TMA4212 project: Exploring a finite difference scheme for the Camassa-Holm equation

Candidate numbers 10031, 10045, 10052

April 3, 2014

Abstract

This paper presents a finite difference scheme for solving the Camassa-Holm equation with periodic boundary conditions. After the scheme has been presented, it is analyzed in terms of stability and consistency. Finally, we look at the numerical results obtained from implementing the scheme in MatLab, and how to reduce the computation time of the implementation.

Introduction

The Camassa-Holm equation was studied first by Camassa and Holm [1] and is defined as

$$u_t - u_{xxt} + 2\kappa u_x + 3uu_x - 2u_x u_{xx} - uu_{xxx} = 0,$$

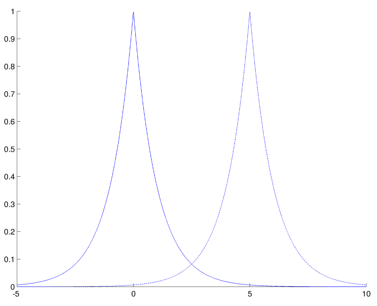
or (if $\kappa = 0$) equivalently in the Hamiltonian form

$$\begin{aligned} m &= u - u_{xx} \\ m_t &= -2mu_x - um_x. \end{aligned} \tag{1}$$

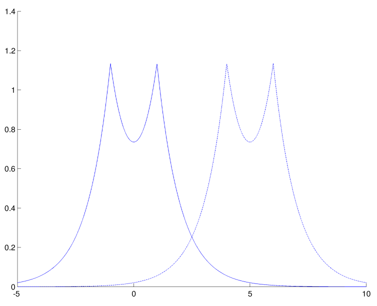
The equation is used to model waves in shallow water. The typical solutions of this equation are solitary waves of permanent shape, traveling at constant speed. If these waves can interact with other waves of the same type without changing their shape or speed, they are called solitons.

In this paper, the only cases considered are when $\kappa = 0$. Removing this term makes the soliton solutions peaked. These solutions are called peakons and have a discontinuity in the first derivative in the peak. Peakons also have a constant traveling speed equal to its height.

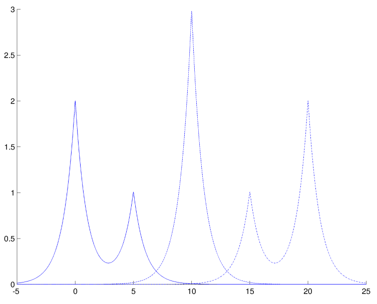
A single peakon is presented in Figure 1a and we see that the peakon retains its shape over time. A linear combination of peakons is called a multipeakon, one can be seen in Figure 1b. Peakons can also interact with other peakons of different size, and in Figure 1c we see that both of the peakons retain their original form after the collision is over. Finally, we see in Figure 1d a peakon and an antipeakon collide. An antipeakon is a negative peakon. What is shown here is the solution that conserves the energy, where the peakons re-emerge after the collision. Another solution, called a non-conservative solution, would involve the two peakons completely canceling each other out.



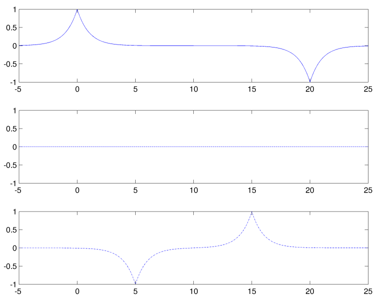
(a) A peakon, t = 0, 5.



(b) A doublepeakon, t = 0, 5.



(c) Two peakons collide, t = 0, 5, 10



(d) Peakon - antipeakon, t = 0, 10, 15

Figure 1: Peakons at different times.

The finite difference scheme

The finite difference scheme considered was the one used by Holden and Raynaud [3]:

$$\begin{aligned} m &= u - D_- D_+ u, \\ m_t &= -D_-(mu) - mD(u), \end{aligned} \tag{2}$$

where the operators are defined as

$$D_+ u_j^n = \frac{u_j^n - u_{j-1}^n}{h}, \quad D_- u_j^n = \frac{u_{j+1}^n - u_j^n}{h}, \quad D = \frac{D_+ + D_-}{2}. \tag{3}$$

The main disadvantage of this scheme is that one has to assume only positive values of u. However, it is debatable whether a negative u makes sense or not.

Our finite difference scheme is based on a scheme presented by Morten Lien Dahlby [2], and is a slight modification of the scheme presented by Holden and Raynaud.

$$\begin{aligned} m &= u - D_- D_+ u, \\ m_t &= -D_-(m(u \vee 0)) - D_+(m(u \wedge 0)) - mD(u), \end{aligned} \tag{4}$$

where $(u \vee 0) = \max(u, 0)$ and $(u \wedge 0) = \min(u, 0)$.

The original scheme by Holden and Raynaud used the D_--operator, making it an upwind method. This would not work for antipeakons. Modifying the scheme to use D_- when $u > 0$ and D_+ when $u < 0$ makes it applicable to waves traveling in either direction. This modification also makes the scheme able to handle more initial conditions and genereally makes the method more stable.

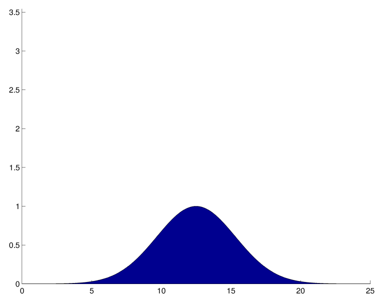
To find an appropriate time step, the Courant–Friedrichs–Lewy (CFL) condition was considered. The CFL condition is a necessary condition for stability in our finite difference scheme, and can be stated as

$$c\frac{\Delta t}{\Delta x} \leq 1,$$

where c is the velocity of the wave, Δt is the length of the time step, and Δx is the length of the spacial step. Rearranging this inequality gives an expression for Δt:

$$\Delta t \leq \frac{\Delta x}{c}$$

An estimate for c is found in the following way: Since we do not know how fast a wave formed by the initial conditions can move, we use a simple heuristic. We transform all the area in the initial condition into a peakon (which always has the area $2c$) and thereby bound the the speed equal to the height of this peakon. A simple illustration can be found in Figures 2a and 2b. The same heuristic can also be used for antipeakons.


(a) Initial condition

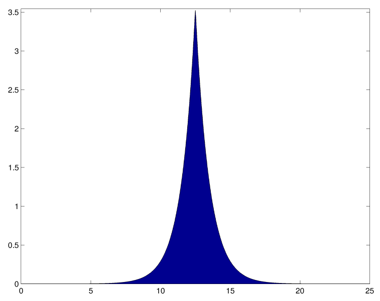

(b) Peakon of same area

Figure 2: Initial condition and peakon of same area, used to estimate c.

Correctness

A preliminary conclusion based on the plot of errors for decreasing spacial stepsize h (Figure 3) shows that decreasing h will improve the approximation. Figure 4 shows that as the spacial stepsize h decreases, our scheme becomes a good approximation of the solution.

The Camassa-Holm equation has one critical property regarding convergence. Not only are the solutions discontinuous in the first derivative, the height is also equal to the speed. This means that any small error produced by a numerical method will amplify quickly. As the height of the peakon becomes lower it also becomes slower and thus quickly loses track of the analytical solution (at least for small grid sizes). If we make the endtime T small enough, we are able to show linear convergence in space as shown in figure 5, with a constant time step. On the other hand, we are not able to show convergence in time (with a constant spatial step). See the loglog plot in figure 6. It does not seem to converge to the peakon solution, but rather to a smooth wave with a bounded error. We have not managed to figure out why this happens. More about this phenomenon can be read in the section about semi-discretization. The fact that the solution gets worse with a small time step makes it important to pick the right time step. The bound on the time step presented earlier (from the CFL condition) works well.

Semi-discretization

Writing

$$m_t = -D_-(mu) - mDu = f(m),$$

one may obtain a system of ODEs which can be solved with standard numerical software, though each evaluation of f requires a transformation from

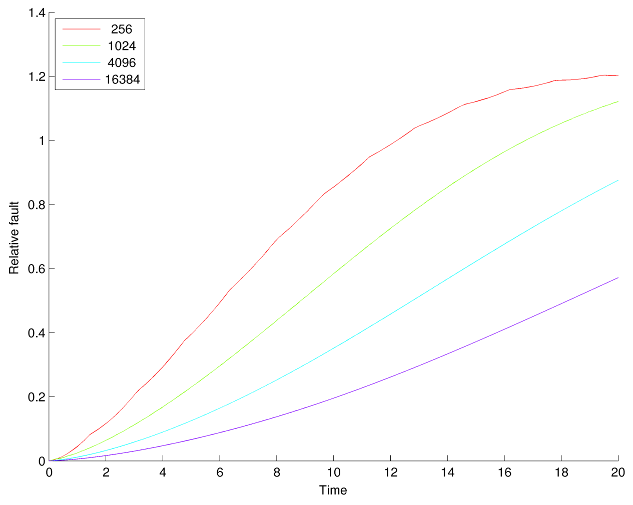


Figure 3: Error plots for decreasing h, constant time and space interval. Initial condition: peakon centered in $x = 0$. End: analytical peakon lies at $x = 20$.

m to u. Additionally, in the end one needs to transform the resulting matrix of m values back to u. Testing a variety of MATLAB's built-in ODE solvers, including *ode45*, *ode15s* and *ode113*, we surprisingly discovered that our explicit scheme based on Euler's method was far superior in every aspect to any other integrator available to us.

The exact reason why is still unknown to us, though we suspect the discontinuous derivatives of the peakon solutions may cause problems for higher order schemes. This is at least consistent with the fact that the higher order schemes appear to smooth out the peaks of the solution.

Another interesting effect was observed when the time step of the explicit scheme was decreased (below the bounded value given by our CFL condition) with the spatial step fixed. Numerical experiments suggest that the solution offered by the explicit scheme converges towards the solution of the higher order time integrators as the time step goes towards zero. In other words, the increased error that results from too small time steps is somehow bounded by the higher-order schemes. We have not yet been able to discover the cause, and our only hypothesis is at best a farfetched and unlikely guess: perhaps our scheme somehow possesses innate damping, which is inaccurately underestimated by Euler's method in such a way that the explicit scheme experiences less damping than the higher-order integrators?

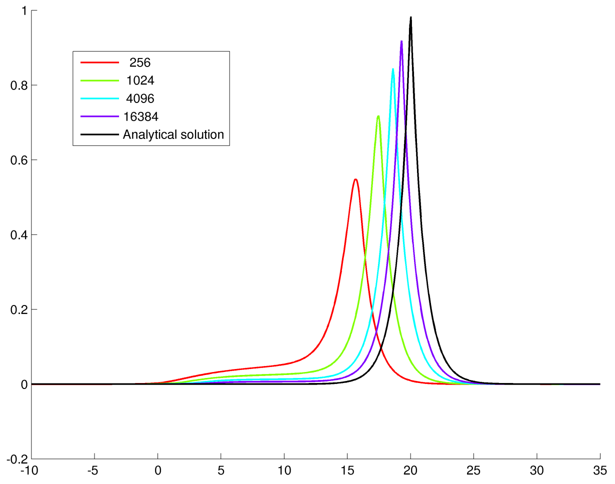


Figure 4: Plot of approximated solution together with analytical solution for decreasing spacial step h. Initial condition: peakon centered in $x = 0$.

Analysis

Invertibility of A

Since the scheme depends on transformation back and forth between m and u, it is crucial that the linear operator A is invertible. Its invertibility can be shown in a few simple steps. First, note that A can be written in circular matrix form

$$\boldsymbol{A} = \begin{pmatrix} (1+\frac{2}{h^2}) & -\frac{1}{h^2} & & -\frac{1}{h^2} \\ -\frac{1}{h^2} & \ddots & \ddots & \\ & \ddots & \ddots & -\frac{1}{h^2} \\ -\frac{1}{h^2} & & -\frac{1}{h^2} & (1+\frac{2}{h^2}) \end{pmatrix}.$$

Then it is fairly easy to see that A is diagonally dominant, since

$$|1+\frac{2}{h^2}| = 1+\frac{2}{h^2} > |-\frac{1}{h^2}| + |-\frac{1}{h^2}| = \frac{2}{h^2}.$$

Since a diagonally dominant matrix is non-singular, A is invertible.

Convergence of Holden Raynaud

Helge Holden and Xavier Raynaud proved the convergence of their finite difference scheme [3], under the assumption that $u_0 - u_0'' \geq 0$ for initial data $u_0 \in H^1([0,1])$. This proof is far beyond the scope of our course, so only an

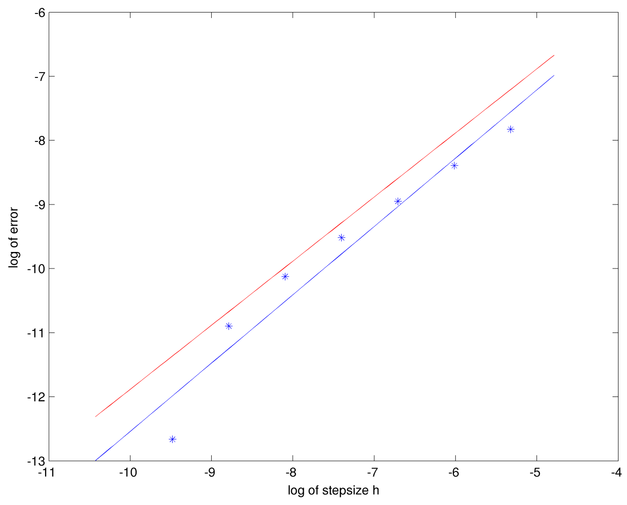


Figure 5: Loglog plot - rate of convergence in space. In this plot, the red reference line has a slope of 1, while the blue line found through numerical experiments has a slope of 1.07. $T = 0.01$, $t = 6.9 \cdot 10^{-5}$,$N = 2^{12}$ to $N = 2^{18}$.

outline of the proof will be provided.

First, it is shown that if $m_i(0) \geq 0 \ \forall \ i$, then any solution $u(t)$ will have $m_i(t) \geq 0 \ \forall \ i$. Further, a uniform bound on the H^1 norm of the sequence u^n in [0,T] is established. This also guarantees the existence of solutions in [0,T] for any given $T > 0$, since $\max_i |u_i^n(t)| = \|u^n(\cdot, t)\|_{L^\infty} \leq \mathcal{O}(1)\|u^n\|_{H^1}$ remains bounded.

After this, it is shown that u_x^n and u_t^n are bounded in $L^2([0,1])$, and that u_x^n has a uniformly bounded total variation. This is enough to prove that one can extract a converging subsequence of u^n, and that the limit is a solution of the Camassa-Holm equation.

This however, is far beyond the scope of this project, so we begin by proving consistency of our method.

Consistency

We only prove consistency for the original scheme in equation (2), but the proof should be similar for the modified scheme as well. With the difference operators as defined in equation (3) we get that:

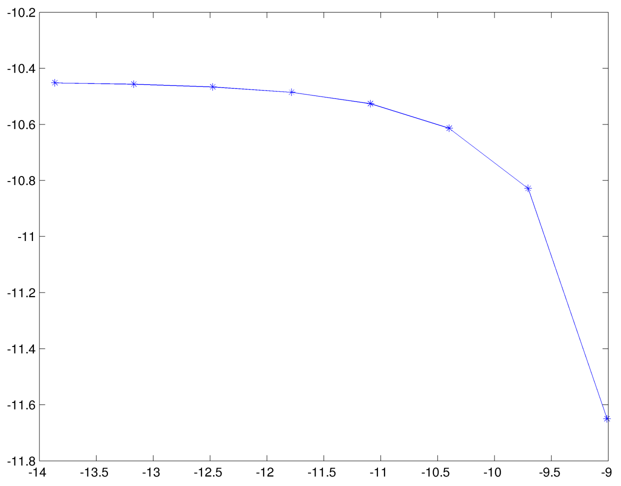


Figure 6: Loglog plot - rate of convergence in time. The method does not converge in time. $T = 0.01$, $t = 2^{-13}$ to $t = 2^{-20}$ and $N = 2^{17}$.

$$
\begin{aligned}
D_-D_+\left(U_j^n\right) &= \frac{\partial^2}{\partial x^2}(u) + \mathcal{O}\left(h^2\right),\\
D_-\left(M_j^n U_j^n\right) &= m\left(\frac{\partial}{\partial x}(u) + \mathcal{O}(h)\right) + u\left(\frac{\partial}{\partial x}(m) + \mathcal{O}\left(h\right)\right),\\
D\left(U_j^n\right) &= \frac{\partial}{\partial x}(u) + \mathcal{O}\left(h^2\right).
\end{aligned}
$$

We use forward Euler to integrate:

$$D_+(m) = \frac{\partial}{\partial t}(m) + \mathcal{O}(k).$$

Putting this back into equation (2)

$$
\begin{aligned}
m &= u - \left(u_{xx} + \mathcal{O}\left(h^2\right)\right)\\
m_t + \mathcal{O}(k) &= -\left(m\left(u_x + \mathcal{O}(h)\right) + u\left(m_x + \mathcal{O}(h)\right)\right) - m\left(u_x + \mathcal{O}(h)\right).
\end{aligned}
$$

Letting the time step and space step go to zero, we have

$$
\begin{aligned}
m &= u - u_{xx}\\
m_t &= -2mu_x - um_x,
\end{aligned}
$$

which is equal to equation (1), so our scheme is consistent.

Stability

The finite difference scheme can be summarized in matrix form in the following way: Define $\boldsymbol{B}$ as the backwards difference matrix operator, $\boldsymbol{F}$ as the forwards difference operator and $\boldsymbol{C}$ as the mean of $\boldsymbol{B}$ and $\boldsymbol{F}$. Furthermore, $\boldsymbol{A}$ is defined as the transformation from $\boldsymbol{u}$ to $\boldsymbol{m}$, ie:

$$\boldsymbol{m} = \boldsymbol{u} - D_- D_+ \boldsymbol{u} = (\boldsymbol{I} - \boldsymbol{B}\boldsymbol{F})\boldsymbol{u} = \boldsymbol{A}\boldsymbol{u},$$

where the difference operators are defined as

$$\boldsymbol{B} = \frac{1}{h}\begin{pmatrix} 1 & & & -1 \\ -1 & \ddots & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix},$$

and

$$\boldsymbol{F} = -\boldsymbol{B}^T.$$

We can rewrite the system in equation (2) as:

$$\begin{aligned} \boldsymbol{m} &= \boldsymbol{A}\boldsymbol{u} \\ \boldsymbol{m}_t &= -\boldsymbol{B}\boldsymbol{M}\boldsymbol{u} - \boldsymbol{M}\boldsymbol{C}\boldsymbol{u} = \boldsymbol{-Q}\boldsymbol{u}, \end{aligned}$$

with $\boldsymbol{M}$ a diagonal matrix with elements from $\boldsymbol{m}$. When we also do the integration and the inverse $\boldsymbol{A}$ we get

$$\begin{aligned} \boldsymbol{m}^{n+1} &= \boldsymbol{m} + k\boldsymbol{m}_t \\ \boldsymbol{m}^{n+1} &= \boldsymbol{m} - k\boldsymbol{Q}\boldsymbol{u} \\ \boldsymbol{u}^{n+1} &= \boldsymbol{A^{-1}}\boldsymbol{m^{n+1}} = \left(\boldsymbol{I} - \boldsymbol{A^{-1}}k\boldsymbol{Q}\right)\boldsymbol{u} = \boldsymbol{Z}\boldsymbol{u}. \end{aligned}$$

To investigate stability we may bound either the spectral radius or the matrix norm of $\boldsymbol{Z}$ by 1. However, this proved to be quite difficult and well beyond our scope, mainly because the $\boldsymbol{Q}$ matrix is dependent upon $\boldsymbol{m}$ (and therefore $\boldsymbol{u}$). Furthermore, the inverse of the tridiagonal (with corners) matrix $\boldsymbol{A}$ does not have an simple analytical solution. Whether this bound exists and behaves well is still a question open to more research.

Von Neumann's stability criterion

An attempt at von Neumann's stability analysis was made, despite the fact that the procedure is primarily for finite difference schemes applied to linear partial differential equations. We define $U_j^n = g^n e^{i\theta j}$. For von Neumann stability, the desired inequality is

$$|g| \leq 1 + \mu k, \tag{5}$$

where $\mu \geq 0$ is a constant. The inequality produced by von Neumann stability analysis was

$$g \leq 1 + \frac{k}{h} g^n e^{i\theta j} \left[\frac{2 - 2e^{-2i\theta} + e^{i\theta} - e^{-i\theta}}{2} \right], \tag{6}$$

which is clearly dependent on h. This makes von Neumann stability analysis inapplicable for our finite difference scheme.

Implementation

Transforming m to u

For each time step when solving the scheme shown in equation (4), one essentially transforms u to m by the transformation matrix $\boldsymbol{A}$, such that $\boldsymbol{m} = \boldsymbol{Au}$. Transforming m back to u usually involves solving the linear system $\boldsymbol{Au} = \boldsymbol{m}$.

Holden and Raynaud [3] present a more efficient way of transforming m to u. Denoting the linear operator from u to m as L, u can be written as a discrete convolution product of m and g:

$$u_i = L^{-1} m_i = \sum_{j \in \mathbb{Z}} g_{i-j} m_j,$$

where g is periodized such that

$$g_i^p = c \frac{e^{-\kappa i} + e^{\kappa(i-n)}}{1 - e^{-\kappa n}},$$

where $i = 0, ..., n-1$, and g is defined by the constants

$$\kappa = \ln \left(1 + \frac{h(\sqrt{4 + h^2} + h)}{2} \right) \qquad c = \frac{1}{1 + \frac{2}{h^2}(1 - e^{-\kappa})}. \tag{7}$$

Note that when comparing to the original paper, the constants here are defined in a slightly different way. The reason is simply that Holden and Raynaud derive g on the interval $x \in [0, 1]$, whereas (7) represents the equivalent extension to any interval.

The convolution can then efficiently be calculated by the Fast Fourier Transform (FFT):

$$u = \mathcal{F}^{-1}(\mathcal{F}(g) \cdot \mathcal{F}(m)),$$

where $\mathcal{F}$ denotes the FFT and $\mathcal{F}^{-1}$ denotes the inverse FFT. Since g is constant, its fourier transform can be precalculated, which saves a lot of computation time. In fact, compared to solving the linear system by standard matrix techniques, we measured the scheme to consistently spend a staggering 77% less time when implemented with the FFT method.

Computation time, memory usage and high-resolution grids

Through active use of MATLAB's profiling tools, we have eliminated any substantial bottlenecks that are not directly required for the scheme to work. One could however shave another estimated $5 - 10\%$ off the running time at the cost of code clarity and readability, but at that point you may want to consider implementations in a more efficient language. In fact, the scheme is so efficient that computational time is typically not an issue unless one is generating a grid whose size is approaching or beyond the amount of memory found on a regular computer.

Due to the slow convergence of the scheme, one may want to generate very large grids to achieve high-resolution simulations for extended time periods. In such cases, the memory required to hold the grid may exceed any practical memory capacity available. As a workaround, we tested a technique where we ran the simulation in a stepwise manner, such that the interval $[0, T]$ would be subdivided into a number of smaller intervals $[\tau_j, \tau_{j+1}]$ for $j = 0, ..., J - 1$, where J is the number of subdivisions. Whenever a subdivision completes, the resulting matrix of that computation is compressed into a smaller matrix by interpolation of the grid, and the next subdivision is computed, using the results at time τ_j from the previous subdivision as its initial condition. Concatenating the compressed matrices yields a reasonably sized matrix whose structure is an accurate approximation of the high-resolution structure. Since the interpolation introduces (typically very small) additional errors, it is not a suitable technique for convergence verification, but it is very applicable in real world scenarios.

Conclusion

Based on papers [3] and [2] we have implemented a finite difference scheme with periodic boundary conditions for which convergence can be verified numerically when the analytic solutions of the input data are known. Furthermore, for other classes of initial conditions where we do not have an analytical solution, the scheme seems to behave in a way that is largely consistent with the literature on the equation.

A lot of time was spent investigating ways to analyze the convergence of the scheme analytically in a way that is within the scope of this course, but it appears that more advanced theory is required for the proof. We settled on proving consistency and presenting a possible stepping stone for an alternative proof to that of Holden and Raynaud which utilizes the fact that the scheme can be described completely by a single - though complicated - matrix. We leave it an open question whether it is possible to analyze this matrix analytically.

During the course of the project, we discovered an interesting and for us inexplicable effect. The forward Euler method in time gave superior results to that of higher order integrators, and it seemed to converge to the solutions offered by the higher order integrators as the time step was reduced and spatial step fixed. This is likely connected with the observations that smaller time steps resulted in greater errors, atleast up to a point where they stabilized. We have been unable to give a proper explanation for this phenomenon, and it may thus warrant further studies.

Finally, we discussed our experiences with the implementation. By profiling our code to figure out the main bottlenecks, we reduced both the computational time and complexity of our code significantly. In particular, the Fourier transformation suggested by Holden and Raynaud brought an immense improvement in computational efficiency compared to solving the linear system. To put computational speed into perspective, we measured the algorithm to generate 100MB of data per second (or equivalently roughly 12 million grid points per second) on an ordinary computer, which quickly places memory as the main restriction rather than computational speed when running simulations. We stress that our method has fairly little overhead in terms of memory - the large memory usage is simply a consequence of storing the computational results. We also briefly discussed how the memory usage issue for very high-resolution simulations can be circumvented by way of *piecewise compression*, which is the term we have coined for the method of running multiple dependent subsequent simulations and downsampling the results on-the-fly.

Finally, as is required, we are obliged to discuss our experiences with working as a team. Like a band of brothers, we have eaten together, laughed together, cried together and worked together, plunging onward into the darkness like a ship without a sail, trying to stay afloat as we crashed into rocks and debris, emerging shipwrecked and without a sense of direction as we were finally carried by a peakon onto the shores of enlightenment and understanding. Summarized in a slightly less dramatic way, we have worked together at nearly all times, sharing responsibilities equally amongst ourselves and checking each other's work. The project as a whole has been a challenging, yet rewarding experience, and working as a team has been an exclusively positive experience.

References

[1] Roberto Camassa and Darryl D Holm. An integrable shallow water equation with peaked solitons. *arXiv preprint patt-sol/9305002*, 1993.

[2] Morten Lien Dahlby. Geometric integration of nonlinear wave equations. 2007.

[3] Helge Holden and Xavier Raynaud. Convergence of a finite difference scheme for the camassa-holm equation. *SIAM journal on numerical analysis*, 44(4):1655–1680, 2006.