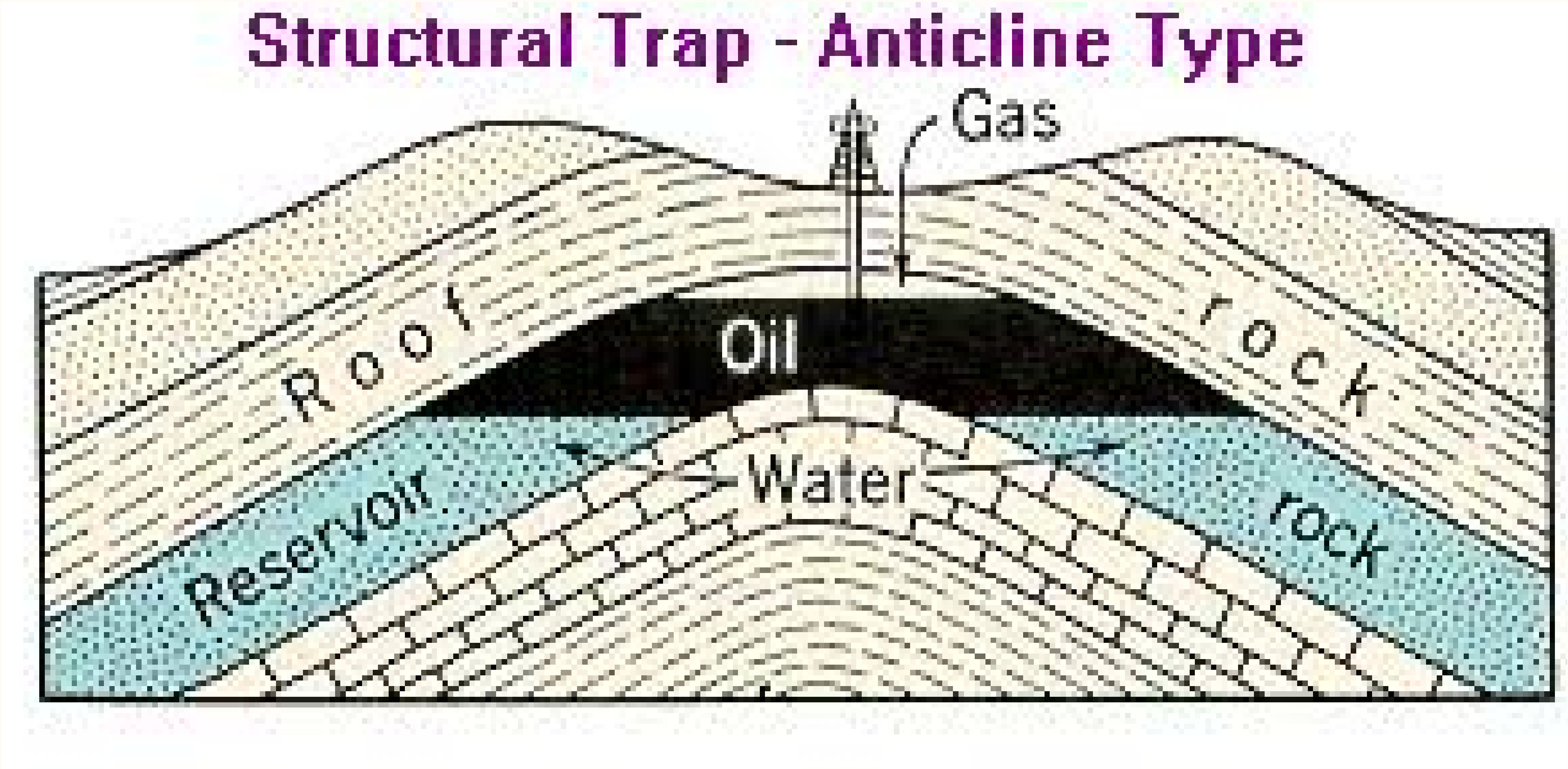
# Modeling of the complex hydrocarbon traps by the shot domain acoustic finite difference method and data-processing

# Introduction

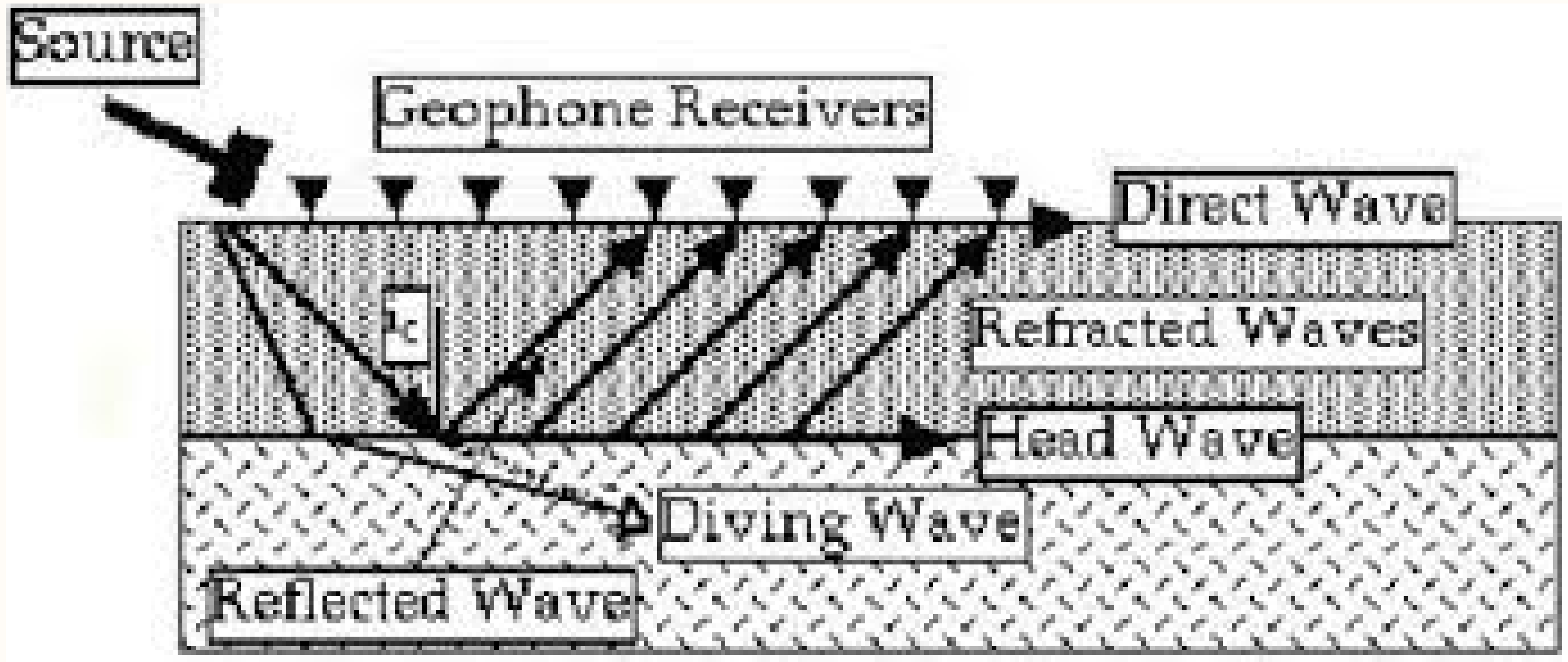
# Complex Hydrocarbon Traps




A geological structure where oil and gas accumulate underground, crucial for petroleum exploration and extraction.

# Introduction to Numerical Modeling in Exploration Seismology

- Simulates seismic waves interacting with subsurface structures.
- Helps understand how hydrocarbon traps appear in seismic data.
- Enables development of effective data processing techniques.



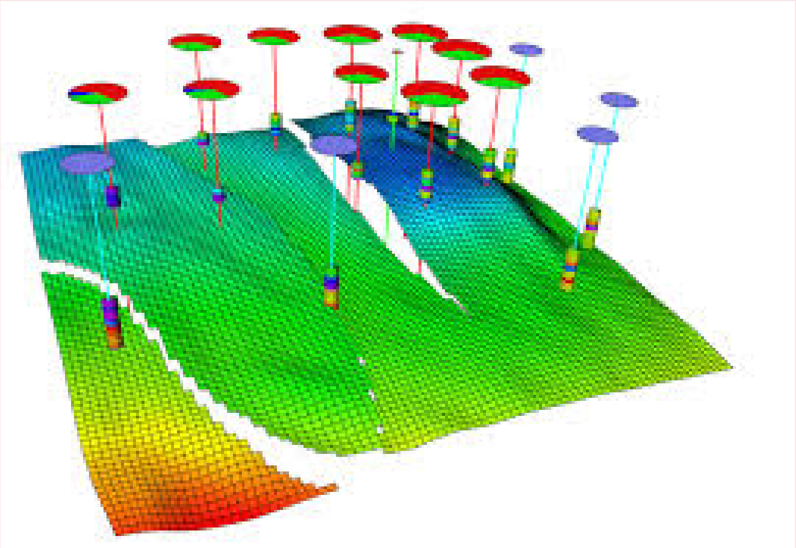
# Acoustic Wave Equation

- Describes sound or seismic wave propagation.
- Derived from fluid mechanics and elasticity theory.
- General form encapsulates wave behavior in the medium.

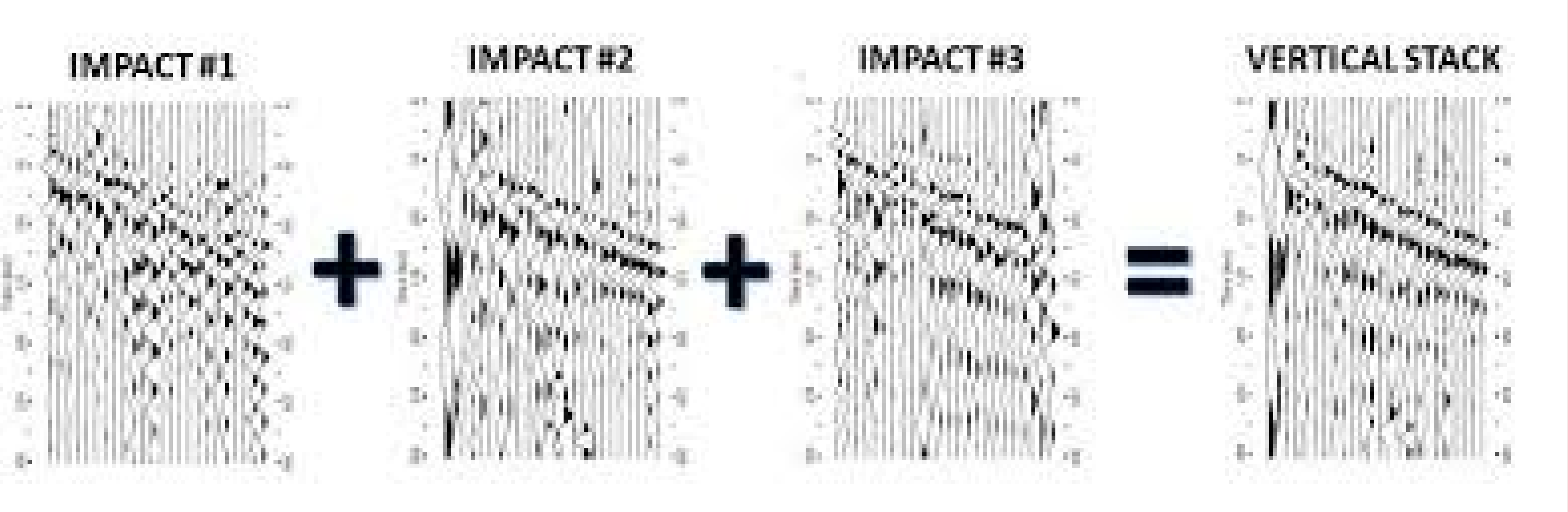
$$\frac{\partial^2 p}{\partial t^2} = v^2 \left( \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} \right)$$

- p is the pressure variation with respect to time and space.
- t is time.
- v is the speed of sound or seismic velocity.

# Modeling Process



Seismic data **simulation.**




**Data processed** to generate stacked sections.

# Data Processing

Stacked sections:

- Combining multiple seismic traces to enhance clarity of subsurface features.
- Improves signal-to-noise ratio.

$$S(t) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} u_i(t)^2}$$

where N is the total number of receivers.

# Determining Trap Locations

## 1. Conversion to Depth

- Converts seismic data from time to depth.
- Represents seismic wave velocity changes with depth.
- Integrates velocity information to convert travel times to depths

## 2. Interpretation

- Locates potential hydrocarbon traps based on seismic reflections.
- Considers structural features and stratigraphic variations.
- Identifies areas with reflections indicating suitable geological structures for trapping hydrocarbons.

# Ricker Wavelet

- Helps in simulating the source.
- The Ricker wavelet shows seismic signals with a clear main frequency and lasts for a specific amount of time.
- Single peak at $t$=$t$peak.
- Amplitude decreases symmetrically as time deviates from peak.

$$A(t) = (1 - 2\pi^2 f^2 t^2) \cdot e^{-\pi^2 f^2 t^2}$$

- f is the central frequency of the wavelet, which determines its frequency content.
- tpeak is the peak time of the wavelet, which controls the time at which the maximum amplitude occurs.
- e is the base of the natural logarithm.
- π is the mathematical constant pi (approximately 3.14159).

# Finite Difference Equation Derivation

Let's derive the finite difference equation to update the pressure field based on the wave equation.

Discretization in Time

Discretization in Space

Combining Discretizations

# Discretization in Time

Second-order central difference scheme for time derivative.

$$\frac{\partial^2 p}{\partial t^2} \approx \frac{p_{i,j,k}^{n+1} - 2p_{i,j,k}^{n} + p_{i,j,k}^{n-1}}{(\Delta t)^2}$$

Where $p_{i,j,k}^{n}$ represents the pressure at grid point $(i, j, k)$ at time step $n$, and $\Delta t$ is the time step.

# Discretization in Space

Central difference approximations for spatial derivatives.

$$\frac{\partial^2 p}{\partial x^2} \approx \frac{p_{i+1,j,k} - 2p_{i,j,k} + p_{i-1,j,k}}{(\Delta x)^2}$$

Similarly, we can approximate the second derivatives with respect to y and z.

# Combining Discretizations

Substituting discretized time and spatial derivatives into the wave equation

$$\frac{p_{i,j,k}^{n+1} - 2p_{i,j,k}^{n} + p_{i,j,k}^{n-1}}{(\Delta t)^2} = v^2 \left( \frac{p_{i+1,j,k} - 2p_{i,j,k} + p_{i-1,j,k}}{(\Delta x)^2} + \frac{p_{i,j+1,k} - 2p_{i,j,k} + p_{i,j-1,k}}{(\Delta y)^2} + \frac{p_{i,j,k+1} - 2p_{i,j,k} + p_{i,j,k-1}}{(\Delta z)^2} \right)$$

# Understanding the Code

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Define parameters
nx = 50  # Number of grid points in x-direction
ny = 50  # Number of grid points in y-direction
nz = 25  # Number of grid points in z-direction
dx = 20  # Grid spacing in x-direction (meters)
dy = 20  # Grid spacing in y-direction (meters)
dz = 20  # Grid spacing in z-direction (meters)
dt = 0.01  # Time step (seconds)
nt = 500  # Number of time steps
vp = 2000  # P-wave velocity (m/s)

# Initialize velocity model (constant velocity)
vel_model = np.ones((nx, ny, nz)) * vp

# Introduce subsurface velocity anomalies
vel_model[0:4, 0:4, 13:15] = vp * 1.2  # Increase velocity in anomaly region
vel_model[20:22, 20:22, 20:21] = vp * 1.5

# Define hydrocarbon trap structure
trap = np.zeros((nx, ny, nz))
trap[0:35, 0:35, 5:10] = 1  # Hydrocarbon trap region
```

The code starts by importing necessary libraries including NumPy for numerical computations and Matplotlib for visualization.

Parameters such as grid size (nx, ny, nz), grid spacing (dx, dy, dz), time step (dt), number of time steps (nt), and P-wave velocity (vp) are defined.

A 3D velocity model (vel_model) is initialized with constant velocity throughout.

Velocity anomalies are introduced in specific regions of the velocity model to simulate subsurface structures.

A hydrocarbon trap structure (trap) is defined

```python
# Initialize wavefield arrays
u = np.zeros((nt, nx, ny, nz))  # Wavefield at current time step
u_prev = np.zeros((nx, ny, nz))  # Wavefield at previous time step
u_next = np.zeros((nx, ny, nz))  # Wavefield at next time step

# Define source function (Ricker wavelet)
def ricker_wavelet(frequency, peak_time, dt, nt):
    t = np.arange(0, nt * dt, dt) - peak_time
    y = (1 - 2 * (np.pi ** 2) * (frequency ** 2) * (t ** 2)) *
     np.exp(-(np.pi ** 2) * (frequency ** 2) * (t ** 2))
    return y

# Define source parameters
source_x = nx // 2  # Source position in x-direction
source_y = ny // 2  # Source position in y-direction
source_z = nz // 4   # Source position in z-direction
frequency = 20       # Source frequency (Hz)
peak_time = 0.05     # Source peak time (seconds)

# Generate source wavelet
source_wavelet = ricker_wavelet(frequency, peak_time, dt, nt)

# Define receiver parameters
num_receivers = 5
receiver_x = np.linspace(0, dx * (nx - 1), num_receivers)
receiver_y = np.linspace(0, dy * (ny - 1), num_receivers)
receiver_z = np.linspace(0, dz * (nz - 1), num_receivers)
```

Arrays to store wavefield at current (u), previous (u_prev), and next (u_next) time steps are initialized.

A Ricker wavelet function (ricker_wavelet) is defined to generate the source wavelet.

Source parameters such as position (source_x, source_y, source_z), frequency (frequency), and peak time (peak_time) are defined.

Source wavelet is generated using the Ricker wavelet function.

Receiver parameters including number of receivers (num_receivers) and their positions (receiver_x, receiver_y, receiver_z) are defined.

```python
# Initialize receiver recordings
receiver_recordings = np.zeros((nt, num_receivers))

# Main time loop (finite difference method)
for i in range(nt):
    # Inject source wavelet
    if i < len(source_wavelet):
        # Ensure we don't access source_wavelet beyond its size
        u[i, source_x, source_y, source_z] = source_wavelet[i]

    # Record seismic wavefield at receivers
    for j in range(num_receivers):
        receiver_recordings[i, j] = u[i, int(receiver_x[j] / dx),
        int(receiver_y[j] / dy), int(receiver_z[j] / dz)]

    # Update wavefield using finite difference method
    u_next[1:-1, 1:-1, 1:-1] = 2 * u[i, 1:-1, 1:-1, 1:-1]
    - u_prev[1:-1, 1:-1, 1:-1] + (vel_model[1:-1, 1:-1, 1:-1] ** 2) * (dt ** 2) * (
            (u[i, 2:, 1:-1, 1:-1] - 2 * u[i, 1:-1, 1:-1, 1:-1]
             + u[i, :-2, 1:-1, 1:-1]) / (dx ** 2) +
            (u[i, 1:-1, 2:, 1:-1] - 2 * u[i, 1:-1, 1:-1, 1:-1]
             + u[i, 1:-1, :-2, 1:-1]) / (dy ** 2) +
            (u[i, 1:-1, 1:-1, 2:] - 2 * u[i, 1:-1, 1:-1, 1:-1]
             + u[i, 1:-1, 1:-1, :-2]) / (dz ** 2))
```

Receiver recordings array (receiver_recordings) is initialized.

Main time loop iterates through each time step
and Source wavelet is injected into the wavefield.

Seismic wavefield is recorded at each receiver.

Finite difference method is used to update the wavefield for the next time step.

```python
    # Apply boundary conditions
    # Bottom boundary condition
    u_next[:, :, -1] = 0

    # Free surface boundary condition
    u_next[:, :, 0] = u_next[:, :, 1]
    # Assuming the derivative normal to the surface is zero

    # Sides boundary condition (assuming periodic boundaries)
    u_next[0, :, :] = u_next[-2, :, :]
    u_next[-1, :, :] = u_next[1, :, :]
    u_next[:, 0, :] = u_next[:, -2, :]
    u_next[:, -1, :] = u_next[:, 1, :]
    u_next[:, :, -1] = u_next[:, :, -2]  # Closing the brackets here

    # Update wavefields for next time step
    u_prev = u[i].copy()  # copy u[i] to u_prev
    u[i] = u_next.copy()

# Plot seismic wavefield with hydrocarbon trap and receiver recordings
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

# Plot velocity anomalies
anomaly_indices = np.where(vel_model > vp)
ax.scatter(anomaly_indices[0] * dx, anomaly_indices[1]
           * dy, anomaly_indices[2] * dz, c= 'green', alpha=0.5)
```

Boundary conditions (bottom, free surface, and sides) are applied to the wavefield.

Seismic wavefields for the next time step are updated.

The code then proceeds to visualize the seismic wavefield and receiver recordings using Matplotlib.

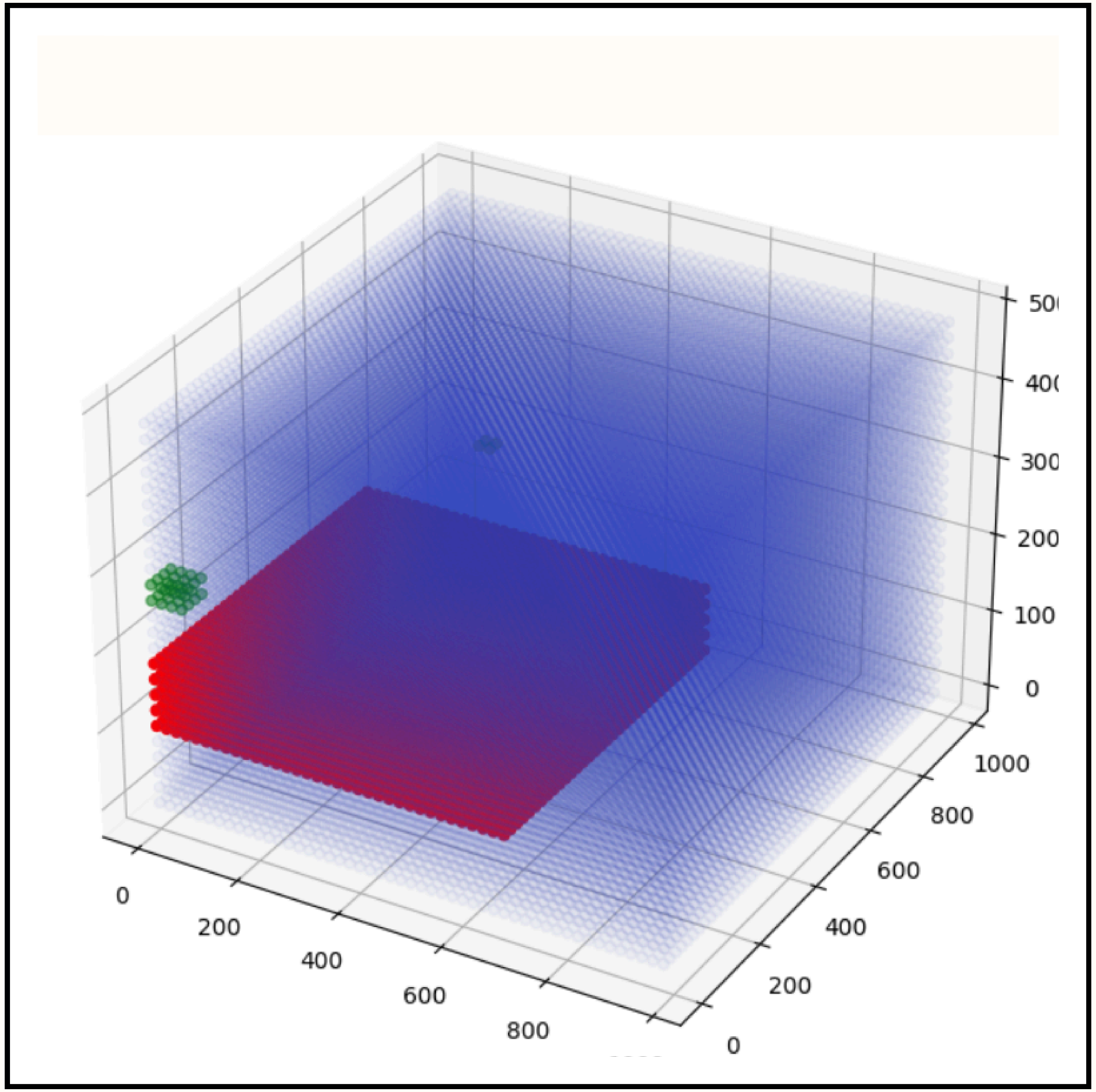
```python
# Plot hydrocarbon trap
trap_indices = np.where(trap == 1)
ax.scatter(trap_indices[0] * dx, trap_indices[1]
           * dy, trap_indices[2] * dz, c='red', alpha=1)

# Plot seismic wavefield
x, y, z = np.meshgrid(np.arange(nx), np.arange(ny), np.arange(nz), indexing='ij')
ax.scatter(x.flatten() * dx, y.flatten()
 * dy, z.flatten() * dz, c=u[-1].flatten(), cmap='coolwarm', alpha=0.05)

# Stack the receiver recordings
stacked_recordings = np.sum(receiver_recordings, axis=1)

# Plot stacked receiver recordings
plt.figure(figsize=(10, 6))
plt.plot(np.arange(nt) * dt, stacked_recordings, color='b')
plt.title('Stacked Receiver Recordings')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.grid(True)
plt.show()
```

Seismic wavefield, velocity anomalies, and hydrocarbon trap are plotted in a 3D space.

Stacked receiver recordings are plotted against time to analyze the seismic data.

Finally, the plots are displayed using plt.show().

# Output



# Numerical Modeling of Seismic Wave Propagation:

# Applications and Significance

## Understanding Seismic Behavior

Understand behaviors with layers, faults, and reservoirs.

## Exploration Seismology

Locate potential hydrocarbon reservoirs underground.

## Modeling Complex Geology

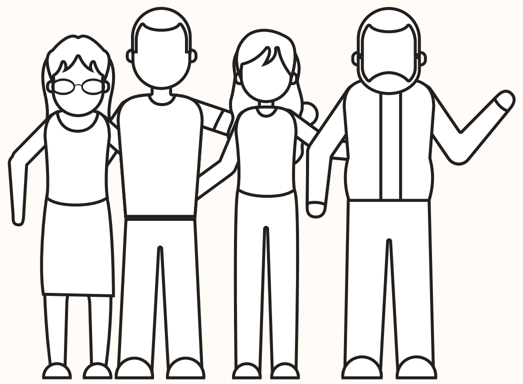
Investigate variations in rock properties and fluid content.

## Testing Data Processing Techniques

Enhance signal quality and geological information extraction.

# References


- Ahmadi, O., Juhlin, C., Malehmir, A., Munck, M. 2013. High-resolution 2D seismic imaging and forward modeling of a polymetallic sulfide deposit at Garpenberg, Central Sweden. Geophysics 78, 339–350.
- Alaei, B. 2006. Seismic Depth Imaging of Complex Structures, an example from Zagros fold thrust belt, Iran. PhD Thesis, University of Bergen.
- Alaei, B., Petersen, S. A. 2007. Geological modeling and finite difference forward realization of a regional section from the Zagros fold-and-thrust belt. Petroleum Geoscience 13, 241–251.
- Alterman, Z., Karal, F. C. 1968. Propagation of elastic waves in layered media by finite-difference methods. Bulletin of the Seismological Society of America 58, 367-398.
- Aminzadeh, F., Brac, J., Kunz, T. 1997. SEG/EAGE 3D Modeling Series No 1. Society of Exploration Geophysicists and the European Association of Geoscientists and Engineers.
- Bansal, R., Sen, M. K. 2008. Finite-difference modelling of S-wave splitting in anisotropic media. Geophysical Prospecting 56, 293-312.



# Our Team

| | | |
|---|---|---|
| 1. | Ayush Bhagat | 21MI10016 |
| 2. | Aryan Luharuwala | 21MI10015 |
| 3. | Bhuneshwar Netam | 21MI31026 |
| 4. | Harsh Agarwal | 21MI10022 |
| 5. | Harshal Yuvraj | 21MI10023 |
| 6. | Pranay Bhaskar | 21MI33015 |
| 7. | Shristi Raushan | 21MI10046 |
| 8. | Naina Sithara | 21MI33013 |

# Thank You