```python
# -*- coding: utf-8 -*-
"""
Created on Tue Sep 30 13:52:57 2025

@author: coled
"""

# ICF HW6 Problem 3

# Part a: Numerically solve for length until M = 1

import numpy as np
import matplotlib.pyplot as plt


def dm_dx(M, f, x, gamma=1.4):

    c1 = (1 + (gamma-1)*0.5 * M **2) / ( (M**2 - 1))
    c2 = (gamma*M**2) * f / ( np.exp(1/(x+1)))
    c3 = -2 / ((x+1)**2)
    return M * c1 * (c3-c2)


dx = 1.0e-3  # m
xmin, xmax = 0.0, 100.0
n_samples = int((xmax - xmin)/dx)
x = np.linspace(xmin, xmax, n_samples)
M = np.zeros_like(x)
M[0] = 6.0
f = 0.005
tol = 1e-10

# stop when choked
def find_choke(M, x, f, tol=1e-3):
    Mi = M.copy()
    i_break = len(x) - 1
    for i in range(len(x) - 1):
        Mi[i+1] = Mi[i] + (x[i+1] - x[i]) * dm_dx(Mi[i], f, x[i])
        if (Mi[i+1] <= 1.0) or (abs(Mi[i+1] - 1.0) < tol):
            i_break = i + 1
            break
    return x[i_break], Mi[:i_break+1], i_break

L1, M_array, i_break = find_choke(M, x, f)

# ---------------- part (b)  ----------------
T0 = 600.0            # K
p0 = 200.0e3          # Pa
R  = 287.0            # J/(kg*K)

x2 = x[:i_break+1]
n2 = len(x2)

u   = np.zeros_like(x2)
A   = np.zeros_like(x2)
```

```
T   = np.zeros_like(x2)
rho = np.zeros_like(x2)
p   = np.zeros_like(x2)
p0_array = np.zeros_like(x2)
T0_array = np.zeros_like(x2)

def pressure_isen(M, gamma=1.4):  # p/p0
    return (1.0 + 0.5*(gamma-1.0)*M**2)**(-gamma/(gamma-1.0))

def flow_property_functions(M, T0=T0, p0=p0, R=R, gamma=1.4):
    c1 = 1.0 + 0.5*(gamma-1.0)*M**2
    T = T0 / c1
    u = M * np.sqrt(gamma*R*T)
    return {"temp": T, "u": u}

# set mass-flow constant using inlet state
props0 = flow_property_functions(M_array[0])
A_init   = np.pi * np.exp(2.0)                  # A(x=0) = pi * e^{2/(0+1)}
u_init   = props0["u"]
p_init   = p0 * pressure_isen(M_array[0])
T_init   = props0["temp"]
rho_init = p_init / (R * T_init)
rho_const = rho_init * u_init * A_init          # m_dot = const

# loop for fields + residence time (using local dx)
time = 0.0
dx_local = x2[1] - x2[0] if len(x2) > 1 else dx
for i in range(n2):
    props = flow_property_functions(M_array[i])
    T[i] = props["temp"]
    T0_array[i] = T0
    u[i] = props["u"]
    A[i] = np.pi * np.exp(2.0 / (x2[i] + 1.0))
    rho[i] = rho_const / (u[i] * A[i])
    p[i] = rho[i] * R * T[i]
    p0_array[i] = p[i] / pressure_isen(M_array[i])

    time += dx_local / u[i]

# ---------------- plots ----------------
fig1, axs = plt.subplots(2, 3, figsize=(11, 6), constrained_layout=True)

axs[0,0].plot(x2, M_array);        axs[0,0].set_title('Mach Number');          axs[0,0].set_xlabel(
axs[0,1].plot(x2, p);              axs[0,1].set_title('Pressure [Pa]');        axs[0,1].set_xlabel(
axs[0,2].plot(x2, p0_array);       axs[0,2].set_title('Stagnation Pressure');  axs[0,2].set_xlabel(
axs[1,0].plot(x2, T0_array);       axs[1,0].set_title('Stagnation T [K]');     axs[1,0].set_xlabel(
axs[1,1].plot(x2, T);              axs[1,1].set_title('Static T [K]');         axs[1,1].set_xlabel(
axs[1,2].plot(x2, np.zeros_like(x2));              axs[1,2].set_title('Free space!');            axs

plt.show()

print(f"Choke length L = {L1:.6f} m")
print(f"Residence time to choke = {time:.6f} s")
```