CHASM (CHerenkov Air Shower Model)

Isaac Buckland[a,*] **and Douglas Bergman**[a]

[a] *Dept. of Physics & Astronomy and High Energy Astrophysics Inst., University of Utah, Utah, USA*

E-mail: isaacbuckland@cosmic.utah.edu, bergman@physics.utah.edu

CHASM (CHerenkov Air Shower Model) is a python package which leverages the universality of charged particles in an extensive air shower to produce a deterministic prediction of the Cherenkov light signal for a given shower profile and geometry. At sampled points throughout the domain of all shower development stages and altitudes, the angular and yield distributions of Cherenkov light have been calculated at an array of distances from a shower axis. Chasm accesses and interpolates between these distributions at runtime to produce the aggregate signal from the whole shower at user defined telescope locations. This paper gives a detailed description of the methods used to compute the Cherenkov distribution tables from universal charged particle energy and angular distributions. It also describes the workflow of CHASM itself, how a shower and axis is constructed, and how a user can customize the simulation. Finally, this paper describes how to install CHASM both from source code on GitHub and using pip (python's built in package manager) and how to use it.

38th International Cosmic Ray Conference (ICRC2023)
26 July - 3 August, 2023
Nagoya, Japan

*Speaker

1. Universal Cherenkov Angular Distribution

The relative number of charged particles at a certain stage of EAS development traveling in a certain direction relative to the shower axis is a convolution of the universal charged particle energy and angular distributions, as the angular distribution is energy dependent. These charged particles produce Cherenkov photons if their energy exceeds the local Cherenkov threshold. These photons are produced into a cone whose opening angle is determined by the local index of refraction. Figure 1 shows the shower axis direction as $\hat{\mathbf{n}}$, shower charged particle direction as $\hat{\mathbf{e}}$, and Cherenkov radiation propagation direction as $\hat{\gamma}$. There are always two charged particle directions with the same polar, but different azimuthal angles which produce Cherenkov light at the same polar angle from the shower axis. The calculation of the relative contributions of these cones at directions relative to the shower axis, as well as the average number of photons produced per charged particle, are described in detail in our recent publication in Astroparticle Physics [1].

Figure 2 shows the Cherenkov angular distribution per unit solid angle at maximum development $t = 0$ and at $\delta = n - 1 = 10^{-4}$ where n is the index of refraction. The integral over charged particle velocities was performed both directly, and using a Monte Carlo method. The bins used to collect the Monte Carlo data were chosen to center around the angles tabulated by the convolution integral. While there are slight differences at small angles the peak angle matches exactly [1]. These distributions were tabulated at an array of shower development stages and indices of refraction and saved as compressed numpy files for access by the CHASM (CHerenkov Air Shower Model) 7 program.

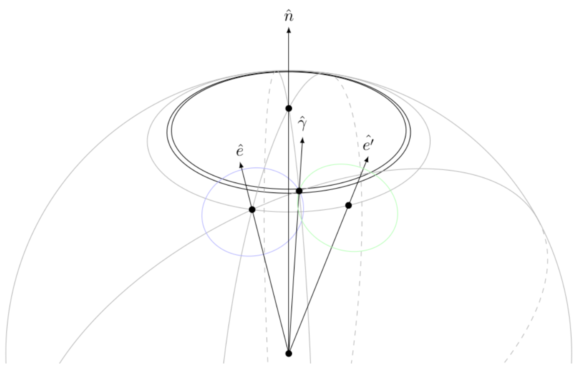


Figure 1: Spherical geometry of Cherenkov cones.

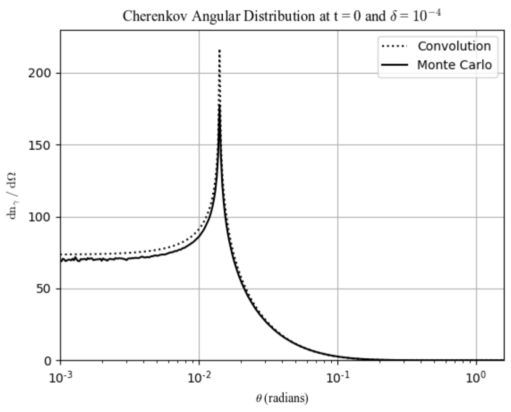


Figure 2: Comparison between Monte Carlo and direct integration.

2. Shower Mesh Sampling

In addition to the universal distributions describing shower particle energy and propagation direction, particle position (in Moliere radii) relative to the axis is also drawn from a universal distribution commonly known as the NKG (Nishimura-Kamata-Greisen) distribution. Fits to CORSIKA (COsmic Ray SImulations for KAskade [5]) simulations also show a strong dependence of

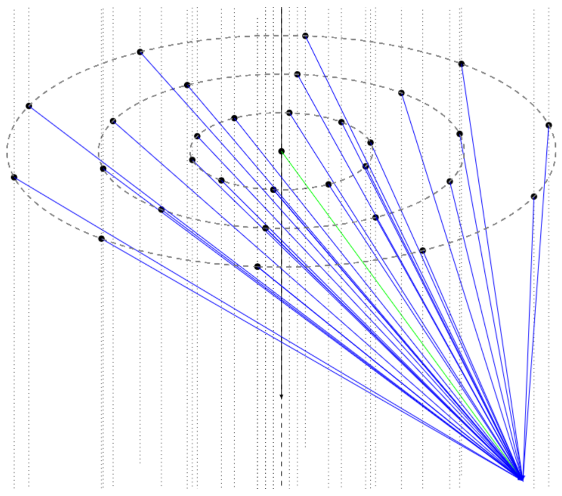


Figure 3: This figure shows the propagation vectors which result when sampling a shower away from the axis. The shower axis is the black vector pointing downward. The green vector, directly from the shower axis to a counter location (i.e. the angle it makes with the axis) is what is sampled without the mesh. The blue vectors begin from points on rings with various radii in Moliere unit space.

the lateral distribution on particle energy [4]. Since both particle Cherenkov production and particle location depend on shower energy, we see a modest dependence of the Cherenkov angular distribution on distance from the axis. The Monte Carlo method described in section 1 was extended to assign charged particles a radii from the energy dependent version of the NKG distribution. This allows for calculation of Cherenkov angular distributions at various distances from the shower axis.

When simulating the aggregate shower Cherenkov signal at a detector, early in the EAS development, or when the distance to the point of interest on the shower axis is large, it is sufficient to approximate the whole shower, i.e. all particles, as lying along the axis itself. However, if a detector is close to an EAS footprint, it becomes necessary to sample Cherenkov production at various distances from the shower axis in order to accurately reproduce the shape of the Cherenkov pulse waveform. CHASM 7 will distribute charged particles to logarithmically spaced rings in Moliere unit space. Figure 3 shows how Cherenkov photon travel vectors change when sampled away from the axis. This effect is shown in figure 5. Using the CHASM 7 *mesh* option not only changes which distribution is sampled, but also which angle is sampled from the distribution.

3. Comparing to CORSIKA IACT

Showers were generated using CORSIKA 7.69 compiled with QGSJETII with Cherenkov signal generated by the IACT extension [5]. Cherenkov counters were defined at increasing distances from the shower core. A set of showers was generated both with and without atmospheric extinction. CHASM simulations were generated using the same shower geometry and profiles. The CORSIKA shower referenced in figures 4 and 5 is a proton shower with primary energy 5×10^6 GeV, X_{max} of 527 g/cm^2, N_{max} of 3.12×10^6 particles. Figure 4 is a comparison of Cherenkov lateral distribution, the total number of photons collected from the whole shower at each spherical counting volume, from both CORSIKA and universality. CHASM uses CORSIKA's atmoshperic extinction tables to calculate extinction along Cherenkov photon travel paths. Figure 5 is a comparison of arrival time distribution of Cherenkov photons at just one counter from CORSIKA and CHASM both with and without mesh sampling. Particles produced by charged particles on the side of the axis closer to the detector arrive before ones produced along the axis. The CHASM mesh option accounts for these photons. A Jupyter notebook which imports the CORSIKA data from an IACT file and creates an analogous CHASM simulation is included in the "demo" directory of the CHASM github repository [2].

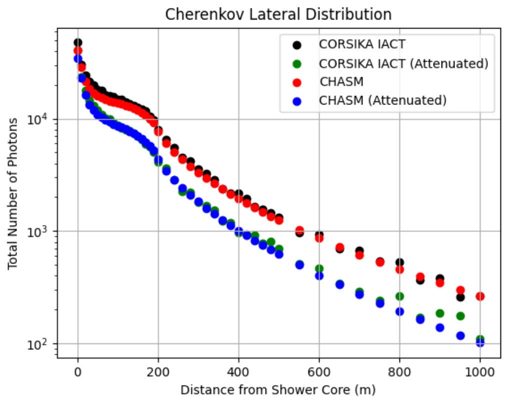


Figure 4: Comparison of lateral distributions.

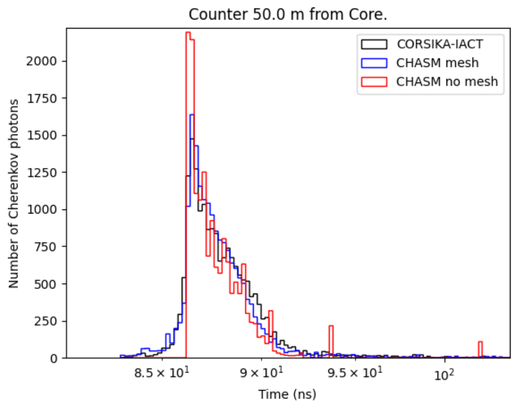


Figure 5: Comparison of photon arrival times.

4. NICHE Monte Carlo Using CHASM Signals

The NICHE (Non-Imaging CHErenkov) detectors are fourteen single-pmt photon counters deployed just to the south of the Telescope Array Middle Drum observatory in the field of view of the TALE telescopes [6]. The array sits at an altitude of 1564 meters. Each NICHE detector has a temporal resolution of five nanoseconds, and a twelve bit FADC (Flash Analog to Digital Conversion) resolution. A detector will trigger if the mean of any eight samples is more than seven sigma above the mean of the previous 1024 samples.

A detector Monte Carlo using CHASM to generate Cherenkov signals is currently in development. Since CHASM can generate shower signals within seconds, simulations of many showers can be produced in reasonable a amount of time. Thus far, shower parameters are generated for sets in various energy bins, and CHASM signals are produced for those showers at the location of each NICHE detector. Photons arriving with a zenith angle greater than 45 degrees are cut. The trigger and FADC electronics are then simulated. These datasets were used to calculate the trigger aperture of the NICHE array. Ten thousand events were generated in each energy bin to calculate the aperture values in figure 6. Future work will involve passing signals to shower reconstruction programs.

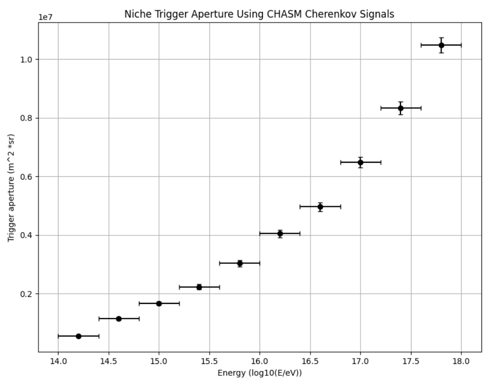


Figure 6: Trigger aperture of the NICHE array.

5. Upward going showers and NuSpaceSim

The first iteration of CHASM is being tested for implementation in nuSpaceSim, a comprehensive neutrino simulation package for space-based and suborbital experiments [8]. Tau neutrinos skimming the Earth may interact via the charged-current interaction in the Earth's crust. The resulting tau particle leaves the Earth and decays, serving as the primary particle in an upward going air shower.

The flexibility of CHASM allows for the same tables to be accessed for development occurring along an upward trajectory. The CHASM demo Jupyter notebook available in the GitHub repository [2] implements CHASM for a hypothetical upward going primary. Figure 7 shows the signal of that shower at an array of counters in orbit facing normal to the shower axis. Since these types of showers will be highly inclined, the curvature of the atmosphere has a significant effect. Figure 8 shows the arrival time distribution at just one counter for the same shower both with and without the CHASM curved atmosphere treatment.

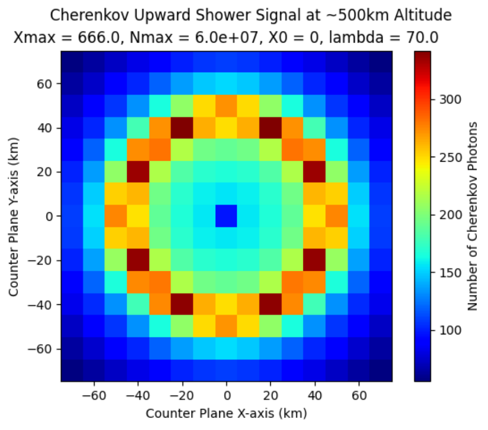


Figure 7: Upward going shower signal with 85 degree zenith angle.

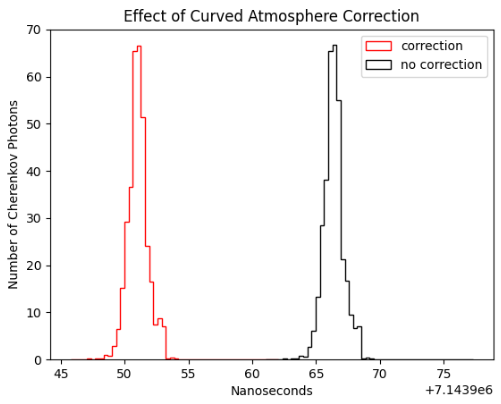
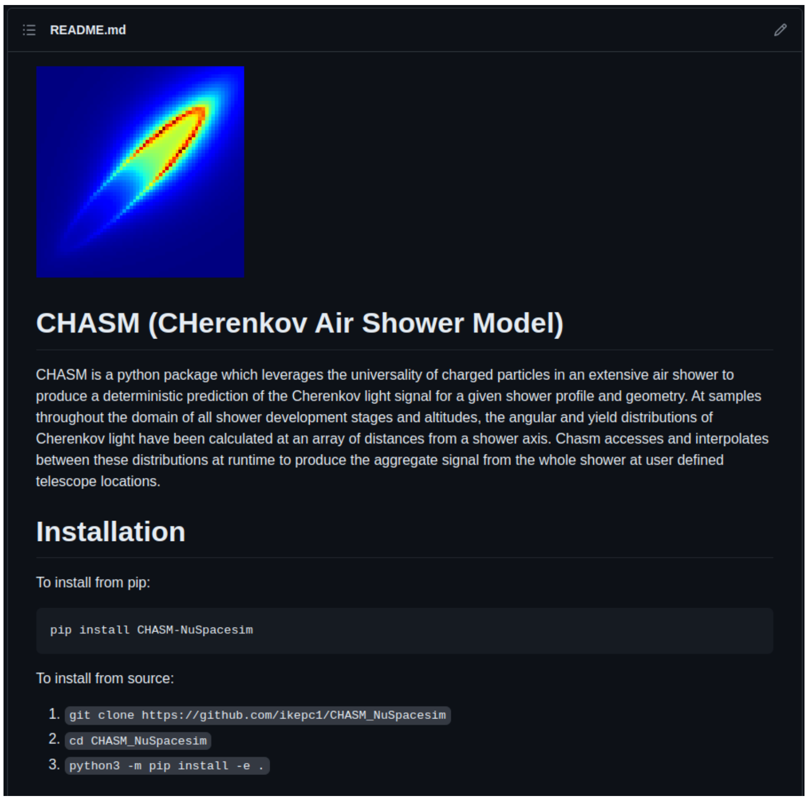


Figure 8: Arrival time distribution.

6. Installing CHASM

CHASM can be installed both from source or using pip. Instructions on how to do so can be found on the github page [2]. A screenshot of the page is shown in figure 9.


README.md

CHASM (CHerenkov Air Shower Model)

CHASM is a python package which leverages the universality of charged particles in an extensive air shower to produce a deterministic prediction of the Cherenkov light signal for a given shower profile and geometry. At samples throughout the domain of all shower development stages and altitudes, the angular and yield distributions of Cherenkov light have been calculated at an array of distances from a shower axis. Chasm accesses and interpolates between these distributions at runtime to produce the aggregate signal from the whole shower at user defined telescope locations.

Installation

To install from pip:

```
pip install CHASM-NuSpacesim
```

To install from source:

1. `git clone https://github.com/ikepc1/CHASM_NuSpacesim`
2. `cd CHASM_NuSpacesim`
3. `python3 -m pip install -e .`

Figure 9: CHASM github page showing the readme file with installation instructions: https://github.com/ikepc1/CHASM_NuSpacesim.

7. CHASM Program

CHASM is a flexible python program in which the user can manually construct a simulation of the Cherenkov light produced by an extensive air shower (EAS). The frame for the simulation itself is named *Simulation* and is available under the CHASM namespace.

```python
import CHASM as ch
sim = ch.Simulation()
```

A user will then add various 'elements' to this simulation. The shower *Axis* defines the direction of the primary particle. The zenith and azimuthal angles (in radians) are defined by the first two positional arguments, respectively. The Cartesian coordinate system of CHASM defines the xy plane as flat on the ground with the z axis vertically upward. The origin of the CHASM coordinate system is where the shower axis intersects with the xy plane, the altitude of which is defined by the third positional argument. If the keyword flag *curved* is set, the depth steps, light propagation, timing, and atmospheric extinction are all calculated using a curved atmosphere.

```
sim.add(ch.DownwardAxis(theta, phi, obs, curved=False))
```

For upward going showers induced by tau neutrinos, the user can also define an upward going axis.

```
sim.add(ch.UpwardAxis(theta, phi, obs, curved=False))
```

The shower profile to be projected onto the axis can be defined using Gaisser Hillas parameters. The positional arguments are depth at maximum, number of shower particles at maximum, depth of first interaction, and the Gaisser Hillas lambda parameter respectively.

```
sim.add(ch.GHShower(Xmax,Nmax,X0,Lambda))
```

A shower profile can also be defined by an array of depths and the number of shower charged particles at each of those depths.

```
sim.add(ch.UserShower(X, nch))
```

Detector locations are either defined as spheres in the same way as in CORSIKA IACT [5] or as flat horizontal apertures. In both cases the user will supply a rank two numpy array of Cartesian vectors to the center of each counter. The user will also define the radius of each counter.

```
sim.add(ch.SphericalCounters(counter_vectors, counter_radius))
sim.add(ch.FlatCounters(counter_vectors, counter_radius))
```

Finally, a user will define the Cherenkov wavelength interval minimum, maximum, and the number of bins with the *Yield* object. For wavelength dependent post-processing, such as ray-tracing, more bins should be used.

```
sim.add(ch.Yield(min_wavelength, max_wavelength , Nbins))
```

Once the simulation has all these necessary elements, the *run* method calculates the Cherenkov signal from each axis step towards each counter. This calculation interpolates between the pre-compiled tables of Cherenkov angular distributions described in section 1. A detailed description of the calculation of a shower's aggregate signal is described in [1]. The run method takes two keyword flags. The first, *mesh* indicates whether to sample the axis using the mesh method described in section 2. The second, *att*, indicates that atmospheric extinction should be taken into account as the photons propagate to the counters.

```
def run(self, mesh: bool = False, att: bool = False) -> ShowerSignal:
```

```
sig = sim.run(mesh=True, att=True)
```

The returned *ShowerSignal* object includes the number of photons coming from each axis point to each counter, their corresponding arrival times, wavelength values, and other attributes such as the profile and arrival cosines for convenience. For use in an interactive python session, only the axis and the counters are included in the container's string representation. The photons array is a rank three numpy array where the first axis is the index of the counter, the second is the index of the wavelength bin, and the third is it's index along the axis. Wavelength is not considered by the CHASM atmospheric refraction, so the times array is only rank two, and thus the same for photons arriving with each wavelength.

```
@dataclass
class ShowerSignal:
    '''This is a data container for a shower simulation's Cherenkov
    Photons, arrival times and counting locations.
    '''
    counters: Counters #counters object
    axis: Axis #axis object
    source_points: np.ndarray = field(repr=False)
    wavelengths: np.ndarray = field(repr=False)
    photons: np.ndarray = field(repr=False)
    times: np.ndarray = field(repr=False)
    charged_particles: np.ndarray = field(repr=False)
    depths: np.ndarray = field(repr=False)
    total_photons: np.ndarray = field(repr=False)
    cos_theta: np.ndarray = field(repr=False)
```

If one is accustomed to dealing with CORSIKA IACT photon bunches, the *ShowerSignal* object has a *get bunches* method which returns a rank 2 numpy array where each entry are the parameters in a usual non-compact CORSIKA IACT photon bunch [5]. It takes the index of the desired telescope as an argument. The x and y coordinates of each bunch's arrival in the shadow of the counter are generated randomly.

```
def get_bunches(self, tel_id: int) -> np.ndarray:
```

Many existing analyses read directly from CORSIKA IACT eventio files [7]. CHASM includes a function *write ei file* which writes the output of the simulation to that same format byte for byte. This will allow existing reconstruction programs for any detector which rely on files of this format to use CHASM instead of CORSIKA IACT. This function takes a *ShowerSignal* object and a target filename as arguments.

```
def write_ei_file(sig: ShowerSignal, filename: str) -> None:
```

References

[1] I. Buckland, et al., Astropart. Phys. 150 (2023) 102832

[2] https://github.com/ikepc1/CHASM_NuSpacesim

[3] D. Bergman, PoS(ICRC2013)0983

[4] S. Lafebre, et al., Astropart. Phys. 31 (2009) 243

[5] D. Heck et al., Report FZKA 6019 (1998)

[6] D. Bergman et al., UHECR (2018) 05001

[7] K. Bernlöhr, eventio – a machine-independent hierarchical data format and its programming interface (2014)

[8] J. Krizmanic, et al., PoS(ICRC2019)936