Off-Axis-NOTE-SIM-11

The Reconstruction of ν_μ and ν_e Monte Carlo events

L.Camilleri, A. Para

Abstract

This memo describes the reconstruction code used for studying the performance of an off-axis detector. The actual results obtained with this code will be described in a subsequent memo.

1 The Generation of Monte Carlo events

The events are produced by a version of GMINOS [1].

1.1 Neutrino energy distribution

It was decided to generate the events with a flat neutrino energy spectrum and to weight the events later on during the analysis according to the spectra prevailing at different possible positions of the detector. Since the ν_μ energy spectrum is sharply peaked at low energy and has a long tail, the events were generated in two stages: flat in energy between 1-3 GeV, and again flat in energy between 3-20 GeV. This made more optimum use of computer time. During the reweighting process the different numbers of events per GeV in the two energy intervals were taken into account.

1.2 Event generator

The event generator used was NEUGEN vs 2.0.

1.3 Detector description

The detector consisted of slabs of absorber material followed by glass RPC's. The absorber of choice is particle board of density 0.7 gm.cm^{-3}. This was simulated by the appropriate combination of air and plastic. Different absorber thicknesses were simulated.

1.4 Output format

The output of each event is in an ADAMO table. It includes information on

- the incident neutrino,
- the produced particles,
- the position of each hit in any active detector (the RPC's in this case) and its corresponding ionization.

2 The Reconstruction code

It can be found in the directory:

/afs/fnal/files/code/off-axis/off-axis_01/code/reconstruction

It consists of three phases:

- Unpacking the hits into arrays correponding to planes of active detectors and strips,
- Track finding and track fitting,
- Output of ntuples.

Most of the code is in subroutine **reco_event_usr.F**.

2.1 Detector description

The detector is decribed by a few variables:

max_plane: the number of active planes in the detector.
strip_width: the width of a strip.
max_strip: the number of strips per active plane.
x_l_edge: lower edge of the x planes.
y_l_edge: lower edge of the y planes.

In order to save computer time, doubling the absorber sampling was studied by ignoring hits in alternate planes rather than regenerating events with twice the absorber thickness. Whether and X and a Y plane or just an X or a Y plane were placed after an absorber plane was also studied by ignoring certain hits. This was governed by four variables: **ievenx, ioddx, ieveny and ioddy**. Setting **ievenx** to 1 or 0 respectively used or ignored the x hits in an even plane. Similarly for the other three variables.

3 Unpacking of hits

After the decoding of the hits into planes and strips, the following variables and arrays were filled.

ihit_hx: Number of strips hit in the x direction.Note that if a strip was hit many times, it resulted in just one hit strip, given the readout nature of RPC's.
hhit_x(i): X position of i'th hit x-strip.
hhit_zx(i): Z position of i'th hit x-strip.
ihit_hy: Number of strips hit in the y direction.
hhit_y(i): y position of i'th hit y-strip.
hhit_zy(i): Z position of i'th hit y-strip.

Three more arrays are filled:

hitx(iplane,irow): Set to 1 if x strip **irow** in plane **iplane** has been hit. Set to zero otherwise.
hity(iplane,irow): Set to 1 if y strip **irow** in plane **iplane** has been hit. Set to zero otherwise.

plane_z(iplane): Set to 1 if plane **iplane** has been hit. Zero otherwise.

The event is contained between x strips **ilow_x** and **iup_x**, y strips **ilow_y** and **iup_y** and planes **if_plane** and **il_plane**.

The net transverse momentum in each view was calculated as the sum, over all the hits in that view, of the sine of the angle subtended by each hit at the upstream-most hit in that view. The rational is that the more energetic a particle the more hits it will give rise to and therefore the more it will contribute to this sum. The sine takes into account the transverse component. The hope was that neutral currents could be rejected using this variable. The result is stored in **peet_x** and **peet_y**.

These arrays and variables are located in common block: **hit_track.inc**.

4 Track finding and fitting

The method used is that of the Hough transform. It is applied separately in each view.

- A series of straight lines spanning all possible lines within the detector is generated by looping over **nx** slopes and **ny** intercepts.
- For each line the number of hits within a tolerance **toler** is recorded.
- The line with the largest number of hits is identified.
- At each plane, the hit closest to this straight line is identified.
- The closest hits that are within **toler_fit** of this line are then fit to a new straight line identified by a new slope and intercept.
- The procedure is repeated twice, at each step, again finding the closest hits to the preceding line and fitting them to a new straight line.
- Although the detector is non-magnetic, particles can still have curved trajectories because of scattering. The final step in the fitting is therefore a parabolic fit done in subroutines **improve_track_x.F** and **improve_track_y.F** for the x and y view respectively. The fit for track **ntr** in the x-view is of the form:

$$\mathrm{aax(ntr)} \times \mathrm{z}^2 + \mathrm{bbx(ntr)} \times \mathrm{z} + \mathrm{ccx(ntr)}$$

- At each hit plane, the hit closest to the straight line found after the last iterations, is recorded. These closest hits are then fit to a quadratic expression, provided they are within **toler_fit** of the straight line.
- ALL hits within **toler_fit** of the quadratic fit are now associated to this track. They are blanked off so as not to be used in subsequent trackD fitting. This is done by setting to 1 the appropriate member of the arrays:

khuse_x(iplane,irow) or **khuse_y(iplane,irow)**, where **iplane** and **irow** are the plane and strip numbers of a used hit, and

jhhuse_x(ihit) or **jhhuse_y(ihit)** where **ihit** is the order number of the used hit.

- This track finding and fitting procedure is then repeated on the unused hits until no more hits are left or until no track is found.
- The whole procedure is applied to both views.

Note that, so far, there has not been any attempt to correlate the two views. For each track, **ntr**, the following parameters are stored in arrays located in common block: **hit_track.inc**.

aax(ntr),bbx(ntr),ccx(ntr): parameters of the fit.

nplanes(ntr): number of planes containing track hits.

beg_x(ntr): Plane number of first plane of track.

end_x(ntr): Plane number of last plane of track.

trlen_x(ntr): Number of planes traversed by track. This differs from **nplanes (ntr)** if there are gaps in the track.

chi_x(ntr): χ^2 per degree of freedom of fit (χ^2 divided by (nplanes - 3)). This only uses the closest hit for each plane.

tr_wid_x(ntr): Track width: the root mean square deviations from the parabolic fit of ALL the hits associated to the track, not just the closest on each plane.

hits_road_x(ntr): Number of hits associated to this track.

igp_x(ntr): The length of the first gap encountered along the track, starting upstream (plane(s), with no hits along the track, followed by a resumption of hits).

igpla_x(ntr): The position of the first gap, defined as the difference between the plane numbers of the last hit plane before the gap and the first plane on the track.

These last two variables are useful in identifying electrons and rejecting conversions.

And similarly for the y-view.

5 Ntuples

An **ntuple** is written for each event. For the moment, only parameters referring to the track with the most hits are included in the ntuple. This should eventually be changed to include parameters from all tracks. The ntuple variables are stored in common block: **cwn_common.inc**.

They are booked in subroutine **reco_hist_usr.F**.

The ntuple list is as follows:

Parameters of the **generated** event:

enu: Neutrino energy.

x: Feynman x.

y: Fraction of total energy into hadrons.

iact: 1 if a Charged current and 0 if a neutral current interaction.

xv, yv, zv: Vertex position.

esh: Amount of energy going to photons.

wei: A weight attached to the event, based on a beam file read at the beginning of teh reconstruction. This is no longer used for the most part.

Parameters of the **reconstructed** event:

nhit_x, nhit_y: Total number of hits in the x and y views.

ivtx_pl, ivty_pl: First hit plane in the x and y views.

ntrx, ntry Number of reconstructed tracks in the x and y views. Note that many of the ntuples have been generated stopping the reconstruction after the first track, so that these variables are never larger than 1.

peet_x, peet_y: Net transverse momentum in the x and y views as defined earlier.

The following variables refer to the track containing the largest number of hits in x and y.

tr_len_x, tr_len_y: Track length.

h_roa_x, h_roa_x: Number of hits associated to track.

chix, chiy: Normalized χ^2.

nplx,nply: Number of planes containing track hits.

aapx,angx,ccpx and aapy,angy,ccpy: Track fit parameters: aax, bbx, ccx and aay, bby, ccy.

bksca_x, bksca_y: Variable intended to indicate whether there was a possibility of a back scatter. Not to be used.

tr_widt_x,tr_widt_y: Track width as defined above.

igap_x, igap_y: The length of the first gap encountered along the track.

ipl_gap_x, ipl_gap_y: The position of this first gap: last hit plane before gap.

itr_beg_x, itr_beg_y: First plane of track.

The ntuples are stored in /afs/fnal.gov/files/code/off-axis/off-axis_02/ntuples.

References

[1] The generation of events will be described in a note by A. Para.