Gravity and Shock Waves in a Shallow Fluid

EAS 230 Programming Project
Fall 2018

by

1)Name:Roy Chowthi Section:A4 Person#50188538
2)Name:Nasiah Johnson Section:A3 Person#50145480

Introduction

This project deals with the behavior of a fluid under the influence of gravity through the use of the shallow water equations. The behavior of the fluid will be observed in a long narrow flume. A flume is basically a narrow channel that water flows through. This was the problem we were trying to solve through the project. The main script file that we used in the project was the PP.m file. The other script files that were used were gravity.m, transport.m, nonlinear.m, and a function called trapezoid.m. The function gravity.m and nonlinear.m are given to us but we had to create the function transport.m. We were also given some data files which are u_lin.dat and h_lin.dat. You are also given the soln.mat file which has the non linear solutions u_nlin.dat, h_nlin.dat, and phi_nlin.dat.

Initializing the Model, Loading and Visualizing Solutions

In the beginning of the project we initialized the variables described in part 1 of the project. This includes the scalar quantities, vectors, and matrices. The scalars that were initialized are L=18, g=9.81,H=0.75, rho=1000, w=1.5, N=128M=512, dx=L/N, dt=(0.9*dx)/sqrt(g*H), The vector quantities that were initialized are x, theta, and t. The matrices that we initialized were of the size N+1 by M. The format we used to initialize our matrices was through the use of this code u=zeros(N+1,M),h=zeros(N+1,M),eta=zeros(N+1,M), phi=zeros(N+1,M). Then the values of the first column of our matrix were filled in using the given formulas.

a)In this part of the project we created the plots for figure 1. The data files for the linear solutions were loaded. The files include the u_lin.dat and the h_lin.dat. The files load the variables. The variable phi is user defined. The variables u, h, and phi are matrices which are 129 by 512. The plot is generated with the first column of each of the each matrix vs the variable x.

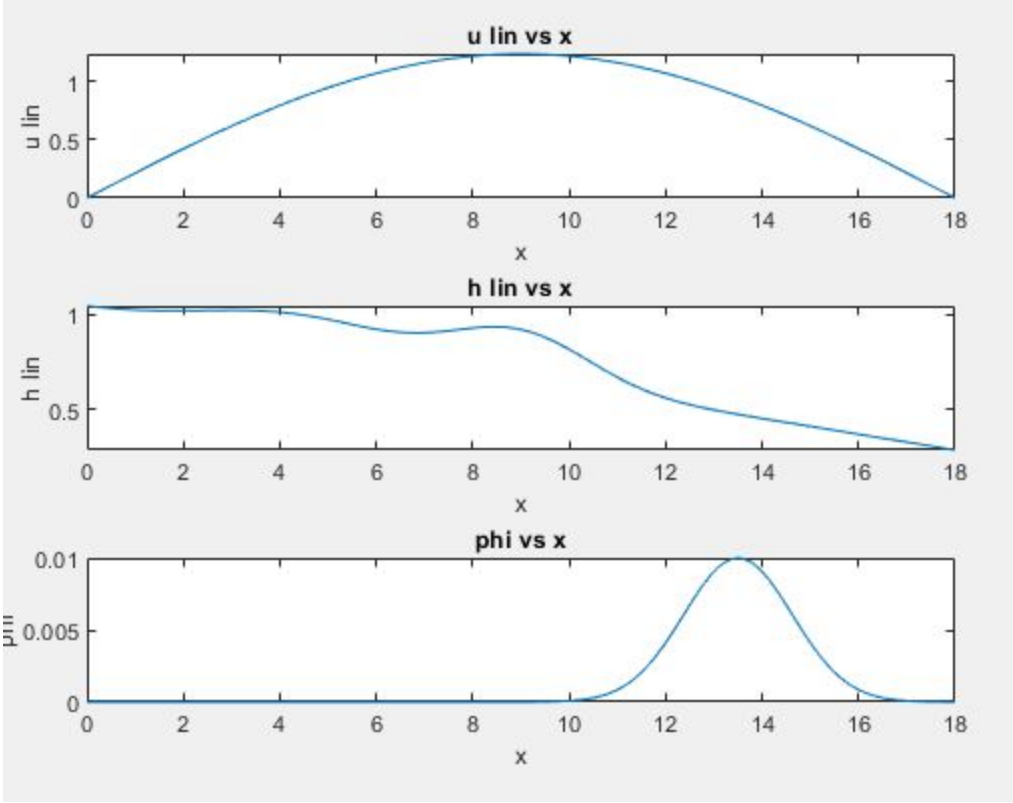

Figure 1: This is the plot for the linear solution for u(x,0), h(x,0), and phi(x,0)

b)In this part of the project we created the plots for figure 2. The nonlinear solutions were loaded from the soln.mat files. The soln.mat file contains three data files u_nlin.dat, h_nlin.dat, and phi_nlin.dat. The variables u ,h and phi are a matrices each 129 by 512. Then using the subplot and plot commands the first column of the variables u, h, and phi were plotted vs the variable x

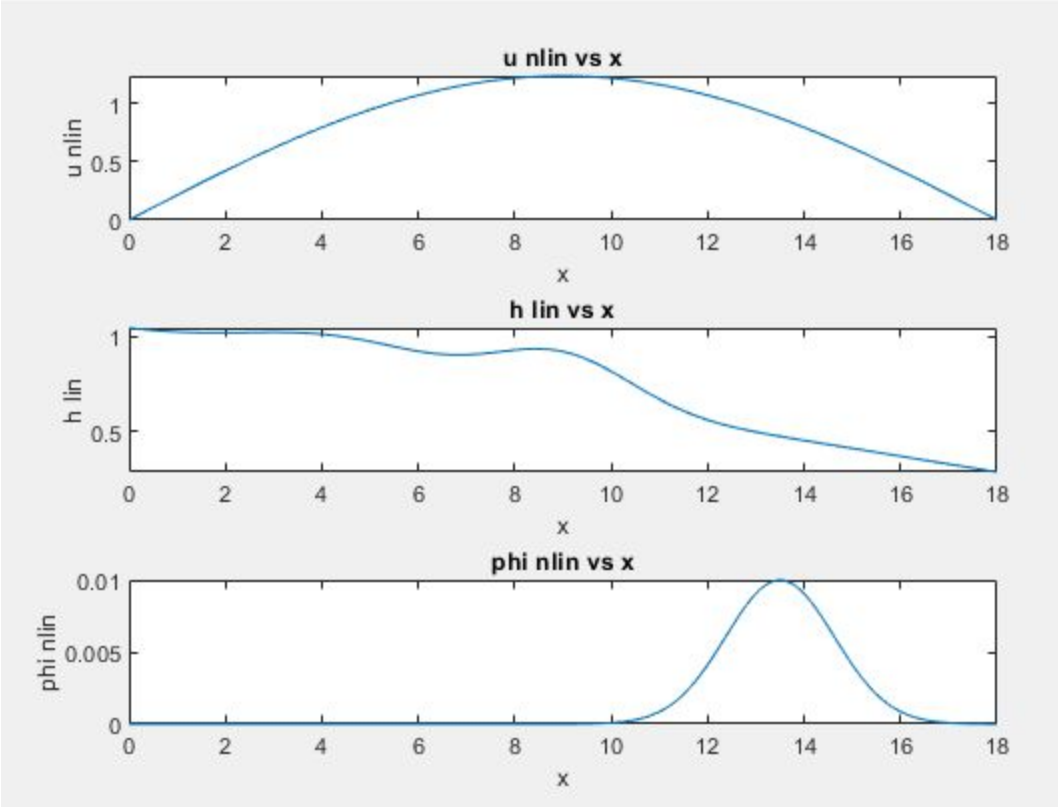

Figure 2: This is the plot for the non-linear solution for u(x,0), h(x,0), and phi(x,0)

c) The plot created in figure 3 plots the nonlinear and linear solutions for h vs x at various time intervals. At the time interval 1 the linear and nonlinear plot is the same. However, from 128 and beyond the nonlinear and linear solutions vary.

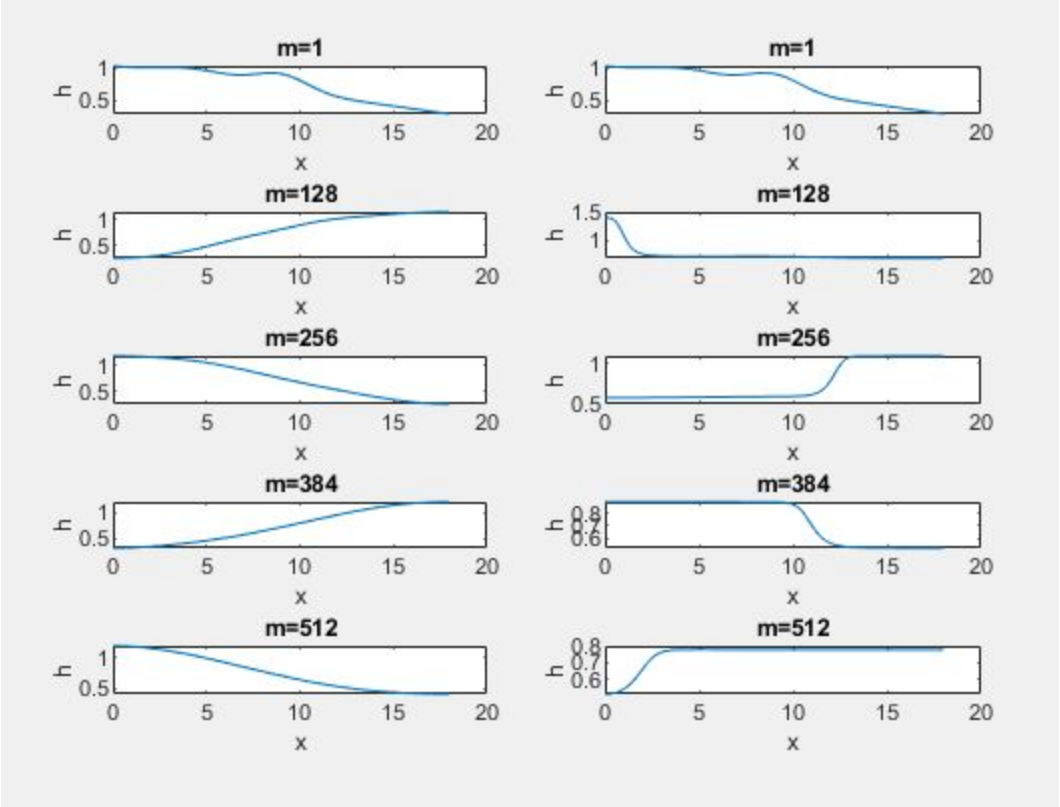

Figure 3: This is the plot of h vs x at the time intervals 1, 128, 256, 384, 512 from rows 1 to 5. Column 1 is the linear solution and column 2 is the nonlinear solution.

d)The animation looks like a wave flowing with the peak changing elevation as the peak moves from the left to the right.

Creating the Models and Reproducing Solutions

a) For creating the non-linear model, a flag was created to determine what model to use. If 1 was selected we did a nonlinear model.If 0 was selected, we did a linear model of the system. After we selected 1 to do the linear model, The function nonlinear was called which is in a for loop. This for loop cycles through different time steps start from 2 and ending at 512. The function inputs are the system variables at the first time step and creates the remaining 511 time steps from the previous column. This is all recorded and placed in a column after it making a 129 by 512 matrix of the variables that's being used in the model. After this, we plotted the x vs phi of time step 1, 128, and 184.

b)For creating the linear model, the flag selection was 0. The user will be asked to enter which model to use based off this selection value. To execute the linear model we needed to call the gravity function and the transport function. The transport function we had to create and call after gravity function. Inside the gravity function, we had it take inputs of phi, u, N, dt, and dx. First, we checked to see if phi and u was size of 129 by 1. If it wasn't, we immediately existed the

function. Next, we had to preallocate a matrix A because we are setting up a Ax = b to calculate our new phi for a specific time step. We know that the indices (1,1) and (1,2) of matrix A are -1 and 1, and indices (129,129) and (129,128) were 1 and -1.We had to preallocate a matrix called b with the previous value of tracer concentration to be us to calculate for new tracer values. X are the new tracer concentrations needed to be solved. The diagonal of matrix A were filled with the coefficients tracer concentration for the three different columns of the same time step. The coefficients for the phi's that will go for the diagonal of matrix A are $-\Delta t * un/2\Delta x$, $1 + \Delta t*(un+1 - un-1)/2\Delta x$, and $\Delta t*un/2\Delta x$, respectively. So the output for this system will be new tracer concentration that will be uploaded to the phi variable of the same time-step. The output was calculated as x = A\b.This will give you 129 new tracer concentrations for each time step.

c)For making a linear model, we had to call gravity and transport function. Gravity created the linear solution for u, h, and eta. Transport created the linear solution for phi(tracer concentration). I plotted the tracer concentration x vs. three different time step, 1, 128, and 184.

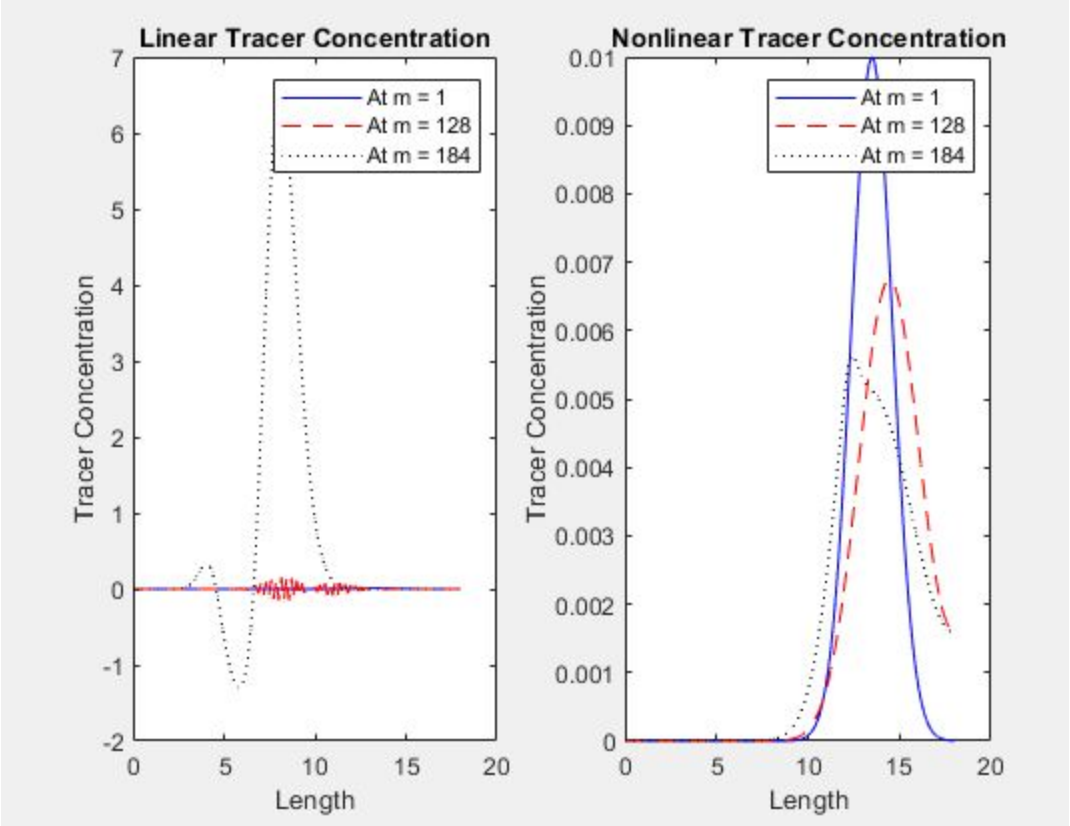

Figure 4:

Analysis

a)The integral was solved with the function trapezoid.m which was called in pp.m to solve the integrals. The script file would loop through each element in the matrix

and then calculate the integral based on that. The trapezoid function looped through fn N+1 times to take the integral which was 129. The

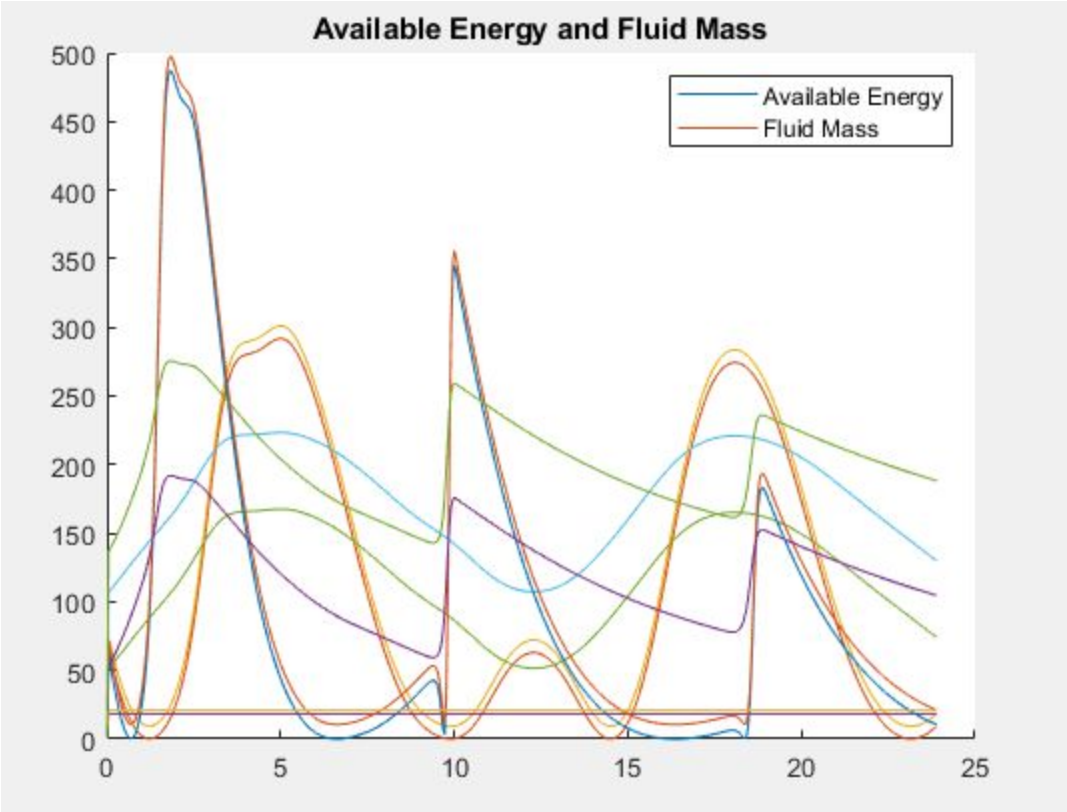

Figure 5: This figure shows the available energy and fluid mass for the figure for the non linear and non linear solutions

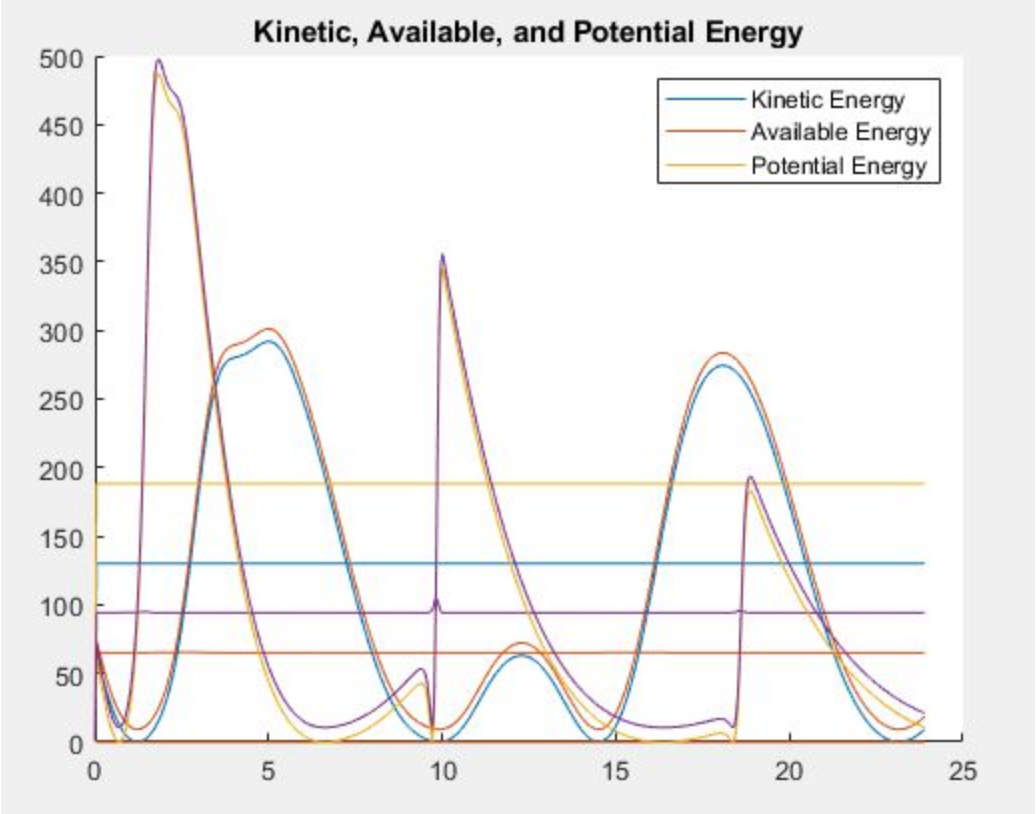

Figure 6: This figure shows the available energy , the kinetic, and potential energy of the linear and non linear solutions

b)The nonlinear has a greater shock in the depth of the fluid. The linear solutions change more gradually compared to the nonlinear solutions.

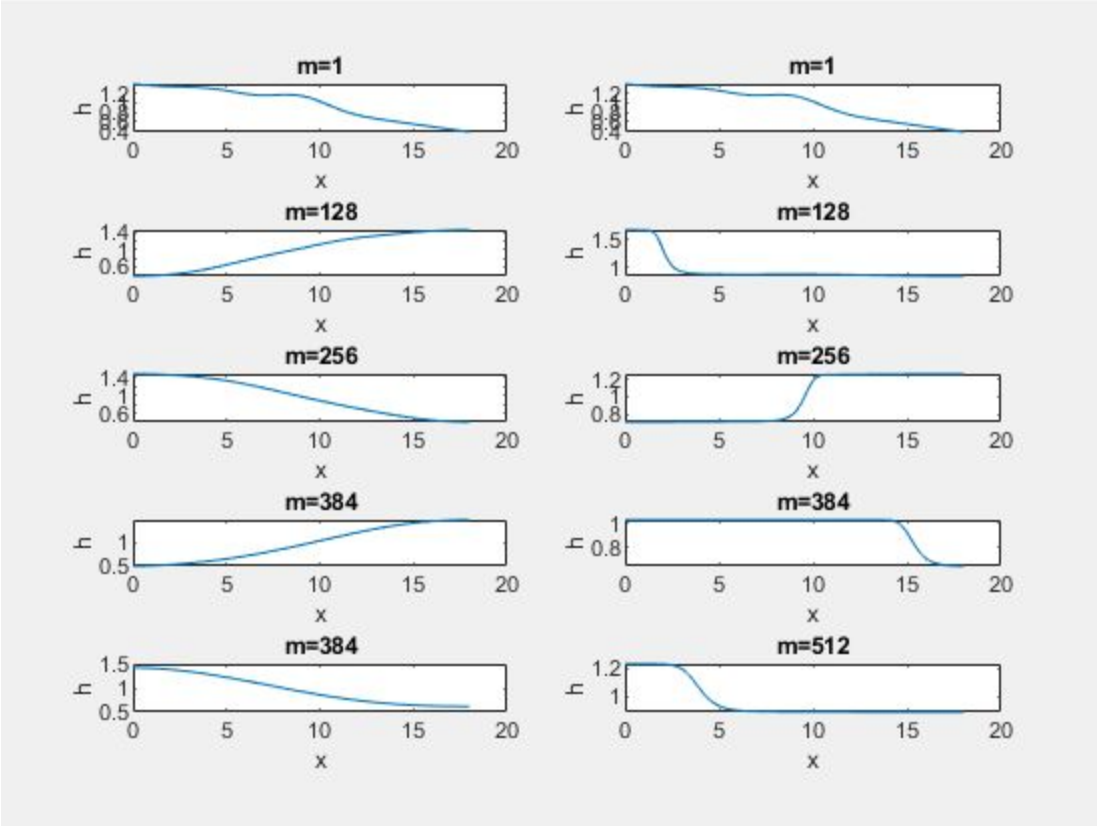

Figure 7: This figure is plotting h vs x for linear and non linear solutions at different time intervals. Column 1 is linear and column 2 is nonlinear.

Conclusion

This project was difficult to complete and interpret what was being done. As we coded the project, we learned the concepts of energy conservation. We learned how to get different data from the first column and make and fill out the rest of a matrix. Trapezoid gave us integrals of the energy, which we plotted. Transport updated tracer concentration values. Gravity updated h,eta, and u. We plotted the data of linear and non linear and it was not too different.

Appendix

PP.m

```
%scalars
L=18;
g=9.81;
H=0.75;
rho=1000;
w=1.5;
N=128;
M=512;
dx=L/N;
dt=(0.9*dx)/sqrt(g*H);
%vectors and 1D arrays
x=linspace(0,N*dx,N+1);
theta=2*pi*x/L;
t=linspace(0,dt*M,M);
%matrices and 2D arrays
u=zeros(N+1,M);
h=zeros(N+1,M);
eta=zeros(N+1,M);
phi=zeros(N+1,M);
u(:,1)=(5/11)*sqrt(g*H)*sin(theta/2);
h(:,1)=(H)*((1/3).*cos((theta-pi)/2 ).^16 + (1/5).*(cos((theta-pi/2)/4).^32)) +
(1-(x/L))+(691/1818);
eta(:,1)=h(:,1)-H;
phi(:,1)=(1/100)*exp(-128*((x./L)-(3/4)).^2);
%part a
%load
load h_lin.dat;
load u_lin.dat;
load soln.mat;
figure(1)
subplot(3,1,1);
plot(x,u_lin(:,1))
xlabel('x')
ylabel('u lin')
title('u lin vs x')
subplot(3,1,2);
```

```
plot(x,h_lin(:,1))
xlabel('x')
ylabel('h lin')
title('h lin vs x')
subplot(3,1,3);
plot(x,phi(:,1))
xlabel('x')
ylabel('phi')
title('phi vs x')
%part b
figure(2)
subplot(3,1,1);
plot(x,u_nlin(:,1))
xlabel('x')
ylabel('u nlin')
title('u nlin vs x')
subplot(3,1,2);
plot(x,h_nlin(:,1))
title('h lin vs x')
xlabel('x')
ylabel('h lin')
subplot(3,1,3);
plot(x,phi_nlin(:,1))
title('phi nlin vs x')
xlabel('x')
ylabel('phi nlin')
%part c
figure(3)
subplot(5,2,1);
plot(x,h_lin(:,1))
title('m=1')
xlabel('x');
ylabel('h');
subplot(5,2,2)
plot(x,h_nlin(:,1))
title('m=1')
xlabel('x');
ylabel('h');
subplot(5,2,3)
```

```
plot(x,h_lin(:,128))
title('m=128')
xlabel('x');
ylabel('h');
subplot(5,2,4)
plot(x,h_nlin(:,128))
title('m=128')
xlabel('x');
ylabel('h');
subplot(5,2,5)
plot(x,h_lin(:,256))
title('m=256')
xlabel('x');
ylabel('h');
subplot(5,2,6)
plot(x,h_nlin(:,256))
title('m=256')
xlabel('x');
ylabel('h');
subplot(5,2,7)
plot(x,h_lin(:,384))
title('m=384')
xlabel('x');
ylabel('h');
subplot(5,2,8)
plot(x,h_nlin(:,384))
title('m=384')
xlabel('x');
ylabel('h');
subplot(5,2,9)
plot(x,h_lin(:,512))
title('m=512')
xlabel('x');
ylabel('h');
subplot(5,2,10)
plot(x,h_nlin(:,512))
title('m=512')
xlabel('x');
ylabel('h');
```

```
%-------------------------------------------------------------------------------
%------------------------------Part 2 -------------------------
flag = 1;
while flag == 1
   if flag == 1
   selection = input('Please enter 0 for linear and 1 for non-linear . If either number is not 0 or
one, you will be asked to do so. ');
       if selection == 1
          flag = 0;
       end
       if selection == 0
          flag = 0;
       end

   end
end

for m = 2:512
   if selection == 1
       %Why are the variables not updating correctly is the variables

       [u(:,m),h(:,m),eta(:,m),phi(:,m)] =
nonlinear(u(:,m-1),h(:,m-1),eta(:,m-1),phi(:,m-1),N,dx,dt,g,H);

       %each loop fill out data table
%          u(:,i)= u_est(:,i);
%          h(:,i)= h_est(:,i);
%          eta(:,i)= eta_est(:,i);
%          phi(:,i)= phi_est(:,i);
   end
   if selection == 0
      [u(:,m),h(:,m),eta(:,m)] = gravity(u(:,m-1),h(:,m-1),eta(:,m-1),N,dx,dt,g,H);
      phi(:,m) = transport(phi(:,m-1),u(:,m-1), N, dt, dx);
   end
end
%to plot both subplots run the program twice the first time type in 0 for
%linear and the second time type in 1 for nonlinear
if selection == 0
```

```
    figure(4)
    hold on
    subplot(1,2,1)
    plot(x,phi(:,1),'-b',x,phi(:,128),'--r',x,phi(:,184),':k');
    title('Linear Tracer Concentration');
    xlabel('Length');
    ylabel('Tracer Concentration');
    legend('At m = 1','At m = 128','At m = 184');
end

 if selection == 1
     figure(4)
     hold on
     subplot(1,2,2)
     plot(x,phi(:,1),'-b',x,phi(:,128),'--r',x,phi(:,184),':k');
     title('Nonlinear Tracer Concentration');
     xlabel('Length');
     ylabel('Tracer Concentration');
     legend('At m = 1','At m = 128','At m = 184');

 end
 hold off
%----------------------------------Part 3---------------------------------

f_n=zeros(N+1,1);
Available_Energy=zeros(N+1,M);
Fluid_Mass=zeros(N+1,M);
Kinetic_Energy=zeros(N+1,M);
Potential_Energy=zeros(N+1,M);
for i=1:1:N+1
    for j=1:1:M
    %available Energy
    f_n(i,1)=((1/2)*h(i,j).*(u(i,j).^2))+((1/2)*g*(eta(i,j).^2));
    lhs_integral = trapezoid(f_n,N,dx);
    Available_Energy(i,j)=(rho*w)*lhs_integral;
    Available_Energy(i,j)=Available_Energy(i,j)/Available_Energy(i,1);
    end
end
for i=1:1:N+1
```

```
    for j=1:1:M
    %Fluid Mass
    f_n(i,1)=h(i,j);
    lhs_integral = trapezoid(f_n,N,dx);
    Fluid_Mass(i,j)=(rho*w)*lhs_integral;
    Fluid_Mass(i,j)=Fluid_Mass(i,j)/Fluid_Mass(i,1);
    end
end
for i=1:1:N+1
    for j=1:1:M
    %Kinetic Energy
    f_n(i,1)=(1/2)*h(i,j).*(u(i,j).^2);
    lhs_integral = trapezoid(f_n,N,dx);
    Kinetic_Energy(i,j)=(rho*w)*lhs_integral;
    Kinetic_Energy(i,j)=Kinetic_Energy(i,j)/Kinetic_Energy(i,1);
    end
end
for i=1:1:N+1
    for j=1:1:M
    %Available Potential Energy
    f_n(i,1)=(1/2)*g*(eta(i,j).^2);
    lhs_integral = trapezoid(f_n,N,dx);
    Potential_Energy(i,j)=(rho*w)*lhs_integral;
    Potential_Energy(i,j)=Potential_Energy(i,j)/Potential_Energy(i,1);
    end
end
figure(5)
hold on
if(selection == 1)
    plot(t,Available_Energy)
    plot(t,Fluid_Mass)
    title('Available Energy and Fluid Mass')
end
if(selection == 0)
    plot(t,Available_Energy)
    plot(t,Fluid_Mass)
    title('Available Energy and Fluid Mass')

end
```

```
legend('Available Energy','Fluid Mass')
hold off

figure(6)
hold on
if selection ==1
   plot(t,Kinetic_Energy)
   plot(t,Available_Energy)
   plot(t,Potential_Energy)
   title('Kinetic, Available, and Potential Energy')
end

if selection ==0
   plot(t,Kinetic_Energy)
   plot(t,Available_Energy)
   plot(t,Potential_Energy)
   title('Kinetic, Available, and Potential Energy')
end
legend('Kinetic Energy', 'Available Energy', 'Potential Energy')
hold off
hold on
%run twice for linear and nonlinear subplots
if selection==0
   figure(7)
   subplot(5,2,1);
   plot(x,h(:,1))
   title('m=1 Linear')
   xlabel('x');
   ylabel('h');
   subplot(5,2,3)
   plot(x,h(:,128))
   title('m=128 Linear')
   xlabel('x');
   ylabel('h');
   subplot(5,2,5)
   plot(x,h(:,256))
   title('m=256 Linear')
   xlabel('x');
   ylabel('h');
```

```
    subplot(5,2,7)
    plot(x,h(:,384))
    title('m=384 Linear')
    xlabel('x');
    ylabel('h');
    subplot(5,2,9)
    plot(x,h(:,512))
    title('m=384 Linear')
    xlabel('x');
    ylabel('h');
end
hold on
if selection==1
    figure(7)
    subplot(5,2,2);
    plot(x,h(:,1))
    title('m=1 Non Linear')
    xlabel('x');
    ylabel('h');
    subplot(5,2,4)
    plot(x,h(:,128))
    title('m=128 Non Linear')
    xlabel('x');
    ylabel('h');
    subplot(5,2,6)
    plot(x,h(:,256))
    title('m=256 Non Linear')
    xlabel('x');
    ylabel('h');
    subplot(5,2,8)
    plot(x,h(:,384))
    title('m=384 Non Linear')
    xlabel('x');
    ylabel('h');
    subplot(5,2,10)
    plot(x,h(:,512))
    title('m=512 Non Linear')
    xlabel('x');
    ylabel('h');
```

```
end
hold off
```

Trapezoid Function

```
function [lhs_integral] = trapezoid(f_n,N,dx)

%assigns the size of the each input variables an array
size_N=size(N);
size_f_n=size(f_n);
size_dx=size(dx);

%determines if the size of the array is equal to the specified value.
%if it isn't equal to the specified value the function returns error
if(size_N~=[1 1])
   error(error)
end
if(size_f_n~=[N+1 1])
   error(error)
end
if(size_dx~=[1 1])
   error(error)
end

%initializes the variable n and the while loop will loop N times and create
%a summation of the previous value of f_n and the current value then
%multiply it by dx
for n=1:1:N
   lhs_integral=((f_n(n,1))+(f_n(n+1,1)))/2*dx;
end
end
```

Transport Function

```
function phi_new = transport(phi,u, N, dt, dx)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
size_phi = size(phi);
```

```
size_u = size(u);

if size_phi(1,1) ~= 129
   error('Matrix of phi is not the correct size');
   if size_phi(1,2) ~= 1
      error('Matrix of phi is not the correct size');
   end
end

if size_u(1,1) ~= 129
   error('Matrix of phi is not the correct size');
   if size_u(1,2) ~= 1
      error('Matrix of phi is not the correct size');
   end
end

A = zeros(N+1);
A(1,1) = -1;
A(1,2) = 1;
A(N+1,N+1) =1;
A(N+1,N) = -1;
b = zeros(129,1);
for n = 2:N
   A(n,n-1) = (-dt)*u(n,1);
   A(n,n) = 1 + (dt*(u(n+1,1) - u(n-1,1)))/(2*dx);
   A(n,n+1) = dt*u(n,1)/(2*dx);
   b(n,1) = phi(n,1);
end

phi_new = A\b;
end
```