

hw__6

December 14, 2025

1 AEROSP 520 Compressible Flow: Homework 6

Jason Chen

1.1 Problem 3

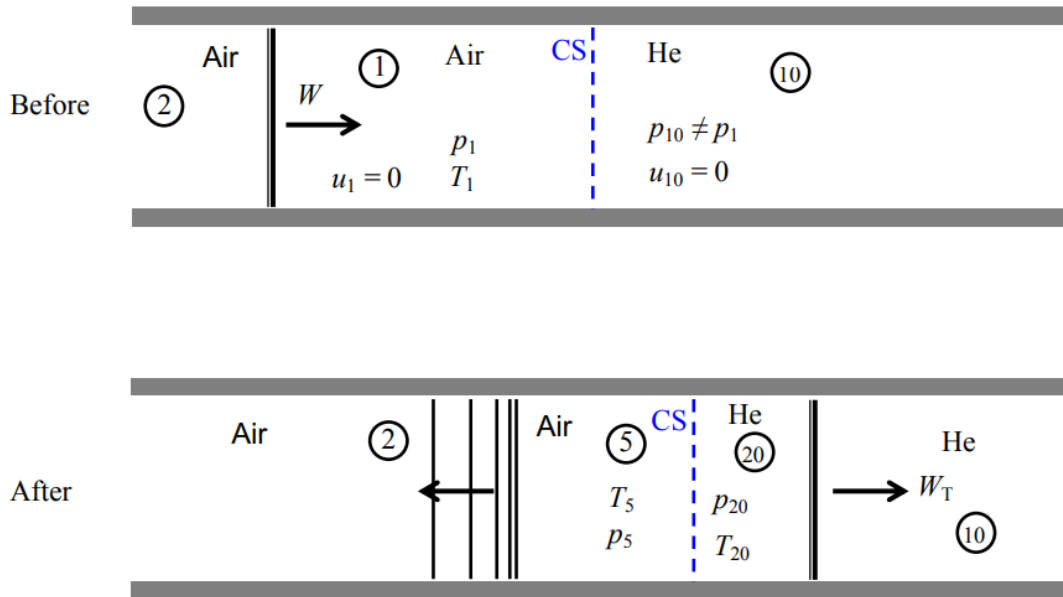


FIGURE 1. Figure for Problem 3

Assume region (1) contains air at $p_1 = 50$ Torr and $T_1 = 300$ K, and region (10) contains helium at $p_{10} = 25$ Torr and $T_{10} = 300$ K. The speed of the primary shock is initially $W = 1000$ m/s.

1.1.1 Part (a)

Prompt: *Indicate what are the conditions across the contact surface that define regions 5 and 20, and hence the final solution of the problem.*

As with any material discontinuity, the velocity magnitude and static pressure on either side of the boundary must be equal, or else there would be transport across the boundary. Thus:

$$u_5 = u_{20}, \quad p_5 = p_{20}$$

1.1.2 Part (b)

Prompt: *Find the pressure, temperature and speed behind the secondary moving shock.*

We are interested in the properties at region 20, after W_T . However, since we do not know the properties of W_T , our solution strategy will be to solve for the LHS of the resultant contact surface (CS) to then solve for region 20. Since we know that region 5 is the result of firstly region 1 being shocked by the incident wave W (resulting in region 2) and then expanded, we will initially solve for the properties of region 2. Since W is a moving shock, we can shift our reference frame to that of the moving shock such that it is stationary, and it is as if there is a 1000 m/s flow (of region 1) shocking and gaining static pressure and temperature as it transforms to region 2.

Thus, we can use the elementary shock jump relations to solve for region 2. The Mach number of the “flow” in region 1 is:

$$M_1 = \frac{W}{a_1} = \frac{W}{\sqrt{\gamma R T_1}}$$

This assumes a CPG, which is valid at these moderate temperatures and pressures. We will use $\gamma = 1.4$ and $R = 287 \text{ J/kg/K}$ for air. Then, we can find the static pressure jump in region 2:

$$\frac{p_2}{p_1} = \frac{2\gamma M_1^2 - (\gamma - 1)}{\gamma + 1}$$

From the Rankine-Hugoniot relations (from mass, momentum, and energy conservation across a moving shock) for a CPG, we can also get the density and temperature ratios:

$$\frac{\rho_2}{\rho_1} = \frac{p_2}{p_1} \frac{T_1}{T_2} = \frac{1 + \frac{\gamma+1}{\gamma-1} \frac{p_2}{p_1}}{\frac{\gamma+1}{\gamma-1} + \frac{p_2}{p_1}}$$

$$\frac{T_2}{T_1} = \frac{p_2}{p_1} \left[\frac{\frac{\gamma+1}{\gamma-1} + \frac{p_2}{p_1}}{1 + \frac{\gamma+1}{\gamma-1} \frac{p_2}{p_1}} \right]$$

This allows us to then solve for the piston velocity u_p , i.e. the induced flow velocity in region 2 after the shock as passed:

$$u_p = u_2 = W \left(1 - \frac{\rho_1}{\rho_2} \right) = \frac{a_1}{\gamma} \left(\frac{p_2}{p_1} - 1 \right) \left[\frac{\frac{2\gamma}{\gamma+1}}{\frac{p_2}{p_1} + \frac{\gamma-1}{\gamma+1}} \right]^{1/2}$$

```
[2]: import numpy as np
from pint import UnitRegistry
ureg = UnitRegistry()
Q_ = ureg.Quantity

# Given/known constants as quantities
p1 = Q_(50.0, 'torr')           # Static pressure in region 1
T1 = Q_(300.0, 'kelvin')        # Temperature in region 1
W = Q_(1000.0, 'meter/second') # Primary shock speed
gamma = 1.4
R = Q_(287.0, 'J/kg/K')
a1 = np.sqrt(gamma * R * T1)
```

```

M1 = (W / a1).to_base_units().magnitude

# Pressure ratio across the moving shock
p2_over_p1 = (2 * gamma * M1**2 - (gamma - 1)) / (gamma + 1)
# Temperature ratio from Rankine-Hugoniot
ratio_term = (gamma + 1)/(gamma - 1)
T2_over_T1 = p2_over_p1 * ((ratio_term + p2_over_p1)/(1 + ratio_term *
    ↪p2_over_p1))
# Density ratio using p/(rho T)
rho2_over_rho1 = p2_over_p1/T2_over_T1

# Piston/induced flow velocity (region 2) from two equivalent forms
u_p = (W * (1 - 1/rho2_over_rho1))
u_p_closed = a1 * (2 / (gamma + 1)) * ((M1**2 - 1) / M1)

print(f"a1: {a1.m_as('m/s'):.3f} m/s")
print(f"M1: {M1:.3f}")
print(f"p2/p1: {p2_over_p1:.3f}")
print(f"T2/T1: {T2_over_T1:.3f}")
print(f"rho2/rho1: {rho2_over_rho1:.3f}")
print(f"u_p: {u_p.m_as('m/s'):.1f} m/s")
print(f"u_p (check): {u_p_closed.m_as('m/s'):.1f} m/s")

```

```

a1: 347.189 m/s
M1: 2.880
p2/p1: 9.512
T2/T1: 2.541
rho2/rho1: 3.744
u_p: 732.9 m/s
u_p (check): 732.9 m/s

```

Now, we want to solve for the post-EW state (region 5) using the properties we just found for region 2. We can analyze this problem by treating the EWs as a left-running expansion wave, and since the Riemann invariant J^+ is constant across such a wave, assuming a CPG we know:

$$u_2 + \frac{2a_2}{\gamma - 1} = u_5 + \frac{2a_5}{\gamma - 1}$$

We can then invoke the isentropic relations. We can do this because isentropic relations are a purely thermodynamic relation between two states, and additionally everything we've done so far is for an isentropic process (in this case, we're dealing with an expansion wave and derived the characteristics given $\frac{Ds}{Dt} = 0$). So:

$$\frac{p_5}{p_2} = \left(\frac{T_5}{T_2} \right)^{\gamma/(\gamma-1)}$$

And since $a^2 = \gamma RT$ for a CPG:

$$\left(\frac{T_5}{T_2} \right) = \left(\frac{a_5^2}{a_2^2} \right) = \left(\frac{a_5}{a_2} \right)^2$$

$$\frac{p_5}{p_2} = \left(\frac{a_5}{a_2} \right)^{2\gamma/(\gamma-1)}$$

Combining the Riemann equation and the above equation, we can parametrize u_5 and p_5 by the local speed of sound a_5 :

$$u_5 = u_2 + \frac{2}{\gamma - 1}(a_2 - a_5)$$

$$p_5 = p_2 \left(\frac{a_5}{a_2} \right)^{\frac{2\gamma}{\gamma-1}}$$

This means that given a_5 , we will be able to find all the other properties of state 5. But to do so, we need more information (more equations). This information will come from the $u_5 = u_{20}$, $p_5 = p_{20}$ boundary condition we found in Part (a). We simply need to repeat the process for the unknown shock W_T , but since we do not know its speed, we need to express its equations and solve simultaneously as a system of equations with the equations for region 2 and 5 above:

$$M_{10} = \frac{W_T}{\sqrt{\gamma_{10} R_{10} T_{10}}}$$

$$u_{20} = \frac{a_{10}}{\gamma_{10}} \left(\frac{p_{20}}{p_{10}} - 1 \right) \left[\frac{\frac{2\gamma_{10}}{\gamma_{10}+1}}{\frac{p_{20}}{p_{10}} + \frac{\gamma_{10}-1}{\gamma_{10}+1}} \right]^{1/2}$$

$$\frac{p_{20}}{p_{10}} = \frac{2\gamma_{10}M_{10}^2 - (\gamma_{10} - 1)}{\gamma_{10} + 1}$$

Invoking the boundary condition:

$$p_{20} = p_{10} \frac{2\gamma_{10}M_{10}^2 - (\gamma_{10} - 1)}{\gamma_{10} + 1} = p_5 = p_2 \left(\frac{a_5}{a_2} \right)^{\frac{2\gamma_1}{\gamma_1-1}}$$

$$u_{20} = \frac{a_{10}}{\gamma_{10}} \left(\frac{p_{20}}{p_{10}} - 1 \right) \left[\frac{\frac{2\gamma_{10}}{\gamma_{10}+1}}{\frac{p_{20}}{p_{10}} + \frac{\gamma_{10}-1}{\gamma_{10}+1}} \right]^{1/2} = u_5 = u_2 + \frac{2}{\gamma_1 - 1}(a_2 - a_5)$$

The unknowns in the equation above are W_T and a_5 , with everything else known through substitution. There are 2 equations and 2 unknowns, so solving numerically:

```
[3]: from scipy.optimize import fsolve

# Helium properties
gamma_10 = 5/3
R_10 = Q_(2077.0, 'J/kg/K')
p_10 = Q_(25.0, 'torr')
T_10 = Q_(300.0, 'kelvin')

p2 = p1 * p2_over_p1
T2 = T1 * T2_over_T1
a2 = np.sqrt(gamma * R * T2)
u2 = u_p
a_10 = np.sqrt(gamma_10 * R_10 * T_10)
```

```

gamma_1 = 1.4

# Extract magnitudes for computation (avoiding Pint complexity in solver)
a_10_mag = a_10.m_as('m/s')
a2_mag = a2.m_as('m/s')
u2_mag = u2.m_as('m/s')
p2_mag = p2.m_as('Pa')
p_10_mag = p_10.m_as('Pa')

# Define the system of equations, unknowns: [W_T, a_5] in SI base units (m/s)
def equations(vars):
    W_T_val, a5_val = vars # Both in m/s
    # Calculate M_10
    M_10 = W_T_val / a_10_mag
    # Calculate p_20/p_10
    p20_over_p10 = (2*gamma_10*M_10**2 - (gamma_10-1)) / (gamma_10 + 1)
    # Calculate p_20 from helium side (in Pa)
    p_20_helium = p_10_mag * p20_over_p10
    # Calculate p_5 from air side (in Pa)
    a5_over_a2 = a5_val / a2_mag
    p_5_air = p2_mag * (a5_over_a2)**(2*gamma_1/(gamma_1-1))
    # Calculate u_20 from helium side (in m/s)
    u_20_helium = (a_10_mag/gamma_10) * (p20_over_p10 - 1) * np.sqrt(
        (2*gamma_10/(gamma_10+1)) / (p20_over_p10 + (gamma_10-1)/(gamma_10+1))
    )
    # Calculate u_5 from air side (in m/s)
    u_5_air = u2_mag + (2/(gamma_1-1)) * (a2_mag - a5_val)
    # Equations (pressure and velocity continuity at contact surface)
    eq1 = p_20_helium - p_5_air # Pa
    eq2 = u_20_helium - u_5_air # m/s
    return [eq1, eq2]

# Solve the system
initial_guess = [500, 300]
solution = fsolve(equations, initial_guess)
W_T_solution = Q_(solution[0], 'm/s')
a5_solution = Q_(solution[1], 'm/s')

# Calculate final properties in region 20
M_10_final = (W_T_solution / a_10).to_base_units().magnitude
p20_over_p10_final = (2*gamma_10*M_10_final**2 - (gamma_10-1)) / (gamma_10 + 1)
p_20 = p_10 * p20_over_p10_final
u_20 = (a_10/gamma_10) * (p20_over_p10_final - 1) * np.sqrt(
    (2*gamma_10/(gamma_10+1)) / (p20_over_p10_final + (gamma_10-1)/(gamma_10+1))
)
ratio_term_10 = (gamma_10 + 1)/(gamma_10 - 1)

```

```

T20_over_T10 = p20_over_p10_final * ((ratio_term_10 + p20_over_p10_final)/(1 +
↳ratio_term_10 * p20_over_p10_final))
T_20 = T_10 * T20_over_T10

print("\nSolution:")
print(f"W_T (transmitted shock speed): {W_T_solution.m_as('m/s'):.2f} m/s")
print(f"a_5 (sound speed in region 5): {a5_solution.m_as('m/s'):.2f} m/s")
print(f"\nRegion 20 properties:")
print(f"p_20: {p_20.to('torr'):.2f}")
print(f"u_20: {u_20.m_as('m/s'):.2f} m/s")
print(f"T_20: {T_20.m_as('kelvin'):.2f} K")

```

Solution:

W_T (transmitted shock speed): 2104.68 m/s

a_5 (sound speed in region 5): 458.30 m/s

Region 20 properties:

p_20: 127.05 torr

u_20: 1208.44 m/s

T_20: 649.20 K

1.1.3 Part (c)

Prompt: *Find the pressure, temperature and speed in the expanded gas (region 5).*

We just need to find temperature T_5 from isentropic relations after the EW since $u_5 = u_{20}$, $p_5 = p_{20}$.

$$T_5 = T_2 \left(\frac{p_5}{p_2} \right)^{(\gamma_1 - 1/\gamma_1)}$$

```

[4]: T_5 = T2 * (p_20/p2)**((gamma_1 - 1)/gamma_1)
print(f"T_5: {T_5.m_as('kelvin'):.2f} K")

```

T_5: 522.76 K

$$u_5 = 1208.44 \text{ m/s}, p_5 = 127.05 \text{ Torr}, T_5 = 522.76 \text{ K}$$

1.1.4 Part (d)

Prompt: *Find the pressure, temperature and speed distributions along the tube at one instant in time (you choose the instant in time). In other words, you need to generate a plot of $p(x)$, $T(x)$, $u(x)$ along the tube across regions 2, 5, 20 and 10 at a given time t .*

We can solve this by dividing the problem into distinct regions, as mentioned in the prompt. Region 2 is uniform and fully defined, and is separated from region 5 through an expansion wave. The expansion wave can be described as a set of simple (centered) characteristics, emanating from the

origin. The leading characteristic (the “head”) will be at location $x_{\text{head}} = (u_2 - a_2)t$ since by definition, a simple left-running C^- characteristic has the slope:

$$\frac{dt}{dx} = \frac{1}{u - a}$$

on the x-t diagram. Thus the tail will have the location $x_{\text{tail}} = (u_5 - a_5)t$. The contact surface, due to the nonpenetration boundary condition, will have a location of $x_{CS} = u_5 t = u_{20} t$. The transmitted shock will have location $x_{\text{shock}} = W_T t$.

In Part (c) we derived the state variables at every region, so the only unsolved region is what happens between the head and tail of the expansion fans. For this, we once again invoke the J^+ Riemann invariant which is constant across these C^- characteristics. We use the similarity variable ξ , which is the basis for using characteristics (it allows us to simplify a system of PDEs into ODEs):

$$\xi = \frac{x}{t}$$

Once again looking at our problem as solving for a horizontal line on the x-t diagram, we can see that for a given t we can vary along x the variable ξ and solve directly for the state properties. So say we're on a C^- line with inverse slope ξ :

$$u - a = \xi$$

Substituting into the J^+ equation:

$$u + \frac{2a}{\gamma - 1} = u_2 + \frac{2a_2}{\gamma - 1} = (\xi + a) + \frac{2a}{\gamma - 1}$$

Simplifying we get:

$$a(\xi) = \frac{2}{\gamma_1 + 1} \left(a_2 + \frac{\gamma_1 - 1}{2} (u_2 - \xi) \right)$$

And the remaining state variables can be found through isentropic relations:

$$T(\xi) = T_2 \left(\frac{a(\xi)}{a_2} \right)^2$$

$$p(\xi) = p_2 \left(\frac{a(\xi)}{a_2} \right)^{\frac{2\gamma_1}{\gamma_1 - 1}}$$

```
[5]: import matplotlib.pyplot as plt

# Extract magnitudes for plotting (all in SI units)
p2_mag = p2.m_as('Pa')
T2_mag = T2.m_as('K')
u2_mag = u2.m_as('m/s')
a2_mag = a2.m_as('m/s')
p5_mag = p_20.m_as('Pa')
T5_mag = T_5.m_as('K')
a5_mag = a5_solution.m_as('m/s')
u5_mag = u_20.m_as('m/s')
```

```

p20_mag = p_20.m_as('Pa')
T20_mag = T_20.m_as('K')
u20_mag = u_20.m_as('m/s')
p10_mag = p_10.m_as('Pa')
T10_mag = T_10.m_as('K')
u10_mag = 0.0
W_T_mag = W_T_solution.m_as('m/s')

t = 50e-6 # 50 microseconds
# Calculate wave positions at time t
x_head = (u2_mag - a2_mag) * t # Expansion head (leftmost, negative since u2_
    ↪ a2)
x_tail = (u5_mag - a5_mag) * t # Expansion tail
x_CS = u5_mag * t # Contact surface
x_shock = W_T_mag * t # Transmitted shock front

print(f"Time: {t*1e6:.1f} s")
print(f"Expansion head: {x_head*1e3:.3f} mm")
print(f"Expansion tail: {x_tail*1e3:.3f} mm")
print(f"Contact surface: {x_CS*1e3:.3f} mm")
print(f"Transmitted shock: {x_shock*1e3:.3f} mm")

# Create x array with some margin on each side
margin = 0.002 # 2 mm margin
x = np.linspace(x_head - margin, x_shock + margin, 2000)
p = np.zeros_like(x)
T = np.zeros_like(x)
u = np.zeros_like(x)

# Fill in properties based on region
for i, xi in enumerate(x):
    if xi < x_head:
        # Region 2 (uniform, behind expansion)
        p[i] = p2_mag
        T[i] = T2_mag
        u[i] = u2_mag
    elif xi < x_tail:
        # Expansion fan - use similarity solution
        xi_t = xi / t # similarity variable
        a_local = (2 / (gamma + 1)) * (a2_mag + (gamma - 1) / 2 * (u2_mag -
    ↪ xi_t))
        u[i] = xi_t + a_local
        T[i] = T2_mag * (a_local / a2_mag)**2
        p[i] = p2_mag * (a_local / a2_mag)**(2 * gamma / (gamma - 1))
    elif xi < x_CS:
        # Region 5 (expanded air)

```

```

        p[i] = p5_mag
        T[i] = T5_mag
        u[i] = u5_mag
    elif xi < x_shock:
        # Region 20 (shocked helium)
        p[i] = p20_mag
        T[i] = T20_mag
        u[i] = u20_mag
    else:
        # Region 10 (undisturbed helium)
        p[i] = p10_mag
        T[i] = T10_mag
        u[i] = u10_mag

# Convert x to mm for plotting
x_mm = x * 1e3
fig, axes = plt.subplots(3, 1, figsize=(10, 8), sharex=True)

# Pressure plot
axes[0].plot(x_mm, p / 1e3, 'b-', linewidth=1.5)
axes[0].set_ylabel('Pressure (kPa)')
axes[0].grid(True, alpha=0.3)
axes[0].axvline(x_head*1e3, color='gray', linestyle='--', alpha=0.5,
    label='Wave boundaries')
axes[0].axvline(x_tail*1e3, color='gray', linestyle='--', alpha=0.5)
axes[0].axvline(x_CS*1e3, color='red', linestyle='--', alpha=0.5,
    label='Contact surface')
axes[0].axvline(x_shock*1e3, color='gray', linestyle='--', alpha=0.5)

# Temperature plot
axes[1].plot(x_mm, T, 'r-', linewidth=1.5)
axes[1].set_ylabel('Temperature (K)')
axes[1].grid(True, alpha=0.3)
axes[1].axvline(x_head*1e3, color='gray', linestyle='--', alpha=0.5)
axes[1].axvline(x_tail*1e3, color='gray', linestyle='--', alpha=0.5)
axes[1].axvline(x_CS*1e3, color='red', linestyle='--', alpha=0.5)
axes[1].axvline(x_shock*1e3, color='gray', linestyle='--', alpha=0.5)

# Velocity plot
axes[2].plot(x_mm, u, 'g-', linewidth=1.5)
axes[2].set_ylabel('Velocity (m/s)')
axes[2].set_xlabel('Position (mm)')
axes[2].grid(True, alpha=0.3)
axes[2].axvline(x_head*1e3, color='gray', linestyle='--', alpha=0.5)
axes[2].axvline(x_tail*1e3, color='gray', linestyle='--', alpha=0.5)
axes[2].axvline(x_CS*1e3, color='red', linestyle='--', alpha=0.5)
axes[2].axvline(x_shock*1e3, color='gray', linestyle='--', alpha=0.5)

```

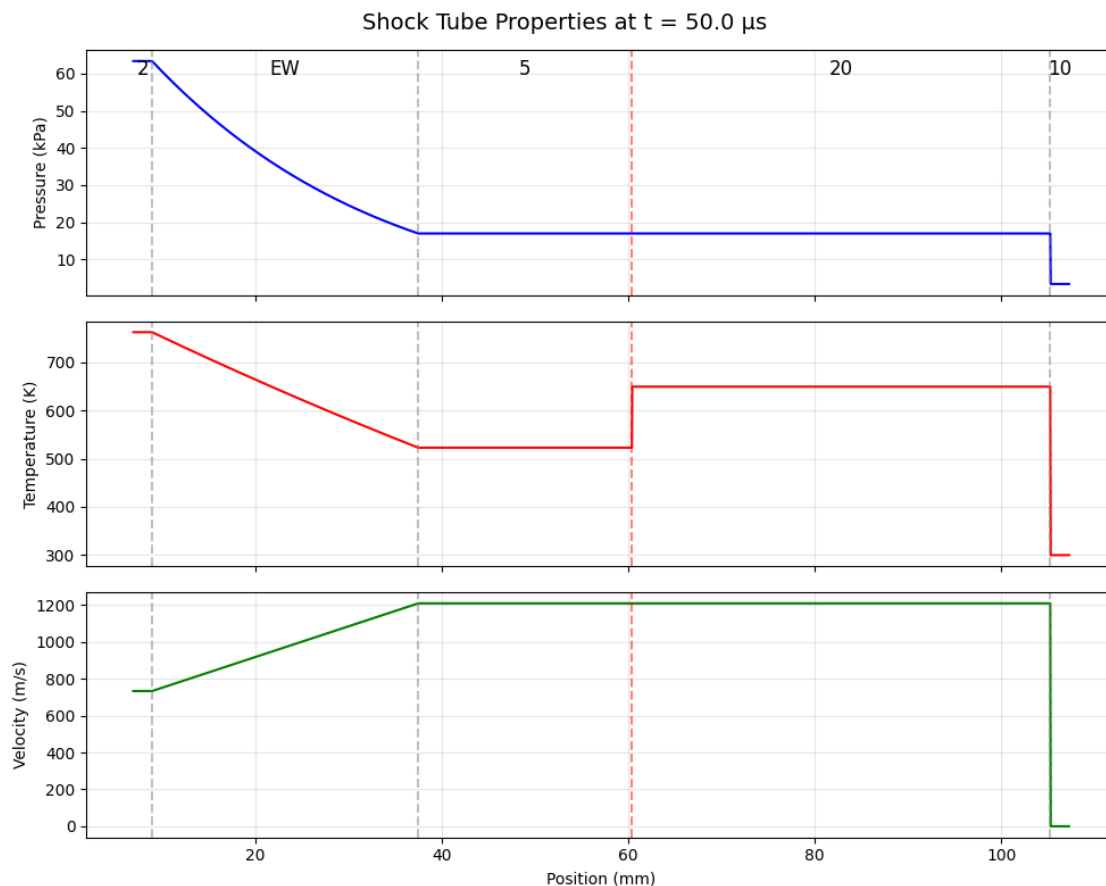
```

# Add region labels to top plot
y_label = axes[0].get_ylim()[1] * 0.9
axes[0].text((x_head*1e3 - margin*1e3/2), y_label, '2', ha='center',
    ↪fontsize=12)
axes[0].text((x_head*1e3 + x_tail*1e3)/2, y_label, 'EW', ha='center',
    ↪fontsize=12)
axes[0].text((x_tail*1e3 + x_CS*1e3)/2, y_label, '5', ha='center', fontsize=12)
axes[0].text((x_CS*1e3 + x_shock*1e3)/2, y_label, '20', ha='center',
    ↪fontsize=12)
axes[0].text((x_shock*1e3 + x_shock*1e3 + margin*1e3)/2, y_label, '10',
    ↪ha='center', fontsize=12)

fig.suptitle(f'Shock Tube Properties at t = {t*1e6:.1f} s', fontsize=14)
plt.tight_layout()
plt.show()

```

Time: 50.0 s
 Expansion head: 8.973 mm
 Expansion tail: 37.507 mm
 Contact surface: 60.422 mm
 Transmitted shock: 105.234 mm



1.1.5 Part (e)

Prompt: *Plot to scale the $x - t$ diagram of the wave system.*

Like before, we'll use the Riemann invariants to plot the characteristics, where the expansion head is:

$$\frac{dt}{dx} = \frac{1}{u_2 - a_2}$$

The tail:

$$\frac{dt}{dx} = \frac{1}{u_5 - a_5}$$

Any intermediate fan:

$$u = u_2 + \frac{2}{\gamma_1 - 1}(a_2 - a)$$

We can compute continuously every a between a_2 or a_5 , or parameterize by ξ as shown above.

```
[ ]: fig, ax = plt.subplots(figsize=(10, 8))
t_max = 100e-6 # 100 microseconds
t_plot = np.linspace(0, t_max, 100)

# Calculate slopes (dx/dt) for each feature
slope_head = u2_mag - a2_mag # Expansion head (negative, goes left)
slope_tail = u5_mag - a5_mag # Expansion tail
slope_CS = u5_mag # Contact surface
slope_shock = W_T_mag # Transmitted shock
n_characteristics = 15
a_values = np.linspace(a5_mag, a2_mag, n_characteristics)

for a_val in a_values:
    # Calculate u from Riemann invariant
    u_val = u2_mag + (2 / (gamma - 1)) * (a2_mag - a_val)
    slope = u_val - a_val
    x_char = slope * t_plot
    ax.plot(x_char * 1e3, t_plot * 1e6, 'b-', linewidth=0.8, alpha=0.6)

# Highlight head and tail of expansion
x_head = slope_head * t_plot
x_tail = slope_tail * t_plot
ax.plot(x_head * 1e3, t_plot * 1e6, 'b-', linewidth=2, label='Expansion fan')

# Plot contact surface
x_CS = slope_CS * t_plot
ax.plot(x_CS * 1e3, t_plot * 1e6, 'r--', linewidth=2, label='Contact surface')

# Plot transmitted shock
x_shock = slope_shock * t_plot
```

```

ax.plot(x_shock * 1e3, t_plot * 1e6, 'k-', linewidth=2, label='Transmitted_
↳shock')

# Add region labels
t_label = t_max * 0.5 * 1e6 # Vertical position for labels

# Region 2: left of expansion head
ax.text(slope_head * t_max * 0.5 * 1e3 - 5, t_label, '2', fontsize=14,
↳ha='center')

# Region 5: between expansion tail and contact surface
x_mid_5 = (slope_tail + slope_CS) / 2 * t_max * 0.5 * 1e3
ax.text(x_mid_5, t_label, '5', fontsize=14, ha='center')

# Region 20: between contact surface and shock
x_mid_20 = (slope_CS + slope_shock) / 2 * t_max * 0.5 * 1e3
ax.text(x_mid_20, t_label, '20', fontsize=14, ha='center')

# Region 10: right of shock
ax.text(slope_shock * t_max * 0.5 * 1e3 + 20, t_label, '10', fontsize=14,
↳ha='center')

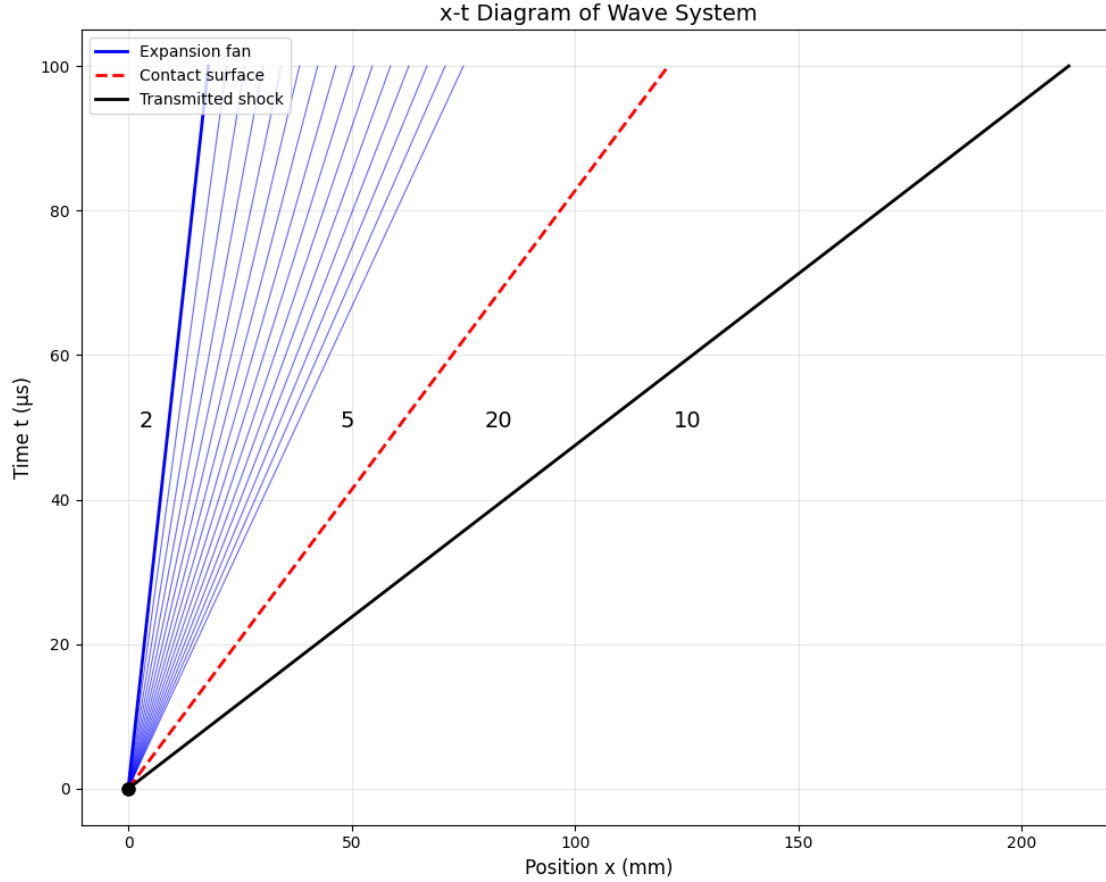
# Mark the origin
ax.plot(0, 0, 'ko', markersize=8)

# Labels and formatting
ax.set_xlabel('Position x (mm)', fontsize=12)
ax.set_ylabel('Time t (s)', fontsize=12)
ax.set_title('x-t Diagram of Wave System', fontsize=14)
ax.legend(loc='upper left')
ax.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

print(f"Wave speeds (dx/dt):")
print(f"Expansion head: {slope_head:.1f} m/s")
print(f"Expansion tail: {slope_tail:.1f} m/s")
print(f>Contact surface: {slope_CS:.1f} m/s")
print(f"Transmitted shock: {slope_shock:.1f} m/s")

```



Wave speeds (dx/dt):

Expansion head: 179.5 m/s

Expansion tail: 750.1 m/s

Contact surface: 1208.4 m/s

Transmitted shock: 2104.7 m/s

1.1.6 Part (f)

Prompt: *For the same initial conditions in region 1, identify under what conditions for region 20, if they exist, the reflected expansion waves is in reality a reflected shock wave.*

We can first answer this question intuitively. As an incident wave hits the contact surface, what happens in/as the reflected wave is purely a result of needing to match some boundary condition or invariant. In our case, we know that for a CS to continue existing, there is nonpenetration ($u_5 = u_{20}$) and pressure match ($p_5 = p_{20}$).

The key property is acoustic impedance $Z = \rho a$, which measures the “stiffness” of the gas. What we saw in the initial conditions given in Part (a) is that air is a lot stiffer (heavier molecule, higher pressure) than the helium at the other side of the interface, so when the incident wave hit the CS, helium was relatively easy to accelerate to satisfy $u_5 = u_{20}$. Thus, we found that the resultant pressure at the interface (matched pressure) is lower than p_2 , so for p_5 to be reached, the gas in

p_2 had to be expanded, and hence the EW. For us to see a reflected shock, the matched pressure p_5 must be higher than p_2 .

There is likely a smarter way to solve the problem “in the right direction,” but a brute force method will be to reuse our solution strategy from Part (a) and iterate on higher initial temperatures and pressures (since this would increase Z) to find when a shock will form. This will create a critical curve where on either side we will find a reflected shock, or a reflected expansion wave.

```
[16]: from scipy.optimize import fsolve, brentq

gamma_He = 5/3
R_He = 2077.0

def get_matched_pressure(p_10_Pa, T_10_K):
    """Solve for matched pressure given region 10 conditions."""
    a_10 = np.sqrt(gamma_He * R_He * T_10_K)

    def equations(vars):
        W_T_val, a5_val = vars
        M_10 = W_T_val / a_10
        if M_10 < 1:
            return [1e10, 1e10]
        p20_over_p10 = (2 * gamma_He * M_10**2 - (gamma_He - 1)) / (gamma_He + 1)
        p_20 = p_10_Pa * p20_over_p10
        u_20 = (a_10 / gamma_He) * (p20_over_p10 - 1) * np.sqrt(
            (2 * gamma_He / (gamma_He + 1)) / (p20_over_p10 + (gamma_He - 1) /
            (gamma_He + 1))
        )
        p_5 = p2_mag * (a5_val / a2_mag) ** (2 * gamma_1 / (gamma_1 - 1))
        u_5 = u2_mag + (2 / (gamma_1 - 1)) * (a2_mag - a5_val)
        return [p_20 - p_5, u_20 - u_5]
    sol = fsolve(equations, [1000, 400], full_output=True)
    if sol[2] != 1:
        return np.nan

    W_T_sol, _ = sol[0]
    M_10 = W_T_sol / a_10
    p20_over_p10 = (2 * gamma_He * M_10**2 - (gamma_He - 1)) / (gamma_He + 1)
    return p_10_Pa * p20_over_p10

# Find critical p_10 for various temperatures
T_10_range = np.linspace(50, 300, 50)
p_10_critical = []

for T_10 in T_10_range:
    try: # Find p_10 where p_matched = p2
        def residual(p_10_Pa):
```

```

        return get_matched_pressure(p_10_Pa, T_10) - p2_mag
    p_crit = brentq(residual, 100, 50000) # Search between ~1 and ~375 Torr
    p_10_critical.append(p_crit)
except:
    p_10_critical.append(np.nan)

p_10_critical = np.array(p_10_critical)
# Plot the critical boundary
fig, ax = plt.subplots(figsize=(8, 5))
ax.plot(T_10_range, p_10_critical / 133.322, 'b-', linewidth=2)
ax.fill_between(T_10_range, p_10_critical / 133.322, 400, alpha=0.3,
               color='red', label='Reflected shock')
ax.fill_between(T_10_range, 0, p_10_critical / 133.322, alpha=0.3,
               color='blue', label='Reflected expansion')
ax.set_xlabel('Region 10 Temperature (K)', fontsize=12)
ax.set_ylabel('Region 10 Pressure (Torr)', fontsize=12)
ax.set_title('Critical Conditions for Reflected Shock (Helium)', fontsize=12)
ax.legend()
ax.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

```

