MEEN 357 DESIGN PROJECT
PHASE 4: DESIGN OF THE EDL + ROVER SYSTEM

Assigned:		**Submission:**	Electronic
Due Date:	Multiple – see Canvas	**Collaboration Type:**	Project Teams
		Grading:	35 points

NOTE: NO COMMUNICATION BETWEEN GROUPS IS ALLOWED

1 PHASE 4

Phase 4 of your project consists of two parts: the main assignment and an extra-credit challenge. The extra-credit challenge is **not optional**—your team must submit a design. The main assignment is relatively open-ended. Your team will be graded based upon how sensible your approach to the problem is and how well you explain what you did.

1.1 Main Phase 4 Assignment

In this phase, you will design an edl+rover system. You should use every trick in your arsenal—visualization, optimization, common sense, maybe even uncommon sense—to design your edl+rover system according to the following problem statement.

> *After an encouraging meeting with the VP of R&D, your team is given the green light to develop a more complete design specification for the entry, descent and landing system as well as the rover. You are told that the rover should be able to travel at least 1 km on one charge of the batteries over terrain for which you already have some topographic information (in experiment1). Finally, you are told that total cost and speed are major design considerations, with rover speed being the more important of the two. You may also consider battery usage as an important consideration.*

Your team must interpret this problem and produce a design that you believe best solves the problem. You may use any technique we have studied in MEEN 357 to arrive at a solution. Your goals are to (a) come up with a design and (b) develop a compelling explanation of your rationale for choosing this design (for your report back to the VP).

All of your analyses are to be conducted using the code distributed with this assignment. The contents of this distribution are documented later in this report. It is an extension of the Phase 3 code distribution to include things like battery selections, motor selections, and a cost model as well as all Phase 2 code.

Your final deliverables are as follows (detailed submission instructions are at the end of this document):

1. A final design for your EDL System and Rover. This is to be submitted as a data file using 'pickle' in Python. The file is to be named **`FA25_SecYY_TeamXX.pickle`**, where XX is your team number and YY is your section number. This data file should contain a dictionary that is identical to what you get if you use the **`define_edl_system`** function, except you should change certain fields to reflect your desired design solution (so if you want a different parachute diameter, you change that field). Note, you will be given code that will save this in the appropriate format for you.
2. A report, submitted in a file named **`FA25_SecYY_TeamXX.pdf`**, where XX is your team number and YY is your section number. This report should contain a complete and professional-quality explanation of your proposed design. This includes
   a. How you formulated the design problem.
   b. Which design solutions you considered.
   c. How you selected your final design.
   d. Which numerical methods you used in this process.
   e. A description of your final design.
   f. Data about its properties/performance.
3. An Individual submission (see Section 5.1 for details)

1.2 Extra Credit Optimization Challenge

Your team must submit a design to compete in the extra-credit optimization challenge. You have one goal for this challenge: to design an EDL system and rover that can land safely and travel along the Martian surface to reach a goal point in the terrain faster than anyone else. **This is due at 6:00 pm on Sunday, December 7. The races will take place in class on Monday, December 8.**

The entry, descent, and landing stages will proceed as they are simulated in the Phase 4 code distribution. The rover stage will proceed similarly to the code you have been given, but **a randomized terrain will be used**. This means your rover will have to be somewhat robust to different terrain scenarios. You may want to test it under different conditions. You can do this easily by changing the terrain angles in the **`define_experiment.py`**.

You must submit your design to Canvas as a data file named **`FA25_SecYY_TeamXX.pickle`**, where XX is your team number. This dictionary should be identical to what you get if you use the **`define_edl_system`** function, except:

1. You should change certain fields to reflect your desired design solution (so if you want a different parachute diameter, you change that field).
2. You should add two new fields to the top level of edl_system:
    a. team_number. Assign it a numerical value equal to your team number.
    b. team_name. A string you can use to give your team a fun name.

Note, the code **`opt_edl_system.py`** is setup to save this file appropriately for you. Just be sure to change the team number and add your team name.

2 PHASE 4 CODE DISTRIBUTION

2.1 Files Provided

The following are the functions, scripts, and data files are provided to you. You are not supposed to modify them in any way.

Function name	*Used first in*	*Function name*	*Used first in*
`battenergy`	Phase 2	`F_gravity`	Phase 1
`get_cost_edl`	Phase 4	`F_net`	Phase 1
`get_cost_rover`	Phase 4	`F_rollingCorr`	Phase 1 - modified
`define_batt_pack`	Phase 4	`get_gear_ratio`	Phase 1
`define_chassis`	Phase 4	`get_local_atm_properties`	Phase 3
`define_edl_system`	Phase 3	`get_mass_edl`	Phase 3
`define_mission_events`	Phase 3	`get_mass_rockets`	Phase 3
`define_motor`	Phase 4	`get_mass_rover`	Phase 3
`define_planet`	Phase 3	`simulate_rover`	Phase 2
`define_rover`	Phase 3	`mechpower`	Phase 2
`define_experiment`	Phase 2	`motorW`	Phase 2
`edl_dynamics`	Phase 3	`rover_dynamics`	Phase 2
`end_of_mission_events`	Phase 2	`simulate_edl`	Phase 3

`F_buoyancy_descent`	Phase 3	`tau_dcmotor`	Phase 1
`F_drag_descent`	Phase 3	`update_edl_state`	Phase 3
`F_drive`	Phase 1	`redefine_edl_system`	Phase 3
`F_gravity_descent`	Phase 3		

(Brief) Description of new files

The following is a brief description of the purpose of each new file. Based on your experiences from Phase 3, you should be capable now of determining any further information about each file you may require to complete your design task.

Function name	***Description***
`get_cost_edl`	edl_system = get_cost_edl(edl_system): Computes cost of the entire edl system (including rover) and updates the corresponding struct fields.
`get_cost_rover`	rover = get_cost_rover(rover): Computes cost of the rover and updates corresponding fields

`define_batt_pack`
`define_chassis`
`define_motor`
	Used to define rover components (these are used to define the motor type, battery pack type, and chassis material for your rover). Usage is defined later in this packet.

`F_rollingCorr`	Corrected rolling resistance for rover. It takes into account the fact that rolling resistance is a function of the wheel radius.

2.2 New and Modified Structures

The following are descriptions of new and modified dictionaries for this phase.

edl_system['rover']['power_subsys']['battery']			
New subdictionary of edl_system['rover']['power_subsys']			
Field Name	*Type*	*Description*	*Example Values*
`battery_type`	String	Type of Battery	'LiFePO4'
`num_modules`	Int	Number of modules	5
`mass`	Float	[kg] Total Mass	3.4860
`cost`	Float	[$] Total Cost	2.25e5
`capacity`	Float	[J] Capacity of Battery	0.9072e5

edl_system['rover']['chassis']			
New subdictionary of edl_system['rover']			
Field Name	*Type*	*Description*	*Example Values*
`type`	String	Chassis material	'steel'
`mass`	Float	[kg] Total mass of chassis	707.2
`specific_strength`	Int/Float	[Nm/kg] Specific strength of chassis material	100
`strength`	Int/Float	[N/A] Metric representing strength of chassis	65900

edl_system['rover']['wheel_assembly']['motor']			
Modified substructure of rover['wheel_assembly'] containing motor parameters			
Field Name	*Type*	*Description*	*Example Values*
`type`	String	Motor type	'torque_he'
`torque_stall`	Float	[Nm] Torque at stall condition	206.25
`torque_noload`	Float	[Nm] Torque at no load condition	0
`speed_noload`	Float	[rad/s] Rotational speed of motor at no load	3.85
`mass`	Float	[kg] Mass of motor	5
`tau_effcy`	nx1 numpy array	tau_effcy[:,0] corresponds to the normalized torque of the motor (ranges from 0 to 1). tau_effcy[:,1] corresponds to the efficiency of the motor at the corresponding torque in tau_effcy[:,0]	
`cost`	Float	Cost of the motor	3.36e5

3 DESCRIPTION OF NEW PHYSICS, DESIGN OPTIONS, AND COST CONSIDERATIONS

3.1 Rolling Resistance

The rolling resistance is now a function of the wheel radius. For further details, please see the implementation in **`F_rollingCorr`**.

3.2 Strength Modeling

There often exists a tradeoff between mass and strength of a structure. For a fixed material (e.g., steel), using more material will make a structure heavier but also able to handle greater loads before failing. In this project, we are concerned with the strength of the rover chassis, since this will need to handle the forces during landing and rover operation.

In general, strength modeling can become very complicated since it depends on geometry, processing, operating conditions, etc., in addition to the quantity of material used. However, for our purposes it is a reasonable first approximation to base our strength estimates strictly on mass. We will use the following model:

Strength = specific_strength*mass_chassis.

Thus, we presume strength increases linearly with the quantity of material used. You will be provided with specific strength information for the materials considered in this project in **`define_chassis`**. You do not need to modify this function in any way.

3.3 Cost Modeling

In real life, an optimal engineering solution is not only about finding the best combination of the components to arrive at the best performance. We must also satisfy technical and programmatic constraints.

You need to consider the cost as a constraint in this design problem. You have been provided with two different functions: **`get_cost_rover`** and **`get_cost_edl.`** These two functions calculate the cost associated with the values of different components such as the rocket fuel, radius of the wheels, gear diameter, mass of the chassis, etc.

3.4 Motor Options

You are provided six different DC electric motors with which you may drive your rover. They all are reasonable for this application, but they differ in terms of their capabilities and other properties. One of your challenges will be to choose the best motor for your rover.

In order to simplify motor selection, you are provided the following function:

Function	Input	Returns
`define_motor`	edl_system motor_type: 'base, base_he, torque, torque_he, speed, speed_he'	edl_system

Motor Types.

Identifier String	Description
`base`	Base motor
`base_he`	Base motor, high efficiency
`torque`	High torque stall
`torque_he`	High torque stall, high efficiency
`speed`	High speed no load
`speed_he`	High speed no load, high efficiency

Using this function is straightforward. Simply call the function with edl_system as well as the motor variant that you want. The function will take the edl_system and return an edl_system with proper motor parameters. You can then add this to a rover dict prior to performing simulation and/or optimization. The following is an example of valid syntax:

```
edl_system = define_motor(edl_system,'base')
```

Under no circumstance should you modify this function.

3.5 Battery Options

You are provided six different types of battery modules. Like with the motors, you must identify the best type of battery pack. However, the batteries have the added complication of you needing to choose the battery type ***and*** the # of modules.

In order to simplify battery pack definition, you are provided with the following function:

Function	Inputs	Returns
`define_batt_pack`	edl_system: dict type: String identifying the battery type num_modules: Positive integer specifying how many modules are in the battery pack	edl_system

No extra data files are required to run this function (everything is defined in the code directly). The valid choices for the battery type string are as follows:

Table 1: Battery types.

Identifier String	Description
`LiFePO4`	36V module made of Lithium-Iron-Phosphate cells
`NiMH`	36V module made of Nickel-Metal-Hydride cells
`NiCD`	36V module made of Nickel-Cadmium cells
`PbAcid-1`	36V module made of Lead-Acid cells, variant 1
`PbAcid-2`	36V module made of Lead-Acid cells, variant 2
`PbAcid-3`	36V module made of Lead-Acid cells, variant 3

Table 1 shows different technologies as well as variations on the same technology. To use the battery pack definition function, you specify a battery type and a quantity of modules to include. For example:

```
edl_system = define_batt_pack(edl_system,'LiFePO4',10)
```

In most cases, a single module lacks enough energy capacity to get the rover very far. Thus, you need to specify multiple modules. These are modeled as battery modules in parallel, which means they run at the same voltage, but can last a lot longer.

Under no circumstances should you modify this function.

3.6 Chassis Options

You are given three different types of materials for the chassis. They can be selected using the following function:

Function	Inputs	Returns
`define_chassis`	edl_system: dict type: String identifying chassis type	edl_system

No extra data files are required to run this function (everything is defined in the code directly). The valid choices for the chassis type string are as follows:

Table 2: Chassis types.

Identifier String	Description
`steel`	Steel with specific strength of 100
`magnesium`	Magnesium with specific strength of 250
`carbon`	Carbon Fiber composite with specific strength of 1000

To use the chassis definition function you use:

```
edl_system = define_chassis(edl_system, ‘steel’)
```

Under no circumstances should you modify this function.

4 SYSTEM DESIGN

The degrees of freedom you have are given by the following design variables:

Design Variables	
Parachute diameter	continuous
Fuel Mass	continuous
Wheel Radius	continuous
Gear Diameter	continuous
Chassis Mass	continuous
Motor Type	discrete
Battery Type and number of modules	discrete
Chassis Material	discrete

The design is subject to the following constraints:

Constraints	
Rover velocity at landing less than (in magnitude)	1 m/s
Distance Traveled on Mars	1000 m
Minimum Strength	40000
Maximum Cost	7.2 E6 $

The degrees of freedom have the following bounds:

Bounds	
Parachute Diameter	14-19 m
Fuel Mass	100-290 kg
Wheel Radius	0.2- 0.7 m
Gear Diameter	0.05-0.12 m
Chassis Mass	250-800 kg

In your report, values for the quantities below should be included.

Optimized parachute diameter	= xxx [m]
Optimized fuel mass	= xxx [kg]
Time to complete EDL mission	= xxx [s]
Rover Velocity at Landing	= xxx [m/s]
Optimized wheel radius	= xxx [m]
Optimized d2	= xxx [m]
Optimized Chassis Mass	= xxx [kg]
Time to Complete Rover Mission	= xxx [s]
Time to Complete Mission	= xxx [s]
Average Velocity	= xxx [m/s]
Distance Traveled	= xxx [m]
Battery Energy per meter	= xxx [J/m]
Total Cost:	= xxx [$]

You must also document the chosen motor, battery, number of battery modules, and chassis material for the rover.

The script, **`opt_edl_system.py`** is set up to work with an objective function, **`obj_fun_time`**, and a constraint function, **`constraints_edl_system`**. This script also can save the 'pickle' data file for you prior to submission. These codes are a good starting point for your design tasks. You should take some time to carefully look through **`opt_edl_system.py`**. This script is setup to run four constrained nonlinear optimization algorithms. These are 'trust-constr', 'SLSQP', 'COBYLA', and 'differential_evolution'. All are state-of-the-art and have their pros and cons. This is a challenging design problem and you will likely want to try all of the optimizer and potentially create loops around them for multi-start (using different initial conditions). These functions assume you will optimize the time to complete a mission and be constrained by cost. However, you are free to choose to optimize for cost subject to a time constraint.

5 SUBMISSIONS PROCEDURE

5.1 Team Submission

Submit all materials on Canvas in a zip file. The zip file should be named P4_MEEN357_SECYY_FA2025_ TEAMXX.zip where YY is your section number and XX is your team number. This should include: your report (Section 5.2.1), your design description (Section 5.2.2), and any supporting code you believe is relevant.

5.1.1 Report – **`FA25_SecYY_TeamXX.pdf`**

For this phase of the project, your team must produce a professional-quality report. Please be complete, but also be concise (I am not grading based upon length of report). The required report contents are:

- A cover page with:
  - Team number, names of team members, team name (if there is one), class section.
  - Aggie Honor Statement with the Following Statement:
    - None of the members of the team communicated in any way with other teams with regards to specific coding issues with this project.
- Main body sections of report:
  - Introduction to the edl + rover design problem.
  - Design results
    - Your formulation of the design problem, based on the statement given in Section 1.1, where cost is included.
    - An explanation of your strategy for solving the problem (what mix of methods did you use, etc.)
    - A summary of your chosen design solution (full specification of the rover)
    - Evidence to support your position that this is a good design for this problem (logical argumentation, graphs, tables, etc.)
- Appendix with any figures or code you believe is relevant.

5.1.2 Design description – **`FA25_SecYY_TeamXX.pickle`**

You must submit a ".pickle" file corresponding to an 'optimized' edl_system (this may be different than your racing rover).

5.1.3 Contributions – FA25_SecYY_TeamXX_collaboration.pdf

You must submit a pdf containing your team member contributions and explanation of any collaborations with other teams.