2 Waves

2.7 Soliton Solutions of the KdV Equation (7 units)

This project assumes only basic knowledge of wave equations and does not rely directly on any Part II lecture course. The Part II(D) course Integrable Systems *may be useful but the project should be quite accessible to those who have not taken this course, using Drazin & Johnson [1] as a suitable reference.*

1 Theory

The Korteweg-de Vries (KdV) equation,

$$u_t + uu_x + \delta^2 u_{xxx} = 0\,, \tag{1}$$

arises in many branches of physics as a model for the evolution and interaction of nonlinear waves. It is well-known that the equation possesses single-soliton solutions of the form $u(x,t) = f(x-ct)$, where

$$f(x) = A\mathrm{sech}^2\left(\frac{x-x_0}{\Delta}\right)\,, \quad \Delta^2 = 12\delta^2/A\,, \quad c = A/3\,, \tag{2}$$

and where A and x_0 are arbitrary constants representing the amplitude and initial location of the soliton respectively. It is supposed that $u(x,t)$ obeys the cyclic boundary conditions,

$$u(x+1,t) = u(x,t)\,, \tag{3}$$

so that only the region $0 \leqslant x \leqslant 1$ need be considered.

2 Questions

Question 1 Verify that the single soliton (2) is indeed a [non-periodic] solution of the KdV equation. For a periodic solution satisfying (3), prove that the mass, M, and the energy, E, of the motion, defined by

$$M \equiv \int_0^1 u(x,t)dx \quad \text{and} \quad E \equiv \int_0^1 \frac{1}{2}u^2(x,t)dx\,, \tag{4}$$

are independent of time.

Question 2 A leap-frog scheme first employed by Zabusky & Kruskal for solving the KdV equation is given on page 184 of Drazin & Johnson [1]. Let u_m^n be the solution at $x = hm$ and $t = kn$ with n, $m = 0$, 1,..., then the KdV equation can be discretised as

$$u_m^{n+1} = u_m^{n-1} - \frac{k}{3h}(u_{m+1}^n + u_m^n + u_{m-1}^n)(u_{m+1}^n - u_{m-1}^n) - \frac{k\delta^2}{h^3}(u_{m+2}^n - 2u_{m+1}^n + 2u_{m-1}^n - u_{m-2}^n)\,. \tag{5}$$

What is the order of this scheme? For $\delta = 1$, this scheme will be stable provided that

$$k \leqslant \frac{h^3}{4 + h^2|u_{\max}|}\,, \tag{6}$$

where $|u_{\max}|$ is the maximum modulus of u. By rescaling the KdV equation, derive the stability condition for $\delta \neq 1$.

Write a program to implement (5). Note that you will need an alternative method for making the first step; explain your choice carefully in your write-up. As as a check on the accuracy of your program, use it to calculate $u(x, 0.75)$ for the initial data

$$u(x,0) = A\,\mathrm{sech}^2\left(\frac{x-x_0}{\Delta}\right), \tag{7}$$

with $\delta = 0.04$, $A = 2.5$ and $x_0 = 0.25$.

Estimate the error in your results, and clearly explain the reasoning behind your choice of the values of k and h. Plot on the same axes graphs of both your numerical solution and the exact solution at $t = 0.75$ Comment on any differences. Does the propagation speed of the numerical solution agree exactly with that of the analytical solution?

Question 3 Describe and comment on the evolution of the initial data corresponding to the sum of two solitons, one with $A = 2.5$ and $x_0 = 0.25$, and a second with $A = 1$ and $x_0 = 0.75$, again with $\delta = 0.04$. Illustrate your answer with plots of the numerical solution at several instants. In addition, use the trapezium rule to calculate the mass and energy associated with your numerical solution at each time step, and comment on their variation with time (you need not output these quantities at every time step).

Question 4 Consider now

$$u(x,0) = \sin(2\pi x)\,. \tag{8}$$

In the case $\delta = 0$ give a very brief qualitative description of the evolution of this initial data. Use your program to investigate the case $\delta = 0.04$ numerically, plotting graphs of the solution at suitable instants. With particular reference to the relative magnitudes of the various terms in the KdV equation, explain why your numerical solution changes character at a certain time; estimate this time. Describe how the solution evolves after this time.

Try different values of δ, both larger and smaller than 0.04, and comment on the results that you obtain.

References

[1] Drazin, P.G. & Johnson, R.S. (1989) *Solitons: An Introduction*, CUP.

[2] Zabusky, N. & Kruskal, M (1965) Interaction of "solitons" in a collisionless plasma and the recurrence of initial states. *Physical Review Letters*, volume 15, number 6, pages 240-3.

2.7 Soliton Solutions of the KdV Equation

Question 1

For clarity let $s = \operatorname{sech}\left(\frac{x-ct-x_0}{\Delta}\right)$ and $t = \tanh\left(\frac{x-ct-x_0}{\Delta}\right)$.

Recall that as $u(x,t) = f(x-ct)$ we have:

$$u(x,t) = As^2.$$

It follows that:

$$u_t = \frac{2Acts^2}{\Delta}$$

$$u_x = \frac{-2Ats^2}{\Delta}$$

$$u_{xxx} = \frac{8Ats^2(2s^2 - t^2)}{\Delta^3}.$$

So:

$$\begin{aligned}
u_t + uu_x + \delta^2 u_{xxx} &= \frac{2Ats^2}{\Delta}\left[c - As^2 + \frac{4\delta^2}{\Delta^2}(2s^2 - t^2)\right] \\
&= \frac{2Ats^2}{\Delta}\left[\frac{A}{3} - As^2 + \frac{A}{3}(2s^2 - t^2)\right] \\
&= \frac{2A^2ts^2}{\Delta}\left[\frac{1}{3} - s^2 + \frac{2s^2}{3} - \frac{t^2}{3}\right] \\
&= \frac{2A^2ts^2}{3\Delta}\left[1 - s^2 - t^2\right] \\
&= 0.
\end{aligned}$$

Noting that $1 - s^2 - t^2 = 0$ is a standard hyperbolic identity.

Now to show that M is independent of time we consider M_t:

$$\begin{aligned}
M_t &= \int_{x=0}^{x=1} u_t(x,t)dx \\
&= \frac{2Ac}{\Delta}\int_{x=0}^{x=1} ts^2 dx.
\end{aligned}$$

We make the substitution $y = \frac{x-ct-x_0}{\Delta}$

$$M_t = 2Ac \int_{x=0}^{x=1} \tanh(y)\,\mathrm{sech}^2(y)dy$$

Now we let $v = sech(y)$:

$$\begin{aligned} M_t &= 2Ac \int_{x=0}^{x=1} v(-dv) \\ &= -c\big[Av^2\big]_{x=0}^{x=1} \\ &= -c\big[u(x,t)\big]_{x=0}^{x=1} \\ &= 0. \end{aligned}$$

The last line is 0 by equation (3) of the project manual (i.e. $u(x+1,t) = u(x,t)$). So it follows that the mass M is independent of time.

$$\begin{aligned} E_t &= \int_{x=0}^{x=1} uu_t dx \\ &= \frac{2A^2c}{\Delta} \int_{x=0}^{x=1} ts^4 dx \\ &= 2A^2c \int_{x=0}^{x=1} \tanh(y)\,\mathrm{sech}^4(y)dy \\ &= 2A^2c \int_{x=0}^{x=1} v^3(-dv) \\ &= -2c\big[A^2v^4\big]_{x=0}^{x=1} \\ &= -2c\big[u^2\big]_{x=0}^{x=1} \\ &= 0. \end{aligned}$$

We get the last line as 0 by similar reasoning to before and so E is independent of time.

Question 2

Let us expand the terms as follows:

$$u_m^{n\pm1} = u(x, t \pm k) = \sum_{i=0}^{N} \frac{(\pm k)^i}{i!} \frac{\partial^i u}{\partial t^i} + \mathcal{O}(k^{N+1})$$

$$u_{m\pm1}^{n} = u(x \pm h, t) = \sum_{j=0}^{N} \frac{(\pm h)^j}{j!} \frac{\partial^j u}{\partial x^j} + \mathcal{O}(h^{N+1})$$

$$u_{m\pm2}^{n} = u(x \pm 2h, t) = \sum_{j=0}^{N} \frac{(\pm 2h)^j}{j!} \frac{\partial^j u}{\partial x^j} + \mathcal{O}(h^{N+1}).$$

So equation (5) yields:

$$
\begin{aligned}
u_m^{n+1} - u_m^{n-1} + \frac{k}{3h}(u_{m+1}^n + u_m^n + u_{m-1}^n)(u_{m+1}^n - u_{m-1}^n) + \frac{k\delta^2}{h^3}(u_{m+2}^n - 2u_{m+1}^n + 2u_{m-1}^n - u_{m-2}^n) \\
= \sum_{i=0}^{N} \frac{(k)^i}{i!}\frac{\partial^i u}{\partial t^i} - \sum_{i=0}^{N} \frac{(-k)^i}{i!}\frac{\partial^i u}{\partial t^i} \\
+ \frac{k}{3h}\left[\sum_{j=0}^{N} \frac{(h)^j}{j!}\frac{\partial^j u}{\partial x^j} + u + \sum_{j=0}^{N} \frac{(-h)^j}{j!}\frac{\partial^j u}{\partial x^j}\right]\left[\sum_{j=0}^{N} \frac{(h)^j}{j!}\frac{\partial^j u}{\partial x^j} - \sum_{j=0}^{N} \frac{(-h)^j}{j!}\frac{\partial^j u}{\partial x^j}\right] \\
+ \frac{k\delta^2}{h^3}\left[\sum_{j=0}^{N} \frac{(2h)^j}{j!}\frac{\partial^j u}{\partial x^j} - 2\sum_{j=0}^{N} \frac{(h)^j}{j!}\frac{\partial^j u}{\partial x^j} + 2\sum_{j=0}^{N} \frac{(-h)^j}{j!}\frac{\partial^j u}{\partial x^j} - \sum_{j=0}^{N} \frac{(-2h)^j}{j!}\frac{\partial^j u}{\partial x^j}\right].
\end{aligned}
$$

We take $k = \mathcal{O}(h^3)$ as on page 184 of Drazin & Johnson [1].

We now consider terms of the first few orders in h:

$$\mathcal{O}(h^0) : u - u + \frac{k\delta^2}{h^3}(u - 2u + 2u - u) = 0$$

$$\mathcal{O}(h) : \frac{k\delta^2}{h^3}(2hu_x - 2hu_x - 2hu_x + 2hu_x) = 0$$

$$\mathcal{O}(h^2) : \frac{k\delta^2}{h^3}(4h^2u_{xx} - 2h^2u_{xx} + 2h^2u_{xx} - 4h^2u_{xx}) = 0$$

$$
\begin{aligned}
\mathcal{O}(h^3) : ku_t + ku_t + \frac{k}{3h}(3u)(2hu_x) + \frac{k\delta^2}{h^3}(\tfrac{8h^3}{3!}u_{xxx} - \tfrac{2h^3}{3!}u_{xxx} - \tfrac{2h^3}{3!}u_{xxx} + \tfrac{8h^3}{3!}u_{xxx}) \\
= 2k(u_t + uu_x + u_{xxx}) \\
= 0
\end{aligned}
$$

$$\mathcal{O}(h^4) : \frac{k\delta^2}{h^3}(16h^3u_{xxxx} - 2h^2u_{xxxx} + 2h^2u_{xxxx} - 16h^3u_{xxxx}) = 0.$$

The terms of order $\mathcal{O}(h^5)$ is non-zero so the scheme is order $\mathcal{O}(h^5)$.

As in Greig & Morris [2] we see that the stability condition for the equation $u_t + \beta u u_x + \epsilon u_{xxx} = 0$ is:

$$p\Big(\beta|u_{\max}| - \frac{2\epsilon}{h^2}\Big) \leq 1.$$

where $p = k/h$.

Applying this to the our equation we let $\beta = 1$ and $\epsilon = \delta^2$ and by rearranging we obtain the stability condition for $\delta \neq 1$:

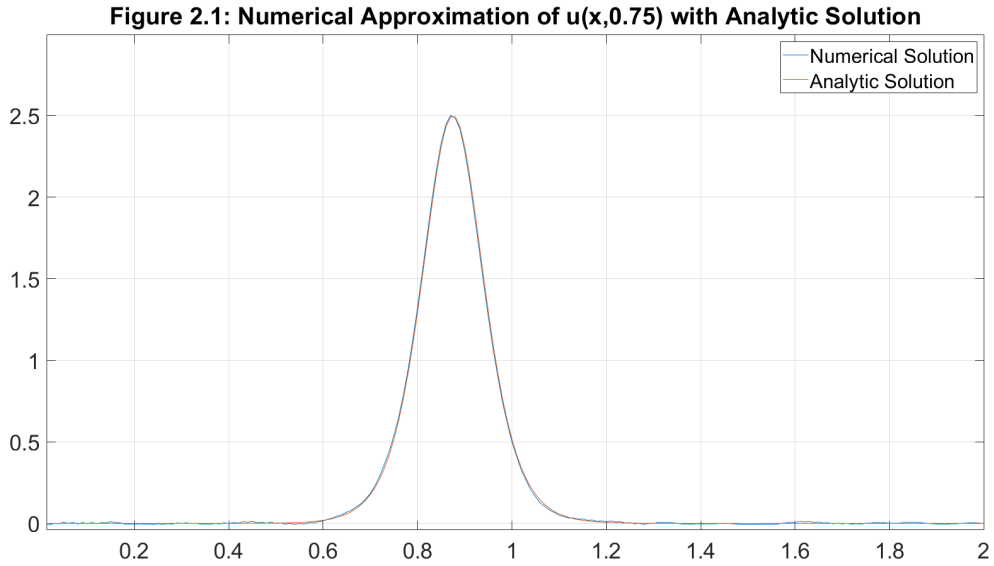
$$k \leq \frac{h^3}{4\delta^2 + h^2|u_{\max}|}.$$

I have written a program to implement method the leap-frog scheme to solve the differential equation numerically callled `Question_2` (whose script can be found on page 17). Note that we cannot use equation (5) exactly to approximate the first time step as we do not have the "u_m^{n-1}" term. Instead we use:

$$u_m^{n+1} = u_m^n - \frac{k}{3h}(u_{m+1}^n + u_m^n + u_{m-1}^n)(u_{m+1}^n - u_{m-1}^n) - \frac{k\delta^2}{h^3}(u_{m+2}^n - 2u_{m+1}^n + 2u_{m-1}^n - u_{m-2}^n)$$

This method yields only $\mathcal{O}(k)$ error but as we have $k = \mathcal{O}(h^3)$ this method is $\mathcal{O}(h^3)$ in the first time step. While a higher amount of error will occur in this step then will in any other step it shouldn't have a significant effect on the overall error as it will only be used for one time step.

I now test the program with the values given in the project manual and generate the following plot using `Question_2b` (found on page 19):



I set `h = 0.01` and `k = 1e-6`. The choice of `k` was chosen with the stability condition in mind, we have an upper bound on it so cannot make it too big. Setting `h` an order of magnitude smaller then `0.01` (with this upper bound on `k` in mind) resulting in MATLAB warning me that the calculation would take up over 100GB and the program would take a long time to run.

As in figure 2.1 (above) we see that the numerical solution and analytic solution both yield a plot with the same shape and a peak at around the same place in the plot. When zoomed in you can see the numerical solution looks a less smooth, "vibrating" around the analytic solution. By running `Question_2c` (the code for which can be found on page 20) we can see the global error is 0.04. Through checking the wave numerically generated against the analytic solution at various times you can see that the propagation speed of the numerical solution agrees exactly with that of the analytic solution.

Question 3

I have modified the program produced in Question 2 to solve the new system with initial data the sum of two solitons. Note I have made a minor modification of numerical method (5) in this script, choosing consider the how thee system behaves in the interval $[-1, 3]$ rather then a positive domain - I will explain why this change was made later. The modified script is called `Question_3` and is found on page 21. I have also used a script to plot the mass and energy of the system called `Question_3b` and found on page 23.

As you can see in figure 3.1, the waves initially have started to merge, but are mostly separated. As time increases and the two peaks of the waves get closer to one another we see that the peak of the left wave is lower and the peak of the right wave is higher (this is demonstrated on figure 3.2). On Figure 3.3 we see that the the two waves have almost completely merged. After they merge the two waves start to separate with the wave on the right of the figure now taller than the one on the right (see figure 3.4). After an amount of time the two waves separate and we now see the tall wave on the right and the short wave on the left (seen on figure 3.5). The figures mentioned here populate the next three pages and gernerated using `Question_2b`.

Now we will consider the mass (M) and energy (E) of the system. As in equation (4):

$$M \equiv \int_0^1 u(x,t)dx \quad E \equiv \int_0^1 \tfrac{1}{2}u^2(x,t)dx \quad (4)$$

With these values of M and E `Question_3b` produces figures 3.6 and 3.7 (found on page 8). We see that both the mass and energy initially are lost very slowly. There is then a period of rapid loss of energy and mass (between time 0.6 and 1). Finally both mass and energy are lost at a slower rate again with both values tending to zero. Note we do not see conservation of these quantities as the solution isn't periodic.

We can however define mass and energy differently so that they are conserved quantities (given our solution $u(x,t)$ tend to 0 as $x \to \pm\infty$ for any time t). Let our alternately defined mass, M_c and energy , E_c be:

$$M_c \equiv \int_{-\infty}^{\infty} u(x,t)dx \quad E_c \equiv \int_{-\infty}^{\infty} \tfrac{1}{2}u^2(x,t)dx \quad (*)$$

Observe that the area under the tails of the curve on either side are negligible for all times. We can't use numerical method (5) to find $u(x,t)$ between $-\infty$ and ∞ but we can choose the range of x s.t:

$$M_c \equiv \int_{-\infty}^{\infty} u(x,t)dx \approx \int_{x_0}^{x_1} u(x,t)dx$$

$$E_c \equiv \int_{-\infty}^{\infty} \frac{1}{2}u^2(x,t)dx \approx \int_{x_0}^{x_1} u^2(x,t)dx$$

I decided that the range $[-1,3]$ would result in a small enough error while not elongating run time of the script too much. With these edited definitions I used `Question_3b` to produce figures 3.8 and 3.9 (found on page 9) and we see that both mass and energy are conserved via the numerical method.

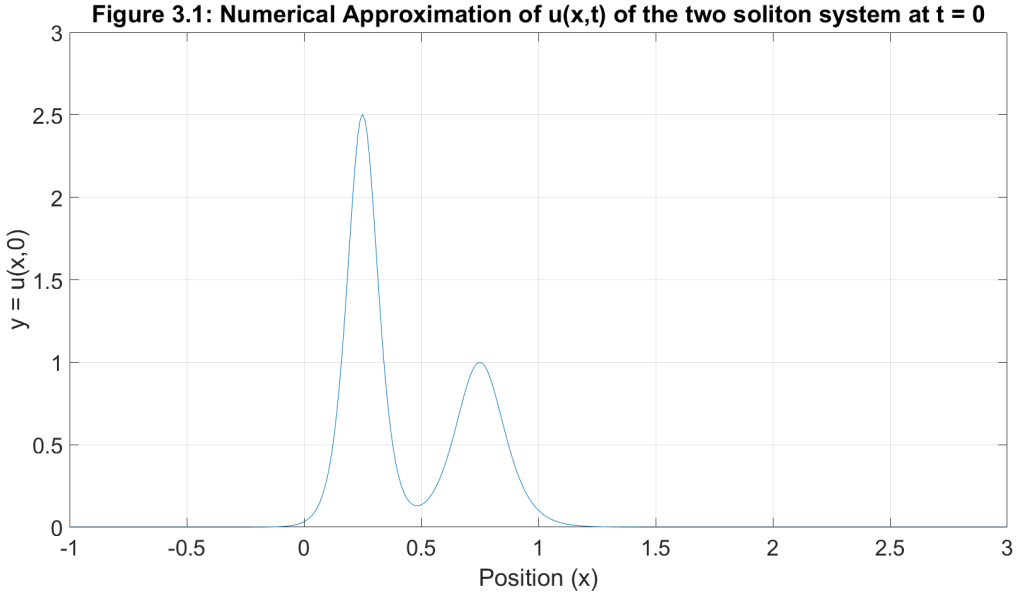


Figure 3.2: Numerical Approximation of u(x,t) of the two soliton system at t = 0.4

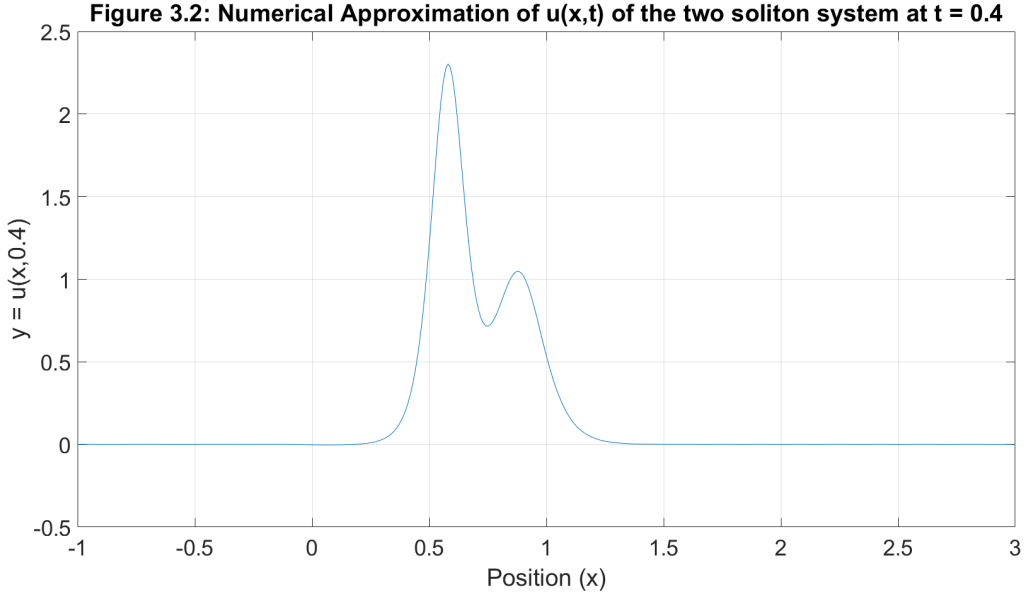


Figure 3.3: Numerical Approximation of u(x,t) of the two soliton system at t = 0.66

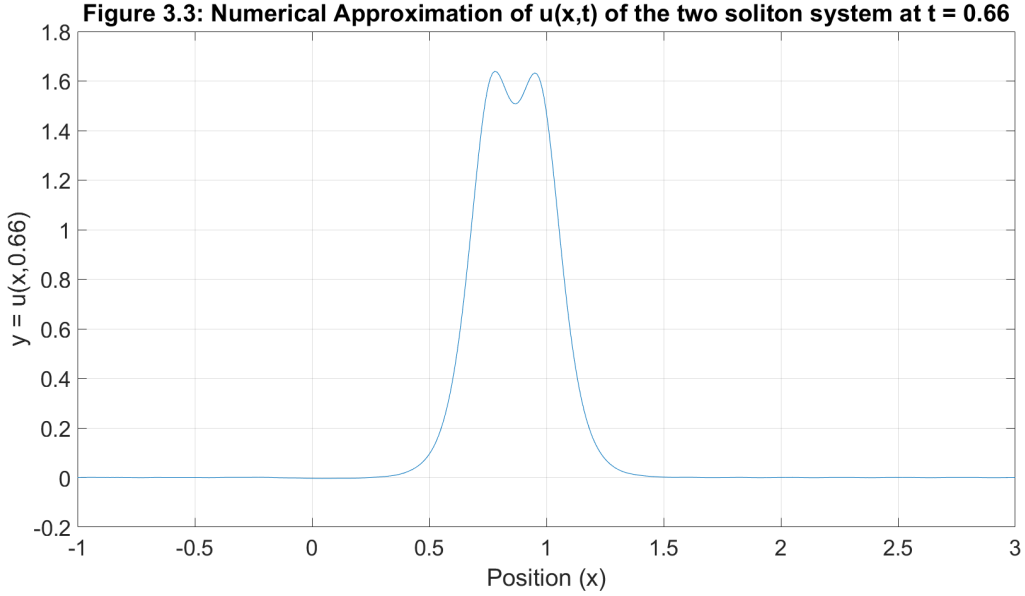


Figure 3.4: Numerical Approximation of u(x,t) of the two soliton system at t = 0.9

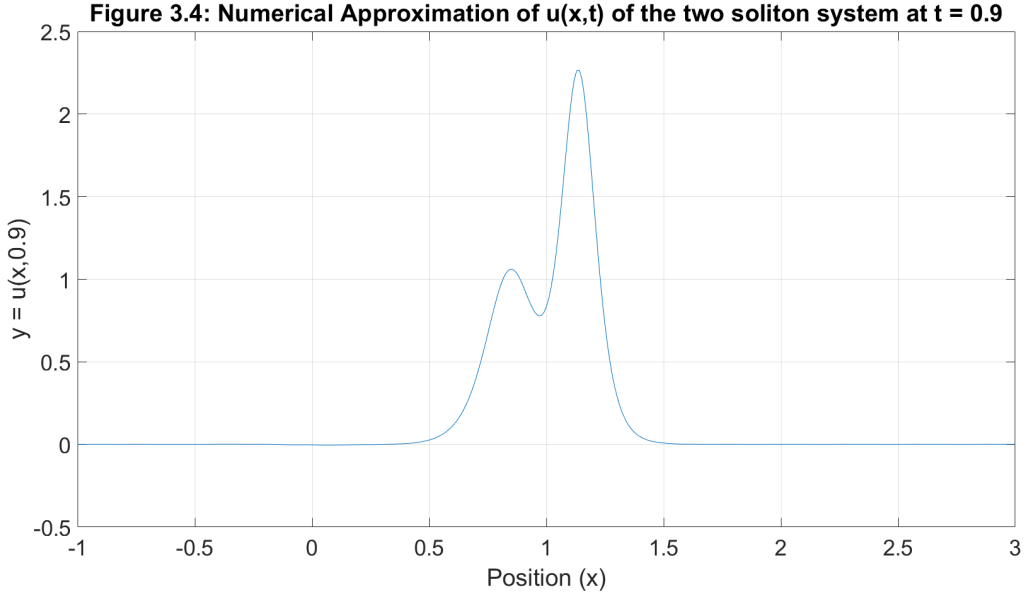


Figure 3.5: Numerical Approximation of u(x,t) of the two soliton system at t = 2

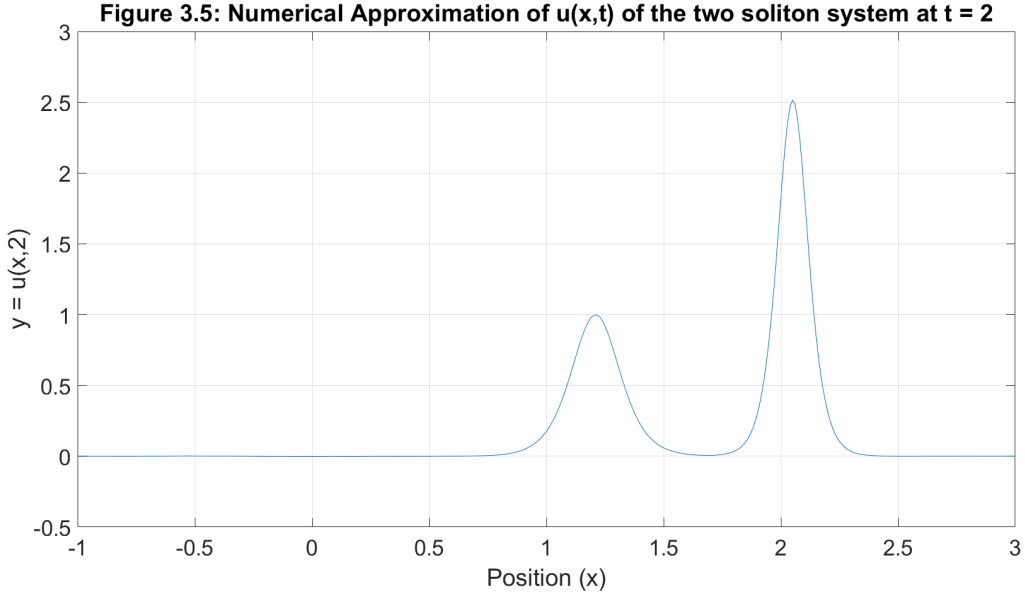


Figure 3.6: Numerical Approximation of Mass (M - as defined by (4)) as time varies

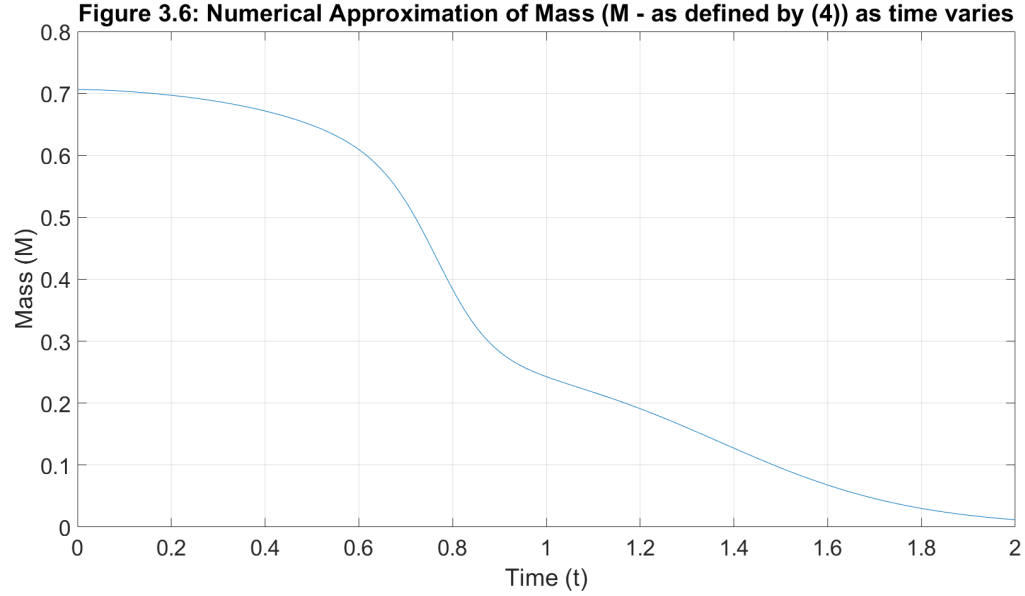


Figure 3.7: Numerical Approximation of Energy (E - as defined by (4)) as time varies

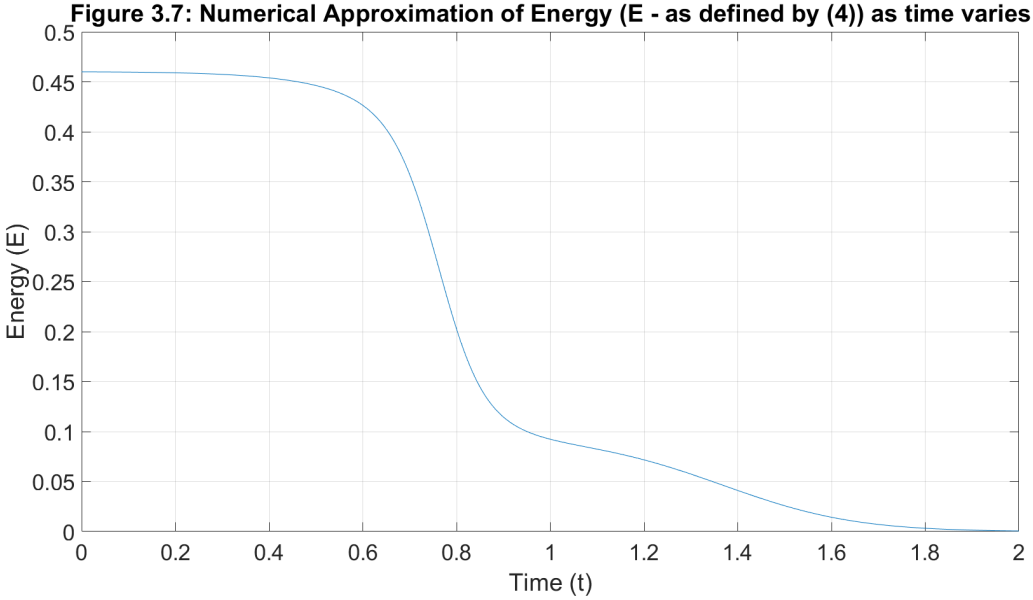


Figure 3.8: Numerical Approximation of Mass (M_c - as defined by (*)) as time varies

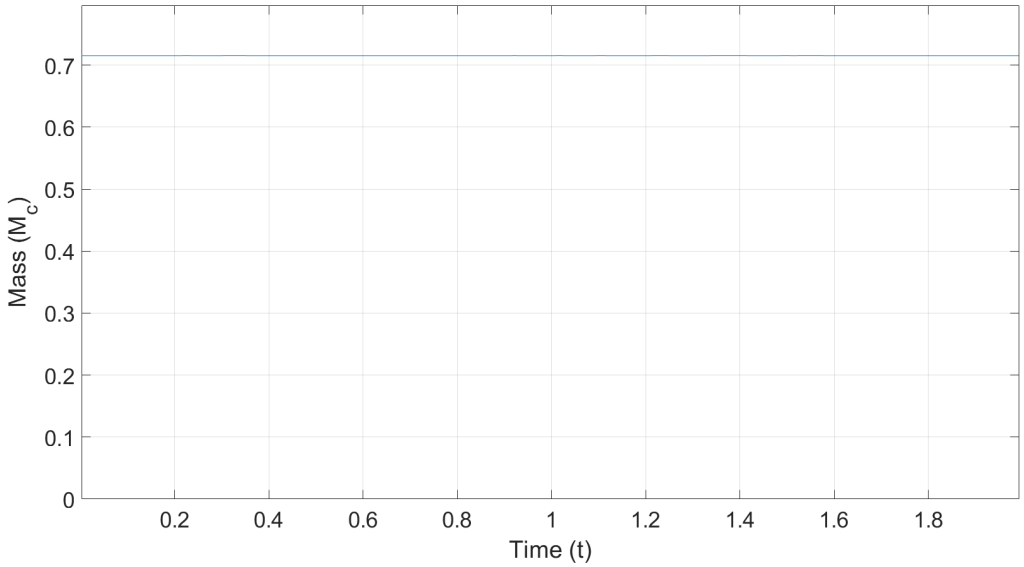
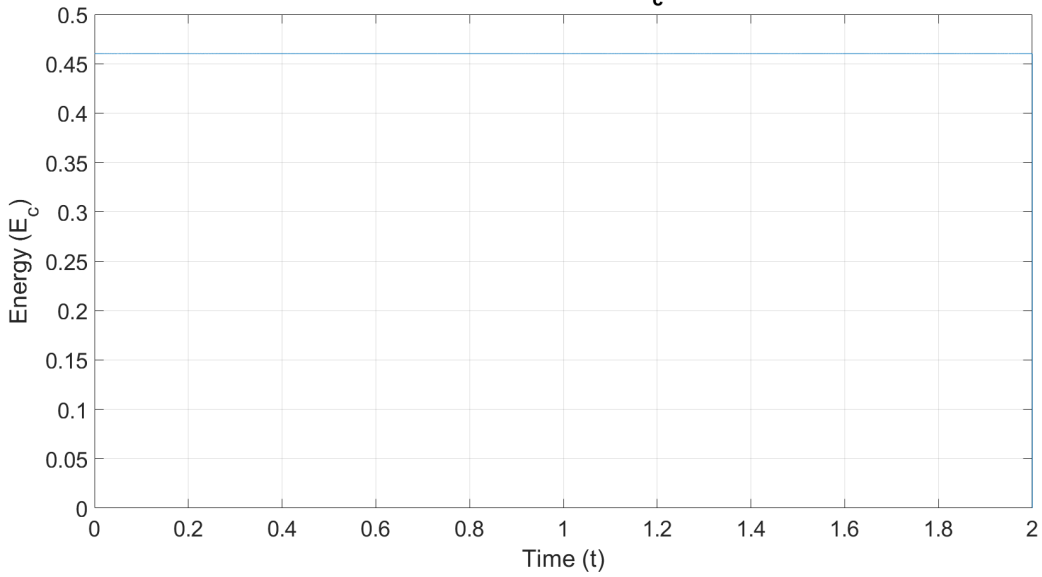


Figure 3.9: Numerical Approximation of Energy (E_c - as defined by (*)) as time varies



Question 4

We are now asked to consider the cyclic boundary conditions:

$$u(x,0) = \sin(2\pi x).$$

In the case $\delta = 0$ the differential equation is reduced to $u_t + uu_x = 0$. This can be solved using method of characteristics analytically, through which we obtain the solution:

$$u(x,t) = \sin(2\pi\zeta) \quad \text{with } \zeta \text{ satisfying:} \quad \zeta = 2\sin(2\pi\zeta).$$

This means that:

$$u_x = 2\pi\zeta_x \cos(2\pi\zeta)$$

and via our initial equation for ζ:

$$\zeta_x = \frac{1}{1 + 2\pi t\cos(2\pi\zeta)}.$$

So at time:

$$t^* = \min_{\zeta} \frac{-1}{2\pi\cos(2\pi\zeta)}$$

the slope u_x becomes infinite.

With this in mind we can describe the evolution. I will describe the motion in the region $[0,1]$ - since we have cyclic boundary conditions we can only need to consider this region. Initially the wave is sinusoidal. As time increases the wave starts to become more and more lopsided until time t^* where the wave there exists a point with a vertical slope. We see this numerically in figures 4.1, 4.2 and 4.3 (found on pages 11 and 12). This effect is known as *wave-breaking*.

This system has cyclic boundary conditions so we can adjust our script to take advantage of this. These adjustments can be found in the program `Question_4` (found on page 24). Note we only need to output $x \in [0,1]$ and we can deduce the values of $u(x,t)$ for all x outside this region as a result of the cyclic boundary condition.

I now set `delta = 0.04` and run Question 4. Initially we see a sinusoidal curve as the boundary conditions require this. In figure 4.4 we find that the wave becomes lopsided initially displaying signs of wave-breaking. As time increases we see that the dispersion effect dominates and the wave doesn't break. The dispersive effect is due to the u_{xxx} term in the differential equation and the wave-breaking effect is due to the uu_x terms. Initially u_{xxx} is very small but as the slope gets straighter the magnitude of this value increases so the amount of dispersion increases. At around time $t \approx 0.3$ we see that the dispersion effect starts to dominate (the slope gets no steeper and the wave is dispersing - see figure 4.5). This causes the u_{xxx} term to reduce in magnitude as the graph looks more smooth (see figure 4.6). As seen in figure 4.7, the recession of this term results in the curve having very steep section (i.e. displays some wave-breaking effect again) and the cycle continues (i.e. the wave disperses again and so on). The final thing worth observing is the highest peak moves from left to right. Due to the cyclic boundary conditions this means, as the wave exists the figure on the right a new wave starts to enter from the left (as shown in figure 4.8).The figures mentioned here can be found on pages 12 - 14.

For values of delta s.t.$\delta \in [0, 0.01)$ we find very similar to the behaviour found for $\delta = 0$. For δ between 0.01 and 0.04 we find that the wave breaks into multiple waves (as shown on figure 4.9 found on page 15).For values slightly larger than 0.04 we find behaviour similar to what we experienced for $\delta = 0.04$. For values much larger than $\delta = 0.04$ we find that the wave more like a purely dispersive wave which dispersive very quickly (as seen on figure 4.10 found on page 15).

Figure 4.1: Numerical Approximation of u(x,t) with cyclic boundary conditions and δ = 0 at t = 0

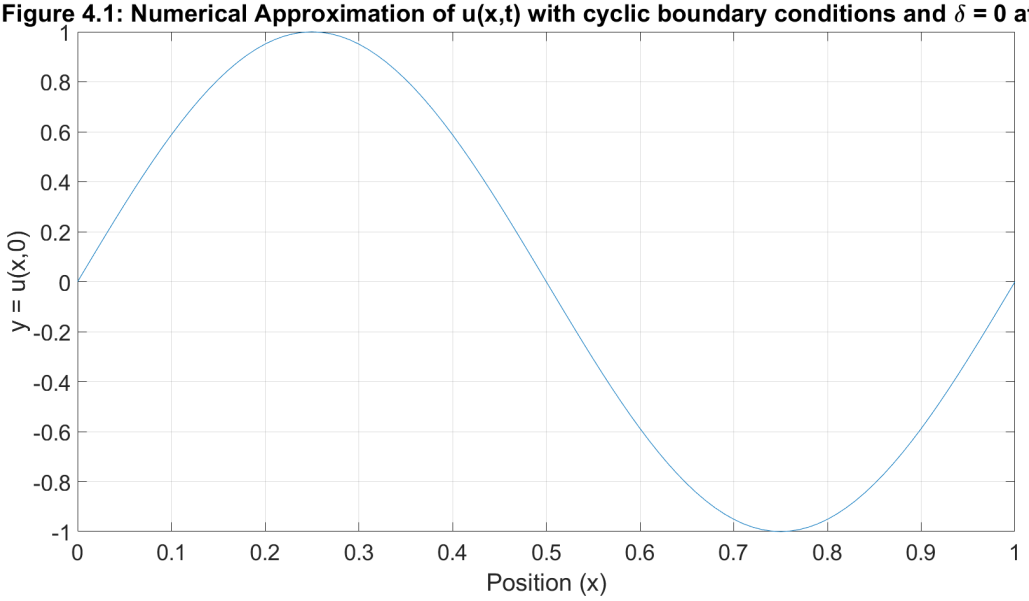


Figure 4.2: Numerical Approximation of u(x,t) with cyclic boundary conditions and δ = 0 at t = 0.1

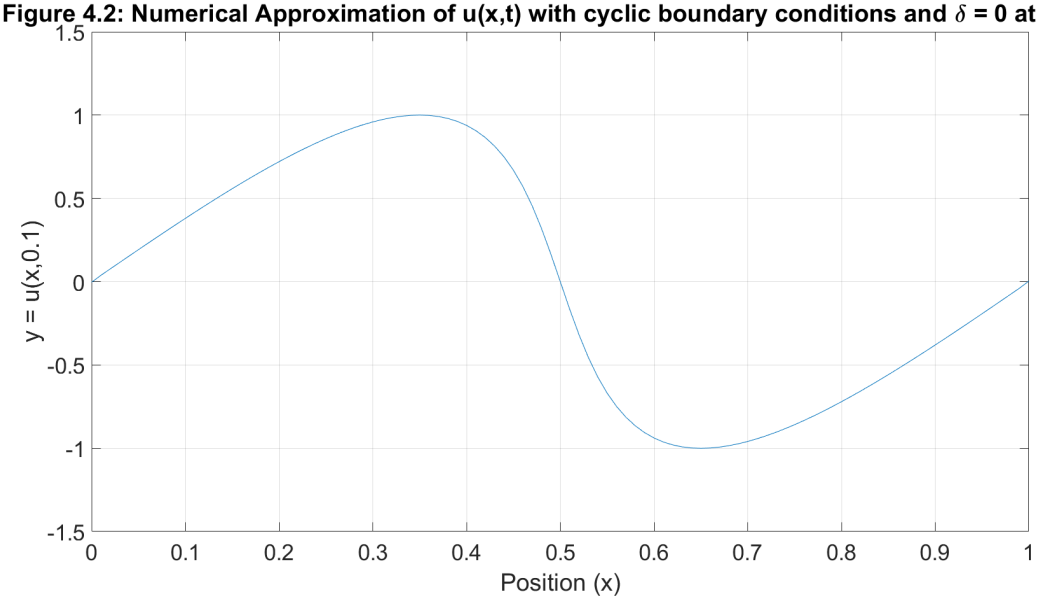


Figure 4.3: Numerical Approximation of u(x,t) with cyclic boundary conditions and δ = 0 at t = 0.2

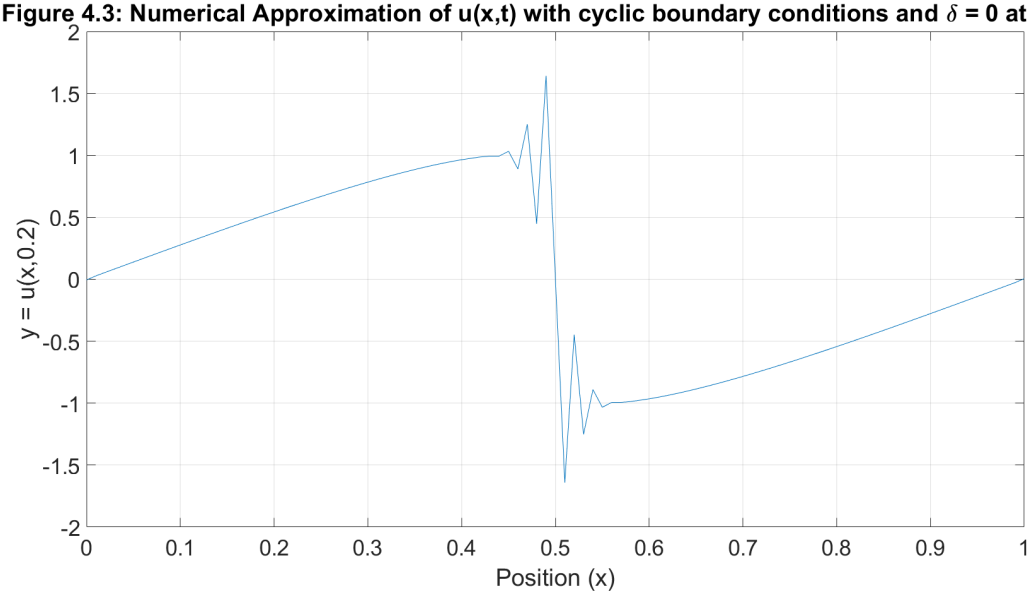


Figure 4.4: Numerical Approximation of u(x,t) with cyclic boundary conditions and δ = 0.04 at t = 0.1

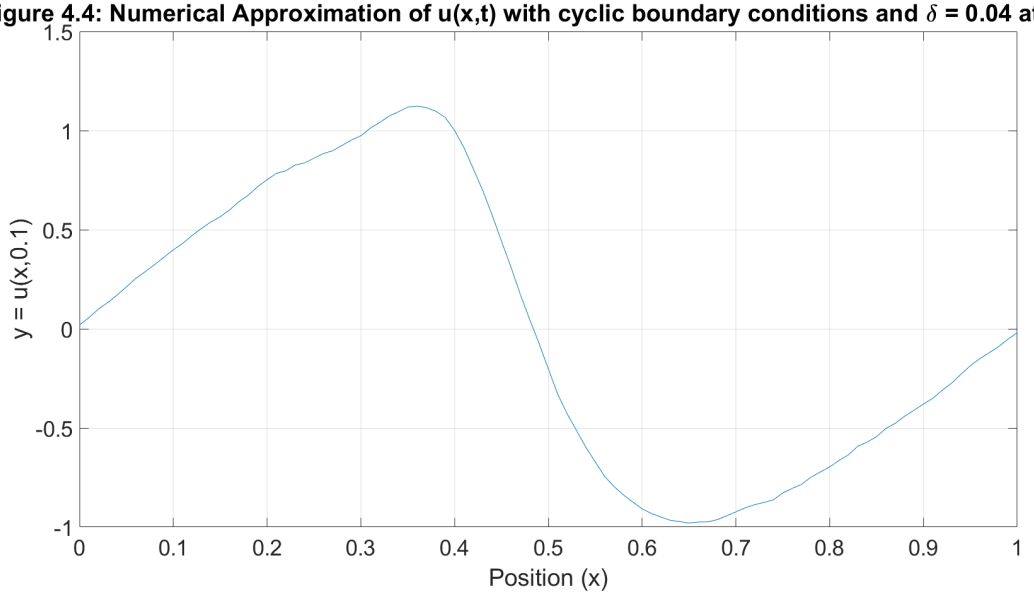


Figure 4.5: Numerical Approximation of u(x,t) with cyclic boundary conditions and δ = 0.04 at t = 0.3

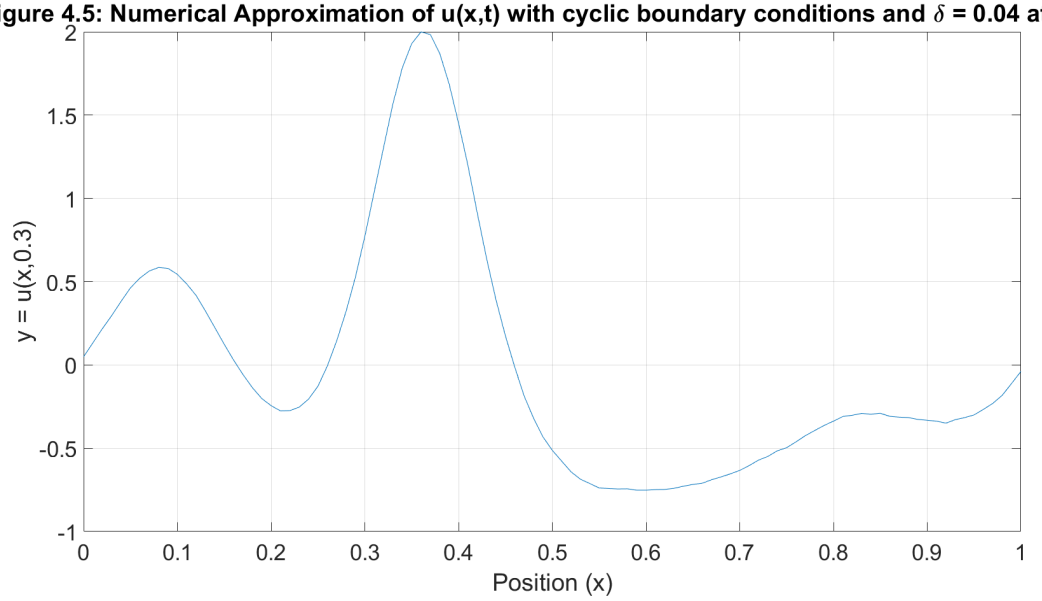


Figure 4.6: Numerical Approximation of u(x,t) with cyclic boundary conditions and δ = 0.04 at t = 0.6

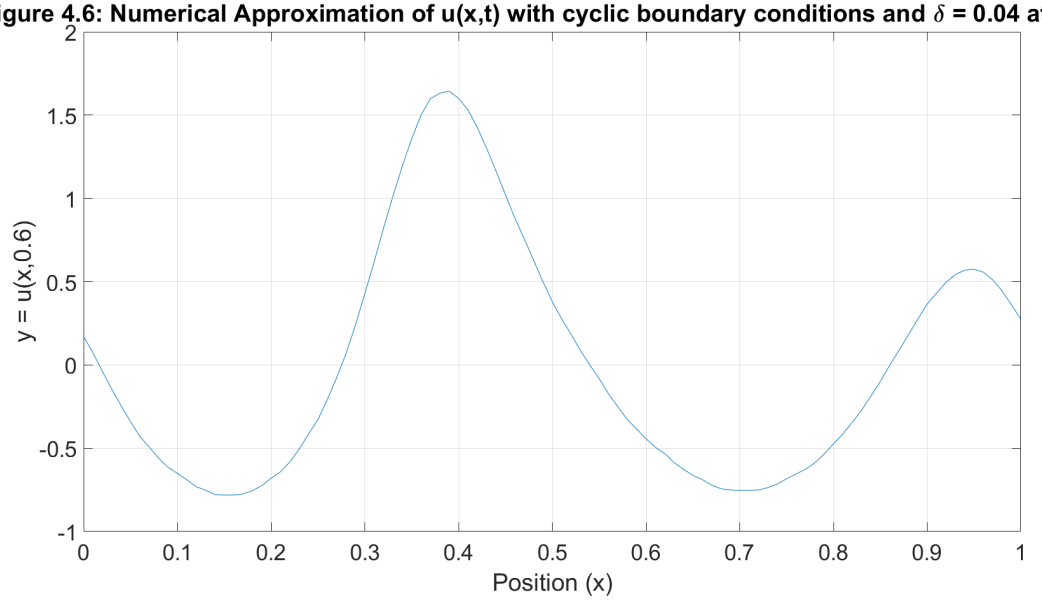


Figure 4.7: Numerical Approximation of u(x,t) with cyclic boundary conditions and δ = 0.04 at t = 0.9

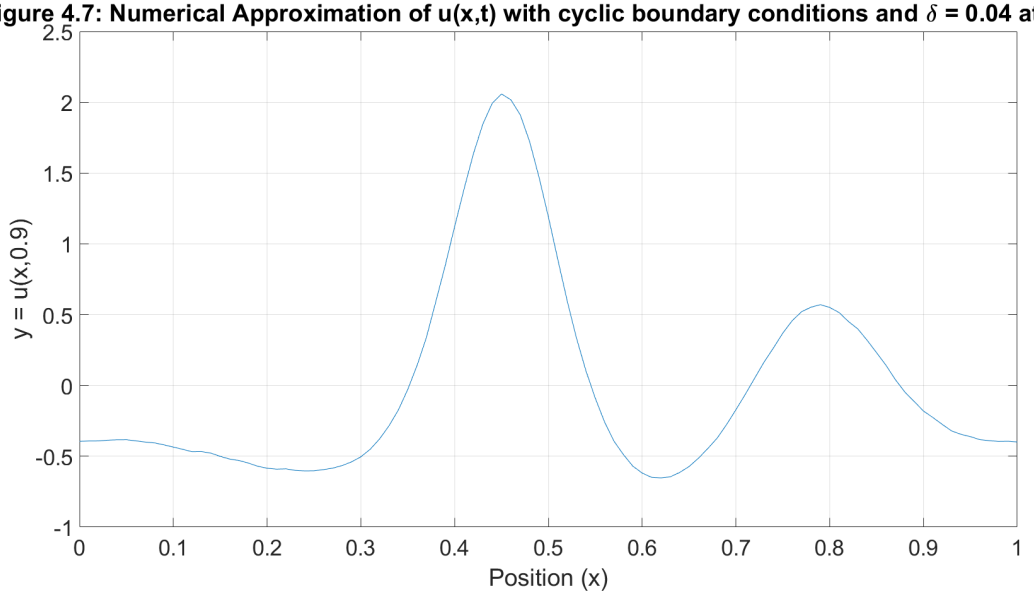


Figure 4.8: Numerical Approximation of u(x,t) with cyclic boundary conditions and δ = 0.04 at t = 3

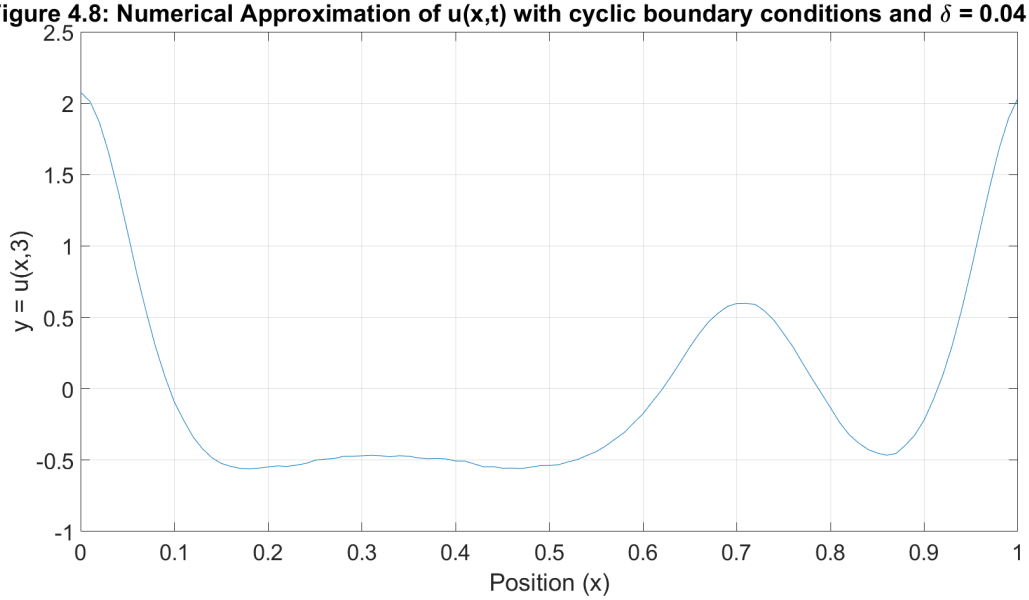


Figure 4.9: Numerical Approximation of u(x,t) with cyclic boundary conditions and δ = 0.02 at t = 2

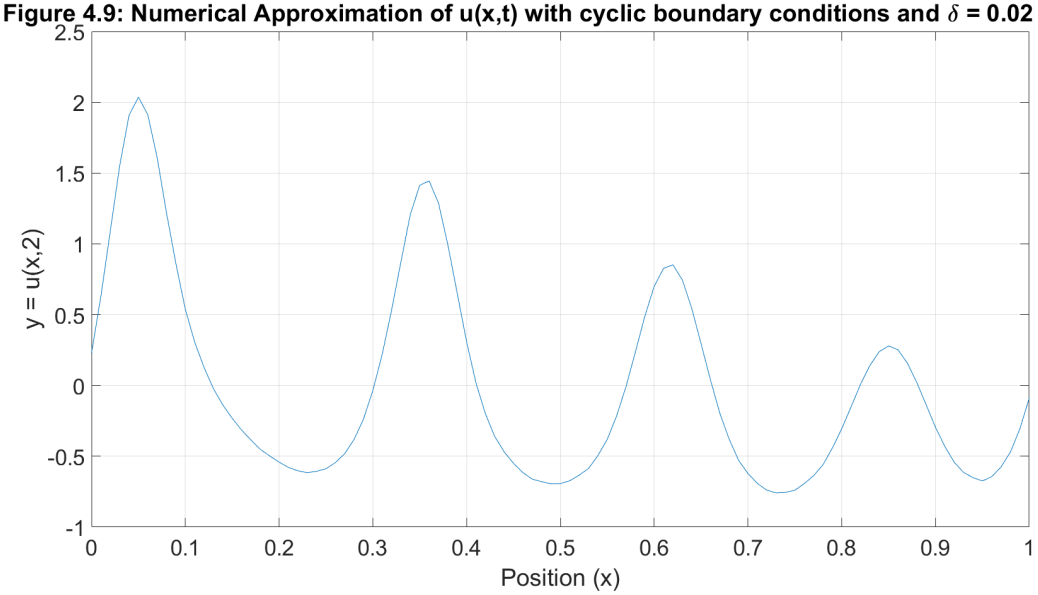
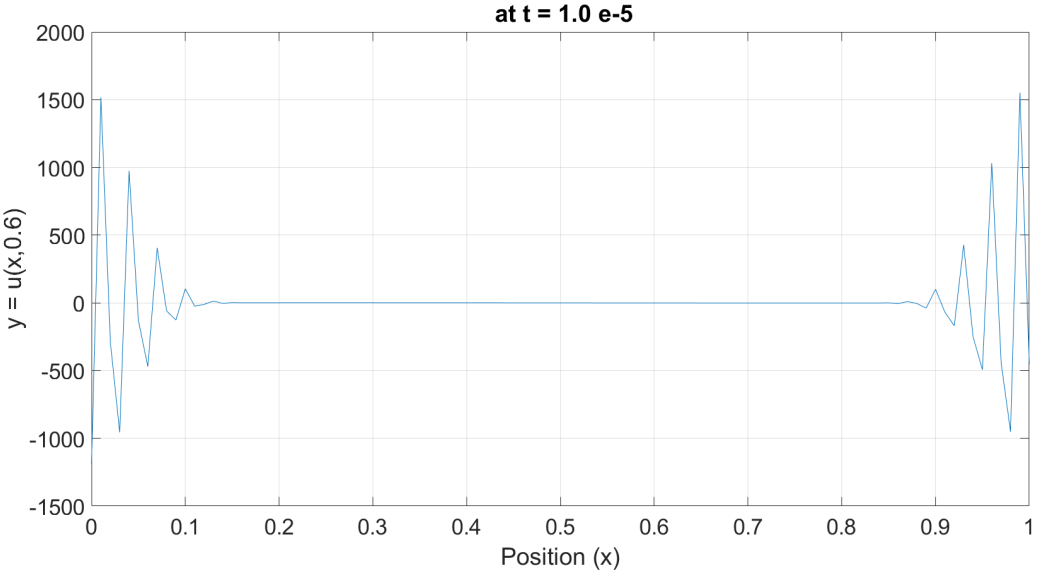


Figure 4.10: Numerical Approximation of u(x,t) with cyclic boundary conditions and δ = 0.9 at t = 1.0 e-5



References

[1] Drazin, P.G. & Johnson, R.S. (1989) *Solitons: An Introduction*, CUP.

[2] Greig, I. S. & Morris, J. L. (1976) *A hopscotch method for the Korteweg-de Vries equation* Journal of Computational Physics, volume 20, pages 64-80.

Appendix: Question 2

This is labelled `Question_2` in folders and used to apply Zabusky & Kruskal's leap-frog scheme.

```
clear; clc;
format long
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Constants
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
h = 0.01; %x = hm     0.01
k =1e-6; %t = kn

x_start = 0;
x_end = 2;

t_start = 0;
t_end = 1;

m_max = (x_end - x_start)/h;
n_max = (t_end - t_start)/k;

A = 2.5;
delta = 0.04;
x0 = 0.75;

c = A/3;
Delta = delta*(sqrt(12/A));

%(h^3)/(4*(delta^2) + A*(h^2))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Equations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%initial conditions
u_initial = @(x) A*(sech((x - x0)/Delta))^2;

%numerical scheme:
u = @(um2, um1, u0, u1, u2, ut) ut - (k/(3*h))*(u1 + u0 + um1)*
(u1 - um1) - ((k*(delta^2))/(h^3))*(u2 - 2*u1 + 2*um1 - um2) ;

%Exact Solution
f = @(x) A*(sech((x-x0)/Delta))^2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Leap Frog Scheme
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = zeros(m_max+1,n_max+1);

%INITIAL TIME STEPS:
for i = 1:m_max+1;
M(i,1) = u_initial((i-1)*h);
end

%SECOND TIME STEPS:
%initial space steps:
M(1,2) = u(0,0, M(1,1), M(2,1), M(3,1), M(1,1));
M(2,2) = u(0, M(1,1), M(2,1), M(3,1), M(4,1), M(2,1));

%intermediate values
for i = 3:m_max -1;
```

```
M(i,2) = u(M(i-2,1), M(i-1,1), M(i,1), M(i+1,1),M(i+2,1),
M(i,1));
end

%final space steps:
M(m_max + 1, 2) =  u(M(m_max-1,1), M(m_max,1), M(m_max+1,1),0,
0,M(m_max+1,1));
M(m_max, 2) =  u(M(m_max-2,1), M(m_max-1,1), M(m_max,1),
M(m_max+1,1),0,M(m_max,1));

%FUTURE TIME STEPS:
for j = 3: n_max+1;
%initial space steps:
M(1,j) = u(0,0,M(1,j-1),M(2,j-1),M(3,j-1),M(1,j-2));
M(2,j) = u(0,M(1,j-1),M(2,j-1),M(3,j-1),M(4,j-1),M(2,j-2));

%intermediate values
for i = 3:m_max -1;
M(i,j) = u(M(i-2,j-1), M(i-1,j-1), M(i,j-1), M(i+1,j-1),
M(i+2,j-1),M(i,j-2));
end

%final space steps:
M(m_max + 1, j) =  u(M(m_max-1,j-1), M(m_max,j-1),
M(m_max+1,j-1),0,0,M(m_max+1,j-2));
M(m_max, j) =  u(M(m_max-2,j-1), M(m_max-1,j-1),
M(m_max,j-1),M(m_max+1,j-1),0,M(m_max,j-2));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%End
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Appendix: Question 2b

This is labelled `Question_2b` in folders and used to apply plot the graph required in Questions 2, 3 and 4.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Constants
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
time =  19; %input time that need output for
time2 = (time-t_start)/k;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plot
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x = x_start:h:x_end;
y = M(:,1+time2);
figure;
plot(x,y);
grid on;
hold on ;
z =  A*(sech((x - (c*time2*k) - x0)/Delta)).^2;
plot(x,z); %delete when necessary
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%End
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Appendix: Question 2c

This is labelled `Question_2c` in folders and used to find the global error in Question 2.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Constants
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
time = 0.75; %input time that need output for
time2 = (time-t_start)/k;
z = @(var) A*(sech((var - (c*time2*k) - x0)/Delta)).^2;
table = [];

for num = 1:1:m_max+1;
E = abs(M(num,time2+1)-z((num-1)*h))
table = [table; E];
end

max(table)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%End
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Appendix: Question 3

This is labelled `Question_3` in folders and used to apply Zabusky & Kruskal's leap-frog scheme.

```
clear; clc;
format long
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Constants
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
h = 0.01; %x = hm     0.01
k =1e-6; %t = kn

x_start = -1;
x_end = 3;

t_start = 0;
t_end = 2;

m_max = (x_end - x_start)/h;
n_max = (t_end - t_start)/k;

A1 = 2.5;
A2 = 1;
delta = 0.04;
x0_1 = 0.25;
x0_2 = 0.75;

c1 = A1/3;
c2 = A2/3;
Delta1 = delta*(sqrt(12/A1));
Delta2 = delta*(sqrt(12/A2));

%(h^3)/(4*(delta^2) + A*(h^2))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Equations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%initial conditions
u_initial = @(x) (A1*(sech((x - x0_1 +x_start)/Delta1))^2) +
(A2*(sech((x - x0_2+ x_start)/Delta2))^2);

%numerical scheme:
u = @(um2, um1, u0, u1, u2, ut) ut - (k/(3*h))*(u1 + u0 + um1)*
(u1 - um1) - ((k*(delta^2))/(h^3))*(u2 - 2*u1 + 2*um1 - um2) ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Leap Frog Scheme
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%M = zeros(1 + m_max,1 + n_max);
M = zeros(401,2.00001e5);

%INITIAL TIME STEPS:
for i = 1:m_max+1;
M(i,1) = u_initial((i-1)*h);
end

%SECOND TIME STEPS:
%initial space steps:
M(1,2) = u(0,0, M(1,1), M(2,1), M(3,1), M(1,1));
```

```
M(2,2) = u(0, M(1,1), M(2,1), M(3,1), M(4,1), M(2,1));

%intermediate values
for i = 3:m_max -1;
M(i,2) = u(M(i-2,1), M(i-1,1), M(i,1), M(i+1,1),M(i+2,1),M(i,1));
end

%final space steps:
M(m_max + 1, 2) =  u(M(m_max-1,1), M(m_max,1), M(m_max+1,1),0,0
,M(m_max+1,1));
M(m_max, 2) =  u(M(m_max-2,1), M(m_max-1,1), M(m_max,1),
M(m_max+1,1),0,M(m_max,1));

%FUTURE TIME STEPS:
for j = 3: n_max+1;
%initial space steps:
M(1,j) = u(0,0,M(1,j-1),M(2,j-1),M(3,j-1),M(1,j-2));
M(2,j) = u(0,M(1,j-1),M(2,j-1),M(3,j-1),M(4,j-1),M(2,j-2));

%intermediate values
for i = 3:m_max -1;
M(i,j) = u(M(i-2,j-1), M(i-1,j-1), M(i,j-1), M(i+1,j-1),
M(i+2,j-1),M(i,j-2));
end

%final space steps:
M(m_max + 1, j) =  u(M(m_max-1,j-1), M(m_max,j-1), M(m_max+1,j-1),
0,0,M(m_max+1,j-2));
M(m_max, j) =  u(M(m_max-2,j-1), M(m_max-1,j-1), M(m_max,j-1),
M(m_max+1,j-1),0,M(m_max,j-2));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%End
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Appendix: Question 3b

This is labelled `Question_3b` in folders find and plot the mass and energy of the system against time.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Trapezium Rule
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

vec_mass = [];
vec_energy = [];
j = 1
while j < n_max+2;
table = [];
%i = 101;
i = 1;
Mass = 0;
Energy = 0;
Mass = 0.5*h*M(1,j);
Energy = 0.25*h*((M(1,j))^2);
%i = 102;
i = 2;

while i < m_max + 1;  % 201
Mass = Mass + h*M(i,j);
Energy = Energy + 0.5*h*((M(i,j))^2);;
i = i + 1;
end

Mass = Mass  + 0.5*h*M(m_max + 1, j);
Energy = Energy + 0.25*h*((M(m_max + 1,j))^2);
vec_mass = [vec_mass ; Mass];
vec_energy = [vec_energy ; Energy];
j = j+1

end

t = t_start:k:t_end;
plot(t,vec_mass);
title('vecmass')
grid on;
figure;
plot(t,vec_energy);
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%End
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Appendix: Question 4

This is labelled `Question_4` in folders and used to apply Zabusky & Kruskal's leap-frog scheme.

```
clear; clc;
format long
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Constants
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
h = 0.01; %x = hm     0.01
k =1e-6; %t = kn

x_start = 0;
x_end = 1;

t_start = 0;
t_end = 20;

m_max = (x_end - x_start)/h;
n_max = (t_end - t_start)/k;

delta = 0.04;


%(h^3)/(4*(delta^2) + A*(h^2))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Equations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%initial conditions
u_initial = @(x) sin(2*pi*x);

%numerical scheme:
u = @(um2, um1, u0, u1, u2, ut) ut - (k/(3*h))*(u1 + u0 + um1)*
(u1 - um1) - ((k*(delta^2))/(h^3))*(u2 - 2*u1 + 2*um1 - um2) ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Leap Frog Scheme
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = zeros(m_max+1,n_max+1);

%INITIAL TIME STEPS:
for i = 1:m_max+1;
M(i,1) = u_initial((i-1)*h);
end

%SECOND TIME STEPS:
%initial space steps:
M(1,2) = u(M(m_max,1),M(m_max+1,1), M(1,1), M(2,1), M(3,1), M(1,1));
M(2,2) = u(M(m_max+1,1), M(1,1), M(2,1), M(3,1), M(4,1), M(2,1));

%intermediate values
for i = 3:m_max -1;
M(i,2) = u(M(i-2,1), M(i-1,1), M(i,1), M(i+1,1),M(i+2,1),M(i,1));
end

%final space steps:
M(m_max + 1, 2) =  u(M(m_max-1,1), M(m_max,1), M(m_max+1,1),M(1,1),
 M(2,1),M(m_max+1,1));
```

```
M(m_max, 2) =  u(M(m_max-2,1), M(m_max-1,1), M(m_max,1),M(m_max+1,1),
M(1,1),M(m_max,1));

%FUTURE TIME STEPS:
for j = 3: n_max+1;
%initial space steps:
M(1,j) = u(M(m_max,j-1),M(m_max+1,j-1),M(1,j-1),M(2,j-1),M(3,j-1),
M(1,j-2));
M(2,j) = u( M(m_max+1,j-1),M(1,j-1),M(2,j-1),M(3,j-1),M(4,j-1),
M(2,j-2));

%intermediate values
for i = 3:m_max -1;
M(i,j) = u(M(i-2,j-1), M(i-1,j-1), M(i,j-1), M(i+1,j-1),M(i+2,j-1),
M(i,j-2));
end

%final space steps:
M(m_max + 1, j) =  u(M(m_max-1,j-1), M(m_max,j-1), M(m_max+1,j-1),
M(1,j-1),M(2,j-1),M(m_max+1,j-2));
M(m_max, j) =  u(M(m_max-2,j-1), M(m_max-1,j-1), M(m_max,j-1),
M(m_max+1,j-1),M(1,j-1),M(m_max,j-2));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%End
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

END