



Bangladesh University of Engineering and Technology
Department of Electrical and Electronic Engineering
Dhaka-1205, Bangladesh

Numerical Technique Laboratory

EEE 212

PROJECT REPORT

Shallow Water System Simulation

Prepared By:

Name: Mushfiqur Rahman Student ID:2106181	Name: Kayes Sami Malitha Student ID:2106184
--	--

Submitted to:

Bejoy Sikder
Lecturer, Department of EEE, BUET
Mrinmoy Kundu
Lecturer, Department of EEE, BUET

Shallow Water System Simulation

In this project, we will simulate the behavior of water in a bounded area in case any disturbance is created. The inputs will be as follows:

- Coordinate of the disturbance
- The time for how long the user wants to see the output.
- Boundary conditions for the water.
- Disturbance parameters.

Objective:

We will solve a set of partial differential equations describing the model and state how the wave on the surface propagates. The output will be the 3-D plot of disturbance on the surface for the given interval of time. An animation of the simulation will be shown. The Shallow Water Equations (SWEs) are a mathematical model that describes the movement of water in shallow areas such as rivers and coastlines. They are used by scientists to gain insights into and make predictions about natural phenomena such as floods, storm surges, tsunamis and coastal erosion. The SWEs are also essential for designing structures that can withstand these events and protect against their potential damage.

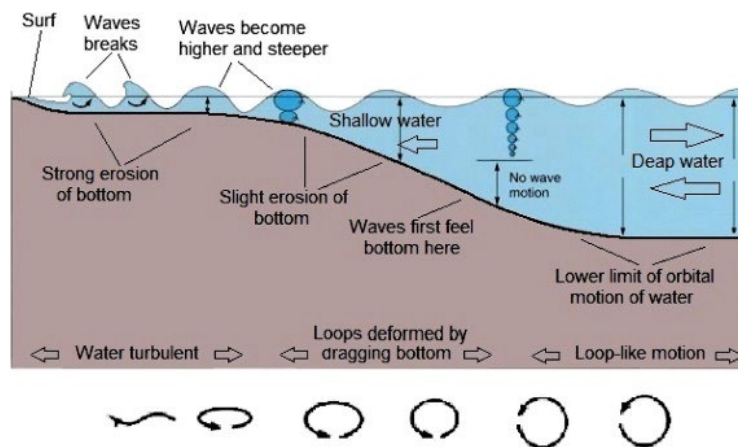


Fig. 1. Changing the motion of water particles.

Theory:

The equations that govern the flow of water in a shallow water system occurs from the Navier-Stokes equation. Navier-Stokes equation, in fluid mechanics, a partial differential equation that describes the flow of incompressible fluids. The Navier-Stokes equation, in modern notation, is

$$\nabla \cdot \mathbf{u} = 0 \quad \text{--- (i)}$$

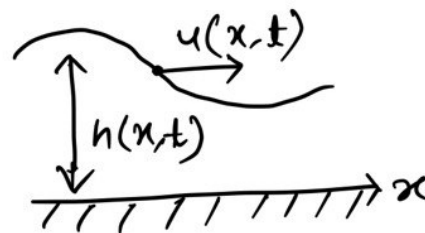
$$\rho \frac{d\mathbf{u}}{dt} = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{F} \quad \text{--- (ii)}$$

\mathbf{u} is the fluid velocity vector, P is the fluid pressure, ρ is the fluid density, ν is the where kinematic viscosity, and ∇^2 is the Laplacian operator and \mathbf{F} is any external force like gravity.

The equations governing the behaviour of shallow water systems can be found by doing a depth integration of the Navier-Stokes equation. Derivation of shallow water equations from Navier-Stokes equation is shown below:

shallow water assumption $\frac{h}{L} \ll 1$. Derivation for 1D eqns considering y direction is homogenous

$$\frac{\partial}{\partial y} (\text{Any quantity}) = 0$$



□ continuity equation for an incompressible flow : $\nabla \cdot \vec{u} = 0$

If we integrate this eqn from $z=0$ to h w.r.t dz .

$$\int_0^h \nabla \cdot \vec{u} \, dz = 0$$

$$\Rightarrow \int_0^h \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) dz = 0$$

Here, $\frac{\partial v}{\partial y} = 0$

$$\Rightarrow \int_0^h \frac{\partial u}{\partial x} \, dz + \omega_h - \omega_0 = 0$$

In this case, there is no slip condition

$$\omega_0 = 0$$

Applying the Leibniz integral rule,

$$\frac{\partial}{\partial z} \int_a^b f(x, z) dx = \int_a^b \frac{\partial f}{\partial z} dx + f(b, z) \frac{\partial b}{\partial z} - f(a, z) \frac{\partial a}{\partial z}$$

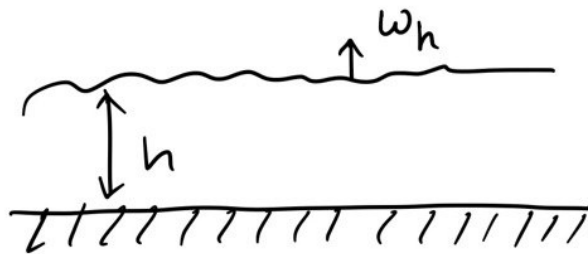
$$\text{So, } \int_0^h \frac{\partial u}{\partial x} dz + w_h = 0$$

$$\Rightarrow \frac{\partial}{\partial x} \int_0^h u dz - u|_{z=h} \frac{\partial h}{\partial x} + u|_{z=0} \frac{\partial 0}{\partial x} + w_h = 0$$

$$\text{Depth average velocity, } \bar{u} = \frac{1}{h} \int_0^h u dz$$

$$\text{So, } \frac{\partial(\bar{u}h)}{\partial x} - u_h \frac{\partial h}{\partial x} + w_h = 0 \quad \text{--- (1)}$$

Apply free surface of water, $z=h$, there is no relative normal flow,



$$\text{At } z = h, \frac{dh}{dt} = \omega_h \Rightarrow \frac{\partial h}{\partial t} + u_h \frac{\partial h}{\partial x} + \cancel{v_h \frac{\partial h}{\partial y}}^0 = \omega_h$$

here, h is a function of (x, t)

$$\text{so, } \omega_h = \frac{\partial h}{\partial t} + u_h \frac{\partial h}{\partial x} \quad \text{--- (11)}$$

$$\text{substituting in eqn (i)} \rightarrow \frac{\partial(\bar{u}h)}{\partial x} + \frac{\partial h}{\partial t} = 0$$

In 2-D water surface, the $\frac{\partial(\bar{v}h)}{\partial y}$ term will add.

$$\text{continuity eqn: } \frac{\partial h}{\partial t} + \frac{\partial(h\bar{u})}{\partial x} + \frac{\partial(h\bar{v})}{\partial y} = 0$$

For the momentum equation, we will consider the shallow water system in 1-D form again.

We know from Euler equation:

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + \omega \frac{\partial u}{\partial z}$$

Navier Stokes equation on the x direction,

$$\frac{\partial u}{\partial t} = -\frac{1}{\rho} \frac{\partial p}{\partial x} \quad [\text{gravity working only on the } z \text{ direction \& viscous forces are ignored}]$$

We know from the 1st Navier Stokes equation,

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \quad \text{multiplied by } u,$$

$$\Rightarrow u \frac{\partial u}{\partial x} + u \frac{\partial v}{\partial y} + u \frac{\partial w}{\partial z} = 0 \quad \text{--- (ii)}$$

The momentum equation in x direction :

$$\frac{\partial u}{\partial t} = -\frac{1}{\rho} \frac{\partial p}{\partial x}$$

$$\Rightarrow \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} = -\frac{1}{\rho} \frac{\partial p}{\partial x};$$

Adding eqn(ii)

$$\Rightarrow \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} + u \frac{\partial u}{\partial x} + u \frac{\partial v}{\partial y} + u \frac{\partial w}{\partial z} = -\frac{1}{\rho} \frac{\partial p}{\partial x} + 0$$

$$\Rightarrow \frac{\partial u}{\partial t} + 2u \frac{\partial u}{\partial x} + \left(v \frac{\partial u}{\partial y} + u \frac{\partial v}{\partial y} \right) +$$

$$\left(w \frac{\partial u}{\partial z} + u \frac{\partial w}{\partial z} \right) = - \frac{1}{\rho} \frac{\partial p}{\partial x}$$

$$\Rightarrow \frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial}{\partial y} (uv) + \frac{\partial}{\partial z} (uw) = - \frac{1}{\rho} \frac{\partial p}{\partial x}$$

However, we considered homogeneity in y-direction,

$$\frac{\partial}{\partial y} (uv) = 0$$

$$\text{So, } \frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial}{\partial z} (uw) = - \frac{1}{\rho} \frac{\partial p}{\partial x}$$

Integrating this from $z=0$ to h w.r.t dz ,

$$\int_0^h \left[\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} \right] dz + uw \Big|_0^h = - \frac{1}{\rho} \int_0^h \frac{\partial p}{\partial x} dz$$

— (v)

$\therefore u_h w_h$ as velocity at the ground is 0.

If we again use the Leibniz rule for integration,

$$\int_0^h \frac{\partial u}{\partial t} dz + u_n \frac{\partial h}{\partial t} - u_0 \frac{\partial h}{\partial t} = \frac{\partial}{\partial t} \int_0^h u dz$$

$$\text{or, } \int_0^h \frac{\partial u}{\partial t} dz = \frac{\partial}{\partial t} \int_0^h u dz - u_n \frac{\partial h}{\partial t}$$

For the 2nd term,

$$\int_0^h \frac{\partial u^r}{\partial x} dz = \frac{\partial}{\partial x} \int_0^h u^r dz - u_n^r \frac{\partial h}{\partial x}$$

putting these values in eqn (v),

$$\left[\left(\frac{\partial}{\partial t} \int_0^h u dz - u_n \frac{\partial h}{\partial t} \right) + \left(\frac{\partial}{\partial x} \int_0^h u^r dz - u_n^r \frac{\partial h}{\partial x} \right) \right] + w_n u_n = - \frac{1}{\rho} \int_0^h \frac{\partial p}{\partial x} dz$$

$$\Rightarrow \frac{\partial}{\partial t} (\bar{u}h) - u_n \left(\frac{\partial h}{\partial t} + u_n \frac{\partial h}{\partial x} - w_n \right) + \frac{\partial}{\partial x} \int_0^h u^r dz = - \frac{1}{\rho} \int_0^h \frac{\partial p}{\partial x} dz$$

$$\left[\text{because } \frac{\partial h}{\partial t} = w_n \Rightarrow \frac{\partial h}{\partial t} + u_n \frac{\partial h}{\partial x} + \cancel{v_n \frac{\partial h}{\partial y}} = w_n \right]$$

velocity shape factor,

$$E = \frac{1}{h\bar{u}^2} \int_0^h u^2 dz, \text{ when } u \text{ is independent of } z$$

then $E=1$ which means $\int_0^h u^2 dz = h\bar{u}^2$;

$$\text{when } \frac{|u - \bar{u}|}{\bar{u}} \leq 20\%$$

then,

$$\frac{\partial}{\partial t}(h\bar{u}) + \frac{\partial}{\partial x}(h\bar{u}^2) = -\frac{1}{\rho} \int_0^h \frac{\partial p}{\partial x} dz$$

Now,

$$\int_0^h \frac{\partial p}{\partial x} dz = \frac{\partial}{\partial x} \int_0^h p dz - \cancel{p(h) \frac{\partial h}{\partial x}} + p(0) \frac{\partial z}{\partial x} \Big|_0^0$$

due to atmospheric pressure

E will change only about 4%.
that why $E \approx 1$ is a pretty good approximation in shallow water systems.

$$\Rightarrow \int_0^h \frac{\partial p}{\partial x} dz = \frac{\partial}{\partial x} \int_0^h (\rho g z) dz$$

- 0 to 0

$$\Rightarrow \int_0^h \frac{\partial p}{\partial x} dz = \frac{\partial}{\partial x} \frac{1}{2} \rho g h^2$$

$$\Rightarrow -\frac{1}{\rho} \int_0^h \frac{\partial p}{\partial x} = -\frac{1}{2} g \frac{\partial}{\partial x} h^2$$

So the final equation is:

$$\frac{\partial}{\partial t} h \bar{u} + \frac{\partial}{\partial x} (h \bar{u}^2 + \frac{1}{2} g h^3) = 0$$

When consider 2D case, a term $\frac{\partial}{\partial y} (h u v)$ occurs at the equation in the L.H.S, because we assumed, $\frac{\partial}{\partial y} (u v) = 0$.

When this term is integrated using Leibnitz formula $\frac{\partial}{\partial y} (h u v)$ occurs.

The momentum equation in x direction

$$\frac{\partial}{\partial t} (h \bar{u}) + \frac{\partial}{\partial x} (h \bar{u}^2 + \frac{1}{2} g h^3) + \frac{\partial}{\partial y} (h u v) = 0$$

The momentum equation in y-direction,

$$\frac{\partial}{\partial t} (h \bar{v}) + \frac{\partial}{\partial y} (h \bar{v}^2 + \frac{1}{2} g h^3) + \frac{\partial}{\partial x} (h u v) = 0$$

So, the shallow water equations are:

$$\frac{\partial h}{\partial t} + \frac{\partial hu}{\partial x} + \frac{\partial hv}{\partial y} = 0$$

$$\frac{\partial hu}{\partial t} + \frac{\partial}{\partial x} \left(hu^2 + \frac{1}{2}gh^2 \right) + \frac{\partial}{\partial y} (huv) = 0$$

$$\frac{\partial hv}{\partial t} + \frac{\partial}{\partial y} \left(hv^2 + \frac{1}{2}gh^2 \right) + \frac{\partial}{\partial x} (huv) = 0$$

Assumptions:

The following assumptions were made while solving the project and deriving the shallow water equations to avoid complexities:

- i) Water is bounded by a square boundary, and the ground is also a square plane. The part of the water in observation of the simulation is bounded by a square. The boundary walls are thought to be reflective. No energy dissipation is considered when the wave hits the boundary wall. The ground at which the water stands is a plane. If the ground deformities were taken as an input parameter, it would make the whole simulation much more complex. The height of the water surface is always constant from the ground.
- ii) Shallow water condition: Depth \ll Length and Width. This is more of a definition of shallow water than an assumption. Shallow water technically means that the surface area of the water body, the length, and the width will be much greater than the depth.
- iii) Water is an incompressible fluid, meaning that the density of the fluid within a fluid parcel, an infinitesimal volume that moves with the flow velocity, is constant. This assumption is directly connected to the Navier-Stokes equation as the first equation is a direct consequence of such assumption.

iv) Water is isotropic. The features of water are the same in all directions. This assumption avoids any discrepancy that may arise if the disturbance is not exactly in the middle.

v) The velocity of wave propagation in water is constant and the same in both the x and y directions. The velocity of a wave in any medium is dependent on the medium material. Since we are focusing our observation only on water systems, and the temperature of water is constant, the speed of wave propagation is also constant. As water is isotropic, the velocity of wave propagation is the same in both x and y directions.

vii) Viscous force is constant. The viscosity of a fluid is a measure of its resistance to deformation at a given rate. The viscous force, $F = \mu Au/h$. Here u is the wave velocity, h is the height from the ground, A is the surface area and μ is the viscosity of the fluid at a given temperature. All these factors are considered constant in previous assumptions. So, the viscous force is also considered constant.

viii) The process is isothermal. As the density of a fluid and viscosity parameters are directly dependent on temperature, we have treated the simulation as an isothermal process to keep such parameters constant.

Methodology:

1. Square Grid generation:

The simulation is performed on a square boundary that is divided into a grid of smaller squares, each with a length of 1 meter. The grid is created using the `meshgrid()` function and each intersection is part of an array. To create this array, `linspace` is used. The choice of a square grid makes the computation easy and helps to solve the PDE using the finite difference method. The matlab code-

```

% Define parameters
L = input("Length of domain: "); % Length of domain1
W = L; % Width of domain
Nx = L+1; % Number of grid points in x direction
Ny = Nx; % Number of grid points in y direction
dx = L / (Nx - 1); % Grid spacing in x direction
dy = W / (Ny - 1); % Grid spacing in y direction
g = 9.81; % Gravity acceleration
[X Y] = meshgrid(linspace(0,L,Nx),linspace(0,W,Ny));

```

The length of the grid is determined by Lx, which is an input parameter in the command prompt. The number of intersections in the grid is represented by Nx and Ny in the x and y direction respectively. The distance between each intersection in the x and y direction is denoted by dx and dy respectively. As the index in the grid matrix can hold integer values, dx and dy are set to 1. The [x y] grid is created using the meshgrid() and linspace() function, which stores information about the water height at each coordinate of the grid.

2. Initial Conditions of disturbance:

The following are the input parameters for the disturbance:

Disturbance coordinates: (xo , yo)

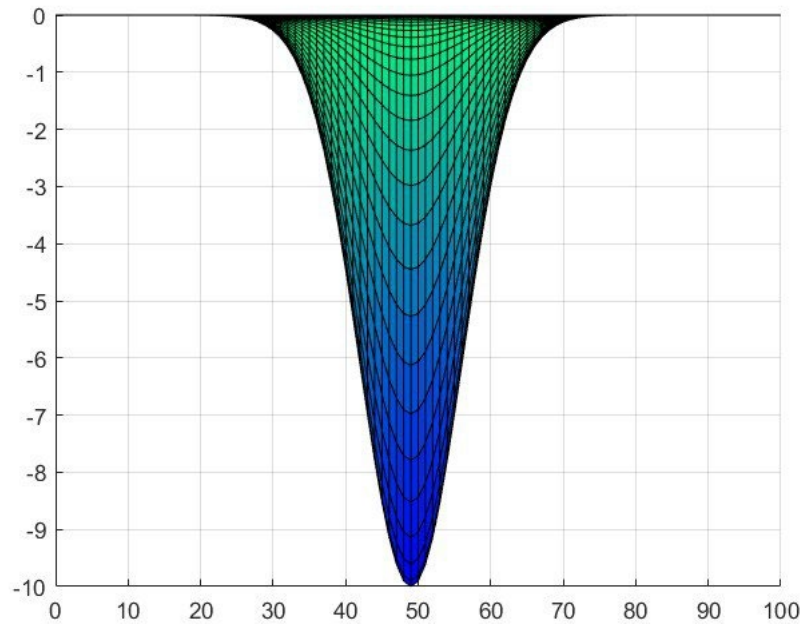
Radius of disturbance: R

and Intensity of disturbance: intensity.

The disturbance is represented by a negative 3D bell curve. The curve's maximum amplitude is the defined height of the water surface from the ground. You can adjust the amplitude by changing its intensity. The height on the 3D plot is fixed from -H to H m. At 100% disturbance intensity, the peak of the disturbance reaches -H height. The formula for the disturbance looks like:

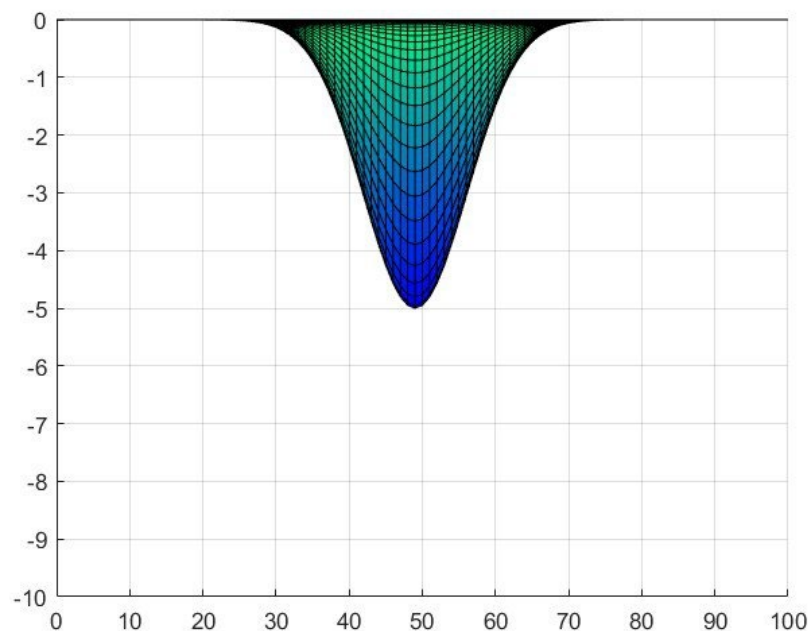
$$h(i,j) = -amp * \exp\left[-\frac{\{(i-x_0)^2 + (j-y_0)^2\}}{\left(\frac{R}{2}\right)^2}\right]$$

Here $h(i,j)$ is the grid height at i,j coordinate. 'amp' is the amplitude, R is the radius of the disturbance and x_0, y_0 are the coordinates where the disturbance is at its peak. The height of the water radially reaches zero from the source point (x_0, y_0) . If height is 10 and intensity is 100%, the initial state of the shallow water system looks like this:



The figure shows a 2D view with width in the x-axis and height in the y-axis.

For height 10 and 50% intensity:



Matlab code:

```
x_co = input("x coordinate of disturbance: "); % x coordinate of
disturbance
while (x_co>L)
    disp("Invalid x coordinate");
    x_co = input("x coordinate of disturbance: ");
end

y_co = input("y coordinate of disturbance: "); % y coordinate of
disturbance
while (y_co>L)
    disp("Invalid y coordinate");
    x_co = input("y coordinate of disturbance: ");
end

R = input("Disturbance radius: "); % disturbance size
while ( (x_co-R)<0 || (y_co-R)<0 )
    disp("Radius exceeds the coordinate");
    R = input("Disturbance radius: ");
end
%Intensity conditions
intensity = input("Disturbance intensity (1-100%) : ");%should be from
1 to 100
while (intensity>100 || intensity<1)
    disp("Invalid intensity,select between 1-100");
    intensity = input("Disturbance intensity (1-100%) : ");
end

damp = input("Damping coefficient (0-10) : ");
while (damp>10 || damp<0)
    disp("Invalid damping coefficient,select between 0-10");
    damp = input("Damping coefficient (0-10) : ");
end
b = damp/(dx*dy*1*1000); % damping factor
amp=H*intensity/100;
T = input("Enter simulation time : "); % Total simulation time
dt = 0.22; % Time step

% Initialize variables
for i=1:Nx
    for j=1:Ny
        h(i,j)=-amp*exp((-((i-x_co)^2 + (j-y_co)^2))/((R/2)^2));
    end
end
end
```


Here the amp is the amplitude of the disturbance which is a product of the height and the intensity of the disturbance.

3. Solving the PDE with the central difference method:

From the assumptions that we made while solving the shallow water system equations turn into one equation when we consider other factors as constants. We solved the partial differential equation using the finite difference method that looks like:

$$\frac{\partial h}{\partial t} + \frac{\partial hu}{\partial x} + \frac{\partial hv}{\partial y} = 0$$
$$\partial h = \partial t * \left(\frac{\partial hu}{\partial x} + \frac{\partial hv}{\partial y} \right)$$
$$h = h + \partial h$$

Here $h(i,j)$ is the grid array that holds the height of the water surface. In our code, $dx=dy=1$ as we can only jump to integer matrix index.

Matlab code:

```
% Calculate dh/dx and dh/dy using forward finite differences
dh_dx = zeros(Nx, Ny);
dh_dy = zeros(Nx, Ny);
for i = 1:Nx-1
    for j = 1:Ny
        dh_dx(i,j) = (h(i+1,j) - h(i,j)) / dx;
    end
end
for i = 1:Nx
    for j = 1:Ny-1
        dh_dy(i,j) = (h(i,j+1) - h(i,j)) / dy;
    end
end
```

```

% Calculate du/dt and dv/dt using shallow water equations
du_dt = -g * dh_dx;
dv_dt = -g * dh_dy;
% Update velocities
u = u + dt * du_dt;
v = v + dt * dv_dt;

% Update water height using continuity equation
dh_dt = zeros(Nx, Ny);
for i = 2:Nx-1
    for j = 2:Ny-1
dh_dt(i,j) = -((u(i,j) - u(i-1,j)) / dx + (v(i,j) - v(i,j1)) / dy);
    end
end
h = h + dt * dh_dt;

```

We are interested in determining the height of the water surface at each point in the future, given the current and past height conditions. Once we have this information, we can use the `surf()` function to plot the height over time concerning the grid.

4. Boundary condition:

We have considered two boundary cases for this system, which are as follows:

1. Free Boundary: This particular type of boundary does not apply any stress on the water, which results in equal height of water at the boundary and near the boundary. As a result, constructive interference takes place at the boundary.
2. Reflective Boundary: This type of boundary acts like a mirror. At this boundary, the height of water is the negative of the height of water near the boundary, which causes destructive interference.

Code for considered boundary condition:

```

% boundary Conditions for free boundary
if bound_cond == 1
    h(:,1)=h(:,2);

```

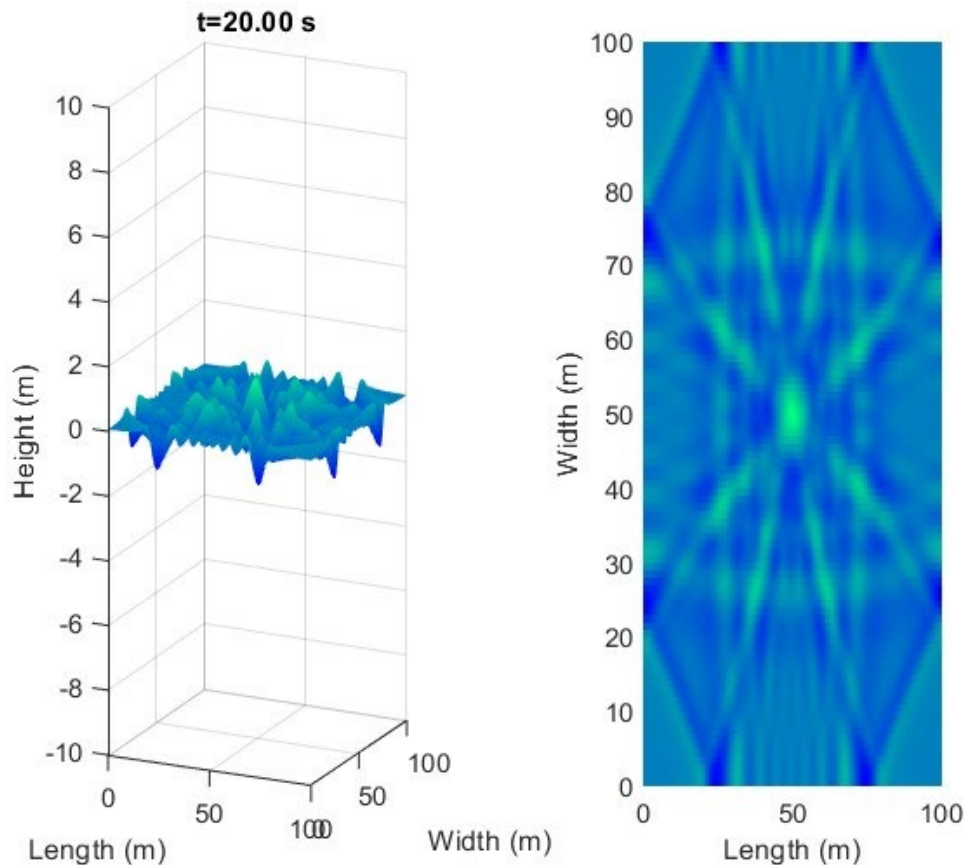
```

h(1,:)=h(2,:);
h(Nx,:)=h(Nx-1,:);
h(:,Ny)=h(:,Ny-1);
else % boundary Conditions for reflective boundary
h(:,1)=-h(:,2);
h(1,:)=h(2,:);
h(Nx,:)=h(Nx-1,:);
h(:,Ny)=-h(:,Ny-1);
end

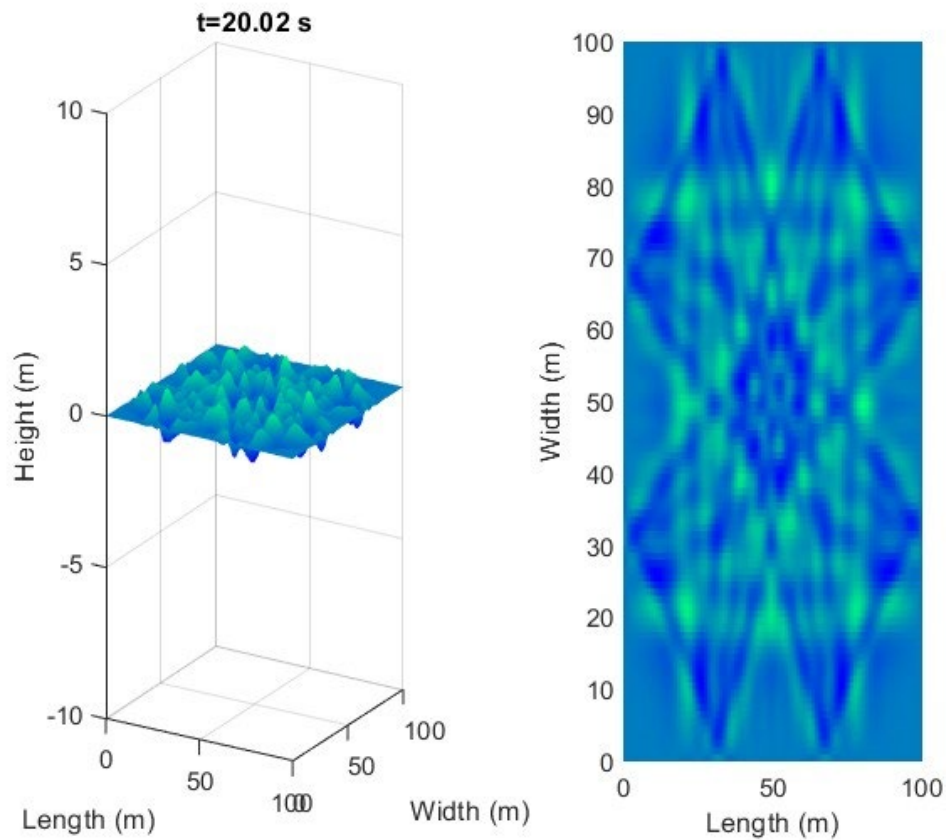
```

The boundary conditions play a vital role in water system simulation. Here is an example of simulation in case of a free and reflective boundary when all parameters are kept the same.

For free boundary the simulation looks like:



For Reflective Boundary:



Here it is clear that the simulation is different for different boundary conditions.

5. Damping Conditions:

During the propagation of disturbance, some environmental factors resist the flow and contribute to the dissipation of energy. Due to such dissipation, damping of the propagating wave will occur. We have taken an input parameter as the Damping coefficient. Three damping cases can be seen in a wave. They are overdamping, critical damping and underdamping. We have used the theory of underdamping to apply an exponential decay with time as this type of damping is mostly seen in reality.

As the mass of the water body is considered constant, the damping coefficient works on the height as,

$$h(t) = h(t) e^{-bt/2}$$

Here b is the damping constant/mass of a water unit($dx*dy*dz*1000$), t is time, the $h(t)$ in the RHS is the height of the water surface calculated by applying the finite difference method without any damping and the $h(t)$ is the damped height.

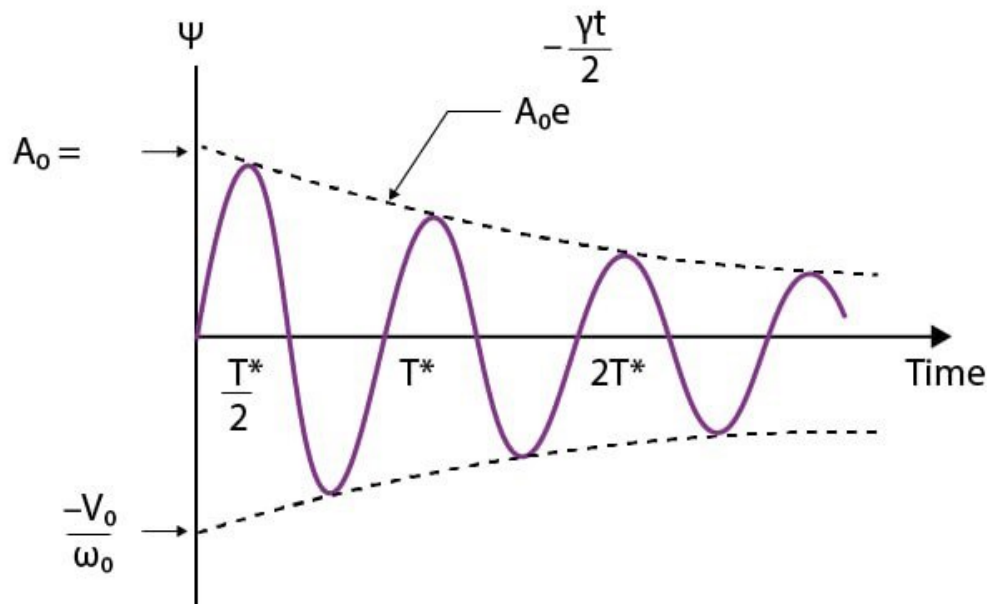


Fig: Damping conditions

Matlab code:

`%Underdamping Decay Condition if damping is given`

`h = h.*exp(-toc*b/2) ;`

`n = max(abs(h),[], "all");`

`if (n < 1e-5) % checking if the water surface is tranquil`

`h(:, :) = 0;`

`disp(toc);`

`disp("Simulation ended,water surface is tranquil now.");`

`break`

`end`

6. Simulation Output:

We displayed the simulation with surface plots using a 'flat' shading for a top view and an 'interp' shading for a 3D view. We used a winter colormap to display the height at each point in shades of blue and green. The brightest shade of green represents the maximum height, and the lowest heights are assigned deep blue shades. We limited the height of the z-axis to the given parameter to observe the decay of height with time.

Matlab Code:

```
figure(1)
subplot(1,2,1);
surf(X, Y, h, 'EdgeColor', 'none');
colormap('winter');
zlim([-H H]);
shading interp
xlabel('Length (m)');
ylabel('Width (m)');
zlabel('Height');
title(sprintf('t=%.2f s', toc));
view(45, 30);

subplot(1,2,2);
surf(X,Y,h);
view(2)
shading flat;
xlabel('Length (m)') ;
ylabel('Width (m)');
```

7. Input Errors:

Input error occurs when

- I. When the height is larger enough than the surface area, the shallow water conditions aren't maintained.
- II. The coordinate for disturbance is out of the boundary.
- III. When the radius of a disturbance is such that it surpasses the boundary, i.e. $x_0+R > L_x$ or $y_0+R > L_y$.
- IV. The intensity of disturbance is not in the range of 1-100%
- V. The damping coefficient is not in the range of 0-10%
- VI. The input for boundary conditions is not given as 1 or 2.

Inputs are taken as so that when an invalid input is encountered, the user is asked to give the correct input for that particular parameter.

Matlab code-

```
% Define parameters
L = input("Length of domain: "); % Length of domain1
W = L; % Width of domain
Nx = L+1; % Number of grid points in x direction
Ny = Nx; % Number of grid points in y direction
dx = L / (Nx - 1); % Grid spacing in x direction
dy = W / (Ny - 1); % Grid spacing in y direction1
g = 9.81; % Gravity acceleration
[X Y] = meshgrid(linspace(0,L,Nx),linspace(0,W,Ny));

H = input("Height of water surface from the ground: "); % height of water surface
from the ground
while (H/(L)>0.1)
    disp("Pani beshi govir!");
    H = input("Height of water surface from the ground: ");
end

x_co = input("x coordinate of disturbance: "); % x coordinate of disturbance
while (x_co>L || x_co<0)
    disp("Invalid x coordinate");
    x_co = input("x coordinate of disturbance: ");
end

y_co = input("y coordinate of disturbance: "); % y coordinate of disturbance
while (y_co>L || y_co<0)
    disp("Invalid y coordinate");
    y_co = input("y coordinate of disturbance: ");
end

R = input("Disturbance radius: "); % disturbance size
while ( (x_co-R)<0 || (y_co-R)<0 )
    disp("Radius exceeds the coordinate");
    R = input("Disturbance radius: ");
end

%Intensity conditions
intensity = input("Disturbance intensity (1-100%) : ");%should be from 1 to 100
while (intensity>100 || intensity<1)
    disp("Invalid intensity,select between 1-100");
    intensity = input("Disturbance intensity (1-100%) : ");
end

damp = input("Damping coefficient (0-10) : ");
while (damp>10 || damp<0)
    disp("Invalid damping coefficient,select between 0-10");
    damp = input("Damping coefficient (0-10) : ");
end

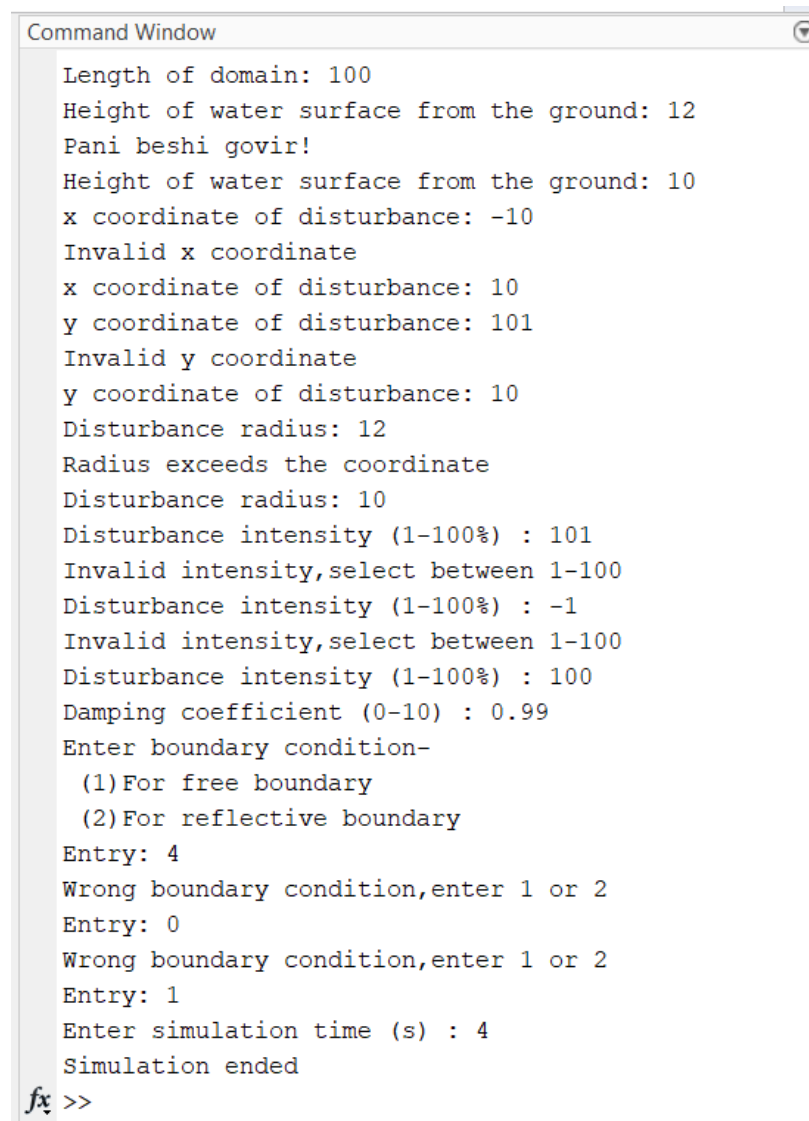
disp("Enter boundary condition-");
disp(" (1)For free boundary");
```

```

disp(" (2)For reflective boundary");
bound_cond = input("Entry: "); % take input 1 or 2
c=0;
while c==0
    if (bound_cond==2 || bound_cond==1)
        c = 1;
    else
        disp('Wrong boundary condition,enter 1 or 2');
        bound_cond = input("Entry: ");
        c = 0;
    end
end
end

```

Input error handling:



```

Command Window
Length of domain: 100
Height of water surface from the ground: 12
Pani beshi govir!
Height of water surface from the ground: 10
x coordinate of disturbance: -10
Invalid x coordinate
x coordinate of disturbance: 10
y coordinate of disturbance: 101
Invalid y coordinate
y coordinate of disturbance: 10
Disturbance radius: 12
Radius exceeds the coordinate
Disturbance radius: 10
Disturbance intensity (1-100%) : 101
Invalid intensity,select between 1-100
Disturbance intensity (1-100%) : -1
Invalid intensity,select between 1-100
Disturbance intensity (1-100%) : 100
Damping coefficient (0-10) : 0.99
Enter boundary condition-
(1)For free boundary
(2)For reflective boundary
Entry: 4
Wrong boundary condition,enter 1 or 2
Entry: 0
Wrong boundary condition,enter 1 or 2
Entry: 1
Enter simulation time (s) : 4
Simulation ended
fx >>

```

A simulation end message pops up in the command prompt to indicate the termination of the simulation.

Result Analysis:

The following is a discussion of the results obtained for different input parameters.

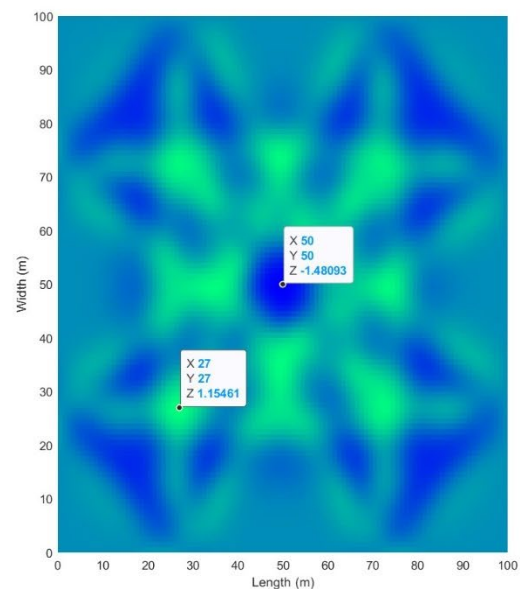
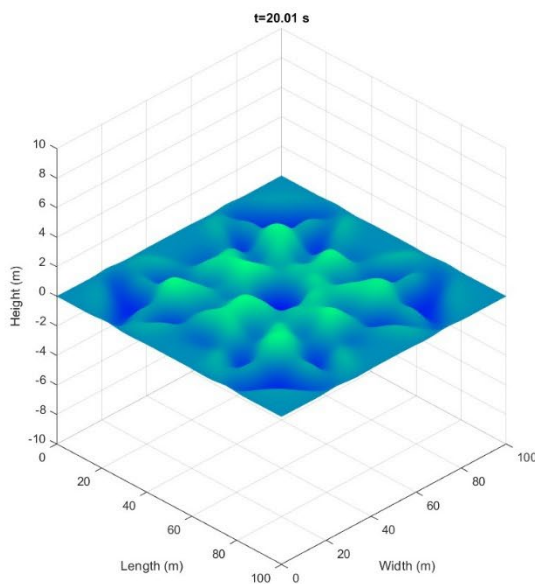
1. Change in Damping Coefficient:

For zero damping condition

Input

```
Command Window
Length of domain: 100
Height of water surface from the ground: 10
x coordinate of disturbance: 50
y coordinate of disturbance: 50
Disturbance radius: 10
Disturbance intensity (1-100%) : 100
Damping coefficient (0-10) : 0
Enter boundary condition-
(1)For free boundary
(2)For reflective boundary
Entry: 2
Enter simulation time (s) : 20
Simulation ended
fx >> |
```

Output

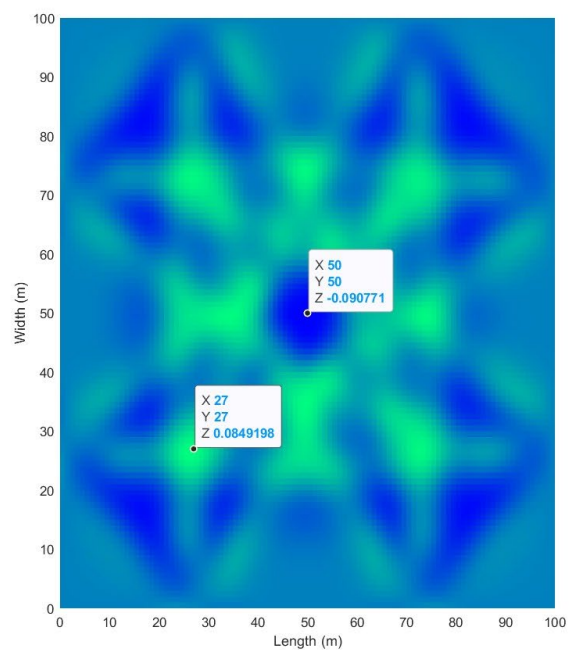
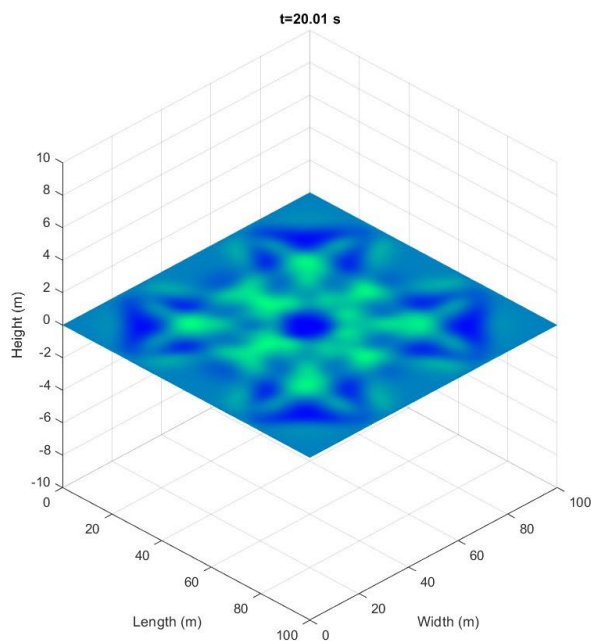


For damping coefficient = 3

Input

```
Command Window
Length of domain: 100
Height of water surface from the ground: 10
x coordinate of disturbance: 50
y coordinate of disturbance: 50
Disturbance radius: 10
Disturbance intensity (1-100%) : 100
Damping coefficient (0-10) : 3
Enter boundary condition-
  (1) For free boundary
  (2) For reflective boundary
Entry: 2
Enter simulation time (s) : 20
Simulation ended
fx >> |
```

Output



It is clear that the height of the water surface, relative to the tranquil state, when damping was added is much less after the same time interval. For the damped case, the water surface is almost flattened meaning that the water is reaching

towards tranquility. Whereas in the undamped case, the water never reaches a tranquil state as there is no energy dissipation.

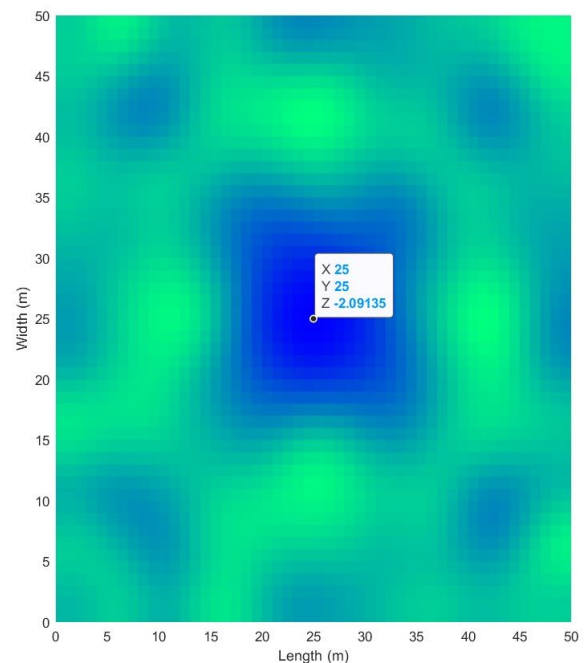
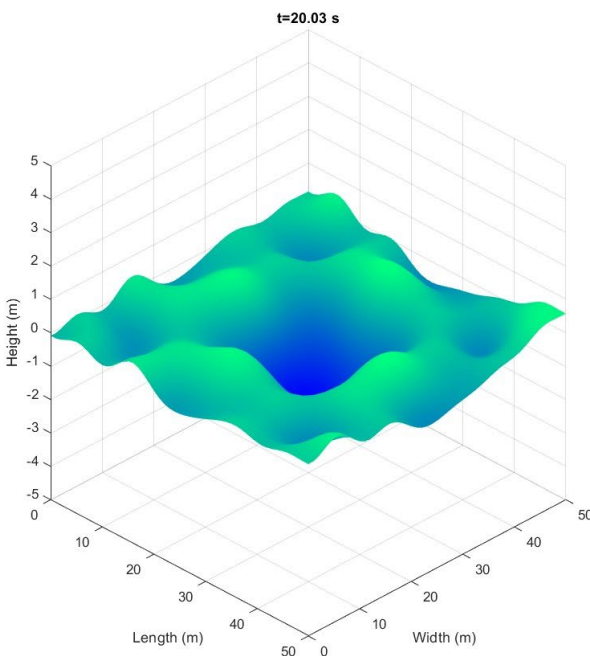
2. Change in the Intensity of Disturbance:

For 100% intensity

Input

```
Command Window
Length of domain: 50
Height of water surface from the ground: 5
x coordinate of disturbance: 25
y coordinate of disturbance: 25
Disturbance radius: 10
Disturbance intensity (1-100%) : 100
Damping coefficient (0-10) : 0
Enter boundary condition-
(1) For free boundary
(2) For reflective boundary
Entry: 1
Enter simulation time (s) : 20
Simulation ended
fx >> |
```

Output

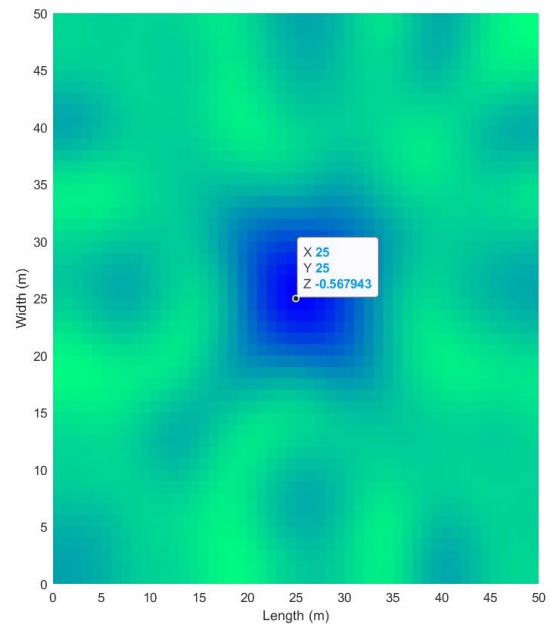
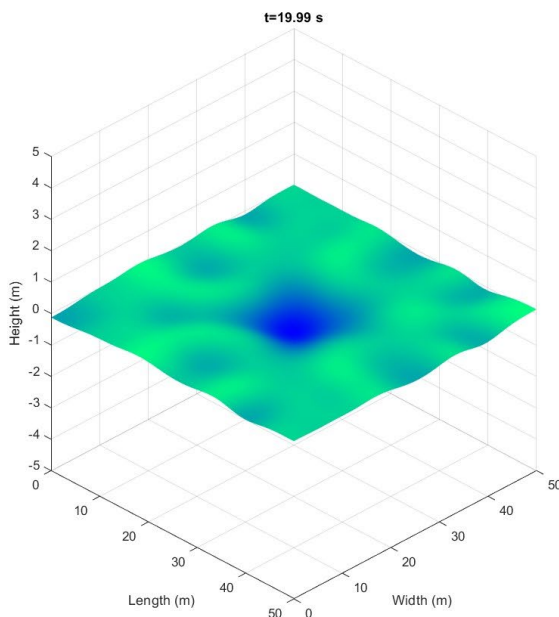


For 20% intensity

Input

```
Command Window
Length of domain: 50
Height of water surface from the ground: 5
x coordinate of disturbance: 25
y coordinate of disturbance: 25
Disturbance radius: 10
Disturbance intensity (1-100%) : 20
Damping coefficient (0-10) : 0
Enter boundary condition-
(1) For free boundary
(2) For reflective boundary
Entry: 1
Enter simulation time (s) : 20
Simulation ended
fx >> |
```

Output



As expected, the wave height is larger when the intensity is 100%. However, the wave pattern for both cases is the same since a change in intensity only affects the energy stored in the wave. So, after the same time interval, the disturbance pattern should be the same.

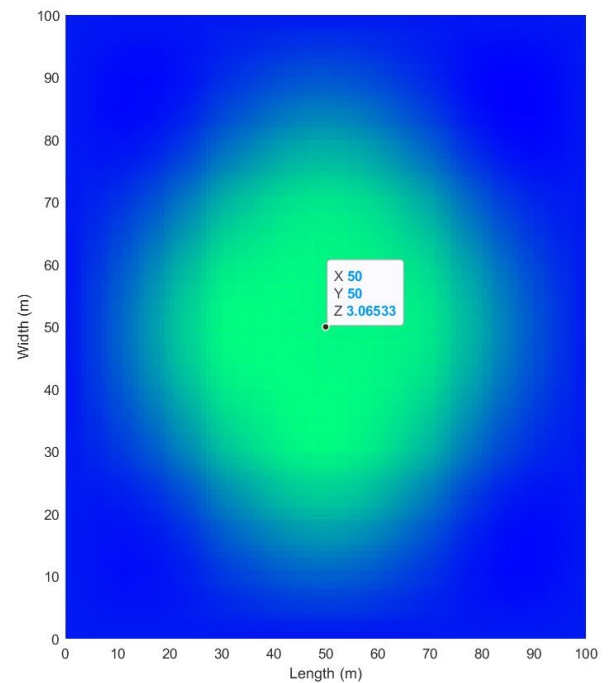
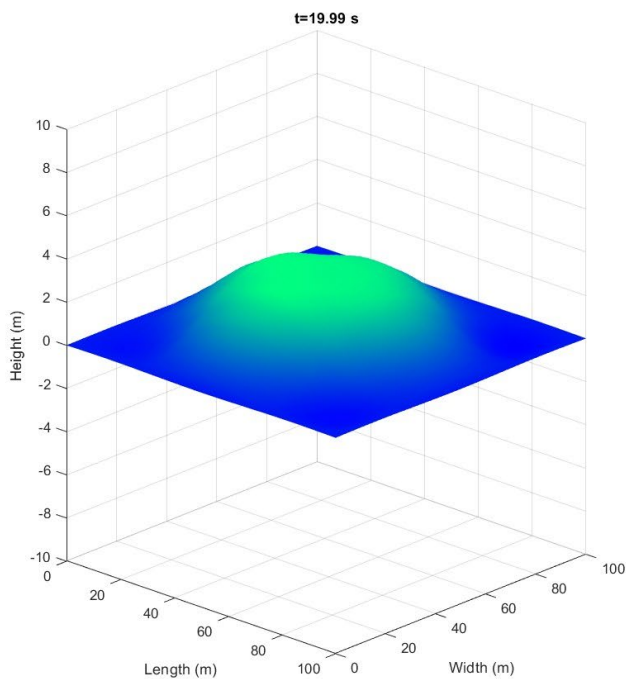
3. Change in Disturbance Radius:

When radius is 50m

Input

```
Command Window
Length of domain: 100
Height of water surface from the ground: 10
x coordinate of disturbance: 50
y coordinate of disturbance: 50
Disturbance radius: 50
Disturbance intensity (1-100%) : 100
Damping coefficient (0-10) : 0
Enter boundary condition-
(1)For free boundary
(2)For reflective boundary
Entry: 2
Enter simulation time (s) : 20
Simulation ended
fx >>
```

Output

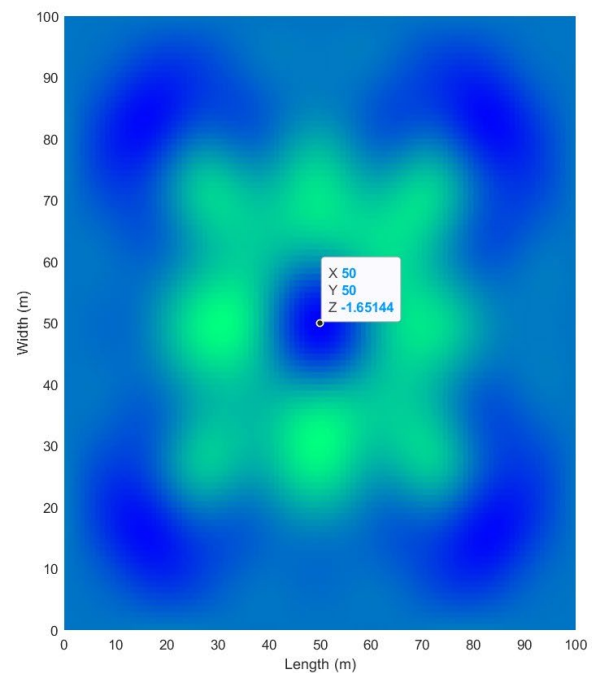
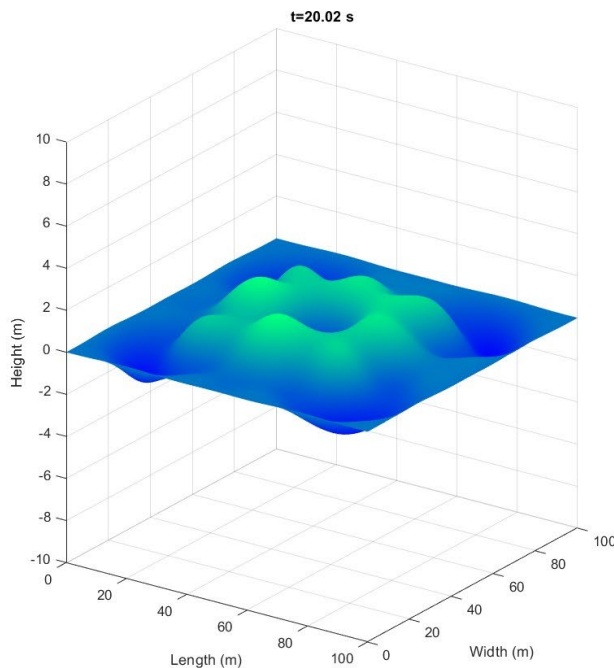


When radius is 20m

Input

```
Command Window
Length of domain: 100
Height of water surface from the ground: 10
x coordinate of disturbance: 50
y coordinate of disturbance: 50
Disturbance radius: 20
Disturbance intensity (1-100%) : 100
Damping coefficient (0-10) : 0
Enter boundary condition-
(1) For free boundary
(2) For reflective boundary
Entry: 2
Enter simulation time (s) : 20
Simulation ended
fx >>
```

Output



When radius is 50m, the entire domain falls under only one wavelength for 100m square region. However, for 20m radius successive peaks are formed before hitting the boundary. So the pattern is much complex.

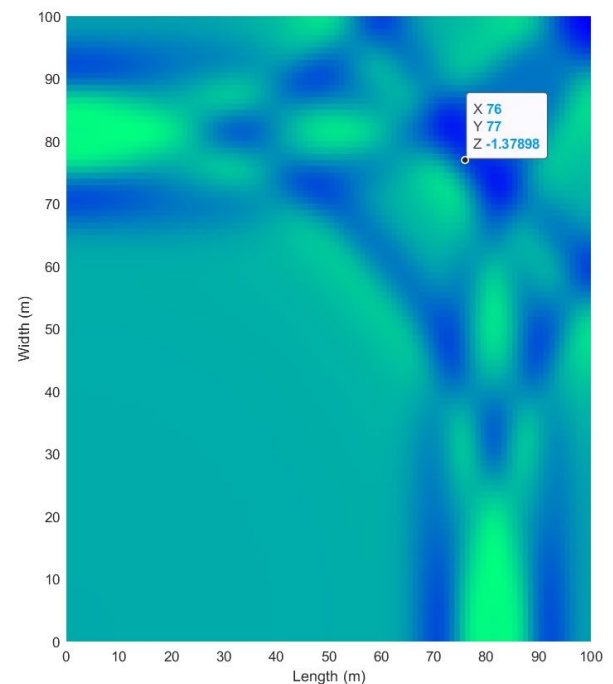
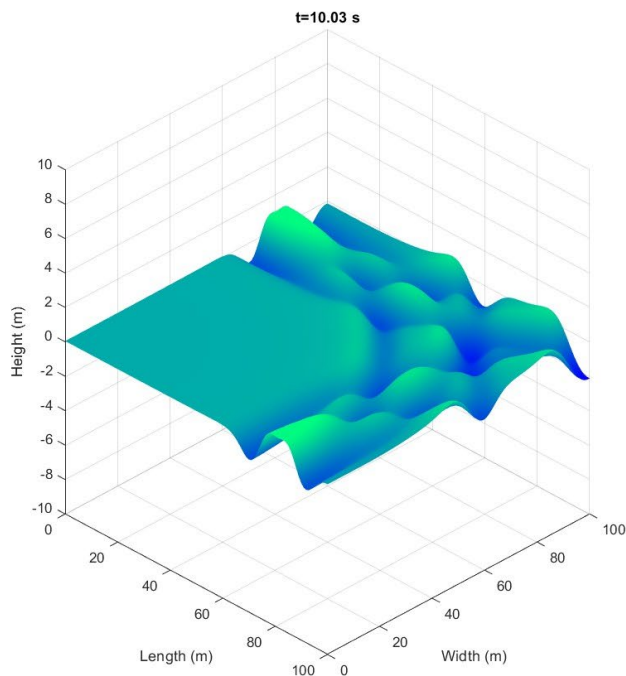
4. Change in Disturbance Coordinate:

When the disturbance is produced at the lower left corner of the region

Input

```
Command Window
Length of domain: 100
Height of water surface from the ground: 10
x coordinate of disturbance: 20
y coordinate of disturbance: 20
Disturbance radius: 10
Disturbance intensity (1-100%) : 100
Damping coefficient (0-10) : 0
Enter boundary condition-
(1)For free boundary
(2)For reflective boundary
Entry: 1
Enter simulation time (s) : 10
Simulation ended
fx >> |
```

Output

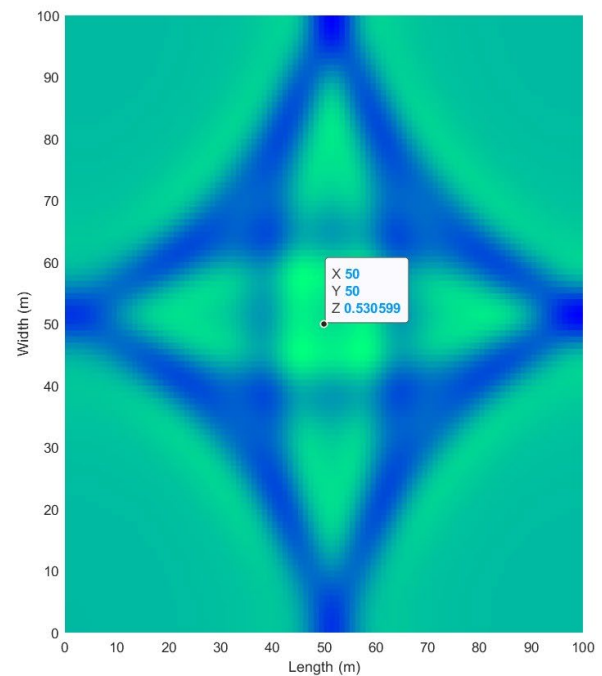
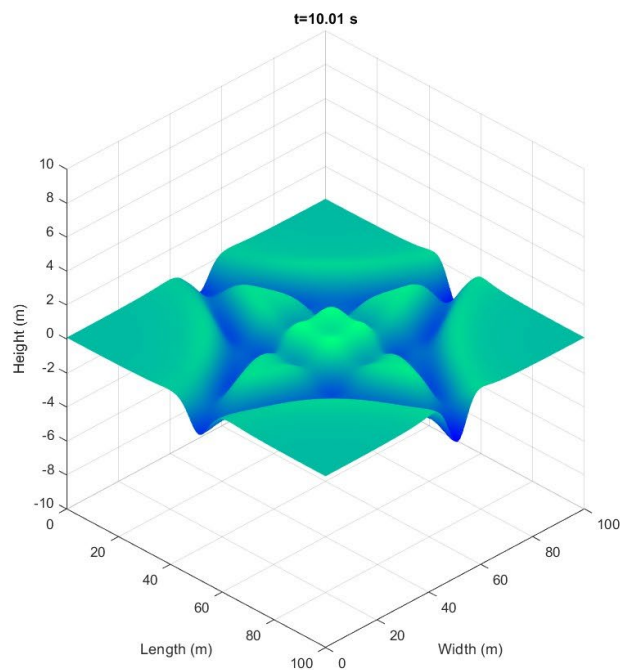


When the disturbance is created at the center

Input

```
Command Window
Length of domain: 100
Height of water surface from the ground: 10
x coordinate of disturbance: 50
y coordinate of disturbance: 50
Disturbance radius: 10
Disturbance intensity (1-100%) : 100
Damping coefficient (0-10) : 0
Enter boundary condition-
  (1) For free boundary
  (2) For reflective boundary
Entry: 1
Enter simulation time (s) : 10
Simulation ended
fx >>
```

Output



We observed that different initial positions produce different outputs, indicating the accuracy of our simulation.

5. Change in boundary condition:

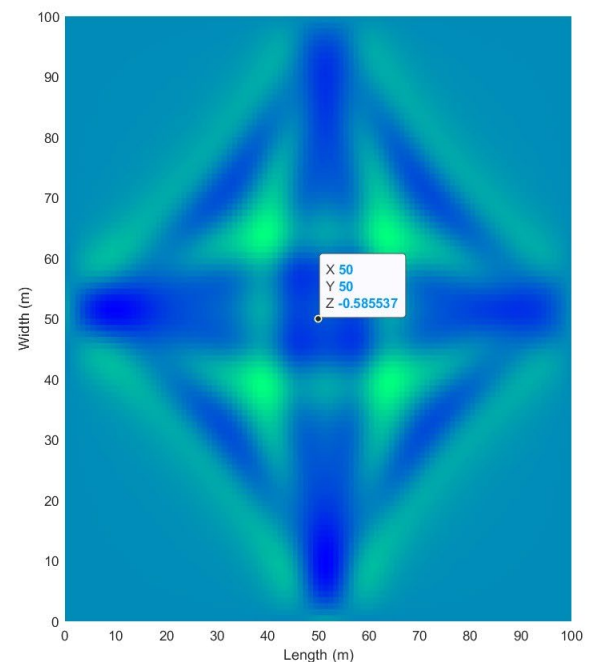
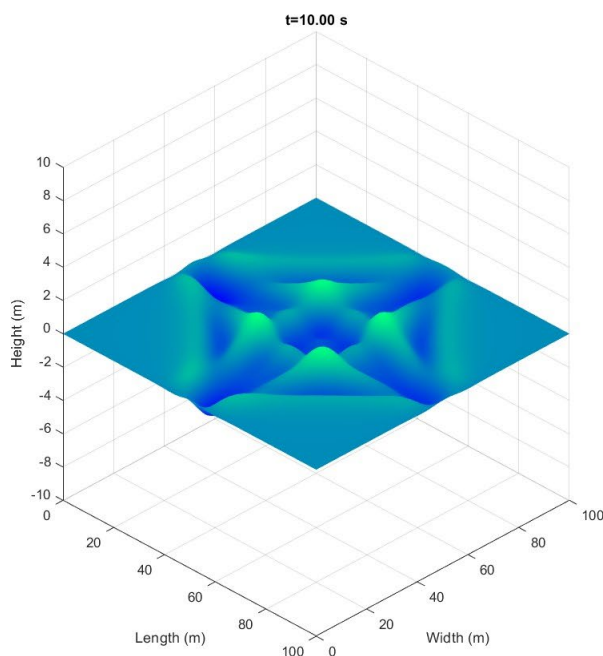
The previous observation for variation in the initial position of disturbance was conducted in the free boundary condition. We can continue the observation for reflective boundary condition.

For reflective boundary and central initial position of disturbance

Input

```
Command Window
Length of domain: 100
Height of water surface from the ground: 10
x coordinate of disturbance: 50
y coordinate of disturbance: 50
Disturbance radius: 10
Disturbance intensity (1-100%) : 100
Damping coefficient (0-10) : 0
Enter boundary condition-
(1)For free boundary
(2)For reflective boundary
Entry: 2
Enter simulation time (s) : 10
Simulation ended
fx >> |
```

Output

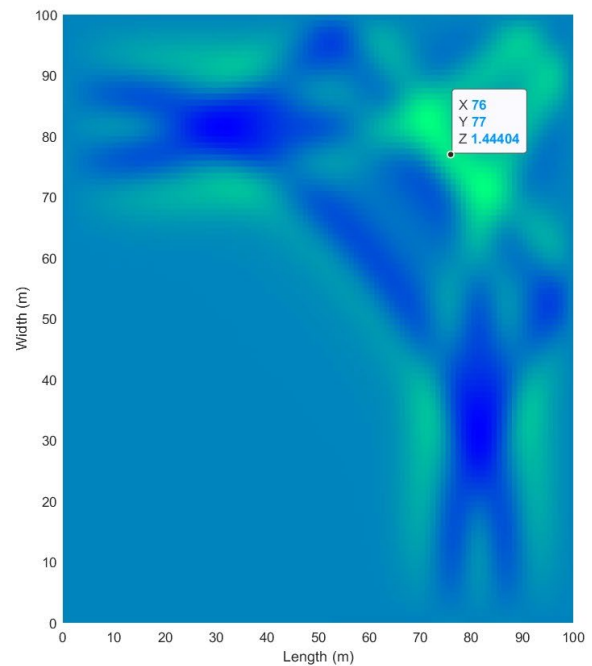
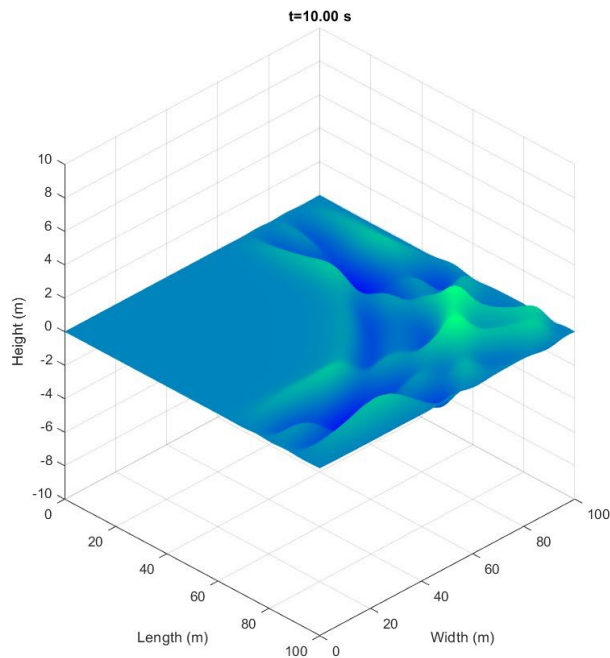


For reflective boundary and corner initial position of disturbance

Input

```
Command Window
Length of domain: 100
Height of water surface from the ground: 10
x coordinate of disturbance: 20
y coordinate of disturbance: 20
Disturbance radius: 10
Disturbance intensity (1-100%) : 100
Damping coefficient (0-10) : 0
Enter boundary condition-
(1) For free boundary
(2) For reflective boundary
Entry: 2
Enter simulation time (s) : 10
Simulation ended
fx >>
```

Output



We see that for the both corner and central initial conditions, the peaks are reversed when the boundary condition is changed. This means that when

boundary condition is changed, constructive interference and destructive interference interchange.

Limitations:

Although this simulation takes multiple inputs and displays animated output over time, it has limitations that can be addressed through updates.

1. The simulation is unable to consider multiple disturbances as inputs.
2. The simulation only shows the surface of the water system.
3. The 3D bell curve model is a simplified representation of disturbances, real-life disturbances are more complex.
4. Water always does not have a square boundary and the ground is not a square plane. Therefore, the height of the water surface varies from the ground.
5. Energy absorption at the boundary is not considered in this simulation.
6. Water is not always incompressible. At extremely high pressure, such as in the depths of the ocean, water can be compressed.
7. The propagation of disturbance causes a change in temperature, altering the kinetic energy of water molecules, so the process is not completely isothermal.
8. Water exhibits anisotropic behavior, meaning its density varies with direction.
9. The velocity of wave propagation in water can vary and is not uniform in both the x and y directions. Due to the non-isothermal process and the anisotropic nature of water, the velocity of propagation is never constant.
10. Assuming constant parameters control viscous forces in this simulation, however, as temperature, height from the ground, and wave velocity change, the viscous force also varies.

Conclusion:

In conclusion, the project performs a great job of simulating a simplified version of the shallow water system. Limitations aroused from the assumptions and approximation can be eliminated by further development of the topics involved.

Despite the shortcomings, the simulation offers a general idea of how water bodies behave.

Contribution:

Code part:

Mushfiquir Rahman (2106181) – Domain initialization, Input parameter with error handling, Disturbance initialization

Kayes Sami Malitha (2106184) – Update for boundary condition, Effect of damping, Animation display

The numerical part of the code was done together.

Report part:

Kayes Sami Malitha (2106184) – Objective, Theory, Assumptions

Mushfiquir Rahman (2106181) – Result analysis, Limitations, Conclusion

The methodology was written together.

References:

1. Mathworks, <https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/moler/exm/chapters/water.pdf>
2. Chorin, A. J. (1968). Numerical Solution of the Navier-Stokes Equations. *Mathematics of Computation*, 22(104), 745–762. <https://doi.org/10.2307/2004575>
3. Navier-Stokes Solver with Finite Difference Method - Retrieved from <https://www.mathworks.com/matlabcentral/answers/1934579-navier-stokes-solver-with-finite-difference-method-unexpected-results>

Main Code:

```
clc; close all; clear all;

% Define parameters
L = input("Length of domain: "); % Length of domain1
W = L; % Width of domain
Nx = L+1; % Number of grid points in x direction
Ny = Nx; % Number of grid points in y direction
dx = L / (Nx - 1); % Grid spacing in x direction
dy = W / (Ny - 1); % Grid spacing in y direction1
g = 9.81; % Gravity acceleration
[X Y] = meshgrid(linspace(0,L,Nx),linspace(0,W,Ny));

H = input("Height of water surface from the ground: "); % height of water surface
from the ground
while (H/(L)>0.1)
    disp("Pani beshi govir!");
    H = input("Height of water surface from the ground: ");
end

x_co = input("x coordinate of disturbance: "); % x coordinate of disturbance
while (x_co>L || x_co<0)
    disp("Invalid x coordinate");
    x_co = input("x coordinate of disturbance: ");
end

y_co = input("y coordinate of disturbance: "); % y coordinate of disturbance
while (y_co>L || y_co<0)
    disp("Invalid y coordinate");
    y_co = input("y coordinate of disturbance: ");
end

R = input("Disturbance radius: "); % disturbance size
while ( (x_co-R)<0 || (y_co-R)<0 )
    disp("Radius exceeds the coordinate");
    R = input("Disturbance radius: ");
end
%Intensity conditions
intensity = input("Disturbance intensity (1-100%) : ");%should be from 1 to 100
while (intensity>100 || intensity<1)
    disp("Invalid intensity,select between 1-100");
    intensity = input("Disturbance intensity (1-100%) : ");
end

damp = input("Damping coefficient (0-10) : ");
while (damp>10 || damp<0)
    disp("Invalid damping coefficient,select between 0-10");
    damp = input("Damping coefficient (0-10) : ");
end

disp("Enter boundary condition-");
disp(" (1)For free boundary");
disp(" (2)For reflective boundary");
```

```

bound_cond = input("Entry: "); % take input 1 or 2
c=0;
while c==0
    if (bound_cond==2 || bound_cond==1)
        c = 1;
    else
        disp('Wrong boundary condition,enter 1 or 2');
        bound_cond = input("Entry: ");
        c = 0;
    end
end

b = damp/(dx*dy*1*1000); % damping factor
amp=H*intensity/100;
T = input("Enter simulation time (s) : "); % Total simulation time
dt = 0.22; % Time step

% Initialize variables
for i=1:Nx
    for j=1:Ny
        h(i,j)=-amp*exp(-(i-x_co)^2 + (j-y_co)^2)/((R/2)^2));
    end
end

u = zeros(Nx, Ny); % Initial x-velocity
v = zeros(Nx, Ny); % Initial y-velocity

% Main loop
tic;
while toc<=T
    % Calculate dh/dx and dh/dy using forward finite differences
    dh_dx = zeros(Nx, Ny);
    dh_dy = zeros(Nx, Ny);
    for i = 1:Nx-1
        for j = 1:Ny
            dh_dx(i,j) = (h(i+1,j) - h(i,j)) / dx;
        end
    end
    for i = 1:Nx
        for j = 1:Ny-1
            dh_dy(i,j) = (h(i,j+1) - h(i,j)) / dy;
        end
    end

    % Calculate du/dt and dv/dt using shallow water equations
    du_dt = -g * dh_dx;
    dv_dt = -g * dh_dy;

    % Update velocities
    u = u + dt * du_dt;
    v = v + dt * dv_dt;

    % Update water height using continuity equation
    dh_dt = zeros(Nx, Ny);
    for i = 2:Nx-1

```

```

        for j = 2:Ny-1
            dh_dt(i,j) = -((u(i,j) - u(i-1,j)) / dx + (v(i,j) - v(i,j-1)) / dy);
        end
    end
    h = h + dt * dh_dt;

    % boundary Conditions for free boundary
    if bound_cond == 1
        h(:,1)=h(:,2);
        h(1,:)=h(2,:);
        h(Nx,:)=h(Nx-1,:);
        h(:,Ny)=h(:,Ny-1);
    else % boundary Conditions for reflective boundary
        h(:,1)=-h(:,2);
        h(1,:)=-h(2,:);
        h(Nx,:)=-h(Nx-1,:);
        h(:,Ny)=-h(:,Ny-1);
    end

    %Underdamping Decay Condition if damping is given
    h = h.*exp(-toc*b/2) ;
    %m = sum(sum(abs(h)));
    n = max(abs(h),[], "all");
    if (n <= 1e-6) % checking if the water surface is tranquil
        h = zeros(size(h));
        disp(toc);
        disp("Simulation ended,water surface is tranquil now.");
        break;
    end

    % plot results
    figure(1)
    subplot(1,2,1);
    surf(X, Y, h, 'EdgeColor', 'none');
    colormap('winter');
    zlim([-H H]);
    shading interp
    xlabel('Length (m)');
    ylabel('Width (m)');
    zlabel('Height (m)');
    title(sprintf('t=%.2f s', toc));
    view(45, 30);

    subplot(1,2,2);
    surf(X,Y,h);
    view(2)
    shading flat;
    xlabel('Length (m)') ;
    ylabel('Width (m)');

end;
if toc>= T
    disp("Simulation ended");
end
return;

```