AEE 553 — Compressible Flow

Department of Mechanical and Aerospace Engineering

Homework 5

Author:
Evan Burke

Instructor:
Dr. Carson Running

30 October 2022

Contents

Homework 5

Problem 1

Using $\frac{p_2}{p_2}$ as the metric for oblique-shock strength, come up with a way to graphically show the relationship between shock strength, β, and $M_{1,\infty}$.

Assumptions:

Assume a weak, attached oblique shock for a range of β and $M_{1,\infty}$ with $\gamma = 1.4$. For all Machs analyzed, a max wave angle of 60° is below the strong shock solution. Using $\beta_{min} = 1/\arcsin(M_{1,\infty})$ ensures all solutions are physically possible for an attached, left-running shock.

Solution:

Note: All calculations performed in Python, see appendix A.

We examine a range of Mach numbers from 2-10 with wave angle β ranging from the minimum value ($\beta_{min} = 1/\arcsin(M_{1,\infty})$) to 60°. Pressure ratios across the oblique shock are calculated using normal shock relations and the component of $M_{1,\infty}$ normal to the wave angle:

$$\frac{p_2}{p_1} = 1 + \frac{2}{\gamma+1}\left(M_{1,\infty}^2 \sin^2\beta - 1\right)$$

Figure 1 shows a 2D scatterplot of pressure ratio versus wave angle for a series of Machs from 2-10. Although shock strength does always increase with wave angle, the pressure ratio shows greater increases for an increase in Mach number. Lower Mach flows cannot experience wave angles as small as higher Mach flows, as shown by the difference between the β_{min} for Mach 2 flow and Mach 10 flow (30° vs. $< 1°$). The greater sensitivity to freestream Mach number indicates that there is no contradictory behavior between normal shocks and oblique shocks. Just like a normal shock, the strength of an oblique shock is dominated by the incoming Mach number.

Figure 2 shows a 3D scatterplot of the same data in figure 1. The 3D visualization hints at the shape of a response surface relationship between pressure ratio, β, and $M_{1,\infty}$. Calculation of analytical sensitivities for highly complex non-linear relationships such as these can be difficult, but developing response models can be useful for design of high-speed flow components such as inlets and nozzles. Despite the greatly increased pressure ratios for a single high-Mach oblique shock, the total pressure recovery associated with such strong single-shock systems is great and should be avoided by replacing the strong shock with a series of weaker shocks.

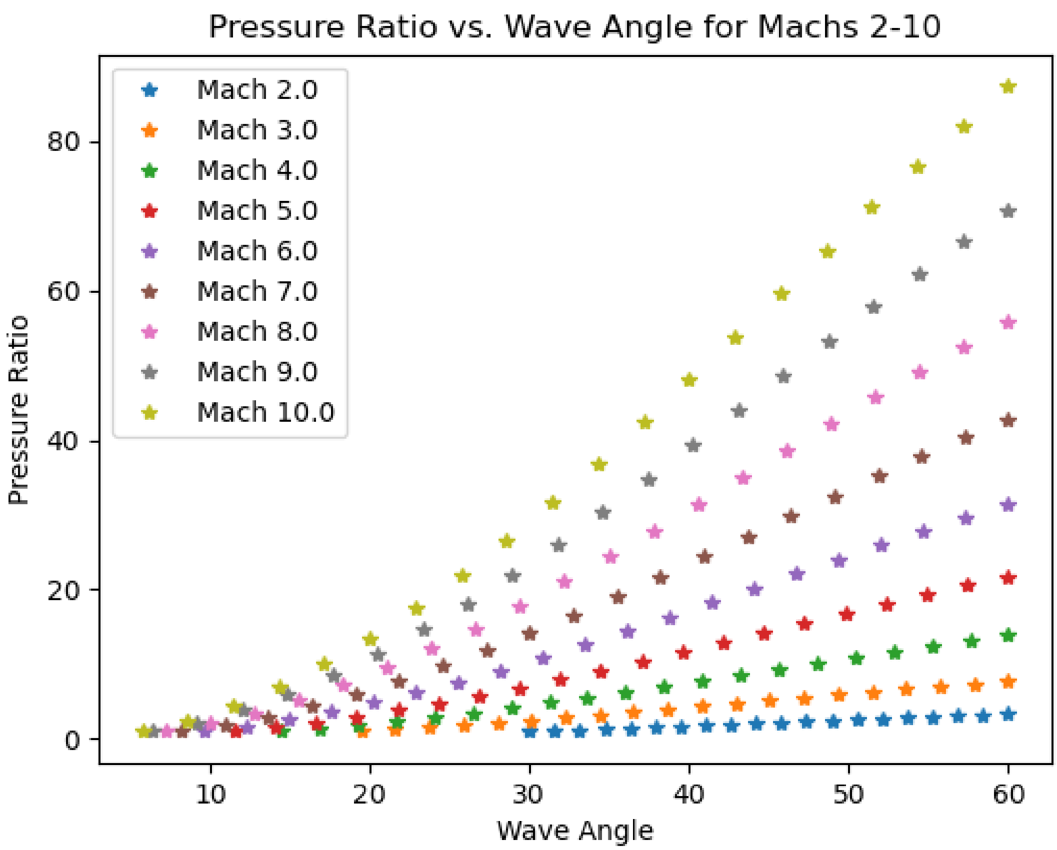

Figure 1: Pressure Ratio vs. Beta

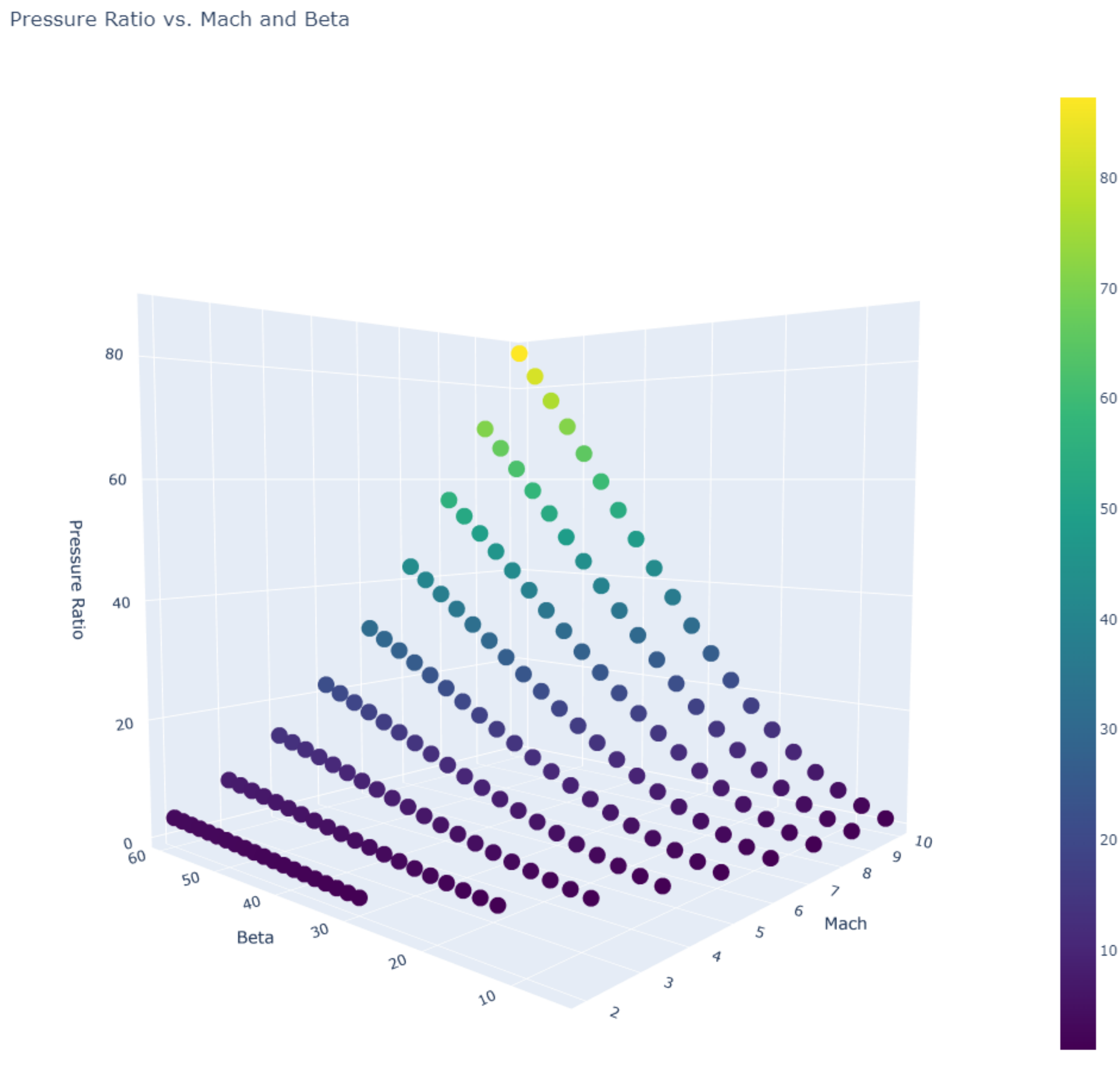

Figure 2: Pressure Ratio vs. Beta and Mach

Problem 3

Calculate the freestream pressure in regions 4 and 4' and the flow direction Φ behind the refracted shocks for $M_1 = 3$, $p_1 = 1\,\text{atm}$, $\theta_2 = 20°$, and $\theta_3 = 15°$.

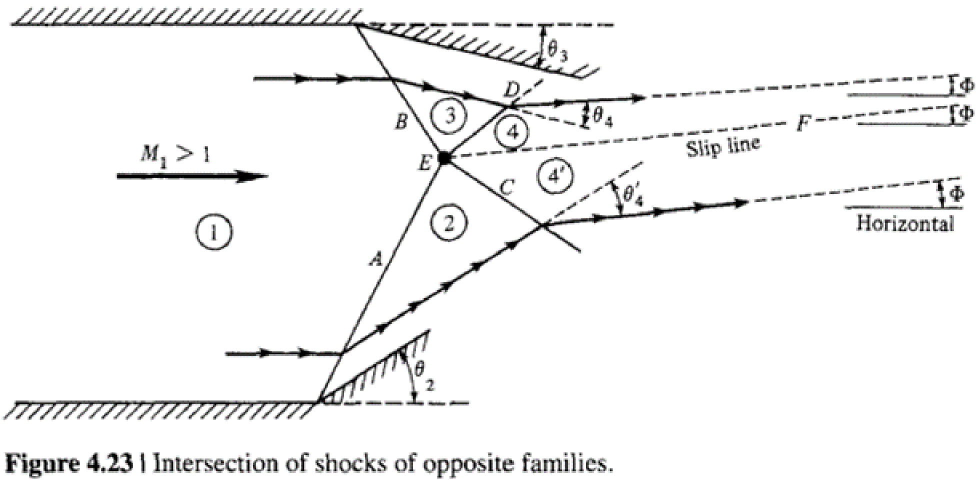

Figure 3: Shock interaction problem setup

Givens:

$M_1 = 3$
$p_1 = 1\,\text{atm}$
$\theta_2 = 20°$
$\theta_3 = 15°$

Assumptions:

Flow in the duct will be considered inviscid, steady, and isentropic outside of the shocks. In each region (1, 2, 3, 4, 4'), flow properties are constant and uniform, only changing across the shocks (A, B, C, D). Changes in area are neglected. There is no heat or work entering/exiting the system. Flow in regions 4 and 4' are oriented in the same direction at an angle Φ from the horizontal, with $p_4 = p_{4'}$. $\Phi = \theta_3 + \theta_4 = \theta_2 + \theta_{4'}$.

Solution:

Note: All calulations performed in MATLAB, see appendix B.

Given M_1 and the two ramp angles, θ_2 and θ_3, shock angles β_2 and β_3 are found via the following relation using a numerical solver:

$$\tan\theta_2 = 2\cot\beta_2\left[\frac{M_1^2\sin^2\beta_2 - 1}{M_1^2(\gamma + \cos 2\beta_2) + 2}\right]$$

$$\tan\theta_3 = 2\cot\beta_3 \left[\frac{M_1^2 \sin^2\beta_3 - 1}{M_1^2\left(\gamma + \cos 2\beta_3\right) + 2}\right]$$

$$\boxed{\beta_2 = 37.7636^\circ \qquad \beta_3 = 32.2404^\circ}$$

Post-oblique shock Mach numbers are calculated using the component of M_1 normal to shock A and shock B, denoted by $M_{1n,2}$ and $M_{1n,3}$, respectively:

$$M_{1n,2} = M_1 \sin\beta_2$$

$$M_{1n,3} = M_1 \sin\beta_3$$

$$M_{2n}^2 = \frac{M_{1n,2}^2 + \frac{2}{\gamma-1}}{\frac{2\gamma}{\gamma-1}M_{1n,2}^2 - 1}$$

$$M_{3n}^2 = \frac{M_{1n,3}^2 + \frac{2}{\gamma-1}}{\frac{2\gamma}{\gamma-1}M_{1n,3}^2 - 1}$$

$$M_2 = \frac{M_{2n}}{\beta_2 - \theta_2}$$

$$M_3 = \frac{M_{3n}}{\beta_3 - \theta_3}$$

$$\boxed{M_2 = 1.9941 \qquad M_3 = 2.2549}$$

Static pressure ratios across shocks A and B are found using oblique shock relations with M_1 and the shock angles:

$$\frac{p_2}{p_1} = 1 + \frac{2}{\gamma+1}\left(M_1^2 \sin^2\beta_2 - 1\right)$$

$$\frac{p_3}{p_1} = 1 + \frac{2}{\gamma+1}\left(M_1^2 \sin^2\beta_3 - 1\right)$$

$$\boxed{p_2 = 3.7713\,\mathrm{atm} \qquad p_3 = 2.8216\,\mathrm{atm}}$$

Solving for the conditions in regions 2 and 3 is a relatively trivial procedure. In order to solve for the conditions in regions 4 and 4', an iterative approach must be used. There are 5 unknowns needed to fully solve for the downstream conditions: θ_4, $\theta_{4'}$, β_4, $\beta_{4'}$ and p_4. The corresponding equations used to solve for state 4 and 4':

- Static pressure ratio across an oblique shock given that $p_4 = p_{4'}$:

$$\frac{p_4}{p_3} = 1 + \frac{2}{\gamma+1}\left(M_3^2 \sin^2\beta_4 - 1\right)$$

$$\frac{p_4}{p_2} = 1 + \frac{2}{\gamma+1}\left(M_2^2 \sin^2\beta_{4'} - 1\right)$$

- $\theta - \beta -$ Mach relations:

$$\tan\theta_4 = 2\cot\beta_4\left[\frac{M_3^2\sin^2\beta_4 - 1}{M_3^2\left(\gamma + \cos 2\beta_4\right) + 2}\right]$$

$$\tan\theta_{4'} = 2\cot\beta_{4'}\left[\frac{M_2^2\sin^2\beta_{4'} - 1}{M_2^2\left(\gamma + \cos 2\beta_{4'}\right) + 2}\right]$$

- The objective function that will be used as a constraint is the relation between turn angles and Φ:

$$\Phi_4 = \theta_3 + \theta_4$$

$$\Phi_{4'} = \theta_2 + \theta_{4'}$$

$$\boxed{\theta_4 - \theta_{4'} + \theta_3 - \theta_2 = 0}$$

The solution technique utilized to determine the correct downstream conditions is know as the secant method and is outlined below:

- Given two points, (x_a, y_a) and (x_b, y_b), the equation for a line connecting these points is given by point slope formula:

$$m = \frac{y_b - y_a}{x_b - x_a}$$

$$y - y_0 = m(x - x_0)$$

- Let $(x_0, y_0) = (x_b, y_b)$:

$$y - y_b = \frac{y_b - y_a}{x_b - x_a}(x - x_b)$$

- Plug in $y = 0$ to solve for the x-intercept of the line:

$$-y_b = \frac{y_b - y_a}{x_b - x_a}(x - x_b)$$

$$x = x_b - y_b \frac{y_b - y_a}{x_b - x_a}$$

- For an iterative solver this scheme becomes the following, known as secant method:

$$x_{i+1} = x_i - y_i \frac{y_i - y_{i-1}}{x_i - x_{i-1}}$$

This method has the advantage of not needing to bound the true value of zero or determine if the sign of y_i changes relative to y_{i-1}. The solution method will involve iterating across values of downstream pressure, p_4, calculating the value of the objective function, and iterating until the value converges to 0 within a chosen tolerance. To better set up the initial conditions, values of the objective function are calculated until a sign change is observed, indicating that the zero lies between the previous two calculated points. Figure 4 shows the objective function versus p_4 plotted to the point of the sign change. The final two values of p_4 will be used as the points x_1 and x_2 to initialize the secant method algorithm.

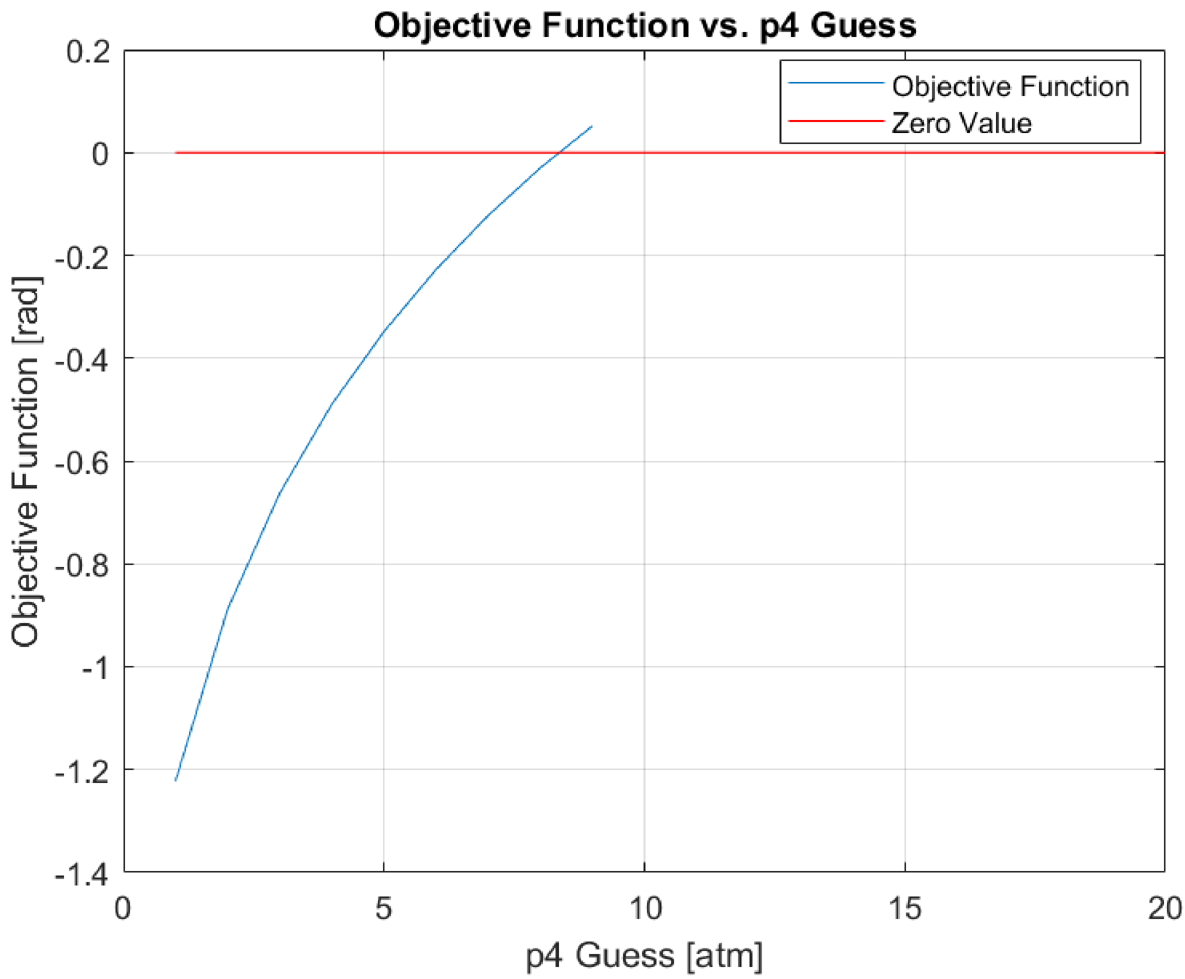

Figure 4: Objective function value plotted against p_4 guess to find sign change

The secant method converges to an objective function value of 0 (tolerance $= 1 \times 10^{-10}$) in 6 iterations, yielding the following values for the unkown variables:

$$\boxed{\theta_4 - \theta_{4'} + \theta_3 - \theta_2 = 0}$$

$$\boxed{p_4 = 8.3526\,\text{atm}}$$

$$\boxed{\theta_4 = 19.80°}$$

$$\boxed{\theta_{4'} = -15.20°}$$

$$\boxed{\Phi = 4.80^\circ}$$

$$\boxed{\beta_4 = 46.55^\circ}$$

$$\boxed{\beta_{4'} = -45.76^\circ}$$

Appendix A Problem 1 Python Code

```
# Compressible Flow
# AEE 553
# Homework 5 - Problem 1
# Evan Burke

from cProfile import label
from cmath import pi
import numpy as np
from matplotlib import pyplot as plt
import shocks as ns
import oblique as os


machs = np.linspace(2,10,num=9,endpoint=True)
print(machs)

gamma = 1.4

def find_theta(M=None,beta=None,gamma=1.4):
    beta = np.deg2rad(beta)
    tanth = 2 / np.tan(beta) * (M**2 * np.sin(beta)**2 - 1) / (M**2 * (
   gamma + np.cos(2*beta)) + 2)
    theta = np.arctan(tanth)
    theta = np.rad2deg(theta)
    #print(theta)
    return theta

data_dict = {}
data = []

for M in machs:
    prs = []
    beta_min = np.arcsin(1/M)*180/pi
    betas = np.linspace(beta_min,60,num=20,endpoint=True)

    for beta in betas:
        M1n = os.get_m1_normal(M1=M,beta=beta)
        M2n = os.get_m2_normal(M1n=M1n)
        pr = ns.get_static_pressure_ratio_normal_shock(M1=M1n)
        prs.append(pr)

    data_dict[M] = ((betas,prs))


fig,ax = plt.subplots()

for M in machs:
```

```python

    data = data_dict[M]
    plt.plot(data[0],data[1],'*',label=f'Mach {M}')

ax.legend()
ax.set_xlabel('Wave Angle')
ax.set_ylabel('Pressure Ratio')
ax.set_title('Pressure Ratio vs. Wave Angle for Machs 2-10')
plt.savefig('../images/problem_1/pr_vs_beta_2D.png')

scatter_data = []

for m in machs:
    foo = data_dict[m]
    bs = foo[0]
    ps = foo[1]
    print(bs,ps)

    for b,p in zip(bs,ps):
        scatter_data.append((m,b,p))

print(scatter_data)

Xs = [point[0] for point in scatter_data]
Ys = [point[1] for point in scatter_data]
Zs = [point[2] for point in scatter_data]

import plotly.graph_objects as go

data=[go.Scatter3d(x=Xs, y=Ys, z=Zs, mode='markers', marker=go.scatter3d.
    Marker(showscale=True), marker_color=Zs, marker_colorscale='Viridis')]

fig = go.Figure(data)

fig.update_layout(
    title='Pressure Ratio vs. Mach and Beta',
    autosize=False,
    width=1000,
    height=1000,
        scene=dict(
        xaxis_title='Mach',
        yaxis_title='Beta',
        zaxis_title='Pressure Ratio',
    ),
)

fig.show()
```

Appendix B Problem 3 MATLAB Code

```
%% Compressible Flow - AEE 553
% Homework 5 - Problem 3
% Evan Burke
% 28 October 2022

clear; close; clc;

% Givens
th2 = 20; th3 = -15; % deg
M1 = 3; p1 = 1;
gamma = 1.4;

% Region 2 and 3 oblique shock solution
b2 = beta_solver(gamma,M1,th2);
b3 = beta_solver(gamma,M1,th3);

M1n2 = M1 * sind(b2);
M1n3 = M1 *sind(b3);

M2n = ((M1n2^2+2/(gamma-1)) / (2*gamma/(gamma-1)*M1n2^2-1))^0.5;
M3n = ((M1n3^2+2/(gamma-1)) / (2*gamma/(gamma-1)*M1n3^2-1))^0.5;

M2 = M2n/sind(b2-th2);
M3 = M3n/sind(b3-th3);

p2 = p1*(1 + 2*gamma/(gamma+1)*(M1n2^2-1));
p3 = p1*(1 + 2*gamma/(gamma+1)*(M1n3^2-1));

% Check for sign change in function
for i=1:20 % know that p4 is greater than 1, 20 seems high enough to find
    one sign change
    [b4,b4p,th4,th4p,diff] = shock_interaction(i,gamma,M2,M3,p2,p3,th2*pi
    /180,th3*pi/180);
    x0(i) = i;
    b4s(i) = b4;
    b4ps(i) = b4p;
    th4s(i) = th4;
    th4ps(i) = th4p;
    diffs(i) = diff;

    if diffs(i) < 0
        polarity(i) = -1;
    else
        polarity(i) = 1;
    end

    if i>1
```

```
        if polarity(i-1) ~= polarity(i)
            break
        end
    end
end

figure
plot(x0,diffs,[1,20],[0,0],'r')
grid
xlabel('P4 guess')
ylabel('Objective Function Value')
legend('Objective Function','Zero Value')

% Solver Initial Conditions
i = 2;
x(1) = x0(end-1); % last value before sign change
x(2) = x0(end); % final value of polarity check, opposite sign as x(end-1)
y(1) = diffs(end-1); % value associated with x(end-1)
y(2) = diffs(end); % value associated with x(2)
m(1) = 1; % initial slope needed for solution, arbitrary

% Bisection Method
while abs(diff) > 1e-5
    [b4,b4p,th4,th4p,diff] = shock_interaction(x(i),gamma,M2,M3,p2,p3,th2*
   pi/180,th3*pi/180);
    y(i) = diff;
    m = (y(i)-y(i-1)) / (x(i)-x(i-1));
    x(i+1) = -y(i)/m + x(i);
    i = i + 1;
end

fprintf('p4 = %f\n',x(i))
fprintf('b4 = %f\n',b4*180/pi)
fprintf('b4p = %f\n',b4p*180/pi)
fprintf('th4 = %f\n',th4*180/pi)
fprintf('th4p = %f\n',th4p*180/pi)

function [b4,b4p,th4,th4p,diff] = shock_interaction(p0,gamma,M2,M3,p2,p3,
   th2,th3)
    syms b4 b4p th4 th4p p4 % declare symbolic vars
    % currently accepts and outputs radians instead of degrees
    eq_1 = p4/p3 == 1 + 2*gamma/(gamma+1) * ((M3*sin(b4))^2-1); % Oblique
   shock eqn  p2/p1 across OS
    eq_2 = p4/p2 == 1 + 2*gamma/(gamma+1) * ((M2*sin(b4p))^2-1); % Oblique
    shock eqn, p2/p1 across OS
    eq_3 = tan(th4) == 2*cot(b4) * (M3^2*sin(b4)^2-1) / (M3^2 *(gamma +
   cos(2*b4)) + 2); % theta-beta-Mach relation
    eq_4 = tan(th4p) == 2*cot(b4p) * (M2^2*sin(b4p)^2-1) / (M2^2 *(gamma +
    cos(2*b4p)) + 2); % theta-beta-Mach relation
    eq_5 = solve(eq_1,b4); % solve for b4
```

```
    eq_6 = solve(eq_2,b4p); % solve for b4'
    eq_7 = solve(eq_3,th4); % solve for th4
    eq_8 = solve(eq_4,th4p); % solve for th4'

    b4i = subs(eq_5,p4,p0); % Placeholder value for beta4, extracting from syms
    b4 = double(abs(b4i(1))); % Converting beta4 val to double, taking positive, forming array
    b4pi = subs(eq_6,p4,p0); % Placeholder value for beta4'
    b4p = double(-abs(b4pi(1))); % Converting beta4' val to double, taking negative, forming array
    th4i = subs(eq_7,b4); % Placeholder value, theta4
    th4 = double(th4i); % Val to double, into array
    th4pi = subs(eq_8,b4p); % Placeholder value, theta4'
    th4p = double(th4pi); % Val to double, into array
    diff = th4 - th4p + th3 - th2; % 'Objective function', want 0 per constraints
end

function [beta] = beta_solver(gamma,M,theta)
    % accepts and returns degrees
    delta=1;
    theta=theta*pi/180;
    lamb = ((M^2-1).^2 - 3*(1 + (gamma-1)/2*M^2) * (1 + (gamma+1)/2*M.^2) * tan(theta)^2)^0.5;
    chi = ((M^2-1)^3 - 9 * (1 + (gamma-1)/2 * M^2) * (1 + (gamma-1)/2 * M^2 + (gamma+1)/4*M^4).*tan(theta)^2)/lamb^3;
    tan_beta = (M^2 - 1 + 2*lamb*cos((4*pi*delta+acos(chi))/3)) / (3 * (1 + (gamma-1)/2*M^2).*tan(theta));
    beta = atan(tan_beta)*180/pi;
end
```