MarlinTPC LikelihoodFitting

Jason Abernathy

HitFinding/TrackSeeder

- HitFinderTopo
  - **Finds hits and combines them into tracks**
  - **Martin gave a good description of how it works**
- TrackSeeder
  - **Calculates the track parameters of the track candidates**
  - **Does an analytic least-squares fit to the hit positions**
  - **Assumes that “almost-straight” tracks are straight**

Track Fitting

- 3 in trunk:
  - **SimpleChiSquared**
  - **SimpleMinimizer**
  - **LikelihoodFitting**
- Likelihood fitter
  - **Works... sometimes?**
    - Rectangular pad layouts work
    - Circular ones don't work unless the tolerance is very high
    - Very sensitive to the drift diffusion used
  - **Very big, needs to be broken up based on**
    - Layout type, whether b-field corrections are used
    - Want it to find the photoelectric calibration data too

Likelihood Overview

- Developed by D. Karlen at Uvic
  - **He implemented it in Java**
- Likelihood(track parameters | data) ==
  Pr(data | track parameters)
- So...
  - **Calculate the probability of observing the distribution of charge given the hypothetical track parameters**
  - **Use Minuit to minimize the neg, log. likelihood**

The Charge Distribution Model

- (Homogeneous case first)
- The primary track is nicely projected onto the readout electronics
- Charge is distributed by convolving an isometric gaussian with the track
  - **“line gaussian”**
- Assumes that:
  - **The pads within a row are roughly rectangular**
    - And the row is not curved
  - **The track is straight within a row**

Charge Distribution Model

- (Non-homogeneous model)
- A bit more work:
  - **Use the track parameters to follow the track in the TPC volume**
  - **At specified intervals (25mm default)**
    - Create an electron cloud
    - Drift the cloud to the endplate using the simulation code
  - **Fit a curve to the projected track**
    - Using a spline from GSL
  - **This distorted curve is the "line-gaussian"**
  - **Same assumptions as before**
    - Things are roughly straight in a pad row

Calculate Expected Distribution

- Integrate the “line gaussian” over required pads
- For each row
  - **Sum the total expected electrons**
  - **Calculate the probability of each pad collecting one of those electrons**
- Use the multinomial distribution to calculate the probability of the **observed** charge occurring
- L(row) == Pr(row) = $\frac{n!}{x_1! \cdots x_k!} p_1^{x_1} \cdots p_k^{x_k}, \quad \text{when } \sum_{i=1}^{k} x_i = n$

Calculate the – log (L)

- So log L(row) =

$$\log\left(\frac{n!\,p_1^{x_1}\ldots p_n^{x_n}}{x_1!\ldots x_n!}\right)=\log(p_1)x_1+\ldots+\log(p_n)x_n+\log\left(\frac{n!}{x_1!\ldots x_n!}\right)=\sum\log(p_i)x_i+constants$$

- Sum up the row log(L)
- Multiply by -1 at the end
- Give the result to Minuit which calculates another set of hypothetical track parameters

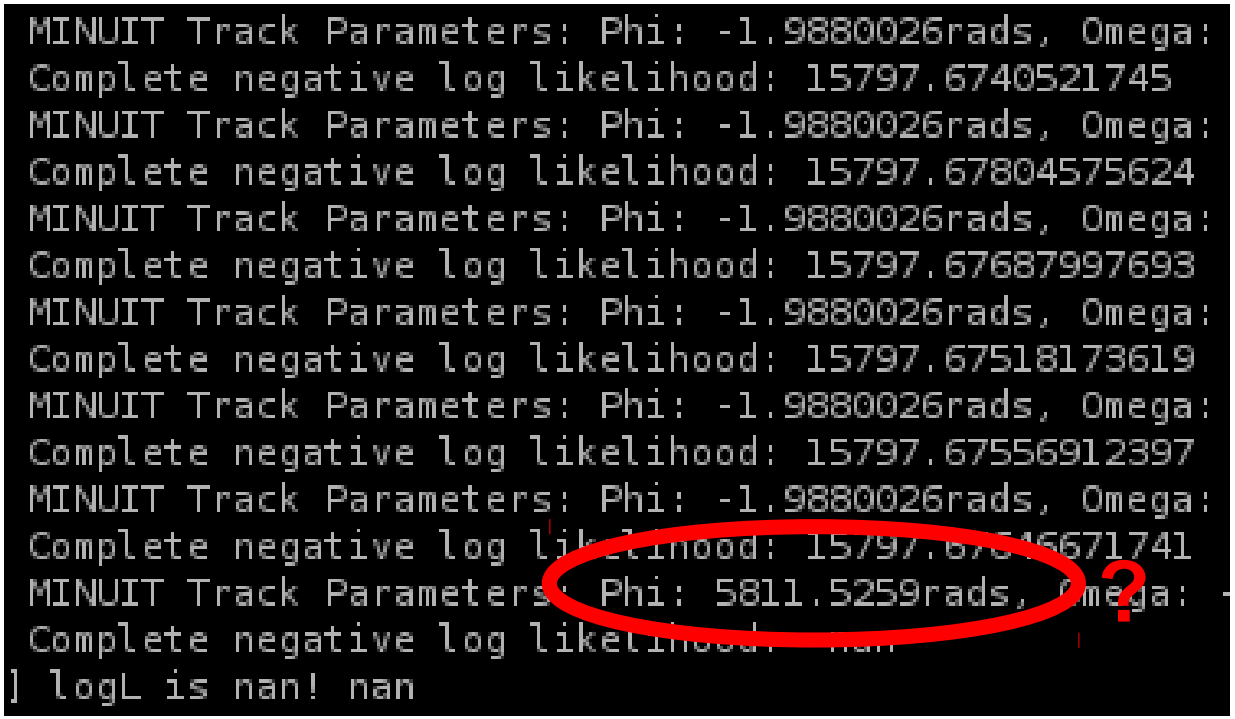
What Should Happen?

- Minuit finds the most likely track
- Parameter covariance is calculated

```
Minuit did successfully converge.
# of function calls: 48
minimum function Value: 15796.64848439
minimum edm: 0.001934680482668
minimum internal state vector: LAVector parameters:
     -1.988002027558
 -9.250080219873e-05
    -0.1592620260437
      1.617002074724
     -1.620328234995
     -1.037247782143

minimum internal covariance matrix: LASymMatrix parameters:
  1.9761598e-09  4.7270737e-12  2.0067998e-07  5.8692851e-10  1.3608833e-07 -5.8248745e-08
  4.7270737e-12  1.1347672e-14  4.8766488e-10  1.4088426e-12  3.2781621e-10 -1.3933115e-10
  2.0067998e-07  4.8766488e-10  3.4658532e-05  6.2251465e-08  1.6024574e-05 -5.9106829e-06
  5.8692851e-10  1.4088426e-12  6.2251465e-08  1.1142777e-08 -3.4870546e-06 -1.7251635e-08
  1.3608833e-07  3.2781621e-10  1.6024574e-05 -3.4870546e-06     0.01015075 -3.9787298e-06
 -5.8248745e-08 -1.3933115e-10 -5.9106829e-06 -1.7251635e-08 -3.9787298e-06  1.7791881e-06


# ext. ||   Name    ||   type  ||     Value     ||  Error +/-

   0   ||       phi ||  free   ||     -1.988002027558 ||3.143373817044e-05
   1   ||     omega ||  free   || -9.250080219873e-05 ||7.532486829552e-08
   2   ||        d0 ||  free   ||    -0.1592620260437 ||0.004162843513088
   3   || tanLambda ||  free   ||      1.617002074724 ||7.46417358234e-05
   4   ||        z0 ||  free   ||     -1.620328234995 ||0.07124166485602
   5   ||    sigma0 || limited ||     0.6949613282769 ||0.002398475487283
```

What Usually Happens...

- Minuit hovers around a minimum, then radically shifts a parameter



What usually happens...

- Or, after trying for a while it just gives up

```
WARNING: Minuit did not converge.

# of function calls: 88
minimum function Value: 15796.58798173
minimum edm: 0.8683345852382
```

- Tried a number of solutions
  - **Scaling parameters to same order of magnitude**
  - **Keeping diffusion constant along track**
  - **Changing the parameter uncertainties**

The Future

- Quick fix which works:
- while(! fit(track, tolerance))
  - **Tolerance *= 10**
- Taking all suggestions
  - **If anyone can find the solution then the next time we meet I'll buy the drinks! =)**
- Implement fitting functions/classes for the photo-electric calibration system