AraSim Document (unofficial version)

Eugene Hong[1]

[1]*Department of Physics, The Ohio State University, Columbus, Ohio 43210, USA*

(Dated: October 11, 2011)

I. INTRODUCTION

Making an manual for AraSim.

II. SETTINGS CLASS

Currently, there are 8 parameter values we can set for AraSim from Settings class. These variables are set as default values When class "Settings" is called, default values which shown on Table. I will be automatically assigned.

Name	Description	Default value
NNU	total number of neutrinos to generate	100
ICE_MODEL	1=depth dependent on index of refraction, 0=off	1
CONSTANTCRUST	set crust density and thickness to constant values	0
CPMSTAMTICETHICKNESS	set ice thickness to the thickness of ice	0
FIXEDELEVATION	fix the elevation to the thickness of ice	0
MOOREBAY	1=use Moore's Bay measured ice field attenuation length for the west land, 0=use South Pole data	0
EXPONENT	set single neutrino energy or neutrino flux model. More detail will be followed	19.
DETECTOR	set detector layout	1

TABLE I: Parameter values for Setting class.

If you want to change any parameter values from default values, you can just set any new values at "setup.txt" file. In setup.txt file, all sentences starts with "/" will be neglected when AraSim read this file. So, you can leave any comments on setup.txt file. When you want to change any parameter value, just put the name of parameter and put "=" and finally put the value which you want. For example, if you want to change "NNU", just type

NNU=1000

in setup.txt file. You can change many parameters and the order of parameters are not important.

III. DETECTOR CLASS

Now, after you set the parameters as previous section, next thing we can look on is setting the detector. For now (091611), there are 3 options when we call Detector class. From the Settings, you can select the layout of detector with parameter "DETECTOR". Table .II shows the options for parameter "DETECTOR".

DETECTOR	Description	Reading file
0	Detector layout as testbed.	testbed_info.txt
1	Small number of stations (same or smaller than 7 stations)	ARA_N_info.txt
2	Large number of ARA stations. Shape as hexagon.	ARA37_info.txt

TABLE II: DETECTOR options and descriptions.

Like in Settings class, Detector class also have default values and this default values can be modified by editing txt files mentioned in Table. II.

Parameter name	Description	Default value
core_x	x position of center of hexagon	0
core_y	y position of center of hexagon	0
R_string	Distance between center of hexagon and string	10m
R_surface	Distance between center of hexagon and surface antenna	60m
z_max	Total depth of string. There is an antenna at the bottom of string.	200m
x_btw	Distance between antennas on the string.	20m
number_of_stations	Total number of stations.	1
station_spacing	Distance between stations.	2000m
antenna_orientation	Setting for antenna orientation	0

TABLE III: DETECTOR 1 parameters which can be modified through ARA_N_info.txt file.

More detailed information about each DETECTOR mode is followed.

A. DETECTOR : 0

In DETECTOR 0, AraSim will read testbed_info.txt file. In testbed_info.txt file, it sets total number of strings (number_of_string = 4) and location of all strings (x, y) and each antennas (2 antennas on each string, z). The value after the "z" indicates the type of the antenna. When number 0 shows after depth "z", it means the antenna is a v-pol bicone antenna while number 1 means the antenna is a h-pol bowtie-slotted cylinder antenna. The location for antennas are from Peter Gorham's spreadsheet.

B. DETECTOR : 1

In DETECTOR 1, AraSim will read ARA_N_info.txt file. DETECTOR 1 is for small number of ARA stations, from 1 to 7 stations. As the number of stations go up, the layout of stations will become hexagon with 2 stations on the side. (layout for 7 stations will be hexagon with 2 stations on the side, which is maximum for DETECTOR 1) In ARA_N_info.txt file, you can set total number of stations (number_of_stations = 1), and distance between center of hexagon and strings (R_string), and distance between center of hexagon and surface antennas (R_surface). In the present version of AraSim, we can't change the number of strings and surface antennas in one station (4 strings and 4 surface antennas). We are planning to make it possible to change number of strings and surface antennas per station later. And the layout of strings and surface antennas are following Fig. 1. For the layout of strings and antennas, we can only change the distances for

Parameter name	Description	Default value
core_x	x position of center of hexagon	0
core_y	y position of center of hexagon	0
R_string	Distance between center of hexagon and string	10m
R_surface	Distance between center of hexagon and surface antenna	60m
z_max	Total depth of string. There is an antenna at the bottom of string.	200m
x_btw	Distance between antennas on the string.	20m
stations_per_side	Total number of stations.	4
station_spacing	Distance between stations.	2000m
antenna_orientation	Setting for antenna orientation	0

TABLE IV: DETECTOR 2 parameters which can be modified through ARA37_info.txt file.

now.

We can also change the center of hexagon. By default, center of hexagon layout is x = 0, y = 0. And in ARA_N_info.txt, you can change the the center location with parameters core_x and core_y. For example, you can set center of hexagon as x = 10m, y = 5m by

core_x = 10

core_y = 5

As a default, borehole antennas are ordered by V, H, V, H pol antennas starting from the bottom of the string. For now, this ordering can not be changed. (can be modified later). We can check the type of antennas by

```
Detector *detector;
int x = detector->stations[i].strings[j].antennas[k].type;
```

above, returned value of type shows what kind of antenna it is. Description about "type" is shown on Table. V.

type	description	antenna gain file
0	V-pol 6in bicone antenna	ARA_bicone6in_output.txt
1	H-pol 23in QSC antenna	ARA_dipoletest1_output.txt

TABLE V: Description of parameter "type" of antennas. Antenna gain file is on /AraSim folder. QSC antenna stands for quad slotted cylinder antenna.

Another property of antenna which we can set is the orientation of the antenna. With the default settings, all antennas (including borehole antennas and surface antennas) are facing x direction. As some antennas have non-azimuthal symmetric responce, layout of antennas' orientation can change the simulation results. So far, there are two options for setting antenna orientation. Table. VI shows the options of parameter "antenna_orientation". antenna_orientation=0 is simple to understand. Just all antennas are facing x direction. antenna_orientation=1 is little bit complex. As there are V-pol and H-pol antennas in each string, when one V-pol and H-pol antenna pair are facing one direction (if x direction), the pairs of antennas near by are facing different direction (then y direction), and so on. So far, there are only two options available. (more options can be available later?)

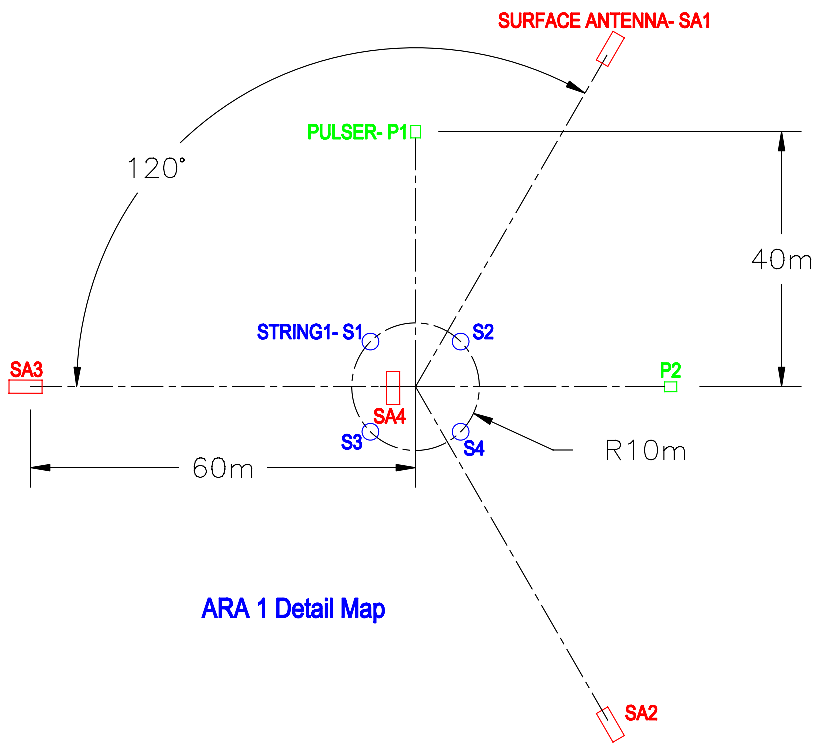

FIG. 1: ARA 1 detail map. Each ARA stations have same layout of strings and surface antennas.

C. DETECTOR : 2

DETECTOR 2 is for the case when there are numerous of stations. In DETECTOR 2 mode, layout of stations is always hexagon, and you have to set the number of stations on the side (number_per_side) which determinds total number of stations. Minimum value for number_per_side is 2, which is same as in DETECTOR 1 with number_of_stations = 7. In DETECTOR 2 mode, you can't input the number_of_stations value as Detector class will automatically calculate that value from number_per_side. Default value for the number of stations on the side is 4, which is the number for ARA-37 (Fig. 3). Other than the layout of stations, other parameters works same as

antenna_orientation	description
0	all antennas facing x direction
1	V-pol and H-pol antenna pairs facing x and y on alternate

TABLE VI: Description of parameter "antenna_orientation". This value can be set on Setting files in Table. III and Table. IV.

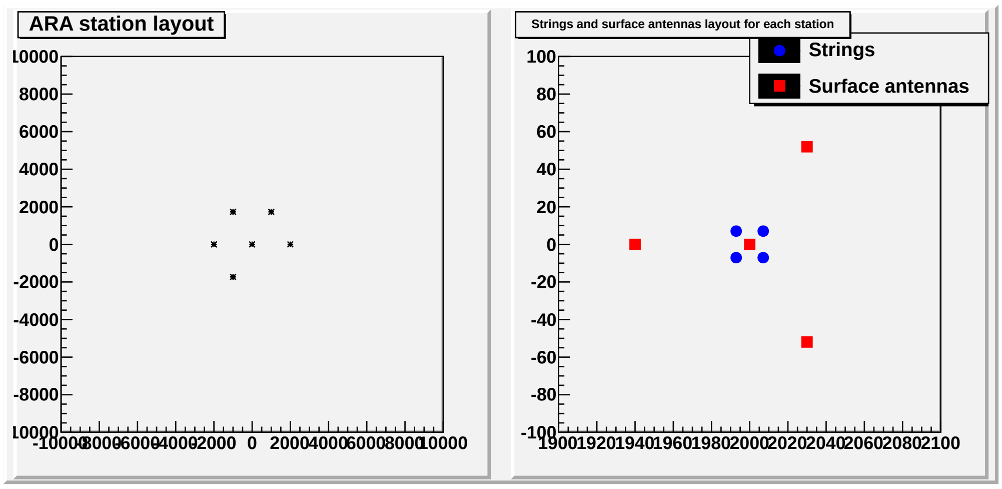

FIG. 2: Result of DETECTOR 1 mode, 6 stations. Left : layout of stations. Right : layout of one station's string and surface antennas.

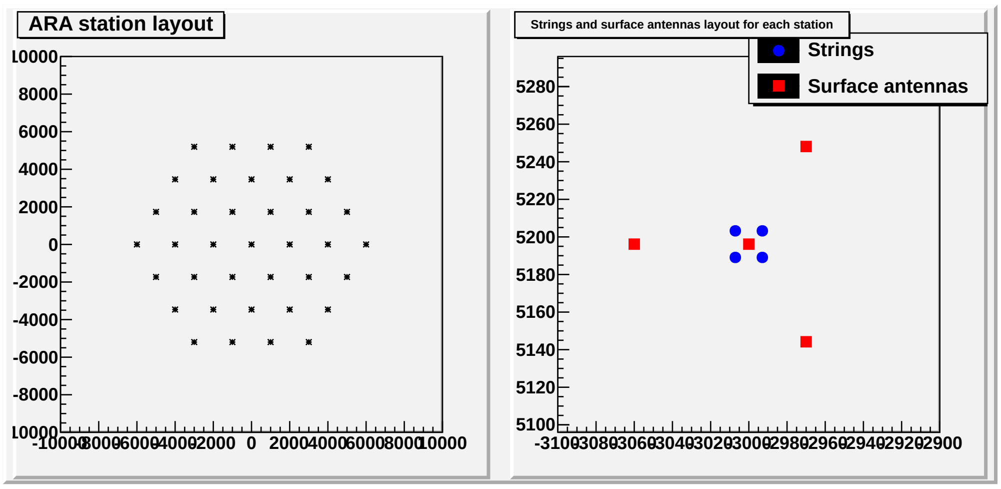

FIG. 3: Result of DETECTOR 2, default setting. ARA 37 station layout is shown on left plot, while layout of each station is shown on right plot.

in DETECTOR 1.

D. Antenna property : Gain

When Detector class is called in AraSim.cc, Detector class will automatically read gain of V-pol and H-pol borehole antennas from text files when DETECTOR mode is in 1 or 2. (so far, antenna gain for testbed DETECTOR 0 mode is not prepaired) The gain of V-pol and H-pol antennas' are obtained from nec2 antenna simulation. File "ARA_bicone6in_output.txt" contains the information about V-pol borehole antennas. It's modified from Peter Gorham's nec2 simulation result. "ARA_dipoletest1_output.txt" file contains the information about simplified quad slotted cylinder antenna (QSC) from nec2c simulation. As slotted antenna can be simplified to dipole antenna, instead of 23in long QSC antenna 23in dipole antenna is used for nec2c simulation. Both files are modified from original nec2 output files to read information more easy.

After Detector class is called with DETECTOR mode in 1 or 2, we can read the gain of antenna in two different ways.

1) read gain of general antenna.

After we Detector class is called, any antenna gain value (not in dB) can be obtained by "GetGain" function. For example,

```
Detector *detector = new Detector(settings1->DETECTOR);
double Gain = detector -> GetGain(700, 20, 200, 0);
```

above, GetGain will return the gain value at 700 MHz, angle theta 20 deg, phi 200 deg, and V-pol antenna (antenna type 0 in Table. V). It is simple to read any antenna's gain at any frequency, and angle. However, it's not conveniant to read gain value of certain antenna, as borehole antennas' layout is not that simple.

2) read gain of certain antenna.

After we call Detector calss with DETECTOR mode 1 or 2, it will make a layout of stations, strings and antennas. We can ask any certain antenna to get gain similar to previous method. For example,

```
Detector *detector = new Detector(settings1->DETECTOR);
double Gain = detector->stations[0].strings[1].antennas[2].GetG(detector, 700, 20, 200);
```

above, double Gain will obtain the gain value of the 3rd antenna from the bottom of the 2nd string in the 1st station at 700 MHz, 20 deg theta and 200 deg phi. GetG function needs to read the address of Detector class called as it needs to know where the stored antenna gain data is. This GetG function in antenna will automatically find the type and orientation of the antenna.

IV. SPECTRA CLASS

Spectra class determines either the energy of single neutrino event or neutrino flux shape for numerous neutrino events. I will introduce how we can call Spectra class for many different neutrino flux model cases, and also how we can obtain flux plots, specific flux value at certain energy, and more.

A. Construct Spectra

When we call Spectra class, one easy way is

```
int EXPONENT = 10;
Spectra *spectra1 = new Spectra( (int) EXPONENT );
```

above, I called Spectra class with name "spectra1". "EXPONENT" value will define which neutrino flux model or energy of single neutrino will be used. Detail explanation about the "EXPONENT" values are shown in Table VII.

EXPONENT values are ordered based on the different cases. EXPONENT values between 1 and 4 are for single neutrino energy case. It means single neutrino event will be produced with the energy you choose. EXPONENT values between 30 and 199 are for neutrino flux model with only proton source cases. And values from 200 are for mixed heavy particle (ex. Iron) and proton or heavy particle dominant cases.

You can add more neutrino flux model by adding flux file in fluxes folder. As you can find README file in fluxes folder, first line of flux model file should be the number of data points. From the second line, first column is energy in log scale and second column is flux of neutrino E^2dN/dE [GeV/(cm^2·str·s)]. Evergy points don't need to be evenly spaced as TSpine will functionalize the flux. Number of points can be 50 by maximum. After you add a file for new neutrino flux model, you can edit a Spectra.cc file to add that file to read. Just open Spectra.cc file and find a line with "else if (EXPONENT==some_number)". You can add new file and assign new EXPONENT number. After you save Spectra.cc file, call Spectra class with the new EXPONENT value you just assigned in AraSim.cc.

B. Useful members in Spectra

◦ **GetNuEnergy()**

After you call new Spectra class with specific EXPONENT value, you can get random neutrino energy which follows neutrino flux by (when EXPONENT is bigger than 10)

```
spectra1 -> GetNuEnergy();
```

above, "spectra1" is assigned Spectra class with name spectra1 as before.

◦ **Getenergy(), GetEdNdEdAdt(), GetE2dNdEdAdt()**

With Getenergy member, you can read array of energy values for each energy bins. You can simply use by

EXPONENT	Description	file in fluxes folder
1	$dN/dE \sim E^{-1}$ single neutrino energy	
2	$dN/dE \sim E^{-2}$ single neutrino energy	
3	$dN/dE \sim E^{-3}$ single neutrino energy	
4	$dN/dE \sim E^{-4}$ single neutrino energy	
30	Engel, Seckel, Stanev (ESS) baseline model	
31	ESS-cosmological constant model	
32	ESS, PRD 64, 093010 Fig 9 for nu_mu + nu_e	essfig9.dat
33	ESS, flux_m3_flat_lamda0.7	essbaseline.dat
34	ESS, without evolution, ftp://ftp.bartol.udel.edu/seckel/ess-gzk/flux_n0_2.7.tab	ess_n0.dat
40	Ahler, PRD 72, 023001 (2005) Fig 4b	ahlers.dat
50	Allard, astro-ph/0605327 v2 22 May 2006, Fig 11 slim solid line	allard.dat
60	Ave, astro-ph/0409316v2 6 Nov 2004, Fig 6 solid curve (upper)	ave_max.dat
61	Ave, astro-ph/0409316v2 6 Nov 2004, Fig 6 dotted curve (lower)	ave_min.dat
70	Kalashev, Kuzmin, Semikoz, Sigl (KKSS) hep-ph/0205050 v3 13 Dec 2002 Fig 7 envolop	kkss_envo.dat
80	Peter Gorham's GZK flux from ANITA proposal (upper blue line)	gzk_peter.dat
90	Waxman GZK flux taken from David Seckel's ftp GZK_update.ps page 1, green curve in the right plot	waxgzk.dat
100	E-2 model.	e-2.dat
110	Yuksel, astro-ph/0610481v2 9 Nov 2006 Fig 4	yuksel_grb.dat
111	Yuksel, astro-ph/0610481v2 9 Nov 2006 Fig 4	yuksel_qso.dat
112	Yuksel, astro-ph/0610481v2 9 Nov 2006 Fig 4	yuksel_sfh.dat
200	Ave, Astro. Part. Phys. 23. (2005) 19-29, Fig 6	Ave2005_Fe_Emax21.0.dat
201	Ave, Astro. Part. Phys. 23. (2005) 19-29, Fig 6	Ave2005_Fe_Emax21.5.dat
202	Ave, Astro. Part. Phys. 23. (2005) 19-29, Fig 6	Ave2005_Fe_Emax22.0.dat
203	Ave, Astro. Part. Phys. 23. (2005) 19-29, Fig 5 dashed line	Ave2005_Fe_hi_evo.dat
204	Ave, Astro. Part. Phys. 23. (2005) 19-29, Fig 5 solid line	Ave2005_Fe_low_evo.dat
210	Stanve, Nuclear Inst. Meth. Phys. Research A 588 (2008) 215-220, Fig 7 pentagon	Stanev2008_heavy.dat
220	Kotera, arXiv:1009.1382v2, Fig 9 blue dashed line	Kotera2010_Fe_only.dat
221	Kotera, arXiv:1009.1382v2, Fig 9 blue dotted line	Kotera2010_Fe_rich.dat
222	Kotera, arXiv:1009.1382v2, Fig 9 top shaded line	Kotera2010_mix_max.dat
223	Kotera, arXiv:1009.1382v2, Fig 9 bottom shaded line	Kotera2010_mix_min.dat

TABLE VII: EXPONENT values description.

double *energy = spectra1 -> Getenergy();

GetEdNdEdAdt and GetE2dNdEdAdt will return the array of neutrino flux values for EdNdEdAdt and E2dNdEdAdt respectively.

◦ **GetE_bin()**

As each flux models have different number of energy bins, sometimes we need to know the bins numbers. With GetE_bin(), we can read the number of energy bins,

int N_Bin = spectra -> GetE_bin();

◦ **IsSpectrum()**

Returns 1 if EXPONENT value is chose for spectrum of neutrino model. If EXPONENT value is set for single energy neutrino, returns 0.

◦ **IsMonoenergetic()**

Similar with above IsSpectrum, but it works oppositely. Retruns 1 if EXPONENT is set for single energy neutrino, and returns 0 if spectrum.

◦ **GetEdNdEdAdt(double E)**

When we use spectrum for neutrino events, with GetEdNdEdAdt we can read specific flux value at certain energy value E. The flux value will be obtained from functionalized TSpline result, so any energy value can be used. Retruned flux value is in $[1/(\mathrm{cm}^2 \cdot \mathrm{str} \cdot \mathrm{s})]$. When energy value E is greater than the maximum energy bin value, it will return the flux value at the maximum energy bin. On contrast, if E is smaller than the minimum energy bin value, it will return the flux value at the minimum energy bin.

double E = 20.0;

double Flux = spectra1 -> GetEdNdEdAdt(E);

◦ **GetE2dNdEdAdt(double E)**

It works similar with GetEdNdEdAdt, but returns E^2dNdEdAdt $[\mathrm{GeV}/(\mathrm{cm}^2 \cdot \mathrm{str} \cdot \mathrm{s})]$ flux value.

◦ **Getmaxflux()**

From the chosen neutrino flux model, it will return the maximum flux value in EdNdEdAdt unit.

◦ **GetGEdNdEdAdt(), GetGE2dNdEdAdt()**

They will retrun TGraph of EdNdEdAdt and E2dNdEdAdt flux respectively. You can use by

TGraph *graph;

graph = spectra1 -> GetGEdNdEdAdt();

◦ **GetSEdNdEdAdt(), GetSE2dNdEdAdt()**

Both GetSEdNdEdAdt and GetSE2dNdEdAdt will return TSpline3 of EdNdEdAdt and

E2dNdEdAdt flux respectively. This TSpline3 plot will functionalize flux and make it possible to read intermediate flux values between data points. You can read by

```
TSpline3 *sp1;
sp1 = spectra1 -> GetSEdNdEdAdt();
```