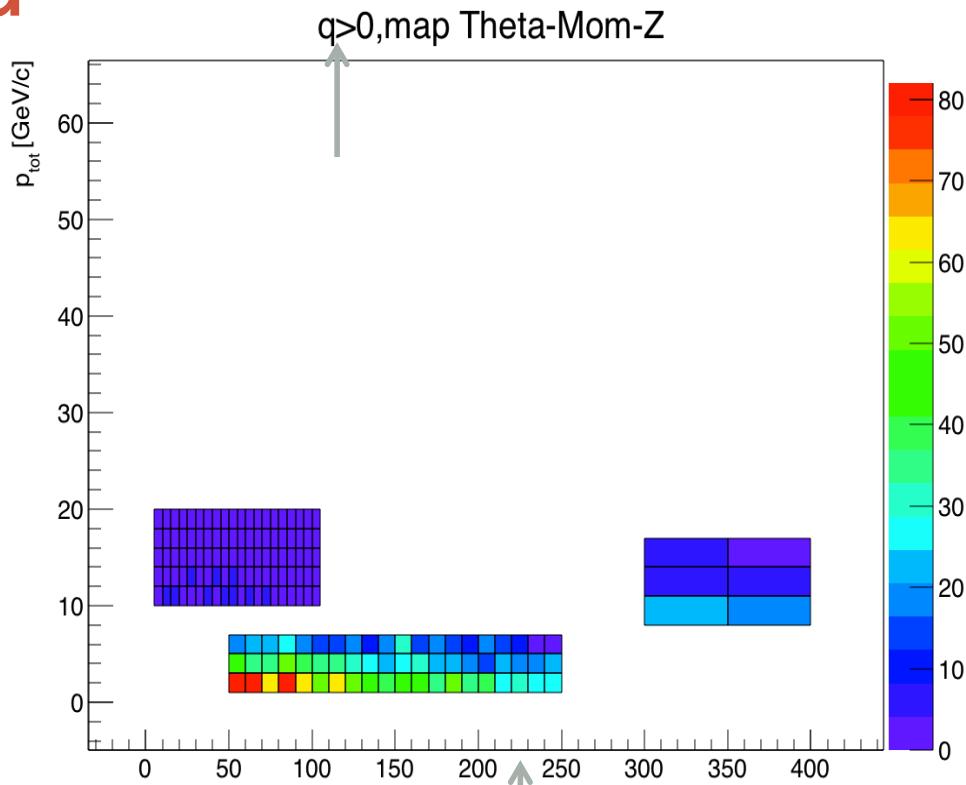
```
TSPhaseSpaceFld mapKin3D_Neg("map_NegTracks", "q<0", mapKin3D.Title(), mapKin3D);
```

.....fill the MapFolder

```
mapKin3D_Pos.GetHistogramFld()->Write();
```

```
mapKin3D_Neg.GetHistogramFld()->Write();
```

```
mapKin3D_Pos.Print();
```



A Map Folder can be filled ...example later

TSPhaseSpaceFld

```
CC " --- Phase Space ---");

TSPHASESPACEMAP mapKin2D("mapKin2D", "map Theta-Mom");
mapKin2D.SetVariable(Theta);
mapKin2D.SetVariable(Mom);

mapKin2D.SetBinning("Theta:Mom", "mymap.theta_mom.txt");

TSPHASESPACEMAP mapKin3D("mapKin3D", "map Theta-Mom-Z");
mapKin3D.SetVariable(Theta);
mapKin3D.SetVariable(Mom);

mapKin3D.SetBinning("Theta:Mom", "mymap.theta_mom.txt");

mapKin3D.SetVariable(ZTarg);
mapKin3D.SetBinning(ZTarg, 3, ZTarg.GetMin(), ZTarg.GetMax());
mapKin3D.Print();

mapKin3D.GetHistogramFld()->Write();

//Create two folders with the same map, one for positive one for negative tracks
TSPHASESPACEFLD mapKin3D_Pos("map_PosTracks", "q>0", mapKin3D.Title(), mapKin3D);
TSPHASESPACEFLD mapKin3D_Neg("map_NegTracks", "q<0", mapKin3D.Title(), mapKin3D);
```

.....fill the MapFolder

```
mapKin3D_Pos.GetHistogramFld()->Write();
mapKin3D_Neg.GetHistogramFld()->Write();

mapKin3D_Pos.Print();
```

```
==== PhaseSpaceFolder map_PosTracks q>0 map Theta-Mom-Z ======
PhaseSpaceMap: mapKin3D map Theta-Mom-Z
Tag: q>0

}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 35
}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 37
}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 31
}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 32
}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 26
}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 26
}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 25
}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 13
tot}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 13
tot}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 19
tot}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 17
tot}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 15
tot}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 27
tot}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 21
tot}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 19
tot}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 10
tot}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 14
tot}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 17
tot}: [1,3] GeV/c z_{targ}:[0,30] cm, entries: 12
}: [3,5] GeV/c z_{targ}:[0,30] cm, entries: 16
}: [5,7] GeV/c z_{targ}:[0,30] cm, entries: 6
}: [3,5] GeV/c z_{targ}:[0,30] cm, entries: 15
}: [5,7] GeV/c z_{targ}:[0,30] cm, entries: 12
}: [3,5] GeV/c z_{targ}:[0,30] cm, entries: 15
}: [5,7] GeV/c z_{targ}:[0,30] cm, entries: 9
}: [3,5] GeV/c z_{targ}:[0,30] cm, entries: 16
}: [5,7] GeV/c z_{targ}:[0,30] cm, entries: 14
tot}: [3,5] GeV/c z_{targ}:[0,30] cm, entries: 17
tot}: [5,7] GeV/c z_{targ}:[0,30] cm, entries: 8
-----
```



Parameters

TSPParameter::

Is a Variable +
an Error, a Status (Free/Fixed)
a StepSize etc.....

```
CC"---- Define Parameters -----");
```

```
TSVariable dEdx_x("dEdx_x","dE/dx","dE/dx","arb. units");
TSVariable m2_x("m2_x","m^{2}","ToF m^{2}","GeV^{2}/c^{4}");

TSPParameter dEdx_sig_0("dEdx_sig_0","#sigma_{dE/dx}","dE/dx width","arb. units");
TSPParameter dEdx_sig_alpha("dEdx_sig_alpha","#alpha","#alpha scaling factor");

TSPParameter dEdx_sig_delta("dEdx_sig_delta","#delta","asymmetry parameter");

TSPParameter dEdx_Pi_mean("dEdx_Pi_mean","m_{dE/dx}-#pi","#pi dE/dx mean","arb. units");

TSPParameter dEdx_mean_PtoPi("dEdx_mean_PtoPi","m_{dE/dx}-P#pi","dE/dx mean","arb. units");
TSPParameter dEdx_mean_KtoPi("dEdx_mean_KtoPi","m_{dE/dx}-K#pi","dE/dx mean","arb. units");
```

TSPParamFunction::

```
TSPParamFunction dEdx_K_mean("dEdx_K_mean", "m_{dE/dx}-K", "K dE/dx mean", "arb. units");
dEdx_K_mean.SetFunction("dEdx_Pi_mean+dEdx_mean_KtoPi", TSArgList(dEdx_Pi_mean,dEdx_mean_KtoPi));
  → type in formula:  $m_K = m_\pi - m_{K\pi}$ 
```

```
TSPParamFunction dEdx_K_sig("dEdx_K_sig", "#sigma_{dE/dx}-K", "dE/dx width", "arb. units");
dEdx_K_sig.SetFunction("dEdx_sig_0*TMath::Power(dEdx_K_mean/dEdx_Pi_mean,dEdx_sig_alpha)"
  ,TSArgList(dEdx_sig_0,dEdx_Pi_mean,dEdx_K_mean,dEdx_sig_alpha));
  ↑ type in formula:  $\sigma_K = \sigma_0(m_K/m_\pi)^\alpha$  → dependencies params
```

dEdx_K_sig.GetValue();	→ Evaluate formula... $\sigma_0(m_K/m_\pi)^\alpha$
dEdx_K_sig.GetError();	→ Calculate Error propagation....
dEdx_K_sig.GetError(cov_matrix);	→ Calculate Error propagation with correlations....

TSPParamFunction::

```
TSPParamFunction dEdx_K_mean("dEdx_K_mean", "m_{dE/dx}-K", "K dE/dx mean", "arb. units");
dEdx_K_mean.SetFunction("dEdx_Pi_mean+dEdx_mean_KtoPi", TSArgList(dEdx_Pi_mean,dEdx_mean_KtoPi));
  → type in formula:  $m_K = m_\pi - m_{K\pi}$ 
```

```
TSPParamFunction dEdx_K_sig("dEdx_K_sig", "#sigma_{dE/dx}-K", "dE/dx width", "arb. units");
dEdx_K_sig.SetFunction("dEdx_sig_0*TMath::Power(dEdx_K_mean/dEdx_Pi_mean,dEdx_sig_alpha)"
  ,TSArgList(dEdx_sig_0,dEdx_Pi_mean,dEdx_K_mean,dEdx_sig_alpha));

dEdx_sig_0.SetValue(0.04);
dEdx_Pi_mean.SetValue(1.13);
dEdx_mean_KtoPi.SetValue(-0.1);
dEdx_sig_alpha.SetValue(0.65);

dEdx_K_sig.Print(); →
```

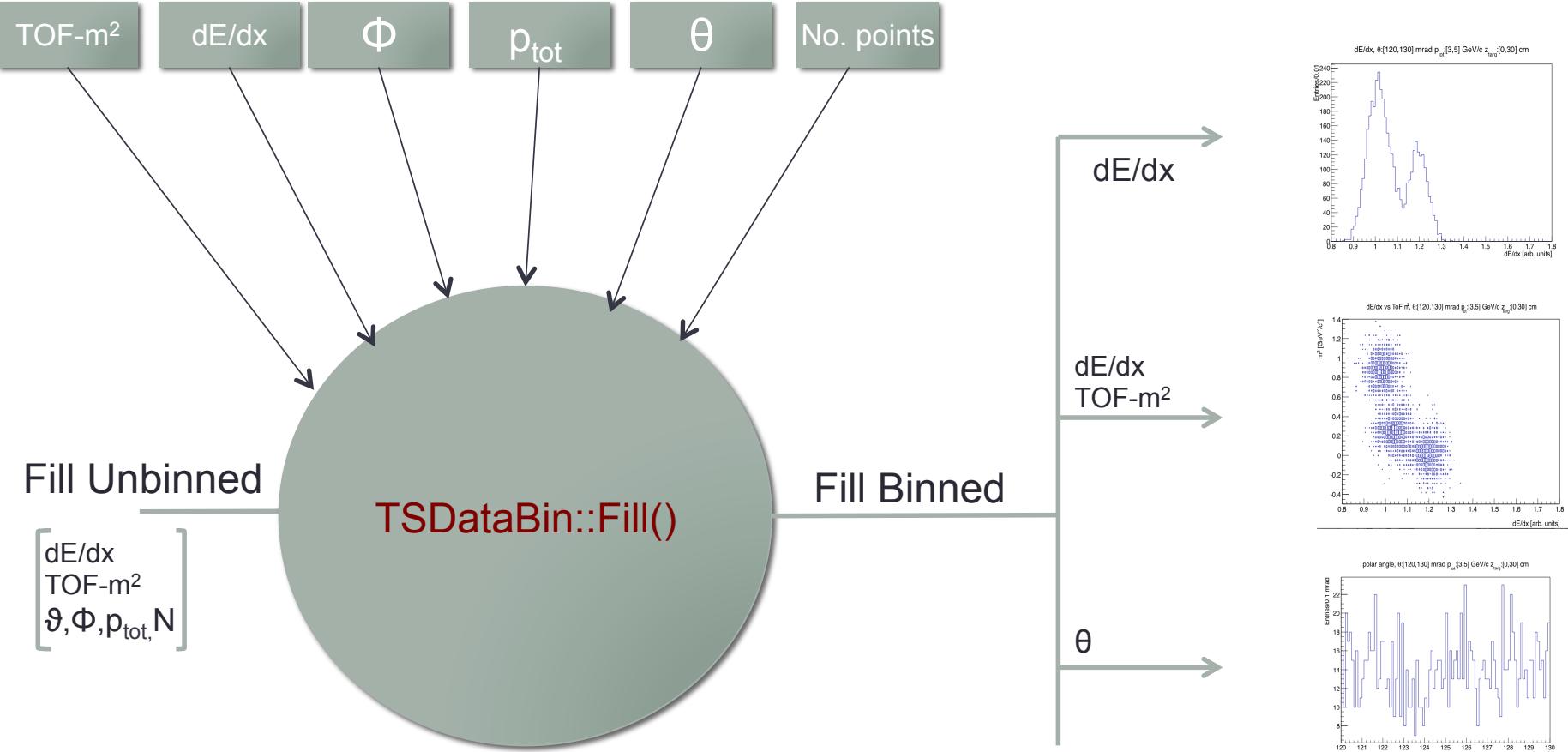
```
--- Print dEdx_K_sig ---
dEdx_sig_0*TMath::Power(dEdx_K_mean/dEdx_Pi_mean,dEdx_sig_alpha)
[0]*TMath::Power([2]/[1],[3])+
value: 0.037662
0 dEdx_sig_0 = 0.04 +/- 0
1 dEdx_Pi_mean = 1.13 +/- 0
2 dEdx_K_mean = 1.03 +/- 0
3 dEdx_sig_alpha = 0.65 +/- 0

Primary Param List:
dEdx_sig_0: dE/dx width #sigma_{dE/dx} value: 0.04
dEdx_Pi_mean: #pi dE/dx mean m_{dE/dx}-#pi value: 1.13
dEdx_mean_KtoPi: dE/dx mean m_{dE/dx}-K#pi value: -0.1
dEdx_sig_alpha: #alpha scaling factor #alpha value: 0.65
---
```

Data Containers

DataBin & DataSetMgr

- 1- Set Variables values
- 2- Fill the DataBin
- 3- iterate



Unbinned (root Tree) filled if selected

All the histograms in the container will be filled with the variable associated to the x/y axis

Variables values

2- Fill the DataBin

3- iterate



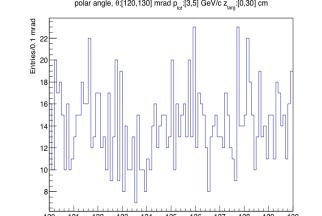
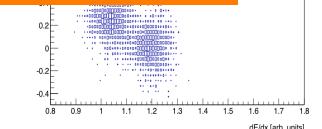
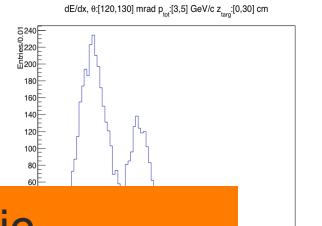
Fill Unbinned

 $[dE/dx$
 $TOF-m^2$
 $\vartheta, \Phi, p_{tot}, N]$

TSDataBin::Fill()

Binning can be automatic
 Useful for large range
 variations over the phase space
 of the same variable: e.g. ToF- m^2

Fill Binned

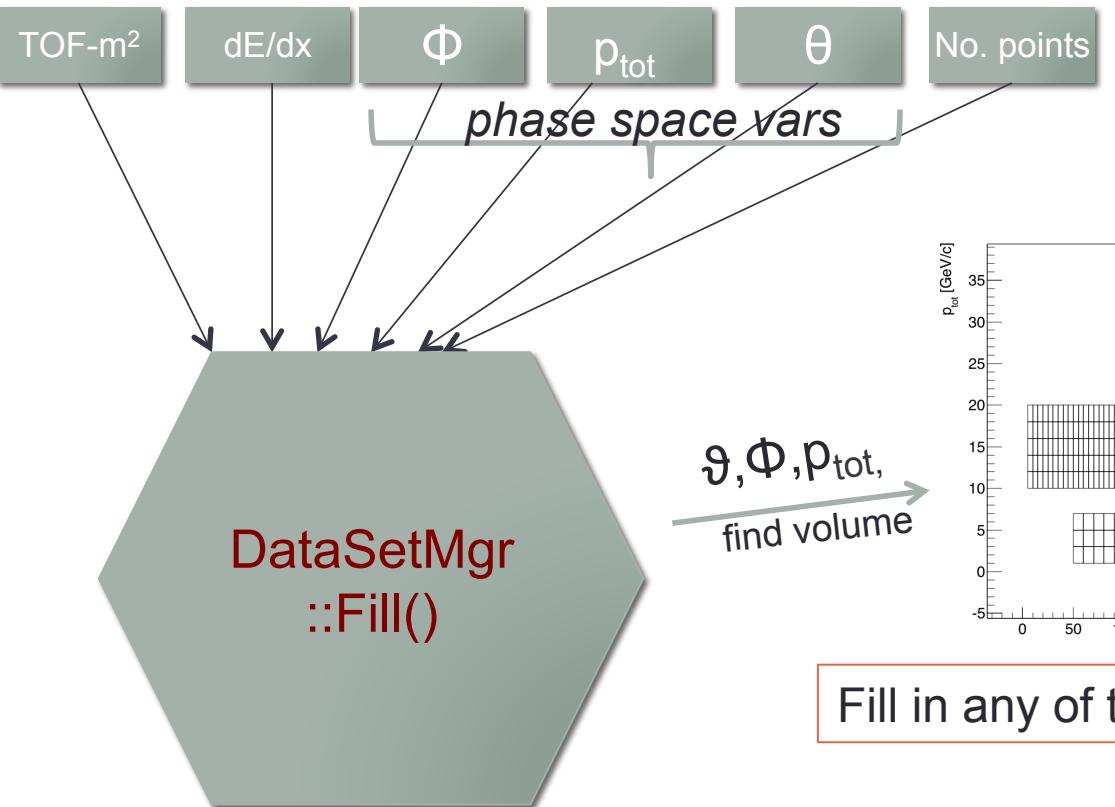
 θ 

Unbinned (root Tree) filled if selected

All the histograms in the container will be filled with the variable associated to the x/y axis

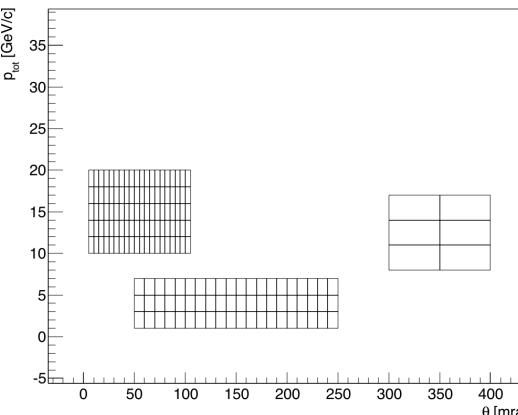
Filling DataSetMgr

- 1- Set Variables values
- 2- Fill the DataSetMgr
- 3- iterate



$\vartheta, \Phi, p_{\text{tot}},$
find volume

map Theta-Mom-Z



Fill in any of the PhaseSpaceMap volumes?

Yes: Fill DataBin for that volume

Fill Unbinned

$[dE/dx$
 TOF-m^2
 $\vartheta, \Phi, p_{\text{tot}}, N]$

Fill Binned

`TSDDataBin ::Fill()`

Dedicated Objects: BetheBloch Functions & Managers

```
cout<<" ===== BB ===== " << endl;
```

```
TSBetheBlochFunc bbF1("bbF1","bb f","bethe_bloch_table.txt");
```

```
double P
```

```
TF1 FBB0
```

```
FBB.Prin
```

```
TSBetheB
```

```
bbF2.Set
```

```
//bbF2.S
```

```
cout<<"b
```

```
bbF2.Print();
```

```
bbF1.BuildTableFromFunction(100);
```

```
bbF2.Write();
```

```
bbF2.BuildTableFromFunction(100);
```

```
bbF2.CloneTable()->Write();
```

Generic Class to handle and switch between different BetheBloch Parameterization (both analytic and tabulated)

```
cout<<" ===== BB ===== " << endl;

TSBetheBlochFunc bbF1("bbF1","bb f","bethe_bloch_table.txt");

double PAR0[5]={0.0663945,9.04358,2.49409,0.164471,0.216649};

TF1 FBB("fBB",BETHE_BLOCH_FUNC::BetheBloch,0.1,300,5);

FBB.Print();

TSBetheBlochFunc bbF2("bbF2","",BETHE_BLOCH_FUNC::BetheBloch,0.1,300,5);

bbF2.SetParameters(PAR0);

//bbF2.SetParameters(BETHE_BLOCH_FUNC::PAR_INPUT);

cout<<"bb: "<<bbF2.Eval(10.)<<" "<<FBB.Eval(10.)<< endl;

bbF2.Print();

bbF1.BuildTableFromFunction(100);

bbF2.Write();
bbF2.BuildTableFromFunction(100);
bbF2.CloneTable()->Write();
```

Dedicated Objects: dE/dx MC generator

Generic Class to generate dE/dx distribution

- 1- Parametric (per track)
- 2- Cluster Level (truncated mean)

(features to add bias smearings etc...)

```
dedxGenCluster.setGammaCut(0.8),  
dedxGenCluster.ApplyCalibrationCorrection(false);
```

```
CC"--- dE/dx generation ----");

TSDEdxGenerator dedxGenTrack("dedxGenGlobal","dE/dx generation at track level");

dedxGenTrack.SetTrackDEdxDistrib("Gaussian",-1,1,0,0.04);
dedxGenTrack.SetMean(0.);

dedxGenTrack.SetBetheBloch(bbFunc);

TF1 *TF1_dEdx_Gen = (TF1*)dedxGenTrack.ImportTrackDEdxDistrib();

TF1_dEdx_Gen->Write();

// TSDEdxGenerator dedxGenCluster("dedxGenCluster","dE/dx generation as truncated mean of cluster charges");

dedxGenCluster.SetClusterChargeDistrib("Gaussian",-1,1,0.,0.4);
dedxGenCluster.SetMean(0.);
dedxGenCluster.SetTruncation(0.6);
dedxGenCluster.ApplyCalibrationCorrection(false);
```

Dedicated Objects: ToF-m² MC generator

```
TSToFGenerator tofGen("tofGen",Mom,ToF);  
  
tofGen.SetParticle("proton", "p", TSParticle::MassP);  
tofGen.SetParticle("kaon", "K", TSParticle::MassKPlus);  
tofGen.SetParticle("pi", "pi", TSParticle::MassPi);
```

Generic Class to generate m^2 distribution

1-Based on Parametrization of Mean
and Resolution as a function of the Momentum

2- Based on m^2 calculation from time of flight

(features to add bias smearings etc..)

```
TSToFGenerator tofGen("tofGen",Mom,ToF);

tofGen.SetParticle("proton","p",TSParticle::MassP);
tofGen.SetParticle("kaon","K",TSParticle::MassKPlus);
tofGen.SetParticle("pion","#pi",TSParticle::MassPiPlus);

//generic particle
tofGen.SetParticle("part","part",TSParticle::MassKPlus);

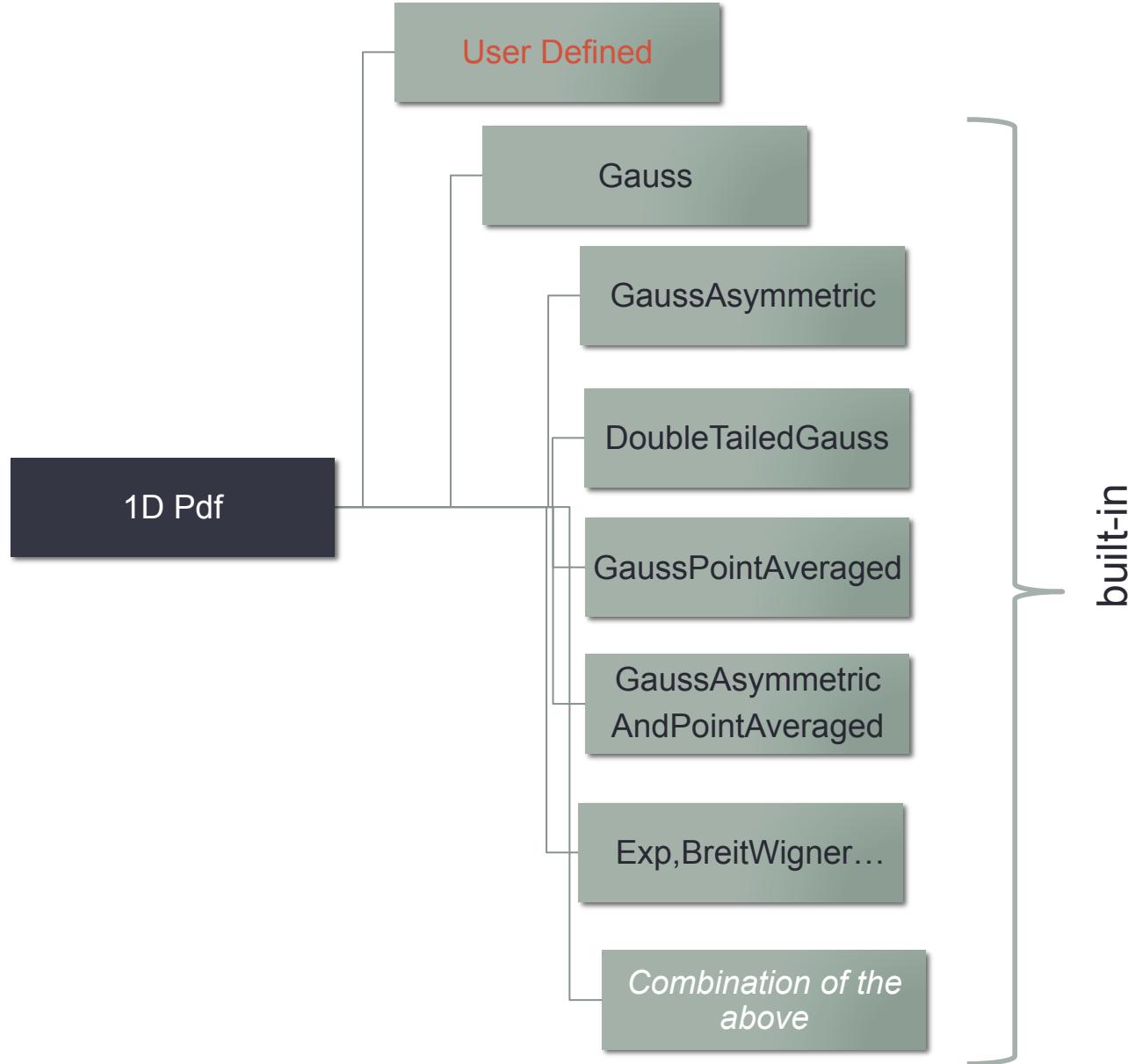
tofGen.SetMass2MeanFunction("proton","linear",m2_m_p0[0],m2_m_p1[0]);
tofGen.SetMass2MeanFunction("kaon","linear",m2_m_p0[1],m2_m_p1[1]);
tofGen.SetMass2MeanFunction("pion","linear",m2_m_p0[2],m2_m_p1[2]);

tofGen.SetMass2WidthFunction("proton","parab",m2_s_p0[0],m2_s_p1[0],m2_s_p2[0]);
tofGen.SetMass2WidthFunction("kaon","parab",m2_s_p0[1],m2_s_p1[1],m2_s_p2[1]);
tofGen.SetMass2WidthFunction("pion","parab",m2_s_p0[2],m2_s_p1[2],m2_s_p2[2]);

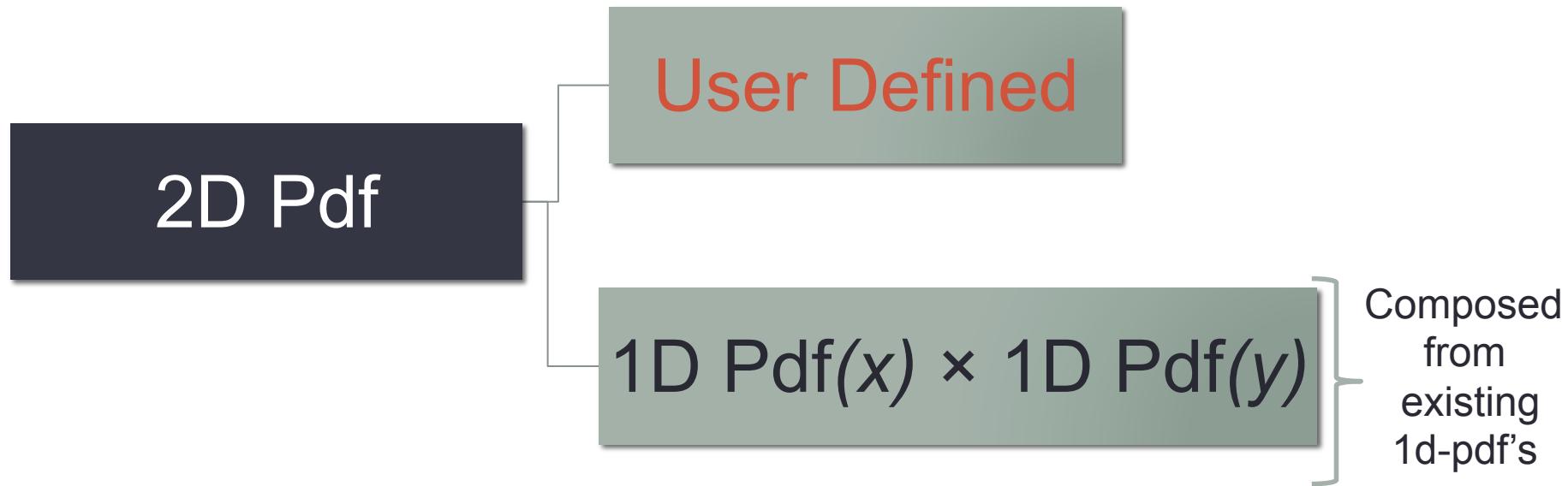
tofGen.SetMass2DistributionFunction("part","double_tailed_gaus");
tofGen.SetParamFunction("part", 0 , "linear", m2_m_p0[0],m2_m_p1[0]);
tofGen.SetParamFunction("part", 1 , "parab", m2_s_p0[0],m2_s_p1[0],m2_s_p2[0]);
tofGen.SetParamFunction("part", 2 , "parab", m2_s_p0[2],m2_s_p1[2],3*m2_s_p2[2]);
tofGen.SetParamFunction("part", 3 , "const", 0.5);
```

Distribution Functions

Distribution Functions



Distribution Functions



Distribution Functions

Model
(1D or 2D)

$$N_1 \times \text{Pdf}_1 + N_2 \times \text{Pdf}_2 + N_3 \times \text{Pdf}_3$$

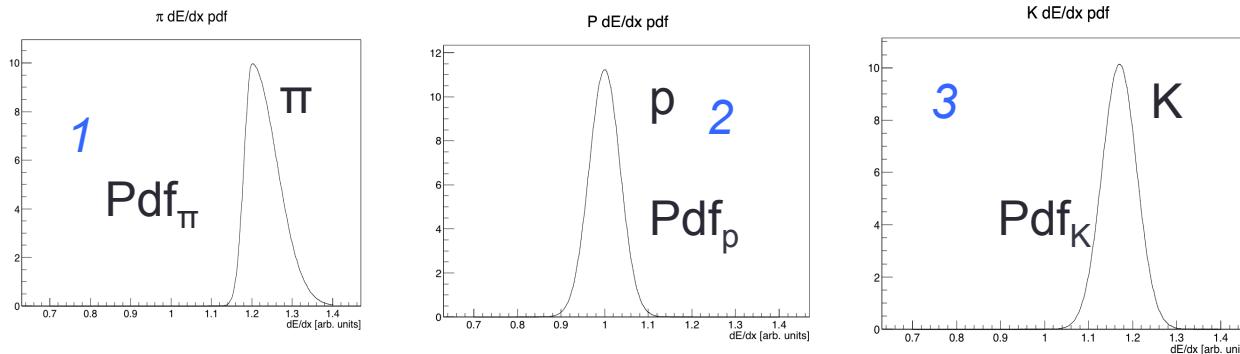
Most General: A function of Pdf's and Coefficients

Most Common: A Linear Combination of Pdf's with Yields



// Build 1D dEdx Single Particles Pdf

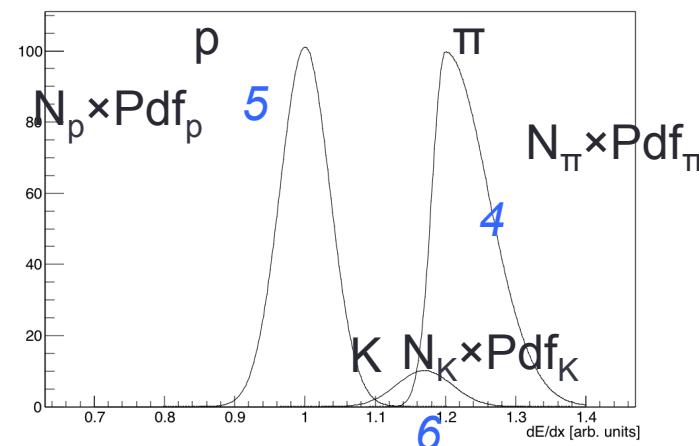
```
TSFunc1DAsymGaussPdf Pi_dEdx_Pdf("Pi_dEdx_Pdf", "#pi dE/dx pdf", dEdx_x,dEdx_Pi_mean, dEdx_Pi_sig, dEdx_sig_delta); 1
TSFunc1DGaussPdf P_dEdx_Pdf("P_dEdx_Pdf", "P dE/dx pdf", dEdx_x, dEdx_P_mean, dEdx_P_sig); 2
TSFunc1DGaussPdf K_dEdx_Pdf("K_dEdx_Pdf", "K dE/dx pdf", dEdx_x, dEdx_K_mean, dEdx_K_sig); 3
```



K_dEdx_Pdf.BuildGraph();

// Build Single Particle dEdx Model

```
TSFuncModel1D Model_dEdx_Pi("Model_dEdx_Pi", "model dE/dx #pi", TSArgList(Pi_dEdx_Pdf), TSArgList(NPi)); 4
TSFuncModel1D Model_dEdx_P("Model_dEdx_P", "model dE/dx P", TSArgList(P_dEdx_Pdf), TSArgList(NP)); 5
TSFuncModel1D Model_dEdx_K("Model_dEdx_K", "model dE/dx K", TSArgList(K_dEdx_Pdf), TSArgList(NK)); 6
```

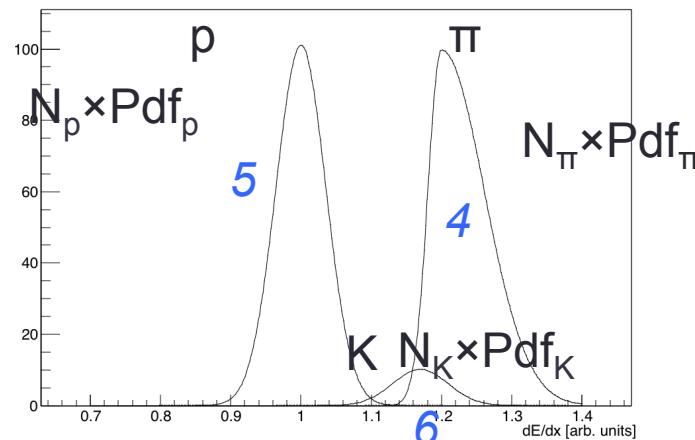
 dE/dx

// Build Single Particle dEdx Model

TSFuncModel1D Model_dEdx_Pi("Model_dEdx_Pi", "model dE/dx #pi", TSArgList(Pi_dEdx_Pdf), TSArgList(NPi));

TSFuncModel1D Model_dEdx_P("Model_dEdx_P", "model dE/dx P", TSArgList(P_dEdx_Pdf), TSArgList(NP)); 5

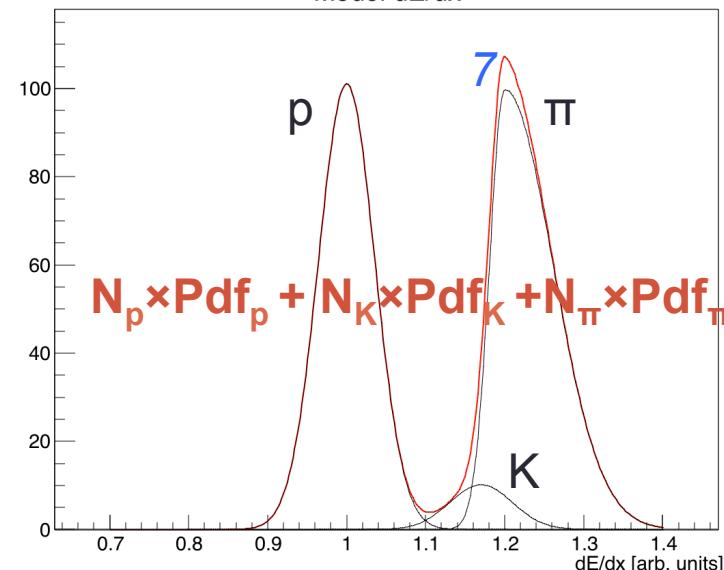
TSFuncModel1D Model_dEdx_K("Model_dEdx_K", "model dE/dx K", TSArgList(K_dEdx_Pdf), TSArgList(NK)); 6



//Build 1D dEdx Model Pdf

TSFuncModel1D Model_dEdx("Model_dEdx", "model dE/dx", TSArgList(P_dEdx_Pdf, K_dEdx_Pdf, Pi_dEdx_Pdf), TSArgList(NP, NK, NPi));

Model dE/dx



Model_dEdx.GetGraph();

dE/dx

TSFunc1DGaussPdf Pi_m2_Pdf("Pi_m2_Pdf", "#pi m^2 pdf", m2_x, m2_Pi_mean, m2_Pi_sig); 1

TSFunc1DGaussPdf P_m2_Pdf("P_m2_Pdf", "P m^2 pdf", m2_x, m2_P_mean, m2_P_sig); 2

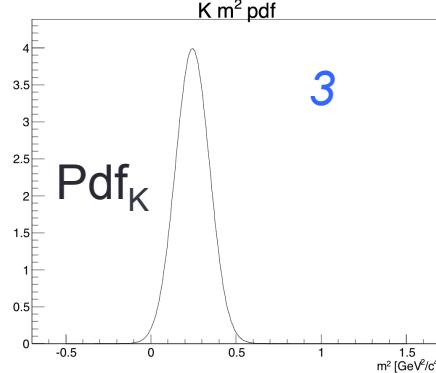
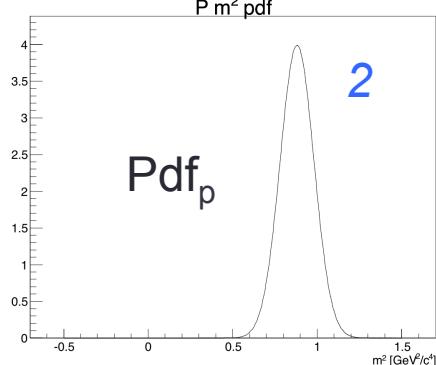
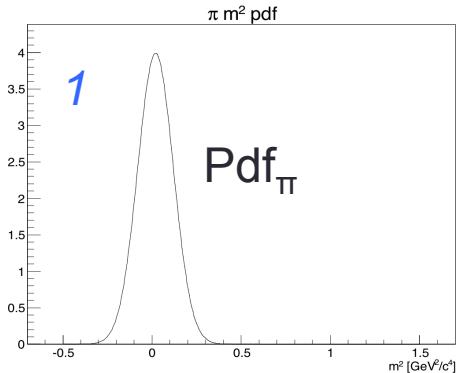
TSFunc1DGaussPdf K_m2_Pdf("K_m2_Pdf", "K m^2 pdf", m2_x, m2_K_mean, m2_K_sig); 3

TSFuncModel1D Model_m2("Model_m2", "model m^2", TSArgList(P_m2_Pdf, K_m2_Pdf, Pi_m2_Pdf), TSArgList(NP, NK, NPi)); 7

TSFuncModel1D Model_m2_Pi("Model_m2_Pi", "model m^2 #pi", TSArgList(Pi_m2_Pdf), TSArgList(NPi)); 4

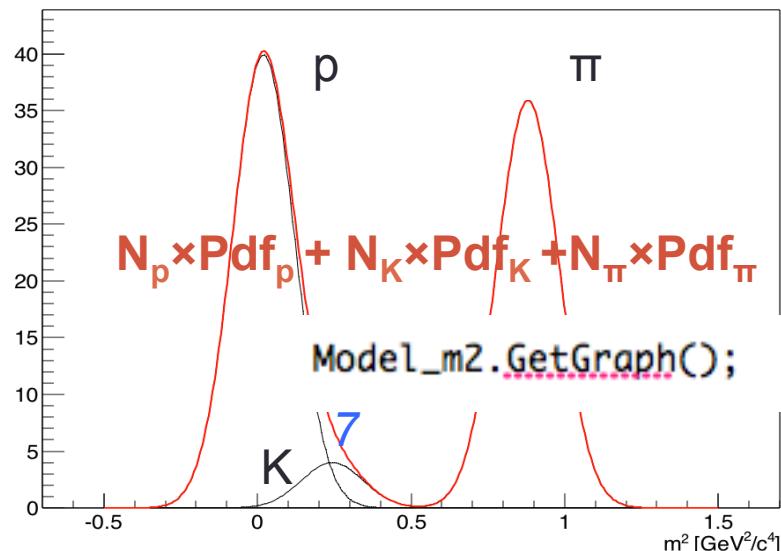
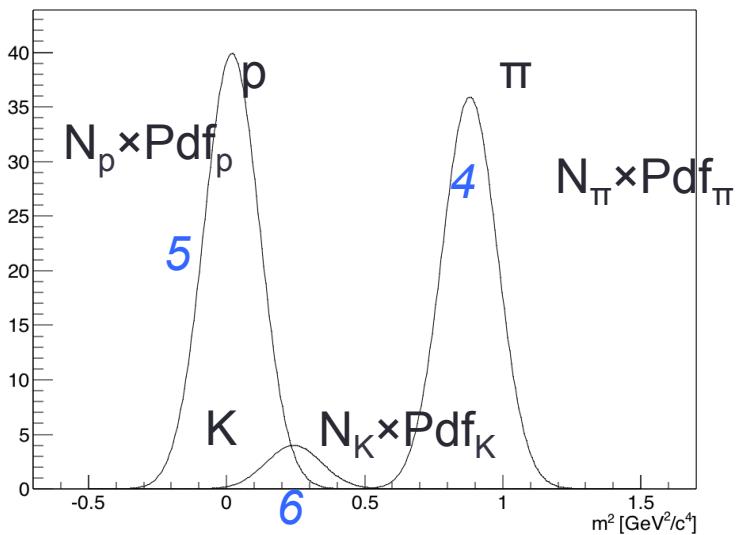
TSFuncModel1D Model_m2_P("Model_m2_P", "model m^2 P", TSArgList(P_m2_Pdf), TSArgList(NP)); 5

TSFuncModel1D Model_m2_K("Model_m2_K", "model m^2 K", TSArgList(K_m2_Pdf), TSArgList(NK)); 6



model m^2

model m^2

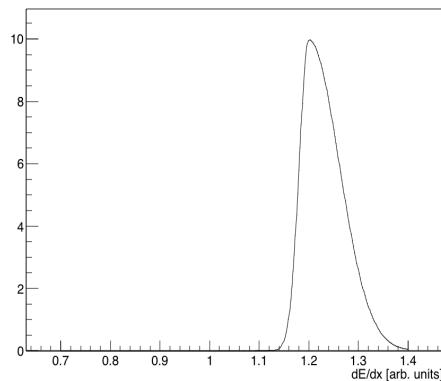


// Buil 2D Single Particle Pdf

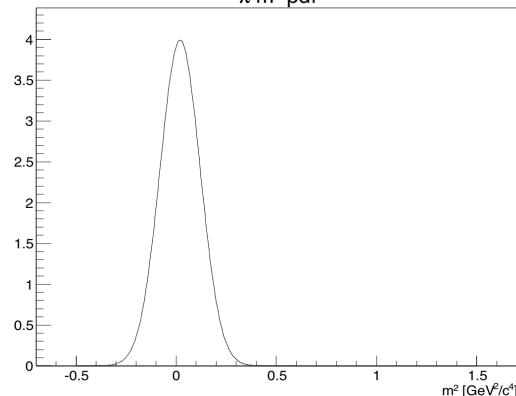


```
TSFunc2DProd Pi_2D_Pdf("Pi_2D_Pdf","#pi dE/dx-m^2 pdf",Pi_dEdx_Pdf,Pi_m2_Pdf);
```

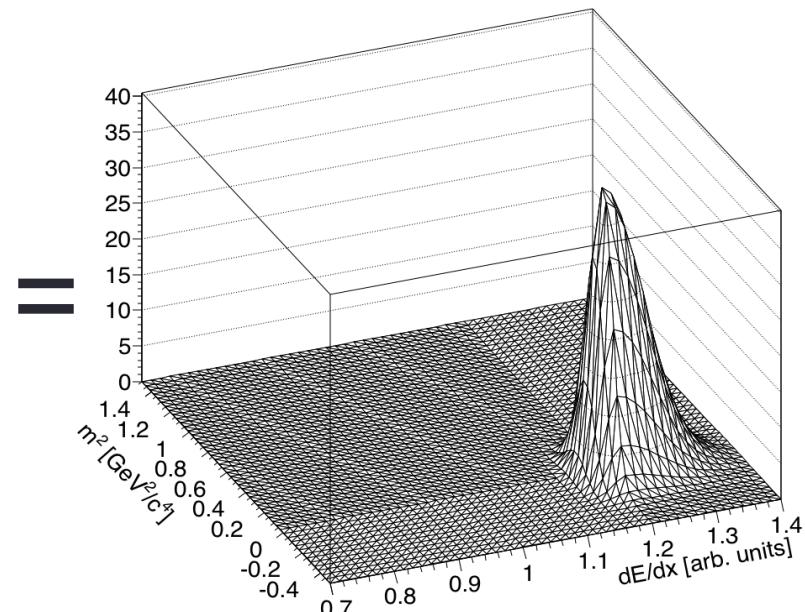
$\pi dE/dx$ pdf



πm^2 pdf



$\pi dE/dx-m^2$ pdf

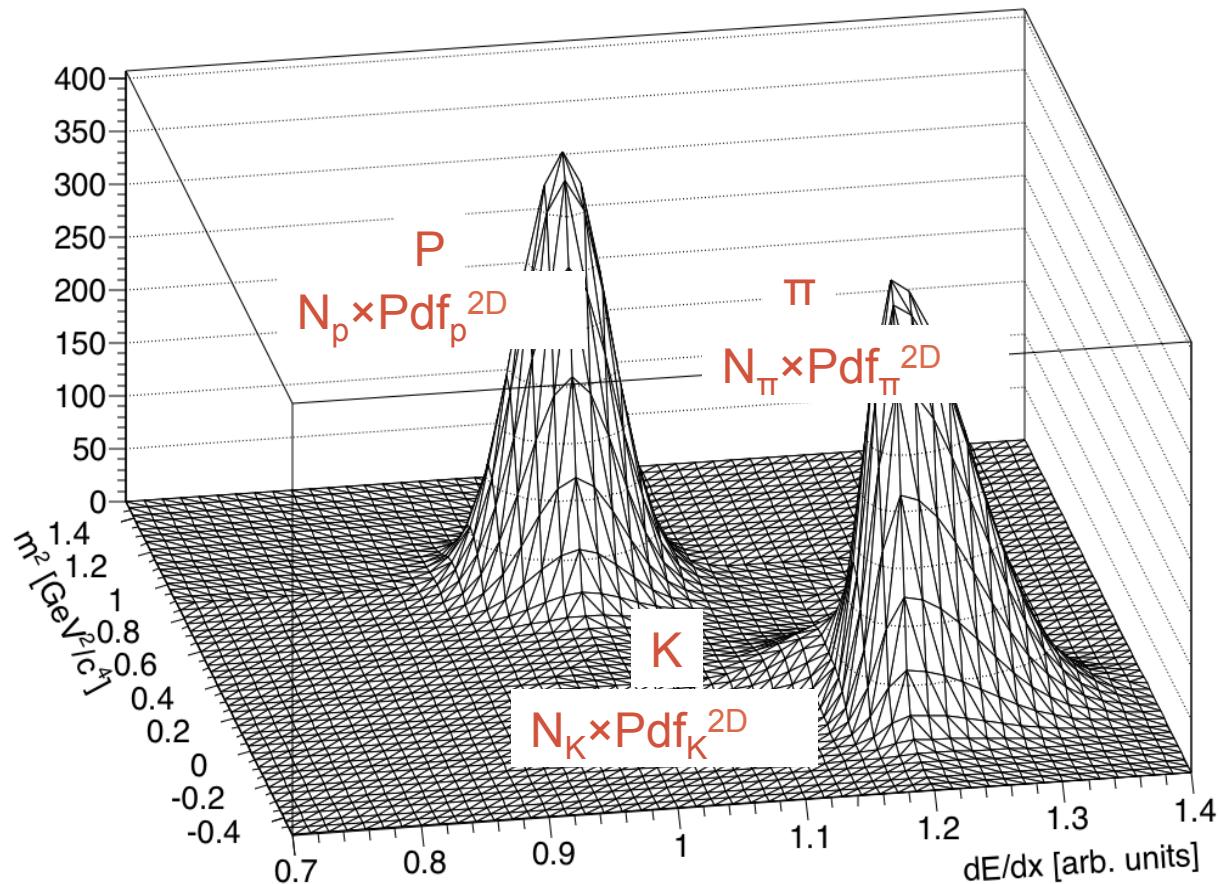


```
Pi_2D_Pdf.BuildGraph2D();
```

dE/dx vs ToF

// Build 2D Model

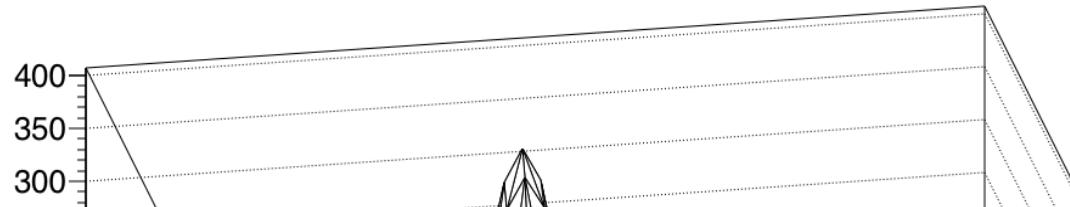
TSFuncModel2D Model_2D("Model_2D","model dEdx-m^2",TSArgList(P_2D_Pdf,K_2D_Pdf,Pi_2D_Pdf),TSArgList(NP,NK,NPi));

model dEdx-m²

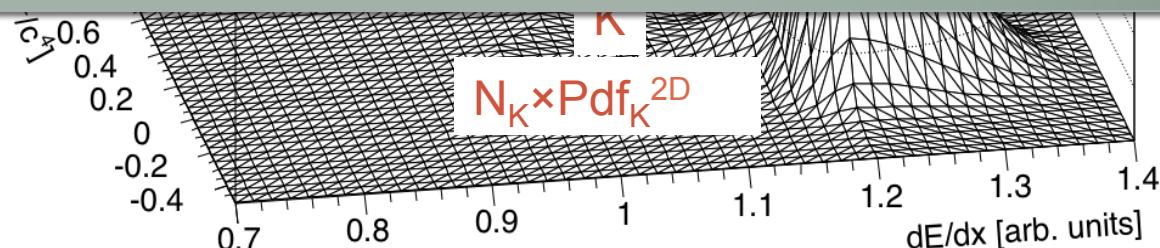
Model_2D.BuildGraph2D();

```
TSFuncModel2D Model_2D("Model_2D","model dEdx-m^2",TSArgList(P_2D_Pdf,K_2D_Pdf,Pi_2D_Pdf),TSArgList(NP,NK,NPi));
```

model $dEdx-m^2$



Model is a Client of the Pdf's and Yields:
If the definition of one Pdf is changed
You don't have to change the implementation of the
Model and of all the "intermediate" objects



Model_2D.BuildGraph2D();

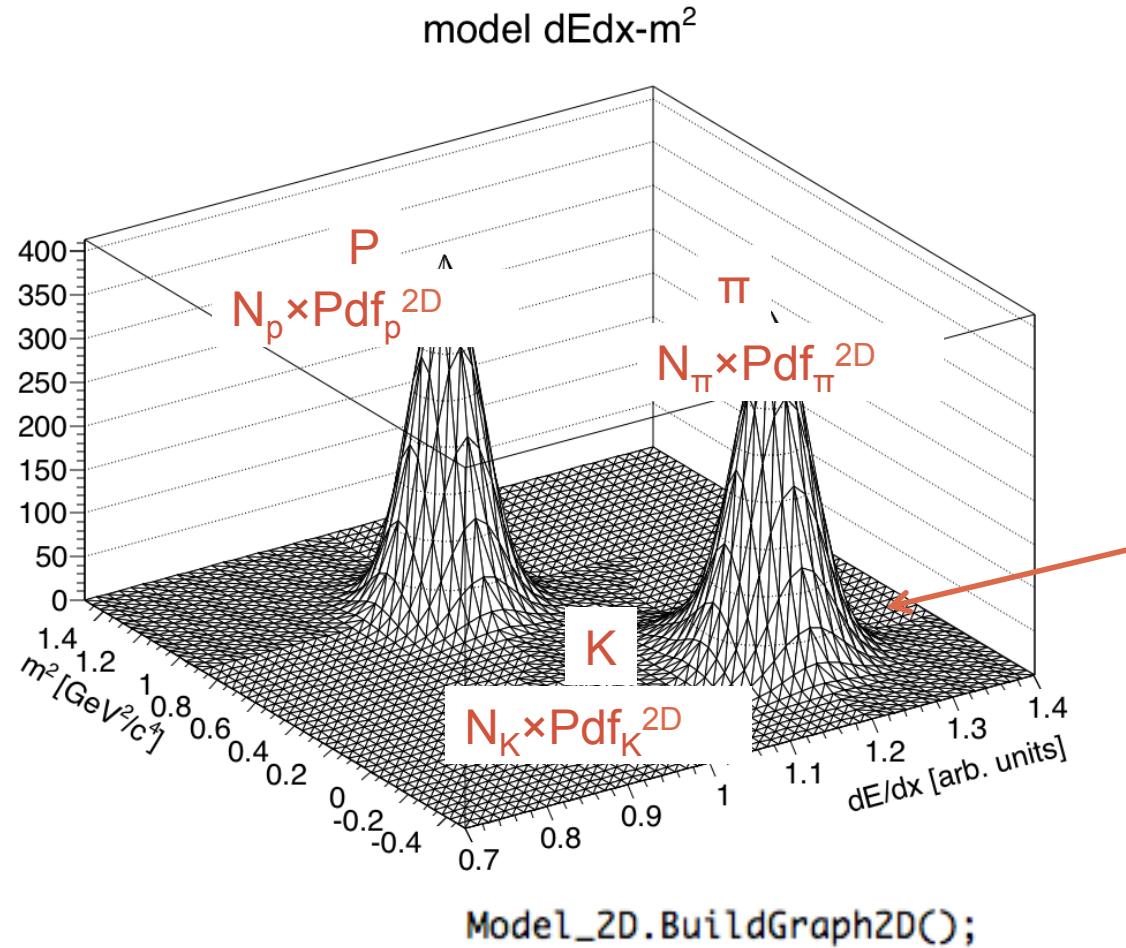
check it out

```
Model_2D.Print();
```

```
===== Print =====
Model: Model_2D model dEdx-m^2
Dimension: 2
Normalization: 1 bin widths: 1 1
Variables: dEdx_x m2_x
Num. of Functions: 3 Num. of Coefs: 3
Formula: linear
Func: 0 P_2D_Pdf
Func: 1 K_2D_Pdf
Func: 2 Pi_2D_Pdf
Coef: 0 NP: 9
Coef: 1 NK: 1
Coef: 2 NPi: 10
Num of Primary Params: 15
dEdx_Pi_mean: #pi dE/dx mean m_{dE/dx}-#pi value: 1.2
dEdx_mean_PtoPi: dE/dx mean m_{dE/dx}-P#pi value: -0.2
dEdx_sig_0: dE/dx width #sigma_{dE/dx} value: 0.04
dEdx_sig_alpha: #alpha scaling factor #alpha value: 0.65
m2_P_mean: P ToF m^2 mean m^2 value: 0.880354
m2_P_sig: P ToF m^2 width #sigma_{m^2} value: 0.1
dEdx_mean_KtoPi: dE/dx mean m_{dE/dx}-K#pi value: -0.03
m2_K_mean: K ToF m^2 mean m^2 value: 0.243707
m2_K_sig: K ToF m^2 width #sigma_{m^2} value: 0.1
dEdx_sig_delta: asymmetry parameter #delta:[0,0.99] value: 0.52
m2_Pi_mean: #pi ToF m^2 mean m^2 value: 0.0194798
m2_Pi_sig: #pi ToF m^2 width #sigma_{m^2} value: 0.1
NP: P yield N_{P} value: 9
NK: K yield N_{K} value: 1
NPi: #pi yield N_{#pi} value: 10
=====
```

```
//TSFunc1DAsymGaussPdf Pi_dEdx_Pdf("Pi_dEdx_Pdf","#pi dE/dx pdf",dEdx_x,dEdx_Pi_mean,dEdx_Pi_sig,dEdx_sig_delta);  
TSFunc1DGaussPdf Pi_dEdx_Pdf("Pi_dEdx_Pdf","#pi dE/dx pdf",dEdx_x,dEdx_Pi_mean,dEdx_Pi_sig);
```

Make dEdx-pdf of a simple Gaussian, Draw Model again



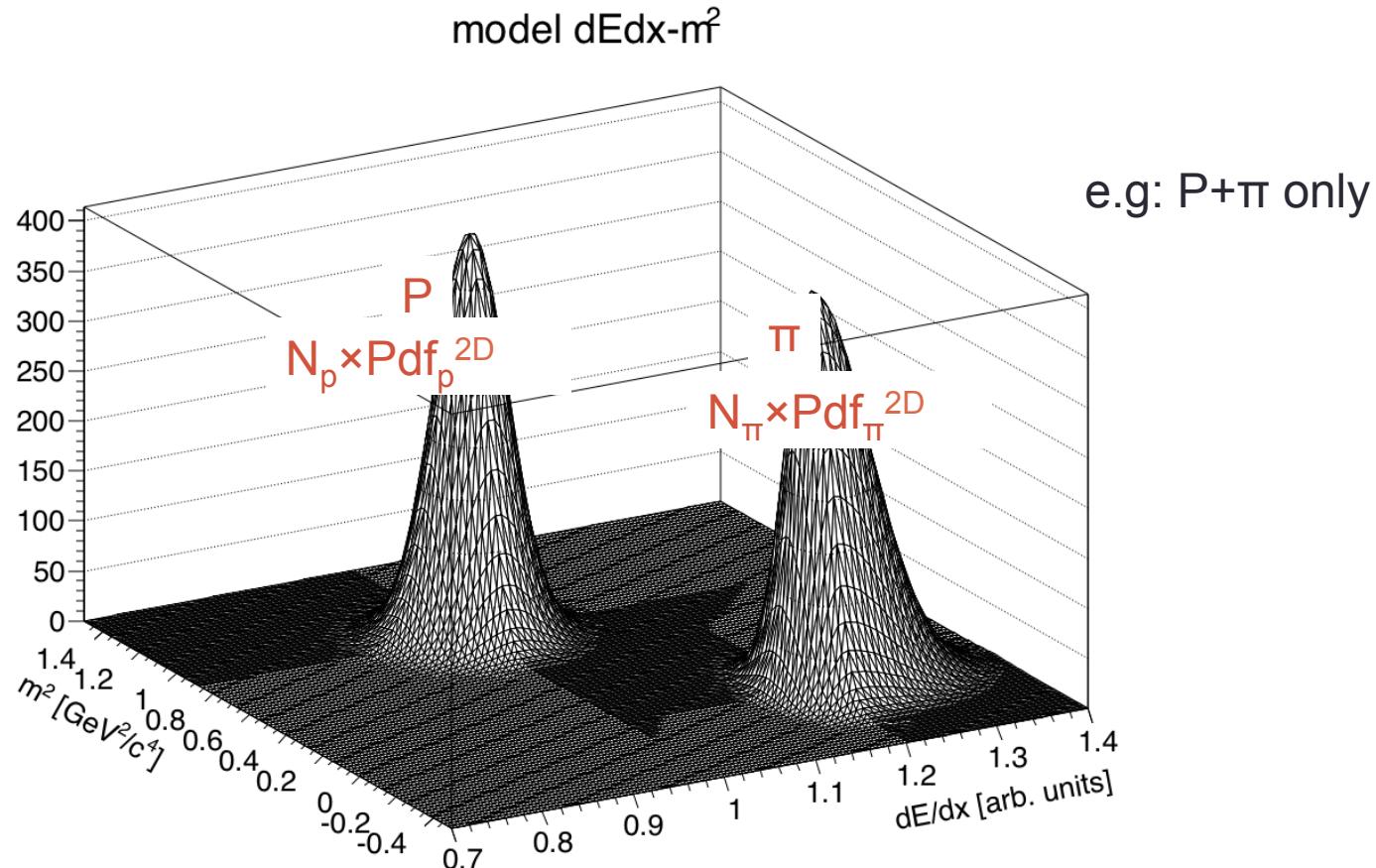
check it out

```
Model_2D.Print();
```

```
===== Print =====
Model: Model_2D model dEdx-m^{2}
Dimension: 2
Normalization: 1 bin widths: 1 1
Variables: dEdx_x m2_x
Num. of Functions: 3 Num. of Coefs: 3
Formula: linear
Func: 0 P_2D_Pdf
Func: 1 K_2D_Pdf
Func: 2 Pi_2D_Pdf
Coef: 0 NP: 9
Coef: 1 NK: 1
Coef: 2 NPi: 10
Num of Primary Params: 14
dEdx_Pi_mean: #pi dE/dx mean m_{dE/dx}-#pi value: 1.2
dEdx_mean_PtoPi: dE/dx mean m_{dE/dx}-P#pi value: -0.2
dEdx_sig_0: dE/dx width #sigma_{dE/dx} value: 0.04
dEdx_sig_alpha: #alpha scaling factor #alpha value: 0.65
m2_P_mean: P ToF m^{2} mean m^{2} value: 0.880354
m2_P_sig: P ToF m^{2} width #sigma_{m^{2}} value: 0.1
dEdx_mean_KtoPi: dE/dx mean m_{dE/dx}-K#pi value: -0.03
m2_K_mean: K ToF m^{2} mean m^{2} value: 0.243707
m2_K_sig: K ToF m^{2} width #sigma_{m^{2}} value: 0.1
m2_Pi_mean: #pi ToF m^{2} mean m^{2} value: 0.0194798
m2_Pi_sig: #pi ToF m^{2} width #sigma_{m^{2}} value: 0.1
NP: P yield N_{P} value: 9
NK: K yield N_{K} value: 1
NPi: #pi yield N_{#pi} value: 10
=====
=====
```

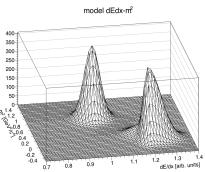
```
TSFuncModel2D Model_2D_PPi("Model_2D_PPi","model dEdx-m^2",TSArgList(P_2D_Pdf,Pi_2D_Pdf),TSArgList(NP,NPi));
```

Prepare as many Model as needed...



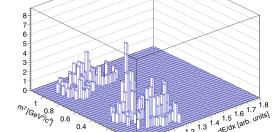
```
Model_2D_PPi.BuildGraph2D();
```

Minimization Function



1D or 2D

Model

Primary Params
($m^2, \sigma_{m2}, \sigma_{dEdx}, \dots$) $dE/dx \text{ m}^2$ 

1D or 2D

Data

Binned
or Unbinned

Minimization Function

 x^2

Likelihood-Binned
Likelihood-Unbinned

.... user defined



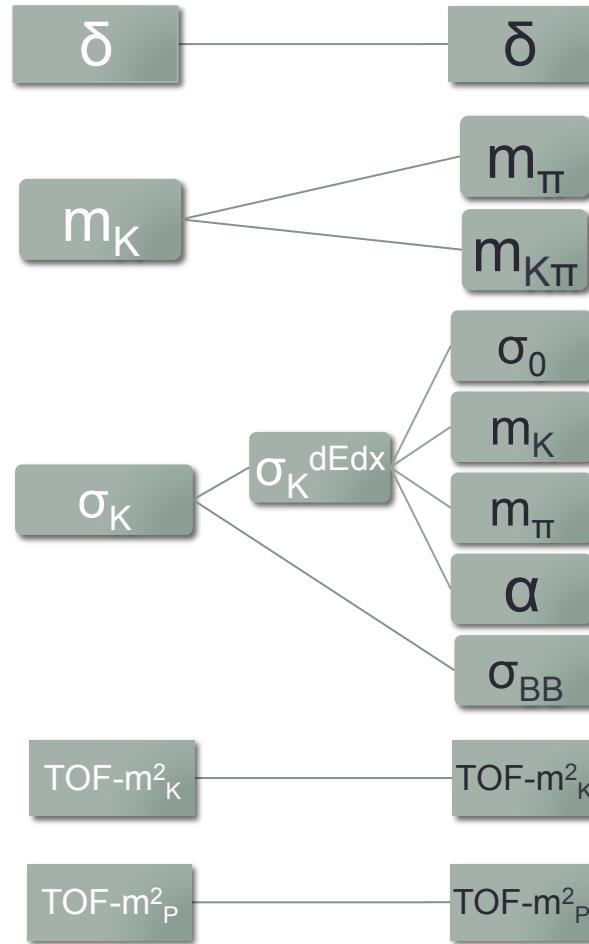
Calculated value
for a given set of Primary Params

Params & Primary Params

Pdf's, Model and then the Minimization Functions depend on the primary params

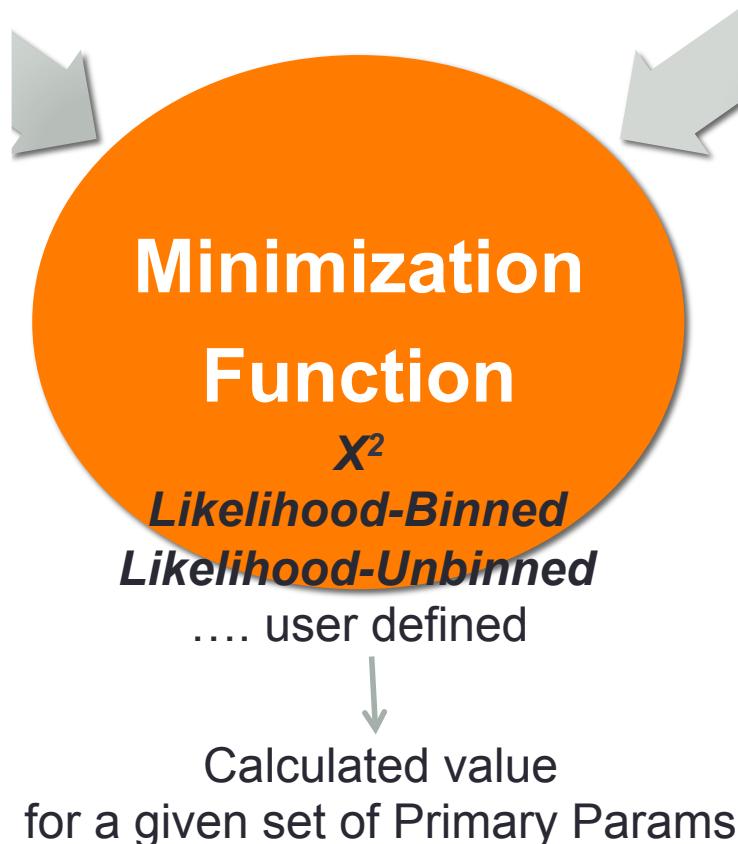
those are the params that will be varied during the Minimization (i.e. by Minuit)

Param Primary Params



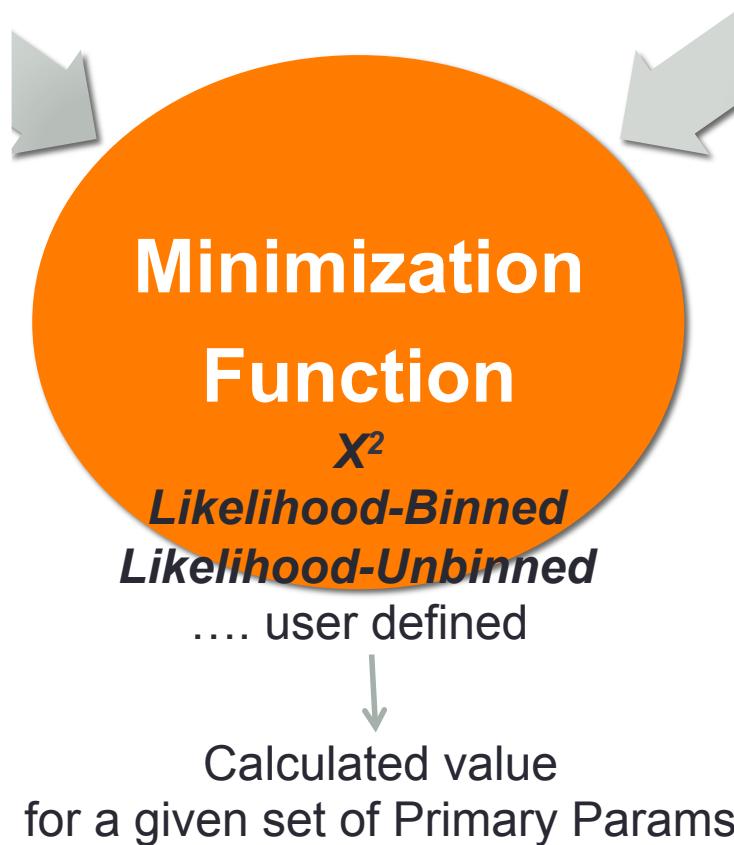
Primary Params List is updated if the definitions of the underlying functions and/or params is changed

```
TSChiSquareCalculator myChi2_x("myChi2_x", "myChi2_x");
```



```
TSChiSquareCalculator myChi2_x("myChi2_x", "myChi2_x");
```

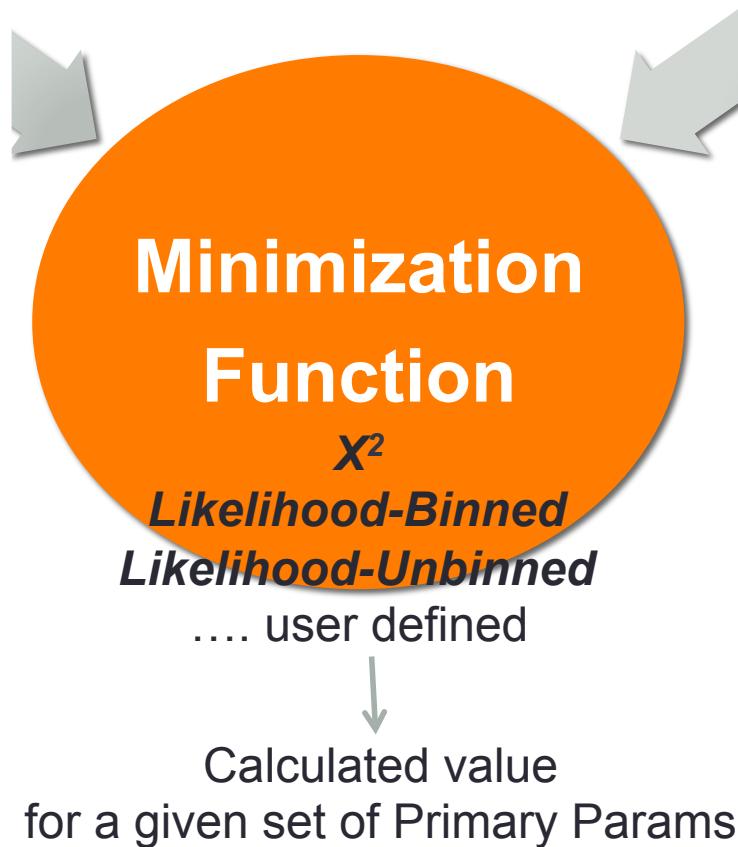
```
myChi2_x.SetModel(Model_dEdx);
```



```
TSChiSquareCalculator myChi2_x("myChi2_x", "myChi2_x");
```

```
myChi2_x.SetModel(Model_dEdx);
```

```
myChi2_x.SetBinnedData(myData, TSArgList(dEdx_x));
```



```
TSChiSquareCalculator myChi2_x("myChi2_x","myChi2_x");
myChi2_x.SetModel(Model_dEdx);
myChi2_x.SaveHistory(true);
myChi2_x.SetBinnedData(myData, TSArgList(dEdx_x));
```

```
myChi2_x.GetModel()->Print();
```

```
TSLikelihoodCalculator myLL_x("myLL_x","myLL_x");
myLL_x.SetModel(Model_dEdx);
myLL_x.SaveHistory(true);
myLL_x.SetUnBinnedData(myData, TSArgList(dEdx_x));
```

```
myLL_x.GetModel()->Print();
```

```
TSChiSquareCalculator myChi2_2d("myChi2_2d","myChi2_2d");
myChi2_2d.SetModel(Model_2D);
myChi2_2d.SaveHistory(true);
myChi2_2d.SetBinnedData(myData, TSArgList(dEdx_x,m2_x));
```

```
myChi2_2d.GetModel()->Print();
```

```
TSLikelihoodCalculator myLL_2d("myLL_2d","myLL_2d");
myLL_2d.SetModel(Model_2D);
myLL_2d.SaveHistory(true);
myLL_2d.SetBinnedData(myData, TSArgList(dEdx_x,m2_x));
myLL_2d.GetModel()->Print();
```

χ^2 1D dE/dx

MLL, 1D dE/dx

Unbinned

χ^2 2D dE/dx

MLL, 2D dE/dx

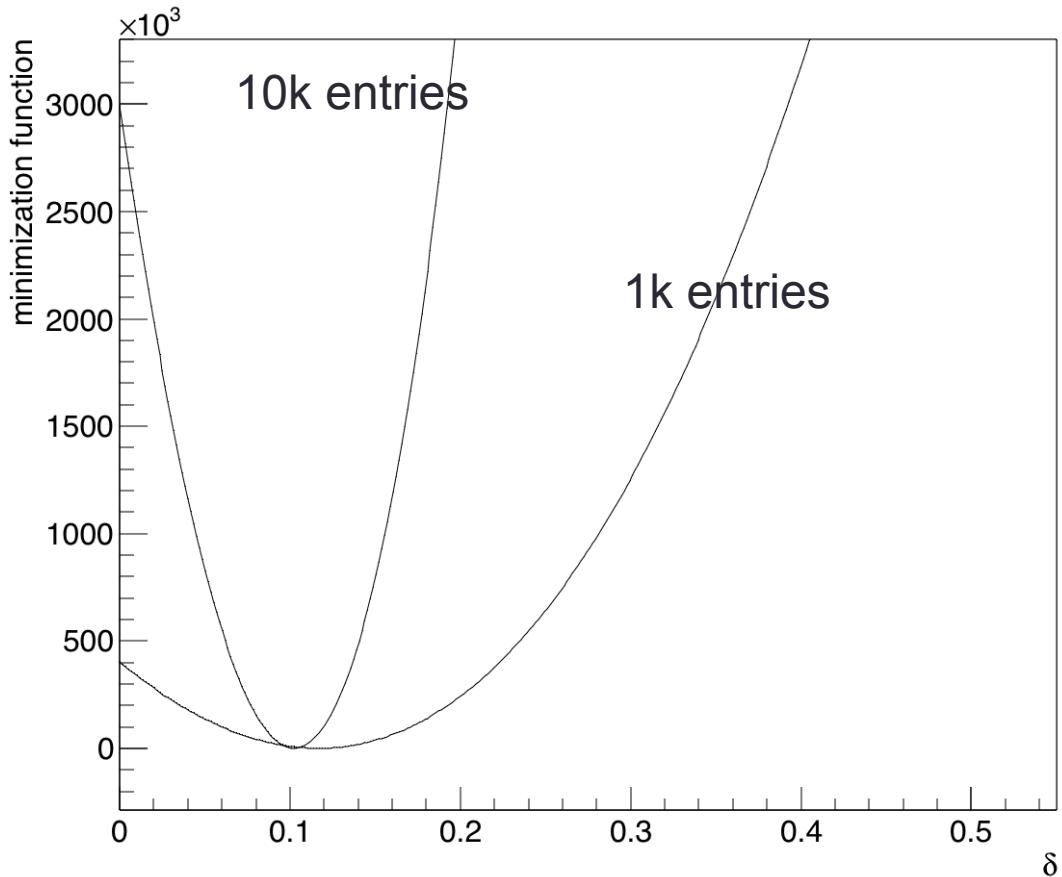
Parameters Scanning

```
dEdx_sig_delta.SetValue(0.1);

myChi2_x.SetNormalizedScannedMin();

TGraph* gScan_delta=myChi2_x.ScanParameter(dEdx_sig_delta,300,0.,0.5);
gScan_delta->Write();

double fmin,par_min;
myChi2_x.GetScannedMinima(fmin,&par_min);
```

param scan: δ 

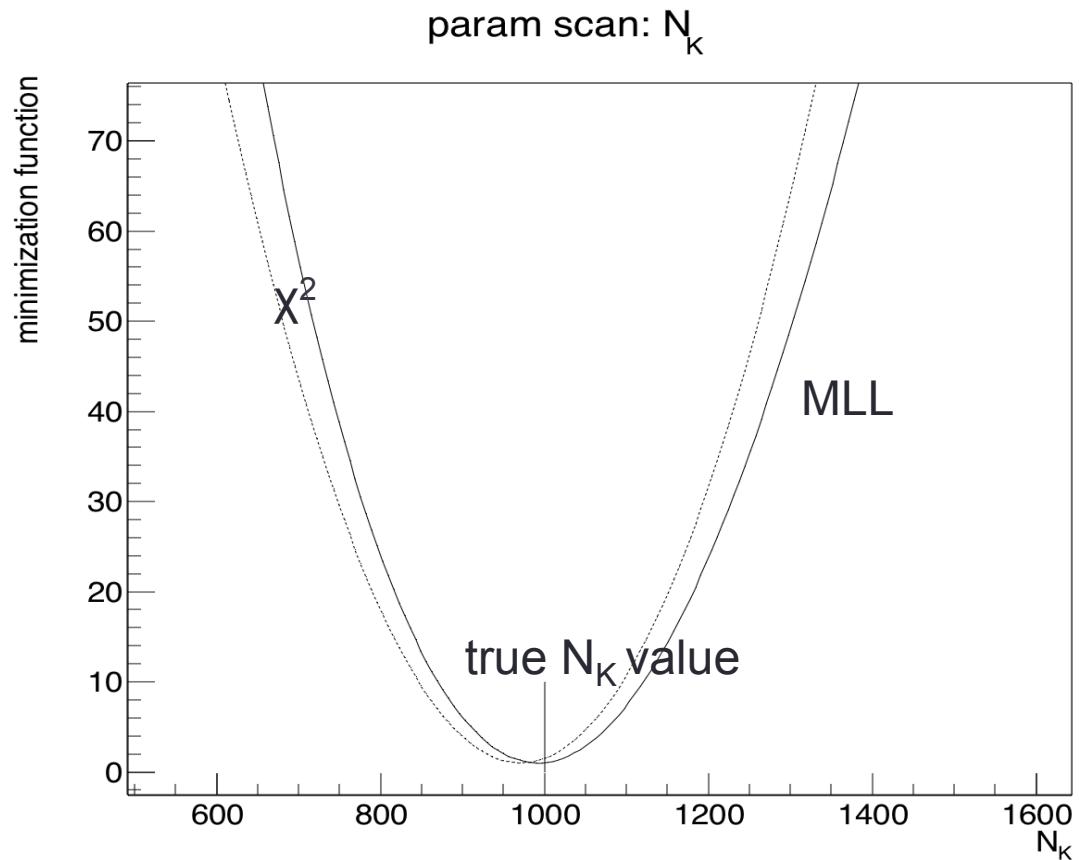
Parameters Scanning

Yield determination: χ^2 vs MLL

```
TGraph* gScan_NK_LL=myLL_x.ScanParameter(NK,300,0.,3000);  
gScan_NK_LL->Write();
```

```
TGraph* gScan_NK_Chi2=myChi2_x.ScanParameter(NK,300,0.,3000);  
gScan_NK_Chi2->Write();
```

Create two Minimizers,
then used them in parallel



Parameters Scanning

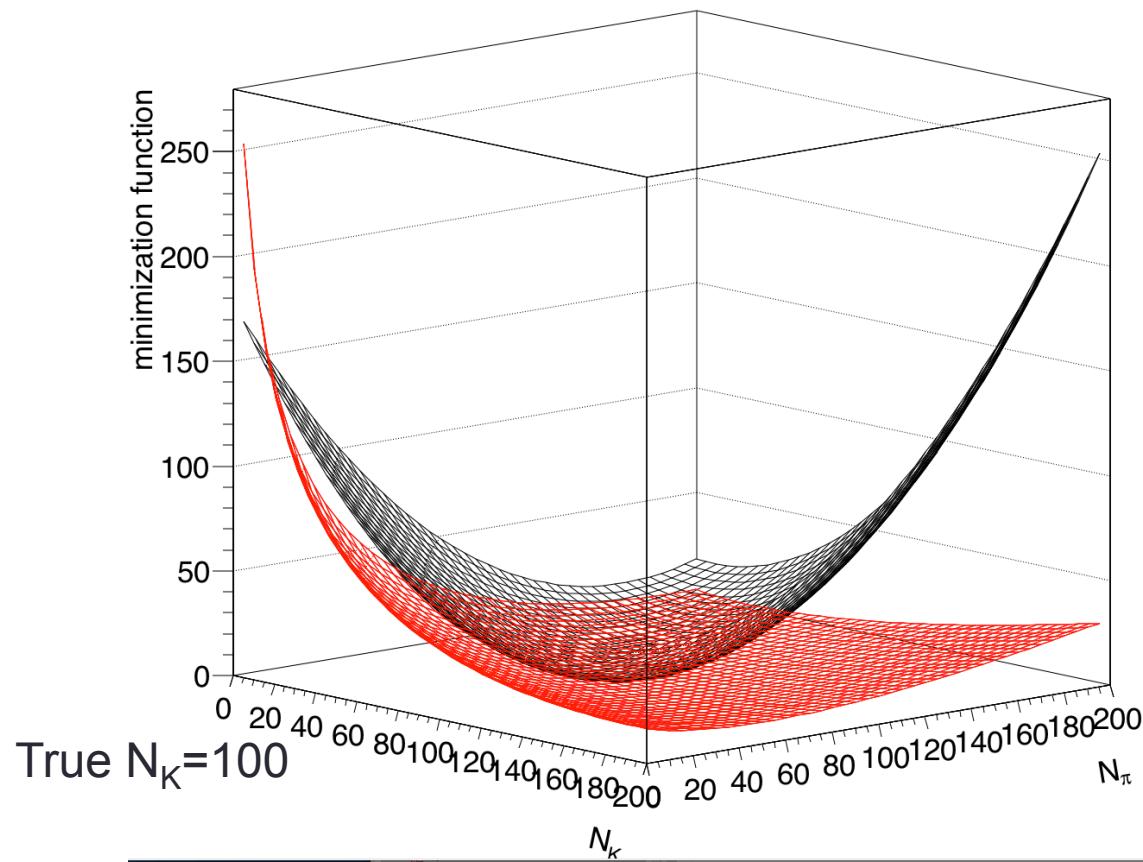
Yield determination: χ^2 vs MLL

```
TGraph2D* gScan_NK_NPi_LL=myLL_x.ScanParameters(NK,NPi,100,0.,200,100,0.,200;  
gScan_NK_NPi_LL->Write();
```

```
TGraph2D* gScan_NK_NPi_Chi2=myChi2_x.ScanParameters(NK,NPi,100,0.,200,100,0.,200);  
gScan_NK_NPi_Chi2->Write();
```

2D scan: N_K vs N_π

Create two Minimizers,
then used them in parallel



Parameters Scanning

Yield determination: 1D vs 2D Minimization Functions

```
TGraph* gScan_NK_Chi2_2D=myChi2_2d.ScanParameter(NK,300,NK_min,NK_max);
gScan_NK_Chi2_2D->Write();
```

```
TGraph* gScan_NK_LL_2D=myLL_2d.ScanParameter(NK,300,NK_min,NK_max);
gScan_NK_LL_2D->Write();
```

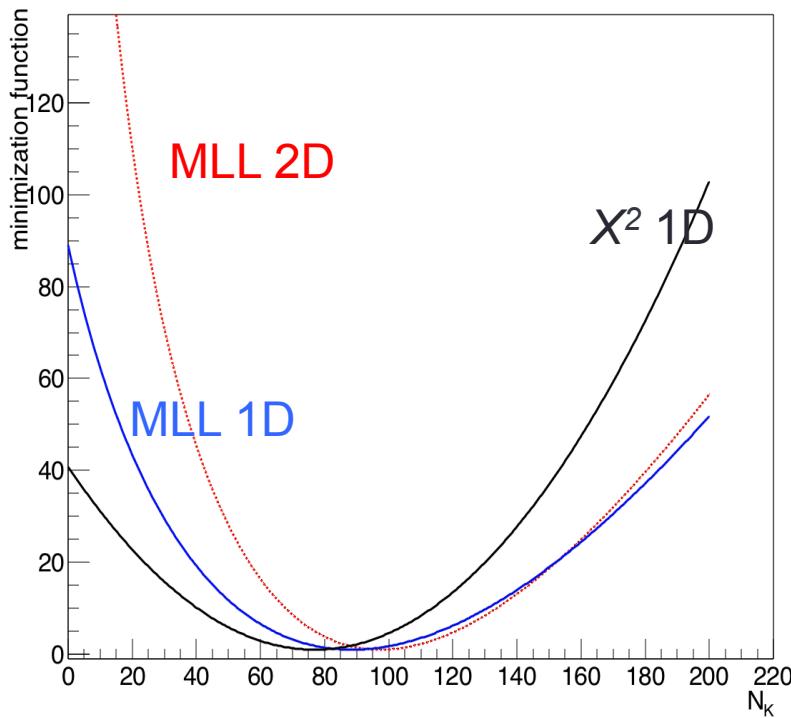
use here the
2D Minim.Funcs
prepared

Yields determination: χ^2 biased at low stat, larger errors

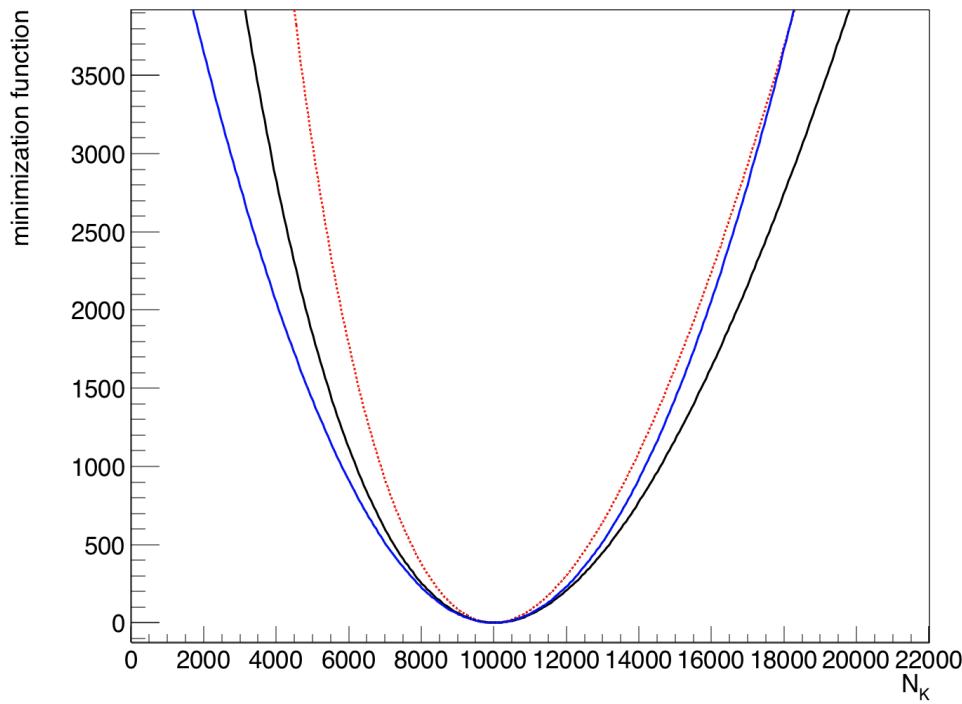
MLL 2D/1D: in 2D better peak discrimination: smaller error

param scan: N_K

param scan: N_K

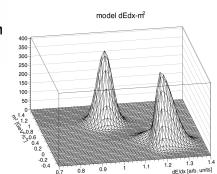


True $N_K=100$



True $N_K=10k$

Minimizers



Primary Params
 $(m^2, \sigma_{m^2}, \sigma_{dEdx}, \dots)$

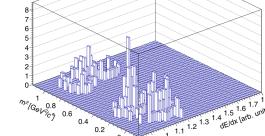
1D or 2D

Model

1D or 2D

Data

$dE/dx \cdot m^2$



Binned
or Unbinned

χ^2
Likelihood-Binned
Likelihood-Unbinned
.... user defined

Minimization
Function

Calculated value
for a given set of Primary Params

Minimizer iteration
(param variation)

$(m^2, \sigma_{m^2}, \sigma_{dEdx}, \dots)^{\text{trial}}$

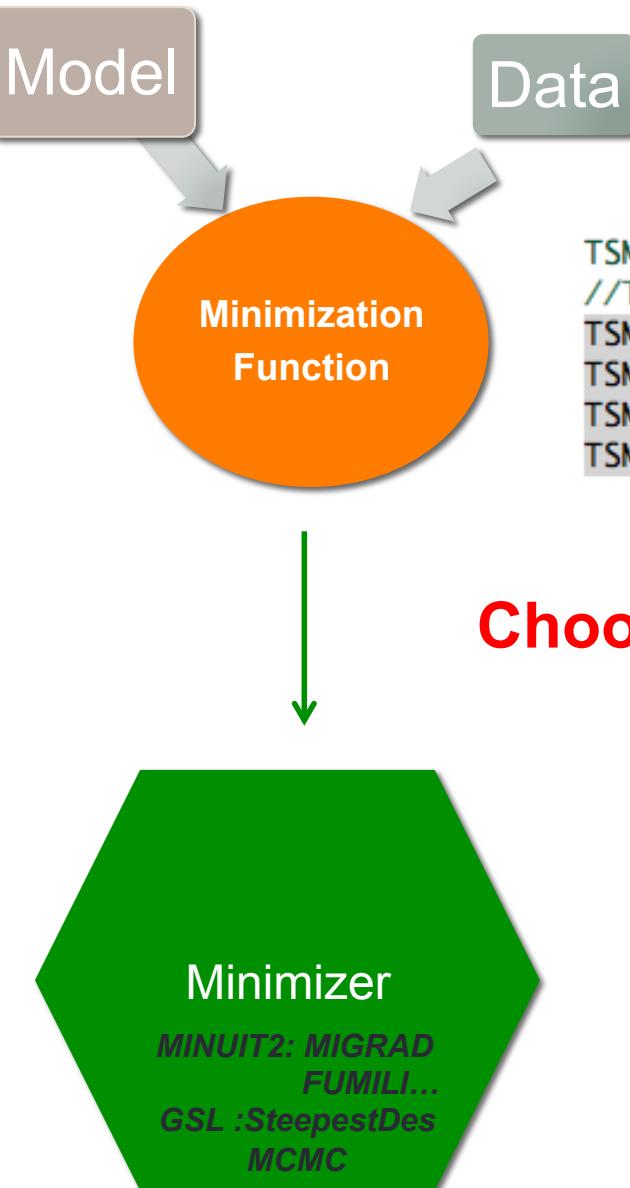
Minimizer

*MINUIT2: MIGRAD
FUMILI...
GSL :SteepestDes
MCMC*

RESULT

$(m^2, \sigma_{m^2}, \sigma_{dEdx}, \dots)^{\text{best-fit}}$

Func. Minimum



Setup Minimizer Choose the algorithm.....

```
TSMinimizer TSMin("Minuit2","Combined");  
//TSMinimizer TSMin("GSLMultiMin","SteepestDescent");  
TSMin.GetMinimizer()->SetPrintLevel(0);  
TSMin.GetMinimizer()->SetMaxFunctionCalls(1e6);  
TSMin.GetMinimizer()->SetMaxIterations(1e4);  
TSMin.GetMinimizer()->SetTolerance(0.001);
```

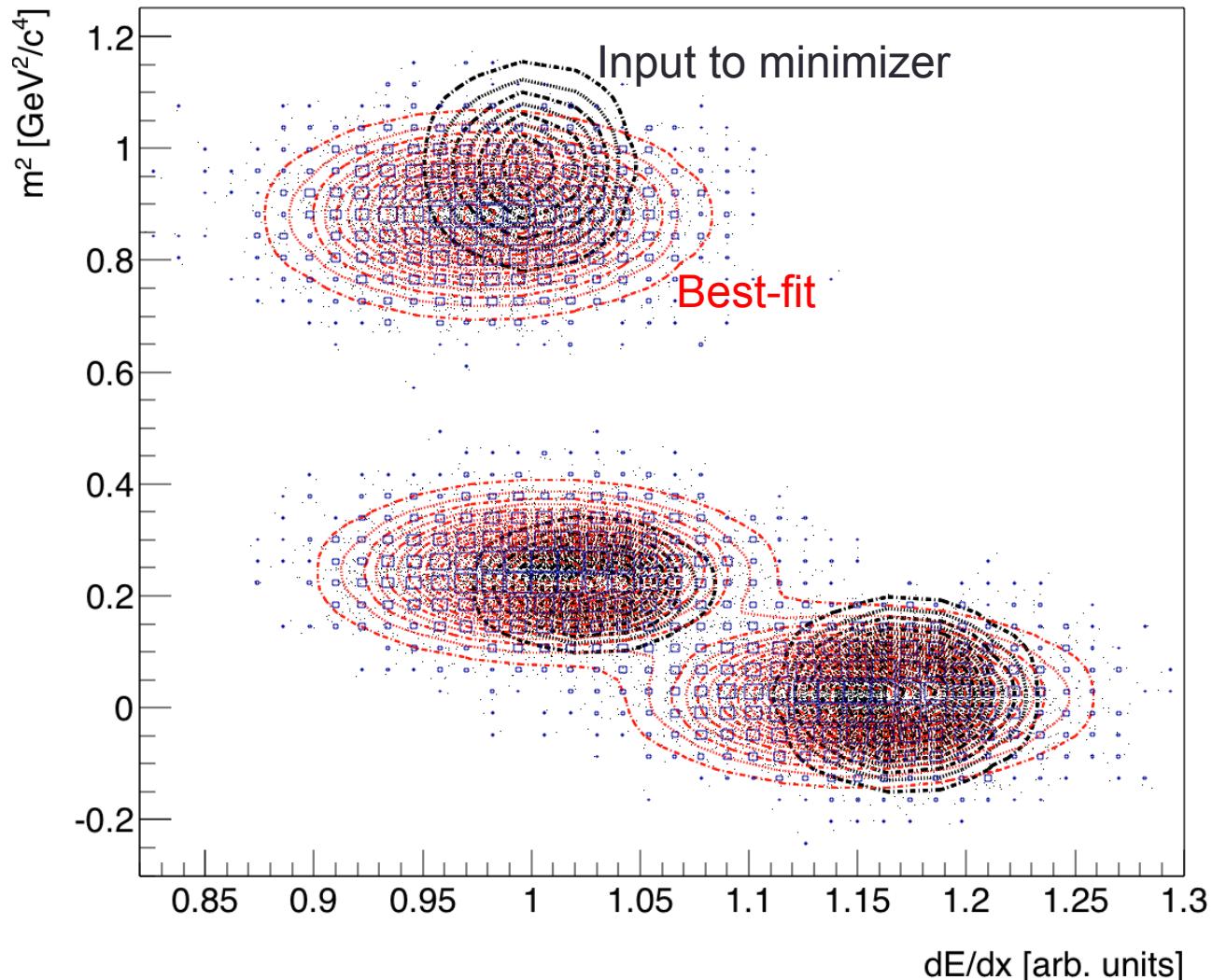
Choose the minimization function.....

```
TSMin.SetFunction(myLL_2d);  
|  
TSMin.Minimize();
```

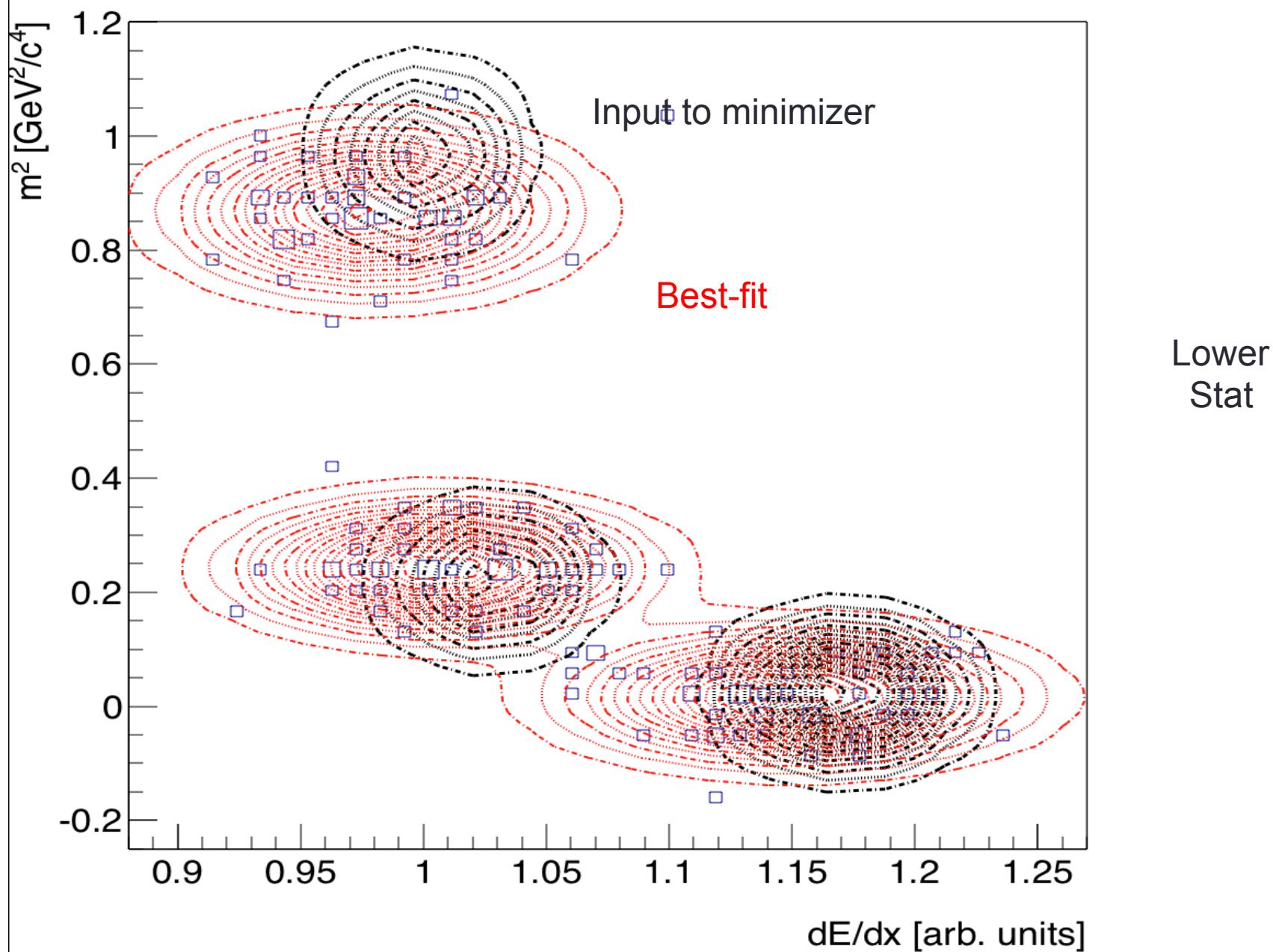
Putting things together

- 1- Setup Phase Space
- 2- Setup Data Bin Container
- 3- Setup DataSetMgr
- 4- Setup ToF Generator
- 5- Setup BetheBloch Function Mgr
- 6- Generate MC-Toy Events
- 7- Fill DataSetMgr
- 8- Setup Model
- 9- Setup Minimizer
- 10- Loop over DataSet
- 11- Fit Data

dE/dx vs ToF m^2 , $p_{\text{tot}}:[2.4,3.2]$ GeV/c $\theta:[0,20]$ mrad

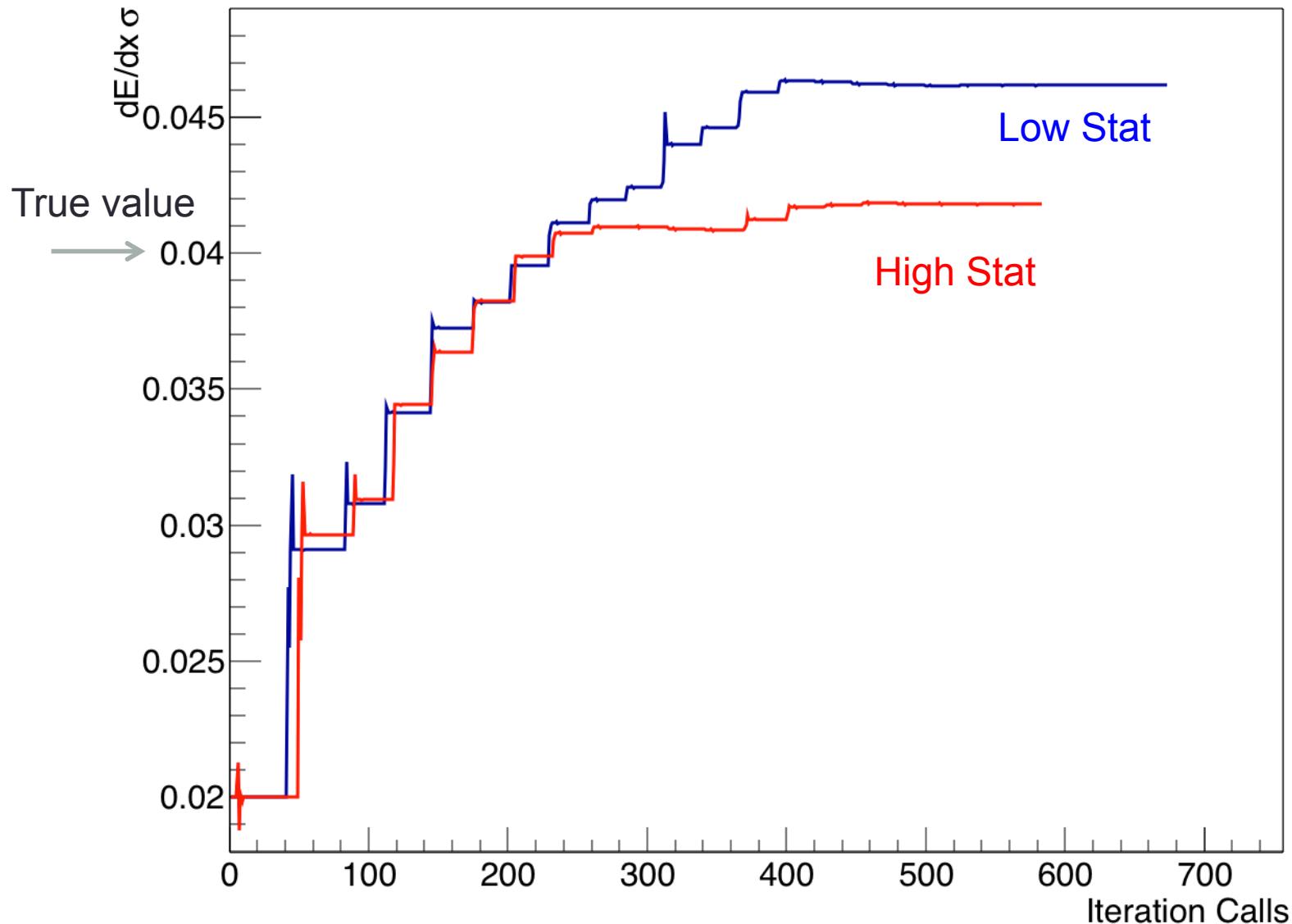


dE/dx vs ToF m^2 , $p_{\text{tot}}:[2.4,3.2]$ GeV/c $\theta:[0,20]$ mrad



Check Minimization History

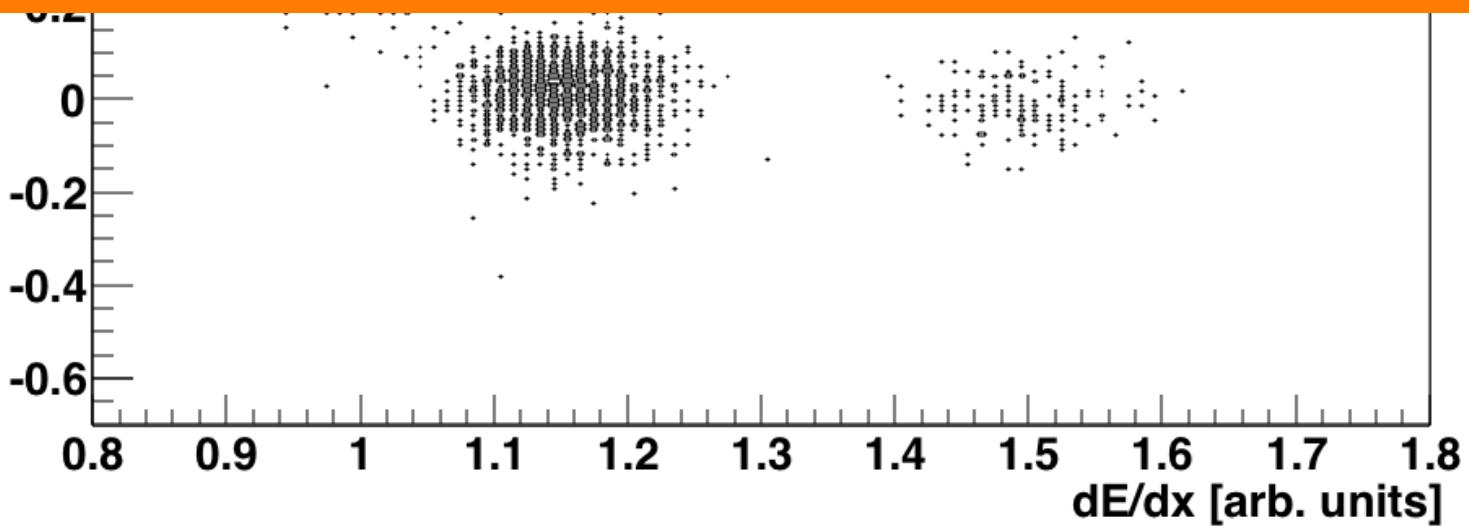
convergency



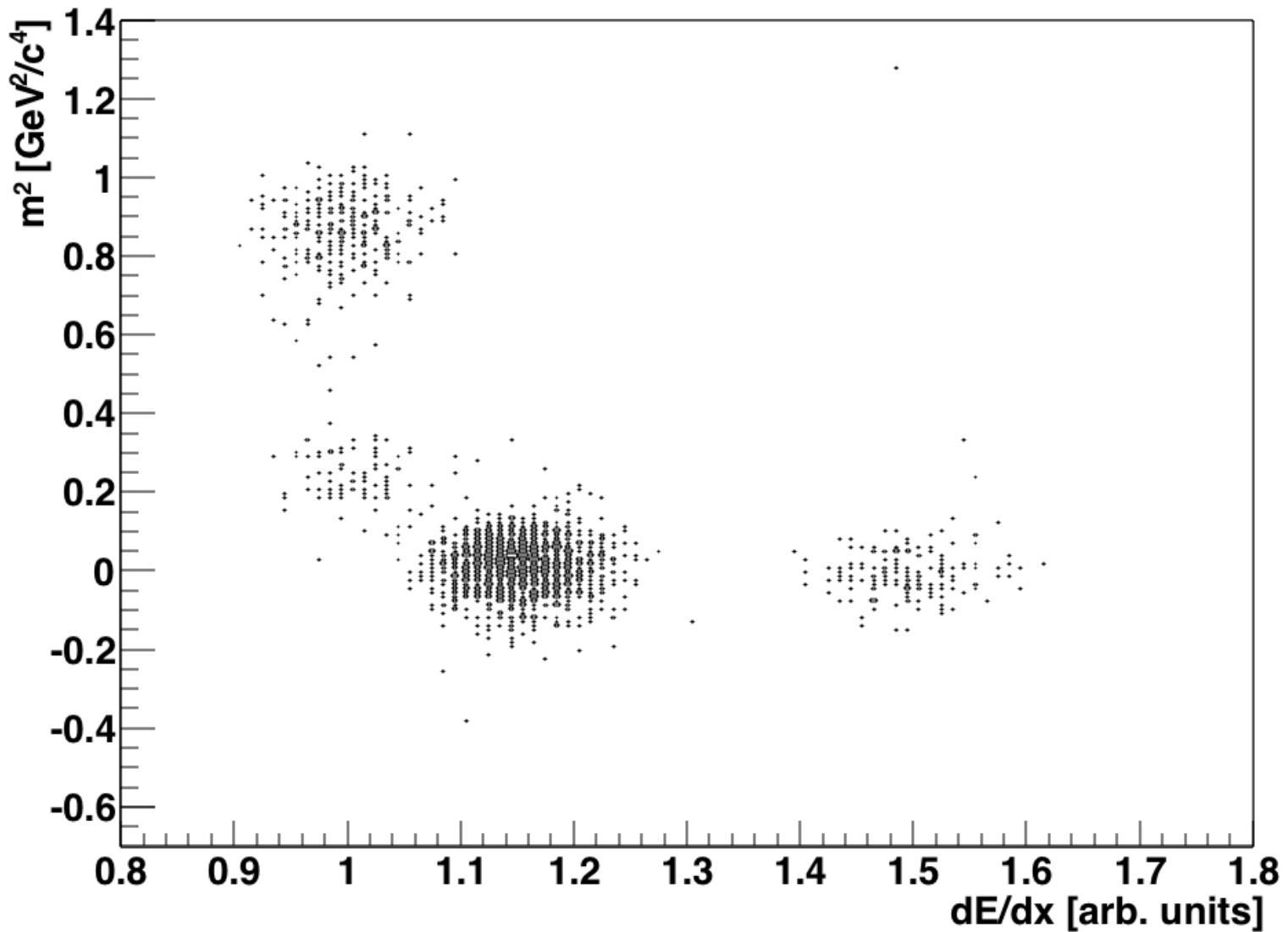
dE/dx vs ToF m^2 , $p_{\text{tot}}: [2.4, 3.2] \text{ GeV}/c$; $\theta: [0, 20] \text{ mrad}$



Fill PhaseSpaceMap/Container with real data



dE/dx vs ToF m^2 , $p_{\text{tot}}: [2.4, 3.2] \text{ GeV}/c$; $\theta: [0, 20] \text{ mrad}$



Adding another component to the model

Build a new component then add it to the existing model

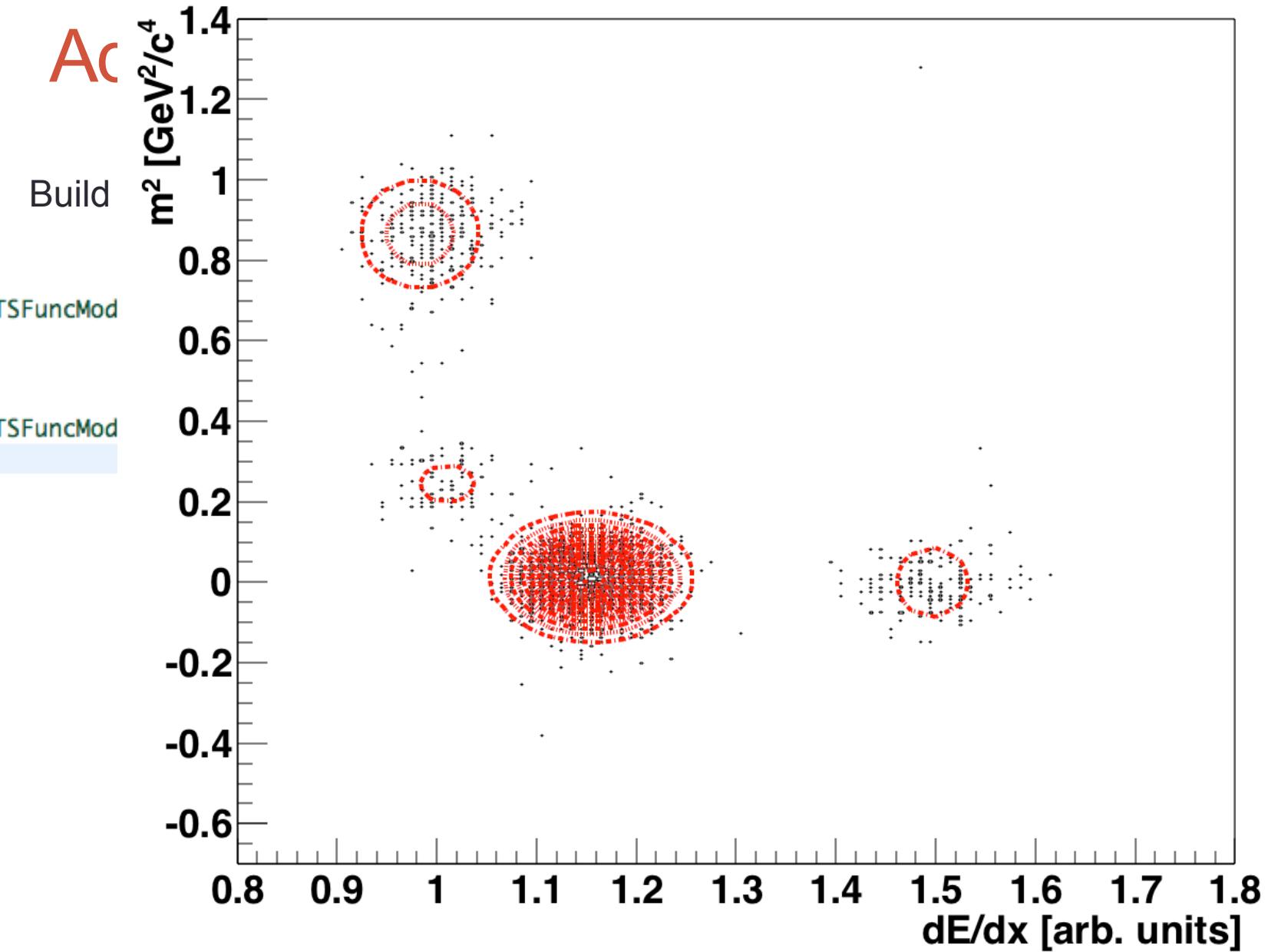
```
TSFuncModel2D Model_2D("Model_2D","model dEdx-m^{|Z|}",TSArgList(P_2D_Pdf,K_2D_Pdf,Pi_2D_Pdf)
,TSArgList(NP,NK,NPi));
```



```
TSFuncModel2D Model_2D("Model_2D","model dEdx-m^{|Z|}",TSArgList(P_2D_Pdf,K_2D_Pdf,Pi_2D_Pdf,E_2D_Pdf)
,TSArgList(NP,NK,NPi,NE));
```

Fit again....

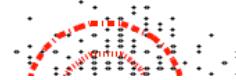
dE/dx vs ToF m^2 , $p_{\text{tot}}:[2.4,3.2]$ GeV/c $\theta:[0,20]$ mrad



dE/dx vs ToF m^2 , $p_{\text{tot}}:[2.4,3.2]$ GeV/c $\theta:[0,20]$ mrad

Ac

Build

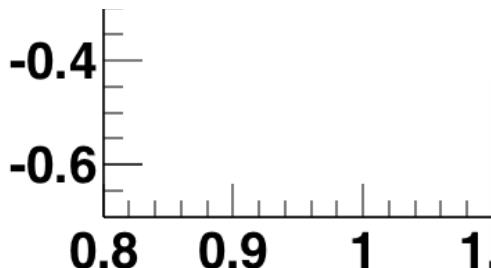
 $m^2 [\text{GeV}^2/\text{c}^4]$ 

```

dEdx_Pi_mean: #pi dE/dx mean m_{dE/dx}-#pi:[0.951389,1.35139] arb. units value: 1.15197 +/-0.000992109 asymm: 0 0
dEdx_mean_PtoPi: dE/dx mean m_{dE/dx}-P#pi:[-0.2,0.2] arb. units value: -0.15419 +/-0.002424217 asymm: 0 0
dEdx_sig_0: dE/dx width #sigma_{dE/dx}:[0.01,0.05] arb. units value: 0.0377323 +/-0.000634838 asymm: 0 0
dEdx_sig_alpha: #alpha scaling factor #alpha:[0,1] value: 0 +/-0 asymm: 0 0 (fixed)
m2_P_mean: P ToF m^{2} mean m^{2} value: 0.865437 +/-0.00543992 asymm: 0 0
m2_P_sig: P ToF m^{2} width #sigma_{m^{2}} value: 0.092536 +/-0.00391036 asymm: 0 0
dEdx_mean_KtoPi: dE/dx mean m_{dE/dx}-K#pi:[-0.2,0.2] arb. units value: -0.140684 +/-0.00416791 asymm: 0 0
m2_K_mean: K ToF m^{2} mean m^{2} value: 0.24564 +/-0.0073036 asymm: 0 0
m2_K_sig: K ToF m^{2} width #sigma_{m^{2}} value: 0.0660412 +/-0.0057192 asymm: 0 0
m2_Pi_mean: #pi ToF m^{2} mean m^{2} value: 0.0127932 +/-0.00170575 asymm: 0 0
m2_Pi_sig: #pi ToF m^{2} width #sigma_{m^{2}} value: 0.0650368 +/-0.00122455 asymm: 0 0
dEdx_E_mean: e dE/dx mean m_{dE/dx}-e value: 1.49633 +/-0.00312603 asymm: 0 0
dEdx_E_sig: e dE/dx width m_{dE/dx}-e value: 0.04 +/-0 asymm: 0 0 (fixed)
m2_E_mean: e ToF m^{2} mean m^{2} value: 0 +/-0 asymm: 0 0 (fixed)
m2_E_sig: e ToF m^{2} width #sigma_{m^{2}} value: 0.1 +/-0 asymm: 0 0 (fixed)
NP: P yield N_{P}:[1,2073] value: 290.98 +/-17.0596 asymm: 0 0
NK: K yield N_{K}:[0,2073] value: 93.118 +/-9.83578 asymm: 0 0
NPi: #pi yield N_{#pi}:[1,2073] value: 1468.89 +/-38.3604 asymm: 0 0
NE: e yield N_{e}:[0,2073] value: 164.013 +/-12.8061 asymm: 0 0

```

2017 entries



```

root [4]
root [4] TF1_Model_dEdx_MC_post->Integral(0.8,1.8)
(Double_t)2.01699866139916750e+01
root [5]
root [5]

```

Integral of the Best Fit Model

dE/dx [arb. units]

$dE/dx, p_{\text{tot}}:[2.4,3.2] \text{ GeV}/c$ $\theta:[0,20] \text{ mrad}$

Ac

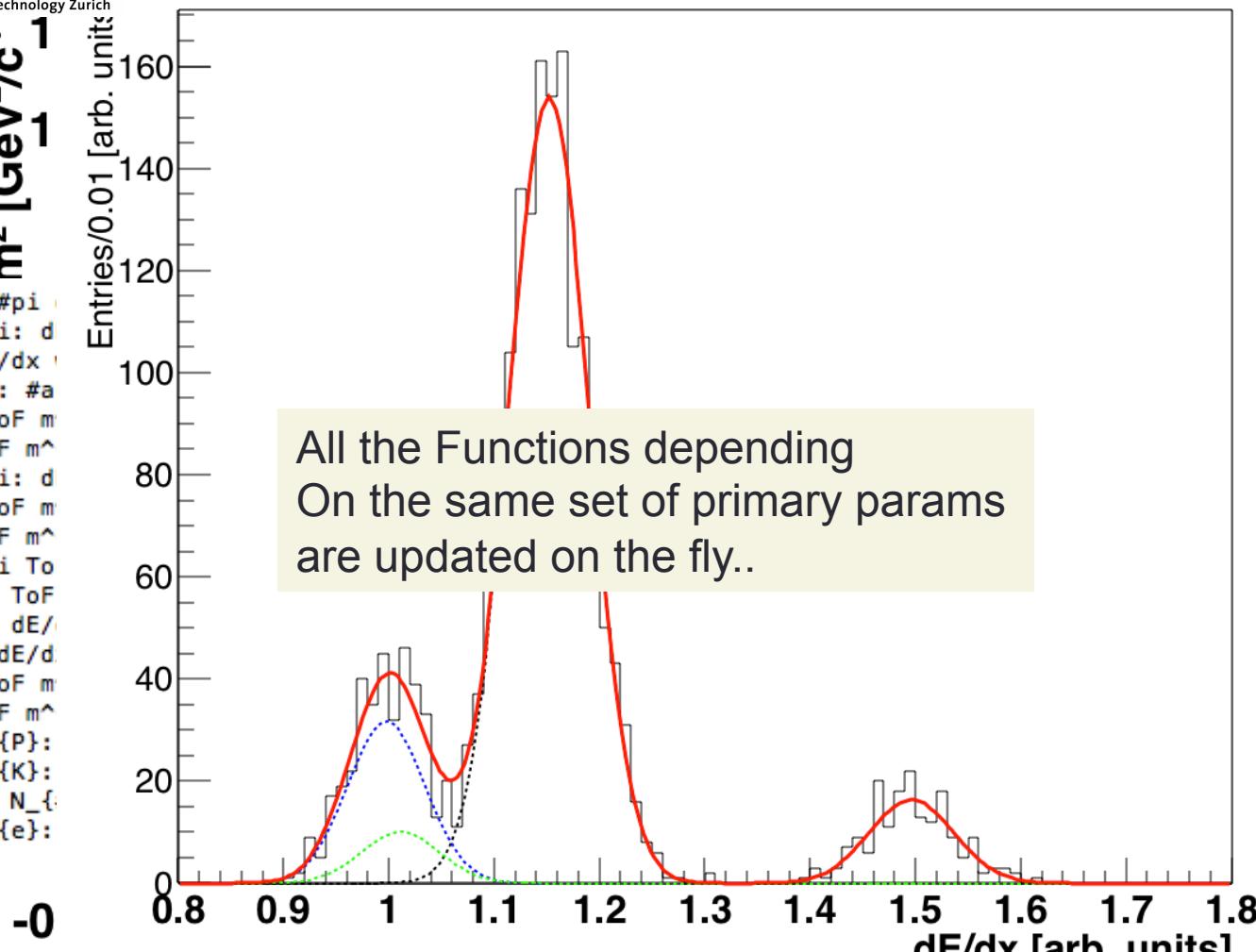
 $m^2 [\text{GeV}^2/c^4]$

Build

```

dEdx_Pi_mean: #pi
dEdx_mean_PtoPi: d
dEdx_sig_0: dE/dx
dEdx_sig_alpha: #a
m2_P_mean: P ToF m
m2_P_sig: P ToF m^
dEdx_mean_KtoPi: d
m2_K_mean: K ToF m
m2_K_sig: K ToF m^
m2_Pi_mean: #pi ToF
m2_Pi_sig: #pi ToF
dEdx_E_mean: e dE/dx
dEdx_E_sig: e dE/dx
m2_E_mean: e ToF m
m2_E_sig: e ToF m^
NP: P yield N_{P}:
NK: K yield N_{K}:
NPi: #pi yield N_{#pi}:
NE: e yield N_{e}:

```



-0.6

0.8 0.9 1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8

$dE/dx [\text{arb. units}]$

root [4] TF1_Model_dEdx_MC_post->Integral(0.8,1.8)
(Double_t)2.01699866139916750e+01

root [5]
root [5]

Integral of the Best Fit Model

Building a Generic Model

e.g.: V^0 analysis

```
TSVariable Yvar("Yvar","y","y","");
TSVariable Pt("Pt","p_{t}","p_{t}","GeV/c");
TSVariable InvMass("InvMass","inv. mass","V^{0} inv mass","GeV/c^{2}");
```

Variables
Y,Pt,InvMass

```
TSPhaseSpaceMap mapKinLam("mapKinY","map y");
mapKinLam.SetVariable(Yvar);
mapKinLam.SetBinning(Yvar,1,0,25,0,75);
```

space

Test Model:

Bkg: Exponential

Sig: Gauss

```
TSParameter Nbkg( Nbk , N_bkgs , N_bkg ),
```

```
TSFunc1DExpPdf Bkg_Pdf("Bkg_pdf","exp pdf bkg",InvMass,Exp_m,Exp_s);
TSFunc1DGaussPdf Sig_Pdf("Sig_pdf","gauss pdf sig",InvMass,Sig_m,Sig_s);
```

Pdf's

```
TSFuncModel1D Model_Bkg("Model_Bkg","model bkg",TSArgList(Bkg_Pdf),TSArgList(Nbkg));
TSFuncModel1D Model_Sig("Model_Sig","model sig",TSArgList(Sig_Pdf),TSArgList(Nsig));
```

Models

```
TSFuncModel1D Model_Sum("Model_Sum","model sum",TSArgList(Bkg_Pdf,Sig_Pdf),TSArgList(Nbkg,Nsig));
```

```
TSVariable Yvar("Yvar", "y", "y", ""); Variables  

TSVariable Pt("Pt", "p_{t}", "p_{t}", "GeV/c"); Y,Pt,InvMass  

TSVariable InvMass("InvMass", "inv. mass", "V^{0} inv mass", "GeV/c^{2}")
```

```
TSPhaseSpaceMap mapKinLam("mapKinY", "map y");  

mapKinLam.SetVariable(Yvar);  

mapKinLam.SetBinning(Yvar, 1, 0.25, 0.75);  

mapKinLam.SetVariable(Pt);  

mapKinLam.SetBinning(Pt, 1, 0.2, 0.4);
```

Variables
Y,Pt,InvMass

Phase Space

```
TSPParameter Exp_m("Exp_m", "Bkg_par_1", "", "");  

TSPParameter Exp_s("Exp_s", "Bkg_par_2", "", "");
```

```
TSPParameter Sig_m("Sig_m", "Signal_mean", "", "GeV/c^{2}");  

TSPParameter Sig_s("Sig_s", "Sigal_width", "", "GeV/c^{2}");
```

```
TSPParameter Nsig("Nsig", "N_{sig}", "N signal");  

TSPParameter Nbkg("Nbkg", "N_{bkg}", "N bkg");
```

```
TSFunc1DExpPdf Bkg_Pdf("Bkg_pdf", "exp pdf bkg", InvMass, Exp_m, Exp_s);  

TSFunc1DGaussPdf Sig_Pdf("Sig_pdf", "gauss pdf sig", InvMass, Sig_m, Sig_s);
```

Params

Pdf's

```
TSFuncModel1D Model_Bkg("Model_Bkg", "model bkg", TSArgList(Bkg_Pdf), TSArgList(Nbkg));  

TSFuncModel1D Model_Sig("Model_Sig", "model sig", TSArgList(Sig_Pdf), TSArgList(Nsig));
```

Models

```
TSFuncModel1D Model_Sum("Model_Sum", "model sum", TSArgList(Bkg_Pdf, Sig_Pdf), TSArgList(Nbkg, Nsig));
```

```
InvMass.SetRange(0.8,1.4);
```

```
TSDataBin binInvM("binInvM","binInvM","binInvM");
binInvM.SetUnbinned(true);
binInvM.SetVariable(InvMass,200,InvMass.GetMin(),InvMass.GetMax());
```

Data container

```
TSDataSetMgr dataSetMgrInvM("dataSetMgrInvM","V^{0}","data set");
dataSetMgrInvM.SetDataBin( binInvM );
dataSetMgrInvM.SetPhaseSpace(mapKinLam);
```

Data Set

```
Yvar.SetValue(0.3);      Exp_m.SetValue(0);
Pt.SetValue(0.3);        Exp_s.SetValue(1./0.5);
                        Sig_m.SetValue(1.11);
                        Sig_s.SetValue(0.01);
```

```
for(int i=0;i<10000;++i){
```

```
    double inv_mass=Bkg_Pdf.getRandom();
```

```
    InvMass.SetValue(inv_mass);
```

```
    dataSetMgrInvM.Fill();
```

```
}
```

```
for(int i=0;i<1000;++i){
```

```
    double inv_mass=Sig_Pdf.getRandom();
```

```
    InvMass.SetValue(inv_mass);
```

```
    dataSetMgrInvM.Fill();
```

```
}
```

Some dummy values

Generate/Fill 10k Bkg events

Generate/Fill 1k Signal events

```
TSLikelihoodCalculator invmLL("invmLL","invmLL");
invmLL.SetModel(Model_Sum);
invmLL.SetBinnedData(Model_Sum);

invmLL.SetBinnedData(binDataInvM, TSArgList(InvMass));
```

Minimization function

```
invmLL.GetModel()->Print();
```

```
TSMin.SetFunction(invmLL);
```

```
Nbkg.SetValue(1000);
Nsig.SetValue(100);
```

```
TSMin.Minimize();
```

Minimizer

Minimization..

```
dataSetMgrInvM.GetDataBin(1)->GetHistoFolder()->Write();
```

Get Histo

```
Model_Sum.BuildTF1()->Write();
```

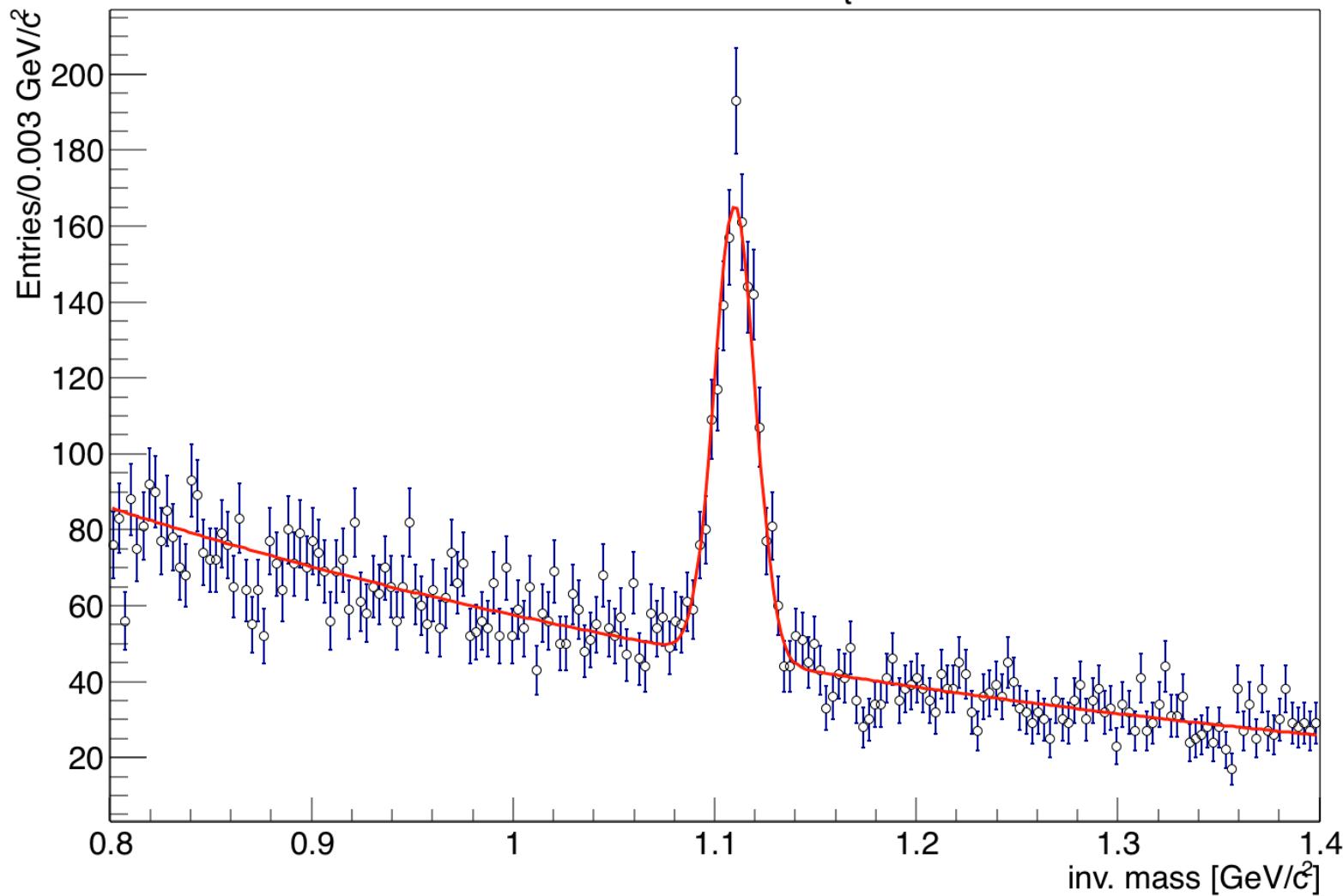
Get Model Func

Model will call its own parameters (now at the best fit values)
to calculate the current values

```
TSLikelihoodCalculator invmLL("invmLL","invmLL");
```

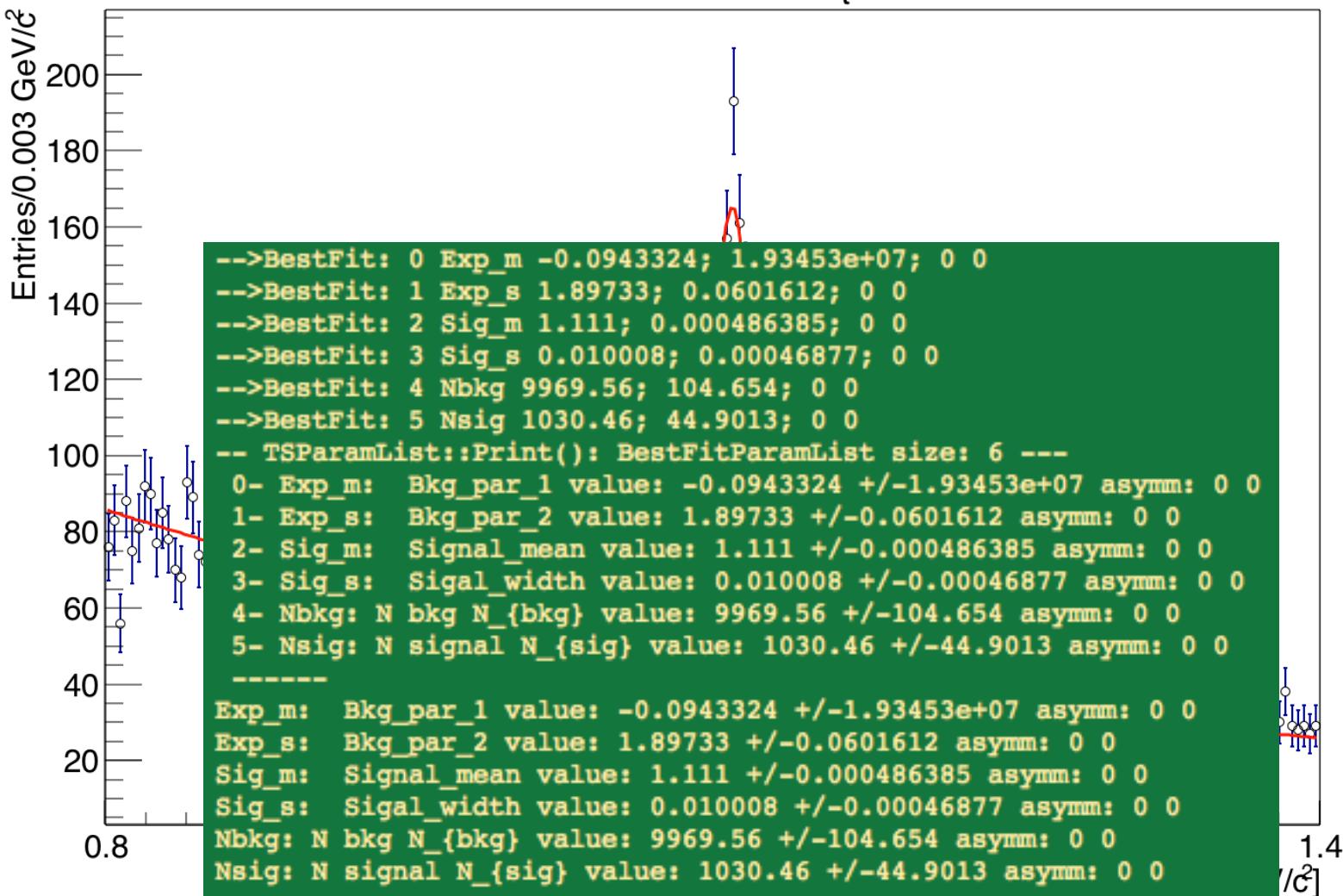
V^0 inv mass, $y:[0.25,0.75]$ $p_t:[0.2,0.4]$ GeV/c

nction



```
TSLikelihoodCalculator invmLL("invmLL","invmLL");
```

V^0 inv mass, $y:[0.25,0.75]$ $p_t:[0.2,0.4]$ GeV/c



nction

GETTING STARTED

← → C  <https://svnweb.cern.ch/trac/>



root / branches @ 11166

Name ▲

- ↑ .. /
- ▷ agrzeszc
- ▷ amarcine
- ▷ cleanup
- ▷ DAQ
- ▷ DCS
- ▷ dEdx
- ▷ owyszyns
- ▷ pro
- ▷ ptaszczel
- ▷ R3D
- ▷ rc1
- ▷ Sh
- ▷ ShineTools 
- ▷ Spalavsk
- ▷ Trigger
- ▷ VMsoft
- ▷ wr_ft

[View changes...](#)



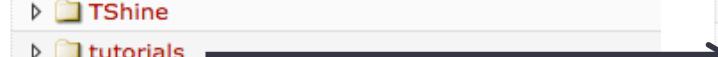
Powered by Trac 0.11.7
By Edgewall Software.

← → C  <https://svnweb.cern.ch/trac/na61soft>



root / branches / ShineTools @ 11166

Name ▲

- ↑ .. /
- ▷ Docs
- ▷ TShine
- ▷ tutorials 

[View changes...](#)



Powered by Trac 0.11.7
By Edgewall Software.

← → C  <https://svnweb.cern.ch/trac/na61s>



root / branches / ShineTools @ 11166

Name ▲

- ↑ .. /
- ▷ Docs
- ▷ TShine
- ▷ tutorials
 - ▷ Data
 - ▷ General
 - ▷ obj
 - .deps
 - bethe_bloch_table.txt
 - Makefile
 - myAna
 - myAna.cc
 - mymap.mom_theta.K.txt
 - mymap.theta_mom.txt
 - output.root
 - README.txt

[View changes...](#)



Powered by Trac 0.11.7
By Edgewall Software.

PROJECT NEXT

Implement results-wrap-up methods

Create skeleton specific for the LT case

**Add Tools for
MC correction/Unfolding**

...

...

Additional Material

CODE Organization

- Top Directory:
 - main: myAna.cc
 - Directories: src, include, ...
 - Makefile: compiles/links myAna.cc and all the source files (*.cc) in ./src
- ./src:
 - source files: .cc
 - Template class/functions implementations: .cct
 - (included at the end of the corresponding header files in ./include)
- ./include:
 - header files: .h

Installation

- Tested on:
 - Mac: OS X 10.9.4

Variables

TSVariable

```
CC".Kinematics_and_Observable_variables.")  
//
```

```
TSVariable Theta("Theta", "#theta", "polar angle", "mrad");  
TSVariable Mom("Mom", "p_{tot}", "momentum", "GeV/c");  
TSVariable ZTarg("ZTarg", "z_{targ}", "z-target", "cm");
```

```
TSVariable dEdx("dEdx", "dE/dx", "dE/dx", "arb. units");  
TSVariable ToF("ToF", "m^{2}", "ToF m^{2}", "GeV^{2}/c^{4}");  
TSVariable NPoints("NPoints", "Num. of Points", "Num. of Points");
```

```
Theta.SetRange(0.,400.);  
Mom.SetRange(0.1,31.);  
ZTarg.SetRange(0.,90.);
```

```
dEdx.SetRange(0.8,1.8);  
ToF.SetRange(-0.5,2);  
NPoints.SetRange(1.,300.);
```

```
TSArgList KinVarList(Theta,Mom,ZTarg);  
  
KinVarList.PrintVariables();
```



Has a Value and a Range

Range def: [min, max]
Check if values is inside:

```
bool is_in = Theta.IsInside(200.);
```

Kinematics and Observable variables
Theta: polar angle #theta:[0,400] mrad value: 0
Mom: momentum p_{tot}:[0.1,31] GeV/c value: 0
ZTarg: z-target z_{targ}:[0,90] cm value: 0

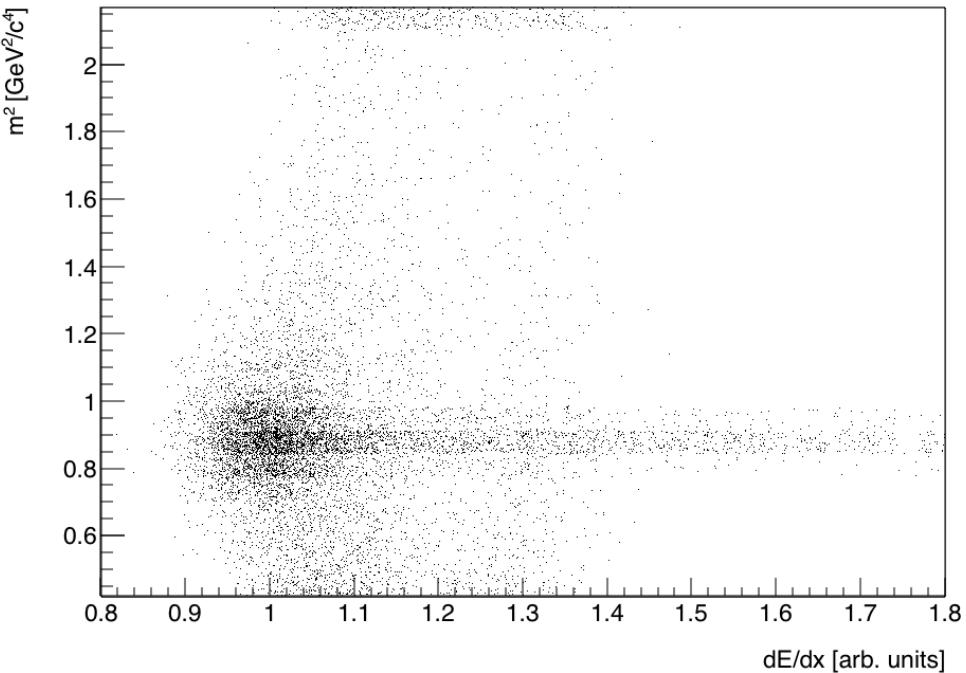
TSVariable

```
CC" Kinematics_and_Observable_variables ")  
//
```

```
TSVariable Theta("Theta", "#theta", "polar angle", "mrad");  
TSVariable Mom("Mom", "p_{tot}", "momentum", "GeV/c");  
TSVariable ZTarg("ZTarg", "z_{targ}", "z-target", "cm");  
  
TSVariable dEdx("dEdx", "dE/dx", "dE/dx", "arb. units");  
TSVariable ToF("ToF", "m^{2}", "ToF m^{2}", "GeV^{2}/c^{4}");
```



dE/dx vs ToF m^2



Generic Particle Class

TSParticle::

```
CC(" --- Particle List ----");

TSParticle P("p","p",TSParticle::MassP,1);
TSParticle N("n","n",TSParticle::MassN,0);
TSParticle PiPlu("PiPlu","#pi^{+}",TSParticle::MassPiPlus,1,TSParticle::TauPiPlus);
TSParticle PiMin("PiMin","#pi^{-}",TSParticle::MassPiMinus,-1,TSParticle::TauPiPlus);
TSParticle Pi0("Pi0","#pi^{0}",TSParticle::MassPi0,0,TSParticle::TauPi0);
TSParticle KPlu("KPlu","K^{+}",TSParticle::MassKPlus,1,TSParticle::TauKPlus);
TSParticle KMin("KMin","K^{-}",TSParticle::MassKMinus,-1,TSParticle::TauKMinus);
TSParticle K0("K0","K^{0}",TSParticle::MassK0,0,TSParticle::TauK0);
TSParticle Posit("Posit","e^{+}",TSParticle::MassPosit,1,TSParticle::TauEle);
TSParticle Ele("Ele","e^{-}",TSParticle::MassPosit,-1,TSParticle::TauEle);
TSParticle MuPlus("MuPlus","#mu^{+}",TSParticle::MassMuPlus,1,TSParticle::TauMuPlus);
TSParticle MuMinus("MuMinus","#mu^{-}",TSParticle::MassMuMinus,-1,TSParticle::TauMuMinus);
TSParticle X("X","X",0,0,0);

TString decay_form= TSParticle::GetReactionForm(TSArgList(PiPlu),"->",TSArgList(MuPlus,X));

cout<<decay_form<<endl;

P.Print();
KPlu.Print();
KMin.Print();
K0.Print();

    --- Particle List ---
#pi^{+}->#mu^{+}+X
p p q:+1 q>0 m=0.938272 GeV/c^2 m^2=0.880354 GeV^2/c^4
KPlu K^{+} q:+1 q>0 m=0.493667 GeV/c^2 m^2=0.243707 GeV^2/c^4
KMin K^{-} q:-1 q<0 m=0.493667 GeV/c^2 m^2=0.243707 GeV^2/c^4
K0 K^{0} q:0 q=0 m=0.497648 GeV/c^2 m^2=0.247654 GeV^2/c^4
```

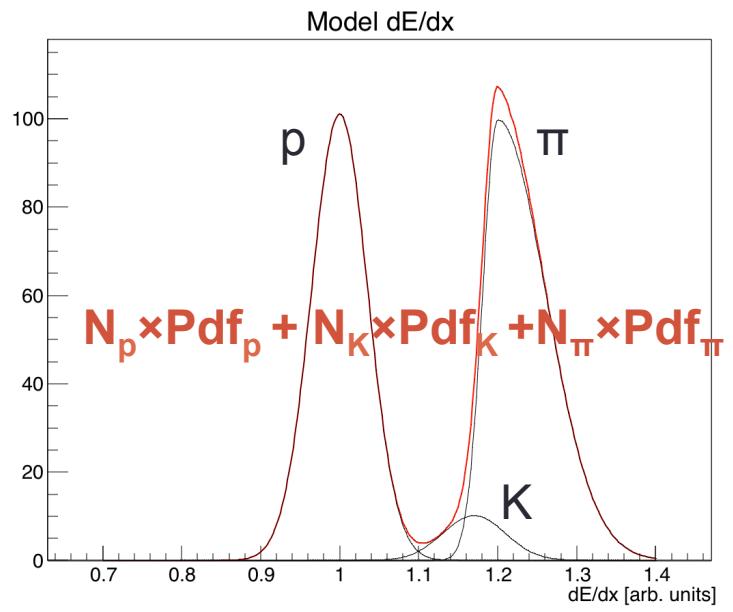
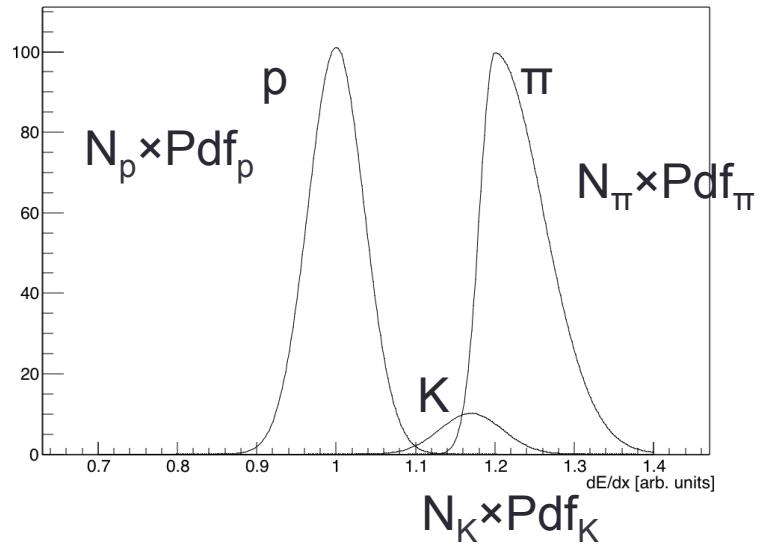
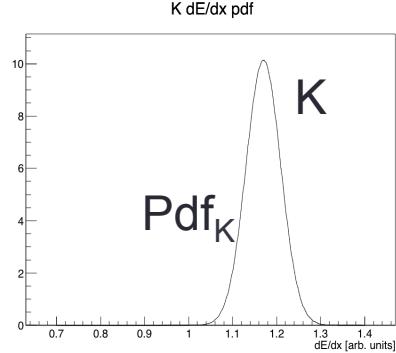
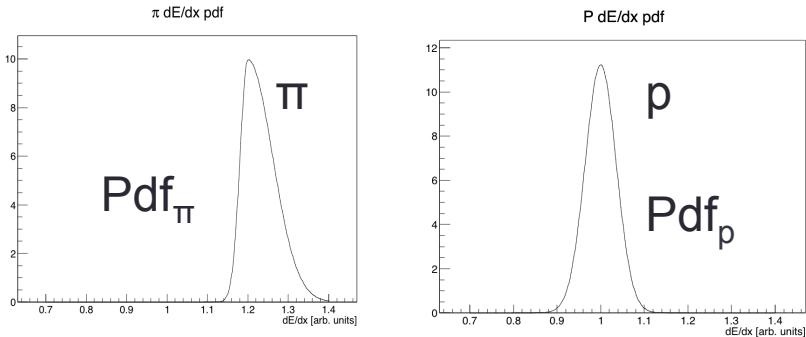


TSParticle::

```
CC("--- Particle List ----");

TSParticle P("p","p",TSParticle::MassP,1);
TSParticle N("n","n",TSParticle::MassN,0);
TSParticle PiPlu("PiPlu","#pi^{+}",TSParticle::MassPiPlus,1,TSParticle::TauPiPlus);
TSParticle PiMin("PiMin","#pi^{-}",TSParticle::MassPiMinus,-1,TSParticle::TauPiPlus);
TSParticle Pi0("Pi0","#pi^{0}",TSParticle::MassPi0,0,TSParticle::TauPi0);
TSParticle KPlu("KPlu","K^{+}",TSParticle::MassKPlus,1,TSParticle::TauKPlus);
TSParticle KMin("KMin","K^{-}",TSParticle::MassKMinus,-1,TSParticle::TauKMinus);
TSParticle K0("K0","K^{0}",TSParticle::MassK0,0,TSParticle::TauK0);
TSParticle Posit("Posit","e^{+}",TSParticle::MassPosit,1,TSParticle::TauEle);
TSParticle Ele("Ele","e^{-}",TSParticle::MassPosit,-1,TSParticle::TauEle);
TSParticle MuPlus("MuPlus","#mu^{+}",TSParticle::MassMuPlus,1,TSParticle::TauMuPlus);
TSParticle MuMinus("MuMinus","#mu^{-}",TSParticle::MassMuMinus,-1,TSParticle::TauMuMinus);
TSParticle X("X","X",0,0,0);
```

```
void SetCharge(int);
void SetMass(double);
void SetTau(double); //s
double GetBeta(double P) const;
double GetGamma(double P) const;
double GetBetaGamma(double) const;
double GetE(double P) const; //GeV
double GetTau() const; //s
double GetCTau() const; //m
double GetDecayTime(double P) const; //s
double GetDecayLength(double P) const; //P in GeV, Len in m
double GetToF(double P) const; //in s
double GenerateDecayLength(double P); //GeV, m
```



1- Parameters are objects

2- Functions are clients of their parameters.

```
TSFunc1DGaussPdf Pi_dEdx_Pdf("Pi_dEdx_Pdf", "#pi dE/dx pdf", dEdx_x, dEdx_Pi_mean, dEdx_Pi_sig);
```

- a) TSFunc1DGaussPdf P_dEdx_Pdf("P_dEdx_Pdf", "P dE/dx pdf", dEdx_x, dEdx_P_mean, dEdx_P_sig);

```
TSFunc1DGaussPdf K_dEdx_Pdf("K_dEdx_Pdf", "K dE/dx pdf", dEdx_x, dEdx_K_mean, dEdx_K_sig);
```

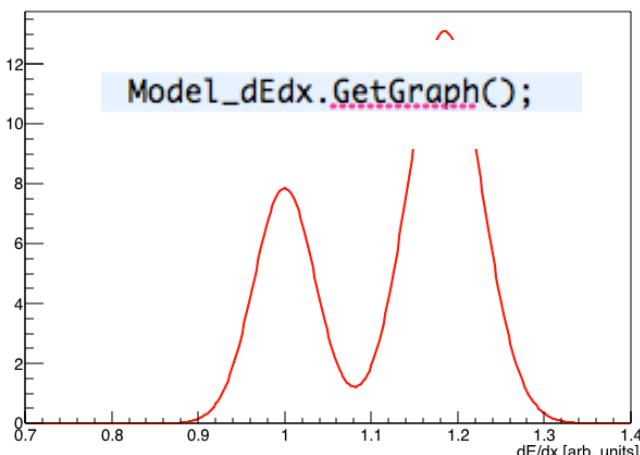
Ex. : I defined the object “**dEdx_sig_delta**”, all the functions clients of “delta” will be calculated with the (same) current value of the parameter.
Just create another delta param if each Pdf needs an independent delta param.

```
TSFunc1DAsymGaussPdf Pi_dEdx_Pdf("Pi_dEdx_Pdf", "#pi dE/dx pdf", dEdx_x, dEdx_Pi_mean, dEdx_Pi_sig, dEdx_sig_delta);
```

- b) TSFunc1DAsymGaussPdf P_dEdx_Pdf("P_dEdx_Pdf", "P dE/dx pdf", dEdx_x, dEdx_P_mean, dEdx_P_sig, dEdx_sig_delta);
TSFunc1DAsymGaussPdf K_dEdx_Pdf("K_dEdx_Pdf", "K dE/dx pdf", dEdx_x, dEdx_K_mean, dEdx_K_sig, dEdx_sig_delta);

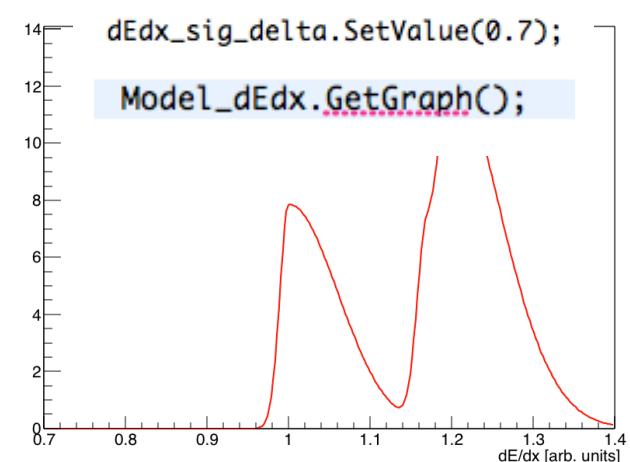
a)

model dE/dx



b)

```
dEdx_sig_delta.SetValue(0.7);
Model_dEdx.GetGraph();
```



Model is a client of
Pi_dEdx_Pdf
P_dEdx_Pdf,
K_dEdx_Pdf