TASK 1

define_edl_system.py

- Note: we are starting the simulation from the point at which the parachute is first deployed to keep things simpler.

parachute

- Deployed at an altitude of 11 km. Ejected at an altitude of 900 meters. Rockets continue to fire through this event

Key	Type	Default value	Units	Description
deployed	boolean	True	-	True means the parachute has been deployed but not ejected
ejected	boolean	False	-	True means the parachute is no longer attached to the system
diameter	int	16.25	[m]	The diameter of the parachute
Cd	int	0.615	-	The coefficient of drag at sub-Mach speeds (nominal for subsonic)
mass	int	185.0	[kg]	Mass of the parachute (wild guess of 185 kg -- no data was found)

rocket

- defines a SINGLE rocket

Key	Type	Default value	Units	Description
on	boolean	False	-	True means the rocket is on
structure_mass	int	8.0	[kg]	Mass of the structure, not including the fuel
initial_fuel_mass	int	230.0	[kg]	The mass of the fuel initially
fuel_mass	int	230.0	[kg]	The current fuel mass will be less than equal to the initial fuel mass
effective_exhaust_velocity	int	4500.0	[m/s]	The effective exhaust velocity of the gasses (times the time rate of change of EDL system mass due to the expelled gases, equals the equivalent force created by the thrus).
max_thrust	int	3100.0	[N]	Max thrust the rocket endures
min_thrust	int	40.0	[N]	Min thrust the rocket endures

speed_control

- functions to keep the craft descending at a fixed speed. Engages when the EDL is traveling at three times the targeted speed. The net effect of the speed controller is to modulate the thrust output from the rockets)

Key	Type	Default value	Units	Description
on	boolean	False	-	indicates whether control mode is activated
Kp	int	2000	-	proportional gain term
Kd	int	20	-	derivative gain term
Ki	int	50	-	integral gain term
target_velocity	int	-3.0	[m/s]	desired descent speed

position_control

- Used to command the craft to hover at an altitude appropriate for the sky crane operation, which is 7.6 meters. Engages when the craft is about 9.1 meters in altitude. Affects the thrust output of the rockets)

Key	Type	Default value	Units	Description
on	boolean	False	-	indicates whether control mode is activated
Kp	int	2000	-	proportional gain term
Kd	int	1000	-	derivative gain term
Ki	int	50	-	integral gain term
target_altitude	int	7.6	[m]	needs to reflect the sky crane cable length

sky_crane

- Once the EDL system reaches an altitude of 7.6 meters, it proceeds to lower the rover using the sky crane. Rover is lowered via a cable at a constant speed. Sufficient to model the rover being lowered at a constant speed relative to the sky crane. Once rover touches ground, sky crane flies away)

Key	Type	Default value	Units	Description
on	boolean	False	-	true means lowering rover mode
danger_altitude	int	4.5	[m]	altitude at which considered too low for safe rover touch down
danger_speed	int	-1.0	[m/s]	speed at which rover would impact to hard on surface
mass	int	35.0	[kg]	Mass of sky crane
area	int	16.0	[m^2]	frontal area for drag calculations
Cd	int	0.9	[-]	coefficient of drag
max_cable	int	7.6	[m]	max length of cable for lowering rover
velocity	int	-0.1	[m]	speed at which sky crane lowers rover

heat_shield

- Ejected at an altitude of 8 km. Does not affect the descent dynamics in an appreciable way. Mass is reduced and the drag coefficient is changed due to ejecting the heat shield)

Key	Default value	Units	Description
ejected	False	-	true means heat shield has been ejected from system
mass	225.0	[kg]	mass of heat shield
diameter	4.5	[m]	Diameter of heat shield
Cd	0.35	[-]	Coefficient of heat shield drag

edl_system

Key	Type	Value	Description
altitude	Numpy variable	np.NaN	system state variable that is updated throughout simulation
velocity	Numpy variable	np.NaN	system state variable that is updated throughout simulation
num_rockets	int	8	system level parameter
volume	int	150	system level parameter
parachute	dict	parachute	The parachute dictionary
heat_shield	dict	heat_shield	The heat shield dictionary
rocket	dict	rocket	The rocket dictionary
speed_control	dict	speed_control	The speed control dictionary
position_control	dict	position _control	The position control dictionary
sky_crane	dict	sky_crane	The sky crane dictionary
rover	dict	rover	The rover dictionary

define_mission_events.py

mission_events

Key	Type	Default value	Units	Description
alt_heatshield_eject	scalar	8000	[m]	Altitude at which the Marvin's heat shield is ejected [m]
alt_parachute_eject	scalar	900	[m]	Altitude at which the Marvin's parachute is ejected [m]
alt_rockets_on	scalar	1800	[m]	Altitude at which the Marvin's variable-thrust solid rockets are initiated [m]
alt_skycrane_on	scalar	7.6	[m]	Altitude at which the Marvin's sky crane is initiated to lower the rover [m]

define_planet.py

high_altitude

Key	Type	Default value	Units	Description
temperature	Function	lambda altitude: -23.4 - 0.00222*altitude	[C]	Temperature as a function of altitude for high altitude scenarios [C]
pressure	Function	lambda altitude: 0.699*np.exp(-0.00009*altitude)	[KPa]	Pressure as a function of altitude for high altitude scenarios [KPa]

low_altitude

Key	Type	Default value	Units	Description
temperature	Function	lambda altitude: -31 - 0.000998*altitude	[C]	Temperature as a function of altitude for low altitude scenarios [C]
pressure	Function	lambda altitude: 0.699*np.exp(-0.00009*altitude)	[KPa]	Pressure as a function of altitude for low altitude scenarios [KPa]

density

Key	Type	Default value	Units	Description
density	Function	lambda temperature, pressure: pressure/(0.1921*(temperature+273.15))	[kg/m^3]	Density as a function of temperature and pressure [kg/m^3]

mars

Key	Type	Default value	Units	Description
g	scalar	-3.72	[m/s^2]	Acceleration due to gravity [m/s^2]
altitude_threshold	scalar	7000	[m]	The altitude threshold of mars [m]
low_altitude	dict	low_altitude	-	The low altitude dict
high_altitude	dict	high_altitude	-	The high altitude dict
density	dict	density	-	The density dict

define_rovers.py

wheel

Key	Type	Default value	Units	Description
radius	scalar	0.20	[m]	Radius of the wheel [m]
mass	scalar	2	[Kg]	Weight of the wheel [kg]

speed_reducer

Key	Type	Default value	Units	Description
type	string	reverted	-	String of text defining the type of speed reducer. For Project Phase 1, the only valid entry is “reverted”.
diam_pinion	scalar	0.04	[m]	Diameter of pinion [m]
diam_gear	scalar	0.06	[m]	Diameter of gear [m]
mass	scalar	1.5	[Kg]	Mass of speed reducer assembly [kg]

motor

Key	Type	Default value	Units	Description
torque_stall	scalar	165	[N-m]	Motor stall torque [N-m]
torque_noload	scalar	0	[N-m]	Motor no-load torque [N-m]
speed_noload	scalar	3.85	[rad/s]	Motor no-load speed [rad/s]
mass	scalar	5.0	[Kg]	Motor mass [kg]
effcy_tau	1D Numpy array	[0, 10, 20, 40, 75, 170]	[N-m]	N-element array of torque data points at which efficiency data is gathered [N-m]
effcy	1D Numpy array	[0, .60, .75, .73, .55, .05]	-	N-element array of efficiency measurements corresponding to torque data [-]

chassis

Key	Type	Default value	Units	Description
mass	scalar	674	[kg]	Mass of chassis [kg]

science_payload

Key	Type	Default value	Units	Description
mass	scalar	80	[kg]	Mass of payload [kg]

power_subsys

Key	Type	Default value	Units	Description
mass	scalar	100	[kg]	Mass of power subsystem [kg]

wheel_assembly

Key	Type	Default value	Units	Description
wheel	dict	wheel	-	The wheel dict
speed_reducer	dict	speed_reducer	-	The speed reducer dict
motor	dict	motor	-	The motor dict

rover

Key	Type	Default value	Units	Description
wheel_assembly	dict	wheel_assembly	-	Wheel assembly dictionary.
chassis	dict	chassis	-	The chassis dictionary.
science_payload	dict	science_payload	-	The science payload dictionary.
power_subsys	dict	power_subsys	-	The power subsystem dictionary.

planet

Key	Type	Default value	Units	Description
g	scalar	3.72	[m/s^s]	Acceleration due to gravity [m/s^2]

end_event

Key	Type	Default value	Units	Description
max_distance	scalar	10	[m]	Max distance the rover travels [m]
max_time	scalar	10000	[s]	Maximum time for the simulation by the ODE solver [s]
min_velocity	scalar	0.01	[m/s]	Minimum velocity necessary to halt the solution of the differential equation. This condition is necessary in case the rover gets 'stuck'. [m/s]

TASK 2

get_mass_rover

get_mass_rover		
General Description		
Computes the mass of the rover defined in the rover field of the edl system struct. Assumes that the rover is defined as a dict corresponding to the specification of project Phase 1.		
Calling syntax		
m = get_mass_rover(edl_system)		
Input Arguments		
edl_system	dict	Data structure containing edl system definition
Return Arguments		
m	scalar float/int	Mass of the rover [kg]
Additional Specifications and Notes		
• Mass is multiplied by 6 to account for the 6 wheels		

get_mass_rockets

get_mass_rockets		
General Description		
Returns the current total mass of all rockets on the edl system.		
Calling syntax		
m = get_mass_rockets(edl_system)		
Input Arguments		
edl_system	dict	Data structure containing edl system definition
Return Arguments		
m	scalar float/int	Mass [kg] but changes based on initial dict values

get_mass_edl

get_mass_edl		
General Description		
Returns the total current mass of the edl system		
Calling syntax		
m = get_mass_edl(edl_system)		
Input Arguments		
edl_system	dict	Data structure containing edl system definition
Return Arguments		
m	scalar float/int	Mass of the total edl system [kg]
Additional Specifications and Notes		
• This function calls **get_mass_rockets()** and **get_mass_rover()**		

get_local_atm_properties

get_local_atm_properties		
General Description		
Returns local atmospheric properties at a given altitude.		
Calling syntax		
density, _,_ = get_local_atm_properties(planet, altitude) _, temperature, _ = get_local_atm_properties(planet, altitude) _, _, pressure = get_local_atm_properties(planet, altitude)		
Input Arguments		
planet	dict	Data structure containing mars' low and high and altitude definition
altitude	scalar float/int	Given altitude [m]
Return Arguments		
density	scalar float/int	returns the atmospheric density in [kg/m^3]. Assumed altitude is specified in meters.
temperature	scalar float/int	returns local temp [C]
pressure	scalar float/int	returns local pressure [KPa]
Additional Specifications and Notes		
• This function is NOT vectorized. It will not accept a vector of altitudes.		

F_bouyancy_decent

F_bouyancy_descent		
General Description		
Computes the net buoyancy force.		
Calling syntax		
F = F_buoyancy_descent(edl_system, planet, altitude):		
Input Arguments		
edl_system	Dictionary	Data structure containing edl system definition
planet	Dictionary	Data structure containing mars' high and low altitude definition
altitude	scalar float/int	Given altitude [m]
Return Arguments		
F	scalar float/int	returns the net buoyancy force in [N]

F_drag_decent

F_drag_descent		
General Description		
Computes the net drag force by determining which part(s) of the EDL system are contributing to drag.		
Calling syntax		
F = F_drag_descent(edl_system, planet, altitude, velocity):		
Input Arguments		
edl_system	Dictionary	Data structure containing edl system definition
planet	Dictionary	Data structure containing mars' high and low altitude definition
altitude	scalar float/int	Given altitude [m]
velocity	scalar float/int	Given velocity [m/s]
Return Arguments		
F	scalar float/int	returns the net buoyancy force in [N]
Additional Specifications and Notes		
• This function should call **get_local_atm_properties** • This function assumes that the coefficient of drag for the parachute is independent of the speed of descent.		

F_gravity_decent

F_gravity_descent		
General Description		
Computes the gravitational force acting on the EDL system		
Calling syntax		
F = F_gravity_descent(edl_system, planet):		
Input Arguments		
edl_system	Dictionary	Data structure containing high altitude definition
planet	Dictionary	Data structure containing gravity on mars
Return Arguments		
F	scalar float/int	returns the gravitational force in [N].
Additional Specifications and Notes		
• This function should call **get_mass_edl**		

v2M_Mars

v2M_Mars		
General Description		
Converts descent speed, v [m/s], to Mach number on Mars as a function of altitude, a [m]. Then, it returns only the absolute value Mach number (i.e., uses model M = abs(v)/v_sound))).		
Calling syntax		
M = v2M_Mars(v, a):		
Input Arguments		
v	scalar float/int	decent speed [m/s]
a	scalar float/int	altitude [m]
Return Arguments		
M	scalar float/int	absolute value of Mach number
Additional Specifications and Notes		
• This function uses a table defining the speed of sound vs. altitude on mars		

thrust_controller

thrust_controller		
General Description		
This function implements a PID Controller for the EDL system. Uses edl_system and planet structs to create a modified edl_system struct. Modifies fields in rocket and telemetry substructs.		
Calling syntax		
edl_system = thrust_controller(edl_system,planet)		
Input Arguments		
edl_system	struct	Data structure containing edl system definition
planet	struct	Data structure containing gravity on mars definition
Return Arguments		
edl_system	struct	Data structure containing modified rocket and telemetry substrucrs in edl system definition
Additional Specifications and Notes		
• This function should validate that the first and second input is a dictionary. If any condition fails, call `raise`		

```
        Exception()
```

edl_events

edl_events		
General Description		
Defines events that occur in EDL System simulation. There are 9 different scenarios listed below 0. Reached altitude to eject heat shield 1. Reached altitude to eject parachute 2. Reached altitude to turn on rockets 3. Reached altitude to turn on crane & altitude control 4. Out of fuel --> y(3)<=0. Terminal. Direction: -1. 5. EDL System crashed at zero altitude 6. Reached speed at which speed-controlled descent is required 7. Reached position at which altitude control is required 8. Rover has touched down on surface of Mars		
Calling syntax		
events = edl_events(edl_system, mission_events):		
Input Arguments		
edl_system	dict	Data structure containing edl system definition
mission_events	dict	Data structure containing mission events definition
Return Arguments		
events	list	List containing a list of the function outcomes of the events occuring in the EDL
Additional Specifications and Notes		
• This function should return the solution to each scenario		

edl_dynamics

edl_dynamcis		
General Description		
Dynamics of EDL as it descends and lowers the rover to the surface.		
Calling syntax		
dydt = edl_events(edl_system, mission_events):		
Input Arguments		
t	numpy.ndarray	tive variabel
y	state vector	State vector containing vel_edl, fuel_mass, ei_vel, ei_pos, vel_rov, pos_rov
edl_system	dict	Data structure containing edl system definition
planet	dict	Data structure containing planet definition
Return Arguments		
dydt	1D numpy array	N-element array containing dy1dt (acceleration), dy2dt (velocity), dmdt (Change in total mass of rockets due to propellant being expelled to produce thrust), e_vel (error integral for velocity error), e_pos (error integral for position (altitude) error), dy6dt (rover acceleration relative to sky crane), dy7dt (rover velocity relative to sky crane)
Additional Specifications and Notes		

- y = [vel_edl; pos_edl; fuel_mass; ei_vel; ei_pos; vel_rov; pos_rov]
  - vel_edl = [m/s] velocity of EDL system
  - altitude_edl = [m] altitude of EDL system
  - fuel_mass = [kg] total mass of fuel in EDL system
  - ei_vel = [m/s] error integral for velocity error
  - ei_pos = [m] error integral for position (altitude) error
  - vel_rov = [m/s] velocity of rover relative to sky crane
  - pos_rov = [m/s] velocity of rover relative to sky crane
- ydot = [accel_edl; vel_edl; dmdt; e_vel; e_pos; accel_rov; vel_rov]
- This function should call **get_mass_edl, F_gravity_descent, F_buoyancy_descent**, and **F_drag_descent/F_drag_descent_mod**
- Specifications: edl altitude, velocity and acceleration are absolute, rov is relative to edl, fuel_mass is total over all rockets
- Note: this is a VARIABLE MASS SYSTEM, which means Newton's second law expressed as F=ma cannot be used. The more fundamental relationship is F = dp/dt, where p is the momentum of the system. (For a particle, p=mv and if m is constant you recover the F=ma form easily.) It is important to consider the momentum contributions of the EDL system and propellant being expelled from the rocket. Ultimately you end up with something that roughly resembles Newton's second law: F_ext + v_rel*(dm/dt) = ma where m is the mass of the EDL, dm/dt is the rate at which mass is changing (due to propellant being expelled), v_rel is the speed at which propellant is being expelled, a is the acceleration of the EDL and F_ext is the sum of other forces acting on the EDL (drag, buoyancy, gravity). Since v_rel*(dm/dt) is a force, we can write this as F_ext + F_thrust = ma, which is very Newton-like.

update_edl_state

update_edl_state		
General Description		
Update status of EDL System based on simulation events		
Calling syntax		
edl_system = update_edl_state(edl_system, TE, YE, Y, ITER_INFO) y0 = update_edl_state(edl_system, TE, YE, Y, ITER_INFO): TERMINATE_SIM = update_edl_state(edl_system, TE, YE, Y, ITER_INFO):		
Input Arguments		
edl_system	dict	Data structure containing all edl system definition
TE	list	sol.t_events form IVT of edl_dynamics
YE	list	sol.y_events form IVT of edl_dynamics
Y	numpy.ndarray	y solutions of IVP of edl_dynamics
ITER_INFO	boo1	Parameter that sets simulation status
Return Arguments		
edl_system	dict	modified system variables
y0	1D numpy array	initial conditions
TERMINATE_SIM	bool	value to end the simulation
Additional Specifications and Notes		
simulation events 0. Reached altitude to eject heat shield 1. Reached altitude to eject parachute 2. Reached altitude to turn on rockets 3. Reached altitude to turn on crane 4. Out of fuel --> y(3)<=0. Terminal. Direction: -1. 5. EDL System crashed at zero altitude 6. Reached speed at which controlled descent is required 7. Reached altitude at which position control is required 8. Rover has touched down on surface of Mars This also updates the rocket mass (due to fuel having been expelled).		

simulate_edl

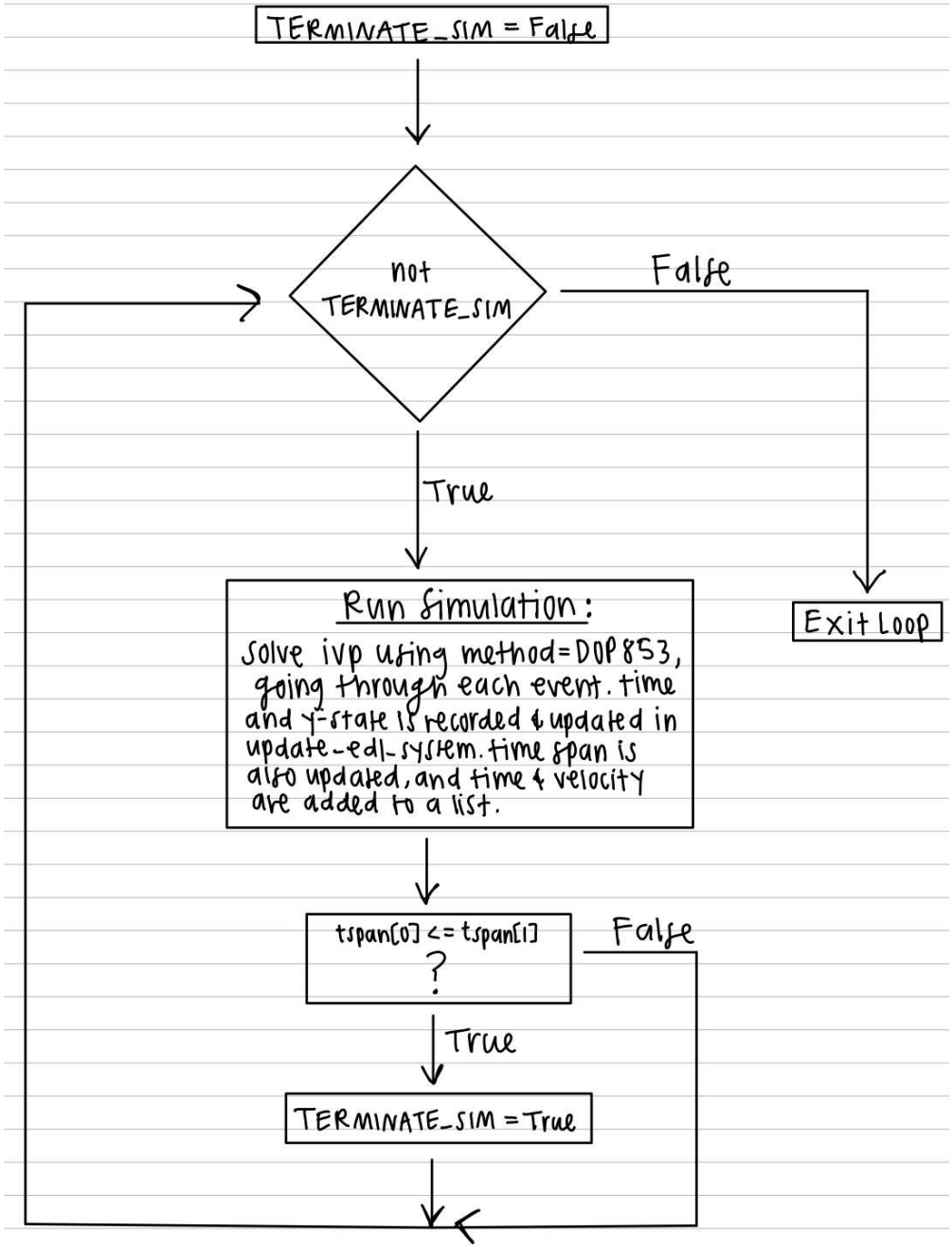
simulate_edl		
General Description		
This simulates the EDL system. It requires a definition of the edl_system, the planet, the mission events, a maximum simulation time and has an optional flag to display detailed iteration information.		
Calling syntax		
T = simulate_edl(edl_system, planet, mission_events, tmax, ITER_INFO) Y = simulate_edl(edl_system, planet, mission_events, tmax, ITER_INFO) edl_system = simulate_edl(edl_system, planet, mission_events, tmax, ITER_INFO)		
Input Arguments		
ITER_INFO	bool	Parameter that sets simulation status
tmax	float	Number that sets the maximum time
mission_events	dict	Data structure containing mission events definition
edl_system	dict	Data structure containing all edl system definition
planet	dict	Data structure containing planet definition
Return Arguments		
T	numpy.ndarray	array with time data for simulation
Y	numpy.ndarray	multi dimensional array with different simulation data [vel, alt, fuel mass, speed error, pos error, vel rov to crane, pos rov to crane]
edl_system	dict	dictionary of system state
Additional Specifications and Notes		

TASK 3

The while loop that is located in the subfunctions.py code under the function called simulate_edl. This while loop uses the ordinary differential equation (ODE) solver DOP853, which is an explicit

Runge-Kutta method of order 8, and calls solve_ivp to produce the results for the initial conditions. The time and y state is then recorded within the loop, and is updated using the update_edl_state function. The time and velocity are added to a list. The while loop will then run until an event triggers the activation of the termination status. This status is determined by the function update_edl_state which contains multiple cases that activate the termination status. The cases that activate the termination of the EDL system include: the EDL crashing before the sky crane is activated, the rocket running out of fuel, the rover altitude being equal to or greater than the sky crane danger altitude and the rover touchdown speed is greater than the sky crane danger speed. If any one of these previous conditions is met, the while loop will stop and return the y-list, time list, and update the edl_system dictionary. A flowchart of the event can be seen below entitled “While Loop Flowchart.”

While Loop Flowchart - Task 3


TERMINATE_SIM = False
not TERMINATE_SIM
False
True
Exit Loop
Run Simulation:
Solve ivp using method=DOP853, going through each event. time and y-state is recorded & updated in update-edl-system. time span is also updated, and time & velocity are added to a list.
tspan[0] <= tspan[1] ?
False
True
TERMINATE_SIM = True

TASK 4

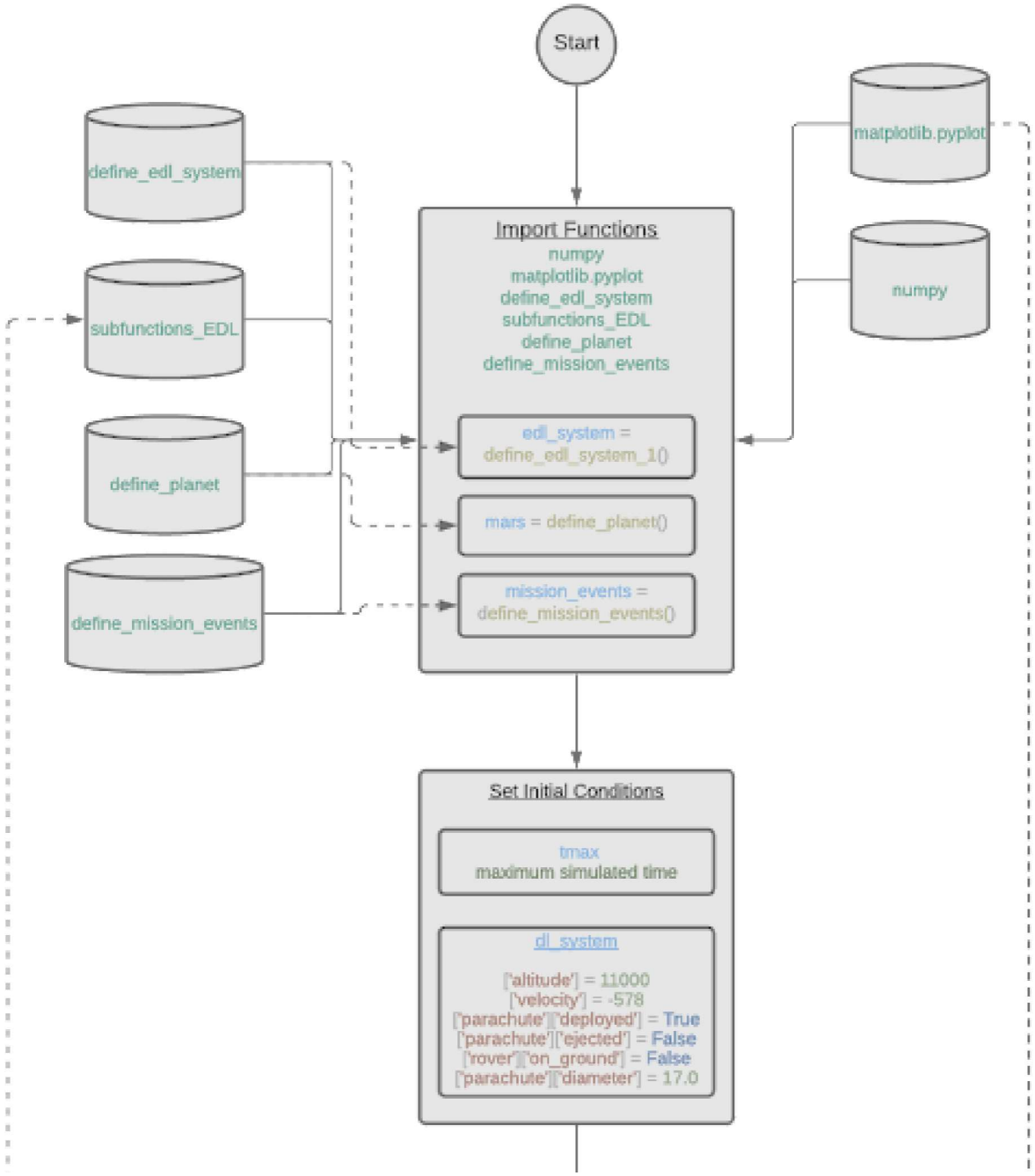
script main_edl_simulation.py has 5 dependencies : matplotlib.pyplot, define_edl_system, define_planet, define_mission_events ,subfunctions_EDL. numpy is also imported but does not need to be imported for the function to run.

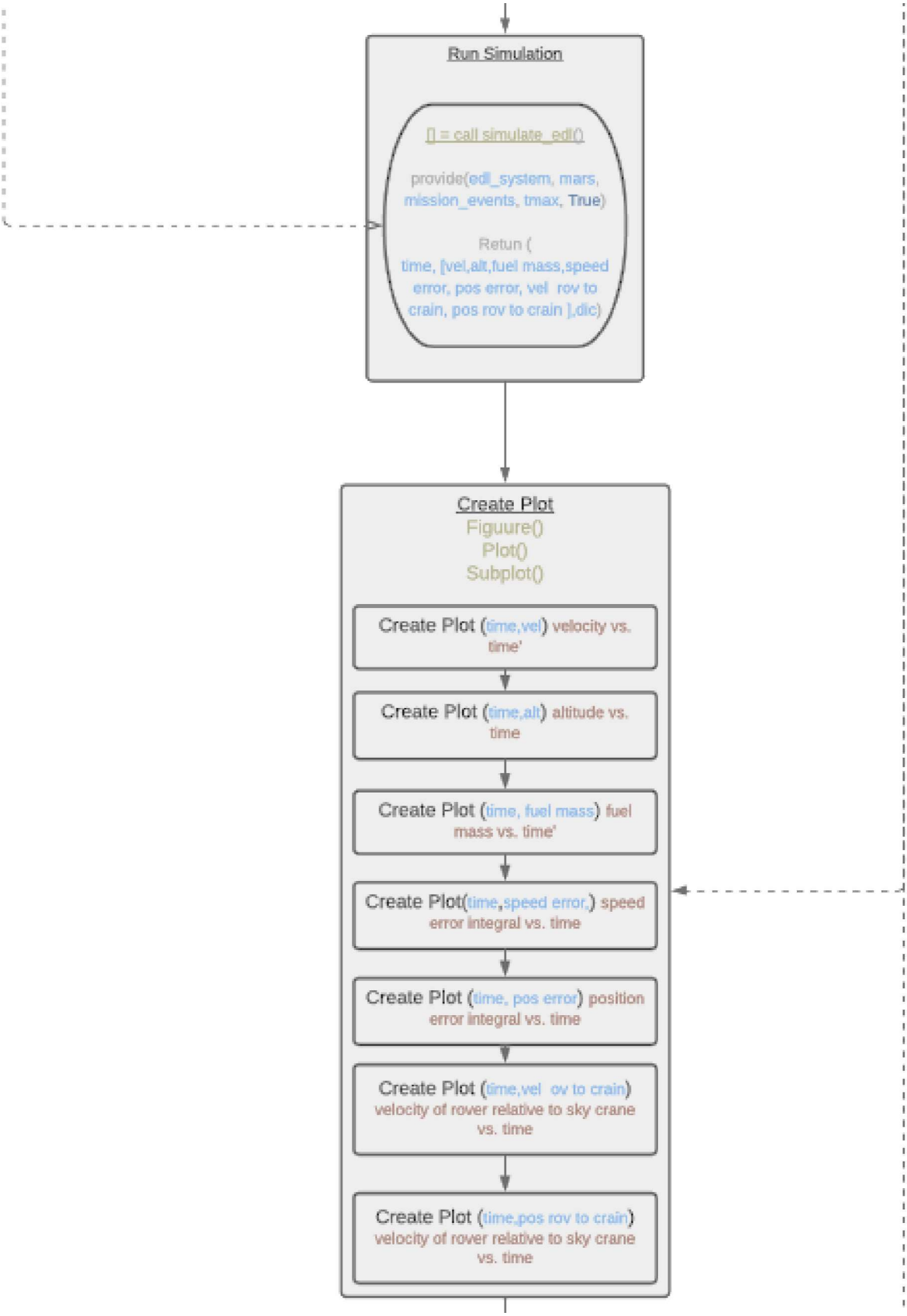
script main_edl_simulation.py relies on the matplotlib library to create all graphs used in the code. Without this the graphs would not be able to be displayed and the python interpreter would return an error when it compiles lines that use matplotlib data types like plot and figures.

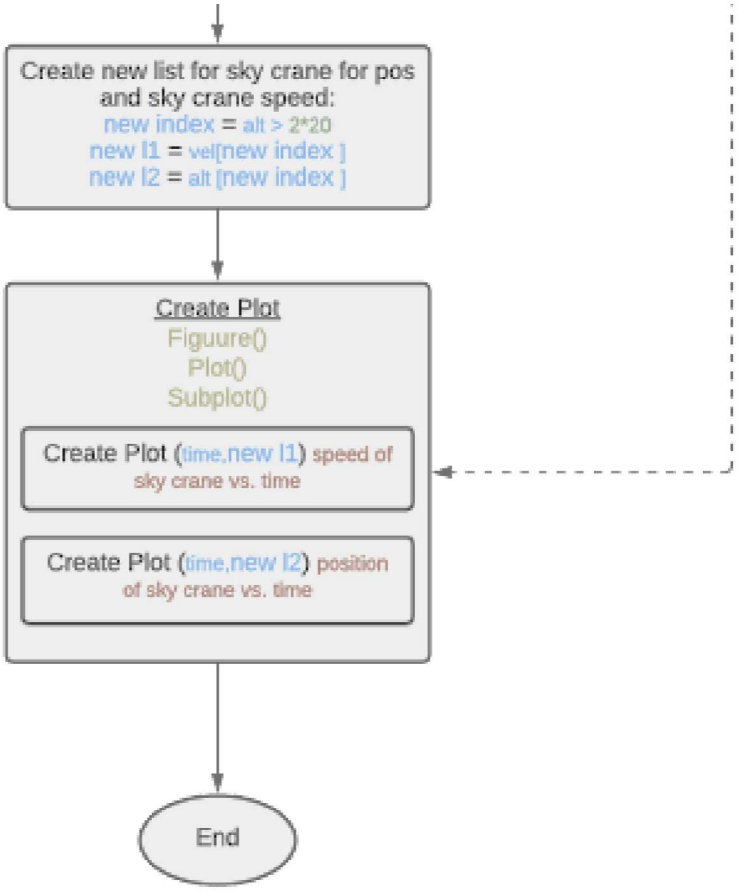
Define_edl_system, define_planet, define_mission_events are all dictionaries that hold critical information used in the simulations. Define_edl_system is a central dictionary that holds all parameters for the rover and crane system. Define_planet holds information like the gravitational force to the planet that is required to accurately run the simulation. And define_mission_events is a dictionary of different critical mission variables .

Lastly, subsfuniton_EDL contains the function simulate_edl which is used to collect all data used in the graph and run each simulation.

See flow chart below: dashed arrows are depencies after being imported. .


Start
define_edl_system
subfunctions_EDL
define_planet
define_mission_events
matplotlib.pyplot
numpy
Import Functions
numpy
matplotlib.pyplot
define_edl_system
subfunctions_EDL
define_planet
define_mission_events
edl_system =
define_edl_system_1()
mars = define_planet()
mission_events =
define_mission_events()
Set Initial Conditions
tmax
maximum simulated time
dl_system
['altitude'] = 11000
['velocity'] = -578
['parachute']['deployed'] = True
['parachute']['ejected'] = False
['rover']['on_ground'] = False
['parachute']['diameter'] = 17.0


Run Simulation
[] = call simulate_edl()
provide(edl_system, mars, mission_events, tmax, True)
Retun (
time, [vel,alt,fuel mass,speed error, pos error, vel rov to crain, pos rov to crain],dic)
Create Plot
Figuure()
Plot()
Subplot()
Create Plot (time,vel) velocity vs. time'
Create Plot (time,alt) altitude vs. time
Create Plot (time, fuel mass) fuel mass vs. time'
Create Plot(time,speed error,) speed error integral vs. time
Create Plot (time, pos error) position error integral vs. time
Create Plot (time,vel ov to crain) velocity of rover relative to sky crane vs. time
Create Plot (time,pos rov to crain) velocity of rover relative to sky crane vs. time


Create new list for sky crane for pos and sky crane speed:
new index = alt > 2*20
new l1 = vel[new index]
new l2 = alt [new index]
Create Plot
Figuure()
Plot()
Subplot()
Create Plot (time,new l1) speed of sky crane vs. time
Create Plot (time,new l2) position of sky crane vs. time
End

TASK 5

Looking at the third graph in the figure below we immediately see the rover only has a successful landing when the diameter of the graph is 16.5 m or larger. This is based on if the rover landed, and the skyward did not malfunction due to it reaching a dangerous speed or altitude. The rover speed at this range when looking at graph 2 in the below figure is constant at around 7m/s. If we want to minimize the amount of time it takes to get to the ground , based on the first figure we know we should use the 16.5 [m] parachute as it is only 200 [s] for the rover to get to the ground. If we want to slow the rover's descent we can increase the parachute diameter which on the graph in the figure below shows an almost linear trend from 16.5 [m] to the last size simulated 19 [m] diameter parachute. To minimize speed and land safely a parachute of 16.5 [m] should be used. To get a more precise diameter we can also run the simulation between 16 -16.5 [m] to determine if there is an intermediary parachute shoot that would further decrease the descent time while letting the rover land safely which can be seen if figure 6.2.

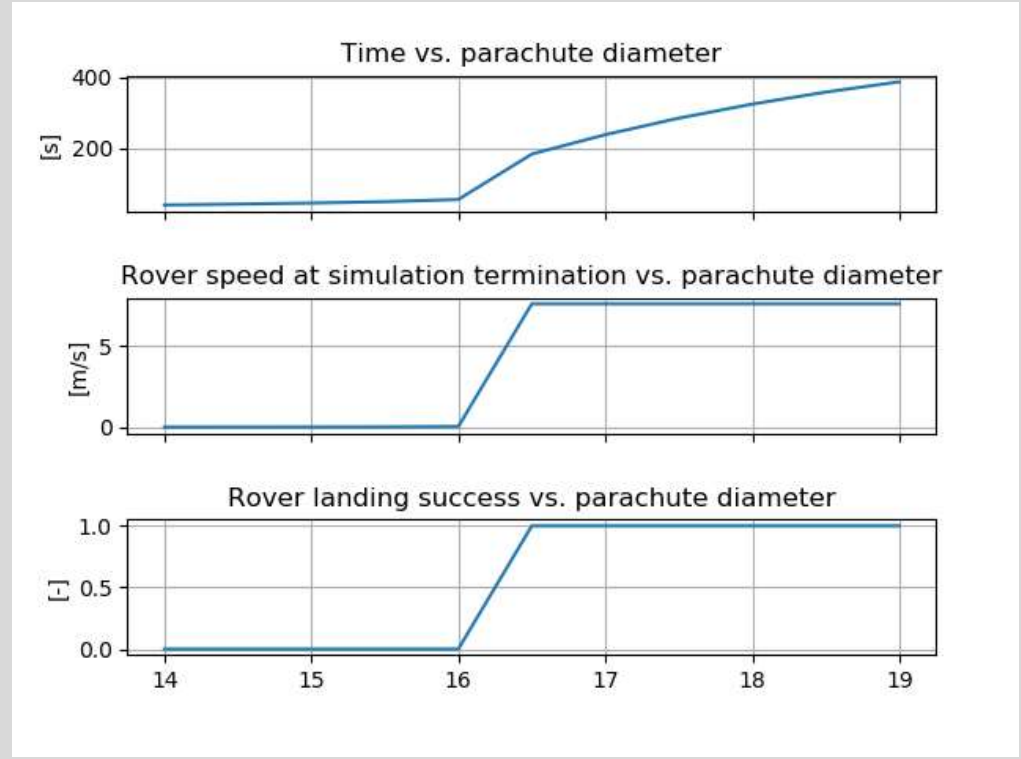


Figure 6.1 : Parachute times 14 - 19 with a 0.5 step size

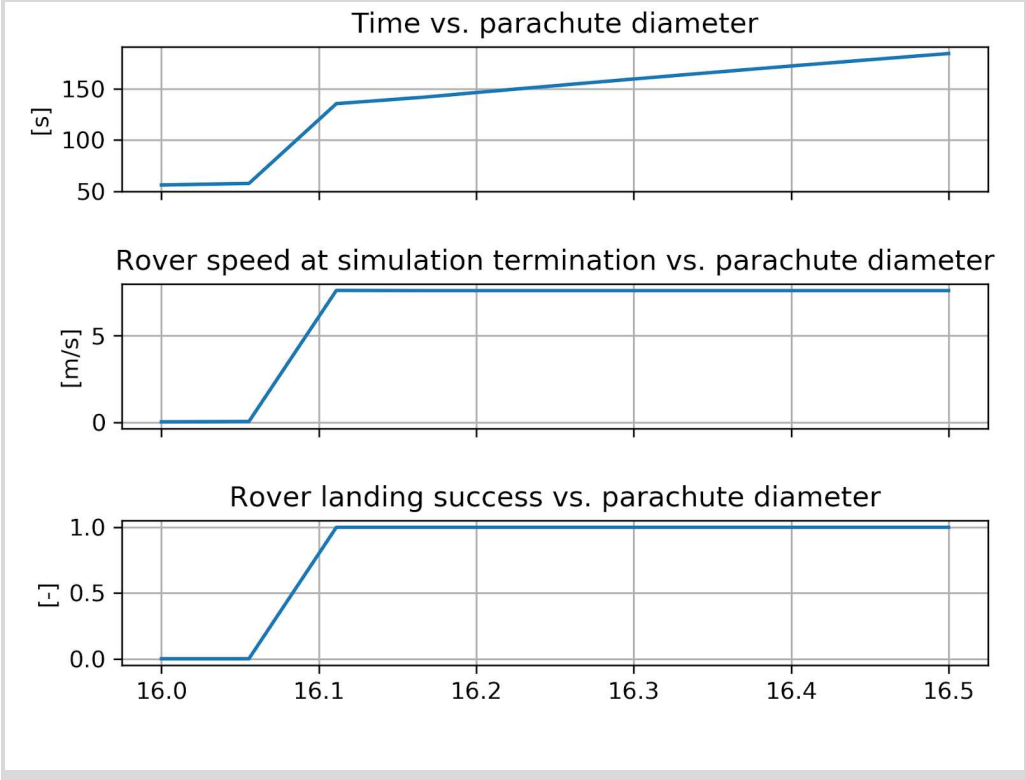
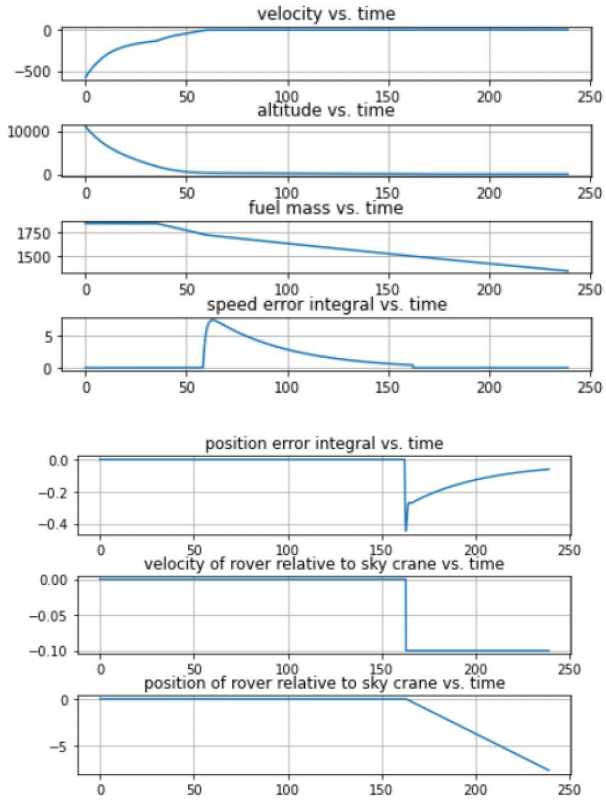


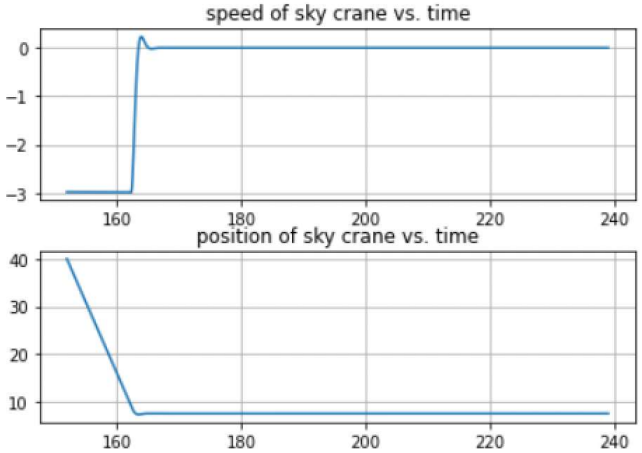
Figure 6.2 : Parachute times 16 - 16.5 with 10 steps

TASK 6

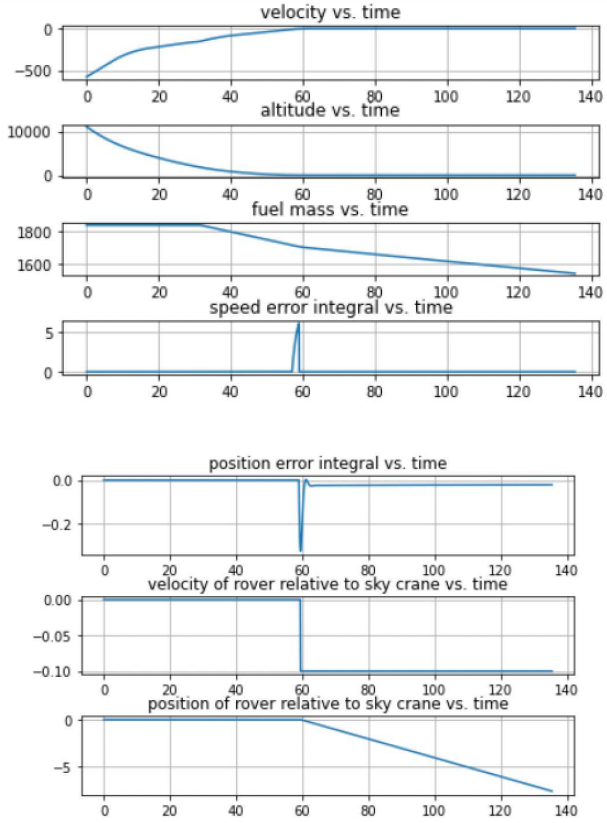
For Task 6, we were asked to create a modified F_drag_descent function, this time accounting for a changing Cd because of its dependency on decent speed. The decent speed first had to be converted to a Mach number, seeing that Cd was dependent on the Mach efficiency factor, MEF, as a function of Mach numbers. First we used the v2M_Mars function to convert velocity to Mach. Then, given a Mach vs MEF chart, we interpolated it to find a corresponding MEF to Mach. We could then use that number to find a modified Cd factor. Below shows the comparison of results with the original function and the modified function.

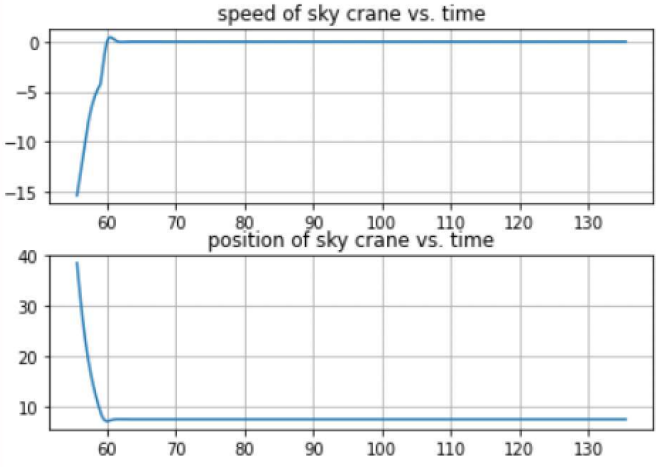
Graphs with original F_drag_descent() function:





Graphs with modified F_drag_descent_mod() function:





Comparing the two sets of graphs above, it is evident that modifying the drag force to be dependent on the rover descent velocity has an overall effect on the landing of the rover. It seems that the rover is descending at a much quicker rate with the modified drag force than with the original drag force. This creates a need to adjust the rover parachute to be within the mission success criteria shown below.

1. The rover speed at landing is no more than 1.0 m/s.
2. The sky crane altitude is no less than 4.5 m.

After running tests with the modified drag function using different diameters, we found that the diameter of the parachute needs to be increased to at least 17 meters. If smaller than 17 meters, the rover landing speed is above 1.0 m/s and the sky crane altitude is less than 4.5m. Shown below are graphs with the condition of the parachute diameter being less than 17m. The graphs show the differences when the diameter is below 17m and how it does not meet the mission success criteria.