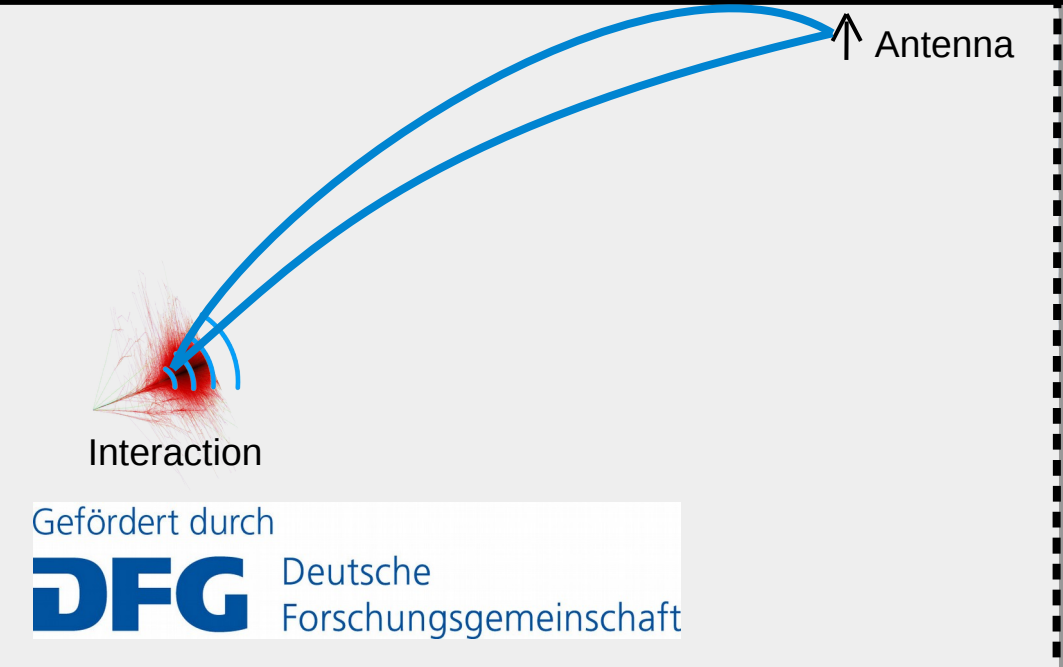
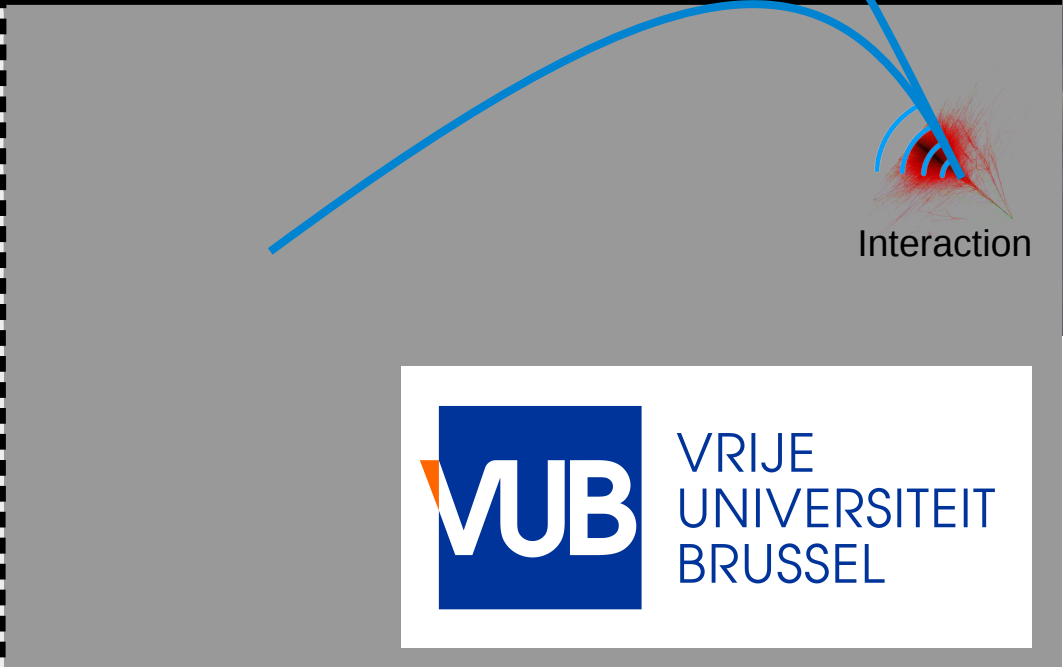
RadioPropa: A Modular Raytracer for In-Matter Radio Propagation

Tobias Winchen

ARENA 2018





Gefördert durch
DFG Deutsche Forschungsgemeinschaft

VUB VRIJE UNIVERSITEIT BRUSSEL

Requirements

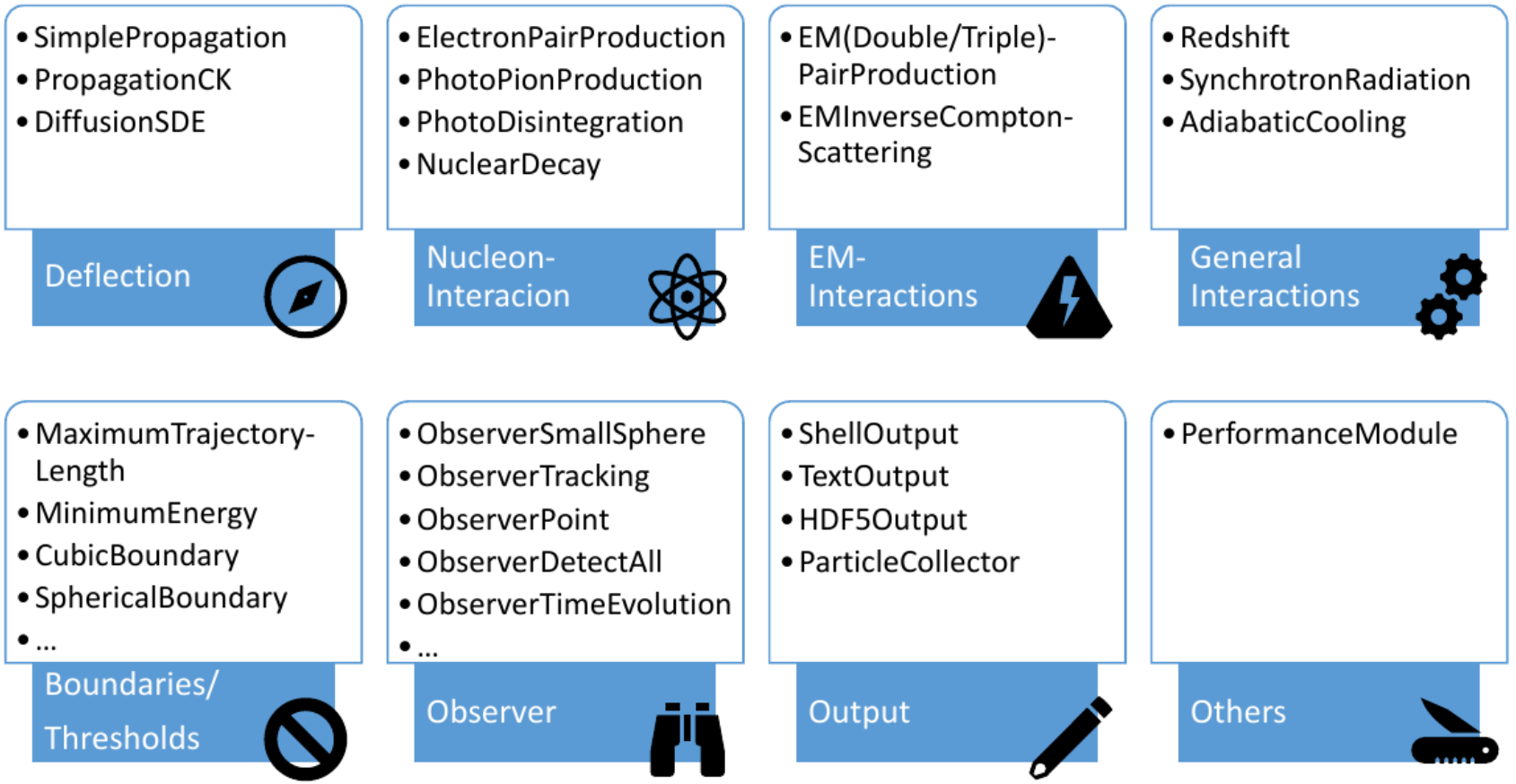
- Versatile geometries
- Arbitrary models for refractive index
- Fast
- No special hardware
- Easily extendable / adaptable

Simulation Framework CRPropa

Rafael Alves Batista[a,b], Julia Becker Tjus[c], Andrej Dundovic[a], Martin Erdmann[d], Christopher Heiter[d], Karl-Heinz Kampert[e], Daniel Kuempel[d], Lukas Merten[c], Gero Müller[d], Günter Sigl[a], Arjen van Vliet[a,f], David Walz[d], Tobias Winchen[d,e,g], Marcus Wirtz[d]

RWTH Aachen University[d], Ruhr Universität Bochum[c], Vrije Universiteit Brussels[g], University Hamburg[a], Radboud University Nijmegen[f], University of Sao Paolo[b], Bergische Universität Wuppertal[e]

Toolbox for Simulations of UHECR Propagation

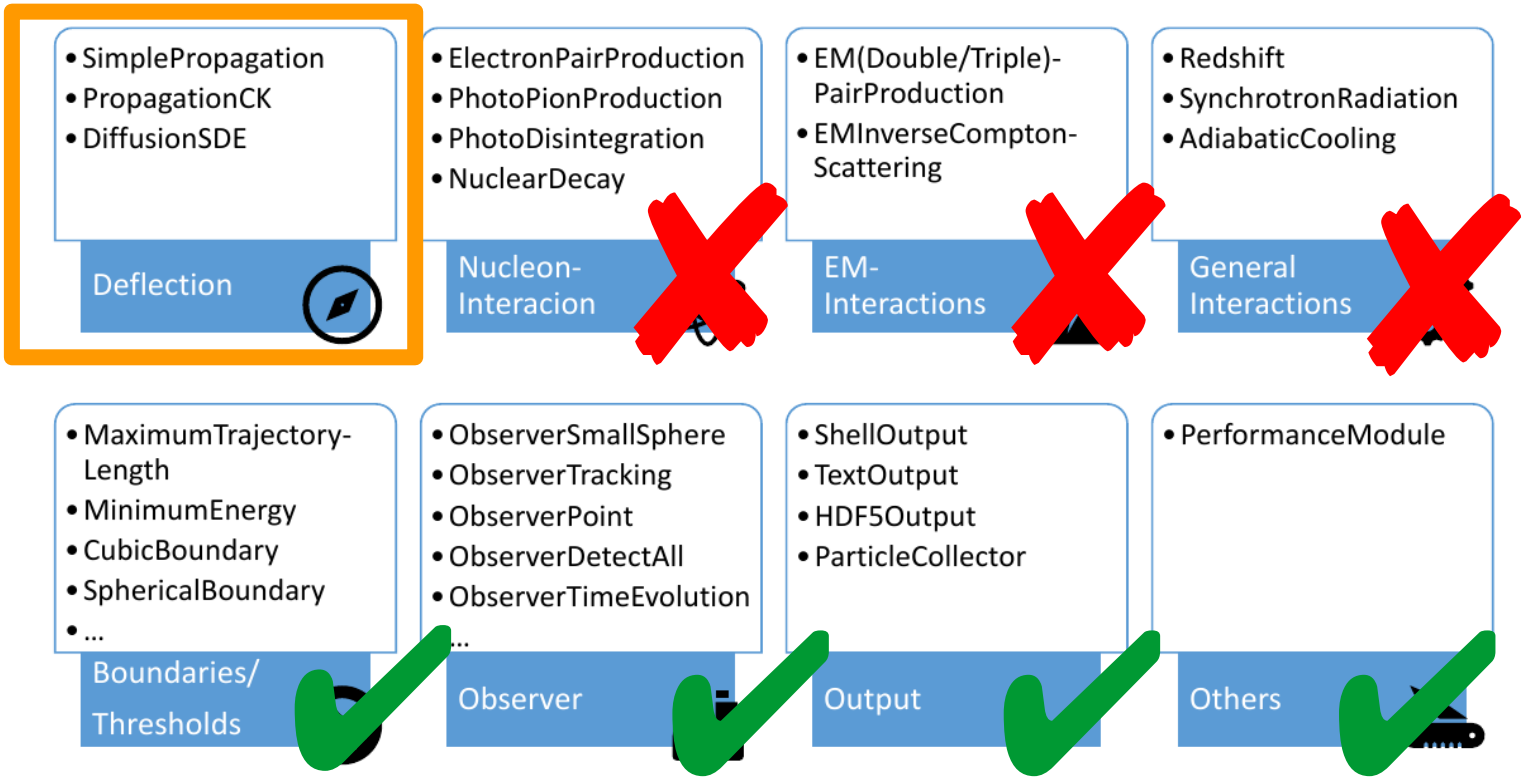


Sketch by L. Merten

Download:
http://crpropa.desy.de

Online demo in your Browser:
http://vispa.physik.rwth-aachen.de

From CRPropa to RadioPropa

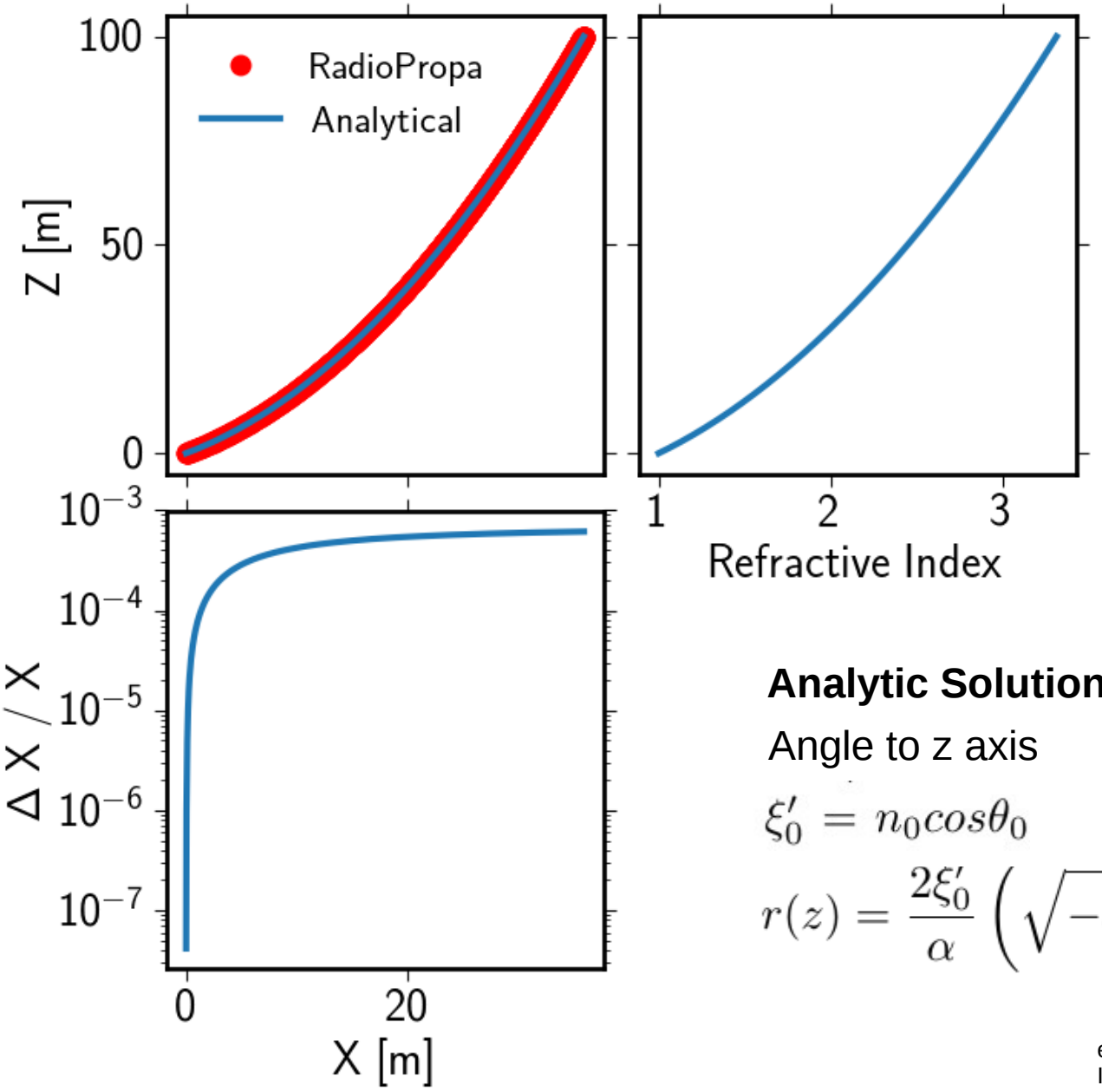


Missing pieces:

- Continuous change of refraction
  - → Propagator module that moves candidate in space
  - → Scalar field to store distribution of refractive index
- Boundaries
  - → Module implementing Fresnel equations
- Adapt data structure to hold amplitude instead of energy, massnumber, redshift, ...

Test: Ray in N^2 linear Medium

Typical test case for ray tracer



$$n^2(z) = n_0^2 + \alpha z$$

Analytic Solution*:

Angle to z axis

$$\xi_0' = n_0 cos\theta_0$$

$$r(z) = \frac{2\xi_0'}{\alpha}\left(\sqrt{-\xi_0'^2 + n_0^2 + \alpha z} - \sqrt{-\xi_0'^2 + n_0^2}\right)$$

e.g. *Q. Mo et al.,
IEEE Transactions on Visualization & Computer Graphics
vol. 22, no. 11, pp. 2493-2506, 2016.
doi:10.1109/TVCG.2015.2509996

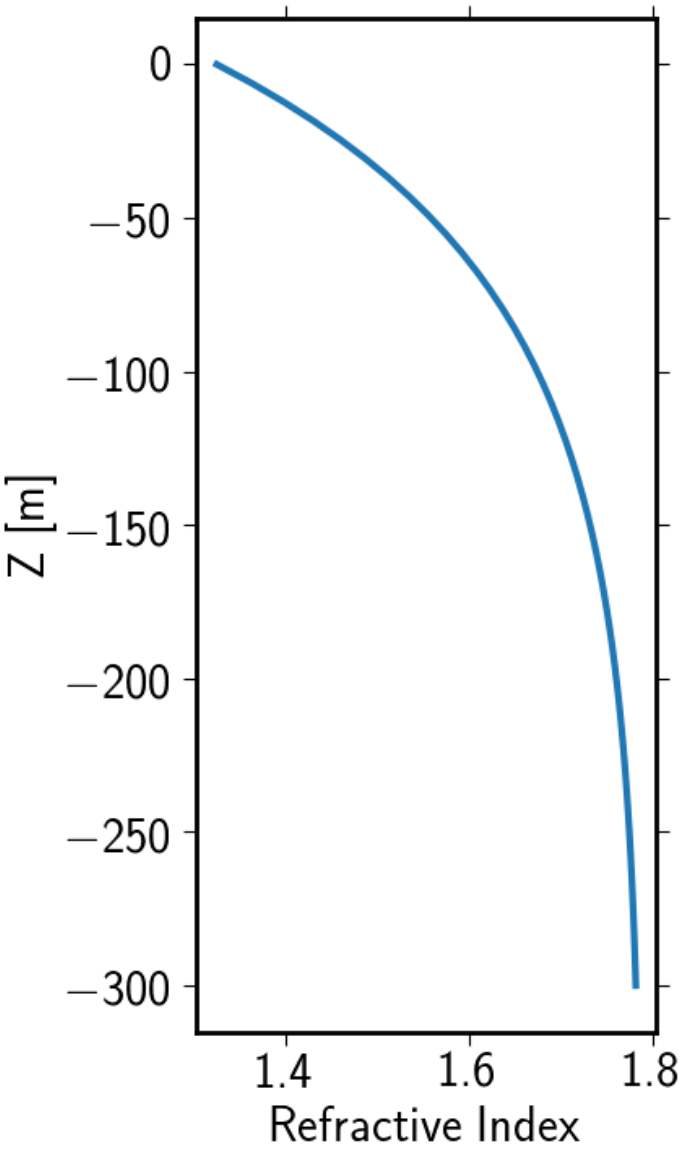
Rays in Ice: Ice Model

Ice Model by Gorham (found on IceCube webpage)

$$n(x, y, z) = a + b(1.0 - e^{-c \cdot z})$$

Implementation

```
56
57 class GorhamIceModel: public ScalarField
58 {
59
60   /*
61     Gorham Ice Model from https://icecube.wisc.edu/~mnewcomb/radio/
62     n = 1.325 + 0.463 * (1.0 - math.exp(-0.0140*depth) )
63    */
64   private:
65     double a,b,c;
66   public:
67     GorhamIceModel() : a(1.325), b(0.463), c(-0.0140)
68     {
69
70     }
71     virtual ~GorhamIceModel()
72     {
73     }
74
75   virtual double getValue(const Vector3d &position) const {
76         return a + b * (1.0 - exp(-1.*c*position.z));
77   };
78
79   virtual Vector3d getGradient(const Vector3d &position) const {
80         Vector3d v(0,0,0);
81         v.z = 1.0 * b * c * exp(-1.*c*position.z);
82         return v;
83   }
84 };
85
```

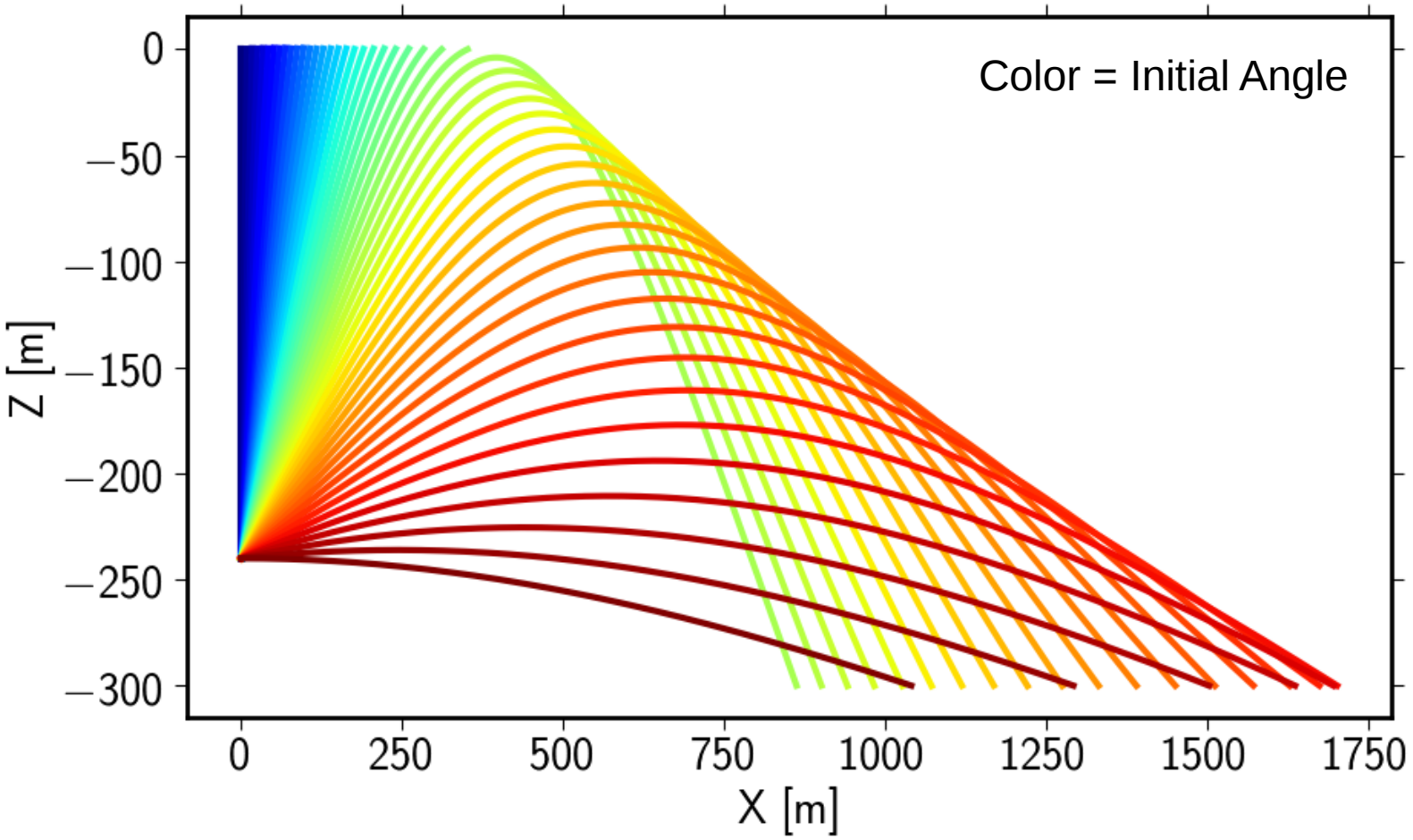


Python implementation also possible, but much! slower

Rays in Ice: Simulation Setup

```
1  import radiopropa
2  import numpy as np
3
4  iceModel = radiopropa.GorhamIceModel()
5
6  if __name__ == "__main__":
7      # simulation setup
8      sim = radiopropa.ModuleList()
9      sim.add(radiopropa.PropagationCK(iceModel, 1E-8, .001, 1.))
10
11     # Observer to stop simulation at z=0m and z=300m
12     obs = radiopropa.Observer()
13     obsz =
14     radiopropa.ObserverSurface(radiopropa.Plane(radiopropa.Vector3d(0,0,0),
15         radiopropa.Vector3d(0,0,1)))
16     obs.add(obsz)
17     obs.setDeactivateOnDetection(True)
18     sim.add(obs)
19
20     obs2 = radiopropa.Observer()
21     obsz2 =
22     radiopropa.ObserverSurface(radiopropa.Plane(radiopropa.Vector3d(0,0,-300 * radiopropa.meter),
23         radiopropa.Vector3d(0,0,1)))
24     obs.add(obsz2)
25     obs2.setDeactivateOnDetection(True)
26     sim.add(obs2)
27
28     # Output
29     output = radiopropa.HDF5Output('output_traj.h5', radiopropa.Output.Trajectory3D)
30     output.setLengthScale(radiopropa.meter)
31     output.enable(radiopropa.Output.SerialNumberColumn)
32     sim.add(output)
33
34     # Source
35     source = radiopropa.Source()
36     source.add(radiopropa.SourcePosition(radiopropa.Vector3d(0, 0, -240. * radiopropa.meter)))
37     source.add(radiopropa.SourceAmplitude(1))
38     source.add(radiopropa.SourceFrequency(1E6))
39
40     #Start rays from 0 - 90 deg
41     for phi in np.linspace(0, 90):
42         z = np.cos(phi * radiopropa.deg)
43         x = np.sin(phi * radiopropa.deg)
44         print('Ray Direction {} deg = ({}, 0, {})'.format(phi, x, z))
45         source.add(radiopropa.SourceDirection(radiopropa.Vector3d(x, 0 , z)))
46         sim.setShowProgress(True)
47         sim.run(source, 1)
48
```

Rays in Ice: Performance

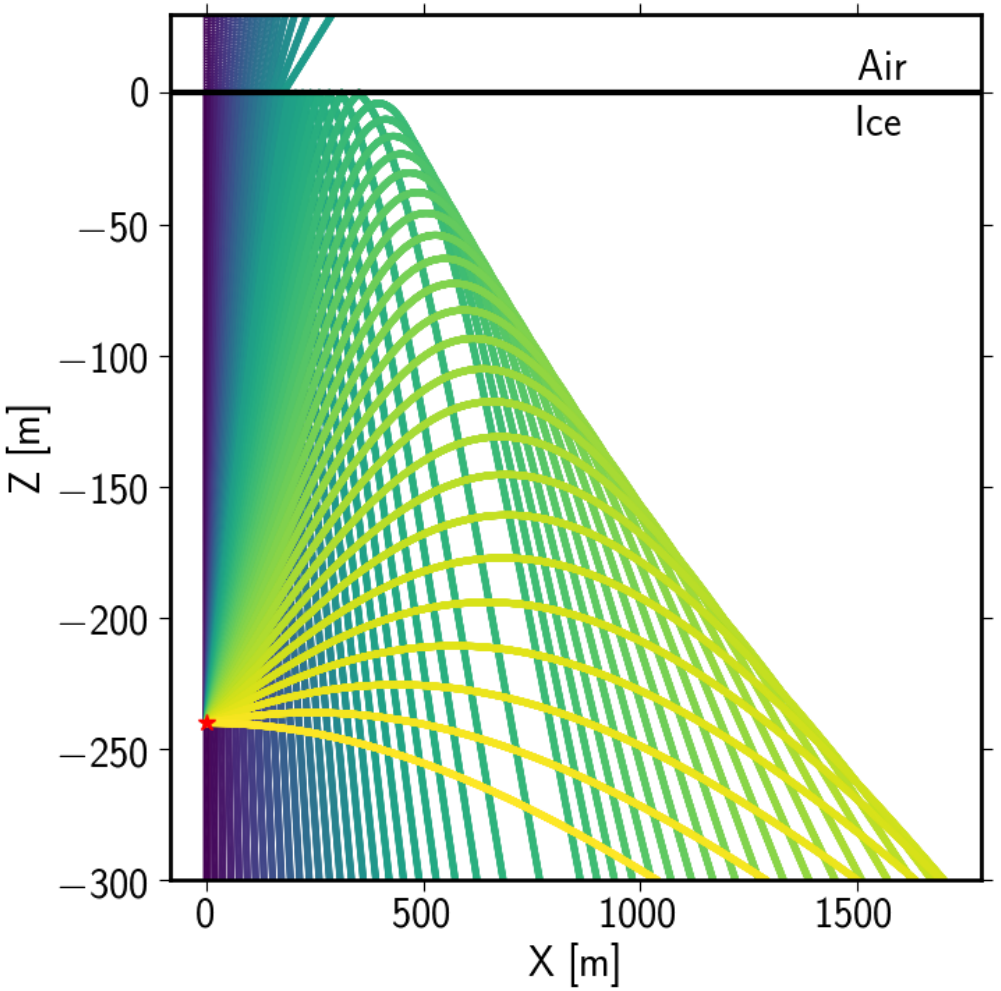


Performance:

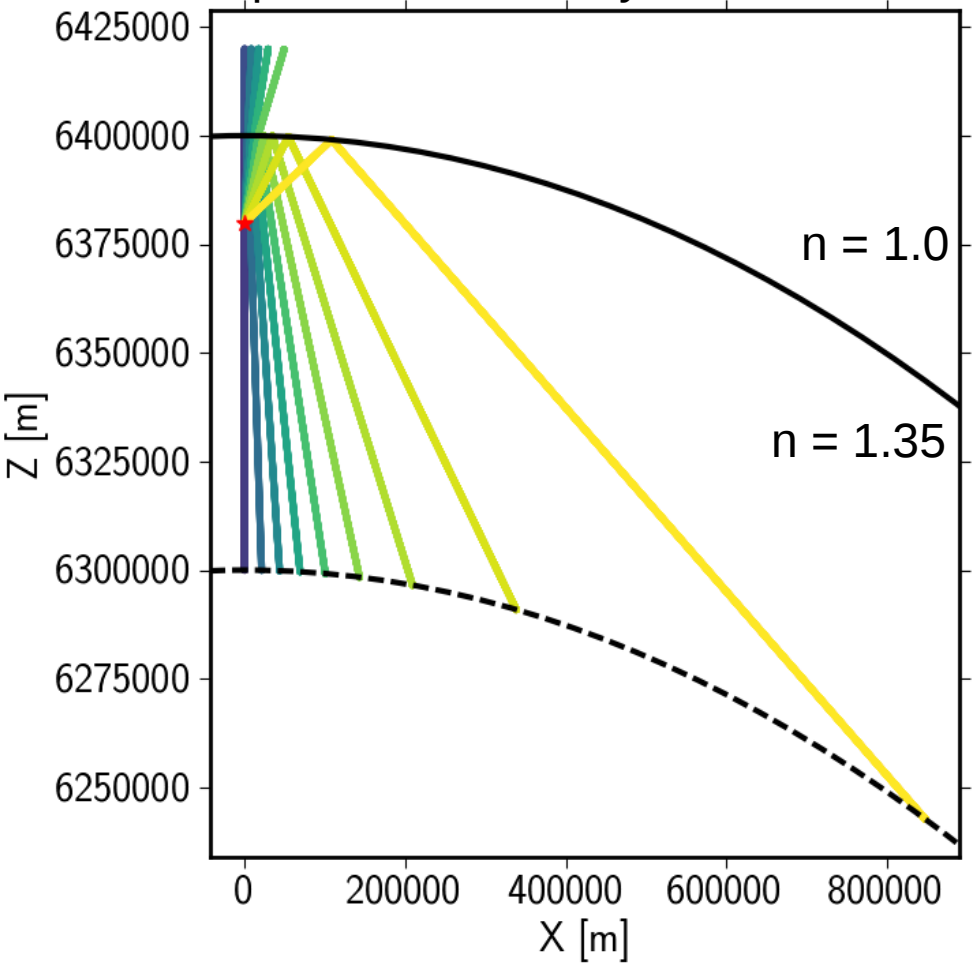
- Laptop Lenovo T470s,
  Intel Core i7-7500U CPU @ 2.70GHz
- No file output
- **100,000 rays / min / core**

Features: Reflection and Refraction

Ice model as before

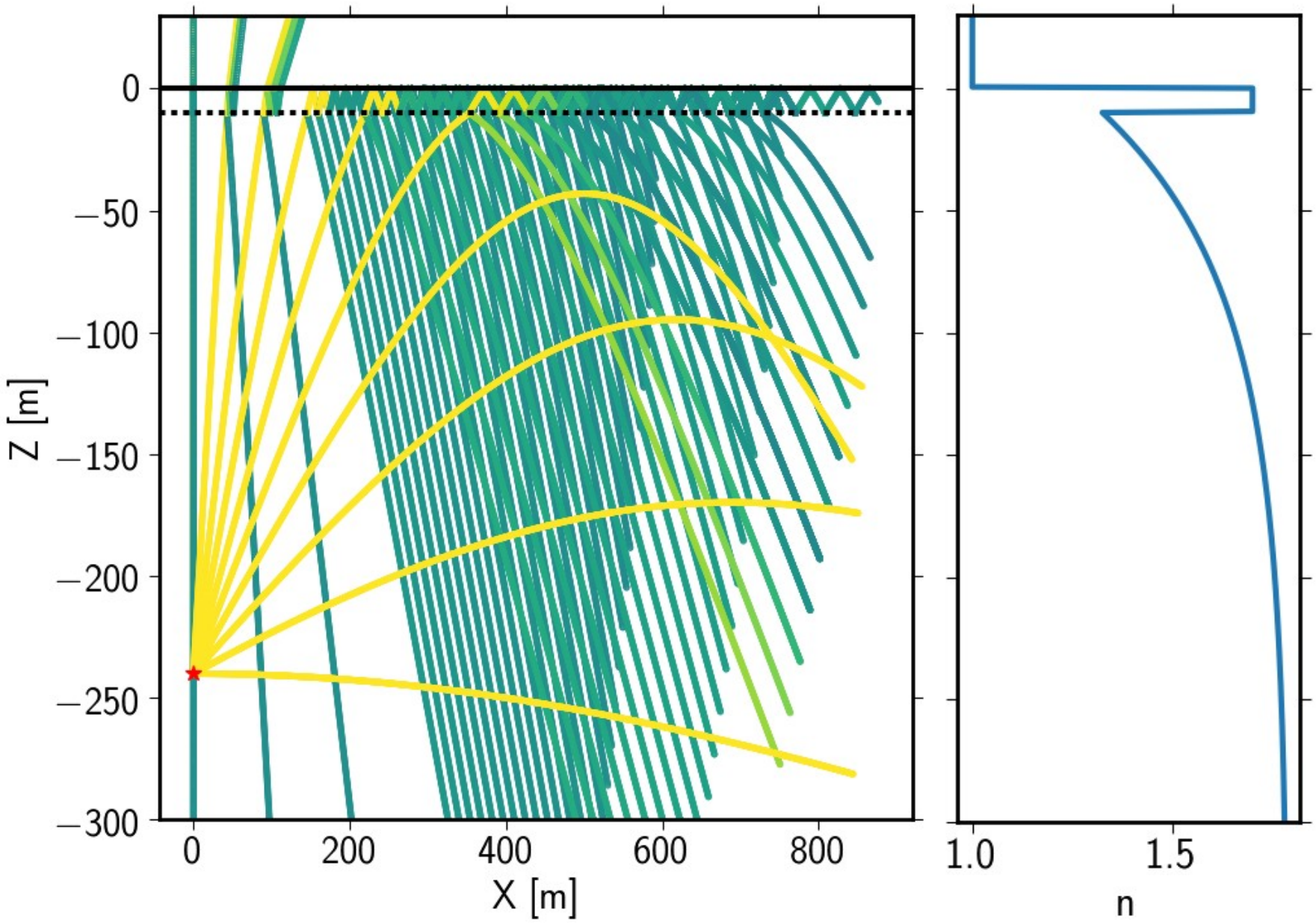


Spherical Geometry



Color = Initial Angle

Features: Partial Reflection + Amplitudes



Color = Amplitude

Conclusion

- Implemented Ray Tracing in ~~CRPropa~~ RadioPropa
  - → Arbitrary continuous index of refraction and source / observer positions
  - → Partial reflection / refraction at boundaries
  - → Maybe possible to add parametrization for additional effects beyond classical ray-tracing (Surface waves, ...)

- Code + Examples:

  https://github.com/TobiasWinchen/RadioPropa

Gefördert durch
DFG Deutsche Forschungsgemeinschaft

tobias.winchen@vub.be