# AE 5385: High Temperature Gasdynamics
# Homework 1

James Grisham

September 22, 2015

## Problem 1

### Problem Statement

(Modified from Anderson, question 1.1) Consider the supersonic and hypersonic flow of air (with constant ratio of specific heats, $\gamma = 1.4$) over a 2D wedge with a half-wedge angle of 20 degrees. Let $\theta$ denote the half-wedge angle and $\beta$ the shock wave angle. Then, $\beta - \theta$ is a measure of the shock layer thickness.

a. Make a plot of $\beta - \theta$ vs the freestream Mach number for $2 \leqslant M_\infty \leqslant 20$. Make some comments as to what Mach number range results in a thin shock layer.

b. Plot the static pressure ratio, static temperature ratio, and the static density ratio across the shock for the same Mach number interval considered in part a.

c. Would you expect to get a smaller or larger shock layer thickness if you have used a cone with a half-angle of 20 degrees instead of the 2D wedge for the same Mach number range? Explain your reasoning.

### Solution

A Python module was written which contains all the oblique shock relations. The code can be seen in the appendix.

#### Part A

Figure 1 shows the $\theta$-$\beta$-$M$ diagram for several Mach numbers from 2 to 20. Holding the deflection angle constant, as the Mach number decreases, the shock-wave angle decreases.

Figure 2 shows the difference between the shock-wave angle and the flow deflection angle for several Mach numbers. In this case, the flow deflection angle was set to $20^\circ$.

The problem statement asks for some comments on which Mach number range produces a "thin" shock layer. The meaning of the word "thin", in this context, seems somewhat subjective. I couldn't find a clear definition of what a "thin" shock layer is so I will just guess based on the relative values of $\beta - \theta$.

Examining Figure 2 reveals that the difference between the shock-wave angle and flow deflection angles are relatively small for freestream Mach numbers ranging from 10 to 20. That is, the shock layers are relatively thin for $10 \leqslant M_\infty \leqslant 20$.

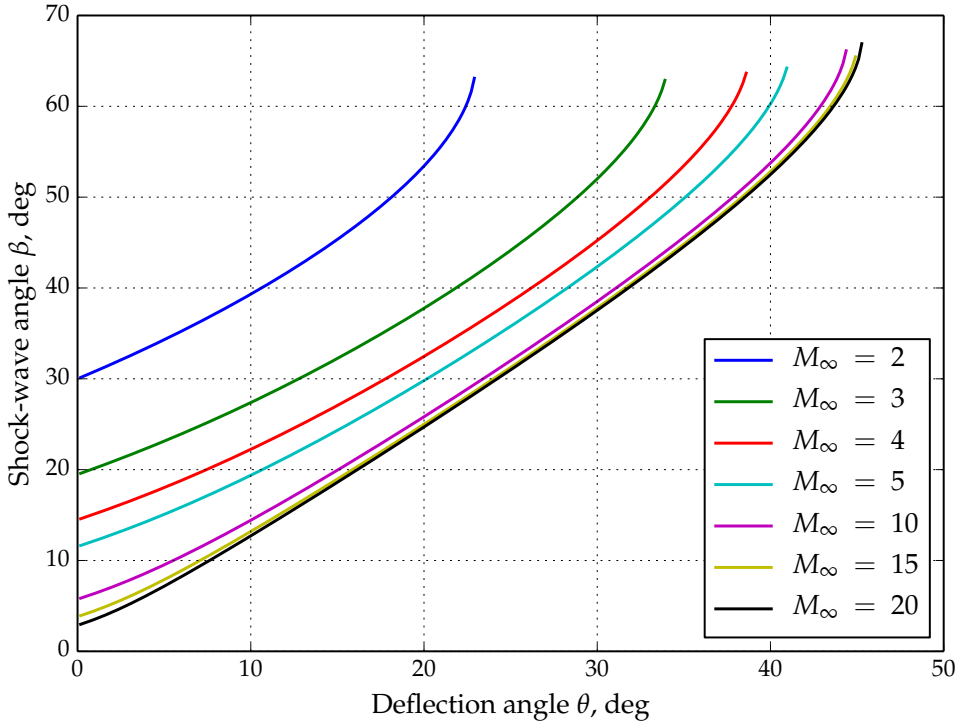



Figure 1: $\theta$-$\beta$-$M$ diagram.

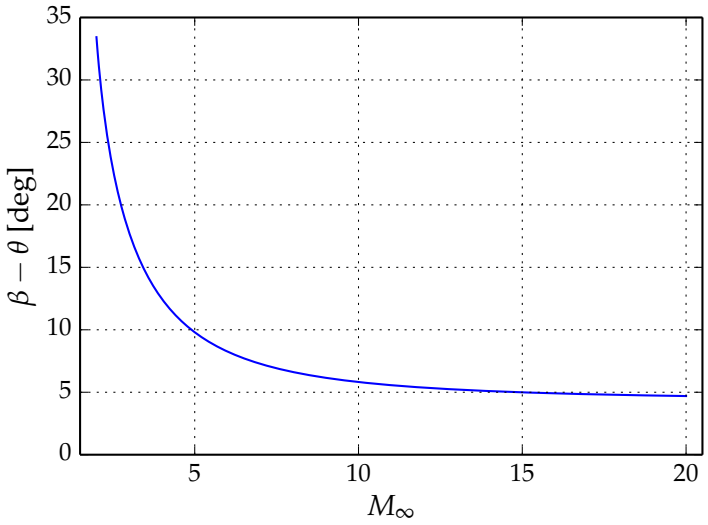



Figure 2: Difference between the shock-wave angle and flow deflection angle for several Mach numbers.

## Part B

Figure 3 shows the pressure, density and temperature ratios across a shock-wave for a flow deflection angle of 20°.

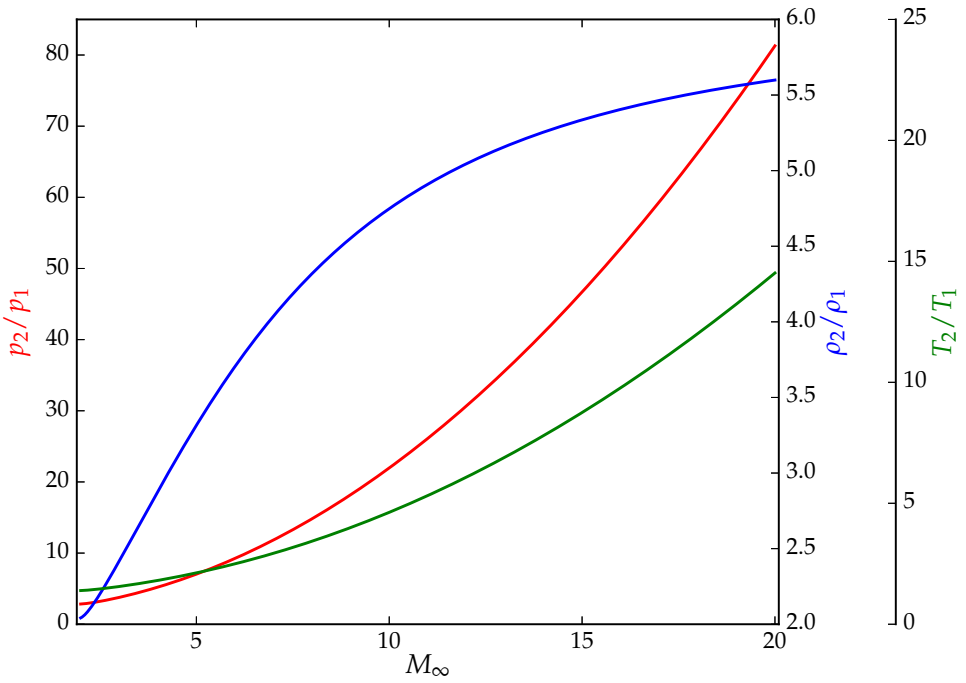



Figure 3: Plot of pressure, density and temperature ratios across shock.

## Part C

In the case of the three dimensional cone, the flow field between the shock and the surface is not uniform as it is in the wedge case. A MATLAB code from a previous gasdynamics course was used to determine the difference between the shock angle and flow deflection angle. This was accomplished by solving a nonlinear ODE. The result can be seen in Figure 4. The plot clearly shows that the cone has a smaller shock layer than the wedge.

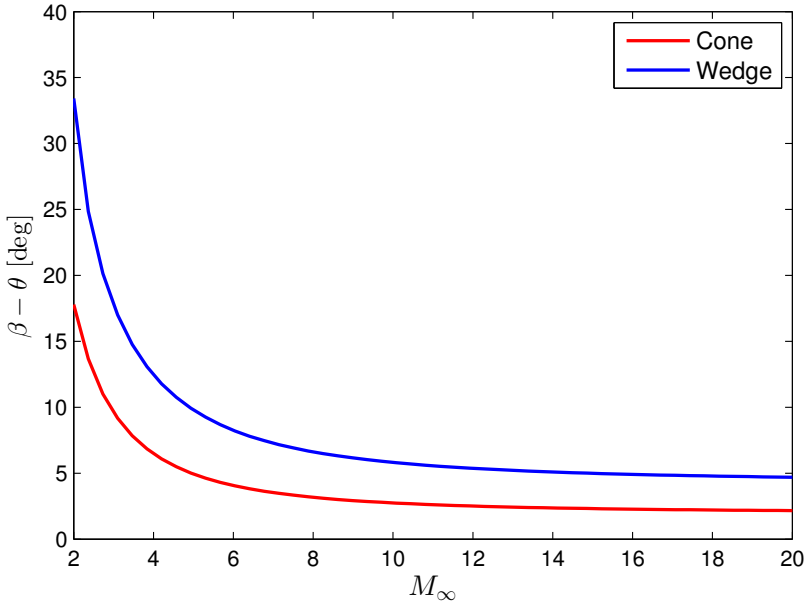



Figure 4: Comparison between shock layers for cone and wedge geometries.

## Problem 2

### Problem Statement

The following points have been selected from the Space Shuttle re-entry trajectory:

| | $V_\infty$ [m/s] | Altitude [m] |
|---|---|---|
| I | 8,047 | 74,980 |
| II | 5,133 | 60,655 |
| III | 3,130 | 49,378 |
| IV | 1,208 | 30,480 |

For each point, use the free-stream properties for air at the given altitude to calculate

(a) Free-stream Mach number.

(b) Free-stream Reynolds number.

(c) The dynamic pressure, kinetic energy per unit mass, and the stagnation enthalpy per unit mass of the free-stream.

(d) Maximum pressure and the corresponding pressure coefficient on the vehicle surface (assume calorically perfect gas for air in your calculations).

(e) State which high temperature effects (vibrational excitation, dissociation, and ionization) will be observed at these trajectory points (hint: use Figure 9.13 from Anderson).

## Solution

The international standard atmosphere will be used to estimate all the above values. The equations used in calculations are listed below:

$$M_\infty = \frac{V_\infty}{\sqrt{\gamma R T}} \tag{1}$$

$$\mathrm{Re} = \frac{\rho V_\infty L}{\mu} \tag{2}$$

$$q = \frac{1}{2}\rho V_\infty^2 \tag{3}$$

$$\mathrm{KE}_{\mathrm{avg}} = \frac{1}{2}V_\infty^2 \tag{4}$$

$$h_0 = c_p T + \frac{1}{2}V_\infty^2 \tag{5}$$

$$p_0 = p_\infty + q \tag{6}$$

$$C_p = \frac{p - p_\infty}{q} \tag{7}$$

The viscosity was calculated using Sutherland's law. The max pressure on the vehicle should be the stagnation pressure which is just the sum of the static and dynamic pressures. The calculations were accomplished using a Python script. The results can be seen below in tabular form.

Table 1: Results.

| $V_\infty$ [m/s] | Altitude [m] | $\rho$ [kg/m$^3$] | $M_\infty$ | Re | $q_\infty$ [Pa] | KE [MJ/kg] | $h_0$ [J/kg] | $p_0$ [Pa] | $C_p$ |
|---|---|---|---|---|---|---|---|---|---|
| 8,047 | 74,980 | 0.000035 | 27.9 | 675,917 | 1,132 | 32.3 | 32,584,652 | 1,134 | 1 |
| 5,133 | 60,655 | 0.000265 | 16.4 | 2,851,633 | 3,494 | 13.2 | 13,418,472 | 3,512 | 1 |
| 3,130 | 49,378 | 0.001058 | 9.49 | 6,375,400 | 5,182 | 4.90 | 5,170,223 | 5,264 | 1 |
| 1,208 | 30,480 | 0.016730 | 3.99 | 44,832,105 | 12,200 | 0.73 | 957,705 | 13,296 | 1 |

Figure 9.13 from Anderson's book shows the temperature ranges of different high temperature effects. Using this information, the below table was created.

Table 2: High temperature effects at different trajectory points.

| Trajectory point | vibrational excitation | dissociation | ionization |
|---|---|---|---|
| I | ✗ | ✓ | ✗ |
| II | ✗ | ✓ | ✗ |
| III | ✗ | ✓ | ✗ |
| IV | ✓ | ✗ | ✗ |

# Problem 3

## Problem Statement

Consider that a 70-deg sphere cone plantary entry capsule with 2.65 m maximum diameter is used for Mars entry. Assume that this system enters the Mars atmosphere with a mass of 830 kg at an entry velocity of 5.45 km/s at an altitude of 125 km and with an entry flight path angle of 5 degrees. Also assume a ballistic entry with constant flight path angle and constant drag coefficient of 1.65. Use a density vs altitude model given by $\rho(h) = \rho_0 e^{-ah}$ where $\rho_0 = 0.057$ kg/m$^3$ and $a = 1.275 \times 10^{-4}$ m$^{-1}$ in your calculations. Use a value of 3.75 m/s$^2$ for the gravitational acceleration on Mars.

a. For the given nominal conditions:

   (i) Determine the ballistic coefficient for the probe.

   (ii) Determine the peak acceleration of the probe in terms of earth g's, the peak deceleration altitude, and the velocity at the peak deceleration conditions.

   (iii) Plot altitude vs velocity and altitude vs deceleration (in terms of earth g's) between the entry altitude and an altitude of 10 km.

b. Now repeat part a by increasing the entry velocity, the flight path angle, the drag coefficient and the $\rho_0$ by +5% from their nominal values. By analyzing the results, discuss the sensitivity of the output quantities asked in part a to the change in each variable (make sure to obtain the results by changing one variable at a time).

## Solution

Summing the forces in the $x$-direction,

$$\sum F_x = -D + W \sin\theta = m\frac{dV}{dt} \tag{8}$$

If $\theta$ is small, $\sin\theta \approx 0$, making the equation

$$m\frac{dV}{dt} = -D = -qSC_D \tag{9}$$

We need a relation between $V$ and $h$.

$$\frac{dV}{dh} = \frac{dV}{dt}\frac{dt}{dx}\frac{dx}{dh} \tag{10}$$

Because $\theta$ is constant,

$$\frac{dx}{dh} = \frac{1}{\sin\theta} \tag{11}$$

Now,

$$\frac{dV}{dh} = \frac{\rho_0 e^{-ah} V S C_D}{2m\sin\theta} \tag{12}$$

Rearranging,

$$\frac{1}{V}\,dV = \frac{\rho_0 S C_D}{2m\sin\theta} e^{-ah}\,dh \tag{13}$$

Integrating yields

$$\int_{V_0}^{V} \frac{1}{V'}\, dV' = \frac{\rho_0 S C_D}{2m \sin\theta} \int_{h_0}^{h} e^{-ah'}\, dh' \tag{14}$$

The result is

$$V(h) = V_0 \exp\left[\frac{-S\, C_D\, \rho_0}{2\, a\, m \sin\theta}\left(e^{-ah} - e^{-ah_0}\right)\right] \tag{15}$$

The deceleration is

$$a(h) = -\frac{C_D S V_0^2 \rho_0}{2m} e^{-ah} \exp\left[-\frac{C_D S \rho_0\left(-e^{-ah_0} + e^{-ah}\right)}{am \sin(\theta)}\right] \tag{16}$$

The maximum deceleration can be found by differentiating (16), setting it equal to zero and solving for $h$. Doing so yields

$$h = \frac{1}{a} \ln\left[\frac{S\, C_D\, \rho_0}{am \sin\theta}\right] \tag{17}$$

Inserting this value into (16),

$$a_{\max} = -\frac{1}{2} a V_0^2 \sin\theta \exp\left[\frac{C_D \rho_0 S e^{-ah_0}}{am \sin\theta} - 1\right] \tag{18}$$

The velocity at the point of max deceleration is

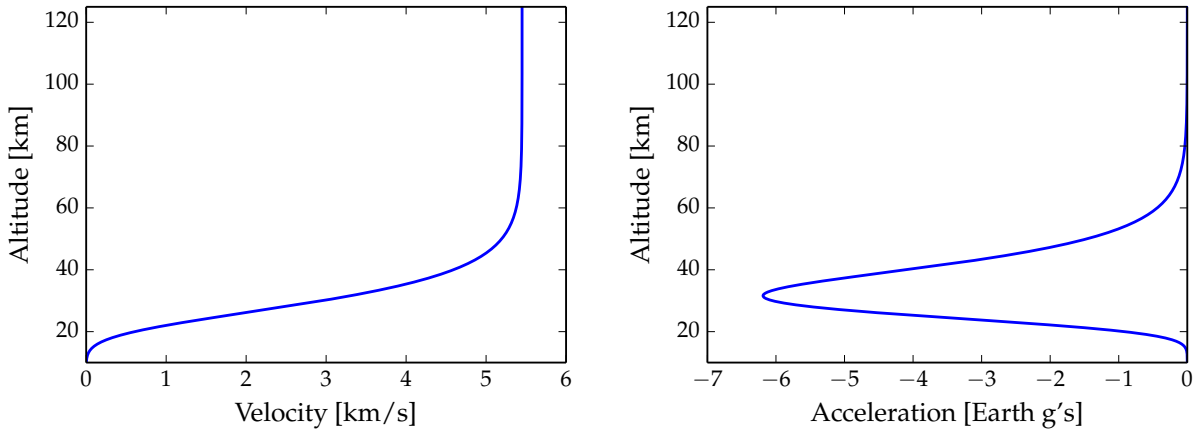
$$V = V_0 \exp\left(\frac{C_D \rho_0 S e^{-ah_0}}{2am \sin\theta} - \frac{1}{2}\right) \tag{19}$$

The ballistic coefficient is given by

$$\beta \equiv \frac{W}{C_d S} \tag{20}$$

The results for the present case are:

```
Altitude at max deceleration: 31605 m
Velocity at max deceleration:  3306 m/s
Max deceleration: -6.188831 earth gs
Ballistic coefficient: 342.01 N/mˆ2
```




(a) Velocity-altitude map.

(b) Deceleration-altitude map.

Figure 5: Plots of Velocity and deceleration vs altitude.

For part b, the sensitivities are reported in terms of percent difference from the baseline trajectory, which is calculated as follows:

$$\% \text{ difference} = \frac{\sigma' - \sigma}{\sigma} \tag{21}$$

where $\sigma$ is the value from the baseline trajectory and $\sigma'$ is the value from the perturbed trajectory. This was implemented using a class in the Python scripting language. A function was defined which takes two trajectory objects and computes the percent difference of certain paramters. The code can be seen in the appendix.

Table 3: Sensitivity of various parameters to a 5% increase in inputs.

| | Output parameters | | | |
|---|---|---|---|---|
| perturbation type | ballistic coefficient | max deceleration | $V$ at max deceleration | $h$ at max deceleration |
| initial altitude | 0.00e+00 | -3.70e-06 | -1.85e-06 | 0.00e+00 |
| initial velocity | 0.00e+00 | 1.03e-01 | 5.00e-02 | 0.00e+00 |
| diameter | 9.30e-02 | 6.91e-07 | 3.45e-07 | 2.42e-02 |
| mass | 5.00e-02 | -3.21e-07 | -1.60e-07 | -1.21e-02 |
| gravitational acceleration | 5.00e-02 | 0.00e+00 | 0.00e+00 | 0.00e+00 |
| drag | -4.76e-02 | 3.37e-07 | 1.68e-07 | 1.21e-02 |
| $a$ | 0.00e+00 | 5.00e-02 | -1.92e-06 | -5.92e-02 |
| $\rho_0$ | 0.00e+00 | 3.37e-07 | 1.68e-07 | 1.21e-02 |
| $\theta$ | 0.00e+00 | 4.99e-02 | -1.60e-07 | -1.21e-02 |

It seems that all the numbers in the table are relatively small meaning that the outputs are insensitive to changes in the inputs. The most sensitive relationship appears to be the max deceleration when subjected to a perturbation in the initial velocity (i.e., the percentage difference is approximately 10%.).

## Code listing

```
import numpy as np

def p_ratio(M, theta, gamma=1.4):
    b1 = beta(M, theta, gamma)
    return 1.0 + 2.0*gamma/(gamma+1.0)*(M**2*np.sin(b1)**2 - 1.0)

def rho_ratio(M, theta, gamma=1.4):
    b1 = beta(M, theta, gamma)
    return (gamma+1.0)*M**2*np.sin(b1)**2/(
        (gamma-1.0)*M**2*np.sin(b1)**2 + 2.0)

def T_ratio(M, theta, gamma=1.4):
    b1 = beta(M, theta, gamma)
    return 1.0 + 2.0*(gamma-1.0)/(gamma+1.0)**2*(M**2*np.sin(b1)**2 - 1.0)/(
        M**2*np.sin(b1)**2)*(gamma*M**2*np.sin(b1)**2 + 1.0)

def beta(M, theta, gamma=1.4):

    # Assuming theta is in radians
    gamma = 1.4
    delta = 1.0  # weak soln, delta = 0 is strong solution

    # Equations
    l = ((M**2 - 1.0)**2 - 3.0*(1.0 + (gamma - 1.0)/2.0*M**2)*
```

```
        (1.0 + (gamma+1.0)/2.0*M**2)*(np.tan(theta))**2)**(1.0/2.0)

    Chi = ((M**2 - 1)**3 - 9*(1 + (gamma-1)/2.0*M**2)*(1 + (gamma-1)/2.0*M**2 +
      (gamma+1)/4*M**4)*(np.tan(theta))**2)/l**3;
    N = M**2 - 1 + 2.0*l*np.cos((4*np.pi*delta + np.arccos(Chi))/3.0);
    D = 3.0*(1 + (gamma-1)/2.0*M**2)*np.tan(theta);
    result = N/D;

    return np.arctan2(N,D)
```

```
#!/usr/bin/env python

import os
import sys
import numpy as np
import oblique_relations
import matplotlib.pyplot as plt
from matplotlib import rc
from mpl_toolkits.axes_grid1 import host_subplot
import mpl_toolkits.axisartist as AA

fs = 14
rc('font',**{'family':'serif','serif':['Palatino']})
rc('text', usetex=True)
rc('legend',**{'fontsize':fs,'numpoints':1})
rc('axes',**{'labelsize':fs})

class theta_beta_m_profile:

    def __init__(self, M, theta):
        self.M = M
        self.theta = theta
        self.beta_l = []
        ctr = 0
        delete_indices = []
        for th in self.theta:
            btmp = oblique_relations.beta(M, th)
            if btmp.imag == 0.0 and ctr != 0:
                self.beta_l.append(btmp)
            else:
                delete_indices.append(ctr)
            ctr += 1
        self.theta = np.delete(self.theta, delete_indices)

# Inputs
Mmin = 2.0
Mmax = 20.0
M = np.array([2.0, 3.0, 4.0, 5.0, 10.0, 15.0, 20.0])
theta = np.linspace(0.0, 90.0, 500)
theta_rad = np.pi/180.0*theta

# Filling in beta
obj_list = []
for Mnum in M:
    obj_list.append(theta_beta_m_profile(Mnum, theta_rad))

# Plotting
fig1 = plt.figure(figsize=(7.0, 7.0/1.3))
for o in obj_list:
  plt.plot(o.theta*180.0/np.pi, np.array(o.beta_l)*180.0/np.pi,
      label=r"$M_\infty \ = \ {:2.0f}$".format(o.M), lw=1.5)

plt.xlabel(r"Deflection angle $\theta$, deg")
plt.ylabel(r"Shock-wave angle $\beta$, deg")
plt.legend(loc=4)
plt.ylim([0.0, 70.0])
plt.grid()
```

```
plt.tight_layout()

# Finding beta - theta for different Mach numbers
Mv = np.linspace(2.0, 20.0, 500)
diff = []
th = 20.0
for Mnum in Mv:
    b = oblique_relations.beta(Mnum, th*np.pi/180.0)
    diff.append(b*180.0/np.pi - th)

# Plotting beta - theta
fig2 = plt.figure(figsize=(5.0, 5.0/1.3))
plt.plot(Mv, diff)
plt.xlabel(r"$M_\infty$")
plt.ylabel(r"$\beta - \theta$ [deg]")
plt.xlim([1.5, 20.5])
plt.grid()
plt.tight_layout()

# Finding pressure, density and temperature ratios for same
# Mach number range
p2qp1 = []
rho2qrho1 = []
T2qT1 = []
for Mnum in Mv:
    p2qp1.append(oblique_relations.p_ratio(Mnum, th*np.pi/180.0))
    rho2qrho1.append(oblique_relations.rho_ratio(Mnum, th*np.pi/180.0))
    T2qT1.append(oblique_relations.T_ratio(Mnum, th*np.pi/180.0))

# Plotting
fig3 = plt.figure(figsize=(8.0, 7.0/1.3))
host = host_subplot(111, axes_class=AA.Axes)
plt.subplots_adjust(right=0.75)
par1 = host.twinx()
par2 = host.twinx()
hoffset = 60
new_fixed_axis = par2.get_grid_helper().new_fixed_axis
par2.axis["right"] = new_fixed_axis(loc="right", axes=par2,
    offset=(hoffset, 0))

par2.axis["right"].toggle(all=True)

host.set_xlim(1.9, 20.1)
host.set_ylim(0.0, 85.0)

host.set_xlabel(r"$M_\infty$")
host.set_ylabel(r"$p_2/p_1$")
par1.set_ylabel(r"$\rho_2/\rho_1$")
par2.set_ylabel(r"$T_2/T_1$")

par2.set_ylim(0, 25)

p1, = host.plot(Mv, p2qp1, "-r", lw=1.5)
p2, = par1.plot(Mv, rho2qrho1, "-b", lw=1.5)
p3, = par2.plot(Mv, T2qT1, "-g", lw=1.5)

host.axis["left"].label.set_color(p1.get_color())
par1.axis["right"].label.set_color(p2.get_color())
par2.axis["right"].label.set_color(p3.get_color())

plt.draw()

# Plotting pressure, density and temperature ratios
#plt.plot(Mv, p2qp1)
#plt.plot(Mv, rho2qrho1)
#plt.plot(Mv, T2qT1)
#plt.xlabel(r"$M_\infty$")

```

```
126
127 # Saving plots
128 if not os.path.isdir("../images/"):
129     os.mkdir("../images")
130 fig1.savefig("../images/theta-beta-M.pdf")
131 fig2.savefig("../images/hw1prob1a.pdf")
132 fig3.savefig("../images/hw1prob1b.pdf")
133
134 plt.show()
```

```
1  #!/usr/bin/env python
2
3  import os
4  import sys
5  from viscosity import mu
6  import numpy as np
7
8  # Inputs
9  gamma = 1.4
10 R = 286.9                    # J/(kg K)
11 L = 32.8                     # m
12 cp = gamma*R/(gamma - 1.0)  # J/(kg K)
13 V = np.array([8047.0, 5133.0, 3130.0, 1208.0])
14 z = np.array([74980.0, 60655.0, 49378.0, 30480.0])
15
16 # Data from www.digitaldutch.com/atmoscalc
17 T = np.array([206.690, 243.616, 270.650, 227.130])
18 p = np.array([2.07477, 18.5378, 82.1477, 1090.16])
19
20 # Assuming gas is calorically perfect
21 rho = p/(R*T)
22 M  = V/np.sqrt(gamma*R*T)
23 Re = rho*V*L/mu(T)
24 q = 0.5*rho*V**2
25 KE = 0.5*V**2
26 h0 = cp*T + 0.5*V**2
27 p0 = p + q
28 cp = (p0 - p)/q
29
30 print("{0:<3} {1:<3} {2:<3} {3:<3} {4:<3} {5:<3} {6:<3} {7:<3}"
31     .format("Rho","M","Re","q","KE","h0","p0","cp"))
32 for ri,mi,rei,qi,kei,h0i,p0i,cpi in zip(rho,M,Re,q,KE,h0,p0,cp):
33     print("{0:<1.6f} {1:<3.4f} {2:<6.1f} {3:<4.3f} {4:<5.2f} {5:<7.0f} {6:<5.2f} {7:<1.0f}"
34         .format(ri, mi, rei, qi, kei, h0i, p0i, cpi))
```

```
1  #!/usr/bin/env python
2
3  import numpy as np
4  import matplotlib.pyplot as plt
5  from matplotlib import rc
6  from mpl_toolkits.axes_grid1 import host_subplot
7  import mpl_toolkits.axisartist as AA
8
9  fs = 14
10 rc('font',**{'family':'serif','serif':['Palatino']})
11 #rc('font', family='Times New Roman')
12 rc('text', usetex=True)
13 rc('legend',**{'fontsize':fs,'numpoints':1})
14 rc('axes',**{'labelsize':fs})
15
16 class trajectory:
17
18     def __init__(self, initial_altitude, initial_velocity, S, m, g,
19         drag_coefficient, a, rho0, theta, output=False):
20         self.h0 = initial_altitude     # m
21         self.V0 = initial_velocity     # m/s
```

```
        self.S = S                      # m^2
        self.m = m                      # kg
        self.g = g                      # m/s^2
        self.earth_g = 9.81             # m/s^2
        self.Cd = drag_coefficient      # dimensionless
        self.a = a                      # 1/m
        self.rho0 = rho0                # kg/m^3
        self.theta = theta              # radians
        self.h_at_max_decel = 1.0/a*np.log(self.S*self.Cd*self.rho0/(self.a*self.m*
          np.sin(self.theta)))
        self.ballistic_coefficient = self.m*self.g/(self.Cd*self.S)
        self.V_at_max_decel = self.velocity(self.h_at_max_decel)
        self.max_decel = self.deceleration(self.h_at_max_decel)
        if output:
            print("Altitude at max deceleration: {0:5.0f} m".format(self.h_at_max_decel))
            print("Velocity at max deceleration: {0:5.0f} m/s".format(self.V_at_max_decel))
            print("Max deceleration: {0:5f} earth gs".format(self.max_decel/self.earth_g))
            print("Ballistic coefficient: {0:3.2f} N/m^2".format(self.ballistic_coefficient))

    def velocity(self, h):
        return self.V0*np.exp(-self.S*self.Cd*self.rho0/(2.0*self.a*self.m*np.sin(self.theta))*
            (np.exp(-self.a*h) - np.exp(-self.a*self.h0)))

    def deceleration(self, h):
        return - self.Cd*self.S*self.V0**2*self.rho0/(2.0*self.m)*np.exp(-self.a*h)*np.exp(-self.Cd*self.S*self.rho0/(

    def percent_diff(self, other):
        beta_diff = -(self.ballistic_coefficient - other.ballistic_coefficient
            )/self.ballistic_coefficient
        amax_diff = -(self.max_decel - other.max_decel)/self.max_decel
        Vamax_diff = -(self.V_at_max_decel - other.V_at_max_decel)/self.V_at_max_decel
        hmax_diff = -(self.h_at_max_decel - other.h_at_max_decel)/self.h_at_max_decel
        return (beta_diff, amax_diff, Vamax_diff, hmax_diff)

#####################################################################
# Part A
#####################################################################

initial_alt = 125.0e3
initial_velocity = 5.45e3
reference_area = np.pi*(2.65/2.0)**2
mass = 830.0
mars_g = 3.75
drag_coeff = 1.65
a = 1.275e-4
rho0 = 0.057
theta = 5.0*np.pi/180.0

base_trajectory = trajectory(initial_alt, initial_velocity, reference_area, mass,
    mars_g, drag_coeff, a, rho0, theta, output=True)

h_vec = np.linspace(10.0e3, 125.0e3, 1000)
V_vec = base_trajectory.velocity(h_vec)
a_vec = base_trajectory.deceleration(h_vec)

fig1 = plt.figure(figsize=(4.5, 4.5/1.3))
plt.plot(V_vec/1.0e3, h_vec/1.0e3, lw=1.5)
plt.ylim([10.0, 125.0])
plt.xlabel("Velocity [km/s]")
plt.ylabel("Altitude [km]")
plt.tight_layout()

fig2 = plt.figure(figsize=(4.5, 4.5/1.3))
plt.plot(a_vec/base_trajectory.earth_g, h_vec/1.0e3, lw=1.5)
plt.xlabel("Acceleration [Earth g's]")
plt.ylabel("Altitude [km]")
plt.ylim([10.0, 125.0])
plt.tight_layout()
```

```
90
91  fig1.savefig("../images/velocity_altitude.pdf")
92  fig2.savefig("../images/velocity_acceleration.pdf")
93
94  ################################################################
95  # Part B
96  ################################################################
97
98  # Defining ''perturbations''
99  percent_change = 1.05
100 dinitial_alt = 125.0e3*percent_change
101 dinitial_velocity = 5.45e3*percent_change
102 dreference_area = np.pi*(2.65*percent_change/2.0)**2
103 dmass = 830.0*percent_change
104 dmars_g = 3.75*percent_change
105 ddrag_coeff = 1.65*percent_change
106 da = 1.275e-4*percent_change
107 drho0 = 0.057*percent_change
108 dtheta = 5.0*np.pi/180.0*percent_change
109
110 # Computing perturbed trajectories
111 perturb_alt = trajectory(dinitial_alt, initial_velocity, reference_area, mass,
112     mars_g, drag_coeff, a, rho0, theta)
113 perturb_vel = trajectory(initial_alt, dinitial_velocity, reference_area, mass,
114     mars_g, drag_coeff, a, rho0, theta)
115 perturb_area = trajectory(initial_alt, initial_velocity, dreference_area, mass,
116     mars_g, drag_coeff, a, rho0, theta)
117 perturb_mass = trajectory(initial_alt, initial_velocity, reference_area, dmass,
118     mars_g, drag_coeff, a, rho0, theta)
119 perturb_g = trajectory(initial_alt, initial_velocity, reference_area, mass,
120     dmars_g, drag_coeff, a, rho0, theta)
121 perturb_drag = trajectory(initial_alt, initial_velocity, reference_area, mass,
122     mars_g, ddrag_coeff, a, rho0, theta)
123 perturb_a = trajectory(initial_alt, initial_velocity, reference_area, mass,
124     mars_g, drag_coeff, da, rho0, theta)
125 perturb_rho0 = trajectory(initial_alt, initial_velocity, reference_area, mass,
126     mars_g, drag_coeff, a, drho0, theta)
127 perturb_theta = trajectory(initial_alt, initial_velocity, reference_area, mass,
128     mars_g, drag_coeff, a, rho0, dtheta)
129
130 # Finding percent difference for each perturbation
131 diff_list = []
132 diff_list.append(base_trajectory.percent_diff(perturb_alt))
133 diff_list.append(base_trajectory.percent_diff(perturb_vel))
134 diff_list.append(base_trajectory.percent_diff(perturb_area))
135 diff_list.append(base_trajectory.percent_diff(perturb_mass))
136 diff_list.append(base_trajectory.percent_diff(perturb_g))
137 diff_list.append(base_trajectory.percent_diff(perturb_drag))
138 diff_list.append(base_trajectory.percent_diff(perturb_a))
139 diff_list.append(base_trajectory.percent_diff(perturb_rho0))
140 diff_list.append(base_trajectory.percent_diff(perturb_theta))
141
142 # Printing table
143 perturbation_list = ["initial altitude", "initial velocity", "diameter",
144 "mass", "gravitational acceleration", "drag", "$a$", "$\\rho_0$", "$\\theta$"]
145 with open("hw1prob3b_table.dat", "w") as t:
146     t.write("{:28} {:5} {:5} {:5} {:5}\n".format("perturbation type", "ballistic_coefficient", \
147       "$a_\\text{max}$", "$V$ at $a_\\text{max}$", "$h$ at $a_\\text{max}$"))
148     for i in range(0, len(perturbation_list)):
149         t.write("{:28} ".format(perturbation_list[i]))
150         for v in diff_list[i]:
151             t.write("{:1.2e} ".format(v))
152         t.write("\n")
153
154 plt.show()
```