A guide to using AraSim

Guy Nir
Weizmann Institute, Rehovot

guy.nir@weizmann.ac.il

Based on AraSim software developed at OSU
Special thanks to
Amy Connolly, Carl Pfendner and Eugene Hong
for writing this software
(and the having patience to explain everything to me).
Updated by Brian Clark October 2017.

October 31, 2017
svn: revision 738

Abstract

There are a lot of options to use in AraSim, most of which are not fully documented. I will try to make some short, clear documentation on these functions. I will also try to make clear which options cannot work together and how to get ceratin results by choosing the right parameters.

Along with this .pdf document is a settings file 'setup.txt' which contains the 'references' section of this document (section 6 on page 22). In it you may find the default values and a quick summary of the different modes for each parameter.

1 Compiling Arasim

1.1 Prerequisites

AraSim has a few prerequisists:

- ⋆ sqlite3: useful for reading geometry files. It is important to have this installed and enabled when installing ROOT.
- ⋆ ROOT: you should already be running root (and, if necessary, have `ROOTSYS` set in your environmental variables).
- ⋆ Boost: a bunch of libraries for doing things faster in c++:
- ⋆ AraRoot: for the data structures and geometry used in ARA analysis.

1. go to the boost website `www.boost.org` and download the tarball. Untar it to get the source directory.
2. inside the directory '`boost_x_xx_x/`' do
   `$ ./bootstrap.sh` and then `$ sudo ./bjam install`
   if you are installing boost on a server/cluster (and can't sudo anything) do
   `$ ./bootstrap.sh --prefix=path/to/installation/directory`
3. Add the environmental variables: add to your .bashrc the line:
   `export BOOST_ROOT=/path/to/boost_x_xx_x/`

1.2 Getting the code

- ⋆ To get the latest version of AraSim go to
  `http://www.physics.ohio-state.edu/~connolly/AraSim/arainstr.html`
- ⋆ or use the svn command[1] (make sure you 'cd' to the place where you want to create the 'AraSim/' directory):
  `$ svn --username <yourusername> co https://`
  `delos.mps.ohio-state.edu/RadioSim/AraSim/ AraSim`
  (write this in one line).
- ⋆ For more instructions regarding the AraSim svn:
  `http://www.physics.ohio-state.edu/~connolly/AraSim/arainstr.html`
- ⋆ To get a username and password contact Amy Connolly at:
  `connolly@physics.osu.edu`

1.3 Compiling AraSim from source

Inside the '`AraSim/`' directory run `$ make`

Along with AraSim we have two useful programs for looking at the resulting 'AraOut.root' file.

1. readTree.cc: this program makes .pdf files with plots of the waveforms inside the output file.
   do `$ make -f M.readTree` to compile it, then run
   `$ ./readTree` or `$ ./readTree outputs/youroutput.root`
2. readGeom.cc: this program plots the geometry of the detector.
   do `$ make -f M.readGeom` to compile, then run
   `$ ./readGeom` or `$ ./readGeom outputs/youroutput.root`

If any problems arise try the following:

- ⋆ Check the prerequisists and the necessary environmental variables:
  `$ echo $ROOTSYS` should give the directory of ROOT (this may not work in fixed location installation method of ROOT).
  `$ echo $BOOST_ROOT` should point to the boost directory. Make sure the distribution number is right (latest is '`boost_1_54_0`' as of this article).
- ⋆ If all else fails: contact Eugene Hong (`ripple90@gmail.com`), Carl Pfendner (`pfendner.1@physics.osu.edu`) or Amy Connolly.

[1] You may have to do $ sudo apt-get install subversion

Figure 1: What to do in case you have any trouble with AraSim

2 Using Arasim

2.1 Basic implementation

Once AraSim is compiled we can run it by doing `$ ./AraSim` which runs the simulation using the default setup file 'setup.txt'. If this file has not been changed AraSim should run 100 neutrino events (not a very long simulation).

The primary output of AraSim is through an output file found in 'outputs/AraOut.root'. This contains three trees:

- ⋆ AraTree: contains the global parameters of the run, such as the detector setup, the neutrino energy spectrum and the earth and ice models used in the simulation.
- ⋆ AraTree2: contains detailed information for each seperate event, such as the position of the interaction vertex, the energy of each simulated neutrino, and lots of other information on the specfic event.
- ⋆ eventTree: This tree contains the data that would be available in real analysis. It simulates the data structures we would get from a real detector. It should have the waveforms as they are picked up by the antennas, and some other information measureable by the detector.

Depending on the data saving mode, these trees may contain more or less data. By default the 'AraTree2' should contain all events generated while 'eventTree' should have data only for globally triggered events.

2.2 Advanced users

After making sure AraSim works well, we should customize the simulation to our own needs. The simplest way to do this is to change parameters inside 'setup.txt'. Once changes are made to this file a new simulation can be run, with the output file being overwritten by the new 'AraOut.root' file. For more explanations on what parameters we can change in a setup file, see section 3.

For serious use of AraSim, we strongly recommend to generate renamed copies of 'setup.txt', for example 'setup.example.txt'. Running `$ ./AraSim setup.example.txt` will use the settings inside this new file.

As a practical example, we may have 'setup.TB.txt' and 'setup.ARA37.txt' which we run one after another, the first with testbed parameters, and the second with a large detector. This lets us keep several sets of our favourite parameters without needing to constantly comment out and rewrite values.

If we don't want our output files to overwrite each time we run AraSim, we should specify as inputs for AraSim not just the setup file, but also a run number. For example:

```
$ ./AraSim setup.TB.txt 3
```

will run AraSim using the testbed parameters inside 'setup.TB.txt' and save the output to 'outputs/AraOut.setup.TB.txt.run3.root'.

We may use this functionality for running AraSim as a real detector, which generates several runs (it will also allow us to start analysis on the first files before all the simulations are done). Instead of running one simulation with 10^6 neutrinos we may run ten simulations of 10^5 neutrinos each, and get ten output files:

```
$ for i in `seq 1 10` ; do ./AraSim setup.txt $i ; done
```

should generate ten files 'outputs/AraOut.setup.txt.#.root', that can be loaded into a chain just like real detector data.

Another useful possiblity is to run simulation on several setup files, saving the output to different output files without necessarily changing the run number:

```
$ ./AraSim setup.TB.txt 0  $ ./AraSim setup.ARA37.txt 0
```

will give us two different outputs, one for each detector setup, both at run number 0.

For setup files that are inside subfolders or on a completely different path, AraSim will truncate the path to the setup file when writing the output files, e.g.:

`$ ./AraSim SETUP/parameter/setup.par.txt 0` will create output file:
`AraOut.setup.par.txt.run0.root` and not the entire path.

If you wish to specify the output directory for AraSim, simply add the path after the run number:
`./AraSim setup.txt 0 /some/other/path/to/store/outputs/`
This is especially useful when running large data sets that do not fit in a user's home directory.

2.3 Super users

For running massive simulations we recommend setting up AraSim on a cluster, where we can send multiple jobs to run. This allows us to run multiple simulations at once, freeing our computer and our time to write user guides for simulation programs.

Depending on the super computer / cluster used we recommend using some queue submission e.g.

```
for i in `seq 0 99`; do qsub -v var=$i ./run_sim.sh
```

(Additional options may include `-N job_name` , `-k oe` (error and output redirection), and `-q which_queue` depending on the system).

The script `run_sim.sh` can do something like:

```
#!/bin/bash  //  ./AraSim setupfile $var
```

along with additional scripts to redirect outputs to logfiles, move the output files etc.

Make sure you ask your local cluster admin for instructions on how to submit multiple jops to the grid.

In any case, the outputs of running AraSim on several machines will all funnel back into the `outputs/` directory, unless you specified a different output (i.e. some hard drive that can store large data sets instead of the home directory). If you do multiple runs with a distinct setup file and run number you can gather them all up at the end (or even after a few finished runs) and start analysis.

2.4 Data Analysis

Analizing the data from AraSim output files divides into two categories. There is the 'behind the scenes' information such as the position of the interaction vertex and the energy of the incoming neutrino, and there is the 'real data' which mimicks the output of a real detector (waveforms primarily).

As explained in section 2.1 there are two trees for 'behind the scenes' information. Most the information in 'AraTree' is stored in the Detector and Settings classes. The specific events (kept in 'AraTree2') are accessible via the Event and Report classes.

The detector style information is saved in 'eventTree', using classes from AraRoot (version 3.13), i.e. 'UsefulIcrrStationEvent'.

If you have analysis code that depends on the AraSim library, you may wish to generate a dynamically loaded library that knows all the classes used by AraSim. Use `./library.sh` to create `libSim.so` . You can now compile any code with the AraSim headers, and link to this library. Make sure it is in your `LD_LIBRARY_PATH`

example:

```
g++ example_loop.cxx -I$SIM -I$SIM/AraRoot -I$SIM/AraRootFormat //
-L$SIM -lSim `root-config --cflags --libs` -o program.exe
```

Where `$SIM` is the directory of AraSim. This program can have access to the trees inside the AraSim output files, reading geometry, event physics and the actual waveforms similarly to the way we access L0 files in AraRoot (with Iccr rather than the newer Atri board). Make sure you actually write the waveforms to file (see section 3.10).

3 Simulation Modes and Parameters

Each simulation is made using the parameters found in 'setup.txt'[2] . Although some explanations are found in comments within the file itself, and some are self-explanatory, a more detailed explanation for each parameter is given below.

These parameters are read into the `Settings` class. If any field is commented out or removed from the setup file, default values are used (as defined in 'Settings.cc').

You may comment out any part of the 'setup.txt' file using '`//`'.

Adding lines like this: '`PARAMETER` = N' will set the appropriate parameter to N. The order of input lines is irrelevant, but if two or more lines set the same parameter, the last one will be used.

3.1 Neutrino energies

♠ `EXPONENT`=19. This determines the energy spectrum of the incoming neutrinos.
For selecting a single energy choose values `EXPONENT` $= N$. When $10 < N < 20$, AraSim will generate neutrinos with energies $E = 10^N$eV. Taking values too low or too high may lead to strange results in the simulation (The ARA detector should work well for neutrinos at energies of $E \sim 10^{15} - 10^{20}$eV only).
In order to simulate real spectra of neutrinos, so each neutrino energy is randomly chosen from a distribution of energies, use `EXPONENT` $= N$ with values $1 < N < 4$, or $N \geq 30$.
For $1 < N < 4$ the distribution is just E^{-N}.
If you wish to give a more specific energy value to all neutrinos, use $510 \leq$ `EXPONENT` ≤ 650, AraSim will give an energy of 10^x eV to all neutrinos, where $x =$ (`EXPONENT` $- 400)/10$. For example, to get $E = 10^{18.3}$ eV neutrinos, use `EXPONENT` $= 583$.

To generate neutrinos at random energies from specific distributions (flux models) use $30 \leq$ `EXPONENT` ≤ 500. The specific choice of flux model can be examined in 'Spectra.cc', with some of the flux model files stored in `fluxes/` .

[2] As explained in section 2.3 it is advisable to make differently named copies of 'setup.txt' and run '$ ./AraSim yoursetup.txt'

3.2 The number of neutrinos simulated

♠ `ONLY_PASSED_EVENTS`=0. Choosing the number of neutrinos generated in the simulation can be made in two ways. If we choose `ONLY_PASSED_EVENTS` = 0 then we can specify how many ν's to generate in the simulation. Thus the total time of simulation is known *a-priory* but the number of ν's globally triggered can vary (can even be zero).

♠ `NNU`=100. In this case (`ONLY_PASSED_EVENTS` = 0) the number of neutrinos generated should be specified in `NNU` = *(# of neutrinos)*.

The second option is `ONLY_PASSED_EVENTS` = 1, where we specify how many ν's pass the trigger. AraSim will generate as many ν's as necessary until the requested number is reached. In this mode the size of the sample that will trigger is known in advance but the run time may vary. If reasonable parameters are not chosen, the run time may be very long.

♠ `NNU_PASSED`=0. When using this mode (`ONLY_PASSED_EVENTS` =1), AraSim will ignore the `NNU` field and look instead at `NNU_PASSED` . This field determines how many neutrinos will globally trigger.

♠ `OUTPUT_TDR_GRAPH`=0. Saves this number of tunnel diode response (TDR) waveforms for debugging.

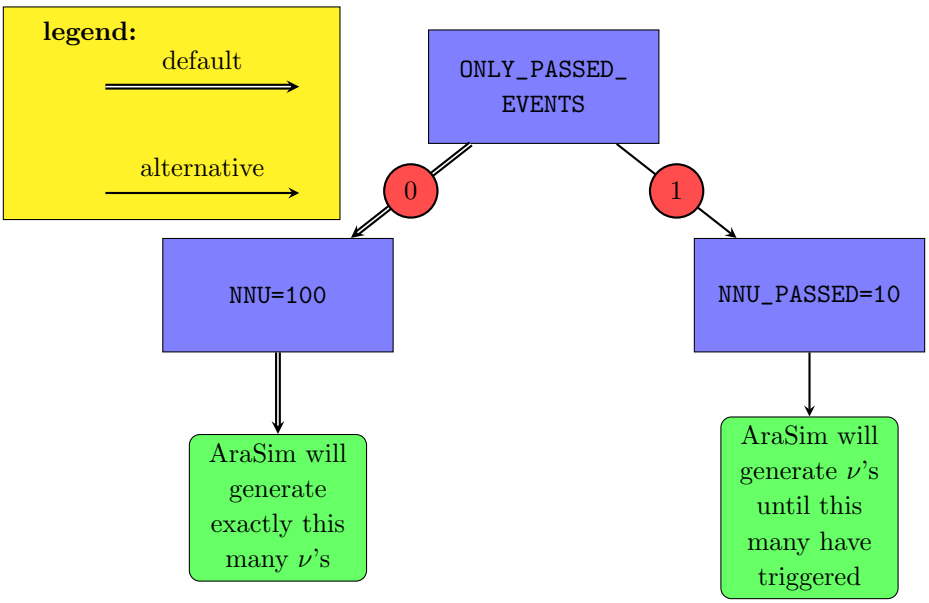

Figure 2: Decision on number of neutrinos to generate

3.3 Generating noise

♠ `NOISE_EVENTS`=16. The number of noise waveforms to be generated as background for the signal.

♠ `NOISE_WAVEFORM_GENERATE_MODE`=0. In the old versions of AraSim, noise waveforms were generated before the simulation. Now we have as default `NOISE_WAVEFORM_GENERATE_MODE` = 0, which generates new noise waveforms for each event. In this mode we need to specify the number of noise waveforms used for each event, so that `NOISE_EVENTS` should be set to the number of channels used in each station. More noise events are not used. If less `NOISE_EVENTS` are generated then they are recycled within an event (not good for us).
If you really want to work in the old methods choose `NOISE_WAVEFORM_GENERATE_MODE` = 1 and choose `NOISE_EVENTS` to be a large number, enough to provide AraSim with enough noise events to use when generating all events.

♠ `NOISE`=0 (default) generates waveforms from a flat thermal distribution. mode 1 uses calibrated Rayleigh distribution noise from testbed data (only used for lower 8 channels - see triggering 3.7).

♠ `NOISE_TEMP_MODE`=0 (default) generate the same noise waveforms for all channels. When using `NOISE` = 1 (Rayleigh distribution), we can choose several modes: mode 1 generate different noise temperatures for each channel; mode 2 generates different noise for the first 8 channels (borehole antennas) and the rest are all the same.

♠ `NOISE_TEMP`=325. If the noise is generated from a thermal distribution (`NOISE` = 0), then the temperature of the noise is determined by `NOISE_TEMP` (default is 325) given in Kelvins. This value is composed of the ice temperature (T_{ice} = 230K) added to the receiver noise (T_{rec} = 95K).

♠ `DATA_BIN_SIZE`=16384. This must be a power of 2. This field is the length of the noise waveforms that are generated before simulation starts. It is used for the old method where all noise waveforms are pregenerated (in `NOISE_WAVEFORM_GENERATE_MODE` = 1). *Also* we use this size in the new mode only to set the *noise floor* by generating a number of noise waveforms of this length and calculating their RMS.

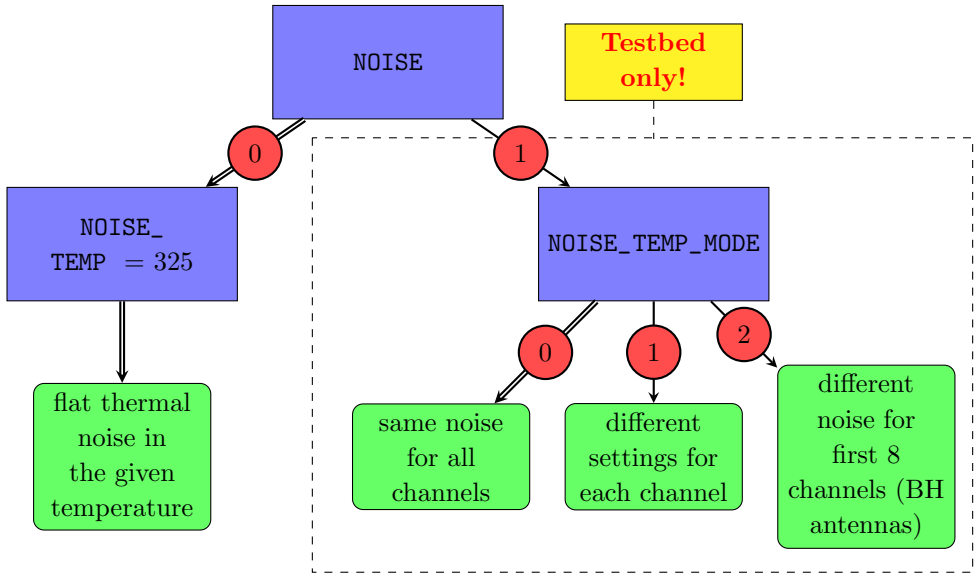

Figure 3: Decision on what noise to generate

3.4 Position of simulated neutrino

♠ `INTERACTION_MODE`=1. AraSim generates a random position for the interaction of the neutrino based on the choice made for `INTERACTION_MODE` :

⋆ **Aeff mode (spherical volume)** (mode 0): choose neutrino location in sphere around detector, but only inside ice. Makes an unbiased sample of events so we can get the effective area directly. This automatically chooses the new `GET_CHORD` mode. Make sure you set `POSNU_RADIUS` to 5km (or more) for this mode.

This mode has the advantage that it already takes into account the survival probability of neutrinos passing through the earth.

♠ `POSNU_RADIUS`=3000, in meters. Determines the radius around the detector in which the neutrinos are generated.

Make sure you choose a reasonable radius for the position of the neutrino interaction. If you choose a large detector, for example ARA37, make sure you increase the size of `POSNU_RADIUS` to include the detector and the area around it (for ARA37 consider using ∼13000)

⋆ **Veff mode (cylindrical volume)** (mode 1, default): Chooses ν interaction position close by to the detector, within a cylinder of radius R. In this mode we get the effective volume. This uses the old `GET_CHORD` function.

♠ `PICK_POSNU_DEPTH`=0 (default) takes all the ice down to the bedrock (according to the earth model used). mode 1 allows the user to choose a maximum depth for the position of the neutrinos.

♠ `MAX_POSNU_DEPTH`=200 in meters depth. When the above is set to mode 1, use this to determine the maximum depth of the position of the generated neutrino.

⋆ **Pick exact** (mode 2): this chooses the position of all ν's to be at the same location.

The position of interaction is built into AraSim at this time (look inside 'Primaries.cc' for `INTERACTION_MODE` ==2).

The default is at $R = 1000$m, $\theta = -\pi/4$ and $\phi = -\pi/6$, which translates into

$$x = 353.55\text{m} \quad y = 612.37\text{m} \quad z = 707.1\text{m}$$

relative to the station center. This mode is useful for testing reconstruction at a specific location, or for making sure all events have a ray trace solution (see subsection 3.5).

♠ `POSNU_THETA` interaction location elevation angle in radians

♠ `POSNU_PHI` interaction location azimuthal angle in radians

♠ `POSNU_R` interaction location radius in meters

⋆ **Above Ice** (mode 4): Throws events in a cylinder above the ice. `PICK_ABOVE_HEIGHT` =3000m (default) sets the height of the cylinder in which events will be thrown. Meant to be used with the arbitrary event generator mode `EVENT_TYPE` =10 (see section on "other parameters").

♠ `NNU_THIS_THETA`=0 (default) chooses randomly any angle within the range $\theta \in [0, \pi]$. mode 1 lets you choose a specific value for θ and some range around that angle.

♠ `NNU_THETA`=0.785 in radians (=45°) the angle chosen when using `NNU_THIS_THETA` = 1.

♠ `NNU_D_THETA`=0.0873 in radians (=5°): the variability above and below the chosen θ, when using `NNU_THIS_THETA` = 1.

♠ `SECONDARIES`=1. mode 0: does not allow any more interactions after the first. mode 1 (default): allow secondary interactions.

♠ `TAUDECAY`=1. mode 0: do not allow secondary τ particle decays. mode 1 (default): let τ particles that are generated from ν_τ to also have a secondary decay in the ice. Only works when `SECONDARIES` is set to 1.

3.5 Ray Solving

♠ RAY_TRACE_ICE_MODEL_PARAMS=0 (default). Use a index of refraction model fitted from the RICE data (https://doi.org/10.3189/172756504781829800). =1 is Soth Pole fitted from RICE 2. =2 is South Pole measurement by Eisen 2003. =3 is South Pole measurement by Gow. =10 and 11 are both models for Moore's Bay. =20 is measurements at Byrd station by Ebimuna 1983. =30 is measurement at Mizuho station by Ebimuna 1983.

♠ RAYSOL_RANGE=5000 in meters. Each neutrino event is generated at a certain distance from the detector (see subsection 3.4). Before the event is tested for triggering it is tested for plausible ray trace solution. This is simply done by comparing the distance of the position of the neutrino (POSNU) from the station center to a radius given by RAYSOL_RANGE . If the neutrino is generated outside of RAYSOL_RANGE then AraSim will not test the trigger and just assume there is no ray solution.
For a single station, make sure this is larger than POSNU_RADIUS in 'pick near', and larger than the default 1000m radius for 'pick exact'.
For multiple stations this test is done for each station seperately, before each station tests for trigger.

3.6 Simulation mode

♠ SIMULATION_MODE=0 (default) uses frequency domain simulations (AVZ parametrized mode in frequency domain). mode 1 uses the new time domain simulation. The new simulation mode is more realistic and takes between 1.5 to 2 times longer. Also, expect less triggers (as the signal gets more dispersed and has a lower chance of triggering enough channels).
for time domain (SIMULATION_MODE =1) only:

♠ SHOWER_MODE=2. Decides on hadronic / EM showers. 0: EM shower only, 1: hadronic shower only. 2 (default): use either EM or HAD shower, depending on signal strength.

♠ SHOWER_STEP=0.001 This decides the physical step size (in meters) for the simulation. Default 1mm is small enough for most applications.

♠ SHOWER_PARAM_MODEL=0 Shower parameters, choose between 0: Jaime's fit, 1: Carl's fit.

♠ OFFCONE_LIMIT=10 (degrees). This limits calculations of radiation more than 10 degrees (default) away from the Cherenkov angle, in order to save calculation time where the emitted radiation is already very weak.

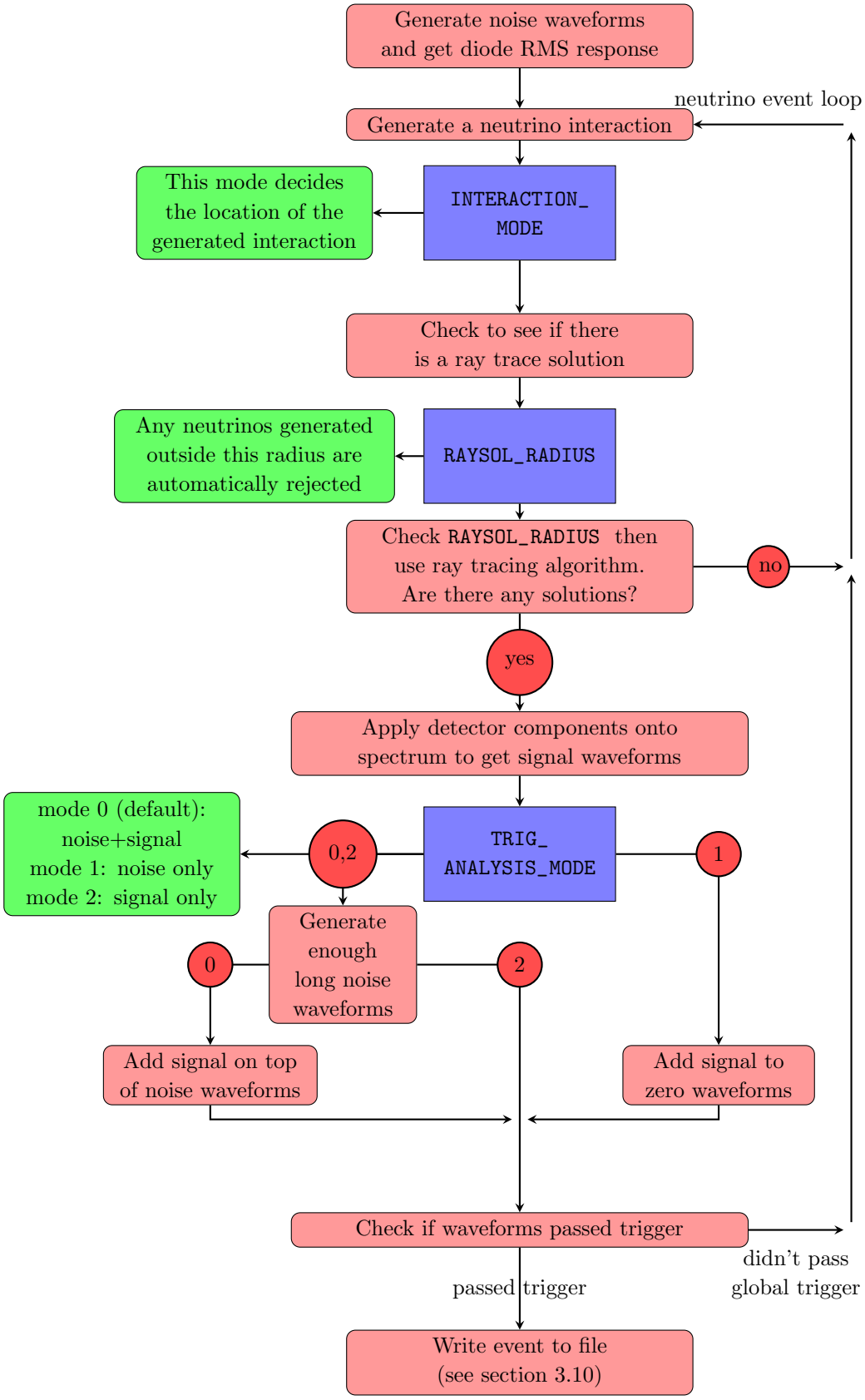

Figure 4: AraSim basic flowchart

3.7 Triggering

♠ `USE_INSTALLED_TRIGGER_SETTINGS`=0 (default) this tells AraSim to use the ideal detector setup for triggering. mode 1 future release.

♠ `TRIG_ANALYSIS_MODE`=0. There are three options to choose:

- ⋆ Signal+Noise (mode 0, default): AraSim will look at noise and signal when deciding if an event triggers or not (this is the realistic mode).
- ⋆ Signal only (mode 1): In this mode the noise is not added to the event waveform. This allows us to see exactly where the peak of the signal is, without any distracting noise. Note: the waveform saved will not have any noise in it (clean signal).
- ⋆ Noise only (mode 2): Here we do not add the signal to the waveforms and check to see if the alone noise passes the trigger. This is useful for simulating min-bias (forced) trigger. Note: the waveforms saved will conatin just noise.

  To use this mode make sure you choose the position of the interaction (see `INTERACTION_MODE` in subsection 3.4) to be close to the station (or use `CALPULSER_ON` > 0 for testbed) so that all events have a ray tracing solution. This ensures every event manages to reach the trigger check (since there is no signal in this mode the distance to the detector is relevant *only* for the ray tracing solution check).

♠ `POWERTHRESHOLD`=-6.15. To determine if an event has passed the trigger, AraSim looks at several parameters. The first is `POWERTHRESHOLD` . This number determines how strong an event is compared to the noise RMS. A value of `POWERTHRESHOLD` = -5 means that a signal needs to be five σ stronger than the noise (five times stronger than the noise RMS). This is always given in negative numbers. The closer you are to zero the weaker the trigger threshold.
Default is `POWERTHRESHOLD` = -6.15 similar to the 100 Hz trigger rate of the actual stations.

♠ `TRIG_WINDOW`=1.1E-7 in seconds (default) is the trigger window time. The default value of 110 nanoseconds is the size of the trigger winow for real stations.
This means that a number of waveforms (exact number is determined by the parameters below) must be above threshold within this time window in order to trigger.

♠ `TRIG_TIMEOUT`=1E-6 in seconds. This is the dead time that the detector needs to reset after an event, to make sure the events are seperate from each other. Default value is 1μs: 1E-6 seconds. This is not yet implemented (it will be useful for secondary interactions).

♠ `TRIG_MODE`=1. Determines the number of channels required for a global trigger:

- ⋆ mode 0: the event triggers whenever at least N channels of any type pass the threshold within the given window. N is determined by the field:
  ♠ `N_TRIG`=3 (default).
- ⋆ `TRIG_MODE` =1 (default): the event triggers by passing either N_V or more Vpol antennas, or N_H or more Hpol antennas, as it does for the current ARA station triggering scheme. The minimal number of antennas that must trigger is determined by
  ♠ `N_TRIG_V` and ♠ `N_TRIG_H` (default is 3 for both).

♠ `TRIG_SCAN_MODE`=0 (default): use old triggering algorithm. mode 1: use new, slightly faster code. mode 2: scan all threshold values and save the results in 'Report' class. This mode takes a little more time, but gives information for each event that triggers what would be the best `POWERTHRESHOLD` it would have triggered on, and how many channels required for it. For example, an event passed the threshold in 3 channels (at `POWERTHRESHOLD` =-6.15) but had large amplitude and could have triggered `POWERTHRESHOLD` =-6.5 in four channels. Useful for scanning low values of `POWERTHRESHOLD` . Information is stored in `TDR_all_sorted` or `TDR_Vpol_sorted` and `TDR_Hpol_sorted` when triggering on Vpol/Hpol seperately. One can then use this information to make powerthreshold cut vs. number of triggers. mode 3: save the single channel triggers and each powerthreshold value at

which the occured. this is saved to `SCT_threshold_pass` in `antenna_r` . Used to get the single channel trigger rate vs powerthreshold.

♠ `TRIG_ONLY_BH_ON`=0 (default) trigger is checked for all channels. mode 1 use only borehole antennas for triggering. Use in **testbed mode only**!

♠ `TRIG_ONLY_LOW_CH_ON`=0 (default) trigger is checked for all channels. mode 1 uses only lower 8 channels (4 Vpol, 4 Hpol). Useful for comparing to testbed, can be used with ideal station. For example use mode 1 with `NOISE_TEMP_MODE` =1, `NOISE` =1, `TRIG_THRESH_MODE` =1 to get the testbed calibrated noise and trigger threshold for lower 8 channels, and trigger on them alone. **do not use in testbed mode**.

♠ `TRIG_THRES_MODE`=0 (default) use the same trigger threshold for all channels. mode 1 gives each channel its own threshold offset from file 'data/thresholdoffset.csv'.

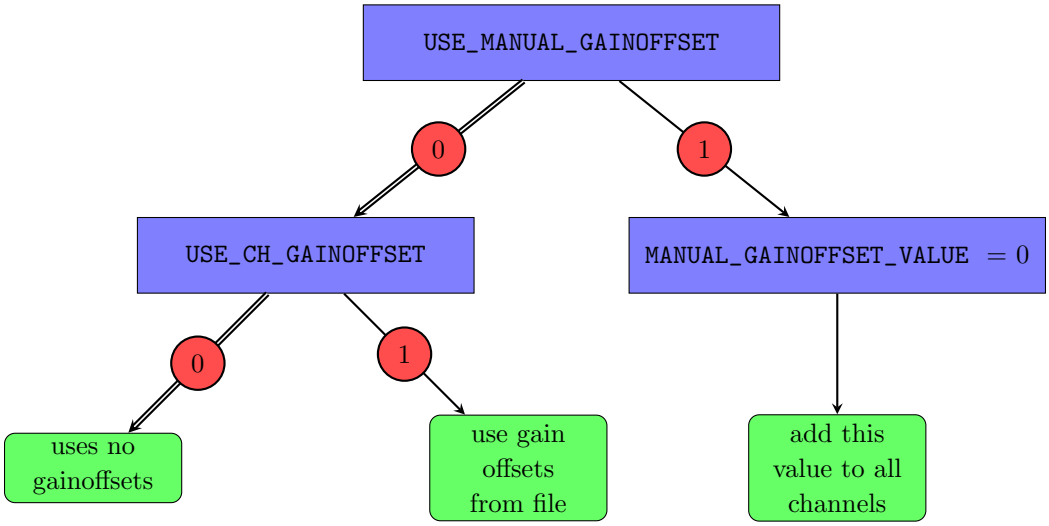

Figure 5: Using specific gain offsets

♠ `USE_MANUAL_GAINOFFSET`=0 (default) this mode passes control of the gain offsets to the `USE_CH_GAINOFFSET` explained below. mode 1 takes one value and adds it to the gain offset for all channels.

♠ `MANUAL_GAINOFFSET_VALUE`=0 is the value added to all antennas when the above `USE_MANUAL_GAINOFFSET` = 1 .

♠ `USE_CH_GAINOFFSET`=0 (default) uses no gain offsets. mode 1 uses specific gain offsets calibrated for testbed from file 'data/preampgainoffset.csv'. Gain is given in volts, so to multiply the power of the waveform by factor $\times a$ we must give an offset of $\times\sqrt{a}$.

♠ `USE_TESTBED_RFCM_ON`=0 (default) doesn't apply any gain changes. mode 1 use the amplification factor in `RFCM_OFFSET` (see below). This is used to cancel the amplification and attenuation caused by all the electronic components between antenna and digitzer.

♠ `RFCM_OFFSET`=80 the amount of dB of amplification we must cancel when using `USE_TESTBED_RFCM_ON` = 0. This is an approximate value (future release will include specific amplifications). **Testbed Only!**

♠ `V_SATURATION`=1 (in Volts). Any points of the digitized waveform that go over 1000mV (default) or below -1000mV will be clipped to that value (i.e. saturation voltage). Note this is given in Volts while the units presented in the waveforms is in mV.

3.8 Ice and Earth models

We can choose the way neutrinos interact when passing through the earth or atmosphere by choosing one of the different earth models. These models affect the weight of the generated neutrinos. A neutrino that goes through lots of earth will have a low chance of survival, and thus lower weight, while a neutrino coming down through the atmosphere will have good survival chances and a high weight. This is used to calculate the total flux of neutrinos and the effective sensitivity of the detector.

♠ `CONSTANTCRUST`=0 (default). This is a left over parameter, do not change this setting.

♠ `CONSTANTICETHICKNESS`=0 (default). This is a left over parameter, do not change this setting.

♠ `FIXEDELEVATION`=0 (default) the elevation is fixed to the thickness of the ice. This is a left over parameter, do not change this setting.

♠ `ATMOSPHERE`=1. mode 0: no atmosphere. mode 1 (default) include atmosphere.

We can also determine the way the radio waves move through the antarctic ice using the different ice models:

♠ `ICE_MODEL`=0. choose 0 (default) for Crust 2.0, or 1 for Bedmap.

♠ `NOFZ`=1. Determines whether the index of refraction changes with depth (mode 1, default) or stays constant (mode 0, not recommended).

♠ `GETCHORD_MODE`=0. Decides how the weights will be calculated for incoming neutrinos, based on their paths and survival chances. 0: (default) use the old code. 1: new code from icemc, which correcly calculates the weight of taus (still needs some testing so it is not default yet).

♠ `taumodes`=1: (default) Tau neutrinos may interact in the bedrock and create a τ that then interacts inside the ice. 0: do not allow these modes.

♠ `MOOREBAY`=0 determines which attenuation length to use: mode 0 (default) for the south pole data, while mode 1 is for Moore's Bay measurements.

3.9 Choosing detector layout

♠ `DETECTOR`: The detector that is simulated in AraSim can be one of three basic types, determined by the parameter `DETECTOR` :

- ⋆ `DETECTOR` mode 0 unused.
- ⋆ `DETECTOR` mode 4: import real station geometry from AraRoot (requires an AraRoot installation. Make sure to choose a `DETECTOR_STATION` .

  ♠ `DETECTOR_STATION`. Must be used with `DETECTOR` =4. Set equal to the station who's geometry you want to import (e.g.`DETECTOR_STATION` =2, =3, etc). Includes the channel mapping to the AraRoot like data structure UsefulAtriStationEvent.
- ⋆ `DETECTOR` mode 1 (default) ideal ARA station(s). Choose a number of stations between 1 and 7, each with an ideal string and antenna layout.

  ♠ `number_of_stations`=1. Determines the number (between 1 and 7) of stations in the array. The stations are added as a small spiral, with the first in the center, and the other six forming a hexagon around it.

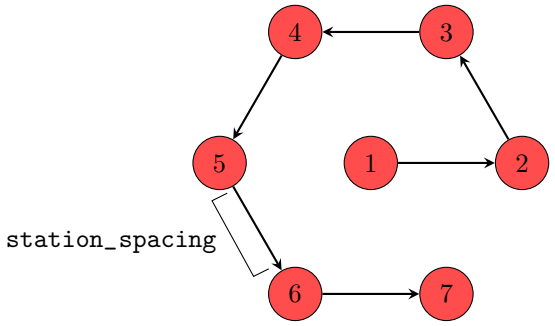

Figure 6: The order at which stations are added in `DETECTOR` = 1 mode

  ♠ `station_spacing`=2000 in meter. Determines the distance between stations in the array (for mode 1 and 2).
- ⋆ `DETECTOR` mode 2: hexagon array of 7 stations and up. Stations are set up in a hexagonal grid.

  ♠ `stations_per_side`=4. Determines how many stations on the side of the hexagon. Default is 4 stations per side, for the ARA37 design. If `stations_per_side` $= N$ then the total number of stations in the hexagonal array will be $3N(N-1)+1$.

There are many things we can change in the layout of the idealized detectors (mode 1 and 2):

♠ `core_x` and ♠ `core_y`: in the global coordinate system this places the center of the array off of the intersection of bins of the earth model[3]. By leaving the default 10000 (in meters) in both `core_x` and `core_y` we guarantee that the detector is close to the pole but not on the intersection.

♠ `number_of_strings_per_station`=4. Determines how many strings for one ideal station.

♠ `R_string`=10 in meters. The radius from the center of the station to the top of the strings.

[3]Each with a $2^\circ \times 2^\circ$ bin size. There are many bins that intersect at the pole itself.

♠ `BORE_HOLE_ANTENNA_LAYOUT` determines the number and ordering of antennas on each string:

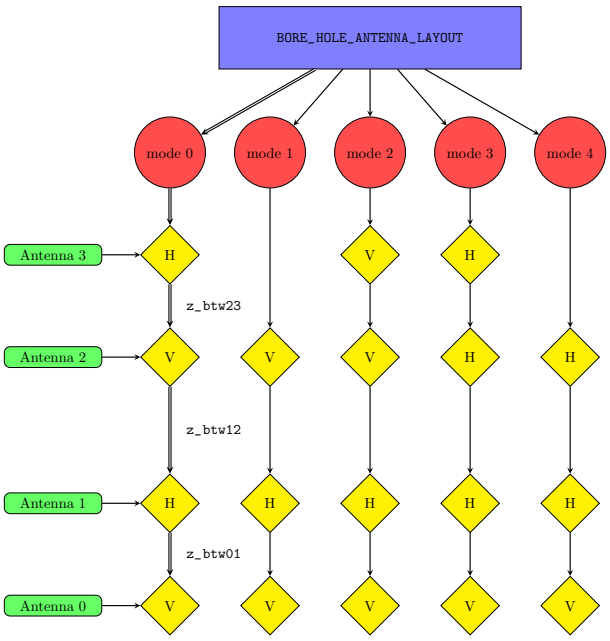

Figure 7: Poisitions of antennas on a string

♠ `z_max`=200, in meters. This sets the depth of the lowest antenna on a string.Positive numbers are used to denote depth.

♠ `BH_ANT_SEP_DIST_ON`=0 the distances between antennas on a string are all the same. mode 1 (default) we get to choose different distances on the string.

♠ `z_btw`=10 (in meters) is the distance between antennas in mode 0.

♠ `z_btw01`=2 (in meters) is the distance between first (lowest) antenna and second in mode 1.

♠ `z_btw12`=15 (in meters) between second and third antenna in mode 1.

♠ `z_btw23`=2 (in meters) between third and fourth (last, if it exists) antenna in mode 1.

⋆ `DETECTOR` mode 3: testbed. This mode simulates the testbed and has several parameters that **can only be used in testbed mode**:

♠ `READGEOM`=0 use idealized detector structure. mode 1 uses the geomtry from sqlite file.

♠ `CALPULSER_ON`=0 has several modes for simulating the calibration pulser.

♢ mode 0 (default) there is no pulser.

♢ mode 1 uses a calibration pulser from 2012, located at $(0.17, -30.01, -17.61)$. This is only an Hpol pulser.

♢ mode 2 uses the Vpol calibration pulser from 2011, at $(36.7, 24.8, -16.1)$

♢ mode 3 uses the Hpol calibration pulser from 2011, at $(34.5, 23.3, -22.1)$

♢ mode 4 uses both Vpol and Hpol, with the location of the pulser at an average of the locations of the real pulsers at $(35.7, 24.1, -19.2)$.

♠ `CALPUL_AMP`=0.15. determines the amplitude of the pulser (this is a manual setting used to match the power of the real pulser. May be useful to change this for different modes of analysis.

♠ `NUM_INSTALLED_STATIONS` may be used in future release to denote the current multi-station detector.

3.10 Writing events to file

For more explanations on the contents of each tree see section 2.1 on page 4.

♠ `DATA_SAVE_MODE`=0. When an event is generated, AraSim will save some data about that event in 'AraTree2', which contains the individual events[4]. What sort of data is stored for each event depends on this parameter:

- ⋆ mode 0 (default): saves all waveforms (these are still spectra since they are generated in Fourier space) at different points in the simulation. The first is one meter from the vertex, the second is after propagation through the ice, and several more are saved for the different ways the electronic components affect the spectra. The final waveforms are recorded if there was a global trigger. If there is no ray trace solution, only the primitive spectra near the vertex are saved. This generates *extremely large output files*.
- ⋆ mode 1: saves only the final waveforms for all events[5].
- ⋆ mode 2: saves only physics data, such as neutrino energy, flavour, position of interaction vertex, and some ray tracing information.

In all modes we will save the physics data for all events that are recorded (which events are recorded to 'AraTree2' is determined by `FILL_TREE_MODE` below).

♠ `DATA_LIKE_OUTPUT`=0. This parameter determines what information is saved to 'eventTree', which is supposed to represent the data which will be available for analysis in a real detector. It contains none of the information AraSim uses internally, only what is measured by the detector. The possible modes are:

- ⋆ mode 0 (default): no data like output.
- ⋆ mode 1 : Output triggered events only.This is the most realistic case in which we only store waveforms for interesting events.
- ⋆ mode 2: Output all events (signal only inserted into data-like output). This saves empty waveforms for non-globally triggered events. Useful for matching the events in 'AraTree2' with those in 'eventTree' but will generate large outputs.

♠ `FILL_TREE_MODE`=0. This parameter decides which events are to be stored in 'AraTree2'.

- ⋆ mode 0 (default): Save all generated neutrino events. When using 'pick unbiased' (`INTERACTION_MODE` = 0) AraSim will save events that occured outside of the ice, although they will not have a ray trace solution and won't trigger.
- ⋆ mode 1: AraSim will not save events outdside the ice, but all other events (events without a ray trace solution) will be saved according to `DATA_SAVE_MODE` . If we are generating events at 'pick near' or 'pick exact' (`INTERACTION_MODE` > 0) then the two modes (0 and 1) are the same.
- ⋆ mode 2: Save only triggered events. In this mode, AraSim will only save to 'AraTree2' events that pass a global trigger. This both reduces output file size drastically, and also makes it easy to match events in these two trees. It does, however, remove much information about events that did not trigger, which are normally saved to 'AraTree2', and may be useful for analysis.

[4] As opposed to 'AraTree' which has the global settings like detector structure and ice models.

[5] It is not certain what is saved in case there is no trigger or ray trace solution. It is probably not very useful data anyway.

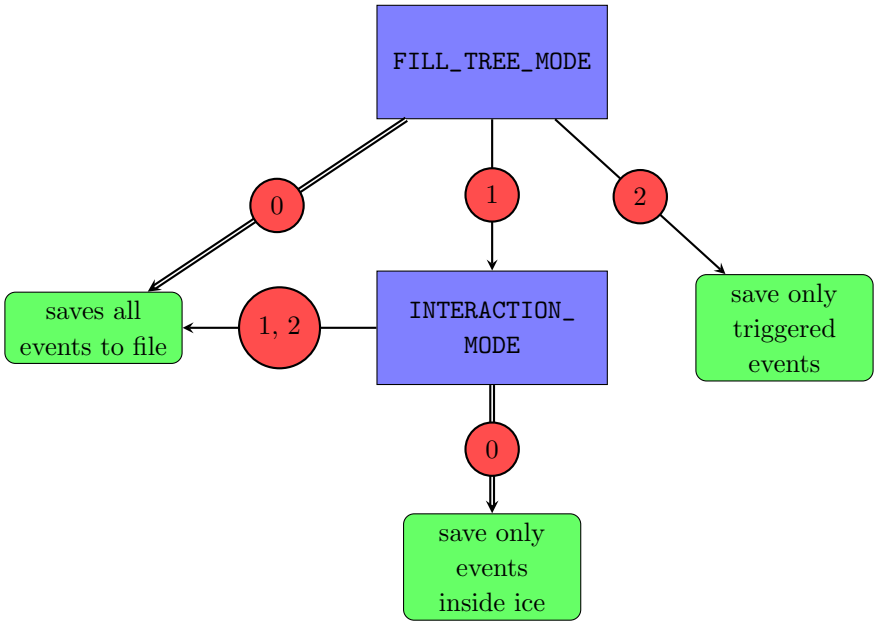

Figure 8: Decision on which events to save

♠ `WAVEFORM_LENGTH`=640 (default). This sets the number of samples desired in the waveform output.

♠ `WAVEFORM_CENTER`. This sets the number of bins offset from the current center at which you wish to examine the output. Default is the trigger bin. Basically how displaced is the trigger window from the triggering bin.

♠ `V_MIMIC_MODE`=0 (deprecated! Testbed only!). Every globally triggered event that is saved has a waveform closed off by a certain time window. The time to start recording and to end recording depends on the timing of the trigger and the time delays between channels. How much edges around the peak of the waveform will be saved into the file will depend on the offsets between channels, including factors such as station geometry and cable delays.
This mode affects the final time window for waveforms that are saved to file.

- ⋆ mode 0 (default): leave the waveform time windows as they are, for an ideal station.
- ⋆ mode 1: add the measured offsets for the testbed data, which moves the window of time for each waveform by a few nanoseconds. **Testbed only!**
- ⋆ mode 2: add some manual offsets in addition to that, which help make the simulated data closer to the real pulser data. **Testbed only!**

Note: if you are not using the Testbed mode (`DETECTOR` =3) you should not be changing this parameter to any value besides the default 0.

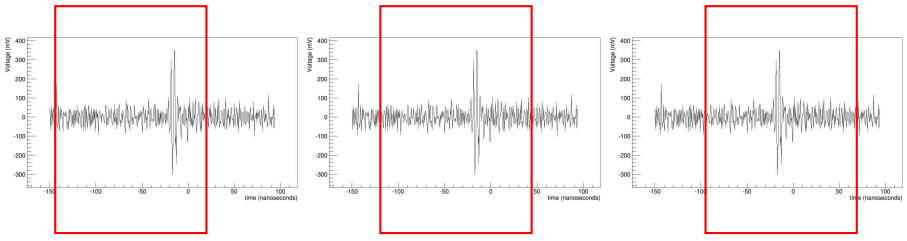

Figure 9: An example of different time windows for modes of V_MIMIC_MODE

3.11 Other Parameters

♠ `RANDOM_MODE`=1. At mode 0 we get the same random seed every run. Useful for debugging. The default mode 1 generates a new random seed assuring randomly generated events.

♠ `EVENT_TYPE`=0 (default) for neutrinos. =10 arbitary event; this mode takes in a waveform file (currently "arbitrary_waveform.txt") as if it was the E-field one meter from the antenna, and adds time delays.

♠ `LPM`=1. mode 0 no LPM effect. mode 1 (default) LPM effect is used.

These two functions decide the size and sample rate of the waveforms.

♠ `NFOUR`=1024. The number of bins in the fourier space waveform. This value must be some power of 2. The eventual waveform in time domain will have `NFOUR` / 2 bins.

♠ `TIMESTEP`=5E-10. The sampling rate for the channels. The defualt value is 5E-10, for 2 GHz sampling rate of actual digitizer.
The time duration of the recorded waveforms we get is decided by:

$$T = N_{\text{time domain}} \times \Delta t$$

where `NFOUR` $= 2N_{\text{time domain}}$ and `TIMESTEP` $= \Delta t$.

♠ `WHICHPARAMETERIZATION`=0. Old definitions from icemc. Do not change this parameter.

♠ `WAVE_TYPE`=0 plane wave. used for ray tracing puproses. Do not change this parameter.

♠ `PHASE`=90 in Degrees. This is used to make the Fourier components imaginary. Do not change this parameter.

♠ `TESTBED_ON`=0 (default). mode 1 is the old implementation. Do not change this parameter.

♠ `CALPUL_OFFCONE_ANGLE`=35 degrees. Old definition for previous implementation of cal pulser. Do not change this parameter.

♠ `MAXT_DIODE`=70E-9. Old parameter definition. Do not change this parameter.

♠ `IWINDOW_DIODE`=4.E-9 / `TIMESTEP` default is 10. Do not change this parameter.

♠ `CONST_MEANDIODE`=-6.5e-15 unused debugging parameter. Do not change this parameter.

♠ `CONST_RMSDIODE`=1.346e-13 unused debugging parameter. Do not change this parameter.

♠ `IDELAYBEFOREPEAK_DIODE`=13.E-9 / `TIMESTEP` defualt is 33. Do not change this parameter.

4 Incompatibilities and other pitfalls

Here is a short (and not comprehensive) list of problems that we may run into while using AraSim. Hopefully this will save some time debugging common errors in running and analysing simulations.

4.1 Incompatible settings

One common problem is using one mode of AraSim that does not work with another setting. The most frequent example is using testbed parameters with non-testbed `DETECTOR` modes. Although AraSim makes a check to prevent such problems a diligent user may still find ways to crash the program or to get strange results.
A good habit is to keep distinct setup files with settings for testbed (which are tweaked as necessary), a seperate file for ideal detector(s), and so on for each commonly used variety of simulation.

- A common issue is with **testbed only** parameters. A list of such parameters is given:
  - `NOISE_TEMP_MODE` $= 1$
  - `TRIG_ONLY_BH_ON` $= 1$
  - `USE_CH_GAINOFFSET` $= 1$
  - `USE_TESTBED_RFCM_ON` $= 1$
  - `RFCM_GAINOFFSET` only checked if the above mode is used.
  - `READGEOM` $= 1$
  - `CALPULSER_ON` > 0
  - `CALPUL_AMP` only checked if the above mode is used.
  - `V_MIMIC_MODE` > 0
- Some parameter changes that should be made when working with a detector array (when using `DETECTOR` $= 1$ with `number_of_stations` > 1) or with `DETECTOR` $= 2$:
  - In 'pick near' mode, make sure the radius given in `POSNU_RADIUS` is large enough to contain the whole array.
  - In such cases the `RAYSOL_RADIUS` may be smaller than `POSNU_RADIUS` .
- Although the 'pick near' function (`INTERACTION_MODE` $= 1$) is very useful for running shorter simulations, using 'pick unbiased' (`INTERACTION_MODE` $= 0$) may be more accurate when trying to simulate the real results of an experiment in Antarctica. Some issues may arise though:
  - Since many neutrinos will be generated outside the `RAYSOL_RADIUS` , it may take a long while before any of them triggers the detector. Increasing this radius will not necessarily make things better, as it will only incur more calls to the ray solver, for events that are hopelessly far away.
  - Using low energy spectra or small detector arrays will also reduce the number of triggered events (remember that even ARA37 is estimated to detect a handful of events a year).
  - In cases of low trigger rate we may get empty output trees (if working in `ONLY_PASSED_EVENTS` $= 0$) or an indefinitely long runtime (in `ONLY_PASSED_EVENTS` $= 1$).

4.2 Problems running AraSim

Here are some tips and known problems one may run into using AraSim.

- When simulation is finished we still need to do some analysis on the 'outputs/AraOut.root' file. Some common issues are:
  - Is the file very small? Are you sure there were any global triggers?
  - Are the simulation parameters reasonable? Running `NNU` $= 100$ in 'pick unbiased' mode will likely cause no triggers.
  - Did you choose the right data save modes (see section 3.10)?
  - Did you load the right libraries (to read classes like Detector, Event, etc..)?

- ♢ Are you looking in the right tree? Remember:
  - eventTree contains the "realistic" data from the detector (only the final waveforms and measureable quantities;
  - AraTree2 conatins the "behind the scenes" data on each event (position and energy of neutrinos, etc); and
  - AraTree contains the global run parameters like detector and neutrino spectrum used.

⋆ Simulation running too long. This usually happens in `ONLY_PASSED_EVENTS` = 1 where AraSim continues to run until the number of global triggers reachs `NNU_PASSED` .

- ♢ Did you choose reasonable parameters? Running `NNU_PASSED` = 100 in 'pick unbiased' might take a *very* long time.
- ♢ Are you using reasonable energies and detector layout to be able to see any neutrinos?
- ♢ Are the neutrinos being generated inside the range of `RAYSOL_RADIUS` ? If not, simulation may run indefinately.
- ♢ In `TRIG_ANALYSIS_MODE` = 2 triggering only runs on noise, so trigger rate may be very low (unless we change the threshold).
- ♢ If run time is $< \infty$ but still too long, consider splitting simulation into several runs (see section 2).

5 Additional Issues

Some more explanations on some of the results we can get from AraSim.

5.1 Working with Weights

working progress

6 Some References

You can always check some of these DocDB sources where some of the features are described.

⋆ `http://ara.physics.wisc.edu/cgi-bin/DocDB/ShowDocument?docid=1533`

7 Reference to AraSim parameters

// Setup file with basic notation. Please report any issues or mistakes:
(guy.nir@weizmann.ac.il)
Date: October 31, 2017// #

##// neutrino energies (user guide: see section 3.1 page 6) //#

#EXPONENT=19 // (DEFAULT=19) the energy spectrum of the neutrinos. #

##// number of neutrinos (user guide: see section 3.2 page 7) //#

#ONLY_PASSED_EVENTS=0 // (DEFAULT=0) 0: runs NNU neutrinos. 1: runs until NNU_PASSED neutrinos globally trigger. #
#NNU=100 // (DEFAULT=100) The total number of neutrinos thrown. #
#NNU_PASSED=10 // (DEFAULT=10) when using ONLY_PASSED_EVENTS this determines how many events will pass the trigger. #
#OUTPUT_TDR_GRAPH=0 // (DEFAULT=0) saves this number of tunnel diode response waeforms. #

##// generating noise (user guide: see section 3.3 page 8) //#

#NOISE_EVENTS=16 // (DEFAULT=16) the number of noise wf generated. #
#NOISE_WAVEFORM_GENERATE_MODE=0 // (DEFAULT=0) new noise wf are generated for each event. 1: old mode all noise wf are made beforehand. #
#NOISE=0 // (DEFAULT=0) 0: flat thermal distribution. 1: calibrated Rayleigh noise #
#NOISE_TEMP_MODE=0 // (DEFAULT=0) 0: same noise wf for all channels. 1: different temperatures for each channel. 2: diff temps for first 8 channels. #
#NOISE_TEMP=325 // (DEFAULT=325) if using thermal distribution, use this temperature (in Kelvins). #
#DATA_BIN_SIZE=16384 // (DEFAULT=16384) size of noise wf used to get noise floor (or to make all noise wf's before simulation when NOISE_WAVEFORM_GENERATE_MODE =1. must be power of 2. #

##// neutrino position (user guide: see section 3.4 page 9) //#

#INTERACTION_MODE=1 // (DEFAULT=1) 0: make nu's all over antarctica. 1: pick near (a cylinder around detector). 2: exact location at (353.55, 612.37, 707.1). 3: pick near-unbiased (a sphere around detector) #
#POSNU_THETA= // (DEFAULT=) Neutrino interaction location elevation angle in radians. For use with mode INTERACTION_MODE =2. #
#POSNU_PHI= // (DEFAULT=) Neutrino interaction location azimuthal angle in radians. For use with mode INTERACTION_MODE =2. #
#POSNU_R= // (DEFAULT=) Neutrino interaction location radius in meters. For use with mode INTERACTION_MODE =2. #
#POSNU_RADIUS=3000 // (DEFAULT=3000) radius around detector in pick near mode. #
#PICKNEARUNBIASED_R=5000 // (DEFAULT=5000) spherical radius around detector in pick near-unbiased mode. #
#PICK_POSNU_DEPTH=0 // (DEFAULT=0) 0: use all ice down to bedrock. 1: choose maximum depth yourself. #
#MAX_POSNU_DEPTH=0 // (DEFAULT=0) the maximum depth when PICK_POSNU_DEPTH =1. #
#NNU_THIS_THETA=0 // (DEFAULT=0) 0: choose nutrino incident angle at random. 1: choose specfic angle range. #
#NNU_THETA=0 // (DEFAULT=0) the angle chosen when NNU_THIS_THETA =1 is chosen. Use radians! #
#NNU_D_THETA=0 // (DEFAULT=0) the range around NNU_THETA (up and down). #
#SECONDARIES=1 // (DEFAULT=1) 0: no secondary interactions. 1: allow secondary interactions. #
#TAUDECAY=1 // (DEFAULT=1) 0: do not allow secondary tau decay. 1: allow secondary decay of tau's, only works when SECONDARIES =1. #

##// ray solving (user guide: see section 3.5 page 10) //#

#RAY_TRACE_ICE_MODEL_PARAMS=0 // (DEFAULT=0) What index of refraction model to use. See Settings.h for list of all available. #
#RAYSOL_RANGE=5000 // (DEFAULT=5000) distance from detector above which we do not attempt a ray solution (there is no trigger check beyond this). #

##// triggering (user guide: see section 3.7 page 12) //#

#USE_INSTALLED_TRIGGER_SETTINGS=0 // (DEFAULT=0) 0: ideal detector trigger settings. 1: use triggering as implemented in real stations (**testbed only!**). #
#TRIG_ANALYSIS_MODE=0 // (DEFAULT=0) 0: signal+noise. 1: signal only. 2: noise only #
#POWERTHRESHOLD=-6.15 // (DEFAULT=-6.15) the trigger threshold (the closer to zero the weaker the threshold. measured in sigmas above the noise floor. #
#TRIG_WINDOW=1.1E-7 // (DEFAULT=1.1E-7) time window in which several channels must trigger to pass global trigger test. #
#TRIG_TIMEOUT=1E-6 // (DEFAULT=1E-6) dead time for detector. #
#TRIG_MODE=1 // (DEFAULT=1) 0: trigger for N antennas of any kind. 1: trigger on either Nh Hpol or Nv Vpol antennas. #
#N_TRIG=3 // (DEFAULT=3) number of total antennas required for a global trigger (if TRIG_MODE =0). #
#N_TRIG_V=3 // (DEFAULT=3) number of Vpol required for a global trigger (if TRIG_MODE =1). #
#N_TRIG_H=3 // (DEFAULT=3) number of Hpol required for a global trigger (if TRIG_MODE =1). #
#TRIG_SCAN_MODE=0 // (DEFAULT=0) triggering algorithm. 0: old code. 1: new, faster code. 2: scan all powerthreshold values. 3: save single channel trigger powerthreshold values, too. #
#TRIG_ONLY_BH_ON=0 // (DEFAULT=0) 0: check trigger on all channels. 1: check only borehole antennas. #
#TRIG_THRES_MODE=0 // (DEFAULT=0) 0: use same threshold for all channels. 1: use seperate thershold for each channel using 'data/thresholdoffset.csv'. #
#USE_MANUAL_GAINOFFSET=0 // (DEFAULT=0) 0: pass control to USE_CH_GAINOFFSET below. 1: add a single value to the offset of all channels. #
#MANUAL_GAINOFFSET_VALUE=0 // (DEFAULT=0) is the offset to the threshold when using USE_MANUAL_GAINOFFSET =1 #
#USE_CH_GAINOFFSET=0 // (DEFAULT=0) 0: use no gain offsets. 1: use specfic gain offsets for each channel using 'data/preampgainoffsets.csv' (**testbed only!**) #
#USE_TESTBED_RFCM_ON=0 // (DEFAULT=0) 0: don't apply specific amplification offsets. 1: use measured TB data to cancel amplification gain. #
#RFCM_OFFSET=80 // (DEFAULT=80) when USE_TESTBED_RFCM_ON =0 use this value to cancel the amplification gain of electronic components. #

##// ice and earth (user guide: see section 3.8 page 14) //#

#CONSTANTCRUST=0 // (DEFAULT=0) This is a left over parameter, do not change these settings. #
#CONSTANTICETHICKNESS=0 // (DEFAULT=0) This is a left over parameter, do not change these settings. #
#FIXEDELEVATION=0 // (DEFAULT=0) Elevation fixed to the thickness of the ice. Don't change this setting. #
#MOOREBAY=0 // (DEFAULT=0) which attenuation length measurements to use. 0: south pole data. 1: Moore's Bay measurements. #
#ATMOSPHERE=1 // (DEFAULT=1) 0: no atmosphere (not recommended). 1: use atmosphere (default). #
#ICE_MODEL=0 // (DEFAULT=0) 0: Crust 2.0 or 1: bedmap (old ice model). #
#NOFZ=1 // (DEFAULT=1) 0: constant index of refraction (not recommended). 1: n changes with depth (default). #
#GETCHORD_MODE=0 // (DEFAULT=0) which code to use for weights of nu's passing the earth / atmosphere. 0: old code. 1: new untested code. #
#taumodes=0 // (DEFAULT=0) 0: no tau modes. 1: allow tau nu's interaction in bedrock to create tau's. works only when GETCHORD_MODE =1. #

##// detector layout (user guide: see section 3.9 page 15) //#

#DETECTOR=1 // (DEFAULT=1) 0: unused. 1: ideal ARA station. can choose 1 to 7 stations. 2: large array in hexagonal design. 3: testbed. 4: Import geometry from AraRoot. #
#DETECTOR_STATION= // (DEFAULT=) 2, 3, etc. What station's geometry to import from AraRoot. Only works with DETECTOR =4 #
#number_of_stations=1 // (DEFAULT=1) number of stations used in DETECTOR =1 mode. #
#station_spacing=2000 // (DEFAULT=2000) separation between stations in modes 1 and 2. in meters. #
#stations_per_side=4 // (DEFAULT=4) number of stations on a side of the hexagon in DETECTOR =2 mode. for N on a side get 3N(N-1)+1 stations. #

#core_x=10000 // (DEFAULT=10000) location of array in global coordinates, used to avoid the bin crossing of 2degree bins. #
#core_y=10000 // (DEFAULT=10000) same as above. #
#number_of_strings_per_station=4 // (DEFAULT=4) number of strings on an ideal station. #
#R_string=10 // (DEFAULT=10) distance of string poisitions from center of station. in meters. #
#BORE_HOLE_ANTENNA_LAYOUT=0 // (DEFAULT=0) order of antennas on the string. 0: HVHV. 1: VHV. 2: VVHV. 3: HHHV. 4: HHV. #
#z_max=200 // (DEFAULT=200) lowest antenna position on the string. in meters depth #
#BH_ANT_SEP_DIST_ON=1 // (DEFAULT=1) 0: all distances are the same (choose **z_btw**). 1: choose each distance youself (use **z_btwAB**). #
#z_btw=10 // (DEFAULT=10) single distance for all BH antennas, used when BH_ANT_SEP_DIST_ON = 0 #
#z_btw01=2 // (DEFAULT=2) distance between lowest antenna and next one. in meters #
#z_btw12=15 // (DEFAULT=15) distance between second and third antenna. #
#z_btw23=2 // (DEFAULT=2) distance between third and last (uppermost) antenna. #
#READGEOM=0 // (DEFAULT=0) 0: use idealized detector layout. 1: read from sqlite file. TB ONLY! #
#CALPULSER_ON=0 // (DEFAULT=0) 0: no calpulser. 1: use 2012 pulser. 2: use 2011 Vpol. 3: use 2011 Hpol. 4: use both V and H pol, average location of pulser. TB ONLY! #
#CALPUL_AMP=0.15 // (DEFAULT=0.15) amplitude modifier for the calpulser, to match the power of real pulser # .

##// writing events (user guide: see section 3.10 page 17) //#

#DATA_SAVE_MODE=0 // (DEFAULT=0) 0: save waveforms and primitive spectra to AraTree2 (include spectra at different stages of simulation). 1: save only waveforms after trigger. 2: save just physics data per event. #
#DATA_LIKE_OUTPUT=0 // (DEFAULT=0) 0: no data-like output. 1: output triggered events only. 2: output all events (signal only inserted into data-like output). #
#FILL_TREE_MODE=0 // (DEFAULT=0) 0: save all events to AraTree2 (triggered or not) 1: don't save events outside of ice (in pick unbiased mode) 2: save just triggered events. #
#WAVEFORM_LENGTH=640 // (DEFAULT=640) Set the number of samples desired in the waveform output #
#WAVEFORM_CENTER= // (DEFAULT=) The number of bins offset from the current center at which you wish to examine the output. Default center is the triggering bin. #
#V_MIMIC_MODE=0 // (DEFAULT=0) 0: save wavforms using standard time window. 1: use offsets from data to get real time windows (TB ONLY!). 2: add manual offsets to thos in 1 (TB ONLY!). #

##// other parameters (user guide: see section 3.11 page 19) //#

#RANDOM_MODE=1 // (DEFAULT=1) 0: same random seed (debugging). 1: new random seed for simulations #
#SIMULATION_MODE=0 // (DEFAULT=0) 0: frequency domain. 1: time domain simulation #
#EVENT_TYPE=0 // (DEFAULT=0) 0: neutrinos. =10 is arbitrary event waveform located in "arbitrary_waveform.txt" file. #
#LPM=1 // (DEFAULT=1) 0: don't use LPM effect. 1: use LPM (default) #
#NFOUR=1024 // (DEFAULT=1024) Fourier space sample size. Twice the number of bins in time domain wf. must be power of 2. #
#TIMESTEP=5E-10 // (DEFAULT=5E-10) sample rate for digitizer. decides the time difference between data points in wf. #