Problem 1. Trapped Particle Radiation

Assume two spacecraft orbit Earth, one at an altitude of 3000km, the other at 20,000km.

Part (a)

Remind yourself — are these orbits above or below geostationary?

> A geostationary orbit (GEO) is a circular geosynchronous orbit in the plane of the Earth's equator with a radius of approximately 42,164 km (measured from the center of the Earth). A satellite in such an orbit is at an altitude of approximately 35,786 km above mean sea level[1].

As such, both of these satellites are below geostationary.

Part (b)

Assuming they are both shielded with 4mm aluminum and have the same electronics, which spacecraft is more likely to suffer Single-Event Upsets? Why?

The most likely place for a satellite to experience a Single-Event Upset is within the Van Allen radiation belts, and one of the safest regions for satellites to be is within the inner and out belt. The inner belt extends from 1,000 to 6,000 kilometers above the Earth, and the outer belt extends from 13,000 to 60,000 kilometers above the Earth's surface.

The inner belt contains high concentrations of electrons in the range of hundreds of keV and energetic protons with energies exceeding 100 MeV, trapped by the relatively strong magnetic fields in the region[2]. Due to this large flux, the satellite in the 3,000km altitude would be more likely to suffer Single-Event Upsets.

Problem 2. Upper Atmosphere Physics

Part (a)

What is the change in density of the exosphere at Hubble Space Telescope (HST) altitude between the date of its fourth Shuttle re-boost (Servicing Mission 3B) and the date of your new spacecrafts re-boost?

The HST is currently orbiting with a Perigee 551.4 km and an Apogee of 555.6 km[3]. Servicing Mission 3B took place in March 2002[4], which was during a solar maximum (see Figure 1). The date of our proposed mission is January 2020, and the next solar cycle is expected to be "one of the weakest in centures"[5], and should peak around 2022. While it's

[1] `https://en.wikipedia.org/wiki/Geosynchronous_orbit`

[2] Gusev, A. A.; Pugacheva, G. I.; Jayanthi, U. B.; Schuch, N. (2003). "Modeling of Low-altitude Quasi-trapped Proton Fluxes at the Equatorial Inner Magnetosphere". Brazilian Journal of Physics 33 (4): 775781. Bibcode:2003BrJPh..33..775G.

[3] "HST Satellite details 1990-037B NORAD 20580". N2YO. January 27, 2015. Retrieved January 27, 2015.

[4] https://en.wikipedia.org/wiki/STS-109

[5] `http://science.nasa.gov/science-news/science-at-nasa/2006/10may_longrange/`

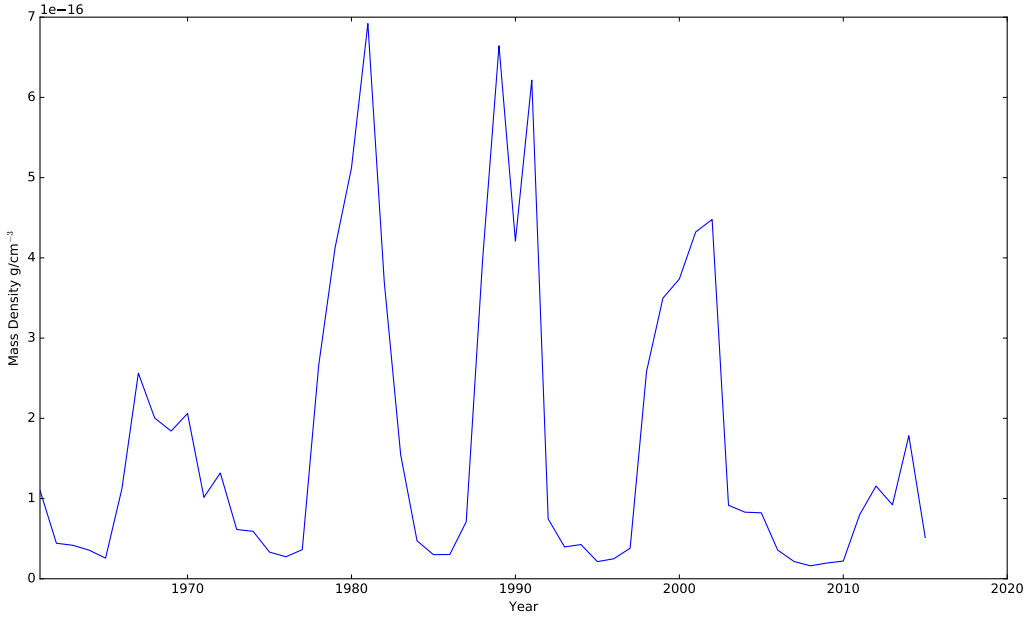

Figure 1: Mass density of exosphere from 1961-2015.

impossible to know what the density of the exosphere will be in 2020, we should expect the density to be 50% to 100% of what is was during Servicing Mission 3B.

Part (b)

Using the NASA NRLMSISE-100 atmosphere model, list the following parameters for todays date and the HST altitude:

(a) Atomic oxygen species density

(b) Atmosphere mass density

(c) Exosphere temperature

The most recent data available is from September 2015. The information for an altitude of 553km is:

(a) Atomic oxygen species density, 1.743E+06 cm^{-3}

(b) Atmosphere mass density, 5.419E-17 g/cm^{-3}

(c) Exosphere temperature, 810 K

Part (c)

What role do each of these environmental parameters play in the mission design of your spacecraft?

Atomic oxygen can erode the external surfaces of polymers on spacecraft, and degrade the internal components of exposed spacecraft components[6]. A larger atmosphere mass density will lead to additional particle collisions with the spacecraft, lower it's orbital velocity and decreasing it's orbit height. A higher exosphere temperature will cause the exosphere to expand and raise it's altitude. This has the same effect as if the spacecraft was in a lower orbit–resulting in higher density of particles.

Problem 3. Orbital Debris and Ballistic Limit Equations (BLE)

Part (a)

Using eqn 2-3 of the NASA MMOD handbook and Fig 2-4 of the NASA Orbital Debris Engineering Model (both in MMOD folder on SmartSite), determine the critical particle diameter for a Probability of No Penetration (PNP) of 0.877. Assume 1m^2 exposed area and a two-week mission.

See the appendix for the software used to create the solutions for this problem.

$$PNP = exp\left[-\sum_{i=1}^{n}(FAt)_i\right]$$

over n spacecraft regions, where F (number/m^2-year), of meteoroid and debris impacts that exceed the failure limits (or ballistic limits), the exposed area, A (m^2), and duration or time exposed to the MMOD flux, t (year). Plugging in the given values, we have

$$\begin{aligned}
PNP &= exp\left[-\sum_{i=1}^{n}(FAt)_i\right] \\
PNP &= exp\left[-(FAt)\right] \\
FAt &= ln(\frac{1}{PNP}) \\
F &= ln(\frac{1}{PNP})\frac{1}{At} \\
F &= ln(\frac{1}{0.877})\frac{1}{(1)(2*7*24*60*60)} \\
F &= 1.09e-07\ [1/m^2/s]
\end{aligned}$$

Looking at Figure 2-4, we find a diameter of $\approx$ 0.1mm.

[6]Banks, Bruce A., et al. "Atomic oxygen effects on spacecraft materials." Ninth International Symposium on Materials in a Space Environment, NASA TM-212484. 2003.

Part (b)

What would the critical particle diameter be for the same PNP if your spacecraft stayed attached to HST for a second re-boost in 2 years?

$$F = ln(\frac{1}{PNP})\frac{1}{At}$$
$$F = ln(\frac{1}{0.877})\frac{1}{(1)(2*365*24*60*60)}$$
$$F = 2.08e-09\ [1/m^2/s]$$

Looking at Figure 2-4, we find a diameter of $\approx$ 0.2mm.

Part (c)

Consider a perpendicular impact of a particle of the size found in part a and another particle with 1mm diameter. Use the Design Equations 4-1 and 4-2 to compute the penetration depth for:

(a) Silicate particle at 0.5g/cm^3 and 23 km/s (micro-meteroid)

(b) Steel particle at 7.8 g/cm^3 and 7 km/s (orbital debris)

With target materials of:

(a) 6061-T6 Alum

  (i) BHN: 95

  (ii) C_t: 5.433 km/s

  (iii) Density: 2.7 g/cm^3

(b) 7075-T6 Alum

  (i) BHN: 150

  (ii) C_t: 6.350 km/s

  (iii) Density: 2.81 g/cm^3

For $\frac{\rho_p}{\rho_t} < 1.5$,

$$P_\infty = 5.24d^{(19/18)}BHN^{-0.25}\left(\frac{\rho_p}{\rho_t}\right)^{0.5}\left(\frac{Vcos\theta}{C_t}\right)^{2/3}$$

For $\frac{\rho_p}{\rho_t} \geq 1.5$,

$$P_\infty = 5.24d^{(19/18)}BHN^{-0.25}\left(\frac{\rho_p}{\rho_t}\right)^{2/3}\left(\frac{Vcos\theta}{C_t}\right)^{2/3}$$

	Micro-meteroid	Orbital Debris
6061	0.166	0.355
7075	0.131	0.278

Table 1: P_∞ (cm) for combinations of particles and target materials

Part (d)

No protection case: Use Performance Equation 4-6 to estimate the required spacecraft wall thickness to avoid detached spall due to impact of the 1mm particle for the four material cases in part c.

$$d_c = \left[\frac{t}{k}\frac{BHN^{0.25}\left(\rho_t/\rho_p\right)^{0.5}}{5.24\left(Vcos\theta/C_t\right)^{2/3}}\right]^{18/19} \implies t = \left[\frac{d_c^{19/18}}{k}\frac{BHN^{0.25}\left(\rho_t/\rho_p\right)^{0.5}}{5.24\left(Vcos\theta/C_t\right)^{2/3}}\right]^{-1}$$

To avoid detached spall with an incidence angle of $\theta = 0$, this simplifies to:

$$t = \left[\frac{d_c^{19/18}}{11.528}\frac{\left(\rho_t/\rho_p\right)^{0.5}}{\left(V/C_t\right)^{2/3}}BHN^{0.25}\right]^{-1}$$

	Micro-meteroid	Orbital Debris
6061	255.192	29.234
7075	209.310	23.978

Table 2: t (cm) for combinations of particles and target materials

Part (e)

Compare your answers in part d to equation 4-4.

The shielding thickness will prevent detached spall, given that:

$$t \leq 2.2P_\infty$$

After investigating our four cases, we find

Part (f)

Whipple Shield design: For the two 1mm particle compositions/velocities in part c, use equations 4-21 and 4-22 to estimate the bumper and rear-wall thickness required to defeat the threat particles. Assume

	Micro-meteroid	Orbital Debris
6061	False	False
7075	False	False

Table 3: Results for combinations of particles and target materials

(a) Particles are spherical

(b) Bumper standoff = 10.2cm (Fig 4-1)

(c) Rear wall: 2219-T87 Alum, 0.5cm thickness (σ = 57000 psi = 57ksi)

(d) Perpendicular velocity impact ($\theta = 0$)

$$t_b = c_b d \rho_p / \rho_b$$
$$t_w = c_w d^{0.5} \left(\rho_p \rho_b\right)^{1/6} \left(M_p\right)^{1/3} V_n / S^{0.5} \left(70/\sigma\right)^{0.5}$$

where c_b depends on the ratio of $S/d = 10.2cm/0.1cm = 102 \implies c_b = 0.2$. Since the particles are spherical, $M_p = \rho_p V_p = \rho_p \frac{4}{3} \pi r_p^3$.

	Micro-meteroid	Orbital Debris
6061	0.004	0.058
7075	0.004	0.056

Table 4: t_b (cm), bumper thickness, for combinations of particles and target materials

	Micro-meteroid	Orbital Debris
6061	0.027	0.033
7075	0.027	0.033

Table 5: t_w (cm), rear wall thickness, for combinations of particles and target materials

Part (g)

Estimate the protection capability limits for your Whipple Shields: Compute the critical projectile performance diameter using equations 4-23, 4-24, and 4-25 for various relative

velocities, for two types of spherical particle, one silicate, the other steel. Assume the same parameters as for part f.

$$V_n \geq 7\text{km/s} \implies d_c = 3.918\, t_w^{2/3} \rho_p^{-1/3} \rho_b^{-1/9} \left(V cos\theta\right)^{-2/3} S^{1/3} \left(\sigma/70\right)^{1/3}$$

$$V_n \leq 3\text{km/s} \implies d_c = \left[\frac{t_w(\sigma/40)^{0.5} + t_b}{0.6(cos^{5/3}\theta)\rho_p^{0.5} V^{2/3}}\right]^{(18/19)}$$

$$3\text{km/s} \leq V_n \leq 7\text{km/s} \implies d_c = \left[\frac{t_w(\sigma/40)^{0.5} + t_b}{1.248\rho_p^{0.5}(cos\theta)}\right]^{(18/19)} * \left[1.75 - (V_n/4)\right] + \left[1.071\, t_w^{2/3} \rho_p^{-1/3} \rho_b^{-1/9} S^{1/3} \left(\sigma/70\right)^{1/3}\right] * (V_n/4 - 0.75)$$

	Micro-meteroid	Orbital Debris
1km/s	0.015	0.045
5km/s	0.047	0.055
10km/s	0.099	0.099
25km/s	0.099	0.099
100km/s	0.099	0.099

Table 6: d_c (cm), critical projectile diameter at shield failure threshold, for combinations of particles impacting 6061-T6 Alum and a rear wall of 2219-T87 Alum

Problem 4. Acoustic Shielding

Part (a)

Download and install a sound-level app on your smartphone.

Instead of using a smartphone app, I used a portable microphone and connected it to my laptop. This allowed me to have better control over the data and analysis.

Part (b)

Measure the sound pressure level (dB) in some noisy environment (home stereo? EFL?).

Five seconds of recorded audio from an African chant[7] was recorded in front of a speaker in my living room, measuring an average volume of -13.8 db (see part c).

Part (c)

Remind yourself: what is the definition of the Decibel, and how is the reference soundpressure level defined?

[7] See `https://www.youtube.com/watch?v=RRgEBogpa9w`.

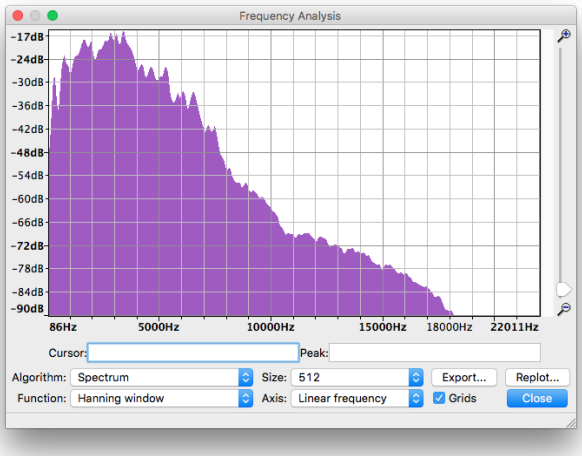

Figure 2: Frequency Analysis of first 5-second block of recorded audio (no shielding).

The decibel (dB) is a logarithmic unit used to express the ratio of two values of a physical quantity. Sound volume can be measured by

$$L_p = 20 \log_{10}\left(\frac{p_{\text{rms}}}{p_{\text{ref}}}\right) \text{ dB},$$

where p_{ref} is the standard reference sound pressure of 20 micropascals in air, and p_{rms} is the RMS value of the measured sound. The reference pressure in air is approximately the threshold of hearing for an average human (dbSPL). The scale I use here, however, is decibels relative to full scale (dBFS). For dbFS, 0 is assigned to a maximum digital level, the point at which clipping occurs. For a rough comparison, a measurement of -13.8 dBFS on my system is approximately 80 dB SPL.

Part (d)

Use a Styrofoam cup (or similar) as a payload shroud for your phone. Plug the open end of the shroud with isolation (tissues?). Measure the change in dB sensed by your phone.

A fuzzy slipper was used as a payload shroud, and the microphone was placed the same distance away from the speaker as before. The African chant was again played for five seconds, and a volume of -24.4db was measured. The resulting difference was 10.6 db average RMS.

Part (e)

Add some kind of insulation to the inside walls of your payload shroud and repeat part d.

A sweater was wrapped around the microphone, which was again placed inside the fuzzy

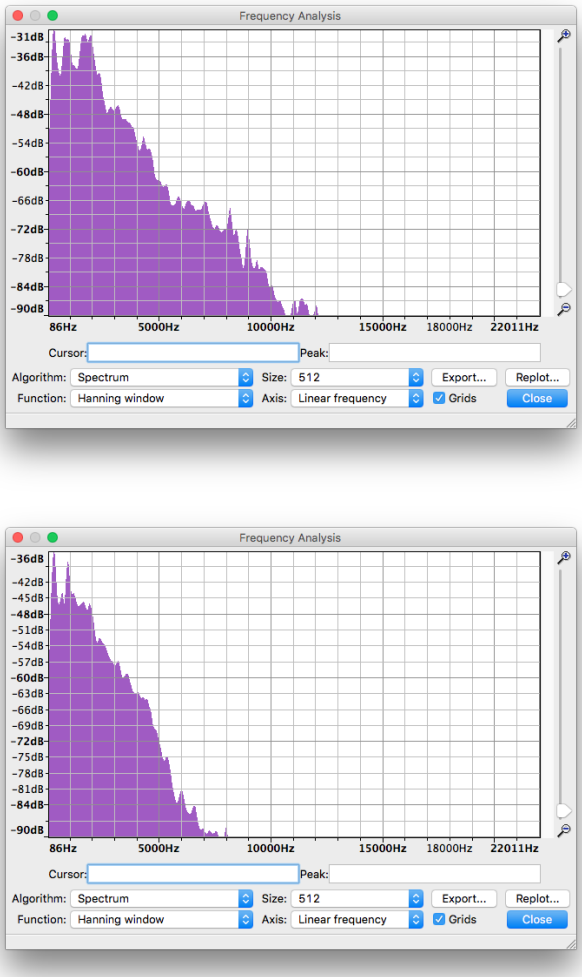

Figure 3: Frequency Analysis of second (some shielding), and third (most shielding) 5-second blocks of recorded audio.

slipper. The African chant was once again played for five seconds, and a volume of -30.2 db was measured. The resulting difference was 5.8 db average RMS.

Part (f)

Discuss results.

'Insulating' our 'payload shroud' reduced the initial volume of -13.8 dB to -30.2 db. Higher frequency wavelengths were especially affected, and the maximum frequency observed at a volume greater than -90 dB decreased from 18000 Hz in the unshield case, to approximately 7500 Hz in the most shielded case. (Depending on your PDF reader, there may or may not be text here: Play Sound, you can press 'Play Sound' to play the three 5-second blocks of recorded audio.)

Problem 5. Numerical Integration Review

Consider the first-order initial-value problem: $\frac{dy}{dt} = t + y$, $y(0) = 0$ with exact solution $y(t) = e^t - t - 1$

Part (a)

Program Euler and Runge-Kutta solvers (write your own) and plot the results over a range t=0 to 1.0, with step sizes h=0.01, 0.1, and 0.5. Plot results (y vs t) and compare to the exact solution.

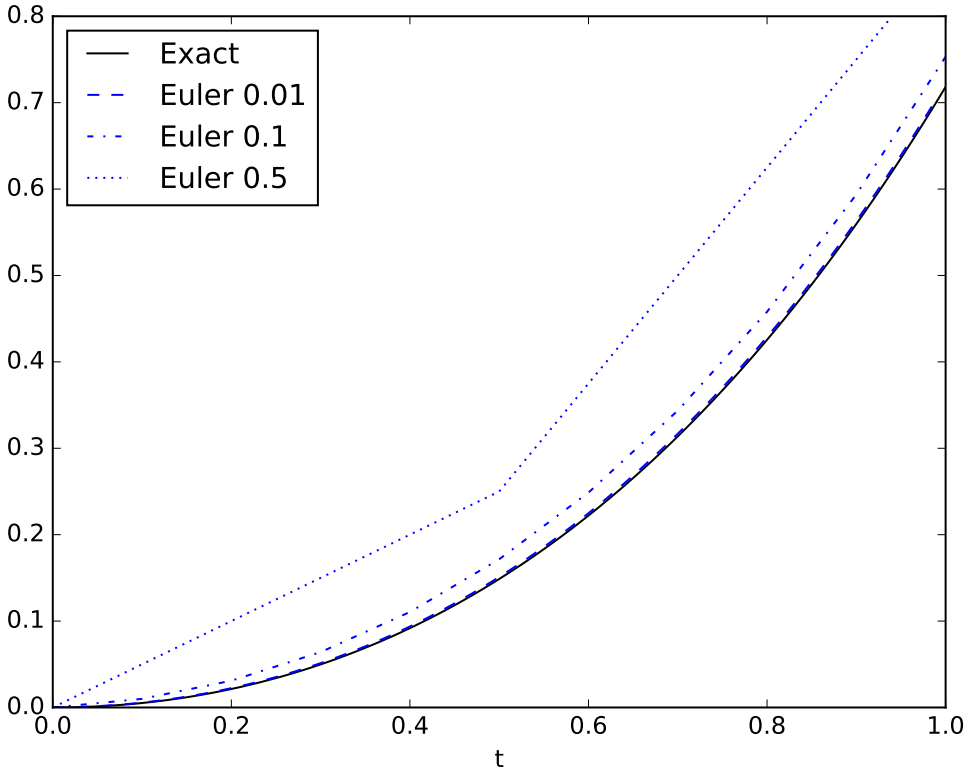

Figure 4: Results from Euler solver.

Part (b)

Repeat step a with library functions for Euler and RK4

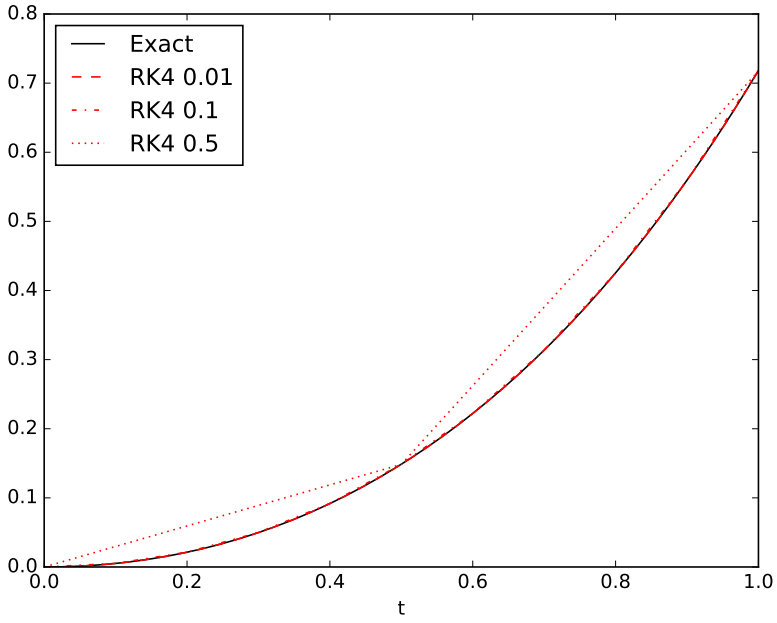

Figure 5: Results from RK4 solver.

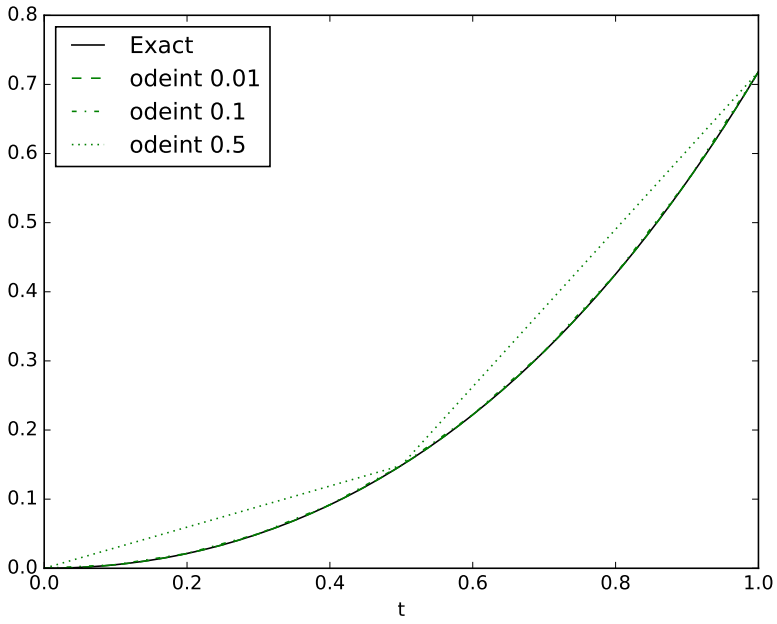

Figure 6: Results from scipy's RK4 solver.

Appendix A

Python Code

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.integrate.odepack import odeint
from tabulate import tabulate


################################################################################
def problem4():
    # Two impact types
    micro_meteroid = {'density': 0.5,
                      'velocity': 23,
                      'diameter': 0.1}

    orbital_debris = {'density': 7.8,
                      'velocity': 7,
                      'diameter': 0.1}

    # Two target materials
    alum_6061_t6 = {'BHN': 95,
                    'C_t': 5.433,
                    'density': 2.7}
    alum_7075_t6 = {'BHN': 150,
                    'C_t': 6.350,
                    'density': 2.81}

    def latex_output(func, theta=0):
        theta = np.deg2rad(theta)

        try:
            micro_6061 = "{:.3f}".format(func(micro_meteroid, alum_6061_t6, theta=theta))
            micro_7075 = "{:.3f}".format(func(micro_meteroid, alum_7075_t6, theta=theta))
            orbital_6061 = "{:.3f}".format(func(orbital_debris, alum_6061_t6, theta=theta
    ))
            orbital_7075 = "{:.3f}".format(func(orbital_debris, alum_7075_t6, theta=theta
    ))

            headers = ["Micro-meteroid", "Orbital Debris"]
```

```
            table = [['6061', micro_6061, orbital_6061],
                     ['7075', micro_7075, orbital_7075]]
        except TypeError:
            res = []
            for material in [micro_meteroid, orbital_debris]:
                for velocity in [1, 5, 10, 25, 100]:
                    temp = material.copy()
                    temp['velocity'] = velocity
                    r = "{:.3f}".format(func(temp, alum_6061_t6, bumper, theta=theta))
                    res.append(r)

            headers = ["Micro-meteroid", "Orbital Debris"]
            table = [['1km/s',   res[0], res[5]],
                     ['5km/s',   res[1], res[6]],
                     ['10km/s',  res[2], res[7]],
                     ['25km/s',  res[3], res[8]],
                     ['100km/s', res[4], res[9]]]

        print(tabulate(table, headers, tablefmt='latex'))

    def P_inf(particle, target, theta=0):
        theta = np.deg2rad(theta)
        if particle['density'] / target['density'] < 1.5:
            exp = 0.5
        else:
            exp = 2/3

        return 5.24 \
               * particle['diameter'] ** (19/18) \
               * target['BHN'] ** (-0.25) \
               * (particle['density']/target['density']) ** (exp) \
               * ((particle['velocity'] * np.cos(theta))/target['C_t']) ** (2/3)

    def wall_thickness(particle, target, theta=0):
        theta = np.deg2rad(theta)
        # for detached spall
        k = 2.2

        n = particle['diameter'] ** (19/18) \
            * target['BHN'] ** (0.25) \
            * (particle['density']/target['density']) ** (0.5)

        d = k * 5.24 \
            * ((particle['velocity'] * np.cos(theta))/target['C_t']) ** (2/3)

        return d/n

    def compare_t_and_p_inf(particle, target, theta=0):
        theta = np.deg2rad(theta)
        t = wall_thickness(particle, target, theta=0)
        p_inf = P_inf(particle, target, theta=0)
        comparison = t <= 2.2 * p_inf
        return comparison

```

```
    bumper = {'S': 10.2,
              'sigma': 57,
              'density': 2.84}

    def t_b(particle, target, theta=0):
        theta = np.deg2rad(theta)
        c_b = 0.2
        return c_b * particle['diameter'] \
               * (particle['density']/target['density'])

    def t_w(particle, target, bumper, theta=0):
        theta = np.deg2rad(theta)
        c_w = 0.16
        V_p = ((4/3) * np.pi * (particle['diameter']/2) ** 3)
        M_p = particle['density'] * V_p
        V_n = particle['velocity'] * np.cos(theta)
        return c_w * particle['diameter'] ** (0.5) \
               * (particle['density'] * target['density']) ** (1/6) \
               * (M_p) ** (1/3) \
               * (V_n/(bumper['S'] ** (0.5))) \
               * (70/bumper['sigma'])**0.5

    def d_c(particle, target, bumper, theta=0):
        theta = np.deg2rad(theta)
        tb = t_b(particle, target, theta)
        tw = t_w(particle, target, bumper, theta)

        if particle['velocity'] >= 7:
            dc = 3.918 * tw**(2/3) \
                 * particle['density']**(-1/3) \
                 * bumper['density']**(-1/9) \
                 * (particle['velocity'] * np.cos(theta))**(-2/3) \
                 * bumper['S']**(1/3) * (bumper['sigma']/70)**(1/3)
        elif particle['velocity'] <= 3:
            n = tw * (bumper['sigma']/40)**(0.5) + tb
            d = 0.6 * (np.cos(theta))**(5/3) \
                * particle['density']**(0.5) \
                * particle['velocity']**(2/3)
            dc = (n / d) ** (18/19)
        else:
            v_n = particle['velocity'] * np.cos(theta)
            dc_f = 1.071 * tw**(2/3) \
                   * particle['density']**(-1/3) \
                   * bumper['density']**(-1/9) \
                   * bumper['S']**(1/3) * (bumper['sigma']/70)**(1/3)

            n = tw * (bumper['sigma']/40)**(0.5) + tb
            d = 1.248 * particle['density']**(0.5) \
                * (np.cos(theta))
            dc_s = (n / d) ** (18/19)

            dc = dc_f * (v_n/4 - 0.75) + dc_s * (1.75 - v_n/4)

        return dc
```

```
145
146 ##############################################################################
147 def prob5():
148     def euler(f, y0, a, b, h):
149         t, y = a, y0
150         res = []
151         while t <= b:
152             res.append([t, y])
153             t += h
154             y += h * f(t, y)
155         return res
156
157     def rk4(f, y0, a, b, h):
158         n = int((b - a)/h)
159         vx = [0]*(n + 1)
160         vy = [0]*(n + 1)
161         vx[0] = x = a
162         vy[0] = y = y0
163         for i in range(1, n + 1):
164             k1 = h*f(x, y)
165             k2 = h*f(x + 0.5*h, y + 0.5*k1)
166             k3 = h*f(x + 0.5*h, y + 0.5*k2)
167             k4 = h*f(x + h, y + k3)
168             vx[i] = x = a + i*h
169             vy[i] = y = y + (k1 + k2 + k2 + k3 + k3 + k4)/6
170         return vx, vy
171
172     def func(t, y):
173         return t + y
174
175     xs = np.concatenate([np.arange(0, 1., 0.01), [1]])
176     ys = [np.exp(x) - x - 1 for x in xs]
177     exact = pd.DataFrame([xs, ys]).T
178     exact.columns = ['t', 'y']
179     hs = [0.01, 0.1, 0.5]
180     styles = ['--', '-.', ':']
181
182     f, ax = plt.subplots()
183     exact.plot(x='t', y='y', ax=ax, style='k', label='Exact')
184     for h, s in zip(hs, styles):
185         res = euler(func, 0, 0, 1, h)
186         res = pd.DataFrame(res)
187         res.columns = ['t', 'y']
188         res.plot(x='t', y='y', ax=ax, style='b' + s, label='Euler ' + str(h))
189     plt.ylim(0, 0.8)
190     plt.savefig('euler.pdf')
191
192     f, ax = plt.subplots()
193     exact.plot(x='t', y='y', ax=ax, style='k', label='Exact')
194     for h, s in zip(hs, styles):
195         vx, vy = rk4(func, 0, 0, 1, h)
196         res = pd.DataFrame([vx, vy]).T
197         res.columns = ['t', 'y']
198         res.plot(x='t', y='y', ax=ax, style='r' + s, label='RK4 ' + str(h))
```

```
    plt.ylim(0, 0.8)
    plt.savefig('rk4.pdf')

    f, ax = plt.subplots()
    exact.plot(x='t', y='y', ax=ax, style='k', label='Exact')
    y0, t0, t1 = 0, 0, 1
    for h, s in zip(hs, styles):
        t = np.concatenate([np.arange(t0, t1, h), [t1]])
        y = odeint(func, [y0], t)[:, 0]
        res = pd.DataFrame([t, y]).T
        res.columns = ['t', 'y']
        res.plot(x='t', y='y', ax=ax, style='g' + s, label='odeint ' + str(h))
    plt.ylim(0, 0.8)
    plt.savefig('odeint.pdf')
```