2.7 Soliton Solutions of the KdV Equation

QUESTION 1

We wish to verify that the single soliton (2) is indeed a [non-periodic] solution of the KdV equation.

We are given that $u(x,t) = Asech^2\left(\frac{x-ct-x_0}{\Delta}\right)$. Begin by computing the partial derivatives of $u(x,t)$:

$$u_t = -2Asech^2\left(\frac{x-ct-x_0}{\Delta}\right)tanh\left(\frac{x-ct-x_0}{\Delta}\right)\times\frac{-c}{\Delta}$$
$$= \frac{6c^2}{\Delta}sech^2\left(\frac{x-ct-x_0}{\Delta}\right)tanh\left(\frac{x-ct-x_0}{\Delta}\right)$$

$$u_x = -2Asech^2\left(\frac{x-ct-x_0}{\Delta}\right)tanh\left(\frac{x-ct-x_0}{\Delta}\right)\times\frac{1}{\Delta}$$
$$= -\frac{6c}{\Delta}sech^2\left(\frac{x-ct-x_0}{\Delta}\right)tanh\left(\frac{x-ct-x_0}{\Delta}\right)$$

$$u_{xx} = -\frac{6c}{\Delta}\frac{\partial}{\partial x}\left(sech^2\left(\frac{x-ct-x_0}{\Delta}\right)tanh\left(\frac{x-ct-x_0}{\Delta}\right)\right)$$
$$= -\frac{6c}{\Delta^2}\left(sech^4\left(\frac{x-ct-x_0}{\Delta}\right) - 2sech^2\left(\frac{x-ct-x_0}{\Delta}\right)tanh^2\left(\frac{x-ct-x_0}{\Delta}\right)\right)$$

$$u_{xxx} = -\frac{6c}{\Delta^2}\frac{\partial}{\partial x}\left(sech^4\left(\frac{x-ct-x_0}{\Delta}\right) - 2sech^2\left(\frac{x-ct-x_0}{\Delta}\right)tanh^2\left(\frac{x-ct-x_0}{\Delta}\right)\right)$$
$$= -\frac{6c}{\Delta^3}\left(-8sech^4\left(\frac{x-ct-x_0}{\Delta}\right)tanh\left(\frac{x-ct-x_0}{\Delta}\right)\right.$$
$$\left. + 4sech^2\left(\frac{x-ct-x_0}{\Delta}\right)tanh^3\left(\frac{x-ct-x_0}{\Delta}\right)\right)$$

Substitute these into the KdV equation:

$$u_t + uu_x + \delta^2 u_{xxx}$$
$$= \frac{6c^2}{\Delta}sech^2\left(\frac{x-ct-x_0}{\Delta}\right)tanh\left(\frac{x-ct-x_0}{\Delta}\right)$$
$$-\frac{18c^2}{\Delta}sech^4\left(\frac{x-ct-x_0}{\Delta}\right)tanh\left(\frac{x-ct-x_0}{\Delta}\right)$$
$$-\frac{3c^2}{2\Delta}\left(-8sech^4\left(\frac{x-ct-x_0}{\Delta}\right)tanh\left(\frac{x-ct-x_0}{\Delta}\right)\right.$$
$$\left. + 4sech^2\left(\frac{x-ct-x_0}{\Delta}\right)tanh^3\left(\frac{x-ct-x_0}{\Delta}\right)\right)$$
$$= \frac{6c^2}{\Delta}sech^2\left(\frac{x-ct-x_0}{\Delta}\right)tanh\left(\frac{x-ct-x_0}{\Delta}\right)$$
$$-\frac{18c^2}{\Delta}sech^4\left(\frac{x-ct-x_0}{\Delta}\right)tanh\left(\frac{x-ct-x_0}{\Delta}\right)$$
$$+\frac{12c^2}{\Delta}sech^4\left(\frac{x-ct-x_0}{\Delta}\right)tanh\left(\frac{x-ct-x_0}{\Delta}\right)$$
$$-\frac{6c^2}{\Delta}sech^2\left(\frac{x-ct-x_0}{\Delta}\right)tanh^3\left(\frac{x-ct-x_0}{\Delta}\right)$$

Using the identity $sech^2\left(\frac{x-ct-x_0}{\Delta}\right) + tanh^2\left(\frac{x-ct-x_0}{\Delta}\right) = 1$, rewrite the last term:

$$
\begin{aligned}
-\frac{6c^2}{\Delta} sech^2\left(\frac{x-ct-x_0}{\Delta}\right) tanh^3\left(\frac{x-ct-x_0}{\Delta}\right) \\
&= -\frac{6c^2}{\Delta} sech^2\left(\frac{x-ct-x_0}{\Delta}\right) tanh\left(\frac{x-ct-x_0}{\Delta}\right)\left[1 - sech^2\left(\frac{x-ct-x_0}{\Delta}\right)\right] \\
&= -\frac{6c^2}{\Delta} sech^2\left(\frac{x-ct-x_0}{\Delta}\right) tanh\left(\frac{x-ct-x_0}{\Delta}\right) \\
&\quad + \frac{6c^2}{\Delta} sech^4\left(\frac{x-ct-x_0}{\Delta}\right) tanh\left(\frac{x-ct-x_0}{\Delta}\right)
\end{aligned}
$$

Substitute this back in:

$$
\begin{aligned}
u_t + uu_x + \delta^2 u_{xxx} \\
&= \frac{6c^2}{\Delta} sech^2\left(\frac{x-ct-x_0}{\Delta}\right) tanh\left(\frac{x-ct-x_0}{\Delta}\right) \\
&\quad - \frac{18c^2}{\Delta} sech^4\left(\frac{x-ct-x_0}{\Delta}\right) tanh\left(\frac{x-ct-x_0}{\Delta}\right) \\
&\quad + \frac{12c^2}{\Delta} sech^4\left(\frac{x-ct-x_0}{\Delta}\right) tanh\left(\frac{x-ct-x_0}{\Delta}\right) \\
&\quad - \frac{6c^2}{\Delta} sech^2\left(\frac{x-ct-x_0}{\Delta}\right) tanh^3\left(\frac{x-ct-x_0}{\Delta}\right) \\
&= \frac{6c^2}{\Delta} sech^2\left(\frac{x-ct-x_0}{\Delta}\right) tanh\left(\frac{x-ct-x_0}{\Delta}\right) \\
&\quad - \frac{18c^2}{\Delta} sech^4\left(\frac{x-ct-x_0}{\Delta}\right) tanh\left(\frac{x-ct-x_0}{\Delta}\right) \\
&\quad + \frac{12c^2}{\Delta} sech^4\left(\frac{x-ct-x_0}{\Delta}\right) tanh\left(\frac{x-ct-x_0}{\Delta}\right) \\
&\quad - \frac{6c^2}{\Delta} sech^2\left(\frac{x-ct-x_0}{\Delta}\right) tanh\left(\frac{x-ct-x_0}{\Delta}\right) \\
&\quad + \frac{6c^2}{\Delta} sech^4\left(\frac{x-ct-x_0}{\Delta}\right) tanh\left(\frac{x-ct-x_0}{\Delta}\right) = 0
\end{aligned}
$$

Hence this solution satisfies the KdV equation, as required.

Now consider the mass, M, and the energy, E, of the motion, defined by the following:

$$M = \int_0^1 u(x,t)\, dx \qquad\qquad E = \int_0^1 \frac{1}{2} u^2(x,t)\, dx$$

We aim to show that both of these quantities are time independent. Start with mass, and begin by taking the time derivative:

$$\frac{dM}{dt} = \frac{d}{dt}\int_0^1 u(x,t)\, dx = \int_0^1 u_t(x,t)\, dx = -\int_0^1 uu_x + \delta^2 u_{xxx}\, dx = -\left[\frac{u^2}{2} + \delta^2 u_{xx}\right]_0^1$$

Note that $u(1,t) = u(0,t)$ from the periodicity of $u(x,t)$, hence $\left[\frac{u^2}{2}\right]_0^1 = 0$.

Now for the second term, recall that

$$u_{xx} = -\frac{6c}{\Delta^2}\left(sech^4\left(\frac{x-ct-x_0}{\Delta}\right) - 2sech^2\left(\frac{x-ct-x_0}{\Delta}\right) tanh^2\left(\frac{x-ct-x_0}{\Delta}\right)\right)$$

Due to the periodicity of $u(x,t)$, we have that $sech^2\left(\frac{1-ct-x_0}{\Delta}\right) = sech^2\left(\frac{-ct-x_0}{\Delta}\right)$, and similarly $sech^4\left(\frac{1-ct-x_0}{\Delta}\right) = sech^4\left(\frac{-ct-x_0}{\Delta}\right)$ (by just squaring both sides). Therefore, from the identity $sech^2\left(\frac{x-ct-x_0}{\Delta}\right) + tanh^2\left(\frac{x-ct-x_0}{\Delta}\right) = 1$, $tanh^2\left(\frac{1-ct-x_0}{\Delta}\right) = tanh^2\left(\frac{-ct-x_0}{\Delta}\right)$ too. Hence, from the definition of u_{xx}, we have that $u_{xx}(1,t) = u_{xx}(0,t)$ too, meaning that the second term$[\delta^2 u_{xx}]_0^1 = 0$ as well. Hence $\frac{dM}{dt} = 0$.

Now focus on the energy quantity, and once again try to take the time derivative:

$$\frac{dE}{dt} = \frac{d}{dt}\int_0^1 \frac{1}{2}u^2(x,t)\,dx = \int_0^1 u(x,t)u_t(x,t)\,dx$$

Multiply the KdV equation throughout by u:

$$uu_t + u^2u_x + \delta^2 uu_{xxx} = 0$$

Rearranging this equation, we get that

$$uu_t = -u^2u_x - \delta^2 uu_{xxx}$$

Substitute this into the integral:

$$\frac{dE}{dt} = -\int_0^1 u^2u_x + \delta^2 uu_{xxx}\;dx$$

Starting by focusing on the first term, using reverse chain rule, we get $\left[\frac{u^3}{3}\right]_0^1$, which is equal to zero due to the periodicity of u. Then use integration by parts to find the second term:

$$\int_0^1 uu_{xxx}\;dx = [uu_{xx}]_0^1 - \int_0^1 u_xu_{xx}\;dx = [uu_{xx}]_0^1 - \left[\frac{{u_x}^2}{2}\right]_0^1$$

We previously showed that u & u_{xx} are both periodic, so boundary term $[uu_{xx}]_0^1 = 0$. Similarly can show that ${u_x}^2$ is periodic too:

$$u_x = -\frac{6c}{\Delta}sech^2\left(\frac{x-ct-x_0}{\Delta}\right)tanh\left(\frac{x-ct-x_0}{\Delta}\right)$$

$${u_x}^2 \propto sech^4\left(\frac{x-ct-x_0}{\Delta}\right)tanh^2\left(\frac{x-ct-x_0}{\Delta}\right)$$

As stated previously, $sech^4\left(\frac{1-ct-x_0}{\Delta}\right) = sech^4\left(\frac{-ct-x_0}{\Delta}\right)$ and $tanh^2\left(\frac{1-ct-x_0}{\Delta}\right) = tanh^2\left(\frac{-ct-x_0}{\Delta}\right)$, so ${u_x}^2(1,t) = {u_x}^2(0,t)$. Therefore, $\left[\frac{{u_x}^2}{2}\right]_0^1 = 0$ too, and so $\frac{dE}{dt} = 0$ as well.

Hence both M and E are time independent.

QUESTION 2

If we let u_m^n be the solution at $x = hm$ and $t = kn$ with $n, m = 0, 1, \ldots$ then the KdV equation can be discretised as:

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

We aim to find the order of this scheme.

To do this, Taylor expand the non u_m^n terms:

- $u_m^{n+1} = u_m^n + k(u_t)_m^n + \frac{k^2}{2}(u_{tt})_m^n + O(k^3)$
- $u_m^{n-1} = u_m^n - k(u_t)_m^n + \frac{k^2}{2}(u_{tt})_m^n + O(k^3)$
- $u_{m+1}^n = u_m^n + h(u_x)_m^n + \frac{h^2}{2}(u_{xx})_m^n + O(h^3)$
- $u_{m-1}^n = u_m^n - h(u_x)_m^n + \frac{h^2}{2}(u_{xx})_m^n + O(h^3)$
- $u_{m+2}^n = u_m^n + 2h(u_x)_m^n + 2h^2(u_{xx})_m^n + O(h^3)$
- $u_{m-2}^n = u_m^n - 2h(u_x)_m^n + 2h^2(u_{xx})_m^n + O(h^3)$

Rearrange the discretised equation:

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

- $u_m^{n+1} - u_m^{n-1} = 2k(u_t)_m^n + O(k^3)$
- $u_{m+1}^n + u_m^n + u_{m-1}^n = 3u_m^n + h^2(u_{xx})_m^n + O(h^3)$
- $u_{m+1}^n - u_{m-1}^n = 2h(u_x)_m^n + O(h^3)$
- $u_{m+2}^n - u_{m-2}^n = 4h(u_x)_m^n + O(h^3)$ and $-2u_{m+1}^n + 2u_{m-1}^n = -4h(u_x)_m^n + O(h^3)$
- So $u_{m+2}^n - 2u_{m+1}^n + 2u_{m-1}^n - u_{m-2}^n = O(h^3)$

Now substitute into the rearranged discretised equation:

$$2k(u_t)_m^n + O(k^3) = -\frac{k}{3h}\left(3u_m^n + h^2(u_{xx})_m^n + O(h^3)\right)\left(2h(u_x)_m^n + O(h^3)\right) - \frac{k\delta^2}{h^3}\left(O(h^3)\right)$$

$$2k(u_t)_m^n + O(k^3) = -\frac{k}{3h}\left(6hu_m^n(u_x)_m^n + 2h^3(u_x)_m^n(u_{xx})_m^n + O(h^3)\right) - k\delta^2\left(O(1)\right)$$

For simplicity, write $u_m^n = u$, $(u_t)_m^n = u_t$, $(u_x)_m^n = u_x$ and $(u_{xx})_m^n = u_{xx}$, so

$$2ku_t + O(k^3) = -\frac{k}{3h}\left(6huu_x + 2h^3u_xu_{xx} + O(h^3)\right) - k\delta^2\left(O(1)\right)$$

$$2ku_t + O(k^3) = -2kuu_x - \frac{2}{3}kh^2u_xu_{xx} + O(h^2) - Ck\delta^2, C \text{ some constant}$$

Divide through by $2k$:

$$u_t + O(k^2) = -uu_x - \frac{1}{3}h^2u_xu_{xx} + O(h^2) - D, D \text{ some constant}$$

$$u_t + uu_x + \frac{1}{3}h^2u_xu_{xx} + D + O(k^2) + O(h^2) = 0$$

$$u_t + uu_x + D + O(k^2) + O(h^2) = 0$$

Comparing this to the original KdV equation, we see that it differs from it by $O(k^2)$ & $O(h^2)$, hence this scheme is of order 2 in both time and space (t and x).

Now, we wish to find the stability condition for δ ≠ 1. Rescale both x and t as such:

$$\tilde{x} = ax \qquad\qquad \tilde{t} = bt$$

, where a and b are both constants we aim to find.

Firstly, $u(x,t) = u(\tilde{x},\tilde{t})$

Now find the required derivatives:

$$u_t = \frac{\partial u}{\partial t} = \frac{\partial u}{\partial \tilde{t}}\frac{\partial \tilde{t}}{\partial t} = b\frac{\partial u}{\partial \tilde{t}} = bu_{\tilde{t}} \qquad u_x = \frac{\partial u}{\partial x} = \frac{\partial u}{\partial \tilde{x}}\frac{\partial \tilde{x}}{\partial x} = a\frac{\partial u}{\partial \tilde{x}} = au_{\tilde{x}}$$

$$u_{xxx} = a^3 u_{\tilde{x}\tilde{x}\tilde{x}}$$

Substitute these into the original KdV equation:

$$bu_{\tilde{t}} + auu_{\tilde{x}} + a^3\delta^2 u_{\tilde{x}\tilde{x}\tilde{x}} = 0$$

We wish for this to be equivalent to the original KdV equation for δ = 1 and choose values for a and b such that this is the case, meaning that we require

$$b = a = a^3\delta^2$$

Solving $a = a^3\delta^2$, we get that $\delta^2 = \frac{1}{a^2}$ hence $a = b = \frac{1}{\delta}$. Substituting this into the scaled equation,

$$\frac{1}{\delta}u_{\tilde{t}} + \frac{1}{\delta}uu_{\tilde{x}} + \frac{1}{\delta}u_{\tilde{x}\tilde{x}\tilde{x}} = 0$$

Multiplying the equation by δ, we indeed get that

$$u_{\tilde{t}} + uu_{\tilde{x}} + u_{\tilde{x}\tilde{x}\tilde{x}} = 0$$

, which is the KdV equation for δ = 1.

As given in the question, this is stable provided that

$$\tilde{k} \le \frac{\tilde{h}^3}{4 + \tilde{h}^2|u_{max}|}$$

- $\tilde{t} = \frac{t}{\delta}$, so $\tilde{k}n = \frac{kn}{\delta} \Rightarrow \tilde{k} = \frac{k}{\delta}$
- $\tilde{x} = \frac{x}{\delta}$, so $\tilde{h}m = \frac{hm}{\delta} \Rightarrow \tilde{h} = \frac{h}{\delta}$
- u_{max} remains the same, as $u(x,t) = u(\tilde{x},\tilde{t})$

Substitute into the inequality:

$$\frac{k}{\delta} \le \frac{\left(\frac{h}{\delta}\right)^3}{4 + \left(\frac{h}{\delta}\right)^2|u_{max}|} \Rightarrow \frac{k}{\delta} \le \frac{h^3}{4\delta^3 + \delta h^2|u_{max}|}$$

Multiplying everything by δ, we get the equivalent stability condition for δ ≠ 1:

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

To carry out the first step of the iteration scheme, to get from u_m^0 to u_m^1, we need to use a different method as with the leapfrog method, we require u_m^{-1} too, which we don't have. Hence, we will use the forward Euler method, as this only needs u_m^0, which is the initial condition given. Using forward Euler, we have that

$$u_m^1 = u_m^0 + k\frac{\partial u}{\partial t}(0)$$

By the KdV equation, $u_t(x,0) = -u(x,0)u_x(x,0) - \delta^2 u_{xxx}(x,0)$.

- $u(x,0) = Asech^2\left(\frac{x-x_0}{\Delta}\right)$
- $u_x(x,0) = -\frac{2A}{\Delta}sech^2\left(\frac{x-x_0}{\Delta}\right)tanh\left(\frac{x-x_0}{\Delta}\right)$
- $u_{xxx}(x,0) = -\frac{2A}{\Delta^3}\left(-8sech^4\left(\frac{x-x_0}{\Delta}\right)tanh\left(\frac{x-x_0}{\Delta}\right) + 4sech^2\left(\frac{x-x_0}{\Delta}\right)tanh^3\left(\frac{x-x_0}{\Delta}\right)\right) = \frac{16A}{\Delta^3}sech^4\left(\frac{x-x_0}{\Delta}\right)tanh\left(\frac{x-x_0}{\Delta}\right) - \frac{8A}{\Delta^3}sech^2\left(\frac{x-x_0}{\Delta}\right)tanh^3\left(\frac{x-x_0}{\Delta}\right)$

So,

$$u_t(x,0) = \left(Asech^2\left(\frac{x-x_0}{\Delta}\right)\right)\left(\frac{2A}{\Delta}sech^2\left(\frac{x-x_0}{\Delta}\right)tanh\left(\frac{x-x_0}{\Delta}\right)\right) - \delta^2\left(\frac{16A}{\Delta^3}sech^4\left(\frac{x-x_0}{\Delta}\right)tanh\left(\frac{x-x_0}{\Delta}\right) - \frac{8A}{\Delta^3}sech^2\left(\frac{x-x_0}{\Delta}\right)tanh^3\left(\frac{x-x_0}{\Delta}\right)\right)$$

Substituting this into the equation for forward Euler, we get that:

$$u_m^1 = Asech^2\left(\frac{x-x_0}{\Delta}\right) + k\left(\left(Asech^2\left(\frac{x-x_0}{\Delta}\right)\right)\left(\frac{2A}{\Delta}sech^2\left(\frac{x-x_0}{\Delta}\right)tanh\left(\frac{x-x_0}{\Delta}\right)\right) - \delta^2\left(\frac{16A}{\Delta^3}sech^4\left(\frac{x-x_0}{\Delta}\right)tanh\left(\frac{x-x_0}{\Delta}\right) - \frac{8A}{\Delta^3}sech^2\left(\frac{x-x_0}{\Delta}\right)tanh^3\left(\frac{x-x_0}{\Delta}\right)\right)\right)$$

After that, we implement the leap-frog iteration scheme.

The code used to carry out the leapfrog iteration scheme is **Code 1** on page 15, labelled as

```
leapfrog(A,delta,x_0,t_final)
```

To check the accuracy of the program, we will use it to calculate $u(x, 0.5)$ for initial condition

$$u(x,0) = Asech^2\left(\frac{x-x_0}{\Delta}\right)$$

with $\delta = 0.03, A = 2\ and\ x_0 = 0.25$.

The code gives the following graph:

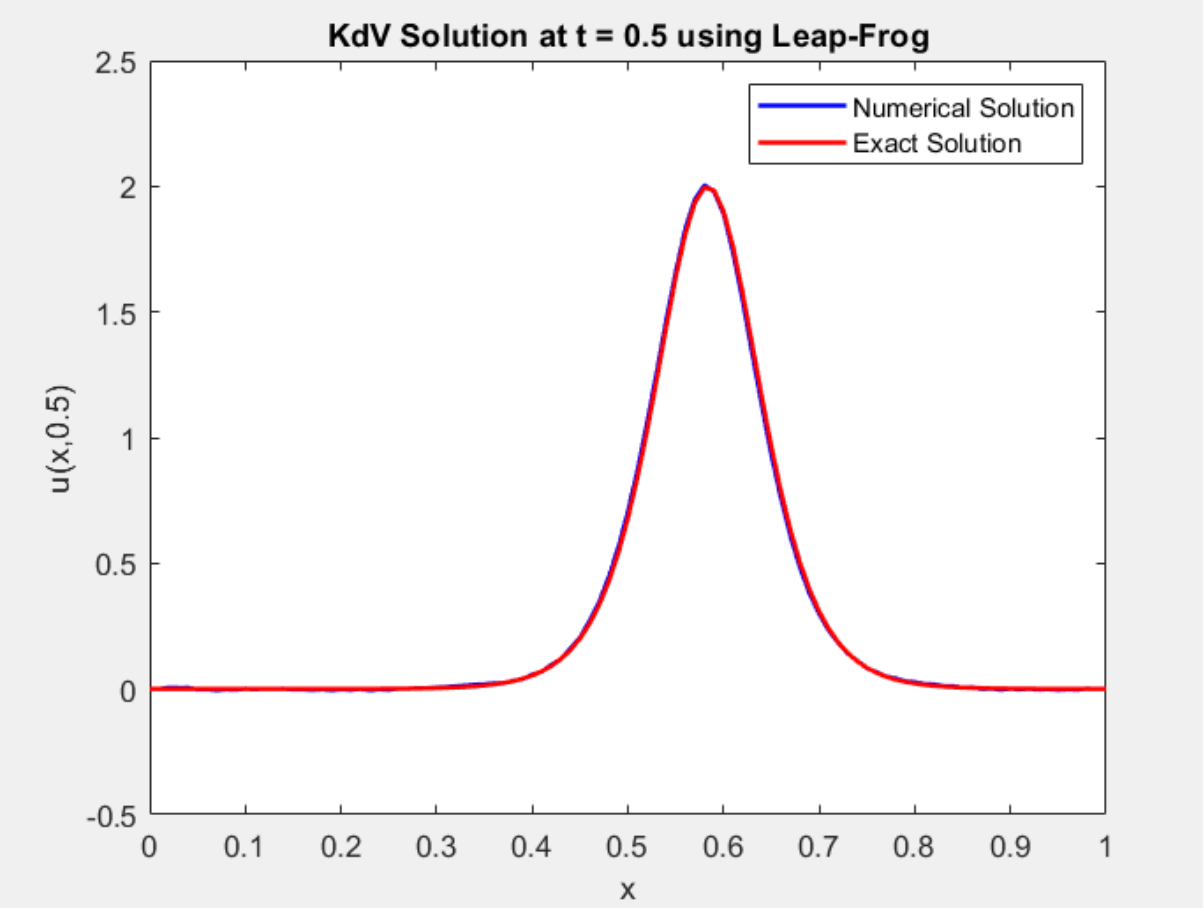

Figure 1: *A plot of the numerical and exact solutions to the KdV equation at* $t = 0.5$*obtained using the leapfrog scheme,* with $\delta = 0.03, A = 2$ *and* $x_0 = 0.25$.

There are inclusively few lines in the program used to plot this graph to also calculate the root mean square error, and the value calculated is **1.02215 x 10^{-2}**, aligning with the scheme being of order 2.

The value of h used, $h = 0.01$, is chosen such that it's small enough to give a good approximation to the solution but inclusively balances with computational efficiency. The value of k is then chosen such that it lies within the stability region,

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

Using this, we can calculate that the maximum value k can take is

$$\frac{(0.01)^3}{4(0.03)^2 + (0.01)^2|2|} = \frac{0.000001}{0.0038} = 2.6316 \times 10^{-4}$$

, taking $u_{max} = 2$. The value of k used in the program, 0.0001, is less than 2.6316×10^{-4}, so lies well within the stability region. Therefore, it is a sensible value of k to use.

The numerical and exact solutions closely match at $t = 0.5$, with minimal amplitude error and the soliton shape preserved. However, zooming closely into the graph, a small phase shift is noticeable – the numerical solution lags ever so slightly behind the exact solution. This tells us that the propagation speed of the numerical solution is very close to that of the exact solution, but a little slower.

QUESTION 3

Next, consider the evolution of the initial data corresponding to the sum of two solitons, one of them with $A = 2$ and $x_0 = 0.25$, and a second with $A = 1$ and $x_0 = 0.75$, again with $\delta = 0.03$.

The program used to carry this out is **Code 2** on page 16, labelled

```
leapfrog_2(t_final)
```

To calculate the mass and energy associated with the numerical soliton at each time step (so for every n loop in the code), use the trapezium rule, most generally written as

$$\int_{x_0}^{x_n} f(x)\,dx = \frac{1}{2}h\left[\left(f(x_0) + f(x_n)\right) + 2\left(f(x_1) + \cdots f(x_{n-1})\right)\right]$$

In the specific context of this problem, the value of h will just be the one used in the code (0.01).

For mass, the various values of $f(x_i)$ for $i = 0, \ldots, n$ correspond to the values of u at each spatial point after each time iteration. For energy, we will define $v = \frac{u^2}{2}$ at each spatial point, and the corresponding $f(x_i)$ values after each time step are given by the updated v.

Based on what we proved in question 1, we expect to see that the values for mass and energy at each time step remain the same, as mass and energy are both independent of time.

The program used for this question outputs the following graphs:

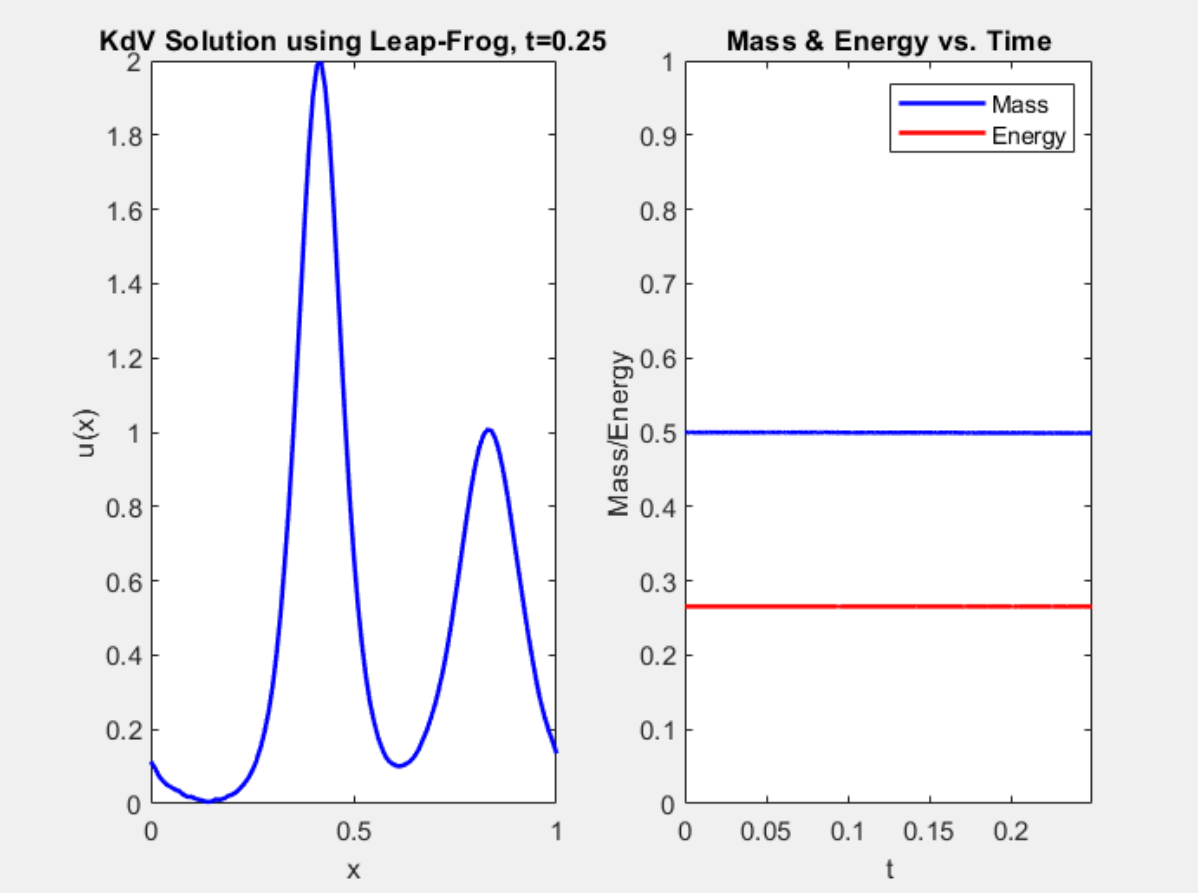

Figure 2: *A plot of the numerical solution to the KdV equation at* $t = 0.25$*obtained using the leapfrog scheme, initialised by the sum of two solitons,* with $\delta = 0.03$*and accompanying evolution of mass and energy.*

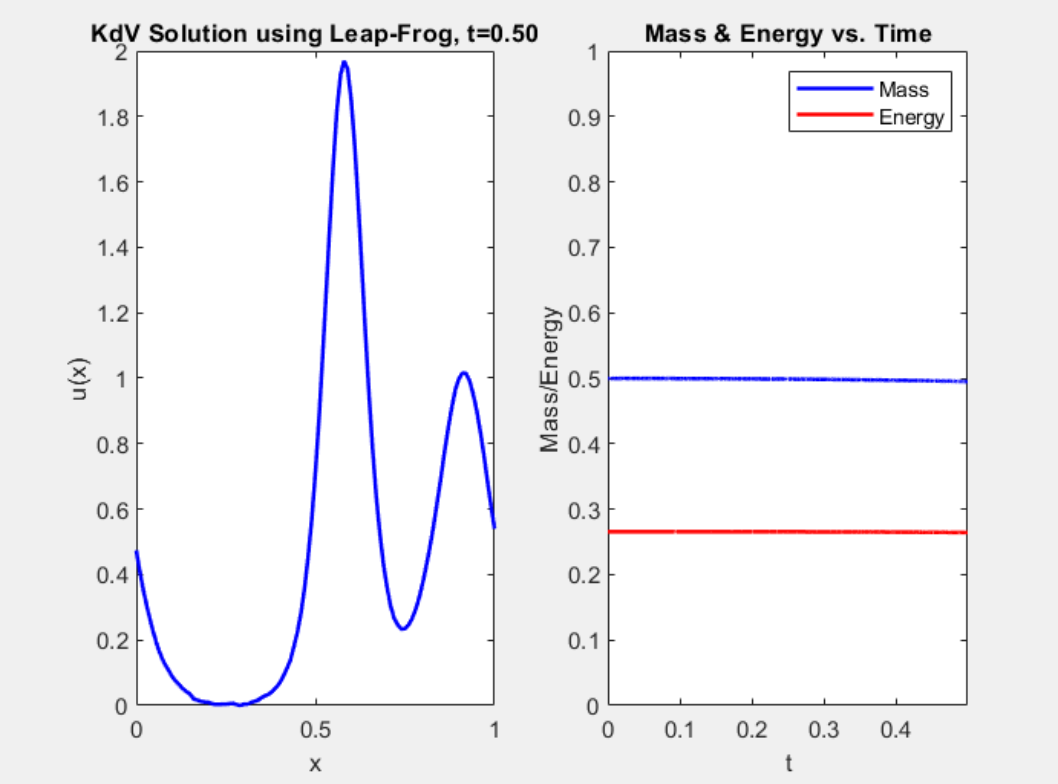

Figure 3: *A plot of the numerical solution to the KdV equation at* $t = 0.5$*obtained using the leapfrog scheme, initialised by the sum of two solitons,* with $\delta = 0.03$*and accompanying evolution of mass and energy.*

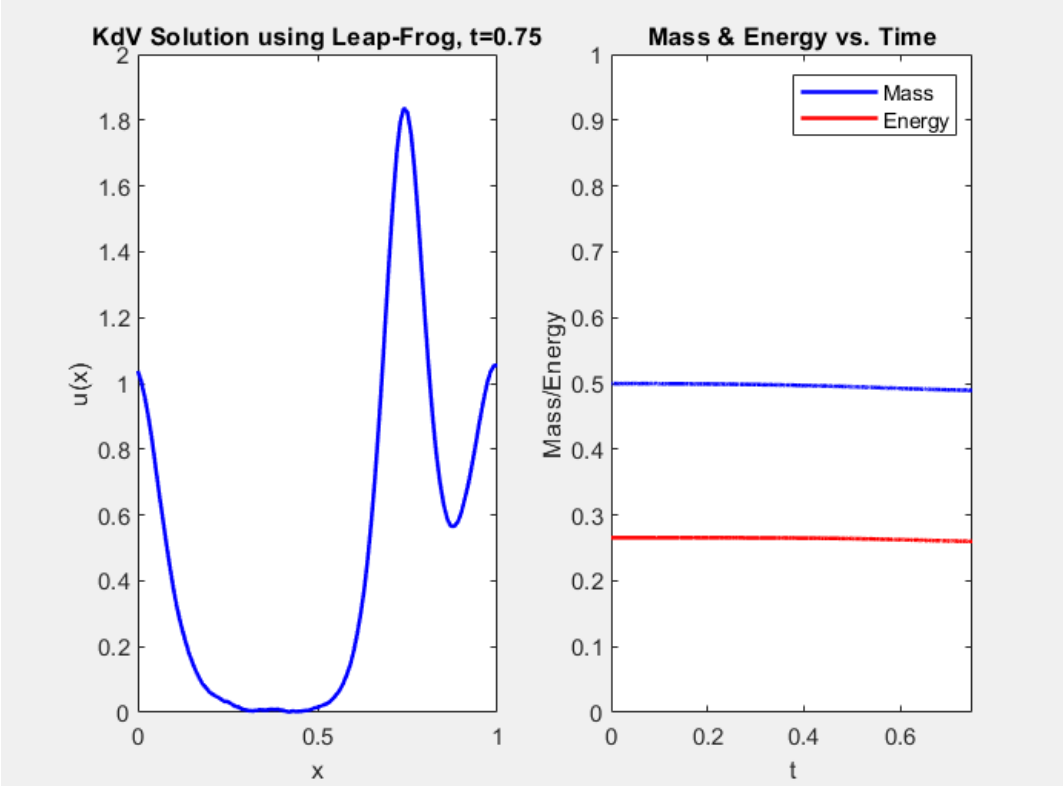

Figure 4: *A plot of the numerical solution to the KdV equation at* $t = 0.75$*obtained using the leapfrog scheme, initialised by the sum of two solitons,* with $\delta = 0.03$*and accompanying evolution of mass and energy.*

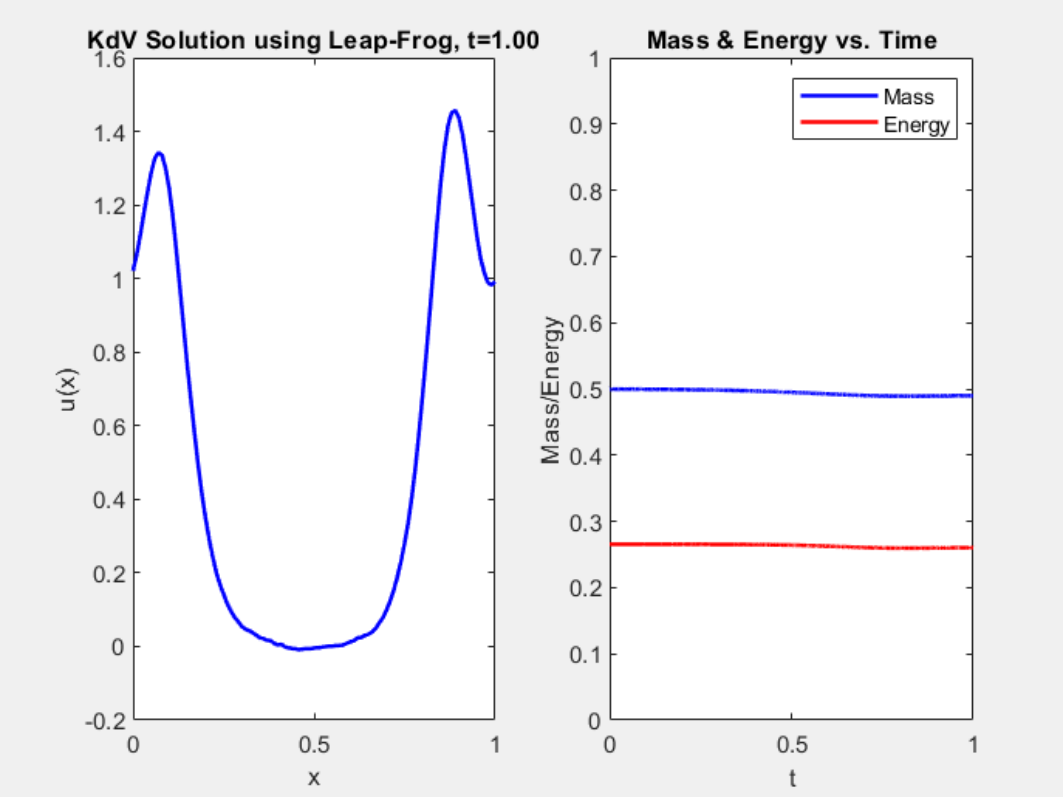

Figure 5: *A plot of the numerical solution to the KdV equation at* $t = 1$ *obtained using the leapfrog scheme, initialised by the sum of two solitons,* with $\delta = 0.03$*and accompanying evolution of mass and energy.*

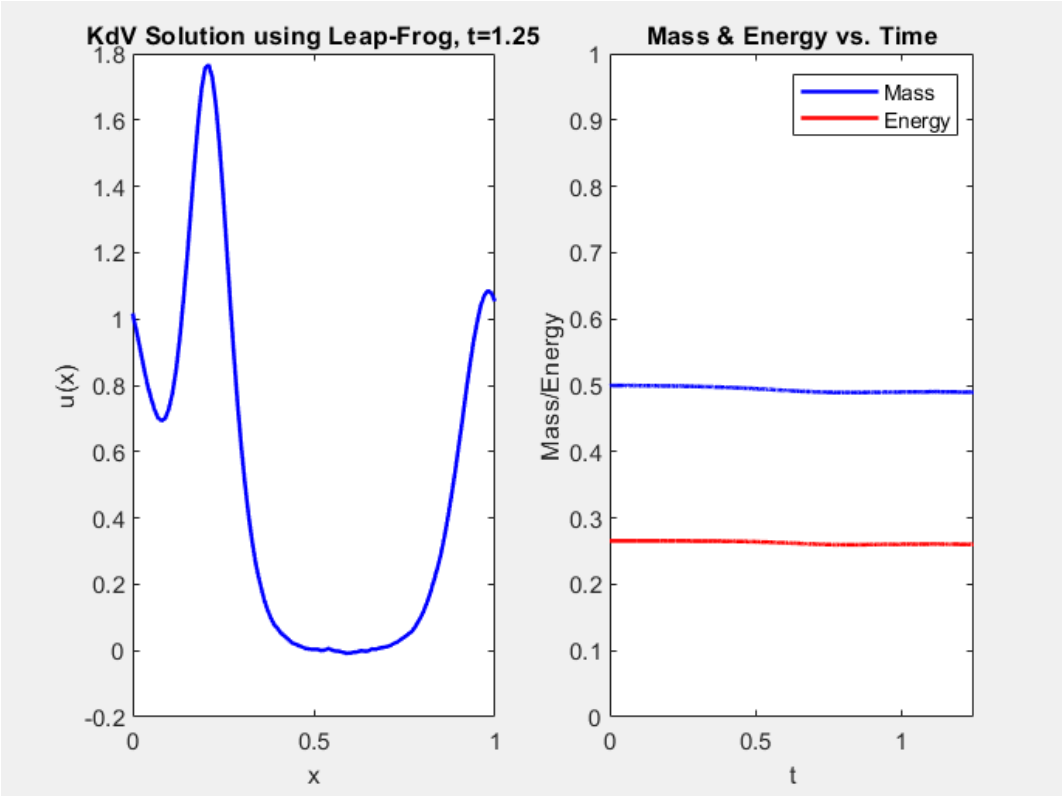

Figure 6: *A plot of the numerical solution to the KdV equation at* $t = 1.25$ *obtained using the leapfrog scheme, initialised by the sum of two solitons,* with $\delta = 0.03$*and accompanying evolution of mass and energy.*

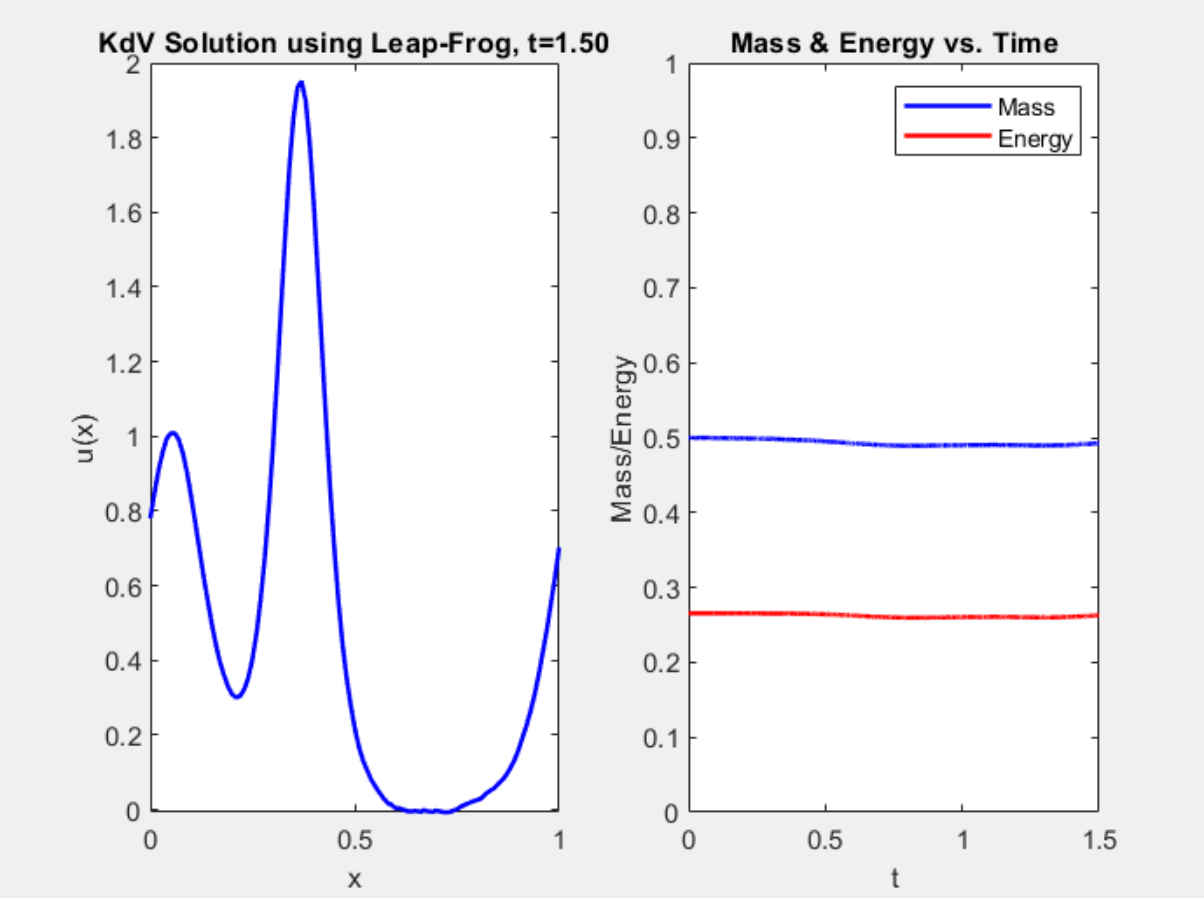

Figure 7: *A plot of the numerical solution to the KdV equation at* $t = 1.5$ *obtained using the leapfrog scheme, initialised by the sum of two solitons,* with $\delta = 0.03$*and accompanying evolution of mass and energy*.

Initially, the taller soliton travels faster than the shorter one, causing the larger wave to catch up. Around $t = 0.75$, the larger soliton overtakes the smaller one, leading to a brief distortion of their shapes. After the interaction, at around $t = 1.25$ to $t = 1.5$, both solitons re-emerge with nearly the same shape and speed as before, hence confirming that the leapfrog method captures the essential soliton dynamics.

As we can see from these graphs, the values of mass and energy associated with each time step seem to remain relatively constant, with approximately values 0.50 and 0.26 respectively, as expected!

QUESTION 4

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

When $\delta = 0$, the equation simplifies to involve only the nonlinear term uu_x. In this case, parts of the wave with higher values of u move faster than parts with lower u. As a result, the wave starts to steepen. Over time, this steepening becomes more extreme, and eventually the wave forms a very sharp front (like a shock). Without dispersion, controlled by the $\delta^2 u_{xxx}$ term, there is nothing to smooth the wave out.

Now, investigate the $\delta = 0.03$ case numerically.

The code for this question will be very similar to the code used in **Question 2**, but we will have to alter the initial condition, and inclusively recalculate the information required to carry out the first step using forward Euler. Recall that

$$u_m^1 = u_m^0 + k\frac{\partial u}{\partial t}(0)$$

By the KdV equation, $u_t(x,0) = -u(x,0)u_x(x,0) - \delta^2 u_{xxx}(x,0)$.

- $u(x,0) = sin(2\pi x)$
- $u_x(x,0) = 2\pi cos\,(2\pi x)$
- $u_{xxx}(x,0) = -8\pi^3 cos\,(2\pi x)$

So,

$$u_t(x,0) = -\big(sin(2\pi x)\big)(2\pi\, cos(2\pi x)) - \delta^2(-8\pi^3\, cos(2\pi x))$$
$$= -2\pi\, sin(2\pi x)cos(2\pi x) + 8\pi^3\,\delta^2 cos(2\pi x) = 8\pi^3\,\delta^2 cos(2\pi x) - \pi sin(4\pi x)$$

Substituting this into the equation for forward Euler, we get that:

$$u_m^1 = sin(2\pi x) + k\big(8\pi^3\delta^2\, cos(2\pi x) - \pi sin(4\pi x)\big)$$

After that, we implement the leap-frog iteration scheme as we did before.

The code used to carry out the leapfrog iteration scheme is **Code 3** on page 18, labelled as

```
leapfrog_3(delta,t_final,k)
```

We use the same time and spatial steps: $h = 10^{-2}, k = 10^{-4}$.

We will see how this initial data evolves over time for $\delta = 0.03$ at different instants; the solutions for $t = 0.1, 0.25, 0.5, 0.75, 1, 2.5$ are plotted below.

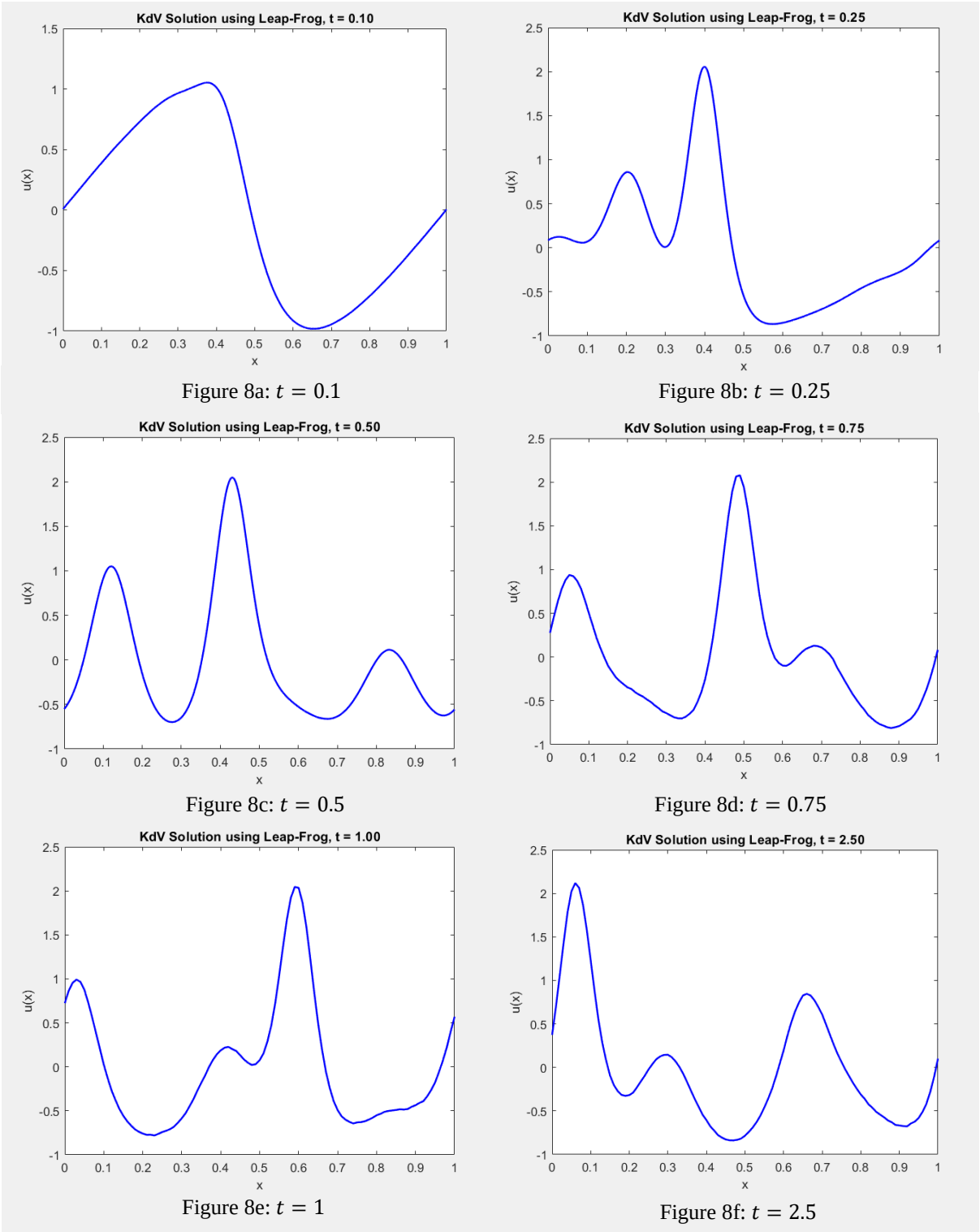

Figure 8a: $t = 0.1$

Figure 8b: $t = 0.25$

Figure 8c: $t = 0.5$

Figure 8d: $t = 0.75$

Figure 8e: $t = 1$

Figure 8f: $t = 2.5$

Figure 8: *A plot of the numerical solutions to the KdV equation obtained using the leapfrog scheme, with initial data* $u(x, 0) = sin(2\pi x) \& \delta = 0.03$.

Recall the KdV equation:

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

Initially, $\delta^2 u_{xxx}$ is really weak compared to uu_x, causing wave steepening – we see shock-like shapes. However, as time passes, the dispersive term becomes more significant; the dispersion spreads the waves and prevents discontinuities, causing the change in shape to a series of different solitons. In the $\delta = 0.03$ case, this change occurs at around $t \approx 0.16$. After this time, the solution transitions from steepening to a train of dispersive waves or solitons.

δ controls the strength of dispersion in the KdV equation, meaning that for smaller δ, we see the solution change character at a later time, and much more dramatically, as there'll be much more rapid steepening at first. In contrast, for larger δ, the dispersion term is stronger and so the solution changes character earlier, with a much smoother transition.

Programs

CODE 1

```
function leapfrog(A,delta,x_0,t_final)
h = 0.01;
k = 0.0001;
Delta = sqrt(12*(delta)^2/A);
x = 0:h:1;
sech = @(z) 1 ./ cosh(z);
u_0 = A*(sech((x-x_0)/Delta)).^2;

%for the first step, use the forward Euler method
u_x0 = -((2*A)/Delta)*((sech((x-x_0)/Delta)).^2).*tanh((x-x_0)/Delta);
u_xxx0_1 = ((16*A)/(Delta)^3)*((sech((x-x_0)/Delta)).^4).*tanh((x-x_0)/Delta);
u_xxx0_2 = ((8*A)/(Delta)^3)*((sech((x-x_0)/Delta)).^2).*(tanh((x-x_0)/Delta)).^3;
u_xxx0 = u_xxx0_1 - u_xxx0_2;

u = u_0 - k*(u_0.*u_x0 + ((delta)^2)*u_xxx0);

u_prev = u_0;
u_new = zeros(size(x));

umax = max(abs(u_0));
k_max = h^3 / (4*(delta)^2 + h^2 * umax);
if k > k_max
    fprintf("Your time step k is too big – instability likely!");
end

N = length(x);

for n = 2:round(t_final/k)
    for m = 1:N
        mp2 = mod(m+1, N) + 1;
        mp1 = mod(m, N) + 1;
        mm1 = mod(m-2, N) + 1;
        mm2 = mod(m-3, N) + 1;

        u_new(m) = u_prev(m) - (k / (3*h)) * (u(mp1) + u(m) + u(mm1)) * (u(mp1) -
u(mm1)) - ((k * delta^2) / h^3) * (u(mp2) - 2*u(mp1) + 2*u(mm1) - u(mm2));
    end
    u_prev = u;
    u = u_new;
end

c = A/3;
u_exact = A*(sech((x-x_0-c*t_final)/Delta).^2);

plot(x, u_new,'b-','LineWidth', 1.5); hold on;
plot( x, u_exact,'r-','LineWidth', 1.5)
legend('Numerical Solution', 'Exact Solution')
xlabel('x'), ylabel('u(x,0.5)')
title('KdV Solution at t = 0.5 using Leap-Frog')
```

```
%finding the root mean square error
error_rmse = sqrt(sum((u_new - u_exact).^2) * h);
fprintf("the estimated (root mean square) error is %.5e\n", error_rmse)

end
```

CODE 2

```
function leapfrog_2(t_final)
h = 0.01;
k = 0.0001;
delta = 0.03;
x = 0:h:1;

A1 = 2;
x0_1 = 0.25;
Delta_1 = sqrt(12*(delta)^2/A1);

A2 = 1;
x0_2 = 0.75;
Delta_2 = sqrt(12*(delta)^2/A2);

sech = @(z) 1 ./ cosh(z);
u0_1 = A1*(sech((x-x0_1)/Delta_1)).^2;
u0_2 = A2*(sech((x-x0_2)/Delta_2)).^2;

%combined initial condition
u_0 = u0_1 + u0_2;

%initialise a vector for mass and energy for the masses and energies to be
%calculated at each time step (indexed based on the value of n)
masses = zeros(1, round(t_final/k));
energies = zeros(1, round(t_final/k));
timesteps = 0.0001:0.0001:t_final;

%for the first step, use the forward Euler method
%have to do separately for first soliton and second soliton and then add

u_x0_1 = -((2*A1)/Delta_1)*((sech((x-x0_1)/Delta_1)).^2).*tanh((x-x0_1)/Delta_1);
u_x0_2 = -((2*A2)/Delta_2)*((sech((x-x0_2)/Delta_2)).^2).*tanh((x-x0_2)/Delta_2);
u_x0 = u_x0_1 + u_x0_2;

u_xxx0_1_1 = ((16*A1)/(Delta_1)^3)*((sech((x-x0_1)/Delta_1)).^4).*tanh((x-
x0_1)/Delta_1);
u_xxx0_2_1 = ((8*A1)/(Delta_1)^3)*((sech((x-x0_1)/Delta_1)).^2).*(tanh((x-
x0_1)/Delta_1)).^3;
u_xxx0_1 = u_xxx0_1_1 - u_xxx0_2_1;

u_xxx0_1_2 = ((16*A2)/(Delta_2)^3)*((sech((x-x0_2)/Delta_2)).^4).*tanh((x-
x0_2)/Delta_2);
u_xxx0_2_2 = ((8*A2)/(Delta_2)^3)*((sech((x-x0_2)/Delta_2)).^2).*(tanh((x-
x0_2)/Delta_2)).^3;
u_xxx0_2 = u_xxx0_1_2 - u_xxx0_2_2;
```

```
u_xxx0 = u_xxx0_1 + u_xxx0_2;

u = u_0 - k*(u_0.*u_x0 + ((delta)^2)*u_xxx0);

%need to find the mass and energy for this first time step: index 1
mass_sum_1 = 0;
for i = 2:length(x)-1
    mass_sum_1 = mass_sum_1 + u(i);
end
masses(1) = (h/2)*((u(1)+u(length(x))) + 2*mass_sum_1);

v = 0.5*(u).^2;

energy_sum_1 = 0;
for j = 2:length(x)-1
    energy_sum_1 = energy_sum_1 + v(j);
end
energies(1) = (h/2)*((v(1)+v(length(x))) + 2*energy_sum_1);

u_prev = u_0;
u_new = zeros(size(x));

umax = max(abs(u_0));
k_max = h^3 / (4*(delta)^2 + h^2 * umax);
if k > k_max
    fprintf("Your time step k is too big – instability likely!");
end

N = length(x);

for n = 2:round(t_final/k)
    for m = 1:N
        mp2 = mod(m+1, N) + 1;
        mp1 = mod(m, N) + 1;
        mm1 = mod(m-2, N) + 1;
        mm2 = mod(m-3, N) + 1;

        u_new(m) = u_prev(m) - (k / (3*h)) * (u(mp1) + u(m) + u(mm1)) * (u(mp1) -
u(mm1)) - ((k * delta^2) / h^3) * (u(mp2) - 2*u(mp1) + 2*u(mm1) - u(mm2));

    end

    u_prev = u;
    u = u_new;

    mass_sum = 0;
    for i = 2:length(x)-1
        mass_sum = mass_sum + u(i);
    end

    masses(n) = (h/2)*((u(1)+u(length(x))) + 2*mass_sum);

    v_new = 0.5*(u).^2;

    energy_sum = 0;
```

```
    for j = 2:length(x)-1
        energy_sum = energy_sum + v_new(j);
    end

    energies(n) = (h/2)*((v_new(1)+v_new(length(x))) + 2*energy_sum);
end

subplot(1,2,1)
plot(x, u_new, 'b-','LineWidth', 1.5);
xlabel('x'), ylabel('u(x)')
title('KdV Solution using Leap-Frog')

subplot(1,2,2)
plot(timesteps, masses, 'b-', 'LineWidth',1.5); hold on;
plot(timesteps, energies, 'r-', 'LineWidth',1.5);
legend('Mass', 'Energy')
xlabel('t'); ylabel('Mass/Energy');
xlim([0 t_final])
ylim([0 1])
title('Mass & Energy vs. Time');

end
```

CODE 3

```
function leapfrog_3(delta,t_final,k)
h = 0.01;
x = 0:h:1;
u_0 = sin(2*pi*x);

%for the first step, use the forward Euler method
u_x0 = 2*pi*cos(2*pi*x);
u_xxx0 = -8*((pi)^3)*cos(2*pi*x);

u = u_0 - k*(u_0.*u_x0 + ((delta)^2)*u_xxx0);

u_prev = u_0;
u_new = zeros(size(x));

umax = max(abs(u_0));
k_max = h^3 / (4*(delta)^2 + h^2 * umax);
if k > k_max
    fprintf("Your time step k is too big – instability likely!");
end

N = length(x);

for n = 2:round(t_final/k)
    for m = 1:N
        mp2 = mod(m+1, N) + 1;
        mp1 = mod(m, N) + 1;
        mm1 = mod(m-2, N) + 1;
        mm2 = mod(m-3, N) + 1;
```

```
        u_new(m) = u_prev(m) - (k / (3*h)) * (u(mp1) + u(m) + u(mm1)) * (u(mp1) -
u(mm1)) - ((k * delta^2) / h^3) * (u(mp2) - 2*u(mp1) + 2*u(mm1) - u(mm2));
    end
    u_prev = u;
    u = u_new;
end

plot(x, u_new, 'b-','LineWidth', 1.5)
xlabel('x'), ylabel('u(x)')
title(sprintf('KdV Solution using Leap-Frog, t = %.2f', t_final))

end
```