1. Function Name: get\_mass\_rover
   1. Calling Syntax: get\_mass\_rover(edl\_system)
   2. Description: This function computes the mass of the rover defined in the rover field of the edl system structure. The mass is calculated as follows: 6 \* motor mass + 6 \* speed reducer mass + 6 \* wheel mass + chassis mass, science payload mass + power subsystem mass. The input, edl\_subsystem, must be a dictionary.
   3. Input Arguments:
      1. edl\_subsystem
         1. Type: Dictionary
         2. Definition: A data structure describing characteristics of the entire EDL system
   4. Output Arguments:
      1. m
         1. Type: Scalar
         2. Definition: A positive scalar stating the total mass of the rover
         3. Units: Kilograms (Kg)
2. Function Name: get\_mass\_rockets
   1. Calling Syntax: get\_mass\_rockets(edl\_system)
   2. Description: This function computes and returns the mass of all the rockets on the edl system. The mass is calculated as follows: number of rockets \* structural mass of rockets + the rocket’s fuel mass. The input, edl\_subsystem, must be a dictionary.
   3. Input Arguments:
      1. edl\_subsystem
         1. Type: Dictionary
         2. Definition: A data structure describing characteristics of the entire EDL system
   4. Output Arguments:
      1. m
         1. Type: Scalar
         2. Definition: A positive scalar stating the total mass of the rockets on the EDL System
         3. Units: Kilograms (Kg)
3. Function Name: get\_mass\_edl
   1. Calling Syntax: get\_mass\_edl(edl\_system)
   2. Description: This function computes and returns the total current mass of the EDL system. To do this, it checks if the parachute and/or the heat shield is still attached. If not, the mass of these components is removed from the returned value. In addition to the mass of the parachute and heat shield, the function sums the mass of the rockets (using get\_mass\_rockets), the mass of the rover (using get\_mass\_rover), and the mass of the sky crane system. The input, edl\_subsystem, must be a dictionary
   3. Input Arguments:
      1. edl\_subsystem
         1. Type: Dictionary
         2. Definition: A data structure describing characteristics of the entire EDL system
   4. Output Arguments:
      1. m
         1. Type: Scalar
         2. Definition: A positive scalar stating the total current mass of the entire EDL System
         3. Units: Kilograms (Kg)
4. get\_local\_atm\_properties
   1. Calling Syntax: get\_local\_atm\_properties(planet, altitude)
   2. Description: Returns the local atmospheric properties at a given altitude. This function returns a temperature and a pressure of the atmosphere depending on if the current altitude of the EDL module is above or below a set threshold. The specific altitude is inputted into a function which then returns a dictionary of temperature and pressure. Once these pressure and temperature values are determined, an air density value is also calculated and returned. The input planet must be a dictionary. The input altitude must be a scalar and cannot be inputted as a vector. The function returns the pressure, temperature, and density as a tuple, in that order. It is important to note, however, that the pressure, temperature, and density can be stored independently of the other 2
   3. Input Arguments:
      1. planet
         1. Type: Dictionary
         2. Definition: A data structure describing characteristics of the planet that the EDL is entering, descending, or landing on.
      2. altitude
         1. Type: Scalar
         2. Definition: The number of meters the EDL Module is away from the planet’s surface
         3. Units: meters (m)
   4. Output Arguments:
      1. density
         1. Type: Scalar
         2. Definition: A scalar describing the air density of the planet’s atmosphere at a pressure and temperature derived from the altitude of the EDL module.
         3. Units: Kilograms (Kg)
      2. temperature
         1. Type: Scalar
         2. Definition: A scalar describing the temperature based on the planet’s atmospheric data and the distance above the planet’s surface.
         3. Units: Celsius (°C)
      3. pressure
         1. Type: Scalar
         2. Definition: A scalar describing the atmospheric pressure based on the planet’s atmospheric characteristics and a specified altitude
         3. Units: Kilopascals (KPa)
5. F\_buoyancy\_descent
   1. Calling Syntax: F\_buoyancy\_descent(edl\_system, planet, altitude)
   2. Describe: This function utilizes the density from get\_local\_atm\_properties, the accelertation due to gravity from the planet’s dictionary, and the volume of the EDL System from its data structure. These values are utilized to calculate the buoyancy force which is pointing strictly upwards. The formula to calculate this force is density \* acceleration due to gravity \* volume of the system. This function is not vectorized as the helper function, get\_local\_atm\_properties, does not output vectors.
   3. Input Arguments:
      1. edl\_system
         1. Type: Dictionary
         2. Definition: A data structure describing characteristics of the entire EDL system
      2. planet
         1. Type: Dictionary
         2. Definition: A data structure describing characteristics of the planet
      3. altitude
         1. Type: Scalar
         2. Definition: Distance from planet’s surface to EDL Module at that time
         3. Units: meters (m)
   4. Output Arguments
      1. F
         1. Type: Scalar
         2. Definition: The buoyancy force exerted on the EDL system in the upward direction and calculated by density \* acceleration due to gravity \* volume of system
         3. Units: Newtons (N)
6. F\_drag\_descent
   1. Calling Syntax: F\_drag\_descent(edl\_system, planet, altitude, velocity)
   2. Describe: This function calculates and returns the drag force on the EDL Module. The drag force is defined as 0.5 \* density \* EDL velocity^2 \* area \* drag coefficient. The drag coefficient and area depend on whether the heat shield has been ejected or not. Furthermore, this function also calculates the drag force based on the status of the parachute and if it has been deployed or ejected. The function considers the drag force for all these possibilities. It is important to note that this function is not vectorized because the helper function, get\_local\_atm\_properties cannot return a vector.
   3. Input Arguments
      1. edl\_system
         1. Type: Dictionary
         2. Definition: A data structure describing characteristics of the entire EDL system
      2. planet
         1. Type: Dictionary
         2. Definition: A data structure describing characteristics of the planet
      3. altitude
         1. Type: Scalar
         2. Definition: Distance from planet’s surface to EDL Module at that time
         3. Units: meters (m)
      4. velocity
         1. Type: Scalar
         2. Definition: The velocity of the EDL System relative to the air that it is flying through.
         3. Units: meters per second (m/s)
   4. Output Arguments
      1. F
         1. Type: Scalar
         2. Definition: The ultimate drag force created from the sum of the air drag and either the parachute, heat shield, or sky crane module drag.
         3. Units: Newtons (N)
7. F\_gravity\_descent
   1. Calling Syntax: F\_gravity\_descent(edl\_system, planet)
   2. Describe: Computes and returns the gravitational force acting on the EDL System. It is governed by EDL Mass \* acceleration due to gravity.
   3. Input Arguments
      1. edl\_system
         1. Type: Dictionary
         2. Definition: A data structure describing characteristics of the entire EDL system
      2. planet
         1. Type: Dictionary
         2. Definition: A data structure describing characteristics of the planet
   4. Output Arguments
      1. F
         1. Type: Scalar
         2. Definition: Gravitational force acting on the EDL system
         3. Units: Newtons (N)
8. v2M\_Mars
   1. Calling Syntax: v2M\_Mars(v, a)
   2. Describe: This function converts a velocity, inputted in m/s, to a Mach number on Mars as a function of altitude. It utilizes the data points for the speed of sound vs. altitude and creates a fit around these points. It is important to note that the function will return the absolute value Mach number.
   3. Input Arguments
      1. v
         1. Type: Scalar or Vector
         2. Definition: The velocity of the EDL as a scalar or a list
         3. Units: meters per second (m/s)
      2. a
         1. Type: Scalar or Vector
         2. Definition: The current altitude of the EDL with respects to the planet’s surface as a scalar or a vector
         3. Units: meters (m)
   4. Output Arguments
      1. M
         1. Type: Scalar or Vector
         2. Definition: The Mach number on Mars
         3. Units: dimensionless
9. thurst\_controller
   1. Calling Syntax: thrust\_controller(edl\_system, planet)
   2. Describe: This function implements a PID controller for the EDL System. It uses edl\_system, a data structure characterizing the EDL System, and planet, a data structure to characterize the planet, to create a modified edl\_system data structure. It also modifies fields in rocket and telemetry sub data structures.
   3. Input Arguments:
      1. edl\_system
         1. Type: Dictionary
         2. Definition: A data structure describing characteristics of the entire EDL system.
      2. planet
         1. Type: Dictionary
         2. Definition: A data structure describing characteristics of the planet.
   4. Output Arguments
      1. edl\_system
         1. Type: Dictionary
         2. Definition: Edited edl\_system data structure with edited rocket and telemetry sub data structures involving
         3. STILL TO FINISH