Computational Physics

Analysis of Large Statistical Data Sets:

Hadron Spectroscopy

Prof. Paul Eugenio
Department of Physics
Florida State University
April 09, 2019

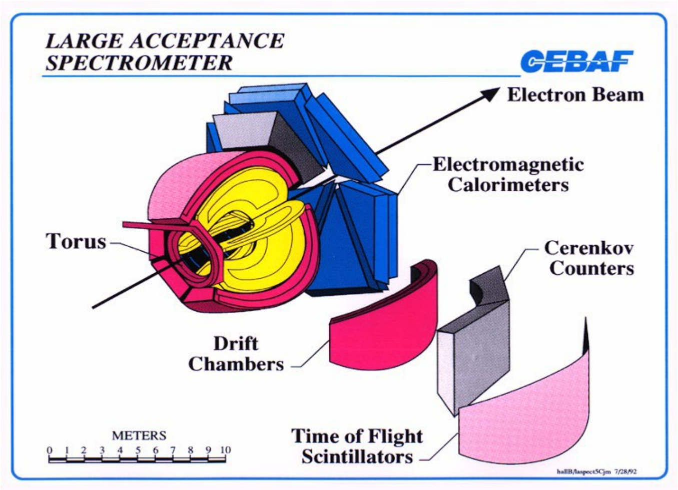
http://comphy.fsu.edu/~eugenio/comphy/

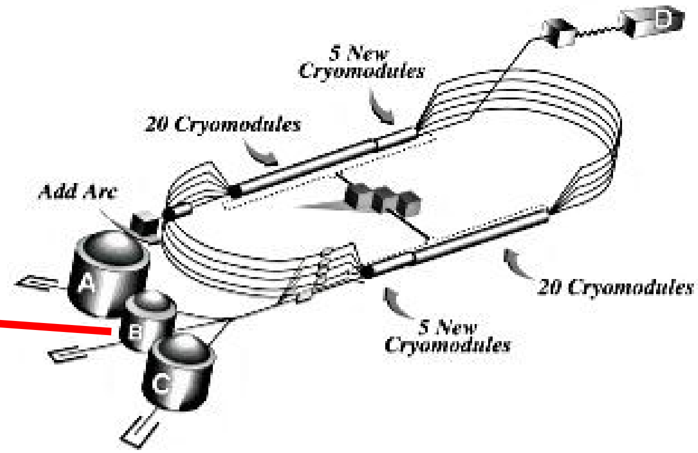
eugenio@fsu.edu

Exercise 10

Analysis of Large Statistical Data Sets

Photoproduction of Mesons





Hadron Spectroscopy

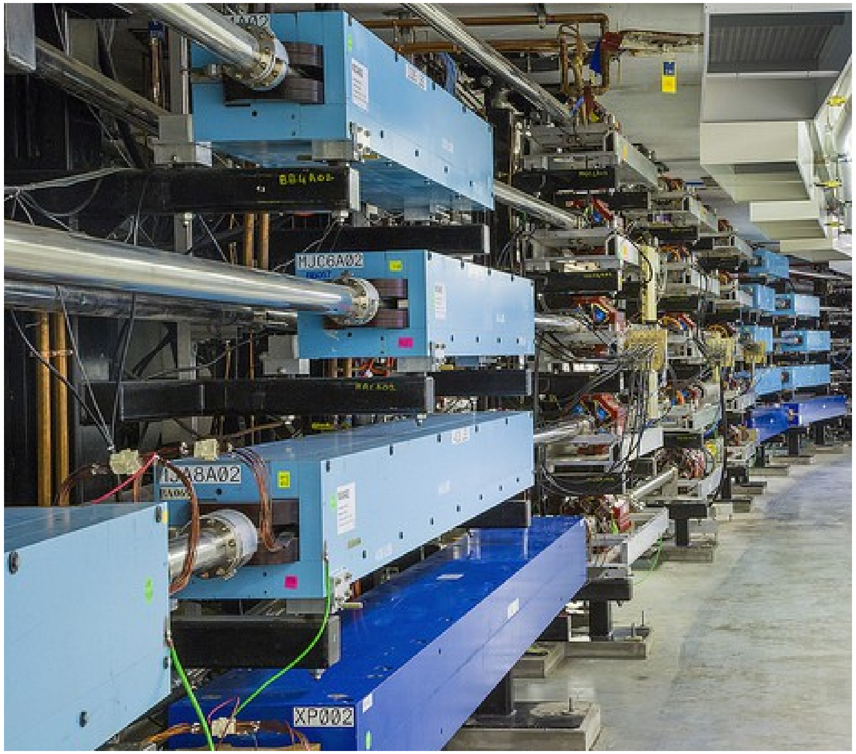
Studies of hadronic spectra via strong decays of hadrons provide insight regarding the Strong Force (*i.e.* QCD) at the confinement scale. These studies have led to phenomenological models for QCD such as the constituent quark model.

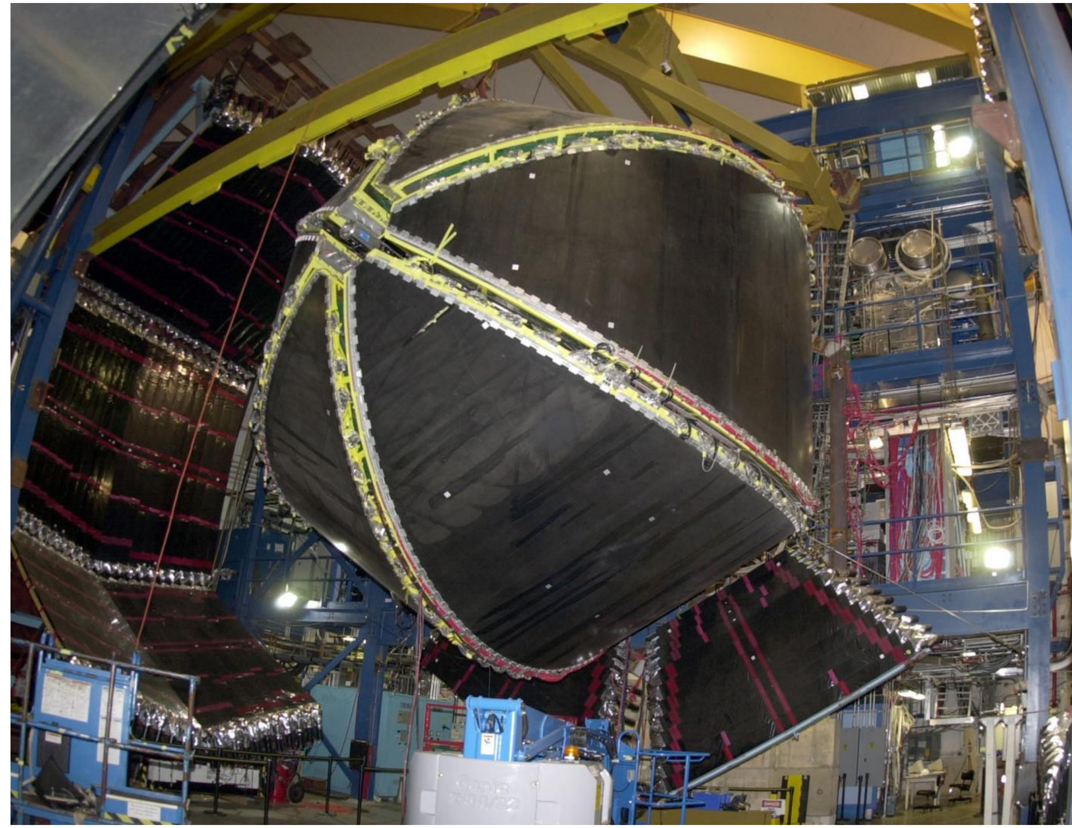
$$\gamma\, p \rightarrow n\, \pi^{+}\, \pi^{+}\, \pi^{-}$$

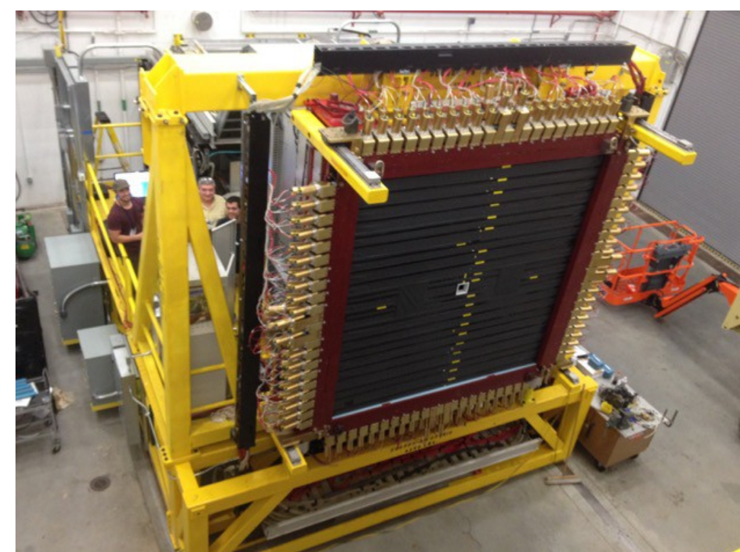
- 5 GeV/*c* tagged photon beam
- Liquid Hydrogen target
- over 1,000,000 events acquired

Goal: Find New Forms of Hadronic Matter

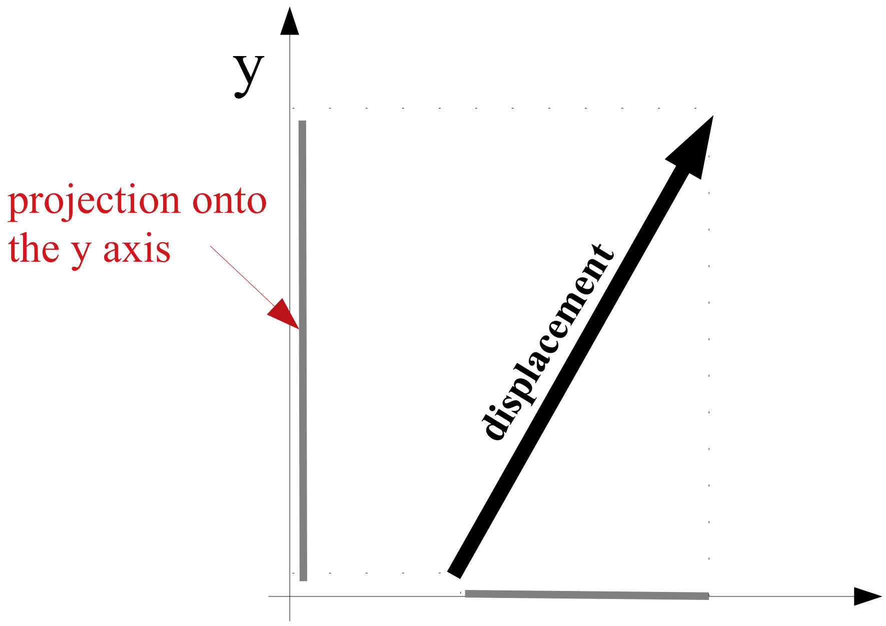
Jefferson National Accelerator Lab







Three-Vectors

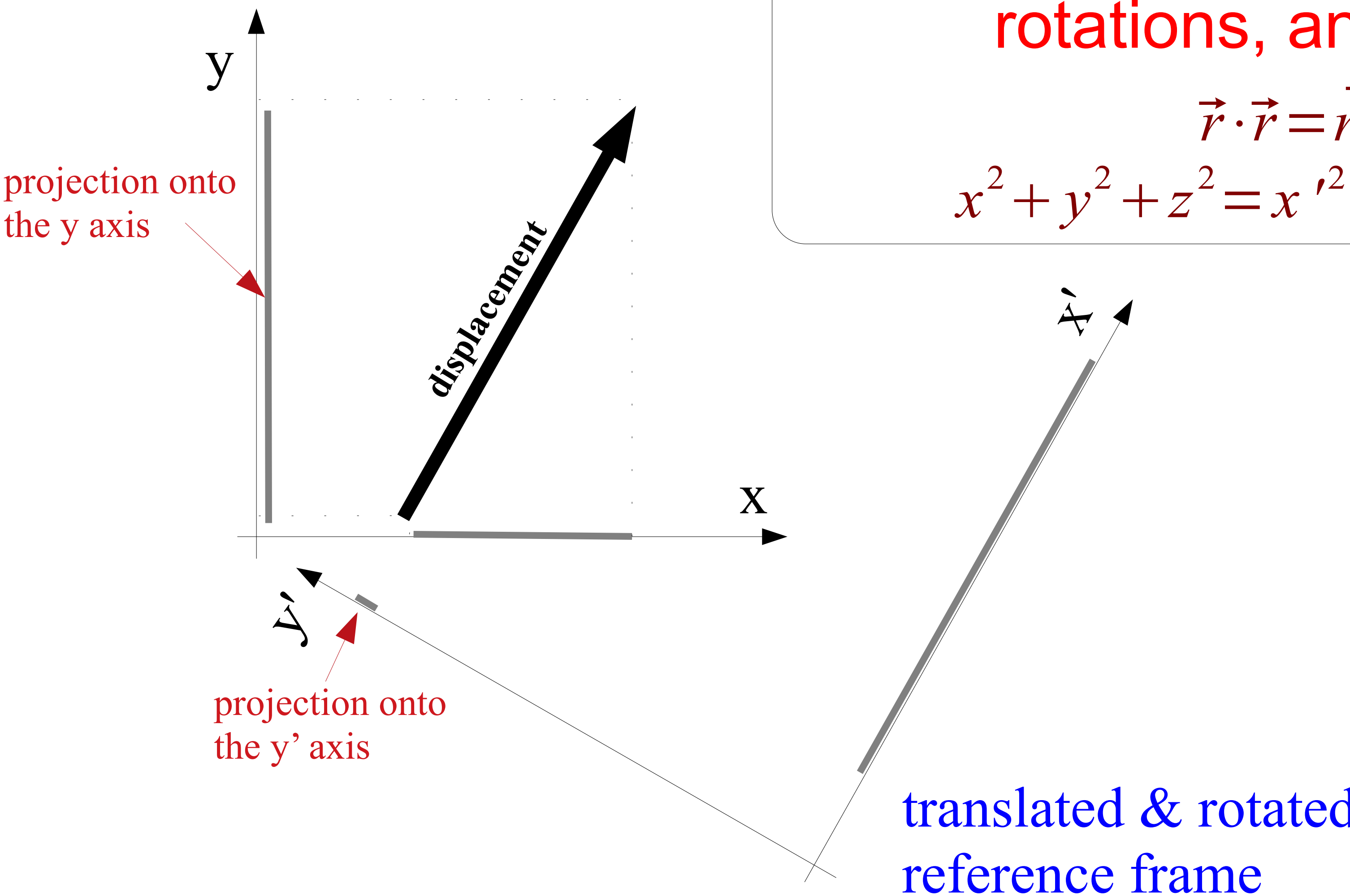


Three-Vectors

The Three-Vector magnitude is invariant under translations, rotations, and boosts.

$$\vec{r}\cdot\vec{r}=\vec{r}\,'\cdot\vec{r}\,'$$

$$x^2+y^2+z^2=x'^2+y'^2+z'^2$$



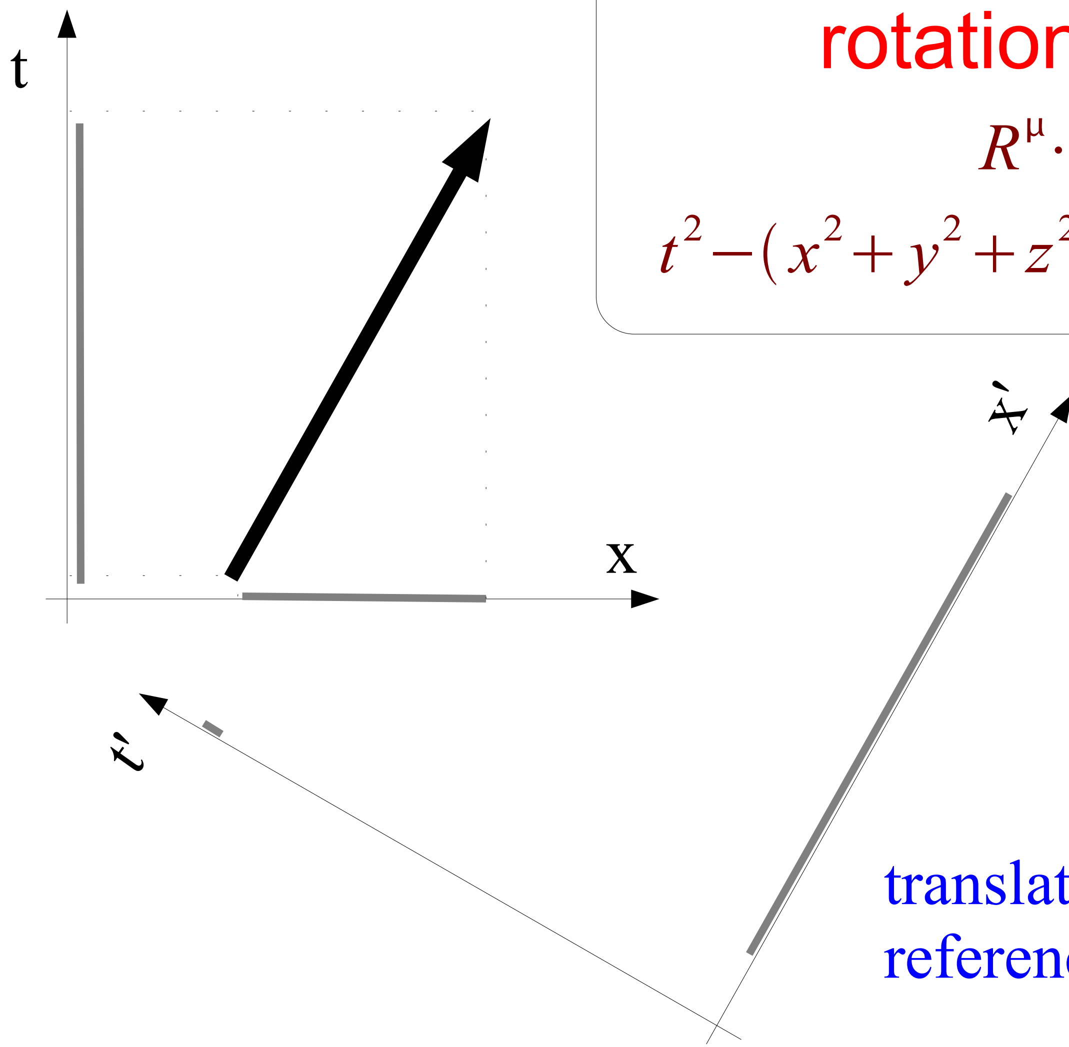
Four-Vectors

The Four-Vector magnitude is invariant under translations, rotations, and boosts.

$$R^{\mu} \cdot R^{\mu} = R'^{\mu} \cdot R'^{\mu}$$

$$t^2 - (x^2 + y^2 + z^2) = t'^2 - (x'^2 + y'^2 + z'^2)$$

units of $c = 1$



translated, rotated, & boosted reference frame

four-momentum

$$P^{\mu}(E, \vec{p})$$

<u>magnitude squared of the four-momentum</u>

natural units(c=1)

$$P^{\mu} \cdot P^{\mu} = P^{\mu} P_{\mu}$$

$$P^{\mu} P_{\mu} = E^2 - \vec{p} \cdot \vec{p} = E^2 - p^2$$

$$E^2 = P^{\mu} P_{\mu} + p^2$$

standard units

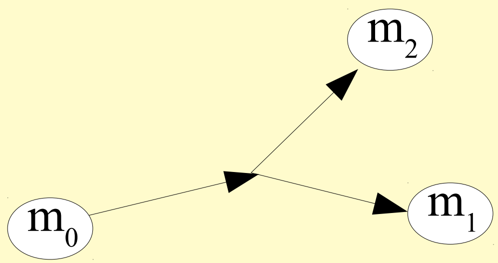
$$E^2 = (P^{\mu} P_{\mu}) c^4 + (pc)^2$$

$$E^2 = (m_o^2) c^4 + (pc)^2$$

Einstein's Equation

The magnitude is the invariant rest mass

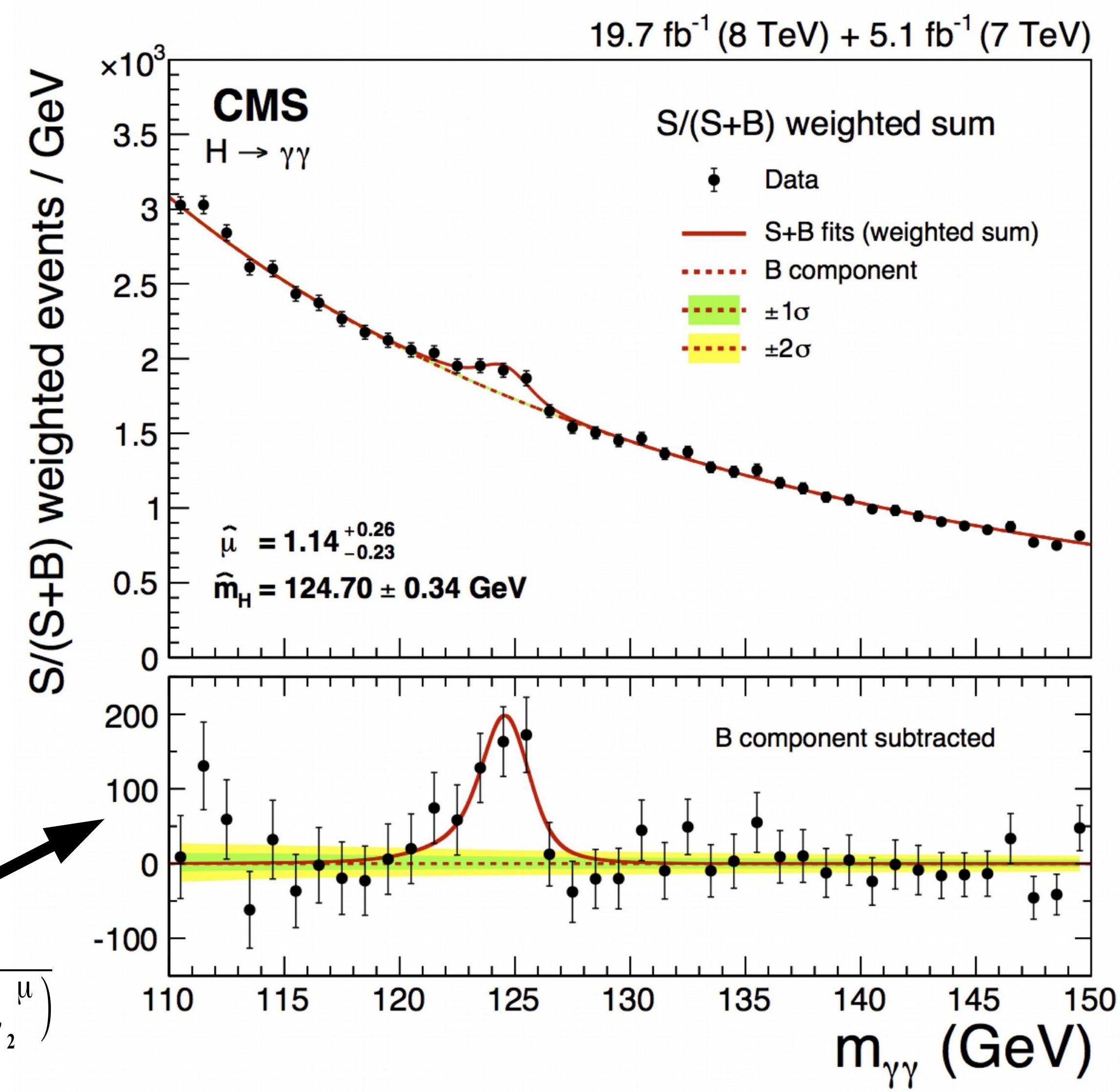
Discovery of the Higgs Boson



$$P_0^{\mu} = P_1^{\mu} + P_2^{\mu} = P_{12}^{\mu}$$

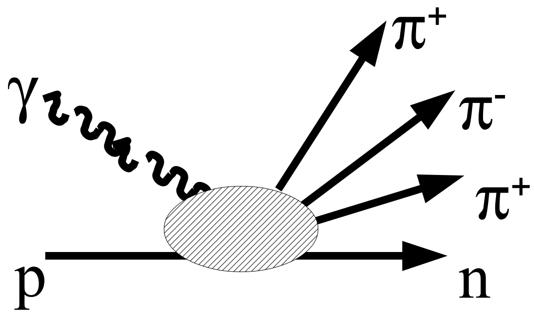
$$P_0^{\mu} \cdot P_0^{\mu} = m_0^{2} = P_{12}^{\mu} \cdot P_{12}^{\mu}$$

$$m_{\gamma\gamma} = \sqrt{P_{\gamma_1\gamma_2}^{\mu} \cdot P_{\gamma_1\gamma_2}^{\mu}}$$
$$= \sqrt{\left(P_{\gamma_1}^{\mu} + P_{\gamma_2}^{\mu}\right) \cdot \left(P_{\gamma_1}^{\mu} + P_{\gamma_2}^{\mu}\right)}$$



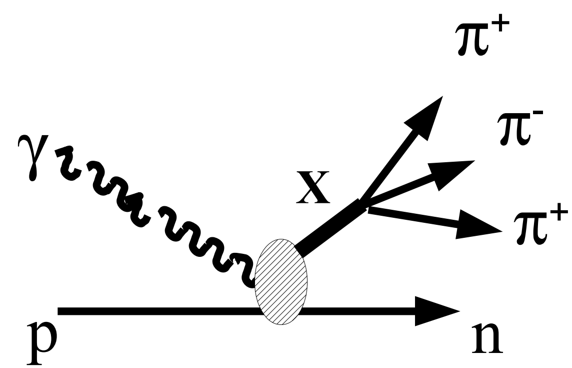
The Physics of Exercise 9

Identify the short lived particles



$\gamma\, p \rightarrow n\, \pi^{+}\ \pi^{+}\ \pi^{-}$

look for →



In the ~~Rest~~ Any Frame of **X**

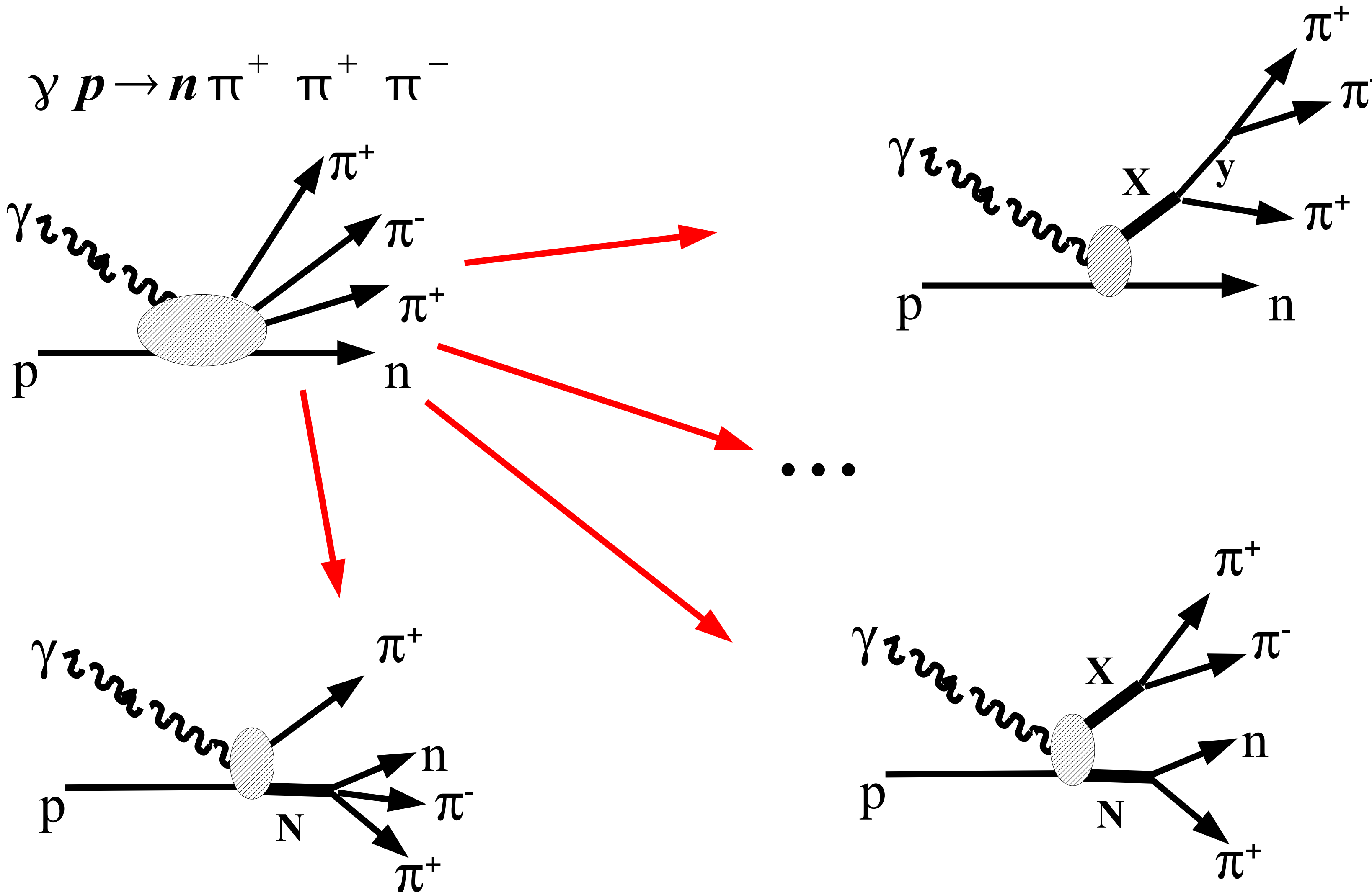
$$\left(P^{\mu}_{\pi^{+}} + P^{\mu}_{\pi^{+}} + P^{\mu}_{\pi^{-}}\right)^{2} = P^{\mu}_{X} \cdot P^{\mu}_{X} = m^{2}_{X}$$

Effective 3π mass

rest mass of X

++

$$\gamma\, p \rightarrow n\, \pi^{+}\ \pi^{+}\ \pi^{-}$$



Reading Ascii Data Files

Given text file

n3pi.dat

```
4
1  0    0.00    0.000   5.000   5.000
8  1    0.432   0.239   1.939   2.038
9 -1     -0.121  -0.192  1.679   1.700
8  1   -0.234   0.233   1.104   1.161
4
1  0    0.00    0.000   5.000   5.000
8  1    0.164   0.239   1.939   2.038
9 -1     -0.221  -0.192  1.679   1.700
8  1   -0.234   0.333   1.104   1.161
4
1  0    0.00    0.000   5.000   5.000
8  1    0.564   0.119   1.939   2.038
9 -1     -0.121  -0.192  1.679   1.700
8  1   -0.114   0.293   1.004   1.161
4
1  0    0.00    0.000   5.000   5.000
8  1    0.164   0.239   1.939   2.038
9 -1     -0.221  -0.192  1.679   1.700
8  1   -0.234   0.333   1.104   1.161
4
1  0    0.00    0.000   5.000   5.000
8  1    0.564   0.119   1.939   2.038
9 -1     -0.121  -0.192  1.679   1.700
8  1   -0.114   0.293   1.004   1.161
        + ...
```

beam

particle not directly observed

$$\gamma\, p \rightarrow n\, \pi^+\, \pi^+\, \pi^-$$

target at rest in lab

Particles observed after interaction

{# of recorded particles in event}

{id} {charge}   {p.x, p.y, p.z, E}

one interaction

```
4
1   0      0.00    0.000   5.000   5.000
8   1      0.564   0.239   1.939   2.038
9  -1    -0.121  -0.192   1.679   1.700
8   1     -0.234   0.233   1.104   1.161
```

Read Data from Input Files

```
 . . .
nEvents = 0
nPiPlus = 0

with open("n3pi.dat", "r") as file:
    # open file and read in ascii events line-by-line
    # the line will contain either the number of particles which
    # indicates start of an event or the line contains particle
    # information: id charge Px Py Pz E
    #
    for line in file:
        word = line.split()
        value = int(word[0])
        if value == 4:
            if nEvents < 2:
                print("\nNew event information")
            nPiPlus = 0
            nEvents += 1
        elif value == 1:
            print("\tPhoton beam with Energy:”, float(word[5]) )
        elif value == 8:
            # pi+ meson
            print("\tPi+[“, nPiPlus, “] with Energy:“, float(word[5]) )
            nPiPlus += 1
        elif value == 9:
            # pi- meson
            print("\tPi- with Energy:”, float(word[5]) )

# Done reading all events from data file
print("Total events read:", nEvents)
```

Read Data from Input Files

```
 . . .
nEvents = 0
nPiPlus = 0

with open("n3pi.dat", "r") as file:
    # open file and read in ascii events line-
    # the line will contain either the number
     # indicates start of an event or the line
     # information: id charge Px Py Pz E
     #
     for line in file:
         word = line.split()
         value = int(word[0])
         if value == 4:
             if nEvents < 2:
                 print("\nNew event information")
             nPiPlus = 0
             nEvents += 1
         elif value == 1:
             print("\tPhoton beam with Energy:”, float(word[5]) )
         elif value == 8:
             # pi+ meson
             print("\tPi+[“, nPiPlus, “] with Energy:”, float(word[5]) )
             nPiPlus += 1
         elif value == 9:
             # pi- meson
             print("\tPi- with Energy:”, float(word[5]) )

# Done reading all events from data file
print("Total events read:", nEvents)
```

n3pi.dat ascii file

4					
1	0	0.00	0.000	5.000	5.000
8	1	0.564	0.239	1.939	2.038
9	-1	-0.121	-0.192	1.679	1.700
8	1	-0.234	0.233	1.104	1.161

Using TLorentzVector

```
from ROOT import TLorentzVector
```

- **See TLorentzVector object description**
  - Declare
    - Beam, PiPlus = TLorentzVector(), 2*[TLorentzVector()]
    - Proton = TLorentzVector(0,0,0,0.938)
  - Set and Get Components
    - Beam.SetPz(5.0)
    - Beam.SetPxPyPzE(0, 0, 5.0, 5.0)
    - print("The beam energy[GeV] is", Beam.E())
  - Four-Vector Arithmetic
    - Neutron = Photon + Proton - (Pip[0] + Pip[1] + Pim)
    - print("neutron is mass", Neutron.Mag())

Four Vector Algebra

$$\gamma\, p \rightarrow n \pi^{+} \pi^{+} \pi^{-}$$

$$P_{\gamma}^{\mu} + P_{p}^{\mu} = P_{n}^{\mu} + P_{\pi^{+}}^{\mu} + P_{\pi^{+}}^{\mu} + P_{\pi^{-}}^{\mu}$$

Use energy-momentum conservation to find the missing neutron Four-Vector

Using a list for particle vectors

Don't do this

```
# photon + proton − (pi+1 + pi+2 + pi−)
#      Using a list to contain the four−vector elements

n =  [ p0[0] − ( p1[0] + p2[0] + p3[0] ]  # Px
n += [ p0[1] − ( p1[1] + p2[1] + p3[1] ]  # Py
n += [ p0[2] − ( p1[2] + p2[2] + p3[2] ]  # Pz
n += [ p0[3] + 0.938 − (p1[3]  + p2[3] + p3[3]) # E
NeutronMass = sqrt( n[3]**2 − ( n[0]**2 + n[1]**2 + n[2]**2 )
```

Using ROOT TLorentzVectors

A better way

```
Proton = ROOT.TLorentzVector(0, 0, 0, 0.938)      # proton, target particle at
rest
# neutron is obtained via energy−momentum conservation
Neutron = Photon + Proton − ( PiPlus[0] + PiPlus[1] + PiMinus )
neutronMass = neutron.Mag()

# or one can directly uses
neutronMass = ( Photon + Proton − (PiPlus[0] + PiPlus[1] + PiMinus) ).Mag()
```

Input data to TLorentzVector

```
import std
Beam = ROOT.TLorentzVector()
 . . .
nEvents, nPiPlus = 0, 0

with open("n3pi.dat", "r") as dataFile:
   # open file and read in ascii data line-by-line
   #
    for line in dataFile:
    word = line.split()  # split line into list of words
    value = int(word[0])

    if value == 4:                # initialize for new event particles
        if nEvents % 10000 == 0:
            print(".", end='')         # print "." to screen every 10k
events
            std.stdout.flush()
           nPiPlus = 0
           nEvents += 1
       elif value == 1 :  # We have the beam
       Beam.SetPxPyPzE( float(word[2]), float(word[3]), float(word[4]),
float(word[5]) )
       elif value == 8 : # We have a pi+
           PiPlus[ nPiPlus ].SetPxPyPzE( float(word[2]), float(word[3]),
float(word[4]), float(word[5]) )
           nPiPlus += 1
       elif value == 9 :  # We have the pi-
       PiMinus.SetPxPyPzE( float(word[2]), float(word[3]), float(word[4]),
float(word[5]) )
        . . .
```

Calculating invariant mass

```
# Adding 4-vectors
S = n +  pip[0] + pip[1] + pim
X = pip[0] + pip[1] + pim


# Calculate InvariantMass[n pi+ pi+ pi-]
mass = S.Mag()
# or
mass = ( n +  pip[0] + pip[1] + pim ).Mag()

# Calculate InvariantMass[pi+ pi+ pi-]
mass3pi = X.Mag()
# or
mass3pi = (pip[0] + pip[1] + pim).Mag()

# Calculate 2 pion Invariant Mass
m2pi = ( pip[0] + pim ).Mag()
```

Histogramming Invariant Mass

```
from ROOT import TH1F
    ## create 1D histograms for invariant mass distribution
    #
    H3pi      = TH1F("h3pi", "Mass(3pi)", 150, 0.8, 2.3)
    H3picut   = TH1F("h3picut", "Mass(3pi)", 150, 0.8, 2.3)
    Hpip1pim  = TH1F("hpip1pim", "Mass(pip1 pim)", 180, 0.0, 1.8)
    Hpip2n    = TH1F("hpip2n", "Mass(pip2 n)", 180, 0.0, 1.8)
     ...
```

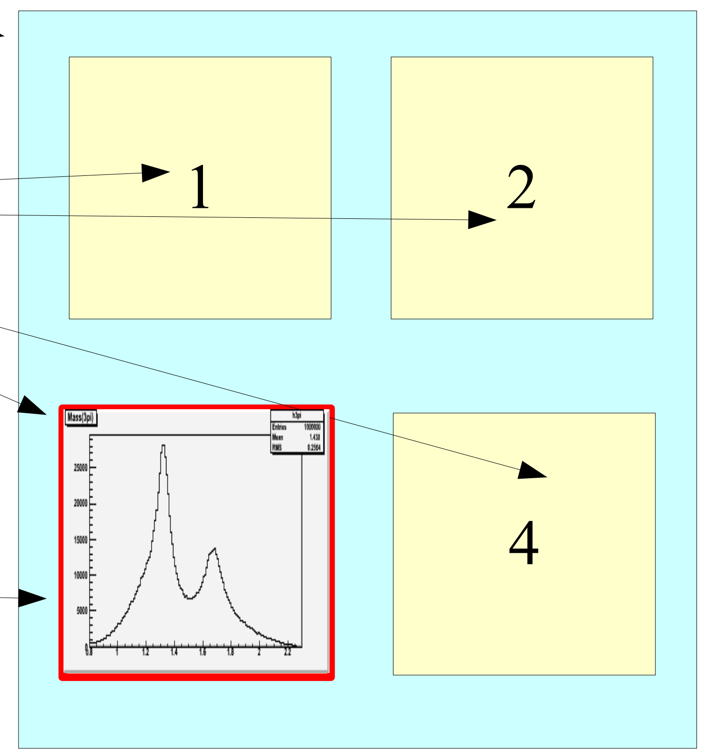
object name — internal name — title — # bins & bin value range

```
    # Fill histograms with values
    #
    H3pi.Fill( (PiPlus[0] + PiPlus[1] + PiMinus).Mag() )

    if (PiPlus[0] + PiMinus).Mag() < 1.0 :
        # 3pi invariant mass for Mass( pip pim) < 1.0 GeV
        H3picut.Fill( (PiPlus[0] + PiPlus[1] + PiMinus).Mag() )

    # 2Pi and nPi invariant mass
    P, Q = PiPlus[0]+ PiMinus, PiPlus[1] + Neutron
    Hpip1pim.Fill( P.Mag() )
    Hpip2n.Fill( Q.Mag() )
      . . .
```

The Canvas

from ROOT import TCanvas

- Creating
  - Canvas = TCanvas("cc","gamma p -> pi pi p",10,10,800,600)
- Zoning
  - Canvas.Divide(2,2)
- Navigating
  - Canvas.cd(3) # change to pad 3
- Drawing
  - H3pi.Draw()



Alternative Approach

Rather than filling histograms as one goes through the data, an alternative approach is to store useful information from the data in a Ntuple or Data Tree

Storing information in a Ntuple

from ROOT import TNtuple

```
# Declaration of an Ntuple for storing calculated values
Ntuple = TNtuple("ntuple", "3pi ntuple", "mn3pi:m3pi:m2pi1")
```

Names are user defines

object name

internal name

title

Data table vector names

```
#
# Fill ntuple with values
#
Ntuple.Fill( S.Mag(), P.Mag(), m2pi )
```

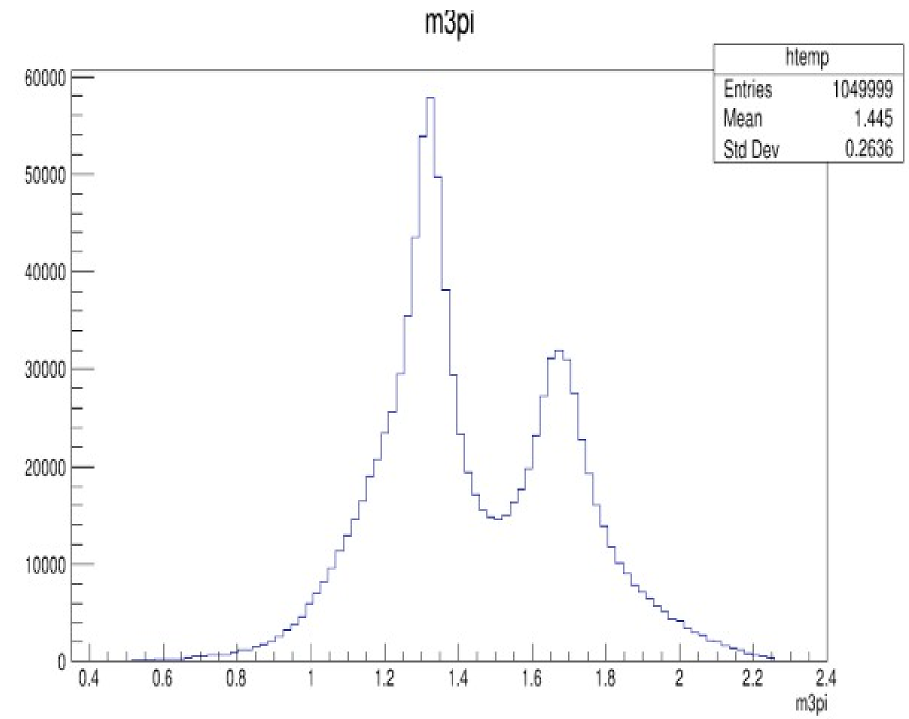
mn3pi	m3pi	m2pi1
<value-evnt1>	<value-evnt1>	<value-evnt1>
<value-evnt2>	<value-evnt2>	<value-evnt2>

User defined labels

mn3pi: Mass of neutron 3 pions
m3pi: Mass of 3 pions
m2pi1: Mass 2 pion (1^{st} combination)

Projecting Histograms from TNtuples

```
#
# Drawing histograms
# from TNtuples
#
Ntuple.Draw("m3pi")
```



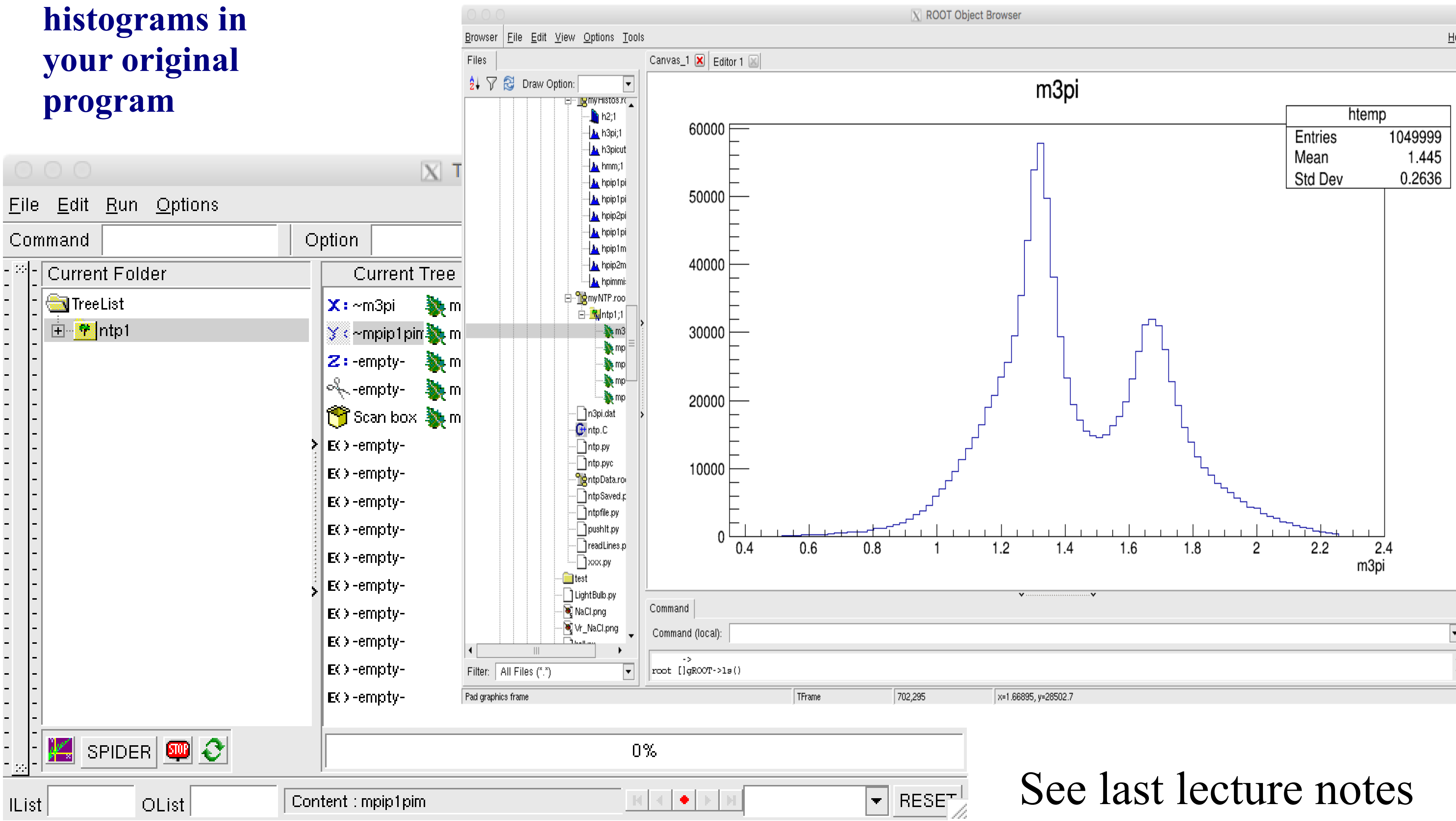
NTuples can be saved in a TFile and analyzed by another program or via ROOT interactively

```
# projection based on
# a selection cut
Ntuple.Draw("m3pi", "m2pi1<0.8")

# projection to a 2D histogram: Mass(3pi) vs Mass(2pi)
Ntuple.Draw("m3pi:m2pi1")
```

Interactively exploring the data via TBrowser

Use TFile to store the ROOT TNtuple or TH1F histograms in your original program

Open “A TBrowser in Python Interactively ”



See last lecture notes