TOOLS FOR INTEGRATED NA61/SHINE ANALYSIS

Silvestro di Luise
ETHZ
NA61/Shine Collaboration Meeting
CERN, Sep. 2014

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

PROJECT DESCRIPTION OF THE MAIN LIBRARIES

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

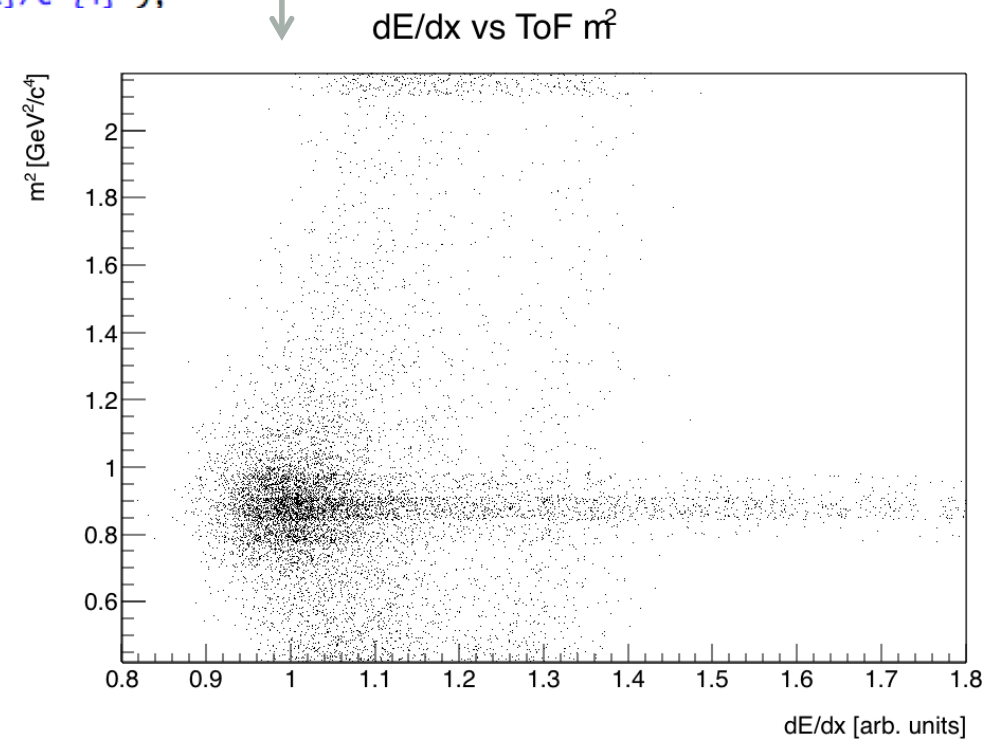
Range def: `[min,max[`
Check if values is inside:

```
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

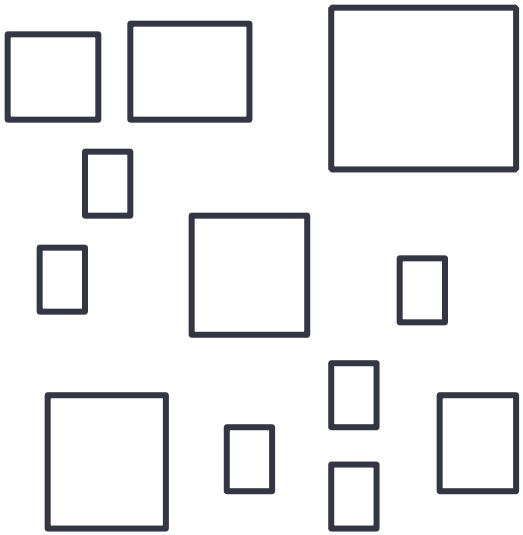
```
CC" Kinematics and Observable variables ")
//


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}");
```



Phase Space
Map & Folders

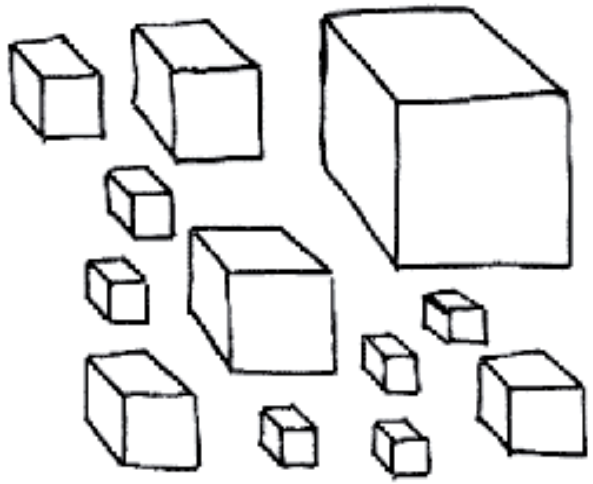
TSPhaseSpaceMap

You can multiply two maps
….of the any dimensions

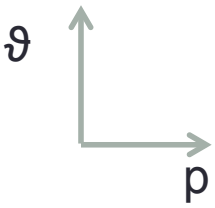


X

=

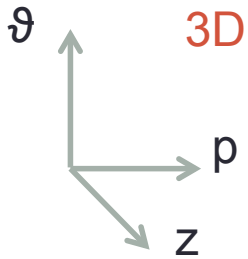


2D



1D

3D



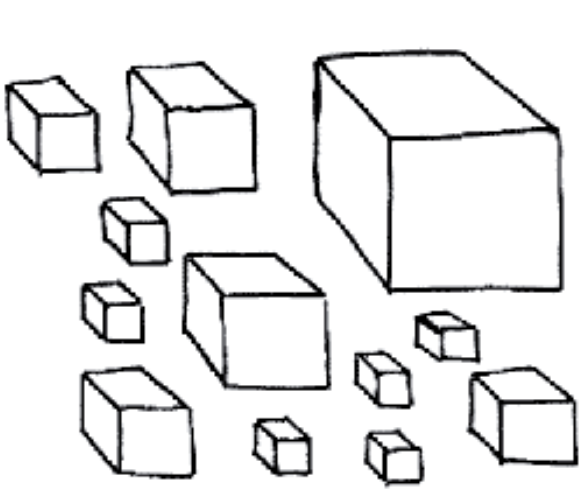
12 volumes

1 volume

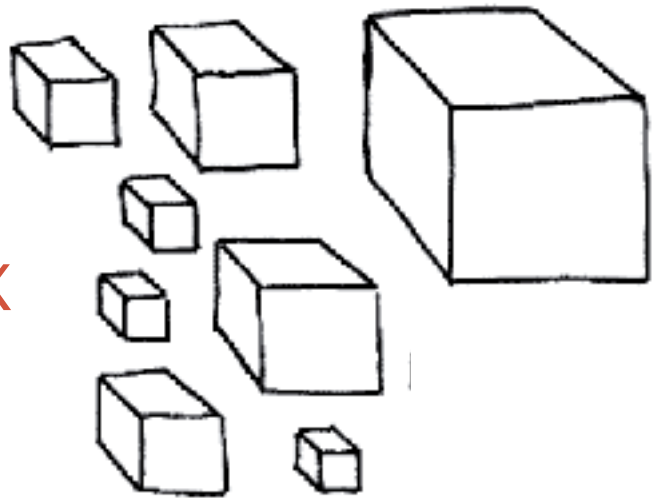
12 volumes

TSPhaseSpaceMap

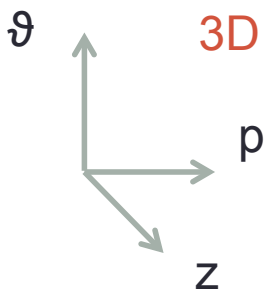
You can multiply two maps
….of the any dimensions



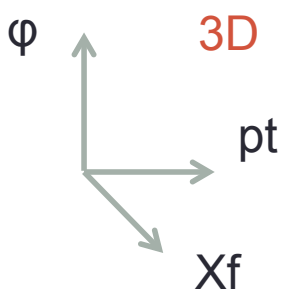
X



=



X



=

9D hyp-vols

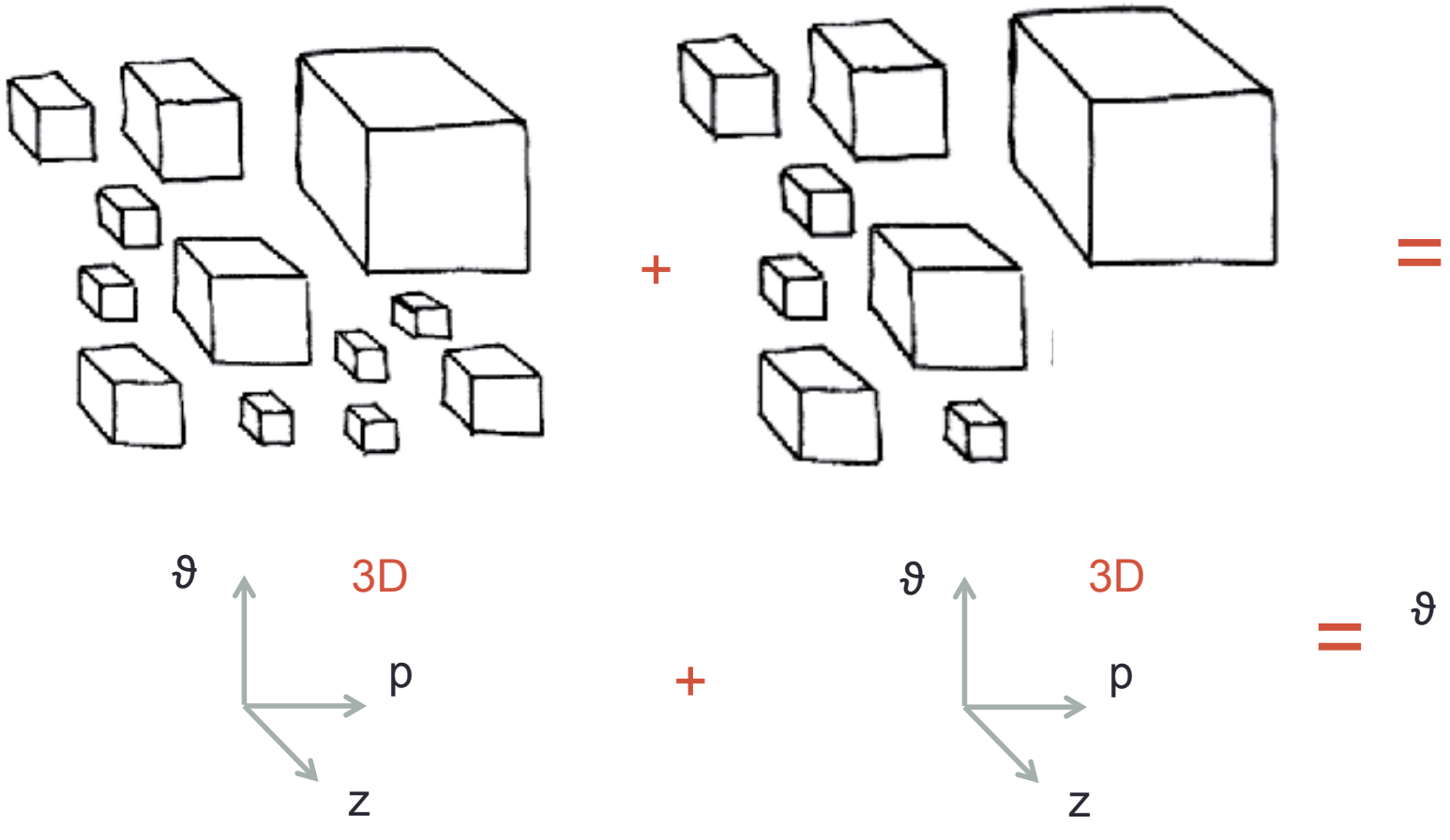
12 volumes

8 volumes

96 volumes

TSPhaseSpaceMap

You can Add two maps
….of the same dimension



12 volumes

8 volumes

20 volumes

TSPhaseSpaceMap

```
C(" --- 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());
```

TSPhaseSpaceMap

```
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.
```

```
C(" --- 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
TSPhaseSpaceFld mapKin3D_Pos("map_PosTracks","q>0",mapKin3D.Title(),mapKin3D);
TSPhaseSpaceFld mapKin3D_Neg("map_NegTracks","q<0",mapKin3D.Title(),mapKin3D);
```

TSPhaseSpaceMap

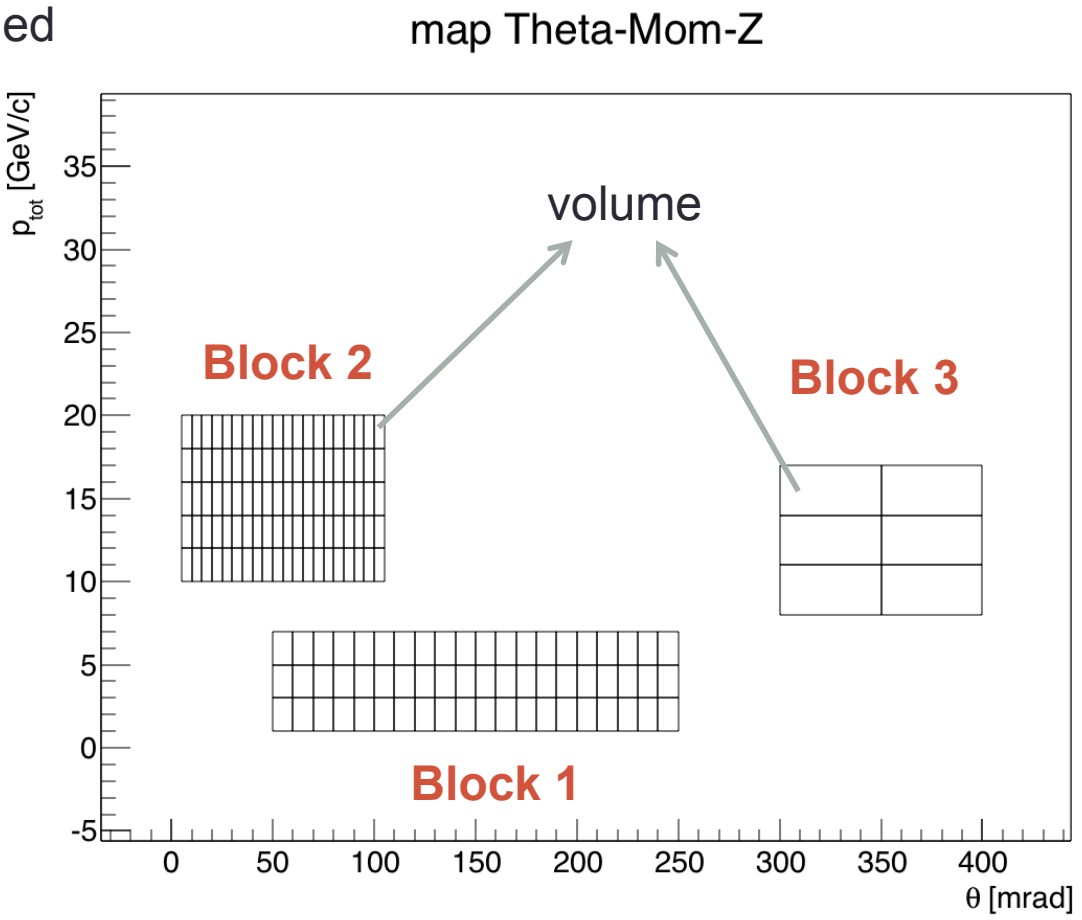
In a block maps (rows) are multiplied
Different blocks are Added

```
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

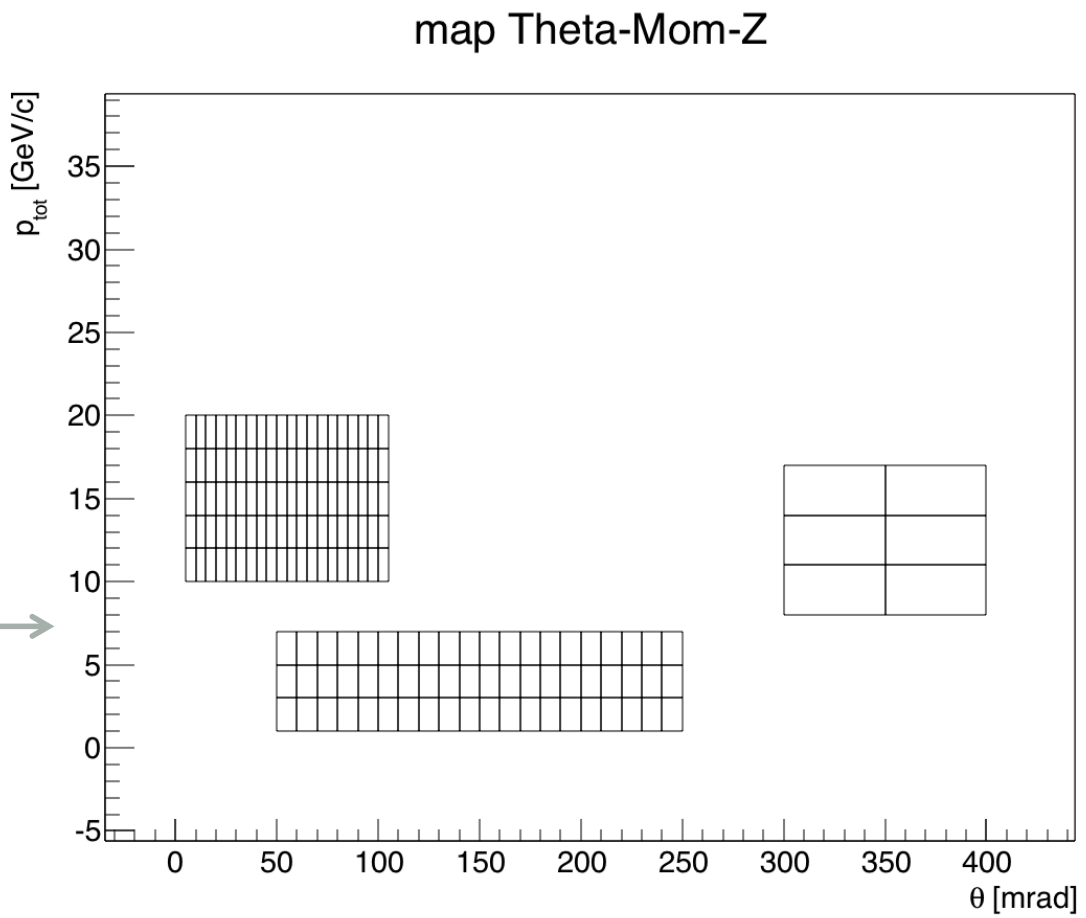
Block 2

Block 3



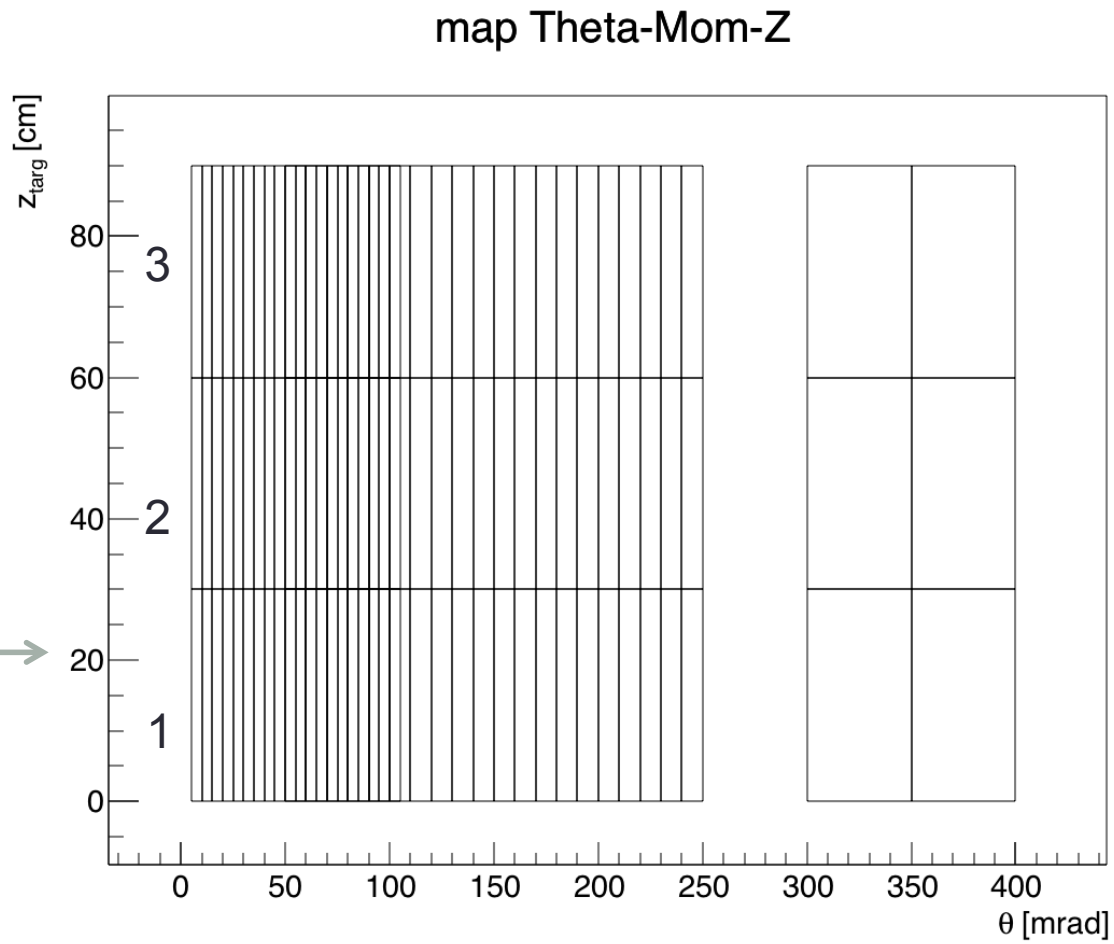
TSPhaseSpaceMap

```
C(" --- 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.GetM

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
```



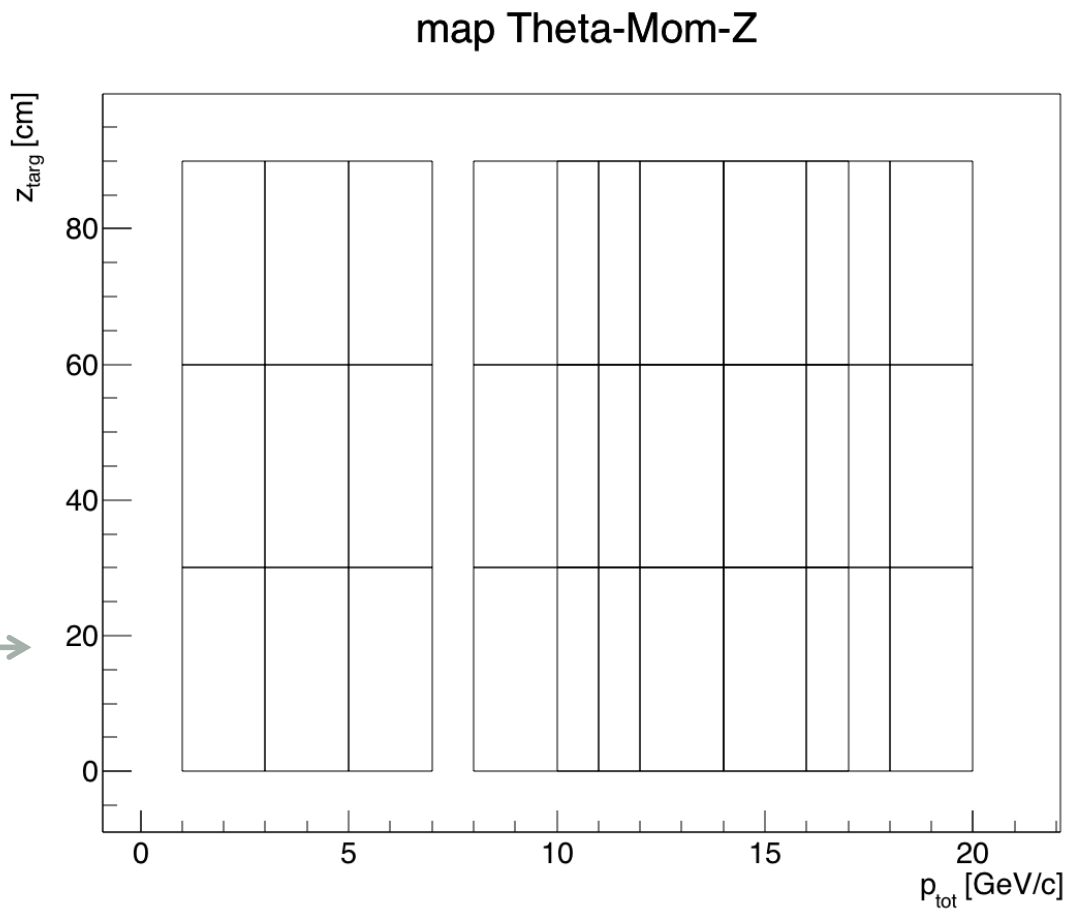
TSPhaseSpaceMap

```
C(" --- Phase Space ---");

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

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

mapKin2D.SetBinning("Theta:Mom","mymap.theta_mom.tx

TSPhaseSpaceMap mapKin3D("mapKin3D","map Theta-Mom-

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

mapKin3D.SetBinning("Theta:Mom","mymap.theta_mom.tx

mapKin3D.SetVariable(ZTarg);

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

mapKin3D.Print();

mapKin3D.GetHistogramFld()->Write();

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



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 positiv
TSPhaseSpaceFld mapKin3D_Pos("map_PosTracks","q>0",mapK
TSPhaseSpaceFld mapKin3D_Neg("map_NegTracks","q<0",mapK
```



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 n
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.SetV
mapKin2D.SetBi

TSPhaseSpaceMa

mapKin3D.SetV
mapKin3D.SetV
mapKin3D.SetBi
mapKin3D.SetV

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 n
TSPhaseSpaceFld mapKin3D_Pos("map_PosTracks","q>0",mapKin3D.Title(
TSPhaseSpaceFld mapKin3D_Neg("map_NegTracks","q<0",mapKin3D.Title(
```

TSPhaseSpaceMap holds the defintion of the PhaseSpace Volumes

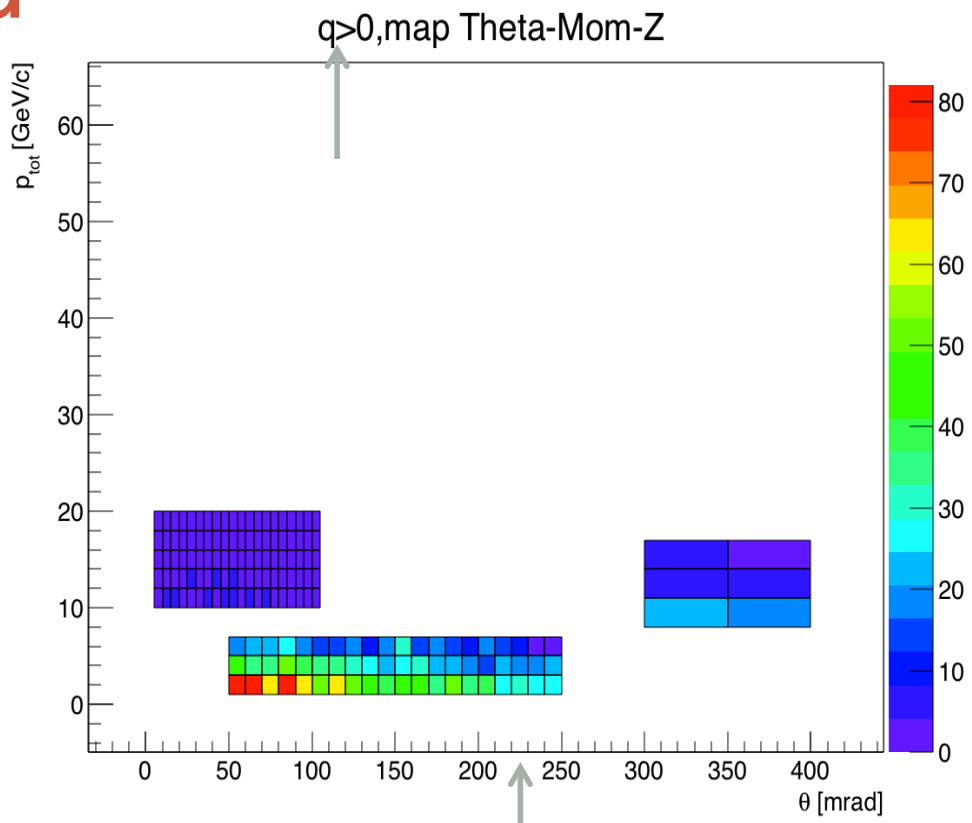
Different Data Set can share the same PhaseSpaceMap
e.g.: RST, WST, q>0, q<0 ….
A TSPhaseSpace Folder is the container to be used to control the occupancy of the Map….

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

0,30] cm
0,30] cm
0,30] cm
0,30] cm
[0,30]

:[0,30] cm
}:[0,30] cm
}:[0,30] cm
}:[0,30] cm
}:[0,30] cm
}:[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
```

TSPhaseSpaceFld

A Map Folder can be filled …example later

```
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();
```

TSPhaseSpaceFld

A Map Folder can be filled …example later

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

```
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);
```

```
}:[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
t}:[1,3] GeV/c z_{targ}:[0,30] cm, entries: 26
ot}:[1,3] GeV/c z_{targ}:[0,30] cm, entries: 26
ot}:[1,3] GeV/c z_{targ}:[0,30] cm, entries: 26
ot}:[1,3] GeV/c z_{targ}:[0,30] cm, entries: 25
ot}:[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
t}:[3,5] GeV/c z_{targ}:[0,30] cm, entries: 16
t}:[5,7] GeV/c z_{targ}:[0,30] cm, entries: 6
t}:[3,5] GeV/c z_{targ}:[0,30] cm, entries: 15
t}:[5,7] GeV/c z_{targ}:[0,30] cm, entries: 12
t}:[3,5] GeV/c z_{targ}:[0,30] cm, entries: 15
t}:[5,7] GeV/c z_{targ}:[0,30] cm, entries: 9
t}:[3,5] GeV/c z_{targ}:[0,30] cm, entries: 16
t}:[5,7] GeV/c z_{targ}:[0,30] cm, entries: 14
ot}:[3,5] GeV/c z_{targ}:[0,30] cm, entries: 17
ot}:[5,7] GeV/c z_{targ}:[0,30] cm, entries: 8
```

…..fill the MapFolder ……

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

mapKin3D_Pos.Print();
```

Parameters

TSParameter::

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

```
C("---- 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}");

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

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

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

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

TSParamFunction::

```
TSParamFunction 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: $m_K = m_\pi - m_{K\pi}$

```
TSParamFunction 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…..

TSParamFunction::

```
TSParamFunction 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: $m_K = m_\pi - m_{K\pi}$

```
TSParamFunction 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();
```

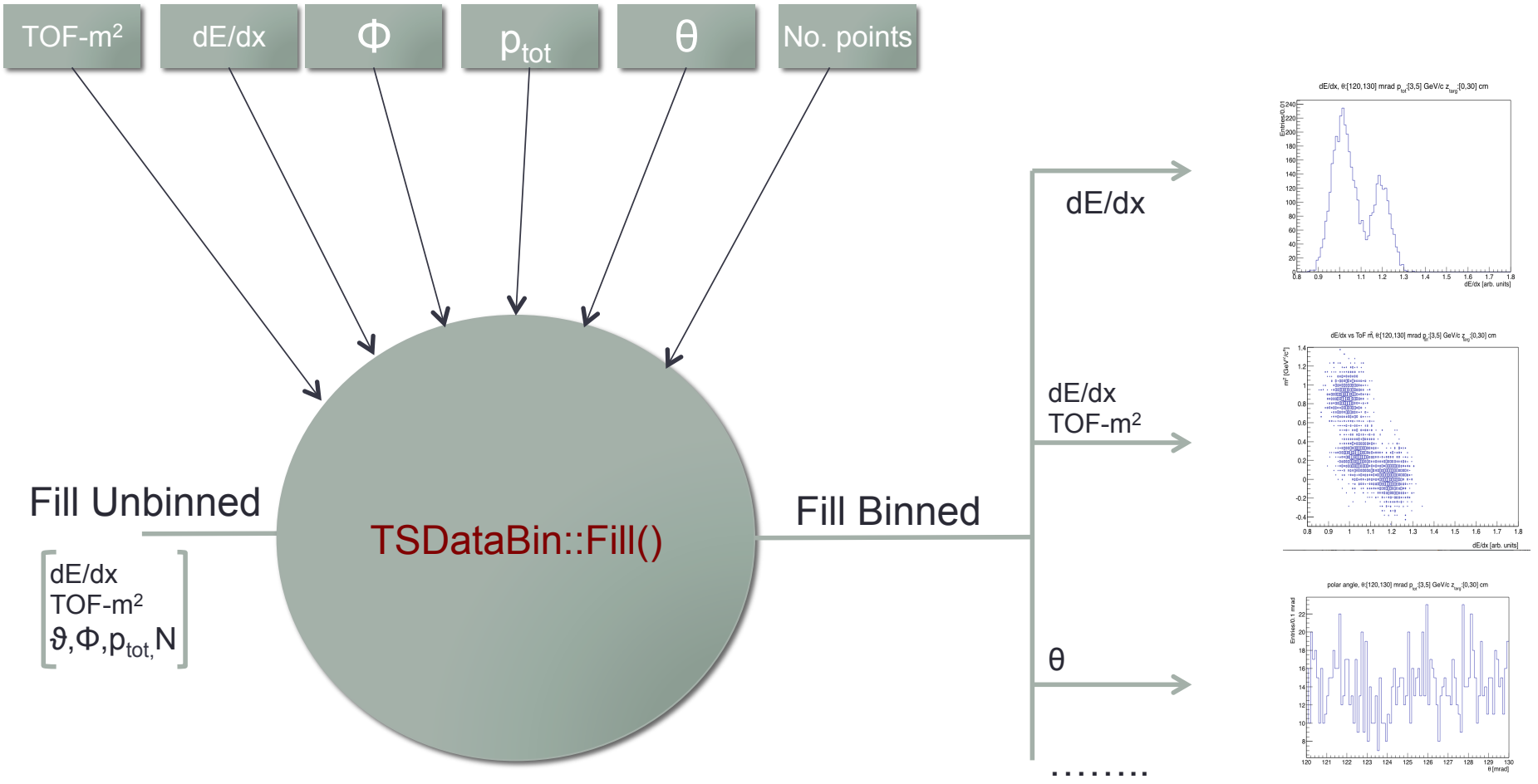
```
--- Print dEdx_K_sig ---
dEdx_sig_0*TMath::Power(dEdx_K_mean/dEdx_Pi_mean,dEdx_sig_alpha)
[0]*TMath::Power([2]/[1],[3])+x
 value: 0.037662
0 dEdx_sig_0 = 0.04 +/- 0
1 dEdx_Pi_mean = 1.13 +/- 0
2 dEdx_K_mean = 1.03 +/- 0
3 dEdx_sig_alpha = 0.65 +/- 0

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

Filling DataBin

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

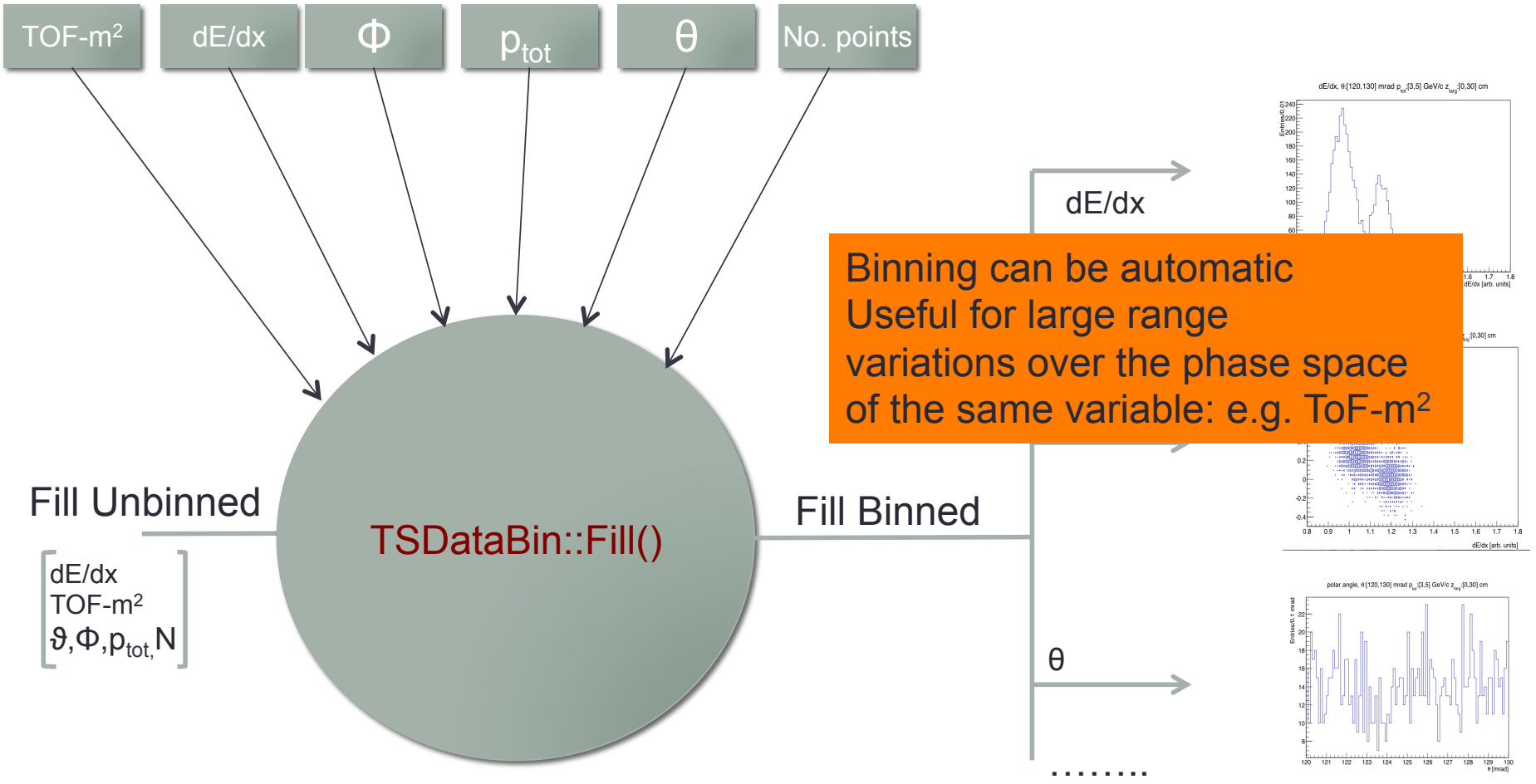


Unbinned (root Tree) filled if selected

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

Filling DataBin

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

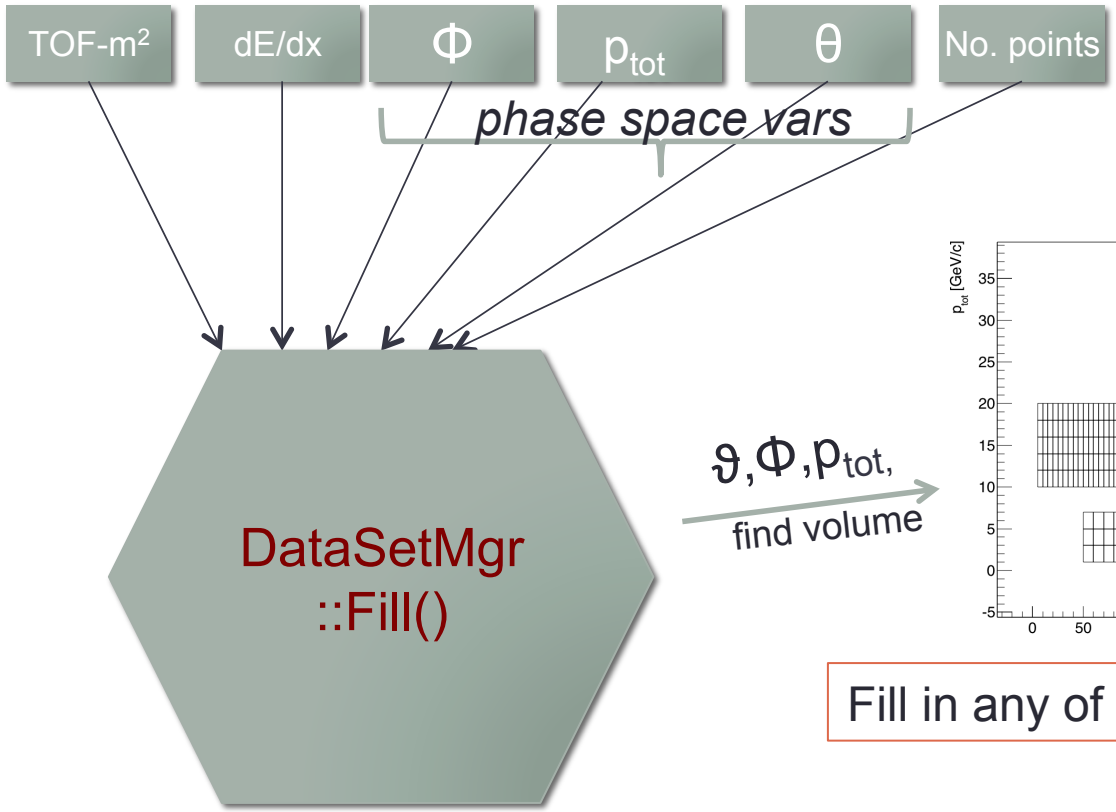


Unbinned (root Tree) filled if selected

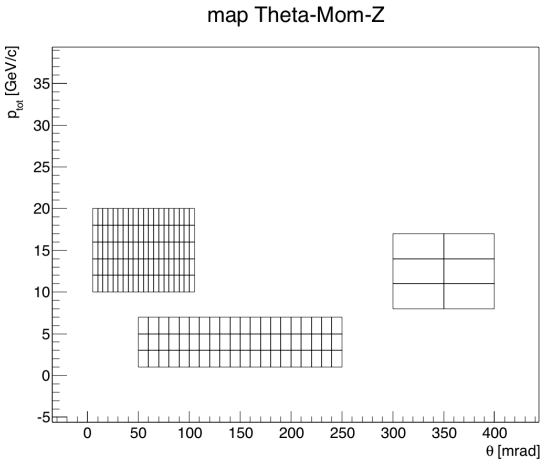
All the histograms in the container will be filled with the variable associated to the x/y axis

Filling DataSetMgr

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



ϑ, Φ, p_{tot},
find volume



Fill in any of the PhaseSpaceMap volumes?

Yes: Fill DataBin for that volume

Fill Unbinned

dE/dx
TOF-m²
ϑ, Φ, p_{tot}, N

TSDataBin
::Fill()

Fill Binned

Dedicated Objects:
BetheBloch
Functions & Managers

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


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

TF1 FBB(
FBB.Prir
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 beetwen 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.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-m^2 MC generator

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

tofGen.SetParticle("proton","p",TSParticle::MassP);
tofGen.SetParticle("kaon","K",TSParticle::MassKPlus);
```

Generic Class to generate m^2 distribution

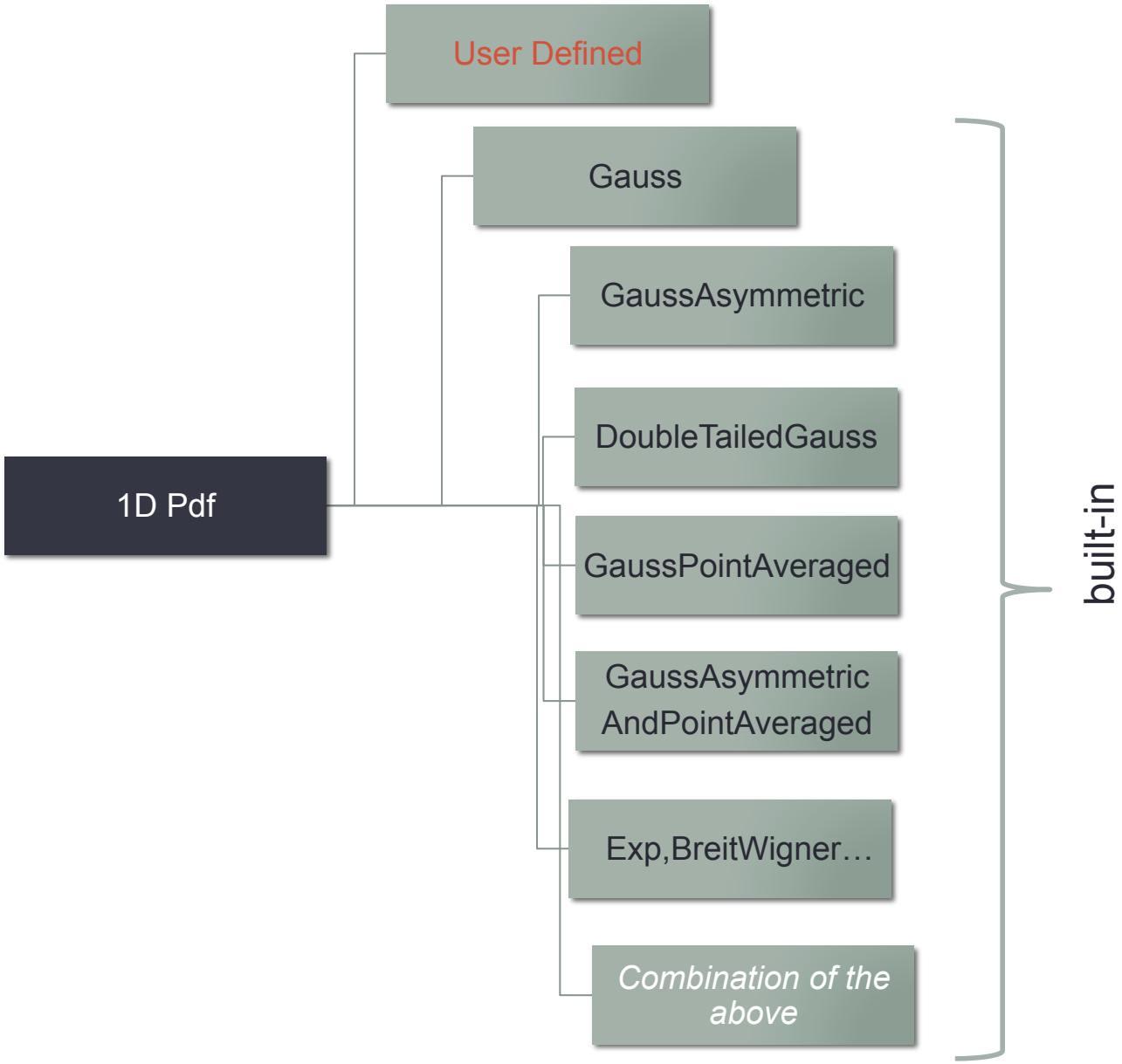
1-Based on Parametrization of Mean and Resolution as a function of the Momentum

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

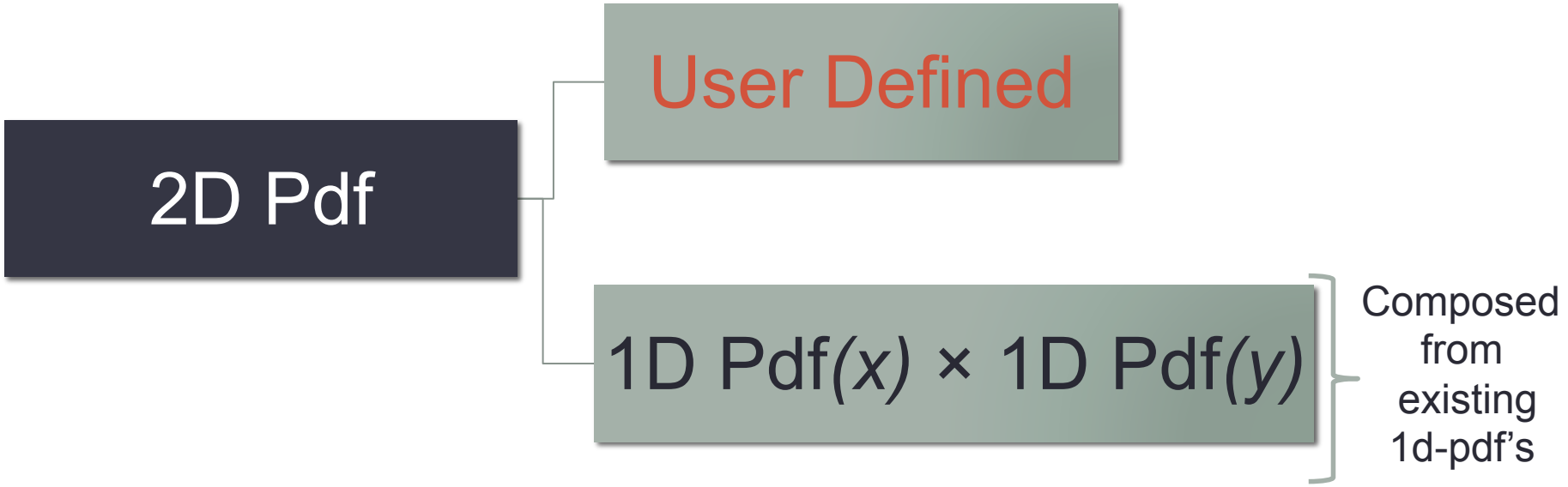
(features to add bias smearings etc..)

Distribution Functions

Distribution Functions



Distribution Functions



Distribution Functions

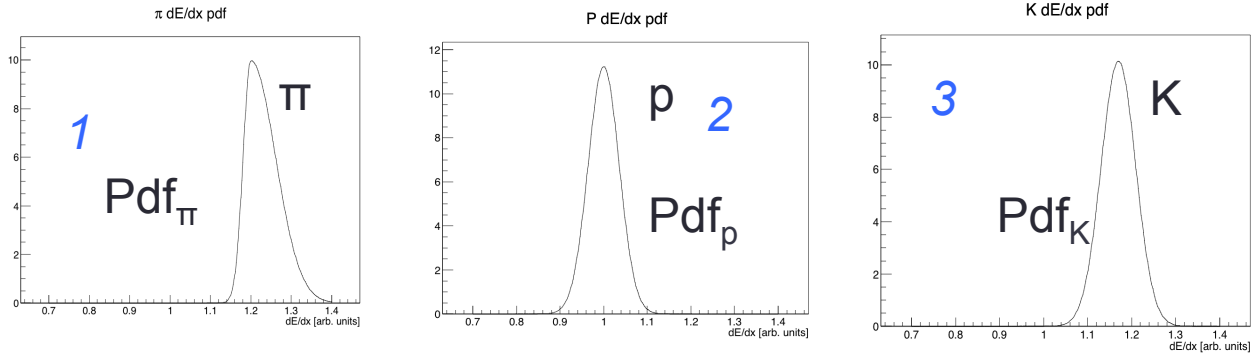
Model (1D or 2D)

$$N_1 \times Pdf_1 + N_2 \times Pdf_2 + N_3 \times Pdf_3$$

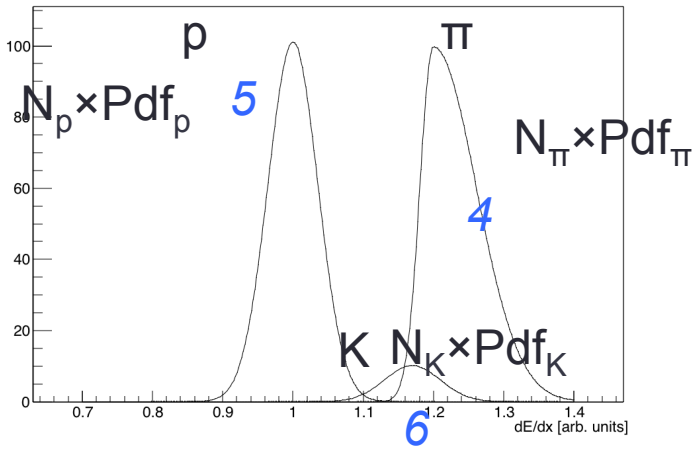
Most General: A function of Pdf's and Coefficients

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
```



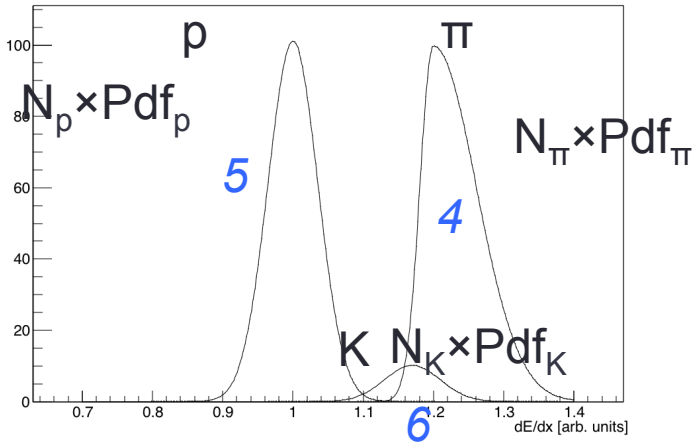

```
K_dEdx_Pdf.BuildGraph();

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

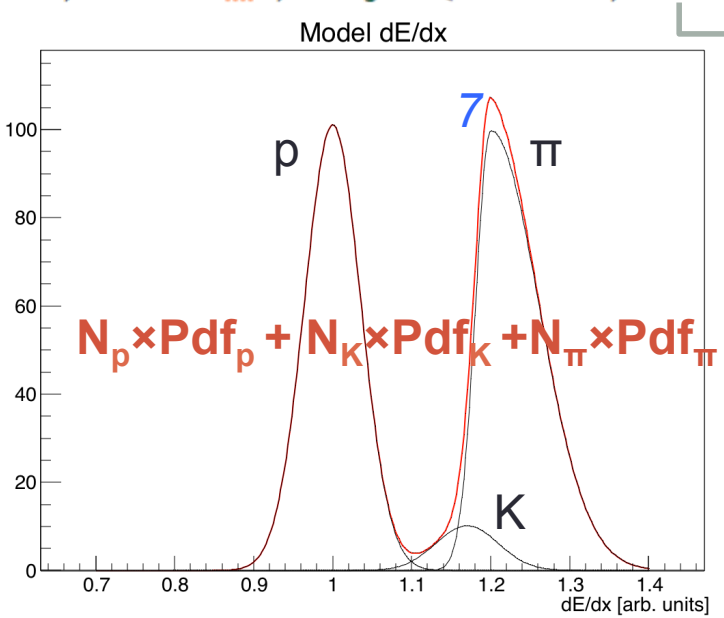
4 5 6




```
//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));
```

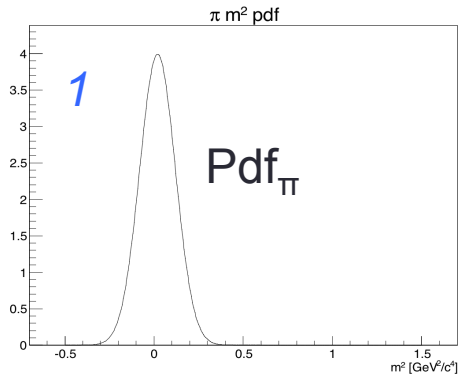
7

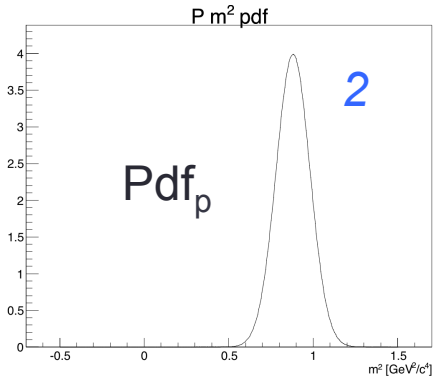
```
Model_dEdx.GetGraph();
```

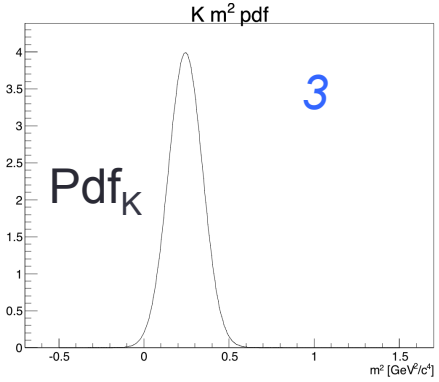


dE/dx

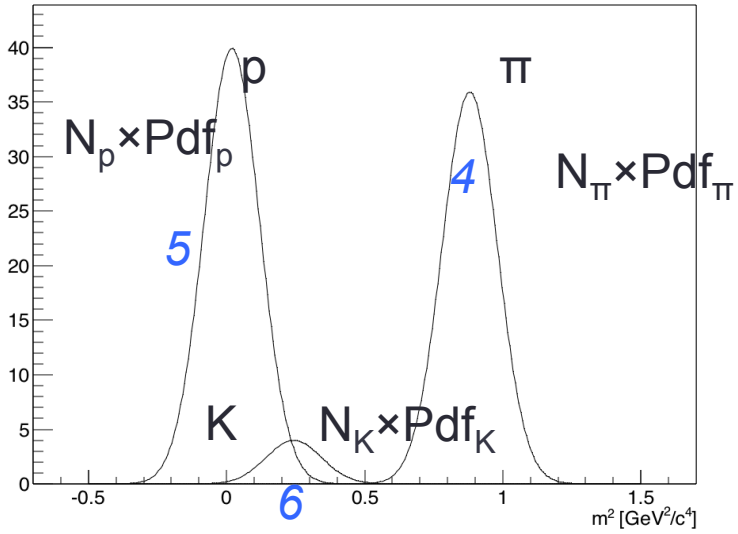
```
//Build 1D Mass2 Single Particles Pdf
TSFunc1DGaussPdf Pi_m2_Pdf("Pi_m2_Pdf","#pi m^{2} pdf",m2_x,m2_Pi_mean,m2_Pi_sig); 1
TSFunc1DGaussPdf P_m2_Pdf("P_m2_Pdf","P m^{2} pdf",m2_x,m2_P_mean,m2_P_sig); 2
TSFunc1DGaussPdf K_m2_Pdf("K_m2_Pdf","K m^{2} pdf",m2_x,m2_K_mean,m2_K_sig); 3

//Build 1D dEdx Model  Pdf
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
```

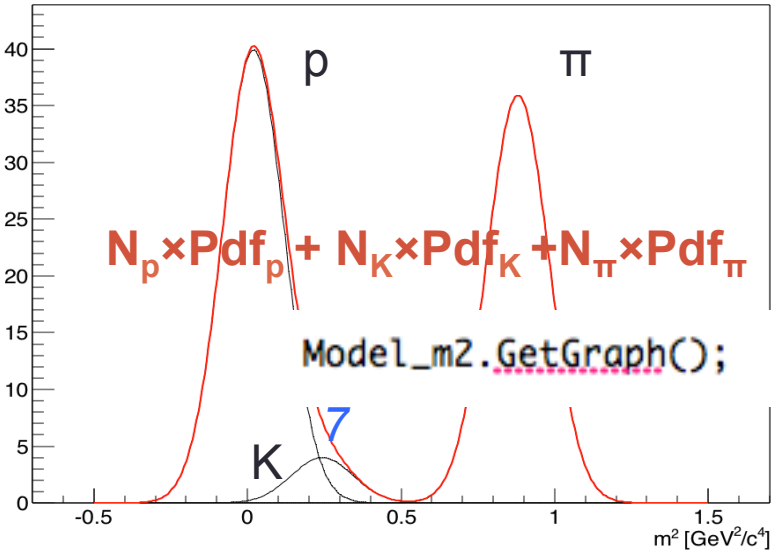






ToF

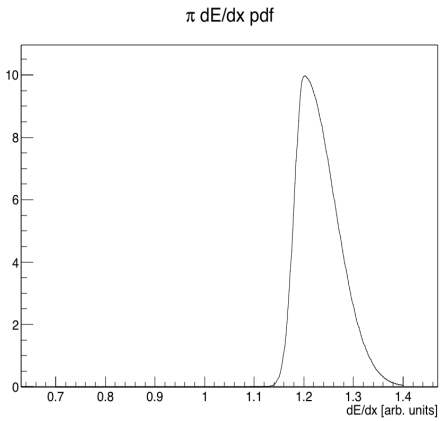




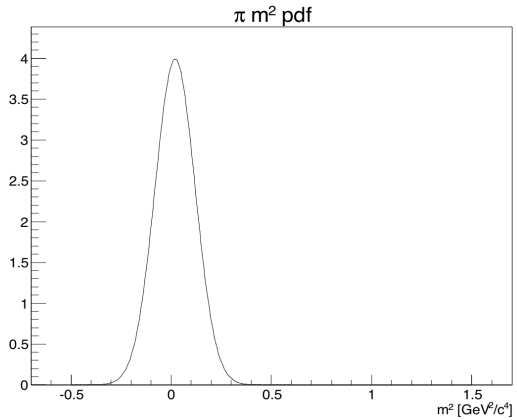
```
// Buil 2D Single Particle Pdf
```

↓

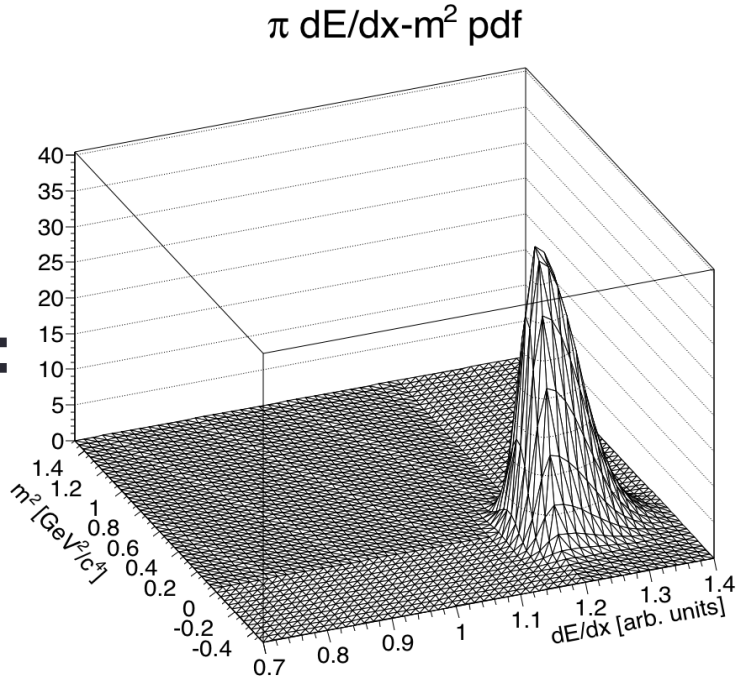
```
TSFunc2DProd Pi_2D_Pdf("Pi_2D_Pdf","#pi dE/dx-m^{2} pdf",Pi_dEdx_Pdf,Pi_m2_Pdf);
```



×



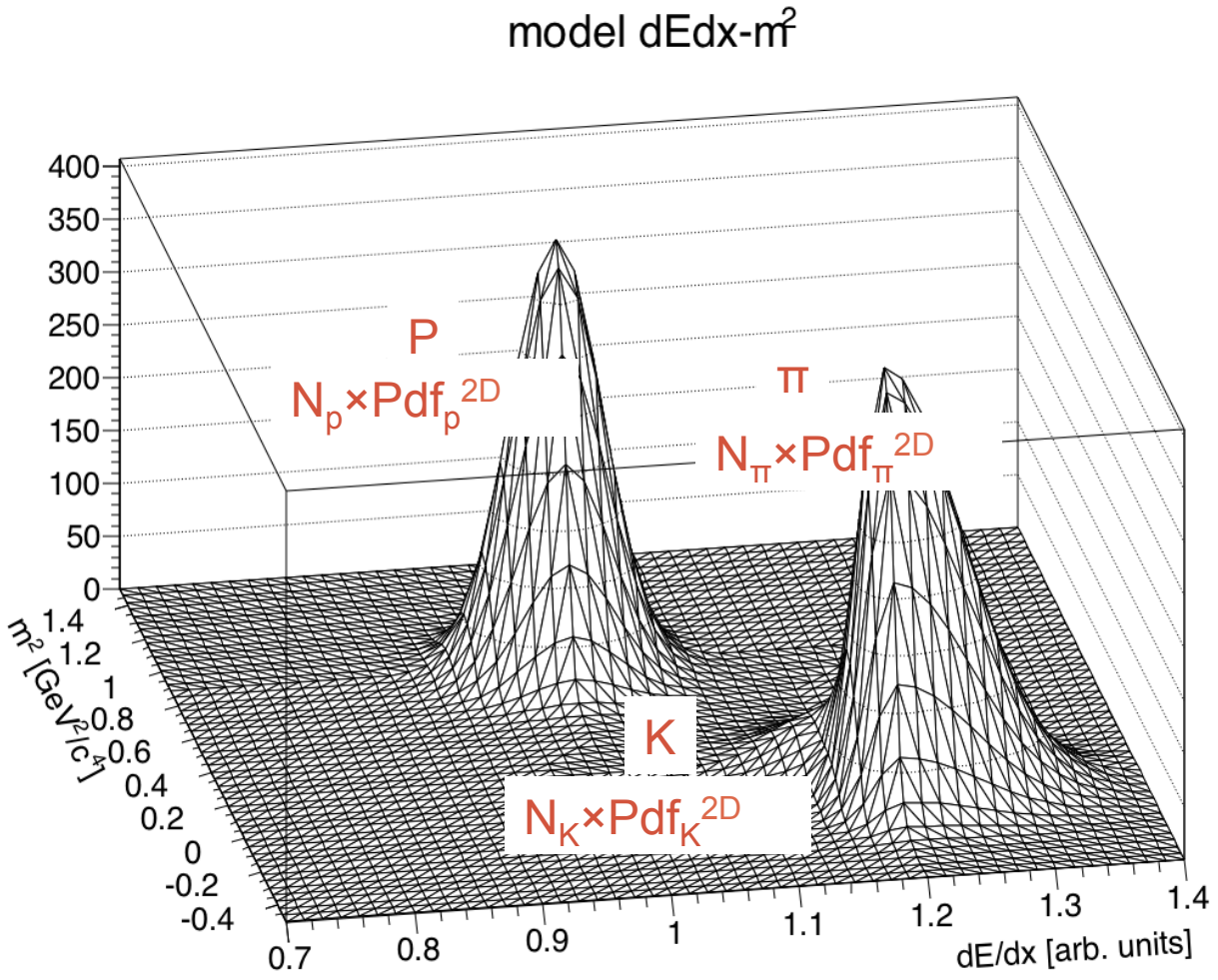
=




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

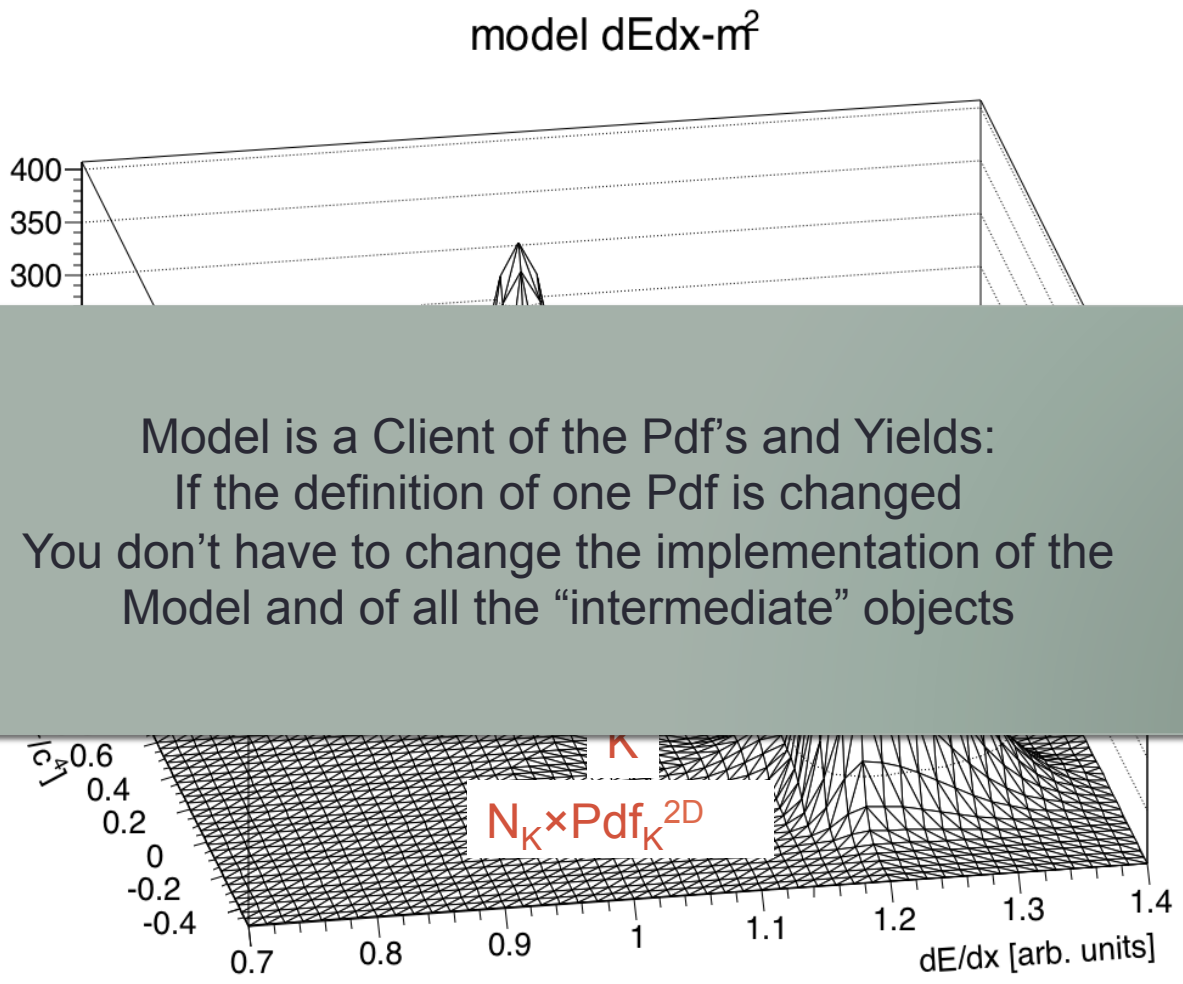
dE/dx vs ToF

```
// 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_2D.BuildGraph2D();
```

```
// 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 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();
```

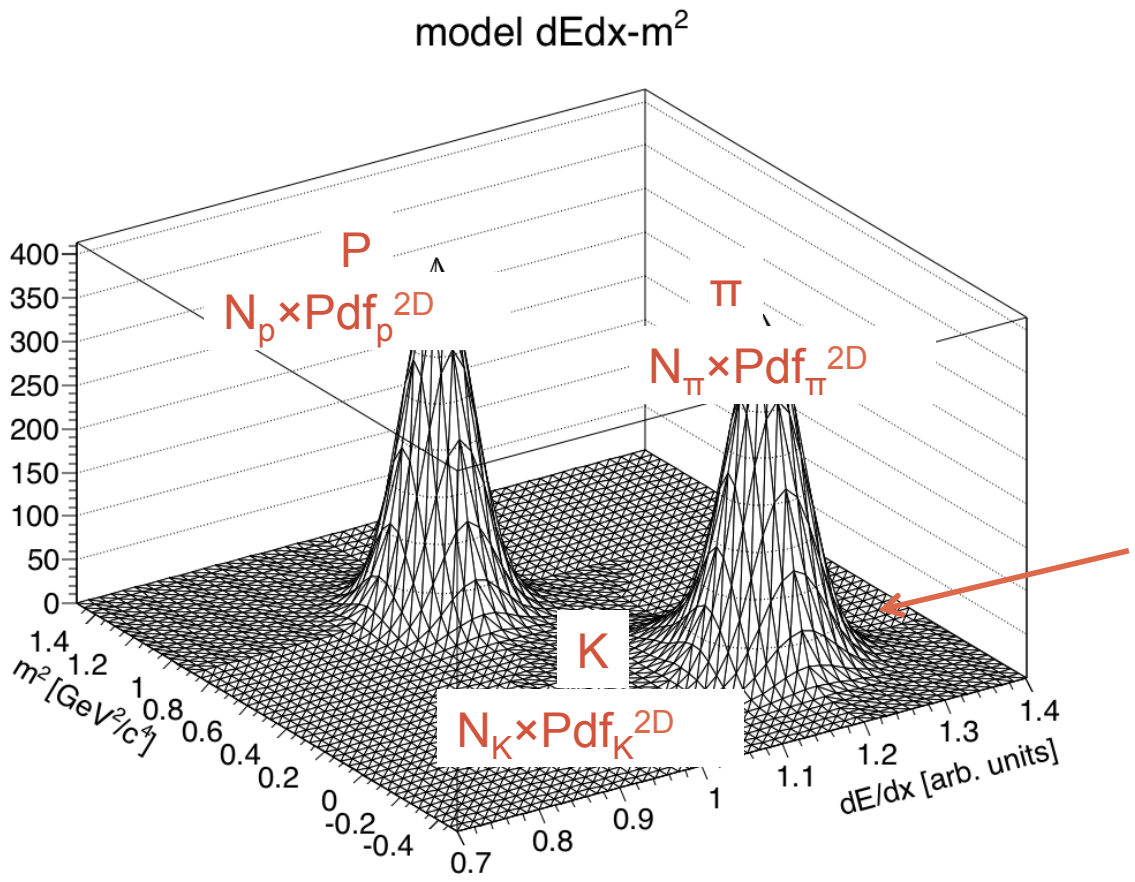
check it out

```
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: 10
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 ….




```
Model_2D.BuildGraph2D();
```

1- Parameters are objects
2- Functions are clients of their parameters.

a)

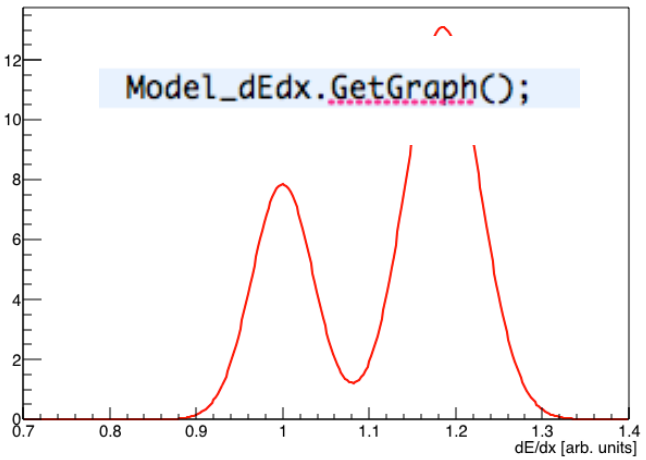
```
TSFunc1DGaussPdf Pi_dEdx_Pdf("Pi_dEdx_Pdf", "#pi dE/dx pdf", dEdx_x,dEdx_Pi_mean, dEdx_Pi_sig);
TSFunc1DGaussPdf P_dEdx_Pdf("P_dEdx_Pdf", "P dE/dx pdf", dEdx_x, dEdx_P_mean, dEdx_P_sig);
TSFunc1DGaussPdf K_dEdx_Pdf("K_dEdx_Pdf", "K dE/dx pdf", dEdx_x, dEdx_K_mean, dEdx_K_sig);
```

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.

b)

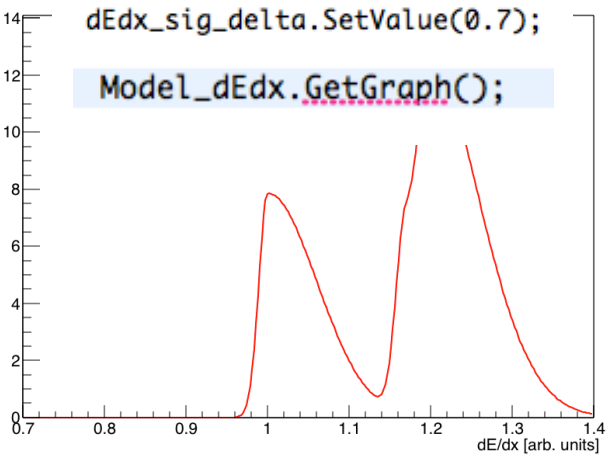
```
TSFunc1DAsymGaussPdf Pi_dEdx_Pdf("Pi_dEdx_Pdf", "#pi dE/dx pdf", dEdx_x,dEdx_Pi_mean, dEdx_Pi_sig, dEdx_sig_delta);
TSFunc1DAsymGaussPdf P_dEdx_Pdf("P_dEdx_Pdf", "P dE/dx pdf", dEdx_x, dEdx_P_mean, dEdx_P_sig, dEdx_sig_delta);
TSFunc1DAsymGaussPdf K_dEdx_Pdf("K_dEdx_Pdf", "K dE/dx pdf", dEdx_x, dEdx_K_mean,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

a)



Model is a client of
Pi_dEdx_Pdf
P_dEdx_Pdf,
K_dEdx_Pdf

b)



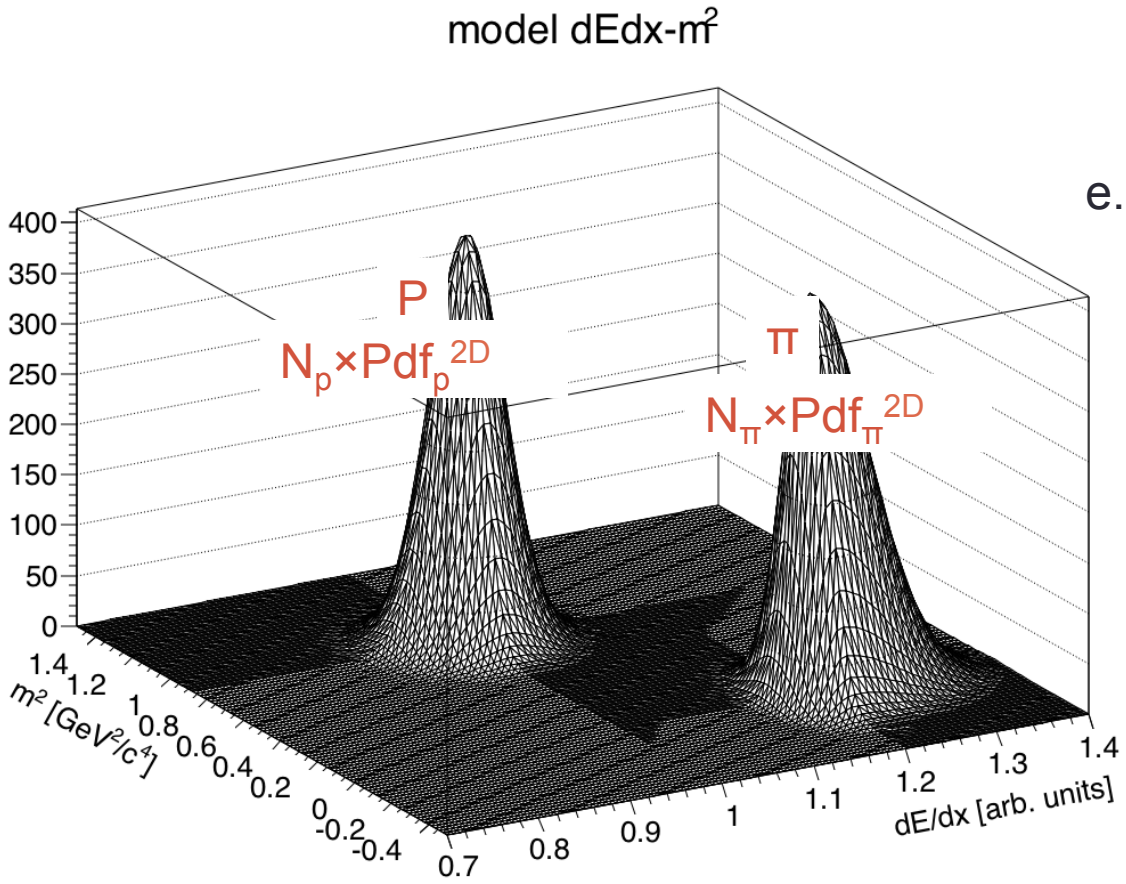
check it out

```
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));
```

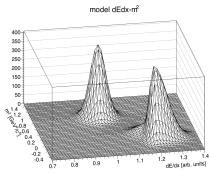
Prepare as many Model as needed…



e.g: P+π only

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

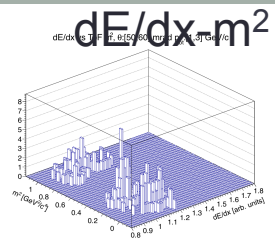
Minimization Function



1D or 2D

Model

Primary Params
(m^2,σ_{m2},σ_{dEdx},….)



dE/dx-m^2

1D or 2D

Data

Binned
or Unbinned

Minimization Function

X^2

Likelihood-Binned

Likelihood-Unbinned

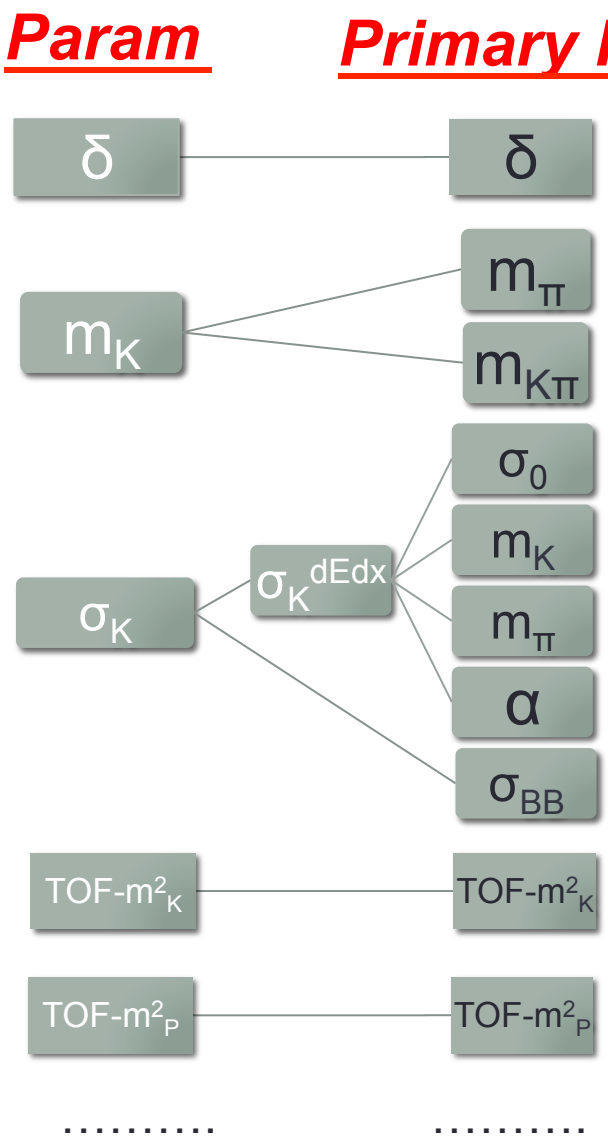
…. user defined

Calculated value
for a given set of Primary Params

Params & Primary Params

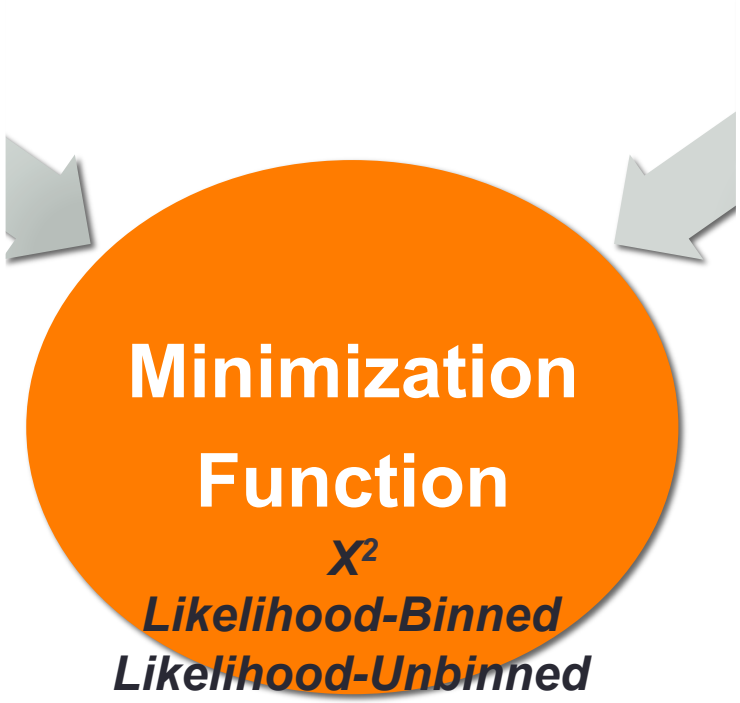
Pdf's, Model and then the Minimization Functions depend on the primary params

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



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));
```

Calculated value
for a given set of Primary Params

```
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();
```

$\chi 2$ 1D dE/dx

```
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();
```

MLL, 1D dE/dx

Unbinned

```
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();
```

X^2 2D dE/dx

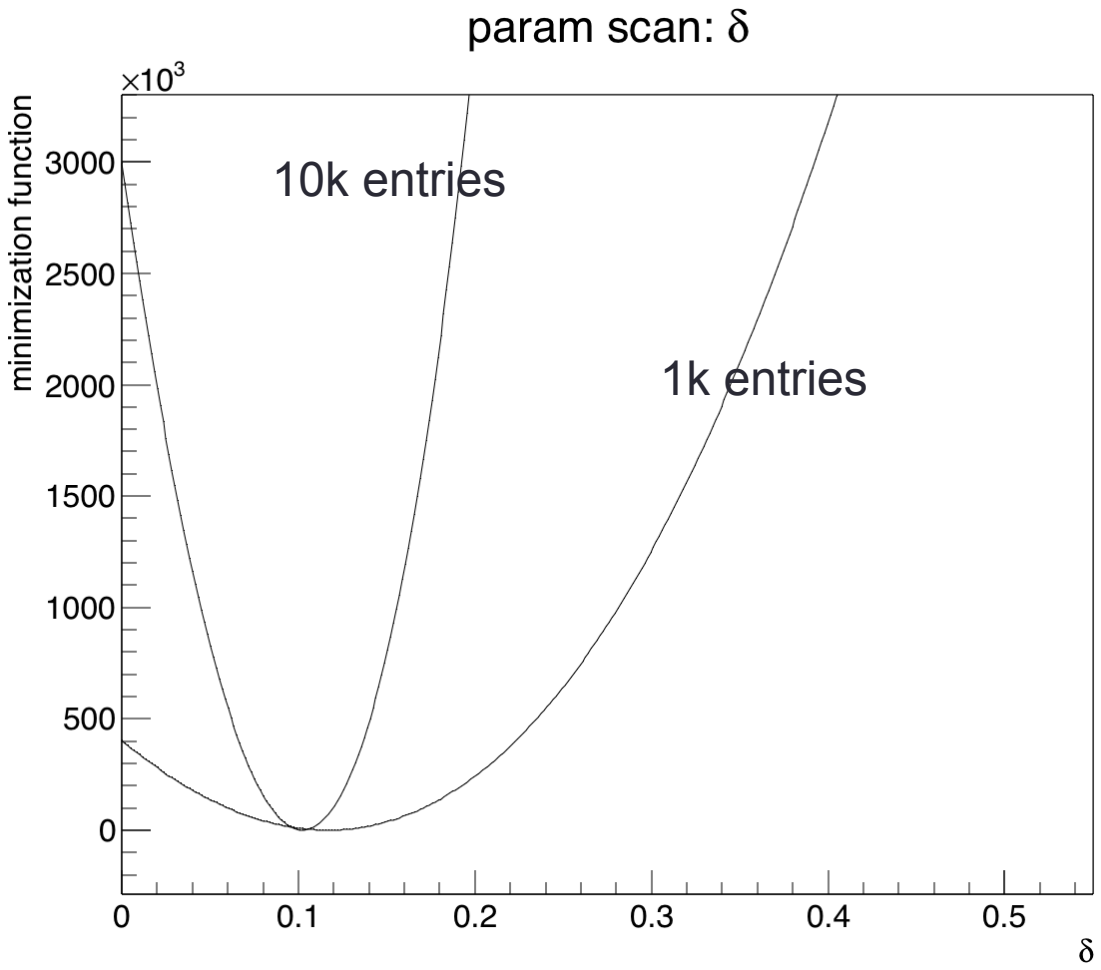
```
TSLikelihoodCalculator myLL_2d("myLL_2d","myLL_2d");
myLL_2d.SetModel(Model_2D);
myLL_2d.SaveHistory(true);
myLL_2d.SetBinnedData(myData,TSArgList(dEdx_x,m2_x));
myLL_2d.GetModel()->Print();
```

MLL, 2D dE/dx

Prepare Four Minimization Functions

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);
```

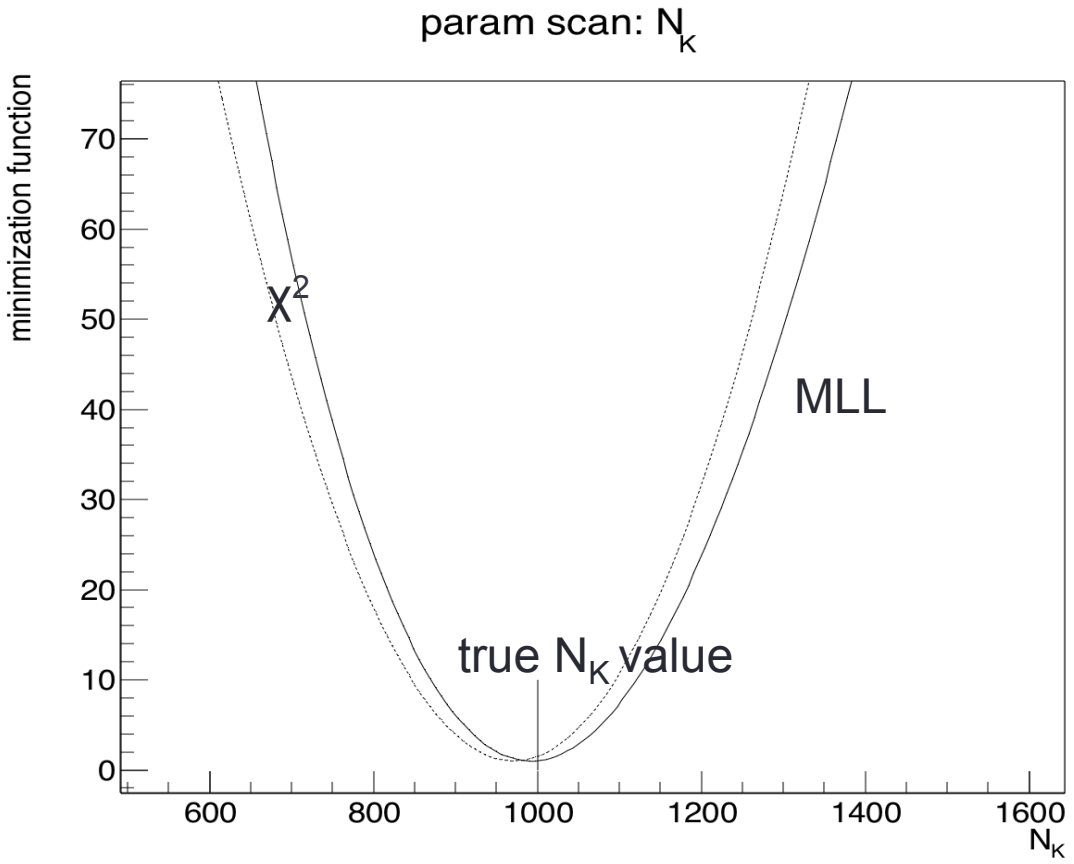


Parameters Scanning

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



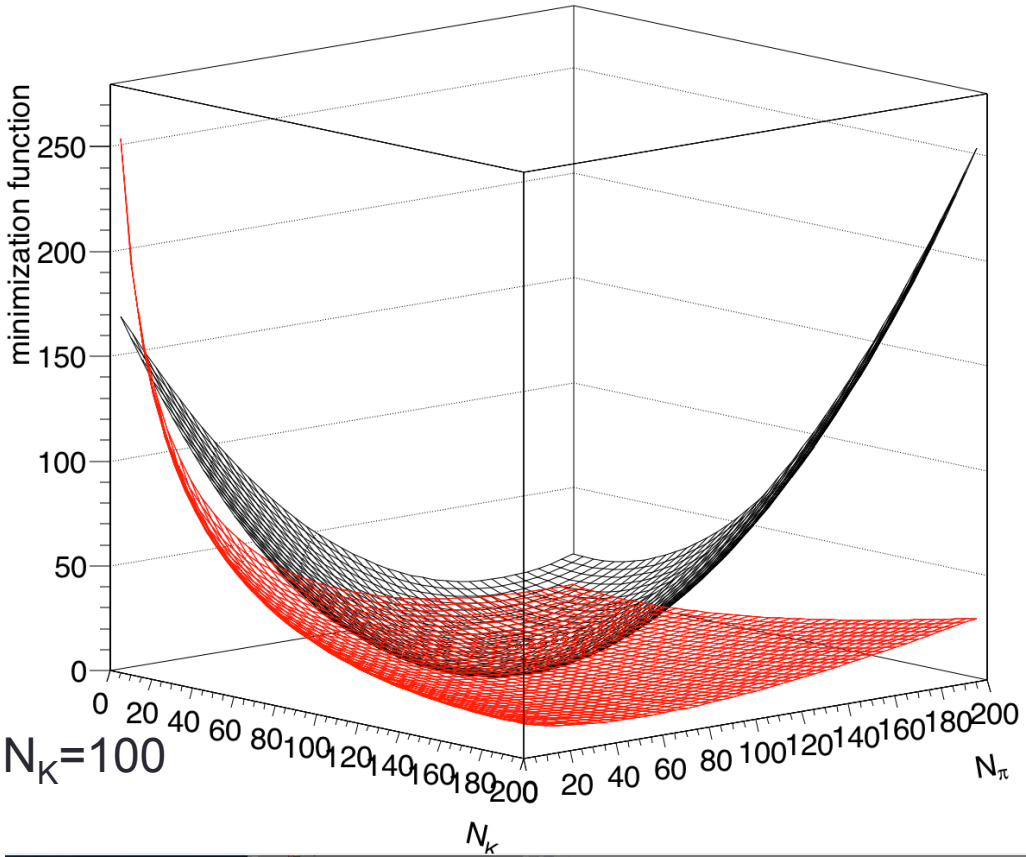
Parameters Scanning

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



True N_K=100

Parameters Scanning

Yield determination: 1D vs 2D Minimization Functions

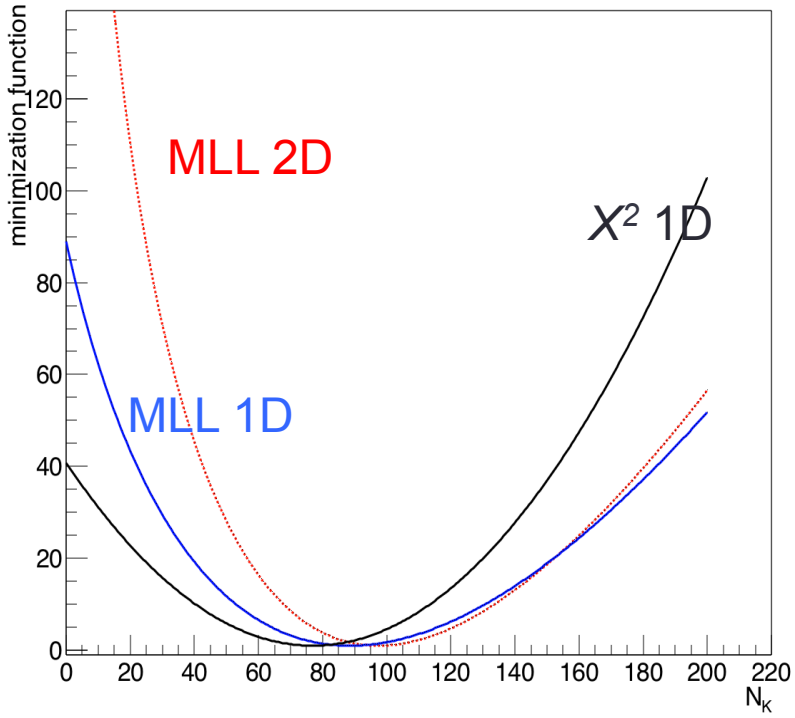
```
TGraph* gScan_NK_Chi2_2D=myChi2_2d.ScanParameter(NK,300,NK_min,NK_max);
gScan_NK_Chi2_2D->Write();

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: X^2 biased at low stat, larger errors

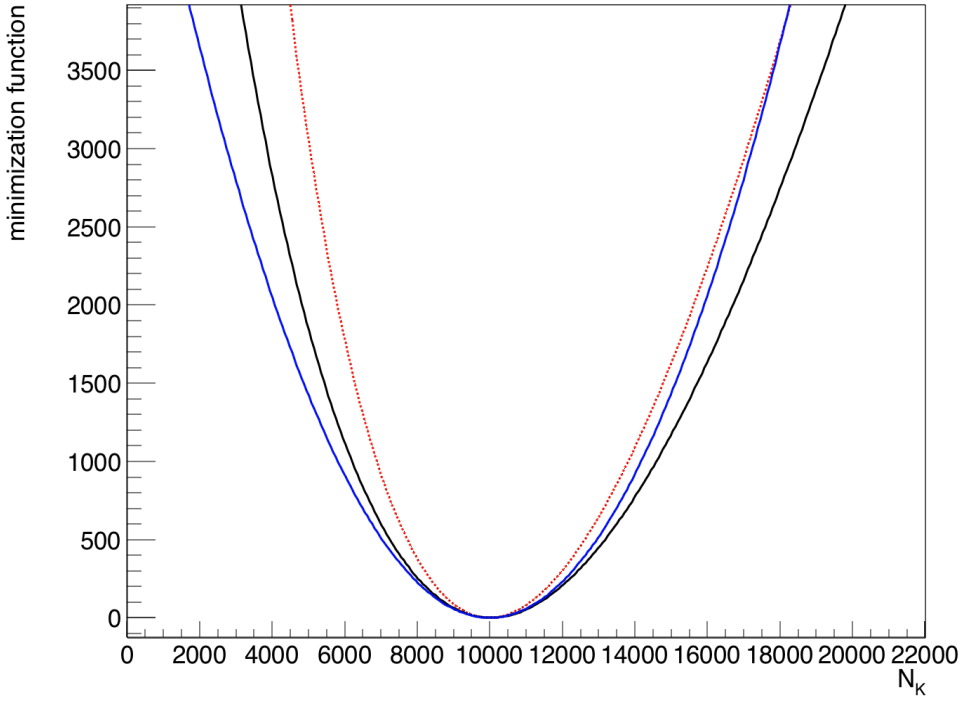
MLL 2D/1D: in 2D better peak discrimination: smaller error

param scan: N_K



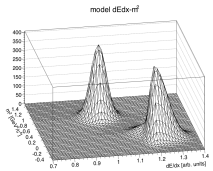
True N_K=100

param scan: N_K



True N_K=10k

Minimizers

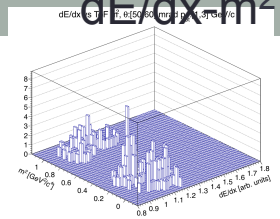


1D or 2D

Primary Params

$(m^2,\sigma_{m2},\sigma_{dEdx},\ldots.)$

Model



1D or 2D

Data

Binned or Unbinned

Minimization Function

X^2

Likelihood-Binned

Likelihood-Unbinned

…. user defined

Minimizer iteration (param variation)

$(m^2,\sigma_{m2},\sigma_{dEdx},\ldots.)^{trial}$

Calculated value for a given set of Primary Params

Minimizer

MINUIT2: MIGRAD

FUMILI…

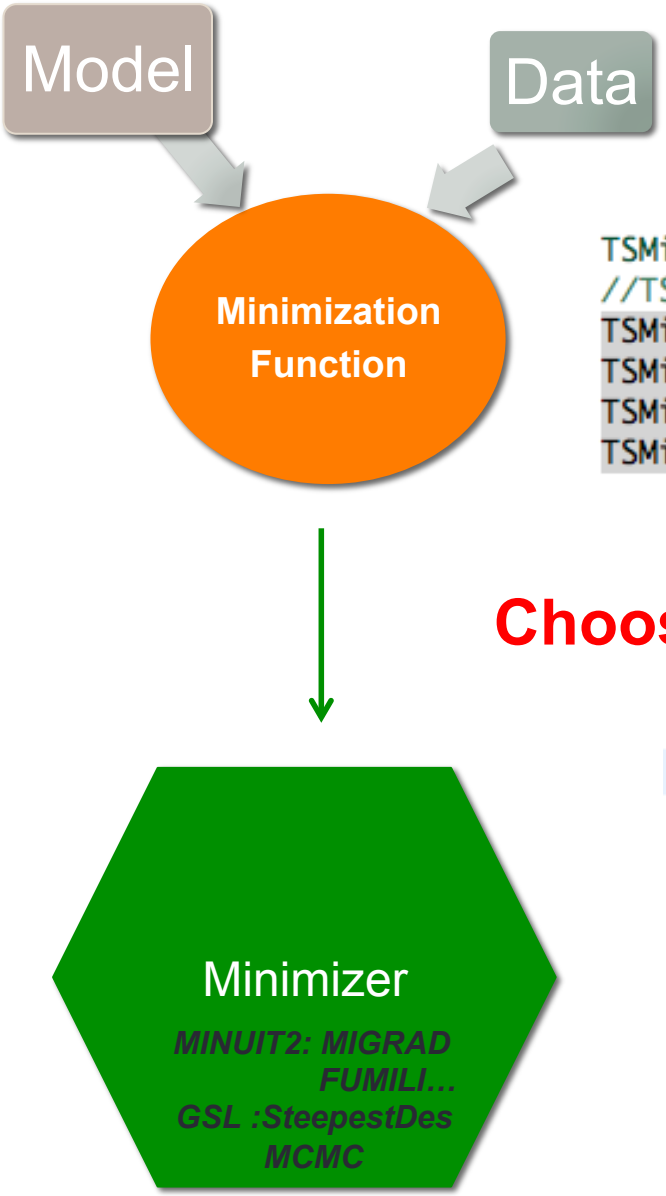
GSL :SteepestDes

MCMC

RESULT

$(m^2,\sigma_{m2},\sigma_{dEdx},\ldots.)^{best\text{-}fit}$

Func. Minimum



Setup Minimizer
Choose the algorithm……

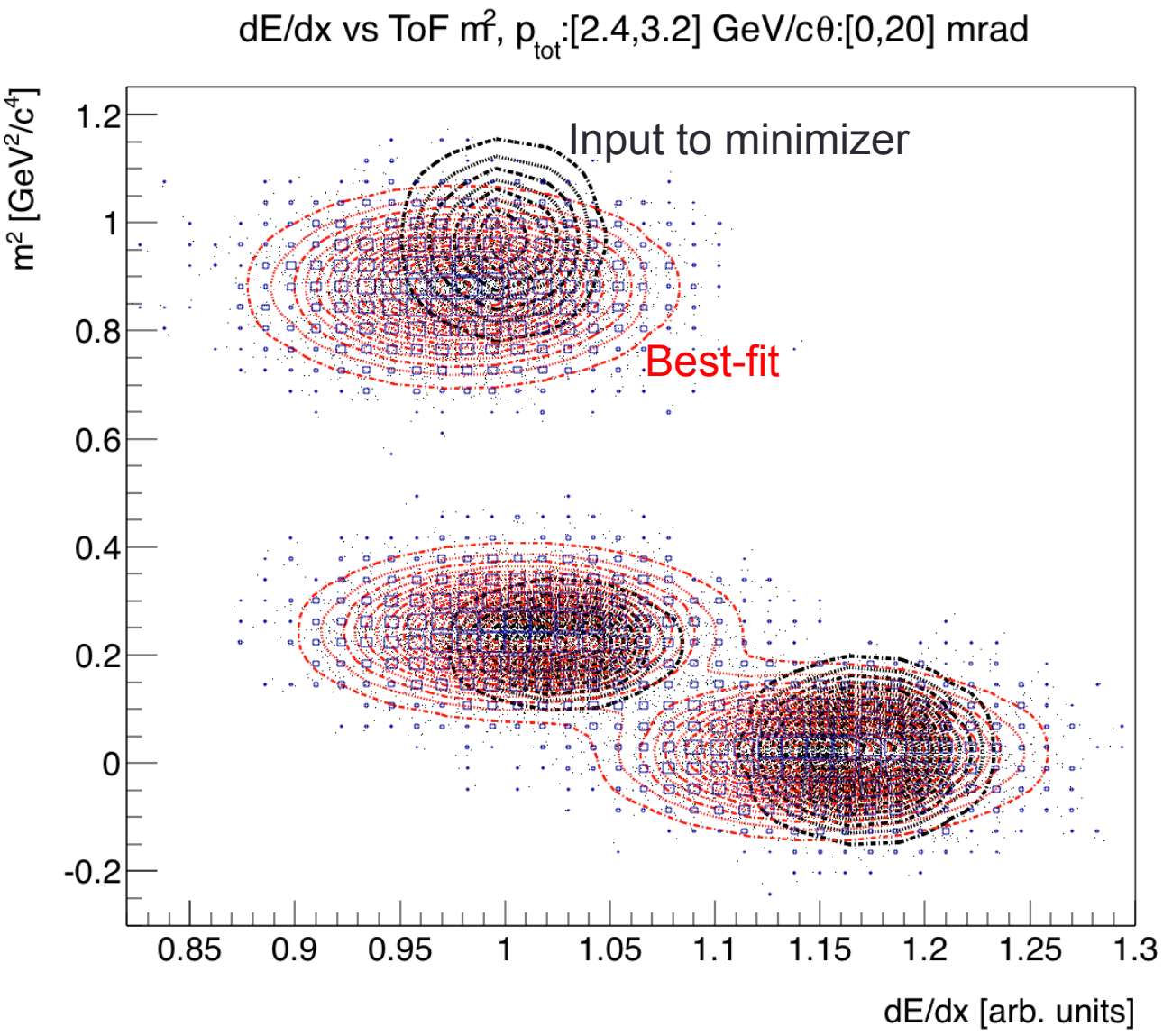
```
TSMinimizer TSMin("Minuit2","Combined");
//TSMinimizer TSMin("GSLMultiMin","SteepestDescent");
TSMin.GetMinimizer()->SetPrintLevel(0);
TSMin.GetMinimizer()->SetMaxFunctionCalls(1e6);
TSMin.GetMinimizer()->SetMaxIterations(1e4);
TSMin.GetMinimizer()->SetTolerance(0.001);
```

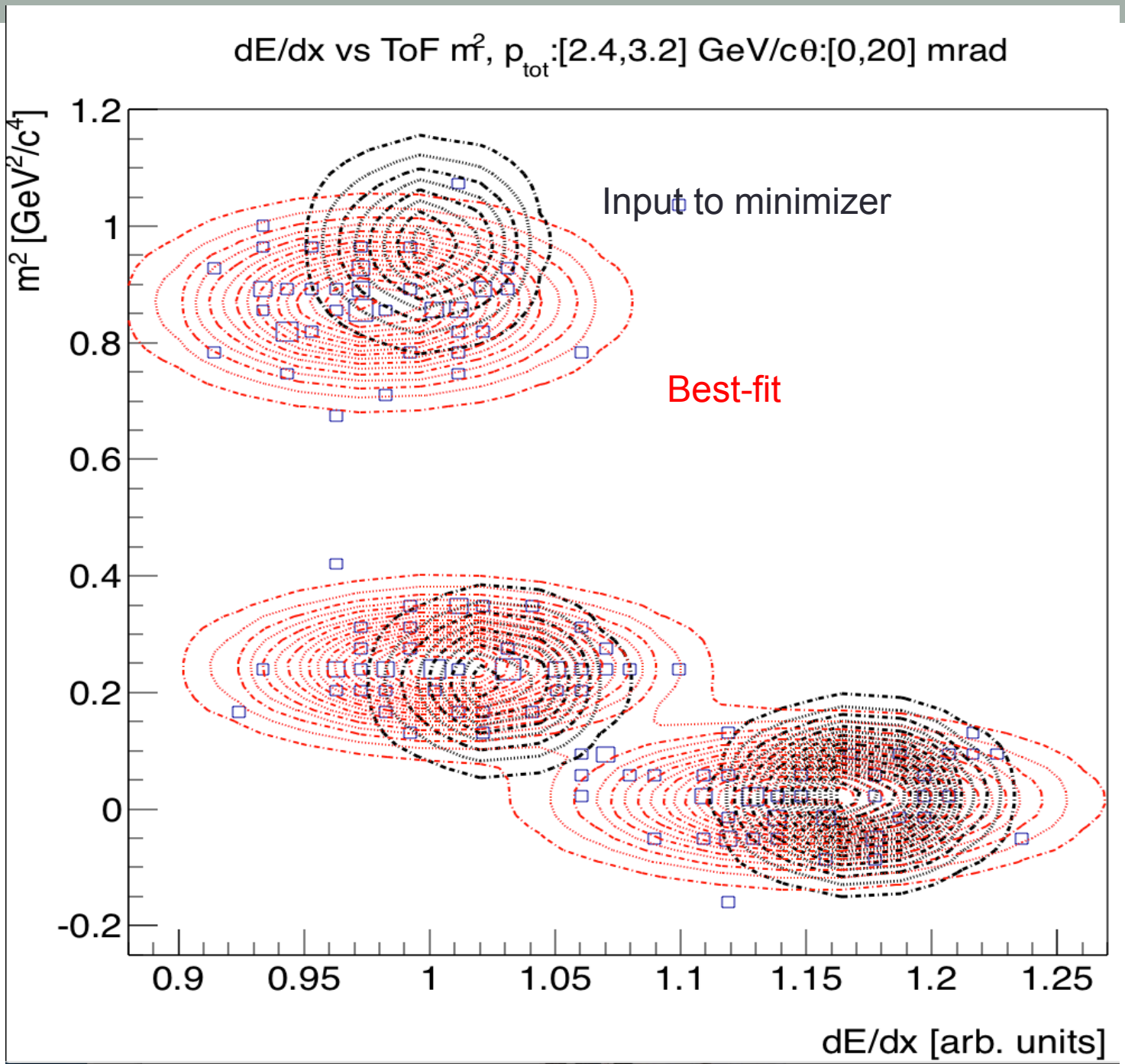
Choose the minimization function……

```
TSMin.SetFunction(myLL_2d);

TSMin.Minimize();
```

Putting things together

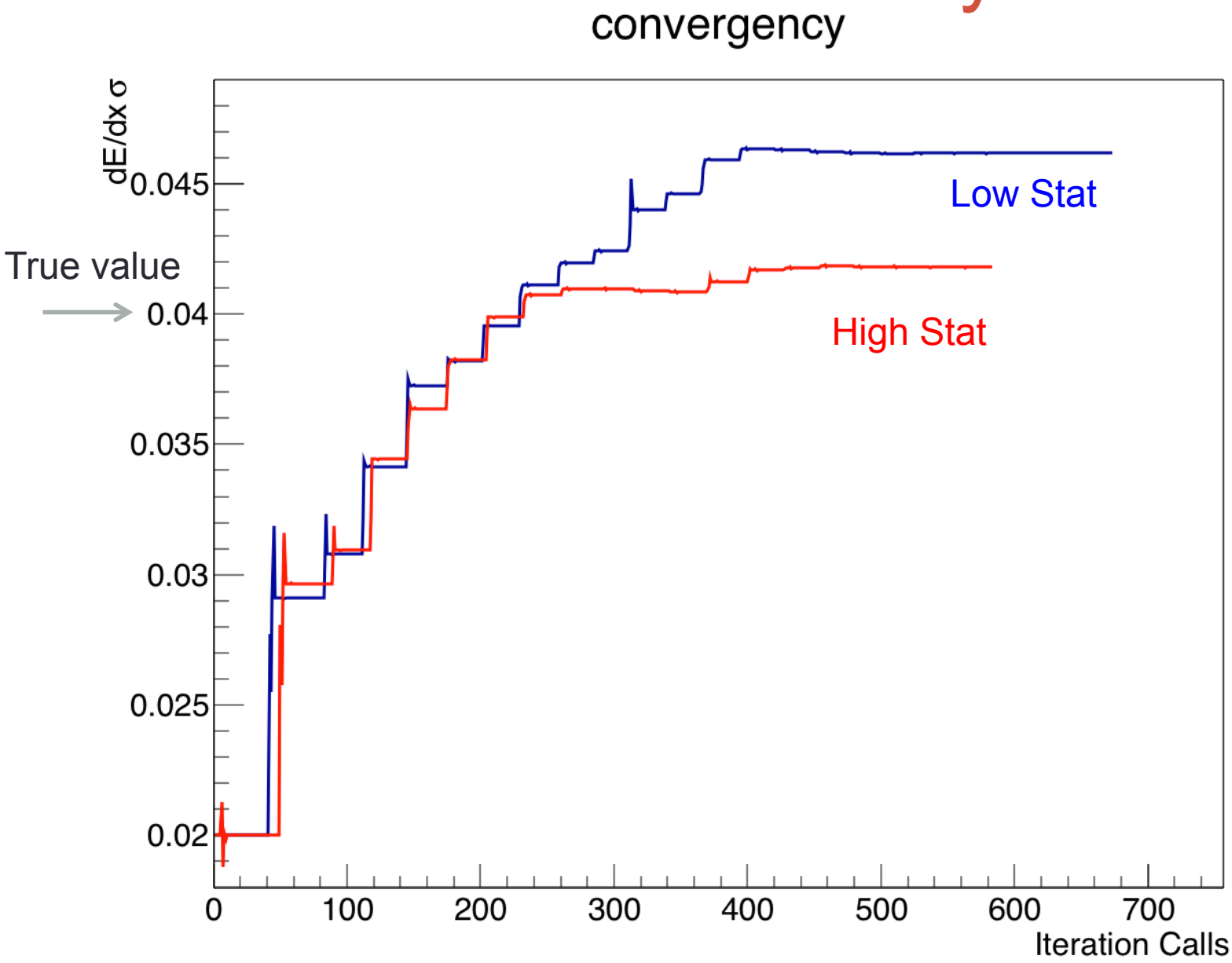
1- Setup Phase Space
2- Setup Data Bin Container
3- Setup DataSetMgr
4- Setup ToF Generator
5- Setup BetheBloch Function Mgr
6- Generate MC-Toy Events
7- Fill DataSetMgr
8- Setup Model
9- Setup Minimizer
10- Loop over DataSet
11- Fit Data

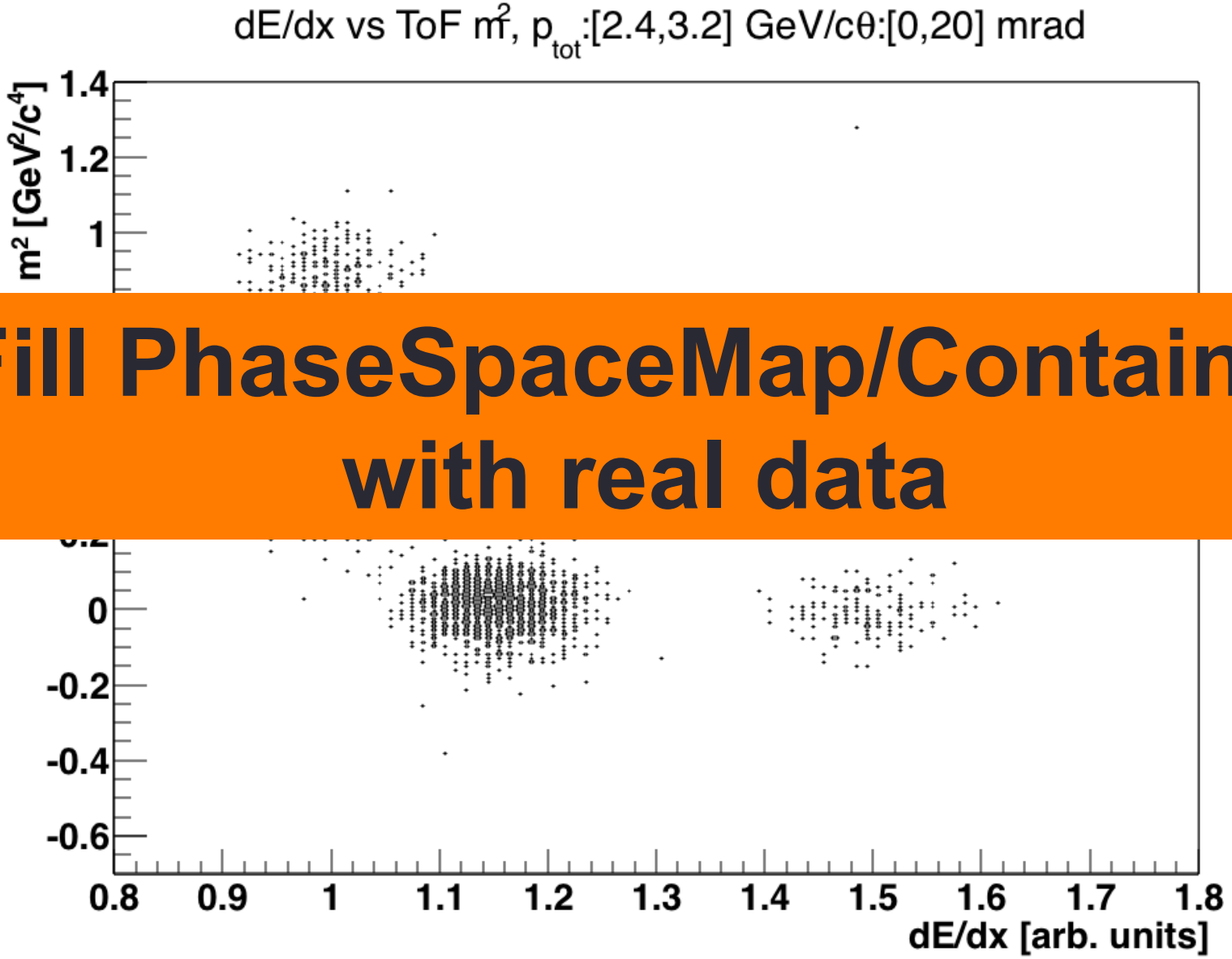

dE/dx vs ToF m², p_tot:[2.4,3.2] GeV/cθ:[0,20] mrad
Input to minimizer
Best-fit
m² [GeV²/c⁴]
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0.85
0.9
0.95
1
1.05
1.1
1.15
1.2
1.25
1.3
dE/dx [arb. units]


dE/dx vs ToF m², p_tot:[2.4,3.2] GeV/cθ:[0,20] mrad
m² [GeV²/c⁴]
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0.9
0.95
1
1.05
1.1
1.15
1.2
1.25
dE/dx [arb. units]
Input to minimizer
Best-fit

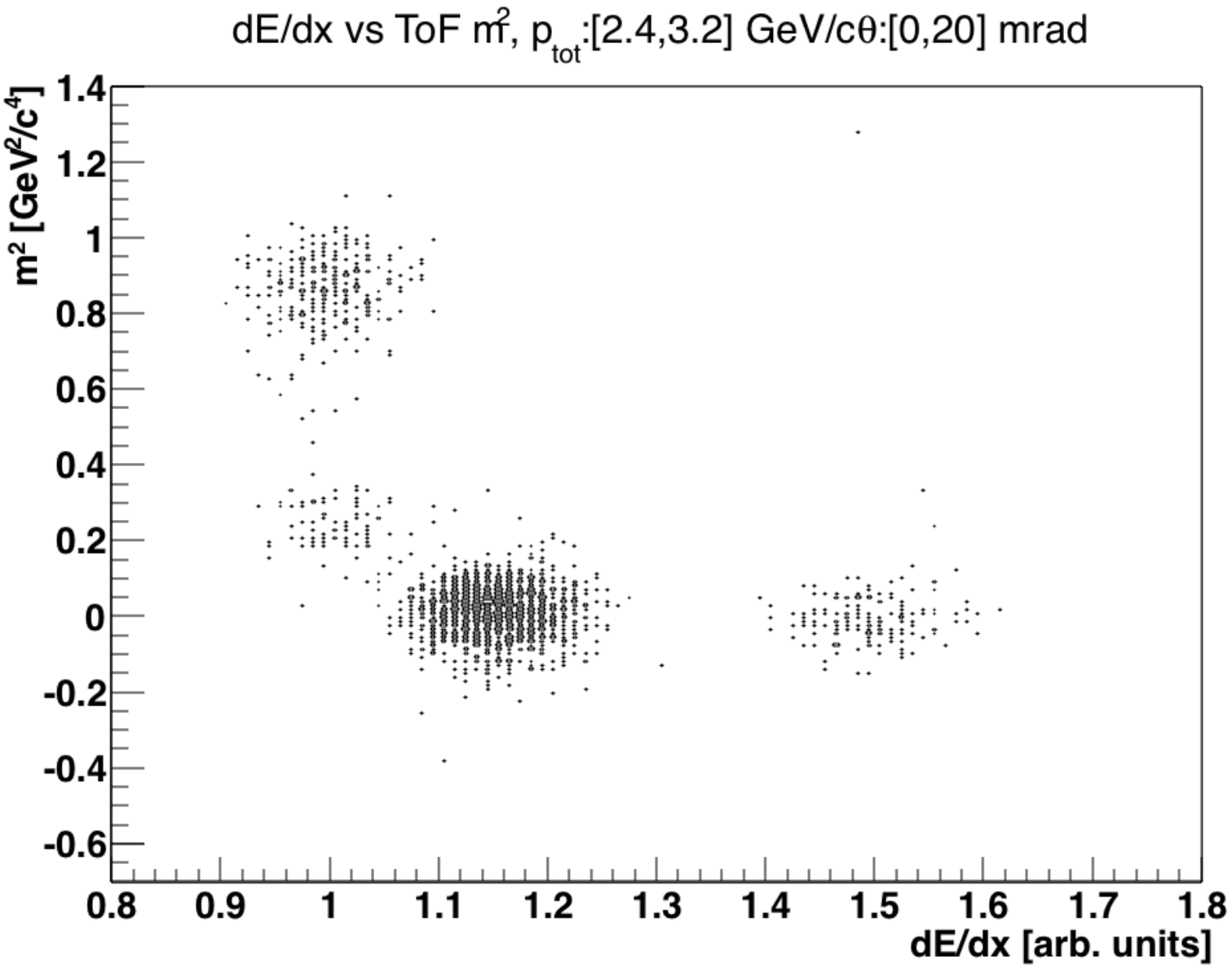
Lower Stat

Check Minimization History





Fill PhaseSpaceMap/Container with real data


dE/dx vs ToF m², p_tot:[2.4,3.2] GeV/cθ:[0,20] mrad
m² [GeV²/c⁴]
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-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 [arb. units]

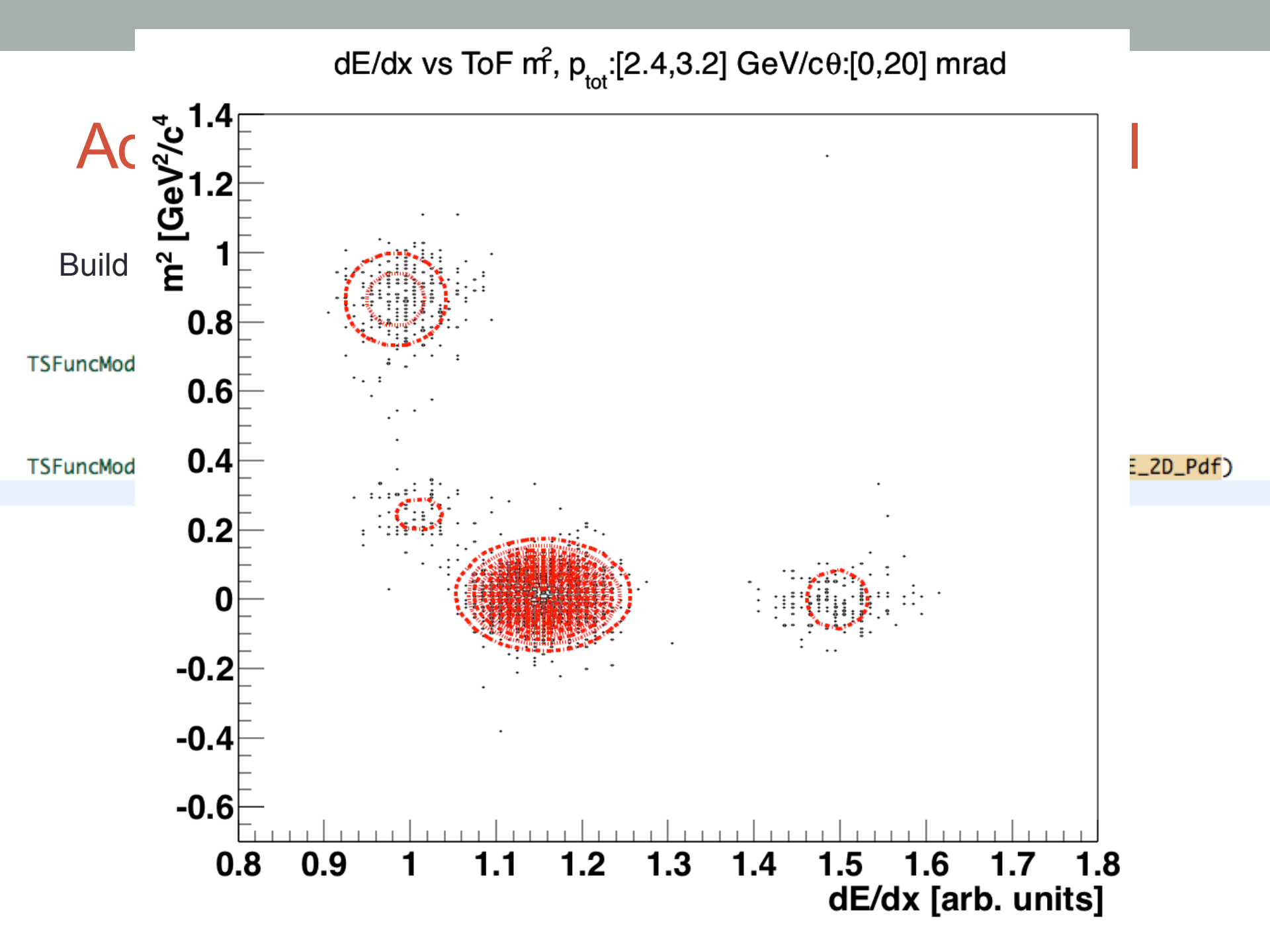
Adding another component to the model

Build a new component then add it to the existing model

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

```
TSFuncModel2D Model_2D("Model_2D","model dEdx-m^{2}",TSArgList(P_2D_Pdf,K_2D_Pdf,Pi_2D_Pdf,E_2D_Pdf)
                                                    ,TSArgList(NP,NK,NPi,NE));
```

Fit again….


dE/dx vs ToF m², p_tot:[2.4,3.2] GeV/cθ:[0,20] mrad
m² [GeV²/c⁴]
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-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 [arb. units]

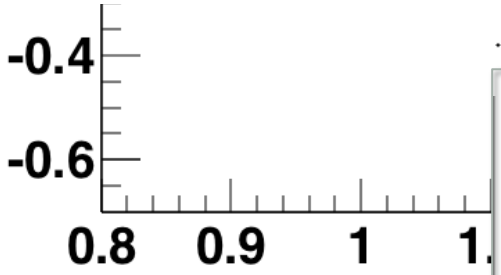
dE/dx vs ToF m^2, p_{tot}:[2.4,3.2] GeV/cθ:[0,20] mrad

Ac

Build


```
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 +/-0.00242417 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



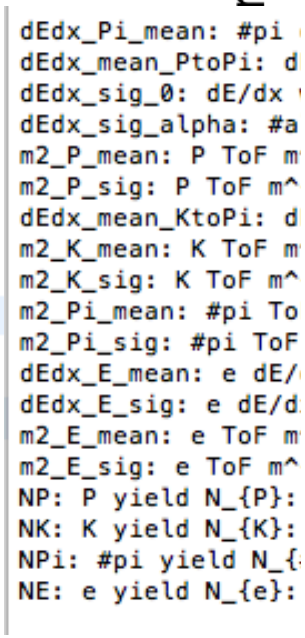

```
root [4]
root [4] TF1_Model_dEdx_MC_post->Integral(0.8,1.8)
(Double_t)2.01699866139916750e+01
root [5]
root [5]
```

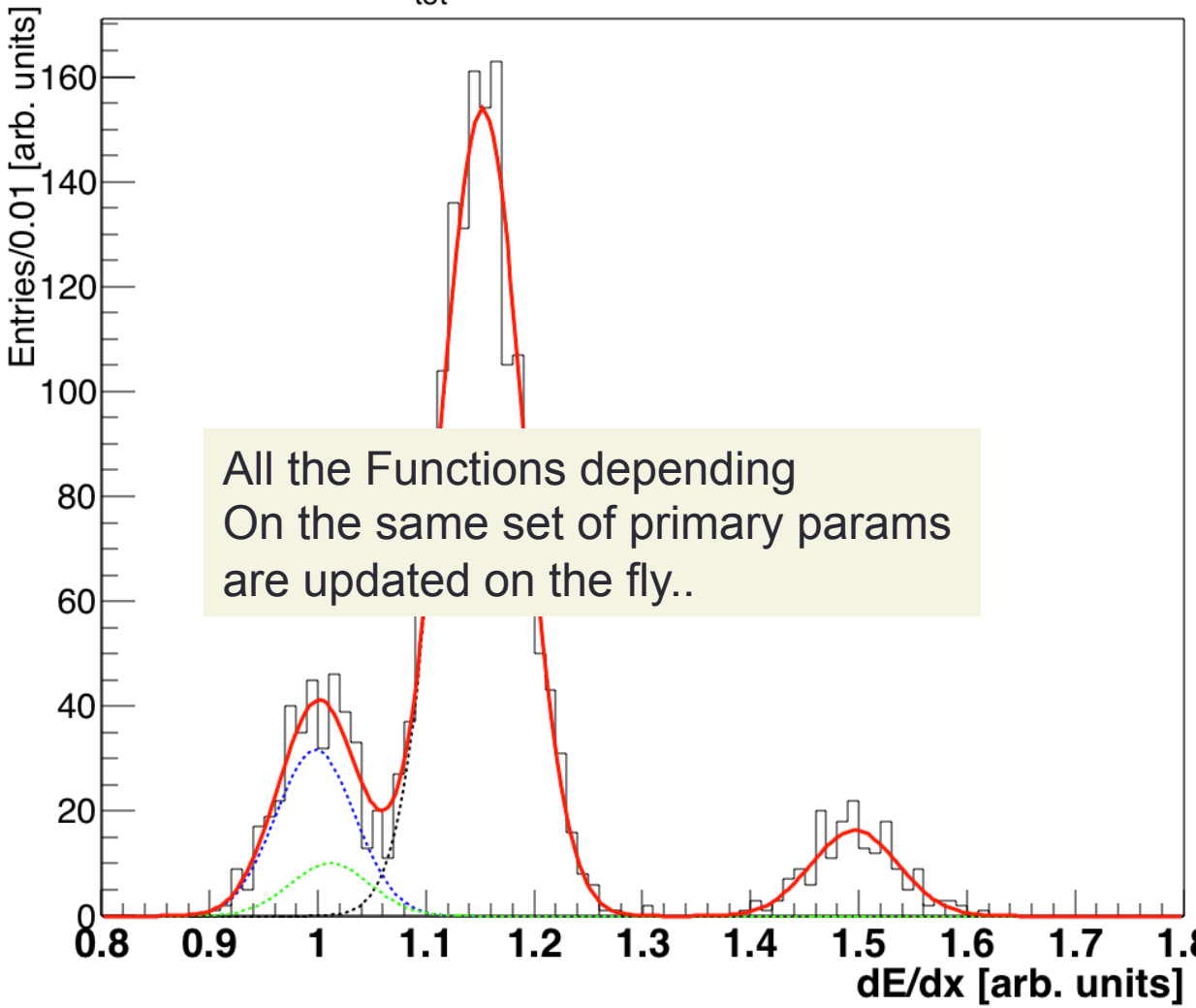
Integral of the Best Fit Model

dE/dx [arb. units]

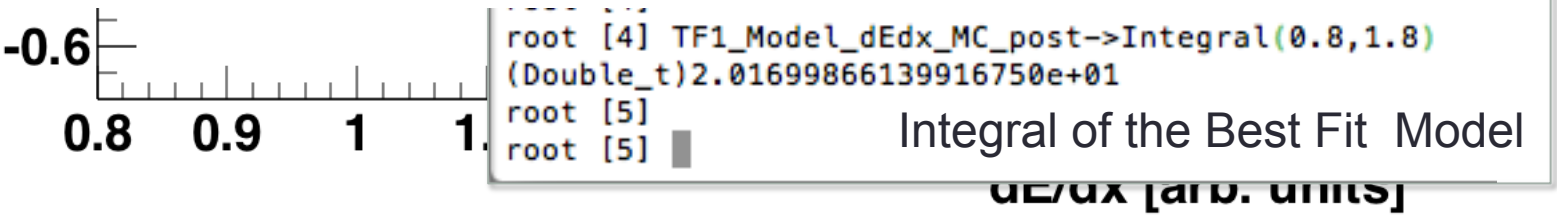
Ac

Build





All the Functions depending
On the same set of primary params
are updated on the fly..



Integral of the Best Fit Model

Building a Generic Model
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}");

TSPhaseSpaceMap mapKinLam("mapKinY","map y");
mapKinLam.SetVariable(Yvar);
```

Variables
Y,Pt,InvMass

pace

Test Model:

Bkg: Exponential

Sig: Gauss

ns

```
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));

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

Models

```
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}");

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);


 TSParameter Exp_m("Exp_m","Bkg_par_1","","");
 TSParameter Exp_s("Exp_s","Bkg_par_2","","");

 TSParameter Sig_m("Sig_m","Signal_mean","","GeV/c^{2}");
 TSParameter Sig_s("Sig_s","Sigal_width","","GeV/c^{2}");

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


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

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

Variables
Y,Pt,InvMass

Phase Space

Params

Pdf's

Models

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

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

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

  Yvar.SetValue(0.3);
  Pt.SetValue(0.3);

  Exp_m.SetValue(0);
  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();
  }
```

Data container

Data Set

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));

invmLL.GetModel()->Print();

TSMin.SetFunction(invmLL);

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

TSMin.Minimize();


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


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

Minimization function

Minimizer

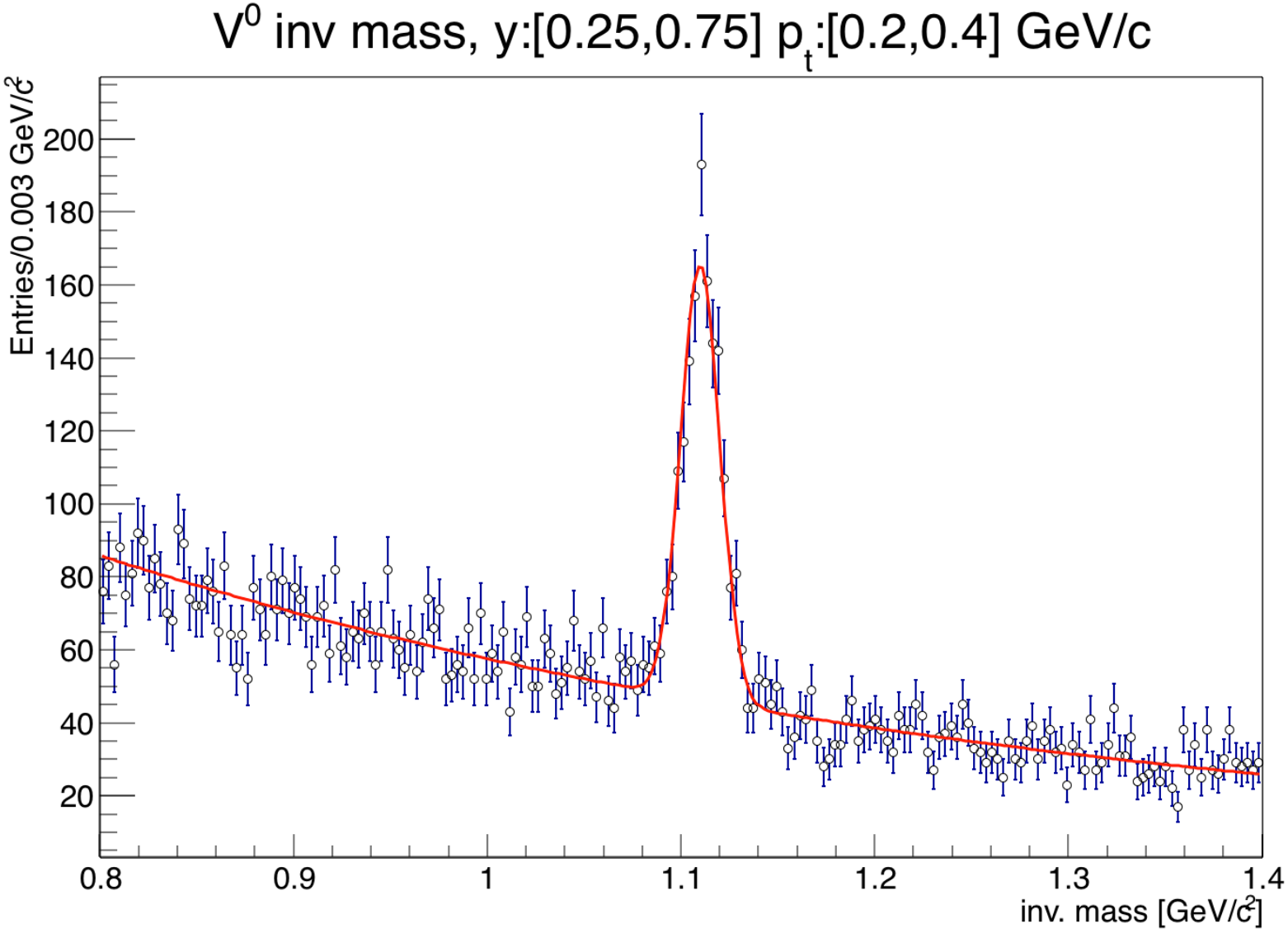
Minimization..

Get Histo

Get Model Func

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

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

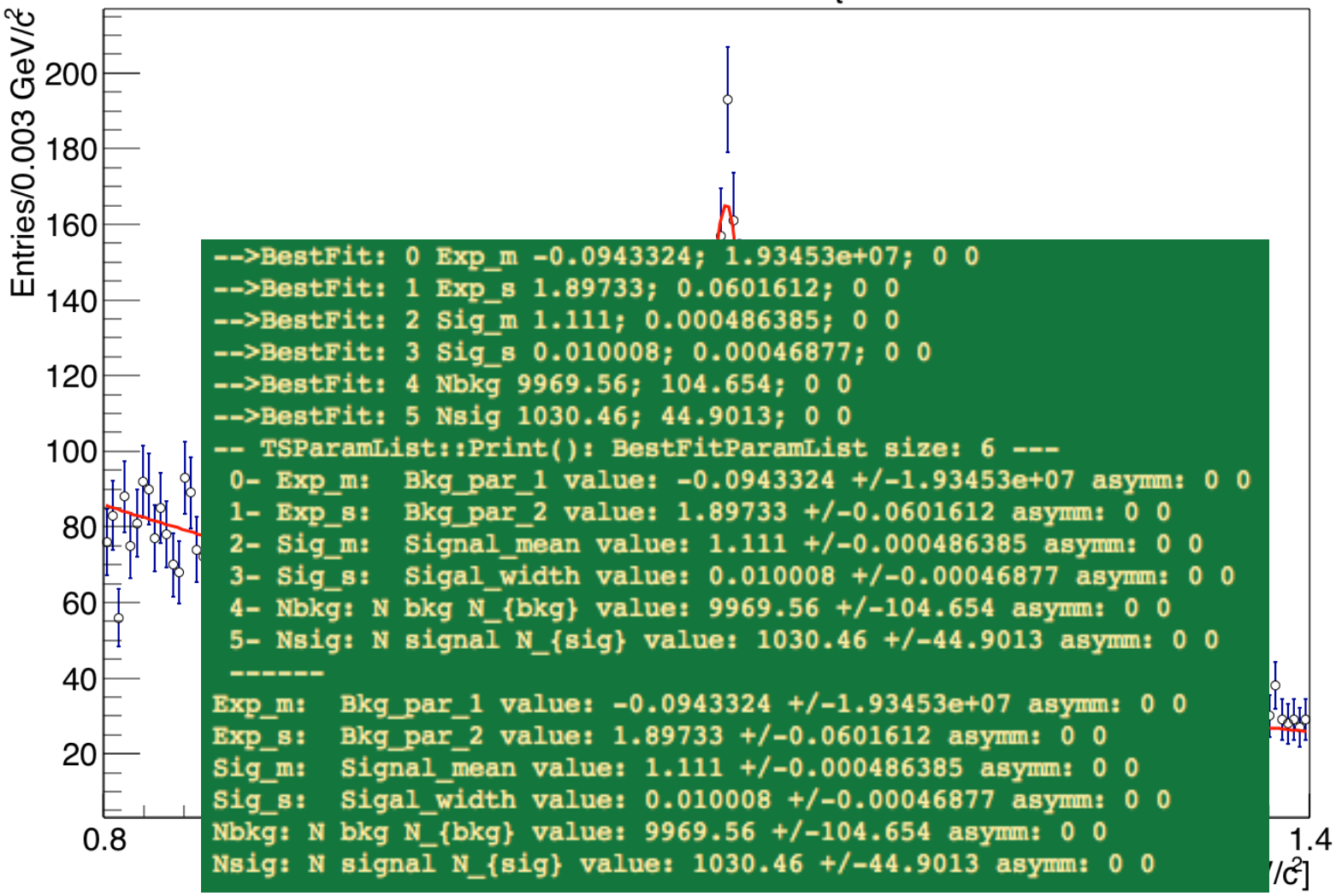


nction

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

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

nction




```
-->BestFit: 0 Exp_m -0.0943324; 1.93453e+07; 0 0
-->BestFit: 1 Exp_s 1.89733; 0.0601612; 0 0
-->BestFit: 2 Sig_m 1.111; 0.000486385; 0 0
-->BestFit: 3 Sig_s 0.010008; 0.00046877; 0 0
-->BestFit: 4 Nbkg 9969.56; 104.654; 0 0
-->BestFit: 5 Nsig 1030.46; 44.9013; 0 0
-- TSParamList::Print(): BestFitParamList size: 6 ---
 0- Exp_m:  Bkg_par_1 value: -0.0943324 +/-1.93453e+07 asymm: 0 0
 1- Exp_s:  Bkg_par_2 value: 1.89733 +/-0.0601612 asymm: 0 0
 2- Sig_m:  Signal_mean value: 1.111 +/-0.000486385 asymm: 0 0
 3- Sig_s:  Sigal_width value: 0.010008 +/-0.00046877 asymm: 0 0
 4- Nbkg: N bkg N_{bkg} value: 9969.56 +/-104.654 asymm: 0 0
 5- Nsig: N signal N_{sig} value: 1030.46 +/-44.9013 asymm: 0 0
 ------
Exp_m:  Bkg_par_1 value: -0.0943324 +/-1.93453e+07 asymm: 0 0
Exp_s:  Bkg_par_2 value: 1.89733 +/-0.0601612 asymm: 0 0
Sig_m:  Signal_mean value: 1.111 +/-0.000486385 asymm: 0 0
Sig_s:  Sigal_width value: 0.010008 +/-0.00046877 asymm: 0 0
Nbkg: N bkg N_{bkg} value: 9969.56 +/-104.654 asymm: 0 0
Nsig: N signal N_{sig} value: 1030.46 +/-44.9013 asymm: 0 0
```

PROJECT NEXT

Add Tools for
MC correctiong/Unfolding

…….

…..

Additional Material

Generic Particle Class

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

Main Tools

- PhaseSpace

Important Features

- Labelling: Automatic,Uniform,Consistent,Customizable

TSParticle::

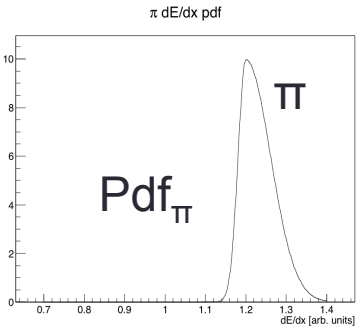
```
C("--- Particle List ----");

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();
```

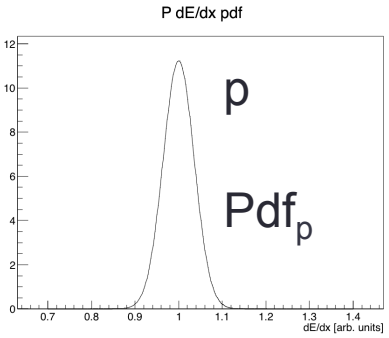
```
--- Particle List ----
#pi^{+}->#mu^{+}+X
p p q:+1 q>0 m=0.938272 GeV/c^{2} m^{2}=0.880354 GeV^{2}/c^{4}
KPlu K^{+} q:+1 q>0 m=0.493667 GeV/c^{2} m^{2}=0.243707 GeV^{2}/c^{4}
KMin K^{-} q:-1 q<0 m=0.493667 GeV/c^{2} m^{2}=0.243707 GeV^{2}/c^{4}
K0 K^{0} q:0 q=0 m=0.497648 GeV/c^{2} m^{2}=0.247654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

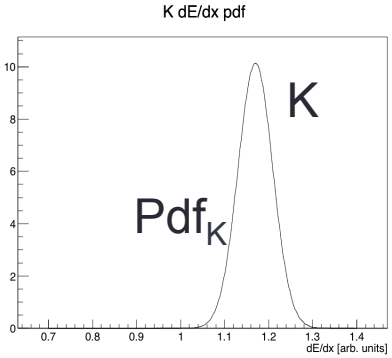
TSParticle::

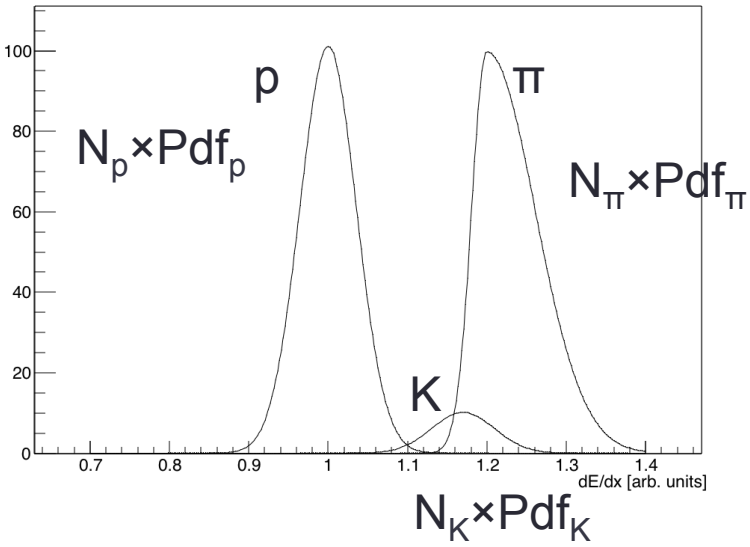
```
C("--- Particle List ----");

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);
```

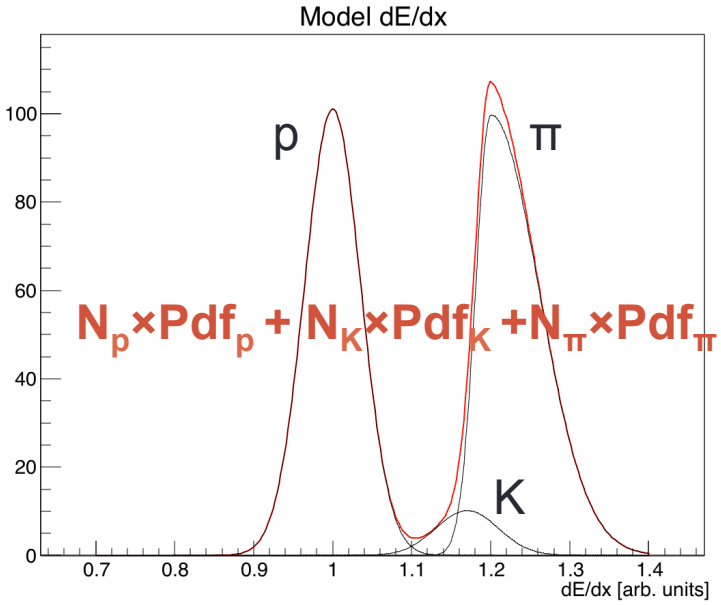
```
void SetCharge(int);
void SetMass(double);
void SetTau(double);//s
double GetBeta(double P) const;
double GetGamma(double P) const;
double GetBetaGamma(double) const;
double GetE(double P) const; //GeV
double GetTau() const;//s
double GetCTau() const;//m
double GetDecayTime(double P) const; //s
double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


π dE/dx pdf
π
Pdf$_\pi$
dE/dx [arb. units]


P dE/dx pdf
p
Pdf$_p$
dE/dx [arb. units]


K dE/dx pdf
K
Pdf$_K$
dE/dx [arb. units]


p
π
K
N$_p$×Pdf$_p$
N$_\pi$×Pdf$_\pi$
N$_K$×Pdf$_K$
dE/dx [arb. units]


Model dE/dx
p
π
K
N$_p$×Pdf$_p$ + N$_K$×Pdf$_K$ +N$_\pi$×Pdf$_\pi$
dE/dx [arb. units]