Numerical Simulation and Analysis of Couette Flow with Oscillating Plates

https://github.com/Blink29/Numerical-Techniques → Term Paper

Name: Paurush Kumar **Roll Number: NA22B002**

Abstract

The goal of this study is to investigate Couette flow caused by oscillating plates using numerical simulation. The velocity field and shear stress distributions are analyzed under varying boundary conditions, fluid viscosities, and oscillation frequencies. Using the FTCS numerical scheme, we compare the effects of in-phase and out-of-phase oscillations, as well as the behavior of Newtonian and non-Newtonian fluids. Convergence analysis is performed to validate the stability and accuracy of the numerical solutions. The results highlight the significant impact of fluid properties and oscillation conditions on the velocity and shear stress distributions, offering insights into applications such as lubrication and oscillatory flow mechanics.

Introduction

Couette flow, the motion of a viscous fluid between two parallel plates, is fundamental in engineering, particularly in applications involving lubrication, machinery oscillation, and flow stability. In this study, we simulate Couette flow where one or both plates oscillate periodically, inducing a velocity field in the fluid.

The primary objective is to examine how boundary conditions, fluid properties, and oscillation patterns affect the velocity distribution and shear stress. The governing equation for the velocity field is the diffusion equation, solved using the FTCS numerical scheme. Special cases, such as non-Newtonian fluids and different oscillatory patterns, are also considered. This study demonstrates how these parameters influence flow behavior and provides valuable insights into engineering applications.

Governing Equations and Numerical Methodology

Governing Equation

The velocity field is governed by the diffusion equation:

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial z^2}$$

where:

- $u(z,t)$: velocity at position z and time t,
- ν: kinematic viscosity of the fluid.

Boundary Conditions

1. **Oscillating lower plate**:

$$u(0,t) = U \sin\left(\frac{2\pi t}{T}\right)$$

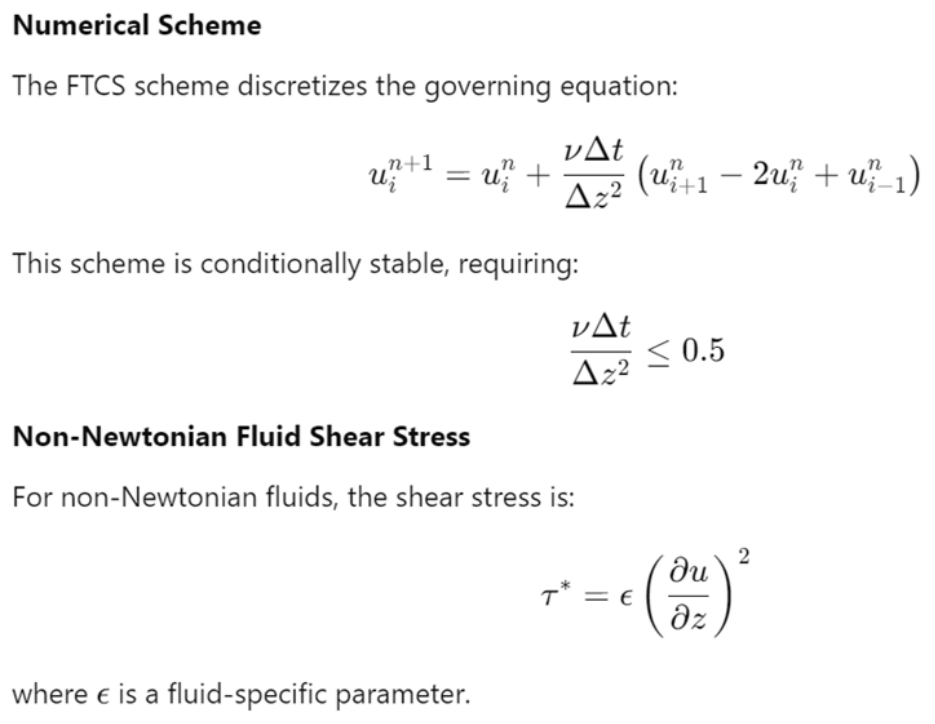
2. **Upper Plate Conditions:**

   - Stationary ($u(H,t) = 0$),
   - In-phase oscillation ($u(H,t) = \frac{U}{2}\sin\left(\frac{2\pi t}{T}\right)$),
   - Out-of-phase oscillation ($u(H,t) = \frac{U}{2}\sin\left(\frac{2\pi t}{T} + \pi\right)$).

For a Newtonian fluid, the shear stress at any point is calculated as:

$$\tau = \nu \frac{\partial u}{\partial z}$$

This is applied at both the oscillating plate (lower plate) and at other locations (e.g., halfway between the plates).



Numerical Scheme

The FTCS scheme discretizes the governing equation:

$$u_i^{n+1} = u_i^n + \frac{\nu \Delta t}{\Delta z^2} \left(u_{i+1}^n - 2u_i^n + u_{i-1}^n \right)$$

This scheme is conditionally stable, requiring:

$$\frac{\nu \Delta t}{\Delta z^2} \leq 0.5$$

Non-Newtonian Fluid Shear Stress

For non-Newtonian fluids, the shear stress is:

$$\tau^* = \epsilon \left(\frac{\partial u}{\partial z} \right)^2$$

where ϵ is a fluid-specific parameter.

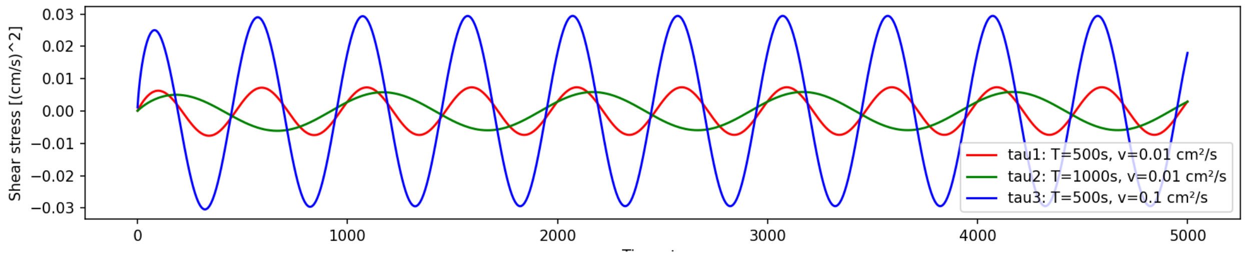
Input Data

- **Maximum Velocity of Lower Plate (U)**: 1 cm/s
- **Oscillation Period (T)**: 500 s
- **Alternate Oscillation Period (T1)**: 1000 s
- **Kinematic Viscosity (v)**: 0.01 cm²/s
- **Alternate Kinematic Viscosity (v1)**: 0.1 cm²/s
- **Non-Newtonian Fluid Constant (ε)**: 0.1
- **Distance Between Plates (H)**: 19 cm
- **Vertical Discretization Step (Δz)**: 1 cm
- **Time Step (Δt)**: 1 s
- **Number of Vertical Discretization Layers (nm)**: 19
- **Number of Time Steps (tm)**: 5000

- **Initial Velocity Distribution (u(z, t=0))**: 0 cm/s (at all vertical layers)
- **Boundary Condition at Lower Plate (u(z=0, t))**: Usin(2πt/T)
- **Boundary Condition at Upper Plate (u(z=H, t))**: 0 cm/s

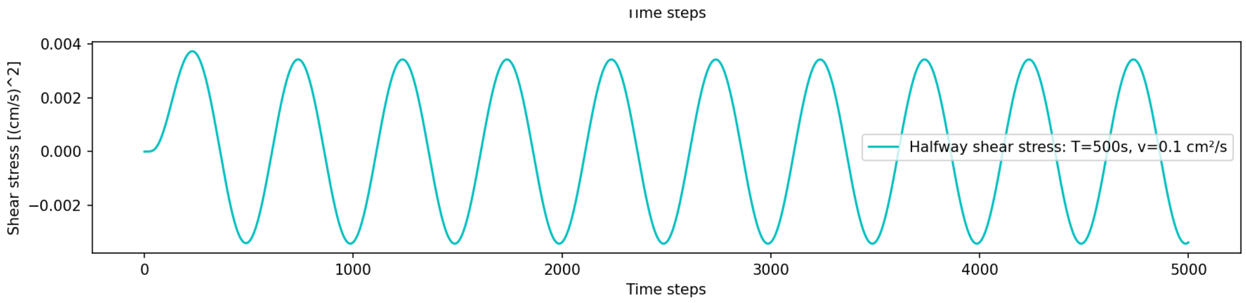
Results

1 . Shear Stress Adjacent to Oscillating Plate



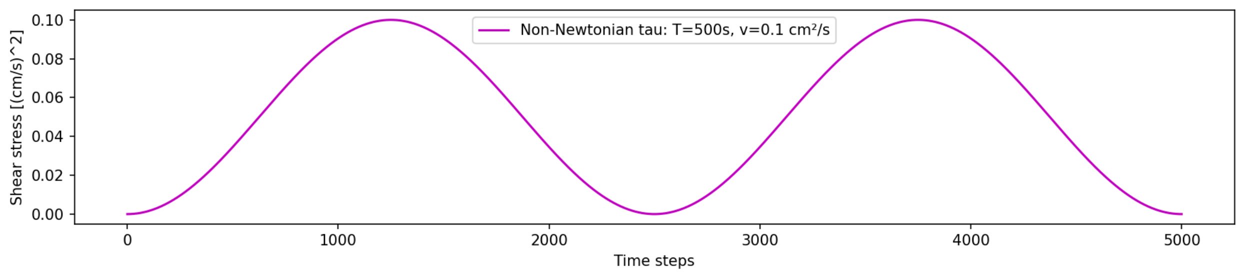
- tau1 (T=500s, v=0.01): Shows moderate amplitude oscillations
- tau2 (T=1000s, v=0.01): Lower frequency due to longer period T
- tau3 (T=500s, v=0.1): Larger amplitude due to higher viscosity v1
- All curves oscillate due to sinusoidal plate motion

2. **Halfway Shear Stress**



- Lower amplitude than wall shear stress
- Shows diffusion of momentum into fluid

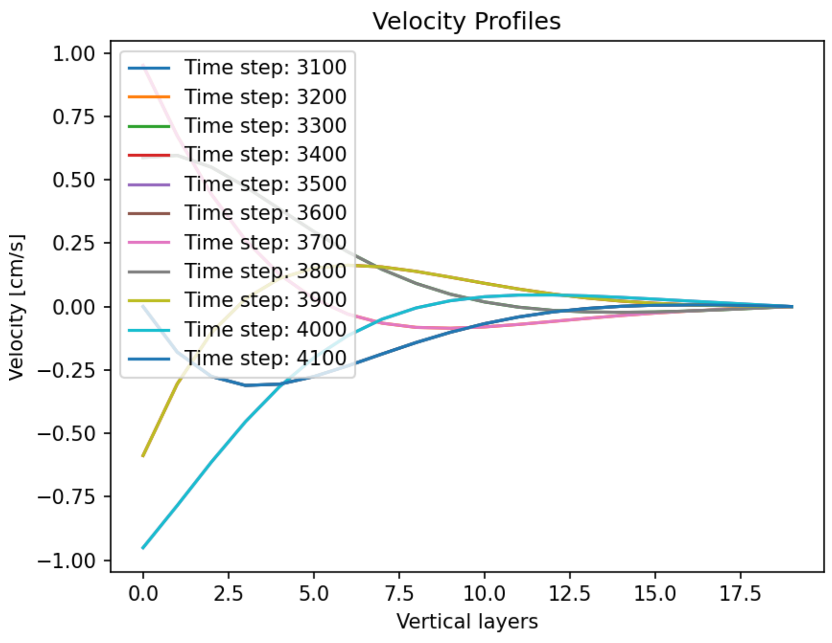
3. **Non-Newtonian Shear Stress**



- Shows squared dependence on velocity gradient ($\tau = \varepsilon(du/dz)^2$)
- Always positive due to squaring
- Double frequency of plate oscillation
- Amplitude modulated by epsilon=0.1

4.

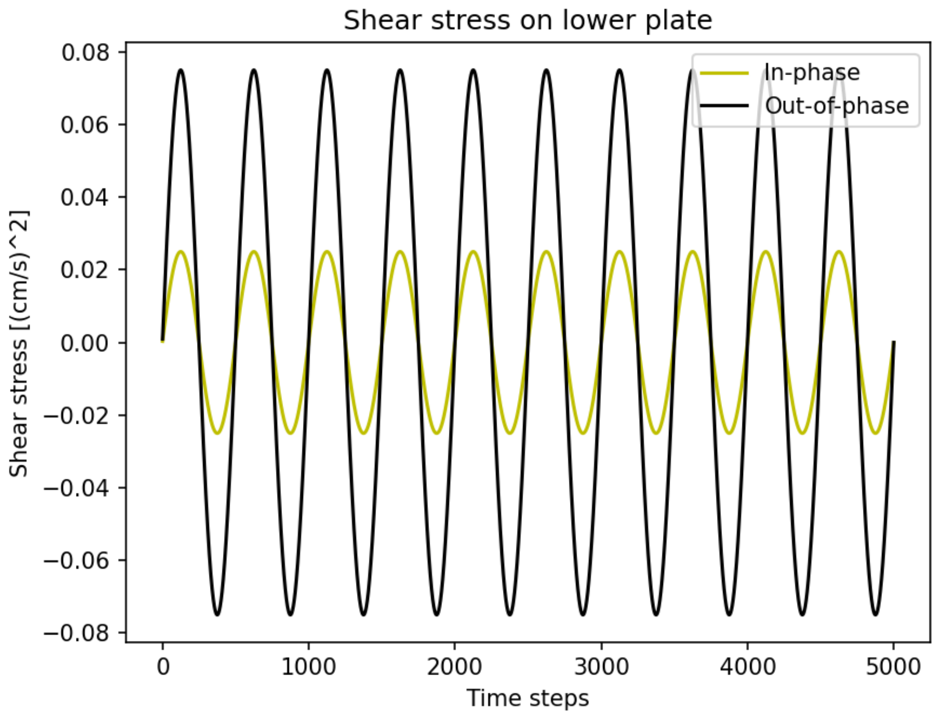
Velocity Profiles



- Show development of boundary layer
- Decay of velocity amplitude with distance from plate

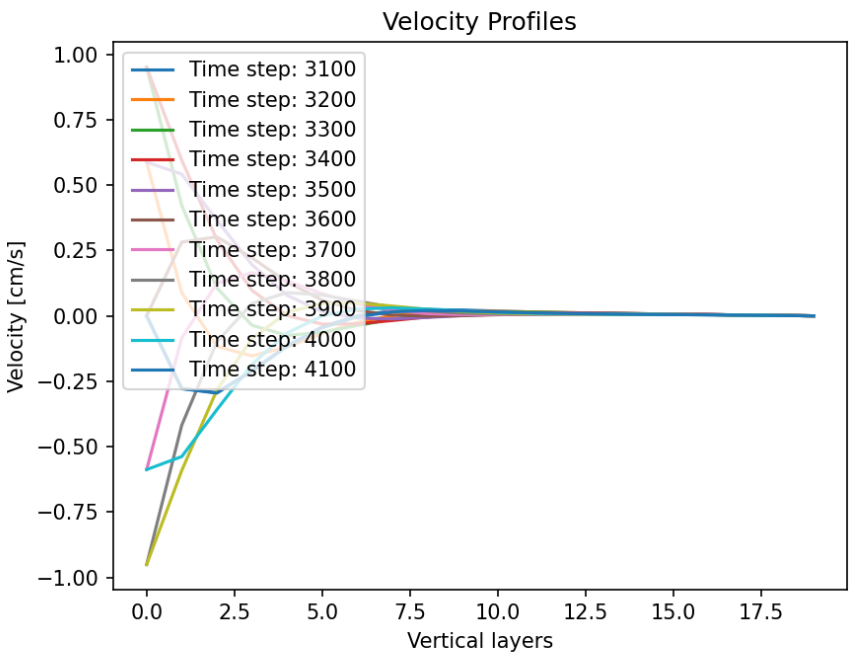
5.

Lower Plate Stresses



- In-phase: Shows constructive interference
- Out-of-phase: Shows destructive interference
- Demonstrates effect of upper plate motion on stress distribution

6. **Convergence**



When I change the dz = 2 and dt = 0.5

- Effectively making the grid coarser. This would typically reduce the resolution of the velocity profile, meaning fewer points to capture the gradients in the fluid's velocity near the wall.
- The fact that the velocity profiles converge quickly when Δz is increased and Δt is decreased might indicate that the fluid dynamics are relatively simple or exhibit periodic behavior that stabilizes rapidly.
- The fact that the velocity profiles converge very quickly in your case suggests that the system might be less sensitive to the vertical discretization size in this particular simulation, possibly due to the nature of the flow (like if it's a relatively smooth flow or the changes in velocity are not as steep).
- By reducing the time step (Δt) from 1 to 0.5, you're effectively increasing the temporal resolution of the simulation. This allows the simulation to capture finer details of the oscillating fluid motion and shear stresses.
- In our case, despite reducing Δt, the velocity profiles still converge quickly, which indicates that the system has reached a steady oscillatory state faster than expected.

The quick convergence of velocity profiles when you increase Δz to 2 and decrease Δt to 0.5 suggests that the system is inherently stable and the numerical scheme is able to capture the primary behavior of the oscillating fluid.

Code

```
import numpy as np
import matplotlib.pyplot as plt

# Input data
U = 1 # Velocity amplitude [cm/s]
T = 500 # Oscillation period [s]
T1 = 1000 # Second oscillation period [s]
v = 0.01 # Kinematic viscosity [cm^2/s]
v1 = 0.1 # Second kinematic viscosity [cm^2/s]
epsilon = 0.1 # Non-Newtonian fluid constant
Dz = 2 # Vertical step [cm]
Dt = 0.5 # Time step [s]
nm = 19 # Number of vertical discretization layers
im = nm + 1 # Number of grid points
tm = 5000 # Number of time steps

# Initialization
u1 = np.zeros(im)
u2 = np.zeros(im)
u3 = np.zeros(im)
tau1 = []
tau2 = []
tau3 = []
half_shear = []
velocity_distributions = []

# Main simulation
for k in range(1, tm + 1):
    Dtt = Dt * k
    # Boundary conditions
    u1[0] = U * np.sin(2 * np.pi * Dtt / T)
    u2[0] = U * np.sin(2 * np.pi * Dtt / T1)
    u3[0] = U * np.sin(2 * np.pi * Dtt / T)
    u3[im-1] = 0  # Upper plate stationary for u3

    # Velocity calculation
    for k1 in range(1, nm):
        u1[k1] = u1[k1] + (v * Dt / Dz**2) * (u1[k1 + 1] - 2
* u1[k1] + u1[k1 - 1])
        u2[k1] = u2[k1] + (v * Dt / Dz**2) * (u2[k1 + 1] - 2
* u2[k1] + u2[k1 - 1])
        u3[k1] = u3[k1] + (v1 * Dt / Dz**2) * (u3[k1 + 1] - 2
* u3[k1] + u3[k1 - 1])

    # Shear stress calculations
    tau1.append((v * (u1[0] - u1[1]) / Dz))
    tau2.append((v * (u2[0] - u2[1]) / Dz))
    tau3.append((v1 * (u3[0] - u3[1]) / Dz))

    # Halfway shear stress calculation
    half_shear.append((v1 * (u3[nm // 2] - u3[nm // 2 + 1]) /
Dz))

    # Store velocity profiles for every 100 time steps betwee
n 3000 and 4000
    if 3000 <= k <= 4000 and k % 100 == 0:
        velocity_distributions.append(u3.copy())

# Non-Newtonian fluid shear stress
non_newtonian_tau = []
t = np.linspace(0, T, tm)

for k in range(tm):
    # Plate velocity at time t
    plate_velocity = U * np.sin(2 * np.pi * t[k] / T)
    # Velocity gradient near the wall (difference between plate and adjacent fluid)
    velocity_gradient = (plate_velocity - 0) / Dz  # Assuming fluid at rest initially

    # Non-Newtonian shear stress
    tau = epsilon * (velocity_gradient)**2
    non_newtonian_tau.append(tau)

# Upper Plate Scenarios:
# (a) Half the velocity, in phase
upper_u_in_phase = [0.5 * U * np.sin(2 * np.pi * Dt * k / T) for k in range(1, tm + 1)]
# (b) Half the velocity, out of phase (180-degree phase shift)
upper_u_out_phase = [-0.5 * U * np.sin(2 * np.pi * Dt * k / T) for k in range(1, tm + 1)]

lower_plate_tau_in_phase = [(0.05 * (U * np.sin(2 * np.pi * Dt * k / T) - upper_u_in_phase[k - 1]) / Dz) for k in range(1, tm + 1)]
lower_plate_tau_out_phase = [(0.05 * (U * np.sin(2 * np.pi * Dt * k / T) - upper_u_out_phase[k - 1]) / Dz) for k in range(1, tm + 1)]

# Plotting results
plt.figure(figsize=(12, 8))

# Shear stress adjacent to the oscillating plate
plt.subplot(3, 1, 1)
plt.plot(range(tm), tau1, 'r', label='tau1: T=500s, v=0.01 cm²/s')
plt.plot(range(tm), tau2, 'g', label='tau2: T=1000s, v=0.01 cm²/s')
plt.plot(range(tm), tau3, 'b', label='tau3: T=500s, v=0.1 cm
²/s')
plt.xlabel('Time steps')
plt.ylabel('Shear stress [(cm/s)^2]')
plt.legend()

# Halfway shear stress
plt.subplot(3, 1, 2)
plt.plot(range(tm), half_shear, 'c', label='Halfway shear str
ess: T=500s, v=0.1 cm²/s')
plt.xlabel('Time steps')
plt.ylabel('Shear stress [(cm/s)^2]')
plt.legend()

# Non-Newtonian fluid shear stress
plt.subplot(3, 1, 3)
plt.plot(range(tm), non_newtonian_tau, 'm', label='Non-Newton
ian tau: T=500s, v=0.1 cm²/s')
plt.xlabel('Time steps')
plt.ylabel('Shear stress [(cm/s)^2]')
plt.legend()

plt.tight_layout()
plt.show()

# Plot velocity profiles
for i, profile in enumerate(velocity_distributions, start=1):
    plt.plot(profile, label=f'Time step: {3000 + i * 100}')
plt.xlabel('Vertical layers')
plt.ylabel('Velocity [cm/s]')
plt.title('Velocity Profiles')
plt.legend()
plt.show()

# Lower plate stresses
plt.plot(range(tm), lower_plate_tau_in_phase, 'y', label='In-
phase')
```

```
plt.plot(range(tm), lower_plate_tau_out_phase, 'k', label='Ou
t-of-phase')
plt.xlabel('Time steps')
plt.ylabel('Shear stress [(cm/s)^2]')
plt.legend()
plt.title('Shear stress on lower plate')
plt.show()
```