CORSIKA processing with gSeaGen

Git release: v7.4.3-test12-4-g7bf5e3c

Piotr Kalaczyński
pkalaczynski@km3net.de

Andrey Romanov
Andrey.Romanov@ge.infn.it

May 7, 2024

1 Introduction

In this note, processing of CORSIKA files using gSeaGen is explained, along with major changes introduced with gSeaGen v7.2.0 (see section 3)

2 Processing corant .evt files [deprecated]

The original approach was to first convert CORSIKA files (binary unformatted fortan files, documented in [1]) to the ANTARES ascii format (.evt). This was done using the corant code, which performed:

- format and unit conversion
- coordinate system transformation from the one used in CORSIKA to the KM3NeT system
- preliminary computation of weights, assuming GST3 [2] for the primary Cosmic Ray flux (weights were completed by adding a detector-dependent factor when processing with gSeaGen)

This approach was deprecated as the .evt format is inefficient for mass processing. Although no longer developed, it is still available for use. To use it, one has to do:

```
gSeaNuEvGen -f "EVT:$DIR/DAT$RUN.evt"
```

$DIR points to the directory where the corant file is located and $RUN is the run number (the standard file naming scheme is DATXXXXXX.evt).

3 Processing CORSIKA binary files [default]

The default way to process CORSIKA files with gSeaGen is:

```
gSeaNuEvGen -f "BIN:$DIR/DAT$RUN"
```

This makes use of a CORSIKA file driver (GSeaCORSIKAFileFlux) developed based on the .evt file driver and *readcorsika.cpp* script by J. Öhlschläger (distributed together with CORSIKA). All functionalities of corant have been absorbed into GSeaCORSIKAFileFlux, along with the sea-level information, which will be saved to the output files if the following option is used:

```
gSeaNuEvGen -f "BIN:$DIR/DAT$RUN" -write 2
```

There has been a number of functionalities added along with the introduction of direct CORSIKA processing in v7. One of them is the extension of the information stored in tracks and addition of track categories: grandmother, mother and muaddi (muon at production point), besides the originally used muon track. These were added to introduce the parent-daughter relation between the particles and accommodate for the additional information. It is summarised in Table 1. This does not mean that now all the information from CORSIKA is stored, but the part important for KM3NeT simulations is extracted.

3.1 Track information

In Table 1 there is a pID for each track. Most of the particle IDs are simply converted from CORSIKA IDs to PDG IDs. However, there is a number of CORSIKA-specific cases that cannot be simply converted due to a lack of PDG counterpart:

```
{ 71 },      // eta -> 2 gamma
{ 72 },      // eta -> 3 pi0
{ 73 },      // eta -> pi+ pi- pi0
{ 74 },      // eta -> pi+ pi- gamma
{ 75 },      // muon+ add. info
{ 76 },      // muon- add. info
```

So far:

Track	x	y	z	$\frac{p_x}{p}$	$\frac{p_y}{p}$	$\frac{p_z}{p}$	E	t	pID	l	E_{loss}
primary	$= 0$	$= 0$	✓	✓	✓	✓	✓	$= 0$	✓	$= 0$	N/A
add. μ info	×	×	×	×	×	×	×	N/A	×	N/A	N/A
grandmother	×	×	×	×	×	×	×	N/A	✓	N/A	N/A
mother	×	×	×	×	×	×	×	N/A	✓	N/A	N/A
muon*	✓	✓	=obslev	✓	✓	✓	✓	✓	✓	$= 0$	×

Now:

Track	x	y	z	$\frac{p_x}{p}$	$\frac{p_y}{p}$	$\frac{p_z}{p}$	E	t	pID	l	E_{loss}
primary	$= 0$	$= 0$	✓	✓	✓	✓	✓	$= 0$	✓	$= 0$	N/A
add. μ info	✓	✓	✓	✓	✓	✓	✓	N/A	✓	N/A	N/A
grandmother	✓	✓	✓	✓	✓	✓	✓	N/A	✓	N/A	N/A
mother	✓	✓	✓	✓	✓	✓	✓	N/A	✓	N/A	N/A
muon*	✓	✓	=obslev	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: S
ummary of the new and corrected information introduced with the direct CORSIKA readout option. The add. μ info track is the muon at its production point. ■ were previously either wrong or placeholders and N/A marks the information not available from CORSIKA. In the table only muon tracks are shown because gSeaGen for now only propagates μ from CORSIKA, however other particles are available at the sea level (not only neutrinos!). pID is the particle ID in PDG numbering scheme, t is the time and l is the travelled distance. Obslev stands for the observation level, which in this case is the sea level.

```
{ 85 },      // decaying mu+ at start
{ 86 },      // decaying mu- at start
{ 95 },      // decaying mu+ at end
{ 96 },      // decaying mu- at end
{ 9900 },    // Cherenkov photons on particle output file
             // (NOT there without the CERENKOV option, which)
             // (is not used for KM3NeT CORSIKA simulations)
```

Listing 1: CORSIKA-specific particle IDs as implemented in GSeaCORSIKAFileFlux.cxx

For these, additionally a special Status value is required to unambigously differentiate between e.g. $\eta \to \pi^+\pi^-\pi^0$, $\eta \to \pi^+\pi^-\gamma$ and a regular η. This has been implemented in km3net-dataformat in *definitions/trkmembers.csv*:

```
int,TRK_ST_FAKECORSIKA,21,fake particle from corant/CORSIKA
    to add parent information (gseagen)
int,TRK_ST_FAKECORSIKA_DEC_MU_START,22,fake particle from
    CORSIKA: decaying mu at start (gseagen)
int,TRK_ST_FAKECORSIKA_DEC_MU_END,23,fake particle from
    CORSIKA: decaying mu at end (gseagen)
int,TRK_ST_FAKECORSIKA_ETA_2GAMMA,24,fake particle from
    CORSIKA: eta -> 2 gamma (gseagen)
int,TRK_ST_FAKECORSIKA_ETA_3PI0,25,fake particle from CORSIKA
    : eta -> 3 pi0 (gseagen)
int,TRK_ST_FAKECORSIKA_ETA_PIP_PIM_PI0,26,fake particle from
    CORSIKA: eta -> pi+ pi- pi0 (gseagen)
int,TRK_ST_FAKECORSIKA_ETA_2PI_GAMMA,27,fake particle from
    CORSIKA: eta -> pi+ pi- gamma (gseagen)
int,TRK_ST_FAKECORSIKA_CHERENKOV_GAMMA,28,fake particle from
    CORSIKA: Cherenkov photons on particle output file (
    gseagen)
```

Listing 2: Definition of CORSIKA-specific particle IDs in km3net-dataformat

One additional information related to parent-daughter relation between particles or particle history in general, which was introduced, are the hadronic and electromagnetic generation counters associated with tracks. This information had to be added to the *usr* field of the track, since there was no space left elsewhere. It is saved under *generation_counter* name. There are in fact 4 different counters available from CORSIKA and they are saved in the following way:

- Grandmother track: electromagnetic (EM) generation counter of the mother particle (yes, mother particle, <u>not</u> grandmother particle!)

- Mother track: hadronic generation counter of the mother particle
- Muaddi track: hadronic generation counter of the muon
- Muon track: extended EM generation counter of the muon

The details of how these counters relate to each other and are incremented are rather complicated and one should best consult [1] for more details.

As already mentioned, CORSIKA does not only produce muons at the sea level. It also produces neutrinos and other particles, e.g. $\bar{\Sigma}^-$, $\bar{\Lambda}$, $\bar{p}$, $\bar{n}$, $K^\pm$, $\pi^\pm$, K_L^0, K_S^0. These particles are naturally only a small fraction (2.7%) of all produced particles and in most cases can be safely ignored. Nevertheless, now it is possible to control what is saved in the output with the combination of `-write` and `-save` options. The possible options are:

1. `-write 1` : only μ at the can are saved
2. `-write 2` : only μ at the can and particles at the sea level are saved
   (a) `-save mu` : at the sea level only μ are saved
   (b) `-save nu` : at the sea level only ν are saved
   (c) `-save lep` : at the sea level μ and ν are saved
   (d) `-save all` : at the sea level all particles are saved

Naturally, if neutrino propagation for files produced with CORSIKA will be added, `-save nu` will also save neutrinos at the can level as well.

3.1.1 Status values

There are several categories of particles, separated by assigning specific Status values. At the sea level the following ones are possible:

```
                // // uncomment for neutrinos to be included for propagation:
                // case 66: // (anti-)neutrino
                // case 67: // (anti-)neutrino
                // case 68: // (anti-)neutrino
                // case 69: // (anti-)neutrino
```

```
            // case 133: // (anti-)neutrino
            // case 134: // (anti-)neutrino
            case 5: // (anti-)muon
            case 6:    { AuxTrack.Status = -1; break;  } // (anti-)muon
            case 85:
            case 86:   { AuxTrack.Status = -22; break; } // decaying mu at start (muon that does NOT reach the observation level!)
            case 95:
            case 96:   { AuxTrack.Status = -23; break; } // decaying mu at end   (muon that does NOT reach the observation level!)
            case 71:   { AuxTrack.Status = -24; break; } // eta -> 2 gamma
            case 72:   { AuxTrack.Status = -25; break; } // eta -> 3 pi0
            case 73:   { AuxTrack.Status = -26; break; } // eta -> pi+ pi- pi0
            case 74:   { AuxTrack.Status = -27; break; } // eta -> pi+ pi- gamma
            case 9900: { AuxTrack.Status = -28; break; } // Cherenkov photons on particle output file
            default:   { AuxTrack.Status = -21; break; } // all other parents/muaddi
```

Listing 3: Status values assigned to different output tracks from CORSIKA

As one may note, the neutrinos are currently assigned Status -21. This is since neutrino propagation is not yet supported for CORSIKA input. The only particles that will be later propagated are the ones with Status -1, i.e. currently only (anti-)muons. After a muon is succesfully transported to the can, the track at the can with Status 1 will be added and the status of the track at the sea will change to -1001. There is also a Status -2, which is only used internally in gSeaGen as an intermediate one, marking muons that reach the depth of the top cap of the can (this status should never appear in the output files!).

3.2 Weight calculation

The weighting of the CORSIKA events is done using the w_3 weights, which are computed as follows:

$$w_3 = w_2 \cdot \phi_{\mathsf{CR}} = A_{\mathsf{gen}} \cdot I_\theta \cdot I_E \cdot E^\gamma \cdot T_{\mathsf{gen}} \cdot \phi_{\mathsf{CR}} \tag{1}$$

,following the recipe from the taglist (for the explaination of symbols, please follow the link to the pdf). In the code it is implemented as follows:

```
//____________________________
void GSeaCORSIKAFileFlux::ComputeWeights(GSeaEvent * SeaEvt){

  // Compute the weights
  // (accodring to https://simulation.pages.km3net.de/taglist/taglist.pdf)

  // fAgen is w1

  double ITheta = 2. * kPi * fabs(fGenPar->PrimCtMax - fGenPar->PrimCtMin);
  double Gamma = fabs(fGenPar->Alpha);
  double IEprim;

  if (Gamma == 1) IEprim = log(MaxEPrim / MinEPrim); // yes, this should be natural logarithm!
  else IEprim = (pow(MaxEPrim, 1. - Gamma) - pow(MinEPrim, 1. - Gamma)) / (1. - Gamma);

  SeaEvt->GenWeight = SeaEvt->Agen * ITheta * IEprim * pow(EPrim, Gamma) * fGenPar->TGen; // w2
  SeaEvt->EvtWeight = SeaEvt->GenWeight * CRFlux(EPrim, PrimID, fGenPar->CRModel); // w3

  this->GetEvtTime(SeaEvt->EvtTime,SeaEvt->EvtTime+1);

  LOG("GSeaCORSIKAFileFlux", pDEBUG) <<"SeaEvt->Agen: "<<SeaEvt->Agen;
  LOG("GSeaCORSIKAFileFlux", pDEBUG) <<"SeaEvt->GenWeight: "<<SeaEvt->GenWeight;
  LOG("GSeaCORSIKAFileFlux", pDEBUG) <<"SeaEvt->EvtWeight: "<<SeaEvt->EvtWeight;

  return;
}
```

Listing 4: Weight formula implementation in GSeaCORSIKAFileFlux

SeaEvt→Agen is computed differently depending on the ‘-rt‘ option. For ‘-rt proj‘ is it the sum of projected areas of the top and side of the can. One can also see how fSeaEvent→DistaMax impacts the projected areas:

```
if (fGenPar->RTOpt.compare("proj")==0) { // proj option
  //  the can cyllinder is increased by the xy and z projections of fDistaMax:
  double height = (fGenPar->Can2-fGenPar->Can1+fGenPar->HBedRock) + fSeaEvent->DistaMax * sqrt( 1. - fFileDriver->OtherTracks[0].D3 * fFileDriver->OtherTracks[0].D3 );
  double radius = fGenPar->Can3 + fSeaEvent->DistaMax * fabs(fFileDriver->OtherTracks[0].D3);
  double xaux,yaux,zaux;  // position sampled at the can surface
  double yaux2;

  //  the projected area of the cyllinder (according to e.g. eq. 10 in DOI:10.1364/AO.42.006710):
  double AreaTop  = kPi * radius * radius * fabs(fFileDriver->OtherTracks[0].D3) ;
  double AreaSide = 2. * height * radius * sqrt( 1. - fFileDriver->OtherTracks[0].D3 * fFileDriver->OtherTracks[0].D3 );
  fSeaEvent->Agen = AreaTop + AreaSide;
```

Listing 5: Computation of the generation area using the ‘-rt proj‘ option

On the other hand, for ‘-rt can‘ option the generation area SeaEvt→Agen is a circle of radius R_{T}, oriented along the primary direction:

```
else if (fGenPar->RTOpt.compare("can")==0) {  // can option

  double Can21 = fGenPar->Can2-fGenPar->Can1;
  double RT = 0.5*sqrt(Can21*Can21+4.*fGenPar->Can3*fGenPar->Can3); // diagonal of the can

  double Xc=0.,Yc=0.;
  double Zc=-fGenPar->SeaBottomRadius+fGenPar->Can1;
  double R=fGenPar->SeaBottomRadius+fGenPar->SiteDepth;

  // generate the primary vertex (fX0,fY0,fZ0: detector - origin coordinates)
  double xDif = fFileDriver->GetfX0()-Xc;
  double yDif = fFileDriver->GetfY0()-Yc;
  zDif = fFileDriver->GetfZ0()-Zc;
  double b  = fFileDriver->OtherTracks[0].D1*xDif + fFileDriver->OtherTracks[0].D2*yDif + fFileDriver->
```

```
OtherTracks[0].D3*zDif;
 double c  = xDif*xDif+yDif*yDif+zDif*zDif-R*R;
 RL = fabs(-b-sqrt(b*b-c));

 RT+=fSeaEvent->DistaMax;
 fSeaEvent->Agen = kPi*RT*RT;
```

Listing 6: Computation of the generation area using the '-rt can' option

The first option, '-rt proj' is the default recommendation, since it is better suited to cover only the necessary generation area and thus provide better statistics (more events reaching the can). Figures 1 and 2 show how the area of the top cap of the can is projected onto the sea surface. In Figures 3 and 4 the same is shown for the side area of the can. The primary direction is the reference point for CORSIKA, as all muon positions at sea are defined with respect to it and this is why it is shown in the mentioned Figures. As one may note in Fig. 3, the curvature of the Earth may play a role (for curved sea surface the distance to the can is slightly smaller) for very horizontal events and is properly accounted for.

The final event weight is computed as follows:

$$w = \frac{w_3}{(1 + n_{\mathsf{retries}}) \cdot n_{\mathsf{generated\_showers}}} \tag{2}$$

, where $n_{\mathsf{generated\_showers}}$ is the number of showers that were generated in CORSIKA and n_{retries} is the number of additional chances given to the event to reach the can (default: 0; see Section 3.3.1). This applied to the events will produce distributions in units of $\frac{1}{\mathsf{s}}$. It has to be noted that w is always computed independently for each primary and also $n_{\mathsf{generated\_showers}}$ should be the number of shower generated for the particular primary, not the sum over all primaries! If the two sub-productions have overlapping E_{primary} ranges for some primary, the $n_{\mathsf{generated\_showers}}$ has to be the sum over the two sub-productions (but again only for that one primary!).

3.2.1 Evaluation of DistaMax

In v7, there has been some optimisations, especially regarding the EeV showers. The code was optimized in terms of memory handling to avoid running out of RAM. Moreover, shooting of the showers at the can has been improved. The value of *fDistaMax* (a parameter estimated at the sea level) is a measure of the lateral spread of muons around the primary axis. It is used

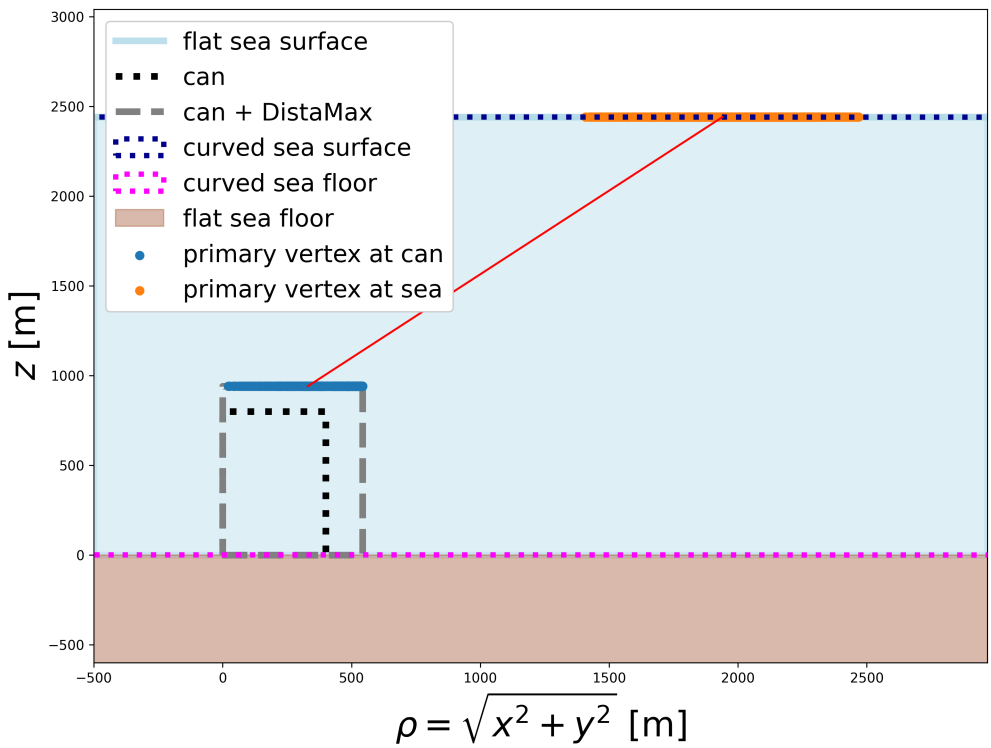

Figure 1: S
ketch of the projection of can top cap area onto the sea surface (after increasing it by the appropriate DistaMax projection onto the horizontal direction) in the side view. Plot made using this jupyter notebook.

to additionally increase the can dimensions to take into account the cases, where the showers hit the can but only with their edge. Previously *fDistaMax* was just computed from all the muon positions. Now it is evaluated only for the muons that have sufficient range (enough energy) to reach the can (see Figure 5). This allows to keep in some cases even 4 times more EeV showers than before (64% instead of 15%). What also changed is the formula used to compute the distance from the shower axis. The original one assumed that muons are travelling parallel to the primary axis and in the same plane as the primary, which is not exactly true (although not a very bad approximation).

A sketch for a single muon is shown in figure 6. The new corrected formula computes *fDista* (which is used to estimate *fDistaMax* for each shower) in a more general way. Instead of

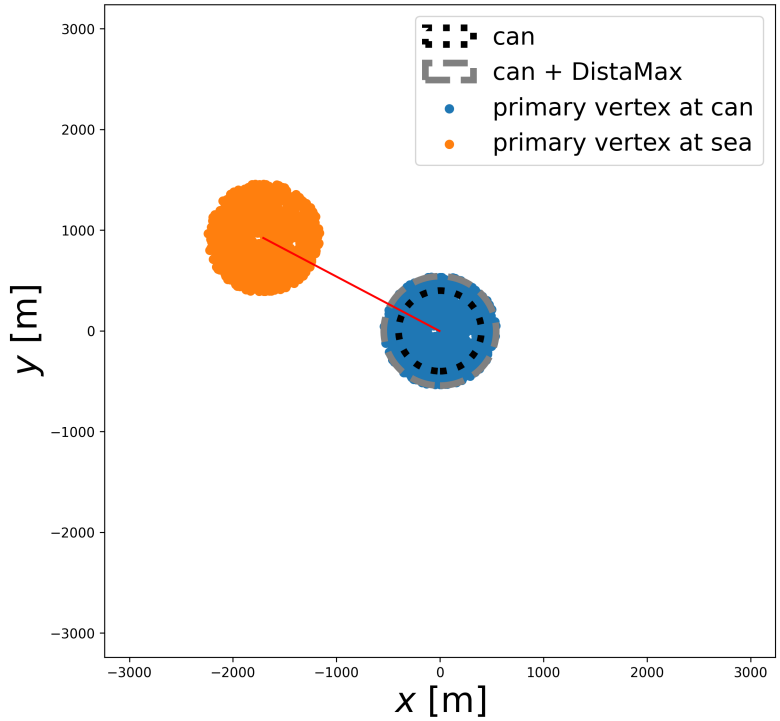

Figure 2: S
ketch of the projection of can top cap area onto the sea surface (after increasing it by the appropriate DistaMax projection onto the horizontal direction) in the top view. Plot made using this jupyter notebook.

```
TMath::Sqrt(TMath::Power(AuxTrack.V1,2)+TMath::Power(AuxTrack
    .V2,2))*TMath::Abs(Tracks[0].D3)
```

now

```
double V2332diff = (OtherTracks[0].V2-AuxTrack.V2)*(
    OtherTracks[0].D3) + (AuxTrack.V3-OtherTracks[0].V3)*(
    OtherTracks[0].D2);
double V3113diff = (OtherTracks[0].V3-AuxTrack.V3)*(
    OtherTracks[0].D1) + (AuxTrack.V1-OtherTracks[0].V1)*(
    OtherTracks[0].D3);
double V1221diff = (OtherTracks[0].V1-AuxTrack.V1)*(
    OtherTracks[0].D2) + (AuxTrack.V2-OtherTracks[0].V2)*(
    OtherTracks[0].D1);
sqrt( V2332diff*V2332diff + V3113diff*V3113diff + V1221diff*
    V1221diff )
```

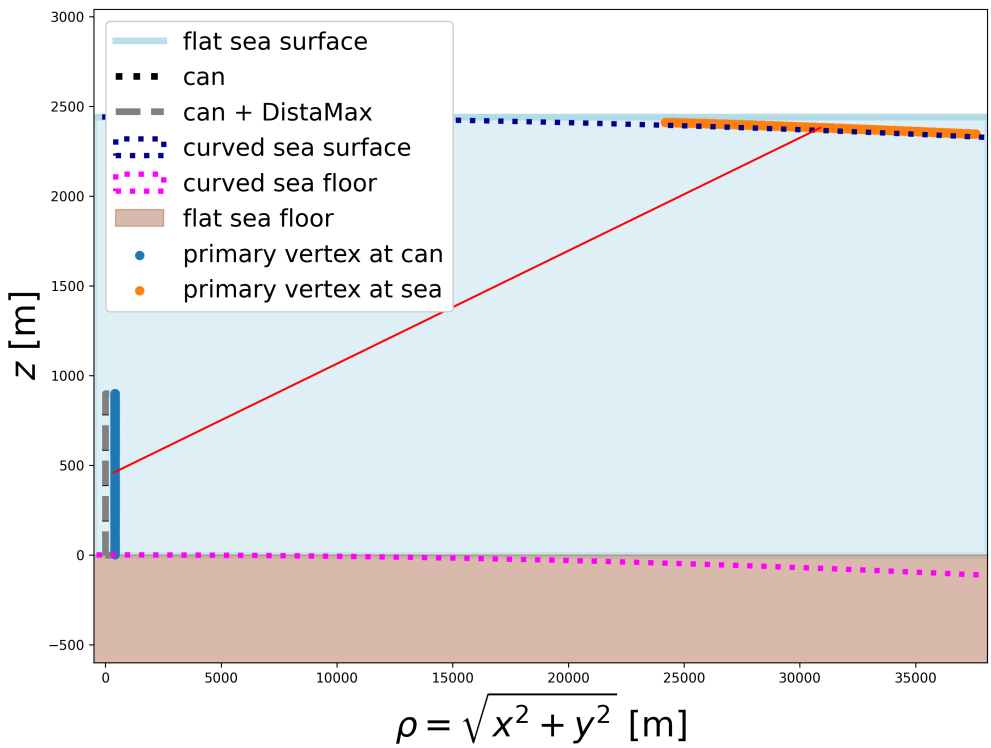

Figure 3: S
ketch of the projection of can side area onto the sea surface (after increasing it by the appropriate DistaMax projection onto the vertical direction) in the side view. Plot made using this jupyter notebook.

is done, which is derived from the general formula for the distance of a point from a line (as given e.g. here).
One further advance has been made in the *fDistaMax* evaluation. Previously, *fDistaMax* was incremented by 100m to account for the rare but possible case that a muon trajectory does not intersect the can but it scatters so luckily that it still ends up in the can. The muon deflection is naturally a function of the initial muon energy and of the travelled distance and 100m is in many cases an overkill, for which the price is paid in lower statistics at the can. With this motivation, the muon deflections for slant depths up to 40km and energies betweend 1 and 10^{12} GeV were sampled using PROPOSAL [3] and a two-step fit was performed. The details may be found in gseagen/issues/84. The fit was then implemented in place of the 100m "safety margin" as follows:

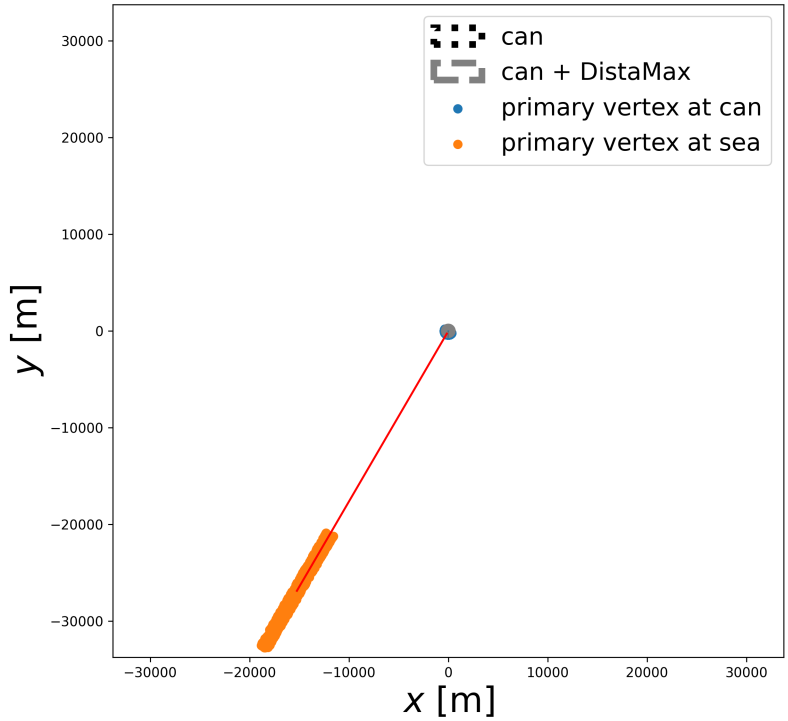

Figure 4: S
ketch of the projection of can side area onto the sea surface (after increasing it by the appropriate DistaMax projection onto the vertical direction) in the top view. Plot made using this jupyter notebook.

```
double V2332diff = (OtherTracks[0].V2-AuxTrack.V2)*(OtherTracks[0].D3) + (AuxTrack.V3-OtherTracks[0].V3)*(OtherTracks[0].D2);
double V3113diff = (OtherTracks[0].V3-AuxTrack.V3)*(OtherTracks[0].D1) + (AuxTrack.V1-OtherTracks[0].V1)*(OtherTracks[0].D3);
double V1221diff = (OtherTracks[0].V1-AuxTrack.V1)*(OtherTracks[0].D2) + (AuxTrack.V2-OtherTracks[0].V2)*(OtherTracks[0].D1);
// fDista is computed as a distance of a point (muon) from the line (primary trajectory)
// pow(10.,...) is a fitted estimation of the lateral muon deflection (see https://git.km3net.de/simulation/gseagen/-/issues/84)
fDistaMax = max( fDistaMax, sqrt( V2332diff*V2332diff +
```

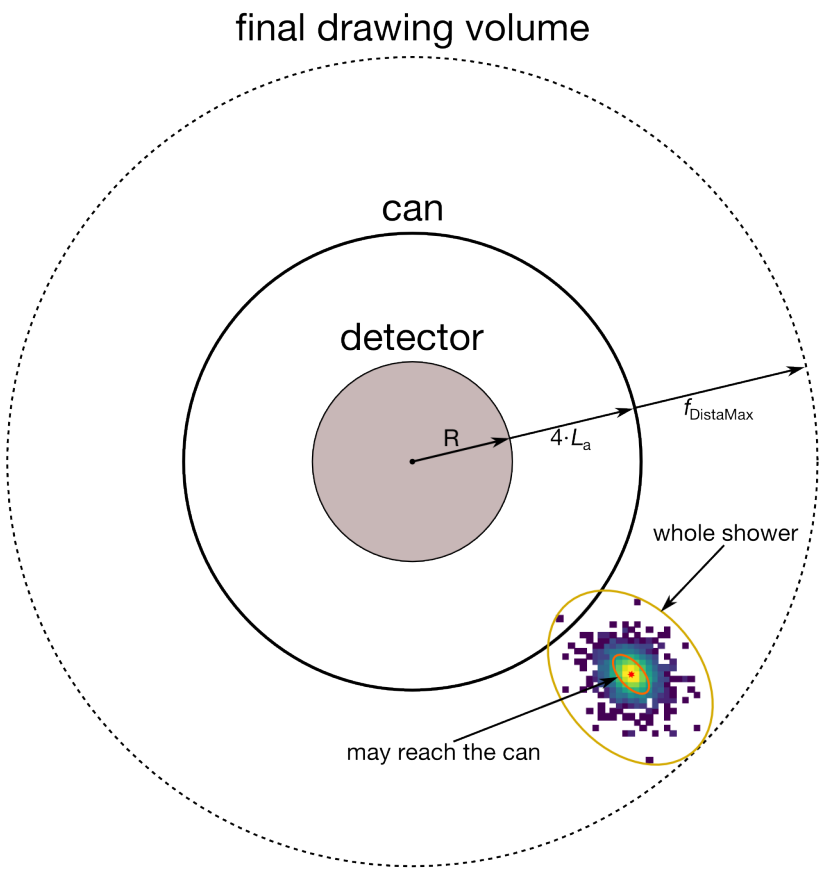

Figure 5: S
ketch of the lateral geometry of shooting the shower at the can. Drawn for the case of a perfectly vertical shower for simplicity.

```
V3113diff*V3113diff + V1221diff*V1221diff ) + pow(10.,(
log10(fMaxDistToCan)*0.67971202-3.07154976)*log10(AuxTrack
.E)+fMaxDistToCan*(-753778.571)+3.97088084) );8.571)
+3.97088084) );
```

Listing 7: Current implementation of the *fDistaMax* calculation.

As one may note, the lateral deflection safety margin is now applied to each muon individually and is computed as a function of the distance from the can (the maximal value is taken as a conservative approach) and initial muon energy.

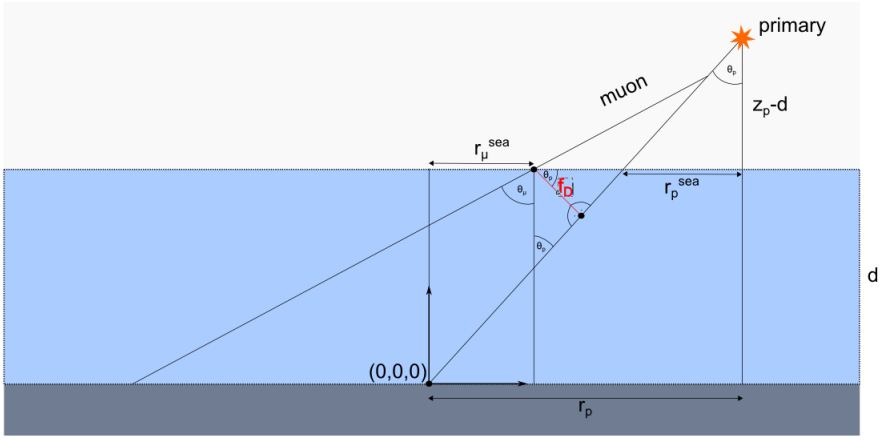

Figure 6: S
ketch of the geometry with a single muon deviating slightly (in the figure it is exaggerated for the sake of readability) from the primary axis.

3.3 New functionalities

There are several new functionalities designed with the intention of improving CORSIKA statistics or facilitating the generation of the new productions.

3.3.1 Retrying showers

The first one is the option to give showers "second chances", activated with `-chances`. This means that e.g. `-chances 100` will give to each shower that misses the can 100 additional chances to hit the can (the default is 0). The number of additional retries it took for an event to hit the can is saved in accordance with km3net-dataformat in *definitions/w2list_gseagen.csv*:

```
int,W2LIST_GSEAGEN_N_RETRIES,19,Number of extra chances given
    to each CORSIKA shower to hit the can
```

This number has to be taken into account when computing the weights, as indicated in Equation 2.

3.3.2 Recycling showers

The second new functionality is still experimental and is NOT recommended for beginners. It allows one to rotate CORSIKA showers around x, y, and

z axes by specifying respective roll, pitch and yaw angles in degrees or radians. The syntax is `--rot-showers UNIT:yaw,pitch,roll`, i.e. if one does `--rot-showers DEG:0.04,0.1,0.003`, gSeaGen will rotate each shower by 0.04° around z axis, 0.1° around y axis and by 0.003° around x axis. The small angles in the example are not by chance, rotating the showers by large angles will ruin the physics as the path travelled by the muons in the air will be completely off given the new direction! For this reason, this option should NOT be used unless the user really knows what he or she is doing. The applied rotation is saved in *definitions/w2list_gseagen.csv*:

```
int,W2LIST_GSEAGEN_CUSTOM_YAW,20,user-specified rotation of CORSIKA showers (around z-axis)
int,W2LIST_GSEAGEN_CUSTOM_PITCH,21,user-specified rotation of CORSIKA showers (around y-axis)
int,W2LIST_GSEAGEN_CUSTOM_ROLL,22,user-specified rotation of CORSIKA showers (around x-axis)
```

Listing 8: Custom rotation as defined in km3net-dataformat.

3.3.3 Reduction of verbosity

Since the TeV subproduction requires simulating millions of showers per run, the log files produced were huge and exceeded the size of the actual CORSIKA output files. This was adressed by introducing the `--corsika-less-verbose 1` option, which will print only information about every 100000-th generated event (instead of 500-th) and will completely skip the information about reading the run or event header.

3.3.4 Muon range tolerance

To aid the attempts to extract as much statistics out of the already available TeV files, an option has been added to allow a little bit more generous treatment of low-energy muons. By doing e.g. `--muon-range-tolerance 50.0`, one can increase the tolerance range for muon ranges by 50.0 meters, hence allowing the muons that have a small probability to hit the can to still get a chance (normally muons with insufficient range are not propagated).

4 Propagation

The muon propagation has been reworked in v7.2.0 and is now performed in the following stages:

1. Muons at the sea level:

   - When reading muons in from a CORSIKA file, *fDistaMax* is evaluated (*fDista* is the lateral distance of the muon from the shower axis, i.e. the primary trajectory). This is later used both for propagation and weighting. There is an extra "safety margin" added to *fDista* for each muon to accomodate for the possibility that muon trajectory could not be crossing the can but the muons could scatter into the can while propagating (see Sec. 3.2.1).
   - Secondary muons from CORSIKA output are exactly at the sea level. Their xy positions are defined with respect to the primary vertex at the sea level (extrapolated from the first interaction point). One has to shift the whole shower on the sea surface to a position, from which it actually has a chance to hit the can. There are two options to do that:
     - `-rt 'can'`: older option, samples showers on a circle covering the detector can projected onto a plane perpendicular to the primary direction. It is described in in some more detail in this pdf. This is not a recommended mode for CORSIKA, since it is inefficient (directions of showers sampled on some part of the area of the circle do not intersect the can, because the projected area of the can is NOT a circle!).
     - `-rt 'proj'`: recommended option, it samples showers on the can surface area projected flat on the curved sea surface (before 'v7.2.0' it was projected between $z_{\mathsf{sea}} - \frac{h_{\mathsf{can}}}{2}$ and $z_{\mathsf{sea}} + \frac{h_{\mathsf{can}}}{2}$, which caused some of the muons in the shower to have longer and some to have shorter path to travel through water). A 3D visualisation of the projection is implemented in the following jupyter notebook. The projection works as follows:
       * we first compute the projected areas of the top cap (A_{top}) and side of the can (A_{side}).
       * we randomly pick if we'll land on the top cap or on the side (the probabilities are proportional to A_{top} and A_{side})

    * we pick a random point on the surface of the can top cap or the side (depending on what we have just rolled)
    * we trace back from this point to the sea surface (using the primary direction)
    * after we know the position at the sea level, we shift the shower to be centered on that point (basically this will be where the primary particle would intersect the sea surface)

2. First stage of propagation: sea level → top cap height of the can:

   - all muons that have sufficient range (the ones that don't are ignored) are propagated from the sea surface to h_{can} (height of the top cap of the can).
   - for each muon it is checked, whether it already hit, missed or did not reach the can yet.
     - if there are muons in the can or pointing at it but not inside yet (they point at the side of the can), we move to the next stage of propagation.
     - if no muon reached the can and none has a chance to do so in 3., but there are muons that just missed the can, we shift horizontally (this counts as retrying and is optional, i.e. only done if `-chances 1` or bigger; see Sec. 3.3.1)).
     - if there is no point in further propagation or shifting, we can also retry by starting over for the current event. Then the event is reset and a new position at the sea surface is picked (this is also done only if `-chances` option is used).
     - if everything fails, the event is unsuccessful and we move to the next one.
     - if the event was successful and there is nothing left to propagate (i.e. all muons either hit the top of the can or failed), we proceed to compute the weights and write it to the output file.

3. Second stage of propagation: top cap height of the can → the can

   - the propagation is picked up from where it finished in 2.
   - if no muons reach the can, the event may be restarted (as mentioned in 2.).

- if the event failed, we move to the next one.
- if the event was successful, we proceed to compute the weights and write it to the output file.

5 Header

The header has been extended to include the information about the hadronic interaction models, CORSIKA version, energy cuts, site coordinates, cosmic ray flux model, primary particle, generation spectrum. An example is shown below:

```
>>> import km3io as ki
>>> r=ki.OfflineReader('DAT000004_PeV_C.gSeaGen.4.aa.root')
>>> print(r.header)
MC Header:
  GXMLPATH: /pbs/throng/km3net/src/gSeaGen/genie3.00.02-hedis-km3net/Generator//genie_xsec/gSeaGen
  coord_site: coord_site(field_0=0.747, field_1=0.10763)
  cut_nu(Emin=300.0, Emax=0, cosTmin=0, cosTmax=0)
  cut_primary(Emin=900000.0, Emax=110000000.0, cosTmin=-0.0523, cosTmax=-1.0)
  cut_seamuon(Emin=1000.0, Emax=0, cosTmin=0, cosTmax=0)
  depth: 2440.0
  drawing: surface
  fixedcan(xcenter=0.0, ycenter=0.0, zmin=0.0, zmax=476.5, radius=430.7)
  flux(type=1000060120, key='GST3', file_1='/sps/km3net/users/kakiczi/tests/lateral_spread/DAT000004_PeV_C', file_2=None)
  genvol(zmin=0, zmax=0, r=0, volume=0, numberOfEvents=2000.0)
  physics(program='CORSIKA', version='UrQMD', date='SIBYLL-2.3d', time=None)
  physics_1: physics_1(field_0='gSeaGen', field_1='PROPOSAL', field_2='6.1.5')
  primary: 1000060120
  seed(program='CORSIKA', level=1000004, iseed=5000004)
  seed_1: seed_1(field_0='gSeaGen', field_1=-1554047128)
  simul(program='CORSIKA', version=7.741, date=210513, time=0)
  simul_1: simul_1(field_0='gSeaGen', field_1='v7.2.0-rc9', field_2=211112, field_3=2300)
  source_mode: FILE
```

```
  spectrum(alpha=2.0)
  start_run(run_id=4)
>>>
```

Listing 9: Header example for a PeV carbon file (printed using km3io).

References

[1] D. Heck, T. Pierog, Extensive air shower simulation with corsika: A user's guide (version 7.7400) (May 2021).
URL https://web.ikp.kit.edu/corsika/usersguide/usersguide.pdf

[2] T. K. Gaisser, T. Stanev, S. Tilav, Cosmic ray energy spectrum from measurements of air showers, Front. Phys. 8 8 (2013) 748–758. doi:10.1007/s11467-013-0319-7.

[3] J. H. Koehne, K. Frantzen, M. Schmitz, T. Fuchs, W. Rhode, D. Chirkin, J. Becker Tjus, PROPOSAL: A tool for propagation of charged leptons, Comput. Phys. Commun. 184 (2013) 2070–2090. doi:10.1016/j.cpc.2013.04.001.