A Geant4 simulation for the CrystalBall@MAMI Setup

Derek Glazier, University of Edinburgh

6th May 2008

1 Introduction

As advances in Geant3 particle tracking were discontinued a number of years ago, it is clear that to achieve as realistic a simulation as possible a move to Geant4 is required. In particular simulation of hadronic interactions is thought to be much improved with G4 and detailed validation exercises have been carried out. G4 uses object-orientated C++ programming.

The code included here contains a G4 simulation for the CB, TAPS, PID, MWPC detector setup for the first and second round of experiments at MAMI.

This report will outline the structure and the status of the simulation, named *A2*, as well as giving some instructions for installing and running, and finally list some thoughts on what still needs to be done.

2 Installation

Installation of this simulation requires a pre-installed release of G4. Users should refer to the G4 webpage for details. The simulation has now been upgraded to run with version Geant4.8.2 and Geant4.8.3 and it is strongly recommended that one of these versions is installed, but it should also run with Geant4.8.1.p01. The main difference for the user is that the hadronic physics lists are now packed with the main source code and do not require explicit compilation and inclusion into the Makefile. For running Geant4.8.1.p01 the physics lists should be compiled by hand.

After the G4 system is installed the user is recommended to try and compile a few of the G4 examples.

The code package can currently be downloaded from http://nuclear.physics.gla.ac.uk/~dglazier/A2. The user can then unpack A2 into their G4WORKDIR and make. A2 uses a ROOT based input/output and so the variable $ROOTSYS should be defined. The executable A2 should be found in $G4WORKDIR/bin/$G4SYSTEM/.

How to run the simulation is explained in sec. 6.

3 The A2 Package

The A2 directory contains all the additional code required to make the simulation work including some example macros for running the simulation in either batch or interactive mode. The directories contain :

docs	the user manuals, currently this document!
cbsim	relevant files from cbsim and the ROOT conversions
macros	macro files to control the batch and interactive running of the simulation
include	the .hh files for the new A2 classes
src	the .cc files for the new A2 classes
pics	some graphics created by A2
acquroot	contains new files required for running with AcquRoot

The file A2.cc controls the simulation by creating the A2 classes and assigning them to the G4 manager classes. It then allows an interactive session to be run if no initial macro is given, i.e just the command A2 is given. Or runs in batch mode if a run macro is given, e.g. A2 macros/doppi0.mac.

The files GNUmakefile and hadronic_lists.gmk are required for compilation. The file G4History.macro is produced by A2 and saves the interactive commands given.

The files CrystalConvert.in and taps.dat come from cbsim and allow a meaningful readout of the detectors. taps07.dat contains the setup for the new TAPS configuration.

4 Geometry

The A2 class A2DetectorConstruction is responsible for all of the geometry built. Currently it can call on three detector setups the Crystal Ball, TAPS and the PID. The Wire Chambers are still to be implemented.

Each detector has it's own class and A2DetectorConstruction decides whether to build it based on a setup macro, see sec.6.1.

4.1 Materials

The majority of materials needed are defined via a NIST database, implemented via the G4NistManager class. A list of the NIST defined materials can be found by typing /material/nist/listMaterials, at the Idle> prompt, or in the file $G4INSTALL/source/materials/src/G4NistMaterialBuilder.cc.

Some materials from cbsim are not found in the NIST database and have been entered "by hand" in A2DetectorConstruction.cc. The extra materials are rohacell, fibreglass, plastic (for PID lightguide), mumetal and LD2.

4.2 The Crystal Ball

The geometry used for the Crystal Ball and surrounding material was taken from the cbsim file of A.Starostin. This can be found in the A2/cbsim directory as ugeom_sasha.F.

4.2.1 Software import from ugeom.F

Two methods were tried to export the crystal ball geometry directly from the cbsim file ugeom.F.

The first was to use the G3toG4 utilities which are part of the G4 libraries. This requires an .rz file of the geometry to be produced by cbsim[1]. The first problem with this method was the Trap shape used in G4 does not accept sides with zero length, which had been used in cbsim to construct the triangular faces of the crystals. However Ken Livingston showed that this could be bypassed by a simple fix in the G4Trap class. After this G4 was able to produce a reasonable visualised likeness to the CB and TAPS, however attempts to track particles through the CB failed as they never underwent any interactions.

Fortran .rz file can be converted to ROOT readable format via the ROOT utility rz2root, so a second attempt to pass cbsim to G4 was made via ROOT's xml interface. This failed due to a mismatch in the syntax of the ROOT xml output and G4 xml input files. However even if such a scheme had been successful it almost certainly would have also failed to track the particles (for reasons explained below).

4.2.2 Manual import

After attempting the two software conversions I decided the best solution would be to manually rewrite the Fortran code in C++ using the G4 libraries. This was realised with the A2DetCrystalBall class. When writing the detector I used both the files ugeom_sasha.F and cbsasha.C (the ROOT conversion) in A2/cbsim for guidance. To simplify matters for myself I decided not to rewrite the prism.F function, which calculates the crystal, major and minor triangle parameters, but to just take the calculated results from the exported ROOT file cbsasha.C.

Initially I followed the cbsim recipe for placing the crystals first in Minor triangles, then the Minors in Major triangles, then the Majors in Hemispheres, then the Hemispheres in the Ball Volume. The crystals were again given a Trap volume, however instead of setting one side=0cm I have instead set it to 1nm, so G4 will not complain about the zero length.

This produced a CB that looked as expected. However, again when I fired the particles no interactions occurred in the ball!

The problem lies with the way G4 handles overlapping volumes. The fact the two hemispheres nearly exactly overlap meant G4 was unable to determine which hemisphere's daughter volumes, i.e Major triangles, it was in.

[1]To do this you need to add the line: call grfile(21,"crystalball.rz","ON") after call GGCLOS in ugeom.F

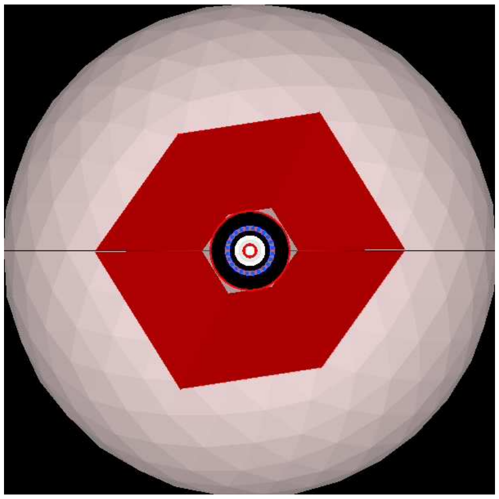


Figure 1: The cut crystals of the ball.

To overcome this I got rid of the 2 hemisphere volumes and place the Major triangles directly in the ball volume and applied the rotation and translation of the hemispheres directly to the Major Triangles. This appeared to give success as particles fired at the ball now interacted.

However it turned out the problem still existed close to the equator of the ball, the reason being that Major triangles here exist for both the upper and lower halves.

My eventual solution to the overlaps problem was to place each crystal individually into the World volume. Each crystal is rotated and translated by the product of the transforms for it's Minor, Major and Hemisphere.

Every crystal now tracks particles!

The next issue to resolve was for the cut crystals in the tunnel regions. Previously this was realised by placing one overriding volume CCUT in the ball. For G4 I chose to subtract off this volume from each individual crystal before placing it in the World volume. This is done via the G4SubtractionSolid class, and just needs the same translation and rotation information as used in placing the crystals and so prove to be relatively straightforward. The cut crystals can be visualised using the RayTracer visualisation and if you look down the beamline you can see that the crystals stop at the tunnel . In addition I was able to convince to myself that it did work by firing particles into the region that should be cut and seeing no interactions.

4.3 TAPS

The TAPS detector was implemented from a straight interpretation of the ugeom_taps.F code. The class is called A2DetTAPS. One slight change I have made is to readout every taps veto detector that fires, previously it was only

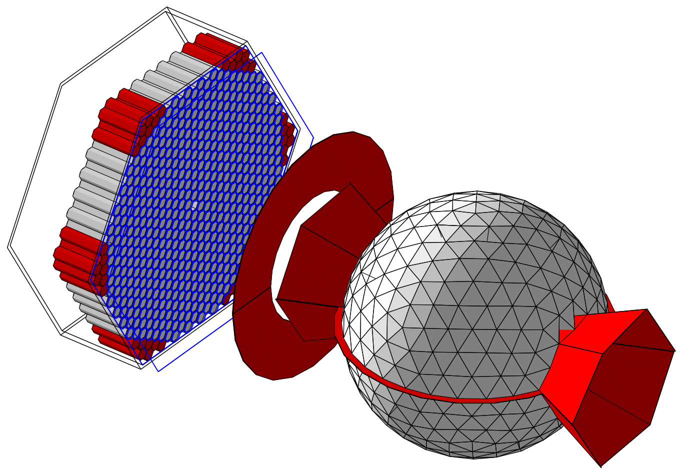


Figure 2: A picture of the CB-TAPS setup using DAWN visualisation. Unfortunately the drawing is not perfect, but this is due to the visualisation driver not colouring the right areas and not the actual geometry. Note, TAPS vetos are in blue and dummies are in red.

read if it's corresponding BaF2 crystal fired. Someone can inform me if I should change back.

TAPS can be configured without the need for recompilation by specifying an input file eg. taps.dat or taps07.dat for the smaller TAPS, the number of active crystals and the distance from the centre of the ball, see section 6.1 for details. By this means TAPS can be setup in it's original configuration or it's new for 2007 configuration by changing a few lines in the setup macro.

Figure 2 shows a visualisation of the CB-TAPS detectors.

4.4 The PID

The implementation for the PID varies from the original, as a more realistic right angular wedge shape is used for the scintillator shape, rather than a symmetric trapezoid. The effect of the change should be small.

PID 2 is also available for use and can be sepecified in the DetectorSetup.mac file. PID2 is rotated by 180° compared to PID1 and the support structures are made from plastic, to give less material at the downstream end. The scintillators are 4mm thick compared to 2mm.

The rotation invalidates the PID_MC.dat AcquRoot file, a new one PID2_MC.dat is in the acquroot directory.

4.5 MWPC

The MWPCs have been implemented by Jamie Robinson of Glasgow University. He has preformed a straight translation of the ugeom.F wire chambers including the modifications of Kashevarov.

4.6 Targets

A choice of either cryo (for LH2, LD2) or solid target have been added. These were taken from ugeom_target.F and ugeom_solid_target.F respectively. The solid target has hardwired lengths for Lithium, graphite, calcium and lead as found in ugeom_solid_target.F, if other materials are needed the lengths should be put onto A2SolidTarget.cc. Instructions on how to select target and material are given in 6.1.

4.7 Trigger

Currently the only trigger put on the tracking is that all particles are killed after 1ms. This is implemented in the A2SteppingAction.cc UserSteppingAction function. Users can change the trigger time here. In principle a more accurate trigger should implemented.

4.8 ToF

Tests have been recently carried out towards adding a time-of-flight to the A2 CB setup and so a general TOF class has now been added to the A2 simulation. The dimensions of the bars and positions can be set via a parameter file so users can try out their own geometeries. These can then be read into AcquRoot with the addition of a few files contained in the A2/acquroot/tof directory from release A2.06.05. I will include full instructions in this section so it is easier for folks to add in.

4.8.1 Modifications to G4 A2 code

The main change is the addition of the A2DetTOF class. At the moment this will just construct bars of plastic scintillator with no extra materials. The geometry can be set using a parameter file, see for example TOF.par. Upon running the simulation a new file TOFpos.dat will be generated, this gives the positions of the (virtual) photomultipliers which are required for the AcquRoot detector setup files when using the TA2LongScint class.

In addition to this new class, changes were made to the A2DetectorContruction and A2DetectorMessenger classes to allow the TOF to be switched on from the DetectorSetup.mac.

Also the A2CBOutput class contains extra branches if TOF is being used :

- tofe, energy deposited in a tof bar
- toft, time of flight of bar
- tofi, index of bar
- tofx, x hit position
- tofy, y hit position

4.8.2 Running A2 with TOF

This just requires the additional lines in DetectorSetup.mac :

/A2/det/useTOF 1

/A2/det/setTOFFile TOF.par

The file TOF.par for 1 veto layer and 2 tof layers is arranged like this[2], (the comments are only in this manual) :

2 ////number of walls, wall⇒same dimension of bars, so 1 for veto 1 for tof bars

8 ////number of bars in a layer of 1st wall

1 ////number of layers of 1st wall

0 ////orientation (rotation of layer wrt z-axis), 0⇒vertical aligned 90⇒horizontal

20 300 1 ////bar dimensions x y z (cm), before rotation

0 ////additional transverse offset of bars (allows overlapping etc.)

0 0 293 //// position of center in the lab x y z (cm)

8 ////number of bars per layer in 2nd wall

2 ////number of layers in 2nd wall

0 0 ////orientation of each layer i.e both vertical

20 300 5 ////dimensions of bars in 2nd layer

0 ////additional transverse offset for bars in 2nd wall

0 0 300 ////position of 2nd wall in lab.

END ////don't forget the end comment!

This should in principle allow most setups to be added in a simple manner, (note I realise I still need to give the walls a rotaion parameter in case the are not positioned perp. to the z axis. Note if you want two identical walls, postioned at different positon just set the number of layers for the second equal to zero and it will copy the previous wall.

4.8.3 Additional code for running AcquRoot

There is already some utilities for scintilltor bars in AcquRoot, inparticular the class TA2LongScint can be found in the $acqu_sys source. I have therefore just used this class to interface the TOF simulation, in practise all that was required was an implementation of the ReadDecoded function. Users should probably familiarise themselves with this function and class before trying too much, basically it assumes each photomultiplier a detector and combines two

[2]the current AcquRoot interface assumes vetos are defined before the real tof bars.

into a bar, reconstructing time of flight, energy and time difference, which then gives the transverse hit position. In the ReadDecoded I only fill the BarHits, MeanEnergyOR, MeanTimeOR and TimeDiffOR arrays and calculate the hit position. One important point is the G4 simulation assumes the vetos have the first numerical indices and so you are required to tell TA2LongScint how many veto bars are used so it can either use them as vetos or ignore them in the case of real ToF bars. This is done with the line NVetos: 8 in the LongScint_MC.dat file. Again it is worth noting that the postions of each PMT is given in the simualtion TOFpos.dat file and these can be cut and pasted direcly into the LongScint_MC.dat file, the positions output will automatically give the correct position indexing for reading the simulated data.

The only other file touched in the $acqu_sys code is the MCBranchID.h which now includes the branches outlined above, and used in TA2LongScint::ReadDecoded.

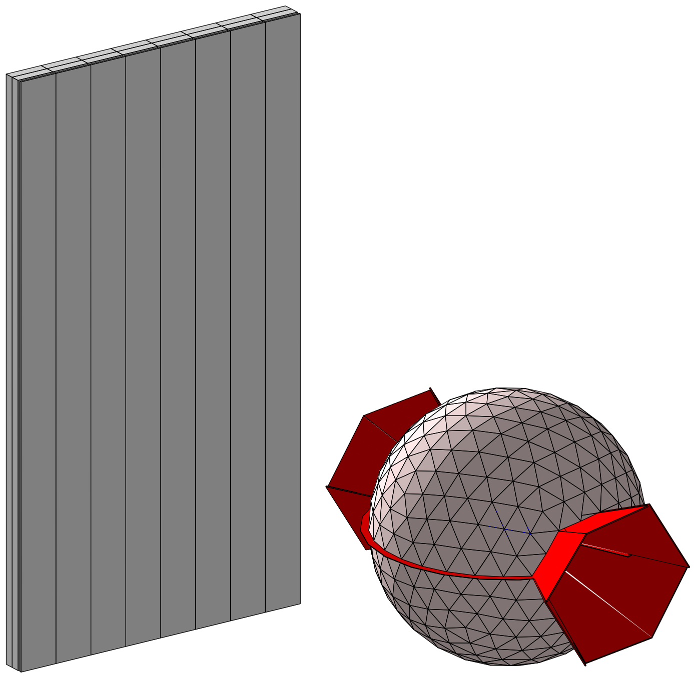
To test the simulation I have also implemeted a first go TOF apparatus for AcquRoot, this is the class TA2TOF.cc. To use this the source should be copied from A2/acquroot/tof/UserSrc to your own acqu/root/src directory. TA2UserAnalysis should be modified to include this new apparatus (just copy whatever you see done for TAPS), and the class should be included in User-LinkDef.h. Each user probably has there own version of these so I will not at the moment try to give a new version of these.

The TA2TOF class contains two detectors (TA2LongScint) veto and real TOFs, these have the setup files LongVeto_MC.dat and LongScint_MC.dat respectively, note the veto one has NVetos: -1 so it knows to read in the veto section of the simulated file.

The recontruct function then looks for some angular coincidences between different bars and vetos and tries to do some PID using the 2D cuts similar to the PID and NAI in the TA2CrystalBall class. I have already generated cut files for protons, neutrons and gammas but not pions yet, these are included in the A2/acquroot/tof/UserData directory. Finally if the particle is a nucleon the energy is calculated from its time-of-flight.

The full UserData tree for stand alone tof analysis is given in the A2/acquroot/tof/UserData directory with TOFMC.Offline being the head.

Below is a visualisation of the CB-TOF setup.



5 Physics Processes and Models

The main reason for transferring to Geant4 is to utilise up-to-date physics models. G4 provides detailed information on all of it's processes and models in its Physics Reference Manual, available on its website.

5.1 Electromagnetic Physics

The electromagnetic physics package is well established and it is expected that the "standard package" will be optimum for our purposes. G4 also offers a "fast" model which gives quicker computional performance at some loss of accuracy, but may be sufficient. Processes included are :

For charged particles: Multiple Scattering, Ionisation, Bremsstrahlung, Synchotron, Cerenkov, Transition Radiation, High Energy Muon Processes and Annihilation.

And for gammas: Compton, Pair Production, Photo-Electric.

These models are developed from the Geant3.2.1 package and include bug fixes and further development.

Just to note, G4 also supplies a Low Energy package which extends these interactions below kEV, and an optical photon package, which can be used for example to track scintillation light using reflection, refraction, absorbtion, Rayleigh and wavelength shifting.

5.2 Hadronic Physics

Geant3.2.1 utilised either the GEISHA or FLUKA models for hadronic interactions (+models from P. Pedroni?). In G4 GEISHA was converted into the G4LElastic and G4LEHadronicInelastic models. In addition a multitude of new models have been developed covering energies from a fraction of a keV up to 1TeV. It is up to the user which models they use and with a large choice of models this can be a tricky task. G4 does provide some standard PhysicsLists selecting applicable models, to help and one of these should be sufficient for standard simulation. I have implemented a flexible A2PhysicsList which allows some choice of models at run time, for instructions on use see Section 6.2. For best simulation results these models should be tested and compared on our detector setup with real experimental data. A program of validation exersizes is planned and all users are incouraged to make their own tests.

G4 defines 3 types of Hadronic Model :

Tabulated based on large databases.

Parameterised faster, parameters determined from fits to data,

Theory-based parameters chosen by comparing with thin target data.

As far is I am currently able to determine our energy regime is best described by some of the theory-based models. For inelastic channels these are the Cascade models (Binary and Bertini), Chiral Invariant Phase Space models and perhaps a Pre-compund model. For Elastic channels we should use the G4HadronElastic model. These models are applicable for nucleons, pions, kaons, hyperons and ions. For more details see the physics reference manual.

For some purposes it may also be useful to utilise the high precision neutron models based on a variety of databases. This model is quoted as being valid from thermal energies up to 20MeV.

Currently recommended Physics List is QGSP_BIC

This uses the Binary Cascade model for inelastic hadronic interactions at out energies.

Note QGSP_BIC_EMV, has fast electromagnetic, QGSP_BIC_EMX has experimental electromagnetic, (do not know what that means currently!). With BERT the Bertini cascade model is used which is faster but possibly not so accurate, though the results may be sufficient for our purposes. If BIC or BERT is not specified then it defaults to the fast GEISHA type parameterisation for inelastic hadronic interactions.

6 Running the simulation

The simulation automatically checks the file DetectorSetup.mac to configure the geometry for the run. In addition the user can supply the name of a run macro for running in batch mode, this includes information on the physics models used as well as the event generator. Alternatively the simulation can be run in interactive mode, see Sec. 6.4.

6.1 Detector Setup

Is controlled via the macros/DetectorSetup.mac file. Users can edit this to built any subset of detector systems, by switching the 1's to 0's. By default :

/A2/det/useCB 1
/A2/det/useTAPS 1
Taps can be configured for its original or 2007 setup.
position and ids
/A2/det/setTAPSFile taps.dat
distance from centre of the ball
/A2/det/setTAPSZ 175 cm
number of active crystals
/A2/det/setTAPSN 510
select PID1 (1) or PID2 (2) or no PID (0)
/A2/det/usePID 1
apply a z offset
/A2/det/setPIDZ 0. cm
add in the wire chambers 1 or not 0
/A2/det/useMWPC 1
The setup of the target is also done here :
/A2/det/useTarget Solid (or Cryo)
/A2/det/targetMaterial G4_Pb (or G4_Ca, ...)

6.2 Physics List

A brief discussion on the physics models is given in Sec. 5. Selection of the physics list must be done pre-initialisation,

Choose a physics list, e.g. QGSP_BIC, QGSP_BERT_EMV, LHEP...
/A2/physics/Physics QGSP_BIC
Initialise the simulation
/run/initialize

NOTE for interactice running this commands are called in the default macro vis.mac that is run on initialisation.

6.3 mkin Input

Users should obtain an mkin event generator file containing the reaction particles of interest. This file should be converted to ROOT via the h2root utility. 2

examples of such files are available alongside this release, kin_pi0p_100000.root and mkin_pin_400mev.root. The setup of A2 to read such files can be done either in batch or interactive mode using the same commands. Using the reaction $\gamma p \rightarrow p\pi^0$ as an example (e.g. kin_pi0p_100000.root), the command sequence follows :

The number of particles to be tracked, 1 proton 2 photons
```
/A2/generator/NToBeTracked 3
```
The mkin indexes of the particles to be tracked
The proton
```
/A2/generator/Track 2
```
The 2 photons
```
/A2/generator/Track 3
/A2/generator/Track 4
```
Note, the π^0 will not be tracked as it's index 1, is not given.
The mkin file, converted to root, to be read
```
/A2/generator/InputFile kin_pi0p_100000.root
```
The particles in the file and those to be tracked will be output to the screen:
```
A2PrimaryGeneratorAction::SetUpRootInput You have chosen to track 3 generated particles.
A2PrimaryGeneratorAction::SetUpRootInput, adding a pi0 as index 1
A2PrimaryGeneratorAction::SetUpRootInput, adding a proton as index 2
A2PrimaryGeneratorAction::SetUpRootInput, adding a gamma as index 3
A2PrimaryGeneratorAction::SetUpRootInput, adding a gamma as index 4
Will track proton with index 2
Will track gamma with index 3
Will track gamma with index 4
```

NOTE if AqcuMC is used as the event generator it must store the output in ntuple format not Tree format.

6.4 Interactive Mode

This allows visualisation of the detectors and supports standard G4 gun/ commands to define the beam or can read an mkin generated file. To run in this mode simply type A2 from the A2 directory. Any detector set in the DetectorSetup.mac is drawn in the World volume. The initial visualisation setup is defined in macros/vis.mac. Currently I prefer the OGLIX style viewer, which requires command line or macro based commands. When using version 4.8.2, this viewer can be used to make movies. Additionally the OGLIXm viewer (comment in line /vis/open OGLIXm) allows changes in the picture using a GUI system, although this can crash if too many commands are given.

The DAWN viewer is best for detailed pictures but requires the downloading of additional software. Users are encouraged to try the many other viewers available.

To run in interactive mode users should just type A2 from their G4WORKDIR. The physics models can be changed in the macros/vis.mac file, as can the default visualisation settings.

To run an mkin event follow the commands in sec. 6.3. Alternatively define a beam using the gun/commands :

/gun/particle proton
/gun/energy 100 MeV
/gun/direction 1 0.1 0

With either method then type /run/beamOn N to run N events. The tracks will be drawn on detector picture. In the case of the ball the crystals hit will be coloured with an intensity depending on the energy deposit. Alternatively the command /A2/event/drawHitOpt option will change the colour coding to depend on either the (option=)time or (option=)depth of the hit. This command has to be run prior to the event. In the case of TAPS and the PID currently the whole detector will be coloured if any element is hit.

For a list of all possible commands users can type help at the Idle> prompt. This includes all standard G4 commands for visualisation, runs, events, geometry... As well as the new commands for A2, found in the A2 help directory.

An output file can be writtein using the command :

/A2/event/setOutputFile test.root

6.5 Phase Space Mode

The A2 simulation can be used to directly produce tracked phase space distributions of particles. This can in turn be used to calculate detection efficiencys, energy loss corrections, angular resolutions etc. The following commands are used :

use phase space mode (1) (0 is command line 6.4, 2 is ROOT input 6.3 which is done outomatically if you specify an input file)

/A2/generator/Mode 1

max and min kinetic energies

/A2/generator/SetTMax 300 MeV
/A2/generator/SetTMin 0 MeV

max and min angle

/A2/generator/SetThetaMax 180 deg
/A2/generator/SetThetaMin 0 deg

for the vertex position

gamma beam widths at target

/A2/generator/SetBeamXSigma 0.05 mm
/A2/generator/SetBeamYSigma 0.05 mm

target Z offset

/A2/generator/SetTargetZ0 0 mm

target thickness

/A2/generator/SetTargetThick 0.1 cm

target x-y radius

/A2/generator/SetTargetRadius 2 cm

The kinematics of the particle are stored for each event in the beam branch of the ROOT output ntuple.

See also doPhaseSpace.mac.

6.6 Batch Mode

Running in batch mode is straightforward, the executable is given a series of commands by a macro file which are implemented before the run. Currently this will contain the mkin setup commands from sec. 6.3, and an output file to write the tracked ntuple. The latter is done via the command :

/A2/event/setOutputFile test.root

For example to track the mkin input file kin_pi0p_100000.root, change the path to this file in macros/doppi0.mac to wherever you have saved the file and give a suitable output file name. The from the A2 directory type :

A2 macros/doppi0.mac

A large amount of set up info will scroll down the screen giving info on the geometry, physics processes etc. If the output file does not exist the simulation will start tracking. If it does exist the user will be prompted to overwrite or supply a new file name.

On my laptop the example above takes around 2 hours to run.

The output file can then be read into AcquRoot as in the case of the cbsim output.

6.7 Output

Currently 3 extra branches have been added to the ROOT output. These are NaI crystal time, TAPS veto ID and TAPS veto energy.

Branches from cbsim that exist but are not filled are eleak, and etapfs.

To run the output file through the AcquRoot reconstruction analysis the command should be :

AcquRoot –offline CBMC.Offline

Where CBMC.Offline is a setup file found in your $acqu/data directory. The name of the G4 output file should be specified in CBMC.Offline. The correct tree of setup files should be chosen for MAMI-B and MAMI-C beamtimes. Examples are given in the A2.acquroot directory

7 Things to be done...

- Include the wire chambers for MAMI C.
- Wire chamber readout.
- TAPS fast and slow gates.
- Optimise tracking algorithm via particle threshold cuts and G4Regions.
- Investigation of different physics processes.

- Validation exercises, e.g. particle detection efficiencies, neutron energy deposits...