```python
import math
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

A = 850         # Area, m^2
Q = 325         # Flow rate m^3/s
alpha = 200     # Constant m^3/2 / s

# Stores time and water level values to be plotted
time_values = []
y_values = []

# Defines dy/dt
def forcingfunc(time, ylevel):
    return 2 * (Q/A) * math.sin(time)**2  - (alpha/A) * ((1 + ylevel)**(3/2))

def simulate(h, label):
    # Simulation parameters
    t = 0           # Time s
    y = 2           # Initial water level m
    maxTime = 10    # Simulation limit s

    times = []
    levels = []

    while t < maxTime:
        times.append(t)
        levels.append(y)
        y += forcingfunc(t, y) * h
        t += h

    time_values.append(times)
    y_values.append(levels)

    # Print the final time and level
    #print(f"Final Time: {t:.2f}, Final Y-level: {y:.2f}")

def simulate_no_store(h, maxTime):
    # Simulation parameters
    t = 0           # Time s
    y = 2           # Initial water level m

    while t < maxTime:
        y += forcingfunc(t, y) * h
        t += h

    return y

# Run simulations with different step sizes
simulate(1, 'h=1')
simulate(0.1, 'h=0.1')
simulate(0.01, 'h=0.01')
simulate(0.001, 'h=0.001')

# Plotting the water level over time
'''
plt.figure(figsize=(10, 8))
for i, label in enumerate(['h=1', 'h=0.1', 'h=0.01', 'h=0.001']):
    plt.plot(time_values[i], y_values[i], linestyle='-', label=label)
plt.title('Water Level Over Time Using Euler\'s Method')
plt.xlabel('Time (s)')
plt.ylabel('Water Level (m)')
plt.legend()
plt.grid(True)
plt.show()
'''

# Create a DataFrame for the results of the last simulation
data = {
    'Time': time_values[-1],
    'Water level': y_values[-1]
}

#Finding errors across different levels
def findErrors(level):
    errors1 = []
    #take 10 samples from data, second set will be 10 times as dense as the first
```

```
    for i in range(0, 10):
        value1 = y_values[level][i * 10**level]
        value2 = y_values[level + 1][i*10**(level + 1)]
        error = 100 * (value1 - value2)/value2
        errors1.append(error)
    return errors1


errors = {
    'h1/.1' : findErrors(0),
    'h.1/.01' : findErrors(1),
    'h.01/.001' : findErrors(2),
}

#Displaying chart of relative errors
'''
df = pd.DataFrame(data)
edf = pd.DataFrame(errors)
print(edf)
'''

#plotting relative errors

plt.figure(figsize=(10,8))
for i, label in enumerate(['h=1->.1', 'h=0.1->.01', 'h=0.01->.001']):
    plt.plot(range(0,10), findErrors(i),label=label)
plt.title('Relative errors')
plt.xlabel('Time (s)')
plt.ylabel('Percent error')
plt.legend()
plt.grid(True)
plt.show()

#displaying water level estimate in terms of h
print("Displaying y estimate in terms of h")
h_vals  = np.arange(.0001 , 1, .0001) #values from 0.001 to 1 in .05 increments
y_vals_at_2 = []

for h in h_vals:
    y_vals_at_2.append(simulate_no_store(h, 2))

#finding erors
errors = [0]
for i in range(1, len(y_vals_at_2)):
    errors.append(100 * (y_vals_at_2[i] - y_vals_at_2[i-1])/y_vals_at_2[i-1] )

plt.figure(figsize =(10,8))
plt.plot(h_vals, errors)
plt.xlabel('Time step size')
plt.ylabel('Relative error percent')
plt.grid(True)
plt.title("Relative error at t=2 for different h")
plt.show()
```