The Reading and Layout of MC output (v3.2)

Julie McEnery

December 20, 2001

1 Layout of the MC output files

Each MC output consists of a file header containing details about the parameters chosen to generate the data. Following the file header is a list of events. Each event has an event header describing parameters for that event such as theta, energy etc. Following the event header are two blocks of information, the first describing the response of the hit PMTs and a second describing the location and properties of particles in the pond and outriggers. For each PMT, we store the list of the arrival times of each photon. This is illustrated schematically below.

```
FILE HEADER
  Event1 Header
         -PMT1 info
              -photon1 at pmt1
              -photon2 at pmt1
              -photon3 at pmt1
              ...
         -PMT2 info
              -photon1 at pmt2
              -photon2 at pmt2
              -photon3 at pmt2
              ...
         -PMT3 info
              -photon1 at pmt3
              -photon2 at pmt3
              -photon3 at pmt3
              ...
         ...
         -particle1 info
         -particle2 info
         ...
  Event2 Header
  ...
```

2 File header

This is found at the top of each MC file. Each entry is preceeded by a !. Most of the parameters are self explanatory. Some that might not be are:

detector type: This is 1 for Milagro and 2 for Milagrito.
baffles/outriggers: 1 we have them, 0 we don't
nreuse corsika : How many times have we reused each corsika event, if this if greater than 0, then we have reused corsika showers with different core positions.
dynamic core: If this is 0 then the cores were thrown uniform in over a square area of length given by the core range parameter. If this is 2 then the cores are thrown uniform in core radius, over a circle of radius described in the core range parameter.

These parameters are not read by the DataRead program, but if you are unsure about the contents of a simulation file you can just look at the file header to see how the dataset was generated.

3 Event Header

Most of the information in the event header is read into the monteInfo structure:

```
eventData.recEvent.monteInfo.partId    = primary particle id
eventData.recEvent.monteInfo.xCoreMC   = xcore
eventData.recEvent.monteInfo.yCoreMC   = ycore
eventData.recEvent.monteInfo.thetaMC   = theta
eventData.recEvent.monteInfo.phiMC     = phi
eventData.recEvent.monteInfo.nShowerMC = number of particle reaching the ground
eventData.recEvent.monteInfo.fintHeight= height of first interaction
eventData.recEvent.monteInfo.ePrimary  = primary particle energy
```

Also written to the event header, but not stored in a data structure during reading are: number of particles hitting the pond, number of electromagnetic particles reaching the ground, total energy of em particles reaching the ground. number of muons reaching the ground, total energy of these muons, number of hadrons reaching the ground, total energy of these hadrons and the throw area used.

4 PMT and photon information

The data read function reads through the PMT information to fill the following variables.

```
eventData->calEvent.calData.nPMTs = number of hit pmts
eventData->calEvent.calData.nAS = number of AS tubes hit
eventData->calEvent.calData.nMUON = number of muon tubes hit
eventData->calEvent.calData.nOUT  = number of outriggers hit
```

For each pmt that was hit by at least one photon we store the the following information:

```
eventData.calEvent.calData.gridNo[i] = tube number
eventData.calEvent.calData.time[igrid] = jittered time
eventData.calEvent.calData.pes[igrid] =  smeared pes
eventData.calEvent.mcData.mcPes[igrid] = unsmeared pes
eventData.calEvent.mcData.mcTime[igrid] = unjittered time
```

Where i goes from 0 to (number of hit pmts)-1.

We also store the arrival time of each photon. The user function procPhot is called from the dataread program. Usually it does nothing, but if someone wanted to write code to process the photons differently (by modeling the electrons more accurately), it could be done by calling this function and rewriting the jittered and smeared pe values back into the calEvent structure.

5 Particle information

We store information about the particles that hit the pond and outriggers. This information is written to the muonMC data structure in the following way:

```
eventData.recEvent.monteInfo.muonMC.muonID[i] = particle id (see below for a list)
eventData.recEvent.monteInfo.muonMC.muonEnergy[i] = particle energy
eventData.recEvent.monteInfo.muonMC.muonTheta[i] = theta
eventData.recEvent.monteInfo.muonMC.muonPhi[i] = phi;
eventData.recEvent.monteInfo.muonMC.muonXMC[i] = particle x postion
eventData.recEvent.monteInfo.muonMC.muonYMC[i] = particle y position
```

1	gamma
2	positron
3	electron
4	neutrino
5	muon+
6	muon-
7	pion0
8	pion+
9	pion-

Table 1: list of particle ids in Milagro MC output files.