Tsunami Squares Implementation Changes to Improve Wave Resolution and Accuracy

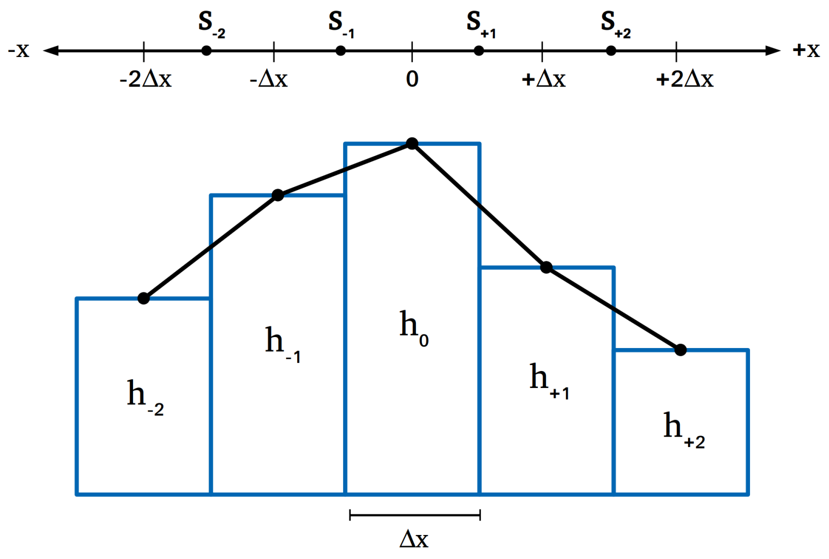
David Grzan[1], Steven Ward[2], John M Wilson[1], John B. Rundle[3], and Andrea Donnellan[4]

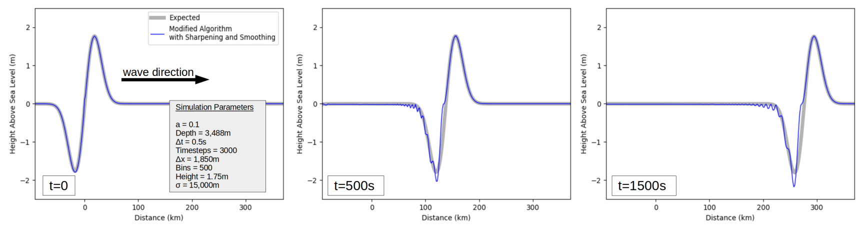
[1]University of California, Davis
[2]University of California, Santa Cruz
[3]University of California - Davis
[4]Jet Propulsion Laboratory, California Institute of Technology & University of Southern California

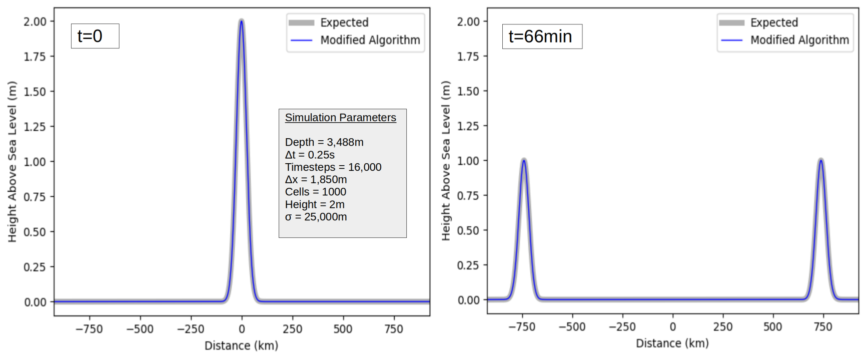
November 22, 2022

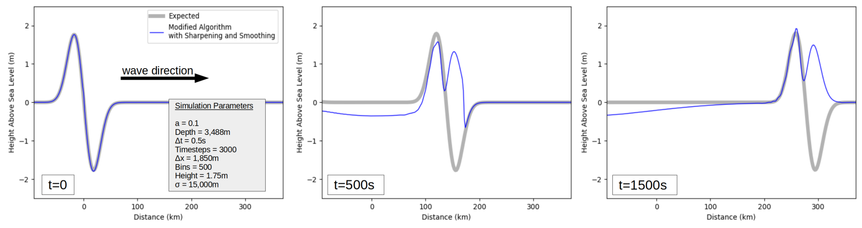
Abstract

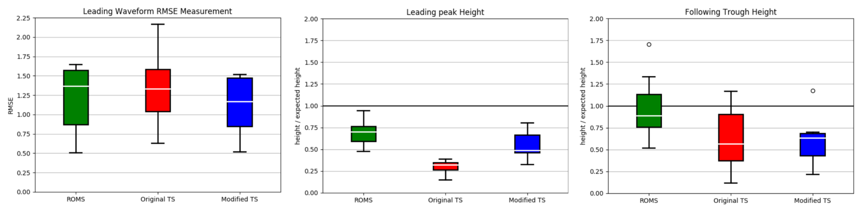
Tsunami Squares is a computationally lightweight tsunami and inundation simulator which utilizes a unique cellular automata technique. We make modifications to the underlying algorithm which result in increased accuracy and enhanced waveform resolution. These improvements leave Tsunami Squares well suited for machine learning applications where large pre-computed tsunami simulation databases are required. Previous implementations relied heavily on a smoothing algorithm which acts as a moving average applied to the water surface heights and velocities to eliminate anomalies at every time step. Although this allowed the simulation to function properly, it brings several unwanted effects such as reduced wave detail and lowered energy. A solution is found by shifting the location at which the water surface gradient is calculated, reducing the amount of anomalies in the simulation, and thus lowering the amount of smoothing needed by a factor of ~ 10. Also introduced is a new method to conserve energy locally, compared to previous methods which reference a simulation-wide energy calculation. We make comparison tests were made using the 2011 Tohoku tsunami along with the 2010 Maule tsunami to demonstrate the improvements made.

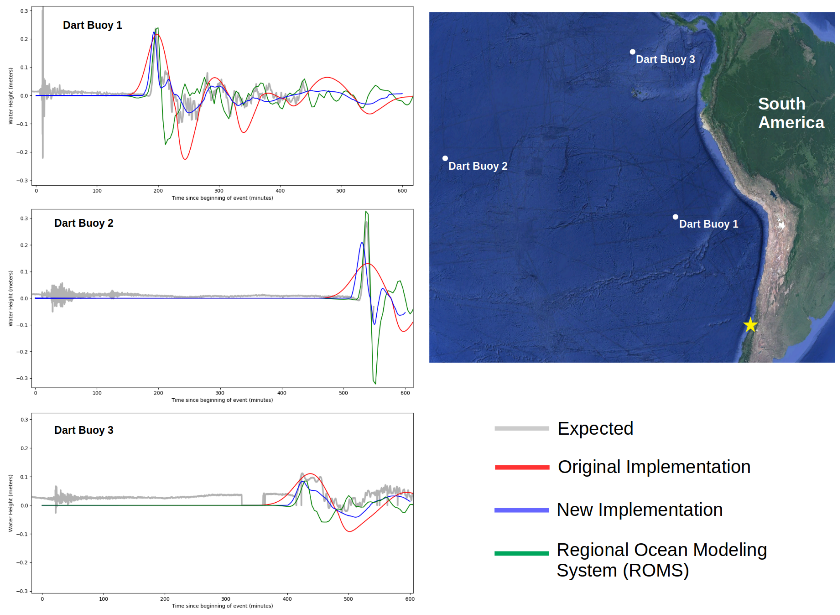



Expected
Modified Algorithm with Sharpening and Smoothing
wave direction
Simulation Parameters
a = 0.1
Depth = 3,488m
Δt = 0.5s
Timesteps = 3000
Δx = 1,850m
Bins = 500
Height = 1.75m
σ = 15,000m
t=0
t=500s
t=1500s
Height Above Sea Level (m)
Distance (km)


t=0
t=66min
Expected
Modified Algorithm
Simulation Parameters
Depth = 3,488m
Δt = 0.25s
Timesteps = 16,000
Δx = 1,850m
Cells = 1000
Height = 2m
σ = 25,000m
Height Above Sea Level (m)
Distance (km)


Expected
Modified Algorithm with Sharpening and Smoothing
wave direction
Simulation Parameters
a = 0.1
Depth = 3,488m
Δt = 0.5s
Timesteps = 3000
Δx = 1,850m
Bins = 500
Height = 1.75m
σ = 15,000m
t=0
t=500s
t=1500s
Height Above Sea Level (m)
Distance (km)


Leading Waveform RMSE Measurement
Leading peak Height
Following Trough Height
RMSE
height / expected height
ROMS
Original TS
Modified TS


Dart Buoy 1
Dart Buoy 2
Dart Buoy 3
Water Height (meters)
Time since beginning of event (minutes)
Dart Buoy 3
Dart Buoy 2
Dart Buoy 1
South America
Expected
Original Implementation
New Implementation
Regional Ocean Modeling System (ROMS)

Tsunami Squares Implementation Changes to Improve Wave Resolution and Accuracy

David P. Grzan, Steve N. Ward, John M. Wilson, John B. Rundle, Andrea Donnellan

September 10, 2020

Abstract

Tsunami Squares is a computationally lightweight tsunami and inundation simulator which utilizes a unique cellular automata technique. We make modifications to the underlying algorithm which result in increased accuracy and enhanced waveform resolution. These improvements leave Tsunami Squares well suited for machine learning applications where large pre-computed tsunami simulation databases are required. Previous implementations relied heavily on a smoothing algorithm which acts as a moving average applied to the water surface heights and velocities to eliminate anomalies at every time step. Although this allowed the simulation to function properly, it brings several unwanted effects such as reduced wave detail and lowered energy. A solution is found by shifting the location at which the water surface gradient is calculated, reducing the amount of anomalies in the simulation, and thus lowering the amount of smoothing needed by a factor of ~ 10. Also introduced is a new method to conserve energy locally, compared to previous methods which reference a simulation-wide energy calculation. We make comparison tests were made using the 2011 Tohoku tsunami along with the 2010 Maule tsunami to demonstrate the improvements made.

1 Introduction

Tsunami simulators are typically faced with a trade-off between accuracy and computational cost. An increase in the resolution or accuracy in a given tsunami simulation is usually compensated by an increase in computational cost. This poses several obvious problems for machine learning applications due to the computational resources needed in the creation of large pre-computed tsunami simulation databases. Recent machine learning applications include utilizing a convolutional neural network and a multilayer perceptron to estimate inundation in real time by linking low resolution simulations calculated in real time to pre-computed high resolution inundation maps (Fauzi & Mizutani, 2020). Other work has utilized a matching scheme using principal component analysis to link a similar database of low and high resolution simulations (Mulia, Gusman, & Satake, 2018, 2020). Other authors have used stochastic earthquake source models for the Nankai-Tonankai earthquake to quantify the uncertainty of a tsunami impact at coastal regions (Goda, Yasuda, Mai, Maruyama, & Mori, 2018). However, the work mentioned uses a collection of pre-computed tsunami source models only on the order of hundreds. Adding fault scenarios is essential to improve the effectiveness of these techniques as problems arise if there is no acceptable match in the database (Mulia et al., 2018, 2020; Fauzi & Mizutani, 2020). However, when increasing the number of simulations in the database, scenarios must be unique to maintain a statistical correlation between low resolution and high resolution simulations (Mulia et al., 2018, 2020).

Presented in this article are improvements made to Tsunami Squares, a computationally lightweight tsunami simulator which utilizes a cellular automata technique, leaving it well suited for these tsunami database creation applications. Originally inspired by a landslide simulation technique which simulates individual particles, or “balls” (Ward & Day, 2006), the method was adapted to water particles and named “Tsunami Balls” (Ward & Day, 2008, 2010, 2011). To obviate the need for millions of individual water particles, a new method, “Tsunami Squares”, was created which instead accelerates and transports “squares” of water to be imparted onto a grid, conserving volume and linear momentum (Xiao, Ward, & Wang, 2015). Another advantage of Tsunami Squares is that it gives no special treatment for wet and dry cells, allowing the free flow of water from sea to land. This method was tested on the 1982 El Picacho landslide in El Salvador (Wang, Ward, & Xiao, 2015), the 1792 Unzen-Mayuyama mega-slide in Japan (Wang, Ward, & Xiao, 2019), and the 2011 Tohoku Tsunami (Wilson et al., 2020). Originally written in Fortran, this method was later ported to C++ and parallelized for reduced computational time (Wilson et al., 2020).

The changes made to the most recent C++ ported version of Tsunami Squares consist of two small changes to how these individual squares of water are accelerated and transported. In addition, a new technique has been developed to ensure energy conservation. These changes allow Tsunami Squares to simulate waves with enhanced accuracy and resolution, whereas before, the simulations did not contain enough detail to be distinguished between other similar simulations, making it unfit for use in machine learning applications.

2 The Tsunami Squares Technique

Tsunami Squares uses a cellular automata method to equivalently solve nonlinear continuity and momentum conservation equations to obtain total column height $H(\mathbf{r}, t)$ and horizontal velocity $\mathbf{v}(\mathbf{r}, t)$ at every specified point for each time step

$$\frac{\partial H(\mathbf{r}, t)}{\partial t} = -\nabla \cdot [\mathbf{v}(\mathbf{r}, t) H(\mathbf{r}, t)] \tag{1}$$

and

$$\frac{\partial H(\mathbf{r}, t)\mathbf{v}(\mathbf{r}, t)}{\partial t} = -\nabla \cdot [\mathbf{v}(\mathbf{r}, t)\mathbf{v}(\mathbf{r}, t) H(\mathbf{r}, t)] - gH(\mathbf{r}, t)\nabla h(\mathbf{r}, t) \tag{2}$$

where g is the strength of gravity and $h(\mathbf{r}, t)$ is the surface height relative to sea level.

Instead of solving these differential equations explicitly, each column of water is propagated individually over Δt according to its position, velocity, and acceleration. After the final position is computed, the square's mass and momentum are split up among the overlapping squares. This process is demonstrated in the following figure.

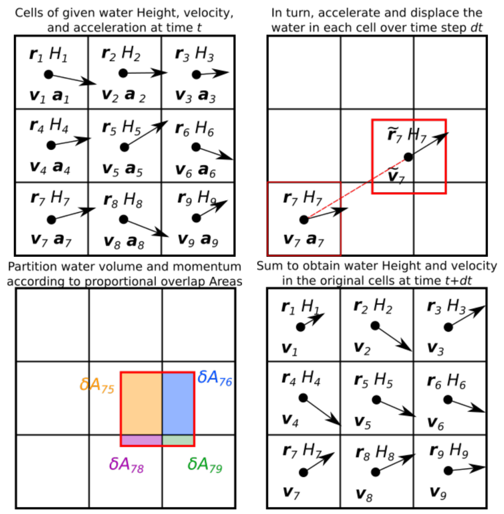

Cells of given water Height, velocity, and acceleration at time t
In turn, accelerate and displace the water in each cell over time step dt
Partition water volume and momentum according to proportional overlap Areas
Sum to obtain water Height and velocity in the original cells at time t+dt
r1 H1 v1 a1
r2 H2 v2 a2
r3 H3 v3 a3
r4 H4 v4 a4
r5 H5 v5 a5
r6 H6 v6 a6
r7 H7 v7 a7
r8 H8 v8 a8
r9 H9 v9 a9
r̃7 H7 ṽ7
δA75
δA76
δA78
δA79

Figure 1: The Tsunami Squares technique. Shown is one time step for one square

At the beginning of each time step, the acceleration of each square is calculated. It is caused by any unevenness on the surface of the water and is therefore expressed by a gradient term

$$\mathbf{a}_i(t) = -g\nabla h_i(t) \tag{3}$$

Once this is determined through a calculation of the slope at that point, the final position can be calculated. However, the slope and final position can be computed multiple ways. Many experiments suggest that new methods to calculate these two quantities yield better performance. This means greater accuracy, improved energy conservation, and more detail contained in the waveform. We will now discuss the differences in implementation and how this impacts the resulting wave.

Previous implementations of Tsunami Squares have calculated the slope by using the central difference approximation. For a 1D wave in the x-direction this is

$$\mathrm{S} = \frac{h_{i+1} - h_{i-1}}{2\Delta x} \tag{4}$$

where Δx is the width of the square and S is the slope. The final distance is then calculated according to standard kinematics

$$\tilde{\mathbf{r}}_i(t) = \mathbf{r}_i + \mathbf{v}_i(t)dt + \tfrac{1}{2}\mathbf{a}_i(t)dt^2 \tag{5}$$

To compare, the current implementation of Tsunami Squares calculates the slope in the following way. It first considers the direction of the individual square-v (square velocity), then takes the slope by considering only itself and the square in that direction. For example in 1D, if square_v points to the right, the slope is calculated by only considering the current height h_i and the height to right of it h_{i+1}.

$$\mathrm{S}_{\mathrm{square\_}v>0} = \frac{h_{i+1} - h_i}{\Delta x} \tag{6}$$

$$\mathrm{S}_{\mathrm{square\_}v<0} = \frac{h_i - h_{i-1}}{\Delta x} \tag{7}$$

The final distance is then calculated by

$$\tilde{\mathbf{r}}_i(t) = \mathbf{r}_i + \mathbf{v}_i(t)dt + \mathbf{a}_i(t)dt^2 \tag{8}$$

which differs from the previous implementation by a factor of $\frac{1}{2}$ in the term containing acceleration. The new distance formula is equivalent to transporting the square according its final velocity $(\mathbf{v}_i(t) + \mathbf{a}_i(t)dt)$ instead of the average velocity as is used in standard kinematic formulas.

A 1D stationary Gaussian pile of water will be simulated and allowed to propagate left and right to show the difference between the original implementation and the new implementation. An expected solution is shown by solving the linear shallow water equations using a finite difference method. We present results using the original implementation defined by equations 4 and 5 followed by the results using the new implementation defined by equations 6, 7 and 8.

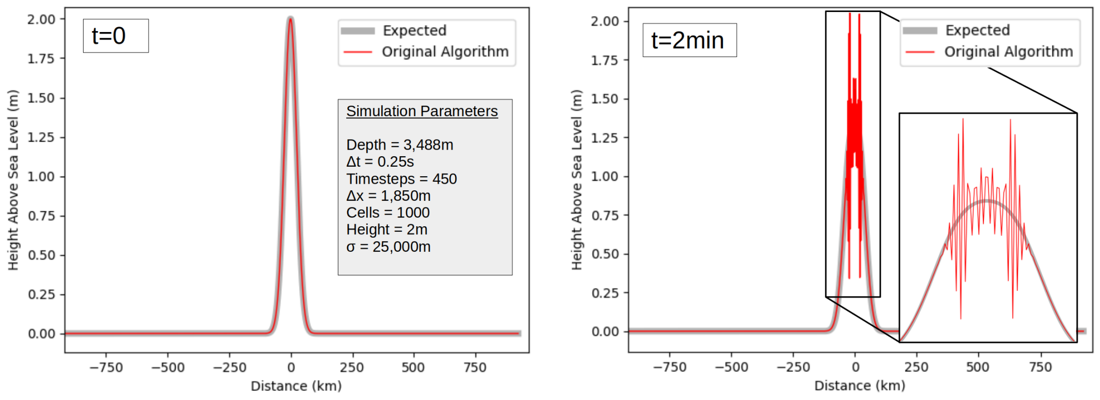


Figure 2: A 1D example of the original Tsunami Squares implementation using equations 4 and 5 starting from an initial stationary Gaussian pile of water (left). The wave propagates for a few time steps before experiencing erroneous behavior (right). Even with smooth initial conditions and small Δt, the original Tsunami Squares implementation cannot stably propagate waves.

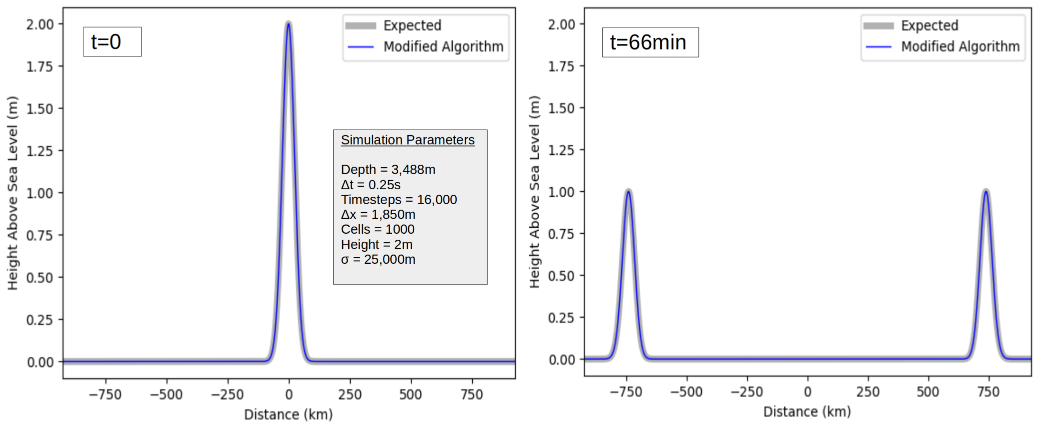


Figure 3: A 1D example of the modified Tsunami Squares implementation using equations 6,7, and 8 starting from an initial stationary Gaussian pile of water (left). The wave then propagates in alignment with the expected analytical result for 16,000 timesteps (right).

Results show clearly that the new way of calculating the slope (equations 6 and 7) in combination with the new way of calculating the final position (equation 8) allows the wave to propagate in exact agreement with the expected solution. Of course, these results are under ideal conditions (small Δt and a smooth waveform). If one is to move on to a simulation with conditions that are not ideal, such as in virtually every realistic tsunami scenario, one needs a way to smooth out the errors that will eventually arise due to sudden changes in bathymetry, effects at shore, and non-ideal waveforms. In the original implementation, such a smoothing algorithm is necessary for the wave to function properly for any condition.

The next section details the algorithm designed to smooth out these errors and explains why if both implementation choices need smoothing, why does it matter which version is used?

3 Smoothing

No matter how well the algorithm works on its own, sharp changes in bathymetry, effects near land, and non-smooth waveforms will cause errors that will grow exponentially over time. Therefore after the Tsunami Squares algorithm has been carried out for each square, a smoothing algorithm is then applied to even out any anomalies that will give rise to these nonphysical effects. To demonstrate the overall effects the smoothing algorithm has on a wave that is experiencing erroneous behavior, smoothing has been applied to the wave produced by the original algorithm in Figure 2.

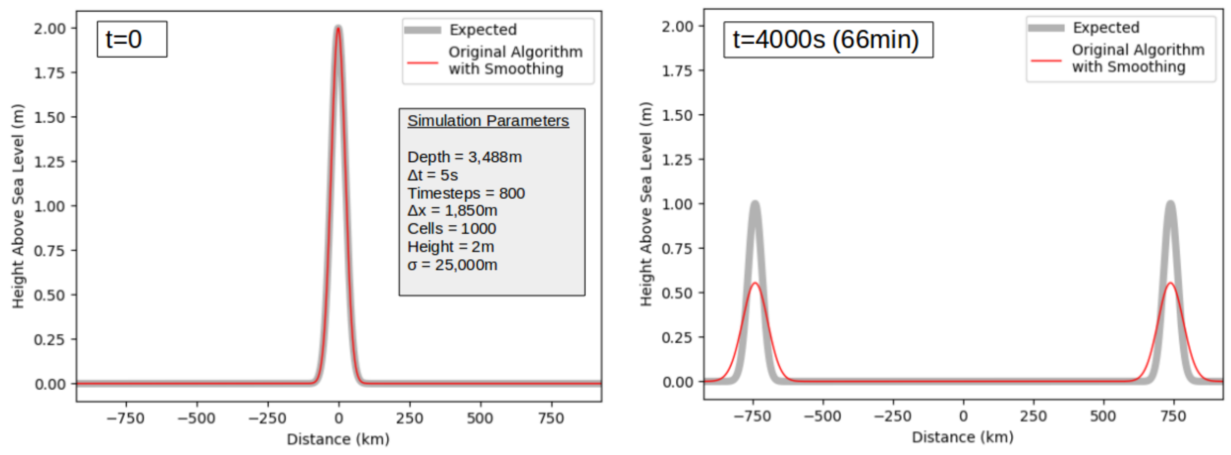


Figure 4: The 1D Tsunami Squares simulation from figure 2 with smoothing (left). The waves then propagate left and right after which the clear effects of smoothing can be seen (right). It results in a wave with a lower amplitude, greater width, and lower energy (55% of initial).

The wave that is produced by the original simulation has errors, but these are removed or reduced by the smoothing procedure, allowing the waves to propagate virtually error-free to left and right. However, as can be seen, the smoothing algorithm gives rise to several unwanted effects. These are listed below.

- Reduced amplitude
- Increased width
- Reduced energy

It is therefore the case that *less smoothing is better*. This then answers the question of why it is more beneficial to use the new implementation over the original implementation, when in practice both require smoothing. It is because the new implementation needs significantly less smoothing, and will therefore produce fewer of these undesired effects. The original implementation needs smoothing to operate, whereas the new implementation merely needs it for maintenance as it can function to a good degree on its own.

Even with the little amount of smoothing needed by the new implementation for maintenance, over large distances these undesirable effects come into play. Most notably, energy is lost. To conserve energy, it must manually added back in. There are multiple techniques that can be used to accomplish this which will be discussed in the next section.

As for how the smoothing algorithm is carried out, it in essence takes some water from higher squares and transfers it to the neighboring lower squares. The following figure depicts a simplified 1D example containing only two squares for the purpose of demonstration.

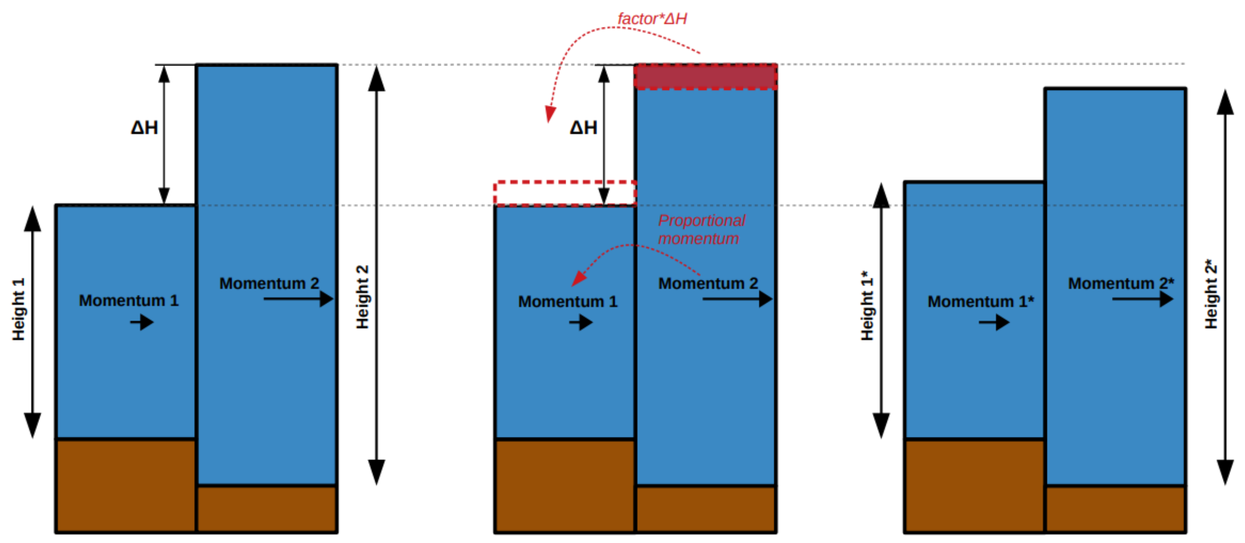


Figure 5: The smoothing algorithm showing the case for two columns of water. This is done by taking water from the right column (the higher column) and moving it to the left column (the lower column). Left: the columns before smoothing is applied. Middle: the algorithm in progress. Right: the final result.

The smoothing is done for both the height and velocity of each square. The amount of water transferred is proportional to the difference in height of the two squares. The factor f determines how much of that height difference is transferred from one square to the other. In Figure 5, this is depicted by the shaded red part in the center picture.

$$h_{\text{transfered}} = (h_2 - h_1) \cdot f \tag{9}$$

$$v_{\text{transfered}} = (v_2 - v_1) \cdot f \tag{10}$$

where h_1 and h_2 and the heights relative to sea level and the factor f is the factor chosen by the user. The higher the value of f, the greater the smoothing.

4 Adding Energy

After changes were made to the Tsunami Squares algorithm detailed in equations 6, 7, and 8, waves can now propagate on their own with enhanced stability (see figures 2 and 3). However, although the modified algorithm is now stable under ideal conditions (smooth waveform, small timesteps, flat seafloor), non-ideal conditions cause the wave to experience errors that grow over time. To remedy this, a smoothing algorithm as been introduced which acts as a moving average applied across all squares in the simulation. This has unfortunate side effects of decreasing the amplitude and energy over time. The solution to these undesired effects is to manually add energy back into the simulation. A new technique used to accomplish this will be described in this section.

The technique described here has the effect of "sharpening" the wave. This means the amplitude of the waveform increases over time and width of the wave is decreases over time. The following figure shows the desired effect.

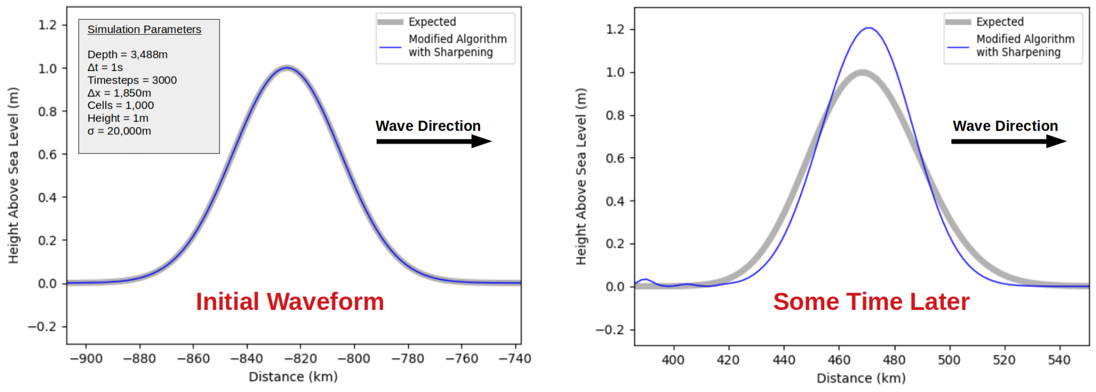


Figure 6: Blue: A positive wave showing the effect of sharpening. Gray: Finite difference solution. Over time the wave's width decreases and its height increases, increasing the energy substantially as well).

This is ideal because the smoothing algorithm has the exact opposite effect on the waveform. The combination of these two effects leaves the original waveform unchanged, while smoothing out any anomalies.

The sharpening effect is accomplished by shifting the acceleration towards the opposite direction of wave-v (wave velocity) by a small amount. This amount is denoted by a, which is defined by the following equation.

$$a = \frac{\text{(distance moved)}}{\Delta x} \tag{11}$$

where Δx is the square length. $a = 1.0$ would therefore shift the acceleration by one full square length or 1 Δx. The following figure shows this process.

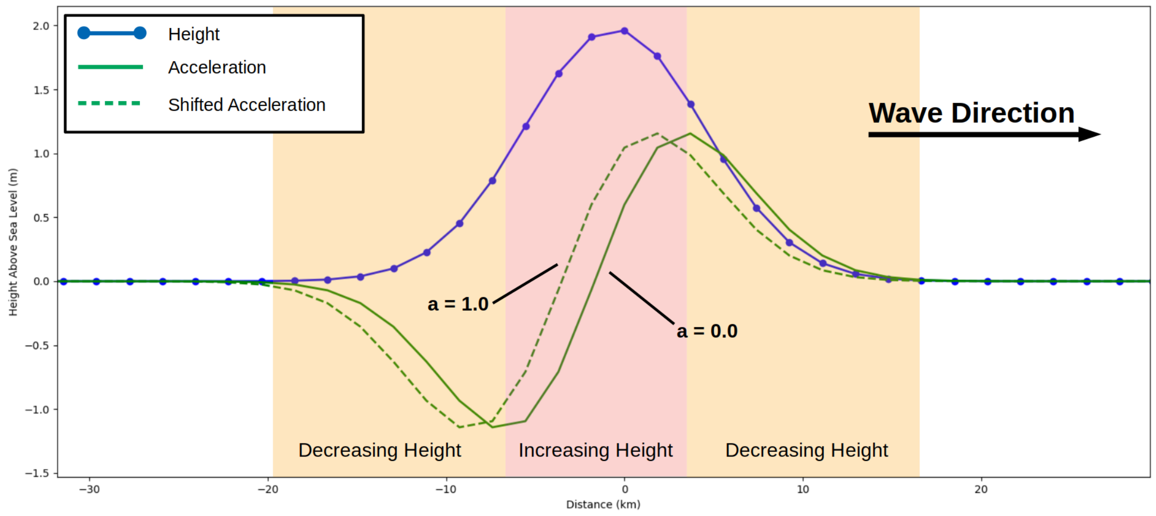


Figure 7: Diagram showing how a shift in the acceleration curve will cause an increase in the acceleration in the center (red shaded region) and a decrease in acceleration in the outer regions (orange shaded regions). An increase in acceleration leads to an increase of velocity which in turns increases the height. This acceleration shift ultimately leads to the a sharpening effect on the wave, increasing its energy.

By shifting the acceleration curve, acceleration is either raised or lowered from its original value. The more the acceleration is shifted, the greater this increase or decrease. Regions where the acceleration is raised cause a rise in height and regions where the acceleration is lowered causes a decrease in height. This is indicated by the red and orange shaded regions in figure 7. An increase in height follows from an increase in acceleration because as the acceleration rises, so do the square velocities, causing the square heights to increase as well.

The diagram below will be used to describe exactly how the slope is taken with this newly added wave-v directional dependency.

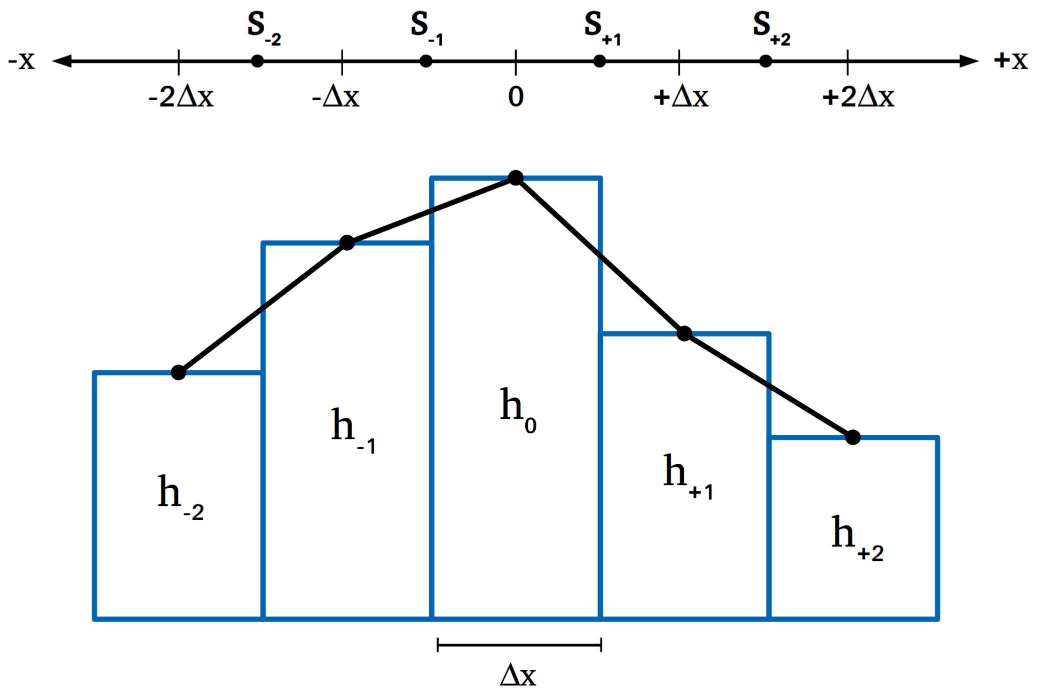


Figure 8: A 1D example showing positions at which the slope is well defined (S_{-2}, S_{-1}, S_{+1}, S_{+2}). Slopes at all positions in between these points can calculated by linear interpolation.

Figure 8 shows distance from the current square and the slopes which are defined at certain locations (S_{-2}, S_{-1}, S_{+1}, S_{+2}). We can further define the slope at every position by a simple linear interpolation. This gives us the opportunity of shifting the acceleration (or equivalently shifting the slope) at intervals smaller than Δx.

Since the acceleration is to be shifted based on the direction of wave-v, one piece of information must first be made clear. In general for well behaved waves, wave-v points in the same direction as square-v if h is positive and wave-v points in the opposite direction as square-v if h is negative. So by looking at square-v and the height, which is well defined for every square, the wave-v direction can be estimated to good approximation.

To demonstrate the process of determining the slope, an example will be given. Say we want to shift the acceleration by $0.25\Delta x$ (a common and well performing choice) for the wave given in figure 7. The first step is to consider square-v to determine if S_{-1} or S_{+1} will be used. This slope is the fundamental algorithm's slope detailed in equations 6 and 7 and acts as the starting point before any shifting has occurred. After checking that the sign of square-v is positive, S_{+1} is taken.

To shift the acceleration, the wave-v direction must be known. It can be estimated that for a square with positive h and positive square-v, wave-v must also be positive. Since we need to shift the slope towards the opposite direction of wave-v, the slope must be taken that distance towards the direction of wave-v. We now know we want to take the slope $0.25\Delta x$ to the right of S_{+1}. We can therefore interpolate between S_{-1} and S_{-1} to find the desired shifted slope.

$$
\begin{aligned}
S_{\text{shifted}} &= S_{+1} + 0.25\Delta x\frac{(S_{+2} - S_{+1})}{\Delta x} \\
&= S_{+1} + 0.25S_{+2} - 0.25S_{+1} \\
&= 0.75S_{+1} + 0.25S_{+2} \\
&= (1-a)S_{+1} + (a)S_{+2}
\end{aligned}
\tag{12}
$$

where written more generally, a is the fraction of Δx the wave is to be shifted.

The shifted slope can be written for all combinations of h and square-v:

$$S_{\text{shifted}} = (1-a)S_{+1} + (a)S_{+2} \qquad (h > 0,\ \text{square-}v > 0) \tag{13}$$

$$S_{\text{shifted}} = (1-a)S_{-1} + (a)S_{-2} \qquad (h > 0,\ \text{square-}v < 0) \tag{14}$$

$$S_{\text{shifted}} = (1-a)S_{+1} + (a)S_{-1} \qquad (h < 0,\ \text{square-}v > 0) \tag{15}$$

$$S_{\text{shifted}} = (1-a)S_{-1} + (a)S_{+1} \qquad (h < 0,\ \text{square-}v < 0) \tag{16}$$

For a given simulation, choosing a value in the range of $0.1 < a < 0.3$ have been tested to perform most accurately. For each value of a, a specific smoothing factor f (equations 9 and 10) can be calculated to exactly cancel out the sharpening effects. This factor f was determined empirically and depends on a, Δx, Δt and height of water column H:

$$f = 0.24\frac{\sqrt{gH}\Delta t \cdot a}{\Delta x} \tag{17}$$

By using a value of $0.1 < a < 0.3$ and the smoothing factor given above, the wave will now conserve energy and preserve shape while eliminating any erroneous behavior.

5 Additional Fixes

The methods described thus far operate as they should for waves which contain only positive or only negative heights. For more complicated waveforms that include both positive components and negative components, problems arise. With the great reduction of smoothing due to the introduction of the new modifications and methods, errors that were previously smoothed over have now surfaced. These errors are shown in the following two figures.

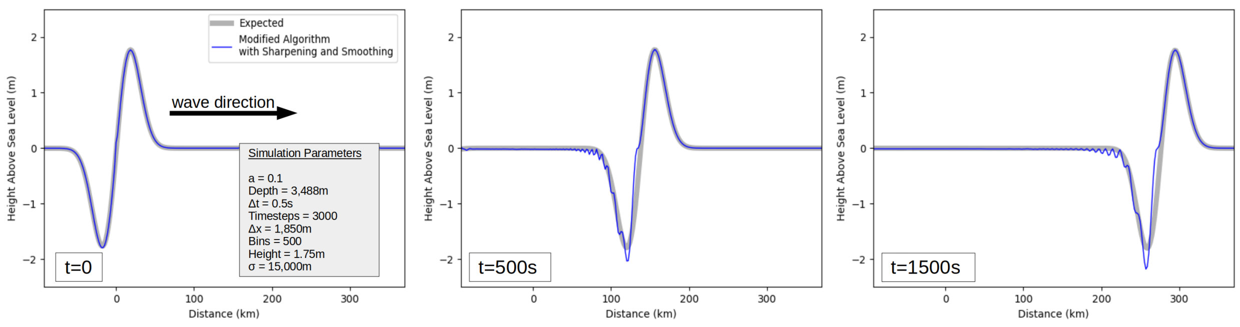


Figure 9: Errors at the boundary between a negative and positive wave, showing the case for a positive slope traveling to the right. This contains the case where there are two squares with square-v pointing away from each other.

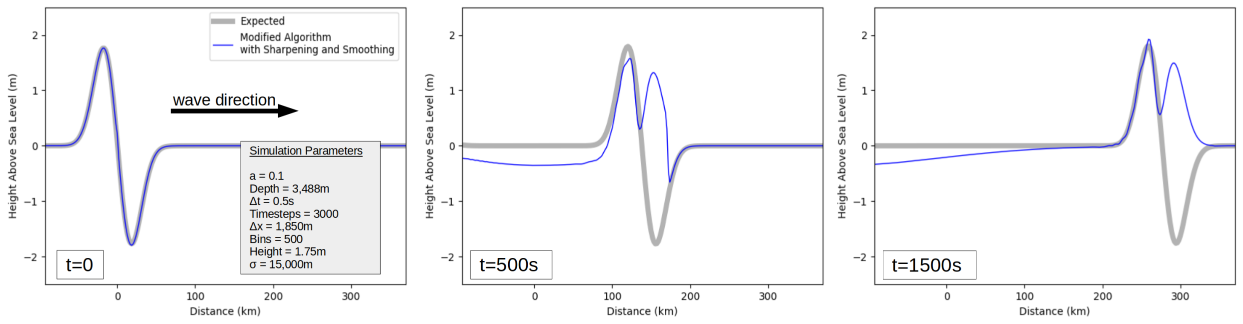


Figure 10: Errors at the boundary between a negative and positive wave, showing the case for a negative slope traveling to the right. This contains the case where there are two squares with square-v pointing towards each other.

The two types of errors shown are due to a change in the direction of square-v from one square to another. The error in the first image is caused by two squares at $h = 0$ which have square velocities pointing away from each other. The error in the second image is caused by two squares which have square velocities pointing inwards towards each other. In well behaved waves, water is exchanged uniformly from left to right or visa versa (indicated by the direction of square-v). If this direction changes there is a discontinuity in this exchange of water, leading to erroneous behavior as shown in the previous two images.

The solution to this problem involves modifying the acceleration and velocity of the two squares which have opposing square-v directions. Not only do these modifications depend on the square-v direction, but also the slope of h at that point (whether it is positive or negative). Therefore there are four total scenarios which must be treated with their own modifications. The four scenarios can be categorized as the cases of figure 9 and 10 plus the cases where the waves are traveling in the opposite direction. These rules are shown below.

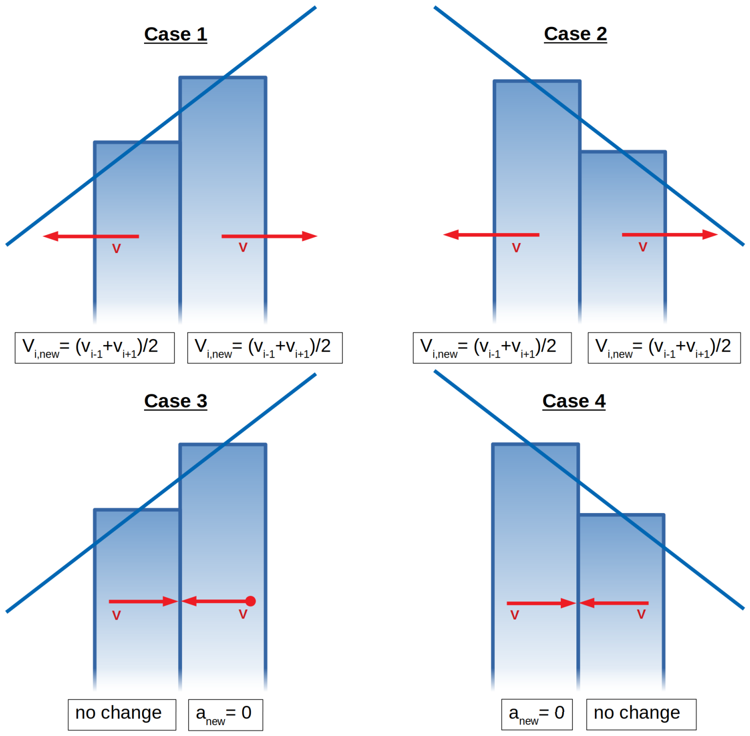


Figure 11: The changes to acceleration and velocity that must be applied to remedy the problems experienced at the boundary between positive and negative square-v shown in figures 9 and 10. There are four scenarios relating to the four unique combinations of positive and negative height and square-v slopes present at the square-$v = 0$ boundary. It is only specified what is changed. If acceleration or velocity is not mentioned, it is left unchanged.

The rules for each pair of squares are also given in the following table. For consistency's sake, case 1 corresponds to the upper left portion of the table, case 2 corresponds to the upper right and so on.

		Height Slope			
		Positive		Negative	
		Left	Right	Left	Right
Velocity Slope	Positive	$v_{i,\text{new}} = \frac{(v_{i-1}+v_{i+1})}{2}$	$v_{i,\text{new}} = \frac{(v_{i-1}+v_{i+1})}{2}$	$v_{i,\text{new}} = \frac{(v_{i-1}+v_{i+1})}{2}$	$v_{i,\text{new}} = \frac{(v_{i-1}+v_{i+1})}{2}$
	Negative	no change	$a_{\text{new}} = 0$	$a_{\text{new}} = 0$	no change

Table 1: Rules to remedy problems located where a pair of squares crosses square-$v = 0$. It is only specified what is changed. If acceleration or velocity is not mentioned, it is left unchanged. These rules are applied separately for both the x and y direction, where left and right correspond to bottom and top squares.

It can be seen that the rules involve setting the acceleration to zero for some squares and setting the velocity to the average velocity for others. With these rules implemented, the simulation can now run stably and reliably, free of the errors observed in figures 9 and 10.

6 Full Simulation Results

The following result compares a simulation using the original implementation compared to several simulations using the new implementation (each with different levels of smoothing). The event used in this comparison is the 2011 Tohoku event with an initial condition obtained by inverting wave gauge data (Song, Fukumori, Shum, & Yi, 2012; Song, 2007).

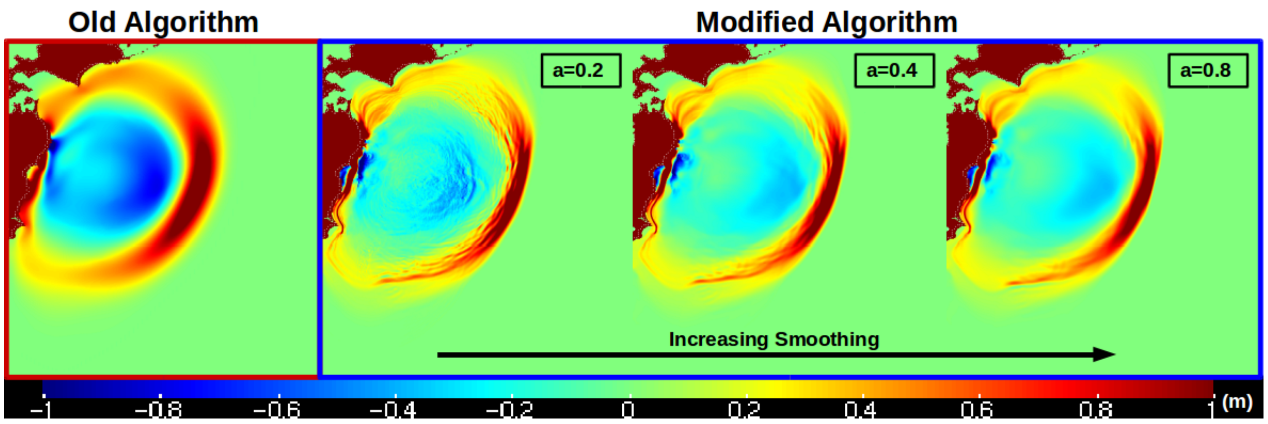


Figure 12: Far left: Results of the 2011 Tohoku event using the original implementation. Right three: Results using the new implementation, showing increasing amounts of sharpening (indicated by the value "a") which corresponding increasing amounts of smoothing.

The value a used in each simulation in figure 12 is proportional to how much smoothing is used (equation 17). It follows from this equation that as the value a increases, in turn so does the amount of smoothing used.

Compared to the original algorithm, the modified algorithm produces a simulation with enhanced detail. This is possible because of the significant reduction in the amount of smoothing necessary. To quantify, the smoothing factor (equations 9 and 10) used in the new implementation is 5-15 times less than the original implementation.

For a more detailed comparison, buoy data from the 2011 Tohoku event and the 2010 Maule event were used to compare the original implementation, new implementation, and the Regional Ocean Modeling System (ROMS). ROMS is a verified wave simulator which solves three dimensional Reynolds-averaged Navier-Stokes equations using a finite difference method and has been applied to several tsunami studies (Haidvogel et al., 2008; Song, Mohtat, & Yim, 2017). It is used in the comparison to give Tsunami Squares a reference to an accurate tsunami simulator which uses the same initial conditions derived by inverting wave gauge data (Song et al., 2012; Song, 2007).

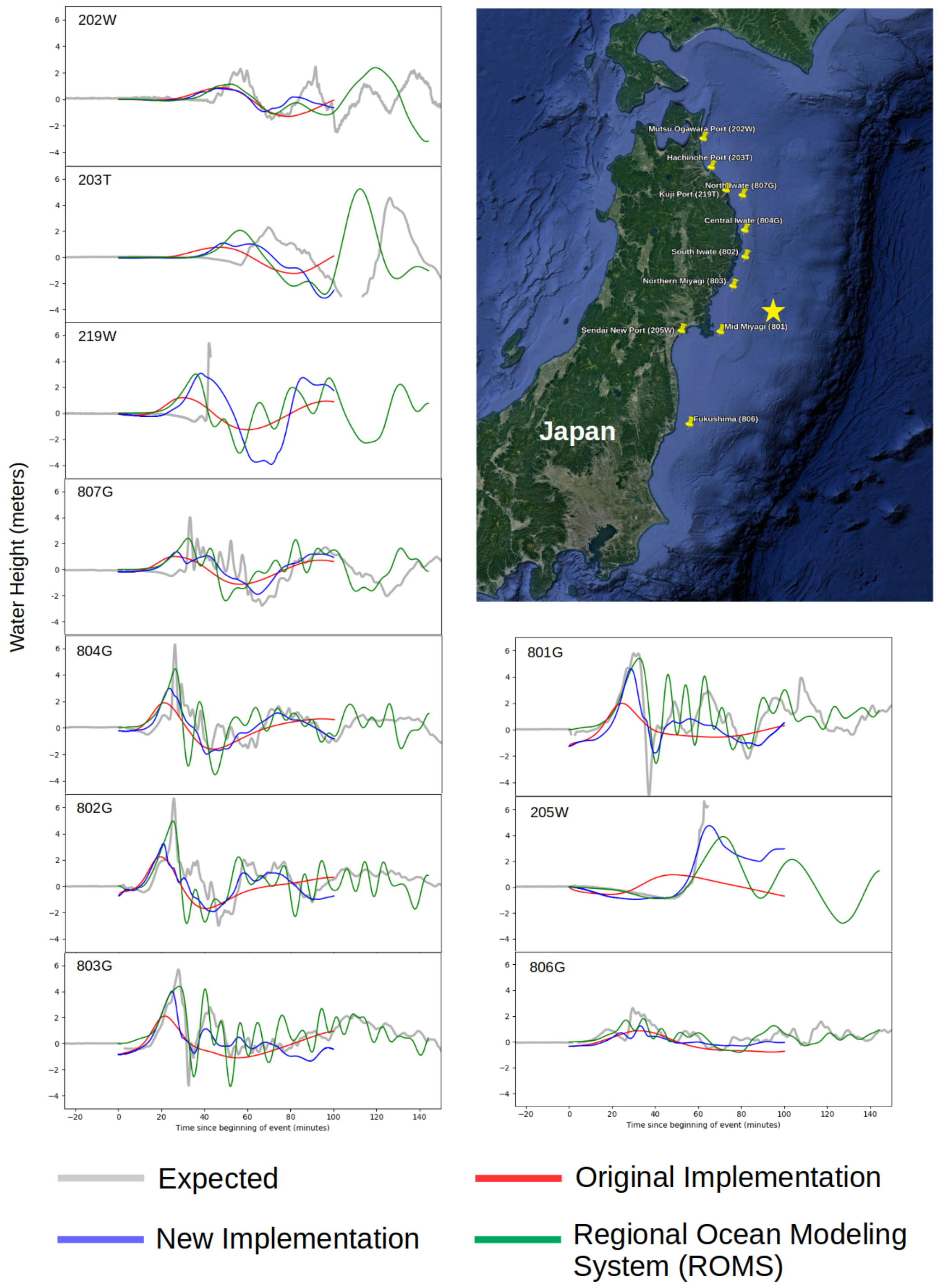


Figure 13: Buoy comparison plots for the 2011 Tohoku Tsunami. The simulation using the new implementation uses an a value of 0.4. Results show better alignment and more detail for the new implementation in nearly all cases.

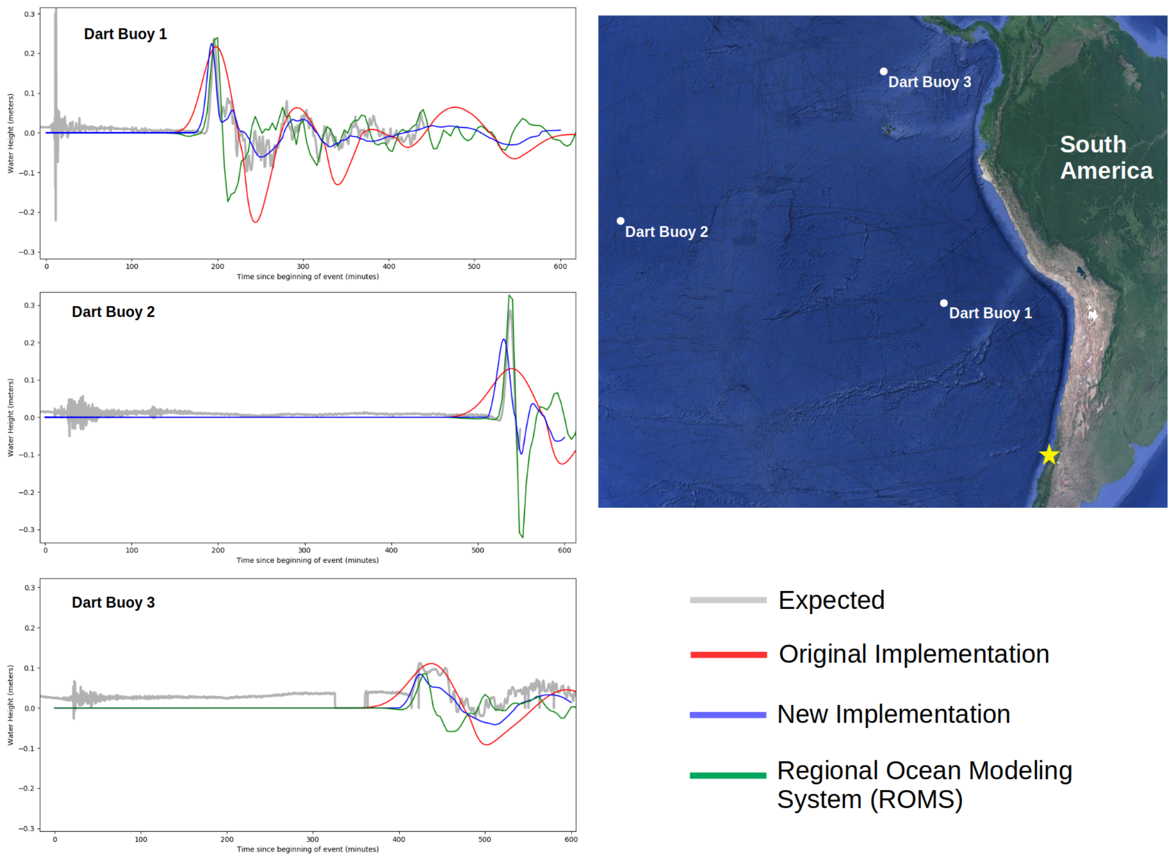


Figure 14: Buoy comparison plots for the 2010 Maule Tsunami. The simulation using the new implementation uses an a value of 0.25. Results show better alignment in nearly all cases.

The most important features in comparison plots such as these are the arrival time, wave height, and width. Arrival time information is important for early warning purposes, and the height and width of the initial peak are an indication overall wave energy. It would also affect the complicated dynamics of water flow onto land.

In the plots shown show overall greater alignment in the aspects of wave height, width, and arrival time. Sources of error in the comparison between Tsunami Squares and ROMS include lower resolution bathymetry near coast which is known to affect wave dynamics (Tang et al., 2008), higher timestep, and using instantaneous seafloor uplift instead of a time-dependent one.

Errors for root mean square error, initial peak height, and following trough height are shown below for the 2011 Tsunami event. Expected values were calculated from buoy measurements.

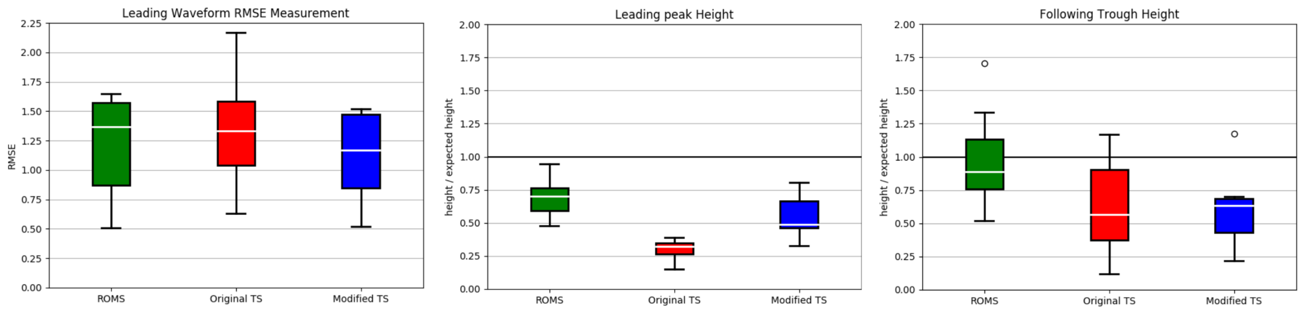


Figure 15: Error analysis for the 2011 Tohoku event. Left: Root mean square error using data from the start of the wave to the first trough. Middle: Ratio of the initial peak height to the height measuremed by the buoy. Right: Ratio of the initial trough to the expected height measured by the buoy. Black line indicates expected result for the middle and right plots.

7 Conclusion

Several modifications were made to the Tsunami Squares algorithm which yield higher accuracy simulations and greater detailed waveforms. Before the changes, Tsunami Squares could not provide enough uniqueness between simulations with similar initial conditions, affecting the effectiveness of a searching algorithm in a given pre-computed tsunami database (Mulia et al., 2018, 2020). The modifications now leave Tsunami Squares well suited for applications which require the creation of such a database.

The changes made to the algorithm involve a simple change to how the slope and distance traveled are calculated for a given square, resulting in a more stable wave which requires significantly less smoothing to function. Smoothing is a added function to Tsunami Squares which takes a given waveform and eliminates any sharp features, allowing the wave to propagate with more stability. However, the more smoothing that is used, the more detail that is lost. So a reduction in the amount of smoothing is something that is highly beneficial for overall performance.

A new method is introduced to conserve energy. It works by shifting the acceleration a small distance towards the opposite direction of wave travel. In doing so, it produces a sharpening effect on the wave which increases the height, decreases the width, and ultimately increases the energy. Since the smoothing decreases height, increases width, and decreases energy, these two effects cancel each other out leaving behind the original wave while smoothing over all of the small anomalies. The advantages of this method is that it doesn't rely on a global calculation of wave energy, meaning it conserves energy locally and allows for energy fluctuation due to complex interactions near shorelines.

Lastly, changes were made to the square-v and acceleration of squares which have opposing square-v directions. The significant reduction of smoothing uncovered errors at these locations that were previously smoothed over in the original algorithm. These errors affected the overall waveform and were detrimental to the overall accuracy of the simulation. The empirically derived fixes have since eliminated these problems.

8 Acknowledgments

The research of DG and JBR was supported by NASA grant (NNX17AI32G) to UC Davis. Portions of the research by AD were carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. None of the authors have identified financial conflicts of interest. The authors thank Tony Song for providing the ROMS simulation data used as a comparison. ROMS is available from https://www.myroms.org. Additional thank you to Ronaldo Ortez for proofreading and edits.

Table 2: Variable Definitions

Variable	Definition
h	height of water surface relative to sea level
H	full height of column of water from surface to floor
d	depth, measured from sea floor to sea level
square-v	velocity of an individual water column
wave-v	wave velocity ($\sqrt{gd}$)
Δx	width of a given square
Δt	length of the timestep
timesteps	the total number of iterations in time
t	the in-simulation time (timesteps $\cdot$ Δt)
Height, σ	The height and standard deviation of a Gaussian initial condition
cells	the number of bins in the simulation
a	value from 0 to 1.0 which sharpens wave, resulting in an increase in wave energy
f	smoothing factor, determines how smooth the wave is and results in decreased total wave energy

9 Appendix

References

Fauzi, A., & Mizutani, N. (2020). Machine learning algorithms for real-time tsunami inundation forecasting: a case study in nankai region. *Pure and Applied Geophysics*, *177*(3), 1437–1450.

Goda, K., Yasuda, T., Mai, P. M., Maruyama, T., & Mori, N. (2018). Tsunami simulations of mega-thrust earthquakes in the nankai–tonankai trough (japan) based on stochastic rupture scenarios. *Geological Society, London, Special Publications*, *456*(1), 55–74.

Haidvogel, D. B., Arango, H., Budgell, W. P., Cornuelle, B. D., Curchitser, E., Di Lorenzo, E., ... others (2008). Ocean forecasting in terrain-following coordinates: Formulation and skill assessment of the regional ocean modeling system. *Journal of Computational Physics*, *227*(7), 3595–3624.

Mulia, I. E., Gusman, A. R., & Satake, K. (2018). Alternative to non-linear model for simulating tsunami inundation in real-time. *Geophysical Journal International*, *214*(3), 2002–2013.

Mulia, I. E., Gusman, A. R., & Satake, K. (2020). Applying a deep learning algorithm to tsunami inundation database of megathrust earthquakes. *Journal of Geophysical Research: Solid Earth*, *125*(9), e2020JB019690.

Song, Y. T. (2007). Detecting tsunami genesis and scales directly from coastal gps stations. *Geophysical Research Letters*, *34*(19).

Song, Y. T., Fukumori, I., Shum, C., & Yi, Y. (2012). Merging tsunamis of the 2011 tohoku-oki earthquake detected over the open ocean. *Geophysical Research Letters*, *39*(5).

Song, Y. T., Mohtat, A., & Yim, S. C. (2017). New insights on tsunami genesis and energy source. *Journal of Geophysical Research: Oceans*, *122*(5), 4238–4256.

Tang, L., Titov, V., Wei, Y., Mofjeld, H., Spillane, M., Arcas, D., ... Newman, J. (2008). Tsunami forecast analysis for the may 2006 tonga tsunami. *Journal of Geophysical Research: Oceans*, *113*(C12).

Wang, J., Ward, S. N., & Xiao, L. (2015). Numerical modelling of rapid, flow-like landslides across 3-d terrains: a tsunami squares approach to el picacho landslide, el salvador, september 19, 1982. *Geophysical Journal International*, *201*(3), 1534–1544.

Wang, J., Ward, S. N., & Xiao, L. (2019). Tsunami squares modelling of the 2015 june 24 hongyanzi landslide generated river tsunami in three gorges reservoir, china. *Geophysical Journal International*, *216*(1), 287–295.

Ward, S. N., & Day, S. (2006). Particulate kinematic simulations of debris avalanches: interpretation of deposits and landslide seismic signals of mount saint helens, 1980 may 18. *Geophysical Journal International*, *167*(2), 991–1004.

Ward, S. N., & Day, S. (2008). Tsunami balls: a granular approach to tsunami runup and inundation. *Communications in computational Physics*, *3*(1), 222–249.

Ward, S. N., & Day, S. (2010). The 1958 lituya bay landslide and tsunamia tsunami ball approach. *Journal of Earthquake and Tsunami*, *4*(04), 285–319.

Ward, S. N., & Day, S. (2011). The 1963 landslide and flood at vaiont reservoir italy. a tsunami ball simulation. *Italian Journal of Geosciences*, *130*(1), 16–26.

Wilson, J. M., Schultz, K. W., Grzan, D., Rundle, J. B., Ward, S. N., Bhaskar, R., … Kaushal, H. (2020). Tsunami squares simulation of megathrust-generated waves: Application to the 2011 tohoku tsunami. *Progress in Disaster Science*, *5*, 100063.

Xiao, L., Ward, S. N., & Wang, J. (2015). Tsunami squares approach to landslide-generated waves: application to gongjiafang landslide, three gorges reservoir, china. *Pure and Applied Geophysics*, *172*(12), 3639–3654.