Cylindrical Apertures to Measure Neutron Lifetime

Tyler J. Mason
July 7, 2022

1 Introduction

The decay lifetime of an isolated Neutron, τ_n, has been studied extensively in the past, however both scrutinized results do not agree on the precise value. The 2 main studies took place at the Institute Laue-Langevin using the "Bottle" method, and at the National Institute of Standards and Technology Center for Neutron Research using the "Beam" method. The "Bottle" method uses ultra-cold neutrons confined in a bottle and counts the remaining neutrons to record decay, and the "Beam" method passes neutrons through a magnetic field and counting the fraction of decay products per neutron passing through. These studies were both verified, but their results are significantly different. The studies report a difference of 8.7±2.0 seconds, a value far outside of the bounds of uncertainty of both experiments. The puzzling thing about this difference is that in studies targeting other particles, the "Beam" and "Bottle' methods produce agreeing results. There are a few possible explanations that are very exciting, such as some interaction of the weak force interfering, or some "exotic" process unknown to us occurring in the experiments. However, on a much duller note, this difference, while unlikely, could be the result of error. To eliminate this possibility, the experiment needs to be repeated with more accurate apparatuses to rule out any significant chance that this observed difference is the result of error.

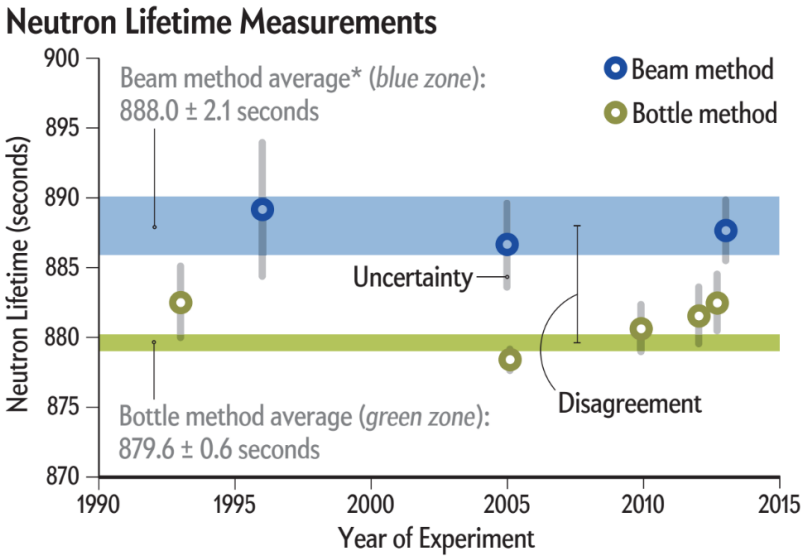

Figure 1: [1] *Diagram showing the disagreement of the 2 experiments over time, demonstrating the need for more accurate experimental methods*

2 Problem and Objectives

As sees above in figure 1, the beam method is naturally more susceptible to error with the nature of how the experiment is conducted, and to move forward in determining the precise lifetime of the neutron we need to find better experimental methods for the beam method. A contributor to the experimental error of the beam method is the detectors located outside of the beam trap. The

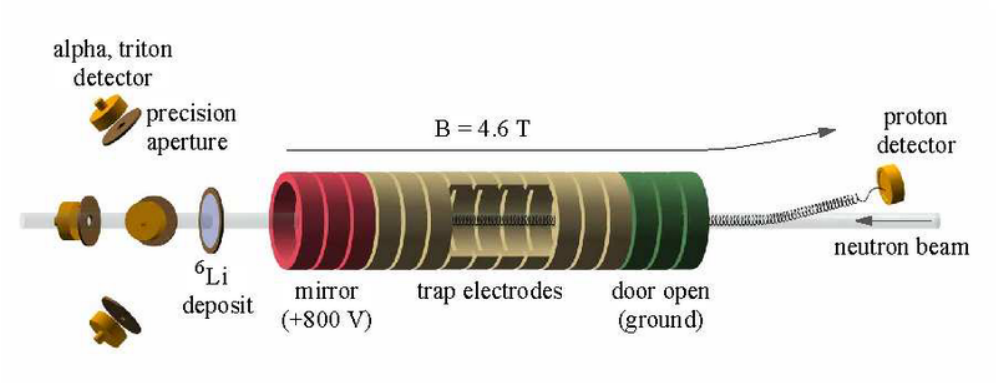

Figure 2: [2] *Schematic showing the detectors and apertures positioned with the proton trap*

width of the neutron beam results in a variation of the origin when considering the solid angle of incident particles to the apertures, therefore particles leaving the trap to be detected all have a differing solid angle to the detectors. This leads to an inaccurate measurement of total neutron flux, and therefore a less accurate experimental result. My mentor, Dr. Christopher Crawford, believes that it is possible to design apertures that create a uniform solid angle for all particles leaving the system, by limiting the solid angle of certain paths using caustics. Therefore my goal is to design precise azimuthally-symmetric apertures for the proposed BL3 experiment to reduce experimental error. Under the guidance of Dr. Crawford, I will work to develop a set of apertures that provides uniform solid angle, and therefore uniform acceptance for every detector, across the width of the neutron beam. My objectives to accomplish this are developing a repeatable coded process for accurately deriving solid angle of a parametric surface, creating an optimization problem with this process, and then producing CAD models using the best results from the optimization problem.

3 Procedure

To begin, I will work back through the work done by Joshua Young on the 2D proof of concept apertures to bring myself up to speed on the project and try

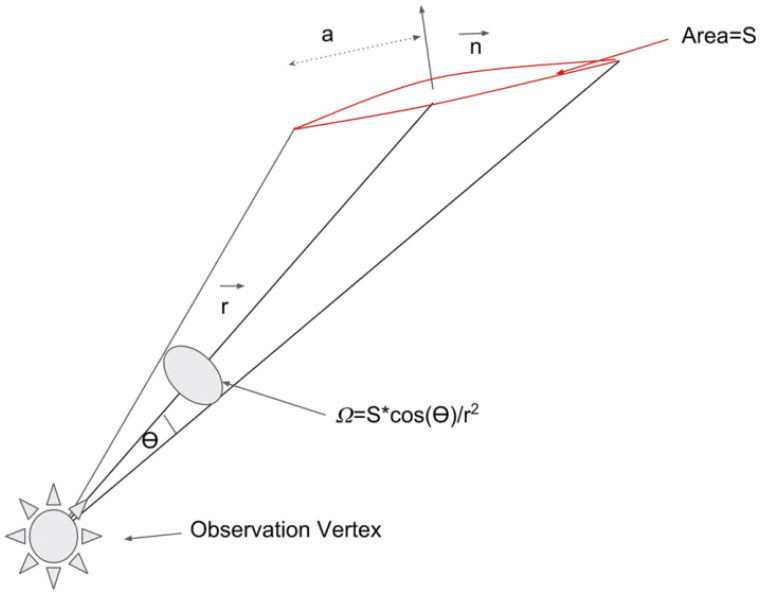

Figure 3: [3] *Graphic showing a similar process of calculating solid angle as proposed, for my work, the area of the surface will be defined as an unsolved integral, allowing for an optimization problem*

and spot any improvements or inefficiencies in the processes. Then I will derive a series of equations used to create an optimization problem who's solutions will be apertures for uniform or as close to uniform acceptance as possible. This will be done by taking advantage of the relationship between the differential solid angle and the differential equation that represents a cone created by the parameterized surface. With the best possible solutions, I plan on modeling them all with CAD and then 3D printing them, creating models to be used for physical testing of the design. With these models, I can begin to consider the best fit for the BL3 proposal, not just on their acceptance rate and uniformity, but on ease of construction, cost, and implementation into their model.

Starting work, I began exploring the capabilities of python and machine learning algorithms to try and optimize surfaces that give not only uniform but high acceptance. I came to realize that python has a shortcoming when working with mathematical functions. When creating defining a mathematical function in python you have to create it in the same way you would an algorithmic function, and it is stored as a unique data type. This makes adding, combining or modifying functions very difficult, and extremely difficult to implement for a learning algorithm.

Leaving python, I went on trying to establish a governing differential equa-

tion that represent equality amongst 2 detectors in the 3-Dimensional case. To simplify my starting point, I left the expressions that represent the aperture and the detector as variables, and wrote an equation that compares the solid angle from the center point to both detectors, an the solid angle from the left and right detector summed. This setup has its advantages, as I can represent the center point as one calculation doubled, as from the center point the 2 apertures are symmetrical. And when considering the solid angle from a non-center point, I can easily determine vectors from the point to the center of the detector and then normalize them, making unit vectors for the solid angle calculation. Then I went back and began to describe the detector in the equation. This proved to be the most challenging, as the most convenient way to describe the detector was with a Cartesian coordinate system, and the rest of my equation used a spherical system. This resulted in a very complex definition that will make modifying the equation to solve it very difficult. There is also the problem that in this equation I am solving for a surface, the aperture itself, and this process was foreign to me. I went about looking online for example problems or text on the subject but came up dry, and when speaking to Dr. Crawford about it, I came to realize that the solution to my equation would not be a well defined aperture, but a bound that would exist on all possible solutions. I do not believe that solving my equation would result in a usable solution, so I stored the idea for later. At this point in my work, I was admittedly lost, and needed to take the problem back down into its most basic roots. With the recommendation from Dr. Crawford, I am going back to a numerical approach using Matlab. I am writing a program that will numerically find the tangent to a curve from a differing starting location, starting in 2 dimensions and then moving into 3. This can be expanded to a program that can define tangent points of surfaces placed around the detectors, and therefore automate the process of evaluating the solid angle for a detector. Then using the same approach Joshua Young used to solve the 2 dimensional case, I will define 3 of the 4 surfaces that the particle could interact with, and using the Matlab program determine points on the 4th undefined surface that will provide uniform acceptance. Repeating this process for a lot of source points across the neutron beam, and then interpolating the results using Matlab splines will result in a final result.

To begin this process, I set the goal of doing this calculation with only data points as inputs. This makes the program slow, but is a purely numerical process. Data that represents surfaces can easily be constructed in Microsoft Excel using functions, but when computing the solid angle, only taking the data points as entries. This makes the program more applicable for different situations, especially surfaces that cannot easily be defined with non-peicewise or irregular functions. Based on Joshua Young's solution, ideal apertures will most likely be difficult to define regularly, so this is important to practicality of this situation. Next I had to make the decision on how to construct my algorithm. I began drawing out ideas for how to replicate the solid angle formula, shown in figure 5. I chose to calculate small differential areas along the surface of the aperture, instead of tracing a path that would represent the aperture projected onto a

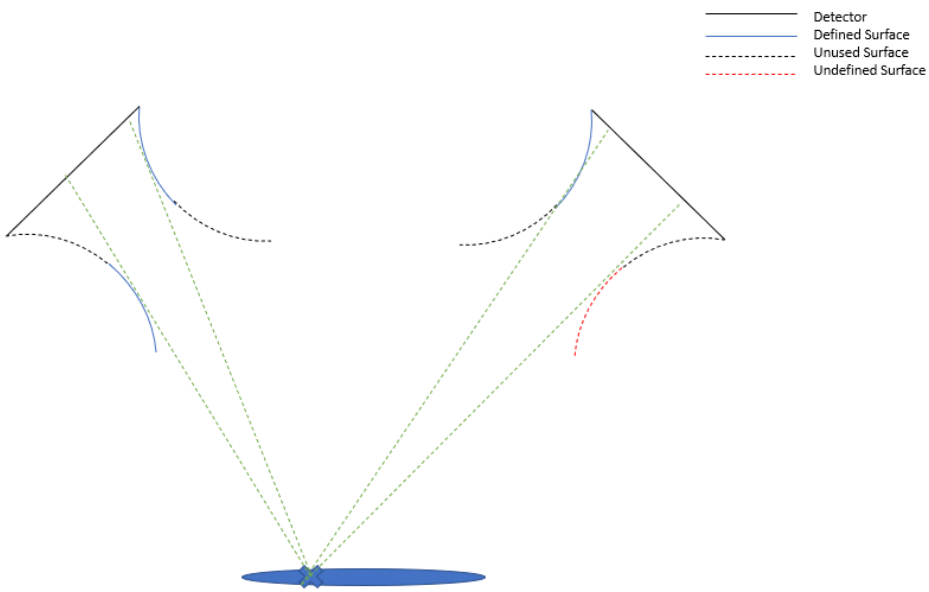

Figure 4: [1] *Diagram showing the "build up" process, assigning 3 of the surfaces to be constant, and using the 4th point of contact to build up a 4th surface that will provide uniformity. A good result will be a set of data points that can be interpolated into a defined function.*

plane perpendicular with the xy plane. I made the decision on the hypothesis that this would allow the program to be more accurate, due to the natural inaccuracy of splines in Matlab. Based on conversations with my research mentor Dr. Crawford, splines are good tools to represent a curve between a number of points, but take a while to compute and run into problems with curvature. Yes their curvature is smooth between data points, but may not accurately represent the curvature of the function that generated those points. I worried that in order to project the surface onto a parallel plane, the arc-length of the splines would create inaccuracy compared to the projection of the surface. Therefore, I elected to try and measure differential areas across the surface. I need to "slice" the aperture into sections, and at first Dr. Crawford suggested cylindrical slices. That is if the aperture is a cylindrical shape, a surface rotated around the z axis, cutting our slices from the xy plane. I realised this would create very thin rings, and it would require a lot of arc-length calculations, and splines, to get an accurate result, so I moved onto looking at a theta z plane, in cylindrical coordinates. This would slice the aperture into a curved line, and I could get away with one arc-length calculation for that entire slice, and I could use rectangular areas. If I took enough slices, my approximations would be accurate enough to get a usable result. At this point, I considered the need to

$$\Omega \equiv \int\int_S \frac{\hat{\mathbf{n}} \cdot d\,\mathbf{a}}{r^2},$$

Figure 5: [1] *Solid angle formula in vector notation*

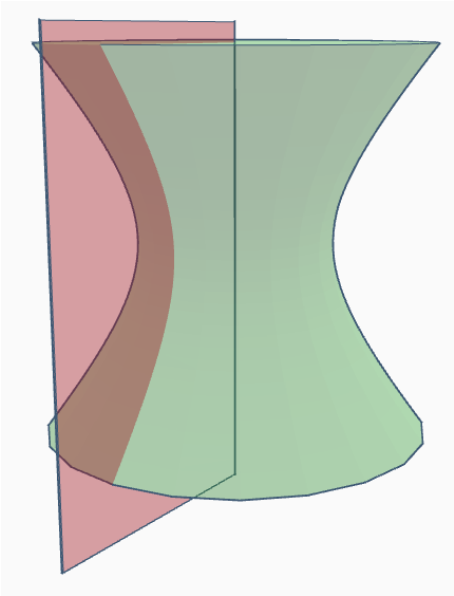

Figure 6: [1] *Model of hypothetical aperture and my proposed slice, the resulting intersection of the 2 bodies would be the slice, and calculations from that point are much simpler than cylindrical slices*

be able to calculate from points other than the origin. That is the end problem this research aims to solve, so including it now, especially if it is a simple method. I decided to go with the idea of shifting the aperture, so that our start point gets moved to the origin. This keeps the math simple, as a lot of my calculations are much simpler if we start from the origin, and can be done by generating data for the aperture at any point at the start, and matching them in an array. This way we don't need to take up economy in the program by moving the aperture every time, we just change the index we call in the array, and the program won't run any slower. Depending on how accurate we want to define the starting point, yes this array may be very very large, but other means of storing the data are possible, there is a method that gets around taking up a lot of economy. To go about this I just consider the angle and the radius that define the starting point, and find that displacement in xy coordinates.

Then adding or subtracting from the coordinates that define the surface. My plan is to at the beginning of the final program, calculate the shifted surface for however many starting points I want to measure, calculate the solid angle for each point, and than graph the solid angle as a function of radius and origin that define the starting point. For the time being, I chose to focus on making the algorithm work when starting at the origin, and then return to shifting the surface once I complete that. There would be no point in over-complicating the program before it is operational, especially when I expect to do a large amount of de-bugging and troubleshooting. For now, the less moving parts the better.

The other steps needed to replicate the formula are a distance and an orthonormal vector to the differential area. The distance is easy enough, and I don't have to get the distance, I can calculate the distance squared as it is what is needed in the formula. The interesting part is the normal. For the normal I chose to use a azimuthal normal, and a radial normal. These 2 angles account for all 3 dimensions needed, and therefore I would only have to calculate 2 dot products, not 3. This saves some economy for the program. I knew from the beginning that the more slices the better my accuracy would be, and with the needed calculations for normal I had to write them efficiently. For the aziumuthal normal, for each slice I am given an azimuthal angle. For a large number of slices, the gap in between slices will be very small, and I can use the points in the next or previous slice, at the same distance "down" the slice, that is the same distance from the origin, and imagine those 2 points on the xy plane. If I determine the slope created by these 2 points, than using arctangent you can determine the angle that line makes with the x axis. This determines the angle azimuthally. For the radial normal, I used a similar method. Going "up" and "down" the slice, finding the slope between those 2 data-points, and then determining the angle this line makes with the x axis. This angle accounts for a normal from the radius, and considers the polar coordinate as well. Now if I consider 2 unit vectors, each describing the angle from the differential surface, I can use those for the dot product in my formula.

The next step is calculating the differential area. This is simple enough, I can do an arc length with the distance from the z axis and the width of my slice, and that gives me a width, and the height is the length of the steps down the slices. Then a simple height times width rectangle approximation will give an area. For this method to work, I need to take a large number of slices, or the rectangle approximation will not be close enough.

To summarize my algorithm at this point, first I shift the aperture so that all of the datapoints match the startpoint, next I slice the aperture n number of times, then calculating differential areas up and down the slice. Then appending that onto a total. Going around azimuthally, I repeat this for 2π radians, and this total will yield a estimate of the solid angle, of any surface you give to it. Possible problems I see with this algorithm is the use of arctangent, while its discontinuities should be alright, as they are run back through sin functions so they

still yield a reasonable result, it is likely that near the extremes, there will be inaccuracy. Another issue may come in very flat surfaces, that is, "short" slices. The used method of calculating normal will have more inaccuracy the flatter the surface is, and using a lot of slices on a flat surface will just stack inaccuracy.

I coded this algorithm into Matlab as described, and began running tests on surfaces. The first surface I tested was a curve, very similar to figure 6 but only rotated around 90 degrees, and the results were promising. I was getting around .3 steridans of solid angle and I considered that to be possible. I then moved to testing a different surface, a disk with radius 5, with its center on the z axis, 5 meters up. I was easily able to determine the solid angle of this surface with a known formula. The formula gives 3.7699 steridians. Unfortunately, when running my program, I was getting results 5 magnitudes smaller than that, something that is obviously incorrect. I went back and ran the program again on a similar curve to the first I tested, and got a reasonable result, .4 steridians. This leads me to believe that the error I foresaw in flat surfaces, or surfaces without height, cause large programs with the program.

I went through trying to find the source of this error, and believe it is how height is sliced, as the heights for each slice are incredibly small. However, I do not understand how this is creating problems in my results. Yes the differential height is very small, but I am doing heights across the entire surface and it should still sum to something reasonable. I went back and tested this by summing just one azimuthal slice, and got a very low value. I hypothesised that this must be the result of their being no change in height, across the entire surface, so I tilted the surface slightly and got a more reasonable result, but still not accurate enough. This result was only 3 orders of magnitude too small. For just one slice, I went through the algorithm and tested a few points, and identified my issue. I had a typo that was causing the differential heights to be too small. I fixed this issue, hopeful to get a good result, and was disappointed to find that my result was now 2 orders of magnitude too large. Searching through my code, the differential height and width slices are still very small, and I am summing them the correct number of times, nothing stood out as unreasonable. Currently, I am still troubleshooting this issue, as even if the aperture is not very flat, flatter portions of it will still have error, and that amount of error won't be acceptable for the purposes of the BL3 Proposal. Carrying on, finishing this program to open possibilities is important, and with the number of people working on this project, will remain my priority. My method also comes with the advantage of being able to be more easily adapted into problems in other scenarios of physics, such as dealing with passable materials that have varied resistance, or more complex surfaces made of multiple materials that have different rates of reflection.

After a few months of being into this process, my research mentor Dr. Crawford came forwards with another method of numerical solving for solid angle. His

method uses a traced outline of the aperture, and a simple formula from there. The complex part of the code is tracing an accurate outline, and how many data points to include on it. That is the current work I have on my plate now, but it is too early in the process to begin documenting it. His method is very accurate for most every circumstance, the only exception being surfaces near the z axis. For apertures, this is unlikely, but still a possibility, that an aperture needs to be placed directly on the z axis to help tune uniformity. To accurately calculate the solid angle of this aperture, we will need another method. Currently my algorithm is the best proposed solution, aside from the fact that I am unable to get it to work. To finalize the aperture design, having a working method for the possibility of that aperture may be crucial, therefore it is important to continue this work. I have done all the debugging I think I am capable of, I have re-wrote my algorithm multiple times, but I still cannot find the source of this error. Continuing forwards will be working on the math, and making a new algorithm that takes into account the size of the aperture and how much it should be sliced. This optimization will hopefully lead to not only more effective results, but a more economical program, something that will be very important when it comes to tuning the apertures.

4 Significance of the Research

The value of this research has far reaching implications. It has the potential to change the way we approach particle detection in the future, for a large range of nuclear physics experiments. Finding ways to do this incredibly accurately, can help close the error bounds of particle experiments, and let us have more conclusive findings more often. It is no secret that particle experiments are very expensive, so anything that science can do to make them more practical will be huge. The BL3 proposal specifically, aims to better understand the neutron and the weak force, a large part of the standard model that we still have questions about. This has wide reaching implications about our understanding of the universe, all the way down to parity violation, antimatter, and the big bang theory.

Solid angle has a lot of other practical applications in physics, to include fluid dynamics, electricity and magnetism. For building precise machinery, or designing precise experiments, a method that can tune uniform solid angle will be invaluable, and open possibilities for engineers, physicists, and scientists in the future.

Solid angle in itself is something also applicable in the animation field. As digital lighting continues to develop, understanding solid angle is a challenge in computing, and methods that can determine the solid angle of virtual objects quickly and economically would be huge in furthering animation.

References

[1] Geoffrey L Greene and Peter Geltenbort. The neutron enigma, Apr 2016.

[2] David Bowman, Leah Broussard, S. Clayton, M. Dewey, N. Fomin, Kyle Grammer, G. Greene, P. Huffman, A. Holley, G.L Jones, C.-Y Liu, Mark Makela, M. Mendenhall, C. Morris, Jonathan Mulholland, K. Nollett, Pattie W., Seppo Penttila, M. Ramsey-Musolf, and Andrew Yue. Determination of the free neutron lifetime. 10 2014.

[3] Glenn Jocher, John Koblanski, Viacheslav Li, Sergey Negrashov, Ryan Dorrill, Kurtis Nishimura, Michinari Sakai, John Learned, and Saeeda Usman. mini-timecube as a neutron scatter camera. *AIP Advances*, 9:035301, 03 2019.