Elizabeth Sharf, Savera Sahai, Michael Hager
MEEN 357-505
Team 14

Rover Project Phase 3

Task 1:

define_rovers			
Field Name	Type	Description	Default/Initial Values
define_rover_1	dict	First rover dictionary test	rover, planet
define_rover_2	dict	Second rover dictionary test	rover, planet
define_rover_3	dict	Third rover dictionary test	rover, planet

Note: The following default/initial values represent numerical values that align with define_rover_1 which may be different for define_rover_2 and define_rover_3. However, the variables stay constant for define_rover_2 and define_rover_3, so we chose to only document this once.

Using Phase 1 Definitions:

rover			
Field Name	Type	Description	Default/Initial Values
wheel_assembly	dict	Wheel assembly dictionary	motor, speed_reducer, wheel
chassis	dict	The chassis dictionary	mass
science_payload	dict	The science payload dictionary	mass
power_subsys	dict	The power subsystem dictionary	mass

wheel_assembly			
Field Name	Type	Description	Default/Initial Values
wheel	dict	The wheel dictionary	mass, radius
speed_reducer	dict	The speed reducer dictionary	type, diam_pinion,

			diam_gear, mass
motor	dict	The motor dictionary	torque_stall, torque_noload, speed_noload, mass, effcy, effcy_tau_

wheel			
Field Name	Type	Description	Default/Initial Values
radius	scalar	Radius of drive wheel [m]	0.3 m
mass	scalar	Mass of one drive wheel [kg]	1 kg

speed_reducer			
Field Name	Type	Description	Default/Initial Values
type	string	String of text defining the type of speed reducer. For Project Phase 1, the only valid entry is “reverted”	“reverted”
diam_pinion	scalar	Diameter of pinion [m]	0.04 m
diam_gear	scalar	Diameter of gear [m]	0.07 m
mass	scalar	Mass of speed reducer assembly [kg]	1.5 kg

motor			
Field Name	Type	Description	Default/Initial Values
torque_stall	scalar	Motor stall torque [N-m]	170 N-m
torque_noload	scalar	Motor no-load torque [N-m]	0 N-m
speed_noload	scalar	Motor no-load speed [rad/s]	3.8 rad/s
mass	scalar	Motor mass [kg]	5 kg

chassis			
Field Name	Type	Description	Default/Initial Values
mass	scalar	Mass of chassis [kg]	659 kg

science_payload			
Field Name	Type	Description	Default/Initial Values
mass	scalar	Mass of science payload [kg]	75 kg

power_subsys			
Field Name	Type	Description	Default/Initial Values
mass	scalar	Mass of power subsystem [kg]	90 kg

planet			
Field Name	Type	Description	Default/Initial Values
g	scalar	Acceleration due to gravity [m/s^2]	3.72 m/s^2

Using Phase 2 Definitions:

experiment			
Field Name	Type	Description	Default/Initial Values
time_range	1D numpy array	Two-element vector containing the start and stop time of the simulation. To be passed to an ODE solver [s]	[0, 2000] s
initial_conditions	1D numpy array	Two-element vector containing initial conditions for your experiment. The elements are defined as follows:	[0.3125, 0]

		initial_conditions[0] : rover velocity [m/s] initial_conditions[1] : rover position [m]	
alpha_dist	1D numpy array	N-element vector of locations corresponding to terrain slope data [m]	[0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000] m
alpha_deg	1D numpy array	N-element vector of terrain slope data for locations given in preceding vector [deg]	[11.509, 2.032, 7.182, 2.478, 5.511, 10.981, 5.601, -0.184, 0.714, 4.151, 4.042] deg
Crr	scalar	Coefficient of rolling resistance. Set this to 0.1 for this phase.	0.1

end_event			
Field Name	Type	Description	Default/Initial Values
max_distance	scalar	Maximum distance to be traverse by the rover [m]	50 m
max_time	scalar	Maximum time for the simulation by the ODE solver [s]	5000 s
min_velocity	scalar	Maximum velocity necessary to halt the solution of the differential equation. This condition is necessary in the case the rover gets 'stuck' [m/s]	0.01 m/s

Phase 3 Definitions:

define_edl_system_1			
Field Name	Type	Description	Default/Initial Values
parachute	dict	The parachute dictionary includes physical definition	deployed, ejected, diameter, Cd, mass

		and state information	
rocket	dict	The rocket dictionary defining a single rocket	on, structure_mass, initial_fuel_mass, fuel_mass, effective_exaust_velocity, max_thrust, min_thrust
speed_control	dict	The speed_control dictionary	on, Kp, Kd, Ki, target_velocity
position_control	dict	The position_control dictionary	on, Kp, Kd, Ki, target_altitude
sky_crane	dict	The sky_crane dictionary lowers the rover onto the surface	on, danger_altitude, danger_speed, mass, area, Cd, mass_cable, velocity
heat_shield	dict	The heat shield dictionary	ejected, mass, diameter, Cd
rover	dict	The redefined rover dictionary	define_rover_4
edl_system	dict	The edl_system dictionary packs everything together and cleans up subdicts	altitude, velocity, num_rockets, volume, parachute, heat_shield, rocket, speed_control, position_control, sky_crane, rover

parachute			
Field Name	Type	Description	Default/Initial Values
deployed	boolean	If true then parachute has been deployed but not ejected	True
ejected	boolean	If true then parachute is no longer attached to system	False
diameter	scalar	The parachute diameter; MSL is about 16 m [m]	16.25 m
Cd	scalar	The nominal constant for subsonic	0.615
mass	scalar	The mass of parachute; represents a wild guess – no data found [kg]	185.0 kg

rocket			
Field Name	Type	Description	Default/Initial Values
on	boolean	If true then the system is running	False
structure_mass	scalar	Represents everything that is not fuel mass [kg]	8 kg
initial_fuel_mass	scalar	The initial fuel mass [kg]	230 kg
fuel_mass	scalar	The current fuel mass <= initial [kg]	230 kg
effective_exhause_velocity	scalar	The effective exhaust velocity [m/s]	4500 m/s
max_thrust	scalar	The maximum thrust [N]	3100 N
min_thrust	scalar	The minimum thrust [N]	40 N

speed_control			
Field Name	Type	Description	Default/Initial Values
on	boolean	If true then the system is running	False
Kp	scalar	Proportional gain term	2000
Kd	scalar	Derivative gain term	20
Ki	scalar	Integral gain term	50
target_velocity	scalar	The desired descent speed [m/s]	-3 m/s

position_control			
Field Name	Type	Description	Default/Initial Values
on	boolean	If true then the system is running	False

Kp	scalar	Proportional gain term	2000
Kd	scalar	Derivative gain term	1000
Ki	scalar	Integral gain term	50
target_altitude	scalar	Needs to reflect the sky crane cable length [m]	7.6 m

sky_crane			
Field Name	Type	Description	Default/Initial Values
on	boolean	If true then means lowered rover mode	False
danger_altitude	scalar	The altitude at which considered too low for safe rover touch down [m]	4.5 m
danger_speed	scalar	The speed at which the rover would impact to hard on surface [m/s]	-1 m
mass	scalar	The sky crane mass [kg]	35 kg
area	scalar	The sky crane area [m^2]	16 m^2
Cd	scalar	The drag coefficient	0.9
max_cable	scalar	The maximum length of cable for lowering rover [m]	7.6 m
velocity	scalar	The speed at which sky crane lowers rover [m/s]	-0.1 m/s

heat_sheild			
Field Name	Type	Description	Default/Initial Values
ejected	boolean	If true then heat shield has been ejected from system	False
mass	scalar	The mass of heat shield [kg]	225 kg
diameter	scalar	The diameter of heat shield [m]	4.5 m
Cd	scalar	The drag coefficient	0.35

rover			
Field Name	Type	Description	Default/Initial Values
define_rover_4	dict	Redefines the rover test dictionary	rover

*Note: the define_rover_4 dictionary is very similar to define_rover_1, define_rover_2, define_rover_3. Therefore, the documentation above for any variables is valid with the alternative initial values.

edl_system			
Field Name	Type	Description	Default/Initial Values
altitude	scalar	System state variable that is updated throughout simulation	np.NaN
velocity	scalar	System state variable that is updated throughout simulation	np.NaN
num_rockets	scalar	System level parameter	8
volume	scalar	System level parameter	150
parachute	dict	The parachute dictionary	parachute (see earlier dict)
heat_sheild	dict	The heat_sheild dictionary	heat_sheild (see earlier dict)
rocket	dict	The rocket dictionary	rocket (see earlier dict)
speed_control	dict	The speed_control dictionary	speed_control (see earlier dict)
position_control	dict	The position control dictionary	position_control (see earlier dict)
sky_crane	dict	The sky crane dictionary	sky_crane (see earlier dict)
rover	dict	Updated rover dictionary	rover (see earlier dict)

mission_events			
Field Name	Type	Description	Default/Initial Values
alt_heatshield_eject	scalar	The alternate head shield ejection value	8000
alt_parachute_eject	scalar	The alternate parachute ejection value	900
alt_rockets_on	scalar	The alternate rocket on value	1800
alt_skycrane_on	scalar	The alternate skycrane on value	7.6

define_planet			
Field Name	Type	Description	Default/Initial Values
high_altitude	dict	The planet information at high altitudes	temperature, pressure
low_altitude	dict	The planet information at low altitudes	temperature, pressure
density	scalar	The density value for the planet [kg/m^3]	lambda temperature, pressure: pressure/(0.1921*(temperature+273.15))
mars	dict	Information on planet mars	g, altitude_threshold, high_altitude, low_altitude, density

high_altitude			
Field Name	Type	Description	Default/Initial Values
temperature	scalar	High altitude temperature [C]	lambda altitude: -23.4 - 0.00222*altitude
pressure	scalar	High altitude pressure [kPa]	lambda altitude: 0.699*np.exp(-0.00009*altitude)

low_altitude			
Field Name	Type	Description	Default/Initial Values
temperature	scalar	Low altitude temperature [C]	lambda altitude: -31 - 0.000998*altitude
pressure	scalar	Low altitude pressure [kPa]	lambda altitude: 0.699*np.exp(-0.00009*altitude)

mars			
Field Name	Type	Description	Default/Initial Values
g	scalar	The mars gravity [m/s^2]	-3.72 m/s^2
altitude_threshold	scalar	The base altitude [m]	7000 m
high_altitude	dict	The updated planet information at high altitudes	low_altitude (see earlier dict)
low_altitude	dict	The updated planet information at low altitudes	low_altitude (see earlier dict)
density	scalar	The updated density [kg/m^3]	density (see earlier dict)

Task 2:

get_mass_rover		
General description		
This function computes the mass of the defined rover in kilograms.		
Calling Syntax		
m = get_mass_rover(edl_system)		
Input Arguments		
edl_system	dict	Data structure specifying the EDL system parameters

Output Arguments		
m	scalar	Rover mass [kg]

get_mass_rockets		
General description		
This function outputs the total mass of the rockets on the EDL system.		
Calling Syntax		
m = get_mass_rockets(edl_system)		
Input Arguments		
edl_system	dict	Data structure specifying the EDL system parameters
Output Arguments		
m	scalar	Total rocket mass [kg]

get_mass_edl		
General description		
This function outputs the total mass of the EDL system.		
Calling Syntax		
m = get_mass_edl(edl_system)		
Input Arguments		
edl_system	dict	Data structure specifying the EDL system parameters

Output Arguments		
m	scalar	Total EDL system mass [kg]

get_local_atm_properties		
General description		
This function outputs the local atmospheric properties at a selected altitude.		
Calling Syntax		
density, temperature, pressure = get_local_atm_properties(planet, altitude)		
Input Arguments		
planet	dict	Data structure that contains the planet’s parameters
altitude	scalar	Altitude [m]
Output Arguments		
density	scalar	Density of the Atmosphere [kg/m^3]
temperature	scalar	Temperature of the Atmosphere [°C]
pressure	scalar	Pressure of the Atmosphere [kPa]

F_buoyancy_descent
General description
This function computes the net buoyancy force for the given EDL system, planet, and altitude.

Calling Syntax		
F = F_buoyancy_descent(edl_system, planet, altitude)		
Input Arguments		
edl_system	dict	Data structure specifying the EDL system parameters
planet	dict	Data structure that contains the planet’s parameters
altitude	scalar	Altitude [m]
Output Arguments		
F	scalar	Net buoyancy force [N]

F_drag_descent		
General description		
This function computes the net drag force of the descent for a given EDL system, planet, altitude, and velocity.		
Calling Syntax		
F = F_drag_descent(edl_sysem, planet, altitude, velocity		
Input Arguments		
edl_system	dict	Data structure specifying the EDL system parameters
planet	dict	Data structure that contains the planet’s parameters
altitude	scalar	Altitude [m]
velocity	scalar	Velocity of the rover [m/s]

Output Arguments		
F	scalar	Net drag force [N]

F_gravity_descent		
General description		
This function computes the gravitational force acting on a given EDL system for a given planet.		
Calling Syntax		
F = F_gravity_descent(edl_system, planet)		
Input Arguments		
edl_system	dict	Data structure specifying the EDL system parameters
planet	dict	Data structure that contains the planet's parameters
Output Arguments		
F	scalar	Gravitational force [N]

v2M_Mars
General description
This function converts a given descent speed, v, to a Mach number as a function of altitude, a, for Mars.
Calling Syntax
M = v2M_Mars(v, a)

Input Arguments		
v	scalar	Descent speed [m/s]
a	scalar	Altitude [m]
Output Arguments		
M	scalar	Absolute value Mach number

thrust_controller		
General description		
This function uses the edl_system and planet structures to create a modified edl_system structure.		
Calling Syntax		
edl_system = thrust_controller(edl_system, planet)		
Input Arguments		
edl_system	dict	Data structure specifying the EDL system parameters
planet	dict	Data structure that contains the planet's parameters
Output Arguments		
edl_system	dict	Updated data structure of EDL system parameters

edl_events
General description

This function defines the events that happen in the EDL System simulation.
Calling Syntax
events = edl_events(edl_system, mission_events)
Input Arguments

edl_system	dict	Data structure specifying the EDL system parameters
mission_events	dict	Data structure with the parameters of the mission event
Output Arguments		
events	dict	Data structure containing the events that occur in the simulation

edl_dynamics
General description
This function finds the dynamics of the EDL system as the rover descends through the atmosphere towards the surface.
Calling Syntax
dy/dt = edl_dynamics(t, y, edl_system, planet)
Input Arguments

t	scalar	The time at which the dynamics are being calculated [s]
y	numpy array	A 7 element array of the dependent variables for the dynamics 1. EDL System velocity [m/s] 2. EDL System altitude [m]

		3. EDL System fuel mass [kg] 4. Velocity error integral [m/s] 5. Altitude/position error integral [m] 6. Rover velocity relative to sky crane [m/s] 7. Rover position relative to sky crane [m]
edl_system	dict	Data structure specifying the EDL system parameters
planet	dict	Data structure that contains the planet's parameters
Output Arguments		
dy/dt	1D numpy array	A 7 element array of the time derivatives of the inpu array 'y' 1. EDL System acceleration [m/s^2] 2. EDL System velocity [m/s] 3. Effective exhaust velocity of propellant being expelled [m/s] 4. Velocity error [m/s] 5. Altitude/position error [m] 6. Rover acceleration relative to sky crane [m/s^2] 7. Rover velocity relative to sky crane [m/s]

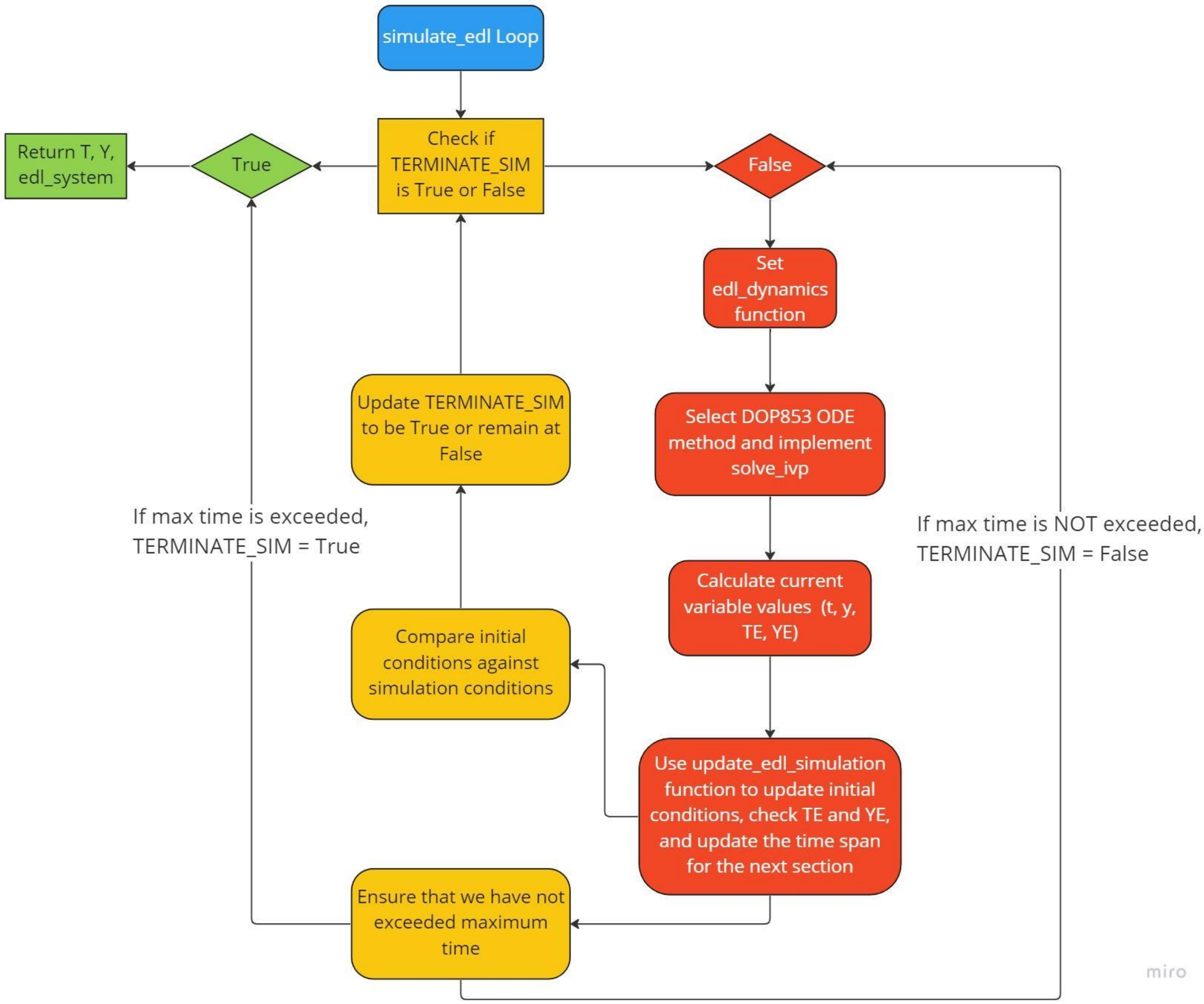
update_edl_state		
General description		
This function updates the EDL System status based on the events of the simulation.		
Calling Syntax		
edl_system, y0, TERMINATE_SIM = update_edl_state(edl_system, TE, YE, Y, ITER_INFO)		
Input Arguments		
edl_system	dict	Data structure specifying the EDL system parameters
TE	vector	The times at which the simulation events occur

YE	vector	The simulation history of the vectors detailed in Y
Y	numpy array	A 7 row N-column numpy array of the state vector history from the simulation 1. EDL System velocity [m/s] 2. EDL System altitude [m] 3. EDL System fuel mass [kg] 4. Velocity error integral [m/s] 5. Altitude/position error integral [m] 6. Rover velocity relative to sky crane [m/s] 7. Rover position relative to sky crane [m]
ITER_INFO	boolean	Optional flag to display detailed iteration information
Output Arguments		
edl_system	dict	Updated data structure of EDL system parameters
y0	vector	New default initial conditions from previous intervals
TERMINATE_SIM	boolean	Returns if the EDL System will continue running or terminate

simulate_edl		
General description		
This function simulates the EDL System.		
Calling Syntax		
T, Y, edl_system = simulate_edl(edl_system, planet, mission_events, tmax, ITER_INFO)		
Input Arguments		
edl_system	dict	Data structure specifying the EDL system

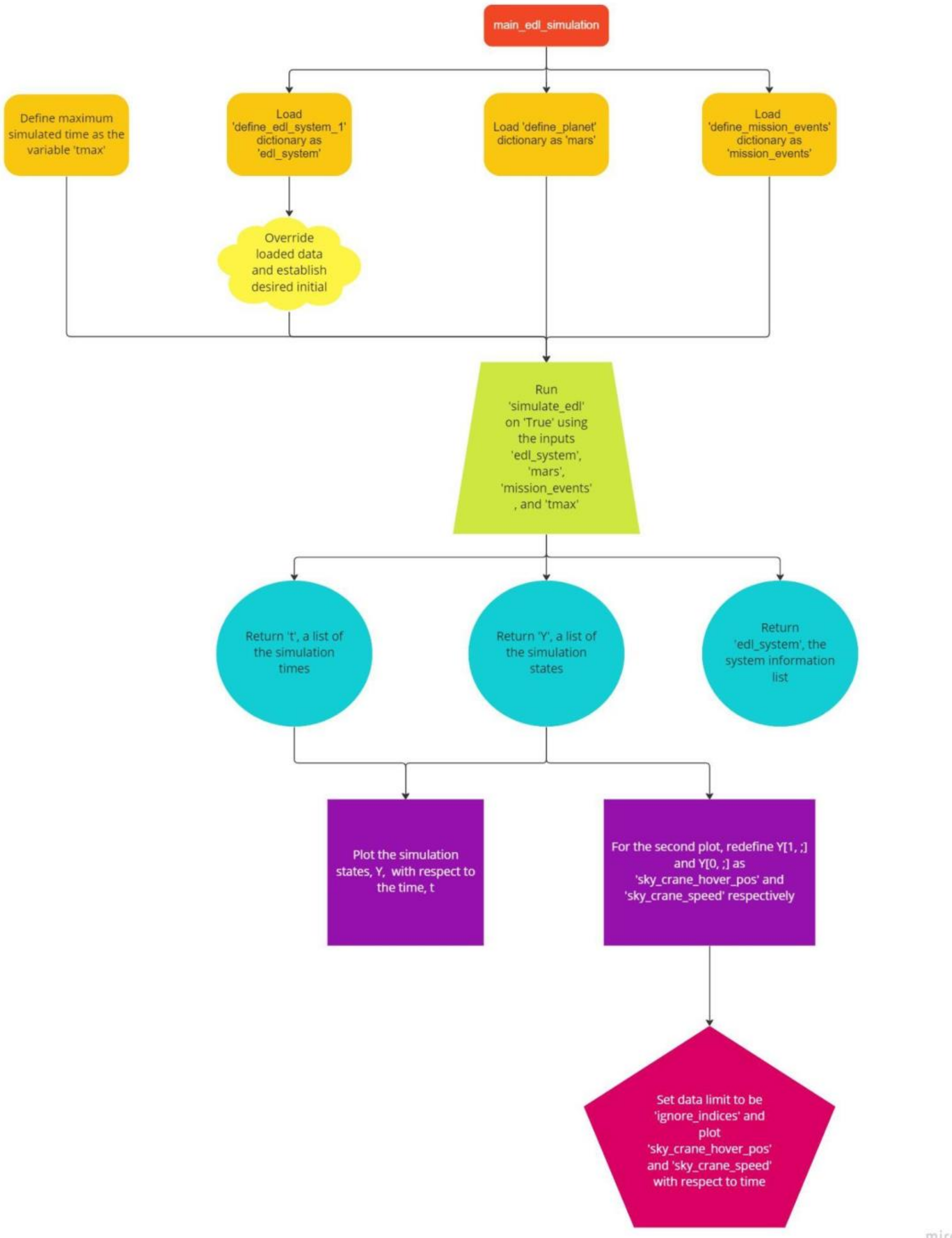
		parameters
planet	dict	Data structure that contains the planet's parameters
mission_events	dict	Data structure that contains the mission events parameters
tmax	scalar	Maximum simulation time [s]
ITER_INFO	boolean	Optional flag to display detailed iteration information
Output Arguments		
T	1D numpy array	Time history in an N-element array [s]
Y	numpy array	A 7 row N-column numpy array containing the simulation state vector history 1. EDL System velocity [m/s] 2. EDL System altitude [m] 3. EDL System fuel mass [kg] 4. Velocity error integral [m/s] 5. Altitude/position error integral [m] 6. Rover velocity relative to sky crane [m/s] 7. Rover position relative to sky crane [m]
edl_system	dict	Updated data structure of EDL system parameters

Task 3:



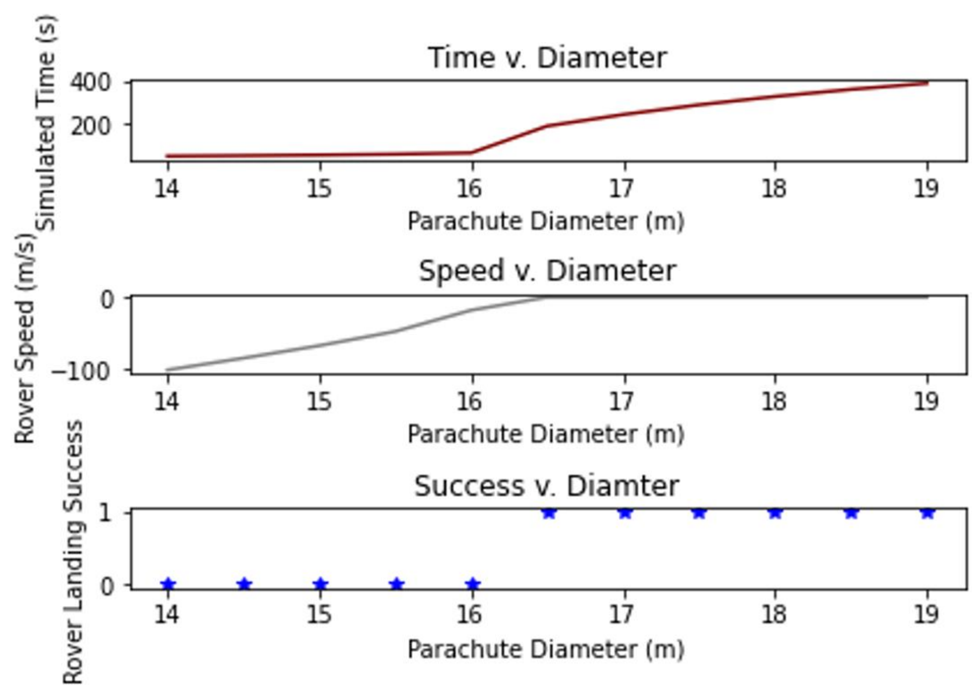
In order to continually check the state of the rover while it lands, the implementation of the while loop within the simulate_edl function is imperative. The loop begins after setting the initial time to zero and checking to see if TERMINATE_SIM is False. If false, then the edl_dynamics and DOP853 method of IVP solver are implemented in order to calculate the current variables. These are t, y, TE, and YE. t_part and Y_part hold solutions for t and y, respectively, in addition to TE and YE each locating a solution within a list of arrays at which an event of t or y are detected. Then, utilizing the update_edl_simulation, we are able to update the initial condition and the time span of the loop. If maximum time is not exceeded thereforth, the loop will return to False and the process is repeated. The edl_system and initial conditions of each event are updated after every time the loop is run. However, if the maximum time is reached or exceeded (and of course, the rover has not crashed), then the TERMINATE_SIM will return as True. If it reads True, then the while loop terminates, the function will exit, and t, y, and edl_system will be returned.

Task 4:



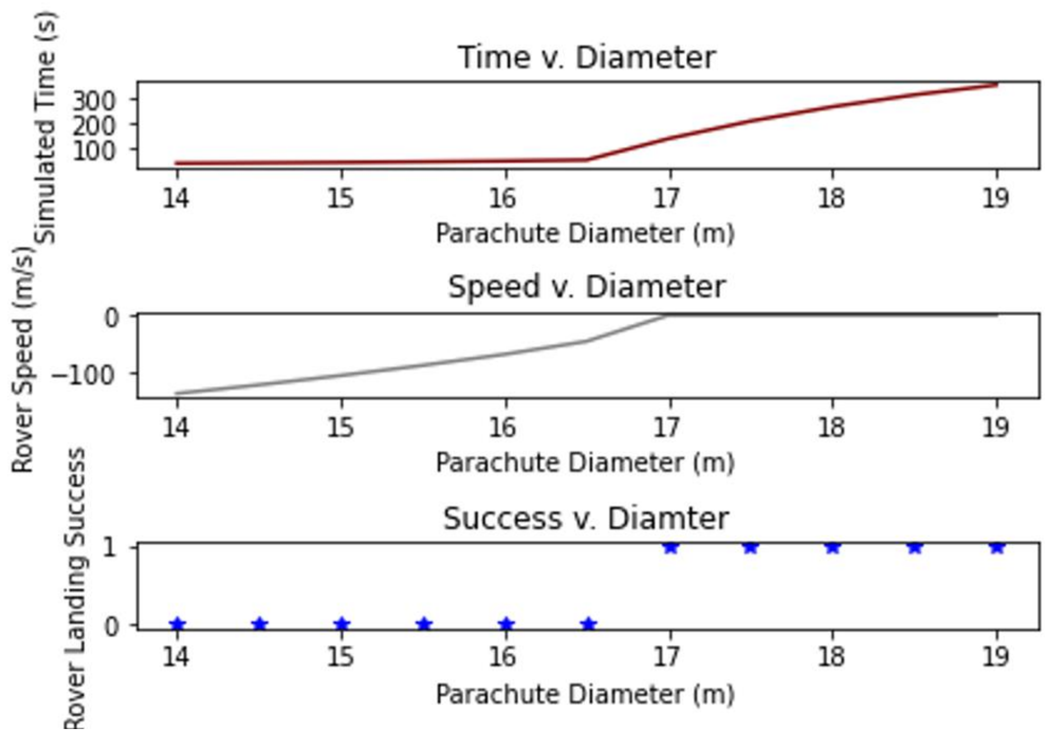
The main_edl_simulation script launches a simulation of the EDL System. The script initially loads the required dictionaries ‘define_edl_system_1’, ‘define_planet’, and ‘define_mission_events’. The dictionary ‘define_edl_system_1’ is defined as ‘edl_system’, ‘define_planet’ is defined as ‘mars’ and ‘define_mission_events’ is defined as ‘mission_events’. Then, the desired initial conditions for altitude, velocity, parachute deployed, parachute ejected, parachute diameter, and rover on ground are defined in 'edl_system' to override values that might be in the loaded data. The maximum simulated time is then defined as 'tmax'. These values are plugged into the functions defined in 'simulate_edl' to return lists 't' and 'Y', as well as the updated 'edl_system'. The list 't' contains the simulation times and the list 'Y' contains the simulation values. Each scenario is then plotted for the values, Y, with respect to time, t. For the next plot, 'sky_crane_hover_pos' is defined to be Y[1, :] and 'sky_crane_speed' is defined to be Y[0, :]. The limiter 'ignore_indices' is then applied to 'sky_crane_hover_pos' and 'sky_crane_speed' and they are iterated by applying np.Nan and the data sets are plotted with respect to time.

Task 5:



After observing the graphs above for task 5, we can conclude that a diameter of 16 meters or less will result in the rover having an unsuccessful landing on planet Mars. On the contrary, a diameter of 16.5 meters and above will result in a successful landing on the planet mars. Our group's recommendation is that we go with a parachute of 17 to 17.5 meters. Even though 16.5 meters is the most effective, having success as well as maximum speed and minimum time of opening with success, there are small chances that our calculations can be slightly off, so going with 17 to 17.5 meters is the next best option. This range of diameters guarantees a safe landing, while also having maximum speed. Any diameter between 16.5 and 19 meters is considered a success, but we are choosing to stay as low as possible in diameter to reduce the amount of time it takes for the parachute to open.

Task 6:



For this task, we recreated the graphs from task 5 by running the same code but changing the "F_drag_descent" function. We created the continuous model for the MEF data by using the "v2M_Mars" function to convert the velocity and altitude into Mach numbers that represent the speed and location of the rover when it's landing. We used the two lists given in the "Phase 3 Project Overview" to be our Mach and MEF values. From there, we set the C_d (coefficient of drag) value to be 0.615, and used the "pchip" function which was imported to create an interpolation function that takes in the values output from the "v2M_Mars" function. After that, we redefined MEF using our interpolation function and then calculated C_{d_mod} by multiplying MEF and C_d. From here, we obtained the density from calling the "get_local_atm_properties" function and used that to calculate for "rhov2" using the equation 0.5 * density * velocity^2. We then do calculations for "ACd_body" and "ACd_parachute" based on whether the parachute is deployed or the heat child is ejected. Lastly, we calculate the ultimate drag force by using the equation rhov2 * (ACd_body + ACd_parachute). Before making these changes, the drag coefficient was set as an independent variable which led to inefficient results, compared to the realistic drag coefficient which is dependent on the altitude and velocity.

After running the graphing and calculation functions using the updated function for drag, the graphs changed slightly. The graphs shifted to the right by 0.5 meters, which indicates that the coefficient of drag being a dependent variable does affect the overall flight of the rover compared to treating the coefficient as an independent variable. In the last task, we stated that the recommended parachute diameter was 17 to 17.5 meters. After our calculations for task 6, we would recommend the parachute diameter to be 17.5 to 18 meters to optimize safety and time of the parachute opening.