## Coltin James Lux

## Computer Science Capstone

## 1 December 2016

Tsunami Wave Modeling 101:

Problem:

Tsunamis (seismic sea waves) are characterized by very large, tall, and powerful waves that are created by an underwater disturbance creating a large amount of energy. An underwater disturbance such as an earthquake, landslide, volcanic eruption, or meteorite can release very large amounts of energy underwater. Tsunamis can be known to travel hundreds of miles per hour and get as tall as a hundred feet or more. Typical tsunami waves will travel in all directions from the occurrence of the underwater disturbance. However, when the wave approaches shore, when the ocean floor rises in height, the wave builds in height due to the amount of energy.

All tsunamis can be dangerous. There have been several destructive tsunamis throughout history and the more we can prepare for such natural disasters the better off and safer we will be. That is, the more tools we have for modeling the behavior of these waves, the more we can prepare for the waves crashing into shore. Thus, in this written work we will be numerically approximating and visualizing tsunami wave behavior using computer software.

To model tsunami waves, we will be using a variation of finite difference methods for two commonly used partial differential equations commonly seen in shallow water wave theory for modeling tsunami waves. That is, we will be numerically approximating solutions for these models. We will be determining the correctness of our models based on velocity and the actual velocity solution given by a function or we will be testing using numerical convergence if there is no actual known solution. We will need to find a solution that preserves momentum and energy over time. Thus, our model must preserve the shape of the wave over time.

In this project, we will be utilizing our knowledge of Numerical Analysis, Calculus, Differential Equations, and Linear Algebra to arrive at the required equations.

For this written work however, we will be focusing of the software that was developed to numerically calculate and visualize these waves. We will be utilizing computer software in the programming language, python.

Approach:

Let us first simplify the problem. We will not be modeling 3D waves, but instead 2D waves. In several applications of shallow water wave theory, we see the utilization of the KDV Equation, Camassa-Holm (CH) Equation, and the Two Component Camassa-Holm (2CH) Equation. To approximate these solutions we will be using a variety of methods for discretizing space and time, including finite difference methods and Runge-Kutta methods. We will be using initial conditions such as functions that are known, plus using collected wave data.

In this work we will not explicitly be discussing the equations, how we arrived at them, or why we used them, but instead how the software was developed to utilize such equations. To develop software that would be able to do such computations, we must first think about any existing APIs that could aid in doing this type of mathematics.

Let us first go over the type of functions and equations that we will be using, and then we can analyze which APIs/tools are available for such computations.

For the equations that were given, we first know that we will be handling data (specifically quantitative data). All the data will consist of vectors. There are also functions such as:

Thus, we will need access to the hyperbolic secant function among others. The mathematics also includes finite difference methods, so we know that we will have to do some type of vector multiplication or vector addition using the data in the vectors. We will also be using matrices and matrix multiplication, along with needing to calculate the inverse of a matrix. On top of all the mathematics, we will need to plot and show the graphs of the data.

After sifting through all the computations that we will be working with, the API that I thought would be the best resource was NumPy/SciPy/MatPlotLib in the language python.

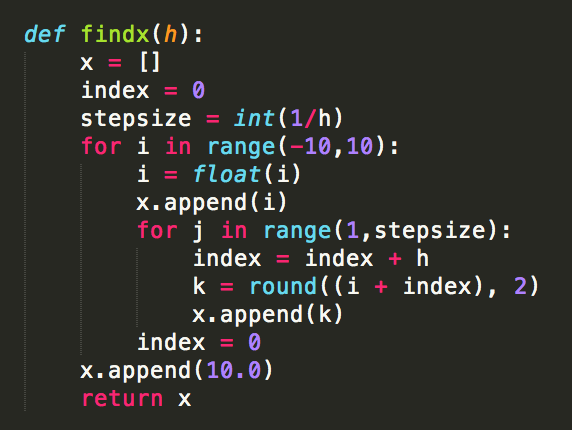
Thus, we will be using the NumPy/SciPy/MatPlotLib libraries and developing this software in python.

Program:

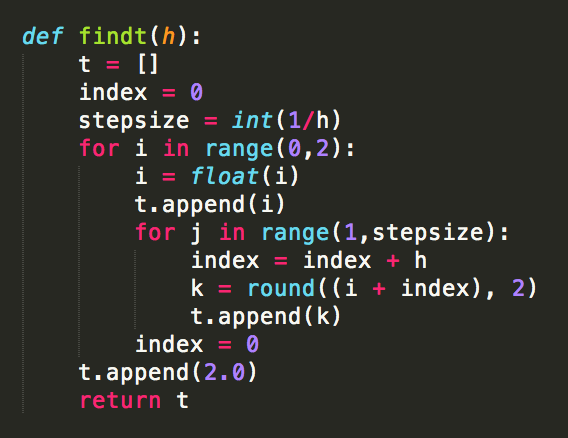
For this project we worked with three different equations, four different approaches, and five different data sets. Thus, altogether we have five different programs. The first program uses the KDV Equation, along with a Finite Difference Method for spatial discretization and the Forward Euler Method for time discretization. The second program uses the KDV Equation, along with a Finite Difference Method for spatial discretization and the third order Runge-Kutta Method for time discretization. The third program uses the Camassa-Holm Equation, along with a Finite Difference Method for spatial discretization and the third order Runge-Kutta Method for time discretization. The fourth program uses the Two Component Camassa-Holm Equation, along with a Finite Difference Method for spatial discretization and the third order Runge-Kutta Method for time discretization. The fifth program uses the Two Component Camassa-Holm Equation, along with a Finite Difference Method for spatial discretization and the third order Runge-Kutta Method for time discretization, *and* tests using real tsunami data.

Let us start with the first program: MathCapstoneFinalKDV.py. Let us first list all the functions that we will be included.

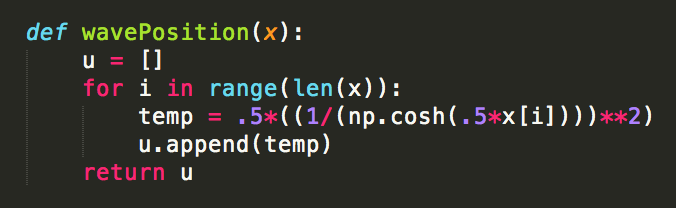
1. Function definition to find the spatial interval (need to create a list to store all the x-values that will be tested, h (step size) as a parameter so that we know which numerical values to put in the list).



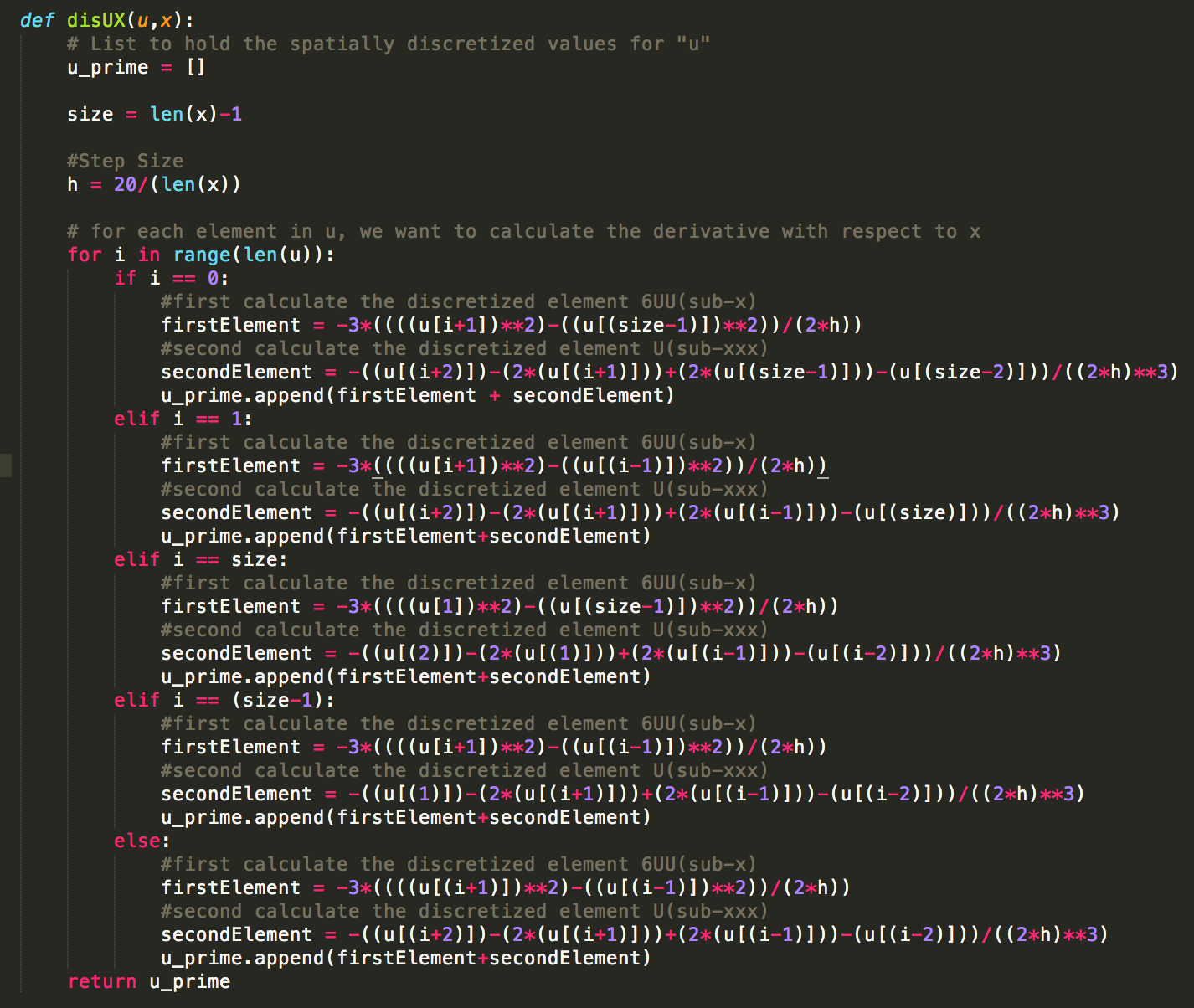
1. Function to find the time interval (need to create a list to store all the t-values that will be tested, h (step size) as a parameter so that we know which numerical values to put in the list).



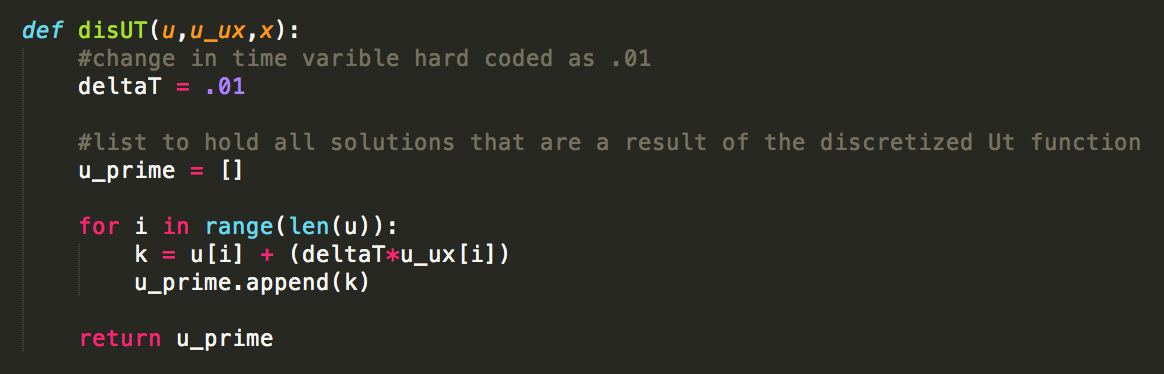
1. Function to find the initial position for velocity (need to use list “x” and equation to find initial “u” position at each location in space)



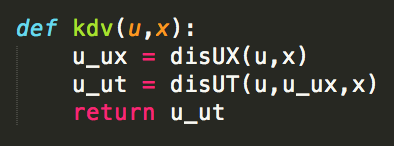
1. Function to use discretized spatial equation (need to use list “u” and equation to find the spatially discretized “u”)



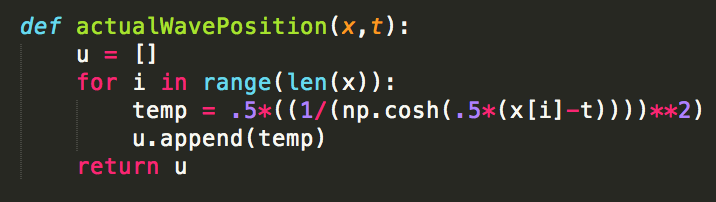
1. Function to use discretized equation for time (need to use spatially discretized list “u” and equation to find the discretized “u” in time, which is “u” for the next time interval)



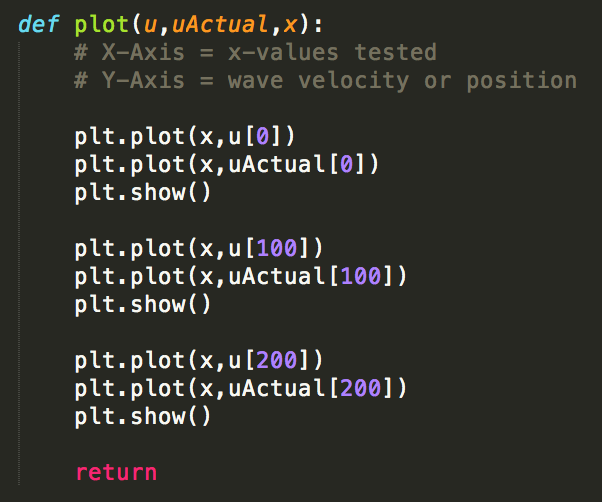
1. Function to call the discretized spatial function and discretized equation for time function. (This function really could have been left out or avoided, but for organizational purposes and clarity I kept it).



1. Function to find the actual position for velocity (need to use list “x” and equation to find “u” position at each location in space for all values in our time interval). In the Main function we will pass the specific value for time in our time interval that we are at.



1. Function to plot the data (use mathplotlib.pyplot to graph). Here, we will have our calculated “u” values and the actual “u” values and will be plotting them together so that we can do comparison and analysis.



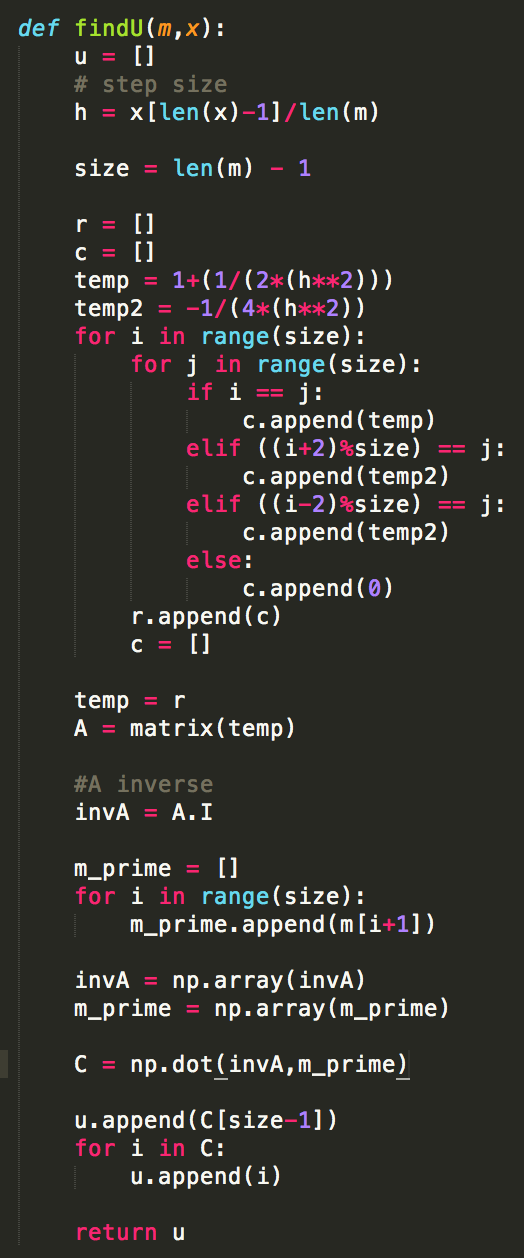
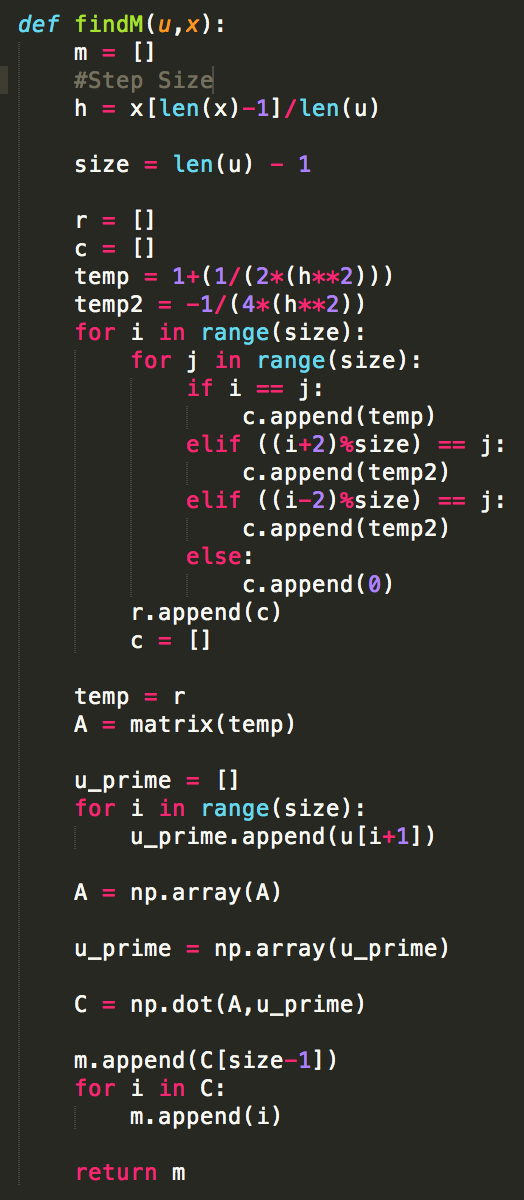
1. Main Function to call all other functions and for organization.



The above program is the most simple of the five that were developed. However, the basic structure is the same for all five programs. The only differences are in the equations that are used, and the addition of a required function that utilized linear algebra. Let us now examine those functions.

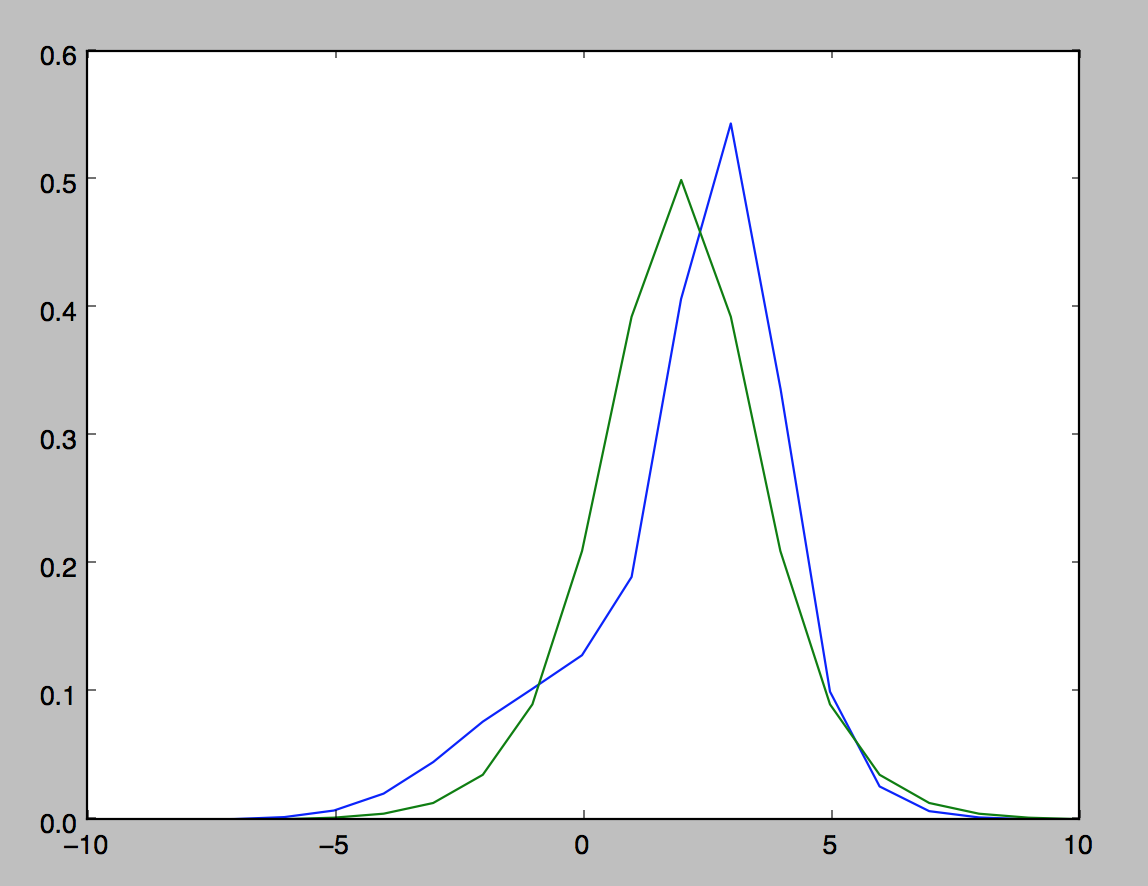
First is the find M function. The mathematics requires us to find a vector from a vector that we currently have. To do so, we will need to use matrices and matrix multiplication to do so. Essentially the mathematics needs us to set up the following type of equation:

Where “m” and “u” are vectors and A is a matrix. To find m, we use matrix multiplication and to find u, we must calculate the inverse A to do that matrix multiplication.

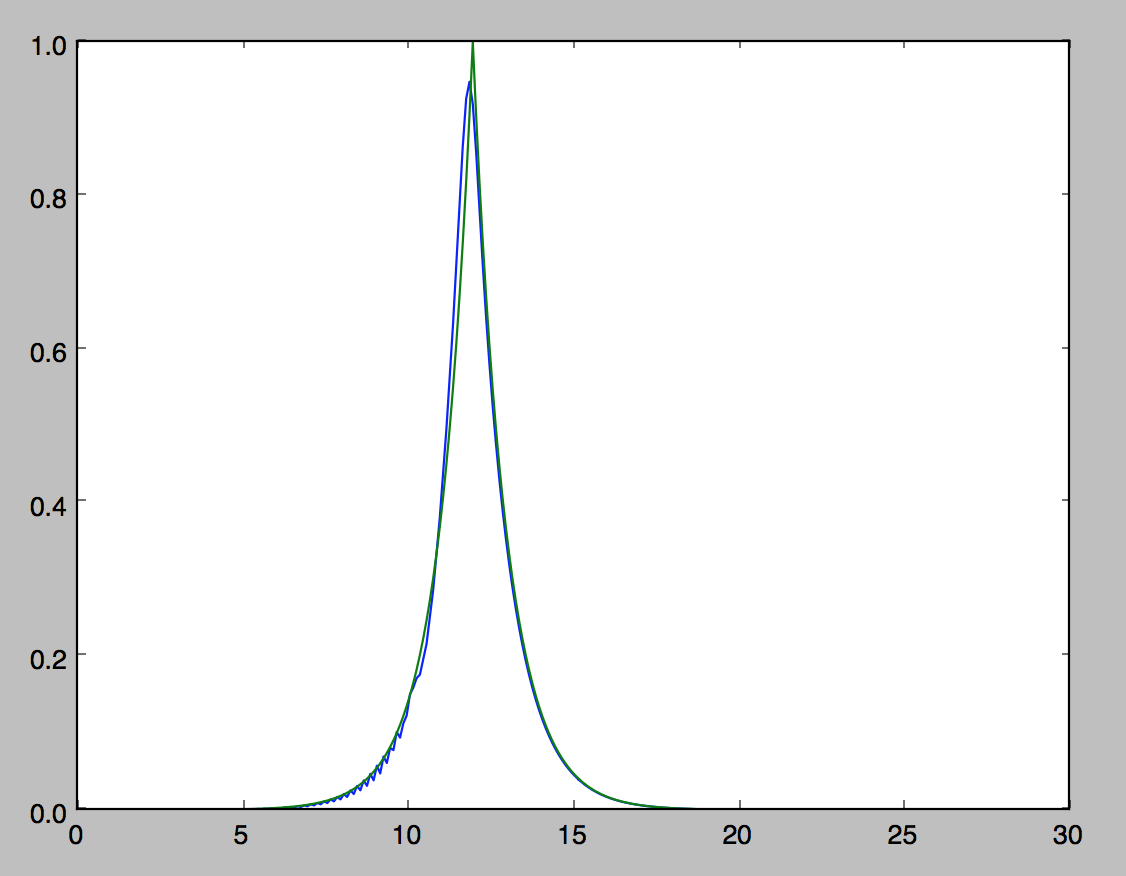


Results:

Now that we have the basic structure of all five algorithms, let us discuss the results! After running each algorithm, the result is a graph. This graph enables us to be able to observe the stability of the mathematical equations that we used. For example, the result of the first program (MathCapstoneFinalKDV.py) is not stable:



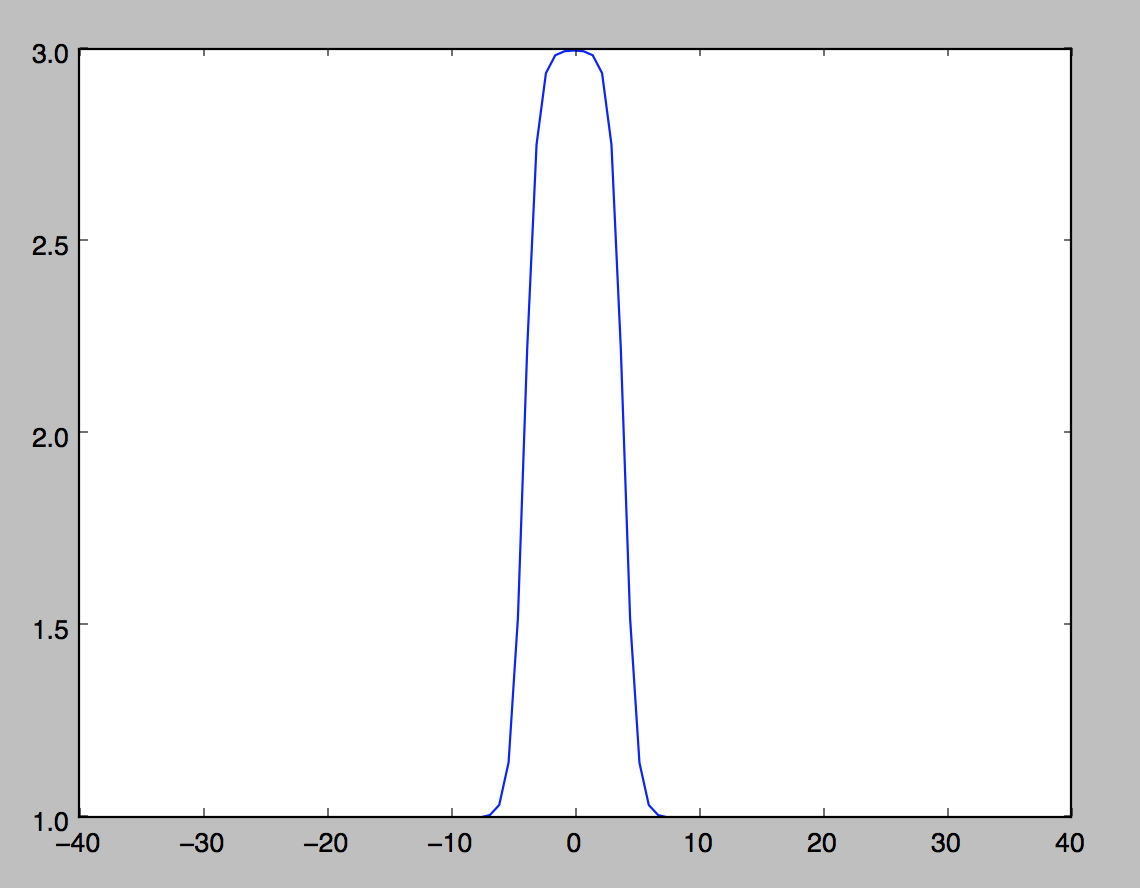
The blue is our calculated velocity at time 2 and the green is the actual solution at time 2. However, if we observe one of the ladder programs we can see that the equation get much more stable and should be preferred over the former. Let us observe the results of the third algorithm (MathCapstoneFinalCH.py).



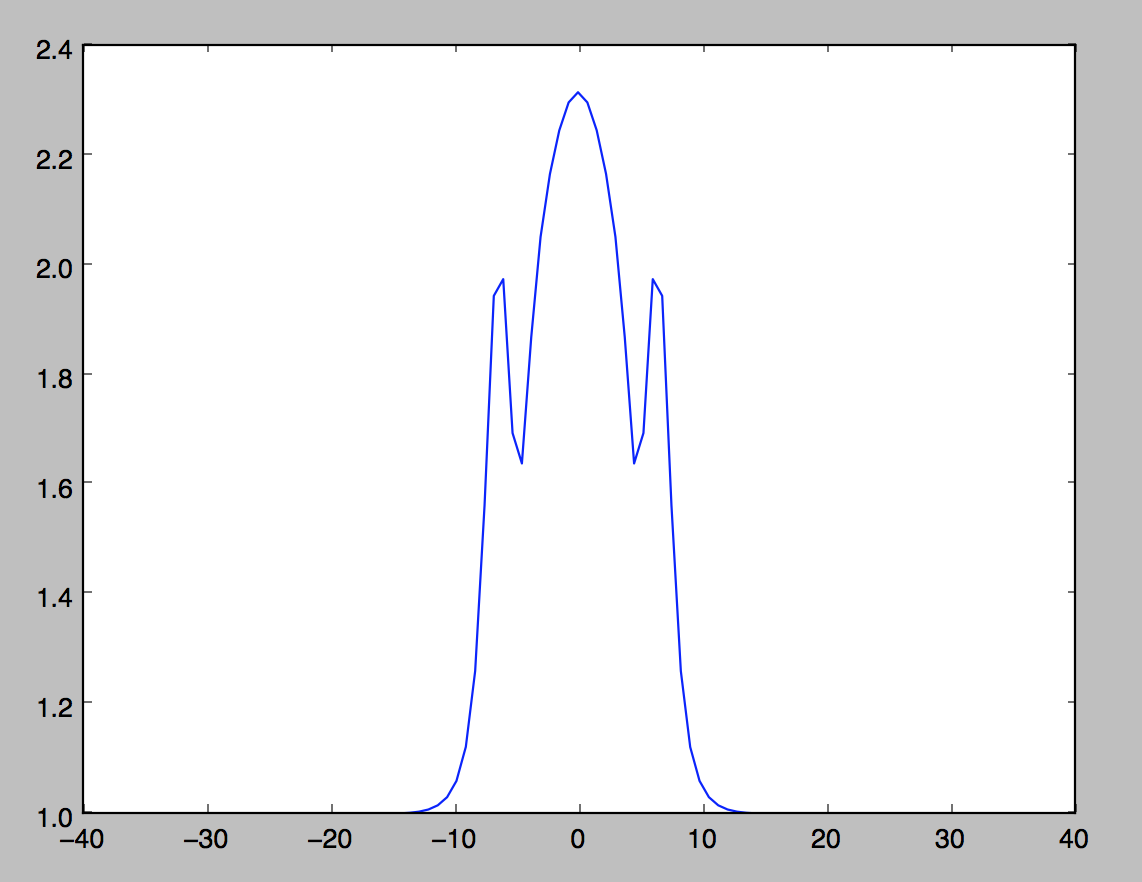
The fourth and fifth algorithms do not have an actual known solution to compare our solution too, so we have to use numerical convergence to test for accuracy. Let us observe the results from the fourth algorithm (MathCapstoneFinal.py):

*100 Points:*

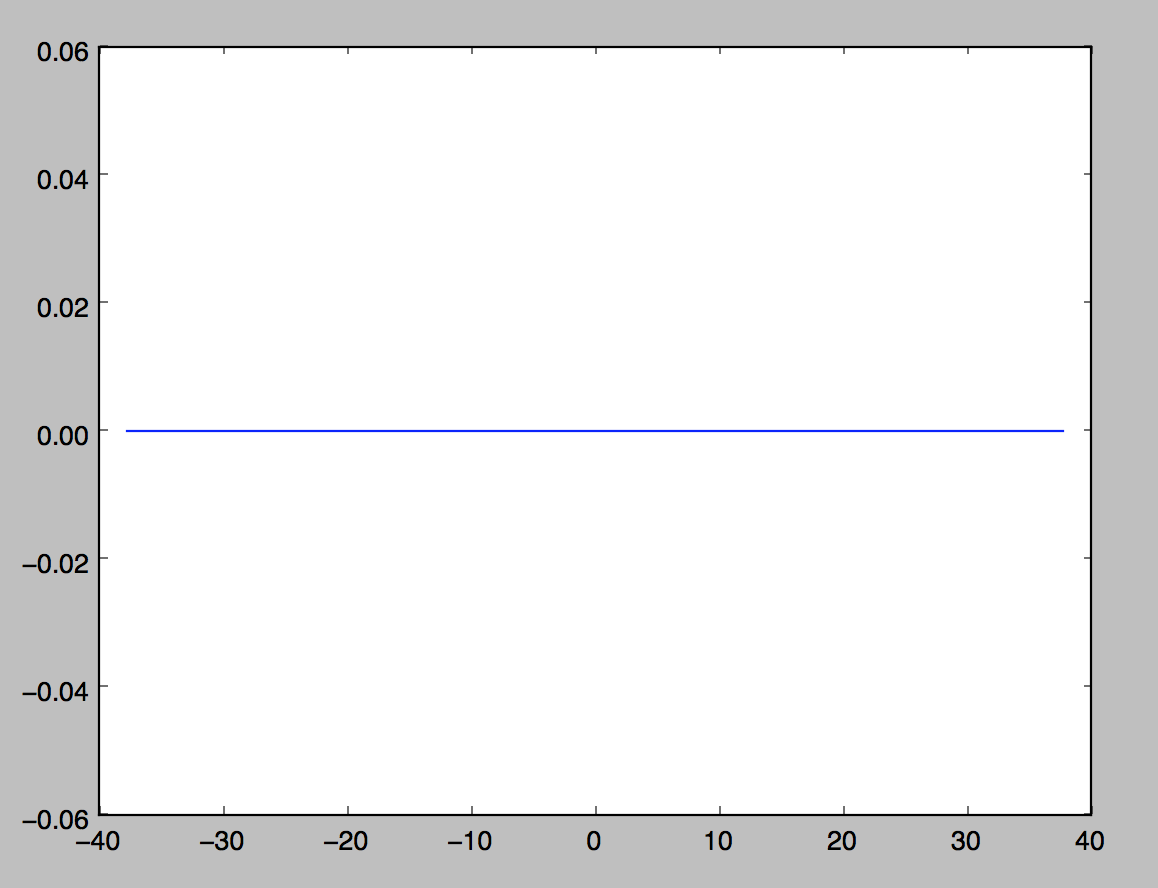
(Initial Condition) Density at Time = 0:



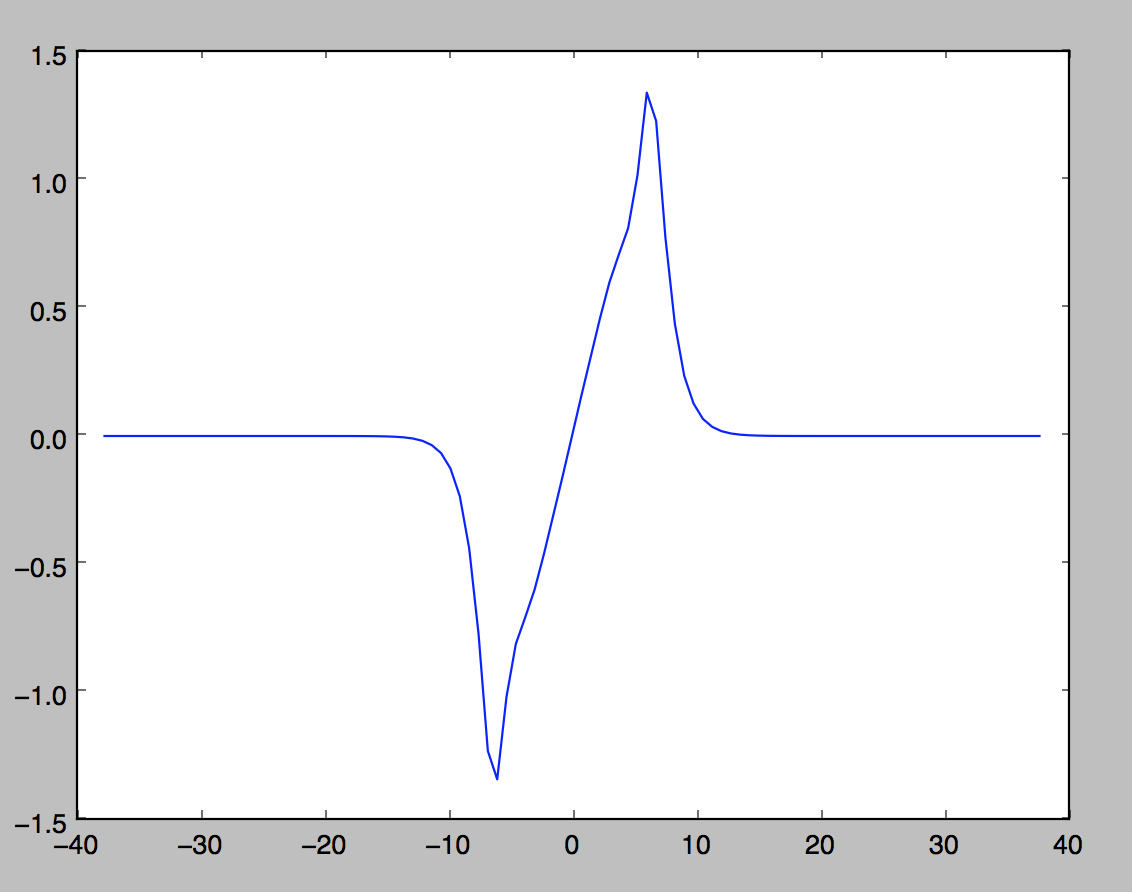
Density at Time = 2:



(Initial Condition) Velocity at Time = 0:

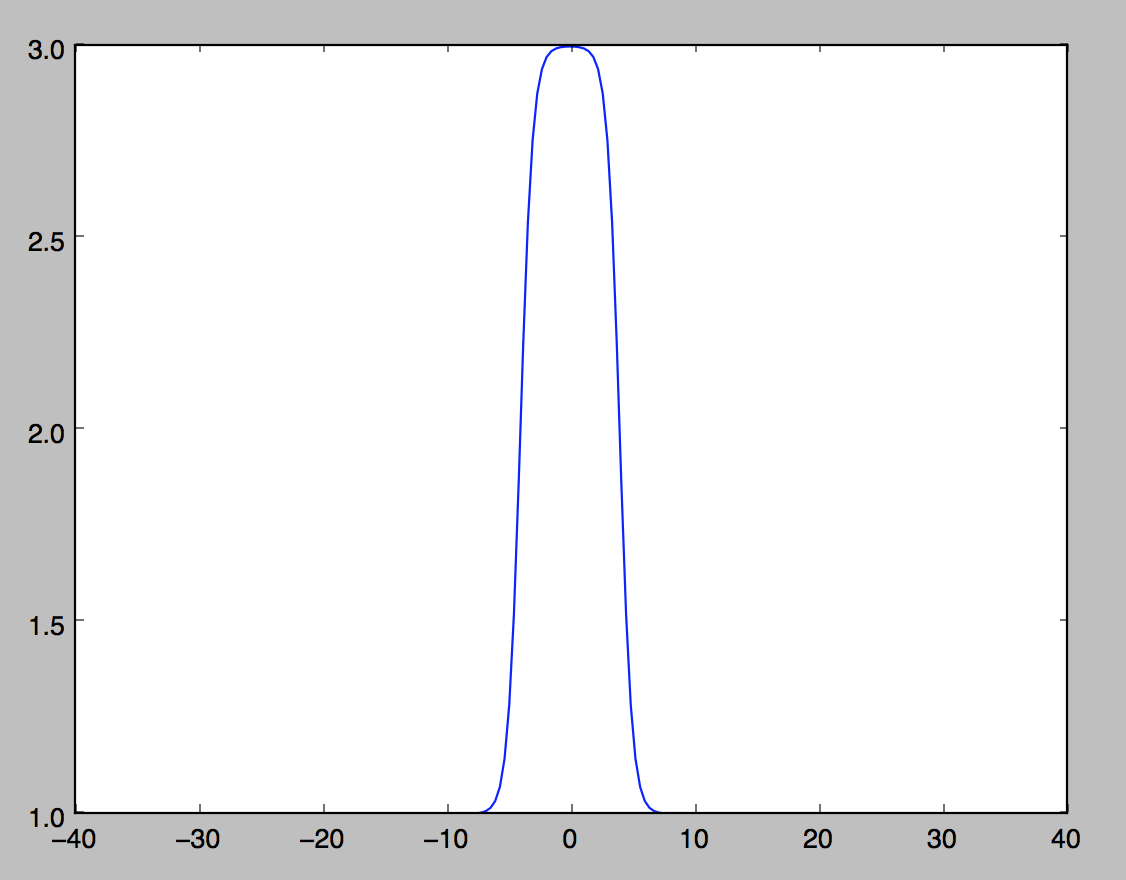


Velocity at Time = 2:

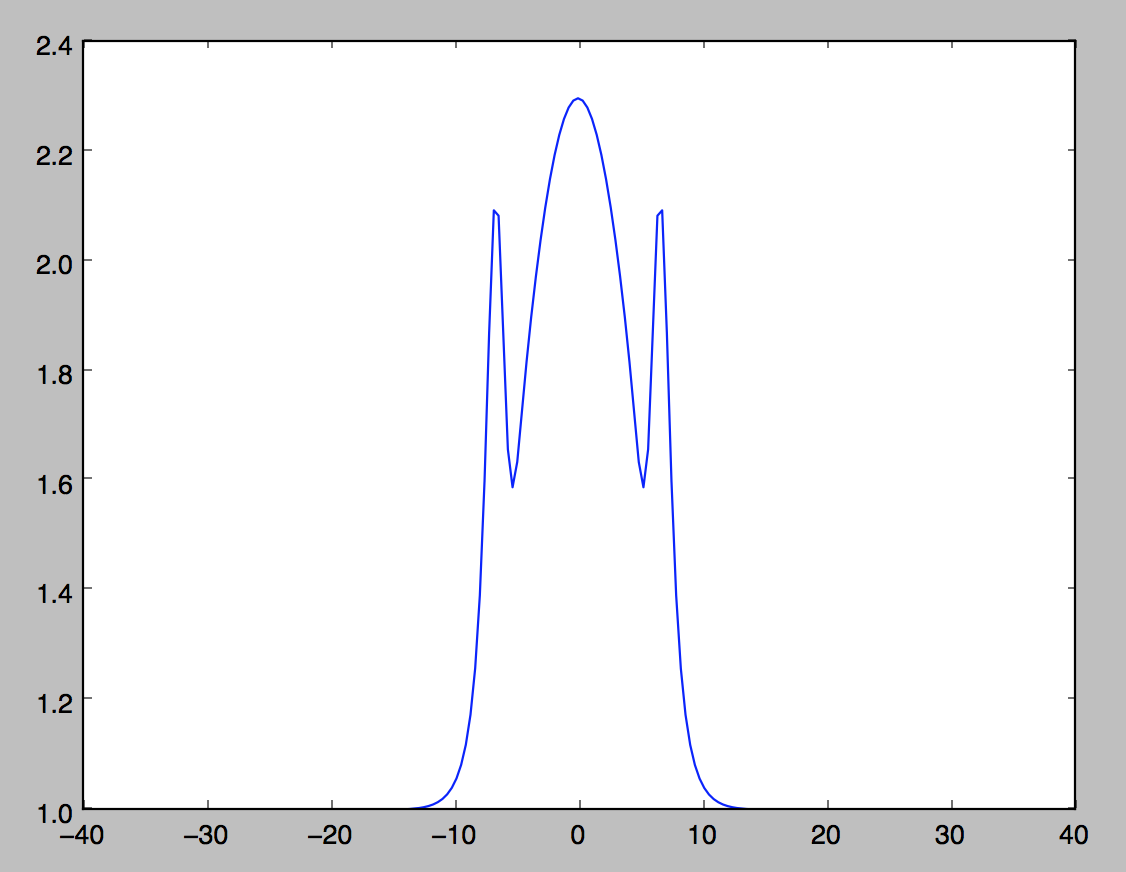


*200 Points:*

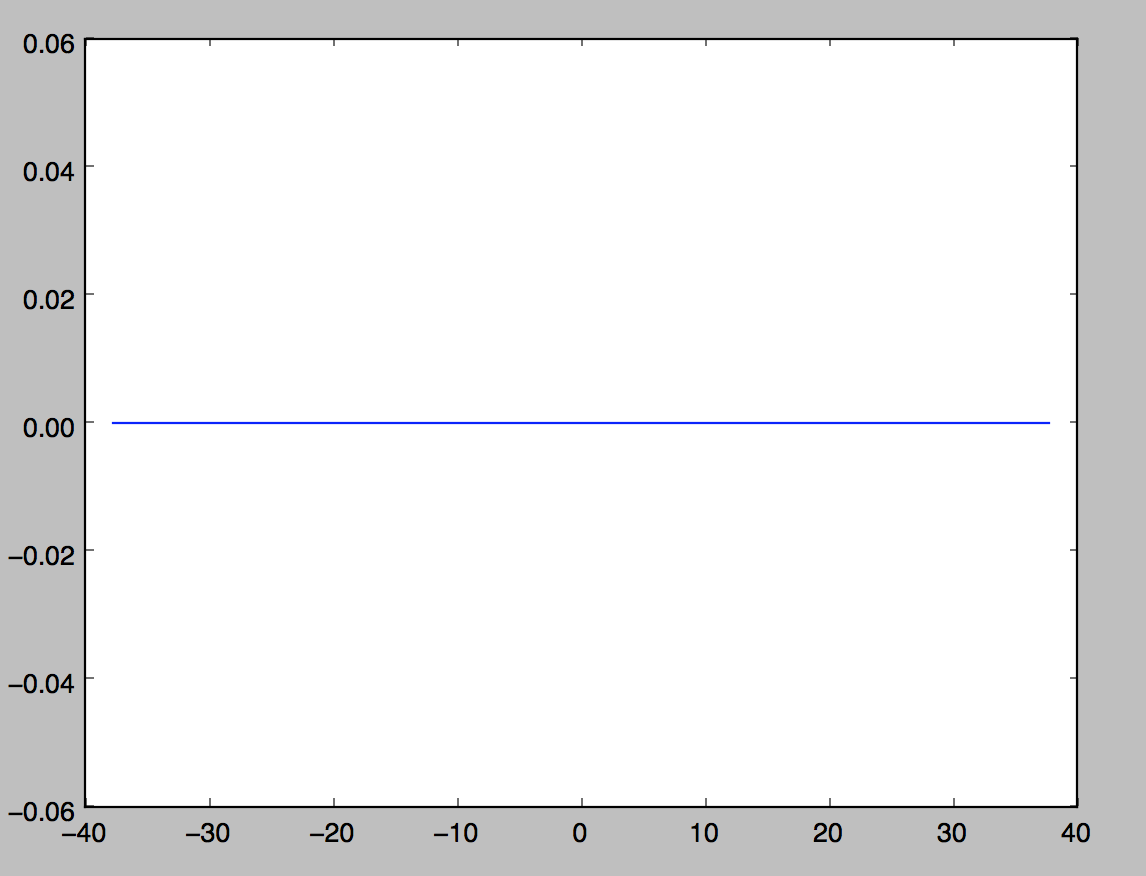
(Initial Condition) Density at Time = 0:



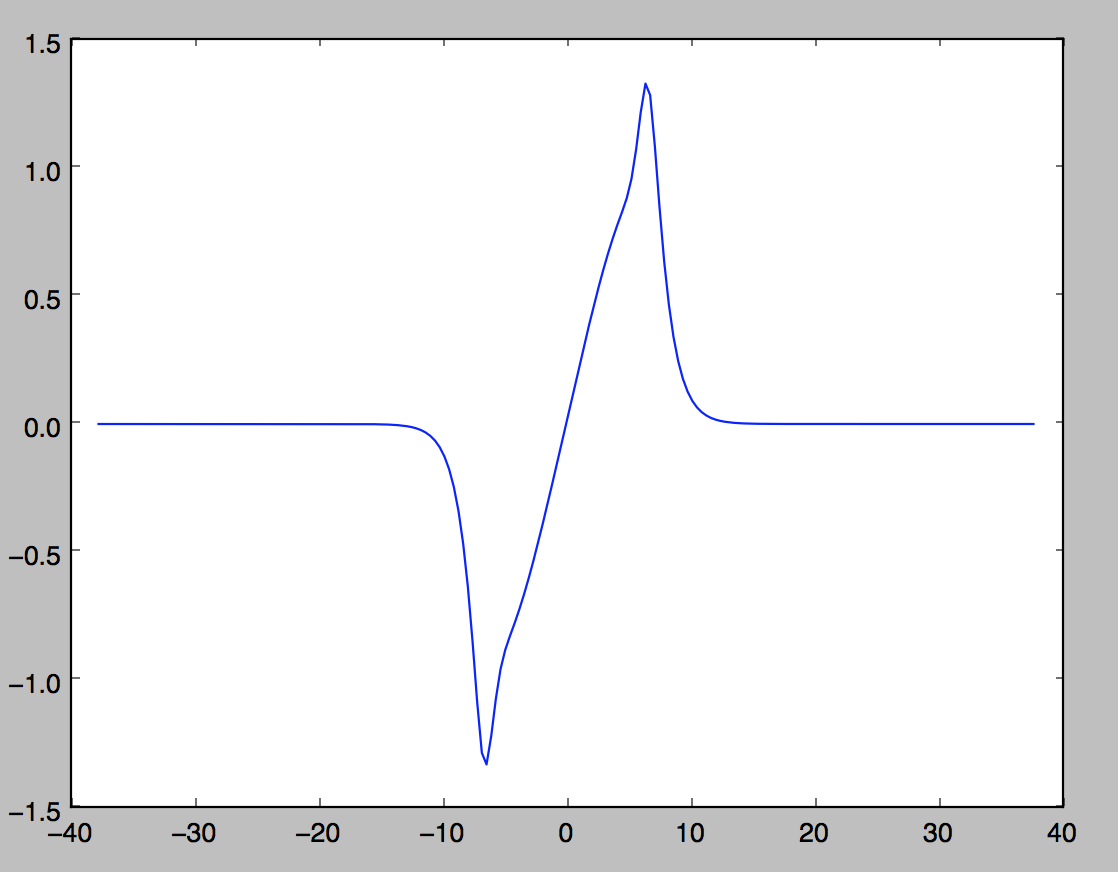
Density at Time = 2:



(Initial Condition) Velocity at Time = 0:

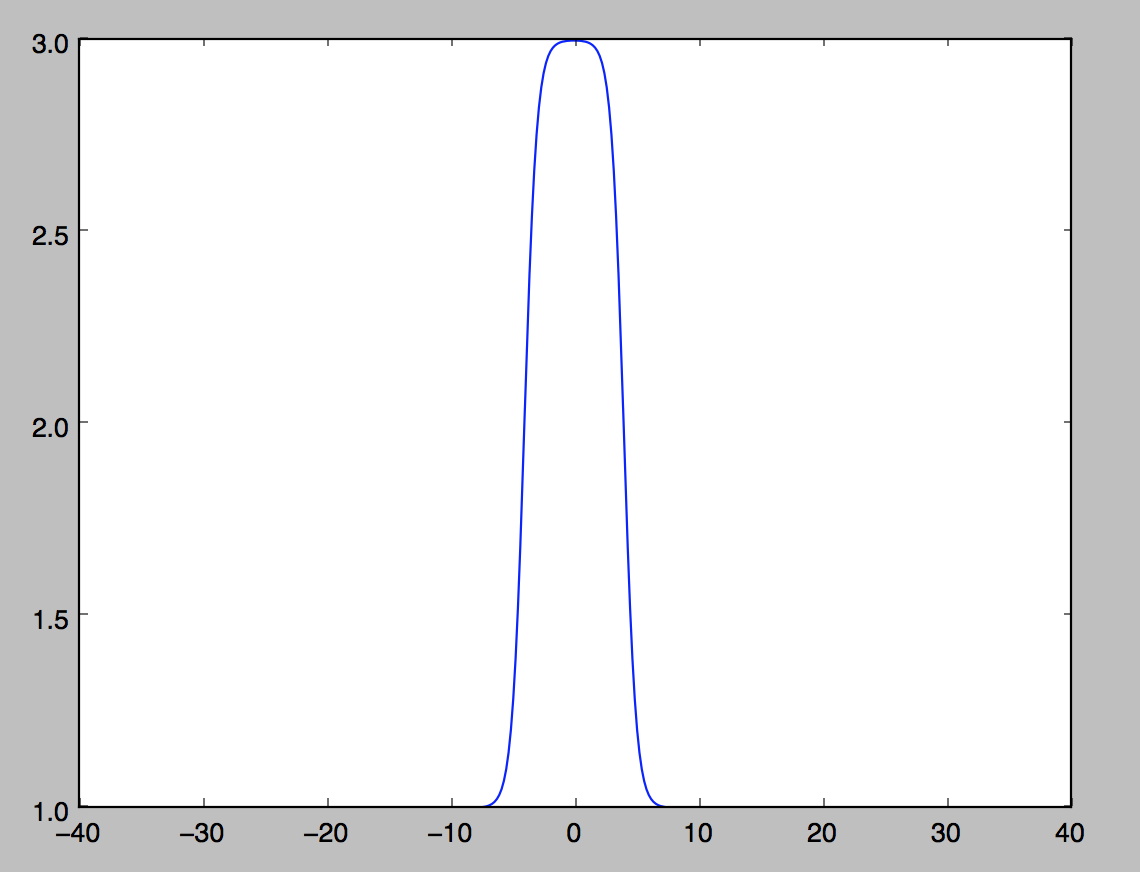


Velocity at Time = 2:

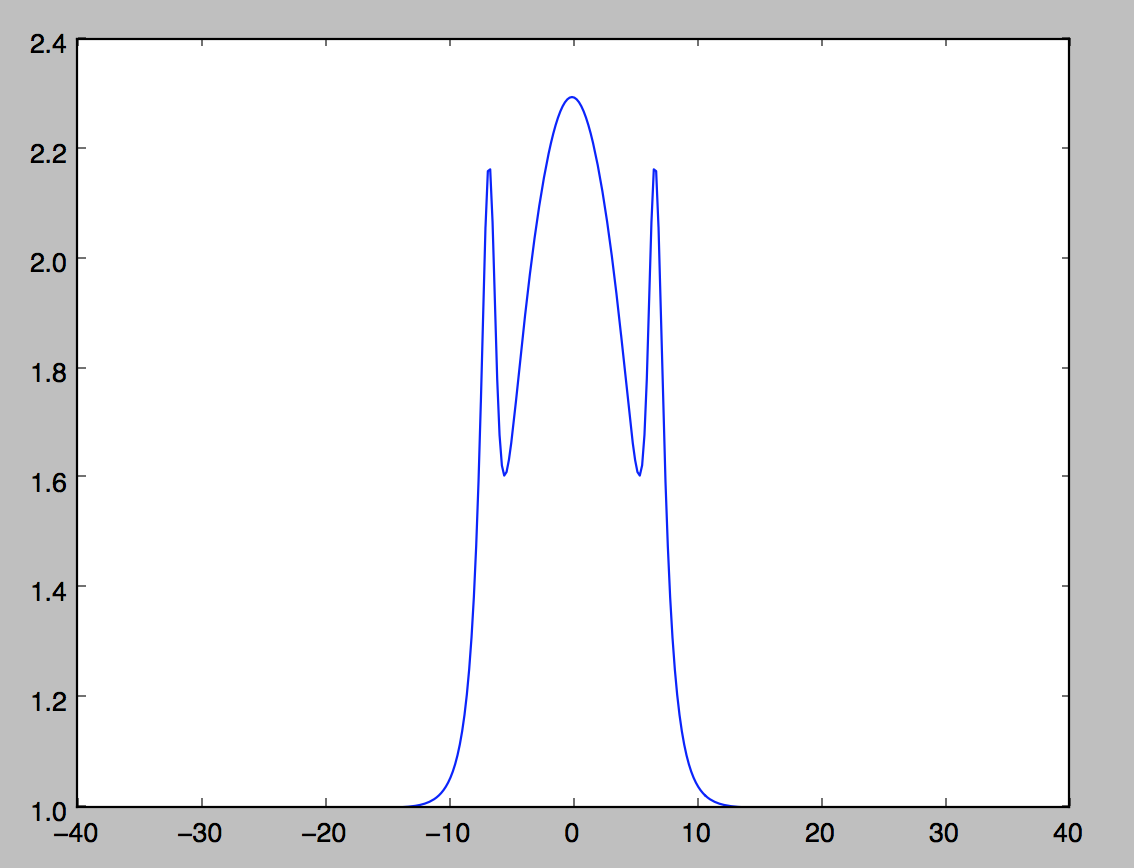


*400 Points:*

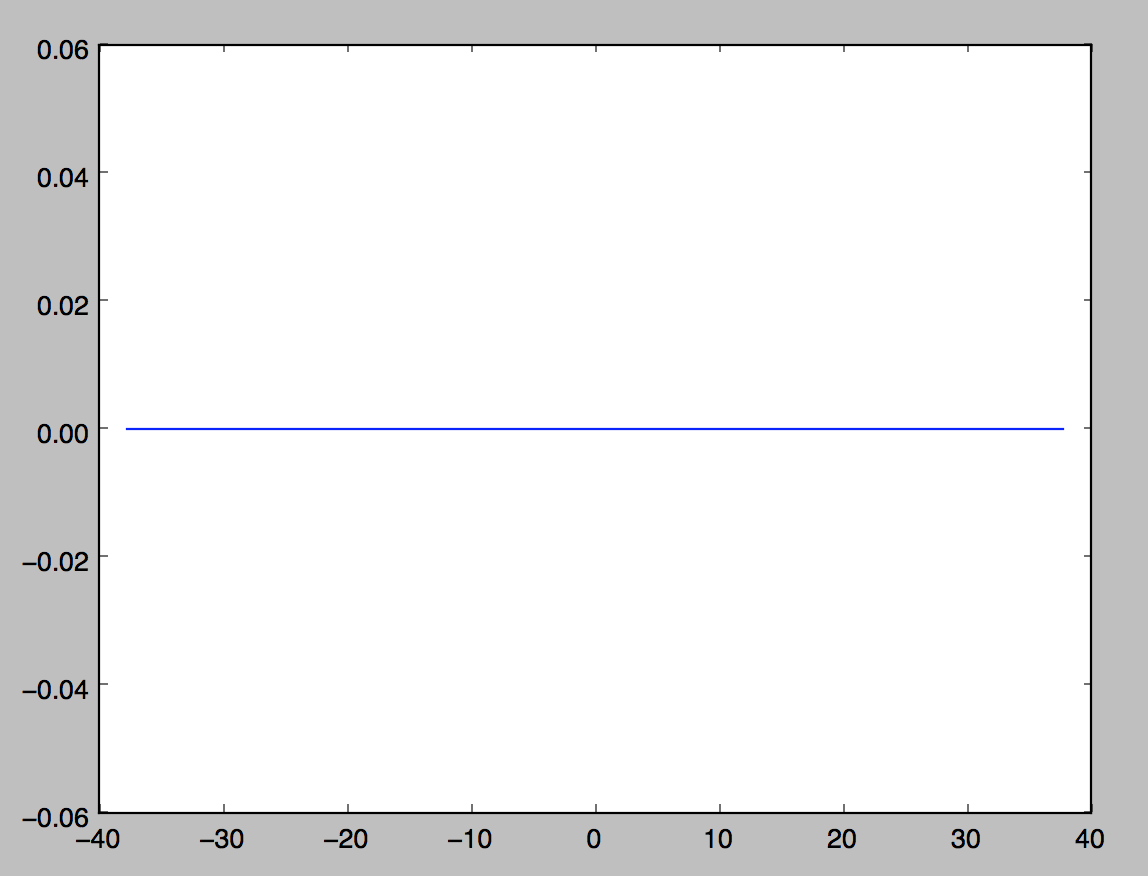
(Initial Condition) Density at Time = 0:



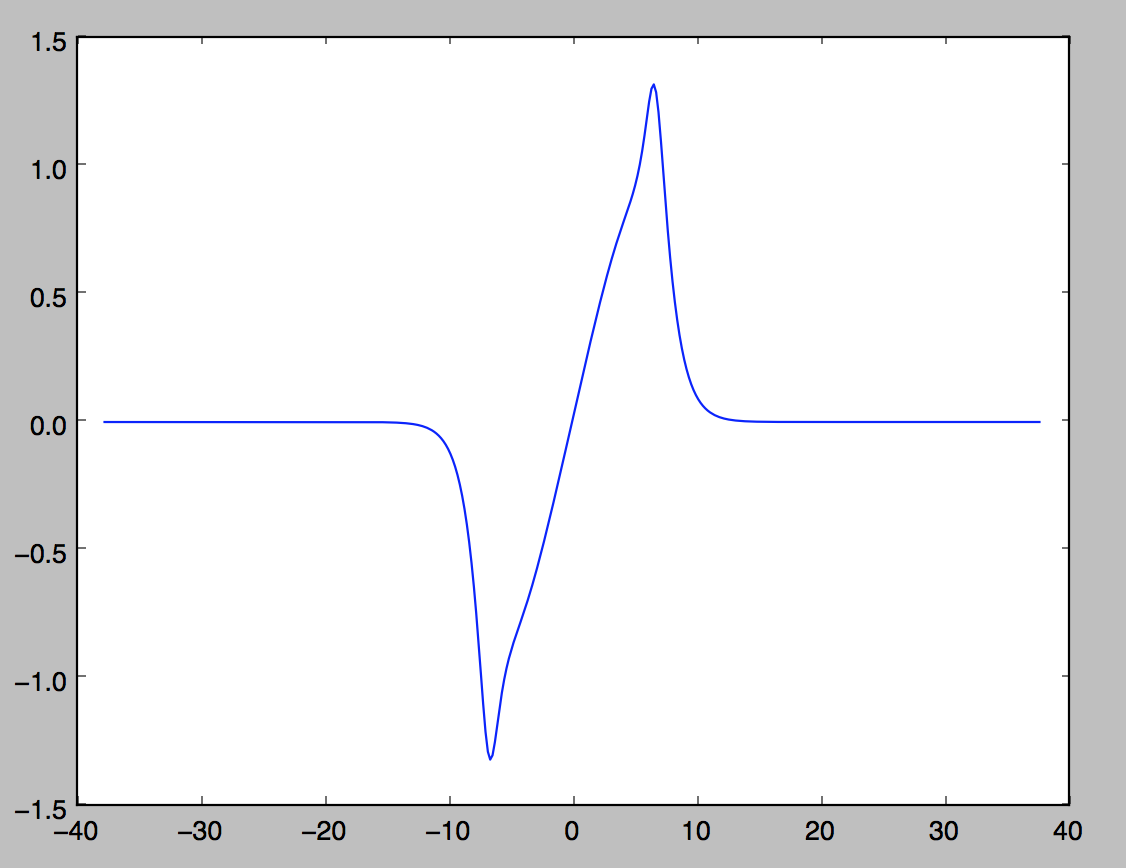
Density at Time = 2:



(Initial Condition) Velocity at Time = 0:

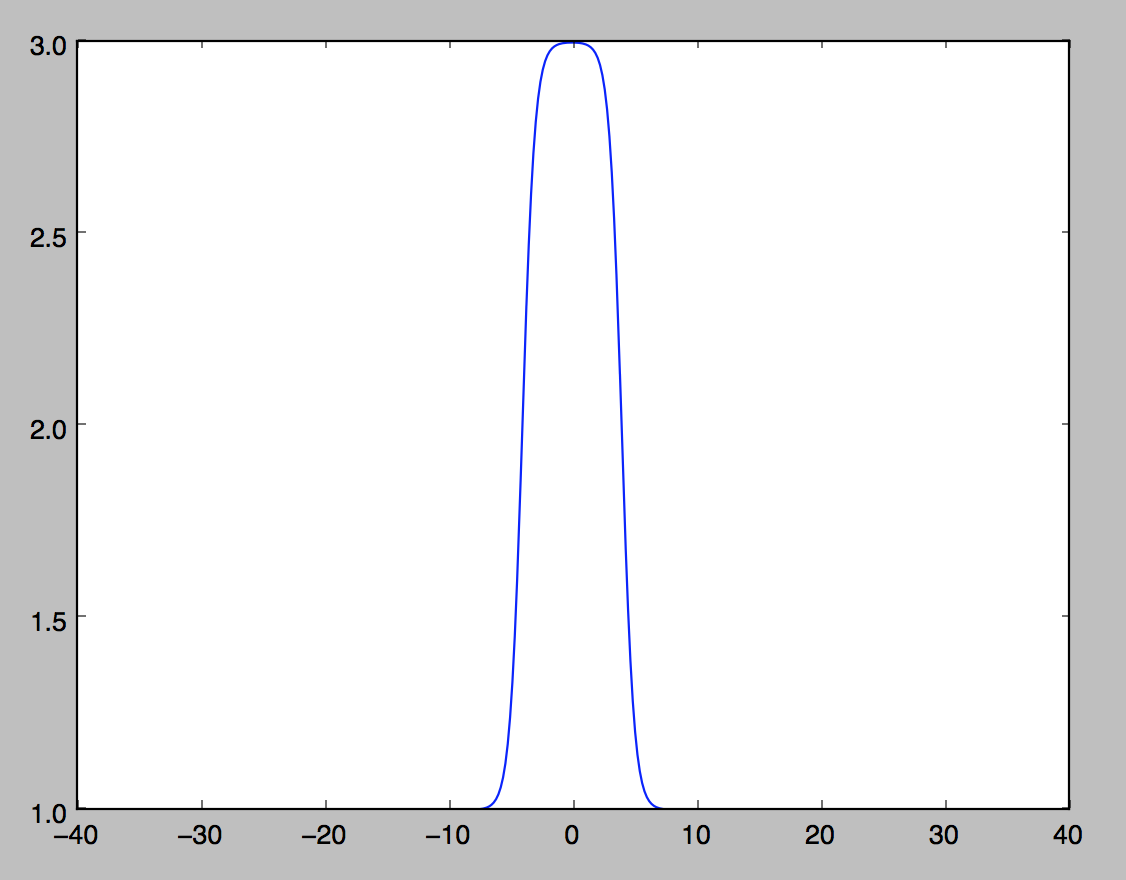


Velocity at Time = 2:

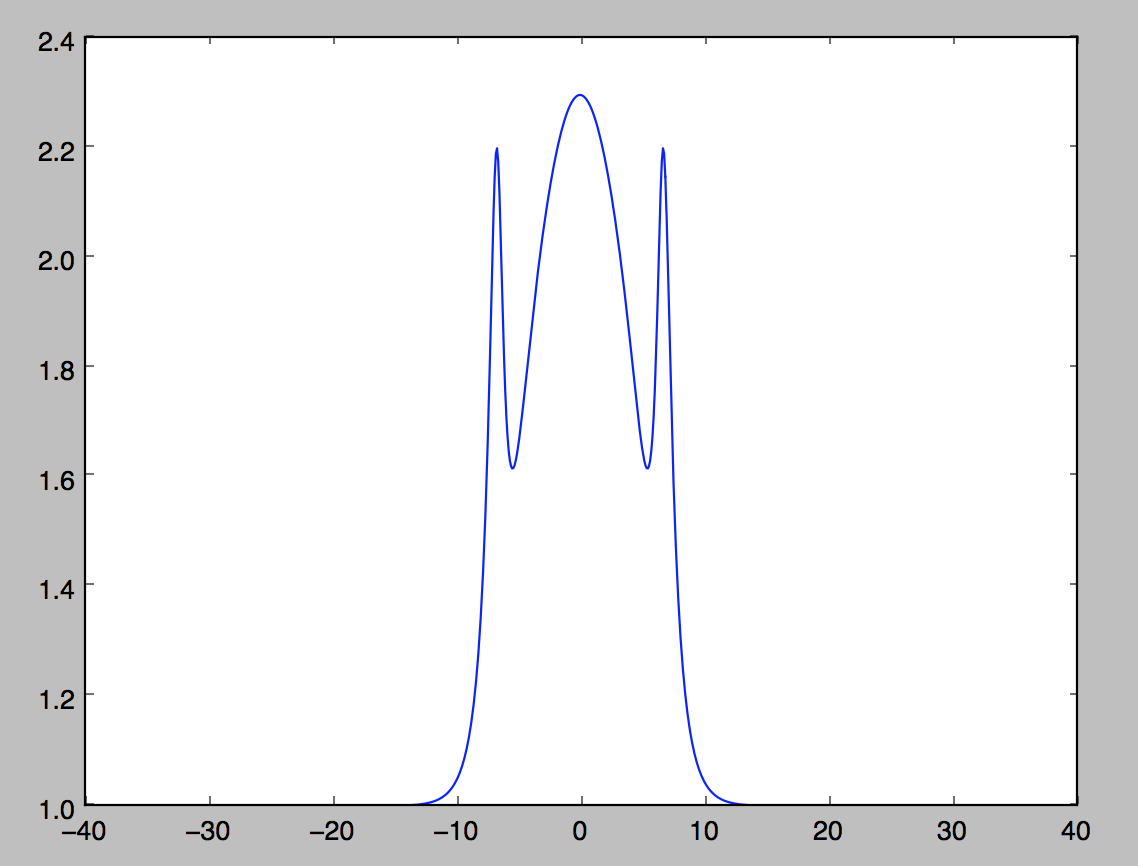


*800 Points:*

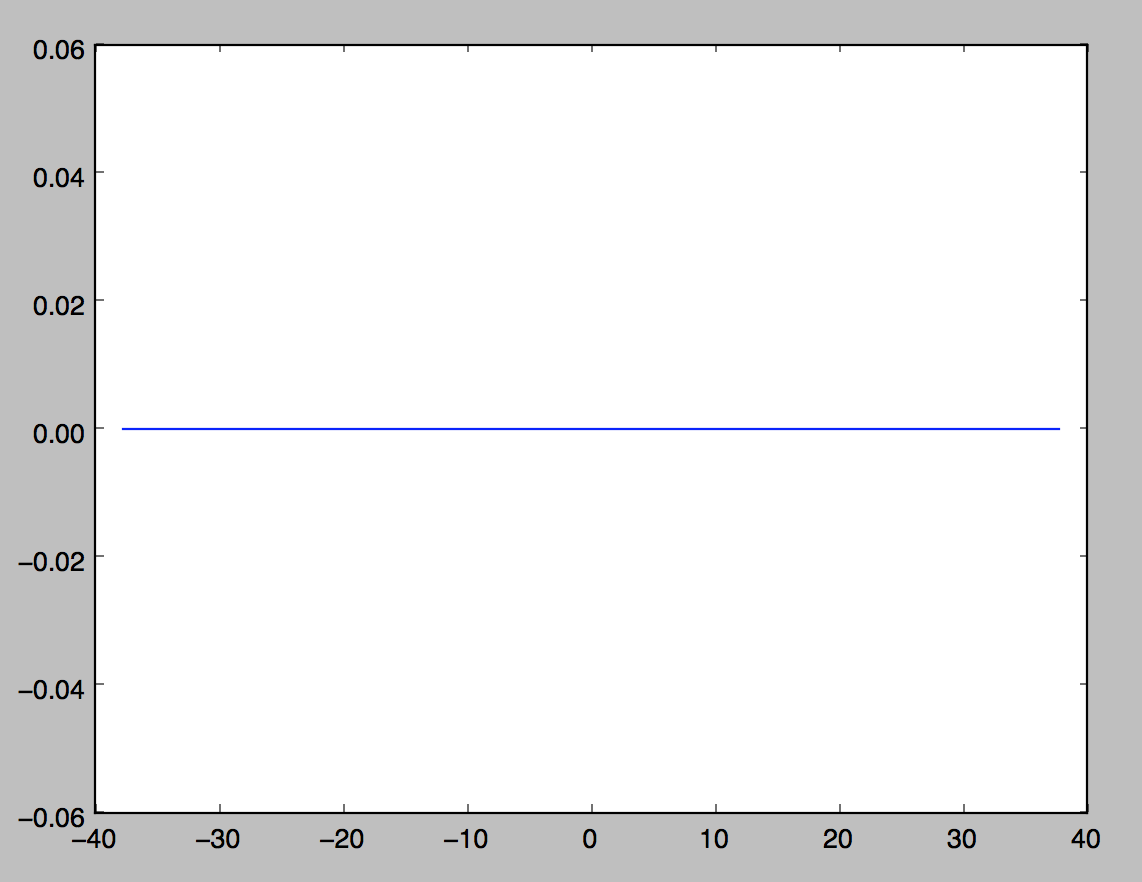
(Initial Condition) Density at Time = 0:



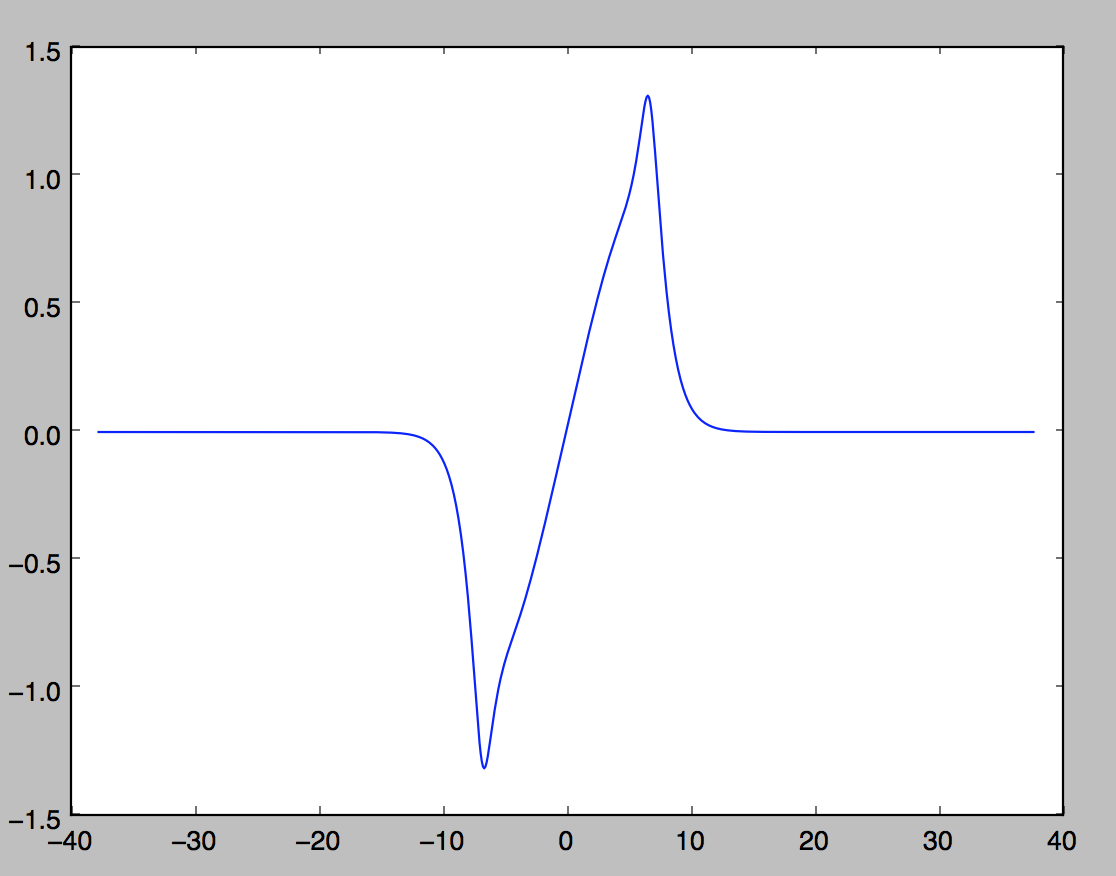
Density at Time = 2:



(Initial Condition) Velocity at Time = 0:



Velocity at Time = 2:



What we can see here is that the solution converges on two specific wave shapes (one shape for density and one for velocity). This is good because we know that the equation *and* the algorithm are stable.

These results are very helpful in visualizing the stability of these equations. If I were to have more time, I would utilize pyplot more. That is, I would make the graphs easier to read and altogether better visualizations. For the purposes of my project however, they did the bare minimum and got the job done. If I were creating this algorithm for a stand-alone purpose, there isn’t much I would change except the visualization aspect.

In order to complete this project, I needed to combine my knowledge of python as a software language, numerical analysis, data structures, linear algebra, calculus, and differential equations. I needed to know how to store the data, access the data in the lists. I needed to use knowledge of regular expressions (in the last algorithm). Finally, I need to combine all these aspects of computer programming in order to develop a program that would numerically solve and visualize these waves and the stability of these waves.

Finally, let us discuss the scalability of this program. The speed of this algorithm is directly correlated with how many data points we are testing. Since we are doing several computations with very intricate equations, the algorithms can get slow very quickly. The most time consuming mathematics that we must compute is the linear algebra and solving the system of equations, which has a time complexity of . Thus, this algorithm would not be scalable to thousands of data points without waiting a significant amount of time before the algorithm delivers the results.

In conclusion, the main purpose of this algorithm was to numerically solve and visualize the stability of several common equations used in shallow water wave theory and it has accomplished that purpose. This project has pushed me in several different ways, but it really has opened by eyes to the endless amount of ways that one student can combine aspects of different college courses. I never thought that I would be able to combine aspects of all of the courses that I did for this project. This project encapsulates aspects from python as a software language, numerical analysis, data structures, linear algebra, calculus, and differential equations. Along with analyzing my project as a whole using aspects from data visualization, among other courses on data analytics.