CompAstroCoursework2

24474

March 2025

0.1 Preface

The command used to compile the file is: ' gcc -o 24474 24474.c -lm -fopenmp -g -O3 ' , it can then be ran with ./24474 The -O3 flag enabled as everything is wrote and tested in a way allowing the aggressive optimisations it provides. However as I do not have access to a windows or a mac based system I cannot guarantee the -O3 flag will provide the same performance benefits, and recommend the -O2 flag instead. I originally had the CSV's get wrote into folders, but this caused errors when the folders were not already made, so for simplicity everything is wrote into the same directory as the compiled c file, remove them all after by running in the same directory as the CSV's, 'rm -f ./*.csv'. (note: this assumes a new directory was made to run the project, if this is ran in a directory with other CSV's it will also delete them) .

1 Q1

I Implemented rejection sampling as a wasteful method. For a not wasteful method I used a direct mapping approach In using the direct mapping approach I tried to invert the function

$$P(\mu)d\mu = \frac{3}{8}(1+\mu^2)d\mu \tag{1}$$

$$P_{cum} = y = \frac{3}{8}\int_{-1}^{\mu} 1+\mu^2 d\mu \tag{2}$$

Where the lower limit is -1 based on the definition of.

$$\mu = cos\theta \tag{3}$$

Integrating (2) and evaluating at the limits we find

$$y = \frac{3}{8}\left[\left(\mu + \frac{\mu^3}{3}\right) - \left(-1 + \frac{(-1)^3}{3}\right)\right] = \frac{3}{8}\left(\mu + \frac{\mu^3}{3} + \frac{4}{3}\right) \tag{4}$$

I rearranged this slightly for implementation, using $1/8 = 0.125$ to avoid floating point division.

$$y = 0.125 * \left(\mu^3 + 3\mu + 4\right) \tag{5}$$

This is not trivially invertible to find μ as a subject. However, as we know the range of μ is between -1 and 1 (due to $\mu = \cos(\theta)$) I have computed the corresponding y value for each of the possible μ's, and stored this in a lookup table. This allows us now to generate a random value for y between 0 and 1 and find the corresponding μ value. I then implemented a binary search that once we have a y value it will search along the table for the corresponding mu value. It will be shown in question q1.c that this is indeed more efficient

1.1 1.b

As seen in figure 1, as the number of samples increases the distribution becomes closer to the desired probability distribution from equation 1, confirming that when the sample size is large we can model a probability distribution using random number generators and a direct mapping approach.

1.2 1.c

The best way too see how the direct mapping lookup table approach is more efficient is simply to measure how long it takes each method to compute a large amount of samples, I chose 500,000. To do this I created a macro that takes in a function and prints the time that function took to complete. The output of which is seen below

```
Total compute time for rejection_sampling: 0.028014 seconds (0.028014 seconds total
    compute time, 1 thread used)
Total compute time for direct_mapping: 0.024613 seconds (0.024613 seconds total
   compute time, 1 thread used)
direct_mapping was faster by 0.003401 seconds
```

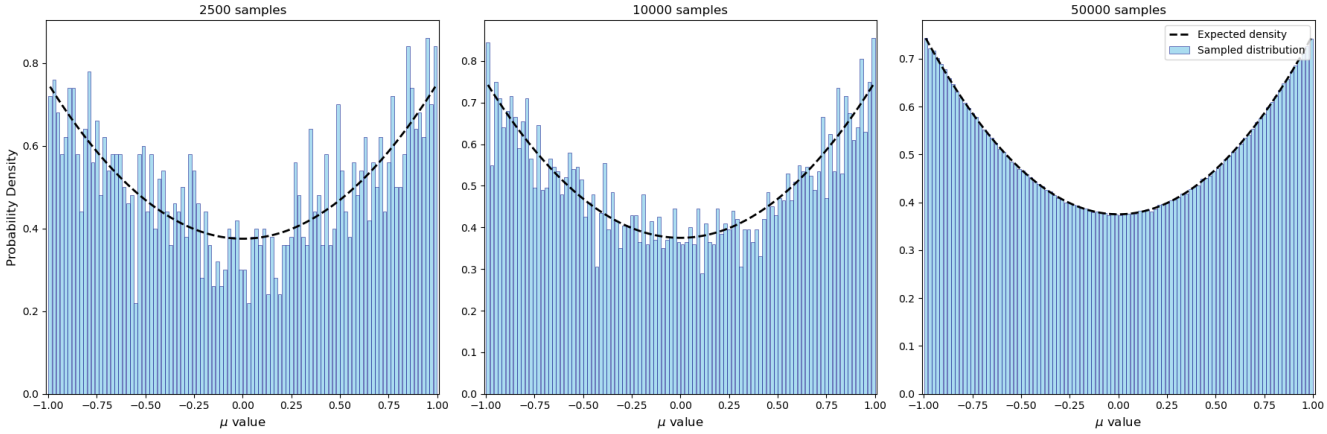

Figure 1: 3 plots showing the density of each μ binned value in the blue bars, and the plot of equation 1 in a dotted black line. As the amount of samples increases the distribution starts to resemble more closely the expected probability density. 100 bins were used to make this graph.

2 Q2

I draw values for t from a random number m between 0 and 1 in a distribution, found by integrating the CDF of,

$$m = P_{cum}(\tau) = \int_0^{\tau} e^{-\tau'} d\tau' = 1 - e^{-\tau}. \tag{6}$$

Therefore

$$\tau = -\ln(1 - m). \tag{7}$$

As the distribution m = 1-m is equally valid, due to the symmetric range between 0 and 1. I will write τ in the form

$$t = -\ln(m). \tag{8}$$

The average mean free path, l is defined by

$$l = \frac{z_{max} - z_{max}}{t}\tau, \tag{9}$$

Where z_{min} and z_{max} are the largest and smallest z values of the medium, and t is the corresponding optical depth for this distance. This will be multiplied to my direction vector to get the distance in each axis the photon moves.

2.1 A random point on a sphere

As a large amount of photons are needed it is important to get all the efficiency I can out of the system. performing a significant amount of sin, cos and especially square root functions can be costly[1]. However it would be beneficial to have the accuracy and also the speed.To do this I will pre-compute a lookup table that associates a random variable u (in the range of 0 to 1) to corresponding x, y , z coordinates that lie on a sphere of radius 1, then I can multiply these values by l to get the distance and direction travelled, and repeat until the photon escapes. First I will start with z, the heigh of the sphere. It can be wrote as

$$z = r\cos(\theta), \tag{10}$$

where r is θ is the inclination from the z axis. A small area on the surface of the sphere can be wrote as,

$$dA = 2\pi r \sin(\theta) d\theta \tag{11}$$

The total surface area of the sphere is $4\pi r^2$. The probability that a randomly chosen point has an angle $\leq \theta$ is given by the cumulative distribution function,

$$P(\theta) = \frac{\int_0^{\theta} 2\pi r \sin\theta' \, d\theta'}{4\pi r^2}. \tag{12}$$

As we are working in a sphere with radius, $r = 1$ this reduces the terms, we can also simplify further and find,

$$P(\theta) = \frac{\int_0^{\theta} \sin\theta' \, d\theta'}{2}. \tag{13}$$

[1]To be fair Sin and Cos have became very efficient functions in C, but some argue that they prefer accuracy over efficiency, but I have not been able to find a reputable source on this. Floating point square root operations and floating point division on the other hand are a bit of a different story, and are just a bit slow.

Evaluating the integral at its limits of 0 and θ we find

$$P(\theta) = \frac{1 - \cos\theta}{2} \tag{14}$$

Since $z = \cos\theta$, the cumulative probability function in terms of z is:

$$P(z) = \frac{1 - z}{2} \tag{15}$$

To generate a uniform sample, we set a uniform random variable $u \in [0, 1]$ equal to $P(z)$:

$$u = \frac{1 - z}{2} \tag{16}$$

Solving for z:

$$z = 2u - 1 \tag{17}$$

Therefore, for a uniform random variable u in the range $[0, 1]$, the transformation $z = 2u - 1$ ensures a correct uniform sampling in the z-direction of a unit sphere. I can find the corresponding y form imagining a 2d circle of radius,

$$r^2 = y_{2d}^2 + z^2 \tag{18}$$

as r =1 for a unit sphere, and rearranging for y.

$$y_{2d} = \sqrt{1.0 - z^2} \tag{19}$$

Then transforming the y into 3d by applying a sinsodial term to simulate rotation.

$$\Phi = f\pi u \tag{20}$$

$$y = y_{2d}\sin(\Phi), \tag{21}$$

and similarly for x

$$x = x_{2d}\cos(\Phi). \tag{22}$$

To show this works from a math point of view we substitute the expressions for x and y into the equation of a sphere of radius 1.

$$r^2 = x^2 + y^2 + z^2 = (\sqrt{1 - z^2}\cos\Phi)^2 + (\sqrt{1 - z^2}\sin\Phi)^2 + z^2. \tag{23}$$

Expanding the terms:

$$(1 - z^2)\cos^2\Phi + (1 - z^2)\sin^2\Phi + z^2. \tag{24}$$

Since $\cos^2\Phi + \sin^2\Phi = 1$, this simplifies to:

$$(1 - z^2) + z^2 = 1. \tag{25}$$

Which is expected as the radius $r^2 = 1$. This also works in a practical sense as the value f is chosen to be high enough to allow for many rotations around a sphere. for this report 1500 was chosen, the resulting plot of the unit sphere can be seen in figure 3. In the code and parts of this report the value of x, y and z might be referred to as $z = d_z$ when relevant ,where $\vec{d}$ is called the direction vector, this is done to make them distinct from the axis used to track the position of the photon. Therefore at each scattering event all we need is a random variable u in the range of 0 and 1 and then a binary search to get a direction vector of the scattering. Then multiply the direction vector by distance moved t and add it to the position of the photon and we have the new position. no more square roots, or sin or cos calculations at each step, until Q3 that is.

2.2 Q2 results

The results of the bins can be seen in Figure 2. The total compute time across all threads is about 5 seconds when running with 8 threads. with a lookup sphere of 50000 total points. as it is a binary search lookup the size of the sphere scales with $O(\log(n))$ hence a large sphere can still result in a fast runtime. The output of the code can be seen below.

```
Q2.
Average count of scattering events for an escaping photon : 61.724376
Total compute time for photon_scattering: 4.816726 seconds (38.533804 seconds total
    compute time, 8 threads used)
```

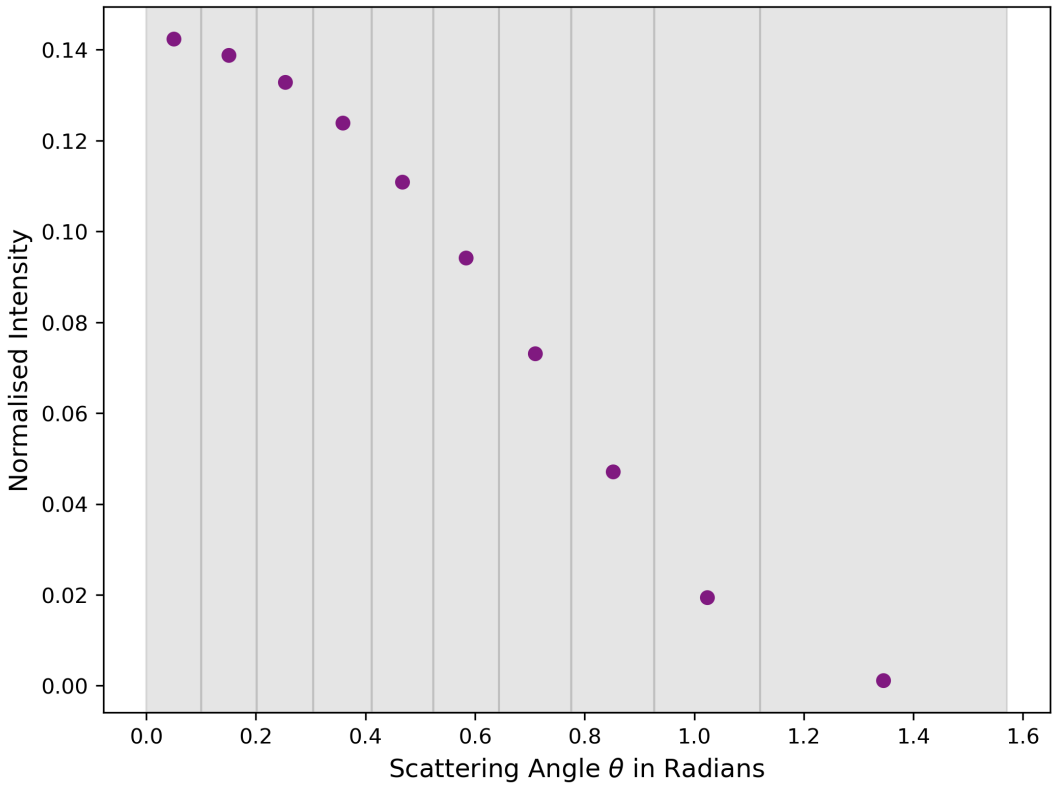

Figure 2: The resulting intensities as a function of θ deviation from the origin measured along the y axis, In purple plotted are the intensity at the midpoint of each bin, and in grey is the size of the bin itself, the size of the bin increase as the distance from the origin increases due to the bins being of equal solid angle.

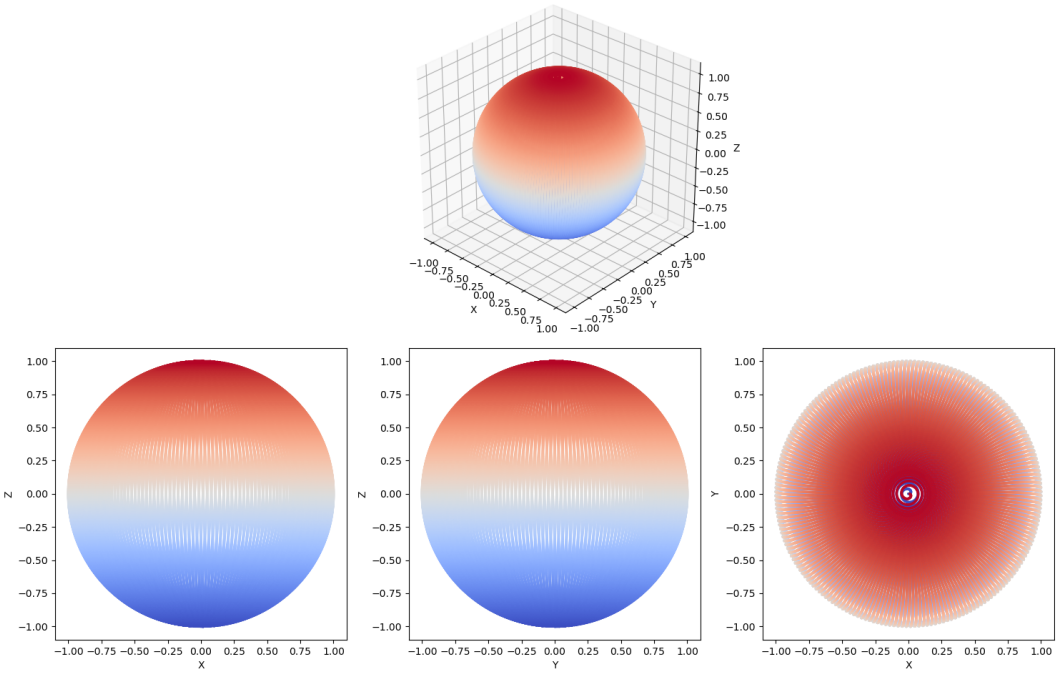

Figure 3: A plot of the displacement direction csv for 50000 total points. Top, a 3D visual of the different possible points that could be taken on a sphere, while it may appear there is less point towards the top of the circle, it is simply that they are spaced closer together as shown by the near solid colour, but as shown by equations 17,21 and 22, it is still leading to a distribution of x,y,z that we desire, to decrease the interval between each spiral I could increase f but to do so I would also need to increase the total points too to ensure each point is visited and not overshot, when overshooting occurs the shape no longer resembles a sphere. This current combination seem to work well. The bottom plots consist of each axis against each other to show that when viewed in 2D it is a circle of radius 1 as expected.

3 Q3

We start with the probability of a scattering event occurring

$$P(\theta)d\Omega = \frac{3}{4}(1 + cos^2\hat{\theta})\frac{d\Omega}{4\pi} \tag{26}$$

Using

$$d\Omega = \sin(\hat{\theta})d\hat{\theta}d\phi \tag{27}$$

and integrating with limits of 0 and $\hat{\theta}$ over $d\hat{\theta}$ and 0 and 2π for $d\phi$, to find $P_{cum}(\hat{\theta})$. (note: no ϕ dependence)

$$P_{cum}(\hat{\theta}) = c = \frac{3}{4}\frac{1}{4\pi}\int_0^{2\pi}\phi d\phi \int_0^{\hat{\theta}}(1 + \cos^2(\hat{\theta}))\sin(\hat{\theta})d\hat{\theta} \tag{28}$$

simplifies to

$$c = \frac{3}{8}\left(\int_0^{\hat{\theta}}\sin(\hat{\theta})d\hat{\theta} + \int_0^{\hat{\theta}}\cos^2(\hat{\theta})\sin(\hat{\theta})d\hat{\theta}\right) \tag{29}$$

The first integral term evaluates to

$$\int_0^{\hat{\theta}}\sin(\hat{\theta})d\hat{\theta} = 1 - \cos(\hat{\theta}) \tag{30}$$

using the substitution $h = cos\hat{\theta}$ on the 2nd integral we find

$$d\hat{\theta} = \frac{-1}{sin(\hat{\theta})}dh \tag{31}$$

$$\int_0^{\hat{\theta}}\cos^2(\hat{\theta})\sin(\hat{\theta})d\hat{\theta} = -\int_1^{\cos\hat{\theta}}h^2dh \tag{32}$$

evaluated to find

$$-\frac{\cos^3(\hat{\theta})}{3} + \frac{1}{3} \tag{33}$$

plugging these evaluated integrals back in we find

$$c = \frac{1}{8}(4 - \cos^3(\hat{\theta}) - 3\cos(\hat{\theta}) \tag{34}$$

where c is a random value between 0 and 1, that gives the corresponding $\hat{\theta}$. However as we can see from the right hand side of figure 4, we can get our value in the z direction by taking

$$z = b_z = L\cos(\hat{\theta}), \tag{35}$$

as we are working in a sphere of radius 1 in all directions, varying θ the cone will form a sphere and as we are looking just from the direction found from a sphere of unit radius, as the displacement comes from l, we can assume $L = r = 1$. Also to allow me to reuse an earlier questions code I will write equation 35 in in this form.

$$b_z = -\cos\hat{\theta} \tag{36}$$

This can be as due to the $\cos^2\hat{\theta}$ term in the probability distribution from equation 26. a value of $\cos\hat{\theta}$ has just as much chance of occurring as -$\cos\hat{\theta}$, thus the same distribution will be mapped. Subbing equation 36 into equation 34, we find

$$c = 0.125 * \left(b_z^3 + 3b_z + 4\right) \tag{37}$$

I will create a lookup table and use this to find the b_z values, and then as each z value is distinct in my displacement direction lookup table, I can reuse it and search the z column to find the corresponding x and y values. Where c is a random number between 0 and 1. However as we can see in figure 4 this will only work if the photon scattering is aligned with the Z axis, but there is a method to resolve this, Have a dummy scattering that takes places for an imaginary photon along the z axis, taking the value of b_z for the z direction. Then finding the cross product, and the angle between the the unit vector of the z axis, and a normalised vector of the current position. Then from this I used Rodrigues rotation formula to rotate the dummy scattering onto the actual frame of the scattering.

3.0.1 Rodrigues rotation formula

Found info about this rotation formula from [1] and [2], Given a vector $\vec{v}$, and a rotation axis $\vec{k}$, and an angle to rotate θ, the rotated vector $\vec{v}_{rot}$ is calculated as:

$$\vec{v}_{rot} = \vec{v}\cos\theta + (\vec{k}\vec{v})\sin\theta + \vec{k}(\vec{k}\cdot\vec{v})(1 - \cos\theta) \tag{38}$$

Here, $\vec{k}\cdot\vec{v}$ denotes the dot product of $\vec{k}$ and $\vec{v}$, and $\vec{k}\times\vec{v}$ denotes their cross product, and θ is the angle between the 2 vectors. This was used over a traditional rotation matrix for the efficiency benefits, and it also looked easier to implement.

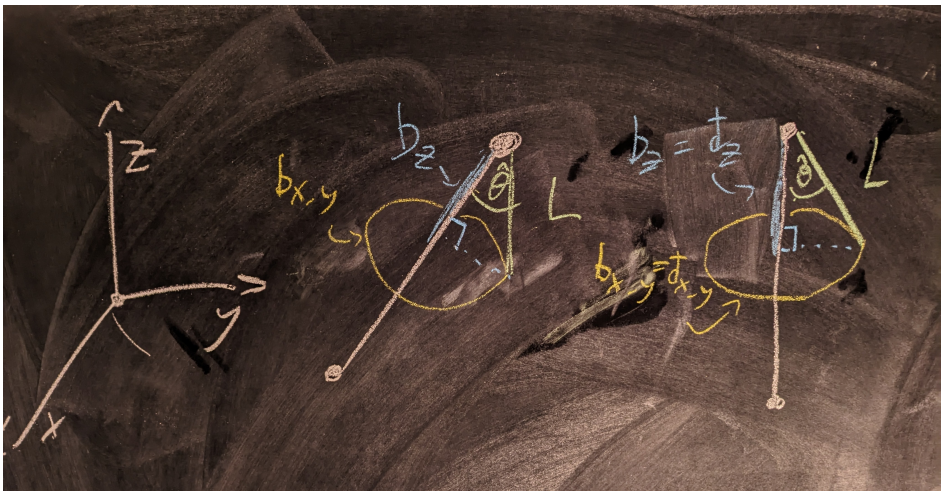

Figure 4: An example of the possible scattering events for rayliegh scattering. A displacement vector from an axis aligned with a photon is defined as $\vec{b}$, and one aligned with our view $\vec{d}$ ($d_z = z$ where z is the displacement on a sphere of radius 1, and similar for $d_x = x$ and $d_y = y$). On the left hand side we have the new direction vectors from the photons perspective when it is not aligned with our axis, the new position of the photon will be somewhere on the yellow circle times by τ .As $b_{x,y,z}$ does not align with the current axis b_z will not be the same as the z displacement vector from our perspective, $\vec{b_z} \neq \vec{d_z}$, so the lookup table cannot be used to get an accurate result. On the right hand side we have the photon travelling in the z direction when the scattering event occurs, as this aligns with our z axis $b_z = d_z$, and then we can use the lookup table to find a corresponding d_x and d_y. For each scattering event there exists an axis that can be found using Rodrigues rotation formula that can be used to rotate a theoretical scattering event on a coordinate system we know, onto a coordinate system in the photons perspective

3.1 Q3 Results

From figure 5 we can see a very narrow distribution of resulting angles from other colours, but for blue light it is much broader suggesting more scattering has occurred. Looking at figure 6, we can see a much greater distribution of x and y values, and in particular it can be seen that most of the light from other colours actually makes it through without scattering. In addition to this I also had printed as part of the code the amount of scattering events that occurred on average for a photon to leave the medium; 58.51 for blue light and 0.062 for other colours, blue light gets scattered more than anything else hence when we look up to the sky we see the scattering of the blue light. Code output can be seen below:

```
Q3.
Blue light
Average count of scattering events for an escaping photon: 58.513847
Total compute time for photon_scattering: 6.485650 seconds (51.885202 seconds total
    compute time, 8 threads used)

Other colours
Average count of scattering events for an escaping photon: 0.062137
Total compute time for photon_scattering: 0.058543 seconds (0.468342 seconds total
    compute time, 8 threads used)
```

4 References

[1] Liang, K.K., 2018. Efficient conversion from rotating matrix to rotation axis and angle by extending Rodrigues' formula. Available from: https://arxiv.org/abs/1810.02999 [Accessed 25 March 2025].
[2] Friedberg, R.M. (2022) 'Rodrigues, Olinde: "Des lois géométriques qui régissent les déplacements d'un systéme solide..." translation and commentary', Available at: https://arxiv.org/abs/2211.07787 (Accessed: 25 March 2025).

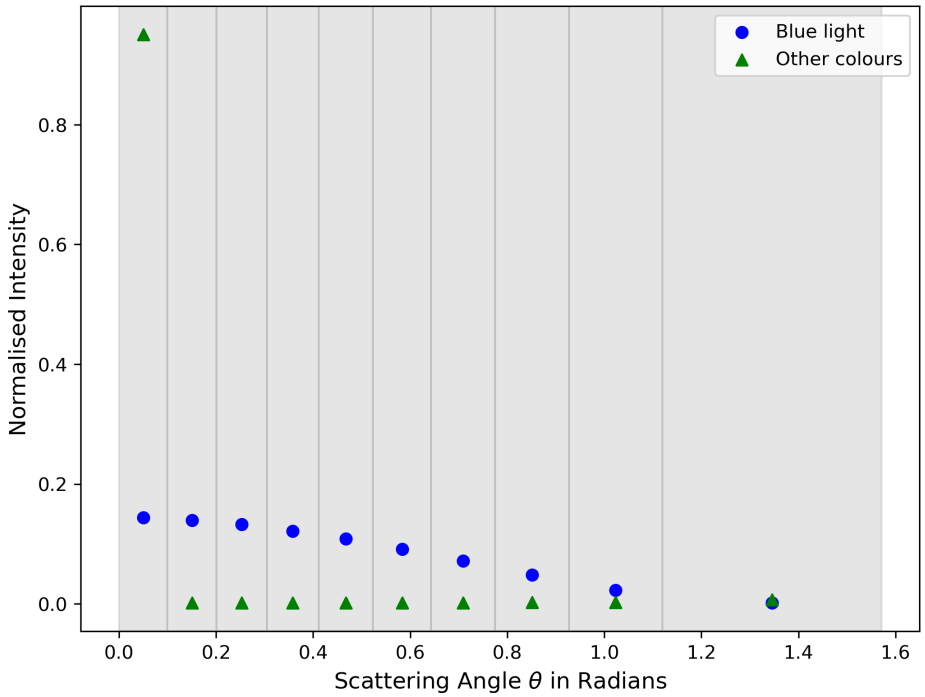

Figure 5: Comparison between Blue light (in blue), and other colours in green, with the size of the bins plotted behind, we can see a sharp drop for green light, suggesting not much scattering has occurred, for blue light we see a more gradual decrease in the distribution suggesting more scattering has taken place

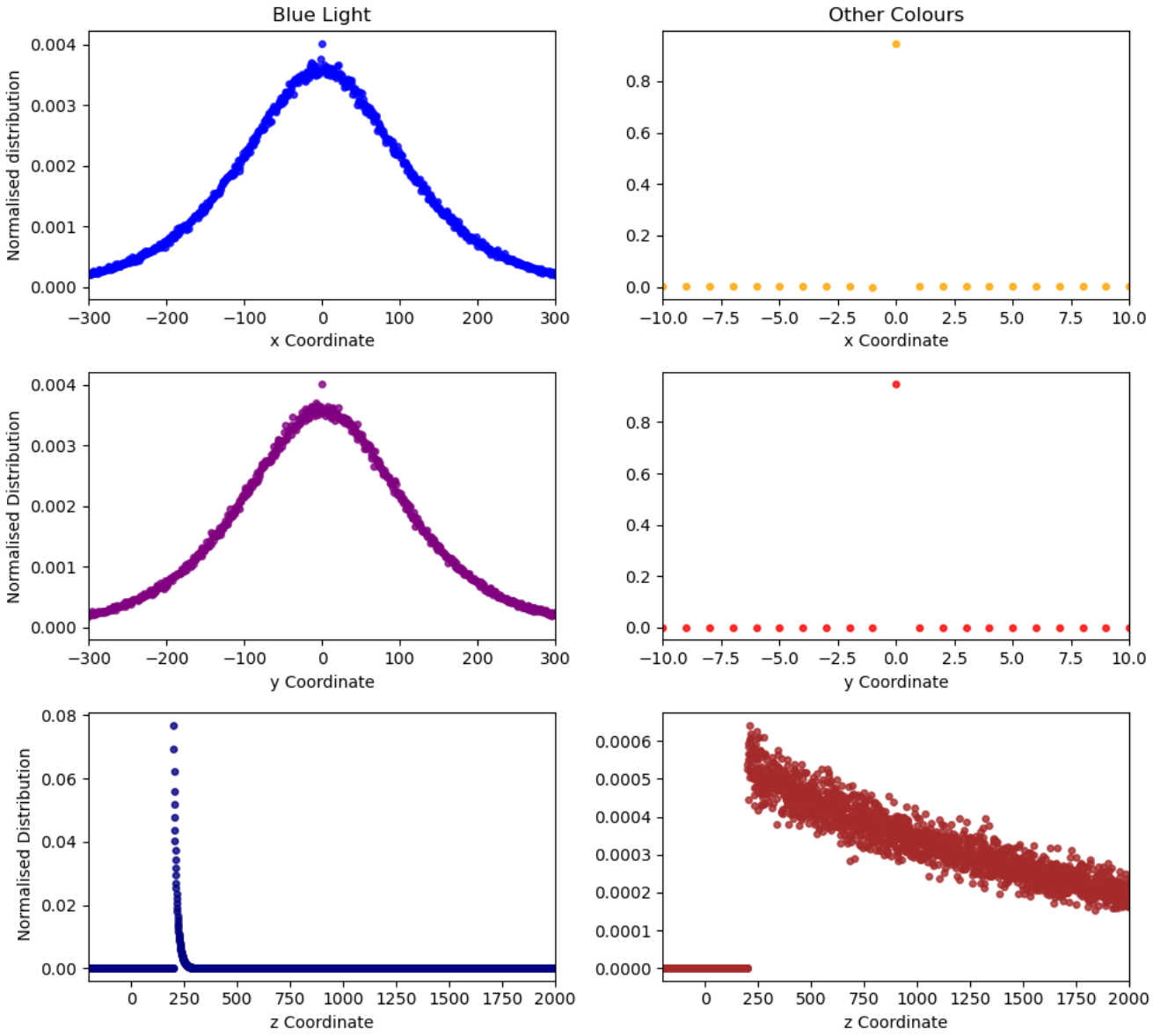

Figure 6: On the left hand size we can see a distribution of the x,y and z coordinates of blue light once it escapes.there is a minor peak at x and y = 0 for photons that did not scatter or interact at all. On the right hand size we see the x, y and z of other colours, from the x and y we see a clear peak, suggesting not much scattering has occured. and we can see from the z values that the photons can travel a very large distance without scattering