Computational Physics Lab

Analysis of Large Data Sets: NTuples

03/31/2009

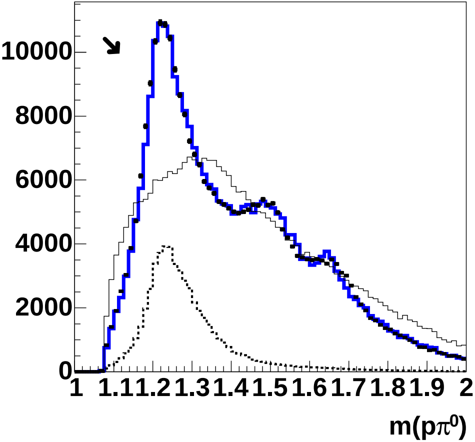
Outline

1. Mass Spectra
   - The Experiment

2. This Week's Project
   - The Physics Behind
   - Units
   - Reading Ascii Data Files

Photoproduction of Mesons

$$\gamma p \rightarrow X \rightarrow \Delta^+\pi^0 \rightarrow p\pi^0\pi^0 \rightarrow p\gamma\gamma\gamma\gamma$$



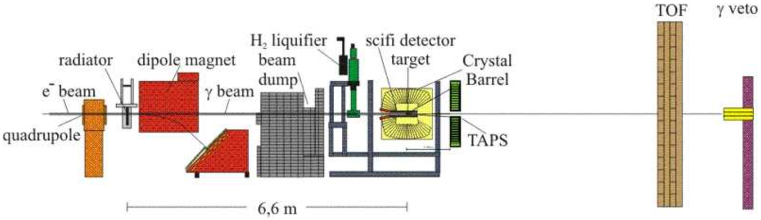
Example:

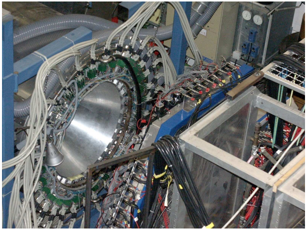
We want to study

$\gamma p \rightarrow N^* \rightarrow p\pi^0\pi^0$

- Measure p, π^0's
- Use inv. mass: $m^2 = E^2 - p^2$

Experimental Setup





Tagged Photons (E_{e^-} = 3.2 GeV):

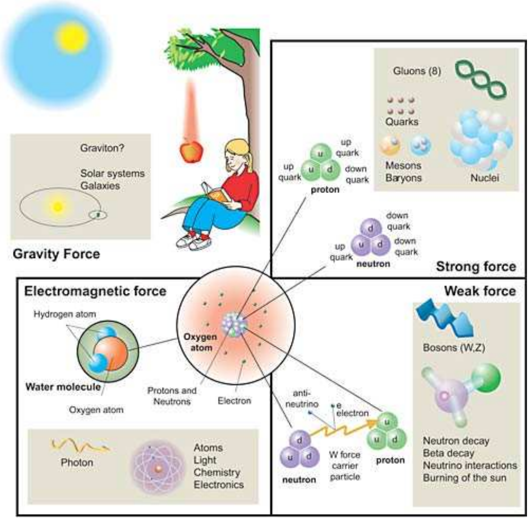
- $0.25 \cdot E_{e^-} \leq E_\gamma \leq 0.95 \cdot E_{e^-}$

➜ 800 MeV $\leq E_\gamma \leq$ 3000 MeV

Outline

1. Mass Spectra
   - The Experiment
2. This Week's Project
   - The Physics Behind
   - Units
   - Reading Ascii Data Files

Photoproduction of Mesons



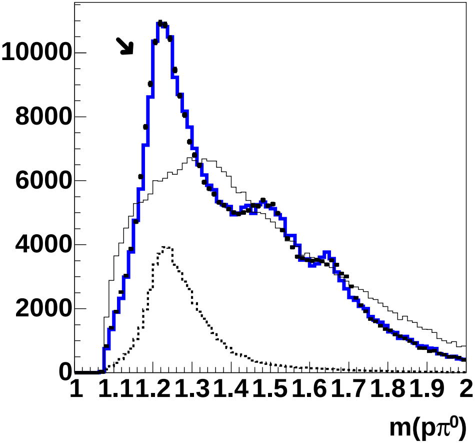
Example:

We want to study

$\gamma p \rightarrow N^* \rightarrow p\pi^0\pi^0$

Photoproduction of Mesons

$\gamma p \rightarrow X \rightarrow \Delta^+\pi^0 \rightarrow p\pi^0\pi^0$



Example:

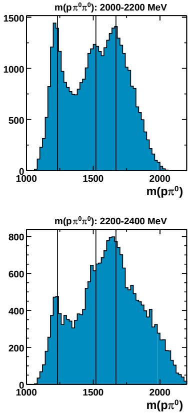
We want to study

$\gamma p \rightarrow N^* \rightarrow p\pi^0\pi^0$

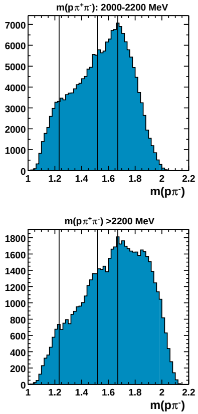
- Measure p, π^0's
- Use inv. mass: $m^2 = E^2 - p^2$

$\gamma p \rightarrow p\pi^0\pi^0$ and $\gamma p \rightarrow p\pi^+\pi^-$ from CB-ELSA and CLAS

- CB-ELSA:



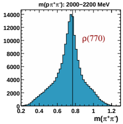
- CLAS:



$$\gamma p \rightarrow N^*/\Delta^* \rightarrow X\pi$$

$$X = \Delta(1232)$$
$$X = D_{13}(1520)$$
$$X = X(1660)$$

$\Rightarrow$ **Similar resonance structures in both data sets !**



$$\gamma p \rightarrow p\rho$$

Units

The electronvolt (symbol eV) is a unit of energy. It is the amount of energy equivalent to that gained by a single unbound electron when it is accelerated through an electrostatic potential difference of one volt, in vacuum. In other words, it is equal to one volt (1 volt = 1 joule per coulomb) multiplied by the (unsigned) charge of a single electron.

One electronvolt is a very small amount of energy:

1 eV = 1.602 176 53(14) $\cdot 10^{-19}$ J (or approximately 0.160 aJ)

The proton has a mass of 0.938 GeV/c^2, making a GeV/c^2 a very convenient unit of mass for particle physics:

- 1 eV/c^2 = 1.783 $\cdot 10^{-36}$ kg
- 1 GeV/c^2 = 1.783 $\cdot 10^{-27}$ kg

Units

In a recorded lecture from 1961 Richard Feynman apologized to his students for this failure by atomic physicists to use the appropriate SI unit (which would be the attojoule):

> A single atom is such a small thing that to talk about its energy in joules would be inconvenient. But instead of taking a definite unit in the same system, like 10^{-20} J, [physicists] have unfortunately chosen, arbitrarily, a funny unit called an electronvolt (eV) ... I am sorry that we do that, but that's the way it is for the physicists.

Opening Input Files

$$\gamma \mathrm{p} \longrightarrow \mathrm{N}^* \longrightarrow \mathrm{p}\pi^0\pi^0$$

```
 21400.000        49.000   1900.610      2.000
   282.366       -59.729    294.642   1024.923
    -7.397      -243.317   1163.580   1196.409
  -274.982       303.030    442.388    617.550

 21400.000      1810.000    936.093      0.000
  -115.682       120.330    762.354   1220.410
  -215.029       -81.591    112.839    289.561
   330.688       -38.740     60.900    364.394

 21400.000      1812.000   1270.843      0.000
   309.611        24.443    885.796   1327.194
  -392.629       220.538    354.049    588.527
    83.016      -244.967     30.999    293.394
       ...
```

run number	event number	E_γ	(float)
proton P_x	P_y	P_z	E
pion[1] P_x	P_y	P_z	E
pion[2] P_x	P_y	P_z	E

File Streams

# include <iostream.h>

# include <fstream.h>

ifstream fin (“input.dat”);

ofstream fout (“output.dat”);

main() {

    float P_x, P_y, P_z, E

    ...

Some kind of loop

        fin $\gg P_x \gg P_y \gg P_z \gg E;$

        fout $\ll P_x \ll P_y \ll P_z \ll E;$

        ...

}

1. The disc file input.dat in the directory from which the program is beeing run is associated with a stream called fin.
2. In a similar way, the disc file output.dat is associated with a stream called fout.

Using Class TLorentzVector

# include <TLorentzVector.h>

- Declare
  - TLorentzVector photon, pip[2], pim, neutron;
  - TLorentzVector proton(0,0,0,0.938);
  - TLorentzVector *vec4;
- Set Components
  - vec4->SetPx(...);
  - vec4->SetPy(...);
  - vec4->SetE(...);
- 4-Vec Arithmetic
  - proton^f = photon + proton^i $-$ piz[0] $-$ piz[1];
  - mass of proton = proton.M() or proton.Mag();