Soliton Analysis
APPM 4350: Fourier Series and Boundary Value Problems
Fall 2021

Authors: Sanjay Kumar Keshava, Mason Divita, and Skylar Gale

Contents

Abstract

In this write-up we study solitons, or solitary waves, in a liquid conduit system. A liquid conduit system consists of a narrow column of a low-density liquid surrounded by a high-density liquid. This type of system supports solitons in the central column. A soliton is explained as a localized traveling wave which moves at a fixed speed, and it is described by a non-linear partial differential equation (known as the conduit equation in this particular case). Typically, the wave-form of a soliton has a single maximum (or minimum) and the amplitude drops to some background value on either side of the maximum (or minimum).

We investigate traveling wave solutions to the conduit equation and use them to derive the speed-amplitude relation for solitons. This relation describes the connection between the speed of a soliton and its maximum (or minimum) amplitude. We then use experimental data from a liquid conduit system to extract the speeds of solitons with a variety of different maximum amplitudes. Next, we compare the experimental results with the expected results based on the analytical speed-amplitude relation. In our case, we find agreement.

We find that traveling wave solutions to the conduit equation (after imposing relevant boundary conditions) satisfy a first-order ordinary differential equation (ODE). Additionally, we numerically solve this ODE using Euler's method. We study how the choice of initial conditions and step size affect the solution. We use the solution to find the spatial wave-form for solitons with arbitrary maximum amplitudes (the shape of the wave-form at a fixed instant in time). Finally, we use experimental data to extract the spatial wave-form of solitons with differing maximum amplitudes and compare them with the expected results from the numerical solution. Once again, we find agreement. Finally, we conclude our write-up by summarizing our findings and suggesting possible future work.

1 Introduction and Mathematical Formulation

In this section, we introduce solitons and discuss how they can be modeled mathematically. In addition, we derive the speed-amplitude relation for solitons in a liquid conduit system.

1.1 What are Solitons?

Solitons, or solitary waves, were discovered by Scottish engineer John Scott Russell in 1834 while he studied water waves in a river channel. In his own words, he describes his first encounter with a soliton by claiming that the wave he saw "rolled forward with great velocity, assuming the form of a large solitary elevation, a rounded, smooth and well-defined heap of water, which continued its course along the channel apparently without change of form or diminution of speed". He decided to call this "the Wave of Translation". Solitons have real-world applications and they arise in the study of fiber optics, magnetism, and nuclear physics. For these reasons, there is much research being done on solitons.

We emphasize that unlike normal waves, solitary waves are "localized". Further, they are stable and can travel over large distances at a fixed speed with no change in their shape. Finally, their speed depends on their maximum (or minimum) amplitude and this is described by the speed-amplitude relation. Solitons are described by non-linear partial differential equations which can be challenging to solve. Below is an example of a soliton profile or wave-form.

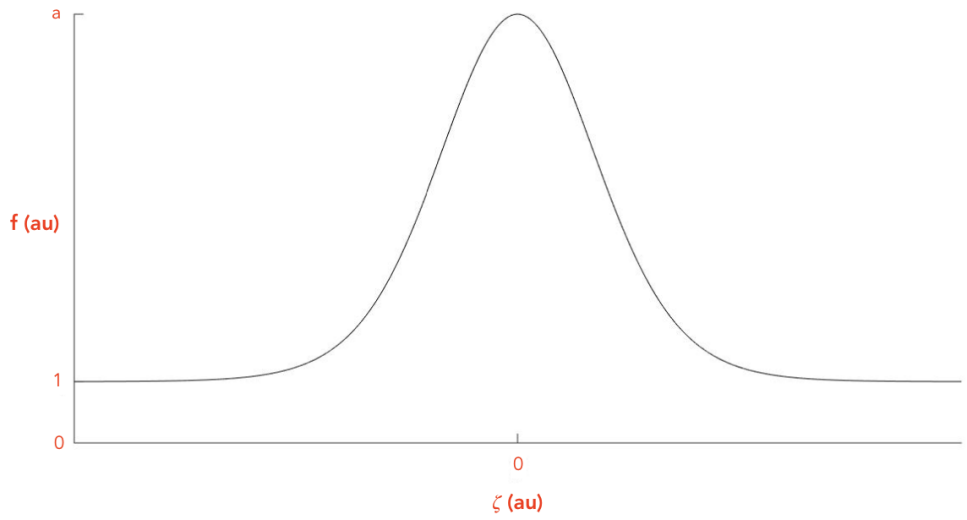

Figure 1: Profile or Wave-Form of a Soliton

In the plot above, the amplitude f of the soliton is given as a function of the position ζ along the direction of wave propagation at a single instant in time. Specifically, $\zeta = z - ct$, where z is the position along the direction of wave propagation, c is the soliton speed and t is time. But we are essentially looking at the case where $t = 0$ and therefore $\zeta = z$. So we can simply plot f against ζ. Like here,

most solitons have a maximum (or minimum) and their amplitude falls to some "background value" on either side (unity in our case). This soliton has a maximum amplitude of a as indicated on the graph.

For our project, we will focus on studying solitons that arise in liquid conduit systems. A liquid conduit system is made up of a narrow column of a low-density liquid surrounded by a high-density liquid. The low-density liquid column at the center supports solitary waves. The solitons that arise in a liquid conduit system can be described by the following equation.

$$A_t + (A^2)_z - (A^2(A^{-1}A_t)_z)_z = 0 \tag{1}$$

Here, $A = A(z,t)$ is the dimensionless cross-sectional area of the soliton in the conduit at position z and time t. This equation can be derived by starting with some physical assumptions and observations, however, we will not delve into the details of the derivation in our write-up. What we have is a partial differential equation and based on the observed properties solitons, we can obtain the following boundary conditions:

$$\lim_{|z|\to\infty} A(z,t) = 1 \tag{2}$$

$$\lim_{|z|\to\infty} A_z(z,t) = 0 \tag{3}$$

$$\lim_{|z|\to\infty} A_{zz}(z,t) = 0 \tag{4}$$

$$A(0,0) = a \tag{5}$$

$$A'(0,0) = 0 \tag{6}$$

For simplicity, the "background" cross-sectional area far from the peak of the soliton has been normalized to unity. The other boundary conditions are evident from the profile or wave-form of a typical soliton. In the next section we will attempt to solve Equation (1) for traveling waves by using the relevant boundary conditions.

1.2 Derivation of the Speed-Amplitude Relation for Solitons

We know that a soliton in a conduit is modeled by Equation (1). Furthermore, we know the properties of a soliton in a conduit imply boundary conditions (2) - (6). At the end of the day, what we have is a higher order non-linear partial differential equation with specific boundary conditions. Due to the non-linearity, the usual methods for solving partial differential equations (such as separation of variables) cannot be used. Instead, we take the approach of finding a specific class of travelling wave solutions. Based on our physical understanding of solitons and how they behave, we assume that the solution to our partial differential equation consists purely of a wave traveling in a single direction. With this in mind, we first perform the following change of variables:

$$A(z,t) = f(\zeta) \tag{7}$$

We let $\zeta = z - ct$ and then, via the chain rule, we can see that the following relations hold

$$A_z = \frac{df}{d\zeta} \cdot \frac{\partial \zeta}{\partial z} = f' \tag{8}$$

$$A_t = \frac{df}{d\zeta} \cdot \frac{\partial \zeta}{\partial t} = -cf' \tag{9}$$

Next, since A_t and A_z are both themselves functions of z and t we can use the results derived above on these functions as well! Considering all this, and the usual product, quotient, and chain rules for derivatives, we can rewrite Equation (1) and the associated boundary conditions as follows

$$-cf' + (f^2)' - (f^2(-cf^{-1}f')')' = 0 \tag{10}$$

$$\lim_{|\zeta|\to\infty} f(\zeta) = 1 \tag{11}$$

$$\lim_{|\zeta|\to\infty} f'(\zeta) = 0 \tag{12}$$

$$\lim_{|\zeta|\to\infty} f''(\zeta) = 0 \tag{13}$$

$$f(0) = a \tag{14}$$

$$f'(0) = 0 \tag{15}$$

Now we can try and solve for the speed-amplitude relation which will help us express c as a function of a. First, we integrate both sides of Equation (10) to obtain

$$-cf + f^2 - f^2(-cf^{-1}f')' = \alpha \tag{16}$$

Where α is an arbitrary constant of integration. Next, we can re-write Equation (16) as follows

$$-cf + f^2 + c(ff'' - (f')^2) = \alpha \tag{17}$$

Taking the limit as $|\zeta|$ becomes very large, we get that

$$\lim_{|\zeta|\to\infty} [-cf + f^2 + c(ff'' - (f')^2)] = -c + 1 = \alpha \tag{18}$$

Therefore we see that $\alpha = 1 - c$ and so, Equation (16) can be re-written as follows

$$-cf + f^2 + cf^2(f^{-1}f')' = 1 - c \tag{19}$$

Now, we can use multiply both sides of Equation (19) by $I = f^{-3}f'$ which behaves as an integrating factor to obtain the following equation

$$-cf^{-2}f' + f^{-1}f' + c(f^{-1}f')(f^{-1}f')' = (1-c)f^{-3}f' \tag{20}$$

The goal is integrate both sides of Equation (20). But first, we will re-write it in the following manner

$$(cf^{-1})' + (ln[f])' + \left(\frac{c(f^{-1}f')^2}{2}\right)' = \left(\frac{(c-1)f^{-2}}{2}\right)' \tag{21}$$

Now we can integrate this very easily, and we simply obtain

$$cf^{-1} + ln[f] + \frac{c(f^{-1}f')^2}{2} = \frac{(c-1)f^{-2}}{2} + \beta \tag{22}$$

Where β is some arbitrary constant of integration. Taking the limit as $|\zeta|$ becomes very large, we get

$$\lim_{|\zeta|\to\infty} \left[cf^{-1} + ln[f] + \frac{c(f^{-1}f')^2}{2} - \frac{(c-1)f^{-2}}{2} \right] = c - \left(\frac{c-1}{2} \right) = \frac{c+1}{2} = \beta \tag{23}$$

So we know β and plugging this back into Equation (22) gives us

$$cf^{-1} + ln[f] + \frac{c(f^{-1}f')^2}{2} - \frac{(c-1)f^{-2}}{2} - \frac{c+1}{2} = 0 \tag{24}$$

Now we can simply let ζ go to zero and then, this gives

$$\frac{c}{a} + ln[a] - \frac{c-1}{2a^2} - \frac{c+1}{2} = 0 \tag{25}$$

And finally, by re-arrangement, we obtain the conduit speed-amplitude relation for solitons as

$$c = \frac{2a^2 ln[a] - a^2 + 1}{(a-1)^2} \tag{26}$$

Next, we can plot c against a to understand how the soliton speed depends on the maximum amplitude in a graphical manner. We note that both c and a are dimensionless quantities as given in Equation (26). First, we study the case where $a > 1$ and relatively small. The plot is below.

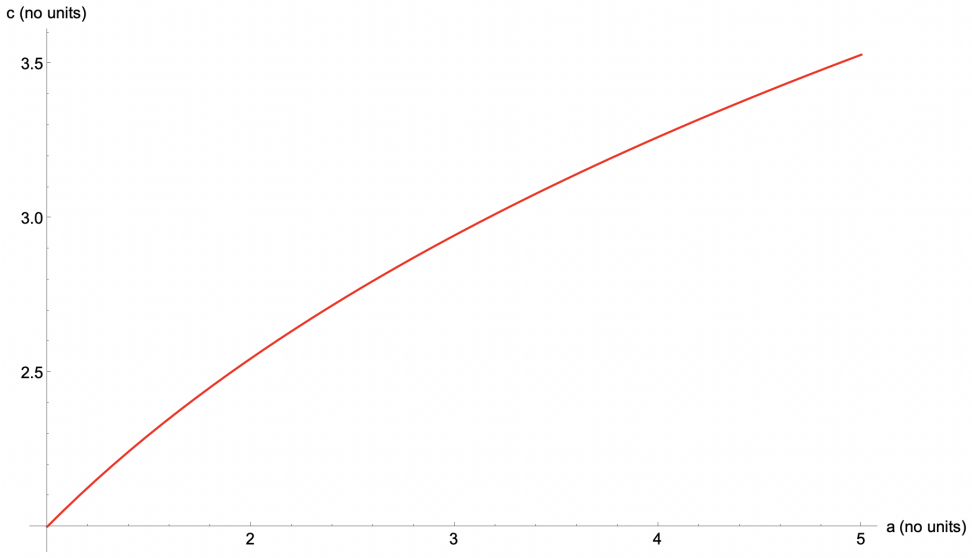

Figure 2: Speed-Amplitude Relation for Solitons in a Conduit where $a > 1$ (and is relatively small)

Next, we also take a look at the large amplitude regime where $a >> 1$ and the plot for this is below.

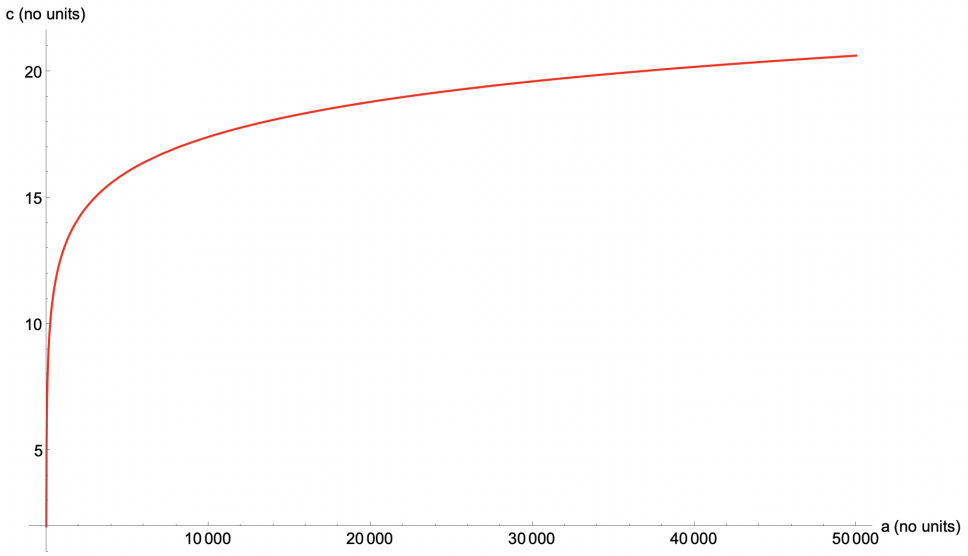

Figure 3: Speed-Amplitude Relation for Solitons in a Conduit when $a >> 1$

Essentially we can see that the larger the maximum amplitude a of a soliton, the greater its speed c. Further, we see that for relatively small a values, the increase in c for a given increase in a is much greater than at large a values. Finally, out of interest we also consider the case where $0 < a < 1$ which corresponds to so called "dark solitons". Since the background amplitude is unity, we see that what we have is a traveling constriction in the conduit (rather than a traveling bulge like in the case where $a > 1$). The plot for this is shown below.

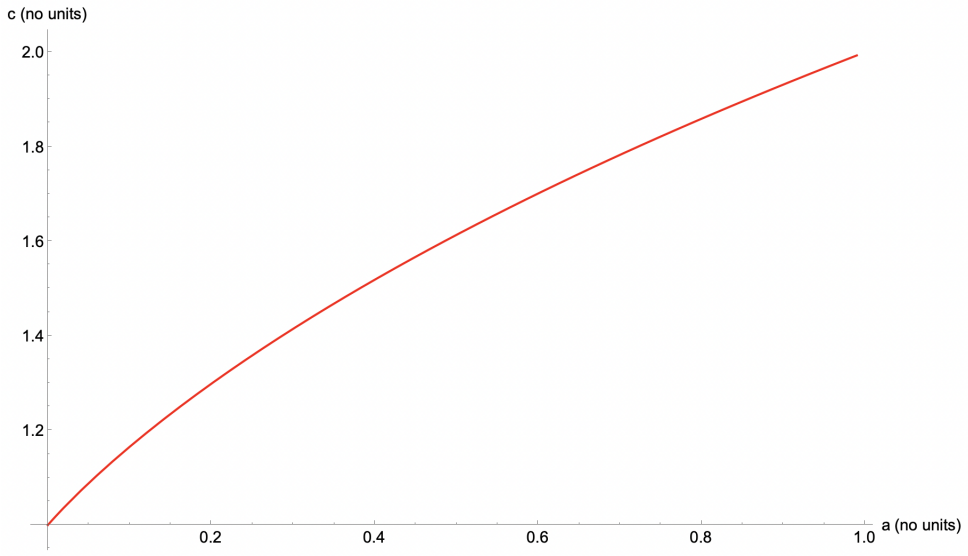

Figure 4: Speed-Amplitude Relation for Solitons in a Conduit when $0 < a < 1$

Here a represents the minimum amplitude of the soliton, and as we can see, the smaller the minimum amplitude a, the lower the speed c. It turns out that dark solitons are unstable in real-life conduits and they tend to collapse into shock waves rather rapidly. Next, we move on to look at some real data and investigate if the speed-amplitude relation that we derived earlier agrees with experimental results.

2 Numerical Work

In this section we determine the speeds of solitons with differing maximum amplitudes using experimental data. We then compare these results with those obtained from the analytical speed-amplitude relation derived earlier. Additionally, we numerically solve the ODE that defines traveling wave solutions to the conduit equation, and we use this to find the profiles or wave-forms of solitons with arbitrary amplitudes. We then compare our results with soliton profiles obtained from experimental data.

2.1 Finding Soliton Speeds Using Experimental Data

Using experimental soliton data-sets provided by the *Dispersive Hydrodynamics Laboratory*, we were able to determine the speed of solitons with differing amplitudes. These data-sets included information about the amplitude of the solitons at multiple points in space and time, the maximum soliton amplitude to be used in calculations (along with the associated error), and a dimensionalization factor for the speed. To be more specific each data set had the following information:

Variable Name	**Variable Type**	**Variable Description**
t	vector	time vector (seconds)
z	vector	spacial vector (cm)
A	matrix	amplitude (dimensionless)
U0	scalar	velocity scale (cm/s)
A0	scalar	soliton amplitude (dimensionless)
A0 error	scalar	soliton amplitude error (dimensionless)

Figure 5: Variables in Experimental Soliton Data-Sets

The t vector is a time vector, the z vector is a spatial position vector, and the A matrix gives the soliton amplitude at each possible combination of spatial position and time. A0 and its error give the maximum amplitude of the soliton along with its associated error. We will use A0 for calculations. Finally, U0 is a speed scale factor which can be used to convert unit-less speeds to those with units.

With all this in mind, we started by creating contour plots to show the soliton amplitude as a function of time and spatial position. We did this for 8 different data-sets in total, however we only display plots from 4 data-sets in our write-up to avoid clutter. These contour plots are shown in Figure (6) below. To be clear, in all the plots, the vertical axis represents time in seconds, the horizontal axis represents spatial position in centimeters, and the color on the contour map is used to indicate the relative amplitude (given by the area ratio) at each point in space and time. Based on our understanding of solitons, ideally speaking, the "background amplitude" should be unity. We note that for some of the plots, the color legend bar starts at a value less than 1 and we attribute this to noise in the experimental data. However, this is not very important as it will not affect our calculations.

Next, we move to calculate the speeds of solitons. In each of the contour plots, we can see multiple maxima (bright green) and this is because there are multiple solitons! As visible, each maxima traces out a diagonal line, and this tells us that the solitons are moving. In particular, we note that a given maxima represents a soliton peak. For the sake of calculation, we only consider the leading edge in each contour plot (this represents a single soliton). In a given contour plot, if we follow the maxima

of the leading edge, we see that as we go forward in time, the position of the maxima moves in space. And we can see that the soliton is moving at a constant speed!

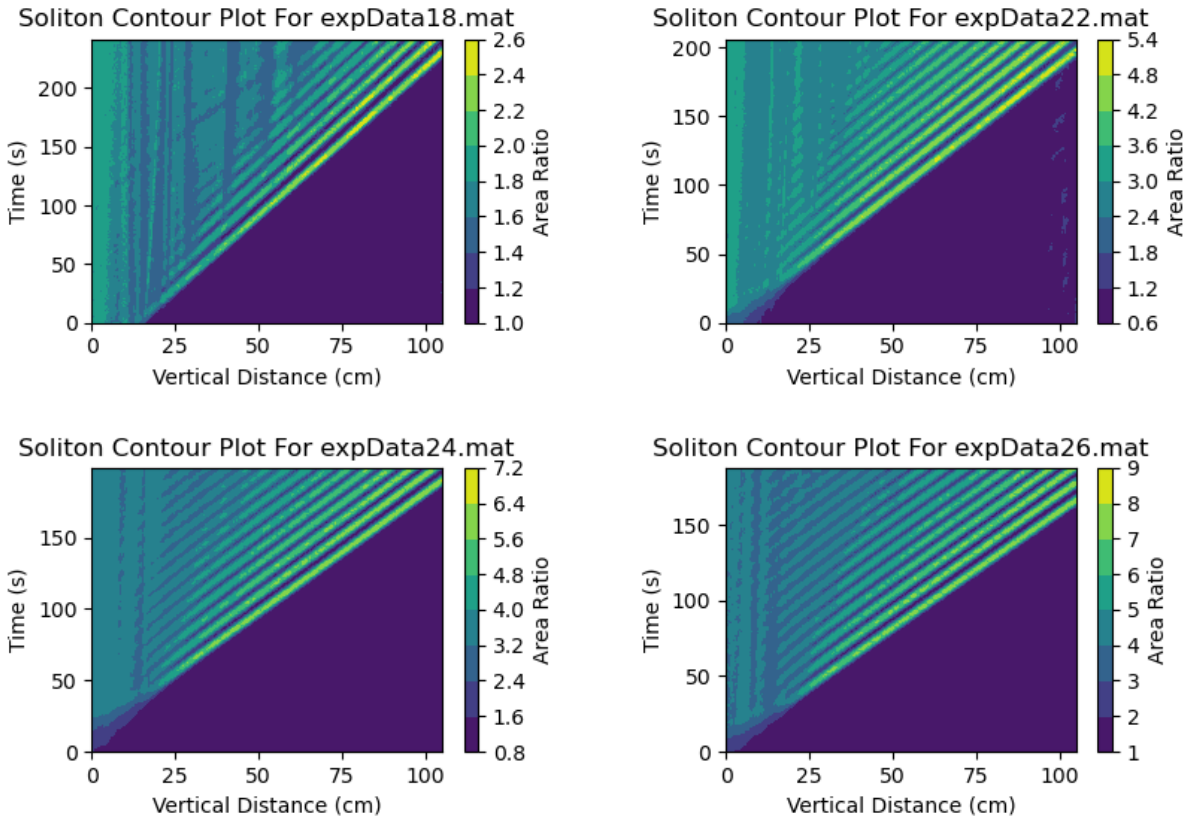

Figure 6: Contour Plots of Amplitude as a Function of Spatial Position and Time

Therefore, for a data-set, the speed of the soliton represented by the leading edge can be found by calculating the reciprocal of the slope of the line traced by the corresponding maxima. In order to determine the slope of this line, we need a good sample of points that describe it, and we can then use a linear fit. To obtain a decent sample of points, we took a slice of the contour plot at a particular amplitude $A_c < A0$ and ignored all the points that were not associated with the leading edge. We then fit the resulting selection of points to a linear function. We found that as long as A_c was close to the maximum amplitude A0, its precise value was unimportant. In other words, the soliton speed we calculated was not very sensitive to changes in A_c. It should be noted that choosing $A_c = A0$ gave us too few points leading to a poor fit. Figure (7) which is given below summarizes our approach. We only include plots from 4 of the 8 data-sets we processed to avoid clutter.

The orange colored points represent data from the leading edge at amplitude A_c. The pale blue line represents the line of best fit through this selection of points. The linear fit allowed us to the calculate the slope of this line, and we simply took the reciprocal to find the soliton speed for that particular data-set. After running the same process for all of the data sets, we arrived at the results given in Figure (8) below.

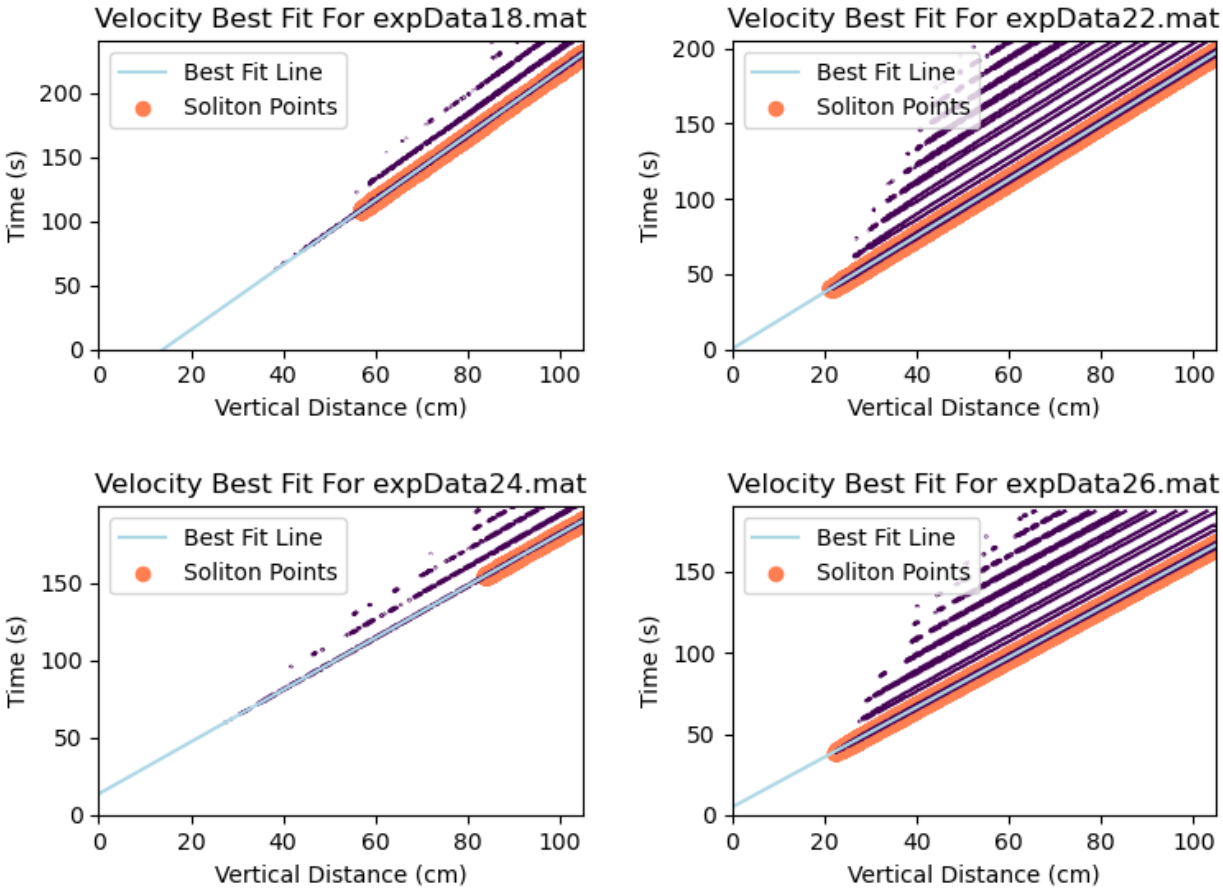

Figure 7: Contour Plot Slice with Line of Best Fit Overlay

Data Set	Amplitude	Experimental Velocity (cm/s)
expData18.mat	2.46	0.39
expData20.mat	3.68	0.47
expData21.mat	4.37	0.5
expData22.mat	4.96	0.53
expData23.mat	5.38	0.57
expData24.mat	6.3	0.6
expData25.mat	7.07	0.63
expData26.mat	7.79	0.65

Figure 8: Experimentally Determined Velocities From Data Sets with Give Amplitudes

In Figure (8), the second column shows the maximum amplitude A0 and the third column shows the experimental soliton speed. From these results, we can verify that solitons with larger amplitudes had greater velocities as predicted by our speed-amplitude relation. Next, we move to do a quantitative comparison to test if these results agree with the derived speed-amplitude relation.

2.2 Reliability of the Speed-Amplitude Relation

While the experimental data follows the trend set by our speed-amplitude relation, we need to investigate how well it actually fits our model. First, we used the analytical speed-amplitude relation (shown below) to find the dimensionless speed of our solitons.

$$c = \frac{2a^2 ln[a] - a^2 + 1}{(a-1)^2} \tag{27}$$

Essentially, we used the above formula to calculate the predicted speed c by replacing every a with A0.

$$c = \frac{2(A0)^2 ln[(A0)] - (A0)^2 + 1}{((A0)-1)^2} \tag{28}$$

We did this for each value of A0 in Figure (8). Next, we used the error in the amplitude A0 to compute the error in our predicted speed c. For this, we used the general error propagation formula

$$\Delta c = \frac{dc}{da}\Delta a \tag{29}$$

Where the derivative is given by

$$\frac{dc}{da} = \frac{-2a ln[a] + 2a^2 - 2}{(a-1)^3} \tag{30}$$

Then we evaluated the derivative at each given amplitude, A_0, and multiplied by the error in amplitude ΔA_0 to get the following:

$$\Delta c = \frac{-2A0 ln[A0] + 2A0^2 - 2}{(A0-1)^3}\Delta A0 \tag{31}$$

Finally, we multiplied our results by the speed scale U0 to dimensionalize them and this gave us the following expression for the soliton speed along with the associated error as a function of the amplitude A0:

$$v \pm \Delta v = U0(c \pm \Delta c) = U0\left[\frac{2(A0)^2 ln[(A0)] - (A0)^2 + 1}{((A0)-1)^2} \pm \frac{-2A0 ln[A0] + 2A0^2 - 2}{(A0-1)^3}\Delta A0\right] \tag{32}$$

Where v is the dimensionalized speed (in cm/s) and Δv is the associated error. Next, we plotted $v \pm \Delta v$ against A0 and also included the experimental soliton speeds calculated earlier. This is shown in Figure (9) below. The blue points represent experimental data, and the red points represent theoretical data corresponding to the same amplitude. the red vertical lines represent the error on the theoretical soliton speed. Finally, the yellow curve is the theoretical curve. We can see that velocity of the experimental solitons fall within the error bars of the analytical result. This tells us that the experimental data agrees with the theoretical prediction. However, all the experimental data points are lower than the analytical prediction by close to the same amount. We may therefore guess that there are either gaps in our model, or unaccounted systematic error in the experimental process. The code

for sections 2.1 and 2.2 was written in Python and is shown in the appendix. Next, we move on to study the wave-forms of solitons.

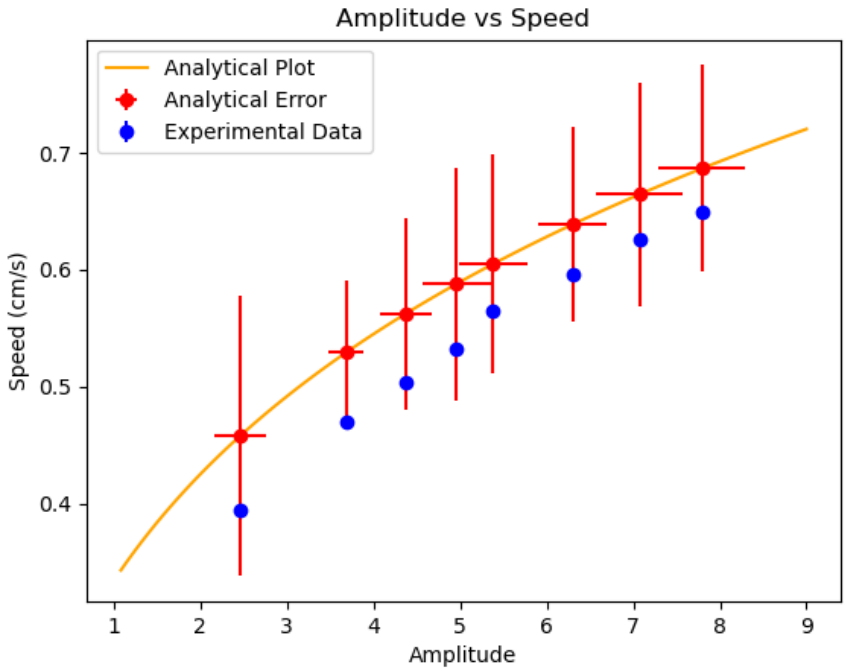

Figure 9: Experimental Velocities And Analytical Velocities with Error

2.3 Deriving a Numerical Solution with Euler's Method

In this section we look at the wave-forms of solitons with different amplitudes and compare experimental data with the theoretical expectations. We explain how we used Euler's method to obtain numerical solutions to Equation (24), how these solutions compare to the experimental data, and discuss important characteristics of the numerical solutions.

Euler's method simply finds an approximate solution to a differential equation by using a step-wise approximation approach. The process in which Euler's method was applied to the soliton problem contains many particular steps. To begin, we consider Equation (24). This equation is a first-order ODE that defines traveling wave solutions to the conduit equation for solitons. It is important to correctly derive the approximation scheme obtained by Euler's method. We first solve for $f' = \frac{df}{d\zeta}$ as shown below.

We begin with

$$cf^{-1} + ln[f] + \frac{c(f^{-1}f')^2}{2} - \frac{(c-1)f^{-2}}{2} - \frac{c+1}{2} = 0 \tag{33}$$

Moving the term with f' to the left hand side gives us

$$\frac{c(f^{-1}f')^2}{2} = \frac{(c-1)f^{-2}}{2} + \frac{c+1}{2} - cf^{-1} - ln[f] \tag{34}$$

Multiplying each side by 2 yields

$$c(f^{-1}f')^2 = (c-1)f^{-2} + c + 1 - 2cf^{-1} - 2ln[f] \tag{35}$$

Dividing each side by c gives us

$$(f^{-1}f')^2 = \frac{(c-1)f^{-2}}{c} + \frac{c+1}{c} - 2f^{-1} - \frac{2ln[f]}{c} \tag{36}$$

Taking the positive square root of each side yields

$$(f^{-1}f') = \sqrt{\frac{(c-1)f^{-2}}{c} + \frac{c+1}{c} - 2f^{-1} - \frac{2ln[f]}{c}} \tag{37}$$

Finally, multiplying each side by f gives us

$$f' = f\sqrt{\frac{(c-1)f^{-2}}{c} + \frac{c+1}{c} - 2f^{-1} - \frac{2ln[f]}{c}} \tag{38}$$

When referencing the MATLAB code used this equation is written equivalently as

$$f' = f\sqrt{\frac{(c-1)}{cf^2} + \frac{c+1}{c} - \frac{2}{f} - \frac{2ln[f]}{c}} \tag{39}$$

Since Equation (39) is only a function of f and c, it can be expressed as $\mathrm{d}f/\mathrm{d}\zeta = g(f,c)$. In this equation, c is given by the speed-amplitude relation derived earlier in this paper (see Equation (26)). Note that $\zeta = z - ct$ in order to relate back to the original equation defining a soliton in terms of $A(z,t)$. The square root in Equation (39) can contain either positive or negative values, where the positive values are located on the left side of the soliton and the negative values are located on the right side. The easiest approach to find a solution with this is to find $\zeta(f)$ and then invert the solution. Therefore, in order to utilize Euler's method, we will be working with $\mathrm{d}\zeta/\mathrm{d}f$ initially instead. One aspect of important notice is that, both when far away from the soliton and when $\zeta = 0$, we have that $\mathrm{d}f/\mathrm{d}\zeta$ is zero. Therefore, to correctly calculate the numerical solution, we must avoid the values ζ = 0, ∞, -∞. To do this, the domain of the problem is restricted to ζ = (-∞,0). From there, we can simply reflect this solution over the +ζ-axis to get the full soliton profile (this also avoid the issues with negative square roots). Knowing that $\mathrm{d}\zeta/\mathrm{d}f = g(f,c)^{-1}$ we get that the equations to build the step wise linear approximation are:

$$\zeta_{i+1} = \zeta_i + \Delta f\frac{d\zeta}{df} = \zeta_i + \frac{\Delta f}{g(f_i,c)} \tag{40}$$

$$f_{i+1} = f_i + \Delta f \tag{41}$$

The next important step to setting up Euler's method is to determine the appropriate initial conditions. In order to create the full soliton profile accurately, the initial point $\zeta_0 = \zeta(f_0)$ needed to be a far distance from the maximum amplitude point and in a region where $f \approx 1$ and therefore $f' \approx 0$. A clear solution can be found by setting $f_0 = 1 + \sigma$ where σ (also known as the tolerance) is some small number. We let σ to be 0.001 initially. We chose ζ_0 to be -20 initially. Further, we set the maximum amplitude to $a = 3$ and therefore our f values ranged from 1 to 3 and we $\Delta f = 0.001$ as our step size.

Given that we start at $i = 0$ here is the basic protocol we followed for implementing Euler's method: (1) ζ_i and f_i are known (2) use Equation (40) to find ζ_{i+1} and then use Equation (41) to find f_{i+1} (3) let i go to $i + 1$ (4) if $i \neq n$ where n is the [# of steps + 1] then start over at step (1), else stop. We note that in general $n = \frac{a-1}{\Delta f}$. At the end, we had a list of ζ values and a list of f values to go along with it. We then performed a reflection, and after generating the full profile solution, we shifted the output ζ vector such that $\zeta(f_n) = 0$. In this case f_n represents the final index of the f output vector, and is equivalent to the maximum amplitude a. The shift is only possible since $f(0) = a$. An example of a complete soliton profile (after being reflected and shifted) is shown below for maximum amplitude $a = 3$.

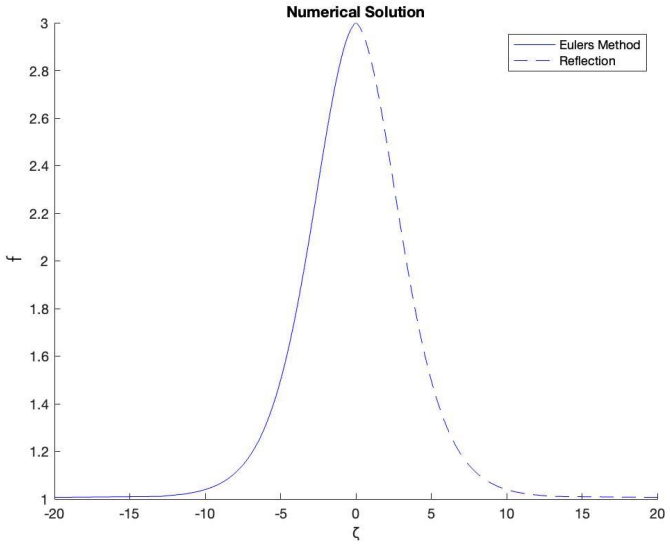

Figure 10: Demonstration of Complete Numerical Solution Implementation for Soliton Profile

As we can see this resembles the typical shape of a soliton, so we can be fairly certain that our approach and steps are correct.

2.4 Important Considerations of the Numerical Solution

After obtaining a numerically-derived solution for the soliton problem, several important characteristics about the solution were evaluated. First, we will consider what happens by adjusting the tolerance σ of the solution (while keeping all other parameters constant and the same as before). As demonstrated in the plots below, changing the tolerance affects how much of the full solution profile is generated. As the tolerance decreases, more of the solution is developed. This is because, as the tolerance increases, f_0 approaches the maximum amplitude of the soliton and the solution does not allow for any points below the starting f_0 value to be generated. Therefore, for an accurate solution a very small tolerance (something like 0.001 or smaller) should be selected since we know the background amplitude needs to be unity.

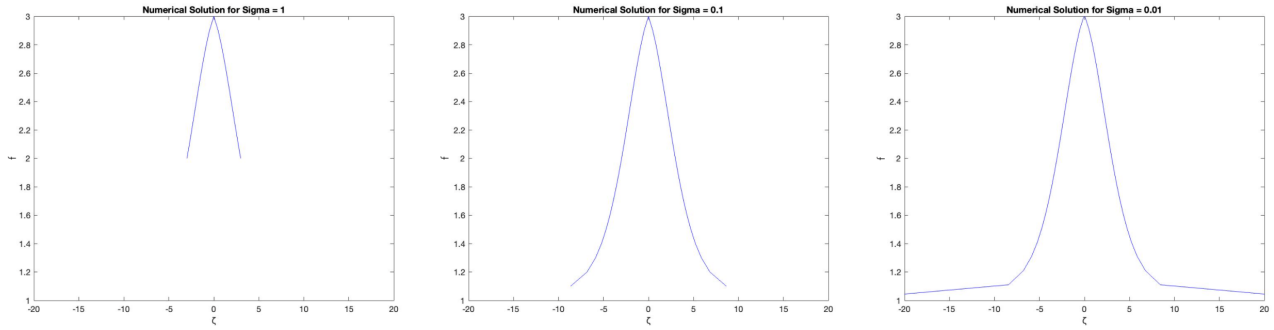

Figure 11: Soliton Profile for Varying Values of Tolerance

Now we focus on the step size Δf (while keeping other parameters constant), changing this value can also impact the solution. Decreasing the step size of the solution generates a more accurate solution, as demonstrated in the figure below. As the step size approaches zero, the solution approaches a limit, specifically the smooth generation of the soliton profile of the given maximum amplitude. This is due to the fact that the step size can impact the number of iterations (i.e. points) generated for the given plot. If the step size is too large, the points generated do not develop a good approximation of the soliton profile since the data points are more "spread out". For smaller step size values, the approximation more closely resembles the profile since more points can be generated and are located "closer" together on the plot (this is directly deduced from how Euler's method works).

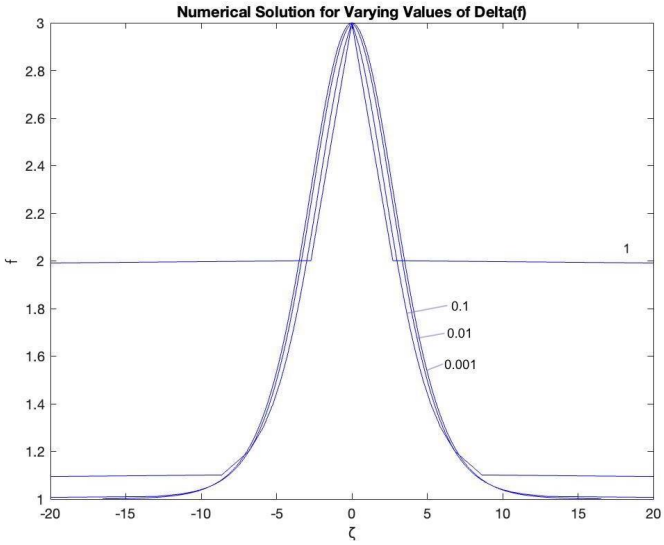

Figure 12: Soliton Profile for Varying Values of Δf

Finally, we look at how the maximum amplitude a affects the soliton solution. As seen in the figure below, as a (also referenced to as A in the plots below) approaches 1, the soliton solution flattens. This is because the limit as ζ approaches positive or negative infinity of f is 1. So, if the maximum amplitude is very close to 1, then it looks as if the soliton is flattening (or in other words there is no soliton).

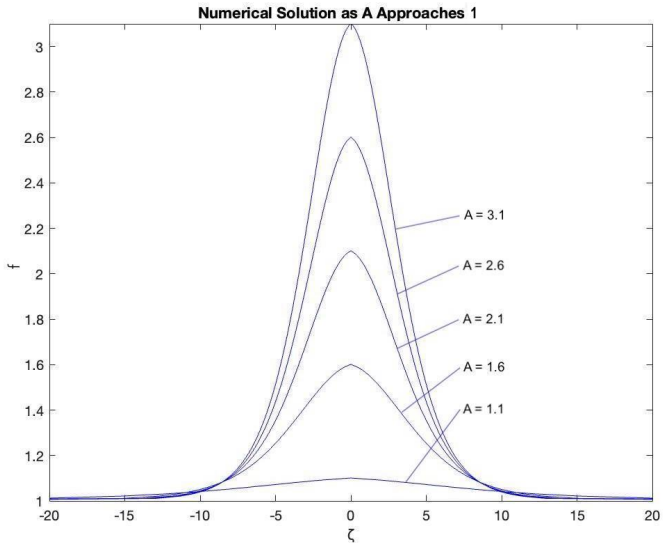

Figure 13: Soliton Profiles as A Approaches Zero

Similarly, as A gets larger, the shape of the profile "stretches" in order to reach the higher maximum amplitude. As seen in the figure below, when plotted against other amplitudes, it looks as if the larger A values generate slightly thinner, taller soliton profiles.

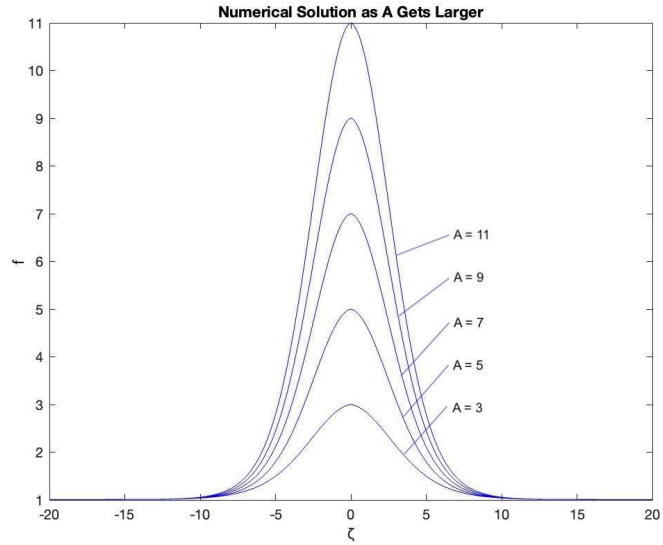

Figure 14: Soliton Profiles as A gets Larger

2.5 Comparing Experimental and Numerical Soliton Wave-Forms

Next, we compare the experimental and numerical soliton wave-form shapes for two different data sets. To find the experimental soliton shape, we begin by creating a contour plot of the amplitude as a function of time and spatial position as before. Then, we choose an appropriate time and we isolate

the leading edge (which represents 1 soliton). Finally, we plot the amplitude versus the spatial position and center the peak at zero to obtain the experimental soliton wave-form shape or profile. We use this profile to find the maximum amplitude of the soliton and then find the corresponding wave-form using our numerical solution. As more terms are added to the numerical solution (smaller step size), it more closely resembled the profiles derived from the experimental data. This is shown in the plots below. In the plots of the numerical solutions, the red profile represents a step size of 1, the green profile represents a step size of 0.1, and the blue profile represents a step size of 0.01. It can clearly be seen that as the step size decreases (i.e. more terms are added to the solution), the profile slowly approaches the experimental soliton shape. However, it should be noted that the experimental soliton profile is somewhat noisy, but we still see overall agreement between the experimental and numerical soliton profiles. The code for sections 2.3, 2.4, and 2.5 was written in MATLAB and it can be found in the appendix.

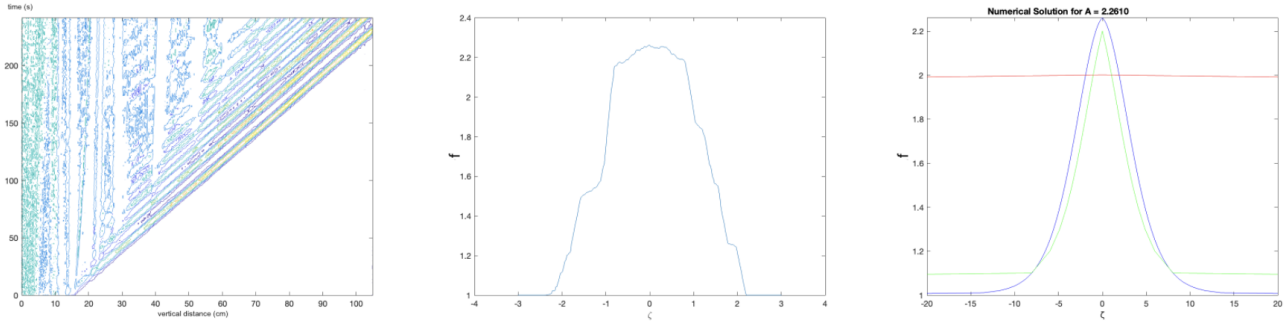

Figure 15: Dataset 18: Experimental Contour Plot, Soliton Profile, and Numerical Solution with Calculated Maximum Amplitude

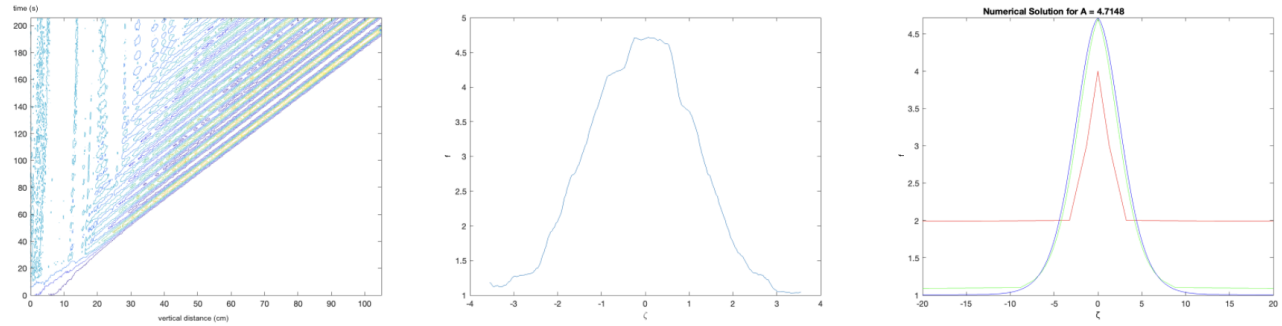

Figure 16: Dataset 22: Experimental Contour Plot, Soliton Profile, and Numerical Solution with Calculated Maximum Amplitude

3 Conclusions and Future Work

For our project, we studied solitons, or solitary waves, that arise in a liquid conduit systems. These waves can be modeled by the conduit equation which is a non-linear partial differential equation.

In section 1.1 we introduced solitons and described them both physically as traveling waves and mathematically in terms of the conduit equation. In section 1.2 we searched for traveling wave solutions

to the conduit equation and using boundary conditions we derived the speed-amplitude relations for solitons. This relations describes how the speed of a soliton depends on its maximum amplitude. We found overall that solitons with a larger amplitude traveled faster.

In section 2.1 found the experimental speeds of solitons with different amplitudes using data provided to us. We used contour plots lines of best fit to achieve this. In section 2.2 we compared the results obtained to the expected results from the analytically derived speed-amplitude relation. From there we determined that the experimental data followed our predicted model within our error range. However, we also found that our experimental results were always lower (by roughly the same amount) when compared with the theoretical prediction. Therefore we believe our model is either incomplete or there is an unaccounted systematic error in the experimental setup. The coding for these sections was done on Python.

In section 2.3 we derived and numerically solved the ODE that represents traveling wave solutions to the conduit equation. This was done using Euler's method and it allowed us to generate the profile of solitons with arbitrary amplitudes. In section 2.4, we studied how the initial conditions and the step size affected the solution for the soliton profile. We noted that in order to obtain a sensible solution, the initial condition needs to be chosen very cautiously and a small step size needs to be used. In section we found soliton profiles (for different maximum amplitudes) using experimental data and we compared them with the results from our numerical solution. Although the experimental results were noisy, we found that the numerical solutions and the experimental results closely resembled each other when the numerical solutions had a small step size (i.e. there were more data points generated using Euler's method). This was as we expected.

Overall, we found that the conduit equation which describes solitons in a liquid conduit was able to model the behaviour of solitons quite well. We found that the theoretical predictions matched experimental results that were obtained by performing measurements on conduit solitons in a lab. As a future project, we could try to investigate why the experimental soliton speeds were always lower than the theoretical speeds. This can be done by modifying the mathematical model for solitons, or by looking more closely at the experimental protocol to try and rule out systematic errors. Further, one could move to study solitons that arise in other areas of physics such as in optical fibres. In particular, it might be worth looking into some concrete real-world applications of solitons.

4 Appendix

4.1 MATLAB Code

```
% Initial conditions
a = 3; % Amplitude

deltaf = 0.0001;  % Step size
range = 1:deltaf:a;  % Range (y-direction) vector
l = numel(range);  % Number of steps to maximum aplitude

f = zeros(size(range));  % Pre-allocate f vector
zeta = zeros(size(range));  % Pre-allocate zeta vector

sigma = 0.001; % Tolerance
f(1) = 1 + sigma; % Initial f value
zeta(1) = -100;  % Initial zeta value

% Speed-Amplitude Relation
c = (2 * (a^2) * log(a) - (a^2) + 1) / ((a - 1)^2);

% Euler's Method
for i = 1:l-1
    % Determine dzeta/df for given f(i)
    g = f(i) * sqrt((c-1)/(c * ((f(i))^2)) + (c+1)/c - 2/(f(i)) - (2*log(f(i)))/c);
    % Calculate next zeta value
    zeta(i+1) = zeta(i) + deltaf/g;
    % Calculate next f value
    f(i+1) = f(i) + deltaf;
end

% Shift curve
zeta = zeta - zeta(l);

% Reflect and plot final solution
figure (1)
plot(zeta,f,'b',-zeta,f,'b')
hold on
xlabel('ζ')
ylabel('A')
title('Numerical Solution')
axis([-20 20 1 a]) % Plot ranges
```

```
% load the appropriate file (anything between 18 and 26)
% contour map

figure(1);
contour(z_vec,t_vec,Amat)

% See the contour map
% Choose a time from the vertical axis
% Look at leading edge
% Choose startz and endz from the horizontal axis to get one soliton

time = 150;
startz = 58;
endz = 80;
dif = abs(time-t_vec(1));

for i = 1:numel(t_vec)
    if abs(time-t_vec(i)) <= dif
        dif = abs(time-t_vec(i));
        j=i;
    end
end
dif2 = abs(startz - z_vec(1));

for i = 1:numel(z_vec)
    if abs(startz - z_vec(i)) <= dif2
        dif2 = abs(startz- z_vec(i));
        k=i;
    end
end
dif3 = abs(endz - z_vec(1));

for i = 1:numel(z_vec)
    if abs(endz - z_vec(i)) <= dif3
        dif3 = abs(endz - z_vec(i));
        l=i;
    end
end
figure(2);
fullprofile = Amat(j,k:l);
[M,I] = max(fullprofile);
if (I>= ((l-k)/2))
    k = l - (2*I);
else
    l = k+(2*I);
end
fullprofile = Amat(j,k:l);

plot((z_vec(k:l)-(z_vec(k+I))),fullprofile)
xlabel('\zeta')
ylabel('A')
Amax = M
```

4.2 Python Code

File - Soliton.py

```python
import numpy as np
import scipy.io
from sympy.utilities.lambdify import lambdify, implemented_function
from sympy.abc import a
import sympy
import matplotlib.pyplot as plt
from prettytable import PrettyTable


# symbolic representation of the analytical solution
sar_fun = (2 * a ** 2 * sympy.log(a) - a ** 2 + 1) / ((a - 1) ** 2)
sar_derivative = sympy.diff(sar_fun, a)


# function representation of the analytical solution
sar_lam = lambdify(a, sar_fun)
sar_der_lam = lambdify(a, sar_derivative)


# variable set up
np.seterr(divide='ignore', invalid='ignore')  # suppresses division by 0 error print out
data = []  # where the .mat files are opened to
X = []  # stores the z_vec array for each data set
Y = []  # stores the t_vec array for each data set
Z = []  # stores the Amat 2D array for each data set
x = []  # stores the x position of all the points to be processed in the best fit function
y = []  # stores the y position of all the points to be processed in the best fit function
x2 = []  # stores the x data to plot the best fit line
y2 = []  # stores the y data to plot the best fit line
Vel = []  # stores the experimental velocity of each soliton
Vel_error = []  # stores the error of the experimental velocity
ana_vel = []  # stores the analytically calculated velocity from each A0
ana_vel_x_error = []  # stores the error in the amplitude from A0_error
ana_vel_y_error = []  # stores the error in the analytical calculation (determined through error propagation)
linFit = []  # stores the lin fit data for each data set
p = []  # stores all the oaths from the slice of contour plot
v = []  # stores the chosen paths from the slice of contour to be used for the best fit function
cp = []  # stores all the contour plots
cV = [2.2, 3, 3.5, 3.5, 3.5, 5.6, 4.8, 5.6]  # amplitude slice values
cPl = [0, 0, 0, 0, 0, 0, 1, 0]  # path to be chosen
fN = ["18", "20", "21", "22", "23", "24", "25", "26"]  # number of .mat files in string form
fileNums = [0, 3, 5, 7]  # index selection of data sets to be displayed


# Opens all the data set files
for i in fN:
```

```
47      data.append(scipy.io.loadmat(
48          "C:\\Users\coolg\Downloads\Experimental_data_soliton_project\
    expData" + i + ".mat"))  # opens the .mat file
49
50
51  # Grabs the contour plot data from each data set
52  for i in range(len(cV)):
53      X.append(data[i].get('z_vec')[0])
54      Y.append(data[i].get('t_vec')[0])
55      Z.append(data[i].get('Amat'))
56
57
58  # Contour plots of each data set
59  fig, axs = plt.subplots(2, 2)
60
61  axs[0, 0].set_title("Soliton Contour Plot For expData" + fN[fileNums[0
    ]] + ".mat")
62  cp1 = axs[0, 0].contourf(X[fileNums[0]], Y[fileNums[0]], Z[fileNums[0]])
63  fig.colorbar(cp1, ax=axs[0, 0], label="Area Ratio")  # Add a colorbar to
    the plot
64
65  axs[0, 1].set_title("Soliton Contour Plot For expData" + fN[fileNums[1
    ]] + ".mat")
66  cp2 = axs[0, 1].contourf(X[fileNums[1]], Y[fileNums[1]], Z[fileNums[1]])
67  fig.colorbar(cp2, ax=axs[0, 1], label="Area Ratio")
68
69  axs[1, 0].set_title("Soliton Contour Plot For expData" + fN[fileNums[2
    ]] + ".mat")
70  cp3 = axs[1, 0].contourf(X[fileNums[2]], Y[fileNums[2]], Z[fileNums[2]])
71  fig.colorbar(cp3, ax=axs[1, 0], label="Area Ratio")
72
73  axs[1, 1].set_title("Soliton Contour Plot For expData" + fN[fileNums[3
    ]] + ".mat")
74  cp4 = axs[1, 1].contourf(X[fileNums[3]], Y[fileNums[3]], Z[fileNums[3]])
75  fig.colorbar(cp4, ax=axs[1, 1], label="Area Ratio")
76
77  for ax in axs.flat:
78      ax.set(xlabel='Vertical Distance (cm)', ylabel='Time (s)')
79  fig.canvas.set_window_title('Contour Plots')
80
81
82  # Gathering the slope data from each data set
83  fig56, ax56 = plt.subplots()  # this will open an extra figure with no
    useful information on it,
84  # there was no other way to get the contour data
85  fig56.canvas.set_window_title('Useless Plot')
86  ax56.set_title('Useless Plot (see line 82)')
87
88  for i in range(len(cV)):
89      cp.append(ax56.contour(X[i], Y[i], Z[i], [cV[i]]))  # makes a contour
    plot at specific amplitude 10
90      p.append(cp[i].collections[0].get_paths()[cPl[i]])  # grabs all the
```

```
90 points from the contour plot
91     v.append(p[i].vertices)
92     x.append(v[i][:, 0])
93     y.append(v[i][:, 1])
94 
95 
96 # Finding the velocities of the solitons from each data set
97 for i in range(len(cV)):
98     linFit.append(np.polyfit(x[i], y[i], 1, cov=True))  # runs a 1st
   degree polynomial fit on the data
99     y2.append(np.linspace(0, 150, 150) * linFit[i][0][0] + linFit[i][0][1
   ])
100     x2.append(np.linspace(0, 150, 150))
101     Vel.append(1 / (linFit[i][0][0]))
102     Vel_error.append(Vel[i] * (linFit[i][1][0][0] / linFit[i][0][0]))
103     ana_vel.append(sar_lam(data[i].get('A0')[0][0]) * data[i].get('U0')[0
   ][0])
104     ana_vel_x_error.append(data[i].get('A0_error')[0][0])
105     ana_vel_y_error.append(sar_der_lam(data[i].get('A0')[0][0]) *
   ana_vel_x_error[i])
106 
107 
108 # Table of velocity Values
109 table = PrettyTable()
110 table.field_names = ["Data Set", "Amplitude", "Experimental Velocity (cm/
   s)"]
111 for i in range(len(cV)):
112     table.add_row(["expData" + fN[i] + ".mat", data[i].get('A0')[0][0],
   round(Vel[i], 2)])
113 
114 print(table)
115 
116 
117 # plots the lin fit on top of the slice of contour plot
118 fig2, axs2 = plt.subplots(2, 2)
119 
120 axs2[0, 0].scatter(x[fileNums[0]], y[fileNums[0]], color='coral')
121 axs2[0, 0].plot(x2[fileNums[0]], y2[fileNums[0]], color='lightblue')
122 cp5 = axs2[0, 0].contour(X[fileNums[0]], Y[fileNums[0]], Z[fileNums[0
   ]], [cV[fileNums[0]]])
123 axs2[0, 0].set_title("Velocity Best Fit For expData" + fN[fileNums[0]] +
   ".mat")
124 axs2[0, 0].legend(["Best Fit Line", "Soliton Points"])
125 
126 axs2[0, 1].scatter(x[fileNums[1]], y[fileNums[1]], color='coral')
127 axs2[0, 1].plot(x2[fileNums[1]], y2[fileNums[1]], color='lightblue')
128 cp6 = axs2[0, 1].contour(X[fileNums[1]], Y[fileNums[1]], Z[fileNums[1
   ]], [cV[fileNums[1]]])
129 axs2[0, 1].set_title("Velocity Best Fit For expData" + fN[fileNums[1]] +
   ".mat")
130 axs2[0, 1].legend(["Best Fit Line", "Soliton Points"])
131 
```

```
132 axs2[1, 0].scatter(x[fileNums[2]], y[fileNums[2]], color='coral')
133 axs2[1, 0].plot(x2[fileNums[2]], y2[fileNums[2]], color='lightblue')
134 cp7 = axs2[1, 0].contour(X[fileNums[2]], Y[fileNums[2]], Z[fileNums[2
    ]], [cV[fileNums[2]]])
135 axs2[1, 0].set_title("Velocity Best Fit For expData" + fN[fileNums[2]] +
    ".mat")
136 axs2[1, 0].legend(["Best Fit Line", "Soliton Points"])
137
138 axs2[1, 1].scatter(x[fileNums[3]], y[fileNums[3]], color='coral')
139 axs2[1, 1].plot(x2[fileNums[3]], y2[fileNums[3]], color='lightblue')
140 cp8 = axs2[1, 1].contour(X[fileNums[3]], Y[fileNums[3]], Z[fileNums[3
    ]], [cV[fileNums[3]]])
141 axs2[1, 1].set_title("Velocity Best Fit For expData" + fN[fileNums[3]] +
    ".mat")
142 axs2[1, 1].legend(["Best Fit Line", "Soliton Points"])
143
144 fig2.canvas.set_window_title('Contour Slice with Best Fit')
145
146 yLimit = [241, 205, 199, 190]
147 xLimit = [105, 105, 105, 105]
148
149 i = 0
150 for ax in axs2.flat:
151     ax.set(xlabel='Vertical Distance (cm)', ylabel='Time (s)')
152     ax.set_ylim([0, yLimit[i]])
153     ax.set_xlim([0, xLimit[i]])
154     i += 1
155
156
157 # Analytical Plot Data
158 x_data = np.linspace(1, 9, 100)
159 y_data = sar_lam(x_data) * data[0].get('U0')[0][0]
160 y_plus_data = y_data + sar_der_lam(x_data)
161 y_minus_data = y_data - sar_der_lam(x_data)
162
163
164 # Experimental Plot Data
165 x_scatter = []
166 for i in range(len(cV)):
167     x_scatter.append(data[i].get('A0')[0][0])
168 y_scatter = Vel
169 y_error = Vel_error
170
171
172 # Opens a 4th figure to show analytical and experimental speed-amplitude
    relation
173 fig3, ax3 = plt.subplots()
174 # analytical function
175 ax3.plot(x_data, y_data, color='orange')
176 # analytical error
177 ax3.errorbar(x_scatter, ana_vel, xerr=ana_vel_x_error, yerr=
    ana_vel_y_error, color='red', fmt="o")
```

```
178 # Experimental data
179 ax3.errorbar(x_scatter, y_scatter, yerr=y_error, color='blue', fmt="o")
180
181 ax3.set(xlabel="Amplitude", ylabel="Speed (cm/s)")
182 ax3.set_title("Amplitude vs Speed")
183 ax3.legend(["Analytical Plot", "Analytical Error", "Experimental Data"])
184 fig3.canvas.set_window_title('Amplitude vs Speed')
185
186
187 # Shows all figures
188 fig.tight_layout()
189 fig2.tight_layout()
190 plt.show()
191
```

5 References

(1) : https://en.wikipedia.org/wiki/Soliton
(2) : https://en.wikipedia.org/wiki/John_Scott_Russell
(3) : "Soliton Analysis: An Adventure in Higher Order, Nonlinear PDE's", Applied Mathematics Department, University of Colorado Boulder, 2018