```python
# -*- coding: utf-8 -*-
"""
Created on Sun Sep 28 17:21:02 2025

@author: coled_agkeohi
"""

import numpy as np

def F_M(M, gamma = 1.4):
   term1 =  (-1.0/(gamma* M**2))
   c1= (gamma+1) / (2 * gamma)
   return  term1 - c1*np.log(M**2 / (1+ 0.5*(gamma-1)*M**2))

f = 0.005
D = 0.02
L = 0.64
fanno_LHS = 4 * f / D
M1 = 2.5
M2  = 2
p1 =  70
T1 = 310


# part a : location of M = 2

F_M1 = F_M(M1)
F_M2 = F_M(M2)

x2 = (F_M2 - F_M1) / fanno_LHS

# part b : find p3

def T_ratio_Fanno(M_up, M_down, gamma = 1.4):

    num = 2 + (gamma-1)*M_up**2.0
    den = 2 + (gamma-1)*M_down**2.0
    return num/den

def P_ratio_fanno(M_up, M_down, gamma = 1.4):
    return (M_up/M_down) * T_ratio_Fanno(M_up, M_down) ** 0.5


def P_ratio_normal_shock(M, gamma = 1.4):
    return (2 * gamma * M**2 - (gamma - 1))/(gamma+1)

p2 = p1 * P_ratio_fanno(M1, M2)

p3 = p2 * P_ratio_normal_shock(M2)

# part c: first determine if it chokes or not

def choke_or_nah(M, x1, L, gamma = 1.4):
    x2 = (F_M(1) - F_M(M)) / (fanno_LHS) + x1
    if (( (F_M(1) - F_M(M)) / (fanno_LHS) + x1) > L):
        return "not gonna choke", x2
```

```
    else:
        return "gonna choke", x2
def mach_normal_shock(M, gamma = 1.4):
    num = (gamma - 1) * M**2 + 2
    den = 2 * gamma* M**2 - (gamma-1)
    return (num/den)**0.5

M3 = mach_normal_shock(M2)
chokes, x4 = choke_or_nah(M3, x2, L)

# now back out the mach number at the exit knowing where it would choke

# classic bisection time

def solve_for_M_exit(x_sonic, L, M_in, tol = 1e-10, maxit = 200):
    F_target = -1.0*(x_sonic-L-F_M(1))
    a = M_in # we know it will creep to sonic
    b = 1.0
    def g(M):
        return F_M(M) - F_target

    ga, gb = g(a), g(b)
    for i in range(maxit):
        c=0.5*(a+b)
        gc = g(c)
        if abs(gc) < tol or 0.5*(b - a) < tol:
            return c
        if ga*gc <= 0.0:
            b, gb = c, gc
        else:
            a, ga = c, gc
    return 0.5*(a + b)

M_exit = solve_for_M_exit(x4, L, M3)




# part d: temeprature at duct exit
def T_normal_shock(M, gamma = 1.4):
    num  = (2*gamma*M**2 - (gamma - 1)) * ((gamma-1)* M**2 +2)
    den = (gamma+1)**2 * M**2
    return num/den

T_exit  = T1 * T_ratio_Fanno(M1, M2) * T_normal_shock(M2) * T_ratio_Fanno(M3, M_exit)
```