MEEN 357 – Phase 3 Documentation

November 14th, 2022

T-32

Evan Warrick, Konstantin Nelson, Andre Branco

Task 1: Data Dictionary Field Documentation

- define_edl_system:
  - parachute = physical parameters of the parachute and its current deployment state
    - deployed = True, parachute has been deployed (if not then False)
    - ejected = False, parachute has not been ejected (if yes then True)
    - diameter = 16.25 m, parachute diameter
    - Cd = 0.615, parachute drag coefficient
    - Mass = 185 kg, mass of the parachute
  - rocket = physical parameters and activation state of a single rocket
    - on = False, whether engine is on or not (If on then True)
    - structure_mass = 8 kg, mass of rocket minus fuel
    - initial_fuel_mass = 230 kg, mass of initial fuel in rocket
    - fuel_mass = 230 kg, current mass of fuel in rocket
    - effective_exhaust_velocity = 45000 m/s, speed of gases leaving rocket nozzle
    - max_thrust = 3100 N, maximum possible thrust of rocket
    - min_thrust = 40 N, minimum possible thrust of rocket when on
  - speed_control = describes the input parameters of the rockets speed controller
    - on = False, shows whether controller is activated (if yes then True)
    - Kp = 2000, proportional gain of controller
    - Kd = 20, derivative gain of controller
    - Ki = 50, integral gain of controller
    - target_velocity = -3 m/s, target velocity input for controller
  - position_control = describes input parameters of rocket position controller
    - on = False, shows whether controller is activated (if yes then True)
    - Kp = 2000, proportional gain of controller
    - Kd = 1000, derivative gain of controller
    - Ki = 50, integral gain of controller
    - Target_altitude = 7.6 m, target altitude input for controller
  - sky_crane = describes physical parameters and activation state of the sky crane

        - on = False, describes whether sky crane has been deployed (if deployed then True)
        - danger_altitude = 4.5 m, minimum altitude for deploying crane
        - danger_speed = -1 m/s, with positive speed going upward from surface, describes then maximum speed with which the rover is lowered to the surface
        - mass = 35 kg, mass of the sky crane
        - area = 16 m^2, drag inducing area of sky crane
        - Cd = 0.9, drag coefficient of sky crane
        - max_cable = 7.6 m, maximum length of sky crane cable
        - velocity = -0.1 m/s, speed at which crane lowers the rover
    - heat_shield = describes physical parameters and activation state of the heat shield
        - ejected = False, describes whether the heat shield is ejected or not (if yes then True)
        - mass = 225 kg, mass of the heat shield
        - diameter = 4.5 m, diameter of heat shield
        - Cd = 0.35, the drag coefficient of the heat shield
    - edl_system = dictionary that combines all dictionaries above into single dictionary. Also describes the current state of rover as calculated in the simulation.
        - altitude = updated system variable (m)
        - velocity = updated system variable (m/s)
        - num_rockets = 8, parameter that describes the number of rockets used in descent
        - volume = 150 m^3, total volume of EDL system entering atmosphere
        - parachute = packaged dictionary seen above
        - heat_shield = packaged dictionary seen above
        - rocket = packaged dictionary seen above
        - speed_control = packaged dictionary seen above
        - position_control = packaged dictionary seen above
        - sky_crane = packaged dictionary seen above
        - rover = define_rover_4, describes which current rover configuration is being used in simulation.
- define_mission_events:
    - mission_events = describes the selected altitudes for which each stage of the descent will start
        - alt_heatshield_ejet = 8000 m, describes altitude at which heat shield will be ejected (ejected = True)
        - alt_parachute_eject = 900 m, describes altitude at which parachute will be ejected (ejected = True)

        - alt_rockets_on = 1800 m, describes altitude at which rockets will be turned on (on = True)
        - alt_skycrane_on = 7.6 m, altitude at which sky crane will lower the rover to the surface (on = True)
- define_planet:
    - high_altitude = describes temperature/pressure curve at high martian altitude
        - temperature = -23.4 - 0.00222*altitude C, High altitude temp. curve
        - pressure = 0.699*np.exp(-0.00009*altitude) KPa, High altitude pressure curve
    - low_altitude = describes temperature/pressure curve at low martian altitude
        - temperature = -31 - 0.000998*altitude C, Low altitude temp. curve
        - pressure = 0.699*np.exp(-0.00009*altitude) KPa, Low altitude pressure curve
    - density = describes air density curve along martian atmosphere
        - density = pressure/(0.1921*(temperature+273.15) kg/m^3
    - mars = describes physical parameters of mars seen above into single dictionary and includes gravitational acceleration.
        - g = -3.72 m/s^2, acceleration due to gravity on Mars
        - altitude_threshold = 7000 m, altitude at which atmosphere begins
        - low_altitude = packaged dictionary seen above
        - high_altitude = packaged dictionary seen above
        - density = packaged dictionary seen above
- define_rovers:
    - physical parameters of the rover used for dynamic analysis. This is repeated for 4 different rover configurations (define_rover_1, define_rover_2, etc.) and values will be listed for each one in order for each parameter field.
    - wheel = physical parameters of the rover wheels
        - radius = 0.3, 0.3, 0.3, 0.2 m, rover wheel radius
        - mass= 1, 2, 2, 2 kg, single rover wheel mass
    - speed_reducer = description and parameters of the rover speed reducer
        - type = reverted, reverted, standard, reverted, describes the type of speed reducer used
        - diam_pinion = 0.04, .04, .04, .04 m, diameter of reducer pinion gear
        - diam_gear = 0.07, .06, .06, .06 m, diameter of reducer driver gear
        - mass = 1.5, 1.5, 1.5, 1.5 kg, mass of speed reducer
    - motor = dynamic parameters of rover motor
        - torque_stall = 170, 180, 180, 165 N*m, torque on motor which causes stall
        - torque_noload = 0, 0, 0, 0 N*m, minimum applied motor torque

        - speed_noload = 3.8, 3.7, 3.7, 3.85 m/s, max rover speed with no applied reduction torque
        - mass = 5, 5, 5, 5 kg, motor mass
    - chassis = physical parameters for rover body
        - mass = 659, 659, 659, 674 kg, mass of rover body
    - science_payload = physical parameters of the onboard payload (not part of rover frame)
        - science_payload = 75, 75, 75, 80 kg, mass of extra payload on rover
    - power_subsys = physical parameters of the rover powering subsystem
        - mass = 90, 90, 90, 100 kg, mass of rover powering system
    - wheel_assembly = packaged dictionaries seen above that pertain to the wheel assembly on the rover
        - wheel = packaged dictionary seen above
        - speed_reducer = packaged dictionary seen above
        - motor = packaged dictionary seen above
    - rover = packaged dictionaries seen above that pertain to the body of the rover
        - wheel_assembly = packaged dictionary seen above
        - chassis = packaged dictionary seen above
        - science_payload = packaged dictionary seen above
        - power_subsys = packaged dictionary seen above
    - planet = physical description parameters of Mars
        - g = 3.72 m/s^2, gravitational acceleration on Mars
    - end_event = describes the parameters for the final stage of the landing, rover touch down and sky crane flies away. Only used after rover configuration 4.
        - max_distance = 10 m, max. distance of rover from Landing spot
        - max_time = 10000 sec, time to run simulation for after landing
        - min_velocity = 0.01 m/s, minimum travel velocity of the rover.

Task 2: Subfunction Documentation

- get_mass_rover:
    - Calling syntax: returns in float form and called on to perform calculation for another function or to obtain stand-alone variable value.
    - Description: Uses rover parameter dictionaries to add up the total weight of each wheel and multiplies it by the number of wheels. Then adds the chassis weight to get the total weight of unloaded rover. Used in dynamic calculations for rover acceleration in edl_system function.
    - Input arguments:
        - edl_system = Dictionary, sub dictionaries with rover and EDL system physical parameters.
    - Output arguments:

        - m = float, describes the total mass of the rover body and wheels (kg)
- get_mass_rockets:
    - Calling syntax: returns in float form and called on to perform calculation for another function or to obtain stand-alone variable value.
    - Description: calls on the dictionaries containing the rocket parameters. Pulls the fields that contain the appropriate masses and number of rockets and sums it up to obtain a total weight of all rockets.
    - Input arguments:
        - edl_system = Dictionary, sub dictionaries with rover and EDL system physical parameters
    - Output arguments:
        - m = float, total mass of the rockets on the edl system (kg)
- get_mass_edl:
    - Calling syntax: returns in float form and called on to perform calculation for another function or to obtain stand-alone variable value.
    - Description: sums the mass of the heat shield, sky crane, and parachute by pulling values found the fields of the edl dictionaries. Has statements to show changes which depend on if there has been any ejections. Used to add up total system mass during descent.
    - Input arguments:
        - edl_system = Dictionary, sub dictionaries with rover and EDL system physical parameters
    - Output arguments:
        - m = float, mass of the main components of EDL system listed above (kg)
- get_local_atm_properties:
    - Calling syntax: Used in calculation of atmospheric properties which are used in drag and buoyancy calculations. These are then used in the dynamics calculations of applied loads on the rover system. These properties may also be used in tracking how the values change as the altitude changes.
    - Description: Takes in curve functions from dictionaries in the planet properties at high and low altitudes. Contains If-Else statements to separate high from low altitude calculations. Performs this for temperature, pressure, and air density.
    - Input arguments:
        - planet = dictionary, contains any physical parameters of planet that affects the physics of the rover (gravity, altitude thresholds, temp. and pressure curves)
        - altitude = integer/float, input value that changes for each point where a dynamic calculation must be performed. Likely looped through multiple values to assess different stages (m)
    - Output arguments:

        - density = float, air density at point of altitude input (kg/m^3)
        - temperature = float, air temperature at point of altitude input (C)
        - pressure = float, air pressure at point of altitude input (KPa)
- F_buoyancy_descent:
    - Calling syntax: called during force calculation at a specific altitude that is used in the dynamic modeling of the rover.
    - Description: uses the previous function to calculate the buoyancy force of the rover at an altitude. Performed by using the temperature, pressure, and density inputs in simple equation calculation.
    - Input arguments:
        - edl_system = dictionary, contains physical parameters of all edl system components
        - planet = dictionary, contains planetary and atmospheric curves and physical parameters
        - altitude = float, input altitude for point of dynamic calculation (m)
    - Output arguments:
        - F = float, value in Newtons of the buoyancy force on the rover (N)
- F_drag_descent:
    - Calling syntax: called during drag force calculation at a specific altitude that is used in the dynamic modeling of the rover.
    - Description: uses the atmospheric conditions function to calculate the drag force of the rover at an altitude. Performed by using the temperature, pressure, and density inputs in conjunction with the velocity at that point to calculate drag. Includes If-Else to differentiate whether or not the heat shield, parachute, and sky crane are ejected/deployed.
    - Input arguments:
        - edl_system = dictionary, contains physical parameters of all edl system components
        - planet = dictionary, contains planetary and atmospheric curves and physical parameters
        - altitude = float, input altitude for point of dynamic calculation (m)
        - velocity = float, current velocity of rover at specified altitude. Based on previous dynamic calculation (m/s)
    - Output arguments:
        - F = float, value of the drag force on the descending rover (N)
- F_gravity_descent:
    - Calling syntax: called during drag force calculation at a specific altitude that is used in the dynamic modeling of the rover. Can be used in stand-alone variable calculation or within another function.

    - Description: Uses the dictionary input of the gravitational acceleration on Mars in conjunction with the calculated mass of the edl system to obtain a weight.
    - Input arguments:
        - edl_system = dictionary, contains physical parameters of all edl system components
        - planet = dictionary, contains planetary and atmospheric curves and physical parameters
    - Output arguments:
        - F = float, force due to gravity (weight) of the edl system (N)
- v2M_Mars:
    - Calling syntax: used to calculate a specific threshold number that can be used in a larger function as a cut-off point in an If-else statement.
    - Description: Converts the speed of the rover in m/s to a Mach number based on the calculated speed of sound (based on planetary properties and atmospheric properties calculated at that altitude). This can differentiate supersonic from subsonic flight.
    - Input arguments:
        - v = float, input argument for speed at specified altitude (m/s)
        - a = float, input altitude at which Mach number is calculated (m)
    - Output arguments:
        - M = float, rover system Mach number
- thrust_controller:
    - Calling syntax: Used consistently functions where calculation of the rocket thrust output are performed. Likely called within another function and looped through to change the input values to obtain target thrust and rover speeds.
    - Description: Takes in the controller gains found within the rocket controller dictionary fields and calculates adjustments to the rocket thrust. Includes multiple If-else statements to check situational instances such as (exceed max thrust, engine on/off, approaching max speed). First calculated thrust using model and then uses the controller gains to make adjustments.
    - Input arguments:
        - edl_system = dictionary, contains physical parameters of all edl system components
        - planet = dictionary, contains planetary and atmospheric curves and physical parameters
    - Output arguments:
        - edl_system = dictionary, updated edl system dictionary to obtain new engine on/off state and required thrust values.
- edl_events:

  - Calling syntax: called on whenever the rover system reaches a new altitude to trigger a stage change. Used in updated edl_system values (similarly to how the function above does it) so will only be used for dictionary field value updates.
  - Description: Starts a new event whenever it sees that a certain target altitude has been reached. This changes the field values in the edl system dictionary that corresponds with what the system will turn on/off or deploy/eject. Repeats this for all 8 events of the descent.
  - Input arguments:
    - edl_system = dictionary, contains physical parameters of all edl system components
    - mission events = dictionary, contains all altitudes that specifically indicate a new stage of descent (values in meters)
  - Output arguments:
    - Events = array, shows all events in a list and which ones have been completed
- edl_dynamics:
  - Calling syntax: used in the velocity, acceleration, fuel mass, and position (state) calculations of the rover and edl system. Called on in functions to calculate new values for state variables.
  - Description: Takes in previous values for state variables and applies the proper changes to the dictionary field values for that specific altitude. Sums up all forces on the rover calculated in previous functions and uses If-else statements to assess new changes in state variables based on whether there have been any changes to the input forces calculated.
  - Input arguments:
    - t = float, time of at the point of simulation. Used to calculated changes in time (sec)
    - y = array, contains values for all current state variables (position, velocity, acceleration) and is used to calculate state changes
    - edl_system = dictionary, contains physical parameters of all edl system components
    - planet = dictionary, contains planetary and atmospheric curves and physical parameters
  - Output arguments:
    - Dydt = array, contains the value of the changes in the state of the edl systems such as changes in: altitude, total mass, velocity, rover acceleration and velocity relative to the sky crane, and error signals)
- update_edl_state:

    - Calling syntax: used within functions to update the values of different dictionaries. Also used to check whether output would require another function to change the system parameters.
    - Description: Uses large If-Else loops to show what changes about the system state at each stage of descent. If a certain check shows that an event has been reached, this is where the edl state is changed based on changes of parameters.
    - Input arguments:
        - edl_system = dictionary, contains physical parameters of all edl system components
        - TE = array, time of event at current point of flight
        - YE = array, state at current event
        - Y = array, final state conditions from previous iteration
        - ITER_INFO = list, contains strings and values that describes the current iteration state.
    - Output arguments:
        - edl_system = dictionary, contains updated physical parameters of all edl system components
        - y0 = array, new values for the current state of the edl system
        - TERMINATE_SIM = logic output, True/False output of whether to keep the simulation running or not.
- simulate_edl:
    - Calling syntax: Used in main calculation script to run through the simulation. Return values can be used as array for graphing or for updating dictionary values.
    - Description: Combines all functions to calculate the state of the system at each point in time. Takes into account changes in atmosphere, mass, and ejections at each stage. Uses ODE solver to calculate dynamics changes as a result of new forces input values and records the values at each step for graphing later.
    - Input arguments:
        - edl_system = dictionary, contains updated physical parameters of all edl system components
        - planet = dictionary, contains planetary and atmospheric curves and physical parameters
        - mission events = dictionary, contains all altitudes that specifically indicate a new stage of descent (values in meters)
        - tmax = float, describes maximum time to run simulation for (sec)
        - ITER_INFO = list, contains strings and values that describes the current iteration state.
    - Output arguments:
        - T = array, contains all times to run simulation at (sec)
        - Y = array, contains the state of the edl system at each point in time. Used for graphing.

- edl_system = dictionary, contains final updated physical parameters of all edl system components

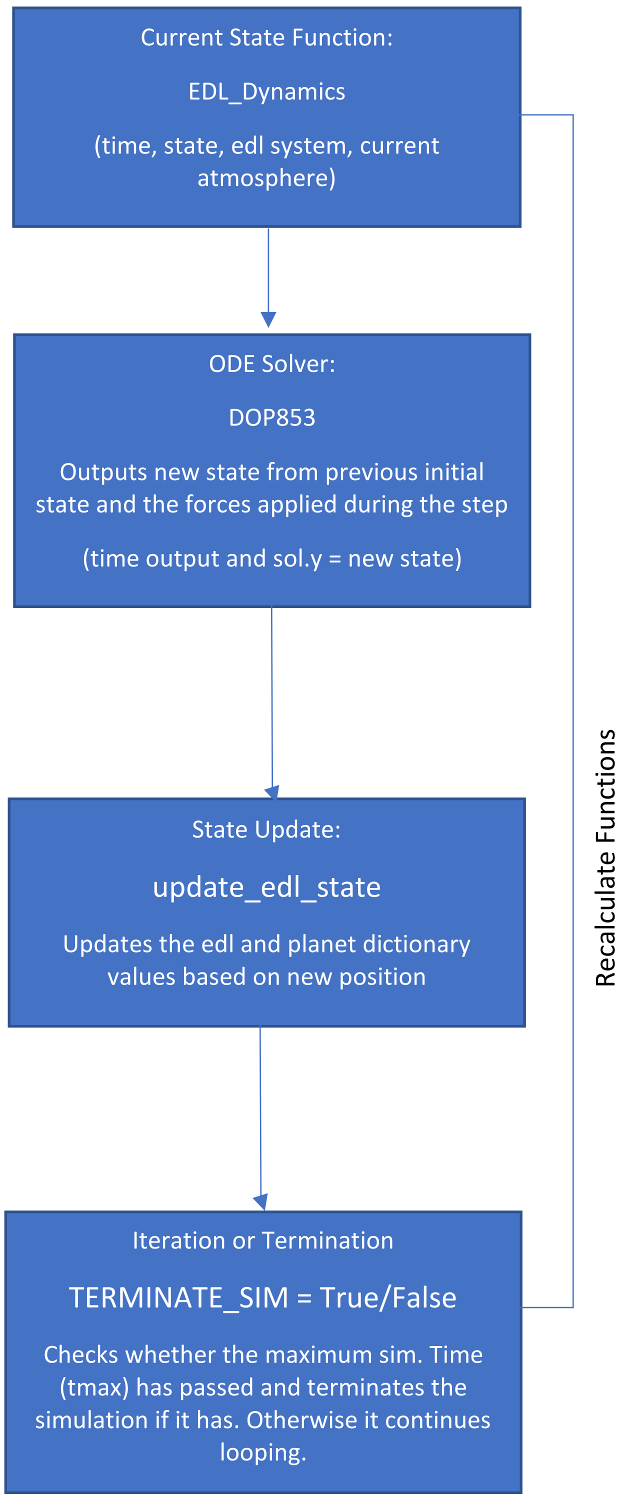
Task 3: Simulate_EDL Loop Documentation

First calls on the edl_dynamics function to calculate the current forces on the rover based on its current state.

Uses ODE Solver DOP853 to apply the new calculated Loads in the dynamics function and to determine the New state of the rover based on changes in mass, Acceleration, and atmosphere. Outputs arrays with New state and time step information.

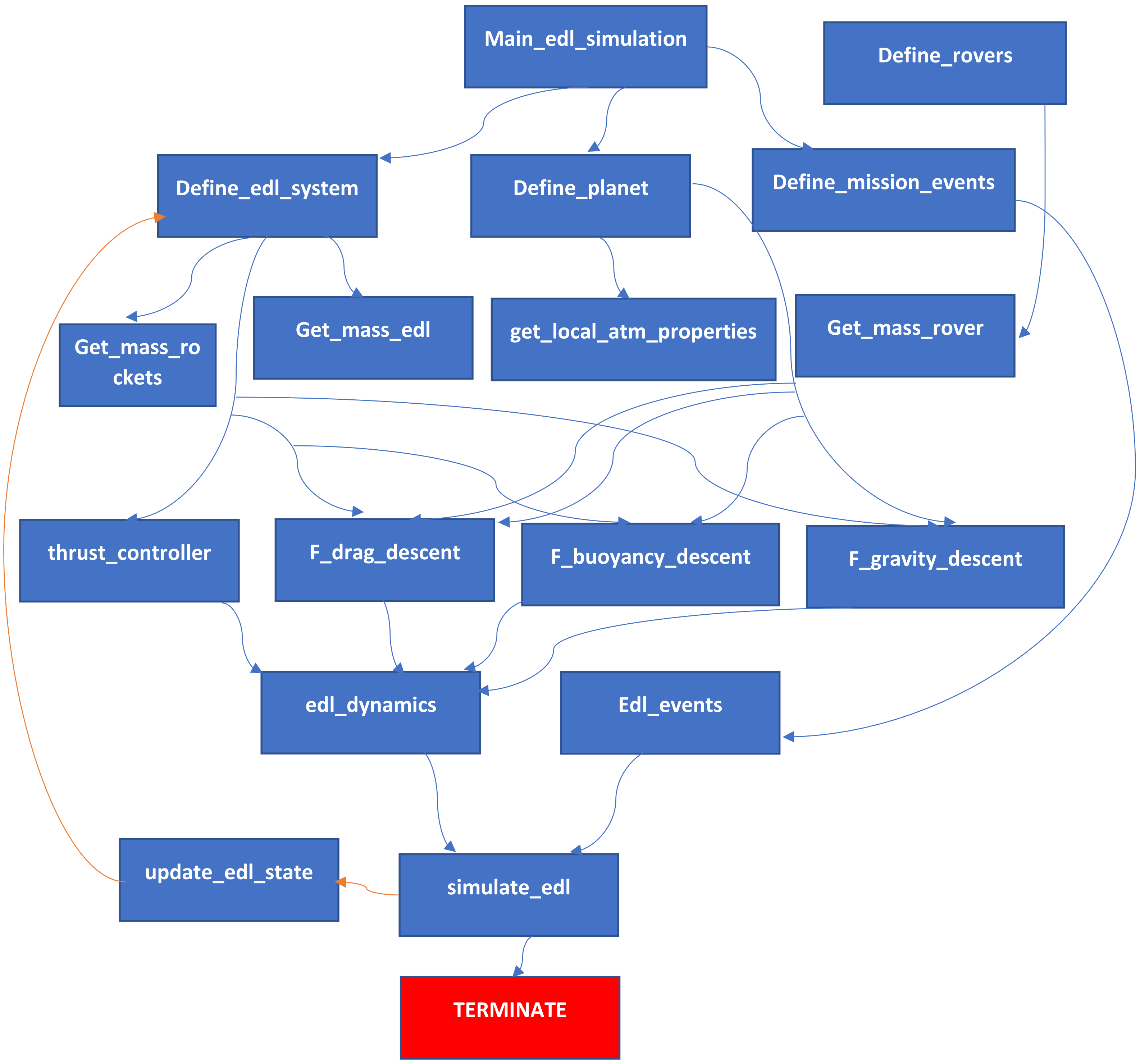
Calls on the update_edl_state function to go back Through all of the edl and planetary dictionaries and Makes the corresponding changes based on conditions Stated within the previous dynamic calculation functions.

After updates have been made to the dictionaries, the Code checks if the maximum run time has been passed For the simulation and will stop the simulation if it has. If not, all functions will be recalculated with new state Values and the dynamics will start with initial conditions


Current State Function:
EDL_Dynamics
(time, state, edl system, current atmosphere)
ODE Solver:
DOP853
Outputs new state from previous initial state and the forces applied during the step
(time output and sol.y = new state)
State Update:
update_edl_state
Updates the edl and planet dictionary values based on new position
Iteration or Termination
TERMINATE_SIM = True/False
Checks whether the maximum sim. Time (tmax) has passed and terminates the simulation if it has. Otherwise it continues looping.
Recalculate Functions

That were the final conditions of the previous iteration.

Task 4: Execution Flow Documentation



General Description: This system shows what happens as the script runs from one iteration of main script being run. The flow begins by the code retrieving state and parameter data from the "define" dictionaries and using those values to perform basic mass and atmospheric condition calculations. These are then used in calculating the applied forces on the rover and used in the dynamics script to obtain a function to be applied in the ODE solver. Concurrently, the stage of descent is being checked in the EDL events function and will make the appropriate ejections for the iteration. All of this is then fed into the simulate EDL function where the ODE solver will output the new state of the system. This information is then fed into update_edl_state so that the next iteration begins with the newest state and atmosphere conditions. Alternatively, if the simulate EDL function sees that the simulation time has exceeded the maximum allowed (user inputted time), then the code will terminate and the state values will be plotted from the main script.

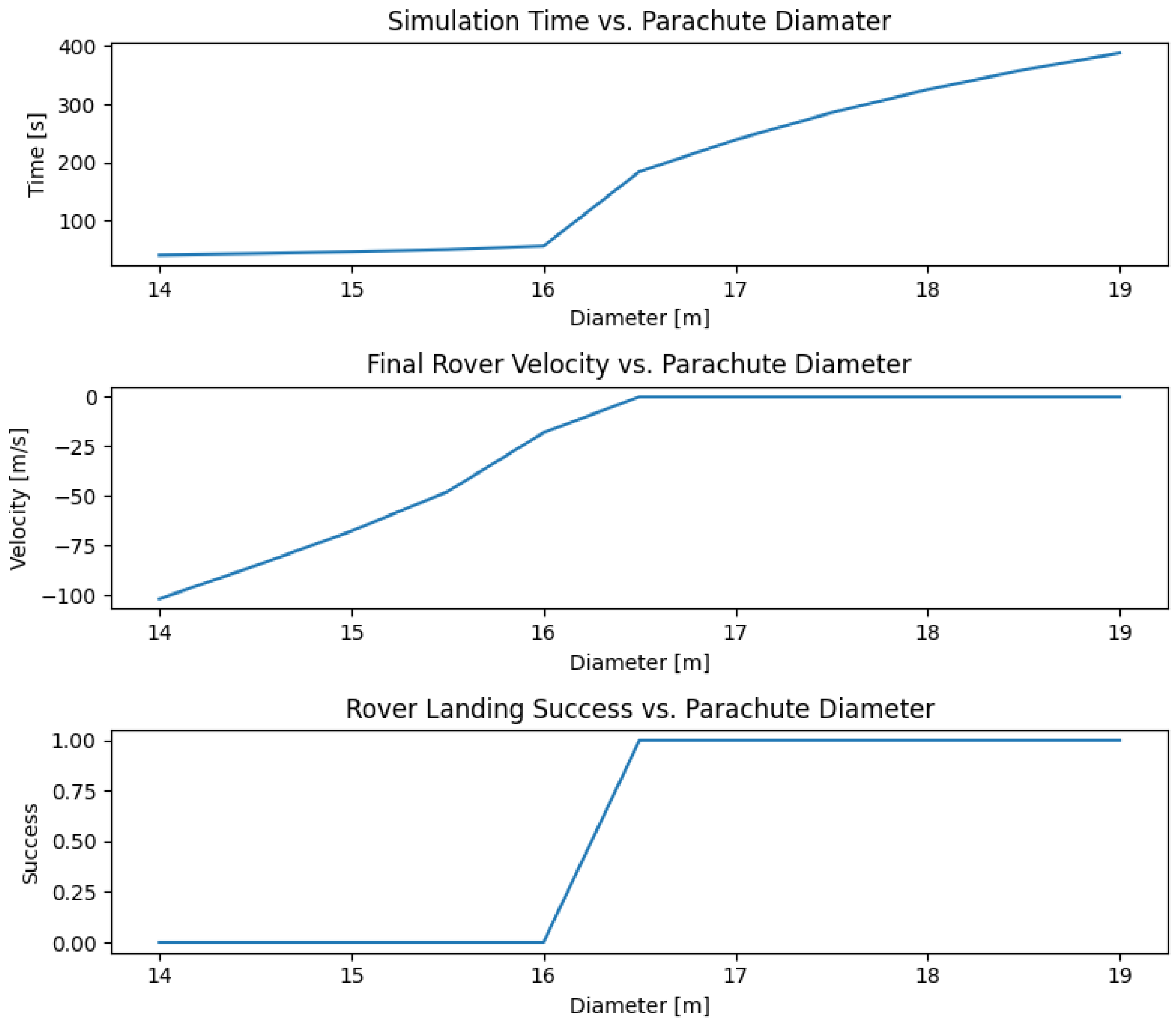
Task 5: Evaluate the Impact of Changing the Parachute Size

As the diameter of the parachute increased, the simulation time increased starting when the diameter was 16.5 meters. As it increased further, the simulation time tapered off, following a logarithmic curve. From the diameter range of 14 meters to 16 meters, the diameter of the parachute did not cause enough drag force for the rover to land safely. Instead, the rover fell to the surface before its velocity could reach a safe level.

As the diameter of the parachute increased, the magnitude of the rover's velocity, decreased. At a diameter of 16.5 meters and onwards, the velocity of the rover arbitrarily approached zero. Since a larger parachute would result in a greater drag force on the rover, the magnitude of the velocity of the rover would decrease more quickly, eventually slowing the rover enough for a safe landing. For diameters less than 16.5 meters, the forces of the parachute, buoyancy, and thrust were not enough to counteract the weight and velocity of the rover before it reached the sky crane phase.

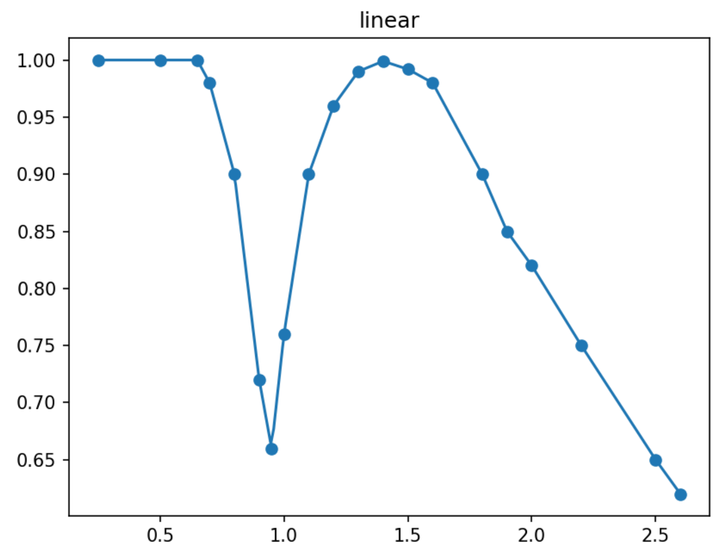
The simulation achieved success for diameter values greater and equal to 16.5 meters and failed for values less than 16.5 meters. Lower diameter values were not sufficient to achieve the critical rover landing speed of 1 m/s with a sky crane altitude of no less than 4.5 meters.

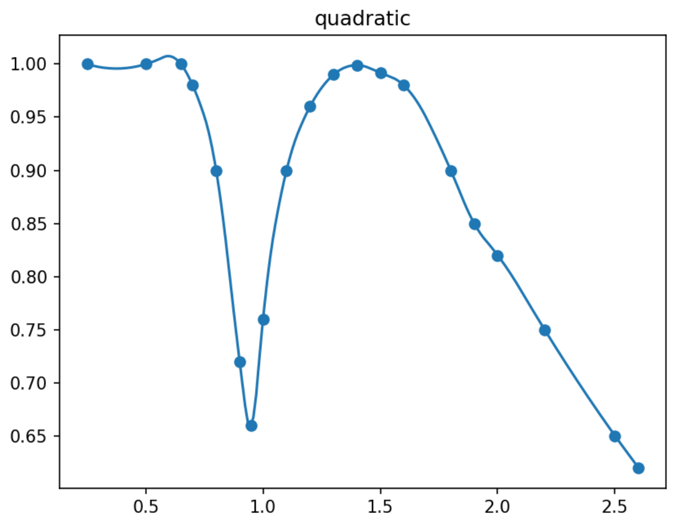
Considering the effects of rover speed, while minimizing simulation time and parachute diameter, the optimal value of parachute diameter is 16.5 meters. A diameter of 16.5 meters was the lowest value in the range of successful diameters. Although each successful diameter value produced an approximately equal final rover speed, a diameter of 16.5 meters resulted in a minimal simulation time due to its balance of drag force: slowing down the rover enough to reach its critical speed while not slowing it too quickly.

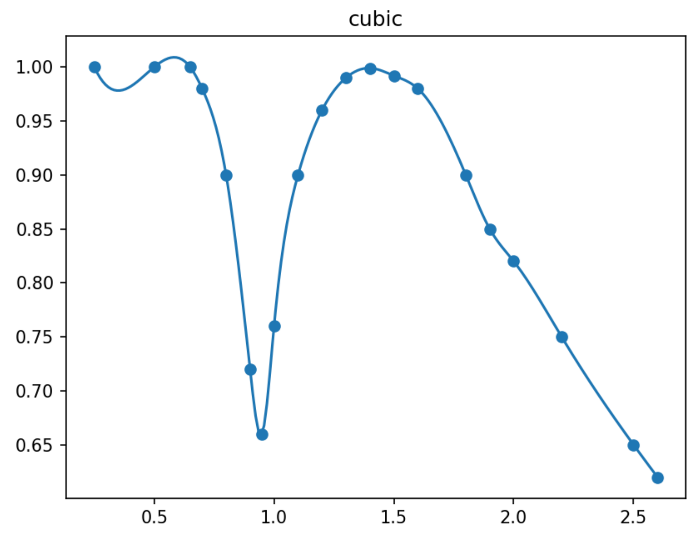


Task 6: Improve the Parachute Drag Model

To create a continuous model of the MEF data, we interpolated all of the points using scipy interp1d. To decide which method to use, we plotted each of them below:





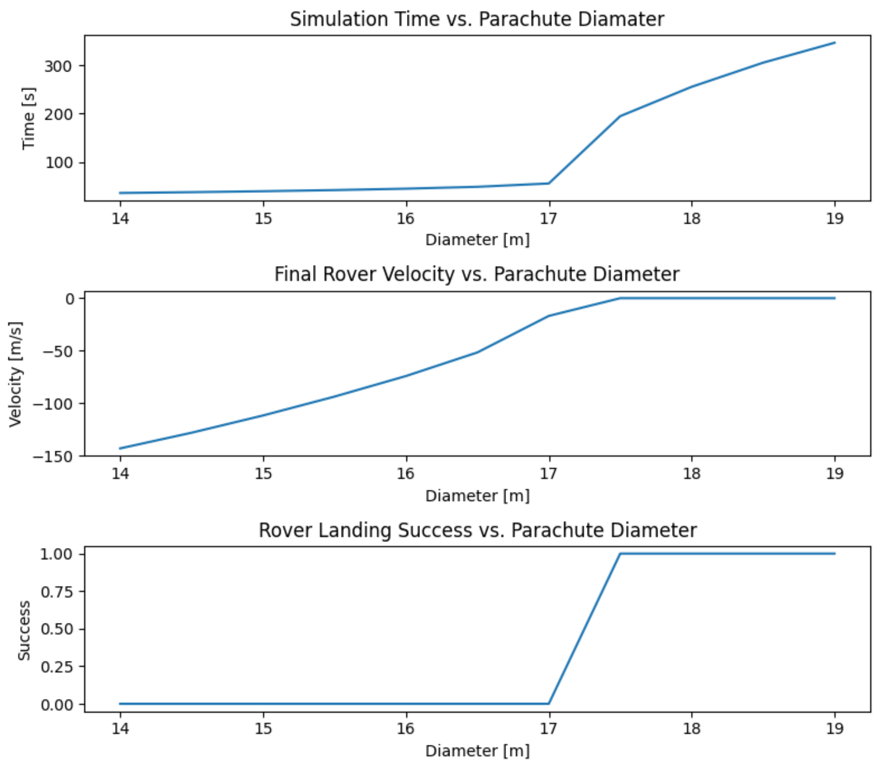


As can be seen with the plots, quadratic and cubic give the best estimations above Mach 0.7 but distort the MEF from Mach 0.25 to 0.7. Linear does not distort the MEF below Mach 0.7 but does not interpolate the rest of the graph as well. So, we decided to use quadratic fit only for the range of 0.7 to 2.6, and not calculate the MEF outside of this range. This has the effect of not distorting the MEF below Mach 0.7 but giving a good fit from Mach 0.7 to 2.6.

The function F_drag_descent() was updated with the following code:

```python
#Mach Effeciency Factor corrention
mach = v2M_Mars(velocity,altitude)
if 0.7<mach and mach <2.6:
    mach_ref = [0.25,0.5,0.65,0.7,0.8,0.9,0.95,1.0,1.1,1.2,1.3,1.4,1.5,1.6,1.8,1.9,2.0,2.2,2.5,2.6]
    MEF_ref = [1.0,1,1,.98,.9,.72,.66,.76,.9,.96,.99,.999,.992,.98,.9,.85,.82,.75,.65,.62]
    fun = interp1d(mach_ref, MEF_ref,kind='quadratic',)
    MEF = fun(mach)
    ACd_body *= MEF
    ACd_parachute *= MEF
```

The updated parachute study with the Mach efficiency factor correction is shown below:



Using this new plot, we assessed that the parachute diameter recommendations needed to be updated. The new recommendations for parachute diameter are from 17.5 to 19 meters, depending on risk tolerance.