Fourier Project: Solitons

University of Colorado at Boulder

APPM 4350

Fourier Series and Boundary Value Problems

Authors:	*Affiliations:*
Aaron Sokolik	Physics
Jaden Wang	Applied Mathematics
Alexey Yermakov	Applied Mathematics

June 11, 2022

Contents

1 Introduction:

Linearization and analysis is a very powerful and useful approximation in many physical situations, yet this methodology can hide very important phenomena. Linear analysis is useful when the nonlinear effects, which are always present, have a very minor impact on the system. When the nonlinear effects are relatively sized compared to the linear, other behaviors appear which linear analysis obscures. Solitons, or solitary waves are such a phenomena and can be understood as occurring when nonlinear effects balance the linear effect of dispersion. This can arise as a result of very high symmetry in the nonlinear equations of these systems such as is in the Korteweg-de Vries equation and which also applies to the conduit equation provided. In this paper we don't explicitly demonstrate the symmetry of these systems mathematically; however, the time-dependent, highly consistent speed and shape of the wave we have in the experimental data should adequately demonstrate the system's symmetry.

Since these effects are important and manifest in many physical systems, we chose to focus on a nonlinear project to gain some familiarity with the solution technique of applying traveling waves. Besides fluid conduits, we've encountered solitons in the analysis of quantum magnetic systems where they could appear as propagating spin waves in 1D Heisenberg spin chains which is a very active area of research. They are also exploited in some current commercial fiber-optic transmission schemes where high-throughput, long-distance links are needed and are proposed as part of transmission standards making up the backbone links in proposals for "5G" telecom networks. Solutions also appear in field theories and nonlinear optical phenomena, where their properties are also exploited.

Our derivation of the solution for the conduit equation and analysis of the experimental data confirm the unique properties distinguishing solitons from dispersive waves. They are extremely stable in shape and speed through time and are fully described by their time-independent amplitude parameter. We believe that further work and insights in the nature of solitons such as their non-interaction could be seen in expanding this analysis to a more complicated experimental system as well as completing a derivation of the conduit equation. Consideration of classifications of nonlinear PDE systems (hyperbolic, geometries...) for which traveling wave solutions generally apply is beyond our work. Below we present our solution derivation and experimental analysis.

2 Mathematical Model:

A wave in a vertical conduit can be described by the following PDE:

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

where the function $A(z,t)$ describes the non-dimensionalized cross-sectional area of the wave at location z and time t. Furthermore, the PDE satisfies the following boundary conditions:

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

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

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

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

Note that (2)–(4) tell us the non-dimensionalized area eventually becomes a constant, and (5) describes that the wave was generated with initial amplitude a and slope 0 because it's a local maximum with respect to z.

If we assume that soliton is a special solution to this PDE, since we know that the soliton is a traveling wave in the $+z$ direction, we can convert this PDE into an ODE using change of variables $\zeta = z - ct$, where c is an unknown constant that represents traveling speed. Thus we let $A(z,t) = f(\zeta)$ and by using the Chain rule we obtain the following boundary value problem:

$$-cf' + (f^2)' - (f^2(-cf^{-1}f')')' = 0 \tag{6}$$

$$\lim_{\zeta\to\pm\infty} f(\zeta) = 1 \tag{7}$$

$$\lim_{\zeta\to\pm\infty} f'(\zeta) = 0 \tag{8}$$

$$\lim_{\zeta\to\pm\infty} f''(\zeta) = 0 \tag{9}$$

$$f(0) = a, f'(0) = 0 \tag{10}$$

where the prime notation is short for $\frac{d}{d\zeta}$.

We aim to reduce this ODE to first order in order to solve numerically. Notice since all terms on LHS are derivatives with respect to ζ, we can integrate both sides with respect to ζ and obtain

$$\begin{aligned} -cf + f^2 - f^2(-cf^{-1}f')' &= D \\ -cf + f^2 - cf^2f^{-2}f' + cf^2f^{-1}f'' &= D \\ -cf + f^2 - cf' + cff'' &= D \end{aligned} \tag{11}$$

We can find D by letting $\zeta \to \infty$ and applying the BCs:

$$\begin{aligned} -c + 1 - 0 + 0 &= D \\ D &= 1 - c \end{aligned} \tag{12}$$

Then the ODE becomes

$$-cf + f^2 - cf' + cff'' = 1 - c \tag{13}$$

To obtain a first order ODE, we need to multiply the integrating factor $f^{-3}f'$ on both side and integrate:

$$\begin{aligned} -cf^{-2}f' + f^{-1}f' - cf^{-3}f'^2 + cf^{-2}f'f'' &= (1-c)f^{-3}f' \\ cf^{-1} + \ln f + \frac{1}{2}cf^{-2}f'^2 &= \frac{1}{2}(c-1)f^{-2} + B \end{aligned} \tag{14}$$

Note that since $\forall\ \zeta, f > 0$ we do not need absolute value signs for $\ln f$.

Again we take $\zeta \to \infty$ and apply the BCs to find the constant B:

$$\begin{aligned} c + 0 + 0 &= \frac{1}{2}(c-1) + B \\ B &= \frac{1}{2}(c+1) \end{aligned} \tag{15}$$

And the ODE becomes:

$$\begin{aligned} cf^{-1} + \ln f + \frac{1}{2}cf^{-2}f'^2 &= \frac{1}{2}(c-1)f^{-2} + \frac{1}{2}(c+1) \\ cf + f^2\ln f + \frac{1}{2}cf'^2 &= \frac{1}{2}(c-1) + \frac{1}{2}(c+1)f^2 \\ g(\zeta, f) \coloneqq f' &= \pm\sqrt{\left(\frac{c+1}{c} - \frac{2}{c}\ln f\right)f^2 - 2f + \frac{c-1}{c}} \end{aligned} \tag{16}$$

It remains to find the constant c using (10). At $\zeta = 0$:

$$\begin{aligned} cf(0) + f(0)^2\ln f(0) + \frac{1}{2}cf(0)'^2 &= \frac{1}{2}(c-1) + \frac{1}{2}(c+1)f(0)^2 \\ ca + a^2\ln a + 0 &= \frac{1}{2}(c-1) + \frac{1}{2}(c+1)a^2 \\ ac - \frac{1}{2}c - \frac{1}{2}a^2c &= -\frac{1}{2} + \frac{1}{2}a^2 - a^2\ln a \\ (2a - 1 - a^2)c &= -1 + a^2 - 2a^2\ln a \\ c &= \frac{2a^2\ln a - a^2 + 1}{(a-1)^2} \end{aligned} \tag{17}$$

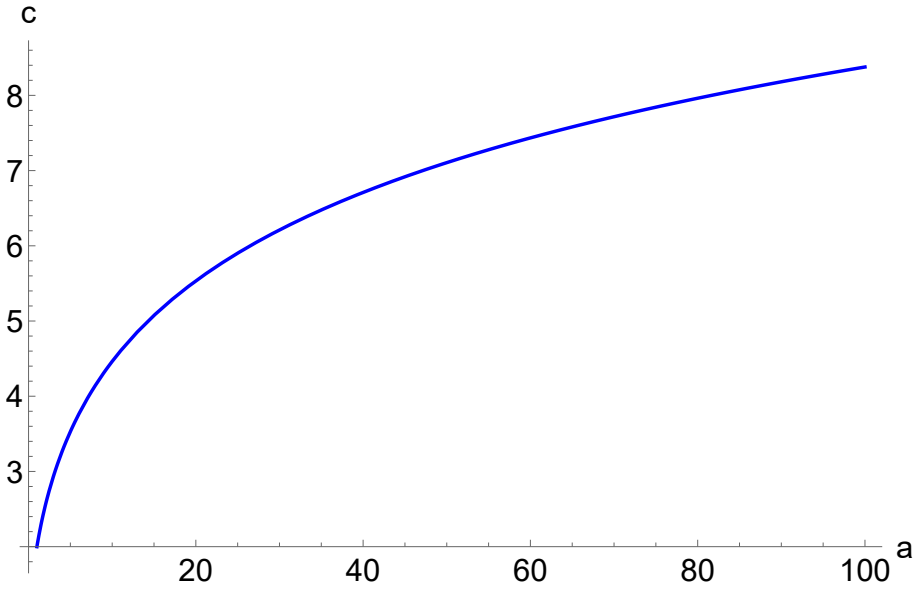


Figure 1: Plot of the theoretical speed-amplitude relation for $a \geq 1$.

The graph of $c(a)$ (Figure 1) for $a \geq 1$ indicates that as the ratio of the cross-sectional area of the soliton and the undisturbed cross-sectional area of the conduit increases, the speed at which the soliton is moving also increases; however, the increase in non-dimensional speed is non-linear and decreases as the ratio a increases. This may be because as the bulge of the soliton grows bigger, it should experience a larger upward buoyancy that increases its speed, yet it also experiences more linear drag proportional to the speed from the surrounding fluid that counteracts this upward force[OC86; LH13]. Thus, the rate of growth in speed slows down with an increase in the ratio a.

3 Experimental Data

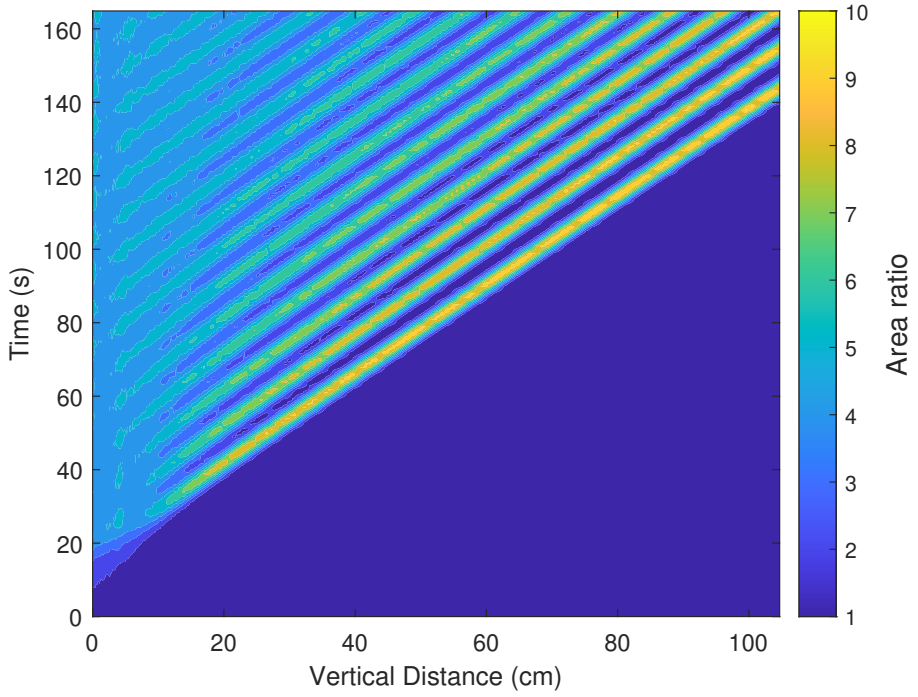


Figure 2: Contour plot of the 48th experiment.

To extract the speed from each data set, we followed this method:

1. Identify the leading edge of the contour plot
2. Select a point along the leading edge near the beginning of the experiment as close to the maximum area ratio as possible.
3. Record the values of the Vertical Distance and Time for this point.
4. Select a point along the leading edge near the end of the experiment as close to the maximum area ratio as possible.
5. Record the values of the Vertical Distance and Time for this point.
6. Calculate the difference between the Vertical Distance of the two points and divide it by the difference between the Time of the two points. This will yield the experimental speed in cm/s.

   Note: Data sets 7 and 35 did not have strictly increasing time vectors, so we did not use those data sets.

We calculated the Experimental Speed by solving for $c(A_0) \cdot U_0$, where A_0 and U_0 were provided for each data set.

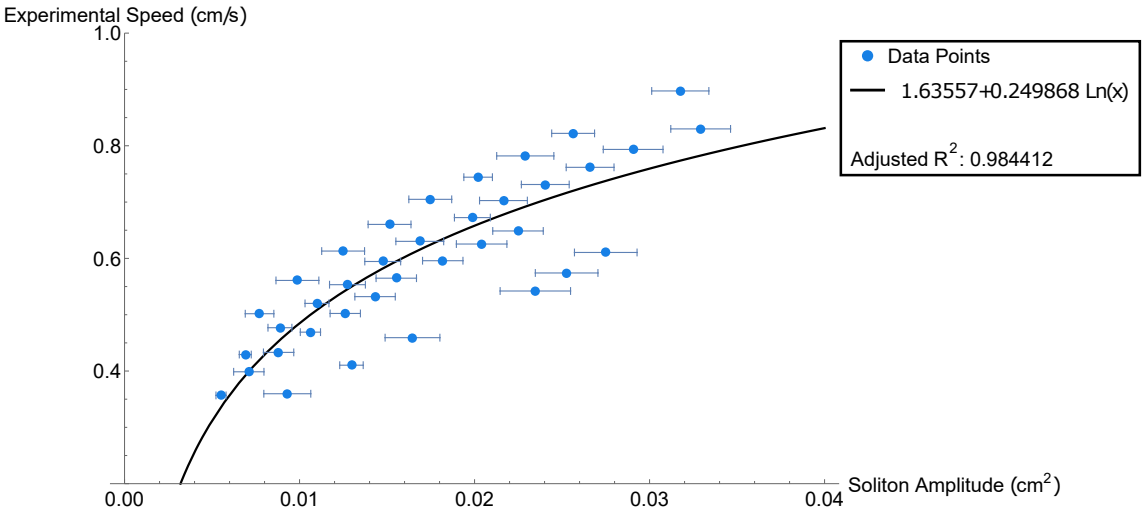


Figure 3: Plot of soliton amplitude and experimental.

From Figure 3 one can see that solitons with larger amplitudes tend to move at a larger velocity than those with smaller amplitudes, which matches our theoretical results (this answers 4.1).

If we look at the theoretical speed-amplitude equation (17), we see that the numerator and denominator both contain quadratic terms of a. Asymptotically, we expect these quadratic terms to cancel each other and leave the speed-amplitude relation to be dominated by the logarithmic term of a. Using this information, we created a logarithmic line of best fit for the experimental speed-amplitude data. This produces an adjusted R^2 of about 0.984, which further supports our theoretical derivation since 98.4% of the variability in the experimental speed can be accounted by the logarithmic regression and experimental amplitude.

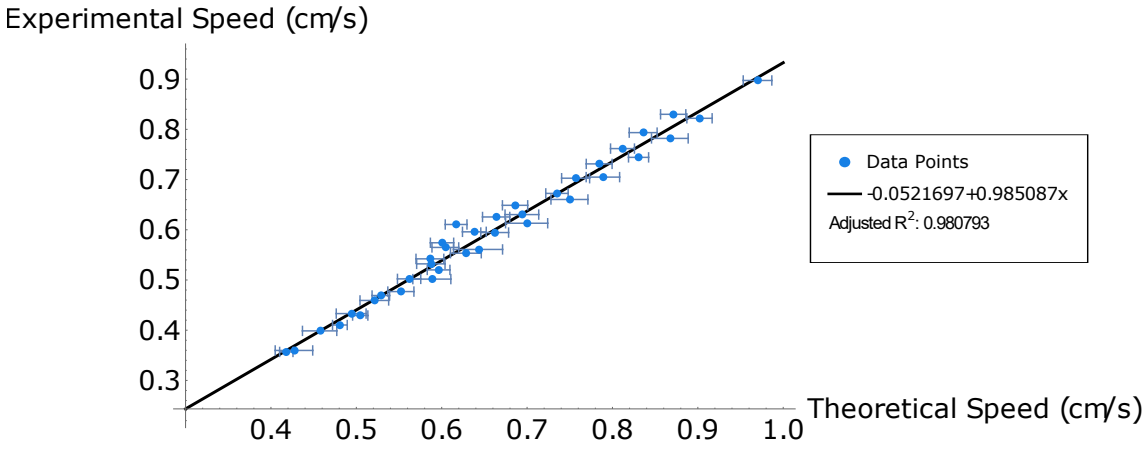


Figure 4: Plot of the theoretical speed-amplitude relation c(a).

From the R^2 value in Figure 4, we conclude that there is a strong linear correlation between our experimental speed and theoretical speed. In particular, 98.1% of the variability in the experimental speed can be accounted by the linear regression and theoretical speed. This is expected since an increase in theoretical speed should correspond to an increase in experimental speed for a given soliton amplitude (this answers 4.2).

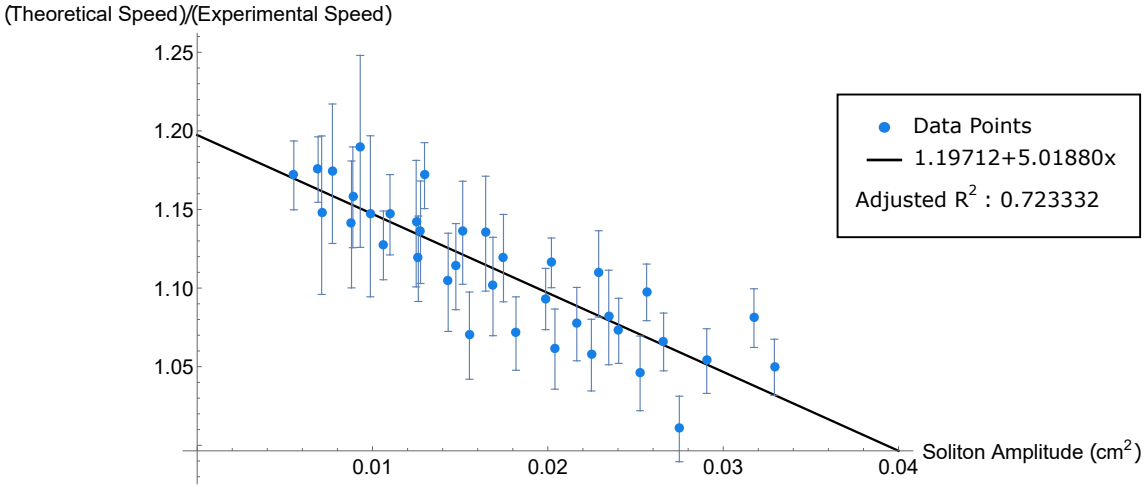


Figure 5: Plot of the Speed Ratio and Soliton Amplitude (Linear fit).

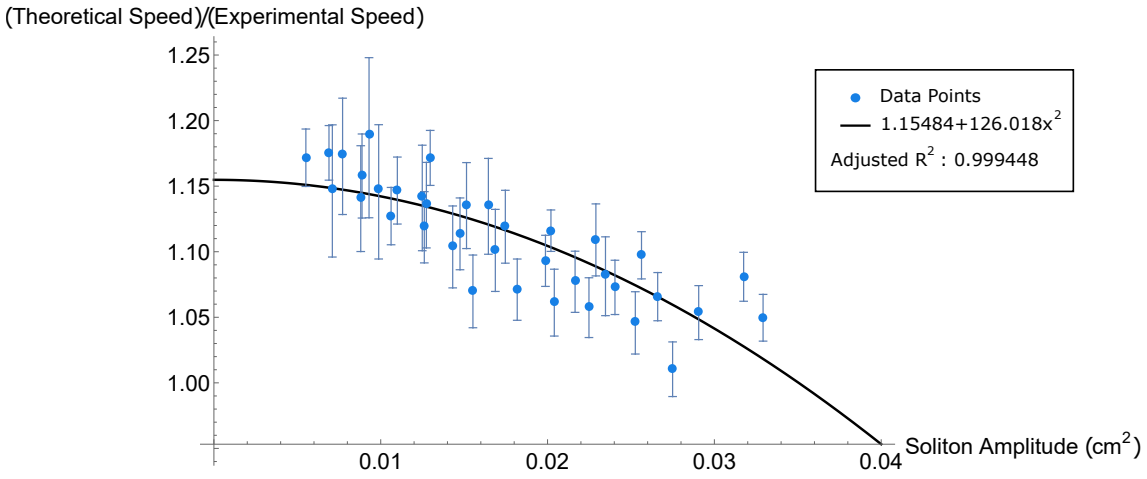


Figure 6: Plot of the Speed Ratio and Soliton Amplitude (Quadratic fit).

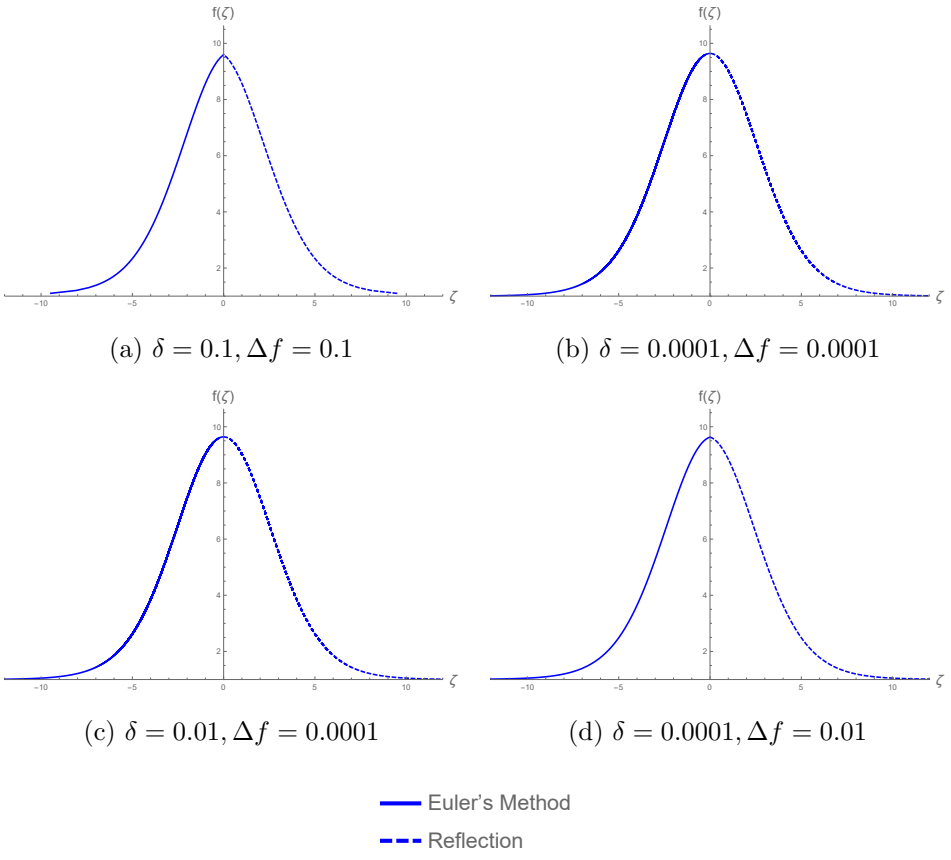
From Figures 5 and 6 we see that the theoretical speed is larger than the experimental speed in all data sets, since the ratio of theoretical speed and experimental speed is always greater than 1. We also see that the soliton amplitude is a good fit for predicting the ratio of theoretical speed and experimental speed due to an R^2 of 0.786. In particular, 78.6% of the variability in the ratio of theoretical speed and experimental speed can be accounted by the linear regression and soliton amplitude. Furthermore, as the soliton amplitude increases the speed ratio moves closer to 1. Using this information we can hypothesize that the theoretical model excludes properties that are present in the experiment. For one, we note that the theoretical speed is always larger than the experimental speed meaning that the theoretical model excludes some opposing forces.

Intuitively, we'd expect the ratio of theoretical and experimental speed to approach 1 asymptotically. Using this intuition, we created a quadratic line of best fit and expected the function to be concave up. An interesting observation is that Figure 6 the quadratic fit is concave down. As soliton amplitude increases, we are moving towards a speed ratio of 1 at a faster rate, which is not what we expected. Possible reasons for this behavior are that we don't have enough datasets to reliably determine the behavior of the speed ratio and amplitude relation. Another reason is we don't have any real reason to believe that the line of best fit would be quadratic, if anything it would be asymptotic (such as $\frac{1}{x}$).

4 Numerical Results:

4.1 Euler's method

Performing Euler's method to approximate $f(\zeta)$ with varying step-sizes and tolerances, we obtain the following figures:



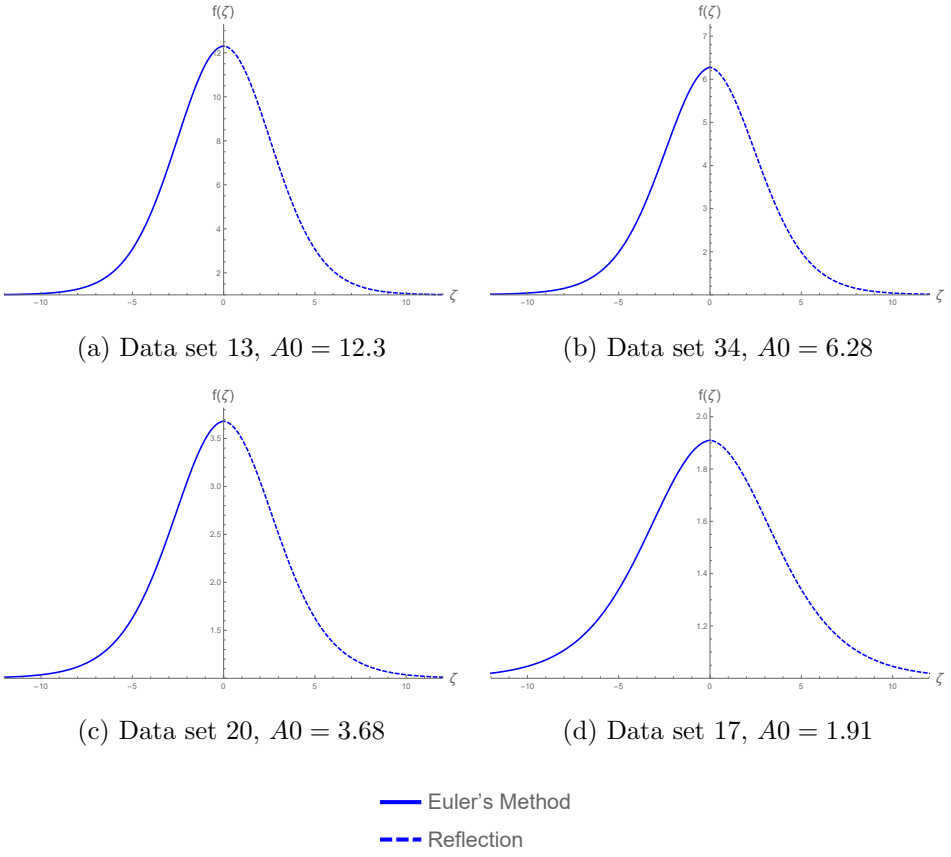
(a) $\delta = 0.1, \Delta f = 0.1$

(b) $\delta = 0.0001, \Delta f = 0.0001$

(c) $\delta = 0.01, \Delta f = 0.0001$

(d) $\delta = 0.0001, \Delta f = 0.01$

Figure 7: Numerical solution of soliton profiles with $A_0 = 9.64$ and varying parameters (Euler's Method).



(a) Data set 13, $A0 = 12.3$

(b) Data set 34, $A0 = 6.28$

(c) Data set 20, $A0 = 3.68$

(d) Data set 17, $A0 = 1.91$

Figure 8: Numerical solution of soliton profiles with $\delta = 0.001, \Delta\zeta = 0.001$ and varying data sets and initial amplitude.

4.2 Runge-Kutta Order 4

Euler's method only exploits the first-order information of the function and has local truncation error of $\mathcal{O}(h)$, which can lead to relatively large errors when the second-order term is large, as is in the case of a soliton without inversion. By using Runge-Kutta Order 4 (RK4), we can exploit up to fourth-order information of the function. The truncation error is only $\mathcal{O}(h^4)$, therefore we no longer need inversion to obtain accurate results. Note that RK4 doesn't require computing the expensive fourth-order Taylor expansion; instead it approximates such expansion using the values of the ODE at 4 well-chosen points. Below we present the algorithm [BF11].

Algorithm 1 Runge-Kutta Order 4

Require: ODE $g(\zeta, f)$, endpoints a, b, number of intervals N, initial condition ζ_0, A_0.

$h = (b-a)/N$; $\zeta = \zeta_0$; $f = A_0$.

for $i = 1, 2, \ldots, N$ **do**

  $K_1 = hg(\zeta, f)$;

  $K_2 = hg(\zeta + h/2, f + K_1/2)$;

  $K_3 = hg(\zeta + h/2, f + K_2/2)$;

  $K_4 = hg(\zeta + h, f + K_3)$;

  $f = f + (K_1 + 2K_2 + 2K_3 + K_4)/6$;

  $\zeta = \zeta_0 + ih$;

end for

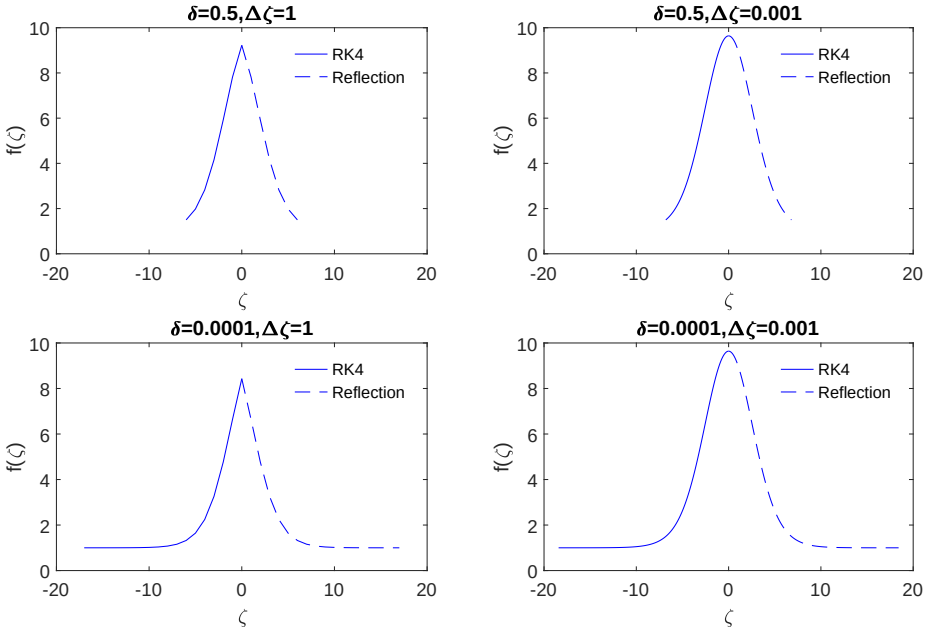


Figure 9: Numerical solution of soliton profiles with $A_0 = 9.64$ and varying parameters (RK4).

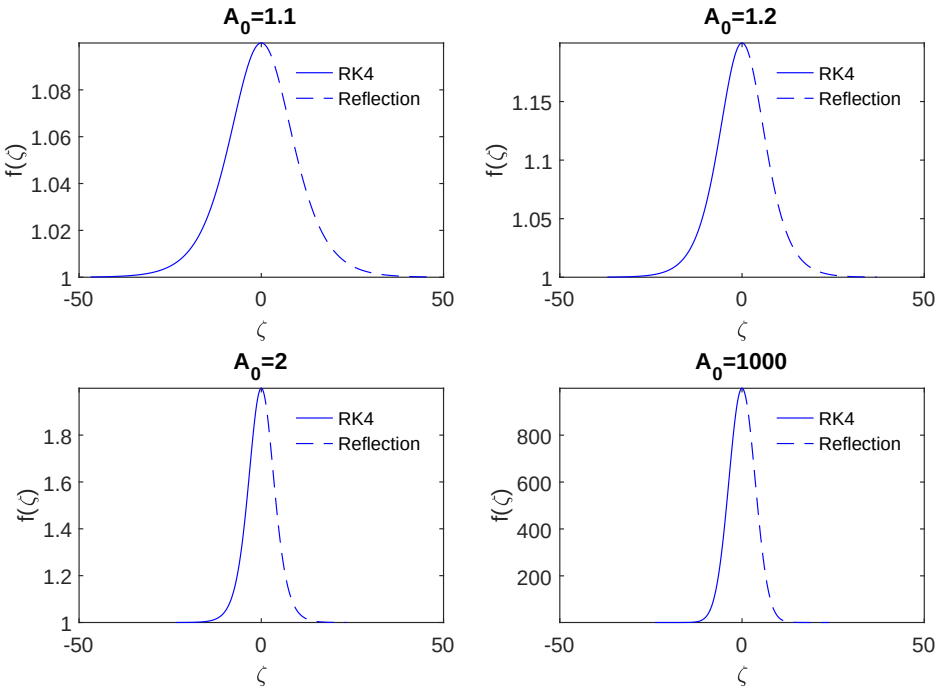


Figure 10: Numerical solution of soliton profiles with $\delta = 0.0001, \Delta\zeta = 0.001$ and varying initial amplitudes.

5 Discussion:

Note: This project analyzes solitons so we will not be comparing Euler's method with RK4.

5.1 Numerical Results

5.1.1 Algorithm Parameters

Based on Figures 7 and 9 we can see that larger step sizes result in sharper peaks and larger δ gives rise to an incomplete profile. The results match our expectation. Since there is significant change of concavity or the second-derivative near the peak, using a large step size prevents the solution to adjust to this change. As we reduce the step size to something reasonable (*e.g.* 0.01), the sharpness at the peak nearly vanishes. Beyond this point, as the step size goes to zero, we see that there isn't a noticeable change in the soliton profile, implying that the profile converges.

Moreover, since the value of f is 1 prior to the release of the soliton, by setting the initial value of f to $1 + \delta$ we cut off the part of the profile where $1 \leq f < 1 + \delta$. Therefore, a larger value of δ implies a larger cutoff which makes the profile appear incomplete. Moreover, if we choose a

$\delta \ll \Delta\zeta$, then (16) becomes

$$f' \approx \pm\sqrt{\left(\frac{c+1}{c} - \frac{2}{c}\ln 1\right)\cdot 1^2 - 2 + \frac{c-1}{c}} = 0.$$

Since a relatively large step size is taken at near 0 slope, the profile would grow much more slowly than expected and the numerical results would significantly deviate from the true solution or even remain flat if δ is sufficiently small.

5.1.2 Initial Amplitude

By varying the initial amplitude, Figures 8 and 10 show that initial amplitude can drastically change both the width and height of the soliton profile. For an initial amplitude very close to 1, as is the case in Figure 8 (d) and Figure 10 (a), clearly the amplitude only goes as high as the initial amplitude, yet the width is much larger than the cases with slightly larger initial amplitudes. Interestingly, notice that the profiles of $A_0 = 2$ and $A_0 = 1000$ in Figure 10 almost have the same width and shape but differ in height, suggesting that A_0 only affects the height but not the width or shape of the soliton after becoming sufficiently large. Experimentally, this implies that assuming the conditions to generate a soliton holds, and given a sufficiently large vertical tube, as we pump the interior liquid at a very fast rate, we should be able to generate a bulge that looks like a disc with cross-sectional area as large as we want but still has a constant, small thickness. In reality, this is unlikely to happen because the assumptions we made for our conduit equation to be valid, such as negligible diffusion between the interior and exterior liquids, would likely break down as the surface of contact grows large.

5.1.3 The Experimental Solitons

Our analysis of the experimental data confirms the unique characteristics of solitons apparent from the traveling wave solution (16,17). We see a wave shape and speed which do not evolve in time, besides traveling, and which are determined by a constant amplitude input. Besides analyzing speed using the slope of our contour plots, we algorithmically extracted the usable soliton profiles from the datasets of which we present sixteen profile sets below.

The algorithm works by detecting the edges of the leading disturbance in the contour plots by analyzing gradients of the "Amat" matrix, providing us a plot of those edges as a sanity check and then extracting the soliton profile "slices" in time. The three plots below show the important steps of detection, extraction and alignment, and fitting.

Below we present the extractions along with the plots of our numerical solutions (RK4) based upon the provided amplitude and amplitude error data. We display error curves for a 95% confidence interval around the 6th-degree polynomial fits. All data below is presented with arbitrary units to match the numerical results.

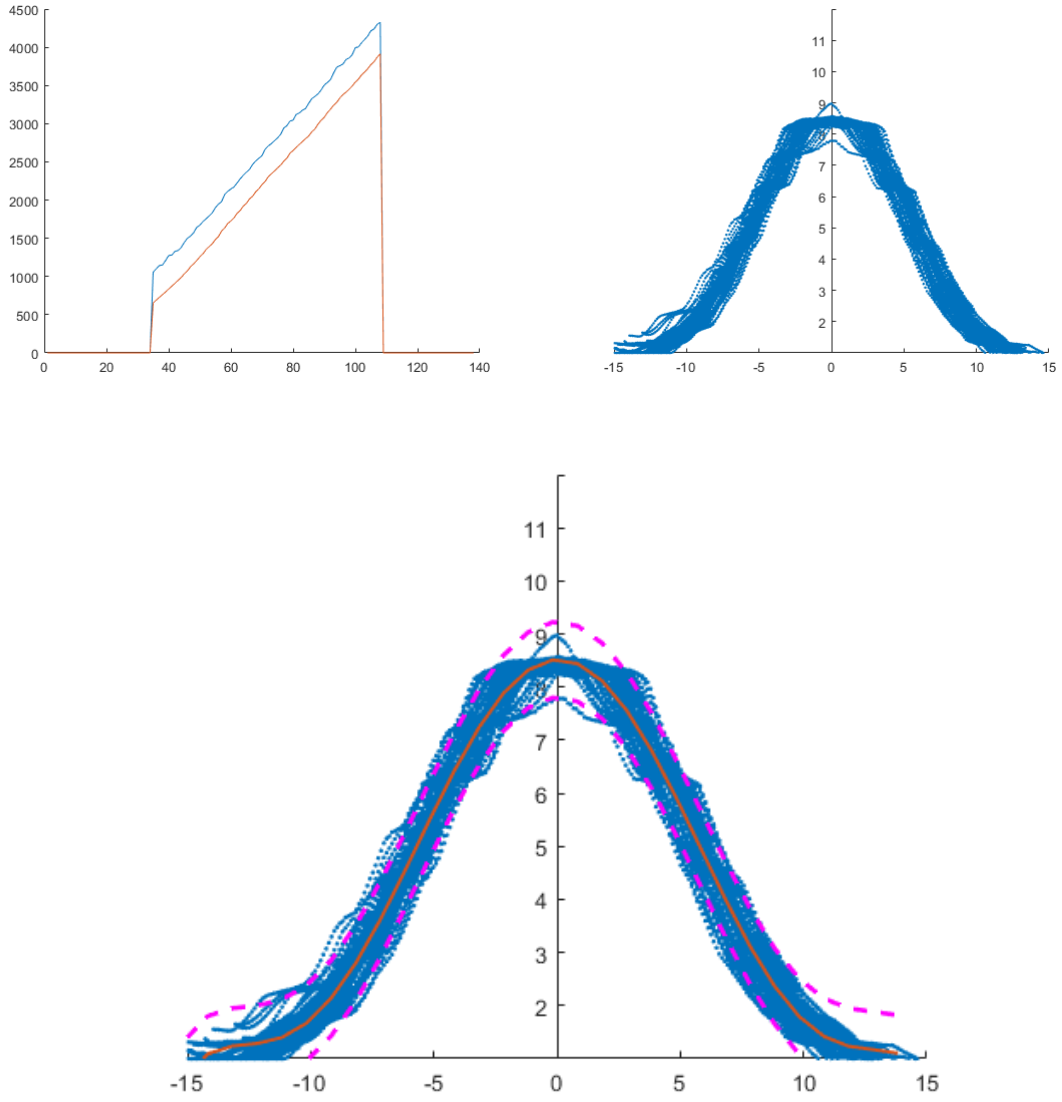


Figure 11: The algorithmic steps to extraction and fitting of non-dimensional soliton profile data. Our fit is a 6th-degree polynomial with R^2 of at least 96%

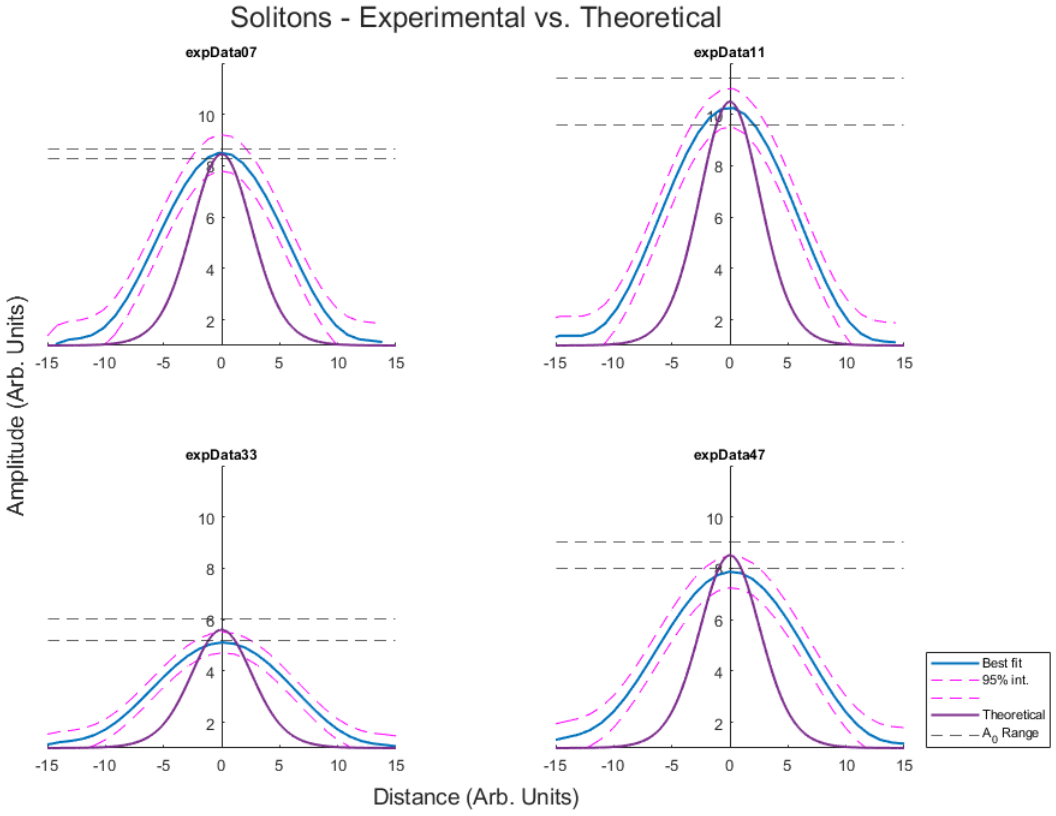


Figure 12: Highlight of extraction and fitting of four datasets with varying amplitudes. Amplitudes extracted underestimate the proper amplitude and the curves are wider and less steep than solitons of the conduit equation.

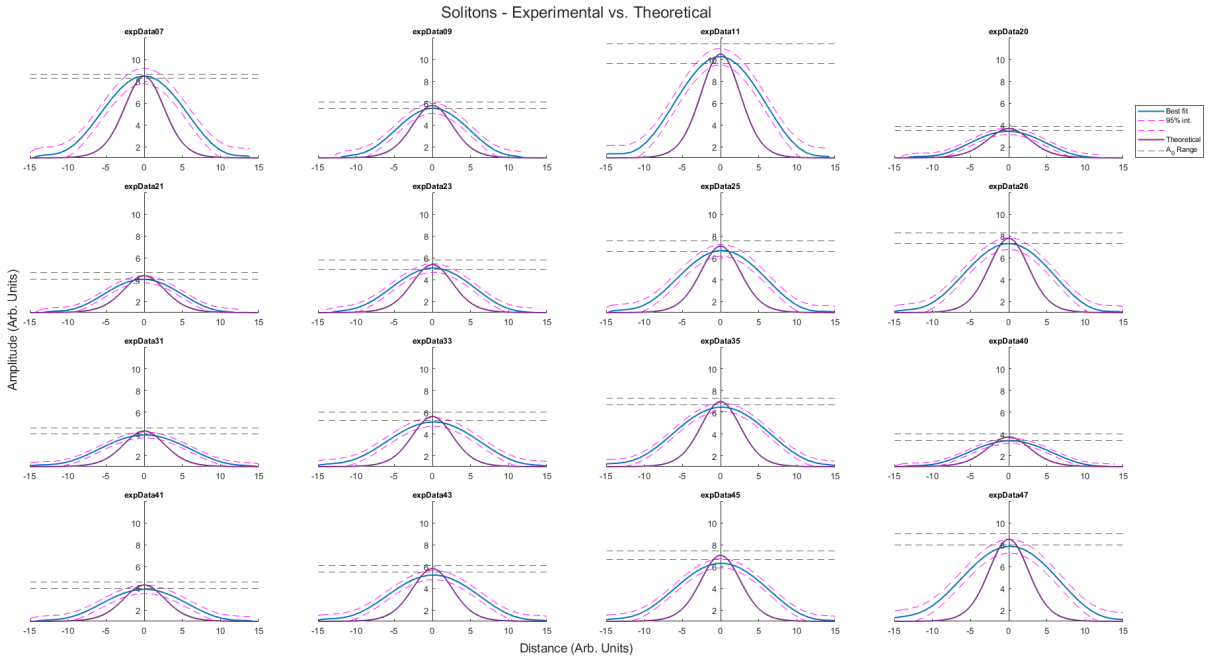


Figure 13: Demonstration of algorithm stability and of results across many datasets

As in our results for the speed derivation, it is clear that the provided data does not properly model all of the features of the soliton but is close enough for our purposes. The amplitude we extracted is consistently lower than that of the correct, provided amplitude and the difference is statistically significant as we are normally at or below the lower error bound on the provided amplitude. As a result, our line-shape is a poor fit to the numerical solution to the conduit equation. Overall though we still see soliton characteristics in the data (this among several other places answers 4.3).

5.2 Resolving the Discrepancies

Our results show consistent systematic deviation between the theoretical predictions and experimental data. Looking back at Figure 5 we see that the theoretical speed was consistently larger than the experimental speed. In addition, Figure 13 showed that for the data provided, our theoretical results had consistently skinnier soliton profiles with larger amplitudes. These two pieces of information together show that the experimental data provided had solitons that had smaller amplitudes and wider profiles that resulted in slower speeds. This deviation can be explained by the experimentation that was used to collect the data provided. Looking at Figure 2, we see that the experiments had multiple waves propagating through the conduit, but we were told to just ex-

amine the leading wave, assuming that it is a soliton. Having multiple waves does not make sense for this experiment, since the trailing waves could result in inconsistent data through interference. Therefore we suspect that the data set was not generated specifically for this project resulting in data that doesn't align with our theoretical model of solitons.

Post Script: This project was very timely for lab research conducted by one the group members and the other members were generous enough to agree to work on this topic. Thank you.

References

[OC86] Peter Olson and Ulrich Christensen. "Solitary wave propagation in a fluid conduit within a viscous matrix". In: *Journal of Geophysical Research: Solid Earth* 91.B6 (1986), pp. 6367–6374.

[BF11] RL Burden and JD Faires. *Numerical Analysis*. 9th ed. Brooks/Cole, Cencage Learning, 2011.

[LH13] Nicholas K Lowman and Mark A Hoefer. "Dispersive hydrodynamics in viscous fluid conduits". In: *Physical Review E* 88.2 (2013), p. 023016.

Appendix:

Code:

```
%Alexey's code
% GIVEN
mu_i = 51/100; %+- 1/100
mu_e = 1200/100; %+- 20/100
rho_i = 1.2286; %+- .0001
rho_e = 1.2587; %+- .0001

% WHAT IS GRAVITY?
g = 9.80665*100;
disp("Assuming g = " + g);
disp(" ");

% SPEED FUNCTION
SpeedFunc1 = @(a) (a.^2 - 2 * a.^2 * log(a) - 1)/(2 * a - a^2 - 1);

% SOLVING FOR NECESSARY VALUES
disp("VALUES:");
epsilon = mu_i/mu_e;
disp("epsilon: " + epsilon);

delta = rho_e - rho_i;
disp("delta: " + delta);

alpha = (((2.^7)*mu_i)/(pi * g * delta)).^(1/4);
disp("alpha: " + alpha);

R_0 = sqrt((8 * U0 * mu_i)/(g * delta));
disp("R_0: " + R_0);

Q = ((2 * R_0)/(alpha)).^4;
disp("Q: " + Q);
disp(" ");

% SOLVING FOR SCALES
disp("SCALES:");
disp("Speed Scale (U): " + U0);
```

```
L = R_0 / (sqrt(8));
disp("Radial Length Scale (L): " + L);

L_2 = L / sqrt(epsilon);
disp("Vertical Length Scale (L_2): " + L_2);

T = (sqrt(8) * mu_i)/(g * R_0 * delta);
disp("Time Scale (T): " + T);

disp("Scaled A0: " + A0 * L);

% SOLVING FOR SPEED
disp("Speed (non-scaled): " + SpeedFunc1(A0));
disp("Speed (scaled): " + SpeedFunc1(A0*L)*U0);

% PRINT A PLOT
[M,c] = contourf(z_vec, t_vec, Amat);
c1 = colorbar;
c1.Label.String = "Area ratio";
c1.Label.FontSize = 12;
xlabel("Vertical Distance(cm)");
ylabel("Time (s)");
c.LineStyle = 'none';

%Jaden's code
c_func = @(a) (a^2 - 2*a^2 * log(a) -1)/(2*a - a^2 -1 );
c = c_func(A0); %A0 is the initial amplitude
s = c*U0; %speed with dimension using U0 as speed
f=@(x,y) sqrt(((c+1)/c -2/c * log(y))*y^2- 2*y + (c-1)/c); %autonomous ODE

%subplots with different parameters
fig1=figure;
epsilons=[0.5,0.5,1e-4,1e-4];
Ns = [1e2,1e5,1e2,1e5];
for j=1:4
epsilon = epsilons(j);
f0 = 1+ epsilon;
a=-100;
```

```
b=0;
N =Ns(j);
[zeta,W] = RK4(f,f0,a,b,N);
for i=1:size(W,1)
    if ~isreal(W(i))
        break
    end
end
zeta_l = zeta-zeta(i-1);
zeta_l = zeta_l(1:i-1);
zeta_r = -flip(zeta_l);
W_l = W(1:i-1);
W_r = flip(W_l);

subplot(2,2,j)
plot(zeta_l,W_l,'b')
hold on
plot(zeta_r,W_r,'b--')
xlim([-20,20]);
ylim([0,10]);
title(['\delta=' num2str(epsilon) ',\Delta\zeta=' num2str((b-a)/N)])
ylabel('f(\zeta)')
xlabel('\zeta')
leg=legend({'RK4','Reflection'},'Location','northeast','FontSize',5);
legend('boxoff')
leg.ItemTokenSize = [16,9];
hold off
end

%subplots with different initial amplitudes
fig2=figure;
A0s = [1.1,1.2,2,1000];
for j=1:4
c = c_func(A0s(j));
f=@(x,y) sqrt(((c+1)/c -2/c * log(y))*y^2- 2*y + (c-1)/c);
epsilon=1e-4;
N =1e5;
f0 = 1+ epsilon;
a=-100;
```

```
b=0;
[zeta,W] = RK4(f,f0,a,b,N);
for i=1:size(W,1)
    if ~isreal(W(i))
        break
    end
end
zeta_l = zeta-zeta(i-1);
zeta_l = zeta_l(1:i-1);
zeta_r = -flip(zeta_l);
W_l = W(1:i-1);
W_r = flip(W_l);

subplot(2,2,j)
plot(zeta_l,W_l,'b')
hold on
plot(zeta_r,W_r,'b--')
xlim([-50,50]);
ylim([1 inf])
title(['A_0=' num2str(A0s(j))])
ylabel('f(\zeta)')
xlabel('\zeta')
leg=legend({'RK4','Reflection'},'Location','northeast','FontSize',5);
legend('boxoff')
leg.ItemTokenSize = [16,9];
hold off
end

%Runge-Kutta Order 4
function [eta,w]=RK4(f,fiv,a,b,N)
    eta = linspace(a,b,N+1)';
    w = zeros(N+1,1);
    k = zeros(4,1);
    h=(b-a)/N;
    w(1) = fiv;
    for i = 1:N
        wi=w(i);

        k(1)=h*f(eta(i),wi);
```

```
        k(2)=h*f(eta(i)+h/2,wi+0.5*k(1));

        k(3)=h*f(eta(i)+h/2,wi+0.5*k(2));

        k(4)=h*f(eta(i)+h,wi+k(3));

        w(i+1)=wi+1/6*(k(1)+2*k(2)+2*k(3)+k(4));
    end
end

%Aaron's Code

%%%%%%%%%%%%
% extract.m %
%%%%%%%%%%%%

clear;

%tune these search parameters for each dataset if needed. Tuning for
%boundaries and gradients.
%inputs array, (lower bounds, upper bounds, x_search_min, )
% in(:,:,1) = ["expData07", 110, 222, 1350, 400, 0.0405, 16];
% in(:,:,2) = ["expData09", 117, 250, 1400, 400, 0.05, 17];
% in(:,:,3) = ["expData11", 80, 158, 1400, 400, 0.0405, 20];
% in(:,:,4) = ["expData20", 75, 135, 1600, 400, 0.0077, 11];
% in(:,:,5) = ["expData21", 58, 140, 1700, 400, 0.05, 17];
% in(:,:,6) = ["expData23", 62, 140, 1500, 400, 0.0405, 20];
% in(:,:,7) = ["expData25", 49, 108, 1700, 400, 0.0405, 20];
% in(:,:,8) = ["expData26", 44, 115, 1100, 400, 0.04, 20];
% in(:,:,9) = ["expData31", 40, 108, 1200, 400, 0.0405, 20];
% in(:,:,10) = ["expData33", 35, 106, 950, 400, 0.0405, 20];
% in(:,:,11) = ["expData35", 42, 97, 1400, 400, 0.0405, 20];
% in(:,:,12) = ["expData40", 65, 125, 900, 300, 0.01, 14];
% in(:,:,13) = ["expData41", 50, 130, 1200, 400, 0.0405, 20];
% in(:,:,14) = ["expData43", 32, 110, 700, 400, 0.0405, 20];
% in(:,:,15) = ["expData45", 32, 95, 1000, 400, 0.0405, 20];
% in(:,:,16) = ["expData47", 35, 108, 900, 400, 0.0405, 20];
```

```
in(:,:,1) = ["expData07", 110, 222, 1350, 400, 0.0405, 20];
in(:,:,2) = ["expData11", 80, 158, 1400, 400, 0.0405, 20];
in(:,:,3) = ["expData33", 35, 106, 950, 400, 0.0405, 20];
in(:,:,4) = ["expData47", 35, 108, 900, 400, 0.0405, 20];

outputFolder = "C:\Users\Aaron Sokolik\Dropbox\CU\APPM
    4350\Experimental_data_soliton_project\plots";
dataDirectory = dir("../Experimental_data_soliton_project/*.mat");

load('theoretical_results.mat');

t = tiledlayout(2,2);
xlabel(t, 'Distance (Arb. Units)', 'FontSize', 14);
ylabel(t, 'Amplitude (Arb. Units)', 'FontSize', 14);
title(t, "Solitons - Experimental vs. Theoretical", 'FontSize', 18);

amplitudes = zeros(length(in));

for I = 4:4 %1 : length(in)

    [rr1, rr2, ff1, ff2, xx_max, xx_min, err_up, err_dn, delta] =
        characterizeSolitons(in(:,:,I));
    %ax = nexttile;

    file_name = char(in(1,1,I));
    file_num = str2double(file_name(end-1:end));

    hold on;
    %plot(xx_max);
    %plot(xx_min);

    %Make plots
    %f = fit(rr1, rr2, 'gauss2')
    %plot(f,rr1,rr2);
    scatter(rr1, rr2, '.');
    ax=gca;
    ax.YAxisLocation='origin';
    ylim([1 12]);
    xlim([-15 15]);
```

```
    plot(ax, ff1, ff2, 'LineWidth', 1.5);
    plot(ax, ff1, ff2+2*delta,'m--', ff1 , ff2-2*delta,'m--');

    data_idx = find([data{:,1}] == file_num);

    if ~isempty(data_idx)
        all_plot_data = data{data_idx, 2};
        plot(all_plot_data(:,1), all_plot_data(:,2), 'LineWidth', 1.5);
    end

    yline(ax, err_up, '--');
    yline(ax, err_dn, '--');

    title(string(in(1,1,I)), 'FontSize', 9);

    if I == 4
        legend(ax, {'Best fit','95% int.','','Theoretical','A_{0} Range'}, 'FontSize', 8,
            'Location','southeastoutside','Orientation','Vertical');
    end

    %disp(kdv_sol(8, ff1, 0))

    %plot(ff1, kdv_sol(8.5, ff1, 0));

    amplitudes(I) = max(ff2);

end

%kdv soliton solution - force fit...
% function Aout = kdv_sol(A0, x, t)
%     Aout = A0./cosh(1/2*sqrt(2*A0).*(x/15-A0*2*t)).^2;
% end

%%%%%%%%%%%%%%%%%%%%%%%%%
% characterizeSolitons.m %
%%%%%%%%%%%%%%%%%%%%%%%%%

function [r1, r2, f1, f2, x_max, x_min, err_up, err_dn, delta] =
    characterizeSolitons(inputs)
```

```
load(append("../Experimental_data_soliton_project/", inputs(1)), 'Amat', 'A0',
    'A0_error', 'z_vec', 'U0');

%get gradients, use this as the quiver plots are very well defined.
[xg,yg] = gradient(Amat);

%define search parameters
t_bounds = [str2double(inputs(2)), str2double(inputs(3))]; %must provide for each
    dataset
x_search_min = str2double(inputs(4));
max_width = str2double(inputs(5));
yg_threshold_width = str2double(inputs(6));
run_sum = str2double(inputs(7)); %density threshold to confirm edge

xypadding = 0;

s = [1,4350];
Amat_dims = size(Amat);

x_max = zeros(1,Amat_dims(1));
x_min = zeros(1,Amat_dims(1));

%perform edge search of leading soliton within t_bounds according to density
    parameters,
%begins at min t and proceeds to max t from far right to leading edge
%output array of xmax and xmin for t from min to max, input to width detection.
for r = t_bounds(1) : t_bounds(2)
    c = s(2); %start search at max x
    while c >= x_search_min
        %disp(c)
        if(Amat(r,c) > 1) %determine if datapoint is our leading edge by looking
            backward in x
            total_vals = 1;
            for w = 1 : run_sum-1
                if((yg(r,c-w-1)-yg(r,c-w)) > 0)
                    total_vals = total_vals + 1;
                end
            end
```

```
            if(total_vals == run_sum) %edge found, set x max and search next time value
                x_max(r) = c;
                %now find the width and record as x_min
                g = c;
                while g >= x_search_min - max_width
                    if((yg(r,g)<0) && ((yg(r,g-1)-yg(r,g)) > 0) && (abs(yg(r,g-1)) <
                        yg_threshold_width))
                        %other edge found, record it
                        x_min(r) = g;
                        break;
                    end
                    g = g-1;
                end
                break;
            end
        end
        c = c-1;
    end
end

%scale length
mui=.51; %g/(cm*s) viscosity
mue=12; %g/(cm*s) viscosity
rhoi=1.2286; %g/cm^3 density
rhoe=1.2587; %g/cm^3 density
g=980; %cm/s^2 acceleration
Delta=rhoe-rhoi; %kg/m^3 density
epsilon = mui/mue; %dimensionless
R0= sqrt(8*U0*mui/(g*Delta)); %radius of conduit
L= R0/sqrt(8); %radial length scale
Lv = L/sqrt(epsilon); %vertical length scale

%Extract data for each soliton at discrete time using the edges
plot_data_width = max(x_max-x_min)+xypadding;

%median filtering for if very poor soliton discovery.
%trick for ignoring zeros to get median of sparse vector
A = x_max-x_min;
A(A==0) = NaN;
```

```
median_width_filter = nanmedian(A);
width_filter = 0.03; %filter for widths +/- 3%, not really needed. Data pretty
    consistent here.

%x,y,t
soliton_data = zeros(length(x_max)*plot_data_width,4);

for h = 1 : length(x_max)
    %disp(abs(x_max(h)-x_min(h)))
    %disp(median_width_filter*(1+width_filter))
   if(x_max(h) ~= 0 && abs(x_max(h)-x_min(h)) < median_width_filter*(1+width_filter)
       && abs(x_max(h)-x_min(h)) > median_width_filter*(1-width_filter))
       xmin = x_min(h);
       count = 1;
       while xmin <= x_max(h)
           soliton_data(((h-1)*plot_data_width+count), 1) = xmin; %column location
           soliton_data(((h-1)*plot_data_width+count), 2) = h; %time - y-axis or
               whatever
           soliton_data(((h-1)*plot_data_width+count), 3) = Amat(h,xmin); %amplitude,
               z axis?
           soliton_data(((h-1)*plot_data_width+count), 4) = z_vec(xmin)/Lv; %spatial
               location
           xmin = xmin + 1;
           count = count + 1;
       end
   end
end


soliton_data = soliton_data(any(soliton_data,2),:); %cut out all the null rows
%disp(soliton_data)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate overall fitting for the soliton time slices %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% rebase all solitons to their width in distance and not absolute distance
% center this width around the peak maximum
rebased_soliton_data = soliton_data;
```

```
    rebase = rebased_soliton_data(1,2);
    rebase_val = rebased_soliton_data(1,4);

    %disp(rebased_soliton_data(1,2));
    %disp(rebased_soliton_data(1,4));

    for j = 1 : length(rebased_soliton_data)
        if(rebased_soliton_data(j,2) ~= rebase)
            rebase = rebased_soliton_data(j,2);
            rebase_val = rebased_soliton_data(j,4);
        end

        %chance of identical max amplitudes causing issues but fine for this data
        [m_val, m_row] = max(rebased_soliton_data(rebased_soliton_data(:,2) == rebase,3));
            %get the center index, m_row
        rebased_soliton_data(j,4) = rebased_soliton_data(j,4) - rebase_val -
            z_vec(m_row)/Lv; %rebase and center on maximum amplitude

    end

    %get the fitting function and return plot data
    [try_fit, SS, mumu] = polyfit(rebased_soliton_data(:,4), rebased_soliton_data(:,3),6);
    x_chart_max = max(rebased_soliton_data(:,4));
    x_chart_min = min(rebased_soliton_data(:,4));
    N = x_chart_max - x_chart_min;
    fit_vals = x_chart_min + (0:N);
    [ampFit, delta] = polyval(try_fit, fit_vals, SS, mumu);
    r1 = rebased_soliton_data(:,4);
    r2 = rebased_soliton_data(:,3);
    f1 = fit_vals;
    f2 = ampFit;
    err_up = A0+A0_error;
    err_dn = A0-A0_error;

end
```

```
(*MathematicaPlots.nb*)

In[1]:= (*Plot of c(a)*)
```

```
In[2]:= c[a_]=(a^2-2a^2 Log[a]-1)/(2a-a^2-1);
In[3]:=
    Plot[c[a],{a,1,100},AxesLabel->{"a","c"},PlotStyle->{Blue},LabelStyle->{FontFamily->"Helvetica",18,GrayL
Out[3]=
In[4]:= (*Comparison of theoretical and experimental speed*)
In[5]:= (*Change import location when necessary:)*)
speed = Import["C:\\Users\\alexe\\Desktop\\4350Project\\Speed.csv"];
In[6]:= (*Plot 1*)
In[7]:= nlm=nlm=NonlinearModelFit[speed[[All,{10,6}]],a Log[x]+b,{a,b},x]
nlm["AdjustedRSquared"]
nlm["RSquared"]
Out[7]= FittedModel[1.63557 +0.249868 Log[x]]
Out[8]= 0.984412
Out[9]= 0.985255
In[10]:= speederr={};
For[i=1,i<=Length[speed],i++,
speederr=Join[speederr,{{Around[Part[speed,i,10],Part[speed,i,11]],Part[speed,i,6]}}];
];
In[12]:= Show[Plot[1.6355736484384644` +0.24986773016036098`
    Log[x],{x,0,0.04},PlotStyle->{Black,
    Thick},LabelStyle->{FontFamily->"Helvetica",18,GrayLevel[0]},PlotRange->{0.2,1},AxesLabel->{"Soliton
    Amplitude (cm^2)","Experimental Speed
    (cm/s)"}],ListPlot[speederr,PlotStyle->{RGBColor[0.09,0.5,0.9]}]]
Out[12]=
In[13]:= (*Plot 2*)
In[14]:= nlm=nlm=LinearModelFit[speed[[All,{7,6}]],x,x]
nlm["AdjustedRSquared"]
nlm["RSquared"]
Out[14]= FittedModel[-0.0521697+0.985087 x]
Out[15]= 0.980793
Out[16]= 0.981327
In[17]:= speederr1={};
For[i=1,i<=Length[speed],i++,
speederr1=Join[speederr1,{{Around[Part[speed,i,7],{Part[speed,i,20],Part[speed,i,19]}],Part[speed,i,6]}}];
];
In[19]:= Show[Plot[-0.052169700901639415`+0.9850868201405651`
    x,{x,.3,1},PlotStyle->{Black,
    Thick},LabelStyle->{FontFamily->"Helvetica",18,GrayLevel[0]},AxesLabel->{"Theoretical
    Speed (cm/s)","Experimental Speed
```

```
    (cm/s)"}],ListPlot[speederr1,PlotStyle->{RGBColor[0.09,0.5,0.9]}],Plot[x,{x,.3,1},PlotStyle->{RGBColor[0
Out[19]=
In[20]:= (*Plot 3*)
In[21]:= speederr2={};
For[i=1,i<=Length[speed],i++,
speederr2=Join[speederr2,{{Part[speed,i,10],Around[Part[speed,i,8],{Part[speed,i,22],Part[speed,i,21]
    } ] } } ];
];
In[23]:= (*Plot of data with R^2*)
lm=LinearModelFit[speed[[All,{10,8}]],x,x]
lm["AdjustedRSquared"]
lm["RSquared"]
Out[23]= FittedModel[1.19712 -5.0188 x]
Out[24]= 0.723332
Out[25]= 0.731017
In[26]:= Show[Plot[1.1971173910320068` -5.018804284018654`
    x,{x,0,0.04},PlotStyle->{Black,
    Thick},LabelStyle->{FontFamily->"Helvetica",18,GrayLevel[0]},AxesLabel->{"Soliton
    Amplitude (cm^2)","(Theoretical Speed)/(Experimental
    Speed)"},PlotRange->All],ListPlot[speederr2,PlotStyle->{RGBColor[0.09,0.5,0.9]}]]
Out[26]=
In[27]:= nlm=nlm=NonlinearModelFit[speed[[All,{10,8}]],a x^2+b,{a,b},x]
nlm["AdjustedRSquared"]
nlm["RSquared"]
Out[27]= FittedModel[1.15484 -126.018 x^2]
Out[28]= 0.999448
Out[29]= 0.999478
In[30]:= Show[Plot[1.1548372022199795` -126.0183447167975`
    x^2,{x,0,0.04},PlotStyle->{Black,
    Thick},LabelStyle->{FontFamily->"Helvetica",18,GrayLevel[0]},AxesLabel->{"Soliton
    Amplitude (cm^2)","(Theoretical Speed)/(Experimental
    Speed)"},PlotRange->All],ListPlot[speederr2,PlotStyle->{RGBColor[0.09,0.5,0.9]}]]


(*Euler.nb*)


In[6]:= c[a_]=(a^2-2a^2 Log[a]-1)/(2a-a^2-1); (*c(a) function*)
g[f_,c_]=Sqrt[f^2 ((c+1)/c-2/c Log[f])-2f+(c-1)/c]; (*Our df/d\[Zeta]=g[f,c] function*)
(*Note: choose \[Delta] and \[CapitalDelta]f such that (1+\[Delta])+n*\[CapitalDelta]f =
    A0 for some n*)
```

```
\[Delta]=.001; (*Tolerance*)
f0=1+\[Delta]; (*Initial f value*)
\[CapitalDelta]f=.001; (*Step size*)
\[Zeta]0=-100; (*Initial \[Zeta]*)
\[Zeta]={\[Zeta]0};(*\[Zeta] vector*)
A0=9.64;(*A0 from experiment 48*)
c0=4.101; (*c(A0) of experiment 48*)
fn={f0}; (*f vector*)
(*For loop creating the f vector (fn) until we reach A0*)
For[i=f0+\[CapitalDelta]f,i<A0,i=i+\[CapitalDelta]f,
fn=Join[fn,{i}]
]
(*For loop for Euler's method, creating the \[Zeta] vector*)
For[i=1,i<Length[fn],i++,
(*Print["i: ", i, ", \[Zeta]: ",Part[\[Zeta],i],", f: ",Part[fn,i], ", d\[Zeta]/df:
    ",1/g[Part[fn,i],c0]];*)
\[Zeta]=Join[\[Zeta],{Part[\[Zeta],i]+\[CapitalDelta]f/g[Part[fn,i],c0]}];
];

\[Zeta]=\[Zeta]-Part[\[Zeta],Length[\[Zeta]]];(*Scale \[Zeta] to have A0 at f=0*)
left= Part[\[Zeta]-Part[\[Zeta],Length[\[Zeta]]],1];(*Get left-most point centered around
    0*)

(*Graph both sides of the soliton*)
Show[(*Left
    part*)ListPlot[Transpose[{\[Zeta],fn}],PlotRange->{{(*(-10(left+1))/10*)-20,20(*(10(left+1))/10*)},{0,(10
    (*Right
    part*)ListPlot[Transpose[{-1(\[Zeta]-Part[\[Zeta],Length[\[Zeta]]]),fn}],PlotRange->All,PlotStyle->{Blue
```