3D Raytrace Algorithm

September 28, 2020

1 Introduction

The purpose of this algorithm is to be able to accurately replicate the trajectory of photons through an imploding ICF (Inertial Confinement Fusion) capsule, as simulate using the ALE Radiation-Hydrodynamics code, *Odin*. The program takes the density profile, as output by *Odin*, and calculates a refractive index, η profile, using the following equation:

$$\eta = \sqrt{1 - \frac{n_e}{n_{crit}}} \tag{1}$$

where n_e is the electron number density, and n_{crit} is the critical density, given by:

$$n_{crit} = \frac{\epsilon_0 m_e \omega^2}{e^2} \tag{2}$$

where ϵ_0 is the vacuum permittivity constant, m is the electron rest mass, ω is the angular frequency of the probing laser, and e is the electron charge. Note that, since $\omega = 2\pi\nu$, we can re-write the above equation as a function of the laser frequency ν, or wavelength λ:

$$n_{crit} = \frac{4\pi^2 \epsilon_0 m_e \nu^2}{e^2} = \frac{4\pi^2 \epsilon_0 m_e c^2}{(e\lambda)^2} \tag{3}$$

Upon reaching the critical density, the laser is reflected away from the center of the plasma. From this equation, we can see that, lasers of higher frequency/smaller wavelength can reach higher densities and penetrate further into the plasma.

The critical density denotes, the maximum number density that a probing laser of angular frequency ω can penetrate at an angle of incidence of $0°$. In order to account for the angle of incidence, θ_0, of the laser, one must use the following equation:

$$n_{turn} = n_{crit} \cos^2 \theta_0 \tag{4}$$

As the laser travels further into the plasma, is is travelling through denser material. As a result of this, it is refracted more, and will have a parabolic trajectory throughout the plasma. The angle of trajectory of the ray is determined by Snell's Law:

$$\frac{\sin \theta_2}{\sin \theta_1} = \frac{\eta_1}{\eta_2} \tag{5}$$

2 The Algorithm

The program begins by reading in the data from the *Odin* output. It reads in the density and grid points of the simulation and then calculates the refractive index as described in equations (1) and (2). Since, the critical density is a function of the laser frequency, λ. This remains as a changeable parameter for the user in order to account for different diagnostics. For the initial diagnostic, X-ray radiography, we will use a wavelength in the order of nanometers as is characteristic of X-rays.

Having loaded the grid and refractive index profile, we can now set the initial conditions for the ray. For this, we have to decided on an initial radius, angle of incidence and angle of location, in both the θ and ϕ directions. Now we position the ray on our grid by performing a scan through the coordinates. First, we scan in the θ-direction, as shown in the image below:

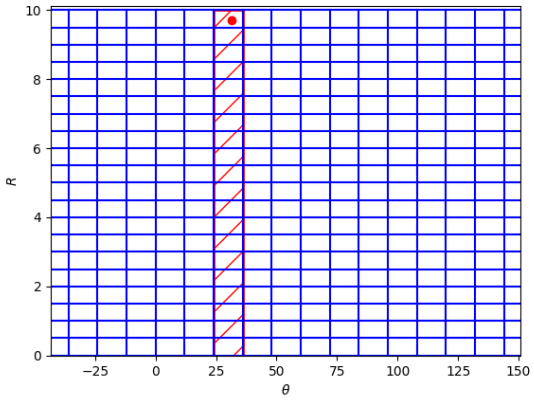

Figure 1: A graphical depiction of the locating algorithm in the Raytracing program. Here we see the algorithm identifying the θ value of the initial position of the ray to be between 24° and 36°.

Afterwards, the program scans in the R direction as shown below:

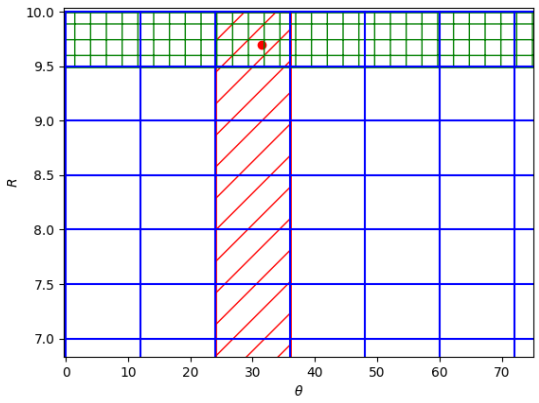

Figure 2: A graphical depiction of the locating algorithm in the Raytracing program. Here we see the algorithm identifying the R value of the initial position of the ray to be between 9.75 and 10 (arbitrary units).

In 3D, the algorithm has to repeat this in the ϕ direction as well:

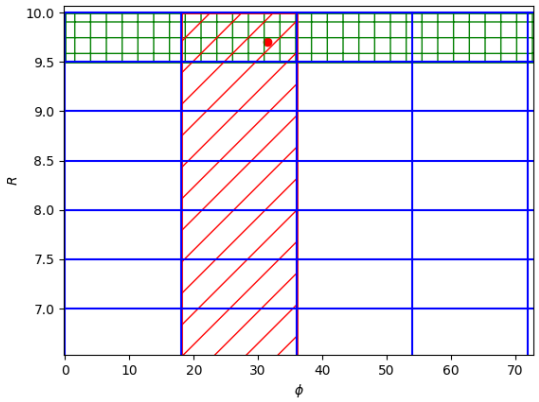

Figure 3: A graphical depiction of the locating algorithm in the Raytracing program. Here we see the algorithm identifying the ϕ value of the initial position of the ray to be between 18° and 36°.

Having found the location of the ray within the grid, the algorithm can determine the refractive index value for this cell. Afterwards, it calculates the nearest cell vertice it will cross, in either R, ϕ, and θ direction. In order to determine which cell edge is nearer, the algorithm needs to calculate all three of the distances. This principle is shown in the Figure below:

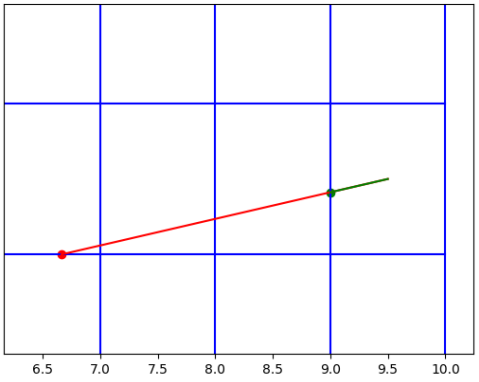

Figure 4: A schematic depiction of the algorithm calculating the distance/timestep until the ray crosses the next x or y vertice. The algorithm will chose the smaller of the two values.

I will now discuss the distance/time calculations needed for the ray crossing each vertice.

2.1 2D θ crossing

The Odin grid has θ_{grid} values which extend over a circle/sphere. Therefore it has values in the range $[-\pi,\pi]$. These values can be considered as gradients:

$$m_\theta = \tan\theta_{grid} \tag{6}$$

Note that θ_{grid} describes the angle of the photon with respect to the capsule, and θ_{traj} is the angle of the photon's trajectory. We start our derivation by stating that:

$$x = x_0 \pm cdt\cos\theta_{traj} \tag{7}$$

$$y = y_0 \pm cdt\sin\theta_{traj} \tag{8}$$

Now we consider the gradient of the grid, m_θ:

$$m_\theta = \frac{y}{x} = \frac{y_0 \pm cdt \sin\theta_{traj}}{x_0 \pm cdt \cos\theta_{traj}} \tag{9}$$

Solving for dt, we get something of the form:

$$dt = \frac{y_0 \pm mx_0}{c(m\cos\theta \pm \sin\theta)} \tag{10}$$

2.2 2D R crossing

Using the definitions of x and y we used in equation 8 and 9, we can calculate the timestep for the ray to cross the R lines. We start by using the value of the grid R_{grid}:

$$R_{grid} = \sqrt{x^2 + y^2} = \sqrt{(x_0 \pm cdt\cos\theta_{traj})^2 + (y_0 \pm cdt\sin\theta_{traj})^2} \tag{11}$$

By taking the square of this value, we get a quadratic equation:

$$c^2dt^2 \pm 2cdt(x_0\cos\theta_{traj} \pm y_0\sin\theta_{traj}) + (R_0^2 - R_{grid}^2) = 0 \tag{12}$$

Using the quadratic formula, we get the solution:

$$dt = \frac{1}{c}\Big(\pm x_0\cos\theta \pm y_0\sin\theta\Big) \pm \frac{1}{c}\sqrt{R_{grid}^2 + (x_0\cos\theta + y_0\sin\theta)^2 - R_0^2} \tag{13}$$

2.3 2D dt value

Note that having calculated the dt values for the ray to cross the R_{grid} and θ_{grid}, then the algorithm chooses the smallest positive value.

The above equations change once we consider the ray in a 3D capsule.

2.4 Moving the algorithm to 3D

For a 3D capsule, we add a ϕ angle to our grid to extend it in the $x-y-z$ coordinates. Note that ϕ has values between [0,2π]. The 3D grid looks like this:

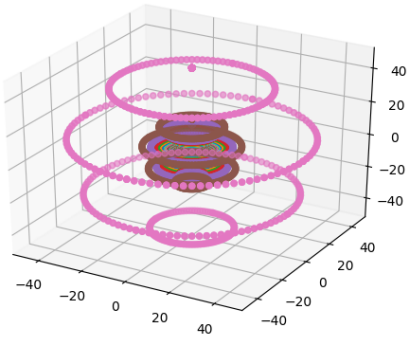

Figure 5: The Odin capsule extended to 3D.

For the purposes of 3D, we consider the following equations of motion for the ray's trajectory:

$$x = x_0 \pm cdt\cos\theta_{traj}\sin\phi_{traj} \tag{14}$$

$$y = y_0 \pm cdt \sin\theta_{traj} \sin\phi_{traj} \tag{15}$$
$$z = z_0 \pm cdt \cos\phi_{traj} \tag{16}$$

Note that once again, we only need consider the R crossing and θ crossing of the rays in the grid - since the ϕ component of the grid is merely an extrapolation of the original $R-\theta$ or $x-y$ grid. Having said that, the R component of the grid, considers the $x-y-z$ components and will form several shells around the center of the capsule. The algorithm can track when the rays intersect these R-shells. We also need to track the θ intersections of the rays, and this is done as a projection onto the 2D grid. Using the new equations of motion for the rays $\vec{v}(x, y, z, \theta, \phi)$, we can see when the ray crosses these θ lines. This is described in more detail below:

2.4.1 3D θ crossing

Similarly to the 2D version, we consider the gradient of the Odin capsule defined by the θ vectors:

$$m_\theta = \tan\theta_{grid} \tag{17}$$

And to derive the value for the timestep dt, we re-arrange the following equation:

$$m_\theta = \frac{y}{x} = \frac{y_0 \pm cdt \sin\theta_{traj} \sin\phi_{traj}}{x_0 \pm cdt \cos\theta_{traj} \sin\phi_{traj}} \tag{18}$$

This gives the solution:

$$dt = \frac{y_0 \pm mx_0}{c(m\cos\theta_{traj} \sin\phi_{traj} \pm \sin\theta_{traj} \sin\phi_{traj})} \tag{19}$$

2.5 3D R crossing

Start off with the following definition:

$$R_{grid}^2 = (x_0 \pm cdt \cos\theta_{traj} \sin\phi_{traj})^2 + (y_0 \pm cdt \sin\theta_{traj} \sin\phi_{traj})^2 + (z_0 \pm cdt \cos\phi_{traj})^2 \tag{20}$$

and we define R_0 as,

$$R_0 = \sqrt{x_0^2 + y_0^2 + z_0^2}. \tag{21}$$

and r_0 as:

$$r_0 = \pm x_0 \cos\theta_{traj} \sin\phi_{traj} \pm y_0 \sin\theta_{traj} \sin\phi_{traj} \pm z_0 \cos\phi_{traj} \tag{22}$$

Once again, we can re-arrange this equation as a quadratic:

$$c^2dt^2 \pm 2cdtr_0 + (R_0^2 - R_{grid}^2) = 0, \tag{23}$$

and solve for dt:

$$dt = \frac{1}{c}(r_0) \pm \frac{1}{c}\sqrt{R_{grid}^2 - R_0^2 + r_0^2} \tag{24}$$

2.6 Algorithm Summary

These are the steps the algorithm takes:

1. User chooses wavelength for incoming rays
2. From the Odin output files, a refractive index profile is generated for the capsule
3. User chooses initial positions (x, y, z) and angles of incidence $(\theta_{traj}, \phi_{traj})$ for a specified number of rays
4. Initial position of ray is located within the grid, by using a scanning routine
5. With the initial "cooridnates" of the ray, then calculate time for ray to cross cell boundaries - choose shortest option
   - Update coordinates
   - Apply coordinates to calculate refractive index of cell
   - Apply Snell's law to calculate new angle of trajectory of Ray
6. Repeat above steps until Ray has left the capsule

3 Results

Having gotten the algorithm to a place where I am fairly happy with it, here are the results of the algorithm working on an Odin implosion simulation, at various stages of it's progression. Firstly, I present the 2D results, and then the 3D results, and how an X-Ray radio-graph is likely to look.

3.1 2D Results

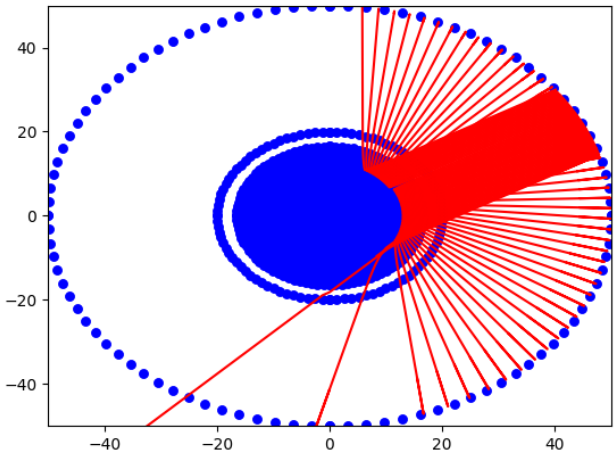

Figure 6: A series of 50 incoming rays travelling through the Odin grid at a range of initial positions and angles of incidence ranging from 30° and 40°. Note that the rays are reflected at the critical surface of teh target.

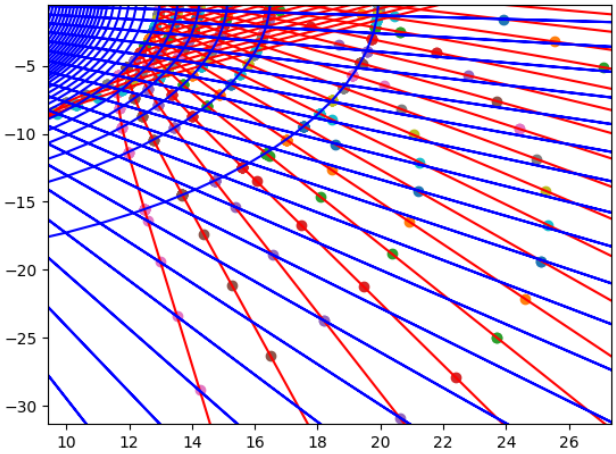

Figure 7: A zoomed section of Figure 6 showing that the algorithm is picking the crossing points of the grid for the rays trajectories.

3.2 3D Results - Initial Test

The next step in the progression of the algorithm was to replicate the 2D results but on a 3D grid. The following Figure shows this result:

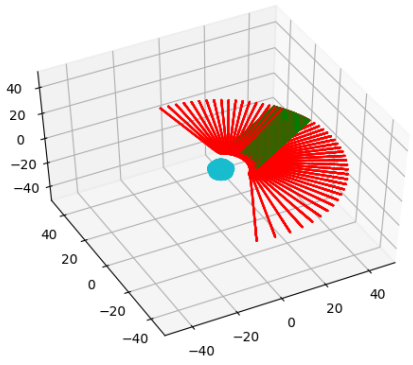

Figure 8: A 3D image of the results shown in Figure 6. This was an initial test case to ensure that the 3D version of the algorithm was working as expected. As you can see, the results are similar. Note that the incoming rays are shown in green.

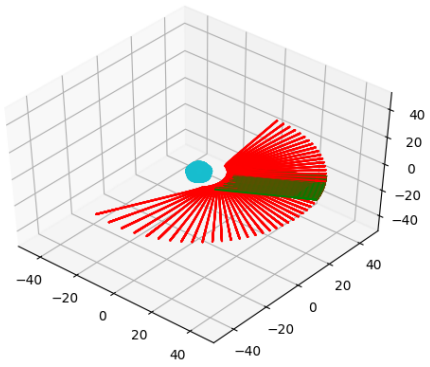

Figure 9: A different angle of Figure 8.

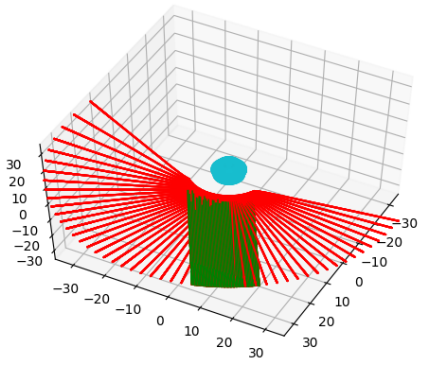

Figure 10: A different angle of Figure 8.

3.3 3D Results - Full

Having successfully tested the 3D algorithm's ability to replicate the 2D results, I moved on to using the algorithm with a range of z for the initial positions of the rays. The results are shown below:

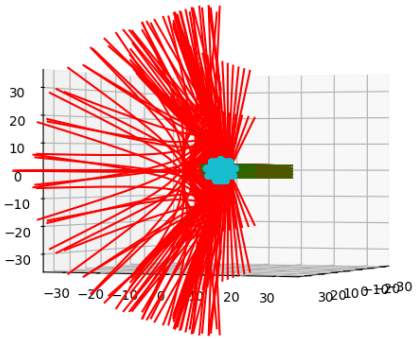

Figure 11: The 3D result of a plane (fixed x) of rays with wavelength $354nm$ incoming into an Odin synthetic capsule.

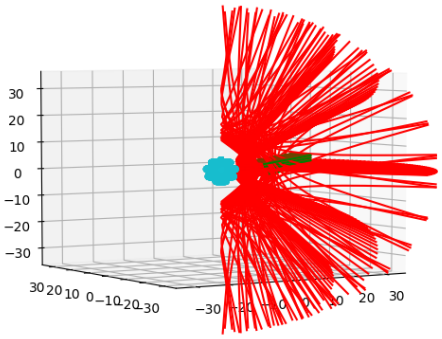

Figure 12: A different angle of Figure 11.

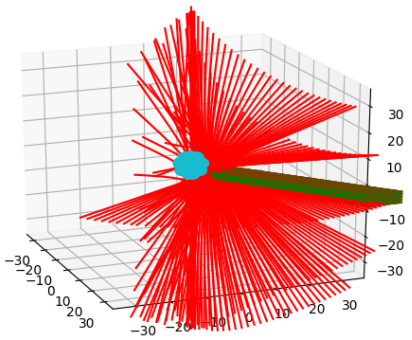

Figure 13: A different angle of Figures 11 and 12.