Wave Function Visualizer

Ramiro Leal-Cavazos
Stanford University
ramiro2@stanford.edu

Abstract

In this paper the implementation of a wave function visualizer in Unity is discussed. The software makes it very simple for the user to plot superposition of energy eigenstates of different quantum systems and evolve them with time according to the Schrodinger equation. Some of the limitations in computational power are discussed, as well as methods of improving some of the computational bottlenecks. The final product has the ability of plotting up to five energy eigenstates simultaneously, while rendering them for a Cardboard VR headset. Moreover, it also has built in all of the energy eigenstates of the hydrogen atom for a demonstration of the plotter capabilities.

1. Introduction

In physics research, advances are often made after developing great levels of intuition about a given physical situation. Often times this intuition is built by looking at how a specific mathematical model looks and changes with respect to its parameters. Tools such as Mathematica have powerful capabilities for visualizing functions, but they are limited to 2D screens. This means that mathematical objects such as probability density functions in 3D space are really hard to fully visualize.

In this paper I will talk about the framework I created that has as a primary goal the plotting of probability density functions in 3D space that can be evolved in time and rendered for virtual reality headsets. Specifically, the software specializes on probability density functions encountered in quantum mechanics, often referred to as wave functions. These wave functions evolve with time according to the Schrodinger equation, and it is this type of time evolution that I have made easy to do for any given wave function.

Even though the time evolution is specific to quantum problems, the function plotter can be used for visualizing any scalar three dimensional field such as the gravitational potential in different regions of space, and many other physics related phenomena.

2. Literature Review

Recently there has been some work done in using VR for data visualizations, such as research by Geoffrey Gray from the University of Washington, who analyzed the usefulness of showing scenes with scatter plots in a head-mounted display. He concluded that "the ability to 'fly around' a data set can reveal surprising results to the data sets structure. Patterns, clusters, banding, outliers and correlations are easy to spot within VR. Data visualization in virtual reality is truly an exploratory tool."[2]

One example of a VR tool for data visualization is Virtualitics, which helps its users analyze and visualize big data. Their tool is primarily aimed at analyzing highly dimensional data sets. It has great built in features such as AI driven analytics. However, it is not design to analyze evolving scalar fields of physical problems, such as in quantum mechanics.

Similarly, there is another tool that is being developed to analyze microscopy data, which allows one to analyze different small scale biological phenomena.[4] One advantage of this tool is that it is created for a very specific use, which is looking at data obtained from microscopes.

Another area of development in the uses of VR is for education purposes. VR has the capacity of creating immersive experiences about phenomena that otherwise would be impossible for students to experience, such as looking at the solar system, or experiencing the quantum world.

A study done on primary schools students in Ireland to see the effects of using virtual reality to teach science topics such as the water cycle showed that the students found great satisfaction from the activity. Over 90% of the students wanted to have VR used more often to learn.[1]

The project I will be talking about in this paper lies at the intersection of the two applications mentioned above. It allows easy data visualization of scalar fields often found in physics, specializing in quantum mechanical wave functions, while at the same time making it easy to create educational tools for physics classes. One educational example is allowing students to see how an electron cloud behaves in a hydrogen atom when the atom is in a superposition of states.

3. Overview of Quantum Mechanics

In quantum mechanics, the state of a system is represented by a mathematical object called a wave function, often represented by the symbol $|\psi\rangle$. To extract values from the wave function, one often evaluates the projection of the wave function onto a specific basis. A commonly used basis is the position basis. One then can write the inner product $\langle x|\psi\rangle$ to denote the projection of the wave function on the position basis vector $|x\rangle$. This is equivalent to the mathematical notation $\psi(x)$.

The principal characteristic of a wave function is that its modulus square in the basis of an observable represents the probability density of measuring that observable. In other words, if we are interested in measuring the position of a quantum object, then $|\langle x|\psi\rangle|^2 dx$ represents the probability of finding that object between positions x and $x + dx$.

The Schrodinger equation describing the evolution of the wave function $|\Psi(t)\rangle$ of a system with a Hamiltonian H is given by

$$i\hbar \frac{\mathrm{d}}{\mathrm{d}t} |\Psi\rangle = H |\Psi\rangle \tag{1}$$

$$\implies |\Psi\rangle = e^{-iHt/\hbar} |\psi\rangle \tag{2}$$

where $|\psi\rangle = |\Psi(0)\rangle$.

By the completeness of the eigenstates of the Hamiltonian, we can write any state as a linear combination of eigenstates $|n\rangle$ with corresponding energy E_n. In other words,

$$|\psi\rangle = \sum_n c_n |n\rangle \tag{3}$$

where the coefficients $c_n \in \mathbb{C}$ satisfy $\sum_n |c_n|^2 = 1$.

Therefore, for any state, its time evolution can be computed as

$$|\Psi\rangle = \sum_n c_n e^{-iE_n t/\hbar} |n\rangle \tag{4}$$

Note that the only time dependence is in the relative phases of the states to one another.

Since in this paper we are interested in plotting the probability density function of the wave function, then we need to be able to compute the quantity $|\langle x|\Psi\rangle|^2$. From section 7.1 in the Mathematical Appendix, we know that such a quantity can be expressed as

$$\begin{aligned} |\langle a|\Psi\rangle|^2 = \sum_{n=1}^{N} |c_n|^2 |\langle a|n\rangle|^2 + \\ 2 \sum_{i<j,i=1,j=2}^{i=N-1,j=N} \mathrm{Re}(c_i c_j \langle a|i\rangle \langle a|j\rangle) \\ \cos[((E_i - E_j)t + \theta_i(a) - \theta_j(a))/\hbar] \end{aligned} \tag{5}$$

Notice that the equation only depends on quantities relating to the physical system that need to be computed once: $c_n, \langle x|n\rangle, E_n, \theta_n(x)$.

4. Method of Plotting

In order to be able to plot the probability density functions, one needs to have a way of visualizing values at different points in space. Since cubes are simple shapes to render, for they have a small number of vertices, then I decided to create a box filled with smaller cubes. The way this was done in Unity is by creating copies of a single cube and placing them at their respective coordinates. This allows for the user to easily change the resolution of the cube, in case a resolution is too computationally expensive. Then, depending on the value of the probability density function at the center of each cube, the color and opacity of the cube is modified. Since probability density functions are always non-negative, then fully transparent cubes can represent zero, and the opacity can be increased for values greater than zero.

For the current version of the function visualizer, the red value of the cubes was kept at 1, while the green and blue values of the cubes were increased linearly with the value of the function at the point. The opacity was also increased linearly with the value of the function. Changing the opacity of the cubes allows the user to see high density regions behind low density regions, and changing the color means that the user can tell the relative value of the function at a given region by how bright the region looks.

4.1. Improvements

In Unity, it turns out that having 40^3 cubes present, which was the resolution used in this paper filling up the plotting region in figure 1, even when transparent, is very computationally expensive. Part of it has to do with the fact that all cubes must be rendered at all times because no cube is completely hidden by another. Moreover, shading has to also be computed for each cube.

To reduce the rendering load, the cubes were set to their most basic settings. In other words, they do not interact with any external light sources. Their color comes purely from ambient shading. The cubes also do not cause shadows on

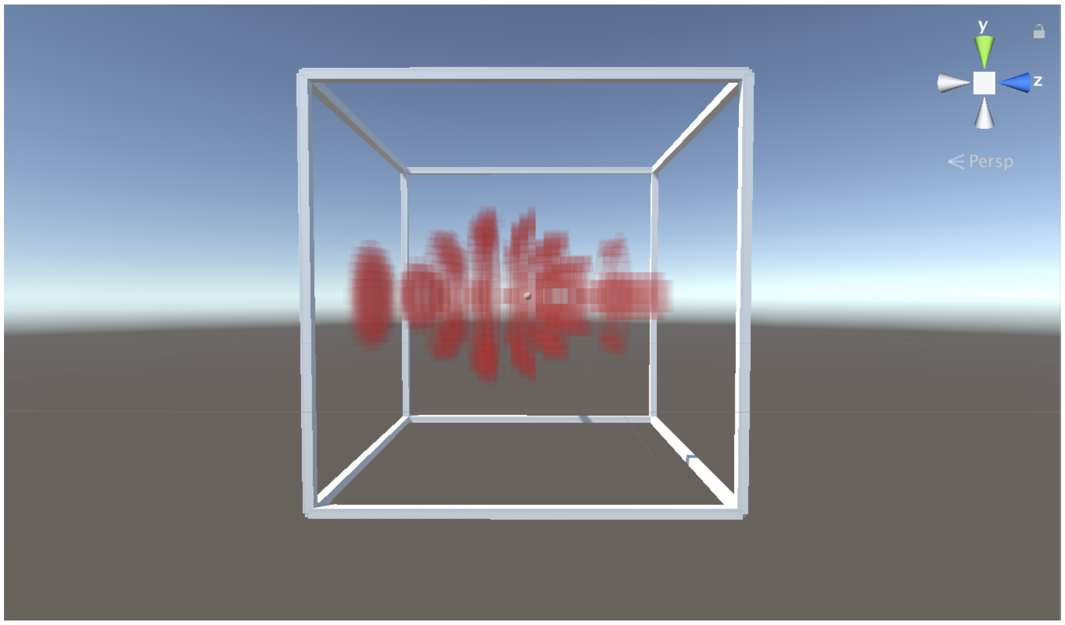

Figure 1. Plotting region in Unity

one another. However, even with these rendering settings, the plotting lagged dramatically.

A different approach to plotting would be to only place cubes in the regions where the function has a value larger than a given threshold. However, this would require making and deleting cubes constantly, which is not very efficient. Instead, what was done was to set as active the cubes with values larger than the threshold, and inactive otherwise. This means that for most functions, the number of cubes being rendered is almost always less than 40^3. Setting cubes to active and inactive was the largest improvement on performance made.

4.2. Rotation

Since the goal of this project is to make a virtual reality environment where one can visualize mathematical functions in space, then ideally one can just walk around the function to see it from different angles. However, it would also be nice to just be able to rotate the function plotted without having to move around. To do this, two methods were considered. The first is to rotate the location of the cubes representing the function. However, this would require to constantly update the position of many three dimensional objects.

Instead what was done was that when evaluating the value of the function, the vector representing the position to evaluate at was rotated by a given angle. The value of the function at that point was then computed, and it was assigned to the cube in the original position of the position vector, creating the effect of rotation. In the demo created, rotations are done by using the up and down arrows and only happen about the horizontal axis. The angle of rotation in degrees is displayed in the upper left hand corner of the plot, as seen in figure 2.

The main limitation with this method of rotating is that it requires re-evaluating the probability density function at every point whenever the angle is changed, and some probability density functions require many calculations to be evaluated.

4.3. Zooming

In order to zoom in and out of the plot, a similar method as for rotations was used. In order to keep the resolution constant, the cubes were left unaltered. The only thing that changes as the user zooms in and out is the length scale used to evaluate the values of the function at each point in space.

This method suffers from the same limitations as the method for rotating the function, which is the fact that the function must be re-evaluated at each point every time the length scale is modified.

4.4. Time Evolution

As mentioned at the end of section 3, the time evolution of the probability density function of a finite linear combination of energy eigenstates, given by equation (5), depends on quantities that must only be computed once. Moreover, the formula in equation (5) remains constant in complexity independent of how many eigenstates are being combined.

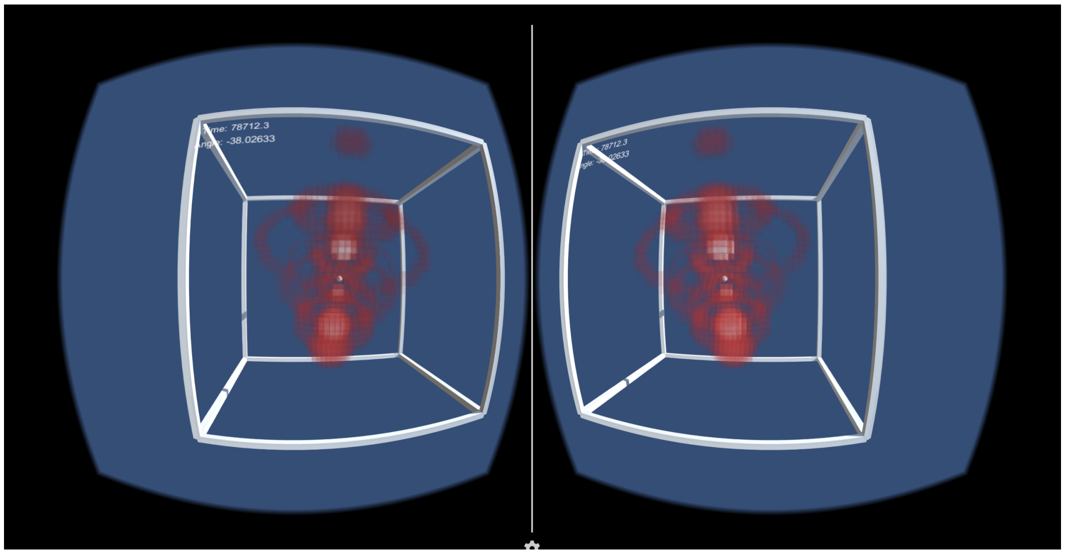

Figure 2. Stereo rendering of the $\psi_{12,7,0}$, $\psi_{13,6,0}$, $\psi_{14,5,0}$, $\psi_{15,4,0}$ and $\psi_{16,3,0}$ energy eigenstates of the Hydrogen atom.

It is this realization that allows for very quick time evolution of wave functions.

In order to make things efficient, the real part of each of the energy eigenstates being plotted at is point is saved into a three dimensional array. The same is done for the angle of complex phase of each of the eigenstates. Lastly, a vector of energies is also saved. Then, all that must be done is combine the matrices and vectors using equation (5), and displaying the final three dimensional matrix.

A nice consequence of the fact that for time evolution one only needs the initial state of the wave function is that the wave function could also be determined numerically and given to the software as a three dimensional array. When evolving in time, the plotter cannot tell the difference between a wave function obtained from a mathematical expression and a wave function numerically computed. Therefore, any system that can be approximated can also be plotted and evolved easily.

Notice that time evolution does not suffer from the main computational drain that rotations and zooming do, for the eigenstates must only be evaluated once.

However, there are still limitations intrinsic to the time evolution. One of the limitations is that the number of computations in equation (5) goes as $\mathcal{O}(n^2)$. Despite this, the computational bottle neck for plotting a small number of wave functions was still the action of evaluating their mathematical form when rotating and zooming on them. Therefore, the current demo only allows the linear combination of up to 5 energy eigenstates, for otherwise it starts to become unusable.

4.5. Max Value Coloring

In order to have the plot of the probability density function always visible regardless of what range of values are being looked at, the maximum opacity is calibrated to be the maximum value of the function in the region being plotted.

If the function is rotated or scaled, the maximum value of the function in the region can potentially change. This means that every time any of those operations is performed, the maximum value needs to be updated. This can be computationally expensive if the function requires many calculations, but it makes sure that whatever is being plot is always visible to the user.

Moreover, since when time evolving a probability density function, what is important is to see how the probability moves around, the maximum value of the function is not updated when evolving time. Similarly, since the maximum is expected to remain almost unchanged when the plot is being rotated, the value is left unchanged during rotations. The maximum value is only updated when the function is scaled.

4.6. Stereo Rendering

In order to create the stereo rendering needed for a virtual reality headset, the Cardboard SDK provided by the Electrical Engineering Department of Stanford University was used.

Since the dimensions of the plotting region are 40^3 in Unity units, this means that the observer has to be standing from very far away to look at the entire plot. Therefore, the effect of depth was rather minimal. In order to improve this, the separation of each eye in the stereo rendering was increased from 0.06 to 16, as can be seen in figure 2. This creates a more exaggerated effect of depth for the user, making it easier to distinguish which parts of the wave function were close by and which ones far away.

5. Hydrogen Atom

To show the capabilities of the program, I decided to include for the user all of the energy eigenstates of the Hydrogen atom, which is one of the few quantum mechanical system that can be solved and has a closed form solution. In atomic units, the solution to the hydrogen atom is[3]

$$\psi_{n\ell m}(r,\theta,\phi) = \sqrt{\left(\frac{2}{n}\right)^3 \frac{(n-\ell-1)!}{2n(n+\ell)!}} \\ e^{-r/n}\left(\frac{2r}{n}\right)^{\ell} L^{2\ell+1}_{n-\ell-1}\left(\frac{2r}{n}\right) Y^m_\ell(\theta,\phi) \quad (6)$$

where L are the generalized Laguerre polynomials and Y are spherical harmonics, and n, ℓ, m are the principal quantum number, the total angular momentum quantum number, and the z-component of angular momentum quantum number, respectively.

5.1. Mathematical Tools

The Laguerre polynomials, as well as the Legendre polynomials that appear in the definition of the spherical harmonics are commonly used in many areas of physics. Therefore, I decided to implement them in the plotting engine.

A consequence of the complicated close form of both types of polynomials is that other important mathematical functions were also included. One example is a function to calculate the factorial of half-integers.

6. Future Improvements

The current state of the wave function visualizer is more of a proof of concept. There are many things that can be improved on. Thus, in this section I will mention a few of the changes that will be implemented in later versions.

First, since we are already storing the values of the wave function at each point into a three dimensional array before displaying anything, then it would be nice to store initially a higher resolution version of the plotting region and use those stored values for scaling and rotating, only re-evaluating the wave function whenever the user has zoomed in more than the resolution stored. This will dramatically increase the computation speed. Additionally, one can also incorporate smart resolution, which becomes coarse when needing to re-evaluate a wave function, and becomes finer with time.

Similarly to how the coloring of the function is determined by what is in the plotting region, it would be nice for the software to be able to detect the major features of the wave function and scale the length scale appropriately so that from the beginning the user is looking at the most important information. This can be done easily with wave functions, for most of them decay as one gets away from the origin. Thus, one can just find a region including all points with values greater than some threshold.

The evaluation of the mathematical expression of the wave function can also be improved by using a library optimally designed for matrix manipulations, such as the numpy library in python. At the current state of the software, for loops are used to determine the different polynomials of the hydrogen atom, and this is something that matrix manipulations could dramatically speed up.

Another useful change will be to make it easy for the user to input the mathematical form of a wave function. Currently the only way of inputting wave functions is by coding them locally in C#. It would be much nicer to be able to write the equation simply using mathematical symbols.

Lastly, head movement tracking of the headset will also be implemented to create a more immersive experience.

7. Mathematical Appendix

7.1. Time Evolution of Quantum Eigenstates

Proposition 7.1. *Consider a state $|\Psi\rangle$ that can be expressed as a finite linear combination of energy eigenstates $|n\rangle$ with energy E_n:*

$$|\Psi\rangle = \sum_n c_n e^{-iE_n t/\hbar} |n\rangle \quad (7)$$

Then, the probability density of such a state in the basis $\{|a\rangle\}$ of the observable A is

$$|\langle a|\Psi\rangle|^2 = \sum_{n=1}^{N} |c_n|^2 |\langle a|n\rangle|^2 + \\ 2\sum_{i<j,i=1,j=2}^{i=N-1,j=N} \mathrm{Re}(c_i c_j \langle a|i\rangle \langle a|j\rangle) \\ \cos[((E_i - E_j)t + \theta_i(a) - \theta_j(a))/\hbar] \quad (8)$$

Proof. The proposition will be proven by induction.

Base Case: $n = 2$

Suppose our state $|\Psi\rangle$ can be expressed as the linear combination of eigenstates $|1\rangle, |2\rangle$:

$$|\Psi\rangle = c_1 e^{-iE_1 t/\hbar} |1\rangle + c_2 e^{-iE_2 t/\hbar} |2\rangle \quad (9)$$

Then, the quantity $|\langle a|\Psi\rangle|^2$, where $|a\rangle$ is an eigenstate of an arbitrary observable A, is given by

$$\begin{aligned}|\langle a|\Psi\rangle|^2 = |c_1|^2|\langle a|1\rangle|^2 + |c_2|^2|\langle a|2\rangle|^2 +\\ c_1c_2^* \langle 2|a\rangle \langle a|1\rangle\, e^{-i(E_1-E_2)t/\hbar} +\\ c_1^*c_2 \langle 1|a\rangle \langle a|2\rangle\, e^{-i(E_2-E_1)t/\hbar}\end{aligned} \tag{10}$$

Since the modulus square of any complex quantity must be real, then the sum of the last two terms must be expressible in terms of purely real quantities. Notice that the two terms are the complex conjugate of one another. To simplify things, we can extract all of the complex phase information from the coefficients and the eigenstates by writing them as

$$c_i \langle a|i\rangle = \mathrm{Re}(c_i \langle a|i\rangle)e^{-i\theta_i(a)} \tag{11}$$

Equation (10) then becomes

$$\begin{aligned}|\langle a|\Psi\rangle|^2 = |c_1|^2|\langle a|1\rangle|^2 + |c_2|^2|\langle a|2\rangle|^2 +\\ 2\,\mathrm{Re}(c_1c_2 \langle a|1\rangle \langle a|2\rangle)\\ \cos[((E_1-E_2)t + \theta_1(a) - \theta_2(a))/\hbar]\end{aligned} \tag{12}$$

Inductive Step: $n = N-1$

Suppose the formula holds for $n = N-1$. In other words, for $|\psi\rangle = \sum_{n=1}^{N-1} c_n |n\rangle$. Then,

$$\begin{aligned}|\langle a|\Psi\rangle|^2 = \sum_{n=1}^{N-1} |c_n|^2|\langle a|n\rangle|^2 +\\ 2 \sum_{i<j,i=1,j=2}^{i=N-2,j=N-1} \mathrm{Re}(c_ic_j \langle a|i\rangle \langle a|j\rangle)\\ \cos[((E_i-E_j)t + \theta_i(a) - \theta_j(a))/\hbar]\end{aligned} \tag{13}$$

Then, for a state $|\Phi\rangle$ defined as

$$|\Phi\rangle = c_N e^{-iE_Nt/\hbar} |N\rangle + |\Psi\rangle \tag{14}$$

the quantity

$$\begin{aligned}|\langle a|\Phi\rangle|^2 = |c_N|^2|\langle a|N\rangle|^2 + |\langle a|\Psi\rangle|^2 +\\ c_N e^{-iE_Nt/\hbar} \langle\Psi|a\rangle \langle a|N\rangle +\\ c_N^* e^{iE_Nt/\hbar} \langle N|a\rangle \langle a|\Psi\rangle\end{aligned} \tag{15}$$

Using the same procedure as in equation (11) for each term in $\langle a|\Psi\rangle$, we see that the last two terms in the above equation reduce to

$$\begin{aligned}c_N e^{-iE_Nt/\hbar} \langle\Psi|a\rangle \langle a|N\rangle + c_N^* e^{iE_Nt/\hbar} \langle N|a\rangle \langle a|\Psi\rangle =\\ 2\sum_{i=1}^{N-1} \mathrm{Re}(c_ic_N \langle a|i\rangle \langle a|N\rangle)\\ \cos[((E_i-E_N)t + \theta_i(a) - \theta_N(a))]\end{aligned} \tag{16}$$

Therefore,

$$\begin{aligned}|\langle a|\Phi\rangle|^2 = \sum_{n=1}^{N} |c_n|^2|\langle a|n\rangle|^2 +\\ 2 \sum_{i<j,i=1,j=2}^{i=N-1,j=N} \mathrm{Re}(c_ic_j \langle a|i\rangle \langle a|j\rangle)\\ \cos[((E_i-E_j)t + \theta_i(a) - \theta_j(a))/\hbar]\end{aligned} \tag{17}$$

Thus, by the inductive hypothesis, the proposition has been proven. *Q.E.D.*

References

[1] D. Bogusevschi1, M. Bratu, I. Ghergulescu, C. H. Muntean, and G.-M. Muntean. Primary school stem education: Using 3d computer-based virtual reality and experimental laboratory simulation in a physics case study, 2018.

[2] G. Gray. Navigating 3d scatter plots in immersive virtual reality, 2016.

[3] R. Shankar. *Principles of Quantum Mechanics*. Springer, 2008.

[4] R. P. Theart, B. Loos, and T. R. Niesler. Virtual reality assisted microscopy data visualization and colocalization analysis. *BMC Bioinformatics*, 18(64), 2017.