TPPTbuilder description

1. Introduction

TPPTbuilder is a part of the TPPT software framework, developed at the LIP-Coimbra group in the frame of the TPPT collaboration. The framework consists of the TPPTsim (simulator), TPPTbuilder (event builder) and TPPTcoinc (coincidence sorter). The main purpose of TPPTbuilder is to define scintillation events, characterized by the time mark and energy, based on the data for energy deposition in the detector's scintillators obtained in TPPTsim using Geant4 simulations. In the next step, these scintillation events are processed by the TPPTcoinc software to find coincidences, which, in turn, are the input data for the image reconstruction.

Geant4 simulations tend to produce highly-fractured deposition data: for example, a single particle can have several deposition nodes as well as create several delta electrons, which, in turn, make independent energy depositions. Therefore, TPPTbuilder must be able to cluster several depositions in the same scintillator to a single event if the time difference between the depositions is small enough. The clustering is made taking into account the properties of the PETsys data acquisition system, such as the integration and dead times. As the TPPTsim output gives energy deposition while the experimental system collects the signals of the optical sensors (proportional to the number of the detected optical photons), TPPTbuilder can also introduce the ”blurring” of the event energy by accounting of the intrinsic energy resolution of the scintillators. Also, it is possible to define a minimum event energy, below which an event is ignored, to reproduce the behavior of the experimental system.

Another function of the TPPTbuilder is to “simulate” the accuracy of the read-out system in attributing a time mark to an event, which for the case of the TPPT detection system results in coincidence time resolution (CTR) of about 200 ps FWHM. In order to take this effect into account, the time of each event can be blurred using a Gaussian distribution with a given sigma.

2. Configuration

The simplest way to configure TPPTbuilder is to edit the settings directly in the builder.cc file. The configuration section can be easily identified as it starts with

```
// --- Start of user inits ---
```

and ends with

```
// --- End of user inits
```

comments. An example of a configuration is given below, with the variables explained in the following text.

```
Config.WorkingDirectory = "/home/user/TMP/";
Config.BinaryInput = true;  Config.InputFileNames = {"SimOutput1e6.bin"};
Config.BinaryOutput = true; Config.OutputFileName = "BuilderOutput1e6.bin";
Config.ClusterTime = 0.1; // ns
Config.IntegrationTime = 40.0; // ns
```

```
Config.DeadTime = 100.0; // ns
Config.CTR = 0.2; // ns
Config.Seed = 100;
Config.EnergyResolution = 0.13; // ratio of FWHM and mean
Config.EnergyThreshold = 0.010; // MeV
Config.TimeRanges = { {0, 1e50} };  // ns
```

- WorkingDirectory gives the path where the input file(s) are located. The output file will also be created in this directory.
- BinaryInput / BinaryOutput flags set to true signify that the input / output files are binary, otherwise they are in the ASCII format
- InputFileNames is an array of input files (e.g. {“file1.dat”, “file2.dat”, “file3.dat”})
- OutputFileName is the output file name
- ClusterTime defines the maximum time in ns for two depositions to be considered belonging to the same event (see 3.2)
- IntegrationTime gives the time in ns during which the deposition clusters are considered to belong to the same event (see 3.3)
- DeadTime defines the time (in ns) after the beginning of the first cluster of an event when the detector becomes sensitive again (see 3.3)
- CTR is the coincidence time resolution (in ns, FWHM)
- Seed is the seed of the random number generator
- EnergyResolution is the intrinsic energy resolution of the scintillator. It is given as the ratio of the FWHM of the photopeak to the mean of the peak.
- EnergyThreshold is the energy threshold (in MeV) below which events are discarded
- TimeRanges is an advanced feature allowing to speed up processing of the data. The default value of { {0, 1e50} } should be used unless very large datasets are to be processed. It allows processing of events in bunches defined by the {timeFrom, timeTo} time ranges (in ns). Also, if the data acquisition windows are fixed and known at this stage, the data processing could be limited to these windows.

Note that each run of the TPPTbuilder creates a configuration file BuilderConfig.json in the working directory, which lists all the parameter values (JSON format). Such files provide an alternative way to configure TPPTbuilder: the parameter values could be read from the file. To use this configuration mode, provide the file name as the first argument when starting the TPPTbuilder executable. In this case the values given directly in the builder.cc have no effect.

3. Workflow details

The workflow consists of four phases: (1) load deposition data from a binary or ascii file; (2) cluster energy deposition nodes; (3) build events; (4) save events to a binary or ascii file.

3.1 Loading of the deposition data

Depending on the configuration of the simulations performed in TPPTsim, the input file(s) are either in ASCII or binary format. The files list the energy deposition records (pairs of time[ns] and energy[MeV]), grouped by the scintillator index. Note that the same scintillator index can appear multiple times in the same file.

For the ASCII files the format is the following:

```
# ScintIndex
Time_1 Energy_1
Time_2 Energy_2
…
Time_n Energy_n
# ScintIndex+1
Time_1 Energy_1
Time_2 Energy_2
…
Time_m Energy_m
…
```

Each line is terminated with '\n' (end of line) symbol. The '#' symbol marks the lines which contain the scintillator index (ScintIndex). Space character separates # symbol and the scintillator index, as well as the time mark and the energy values.

For the binary files the format is the following:

```
0xEE ScintIndex(int)
0xFF Time_1(double) Energy_1(double)
0xFF Time_2(double) Energy_2(double)
…
0xFF Time_n(double) Energy_n(double)
0xEE ScintIndex+1(int)
0xFF Time_1(double) Energy_1(double)
0xFF Time_2(double) Energy_2(double)
…
0xFF Time_m(double) Energy_m(double)
…
```

Note that 0xEE and 0xFF chars are used to indicate the type of the following record: scintillator index or time/energy record. There are no other delimiters.

3.2 Clustering of the energy deposition nodes

After the loading phase, the energy deposition nodes (pairs of time and energy) are stored separately for each scintillator. Independently for each scintillator, the clustering procedures

operate over the list of nodes, and if two nodes are found with the time difference between them less than the ClusterTime value, the nodes are merged into one. The merging procedure for (Time1,Energy1) and (Time2,Energy2) nodes is the following:

Energy = Energy1 + Energy2
Time = (Time1*Energy1 + Time2*Energy2) / Energy

The clustering is performed in two phases. The first one (pre-clustering) operates directly on the loaded data, resulting in merging of depositions of “local” groups of nodes generated by the same particle, and thus consecutive in time.

As the time mark of the tracked particles in Geant4 can change from one to another, a sorting procedure is then applied to the pre-clustered data to form a list of nodes consecutive in time. Finally, the last step of the clustering procedure is conducted over this dataset. It is performed in iterations, passing the event dataset from the beginning to the end at each of them and merging the nodes. The process is stopped when during one iteration no merges were performed.

3.3. Building of events

The event building is performed in three phases. The first phase is grouping of the deposition clusters (defined in 3.2) to events. The grouping procedure, performed for each scintillator separately, is the following:

The first deposition cluster defines the time stamp of the first event. All the following clusters, which have the time value different from the one of the first cluster by less than the IntegrationTime, add deposition energy to the event energy. This procedure “imitates” the PETsys data acquisition approach: the detection time of the first photons defines the time stamp of the event, and the event’s energy is obtained by integration of the signal from the sensor during a certain time. The following clusters are ignored while the difference of their time value and the previous event is less than the DeadTime. After that a new event is started using the same procedure.

The second phase introduces the energy blurring due to the intrinsic energy resolution of the scintillators. For each event, the event energy is replaced with a value sampled from the Gaussian distribution with the mean value equal to the energy, and sigma = energy * EnergyResolution / 2.355.

The third phase introduces the blurring of the time to simulate the CTR of the system. For each event, the event time is replaced with a value sampled from the Gaussian distribution with the mean value equal to the time and sigma = CTR / 2.355 / sqrt(2.0). Square root of two is introduced to give the uncertainty in time difference for a pair of depositions equal to CTR.

3.4 Saving the event data

Depending on the configuration flag BinaryOutput, the event data are saved in an ASCII or binary file. The format is exactly the same as the one used for the input files: the pairs of time and energy giving the events for each scintillator. Note that each scintillator index appears only once in the output file.

For testing purposes, two additional ASCII files are generated: Builder-Energy.txt and Builder-Time.txt. The files provide distributions of the event energy and time: the first one contains pairs of energy(MeV) and the corresponding number of events; the second one contains pairs of event time(ns) and the corresponding number of events. The lines are '\n' (end-of-line) terminated and a space character is used as the delimiter. The histogram binning and ranges are defined in Write.cc file (see the constructor of the Writer class), e.g.:

```
histEnergy = new Hist1D(1000, 0, 1.0);    // [MeV]
histTime   = new Hist1D(100,  0, 2e+12);  // [ns]
```

The first argument is the number of bins, while the second and the third ones define the histogram range (from and to, respectively).