FPP Analysis with the C++ Analyzer

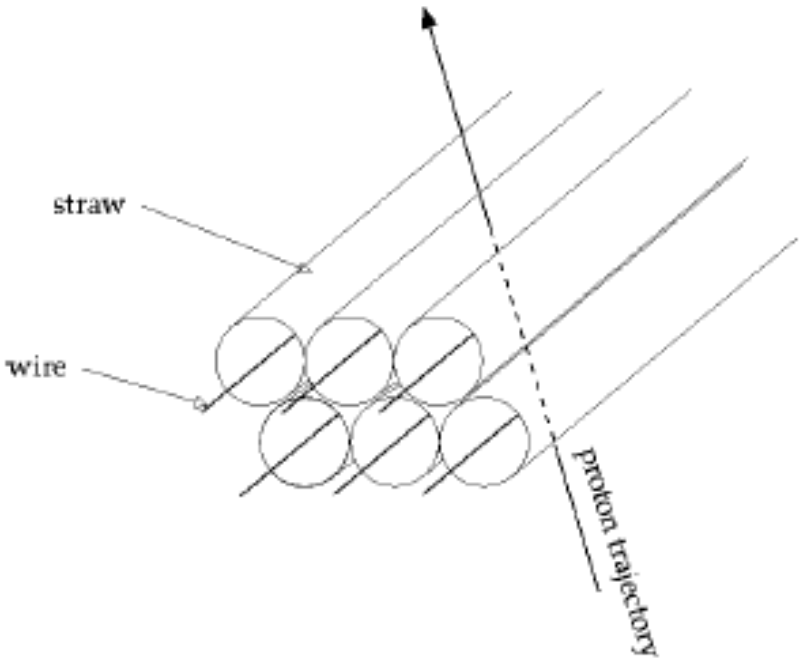
Jackie Glister

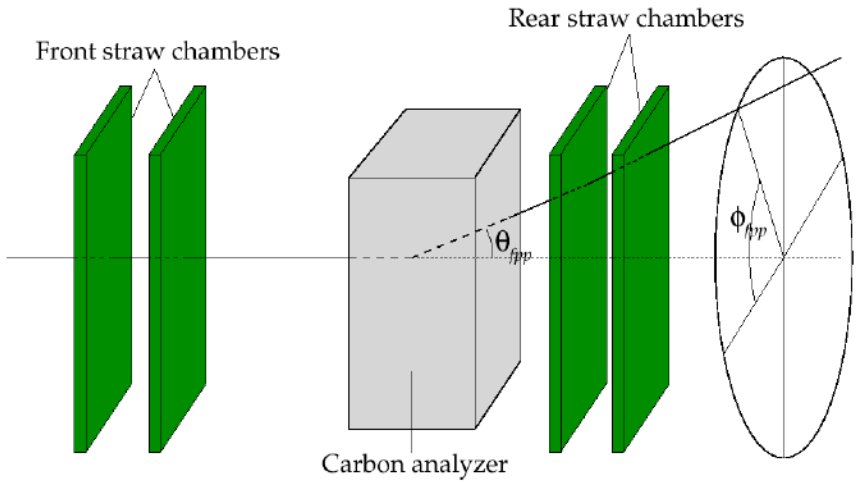
Dalhousie University & Saint Mary's University

Hall A Analysis Workshop
January 6, 2007

FPP - Focal Plane Polarimeter

- 4 chambers with 6 layers of straws each
- Wires are multiplexed into groups of 8
- Width of pulse used to determine the straw number





FPP library - libFpp.so

- Fpp library created Summer 2006 for LEDEX
- Converted from old FORTRAN ESPACE code
- Library is made up of three classes:
  THaFpp
  THaFppPlane
  THaFppHit

THaFppHit

For each hit in the fpp, an instance of THaFppHit is created with the following info:

- wiregroup number
- leading and trailing edge of tdc pulse

Then the following quantities are calculated and saved for each hit:

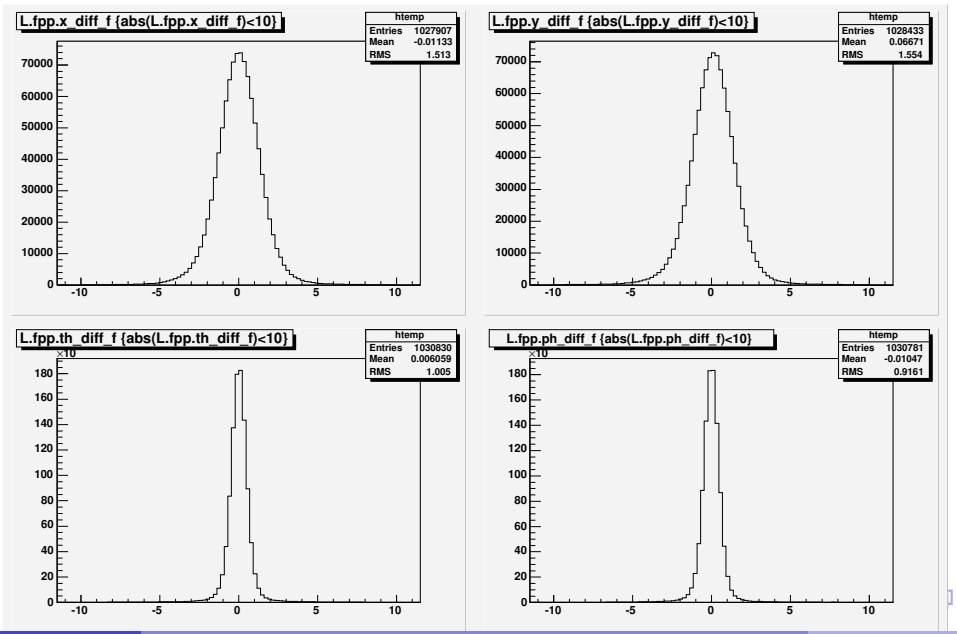
- width of tdc pulse
- straw number
- drift distance to wire

THaFppPlane

- Total of 24 planes
- Planes are labeled u1-11, v1-11, u and v are at 45^o to horizontal and are 90^o to one another
- There are also two planes labeled x1-2, which are vertical, but are not currently used for tracking
- Each plane has specific database entries such as wire spacing and geometric location, which are used to determine hit location in (x,y) space
- Efficiency of the planes may be plotted (L.fpp.u1.efficiency): 1 if there is a hit, 0 if there isn't & 2 if there is no hit predicted

THaFpp

- Contains all tracking routines
- Several of the old ESPACE routines are loaded in this class
- Front and rear tracks used to calculate θ and ϕ of scatter in carbon analyzer (L.fpp.th_az and L.fpp.ph_az)
- In order to evaluate alignment, differences in x, y, θ & ϕ calculated from vdc and front/rear fpp tracks are evaluated at z = 0 & 350 cm (L.fpp.<var>_diff_f/r, <var> = x, y, th or ph)



Loading the libFpp.so Library and Adding the FPP

- Some of the tracking routines were left in FORTRAN and loaded in, so the g2c library is needed for some definitions
- The following lines should be at the beginning of the analyzer script (> analyzer script.C)

```
gSystem->Load("/usr/lib/gcc-lib/i386-redhat-linux/3.2.3/libg2c.so");
gSystem->Load("../libFpp");
```

- Once the left arm spectrometer is defined, the FPP may be added to it

```
THaApparatus* HRSL = new THaHRS("L","Left arm HRS");
gHaApps->Add( HRSL );
HRSL->AddDetector(new THaFpp("fpp","Focal Plane Polarimeter"));
```

The Database

- Two files are required: 'detmap.config' (wiregroup detector map) & 'db_L.fpp.dat' (fpp database) and must be located where you run the program
- 'db_L.fpp.dat' is time stamped
  (*i.e.* − − − − − − − −[2000-01-21 20:17:41])
  so that the proper info is read based on the time stamp in the raw data file
- Format of database is L.fpp. <variable>, where <variable> includes options, geometry, demultiplexing limits, time offsets, drift velocity coefficients and alignment coefficients

```
L.fpp. alignment 0
L.fpp. geometry 0.  0.  1.915899992  0.  0.  0.  0.  0. 24
L.fpp. DemuxCuts
   16.88  27.50  39.00  50.00  59.00  70.00  82.50  94.00 107.50
   ....
```

The Database - Cont.

- See webpage by Mark Jones: http://www.jlab.org/ jones/e93027/software.html for a description of all options in database, but note
  icalib $\leftrightarrow$ calib_opt
  ialign $\leftrightarrow$ align_opt
  ana_fpp_front/rear $\leftrightarrow$ ana_front/rear
  pulser_fpp $\leftrightarrow$ demux_stats
  evdsp_fpp $\leftrightarrow$ evdisplay
- Also, http://www.jlab.org/ jones/fpp_root/FPP_software.html for scripts to calculate demux cuts, time offsets and drift velocity coefficients

Defining the Variables

A output definition file must be loaded in the script file to plot variables

```
analyzer->SetOdefFile("output.def");
```

Variables are either specific to a plane, L.fpp.plane.variable
(straw, wiregroup, efficiency, *etc.*)

```
variable L.fpp.u1.straw
variable L.fpp.v11.wiregroup
cut C1 L.fpp.u1.ngoodhit == 1
```

or belong to the FPP as a whole, L.fpp.variable

```
variable L.fpp.th_az
variable L.fpp.ph_az
cut C2 L.fpp.conetest==1
```

Defining the Variables - Cont.

If you want *all* the FPP variables or, say, all the u1 plane variables put the following into the output.def file:

```
block L.fpp.*
block L.fpp.u1.*
```

You can also create histograms with user-defined binning:

```
TH1F u1w 'u1 wiregroup hits' L.fpp.u1.wire 35 0 35 C3
```

How to Plot Variables

You may either plot the histograms, with predefined cuts:

```
u1w->Draw();
```

Plotting variables from the Tree (defined with 'variable' in output.def):

```
T->Draw("L.fpp.u1.straw");
T->Draw("L.fpp.th_az","L.fpp.th_az < 1000.");
```

Note that any variable that is general to the fpp (*i.e.* th_az) is set to a very large number (1e37) if tracking fails, so you should use a < 1000. cut to see a reasonable plot

How to Plot Variables - Cont.

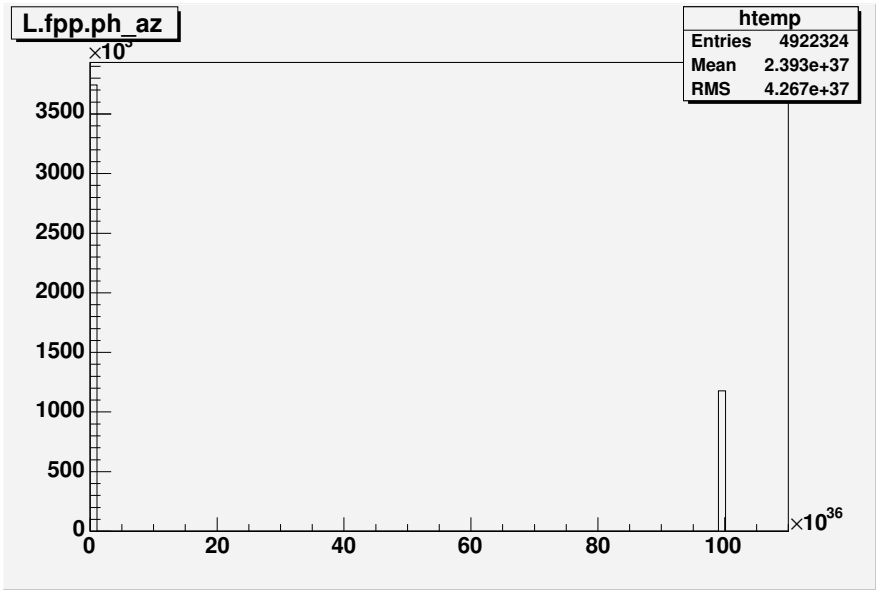


Figure: T->Draw("L.fpp.ph_az");

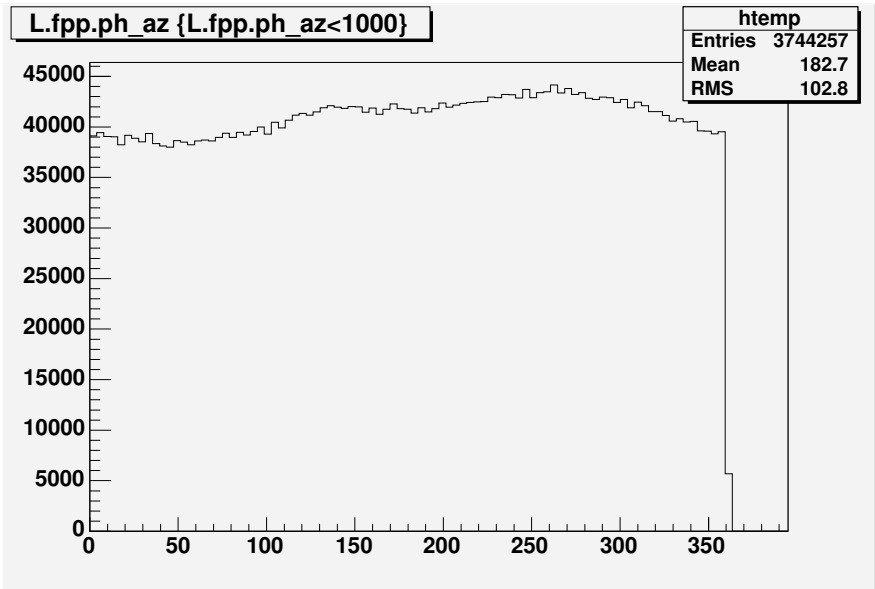
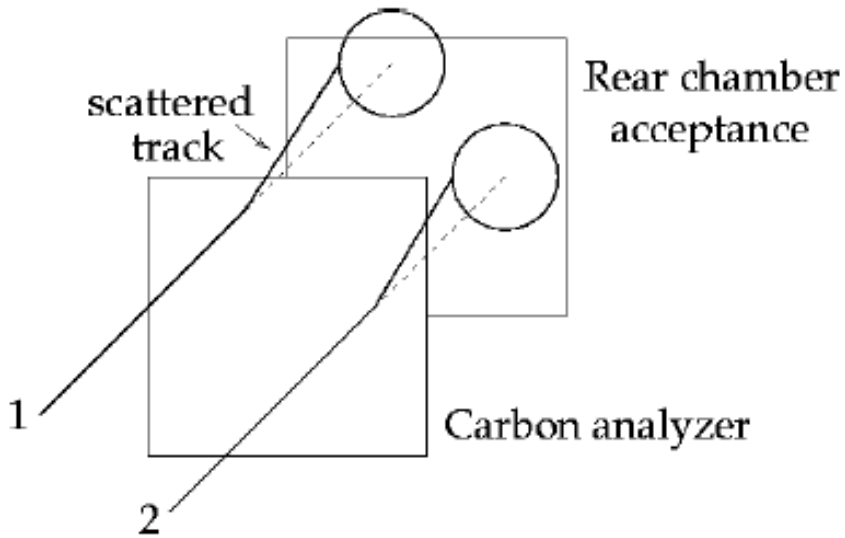


Figure: T->Draw("L.fpp.ph_az", "L.fpp.ph_az < 1000.");

Conetest and False Asymmetries

- Two conetest methods to help remove false asymmetries: older and new one from Ed Brash
- Old conetest (apply cut L.fpp.conetest==1) removes hits for which x, y $\pm$ radius given by θ of scatter is not within physical chamber area
- New conetest (apply cut L.fpp.coneteste==1) replaces physical chamber area with polygon defined by responsive chamber area and calculates 8 points on circumference around hit which must lie inside this polygon



Variables needed for Polarization Analysis

The following variables must be added to the tree in order to extract polarization observables using PALM (Polarization Analysis using Maximum Likelihood Method):

```
#Fpp variables
variable L.fpp.zclose
variable L.fpp.th_az
variable L.fpp.ph_az
variable L.fpp.conetest

#target variables (for cuts)
variable L.tr.tg_dp
variable L.tr.tg_th
variable L.tr.tg_ph
variable L.tr.tg_y

#beam helicity
variable Beam.HL.helicity
```

Some Extra Info

- A List of all variables can be found at: /u/home/jglister/GradWork/ana/FppLib/variables.txt
- The Fpp Library and the database files (db_L.fpp.dat & detmap.config) can be found at /work/halla/ledex/adaq/analyzer/Fpp
- Example script file is /work/halla/ledex/adaq/analyzer/replay-split-gd.C which loads an example output definition file /work/halla/ledex/adaq/analyzer/def_files/ replay_output_gd.def