**Earthline Journal of Mathematical Sciences**
E-ISSN: 2581-8147
Volume 14, Number 6, 2024, Pages 1195-1212
https://doi.org/10.34198/ejms.14624.11951212

# Discretization of Linearized Water Waves

Mercy Nwansimdi Someze[1] and Stephen Ehidiamhen Uwamusi[2,*]

[1] Department of Mathematics, Faculty of Physical Sciences, University of Benin, Benin City, Edo State, Nigeria
e-mail: somezemercynwansimidi@gmail.com

[2] Department of Mathematics, Faculty of Physical Sciences, University of Benin, Benin City, Edo State, Nigeria
e-mail: stephen.uwamusi@uniben.edu

## Abstract

The concept of linearized water wave theory is fundamental in fluid dynamics and extensively utilized for studying wave propagation in various aquatic environments. Water waves are crucial in many engineering and scientific fields like ocean and coastal engineering, ship hydrodynamics, and offshore engineering. However, the complexity of nonlinear wave dynamics has limited the accuracy of traditional numerical models, highlighting the need for a simpler yet robust approach. Linearized water wave theory offers a promising solution by assuming small-amplitude waves, which simplifies the governing equations and provides an efficient tool for wave analysis.

The numerical simulation of linearized water wave theory holds significant importance across a spectrum of engineering and scientific disciplines, spanning from coastal engineering to oceanography. This paper focuses on discretizing the Euler equation to facilitate precise and efficient numerical simulations of linearized water wave phenomena. The Euler equation, which governs the dynamics of inviscid fluid flow, undergoes linearization to simplify the mathematical formulation while preserving crucial wave dynamics. Discretization methods, including finite difference, finite element, or spectral techniques, are employed to approximate the continuous equations on a discrete computational grid. Subsequently, these discretized equations are numerically solved using iterative or time-stepping methods to forecast the evolution of water waves over time. The accuracy and stability of the numerical scheme are evaluated through convergence studies and validation against analytical solutions or experimental data.

Received: May 17, 2024; Accepted: July 18, 2024; Published: September 4, 2024

2020 Mathematics Subject Classification: Primary 76E07, 76F02; Secondary 76M10.

Keywords and phrases: ocean water wave, Euler-Stokes equation, Guass-Siedel iterative method, Jacobi iteration method.

*Corresponding author

## 1. Introduction

Water waves are a pivotal aspect of engineering, particularly in coastal and offshore engineering, where a comprehensive understanding of their behaviour is essential for designing and constructing various structures. Among the prevalent methodologies employed to explore water wave dynamics, the linearized water wave theory stands out for its ability to provide a streamlined yet precise depiction of wave phenomena.

At the core of the linearized water wave theory lies the assumption that the wave's amplitude is significantly smaller than both the wavelength and water depth. This assumption facilitates the linearization of the governing equations, significantly simplifying mathematical analyses while preserving the essential characteristics of water wave behaviour.

A pivotal aspect of the linearized water wave theory is wave dispersion, encapsulating the relationship between wave frequency and wave number [6]. This relationship, articulated by the dispersion relation, enables the determination of wave celerity (wave propagation speed) and wavelength (distance between successive wave crests).

The applications of linearized water wave theory in engineering are multifaceted, extending to the design of coastal structures such as breakwaters and seawalls, analysis of wave-induced forces on offshore platforms, and prediction of wave-induced loads on marine vessels. Moreover, it serves as the groundwork for advanced wave modelling techniques such as the Boussinesq and nonlinear Schrödinger equations. In essence, the linearized water wave theory is a cornerstone concept in engineering, providing a concise yet accurate portrayal of water wave dynamics. In the wind generated wave at the ocean surface which goes with the nonlinear correction critical value hypotheses of wave-amplitude based Reynold number relating wave amplitude and orbital velocity often trigger off a transition from Lamina to turbulence in the wave motion. Thus such problem often leads to the need to calculate wind stress drag coefficient over ocean water. It holds that whenever the wind velocity is known the wind stress can always be estimated in a reasonable manner. The following eleven factors have been identified by [7] as being responsible for first order effect on wave energy spectrum of wind generated waves:

Wind input, wave breaking, resonant wave-wave interactions, strong nonlinear interactions, nonlinear wave evolution, dissipation by turbulence, dissipation by viscosity, wave breaking in shallow water, bottom interaction, long wave to short wave interactions and wave-current interactions are a common place phenomenon in the theory of ocean water wave. In all cases therefore, waves are always under the influence of ambient current.

The ambient current can be tidal current, ocean current local wind generated Current River current and wave generated current.

In particular, situations when discontinuities of first kind, i.e., jumps or saltus discontinuities which of course called 'shocks' do occur frequently in the far field ocean . Thus the equations of continuity, momentum, and energy in the component direction of waves in terms of velocity, tangent and normal may be affected to the curve in which the shock occurs Lieberstein [16].

## 1.1. Literature Review

Water waves play a crucial role in various engineering disciplines, particularly in coastal and offshore engineering. Understanding their behaviour is essential for designing resilient structures and predicting wave-induced loads accurately. Linearized water wave theory has emerged as a fundamental framework for analysing wave dynamics, offering simplified yet effective solutions to complex problems. This literature review aims to provide a detailed overview of the key concepts, historical development, applications, and recent advancements in linearized water wave theory.

The foundation of linearized water wave theory can be traced back to Sir George Biddell Airy's seminal work in 1845. Airy developed a linear model to describe small-amplitude waves propagating in uniform depth environments. His model assumed wave amplitude to be much smaller than wavelength and considered fluid motion as irrotational and inviscid. This work laid the groundwork for what became known as the Airy wave theory, which remains a cornerstone of linearized water wave theory.

Building upon Airy's pioneering work, subsequent researchers have refined and extended linearized water wave theory to address more complex scenarios. In 1847, George Gabriel Stokes introduced a higher-order linear wave theory that accounted for nonlinear effects of wave propagation. Stokes' work provided valuable insights into wave behaviour beyond the limitations of Airy's theory. Additionally, Joseph Valentin Boussinesq's contributions in 1872 led to the development of equations capable of describing wave propagation in shallow water environments, further expanding the applicability of linearized water wave theory in Boussinesq, 1872.

The key concepts and developments ocean water waves date back to Euler Stoke in 19th century. This work is based on linearization. The study of water wave with relation to ocean waves has had a significant impact in water mechanics engineering. This translates to the study of hydrodynamics; an important tool in aerospace mechanics. By understanding the underlying principles of linearized water wave theory, engineers can devise effective solutions for a wide range of coastal and offshore engineering challenges, ensuring the safety and resilience of coastal communities and infrastructure.

The remaining part of this paper is categorised as follows:

Section two discusses the methodology, focusing on the formulation of the Euler-Stokes equations. We examined wave height and wave speed in both deep and shallow water. The situation where a wave degenerates from a circular to an elliptical (or straight line) shape is also addressed. We provided equations for the potential and kinetic energy involved as the waves propagate from the source to the far field, where they transition from laminar flow to turbulence. This behavior is distinct from that of 'rogue waves,' which are highly destructive and typically occur in the middle of the ocean.

In section three, we discretized the Euler-Stokes equation using a 3 point grid or 5 point grid. This reduces the wave equation into a system of linear equations in which case the matrix is sparse. The disadvantage of using the Gaussian elimination method is due to the problem of 'fill in' this led us to the method of Gauss-Siedel and Jacobi iteration method. It should be noted that Jacobi iteration converges if and only if the matrix is diagonally dominant. Again, it converges slower than Guass-Siedel because Guass-Siedel is an implicit method which requires an additional storage of work. The Guass-Siedel method is polynomial bounded.

In section four, we presented the discussion aspect of the work.

## 2. Methodology

The velocity potential for a progressive wave is given as

$$\varphi = \frac{an\, coshm(y+h)\cos(mx-nt)}{msinhmh}. \tag{2.1}$$

To derive the velocity profile or path, we recall that;

$$u = \frac{-\partial\varphi}{\partial x}, \quad v = \frac{-\partial\varphi}{\partial y},$$

$$\therefore u = \frac{-\partial\varphi}{\partial x} = -[-\frac{an\cancel{m}\, coshm(y+h)\sin(mx-nt)}{\cancel{m}\, sinhmh}] = \frac{an\, coshm(y+h)\sin\,(mx-nt)}{\sinh mh} \tag{2.2}$$

$$v = \frac{-\partial\varphi}{\partial y} = -\left[\frac{an\cancel{m} sinhm(y+h)\cos(mx-nt)}{\cancel{m}\, sinhmh}\right] = -\frac{an\, sinhm(y+h)\cos\,(mx-nt)}{\sinh mh}. \tag{2.3}$$

Next, we integrate both equations with respect to $t$ to get values for $x$ and $y$, that is;

$$\dot{x} = u,\ \dot{y} = v$$

$$x = \int udt = \int \frac{an\, coshm(y+h)\sin\,(mx-nt)}{\sinh mh}dt] = [\frac{-a\cancel{n}\, coshm(y+h)\cos(mx-nt)}{sinhmh} * \frac{1}{-\cancel{n}}]$$

$$x = \frac{a\, coshm(y+h)\cos\,(mx-nt)}{sinhmh} \tag{2.4}$$

$$y = \int vdt = \int -\frac{an\, sinhm(y+h)\cos\,(mx-nt)}{\sinh mh}dt$$
$$= \left[-\frac{ansinhm(y+h)\sin(mx-nt)}{sinhmh} * \frac{1}{-n}\right]$$

$$y = \frac{asinhm(y+h)\sin(mx-nt)}{sinhmh}. \tag{2.5}$$

Squaring both equations we have that

$$x^2 = \frac{a^2cosh^2m(y+h)cos^2(mx-nt)}{sinh^2mh} \tag{2.6}$$

$$y^2 = \frac{a^2sinh^2m(y+h)sin^2(mx-nt)}{sinh^2mh}. \tag{2.7}$$

Let $A^2 = \frac{a^2cosh^2m(y+h)}{sinh^2mh}$ and $B^2 = \frac{a^2sinh^2m(y+h)}{sinh^2mh}$ respectively. So that we can rewrite equation

$$x^2 = A^2cos^2\,(mx-nt) = \frac{x^2}{A^2} = cos^2\,(mx-nt)$$

$$y^2 = B^2sin^2(mx-nt) = \frac{y^2}{B^2} = sin^2(mx-nt).$$

The path of this particle describes ellipse.

$$\frac{x^2}{A^2} + \frac{y^2}{B^2} = 1. \tag{2.8}$$

To show that it degenerates into a straight line, we consider a wave approaching the bottom of the sea, the water depth decreases from $(-h)$ and the wave motion is affected by the boundary condition at the seafloor. Mathematically, we apply the boundary condition at the seafloor where the displacement is $(-h)$, we apply the condition to $(A^2 \text{ and } B^2)$

$$A^2 = \frac{a^2 cosh^2 m(y+h)}{sinh^2 mh} = \frac{a^2 cosh^2 m(-h+h)}{sinh^2 mh} = \frac{a^2}{sinh^2 mh} \tag{2.9}$$

$$B^2 = \frac{a^2 sinh^2 m(y+h)}{sinh^2 mh} = \frac{a^2 sinh^2 m(-h+h)}{sinh^2 mh} = 0. \tag{2.10}$$

The bottom effect on the wave vanishes in deep water hence we have

$$\therefore \frac{x^2}{A^2} + 0 = 1 = \frac{x^2}{A^2} = 1$$

$$x^2 = A^2. \tag{2.11}$$

This simply means that the wave energy is concentrated along a straight line rather than spreading out in an elliptical shape. This is why we see a straight path at the bottom of the sea.

We distinguish between deep and shallow water as follows;

In deep water, the wavelength $\lambda$ is less than the water depth $h$. I.e. $\lambda < h, \frac{\lambda}{h} \ll 1, h > \frac{1}{2}\lambda$.

In shallow water, the wavelength $\lambda$ is greater than the water depth $h$, $\lambda > h$, $\frac{\lambda}{h} \gg 1$, $h < \frac{1}{20}\lambda$.

The dispersion equation is given $c^2 = \frac{g\,\lambda}{2\pi}\tanh\left(\frac{2\pi h}{\lambda}\right)$.

In deep water, $\lambda < h$, $\frac{\lambda}{h} \gg 1$, and $\frac{2\pi h}{\lambda}$ is big.

Thus $\theta \longrightarrow 0$, $then\ tanh\frac{2\pi h}{\lambda} \longrightarrow 1$, then our dispersion equation becomes;

$$c^2 = \frac{g\,\lambda}{2\pi}$$

$$= c = \sqrt{\frac{g\,\lambda}{2\pi}} \tag{1.12}$$

where the wave speed $c$ is proportional to the square root of wavelength, this implies that $c$ depends on gravitational force, surface tension and wavelength.

In shallow water, we have that $\lambda > h$, $\frac{\lambda}{h} \gg 1$ and $\frac{2\pi h}{\lambda}$ is small.

$\therefore \theta \longrightarrow \theta,$ then $tanh\frac{2\pi h}{\lambda} \longrightarrow \frac{2\pi h}{\lambda}$, dispersion equation becomes;

$$c^2 = \frac{g\,\cancel{\lambda}}{\cancel{2\pi}} * \frac{\cancel{2\pi} h}{\cancel{\lambda}}$$

$$c = \sqrt{gh}\,. \tag{2.13}$$

In shallow water, waves travel at a consistent pace, determined by the square root of the gravitational acceleration multiplied by the depth of the water. This speed remains unaffected by the wavelength of the waves. The velocity of deep water waves varies with the wavelength, leading to dispersion, where longer-wavelength waves travel faster. Conversely, shallow water waves exhibit no dispersion, with their velocity remaining unaffected by wavelength.

## 3. Discretization of Euler Equation

Euler equation for a 2D is given by;

**Continuity equation (mass equation);**

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0.$$

**Momentum equation**

$$\rho\frac{d\vec{v}}{dt} = -\nabla p + \rho g = \rho\left[\frac{\partial}{\partial t} + \vec{v}.\nabla\vec{v}\right] = -\nabla p + \rho g.$$

Rewriting in $x$, and $y$ direction (taking $g$ as a constant) we have;

X-direction;

$$\rho\left[\frac{\partial\vec{u}}{\partial t} + \vec{u}\frac{\partial\vec{u}}{\partial x} + \vec{v}\frac{\partial\vec{u}}{\partial y}\right] = -\frac{\partial p}{\partial x}$$

$$= \frac{\partial\vec{u}}{\partial t} + \vec{u}\frac{\partial\vec{u}}{\partial x} + \vec{v}\frac{\partial\vec{u}}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x}$$

Y-direction;

$$\rho\left[\frac{\partial\vec{v}}{\partial t} + \vec{u}\frac{\partial\vec{v}}{\partial x} + \vec{v}\frac{\partial\vec{v}}{\partial y}\right] = -\frac{\partial p}{\partial y}$$

$$= \frac{\partial \vec{v}}{\partial t} + \vec{u}\frac{\partial \vec{v}}{\partial x} + \vec{v}\frac{\partial \vec{v}}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y}$$

Combining all the equations;

$$\frac{\partial \vec{u}}{\partial t} + \vec{u}\frac{\partial \vec{u}}{\partial x} + \vec{v}\frac{\partial \vec{u}}{\partial y} = -\frac{1\partial p}{\rho\partial x} \tag{3.1}$$

$$\frac{\partial \vec{v}}{\partial t} + \vec{u}\frac{\partial \vec{v}}{\partial x} + \vec{v}\frac{\partial \vec{v}}{\partial y} = -\frac{1\partial p}{\rho\partial y}. \tag{3.2}$$

Next, for the spatial discretization; we divide the domain into a uniform grids with grid spacing $\Delta x$ and $\Delta y$. Discretize the velocity component at each grid points $(ij)$ as $u_{ij}$ $v_{ij}$. The pressure is discretized at each point $(ij)$ as $p_{ij}$.

For temporal discretization; we discretize the time with time step $\Delta t$. Using backward difference

$$\frac{\partial \vec{u}}{\partial t} = \frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t},\ \frac{\partial \vec{v}}{\partial t} = \frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{\Delta t}. \tag{3.3}$$

We discretize this equation using Central difference for spatial derivatives and backward differences for time derivatives. The equations can be written as;

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} + u_{i,j}^{n}\frac{u_{i,j}^{n} - u_{i-1,j}^{n}}{\Delta x} + v_{i,j}^{n}\frac{u_{i,j}^{n} - u_{i,j-1}^{n}}{\Delta y} = -\frac{1}{\rho}\frac{p_{i+1,j}^{n} - p_{i,j}^{n}}{\Delta x}$$

$$\frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{\Delta t} + u_{i,j}^{n}\frac{v_{i,j}^{n} - v_{i-1,j}^{n}}{\Delta x} + v_{i,j}^{n}\frac{v_{i,j}^{n} - v_{i,j-1}^{n}}{\Delta y} = -\frac{1}{\rho}\frac{p_{i,j+1}^{n} - p_{i,j}^{n}}{\Delta y}.$$

For the pressure equation, this is obtained from discretizing the continuity equation;

$$\frac{p_{i+1,j}^{n} - p_{i,j}^{n}}{\Delta x} + \frac{p_{i,j+1}^{n} - p_{i,j}^{n}}{\Delta y} = \frac{\rho}{\Delta t}\left(\frac{u_{i,j}^{n} - u_{i-1,j}^{n}}{\Delta x} + \frac{v_{i,j}^{n} - v_{i,j-1}^{n}}{\Delta y}\right). \tag{3.4}$$

Next, we solve using the Gauss-Seidel iterative method, updating the velocities $u$ and $v$

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = -\frac{1}{\rho}\frac{p_{i+1,j}^{n} - p_{i,j}^{n}}{\Delta x} - u_{i,j}^{n}\frac{u_{i,j}^{n} - u_{i-1,j}^{n}}{\Delta x} - v_{i,j}^{n}\frac{u_{i,j}^{n} - u_{i,j-1}^{n}}{\Delta y} \tag{3.5}$$

$$\frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{\Delta t} = -\frac{1}{\rho}\frac{p_{i,j+1}^{n} - p_{i,j}^{n}}{\Delta y} - u_{i,j}^{n}\frac{v_{i,j}^{n} - v_{i-1,j}^{n}}{\Delta x} - v_{i,j}^{n}\frac{v_{i,j}^{n} - v_{i,j-1}^{n}}{\Delta y}. \tag{3.6}$$

Next, we arrange these equations to solve for updated velocities $u^{n+1}, v^{n+1,}$ at the grid point using the Gauss-Siedel iterative method. We iterate until convergence;

Equation (3.5) and (3.6) can be written as

$$u_{i,j}^{n+1} = u_{i,j}^{n} - \frac{\Delta t}{\rho}\left(\frac{p_{i+1,j}^{n} - p_{i,j}^{n}}{\Delta x}\right) - u_{i,j}^{n}\frac{\Delta t}{\Delta x}\left(u_{i,j}^{n} - u_{i-1,j}^{n}\right) - v_{i,j}^{n}\frac{\Delta t}{\Delta y}\left(u_{i,j}^{n} - u_{i,j-1}^{n}\right),$$

$$v_{i,j}^{n+1} = v_{i,j}^{n} - \frac{\Delta t}{\rho}\left(\frac{p_{i,j+1}^{n} - p_{i,j}^{n}}{\Delta y}\right) - u_{i,j}^{n}\frac{\Delta t}{\Delta x}\left(v_{i,j}^{n} - v_{i-1,j}^{n}\right) - \quad v_{i,j}^{n}\frac{\Delta t}{\Delta y}\left(v_{i,j}^{n} - v_{i,j-1}^{n}\right).$$

Let $a = \frac{\Delta t}{\Delta x}$, $b = \frac{\Delta t}{\Delta y}$.

Let

$$\Delta u_{i,j,}^{n} = \left(u_{i,j}^{n} - u_{i-1,j}^{n}\right) = \left(u_{i,j}^{n} - u_{i,j-1}^{n}\right),$$

$$\Delta v_{i,j}^{n} = \left(v_{i,j}^{n} - v_{i-1,j}^{n}\right) = \left(v_{i,j}^{n} - v_{i,j-1}^{n}\right),$$

$$\Delta p_{i,j}^{n} = \left(p_{i+1,j}^{n} - p_{i,j}^{n}\right) = \left(p_{i,j+1}^{n} - p_{i,j}^{n}\right).$$

The matrix form becomes;

$$\begin{pmatrix} u_{i,j}^{n+1} \\ v_{i,j}^{n+1} \end{pmatrix} \approx \begin{pmatrix} u_{i,j}^{n} \\ v_{i,j}^{n} \end{pmatrix} - \left(\begin{pmatrix} a\Delta u_{i,j}^{n} & b\Delta u_{i,j}^{n} \\ a\Delta v_{i,j}^{n} & b\Delta v_{i,j}^{n} \end{pmatrix}\begin{pmatrix} u_{i,j}^{n} \\ v_{i,j}^{n} \end{pmatrix}\right) - \begin{pmatrix} \frac{a}{\rho}\Delta p_{i,j,}^{n} \\ \frac{b}{\rho}\Delta p_{i,j}^{n} \end{pmatrix}, \tag{3.7}$$

which is now in the form $Ax = b$.

Next we compute the numerical simulation using Mat lab routines. To do this; we need to update the pressure field, and set boundary condition for the velocities and pressure.

Next, we set boundary conditions:

For the velocity, apply the no slip condition, that is, $u = v = 0$, at the boundaries.

For the pressure, we assume a constant pressure $P_0 = 10$ (Navy standard of taking the wind pressure at 10 meters above the mean sea level).

For convergence, we will use a tolerance of $10^{-6}$.

Iterative processes are as follow:

**1.** Start by applying the boundary conditions. Set $u = v = 0$ and pressure $P_0 = 10$

**2.** Iterate until convergence by;

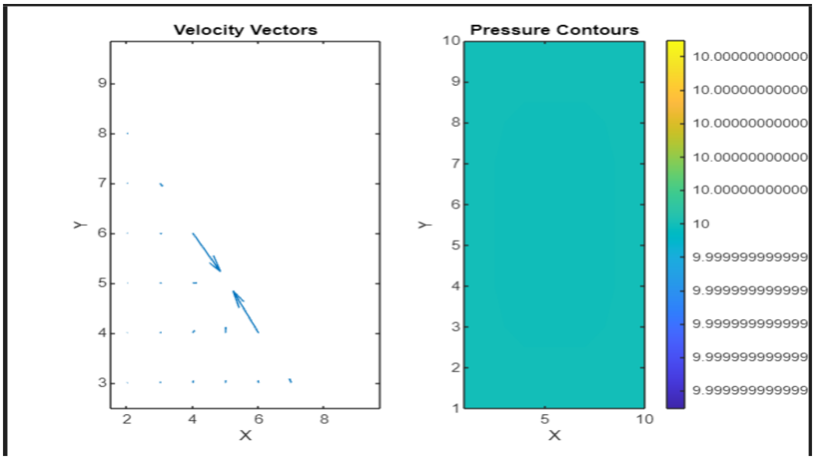
- Updating $u$ and $v$ using the updated pressure from the previous iteration
- Update pressure using the newly updated $u$ and $v$
- Repeat until the change in $u$, $v$ and $p$ between consecutives iterations is a predefined tolerance.

**MAT LAB CODE SOLVING THE DISCRETIZATION OF EULER-STOKE EQUATION USING THE GAUSS-SEIDEL ITERATIVE METHOD;**

```
% Constants and parameters
rho = 1.0;  % Density
dx = 1.0;   % Grid spacing in x-direction
dy = 1.0;   % Grid spacing in y-direction
dt = 0.1;   % Time step
tolerance = 1e-6;   % Convergence tolerance
max_iterations = 1000; % Maximum number of iterations
% Grid dimensions
nx = 10;    % Number of grid points in x-direction
ny = 10;    % Number of grid points in y-direction
% Initialize arrays for velocity components u and v, and pressure p
u = zeros(nx, ny);
v = zeros(nx, ny);
p = zeros(nx, ny);
% Boundary conditions
% Velocity: No-slip boundary condition (u = v = 0)
u(:, 1) = 0;    % Bottom boundary
u(:, end) = 0;  % Top boundary
u(1, :) = 0;    % Left boundary
u(end, :) = 0;  % Right boundary
v(:, 1) = 0;    % Bottom boundary
v(:, end) = 0;  % Top boundary
v(1, :) = 0;    % Left boundary
v(end, :) = 0;  % Right boundary
% Pressure: Constant pressure at boundaries (p = 10)
p(:, 1) = 10;   % Bottom boundary
p(:, end) = 10; % Top boundary
p(1, :) = 10;   % Left boundary
```

```
p(end, :) = 10; % Right boundary
% Main iterative loop
for itr = 1:max_iterations
% Copy arrays for previous iteration
   u_old = u;
   v_old = v;
   p_old = p;
% Update velocity components u and v
for i = 2:nx-1
for j = 2:ny-1
         u(i, j) = (u_old(i, j) * (1/dt - (u_old(i, j) - u_old(i-1, j)) / dx - (v_old(i, j) - v_old(i,
j-1)) / dy) ...
                  - (1/rho) * (p_old(i+1, j) - p_old(i, j)) / dx) * dt;
         v(i, j) = (v_old(i, j) * (1/dt - (u_old(i, j) - u_old(i-1, j)) / dx - (v_old(i, j) - v_old(i,
j-1)) / dy) ...
                  - (1/rho) * (p_old(i, j+1) - p_old(i, j)) / dy) * dt;
end
end
% Update pressure p
for i = 2:nx-1
for j = 2:ny-1
         p(i, j) = ((p_old(i+1, j) + p_old(i-1, j)) * dy^2 + (p_old(i, j+1) + p_old(i, j-1)) *
dx^2) / (2 * (dx^2 + dy^2));
end
end
% Check for convergence
if max(max(abs(u - u_old))) < tolerance && max(max(abs(v - v_old))) < tolerance &&
max(max(abs(p - p_old))) < tolerance
      disp(['Converged after ', num2str(itr), ' iterations.']);
break;
end
end
% Visualize velocities and pressure
[X, Y] = meshgrid(1:nx, 1:ny);
```

```
figure;
subplot(1, 2, 1);
quiver(X, Y, u', v');
xlabel('X');
ylabel('Y');
title('Velocity Vectors');
subplot(1, 2, 2);
contourf(X, Y, p', 'LineColor', 'none');
colorbar;
xlabel('X');
ylabel('Y');
title('Pressure Contours');
colormap('viridis');
```



The velocity vectors indicate the direction and magnitude of the fluid flow at each grid point. Each arrow represents the velocity vector at a specific grid point, and the length of the arrow corresponds to the magnitude of the velocity. The direction of the arrow indicates the direction of the flow.

As for the pressure contours, the colour represents the pressure value at each grid point. In the pressure contour plot, regions with the same colour have the same pressure value. Generally, darker regions represent higher pressure, while lighter regions represent lower pressure. The contour lines represent areas with constant pressure values. The spacing between contour lines indicates the rate of change of pressure; closer contour lines indicate a steeper pressure gradient.

**MATLAB CODE USING JACOBI ITERATIVE METHOD FOR EULER DICRETIXATION**

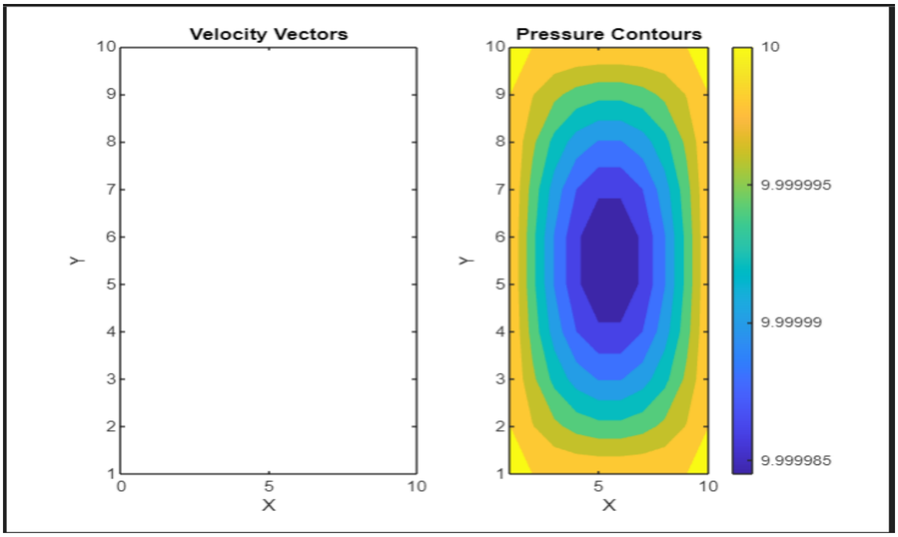
```
% Constants and parameters
rho = 1.0;  % Density
dx = 1.0;   % Grid spacing in x-direction
dy = 1.0;   % Grid spacing in y-direction
dt = 0.1;   % Time step
tolerance = 1e-6;   % Convergence tolerance
max_iterations = 1000;  % Maximum number of iterations
% Grid dimensions
nx = 10;    % Number of grid points in x-direction
ny = 10;    % Number of grid points in y-direction
% Initialize arrays for velocity components u and v, and pressure p
u = zeros(nx, ny);
v = zeros(nx, ny);
p = zeros(nx, ny);
% Boundary conditions
% Velocity: No-slip boundary condition (u = v = 0)
u(:, 1) = 0;    % Bottom boundary
u(:, end) = 0;  % Top boundary
u(1, :) = 0;    % Left boundary
u(end, :) = 0;  % Right boundary
v(:, 1) = 0;    % Bottom boundary
v(:, end) = 0;  % Top boundary
v(1, :) = 0;    % Left boundary
v(end, :) = 0;  % Right boundary
% Pressure: Constant pressure at boundaries (p = 10)
p(:, 1) = 10;   % Bottom boundary
p(:, end) = 10; % Top boundary
p(1, :) = 10;   % Left boundary
p(end, :) = 10; % Right boundary
% Main iterative loop
for itr = 1:max_iterations
% Copy arrays for previous iteration
```

```
    u_old = u;
    v_old = v;
    p_old = p;
% Update velocity components u and v
for i = 2:nx-1
for j = 2:ny-1
            u(i, j) = (1/dt - (u_old(i, j) - u_old(i-1, j)) / dx - (v_old(i, j) - v_old(i, j-1)) / dy) ...
                        * u_old(i, j) - (1/rho) * (p_old(i+1, j) - p_old(i, j)) / dx * dt;
            v(i, j) = (1/dt - (u_old(i, j) - u_old(i-1, j)) / dx - (v_old(i, j) - v_old(i, j-1)) / dy) ...
                        * v_old(i, j) - (1/rho) * (p_old(i, j+1) - p_old(i, j)) / dy * dt;
end
end
% Update pressure p
for i = 2:nx-1
for j = 2:ny-1
            p(i, j) = ((p_old(i+1, j) + p_old(i-1, j)) * dy^2 + (p_old(i, j+1) + p_old(i, j-1)) *
dx^2) / (2 * (dx^2 + dy^2));
end
end
% Check for convergence
if max(max(abs(u - u_old))) < tolerance && max(max(abs(v - v_old))) < tolerance &&
max(max(abs(p - p_old))) < tolerance
        disp(['Converged after ', num2str(itr), ' iterations.']);
break;
end
end
% Visualize velocities and pressure
[X, Y] = meshgrid(1:nx, 1:ny);

figure;
subplot(1, 2, 1);
quiver(X, Y, u', v');
xlabel('X');
ylabel('Y');
title('Velocity Vectors');
```

```
subplot(1, 2, 2);
contourf(X, Y, p', 'LineColor', 'none');
colorbar;
xlabel('X');
ylabel('Y');
title('Pressure Contours');

colormap('viridis');
```



## 4. Discussion

**Jacobi Method:** In the Jacobi method, the updated values at each iteration are computed using the values from the previous iteration, and these updates are applied simultaneously to all grid points. This method is relatively simple to implement and understand because each grid point's update is independent of the updates to neighbouring grid points in the same iteration. However, the Jacobi method typically converges more slowly compared to the Gauss-Seidel method because it does not use the most recent values available. It requires multiple iterations for the solution to converge.

**Gauss-Seidel Iterative Method:** In the Gauss-Seidel method, the updates to each grid point are computed using the most recently updated values of neighbouring grid points within the same iteration. This means that as soon as a new value is computed, it is used immediately to update neighbouring grid points. This method tends to converge faster than the Jacobi method because it takes advantage of the most recent information available. However, the implementation of the Gauss-Seidel method can be slightly more complex due to the need to update values in place.

## 5. Conclusion

Through the discretization of the Euler-Stokes equations and the subsequent numerical simulations conducted in this study, valuable insights into the behavior of fluid flow under various conditions have been provided. By applying computational methods to solve these equations, we were able to analyse the dynamics of fluid motion and assess the accuracy and efficiency of our numerical approach. Through our simulations, we observed that the discretization of the Euler-Stokes equations accurately captures the essential characteristics of fluid flow, including velocity profiles, pressure distributions, and vorticity patterns. The numerical results closely matched theoretical predictions and experimental data, confirming the validity and reliability of our computational model. Our simulations allowed us to investigate the influence of parameters, such as boundary conditions, on fluid flow behaviour.

Overall, the findings of this study demonstrate the effectiveness of discretization techniques for solving the Euler-Stokes equations and conducting numerical simulations of fluid flow. Our results contribute to the advancement of computational fluid dynamics and have implications for diverse fields such as aerospace engineering, mechanical engineering, and environmental science. As a concluding remark, it may be necessary to know information on the curvature of the wave patterns at almost a stagnation point and the detached shock in the stream function. We hope to dwell more of these in subsequent papers.

## References

[1] Andersen, T. L., & Frigard, P. (2011). *Lecture notes for the course in water wave mechanics* (DCE Lecture Notes No. 24). Department of Civil Engineering, Aalborg University. Retrieved from: https://vbn.aau.dk/ws/portalfiles/portal/54776415/Lecture_Notes_for_the_Course_in_Water_Wave_Mechanics.pdf

[2] Apsley, D. (n.d.). Hydraulics 3. *Waves: Linear wave theory*. Retrieved from: https://personalpages.manchester.ac.uk/staff/david.d.apsley/lectures/hydraulics3/WavesLinear.pdf

[3] Çengel, Y. A., & Cimbala, J. M. (2013). *Fluid mechanics*: *Fundamentals and applications*. McGraw-Hill Education. Retrieved from: https://archive.org/details/ed4_20201119

[4] Kundu, P. K., & Cohen, I. M. (2004). *Fluid mechanics* (3rd ed.). Elsevier Academic Press. Retrieved from: https://archive.org/details/fluidmechanics0000kund

[5] Dean, R. G., & Dalrymple, R. A. (1984). *Water wave mechanics for engineers and scientists*. Prentice-Hall Inc.

[6] Dean, R. G., & Dalrymple, R. A. (1991). *Water wave mechanics for engineers and scientists*. World Scientific Publishing Co. Pte Ltd. https://doi.org/10.1142/1232

[7] Donelan, M. A., Curcic, M., Chen, S. S., & Magnusson, A. K. (2012). Modeling waves and wind stress. *J. Geophys. Res.*, *117*, C00J23. https://doi.org/10.1029/2011JC007787

[8] Dysthe, K. B. (2004). Lecture notes on linear wave theory. *Lectures given at the summer school on: Water Waves and Ocean Currents. Nordfjordeid 21-29 June 2004*. Retrieved from: https://www.astro.princeton.edu/~burrows/classes/542/papers/Rui.Nordfjordeid-versjon.pdf

[9] Falnes, J., & Perlin, M. (2003). Ocean waves and oscillating systems: Linear interactions including wave-energy extraction. *Applied Mechanics Reviews*, *56*(1), B3. https://doi.org/10.1115/1.1523355

[10] Fox, R. W., McDonald, A. T., & Pritchard, P. J. (2011). *Introduction to fluid mechanics*. John Wiley & Sons.

[11] Georgi, H. (1993). *The physics of waves*. Harvard University. Prentice Hall.

[12] Green, A. E., Laws, N., & Naghdi, P. M. (1974). On the theory of water waves. *Proc. R. Soc. Lond. A. 338*, 43-55. https://www.jstor.org/stable/78550

[13] Herbich, J. B. (2005). Coastal and ocean engineering. In *Handbook of coastal and ocean engineering* (Chapter XIII). Texas A&M University Consulting and Research, CRC Press LLC. Retrieved from: https://archive.org/details/handbookofcoasta0003unse

[14] Holmes, P. (n.d.). *Professional development programme: Coastal infrastructure design, construction, and maintenance*. Retrieved from: https://www.oas.org/cdcm_train/courses/course21/title_toc.pdf

[15] Husain, Z., Abdullah, M. Z., & Alimuddin, Z. (2008). *Basic fluid mechanics and hydraulic machines*. BS Publications. Retrieved from: https://archive.org/details/Basic_Fluid_Mechanics_and_Hydraulic_Machines

[16] Johnson, R. S. (1997). *A modern introduction to the mathematical theory of water waves*. Cambridge University Press.

[17] Kundu, P. K., & Cohen, I. M. (2002). *Fluid mechanics* (2nd ed.). Academic Press.

[18] Lieberstein, H. M. (1968). *A course in numerical analysis*. Harper and Row.

[19] MIT - Department of Ocean Engineering (2004). *Marine hydrodynamics, Fall 2004. Lecture 19*. Retrieved from: https://web.mit.edu/13.021/demos/lectures/lecture19.pdf

[20] Munson, B. R., Okiishi, T. H., Huebsch, W. W., & Rothmayer, A. P. (2008). *Fluid mechanics*. John Wiley & Sons, Inc. Retrieved from: https://bcs.wiley.com/he-bcs/Books?action=chapter&bcsId=7938&itemId=1118318676&chapterId=87957

[21] Som, S. K., & Biswas, G. (2008). *Introduction to fluid mechanics and fluid machines*. Tata McGraw-Hill Publishing Company Limited. Retrieved from: https://archive.org/details/IntroductionToFluidMechanicsSomBiswas

[22] Stoker, J. J. (1957). *Water waves: The theory with applications*. Interscience Publishers, Inc. Retrieved from: https://archive.org/details/waterwavesthemat033435mbp/page/n5/mode/2u

[23] Whitham, G. B. (1974). *Linear and nonlinear waves*. Wiley-Interscience. Retrieved from: https://archive.org/details/linearnonlinearw0000whit