
AMAT

Release v2.2.1

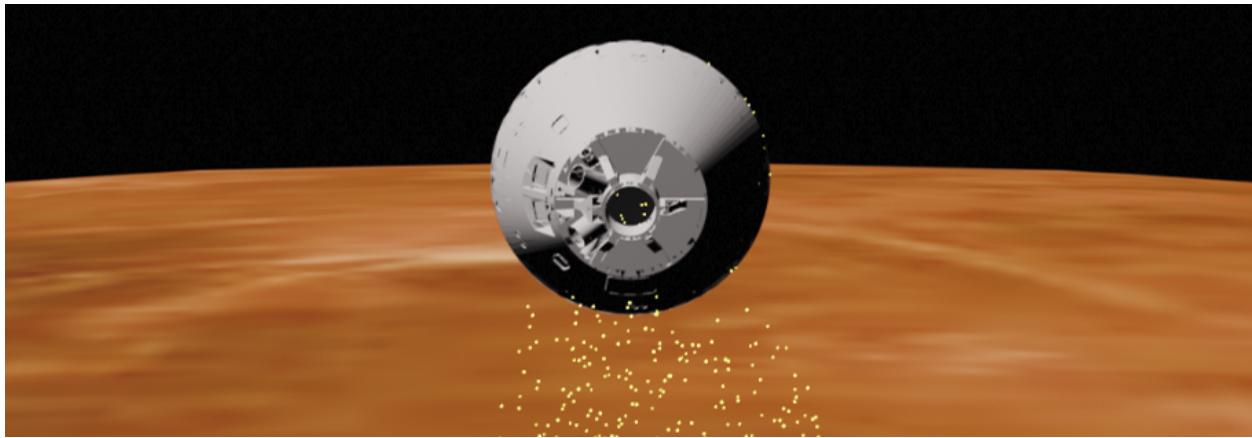
Athul P. Girija

Nov 04, 2021

CONTENTS:

1	About AMAT	3
1.1	Overview	3
1.2	History	3
1.3	Related software	4
1.4	Future ideas	5
1.5	Note from the original author	5
2	Installation	7
2.1	Option 1 : Install from pip (recommended)	7
2.2	Option 2 : Install from source	8
2.3	Option 3 : Install in a virutalenv	9
2.4	AMAT Usage	9
3	Capabilities	11
3.1	Sample results	11
3.2	What kind of problems can AMAT solve?	13
4	Example Jupyter Notebooks	15
4.1	Example - 01 - Hello World!	15
4.2	Example - 02 - Atmosphere	15
4.3	Example - 03 - Venus Aerocapture: Part 1	16
4.4	Example - 04 - Venus Aerocapture: Part 2	21
4.5	Example - 05 - Titan Aerocapture: Part 1	25
4.6	Example - 06 - Titan Aerocapture: Part 2	32
4.7	Example - 07 - Venus Aerocapture: Part 3	39
4.8	Example - 08 - Venus Aerocapture: Part 4	47
4.9	Example - 09 - Uranus Aerocapture	55
4.10	Example - 10 - Neptune Aerocapture - Part 1	63
4.11	Example - 11 - Neptune Aerocapture - Part 2a: Monte Carlo Simulations	72
4.12	Example - 12 - Neptune Aerocapture - Part 2b: Monte Carlo Simulations	78
4.13	Example - 13 - Venus Aerocapture - Part 5a: Monte Carlo Simulations	83
4.14	Example - 14 - Fire-II - Earth	87
4.15	Example - 15 - Apollo-AS-201 - Earth	89
4.16	Example - 16 - Apollo-AS-202 - Earth	92
4.17	Example - 17 - Apollo-4 - Earth	94
4.18	Example - 18 - Apollo-6 - Earth	96
4.19	Example - 19 - PAET - Earth	98
4.20	Example - 20 - Viking-1 - Mars	101
4.21	Example - 21 - Viking-2 - Mars	103
4.22	Example - 22 - PV Small North - Venus	105

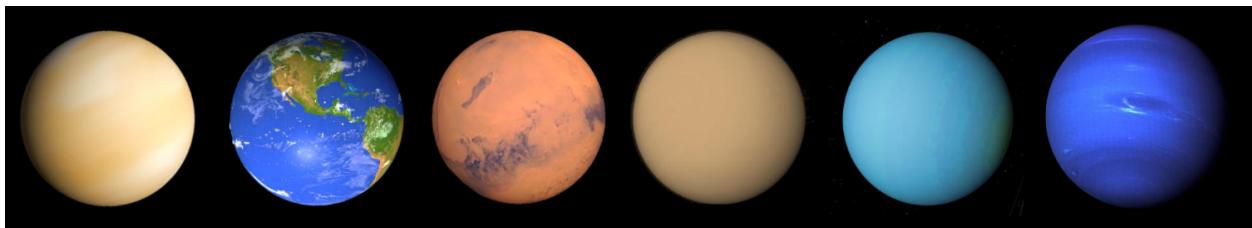
4.23	Example - 23 - PV Small Night - Venus	107
4.24	Example - 24 - PV Small Day - Venus	110
4.25	Example - 25 - PV Large - Venus	112
4.26	Example - 26 - Galileo - Jupiter	114
4.27	Example - 27 - OREX - Earth	116
4.28	Example - 28 - Pathfinder - Mars	119
4.29	Example - 29 - Mirka - Earth	121
4.30	Example - 30 - Huygens - Titan	123
4.31	Example - 31 - Atm. Reentry Demonstrator - Earth	125
4.32	Example - 32 - Deep Space 2 - Mars	128
4.33	Example - 33 - Stardust - Earth	130
4.34	Example - 34 - Genesis - Earth	132
4.35	Example - 35 - Hayabusa - Earth	134
4.36	Example - 36 - Beagle-2 - Mars	137
4.37	Example - 37 - Opportunity - Mars	139
4.38	Example - 38 - Curiosity - Mars	141
4.39	Example - 39 - Oceanus - Saturn (Concept)	143
4.40	Example - 40 - Oceanus - Uranus (Concept)	146
4.41	Example - 41 - Hera - Saturn (Concept)	148
4.42	Example - 42 - Dragonfly - Titan (Planned)	151
4.43	Example - 43 - Ice Giant Pre-Decadal - Uranus Probe (Concept)	153
4.44	Example - 44 - Ice Giant Pre-Decadal - Neptune Probe (Concept)	156
4.45	Example - 45 - ADEPT ViTaL - Venus (Concept)	158
4.46	Example - 46 - Uranus Probe - Decadal Study - 2010 (Concept)	161
4.47	Example - 47 - Venus Flagship - Decadal Study - 2010 (Concept)	164
4.48	Example - 48 - Crew Module Atmospheric Re-entry Experiment	167
4.49	Example - 49 - Earth SmallSat Aerocapture Demonstration - Part 1	170
4.50	Example - 50 - Earth SmallSat Aerocapture Demonstration - Part 2	176
4.51	Example - 51 - Mars SmallSat Aerocapture Demonstration - Part 1	188
4.52	Example - 52 - Mars SmallSat Aerocapture Demonstration - Part 2	196
4.53	Example - 53 - Mars SmallSat Aerocapture Demonstration - Part 3	208
4.54	Example - 54 - ExoMars 2016 - Mars	221
4.55	Example - 55 - Titan Aerocapture Systems Study - Part 1	223
4.56	Example - 56 - Venus Entry Tradespace - Carpet Plot	231
4.57	Example - 57 - Dragnofly (Titan) Entry Tradespace - Carpet Plot	236
5	API Reference	243
6	Contributions	293
7	Credits	295
7.1	Extras	295
8	References	297
9	Module Index	299
10	Index	301
Python Module Index		303
Index		305



AMAT is designed to provide rapid mission analysis capability for aerocapture mission concepts to the planetary science community.

See [Jupyter notebooks](#) to get started or refer to [examples](#) in the GitHub repository.

AMAT allows the user to perform low-fidelity broad sweep parametric studies; as well as high fidelity Monte Carlo simulations to quantify aerocapture performance. AMAT comes with a suite of interplanetary trajectories, planetary atmosphere models, aeroheating correlations, and guidance algorithms for rapid conceptual mission design. AMAT provides aerocapture and Entry, Descent, Landing (EDL) mission analysis capability for Venus, Earth, Mars, Jupiter, Saturn, Titan, Uranus, and Neptune.



For sub-routine documentation, see [modindex](#)

**CHAPTER
ONE**

ABOUT AMAT

1.1 Overview

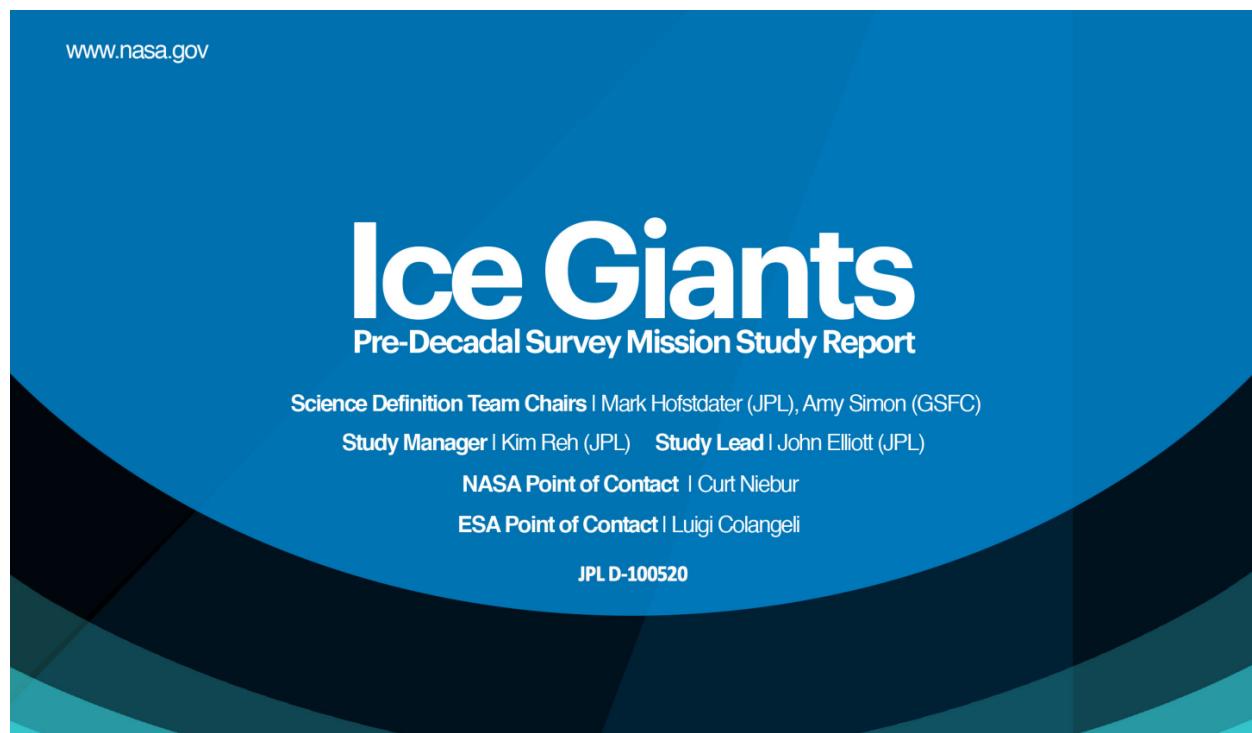
AMAT is an open source collection of Python subroutines for rapid conceptual design of aerocapture and atmospheric Entry, Descent, and Landing (EDL) missions in a Jupyter environment.

AMAT comes with a suite of tools to allow end-to-end conceptual design of aerocapture missions: launch vehicle performance calculator, database of interplanetary trajectories, atmosphere models, vehicle control techniques, and aeroheating models. AMAT supports analysis for all atmosphere-bearing Solar System destinations for both lift and drag modulation control techniques.

AMAT has been extensively used in various aerocapture mission studies at the [Advanced Astrodynamics Concepts](#) (AAC) research group at Purdue University in collaboration with the NASA Jet Propulsion Laboratory (JPL).

1.2 History

The lack of a rapid mission design tool for aerocapture mission concepts was identified by the NASA [Ice Giants Pre Decadal](#) (IGPD) Study led by JPL in 2016. This meant there was no quick way of performing architectural level assessments without resorting to resource intensive, subsystem-level design exercises such as the NASA [Aerocapture Systems Analysis](#) Team studies in 2004.



A team of researchers at Purdue University (Saikia et al.) led the aerocapture assessment studies in support of IGPD. Graduate researchers have since then further developed and extended the methods and tools for other atmosphere-bearing Solar System destinations. The focus was on developing an integrated systems engineering framework to allow mission designers to quickly evaluate the feasibility and performance of aerocapture mission concepts. Ye Lu and Athul P. Girija from the AAC research group conceptualized the **aerocapture feasibility charts**, now a commonly used graphical method for aerocapture mission design. An earlier version of the feasibility charts was presented by Saikia et al. in the IGPD study report.

Athul P. Girija formulated a systems framework for rapid conceptual design of aerocapture missions for his doctoral thesis. Much of the AMAT source code was originally written in support of his Ph.D. dissertation work. AMAT was first publicly released in November 2019, and has since then been maintained by the author at Purdue University. In the spirit of [open code for open science](#), AMAT is free and open-source to foster universal access to the knowledge, and allow reproducibility of results by other researchers. Suggestions for improvement and potential contributions are greatly welcome.

1.3 Related software

There are industrial software tools which offer mission analysis capabilities for aerocapture missions. These offer much higher fidelity, but are also substantially more complex to set up and run. Such fidelity is most often not required at the level conceptual studies. These tools are not in the public domain, and is generally available only to U.S. government affiliated labs, and U.S. persons at academic or industrial institutions.

- **POST2:** According to NASA Langley Research Center, “The Program to Optimize Simulated Trajectories II (POST2) is a generalized point mass, discrete parameter targeting and optimization program. POST2 provides the capability to target and optimize point mass trajectories for multiple powered or un-powered vehicles near an arbitrary rotating, oblate planet. POST2 has been used successfully to solve a wide variety of atmospheric ascent and entry problems, as well as exo-atmospheric orbital transfer problems.”
- **DSENDS:** According to NASA Jet Propulsion Laboratory, “DSENDS is a high-fidelity spacecraft simulator for Entry, Descent and Landing (EDL) on planetary and small-bodies. DSENDS (Dynamics Simulator for

Entry, Descent and Surface landing) is an EDL-specific extension of a JPL multi-mission simulation toolkit Darts/Dshell which is capable of modeling spacecraft dynamics, devices, and subsystems, and is in use by interplanetary and science-craft missions such as Cassini, Galileo, SIM, and Starlight. DSENDS is currently in use by the JPL Mars Science Laboratory project to provide a high-fidelity testbed for the test of precision landing and hazard avoidance functions for future Mars missions. “

1.4 Future ideas

Some things I would like to implement in the future:

- Pairing AMAT with [Blender](#) and [NASA 3D models](#) of planets and spacecraft to produce high resolution renders of aerocapture vehicle trajectories.
- Improved guidance schemes for lift and drag modulation aerocapture such as direct force control.
- Improved support for EDL mission concepts in the areas of precision landing, parachute dynamics, terminal descent and landing phases.

1.5 Note from the original author

I am [Athul P Girija](#), an aerospace engineer with a passion for robotic exploration of our Solar System. At the time of writing, I am a Ph.D. candidate in the Advanced Astrodynamics Concepts (AAC) research group led by Prof. Sarag Saikia and Prof. James Longuski at Purdue University. My interests lie in the general areas of planetary science, mission design and concept formulation, open-source software, and scientific programming.

AMAT is available for general public use under the GNU GPLv3 license.

[Google Scholar](#)

**CHAPTER
TWO**

INSTALLATION

Note: AMAT is designed to work with Python 3.0 or greater. You must have a Python 3 installation in your system. AMAT has been tested on Linux/MacOS and Windows machines.

There are three ways to install AMAT.

2.1 Option 1 : Install from pip (recommended)

Note: Python Package Index limits the amount of additional data that can be packaged in the distribution, hence all data cannot be included in the built version. You will need to clone the GitHub repository to get the required data files, examples, and start using AMAT.

For Linux/MacOS machines:

- \$ pip install AMAT
- \$ git clone https://github.com/athulpg007/AMAT.git

If you are unable to clone the repository, you can download the repository as a .zip file from GitHub and extract it. Once AMAT is installed, run an example Jupyter notebook to check everything works correctly.

- \$ cd AMAT/examples
- \$ jupyter-notebook

Note: Note that you will need jupyterlab and pandas (for some examples) to run the example notebooks. Use `pip install jupyterlab pandas` to install Jupyter and pandas if it is not already installed on your system.

This will display the full list of example Jupyter notebooks included with AMAT. Open and run the `example-01-hello-world` notebook to get started with AMAT.

For Windows machines:

- \$ pip install AMAT

(You must have Anaconda installed. Use the pip command from the Anaconda Prompt terminal). Open a Windows Powershell terminal and clone the GitHub repository. You must have Git installed.

- \$ git clone https://github.com/athulpg007/AMAT.git

Run an example Jupyter notebook. From the Anaconda Prompt terminal:

- \$ cd AMAT/examples
- \$ jupyter-notebook

Note: Note that you will need jupyterlab and pandas (for some examples) to run the example notebooks. Use pip install jupyterlab pandas to install Jupyter and pandas if it is not already installed on your system.

This will display the full list of example Jupyter notebooks included with AMAT. Open and run the example-01-hello-world notebook to get started with AMAT.

2.2 Option 2 : Install from source

This will clone the repository from GitHub and install AMAT from the source code.

For Linux/MacOS machines:

- \$ pip install numpy scipy matplotlib

Clone the GitHub repository and install AMAT.

- \$ git clone https://github.com/athulpg007/AMAT.git
- \$ cd AMAT
- \$ python setup.py install
- \$ cd examples
- \$ jupyter-notebook

Note: Note that you will need jupyterlab and pandas (for some examples) to run the example notebooks. Use pip install jupyterlab pandas to install Jupyter and pandas if it is not already installed on your system.

For Windows machines (from the Anaconda Prompt terminal):

- \$ pip install numpy scipy matplotlib

Open a Windows Powershell terminal, clone the GitHub repository and install AMAT.

- \$ git clone https://github.com/athulpg007/AMAT.git
- \$ cd AMAT
- \$ python setup.py install
- \$ cd examples
- \$ jupyter-notebook

Note: Note that you will need jupyterlab and pandas (for some examples) to run the example notebooks. Use pip install jupyterlab pandas to install Jupyter and pandas if it is not already installed on your system.

To uninstall AMAT:

1. If you installed AMAT using pip:

- \$ pip uninstall AMAT

2. If you installed AMAT from source, from the main AMAT directory:

- \$ python setup.py develop -u

This will remove the AMAT installation from Python. You may simply delete the root folder where AMAT was installed to completely remove the files.

2.3 Option 3 : Install in a virutalenv

If you plan to modify the source code or add features, the recommended option is to install it in a virtual environment.

1. Change directory to where you want the virtual environment to be created.
 - \$ cd home/path
2. Create a virutal environment and activate it.

On Linux/MacOS machines:

- \$ python3 -m venv env1
- \$ source env1/bin/activate

On Windows machines (from Anaconda Prompt):

- \$ conda create --name env1
- \$ conda activate env1
- \$ conda install pip

4. Follow the steps outlined in Option #2 (build from source) to clone the repository and install AMAT. If you make changes to the source code, remove the existing installation, update the setup file with a new version number, and re-install:

- \$ python setup.py develop -u
- \$ python setup.py install

If you want to create a new distribution package:

- \$ python3 setup.py sdist bdist_wheel

To re-make docs if you made changes to the source code (you must have Sphinx installed):

- \$ cd ~root/docs
- \$ sphinx-apidoc -f -o source/ ../
- \$ make html

If you added a new AMAT module, appropriate changes must be made to docs/source/AMAT.rst.

2.4 AMAT Usage

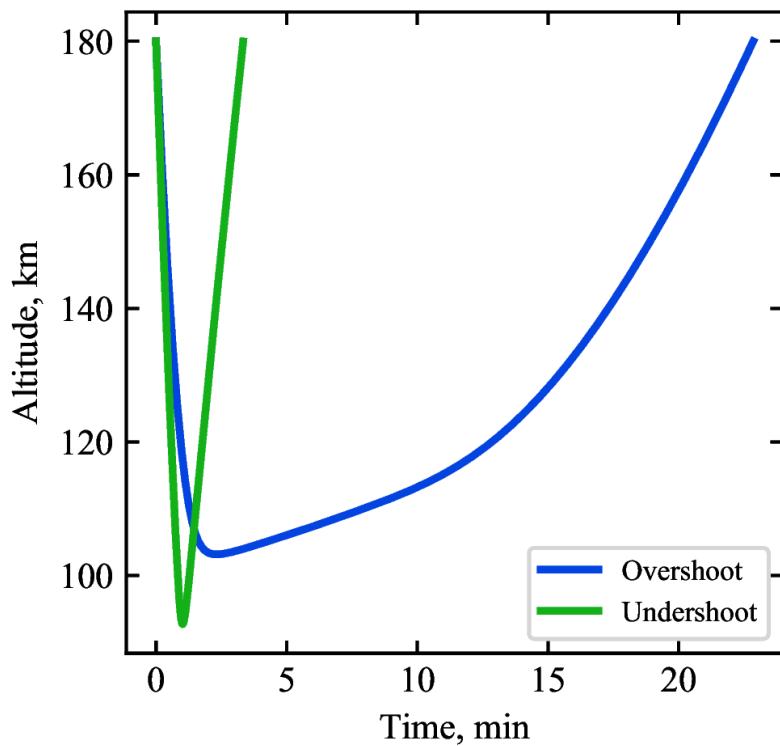
- from AMAT.planet import Planet
- from AMAT.vehicle import Vehicle
- from AMAT.launcher import Launcher

CAPABILITIES

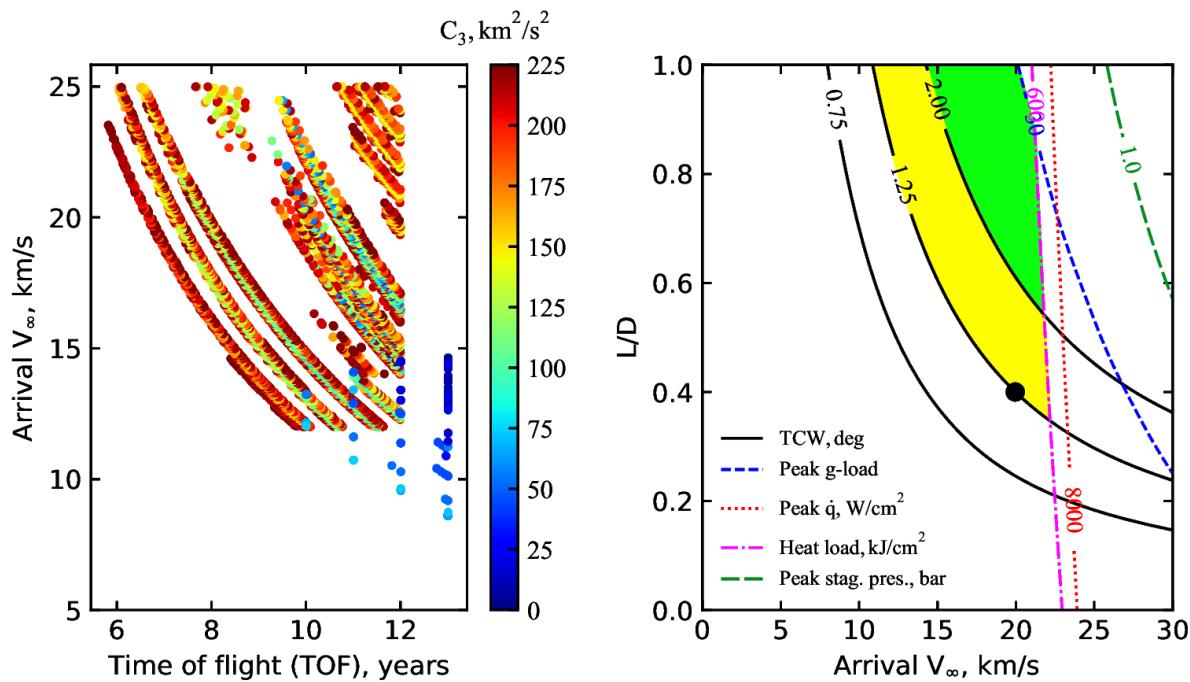
AMAT allows the user to perform low-fidelity broad sweep parametric studies; as well as high fidelity Monte Carlo simulations to quantify aerocapture performance. AMAT supports analysis for all atmosphere-bearing destinations in the Solar System: Venus, Earth, Mars, Jupiter, Saturn, Titan, Uranus, and Neptune.

3.1 Sample results

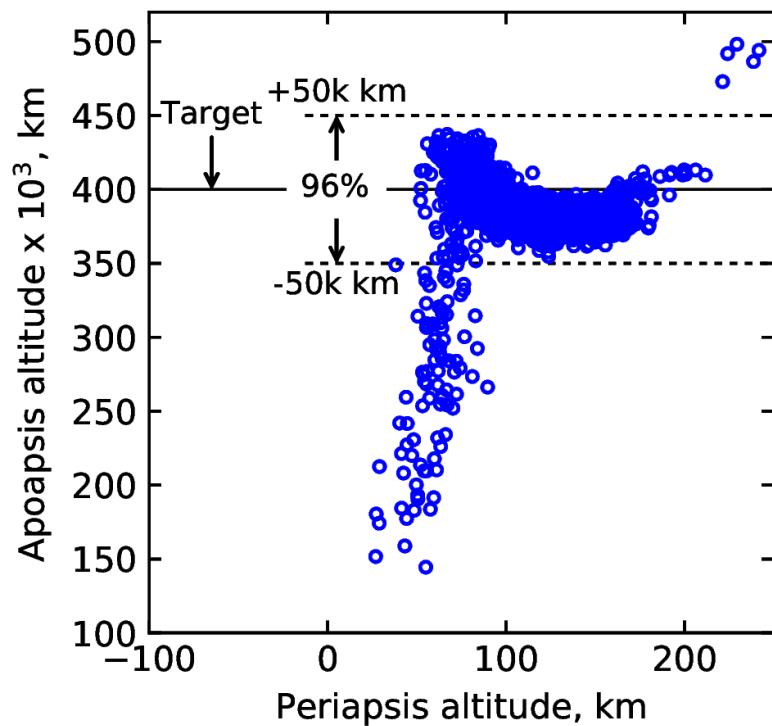
3.1.1 Venus Aerocapture Trajectory



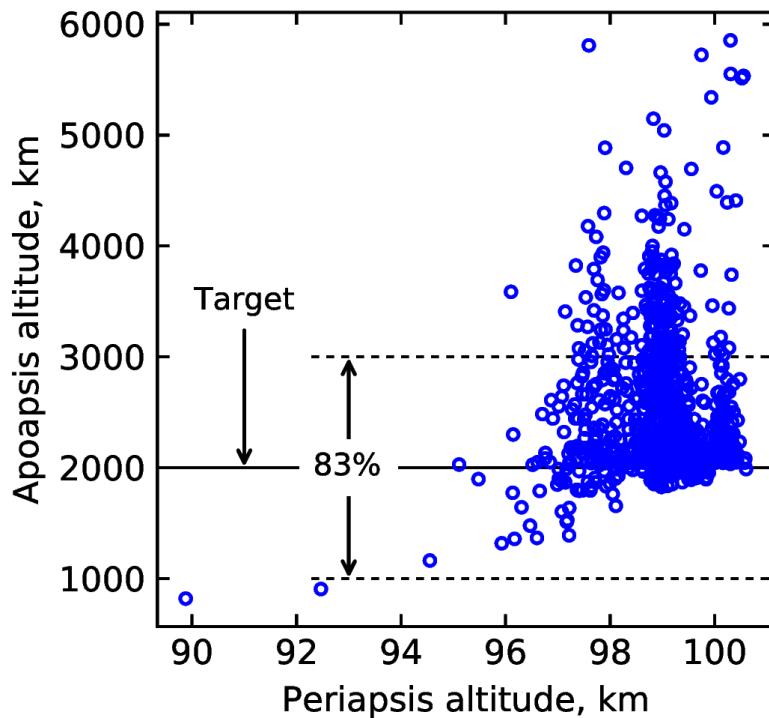
3.1.2 Neptune Aerocapture Feasibility Chart



3.1.3 Monte Carlo simulations - Neptune aerocapture



3.1.4 Monte Carlo simulations - SmallSat aerocapture at Venus



3.2 What kind of problems can AMAT solve?

AMAT can be used to quickly assess the feasibility of an aerocapture mission concept using aerocapture feasibility charts, and perform trade studies involving vehicle type, control authority, thermal protection system materials, and useful delivered mass to orbit. AMAT can also be used to set up and run high-fidelity Monte Carlo simulations of aerocapture trajectories considering delivery errors, atmospheric uncertainties, and aerodynamic uncertainties to evaluate system performance under uncertainty.

EXAMPLE JUPYTER NOTEBOOKS

4.1 Example - 01 - Hello World!

In this ‘hello world’ program, you will learn to use AMAT to create a planet object.

First let us import the Planet class from AMAT

```
[1]: from AMAT.planet import Planet
```

Now let us create a planet object which represents Venus.

```
[2]: planet = Planet("VENUS")
```

Let us look at the an attribute of the created object. For example let us, print the radius of the planet. A full list of attributes and functions can be obtained using help(planet).

```
[3]: print(planet.RP)  
6051800.0
```

Congratulations! You have now created a planet object using AMAT. In the next example, we will add an atmosphere model to this planet object.

4.2 Example - 02 - Atmosphere

In this example, you will learn to use add an atmosphere model to planets.

Let us re-use the code from example-01 to create a planet object for Venus.

```
[1]: from AMAT.planet import Planet  
planet = Planet("VENUS")
```

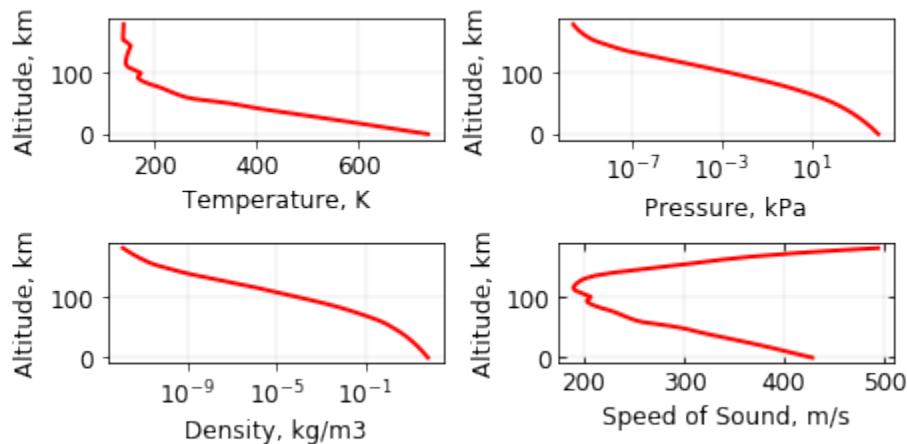
We are now ready to add an atmosphere model to the planet object.

AMAT stores atmospheric data in the form of look up tables located in ~root/atmdata. Typically, data is stored in the following format.

#Z(m)	Temp(K)	Pres (Nm2)	rho(kgm3)	a (m/s)
0	735.30	9.209E+06	6.479E+01	428.03
1000	727.70	8.645E+06	6.156E+01	425.46
2000	720.20	8.109E+06	5.845E+01	422.88
3000	712.40	7.601E+06	5.547E+01	420.27
4000	704.60	7.120E+06	5.262E+01	417.63
5000	696.80	6.666E+06	4.987E+01	415.09

```
[2]: # the atmosphere model provided in atmdatas/Venus/venus-gram-avg.dat.
# The columns for height, Temp, pressure, density are 0, 1, 2, 3
planet.loadAtmosphereModel('..../atmdatas/Venus/venus-gram-avg.dat', 0 , 1 ,2, 3)
```

```
[3]: # Let us now use the checkAtmProfiles function to inspect the atmospheric profiles.
planet.checkAtmProfiles()
```



```
[4]: # Compute the scale height at the Venusian surface for illustration.
# planet.density_int is the interpolation function created by planet object
# when atmosphere model is loaded.
planet.scaleHeight(0, planet.density_int)
```

```
[4]: 16127.792366356383
```

Congratulations! Your planet now has an atmosphere model. In the next example, we will compute aerocapture trajectories for a vehicle flying in the Venusian atmosphere.

4.3 Example - 03 - Venus Aerocapture: Part 1

In this example, you will learn to create a vehicle object in AMAT and simulate its aerocapture trajectory in the Venusian atmosphere.

For reference, we will re-create a few results from the paper “Craig and Lyne, Parametric Study of Aerocapture for Missions to Venus, Journal of Spacecraft and Rockets, Vol. 42, No. 6., 2005. DOI:10.2514/1.2589

Let us re-use the code from example-01, 02 to create a planet object for Venus and add an atmosphere model. In addition to Planet, import the Vehicle class from AMAT for this example.

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle

# Create a planet object
planet=Planet("VENUS")

# Load an nominal atmospheric profile with height, temp, pressure, density data
planet.loadAtmosphereModel('../atmdata/Venus/venus-gram-avg.dat', 0 , 1 ,2, 3)
```

```
[2]: # Create a vehicle object flying in the target planet atmosphere.
# with params m = 300 kg, beta = 78.0, L/D = 0.35, A = 3.1416, AoA = 0, RN = 1.54

# These values are taken from the reference article mentioned above.
vehicle=Vehicle('Apollo', 300.0, 78.0, 0.35, 3.1416, 0.0, 1.54, planet)
```

Set initial conditions for the vehicle at atmospheric entry interface.

```
[3]: # h0 = 180 km, LON = 0 deg, LAT = 0 deg
# v0 = 12 km/s, HDG = 0 deg, FPA = 0 deg
# DOWNRANGE0 = 0 deg, HEATLOAD0 = 0.

# See help(vehicle) for more details.
vehicle.setInitialState(180.0,0.0,0.0,12.0,0.0,-4.5,0.0,0.0)

# Set solver tolerance = 1E-6 (recommended value)
# Setting this too low can result in long execution times.
vehicle.setSolverParams(1E-6)
```

```
[4]: # Compute the overshoot and undershoot limit EFPA

# Set max. propagation time = 2400.0 secs.
# Set max. time step = 0.1 sec.
# Set low value for guess = -80.0 deg
# Set high value for guess = -4.0 deg
# Set EFPA tolerance = 1E-10 (recommended)
# Set target apoapsis = 407 km

# This calculation migt take a couple of minutes. Hang on!
overShootLimit, exitflag_os  = vehicle.findOverShootLimit (2400.0,0.1,-80.0,-4.0,1E-
→10,407.0)
underShootLimit,exitflag_us  = vehicle.findUnderShootLimit(2400.0,0.1,-80.0,-4.0,1E-
→10,407.0)
```

```
[5]: # exitflag_os and exitflag_us will be set to 1 if a solution was found. Otherwise, it_
→will be 0.
print(exitflag_os)
print(exitflag_us)

1.0
1.0
```

```
[6]: # print the overshoot and undershoot limits we just computed.
print("Overshoot limit : "+str('{:.4f}'.format(overShootLimit))+ " deg")
print("Undershoot limit : "+str('{:.4f}'.format(underShootLimit))+ " deg")
```

Overshoot limit : -7.0519 deg
Undershoot limit : -9.4396 deg

These are the limiting flight path angles for our vehicle at Venus. Let us now calculate these trajectories, and their associated deceleration and heating profiles.

```
[7]: import matplotlib.pyplot as plt
from matplotlib import rcParams

# Reset initial conditions and propagate overshoot trajectory
vehicle.setInitialState(180.0,0.0,0.0,12.0,0.0,overShootLimit,0.0,0.0)
vehicle.propagateEntry (2400.0,0.1,180.0)

# Extract and save variables to plot
t_min_os      = vehicle.t_minc
h_km_os       = vehicle.h_kmc
acc_net_g_os  = vehicle.acc_net_g
q_stag_con_os = vehicle.q_stag_con
q_stag_rad_os = vehicle.q_stag_rad

# Reset initial conditions and propagate undershoot trajectory
vehicle.setInitialState(180.0,0.0,0.0,12.0,0.0,underShootLimit,0.0,0.0)
vehicle.propagateEntry (2400.0,0.1,0.0)

# Extract and save variable to plot
t_min_us      = vehicle.t_minc
h_km_us       = vehicle.h_kmc
acc_net_g_us  = vehicle.acc_net_g
q_stag_con_us = vehicle.q_stag_con
q_stag_rad_us = vehicle.q_stag_rad

'''

Create fig #1 - altitude history of aerocapture maneuver
''''

fig = plt.figure()
fig.set_size_inches([3.25,3.25])
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.plot(t_min_os , h_km_os, linestyle='solid' , color='xkcd:blue', linewidth=2.0, ↴label='Overshoot')
plt.plot(t_min_us , h_km_us, linestyle='solid' , color='xkcd:green', linewidth=2.0, ↴label='Undershoot')

plt.xlabel('Time, min', fontsize=10)
plt.ylabel("Altitude, km", fontsize=10)

ax = plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
plt.tick_params(direction='in')
plt.tick_params(axis='x', labelsize=10)
plt.tick_params(axis='y', labelsize=10)

plt.legend(loc='lower right', fontsize=8)

plt.savefig('../plots/craig-lyne-altitude.png',bbox_inches='tight')
plt.savefig('../plots/craig-lyne-altitude.pdf', dpi=300,bbox_inches='tight')
```

(continues on next page)

(continued from previous page)

```

plt.savefig('../plots/craig-lyne-altitude.eps', dpi=300, bbox_inches='tight')

plt.show()

fig = plt.figure()
fig.set_size_inches([3.25,3.25])
plt.rc('font',family='Times New Roman')
plt.plot(t_min_os , acc_net_g_os, linestyle='solid' , color='xkcd:blue', linewidth=1.0,
         label='Overshoot')
plt.plot(t_min_us , acc_net_g_us, linestyle='solid' , color='xkcd:green', linewidth=1.0,
         label='Undershoot')

plt.xlabel('Time, min', fontsize=10)
plt.ylabel("Deceleration, Earth g", fontsize=10)

ax = plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
plt.tick_params(direction='in')
plt.tick_params(axis='x', labelsize=10)
plt.tick_params(axis='y', labelsize=10)

plt.legend(loc='upper right', fontsize=8)

plt.savefig('../plots/craig-lyne-deceleration.png',bbox_inches='tight')
plt.savefig('../plots/craig-lyne-deceleration.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/craig-lyne-deceleration.eps', dpi=300,bbox_inches='tight')

plt.show()

fig = plt.figure()
fig.set_size_inches([3.25,3.25])
plt.rc('font',family='Times New Roman')
plt.plot(t_min_os , q_stag_con_os, linestyle='solid' , color='xkcd:blue', linewidth=1.0,
         label='Overshoot convective')
plt.plot(t_min_os , q_stag_rad_os, linestyle='solid' , color='xkcd:red', linewidth=1.0,
         label='Overshoot radiative')
plt.plot(t_min_us , q_stag_con_us, linestyle='solid' , color='xkcd:magenta',
         linewidth=1.0, label='Undershoot convective')
plt.plot(t_min_us , q_stag_rad_us, linestyle='solid' , color='xkcd:green', linewidth=1.0,
         label='Undershoot radiative')

plt.xlabel('Time, min', fontsize=10)
plt.ylabel("Stagnation-point heat rate, "+r'$W/cm^2$', fontsize=10)

ax = plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
plt.tick_params(direction='in')
plt.tick_params(axis='x', labelsize=10)
plt.tick_params(axis='y', labelsize=10)

plt.legend(loc='upper right', fontsize=8)

```

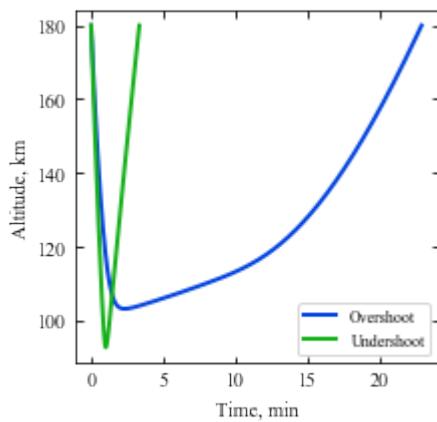
(continues on next page)

(continued from previous page)

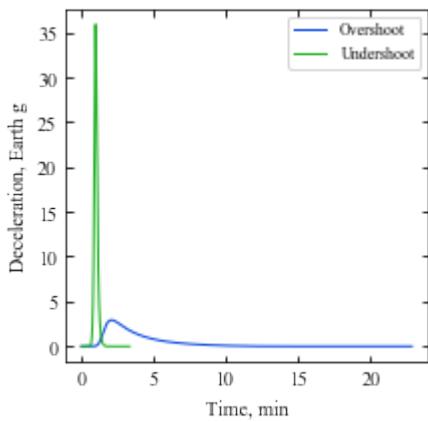
```
plt.savefig('../plots/craig-lyne-heating.png',bbox_inches='tight')
plt.savefig('../plots/craig-lyne-heating.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/craig-lyne-heating.eps', dpi=300,bbox_inches='tight')

plt.show()
```

The PostScript backend does not support transparency; partially transparent artists
 ↵will be rendered opaque.
 The PostScript backend does not support transparency; partially transparent artists
 ↵will be rendered opaque.
 The PostScript backend does not support transparency; partially transparent artists
 ↵will be rendered opaque.
 The PostScript backend does not support transparency; partially transparent artists
 ↵will be rendered opaque.



The PostScript backend does not support transparency; partially transparent artists
 ↵will be rendered opaque.
 The PostScript backend does not support transparency; partially transparent artists
 ↵will be rendered opaque.
 The PostScript backend does not support transparency; partially transparent artists
 ↵will be rendered opaque.
 The PostScript backend does not support transparency; partially transparent artists
 ↵will be rendered opaque.



The PostScript backend does not support transparency; partially transparent artists
 ↵will be rendered opaque.

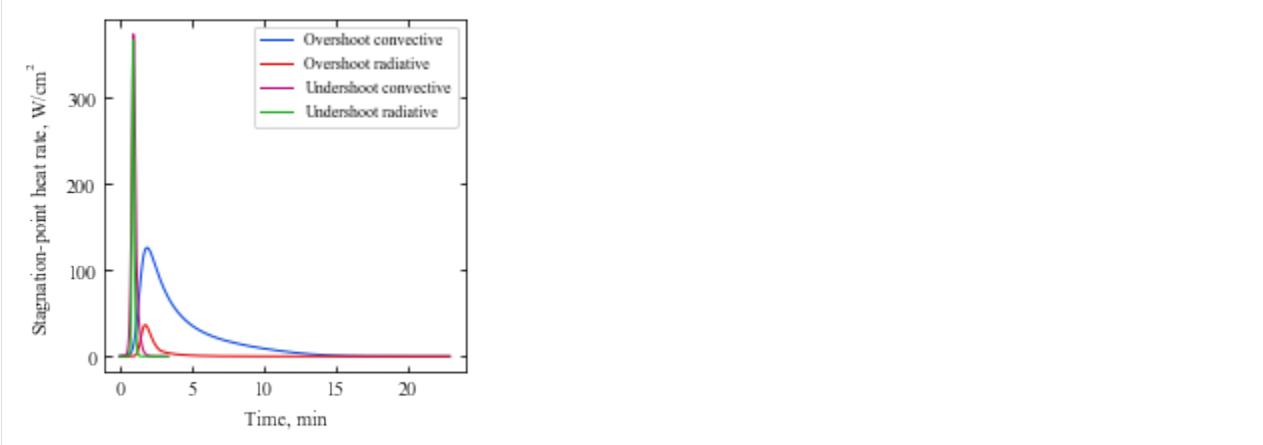
(continues on next page)

(continued from previous page)

The PostScript backend does not support transparency; partially transparent artists [will be rendered opaque](#).

The PostScript backend does not support transparency; partially transparent artists [will be rendered opaque](#).

The PostScript backend does not support transparency; partially transparent artists [will be rendered opaque](#).



The plots are now saved in plots/craig-lyne-* and should match with the results from the paper.

Congratulations! You have simulated an aerocapture vehicle flying in the Venusian atmosphere. In the next tutorial, we will compute the corridor widths for drag modulation aerocapture at Venus.

4.4 Example - 04 - Venus Aerocapture: Part 2

In this example, you will learn to create a drag modulation vehicle object and compute the theoretical corridor width available for drag modulation aerocapture at Venus.

For reference, we will re-create a figure from the paper “Putnam and Braun, Drag-Modulation Flight-Control System Options for Planetary Aerocapture, Journal of Spacecraft and Rockets, Vol. 51, No. 1, 2014. DOI: 10.2514/1.A32589

Import AMAT and required libraries

```
[2]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle

import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rcParams
import os
```

```
[3]: # Create a planet object
planet=Planet("VENUS")

# Load an nominal atmospheric profile with height, temp, pressure, density data
planet.loadAtmosphereModel('../atmdata/Venus/venus-gram-avg.dat', 0 , 1 ,2, 3)

# The reference paper uses 150 km as entry interface altitude (vs 180 km as pre-defined in AMAT).
# Hence reset planet.h_skip to 150 km
```

(continues on next page)

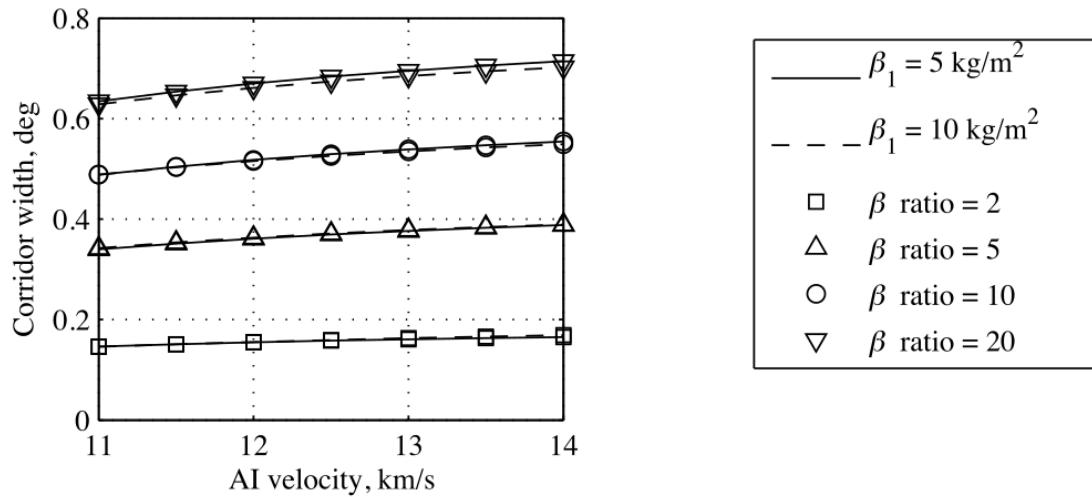
(continued from previous page)

```
planet.h_skip = 150000.0
# Create a vehicle object flying in the target planet atmosphere
vehicle=Vehicle('DMVehicle', 1500.0, 50.0, 0.0, 3.1416, 0.0, 0.10, planet)

# Set vehicle conditions at entry interface
# Note that the EFPA initial value is arbitrary. It will be computed and reset later.
vehicle.setInitialState(150.0,0.0,0.0,11.0,0.0,-4.5,0.0,0.0)
vehicle.setSolverParams(1E-6)
```

[4]: `from IPython.display import Image
Image(filename='../plots/putnam-2014-reference.png')`

[4]:



This is the figure that we want to reproduce using AMAT. Let us start setting up the simulation parameters.

[7]: `# Define the AI velocity and ballistic ratio range
VAI_array = np.linspace(11,14,7)
BR_array = np.array([2.0,5.0,10.0,20.0])

NOTE: You will get a FileExistsError if you run this as such. Re-name the directory ↴ to something else.

Create a directory in ../data/ to store the data
os.makedirs('../data/putnamBraun2013')

Store the speed range and ballistic coefficient range
np.savetxt('../data/putnamBraun2013/VAI_array.txt', VAI_array)
np.savetxt('../data/putnamBraun2013/BR_array.txt', BR_array)

Create two empty matrices to store the TCW values
TCW_array1 = np.zeros((len(VAI_array), len(BR_array)))
TCW_array2 = np.zeros((len(VAI_array), len(BR_array)))

Define the two values for beta_1 used in the paper.
beta11 = 5.0
beta12 = 10.0`

```
[8]: # Compute the corridor width values to create the figure.
# NOTE: This calculation will take several minutes.
print('Running beta1=5.0')

for i in range(0,len(VAI_array)):
    for j in range(0,len(BR_array)):
        vehicle.setInitialState(150.0,0.0,0.0,VAI_array[i],0.0,-4.5,0.0,0.0)
        vehicle.setDragModulationVehicleParams(beta11, BR_array[j])
        TCW_array1[i,j] = vehicle.computeTCWD(2400.0, 0.1,-80.0,-4.0,1E-10,400.0)
        print('VAI: '+str(VAI_array[i])+' km/s, BETA RATIO: '+str(BR_array[j])+' TCW:
        ↪'+str(TCW_array1[i,j])+' deg.')

np.savetxt('../data/putnamBraun2013/TCW_array1.txt', TCW_array1)

print('Running beta1=10.0')
for i in range(0,len(VAI_array)):
    for j in range(0,len(BR_array)):
        vehicle.setInitialState(150.0,0.0,0.0,VAI_array[i],0.0,-4.5,0.0,0.0)
        vehicle.setDragModulationVehicleParams(beta12, BR_array[j])
        TCW_array2[i,j] = vehicle.computeTCWD(2400.0, 0.1,-80.0,-4.0,1E-10,400.0)
        print('VAI: '+str(VAI_array[i])+' km/s, BETA RATIO: '+str(BR_array[j])+' TCW:
        ↪'+str(TCW_array2[i,j])+' deg.')

np.savetxt('../data/putnamBraun2013/TCW_array2.txt', TCW_array2)
print('Done!')


Running beta1=5.0
VAI: 11.0 km/s, BETA RATIO: 20.0 TCW: 0.6330942395761667 deg.
VAI: 14.0 km/s, BETA RATIO: 20.0 TCW: 0.7133574859435612 deg.
Running beta1=10.0
VAI: 11.0 km/s, BETA RATIO: 20.0 TCW: 0.6273882169480203 deg.
VAI: 14.0 km/s, BETA RATIO: 20.0 TCW: 0.7018732332253421 deg.
Done!
```

Load the data and make the plot.

```
[9]: VAI_array = np.loadtxt('../data/putnamBraun2013/VAI_array.txt')
BR_array = np.loadtxt('../data/putnamBraun2013/BR_array.txt')
TCW_array1 = np.loadtxt('../data/putnamBraun2013/TCW_array1.txt')
TCW_array2 = np.loadtxt('../data/putnamBraun2013/TCW_array2.txt')

fig = plt.figure()
fig.set_size_inches([3.25,3.25])
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.plot(VAI_array, TCW_array1[:,0], linestyle='-', linewidth=0.75, marker='s',ms=6,
         ↪markerfacecolor="None", markeredgewidth=0.75, color='black
         ↪', clip_on=False)
plt.plot(VAI_array, TCW_array1[:,1], linestyle='-', linewidth=0.75, marker='^',ms=6,
         ↪markerfacecolor="None", markeredgewidth=0.75, color='black
         ↪', clip_on=False)
plt.plot(VAI_array, TCW_array1[:,2], linestyle='-', linewidth=0.75, marker='o',ms=6,
         ↪markerfacecolor="None", markeredgewidth=0.75, color='black
         ↪', clip_on=False)
```

(continues on next page)

(continued from previous page)

```

plt.plot(VAI_array, TCW_array1[:,3], linestyle='--', linewidth=0.75, marker='v', ms=6,_
         markerfacecolor="None", markeredgecolor='black', markeredgewidth=0.75, color='black'
         , clip_on=False)

plt.plot(VAI_array, TCW_array2[:,0], linestyle='--', linewidth=0.75, marker='s', ms=6,_
         markerfacecolor="None", markeredgecolor='black', markeredgewidth=0.75, color='black'
         , clip_on=False)
plt.plot(VAI_array, TCW_array2[:,1], linestyle='--', linewidth=0.75, marker='^', ms=6,_
         markerfacecolor="None", markeredgecolor='black', markeredgewidth=0.75, color='black'
         , clip_on=False)
plt.plot(VAI_array, TCW_array2[:,2], linestyle='--', linewidth=0.75, marker='o', ms=6,_
         markerfacecolor="None", markeredgecolor='black', markeredgewidth=0.75, color='black'
         , clip_on=False)
plt.plot(VAI_array, TCW_array2[:,3], linestyle='--', linewidth=0.75, marker='v', ms=6,_
         markerfacecolor="None", markeredgecolor='black', markeredgewidth=0.75, color='black'
         , clip_on=False)

plt.xlabel('Entry velocity, km/s', fontsize=10)
plt.ylabel('Corridor width, deg', fontsize=10)
plt.yticks(np.arange(0, 0.9, step=0.2))
plt.xticks(np.arange(11.0, 14.5, step=1.0))

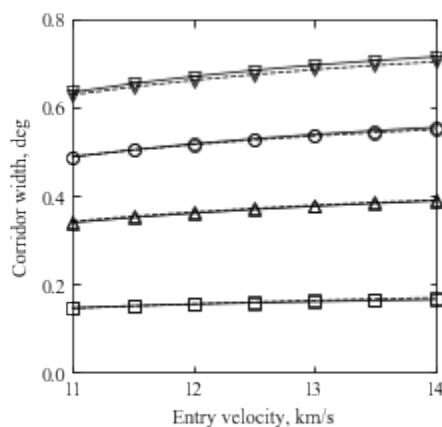
plt.xlim([11.0,14.0])
plt.ylim([0.0, 0.8])

ax = plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
plt.tick_params(direction='in')
plt.tick_params(axis='x', labelsize=10)
plt.tick_params(axis='y', labelsize=10)

plt.savefig('../plots/putnam-braun-2013.png', bbox_inches='tight')
plt.savefig('../plots/putnam-braun-2013.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/putnam-braun-2013.eps', dpi=300, bbox_inches='tight')

plt.show()

```



The plots are now saved in plots/putnam-braun-* and should match with the results from the paper.

Congratulations! You have now computed the corridor widths for a drag modulation aerocapture vehicle at Venus. In the next example, we will create an aerocapture feasibility chart for Titan.

[]:

4.5 Example - 05 - Titan Aerocapture: Part 1

In this example, you will learn to create an aerocapture feasibility chart for Titan.

For reference, we will re-create a figure from the paper “Lu and Saikia, Feasibility Assessment of Aerocapture for Future Titan Orbiter Missions, Journal of Spacecraft and Rockets, Vol. 55, No. 5, 2018”. DOI: 10.2514/1.A34121

```
[3]: from IPython.display import Image
Image(filename='../plots/lu-saikia-reference-a.png', width=400)
```

[3]:

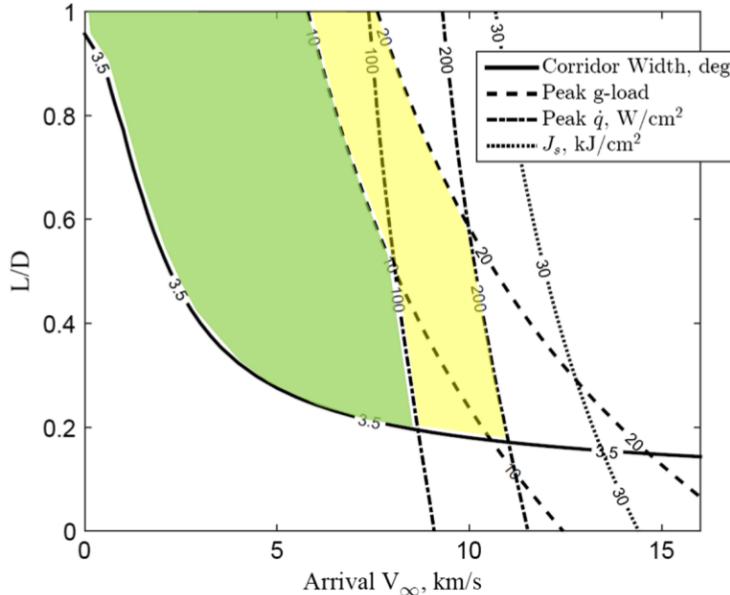


Fig. 16 Aerocapture applicable regions for β of 90 kg/m².

We will use AMAT to recreate this figure.

```
[4]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle

import numpy as np
from scipy import interpolate
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.patches import Polygon
import os
```

```
[6]: # Create a planet object for Titan
planet=Planet("TITAN")
```

(continues on next page)

(continued from previous page)

```
# Load an nominal atmospheric profile with height, temp, pressure, density data
planet.loadAtmosphereModel('../atmdata/Titan/titan-gram-avg.dat', 0 , 1 , 2, 3)

# Define the range for arrival Vinf and vehicle L/D
vinf_kms_array = np.linspace( 0.0, 16.0, 17)
LD_array       = np.linspace( 0.0, 1.0 , 11)

#vinf_kms_array = np.linspace( 0.0, 16.0, 2)
#LD_array       = np.linspace( 0.0, 1.0 , 2)
```

```
[7]: # Create a directory to store the data.
# NOTE: You will get an error if the file already exists,
# Rename the folder to something else.

os.makedirs('../data/luSaikia2018a')

# Use a runID to prefix the files for easy access for post-processing.
runID = 'BC90RAP1700'

num_total      = len(vinf_kms_array)*len(LD_array)
count = 1

# Compute the inertial entry speed from the hyperbolic excess speed.
v0_kms_array = np.zeros(len(vinf_kms_array))
v0_kms_array[:] = np.sqrt(1.0*(vinf_kms_array[:]*1E3)**2.0 + \
                         2*np.ones(len(vinf_kms_array))*\
                         planet.GM/(planet.RP+1000.0*1.0E3))/1.0E3

# Initialize matrices to store data.
overShootLimit_array = np.zeros((len(v0_kms_array),len(LD_array)))
underShootLimit_array = np.zeros((len(v0_kms_array),len(LD_array)))
exitflag_os_array   = np.zeros((len(v0_kms_array),len(LD_array)))
exitflag_us_array   = np.zeros((len(v0_kms_array),len(LD_array)))
TCW_array           = np.zeros((len(v0_kms_array),len(LD_array)))
```

```
[8]: # Compute the corridor width over the defined Vinf and L/D matrix.
# Note this will take maybe about an hour.
# If you simply want to create the plot, you can load the existing data
# in the ../data/luSaikia2018a folder as done below this cell.

for i in range(0,len(v0_kms_array)):
    for j in range(0,len(LD_array)):
        vehicle=Vehicle('Kraken', 1000.0, 90.0, LD_array[j], 3.1416, 0.0, 1.00,_
        ↪planet)
        vehicle.setInitialState(1000.0,0.0,0.0,v0_kms_array[i],0.0,-4.5,0.0,0.0)
        vehicle.setSolverParams(1E-5)
        overShootLimit_array[i,j], exitflag_os_array[i,j] = \
        vehicle.findOverShootLimit (6000.0, 1.0 , -88.0, -2.0, 1E-10, 1700.0)
        underShootLimit_array[i,j], exitflag_us_array[i,j] = \
        vehicle.findUnderShootLimit(6000.0, 1.0 , -88.0, -2.0, 1E-10, 1700.0)

        TCW_array[i,j] = overShootLimit_array[i,j] - underShootLimit_array[i,j]

        print("Run #"+str(count)+" of "+ str(num_total)+\
```

(continues on next page)

(continued from previous page)

```

": Arrival V_infty: "+str(vinf_kms_array[i]) + \
" km/s", L/D:"+str(LD_array[j]) + \
" OSL: "+str(overShootLimit_array[i,j]) + \
" USL: "+str(underShootLimit_array[i,j]) + \
", TCW: "+str(TCW_array[i,j]) + \
" EFOS: "+str(exitflag_os_array[i,j]) + \
" EFUS: "+str(exitflag_us_array[i,j]))


count = count +1

np.savetxt('..../data/luSaikia2018a/' + runID + 'vinf_kms_array.txt', vinf_kms_array)
np.savetxt('..../data/luSaikia2018a/' + runID + 'v0_kms_array.txt', v0_kms_array)
np.savetxt('..../data/luSaikia2018a/' + runID + 'LD_array.txt', LD_array)
np.savetxt('..../data/luSaikia2018a/' + runID + 'overShootLimit_array.txt', overShootLimit_ \
array)
np.savetxt('..../data/luSaikia2018a/' + runID + 'exitflag_os_array.txt', exitflag_os_array)
np.savetxt('..../data/luSaikia2018a/' + runID + 'undershootLimit_array.txt', underShootLimit_ \
array)
np.savetxt('..../data/luSaikia2018a/' + runID + 'exitflag_us_array.txt', exitflag_us_array)
np.savetxt('..../data/luSaikia2018a/' + runID + 'TCW_array.txt', TCW_array)

Run #1 of 4: Arrival V_infty: 0.0 km/s, L/D:0.0 OSL: -24.635471317420524 USL: -24. \
635471317420524, TCW: 0.0 EFOS: 1.0 EFUS: 1.0
Run #2 of 4: Arrival V_infty: 0.0 km/s, L/D:1.0 OSL: -23.422944245403414 USL: -27. \
0515137471466, TCW: 3.6285695017431863 EFOS: 1.0 EFUS: 1.0
Run #3 of 4: Arrival V_infty: 16.0 km/s, L/D:0.0 OSL: -37.36673820708165 USL: -37. \
36673820708165, TCW: 0.0 EFOS: 1.0 EFUS: 1.0
Run #4 of 4: Arrival V_infty: 16.0 km/s, L/D:1.0 OSL: -33.47214682791673 USL: -73. \
89901192915568, TCW: 40.426865101238945 EFOS: 1.0 EFUS: 1.0

```

Compute the deceleration and heating loads.

```
[9]: acc_net_g_max_array      = np.zeros((len(v0_kms_array), len(LD_array)))
stag_pres_atm_max_array    = np.zeros((len(v0_kms_array), len(LD_array)))
q_stag_total_max_array    = np.zeros((len(v0_kms_array), len(LD_array)))
heatload_max_array         = np.zeros((len(v0_kms_array), len(LD_array)))

for i in range(0, len(v0_kms_array)):
    for j in range(0, len(LD_array)):
        vehicle=Vehicle('Kraken', 1000.0, 90.0, LD_array[j], 3.1416, 0.0, 1.00, \
planet)
        vehicle.setInitialState(1000.0, 0.0, 0.0, v0_kms_array[i], 0.0, overShootLimit_ \
array[i,j], 0.0, 0.0)
        vehicle.setSolverParams(1E-5)

        vehicle.propogateEntry(6000.0, 1.0, 180.0)

        # Extract and save variables to plot
        t_min_os           = vehicle.t_minc
        h_km_os            = vehicle.h_kmc
        acc_net_g_os       = vehicle.acc_net_g
        q_stag_con_os     = vehicle.q_stag_con
        q_stag_rad_os     = vehicle.q_stag_rad
        rc_os              = vehicle.rc
        vc_os              = vehicle.vc
        stag_pres_atm_os  = vehicle.computeStagPres(rc_os, vc_os) / (1.01325E5)
        heatload_os         = vehicle.heatload
```

(continues on next page)

(continued from previous page)

```

        vehicle=Vehicle('Kraken', 1000.0, 90.0, LD_array[j], 3.1416, 0.0, 1.00,
    ↪planet)
        vehicle.setInitialState(1000.0,0.0,0.0,v0_kms_array[i],0.0,underShootLimit_
    ↪array[i,j],0.0,0.0)
        vehicle.setSolverParams(1E-5)

        vehicle.propagateEntry(6000.0, 1.0, 0.0)

        # Extract and save variable to plot
        t_min_us           = vehicle.t_minc
        h_km_us            = vehicle.h_kmc
        acc_net_g_us       = vehicle.acc_net_g
        q_stag_con_us     = vehicle.q_stag_con
        q_stag_rad_us     = vehicle.q_stag_rad
        rc_us              = vehicle.rc
        vc_us              = vehicle.vc
        stag_pres_atm_us  = vehicle.computeStagPres(rc_us,vc_us) / (1.01325E5)
        heatload_us         = vehicle.heatload

        q_stag_total_os   = q_stag_con_os + q_stag_rad_os
        q_stag_total_us   = q_stag_con_us + q_stag_rad_us

        acc_net_g_max_array[i,j]      = max(max(acc_net_g_os),max(acc_net_g_us))
        stag_pres_atm_max_array[i,j]  = max(max(stag_pres_atm_os),max(stag_pres_atm_
    ↪os))
        q_stag_total_max_array[i,j]   = max(max(q_stag_total_os),max(q_stag_total_us))
        heatload_max_array[i,j]       = max(max(heatload_os),max(heatload_us))

        print("V_infty: "+str(vinf_kms_array[i])+" km/s", L/D: "+str(LD_array[j])+"_
    ↪G_MAX: "+str(acc_net_g_max_array[i,j])+" QDOT_MAX: "+str(q_stag_total_max_array[i,_
    ↪j])+" J_MAX: "+str(heatload_max_array[i,j])+" STAG. PRES: "+str(stag_pres_atm_max_
    ↪array[i,j])))

np.savetxt('~/data/luSaikia2018a/' + runID + 'acc_net_g_max_array.txt', acc_net_g_max_
    ↪array)
np.savetxt('~/data/luSaikia2018a/' + runID + 'stag_pres_atm_max_array.txt', stag_pres_atm_
    ↪max_array)
np.savetxt('~/data/luSaikia2018a/' + runID + 'q_stag_total_max_array.txt', q_stag_total_
    ↪max_array)
np.savetxt('~/data/luSaikia2018a/' + runID + 'heatload_max_array.txt', heatload_max_array)

```

```

V_infty: 0.0 km/s, L/D: 0.0 G_MAX: 0.0810485245607735 QDOT_MAX: 1.0607636054752763 J_
    ↪MAX: 1020.003796275038 STAG. PRES: 0.000720422444467468
V_infty: 0.0 km/s, L/D: 1.0 G_MAX: 0.20202892551813917 QDOT_MAX: 1.3893301148573975 J_
    ↪MAX: 1252.170615904772 STAG. PRES: 0.0004253391968484517
V_infty: 16.0 km/s, L/D: 0.0 G_MAX: 17.704051243426182 QDOT_MAX: 536.4530046363565 J_
    ↪MAX: 35907.65472341787 STAG. PRES: 0.15433527803169042
V_infty: 16.0 km/s, L/D: 1.0 G_MAX: 125.7389398961218 QDOT_MAX: 1065.8317771270392 J_
    ↪MAX: 63948.52565398295 STAG. PRES: 0.044601211536270044

```

We are now ready to create the plot!

```
[17]: x = np.loadtxt('~/data/luSaikia2018a/' + runID + 'vinf_kms_array.txt')
y = np.loadtxt('~/data/luSaikia2018a/' + runID + 'LD_array.txt')
```

(continues on next page)

(continued from previous page)

```

Z1 = np.loadtxt('../data/luSaikia2018a/' + runID + 'TCW_array.txt')
G1 = np.loadtxt('../data/luSaikia2018a/' + runID + 'acc_net_g_max_array.txt')
Q1 = np.loadtxt('../data/luSaikia2018a/' + runID + 'q_stag_total_max_array.txt')
H1 = np.loadtxt('../data/luSaikia2018a/' + runID + 'heatload_max_array.txt')

f1 = interpolate.interp2d(x, y, np.transpose(Z1), kind='cubic')
g1 = interpolate.interp2d(x, y, np.transpose(G1), kind='cubic')
q1 = interpolate.interp2d(x, y, np.transpose(Q1), kind='cubic')
h1 = interpolate.interp2d(x, y, np.transpose(H1), kind='cubic')

x_new = np.linspace(0.0, 16, 170)
y_new = np.linspace(0.0, 1.0, 110)
z_new = np.zeros((len(x_new), len(y_new)))

z1_new = np.zeros((len(x_new), len(y_new)))
g1_new = np.zeros((len(x_new), len(y_new)))
q1_new = np.zeros((len(x_new), len(y_new)))
h1_new = np.zeros((len(x_new), len(y_new)))

for i in range(0, len(x_new)):
    for j in range(0, len(y_new)):

        z1_new[i, j] = f1(x_new[i], y_new[j])
        g1_new[i, j] = g1(x_new[i], y_new[j])
        q1_new[i, j] = q1(x_new[i], y_new[j])
        h1_new[i, j] = h1(x_new[i], y_new[j])

Z1 = z1_new
G1 = g1_new
Q1 = q1_new
H1 = h1_new/1000.0

X, Y = np.meshgrid(x_new, y_new)

Zlevels = np.array([2.5, 3.5])
Glevels = np.array([10.0, 20.0])
Qlevels = np.array([100.0, 200.0])
Hlevels = np.array([30.0])

fig = plt.figure()
fig.set_size_inches([3.25, 3.25])
plt.ion()
plt.rc('font', family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

ZCS1 = plt.contour(X, Y, np.transpose(Z1), levels=Zlevels, colors='black')
plt.clabel(ZCS1, inline=1, fontsize=10, colors='black', fmt='%.1f', inline_spacing=1)
ZCS1.collections[0].set_linewidths(1.50)
ZCS1.collections[1].set_linewidths(1.50)
ZCS1.collections[0].set_label(r'$TCW, deg.$')

GCS1 = plt.contour(X, Y, np.transpose(G1), levels=Glevels, colors='blue', linestyles=
    '--dashed')
plt.clabel(GCS1, inline=1, fontsize=10, colors='blue', fmt='%.d', inline_spacing=0)

```

(continues on next page)

(continued from previous page)

```

GCS1.collections[0].set_linewidths(1.50)
GCS1.collections[1].set_linewidths(1.50)
GCS1.collections[0].set_label(r'$g$'+r'-load')

QCS1 = plt.contour(X, Y, np.transpose(Q1), levels=Qlevels, colors='red', linestyles=
    ↪'dotted', zorder=11)
plt.clabel(QCS1, inline=1, fontsize=10, colors='red', fmt='%.d', inline_spacing=0)
QCS1.collections[0].set_linewidths(1.50)
QCS1.collections[1].set_linewidths(1.50)
QCS1.collections[0].set_label(r'$\dot{q}^{'}, '+r'$W/cm^2$')

HCS1 = plt.contour(X, Y, np.transpose(H1), levels=Hlevels, colors='xkcd:brown',
    ↪linestyles='dashdot')
plt.clabel(HCS1, inline=1, fontsize=10, colors='xkcd:brown', fmt='%.d', inline_spacing=0)
HCS1.collections[0].set_linewidths(1.75)
HCS1.collections[0].set_label(r'$Q$+', '+r'$kJ/cm^2$')

plt.xlabel("Arrival "+r'$V_\infty$'+r', km/s', fontsize=10)
plt.ylabel("L/D", fontsize=10)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
ax = plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
plt.legend(loc='upper right', fontsize=8)

dat0 = ZCS1.allsegs[1][0]
x1,y1=dat0[:,0],dat0[:,1]
F1 = interpolate.interp1d(x1, y1, kind='linear', fill_value='extrapolate', bounds_-
    ↪error=False)

dat1 = GCS1.allsegs[0][0]
x2,y2=dat1[:,0],dat1[:,1]
F2 = interpolate.interp1d(x2, y2, kind='linear', fill_value='extrapolate', bounds_-
    ↪error=False)

dat2 = QCS1.allsegs[0][0]
x3,y3= dat2[:,0],dat2[:,1]
F3 = interpolate.interp1d(x3, y3, kind='linear', fill_value='extrapolate', bounds_-
    ↪error=False)

x4 = np.linspace(0,30,301)
y4 = F1(x4)
y5 = F2(x4)
y6 = F3(x4)

y7 = np.minimum(y5,y6)

plt.fill_between(x4, y4, y7, where=y4<=y7, color='xkcd:neon green')
plt.xlim([0.0,16.0])
plt.ylim([0.0,1.0])

plt.savefig('../plots/LuSaikia2018a-fig16.png',bbox_inches='tight')

```

(continues on next page)

(continued from previous page)

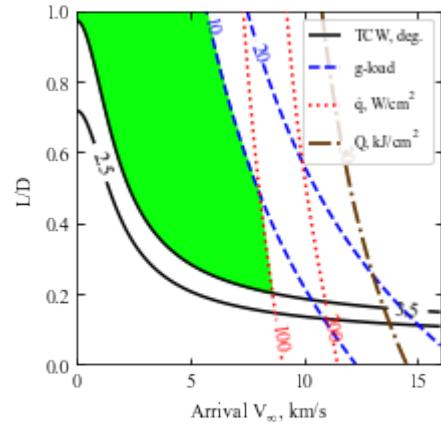
```
plt.savefig('..../plots/LuSaikia2018a-fig16.pdf', dpi=300, bbox_inches='tight')
plt.savefig('..../plots/LuSaikia2018a-fig16.eps', dpi=300, bbox_inches='tight')
```

The PostScript backend does not support transparency; partially transparent artists
will be rendered opaque.

The PostScript backend does not support transparency; partially transparent artists
will be rendered opaque.

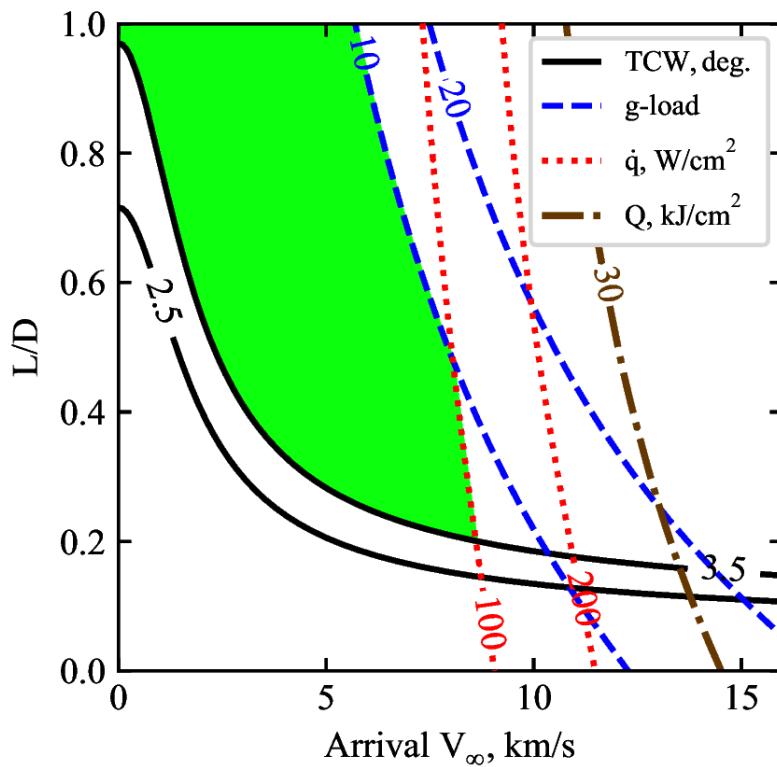
The PostScript backend does not support transparency; partially transparent artists
will be rendered opaque.

The PostScript backend does not support transparency; partially transparent artists
will be rendered opaque.



```
[18]: from IPython.display import Image
Image(filename='..../plots/LuSaikia2018a-fig16-higher-res.png', width=400)
```

[18]:



The plots are now saved in plots/LuSaikia2018a and should match with the results from the paper.

Congratulations! You have created an aerocapture feasibility chart for Titan. In the next example, you will recreate the same chart for a different vehicle ballistic coefficient.

4.6 Example - 06 - Titan Aerocapture: Part 2

In this example, you will learn to create an aerocapture feasibility chart for Titan.

For reference, we will re-create another figure from the paper “Lu and Saikia, Feasibility Assessment of Aerocapture for Future Titan Orbiter Missions, Journal of Spacecraft and Rockets, Vol. 55, No. 5, 2018”. DOI: 10.2514/1.A34121

```
[2]: from IPython.display import Image
Image(filename='../plots/lu-saikia-reference-b.png', width=600)
```

[2]:

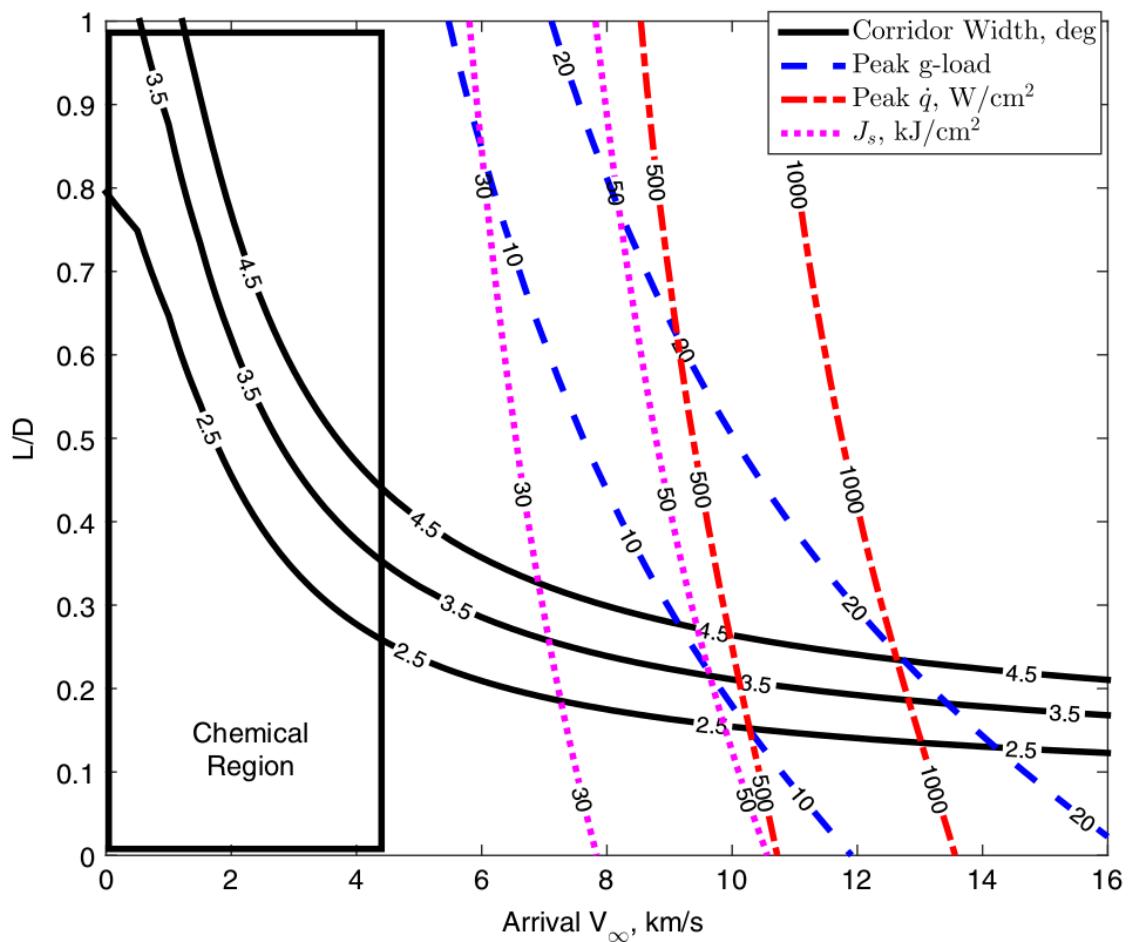


Fig. 18 Constraints summarized in a single plot for $\beta = 800 \text{ kg/m}^2$.

We will use AMAT to recreate this figure.

```
[4]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

(continues on next page)

(continued from previous page)

```

import numpy as np
from scipy import interpolate
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.patches import Polygon
import os

```

```

[5]: # Create a planet object for Titan
planet=Planet("TITAN")

# Load an nominal atmospheric profile with height, temp, pressure, density data
planet.loadAtmosphereModel('../atmdata/Titan/titan-gram-avg.dat', 0 , 1 , 2, 3)

# Define the range for arrival Vinf and vehicle L/D
#vinf_kms_array = np.linspace( 0.0,      16.0,   17)
#LD_array       = np.linspace( 0.0,      1.0 ,  11)

vinf_kms_array = np.linspace( 0.0,      16.0,   2)
LD_array       = np.linspace( 0.0,      1.0 ,  2)

```

```

[8]: # Create a directory to store the data.
# NOTE: You will get an error if the file already exists,
# Rename the folder to something else.

os.makedirs('../data/luSaikia2018b')

# Use a runID to prefix the files for easy access for post-processing.
runID = 'BC800RAP1700'

num_total      = len(vinf_kms_array)*len(LD_array)
count = 1

# Compute the inertial entry speed from the hyperbolic excess speed.
v0_kms_array    = np.zeros(len(vinf_kms_array))
v0_kms_array[:] = np.sqrt(1.0*(vinf_kms_array[:]*1E3)**2.0 + \
                          2*np.ones(len(vinf_kms_array))*\
                          planet.GM/(planet.RP+1000.0*1.0E3))/1.0E3

# Initialize matrices to store data.
overShootLimit_array = np.zeros((len(v0_kms_array),len(LD_array)))
underShootLimit_array = np.zeros((len(v0_kms_array),len(LD_array)))
exitflag_os_array    = np.zeros((len(v0_kms_array),len(LD_array)))
exitflag_us_array    = np.zeros((len(v0_kms_array),len(LD_array)))
TCW_array           = np.zeros((len(v0_kms_array),len(LD_array)))

```

```

[9]: # Compute the corridor width over the defined Vinf and L/D matrix.
# Note this will take maybe about an hour.
# If you simply want to create the plot, you can load the existing data
# in the ../data/luSaikia2018b folder as done below this cell.

for i in range(0,len(v0_kms_array)):
    for j in range(0,len(LD_array)):
        vehicle=Vehicle('Kraken', 1000.0, 800.0, LD_array[j], 3.1416, 0.0, 1.00,
                     planet)

```

(continues on next page)

(continued from previous page)

```

vehicle.setInitialState(1000.0,0.0,0.0,v0_kms_array[i],0.0,-4.5,0.0,0.0)
vehicle.setSolverParams(1E-5)
overShootLimit_array[i,j], exitflag_os_array[i,j] = \
vehicle.findOverShootLimit (6000.0, 1.0 , -88.0, -2.0, 1E-10, 1700.0)
underShootLimit_array[i,j], exitflag_us_array[i,j] = \
vehicle.findUnderShootLimit(6000.0, 1.0 , -88.0, -2.0, 1E-10, 1700.0)

TCW_array[i,j] = overShootLimit_array[i,j] - underShootLimit_array[i,j]

print("Run #"+str(count)+" of "+ str(num_total)+\
      ": Arrival V_infty: "+str(vinf_kms_array[i])+\
      " km/s", L/D:"+str(LD_array[j]) +\
      " OSL: "+str(overShootLimit_array[i,j])+\
      " USL: "+str(underShootLimit_array[i,j])+\
      ", TCW: "+str(TCW_array[i,j])+\
      " EFOS: "+str(exitflag_os_array[i,j])+\
      " EFUS: "+str(exitflag_us_array[i,j]))

count = count +1

np.savetxt('..../data/luSaikia2018b/' +runID+'vinf_kms_array.txt',vinf_kms_array)
np.savetxt('..../data/luSaikia2018b/' +runID+'v0_kms_array.txt',v0_kms_array)
np.savetxt('..../data/luSaikia2018b/' +runID+'LD_array.txt',LD_array)
np.savetxt('..../data/luSaikia2018b/' +runID+'overShootLimit_array.txt',overShootLimit_\
array)
np.savetxt('..../data/luSaikia2018b/' +runID+'exitflag_os_array.txt',exitflag_os_array)
np.savetxt('..../data/luSaikia2018b/' +runID+'undershootLimit_array.txt',underShootLimit_\
array)
np.savetxt('..../data/luSaikia2018b/' +runID+'exitflag_us_array.txt',exitflag_us_array)
np.savetxt('..../data/luSaikia2018b/' +runID+'TCW_array.txt',TCW_array)

Run #1 of 4: Arrival V_infty: 0.0 km/s, L/D:0.0 OSL: -26.956788590394353 USL: -26.\
956788590394353, TCW: 0.0 EFOS: 1.0 EFUS: 1.0
Run #2 of 4: Arrival V_infty: 0.0 km/s, L/D:1.0 OSL: -25.871938262091135 USL: -29.\
038724038098735, TCW: 3.1667857760076004 EFOS: 1.0 EFUS: 1.0
Run #3 of 4: Arrival V_infty: 16.0 km/s, L/D:0.0 OSL: -39.84376757530299 USL: -39.\
84376757530299, TCW: 0.0 EFOS: 1.0 EFUS: 1.0
Run #4 of 4: Arrival V_infty: 16.0 km/s, L/D:1.0 OSL: -36.570328679677914 USL: -73.\
23443301925363, TCW: 36.664104339575715 EFOS: 1.0 EFUS: 1.0

```

Compute the deceleration and heating loads.

```
[11]: acc_net_g_max_array      = np.zeros((len(v0_kms_array),len(LD_array)))
stag_pres_atm_max_array     = np.zeros((len(v0_kms_array),len(LD_array)))
q_stag_total_max_array     = np.zeros((len(v0_kms_array),len(LD_array)))
heatload_max_array          = np.zeros((len(v0_kms_array),len(LD_array)))

for i in range(0,len(v0_kms_array)):
    for j in range(0,len(LD_array)):
        vehicle=Vehicle('Kraken', 1000.0, 800.0, LD_array[j], 3.1416, 0.0, 1.00,_
planet)
        vehicle.setInitialState(1000.0,0.0,0.0,v0_kms_array[i],0.0,overShootLimit_\
array[i,j],0.0,0.0)
        vehicle.setSolverParams(1E-5)

        vehicle.propogateEntry(6000.0, 1.0, 180.0)
```

(continues on next page)

(continued from previous page)

```

# Extract and save variables to plot
t_min_os      = vehicle.t_minc
h_km_os       = vehicle.h_kmc
acc_net_g_os  = vehicle.acc_net_g
q_stag_con_os = vehicle.q_stag_con
q_stag_rad_os = vehicle.q_stag_rad
rc_os          = vehicle.rc
vc_os          = vehicle.vc
stag_pres_atm_os = vehicle.computeStagPres(rc_os, vc_os) / (1.01325E5)
heatload_os    = vehicle.heatload

        vehicle=Vehicle('Kraken', 1000.0, 800.0, LD_array[j], 3.1416, 0.0, 1.00,
planet)
        vehicle.setInitialState(1000.0,0.0,0.0,v0_kms_array[i],0.0,underShootLimit_
array[i,j],0.0,0.0)
        vehicle.setSolverParams(1E-5)

        vehicle.propogateEntry(6000.0, 1.0, 0.0)

# Extract and save variable to plot
t_min_us      = vehicle.t_minc
h_km_us       = vehicle.h_kmc
acc_net_g_us  = vehicle.acc_net_g
q_stag_con_us = vehicle.q_stag_con
q_stag_rad_us = vehicle.q_stag_rad
rc_us          = vehicle.rc
vc_us          = vehicle.vc
stag_pres_atm_us = vehicle.computeStagPres(rc_us, vc_us) / (1.01325E5)
heatload_us    = vehicle.heatload

q_stag_total_os = q_stag_con_os + q_stag_rad_os
q_stag_total_us = q_stag_con_us + q_stag_rad_us

acc_net_g_max_array[i,j]      = max(max(acc_net_g_os),max(acc_net_g_us))
stag_pres_atm_max_array[i,j]   = max(max(stag_pres_atm_os),max(stag_pres_atm_
os))
q_stag_total_max_array[i,j]   = max(max(q_stag_total_os),max(q_stag_total_us))
heatload_max_array[i,j]        = max(max(heatload_os),max(heatload_us))

        print("V_infty: "+str(vinf_kms_array[i])+" km/s", L/D: "+str(LD_array[j])+"_
G_MAX: "+str(acc_net_g_max_array[i,j])+" QDOT_MAX: "+str(q_stag_total_max_array[i,_
j])+" J_MAX: "+str(heatload_max_array[i,j])+" STAG. PRES: "+str(stag_pres_atm_max_
array[i,j])))

np.savetxt('..../data/luSaikia2018b/' + runID + 'acc_net_g_max_array.txt', acc_net_g_max_
array)
np.savetxt('..../data/luSaikia2018b/' + runID + 'stag_pres_atm_max_array.txt', stag_pres_atm_
max_array)
np.savetxt('..../data/luSaikia2018b/' + runID + 'q_stag_total_max_array.txt', q_stag_total_
max_array)
np.savetxt('..../data/luSaikia2018b/' + runID + 'heatload_max_array.txt', heatload_max_array)


```

```

V_infty: 0.0 km/s, L/D: 0.0 G_MAX: 0.08593800433366519 QDOT_MAX: 3.3840702698532916 J_
MAX: 3105.3432488210888 STAG. PRES: 0.006791519497339366
V_infty: 0.0 km/s, L/D: 1.0 G_MAX: 0.2138845693807405 QDOT_MAX: 4.4347653958262985 J_
MAX: 3779.309589359269 STAG. PRES: 0.004034330122158686

```

(continues on next page)

(continued from previous page)

```
V_infty: 16.0 km/s, L/D: 0.0 G_MAX: 19.241162906177003 QDOT_MAX: 1655.1562320104672 J_
↳MAX: 104695.16040529005 STAG. PRES: 1.490930766802923
V_infty: 16.0 km/s, L/D: 1.0 G_MAX: 152.35197757039305 QDOT_MAX: 3435.125582934429 J_
↳MAX: 187897.62494657672 STAG. PRES: 0.4145223323207587
```

We are now ready to create the plot!

```
[14]: x = np.loadtxt('../data/luSaikia2018b/' + runID + 'vinf_kms_array.txt')
y = np.loadtxt('../data/luSaikia2018b/' + runID + 'LD_array.txt')

Z1 = np.loadtxt('../data/luSaikia2018b/' + runID + 'TCW_array.txt')
G1 = np.loadtxt('../data/luSaikia2018b/' + runID + 'acc_net_g_max_array.txt')
Q1 = np.loadtxt('../data/luSaikia2018b/' + runID + 'q_stag_total_max_array.txt')
H1 = np.loadtxt('../data/luSaikia2018b/' + runID + 'heatload_max_array.txt')

f1 = interpolate.interp2d(x, y, np.transpose(Z1), kind='cubic')
g1 = interpolate.interp2d(x, y, np.transpose(G1), kind='cubic')
q1 = interpolate.interp2d(x, y, np.transpose(Q1), kind='cubic')
h1 = interpolate.interp2d(x, y, np.transpose(H1), kind='cubic')

x_new = np.linspace(0.0, 16, 170)
y_new = np.linspace(0.0, 1.0, 110)
z_new = np.zeros((len(x_new), len(y_new)))

z1_new = np.zeros((len(x_new), len(y_new)))
g1_new = np.zeros((len(x_new), len(y_new)))
q1_new = np.zeros((len(x_new), len(y_new)))
h1_new = np.zeros((len(x_new), len(y_new)))

for i in range(0, len(x_new)):
    for j in range(0, len(y_new)):

        z1_new[i, j] = f1(x_new[i], y_new[j])
        g1_new[i, j] = g1(x_new[i], y_new[j])
        q1_new[i, j] = q1(x_new[i], y_new[j])
        h1_new[i, j] = h1(x_new[i], y_new[j])

Z1 = z1_new
G1 = g1_new
Q1 = q1_new
H1 = h1_new / 1000.0

X, Y = np.meshgrid(x_new, y_new)

Zlevels = np.array([2.5, 3.5, 4.5])
Glevels = np.array([10.0, 20.0])
Qlevels = np.array([500.0, 1000.0])
Hlevels = np.array([30.0, 50.0])

fig = plt.figure()
fig.set_size_inches([6.5, 6.5])
plt.ion()
plt.rc('font', family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
```

(continues on next page)

(continued from previous page)

```

ZCS1 = plt.contour(X, Y, np.transpose(Z1), levels=Zlevels, colors='black')
plt.clabel(ZCS1, inline=1, fontsize=14, colors='black', fmt='%.1f', inline_spacing=1)
ZCS1.collections[0].set_linewidths(1.50)
ZCS1.collections[1].set_linewidths(1.50)
ZCS1.collections[2].set_linewidths(1.50)
ZCS1.collections[0].set_label(r'$TCW, deg$')

GCS1 = plt.contour(X, Y, np.transpose(G1), levels=Glevels, colors='blue', linestyles=
    ↪'dashed')
plt.clabel(GCS1, inline=1, fontsize=14, colors='blue', fmt='%d', inline_spacing=0)
GCS1.collections[0].set_linewidths(1.50)
GCS1.collections[1].set_linewidths(1.50)
GCS1.collections[0].set_label(r'$g$'+r'-load')

QCS1 = plt.contour(X, Y, np.transpose(Q1), levels=Qlevels, colors='red', linestyles=
    ↪'dotted', zorder=11)
plt.clabel(QCS1, inline=1, fontsize=14, colors='red', fmt='%d', inline_spacing=0)
QCS1.collections[0].set_linewidths(1.50)
QCS1.collections[1].set_linewidths(1.50)
QCS1.collections[0].set_label(r'$\dot{q}$'+', '+r'$W/cm^2$')

HCS1 = plt.contour(X, Y, np.transpose(H1), levels=Hlevels, colors='xkcd:brown',
    ↪linestyles='dashdot')
plt.clabel(HCS1, inline=1, fontsize=14, colors='xkcd:brown', fmt='%d', inline_spacing=0)
HCS1.collections[0].set_linewidths(1.75)
HCS1.collections[1].set_linewidths(1.75)
HCS1.collections[0].set_label(r'$Q$'+', '+r'$kJ/cm^2$')

plt.xlabel("Arrival "+r'$V_\infty$'+r', km/s', fontsize=16)
plt.ylabel("L/D", fontsize=16)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
ax = plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
plt.legend(loc='upper right', fontsize=16)

dat0 = ZCS1.allsegs[2][0]
x1,y1=dat0[:,0],dat0[:,1]
F1 = interpolate.interp1d(x1, y1, kind='linear', fill_value='extrapolate', bounds_-
    ↪error=False)

dat1 = GCS1.allsegs[0][0]
x2,y2=dat1[:,0],dat1[:,1]
F2 = interpolate.interp1d(x2, y2, kind='linear', fill_value='extrapolate', bounds_-
    ↪error=False)

dat2 = HCS1.allsegs[0][0]
x3,y3= dat2[:,0],dat2[:,1]
F3 = interpolate.interp1d(x3, y3, kind='linear', fill_value='extrapolate', bounds_-
    ↪error=False)

x4 = np.linspace(0, 30, 301)

```

(continues on next page)

(continued from previous page)

```

y4 = F1(x4)
y5 = F2(x4)
y6 = F3(x4)

y7 = np.minimum(y5,y6)

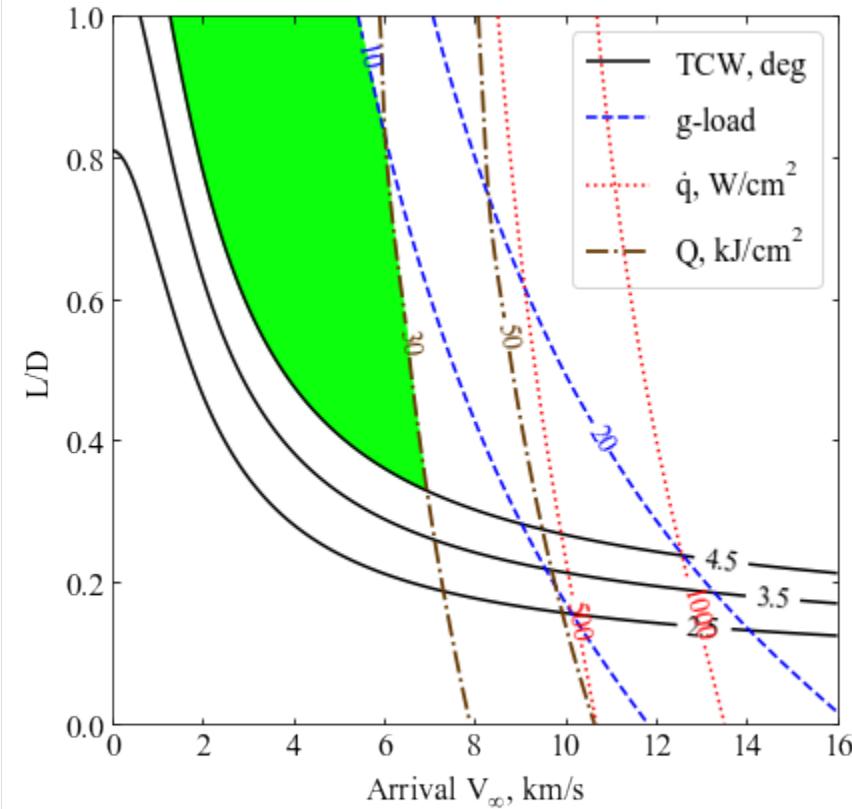
plt.fill_between(x4, y4, y7, where=y4<=y7,color='xkcd:neon green')
plt.xlim([0.0,16.0])
plt.ylim([0.0,1.0])

plt.savefig('../plots/LuSaikia2018b-fig18.png',bbox_inches='tight')
plt.savefig('../plots/LuSaikia2018b-fig18.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/LuSaikia2018b-fig18.eps', dpi=300,bbox_inches='tight')

plt.show()

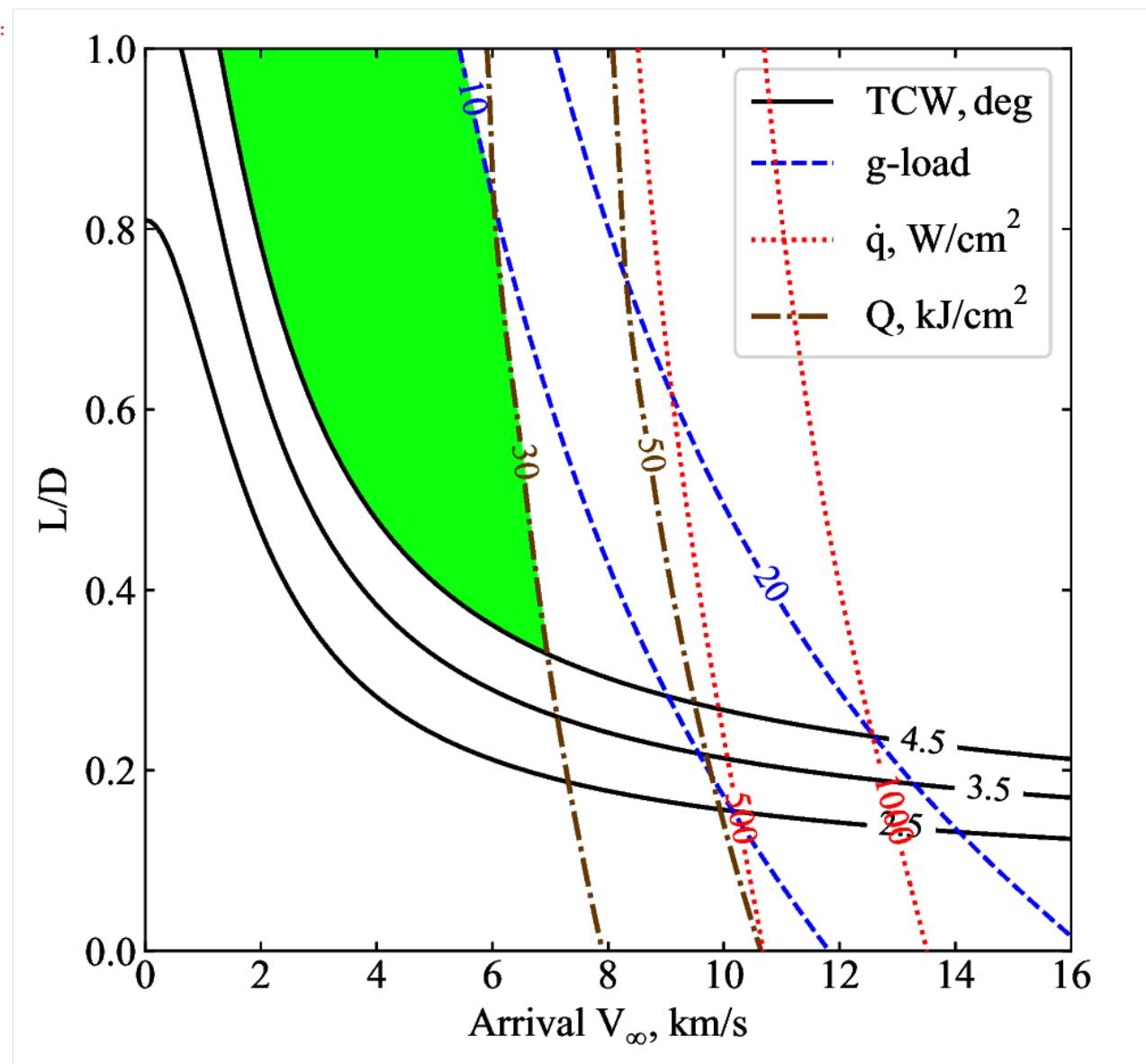
```

The PostScript backend does not support transparency; partially transparent artists
 ↵ will be rendered opaque.
 The PostScript backend does not support transparency; partially transparent artists
 ↵ will be rendered opaque.
 The PostScript backend does not support transparency; partially transparent artists
 ↵ will be rendered opaque.
 The PostScript backend does not support transparency; partially transparent artists
 ↵ will be rendered opaque.



```
[16]: from IPython.display import Image
Image(filename='../plots/luSaikia2018b-higher-res.png', width=600)
```

[16]:



The plots are now saved in plots/LuSaikia2018b and should match with the results from the paper.

Congratulations! You have created another aerocapture feasibility chart for Titan. In the next example, you will recreate the similar feasibility charts for Venus.

4.7 Example - 07 - Venus Aerocapture: Part 3

In this example, you will learn to create an aerocapture feasibility chart for Venus.

For reference, we will re-create a figure from the paper “Girija, Lu, and Saikia, Feasibility and Mass-Benefit Analysis of Aerocapture for Missions to Venus, Journal of Spacecraft and Rockets, Vol. 57, No. 1” DOI: 10.2514/1.A34529

```
[1]: from IPython.display import Image
Image(filename='../plots/girijaYe2020a-reference.png', width=600)
```

[1]:

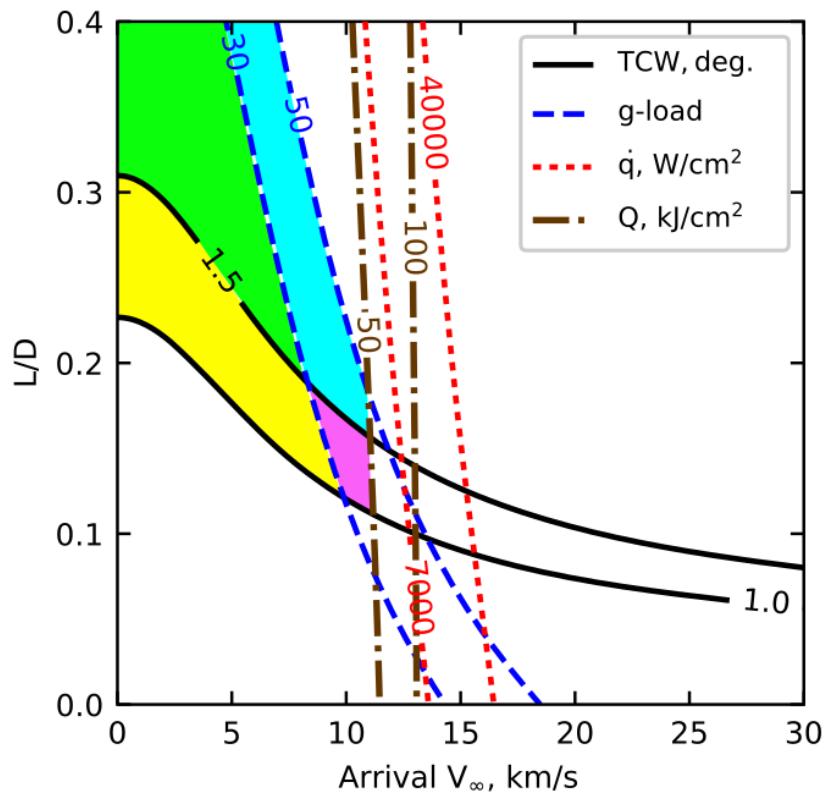


Fig. 12 Feasible design space for lift modulation aerocapture with $\beta = 50 \text{ kg/m}^2$.

```
[3]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle

import numpy as np
from scipy import interpolate

import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.patches import Polygon
import os
```

```
[4]: # Create a planet object
planet=Planet("VENUS")
planet.h_skip = 150000.0

# Load an nominal atmospheric profile with height, temp, pressure, density data
planet.loadAtmosphereModel('../atmdata/Venus/venus-gram-avg.dat', 0 , 1 ,2, 3 )

vinf_kms_array = np.linspace( 0.0,      30.0,    11)
LD_array       = np.linspace( 0.0,      0.4 ,    11)
```

```
[5]: os.makedirs('../data/girijaYe2019a')
runID = 'BC50RAP400EI150-'

num_total      = len(vinf_kms_array)*len(LD_array)
count = 1

v0_kms_array    = np.zeros(len(vinf_kms_array))
v0_kms_array[:] = np.sqrt(1.0*(vinf_kms_array[:]*1E3)**2.0 +\
                         2*np.ones(len(vinf_kms_array))*\
                         planet.GM/(planet.RP+150.0*1.0E3))/1.0E3

overShootLimit_array = np.zeros((len(v0_kms_array),len(LD_array)))
underShootLimit_array = np.zeros((len(v0_kms_array),len(LD_array)))
exitflag_os_array = np.zeros((len(v0_kms_array),len(LD_array)))
exitflag_us_array = np.zeros((len(v0_kms_array),len(LD_array)))
TCW_array = np.zeros((len(v0_kms_array),len(LD_array)))

[6]: # NOTE: This will take about an hour.

for i in range(0,len(v0_kms_array)):
    for j in range(0,len(LD_array)):
        vehicle=Vehicle('Apollo', 1000.0, 50.0, LD_array[j], 3.1416, 0.0, 1.00,_
        ↪planet)
        vehicle.setInitialState(150.0,0.0,0.0,v0_kms_array[i],0.0,-4.5,0.0,0.0)
        vehicle.setSolverParams(1E-6)
        overShootLimit_array[i,j], exitflag_os_array[i,j] = vehicle.
        ↪findOverShootLimit (2400.0, 0.1, -80.0, -4.0, 1E-10, 400.0)
        underShootLimit_array[i,j], exitflag_us_array[i,j] = vehicle.
        ↪findUnderShootLimit(2400.0, 0.1, -80.0, -4.0, 1E-10, 400.0)

        TCW_array[i,j] = overShootLimit_array[i,j] - underShootLimit_array[i,j]

        print("Run #"+str(count)+" of "+ str(num_total)+": Arrival V_infty:
        ↪"+str(vinf_kms_array[i])+" km/s+", L/D:"+str(LD_array[j]) + " OSL:
        ↪"+str(overShootLimit_array[i,j])+" USL: "+str(underShootLimit_array[i,j])+", TCW:
        ↪"+str(TCW_array[i,j])+" EFOS: "+str(exitflag_os_array[i,j])+ " EFUS: "+str(exitflag_-
        ↪us_array[i,j]))
        count = count +1

np.savetxt('../data/girijaYe2019a/'+runID+'vinf_kms_array.txt',vinf_kms_array)
np.savetxt('../data/girijaYe2019a/'+runID+'v0_kms_array.txt',v0_kms_array)
np.savetxt('../data/girijaYe2019a/'+runID+'LD_array.txt',LD_array)
np.savetxt('../data/girijaYe2019a/'+runID+'overShootLimit_array.txt',overShootLimit_-
array)
np.savetxt('../data/girijaYe2019a/'+runID+'exitflag_os_array.txt',exitflag_os_array)
np.savetxt('../data/girijaYe2019a/'+runID+'undershootLimit_array.txt',underShootLimit_-
array)
np.savetxt('../data/girijaYe2019a/'+runID+'exitflag_us_array.txt',exitflag_us_array)
np.savetxt('../data/girijaYe2019a/'+runID+'TCW_array.txt',TCW_array)

Run #1 of 4: Arrival V_infty: 0.0 km/s, L/D:0.0 OSL: -5.0166317711664306 USL: -5.
↪0166317711664306, TCW: 0.0 EFOS: 1.0 EFUS: 1.0
Run #2 of 4: Arrival V_infty: 0.0 km/s, L/D:0.4 OSL: -4.650539502421452 USL: -6.
↪815560043607547, TCW: 2.165020541186095 EFOS: 1.0 EFUS: 1.0
Run #3 of 4: Arrival V_infty: 30.0 km/s, L/D:0.0 OSL: -7.44011820455853 USL: -7.
↪44011820455853, TCW: 0.0 EFOS: 1.0 EFUS: 1.0
```

(continues on next page)

(continued from previous page)

Run #4 of 4: Arrival V_infty: 30.0 km/s, L/D:0.4 OSL: -6.604976565784455 USL: -22.
 ↵98797720550283, TCW: 16.383000639718375 EFOS: 1.0 EFUS: 1.0

```
[7]: acc_net_g_max_array      = np.zeros((len(v0_kms_array), len(LD_array)))
stag_pres_atm_max_array    = np.zeros((len(v0_kms_array), len(LD_array)))
q_stag_total_max_array     = np.zeros((len(v0_kms_array), len(LD_array)))
heatload_max_array         = np.zeros((len(v0_kms_array), len(LD_array)))

for i in range(0,len(v0_kms_array)):
    for j in range(0,len(LD_array)):
        vehicle=Vehicle('Apollo', 1000.0, 50.0, LD_array[j], 3.1416, 0.0, 1.00,
        ↵planet)
        vehicle.setInitialState(150.0,0.0,0.0,v0_kms_array[i],0.0,overShootLimit_
        ↵array[i,j],0.0,0.0)
        vehicle.setSolverParams(1E-6)
        vehicle.propagateEntry (2400.0, 0.1, 180.0)

        # Extract and save variables to plot
        t_min_os      = vehicle.t_minc
        h_km_os       = vehicle.h_kmc
        acc_net_g_os  = vehicle.acc_net_g
        q_stag_con_os = vehicle.q_stag_con
        q_stag_rad_os = vehicle.q_stag_rad
        rc_os          = vehicle.rc
        vc_os          = vehicle.vc
        stag_pres_atm_os = vehicle.computeStagPres(rc_os,vc_os) / (1.01325E5)
        heatload_os    = vehicle.heatload

        vehicle=Vehicle('Apollo', 1000.0, 50.0, LD_array[j], 3.1416, 0.0, 1.00,
        ↵planet)
        vehicle.setInitialState(150.0,0.0,0.0,v0_kms_array[i],0.0,underShootLimit_
        ↵array[i,j],0.0,0.0)
        vehicle.setSolverParams(1E-6)
        vehicle.propagateEntry (2400.0, 0.1, 0.0)

        # Extract and save variable to plot
        t_min_us      = vehicle.t_minc
        h_km_us       = vehicle.h_kmc
        acc_net_g_us  = vehicle.acc_net_g
        q_stag_con_us = vehicle.q_stag_con
        q_stag_rad_us = vehicle.q_stag_rad
        rc_us          = vehicle.rc
        vc_us          = vehicle.vc
        stag_pres_atm_us = vehicle.computeStagPres(rc_us,vc_us) / (1.01325E5)
        heatload_us    = vehicle.heatload

        q_stag_total_os = q_stag_con_os + q_stag_rad_os
        q_stag_total_us = q_stag_con_us + q_stag_rad_us

        acc_net_g_max_array[i,j]      = max(max(acc_net_g_os),max(acc_net_g_us))
        stag_pres_atm_max_array[i,j]   = max(max(stag_pres_atm_os),max(stag_pres_atm_
        ↵os))
        q_stag_total_max_array[i,j]   = max(max(q_stag_total_os),max(q_stag_total_us))
        heatload_max_array[i,j]       = max(max(heatload_os),max(heatload_us))
```

(continues on next page)

(continued from previous page)

```

print("V_infty: "+str(vinf_kms_array[i])+", L/D: "+str(LD_array[j])+"_"
˓→G_MAX: "+str(acc_net_g_max_array[i,j])+" QDOT_MAX: "+str(q_stag_total_max_array[i,
˓→j])+" J_MAX: "+str(heatload_max_array[i,j])+" STAG. PRES: "+str(stag_pres_atm_max_
˓→array[i,j]))

np.savetxt('..../data/girijaYe2019a/' + runID + 'acc_net_g_max_array.txt', acc_net_g_max_
˓→array)
np.savetxt('..../data/girijaYe2019a/' + runID + 'stag_pres_atm_max_array.txt', stag_pres_atm_
˓→max_array)
np.savetxt('..../data/girijaYe2019a/' + runID + 'q_stag_total_max_array.txt', q_stag_total_
˓→max_array)
np.savetxt('..../data/girijaYe2019a/' + runID + 'heatload_max_array.txt', heatload_max_array)

V_infty: 0.0 km/s, L/D: 0.0 G_MAX: 3.953230700877381 QDOT_MAX: 108.34036601208729 J_
˓→MAX: 10463.898911024176 STAG. PRES: 0.019145177375447076
V_infty: 0.0 km/s, L/D: 0.4 G_MAX: 15.806088566064293 QDOT_MAX: 209.94886974217718 J_
˓→MAX: 18096.44195688157 STAG. PRES: 0.006702446102743174
V_infty: 30.0 km/s, L/D: 0.0 G_MAX: 139.56205504004248 QDOT_MAX: 34169148.177942365 J_
˓→MAX: 236829784.25897062 STAG. PRES: 0.6754469940813033
V_infty: 30.0 km/s, L/D: 0.4 G_MAX: 1480.2009655208392 QDOT_MAX: 396622642.1911957 J_
˓→MAX: 353048193.59636647 STAG. PRES: 0.1343203200123287

```

```
[8]: x = np.loadtxt('..../data/girijaYe2019a/' + runID + 'vinf_kms_array.txt')
y = np.loadtxt('..../data/girijaYe2019a/' + runID + 'LD_array.txt')

Z1 = np.loadtxt('..../data/girijaYe2019a/' + runID + 'TCW_array.txt')
G1 = np.loadtxt('..../data/girijaYe2019a/' + runID + 'acc_net_g_max_array.txt')
Q1 = np.loadtxt('..../data/girijaYe2019a/' + runID + 'q_stag_total_max_array.txt')
H1 = np.loadtxt('..../data/girijaYe2019a/' + runID + 'heatload_max_array.txt')
S1 = np.loadtxt('..../data/girijaYe2019a/' + runID + 'stag_pres_atm_max_array.txt')

f1 = interpolate.interp2d(x, y, np.transpose(Z1), kind='cubic')
g1 = interpolate.interp2d(x, y, np.transpose(G1), kind='cubic')
q1 = interpolate.interp2d(x, y, np.transpose(Q1), kind='cubic')
h1 = interpolate.interp2d(x, y, np.transpose(H1), kind='cubic')
s1 = interpolate.interp2d(x, y, np.transpose(S1), kind='cubic')

x_new = np.linspace( 0.0, 30, 110)
y_new = np.linspace( 0.0, 0.4 ,110)
z_new = np.zeros((len(x_new),len(y_new)))

z1_new = np.zeros((len(x_new),len(y_new)))
g1_new = np.zeros((len(x_new),len(y_new)))
q1_new = np.zeros((len(x_new),len(y_new)))
h1_new = np.zeros((len(x_new),len(y_new)))
s1_new = np.zeros((len(x_new),len(y_new)))

for i in range(0,len(x_new)):
    for j in range(0,len(y_new)):

        z1_new[i,j] = f1(x_new[i],y_new[j])
        g1_new[i,j] = g1(x_new[i],y_new[j])
        q1_new[i,j] = q1(x_new[i],y_new[j])

```

(continues on next page)

(continued from previous page)

```

h1_new[i,j] = h1(x_new[i],y_new[j])
s1_new[i,j] = s1(x_new[i],y_new[j])

Z1 = z1_new
G1 = g1_new
Q1 = q1_new
S1 = s1_new
H1 = h1_new/1000.0

X, Y = np.meshgrid(x_new, y_new)

Zlevels = np.array([1.0,1.5])
Glevels = np.array([30.0, 50.0])
Qlevels = np.array([7000.0, 40000.0])
Hlevels = np.array([50.0, 100.0])
Slevels = np.array([0.8])

fig = plt.figure()
fig.set_size_inches([3.25,3.25])
plt.ion()
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

ZCS1 = plt.contour(X, Y, np.transpose(Z1), levels=Zlevels, colors='black')
plt.clabel(ZCS1, inline=1, fontsize=10, colors='black', fmt='%.1f', inline_spacing=1)
ZCS1.collections[0].set_linewidths(1.50)
ZCS1.collections[1].set_linewidths(1.50)
ZCS1.collections[0].set_label(r'$TCW, deg.$')

GCS1 = plt.contour(X, Y, np.transpose(G1), levels=Glevels, colors='blue',linestyles=
    ↪'dashed')
plt.clabel(GCS1, inline=1, fontsize=10, colors='blue', fmt='%.d', inline_spacing=0)
GCS1.collections[0].set_linewidths(1.50)
GCS1.collections[1].set_linewidths(1.50)
GCS1.collections[0].set_label(r'$g$'+r'-load')

QCS1 = plt.contour(X, Y, np.transpose(Q1), levels=Qlevels, colors='red',linestyles=
    ↪'dotted',zorder=11)
plt.clabel(QCS1, inline=1, fontsize=10, colors='red', fmt='%.d', inline_spacing=0)
QCS1.collections[0].set_linewidths(1.50)
QCS1.collections[1].set_linewidths(1.50)
QCS1.collections[0].set_label(r'$\dot{q}+$', '+r'$W/cm^2$')

HCS1 = plt.contour(X, Y, np.transpose(H1), levels=Hlevels, colors='xkcd:brown',
    ↪linestyles='dashdot')
plt.clabel(HCS1, inline=1, fontsize=10, colors='xkcd:brown', fmt='%.d', inline_spacing=0)
HCS1.collections[0].set_linewidths(1.75)
HCS1.collections[1].set_linewidths(1.75)
HCS1.collections[0].set_label(r'$Q$+', 'r'$kJ/cm^2$')

#SCS1 = plt.contour(X, Y, transpose(S1), levels=Slevels, colors='cyan')

```

(continues on next page)

(continued from previous page)

```

# plt.clabel(SCS1, inline=1, fontsize=10, colors='xkcd:neon green', fmt='%.1f', inline_
→spacing=1)
#SCS1.collections[0].set_linewidths(1.75)
#SCS1.collections[0].set_label(r'$Peak$'+r' '+r'$stag. pressure,atm$')

plt.xlabel("Arrival "+r'$V_\infty$'+r', km/s' ,fontsize=10)
plt.ylabel("L/D",fontsize=10)
plt.xticks(fontsize=10)
plt.yticks(np.array([ 0.0,  0.1,  0.2,  0.3,  0.4]),fontsize=10)
ax = plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
plt.legend(loc='upper right', fontsize=8)

dat0 = ZCS1.allsegs[1][0]
x1,y1=dat0[:,0],dat0[:,1]
F1 = interpolate.interp1d(x1, y1, kind='linear',fill_value='extrapolate', bounds_
→error=False)

dat1 = GCS1.allsegs[0][0]
x2,y2=dat1[:,0],dat1[:,1]
F2 = interpolate.interp1d(x2, y2, kind='linear',fill_value='extrapolate', bounds_
→error=False)

dat2 = HCS1.allsegs[0][0]
x3,y3=dat2[:,0],dat2[:,1]
F3 = interpolate.interp1d(x3, y3, kind='linear',fill_value='extrapolate', bounds_
→error=False)

dat0a = ZCS1.allsegs[0][0]
x1a,y1a=dat0a[:,0],dat0a[:,1]
F1a = interpolate.interp1d(x1a, y1a, kind='linear',fill_value='extrapolate', bounds_
→error=False)

x4 = np.linspace(0,30,301)
y4 = F1(x4)
y4a =F1a(x4)
y5 = F2(x4)
y6 = F3(x4)

y7 = np.minimum(y5,y6)
y8 = np.minimum(y4,y5)

plt.fill_between(x4, y4, y7, where=y4<=y7,color='xkcd:neon green')

plt.fill_between(x4, y4a, y8, where=y4a<=y8,color='xkcd:bright yellow')

plt.xlim([0.0,30.0])
plt.ylim([0.0,0.4])

plt.savefig('../plots/giri jaYe2019a-fig12.png',bbox_inches='tight')
plt.savefig('../plots/giri jaYe2019a-fig12.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/giri jaYe2019a-fig12.eps', dpi=300,bbox_inches='tight')

```

(continues on next page)

(continued from previous page)

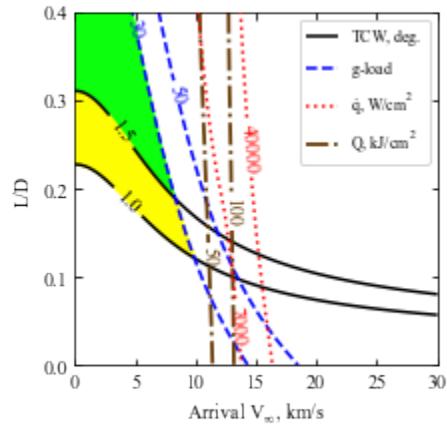
```
plt.show()
```

The PostScript backend does not support transparency; partially transparent artists
will be rendered opaque.

The PostScript backend does not support transparency; partially transparent artists
will be rendered opaque.

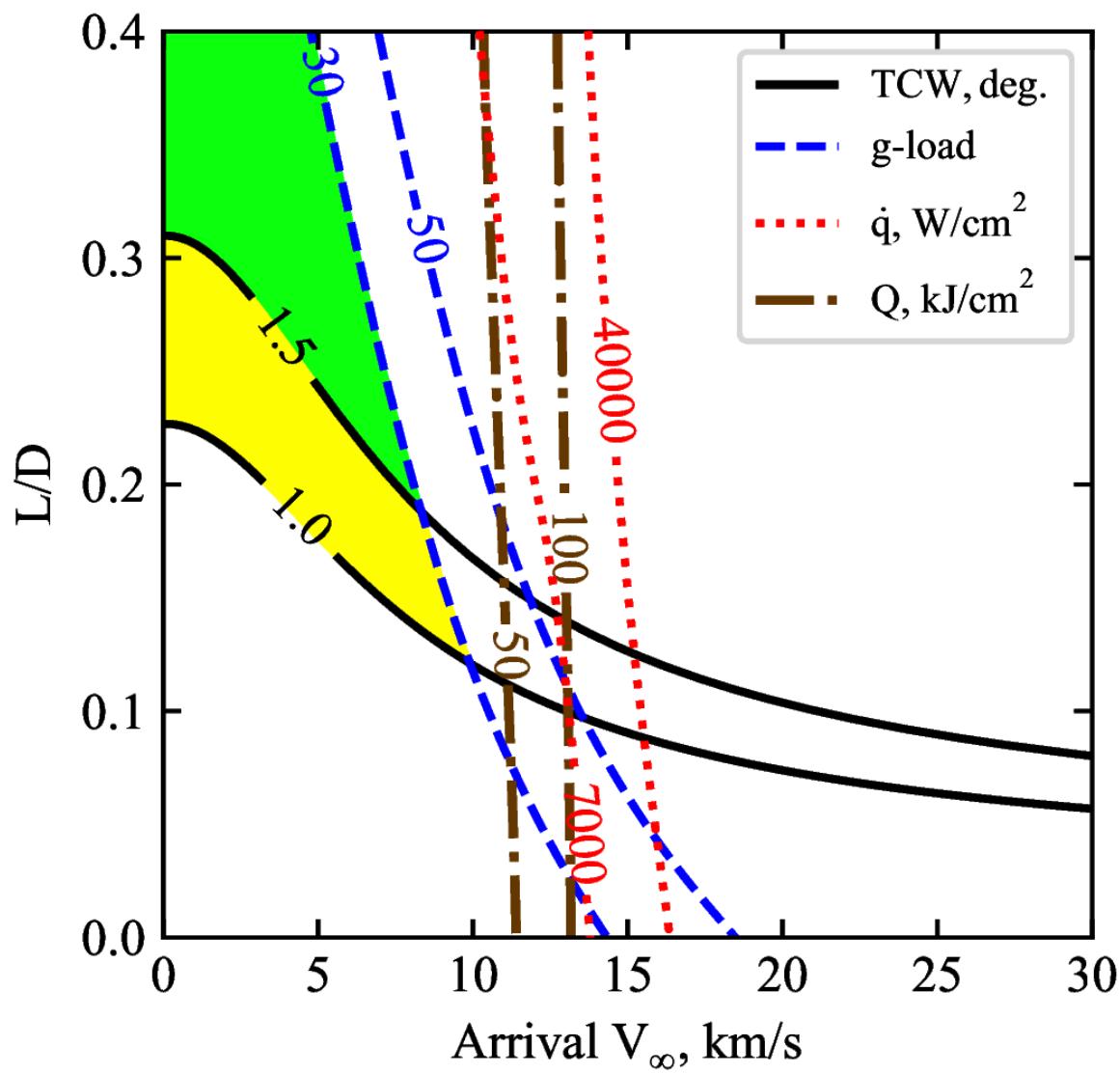
The PostScript backend does not support transparency; partially transparent artists
will be rendered opaque.

The PostScript backend does not support transparency; partially transparent artists
will be rendered opaque.



```
[11]: from IPython.display import Image
Image(filename='../plots/girijaYe2020a-higher-res.png', width=600)
```

[11]:



The plots are now saved in plots/girijaYe2019a and should match with the results from the paper.

Congratulations! You are now becoming an expert at creating aerocapture feasibility charts using AMAT. In the next example, we will create a similar feasibility chart for drag modulation aerocapture at Venus.

4.8 Example - 08 - Venus Aerocapture: Part 4

In this example, you will learn to create a drag modulation aerocapture feasibility chart for Venus.

For reference, we will re-create another figure from the paper “Girija, Lu, and Saikia, Feasibility and Mass-Benefit Analysis of Aerocapture for Missions to Venus, Journal of Spacecraft and Rockets, Vol. 57, No. 1” DOI: 10.2514/1.A34529

```
[1]: from IPython.display import Image
Image(filename='../plots/girijaYe2020c-reference.png', width=600)
```

[1]:

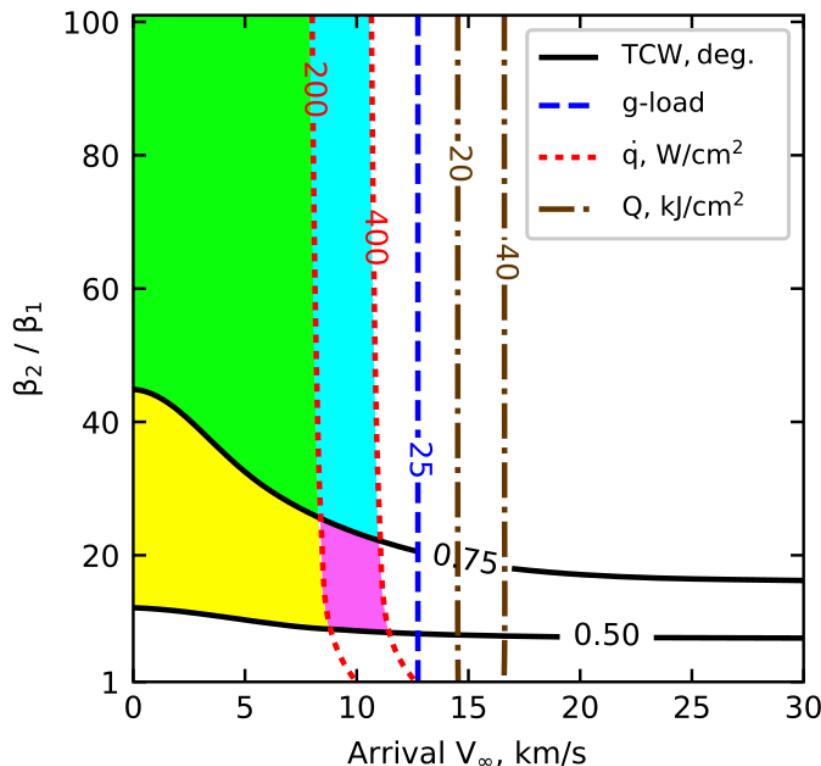


Fig. 16 Feasible design space for drag modulation aerocapture with $\beta_1 = 5 \text{ kg/m}^2$.

```
[2]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle

import numpy as np
from scipy import interpolate

import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.patches import Polygon
import os
```

```
[3]: # Create a planet object
planet=Planet("VENUS")
planet.h_skip = 150000.0

# Load an nominal atmospheric profile with height, temp, pressure, density data
planet.loadAtmosphereModel('../atmdata/Venus/venus-gram-avg.dat', 0 , 1 ,2, 3)

vinf_kms_array = np.linspace( 0.0,    30.0,    11)
betaRatio_array = np.linspace( 1.0,   101.0 , 11)

#vinf_kms_array = np.linspace( 0.0,    30.0,    2)
#betaRatio_array = np.linspace( 1.0,   101.0 , 2)
```

```
[4]: beta1 = 5.0

os.makedirs('../data/girijaYe2019c')
runID = 'DMBC5RAP400EI150'

[5]: v0_kms_array = np.zeros(len(vinf_kms_array))
v0_kms_array[:] = np.sqrt(1.0*(vinf_kms_array[:]*1E3)**2.0 + 2*np.ones(len(vinf_kms_
array))*planet.GM/(planet.RP+150.0*1.0E3))/1.0E3

overShootLimit_array = np.zeros((len(v0_kms_array),len(betaRatio_array)))
underShootLimit_array = np.zeros((len(v0_kms_array),len(betaRatio_array)))
exitflag_os_array = np.zeros((len(v0_kms_array),len(betaRatio_array)))
exitflag_us_array = np.zeros((len(v0_kms_array),len(betaRatio_array)))
TCW_array = np.zeros((len(v0_kms_array),len(betaRatio_array)))

[6]: for i in range(0,len(v0_kms_array)):
    for j in range(0,len(betaRatio_array)):
        vehicle=Vehicle('DMVehicle', 1500.0, beta1, 0.0, 3.1416, 0.0, 0.10, planet)
        vehicle.setInitialState(150.0,0.0,0.0,v0_kms_array[i],0.0,-4.5,0.0,0.0)
        vehicle.setSolverParams(1E-6)
        vehicle.setDragModulationVehicleParams(beta1,betaRatio_array[j])

        underShootLimit_array[i,j], exitflag_us_array[i,j] = vehicle.
        ↪findUnderShootLimitD(2400.0, 0.1, -80.0,-4.0,1E-10,400.0)
        overShootLimit_array[i,j], exitflag_os_array[i,j] = vehicle.
        ↪findOverShootLimitD (2400.0, 0.1, -80.0,-4.0,1E-10,400.0)

        TCW_array[i,j] = overShootLimit_array[i,j] - underShootLimit_array[i,j]

        print('VINF: '+str(vinf_kms_array[i])+' km/s, BETA RATIO: '+str(betaRatio_
        ↪array[j])+' TCW: '+str(TCW_array[i,j])+' deg.')

np.savetxttxt('../data/girijaYe2019c/'+runID+'vinf_kms_array.txt',vinf_kms_array)
np.savetxttxt('../data/girijaYe2019c/'+runID+'v0_kms_array.txt',v0_kms_array)
np.savetxttxt('../data/girijaYe2019c/'+runID+'betaRatio_array.txt',betaRatio_array)
np.savetxttxt('../data/girijaYe2019c/'+runID+'overShootLimit_array.txt',overShootLimit_
        ↪array)
np.savetxttxt('../data/girijaYe2019c/'+runID+'exitflag_os_array.txt',exitflag_os_array)
np.savetxttxt('../data/girijaYe2019c/'+runID+'underShootLimit_array.txt',underShootLimit_
        ↪array)
np.savetxttxt('../data/girijaYe2019c/'+runID+'exitflag_us_array.txt',exitflag_us_array)
np.savetxttxt('../data/girijaYe2019c/'+runID+'TCW_array.txt',TCW_array)

VINF: 0.0 km/s, BETA RATIO: 1.0 TCW: 0.0 deg.
VINF: 0.0 km/s, BETA RATIO: 101.0 TCW: 0.8981981236029242 deg.
VINF: 30.0 km/s, BETA RATIO: 1.0 TCW: 0.0 deg.
VINF: 30.0 km/s, BETA RATIO: 101.0 TCW: 1.192248803385155 deg.

[8]: acc_net_g_max_array = np.zeros((len(v0_kms_array),len(betaRatio_array)))
stag_pres_atm_max_array = np.zeros((len(v0_kms_array),len(betaRatio_array)))
q_stag_total_max_array = np.zeros((len(v0_kms_array),len(betaRatio_array)))
heatload_max_array = np.zeros((len(v0_kms_array),len(betaRatio_array)))

for i in range(0,len(v0_kms_array)):
```

(continues on next page)

(continued from previous page)

```

for j in range(0,len(betaRatio_array)):
    vehicle=Vehicle('DMVehicle', 1500.0, beta1, 0.0, 3.1416, 0.0, 0.10, planet)
    vehicle.setInitialState(150.0,0.0,0.0,v0_kms_array[i],0.0,overShootLimit_
    ↪array[i,j],0.0,0.0)
    vehicle.setSolverParams(1E-6)

    vehicle.propogateEntry (2400.0, 0.1, 0.0)

    # Extract and save variables to plot
    t_min_os      = vehicle.t_minc
    h_km_os       = vehicle.h_kmc
    acc_net_g_os  = vehicle.acc_net_g
    q_stag_con_os = vehicle.q_stag_con
    q_stag_rad_os = vehicle.q_stag_rad
    rc_os          = vehicle.rc
    vc_os          = vehicle.vc
    stag_pres_atm_os = vehicle.computeStagPres(rc_os,vc_os) / (1.01325E5)
    heatload_os   = vehicle.heatload

    vehicle=Vehicle('DMVehicle', 1500.0, beta1*betaRatio_array[j], 0.0, 3.1416, 0.
    ↪0, 0.10, planet)
    vehicle.setInitialState(150.0,0.0,0.0,v0_kms_array[i],0.0,underShootLimit_
    ↪array[i,j],0.0,0.0)
    vehicle.setSolverParams( 1E-6)

    vehicle.propogateEntry (2400.0, 0.1, 0.0)

    # Extract and save variable to plot
    t_min_us      = vehicle.t_minc
    h_km_us       = vehicle.h_kmc
    acc_net_g_us  = vehicle.acc_net_g
    q_stag_con_us = vehicle.q_stag_con
    q_stag_rad_us = vehicle.q_stag_rad
    rc_us          = vehicle.rc
    vc_us          = vehicle.vc
    stag_pres_atm_us = vehicle.computeStagPres(rc_us,vc_us) / (1.01325E5)
    heatload_us   = vehicle.heatload

    q_stag_total_os  = q_stag_con_os + q_stag_rad_os
    q_stag_total_us  = q_stag_con_us + q_stag_rad_us

    acc_net_g_max_array[i,j]      = max(max(acc_net_g_os),max(acc_net_g_us))
    stag_pres_atm_max_array[i,j]   = max(max(stag_pres_atm_os),max(stag_pres_atm_
    ↪os))
    q_stag_total_max_array[i,j]   = max(max(q_stag_total_os),max(q_stag_total_us))
    heatload_max_array[i,j]       = max(max(heatload_os),max(heatload_us))

    print("V_infty: "+str(vinf_kms_array[i])+" km/s", BR: "+str(betaRatio_
    ↪array[j])+" G_MAX: "+str(acc_net_g_max_array[i,j])+" QDOT_MAX: "+str(q_stag_total_
    ↪max_array[i,j])+" J_MAX: "+str(heatload_max_array[i,j])+" STAG. PRES: "+str(stag_
    ↪pres_atm_max_array[i,j]))

np.savetxt('../data/girijaYe2019c/'+runID+'acc_net_g_max_array.txt',acc_net_g_max_
↪array)
np.savetxt('../data/girijaYe2019c/'+runID+'stag_pres_atm_max_array.txt',stag_pres_atm_
↪max_array)

```

(continues on next page)

(continued from previous page)

```

np.savetxt('..../data/girijaYe2019c/' + runID + 'q_stag_total_max_array.txt', q_stag_total_
           ↵max_array)
np.savetxt('..../data/girijaYe2019c/' + runID + 'heatload_max_array.txt', heatload_max_array)

V_infty: 0.0 km/s, BR: 1.0 G_MAX: 4.2619678756684065 QDOT_MAX: 107.04783114721339 J_
           ↵MAX: 9963.95886102549 STAG. PRES: 0.00206384918146089
V_infty: 0.0 km/s, BR: 101.0 G_MAX: 4.2619678756684065 QDOT_MAX: 1069.8045659113661 J_
           ↵MAX: 105153.41622997135 STAG. PRES: 0.00206384918146089
V_infty: 30.0 km/s, BR: 1.0 G_MAX: 143.43437041606074 QDOT_MAX: 763604.549664235 J_
           ↵MAX: 4964898.795992032 STAG. PRES: 0.06941820444237855
V_infty: 30.0 km/s, BR: 101.0 G_MAX: 143.43437041606074 QDOT_MAX: 182872814.4356986 J_
           ↵MAX: 1231040818.2242763 STAG. PRES: 0.06941820444237855

```

```

[9]: x = np.loadtxt('..../data/girijaYe2019c/' + runID + 'vinf_kms_array.txt')
y = np.loadtxt('..../data/girijaYe2019c/' + runID + 'betaRatio_array.txt')

Z1 = np.loadtxt('..../data/girijaYe2019c/' + runID + 'TCW_array.txt')
G1 = np.loadtxt('..../data/girijaYe2019c/' + runID + 'acc_net_g_max_array.txt')
Q1 = np.loadtxt('..../data/girijaYe2019c/' + runID + 'q_stag_total_max_array.txt')
H1 = np.loadtxt('..../data/girijaYe2019c/' + runID + 'heatload_max_array.txt')
S1 = np.loadtxt('..../data/girijaYe2019c/' + runID + 'stag_pres_atm_max_array.txt')

f1 = interpolate.interp2d(x, y, np.transpose(Z1), kind='cubic')
g1 = interpolate.interp2d(x, y, np.transpose(G1), kind='cubic')
q1 = interpolate.interp2d(x, y, np.transpose(Q1), kind='cubic')
h1 = interpolate.interp2d(x, y, np.transpose(H1), kind='cubic')
s1 = interpolate.interp2d(x, y, np.transpose(S1), kind='cubic')

x_new = np.linspace( 0.0,    30,    110)
y_new = np.linspace( 0.0,   101 , 110)

z1_new = np.zeros((len(x_new), len(y_new)))
g1_new = np.zeros((len(x_new), len(y_new)))
q1_new = np.zeros((len(x_new), len(y_new)))
h1_new = np.zeros((len(x_new), len(y_new)))
s1_new = np.zeros((len(x_new), len(y_new)))

for i in range(0, len(x_new)):
    for j in range(0, len(y_new)):

        z1_new[i,j] = f1(x_new[i],y_new[j])
        g1_new[i,j] = g1(x_new[i],y_new[j])
        q1_new[i,j] = q1(x_new[i],y_new[j])
        h1_new[i,j] = h1(x_new[i],y_new[j])
        s1_new[i,j] = s1(x_new[i],y_new[j])

Z1 = z1_new
G1 = g1_new
Q1 = q1_new
S1 = s1_new
H1 = h1_new/1000.0

X, Y = np.meshgrid(x_new, y_new)

```

(continues on next page)

(continued from previous page)

```

Zlevels = np.array([0.5,0.75])
Glevels = np.array([25.0])
Qlevels = np.array([200.0, 400.0])
Hlevels = np.array([20.0,40.0])

fig = plt.figure()
fig.set_size_inches([3.25,3.25])
plt.ion()
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

ZCS1 = plt.contour(X, Y, np.transpose(Z1), levels=Zlevels, colors='black')
plt.clabel(ZCS1, inline=1, fontsize=10, colors='black',fmt='%.2f',inline_spacing=1)
ZCS1.collections[0].set_linewidths(1.50)
ZCS1.collections[1].set_linewidths(1.50)
ZCS1.collections[0].set_label(r'$TCW, deg.$')

GCS1 = plt.contour(X, Y, np.transpose(G1), levels=Glevels, colors='blue',linestyles=
    ↪'dashed')
Glabels=plt.clabel(GCS1, inline=1, fontsize=10, colors='blue',fmt='%d',inline_
    ↪spacing=0)
GCS1.collections[0].set_linewidths(1.50)

GCS1.collections[0].set_label(r'$g$'+r'-load')

QCS1 = plt.contour(X, Y, np.transpose(Q1), levels=Qlevels, colors='red',linestyles=
    ↪'dotted',zorder=11)
plt.clabel(QCS1, inline=1, fontsize=10, colors='red',fmt='%d',inline_spacing=0)
QCS1.collections[0].set_linewidths(1.50)
QCS1.collections[1].set_linewidths(1.50)
QCS1.collections[0].set_label(r'$\dot{q}$+', '+r'$W/cm^2$')

HCS1 = plt.contour(X, Y, np.transpose(H1), levels=Hlevels, colors='xkcd:brown',
    ↪linestyles='dashdot')
Hlabels=plt.clabel(HCS1, inline=1, fontsize=10, colors='xkcd:brown',fmt='%d',inline_
    ↪spacing=0)
HCS1.collections[0].set_linewidths(1.75)
HCS1.collections[1].set_linewidths(1.75)
HCS1.collections[0].set_label(r'$Q$+', '+r'$kJ/cm^2$')

for l in Hlabels:
    l.set_rotation(-90)
for l in Glabels:
    l.set_rotation(-90)

#SCS1 = plt.contour(X, Y, transpose(S1), levels=Slevels, colors='cyan')
#plt.clabel(SCS1, inline=1, fontsize=10, colors='xkcd:neon green',fmt='%.1f',inline_
    ↪spacing=1)
#SCS1.collections[0].set_linewidths(1.75)
#SCS1.collections[0].set_label(r'$Peak$'+r' '+'r'$stag. pressure,atm$')

plt.xlabel("Arrival "+r'$V_\infty$'+r', km/s' ,fontsize=10)
plt.ylabel(r'$\beta_2$'+r' / '+r'$\beta_1$' ,fontsize=10)

```

(continues on next page)

(continued from previous page)

```

plt.xticks(fontsize=10)
plt.yticks(np.array([ 1, 20, 40, 60, 80, 100]), fontsize=10)
ax = plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
plt.legend(loc='upper right', fontsize=8)

dat0 = ZCS1.allsegs[1][0]
x1,y1=dat0[:,0],dat0[:,1]
F1 = interpolate.interp1d(x1, y1, kind='linear', fill_value='extrapolate', bounds_
↪error=False)

dat1 = QCS1.allsegs[0][0]
x2,y2=dat1[:,0],dat1[:,1]
F2 = interpolate.interp1d(x2, y2, kind='linear', fill_value='extrapolate', bounds_
↪error=False)

dat0a = ZCS1.allsegs[0][0]
xla,yla=dat0a[:,0],dat0a[:,1]
Fla = interpolate.interp1d(xla, yla, kind='linear', fill_value='extrapolate', bounds_
↪error=False)

x3 = np.linspace(0,30,301)
y3 = F1(x3)
y4 = F2(x3)
y4a = Fla(x3)

y8 = np.minimum(y3,y4)

plt.fill_between(x3, y3, y4, where=y3<=y4, color='xkcd:neon green')
plt.fill_between(x3, y4a, y8, where=y4a<=y8, color='xkcd:bright yellow')

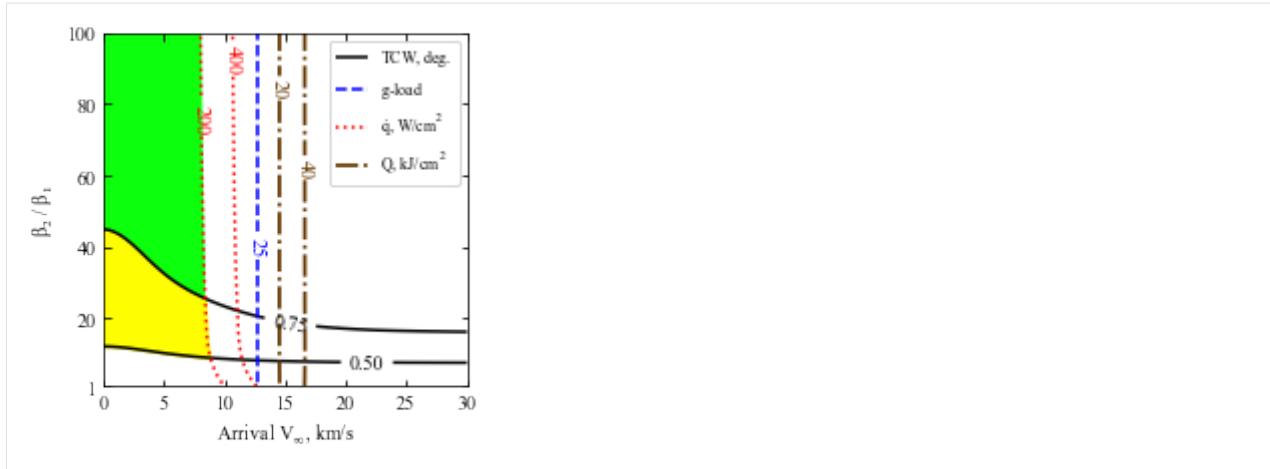
plt.xlim([0.0,30.0])
plt.ylim([1.0,100])

plt.savefig('../plots/giri jaYe2019c-fig16.png',bbox_inches='tight')
plt.savefig('../plots/giri jaYe2019c-fig16.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/giri jaYe2019c-fig16.eps', dpi=300,bbox_inches='tight')

plt.show()

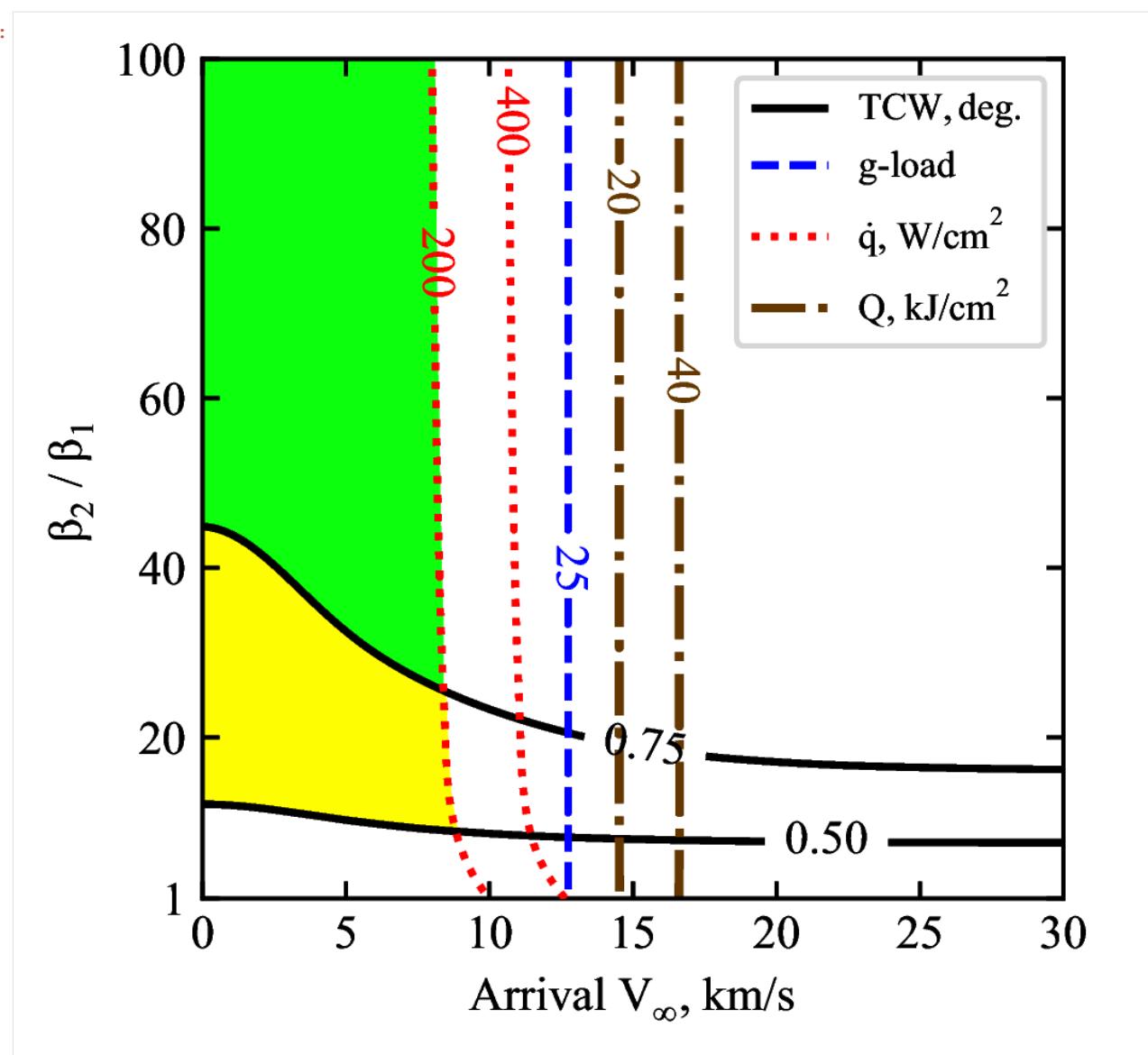
/home/athul/anaconda3/lib/python3.7/site-packages/scipy/interpolate/interpolate.py:
↪609: RuntimeWarning: divide by zero encountered in true_divide
    slope = (y_hi - y_lo) / (x_hi - x_lo)[:, None]
The PostScript backend does not support transparency; partially transparent artists_
↪will be rendered opaque.
The PostScript backend does not support transparency; partially transparent artists_
↪will be rendered opaque.
The PostScript backend does not support transparency; partially transparent artists_
↪will be rendered opaque.
The PostScript backend does not support transparency; partially transparent artists_
↪will be rendered opaque.

```



```
[10]: from IPython.display import Image
Image(filename='../plots/girijaYe2020c-higher-res.png', width=600)
```

[10]:



The plots are now saved in plots/girijaYe2019c.

Congratulations! You could now create the results for referenced journal article in less than a day. It took nearly a year for the authors to put everything together to create these results!

4.9 Example - 09 - Uranus Aerocapture

In this example, we will create combined interplanetary and aerocapture feasibility charts for Uranus.

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle

import numpy as np
from scipy import interpolate
import pandas as pd
```

(continues on next page)

(continued from previous page)

```
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.patches import Polygon
import os
```

```
[3]: # Create a planet object
planet=Planet("URANUS")

# Load an nominal atmospheric profile with height, temp, pressure, density data
planet.loadAtmosphereModel('../atmdata/Uranus/uranus-ames.dat', 0 , 1 , 2, 3)

vinf_kms_array = np.linspace( 0.0,      30.0,    11)
LD_array       = np.linspace( 0.0,      1.0 ,    11)

#vinf_kms_array = np.linspace( 0.0,      30.0,    2)
#LD_array       = np.linspace( 0.0,      1.0 ,    2)
```

```
[4]: os.makedirs('../data/girijaSaikia2019a')
runID = '20DAYBC200FINEGRID1'

num_total      = len(vinf_kms_array)*len(LD_array)
count = 1

v0_kms_array   = np.zeros(len(vinf_kms_array))
v0_kms_array[:] = np.sqrt(1.0*(vinf_kms_array[:]*1E3)**2.0 +\
                          2*np.ones(len(vinf_kms_array))*\
                          planet.GM/(planet.RP+1500.0*1.0E3))/1.0E3

overShootLimit_array = np.zeros((len(v0_kms_array),len(LD_array)))
underShootLimit_array = np.zeros((len(v0_kms_array),len(LD_array)))
exitflag_os_array   = np.zeros((len(v0_kms_array),len(LD_array)))
exitflag_us_array   = np.zeros((len(v0_kms_array),len(LD_array)))
TCW_array          = np.zeros((len(v0_kms_array),len(LD_array)))
```

```
[5]: for i in range(0,len(v0_kms_array)):
    for j in range(0,len(LD_array)):
        vehicle=Vehicle('Oberon', 1000.0, 200.0, LD_array[j], 3.1416, 0.0, 1.00,_
        ↪planet)
        vehicle.setInitialState(1500.0,0.0,0.0,v0_kms_array[i],0.0,-4.5,0.0,0.0)
        vehicle.setSolverParams(1E-6)
        overShootLimit_array[i,j], exitflag_os_array[i,j] = vehicle.
        ↪findOverShootLimit (2400.0, 0.1, -80.0, -4.0, 1E-10, 1491329.10)
        underShootLimit_array[i,j], exitflag_us_array[i,j] = vehicle.
        ↪findUnderShootLimit(2400.0, 0.1, -80.0, -4.0, 1E-10, 1491329.10)

        TCW_array[i,j] = overShootLimit_array[i,j] - underShootLimit_array[i,j]

        print("Run #"+str(count)+" of "+ str(num_total)+": Arrival V_infty:
        ↪"+str(vinf_kms_array[i])+" km/s+", L/D:"+str(LD_array[j]) + " OSL:
        ↪"+str(overShootLimit_array[i,j])+" USL: "+str(underShootLimit_array[i,j])+", TCW:
        ↪"+str(TCW_array[i,j])+" EFOS: "+str(exitflag_os_array[i,j])+ " EFUS: "+str(exitflag_
        ↪us_array[i,j]))
        count = count +1
```

(continues on next page)

(continued from previous page)

```

np.savetxt('..../data/girijaSaikia2019a/' + runID + 'vinf_kms_array.txt', vinf_kms_array)
np.savetxt('..../data/girijaSaikia2019a/' + runID + 'v0_kms_array.txt', v0_kms_array)
np.savetxt('..../data/girijaSaikia2019a/' + runID + 'LD_array.txt', LD_array)
np.savetxt('..../data/girijaSaikia2019a/' + runID + 'overShootLimit_array.txt',
           ↪ overShootLimit_array)
np.savetxt('..../data/girijaSaikia2019a/' + runID + 'exitflag_os_array.txt', exitflag_os_
           ↪ array)
np.savetxt('..../data/girijaSaikia2019a/' + runID + 'undershootLimit_array.txt',
           ↪ underShootLimit_array)
np.savetxt('..../data/girijaSaikia2019a/' + runID + 'exitflag_us_array.txt', exitflag_us_
           ↪ array)
np.savetxt('..../data/girijaSaikia2019a/' + runID + 'TCW_array.txt', TCW_array)

Run #1 of 4: Arrival V_infny: 0.0 km/s, L/D:0.0 OSL: -8.036447100388614 USL: -8.
           ↪ 036447100388614, TCW: 0.0 EFOS: 1.0 EFUS: 1.0
Run #2 of 4: Arrival V_infny: 0.0 km/s, L/D:1.0 OSL: -7.976950472933822 USL: -8.
           ↪ 093977107666433, TCW: 0.11702663473261055 EFOS: 1.0 EFUS: 1.0
Run #3 of 4: Arrival V_infny: 30.0 km/s, L/D:0.0 OSL: -14.479022156530846 USL: -14.
           ↪ 479022156530846, TCW: 0.0 EFOS: 1.0 EFUS: 1.0
Run #4 of 4: Arrival V_infny: 30.0 km/s, L/D:1.0 OSL: -13.721204087287333 USL: -21.
           ↪ 917360380572063, TCW: 8.19615629328473 EFOS: 1.0 EFUS: 1.0

```

```

[6]: acc_net_g_max_array      = np.zeros((len(v0_kms_array), len(LD_array)))
stag_pres_atm_max_array    = np.zeros((len(v0_kms_array), len(LD_array)))
q_stag_total_max_array    = np.zeros((len(v0_kms_array), len(LD_array)))
heatload_max_array         = np.zeros((len(v0_kms_array), len(LD_array)))

for i in range(0, len(v0_kms_array)):
    for j in range(0, len(LD_array)):
        vehicle=Vehicle('Oberon', 1000.0, 200.0, LD_array[j], 3.1416, 0.0, 1.00,
                         ↪ planet)
        vehicle.setInitialState(1500.0, 0.0, 0.0, v0_kms_array[i], 0.0, overShootLimit_
                         ↪ array[i,j], 0.0, 0.0)
        vehicle.setSolverParams(1E-6)

        vehicle.propogateEntry(2400.0, 0.1, 180.0)

        # Extract and save variables to plot
        t_min_os          = vehicle.t_minc
        h_km_os           = vehicle.h_kmc
        acc_net_g_os      = vehicle.acc_net_g
        q_stag_con_os     = vehicle.q_stag_con
        q_stag_rad_os     = vehicle.q_stag_rad
        rc_os              = vehicle.rc
        vc_os              = vehicle.vc
        stag_pres_atm_os  = vehicle.computeStagPres(rc_os, vc_os) / (1.01325E5)
        heatload_os        = vehicle.heatload

        vehicle=Vehicle('Oberon', 1000.0, 200.0, LD_array[j], 3.1416, 0.0, 1.00,
                         ↪ planet)
        vehicle.setInitialState(1500.0, 0.0, 0.0, v0_kms_array[i], 0.0, underShootLimit_
                         ↪ array[i,j], 0.0, 0.0)
        vehicle.setSolverParams(1E-6)

        vehicle.propogateEntry(2400.0, 0.1, 0.0)

```

(continues on next page)

(continued from previous page)

```

# Extract and save variable to plot
t_min_us      = vehicle.t_minc
h_km_us       = vehicle.h_kmc
acc_net_g_us  = vehicle.acc_net_g
q_stag_con_us = vehicle.q_stag_con
q_stag_rad_us = vehicle.q_stag_rad
rc_us          = vehicle.rc
vc_us          = vehicle.vc
stag_pres_atm_us = vehicle.computeStagPres(rc_us, vc_us) / (1.01325E5)
heatload_us    = vehicle.heatload

q_stag_total_os  = q_stag_con_os + q_stag_rad_os
q_stag_total_us  = q_stag_con_us + q_stag_rad_us

acc_net_g_max_array[i,j]      = max(max(acc_net_g_os),max(acc_net_g_us))
stag_pres_atm_max_array[i,j]   = max(max(stag_pres_atm_os),max(stag_pres_atm_
os))
q_stag_total_max_array[i,j]   = max(max(q_stag_total_os),max(q_stag_total_us))
heatload_max_array[i,j]        = max(max(heatload_os),max(heatload_us))

print("V_infty: "+str(vinf_kms_array[i])+" km/s", L/D: "+str(LD_array[j])+"_
G_MAX: "+str(acc_net_g_max_array[i,j])+" QDOT_MAX: "+str(q_stag_total_max_array[i,
j])+" J_MAX: "+str(heatload_max_array[i,j])+" STAG. PRES: "+str(stag_pres_atm_max_
array[i,j])))

np.savetxt('..../data/girijaSaikia2019a/' + runID + 'acc_net_g_max_array.txt', acc_net_g_max_
array)
np.savetxt('..../data/girijaSaikia2019a/' + runID + 'stag_pres_atm_max_array.txt', stag_pres_-
atm_max_array)
np.savetxt('..../data/girijaSaikia2019a/' + runID + 'q_stag_total_max_array.txt', q_stag_-
total_max_array)
np.savetxt('..../data/girijaSaikia2019a/' + runID + 'heatload_max_array.txt', heatload_max_-
array)

V_infty: 0.0 km/s, L/D: 0.0 G_MAX: 0.05394060815860674 QDOT_MAX: 74.06491463780053 J_
MAX: 41374.247055160056 STAG. PRES: 0.0010473053416354748
V_infty: 0.0 km/s, L/D: 1.0 G_MAX: 0.08366575834728353 QDOT_MAX: 77.03205462674941 J_
MAX: 42091.96352516038 STAG. PRES: 0.0010461973471433495
V_infty: 30.0 km/s, L/D: 0.0 G_MAX: 12.243125885881222 QDOT_MAX: 28618.290301651356 J_
MAX: 2374794.7913468257 STAG. PRES: 0.23721295548140664
V_infty: 30.0 km/s, L/D: 1.0 G_MAX: 127.05514958454569 QDOT_MAX: 51744.90580336135 J_
MAX: 3659758.2791496087 STAG. PRES: 0.0539571845570865

```

```

[7]: N1 = pd.read_excel('..../interplanetary-data/Uranus/U1.xlsx', sheet_name='Sheet1')
N2 = pd.read_excel('..../interplanetary-data/Uranus/U2.xlsx', sheet_name='Sheet1')
N3 = pd.read_excel('..../interplanetary-data/Uranus/U3.xlsx', sheet_name='Sheet1')
N4 = pd.read_excel('..../interplanetary-data/Uranus/U4.xlsx', sheet_name='Sheet1')
N5 = pd.read_excel('..../interplanetary-data/Uranus/Uranus.xlsx', sheet_name='Uranus')

TOF1 = N1['Atof'].values
TOF2 = N2['Atof'].values
TOF3 = N3['Atof'].values
TOF4 = N4['Atof'].values
TOF5 = N5['TOF'].values*365.0

VINF1 = N1['Avinf'].values

```

(continues on next page)

(continued from previous page)

```

VINF2 = N2['Avinf'].values
VINF3 = N3['Avinf'].values
VINF4 = N4['Avinf'].values
VINF5 = N5['ArrVinf_mag'].values

LC31 = N1['LC3'].values
LC32 = N2['LC3'].values
LC33 = N3['LC3'].values
LC34 = N4['LC3'].values
LC35 = N5['C3'].values

TOF    = np.concatenate((TOF1, TOF2, TOF3, TOF4), axis=0)
TOF_y = TOF / 365.0

VINF_kms = np.concatenate((VINF1, VINF2, VINF3, VINF4), axis=0)

LC3 = np.concatenate((LC31, LC32, LC33, LC34), axis=0)

# plt.axhline(y=13.5, linewidth=1, linestyle='dotted', color='black', zorder=0)
# plt.axvline(x=13.0, linewidth=1, linestyle='dotted', color='black', zorder=0)

fig, axes = plt.subplots(1, 2, figsize = (6.5, 3.5))
fig.tight_layout()
plt.subplots_adjust(wspace=0.30)
plt.rc('font', family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

a0 = axes[0].scatter(TOF5 / 365.0, VINF5, c=LC35, cmap='jet', vmin=0, vmax=LC35.max(),  

                     zorder=10, s=5.0)
a1 = axes[0].scatter(TOF_y, VINF_kms, c=LC3, cmap='jet', vmin=0, vmax=LC35.max(),  

                     zorder=11, s=5.0)
cbar = fig.colorbar(a0, ax=axes[0])
cbar.ax.tick_params(labelsize=10)
cbar.set_label(r'$C_3, km^2/s^2$', labelpad=-27, y=1.10, rotation=0, fontsize=10)
cbar.ax.tick_params(axis='y', direction='in')

axes[0].tick_params(direction='in')
axes[0].yaxis.set_ticks_position('both')
axes[0].xaxis.set_ticks_position('both')

axes[0].set_xlabel("Time of flight (TOF), years", fontsize=10)
axes[0].set_ylabel("Arrival "+r'$V_{\infty}$', km/s, fontsize=10)

axes[0].set_yticks(np.arange(5, 26, step=5))
axes[0].set_xticks(np.arange(6, 13, step=2))

axes[0].tick_params(axis='x', labelsize=10)
axes[0].tick_params(axis='y', labelsize=10)

x = np.loadtxt('../data/girijaSaikia2019a/' + runID + 'vinf_kms_array.txt')
y = np.loadtxt('../data/girijaSaikia2019a/' + runID + 'LD_array.txt')

z1 = np.loadtxt('../data/girijaSaikia2019a/' + runID + 'TCW_array.txt')

```

(continues on next page)

(continued from previous page)

```

G1 = np.loadtxt('../data/girijaSaikia2019a/' + runID + 'acc_net_g_max_array.txt')
Q1 = np.loadtxt('../data/girijaSaikia2019a/' + runID + 'q_stag_total_max_array.txt')
H1 = np.loadtxt('../data/girijaSaikia2019a/' + runID + 'heatload_max_array.txt')
S1 = np.loadtxt('../data/girijaSaikia2019a/' + runID + 'stag_pres_atm_max_array.txt')

f1 = interpolate.interp2d(x, y, np.transpose(Z1), kind='cubic')
g1 = interpolate.interp2d(x, y, np.transpose(G1), kind='cubic')
q1 = interpolate.interp2d(x, y, np.transpose(Q1), kind='cubic')
h1 = interpolate.interp2d(x, y, np.transpose(H1), kind='cubic')
s1 = interpolate.interp2d(x, y, np.transpose(S1), kind='cubic')

x_new = np.linspace( 0.0,    30,   110)
y_new = np.linspace( 0.0,    1.0 ,110)
z_new = np.zeros((len(x_new),len(y_new)))

z1_new = np.zeros((len(x_new),len(y_new)))
g1_new = np.zeros((len(x_new),len(y_new)))
q1_new = np.zeros((len(x_new),len(y_new)))
h1_new = np.zeros((len(x_new),len(y_new)))
s1_new = np.zeros((len(x_new),len(y_new)))

for i in range(0,len(x_new)):
    for j in range(0,len(y_new)):

        z1_new[i,j] = f1(x_new[i],y_new[j])
        g1_new[i,j] = g1(x_new[i],y_new[j])
        q1_new[i,j] = q1(x_new[i],y_new[j])
        h1_new[i,j] = h1(x_new[i],y_new[j])
        s1_new[i,j] = s1(x_new[i],y_new[j])

Z1 = z1_new
G1 = g1_new
Q1 = q1_new
S1 = s1_new
H1 = h1_new/1000.0

X, Y = np.meshgrid(x_new, y_new)

Zlevels = np.array([0.5,1.0,1.5,2.0])
Glevels = np.array([30.0])
Qlevels = np.array([8000.0])
Hlevels = np.array([700.0])
Slevels = np.array([1.0])

ZCS1 = axes[1].contour(X, Y, np.transpose(Z1), levels=Zlevels, colors='black')

plt.clabel(ZCS1, inline=1, fontsize=10, colors='black', fmt='%.1f', inline_spacing=1)
ZCS1.collections[0].set_linewidths(1.0)
ZCS1.collections[1].set_linewidths(1.0)
ZCS1.collections[2].set_linewidths(1.0)
ZCS1.collections[3].set_linewidths(1.0)
ZCS1.collections[0].set_label(r'$TCW, deg$')

```

(continues on next page)

(continued from previous page)

```

GCS1 = axes[1].contour(X, Y, np.transpose(G1), levels=Glevels, colors='blue',  

                       linestyles='dashed')

plt.clabel(GCS1, inline=1, fontsize=10, colors='blue', fmt=' %d')
GCS1.collections[0].set_linewidths(1.0)
GCS1.collections[0].set_label(r'$Peak$'+ r' '+r'$g$'+r'-load')

QCS1 = axes[1].contour(X, Y, np.transpose(Q1), levels=Qlevels, colors='red',  

                       linestyles='dotted')

plt.clabel(QCS1, inline=1, fontsize=10, colors='red', fmt=' %d')
QCS1.collections[0].set_linewidths(1.0)
QCS1.collections[0].set_label(r'$Peak$'+r' '+r'$\dot{q}$+', '+r'$W/cm^2$')

HCS1 = axes[1].contour(X, Y, np.transpose(H1), levels=Hlevels, colors='magenta',  

                       linestyles='dashdot')
plt.xlim([0.0,30.0])
plt.clabel(HCS1, inline=1, fontsize=10, colors='magenta', fmt=' %d', inline_spacing=1)
HCS1.collections[0].set_linewidths(1.0)
HCS1.collections[0].set_label(r'$Heat$'+r' '+r'$ load, kJ/cm^2$')

SCS1 = axes[1].contour(X, Y, np.transpose(S1), levels=Slevels, colors='xkcd:emerald',  

                       linestyles='dashed')
plt.xlim([0.0,30.0])
plt.clabel(SCS1, inline=1, fontsize=10, colors='xkcd:emerald green', fmt='%.1f', inline_  

                           spacing=1)
SCS1.collections[0].set_linewidths(1.0)
SCS1.collections[0].set_label(r'$Peak$'+r' '+r'$ stag. pres., bar$')

for c in SCS1.collections:  

    c.set_dashes([(0.5, (7.0, 2.0))])

axes[1].tick_params(direction='in')
axes[1].yaxis.set_ticks_position('both')
axes[1].xaxis.set_ticks_position('both')

#axes[1].set(xlabel="Exam score-1", ylabel="Exam score-2")

axes[1].set_xlabel("Arrival "+r'$V_{\infty}$'+r', km/s', fontsize=10)
axes[1].set_ylabel("L/D", fontsize=10)

axes[1].tick_params(axis='x', labelsize=10)
axes[1].tick_params(axis='y', labelsize=10)

legend1 = axes[1].legend(loc='lower left', fontsize=8, frameon=False)

dat0 = ZCS1.allsegs[3][0]
x1,y1=dat0[:,0],dat0[:,1]
F1 = interpolate.interp1d(x1, y1, kind='linear', fill_value='extrapolate', bounds_  

                           error=False)

```

(continues on next page)

(continued from previous page)

```
dat1 = GCS1.allsegs[0][0]
x2,y2=dat1[:,0],dat1[:,1]
F2 = interpolate.interp1d(x2, y2, kind='linear', fill_value='extrapolate', bounds_
↪error=False)

dat2 = HCS1.allsegs[0][0]
x3,y3= dat2[:,0],dat2[:,1]
F3 = interpolate.interp1d(x3, y3, kind='linear', fill_value='extrapolate', bounds_
↪error=False)

x4 = np.linspace(0,30,301)
y4 = F1(x4)
y5 = F2(x4)
y6 = F3(x4)

y7 = np.minimum(y5,y6)

plt.fill_between(x4, y4, y7, where=y4<=y7,color='xkcd:neon green')

dat0 = ZCS1.allsegs[2][0]
x1,y1=dat0[:,0],dat0[:,1]
F1 = interpolate.interp1d(x1, y1, kind='linear', fill_value='extrapolate', bounds_
↪error=False)

dat1 = ZCS1.allsegs[3][0]
x2,y2=dat1[:,0],dat1[:,1]
F2 = interpolate.interp1d(x2, y2, kind='linear', fill_value='extrapolate', bounds_
↪error=False)

dat2 = HCS1.allsegs[0][0]
x3,y3= dat2[:,0],dat2[:,1]
F3 = interpolate.interp1d(x3, y3, kind='linear', fill_value='extrapolate', bounds_
↪error=False)

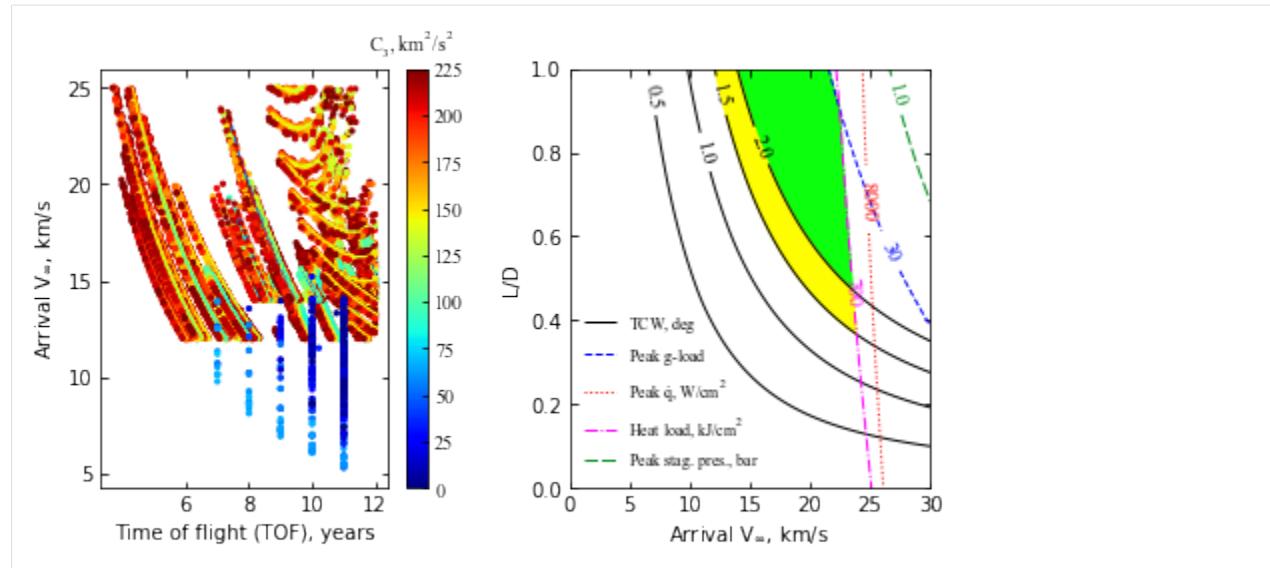
x4 = np.linspace(0,30,301)
y4 = F1(x4)
y5 = F2(x4)
y6 = F3(x4)

y7 = np.minimum(y5,y6)

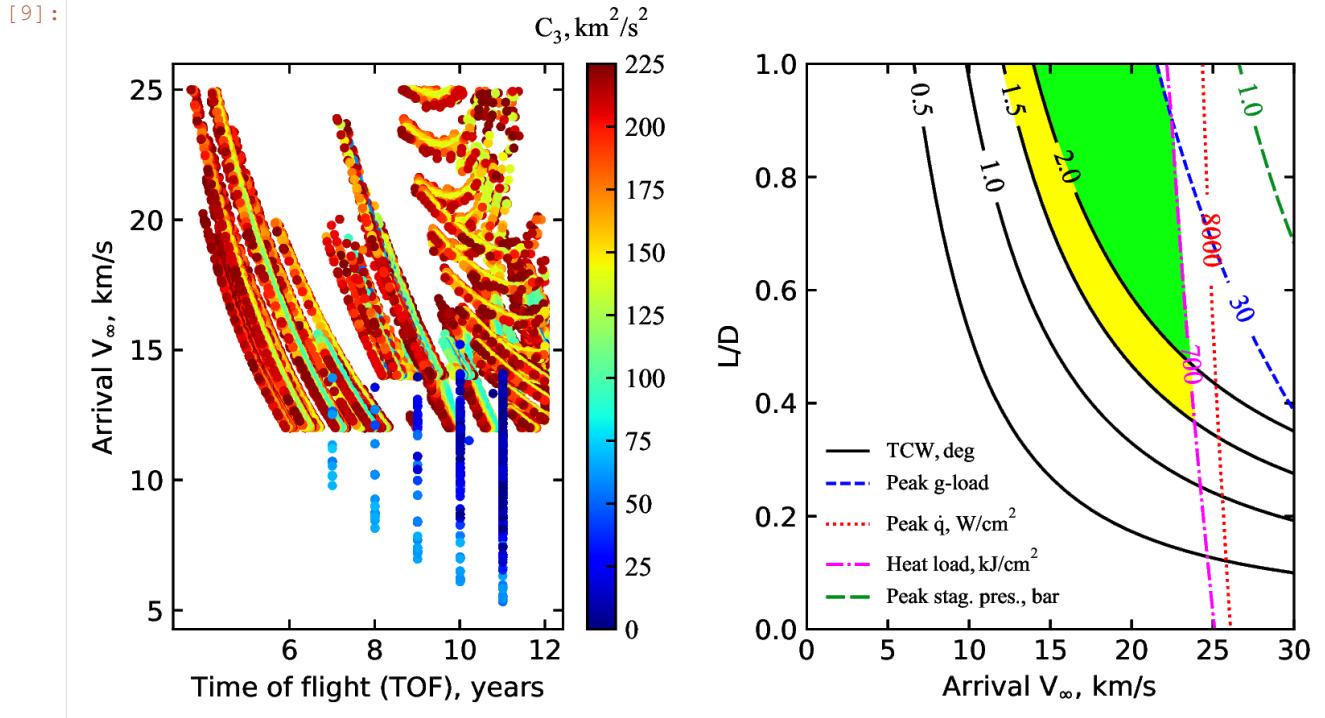
plt.fill_between(x4, y4, y7, where=y4<=y7,color='xkcd:bright yellow')
plt.xlim([0.0,30.0])
plt.ylim([0.0,1.0])

plt.savefig('../plots/girijaSaikia2019a.png',bbox_inches='tight')
plt.savefig('../plots/girijaSaikia2019a.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/girijaSaikia2019a.eps', dpi=300,bbox_inches='tight')

plt.show()
```



```
[9]: from IPython.display import Image
Image(filename='../plots/girijaSaikia2019a-high-res.png', width=800)
```



4.10 Example - 10 - Neptune Aerocapture - Part 1

In this example, we will create combined interplanetary and aerocapture feasibility charts for Neptune.

We re-create the feasibility chart from the paper “Girija, Saikia, Longuski et al. Feasibility and Performance Analysis of Neptune Aerocapture Using Heritage Blunt-Body Aeroshells, Journal of Spacecraft and Rockets, June, 2020, In press. DOI: 10.2514/1.A34719

```
[3]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle

import numpy as np
from scipy import interpolate
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.patches import Polygon
import os

[4]: # Create a planet object
planet=Planet("NEPTUNE")

# Load an nominal atmospheric profile with height, temp, pressure, density data
planet.loadAtmosphereModel('../atmdata/Neptune/neptune-gram-avg.dat', 0 , 7 , 6 , 5 , \
                           heightInKmFlag=True)

vinf_kms_array = np.linspace( 0.0,    30.0,   11)
LD_array       = np.linspace( 0.0,     1.0 ,  11)

#vinf_kms_array = np.linspace( 0.0,    30.0,   2)
#LD_array       = np.linspace( 0.0,     1.0 ,  2)

[5]: os.makedirs('../data/girijaSaikia2019b')
runID = '20DAY'

num_total      = len(vinf_kms_array)*len(LD_array)
count = 1

v0_kms_array   = np.zeros(len(vinf_kms_array))
v0_kms_array[:] = np.sqrt(1.0*(vinf_kms_array[:]*1E3)**2.0 + \
                          2*np.ones(len(vinf_kms_array))*\
                          planet.GM/(planet.RP+1000.0*1.0E3))/1.0E3

overShootLimit_array = np.zeros((len(v0_kms_array),len(LD_array)))
underShootLimit_array = np.zeros((len(v0_kms_array),len(LD_array)))
exitflag_os_array   = np.zeros((len(v0_kms_array),len(LD_array)))
exitflag_us_array   = np.zeros((len(v0_kms_array),len(LD_array)))
TCW_array          = np.zeros((len(v0_kms_array),len(LD_array)))

[6]: for i in range(0,len(v0_kms_array)):
    for j in range(0,len(LD_array)):
        vehicle=Vehicle('Trident', 1000.0, 200.0, LD_array[j], 3.1416, 0.0, 1.00,_
                         planet)
        vehicle.setInitialState(1000.0,0.0,0.0,v0_kms_array[i],0.0,-4.5,0.0,0.0)
        vehicle.setSolverParams(1E-6)
        overShootLimit_array[i,j], exitflag_os_array[i,j] = vehicle.
        ↪findOverShootLimit (2400.0, 0.1, -80.0, -4.0, 1E-10, 1553575.10)
        underShootLimit_array[i,j], exitflag_us_array[i,j] = vehicle.
        ↪findUnderShootLimit(2400.0, 0.1, -80.0, -4.0, 1E-10, 1553575.10)

        TCW_array[i,j] = overShootLimit_array[i,j] - underShootLimit_array[i,j]

        print("Run #"+str(count)+" of "+ str(num_total)+": Arrival V_infty:
        ↪"+str(vinf_kms_array[i])+" km/s"+", T/D:"+str(LD_array[j]) + " OSL:
        ↪"+str(overShootLimit_array[i,j])+" USL: "+str(underShootLimit_array[i,j])+", TCW:
        ↪"+str(TCW_array[i,j])+" EFOS: "+str(exitflag_os_array[i,j])+" EFUS: "+str(exitflag_"
        ↪64us_array[i,j]))
```

(continued from previous page)

```

count = count +1

np.savetxt('..../data/girijaSaikia2019b/' +runID+'vinf_kms_array.txt',vinf_kms_array)
np.savetxt('..../data/girijaSaikia2019b/' +runID+'v0_kms_array.txt',v0_kms_array)
np.savetxt('..../data/girijaSaikia2019b/' +runID+'LD_array.txt',LD_array)
np.savetxt('..../data/girijaSaikia2019b/' +runID+'overShootLimit_array.txt',
           ↵overShootLimit_array)
np.savetxt('..../data/girijaSaikia2019b/' +runID+'exitflag_os_array.txt',exitflag_os_
           ↵array)
np.savetxt('..../data/girijaSaikia2019b/' +runID+'undershootLimit_array.txt',
           ↵underShootLimit_array)
np.savetxt('..../data/girijaSaikia2019b/' +runID+'exitflag_us_array.txt',exitflag_us_
           ↵array)
np.savetxt('..../data/girijaSaikia2019b/' +runID+'TCW_array.txt',TCW_array)

Run #1 of 4: Arrival V_infny: 0.0 km/s, L/D:0.0 OSL: -10.88548090497352 USL: -10.
           ↵88548090497352, TCW: 0.0 EFOS: 1.0 EFUS: 1.0
Run #2 of 4: Arrival V_infny: 0.0 km/s, L/D:1.0 OSL: -10.83489821571493 USL: -10.
           ↵93779864834869, TCW: 0.10290043263375992 EFOS: 1.0 EFUS: 1.0
Run #3 of 4: Arrival V_infny: 30.0 km/s, L/D:0.0 OSL: -14.18370732049516 USL: -14.
           ↵18370732049516, TCW: 0.0 EFOS: 1.0 EFUS: 1.0
Run #4 of 4: Arrival V_infny: 30.0 km/s, L/D:1.0 OSL: -13.15751718847605 USL: -20.
           ↵524218087299232, TCW: 7.366700898823183 EFOS: 1.0 EFUS: 1.0

```

```

[7]: acc_net_g_max_array      = np.zeros((len(v0_kms_array),len(LD_array)))
stag_pres_atm_max_array    = np.zeros((len(v0_kms_array),len(LD_array)))
q_stag_total_max_array    = np.zeros((len(v0_kms_array),len(LD_array)))
heatload_max_array         = np.zeros((len(v0_kms_array),len(LD_array)))

for i in range(0,len(v0_kms_array)):
    for j in range(0,len(LD_array)):
        vehicle=Vehicle('Trident', 1000.0, 200.0, LD_array[j], 3.1416, 0.0, 1.00,
                         ↵planet)
        vehicle.setInitialState(1000.0,0.0,0.0,v0_kms_array[i],0.0,overShootLimit_
        ↵array[i,j],0.0,0.0)
        vehicle.setSolverParams(1E-6)

        vehicle.propogateEntry(2400.0, 0.1, 180.0)

        # Extract and save variables to plot
        t_min_os          = vehicle.t_minc
        h_km_os           = vehicle.h_kmc
        acc_net_g_os      = vehicle.acc_net_g
        q_stag_con_os     = vehicle.q_stag_con
        q_stag_rad_os     = vehicle.q_stag_rad
        rc_os              = vehicle.rc
        vc_os              = vehicle.vc
        stag_pres_atm_os  = vehicle.computeStagPres(rc_os,vc_os) / (1.01325E5)
        heatload_os        = vehicle.heatload

        vehicle=Vehicle('Trident', 1000.0, 200.0, LD_array[j], 3.1416, 0.0, 1.00,
                         ↵planet)
        vehicle.setInitialState(1000.0,0.0,0.0,v0_kms_array[i],0.0,underShootLimit_
        ↵array[i,j],0.0,0.0)
        vehicle.setSolverParams(1E-6)

        vehicle.propogateEntry(2400.0, 0.1, 0.0)

```

(continues on next page)

(continued from previous page)

```

# Extract and save variable to plot
t_min_us      = vehicle.t_minc
h_km_us       = vehicle.h_kmc
acc_net_g_us  = vehicle.acc_net_g
q_stag_con_us = vehicle.q_stag_con
q_stag_rad_us = vehicle.q_stag_rad
rc_us          = vehicle.rc
vc_us          = vehicle.vc
stag_pres_atm_us = vehicle.computeStagPres(rc_us, vc_us) / (1.01325E5)
heatload_us    = vehicle.heatload

q_stag_total_os  = q_stag_con_os + q_stag_rad_os
q_stag_total_us  = q_stag_con_us + q_stag_rad_us

acc_net_g_max_array[i,j]      = max(max(acc_net_g_os),max(acc_net_g_us))
stag_pres_atm_max_array[i,j]   = max(max(stag_pres_atm_os),max(stag_pres_atm_
os))
q_stag_total_max_array[i,j]   = max(max(q_stag_total_os),max(q_stag_total_us))
heatload_max_array[i,j]        = max(max(heatload_os),max(heatload_us))

print("V_infty: "+str(vinf_kms_array[i])+" km/s", L/D: "+str(LD_array[j])+"_
G_MAX: "+str(acc_net_g_max_array[i,j])+" QDOT_MAX: "+str(q_stag_total_max_array[i,
j])+" J_MAX: "+str(heatload_max_array[i,j])+" STAG. PRES: "+str(stag_pres_atm_max_
array[i,j]))


np.savetxt('~/data/girijaSaikia2019b/' + runID + 'acc_net_g_max_array.txt', acc_net_g_max_
array)
np.savetxt('~/data/girijaSaikia2019b/' + runID + 'stag_pres_atm_max_array.txt', stag_pres_
atm_max_array)
np.savetxt('~/data/girijaSaikia2019b/' + runID + 'q_stag_total_max_array.txt', q_stag_
total_max_array)
np.savetxt('~/data/girijaSaikia2019b/' + runID + 'heatload_max_array.txt', heatload_max_
array)

V_infty: 0.0 km/s, L/D: 0.0 G_MAX: 0.1330431053594133 QDOT_MAX: 140.91155406501474 J_
MAX: 29874.621801169895 STAG. PRES: 0.002580414807141991
V_infty: 0.0 km/s, L/D: 1.0 G_MAX: 0.19458771430807223 QDOT_MAX: 143.0988625244372 J_
MAX: 30285.689932384335 STAG. PRES: 0.0024914483287594756
V_infty: 30.0 km/s, L/D: 0.0 G_MAX: 17.53516238321169 QDOT_MAX: 50974.88161005046 J_
MAX: 3359148.4906504373 STAG. PRES: 0.3397375503761374
V_infty: 30.0 km/s, L/D: 1.0 G_MAX: 120.73071641252108 QDOT_MAX: 93546.94104353685 J_
MAX: 5072351.332692758 STAG. PRES: 0.08214636492111621

```

```

[8]: N1 = pd.read_excel('~/interplanetary-data/Neptune/N1.xlsx', sheet_name='Sheet1')
N2 = pd.read_excel('~/interplanetary-data/Neptune/N2.xlsx', sheet_name='Sheet1')
N3 = pd.read_excel('~/interplanetary-data/Neptune/N3.xlsx', sheet_name='Sheet1')
N4 = pd.read_excel('~/interplanetary-data/Neptune/N4.xlsx', sheet_name='Sheet1')
N5 = pd.read_excel('~/interplanetary-data/Neptune/Neptune.xlsx', sheet_name='Neptune
')

TOF1 = N1['Atof'].values
TOF2 = N2['Atof'].values
TOF3 = N3['Atof'].values
TOF4 = N4['Atof'].values
TOF5 = N5['TOF'].values*365.0

```

(continues on next page)

(continued from previous page)

```

VINF1 = N1['Avinf'].values
VINF2 = N2['Avinf'].values
VINF3 = N3['Avinf'].values
VINF4 = N4['Avinf'].values
VINF5 = N5['ArrVinf_mag'].values

LC31 = N1['LC3'].values
LC32 = N2['LC3'].values
LC33 = N3['LC3'].values
LC34 = N4['LC3'].values
LC35 = N5['C3'].values

TOF    = np.concatenate((TOF1, TOF2, TOF3, TOF4), axis=0)
TOF_y = TOF / 365.0

VINF_kms = np.concatenate((VINF1, VINF2, VINF3, VINF4), axis=0)

LC3 = np.concatenate((LC31, LC32, LC33, LC34), axis=0)

# plt.axhline(y=13.5, linewidth=1, linestyle='dotted', color='black', zorder=0)
# plt.axvline(x=13.0, linewidth=1, linestyle='dotted', color='black', zorder=0)

fig, axes = plt.subplots(1, 2, figsize = (6.5, 3.5))
fig.tight_layout()
plt.subplots_adjust(wspace=0.30)
plt.rc('font', family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

a0 = axes[0].scatter(TOF5 / 365.0, VINF5, c=LC35, cmap='jet', vmin=0, vmax=LC35.max(), zorder=10, s=5.0)
a1 = axes[0].scatter(TOF_y, VINF_kms, c=LC3, cmap='jet', vmin=0, vmax=LC35.max(), zorder=11, s=5.0)
cbar = fig.colorbar(a0, ax=axes[0])
cbar.ax.tick_params(labelsize=10)
cbar.set_label(r'$C_3, km^2/s^2$', labelpad=-27, y=1.10, rotation=0, fontsize=10)
cbar.ax.tick_params(axis='y', direction='in')

axes[0].tick_params(direction='in')
axes[0].yaxis.set_ticks_position('both')
axes[0].xaxis.set_ticks_position('both')

axes[0].set_xlabel("Time of flight (TOF), years", fontsize=10)
axes[0].set_ylabel("Arrival "+r'$V_{\infty}$', fontsize=10)

axes[0].set_yticks(np.arange(5, 26, step=5))
axes[0].set_xticks(np.arange(6, 13, step=2))

axes[0].tick_params(axis='x', labelsize=10)
axes[0].tick_params(axis='y', labelsize=10)

x = np.loadtxt('../data/girijaSaikia2019b/' + runID + 'vinf_kms_array.txt')
y = np.loadtxt('../data/girijaSaikia2019b/' + runID + 'LD_array.txt')

```

(continues on next page)

(continued from previous page)

```

Z1 = np.loadtxt('../data/girijaSaikia2019b/' + runID + 'TCW_array.txt')
G1 = np.loadtxt('../data/girijaSaikia2019b/' + runID + 'acc_net_g_max_array.txt')
Q1 = np.loadtxt('../data/girijaSaikia2019b/' + runID + 'q_stag_total_max_array.txt')
H1 = np.loadtxt('../data/girijaSaikia2019b/' + runID + 'heatload_max_array.txt')
S1 = np.loadtxt('../data/girijaSaikia2019b/' + runID + 'stag_pres_atm_max_array.txt')

f1 = interpolate.interp2d(x, y, np.transpose(Z1), kind='cubic')
g1 = interpolate.interp2d(x, y, np.transpose(G1), kind='cubic')
q1 = interpolate.interp2d(x, y, np.transpose(Q1), kind='cubic')
h1 = interpolate.interp2d(x, y, np.transpose(H1), kind='cubic')
s1 = interpolate.interp2d(x, y, np.transpose(S1), kind='cubic')

x_new = np.linspace(0.0, 30, 110)
y_new = np.linspace(0.0, 1.0, 110)
z_new = np.zeros((len(x_new), len(y_new)))

z1_new = np.zeros((len(x_new), len(y_new)))
g1_new = np.zeros((len(x_new), len(y_new)))
q1_new = np.zeros((len(x_new), len(y_new)))
h1_new = np.zeros((len(x_new), len(y_new)))
s1_new = np.zeros((len(x_new), len(y_new)))

for i in range(0, len(x_new)):
    for j in range(0, len(y_new)):

        z1_new[i, j] = f1(x_new[i], y_new[j])
        g1_new[i, j] = g1(x_new[i], y_new[j])
        q1_new[i, j] = q1(x_new[i], y_new[j])
        h1_new[i, j] = h1(x_new[i], y_new[j])
        s1_new[i, j] = s1(x_new[i], y_new[j])

Z1 = z1_new
G1 = g1_new
Q1 = q1_new
S1 = s1_new
H1 = h1_new/1000.0

X, Y = np.meshgrid(x_new, y_new)

Zlevels = np.array([0.75, 1.25, 2.0])
Glevels = np.array([30.0])
Qlevels = np.array([8000.0])
Hlevels = np.array([600.0])
Slevels = np.array([1.0])

ZCS1 = axes[1].contour(X, Y, np.transpose(Z1), levels=Zlevels, colors='black')

plt.clabel(ZCS1, inline=1, fontsize=9, colors='black', fmt='%.2f', inline_spacing=1)
ZCS1.collections[0].set_linewidths(1.0)
ZCS1.collections[1].set_linewidths(1.0)
ZCS1.collections[2].set_linewidths(1.0)

```

(continues on next page)

(continued from previous page)

```

ZCS1.collections[0].set_label(r'$TCW, deg$')

GCS1 = axes[1].contour(X, Y, np.transpose(G1), levels=Glevels, colors='blue',  

    linestyles='dashed')

plt.clabel(GCS1, inline=1, fontsize=9, colors='blue', fmt='%d')
GCS1.collections[0].set_linewidths(1.0)
GCS1.collections[0].set_label(r'$Peak$'+ r' '+r'$g$'+r'$-load$')

QCS1 = axes[1].contour(X, Y, np.transpose(Q1), levels=Qlevels, colors='red',  

    linestyles='dotted')

plt.clabel(QCS1, inline=1, fontsize=9, colors='red', fmt='%d')
QCS1.collections[0].set_linewidths(1.0)
QCS1.collections[0].set_label(r'$Peak$'+r' '+r'$\dot{q}$'+', '+r'$W/cm^2$')

HCS1 = axes[1].contour(X, Y, np.transpose(H1), levels=Hlevels, colors='magenta',  

    linestyles='dashdot')
plt.xlim([0.0,30.0])
plt.clabel(HCS1, inline=1, fontsize=9, colors='magenta', fmt='%d', inline_spacing=1)
HCS1.collections[0].set_linewidths(1.0)
HCS1.collections[0].set_label(r'$Heat$'+ ' '+r'$ load, kJ/cm^2$')

SCS1 = axes[1].contour(X, Y, np.transpose(S1), levels=Slevels, colors='xkcd:emerald',  

    green, linestyles='dashed')
plt.xlim([0.0,30.0])
plt.clabel(SCS1, inline=1, fontsize=9, colors='xkcd:emerald green', fmt='%.1f', inline_  

    spacing=1)
SCS1.collections[0].set_linewidths(1.0)
SCS1.collections[0].set_label(r'$Peak$'+r' '+'stag. pres., bar')

for c in SCS1.collections:
    c.set_dashes([(0.5, (7.0, 2.0))])

axes[1].tick_params(direction='in')
axes[1].yaxis.set_ticks_position('both')
axes[1].xaxis.set_ticks_position('both')

#axes[1].set_xlabel("Exam score-1", ylabel="Exam score-2")

axes[1].set_xlabel("Arrival "+r'$V_{\infty}$'+r', km/s', fontsize=10)
axes[1].set_ylabel("L/D", fontsize=10)

axes[1].tick_params(axis='x', labelsize=10)
axes[1].tick_params(axis='y', labelsize=10)

legend1 = axes[1].legend(loc='lower left', fontsize=8, frameon=False)

plt.scatter(19.96, 0.40, s=40, c='k', marker='o', zorder=25)

```

(continues on next page)

(continued from previous page)

```

dat0 = ZCS1.allsegs[2][0]
x1,y1=dat0[:,0],dat0[:,1]
F1 = interpolate.interp1d(x1, y1, kind='linear', fill_value='extrapolate', bounds_
↪error=False)

dat1 = GCS1.allsegs[0][0]
x2,y2=dat1[:,0],dat1[:,1]
F2 = interpolate.interp1d(x2, y2, kind='linear', fill_value='extrapolate', bounds_
↪error=False)

dat2 = HCS1.allsegs[0][0]
x3,y3= dat2[:,0],dat2[:,1]
F3 = interpolate.interp1d(x3, y3, kind='linear', fill_value='extrapolate', bounds_
↪error=False)

x4 = np.linspace(0,30,301)
y4 = F1(x4)
y5 = F2(x4)
y6 = F3(x4)

y7 = np.minimum(y5,y6)

plt.fill_between(x4, y4, y7, where=y4<=y7,color='xkcd:neon green')

dat0 = ZCS1.allsegs[1][0]
x1,y1=dat0[:,0],dat0[:,1]
F1 = interpolate.interp1d(x1, y1, kind='linear', fill_value='extrapolate', bounds_
↪error=False)

dat1 = ZCS1.allsegs[2][0]
x2,y2=dat1[:,0],dat1[:,1]
F2 = interpolate.interp1d(x2, y2, kind='linear', fill_value='extrapolate', bounds_
↪error=False)

dat2 = HCS1.allsegs[0][0]
x3,y3= dat2[:,0],dat2[:,1]
F3 = interpolate.interp1d(x3, y3, kind='linear', fill_value='extrapolate', bounds_
↪error=False)

x4 = np.linspace(0,30,301)
y4 = F1(x4)
y5 = F2(x4)
y6 = F3(x4)

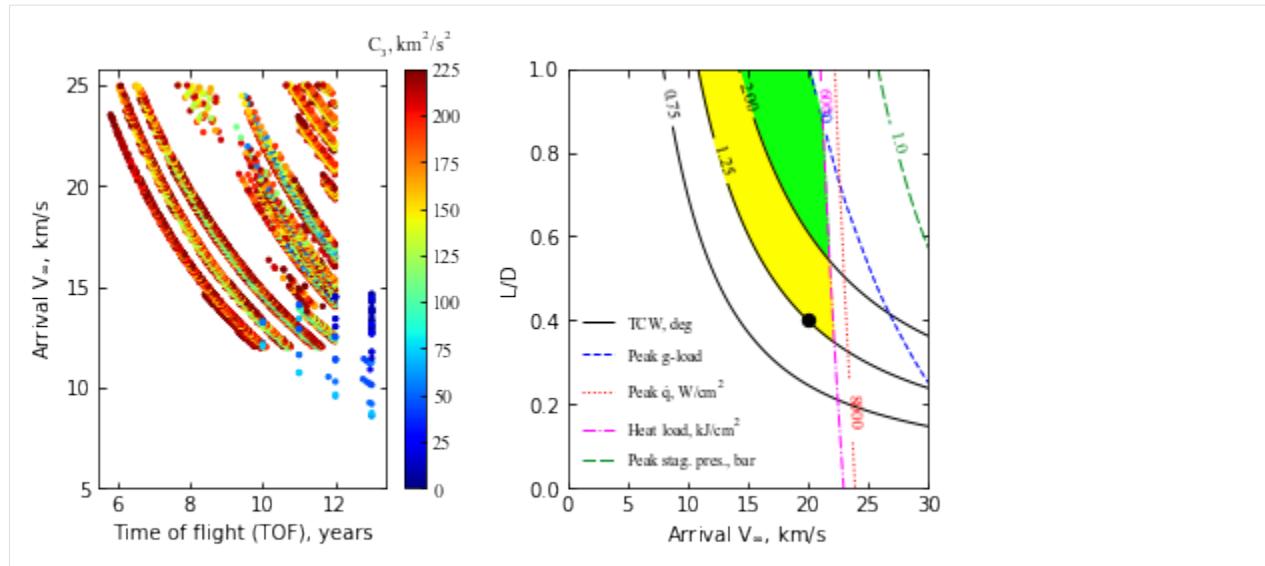
y7 = np.minimum(y5,y6)

plt.fill_between(x4, y4, y7, where=y4<=y7,color='xkcd:bright yellow')
plt.xlim([0.0,30.0])
plt.ylim([0.0,1.0])

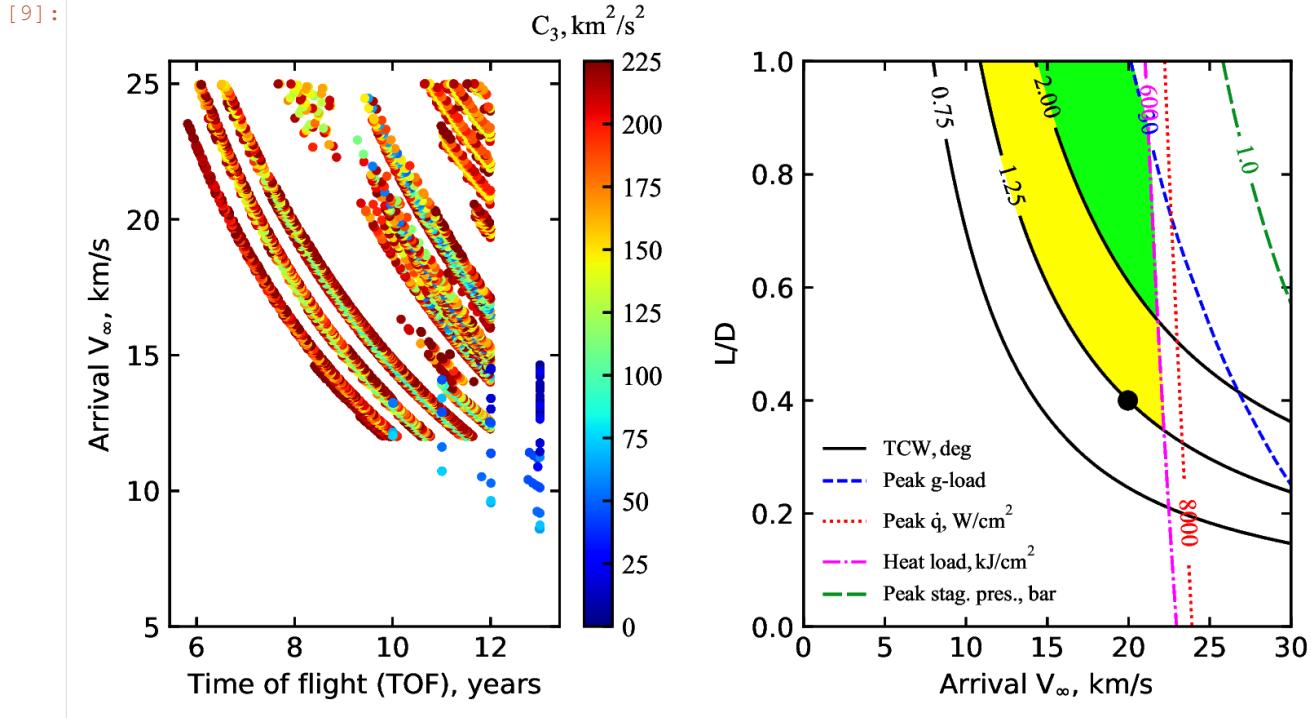
plt.savefig('../plots/girijaSaikia2019b.png',bbox_inches='tight')
plt.savefig('../plots/girijaSaikia2019b.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/girijaSaikia2019b.eps', dpi=300,bbox_inches='tight')

plt.show()

```



```
[9]: from IPython.display import Image
Image(filename='../plots/girijaSaikia2019b-high-res.png', width=800)
```



The plots are now saved in plots/girijaSaikia2019b

Congratulations! You have created the aerocapture feasibility chart for Neptune. The black dot indicates a baseline reference design selected for Monte Carlo analysis in the next example.

4.11 Example - 11 - Neptune Aerocapture - Part 2a: Monte Carlo Simulations

In this example, we will use AMAT to perform Monte Carlo simulations to assess aerocapture vehicle performance.

We reproduce the example Monte Carlo results from “Girija, Saikia, Longuski et al. Feasibility and Performance Analysis of Neptune Aerocapture Using Heritage Blunt-Body Aeroshells, Journal of Spacecraft and Rockets, June, 2020, In press. DOI: 10.2514/1.A34719. Refer Section VIII A: Results for `prograde` entry with maximum range of FMINMAX.

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle

import numpy as np
from scipy import interpolate
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.patches import Polygon
```

```
[2]: # Create a planet object
planet=Planet("NEPTUNE")

# Load an nominal atmospheric profile with height, temp, pressure, density data
planet.loadAtmosphereModel('../atmdata/Neptune/neptune-gram-avg.dat', 0 , 7 , 6 , 5 ,
                           ↴
                           heightInKmFlag=True)

# Create a vehicle object
vehicle=Vehicle('Trident', 1000.0, 200.0, 0.40, 3.1416, 0.0, 1.00, planet)

# Set vehicle conditions at entry interface
# The EFPA selection process is described in Sec. VII in the reference article.
vehicle.setInitialState(1000.0, 0.0, 0.0, 28.00, 0.0,-13.85, 0.0, 0.0)
vehicle.setSolverParams(1E-6)
```

```
[3]: # Set the guidance parameters described in the paper.
# See the function description for parameter details.

# Set max roll rate constraint to 30 deg/s
vehicle.setMaxRollRate(30.0)

# Set Ghdot = 75
# Set Gq = 3.0
# Set v_switch_kms = 18.9
# Set low_Alt_km = 120
# Set numPoints_lowAlt = 101
# Set hdot_threshold = -500 m/s
vehicle.setEquilibriumGlideParams(75.0, 3.0, 18.9, 120.0, 101, -500.0)

# Set target orbit parameters
# periapsis = 4000.0 km
# apoapsis = 400,000 km
# apoapsis tolerance = 10 km
vehicle.setTargetOrbitParams(4000.0, 400.0E3, 10.0E3)
```

```
[4]: # Set path to atmfiles with randomly perturbed atmosphere files.
```

```
atmfiles = ['../atmdata/Neptune/FMINMAX-10L.txt',
            '../atmdata/Neptune/FMINMAX-08L.txt',
            '../atmdata/Neptune/FMINMAX-06L.txt',
            '../atmdata/Neptune/FMINMAX-04L.txt',
            '../atmdata/Neptune/FMINMAX-02L.txt',
            '../atmdata/Neptune/FMINMAX+00L.txt',
            '../atmdata/Neptune/FMINMAX+02L.txt',
            '../atmdata/Neptune/FMINMAX+04L.txt',
            '../atmdata/Neptune/FMINMAX+06L.txt',
            '../atmdata/Neptune/FMINMAX+08L.txt',
            '../atmdata/Neptune/FMINMAX+10L.txt']
```

```
[5]: # Set up Monte Carlo simulation parameters
```

```
# See function description for details.

# NPOS = 1086
# NMONTE = 200

vehicle.setupMonteCarloSimulation(1086, 200, atmfiles, 0, 1, 2, 3, 4, True, \
                                  -13.85, 0.11, 0.40, 0.013, 0.5, 0.1, 2400.0)
```

```
[6]: # Run the Monte Carlo simulation.
```

```
# N = 10 shown here, run for a few thousand to be realistic. This will take several hours.

vehicle.runMonteCarlo(10, '../data/girijaSaikia2020a/MCB1')

BATCH :../data/girijaSaikia2020a/MCBX, RUN #: 1, PROF: ../atmdata/Neptune/FMINMAX-04L.
        ↪txt, SAMPLE #: 22, EFPA: -13.97, SIGMA: -1.04, LD: 0.41, APO : 379646.10
BATCH :../data/girijaSaikia2020a/MCBX, RUN #: 2, PROF: ../atmdata/Neptune/FMINMAX+10L.
        ↪txt, SAMPLE #: 194, EFPA: -13.86, SIGMA: 1.66, LD: 0.39, APO : 397722.98
BATCH :../data/girijaSaikia2020a/MCBX, RUN #: 3, PROF: ../atmdata/Neptune/FMINMAX+04L.
        ↪txt, SAMPLE #: 188, EFPA: -13.89, SIGMA: -0.71, LD: 0.39, APO : 367641.10
BATCH :../data/girijaSaikia2020a/MCBX, RUN #: 4, PROF: ../atmdata/Neptune/FMINMAX+10L.
        ↪txt, SAMPLE #: 125, EFPA: -13.79, SIGMA: -1.81, LD: 0.41, APO : 360589.69
BATCH :../data/girijaSaikia2020a/MCBX, RUN #: 5, PROF: ../atmdata/Neptune/FMINMAX-02L.
        ↪txt, SAMPLE #: 172, EFPA: -13.83, SIGMA: 1.24, LD: 0.40, APO : 383384.65
BATCH :../data/girijaSaikia2020a/MCBX, RUN #: 6, PROF: ../atmdata/Neptune/FMINMAX+02L.
        ↪txt, SAMPLE #: 29, EFPA: -13.77, SIGMA: -0.44, LD: 0.39, APO : 372647.16
BATCH :../data/girijaSaikia2020a/MCBX, RUN #: 7, PROF: ../atmdata/Neptune/FMINMAX-04L.
        ↪txt, SAMPLE #: 63, EFPA: -13.71, SIGMA: 0.76, LD: 0.39, APO : 376441.97
BATCH :../data/girijaSaikia2020a/MCBX, RUN #: 8, PROF: ../atmdata/Neptune/FMINMAX+00L.
        ↪txt, SAMPLE #: 162, EFPA: -13.67, SIGMA: -2.19, LD: 0.41, APO : 380977.85
BATCH :../data/girijaSaikia2020a/MCBX, RUN #: 9, PROF: ../atmdata/Neptune/FMINMAX+06L.
        ↪txt, SAMPLE #: 60, EFPA: -13.73, SIGMA: 0.86, LD: 0.40, APO : 382393.74
BATCH :../data/girijaSaikia2020a/MCBX, RUN #: 10, PROF: ../atmdata/Neptune/
        ↪FMINMAX+00L.txt, SAMPLE #: 161, EFPA: -13.77, SIGMA: 0.79, LD: 0.40, APO : 368610.27
```

```
[7]: # Post process Monte Carlo simulation data.
```

```
peri = np.loadtxt('../data/girijaSaikia2020a/MCB1/terminal_periapsis_arr.txt')
apoa = np.loadtxt('../data/girijaSaikia2020a/MCB1/terminal_apoapsis_arr.txt')
```

(continues on next page)

(continued from previous page)

```

peri_dv = np.loadtxt('../data/girijaSaikia2020a/MCB1/periapsis_raise_DV_arr.txt')

del_index1 = np.where(apoa < 0)
del_index2 = np.where(apoa>800.0E3)

del_index = np.concatenate((del_index1, del_index2), axis=1)

print('Simulation statistics')
print('-----')
print("No. of cases escaped :" +str(len(del_index1[0])))
print("No. of cases with apo. alt > 800.0E3 km :" +str(len(del_index2[0])))

Simulation statistics
-----
No. of cases escaped :3
No. of cases with apo. alt > 800.0E3 km :0

```

```
[8]: # Remove escaped cases before plotting
peri_new = np.delete(peri, del_index)
apoa_new = np.delete(apoa, del_index)
peri_dv_new = np.delete(peri_dv, del_index)
```

```
[9]: # Create apoapsis dispersion plot.

fig = plt.figure()
fig.set_size_inches([3.25,3.25])
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

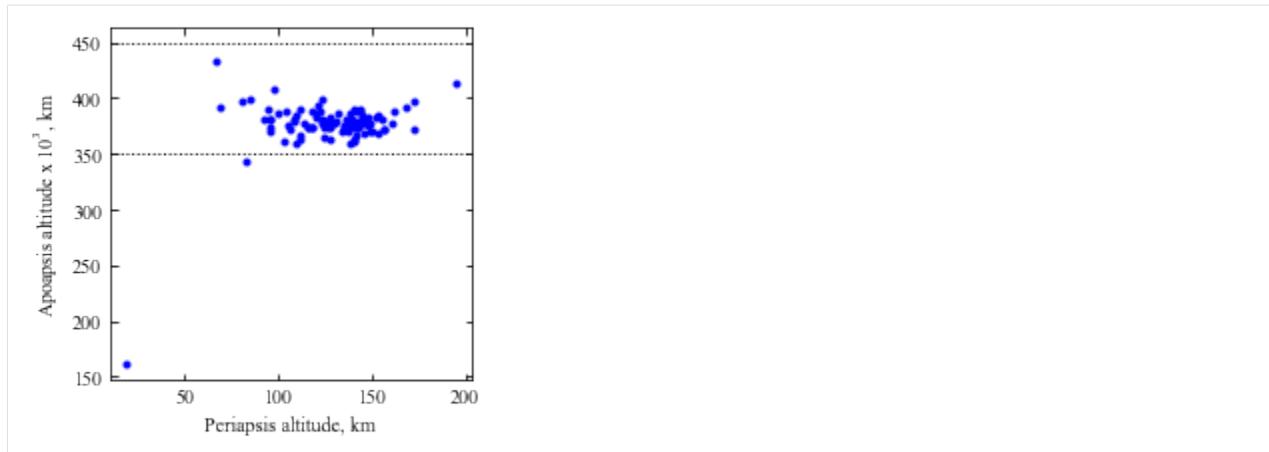
plt.plot(peri_new, apoa_new/1000.0, 'bo', markersize=3)

plt.xlabel('Periapsis altitude, km', fontsize=10)
plt.ylabel('Apoapsis altitude x '+r'$10^{3}$', km, fontsize=10)

plt.axhline(y=350.0, linewidth=1, color='k', linestyle='dotted')
plt.axhline(y=450.0, linewidth=1, color='k', linestyle='dotted')

ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=10)
ax.tick_params(axis='y',labelsize=10)

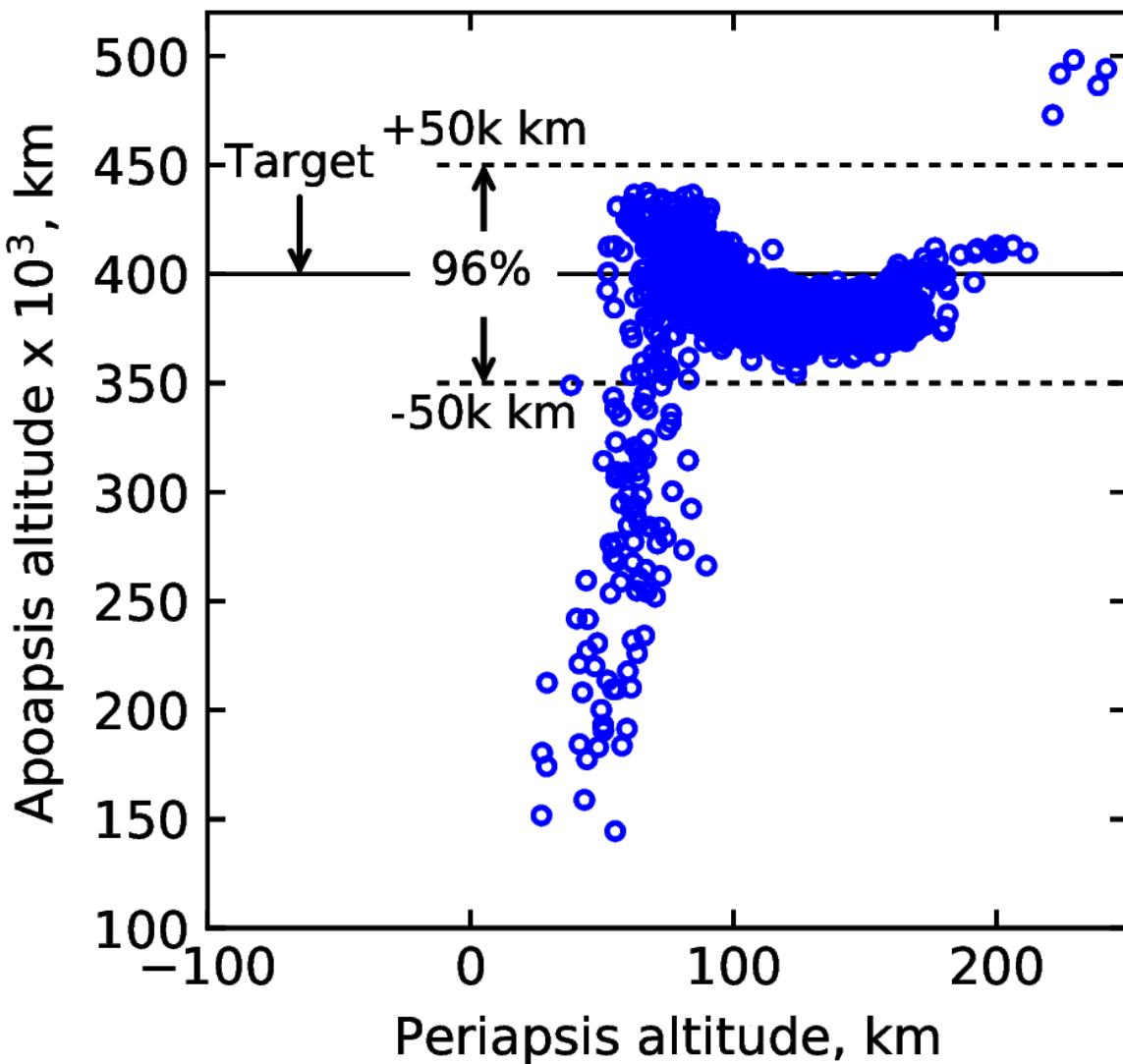
plt.savefig('../plots/girijaSaikia2020a-fig-15-N100.png',bbox_inches='tight')
plt.savefig('../plots/girijaSaikia2020a-fig-15-N100.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/girijaSaikia2020a-fig-15-N100.eps', dpi=300,bbox_inches='tight')
```



This plot only has 100 trajectories. Below is a similar plot with 5000 trajectories

```
[15]: from IPython.display import Image  
Image(filename='../plots/prograde-higher-res.png', width=600)
```

[15]:



[13]: # Create histogram of periapse raise manuever DV

```

fig = plt.figure()
fig.set_size_inches([3.25,3.25])
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.hist(peri_dv_new, bins=100, color='xkcd:periwinkle')
plt.xlabel('Periapse raise '+r'$\Delta V$'+', m/s', fontsize=10)
plt.ylabel('Number of cases', fontsize=10)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=10)
ax.tick_params(axis='y',labelsize=10)

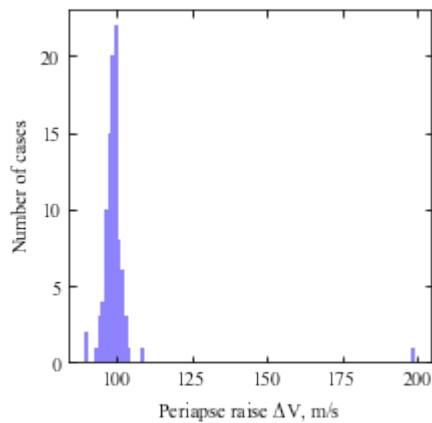
```

(continues on next page)

(continued from previous page)

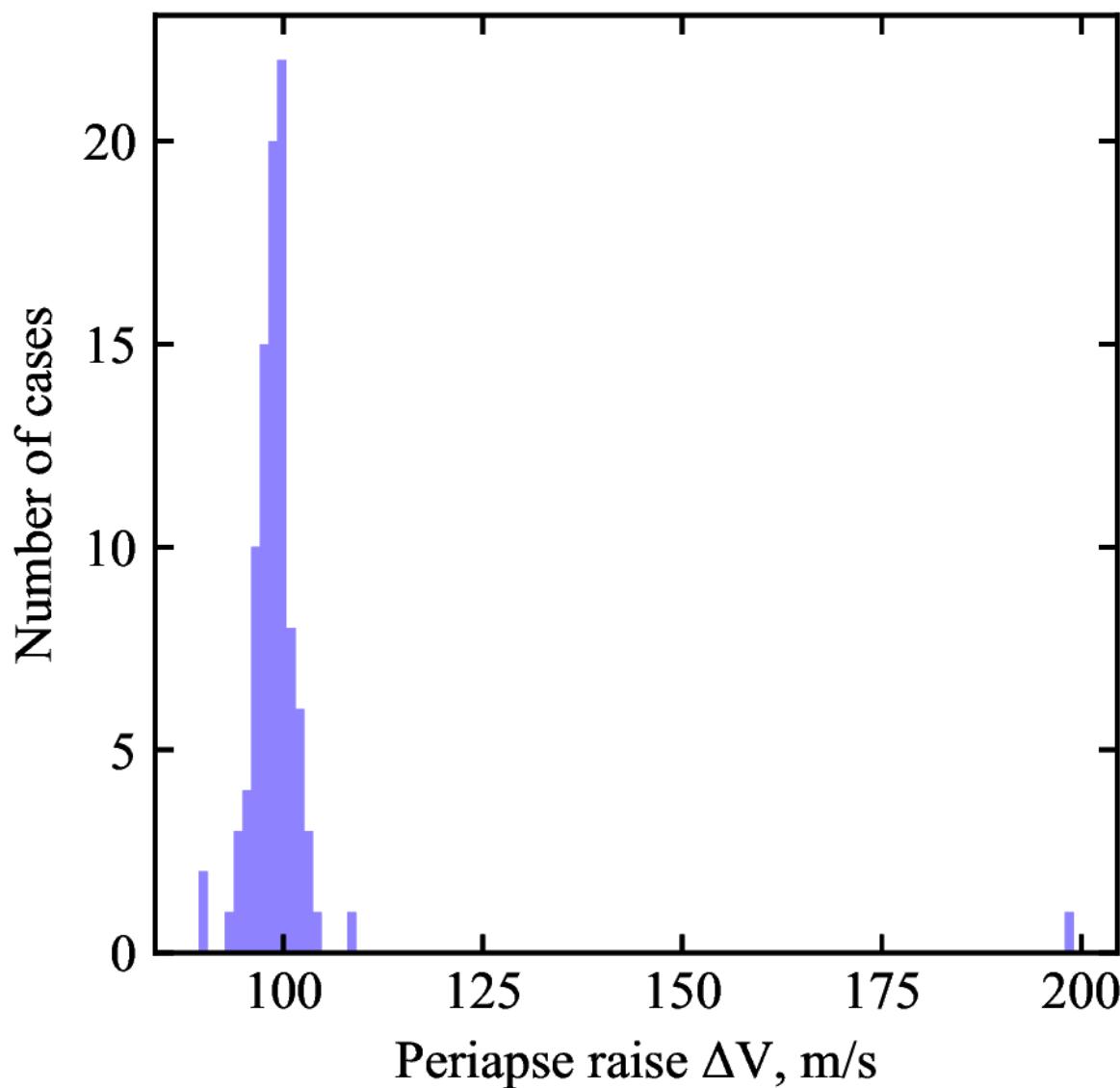
```
plt.savefig('../plots/girijaSaikia2020a-prm-histogram.png',bbox_inches='tight')
plt.savefig('../plots/girijaSaikia2020a-prm-histogram.pdf', dpi=300,bbox_inches='tight
')
plt.savefig('../plots/girijaSaikia2020a-prm-histogram.eps', dpi=300,bbox_inches='tight
')

plt.show()
```



```
[14]: from IPython.display import Image
Image(filename='../plots/girijaSaikia2020a-PRM-higher-res.png', width=600)
```

[14] :



4.12 Example - 12 - Neptune Aerocapture - Part 2b: Monte Carlo Simulations

In this example, we will demonstrate the another published result for aerocapture at Neptune.

We reproduce the example Monte Carlo results from “Girija, Saikia, Longuski et al. Feasibility and Performance Analysis of Neptune Aerocapture Using Heritage Blunt-Body Aeroshells, Journal of Spacecraft and Rockets, June, 2020, In press. DOI: 10.2514/1.A34719. Refer Section VIII A: Results for **retrograde** entry with maximum range of FMINMAX.

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

(continues on next page)

(continued from previous page)

```
import numpy as np
from scipy import interpolate
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.patches import Polygon
```

```
[2]: # Create a planet object
planet=Planet("NEPTUNE")

# Load an nominal atmospheric profile with height, temp, pressure, density data
planet.loadAtmosphereModel('../atmdata/Neptune/neptune-gram-avg.dat', 0, 7, 6, 5, \
    ↴
        heightInKmFlag=True)

# Create a vehicle object
vehicle=Vehicle('Trident', 1000.0, 200.0, 0.40, 3.1416, 0.0, 1.00, planet)

# Set vehicle conditions at entry interface
# Note these conditions are for retrograde equatorial entry
# The EFPA selection process is described in Sec. VII in the reference article.
vehicle.setInitialState(1000.0, 0.0, 0.0, 33.00, 180.0,-11.43, 0.0, 0.0)
vehicle.setSolverParams(1E-6)
```

```
[3]: vehicle.setMaxRollRate(30.0)
vehicle.setEquilibriumGlideParams(75.0, 3.0, 18.9, 120.0, 101, -500.0)
vehicle.setTargetOrbitParams(4000.0, 400.0E3, 10.0E3)
```

```
[4]: atmfiles = ['../atmdata/Neptune/FMINMAX-10L.txt',
    '../atmdata/Neptune/FMINMAX-08L.txt',
    '../atmdata/Neptune/FMINMAX-06L.txt',
    '../atmdata/Neptune/FMINMAX-04L.txt',
    '../atmdata/Neptune/FMINMAX-02L.txt',
    '../atmdata/Neptune/FMINMAX+00L.txt',
    '../atmdata/Neptune/FMINMAX+02L.txt',
    '../atmdata/Neptune/FMINMAX+04L.txt',
    '../atmdata/Neptune/FMINMAX+06L.txt',
    '../atmdata/Neptune/FMINMAX+08L.txt',
    '../atmdata/Neptune/FMINMAX+10L.txt']
```

```
[5]: vehicle.setupMonteCarloSimulation(1086, 200, atmfiles, 0, 1, 2, 3, 4, True, \
    ↴
        -11.43, 0.11, 0.40, 0.013, 0.5, 0.1, 2400.0)
```

```
[6]: # N = 10 shown here, run for a few thousand to be realistic. This will take several \
    ↴hours.
```

```
vehicle.runMonteCarlo(100, '../data/girijaSaikia2020b/MCB1')
```

```
BATCH :../data/girijaSaikia2020b/MCBX, RUN #: 1, PROF: ../atmdata/Neptune/FMINMAX+06L. \
    ↴txt, SAMPLE #: 42, EFPA: -11.53, SIGMA: 1.86, LD: 0.41, APO : 210233.74
BATCH :../data/girijaSaikia2020b/MCBX, RUN #: 2, PROF: ../atmdata/Neptune/FMINMAX+00L. \
    ↴txt, SAMPLE #: 82, EFPA: -11.47, SIGMA: 0.65, LD: 0.42, APO : 404021.05
BATCH :../data/girijaSaikia2020b/MCBX, RUN #: 3, PROF: ../atmdata/Neptune/FMINMAX-08L. \
    ↴txt, SAMPLE #: 197, EFPA: -11.39, SIGMA: -0.66, LD: 0.39, APO : 389028.95
```

(continues on next page)

(continued from previous page)

```
BATCH :./data/girijaSaikia2020b/MCBX, RUN #: 4, PROF: ./atmdata/Neptune/FMINMAX+06L.
→txt, SAMPLE #: 172, EFPA: -11.34, SIGMA: 0.10, LD: 0.40, APO : 374039.92
BATCH :./data/girijaSaikia2020b/MCBX, RUN #: 5, PROF: ./atmdata/Neptune/FMINMAX+08L.
→txt, SAMPLE #: 160, EFPA: -11.48, SIGMA: -0.25, LD: 0.41, APO : 294339.22
BATCH :./data/girijaSaikia2020b/MCBX, RUN #: 6, PROF: ./atmdata/Neptune/FMINMAX+04L.
→txt, SAMPLE #: 196, EFPA: -11.46, SIGMA: 0.90, LD: 0.40, APO : 404630.18
BATCH :./data/girijaSaikia2020b/MCBX, RUN #: 7, PROF: ./atmdata/Neptune/FMINMAX-10L.
→txt, SAMPLE #: 167, EFPA: -11.44, SIGMA: 1.80, LD: 0.38, APO : 386631.37
BATCH :./data/girijaSaikia2020b/MCBX, RUN #: 8, PROF: ./atmdata/Neptune/FMINMAX+08L.
→txt, SAMPLE #: 78, EFPA: -11.53, SIGMA: -0.14, LD: 0.42, APO : 209689.06
BATCH :./data/girijaSaikia2020b/MCBX, RUN #: 9, PROF: ./atmdata/Neptune/FMINMAX+08L.
→txt, SAMPLE #: 135, EFPA: -11.46, SIGMA: 0.30, LD: 0.40, APO : 245398.87
BATCH :./data/girijaSaikia2020b/MCBX, RUN #: 10, PROF: ./atmdata/Neptune/
→FMINMAX+10L.txt, SAMPLE #: 45, EFPA: -11.34, SIGMA: 0.40, LD: 0.40, APO : 410500.72
```

```
[7]: peri = np.loadtxt('~/data/girijaSaikia2020b/MCB1/terminal_periapsis_arr.txt')
apoa = np.loadtxt('~/data/girijaSaikia2020b/MCB1/terminal_apoapsis_arr.txt')

peri_dv = np.loadtxt('~/data/girijaSaikia2020b/MCB1/periapsis_raise_DV_arr.txt')

del_index1 = np.where(apoa < 0)
del_index2 = np.where(apoa>800.0E3)

del_index = np.concatenate((del_index1, del_index2), axis=1)

print('Simulation statistics')
print('-----')
print("No. of cases escaped :" +str(len(del_index1[0])))
print("No. of cases with apo. alt > 800.0E3 km :" +str(len(del_index2[0])))

Simulation statistics
-----
No. of cases escaped :0
No. of cases with apo. alt > 800.0E3 km :0
```

```
[8]: peri_new = np.delete(peri, del_index)
apoa_new = np.delete(apoa, del_index)
peri_dv_new = np.delete(peri_dv, del_index)
```

```
[9]: fig = plt.figure()
fig.set_size_inches([3.25,3.25])
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.plot(peri_new, apoa_new/1000.0, 'bo', markersize=3)

plt.xlabel('Periapsis altitude, km', fontsize=10)
plt.ylabel('Apoapsis altitude x '+r'$10^3$+', km', fontsize=10)

plt.axhline(y=350.0, linewidth=1, color='k', linestyle='dotted')
plt.axhline(y=450.0, linewidth=1, color='k', linestyle='dotted')
```

(continues on next page)

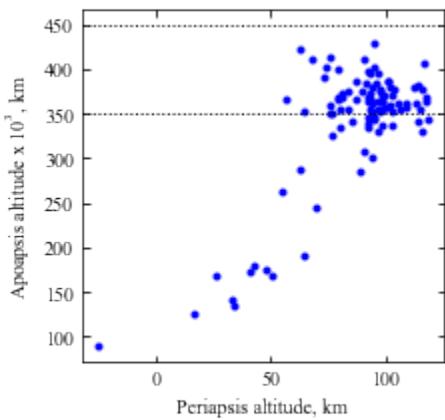
(continued from previous page)

```

ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=10)
ax.tick_params(axis='y',labelsize=10)

plt.savefig('../plots/girijaSaikia2020b-fig-13-N100.png',bbox_inches='tight')
plt.savefig('../plots/girijaSaikia2020b-fig-13-N100.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/girijaSaikia2020b-fig-13-N100.eps', dpi=300,bbox_inches='tight')

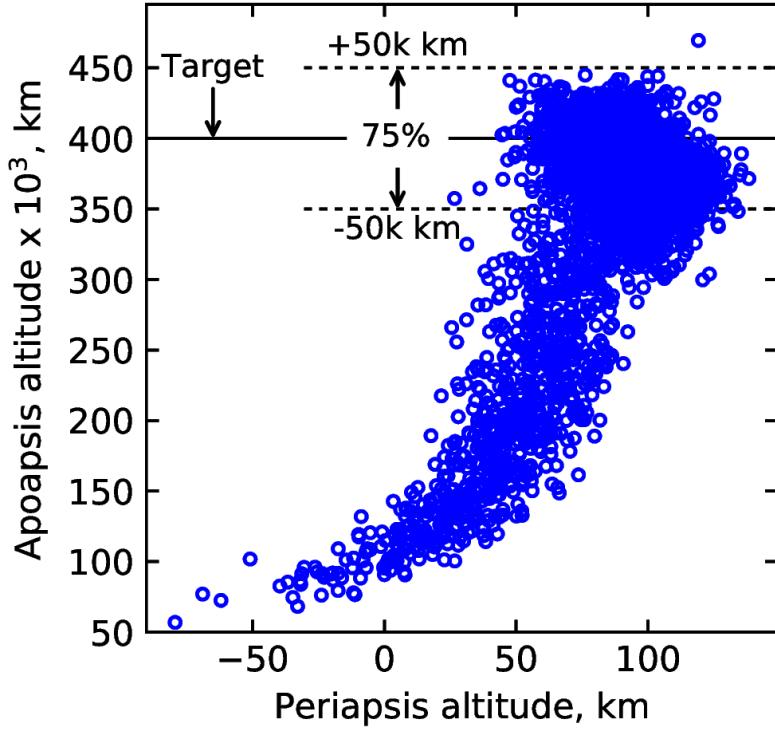
```



This plot only has 100 trajectories. Below is a similar plot with 5000 trajectories

```
[11]: from IPython.display import Image
Image(filename='../plots/retrograde-higher-res.png', width=400)
```

[11]:



[12]:

```

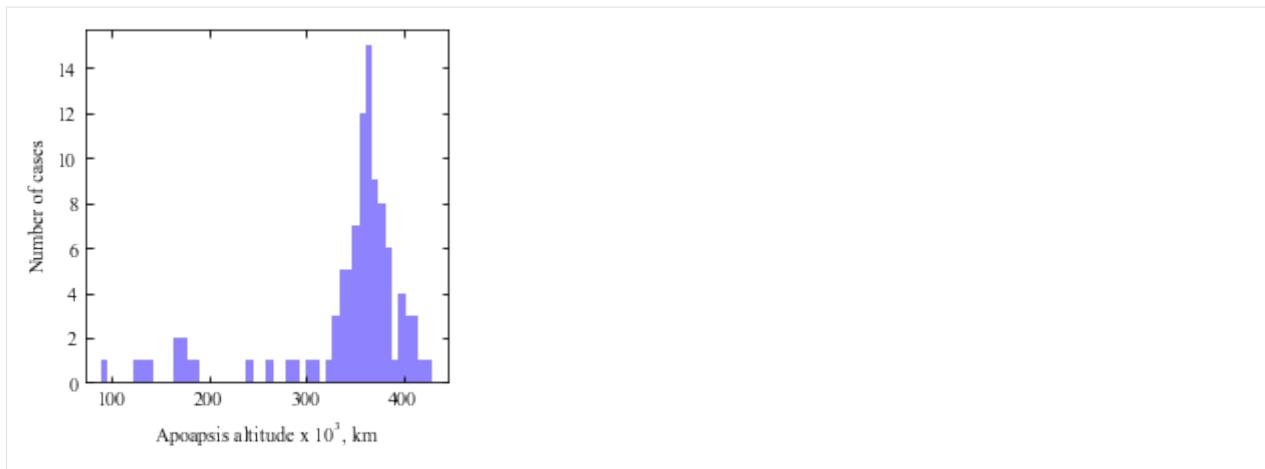
fig = plt.figure()
fig.set_size_inches([3.25,3.25])
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.hist(apoa_new/1000.0, bins=50, color='xkcd:periwinkle')
plt.xlabel('Apoapsis altitude x '+r'$10^3$'+', km', fontsize=10)
plt.ylabel('Number of cases', fontsize=10)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=10)
ax.tick_params(axis='y',labelsize=10)

plt.savefig('../plots/girijaSaikia2020b-apo-histogram-N100.png',bbox_inches='tight')
plt.savefig('../plots/girijaSaikia2020b-apo-histogram-N100.pdf', dpi=300,bbox_inches=
    ↪'tight')
plt.savefig('../plots/girijaSaikia2020b-apo-histogram-N100.eps', dpi=300,bbox_inches=
    ↪'tight')

plt.show()

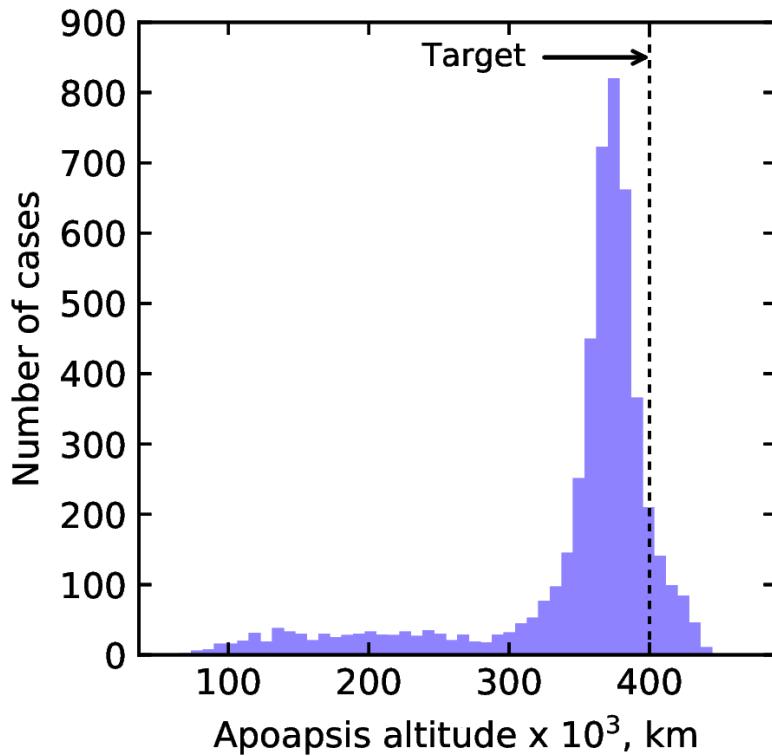
```



With more runs, the apoapsis histograms is as shown below.

```
[16]: from IPython.display import Image
Image(filename='../plots/retrograde-apo-hist-higher-res.png', width=400)
```

[16]:



Congratulations! You have now performed and post-processed the results from a Monte Carlo simulation for aerocapture at Neptune.

4.13 Example - 13 - Venus Aerocapture - Part 5a: Monte Carlo Simulations

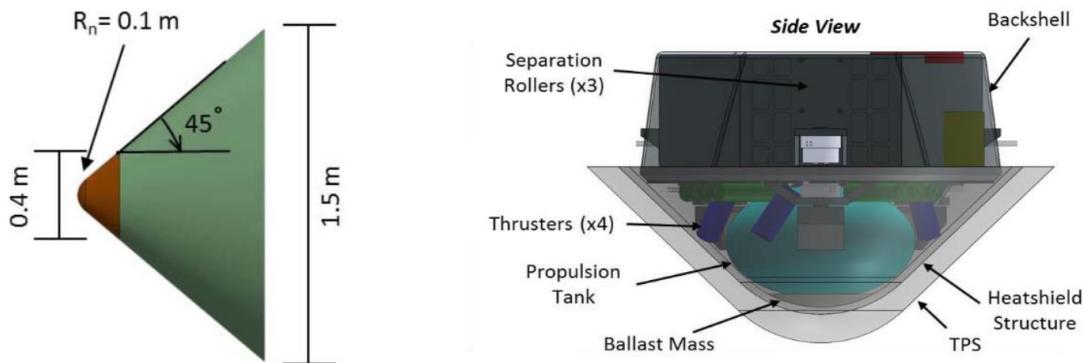
In this example, we will demonstrate the application of aerocapture for SmallSat missions to Venus.

We analyze the design proposed by “Austin et al. SmallSat Aerocapture to Enable a New Paradigm of Planetary Missions, IEEE Aerospace Conference, 2019, Big Sky, MT. DOI: 10.1109/AERO.2019.8742220

Shown below is the aerocapture vehicle design for Venus SmallSat proposed by Austin et al. The design consists of a drag skirt (shown in green), which is jettisoned. The vehicle parameters are $m = 68.1 \text{ kg}$, $\beta_1 = 38.1 \text{ kg/m}^2$, $\beta_2/\beta_1 = 7.5$. The objective is to insert the small satellite (shown in brown) into a 2,000 km x 200 km orbit around Venus. We will use AMAT to perform Monte Carlo analysis to assess aerocapture performance.

```
[3]: from IPython.display import Image
Image(filename='../plots/drag-modulation-vehicle.png', width=800)
```

[3]:



```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
from scipy import interpolate
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.patches import Polygon
```

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("VENUS")
planet.loadAtmosphereModel('../atmdata/Venus/venus-gram-avg.dat', 0 , 1 ,2, 3)
planet.h_skip = 150000.0

# Set up the drag modulation vehicle.
vehicle=Vehicle('DMVehicle', 68.2, 38.1, 0.0, 3.1416, 0.0, 0.10, planet)
vehicle.setInitialState(150.0,0.0,0.0,11.0,0.0,-5.50,0.0,0.0)
vehicle.setSolverParams(1E-6)
vehicle.setDragModulationVehicleParams(38.1,7.5)

# Set up the drag modulation entry phase guidance parameters.
vehicle.setDragEntryPhaseParams(6.0, 80.0, 101, -300.0)

# Set the target orbit parameters.
vehicle.setTargetOrbitParams(200.0, 2000.0, 50.0)

# Define the path to atmospheric files to be used for the Monte Carlo simulations.
atmfiles = ['../atmdata/Venus/LAT80S.txt',
            '../atmdata/Venus/LAT60S.txt',
            '../atmdata/Venus/LAT40S.txt',
```

(continues on next page)

(continued from previous page)

```

'../atmdata/Venus/LAT20S.txt',
'../atmdata/Venus/LAT10S.txt',
'../atmdata/Venus/LAT80N.txt',
'../atmdata/Venus/LAT60N.txt',
'../atmdata/Venus/LAT40N.txt',
'../atmdata/Venus/LAT20N.txt',
'../atmdata/Venus/LAT10N.txt',
]

# Set up the Monte Carlo simulation for drag modulation.
# NPOS = 151, NMONTE = 200
# Target EFPA = -5.40 deg
# EFPA 1-sigma error = +/- 0.033 deg
# Nominal beta_1 = 38.1 kg/m2
# beta_1 1-sigma = 0.0
# guidance time step for entry = 0.1s (Freq. = 10 Hz)
# guidance time step after jettison = 1.0 s
# max. solver time step = 0.1 s
# max. time used by solver = 1200 s

vehicle.setupMonteCarloSimulationD(151, 200, atmfiles, 0, 1, 2, 3, 4, True,
                                   -5.40, 0.033, 38.1, 0.0,
                                   0.1, 1.0, 0.1, 1200.0)

```

[4]: # N = 10 shown here, run for a few thousand to be realistic. This will take several hours.

```
vehicle.runMonteCarloD(10, '../data/austin2019/MCB1')
```

```

BATCH :../data/austin2019/MCX1, RUN #: 1, PROF: ../atmdata/Venus/LAT10N.txt, SAMPLE #:
→ 51, EFPA: -5.43, SIGMA: -1.67, APO : 2881.64
BATCH :../data/austin2019/MCX1, RUN #: 2, PROF: ../atmdata/Venus/LAT40S.txt, SAMPLE #:
→ 77, EFPA: -5.41, SIGMA: 0.10, APO : 2290.41
BATCH :../data/austin2019/MCX1, RUN #: 3, PROF: ../atmdata/Venus/LAT80N.txt, SAMPLE #:
→ 108, EFPA: -5.43, SIGMA: 2.31, APO : 4615.14
BATCH :../data/austin2019/MCX1, RUN #: 4, PROF: ../atmdata/Venus/LAT60S.txt, SAMPLE #:
→ 195, EFPA: -5.41, SIGMA: 1.61, APO : 2753.68
BATCH :../data/austin2019/MCX1, RUN #: 5, PROF: ../atmdata/Venus/LAT40S.txt, SAMPLE #:
→ 194, EFPA: -5.40, SIGMA: 0.26, APO : 2283.65
BATCH :../data/austin2019/MCX1, RUN #: 6, PROF: ../atmdata/Venus/LAT80N.txt, SAMPLE #:
→ 61, EFPA: -5.37, SIGMA: -1.13, APO : 2059.74
BATCH :../data/austin2019/MCX1, RUN #: 7, PROF: ../atmdata/Venus/LAT20N.txt, SAMPLE #:
→ 126, EFPA: -5.38, SIGMA: -0.53, APO : 2918.58
BATCH :../data/austin2019/MCX1, RUN #: 8, PROF: ../atmdata/Venus/LAT80N.txt, SAMPLE #:
→ 188, EFPA: -5.42, SIGMA: 2.00, APO : 2318.39
BATCH :../data/austin2019/MCX1, RUN #: 9, PROF: ../atmdata/Venus/LAT40N.txt, SAMPLE #:
→ 8, EFPA: -5.40, SIGMA: 1.06, APO : 2264.39
BATCH :../data/austin2019/MCX1, RUN #: 10, PROF: ../atmdata/Venus/LAT80N.txt, SAMPLE #:
→ 183, EFPA: -5.42, SIGMA: 0.05, APO : 2667.25

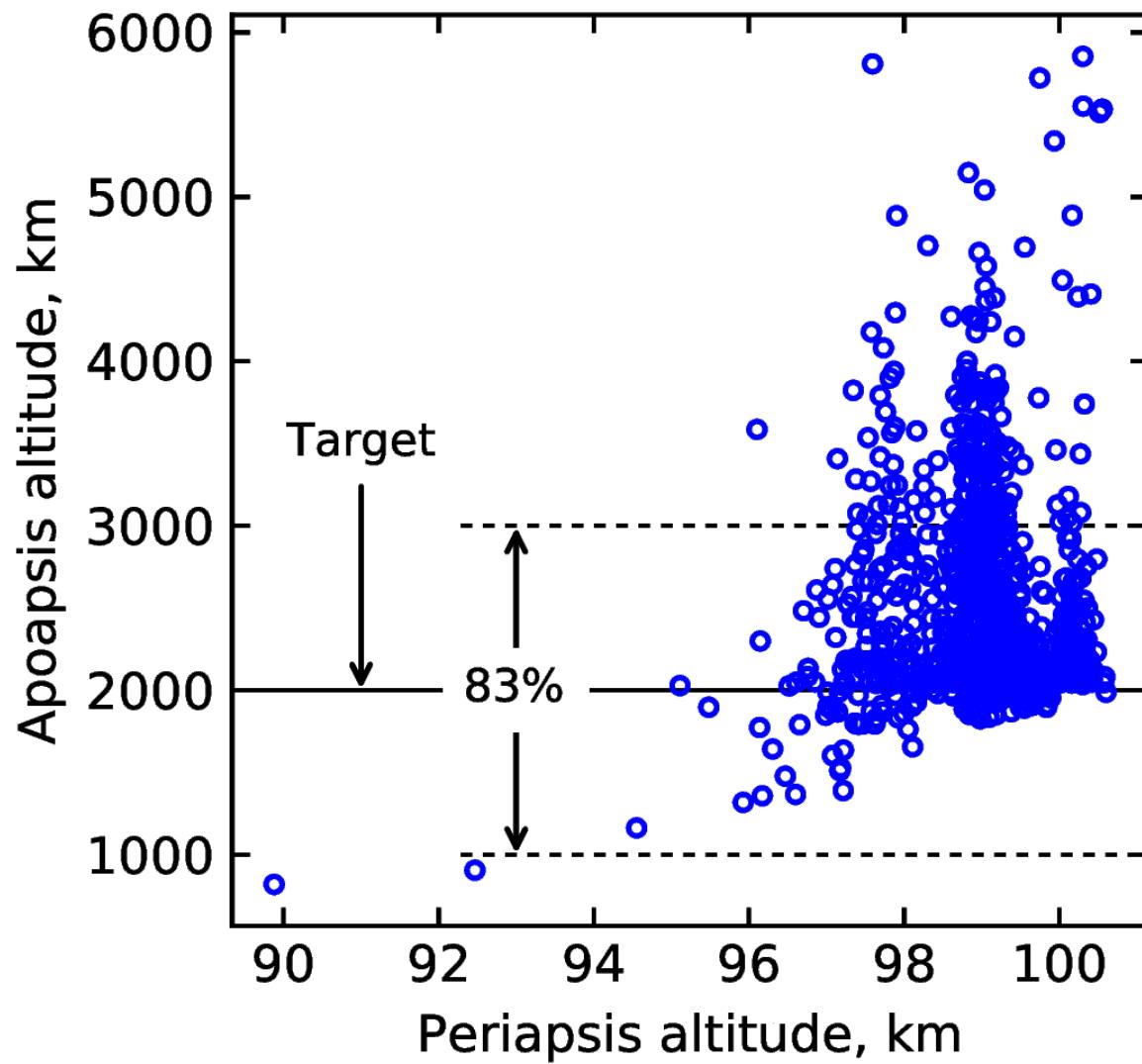
```

Shown below is the apoapsis dispersion for 1000 runs.

[5]:

```
from IPython.display import Image
Image(filename='../plots/austin-drag-modulation-N1000.png', width=600)
```

[5]:

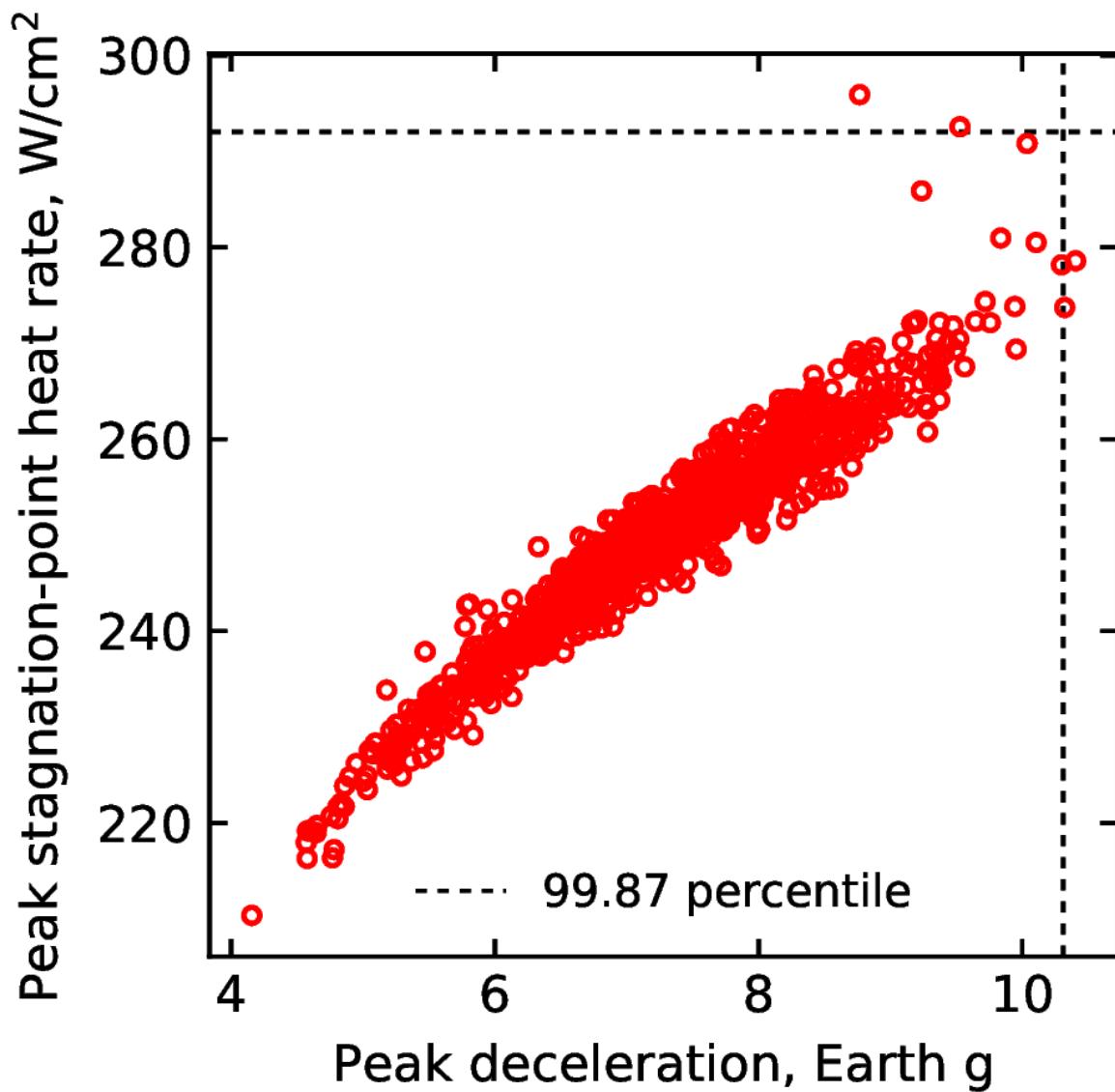


No cases resulted in escape or crashing into the planet. 3 of 1000 cases resulted in apoapsis exceeding 6000 km.

Shown below is the heating rate and g-load dispersion.

```
[6]: from IPython.display import Image  
Image(filename='../plots/austin-drag-modulation-heat.png', width=600)
```

[6]:



4.14 Example - 14 - Fire-II - Earth

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Fire II technology demonstrator for the Apollo heating re-entry environment. See: https://en.wikipedia.org/wiki/Project_FIRE

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("EARTH")
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0 , 1 , 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Fire-II', 86.5, 180.0, 0.0, 0.354, 0.0, 0.805, planet)

[5]: # Set up entry parameters
vehicle.setInitialState(120.0,0.0,0.0,11.35,0.0,-14.70,0.0,0.0)

[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
```

(continues on next page)

(continued from previous page)

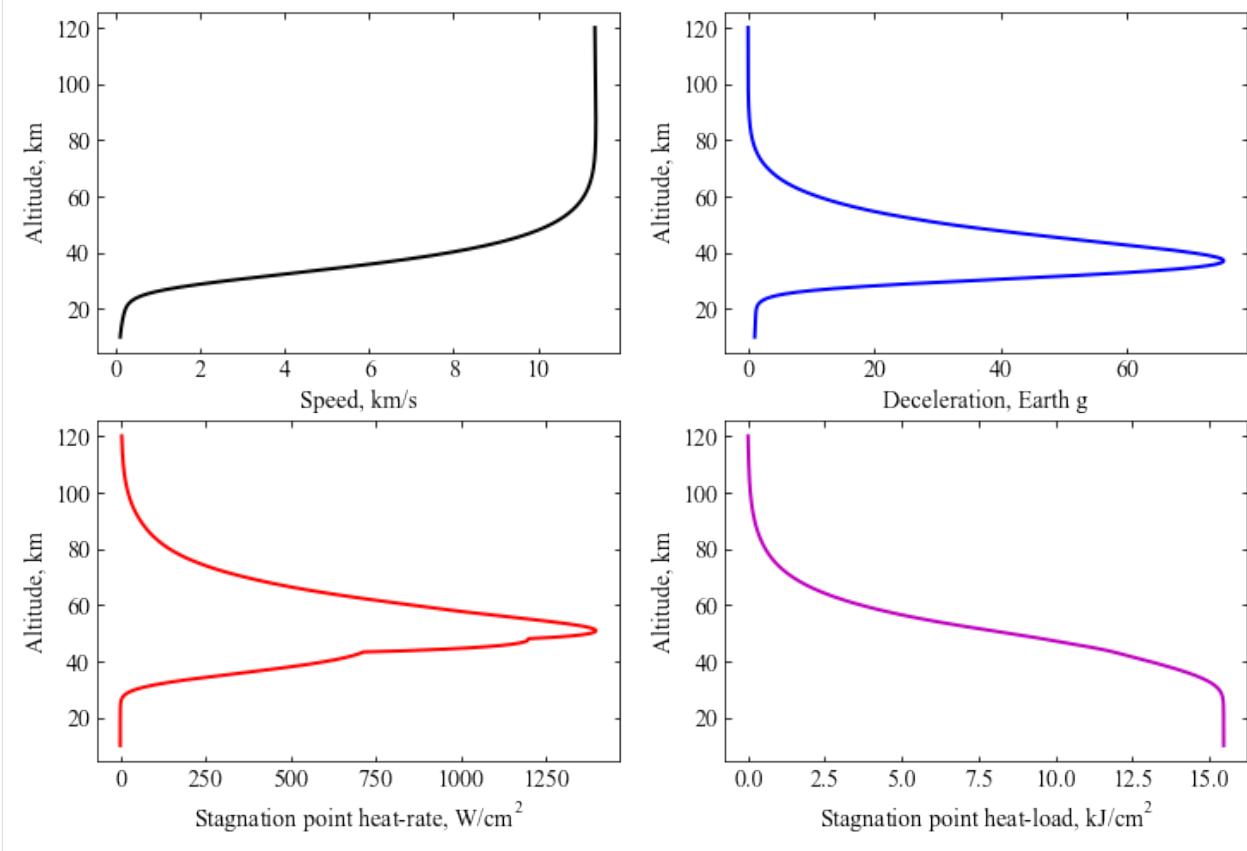
```

plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$\text{kJ}/\text{cm}^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/fire-II-earth.png', bbox_inches='tight')
plt.savefig('../plots/fire-II-earth.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/fire-II-earth.eps', dpi=300, bbox_inches='tight')

plt.show()

```



[]:

4.15 Example - 15 - Apollo-AS-201 - Earth

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Apollo AS-201 suborbital flight test. <https://en.wikipedia.org/wiki/AS-201>

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("EARTH")
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Apollo-AS-201', 5400.0, 400.0, 0.3, 12.0, 0.0, 3.0, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(120.0,0.0,0.0,7.67,0.0,-9.03,0.0,0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)
```

```
[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0,0.1,60.0)
# bank angle = 60 deg arbitrary, actual profile will give more realistic results
```

```
[8]: # import rcParams to set figure font type
from matplotlib import rcParams
```

```
[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc, 'r-', linewidth=2.0)
```

(continues on next page)

(continued from previous page)

```

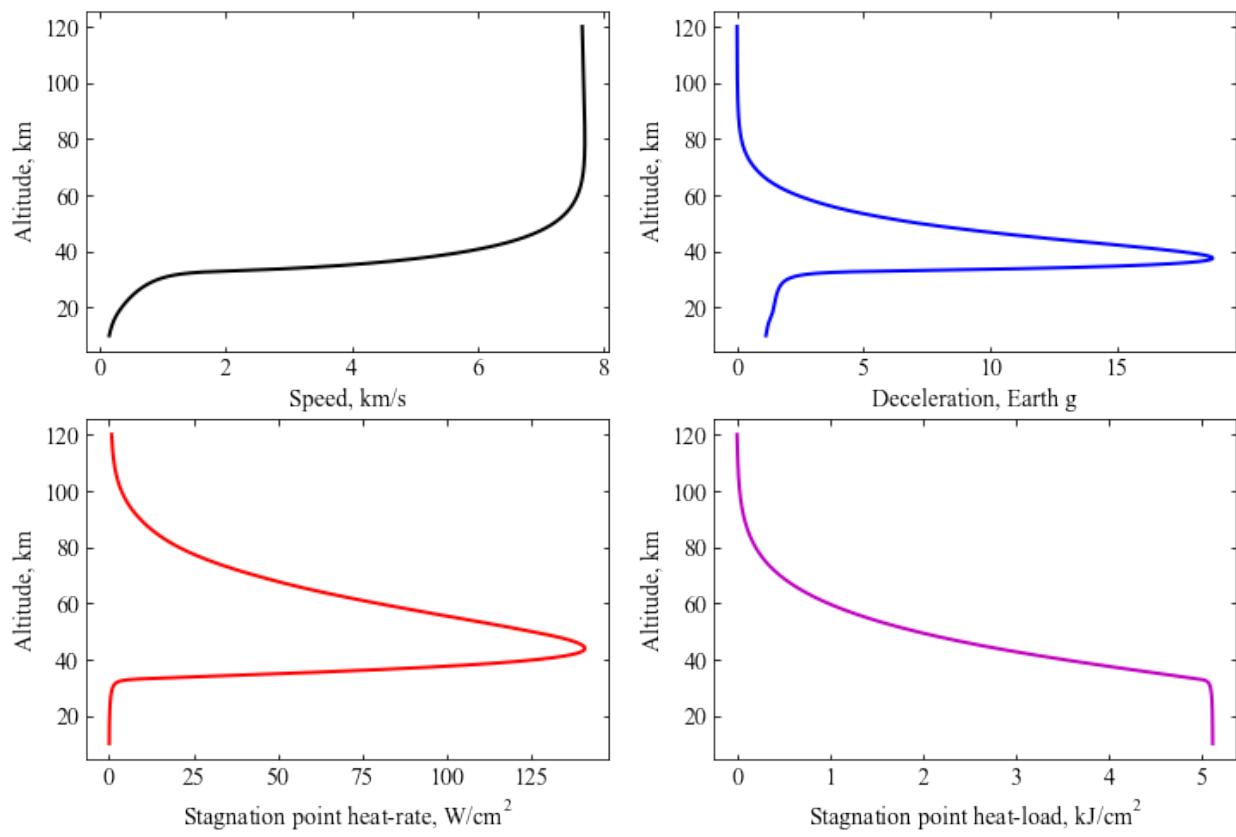
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/apollo-as-201-earth.png', bbox_inches='tight')
plt.savefig('../plots/apollo-as-201-earth.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/apollo-as-201-earth.eps', dpi=300, bbox_inches='tight')

plt.show()

```



[]:

4.16 Example - 16 - Apollo-AS-202 - Earth

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Apollo AS-202 suborbital flight test. <https://en.wikipedia.org/wiki/AS-202>

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("EARTH")
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0 , 1 ,2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Apollo-AS-202', 5400.0, 400.0, 0.3, 12.0, 0.0, 3.0, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(120.0,0.0,0.0,8.29,0.0,-3.53,0.0,0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)
```

```
[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0,0.1,70.0)
# bank angle = 70 deg arbitrary, actual profile will give more realistic results
```

```
[8]: # import rcParams to set figure font type
from matplotlib import rcParams
```

```
[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
```

(continues on next page)

(continued from previous page)

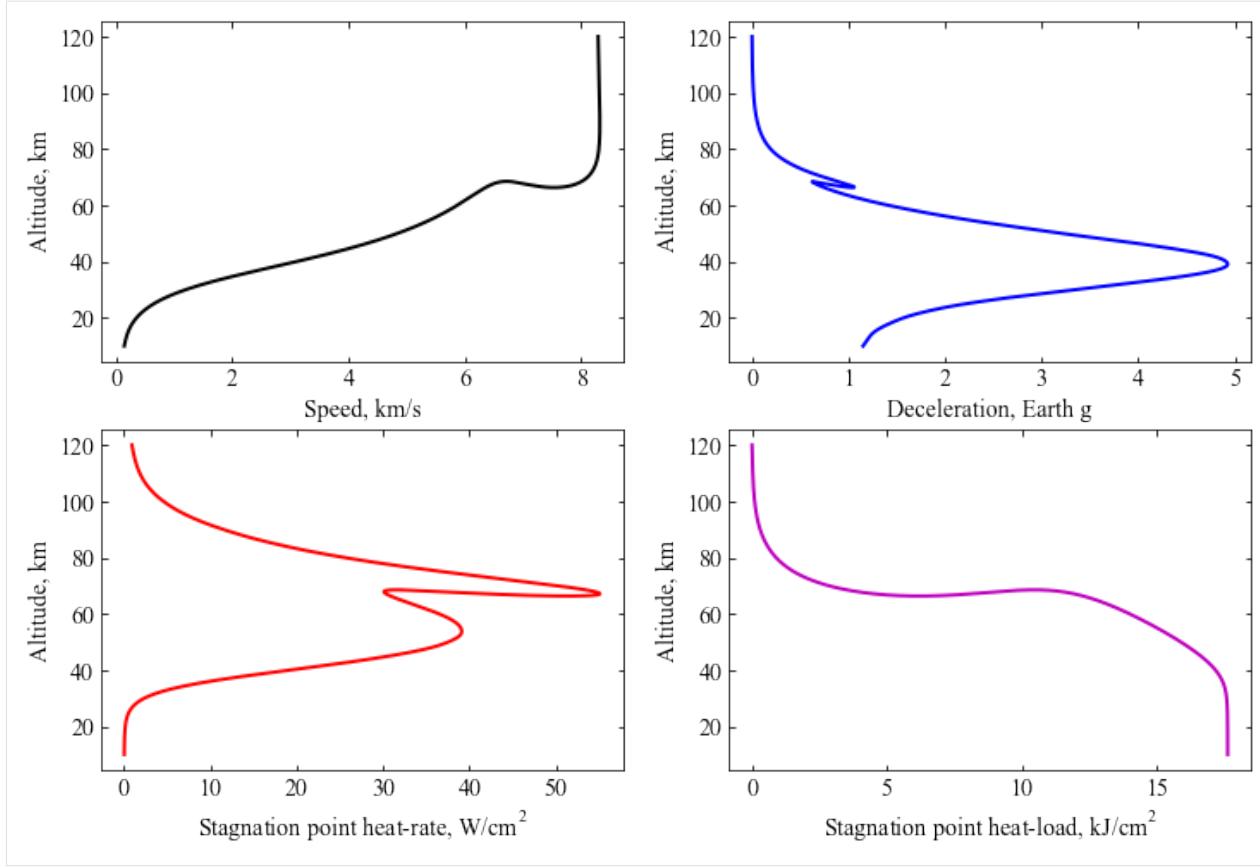
```
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc, 'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/apollo-as-202-earth.png', bbox_inches='tight')
plt.savefig('../plots/apollo-as-202-earth.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/apollo-as-202-earth.eps', dpi=300, bbox_inches='tight')

plt.show()
```



[]:

4.17 Example - 17 - Apollo-4 - Earth

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Apollo-4 flight test. https://en.wikipedia.org/wiki/Apollo_4

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("EARTH")
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0 , 1 ,2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Apollo-4', 5400.0, 400.0, 0.3, 12.0, 0.0, 3.0, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(120.0,0.0,0.0,10.73,0.0,-7.13,0.0,0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0,0.1,70.0)
# bank angle = 70 deg arbitrary, actual profile will give more realistic results

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

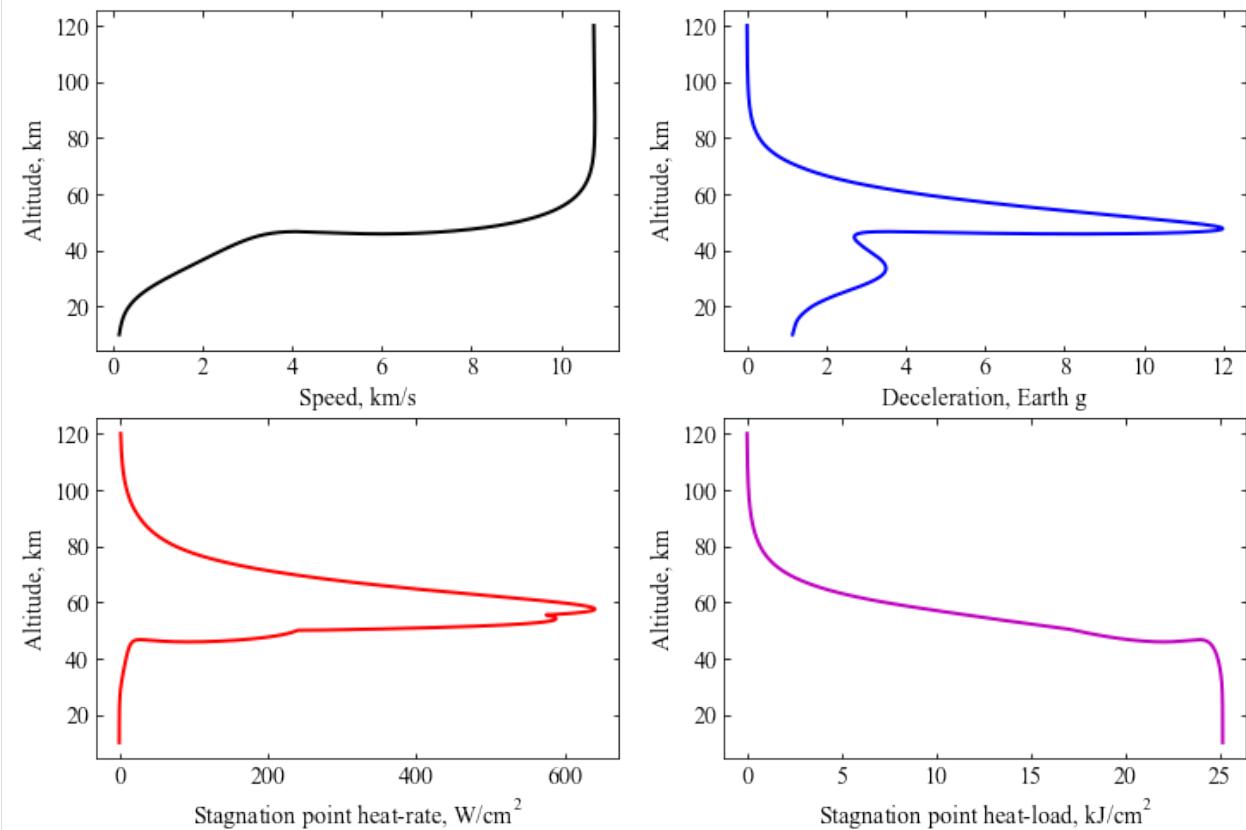
```

ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/apollo-4-earth.png', bbox_inches='tight')
plt.savefig('../plots/apollo-4-earth.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/apollo-4-earth.eps', dpi=300, bbox_inches='tight')

plt.show()

```



4.18 Example - 18 - Apollo-6 - Earth

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Apollo-6 flight test. https://en.wikipedia.org/wiki/Apollo_6

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("EARTH")
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Apollo-6', 5400.0, 400.0, 0.3, 12.0, 0.0, 3.0, planet)

[5]: # Set up entry parameters
vehicle.setInitialState(1200.0,0.0,0.0,9.60,0.0,-5.90,0.0,0.0)

[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0,0.1,60.0)
# bank angle = 70 deg arbitrary, actual profile will give more realistic results

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)
```

(continues on next page)

(continued from previous page)

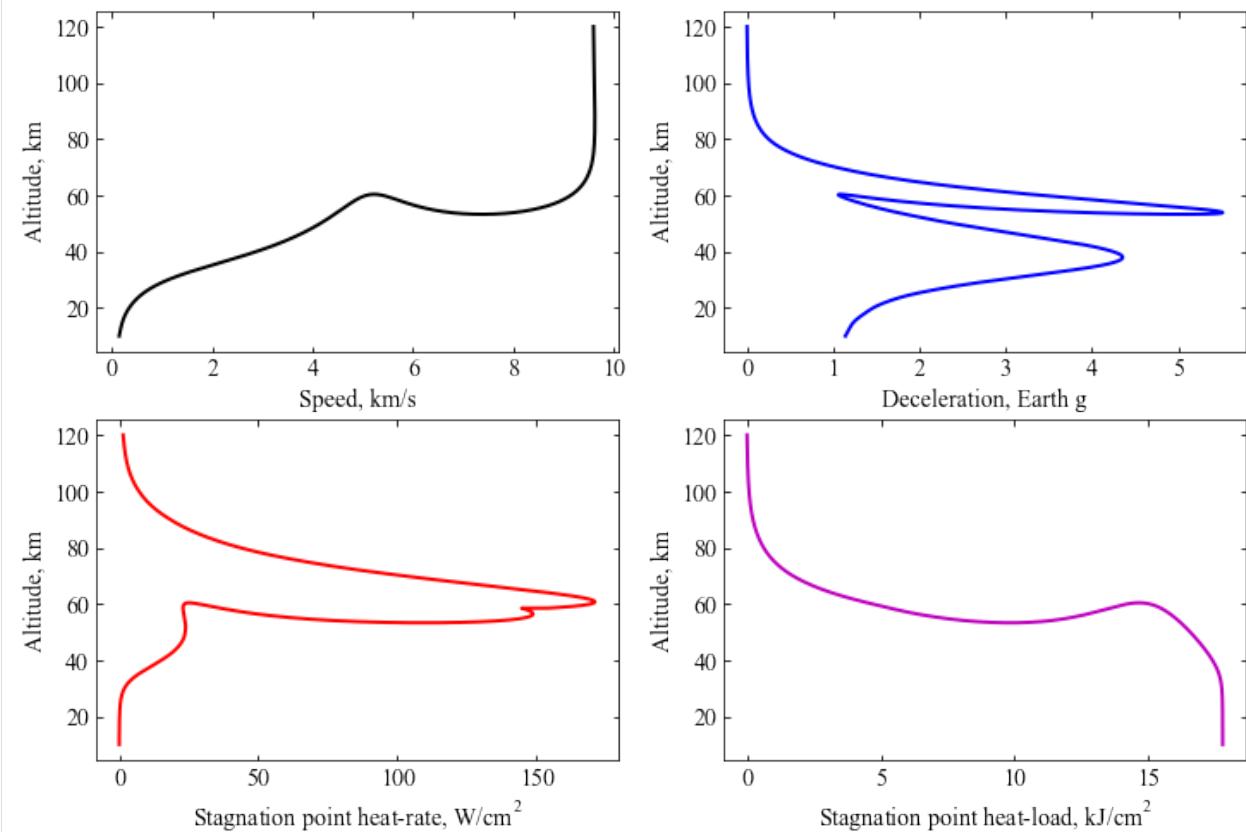
```

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$\text{kJ}/\text{cm}^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/apollo-6-earth.png', bbox_inches='tight')
plt.savefig('../plots/apollo-6-earth.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/apollo-6-earth.eps', dpi=300, bbox_inches='tight')

plt.show()

```



4.19 Example - 19 - PAET - Earth

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the PAET flight test. <https://ntrs.nasa.gov/search.jsp?R=19720045287>

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("EARTH")
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('PAET', 62.1, 69, 0.0, 0.66, 0.0, 0.46, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(90.0, 0.0, 0.0, 6.56, 0.0, -40.8, 0.0, 0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)
```

```
[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0, 0.1, 0.0)
# bank angle = 70 deg arbitrary, actual profile will give more realistic results
```

```
[8]: # import rcParams to set figure font type
from matplotlib import rcParams
```

```
[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc, 'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
```

(continues on next page)

(continued from previous page)

```

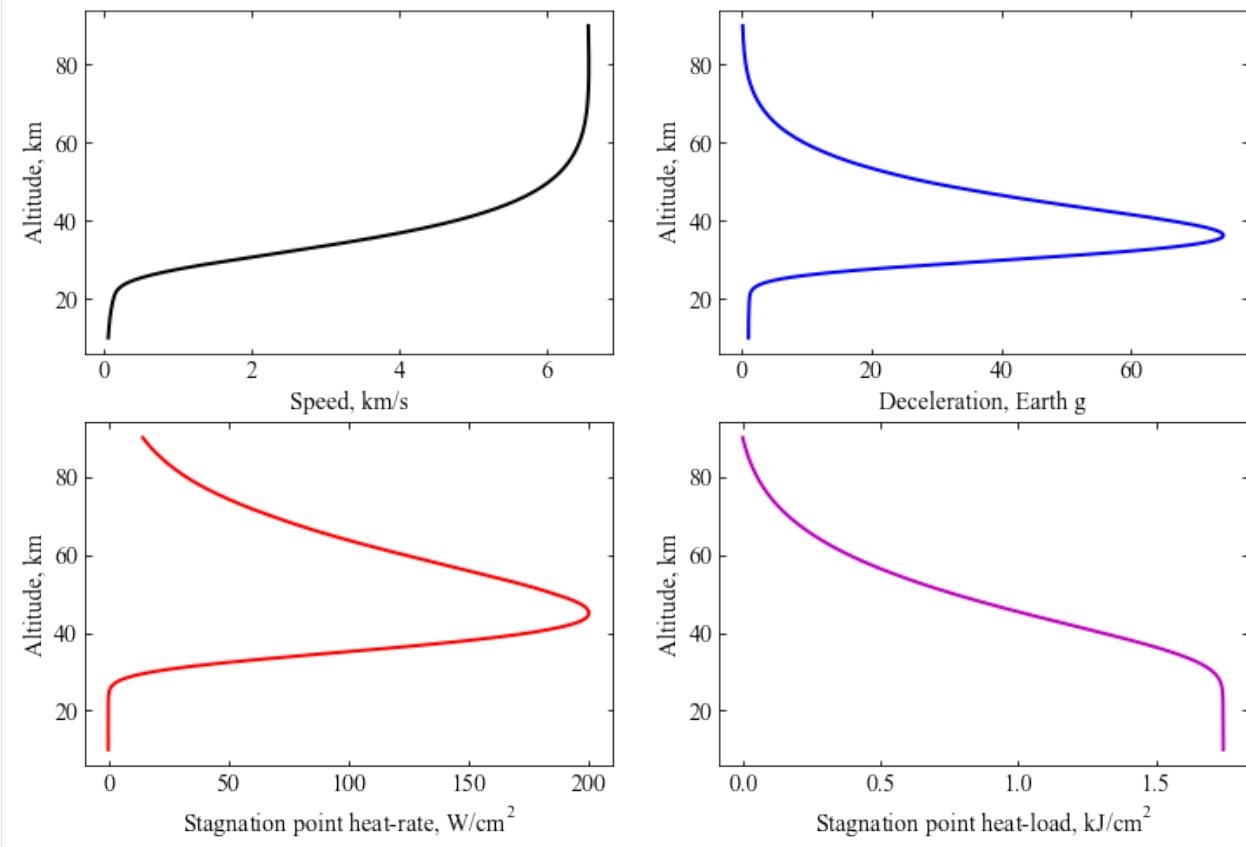
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$\text{kJ}/\text{cm}^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/paet-earth.png', bbox_inches='tight')
plt.savefig('../plots/paet-earth.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/paet-earth.eps', dpi=300, bbox_inches='tight')

plt.show()

```



4.20 Example - 20 - Viking-1 - Mars

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Viking-1 Lander. https://en.wikipedia.org/wiki/Viking_1

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("MARS")
planet.loadAtmosphereModel('../atmdata/Mars/mars-gram-avg.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Viking-1', 980, 63.0, 0.18, 9.65, 0.0, 0.88, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(1200.0, 0.0, 0.0, 4.42, 0.0, -16.99, 0.0, 0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)
```

```
[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0, 0.1, 20.0)
# bank angle = 20 deg arbitrary, actual profile will give more realistic results
```

```
[8]: # import rcParams to set figure font type
from matplotlib import rcParams
```

```
[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

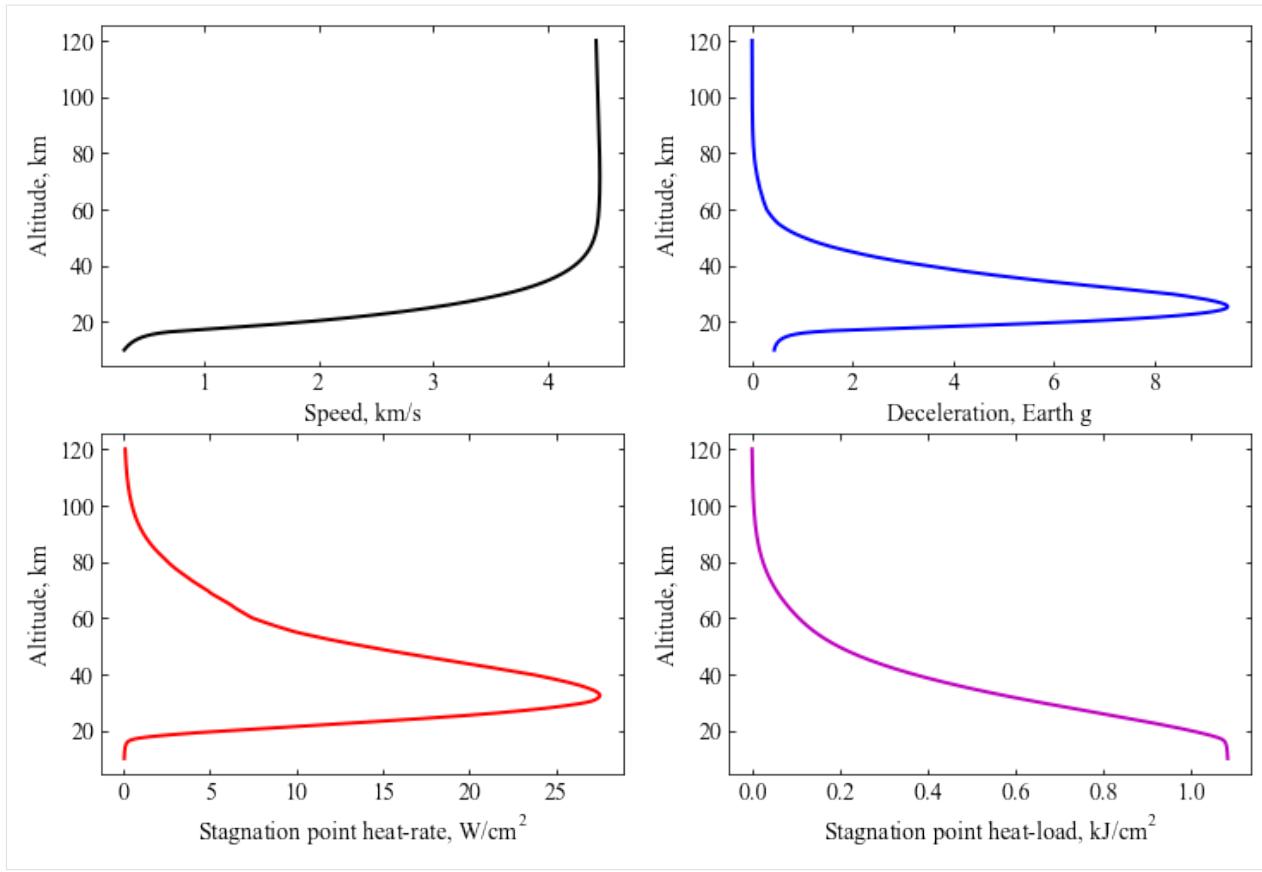
```
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc, 'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.savefig('../plots/viking-1-mars.png',bbox_inches='tight')
plt.savefig('../plots/viking-1-mars.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/viking-1-mars.eps', dpi=300,bbox_inches='tight')

plt.show()
```



[]:

4.21 Example - 21 - Viking-2 - Mars

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Viking-2 Lander. https://en.wikipedia.org/wiki/Viking_2

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("MARS")
planet.loadAtmosphereModel('../atmdata/Mars/mars-gram-avg.dat', 0 , 1 ,2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Viking-2', 982, 63.6, 0.18, 9.65, 0.0, 0.88, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(120.0,0.0,0.0,4.48,0.0,-17.08,0.0,0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0,0.1,20.0)
# bank angle = 20 deg arbitrary, actual profile will give more realistic results

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
```

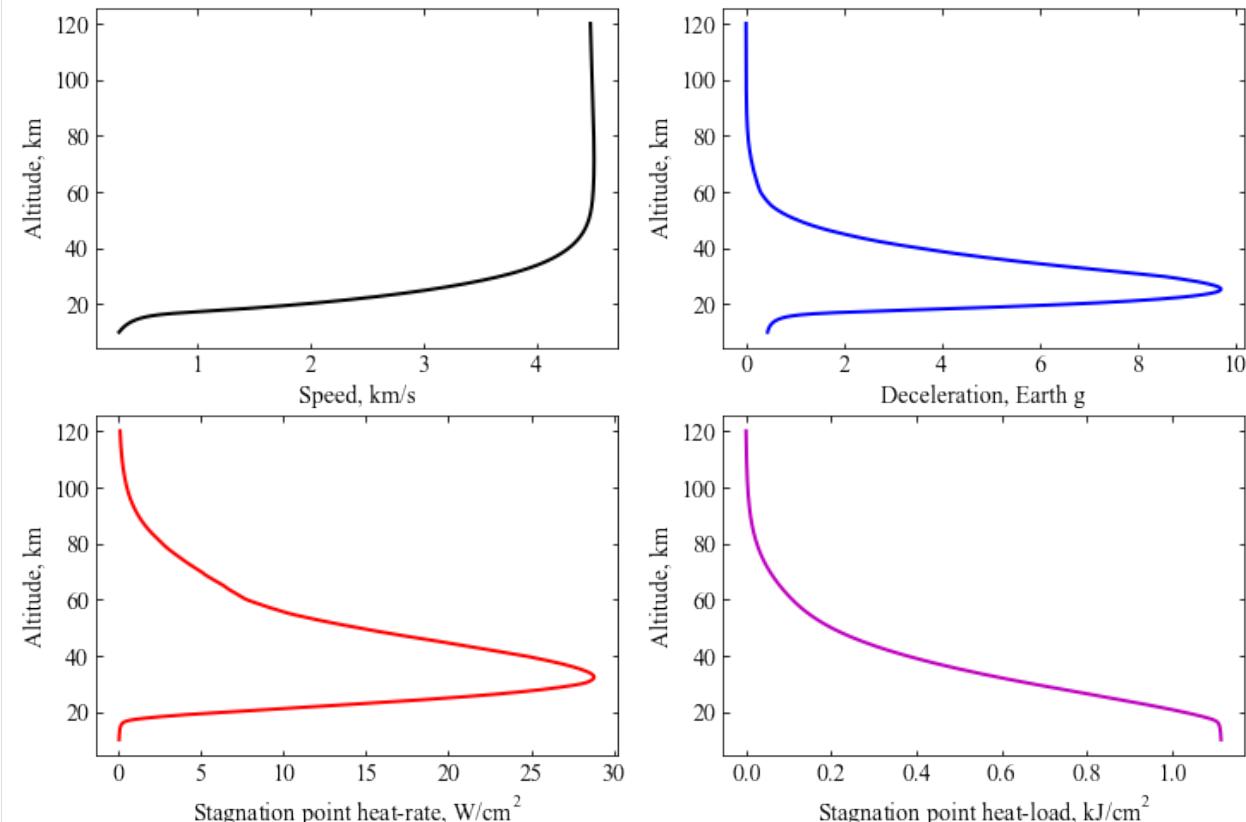
(continues on next page)

(continued from previous page)

```
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)
```

```
plt.savefig('../plots/viking-2-mars.png', bbox_inches='tight')
plt.savefig('../plots/viking-2-mars.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/viking-2-mars.eps', dpi=300, bbox_inches='tight')
```

```
plt.show()
```



```
[ ]:
```

4.22 Example - 22 - PV Small North - Venus

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Pioneer Venus Small North probe. https://en.wikipedia.org/wiki/Pioneer_Venus_Multiprobe

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("VENUS")
planet.loadAtmosphereModel('../atmdata/Venus/venus-gram-avg.dat', 0 , 1 , 2, 3)

[4]: # Set up the vehicle
vehicle=Vehicle('PV-Small-North', 91, 190, 0.00, 0.46, 0.0, 0.19, planet)

[5]: # Set up entry parameters
vehicle.setInitialState(180.0,0.0,0.0,11.54,0.0,-68.7,0.0,0.0)

[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0,0.1,0.0)
# bank angle = 20 deg arbitrary, actual profile will give more realistic results

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

```

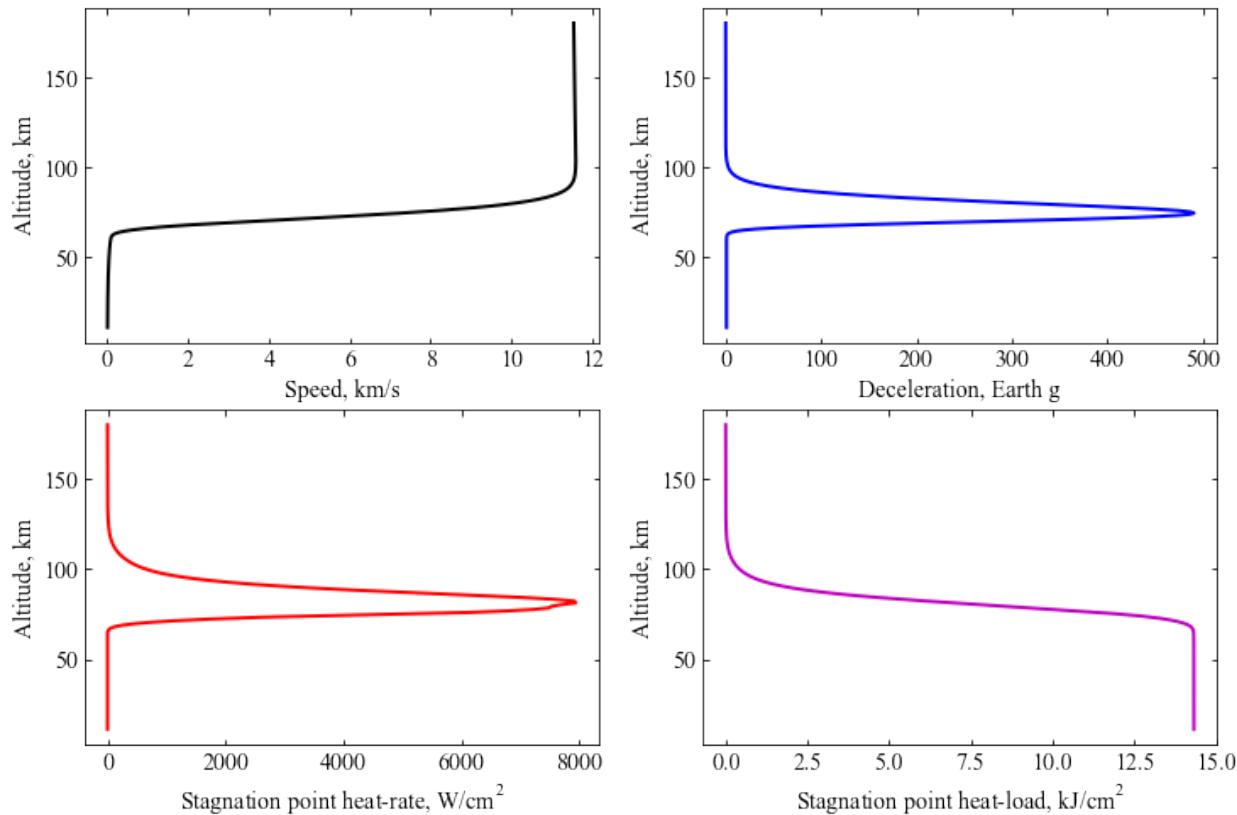
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$\text{kJ}/\text{cm}^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/pv-small-night-venus.png', bbox_inches='tight')
plt.savefig('../plots/pv-small-night-venus.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/pv-small-night-venus.eps', dpi=300, bbox_inches='tight')

plt.show()

```



4.23 Example - 23 - PV Small Night - Venus

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Pioneer Venus Small Night probe. https://en.wikipedia.org/wiki/Pioneer_Venus_Multiprobe

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("VENUS")
planet.loadAtmosphereModel('../atmdata/Venus/venus-gram-avg.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('PV-Small-Night', 91, 190, 0.00, 0.46, 0.0, 0.19, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(180.0,0.0,0.0,11.54,0.0,-41.5,0.0,0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)
```

```
[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0,0.1,0.0)
```

```
[8]: # import rcParams to set figure font type
from matplotlib import rcParams
```

```
[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
```

(continues on next page)

(continued from previous page)

```

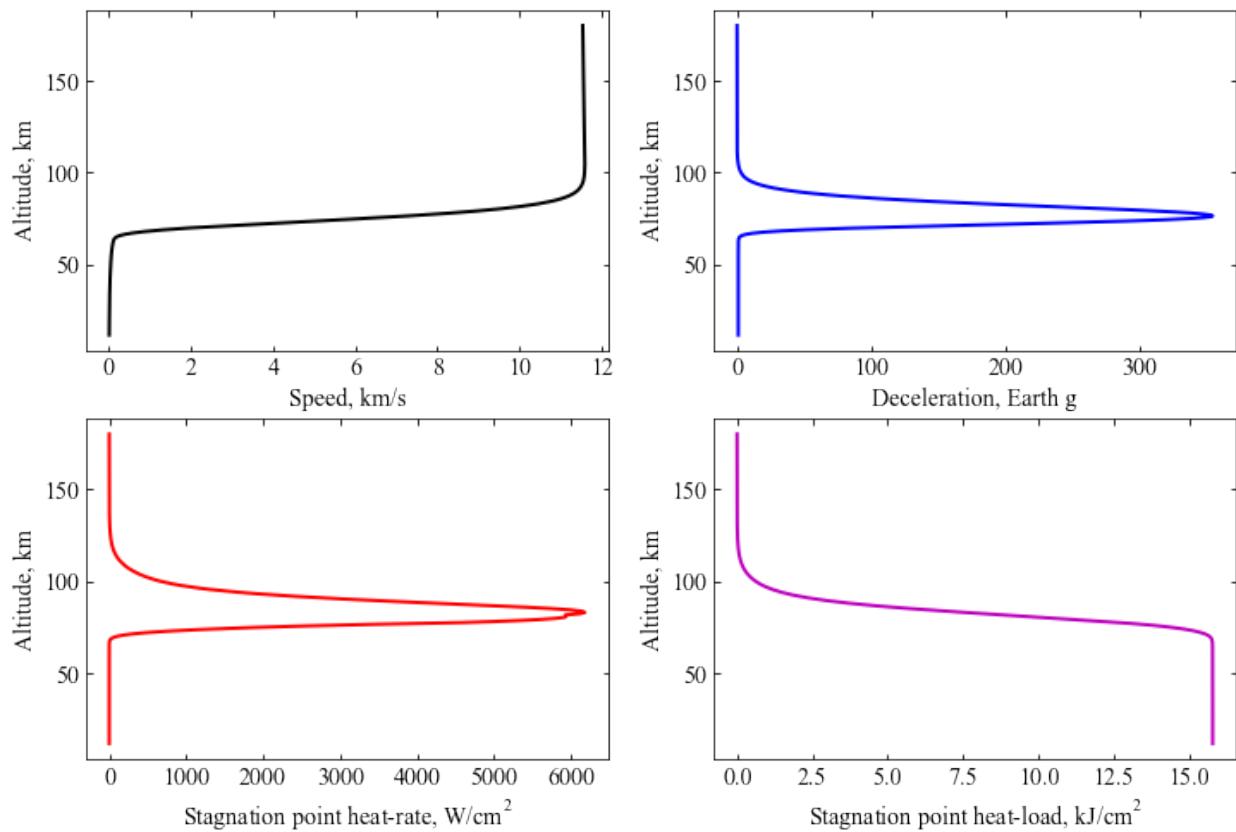
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$\text{kJ/cm}^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/pv-small-night-venus.png', bbox_inches='tight')
plt.savefig('../plots/pv-small-night-venus.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/pv-small-night-venus.eps', dpi=300, bbox_inches='tight')

plt.show()

```



[]:

4.24 Example - 24 - PV Small Day - Venus

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Pioneer Venus Small Day probe. https://en.wikipedia.org/wiki/Pioneer_Venus_Multiprobe

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("VENUS")
planet.loadAtmosphereModel('../atmdata/Venus/venus-gram-avg.dat', 0 , 1 , 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('PV-Small-Day', 91, 190, 0.00, 0.46, 0.0, 0.19, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(180.0,0.0,0.0,11.54,0.0,-25.4,0.0,0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)
```

```
[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0,0.1,0.0)
```

```
[8]: # import rcParams to set figure font type
from matplotlib import rcParams
```

```
[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g',fontsize=14)
```

(continues on next page)

(continued from previous page)

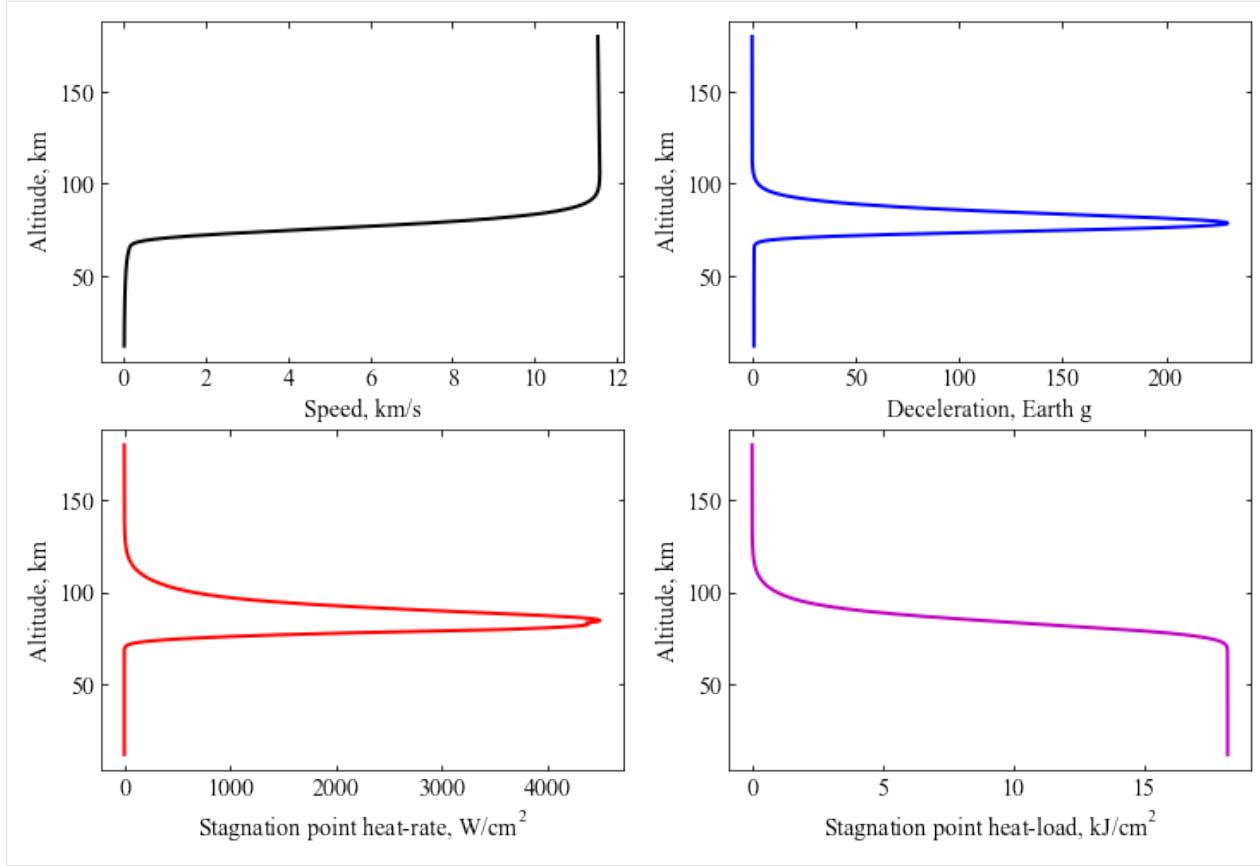
```
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc, 'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/pv-small-day-venus.png', bbox_inches='tight')
plt.savefig('../plots/pv-small-day-venus.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/pv-small-day-venus.eps', dpi=300, bbox_inches='tight')

plt.show()
```



[]:

4.25 Example - 25 - PV Large - Venus

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Pioneer Venus Large probe. https://en.wikipedia.org/wiki/Pioneer_Venus_Multiprobe

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("VENUS")
planet.loadAtmosphereModel('../atmdata/Venus/venus-gram-avg.dat', 0 , 1 , 2 , 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('PV-Large', 316 , 188, 0.00, 1.59, 0.0, 0.36, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(180.0,0.0,0.0,11.54,0.0,-32.4,0.0,0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

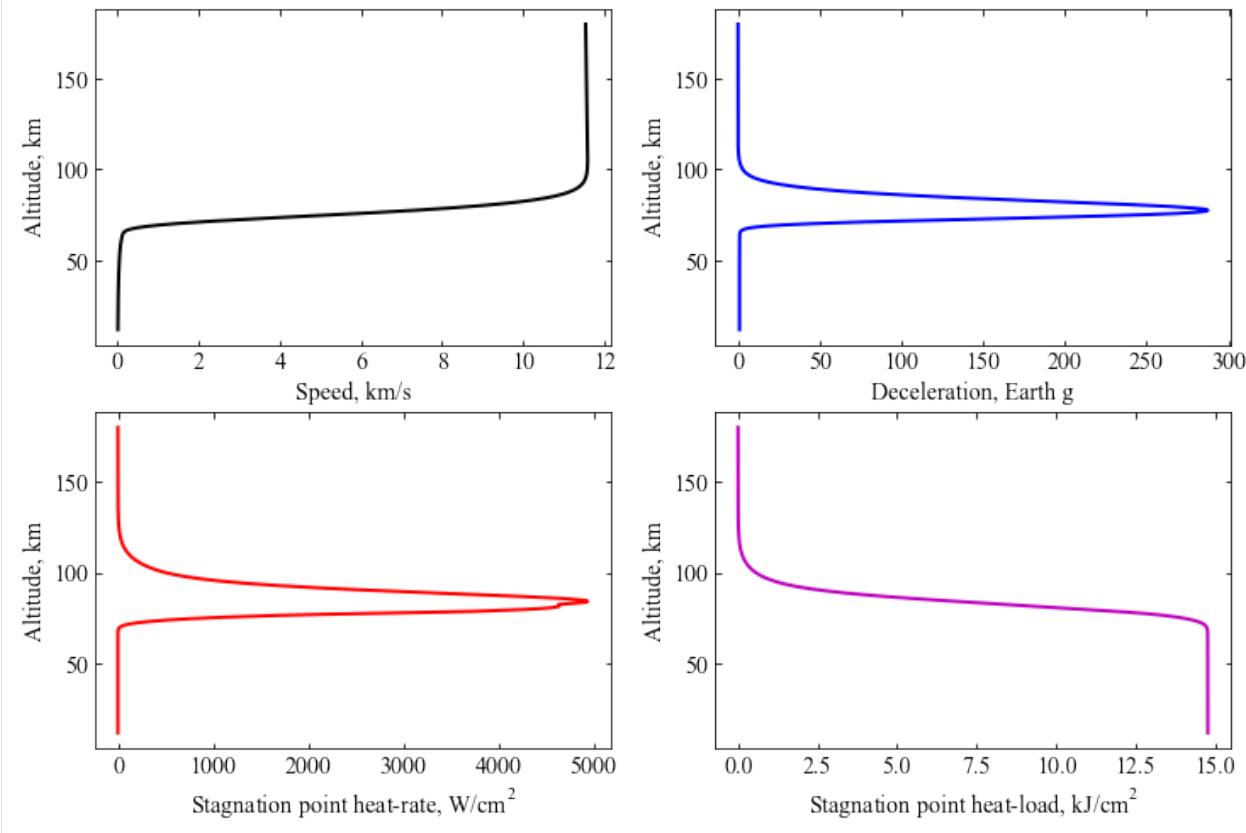
```

ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/pv-large-venus.png', bbox_inches='tight')
plt.savefig('../plots/pv-large-venus.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/pv-large-venus.eps', dpi=300, bbox_inches='tight')

plt.show()

```



4.26 Example - 26 - Galileo - Jupiter

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Galileo entry probe. https://en.wikipedia.org/wiki/Galileo_Probe

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("JUPITER")
planet.h_trap = -130.0E3
planet.loadAtmosphereModel('../atmdata/Jupiter/jupiter-galileo-asi.dat', 0, 1, 2, 3,
                           heightInKmFlag=True)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Galileo', 335.0 , 256.0, 0.0, 1.26, 0.0, 0.22, planet)

[5]: # Set up entry parameters
vehicle.setInitialState(450.0,0.0,0.0,47.37,0.0,-8.5,0.0,0.0)

[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (9400.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
```

(continues on next page)

(continued from previous page)

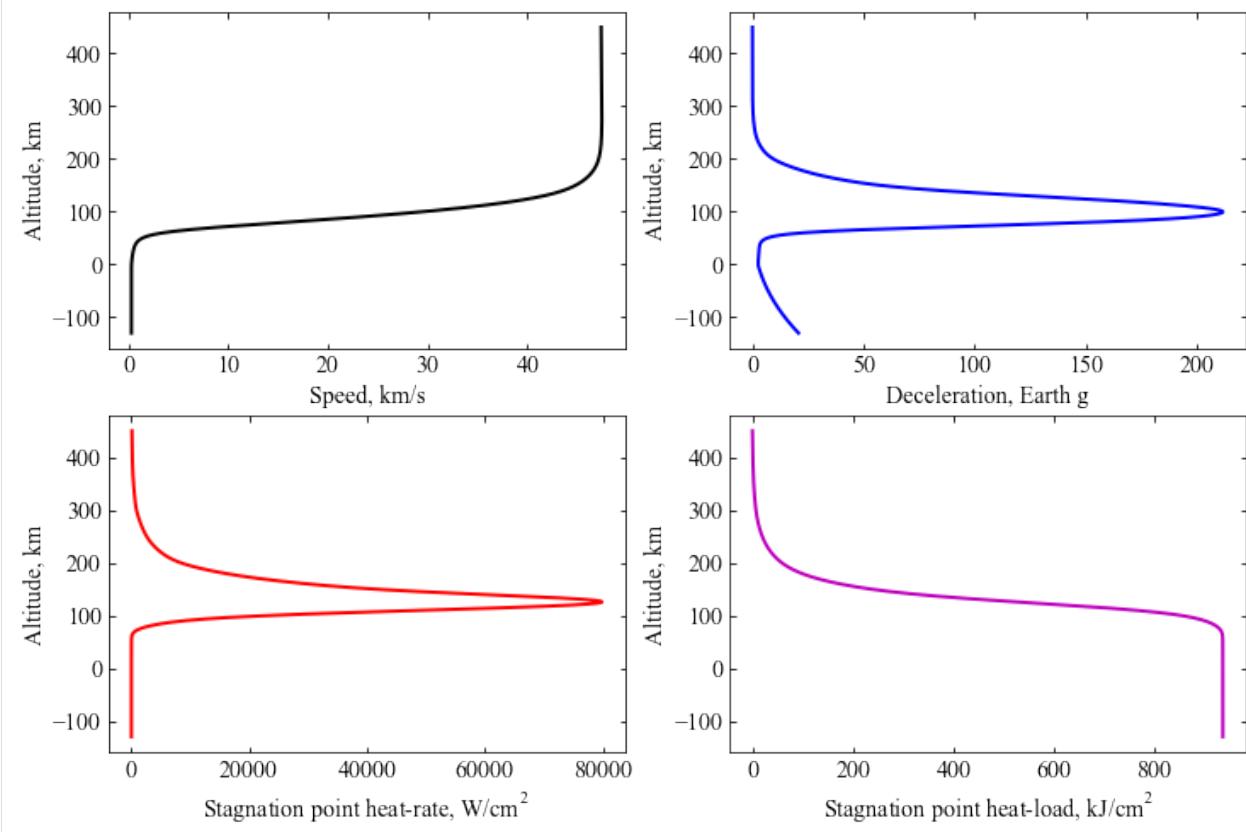
```

plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$\text{kJ}/\text{cm}^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/galileo-jupiter.png', bbox_inches='tight')
plt.savefig('../plots/galileo-jupiter.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/galileo-jupiter.eps', dpi=300, bbox_inches='tight')

plt.show()

```



Results are off because presumably because of poor validity of heating correlations under extreme entry conditions at Jupiter.

4.27 Example - 27 - OREX - Earth

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the OREX entry probe. <http://adsabs.harvard.edu/full/1995ESASP.367..271I>

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("EARTH")
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('OREX', 761, 56, 0.00, 9.08, 0.0, 1.35, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(120.0,0.0,0.0,7.43,0.0,-3.17,0.0,0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)
```

```
[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0,0.1,0.0)
```

```
[8]: # import rcParams to set figure font type
from matplotlib import rcParams
```

```
[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
```

(continues on next page)

(continued from previous page)

```

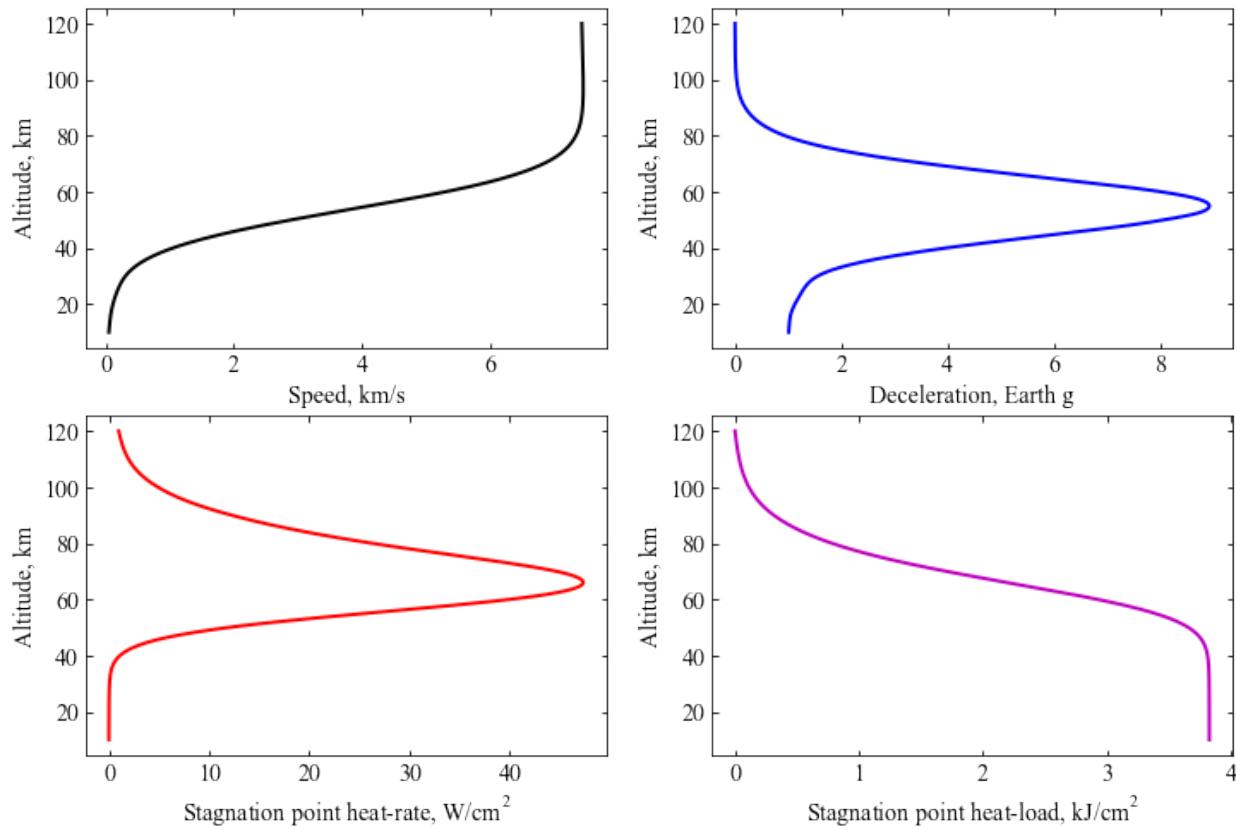
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$\text{kJ}/\text{cm}^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/orex-earth.png', bbox_inches='tight')
plt.savefig('../plots/orex-earth.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/orex-earth.eps', dpi=300, bbox_inches='tight')

plt.show()

```



4.28 Example - 28 - Pathfinder - Mars

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle

[2]: import numpy as np
import matplotlib.pyplot as plt

This notebook simulates the atmospheric entry of the Pathfinder mission. https://en.wikipedia.org/wiki/Mars\_Pathfinder

[3]: # Set up the planet and atmosphere model.
planet=Planet("MARS")
planet.loadAtmosphereModel('../atmdata/Mars/mars-gram-avg.dat', 0, 1, 2, 3)

[4]: # Set up the vehicle
vehicle=Vehicle('Pathfinder', 586, 62.0, 0.0, 5.52, 0.0, 0.66, planet)

[5]: # Set up entry parameters
vehicle.setInitialState(120.0, 0.0, 0.0, 7.48, 0.0, -14.06, 0.0, 0.0)

[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0, 0.1, 0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

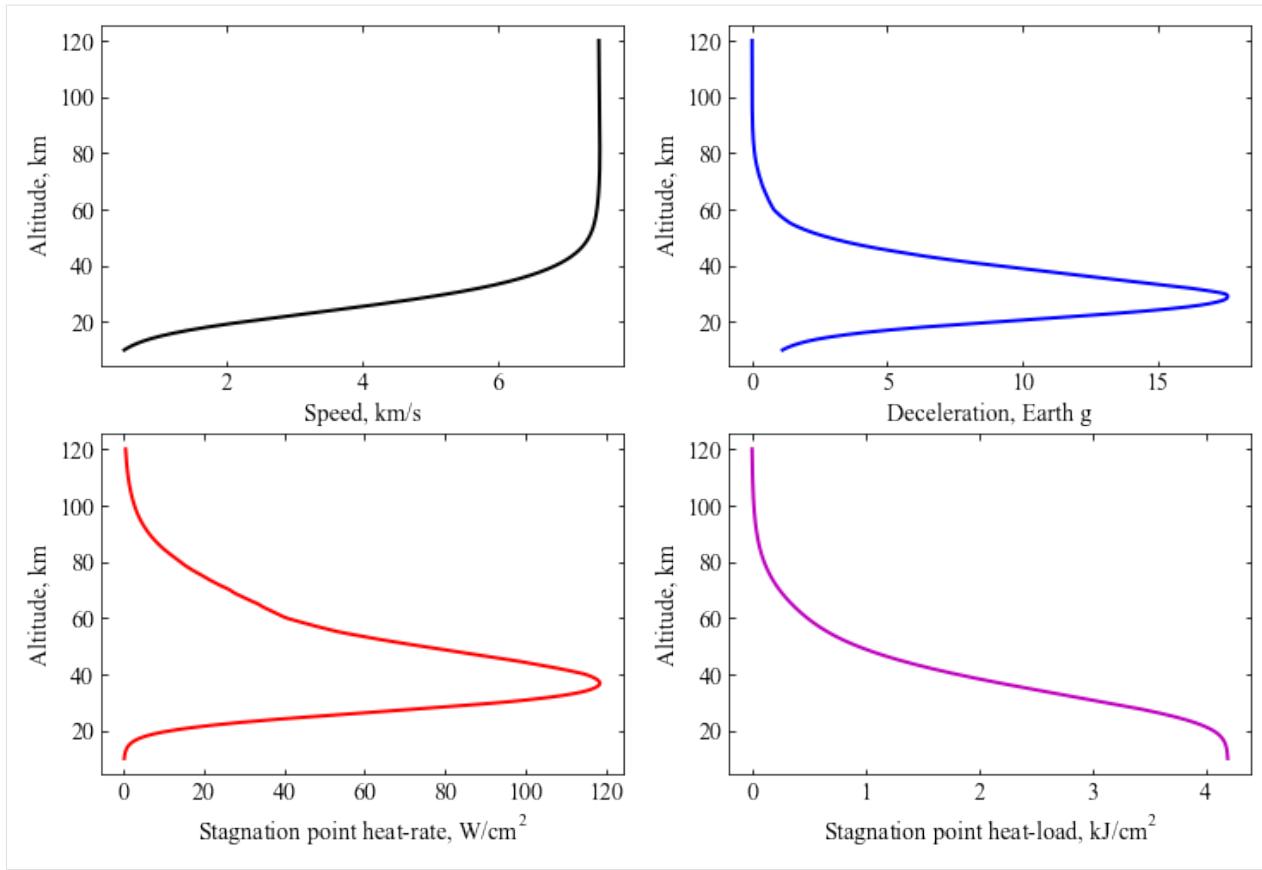
```
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc, 'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.savefig('../plots/pathfinder-mars.png',bbox_inches='tight')
plt.savefig('../plots/pathfinder-mars.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/pathfinder-mars.eps', dpi=300,bbox_inches='tight')

plt.show()
```



4.29 Example - 29 - Mirka - Earth

```
[1]: from AMAT.planet import Planet
      from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
      import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Mirka entry probe. <http://adsabs.harvard.edu/full/2006ESASP.631E..26B>

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("EARTH")
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0 , 1 , 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Mirka', 154.0 , 214.9, 0.00, 0.785, 0.0, 0.5, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(120.0,0.0,0.0,7.60,0.0,-2.51,0.0,0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)
```

```
[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (2400.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

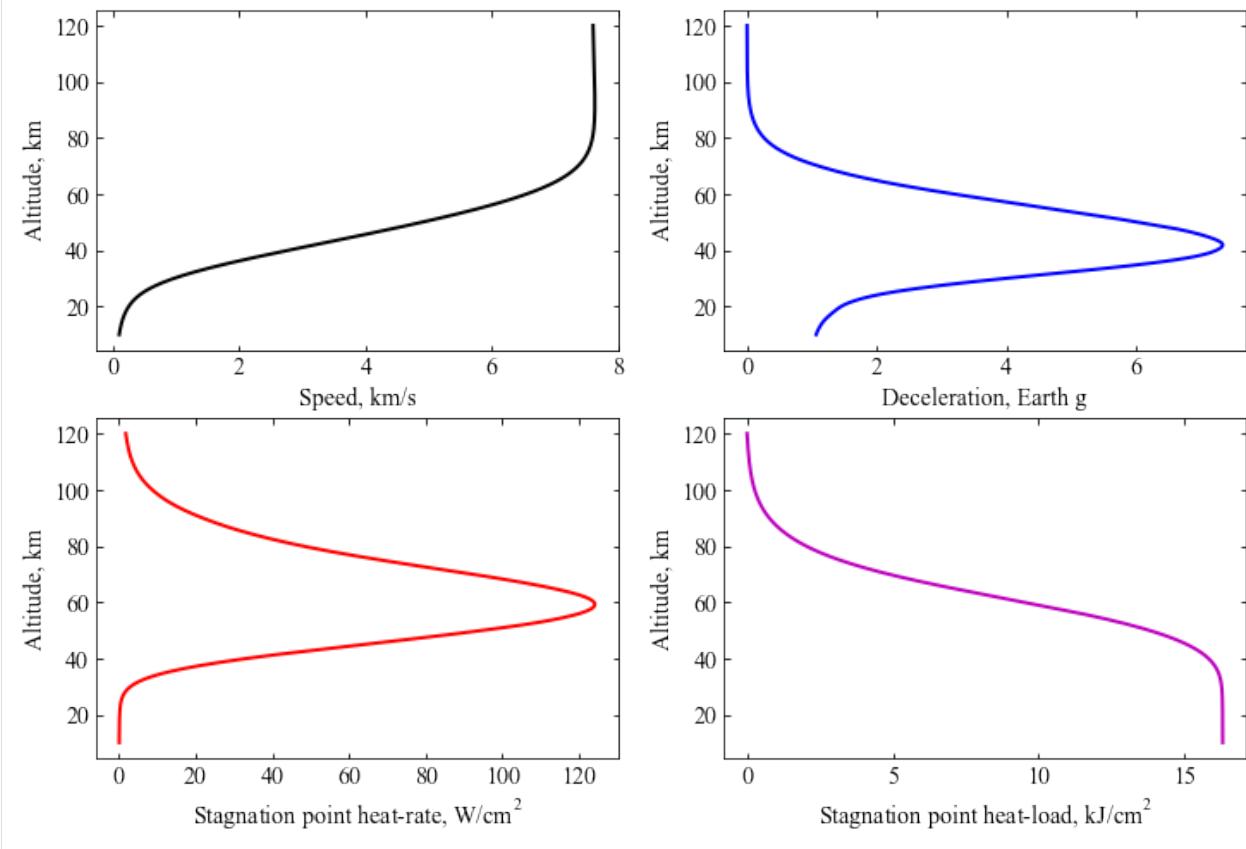
plt.savefig('../plots/mirka-earth.png',bbox_inches='tight')
```

(continues on next page)

(continued from previous page)

```
plt.savefig('../plots/mirka-earth.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/mirka-earth.eps', dpi=300, bbox_inches='tight')

plt.show()
```



4.30 Example - 30 - Huygens - Titan

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Huygens probe. [https://en.wikipedia.org/wiki/Huygens_\(spacecraft\)](https://en.wikipedia.org/wiki/Huygens_(spacecraft))

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("TITAN")
planet.h_skip = 1270.0E3
planet.loadAtmosphereModel('../atmdata/Titan/titan-gram-avg.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Huygens', 318.0, 35.0, 0.00, 5.73, 0.0, 1.25, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(1270.0,0.0,0.0,6.0,0.0,-65.4,0.0,0.0)

[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (120*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
```

(continues on next page)

(continued from previous page)

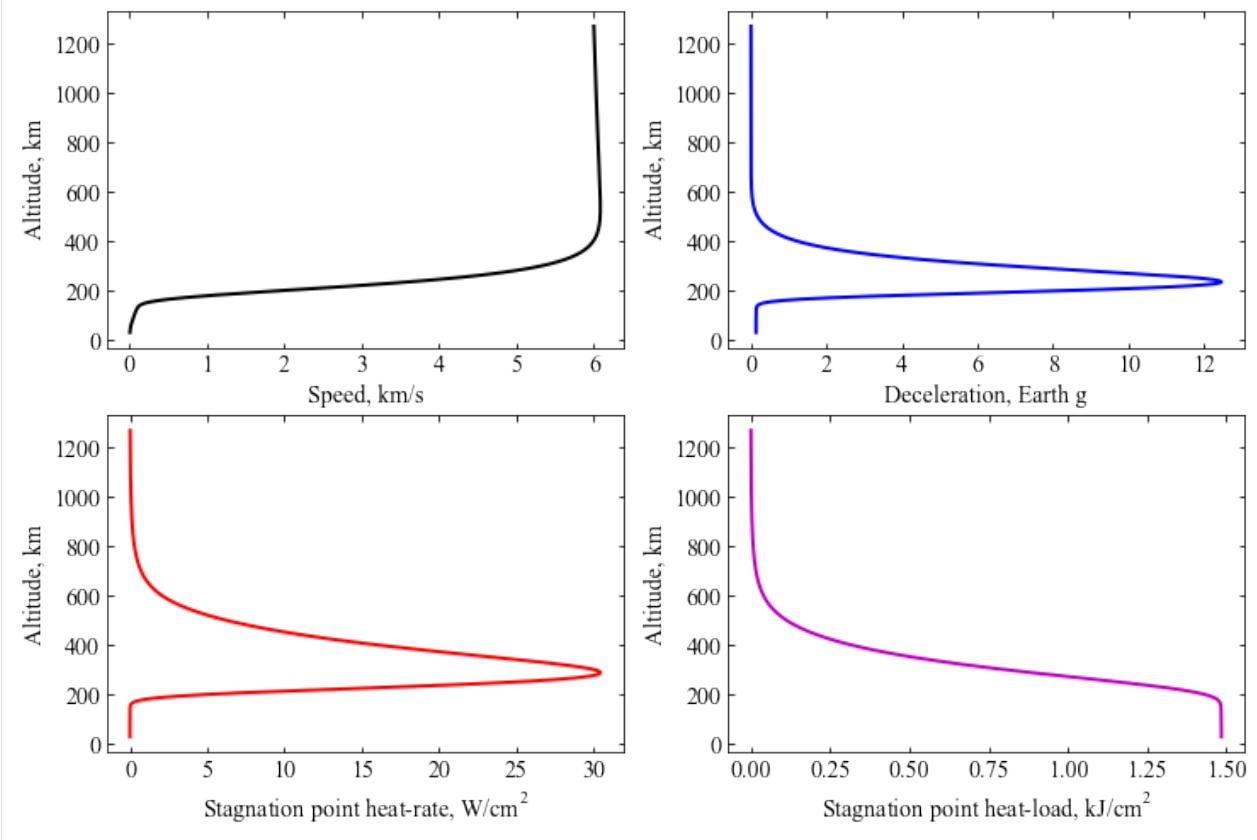
```

ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/huygens-titan.png', bbox_inches='tight')
plt.savefig('../plots/huygens-titan.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/huygens-titan.eps', dpi=300, bbox_inches='tight')

plt.show()

```



Radiative heating is neglected in these calculations.

4.31 Example - 31 - Atm. Reentry Demonstrator - Earth

```

[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle

[2]: import numpy as np
import matplotlib.pyplot as plt

```

This notebook simulates the atmospheric entry of the ESA Atmospheric Reentry Demonstrator. https://en.wikipedia.org/wiki/Atmospheric_Reentry_Demonstrator

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("EARTH")
planet.h_skip = 120.0E3
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0 , 1 , 2, 3)

[4]: # Set up the vehicle
vehicle=Vehicle('ARD', 2715.0 , 403.0, 0.00, 6.15, 0.0, 3.36, planet)

[5]: # Set up entry parameters
vehicle.setInitialState(120.0,0.0,0.0,7.54,0.0,-2.6,0.0,0.0)

[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (30*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

```

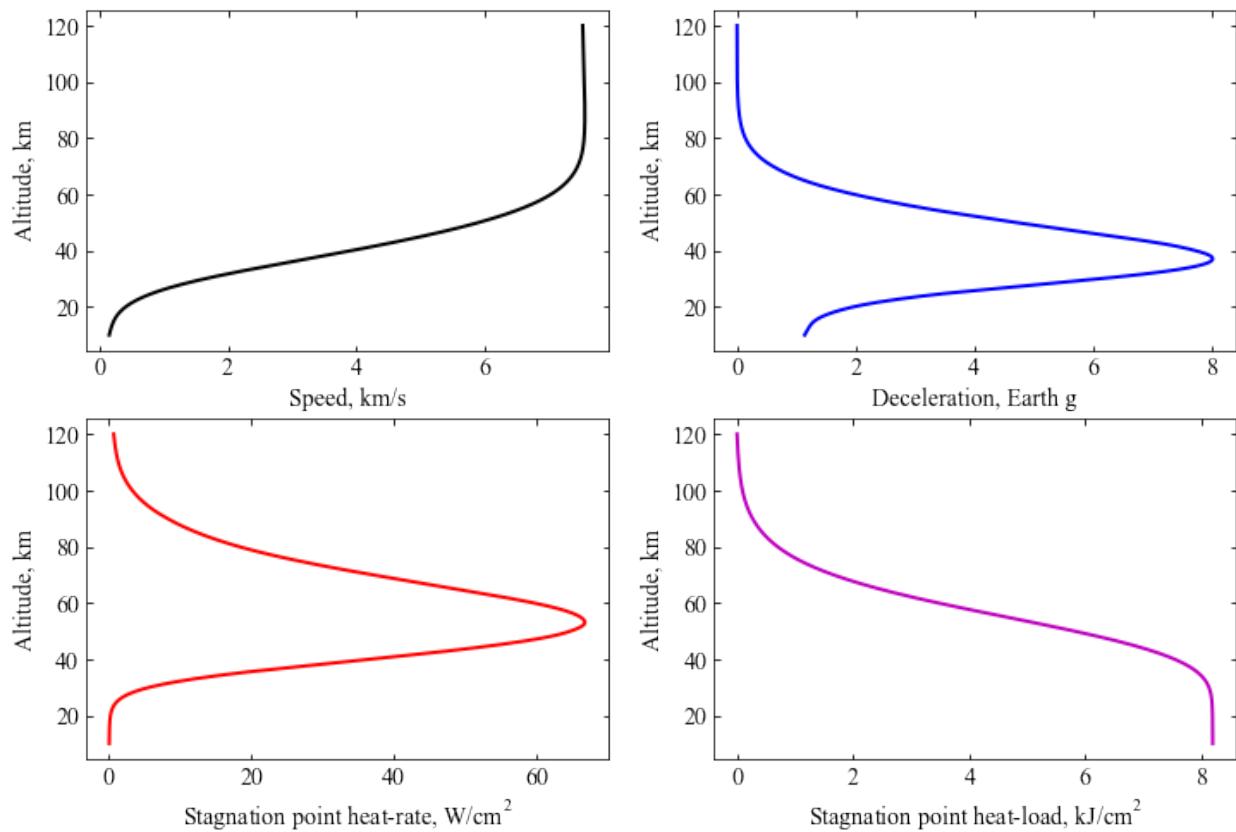
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$\text{kJ}/\text{cm}^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/ard-earth.png', bbox_inches='tight')
plt.savefig('../plots/ard-earth.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/ard-earth.eps', dpi=300, bbox_inches='tight')

plt.show()

```



4.32 Example - 32 - Deep Space 2 - Mars

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle

[2]: import numpy as np
import matplotlib.pyplot as plt

This notebook simulates the atmospheric entry of the Deep Space 2 aeroshells. https://en.wikipedia.org/wiki/Deep\_Space\_2

[3]: # Set up the planet and atmosphere model.
planet=Planet("MARS")
planet.h_skip = 128.0E3
planet.loadAtmosphereModel('../atmdata/Mars/mars-gram-avg.dat', 0 , 1 , 2, 3)

[4]: # Set up the vehicle
vehicle=Vehicle('DS2', 3.67, 36.2, 0.00, 0.096, 0.0, 0.0875, planet)

[5]: # Set up entry parameters
vehicle.setInitialState(128.0,0.0,0.0,6.90,0.0,-13.25,0.0,0.0)

[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (30*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

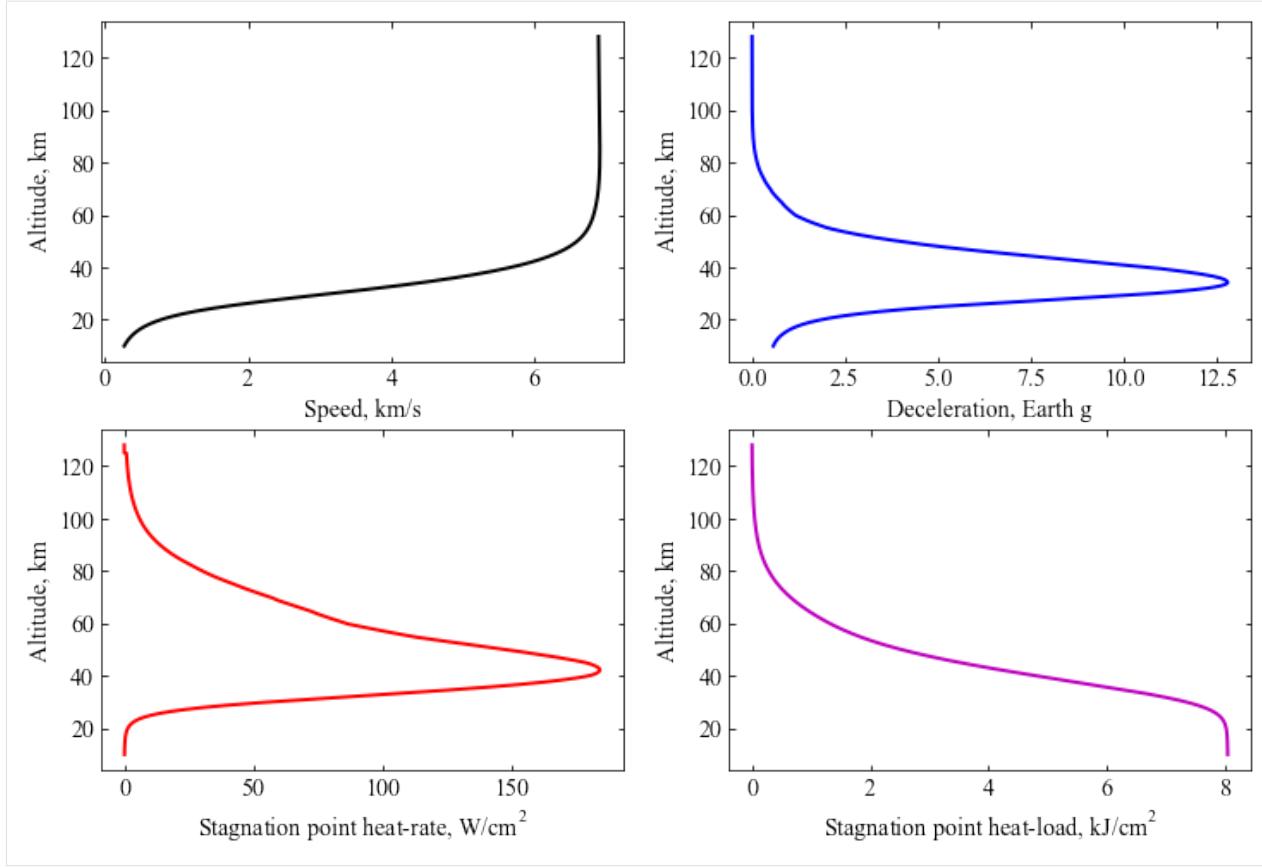
```
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc, 'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.savefig('../plots/deep-space-2-mars.png',bbox_inches='tight')
plt.savefig('../plots/deep-space-2-mars.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/deep-space-2-mars.eps', dpi=300,bbox_inches='tight')

plt.show()
```



[]:

4.33 Example - 33 - Stardust - Earth

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Stardust sample return aeroshell. [https://en.wikipedia.org/wiki/Stardust_\(spacecraft\)](https://en.wikipedia.org/wiki/Stardust_(spacecraft))

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("EARTH")
planet.h_skip = 125.0E3
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0 , 1 , 2 , 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Stardust', 45.8, 60.0, 0.00, 0.52, 0.0, 0.23, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(125.0,0.0,0.0,12.6,0.0,-8.2,0.0,0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (30*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

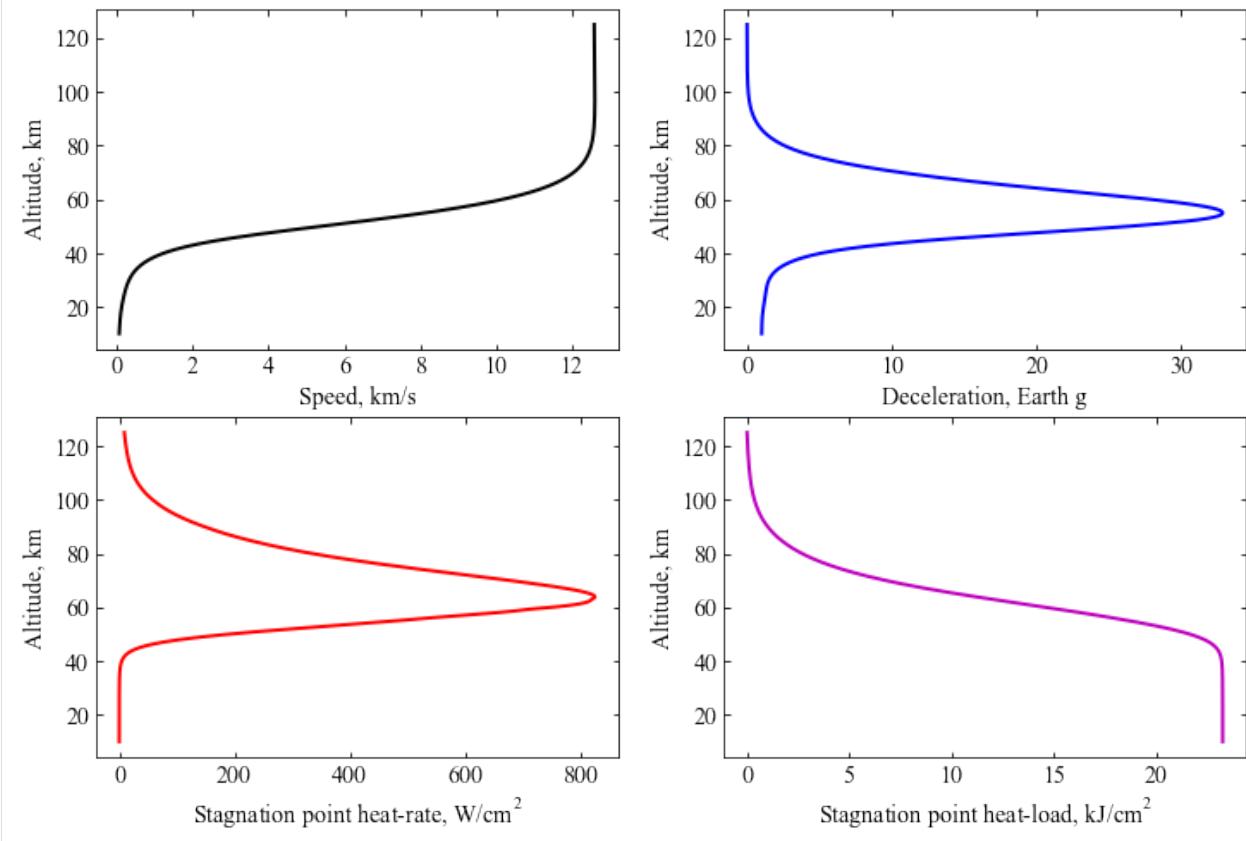
```

ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/stardust-earth.png', bbox_inches='tight')
plt.savefig('../plots/stardust-earth.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/stardust-earth.eps', dpi=300, bbox_inches='tight')

plt.show()

```



4.34 Example - 34 - Genesis - Earth

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Genesis sample return aeroshell. [https://en.wikipedia.org/wiki/Genesis_\(spacecraft\)](https://en.wikipedia.org/wiki/Genesis_(spacecraft))

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("EARTH")
planet.h_skip = 125.0E3
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Genesis', 210, 80, 0.00, 1.78, 0.0, 0.43, planet)

[5]: # Set up entry parameters
vehicle.setInitialState(125.0,0.0,0.0,10.8,0.0,-8.0,0.0,0.0)

[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (30*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
```

(continues on next page)

(continued from previous page)

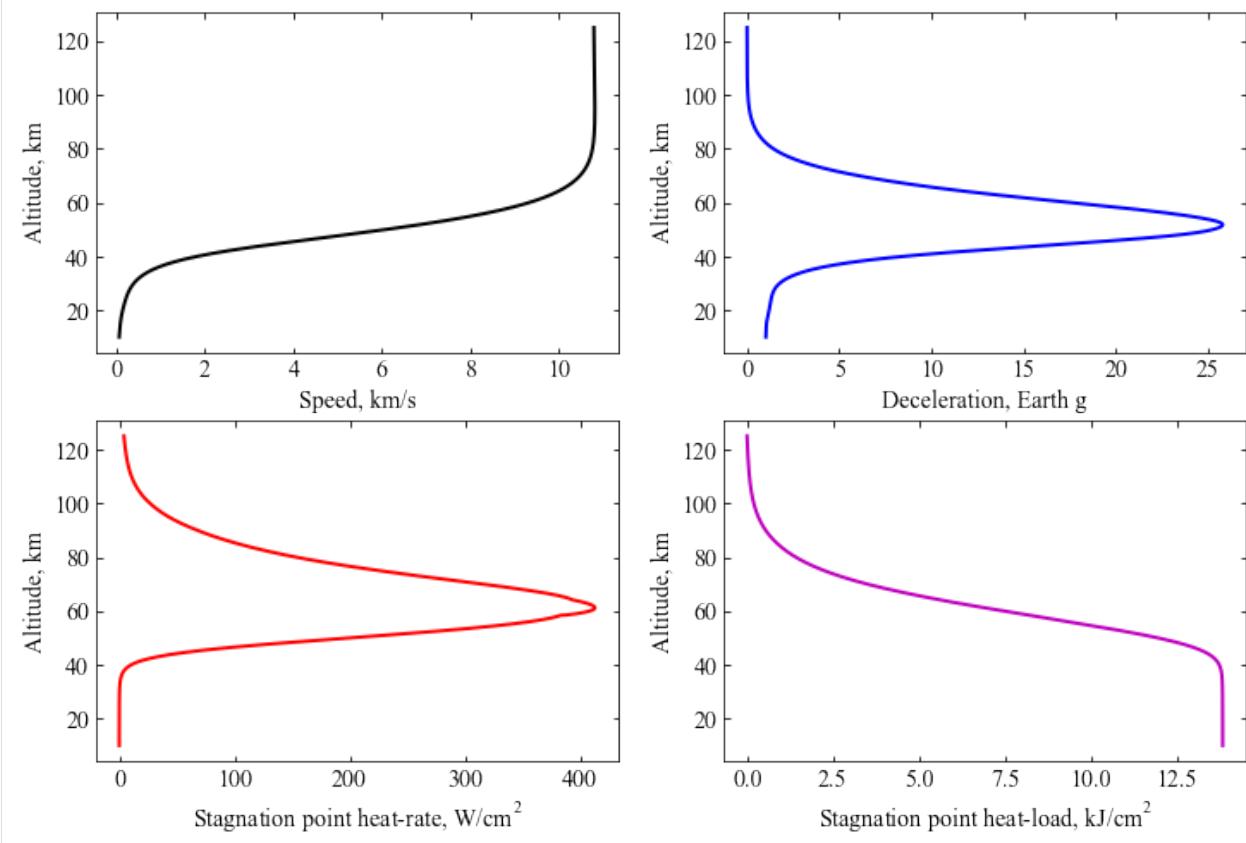
```

plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$\text{kJ}/\text{cm}^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/genesis-earth.png', bbox_inches='tight')
plt.savefig('../plots/genesis-earth.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/genesis-earth.eps', dpi=300, bbox_inches='tight')

plt.show()

```



4.35 Example - 35 - Hayabusa - Earth

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Hayabusa sample return aeroshell. [https://en.wikipedia.org/wiki/Hayabusa_\(spacecraft\)](https://en.wikipedia.org/wiki/Hayabusa_(spacecraft))

Hayabusa

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("EARTH")
planet.h_skip = 120.0E3
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0, 1, 2, 3)

[4]: # Set up the vehicle
vehicle=Vehicle('Hayabusa', 16.27, 114.0, 0.00, 0.128, 0.0, 0.202, planet)

[5]: # Set up entry parameters
vehicle.setInitialState(120.0,0.0,0.0,11.3,0.0,-13.8,0.0,0.0)

[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (30*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

```

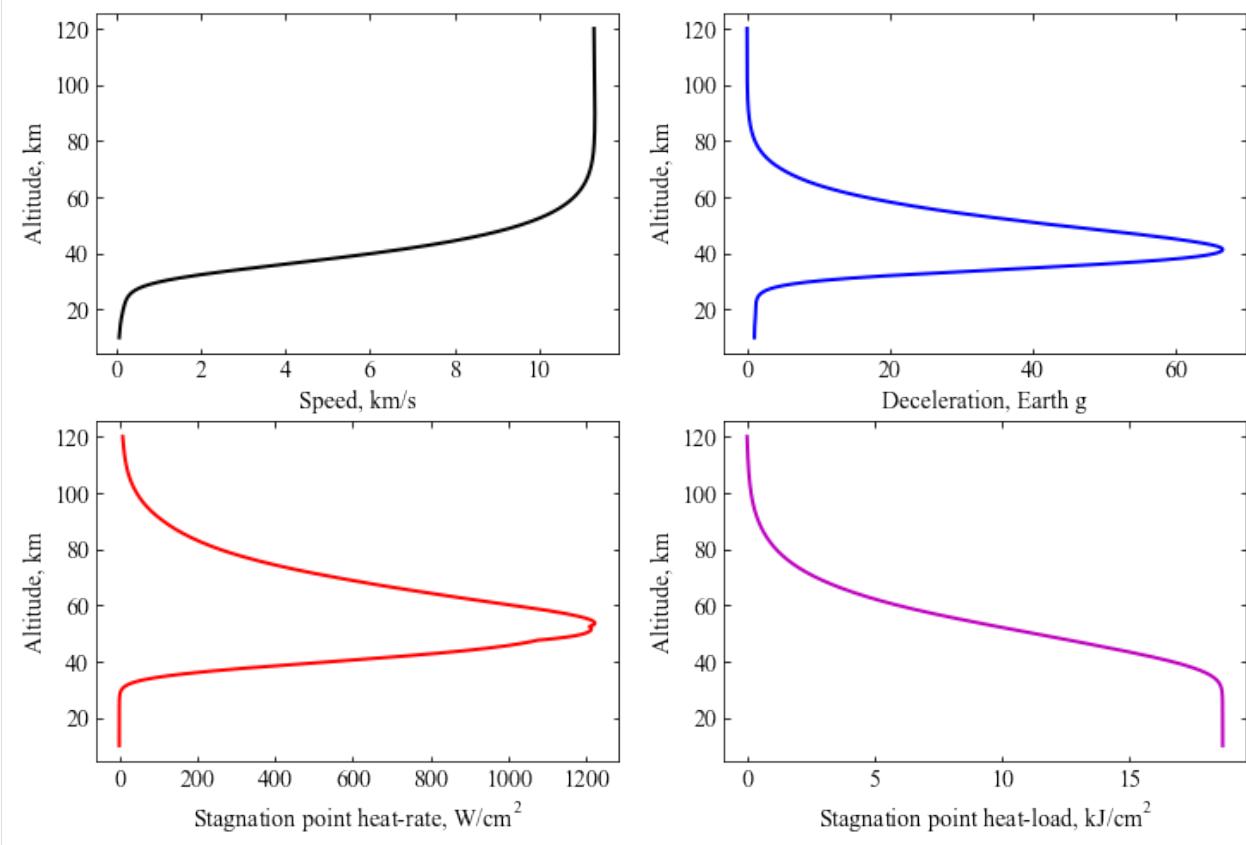
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$\text{kJ/cm}^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.savefig('../plots/hayabusa-earth.png',bbox_inches='tight')
plt.savefig('../plots/hayabusa-earth.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/hayabusa-earth.eps', dpi=300,bbox_inches='tight')

plt.show()

```



4.36 Example - 36 - Beagle-2 - Mars

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Beagle 2 aeroshell https://en.wikipedia.org/wiki/Beagle_2

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("MARS")
planet.h_skip = 120.0E3
planet.loadAtmosphereModel('../atmdata/Mars/mars-gram-avg.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Beagle-2', 68.46, 70.0, 0.00, 0.67, 0.0, 0.417, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(120.0, 0.0, 0.0, 5.40, 0.0, -15.8, 0.0, 0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)
```

```
[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (30*60.0, 0.1, 0.0)
```

```
[8]: # import rcParams to set figure font type
from matplotlib import rcParams
```

```
[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

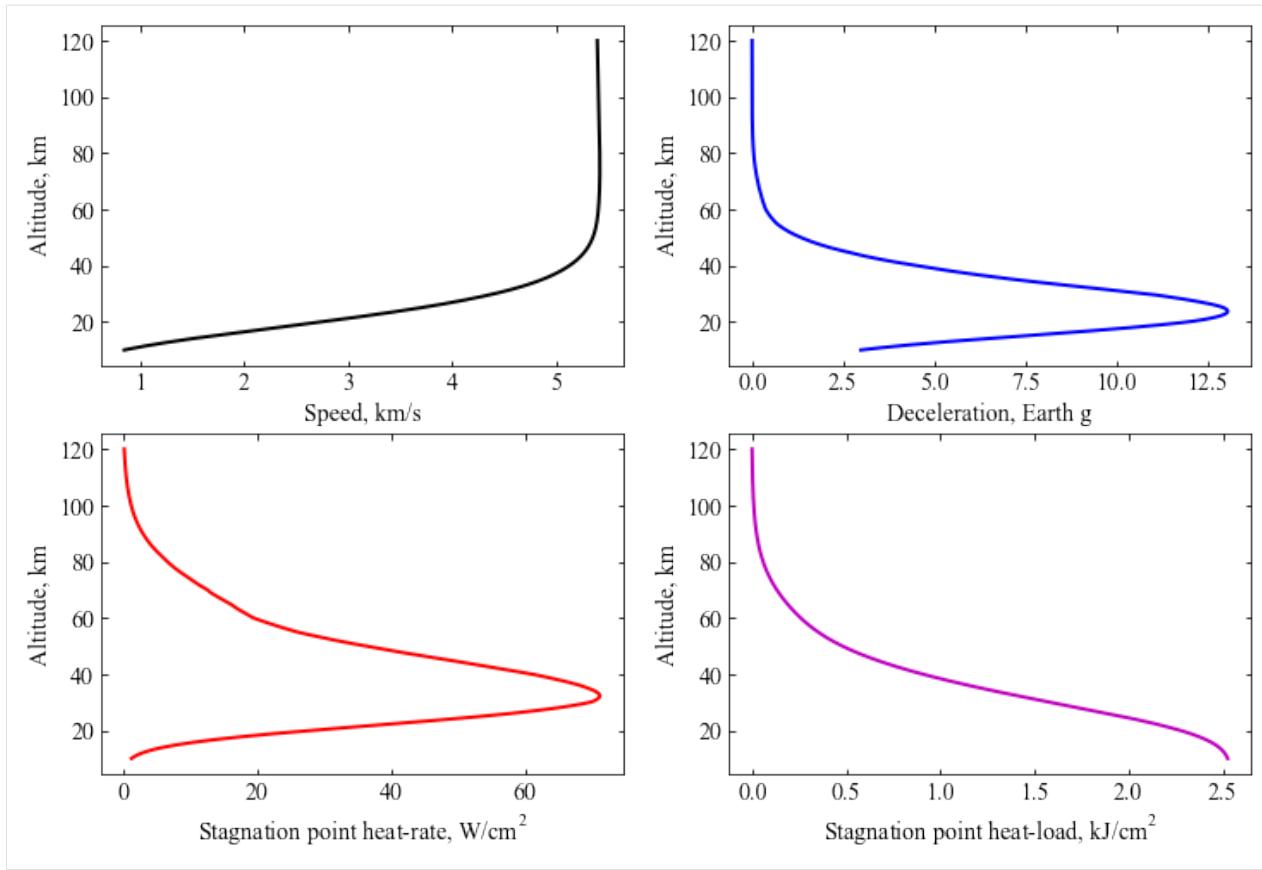
```
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc, 'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.savefig('../plots/beagle-2-mars.png',bbox_inches='tight')
plt.savefig('../plots/beagle-2-mars.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/beagle-2-mars.eps', dpi=300,bbox_inches='tight')

plt.show()
```



4.37 Example - 37 - Opportunity - Mars

```
[1]: from AMAT.planet import Planet
      from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
      import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Opportunity aeroshell (MER-B) [https://en.wikipedia.org/wiki/Opportunity_\(rover\)](https://en.wikipedia.org/wiki/Opportunity_(rover))

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("MARS")
planet.h_skip = 125.0E3
planet.loadAtmosphereModel('../atmdata/Mars/mars-gram-avg.dat', 0 , 1 , 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Opportunity', 836, 88.0, 0.00, 5.52, 0.0, 0.66, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(125.0,0.0,0.0,5.55,0.0,-11.5,0.0,0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (30*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

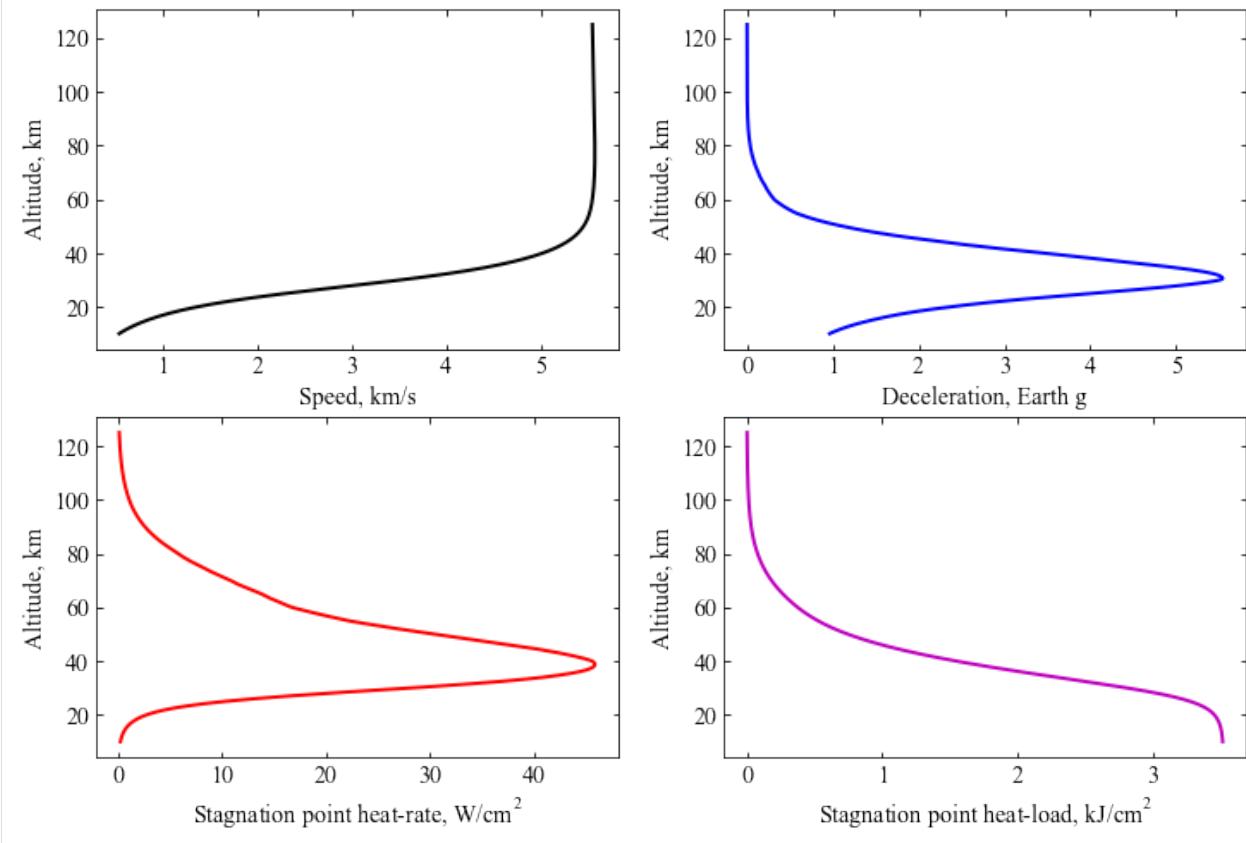
```

ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/opportunity-mars.png', bbox_inches='tight')
plt.savefig('../plots/opportunity-mars.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/opportunity-mars.eps', dpi=300, bbox_inches='tight')

plt.show()

```



4.38 Example - 38 - Curiosity - Mars

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Curiosity aeroshell (MSL) [https://en.wikipedia.org/wiki/Curiosity_\(rover\)](https://en.wikipedia.org/wiki/Curiosity_(rover))

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("MARS")
planet.h_skip = 126.0E3
```

(continues on next page)

(continued from previous page)

```
planet.h_trap = 2.0E3
planet.loadAtmosphereModel('../atmdata/Mars/mars-gram-avg.dat', 0, 1, 2, 3)
```

[4]: # Set up the vehicle
`vehicle=Vehicle('Curiosity', 3257.0, 146.0, 0.0, np.pi*4.5**2.0, 0.0, 1.125, planet)`

[5]: # Set up entry parameters
`vehicle.setInitialState(125.0, 0.0, 0.0, 6.08, 0.0, -15.48, 0.0, 0.0)`

[6]: # Set up solver
`vehicle.setSolverParams(1E-6)`

[7]: # Propogate vehicle entry trajectory
`vehicle.propogateEntry (30*60.0, 0.1, 0.0)`

[8]: # import rcParams to set figure font type
`from matplotlib import rcParams`

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'\$W/cm^2\$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)

(continues on next page)

(continued from previous page)

```

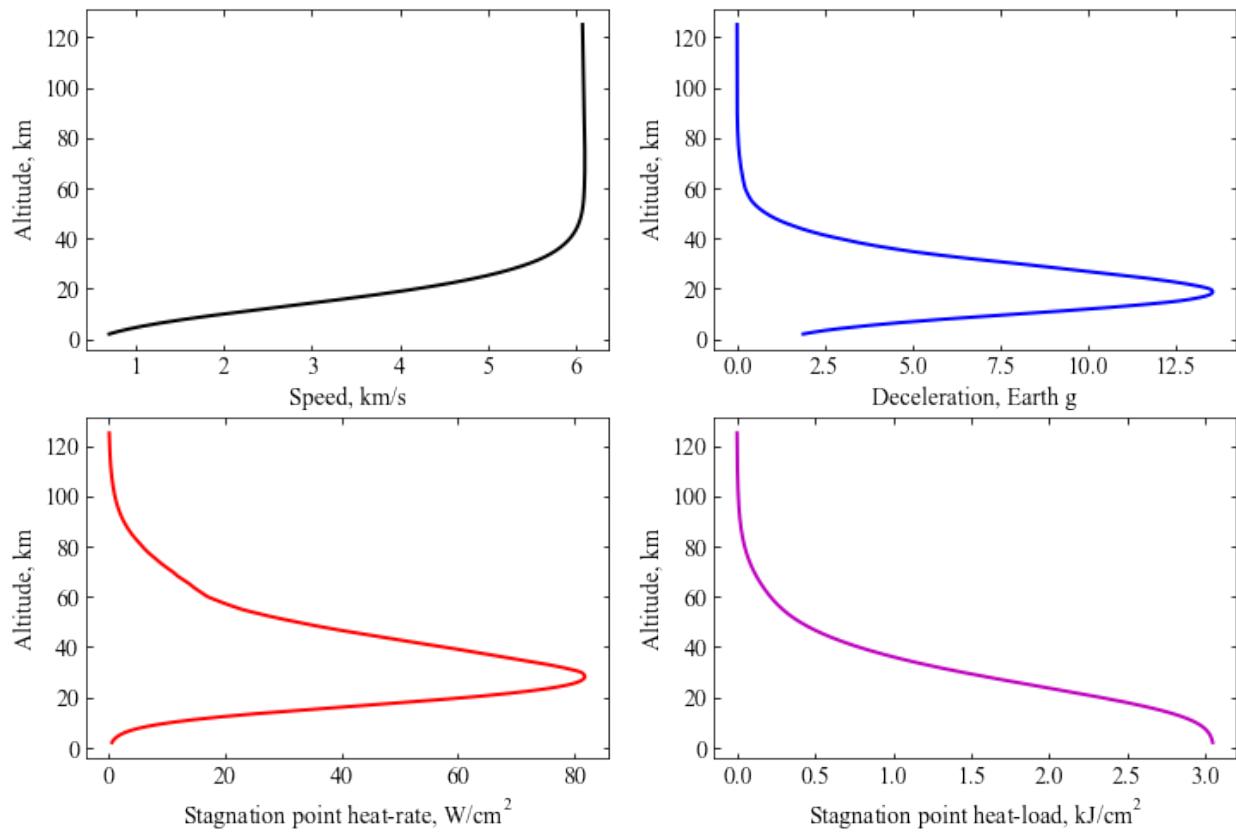
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$\text{kJ/cm}^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/curiosity-mars.png', bbox_inches='tight')
plt.savefig('../plots/curiosity-mars.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/curiosity-mars.eps', dpi=300, bbox_inches='tight')

plt.show()

```



4.39 Example - 39 - Oceanus - Saturn (Concept)

```
[1]: from AMAT.planet import Planet
      from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Saturn entry probe which was part of the Oceanus mission study
<https://doi.org/10.1016/j.asr.2017.02.012>

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("SATURN")
planet.h_skip = 1200.0E3
planet.h_trap = -100.0E3
planet.loadAtmosphereModel('../atmdata/Saturn/saturn-nominal.dat', 0, 1, 2, 3,
                           heightInKmFlag=True)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Oceanus-Saturn-Probe', 140.0, 127.0, 0.0, np.pi*1.0**2.0*0.25, 0.0,
                0.18, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(1200.0, 0.0, 0.0, 28.46, 0.0, -11.40, 0.0, 0.0)
```

```
[6]: # Set up solver
vehicle.setSolverParams(1E-6)
```

```
[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (100*60.0, 0.1, 0.0)
```

```
[8]: # import rcParams to set figure font type
from matplotlib import rcParams
```

```
[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)
```

(continues on next page)

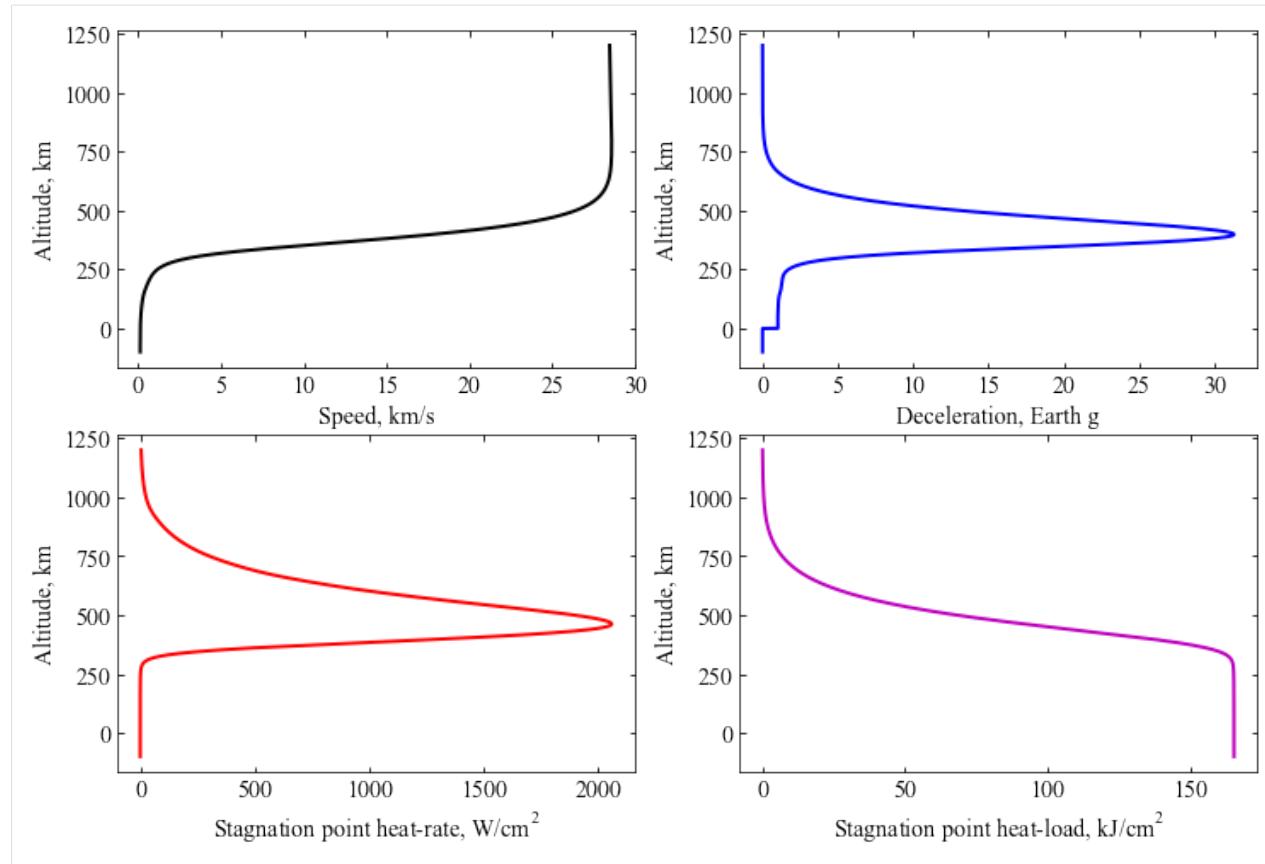
(continued from previous page)

```
plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc, 'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/oceanus-saturn-probe.png', bbox_inches='tight')
plt.savefig('../plots/oceanus-saturn-probe.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/oceanus-saturn-probe.eps', dpi=300, bbox_inches='tight')

plt.show()
```



The max. heat rate is smaller than reported in the paper by about a factor of 3. The difference is attributed to different heating correlations used.

4.40 Example - 40 - Oceanus - Uranus (Concept)

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Uranus entry probe which was part of the Oceanus mission study
<https://doi.org/10.1016/j.asr.2017.02.012>

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("URANUS")
planet.h_skip = 1000.0E3
planet.h_trap = -100.0E3
planet.loadAtmosphereModel('../atmdata/Uranus/uranus-ames.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Oceanus-Uranus-Probe', 120.0, 170.0, 0.0, np.pi*0.8**2.0*0.25, 0.0, 0.28, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(1000.0, 0.0, 0.0, 21.47, 0.0, -31.5, 0.0, 0.0)

[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (100*60.0, 0.1, 0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
```

(continues on next page)

(continued from previous page)

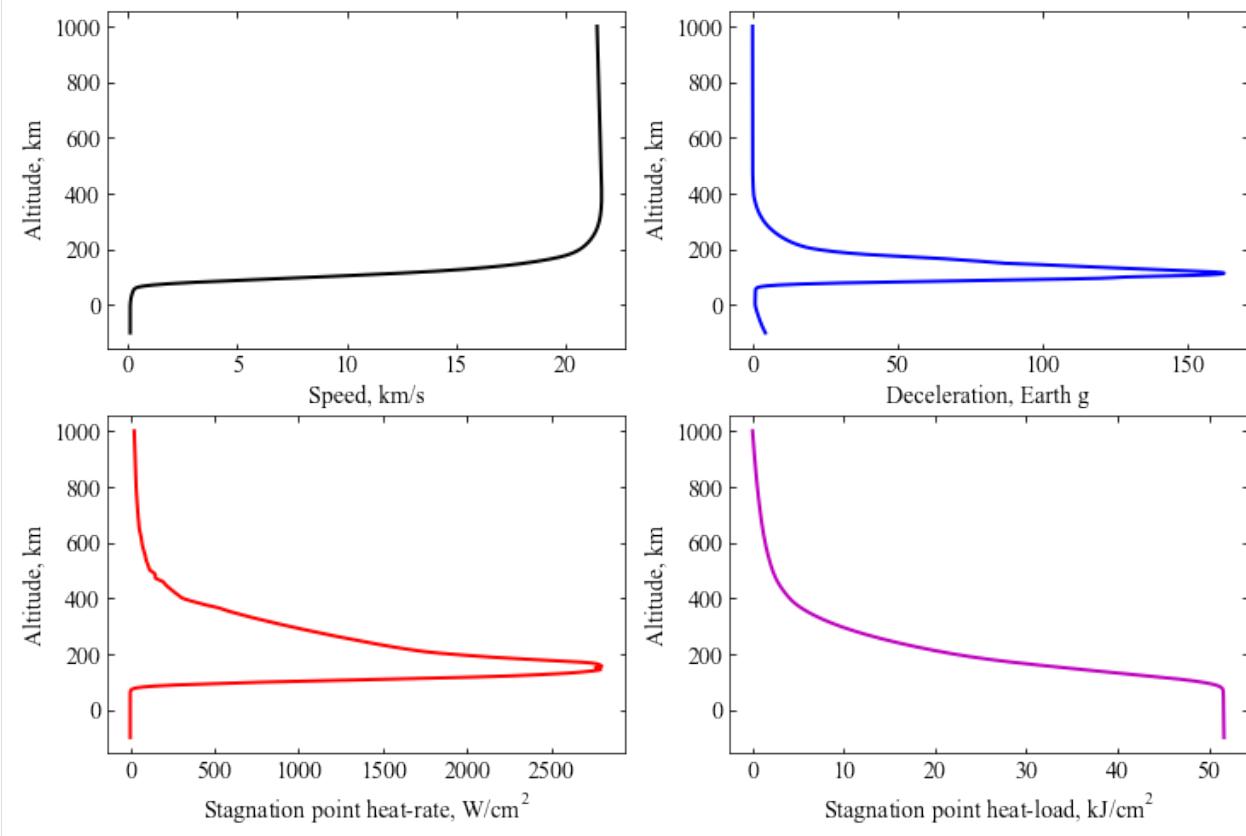
```

ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/oceanus-uranus-probe.png', bbox_inches='tight')
plt.savefig('../plots/oceanus-uranus-probe.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/oceanus-uranus-probe.eps', dpi=300, bbox_inches='tight')

plt.show()

```



4.41 Example - 41 - Hera - Saturn (Concept)

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Saturn entry probe which was part of the Hera mission concept study. <http://dx.doi.org/10.1016/j.pss.2015.06.020>

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("SATURN")
planet.h_skip = 1000.0E3
planet.h_trap = -100.0E3
planet.loadAtmosphereModel('../atmdata/Saturn/saturn-nominal.dat', 0 , 1 , 2, 3,
                           ↪heightInKmFlag=True)
```



```
[4]: # Set up the vehicle
vehicle1=Vehicle('Hera-Saturn-Probe-steep', 220, 269, 0.0, np.pi*1.0**2.0*0.25, 0.0,
                  ↪0.18, planet)
vehicle2=Vehicle('Hera-Saturn-Probe-shallow', 220, 269, 0.0, np.pi*1.0**2.0*0.25, 0.0,
                  ↪ 0.18, planet)
```



```
[5]: # Set up entry parameters
vehicle1.setInitialState(1000.0,0.0,0.0,26.3,0.0,-22.0,0.0,0.0)
vehicle2.setInitialState(1000.0,0.0,0.0,26.3,0.0,-9.0,0.0,0.0)
```



```
[6]: # Set up solver
vehicle1.setSolverParams(1E-6)
vehicle2.setSolverParams(1E-6)
```



```
[7]: # Propogate vehicle entry trajectory
vehicle1.propogateEntry (100*60.0,0.1,0.0)
vehicle2.propogateEntry (100*60.0,0.1,0.0)
```



```
[8]: # import rcParams to set figure font type
from matplotlib import rcParams
```



```
[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle1.v_kmsc, vehicle1.h_kmc, 'k-', linewidth=2.0, label='Steep')
plt.plot(vehicle2.v_kmsc, vehicle2.h_kmc, 'k--', linewidth=2.0, label='Shallow')
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
plt.legend(loc='upper left', fontsize=12, frameon=False)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle1.acc_net_g, vehicle1.h_kmc, 'b-', linewidth=2.0, label='Steep')
plt.plot(vehicle2.acc_net_g, vehicle2.h_kmc, 'b--', linewidth=2.0, label='Shallow')
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
plt.legend(loc='upper right', fontsize=12, frameon=False)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

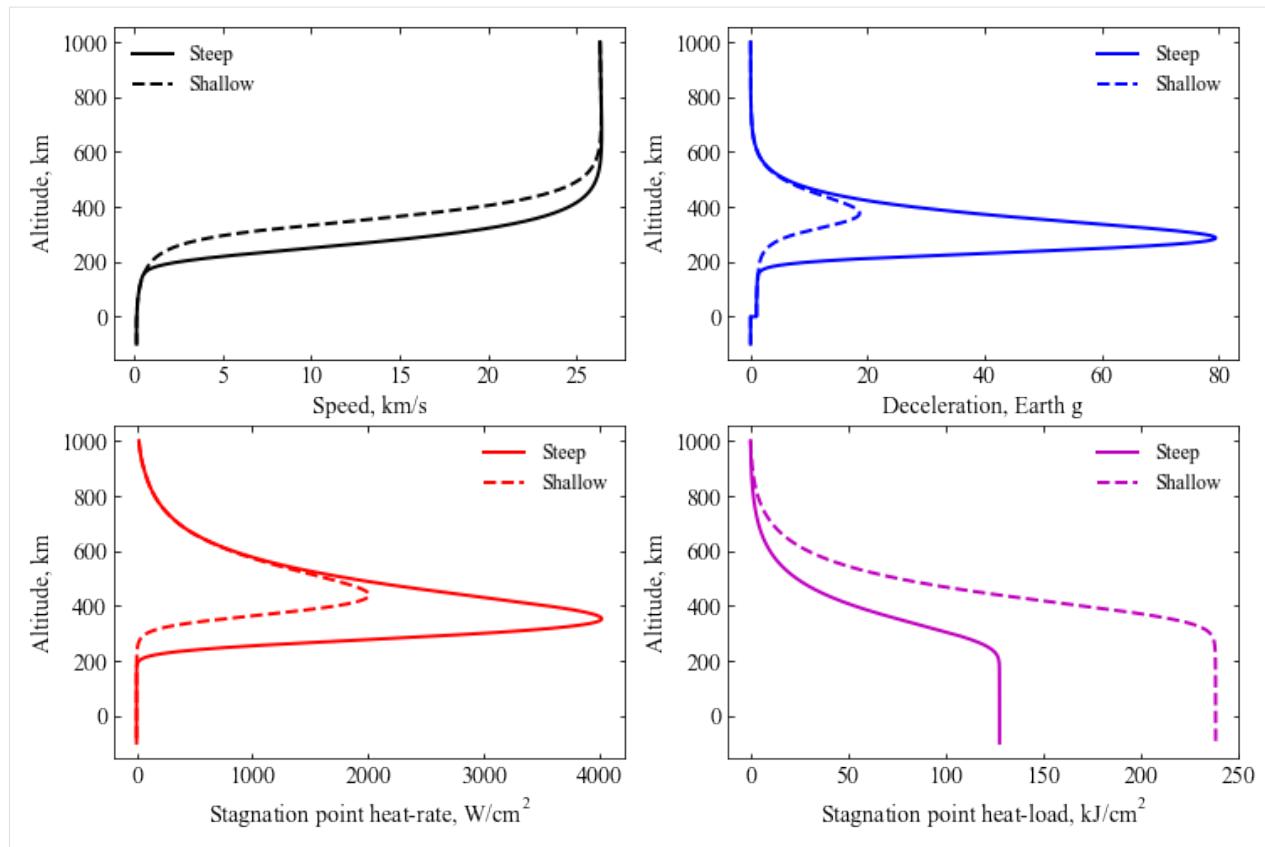
```
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle1.q_stag_total, vehicle1.h_kmc, 'r-', linewidth=2.0, label='Steep')
plt.plot(vehicle2.q_stag_total, vehicle2.h_kmc, 'r--', linewidth=2.0, label='Shallow')
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
plt.legend(loc='upper right', fontsize=12, frameon=False)
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle1.heatload/1.0E3, vehicle1.h_kmc, 'm-', linewidth=2.0, label='Steep')
plt.plot(vehicle2.heatload/1.0E3, vehicle2.h_kmc, 'm--', linewidth=2.0, label='Shallow
    -->
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
plt.legend(loc='upper right', fontsize=12, frameon=False)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.savefig('../plots/hera-saturn-probe.png',bbox_inches='tight')
plt.savefig('../plots/hera-saturn-probe.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/hera-saturn-probe.eps', dpi=300,bbox_inches='tight')

plt.show()
```



4.42 Example - 42 - Dragonfly - Titan (Planned)

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Dragonfly probe at Titan. [https://en.wikipedia.org/wiki/Dragonfly_\(spacecraft\)](https://en.wikipedia.org/wiki/Dragonfly_(spacecraft))

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("TITAN")
planet.h_skip = 1270.0E3
planet.h_trap = 0.0E3
planet.loadAtmosphereModel('../atmdata/Titan/titan-gram-avg.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle1=Vehicle('Dragonfly', 1000.0, 140.0, 0.0, np.pi*3.7**2.0*0.25, 0.0, 0.43,_
planet)
```

```
[5]: # Set up entry parameters
vehicle1.setInitialState(1270.0,0.0,0.0,7.3,0.0,-49.7,0.0,0.0)
```

```
[6]: # Set up solver
vehicle1.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle1.propogateEntry (120*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle1.v_kmsc, vehicle1.h_kmc, 'k-', linewidth=2.0)

plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle1.acc_net_g, vehicle1.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle1.q_stag_total, vehicle1.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle1.heatload/1.0E3, vehicle1.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

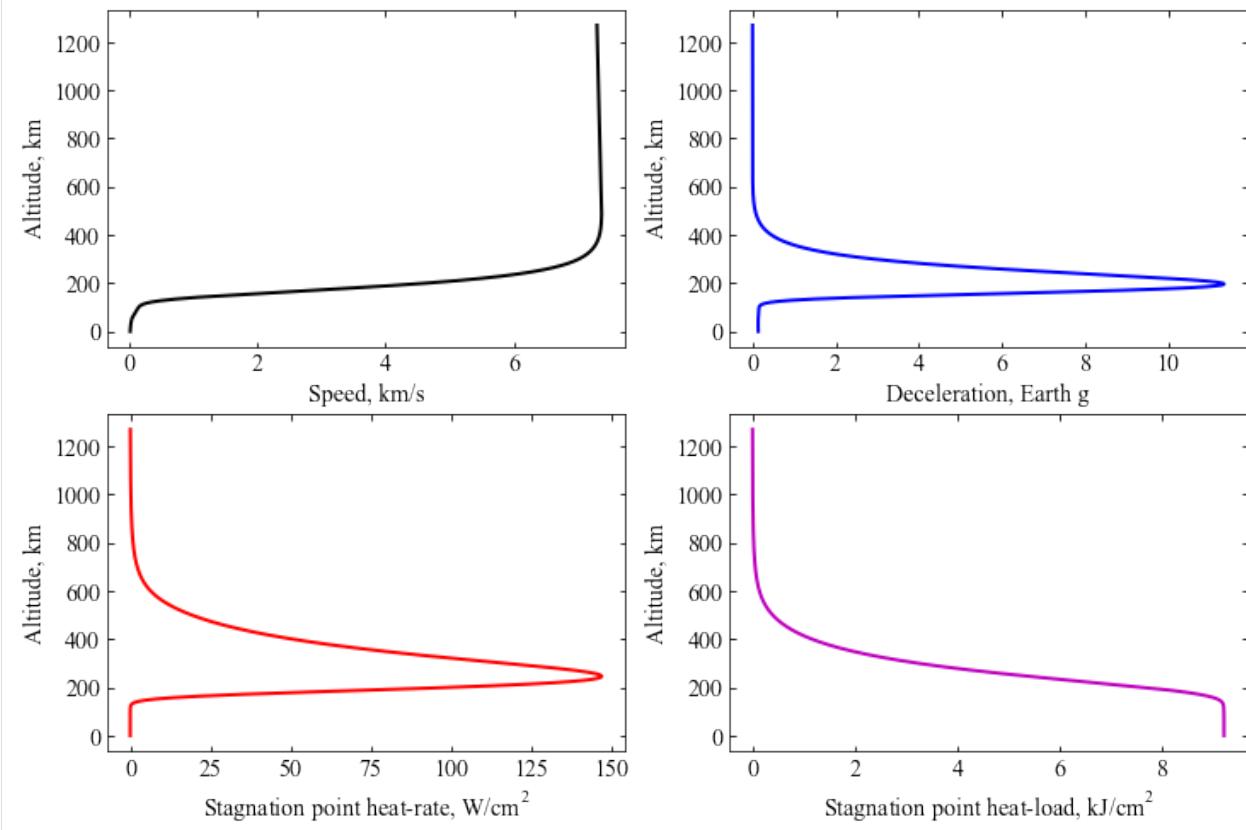
```

ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/dragnofly-titan.png', bbox_inches='tight')
plt.savefig('../plots/dragnofly-titan.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/dragnofly-titan.eps', dpi=300, bbox_inches='tight')

plt.show()

```



Vehicle geometry and ballistic coefficient data is not publicly available in the literature. Hence, results are only approximate.

4.43 Example - 43 - Ice Giant Pre-Decadal - Uranus Probe (Concept)

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Uranus entry probe described in Sec. A.5.3 of the Ice Giants Pre-Decadal Mission Study. https://www.lpi.usra.edu/icegiants/mission_study/Full-Report.pdf

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("URANUS")
planet.h_skip = 1015.0E3
planet.h_trap = -100.0E3
planet.loadAtmosphereModel('../atmdata/Uranus/uranus-ames.dat', 0 , 1 , 2, 3)

[4]: # Set up the vehicle
vehicle1=Vehicle('igpd-uranus', 325.0, 205.0, 0.0, np.pi*1.2**2.0*0.25, 0.0, 0.20, ↵
                  planet)

[5]: # Set up entry parameters
vehicle1.setInitialState(1015.0,0.0,0.0,23.10,0.0,-30.0,0.0,0.0)

[6]: # Set up solver
vehicle1.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle1.propogateEntry (120*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle1.v_kmsc, vehicle1.h_kmc, 'k-', linewidth=2.0)

plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle1.acc_net_g, vehicle1.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle1.q_stag_total, vehicle1.h_kmc, 'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
```

(continues on next page)

(continued from previous page)

```

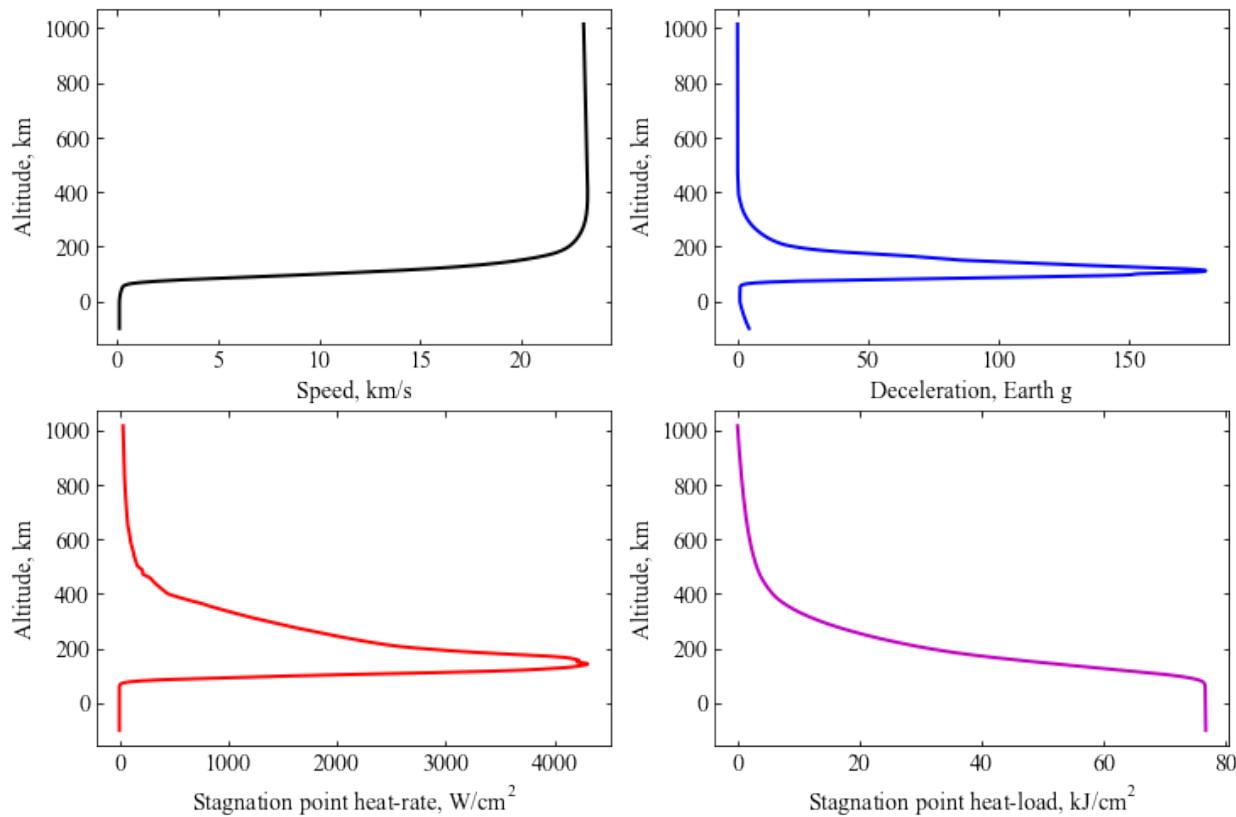
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle1.heatload/1.0E3, vehicle1.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$\text{kJ/cm}^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/igpd-uranus.png', bbox_inches='tight')
plt.savefig('../plots/igpd-uranus.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/igpd-uranus.eps', dpi=300, bbox_inches='tight')

plt.show()

```



Discrepancies are attributed to initial state being not properly converted to planet-relative frame, and different heating relations used in the studies.

4.44 Example - 44 - Ice Giant Pre-Decadal - Neptune Probe (Concept)

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle

[2]: import numpy as np
import matplotlib.pyplot as plt

This notebook simulates the atmospheric entry of the Neptune entry probe described in Sec. A.5.3 of the Ice Giants Pre-Decadal Mission Study. https://www.lpi.usra.edu/icegiants/mission\_study/Full-Report.pdf

[3]: # Set up the planet and atmosphere model.
planet=Planet("NEPTUNE")
planet.h_skip = 1000.0E3
planet.h_trap = -100.0E3
planet.loadAtmosphereModel('..../atmdata/Neptune/neptune-gram-avg.dat', 0 , 7 , 6, 5 ,
                           ↴heightInKmFlag=True)

[4]: # Set up the vehicle
vehicle1=Vehicle('igpd-neptune', 325.0, 205.0, 0.0, np.pi*1.2**2.0*0.25, 0.0, 0.20,
                  ↴planet)

[5]: # Set up entry parameters
vehicle1.setInitialState(1000.0,0.0,0.0,25.73,0.0,-20.0,0.0,0.0)

[6]: # Set up solver
vehicle1.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle1.propogateEntry (120.0*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle1.v_kmsc, vehicle1.h_kmc, 'k-', linewidth=2.0)

plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle1.acc_net_g, vehicle1.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
```

(continues on next page)

(continued from previous page)

```

plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

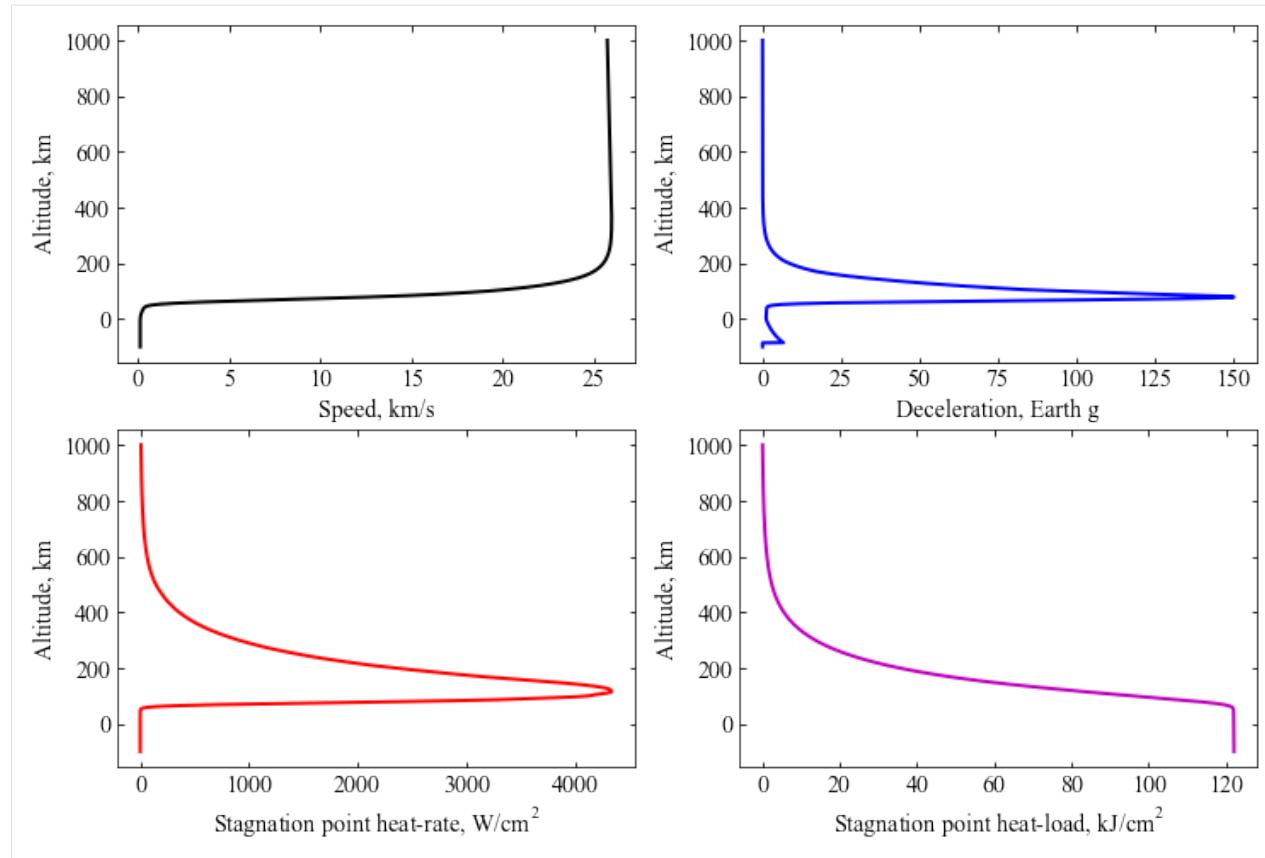
plt.subplot(2, 2, 3)
plt.plot(vehicle1.q_stag_total, vehicle1.h_kmc, 'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle1.heatload/1.0E3, vehicle1.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/igpd-neptune.png', bbox_inches='tight')
plt.savefig('../plots/igpd-neptune.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/igpd-neptune.eps', dpi=300, bbox_inches='tight')

plt.show()

```



Discrepancies are attributed to initial state being not properly converted to planet-relative frame, and different heating relations used in the studies.

4.45 Example - 45 - ADEPT ViTaL - Venus (Concept)

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the ADEPT ViTaL entry system concept. <https://ieeexplore.ieee.org/document/6497176>

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("VENUS")
planet.h_skip = 200.0E3
planet.h_trap = 20.0E3
planet.loadAtmosphereModel('../atmdata/Venus/venus-gram-avg.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle1=Vehicle('adept-vital', 1620.0, 38.0, 0.0, np.pi*6.0**2.0*0.25, 0.0, 3.0, ↴planet)
```

```
[5]: # Set up entry parameters
vehicle1.setInitialState(200.0,0.0,0.0,10.8,0.0,-8.25,0.0,0.0)

[6]: # Set up solver
vehicle1.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle1.propogateEntry (120.0*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle1.v_kmsc, vehicle1.h_kmc, 'k-', linewidth=2.0)

plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle1.acc_net_g, vehicle1.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle1.q_stag_total, vehicle1.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle1.heatload/1.0E3, vehicle1.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
```

(continues on next page)

(continued from previous page)

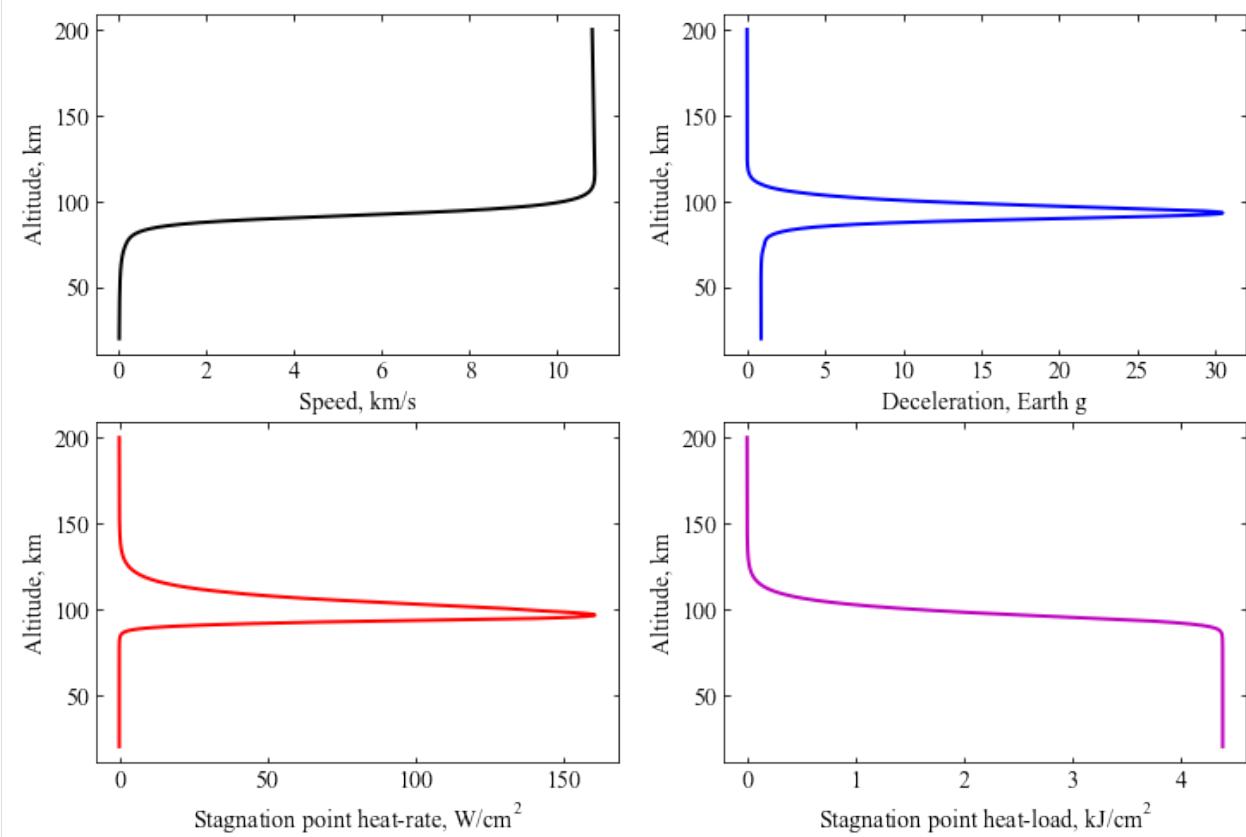
```

ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.savefig('../plots/adept-vital-venus.png',bbox_inches='tight')
plt.savefig('../plots/adept-vital-venus.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/adept-vital-venus.eps', dpi=300,bbox_inches='tight')

plt.show()

```



```

[20]: plt.figure(figsize=(10,6))
plt.plot(vehicle1.stag_pres_atm[0:1700], vehicle1.q_stag_total[0:1700], 'r-', linewidth=2.0)
plt.xlabel('Stagnation pressure, atm', fontsize=14)
plt.ylabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.savefig('../plots/adept-vital-venus-thermal.png',bbox_inches='tight')
plt.savefig('../plots/adept-vital-venus.thermal.pdf', dpi=300,bbox_inches='tight')

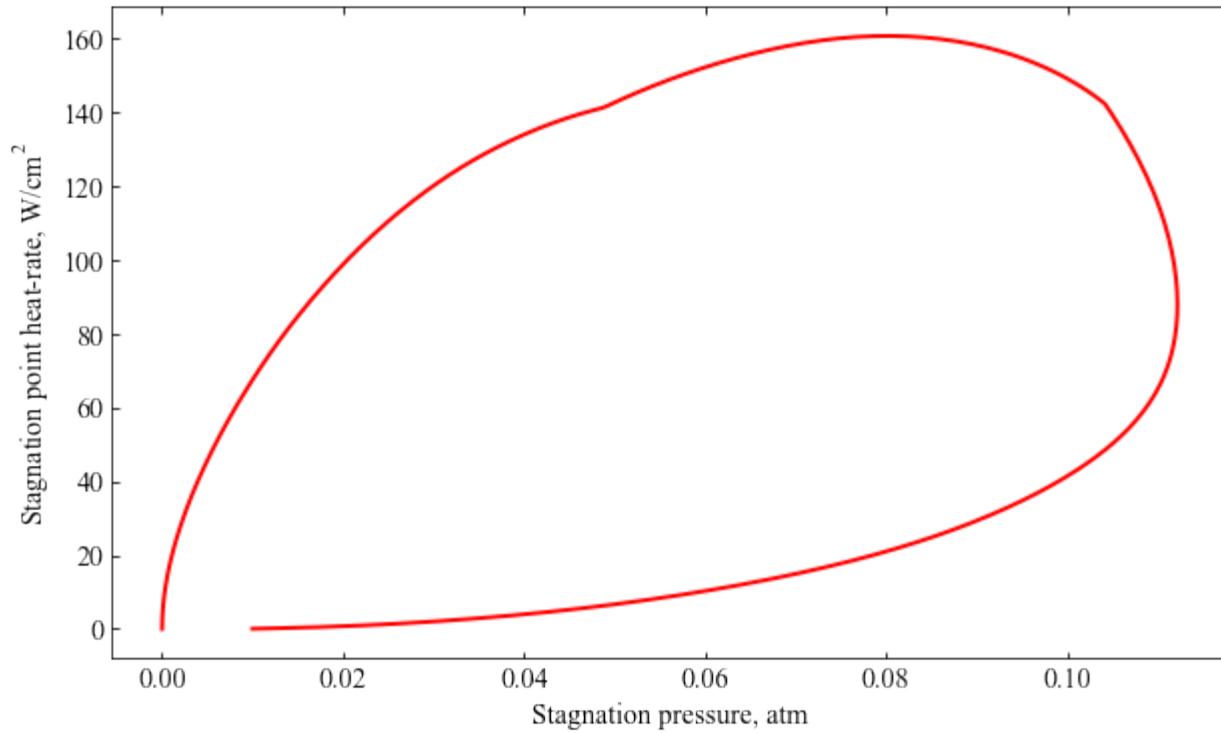
```

(continues on next page)

(continued from previous page)

```
plt.savefig('../plots/adept-vital-venus.thermal.eps', dpi=300, bbox_inches='tight')

plt.show()
```



4.46 Example - 46 - Uranus Probe - Decadal Study - 2010 (Concept)

```
[1]: from AMAT.planet import Planet
      from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
      import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Uranus entry probe studied during the 2010 Decadal Mission Concept Study (page 42). https://sites.nationalacademies.org/cs/groups/ssbsite/documents/webpage/ssb_059323.pdf

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("URANUS")
planet.h_skip = 500.0E3
planet.h_trap = -100.0E3
planet.loadAtmosphereModel('../atmdata/Uranus/uranus-ames.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle1=Vehicle('uranus-probe', 88.0, 190.0, 0.0, 0.46, 0.0, 0.19, planet)
```

```
[5]: # Set up entry parameters
vehicle1.setInitialState(500.0, 0.0, 0.0, 22.3, 0.0, -68, 0.0, 0.0)
```

```
[6]: # Set up solver
vehicle1.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle1.propogateEntry (120.0*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle1.v_kmsc, vehicle1.h_kmc, 'k-', linewidth=2.0)

plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle1.acc_net_g, vehicle1.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle1.q_stag_total, vehicle1.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle1.heatload/1.0E3, vehicle1.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

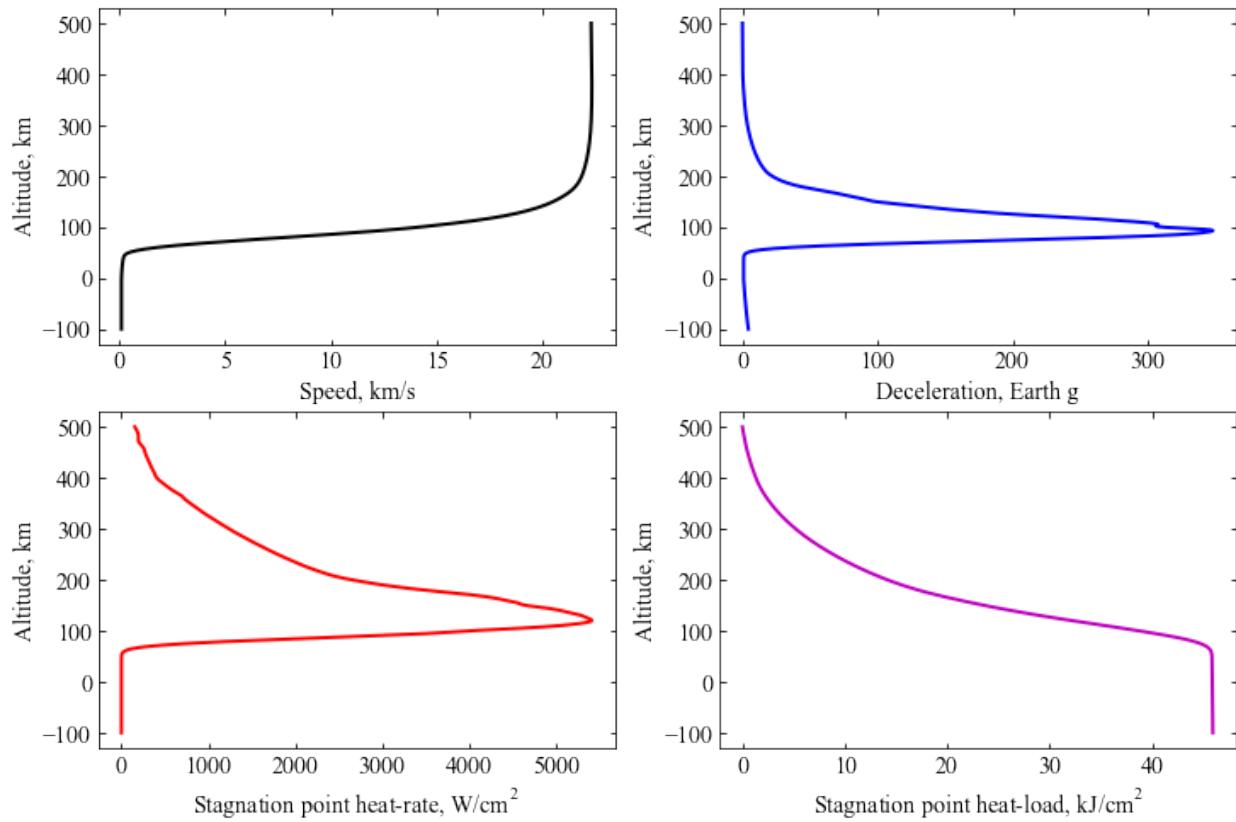
```

ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/decadal-2010-uranus-probe.png', bbox_inches='tight')
plt.savefig('../plots/decadal-2010-uranus-probe.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/decadal-2010-uranus-probe.eps', dpi=300, bbox_inches='tight')

plt.show()

```



```

[10]: plt.figure(figsize=(6, 6))
plt.plot(vehicle1.stag_pres_atm[0:1700], vehicle1.q_stag_total[0:1700], 'r-', linewidth=2.0)
plt.xlabel('Stagnation pressure, atm', fontsize=14)
plt.ylabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

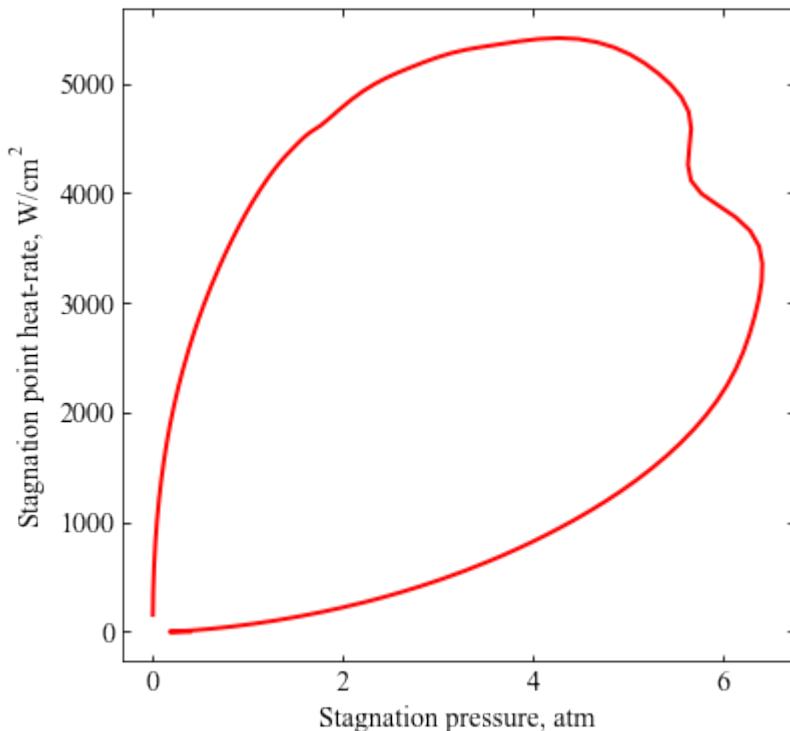
plt.savefig('../plots/decadal-2010-uranus-probe-thermal.png', bbox_inches='tight')
plt.savefig('../plots/decadal-2010-uranus-probe-thermal.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/decadal-2010-uranus-probe-thermal.eps', dpi=300, bbox_inches='tight')

```

(continues on next page)

(continued from previous page)

```
plt.show()
```



4.47 Example - 47 - Venus Flagship - Decadal Study - 2010 (Concept)

```
[1]: from AMAT.planet import Planet
      from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
      import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Venus Flagship Mission, 2010 Decadal Mission Concept Study (page 159). <https://vfm.jpl.nasa.gov/files/Venus+Flagship+Mission+Study+090501-compressed.pdf>

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("VENUS")
planet.h_skip = 220.0E3
planet.h_trap = 10.0E3
planet.loadAtmosphereModel('../atmdata/Venus/venus-gram-avg.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle1=Vehicle('venus-flagship', 2020.0, 349.0, 0.0, np.pi*2.65**2.0*0.25, 0.0, 0.
                  ↴30, planet)
```

```
[5]: # Set up entry parameters
vehicle1.setInitialState(220.0, 0.0, 0.0, 11.1, 0.0, -13.8, 0.0, 0.0)
```

```
[6]: # Set up solver
vehicle1.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle1.propogateEntry (120.0*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle1.v_kmsc, vehicle1.h_kmc, 'k-', linewidth=2.0)

plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle1.acc_net_g, vehicle1.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle1.q_stag_total, vehicle1.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle1.heatload/1.0E3, vehicle1.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

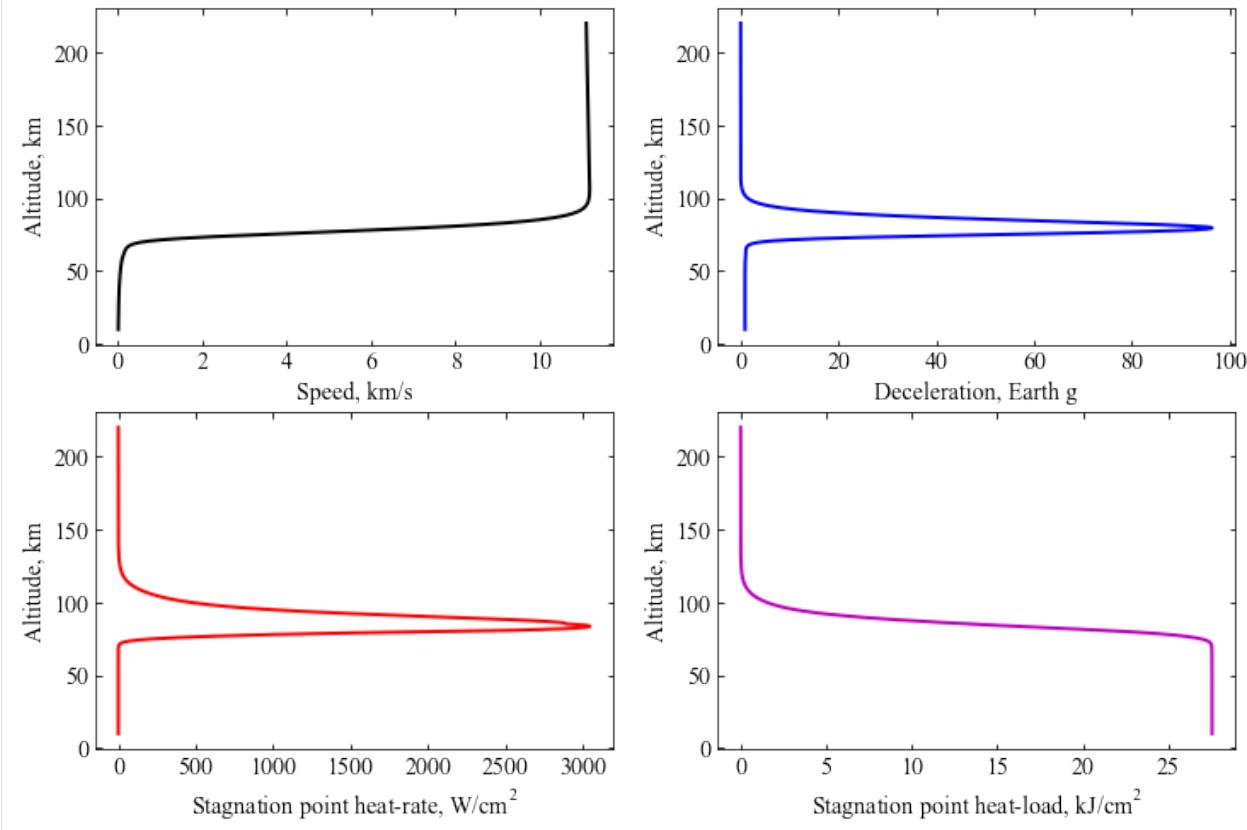
```

ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.savefig('../plots/decadal-2010-venus-flagship.png',bbox_inches='tight')
plt.savefig('../plots/decadal-2010-venus-flagship.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/decadal-2010-venus-flagship.eps', dpi=300,bbox_inches='tight')

plt.show()

```



```

[10]: plt.figure(figsize=(6, 6))
plt.plot(vehicle1.stag_pres_atm[0:1700], vehicle1.q_stag_total[0:1700], 'r-', linewidth=2.0)
plt.xlabel('Stagnation pressure, atm', fontsize=14)
plt.ylabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

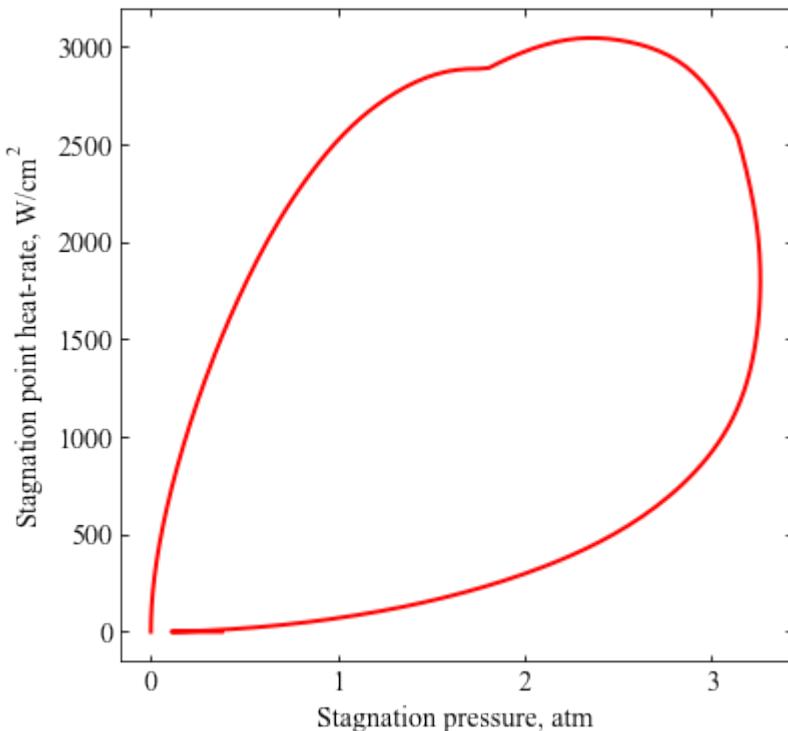
plt.savefig('../plots/decadal-2010-venus-flagship-thermal.png',bbox_inches='tight')
plt.savefig('../plots/decadal-2010-venus-flagship-thermal.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/decadal-2010-venus-flagship-thermal.eps', dpi=300,bbox_inches='tight')

```

(continues on next page)

(continued from previous page)

```
plt.show()
```



4.48 Example - 48 - Crew Module Atmospheric Re-entry Experiment

```
[1]: from AMAT.planet import Planet
      from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
      import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of Crew Module Atmospheric Re-entry Experiment. https://www.isro.gov.in/sites/default/files/flipping_book/LVM-3/CARE/files/assets/common/downloads/LVM3-brochure.pdf

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("EARTH")
planet.h_skip = 126.0E3
planet.h_trap = 0.0E3
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle1=Vehicle('care', 3735.0, 350.0, 0.0, np.pi*3.1**2.0*0.25, 0.0, 3.0, planet)
```

```
[5]: # Set up entry parameters
vehicle1.setInitialState(126.0,0.0,0.0,5.326,0.0,0.0,0.0,0.0)
```

```
[6]: # Set up solver
vehicle1.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle1.propogateEntry (120.0*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle1.v_kmsc, vehicle1.h_kmc, 'k-', linewidth=2.0)

plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle1.acc_net_g, vehicle1.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle1.q_stag_total, vehicle1.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle1.heatload/1.0E3, vehicle1.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
```

(continues on next page)

(continued from previous page)

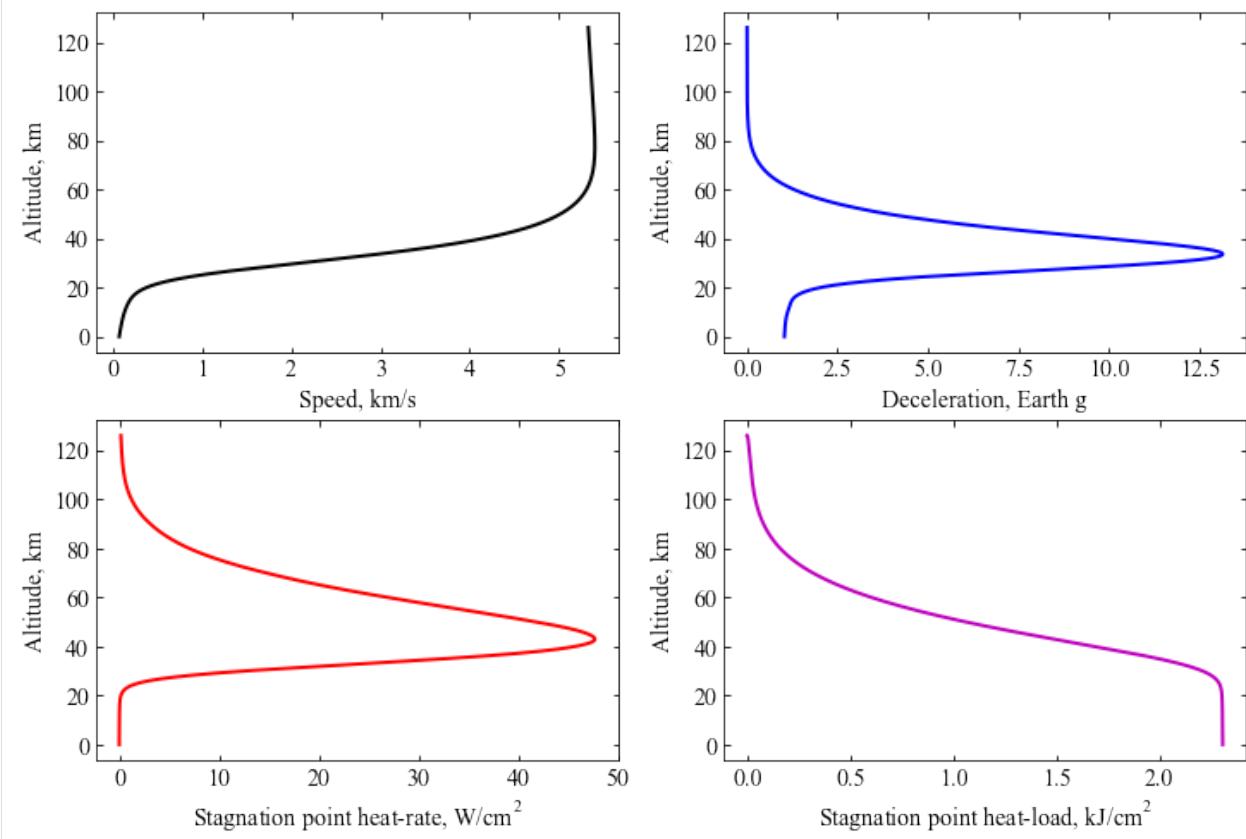
```

ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/crew-module-atmospheric-re-entry-experiment-earth.png', bbox_inches='tight')
plt.savefig('../plots/crew-module-atmospheric-re-entry-experiment-earth.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/crew-module-atmospheric-re-entry-experiment-earth.eps', dpi=300, bbox_inches='tight')

plt.show()

```



```

[10]: plt.figure(figsize=(6, 6))
plt.plot(vehicle1.stag_pres_atm[0:2700], vehicle1.q_stag_total[0:2700], 'r-', linewidth=2.0)
plt.xlabel('Stagnation pressure, atm', fontsize=14)
plt.ylabel('Stagnation point heat-rate, '+r'$W/cm^2$', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

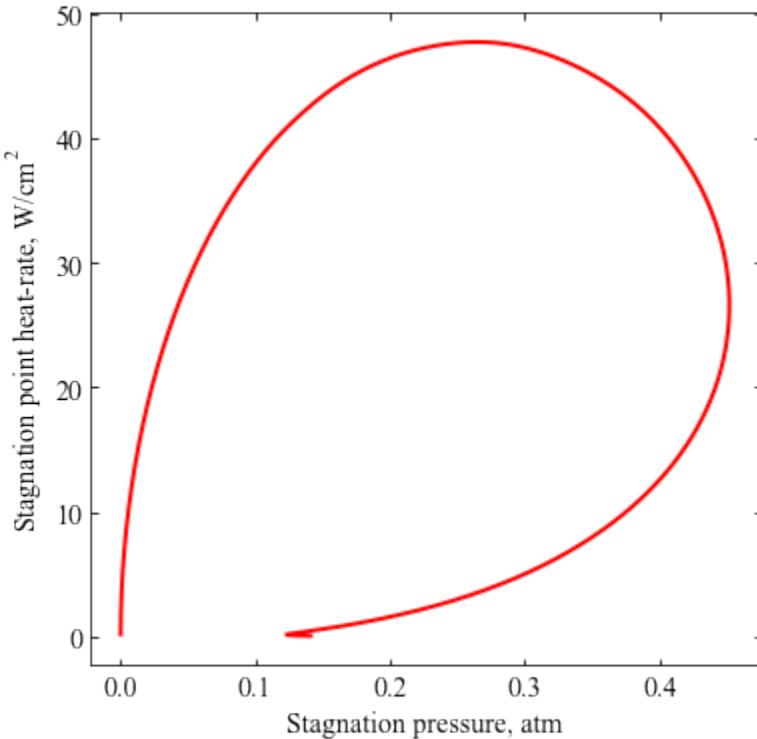
plt.savefig('../plots/crew-module-atmospheric-re-entry-experiment-earth-thermal.png',
bbox_inches='tight')

```

(continues on next page)

(continued from previous page)

```
plt.savefig('../plots/crew-module-atmospheric-re-entry-experiment-earth-thermal.pdf',  
          dpi=300, bbox_inches='tight')  
plt.savefig('../plots/crew-module-atmospheric-re-entry-experiment-earth-thermal.eps',  
          dpi=300, bbox_inches='tight')  
  
plt.show()
```



4.49 Example - 49 - Earth SmallSat Aerocapture Demonstration - Part 1

This examples reproduces results from M.S.Werner and R.D.Braun, Mission Design and Performance Analysis of a Smallsat Aerocapture Flight Test, Journal of Spacecraft and Rockets, DOI: 10.2514/1.A33997.

```
[1]: from IPython.display import Image  
Image(filename='../plots/werner-smallsat.png', width=500)
```

[1]:

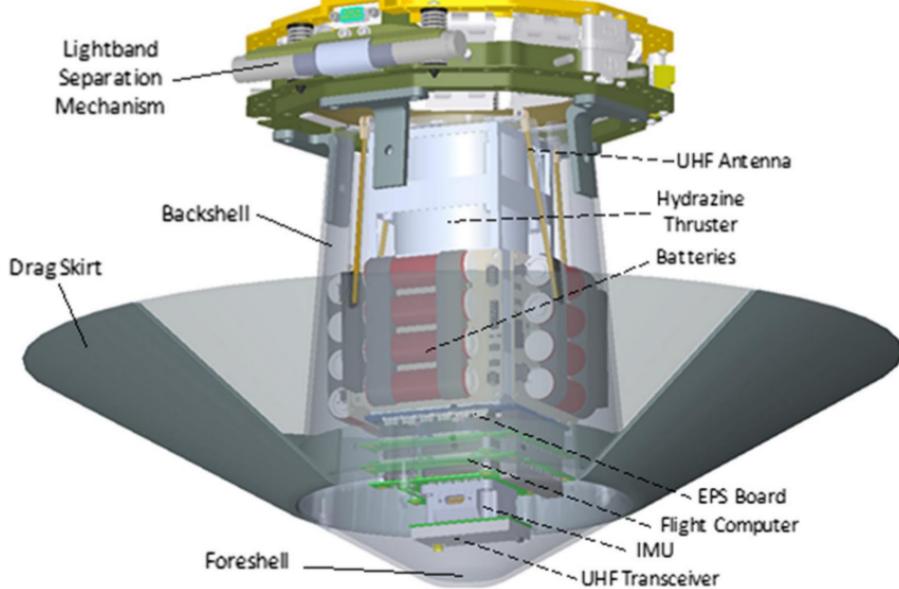


Image credit: M.S.Werner and R.D.Braun

```
[2]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[3]: import numpy as np
from scipy import interpolate
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib import rcParams
```

```
[4]: planet = Planet('EARTH')
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0, 1, 2, 3)
planet.h_skip = 125.0E3
```

```
[5]: vehicle=Vehicle('EarthSmallSat', 25.97, 66.4, 0.0, np.pi*0.25**2, 0.0, 0.0563, planet)
vehicle.setInitialState(125.0,0.0,0.0,9.8,0.0,-5.00,0.0,0.0)
vehicle.setSolverParams(1E-6)
vehicle.setDragModulationVehicleParams(66.4,4.72)
```

First, find the corridor bounds to select a target nominal EFPA for a nominal Earth atmosphere.

```
[6]: underShootLimit, exitflag_us = vehicle.findUnderShootLimitD(2400.0, 0.1, -30.0,-2.0, -1E-10, 1760.0)
overShootLimit , exitflag_os = vehicle.findOverShootLimitD(2400.0, 0.1, -30.0,-2.0, -1E-10, 1760.0)

print(underShootLimit, exitflag_us)
print(overShootLimit, exitflag_os)

-5.141069082768809 1.0
-4.649618244053272 1.0
```

It is good practice to see how these corridor bounds vary with minimum, average, and maximum density atmospheres.

We will check how much these corridor bounds change using +/- 3-sigma bounds from EARTH GRAM output files.

Load three density profiles for minimum, average, and maximum scenarios from a GRAM output file.

```
[7]: ATM_height, ATM_density_low, ATM_density_avg, ATM_density_high, ATM_density_pert =  
    ↪planet.loadMonteCarloDensityFile3('../atmdata/Earth/LAT00N.txt', 1, 4, 14, 9,  
    ↪heightInKmFlag=True)  
density_int_low = planet.loadAtmosphereModel5(ATM_height, ATM_density_low, ATM_  
    ↪density_avg, ATM_density_high, ATM_density_pert, -3.0, 141, 200)  
density_int_avg = planet.loadAtmosphereModel5(ATM_height, ATM_density_low, ATM_  
    ↪density_avg, ATM_density_high, ATM_density_pert, 0.0, 141, 200)  
density_int_hig = planet.loadAtmosphereModel5(ATM_height, ATM_density_low, ATM_  
    ↪density_avg, ATM_density_high, ATM_density_pert, +3.0, 141, 200)
```

Set the planet density_int attribute to density_int_low

```
[8]: planet.density_int = density_int_low
```

Compute the corridor bounds for low density atmosphere.

```
[9]: underShootLimit, exitflag_us = vehicle.findUnderShootLimitD(2400.0, 0.1, -30.0, -2.0,  
    ↪1E-10, 1760.0)  
overShootLimit, exitflag_os = vehicle.findOverShootLimitD(2400.0, 0.1, -30.0, -2.0,  
    ↪1E-10, 1760.0)  
  
print(underShootLimit, exitflag_us)  
print(overShootLimit, exitflag_os)  
  
-5.2014596410954255 1.0  
-4.72837330459879 1.0
```

Repeat for average and maximum density atmospheres.

```
[10]: planet.density_int = density_int_avg
```

```
[11]: underShootLimit, exitflag_us = vehicle.findUnderShootLimitD(2400.0, 0.1, -30.0, -2.0,  
    ↪1E-10, 1760.0)  
overShootLimit, exitflag_os = vehicle.findOverShootLimitD(2400.0, 0.1, -30.0, -2.0,  
    ↪1E-10, 1760.0)  
  
print(underShootLimit, exitflag_us)  
print(overShootLimit, exitflag_os)  
  
-5.14605522868078 1.0  
-4.6526982840077835 1.0
```

```
[12]: planet.density_int = density_int_hig
```

```
[13]: underShootLimit, exitflag_us = vehicle.findUnderShootLimitD(2400.0, 0.1, -30.0, -2.0,  
    ↪1E-10, 1760.0)  
overShootLimit, exitflag_os = vehicle.findOverShootLimitD(2400.0, 0.1, -30.0, -2.0,  
    ↪1E-10, 1760.0)  
  
print(underShootLimit, exitflag_us)  
print(overShootLimit, exitflag_os)  
  
-5.096641375908803 1.0  
-4.5848947728809435 1.0
```

The above numbers indicate that corridor bounds do not vary that much with atmospheric variations. The corridor is approximately 0.50 deg wide.

```
[14]: 5.09-4.58
[14]: 0.5099999999999998
```

The mid-corridor is typically a good place to start. We will use the mean of the corridor bounds for the average atmosphere.

```
[15]: 0.5*(-4.65-5.15)
[15]: -4.9
```

We will propagate a nominal guided trajectory to reproduce Fig. 5 from the paper.

```
[16]: # Set planet.h_low to 10 km, if vehicle dips below this level
# trajectory is terminated.
planet.h_low=10.0E3

# Set target orbit = 180 km x 1760, tolerance = 50 km
vehicle.setTargetOrbitParams(180.0, 1760.0, 50.0)

# Set entry phase parameters
# v_switch_kms = 5.0, lowAlt_km = 50.0,
# numPoints_lowAlt = 101, hdot_threshold = -200.0 m/s.
# These are somewhat arbitrary based on experience.
vehicle.setDragEntryPhaseParams(5.0, 50.0, 101, -200.0)

# Set beta_1 and beta_ratio
vehicle.setDragModulationVehicleParams(66.4, 4.72)

# Set vehicle initial state
vehicle.setInitialState(125.0, 0.0, 0.0, 9.8, 0.0, -4.90, 0.0, 0.0)
```

```
[17]: # Propagate a single vehicle trajectory
vehicle.propogateGuidedEntryD(1.0, 1.0, 0.1, 2400.0)
```

```
[18]: plt.figure(figsize=(6,4))
plt.rc('font', family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.plot(vehicle.v_kms_full, vehicle.h_km_full, 'r-', linewidth=2.0)

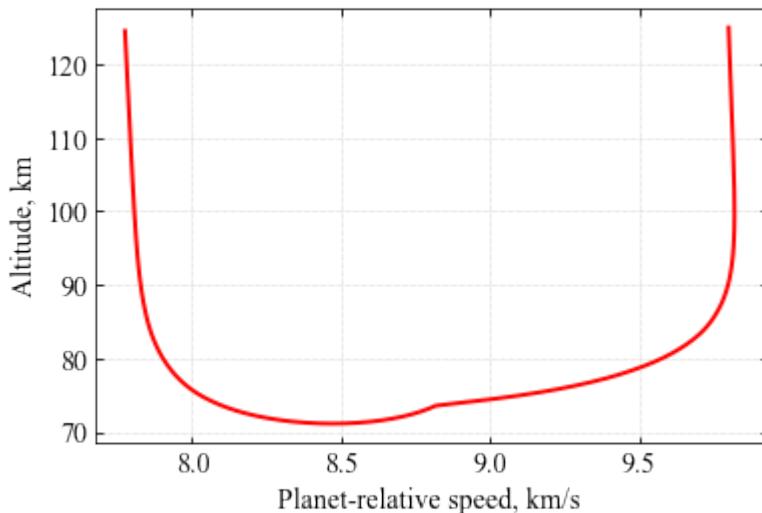
plt.xlabel('Planet-relative speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)
plt.grid(linestyle='dotted', linewidth=0.5)

plt.savefig('../plots/werner-smallsat-nominal-altitude-speed.png', bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-nominal-altitude-speed.pdf', dpi=300, bbox_
    ↪inches='tight')
plt.savefig('../plots/werner-smallsat-nominal-altitude-speed.eps', dpi=300, bbox_
    ↪inches='tight')
```

(continues on next page)

(continued from previous page)

```
plt.show()
```

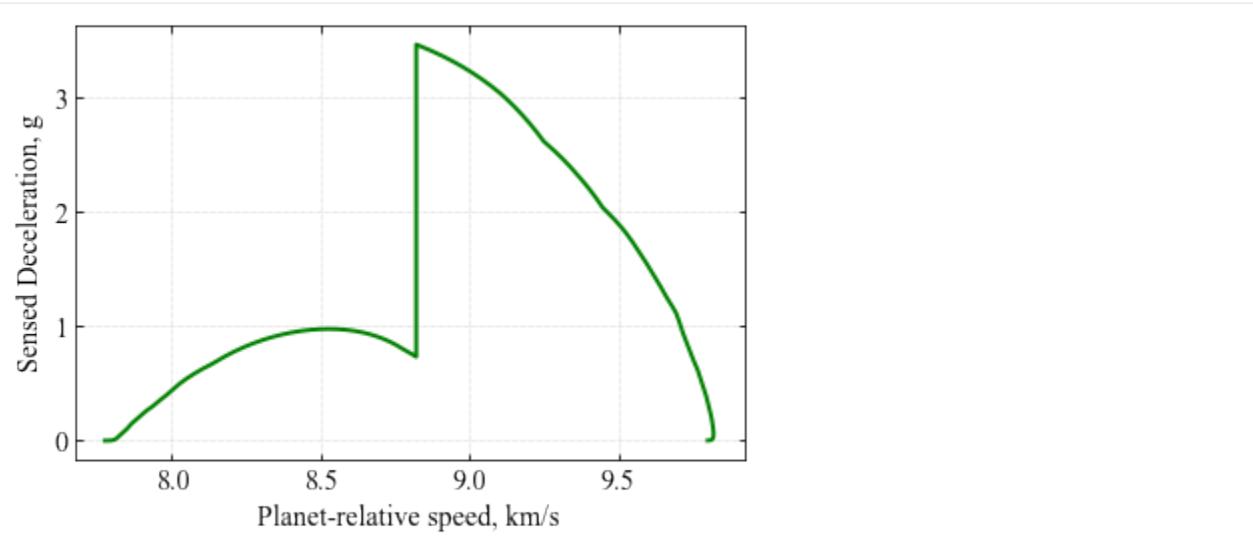


```
[19]: plt.figure(figsize=(6, 4))
plt.rc('font', family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.plot(vehicle.v_kms_full, vehicle.acc_net_g_full, 'g-', linewidth=2.0)

plt.xlabel('Planet-relative speed, km/s', fontsize=14)
plt.ylabel('Sensed Deceleration, g', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)
plt.grid(linestyle='dotted', linewidth=0.5)

plt.savefig('../plots/werner-smallsat-nominal-speed-decel.png', bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-nominal-speed-decel.pdf', dpi=300, bbox_inches=
    ↪'tight')
plt.savefig('../plots/werner-smallsat-nominal-speed-decel.eps', dpi=300, bbox_inches=
    ↪'tight')

plt.show()
```

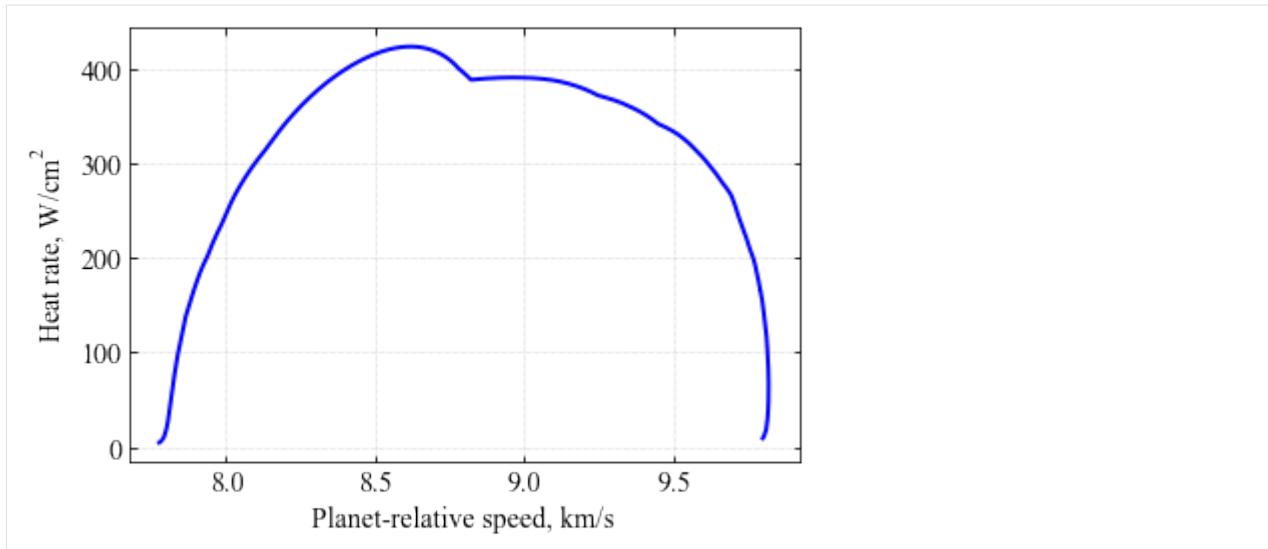


```
[20]: plt.figure(figsize=(6, 4))
plt.rc('font', family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.plot(vehicle.v_kms_full, vehicle.q_stag_total_full, 'b-', linewidth=2.0)

plt.xlabel('Planet-relative speed, km/s', fontsize=14)
plt.ylabel('Heat rate, '+r'$W/cm^2$', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)
plt.grid(linestyle='dotted', linewidth=0.5)

plt.savefig('../plots/werner-smallsat-nominal-speed-heat.png', bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-nominal-speed-heat.pdf', dpi=300, bbox_inches=
    'tight')
plt.savefig('../plots/werner-smallsat-nominal-speed-heat.eps', dpi=300, bbox_inches=
    'tight')

plt.show()
```



```
[21]: vehicle.terminal_apoapsis
```

```
[21]: 1759.9133624138142
```

4.50 Example - 50 - Earth SmallSat Aerocapture Demonstration - Part 2

In this example, we will use Monte Carlo simulations to assess flight performance. Reference: M.S.Werner and R.D.Braun, Mission Design and Performance Analysis of a Smallsat Aerocapture Flight Test, Journal of Spacecraft and Rockets, DOI: 10.2514/1.A33997

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```



```
[2]: import numpy as np
from scipy import interpolate
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.patches import Polygon
```



```
[3]: # Set up the planet and atmosphere model.
planet=Planet("EARTH")
planet.loadAtmosphereModel('../atmdata/Earth/earth-gram-avg.dat', 0 , 1 , 2, 3)
planet.h_skip = 125000.0
planet.h_low=10.0E3

# Set up the drag modulation vehicle.
vehicle=Vehicle('EarthSmallSat', 2.82, 66.4, 0.0, np.pi*0.25**2, 0.0, 0.0563, planet)
vehicle.setInitialState(125.0,0.0,0.0,9.8,0.0,-5.00,0.0,0.0)
vehicle.setSolverParams(1E-6)
vehicle.setDragModulationVehicleParams(66.4,4.72)
```

(continues on next page)

(continued from previous page)

```

# Set up the drag modulation entry phase guidance parameters.
vehicle.setDragEntryPhaseParams(5.0, 50.0, 101, -200.0)

# Set the target orbit parameters.
vehicle.setTargetOrbitParams(180.0, 1760.0, 50.0)

# Define the path to atmospheric files to be used for the Monte Carlo simulations.
#atmfiles = ['../atmdata/Earth/LAT00N.txt']

atmfiles = ['../atmdata/Earth/LAT00N.txt',
            '../atmdata/Earth/LAT20N.txt',
            '../atmdata/Earth/LAT20S.txt',
            '../atmdata/Earth/LAT40N.txt',
            '../atmdata/Earth/LAT40S.txt',
            '../atmdata/Earth/LAT60N.txt',
            '../atmdata/Earth/LAT60S.txt',
            '../atmdata/Earth/LAT80N.txt',
            '../atmdata/Earth/LAT80S.txt']

# Set up the Monte Carlo simulation for drag modulation.
# NPOS = 141, NMONTE = 200
# Target EFPA = -4.85 deg
# EFPA 1-sigma error = +/- 0.04 deg
# Nominal beta_1 = 66.4 kg/m2
# beta_1 1-sigma = 0.0
# guidance time step for entry = 1.0s (Freq. = 1 Hz)
# guidance time step after jettison = 1.0 s
# max. solver time step = 0.1 s
# max. time used by solver = 2400 s

vehicle.setupMonteCarloSimulationD_Earth(141, 200, atmfiles, 1, 4, 14, 9, True,
                                         -4.85, 0.04, 66.4, 0.0,
                                         1.0, 1.0, 0.1, 2400.0)

# Run 200 trajectories
vehicle.runMonteCarloD_Earth(200, '../data/werner2019/MCB1')

BATCH :../data/werner2019/MCB1, RUN #: 1, PROF: ../atmdata/Earth/LAT40S.txt, SAMPLE #:
  ↵ 107, EFPA: -4.83, SIGMA: 0.22, APO : 1915.51
BATCH :../data/werner2019/MCB1, RUN #: 2, PROF: ../atmdata/Earth/LAT80S.txt, SAMPLE #:
  ↵ 14, EFPA: -4.79, SIGMA: -1.00, APO : 2874.42
BATCH :../data/werner2019/MCB1, RUN #: 3, PROF: ../atmdata/Earth/LAT20S.txt, SAMPLE #:
  ↵ 138, EFPA: -4.84, SIGMA: 0.54, APO : 1901.08
BATCH :../data/werner2019/MCB1, RUN #: 4, PROF: ../atmdata/Earth/LAT40S.txt, SAMPLE #:
  ↵ 126, EFPA: -4.83, SIGMA: 0.10, APO : 1877.14
BATCH :../data/werner2019/MCB1, RUN #: 5, PROF: ../atmdata/Earth/LAT80S.txt, SAMPLE #:
  ↵ 138, EFPA: -4.85, SIGMA: 0.23, APO : 1676.20
BATCH :../data/werner2019/MCB1, RUN #: 6, PROF: ../atmdata/Earth/LAT20S.txt, SAMPLE #:
  ↵ 18, EFPA: -4.99, SIGMA: -1.71, APO : 1533.84
BATCH :../data/werner2019/MCB1, RUN #: 7, PROF: ../atmdata/Earth/LAT60N.txt, SAMPLE #:
  ↵ 22, EFPA: -4.93, SIGMA: 0.05, APO : 1855.16
BATCH :../data/werner2019/MCB1, RUN #: 8, PROF: ../atmdata/Earth/LAT00N.txt, SAMPLE #:
  ↵ 38, EFPA: -4.89, SIGMA: -0.84, APO : 2005.10
BATCH :../data/werner2019/MCB1, RUN #: 9, PROF: ../atmdata/Earth/LAT80S.txt, SAMPLE #:
  ↵ 187, EFPA: -4.79, SIGMA: -0.30, APO : 2394.91
BATCH :../data/werner2019/MCB1, RUN #: 10, PROF: ../atmdata/Earth/LAT80S.txt, SAMPLE #:
  ↵ #: 123, EFPA: -4.77, SIGMA: 0.85, APO : 2404.43
BATCH :../data/werner2019/MCB1, RUN #: 11, PROF: ../atmdata/Earth/LAT80S.txt, SAMPLE #:
  ↵ #: 143, EFPA: -4.84, SIGMA: -0.55, APO : 2553.98

```

(continues on next page)

(continued from previous page)

```
BATCH :./data/werner2019/MCB1, RUN #: 12, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 82, EFPA: -4.82, SIGMA: -1.59, APO : 1813.22
BATCH :./data/werner2019/MCB1, RUN #: 13, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 82, EFPA: -4.87, SIGMA: 0.05, APO : 1944.94
BATCH :./data/werner2019/MCB1, RUN #: 14, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 77, EFPA: -4.82, SIGMA: 1.32, APO : 1789.71
BATCH :./data/werner2019/MCB1, RUN #: 15, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 32, EFPA: -4.88, SIGMA: 1.99, APO : 1788.18
BATCH :./data/werner2019/MCB1, RUN #: 16, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 186, EFPA: -4.89, SIGMA: 0.75, APO : 2094.33
BATCH :./data/werner2019/MCB1, RUN #: 17, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 160, EFPA: -4.80, SIGMA: 0.80, APO : 2229.93
BATCH :./data/werner2019/MCB1, RUN #: 18, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 196, EFPA: -4.91, SIGMA: -0.04, APO : 1757.71
BATCH :./data/werner2019/MCB1, RUN #: 19, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 134, EFPA: -4.87, SIGMA: -1.32, APO : 2048.48
BATCH :./data/werner2019/MCB1, RUN #: 20, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 75, EFPA: -4.83, SIGMA: 0.33, APO : 2973.29
BATCH :./data/werner2019/MCB1, RUN #: 21, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 132, EFPA: -4.88, SIGMA: -0.51, APO : 2467.30
BATCH :./data/werner2019/MCB1, RUN #: 22, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 177, EFPA: -4.75, SIGMA: -0.25, APO : 1815.59
BATCH :./data/werner2019/MCB1, RUN #: 23, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 107, EFPA: -4.90, SIGMA: 2.15, APO : 1641.88
BATCH :./data/werner2019/MCB1, RUN #: 24, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 77, EFPA: -4.87, SIGMA: 0.79, APO : 2346.24
BATCH :./data/werner2019/MCB1, RUN #: 25, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 56, EFPA: -4.79, SIGMA: 0.52, APO : 2748.76
BATCH :./data/werner2019/MCB1, RUN #: 26, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 163, EFPA: -4.81, SIGMA: 0.91, APO : 1921.23
BATCH :./data/werner2019/MCB1, RUN #: 27, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 198, EFPA: -4.85, SIGMA: -1.79, APO : 1978.05
BATCH :./data/werner2019/MCB1, RUN #: 28, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 180, EFPA: -4.89, SIGMA: -0.43, APO : 1856.87
BATCH :./data/werner2019/MCB1, RUN #: 29, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 171, EFPA: -4.83, SIGMA: 0.45, APO : 1940.96
BATCH :./data/werner2019/MCB1, RUN #: 30, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 38, EFPA: -4.80, SIGMA: 0.23, APO : 1808.58
BATCH :./data/werner2019/MCB1, RUN #: 31, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 1, EFPA: -4.85, SIGMA: -1.00, APO : 2002.24
BATCH :./data/werner2019/MCB1, RUN #: 32, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 49, EFPA: -4.84, SIGMA: 0.04, APO : 2074.28
BATCH :./data/werner2019/MCB1, RUN #: 33, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 105, EFPA: -4.85, SIGMA: -0.88, APO : 2267.95
BATCH :./data/werner2019/MCB1, RUN #: 34, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 54, EFPA: -4.79, SIGMA: 3.02, APO : 1817.77
BATCH :./data/werner2019/MCB1, RUN #: 35, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 142, EFPA: -4.89, SIGMA: -1.61, APO : 959.84
BATCH :./data/werner2019/MCB1, RUN #: 36, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 43, EFPA: -4.84, SIGMA: -1.32, APO : 1808.74
BATCH :./data/werner2019/MCB1, RUN #: 37, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 117, EFPA: -4.83, SIGMA: 0.46, APO : 1806.22
BATCH :./data/werner2019/MCB1, RUN #: 38, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 126, EFPA: -4.97, SIGMA: -0.78, APO : 1885.88
BATCH :./data/werner2019/MCB1, RUN #: 39, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 73, EFPA: -4.92, SIGMA: 1.39, APO : 2295.27
BATCH :./data/werner2019/MCB1, RUN #: 40, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 41, EFPA: -4.79, SIGMA: -0.06, APO : 1910.13
```

(continues on next page)

(continued from previous page)

```

BATCH :./data/werner2019/MCB1, RUN #: 41, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 29, EFPA: -4.84, SIGMA: 0.12, APO : 1860.60
BATCH :./data/werner2019/MCB1, RUN #: 42, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 58, EFPA: -4.81, SIGMA: 0.82, APO : 1810.06
BATCH :./data/werner2019/MCB1, RUN #: 43, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 88, EFPA: -4.86, SIGMA: -0.66, APO : 5832.76
BATCH :./data/werner2019/MCB1, RUN #: 44, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 164, EFPA: -4.82, SIGMA: 0.47, APO : 3232.68
BATCH :./data/werner2019/MCB1, RUN #: 45, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 93, EFPA: -4.84, SIGMA: 0.51, APO : 1893.28
BATCH :./data/werner2019/MCB1, RUN #: 46, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 141, EFPA: -4.87, SIGMA: -0.69, APO : 2347.15
BATCH :./data/werner2019/MCB1, RUN #: 47, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 115, EFPA: -4.82, SIGMA: -2.88, APO : 2131.86
BATCH :./data/werner2019/MCB1, RUN #: 48, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 96, EFPA: -4.84, SIGMA: 0.66, APO : 2325.42
BATCH :./data/werner2019/MCB1, RUN #: 49, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 183, EFPA: -4.83, SIGMA: -0.20, APO : 1924.36
BATCH :./data/werner2019/MCB1, RUN #: 50, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 14, EFPA: -4.80, SIGMA: 0.36, APO : 1848.65
BATCH :./data/werner2019/MCB1, RUN #: 51, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 148, EFPA: -4.78, SIGMA: 0.50, APO : 2519.58
BATCH :./data/werner2019/MCB1, RUN #: 52, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 4, EFPA: -4.86, SIGMA: 0.24, APO : 1934.50
BATCH :./data/werner2019/MCB1, RUN #: 53, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 44, EFPA: -4.89, SIGMA: -1.57, APO : 2895.36
BATCH :./data/werner2019/MCB1, RUN #: 54, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 196, EFPA: -4.88, SIGMA: 0.86, APO : 1813.61
BATCH :./data/werner2019/MCB1, RUN #: 55, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 93, EFPA: -4.87, SIGMA: 0.60, APO : 1830.42
BATCH :./data/werner2019/MCB1, RUN #: 56, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 73, EFPA: -4.92, SIGMA: -0.96, APO : 1713.14
BATCH :./data/werner2019/MCB1, RUN #: 57, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 175, EFPA: -4.81, SIGMA: 0.65, APO : 1807.05
BATCH :./data/werner2019/MCB1, RUN #: 58, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 77, EFPA: -4.82, SIGMA: 0.28, APO : 1840.52
BATCH :./data/werner2019/MCB1, RUN #: 59, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 110, EFPA: -4.82, SIGMA: 0.26, APO : 1889.03
BATCH :./data/werner2019/MCB1, RUN #: 60, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 174, EFPA: -4.79, SIGMA: -0.74, APO : 2839.38
BATCH :./data/werner2019/MCB1, RUN #: 61, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 151, EFPA: -4.92, SIGMA: -1.81, APO : 1810.20
BATCH :./data/werner2019/MCB1, RUN #: 62, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 93, EFPA: -4.82, SIGMA: 1.75, APO : 1809.73
BATCH :./data/werner2019/MCB1, RUN #: 63, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 52, EFPA: -4.87, SIGMA: 0.78, APO : 1810.62
BATCH :./data/werner2019/MCB1, RUN #: 64, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 67, EFPA: -4.86, SIGMA: -0.45, APO : 1700.97
BATCH :./data/werner2019/MCB1, RUN #: 65, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 12, EFPA: -4.82, SIGMA: 0.32, APO : 2646.75
BATCH :./data/werner2019/MCB1, RUN #: 66, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 183, EFPA: -4.90, SIGMA: -0.44, APO : 333.28
BATCH :./data/werner2019/MCB1, RUN #: 67, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 59, EFPA: -4.90, SIGMA: -0.17, APO : 1692.33
BATCH :./data/werner2019/MCB1, RUN #: 68, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 18, EFPA: -4.84, SIGMA: 0.51, APO : 1928.06
BATCH :./data/werner2019/MCB1, RUN #: 69, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 25, EFPA: -4.85, SIGMA: 0.94, APO : 1940.93

```

(continues on next page)

(continued from previous page)

```
BATCH :./data/werner2019/MCB1, RUN #: 70, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 151, EFPA: -4.88, SIGMA: 1.75, APO : 1873.31
BATCH :./data/werner2019/MCB1, RUN #: 71, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 130, EFPA: -4.84, SIGMA: -0.57, APO : 2509.63
BATCH :./data/werner2019/MCB1, RUN #: 72, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 33, EFPA: -4.90, SIGMA: 0.97, APO : 1824.60
BATCH :./data/werner2019/MCB1, RUN #: 73, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 154, EFPA: -4.88, SIGMA: 0.57, APO : 1485.52
BATCH :./data/werner2019/MCB1, RUN #: 74, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 21, EFPA: -4.82, SIGMA: -1.09, APO : 1833.94
BATCH :./data/werner2019/MCB1, RUN #: 75, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 27, EFPA: -4.85, SIGMA: -0.38, APO : 2849.74
BATCH :./data/werner2019/MCB1, RUN #: 76, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 11, EFPA: -4.90, SIGMA: 0.87, APO : 2129.86
BATCH :./data/werner2019/MCB1, RUN #: 77, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 149, EFPA: -4.87, SIGMA: 0.24, APO : 1729.69
BATCH :./data/werner2019/MCB1, RUN #: 78, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 20, EFPA: -4.86, SIGMA: 0.11, APO : 1778.55
BATCH :./data/werner2019/MCB1, RUN #: 79, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 61, EFPA: -4.85, SIGMA: -0.36, APO : 1967.42
BATCH :./data/werner2019/MCB1, RUN #: 80, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 151, EFPA: -4.80, SIGMA: -0.53, APO : 1867.09
BATCH :./data/werner2019/MCB1, RUN #: 81, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 9, EFPA: -4.87, SIGMA: 0.24, APO : 1780.05
BATCH :./data/werner2019/MCB1, RUN #: 82, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 78, EFPA: -4.78, SIGMA: -0.15, APO : 4029.84
BATCH :./data/werner2019/MCB1, RUN #: 83, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 14, EFPA: -4.88, SIGMA: -0.21, APO : 124.35
BATCH :./data/werner2019/MCB1, RUN #: 84, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 152, EFPA: -4.80, SIGMA: 0.27, APO : 1906.28
BATCH :./data/werner2019/MCB1, RUN #: 85, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 141, EFPA: -4.88, SIGMA: -0.44, APO : 1893.22
BATCH :./data/werner2019/MCB1, RUN #: 86, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 171, EFPA: -4.85, SIGMA: 0.90, APO : 706.47
BATCH :./data/werner2019/MCB1, RUN #: 87, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 87, EFPA: -4.82, SIGMA: 0.13, APO : 1825.07
BATCH :./data/werner2019/MCB1, RUN #: 88, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 70, EFPA: -4.83, SIGMA: -1.14, APO : 1918.20
BATCH :./data/werner2019/MCB1, RUN #: 89, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 166, EFPA: -4.83, SIGMA: -0.06, APO : 1752.07
BATCH :./data/werner2019/MCB1, RUN #: 90, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 40, EFPA: -4.84, SIGMA: -0.77, APO : 1837.95
BATCH :./data/werner2019/MCB1, RUN #: 91, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 189, EFPA: -4.87, SIGMA: -0.62, APO : 1843.50
BATCH :./data/werner2019/MCB1, RUN #: 92, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 127, EFPA: -4.87, SIGMA: -0.79, APO : 2685.77
BATCH :./data/werner2019/MCB1, RUN #: 93, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 131, EFPA: -4.85, SIGMA: 0.13, APO : 1829.54
BATCH :./data/werner2019/MCB1, RUN #: 94, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 97, EFPA: -4.86, SIGMA: 0.20, APO : 1769.45
BATCH :./data/werner2019/MCB1, RUN #: 95, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 37, EFPA: -4.84, SIGMA: 0.08, APO : 2547.89
BATCH :./data/werner2019/MCB1, RUN #: 96, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 14, EFPA: -4.88, SIGMA: -1.27, APO : 2019.88
BATCH :./data/werner2019/MCB1, RUN #: 97, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 53, EFPA: -4.91, SIGMA: 0.98, APO : 1816.88
BATCH :./data/werner2019/MCB1, RUN #: 98, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 197, EFPA: -4.82, SIGMA: 0.22, APO : 1804.41
```

(continues on next page)

(continued from previous page)

```

BATCH :./data/werner2019/MCB1, RUN #: 99, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 195, EFPA: -4.79, SIGMA: 0.75, APO : 1966.88
BATCH :./data/werner2019/MCB1, RUN #: 100, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 107, EFPA: -4.89, SIGMA: -1.37, APO : 2458.49
BATCH :./data/werner2019/MCB1, RUN #: 101, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 1, EFPA: -4.88, SIGMA: -2.25, APO : 5353.50
BATCH :./data/werner2019/MCB1, RUN #: 102, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 40, EFPA: -4.89, SIGMA: -0.66, APO : 2664.62
BATCH :./data/werner2019/MCB1, RUN #: 103, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 114, EFPA: -4.91, SIGMA: 0.64, APO : 1700.72
BATCH :./data/werner2019/MCB1, RUN #: 104, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 13, EFPA: -4.92, SIGMA: -1.19, APO : 22.63
BATCH :./data/werner2019/MCB1, RUN #: 105, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 132, EFPA: -4.78, SIGMA: 0.83, APO : 1815.90
BATCH :./data/werner2019/MCB1, RUN #: 106, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 25, EFPA: -4.88, SIGMA: 0.37, APO : 2613.84
BATCH :./data/werner2019/MCB1, RUN #: 107, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 184, EFPA: -4.86, SIGMA: -0.02, APO : 1823.35
BATCH :./data/werner2019/MCB1, RUN #: 108, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 52, EFPA: -4.83, SIGMA: 0.86, APO : 2309.78
BATCH :./data/werner2019/MCB1, RUN #: 109, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 48, EFPA: -4.87, SIGMA: -0.43, APO : 1908.16
BATCH :./data/werner2019/MCB1, RUN #: 110, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 89, EFPA: -4.84, SIGMA: -1.45, APO : 2984.50
BATCH :./data/werner2019/MCB1, RUN #: 111, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 96, EFPA: -4.83, SIGMA: 0.13, APO : 1895.90
BATCH :./data/werner2019/MCB1, RUN #: 112, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 176, EFPA: -4.90, SIGMA: 1.10, APO : 22.69
BATCH :./data/werner2019/MCB1, RUN #: 113, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 67, EFPA: -4.85, SIGMA: 0.49, APO : 1760.06
BATCH :./data/werner2019/MCB1, RUN #: 114, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 72, EFPA: -4.83, SIGMA: 1.05, APO : 1822.80
BATCH :./data/werner2019/MCB1, RUN #: 115, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 84, EFPA: -4.84, SIGMA: -0.73, APO : 2175.26
BATCH :./data/werner2019/MCB1, RUN #: 116, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 171, EFPA: -4.83, SIGMA: 1.08, APO : 1878.67
BATCH :./data/werner2019/MCB1, RUN #: 117, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 68, EFPA: -4.86, SIGMA: -1.16, APO : 1808.78
BATCH :./data/werner2019/MCB1, RUN #: 118, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 200, EFPA: -4.84, SIGMA: 1.32, APO : 1766.54
BATCH :./data/werner2019/MCB1, RUN #: 119, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 51, EFPA: -4.80, SIGMA: 0.01, APO : 1793.98
BATCH :./data/werner2019/MCB1, RUN #: 120, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 14, EFPA: -4.81, SIGMA: -0.63, APO : 2147.00
BATCH :./data/werner2019/MCB1, RUN #: 121, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 136, EFPA: -4.87, SIGMA: -0.15, APO : 3850.89
BATCH :./data/werner2019/MCB1, RUN #: 122, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 10, EFPA: -4.80, SIGMA: -0.83, APO : 3750.97
BATCH :./data/werner2019/MCB1, RUN #: 123, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 193, EFPA: -4.91, SIGMA: 0.53, APO : 22.77
BATCH :./data/werner2019/MCB1, RUN #: 124, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 112, EFPA: -4.84, SIGMA: -0.21, APO : 1965.62
BATCH :./data/werner2019/MCB1, RUN #: 125, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 135, EFPA: -4.89, SIGMA: 0.06, APO : 22.54
BATCH :./data/werner2019/MCB1, RUN #: 126, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 164, EFPA: -4.84, SIGMA: 0.21, APO : 1941.02
BATCH :./data/werner2019/MCB1, RUN #: 127, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 17, EFPA: -4.87, SIGMA: -1.32, APO : 2672.25

```

(continues on next page)

(continued from previous page)

```
BATCH :./data/werner2019/MCB1, RUN #: 128, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 124, EFPA: -4.80, SIGMA: 1.49, APO : 1803.72
BATCH :./data/werner2019/MCB1, RUN #: 129, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 195, EFPA: -4.80, SIGMA: 0.22, APO : 1835.42
BATCH :./data/werner2019/MCB1, RUN #: 130, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 102, EFPA: -4.80, SIGMA: 1.93, APO : 1835.56
BATCH :./data/werner2019/MCB1, RUN #: 131, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 18, EFPA: -4.84, SIGMA: -0.79, APO : 1811.93
BATCH :./data/werner2019/MCB1, RUN #: 132, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 4, EFPA: -4.84, SIGMA: -0.69, APO : 2030.44
BATCH :./data/werner2019/MCB1, RUN #: 133, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 37, EFPA: -4.86, SIGMA: 0.16, APO : 1812.47
BATCH :./data/werner2019/MCB1, RUN #: 134, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 181, EFPA: -4.84, SIGMA: -3.70, APO : 1818.59
BATCH :./data/werner2019/MCB1, RUN #: 135, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 4, EFPA: -4.79, SIGMA: -0.71, APO : 1868.37
BATCH :./data/werner2019/MCB1, RUN #: 136, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 184, EFPA: -4.85, SIGMA: -1.87, APO : 1771.29
BATCH :./data/werner2019/MCB1, RUN #: 137, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 102, EFPA: -4.85, SIGMA: -0.48, APO : 2188.19
BATCH :./data/werner2019/MCB1, RUN #: 138, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 99, EFPA: -4.75, SIGMA: -0.85, APO : 9823.91
BATCH :./data/werner2019/MCB1, RUN #: 139, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 138, EFPA: -4.86, SIGMA: 0.28, APO : 1777.30
BATCH :./data/werner2019/MCB1, RUN #: 140, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 166, EFPA: -4.83, SIGMA: -0.17, APO : 1900.76
BATCH :./data/werner2019/MCB1, RUN #: 141, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 160, EFPA: -4.89, SIGMA: 0.06, APO : 1798.07
BATCH :./data/werner2019/MCB1, RUN #: 142, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 196, EFPA: -4.88, SIGMA: 0.33, APO : 1885.40
BATCH :./data/werner2019/MCB1, RUN #: 143, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 19, EFPA: -4.89, SIGMA: 2.38, APO : 1298.20
BATCH :./data/werner2019/MCB1, RUN #: 144, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 185, EFPA: -4.90, SIGMA: 0.52, APO : 1942.11
BATCH :./data/werner2019/MCB1, RUN #: 145, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 95, EFPA: -4.83, SIGMA: 2.00, APO : 2038.34
BATCH :./data/werner2019/MCB1, RUN #: 146, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 11, EFPA: -4.85, SIGMA: 0.68, APO : 1814.08
BATCH :./data/werner2019/MCB1, RUN #: 147, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 187, EFPA: -4.90, SIGMA: -0.51, APO : 1868.46
BATCH :./data/werner2019/MCB1, RUN #: 148, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 81, EFPA: -4.85, SIGMA: 0.87, APO : 1835.14
BATCH :./data/werner2019/MCB1, RUN #: 149, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 67, EFPA: -4.78, SIGMA: 1.77, APO : 2056.58
BATCH :./data/werner2019/MCB1, RUN #: 150, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 47, EFPA: -4.84, SIGMA: 0.08, APO : 2000.82
BATCH :./data/werner2019/MCB1, RUN #: 151, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 28, EFPA: -4.80, SIGMA: 0.33, APO : 4423.67
BATCH :./data/werner2019/MCB1, RUN #: 152, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 14, EFPA: -4.89, SIGMA: -0.18, APO : 1826.45
BATCH :./data/werner2019/MCB1, RUN #: 153, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 35, EFPA: -4.83, SIGMA: 0.58, APO : 1913.70
BATCH :./data/werner2019/MCB1, RUN #: 154, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 49, EFPA: -4.83, SIGMA: 0.04, APO : 1839.21
BATCH :./data/werner2019/MCB1, RUN #: 155, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 99, EFPA: -4.85, SIGMA: -1.20, APO : 1911.89
BATCH :./data/werner2019/MCB1, RUN #: 156, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 93, EFPA: -4.91, SIGMA: -2.07, APO : 2050.29
```

(continues on next page)

(continued from previous page)

```

BATCH :./data/werner2019/MCB1, RUN #: 157, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 150, EFPA: -4.82, SIGMA: 0.80, APO : 2043.39
BATCH :./data/werner2019/MCB1, RUN #: 158, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 93, EFPA: -4.85, SIGMA: -0.89, APO : 2905.72
BATCH :./data/werner2019/MCB1, RUN #: 159, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 119, EFPA: -4.87, SIGMA: 1.05, APO : 1762.34
BATCH :./data/werner2019/MCB1, RUN #: 160, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 88, EFPA: -4.86, SIGMA: 0.97, APO : 1768.09
BATCH :./data/werner2019/MCB1, RUN #: 161, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 137, EFPA: -4.82, SIGMA: 0.27, APO : 1874.13
BATCH :./data/werner2019/MCB1, RUN #: 162, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 113, EFPA: -4.85, SIGMA: -0.48, APO : 1808.50
BATCH :./data/werner2019/MCB1, RUN #: 163, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 90, EFPA: -4.82, SIGMA: -0.51, APO : 4621.58
BATCH :./data/werner2019/MCB1, RUN #: 164, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 119, EFPA: -4.93, SIGMA: -0.25, APO : 22.55
BATCH :./data/werner2019/MCB1, RUN #: 165, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 188, EFPA: -4.80, SIGMA: 0.77, APO : 2672.75
BATCH :./data/werner2019/MCB1, RUN #: 166, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 77, EFPA: -4.93, SIGMA: -0.62, APO : 1391.23
BATCH :./data/werner2019/MCB1, RUN #: 167, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 55, EFPA: -4.88, SIGMA: -0.00, APO : 1905.84
BATCH :./data/werner2019/MCB1, RUN #: 168, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 162, EFPA: -4.86, SIGMA: -0.02, APO : 1887.71
BATCH :./data/werner2019/MCB1, RUN #: 169, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 6, EFPA: -4.88, SIGMA: 0.29, APO : 2017.17
BATCH :./data/werner2019/MCB1, RUN #: 170, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 159, EFPA: -4.84, SIGMA: 0.94, APO : 1814.33
BATCH :./data/werner2019/MCB1, RUN #: 171, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 200, EFPA: -4.87, SIGMA: -0.82, APO : 3258.49
BATCH :./data/werner2019/MCB1, RUN #: 172, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 57, EFPA: -4.93, SIGMA: -1.65, APO : 2096.31
BATCH :./data/werner2019/MCB1, RUN #: 173, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 92, EFPA: -4.90, SIGMA: -0.13, APO : 1450.62
BATCH :./data/werner2019/MCB1, RUN #: 174, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 67, EFPA: -4.85, SIGMA: -1.60, APO : 2194.75
BATCH :./data/werner2019/MCB1, RUN #: 175, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 20, EFPA: -4.82, SIGMA: -1.65, APO : 1855.57
BATCH :./data/werner2019/MCB1, RUN #: 176, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 111, EFPA: -4.85, SIGMA: 0.13, APO : 2629.28
BATCH :./data/werner2019/MCB1, RUN #: 177, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 86, EFPA: -4.90, SIGMA: -0.57, APO : 1953.92
BATCH :./data/werner2019/MCB1, RUN #: 178, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 200, EFPA: -4.82, SIGMA: 0.17, APO : 3527.15
BATCH :./data/werner2019/MCB1, RUN #: 179, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 73, EFPA: -4.80, SIGMA: 0.96, APO : 1870.57
BATCH :./data/werner2019/MCB1, RUN #: 180, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 70, EFPA: -4.85, SIGMA: 0.25, APO : 1813.48
BATCH :./data/werner2019/MCB1, RUN #: 181, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 181, EFPA: -4.91, SIGMA: 0.02, APO : 1800.06
BATCH :./data/werner2019/MCB1, RUN #: 182, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 135, EFPA: -4.86, SIGMA: 1.21, APO : 2210.22
BATCH :./data/werner2019/MCB1, RUN #: 183, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 200, EFPA: -4.83, SIGMA: 0.42, APO : 1796.40
BATCH :./data/werner2019/MCB1, RUN #: 184, PROF: ./atmdata/Earth/LAT40N.txt, SAMPLE
→#: 34, EFPA: -4.84, SIGMA: 1.11, APO : 1853.91
BATCH :./data/werner2019/MCB1, RUN #: 185, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 136, EFPA: -4.93, SIGMA: 0.08, APO : 1870.74

```

(continues on next page)

(continued from previous page)

```
BATCH :./data/werner2019/MCB1, RUN #: 186, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 144, EFPA: -4.90, SIGMA: 0.93, APO : 1764.16
BATCH :./data/werner2019/MCB1, RUN #: 187, PROF: ./atmdata/Earth/LAT80N.txt, SAMPLE
→#: 67, EFPA: -4.88, SIGMA: -2.00, APO : 2258.70
BATCH :./data/werner2019/MCB1, RUN #: 188, PROF: ./atmdata/Earth/LAT80S.txt, SAMPLE
→#: 81, EFPA: -4.92, SIGMA: -0.95, APO : 22.62
BATCH :./data/werner2019/MCB1, RUN #: 189, PROF: ./atmdata/Earth/LAT20N.txt, SAMPLE
→#: 164, EFPA: -4.84, SIGMA: -1.46, APO : 1904.96
BATCH :./data/werner2019/MCB1, RUN #: 190, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 112, EFPA: -4.87, SIGMA: -1.30, APO : 1737.33
BATCH :./data/werner2019/MCB1, RUN #: 191, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 111, EFPA: -4.88, SIGMA: -1.19, APO : 1818.10
BATCH :./data/werner2019/MCB1, RUN #: 192, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 35, EFPA: -4.83, SIGMA: -0.38, APO : 3184.55
BATCH :./data/werner2019/MCB1, RUN #: 193, PROF: ./atmdata/Earth/LAT60S.txt, SAMPLE
→#: 184, EFPA: -4.91, SIGMA: -0.57, APO : 1939.20
BATCH :./data/werner2019/MCB1, RUN #: 194, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 96, EFPA: -4.85, SIGMA: 1.09, APO : 1742.70
BATCH :./data/werner2019/MCB1, RUN #: 195, PROF: ./atmdata/Earth/LAT00N.txt, SAMPLE
→#: 22, EFPA: -4.86, SIGMA: -0.01, APO : 1843.61
BATCH :./data/werner2019/MCB1, RUN #: 196, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 73, EFPA: -4.82, SIGMA: -1.92, APO : 1873.99
BATCH :./data/werner2019/MCB1, RUN #: 197, PROF: ./atmdata/Earth/LAT20S.txt, SAMPLE
→#: 26, EFPA: -4.84, SIGMA: 0.28, APO : 1885.78
BATCH :./data/werner2019/MCB1, RUN #: 198, PROF: ./atmdata/Earth/LAT40S.txt, SAMPLE
→#: 122, EFPA: -4.85, SIGMA: -1.06, APO : 1897.63
BATCH :./data/werner2019/MCB1, RUN #: 199, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 114, EFPA: -4.84, SIGMA: -0.12, APO : 3436.07
BATCH :./data/werner2019/MCB1, RUN #: 200, PROF: ./atmdata/Earth/LAT60N.txt, SAMPLE
→#: 4, EFPA: -4.86, SIGMA: -0.87, APO : 3910.03
```

Load the terminal apoapsis, deceleration and heat rate data.

```
[85]: apoap = np.loadtxt('../data/werner2019/MCB1/terminal_apoapsis_arr.txt')
peria = np.loadtxt('../data/werner2019/MCB1/terminal_periapsis_arr.txt')
decel = np.loadtxt('../data/werner2019/MCB1/acc_net_g_max_arr.txt')
heatr = np.loadtxt('../data/werner2019/MCB1/q_stag_max_arr.txt')
```

Remove cases which resulted in apoapsis < 1000 km or < 6000 km.

```
[86]: del_index_low = np.where(apoap < 1000)
print(del_index_low)

(array([ 34,   65,   82,   85,  103,  111,  122,  124,  163,  187]),)
```

Seven cases resulted in apoapsis less than 1000 km, and are considered to crash.

Remove these entries from the data before plotting.

```
[87]: apoap_clean = np.delete(apoap, del_index_low)
peria_clean = np.delete(peria, del_index_low)
decel_clean = np.delete(decel, del_index_low)
heatr_clean = np.delete(heatr, del_index_low)
```

Also remove cases with apoapsis > 3000 km.

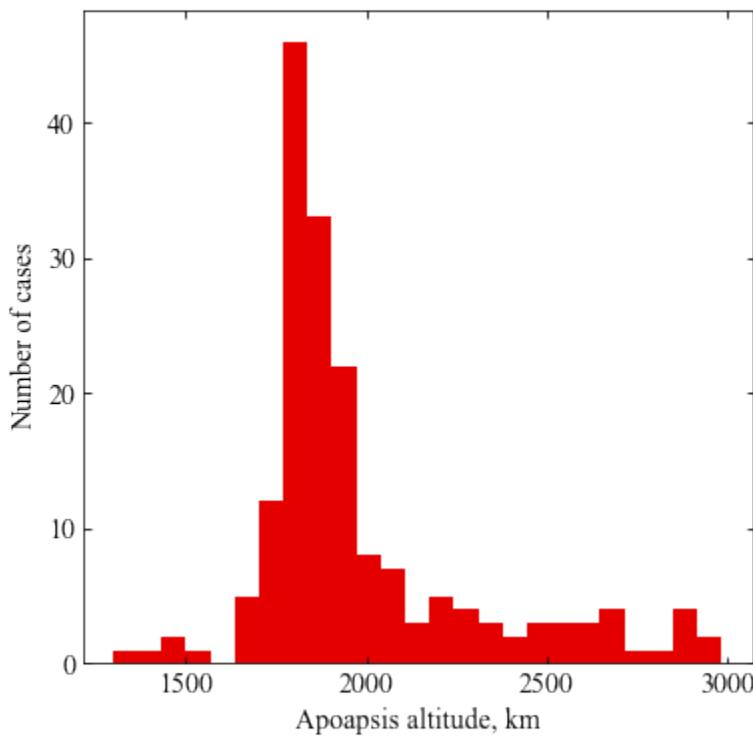
```
[88]: del_index_high = np.where(apoap_clean > 3000)
print(del_index_high)
```

```
(array([ 41,  42,  79,  96, 114, 115, 129, 142, 154, 161, 168, 181, 188,
       189]),)
```

```
[89]: apoap_clean2 = np.delete(apoap_clean, del_index_high)
peria_clean2 = np.delete(peria_clean, del_index_high)
decel_clean2 = np.delete(decel_clean, del_index_high)
heatr_clean2 = np.delete(heatr_clean, del_index_high)
```

```
[95]: plt.figure(figsize=(6,6))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.hist(apoap_clean2, bins=25, color='xkcd:red')
plt.xlabel('Apoapsis altitude, km', fontsize=14)
plt.ylabel('Number of cases', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/werner-smallsat-apoa-hist.png',bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-apoa-hist.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-apoa-hist.eps', dpi=300,bbox_inches='tight')
plt.show()
```



```
[96]: plt.figure(figsize=(6,6))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
```

(continues on next page)

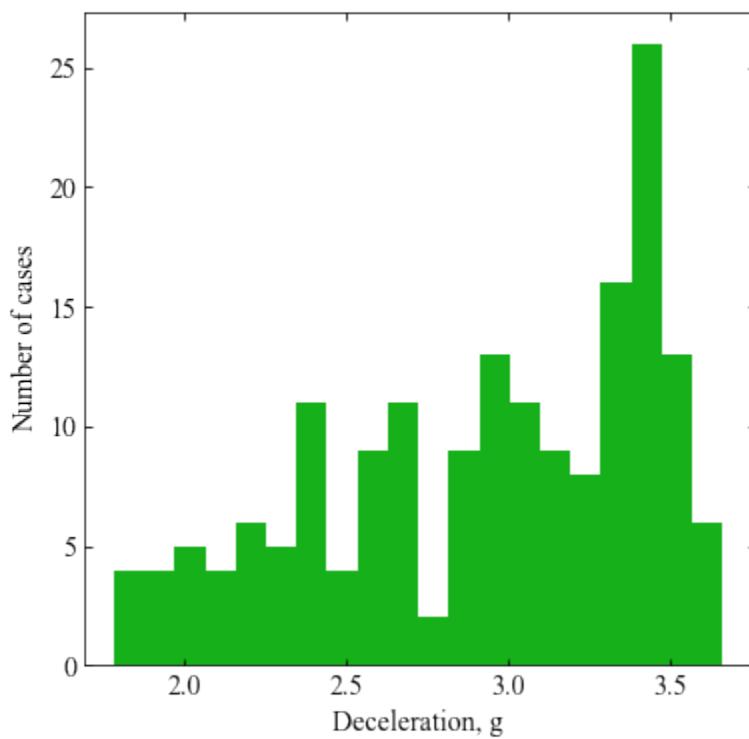
(continued from previous page)

```

plt.rcParams.update(params)
plt.hist(decel_clean2, bins=20, color='xkcd:green')
plt.xlabel('Deceleration, g', fontsize=14)
plt.ylabel('Number of cases', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/werner-smallsat-decel-hist.png', bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-decel-hist.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-decel-hist.eps', dpi=300, bbox_inches='tight')
plt.show()

```



```

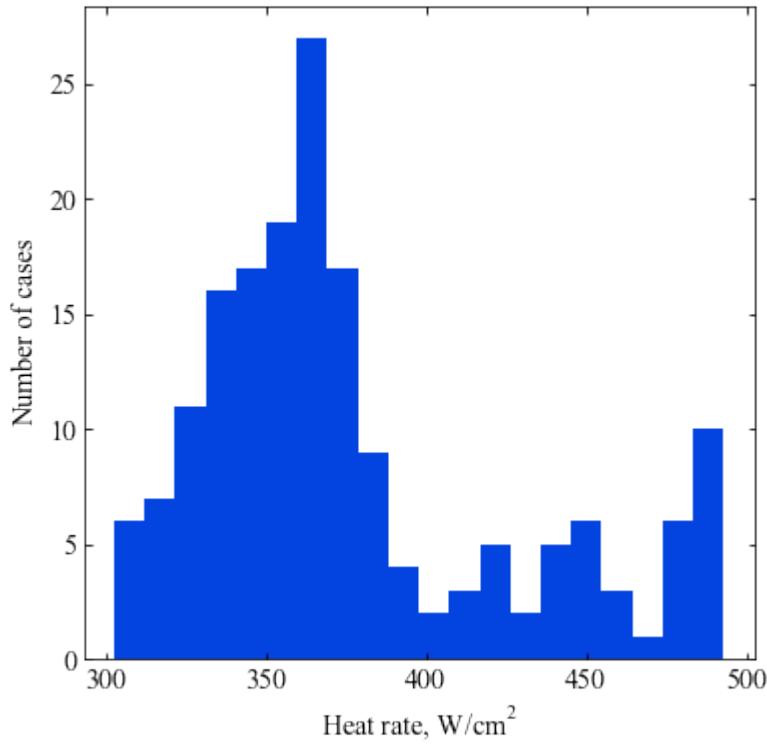
[97]: plt.figure(figsize=(6, 6))
plt.rc('font', family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.hist(heatr_clean2, bins=20, color='xkcd:blue')
plt.xlabel('Heat rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Number of cases', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

```

(continues on next page)

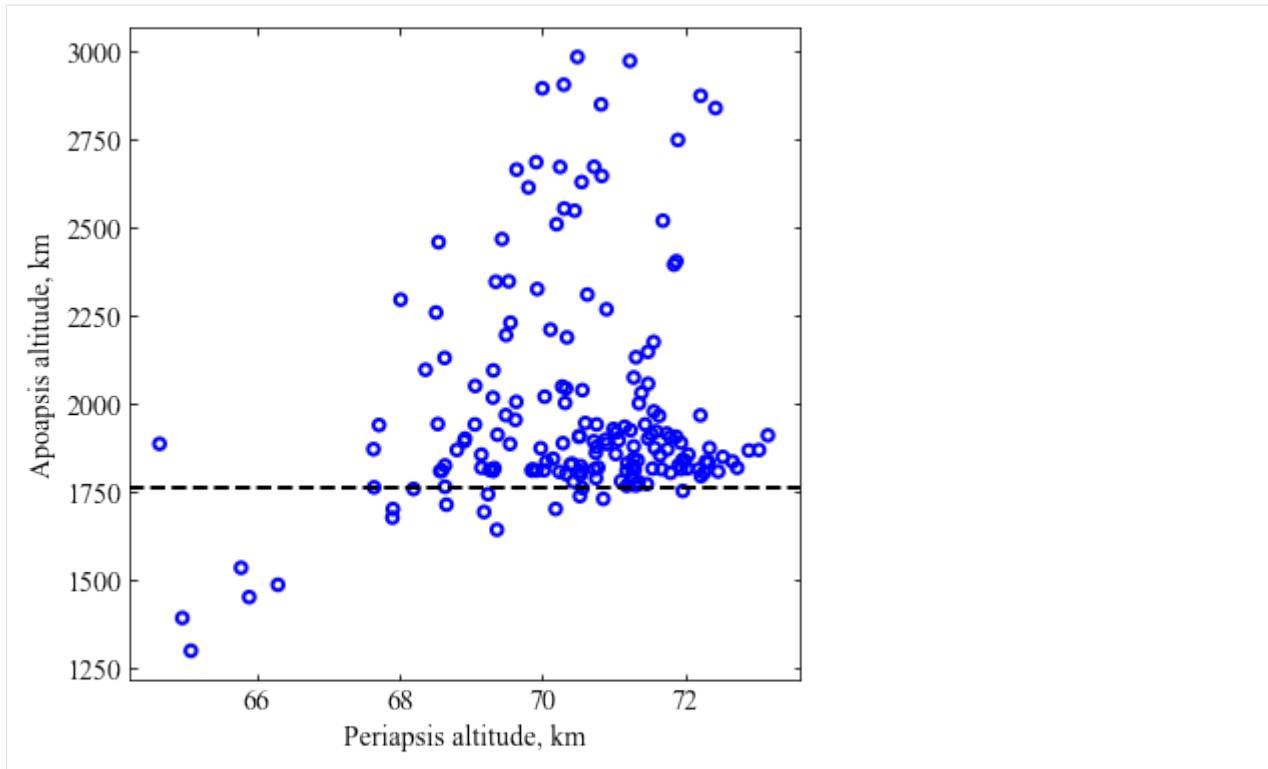
(continued from previous page)

```
plt.savefig('../plots/werner-smallsat-heat-hist.png',bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-heat-hist.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-heat-hist.eps', dpi=300,bbox_inches='tight')
plt.show()
```



```
[98]: plt.figure(figsize=(6,6))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.scatter(peria_clean2, apoap_clean2, s=30, facecolors='none', edgecolors='b', lw=2)
plt.axhline(y=1760, lw=2.0, ls='dashed', color='k')
plt.xlabel('Periapsis altitude, km', fontsize=14)
plt.ylabel('Apoapsis altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.savefig('../plots/werner-smallsat-apoa-peri.png',bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-apoa-peri.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-apoa-peri.eps', dpi=300,bbox_inches='tight')
plt.show()
```



Find the percentage of cases which achieved apoapsis between 1000 and 3000 km.

```
[94]: (np.size(apoap) - np.size(del_index_low) - np.size(del_index_high)) * 100 / np.size(apoap)
[94]: 88.0
```

These results are different from those in the paper because of the different guidance schemes used here. This exercise did not attempt to optimize the guidance parameters. Further study can improve the apoapsis targeting.

4.51 Example - 51 - Mars SmallSat Aerocapture Demonstration - Part 1

This examples reproduces results from M.S.Werner and R.D.Braun, Mission Design and Performance Analysis of a Smallsat Aerocapture Flight Test, Journal of Spacecraft and Rockets, DOI: 10.2514/1.A33997.

```
[1]: from IPython.display import Image
Image(filename='../plots/werner-smallsat.png', width=500)
```

[1]:

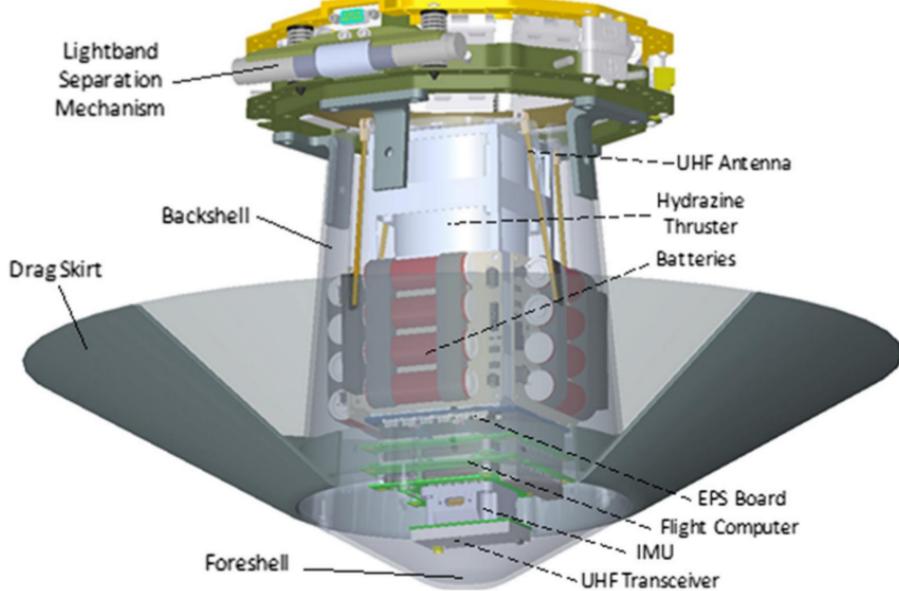


Image credit: M.S.Werner and R.D.Braun

```
[2]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[3]: import numpy as np
from scipy import interpolate
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib import rcParams
```

```
[4]: planet = Planet('MARS')
planet.loadAtmosphereModel('../atmdata/Mars/mars-gram-avg.dat', 0, 1, 2, 3)
planet.h_skip = 150.0E3
```

```
[5]: # Set up two vehicle objects, one for 450 km circular, one for 1-sol orbit
vehicle1=Vehicle('MarsSmallSat1', 25.97, 66.4, 0.0, np.pi*0.25**2, 0.0, 0.0563,_
                  ↵planet)
vehicle2=Vehicle('MarsSmallSat2', 25.97, 66.4, 0.0, np.pi*0.25**2, 0.0, 0.0563,_
                  ↵planet)

vehicle1.setInitialState(150.0,0.0,0.0,5.74,0.0,-5.00,0.0,0.0)
vehicle2.setInitialState(150.0,0.0,0.0,5.74,0.0,-5.00,0.0,0.0)

vehicle1.setSolverParams(1E-6)
vehicle2.setSolverParams(1E-6)

vehicle1.setDragModulationVehicleParams(66.4,4.72)
vehicle2.setDragModulationVehicleParams(66.4,4.72)
```

First, find the corridor bounds to select a target nominal EFPA for a nominal Mars atmosphere.

```
[6]: underShootLimit, exitflag_us = vehicle1.findUnderShootLimitD(2400.0, 0.1, -30.0,-2.0,_
    ↪1E-10, 450.0)
overShootLimit , exitflag_os = vehicle1.findOverShootLimitD(2400.0, 0.1, -30.0,-2.0,_
    ↪1E-10, 450.0)

print("450 km circ.")
print("-----")
print(underShootLimit, exitflag_us)
print(overShootLimit, exitflag_os)
print("-----")
underShootLimit, exitflag_us = vehicle2.findUnderShootLimitD(2400.0, 0.1, -30.0,-2.0,_
    ↪1E-10, 33000.0)
overShootLimit , exitflag_os = vehicle2.findOverShootLimitD(2400.0, 0.1, -30.0,-2.0,_
    ↪1E-10, 33000.0)

print("1 sol.")
print("-----")
print(underShootLimit, exitflag_us)
print(overShootLimit, exitflag_os)
print("-----")

450 km circ.
-----
-12.472027218049334 1.0
-11.777824826203869 1.0
-----
1 sol.
-----
-12.142672350550129 1.0
-11.459108593211567 1.0
-----
```

It is good practice to see how these corridor bounds vary with minimum, average, and maximum density atmospheres. We will check how much these corridor bounds change using +/- 3-sigma bounds from Mars GRAM output files.

Load three density profiles for minimum, average, and maximum scenarios from a GRAM output file.

```
[7]: ATM_height, ATM_density_low, ATM_density_avg, ATM_density_high, ATM_density_pert =_
    ↪planet.loadMonteCarloDensityFile2('../atmdata/Mars/LAT00N.txt', 0, 1, 2, 3, 4,_
    ↪heightInKmFlag=True)
density_int_low = planet.loadAtmosphereModel5(ATM_height, ATM_density_low, ATM_-
    ↪density_avg, ATM_density_high, ATM_density_pert, -3.0, 156, 200)
density_int_avg = planet.loadAtmosphereModel5(ATM_height, ATM_density_low, ATM_-
    ↪density_avg, ATM_density_high, ATM_density_pert, 0.0, 156, 200)
density_int_high = planet.loadAtmosphereModel5(ATM_height, ATM_density_low, ATM_-
    ↪density_avg, ATM_density_high, ATM_density_pert, +3.0, 156, 200)
```

Set the planet density_int attribute to density_int_low

```
[8]: planet.density_int = density_int_low
```

Compute the corridor bounds for low density atmosphere.

```
[9]: underShootLimit, exitflag_us = vehicle1.findUnderShootLimitD(2400.0, 0.1, -30.0,-2.0,_
    ↪1E-10, 450.0)
overShootLimit , exitflag_os = vehicle1.findOverShootLimitD(2400.0, 0.1, -30.0,-2.0,_
    ↪1E-10, 450.0)
```

(continues on next page)

(continued from previous page)

```

print("450 km circ.")
print("-----")
print(underShootLimit, exitflag_us)
print(overShootLimit, exitflag_os)
print("-----")

underShootLimit, exitflag_us = vehicle2.findUnderShootLimitD(2400.0, 0.1, -30.0,-2.0,
                                                               ↵ 1E-10, 33000.0)
overShootLimit , exitflag_os = vehicle2.findOverShootLimitD(2400.0, 0.1, -30.0,-2.0,
                                                               ↵1E-10, 33000.0)
print("1-sol.")
print("-----")
print(underShootLimit, exitflag_us)
print(overShootLimit, exitflag_os)
print("-----")

/home/athul/anaconda3/lib/python3.7/site-packages/AMAT-2.1.2-py3.7.egg/AMAT/vehicle.
→py:504: RuntimeWarning: invalid value encountered in sqrt
ans[:] = 1.8980E-8 * (rho_vec[:]/self.RN)**0.5 * v[:]**3.0

450 km circ.
-----
-12.605845816528017 1.0
-11.993464708328247 1.0
-----
1-sol.
-----
-12.326484654251544 1.0
-11.651159566965362 1.0
-----
```

Repeat for average and maximum density atmospheres.

```
[10]: planet.density_int = density_int_avg

[11]: underShootLimit, exitflag_us = vehicle1.findUnderShootLimitD(2400.0, 0.1, -30.0,-2.0,
                                                               ↵ 1E-10, 450.0)
overShootLimit , exitflag_os = vehicle1.findOverShootLimitD(2400.0, 0.1, -30.0,-2.0,
                                                               ↵1E-10, 450.0)

print("450 km circ.")
print("-----")
print(underShootLimit, exitflag_us)
print(overShootLimit, exitflag_os)
print("-----")

underShootLimit, exitflag_us = vehicle2.findUnderShootLimitD(2400.0, 0.1, -30.0,-2.0,
                                                               ↵ 1E-10, 33000.0)
overShootLimit , exitflag_os = vehicle2.findOverShootLimitD(2400.0, 0.1, -30.0,-2.0,
                                                               ↵1E-10, 33000.0)
print("1-sol.")
print("-----")
print(underShootLimit, exitflag_us)
print(overShootLimit, exitflag_os)
print("-----")

450 km circ.
-----
```

(continues on next page)

(continued from previous page)

```
-12.545852635754272 1.0
-11.894532441678166 1.0
-----
1-sol.
-----
-12.250274132129562 1.0
-11.517179594695335 1.0
```

[12]: planet.density_int = density_int_hig

```
underShootLimit, exitflag_us = vehicle1.findUnderShootLimitD(2400.0, 0.1, -30.0,-2.0,
                                                               ↵ 1E-10, 450.0)
overShootLimit , exitflag_os = vehicle1.findOverShootLimitD(2400.0, 0.1, -30.0,-2.0,
                                                               ↵1E-10, 450.0)

print("450 km circ.")
print("-----")
print(underShootLimit, exitflag_us)
print(overShootLimit, exitflag_os)
print("-----")

underShootLimit, exitflag_us = vehicle2.findUnderShootLimitD(2400.0, 0.1, -30.0,-2.0,
                                                               ↵ 1E-10, 33000.0)
overShootLimit , exitflag_os = vehicle2.findOverShootLimitD(2400.0, 0.1, -30.0,-2.0,
                                                               ↵1E-10, 33000.0)
print("1-sol.")
print("-----")
print(underShootLimit, exitflag_us)
print(overShootLimit, exitflag_os)
print("-----")

450 km circ.
-----
-12.486791579358396 1.0
-11.79002801637398 1.0
-----
1-sol.
-----
-12.178909841590212 1.0
-11.394518235101714 1.0
```

The above numbers indicate that corridor bounds do not vary that much with atmospheric variations. The corridor is approximately 0.70 deg wide for both target orbits.

[14]: 12.47 - 11.77

[14]: 0.7000000000000011

[15]: 12.14 - 11.45

[15]: 0.6900000000000013

The mid-corridor is typically a good place to start. We will use the mean of the corridor bounds for the average atmosphere.

```
[16]: 0.5*(-12.47-11.77) # 450 km
[16]: -12.120000000000001
```

```
[17]: 0.5*(-12.14-11.45) # 1-sol
[17]: -11.795
```

We will propagate a nominal guided trajectory to reproduce Fig. 10 from the paper.

```
[18]: planet.loadAtmosphereModel('../atmdata/Mars/mars-gram-avg.dat', 0, 1, 2, 3)
```

```
[19]: # Set planet.h_low to 10 km, if vehicle dips below this level
# trajectory is terminated.
planet.h_low=10.0E3

# Set target orbit = 450 km x 450 km, tolerance = 50 km
vehicle1.setTargetOrbitParams(450.0, 450.0, 20.0)
vehicle2.setTargetOrbitParams(450.0, 33000.0, 20.0)

# Set entry phase parameters
# v_switch_kms = 2.0, lowAlt_km = 20.0,
# numPoints_lowAlt = 101, hdot_threshold = -200.0 m/s.
# These are somewhat arbitrary based on experience.
vehicle1.setDragEntryPhaseParams(2.0, 20.0, 101, -200.0)
vehicle2.setDragEntryPhaseParams(2.0, 20.0, 101, -200.0)

# Set beta_1 and beta_ratio
vehicle1.setDragModulationVehicleParams(66.4, 4.72)
vehicle2.setDragModulationVehicleParams(66.4, 4.72)

# Set vehicle initial state
vehicle1.setInitialState(150.0, 0.0, 0.0, 5.74, 0.0, -12.12, 0.0, 0.0)
vehicle2.setInitialState(150.0, 0.0, 0.0, 5.74, 0.0, -11.80, 0.0, 0.0)
```

```
[20]: # Propogate a single vehicle trajectory
vehicle1.propogateGuidedEntryD(1.0, 1.0, 0.1, 2400.0)
vehicle2.propogateGuidedEntryD(1.0, 1.0, 0.1, 2400.0)
```

```
[21]: plt.figure(figsize=(6,4))
plt.rc('font', family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.plot(vehicle1.v_kms_full, vehicle1.h_km_full, 'r-', linewidth=2.0)
plt.plot(vehicle2.v_kms_full, vehicle2.h_km_full, 'r--', linewidth=2.0)

plt.xlabel('Planet-relative speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)
plt.grid(linestyle='dotted', linewidth=0.5)

plt.savefig('../plots/werner-smallsat-nominal-altitude-speed-mars.png', bbox_inches=
    'tight')
```

(continues on next page)

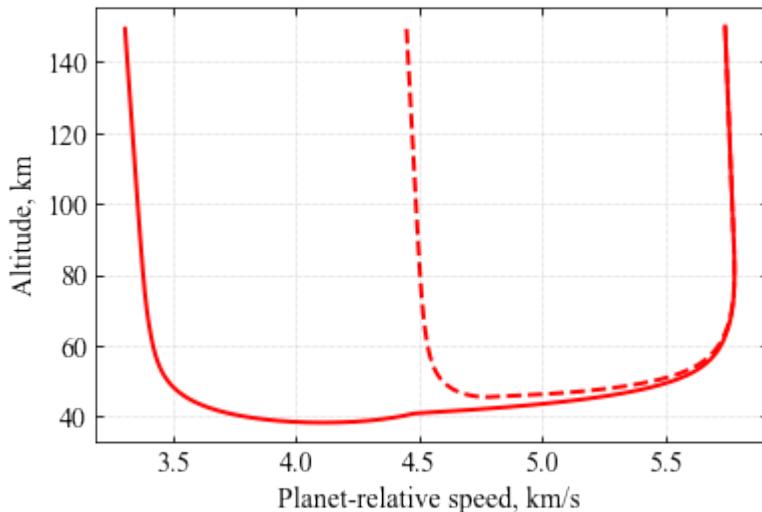
(continued from previous page)

```

plt.savefig('../plots/werner-smallsat-nominal-altitude-speed-mars.pdf', dpi=300, bbox_
    ↪inches='tight')
plt.savefig('../plots/werner-smallsat-nominal-altitude-speed-mars.eps', dpi=300, bbox_
    ↪inches='tight')

plt.show()

```



```

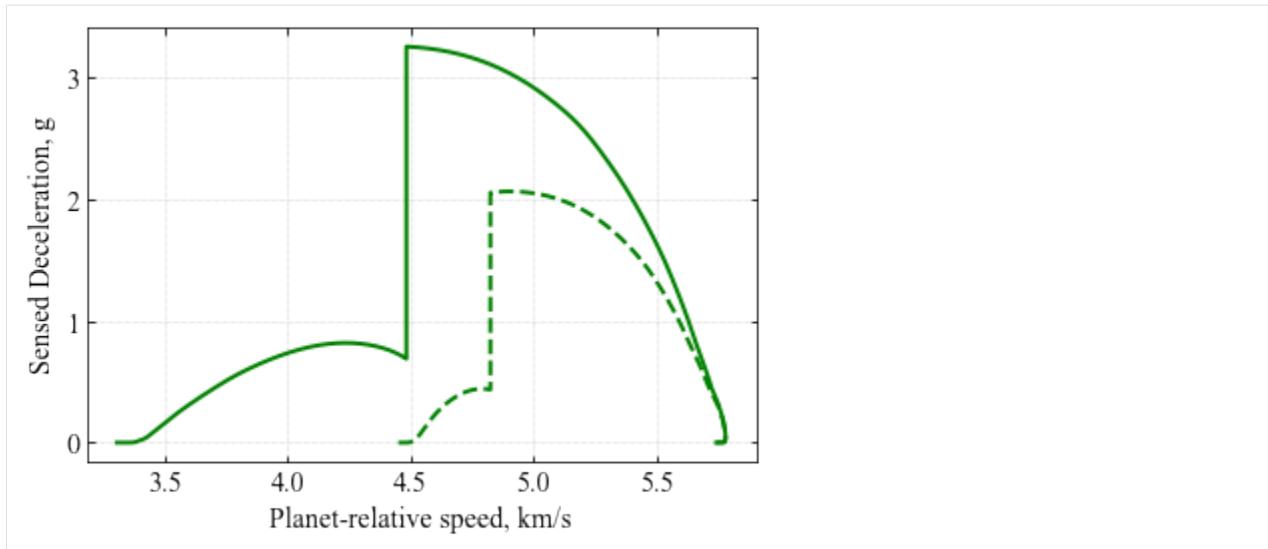
[22]: plt.figure(figsize=(6,4))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.plot(vehicle1.v_kms_full, vehicle1.acc_net_g_full, 'g-', linewidth=2.0)
plt.plot(vehicle2.v_kms_full, vehicle2.acc_net_g_full, 'g--', linewidth=2.0)

plt.xlabel('Planet-relative speed, km/s', fontsize=14)
plt.ylabel('Sensed Deceleration, g', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)
plt.grid(linestyle='dotted', linewidth=0.5)

plt.savefig('../plots/werner-smallsat-nominal-speed-decel-mars.png',bbox_inches='tight
    ↪')
plt.savefig('../plots/werner-smallsat-nominal-speed-decel-mars.pdf', dpi=300, bbox_
    ↪inches='tight')
plt.savefig('../plots/werner-smallsat-nominal-speed-decel-mars.eps', dpi=300, bbox_
    ↪inches='tight')

plt.show()

```

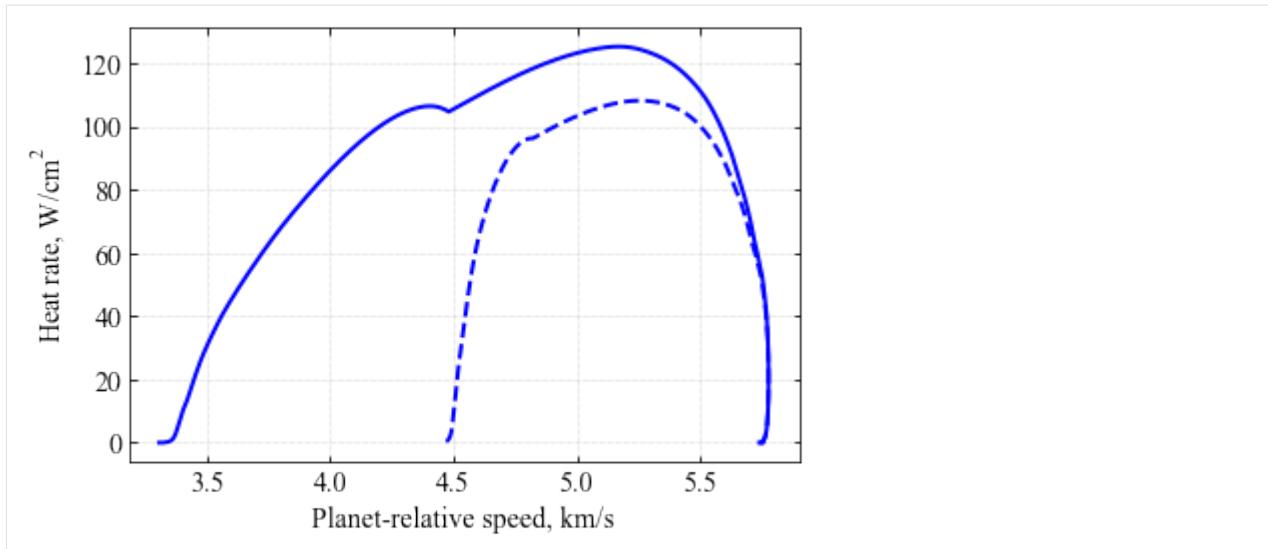


```
[23]: plt.figure(figsize=(6, 4))
plt.rc('font', family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.plot(vehicle1.v_kms_full, vehicle1.q_stag_total_full, 'b-', linewidth=2.0)
plt.plot(vehicle2.v_kms_full, vehicle2.q_stag_total_full, 'b--', linewidth=2.0)

plt.xlabel('Planet-relative speed, km/s', fontsize=14)
plt.ylabel('Heat rate, '+r'$W/cm^2$', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)
plt.grid(linestyle='dotted', linewidth=0.5)

plt.savefig('../plots/werner-smallsat-nominal-speed-heat-mars.png',bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-nominal-speed-heat-mars.pdf', dpi=300,bbox_
    ↪inches='tight')
plt.savefig('../plots/werner-smallsat-nominal-speed-heat-mars.eps', dpi=300,bbox_
    ↪inches='tight')

plt.show()
```



```
[24]: vehicle1.terminal_apoapsis
```

```
[24]: 612.6745425743503
```

```
[25]: vehicle2.terminal_apoapsis
```

```
[25]: 32890.38040256599
```

4.52 Example - 52 - Mars SmallSat Aerocapture Demonstration - Part 2

In this example, we will use Monte Carlo simulations to assess flight performance. Reference: M.S.Werner and R.D.Braun, Mission Design and Performance Analysis of a Smallsat Aerocapture Flight Test, Journal of Spacecraft and Rockets, DOI: 10.2514/1.A33997

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
from scipy import interpolate
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.patches import Polygon
```

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("MARS")
planet.loadAtmosphereModel('../atmdata/Mars/mars-gram-avg.dat', 0, 1, 2, 3)
planet.h_skip = 150000.0
planet.h_low=10.0E3

# Set up the drag modulation vehicle.
vehicle=Vehicle('MarsSmallSat1', 25.97, 66.4, 0.0, np.pi*0.25**2, 0.0, 0.0563, planet)
```

(continues on next page)

(continued from previous page)

```

vehicle.setInitialState(150.0,0.0,0.0,5.74,0.0,-12.12,0.0,0.0)
vehicle.setSolverParams(1E-6)
vehicle.setDragModulationVehicleParams(66.4,4.72)

# Set up the drag modulation entry phase guidance parameters.
vehicle.setDragEntryPhaseParams(2.0, 15.0, 101, -200.0)

# Set the target orbit parameters.
vehicle.setTargetOrbitParams(450.0, 450.0, 20.0)

# Define the path to atmospheric files to be used for the Monte Carlo simulations.
atmfiles = ['../atmdata/Mars/LAT00N-N1000.txt']

# Set up the Monte Carlo simulation for drag modulation.
# NPOS = 156, NMONTE = 1000
# Target EFPA = -12.05 deg
# EFPA 1-sigma error = +/- 0.067 deg
# Nominal beta_1 = 66.4 kg/m2
# beta_1 1-sigma = 0.0
# guidance time step for entry = 1.0s (Freq. = 1 Hz)
# guidance time step after jettison = 1.0 s
# max. solver time step = 0.1 s
# max. time used by solver = 2400 s

vehicle.setupMonteCarloSimulationD(156, 1000, atmfiles, 0 , 1, 2, 3, 4, True,
                                  -12.05, 0.0667, 66.4, 0.0,
                                  1.0, 1.0, 0.1, 2400.0)

# Run 200 trajectories
vehicle.runMonteCarloD(200, '../data/werner2019/MCB1-Mars')

BATCH :../data/werner2019/MCB1-Mars, RUN #: 1, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
    ↵ SAMPLE #: 473, EFPA: -12.02, SIGMA: 0.05, APO : 470.71
BATCH :../data/werner2019/MCB1-Mars, RUN #: 2, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
    ↵ SAMPLE #: 948, EFPA: -12.11, SIGMA: -0.68, APO : 249.09
BATCH :../data/werner2019/MCB1-Mars, RUN #: 3, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
    ↵ SAMPLE #: 982, EFPA: -12.11, SIGMA: -0.00, APO : 183.77
BATCH :../data/werner2019/MCB1-Mars, RUN #: 4, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
    ↵ SAMPLE #: 909, EFPA: -12.14, SIGMA: 0.97, APO : 165.56
BATCH :../data/werner2019/MCB1-Mars, RUN #: 5, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
    ↵ SAMPLE #: 795, EFPA: -11.96, SIGMA: 0.19, APO : 467.54
BATCH :../data/werner2019/MCB1-Mars, RUN #: 6, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
    ↵ SAMPLE #: 493, EFPA: -12.00, SIGMA: 0.10, APO : 442.98
BATCH :../data/werner2019/MCB1-Mars, RUN #: 7, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
    ↵ SAMPLE #: 255, EFPA: -12.00, SIGMA: 0.47, APO : 474.19
BATCH :../data/werner2019/MCB1-Mars, RUN #: 8, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
    ↵ SAMPLE #: 772, EFPA: -12.01, SIGMA: -0.41, APO : 470.02
BATCH :../data/werner2019/MCB1-Mars, RUN #: 9, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
    ↵ SAMPLE #: 819, EFPA: -12.08, SIGMA: -0.51, APO : 383.96
BATCH :../data/werner2019/MCB1-Mars, RUN #: 10, PROF: ../atmdata/Mars/LAT00N-N1000.
    ↵txt, SAMPLE #: 707, EFPA: -12.11, SIGMA: -0.56, APO : 396.13
BATCH :../data/werner2019/MCB1-Mars, RUN #: 11, PROF: ../atmdata/Mars/LAT00N-N1000.
    ↵txt, SAMPLE #: 348, EFPA: -12.15, SIGMA: -0.59, APO : 30.81
BATCH :../data/werner2019/MCB1-Mars, RUN #: 12, PROF: ../atmdata/Mars/LAT00N-N1000.
    ↵txt, SAMPLE #: 183, EFPA: -12.07, SIGMA: 1.09, APO : 314.32
BATCH :../data/werner2019/MCB1-Mars, RUN #: 13, PROF: ../atmdata/Mars/LAT00N-N1000.
    ↵txt, SAMPLE #: 236, EFPA: -12.05, SIGMA: 1.01, APO : 307.66

```

(continues on next page)

(continued from previous page)

```
BATCH :./data/werner2019/MCB1-Mars, RUN #: 14, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 517, EFPA: -12.05, SIGMA: 1.79, APO : 262.04
BATCH :./data/werner2019/MCB1-Mars, RUN #: 15, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 643, EFPA: -12.15, SIGMA: -0.94, APO : 626.13
BATCH :./data/werner2019/MCB1-Mars, RUN #: 16, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 95, EFPA: -11.93, SIGMA: -0.67, APO : 477.67
BATCH :./data/werner2019/MCB1-Mars, RUN #: 17, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 718, EFPA: -12.02, SIGMA: 0.07, APO : 410.59
BATCH :./data/werner2019/MCB1-Mars, RUN #: 18, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 18, EFPA: -12.07, SIGMA: -1.54, APO : 444.56
/home/athul/anaconda3/lib/python3.7/site-packages/AMAT-2.1.2-py3.7.egg/AMAT/vehicle.
↪py:504: RuntimeWarning: invalid value encountered in sqrt
    ans[:] = 1.8980E-8 * (rho_vec[:]/self.RN)**0.5 * v[:]**3.0
BATCH :./data/werner2019/MCB1-Mars, RUN #: 19, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 42, EFPA: -12.08, SIGMA: -0.68, APO : 462.32
BATCH :./data/werner2019/MCB1-Mars, RUN #: 20, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 291, EFPA: -11.98, SIGMA: 0.15, APO : 428.46
BATCH :./data/werner2019/MCB1-Mars, RUN #: 21, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 54, EFPA: -12.16, SIGMA: 1.26, APO : 30.55
BATCH :./data/werner2019/MCB1-Mars, RUN #: 22, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 471, EFPA: -12.06, SIGMA: -0.18, APO : 429.90
BATCH :./data/werner2019/MCB1-Mars, RUN #: 23, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 404, EFPA: -11.98, SIGMA: -0.57, APO : 445.32
BATCH :./data/werner2019/MCB1-Mars, RUN #: 24, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 206, EFPA: -11.97, SIGMA: -0.55, APO : 473.24
BATCH :./data/werner2019/MCB1-Mars, RUN #: 25, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 913, EFPA: -12.10, SIGMA: 0.73, APO : 346.17
BATCH :./data/werner2019/MCB1-Mars, RUN #: 26, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 24, EFPA: -12.07, SIGMA: 0.01, APO : 537.43
BATCH :./data/werner2019/MCB1-Mars, RUN #: 27, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 398, EFPA: -12.02, SIGMA: 0.96, APO : 434.48
BATCH :./data/werner2019/MCB1-Mars, RUN #: 28, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 318, EFPA: -12.05, SIGMA: -0.05, APO : 445.43
BATCH :./data/werner2019/MCB1-Mars, RUN #: 29, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 200, EFPA: -12.04, SIGMA: 0.59, APO : 418.05
BATCH :./data/werner2019/MCB1-Mars, RUN #: 30, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 237, EFPA: -12.08, SIGMA: 0.45, APO : 397.03
BATCH :./data/werner2019/MCB1-Mars, RUN #: 31, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 708, EFPA: -12.00, SIGMA: -1.92, APO : 484.91
BATCH :./data/werner2019/MCB1-Mars, RUN #: 32, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 727, EFPA: -12.03, SIGMA: -1.57, APO : 470.89
BATCH :./data/werner2019/MCB1-Mars, RUN #: 33, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 526, EFPA: -11.99, SIGMA: -0.34, APO : 470.23
BATCH :./data/werner2019/MCB1-Mars, RUN #: 34, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 479, EFPA: -12.07, SIGMA: 0.45, APO : 477.59
BATCH :./data/werner2019/MCB1-Mars, RUN #: 35, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 454, EFPA: -12.13, SIGMA: 1.18, APO : 219.58
BATCH :./data/werner2019/MCB1-Mars, RUN #: 36, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 573, EFPA: -11.91, SIGMA: -0.10, APO : 528.42
BATCH :./data/werner2019/MCB1-Mars, RUN #: 37, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 207, EFPA: -12.01, SIGMA: 0.82, APO : 436.65
BATCH :./data/werner2019/MCB1-Mars, RUN #: 38, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 597, EFPA: -12.08, SIGMA: 0.11, APO : 375.03
BATCH :./data/werner2019/MCB1-Mars, RUN #: 39, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 379, EFPA: -12.10, SIGMA: 1.37, APO : 231.86
BATCH :./data/werner2019/MCB1-Mars, RUN #: 40, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 848, EFPA: -12.12, SIGMA: -2.18, APO : 370.00
```

(continues on next page)

(continued from previous page)

```

BATCH :./data/werner2019/MCB1-Mars, RUN #: 41, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 257, EFPA: -12.01, SIGMA: -0.02, APO : 439.20
BATCH :./data/werner2019/MCB1-Mars, RUN #: 42, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 738, EFPA: -12.01, SIGMA: -1.37, APO : 435.96
BATCH :./data/werner2019/MCB1-Mars, RUN #: 43, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 847, EFPA: -12.06, SIGMA: 0.42, APO : 287.91
BATCH :./data/werner2019/MCB1-Mars, RUN #: 44, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 892, EFPA: -12.12, SIGMA: -0.27, APO : 485.94
BATCH :./data/werner2019/MCB1-Mars, RUN #: 45, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 229, EFPA: -12.05, SIGMA: -1.43, APO : 457.72
BATCH :./data/werner2019/MCB1-Mars, RUN #: 46, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 471, EFPA: -11.96, SIGMA: -0.19, APO : 469.02
BATCH :./data/werner2019/MCB1-Mars, RUN #: 47, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 107, EFPA: -12.00, SIGMA: 0.39, APO : 423.73
BATCH :./data/werner2019/MCB1-Mars, RUN #: 48, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 128, EFPA: -12.03, SIGMA: 1.40, APO : 317.12
BATCH :./data/werner2019/MCB1-Mars, RUN #: 49, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 427, EFPA: -12.10, SIGMA: -0.70, APO : 29.04
BATCH :./data/werner2019/MCB1-Mars, RUN #: 50, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 497, EFPA: -12.07, SIGMA: -0.69, APO : 446.25
BATCH :./data/werner2019/MCB1-Mars, RUN #: 51, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 206, EFPA: -11.99, SIGMA: 0.40, APO : 431.06
BATCH :./data/werner2019/MCB1-Mars, RUN #: 52, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 579, EFPA: -12.04, SIGMA: -0.30, APO : 404.20
BATCH :./data/werner2019/MCB1-Mars, RUN #: 53, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 738, EFPA: -11.97, SIGMA: 0.37, APO : 455.28
BATCH :./data/werner2019/MCB1-Mars, RUN #: 54, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 781, EFPA: -12.11, SIGMA: -1.31, APO : 307.05
BATCH :./data/werner2019/MCB1-Mars, RUN #: 55, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 608, EFPA: -12.09, SIGMA: -0.49, APO : 280.04
BATCH :./data/werner2019/MCB1-Mars, RUN #: 56, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 586, EFPA: -11.89, SIGMA: -0.31, APO : 3114.30
BATCH :./data/werner2019/MCB1-Mars, RUN #: 57, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 170, EFPA: -12.08, SIGMA: 1.99, APO : 359.22
BATCH :./data/werner2019/MCB1-Mars, RUN #: 58, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 555, EFPA: -12.21, SIGMA: 0.20, APO : 30.35
BATCH :./data/werner2019/MCB1-Mars, RUN #: 59, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 870, EFPA: -11.96, SIGMA: -1.26, APO : 2180.48
BATCH :./data/werner2019/MCB1-Mars, RUN #: 60, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 218, EFPA: -12.07, SIGMA: 0.67, APO : 392.25
BATCH :./data/werner2019/MCB1-Mars, RUN #: 61, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 136, EFPA: -12.05, SIGMA: 0.93, APO : 243.16
BATCH :./data/werner2019/MCB1-Mars, RUN #: 62, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 819, EFPA: -12.03, SIGMA: 0.07, APO : 475.63
BATCH :./data/werner2019/MCB1-Mars, RUN #: 63, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 398, EFPA: -11.97, SIGMA: 0.79, APO : 422.55
BATCH :./data/werner2019/MCB1-Mars, RUN #: 64, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 899, EFPA: -12.07, SIGMA: 1.07, APO : 384.39
BATCH :./data/werner2019/MCB1-Mars, RUN #: 65, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 665, EFPA: -12.12, SIGMA: -0.33, APO : 628.14
BATCH :./data/werner2019/MCB1-Mars, RUN #: 66, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 727, EFPA: -12.08, SIGMA: -1.20, APO : 562.68
BATCH :./data/werner2019/MCB1-Mars, RUN #: 67, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 767, EFPA: -11.99, SIGMA: -0.34, APO : 466.43
BATCH :./data/werner2019/MCB1-Mars, RUN #: 68, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 21, EFPA: -11.97, SIGMA: -1.06, APO : 486.86
BATCH :./data/werner2019/MCB1-Mars, RUN #: 69, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 104, EFPA: -12.02, SIGMA: 0.82, APO : 439.20

```

(continues on next page)

(continued from previous page)

```
BATCH :./data/werner2019/MCB1-Mars, RUN #: 70, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 850, EFPA: -12.08, SIGMA: 0.15, APO : 333.87
BATCH :./data/werner2019/MCB1-Mars, RUN #: 71, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 63, EFPA: -12.14, SIGMA: 0.36, APO : 329.06
BATCH :./data/werner2019/MCB1-Mars, RUN #: 72, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 898, EFPA: -12.12, SIGMA: -1.67, APO : 285.72
BATCH :./data/werner2019/MCB1-Mars, RUN #: 73, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 861, EFPA: -12.00, SIGMA: -0.27, APO : 478.64
BATCH :./data/werner2019/MCB1-Mars, RUN #: 74, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 883, EFPA: -11.98, SIGMA: 0.52, APO : 470.65
BATCH :./data/werner2019/MCB1-Mars, RUN #: 75, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 204, EFPA: -12.07, SIGMA: -0.67, APO : 402.21
BATCH :./data/werner2019/MCB1-Mars, RUN #: 76, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 431, EFPA: -12.09, SIGMA: -1.27, APO : 443.38
BATCH :./data/werner2019/MCB1-Mars, RUN #: 77, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 51, EFPA: -11.99, SIGMA: 1.09, APO : 467.80
BATCH :./data/werner2019/MCB1-Mars, RUN #: 78, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 835, EFPA: -12.16, SIGMA: -0.06, APO : 180.61
BATCH :./data/werner2019/MCB1-Mars, RUN #: 79, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 424, EFPA: -12.20, SIGMA: -1.29, APO : 652.95
BATCH :./data/werner2019/MCB1-Mars, RUN #: 80, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 558, EFPA: -11.91, SIGMA: 0.10, APO : 616.51
BATCH :./data/werner2019/MCB1-Mars, RUN #: 81, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 184, EFPA: -11.96, SIGMA: 0.77, APO : 475.03
BATCH :./data/werner2019/MCB1-Mars, RUN #: 82, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 993, EFPA: -11.97, SIGMA: 0.64, APO : 453.54
BATCH :./data/werner2019/MCB1-Mars, RUN #: 83, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 501, EFPA: -11.99, SIGMA: -0.83, APO : 468.49
BATCH :./data/werner2019/MCB1-Mars, RUN #: 84, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 493, EFPA: -11.99, SIGMA: -0.11, APO : 508.56
BATCH :./data/werner2019/MCB1-Mars, RUN #: 85, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 638, EFPA: -12.14, SIGMA: -0.23, APO : 29.83
BATCH :./data/werner2019/MCB1-Mars, RUN #: 86, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 340, EFPA: -12.22, SIGMA: 1.23, APO : 29.68
BATCH :./data/werner2019/MCB1-Mars, RUN #: 87, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 218, EFPA: -12.07, SIGMA: 0.48, APO : 385.04
BATCH :./data/werner2019/MCB1-Mars, RUN #: 88, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 318, EFPA: -11.93, SIGMA: 1.64, APO : 483.52
BATCH :./data/werner2019/MCB1-Mars, RUN #: 89, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 817, EFPA: -12.13, SIGMA: 0.94, APO : 384.50
BATCH :./data/werner2019/MCB1-Mars, RUN #: 90, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 894, EFPA: -12.07, SIGMA: 0.56, APO : 264.38
BATCH :./data/werner2019/MCB1-Mars, RUN #: 91, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 541, EFPA: -12.09, SIGMA: -0.64, APO : 389.50
BATCH :./data/werner2019/MCB1-Mars, RUN #: 92, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 160, EFPA: -12.12, SIGMA: -1.07, APO : 328.40
BATCH :./data/werner2019/MCB1-Mars, RUN #: 93, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 723, EFPA: -12.07, SIGMA: 1.14, APO : 262.13
BATCH :./data/werner2019/MCB1-Mars, RUN #: 94, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 473, EFPA: -12.03, SIGMA: -1.37, APO : 467.98
BATCH :./data/werner2019/MCB1-Mars, RUN #: 95, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 803, EFPA: -12.08, SIGMA: -0.36, APO : 488.40
BATCH :./data/werner2019/MCB1-Mars, RUN #: 96, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 571, EFPA: -12.01, SIGMA: 0.73, APO : 336.25
BATCH :./data/werner2019/MCB1-Mars, RUN #: 97, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 941, EFPA: -12.13, SIGMA: 0.54, APO : 393.71
BATCH :./data/werner2019/MCB1-Mars, RUN #: 98, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 337, EFPA: -12.08, SIGMA: -0.36, APO : 300.38
```

(continues on next page)

(continued from previous page)

```

BATCH :./data/werner2019/MCB1-Mars, RUN #: 99, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 870, EFPA: -12.04, SIGMA: -0.65, APO : 485.49
BATCH :./data/werner2019/MCB1-Mars, RUN #: 100, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 740, EFPA: -12.15, SIGMA: -1.16, APO : 672.33
BATCH :./data/werner2019/MCB1-Mars, RUN #: 101, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 261, EFPA: -12.10, SIGMA: 1.70, APO : 28.73
BATCH :./data/werner2019/MCB1-Mars, RUN #: 102, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 139, EFPA: -12.03, SIGMA: -0.76, APO : 477.85
BATCH :./data/werner2019/MCB1-Mars, RUN #: 103, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 828, EFPA: -12.14, SIGMA: 1.64, APO : 29.30
BATCH :./data/werner2019/MCB1-Mars, RUN #: 104, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 523, EFPA: -11.99, SIGMA: 1.19, APO : 299.59
BATCH :./data/werner2019/MCB1-Mars, RUN #: 105, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 86, EFPA: -12.08, SIGMA: -0.91, APO : 391.42
BATCH :./data/werner2019/MCB1-Mars, RUN #: 106, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 650, EFPA: -12.08, SIGMA: 1.97, APO : 302.54
BATCH :./data/werner2019/MCB1-Mars, RUN #: 107, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 819, EFPA: -12.04, SIGMA: 0.84, APO : 351.63
BATCH :./data/werner2019/MCB1-Mars, RUN #: 108, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 567, EFPA: -12.02, SIGMA: 1.34, APO : 565.22
BATCH :./data/werner2019/MCB1-Mars, RUN #: 109, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 826, EFPA: -12.02, SIGMA: 0.21, APO : 449.97
BATCH :./data/werner2019/MCB1-Mars, RUN #: 110, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 814, EFPA: -11.95, SIGMA: -2.35, APO : 470.46
BATCH :./data/werner2019/MCB1-Mars, RUN #: 111, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 383, EFPA: -12.07, SIGMA: 0.18, APO : 438.04
BATCH :./data/werner2019/MCB1-Mars, RUN #: 112, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 966, EFPA: -12.04, SIGMA: 0.45, APO : 346.67
BATCH :./data/werner2019/MCB1-Mars, RUN #: 113, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 121, EFPA: -12.02, SIGMA: 1.17, APO : 311.76
BATCH :./data/werner2019/MCB1-Mars, RUN #: 114, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 470, EFPA: -11.98, SIGMA: -0.82, APO : 491.70
BATCH :./data/werner2019/MCB1-Mars, RUN #: 115, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 43, EFPA: -12.00, SIGMA: -0.37, APO : 445.74
BATCH :./data/werner2019/MCB1-Mars, RUN #: 116, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 363, EFPA: -12.13, SIGMA: -1.66, APO : 401.13
BATCH :./data/werner2019/MCB1-Mars, RUN #: 117, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 592, EFPA: -12.06, SIGMA: -1.24, APO : 404.40
BATCH :./data/werner2019/MCB1-Mars, RUN #: 118, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 27, EFPA: -12.09, SIGMA: -1.55, APO : 332.11
BATCH :./data/werner2019/MCB1-Mars, RUN #: 119, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 825, EFPA: -12.12, SIGMA: 0.23, APO : 308.05
BATCH :./data/werner2019/MCB1-Mars, RUN #: 120, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 483, EFPA: -11.98, SIGMA: 1.25, APO : 429.94
BATCH :./data/werner2019/MCB1-Mars, RUN #: 121, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 8, EFPA: -12.05, SIGMA: 0.07, APO : 345.33
BATCH :./data/werner2019/MCB1-Mars, RUN #: 122, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 343, EFPA: -11.95, SIGMA: 0.18, APO : 484.28
BATCH :./data/werner2019/MCB1-Mars, RUN #: 123, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 311, EFPA: -12.15, SIGMA: 1.17, APO : 30.46
BATCH :./data/werner2019/MCB1-Mars, RUN #: 124, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 426, EFPA: -12.06, SIGMA: -1.71, APO : 422.83
BATCH :./data/werner2019/MCB1-Mars, RUN #: 125, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 369, EFPA: -12.08, SIGMA: 1.58, APO : 270.59
BATCH :./data/werner2019/MCB1-Mars, RUN #: 126, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 291, EFPA: -11.98, SIGMA: -0.21, APO : 440.37
BATCH :./data/werner2019/MCB1-Mars, RUN #: 127, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 328, EFPA: -12.13, SIGMA: -0.21, APO : 252.96

```

(continues on next page)

(continued from previous page)

```
BATCH :./data/werner2019/MCB1-Mars, RUN #: 128, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 107, EFPA: -12.08, SIGMA: 1.50, APO : 481.17
BATCH :./data/werner2019/MCB1-Mars, RUN #: 129, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 398, EFPA: -12.00, SIGMA: 1.43, APO : 381.01
BATCH :./data/werner2019/MCB1-Mars, RUN #: 130, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 472, EFPA: -12.01, SIGMA: -0.72, APO : 469.38
BATCH :./data/werner2019/MCB1-Mars, RUN #: 131, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 860, EFPA: -12.11, SIGMA: 0.83, APO : 252.62
BATCH :./data/werner2019/MCB1-Mars, RUN #: 132, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 64, EFPA: -12.02, SIGMA: 1.19, APO : 448.25
BATCH :./data/werner2019/MCB1-Mars, RUN #: 133, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 113, EFPA: -12.03, SIGMA: 1.44, APO : 379.96
BATCH :./data/werner2019/MCB1-Mars, RUN #: 134, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 996, EFPA: -11.97, SIGMA: -0.72, APO : 458.85
BATCH :./data/werner2019/MCB1-Mars, RUN #: 135, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 496, EFPA: -12.09, SIGMA: -1.28, APO : 404.90
BATCH :./data/werner2019/MCB1-Mars, RUN #: 136, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 975, EFPA: -12.14, SIGMA: -0.21, APO : 845.32
BATCH :./data/werner2019/MCB1-Mars, RUN #: 137, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 196, EFPA: -12.04, SIGMA: 0.75, APO : 290.71
BATCH :./data/werner2019/MCB1-Mars, RUN #: 138, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 398, EFPA: -12.15, SIGMA: -0.36, APO : 356.07
BATCH :./data/werner2019/MCB1-Mars, RUN #: 139, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 798, EFPA: -12.11, SIGMA: -0.45, APO : 407.26
BATCH :./data/werner2019/MCB1-Mars, RUN #: 140, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 850, EFPA: -11.99, SIGMA: -0.63, APO : 429.75
BATCH :./data/werner2019/MCB1-Mars, RUN #: 141, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 364, EFPA: -12.01, SIGMA: -2.51, APO : 470.25
BATCH :./data/werner2019/MCB1-Mars, RUN #: 142, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 497, EFPA: -12.12, SIGMA: -0.11, APO : 197.46
BATCH :./data/werner2019/MCB1-Mars, RUN #: 143, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 942, EFPA: -12.06, SIGMA: -0.13, APO : 392.91
BATCH :./data/werner2019/MCB1-Mars, RUN #: 144, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 292, EFPA: -11.99, SIGMA: -0.16, APO : 462.44
BATCH :./data/werner2019/MCB1-Mars, RUN #: 145, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 457, EFPA: -11.99, SIGMA: -1.14, APO : 474.83
BATCH :./data/werner2019/MCB1-Mars, RUN #: 146, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 202, EFPA: -12.07, SIGMA: 0.95, APO : 352.26
BATCH :./data/werner2019/MCB1-Mars, RUN #: 147, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 411, EFPA: -12.05, SIGMA: -1.34, APO : 439.96
BATCH :./data/werner2019/MCB1-Mars, RUN #: 148, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 926, EFPA: -12.08, SIGMA: 0.69, APO : 339.36
BATCH :./data/werner2019/MCB1-Mars, RUN #: 149, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 341, EFPA: -12.15, SIGMA: 1.01, APO : 177.54
BATCH :./data/werner2019/MCB1-Mars, RUN #: 150, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 897, EFPA: -12.03, SIGMA: 1.11, APO : 248.00
BATCH :./data/werner2019/MCB1-Mars, RUN #: 151, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 759, EFPA: -12.04, SIGMA: -0.16, APO : 424.57
BATCH :./data/werner2019/MCB1-Mars, RUN #: 152, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 736, EFPA: -12.01, SIGMA: -1.08, APO : 476.70
BATCH :./data/werner2019/MCB1-Mars, RUN #: 153, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 367, EFPA: -12.05, SIGMA: -2.00, APO : 439.34
BATCH :./data/werner2019/MCB1-Mars, RUN #: 154, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 715, EFPA: -12.10, SIGMA: -1.63, APO : 444.77
BATCH :./data/werner2019/MCB1-Mars, RUN #: 155, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 196, EFPA: -12.03, SIGMA: 1.02, APO : 334.06
BATCH :./data/werner2019/MCB1-Mars, RUN #: 156, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 822, EFPA: -12.12, SIGMA: -0.38, APO : 251.38
```

(continues on next page)

(continued from previous page)

```

BATCH :./data/werner2019/MCB1-Mars, RUN #: 157, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 77, EFPA: -12.05, SIGMA: 0.36, APO : 343.02
BATCH :./data/werner2019/MCB1-Mars, RUN #: 158, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 582, EFPA: -11.99, SIGMA: 0.97, APO : 371.69
BATCH :./data/werner2019/MCB1-Mars, RUN #: 159, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 483, EFPA: -12.10, SIGMA: 1.34, APO : 294.31
BATCH :./data/werner2019/MCB1-Mars, RUN #: 160, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 230, EFPA: -12.08, SIGMA: -0.16, APO : 398.86
BATCH :./data/werner2019/MCB1-Mars, RUN #: 161, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 171, EFPA: -12.08, SIGMA: -0.31, APO : 434.57
BATCH :./data/werner2019/MCB1-Mars, RUN #: 162, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 431, EFPA: -12.06, SIGMA: -1.51, APO : 437.77
BATCH :./data/werner2019/MCB1-Mars, RUN #: 163, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 744, EFPA: -12.02, SIGMA: 0.56, APO : 425.09
BATCH :./data/werner2019/MCB1-Mars, RUN #: 164, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 808, EFPA: -12.10, SIGMA: -0.18, APO : 318.38
BATCH :./data/werner2019/MCB1-Mars, RUN #: 165, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 678, EFPA: -11.99, SIGMA: 0.96, APO : 425.24
BATCH :./data/werner2019/MCB1-Mars, RUN #: 166, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 815, EFPA: -12.06, SIGMA: -0.91, APO : 407.80
BATCH :./data/werner2019/MCB1-Mars, RUN #: 167, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 929, EFPA: -12.06, SIGMA: -2.28, APO : 436.81
BATCH :./data/werner2019/MCB1-Mars, RUN #: 168, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 902, EFPA: -12.07, SIGMA: -0.27, APO : 389.17
BATCH :./data/werner2019/MCB1-Mars, RUN #: 169, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 739, EFPA: -12.04, SIGMA: -0.56, APO : 475.41
BATCH :./data/werner2019/MCB1-Mars, RUN #: 170, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 116, EFPA: -12.10, SIGMA: -1.21, APO : 345.25
BATCH :./data/werner2019/MCB1-Mars, RUN #: 171, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 521, EFPA: -12.03, SIGMA: 0.19, APO : 436.42
BATCH :./data/werner2019/MCB1-Mars, RUN #: 172, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 706, EFPA: -12.18, SIGMA: -0.71, APO : 166.46
BATCH :./data/werner2019/MCB1-Mars, RUN #: 173, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 918, EFPA: -12.04, SIGMA: -0.41, APO : 416.91
BATCH :./data/werner2019/MCB1-Mars, RUN #: 174, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 730, EFPA: -12.09, SIGMA: 0.29, APO : 265.49
BATCH :./data/werner2019/MCB1-Mars, RUN #: 175, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 773, EFPA: -12.00, SIGMA: -1.76, APO : 463.55
BATCH :./data/werner2019/MCB1-Mars, RUN #: 176, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 976, EFPA: -12.09, SIGMA: -1.97, APO : 418.32
BATCH :./data/werner2019/MCB1-Mars, RUN #: 177, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 522, EFPA: -11.99, SIGMA: 0.59, APO : 396.80
BATCH :./data/werner2019/MCB1-Mars, RUN #: 178, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 284, EFPA: -12.15, SIGMA: 1.21, APO : 30.75
BATCH :./data/werner2019/MCB1-Mars, RUN #: 179, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 773, EFPA: -11.97, SIGMA: 0.05, APO : 448.87
BATCH :./data/werner2019/MCB1-Mars, RUN #: 180, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 928, EFPA: -11.99, SIGMA: -1.09, APO : 445.25
BATCH :./data/werner2019/MCB1-Mars, RUN #: 181, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 519, EFPA: -12.07, SIGMA: 0.02, APO : 417.03
BATCH :./data/werner2019/MCB1-Mars, RUN #: 182, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 330, EFPA: -12.13, SIGMA: 0.62, APO : 389.94
BATCH :./data/werner2019/MCB1-Mars, RUN #: 183, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 241, EFPA: -12.10, SIGMA: -1.98, APO : 405.54
BATCH :./data/werner2019/MCB1-Mars, RUN #: 184, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 855, EFPA: -11.98, SIGMA: -0.27, APO : 478.47
BATCH :./data/werner2019/MCB1-Mars, RUN #: 185, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 381, EFPA: -12.08, SIGMA: 0.99, APO : 244.64

```

(continues on next page)

(continued from previous page)

```
BATCH :./data/werner2019/MCB1-Mars, RUN #: 186, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 182, EFPA: -11.95, SIGMA: 2.14, APO : 474.76
BATCH :./data/werner2019/MCB1-Mars, RUN #: 187, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 52, EFPA: -12.08, SIGMA: -0.18, APO : 486.57
BATCH :./data/werner2019/MCB1-Mars, RUN #: 188, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 296, EFPA: -12.03, SIGMA: -0.24, APO : 481.16
BATCH :./data/werner2019/MCB1-Mars, RUN #: 189, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 737, EFPA: -12.13, SIGMA: 0.89, APO : 326.65
BATCH :./data/werner2019/MCB1-Mars, RUN #: 190, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 985, EFPA: -11.97, SIGMA: -0.81, APO : 489.30
BATCH :./data/werner2019/MCB1-Mars, RUN #: 191, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 707, EFPA: -12.19, SIGMA: -1.24, APO : 396.14
BATCH :./data/werner2019/MCB1-Mars, RUN #: 192, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 385, EFPA: -12.05, SIGMA: -0.86, APO : 425.49
BATCH :./data/werner2019/MCB1-Mars, RUN #: 193, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 594, EFPA: -12.02, SIGMA: -1.46, APO : 488.71
BATCH :./data/werner2019/MCB1-Mars, RUN #: 194, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 576, EFPA: -12.05, SIGMA: -1.01, APO : 457.67
BATCH :./data/werner2019/MCB1-Mars, RUN #: 195, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 189, EFPA: -12.06, SIGMA: 0.96, APO : 401.33
BATCH :./data/werner2019/MCB1-Mars, RUN #: 196, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 220, EFPA: -12.04, SIGMA: 1.90, APO : 239.64
BATCH :./data/werner2019/MCB1-Mars, RUN #: 197, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 714, EFPA: -12.09, SIGMA: -0.08, APO : 430.14
BATCH :./data/werner2019/MCB1-Mars, RUN #: 198, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 992, EFPA: -12.13, SIGMA: 0.62, APO : 618.85
BATCH :./data/werner2019/MCB1-Mars, RUN #: 199, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 446, EFPA: -12.12, SIGMA: 0.34, APO : 590.72
BATCH :./data/werner2019/MCB1-Mars, RUN #: 200, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 574, EFPA: -12.07, SIGMA: 0.05, APO : 389.68
```

Load the terminal apoapsis, deceleration and heat rate data.

```
[4]: apoap = np.loadtxt('../data/werner2019/MCB1-Mars/terminal_apoapsis_arr.txt')
peria = np.loadtxt('../data/werner2019/MCB1-Mars/terminal_periapsis_arr.txt')
decel = np.loadtxt('../data/werner2019/MCB1-Mars/acc_net_g_max_arr.txt')
heatr = np.loadtxt('../data/werner2019/MCB1-Mars/q_stag_max_arr.txt')
```

Remove cases which resulted in apoapsis < 100 km or > 800 km.

```
[5]: del_index_low = np.where(apoap < 100)
print(del_index_low)

(array([ 10,   20,   48,   57,   84,   85,  100,  102,  122,  177]),)
```

Seven cases resulted in apoapsis less than 100 km, and are considered to crash.

Remove these entries from the data before plotting.

```
[6]: apoap_clean = np.delete(apoap, del_index_low)
peria_clean = np.delete(peria, del_index_low)
decel_clean = np.delete(decel, del_index_low)
heatr_clean = np.delete(heatr, del_index_low)
```

Also remove cases with apoapsis > 800 km.

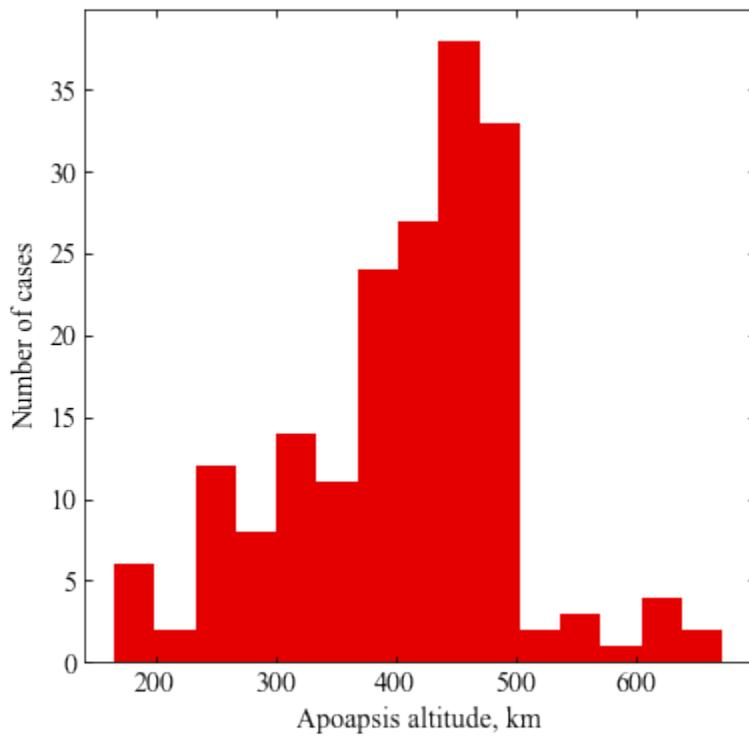
```
[7]: del_index_high = np.where(apoap_clean > 800)
print(del_index_high)
```

```
(array([ 52,  54, 126]),)
```

```
[8]: apoap_clean2 = np.delete(apoap_clean, del_index_high)
peria_clean2 = np.delete(peria_clean, del_index_high)
decel_clean2 = np.delete(decel_clean, del_index_high)
heatr_clean2 = np.delete(heatr_clean, del_index_high)
```

```
[15]: plt.figure(figsize=(6,6))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.hist(apoap_clean2, bins=15, color='xkcd:red')
plt.xlabel('Apoapsis altitude, km', fontsize=14)
plt.ylabel('Number of cases', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/werner-smallsat-apoa-hist-mars.png',bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-apoa-hist-mars.pdf', dpi=300,bbox_inches='tight
')
plt.savefig('../plots/werner-smallsat-apoa-hist-mars.eps', dpi=300,bbox_inches='tight
')
plt.show()
```



```
[10]: plt.figure(figsize=(6,6))
plt.rc('font',family='Times New Roman')
```

(continues on next page)

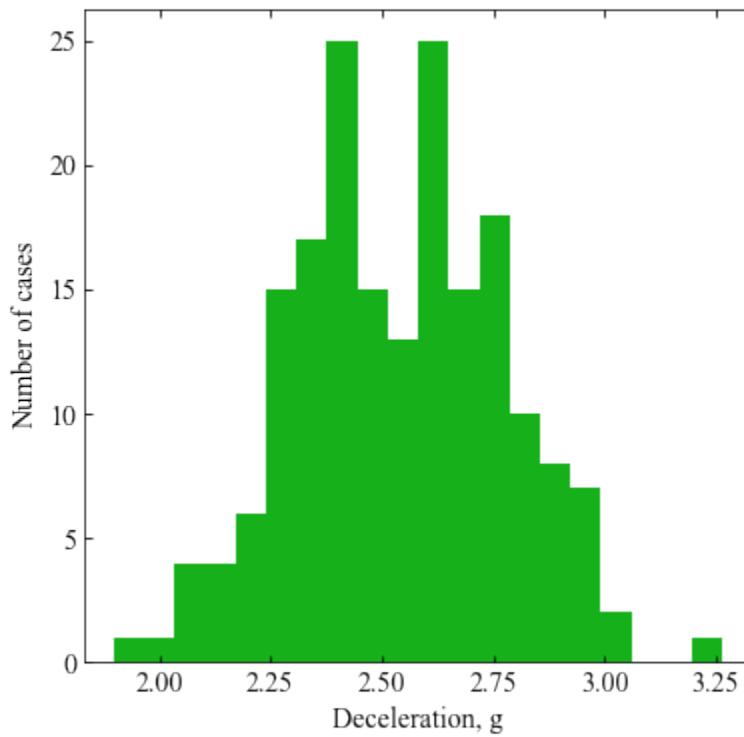
(continued from previous page)

```

params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.hist(decel_clean2, bins=20, color='xkcd:green')
plt.xlabel('Deceleration, g', fontsize=14)
plt.ylabel('Number of cases', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/werner-smallsat-decel-hist-mars.png',bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-decel-hist-mars.pdf', dpi=300,bbox_inches='tight'
           ↪)
plt.savefig('../plots/werner-smallsat-decel-hist-mars.eps', dpi=300,bbox_inches='tight
           ↪')
plt.show()

```



```

[11]: plt.figure(figsize=(6, 6))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.hist(heatr_clean2, bins=20, color='xkcd:blue')
plt.xlabel('Heat rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Number of cases', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')

```

(continues on next page)

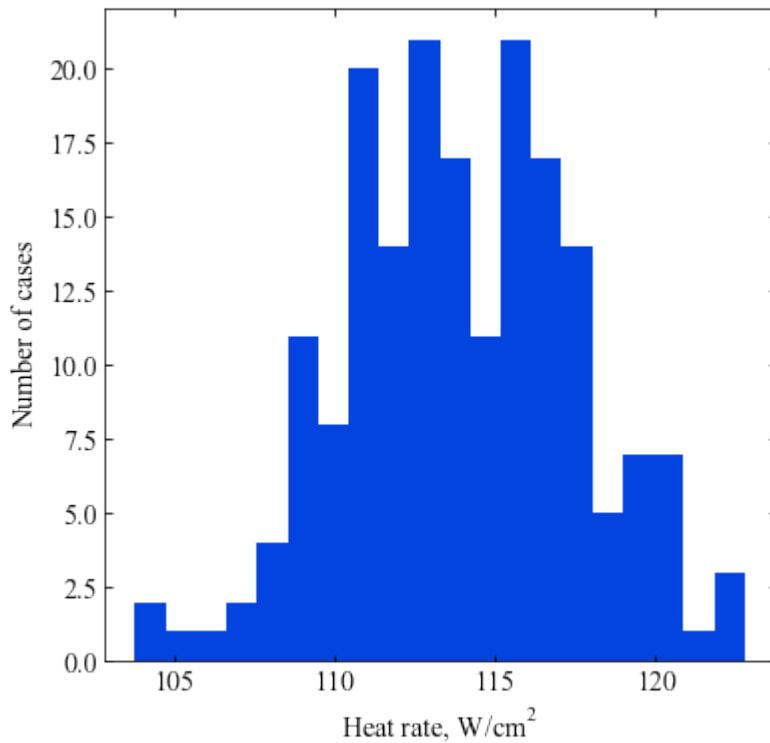
(continued from previous page)

```

ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/werner-smallsat-heat-hist-mars.png', bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-heat-hist-mars.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-heat-hist-mars.eps', dpi=300, bbox_inches='tight')
plt.show()

```



```

[14]: plt.figure(figsize=(6, 6))
plt.rc('font', family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.scatter(peria_clean2, apoap_clean2, s=30, facecolors='none', edgecolors='b', lw=2)
plt.axhline(y=450, lw=2.0, ls='dashed', color='k')
plt.xlabel('Periapsis altitude, km', fontsize=14)
plt.ylabel('Apoapsis altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

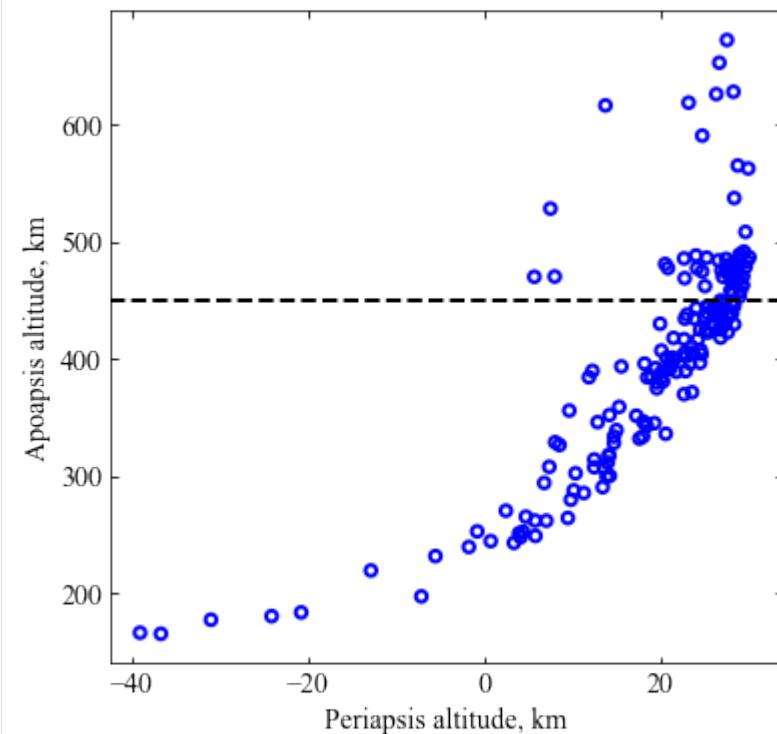
plt.savefig('../plots/werner-smallsat-apoa-peri-mars.png', bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-apoa-peri-mars.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-apoa-peri-mars.eps', dpi=300, bbox_inches='tight')

```

(continues on next page)

(continued from previous page)

```
plt.show()
```



Find the percentage of cases which achieved apoapsis between 100 and 800 km.

```
[13]: (np.size(apoap) - np.size(del_index_low) - np.size(del_index_high)) * 100 / np.size(apoap)
[13]: 93.5
```

These results are different from those in the paper because of the different guidance schemes used here. This exercise did not attempt to optimize the guidance parameters. Further study can improve the apoapsis targeting.

4.53 Example - 53 - Mars SmallSat Aerocapture Demonstration - Part 3

In this example, we will use Monte Carlo simulations to assess flight performance. Reference: M.S.Werner and R.D.Braun, Mission Design and Performance Analysis of a Smallsat Aerocapture Flight Test, Journal of Spacecraft and Rockets, DOI: 10.2514/1.A33997

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle

[2]: import numpy as np
from scipy import interpolate
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.patches import Polygon
```

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("MARS")
planet.loadAtmosphereModel('../atmdata/Mars/mars-gram-avg.dat', 0 , 1 ,2, 3)
planet.h_skip = 150000.0
planet.h_low=10.0E3

# Set up the drag modulation vehicle.
vehicle=Vehicle('MarsSmallSat1', 25.97, 66.4, 0.0, np.pi*0.25**2, 0.0, 0.0563, planet)

vehicle.setInitialState(150.0,0.0,0.0,5.74,0.0,-12.12,0.0,0.0)
vehicle.setSolverParams(1E-6)
vehicle.setDragModulationVehicleParams(66.4,4.72)

# Set up the drag modulation entry phase guidance parameters.
vehicle.setDragEntryPhaseParams(2.0, 15.0, 101, -200.0)

# Set the target orbit parameters.
vehicle.setTargetOrbitParams(450.0, 33000.0, 20.0)

# Define the path to atmospheric files to be used for the Monte Carlo simulations.
atmfiles = ['../atmdata/Mars/LAT00N-N1000.txt']

# Set up the Monte Carlo simulation for drag modulation.
# NPOS = 156, NMONTE = 1000
# Target EFPA = -12.05 deg
# EFPA 1-sigma error = +/- 0.067 deg
# Nominal beta_1 = 66.4 kg/m2
# beta_1 1-sigma = 0.0
# guidance time step for entry = 1.0s (Freq. = 1 Hz)
# guidance time step after jettison = 1.0 s
# max. solver time step = 0.1 s
# max. time used by solver = 2400 s

vehicle.setupMonteCarloSimulationD(156, 1000, atmfiles, 0 , 1, 2, 3, 4, True,
                                  -11.80, 0.0667, 66.4, 0.0,
                                  1.0, 1.0, 0.1, 2400.0)

# Run 200 trajectories
vehicle.runMonteCarloD(200, '../data/werner2019/MCB2-Mars')

BATCH :../data/werner2019/MCB2-Mars, RUN #: 1, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
→ SAMPLE #: 323, EFPA: -11.79, SIGMA: -0.29, APO : 32577.73
BATCH :../data/werner2019/MCB2-Mars, RUN #: 2, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
→ SAMPLE #: 641, EFPA: -11.81, SIGMA: 0.18, APO : 31263.06
BATCH :../data/werner2019/MCB2-Mars, RUN #: 3, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
→ SAMPLE #: 120, EFPA: -11.93, SIGMA: -2.68, APO : 32488.87
BATCH :../data/werner2019/MCB2-Mars, RUN #: 4, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
→ SAMPLE #: 971, EFPA: -11.92, SIGMA: 1.56, APO : 34683.39

/home/athul/anaconda3/lib/python3.7/site-packages/AMAT-2.1.3-py3.7.egg/AMAT/vehicle.
→py:504: RuntimeWarning: invalid value encountered in sqrt
    ans[:] = 1.8980E-8 * (rho_vec[:]/self.RN)**0.5 * v[:]**3.0

BATCH :../data/werner2019/MCB2-Mars, RUN #: 5, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
→ SAMPLE #: 165, EFPA: -11.83, SIGMA: -2.05, APO : 33118.19
BATCH :../data/werner2019/MCB2-Mars, RUN #: 6, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
→ SAMPLE #: 529, EFPA: -11.75, SIGMA: 0.25, APO : 32952.30
BATCH :../data/werner2019/MCB2-Mars, RUN #: 7, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
→ SAMPLE #: 57, EFPA: -11.88, SIGMA: 0.37, APO : 33138.68
BATCH :../data/werner2019/MCB2-Mars, RUN #: 8, PROF: ../atmdata/Mars/LAT00N-N1000.txt,
→ SAMPLE #: 158, EFPA: -11.77, SIGMA: 0.27, APO : 31926.75

```

(continues on next page)

(continued from previous page)

```
BATCH :./data/werner2019/MCB2-Mars, RUN #: 9, PROF: ./atmdata/Mars/LATOON-N1000.txt,
→ SAMPLE #: 623, EFPA: -11.76, SIGMA: 0.86, APO : 32279.26
BATCH :./data/werner2019/MCB2-Mars, RUN #: 10, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 335, EFPA: -11.91, SIGMA: -1.30, APO : 34055.62
BATCH :./data/werner2019/MCB2-Mars, RUN #: 11, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 360, EFPA: -11.79, SIGMA: -1.09, APO : 32848.65
BATCH :./data/werner2019/MCB2-Mars, RUN #: 12, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 679, EFPA: -11.77, SIGMA: 1.58, APO : 31324.80
BATCH :./data/werner2019/MCB2-Mars, RUN #: 13, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 820, EFPA: -11.86, SIGMA: -0.50, APO : 34666.91
BATCH :./data/werner2019/MCB2-Mars, RUN #: 14, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 522, EFPA: -11.76, SIGMA: -1.80, APO : 32589.29
BATCH :./data/werner2019/MCB2-Mars, RUN #: 15, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 356, EFPA: -11.81, SIGMA: -0.98, APO : 32025.72
BATCH :./data/werner2019/MCB2-Mars, RUN #: 16, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 400, EFPA: -11.82, SIGMA: -0.93, APO : 32681.15
BATCH :./data/werner2019/MCB2-Mars, RUN #: 17, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 153, EFPA: -11.77, SIGMA: 2.31, APO : 30904.72
BATCH :./data/werner2019/MCB2-Mars, RUN #: 18, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 321, EFPA: -11.83, SIGMA: -1.58, APO : 32500.60
BATCH :./data/werner2019/MCB2-Mars, RUN #: 19, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 243, EFPA: -11.92, SIGMA: -0.53, APO : 32899.73
BATCH :./data/werner2019/MCB2-Mars, RUN #: 20, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 220, EFPA: -11.82, SIGMA: 0.79, APO : 29928.16
BATCH :./data/werner2019/MCB2-Mars, RUN #: 21, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 761, EFPA: -11.74, SIGMA: 0.01, APO : 32632.82
BATCH :./data/werner2019/MCB2-Mars, RUN #: 22, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 518, EFPA: -11.83, SIGMA: 1.50, APO : 30531.60
BATCH :./data/werner2019/MCB2-Mars, RUN #: 23, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 624, EFPA: -11.68, SIGMA: 0.79, APO : 32593.76
BATCH :./data/werner2019/MCB2-Mars, RUN #: 24, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 553, EFPA: -11.74, SIGMA: 0.73, APO : 33146.31
BATCH :./data/werner2019/MCB2-Mars, RUN #: 25, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 824, EFPA: -11.76, SIGMA: -0.21, APO : 32564.92
BATCH :./data/werner2019/MCB2-Mars, RUN #: 26, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 815, EFPA: -11.76, SIGMA: -0.58, APO : 31947.94
BATCH :./data/werner2019/MCB2-Mars, RUN #: 27, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 543, EFPA: -11.84, SIGMA: -1.83, APO : 32945.79
BATCH :./data/werner2019/MCB2-Mars, RUN #: 28, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 855, EFPA: -11.77, SIGMA: -2.10, APO : 33343.44
BATCH :./data/werner2019/MCB2-Mars, RUN #: 29, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 863, EFPA: -11.90, SIGMA: 0.89, APO : 30733.56
BATCH :./data/werner2019/MCB2-Mars, RUN #: 30, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 809, EFPA: -11.72, SIGMA: -0.91, APO : 33349.24
BATCH :./data/werner2019/MCB2-Mars, RUN #: 31, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 805, EFPA: -11.69, SIGMA: -0.54, APO : 32828.30
BATCH :./data/werner2019/MCB2-Mars, RUN #: 32, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 938, EFPA: -11.85, SIGMA: 2.03, APO : 34293.72
BATCH :./data/werner2019/MCB2-Mars, RUN #: 33, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 271, EFPA: -11.90, SIGMA: -0.44, APO : 33137.81
BATCH :./data/werner2019/MCB2-Mars, RUN #: 34, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 579, EFPA: -11.79, SIGMA: 0.83, APO : 32171.64
BATCH :./data/werner2019/MCB2-Mars, RUN #: 35, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 947, EFPA: -11.91, SIGMA: -1.39, APO : 31584.52
BATCH :./data/werner2019/MCB2-Mars, RUN #: 36, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 851, EFPA: -11.67, SIGMA: 0.35, APO : 32554.18
BATCH :./data/werner2019/MCB2-Mars, RUN #: 37, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 917, EFPA: -11.78, SIGMA: -0.55, APO : 33133.24
```

(continues on next page)

(continued from previous page)

```

BATCH :./data/werner2019/MCB2-Mars, RUN #: 38, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 34, EFPA: -11.84, SIGMA: 0.52, APO : 33299.46
BATCH :./data/werner2019/MCB2-Mars, RUN #: 39, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 417, EFPA: -11.82, SIGMA: 1.67, APO : 33257.16
BATCH :./data/werner2019/MCB2-Mars, RUN #: 40, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 905, EFPA: -11.79, SIGMA: -0.99, APO : 32695.03
BATCH :./data/werner2019/MCB2-Mars, RUN #: 41, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 562, EFPA: -11.83, SIGMA: -0.14, APO : 32995.77
BATCH :./data/werner2019/MCB2-Mars, RUN #: 42, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 951, EFPA: -11.82, SIGMA: -0.46, APO : 31976.55
BATCH :./data/werner2019/MCB2-Mars, RUN #: 43, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 572, EFPA: -11.88, SIGMA: -0.50, APO : 33227.59
BATCH :./data/werner2019/MCB2-Mars, RUN #: 44, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 445, EFPA: -11.84, SIGMA: -0.92, APO : 32261.02
BATCH :./data/werner2019/MCB2-Mars, RUN #: 45, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 127, EFPA: -11.75, SIGMA: -0.11, APO : 32204.43
BATCH :./data/werner2019/MCB2-Mars, RUN #: 46, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 834, EFPA: -11.80, SIGMA: -1.26, APO : 32546.44
BATCH :./data/werner2019/MCB2-Mars, RUN #: 47, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 430, EFPA: -11.83, SIGMA: 0.05, APO : 32395.95
BATCH :./data/werner2019/MCB2-Mars, RUN #: 48, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 764, EFPA: -11.78, SIGMA: -2.64, APO : 32847.73
BATCH :./data/werner2019/MCB2-Mars, RUN #: 49, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 428, EFPA: -11.81, SIGMA: 0.65, APO : 33907.42
BATCH :./data/werner2019/MCB2-Mars, RUN #: 50, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 748, EFPA: -11.82, SIGMA: -0.93, APO : 30363.41
BATCH :./data/werner2019/MCB2-Mars, RUN #: 51, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 576, EFPA: -11.78, SIGMA: 0.36, APO : 32627.78
BATCH :./data/werner2019/MCB2-Mars, RUN #: 52, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 621, EFPA: -11.73, SIGMA: 0.33, APO : 32560.41
BATCH :./data/werner2019/MCB2-Mars, RUN #: 53, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 746, EFPA: -11.88, SIGMA: 3.59, APO : 27525.65
BATCH :./data/werner2019/MCB2-Mars, RUN #: 54, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 299, EFPA: -11.83, SIGMA: -0.79, APO : 33239.30
BATCH :./data/werner2019/MCB2-Mars, RUN #: 55, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 757, EFPA: -11.72, SIGMA: 0.41, APO : 33163.05
BATCH :./data/werner2019/MCB2-Mars, RUN #: 56, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 568, EFPA: -11.76, SIGMA: -1.06, APO : 33302.55
BATCH :./data/werner2019/MCB2-Mars, RUN #: 57, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 945, EFPA: -11.66, SIGMA: -0.64, APO : 32926.05
BATCH :./data/werner2019/MCB2-Mars, RUN #: 58, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 729, EFPA: -11.76, SIGMA: 2.08, APO : 32448.62
BATCH :./data/werner2019/MCB2-Mars, RUN #: 59, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 589, EFPA: -11.69, SIGMA: -1.75, APO : 33098.32
BATCH :./data/werner2019/MCB2-Mars, RUN #: 60, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 488, EFPA: -11.80, SIGMA: -0.44, APO : 33247.83
BATCH :./data/werner2019/MCB2-Mars, RUN #: 61, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 193, EFPA: -11.69, SIGMA: -0.68, APO : 32782.73
BATCH :./data/werner2019/MCB2-Mars, RUN #: 62, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 306, EFPA: -11.79, SIGMA: 0.12, APO : 32744.57
BATCH :./data/werner2019/MCB2-Mars, RUN #: 63, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 459, EFPA: -11.97, SIGMA: 1.23, APO : 29849.99
BATCH :./data/werner2019/MCB2-Mars, RUN #: 64, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 478, EFPA: -11.85, SIGMA: -2.99, APO : 32495.06
BATCH :./data/werner2019/MCB2-Mars, RUN #: 65, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 30, EFPA: -11.88, SIGMA: 1.08, APO : 30837.00
BATCH :./data/werner2019/MCB2-Mars, RUN #: 66, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 752, EFPA: -11.69, SIGMA: -1.15, APO : 32767.21

```

(continues on next page)

(continued from previous page)

```
BATCH :./data/werner2019/MCB2-Mars, RUN #: 67, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 584, EFPA: -11.80, SIGMA: 0.31, APO : 32557.71
BATCH :./data/werner2019/MCB2-Mars, RUN #: 68, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 632, EFPA: -11.86, SIGMA: -0.36, APO : 32676.39
BATCH :./data/werner2019/MCB2-Mars, RUN #: 69, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 709, EFPA: -11.82, SIGMA: -0.93, APO : 32634.53
BATCH :./data/werner2019/MCB2-Mars, RUN #: 70, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 154, EFPA: -11.87, SIGMA: 0.63, APO : 29549.68
BATCH :./data/werner2019/MCB2-Mars, RUN #: 71, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 468, EFPA: -11.77, SIGMA: 0.31, APO : 33146.20
BATCH :./data/werner2019/MCB2-Mars, RUN #: 72, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 428, EFPA: -11.77, SIGMA: -1.48, APO : 33119.29
BATCH :./data/werner2019/MCB2-Mars, RUN #: 73, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 768, EFPA: -11.83, SIGMA: -1.23, APO : 32376.70
BATCH :./data/werner2019/MCB2-Mars, RUN #: 74, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 447, EFPA: -11.74, SIGMA: -0.32, APO : 32195.44
BATCH :./data/werner2019/MCB2-Mars, RUN #: 75, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 556, EFPA: -11.75, SIGMA: -0.06, APO : 32280.00
BATCH :./data/werner2019/MCB2-Mars, RUN #: 76, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 758, EFPA: -11.79, SIGMA: 0.25, APO : 30210.66
BATCH :./data/werner2019/MCB2-Mars, RUN #: 77, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 168, EFPA: -11.93, SIGMA: 2.43, APO : 27460.65
BATCH :./data/werner2019/MCB2-Mars, RUN #: 78, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 143, EFPA: -11.79, SIGMA: 0.25, APO : 33456.57
BATCH :./data/werner2019/MCB2-Mars, RUN #: 79, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 754, EFPA: -11.80, SIGMA: -0.14, APO : 33714.98
BATCH :./data/werner2019/MCB2-Mars, RUN #: 80, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 180, EFPA: -11.71, SIGMA: -0.71, APO : 33022.70
BATCH :./data/werner2019/MCB2-Mars, RUN #: 81, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 898, EFPA: -11.77, SIGMA: -2.06, APO : 32980.24
BATCH :./data/werner2019/MCB2-Mars, RUN #: 82, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 268, EFPA: -11.87, SIGMA: 0.47, APO : 32306.71
BATCH :./data/werner2019/MCB2-Mars, RUN #: 83, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 774, EFPA: -11.77, SIGMA: -2.42, APO : 32904.05
BATCH :./data/werner2019/MCB2-Mars, RUN #: 84, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 530, EFPA: -11.84, SIGMA: 0.14, APO : 32434.25
BATCH :./data/werner2019/MCB2-Mars, RUN #: 85, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 67, EFPA: -11.67, SIGMA: 1.93, APO : 33094.71
BATCH :./data/werner2019/MCB2-Mars, RUN #: 86, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 461, EFPA: -11.85, SIGMA: -0.61, APO : 30606.20
BATCH :./data/werner2019/MCB2-Mars, RUN #: 87, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 243, EFPA: -11.85, SIGMA: -1.20, APO : 33668.52
BATCH :./data/werner2019/MCB2-Mars, RUN #: 88, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 997, EFPA: -11.86, SIGMA: -0.07, APO : 31123.70
BATCH :./data/werner2019/MCB2-Mars, RUN #: 89, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 722, EFPA: -11.91, SIGMA: 1.14, APO : 30216.72
BATCH :./data/werner2019/MCB2-Mars, RUN #: 90, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 348, EFPA: -11.89, SIGMA: -1.22, APO : 33758.54
BATCH :./data/werner2019/MCB2-Mars, RUN #: 91, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 145, EFPA: -11.83, SIGMA: 1.17, APO : 30788.83
BATCH :./data/werner2019/MCB2-Mars, RUN #: 92, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 62, EFPA: -11.69, SIGMA: -0.52, APO : 33140.54
BATCH :./data/werner2019/MCB2-Mars, RUN #: 93, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 994, EFPA: -11.78, SIGMA: 0.41, APO : 31145.02
BATCH :./data/werner2019/MCB2-Mars, RUN #: 94, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 512, EFPA: -11.75, SIGMA: 0.06, APO : 32627.69
BATCH :./data/werner2019/MCB2-Mars, RUN #: 95, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 222, EFPA: -11.77, SIGMA: 0.23, APO : 32152.49
```

(continues on next page)

(continued from previous page)

```

BATCH :./data/werner2019/MCB2-Mars, RUN #: 96, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 611, EFPA: -11.58, SIGMA: -2.00, APO : -59599.06
BATCH :./data/werner2019/MCB2-Mars, RUN #: 97, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 30, EFPA: -11.79, SIGMA: -1.34, APO : 33205.77
BATCH :./data/werner2019/MCB2-Mars, RUN #: 98, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 732, EFPA: -11.75, SIGMA: -0.29, APO : 33147.11
BATCH :./data/werner2019/MCB2-Mars, RUN #: 99, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 64, EFPA: -11.68, SIGMA: 0.07, APO : 37376.72
BATCH :./data/werner2019/MCB2-Mars, RUN #: 100, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 343, EFPA: -11.70, SIGMA: 0.99, APO : 32539.10
BATCH :./data/werner2019/MCB2-Mars, RUN #: 101, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 906, EFPA: -11.84, SIGMA: 0.31, APO : 32517.83
BATCH :./data/werner2019/MCB2-Mars, RUN #: 102, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 533, EFPA: -11.76, SIGMA: -1.08, APO : 33033.41
BATCH :./data/werner2019/MCB2-Mars, RUN #: 103, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 329, EFPA: -11.71, SIGMA: 0.74, APO : 32401.12
BATCH :./data/werner2019/MCB2-Mars, RUN #: 104, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 510, EFPA: -11.83, SIGMA: -1.40, APO : 33176.79
BATCH :./data/werner2019/MCB2-Mars, RUN #: 105, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 778, EFPA: -11.77, SIGMA: 0.74, APO : 32348.53
BATCH :./data/werner2019/MCB2-Mars, RUN #: 106, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 555, EFPA: -11.80, SIGMA: -0.74, APO : 32509.25
BATCH :./data/werner2019/MCB2-Mars, RUN #: 107, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 803, EFPA: -11.74, SIGMA: -0.58, APO : 33237.60
BATCH :./data/werner2019/MCB2-Mars, RUN #: 108, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 108, EFPA: -11.83, SIGMA: 1.76, APO : 33437.03
BATCH :./data/werner2019/MCB2-Mars, RUN #: 109, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 906, EFPA: -11.75, SIGMA: -0.77, APO : 32963.40
BATCH :./data/werner2019/MCB2-Mars, RUN #: 110, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 655, EFPA: -11.81, SIGMA: 0.13, APO : 32307.01
BATCH :./data/werner2019/MCB2-Mars, RUN #: 111, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 972, EFPA: -11.75, SIGMA: -0.53, APO : 32921.98
BATCH :./data/werner2019/MCB2-Mars, RUN #: 112, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 651, EFPA: -11.72, SIGMA: 1.04, APO : 33145.04
BATCH :./data/werner2019/MCB2-Mars, RUN #: 113, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 202, EFPA: -11.80, SIGMA: 0.89, APO : 32833.74
BATCH :./data/werner2019/MCB2-Mars, RUN #: 114, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 596, EFPA: -11.78, SIGMA: 0.21, APO : 32579.24
BATCH :./data/werner2019/MCB2-Mars, RUN #: 115, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 911, EFPA: -11.88, SIGMA: -0.81, APO : 35026.98
BATCH :./data/werner2019/MCB2-Mars, RUN #: 116, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 978, EFPA: -11.76, SIGMA: 0.09, APO : 33354.20
BATCH :./data/werner2019/MCB2-Mars, RUN #: 117, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 194, EFPA: -11.80, SIGMA: -1.53, APO : 33170.99
BATCH :./data/werner2019/MCB2-Mars, RUN #: 118, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 63, EFPA: -11.81, SIGMA: -1.53, APO : 31784.22
BATCH :./data/werner2019/MCB2-Mars, RUN #: 119, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 815, EFPA: -11.93, SIGMA: 0.41, APO : 32031.26
BATCH :./data/werner2019/MCB2-Mars, RUN #: 120, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 953, EFPA: -11.76, SIGMA: -0.48, APO : 32281.00
BATCH :./data/werner2019/MCB2-Mars, RUN #: 121, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 288, EFPA: -11.74, SIGMA: -1.51, APO : 32770.89
BATCH :./data/werner2019/MCB2-Mars, RUN #: 122, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 818, EFPA: -11.87, SIGMA: -1.11, APO : 31876.10
BATCH :./data/werner2019/MCB2-Mars, RUN #: 123, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 404, EFPA: -11.80, SIGMA: -0.95, APO : 34060.29
BATCH :./data/werner2019/MCB2-Mars, RUN #: 124, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 231, EFPA: -11.88, SIGMA: 2.04, APO : 28912.72

```

(continues on next page)

(continued from previous page)

```
BATCH :./data/werner2019/MCB2-Mars, RUN #: 125, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 807, EFPA: -11.86, SIGMA: -0.17, APO : 32252.30
BATCH :./data/werner2019/MCB2-Mars, RUN #: 126, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 873, EFPA: -11.78, SIGMA: 1.00, APO : 30770.96
BATCH :./data/werner2019/MCB2-Mars, RUN #: 127, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 833, EFPA: -11.83, SIGMA: 0.53, APO : 33261.34
BATCH :./data/werner2019/MCB2-Mars, RUN #: 128, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 252, EFPA: -11.87, SIGMA: -0.85, APO : 30178.20
BATCH :./data/werner2019/MCB2-Mars, RUN #: 129, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 565, EFPA: -11.68, SIGMA: -0.44, APO : 33026.39
BATCH :./data/werner2019/MCB2-Mars, RUN #: 130, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 382, EFPA: -11.83, SIGMA: 0.03, APO : 32210.04
BATCH :./data/werner2019/MCB2-Mars, RUN #: 131, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 612, EFPA: -11.73, SIGMA: -1.17, APO : 48514.86
BATCH :./data/werner2019/MCB2-Mars, RUN #: 132, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 937, EFPA: -11.80, SIGMA: 1.55, APO : 31901.56
BATCH :./data/werner2019/MCB2-Mars, RUN #: 133, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 894, EFPA: -11.83, SIGMA: -0.26, APO : 31639.84
BATCH :./data/werner2019/MCB2-Mars, RUN #: 134, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 436, EFPA: -11.78, SIGMA: 1.19, APO : 32842.71
BATCH :./data/werner2019/MCB2-Mars, RUN #: 135, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 856, EFPA: -11.76, SIGMA: 1.05, APO : 31218.55
BATCH :./data/werner2019/MCB2-Mars, RUN #: 136, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 731, EFPA: -11.73, SIGMA: 0.53, APO : 33138.80
BATCH :./data/werner2019/MCB2-Mars, RUN #: 137, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 365, EFPA: -11.91, SIGMA: -0.43, APO : 29544.45
BATCH :./data/werner2019/MCB2-Mars, RUN #: 138, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 131, EFPA: -11.86, SIGMA: -1.29, APO : 32882.69
BATCH :./data/werner2019/MCB2-Mars, RUN #: 139, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 928, EFPA: -11.76, SIGMA: 0.51, APO : 32908.79
BATCH :./data/werner2019/MCB2-Mars, RUN #: 140, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 675, EFPA: -11.84, SIGMA: -0.44, APO : 33420.71
BATCH :./data/werner2019/MCB2-Mars, RUN #: 141, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 771, EFPA: -11.78, SIGMA: -0.55, APO : 32648.17
BATCH :./data/werner2019/MCB2-Mars, RUN #: 142, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 218, EFPA: -11.94, SIGMA: -0.58, APO : 32131.56
BATCH :./data/werner2019/MCB2-Mars, RUN #: 143, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 674, EFPA: -11.75, SIGMA: -1.88, APO : 32942.75
BATCH :./data/werner2019/MCB2-Mars, RUN #: 144, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 526, EFPA: -11.84, SIGMA: 1.02, APO : 32150.60
BATCH :./data/werner2019/MCB2-Mars, RUN #: 145, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 414, EFPA: -11.88, SIGMA: 0.04, APO : 30167.78
BATCH :./data/werner2019/MCB2-Mars, RUN #: 146, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 971, EFPA: -11.83, SIGMA: -0.76, APO : 35221.26
BATCH :./data/werner2019/MCB2-Mars, RUN #: 147, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 644, EFPA: -11.90, SIGMA: 1.13, APO : 31257.53
BATCH :./data/werner2019/MCB2-Mars, RUN #: 148, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 863, EFPA: -11.85, SIGMA: -0.23, APO : 32058.29
BATCH :./data/werner2019/MCB2-Mars, RUN #: 149, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 304, EFPA: -11.77, SIGMA: -0.91, APO : 33121.24
BATCH :./data/werner2019/MCB2-Mars, RUN #: 150, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 202, EFPA: -11.91, SIGMA: -1.65, APO : 32489.36
BATCH :./data/werner2019/MCB2-Mars, RUN #: 151, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 219, EFPA: -11.82, SIGMA: -0.08, APO : 31031.43
BATCH :./data/werner2019/MCB2-Mars, RUN #: 152, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 225, EFPA: -11.82, SIGMA: -1.00, APO : 33058.62
BATCH :./data/werner2019/MCB2-Mars, RUN #: 153, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 298, EFPA: -11.74, SIGMA: -1.00, APO : 32822.08
```

(continues on next page)

(continued from previous page)

```

BATCH :./data/werner2019/MCB2-Mars, RUN #: 154, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 120, EFPA: -11.72, SIGMA: -0.17, APO : 33055.29
BATCH :./data/werner2019/MCB2-Mars, RUN #: 155, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 73, EFPA: -11.77, SIGMA: -1.34, APO : 33135.85
BATCH :./data/werner2019/MCB2-Mars, RUN #: 156, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 890, EFPA: -11.74, SIGMA: -1.78, APO : 32952.39
BATCH :./data/werner2019/MCB2-Mars, RUN #: 157, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 414, EFPA: -11.82, SIGMA: 0.45, APO : 29928.26
BATCH :./data/werner2019/MCB2-Mars, RUN #: 158, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 522, EFPA: -11.86, SIGMA: 1.85, APO : 34118.29
BATCH :./data/werner2019/MCB2-Mars, RUN #: 159, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 489, EFPA: -11.81, SIGMA: 0.70, APO : 32025.62
BATCH :./data/werner2019/MCB2-Mars, RUN #: 160, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 644, EFPA: -11.76, SIGMA: -0.35, APO : 32760.02
BATCH :./data/werner2019/MCB2-Mars, RUN #: 161, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 764, EFPA: -11.91, SIGMA: -0.32, APO : 29380.60
BATCH :./data/werner2019/MCB2-Mars, RUN #: 162, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 596, EFPA: -11.72, SIGMA: 0.18, APO : 32955.43
BATCH :./data/werner2019/MCB2-Mars, RUN #: 163, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 498, EFPA: -11.88, SIGMA: -1.44, APO : 34005.35
BATCH :./data/werner2019/MCB2-Mars, RUN #: 164, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 615, EFPA: -11.78, SIGMA: 1.76, APO : 32286.27
BATCH :./data/werner2019/MCB2-Mars, RUN #: 165, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 944, EFPA: -11.87, SIGMA: 0.30, APO : 31066.50
BATCH :./data/werner2019/MCB2-Mars, RUN #: 166, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 406, EFPA: -11.83, SIGMA: -0.12, APO : 31599.04
BATCH :./data/werner2019/MCB2-Mars, RUN #: 167, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 829, EFPA: -11.78, SIGMA: -0.43, APO : 34139.57
BATCH :./data/werner2019/MCB2-Mars, RUN #: 168, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 180, EFPA: -11.82, SIGMA: -0.42, APO : 32186.09
BATCH :./data/werner2019/MCB2-Mars, RUN #: 169, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 765, EFPA: -11.70, SIGMA: -1.12, APO : 32991.40
BATCH :./data/werner2019/MCB2-Mars, RUN #: 170, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 267, EFPA: -11.82, SIGMA: 0.33, APO : 32976.77
BATCH :./data/werner2019/MCB2-Mars, RUN #: 171, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 304, EFPA: -11.74, SIGMA: 0.27, APO : 32222.07
BATCH :./data/werner2019/MCB2-Mars, RUN #: 172, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 176, EFPA: -11.86, SIGMA: -1.23, APO : 32461.76
BATCH :./data/werner2019/MCB2-Mars, RUN #: 173, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 303, EFPA: -11.79, SIGMA: 1.18, APO : 30793.02
BATCH :./data/werner2019/MCB2-Mars, RUN #: 174, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 563, EFPA: -11.83, SIGMA: -1.24, APO : 33081.93
BATCH :./data/werner2019/MCB2-Mars, RUN #: 175, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 635, EFPA: -11.77, SIGMA: 0.56, APO : 31688.48
BATCH :./data/werner2019/MCB2-Mars, RUN #: 176, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 59, EFPA: -11.79, SIGMA: -1.15, APO : 32489.12
BATCH :./data/werner2019/MCB2-Mars, RUN #: 177, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 42, EFPA: -11.67, SIGMA: -0.55, APO : 33136.43
BATCH :./data/werner2019/MCB2-Mars, RUN #: 178, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 853, EFPA: -11.77, SIGMA: -1.97, APO : 32841.73
BATCH :./data/werner2019/MCB2-Mars, RUN #: 179, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 367, EFPA: -11.79, SIGMA: -0.33, APO : 32174.99
BATCH :./data/werner2019/MCB2-Mars, RUN #: 180, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 390, EFPA: -11.98, SIGMA: -0.04, APO : 37951.84
BATCH :./data/werner2019/MCB2-Mars, RUN #: 181, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 210, EFPA: -11.85, SIGMA: 0.31, APO : 32692.22
BATCH :./data/werner2019/MCB2-Mars, RUN #: 182, PROF: ./atmdata/Mars/LATOON-N1000.
↪txt, SAMPLE #: 185, EFPA: -11.84, SIGMA: 2.36, APO : 30077.14

```

(continues on next page)

(continued from previous page)

```
BATCH :./data/werner2019/MCB2-Mars, RUN #: 183, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 710, EFPA: -11.84, SIGMA: -0.56, APO : 31625.02
BATCH :./data/werner2019/MCB2-Mars, RUN #: 184, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 891, EFPA: -11.80, SIGMA: 0.02, APO : 32301.29
BATCH :./data/werner2019/MCB2-Mars, RUN #: 185, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 643, EFPA: -11.82, SIGMA: 0.07, APO : 31366.44
BATCH :./data/werner2019/MCB2-Mars, RUN #: 186, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 414, EFPA: -11.80, SIGMA: -0.10, APO : 30194.61
BATCH :./data/werner2019/MCB2-Mars, RUN #: 187, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 282, EFPA: -11.76, SIGMA: -0.25, APO : 31947.25
BATCH :./data/werner2019/MCB2-Mars, RUN #: 188, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 188, EFPA: -11.83, SIGMA: 1.38, APO : 32456.24
BATCH :./data/werner2019/MCB2-Mars, RUN #: 189, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 365, EFPA: -11.89, SIGMA: 0.36, APO : 30324.04
BATCH :./data/werner2019/MCB2-Mars, RUN #: 190, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 276, EFPA: -11.82, SIGMA: -0.62, APO : 32557.33
BATCH :./data/werner2019/MCB2-Mars, RUN #: 191, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 277, EFPA: -11.78, SIGMA: -1.29, APO : 32861.95
BATCH :./data/werner2019/MCB2-Mars, RUN #: 192, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 397, EFPA: -11.67, SIGMA: 0.77, APO : 32877.94
BATCH :./data/werner2019/MCB2-Mars, RUN #: 193, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 428, EFPA: -11.82, SIGMA: 0.55, APO : 32704.97
BATCH :./data/werner2019/MCB2-Mars, RUN #: 194, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 307, EFPA: -11.87, SIGMA: -0.03, APO : 32033.48
BATCH :./data/werner2019/MCB2-Mars, RUN #: 195, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 191, EFPA: -11.84, SIGMA: -0.11, APO : 32431.21
BATCH :./data/werner2019/MCB2-Mars, RUN #: 196, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 837, EFPA: -11.77, SIGMA: -0.40, APO : 31984.49
BATCH :./data/werner2019/MCB2-Mars, RUN #: 197, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 419, EFPA: -11.74, SIGMA: 1.65, APO : 32412.90
BATCH :./data/werner2019/MCB2-Mars, RUN #: 198, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 922, EFPA: -11.89, SIGMA: -1.35, APO : 33031.33
BATCH :./data/werner2019/MCB2-Mars, RUN #: 199, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 505, EFPA: -11.85, SIGMA: 0.12, APO : 33073.55
BATCH :./data/werner2019/MCB2-Mars, RUN #: 200, PROF: ./atmdata/Mars/LATOON-N1000.
→txt, SAMPLE #: 410, EFPA: -11.83, SIGMA: -0.19, APO : 31926.59
```

Load the terminal apoapsis, deceleration and heat rate data.

```
[14]: apoap = np.loadtxt('../data/werner2019/MCB2-Mars/terminal_apoapsis_arr.txt')
peria = np.loadtxt('../data/werner2019/MCB2-Mars/terminal_periapsis_arr.txt')
decel = np.loadtxt('../data/werner2019/MCB2-Mars/acc_net_g_max_arr.txt')
heatr = np.loadtxt('../data/werner2019/MCB2-Mars/q_stag_max_arr.txt')
```

Remove cases which resulted in apoapsis < 30000 km or > 36000 km.

```
[15]: del_index_low = np.where(apoap < 28000)
print(del_index_low)

(array([52, 76, 95]),)
```

Seven cases resulted in apoapsis less than 25000 km, and are considered to crash.

Remove these entries from the data before plotting.

```
[16]: apoap_clean = np.delete(apoap, del_index_low)
peria_clean = np.delete(peria, del_index_low)
```

(continues on next page)

(continued from previous page)

```
decel_clean = np.delete(decel, del_index_low)
heatr_clean = np.delete(heatr, del_index_low)
```

Also remove cases with apoapsis > 36000 km.

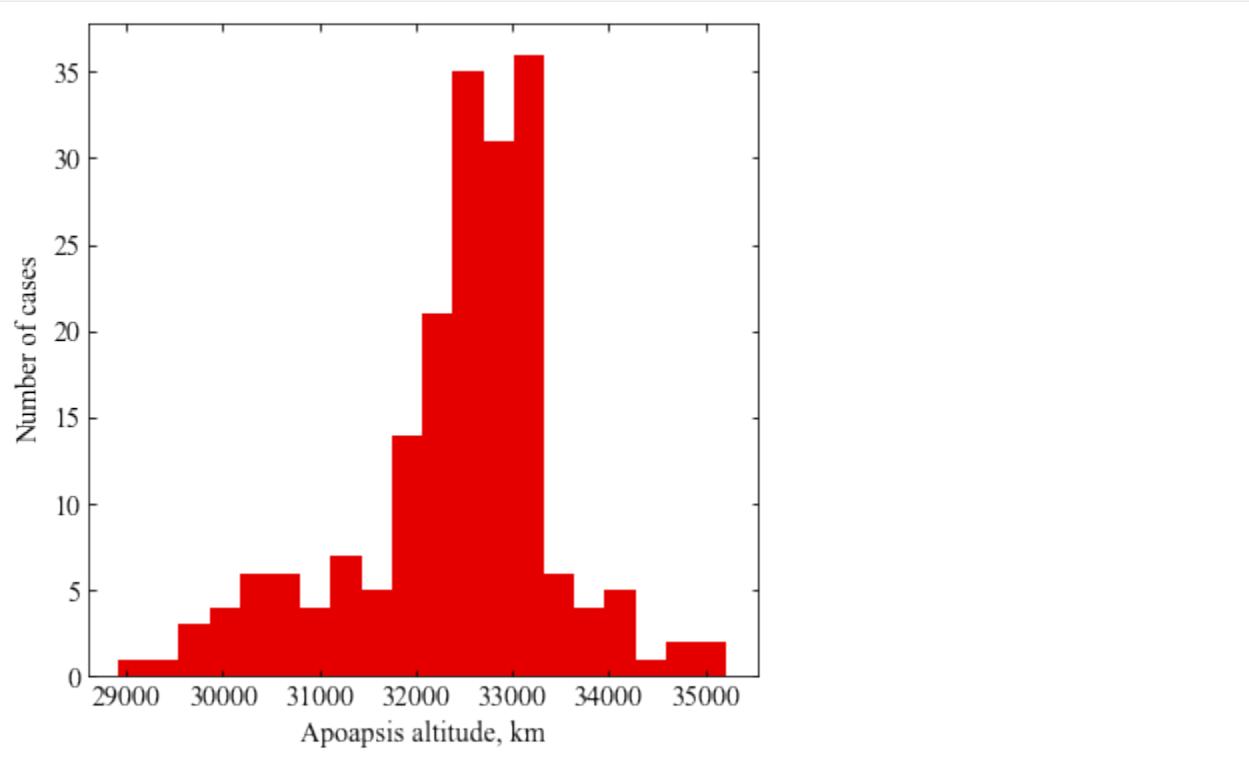
```
[17]: del_index_high = np.where(apoap_clean > 36000)
print(del_index_high)

(array([ 95, 127, 176]),)
```

```
[18]: apoap_clean2 = np.delete(apoap_clean, del_index_high)
peria_clean2 = np.delete(peria_clean, del_index_high)
decel_clean2 = np.delete(decel_clean, del_index_high)
heatr_clean2 = np.delete(heatr_clean, del_index_high)
```

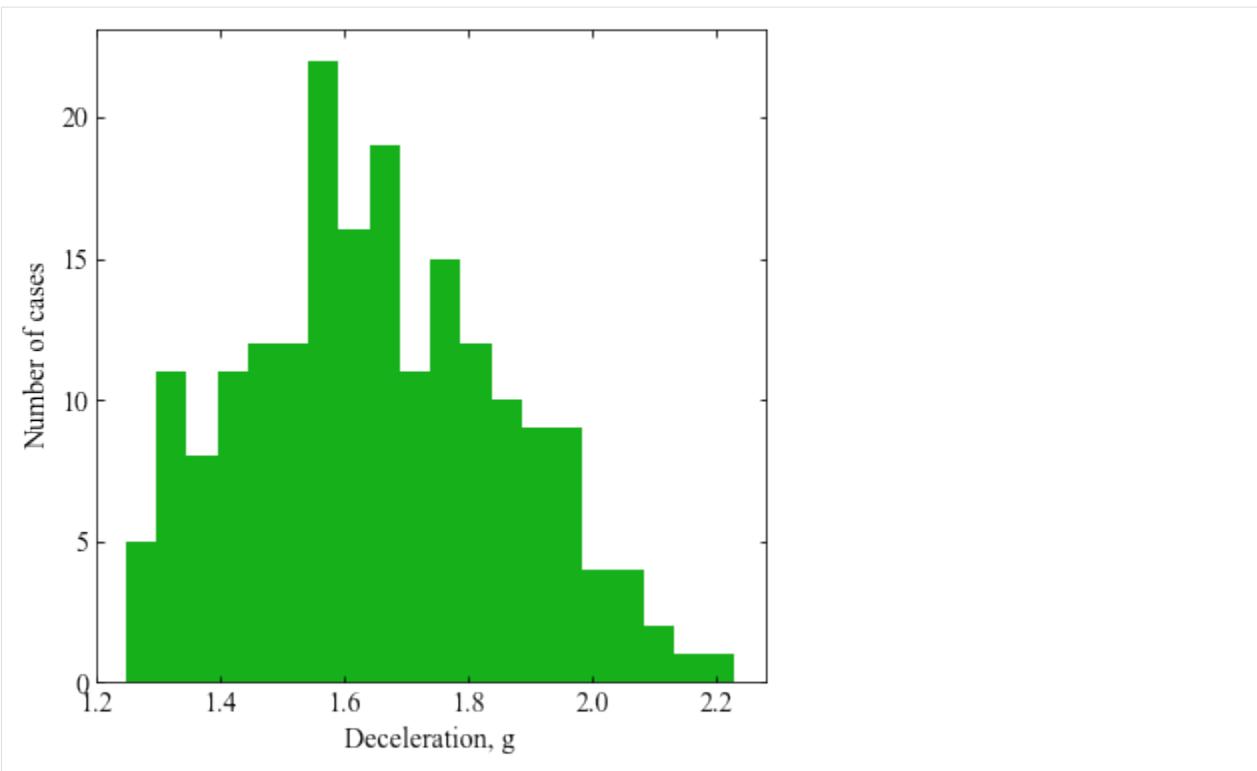
```
[25]: plt.figure(figsize=(6,6))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.hist(apoap_clean2, bins=20, color='xkcd:red')
plt.xlabel('Apoapsis altitude, km', fontsize=14)
plt.ylabel('Number of cases', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.savefig('../plots/werner-smallsat-apoa-hist-mars-2.png',bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-apoa-hist-mars-2.pdf', dpi=300,bbox_inches=
    ~'tight')
plt.savefig('../plots/werner-smallsat-apoa-hist-mars-2.eps', dpi=300,bbox_inches=
    ~'tight')
plt.show()
```



```
[20]: plt.figure(figsize=(6, 6))
plt.rc('font', family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.hist(decel_clean2, bins=20, color='xkcd:green')
plt.xlabel('Deceleration, g', fontsize=14)
plt.ylabel('Number of cases', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

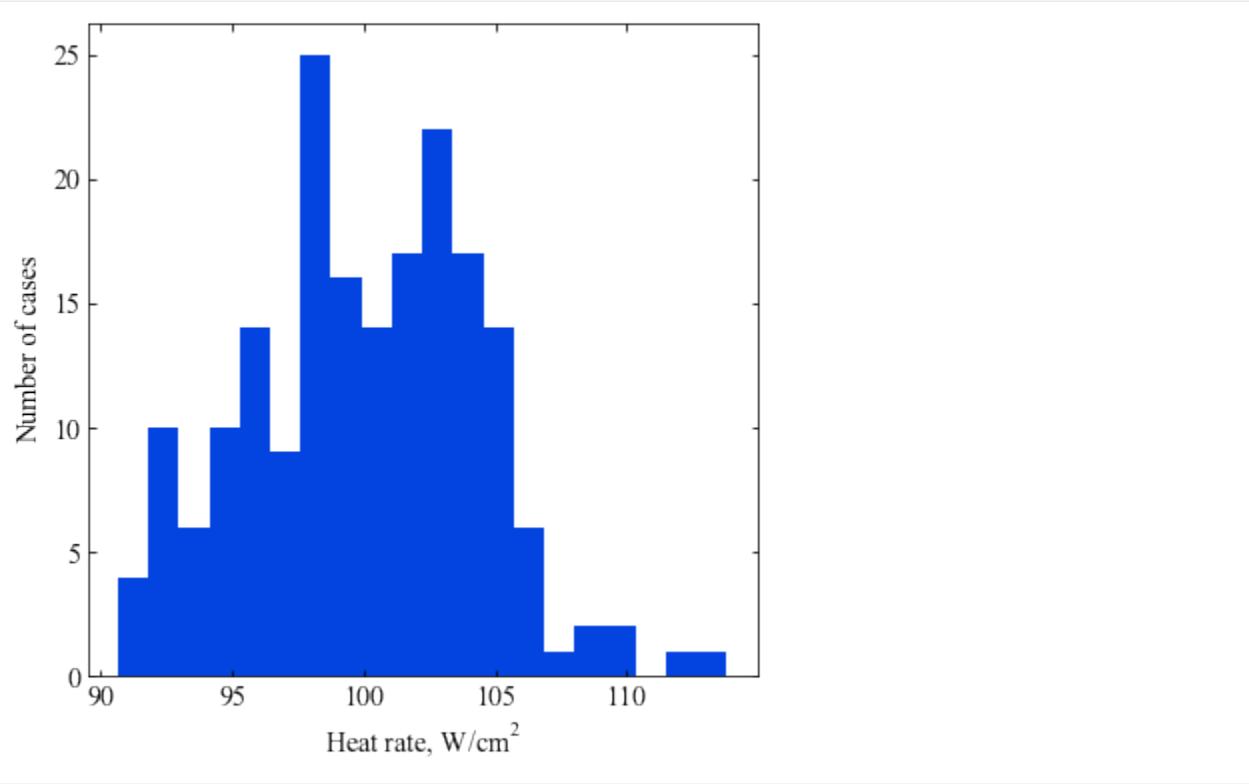
plt.savefig('../plots/werner-smallsat-decel-hist-mars-2.png',bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-decel-hist-mars-2.pdf', dpi=300,bbox_inches=
    ↪'tight')
plt.savefig('../plots/werner-smallsat-decel-hist-mars-2.eps', dpi=300,bbox_inches=
    ↪'tight')
plt.show()
```



```
[21]: plt.figure(figsize=(6, 6))
plt.rc('font', family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.hist(heatr_clean2, bins=20, color='xkcd:blue')
plt.xlabel('Heat rate, '+r'$W/cm^2$', fontsize=14)
plt.ylabel('Number of cases', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

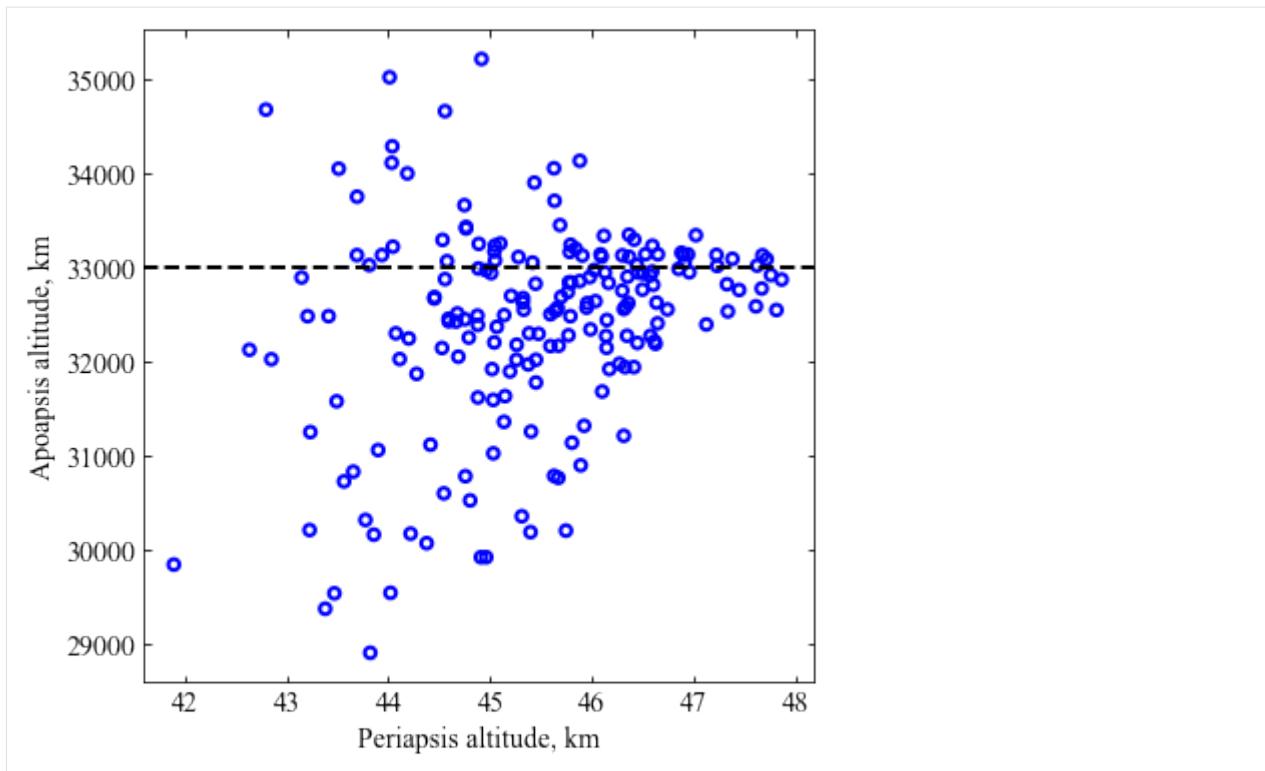
plt.savefig('../plots/werner-smallsat-heat-hist-mars-2.png',bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-heat-hist-mars-2.pdf', dpi=300,bbox_inches=
    ↪'tight')
plt.savefig('../plots/werner-smallsat-heat-hist-mars-2.eps', dpi=300,bbox_inches=
    ↪'tight')
plt.show()

/home/athul/anaconda3/lib/python3.7/site-packages/numpy/lib/histograms.py:824:_
    ↪RuntimeWarning: invalid value encountered in greater_equal
    keep = (tmp_a >= first_edge)
/home/athul/anaconda3/lib/python3.7/site-packages/numpy/lib/histograms.py:825:_
    ↪RuntimeWarning: invalid value encountered in less_equal
    keep &= (tmp_a <= last_edge)
```



```
[22]: plt.figure(figsize=(6, 6))
plt.rc('font', family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.scatter(peria_clean2, apoap_clean2, s=30, facecolors='none', edgecolors='b', lw=2)
plt.axhline(y=33000, lw=2.0, ls='dashed', color='k')
plt.xlabel('Periapsis altitude, km', fontsize=14)
plt.ylabel('Apoapsis altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/werner-smallsat-apoa-peri-mars-2.png', bbox_inches='tight')
plt.savefig('../plots/werner-smallsat-apoa-peri-mars-2.pdf', dpi=300, bbox_inches=
    ↪'tight')
plt.savefig('../plots/werner-smallsat-apoa-peri-mars-2.eps', dpi=300, bbox_inches=
    ↪'tight')
plt.show()
```



Find the percentage of cases which achieved apoapsis between 100 and 800 km.

```
[23]: (np.size(apoap) - np.size(del_index_low) - np.size(del_index_high)) * 100 / np.size(apoap)
[23]: 97.0
```

4.54 Example - 54 - ExoMars 2016 - Mars

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

This notebook simulates the atmospheric entry of the Schiaparelli EDM, flown in the ExoMars 2020 mission. https://en.wikipedia.org/wiki/Schiaparelli_EDM

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("MARS")
planet.h_skip = 120.0E3
planet.h_trap = 2.0E3
planet.loadAtmosphereModel('../atmdata/Mars/mars-gram-avg.dat', 0, 1, 2, 3)
```

```
[4]: # Set up the vehicle
vehicle=Vehicle('Schiaparelli', 577.0, 75.0, 0.00, np.pi*2.4**2.0, 0.0, 0.6, planet)
```

```
[5]: # Set up entry parameters
vehicle.setInitialState(120.0,0.0,0.0,5.900,0.0,-12.48,0.0,0.0)

[6]: # Set up solver
vehicle.setSolverParams(1E-6)

[7]: # Propogate vehicle entry trajectory
vehicle.propogateEntry (30*60.0,0.1,0.0)

[8]: # import rcParams to set figure font type
from matplotlib import rcParams

[9]: fig = plt.figure(figsize=(12,8))
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

plt.subplot(2, 2, 1)
plt.plot(vehicle.v_kmsc, vehicle.h_kmc, 'k-', linewidth=2.0)
plt.xlabel('Speed, km/s', fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 2)
plt.plot(vehicle.acc_net_g, vehicle.h_kmc, 'b-', linewidth=2.0)
plt.xlabel('Deceleration, Earth g',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 3)
plt.plot(vehicle.q_stag_total, vehicle.h_kmc,'r-', linewidth=2.0)
plt.xlabel('Stagnation point heat-rate, '+r'$W/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x',labelsize=14)
ax.tick_params(axis='y',labelsize=14)

plt.subplot(2, 2, 4)
plt.plot(vehicle.heatload/1.0E3, vehicle.h_kmc, 'm-', linewidth=2.0)
plt.xlabel('Stagnation point heat-load, '+r'$kJ/cm^2$',fontsize=14)
plt.ylabel('Altitude, km', fontsize=14)
ax=plt.gca()
```

(continues on next page)

(continued from previous page)

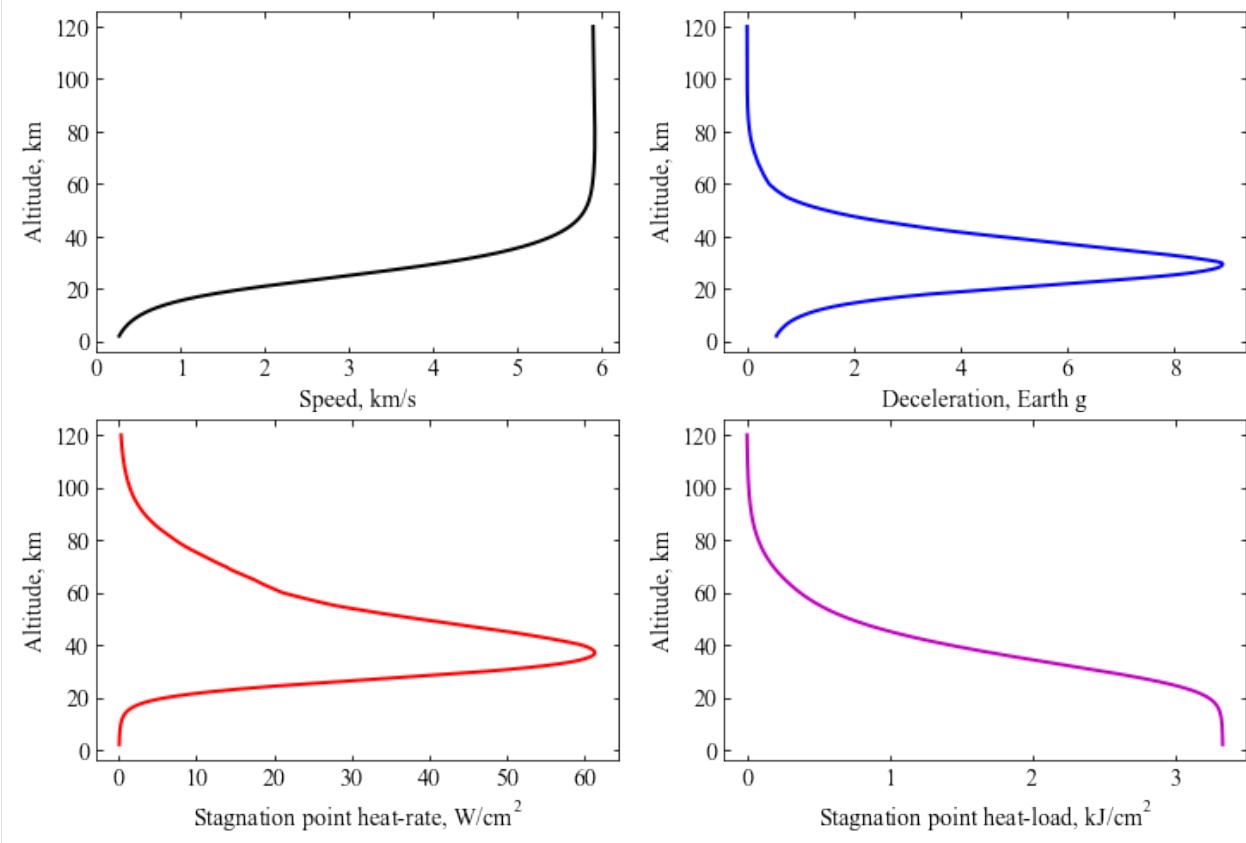
```

ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)

plt.savefig('../plots/exomars-2016-mars.png', bbox_inches='tight')
plt.savefig('../plots/exomars-2016-mars.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/exomars-2016-mars.eps', dpi=300, bbox_inches='tight')

plt.show()

```



Compare to results from <https://doi.org/10.1016/j.actaastro.2018.05.029>

4.55 Example - 55 - Titan Aerocapture Systems Study - Part 1

This examples reproduces results from Way, Powell et al., Aerocapture Simulation and Performance for the Titan Explorer Mission, 39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit 20-23 July 2003, Huntsville, Alabama. <https://doi.org/10.2514/6.2003-4951>

```
[1]: from IPython.display import Image
Image(filename='../plots/titan-aerocapture-systems.png', width=500)
```

[1]:

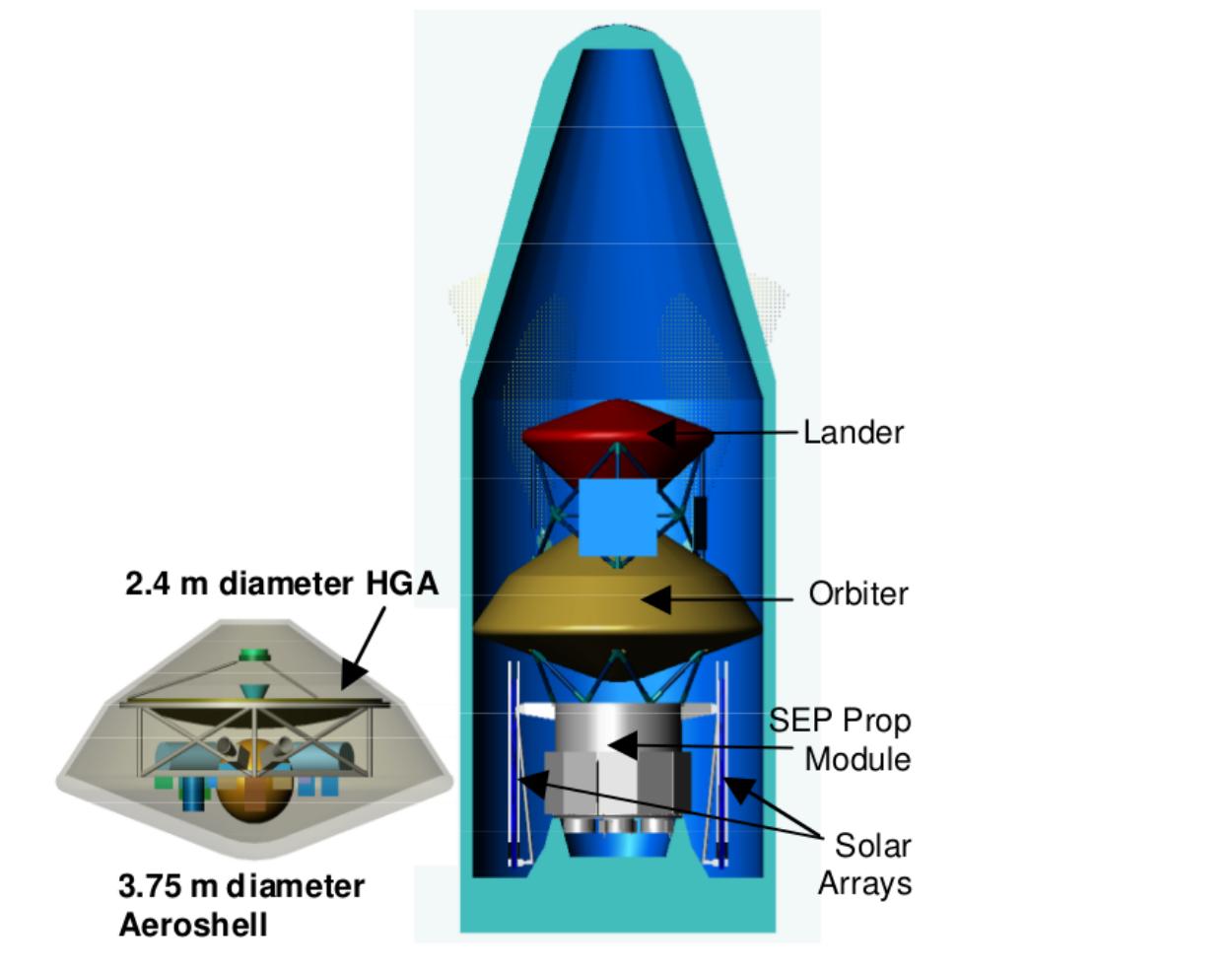


Image credit: Lockwood et al.

```
[2]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[3]: import numpy as np
from scipy import interpolate
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib import rcParams
```

```
[4]: planet = Planet('TITAN')
planet.loadAtmosphereModel('../atmdata/Titan/titan-gram-avg.dat', 0, 1, 2, 3)
planet.h_skip = 1000.0E3
```

```
[5]: vehicle=Vehicle('TitanAC', 818.0, 90.0, 0.25, np.pi*0.25*3.75**2, 0.0, 0.91, planet)
vehicle.setInitialState(1000.0,0.0,0.0,6.5,0.0,-25.00,0.0,0.0)
vehicle.setSolverParams(1E-6)
```

First, find the corridor bounds to select a target nominal EFPA for a nominal Mars atmosphere.

```
[7]: underShootLimit, exitflag_us = vehicle.findUnderShootLimit(60.0*60.0, 1.0, -50.0,-5.0,
                                                               ↪ 1E-10, 1700.0)
overShootLimit , exitflag_os = vehicle.findOverShootLimit(60.0*60.0, 1.0, -50.0,-5.0,
                                                               ↪ 1E-10, 1700.0)

print("Corridor bounds for nominal atmosphere.")
print("-----")
print(underShootLimit, exitflag_us)
print(overShootLimit, exitflag_os)
print("-----")

Corridor bounds for nominal atmosphere.
-----
-37.83097717905548 1.0
-34.28877212249972 1.0
-----
```

It is good practice to see how these corridor bounds vary with minimum, average, and maximum density atmospheres. We will check how much these corridor bounds change using +/- 3-sigma bounds from Titan GRAM output files.

Load three density profiles for minimum, average, and maximum scenarios from a GRAM output file.

```
[9]: ATM_height1, ATM_density_low1, ATM_density_avg1, ATM_density_high1, ATM_density_pert1 =
      ↪ planet.loadMonteCarloDensityFile2('../atmdata/Titan/FMINMAX-10.txt', 0, 1, 2, 3,
      ↪ 4, heightInKmFlag=True)
ATM_height2, ATM_density_low2, ATM_density_avg2, ATM_density_high2, ATM_density_pert2 =
      ↪ planet.loadMonteCarloDensityFile2('../atmdata/Titan/FMINMAX+00.txt', 0, 1, 2, 3,
      ↪ 4, heightInKmFlag=True)
ATM_height3, ATM_density_low3, ATM_density_avg3, ATM_density_high3, ATM_density_pert3 =
      ↪ planet.loadMonteCarloDensityFile2('../atmdata/Titan/FMINMAX+10.txt', 0, 1, 2, 3,
      ↪ 4, heightInKmFlag=True)

density_int_low = planet.loadAtmosphereModel5(ATM_height1, ATM_density_low1, ATM_
      ↪ density_avg1, ATM_density_high1, ATM_density_pert1, -3.0, 201, 1)
density_int_avg = planet.loadAtmosphereModel5(ATM_height2, ATM_density_low2, ATM_
      ↪ density_avg2, ATM_density_high2, ATM_density_pert2, 0.0, 201, 1)
density_int_hig = planet.loadAtmosphereModel5(ATM_height3, ATM_density_low3, ATM_
      ↪ density_avg3, ATM_density_high3, ATM_density_pert3, +3.0, 201, 1)
```

Set the planet density_int attribute to density_int_low

```
[10]: planet.density_int = density_int_low
```

Compute the corridor bounds for low density atmosphere.

```
[11]: underShootLimit, exitflag_us = vehicle.findUnderShootLimit(60.0*60.0, 1.0, -50.0,-5.0,
                                                               ↪ 1E-10, 1700.0)
overShootLimit , exitflag_os = vehicle.findOverShootLimit(60.0*60.0, 1.0, -50.0,-5.0,
                                                               ↪ 1E-10, 1700.0)

print("Low density atmosphere corridor bounds.")
print("-----")
print(underShootLimit, exitflag_us)
print(overShootLimit, exitflag_os)
print("-----")
```

```
Low density atmosphere corridor bounds.
```

```
-----  
-38.47725151913437 1.0  
-35.15044797081828 1.0  
-----
```

Repeat for average and maximum density atmospheres.

```
[13]: planet.density_int = density_int_avg
```

```
[14]: underShootLimit, exitflag_us = vehicle.findUnderShootLimit(60.0*60.0, 1.0, -50.0,-5.0,  
    ↪ 1E-10, 1700.0)  
overShootLimit , exitflag_os = vehicle.findOverShootLimit(60.0*60.0, 1.0, -50.0,-5.0,  
    ↪ 1E-10, 1700.0)  
  
print("Average density atmosphere corridor bounds.")  
print("-----")  
print(underShootLimit, exitflag_us)  
print(overShootLimit, exitflag_os)  
print("-----")
```

```
Average density atmosphere corridor bounds.
```

```
-----  
-37.86181146377203 1.0  
-34.38959433911805 1.0  
-----
```

```
[15]: planet.density_int = density_int_hig
```

```
[16]: underShootLimit, exitflag_us = vehicle.findUnderShootLimit(60.0*60.0, 1.0, -50.0,-5.0,  
    ↪ 1E-10, 1700.0)  
overShootLimit , exitflag_os = vehicle.findOverShootLimit(60.0*60.0, 1.0, -50.0,-5.0,  
    ↪ 1E-10, 1700.0)  
  
print("High density atmosphere corridor bounds.")  
print("-----")  
print(underShootLimit, exitflag_us)  
print(overShootLimit, exitflag_os)  
print("-----")
```

```
High density atmosphere corridor bounds.
```

```
-----  
-37.287857468882066 1.0  
-33.65637210436944 1.0  
-----
```

The corridor is approximately 3.5 deg wide for the nominal atmosphere.

```
[17]: 37.86 - 34.39
```

```
[17]: 3.469999999999999
```

The mid-corridor is typically a good place to start. We will use the mean of the corridor bounds for the average atmosphere.

```
[18]: 0.5*(-37.86 - 34.39)
[18]: -36.125
```

We will propagate a bounding trajectories to illustrate full lift up and full lift down aerocapture trajectories at Titan using this vehicle.

```
[21]: planet.density_int = density_int_avg
```

```
[27]: # Reset initial conditions and propagate overshoot trajectory
vehicle.setInitialState(1000.0,0.0,0.0,6.5,0.0,-34.38959433911805,0.0,0.0)
vehicle.propagateEntry (60.0*60.0,1.0,180.0)

# Extract and save variables to plot
t_min_os      = vehicle.t_minc
h_km_os       = vehicle.h_kmc
acc_net_g_os  = vehicle.acc_net_g
q_stag_con_os = vehicle.q_stag_con
q_stag_rad_os = vehicle.q_stag_rad

# Reset initial conditions and propagate undershoot trajectory
vehicle.setInitialState(1000.0,0.0,0.0,6.5,0.0,-37.86181146377203,0.0,0.0)
vehicle.propagateEntry (60.0*60.0,1.0,0.0)

# Extract and save variable to plot
t_min_us      = vehicle.t_minc
h_km_us       = vehicle.h_kmc
acc_net_g_us  = vehicle.acc_net_g
q_stag_con_us = vehicle.q_stag_con
q_stag_rad_us = vehicle.q_stag_rad

'''

Create fig #1 - altitude history of aerocapture maneuver
'''

fig = plt.figure()
fig.set_size_inches([6.5,6.5])
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.plot(t_min_os , h_km_os, linestyle='solid' , color='xkcd:blue', linewidth=2.0, ↵
         label='Overshoot')
plt.plot(t_min_us , h_km_us, linestyle='solid' , color='xkcd:green', linewidth=2.0, ↵
         label='Undershoot')

plt.xlabel('Time, min', fontsize=14)
plt.ylabel("Altitude, km", fontsize=14)

ax = plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
plt.tick_params(direction='in')
plt.tick_params(axis='x', labelsize=14)
plt.tick_params(axis='y', labelsize=14)
```

(continues on next page)

(continued from previous page)

```

plt.legend(loc='lower right', fontsize=14)

plt.savefig('../plots/titan-systems-altitude.png', bbox_inches='tight')
plt.savefig('../plots/titan-systems-altitude.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/titan-systems-altitude.eps', dpi=300, bbox_inches='tight')

plt.show()

fig = plt.figure()
fig.set_size_inches([6.5, 6.5])
plt.rc('font', family='Times New Roman')
plt.plot(t_min_os , acc_net_g_os, linestyle='solid' , color='xkcd:blue', linewidth=2.0,
         label='Overshoot')
plt.plot(t_min_us , acc_net_g_us, linestyle='solid' , color='xkcd:green', linewidth=2.0,
         label='Undershoot')

plt.xlabel('Time, min', fontsize=14)
plt.ylabel("Deceleration, Earth g", fontsize=14)

ax = plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
plt.tick_params(direction='in')
plt.tick_params(axis='x', labelsize=14)
plt.tick_params(axis='y', labelsize=14)

plt.legend(loc='upper right', fontsize=14)

plt.savefig('../plots/titan-systems-deceleration.png', bbox_inches='tight')
plt.savefig('../plots/titan-systems-deceleration.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/titan-systems-deceleration.eps', dpi=300, bbox_inches='tight')

plt.show()

fig = plt.figure()
fig.set_size_inches([6.5, 6.5])
plt.rc('font', family='Times New Roman')
plt.plot(t_min_os , q_stag_con_os, linestyle='solid' , color='xkcd:blue', linewidth=2.0,
         label='Overshoot convective')
plt.plot(t_min_us , q_stag_con_us, linestyle='solid' , color='xkcd:magenta',
         linewidth=2.0, label='Undershoot convective')

plt.xlabel('Time, min', fontsize=14)
plt.ylabel("Stagnation-point heat rate, "+r'$W/cm^2$', fontsize=14)
ax = plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
plt.tick_params(direction='in')
plt.tick_params(axis='x', labelsize=14)
plt.tick_params(axis='y', labelsize=14)

plt.legend(loc='upper right', fontsize=14)

```

(continues on next page)

(continued from previous page)

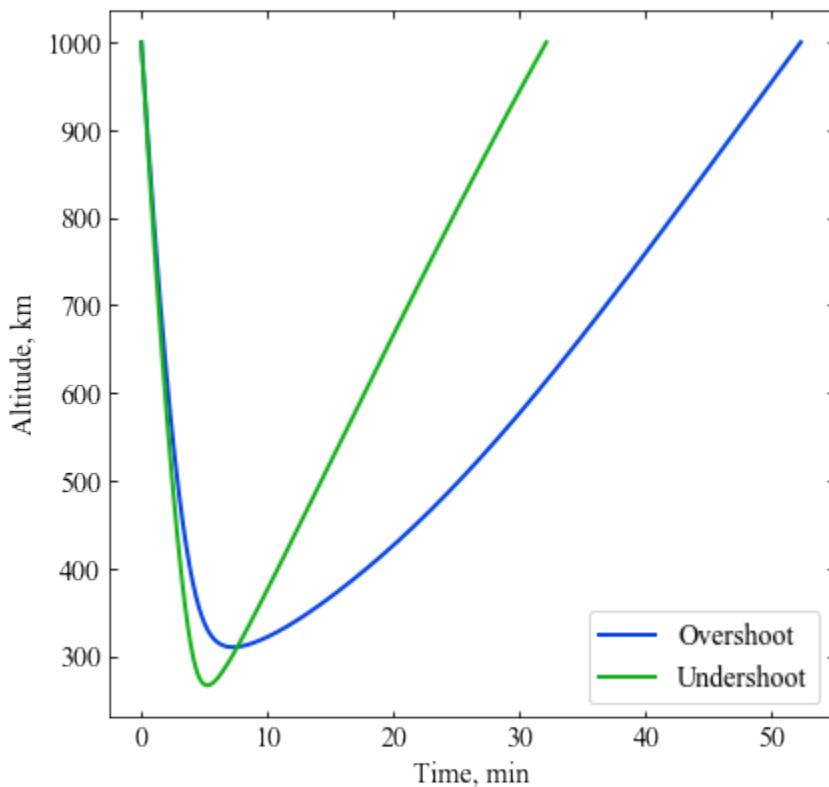
```

plt.savefig('../plots/titan-systems-heating.png',bbox_inches='tight')
plt.savefig('../plots/titan-systems-heating.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/titan-systems-heating.eps', dpi=300,bbox_inches='tight')

plt.show()

The PostScript backend does not support transparency; partially transparent artists
↳ will be rendered opaque.
The PostScript backend does not support transparency; partially transparent artists
↳ will be rendered opaque.
The PostScript backend does not support transparency; partially transparent artists
↳ will be rendered opaque.
The PostScript backend does not support transparency; partially transparent artists
↳ will be rendered opaque.

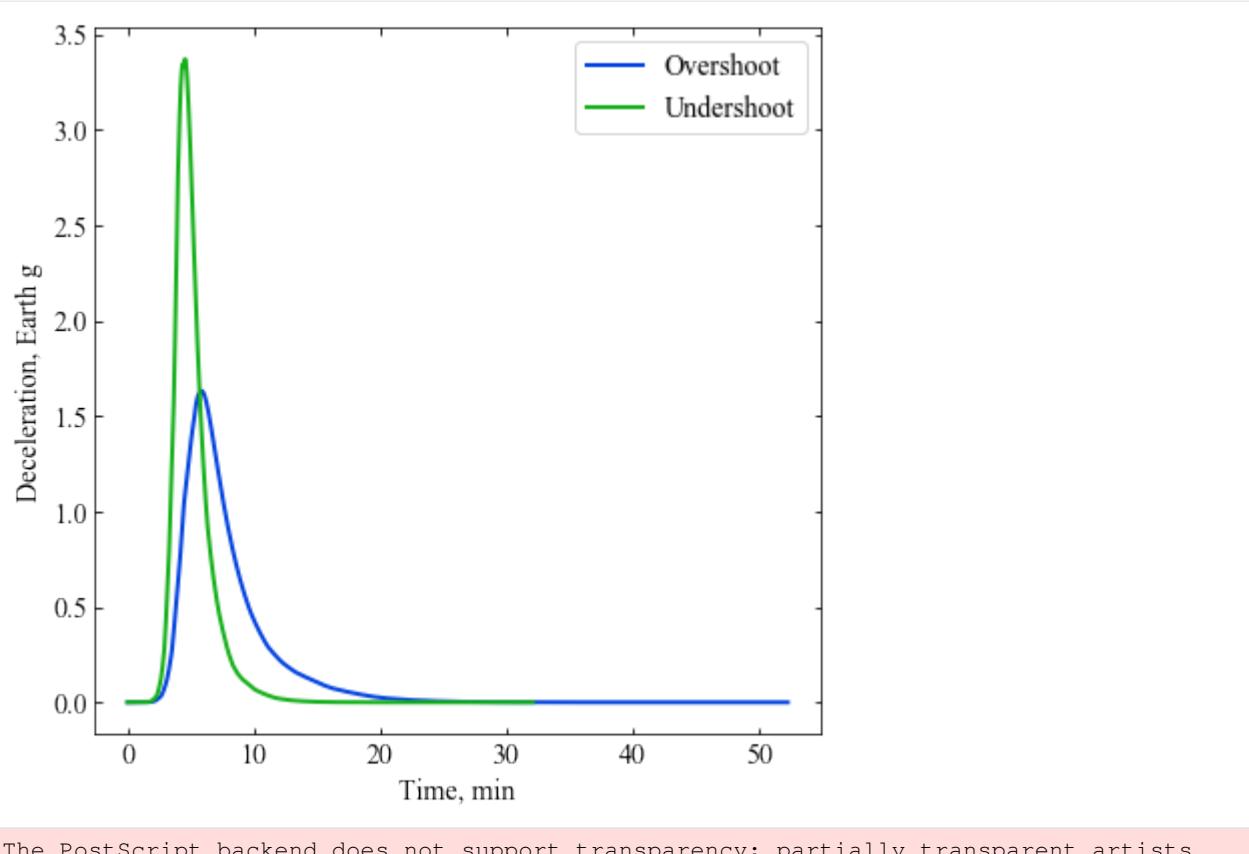
```



```

The PostScript backend does not support transparency; partially transparent artists
↳ will be rendered opaque.
The PostScript backend does not support transparency; partially transparent artists
↳ will be rendered opaque.
The PostScript backend does not support transparency; partially transparent artists
↳ will be rendered opaque.
The PostScript backend does not support transparency; partially transparent artists
↳ will be rendered opaque.

```

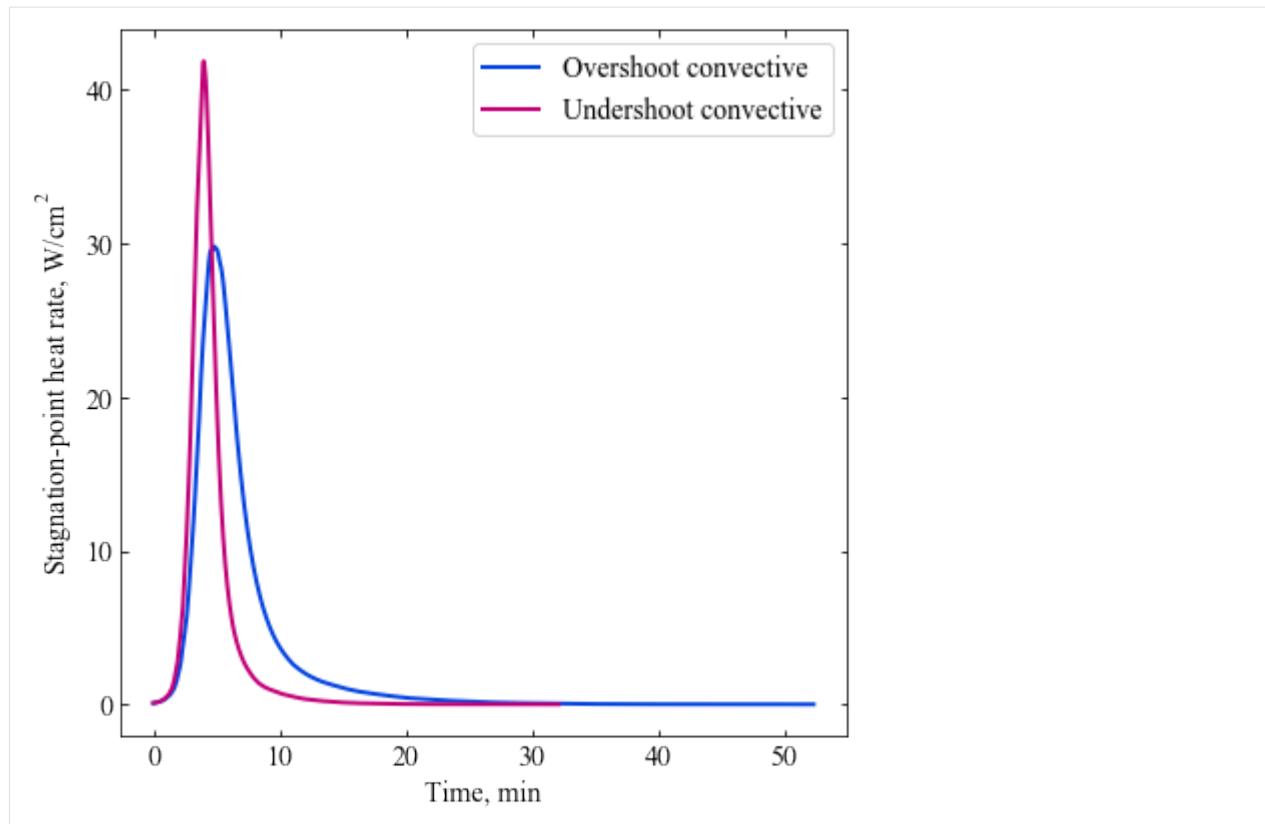


The PostScript backend does not support transparency; partially transparent artists ↴ will be rendered opaque.

The PostScript backend does not support transparency; partially transparent artists ↴ will be rendered opaque.

The PostScript backend does not support transparency; partially transparent artists ↴ will be rendered opaque.

The PostScript backend does not support transparency; partially transparent artists ↴ will be rendered opaque.



4.56 Example - 56 - Venus Entry Tradespace - Carpet Plot

This example will create an EDL carpet plot showing the trade space for high ballistic coefficient entry systems at Venus

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle

[2]: import numpy as np
import matplotlib.pyplot as plt

[3]: # Set up the planet and atmosphere model.
planet=Planet("VENUS")
planet.loadAtmosphereModel('../atmdata/Venus/venus-gram-avg.dat', 0 , 1 , 2 , 3)

[4]: betaRange = np.linspace(100,400,11)
gammaRange = np.linspace(-30,-8,11)

acc_net_g_max_array      = np.zeros((len(betaRange),len(gammaRange)))
q_stag_total_max_array   = np.zeros((len(betaRange),len(gammaRange)))
heatload_max_array        = np.zeros((len(betaRange),len(gammaRange)))
dyn_pres_array            = np.zeros((len(betaRange),len(gammaRange)))

for i, beta in enumerate(betaRange):
    for j, gamma in enumerate(gammaRange):
```

(continues on next page)

(continued from previous page)

```

    vehicle=Vehicle('vehicle1', 300.0, beta, 0.00, 0.25*np.pi*1.00**2, 0.00, 0.20,
→ planet)
    vehicle.setInitialState(180.0,0.0,0.0,11.30,0.0,gamma,0.0,0.0)
    vehicle.setSolverParams(1E-6)
    vehicle.propogateEntry (2400.0,0.1,0.0)

    print("G_MAX: "+str("{:.2f}".format(max(vehicle.acc_net_g)))+", Q_MAX: "+str(
→ {:.2f}".format(max(vehicle.q_stag_total)))+", J_MAX: "+str("{:.2f}".
→ format(max(vehicle.heatload/1.0E3)))+", P_MAX: "+str("{:.2f}".format(max(vehicle.
→ dyn_pres_atm))) )

    acc_net_g_max_array[i,j] = max(vehicle.acc_net_g)
    q_stag_total_max_array[i,j] = max(vehicle.q_stag_total)
    heatload_max_array[i,j] = max(vehicle.heatload/1.0E3)
    dyn_pres_array[i,j] = max(vehicle.dyn_pres_atm)

G_MAX: 272.16, Q_MAX: 3000.63, J_MAX: 10.39, P_MAX: 2.63
G_MAX: 253.65, Q_MAX: 2867.43, J_MAX: 10.69, P_MAX: 2.45
G_MAX: 234.89, Q_MAX: 2728.31, J_MAX: 11.03, P_MAX: 2.27
G_MAX: 215.69, Q_MAX: 2585.18, J_MAX: 11.43, P_MAX: 2.09
G_MAX: 195.63, Q_MAX: 2430.17, J_MAX: 11.91, P_MAX: 1.89
G_MAX: 174.22, Q_MAX: 2261.27, J_MAX: 12.50, P_MAX: 1.69
G_MAX: 151.94, Q_MAX: 2079.58, J_MAX: 13.25, P_MAX: 1.47
G_MAX: 128.85, Q_MAX: 1883.25, J_MAX: 14.24, P_MAX: 1.25
G_MAX: 103.38, Q_MAX: 1655.23, J_MAX: 15.66, P_MAX: 1.00
G_MAX: 74.80, Q_MAX: 1378.07, J_MAX: 18.04, P_MAX: 0.72
G_MAX: 35.61, Q_MAX: 932.10, J_MAX: 24.52, P_MAX: 0.34
G_MAX: 265.99, Q_MAX: 3520.22, J_MAX: 12.28, P_MAX: 3.35
G_MAX: 248.72, Q_MAX: 3365.41, J_MAX: 12.61, P_MAX: 3.13
G_MAX: 230.39, Q_MAX: 3197.74, J_MAX: 13.00, P_MAX: 2.90
G_MAX: 211.15, Q_MAX: 3019.15, J_MAX: 13.45, P_MAX: 2.66
G_MAX: 191.12, Q_MAX: 2830.51, J_MAX: 14.00, P_MAX: 2.40
G_MAX: 170.83, Q_MAX: 2635.39, J_MAX: 14.67, P_MAX: 2.15
G_MAX: 149.45, Q_MAX: 2421.99, J_MAX: 15.51, P_MAX: 1.88
G_MAX: 126.21, Q_MAX: 2182.44, J_MAX: 16.63, P_MAX: 1.59
G_MAX: 101.57, Q_MAX: 1914.77, J_MAX: 18.25, P_MAX: 1.28
G_MAX: 73.11, Q_MAX: 1583.67, J_MAX: 20.98, P_MAX: 0.92
G_MAX: 34.27, Q_MAX: 1061.26, J_MAX: 28.56, P_MAX: 0.43
G_MAX: 261.09, Q_MAX: 4007.68, J_MAX: 14.05, P_MAX: 4.04
G_MAX: 243.61, Q_MAX: 3822.52, J_MAX: 14.42, P_MAX: 3.77
G_MAX: 225.79, Q_MAX: 3628.03, J_MAX: 14.85, P_MAX: 3.50
G_MAX: 207.50, Q_MAX: 3426.40, J_MAX: 15.35, P_MAX: 3.21
G_MAX: 188.32, Q_MAX: 3212.03, J_MAX: 15.95, P_MAX: 2.92
G_MAX: 167.89, Q_MAX: 2979.14, J_MAX: 16.68, P_MAX: 2.60
G_MAX: 146.65, Q_MAX: 2731.82, J_MAX: 17.62, P_MAX: 2.27
G_MAX: 124.52, Q_MAX: 2462.83, J_MAX: 18.86, P_MAX: 1.93
G_MAX: 99.77, Q_MAX: 2150.49, J_MAX: 20.65, P_MAX: 1.55
G_MAX: 71.95, Q_MAX: 1773.14, J_MAX: 23.69, P_MAX: 1.11
G_MAX: 33.21, Q_MAX: 1173.83, J_MAX: 32.28, P_MAX: 0.51
G_MAX: 257.65, Q_MAX: 4478.48, J_MAX: 15.76, P_MAX: 4.74
G_MAX: 240.29, Q_MAX: 4266.81, J_MAX: 16.15, P_MAX: 4.42
G_MAX: 222.38, Q_MAX: 4041.95, J_MAX: 16.61, P_MAX: 4.09
G_MAX: 203.91, Q_MAX: 3806.99, J_MAX: 17.15, P_MAX: 3.75
G_MAX: 185.22, Q_MAX: 3565.21, J_MAX: 17.80, P_MAX: 3.41
G_MAX: 165.72, Q_MAX: 3308.80, J_MAX: 18.60, P_MAX: 3.05
G_MAX: 144.71, Q_MAX: 3026.35, J_MAX: 19.61, P_MAX: 2.66
G_MAX: 122.50, Q_MAX: 2720.55, J_MAX: 20.96, P_MAX: 2.25

```

(continues on next page)

(continued from previous page)

G_MAX: 98.66, Q_MAX: 2375.37, J_MAX: 22.91, P_MAX: 1.81
 G_MAX: 70.90, Q_MAX: 1948.59, J_MAX: 26.23, P_MAX: 1.30
 G_MAX: 32.27, Q_MAX: 1276.52, J_MAX: 35.77, P_MAX: 0.59
 G_MAX: 254.43, Q_MAX: 4919.54, J_MAX: 17.40, P_MAX: 5.42
 G_MAX: 237.40, Q_MAX: 4688.30, J_MAX: 17.83, P_MAX: 5.05
 G_MAX: 219.88, Q_MAX: 4444.41, J_MAX: 18.32, P_MAX: 4.68
 G_MAX: 201.62, Q_MAX: 4184.25, J_MAX: 18.90, P_MAX: 4.29
 G_MAX: 182.64, Q_MAX: 3905.04, J_MAX: 19.59, P_MAX: 3.89
 G_MAX: 163.21, Q_MAX: 3616.24, J_MAX: 20.44, P_MAX: 3.48
 G_MAX: 143.05, Q_MAX: 3309.04, J_MAX: 21.52, P_MAX: 3.05
 G_MAX: 121.02, Q_MAX: 2966.83, J_MAX: 22.97, P_MAX: 2.58
 G_MAX: 97.41, Q_MAX: 2585.49, J_MAX: 25.06, P_MAX: 2.07
 G_MAX: 69.87, Q_MAX: 2112.27, J_MAX: 28.64, P_MAX: 1.49
 G_MAX: 31.52, Q_MAX: 1372.81, J_MAX: 39.06, P_MAX: 0.67
 G_MAX: 252.35, Q_MAX: 5353.22, J_MAX: 19.00, P_MAX: 6.11
 G_MAX: 235.13, Q_MAX: 5089.09, J_MAX: 19.45, P_MAX: 5.69
 G_MAX: 217.49, Q_MAX: 4822.59, J_MAX: 19.97, P_MAX: 5.26
 G_MAX: 199.44, Q_MAX: 4543.29, J_MAX: 20.58, P_MAX: 4.83
 G_MAX: 180.82, Q_MAX: 4241.77, J_MAX: 21.32, P_MAX: 4.38
 G_MAX: 161.27, Q_MAX: 3915.87, J_MAX: 22.22, P_MAX: 3.90
 G_MAX: 141.17, Q_MAX: 3577.07, J_MAX: 23.37, P_MAX: 3.42
 G_MAX: 119.86, Q_MAX: 3206.83, J_MAX: 24.90, P_MAX: 2.90
 G_MAX: 96.13, Q_MAX: 2782.97, J_MAX: 27.13, P_MAX: 2.33
 G_MAX: 69.14, Q_MAX: 2271.19, J_MAX: 30.95, P_MAX: 1.67
 G_MAX: 30.88, Q_MAX: 1462.05, J_MAX: 42.21, P_MAX: 0.75
 G_MAX: 250.31, Q_MAX: 5781.17, J_MAX: 20.57, P_MAX: 6.78
 G_MAX: 233.43, Q_MAX: 5491.90, J_MAX: 21.04, P_MAX: 6.33
 G_MAX: 215.73, Q_MAX: 5191.43, J_MAX: 21.58, P_MAX: 5.85
 G_MAX: 197.61, Q_MAX: 4883.73, J_MAX: 22.23, P_MAX: 5.36
 G_MAX: 179.11, Q_MAX: 4562.31, J_MAX: 23.00, P_MAX: 4.85
 G_MAX: 159.81, Q_MAX: 4213.52, J_MAX: 23.95, P_MAX: 4.33
 G_MAX: 139.55, Q_MAX: 3835.77, J_MAX: 25.16, P_MAX: 3.78
 G_MAX: 118.58, Q_MAX: 3435.06, J_MAX: 26.78, P_MAX: 3.21
 G_MAX: 95.22, Q_MAX: 2975.82, J_MAX: 29.13, P_MAX: 2.58
 G_MAX: 68.52, Q_MAX: 2423.99, J_MAX: 33.18, P_MAX: 1.86
 G_MAX: 30.28, Q_MAX: 1544.69, J_MAX: 45.25, P_MAX: 0.82
 G_MAX: 248.40, Q_MAX: 6186.87, J_MAX: 22.11, P_MAX: 7.45
 G_MAX: 231.77, Q_MAX: 5881.79, J_MAX: 22.59, P_MAX: 6.95
 G_MAX: 214.30, Q_MAX: 5559.95, J_MAX: 23.16, P_MAX: 6.43
 G_MAX: 196.20, Q_MAX: 5215.47, J_MAX: 23.83, P_MAX: 5.89
 G_MAX: 177.53, Q_MAX: 4867.83, J_MAX: 24.64, P_MAX: 5.33
 G_MAX: 158.47, Q_MAX: 4498.81, J_MAX: 25.64, P_MAX: 4.75
 G_MAX: 138.41, Q_MAX: 4093.45, J_MAX: 26.91, P_MAX: 4.15
 G_MAX: 117.26, Q_MAX: 3653.50, J_MAX: 28.60, P_MAX: 3.52
 G_MAX: 94.47, Q_MAX: 3165.29, J_MAX: 31.07, P_MAX: 2.83
 G_MAX: 67.84, Q_MAX: 2568.86, J_MAX: 35.33, P_MAX: 2.04
 G_MAX: 29.70, Q_MAX: 1622.53, J_MAX: 48.18, P_MAX: 0.89
 G_MAX: 246.66, Q_MAX: 6590.07, J_MAX: 23.62, P_MAX: 8.12
 G_MAX: 230.04, Q_MAX: 6264.13, J_MAX: 24.12, P_MAX: 7.57
 G_MAX: 212.88, Q_MAX: 5919.42, J_MAX: 24.72, P_MAX: 7.01
 G_MAX: 195.04, Q_MAX: 5550.66, J_MAX: 25.41, P_MAX: 6.42
 G_MAX: 176.40, Q_MAX: 5166.63, J_MAX: 26.26, P_MAX: 5.80
 G_MAX: 157.18, Q_MAX: 4771.07, J_MAX: 27.29, P_MAX: 5.17
 G_MAX: 137.36, Q_MAX: 4344.64, J_MAX: 28.61, P_MAX: 4.52
 G_MAX: 116.17, Q_MAX: 3868.27, J_MAX: 30.38, P_MAX: 3.82
 G_MAX: 93.67, Q_MAX: 3347.28, J_MAX: 32.97, P_MAX: 3.08
 G_MAX: 67.12, Q_MAX: 2707.20, J_MAX: 37.43, P_MAX: 2.21

(continues on next page)

(continued from previous page)

```
G_MAX: 29.15, Q_MAX: 1697.19, J_MAX: 51.02, P_MAX: 0.96
G_MAX: 245.34, Q_MAX: 6979.82, J_MAX: 25.11, P_MAX: 8.79
G_MAX: 228.57, Q_MAX: 6630.50, J_MAX: 25.63, P_MAX: 8.19
G_MAX: 211.45, Q_MAX: 6263.85, J_MAX: 26.25, P_MAX: 7.57
G_MAX: 193.86, Q_MAX: 5879.23, J_MAX: 26.97, P_MAX: 6.94
G_MAX: 175.45, Q_MAX: 5466.34, J_MAX: 27.84, P_MAX: 6.28
G_MAX: 156.14, Q_MAX: 5036.07, J_MAX: 28.92, P_MAX: 5.59
G_MAX: 136.32, Q_MAX: 4586.26, J_MAX: 30.29, P_MAX: 4.88
G_MAX: 115.36, Q_MAX: 4083.42, J_MAX: 32.13, P_MAX: 4.13
G_MAX: 92.80, Q_MAX: 3522.46, J_MAX: 34.82, P_MAX: 3.32
G_MAX: 66.48, Q_MAX: 2841.40, J_MAX: 39.47, P_MAX: 2.38
G_MAX: 28.67, Q_MAX: 1769.70, J_MAX: 53.79, P_MAX: 1.03
G_MAX: 244.16, Q_MAX: 7371.23, J_MAX: 26.58, P_MAX: 9.45
G_MAX: 227.41, Q_MAX: 6990.31, J_MAX: 27.12, P_MAX: 8.80
G_MAX: 210.19, Q_MAX: 6604.05, J_MAX: 27.76, P_MAX: 8.14
G_MAX: 192.63, Q_MAX: 6199.32, J_MAX: 28.50, P_MAX: 7.46
G_MAX: 174.45, Q_MAX: 5764.68, J_MAX: 29.40, P_MAX: 6.75
G_MAX: 155.36, Q_MAX: 5299.71, J_MAX: 30.52, P_MAX: 6.01
G_MAX: 135.40, Q_MAX: 4819.17, J_MAX: 31.94, P_MAX: 5.24
G_MAX: 114.61, Q_MAX: 4293.05, J_MAX: 33.84, P_MAX: 4.44
G_MAX: 91.97, Q_MAX: 3692.51, J_MAX: 36.63, P_MAX: 3.56
G_MAX: 65.98, Q_MAX: 2974.32, J_MAX: 41.47, P_MAX: 2.55
G_MAX: 28.25, Q_MAX: 1839.89, J_MAX: 56.49, P_MAX: 1.09
```

[5]: `from scipy import interpolate`

```
f1 = interpolate.interp2d(betaRange, gammaRange, np.transpose(acc_net_g_max_array),  
    kind='cubic')  
q1 = interpolate.interp2d(betaRange, gammaRange, np.transpose(q_stag_total_max_array),  
    kind='cubic')  
h1 = interpolate.interp2d(betaRange, gammaRange, np.transpose(heatload_max_array),  
    kind='cubic')  
s1 = interpolate.interp2d(betaRange, gammaRange, np.transpose(dyn_pres_array), kind=  
    'cubic')  
  
x_new = np.linspace(100, 400, 110)  
y_new = np.linspace(-30, -8, 110)  
  
g1_new = np.zeros((len(x_new), len(y_new)))  
q1_new = np.zeros((len(x_new), len(y_new)))  
h1_new = np.zeros((len(x_new), len(y_new)))  
s1_new = np.zeros((len(x_new), len(y_new)))  
  
for i in range(0, len(x_new)):  
    for j in range(0, len(y_new)):  
  
        g1_new[i,j] = f1(x_new[i],y_new[j])  
        q1_new[i,j] = q1(x_new[i],y_new[j])  
        h1_new[i,j] = h1(x_new[i],y_new[j])  
        s1_new[i,j] = s1(x_new[i],y_new[j])  
  
X, Y = np.meshgrid(x_new, y_new)  
  
fig = plt.figure()  
fig.set_size_inches([8.00, 8.00])  
plt.ion()
```

(continues on next page)

(continued from previous page)

```

plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

GCS1 = plt.contour(Y, X, np.transpose(g1_new), levels=np.linspace(40,200,9),  

    linewidths=2.0, colors='black')
plt.clabel(GCS1, inline=1, fontsize=14, colors='black',fmt='%.3d',inline_spacing=1)
GCS1.collections[0].set_label(r'g-load')

QCS1 = plt.contour(Y, X, np.transpose(q1_new), levels=np.linspace(1000,7000,13),  

    linewidths=2.0, colors='red')
plt.clabel(QCS1, inline=1, fontsize=14, colors='red',fmt='%.3d',inline_spacing=1)
QCS1.collections[0].set_label(r'$\dot{q}+' , '+r'$W/cm^2$')

HCS1 = plt.contour(Y, X, np.transpose(h1_new), levels=np.linspace(12,48,10),  

    linewidths=2.0, colors='magenta')
plt.clabel(HCS1, inline=1, fontsize=14, colors='magenta',fmt='%.3d',inline_spacing=1)
HCS1.collections[0].set_label(r'$Q$+', '+r'$kJ/cm^2$')

SCS1 = plt.contour(Y, X, np.transpose(s1_new), linewidths=2.0, colors='blue')
plt.clabel(SCS1, inline=1, fontsize=12, colors='blue',fmt='%.2f',inline_spacing=1)
SCS1.collections[0].set_label(r'$dyn. pres.'+' , '+r'$atm$')

plt.xlabel("Planet relative EFPA "+r'$\gamma_e$'+r', deg', fontsize=16)
plt.ylabel(r'$\beta$, '+r'$kg/m^2$', fontsize=16)
plt.xticks(np.linspace(-30,-8,12),fontsize=16)
plt.yticks(fontsize=16)
ax = plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')

plt.legend(loc='lower left', fontsize=16, framealpha=0.8)

ax.xaxis.set_tick_params(direction='in', which='both')
ax.yaxis.set_tick_params(direction='in', which='both')

ax.xaxis.set_tick_params(width=2, length=8)
ax.yaxis.set_tick_params(width=2, length=8)

ax.xaxis.set_tick_params(width=1, length=6, which='minor')
ax.yaxis.set_tick_params(width=1, length=6, which='minor')

ax.xaxis.grid(which='major', color='k', linestyle='dotted', linewidth=0.0)
ax.xaxis.grid(which='minor', color='k', linestyle='dotted', linewidth=0.0)

ax.yaxis.grid(which='major', color='k', linestyle='dotted', linewidth=0.0)
ax.yaxis.grid(which='minor', color='k', linestyle='dotted', linewidth=0.0)

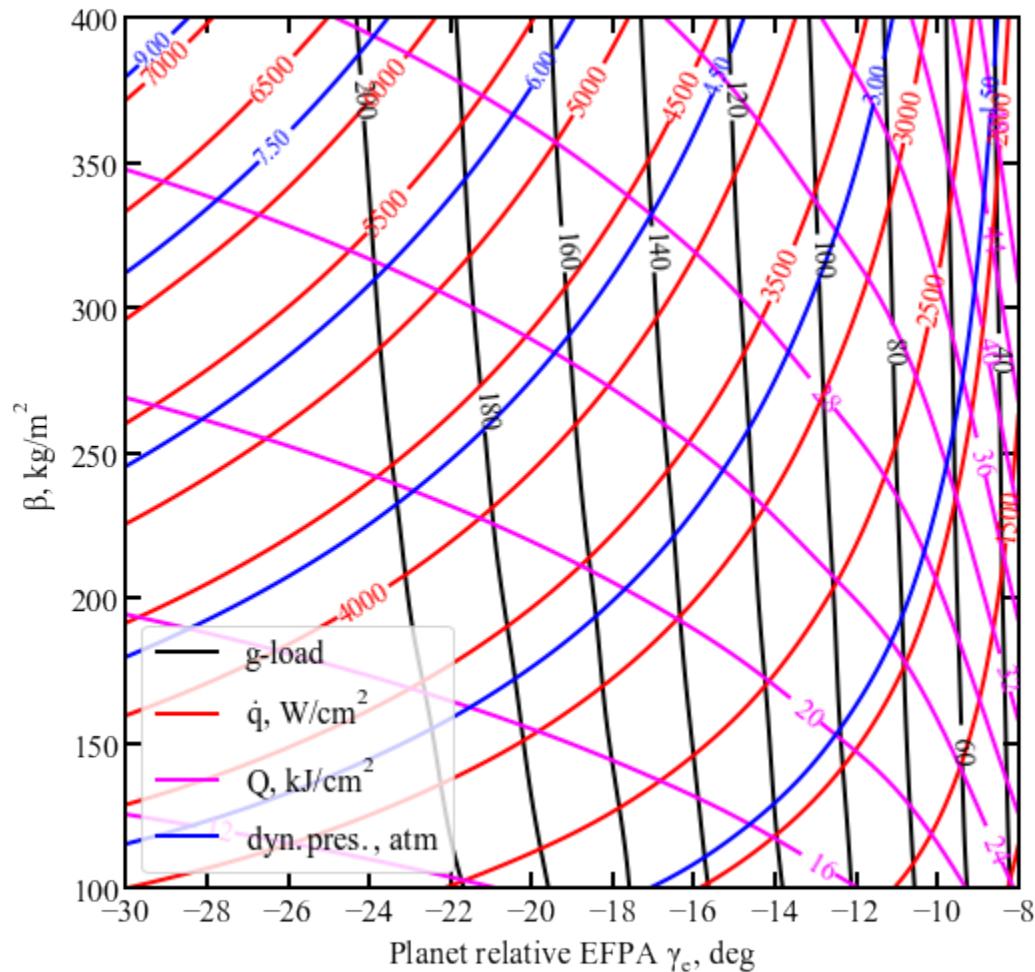
for axis in ['top', 'bottom', 'left', 'right']:
    ax.spines[axis].set_linewidth(2)

plt.savefig('../plots/venus-carpet-plot.png', dpi=300,bbox_inches='tight')
plt.savefig('../plots/venus-carpet-plot.pdf', dpi=300,bbox_inches='tight')
plt.savefig('../plots/venus-carpet-plot.eps', dpi=300,bbox_inches='tight')

```

The PostScript backend does not support transparency; partially transparent artists
→will be rendered opaque.

The PostScript backend does not support transparency; partially transparent artists
→will be rendered opaque.



[]:

4.57 Example - 57 - Dragnofly (Titan) Entry Tradespace - Carpet Plot

This example will create an EDL carpet plot showing the trade space for Dragnofly entry vehicle at Titan. Based on work presented in “Preliminary Interplanetary Mission Design and Navigation for the Dragonfly New Frontiers Mission Concept, AAS 18-416”

```
[1]: from AMAT.planet import Planet
from AMAT.vehicle import Vehicle
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt
```

```
[3]: # Set up the planet and atmosphere model.
planet=Planet("TITAN")
planet.h_skip = 1270.0E3
planet.h_trap = 0.0E3
planet.loadAtmosphereModel('../atmdata/Titan/titan-gram-avg.dat', 0, 1, 2, 3)

[4]: speedRange = np.linspace(7, 9, 11)
gammaRange = np.linspace(-60,-46, 11)

acc_net_g_max_array      = np.zeros((len(speedRange),len(gammaRange)))
q_stag_total_max_array   = np.zeros((len(speedRange),len(gammaRange)))
heatload_max_array       = np.zeros((len(speedRange),len(gammaRange)))
dyn_pres_array           = np.zeros((len(speedRange),len(gammaRange)))

for i, speed in enumerate(speedRange):
    for j, gamma in enumerate(gammaRange):
        vehicle=Vehicle('Dragonfly', 1000.0, 140.0, 0.0, np.pi*3.7**2.0*0.25, 0.0, 0.
←43, planet)
        vehicle.setInitialState(1270.0,0.0,0.0,speed,0.0,gamma,0.0,0.0)
        vehicle.setSolverParams(1E-6)
        vehicle.propogateEntry (120*60,1.0,0.0)

        print("G_MAX: "+str("{:.2f}".format(max(vehicle.acc_net_g)))+", Q_MAX: "+str(
←"{:.2f}".format(max(vehicle.q_stag_total)))+", J_MAX: "+str("{:.2f}".
←format(max(vehicle.heatload/1.0E3)))+", P_MAX: "+str("{:.2f}".format(max(vehicle.
←dyn_pres_atm))) )

        acc_net_g_max_array[i,j] = max(vehicle.acc_net_g)
        q_stag_total_max_array[i,j] = max(vehicle.q_stag_total)
        heatload_max_array[i,j] = max(vehicle.heatload/1.0E3)
        dyn_pres_array[i,j] = max(vehicle.dyn_pres_atm)

G_MAX: 16.17, Q_MAX: 158.40, J_MAX: 7.08, P_MAX: 0.22
G_MAX: 15.53, Q_MAX: 155.48, J_MAX: 7.20, P_MAX: 0.21
G_MAX: 14.86, Q_MAX: 152.30, J_MAX: 7.34, P_MAX: 0.20
G_MAX: 14.15, Q_MAX: 148.93, J_MAX: 7.49, P_MAX: 0.19
G_MAX: 13.40, Q_MAX: 145.24, J_MAX: 7.67, P_MAX: 0.18
G_MAX: 12.60, Q_MAX: 141.26, J_MAX: 7.87, P_MAX: 0.17
G_MAX: 11.75, Q_MAX: 136.89, J_MAX: 8.10, P_MAX: 0.16
G_MAX: 10.84, Q_MAX: 132.13, J_MAX: 8.37, P_MAX: 0.15
G_MAX: 9.86, Q_MAX: 126.80, J_MAX: 8.70, P_MAX: 0.13
G_MAX: 8.80, Q_MAX: 120.85, J_MAX: 9.10, P_MAX: 0.12
G_MAX: 7.65, Q_MAX: 114.04, J_MAX: 9.60, P_MAX: 0.10
G_MAX: 17.05, Q_MAX: 171.82, J_MAX: 7.48, P_MAX: 0.23
G_MAX: 16.38, Q_MAX: 168.64, J_MAX: 7.61, P_MAX: 0.22
G_MAX: 15.68, Q_MAX: 165.21, J_MAX: 7.75, P_MAX: 0.21
G_MAX: 14.92, Q_MAX: 161.53, J_MAX: 7.92, P_MAX: 0.20
G_MAX: 14.12, Q_MAX: 157.52, J_MAX: 8.10, P_MAX: 0.19
G_MAX: 13.28, Q_MAX: 153.18, J_MAX: 8.32, P_MAX: 0.18
G_MAX: 12.38, Q_MAX: 148.41, J_MAX: 8.56, P_MAX: 0.17
G_MAX: 11.41, Q_MAX: 143.22, J_MAX: 8.85, P_MAX: 0.15
G_MAX: 10.38, Q_MAX: 137.40, J_MAX: 9.20, P_MAX: 0.14
G_MAX: 9.26, Q_MAX: 130.92, J_MAX: 9.62, P_MAX: 0.13
G_MAX: 8.03, Q_MAX: 123.51, J_MAX: 10.16, P_MAX: 0.11
G_MAX: 17.96, Q_MAX: 185.95, J_MAX: 7.89, P_MAX: 0.24
G_MAX: 17.26, Q_MAX: 182.53, J_MAX: 8.02, P_MAX: 0.23
```

(continues on next page)

(continued from previous page)

G_MAX: 16.51, Q_MAX: 178.80, J_MAX: 8.18, P_MAX: 0.22
G_MAX: 15.71, Q_MAX: 174.81, J_MAX: 8.35, P_MAX: 0.21
G_MAX: 14.87, Q_MAX: 170.48, J_MAX: 8.55, P_MAX: 0.20
G_MAX: 13.97, Q_MAX: 165.73, J_MAX: 8.78, P_MAX: 0.19
G_MAX: 13.02, Q_MAX: 160.59, J_MAX: 9.04, P_MAX: 0.18
G_MAX: 12.00, Q_MAX: 154.90, J_MAX: 9.34, P_MAX: 0.16
G_MAX: 10.91, Q_MAX: 148.62, J_MAX: 9.71, P_MAX: 0.15
G_MAX: 9.72, Q_MAX: 141.56, J_MAX: 10.16, P_MAX: 0.13
G_MAX: 8.42, Q_MAX: 133.48, J_MAX: 10.73, P_MAX: 0.11
G_MAX: 18.90, Q_MAX: 200.88, J_MAX: 8.31, P_MAX: 0.26
G_MAX: 18.16, Q_MAX: 197.08, J_MAX: 8.45, P_MAX: 0.25
G_MAX: 17.36, Q_MAX: 193.07, J_MAX: 8.62, P_MAX: 0.24
G_MAX: 16.53, Q_MAX: 188.83, J_MAX: 8.80, P_MAX: 0.22
G_MAX: 15.63, Q_MAX: 184.12, J_MAX: 9.01, P_MAX: 0.21
G_MAX: 14.69, Q_MAX: 178.97, J_MAX: 9.25, P_MAX: 0.20
G_MAX: 13.68, Q_MAX: 173.42, J_MAX: 9.52, P_MAX: 0.19
G_MAX: 12.61, Q_MAX: 167.28, J_MAX: 9.85, P_MAX: 0.17
G_MAX: 11.45, Q_MAX: 160.45, J_MAX: 10.24, P_MAX: 0.16
G_MAX: 10.20, Q_MAX: 152.76, J_MAX: 10.71, P_MAX: 0.14
G_MAX: 8.82, Q_MAX: 144.00, J_MAX: 11.32, P_MAX: 0.12
G_MAX: 19.86, Q_MAX: 216.50, J_MAX: 8.74, P_MAX: 0.27
G_MAX: 19.07, Q_MAX: 212.61, J_MAX: 8.89, P_MAX: 0.26
G_MAX: 18.24, Q_MAX: 208.27, J_MAX: 9.07, P_MAX: 0.25
G_MAX: 17.36, Q_MAX: 203.47, J_MAX: 9.26, P_MAX: 0.24
G_MAX: 16.42, Q_MAX: 198.44, J_MAX: 9.48, P_MAX: 0.22
G_MAX: 15.43, Q_MAX: 192.96, J_MAX: 9.73, P_MAX: 0.21
G_MAX: 14.37, Q_MAX: 186.92, J_MAX: 10.02, P_MAX: 0.19
G_MAX: 13.23, Q_MAX: 180.27, J_MAX: 10.37, P_MAX: 0.18
G_MAX: 12.01, Q_MAX: 172.88, J_MAX: 10.78, P_MAX: 0.16
G_MAX: 10.69, Q_MAX: 164.59, J_MAX: 11.28, P_MAX: 0.14
G_MAX: 9.23, Q_MAX: 155.08, J_MAX: 11.93, P_MAX: 0.13
G_MAX: 20.85, Q_MAX: 233.07, J_MAX: 9.18, P_MAX: 0.28
G_MAX: 20.02, Q_MAX: 228.84, J_MAX: 9.34, P_MAX: 0.27
G_MAX: 19.14, Q_MAX: 224.14, J_MAX: 9.53, P_MAX: 0.26
G_MAX: 18.21, Q_MAX: 219.09, J_MAX: 9.73, P_MAX: 0.25
G_MAX: 17.23, Q_MAX: 213.61, J_MAX: 9.96, P_MAX: 0.23
G_MAX: 16.18, Q_MAX: 207.62, J_MAX: 10.23, P_MAX: 0.22
G_MAX: 15.06, Q_MAX: 201.11, J_MAX: 10.54, P_MAX: 0.20
G_MAX: 13.87, Q_MAX: 193.95, J_MAX: 10.90, P_MAX: 0.19
G_MAX: 12.58, Q_MAX: 185.96, J_MAX: 11.33, P_MAX: 0.17
G_MAX: 11.19, Q_MAX: 176.98, J_MAX: 11.86, P_MAX: 0.15
G_MAX: 9.66, Q_MAX: 166.73, J_MAX: 12.55, P_MAX: 0.13
G_MAX: 21.86, Q_MAX: 250.41, J_MAX: 9.64, P_MAX: 0.30
G_MAX: 20.99, Q_MAX: 245.89, J_MAX: 9.81, P_MAX: 0.28
G_MAX: 20.07, Q_MAX: 240.82, J_MAX: 10.00, P_MAX: 0.27
G_MAX: 19.09, Q_MAX: 235.40, J_MAX: 10.21, P_MAX: 0.26
G_MAX: 18.05, Q_MAX: 229.44, J_MAX: 10.46, P_MAX: 0.24
G_MAX: 16.95, Q_MAX: 223.06, J_MAX: 10.74, P_MAX: 0.23
G_MAX: 15.78, Q_MAX: 216.02, J_MAX: 11.06, P_MAX: 0.21
G_MAX: 14.52, Q_MAX: 208.27, J_MAX: 11.44, P_MAX: 0.20
G_MAX: 13.17, Q_MAX: 199.71, J_MAX: 11.90, P_MAX: 0.18
G_MAX: 11.70, Q_MAX: 190.02, J_MAX: 12.46, P_MAX: 0.16
G_MAX: 10.09, Q_MAX: 178.96, J_MAX: 13.19, P_MAX: 0.14
G_MAX: 22.90, Q_MAX: 268.79, J_MAX: 10.10, P_MAX: 0.31
G_MAX: 21.98, Q_MAX: 263.74, J_MAX: 10.28, P_MAX: 0.30
G_MAX: 21.02, Q_MAX: 258.36, J_MAX: 10.48, P_MAX: 0.28
G_MAX: 19.98, Q_MAX: 252.50, J_MAX: 10.71, P_MAX: 0.27

(continues on next page)

(continued from previous page)

```
G_MAX: 18.90, Q_MAX: 246.07, J_MAX: 10.97, P_MAX: 0.26
G_MAX: 17.74, Q_MAX: 239.21, J_MAX: 11.26, P_MAX: 0.24
G_MAX: 16.51, Q_MAX: 231.61, J_MAX: 11.60, P_MAX: 0.22
G_MAX: 15.19, Q_MAX: 223.37, J_MAX: 12.00, P_MAX: 0.21
G_MAX: 13.77, Q_MAX: 214.09, J_MAX: 12.48, P_MAX: 0.19
G_MAX: 12.23, Q_MAX: 203.71, J_MAX: 13.08, P_MAX: 0.17
G_MAX: 10.53, Q_MAX: 191.78, J_MAX: 13.84, P_MAX: 0.14
G_MAX: 23.96, Q_MAX: 287.81, J_MAX: 10.58, P_MAX: 0.32
G_MAX: 23.00, Q_MAX: 282.40, J_MAX: 10.77, P_MAX: 0.31
G_MAX: 21.99, Q_MAX: 276.65, J_MAX: 10.98, P_MAX: 0.30
G_MAX: 20.90, Q_MAX: 270.37, J_MAX: 11.22, P_MAX: 0.28
G_MAX: 19.77, Q_MAX: 263.58, J_MAX: 11.49, P_MAX: 0.27
G_MAX: 18.55, Q_MAX: 256.16, J_MAX: 11.79, P_MAX: 0.25
G_MAX: 17.27, Q_MAX: 248.04, J_MAX: 12.15, P_MAX: 0.23
G_MAX: 15.87, Q_MAX: 239.12, J_MAX: 12.57, P_MAX: 0.22
G_MAX: 14.39, Q_MAX: 229.22, J_MAX: 13.08, P_MAX: 0.19
G_MAX: 12.77, Q_MAX: 218.02, J_MAX: 13.70, P_MAX: 0.17
G_MAX: 10.99, Q_MAX: 205.21, J_MAX: 14.51, P_MAX: 0.15
G_MAX: 25.02, Q_MAX: 307.93, J_MAX: 11.07, P_MAX: 0.34
G_MAX: 24.04, Q_MAX: 302.11, J_MAX: 11.27, P_MAX: 0.33
G_MAX: 22.97, Q_MAX: 295.87, J_MAX: 11.49, P_MAX: 0.31
G_MAX: 21.85, Q_MAX: 289.11, J_MAX: 11.74, P_MAX: 0.30
G_MAX: 20.66, Q_MAX: 281.89, J_MAX: 12.02, P_MAX: 0.28
G_MAX: 19.38, Q_MAX: 273.95, J_MAX: 12.34, P_MAX: 0.26
G_MAX: 18.03, Q_MAX: 265.17, J_MAX: 12.72, P_MAX: 0.24
G_MAX: 16.58, Q_MAX: 255.62, J_MAX: 13.16, P_MAX: 0.22
G_MAX: 15.02, Q_MAX: 245.01, J_MAX: 13.69, P_MAX: 0.20
G_MAX: 13.32, Q_MAX: 233.04, J_MAX: 14.35, P_MAX: 0.18
G_MAX: 11.46, Q_MAX: 219.25, J_MAX: 15.20, P_MAX: 0.16
G_MAX: 26.15, Q_MAX: 328.78, J_MAX: 11.57, P_MAX: 0.35
G_MAX: 25.10, Q_MAX: 322.67, J_MAX: 11.77, P_MAX: 0.34
G_MAX: 23.98, Q_MAX: 315.87, J_MAX: 12.01, P_MAX: 0.32
G_MAX: 22.82, Q_MAX: 308.77, J_MAX: 12.27, P_MAX: 0.31
G_MAX: 21.57, Q_MAX: 300.83, J_MAX: 12.56, P_MAX: 0.29
G_MAX: 20.23, Q_MAX: 292.45, J_MAX: 12.90, P_MAX: 0.27
G_MAX: 18.82, Q_MAX: 283.15, J_MAX: 13.29, P_MAX: 0.25
G_MAX: 17.30, Q_MAX: 272.93, J_MAX: 13.76, P_MAX: 0.23
G_MAX: 15.66, Q_MAX: 261.53, J_MAX: 14.31, P_MAX: 0.21
G_MAX: 13.89, Q_MAX: 248.70, J_MAX: 15.00, P_MAX: 0.19
G_MAX: 11.93, Q_MAX: 233.97, J_MAX: 15.90, P_MAX: 0.16
```

```
[5]: from scipy import interpolate
```

```
[6]: f1 = interpolate.interp2d(speedRange, gammaRange, np.transpose(acc_net_g_max_array),  
    kind='cubic')  
q1 = interpolate.interp2d(speedRange, gammaRange, np.transpose(q_stag_total_max_  
    array), kind='cubic')  
h1 = interpolate.interp2d(speedRange, gammaRange, np.transpose(heatload_max_array),  
    kind='cubic')  
s1 = interpolate.interp2d(speedRange, gammaRange, np.transpose(dyn_pres_array), kind=  
    'cubic')  
  
x_new = np.linspace(7, 9, 110)  
y_new = np.linspace(-60, -46, 110)
```

(continues on next page)

(continued from previous page)

```

g1_new = np.zeros((len(x_new), len(y_new)))
q1_new = np.zeros((len(x_new), len(y_new)))
h1_new = np.zeros((len(x_new), len(y_new)))
s1_new = np.zeros((len(x_new), len(y_new)))

for i in range(0, len(x_new)):
    for j in range(0, len(y_new)):

        g1_new[i,j] = f1(x_new[i],y_new[j])
        q1_new[i,j] = q1(x_new[i],y_new[j])
        h1_new[i,j] = h1(x_new[i],y_new[j])
        s1_new[i,j] = s1(x_new[i],y_new[j])

X, Y = np.meshgrid(x_new, y_new)

fig = plt.figure()
fig.set_size_inches([8.00,8.00])
plt.ion()
plt.rc('font',family='Times New Roman')
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)

GCS1 = plt.contour(Y, X, np.transpose(g1_new), levels=15, linewidths=2.0, colors=
    ↪'black')
plt.clabel(GCS1, inline=1, fontsize=14, colors='black',fmt='%.3d',inline_spacing=1)
GCS1.collections[0].set_label(r'g-load')

QCS1 = plt.contour(Y, X, np.transpose(q1_new), levels=15, linewidths=2.0, colors='red
    ↪')
plt.clabel(QCS1, inline=1, fontsize=14, colors='red',fmt='%.3d',inline_spacing=1)
QCS1.collections[0].set_label(r'$\dot{q}+$', '+r'$W/cm^2$')

HCS1 = plt.contour(Y, X, np.transpose(h1_new), levels=15, linewidths=2.0, colors=
    ↪'magenta')
plt.clabel(HCS1, inline=1, fontsize=14, colors='magenta',fmt='%.3d',inline_spacing=1)
HCS1.collections[0].set_label(r'$Q$+', '+r'$kJ/cm^2$')

plt.xlabel("Planet relative EFPA "+r'$\gamma_e$'+r', deg', fontsize=16)
plt.ylabel(r'Entry speed, km/s', fontsize=16)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
ax = plt.gca()
ax.tick_params(direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')

plt.legend(loc='lower left', fontsize=16, framealpha=0.8)

ax.xaxis.set_tick_params(direction='in', which='both')
ax.yaxis.set_tick_params(direction='in', which='both')

ax.xaxis.set_tick_params(width=2, length=8)
ax.yaxis.set_tick_params(width=2, length=8)

ax.xaxis.set_tick_params(width=1, length=6, which='minor')
ax.yaxis.set_tick_params(width=1, length=6, which='minor')

```

(continues on next page)

(continued from previous page)

```

ax.xaxis.grid(which='major', color='k', linestyle='dotted', linewidth=0.0)
ax.xaxis.grid(which='minor', color='k', linestyle='dotted', linewidth=0.0)

ax.yaxis.grid(which='major', color='k', linestyle='dotted', linewidth=0.0)
ax.yaxis.grid(which='minor', color='k', linestyle='dotted', linewidth=0.0)

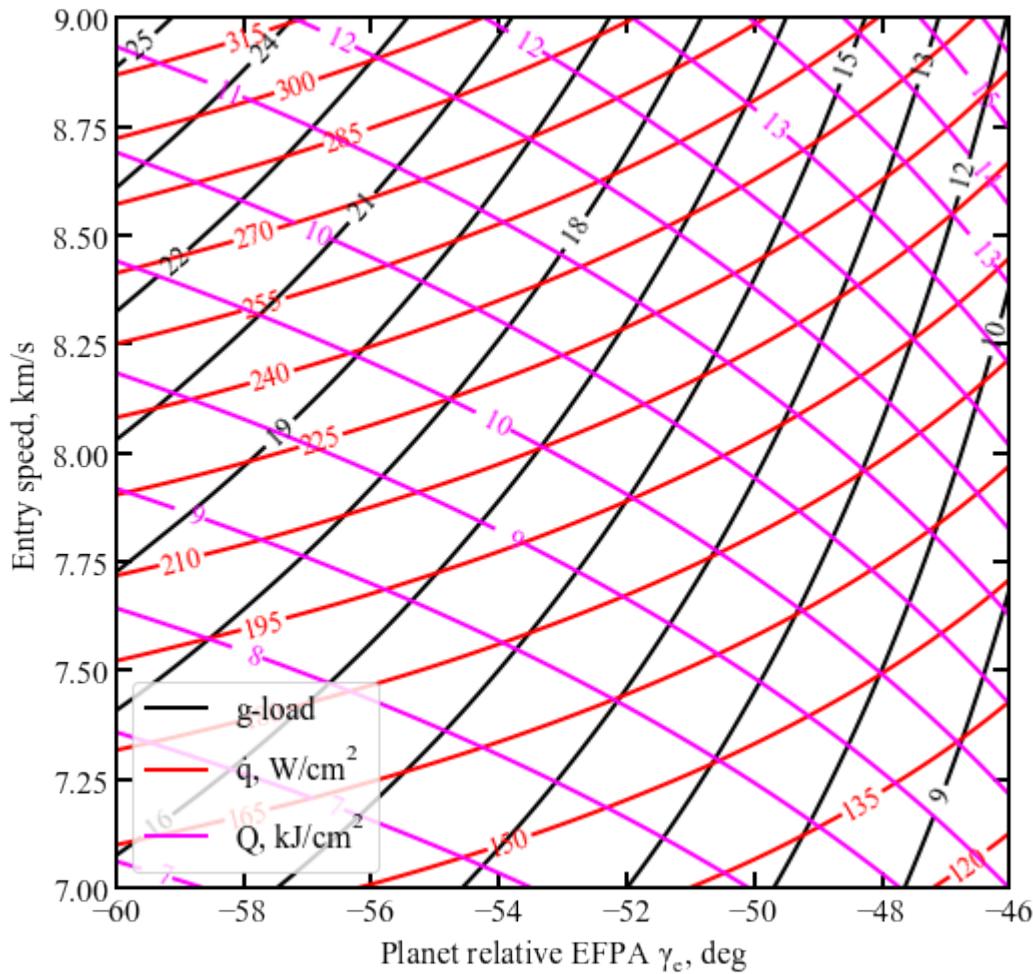
for axis in ['top', 'bottom', 'left', 'right']:
    ax.spines[axis].set_linewidth(2)

plt.savefig('../plots/dragnofly-sweep.png', dpi=300, bbox_inches='tight')
plt.savefig('../plots/dragnofly-sweep.pdf', dpi=300, bbox_inches='tight')
plt.savefig('../plots/dragnofly-sweep.eps', dpi=300, bbox_inches='tight')

```

The PostScript backend does not support transparency; partially transparent artists
 ↪will be rendered opaque.

The PostScript backend does not support transparency; partially transparent artists
 ↪will be rendered opaque.



Compare the figure to Fig. 11 presented in the paper. The heat rate and heat loads are somewhat off (by a factor of about 2 in some cases) because of the simple models heating correlations used. This is an area which could be improved in future versions of AMAT.

API REFERENCE

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

AMAT provides mission analysis capability for aerocapture and Entry, Descent, and Landing (EDL) concepts for planetary science missions.

Copyright (C) 2021 Athul Pradeepkumar Girija.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License Version 3 as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

class AMAT.planet.Planet (planetID)

The Planet class is used to store planetary constants, load atmospheric data from look-up tables, and define non-dimensional parameters used in the simulations.

ID

String identifier of planet object

Type str**RP**

Mean equatorial radius of the target planet in meters

Type float**OMEGA**

Mean angular velocity of rotation of the planet about its axis of rotation in rad/s

Type float**GM**

Standard gravitational parameter of the planet in m³/s²

Type float**rho0**

Reference atmospheric density at the surface of the target planet in kg/m³

Type float**CPCV**

Specific heat ratio CP/CV at the surface of the planet

Type float**J2**

zonal harmonic coefficient J2

Type float
J3
zonal harmonic coefficient J3
Type float
h_thres
Atmospheric model cutoff altitude in meters, density is set to 0, if altitude exceeds h_thres
Type float
h_skip
If vehicle altitude exceeds this value, trajectory is cut off and vehicle is assumed to skip off the atmosphere
Type float
h_trap
If vehicle altitude falls below this value, trajectory is cut off and vehicle is assumed to hit the surface
Type float
h_low
If terminal altitude is below this value vehicle is assumed to be trapped in the atmosphere.
Type float
vref
Reference velocity for non-dimensionalization of entry equations
Type float
tau
Reference timescale used to non-dimensionalize time, angular rates
Type float
OMEGABAR
Reference non-dimensional angular rate of planet's rotation
Type float
EARTHG
Reference value of acceleration due to Earth's gravity
Type float
ATM
Array containing the data loaded from atmospheric lookup file
Type numpy.ndarray
ATM_height
Array containing height values from atm. look up dat file
Type numpy.ndarray
ATM_temp
Array containing temperature values from atm. look up dat file
Type numpy.ndarray
ATM_pressure
Array containing pressure values from atm. look up dat file
Type numpy.ndarray

ATM_density

Array containing density values from atm. look up dat file

Type numpy.ndarray

ATM_sonic

Array containing computed sonic speed values

Type numpy.ndarray

temp_int

Function which interpolates temperature as function of height

Type scipy.interpolate.interp1d

pressure_int

Function which interpolates pressure as function of height

Type scipy.interpolate.interp1d

density_int

Function which interpolates density as function of height

Type scipy.interpolate.interp1d

sonic_int

Function which interpolates sonic speed as function of height

Type scipy.interpolate.interp1d

avectorized(h)

Returns atmospheric sonic speed, vector at altitudes array h[:] in meters

Parameters **h** (numpy.ndarray) – altitude h[:] at which sonic speed is desired

Returns **ans** – returns the sonic speed at altitudes h[:], K

Return type numpy.ndarray

checkAtmProfiles (h0=0.0, dh=1000.0)

Function to check the loaded atmospheric profile data. Plots temperature, pressure, density and sonic speed as function of altitude.

Parameters

- **h0** (float, optional) – lower limit of altitude, defaults to 0.0
- **dh** (float, optional) – height interval

Returns

- *A plot showing the atmospheric profiles loaded*
- *from the lookup tables*

computeH(r)

Returns altitude h, as a function of radial distance r, METERS

Parameters **r** (float) – radial distance in meters

Returns **h** – h = r - RP

Return type float

computeR(h)

Returns radial distance r, as a function of altitude h, METERS

Parameters **h** (float) – altitude in meters

Returns **r** – radial distance $r=RP+h$

Return type float

density(*h*)

Returns atmospheric density, scalar value, at altitude *h* (in meters)

Parameters **h** (*float*) – altitude in meters

Returns **ans** – atmospheric density at height *h*

Return type float

densityvectorized(*h*)

Returns atmospheric density, vector at altitudes array $h[:]$ in meters

Parameters **h** (*numpy.ndarray*) – altitude $h[:]$ at which atmospheric density is desired

Returns **ans** – returns the atmospheric density at altitudes $h[:, K]$

Return type numpy.ndarray

dimensionalize(*tbar*, *rbar*, *theta*, *phi*, *vbar*, *psi*, *gamma*, *drangebar*)

Computes dimensional trajectory state variables from non-dimensional trajectory state variables

Parameters

- **rbar** (*float*) – non-dimensional radial distance in meters
- **theta** (*float*) – longitude theta(RADIANS), theta in [-PI,PI]
- **phi** (*float*) – latitude phi (RADIANS), phi in (-PI/2, PI/2)
- **vbar** (*float*) – non-dimensional planet-relative speed, m/s
- **psi** (*float*) – heading angle, radians
- **gamma** (*float*) – planet-relative flight-path angle, radians
- **drangebar** (*float*) – non-dimensional downrange distance measured from entry-interface

Returns

- **r** (*float*) – radial distance in meters
- **theta** (*float*) – longitude theta(RADIANS), theta in [-PI,PI]
- **phi** (*float*) – latitude phi (RADIANS), phi in (-PI/2, PI/2)
- **v** (*float*) – planet-relative speed, m/s
- **psi** (*float*) – heading angle, radians
- **gamma** (*float*) – planet-relative flight-path angle, radians
- **drange** (*float*) – downrange distance measured from entry-interface

loadAtmosphereModel(*datfile*, *heightCol*, *tempCol*, *presCol*, *densCol*, *intType='cubic'*,
heightInKmFlag=False)

Load atmospheric model from a look up table with height, temperature, pressure, and density

Parameters

- **datfile** (*str*) – file containing atmospheric lookup table
- **heightCol** (*int*) – column number of height values, assumes unit = meters (first column = 0, second column = 1, etc.)

- **presCol** (*int*) – column number of pressure values, assumes unit = Pascals (first column = 0, second column = 1, etc.)
- **densCol** (*int*) – column number of density values, assumes unit = kg/m³ (first column = 0, second column = 1, etc.)
- **intType** (*str, optional*) – interpolation type: ‘linear’, ‘quadratic’ or ‘cubic’ defaults to ‘cubic’
- **heightInKmFlag** (*bool, optional*) – optional, set this to True if heightCol has units of km, False by default

loadAtmosphereModel5 (*ATM_height, ATM_density_low, ATM_density_avg, ATM_density_high, ATM_density_pert, sigmaValue, NPOS, i*)

Read and create density_int for a single entry from a list of perturbed monte carlo density profiles.

Parameters

- **ATM_height** (*numpy.ndarray*) – height array, m
- **ATM_density_low** (*numpy.ndarray*) – low density array, kg/m³
- **ATM_density_avg** (*numpy.ndarray*) – avg. density array, kg/m³
- **ATM_density_high** (*numpy.ndarray*) – high density array, kg/m³
- **ATM_density_pert** (*numpy.ndarray*) – 1 sigma mean deviation from avg
- **sigmaValue** (*float*) – mean desnity profile sigma deviation value (intended as input from a normal distribution with mean=0, sigma=1)
- **NPOS** (*int*) – NPOS value from GRAM model equals the number of positions (altitude) for which density value is available in look up table.

Returns **density_int** – density interpolation function

Return type `scipy.interpolate.interpolate.interp1d`

loadAtmosphereModel6 (*ATM_height, ATM_density_low, ATM_density_avg, ATM_density_high, sigmaValue, NPOS, i*)

Read and create mean density profile for a single entry from a list of perturbed monte carlo density profiles.

Parameters

- **ATM_height** (*numpy.ndarray*) – height array, m
- **ATM_density_low** (*numpy.ndarray*) – low density array, kg/m³
- **ATM_density_avg** (*numpy.ndarray*) – avg. density array, kg/m³
- **ATM_density_high** (*numpy.ndarray*) – high density array, kg/m³
- **sigmaValue** (*float*) – mean desnity profile sigma deviation value (intended as input from a normal distribution with mean=0, sigma=1)
- **NPOS** (*int*) – NPOS value from GRAM model equals the number of positions (altitude) for which density value is available in look up table.

Returns **density_int** – density interpolation function

Return type `scipy.interpolate.interpolate.interp1d`

loadMonteCarloDensityFile2 (*atmfile, heightCol, densLowCol, densAvgCol, densHighCol, densTotalCol, heightInKmFlag=False*)

Loads a Monte Carlo density look up table from GRAM-Model output

Parameters

- **atmfile** (*str*) – filename, contains mean density profile data
- **heightCol** (*int*) – column number of height values, assumes unit = meters (first column = 0, second column = 1, etc.)
- **densLowCol** (*int*) – column number of the low mean density
- **densAvgCol** (*int*) – column number of the average mean density
- **densHigCol** (*int*) – column number of the high mean density
- **densTotalCol** (*int*) – column number of the total (=mean + perturb.) density
- **heightInKmFlag** (*bool, optional*) – optional, set this to True if heightCol has units of km, False by default

Returns

- **ATM_height** (*numpy.ndarray*) – height array, m
- **ATM_density_low** (*numpy.ndarray*) – low density array, kg/m³
- **ATM_density_avg** (*numpy.ndarray*) – avg. density array, kg/m³
- **ATM_density_high** (*numpy.ndarray*) – high density array, kg/m³
- **ATM_density_pert** (*numpy.ndarray*) – 1 sigma mean deviation from avg

loadMonteCarloDensityFile3 (*atmfile, heightCol, densAvgCol, densSD_percCol, densTotalCol, heightInKmFlag=False*)

Loads a Monte Carlo density look up table from GRAM-Model output, Utility function for EARTH-GRAM atmospheric data

Parameters

- **atmfile** (*str*) – filename, contains mean density profile data
- **heightCol** (*int*) – column number of height values, assumes unit = meters (first column = 0, second column = 1, etc.)
- **densAvgCol** (*int*) – column number of the average mean density
- **densSD_percCol** (*int*) – column number of mean density one sigma SD
- **densTotalCol** (*int*) – column number of the total (=mean + perturb.) density
- **heightInKmFlag** (*bool, optional*) – optional, set this to True if heightCol has units of km, False by default

Returns

- **ATM_height** (*numpy.ndarray*) – height array, m
- **ATM_density_low** (*numpy.ndarray*) – low density array, kg/m³
- **ATM_density_avg** (*numpy.ndarray*) – avg. density array, kg/m³
- **ATM_density_high** (*numpy.ndarray*) – high density array, kg/m³
- **ATM_density_pert** (*numpy.ndarray*) – 1 sigma mean deviation from avg

nSigmaFunc (*x*)

Utility function. Returns 1 if *x*<0, 0.0 otherwise

Parameters **x** (*float*) – input x

Returns **ans** – 1 if *x*<0, 0.0 otherwise

Return type float

nonDimState (*r, theta, phi, v, psi, gamma, drange*)

Computes non-dimensional trajectory state variables from dimensional trajectory state variables

Parameters

- ***r*** (*float*) – radial distance in meters
- ***theta*** (*float*) – longitude theta(RADIANS), theta in [-PI,PI]
- ***phi*** (*float*) – latitude phi (RADIANS), phi in (-PI/2, PI/2)
- ***v*** (*float*) – planet-relative speed, m/s
- ***psi*** (*float*) – heading angle, radians
- ***gamma*** (*float*) – planet-relative flight-path angle, radians
- ***drange*** (*float*) – downrange distance measured from entry-interface

Returns

- ***rbar*** (*float*) – non-dimensional radial distance in meters
- ***theta*** (*float*) – longitude theta(RADIANS), theta in [-PI,PI]
- ***phi*** (*float*) – latitude phi (RADIANS), phi in (-PI/2, PI/2)
- ***vbar*** (*float*) – non-dimensional planet-relative speed, m/s
- ***psi*** (*float*) – heading angle, radians
- ***gamma*** (*float*) – planet-relative flight-path angle, radians
- ***drangebar*** (*float*) – non-dimensional downrange distance measured from entry-interface

pSigmaFunc (*x*)

Utility function. Returns 1 if *x*>=0, 0.0 otherwise

Parameters ***x*** (*float*) – input *x***Returns** ***ans*** – 1 if *x*>=0, 0.0 otherwise**Return type** float**pressurevectorized** (*r*)

Returns atmospheric pressure, vector at radial distance array *r*[:] in meters

Parameters ***r*** (*numpy.ndarray*) – radial distance *r*[:] at which pressure is desired**Returns** ***ans*** – returns the atmospheric pressure at radial distance *r*[:]**Return type** numpy.ndarray**presvectorized** (*h*)

Returns atmospheric pressure, vector at altitudes array *h*[:] in meters

Parameters ***h*** (*numpy.ndarray*) – altitude *h*[:] at which atmospheric pressure is desired**Returns** ***ans*** – returns the atmospheric pressure at altitudes *h*[:], K**Return type** numpy.ndarray**rbar** (*r*)

Returns non-dimensional rbar=r/RP

Parameters ***r*** (*float*) – radial distance in meters**Returns** ***ans*** – non-dimensional rbar**Return type** float

rho (*r, theta, phi*)

Returns atmospheric density rho, scalar, as a function of radial distance from the target planet center *r* as well as longitude *theta* and latitude *phi*

Parameters

- **r** (*float*) – radial distance *r* measured from the planet center
- **theta** (*float*) – longitude *theta*(RADIANS), *theta* in [-PI,PI]
- **phi** (*float*) – latitude *phi* (RADIANS), *phi* in (-PI/2, PI/2)

Returns **ans** – returns the atmospheric density at (*r,theta,phi*)

Return type numpy.ndarray

rho2 (*rbar, theta, phi*)

Returns atmospheric density rho, scalar, as a function of non-dimensional radial distance *rbar*, longitude *theta*, and latitude *phi*

Parameters

- **rbar** (*float*) – nondimensional radial distance *rbar* measured from the planet center
- **theta** (*float*) – longitude *theta*(RADIANS), *theta* in [-PI,PI]
- **phi** (*float*) – latitude *phi* (RADIANS), *phi* in (-PI/2, PI/2)

Returns **ans** – returns the atmospheric density at (*rbar,theta,phi*)

Return type float

rhobar (*rbar, theta, phi*)

Returns non-dimensional density *rhobar* = *rho* / *rho0* as a function of non-dimensional radial distance *rbar*, longitude *theta*, and latitude *phi*

Parameters

- **rbar** (*float*) – nondimensional radial distance *rbar* measured from the planet center
- **theta** (*float*) – longitude *theta*(RADIANS), *theta* in [-PI,PI]
- **phi** (*float*) – latitude *phi* (RADIANS), *phi* in (-PI/2, PI/2)

Returns **ans** – non-dimensional density at (*rbar,theta,phi*)

Return type float

rhovectorized (*r*)

Returns atmospheric density, vector at radial distance array *r*[:] in meters

Parameters **r** (*numpy.ndarray*) – radial distance *r*[:] at which density is desired

Returns **ans** – returns the atmospheric density at radial distance *r*[:]

Return type numpy.ndarray

scaleHeight (*h, density_int*)

Returns the scale height as a function of altitude for given density profile

Parameters

- **h** (*float*) – altitude at which scale height is desired
- **density_int** (*scipy.interpolate.interpolate.interp1d*) – density interpolation function
- ---

Returns `ans` – scale height, meters

Return type float

sonicvectorized(r)

Returns atmospheric sonic speed, vector at radial distance array `r[:]` in meters

Parameters `r` (`numpy.ndarray`) – radial distance `r[:]` at which sonic speed is desired

Returns `ans` – returns the atmospheric speed at radial distance `r[:]`

Return type `numpy.ndarray`

temperaturevectorized(r)

Returns atmospheric temperature, vector at radial distance array `r[:]` in meters

Parameters `r` (`numpy.ndarray`) – radial distance `r[:]` at which temperature is desired

Returns `ans` – returns the atmospheric temperature at radial distance `r[:]`

Return type `numpy.ndarray`

tempvectorized(h)

Returns atmospheric temperature, vector at altitudes array `h[:]` in meters

Parameters `h` (`numpy.ndarray`) – altitude `h[:]` at which atmospheric temperature is desired

Returns `ans` – returns the atmospheric temperature at altitudes `h[:]`, K

Return type `numpy.ndarray`

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

AMAT provides mission analysis capability for aerocapture and Entry, Descent, and Landing (EDL) concepts for planetary science missions.

Copyright (C) 2021 Athul Pradeepkumar Girija.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License Version 3 as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

class AMAT.vehicle.Vehicle(`vehicleID, mass, beta, LD, A, alpha, RN, planetObj`)

The Vehicle class is used to define vehicle parameters, such as mass, aerodynamics, and aeroheating relations.

vehicleID

identifier string for vehicle object

Type str

mass

vehicle mass, kg

Type float

beta

vehicle ballistic coefficient, kg/m²

Type float

LD

vehicle lift-to-drag ratio

Type float

A

vehicle reference aerodynamic area, m²

Type float

alpha

vehicle angle-of-attack, currently not implemented, rad

Type float

RN

vehicle nose-radius, m

Type float

planetObj

planet object associated with the vehicle, indicates the planetary atmosphere in which the vehicle will operate in

Type *planet.Planet*

Abar

vehicle non-dimensional reference area

Type float

mbar

vehicle non-dimensional mass

Type float

CD

vehicle drag coefficient

Type float

CL

vehicle lift coefficient

Type float

h0_km

initial vehicle altitude at atmospheric interface / other start point, meters

Type float

theta0_deg

initial vehicle longitude at atmospheric interface / other start point, degrees

Type float

phi0_deg

initial vehicle latitude at atmospheric interface / other start point, degrees

Type float

v0_kms

initial vehicle speed (planet-relative) at atmospheric interface / other start point, km/s

Type float

psi0_deg

initial vehicle heading at atmospheric interface / other start point, degrees

Type float

gamma0_deg
initial vehicle flight-path angle at atmospheric interface / other start point, degrees
Type float

drange0_km
initial vehicle downrange at atmospheric interface / other start point, km
Type float

heatLoad0
initial vehicle heatload at atmospheric interface / other start point, J/cm2
Type float

tol
solver tolerance, currently both abstol and reltol are set to this value
Type float

index
array index of the event location if one was detected, terminal index otherwise
Type int

exitflag
flag to indicate and classify event occurrence or lack of it
Type int

tc
truncated time array, sec
Type numpy.ndarray

rc
truncated radial distance array, m
Type numpy.ndarray

thetac
truncated longitude solution array, rad
Type numpy.ndarray

phic
truncated latitude solution, rad
Type numpy.ndarray

vc
truncated speed solution array, m/s
Type numpy.ndarray

psic
truncated heading angle solution, rad
Type numpy.ndarray

gammac
truncated flight-path angle solution, rad
Type numpy.ndarray

drangec
truncated downrange solution array, m

Type numpy.ndarray
t_minc
truncated time array, minutes
Type numpy.ndarray
h_kmc
truncated altitude array, km
Type numpy.ndarray
v_kmsc
truncated speed array, km/s
Type numpy.ndarray
phi_degC
truncated latitude array, deg
Type numpy.ndarray
psi_degC
truncated heading angle, deg
Type numpy.ndarray
theta_degC
truncated longitude array, deg
Type numpy.ndarray
gamma_degC
truncated flight-path angle array, deg
Type numpy.ndarray
drange_kmc
truncated downrange array, km
Type numpy.ndarray
acc_net_g
acceleration load solution, Earth G
Type numpy.ndarray
acc_drag_g
drag acceleration solution, Earth G
Type numpy.ndarray
dyn_pres_atm
dynamic pressure solution, Pa
Type numpy.ndarray
stag_pres_atm
stagnation pressure solution, Pa
Type numpy.ndarray
q_stag_con
stagnation-point convective heat rate, W/cm²
Type numpy.ndarray

q_stag_rad
stagnation-point radiative heat rate, W/cm²

Type numpy.ndarray

q_stag_total
stagnation-point radiative total heat rate, W/cm²

Type numpy.ndarray

heatload
stagnation point heat load, J/cm²

Type numpy.ndarray

maxRollRate
maximum allowed roll rate, degrees per second

Type float

beta1
beta1 (smaller ballistic coeff.) for drag modulation, kg/m²

Type float

betaRatio
ballistic coefficient ratio for drag modulation

Type float

target_peri_km
vehicle target periapsis altitude, km

Type float

target_apo_km
vehicle target apoapsis altitude, km

Type float

target_apo_km_tol
vehicle target apoapsis altitude error tolerance, km used by guidance algorithm

Type float

Ghdot
Ghdot term for vehicle equilibrium glide phase

Type float

Gq
Gq term for vehicle equilibrium glide phase guidance

Type float

v_switch_kms
speed below which eq. glide phase is terminated

Type float

t_step_array
time step array for guided aerocapture trajectory, min

Type numpy.ndarray

delta_deg_array
bank angle array for guided aerocapture trajectory, deg

Type numpy.ndarray
hdot_array
altitude rate array for guided aerocapture trajectory, m/s

Type numpy.ndarray
hddot_array
altitude acceleration array for guided aerocapture, m/s²

Type numpy.ndarray
qref_array
reference dynamic pressure array for guided aerocapture, Pa

Type numpy.ndarray
q_array
actual dynamic pressure array for guided aerocapture, Pa

Type numpy.ndarray
h_step_array
altitude array for guided aerocapture, meters

Type numpy.ndarray
acc_step_array
acceleration array for guided aerocapture, Earth G

Type numpy.ndarray
acc_drag_array
acceleration due to drag for guided aerocapture, Earth G

Type numpy.ndarray
density_mes_array
measured density array during descending leg, kg/m³

Type numpy.ndarray
density_mes_int
measured density interpolation function

Type scipy.interpolate.interpolate.interp1d

minAlt
minimum altitude at which density measurement is available, km

Type float

lowAlt_km
lower altitude to which density model is to be extrapolated based on available measurements, km

Type float

numPoints_lowAlt
number of points to evaluate extrapolation at below the altitude where measurements are available

Type int

hdot_threshold
threshold altitude rate (m/s) above which density measurement is terminated and apoapsis prediction is initiated

Type float

t_min_eg

time solution of equilibrium glide phase, min

Type numpy.ndarray

h_km_eg

altitude array of equilibrium glide phase, km

Type numpy.ndarray

v_kms_eg

speed solution of equilibrium glide phase, km/s

Type numpy.ndarray

theta_deg_eg

longitude solution of equilibrium glide phase, deg

Type numpy.ndarray

phi_deg_eg

latitude solution of equilibrium glide phase, deg

Type numpy.ndarray

psi_deg_eg

heading angle solution of equilibrium glide phase, deg

Type numpy.ndarray

gamma_deg_eg

flight-path angle solution of eq. glide phase, deg

Type numpy.ndarray

drange_km_eg

downrange solution of eq. glide phase, km

Type numpy.ndarray

acc_net_g_eg

acceleration solution of eq. glide phase, Earth G

Type numpy.ndarray

dyn_pres_atm_eg

dynamic pressure solution of eq. glide phase, atm

Type numpy.ndarray

stag_pres_atm_eg

stagnation pressure solution of eq. glide phase, atm

Type numpy.ndarray

q_stag_total_eg

stag. point total heat rate of eq. glide phase, W/cm²

Type numpy.ndarray

heatload_eg

stag. point heatload solution of eq. glide phase, J/cm²

Type numpy.ndarray

t_switch

switch time from eq. glide to exit phase, min

Type float
h_switch
altitude at which guidance switched to exit phase, km
Type float
v_switch
speed at which guidance switched to exit phase, km/s
Type float
p_switch
bank angle at which guidance switched to exit phase, deg
Type float
t_min_full
time solution of full (eq. gllide + exit phase), min
Type numpy.ndarray
h_km_full
altitude array of full (eq. gllide + exit phase), km
Type numpy.ndarray
v_kms_full
speed solution of full (eq. gllide + exit phase), km/s
Type numpy.ndarray
theta_deg_full
longitude solution of full (eq. gllide + exit phase), deg
Type numpy.ndarray
phi_deg_full
latitude solution of full (eq. gllide + exit phase), deg
Type numpy.ndarray
psi_deg_full
heading angle solution of full (eq. gllide + exit phase), deg
Type numpy.ndarray
gamma_deg_full
flight-path angle solution of full (eq. gllide + exit phase), deg
Type numpy.ndarray
drange_km_full
downrange solution of full (eq. gllide + exit phase), km
Type numpy.ndarray
acc_net_g_full
acceleration solution of full (eq. gllide + exit phase), Earth G
Type numpy.ndarray
dyn_pres_atm_full
dynamic pressure solution of full (eq. gllide + exit phase), atm
Type numpy.ndarray

stag_pres_atm_full
stagnation pressure solution of full (eq. gllide + exit phase), atm
Type numpy.ndarray

q_stag_total_full
stag. point total heat rate of full (eq. gllide + exit phase), W/cm2
Type numpy.ndarray

heatload_full
stag. point heatload solution of full (eq. gllide + exit phase), J/cm2
Type numpy.ndarray

NPOS
NPOS value from GRAM model output is the number of data points (altitude) in each atm. profile
Type int

NMONTE
NMONTE is the number of Monte Carlo atm profiles from GRAM model output
Type int

heightCol
column number of altitude values in Monte Carlo density file
Type int

densLowCol
column number of low density value in Monte Carlo density file
Type int

densAvgCol
column number of avg. density value in Monte Carlo density file
Type int

densHighCol
column number of high density value in Monte Carlo density file
Type int

densTotalCol
column number of total density value in Monte Carlo density file
Type int

heightInKmFlag
set to True if height values are in km
Type bool

nominalEFPA
nominal entry-flight-path angle
Type float

EFPA_1sigma_value
1-sigma error for EFPA
Type float

nominalLD
nominal vehicle lift-to-drag ratio

Type float
LD_1sigma_value
1-sigma error for vehicle lift-to-drag ratio
Type float
vehicleCopy
copy of the original vehicle object
Type Vehicle.vehicle
timeStep
guidance cycle time, sec
Type float
dt
max. solver timestep, sec
Type float
maxTimeSecs
maximum propagation time used by guidance algorithm, sec
Type float
t_min_en
time solution of entry phase (DM), min
Type numpy.ndarray
h_km_en
altitude solution of entry phase (DM), km
Type numpy.ndarray
v_kms_en
velocity solution of entry phase (DM), km/s
Type numpy.ndarray
theta_deg_en
longitude solution of entry phase (DM), deg
Type numpy.ndarray
phi_deg_en
latitude solution of entry phase (DM), deg
Type numpy.ndarray
psi_deg_en
heading angle solution of entry phase (DM), deg
Type numpy.ndarray
gamma_deg_en
FPA solution of entry phase (DM), deg
Type numpy.ndarray
drange_km_en
downrange solution of entry phase (DM), km
Type numpy.ndarray

acc_net_g_en
acceleration solution of entry phase, Earth g

Type numpy.ndarray

dyn_pres_atm_en
dynamic pressures solution of entry phase, atm

Type numpy.ndarray

stag_pres_atm_en
stagnation pressure solution of entry phase, atm

Type numpy.ndarray

q_stag_total_en
heat rate solution of entry phase, W/cm²

Type numpy.ndarray

heatload_en
heatload solution of entry phase, J/cm²

Type float

D(*r, theta, phi, v*)

Computes the vehicle aerodynamic drag, as a function of the vehicle location(*r,theta, phi*), and velocity.

Parameters

- **r** (*float*) – radial position value, scalar, m
- **theta** (*float*) – longitude, radians
- **phi** (*float*) – latitude, radians
- **v** (*float*) – planet-relative speed, m/s

Returns **ans** – aerodynamic drag force, N

Return type float

Dbar(*rbar, theta, phi, vbar*)

Computes the non-dimensional vehicle aerodynamic drag, as a function of the vehicle location(*r,theta, phi*), and velocity.

Parameters

- **rbar** (*float*) – non-dimensional radial position
- **theta** (*float*) – longitude, radians
- **phi** (*float*) – latitude, radians
- **vbar** (*float*) – non-dimensional planet-relative speed, m/s

Returns **ans** – non-dimensional aerodynamic drag force

Return type float

Dvectorized(*r, theta, phi, v*)

Vectorized version of the D() function

Computes the vehicle aerodynamic drag array over the provided trajectory array, as a function of the vehicle location array (*r[:,theta[:, phi[:]]*), and velocity.

Parameters

- **r** (`numpy.ndarray`) – radial position array, m
- **theta** (`numpy.ndarray`) – longitude array, radians
- **phi** (`numpy.ndarray`) – latitude array, radians
- **v** (`numpy.ndarray`) – planet-relative speed array, m/s

Returns `ans` – aerodynamic drag force array, N

Return type `numpy.ndarray`

EOM (`y, t, delta`)

Define the EoMs to propogate the 3DoF trajectory inside the atmosphere of a an oblate rotating planet.

Reference 1: Vinh, Chapter 3. Reference 2: Lescynzki, MS Thesis, NPS.

Parameters

- **y** (`numpy.ndarray`) – trajectory state vector
- **t** (`numpy.ndarray`) – trajectory time vector
- **delta** (`float`) – bank angle, rad

Returns `ans` – derivate vector of state, process equations

Return type `dydt`

L (`r, theta, phi, v`)

Computes the vehicle aerodynamic lift, as a function of the vehicle location(`r,theta, phi`), and velocity.

Parameters

- **r** (`float`) – radial position value, scalar, m
- **theta** (`float`) – longitude, radians
- **phi** (`float`) – latitude, radians
- **v** (`float`) – planet-relative speed, m/s

Returns `ans` – aerodynamic lift force, N

Return type `float`

Lbar (`rbar, theta, phi, vbar`)

Computes the non-dimensional vehicle aerodynamic lift, as a function of the vehicle location(`r,theta, phi`), and velocity.

Parameters

- **rbar** (`float`) – non-dimensional radial position
- **theta** (`float`) – longitude, radians
- **phi** (`float`) – latitude, radians
- **vbar** (`float`) – non-dimensional planet-relative speed, m/s

Returns `ans` – non-dimensional aerodynamic lift force

Return type `float`

Lvectorized (`r, theta, phi, v`)

Vectorized version of the L() function

Computesthe vehicle aerodynamic lift array over the provided trajectory array, as a function of the vehicle location array (`r[:,],theta[:,], phi[:,]`), and velocity.

Parameters

- **r** (*numpy.ndarray*) – radial position array, m
- **theta** (*numpy.ndarray*) – longitude array, radians
- **phi** (*float*) – latitude array, radians
- **v** (*float*) – planet-relative speed array, m/s

Returns **ans** – aerodynamic lift force array, N**Return type** *numpy.ndarray***a_n** (*r, theta, phi, v, delta*)

Function to return normal acceleration term a_n;

a_n is the normal acceleration term along perpendicular to the velocity vector, in the plane of the trajectory.

Current formulation does not include thrust, can be added here later in place of 0.0

Parameters

- **r** (*float*) – radial position value, scalar, m
- **theta** (*float*) – longitude, radians
- **phi** (*float*) – latitude, radians
- **v** (*float*) – planet-relative speed, m/s
- **delta** (*float*) – bank angle, rad

Returns **ans** – normal acceleration term a_n**Return type** float**a_nbar** (*rbar, theta, phi, vbar, delta*)

Function to return non-dimensional normal acceleration term a_nbar;

a_nbar is the tangential acceleration term along the direction of the velocity vector.

Current formulation does not include thrust, can be added here later in place of 0.0

Parameters

- **rbar** (*float*) – non-dimensional radial position
- **theta** (*float*) – longitude, radians
- **phi** (*float*) – latitude, radians
- **vbar** (*float*) – non-dimensional planet-relative speed, m/s
- **delta** (*float*) – bank angle, rad

Returns **ans** – non-dimensional tangential acceleration term a_nbar**Return type** float**a_nvectorized** (*r, theta, phi, v, delta*)

Vectorized version of the a_n() function

Parameters

- **r** (*numpy.ndarray*) – radial position array, m
- **theta** (*numpy.ndarray*) – longitude array, radians
- **phi** (*numpy.ndarray*) – latitude array, radians

- **v** (*numpy.ndarray*) – planet-relative speed array, m/s
- **delta** (*float*) – bank angle, rad

Returns **ans** – normal acceleration array, m/s²

Return type *numpy.ndarray*

a_s (*r, theta, phi, v, delta*)

Function to return tangential acceleration term a_s;

a_s is the tangential acceleration term along the direction of the velocity vector.

Current formulation does not include thrust, can be added here later in place of 0.0

Parameters

- **r** (*float*) – radial position value, scalar, m
- **theta** (*float*) – longitude, radians
- **phi** (*float*) – latitude, radians
- **v** (*float*) – planet-relative speed, m/s
- **delta** (*float*) – bank angle, rad

Returns **ans** – tangential acceleration term a_s

Return type float

a_sbar (*rbar, theta, phi, vbar, delta*)

Function to return non-dimensional tangential acceleration term a_sbar;

a_sbar is the tangential acceleration term along the direction of the velocity vector.

Current formulation does not include thrust, can be added here later in place of 0.0

Parameters

- **rbar** (*float*) – non-dimensional radial position
- **theta** (*float*) – longitude, radians
- **phi** (*float*) – latitude, radians
- **vbar** (*float*) – non-dimensional planet-relative speed, m/s
- **delta** (*float*) – bank angle, rad

Returns **ans** – non-dimensional tangential acceleration term a_sbar

Return type float

a_svectorized (*r, theta, phi, v, delta*)

Vectorized version of the a_s() function

Parameters

- **r** (*numpy.ndarray*) – radial position array, m
- **theta** (*numpy.ndarray*) – longitude array, radians
- **phi** (*numpy.ndarray*) – latitude array, radians
- **v** (*numpy.ndarray*) – planet-relative speed array, m/s
- **delta** (*float*) – bank angle, rad

Returns **ans** – tangential acceleration array, m/s²

Return type numpy.ndarray

a_w(*r, theta, phi, v, delta*)

Function to return binormal acceleration term a_w;

a_n is the binormal acceleration term along perpendicular to the velocity vector, perpendicular to the plane of the trajectory.

Current formulation does not include thrust, can be added here later in place of 0.0

Parameters

- **r** (*float*) – radial position value, scalar, m
- **theta** (*float*) – longitude, radians
- **phi** (*float*) – latitude, radians
- **v** (*float*) – planet-relative speed, m/s
- **delta** (*float*) – bank angle, rad

Returns **ans** – binormal acceleration term a_n

Return type float

a_wbar(*rbar, theta, phi, vbar, delta*)

Function to return non-dimensional normal acceleration term a_wbar;

a_wbar is the tangential acceleration term along the direction of the velocity vector.

Current formulation does not include thrust, can be added here later in place of 0.0

Parameters

- **rbar** (*float*) – non-dimensional radial position
- **theta** (*float*) – longitude, radians
- **phi** (*float*) – latitude, radians
- **vbar** (*float*) – non-dimensional planet-relative speed, m/s
- **delta** (*float*) – bank angle, rad

Returns **ans** – non-dimensional tangential acceleration term a_wbar

Return type float

a_wvectorized(*r, theta, phi, v, delta*)

Vectorized version of the a_w() function

Parameters

- **r** (*numpy.ndarray*) – radial position array, m
- **theta** (*numpy.ndarray*) – longitude array, radians
- **phi** (*numpy.ndarray*) – latitude array, radians
- **v** (*numpy.ndarray*) – planet-relative speed array, m/s
- **delta** (*float*) – bank angle, rad

Returns **ans** – binormal acceleration array, m/s2

Return type numpy.ndarray

cfgammabar (*rbar, phi, vbar, psi, gamma*)

Function to return non dimensional centrifugal acceleration term cfgammabar

cfgammabar is the non dimensional centrifugal acceleration term in the EOM

Parameters

- **rbar** (*float*) – non-dimensional radial position
- **phi** (*float*) – latitude, radians
- **vbar** (*float*) – non-dimensional planet-relative speed, m/s
- **psi** (*float*) – heading angle, rad
- **gamma** (*float*) – flight-path angle, rad

Returns **ans** – non dimensional centrifugal acceleration cfpsibar**Return type** float**cfpsibar** (*rbar, phi, vbar, psi, gamma*)

Function to return non dimensional centrifugal acceleration term cfpsibar

cfpsibar is the non dimensional centrifugal acceleration term in the EOM

Parameters

- **rbar** (*float*) – non-dimensional radial position
- **phi** (*float*) – latitude, radians
- **vbar** (*float*) – non-dimensional planet-relative speed, m/s
- **psi** (*float*) – heading angle, rad
- **gamma** (*float*) – flight-path angle, rad

Returns **ans** – non dimensional centrifugal acceleration cfpsibar**Return type** float**cfvbar** (*rbar, phi, vbar, psi, gamma*)

Function to return non dimensional centrifugal acceleration term cfvbar

cfvbar is the non dimensional centrifugal acceleration term cfvbar in the EOM

Parameters

- **rbar** (*float*) – non-dimensional radial position
- **phi** (*float*) – latitude, radians
- **vbar** (*float*) – non-dimensional planet-relative speed, m/s
- **psi** (*float*) – heading angle, rad
- **gamma** (*float*) – flight-path angle, rad

Returns **ans** – non dimensional centrifugal acceleration cfvbar**Return type** float**classifyTrajectory** (*r*)

This function checks the trajectory for “events” which are used to truncate the trajectory

A “skip out event” is said to occur as when the vehicle altitude did not hit the surface and exceeds the prescribed skip out altitude.

A “time out event” is said to occur if the vehicle did not exceed the skip out altitude, and did not reach the trap in altitude.

A “trap in” event is said to occur when the vehicle altitude falls below the prescribed trap in altitude.

This function checks for these events and returns the array index of the “event” location and an exitflag exitflag to indicate if an event was detected or not. exitflag = 1.0 indicates “skip out event” has occurred. exitflag = 0.0 indicates no “event” was detected in the trajectory, consider increasing simulation time. exitflag = -1.0 indicates “trap in event” has occurred.

Parameters **r** (*numpy.ndarray*) – dimensional radial distance solution, meters

Returns

- **index** (*int*) – array index of the event location if one was detected, terminal index otherwise
- **exitflag** (*int*) – flag to indicate and classify event occurrence or lack of it

cogammabar (*rbar, phi, vbar, psi, gamma*)

Function to return non dimensional Coriolis acceleration term cogammabar

Parameters

- **rbar** (*float*) – non-dimensional radial position
- **phi** (*float*) – latitude, radians
- **vbar** (*float*) – non-dimensional planet-relative speed, m/s
- **psi** (*float*) – heading angle, rad
- **gamma** (*float*) – flight-path angle, rad

Returns **ans** – non dimensional Coriolis acceleration cogammabar

Return type float

computeAccelerationDrag (*tc, rc, thetac, phic, vc, index, delta*)

This function computes the drag acceleration load (Earth G’s) over the entire trajectory from trajectory data returned by the solver.

Parameters

- **tc** (*numpy.ndarray*) – truncated time array, sec
- **rc** (*numpy.ndarray*) – truncated radial distance array, m
- **thetac** (*numpy.ndarray*) – truncated longitude array, rad
- **phic** (*numpy.ndarray*) – truncated latitude array, rad
- **vc** (*numpy.ndarray*) – truncated speed array, m
- **index** (*int*) – array index of detected event location / max index if no event detected
- **delta** (*float*) – bank angle, rad

Returns **acc_drag_g** – drag acceleration load (Earth G’s) over the entire trajectory

Return type *numpy.ndarray*

computeAccelerationLoad (*tc, rc, thetac, phic, vc, index, delta*)

This function computes the acceleration load (Earth G’s) over the entire trajectory from trajectory data returned by the solver.

Parameters

- **tc** (*numpy.ndarray*) – truncated time array, sec

- **rc** (*numpy.ndarray*) – truncated radial distance array, m
- **thetac** (*numpy.ndarray*) – truncated longitude array, rad
- **phic** (*numpy.ndarray*) – truncated latitude array, rad
- **vc** (*numpy.ndarray*) – truncated speed array, m
- **index** (*int*) – array index of detected event location / max index if no event detected
- **delta** (*float*) – bank angle, rad

Returns `acc_net_g` – acceleration load (Earth G's) over the entire trajectory

Return type `numpy.ndarray`

computeAngMomScalar (*terminal_r*, *terminal_v*, *terminal_g*)

This function computes the specific angular momentum (orbital) of the vehicle at an instance given its current radial distance, speed, and flight-path angle.

Parameters

- **terminal_r** (*float*) – radial distance, meters
- **terminal_v** (*float*) – speed, meters/sec
- **terminal_g** (*float*) – flight-path angle, rad

Returns `ans` – specific angular momentum, SI units

Return type `float`

computeDynPres (*r*, *v*)

This function computes the dynamic pressure over the entire trajectory.

Parameters

- **r** (*numpy.ndarray*) – radial distance array, m
- **v** (*numpy.ndarray*) – speed array, m/s

Returns `ans` – dynamic pressure, Pa

Return type `numpy.ndarray`

computeEccScalar (*h*, *E*)

This function computes the eccentricity of the orbit given its specific angular momentum, and total specific mechanical energy.

Parameters

- **h** (*float*) – specific angular momentum, SI units
- **E** (*float*) – specific energy, J/kg

Returns `ans` – eccentricity value

Return type `float`

computeEnergy (*rc*, *vc*)

This function computes the total specific mechanical energy of the vehicle over the entire trajectory.

Parameters

- **rc** (*numpy.ndarray*) – radial distance array, m
- **vc** (*numpy.ndarray*) – speed array, m/s

Returns `ans` – specific energy, J/kg

Return type numpy.ndarray

computeEnergyScalar(*r, v*)

This function computes the total specific mechanical energy of the vehicle at an instance.

Parameters

- **r** (*float*) – radial distance, m
- **v** (*float*) – speed array, m/s

Returns **ans** – specific energy, J/kg

Return type float

computeHeatingForMultipleRN(*tc, rc, vc, rn_array*)

This function computes the max. stag. pont heating rate and heat load for an array of nose radii.

Parameters

- **tc** (*numpy.ndarray*) – truncated time array, sec
- **rc** (*numpy.ndarray*) – truncated radial distance array, m
- **vc** (*numpy.ndarray*) – speed array, m/s
- **rn_array** (*numpy.ndarray*) – nose radius array, m

Returns

- **q_stag_max** (*numpy.ndarray*) – max. heat rate, W/cm²
- **heatload** (*numpy.ndarray*) – max. heatload, J/cm²

computeMach(*rc, vc*)

This function computes the Mach. no over the entire trajectory.

Parameters

- **rc** (*numpy.ndarray*) – radial distance array, m
- **vc** (*numpy.ndarray*) – speed array, m/s

Returns **ans** – Mach no.

Return type numpy.ndarray

computeSemiMajorAxisScalar(*E*)

This function computes the semi-major axis of the orbit given its total specific mechanical energy.

Parameters **E** (*float*) – specific energy, J/kg

Returns **ans** – semi major axis, km

Return type float

computeStagPres(*rc, vc*)

This function computes the stag. pressure over the entire trajectory.

Parameters

- **rc** (*numpy.ndarray*) – radial distance array, m
- **vc** (*numpy.ndarray*) – speed array, m/s

Returns **ans** – stag. pressure, Pa

Return type numpy.ndarray

computeStagTemp (*rc, vc*)

This function computes the stag. temperature over the entire trajectory.

Parameters

- **rc** (*numpy.ndarray*) – radial distance array, m
- **vc** (*numpy.ndarray*) – speed array, m/s

Returns **ans** – stag. temperature, K

Return type *numpy.ndarray*

computeTCW (*t_sec, dt, gamma0_deg_guess_low_os, gamma0_deg_guess_high_os, gamma0_deg_guess_low_us, gamma0_deg_guess_high_us, gamma_deg_tol_os, gamma_deg_tol_us, targetApopasisAltitude_km*)

Computes the theoretical corridor width (TCW) for lift modulation aerocapture.

TCW = overShootLimit - underShootLimit

Parameters

- **t_sec** (*float*) – propagation time, seconds
- **dt** (*float*) – max. time step, seconds
- **gamma0_deg_guess_low_os** (*float*) – lower bound for the guess of overshoot limit FPA, deg
- **gamma0_deg_guess_high_os** (*float*) – upper bound for the guess of overshoot limit FPA, deg
- **gamma0_deg_guess_low_us** (*float*) – lower bound for the guess of undershoot limit FPA, deg
- **gamma0_deg_guess_high_us** (*float*) – upper bound for the guess of undershoot limit FPA, deg
- **gamma_deg_tol_os** (*float*) – desired accuracy for computation of the overshoot limit, deg
- **gamma_deg_tol_us** (*float*) – desired accuracy for computation of the undershoot limit, deg
- **targetApopasisAltitude_km** (*float*) – target apoapsis altitude , km

Returns **TCW** – Theoretical Corridor Width, deg

Return type float

computeTCW2 (*t_sec, dt, gamma0_deg_guess_low_os, gamma0_deg_guess_high_os, gamma0_deg_guess_low_us, gamma0_deg_guess_high_us, gamma_deg_tol_os, gamma_deg_tol_us, targetApopasisAltitude_km*)

Computes the theoretical corridor width (TCW) for lift modulation aerocapture. Includes effect of planet rotation.

TCW = overShootLimit - underShootLimit

Parameters

- **t_sec** (*float*) – propagation time, seconds
- **dt** (*float*) – max. time step, seconds
- **gamma0_deg_guess_low_os** (*float*) – lower bound for the guess of overshoot limit FPA, deg

- **gamma0_deg_guess_high_os** (*float*) – upper bound for the guess of overshoot limit FPA, deg
- **gamma0_deg_guess_low_us** (*float*) – lower bound for the guess of undershoot limit FPA, deg
- **gamma0_deg_guess_high_us** (*float*) – upper bound for the guess of undershoot limit FPA, deg
- **gamma_deg_tol_os** (*float*) – desired accuracy for computation of the overshoot limit, deg
- **gamma_deg_tol_us** (*float*) – desired accuracy for computation of the undershoot limit, deg
- **targetApopasisAltitude_km** (*float*) – target apoapsis altitude , km

Returns TCW – Theoretical Corridor Width, deg

Return type float

computeTCWD (*t_sec*, *dt*, *gamma0_deg_guess_low*, *gamma0_deg_guess_high*, *gamma_deg_tol*, *targetApopasisAltitude_km*)

Computes the theoretical corridor width (TCWD) for drag modulation aerocapture.

TCWD = overShootLimit - underShootLimit

Parameters

- **t_sec** (*float*) – propagation time, seconds
- **dt** (*float*) – max. time step, seconds
- **gamma0_deg_guess_low_os** (*float*) – lower bound for the guess of overshoot limit FPA, deg
- **gamma0_deg_guess_high_os** (*float*) – upper bound for the guess of overshoot limit FPA, deg
- **gamma0_deg_guess_low_us** (*float*) – lower bound for the guess of undershoot limit FPA, deg
- **gamma0_deg_guess_high_us** (*float*) – upper bound for the guess of undershoot limit FPA, deg
- **gamma_deg_tol_os** (*float*) – desired accuracy for computation of the overshoot limit, deg
- **gamma_deg_tol_us** (*float*) – desired accuracy for computation of the undershoot limit, deg
- **targetApopasisAltitude_km** (*float*) – target apoapsis altitude , km

Returns TCWD – Theoretical Corridor Width (Drag Modulation), deg

Return type float

compute_ApoapsisAltitudeKm (*terminal_r*, *terminal_v*, *terminal_g*, *terminal_theta*, *terminal_phi*, *terminal_psi*)

Compute the apoapsis altitude given conditions at atmospheric exit interface. Note this function includes correction to account for rotation of planet.

Terminal values refer to those at atmospheric exit.

Parameters

- **terminal_r** (*float*) – radial distance, m

- **terminal_v** (*float*) – terminal speed, m/s
- **terminal_g** (*float*) – terminal FPA, rad
- **terminal_theta** (*float*) – terminal longitude, rad
- **terminal_phi** (*float*) – terminal latitude, rad
- **terminal_psi** (*float*) – terminal heading angle

Returns **hp_km** – apoapsis altitude, km

Return type float

compute_PeriapsisAltitudeKm (*terminal_r*, *terminal_v*, *terminal_g*, *terminal_theta*, *terminal_phi*, *terminal_psi*)

Compute the periapsis altitude given conditions at atmospheric exit interface. Note this function includes correction to account for rotation of planet.

Terminal values refer to those at atmospheric exit.

Parameters

- **terminal_r** (*float*) – radial distance, m
- **terminal_v** (*float*) – terminal speed, m/s
- **terminal_g** (*float*) – terminal FPA, rad
- **terminal_theta** (*float*) – terminal longitude, rad
- **terminal_phi** (*float*) – terminal latitude, rad
- **terminal_psi** (*float*) – terminal heading angle

Returns **hp_km** – periapsis altitude, km

Return type float

compute_apoapsis_raise_DV (*peri_km_current*, *apo_km_current*, *apo_km_target*)

Compute the propulsive DV to raise the orbit apoapsis to the target value.

Parameters

- **peri_km_current** (*float*) – current periapsis altitude, km
- **apo_km_current** (*float*) – current apoapsis altitude, km
- **apo_km_target** (*float*) – target apoapsis altitude, km

Returns **dV** – apoapsis raise DV, m/s

Return type float

compute_periapsis_raise_DV (*current_peri_km*, *current_apo_km*, *target_peri_km*)

Compute the propulsive DV to raise the orbit periapsis to the target value.

Parameters

- **current_peri_km** (*float*) – current periapsis altitude, km
- **current_apo_km** (*float*) – current apoapsis altitude, km
- **target_peri_km** (*float*) – target periapsis altitude, km

Returns **dV** – periapse raise DV, m/s

Return type float

convertToKPa (pres)

Convert a pressure solution array from Pa to kPa.

Parameters **pres** (*numpy.ndarray*) – pressure (static/dynamic/total), Pascal (Pa)

Returns **ans** – pressure (static/dynamic/total), kiloPascal (kPa)

Return type *numpy.ndarray*

convertToPerCm2 (heatrate)

Convert a heat rate from W/m² to W/cm².

Parameters **heatrate** (*numpy.ndarray*) – stagnation-point heat rate, W/m²

Returns **ans** – stagnation-point heat rate, W/cm²

Return type *numpy.ndarray*

convertToPlotUnits (t, r, v, phi, psi, theta, gamma, drange)

Convert state vector components to units appropriate for evolution plots.

Parameters

- **t** (*numpy.ndarray*) – time array, sec
- **r** (*numpy.ndarray*) – radial distance array, m
- **v** (*numpy.ndarray*) – speed array, m
- **phi** (*numpy.ndarray*) – latitude array, rad
- **psi** (*numpy.ndarray*) – heading angle array, rad
- **theta** (*numpy.ndarray*) – longitude array, rad
- **gamma** (*numpy.ndarray*) – flight path angle array, rad
- **drange** (*numpy.ndarray*) – downrange array, meters

Returns

- **t_min** (*numpy.ndarray*) – time array, minutes
- **h_km** (*numpy.ndarray*) – altitude array, km
- **v_kms** (*numpy.ndarray*) – speed array, km/s
- **phi_deg** (*numpy.ndarray*) – latitude array, deg
- **psi_deg** (*numpy.ndarray*) – heading angle array, deg
- **theta_deg** (*numpy.ndarray*) – longitude array, deg
- **gamma_deg** (*numpy.ndarray*) – flight path angle array, deg
- **drange_km** (*numpy.ndarray*) – downrange array, km

copsibar (rbar, phi, vbar, psi, gamma)

Function to return non dimensional Coriolis acceleration term copsibar

Parameters

- **rbar** (*float*) – non-dimensional radial position
- **phi** (*float*) – latitude, radians
- **vbar** (*float*) – non-dimensional planet-relative speed, m/s
- **psi** (*float*) – heading angle, rad

- **gamma** (*float*) – flight-path angle, rad

Returns **ans** – non dimensional Coriolis acceleration copisbar

Return type float

createDensityMeasuredFunction (*h_step_array*, *density_mes_array*, *lowAlt_km*, *numPoints_lowAlt*)

Computes a density function based on measurements made during the descending leg of the aerocapture maneuver.

Parameters

- **h_step_array** (*numpy.ndarray*) – height array at which density is measured, km
- **density_mes_array** (*float*) – density array corresponding to h_step_array, kg/m³
- **lowAlt_km** (*float*) – lower altitude to which density model is to be extrapolated based on available measurements, km
- **numPoints_lowAlt** (*int*) – number of points to evaluate extrapolation at below the altitude where measurements are available.

Returns

- **density_mes_int** (*scipy.interpolate.interpolate.interp1d*) – interpolated measured density lookup function
- **minAlt** (*float*) – minimum altitude at which density measurements were available

createQPlot (*t_sec*, *dt*, *delta_deg*)

Creates q-plots as described by Cerimele and Gamble, 1985.

Parameters

- **t_sec** (*float*) – propagation time, seconds
- **dt** (*float*) – max. solver time step
- **delta_deg** (*float*) – commanded bank angle, degrees

Returns

Return type plt.plot object

dummyVehicle (*density_mes_int*)

Create a copy of the vehicle object which uses a measured density profile for propagation.

Parameters **density_mes_int** (*scipy.interpolate.interpolate.interp1d*)
– density interpolation function

Returns **vehicleCopy** – dummy vehicle object

Return type vehicle object

findEFPALimitD (*t_sec*, *dt*, *gamma0_deg_guess_low*, *gamma0_deg_guess_high*, *gamma_deg_tol*,
targetApopasisAltitude_km)

This function computes the limiting EFPA for drag modulation aerocapture.

A bisection algorithm is used to compute the limit.

Parameters

- **t_sec** (*float*) – propagation time, seconds
- **dt** (*float*) – max. time step, seconds
- **gamma0_deg_guess_low** (*float*) – lower bound for the guess of limit FPA, deg

- **gamma0_deg_guess_high** (*float*) – upper bound for the guess of limit FPA, deg
- **gamma_deg_tol** (*float*) – desired accuracy for computation of the limit, deg
- **targetApopasisAltitude_km** (*float*) – target apoapsis altitude , km

Returns

- **EFPALimit** (*float*) – limit EFPA, deg
- **exitflag** (*float*) – flag to indicate if a solution could not be found for the limit EFPA
- *exitflag = 1.0 indicates over shoot limit was found.*
- *exitflag = 0.0 indicates overshoot limit was not found*
- *within user specified bounds.*

findOverShootLimit (*t_sec*, *dt*, *gamma0_deg_guess_low*, *gamma0_deg_guess_high*,
 gamma_deg_tol, *targetApopasisAltitude_km*)

Computes the overshoot limit entry flight-path angle for aerocapture vehicle using bisection algorithm.

This is shallowest entry flight path angle for which a full lift down trajectory gets the vehicle captured into a post atmospheric exit orbit with the desired target apoapsis altitude.

Note: the overshoot limit entry flight path angle should be computed with an accuracy of at least 10 decimal places to ensure the correct atmospheric trajectory is simulated.

A bisection algorithm is used to compute the overshoot limit.

Parameters

- **t_sec** (*float*) – propogation time, seconds
- **dt** (*float*) – max. time step, seconds
- **gamma0_deg_guess_low** (*float*) – lower bound for the guess of overshoot limit FPA, deg
- **gamma0_deg_guess_high** (*float*) – upper bound for the guess of overshoot limit FPA, deg
- **gamma_deg_tol** (*float*) – desired accuracy for computation of the overshoot limit, deg
- **targetApopasisAltitude_km** (*float*) – target apoapsis altitude , km

Returns

- **overShootLimit** (*float*) – overshoot limit EFPA, deg
- **exitflag_os** (*float*) – flag to indicate if a solution could not be found for the overshoot limit
- *exitflag_os = 1.0 indicates over shoot limit was found.*
- *exitflag_os = 0.0 indicates overshoot limit was not found*
- *within user specified bounds.*

findOverShootLimit2 (*t_sec*, *dt*, *gamma0_deg_guess_low*, *gamma0_deg_guess_high*,
 gamma_deg_tol, *targetApopasisAltitude_km*)

Computes the overshoot limit entry flight-path angle for aerocapture vehicle using bisection algorithm. Includes effect of planet rotation on inertial speed.

This is shallowest entry flight path angle for which a full lift down trajectory gets the vehicle captured into a post atmospheric exit orbit with the desired target apoapsis altitude.

Note: the overshoot limit entry flight path angle should be computed with an accuracy of at least 10 decimal places to ensure the correct atmospheric trajectory is simulated.

A bisection algorithm is used to compute the overshoot limit.

Parameters

- **t_sec** (*float*) – propagation time, seconds
- **dt** (*float*) – max. time step, seconds
- **gamma0_deg_guess_low** (*float*) – lower bound for the guess of overshoot limit FPA, deg
- **gamma0_deg_guess_high** (*float*) – upper bound for the guess of overshoot limit FPA, deg
- **gamma_deg_tol** (*float*) – desired accuracy for computation of the overshoot limit, deg
- **targetApopasisAltitude_km** (*float*) – target apoapsis altitude , km

Returns

- **overShootLimit** (*float*) – overshoot limit EFPA, deg
- **exitflag_os** (*float*) – flag to indicate if a solution could not be found for the overshoot limit
 - *exitflag_os = 1.0 indicates over shoot limit was found.*
 - *exitflag_os = 0.0 indicates overshoot limit was not found*
 - *within user specified bounds.*

findOverShootLimitD (*t_sec*, *dt*, *gamma0_deg_guess_low*, *gamma0_deg_guess_high*,
 gamma_deg_tol, *targetApopasisAltitude_km*)

This function computes the limiting overshoot EFPA for drag modulation aerocapture.

Parameters

- **t_sec** (*float*) – propagation time, seconds
- **dt** (*float*) – max. time step, seconds
- **gamma0_deg_guess_low** (*float*) – lower bound for the guess of limit FPA, deg
- **gamma0_deg_guess_high** (*float*) – upper bound for the guess of limit FPA, deg
- **gamma_deg_tol** (*float*) – desired accuracy for computation of the limit, deg
- **targetApopasisAltitude_km** (*float*) – target apoapsis altitude , km

Returns

- **overShootLimitD** (*float*) – overshoot limit EFPA, deg
- **exitflagD_os** (*float*) – flag to indicate if a solution could not be found for the overshoot limit EFPA
 - *exitflagD_os = 1.0 indicates over shoot limit was found.*
 - *exitflagD_os = 0.0 indicates overshoot limit was not found*
 - *within user specified bounds.*

findUnderShootLimit (*t_sec*, *dt*, *gamma0_deg_guess_low*, *gamma0_deg_guess_high*,
 gamma_deg_tol, *targetApopasisAltitude_km*)

Computes the undershoot limit entry flight-path angle for aerocapture vehicle using bisection algorithm.

This is steepest entry flight path angle for which a full lift up trajectory gets the vehicle captured into a post atmospheric exit orbit with the desired target apoapsis altitude.

Note: the undershoor limit entry flight path angle should be computed with an accuracy of at least 6 decimal places to ensure the correct atmospheric trajectory is simulated.

A bisection algorithm is used to compute the undershoot limit.

Parameters

- **t_sec** (*float*) – propogation time, seconds
- **dt** (*float*) – max. time step, seconds
- **gamma0_deg_guess_low** (*float*) – lower bound for the guess of overshoot limit FPA, deg
- **gamma0_deg_guess_high** (*float*) – upper bound for the guess of overshoot limit FPA, deg
- **gamma_deg_tol** (*float*) – desired accuracy for computation of the overshoot limit, deg
- **targetApopasisAltitude_km** (*float*) – target apoapsis altitude , km

Returns

- **underShootLimit** (*float*) – overshoot limit EFPA, deg
- **exitflag_us** (*float*) – flag to indicate if a solution could not be found for the undershoot limit
 - *exitflag_us = 1.0 indicates undershoot limit was found.*
 - *exitflag_us = 0.0 indicates overshoot limit was not found*
 - *within user specified bounds.*

findUnderShootLimit2 (*t_sec*, *dt*, *gamma0_deg_guess_low*, *gamma0_deg_guess_high*,
 gamma_deg_tol, *targetApopasisAltitude_km*)

Computes the undershoot limit entry flight-path angle for aerocapture vehicle using bisection algorithm. Includes effect of planet rotation on inertial speed.

This is steepest entry flight path angle for which a full lift up trajectory gets the vehicle captured into a post atmospheric exit orbit with the desired target apoapsis altitude.

Note: the undershoor limit entry flight path angle should be computed with an accuracy of at least 6 decimal places to ensure the correct atmospheric trajectory is simulated.

A bisection algorithm is used to compute the undershoot limit.

Parameters

- **t_sec** (*float*) – propogation time, seconds
- **dt** (*float*) – max. time step, seconds
- **gamma0_deg_guess_low** (*float*) – lower bound for the guess of overshoot limit FPA, deg
- **gamma0_deg_guess_high** (*float*) – upper bound for the guess of overshoot limit FPA, deg
- **gamma_deg_tol** (*float*) – desired accuracy for computation of the overshoot limit, deg
- **targetApopasisAltitude_km** (*float*) – target apoapsis altitude , km

Returns

- **underShootLimit** (*float*) – overshoot limit EFPA, deg
- **exitflag_us** (*float*) – flag to indicate if a solution could not be found for the undershoot limit
 - *exitflag_us = 1.0 indicates undershoot limit was found.*
 - *exitflag_us = 0.0 indicates overshoot limit was not found*
- *within user specified bounds.*

findUnderShootLimitD (*t_sec*, *dt*, *gamma0_deg_guess_low*, *gamma0_deg_guess_high*,
 gamma_deg_tol, *targetApopasisAltitude_km*)

This function computes the limiting undershoot EFPA for drag modulation aerocapture.

Parameters

- **t_sec** (*float*) – propagation time, seconds
- **dt** (*float*) – max. time step, seconds
- **gamma0_deg_guess_low** (*float*) – lower bound for the guess of limit FPA, deg
- **gamma0_deg_guess_high** (*float*) – upper bound for the guess of limit FPA, deg
- **gamma_deg_tol** (*float*) – desired accuracy for computation of the limit, deg
- **targetApopasisAltitude_km** (*float*) – target apoapsis altitude , km

Returns

- **underShootLimitD** (*float*) – undershoot limit EFPA, deg
- **exitflagD_us** (*float*) – flag to indicate if a solution could not be found for the undershoot limit EFPA
 - *exitflagD_us = 1.0 indicates undershoot limit was found.*
 - *exitflagD_us = 0.0 indicates undershoot limit was not found*
- *within user specified bounds.*

gnbar (*rbar*, *phi*, *gamma*, *psi*)

Returns the non-dimensional gravity normal acceleration term gnbar.

Parameters

- **rbar** (*float*) – non-dimensional radial position
- **phi** (*float*) – latitude, radians
- **gamma** (*float*) – flight-path angle, rad
- **psi** (*float*) – heading angle, rad

Returns **ans** – non-dimensional gravity normal acceleration term gnbar

Return type float

gphibar (*rbar*, *phi*)

Returns the non-dimensional gravity latitudinal acceleration term gphibar.

Parameters

- **rbar** (*float*) – non-dimensional radial position
- **phi** (*float*) – latitude, radians

Returns `ans` – non-dimensional gravity latitudinal acceleration term gphibar

Return type float

grbar (`rbar, phi`)

Returns the non-dimensional gravity radial acceleration term grbar.

grbar is the non dimensional gravity radial acceleration term grbar in the EOM

Parameters

- **rbar** (`float`) – non-dimensional radial position
- **phi** (`float`) – latitude, radians

Returns `ans` – non-dimensional radial acceleration term grbar

Return type float

gsbar (`rbar, phi, gamma, psi`)

Returns the non-dimensional gravity tangential acceleration term gsbar.

Parameters

- **rbar** (`float`) – non-dimensional radial position
- **phi** (`float`) – latitude, radians
- **gamma** (`float`) – flight-path angle, rad
- **psi** (`float`) – heading angle, rad

Returns `ans` – non-dimensional gravity tangential acceleration term gsbar

Return type float

gthetabar (`rbar, phi`)

Returns the non-dimensional gravity longitudinal acceleration term.

gthetabar is the non dimensional longitudinal gravity acceleration term grbar in the EOM

Parameters

- **rbar** (`float`) – non-dimensional radial position
- **phi** (`float`) – latitude, radians

Returns `ans` – non-dimensional longitudinal gravity acceleration term gthetabar

Return type float

gwbar (`rbar, phi, gamma, psi`)

Returns the non-dimensional gravity binormal acceleration term gwbar.

Parameters

- **rbar** (`float`) – non-dimensional radial position
- **phi** (`float`) – latitude, radians
- **gamma** (`float`) – flight-path angle, rad
- **psi** (`float`) – heading angle, rad

Returns `ans` – non-dimensional gravity binormal acceleration term gwbar

Return type float

hitsTargetApoapsis (*t_sec*, *dt*, *delta_deg*, *targetApopasisAltitude_km*)

This function is used to check if the vehicle undershoots or overshoots. Does not include effect of planet rotation to compute inertial speed.

Returns +1 if the vehicle is captured into an orbit with the required target apoapsis alt, -1 otherwise.

Parameters

- **t_sec** (*float*) – propogation time, seconds
- **dt** (*float*) – max. time step, seconds
- **delta_deg** (*float*) – bank angle command, deg
- **targetApopasisAltitude_km** (*float*) – target apoapsis altitude , km

Returns **ans** – -1 indicates overshoot, +1 indicates undershoot

Return type int

hitsTargetApoapsis2 (*t_sec*, *dt*, *delta_deg*, *targetApopasisAltitude_km*)

This function is used to check if the vehicle undershoots or overshoots. Includes effect of planet rotation to calculate inertial speed.

Returns +1 if the vehicle is captured into an orbit with the required target apoapsis alt, -1 otherwise.

Parameters

- **t_sec** (*float*) – propogation time, seconds
- **dt** (*float*) – max. time step, seconds
- **delta_deg** (*float*) – bank angle command, deg
- **targetApopasisAltitude_km** (*float*) – target apoapsis altitude , km

Returns **ans** – -1 indicates overshoot, +1 indicates undershoot

Return type int

isCaptured (*t_sec*, *dt*, *delta_deg*)

This function determines if the vehicle is captured. Returns -1 if the vehicle is captured, +1 otherwise.

Parameters

- **t_sec** (*float*) – propogation time, seconds
- **dt** (*float*) – max. time step, seconds
- **delta_deg** (*float*) – bank angle command, deg

Returns ans

Return type int

makeBasicEntryPlots()

This function creates the evolution plots of the altitude, speed, deceleration, and heat rate

Parameters **None**. –

Returns

Return type 1 image with 4 subplots

predictApoapsisAltitudeKm_afterJettision (*h0_km*, *theta0_deg*, *phi0_deg*, *v0_kms*,
gamma0_deg, *psi0_deg*, *drange0_km*,
heatLoad0, *t_sec*, *dt*, *delta_deg*, *den-
sity_mes_int*)

Compute the apoapsis altitude at exit if the drag skirt is jettisoned at the current vehicle state.

Parameters

- **h0_km** (*float*) – current vehicle altitude, km
- **theta0_deg** (*float*) – current vehicle longitude, deg
- **phi0_deg** (*float*) – current vehicle latitude, deg
- **v0_kms** (*float*) – current vehicle speed, km/s
- **gamma0_deg** (*float*) – current FPA, deg
- **psi0_deg** (*float*) – current heading angle, deg
- **drange0_km** (*float*) – current downrange, km
- **heatLoad0** (*float*) – current heatload, J/cm²
- **t_sec** (*float*) – propagation time, seconds
- **dt** (*float*) – max. solver timestep
- **delta_deg** (*float*) – commanded bank angle, deg
- **density_mes_int** (*scipy.interpolate.interpolate.interp1d*) – measured density interpolation function

Returns **terminal_apoapsis_km** – apoapsis altitude achieved if drag skirt is jettisoned at the current time, km

Return type float

```
predictApoapsisAltitudeKm_withLiftUp(h0_km, theta0_deg, phi0_deg, v0_kms,
                                         gamma0_deg, psi0_deg, drange0_km, heat-
                                         Load0, t_sec, dt, delta_deg, density_mes_int)
```

Compute apoapsis altitude using full lift up bank command from current vehicle state till atmospheric exit.

Parameters

- **h0_km** (*float*) – current vehicle altitude, km
- **theta0_deg** (*float*) – current vehicle longitude, deg
- **phi0_deg** (*float*) – current vehicle latitude, deg
- **v0_kms** (*float*) – current vehicle speed, km/s
- **gamma0_deg** (*float*) – current FPA, deg
- **psi0_deg** (*float*) – current heading angle, deg
- **drange0_km** (*float*) – current downrange, km
- **heatLoad0** (*float*) – current heatload, J/cm²
- **t_sec** (*float*) – propagation time, seconds
- **dt** (*float*) – max. solver timestep
- **delta_deg** (*float*) – commanded bank angle, deg
- **density_mes_int** (*scipy.interpolate.interpolate.interp1d*) – measured density interpolation function

```
propogateEntry(t_sec, dt, delta_deg)
```

Propogates the vehicle state for a specified time using initial conditions, vehicle properties, and atmospheric profile data.

Parameters

- **t_sec** (*float*) – propagation time, seconds
- **dt** (*float*) – max. time step, seconds
- **delta_deg** (*float*) – bank angle command, deg

propogateEntry2 (*h0_km, theta0_deg, phi0_deg, v0_kms, gamma0_deg, psi0_deg, drange0_km, heatLoad0, t_sec, dt, delta_deg, density_mes_int*)

Utility propagator routine for prediction of atmospheric exit conditions which is then supplied to the apoapis prediction module.

Propagates the vehicle state for using the measured atmospheric profile during the descending leg.

Parameters

- **h0_km** (*float*) – current altitude, km
- **theta0_deg** (*float*) – current longitude, deg
- **phi0_deg** (*float*) – current latitude, deg
- **v0_kms** (*float*) – current speed, km/s
- **gamma0_deg** (*float*) – current FPA, deg
- **psi0_deg** (*float*) – current heading angle, deg
- **drange0_km** (*float*) – current downrange, km
- **heatLoad0** (*float*) – current heatload, J/cm²
- **t_sec** (*float*) – propagation time, seconds
- **dt** (*float*) – max. time step, seconds
- **delta_deg** (*float*) – bank angle command, deg

Returns

- **t_minc** (*numpy.ndarray*) – time solution array, min
- **h_kmc** (*numpy.ndarray*) – altitude solution array, km
- **v_kmsc** (*numpy.ndarray*) – speed solution array, km/s
- **phi_degc** (*numpy.ndarray*) – latitude solution array, deg
- **psi_degc** (*numpy.ndarray*) – heading angle solution array, deg
- **theta_degc** (*numpy.ndarray*) – longitude solution array, deg
- **gamma_degc** (*numpy.ndarray*) – FPA solution array, deg
- **drange_kmc** (*numpy.ndarray*) – downrange solution array, km
- **exitflag** (*int*) – exitflag
- **acc_net_g** (*numpy.ndarray*) – acceleration solution array, Earth g
- **dyn_pres_atm** (*numpy.ndarray*) – dynamic pressure solution array, atm
- **stag_pres_atm** (*numpy.ndarray*) – stagnation pressure array, atm
- **q_stag_total** (*numpy.ndarray*) – stagnation point heat rate array
- **heatload** (*numpy.ndarray*) – stagnation point heat load
- **acc_drag_g** (*numpy.ndarray*) – acceleration due to drag, Earth g

propogateEntry2D (*h0_km, theta0_deg, phi0_deg, v0_kms, gamma0_deg, psi0_deg, drange0_km, heatLoad0, t_sec, dt, delta_deg, density_mes_int*)

Utility propogator routine for prediction of atmospheric exit conditions which is then supplied to the apoapis prediction module.

Propogates the vehicle state for using the measured atmospheric profile during the descending leg.

Parameters

- **h0_km** (*float*) – current altitude, km
- **theta0_deg** (*float*) – current longitude, deg
- **phi0_deg** (*float*) – current latitude, deg
- **v0_kms** (*float*) – current speed, km/s
- **gamma0_deg** (*float*) – current FPA, deg
- **psi0_deg** (*float*) – current heading angle, deg
- **drange0_km** (*float*) – current downrange, km
- **heatLoad0** (*float*) – current heatload, J/cm²
- **t_sec** (*float*) – propogation time, seconds
- **dt** (*float*) – max. time step, seconds
- **delta_deg** (*float*) – bank angle command, deg

Returns

- **t_minc** (*numpy.ndarray*) – time solution array, min
- **h_kmc** (*numpy.ndarray*) – altitude solution array, km
- **v_kmsec** (*numpy.ndarray*) – speed solution array, km/s
- **phi_degc** (*numpy.ndarray*) – latitude solution array, deg
- **psi_degc** (*numpy.ndarray*) – heading angle solution array, deg
- **theta_degc** (*numpy.ndarray*) – longitude solution array, deg
- **gamma_degc** (*numpy.ndarray*) – FPA solution array, deg
- **drange_kmc** (*numpy.ndarray*) – downrange solution array, km
- **exitflag** (*int*) – exitflag
- **acc_net_g** (*numpy.ndarray*) – acceleration solution array, Earth g
- **dyn_pres_atm** (*numpy.ndarray*) – dynamic pressure solution array, atm
- **stag_pres_atm** (*numpy.ndarray*) – stagnation pressure array, atm
- **q_stag_total** (*numpy.ndarray*) – stagnation point heat rate array
- **heatload** (*numpy.ndarray*) – stagnation point heat load
- **acc_drag_g** (*numpy.ndarray*) – acceleration due to drag, Earth g

propogateEntryPhased (*timeStep, dt, maxTimeSecs*)

Implements the entry phase of the guided drag modulation aerocapture (Single-event discrete drag modulation).

The entry phase is defined from the atmospheric entry interface till drag skirt jettison.

Parameters

- **timeStep** (*float*) – Guidance cycle time, seconds
- **dt** (*float*) – Solver max. time step, seconds
- **maxTimeSecs** (*float*) – max. time for propagation, seconds

propogateEquilibriumGlide (*timeStep*, *dt*, *maxTimeSecs*)

Implements the equilibrium glide phase of the guidance scheme.

Parameters

- **timeStep** (*float*) – Guidance cycle time, seconds
- **dt** (*float*) – Solver max. time step, seconds
- **maxTimeSecs** (*float*) – max. time for propagation, seconds

propogateExitPhase (*timeStep*, *dt*, *maxTimeSecs*)

Implements the exit phase of the guidance scheme (full lift-up).

Parameters

- **timeStep** (*float*) – Guidance cycle time, seconds
- **dt** (*float*) – Solver max. time step, seconds
- **maxTimeSecs** (*float*) – max. time for propagation, seconds

propogateExitPhaseD (*timeStep*, *dt*, *maxTimeSecs*)

Implements the exit phase of the guidance scheme (full lift-up).

Parameters

- **timeStep** (*float*) – Guidance cycle time, seconds
- **dt** (*float*) – Solver max. time step, seconds
- **maxTimeSecs** (*float*) – max. time for propagation, seconds

propogateGuidedEntry (*timeStep*, *dt*, *maxTimeSecs*)

Implements the full guidance scheme (eq. glide + exit phase)

Parameters

- **timeStep** (*float*) – Guidance cycle time, seconds
- **dt** (*float*) – Solver max. time step, seconds
- **maxTimeSecs** (*float*) – max. time for propagation, seconds

propogateGuidedEntryD (*timeStepEntry*, *timeStepExit*, *dt*, *maxTimeSecs*)

Implements the full guidance scheme for drag modulation aerocapture (entry phase + exit phase)

Parameters

- **timeStepEntry** (*float*) – Guidance cycle time (entry phase), seconds
- **timeStepExit** (*float*) – Guidance cycle time (exit phase), seconds
- **dt** (*float*) – Solver max. time step, seconds
- **maxTimeSecs** (*float*) – max. time for propagation, seconds

psuedoController (*DeltaCMD_deg_command*, *Delta_deg_ini*, *timestep*)

Pseudo controller implemenetation for maximum roll rate constraint.

Parameters

- **DeltaCMD_deg_command** (*float*) – commanded bank angle from guidance algorithm, deg
- **Delta_deg_ini** (*float*) – current vehicle bank angle, deg
- **maxBankRate** (*float*) – maximum allowed roll rate, deg/s
- **timestep** (*float*) – guidance cycle timestep

Returns

- **DeltaCMD_deg** (*float*) – actual bank angle response using pseudocontroller, deg
- **Delta_deg_ini** (*float*) – current bank angle, same as actual bank angle (is redundant)

qStagConvective (*r, v*)

This function defines the convective stagnation-point heating relationships. Edit the parameters in the source-code if you wish to modify these values.

Sources : Sutton-Graves relationships, NASA Neptune Orbiter with Probes Vision Report, Bienstock et al.

Parameters

- **r** (*numpy.ndarray*) – radial distance solution array of trajectory, m
- **v** (*numpy.ndarray*) – planet-relative speed array of trajectory, m/s

Returns **ans** – convective stagnation-point heating rate array, W/cm²**Return type** numpy.ndarray**qStagRadiative** (*r, v*)

This function defines the radiative stagnation-point heating relationships. Edit the parameters in the source-code if you wish to modify these values.

Radiative heating is currently set to 0 for Mars and Titan, though these may not be negligible under certain conditions.

Sources : Craig and Lyne, 2005; Brandis and Johnston, 2014; NASA Vision Neptune orbiter with probes, Contract No. NNH04CC41C

Parameters

- **r** (*numpy.ndarray*) – radial distance solution array of trajectory, m
- **v** (*numpy.ndarray*) – planet-relative speed array of trajectory, m/s

Returns **ans** – radiative stagnation-point heating rate array, W/cm²**Return type** numpy.ndarray**qStagTotal** (*r, v*)

Computes the total heat rate which is the sum of the convective and radiative heating rates.

Parameters

- **r** (*numpy.ndarray*) – radial distance solution array of trajectory, m
- **v** (*numpy.ndarray*) – planet-relative speed array of trajectory, m/s

Returns **ans** – total stagnation-point heating rate array, W/cm²**Return type** numpy.ndarray**runMonteCarlo** (*N, mainFolder*)

Run a Monte Carlo simulation for lift modulation aerocapture.

Parameters

- **N** (*int*) – Number of trajectories
- **mainFolder** (*str*) – path where data is to be stored

runMonteCarloD (*N, mainFolder*)

Run a Monte Carlo simulation for drag modulation aerocapture.

Parameters

- **N** (*int*) – Number of trajectories
- **mainFolder** (*str*) – path where data is to be stored

runMonteCarloD_Earth (*N, mainFolder*)

Run a Monte Carlo simulation for drag modulation aerocapture. (Earth application)

Parameters

- **N** (*int*) – Number of trajectories
- **mainFolder** (*str*) – path where data is to be stored

setDragEntryPhaseParams (*v_switch_kms, lowAlt_km, numPoints_lowAlt, hdot_threshold*)

Set entry phase guidance parameters for drag modulation

Parameters

- **v_switch_kms** (*float*) – speed below which entry phase is terminated
- **lowAlt_km** (*float*) – lower altitude to which density model is to be extrapolated based on available measurements, km
- **numPoints_lowAlt** (*int*) – number of points to evaluate extrapolation at below the altitude where measurements are available
- **hdot_threshold** (*float*) – threshold altitude rate (m/s) above which density measurement is terminated and apoapsis prediction is initiated

setDragModulationVehicleParams (*beta1, betaRatio*)

Set the betal and betaRatio params for a drag modulation vehicle.

Parameters

- **betal** (*float*) – small value of ballistic coefficient, kg/m²
- **betaRatio** (*float*) – ballistic coefficient ratio

setEquilibriumGlideParams (*GhDot, Gq, v_switch_kms, lowAlt_km, numPoints_lowAlt, hdot_threshold*)

Set equilibrium glide phase guidance parameters

Parameters

- **GhDot** (*float*) – Ghdot term
- **Gq** (*float*) – Gq term
- **v_switch_kms** (*float*) – speed below which eq. glide phase is terminated
- **lowAlt_km** (*float*) – lower altitude to which density model is to be extrapolated based on available measurements, km
- **numPoints_lowAlt** (*int*) – number of points to evaluate extrapolation at below the altitude where measurements are available

- **hdot_threshold** (*float*) – threshold altitude rate (m/s) above which density measurement is terminated and apoapsis prediction is initiated

setInitialState (*h0_km*, *theta0_deg*, *phi0_deg*, *v0_kms*, *psi0_deg*, *gamma0_deg*, *drange0_km*, *heatLoad0*)

Set initial vehicle state at atmospheric entry interface

Parameters

- **h0_km** (*float*) – initial vehicle altitude at atmospheric interface / other start point, meters
- **theta0_deg** (*float*) – initial vehicle longitude at atmospheric interface / other start point, degrees
- **phi0_deg** (*float*) – initial vehicle latitude at atmospheric interface / other start point, degrees
- **v0_kms** (*float*) – initial vehicle speed (planet-relative) at atmospheric interface / other start point, km/s
- **psi0_deg** (*float*) – initial vehicle heading at atmospheric interface / other start point, degrees
- **gamma0_deg** (*float*) – initial vehicle flight-path angle at atmospheric interface / other start point, degrees
- **drange0_km** (*float*) – initial vehicle downrange at atmospheric interface / other start point, km
- **heatLoad0** (*float*) – initial vehicle heatload at atmospheric interface / other start point, J/cm²

setMaxRollRate (*maxRollRate*)

Set the maximum allowed vehicle roll rate (deg/s)

Parameters **maxRollRate** (*float*) – maximum roll rate, degrees per second

setSolverParams (*tol*)

Set the solver parameters.

Parameters **tol** (*float*) – solver tolerance, currently both abstol and reltol are set to this value

setTargetOrbitParams (*target_peri_km*, *target_apo_km*, *target_apo_km_tol*)

Set the target capture orbit parameters.

Parameters

- **target_peri_km** (*float*) – target periapsis altitude, km
- **target_apo_km** (*float*) – target apoapsis altitude, km
- **target_apo_km_tol** (*float*) – target apoapsis altitude error tolerance, km used by guidance algorithm

setupMonteCarloSimulation (*NPOS*, *NMONTE*, *atmfiles*, *heightCol*, *densLowCol*, *densAvgCol*, *densHighCol*, *densTotalCol*, *heightInKmFlag*, *nominalEFPA*, *EFPA_1sigma_value*, *nominalLD*, *LD_1sigma_value*, *timeStep*, *dt*, *maxTimeSecs*)

Set the Monte Carlo simulation parameters.

Parameters

- **NPOS** (*int*) – NPOS value from GRAM model output is the number of data points (altitude) in each atm. profile
- **NMONTE** (*int*) – NMONTE is the number of Monte Carlo atm profiles from GRAM model output
- **atmfiles** (*str*) – location of atmospheric files used in Monte Carlo simulation
- **heightCol** (*int*) – column index of height values in atmfiles
- **densLowCol** (*int*) – column index of low density (-1 sigma) values in atmfiles
- **densAvgCol** (*int*) – column index of average density values in atmfiles
- **densHighCol** (*int*) – column index of high density values (+1 sigma) in atmfiles
- **densTotalCol** (*int*) – index of perturbed (=avg + pert.) density values
- **heightInKmFlag** (*bool*) – set to True if height values in atmfiles are in km
- **nominalEFPA** (*float*) – Nominal (target EFPA) value, deg
- **EFPA_1sigma_value** (*float*) – 1-sigma error for EFPA (from naviation analysis)
- **nominalLD** (*float*) – Nominal value of vehicle L/D
- **LD_1sigma_value** (*float*) – 1-sigma error for L/D (from vehicle aero. design data)
- **timeStep** (*float*) – Guidance cycle time step, sec
- **dt** (*float*) – max. solver time step
- **maxTimeSecs** (*float*) – max. time used for propogation used by guidance scheme

```
setupMonteCarloSimulationD(NPOS, NMONTE, atmfiles, heightCol, densLowCol, densAvgCol,
                           densHighCol, densTotalCol, heightInKmFlag, nominalEFPA,
                           EFPA_1sigma_value, nominalbeta1, beta1_1sigma_value,
                           timeStepEntry, timeStepExit, dt, maxTimeSecs)
```

Set the Monte Carlo simulation parameters for drag modulation aerocapture.

Parameters

- **NPOS** (*int*) – NPOS value from GRAM model output is the number of data points (altitude) in each atm. profile
- **NMONTE** (*int*) – NMONTE is the number of Monte Carlo atm profiles from GRAM model output
- **atmfiles** (*str*) – location of atmospheric files used in Monte Carlo simulation
- **heightCol** (*int*) – column index of height values in atmfiles
- **densLowCol** (*int*) – column index of low density (-1 sigma) values in atmfiles
- **densAvgCol** (*int*) – column index of average density values in atmfiles
- **densHighCol** (*int*) – column index of high density values (+1 sigma) in atmfiles
- **densTotalCol** (*int*) – index of perturbed (=avg + pert.) density values
- **heightInKmFlag** (*bool*) – set to True if height values in atmfiles are in km
- **nominalEFPA** (*float*) – Nominal (target EFPA) value, deg
- **EFPA_1sigma_value** (*float*) – 1-sigma error for EFPA (from naviation analysis)
- **nominalbeta1** (*float*) – Nominal value of vehicle ballistic coeff.

- **beta1_1sigma_value** (*float*) – 1-sigma error for beta1 (from vehicle aero. design data)
- **timeStepEntry** (*float*) – Guidance cycle time step for entry phase, sec
- **timeStepExit** (*float*) – Guidance time step for exit phase, sec
- **dt** (*float*) – max. solver time step
- **maxTimeSecs** (*float*) – max. time used for propagation used by guidance scheme

setupMonteCarloSimulationD_Earth (*NPOS*, *NMONTE*, *atmfiles*, *heightCol*, *densAvgCol*, *densSD_percCol*, *densTotalCol*, *heightInKmFlag*, *nominalEFPA*, *EFPA_1sigma_value*, *nominalbeta1*, *beta1_1sigma_value*, *timeStepEntry*, *timeStepExit*, *dt*, *maxTimeSecs*)

Set the Monte Carlo simulation parameters for drag modulation aerocapture. (Earth aerocapture)

Parameters

- **NPOS** (*int*) – NPOS value from GRAM model output is the number of data points (altitude) in each atm. profile
- **NMONTE** (*int*) – NMONTE is the number of Monte Carlo atm profiles from GRAM model output
- **atmfiles** (*str*) – location of atmospheric files used in Monte Carlo simulation
- **heightCol** (*int*) – column index of height values in atmfiles
- **densAvgCol** (*int*) – column index of average density values in atmfiles
- **densSD_percCol** (*int*) – column number of mean density one sigma SD
- **densTotalCol** (*int*) – index of perturbed (=avg + pert.) density values
- **heightInKmFlag** (*bool*) – set to True if height values in atmfiles are in km
- **nominalEFPA** (*float*) – Nominal (target EFPA) value, deg
- **EFPA_1sigma_value** (*float*) – 1-sigma error for EFPA (from navigation analysis)
- **nominalbeta1** (*float*) – Nominal value of vehicle ballistic coeff.
- **beta1_1sigma_value** (*float*) – 1-sigma error for beta1 (from vehicle aero. design data)
- **timeStepEntry** (*float*) – Guidance cycle time step for entry phase, sec
- **timeStepExit** (*float*) – Guidance time step for exit phase, sec
- **dt** (*float*) – max. solver time step
- **maxTimeSecs** (*float*) – max. time used for propagation used by guidance scheme

solveTrajectory (*rbar0*, *theta0*, *phi0*, *vbar0*, *psi0*, *gamma0*, *drangebar0*, *t_sec*, *dt*, *delta*)

Function to propagate a single atmospheric entry trajectory given entry interface / other initial conditions and bank angle delta.

Reference 1: Vinh, Chapter 3. Reference 2: Lescynski, MS Thesis, NPS.

Parameters

- **rbar0** (*float*) – non-dimensional radial distance initial condition
- **theta0** (*float*) – longitude initial condition, rad
- **phi0** (*float*) – latitude initial condition, rad

- **vbar0** (*float*) – non-dimensional planet-relative speed initial condition
- **psi0** (*float*) – heading angle initial condition, rad
- **gamma0** (*float*) – entry flight-path angle initial condition, rad
- **drangebar0** (*float*) – non-dimensional downrange initial condition
- **t_sec** (*float*) – time in seconds for which propagation is done
- **dt** (*float*) – max. time step size in seconds
- **delta** (*float*) – bank angle command, rad

Returns

- **tbar** (*numpy.ndarray*) – nondimensional time at which solution is computed
- **rbar** (*numpy.ndarray*) – nondimensional radial distance solution
- **theta** (*numpy.ndarray*) – longitude solution, rad
- **phi** (*numpy.ndarray*) – latitude array, rad
- **vbar** (*numpy.ndarray*) – nondimensional velocity solution
- **psi** (*numpy.ndarray, rad*) – heading angle solution, rad
- **gamma** (*numpy.ndarray*) – flight-path angle, rad
- **drangebar** (*numpy.ndarray*) – downrange solution, meters

truncateTrajectory (*t, r, theta, phi, v, psi, gamma, drange, index*)

This function truncates the full trajectory returned by the solver to the first event location.

The full trajectory returned by the solver could have skipped out exceeding the skip out altitude, or could have descended below the trap in altitude or even below the surface.

This function helps ensure that we truncate the trajectory to what we actually need, i.e. till a skip out / trap in event.

Parameters

- **t** (*numpy.ndarray*) – time array, sec
- **r** (*numpy.ndarray*) – radial distance array, m
- **v** (*numpy.ndarray*) – speed array, m
- **phi** (*numpy.ndarray*) – latitude array, rad
- **psi** (*numpy.ndarray*) – heading angle array, rad
- **theta** (*numpy.ndarray*) – longitude array, rad
- **gamma** (*numpy.ndarray*) – flight path angle array, rad
- **drange** (*numpy.ndarray*) – downrange array, meters
- **index** (*int*) – array index of detected event location / max index if no event detected

Returns

- **t** (*numpy.ndarray*) – truncated time array, sec
- **r** (*numpy.ndarray*) – truncated radial distance array, m
- **v** (*numpy.ndarray*) – truncated speed array, m
- **phi** (*numpy.ndarray*) – truncated latitude array, rad

- **psi** (*numpy.ndarray*) – truncated heading angle array, rad
- **theta** (*numpy.ndarray*) – truncated longitude array, rad
- **gamma** (*numpy.ndarray*) – truncated flight path angle array, rad
- **drange** (*numpy.ndarray*) – truncated downrange array, meters

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

AMAT provides mission analysis capability for aerocapture and Entry, Descent, and Landing (EDL) concepts for planetary science missions.

Copyright (C) 2021 Athul Pradeepkumar Girija.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License Version 3 as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

class AMAT.launcher.Launcher(*launcherID*)

The Launcher class is used to estimate launch vehicle performance.

ID

String identifier of planet object

Type str

XY

contains C3 in column 1, launch mass in column 2

Type numpy.ndarray

performanceQuery(*C3*)

Returns the launch capability of the vehicle for a specified C3.

Parameters **C3** (*float*) – launch C3, km²/s²

Returns **mass** – launch mass capability, kg

Return type float

**CHAPTER
SIX**

CONTRIBUTIONS

If you wish to contribute or report an issue, feel free to e-mail me [here](#) or to open an [issue](#) or create a [pull](#) request here from the [repository](#).

If you wish to make a contribution, you can do as follows:

- fork the GitHub repository
- create a feature branch from *master*
- add your feature and document it
- run the tests
- open a pull request

**CHAPTER
SEVEN**

CREDITS

AMAT was developed at the Advanced Astrodynamics Concepts (AAC) research group at Purdue University. Parts of the AMAT source code were originally developed in support of contracts between AAC and the Jet Propulsion Laboratory for various aerocapture mission studies between 2016 and 2020. Samples of atmospheric data from Global Reference Atmospheric Model (GRAM) software is used for illustration purpose only, and was developed by NASA Marshall Space Flight Center. The use of these GRAM models does not imply endorsement by NASA in any way whatsoever. A minimal working set of atmospheric profiles is included with AMAT to run the example notebooks. A minimal working interplanetary trajectory dataset is included with AMAT. The dataset was generated at Purdue University using the STOUR software package by Alec Mudek, and is also derived from trajectories published in the NASA Ice Giants Pre-Decadal Mission Study. The author plans to augment the interplanetary dataset with more publicly available information as it becomes available.

7.1 Extras

The AMAT repository includes representative atmospheric profiles for Solar System bodies, an Excel sheet with a comprehensive literature review of aerocapture, sample feasibility charts for aerocapture at all destinations, reference journal articles (by the author), a PDF version of the documentation, and the author's Ph.D. dissertation which provides more details on the methods and algorithms implemented in AMAT.

**CHAPTER
EIGHT**

REFERENCES

Results from these articles are also used as benchmark examples.

1. Craig, Scott, and James Evans Lyne. Parametric Study of Aerocapture for Missions to Venus. *Journal of Spacecraft and Rockets*, Vol. 42, No. 6, pp. 1035-1038. [craiglyne2005](#)
2. Putnam and Braun, Drag-Modulation Flight-Control System Options for Planetary Aerocapture, *Journal of Spacecraft and Rockets*, Vol. 51, No. 1, 2014. [putnamBraun2014](#)
3. Lu, Ye, and Sarag J. Saikia. Feasibility Assessment of Aerocapture for Future Titan Orbiter Missions. *Journal of Spacecraft and Rockets*, Vol. 55, No. 5, pp. 1125-1135. [luSaikia2018](#)
4. Girija, A. P., Lu, Y., & Saikia, S. J. Feasibility and Mass-Benefit Analysis of Aerocapture for Missions to Venus. *Journal of Spacecraft and Rockets*, Vol. 57, No. 1, pp. 58-73. [girijaSaikia2020a](#)
5. Girija, A. P. et al. Feasibility and Performance Analysis of Neptune Aerocapture Using Heritage Blunt-Body Aeroshells, *Journal of Spacecraft and Rockets*, Vol. 57, No. 6, pp. 1186-1203. [girijaSaikia2020b](#)

**CHAPTER
NINE**

MODULE INDEX

modindex

**CHAPTER
TEN**

INDEX

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

`AMAT.launcher`, 291
`AMAT.planet`, 243
`AMAT.vehicle`, 251

INDEX

A

a (*AMAT.vehicle.Vehicle attribute*), 251
a_n () (*AMAT.vehicle.Vehicle method*), 263
a_nbar () (*AMAT.vehicle.Vehicle method*), 263
a_nvvectorized () (*AMAT.vehicle.Vehicle method*), 263
a_s () (*AMAT.vehicle.Vehicle method*), 264
a_sbar () (*AMAT.vehicle.Vehicle method*), 264
a_svvectorized () (*AMAT.vehicle.Vehicle method*), 264
a_w () (*AMAT.vehicle.Vehicle method*), 265
a_wbar () (*AMAT.vehicle.Vehicle method*), 265
a_wvvectorized () (*AMAT.vehicle.Vehicle method*), 265
Abar (*AMAT.vehicle.Vehicle attribute*), 252
acc_drag_array (*AMAT.vehicle.Vehicle attribute*), 256
acc_drag_g (*AMAT.vehicle.Vehicle attribute*), 254
acc_net_g (*AMAT.vehicle.Vehicle attribute*), 254
acc_net_g_eg (*AMAT.vehicle.Vehicle attribute*), 257
acc_net_g_en (*AMAT.vehicle.Vehicle attribute*), 260
acc_net_g_full (*AMAT.vehicle.Vehicle attribute*), 258
acc_step_array (*AMAT.vehicle.Vehicle attribute*), 256
alpha (*AMAT.vehicle.Vehicle attribute*), 252
AMAT.launcher (*module*), 291
AMAT.planet (*module*), 243
AMAT.vehicle (*module*), 251
ATM (*AMAT.planet.Planet attribute*), 244
ATM_density (*AMAT.planet.Planet attribute*), 244
ATM_height (*AMAT.planet.Planet attribute*), 244
ATM_pressure (*AMAT.planet.Planet attribute*), 244
ATM_sonic (*AMAT.planet.Planet attribute*), 245
ATM_temp (*AMAT.planet.Planet attribute*), 244
avectorized () (*AMAT.planet.Planet method*), 245

B

beta (*AMAT.vehicle.Vehicle attribute*), 251
beta1 (*AMAT.vehicle.Vehicle attribute*), 255
betaRatio (*AMAT.vehicle.Vehicle attribute*), 255

C

CD (*AMAT.vehicle.Vehicle attribute*), 252
cfgammabar () (*AMAT.vehicle.Vehicle method*), 265
cfpsibar () (*AMAT.vehicle.Vehicle method*), 266
cfvbar () (*AMAT.vehicle.Vehicle method*), 266
checkAtmProfiles () (*AMAT.planet.Planet method*), 245
CL (*AMAT.vehicle.Vehicle attribute*), 252
classifyTrajectory () (*AMAT.vehicle.Vehicle method*), 266
cogammabar () (*AMAT.vehicle.Vehicle method*), 267
compute_apoapsis_raise_DV ()
 (*AMAT.vehicle.Vehicle method*), 272
compute_ApoapsisAltitudeKm ()
 (*AMAT.vehicle.Vehicle method*), 271
compute_periapsis_raise_DV ()
 (*AMAT.vehicle.Vehicle method*), 272
compute_PeriapsisAltitudeKm ()
 (*AMAT.vehicle.Vehicle method*), 272
computeAccelerationDrag ()
 (*AMAT.vehicle.Vehicle method*), 267
computeAccelerationLoad ()
 (*AMAT.vehicle.Vehicle method*), 267
computeAngMomScalar () (*AMAT.vehicle.Vehicle method*), 268
computeDynPres () (*AMAT.vehicle.Vehicle method*), 268
computeEccScalar () (*AMAT.vehicle.Vehicle method*), 268
computeEnergy () (*AMAT.vehicle.Vehicle method*), 268
computeEnergyScalar () (*AMAT.vehicle.Vehicle method*), 269
computeH () (*AMAT.planet.Planet method*), 245
computeHeatingForMultipleRN ()
 (*AMAT.vehicle.Vehicle method*), 269
computeMach () (*AMAT.vehicle.Vehicle method*), 269
computeR () (*AMAT.planet.Planet method*), 245
computeSemiMajorAxisScalar ()
 (*AMAT.vehicle.Vehicle method*), 269
computeStagPres () (*AMAT.vehicle.Vehicle method*), 269

computeStagTemp () *(AMAT.vehicle.Vehicle method)*, 269
 computeTCW () *(AMAT.vehicle.Vehicle method)*, 270
 computeTCW2 () *(AMAT.vehicle.Vehicle method)*, 270
 computeTCWD () *(AMAT.vehicle.Vehicle method)*, 271
 convertToKPa () *(AMAT.vehicle.Vehicle method)*, 272
 convertToPerCm2 () *(AMAT.vehicle.Vehicle method)*, 273
 convertToPlotUnits () *(AMAT.vehicle.Vehicle method)*, 273
 copsibar () *(AMAT.vehicle.Vehicle method)*, 273
 CPCV *(AMAT.planet.Planet attribute)*, 243
 createDensityMeasuredFunction ()
(AMAT.vehicle.Vehicle method), 274
 createQPlot () *(AMAT.vehicle.Vehicle method)*, 274

D

D () *(AMAT.vehicle.Vehicle method)*, 261
 Dbar () *(AMAT.vehicle.Vehicle method)*, 261
 delta_deg_array *(AMAT.vehicle.Vehicle attribute)*, 255
 densAvgCol *(AMAT.vehicle.Vehicle attribute)*, 259
 densHighCol *(AMAT.vehicle.Vehicle attribute)*, 259
 density () *(AMAT.planet.Planet method)*, 246
 density_int *(AMAT.planet.Planet attribute)*, 245
 density_mes_array *(AMAT.vehicle.Vehicle attribute)*, 256
 density_mes_int *(AMAT.vehicle.Vehicle attribute)*, 256
 densityvectorized () *(AMAT.planet.Planet method)*, 246
 densLowCol *(AMAT.vehicle.Vehicle attribute)*, 259
 densTotalCol *(AMAT.vehicle.Vehicle attribute)*, 259
 dimensionalize () *(AMAT.planet.Planet method)*, 246
 drange0_km *(AMAT.vehicle.Vehicle attribute)*, 253
 drange_km_eg *(AMAT.vehicle.Vehicle attribute)*, 257
 drange_km_en *(AMAT.vehicle.Vehicle attribute)*, 260
 drange_km_full *(AMAT.vehicle.Vehicle attribute)*, 258
 drange_kmc *(AMAT.vehicle.Vehicle attribute)*, 254
 drangec *(AMAT.vehicle.Vehicle attribute)*, 253
 dt *(AMAT.vehicle.Vehicle attribute)*, 260
 dummyVehicle () *(AMAT.vehicle.Vehicle method)*, 274
 Dvectorized () *(AMAT.vehicle.Vehicle method)*, 261
 dyn_pres_atm *(AMAT.vehicle.Vehicle attribute)*, 254
 dyn_pres_atm_eg *(AMAT.vehicle.Vehicle attribute)*, 257
 dyn_pres_atm_en *(AMAT.vehicle.Vehicle attribute)*, 261
 dyn_pres_atm_full *(AMAT.vehicle.Vehicle attribute)*, 258

E

EARTHG *(AMAT.planet.Planet attribute)*, 244
 EFPA_1sigma_value *(AMAT.vehicle.Vehicle attribute)*, 259
 EOM () *(AMAT.vehicle.Vehicle method)*, 262
 exitflag *(AMAT.vehicle.Vehicle attribute)*, 253

F

findEFPALimitD () *(AMAT.vehicle.Vehicle method)*, 274
 findOverShootLimit () *(AMAT.vehicle.Vehicle method)*, 275
 findOverShootLimit2 () *(AMAT.vehicle.Vehicle method)*, 275
 findOverShootLimitD () *(AMAT.vehicle.Vehicle method)*, 276
 findUnderShootLimit () *(AMAT.vehicle.Vehicle method)*, 276
 findUnderShootLimit2 () *(AMAT.vehicle.Vehicle method)*, 277
 findUnderShootLimitD () *(AMAT.vehicle.Vehicle method)*, 278

G

gamma0_deg *(AMAT.vehicle.Vehicle attribute)*, 252
 gamma_deg_eg *(AMAT.vehicle.Vehicle attribute)*, 257
 gamma_deg_en *(AMAT.vehicle.Vehicle attribute)*, 260
 gamma_deg_full *(AMAT.vehicle.Vehicle attribute)*, 258
 gamma_degc *(AMAT.vehicle.Vehicle attribute)*, 254
 gammac *(AMAT.vehicle.Vehicle attribute)*, 253
 Ghdot *(AMAT.vehicle.Vehicle attribute)*, 255
 GM *(AMAT.planet.Planet attribute)*, 243
 gnbar () *(AMAT.vehicle.Vehicle method)*, 278
 gphibar () *(AMAT.vehicle.Vehicle method)*, 278
 Gq *(AMAT.vehicle.Vehicle attribute)*, 255
 grbar () *(AMAT.vehicle.Vehicle method)*, 279
 gsbar () *(AMAT.vehicle.Vehicle method)*, 279
 gthetabar () *(AMAT.vehicle.Vehicle method)*, 279
 gwbar () *(AMAT.vehicle.Vehicle method)*, 279

H

h0_km *(AMAT.vehicle.Vehicle attribute)*, 252
 h_km_eg *(AMAT.vehicle.Vehicle attribute)*, 257
 h_km_en *(AMAT.vehicle.Vehicle attribute)*, 260
 h_km_full *(AMAT.vehicle.Vehicle attribute)*, 258
 h_kmc *(AMAT.vehicle.Vehicle attribute)*, 254
 h_low *(AMAT.planet.Planet attribute)*, 244
 h_skip *(AMAT.planet.Planet attribute)*, 244
 h_step_array *(AMAT.vehicle.Vehicle attribute)*, 256
 h_switch *(AMAT.vehicle.Vehicle attribute)*, 258
 h_thres *(AMAT.planet.Planet attribute)*, 244
 h_trap *(AMAT.planet.Planet attribute)*, 244

hddot_array (*AMAT.vehicle.Vehicle attribute*), 256
 hdot_array (*AMAT.vehicle.Vehicle attribute*), 256
 hdot_threshold (*AMAT.vehicle.Vehicle attribute*),
 256
 heatload (*AMAT.vehicle.Vehicle attribute*), 255
 heatLoad0 (*AMAT.vehicle.Vehicle attribute*), 253
 heatload_eg (*AMAT.vehicle.Vehicle attribute*), 257
 heatload_en (*AMAT.vehicle.Vehicle attribute*), 261
 heatload_full (*AMAT.vehicle.Vehicle attribute*), 259
 heightCol (*AMAT.vehicle.Vehicle attribute*), 259
 heightInKmFlag (*AMAT.vehicle.Vehicle attribute*),
 259
 hitsTargetApoapsis () (*AMAT.vehicle.Vehicle*
 method), 279
 hitsTargetApoapsis2 () (*AMAT.vehicle.Vehicle*
 method), 280

I

ID (*AMAT.launcher.Launcher attribute*), 291
 ID (*AMAT.planet.Planet attribute*), 243
 index (*AMAT.vehicle.Vehicle attribute*), 253
 isCaptured () (*AMAT.vehicle.Vehicle method*), 280

J

J2 (*AMAT.planet.Planet attribute*), 243
 J3 (*AMAT.planet.Planet attribute*), 244

L

L () (*AMAT.vehicle.Vehicle method*), 262
 Launcher (*class in AMAT.launcher*), 291
 Lbar () (*AMAT.vehicle.Vehicle method*), 262
 LD (*AMAT.vehicle.Vehicle attribute*), 251
 LD_1sigma_value (*AMAT.vehicle.Vehicle attribute*),
 260
 loadAtmosphereModel () (*AMAT.planet.Planet*
 method), 246
 loadAtmosphereModel5 () (*AMAT.planet.Planet*
 method), 247
 loadAtmosphereModel6 () (*AMAT.planet.Planet*
 method), 247
 loadMonteCarloDensityFile2 ()
 (*AMAT.planet.Planet method*), 247
 loadMonteCarloDensityFile3 ()
 (*AMAT.planet.Planet method*), 248
 lowAlt_km (*AMAT.vehicle.Vehicle attribute*), 256
 lvectorized () (*AMAT.vehicle.Vehicle method*), 262

M

makeBasicEntryPlots () (*AMAT.vehicle.Vehicle*
 method), 280
 mass (*AMAT.vehicle.Vehicle attribute*), 251
 maxRollRate (*AMAT.vehicle.Vehicle attribute*), 255
 maxTimeSecs (*AMAT.vehicle.Vehicle attribute*), 260
 mbar (*AMAT.vehicle.Vehicle attribute*), 252

minAlt (*AMAT.vehicle.Vehicle attribute*), 256

N

NMONTE (*AMAT.vehicle.Vehicle attribute*), 259
 nominalEFPA (*AMAT.vehicle.Vehicle attribute*), 259
 nominalLD (*AMAT.vehicle.Vehicle attribute*), 259
 nonDimState () (*AMAT.planet.Planet method*), 248
 NPOS (*AMAT.vehicle.Vehicle attribute*), 259
 nSigmaFunc () (*AMAT.planet.Planet method*), 248
 numPoints_lowAlt (*AMAT.vehicle.Vehicle*
 attribute), 256

O

OMEGA (*AMAT.planet.Planet attribute*), 243
 OMEGabar (*AMAT.planet.Planet attribute*), 244

P

p_switch (*AMAT.vehicle.Vehicle attribute*), 258
 performanceQuery () (*AMAT.launcher.Launcher*
 method), 291
 phi0_deg (*AMAT.vehicle.Vehicle attribute*), 252
 phi_deg_eg (*AMAT.vehicle.Vehicle attribute*), 257
 phi_deg_en (*AMAT.vehicle.Vehicle attribute*), 260
 phi_deg_full (*AMAT.vehicle.Vehicle attribute*), 258
 phi_degc (*AMAT.vehicle.Vehicle attribute*), 254
 phic (*AMAT.vehicle.Vehicle attribute*), 253
 Planet (*class in AMAT.planet*), 243
 planetObj (*AMAT.vehicle.Vehicle attribute*), 252
 predictApoapsisAltitudeKm_afterJettision ()
 (*AMAT.vehicle.Vehicle method*), 280
 predictApoapsisAltitudeKm_withLiftUp ()
 (*AMAT.vehicle.Vehicle method*), 281
 pressure_int (*AMAT.planet.Planet attribute*), 245
 pressurevectorized () (*AMAT.planet.Planet*
 method), 249
 presvectorized () (*AMAT.planet.Planet method*),
 249
 propogateEntry () (*AMAT.vehicle.Vehicle*
 method), 281
 propogateEntry2 () (*AMAT.vehicle.Vehicle*
 method), 282
 propogateEntry2D () (*AMAT.vehicle.Vehicle*
 method), 282
 propogateEntryPhaseD () (*AMAT.vehicle.Vehicle*
 method), 283
 propogateEquilibriumGlide ()
 (*AMAT.vehicle.Vehicle method*), 284
 propogateExitPhase () (*AMAT.vehicle.Vehicle*
 method), 284
 propogateExitPhaseD () (*AMAT.vehicle.Vehicle*
 method), 284
 propogateGuidedEntry () (*AMAT.vehicle.Vehicle*
 method), 284

propogateGuidedEntryD()
 (*AMAT.vehicle.Vehicle method*), 284
 psi0_deg (*AMAT.vehicle.Vehicle attribute*), 252
 psi_deg_eg (*AMAT.vehicle.Vehicle attribute*), 257
 psi_deg_en (*AMAT.vehicle.Vehicle attribute*), 260
 psi_deg_full (*AMAT.vehicle.Vehicle attribute*), 258
 psi_degc (*AMAT.vehicle.Vehicle attribute*), 254
 psic (*AMAT.vehicle.Vehicle attribute*), 253
 pSigmaFunc() (*AMAT.planet.Planet method*), 249
 psuedoController() (*AMAT.vehicle.Vehicle method*), 284

Q

q_array (*AMAT.vehicle.Vehicle attribute*), 256
 q_stag_con (*AMAT.vehicle.Vehicle attribute*), 254
 q_stag_rad (*AMAT.vehicle.Vehicle attribute*), 254
 q_stag_total (*AMAT.vehicle.Vehicle attribute*), 255
 q_stag_total_eg (*AMAT.vehicle.Vehicle attribute*),
 257
 q_stag_total_en (*AMAT.vehicle.Vehicle attribute*),
 261
 q_stag_total_full (*AMAT.vehicle.Vehicle attribute*), 259
 qref_array (*AMAT.vehicle.Vehicle attribute*), 256
 qStagConvective() (*AMAT.vehicle.Vehicle method*), 285
 qStagRadiative() (*AMAT.vehicle.Vehicle method*),
 285
 qStagTotal() (*AMAT.vehicle.Vehicle method*), 285

R

rbar() (*AMAT.planet.Planet method*), 249
 rc (*AMAT.vehicle.Vehicle attribute*), 253
 rho() (*AMAT.planet.Planet method*), 249
 rho0 (*AMAT.planet.Planet attribute*), 243
 rho2() (*AMAT.planet.Planet method*), 250
 rhobar() (*AMAT.planet.Planet method*), 250
 rhovectorized() (*AMAT.planet.Planet method*),
 250
 RN (*AMAT.vehicle.Vehicle attribute*), 252
 RP (*AMAT.planet.Planet attribute*), 243
 runMonteCarlo() (*AMAT.vehicle.Vehicle method*),
 285
 runMonteCarloD() (*AMAT.vehicle.Vehicle method*),
 286
 runMonteCarloD_Earth() (*AMAT.vehicle.Vehicle method*), 286

S

scaleHeight() (*AMAT.planet.Planet method*), 250
 setDragEntryPhaseParams()
 (*AMAT.vehicle.Vehicle method*), 286
 setDragModulationVehicleParams()
 (*AMAT.vehicle.Vehicle method*), 286

setEquilibriumGlideParams()
 (*AMAT.vehicle.Vehicle method*), 286
 setInitialState() (*AMAT.vehicle.Vehicle method*), 287
 setMaxRollRate() (*AMAT.vehicle.Vehicle method*),
 287
 setSolverParams() (*AMAT.vehicle.Vehicle method*), 287
 setTargetOrbitParams() (*AMAT.vehicle.Vehicle method*), 287
 setupMonteCarloSimulation()
 (*AMAT.vehicle.Vehicle method*), 287
 setupMonteCarloSimulationD()
 (*AMAT.vehicle.Vehicle method*), 288
 setupMonteCarloSimulationD_Earth()
 (*AMAT.vehicle.Vehicle method*), 289
 solveTrajectory() (*AMAT.vehicle.Vehicle method*), 289
 sonic_int (*AMAT.planet.Planet attribute*), 245
 sonicvectorized() (*AMAT.planet.Planet method*),
 251
 stag_pres_atm (*AMAT.vehicle.Vehicle attribute*), 254
 stag_pres_atm_eg (*AMAT.vehicle.Vehicle attribute*), 257
 stag_pres_atm_en (*AMAT.vehicle.Vehicle attribute*), 261
 stag_pres_atm_full (*AMAT.vehicle.Vehicle attribute*), 258

T

t_min_eg (*AMAT.vehicle.Vehicle attribute*), 256
 t_min_en (*AMAT.vehicle.Vehicle attribute*), 260
 t_min_full (*AMAT.vehicle.Vehicle attribute*), 258
 t_minc (*AMAT.vehicle.Vehicle attribute*), 254
 t_step_array (*AMAT.vehicle.Vehicle attribute*), 255
 t_switch (*AMAT.vehicle.Vehicle attribute*), 257
 target_apo_km (*AMAT.vehicle.Vehicle attribute*), 255
 target_apo_km_tol (*AMAT.vehicle.Vehicle attribute*), 255
 target_peri_km (*AMAT.vehicle.Vehicle attribute*),
 255
 tau (*AMAT.planet.Planet attribute*), 244
 tc (*AMAT.vehicle.Vehicle attribute*), 253
 temp_int (*AMAT.planet.Planet attribute*), 245
 temperaturevectorized() (*AMAT.planet.Planet method*), 251
 tempvectorized() (*AMAT.planet.Planet method*),
 251

theta0_deg (*AMAT.vehicle.Vehicle attribute*), 252
 theta_deg_eg (*AMAT.vehicle.Vehicle attribute*), 257
 theta_deg_en (*AMAT.vehicle.Vehicle attribute*), 260
 theta_deg_full (*AMAT.vehicle.Vehicle attribute*),
 258
 theta_degc (*AMAT.vehicle.Vehicle attribute*), 254

theta_c (*AMAT.vehicle.Vehicle attribute*), 253
timeStep (*AMAT.vehicle.Vehicle attribute*), 260
tol (*AMAT.vehicle.Vehicle attribute*), 253
truncateTrajectory () (*AMAT.vehicle.Vehicle method*), 290

V

v0_kms (*AMAT.vehicle.Vehicle attribute*), 252
v_kms_eg (*AMAT.vehicle.Vehicle attribute*), 257
v_kms_en (*AMAT.vehicle.Vehicle attribute*), 260
v_kms_full (*AMAT.vehicle.Vehicle attribute*), 258
v_kmsc (*AMAT.vehicle.Vehicle attribute*), 254
v_switch (*AMAT.vehicle.Vehicle attribute*), 258
v_switch_kms (*AMAT.vehicle.Vehicle attribute*), 255
vc (*AMAT.vehicle.Vehicle attribute*), 253
Vehicle (*class in AMAT.vehicle*), 251
vehicleCopy (*AMAT.vehicle.Vehicle attribute*), 260
vehicleID (*AMAT.vehicle.Vehicle attribute*), 251
Vref (*AMAT.planet.Planet attribute*), 244

X

XY (*AMAT.launcher.Launcher attribute*), 291