MilliQan Simulation Manual

Gabriel Magill
gmagill@perimeterinstitute.ca[1, 2]
[1]*Perimeter Institute for Theoretical Physics*
[2]*McMaster University*
(Dated: December 14, 2016)

We describe all the contents of the milliQan folder. If any bugs or outdated information is detected, please communicate them to gmagill@perimeterinstitute.ca.

I. INSTALLATION

Detailed installation instructions are located in /milliQan/README/. We give here a useful overview that you should read in addition. The first step is to install Geant4 (G4), preferably with Qt and the external Datasets (see G4 website for full details). Once that is done you can install the simulation. The link for the Github repository is `https://github.com/OSUmilliQan/milliQan`. You should clone this repo in a local directory. To install the simulation, you can do:

cd PathOutside_milliQan
mkdir BuildMilliQ
cd BuildMilliQ
cmake -DGeant4_DIR=/PATH/geant4.10.2-build/lib/Geant4-10.2.0/ PATH/milliQan/geant4/
(the first link is where the G4 build directory is installed, the second is the milliQan/geant4 source code is located)
make
./MilliQ
or you can run it from the macro:
./MilliQ PATH/milliQan/geant4/config/mcp.mac PATH/milliQan/geant4/config/default.ini
or
./MilliQ PATH/milliQan/geant4/config/mcp.mac PATH/milliQan/geant4/config/cosmicmuons.ini
In order to properly compile/run, You need to edit the file config/particles.ini or config/CosmicMuons/CMparticles.ini. You need to specify the path to the kinematic distribution .txt (it cannot be a .dat file). There is a different distribution for charges 0.001, 0.01, 0.1 and for 48 mass points. These kinematic distributions are included in milliQan/SourceFiles.

II. MILLIQAN FOLDER

A. Macro File

In /milliQan/BASHScript/, there are files called run.mCP.sh and submit.mcp.exclusions.sh. This a big bash script that cycles through masses, electric charges, particles, etc. and saves all the output. There is a setting to run on a computer cluster or a laptop. You can modify it to your architecture.
Important: be sure to specify your correct paths in /milliQan/BASHScript/paths.mCP.sh.

III. GEANT4 SIMULATION CLASSES

In the following, all the classes are prefixed by MilliQ, and suffixed by .cc. They all have a partner .hh file. The construction of the detector is a bit involved. DetectorBlockLV constructs a block (a scint + pmt). This is called by DetectorStackLV which parametrizes all the blocks into one stack (one layer of the detector) using DetectorBlockParameterisation. DetectorStackLV is called by DetectorConstruction, which parametrizes all the stacks into one detector using DetectorStackParameterisation.

A. DetectorBlockLV

Constructs the scintillator, light guide and PMT from the dimensions provided by DetectorConstruction. Sets the PMT efficiency (detection) spectrum, polished surfaces, reflectivity, R index. Makes PMT and scints sensitive (in the G4 sense).

B. DetectorConstruction

Calls the StackLV and Stack Parametrization, which in turns creates everything else about the detector. Sets all the dimensions, number of blocks, number of stacks, and the shield. Constructs all the materials. Sets reflectivity, absorption length, fast time constants and all the other detector settings.

C. MilliQDetectorShield

Is called by DetectorConstruction.cc, and creates two shields around the detector. All the shield details are given in Geometry configuration file.

D. DetectorStackLV

Creates one stack.

E. DetectorStackParameterisation

Parametrizes all the stacks into the detector.

F. EventAction

Collects all the hits from the scintillators and PMT collections and passes them to Analysis. Obtains Analysis output and fills the .root files event per event.

G. Monopole

Constructs the mCP from sratch. Defines all the quantum numbers.

H. MonopoleEquation

User defined EOM.

I. MonopoleFieldSetup

Used to setup EOM, magnetic field stepper and chord finder for a mCP going through a magnetic field. Currently only works with global magnetic field.

J. MonopolePhysics

Used to specify the charge, mass, and physics of the mCP. As it currently is, you cannot change the mass or electric charge of the mCP without recompiling the program. The charge and masses are set in the configuration file.

K. PMTHit, PMTSD, ScintHit & ScintSD

Gets called upon by Stepping Action when a particle enters the sensitive components. Records the position, time and energy of particles. Packages up everything in hit collections, and passes information to EventAction.

L. PrimaryGeneratorAction

Sets the particle gun, it's momentum, energy and position. Has a function that reads in 4 vectors and coordinates from external files. Can't set the number of events in a run to be larger than the size of the external file ∼25000 events. The simulation assumes a parameter file that tells it a list of momentums and vertices. However, in a Qt visualization, you can override this parameter file using /gun/momentumxvertex and /gun/xvertex to specify various vertices and momentum choices (where x can be x, y or z). Setting the momentums only affects the normalized direction. The overall energy (which determines the norm of momentums) must also be set, using /gun/calibenergy.

M. RunAction

Creates the .root files. There are 3 main containers in the ROOT file. The first replicates the CAEN output. The second is a list of important information. The third is the dE/dx data that is associated to a particle. There is an option in DetectorConstruction (fAlternate = 1) that turns off all the geometry.

N. DataFormat

Is to be thought of in conjunction with RunAction described above. Refer to both to find the needed information.

O. UserEventInformation

Keeps track of number of photons emitted.

P. Waveform

In our G4 simulation, the only information we can extract are the times of PMT hits. To convert this to a waveform, given one photoelectron incident on the PMT at a given time, we generate a single photo-electron waveform. We do this for every hit in every active channels. On top of this, we put in background noise (randomly generated small numbers) in all channels at all times. The single photo-electron waveform is currently obtained from: The IceCube data acquisition system: Signal capture, digitization, and timestamping (0810.4930). A waveform as output by CAEN is a collection of ADC counts containing voltages or charges. The resolution of CAEN is 12 bits, the Vpp is 2.5V, and 1ADC=$2.5V/(2^{12}-1)$=0.61mV.

Q. Analysis

The online triggers and the data output in the ROOT file via RunAction is calculated here. We describe in detail what is output. The quantities in () bracket shows the length of each vector, and times are in ns, energies are in MeV, voltages in mV.

1. Online Triggers

Online triggers are performed to see if G4 should output anything to a ROOT file. We begin with the whole detector (Y x Z x NStacks), where Y is the number of blocks up, Z is the number of blocks across and NStacks is the number of layers. We then divide it into Y×Z/4 modules. Each module is a $2 \times 2 \times$ NStacks "subdetector". In each stack of a given module, a group is formed by two neighbouring PMTs along Y (with the same Z). A coincidence within 150ns (see config files) in any two groups of a module sets a flag to true for the Online Trigger.

2. MilliQCAEN

In the event of an Online Trigger, the following MilliQCAEN information is output to ROOT:

- std::vector⟨G4int⟩ WaveformLengthPerChannel (#channels): for each channel, output the number of recorded ADC samples.
- std::vector⟨G4float⟩ WaveformVoltage (long): a continuous vector, filled with all the voltage samples for all the channels. To be used in conjunction with WaveformLengthPerChannel, which specifies how many consecutive elements correspond to each channel. If a channel hasn't seen any activity, there will 0 samples associated with it.
- std::float TimePerSample: Outputs the time corresponding to one ADC sample. This is digitizer dependent.
- std::float FirstPMTTime: For each event, the waveform length per channel (equal for all active channels) is big enough to hold all the voltages generated from each PMT hit. FirstPMTTime corresponds to some local time corresponding to when the first sample was taken.

3. MilliQAll

We now describe what is output in the MilliQAll output of the ROOT file. For each event, we output the following pre-calculated potentially useful quantities:

- std::vector⟨G4int⟩ ActivePMT(#Active PMTs): A list with ID of all active PMTs
- std::vector⟨G4double⟩ PmtMedianHitTimes(#channels): Gives median hit times of all PMT hits in a channel
- std::vector⟨G4double⟩ PmtAllHitTimes(Total#PMThits): Gives all pmt hit times, in a continuous vector. The number of entries corresponding to each channel can be inferred via NumberPMTHits.
- std::vector⟨G4int⟩ NumberPMTHits(#channels): Gives number of PMT hits in a channel.
- std::vector⟨G4double⟩ TimeOfFlight(#channels): Gives median PMT hit time minus scintillator first hit time in a channel
- std::vector⟨G4double⟩ TotalEnergyDeposit(#channels): Gives total energy deposit in scintillator channel
- G4int FirstHitScintillator: Gives ID of first scintillator hit by mCP particle
- G4int PhotonCountAllScintillators: Gives total photon count across all scintillators in an event

IV. CONFIGURATION FILES

In milliQan/geant4/config, there are a number of configuration files. In general, if making any changes to configuration files, you must delete the inside and rebuild the MilliQBuild Directory in order for the program to properly recognize these changes. If running via the bash script, it deletes/builds from scratch automatically.

1. PMT

Used for PMTs information, specifies: OuterRadius, Radius, CathodeRadius, CathodeHeight, CathodeDepth, Length, HousingReflectivity, PhotonEnergies Efficiency spectrum, ReR and ImR, timePERsample, RIndex, AbsLength for a variety of PMTs.

2. Geometry

Used for detector information, specifies: Version, NBlocks_X, NBlocks_Y, NBlocks_Z, NStacks, BetweenBlockSpacing_X, BetweenBlockSpacing_Y, BetweenBlockSpacing_Z, Offset_X, Offset_Y, Offset_Z, X Y Z length, LightGuideLength

3. Scintillator

Used for scintillator information, specifies: Density, CarbonContent, HydrogenContent, PhotonEnergies, FastScintOutput, RIndex, AbsLength, ScintillationYield, ResolutionScale, FastTimeConstant, FastScintillationRiseTime, SlowTimeConstant, YieldRatio, BirksConstant.
Used for housing and light guide information, specifies: PhotonEnergies, Efficiencies, Thickness, Reflectivity.

4. Others

particles.ini, specifies: Particle, MagneticCharge, ElectricCharge, MonopoleMass, FileName (where particle distributions are located), PathName (where FileName are located), coincidenceThreshold (for online trigger).
mcp.mac, specifies: Number of events in a run.

5. Cosmic Muons

In milliQan/geant4/config/CosmicMuons, you will find all the analogous config files for the cosmic muons run. These should all be set, but some details of the waveform output, PMT and Scintillators will differ from David Stuart's setup. Good to check over the settings!

V. OFFLINE ANALYSIS

A. AnalyseGeant4ROOT

Located in /milliQan/OfflineAnalysis/. Has all the handles and structures to read in ROOT files generated from Geant4, calculate Offline Triggers, and output detector efficiencies to a good format. It also reads the helper file generated by the bash script of Geant4 which specifies how many events were simulated, how many events passed the online trigger, the masses/charges of the mCP in the event. The location of this helper file is specified in /milliQan/BASHScript/paths.mCP.sh under LaptopResults. AnalyseGeant4ROOT also outputs some useful data to text files. Currently, this part of the analysis is coded to work on 3 stacks. It begins by reading in the waveforms for each channels for each event, and identifying the peaks in the waveform using TSpectrum. There is currently an issue with TSpectrum in that it doesn't identify smaller sub-peaks that are visible to the eye. Tweaking the parameters of TSpectrum might solve this. Then, given the times of all the peaks for each channels, it checks if there are 3 channels in different layers with 3 signal peaks in a given time window. The user can choose exactly how this is done via the options:

- ParticleTrajectory: Given a hit in one layer, the hit in the 2nd layer must fall within a 3×3 grid centered around the hit of the 1st layer. The hit in the 3rd layer must fall either in the PMT consecutive to that of the 2nd layer, or in a neighbouring PMT in the direction of motion of 2nd layer hit relative to the 1st layer hit (think straight diagonal line with bending, but no zig-zags). Impose time window.
- BackToBack: All hits must be in consecutive PMTs in each layer. Impose time window.
- AllNeighbours: Given a hit in one layer, the hit in the next layer must fall within a 3×3 grid centered around the hit of the previous layer. Impose time window.
- ThreeFoldAnywhere: 3 Hits anywhere. No time window.

If it succeeds in identifying a signal event, it writes the result to an Efficiency file. The *config* file associated to this programme, located in milliQan/OfflineAnalysis/AnalyseGeant4ROOT/ConfigFiles/, is used to specify: where are the ROOT files located, what are the details of the detector, which particle type are we considering (mCP_UFO, Y1S, Y2S, Y3S, JPsi), the coincidence threshold, the waveform voltage below which we call it a background, the time per digitizer sample, and the offline trigger strategy. It also can be used to specify which data of MilliQAll we want to output to .m format, for which particles, for which charges, and masses.

B. noWaveformSensitivities

After running AnalyseGeant4ROOT, an efficiency file is produced.
IMPORTANT: This efficiency file needs to be ordered according to the mass and charge of the mCP particles. So either order the efficiency file after generating it, or order the helper file of Geant4 described at the beginning of the previous subsection.
If this generated efficiency file contains all the masses, charges and particles (mCP_UFO, Y1S, ...), noWaveform-Sensitivities.cc can calculate a charge versus mass sensitivity plot. A vector of Number of Background events must be specified at the beginning of the file, to indicate the number of background events given 300 and 3000 fb^{-1} of LHC luminosity. The backgrounds will depend on the offline trigger strategy, and on how well we can differentiate dark current from signal events via a waveform analysis template matching. The details on how it calculates the sensitivities is located in /milliQan/README/Set_Limits_milliQ_notes.pdf (courtesy of Itay Yavin). The values for the cross sections, geometric acceptances and efficiencies should be saved and located in /milliQan/DataFiles. Upon running the noWaveformSensitivities executable, a number of arguments are provided to point it to all the correct directories. Details are in the main function.

VI. COSMIC MUONS

This section is to describe the work that has been done towards simulating cosmic muons with our G4 simulation. The configuration files for this setup are located in /milliQan/geant4/config/CosmicMuons/. It employs a much smaller version of the milliQan detector with only one layer, meant to replicate David Stuart's setup. The incident muons are generated according to a $\cos^2(\theta)$ distribution in the file /milliQan/geant4/config/CosmicMuons/GenerateMuonDistributions.nb. To get this working, the output of the G4 simulation using the bash script with proc variable set to cosmicmuons is to be read in with AnalyseGeant4Root.cc and plotted via PlotGeant4Output.nb. Sample plots already exist in the mathematica notebook. To open up this configuration for visualization via Qt, one must go inside /milliQan/geant4/MilliQ.cc and change the default configuration file from default.ini to cosmicmuons.ini and recompile.

VII. CONCLUSION

This simulation was created by Gabriel Magill (Perimeter Institute for Theoretical Physics) and James London (Ohio State University). Configuration files interface were developed by Brian Francis (Ohio State University). If you have any questions, please contact gmagill@perimeterinstitute.ca.