import numpy as np

import matplotlib.pyplot as plt

# === 맥스웰 함수 정의 ===

def maxwell\_scalar(mach, base\_value):

"""

맥스웰 수를 기반으로 스칼라 값을 계산하는 함수.

- mach: 맥스웰 수

- base\_value: 기준 값 (스칼라 값의 초기값)

- 반환값: base\_value \* mach^2

"""

return base\_value \* mach\*\*2

# === 물리적 상수 및 초기 조건 ===

mach = 0.3 # 맥스웰 수

c\_sound = 343.0 # 음속 (m/s), 공기 중 기준

# 모든 스칼라 값은 맥스웰 함수로 정의

rho = maxwell\_scalar(mach, 1.0) # 유체 밀도 (kg/m^3)

mu = maxwell\_scalar(mach, 0.1) # 점성 계수 (Pa·s)

u\_top = mach \* c\_sound # 위쪽 경계 속도

nx, ny = 41, 41 # 격자 크기 (nx × ny)

lx, ly = 2.0, 2.0 # 도메인 크기 (x, y 방향 길이)

dx, dy = lx / (nx - 1), ly / (ny - 1) # 격자 간격

nt = 500 # 시간 스텝 수

dt = maxwell\_scalar(mach, 0.01) # 시간 간격 (초 단위)

# === 격자와 초기 필드 설정 ===

x = np.linspace(0, lx, nx)

y = np.linspace(0, ly, ny)

X, Y = np.meshgrid(x, y)

u = np.zeros((ny, nx)) # x 방향 속도 필드

v = np.zeros((ny, nx)) # y 방향 속도 필드

p = np.zeros((ny, nx)) # 압력 필드

# === 경계 조건 함수 ===

def apply\_boundary\_conditions(u, v, p):

"""

격자 경계에 물리적 조건을 적용합니다.

- u[0, :]: 아래 경계 (속도 0, 벽 조건)

- u[-1, :]: 위 경계 (맥스웰 수에 따른 속도 설정)

- v[:, 0], v[:, -1]: 좌우 경계에서 수직 속도 0

"""

u[0, :] = 0 # 아래 경계: 속도 0

u[-1, :] = u\_top # 위 경계: 맥스웰 수 기반 속도

v[:, 0] = 0 # 왼쪽 경계: 수직 속도 0

v[:, -1] = 0 # 오른쪽 경계: 수직 속도 0

return u, v, p

# === 나비에-스토크스 계산 함수 ===

def navier\_stokes(u, v, p, rho, mu, dx, dy, dt, nt):

"""

나비에-스토크스 방정식을 풀기 위한 함수.

"""

for \_ in range(nt):

un = u.copy()

vn = v.copy()

pn = p.copy()

# x 방향 운동량 방정식 (속도 u)

u[1:-1, 1:-1] = (

un[1:-1, 1:-1]

- un[1:-1, 1:-1] \* dt / dx \* (un[1:-1, 1:-1] - un[1:-1, :-2])

- vn[1:-1, 1:-1] \* dt / dy \* (un[1:-1, 1:-1] - un[:-2, 1:-1])

+ mu \* dt / dx\*\*2 \* (un[1:-1, 2:] - 2 \* un[1:-1, 1:-1] + un[1:-1, :-2])

+ mu \* dt / dy\*\*2 \* (un[2:, 1:-1] - 2 \* un[1:-1, 1:-1] + un[:-2, 1:-1])

)

# y 방향 운동량 방정식 (속도 v)

v[1:-1, 1:-1] = (

vn[1:-1, 1:-1]

- un[1:-1, 1:-1] \* dt / dx \* (vn[1:-1, 1:-1] - vn[1:-1, :-2])

- vn[1:-1, 1:-1] \* dt / dy \* (vn[1:-1, 1:-1] - vn[:-2, 1:-1])

+ mu \* dt / dx\*\*2 \* (vn[1:-1, 2:] - 2 \* vn[1:-1, 1:-1] + vn[1:-1, :-2])

+ mu \* dt / dy\*\*2 \* (vn[2:, 1:-1] - 2 \* vn[1:-1, 1:-1] + vn[:-2, 1:-1])

)

# 압력 필드 갱신 (연속 방정식 단순화)

p[1:-1, 1:-1] = (

pn[1:-1, 1:-1]

- rho \* (dt / dx \* (u[1:-1, 2:] - u[1:-1, :-2])

+ dt / dy \* (v[2:, 1:-1] - v[:-2, 1:-1]))

)

# 경계 조건 적용

u, v, p = apply\_boundary\_conditions(u, v, p)

return u, v, p

# === 시뮬레이션 수행 ===

u, v, p = navier\_stokes(u, v, p, rho, mu, dx, dy, dt, nt)

# === 결과 시각화 ===

plt.figure(figsize=(11, 7))

# 압력 분포 시각화

plt.contourf(X, Y, p, alpha=0.5, cmap="viridis")

plt.colorbar(label="Pressure")

# 속도 벡터 필드 시각화

plt.quiver(X, Y, u, v, color="red")

plt.xlabel("X")

plt.ylabel("Y")

plt.title("Velocity Field and Pressure Contours")

plt.show()