On 2D Shallow Water Wave Equation

Kenta Funada, Shwetabh Singh, Vrutant Patel

May 11, 2022

Project Report - AOE 5404

Abstract

Shallow water equations define a certain class of waves with wavelengths much larger than the depth of the waves and are observed in nature in forms of tides and tsunamis. In this project, We numerically solve the Shallow waves equation, for both 1D and 2D case using the Lax-Wendroff scheme.We also try to look at the impact of barriers on the solution and how propagation of multiple drops happen.

Introduction

For the course project, we are trying to solve the 2D wave equation for certain physical conditions. The general wave equation is

$$\bigtriangledown \Psi = \frac{1}{c^2} \frac{\partial^2 \Psi}{\partial t^2} \tag{1}$$

which is a second order linear partial differential equation which describes the propagation of waves in a media. The characteristics of the media are included in the dispersion relation. This gives us a way to model a lot of wave propagation phenomena in various physical media like water, oil etc. For 2D it becomes

$$\frac{\partial^2 \Psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = \frac{1}{c^2} \frac{\partial^2 \Psi}{\partial t^2} \tag{2}$$

Shallow Water wave equations as a model were first described in 1871 by Saint-Venant to model a flow in open channel. We can look at them as a special case of 2D Navier Stokes equation under the assumption that the wavelength of the modeled phenomenon is larger than the depth of the basin, hence Shallow water. Physically this model can model a water wave on the surface of the body like a tsunami or a drop of water or oil falling on a stagnant surface. The damping can model physical situations like a tsunami over a shallow water conditions [3]. The model is important from an engineering point of view as it can help model rivers, flooding, dam breaks, overland flow and tsunamis over different topography as well.

Methodology

We can utilise a lot of standard techniques and schemes to solveour problem, namely: classic Runge-Kutta methods, the MacCormack and Lax-Wendroff,

LeapFrog method, Lax–Friedrichs method. [5] From the above-mentioned methods only 2 explicit finite difference methods are more accurate in space and time for the shallow water equations and applicable in our project are: 1) MacCormack method and 2) Lax-Wendroff method [5] Here the MacCormack method is more straightforward in the applications. However, the main drawback is that local error control might cause an issue. It also does not take care of diffusive errors in the output [2].

Before moving further to implementing it, let us take a brief look at the method. Lax-Wendroff is based on the Runge-Kutta 2 method [1].

$$x_m = x_0 + \frac{1}{2}\triangle T f(x_0) \tag{3}$$

$$x_1 = x_0 + \triangle T f(x_m) \tag{4}$$

As we can see a step is divided in two "half steps" here, by doing so and taking derivative there, this method produces results which are second order accurate in time. Firstly we reduce the notation by representing all the arguments of t derivative with U matrix, with x derivative as $F(U)$ matrix and with y derivative as $G(U)$ matrix, so the equations become [1]

$$\frac{\partial U}{\partial t} + \frac{\partial F(U)}{\partial x} + \frac{\partial G(U)}{\partial y} = 0 \tag{5}$$

This equation is apparently very common equation regarding flows, called Hyperbolic Conservation Law. Then, we create a finite square difference grid with a vector-valued solution centered in the grid cells. And then we define a quantity, $U^n_{i,j}$, which is a three component vector at each grid cell i,j that evolves with time.

Now, in this specific method , the numerical solution is calculated in two steps per cycle of solution. The fist "half step" defines the quantity at midpoint of edges of the grid, and the second "half step" uses the values calculated above to define the center of the squares of the grid. Mathematically, the first half step is:

$$U^{n+1/2}_{i+1/2,j} = \frac{1}{2}(U^n_{i+1,j} + U^n_{i,j}) - \frac{\Delta t}{2\Delta x}(F^n_{i+1,j} + F^n_{i,j}) \tag{6}$$

$$U^{n+1/2}_{i,j+1/2} = \frac{1}{2}(U^n_{i,j+1} + U^n_{i,j}) - \frac{\Delta t}{2\Delta y}(G^n_{i+1,j} + G^n_{i,j}) \tag{7}$$

and the second half step is

$$U_{i,j}^{n+1} = U_{i,j}^{n} - \frac{\Delta t}{\Delta x}(F_{i+1/2,j}^{n+1/2} - F_{i-1/2,j}^{n+1/2}) - \frac{\Delta t}{\Delta y}(G_{i+1/2,j}^{n+1/2} - G_{i-1/2,j}^{n+1/2}) \quad (8)$$

The time limit is 25 seconds, and with an increment of 0.01 seconds.The loop is run over the increment in time, and solved over spatial increments. The droplet size was taken as height = 1.5 and width = 21 units. The drop's shape is taken to be a Gaussian. By changing the height and width of the drop dimension, we can generate different sizes of the drops. Moreover, every 0.01 seconds, the drop size varies during interactions with the water surface. Also, we can control the speed of the dropping the drop and subsequent plots. The position of the water drop is defined in the code as i and j, which are the x and y coordinates of the drop's position, respectively.

Results

The model solved with the method mentioned gave us these results. We used the 3D plotting function for the 3D plot, where wave propagation happens in x and y directions and the height of the wave changes. The numerical results are displayed in Figure 1,2,3,4 and 5 of various simulations.

1D Model

we started by solving the model for a 1D case. The results of the simulation are displayed below

2D Model

The timestamps were chosen as such to match with (Lundgren's results) [5], so that we would be able to compare our results.

Multiple Drops

We also tried to drop 2 drops simultaneously, but chose the position of dropping as random to cover maximum scenarios and observe anything interesting.

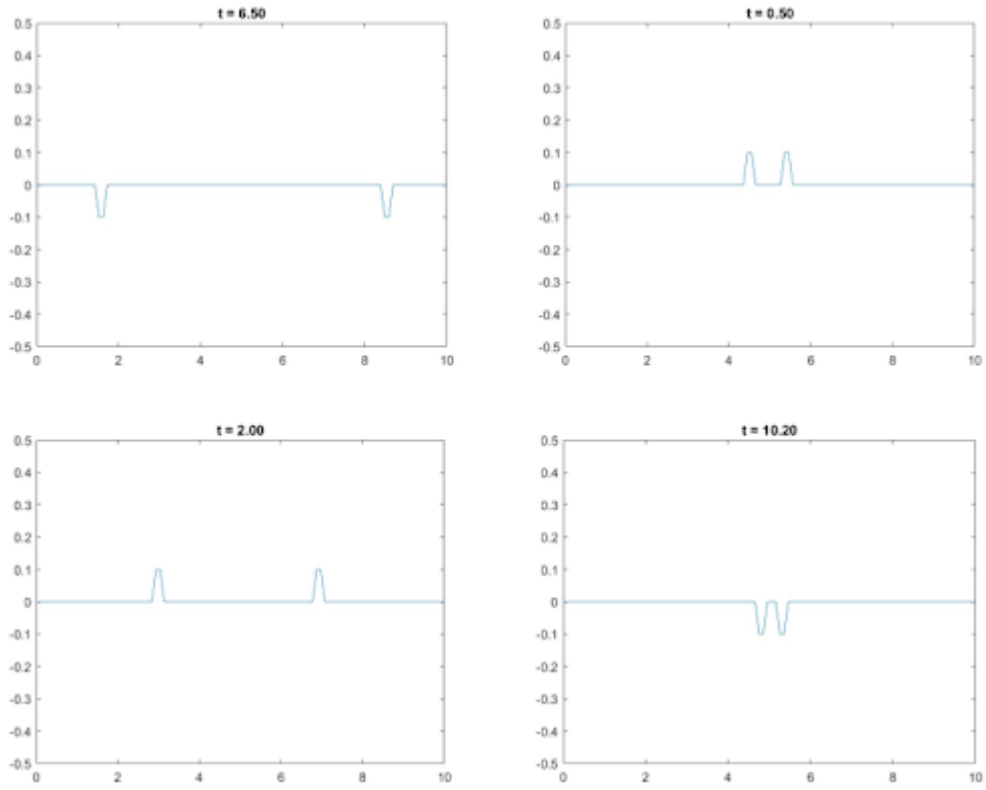

Figure 1: Numerical solution of Shallow Water Wave equation at different time stamps in 1D

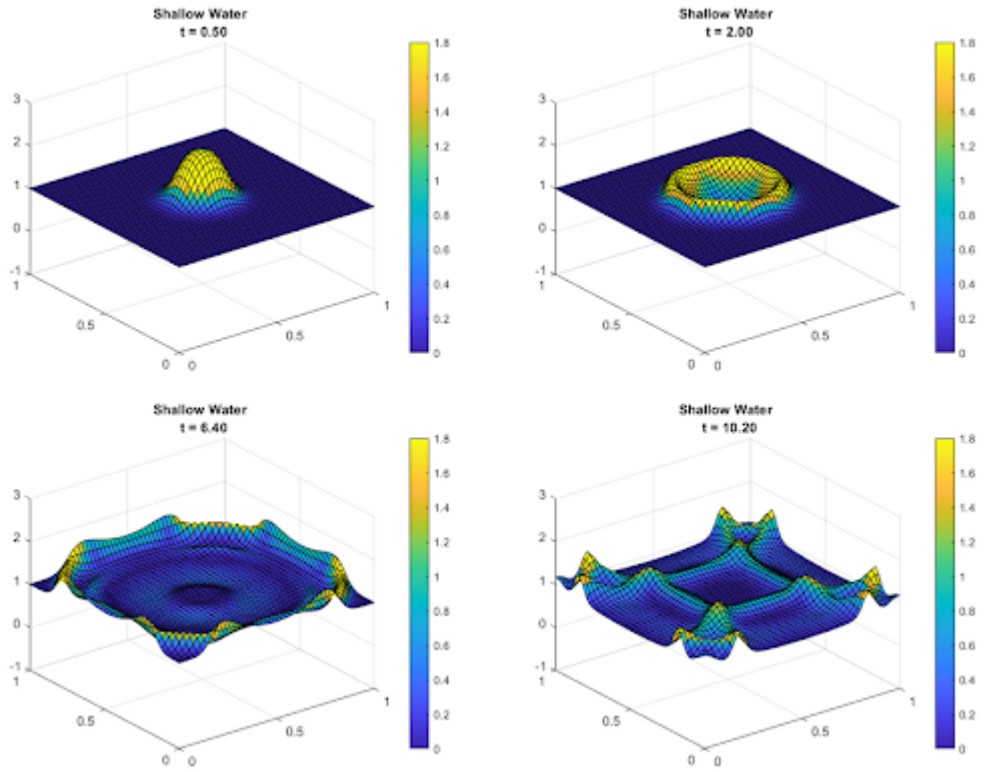

Figure 2: Numerical solution of Shallow Water Wave equation at different time stamps in 2D

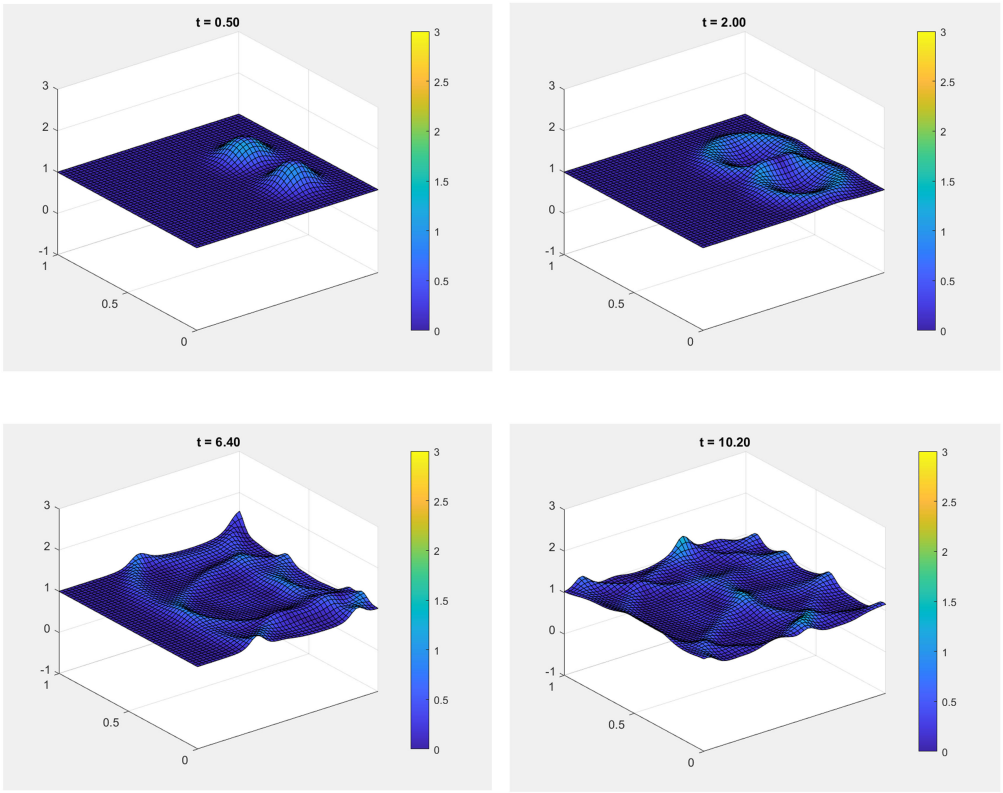

Figure 3: Numerical solution of Shallow Water Wave equation at different time stamps in 2D with 2 drops

With Barrier

We tried to incorporate a barrier in our grid to mimic a flood wall. We were unable to create a line barrier, so we created a point barrier as a start. The point, to act like a barrier, has reflecting boundary condition, and hence reflects any waveform that comes at the point. We can clearly see the barrier

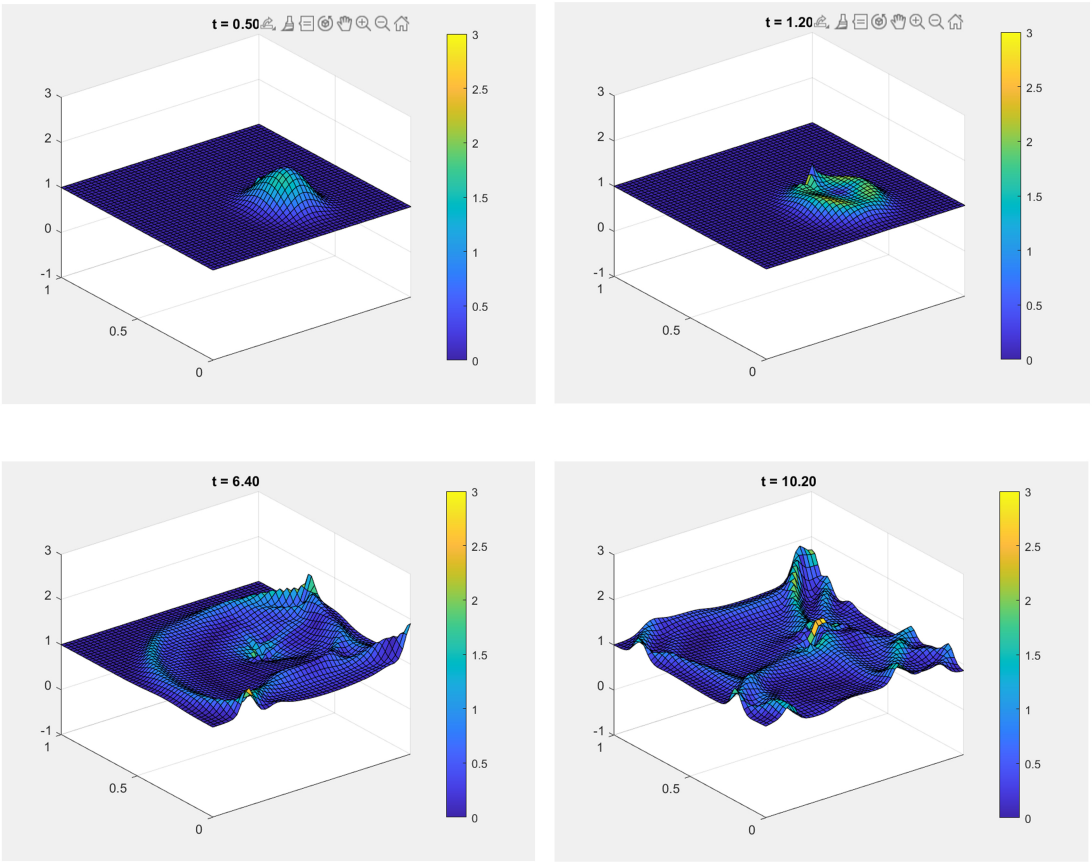

Figure 4: Numerical solution of Shallow Water Wave equation at different time stamps in 2D with 2 drops with a point barrier

in the plot. The yellow patch shows the immediate reflecting condition and hence propagation can be seen impacted due to that.

Verification

We are currently using the reference [5] for verification of our results of solution of wave equation. In the reference the, author uses different numerical solution method than ours, ie Runge Kutta methods instead of Lax–Wendroff, which gives us an opportunity to compare same model for different methods. The 2D wave equation solution from our model with reflective boundary conditions came out to be Results in our refence solutions were One ma-

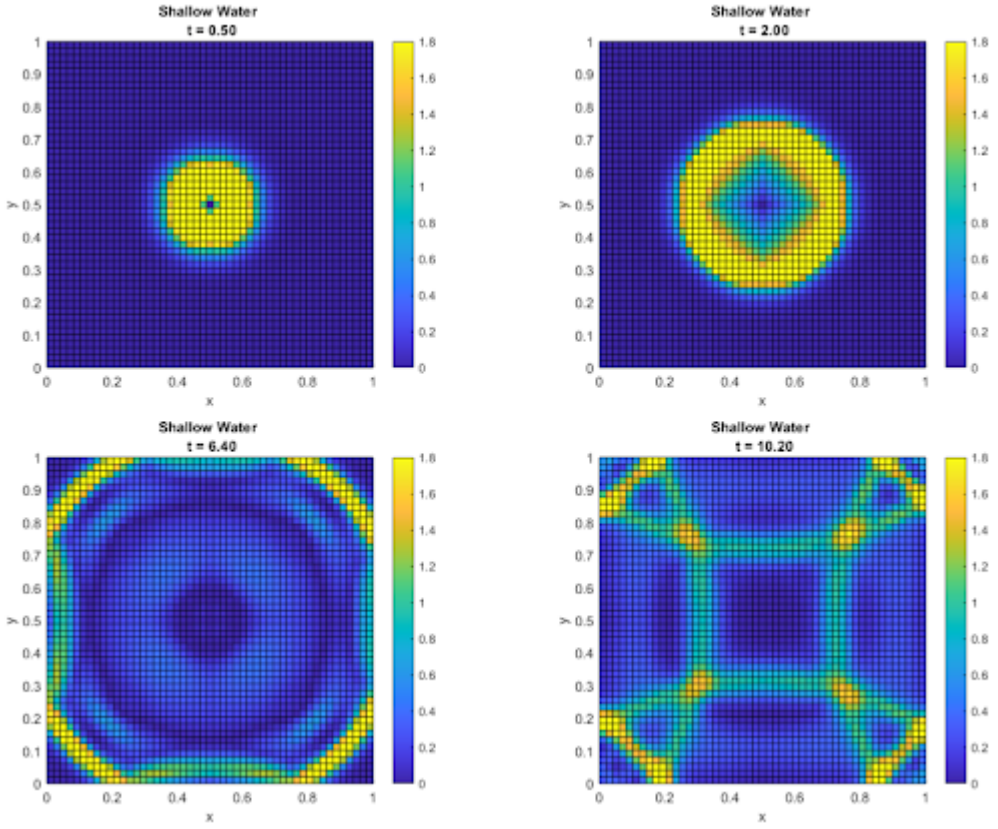

Figure 5: Numerical solution of 2D Shallow Water Wave equation at different time stamps

jor difference in approach was that we used the Lax-Wendroff method while

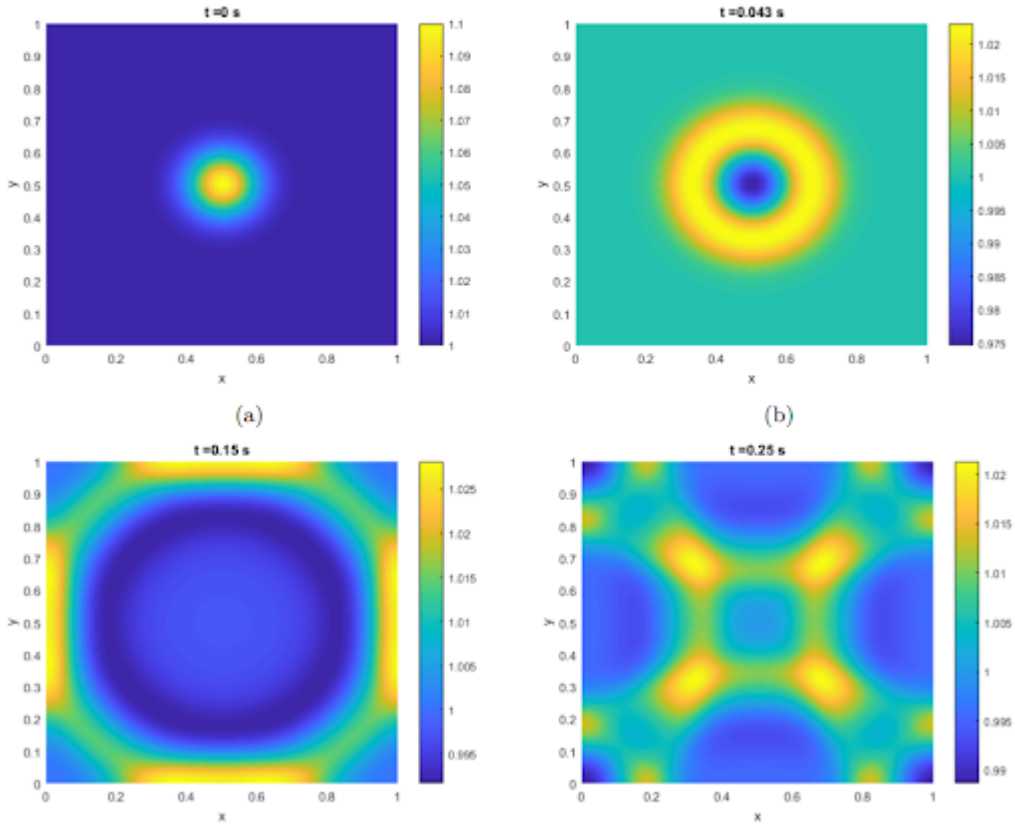

Figure 6: Numerical solution of 2d Shallow Water Wave equation at different time stamps [5]

Lundgren used the 4th order Runge-Kutta method [5]. The time taken was different values due to various conditions which include the dimensions of the drop, the height where the drop is dropped from, and defining the change in time. So we compared our result with our defined time which outputted similar results for each case. Also, the height of the wave/the value for the color bar was different due to the difference in drop conditions, initialization, and some other factors. For the case except for the third case, the contour plot follows the same pattern as the reference results. The third case looks a little different than the reference, it could come from using a different method, the way to define the boundary conditions, drop conditions, and some other factors. Overall, our results seem trustworthy compared to reference results, so we can use this code as a starting point for future work.

Accuracy and Errors

Next thing we wanted to look at was the order of accuracy and error of the model we used. Since we only used one solution scheme and due to the time constraints, it was hard to compare the solutions to a true value and other methods. The error and order of accuracy, although, for the model has been discussed a lot in contemporary literature. A study conducted by Changna Lu et al [4], compared the accuracy and order analysis of the Lax Wendroff Scheme with a RK3 method for a number of 1D and 2D shallow water models with flat bottom topography. This study closely resembles our own, and the results can be extrapolated within reasonable confidence.

The study used A Lax-Wendroff-type procedure with the high order finite volume simple weighted essentially nonoscillatory (SWENO) Lax Wendroff scheme and compared it with a WENO RK3. The results were as follows for both 1D and 2D case. The study concluded that the order of accuracy of the both methods was on the order of 5 and Lax Wendroff scheme performing better than the RK3 method, in both the accuracy of results and lesser computation time.

We also tried to vary the grid size to see the variation in the solution, for a given simulation under same conditions. We fixed our drop position and then measured the height variation of a position and its evolution with respect to time and solved the model for grid sizes [50, 60, 70, 80] points.

We can see the plots of different grids shifted comped to each other. This is expected, since all thing kept constant, the time taken to reach that point by the wave is a little more in larger grid sizes than smaller ones. We can

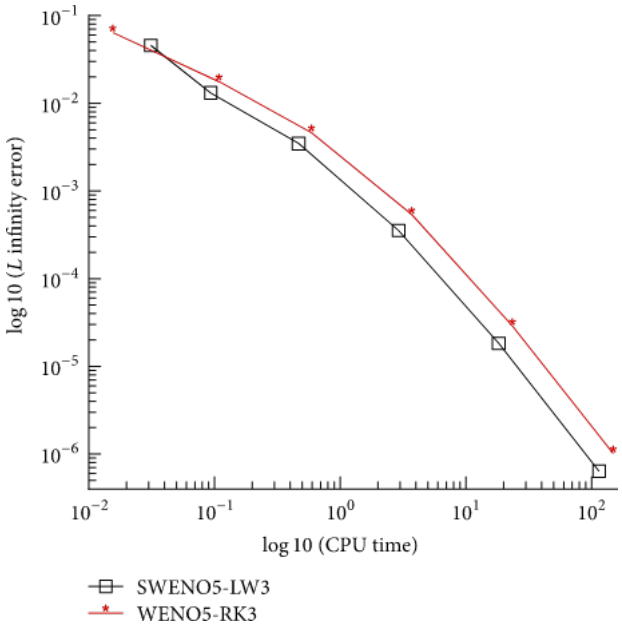

Figure 7: L1 errors and CPU time for the 1D order test [4]

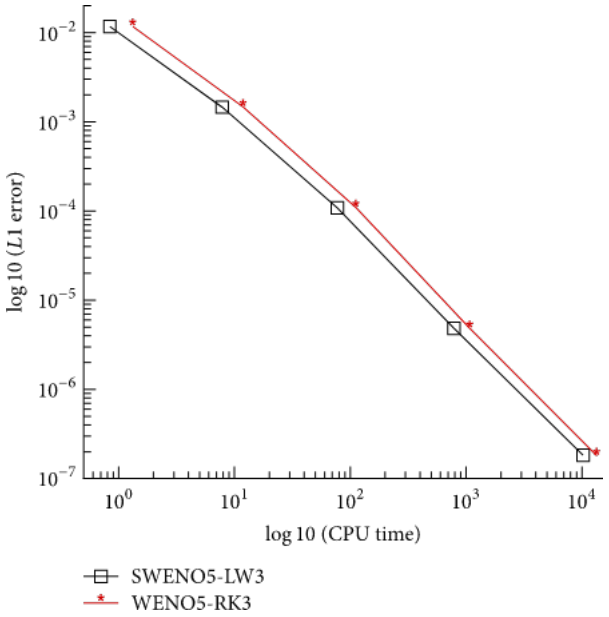

Figure 8: L1 errors and CPU time for the 2D order test [4]

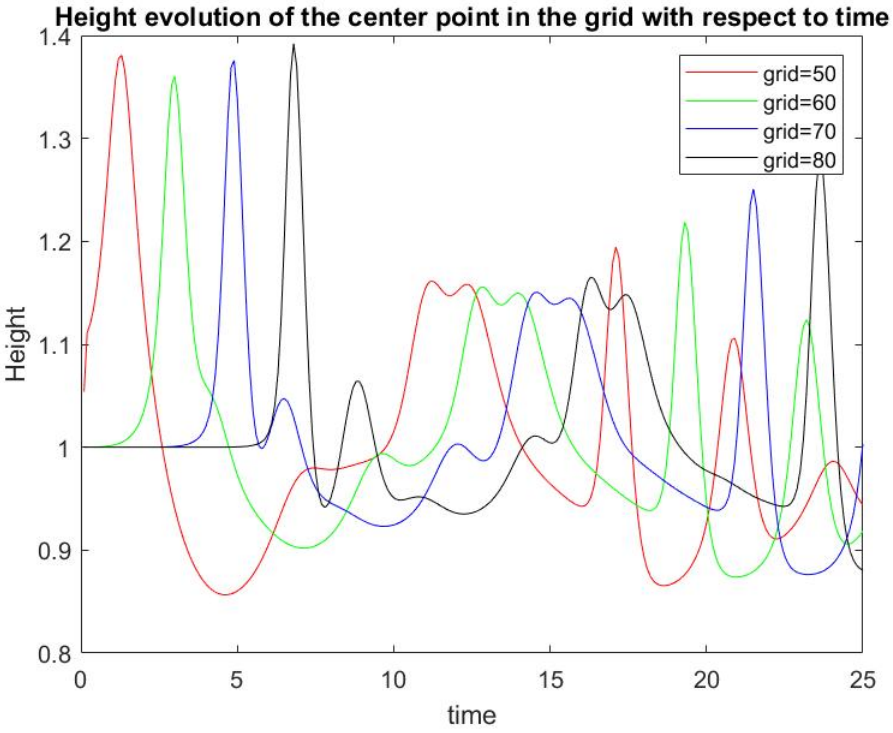

Figure 9: Height evolution of the central grid point with respect to time

with the higher grid size, we get more accurate results of simulation, albeit the trade off is more computation times.

Conclusion

The purpose of this project was to simulate the shallow water wave in 2D model as a function of time using the 2D shallow water wave equation. For simulation, we solve the 2D shallow water wave equation using the Lax-Wendorff method which is based on 2nd order Runge-Kutta method. Then we compared the simple case with the reference paper which used the 4th Runge-Kutta method to verify our result. Using the verified code as the starting point, we made the simulation with multiple drops at a random position and a simulation with a point barrier at a fixed point. As the result, we were able to compute those simulations and it can be used to predict the behavior of the water surface as a function of time.

The project offers a multitude of opportunities to extend the studies. One of the modifications that we think would be very valuable to look at would be including damping the equation and trying to solve for it, to model more realistic scenarios. Another future upgrade could be to extend the point barrier that we took to a line, which should be smaller than the grid length, thus only stopping a section of the wave. This scenario can help model flood walls.

References

[1] John Burkardt. NUMERICAL SOLUTION OF THE SHALLOW WATER EQUATIONS, March 2010.

[2] David Griffiths and D. J. Higham. MacCormack's method for advection-reaction equations. 2001.

[3] Hans Petter Langtangen and Svein Linge. *Finite Difference Computing with PDEs: A Modern Software Approach*, volume 16 of *Texts in Computational Science and Engineering*. Springer International Publishing, Cham, 2017.

[4] Changna Lu, Luoyan Xie, and Hongwei Yang. The Simple Finite Volume Lax-Wendroff Weighted Essentially Nonoscillatory Schemes for Shallow Water Equations with Bottom Topography. *Mathematical Problems in Engineering*, 2018:e2652367, February 2018.

[5] Lukas Lundgren. *Efficient numerical methods for the shallow water equations*. PhD thesis, Uppsala University, Uppsala, June 2018.

Appendix A

5/11/22 4:12 PM C:\Users...\shwetabh_project_doubledrop.m 1 of 3

```
clear;
% This program solves the shallow water equation for a singular drop
% falling on surface of water
%% defining space and other parameters
n = 50;                  % grid
g = 9.8;                 % gravitational accleration
dx = 1.0;
dy = dx;
dt = 0.01;   % you can change the speed of simulaiton by
              % changing the dt to a smaller value

T = 25;                  % time limit
d = drop(1.5,21);        % drop's dimensions

b = 0.5;

[surfplot,top] = initgraphics(n);

while 1==1 %This loop restarts the solving, after time limit is reached at t = T
%     pause(0.5)
    % defining Variable (Zeroes)
    h = ones(n+2,n+2);
    hx = zeros(n+1,n+1);
    hy = zeros(n+1,n+1);

    U = zeros(n+2,n+2);
    Ux = zeros(n+1,n+1);
    Uy = zeros(n+1,n+1);

    V = zeros(n+2,n+2);
    Vx = zeros(n+1,n+1);
    Vy = zeros(n+1,n+1);

    %% Solution Loop

    %Looping conditionals
    t = 0;
    s = 0;
    ss = 0;
    %loop
    while t<T
       t = t+dt;                % Loop counter
       ss = ss+1;
       % defining water drop position
       % This loop conditions the drop to drop only oncce per cycle of
       % solving
       if  s <2
           q1 = rand;q2 = rand;
           a = 21;                          % taken from width of the drop
           i = ceil(q1*25)+(1:a);           % x coordinate of the center of the drop
```

```
            j = ceil(q2*29)+(1:a);          % y coordinate of the center of the drop
            h(i,j) = h(i,j) + b*d;          % max height of the center of the drop, it↙
can be changed by changing value of b above
            s = s+1;
        end
        % position of the drop on the grid can be changed by changing q1 and
        % q2 between 0 & 1


        % Refleting Boundary conditions
        h(:,1) = h(:,2);      U(:,1) = U(:,2);       V(:,1) = -V(:,2);
        h(:,n+2) = h(:,n+1);  U(:,n+2) = U(:,n+1);   V(:,n+2) = -V(:,n+1);
        h(1,:) = h(2,:);      U(1,:) = -U(2,:);      V(1,:) = V(2,:);
        %barrier
        h(n+2,:) = h(n+1,:);  U(n+2,:) = -U(n+1,:);  V(n+2,:) = V(n+1,:);

        %%
        % Solving first half step
        % In x
        i = 1:n+1;
        j = 1:n;

        hx(i,j) = (h(i+1,j+1)+h(i,j+1))/2 - dt/(2*dx)*(U(i+1,j+1)-U(i,j+1));

        Ux(i,j) = (U(i+1,j+1)+U(i,j+1))/2 -  ...
                  dt/(2*dx)*((U(i+1,j+1).^2./h(i+1,j+1) + g/2*h(i+1,j+1).^2) - ...
                             (U(i,j+1).^2./h(i,j+1) + g/2*h(i,j+1).^2));

        Vx(i,j) = (V(i+1,j+1)+V(i,j+1))/2 - ...
                  dt/(2*dx)*((U(i+1,j+1).*V(i+1,j+1)./h(i+1,j+1)) - ...
                             (U(i,j+1).*V(i,j+1)./h(i,j+1)));
        %Same In y
        i = 1:n;
        j = 1:n+1;

        hy(i,j) = (h(i+1,j+1)+h(i+1,j))/2 - dt/(2*dy)*(V(i+1,j+1)-V(i+1,j));

        Uy(i,j) = (U(i+1,j+1)+U(i+1,j))/2 - ...
                  dt/(2*dy)*((V(i+1,j+1).*U(i+1,j+1)./h(i+1,j+1)) - ...
                             (V(i+1,j).*U(i+1,j)./h(i+1,j)));
        Vy(i,j) = (V(i+1,j+1)+V(i+1,j))/2 - ...
                  dt/(2*dy)*((V(i+1,j+1).^2./h(i+1,j+1) + g/2*h(i+1,j+1).^2) - ...
                             (V(i+1,j).^2./h(i+1,j) + g/2*h(i+1,j).^2));

        % Second half step
        i = 2:n+1;
        j = 2:n+1;


        h(i,j) = h(i,j) - (dt/dx)*(Ux(i,j-1)-Ux(i-1,j-1)) - ...
```

```
                        (dt/dy)*(Vy(i-1,j)-Vy(i-1,j-1));

        U(i,j) = U(i,j) - (dt/dx)*((Ux(i,j-1).^2./hx(i,j-1) + g/2*hx(i,j-1).^2) - ...
                          (Ux(i-1,j-1).^2./hx(i-1,j-1) + g/2*hx(i-1,j-1).^2)) ...
                        - (dt/dy)*((Vy(i-1,j).*Uy(i-1,j)./hy(i-1,j)) - ...
                          (Vy(i-1,j-1).*Uy(i-1,j-1)./hy(i-1,j-1)));

        V(i,j) = V(i,j) - (dt/dx)*((Ux(i,j-1).*Vx(i,j-1)./hx(i,j-1)) - ...
                          (Ux(i-1,j-1).*Vx(i-1,j-1)./hx(i-1,j-1))) ...
                        - (dt/dy)*((Vy(i-1,j).^2./hy(i-1,j) + g/2*hy(i-1,j).^2) - ...
                          (Vy(i-1,j-1).^2./hy(i-1,j-1) + g/2*hy(i-1,j-1).^2));


        % Values of variables are stored and updated
        if ss == 10
        % The value of ss can be used control the speed of dropping the drop and↙
subsequent plot

           k = abs(U(i,j)) + abs(V(i,j));
           set(surfplot,'zdata',h(i,j),'cdata',k);
           set(top,'string',sprintf('t = %.2f',t))
           drawnow
           ss=0;
        end
     end
end
%% Functions
%drop Function
function d = drop(h,w) %h - height , w - width
   [x,y] = ndgrid(-1:(2/(w-1)):1);
   d = h*exp(-5*(x.^2+y.^2)); % drop's Shape is taken to be something like a Gaussian
end
  %%
% Surf plot of the function
function [surfplot,top] = initgraphics(n)
   clf
   x = (0:n-1)/(n-1);
   surfplot = surf(x,x,ones(n,n),zeros(n,n));
   axis([0 1 0 1 -1 3])
    caxis([0 3])
   shg
   colorbar;
   top = title('Shallow Water');
end
```