Geodesic and Wavepacket Propagation through Schwarzschild Spacetime

Ari Cooper

May 9, 2017

0.1 Introduction

Background Through taking Cosmology with Professor Watkins at the Physics department, I was introduced to a lot of interesting and complicated concepts that challenge our preconceptions of the natural world. One of these such concepts, is the force of gravity. Everyone understands what gravity is and what it does, in the general sense. But if you were to ask someone why, most would have no idea where to begin.

This is where I think there is a lot of potential. In Cosmology, I learned about world lines, which refer to the path taken by a particle between two points in spacetime. The path of a free particle over its existence can tell us a lot about the forces acting on it. There is a certain path that we can predict, and show, a free particle would take in spacetime. This path is defined as the world line with the longest proper time, or as the world line with the straightest possible path through spacetime, and is called a geodesic [5]. Thus taking an apple and dropping it from your hand to the ground is allowing the apple to follow a geodesic from your hand to the ground where it was stopped again from following the pull of gravity. Why does the apple do this? Well, according to the Geodesic Hypothesis, free particles want to follow geodesics, and thus when an outside force is absent, it will follow its geodesic path [5]. It is important to note that the Geodesic Hypothesis also dictates that geodesics are the longest proper time between two points [5]. This explains why General relativity says that the closer you are to a heavy mass, such as Earth, the slower time will move for you. For example GPS satellites in orbit of Earth have to be calibrated for the time difference as time runs faster for the satellites than it does for you driving your car on the surface of the Earth [3].

Taking another look at how objects accelerate towards the ground when we drop them may make more sense. When you drop an object high off the ground it maintains the same downward acceleration as we all know, the force of gravity, but why does this acceleration happen? Therefore, since the object follows a geodesic, it will fall along its geodesic path, which travels to the center of the Earth. Along this path the rate of time is constantly changing, where the closer the apple is to the center of the Earth, the slower time will travel. Therefore according to the Geodesic Hypothesis, the apple will travel slower far from Earth and move faster the closer it comes in order to accrue the longest possible proper time. That is why there is an acceleration towards the center of a massive object like the Earth.

From this it is clear that gravity is not actually a force but rather just a result of matter traveling in a curved spacetime and furthermore, general relativity. Taking a new approach to understanding gravity is something I think everyone, even without a science background, can enjoy if only because its such a strange concept initially.

Proposal Now taking this a step further, a wave packet should also follow a geodesic derived from the space-time metric in the same manner as a free particle. Since a free particle can be represented as a wave packet, the physics should follow accordingly. The difficulty though, is the concept of plotting a wave packet through a changing spacetime, in order to visualize gravity. This perspective hasnt been looked at or even explored very thoroughly, so I thought it would be fascinating to set up a simulation which visualizes these packet's behavior and path through a metric of spacetime. Through this simulation one would be able to visually see that as the wave packet travels through spacetime the rate of time changes, which is reflected in the wave packet's behavior, namely its group velocity.

To accomplish this I will use MatLab to create the simulation using a GUI through the built in GUIDE tool. MatLab is designed for various computational sciences, complete with many built in functions for mathematics and graphics, making it a useful resource for this project. The simulation with be user controlled through the GUI, allowing for various manipulations of variables in order to visualize the propagation of a wave packet through Schwarzschild spacetime in the presence of a spherical mass.

0.2 Implementation

Matlab There is some sample code from other projects or demos of certain related MatLab functions. I will need to research more about implementing GUIs and creating a 3-D dynamic plot, which is what I plan to use. In order to constantly update the animation while the code is running, I can use the drawnow command and generate real-time results so that the user will be able to mess around with some variables and see how they affect the wave packets propagation. Most of the variables will be set for initializing the wave packet however.

Creating the dynamic function of time over the plot will be the interesting part. This is where the majority of my time will be focused. The goal is to incorporate a function of time that changes throughout the 3-D space of the animation in order to visualize the wave packet. This function of time can be generated using the geodesic equation in order to calculate the path of the wavepacket through spacetime around a mass which I will get into in more detail later on. I can then use the geodesic path of the packet to extrapolate the curvature of time along those points, which will then allow me to plug in the time at each point in order to solve the time dependent Schrodinger equation and thus find the behavior of the wavepacket along its geodesic.

The first step of the project involves creating a GUI for the program so that a user can use it and see the simulation easily. Having made GUI in NetBeans with java I have a basic understanding, but there are commands specific to MatLab I need to know. First of all, to create a GUI in MatLab I have to use GUIDE, the graphical user interface development environment that comes with MatLab.

Metric In order to plot the probability density of a wave packet, I must first define the metric in which I will be working. In order for this simulation to be enlightening, at least in a strictly technical manner, I will be using the Schwarzchild metric, a solution to the Einstein Equations [5]. In studying General Relativity different metrics can be useful for different calculations. For the purpose of this simulation, I will be showing how a wave packet in Schwarzchild spacetime will propagate along its geodesic. The Schwarzschild metric is a good choice as it is a spherically symmetric metric, diagonal, and time-independent, which will prove to make many calculations simpler. It allows for easier calculations of geodesics and conceptual understanding of curved space and time around massive spherical objects like stars and planets.

Taking a closer look at the Schwarzchild metric, $ds^2 = -(1 - \frac{r_s}{r})dt^2 + (1 - \frac{r_s}{r})^{-1}dr^2 + r^2 d\theta^2 + r^2 sin^2\theta d\phi^2$, we can see that there are four parts to this equation. In order from left to right we have our coordinates (t, r, θ, ϕ) . These are all the variables that define the curvature of 4D spacetime, t being time, r being the circumferential radial coordinate, r_s is a constant called the Schwarzchild radius, θ and ϕ are spherical coordinates. We can see that as r approaches ∞, the metric becomes $ds^2 = -dt^2 + dr^2 + r^2 d\theta^2 + r^2 sin^2\theta d\phi^2$. This is flat spacetime in spherical coordinates, or the Minowski metric which tells us that spacetime isn't curved far away from a static spherical mass, but the closer to the object, say Earth, the more curved spacetime becomes[5]. This metric holds true for $r > 2GM$, and this is where r_s, the Schwarzschild radius comes in. This radius shows that the metric falls apart at this point, making it hard to describe the spacetime very close to a mass like what heppens near the event horizon of a black hole. For my purposes here, I will not be dealing with such a senario and thus this limitation can be afforded. Thus if we substitute $r_s = 2GM$, we are left with the final Schwarzchild Metric tensor I will use :

$$ds^2 = -(1 - \frac{2GM}{r})dt^2 + (1 - \frac{2GM}{r})^{-1}dr^2 + r^2 d\theta^2 + r^2 sin^2\theta d\phi^2 \tag{1}$$

Geodesics From here it is important to grasp what following a geodesic in curved spacetime entails. Geodesics in curved spacetime can be defined in a variety of equivalent ways. I will treat geodesics as solutions of a set of ordinary second-order differential equations. The initial conditions for these equations are an initial position and direction of a geodesic, which will be the point of emition of the wave packet. The geometry of a spacetime completely determines the geodesics. We can use geodesics to study both the curvature of spacetime and the physical motion of objects in that spacetime. Thus the visualization of spacetime via display of geodesics provides both geometric and physical insight [3].

We can then solve the geodesic equation using the Schwarzchild Metric Tensor introduced earlier. In this way I will be able to create a visual representation of the geodesic path, which is made simpler due to certain conserved quantities that arise from the fact that this metric is diagonal and time independent. The fact that it is diagonal allows us to ignore many components of the tensor since they become zero. Also, analyzing these components we see that they are time independent and all are dependent on r[5]. The form of the Geodesic Equation that will be useful for these calculations takes the form :

$$0 = \frac{d}{d\tau}\left(g_{\mu\nu}\frac{dx^{\nu}}{d\tau}\right) - \frac{1}{2}\partial_{\mu}g_{\alpha\beta}\frac{dx^{\alpha}}{d\tau}\frac{dx^{\beta}}{d\tau} \tag{2}$$

Using this equation we can plug in the four components, (t, r, θ, ϕ), for μ. When plugging in these components we find that the t and the ϕ components lead to conserved quantities as the metric is diagonal, and is time and ϕ independent. From the $\mu = t$ component we get constant $= -g_{tt}\frac{dt}{d\tau} = \left(1 - \frac{2GM}{r}\right)\frac{dt}{d\tau} \equiv e$ and from the $\mu = \phi$ component we get constant $= g_{\phi\phi}\frac{d\phi}{d\tau} = r^2 sin^2\theta\frac{d\phi}{d\tau} \equiv l$. From these equations we can then acquire the radial equation of motion by solving for the $\mu = r$ component in terms of e and l. The $\mu = r$ component of the geodesic equation is a bit more complicated as all of the metric components depend on r, but using the constants we just found along with a few other tricks, the radial equation of motion turns out to be :

$$\frac{1}{2}(e^2 - 1) = \frac{1}{2}\left(\frac{dr}{d\tau}\right)^2 - \frac{GM}{r} + \frac{l^2}{2r^2} - \frac{GMl^2}{r^3} \equiv E \tag{3}$$

From this equation I will be able to plot the predicted geodesic path of the wave packet, which is confined to the $\phi = \frac{\pi}{2}$ plane, simplifying things considerably. In order to write a program which accomplishes this, I referred to a method mentioned in Moore in A General Relativity Workbook, which I will outline briefly. An easier way to actually be able to plot the course of a particle is to use the form of the equations for radial and phi :

$$\left(\frac{dr}{d\tau}\right)^2 = -\frac{GM}{r^2} + \frac{l^2}{r^3} - \frac{3GMl^2}{r^4} ...and... \left(\frac{d\phi}{d\tau}\right)^2 = \frac{l}{r^2} \tag{4}$$

From these equations we need to generate an equation for $r(\tau)$ and $\phi(\tau)$. This requires replacing the double-derivatives with finite difference approximations which is done by doing Taylor Series Expansions. From the expansions Moores able to generate the equation for $r(\tau + \Delta\tau) = r_{n+1}$ and ϕ_{n+1} :

$$r_{n+1} = 2r_n - r_{n-1} + \Delta\tau^2(-\frac{GM}{r_n^2} + \frac{l^2}{r_n^3} - \frac{3GMl^2}{r_n^4}) \tag{5}$$

$$\phi_{n+1} = \phi_n + \Delta\tau \frac{l}{[\frac{1}{2}(r_{n+1} + r_n)]^2} \tag{6}$$

Notice that there must be some initial conditions since we have to know the time step behind of us in order to generate the next. This was accounted for by assuming we are at an extreme point in the orbit and therefore $\frac{dr}{d\tau} = 0$. This allows for some assumptions which let us include a half step before the first time step, $r_{\frac{1}{2}}$. By using an initial angular momentum and the initial r, l and $\phi_{\frac{1}{2}}$ are calculated and with a small enough choice for the time step $\Delta\tau$, we have everything we need for future time steps. Using this equation, I implemented a loop over a period which was proportional to the period of a circular orbit τ_c and iterated over the equation, storing values in an array. On each step I added the r and ϕ values into an array GP as a vector, with the index being the time step. This way I could easily plot a full orbit no matter how large the initial angular velocity. The propagation of the particle following the calculated trajectory will be represented using the animatedline function in MatLab. I can plot the trajectories in three dimensions by using the curvature of space and time represented by the r and t metric components respectively to create a function of $z(r)$. In the same plot I will implement Flamm's paranoid, the common representation of space curvature around a mass. The user will have the option to see the 3D planar trajectory, 2D flat spacetime trajectory, or the 3D model of the planar trajectories rotated around the origin, creating a representation that is designed to remind the user of that I am only showing one plane of an orbit in the formerly mention 3D model.

Wave Packet Propagation Now that I have a proper visual representation of the geodesic in Schwarzschild spacetime, I can plot the wave packet's propagation through this spacetime and see if it truly does follow the predicted geodesic. Firstly, a wave packet is a short burst of localized waves that travel at a group velocity together. The packet is made up of the superposition of an infinite number of waves each with varying wave number, phase, and amplitude. Each component wave has a corresponding wave function which is a solution to the wave equation. For my purposes here, I will be using the a relativistic form of the Schrodinger equation, the Klein-Gordon equation as my wave equation :

$$\partial_t^2\Psi(r,t) - c^2\nabla^2\Psi(r,t) + \mu^2c^2\Psi(r,t) = 0 \tag{7}$$

where $\mu = \frac{m_0 c}{\hbar}$. This equation is a partial differential equation that governs the quantum evolution of wave functions for relativistic spinless particles. The KGE is more applicable to various situations than the Schrodinger equation, so it will become useful in understanding relativistic and non-relativistic particles [6]. There are many methods for solving this equation, each with their own benefits and drawbacks. Which method I choose will depend on how accurate is will be for this kind of simulation, and whether or not I will have to create my own wave packet solver to solve parts of the equation for me. In order to accommodate for using a spherical metric I can use a version of the Klein-Gordon equation for curved spacetime

$$0 = -g^{\mu\nu}\partial_\mu\partial_\nu\Psi + g^{\mu\nu}\Gamma^{\sigma}_{\mu\nu}\partial_\sigma\Psi + \frac{m^2c^2}{\hbar^2}\Psi \tag{8}$$

$$= \frac{-1}{\sqrt{-g}}\partial_\mu\left(g^{\mu\nu}\sqrt{-g}\partial_\nu\Psi\right) + \frac{m^2c^2}{\hbar^2}\Psi \tag{9}$$

From this equation I will be able to gather the path of propagation through the Schwarzschild spacetime by solving for the wave equation in Schwarzschild coordinates :

$$0 = \left[\left(\frac{r-1}{r}\right)^2 \frac{\partial^2}{\partial r^2} + \frac{(r-1)}{r^3}\frac{\partial}{\partial r} - \frac{\partial^2}{\partial t^2} - V(r)\right]\Psi(r,t) \tag{10}$$

An easy solution for this wave equation to make a wave packet is a Gaussian wave packet. Using this as the initial condition for the wave function, I can plot its peak along the orbital plane and watch its time evolution move the center of the peak along the predetermined path. In order to create a function which generates the evolution of the wave packet at each time step, I have a number of possible methods in which to solve the partial differential equation which is the wave equation. I found that using an implicit finite difference method would be the most useful way in which to program a method in which to solve the equation. I found a good method in a paper done by Pierce which includes how to set initial boundary and initial conditions in order to calculated the time evolution of the packet[7]. The equation for the new wavepacket solution for the 1D equation was given as:

$$u_n^{k+1} = r^2u_{n+1}^k + 2(1-r^2)u_n^k + r^2u_{n-1}^k - u_n^{k-1} \tag{11}$$

This difference approximation is done for the wave equation $u_{tt} = r^2u_{xx}$, and in order to simplify things, I have simplified the KGE to simply be the wave equation with $r = g_{tt}$ for the propagation of a wave packet from rest. To see that the packet accelerates from the initial radius down towards $x = 2$, where g_{tt} goes to infinity. A similar method can take place in order to calculate the 2D wave equation in r and ϕ, in a later implementation of this program.

GUIDE and FEAtool GUIDE is a built in feature of MatLab designed to make GUIs while FEAtool is a separate toolbox that I have downloaded a free version of. There is a distinct difference between the two. While GUIDE is designed for creating a GUI from scratch, the FEAtool is a MatLab toolbox for modeling and simulation of physics, partial differential equations, and mathematical problems. To do this, they use an all-in-one package which includes a graphical user interface to do it all in. FEAtool has various physics modes from which to choose from, none of which I think I will actually be using as they dont further the simulation Im trying to make. It does have a very useful feature here though, which allows a user defined equation. Furthermore it has plenty of options detailing the process of your equation complete with variable declarations and initialization of the function. Thus I would be focusing more on the actual model aspect of the simulation involving the physics.

Due to this difference in purpose between GUIDE and FEAtool, I plan to use FEAtool as a means of easily implementing and solving an equation that I want to test to make sure it works. This can become very helpful as I will be able to test my method step by step with FEAtool, ensuring that any error in the GUI I create is not due to an error in the mathematics. In this way, I can test out the various wave equation and geodesic equation solvers I will need to use and make any changes necessary before making a mess of anything in the GUI within GUIDE.

GUIDE To start GUIDE from Matlab, type guide into the Matlab prompt. This will bring up the GUIDE Quick Start dialog box, and from there I will start a Blank GUI. A larger window pops up with the GUI editor. This editor is where all of the components and plots are available to drag and place into the GUI space. For my purposes I will create a separate window for holding the settings and other controls as I dont want to clutter the plot too much. GUIDE automatically creates a file in Matlab which saves all of the preferences and places components of the GUI. It is also in this script that I will be able to link various buttons and other inputs that need to be accessed to update the model. Most of the processing will commence when the user presses the simulate button, which will render the simulation with the given initial conditions [1]

Having looked into the differences of designing a GUI in Matlab compared to Netbeans, there are a few key points. One is that fields and variables are held in handles in Matlab. To access a variable one has to call the handle first which is correlated to the object that is being called. For instance if I want a button to perform an action and change a variable the handle may be given to me in the callback's parameter, or I will have to call the handle myself, as for a global variable. I can then access the handle and proceed to change it, finishing with a method which saves the changes I have made [4]. I will need to save my initial conditions for the particle I am simulate in this manner as I will need to access them for both solving the geodesic equation and the wave packet equation. It is this code where most of the physics will be calculated. There have been other projects for simulating wave packet propagation, using the TDSE, time-independent Schrodinger Equation, similar to what I am trying to do. Using this code as an example, I will be able to create a more specialized model.

0.3 Conclusion

Next Steps At this point I have outlined the general method I will be using to implement a GUI in Matlab. From here I will be able to take the next steps and figure out the best way of generating the results I want. I have come across various equation solvers that allow for the input of variables that will generate solutions, which I could just include in my project to expedite some of the math involved. For other calculations, I will have to create my own functions as well.

As a finished product I want the program to be easy to use, and appeal to any level of understanding of gravity. At the very least, the simulation will be aesthetically pleasing and fun to watch. I might even implement a way to save the video or take a snapshot if a certain path is cool or interesting in some way. Ideally, this will be a proof of concept of wave packet propagation depicted in an artistic way with clear representation.

0.4 Addendum

User Guide Upon launching the simulator, three windows will appear. One window controls the parameters for the orbits, one contains the main plot and toggles for plot type, and lastly a figure with a plot of a geodesic of a particle at rest. The control window allows for the manipulation of simulation duration, meaning for ho long the program calculates the trajectory. At one the orbit will plot for one full rotation. The user can also manipulate the initial angular momentum of the particle, thus changing its path around the center mass. Furthermore, the distance from the center of the mass can be manipulated, and finally the mass of the object orbiting, scaling GM.

Once the initial conditions are chosen in the control window you can then select the type of plot from the main window with Flamm's paraboloid plot. The user can then choose to see the orbits through either the wave packet propagation calculation or the geodesic calculation by select one in the drop down menu. There is then a choice of three graphs that display the trajectories in different ways. One is a flat space representation on a polar plot, the next a 3D model of the planar orbit plotted above Flamm's paraboloid to show the curvature of spacetime. Lastly there is a 3D plot which takes the planar orbit and rotates it over and over to create a sphere, illustrating the usefulness of a spherical metric.

What I Learned In constructing this project I have faced many challenges, overcoming many, while learning a lot. Firstly, becoming familiar with Matlab took some time. It quickly became clear that there are a lot of differences between designing GUIs in Java and designing them in Matlab using GUIDE. The handle structure associated with every object you create was novel at first, but proved to be useful and easy to organize down the line. On top of fully understanding GUIDE, becoming familiar with the various methods of plotting was crucial as I took a good deal of time just figuring out the best way to visually represent the metric as a plot. This required a lot of trial and error with the Schwarzschild metric, which didn't prove to be very useful; teaching me to take a step back and critically think about something before diving into trying to represent a metric tensor by testing it on different functions. Later on I got more practice with functions and sharing variables across functions in Matlab, culminating in being able to generate arrays with the trajectory in one function and use it in another to plot it in the main window as an animated line.

I also got to use the Symbolic Math Toolbox from Matlab to solve some equations involved in the initialization function. This toolbox let me declare symbolic variables which I could then differentiate with respect to certain variables and essentially solve calculus and algebra problems analytically instead of just numerically. Using it didn't take very long to figure out, and it proved to be extremely quick and much simpler than having to set up an algorithm to solve a similar type of problem. I ended up calculating components of the metric, the inverse metric, and the Christoffell symbols of the metric using the Symbolic Math Toolbox. These calculations led to plotting the particle from rest, as well as generating Flamm's paraboloid.

I also had to review a good portion of the Cosmology Workbook I used in a past class as there are many important concepts that I wanted to make sure I had right, as well as the fact that there are many good example problems and ideas for programs within it. I was able to base my method for path calculation off of the one outlined in the book. For propagation of wave packets, I needed to do a lot of my own research in order to figure out an efficient and doable method in which to calculate the time evolution of a wave packet. I could not use just any wave equation solver as I was trying to deal with a specific version of the KGE, but found that it would be beneficial to just use a tweaked version of the wave equation to simplify the calculation. From there I learned about different ways in which to solve a hyperbolic partial differential equation and found that I could use a finite difference method. There are many ways in which to implement a finite difference method, but I went with an implicit form which uses past solutions of the equation to approximate derivatives by central differences in both space and time. Using the algorithm, I implemented a program in Matlab which added to a two dimensional array that contained the solutions to the wave equation at each time step for every position. This way the evolution could be calculated and I can display Gaussian wave packet's center move as time goes on.

What would I have done differently ? If I could do this project over again there are a few things i would have implemented differently. Getting started on this project took a long time, wasting precious time I could have been using to make progress. Once I got into it I didn't mind working on it as much, but the initial motivation to get a decent portion done to begin with was difficult. For this reason I didn't get to work on wave packet propagation as much as I would have liked as I got too excited about different ways in which I could plot the geodesic trajectories. I also would have chosen to implement the program in Python instead of Matlab as Python is open source and has hundreds if not thousands of free downloadable content uploaded by other users. Many of the downloadable packages for Python come straight out of Matlab anyways, which would have made the transition more than bearable. Python seems to be the up and coming scientific programming language and I would like to be a part of that.

Problems Of the many stages in the process of completing this simulation, there were a few problems which had to be addressed in order to continue. A large problem I faced was figuring out how best to represent spacetime. I fooled around with the metric tensor, a diagonal metric which consists of the four components of the Schwarzschild metric in spherical coordinates. There are many functions for which take vectors and matrices and use the data in order to create plots. I tried using some of these functions, like streamline to plot a good visual representation of the metric, but many attempts didn't work or were not useful visually. Instead of taking this rather trial and error approach, I decided to look into the literature I had accrued on the subject and found a useful method in which to represent a plane of spacetime as a paraboloid. This visual allows the user to conceptualize how space and time both curve around a mass. In order to represent the spatial curvature, the book outlines a method in which holding the time component and phi constant, you can represent the simplified metric as a metric in cylindrical coordinates. Thus $z(r)$ describes the height of the plane at a corresponding r value. I then went further and used the same technique to show the curvature of time using the time component of the geodesic equation. I decided to include this in the paraboloid, called Flamm's Parabaloid, as the color of the plane. As the plane gets closer to $r = 0$, the color changes to a darker and darker blue, while farther away it turns red and then yellow. I have this be the first thing that pops up when the program runs as I think it does a good job of illustrating the the meaning of gravity.

I also had to overcome a problem when figuring out how to simulate the wavepacket propagation. Solving the second order wave equation involves solving second order partial derivatives. To do this I had to look at my options. There is a pde solver in Matlab, but of course, it is another toolbox which has to be bought. On top of that, it seemed to only apply to specific situations, and required fields that were not necessary for my calculation. Upon further research I found that an implicit finite difference method should work best. I implemented a method in which they derive a finite difference formula from the wave equation and then represent it as an equation of matrices using a tridiagonal matrix and a column vector. After implementing this in a Matlab program I was able to create an oscillating standing wave, but couldn't get it to propagate. Later I tried a centered difference method which used a formula that allowed for the calculation of the wavepacket u_n^{k+1}. After setting initial conditions I could then propagate the wave into the future.

More Time If I had another month to work on this project I would finish the wave packet propagation that I didn't get to complete in its entirety. As of now I have a one dimensional representation of wave packet propagation in Schwarzschild spacetime, but I wanted to be able to show two dimensional propagation, if not three dimensional, which wouldn't be much harder as the orbits are planar. Judging by the fact that getting one dimensional wave packet propagation was difficult to simulate, two dimensional propagation would have its own hang-ups, but I think it could be done within a month. Furthermore, if I had stuck with the original KGE for curved space, I would have to derive my own finite difference algorithm which could take a lot longer, unless I figured out a more efficient way of tackling the problem.

Advice To all the students who are planning on starting thesis soon, or in the next coming years, start thinking about what you want to do now. It is not too early to start thinking about what project you what to accomplish. It is crucial that you start working on ideas now so that you can have an idea of what you like doing. A concept that seems great in your head might not be very fun to actually sit down and program for a few months. Just as important is creating a plan that you take the time to critically think about; it will help you in the future. Spending 10 minutes coming up with a plan is easy, but its not going to help you very much and you won't even take it seriously if it doesn't mean that much to you.

Bibliography

[1] Brandt, Siegmund, Hans Dieter Dahmen, and Tilo Stroh. [*Interactive Quantum Mechanics*]. New York: Springer, 2003. Print.

[2] Bridson, Robert. [*Computational Physics in Film*]. Science 330.6012 (2010): 1756-757. Web. 5 Nov. 2016.

[3] Bryson, Steve [*Virtual Spacetime: An Environment for the Visualization of Curved Spacetimes via Geodesic Flows*]
`https://www.nas.nasa.gov/assets/pdf/techreports/1992/rnr-92-009.pdf`

[4] Giordano, Nicholas J., and Hisao Nakanishi. [*Computational Physics*]. Upper Saddle River, NJ: Pearson/Prentice Hall, 2006. Print.

[5] Moore, Thomas A. [*A General Relativity Workbook*]. Mill Valley, CA: U Science, 2013. Print.

[6] R.Nave [*Quantum Physics*]
`http://hyperphysics.phy-astr.gsu.edu/hbase/quacon.html#quacon`

[7] Peirce, Anthony
`"Solving Partial Differential Equations Using Finite Difference Method."`
`Algorithms and Parallel Computing (2011): 323-29. Web.`