Phase 3 Report

Jacob Hargreaves
David Guess
Ian Wilhite

Task 1:

define_mission_events.py

mission_events			
Field Name	***Value***	***Units***	***Description***
alt_heatshield_eject	8000	Meters	The altitude where the heat shield ejects
alt_parachute_eject	900	Meters	The altitude where the parachute ejects
alt_rockets_on	1800	Meters	The altitude where the rockets turn on
alt_skycrane_on	7.6	Meters	The altitude where the sky crane turns on

define_planet.py

high_altitude			
Field Name	***Value***	***Units***	***Description***
temperature	-38.94 @ 7000 m	Celsius	The temperature based on altitude above 7000 meters
pressure	0.000646 @ 7000 m	KPa	The temperature based on altitude above 7000 meters

low_altitude			
Field Name	***Value***	***Units***	***Description***
temperature	-31 @ 0 m	Celsius	The temperature based on altitude below 7000 meters

pressure	0.669 @ 0 m	KPa	The temperature based on altitude below 7000 meters

mars			
Field Name	***Value***	***Units***	***Description***
g	-3.72	m/s^2	The value of gravity on Mars
altitude_threshold	7000	KPa	The temperature based on altitude below 7000 meters
low_altitude	-	-	The dictionary low_altitude
high_altitude	-	-	The dictionary high_altitude
density	-	-	The dictionary density

define_rovers.py

wheel, rover 1			
Field Name	***Default Value***	***Units***	***Description***
radius	0.30	Meters	Radius of the wheel
mass	1	kilograms	Mass of wheel

wheel, rover 2			
Field Name	***Default Value***	***Units***	***Description***
radius	0.30	Meters	Radius of the wheel
mass	2	kilograms	Mass of wheel

wheel, rover 3			
Field Name	***Default Value***	***Units***	***Description***
radius	0.30	Meters	Radius of the wheel
mass	2	kilograms	Mass of wheel

wheel, rover 4			
Field Name	***Default Value***	***Units***	***Description***
radius	0.20	Meters	Radius of the wheel
mass	2	kilograms	Mass of wheel

speed_reducer, rover 1			
Field Name	***Default Value***	***Units***	***Description***
type	reverted	-	Radius of the wheel
diam_pinion	.04	Meters	Diameter of the pinion
diam_gear	.07	Meters	Diameter of the gear
mass	1.5	kilograms	Mass of speed reducer

speed_reducer, rover 2, 3, & 4			
Field Name	***Default Value***	***Units***	***Description***
type	reverted	-	Radius of the wheel
diam_pinion	.04	Meters	Diameter of the pinion
diam_gear	.06	Meters	Diameter of the gear
mass	1.5	kilograms	Mass of speed reducer

motor, rover 1			
Field Name	***Default Value***	***Units***	***Description***
torque_stall	170	-	Torque of the motor while stalling
torque_noload	0	N*m	Torque of the motor with no load
speed_noload	3.80	Rad/s	Angular speed of the motor with no load
mass	5.0	kilograms	Mass of the motor

motor, rover 2 & 3			
Field Name	***Default Value***	***Units***	***Description***
torque_stall	180	-	Torque of the motor while stalling
torque_noload	0	N*m	Torque of the motor with no load

speed_noload	3.70	Rad/s	Angular speed of the motor with no load
mass	5.0	kilograms	Mass of the motor

motor, rover 4			
Field Name	***Default Value***	***Units***	***Description***
torque_stall	165	-	Torque of the motor while stalling
torque_noload	0	N*m	Torque of the motor with no load
speed_noload	3.85	Rad/s	Angular speed of the motor with no load
mass	5.0	kilograms	Mass of the motor

chassis, rover 1, 2, & 3			
Field Name	***Default Value***	***Units***	***Description***
mass	659	kilograms	Mass of chassis

chassis, rover 4			
Field Name	***Default Value***	***Units***	***Description***
mass	674	kilograms	Mass of chassis

science_payload, rover 1, 2, & 3			
Field Name	***Default Value***	***Units***	***Description***
mass	75	kilograms	Mass of science payload

science_payload, rover 4			
Field Name	***Default Value***	***Units***	***Description***
mass	80	kilograms	Mass of science payload

power_subsys, rover 1, 2, & 3			
Field Name	***Default Value***	***Units***	***Description***
mass	90	kilograms	Mass of power subsystem

power_subsys, rover 4			
Field Name	***Default Value***	***Units***	***Description***
mass	100	kilograms	Mass of power subsystem

rover			
Field Name	***Default Value***	***Units***	***Description***
wheel	wheel	-	Dictionary of the wheel
speed_reducer	speed_reducer	-	Dictionary of the speed_reducer
motor	motor	-	Dictionary of the motor

planet			
Field Name	***Default Value***	***Units***	***Description***
g	3.72	m/s^2	Gravity value on the planet (Mars)

define_edl_system.py

parachute			
Field Name	***Value***	***Units***	***Description***
deployed	True	-	True means it has been deployed but not ejected
ejected	False	-	True means the parachute is no longer attached to the system

diameter	16.25	m	Diameter of the parachute
Cd	0.615	-	Coefficient of drag on the parachute
mass	185.0	kg	Mass of the parachute

rocket			
Field Name	***Value***	***Units***	***Description***
on	False	-	Indicates if the rocket is currently on.
structure_mass	8.0	kg	Mass of the rocket structure excluding fuel.
initial_fuel_mass	230.0	kg	Initial mass of fuel.
fuel_mass	230.0	kg	Current fuel mass (less than or equal to initial_fuel_mass).
effective_exhaust_velocity	4500.0	m/s	Effective exhaust velocity of the rocket.
max_thrust	3100.0	N	Maximum thrust generated by the rocket.
min_thrust	40.0	N	Minimum thrust generated by the rocket.

speed_control			
Field Name	***Value***	***Units***	***Description***
on	False	-	Indicates if the speed control mode is activated.
Kp	2000	-	Proportional gain term for speed control.
Kd	20	-	Derivative gain term for speed control.
Ki	50	-	Integral gain term for speed control.
target_velocity	-3.0	m/s	Desired descent speed.

position_control			
Field Name	***Value***	***Units***	***Description***
on	False	-	Indicates if the position control mode is activated.
Kp	2000	-	Proportional gain term for position control.
Kd	1000	-	Derivative gain term for position control.
Ki	50	-	Integral gain term for position control.
target_altitude	7.6	m	Target altitude, reflecting the sky crane cable length.

sky_crane			
Field Name	***Value***	***Units***	***Description***
on	False	-	Indicates if the sky crane is lowering the rover.
danger_altitude	4.5	m	Altitude considered too low for safe rover touchdown.
danger_speed	-1.0	m/s	Speed below which the rover would impact too hard on the surface.
mass	35.0	kg	Mass of the sky crane.
area	16.0	m^2	Frontal area used for drag calculations.
Cd	0.9	-	Coefficient of drag for the sky crane.
max_cable	7.6	m	Maximum length of cable for lowering the rover.
velocity	-0.1	m/s	Speed at which the sky crane lowers the rover.

heat_shield			
Field Name	***Value***	***Units***	***Description***
ejected	False	-	True if the heat shield has been ejected from the system.
mass	225.0	kg	Mass of the heat shield.

diameter	4.5	m	Diameter of the heat shield.
Cd	0.35	-	Drag coefficient of the heat shield.

edl_system			
Field Name	***Value***	***Units***	***Description***
altitude	NaN	m	Current altitude of the system, updated throughout the simulation.
velocity	NaN	m/s	Current velocity of the system, updated throughout the simulation.
num_rockets	8	-	Total number of rockets in the system.
volume	150	m^3	Volume of the system.
parachute	parachute	-	Parachute dictionary with deployment and drag properties.
heat_shield	heat_shield	-	Heat shield dictionary with mass and drag properties.
rocket	rocket	-	Rocket dictionary with thrust and fuel properties.
speed_control	speed_control	-	Speed control dictionary for descent speed management.
position_control	position_control	-	Position control dictionary for altitude management.

sky_crane	sky_crane	-	Sky crane dictionary for rover lowering.
rover	rover	-	Rover dictionary (defined in another file).

Task 2:

get_mass_rover			
Calling Syntax	***Description***	***Input Arguments***	***Output Arguments***
get_mass_rover(edl_system)	Computes the total mass of the rover based on the specifications in the edl_system dictionary, as per Phase 1 requirements.	edl_system: dictionary containing rover components and their respective masses	m: total mass of the rover (float)

get_mass_rockets			
Calling Syntax	***Description***	***Input Arguments***	***Output Arguments***
get_mass_rockets(edl_system)	Returns the current total mass of all rockets on the EDL system.	edl_system: dictionary containing the number of rockets and the masses of the rocket components	m: total mass of all rockets (float)

get_mass_edl			
Calling Syntax	***Description***	***Input Arguments***	***Output Arguments***

get_mass_edl(edl_system)	Returns the total current mass of the entire EDL system, accounting for any ejected components.	edl_system: dictionary containing the masses of the EDL system's components, including parachute, heat shield, rockets, sky crane, and rover	m: total current mass of the EDL system (float)

get_local_atm_properties			
Calling Syntax	***Description***	***Input Arguments***	***Output Arguments***
get_local_atm_properties(planet, altitude)	Returns local atmospheric properties—density, temperature, and pressure—at a given altitude.	planet: dictionary with atmospheric properties and calculation functions for density, temperature, and pressure based on altitude; altitude: altitude above the planet’s surface (meters)	density: atmospheric density (kg/m³); temperature: local temperature (°C); pressure: local pressure (kPa)

F_buoyancy_descent			
Calling Syntax	***Description***	***Input Arguments***	***Output Arguments***
F_buoyancy_descent(edl_system, planet, altitude)	Computes the net buoyancy force acting on the EDL system during descent.	edl_system: dictionary containing volume of the EDL system; planet: dictionary with gravity and atmospheric functions; altitude: altitude above the planet’s surface (meters)	F: net buoyancy force (float)

F_drag_descent			
Calling Syntax	***Description***	***Input Arguments***	***Output Arguments***

F_drag_descent(edl_system, planet, altitude, velocity)	Computes the net drag force acting on the EDL system during descent.	edl_system: dictionary containing the components of the EDL system; planet: dictionary with atmospheric properties; altitude: altitude above the planet's surface (meters); velocity: current descent velocity (m/s)	F: net drag force (float)

F_gravity_descent			
Calling Syntax	***Description***	***Input Arguments***	***Output Arguments***
F_gravity_descent(edl_system, planet)	Computes the gravitational force acting on the EDL system.	edl_system: dictionary with the mass of the EDL system; planet: dictionary with gravity parameter	F: gravitational force (float)

v2M_Mars			
Calling Syntax	***Description***	***Input Arguments***	***Output Arguments***
v2M_Mars(v, a)	Converts descent speed to Mach number on Mars as a function of altitude.	v: descent speed (m/s); a: altitude (m)	M: Mach number (float)

thrust_controller			
Calling Syntax	***Description***	***Input Arguments***	***Output Arguments***
thrust_controller(edl_system, planet)	Implements a PID Controller for the EDL system to adjust thrust based on errors in velocity control.	edl_system: dictionary containing the control parameters, telemetry, and rocket specifications; planet: dictionary with gravity information	edl_system: modified dictionary with updated thrust and telemetry data

edl_events			
Calling Syntax	***Description***	***Input Arguments***	***Output Arguments***
edl_events(edl_system, mission_events)	Defines events occurring in EDL System simulation	edl_system: EDL system state, mission_events: mission-specific event altitude/speed values	events: List of event functions for EDL conditions

edl_dynamics			
Calling Syntax	***Description***	***Input Arguments***	***Output Arguments***
edl_dynamics(t, y, edl_system, planet)	Calculates EDL dynamics as it descends to the Mars surface	t: Time, y: State vector, edl_system: EDL system details, planet: Planet details affecting EDL	dydt: Array of rates of change in state vector variables

update_edl_state			
Calling Syntax	***Description***	***Input Arguments***	***Output Arguments***
update_edl_state(edl_system, TE, YE, Y, ITER_INFO)	Updates the status of the EDL system based on simulation events, like ejection, activation, and landing conditions.	edl_system (dict): EDL system state, TE (list): Event times, YE (list): State at event times, Y (array): States, ITER_INFO (bool): Logging flag	edl_system (dict): Updated system, y0 (array): New initial conditions, TERMINATE_SIM (bool): Simulation termination flag

simulate_edl			
Calling Syntax	***Description***	***Input Arguments***	***Output Arguments***

simulate_edl(edl_system, planet, mission_events, tmax, ITER_INFO)	Runs the simulation of the EDL system through iterative time steps until termination based on events or time limit	edl_system (dict): System parameters, planet (dict): Planetary constants, mission_events (dict): Event conditions, tmax (float): Max time, ITER_INFO (bool): Logging flag	T (array): Simulation time steps, Y (array): State vectors, edl_system (dict): Final system state

Task 3:

IVP Solver
The while loop in simulate_edl uses solve_ivp with the DOP853 method, an 8th-order Runge-Kutta method that provides higher accuracy compared to other methods like RK45. This higher precision is essential due to the specific event criteria we are monitoring, requiring close alignment with actual values to correctly detect each event.

solve_ivp is called with the following parameters:

- fun: A lambda function representing the system's dynamics, edl_dynamics(t, y, edl_system, planet).
- tspan: A tuple defining the simulation time span, (0, tmax).
- y0: An array of initial state variables, including initial velocity, altitude, fuel mass, and other state indicators.
- method: Set to "DOP853" to ensure high accuracy for Martian descent simulation.
- events: A set of conditions to trigger specific events, as defined by edl_events(edl_system, mission_events).
- max_step: The maximum step size for the solver, set to 0.1 for finer granularity.

Each iteration of solve_ivp simulates until one of the defined events occurs. When an event triggers, the solver captures t (time) and y (state variables like altitude, fuel, etc.), which are then updated within the update_edl_state function.

Updating the EDL System
The update_edl_state function is crucial for this loop, as it processes the event results from solve_ivp and updates edl_system accordingly. This function also sets up the initial conditions y0 for the next loop iteration and checks if the simulation should terminate by updating the TERMINATE_SIM flag.

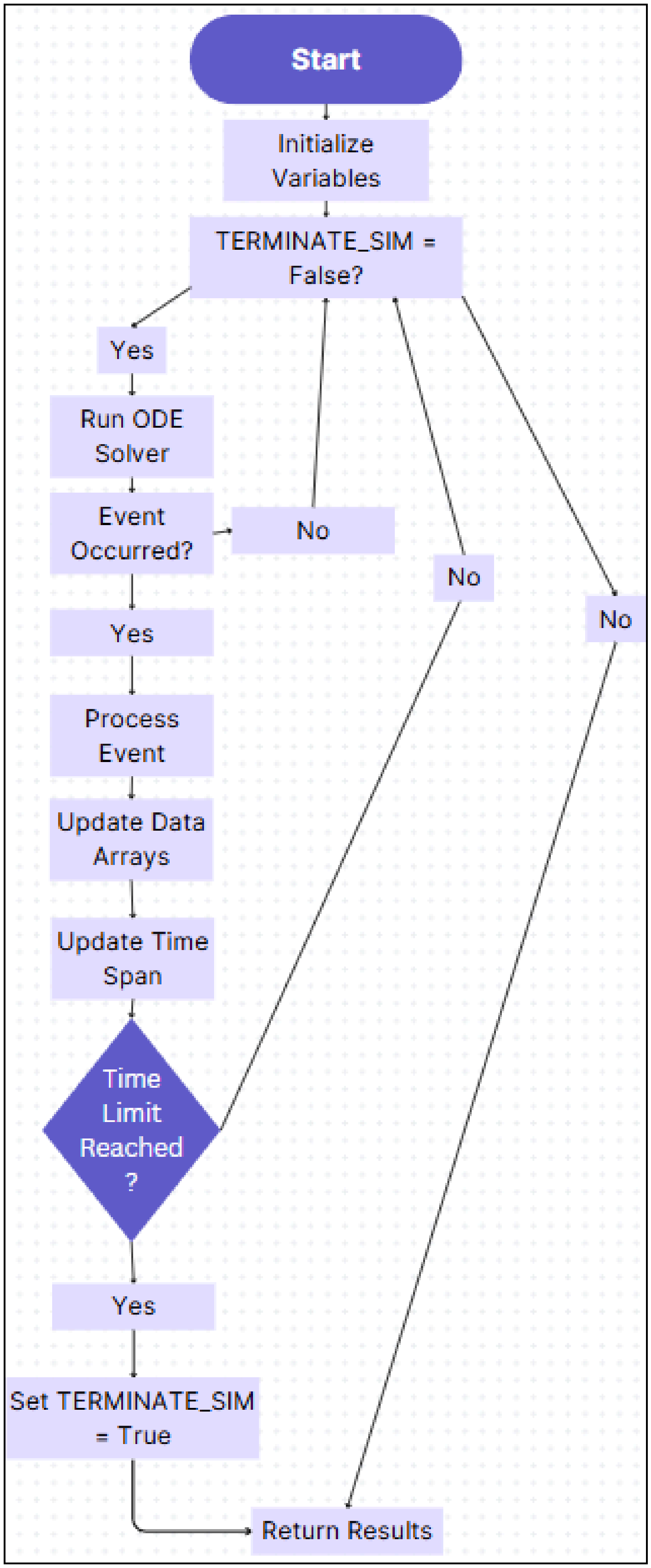
Key events handled by update_edl_state include:
1. Heat Shield Ejection: Heat shield is detached when a certain altitude is reached.
2. Parachute Ejection: Parachute is released based on altitude criteria.
3. Rockets Activation: Rockets are ignited for descent control at specific altitude.
4. Sky Crane Activation: Sky crane system is engaged to lower the rover.
5. Out of Rocket Fuel: Fuel depletion triggers a termination condition if descent is not complete.
6. Sky Crane Safety Failure: The rover crashes if descent control or sky crane fails.
7. Speed Controller Activation: Engages to maintain a target descent speed.
8. Altitude Controller Activation: Altitude controller engages as the sky crane takes over, disabling speed control.
9. Rover Grounded: Reaches the surface (altitude of zero), ending the simulation.

Explanation of the Loop

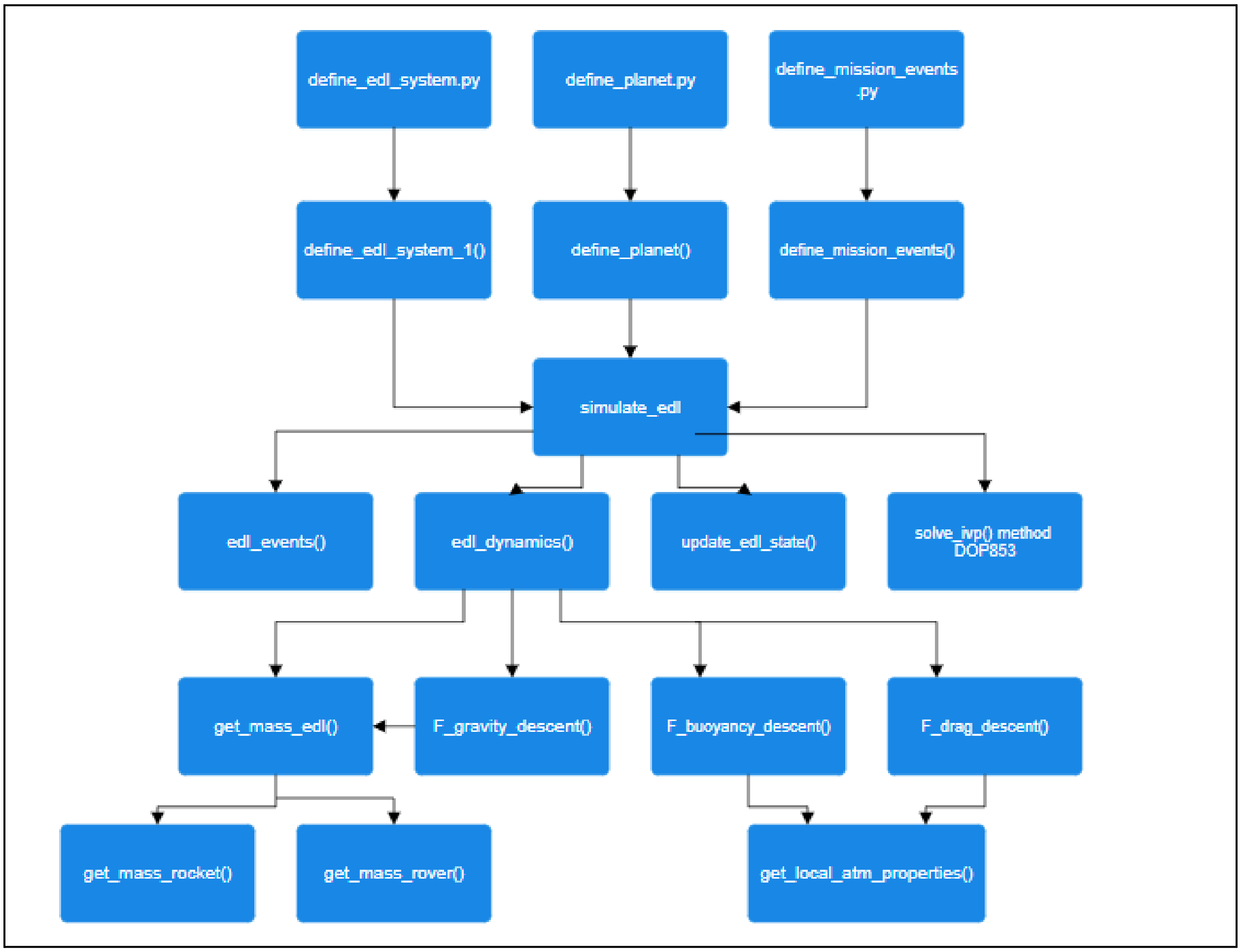
The while loop in simulate_edl repeats until TERMINATE_SIM is set to True. Termination occurs when time runs out, the rover safely lands, fuel depletes, or a crash is detected.

The process within each loop iteration is as follows:
1. A lambda function for the system dynamics (fun) is passed to solve_ivp.
2. solve_ivp runs the simulation until one of the specified events occurs, producing arrays of t (time) and y (state variables) for the event.
3. update_edl_state then adjusts edl_system, sets y0 for the next pass, and updates TERMINATE_SIM if an exit condition is reached.
4. The loop appends time (T) and state variables (Y) for further analysis, updating tspan with new bounds for the next simulation pass.

This iterative simulation loop accurately models different operational phases of the EDL sequence: parachute deployment, rocket firing, sky crane descent, and finally, rover touchdown.

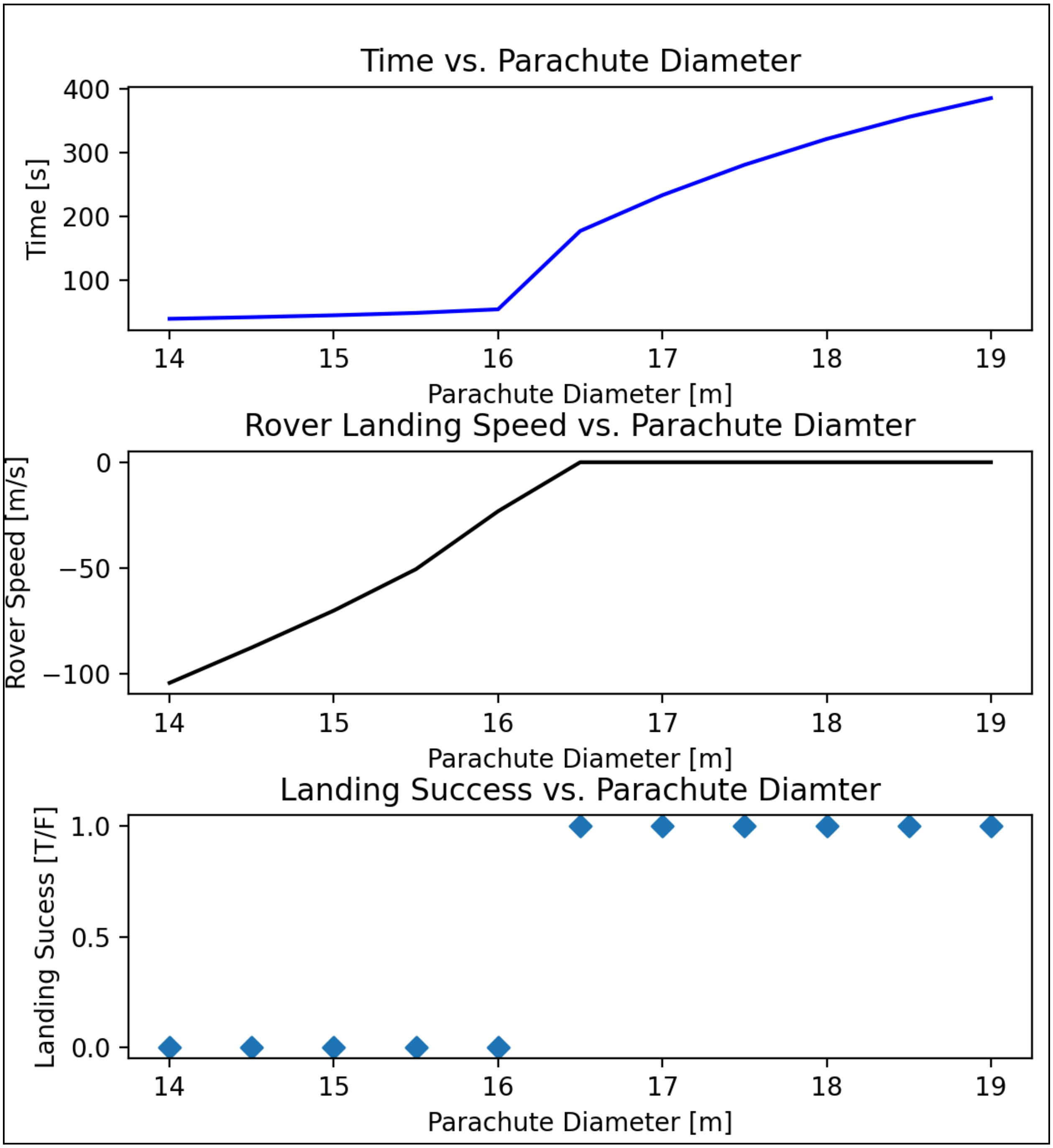

Start
Initialize Variables
TERMINATE_SIM = False?
Yes
Run ODE Solver
Event Occurred?
No
No
No
Yes
Process Event
Update Data Arrays
Update Time Span
Time Limit Reached ?
Yes
Set TERMINATE_SIM = True
Return Results

Task 4:



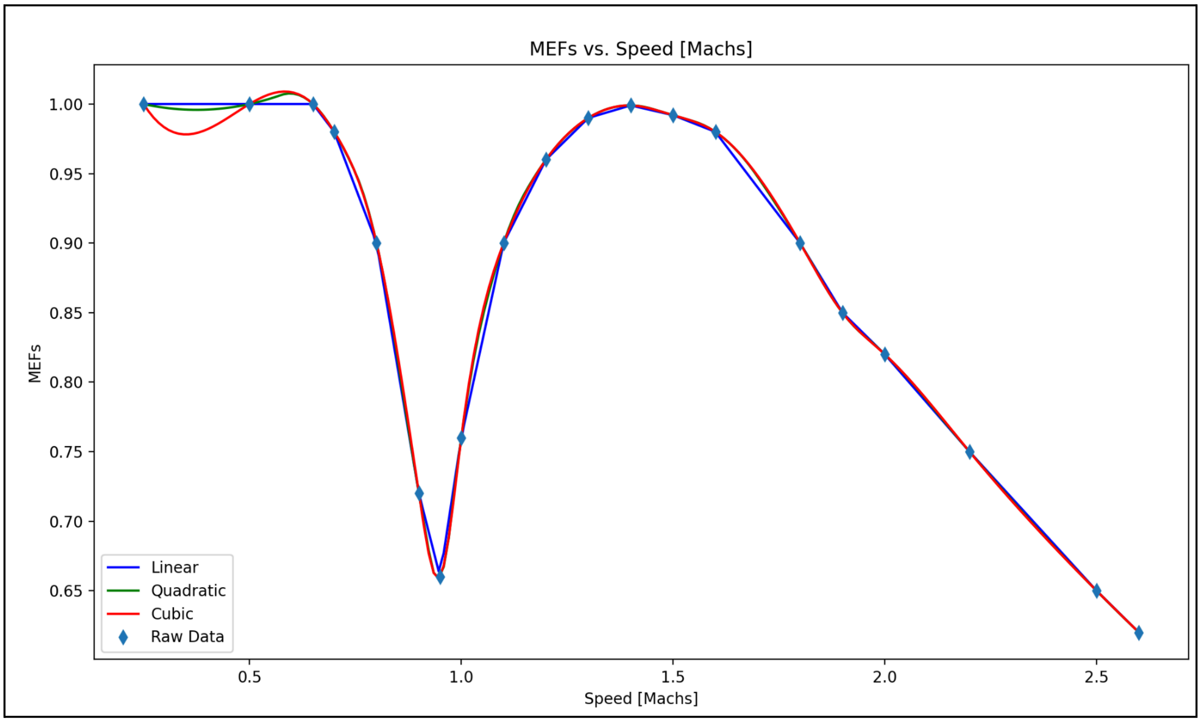
This flowchart begins with the imported files, followed by the loading of key dictionaries. These dictionaries help establish variables that are then passed into simulate_edl. Inside simulate_edl, we track each function call: the flow splits to cover edl_events, update_edl_state, and the solve_ivp function. From solve_ivp, we see the call to edl_dynamics, which itself branches out to get_mass_edl, F_gravity_descent, F_buoyancy_descent, and F_drag_descent. In turn, F_gravity_descent calls get_mass_edl again, which then leads to two final functions, concluding this branch. Meanwhile, both F_buoyancy_descent and F_drag_descent further call get_local_atm_properties, completing the flow.

Task 5:



The results in Figure 1 show the effects of parachute diameter on simulation time, rover landing speed, and landing success. To ensure a successful landing, only parachutes with diameters between 16.5 and 19 m are viable, as smaller sizes led to failed landings. Among these, minimizing landing time is preferred, and the first graph confirms that larger diameters increase descent time. Therefore, we recommend a 16.5 m parachute, which provides a safe landing in the shortest time.

Task 6:



6.1)

The continuous model was created by interpolating the data provided. This data came from simulation, meaning that the error is negligible and there is no need for a regression. We chose to use the quadratic interpolation as a representation of the data provided, because it would best extrapolate the data to the range outside the data provided. To generate this plot, we

6.2)

After incorporating the function into the parachute drag model, we re-evaluated our parachute diameter recommendations. By updating the drag coefficient and re-running our analysis, we observed a shift in the recommended parachute diameter range. Our previous recommendation of 16.5 m no longer guaranteed a successful landing, as the range shifted to [17, 19] m. Consequently, we now recommend a 17 m parachute diameter for optimal landing performance within the updated model.