LHRS Analysis for d_2^n

β Source Code, Positive Polarity Data Quality

D. Flay

Hadronic & Nuclear Physics Group
Temple University Physics Department

9/16/10

Outline

LHRS β (1)

How is β Calculated?

- From the `THaHRS` class:

$$t = t_0 + \ell/\beta c$$

  - $\ell = \ell' - \ell_{\text{ref}} =$ the event's pathlength (to S2m) $-$ the pathlength to the reference plane (S1)
  - $t_0 =$ time of the track at S1
  - $t =$ (corrected) time at paddle (in S2m) (?)
  - 't_0 and β are solved for'

- The calculation is a bit more complicated than this...

LHRS β (2)

Source Code

```
track = static_cast<THaTrack*>(Tracks->At(i));
THaTrackProj* tr_ref = static_cast<THaTrackProj*>
  (sc_ref->GetTrackHits()->At(i));

Double_t pathlref = tr_ref->GetPathLen();

Double_t wgt_sum=0.,wx2=0.,wx=0.,wxy=0.,wy=0.;
Int_t ncnt=0;

TIter nextSc( fNonTrackingDetectors );
THaNonTrackingDetector *det;
while ( ( det = static_cast<THaNonTrackingDetector*>(nextSc()) ) ) {
  THaScintillator *sc = dynamic_cast<THaScintillator*>(det);
  if ( !sc ) continue;

  const THaTrackProj *trh = static_cast<THaTrackProj*>(sc->GetTrackHits()->At(i));

  Int_t pad = trh->GetChannel();
  if (pad<0) continue;
  Double_t pathl = (trh->GetPathLen()-pathlref);
  Double_t time = (sc->GetTimes())[pad];
  Double_t wgt = (sc->GetTuncer())[pad];

  if (pathl>.5*kBig || time>.5*kBig) continue;
  if (wgt>0) wgt = 1./(wgt*wgt);
  else continue;
```

LHRS β (3)

Source Code

```
 wgt_sum += wgt;
 wx2  += wgt*pathl*pathl;
 wx   += wgt*pathl;
 wxy  += wgt*pathl*time;
 wy   += wgt*time;
 ncnt++;
}
 Double_t beta = kBig;
 Double_t dbeta = kBig;
 Double_t time = kBig;
 Double_t dt = kBig;

 Double_t delta = wgt_sum*wx2-wx*wx;

 if (delta != 0.) {
  time = (wx2*wy-wx*wxy)/delta;
  dt = TMath::Sqrt(wx2/delta);
  Double_t invbeta = (wgt_sum*wxy-wx*wy)/delta;
  ...
 }
```

- wgt $= dt$ (uncertainty in corrected time for paddle)
- To make this more readable, define:
  - wgt_sum $= \sum_i \frac{1}{dt_i^2}$ $[1/\mathrm{s}^2]$
  - wx2 $= \sum_i \frac{\ell_i^2}{dt_i^2}$ $[\mathrm{m}^2/\mathrm{s}^2]$
  - wx $= \sum_i \frac{\ell_i}{dt_i^2}$ $[\mathrm{m}/\mathrm{s}^2]$
  - wy $= \sum_i \frac{t_i}{dt_i^2}$ $[1/\mathrm{s}]$
  - wxy $= \sum_i \frac{\ell_i t_i}{dt_i^2}$ $[\mathrm{m}/\mathrm{s}]$

LHRS β (4)

Source Code

- What is t_i (`time`)?:
  - `time = fTime = 0.5*(fLT_c + fRT_c) - fSize[1]/fCn`
  - `fCn` = the speed of light in the material
  - What is `fSize[1]`?
    - Seems to be the paddle size — but not sure — is it correct?
    - Why is this quantity being subtracted? A correction for the paddle thickness?
- What is `wgt`?
  - `wgt = fdTime[i] = fResolution/sqrt2`
    - Note: no index `i` on `fResolution`

LHRS β (5)

Source Code

- Using the definitions shown on the previous slides:

$$\frac{1}{\beta} = \frac{\left(\sum \frac{1}{dt^2}\right)\left(\sum \frac{\ell t}{dt^2}\right) - \left(\sum \frac{\ell}{dt^2}\right)\left(\sum \frac{t}{dt^2}\right)}{\left(\sum \frac{1}{dt^2}\right)\left(\sum \frac{\ell^2}{dt^2}\right) - \left(\sum \frac{\ell}{dt^2}\right)\left(\sum \frac{\ell}{dt^2}\right)} [\text{s/m}]$$

- When the factor c is incorporated, β becomes dimensionless:

```
beta = 1./(c*invbeta);
dbeta = TMath::Sqrt(wgt_sum/delta)/(c*invbeta*invbeta);
```

- This looks like the statistical average of a quantity:

$$\langle x \rangle = \frac{\sum_i x_i e^{-f(x_i)}}{\sum_i e^{-f(x_i)}}$$

LHRS β (6)

Source Code

- From a mathematical standpoint, things look OK in the *calculation* of β
- However, the way the time average for a paddle in S2m is obtained is strange...
  - What to make of `fSize[1]`?
    - This could create an *offeset* if anything, but doesn't seem likely to be the issue with the jitter

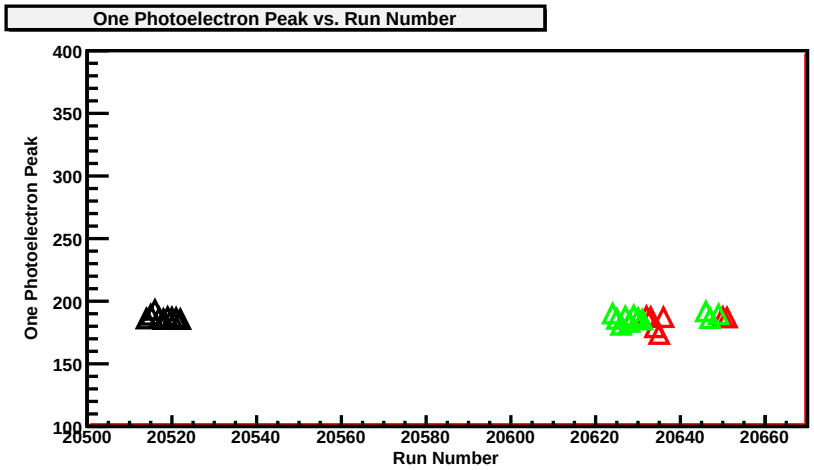
LHRS Data Quality (1)

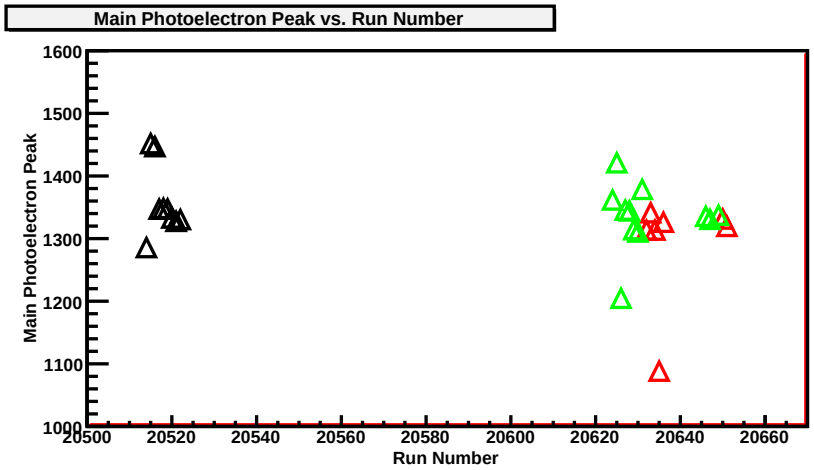
Positive Polarity Data: $p = 0.60, 0.80$ GeV

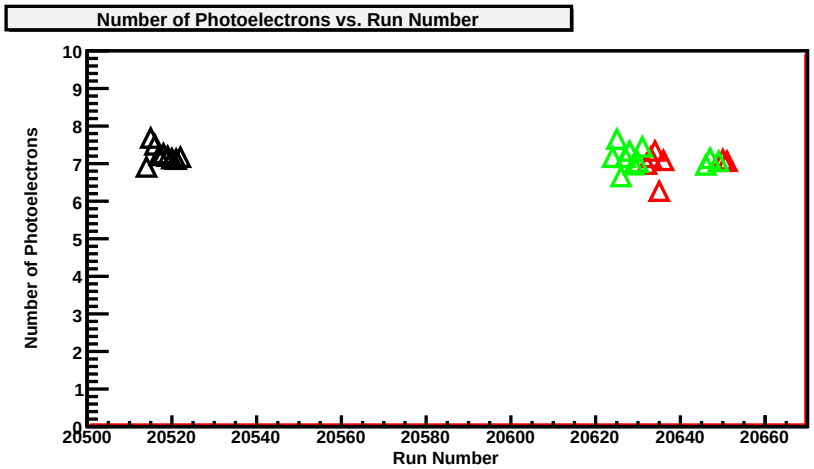
- LHRS momentum setting: $p = 0.60$ GeV
  - Runs examined:
    - 20512–20522 (4-pass)
    - 20632–20636 (5-pass)
      20560, 20561 (5-pass)
- LHRS momentum setting: $p = 0.80$ GeV
  - Runs examined:
    - 20624–20631 (5-pass)
      20646–20649 (5-pass)
- Will add more momentum bins as we go. . .

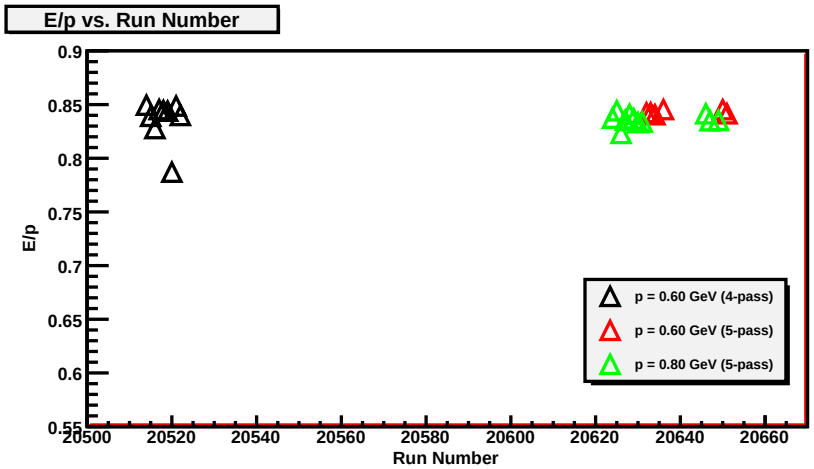
LHRS Data Quality (2)

Positive Polarity Data: $p = 0.60, 0.80$ GeV



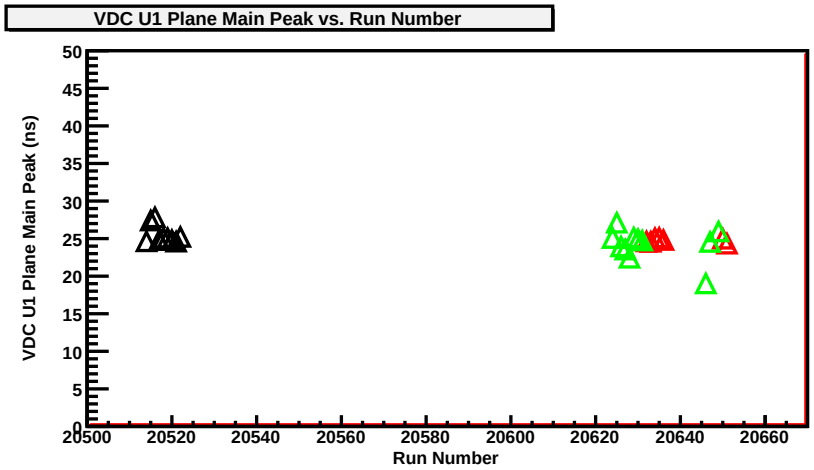


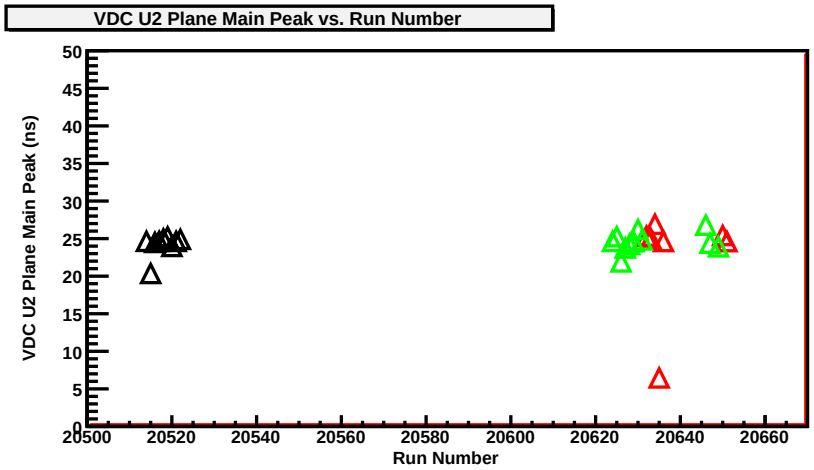


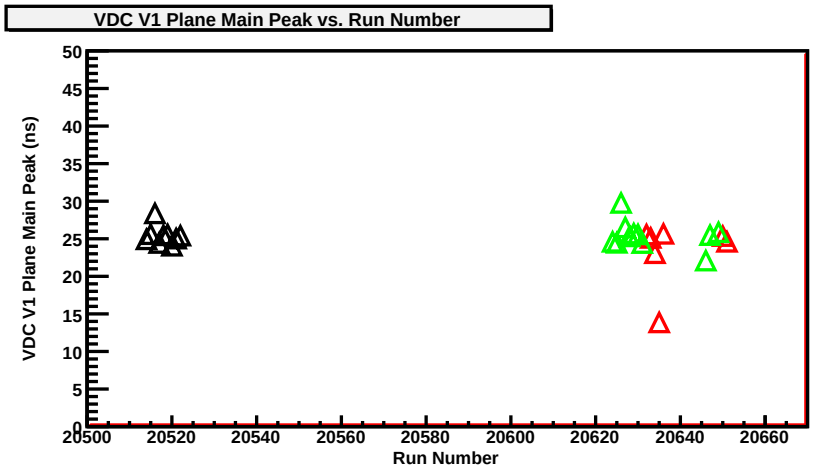


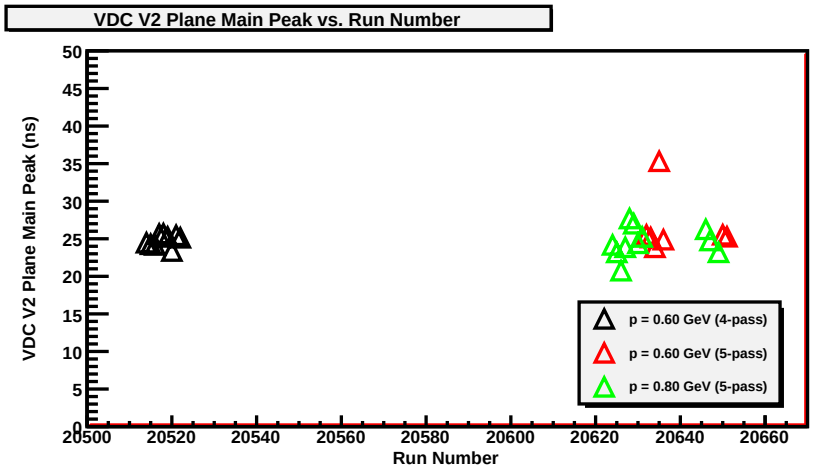
LHRS Data Quality (3)

Positive Polarity Data: $p = 0.60, 0.80$ GeV









Summary

- LHRS β
  - The calculation looks OK in the source code
  - Need to figure out the significance of `fSize[1]`
- Positive Polarity Data Quality
  - Doesn't look too bad — we do have some outliers that should be looked into
    - Maybe just low statistics? There are a few runs that have $\sim$ 20000–30000 events...

What's Next?

- LHRS β
  - Look into `fSize[1]`
- Data Quality
  - Implement scintillator checks
    - S1, S2m TDC times — are these effectively complete?
  - Beam trip checks
  - Positive Polarity
    - Look into the outliers seen in the plots
    - Add on more momentum bins
  - Negative Polarity
    - Double-check run list to make sure it agrees with the positive polarity run set (thanks to StartType.pl changes this week)
- Acceptance
  - Huan's code was translated from Vince's: fortran to C++
  - Got it working: I was missing the library 'f2c' and I didn't have gfortran installed, among other fortran packages
  - Start diving in shortly...