Introduction

Howdy!
My name is Ian Wilhite and I am a Senior in Robotics and Controls Engineering at Texas A&M! This portfolio contains a collection of reports and presentations from my engineering courses, project teams, programs, and competitions. The pdf content can be found on my github, and the start of each section leads with an overview of context, content, skills, and relevance. I would highly reccomend using the hyperlinks or ctrl-f to navigate this document. Thank you for taking the time to review my background, and I hope you get to meet a piece of me through my work!

Contents

Back to Contents

1. Autonomous Aerospace Systems

Summary

Autonomous Aerospace Systems (AERO 489) is a special topics course taught by Dr. John Valasek and Dr. Daniel Selva covering state space representation, uninformed and informed search, deterministic and stochastic world modeling, Reinforcement Learning (RL) models and training methods, and cumulated in a final project in which we modeled a dynamic game and trained an RL agent to play it.

- HW1: Uninformed and Informed Search in Deterministic and Non-Deterministic Environments
- HW2: First Order Logic (FOL) and Problem Structure
- HW3: Policy Comparison on Non-deterministic problems
- HW4: Q Learning - Foundations of Policy-Based RL
- HW5: Hyperparameter Tuning and Model Comparisons
- Final Project Report (Complete)

Contributors: Ian Wilhite (Solo Course Project)

Key Skills: Autonomous Systems, Aerospace Engineering, Control Theory, Reinforcement Learning, Gymnasium, Stable Baselines, Python.

Relevance: The course highlighted the practical side of RL and ML implementation on real world systems based on system modeling, fundamental understandings, model selection, and implementation structure. The homeworks and projects provided an opporuntity to watch policies be developed.

Ian Wilhite

Aero 489 – Autonomous Systems

2/5/2025

Preliminary (10 points)

1) As a search problem, the state must contain the rovers location, the action must include rover traversal in all directions (and account for edge collision), and the search problem must also track the path cost in addition to the state. The world must contain cells with danger and a goal (science).
2) Because the environment is known, static, and fully observable, our rover begins the problem knowing all necessary information to search, the information it knows cannot change, and the information it knows is complete. Because it has all information it needs, and that information cannot change, it is not necessary to sense.

Basic Search (20 points)

1) Not all algorithms find the optimal solution: Depth-First Search (DFS) does not *always* find the optimal solution, because DFS will pick one potential path from its options and follow it to completion, and will return the first feasible solution it finds, which may or may not be the optimal solution, while other algorithms prioritize less costly solutions. BFS is less efficient than UCS because it must search all parallel paths but significantly outperforms DFS because it is more likely to find a *more* efficient solution in sweeping its options laterally. A* is the most efficient solution because it is the only informed search of the group and can rely on the Manhattan heuristic to approximate the goodness of a cell towards the optimal solution. For random problems, not all methods find the "correct"/optimal solution.
    a. Optimal Solution:

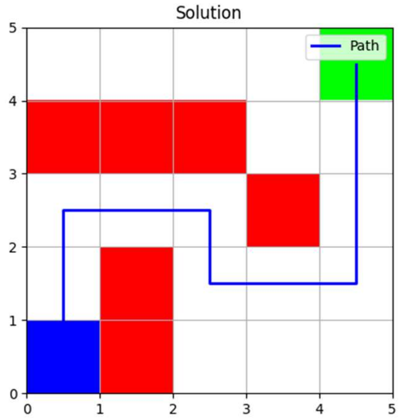


Fig. 1, sample problem with provided solution

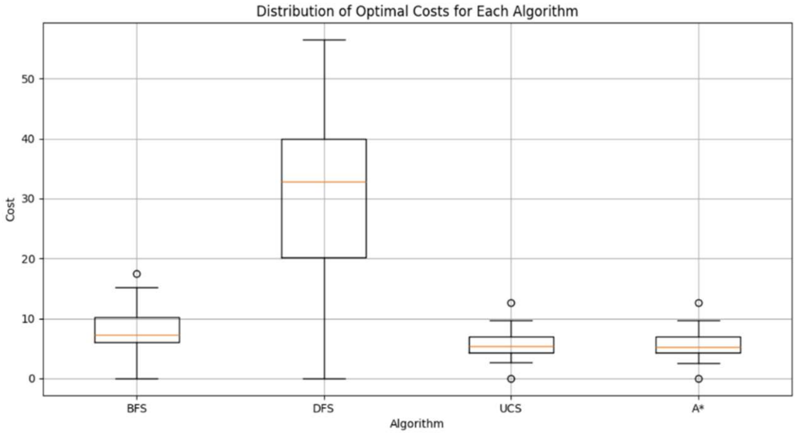


Fig. 2, Costs found by each algorithm

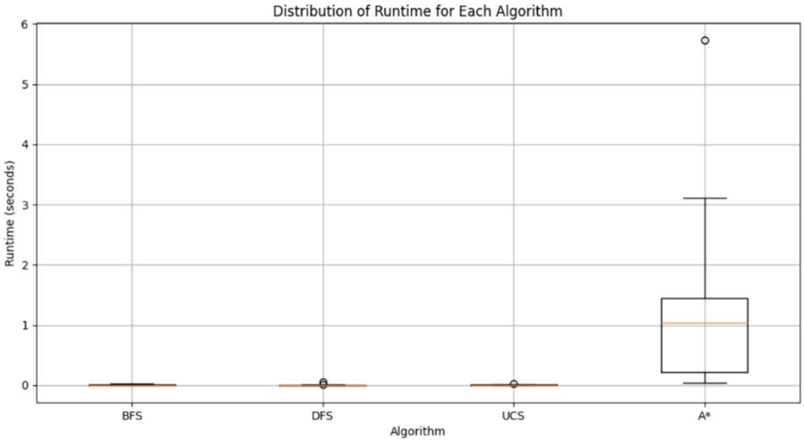
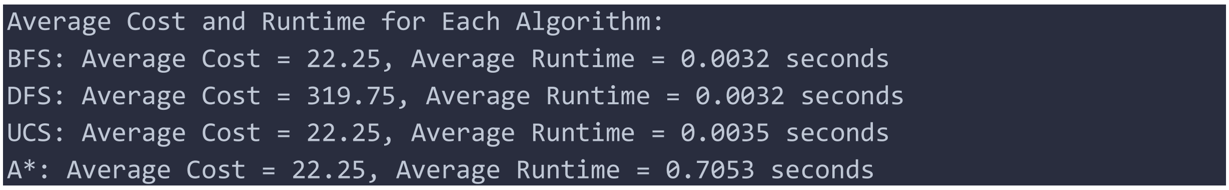


Fig. 3, Runtimes taken by each algorithm

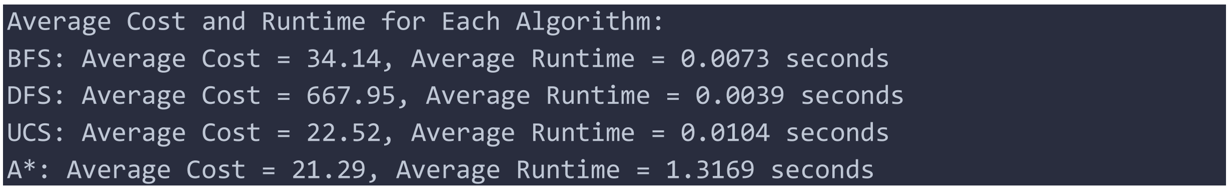
2) In cost performance, the DFS did not find the optimal solution due to its tendency to burrow into an incorrect solution and return a local, suboptimal, solution. A* had a much larger runtime to find the solution. This means that the optimal search to use for this situation would be UCS.



```
Average Cost and Runtime for Each Algorithm:
BFS: Average Cost = 22.25, Average Runtime = 0.0032 seconds
DFS: Average Cost = 319.75, Average Runtime = 0.0032 seconds
UCS: Average Cost = 22.25, Average Runtime = 0.0035 seconds
A*: Average Cost = 22.25, Average Runtime = 0.7053 seconds
```

Fig. 4, Code output of search algorithms without Danger

3) By allowing the search to enter dangerous cells with a penalty, many of the less efficient algorithms average goes up by the increase of possible-but-less-optimal solutions, while A* can take the most advantage of the new opportunity and is the only algorithm to decrease its average cost. All algorithms required more runtime (about double) to search the larger pool of possible solutions.



```
Average Cost and Runtime for Each Algorithm:
BFS: Average Cost = 34.14, Average Runtime = 0.0073 seconds
DFS: Average Cost = 667.95, Average Runtime = 0.0039 seconds
UCS: Average Cost = 22.52, Average Runtime = 0.0104 seconds
A*: Average Cost = 21.29, Average Runtime = 1.3169 seconds
```

Fig. 5, Code output of search algorithms with Danger

Part 2:

Non-Deterministic World (30 points)

1) Changes:
    a. Creating transition_with_failure function that only moves the rover 75% of the time, and leaves the rover in its current state 25% of the time.
    b. Creating a transition_with_danger_and_failure function that is identical to the transition_with_danger except it calls the transition_with_failure for the base transition function then adds danger to the cost.
2) I believe that the solution will look similar to the optimal solution found by A*, only with nested conditions such that each step will fall into a conditional while loop where it will perform each action sequentially until the path is complete.

3) Yes, the algorithm completes the search, but no, it does not come close to other deterministic methods in terms of performance. It is not reasonable to compare

these, however, because the and-or search cannot assume that each function will work, it is extending additional effort to branch out along each potential option, and is also searching the cases in which the rover does not move, however this causes additional computation as it is simply in the same situation it previously was. It is worth noting that the And-Or search implemented returned solutions similar to those generated with a traditional Depth-First-Search.

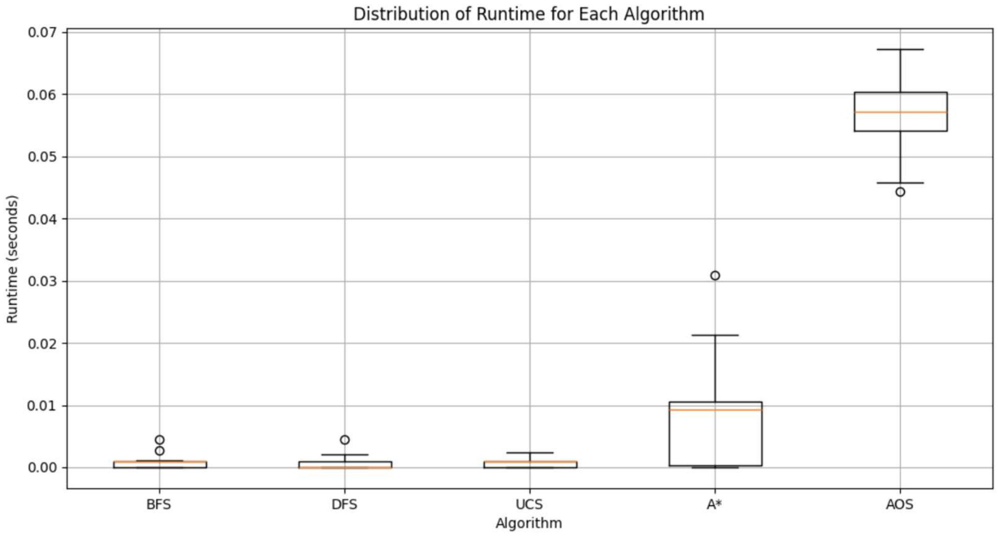


Fig. 6, Runtimes with Danger including AOS

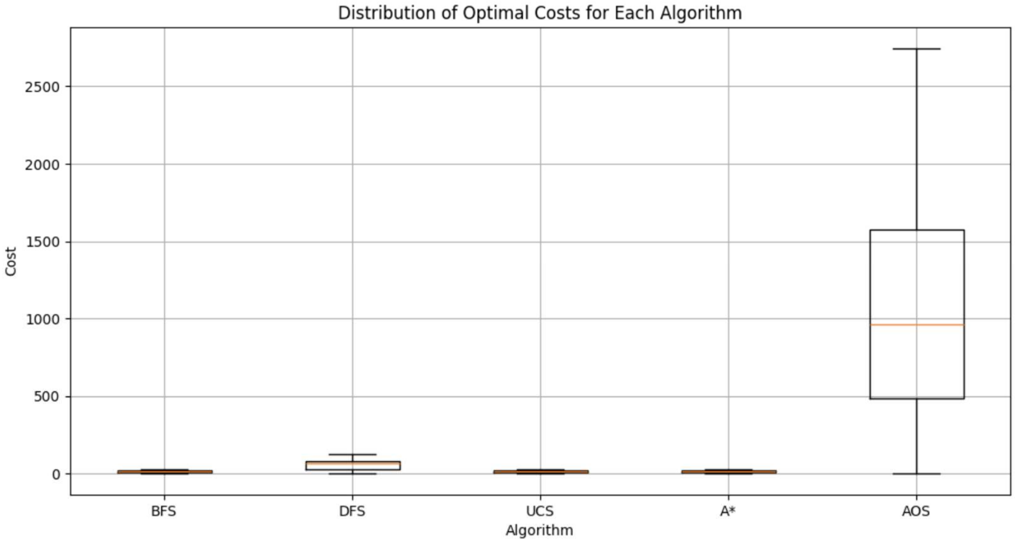


Fig. 7, Costs with Danger including AOS

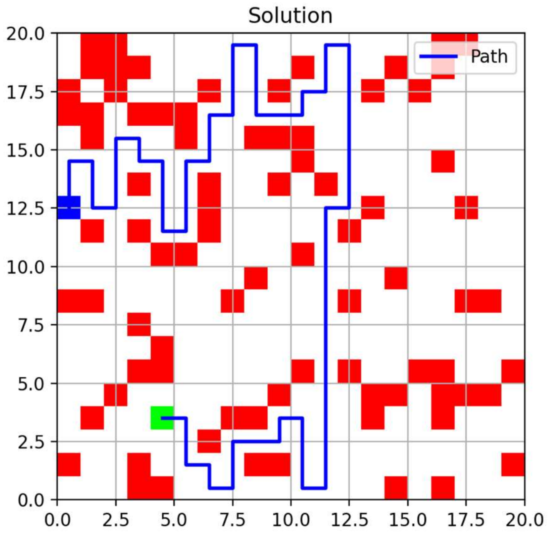


Fig. 8, 1st Sample AOS search solution

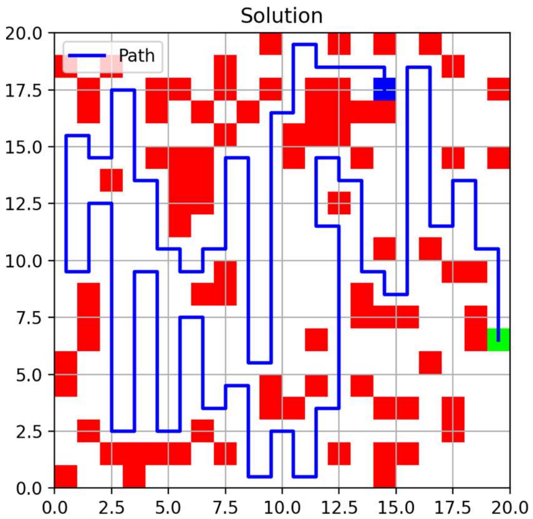
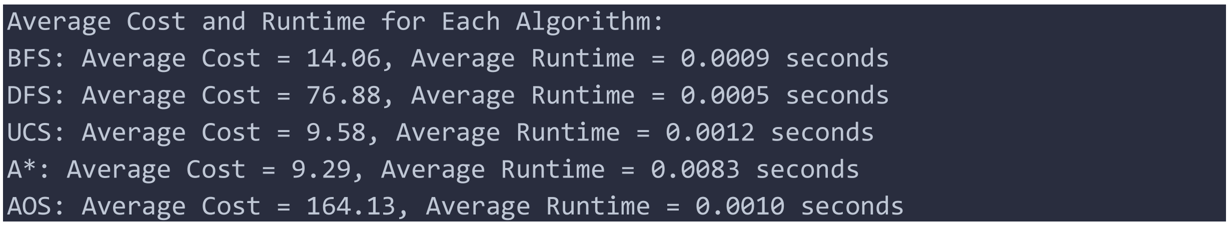


Fig. 9, 2nd Sample AOS search solution

4) Plots provided along with a sample of the path generated from the function.



```
Average Cost and Runtime for Each Algorithm:
BFS: Average Cost = 14.06, Average Runtime = 0.0009 seconds
DFS: Average Cost = 76.88, Average Runtime = 0.0005 seconds
UCS: Average Cost = 9.58, Average Runtime = 0.0012 seconds
A*: Average Cost = 9.29, Average Runtime = 0.0083 seconds
AOS: Average Cost = 164.13, Average Runtime = 0.0010 seconds
```

Fig. 10, Code output with Danger including AOS

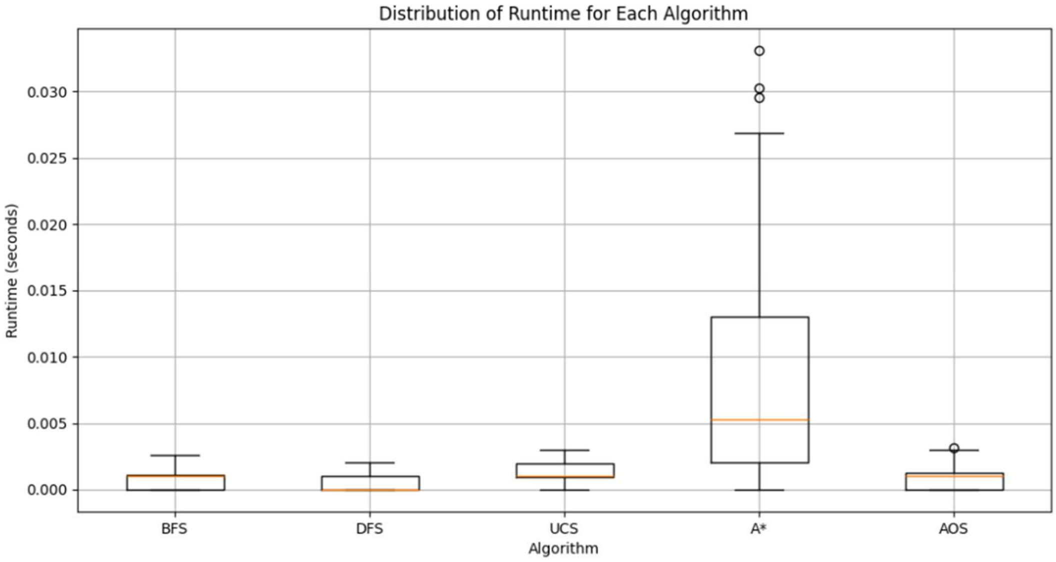


Fig. 11, Runtimes with Danger including AOS

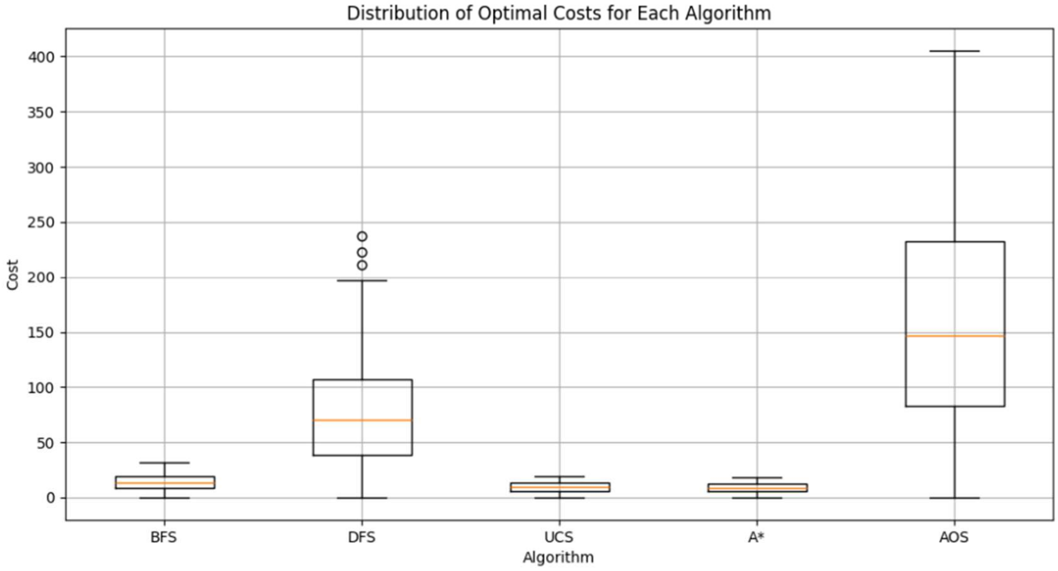


Fig. 12, Costs with Danger including AOS

Unknown World (10 points + 10 bonus)

1) To search the world without knowing the location of the alpha cell, a new heuristic would need to be developed for A*, involving the closeness using the spectrometer, given that the conditions for the search heuristic can be met. The noise must not break the admissibility of the heuristic, which a sensor would likely not so long as the majority of the noise comes from offset and proportional error, and that very little comes from randomness error.

AERO 489 – Selva
Ian Wilhite

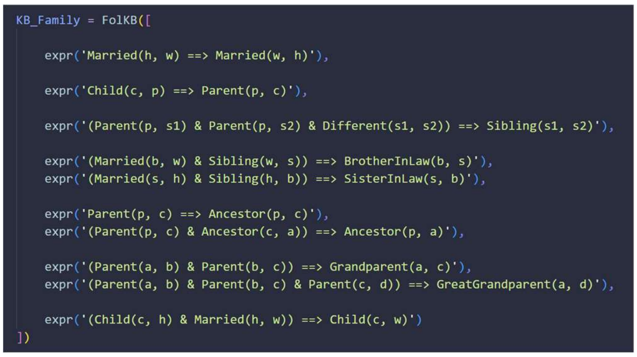
2/24/2025

Short questions (10pts)

1) Syntax vs semantic
   a. Syntax is the structure of the notation for the problem. Semantics are the intended meaning of the problem.
   b. The difference between the two can arise when the syntax of the problem can "misunderstand" the intended result, like in the "everybody loves somebody" example in class that results in one "somebody" that everybody loves.
2) Predicate vs Function
   a. A predicate is a Boolean logical statement that evaluates to true or false. A function maps an input to an output and can return different data types. A predicate can be viewed as a type of function that only returns a Boolean.
   b. The IsEven(int num) returns [True, False] and would be considered a **predicate**, because the argument is a boolean.
   c. The SquareNum(int num) returns [int squaredNum], and would be considered a **function** because the return argument is not a boolean.
3) Action schema
   a. The action schema is the notation that describes the outcome of when a given action is applied to an object in a state.
4) Ground atomic sentence
   a. A ground atomic sentence is one that explains a particular relationship or fundamental existence of the world. Ie: Married(John, Jane)
5) Database vs standard FOL semantics
   a. Database semantics assume that information not explicitly given is false.
   b. FOL semantics will not assume anything about fact, and will claim that it could be either true or false.

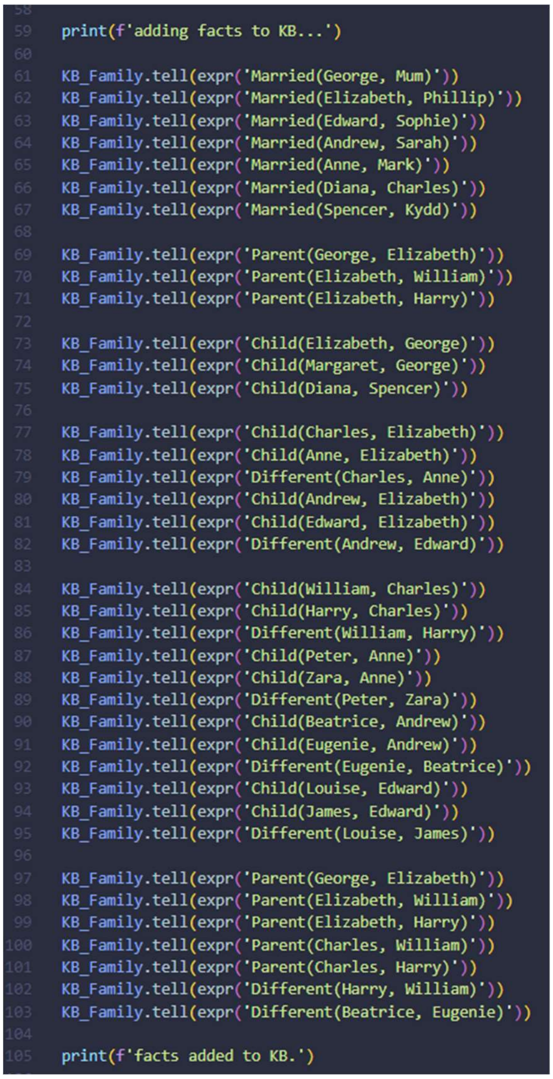
Problem 1: FOL

a) Tell rules



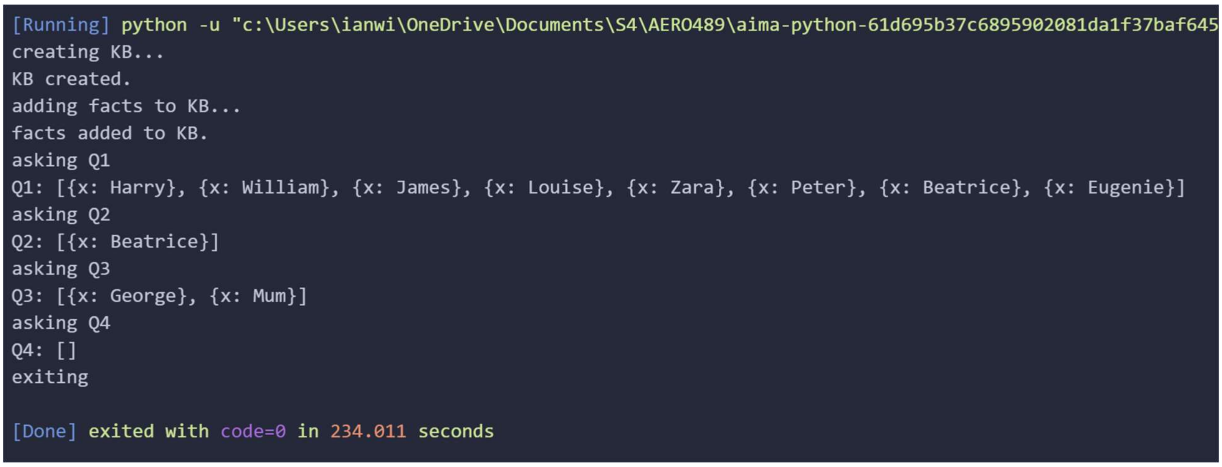
```
KB_Family = FolKB([

    expr('Married(h, w) ==> Married(w, h)'),

    expr('Child(c, p) ==> Parent(p, c)'),

    expr('(Parent(p, s1) & Parent(p, s2) & Different(s1, s2)) ==> Sibling(s1, s2)'),

    expr('(Married(b, w) & Sibling(w, s)) ==> BrotherInLaw(b, s)'),
    expr('(Married(s, h) & Sibling(h, b)) ==> SisterInLaw(s, b)'),

    expr('Parent(p, c) ==> Ancestor(p, c)'),
    expr('(Parent(p, c) & Ancestor(c, a)) ==> Ancestor(p, a)'),

    expr('(Parent(a, b) & Parent(b, c)) ==> Grandparent(a, c)'),
    expr('(Parent(a, b) & Parent(b, c) & Parent(c, d)) ==> GreatGrandparent(a, d)'),

    expr('(Child(c, h) & Married(h, w)) ==> Child(c, w)')
])
```

b) Tell facts

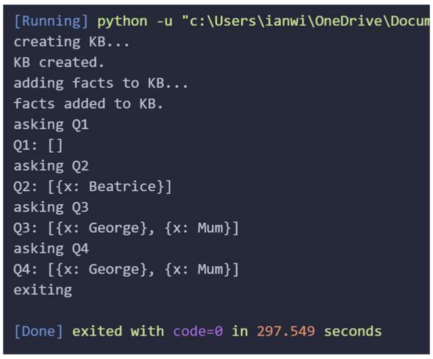


```
print(f'adding facts to KB...')

KB_Family.tell(expr('Married(George, Mum)'))
KB_Family.tell(expr('Married(Elizabeth, Phillip)'))
KB_Family.tell(expr('Married(Edward, Sophie)'))
KB_Family.tell(expr('Married(Andrew, Sarah)'))
KB_Family.tell(expr('Married(Anne, Mark)'))
KB_Family.tell(expr('Married(Diana, Charles)'))
KB_Family.tell(expr('Married(Spencer, Kydd)'))

KB_Family.tell(expr('Parent(George, Elizabeth)'))
KB_Family.tell(expr('Parent(Elizabeth, William)'))
KB_Family.tell(expr('Parent(Elizabeth, Harry)'))

KB_Family.tell(expr('Child(Elizabeth, George)'))
KB_Family.tell(expr('Child(Margaret, George)'))
KB_Family.tell(expr('Child(Diana, Spencer)'))

KB_Family.tell(expr('Child(Charles, Elizabeth)'))
KB_Family.tell(expr('Child(Anne, Elizabeth)'))
KB_Family.tell(expr('Different(Charles, Anne)'))
KB_Family.tell(expr('Child(Andrew, Elizabeth)'))
KB_Family.tell(expr('Child(Edward, Elizabeth)'))
KB_Family.tell(expr('Different(Andrew, Edward)'))

KB_Family.tell(expr('Child(William, Charles)'))
KB_Family.tell(expr('Child(Harry, Charles)'))
KB_Family.tell(expr('Different(William, Harry)'))
KB_Family.tell(expr('Child(Peter, Anne)'))
KB_Family.tell(expr('Child(Zara, Anne)'))
KB_Family.tell(expr('Different(Peter, Zara)'))
KB_Family.tell(expr('Child(Beatrice, Andrew)'))
KB_Family.tell(expr('Child(Eugenie, Andrew)'))
KB_Family.tell(expr('Different(Eugenie, Beatrice)'))
KB_Family.tell(expr('Child(Louise, Edward)'))
KB_Family.tell(expr('Child(James, Edward)'))
KB_Family.tell(expr('Different(Louise, James)'))

KB_Family.tell(expr('Parent(George, Elizabeth)'))
KB_Family.tell(expr('Parent(Elizabeth, William)'))
KB_Family.tell(expr('Parent(Elizabeth, Harry)'))
KB_Family.tell(expr('Parent(Charles, William)'))
KB_Family.tell(expr('Parent(Charles, Harry)'))
KB_Family.tell(expr('Different(Harry, William)'))
KB_Family.tell(expr('Different(Beatrice, Eugenie)'))

print(f'facts added to KB.')
```

c) Ask
   a. Elizabeth’s grandchildren
   b. Eugenie’s siblings
   c. Zara’s great-grandparents
   d. Eugenie’s ancestors



```
[Running] python -u "c:\Users\ianwi\OneDrive\Documents\S4\AERO489\aima-python-61d695b37c6895902081da1f37baf645
creating KB...
KB created.
adding facts to KB...
facts added to KB.
asking Q1
Q1: [{x: Harry}, {x: William}, {x: James}, {x: Louise}, {x: Zara}, {x: Peter}, {x: Beatrice}, {x: Eugenie}]
asking Q2
Q2: [{x: Beatrice}]
asking Q3
Q3: [{x: George}, {x: Mum}]
asking Q4
Q4: []
exiting

[Done] exited with code=0 in 234.011 seconds
```



```
[Running] python -u "c:\Users\ianwi\OneDrive\Docum
creating KB...
KB created.
adding facts to KB...
facts added to KB.
asking Q1
Q1: []
asking Q2
Q2: [{x: Beatrice}]
asking Q3
Q3: [{x: George}, {x: Mum}]
asking Q4
Q4: [{x: George}, {x: Mum}]
exiting

[Done] exited with code=0 in 297.549 seconds
```

Problem 2: Europa Rover

1) Domain:
   a. Requirements
      i. STRIPS is the most common basic notation that the PDDL system will use FOL to model the system.
   b. Predicates (variables to track)
      i. Rover location – the location of the rover on Europa (note: we do not need to track location alpha because it is static and does not move)
      ii. HasBiosample – whether or not the rover has obtained the sample
      iii. HasComponent – whether or not the rover has the component
      iv. BatteryCharged – whether or not the battery is charged

v. Drilled (based on location) – whether or not the current location has been drilled

vi. SoftwareUpdated – whether or not the rover software has been updated

vii. PreheatedDrill – whether or not the drilled has been preheated

viii. AnalysisCompleted – whether or not the analysis is complete

ix. ContactAvailible – whether or not transmission contact is available

x. DataTransmitted – whether or not the data has been transmitted (and the mission has been completed)

c. Actions

i. Charge

ii. Move

iii. Update software

iv. Retrieve components

v. Preheat drill

vi. Drill

vii. Analyze sample

viii. Transmit data

2) See attached files.

Ian Wilhite

AERO 489: Selva, Valasek

3/4/2025

Question 1: Formulate this problem as a Markov Decision Process. Specifically, provide the states, actions, rewards, and transition model.

State:

- The location of the rover within the world (x,y)
- The danger of the cells, as stored within the grid.

Actions:

- Move Up, (x,y) -> (x, y + 1)
- Move Right, (x,y) -> (x + 1, y)
- Move Down, (x,y) -> (x, y - 1)
- Move Left, (x,y) -> (x - 1, y)

Rewards:

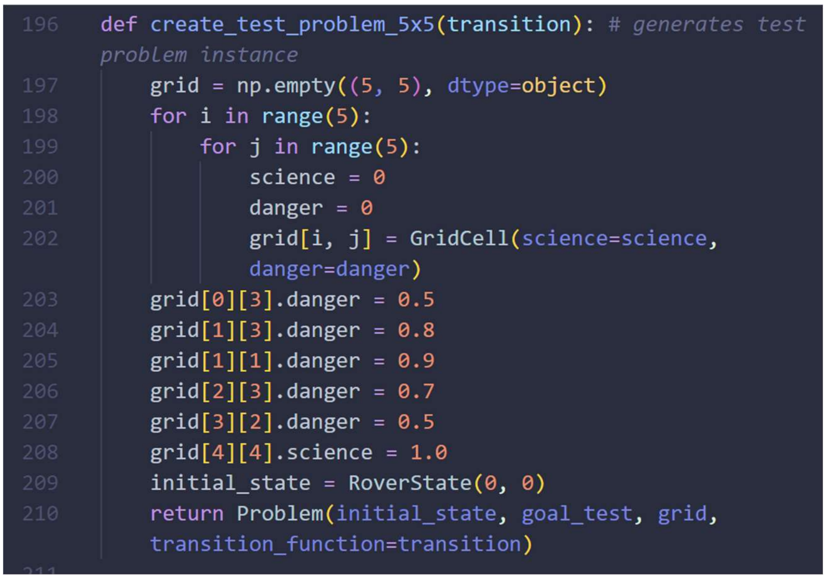
- Move cost: -0.05 (included on unsuccessful moves)
- Goal reward: +1.00
- Crater cost: -1.00

Transition Model (for all actions):

- Move successfully -> 80%
- Stay in current cell -> 10%
- Slide towards center of grid -> 10%

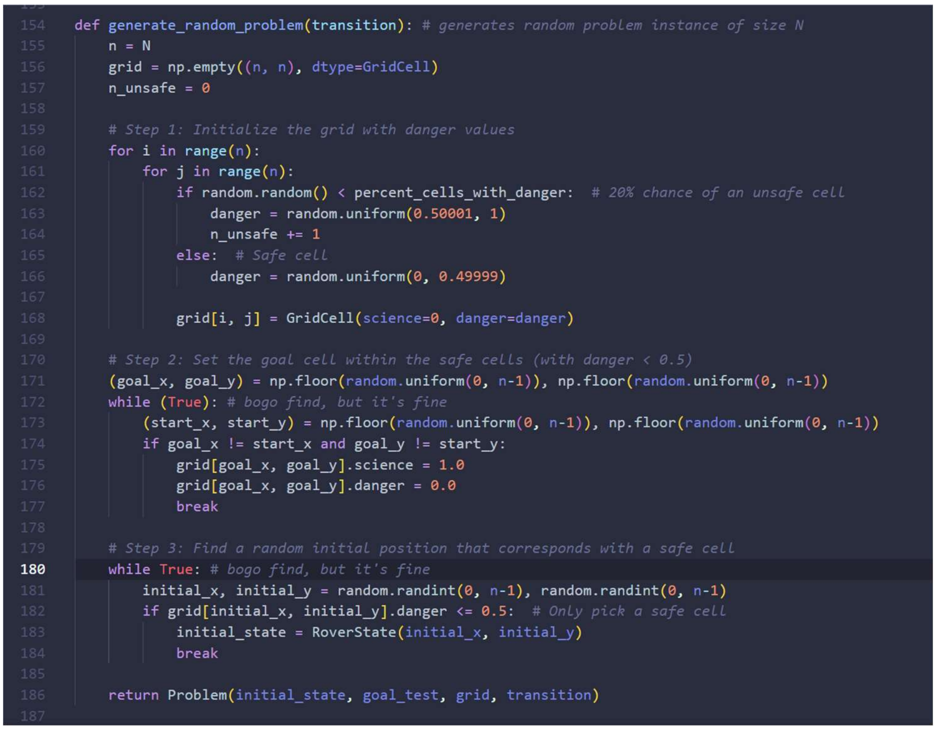
Question 2: Write code in Python that implements the environment. This should include:

- A function to create a fixed test world (e.g., start cell in (1,1), goal in (5,5), danger cells in (1,4), (2,4), (2,2), (3,4), crater in (4,3)).



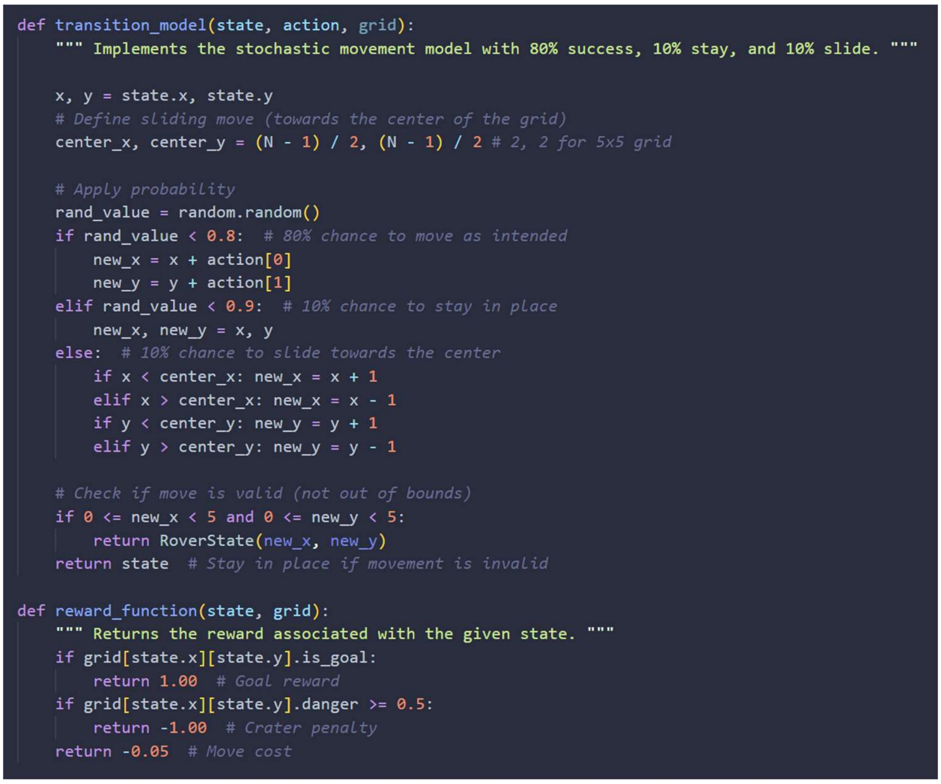
```python
def create_test_problem_5x5(transition): # generates test problem instance
    grid = np.empty((5, 5), dtype=object)
    for i in range(5):
        for j in range(5):
            science = 0
            danger = 0
            grid[i, j] = GridCell(science=science, danger=danger)
    grid[0][3].danger = 0.5
    grid[1][3].danger = 0.8
    grid[1][1].danger = 0.9
    grid[2][3].danger = 0.7
    grid[3][2].danger = 0.5
    grid[4][4].science = 1.0
    initial_state = RoverState(0, 0)
    return Problem(initial_state, goal_test, grid, transition_function=transition)
```

- A function to create a random world of this type



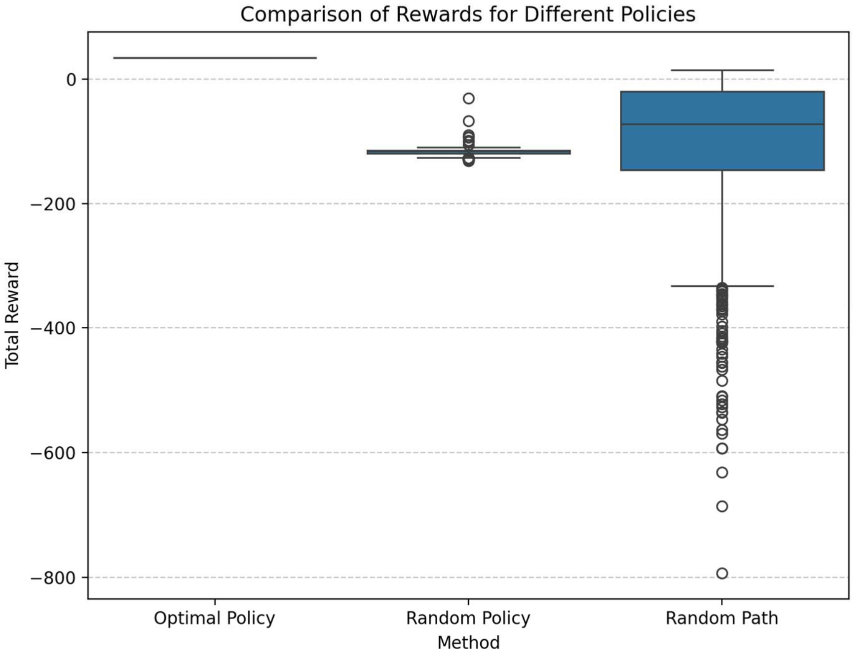
```python
def generate_random_problem(transition): # generates random problem instance of size N
    n = N
    grid = np.empty((n, n), dtype=GridCell)
    n_unsafe = 0

    # Step 1: Initialize the grid with danger values
    for i in range(n):
        for j in range(n):
            if random.random() < percent_cells_with_danger:  # 20% chance of an unsafe cell
                danger = random.uniform(0.50001, 1)
                n_unsafe += 1
            else:  # Safe cell
                danger = random.uniform(0, 0.49999)

            grid[i, j] = GridCell(science=0, danger=danger)

    # Step 2: Set the goal cell within the safe cells (with danger < 0.5)
    (goal_x, goal_y) = np.floor(random.uniform(0, n-1)), np.floor(random.uniform(0, n-1))
    while (True): # bogo find, but it's fine
        (start_x, start_y) = np.floor(random.uniform(0, n-1)), np.floor(random.uniform(0, n-1))
        if goal_x != start_x and goal_y != start_y:
            grid[goal_x, goal_y].science = 1.0
            grid[goal_x, goal_y].danger = 0.0
            break

    # Step 3: Find a random initial position that corresponds with a safe cell
    while True: # bogo find, but it's fine
        initial_x, initial_y = random.randint(0, n-1), random.randint(0, n-1)
        if grid[initial_x, initial_y].danger <= 0.5:  # Only pick a safe cell
            initial_state = RoverState(initial_x, initial_y)
            break

    return Problem(initial_state, goal_test, grid, transition)
```

- Functions implementing the transition model and the reward function. Given a current state and action, they should return the next state and the reward respectively (or you can do this in the same function).



```
def transition_model(state, action, grid):
    """ Implements the stochastic movement model with 80% success, 10% stay, and 10% slide. """

    x, y = state.x, state.y
    # Define sliding move (towards the center of the grid)
    center_x, center_y = (N - 1) / 2, (N - 1) / 2 # 2, 2 for 5x5 grid

    # Apply probability
    rand_value = random.random()
    if rand_value < 0.8:  # 80% chance to move as intended
        new_x = x + action[0]
        new_y = y + action[1]
    elif rand_value < 0.9:  # 10% chance to stay in place
        new_x, new_y = x, y
    else:  # 10% chance to slide towards the center
        if x < center_x: new_x = x + 1
        elif x > center_x: new_x = x - 1
        if y < center_y: new_y = y + 1
        elif y > center_y: new_y = y - 1

    # Check if move is valid (not out of bounds)
    if 0 <= new_x < 5 and 0 <= new_y < 5:
        return RoverState(new_x, new_y)
    return state  # Stay in place if movement is invalid

def reward_function(state, grid):
    """ Returns the reward associated with the given state. """
    if grid[state.x][state.y].is_goal:
        return 1.00  # Goal reward
    if grid[state.x][state.y].danger >= 0.5:
        return -1.00  # Crater penalty
    return -0.05  # Move cost
```

Question 3: Write code for an agent that follows a given policy, specified as a look up table (state, action). Run an agent using a random policy (i.e., that moves randomly in this environment until reaching a terminal state) in the test problem 1000 times. Make a boxplot of the total undiscounted rewards collected by the agent.

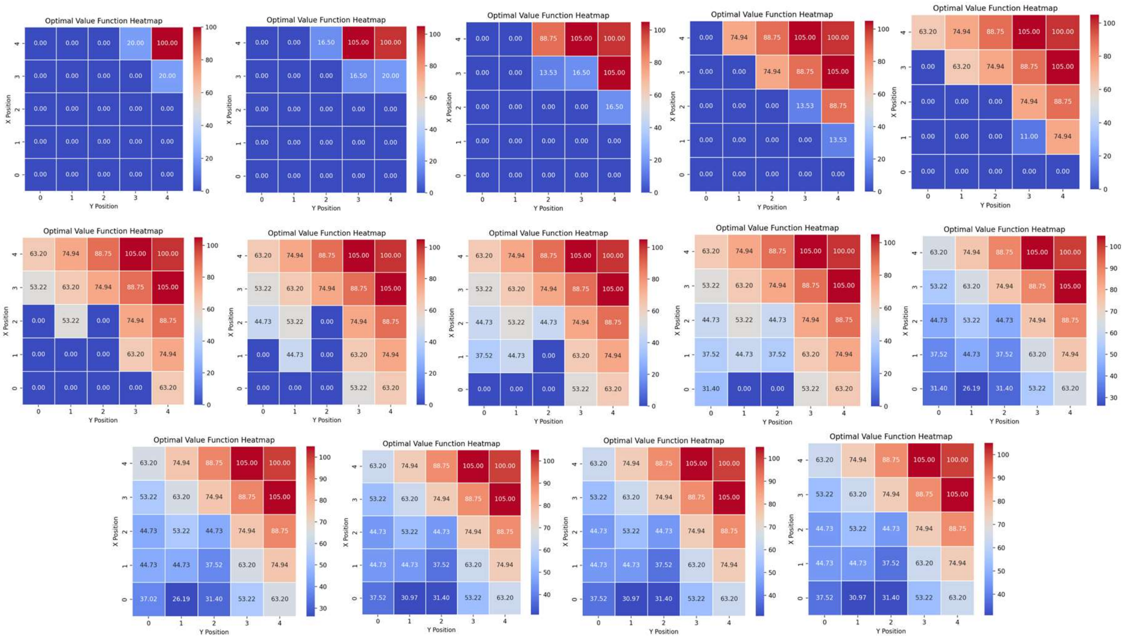


Random policy method represents that a policy was randomly generated then applied, if an infinite loop occurred then a penalty of -100 was applied and the episode terminated. The high cost in the random policy was often a result of the repeated high failure penalty. The random policy method has a cluster just below -100, because its failure is often due to the -100 failure penalty due to looping.

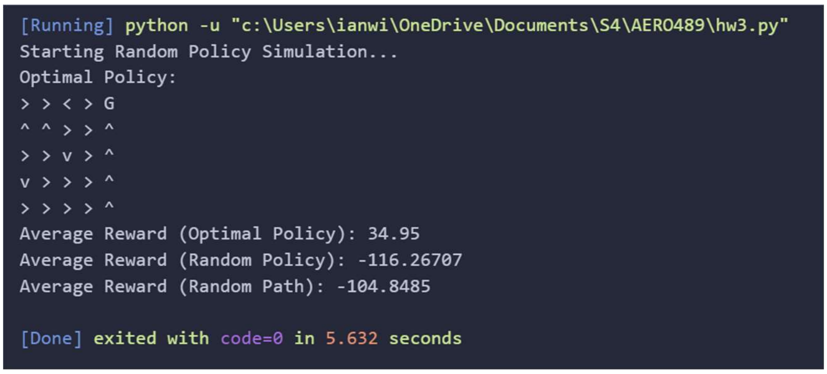
Random path indicates that a random move was applied at each possible situation, where the agent was allowed to cross back over the same location multiple times. The high cost in the random path was often due to the accumulation of the movement cost for 100+ moves to solve the problem. The random path is dispersed much more broadly as it has the chance to converge sooner, and a much lower spread as it allows for repeat cell visits.

The optimal policy agent only has a single value because there is no random element to the design process.

Question 4: Write an algorithm that implements value iteration. Compute the optimal value function and the optimal policy for the test problem using your algorithm (show the outputs visually). Evaluate an agent following the optimal policy and compare the total rewards obtained with those of the agent following a random policy



The optimal policy unsurprisingly performs much better than the other two random-based models at solving the problem and avoiding error. It is worth noting that there is an error in the policy generated in the top middle where the policy directs the agent to move left, when it should move right. This error might be caused by a discrepancy between the values being displayed and the combination of (reward + discount_factor * value[next_state]) which could be caused by the presence of a crater. This type of local maximum would result in an infinite loop in the policy as it would converge on a non-goal value.



```
[Running] python -u "c:\Users\ianwi\OneDrive\Documents\S4\AERO489\hw3.py"
Starting Random Policy Simulation...
Optimal Policy:
> > < > G
^ ^ > > ^
> > v > ^
v > > > ^
> > > > ^
Average Reward (Optimal Policy): 34.95
Average Reward (Random Policy): -116.26707
Average Reward (Random Path): -104.8485

[Done] exited with code=0 in 5.632 seconds
```

AERO 489

Valasek, Selva

4/3/2025

Ian Wilhite

HW4: Q Learning:

a) The reward matrix to represent the problem contains 100 utility for any move that results in being in the goal state (including staying in the goal state). In the first episode (1), we establish the impossible moves as -1, the successful moves into the goal state with a reward state of 100, and other possible moves as having 0 utility.

$$Q_{ep1} = \begin{bmatrix} 0 & -1 & -1 & -1 & 0 & -1 \\ -1 & 0 & -1 & 0 & -1 & 100 \\ -1 & -1 & 0 & 0 & -1 & -1 \\ -1 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & -1 & 0 & 0 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \tag{1}$$

b) In the second episode (2), we iterate based on the potential utility of the options provided to the agent at any given time. We include staying in the current position as a valid option to allow the agent to save the utility of its current position, and to ease the extraction of the convergent utility of the entire system.

$$Q_{ep2} = \begin{bmatrix} 0 & -1 & -1 & -1 & 0 & -1 \\ -1 & 0 & -1 & 0 & -1 & 100 \\ -1 & -1 & 0 & 0 & -1 & -1 \\ -1 & 80 & 0 & 0 & 0 & -1 \\ 0 & -1 & -1 & 0 & 0 & 100 \\ -1 & 0 & -1 & -1 & 80 & 100 \end{bmatrix} \tag{2}$$

$$Q_{ep3} = \begin{bmatrix} 0 & -1 & -1 & -1 & 80 & -1 \\ -1 & 0 & -1 & 64 & -1 & 100 \\ -1 & -1 & 0 & 64 & -1 & -1 \\ -1 & 80 & 0 & 0 & 80 & -1 \\ 0 & -1 & -1 & 0 & 80 & 100 \\ -1 & 0 & -1 & -1 & 80 & 100 \end{bmatrix} \tag{3}$$

$$Q_{ep4} = \begin{bmatrix} 64 & -1 & -1 & -1 & 80 & -1 \\ -1 & 0 & -1 & 64 & -1 & 100 \\ -1 & -1 & 0 & 64 & -1 & -1 \\ -1 & 80 & 0 & 0 & 80 & -1 \\ 64 & -1 & -1 & 64 & 80 & 100 \\ -1 & 0 & -1 & -1 & 80 & 100 \end{bmatrix} \tag{4}$$

$$Q_{ep5} = \begin{bmatrix} 64 & -1 & -1 & -1 & 80 & -1 \\ -1 & 0 & -1 & 64 & -1 & 100 \\ -1 & -1 & 0 & 64 & -1 & -1 \\ -1 & 80 & 51.2 & 0 & 80 & -1 \\ 64 & -1 & -1 & 64 & 80 & 100 \\ -1 & 0 & -1 & -1 & 80 & 100 \end{bmatrix} \tag{5}$$

c) The final convergent matrix can be found by applying the process many times. After 1000 trials, the convergent matrix was found in (6). Some observations to note include that all columns contain either -1 or a constant. This represents that after converging, moving to that state from any other state will result in the same utility, or conversely, that the value of a given state is independent of the direction of approach. Recalling that the diagonal represents that utility of staying in a given state, combined that all columns contain the same value, the diagonal of the convergent matrix can be utilized to develop a common utility for each state.

$$Q_{conv.} = \begin{bmatrix} 64 & -1 & -1 & -1 & 80 & -1 \\ -1 & 80 & -1 & 64 & -1 & 100 \\ -1 & -1 & 51.2 & 64 & -1 & -1 \\ -1 & 80 & 51.2 & 64 & 80 & -1 \\ 64 & -1 & -1 & 64 & 80 & 100 \\ -1 & 80 & -1 & -1 & 80 & 100 \end{bmatrix} \tag{6}$$

By evaluating the diagonal, we can observe that the convergent Q matrix represents the intuition of the initial problem. The goal state is worth the full goal value of 100. States one and four that allow for one move to the goal state have value of the goal state multiplied by the discount factor, or 80. The subsequent states are each valued at their optimal move value multiplied by the discount factor, or valued at the goal state value multiplied by the discount factor to the power of the minimum number of moves to the goal state. This Q matrix would create an optimal policy to provide an appropriate optimal policy to exit the house.

AERO 489

Valasek, Selva

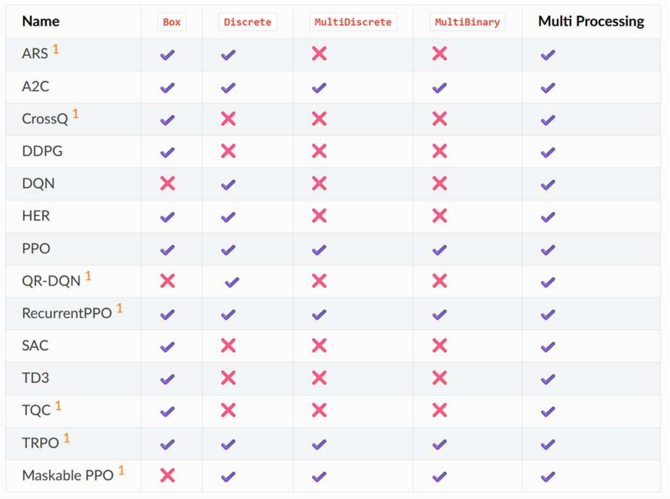
Ian Wilhite

4/13/2025

Acrobot-v1 Machine Learning Hyperparameter Study

Discussion:

1) Gymnasium Environment:
   a. Environment: Acrobot-v1
   b. It is a double pendulum driven from the joint between the arms [1]. The objective is to upright the assembly above the center pivot. This environment has a box observation space and a discrete action space with 3 options for the agent at a given timestep: apply positive torque, apply no torque, apply negative torque [1].
2) Three candidate algorithms:
   a. From the stable baselines compatibility table in Figure 1, we can identify that acrobot requires a discrete action space, meaning that Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C), and Deep Q-Network (DQN) would work with the environment [2]. These algorithms were selected based on their popular usage for a variety of applications and Dr. Valasek's common mention of A2C.



Name	Box	Discrete	MultiDiscrete	MultiBinary	Multi Processing
ARS [1]	✓	✓	✗	✗	✓
A2C	✓	✓	✓	✓	✓
CrossQ [1]	✓	✗	✗	✗	✓
DDPG	✓	✗	✗	✗	✓
DQN	✗	✓	✗	✗	✓
HER	✓	✓	✗	✗	✓
PPO	✓	✓	✓	✓	✓
QR-DQN [1]	✗	✓	✗	✗	✓
RecurrentPPO [1]	✓	✓	✓	✓	✓
SAC	✓	✗	✗	✗	✓
TD3	✓	✗	✗	✗	✓
TQC [1]	✓	✗	✗	✗	✓
TRPO [1]	✓	✓	✓	✓	✓
Maskable PPO [1]	✗	✓	✓	✓	✓

Figure 1. Stable Baselines algorithm environment compatibility

3) Hyperparameter Study:

a. The two hyperparameters I will evaluate are Learning Rate and the Discount Factor. I believe that the learning rate will affect the speed of convergence for the model, as well as the stability of the convergence achieved, whereas the discount factor will determine the performance and aggressiveness of the model in attempting to achieve high performance. An initial investigation can find a workable range for subsequent experiments. Using an average of three trials, a mean can be calculated and plotted in a heat map in Figure 2.

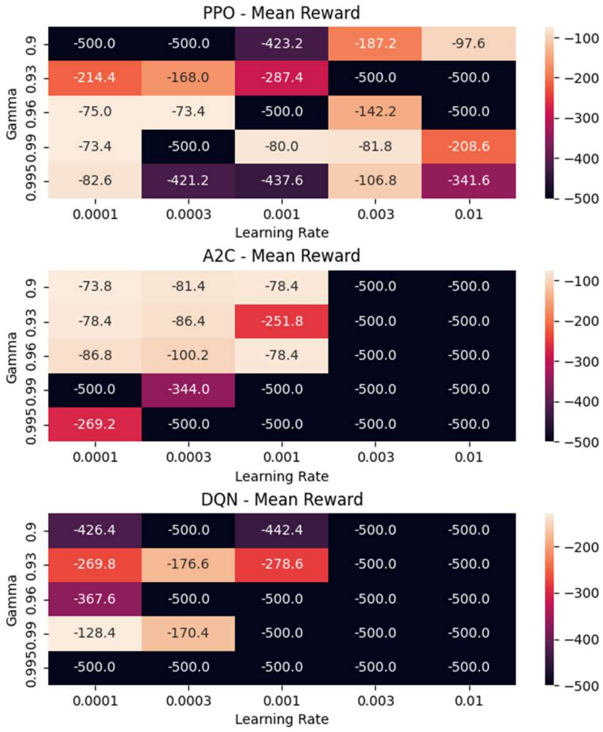
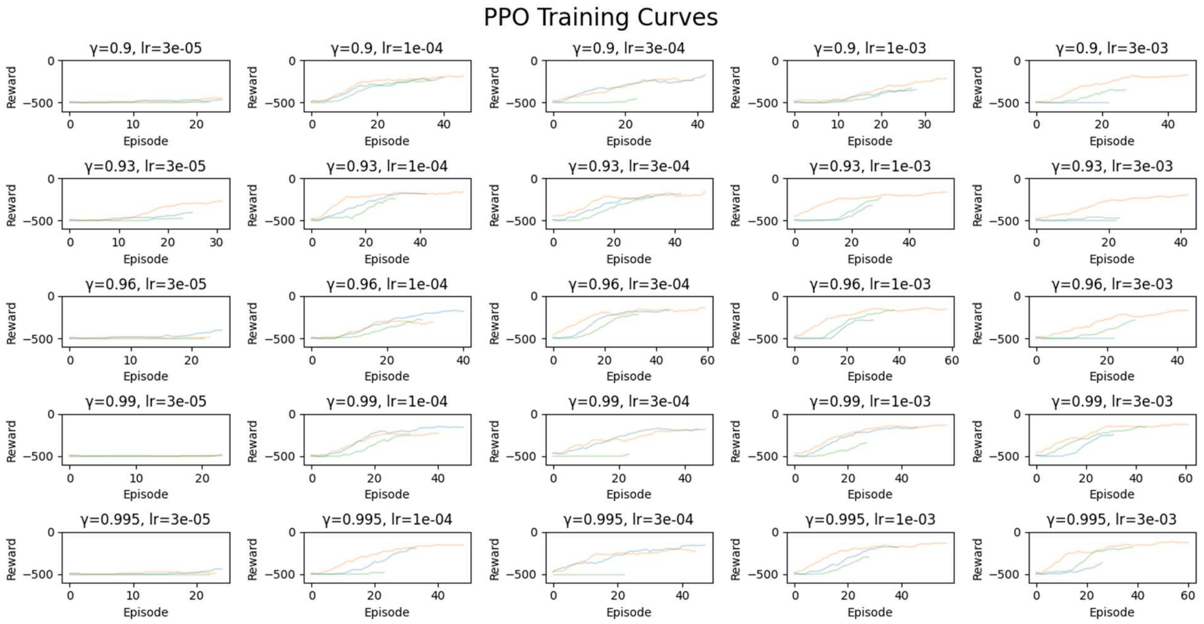


Figure 2. Hyperparameter study for 10,000 episodes

Many of the models simply did not converge, and therefore the training size needed to be increased, however the range seemed promising. It is worth noting that PPO succeeded with little trend, A2C showed a clear successful and unsuccessful region, and DQN showed little progress reinforcing the similar range to A2C.

With an increased timestep, the training curves for each model could be constructed with respect to the discount factor and learning rate. In Figure 4, it can be observed that with lower learning rates (the left columns) that the models converged slower or negligibly, whereas with higher learning rates (the right columns), the models demonstrated oscillation after convergence or suffered failure altogether. The discount factor showed minimal impact, but broadly that a lower discount factor (top rows) created impatient models, that tended to be more aggressive in improving its rewards per episode, whereas a higher discount factor (bottom rows) created a more patient model that converged slower with fewer instances of decreasing rewards between episodes.


PPO Training Curves

γ=0.9, lr=3e-05 | γ=0.9, lr=1e-04 | γ=0.9, lr=3e-04 | γ=0.9, lr=1e-03 | γ=0.9, lr=3e-03

γ=0.93, lr=3e-05 | γ=0.93, lr=1e-04 | γ=0.93, lr=3e-04 | γ=0.93, lr=1e-03 | γ=0.93, lr=3e-03

γ=0.96, lr=3e-05 | γ=0.96, lr=1e-04 | γ=0.96, lr=3e-04 | γ=0.96, lr=1e-03 | γ=0.96, lr=3e-03

γ=0.99, lr=3e-05 | γ=0.99, lr=1e-04 | γ=0.99, lr=3e-04 | γ=0.99, lr=1e-03 | γ=0.99, lr=3e-03

γ=0.995, lr=3e-05 | γ=0.995, lr=1e-04 | γ=0.995, lr=3e-04 | γ=0.995, lr=1e-03 | γ=0.995, lr=3e-03

Figure 4. PPO Learning curves for 15,000 episodes w.r.t. discount factor and learning rate

For deep Q learning, all models across the board suffered from increased volatility in convergence. Notably, with very low learning rates they ended training sooner because they converged at negligible rewards.

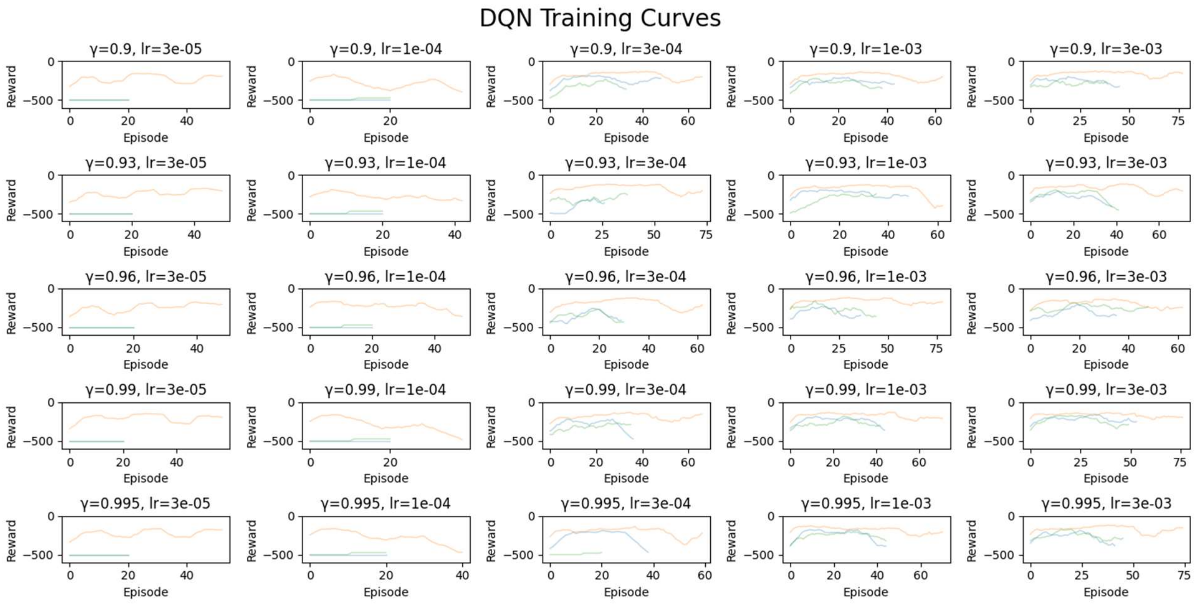


Figure 5. DQN Learning curves for 15,000 episodes w.r.t. discount factor and learning rate

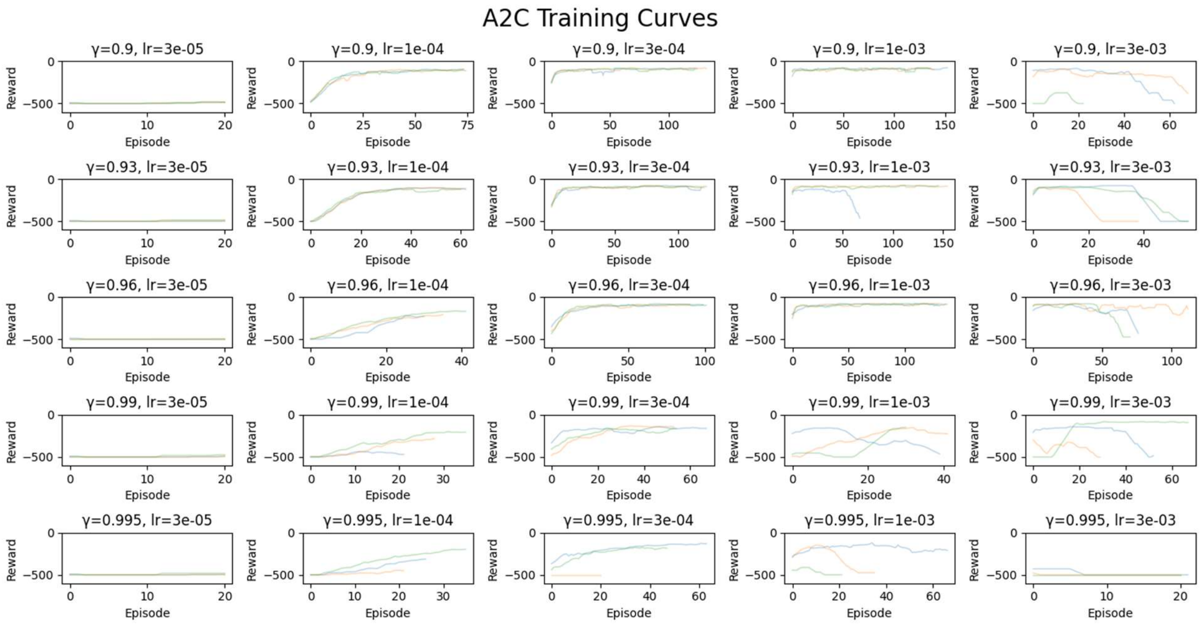


Figure 6. A2C Learning curves for 15,000 episodes w.r.t. discount factor and learning rate

A2C Learning clearly demonstrates the trends stated before, that as the discount factor decreases the model becomes impatient, particularly where the learning rate is effective to converge. When the learning rate is too high, the model does not converge, and when the learning rate is too low, it will not converge or will converge slowly.

4) Direct comparison

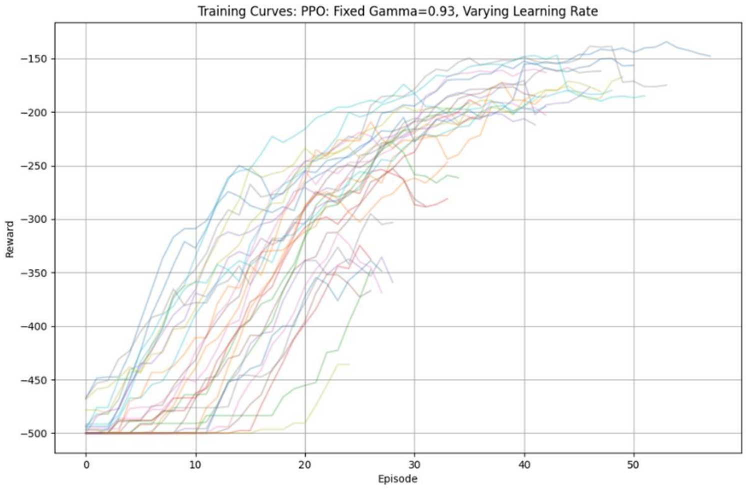


Figure 7. PPO Learning Curves while varying learning rate

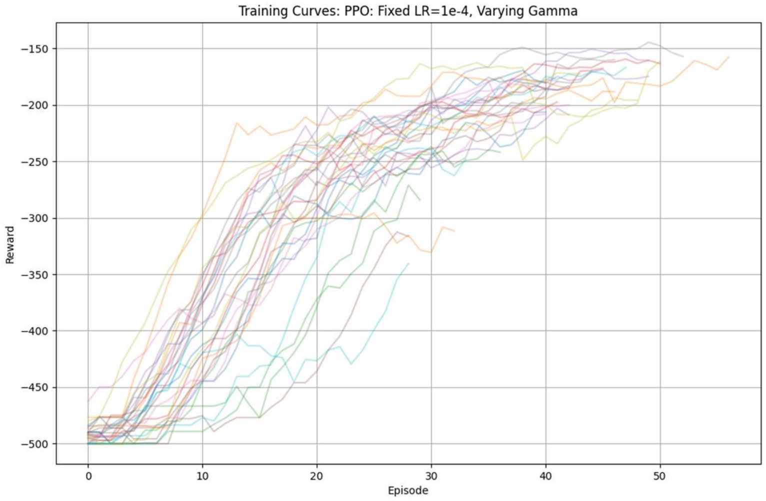


Figure 8. PPO Learning Curves while varying discount factor

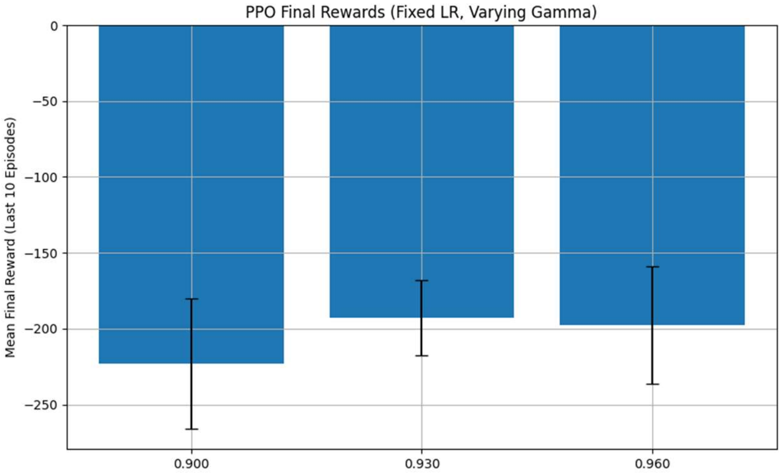


Figure 9. PPO rewards varying discount factor

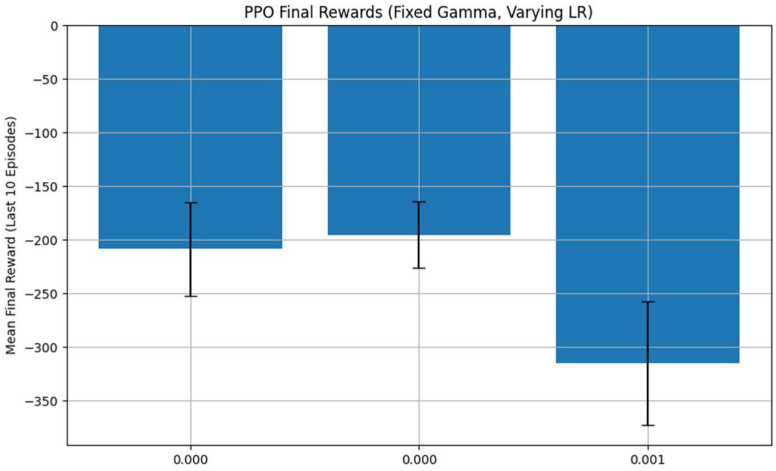


Figure 10. PPO rewards varying learning rate

PPO can be observed as converging consistently, and showing that the lower discount factors decrease the mean resulting reward. It should be noted that the ending rewards are not indicative of the peak rewards or the time to reach peak performance, only that after many iterations

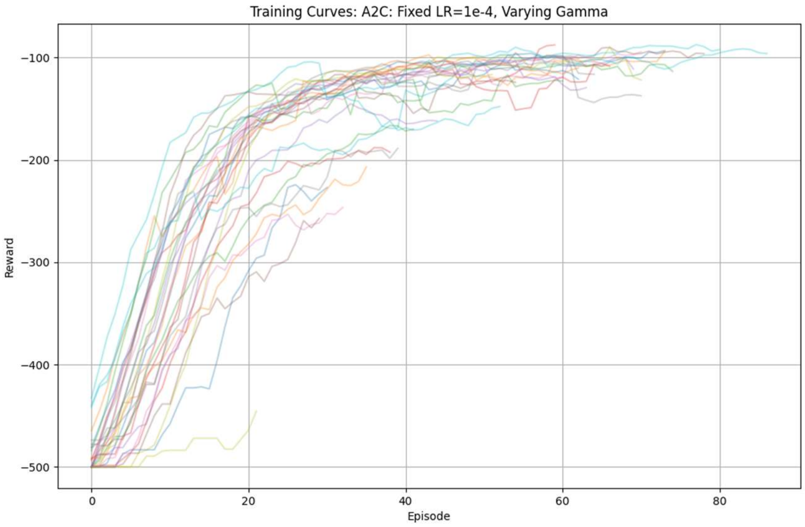


Figure 11. A2C Learning Curves while varying discount factor

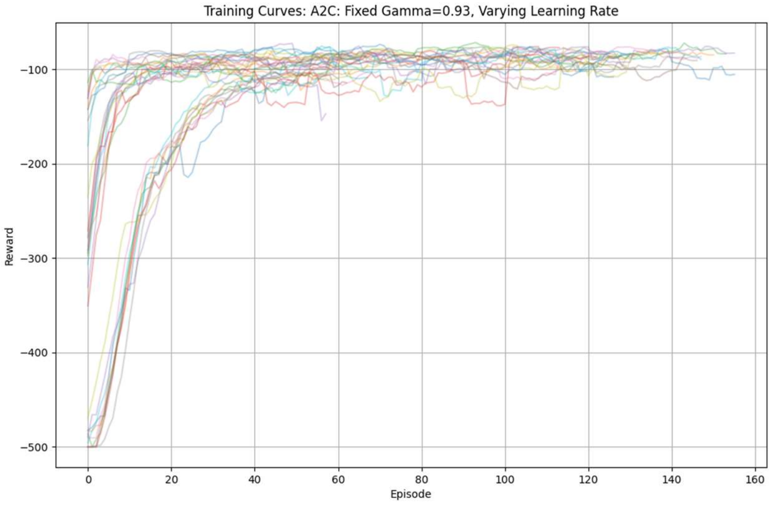


Figure 12. A2C Learning Curves while varying learning rate

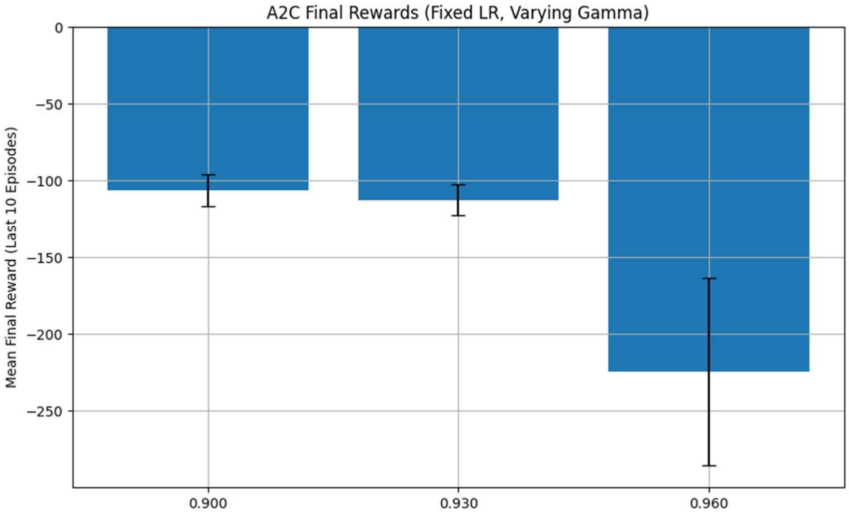


Figure 13. A2C rewards varying discount factor

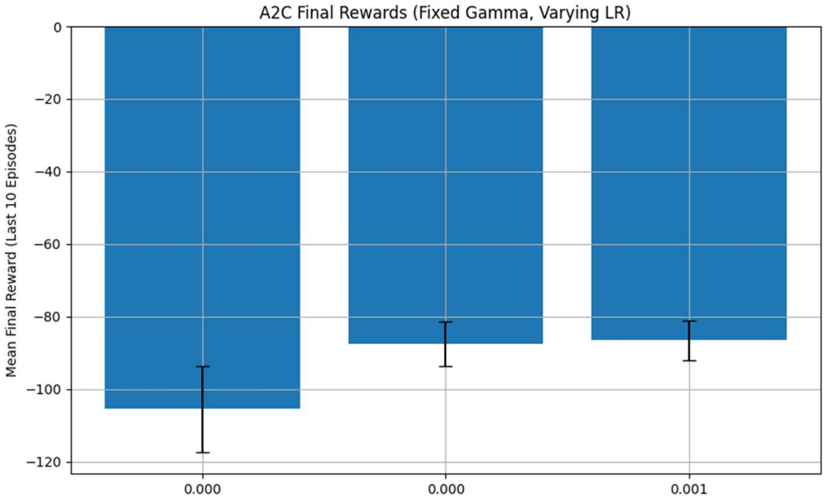


Figure 14. A2C rewards varying learning rate

A2C can be seen as noticeably continuing many of the otherwise observed trends with a lower uncertainty and consistency not only in the final results but also the learning curves path to convergence. The differing learning curves clearly show the paths that each factor takes to convergence. Noting the scale that nearly all of the A2C models outperform the PPO models on similar metrics.

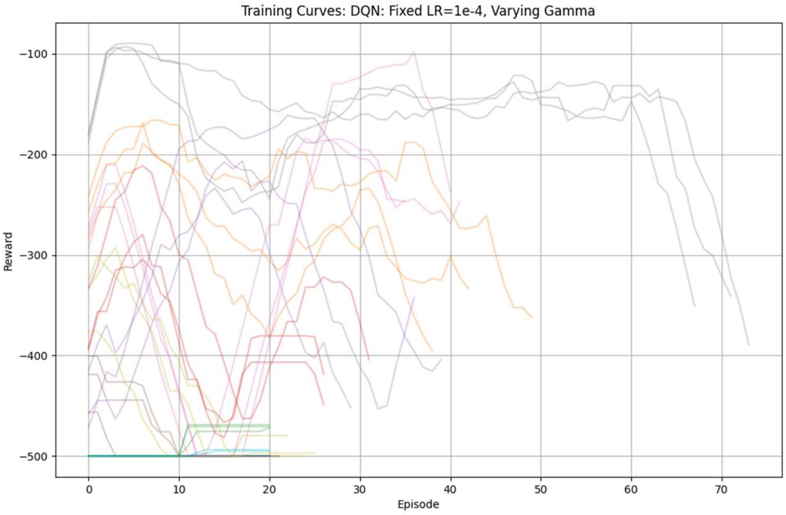


Figure 15. DQN Learning Curves while varying discount

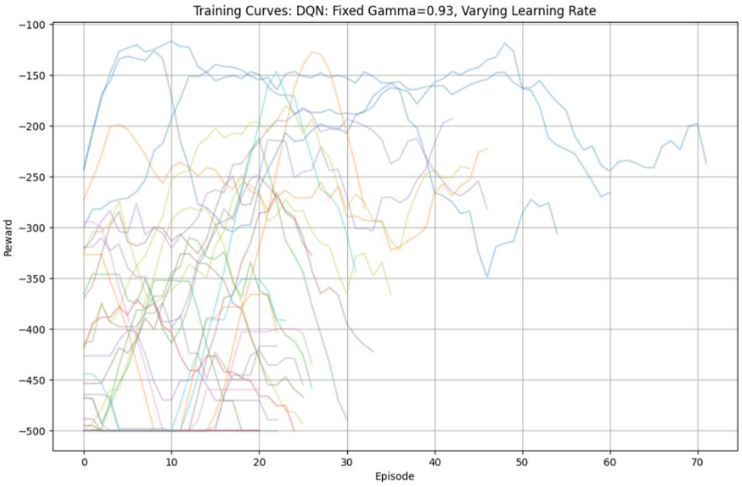


Figure 16. DQN Learning Curves while varying learning rate

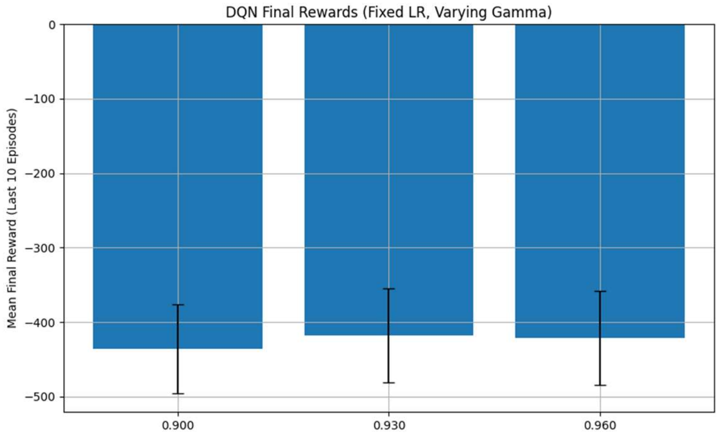


Figure 17. DQN rewards varying discount factor

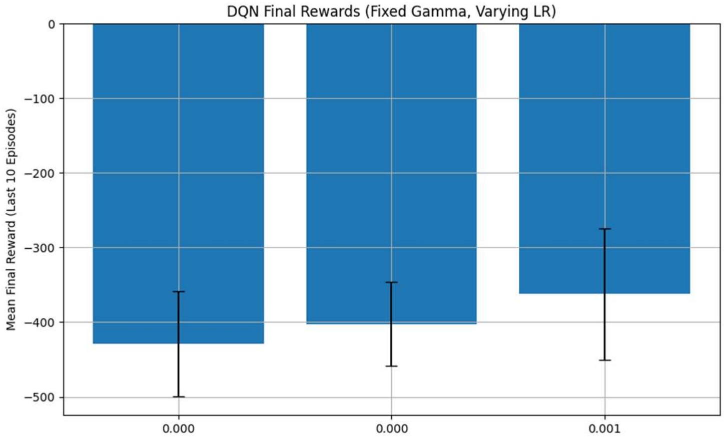


Figure 18. DQN rewards varying learning rate

While PPO demonstrated consistent convergence behavior, A2C outperformed it in both average final rewards and consistency across trials. DQN exhibited unstable learning and failed to converge in most settings, although a few isolated cases approached competitive performance.

5) Conclusion

This study evaluated the impact learning rate and discount factor on PPO, A2C, DQN algorithms using the Acrobot-v1 environment. Results showed that: Low learning rate led to non-convergence, whereas high learning rate led to oscillatory behavior; Low discount factor led to aggressive models that converged quickly while high discount factors created more patient models that converged slower but rarely took any steps that decreased rewards.

a. A2C consistently outperformed PPO and DQN in similar ranges, however it is worth exploring a wider range of learning rates and discount factors as trends for optimality indicated that different models preferred different learning rates and discount factors. PPO could benefit from higher discount factors, A2C could benefit from lower discount factors, and DQN had little impact from discount factors. PPO could benefit from a lower learning rate, whereas A2C and DQN could benefit from a higher learning rate.
b. A2C was the easiest to train, followed by PPO, and I could not get DQN to reliably converge. A2C demonstrated the most consistent convergent given the same hyperparameters.
c. PPO seemed the easiest to tune in the sense that poor hyperparameters could be accounted for with additional training episodes. A2C had a very clear optimal region of performance with respect to its hyperparameters, meaning that although it was harder to find this optimal region of performance, inside there was consistent and rapidly converging high performance. DQN did not converge consistently across any region of hyperparameters and was consistently exiting training episodes, indicating a need for continued variance in the hyperparameters provided.
d. This is the most I have worked with machine learning models thus far, and I think it was an interesting way to better understand the role that hyperparameters can play in the training process for various models. I learned about the characteristics of PPO, A2C, and DQN, and the impacts that learning rate and discount factor have on the performance of various algorithms. I think that this has been one of the most beneficial assignments I have seen in the class so far with respect to applications of the concepts that we have been learning up to this point

Future work could include a broader range of learning rates and discount factors applied, increased trials to decrease uncertainty and determine trends more consistently, and optimization techniques applied to identify optimal parameters for different algorithms.

Appendix

[1] Farama Foundation. *Acrobot-v1 Environment Documentation*. Gymnasium Classic Control Environments. Available at: https://gymnasium.farama.org/environments/classic_control/acrobot/. Accessed April 13, 2025.

[2] Stable-Baselines3 Developers. *Algorithm Guide and Environment Compatibility*. Stable-Baselines3 Documentation. Available at: https://stable-baselines3.readthedocs.io/en/master/guide/algos.html. Accessed April 13, 2025.

AERO 489
Project #2: Final
5/5/2025
Ian Wilhite

1) I chose to implement the 2-rope ball environment.
    a. The environment includes a random starting horizontal velocity, bar angle, angular frequency, and vertical velocity.
    b. The goal as stated was to raise the ball to a height of 10 without falling off the bar, and terminal conditions to ensure the model was meeting real-world constraints.
2) The reward structure and agent terminal conditions were structured to aide the agent to complete the assigned task and penalize suboptimalities in its completion of the task.

   Reward structure:
    a. Proportional height incentive – rewarding the agent for positive movement towards the goal condition.
    b. Proportional time disincentive – penalizing the agent proportionally for longer solutions.
    c. Nonlinear ball position penalty – minor penalty for small deviations from the middle of the ball, and higher penalties for allowing the ball to be closer to the edges of the bar.
    d. Minor maximum angle penalty – disincentivizing the agent from tilting the bar a large amount because of the time required to untilt the bar.

   Terminal condition structure:
    a. Checking the x position of the bar is not off the bar – if the ball has fallen off the edge of the bar the trial has ended.
    b. Checking the bar is upright – ending the trial if theta between negative and positive one-half pi.
    c. Checking the height is greater than negative 10 – this was added to prevent agent from falling into the void during training.
    d. Checking the height is greater than 10 – the success condition.

3) I chose to implement PPO because of its broad convergence in a variety of environments, and its notorious ease to train. During the training process, I varied

the ratio between the time and height proportional rewards, as well as the various minor penalties provided until the agent satisfactorily performed.

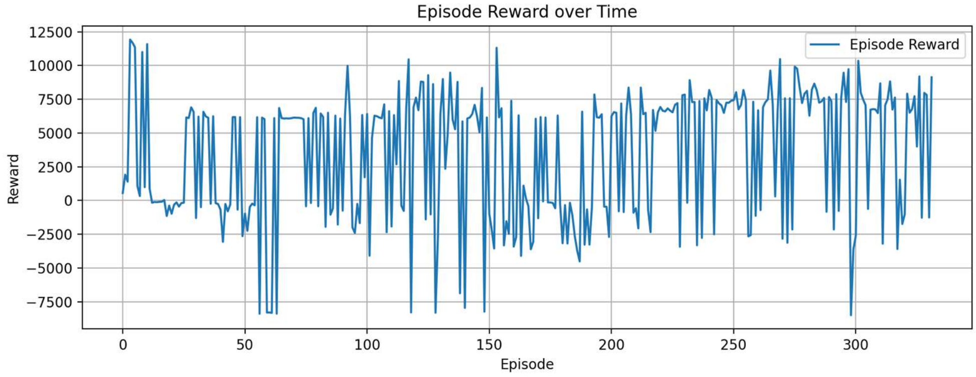
4) Short report:

This project allowed me to learn how to render real world scenarios and to apply the concepts of RL to real situations that I may face in industry. This project truly served to combine many of the skills of this course into a single relevant project relevant to the ways that these tools are currently being applied.

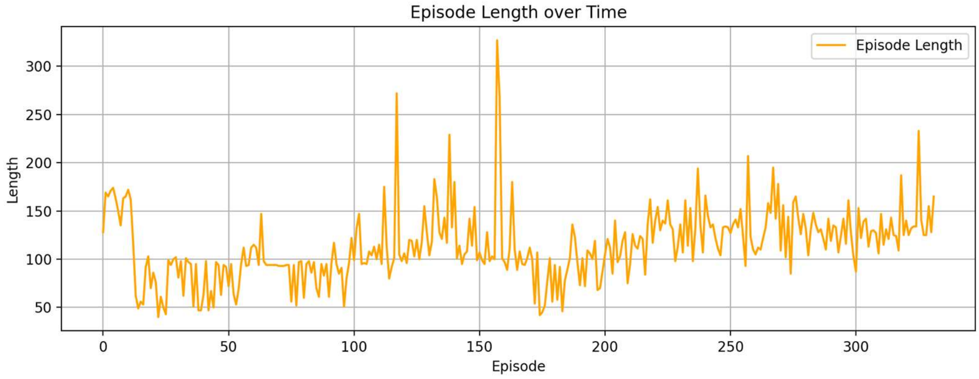
In designing the environment, a major aspect was determining which variables made a major impact on the state and would be required to fully describe any possible given state. Many variables excluded to decrease training time, and therefore the minimum number of variables possible were provided. The horizontal position of the bar and angle were added to represent the position of the system. Their derivatives were added to allow the agent to predict the motion of the system into the future. Their second derivatives were not added because the action the agent took would directly affect it, and therefore the agent did not need to know its own previous action, as it only needed to know what to do next. The height of the bar was intentionally not added to prevent the agent from learning to jerk the cables near the beginning or end, but rather to encourage the agent to act consistently throughout the trial.

The reward structure was revised continuously while the agent was being trained. The combination of linear, nonlinear, and weightings of rewards greatly changed the reaction of the agent to various states, and therefore how it responded to it. Reward shaping drastically affected nearly every aspect of the model's behavior and performance and eventually led to the success of the model. After pivoting to SAV, in tuning, I found that I needed to lower the learning rate to secure a more stable convergence, which finally secured a success rate of 79% in the randomly initialized environment.

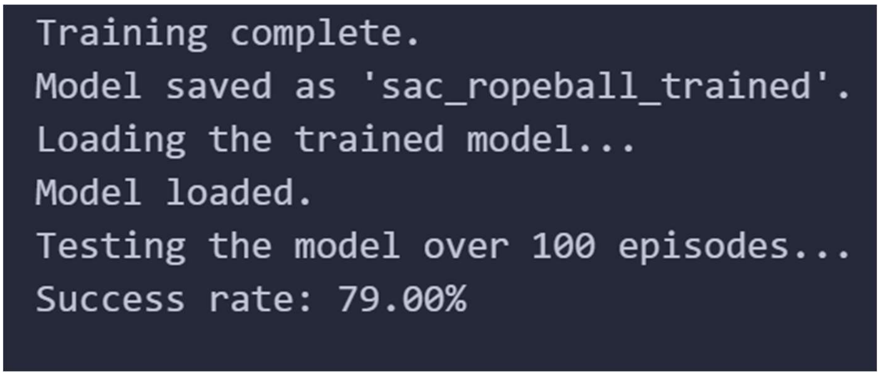
I chose to implement Proximal Policy Optimization (PPO) initially because of its known simplicity in convergence as I was testing the environment, however, I chose to pivot to Soft Actor Critic (SAC) for its ability to perform well in environments with high entropy, making it more suitable for modeling a system of multiple variables and lower training times.



The reward over episode graph shows the improvement of the model, both initially and in stability as the frequency of success increases but could still allow for another decrease in the learning rate and additional trials to allow the model to succeed with higher frequency.

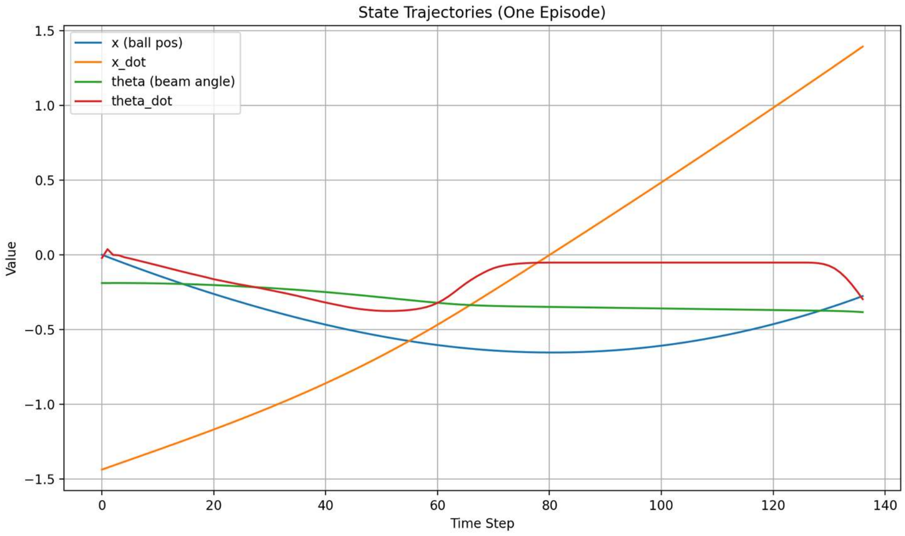


The episode length over time graph shows another perspective of the convergence strength. Initially, there is a sharp decrease in episode time as the model identifies the time penalty, and starts working around it as it learns to stabilize the ball. Around episode 200, the episode length started to rise alongside consistent episodic success, where the model was learning what tradeoffs were worth losing the time penalty points.

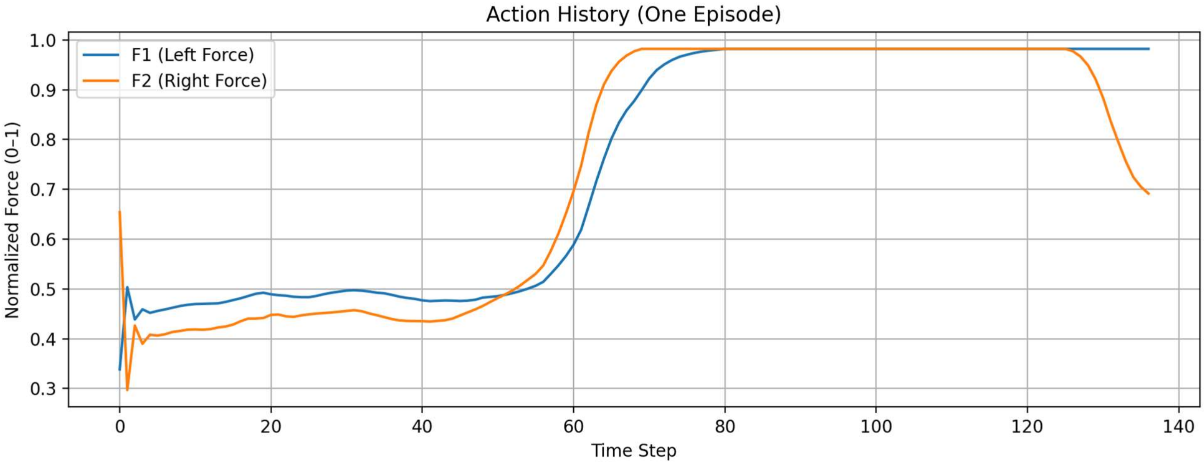


```
Training complete.
Model saved as 'sac_ropeball_trained'.
Loading the trained model...
Model loaded.
Testing the model over 100 episodes...
Success rate: 79.00%
```

To evaluate the model, 100 episodes with random initial angles, angular velocities, and ball speeds were performed, and the success rate for the trials were identified. The agent succeeded in 79 of the 100 episodes, for a success rate of 79% and securely meeting the established threshold of 70%.



The state trajectories plot indicates the agent is able to observe the negative ball position and velocity, and apply a positive torque to counteract the balls motion. The agent is able to correct for and maintain the bars angle, and minimizes the bar's angular acceleration then maintains it.



The system can be observed balancing the ball initially until time step 60, then maximizing a consistent force to raise the bar to the desired height as quickly as possible. The agent can be observed near the end of the trial as accounting for the ball's position crossing the center of the bar and beginning to tilt the bar to correct for overshoot.

Overall, the agent has learned to stabilize and raise the bar to the desired position with satisfactory success. The modeling of the environment represents skills used in industry, while the learning and tuning process is a final application of the skills covered in the second half of the course. This has been certainly one of the most unique classes I have taken, and I truly believe that I have learned a lot from it.

Back to Contents

2. Dynamics and Vibrations

Summary

This section contains projects from an undergraduate dynamics course evaluating dynamic problems, modeling approaches, and foundational physical simulation.

- P1: Bungee Jumping
- P2: Piston Kinematics
- P3: Mixer Torques & Kinematics

Contributors: Tori Abell, Alois Campbell, Jake Smith, Eddy Silva, Ian Wilhite

Key Skills: Dynamics, Vibrations, System Modeling, Validation, and Technical Writing.

Relevance: The projects emphasize direct conclusions based on experimental results. In these works, I often worked to write the code for the simulations and generate the charts to represent the teams findings.

Computational Assignment 1 - Bungee Jumping

TO: Adolfo Delgado
FROM: Group 7 of MEEN 363 - 502 (Tori Abell, Alois Campbell, Eddy Silva, Ian Wilhite)
Subject: Assignment 1, Bungee Jumper
Date: 02/21/2025

EXECUTIVE SUMMARY

The objective is to analyze how the bungee cord's material properties affect the jumper's motion. The height of the jump, the weight of the person, the stiffness of the bungee cord material, the damping effect of the material, and the air resistance all affect the motion of the jumper. The equation of motion was derived by using Newton's second law and summing the forces in the vertical direction. Two main equations were formed, one considering the damping from the cord being stretched and the air drag and the other considering when the cord was not in tension, the person only experiencing damping from air resistance. Based on the general equation of motion, the natural frequency of the jumper depends on the cord stiffness (k), and the mass of the person. As expected, a higher stiffness, k, results in a higher natural frequency. A lower cord stiffness requires a higher initial jump height because it allows the cord to stretch more and the person to fall further. A higher cord stiffness would result in a smaller initial jump height required, but the person is more likely to experience injury from a higher acceleration and force acting on them.

METHOD

The first step in creating the equation of motion for the bungee jumper is to create a free-body diagram (FBD). As seen in **Figure 1**, there are three forces acting on the jumper throughout most of the fall; the jumper's weight, total damping, and the spring force in the cord. The result of Newton's Second Law equation from the FBD is **Equation 1**. Using the motion equation and the "solve_ivp" scipy function in Python, a second-order differential equation can be solved for position, velocity, and acceleration. To analyze the effect of the cord's stiffness on the motion of the jumper, 10 evenly spaced k values between 40 N/m and 240 N/m were used in the position, velocity, and acceleration equations and plotted as functions of time. Three plots were then made for the material loss factor (η), seen in **Equation 2**, with values ranging from 0.15 to 0.30. For clarity, the simulations assume an initial jump height of 100m with the cord considered taut at 82m from the ground (x=0m).

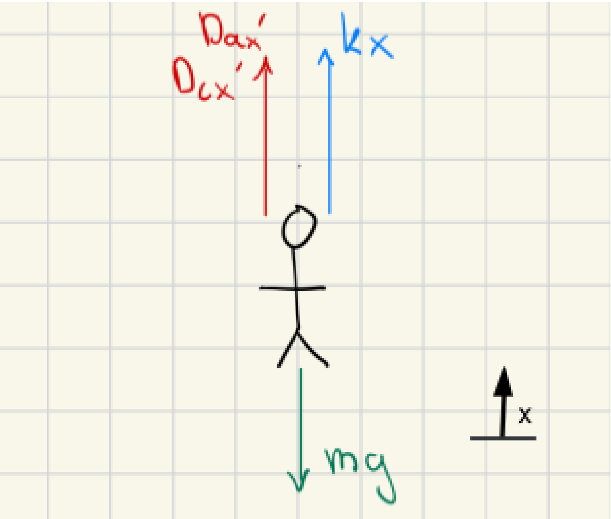


Figure 1: Free Body Diagram of Bungee Jumper

An important factor to consider with this problem is that the damping and spring force caused by the cord are inactive if the jumper is within 18 meters of the platform. 18 meters is the length of the cord so it will only be in tension after that distance. This means in **Equation 2**, D is equal to D_a within 18 meters.

Equations:
Motion Equation:

$$\Sigma F = mx'' = -mg + k(x_0 - l - x) + Dx' \qquad \text{Equation 1}$$

- x_0: Initial Position (m)
- x: Position (m)
- x': Velocity (m/s)
- x'': Acceleration (m/s^2)
- l: Length of Bungee Cord
- m: Mass (kg)
- k: Cord Stiffness (N/m)
- D: Total Damping Coefficient (Ns/m)

Total Damping Coefficient:

$$D = D_C + D_a = \frac{k\eta}{\omega_n} + D_a \qquad \text{Equation 2}$$

- D_C: Viscous Damping Coefficient (Ns/m)
- D_a: Air Resistance Equivalent Damping (8 Ns/m)

To better understand and analyze the jumper's motion, **Equation 3** showcases the natural frequency in terms of the equivalent cord stiffness and mass in the system. Meanwhile, **Equation 4** describes the cord's spring constant in terms of its material and physical properties.

Natural Frequency:

$$\omega_n = \sqrt{\frac{k}{m}} \qquad \text{Equation 3}$$

Cord Stiffness:

$$k = \frac{EA}{l} \qquad \text{Equation 4}$$

- E: Elastic Modulus (MPa)
- A: Cross-sectional area of Bungee Cord (m^2)
- l: Length of Bungee Cord (m)

PROCEDURE

A model was constructed to simulate the motion of the bungee jumper using the initial value problem (IVP) solver built into the Python Scipy library. First, a set of constants and initial values were set such that the simulation could use some values as constants and vary others over a provided range and step. Arrays were constructed to contain the values for each combination of the spring constant and loss factor. Each simulation was run and the arrays were propagated using the solution found with the IVP solver. A free fall curve was plotted to show motion without elastic force or damping resistance. A horizontal line representing the point of zero elastic deflection was plotted. The stored values were then printed to the console and stored as a CSV for processing.

The equilibrium points approach the point of zero elastic deflection (the rope length from the jump height) as the k value of the spring increases. This phenomenon follows the understanding that stiffer springs will deflect less under equal loading conditions.

In the reported times of maximum acceleration, the third reported value aligning with k=40 and η=0.30 shows the maximum acceleration at t=0. This discrepancy can be attributed to the increasing η value which led to the first oscillation having a lower maximum magnitude of acceleration than the free fall region. The reported time value represents the initial acceleration due to gravity, the point of minimum velocity, and before the bungee cord is engaged. This is to be expected because, during the first oscillation, the peak remains below the equilibrium point of the spring. Therefore, the body remains in damped harmonic motion after the initial free fall. As seen in the free fall region of the acceleration chart of **Figure A**, the acceleration begins only due to gravity; however, as the body falls and accelerates, the air resistance increases proportionally to the velocity, leading to an exponential decrease in the magnitude of the acceleration as the downward velocity of the jumper leads to a positive force on the body.

This implies several things: that the acceleration is at a maximum in the free fall region, that once the body enters the damped harmonic motion region it can never re-enter the free fall region, and that within the initial free fall region, the acceleration is at its maximum at the initial state. Therefore, the reported time value for the acceleration must be correct.

For reported values, both parameters were varied simultaneously, but to aid in visualizing trends, each was plotted while the other was fixed, as depicted in **Figures 2 and 3**. The simulations were run for 20 seconds as these would allow for quicker computing times, and the jumper would be considered to be moving slow enough to be recovered safely.

RESULTS and DISCUSSION

With the equations of motion, velocity, and acceleration solved using Python's matplotlib library, **Figures 2 and 3** show the position of the jumper as a function of time with varying K and η values. As per the assumptions, these figures include a "free fall" for the first 18m of descent from an initial position of 100m from the ground (x=0m), as well as a sample "free fall" with no bungee cord for reference. The 18m when the cord begins to act on the jumper are also marked by a dotted line in both graphs.

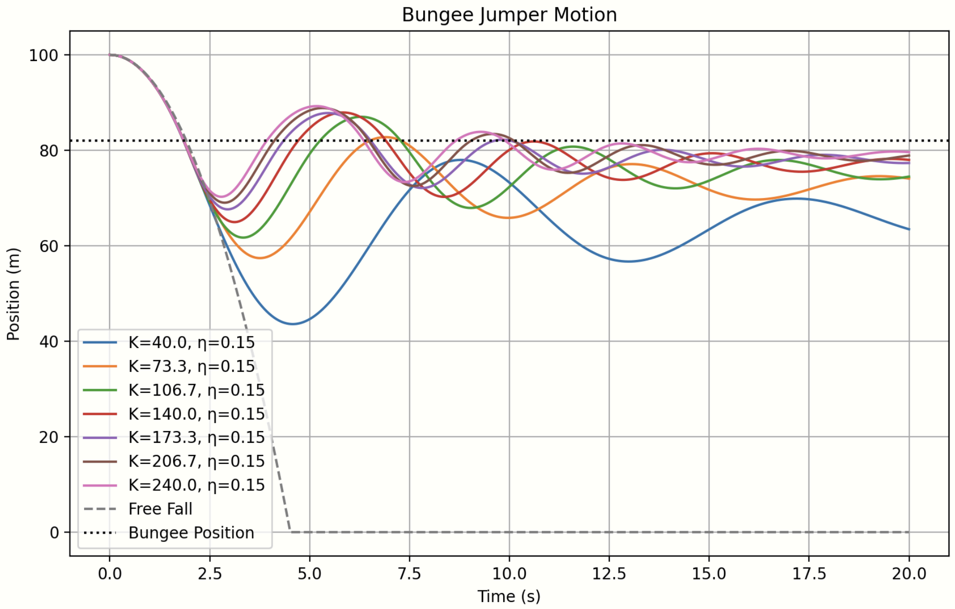


Figure 2: Bungee Jumper position as a function of time with varying K values

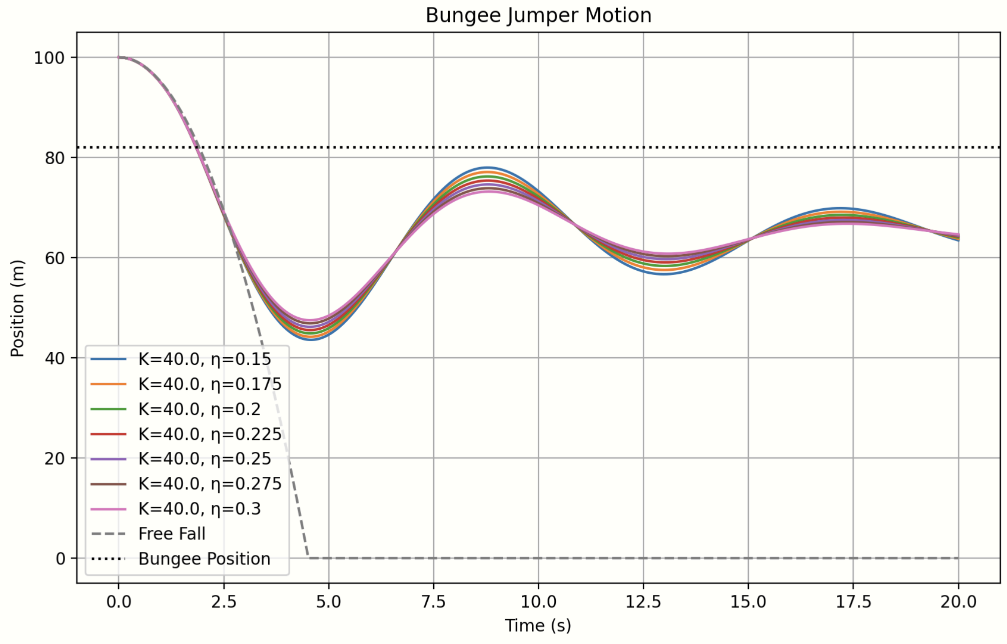


Figure 3: Bungee Jumper position as a function of time with varying η values

From **Figure 2**, it can be seen that there is a lot of variation in both the amplitude and period of the oscillatory motion of the jumper. As these values of k are bounded between 40N/m and 240N/m and η between 0.15 and 0.30, a minimum fall position, maximum velocity, and maximum acceleration can all be identified using **Table A** and interpreting **Figures 2, 3, and A**. At k=40N/m and η = 0.15, the absolute lowest point the jumper reaches is 45.8m from the ground after 4.8s, his maximum acceleration of 11.4m/s^2 is obtained after 4.4s and his maximum absolute velocity is 18.7m/s at 2.5s. In terms of his position, this would mean that maximum acceleration is when he is being pulled up before the first bungee recoil and maximum speed is at the end of the freefall. Furthermore, all of these maxima emerge from the first ½ period of motion, further insinuating the importance of the cord's stretching abilities to dictate the subsequent oscillations. Additionally, from varying k and η coefficients, k was noticed to have a larger impact on the behavior of the jump following a trend of lower maximum acceleration and bigger fall height as k decreased. From **Table A**, this is seen by the k=240N/m value resulting in a maximum acceleration of 27.1m/s^2 and a minimum tower height of 28.2m, which are significantly more dangerous than the previously seen k=40N/m values. Following this line of reasoning, it would be reasonable to assume that is the reason why jumpers can fall without sustaining serious injury from a high tower. The force experienced at any moment in time by the jumper is slowly being "lost" in the cord the more it stretches and results in a less violent jolt when it comes time to reverse the direction of their trajectory. Meanwhile, maximum velocity remained more or less constant throughout all trials (around -18m/s) at the tail end of the freefall.

As such, the simulation showed that an increasing K coefficient would lead to a higher maximum acceleration, making k=40N/m the "safest" option while still allowing for the jumper to experience the longest fall. This is consistent, as k depends on the elastic modulus, cross-sectional area, and the original length of the cable. The original cable length is given as 18m. Natural rubber or other elastomers are typically used for the core of the cable. The cable is typically covered by a woven synthetic material such as nylon to protect the rubber from heat/sunlight. Based on these two considerations, the material selection for the core is narrowed down to natural rubber, which has an elastic modulus in the range of 3.45Mpa to 24.1Mpa [1].

Another important consideration is the possibility of stress relaxation and hysteresis of the bungee cable, which would result in permanent deformation and weakening of the cable and safety concerns. "Hysteretic" would refer to the uneven forces being exerted on the person between falling downward and upward. It also is dependent on the uneven compression and tension rope friction properties (i.e. the rope can compress easier than stretch or vice-versa). Since cord manufacturing affects η and K, the mechanical properties of the rope affect the studied displacement, velocity, and acceleration. Therefore, the manufacturing process of the rubber should involve vulcanization of the rubber to reduce the possibility of stress relaxation. Vulcanization introduces cross-links between the polymer chains, strengthening the material. Research proves that more crosslinks produced during the vulcanization process result in the material being capable of withstanding more stress without permanent deformation[2]. Vulcanization also leads to higher elasticity (E=1.5Mpa [3]), and consequently, a lower stiffness(k). Furthermore, vulcanization improves the cable's ability to withstand hysteresis, because the cable will also be stronger under compression after vulcanization[2]. In particular, carbon-containing cross-links are proven to increase the lifespan of vulcanizates[2] compared to other materials typically used to vulcanize natural rubber. Assuming the diameter of the cord is around 2.5cm, the resulting stiffness value for a fixed cord length of 18m, using **Equation 4**, is 41 N/m. The k value corresponds to the lowest k value from previous evaluations, indicating that it will meet the vertical distance constraints too. Based on the figures, the k value of 40 N/m results in enjoyable conditions for the jumper, and the jumper will experience minimal amounts of upward acceleration. The maximum acceleration the jumper will reach is just under 15m/s^2, or less than 1.5g, which is less than a person will experience on a typical roller coaster[4], but still more than they experience on a daily basis. They would also experience only around 3 ups/downs within 20 seconds, the assumed time it takes for their velocity to be low enough to be safely recoverable. Thus, the jumper will have fun without putting their safety at risk.

Finally, ride expectations also affect the model in unaccounted-for ways. They have to do with the behavior of the person dropping. As this is a person, they can flail their arms in/out to decrease/expand their surface area in contact with the air. This could make the assumed constant damping coefficient of the air (Da=8) change at a moment's notice - impacting the simulation by slowing or quickening descent.

CONCLUSIONS

In conclusion, the jumper can complete the jump without sustaining injury by observing the parameters that influence maximum acceleration and velocity. Such parameters include the cord stiffness (k), the cord length, the jumper's vertical starting position, the cord's loss factor, and the manufacturing and mechanical properties of the cord. Ultimately, a lower K value and higher η value would mean that the jumper experiences less maximum acceleration while also coming to the equilibrium position as quickly as possible. This would ensure the jumper has a smoother experience that also comes to an end quicker as opposed to violently being pulled back and suspended in oscillation for longer. Overall, taking into account the safety and enjoyment of the person jumping, it is most ideal to choose a vulcanized natural rubber cord with a stiffness of around 40N/m, and for the jumper to start at an absolute minimum height of 55 meters. Then, the jumper will not be injured due to acceleration or due to hitting the ground.

APPENDIX

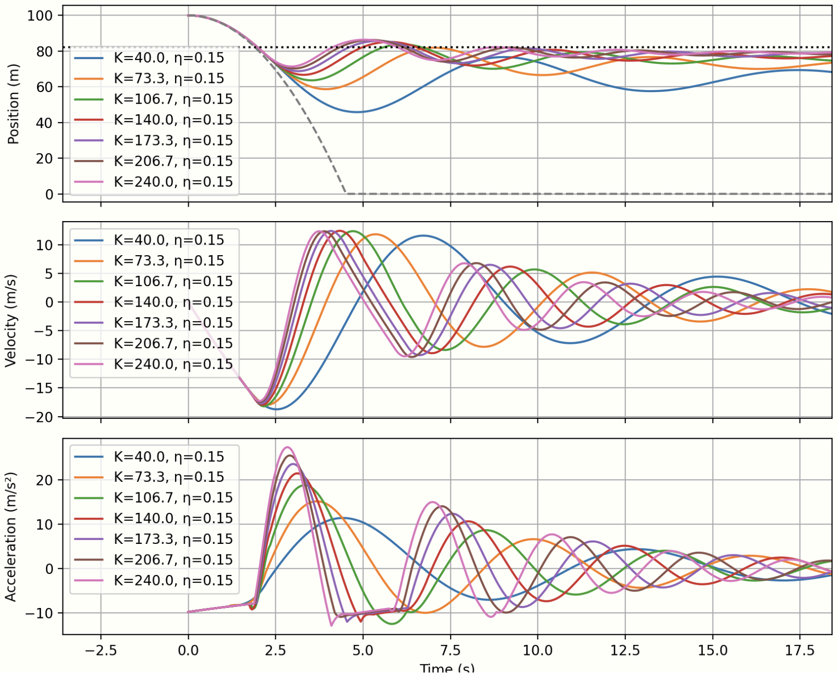


Figure A: Modeled Equations of Motion, Velocity, and Acceleration using varied K and η values.

Table A: Numerical values of select cases in the simulation with accompanying maximum acceleration, velocity, and deflection

Eta	*K (N/m)*	*Max_acc (m/s^2)*	*Max_acc _time (s)*	*Max_vel (m/s)*	*Max_vel_ time (s)*	*Min_pos (m)*	*Min_pos_ time (s)*	Minimum Tower Height req. (m)	Cord Stretched Length (m)
0.150	*40.0*	*11.4*	*4.4*	*18.7*	*2.5*	*45.8*	*4.8*	54.2	36.2
0.225	*40.0*	*10.5*	*4.3*	*18.1*	*2.5*	*47.8*	*4.8*	52.2	34.2
0.300	*40.0*	*9.8*	*0.0*	*17.6*	*2.4*	*49.5*	*4.8*	50.5	32.5
0.150	*68.6*	*14.7*	*3.7*	*18.0*	*2.3*	*57.5*	*4.0*	42.5	24.5
0.300	*68.6*	*12.5*	*3.6*	*16.8*	*2.2*	*60.3*	*4.0*	39.7	21.7
0.150	*97.1*	*18.0*	*3.4*	*18.3*	*2.2*	*62.4*	*3.6*	37.6	19.6
0.300	*97.1*	*15.9*	*3.2*	*17.8*	*2.1*	*64.3*	*3.6*	35.7	17.7
0.150	*240.0*	*27.4*	*2.8*	*17.3*	*2.0*	*71.3*	*2.9*	28.7	10.7
0.300	*240.0*	*27.1*	*2.7*	*18.0*	*2.1*	*71.8*	*2.9*	28.2	10.2

REFERENCES

[1] “NR- Natural Rubber,” Rahco-Rubber, https://rahco-rubber.com/materials/nr-natural-rubber/.
[2]”Natural Rubber, Vulcanized (NR, IR, Polyisoprene),” Matweb, https://www.matweb.com/search/datasheet_print.aspx?matguid=6588439546ac4492965c894ddff3f5da.
[3] S Baxter, M.A.A Wilson, 1963, “Stress relaxation of vulcanized rubbers III — Experimental study, Polymer,” Volume 4, Pages 163-173, ISSN 0032-3861, https://doi.org/10.1016/0032-3861(63)90023-5.
[4] “How a Coaster Moves,” Coaster Force, https://coasterforce.com/physics/.

Computational Assignment 2 - Kinematics of Crankshaft and Rod

TO: Adolfo Delgado
FROM: Group 7 of MEEN 363 - 502 (Tori Abell, Alois Campbell, Jake Smith, Eddy Silva, Ian Wilhite)
Subject: Assignment 2, Crankshaft and Rod
Date: 04/02/2025

EXECUTIVE SUMMARY

The objective is to derive and analyze the displacement, velocity, and acceleration of the slider (point B) and the center of mass of the connecting rod (point G) as functions of the crank angle (θ). Using a constant crank angular velocity of 300 rpm, kinematic equations for point B and point G were derived using the geometric method to describe the motion of the system. The analysis was performed for three different crank-to-rod length ratios: P = 0.1, 0.3, 0.5. Computational simulations generated time-based plots of position, velocity, and acceleration for both point B and point G over two full revolutions of the crank. The results show significant variation in position, velocity, and acceleration of points B and G depending on the value of P. As P increases, the magnitude and variation of the slider's velocity and acceleration increase, reflecting stronger dynamic effects. This analysis confirms the importance of geometric ratios in crank-slider mechanisms and supports using Newton's Second Law to identify force trends acting on the system components.

METHOD

To obtain kinematic equations for point B, a geometric approach was used. It consisted of adding the horizontal lengths of the members and setting that equal to the i vector position of B (x_B) at any given time. The vertical distance with respect to B (y_B) was set to 0 as there is no vertical translation at that point. Nevertheless, a kinematic constraint was determined by setting the bar heights equal to each other. The results are the **Equations (1) and (2)** for position below. Taking the derivative of these equations results in the equations for velocity and acceleration - **Equations 3, 4, 5, and 6**. The position, velocity, and acceleration of point G were then evaluated using vector properties based on the given position of G (center of rod l_2).

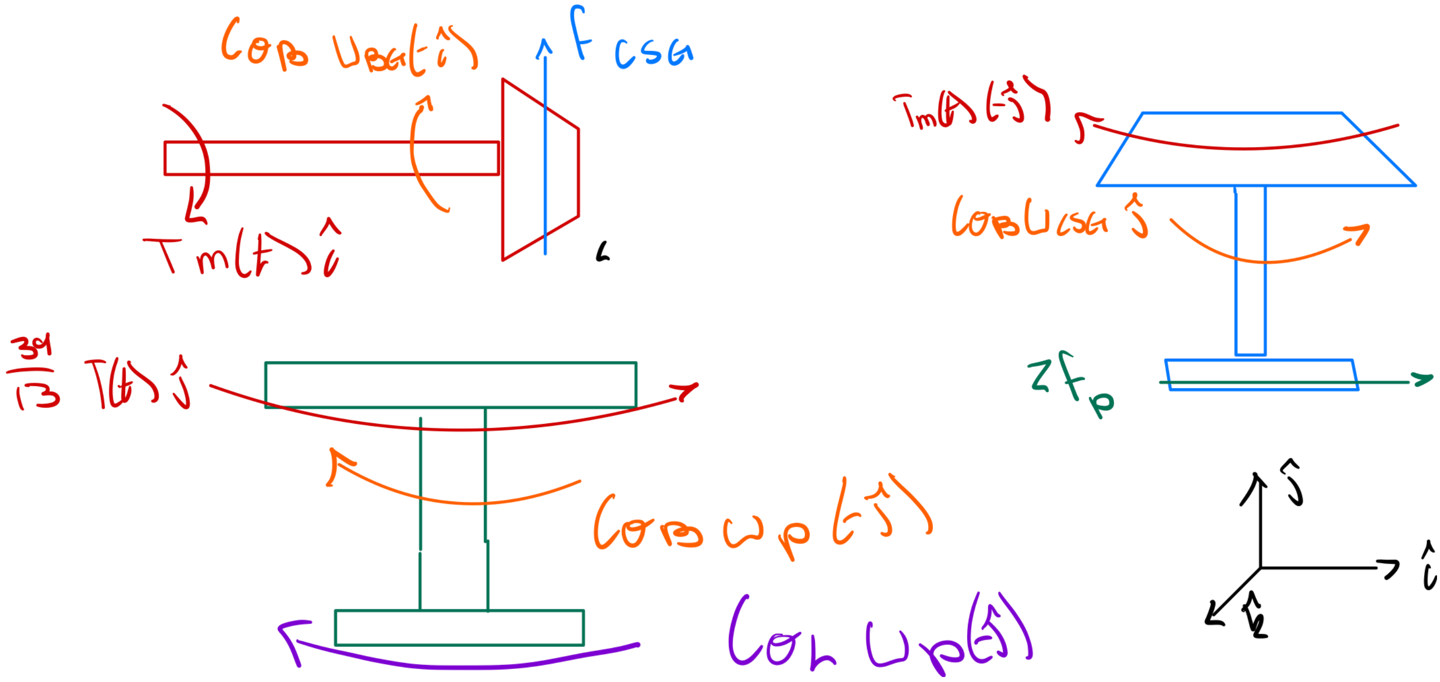


Figure 1: Diagram of Bodies

Geometrical derivation of equations of motion for point B:

Given:

$P = \frac{l_1}{l_2}$, θ'=ω=300 rpm, $l_T = 7ft$, $l_T = l_1 + l_2$

Position (X, Y Components)

$$l_1 cos(\theta) + l_2 cos(\phi) = x_B \tag{1}$$

$$l_1 sin(\theta) = l_2 sin(\phi) \tag{2}$$

Velocity (X, Y Components)

$$- l_1 sin(\theta) * \theta' = l_2 sin(\phi) * \phi' + \dot{x}_B \tag{3}$$

$$l_1 cos(\theta) * \theta' = l_2 cos(\phi) * \phi' \tag{4}$$

Acceleration (X, Y Components)

$$- l_1 sin(\theta) * \theta'' - l_1 cos(\theta) * \theta'^2 - l_2 cos(\phi) * \phi'^2 = l_2 sin(\phi) * \phi'' + \ddot{x}_B \tag{5}$$

$$l_2 cos(\phi) * \phi'' = l_2 sin(\phi) * \phi'^2 + l_1 cos(\theta) * \theta'' - l_1 sin(\theta) * \theta'^2 \tag{6}$$

θ'' term goes to 0 due to constant velocity, zero acceleration condition given

The vector equation properties to find the position, velocity, and acceleration of point G from the EOM derived for point B:

$$r_{OG} = r_{OB} + r_{BG} = x_B(i) + \frac{l_2}{2} * (cos(\phi)(- i) + sin(\phi)j) \tag{7}$$

$$v_G = v_B + \omega_{BG} \times r_{BG} = \dot{x}_B(- i) + \phi'(- k) \times \frac{l_2}{2} * (cos(\phi)(- i) + sin(\phi)j) \tag{8}$$

$$a_G = a_B + \omega_{BG}' \times r_{BG} + \omega_{BG} \times \omega_{BG} \times r_{BG}$$

$$a_G = \ddot{x}_B(- i) + \phi''(- k) \times (\frac{l_2}{2} * (cos(\phi)(- i) + sin(\phi)j)) + \phi'^2(\frac{l_2}{2})(cos(\phi)(i) - sin(\phi)j) \tag{9}$$

Cramer’s Rule

To analytically solve for the variables in the equations above, Cramer's Rule was used. Below is an example using the angular velocities.

$$\begin{bmatrix} - l_1 sin(\theta) & - l_2 sin(\phi) \\ l_1 cos(\theta) & l_2 cos(\phi) \end{bmatrix} \begin{bmatrix} \theta' \\ \phi' \end{bmatrix} = \begin{bmatrix} \dot{x}_B \\ 0 \end{bmatrix}$$

$$\theta' = \begin{vmatrix} \dot{x}_B & - l_2 sin\phi \\ 0 & l_2 cos\phi \end{vmatrix} \div \begin{vmatrix} - l_1 sin(\theta) & - l_2 sin(\phi) \\ l_1 cos(\theta) & l_2 cos(\phi) \end{vmatrix}$$

$$\phi' = \begin{vmatrix} -l_1 sin(\theta) & \dot{x}_B \\ l_1 cos(\theta) & 0 \end{vmatrix} \div \begin{vmatrix} -l_1 sin(\theta) & -l_2 sin(\phi) \\ l_1 cos(\theta) & l_2 cos(\phi) \end{vmatrix}$$

After finding a general solution for position, velocity, and acceleration about points B and G, Python was used to create a data set spanning 2 periods using time as the x-axis and the solved terms in the y (seen in the figures below).

PROCEDURE

Using the derived kinematic equations (**Equations 1-6**) for point B, the numpy library in Python was used to construct the model and plot the results. The plots were constructed in terms of P, which is the ratio between l_1 and l_2, to observe the effect of varying the ratio between l_1 and l_2. These figures were compiled to create three subplots in **Figure 2**.

A similar approach was taken for the kinematic equations of point G (**Equations 7-9**) in **Figures 3, 4, 5, 6, and 7**. To ensure no mistakes were made, the kinematic equations of point A were derived geometrically and averaged with B to determine an alternate expression of G (**Appendix Equations**). Since the AB bar is symmetric, the center of mass must be equally positioned between points A and B, rendering this alternate version acceptable for the simulation. In other words, the X position of G is the average of A and B, and the Y position of point G is found by dividing the Y position of A by two. Since this latter approach was significantly easier to code, it was the approach used in the final simulation.

Once these equations and parameters were set, the given constant velocity of θ' (=300 rpm) and θ'' (=0) were used to solve for the rest of the unknowns using the built-in numpy solver, ultimately rendering all the tables and figures discussed.

RESULTS and DISCUSSION

Figure 2 shows the horizontal position, velocity, and acceleration of point B with varying ratios between l_1 and l_2 (P). When P equals 0.1, for example, l_1 is 10% of l_2. The length of l_1 plus l_2 is given as 7ft, therefore when P=0.1, l_1=0.7ft and l_2 =6.3ft. When point A moves below y=0 in the positive x-axis, point A and point B begin moving in the -x direction, until link OA and link AB are both horizontal and lying along the x-axis at y=0. At this position, the horizontal location of points B and A will be at their minimum. Point A will be at x = -0.7ft and point B will be at x = 6.3 ft. **Figure 2** displays this expected result when the position of B is at its minimum, at 6.3ft. The same logic applies to all values of P. Furthermore, the graph displays three points at which the horizontal location of B is 7ft. At these points, links OA and AB are both horizontal along the x-axis, and the horizontal location of A is positive. The first time this occurs is when θ and ϕ are zero. Later occurrences occur at the end of each rotation until reaching the max angle simulated (2π radians).

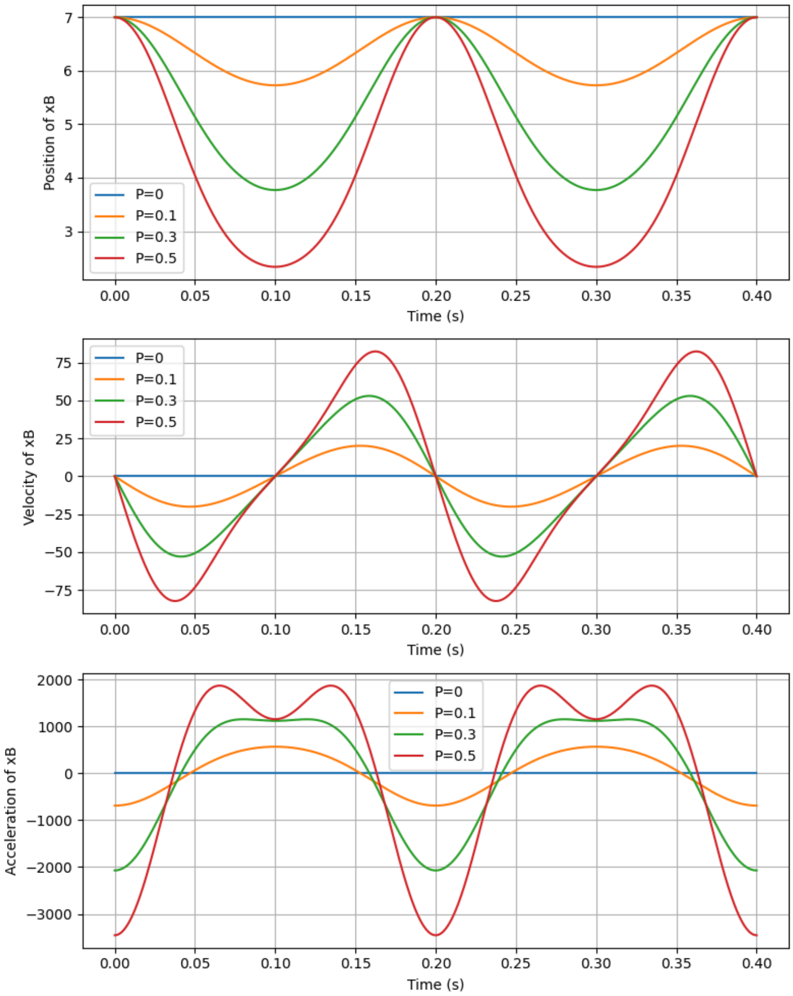


Figure 2: Plot of point B horizontal position, velocity, and acceleration as a function of time with varying P values.

Additionally, the maximum and minimum velocities of point B and the time they occur are displayed in **Figure 2** and recorded in **Table 1** in the appendix. The maximum and minimum acceleration for point B and the time they occur are recorded in **Table 2** in the appendix. The maximum/minimum velocity magnitude is approximately 82.3 ft/s. The magnitude of the minimum acceleration is 3454 ft/s^2, and the magnitude of the maximum acceleration is approximately 1150 ft/s^2. Both the maximum and minimum velocities and acceleration occur when P =0.5, right before and after point A is on the y-axis. The positive maximum acceleration (expanding crankshaft) and by extension the maximum force takes place right before and after the two bars are horizontal to each other, with point A being in the negative x direction. These accelerations also result in accompanying maximum magnitudes of velocity, which come about from these abrupt acceleration local maxima. In the inverse case, where point A is located in the positive x-axis at y=0, the acceleration is at its absolute maximum with the rod fully extended and preparing to contract/expand fully, depending on the cycle direction.

Figure 3 displays the varying vertical position of point G and the corresponding horizontal position. Point G's motion is a combination of a circular motion from the crank rotation, θ, and the back and forth translations of point B. Due to the combined effects, point G's motion is oblong/"egg-shaped", becoming thicker at the bottom of the stroke than the top. This asymmetrical motion begins to appear differently as L_1/L_2 decreases, looking more like a symmetrical oval. This is because when L_1 is small compared to $L_{2,}$ the motion of L_2 acts more like a linear translation, as the circular motion is less prevalent. This leads to the motion of G being thinner and more symmetrical.

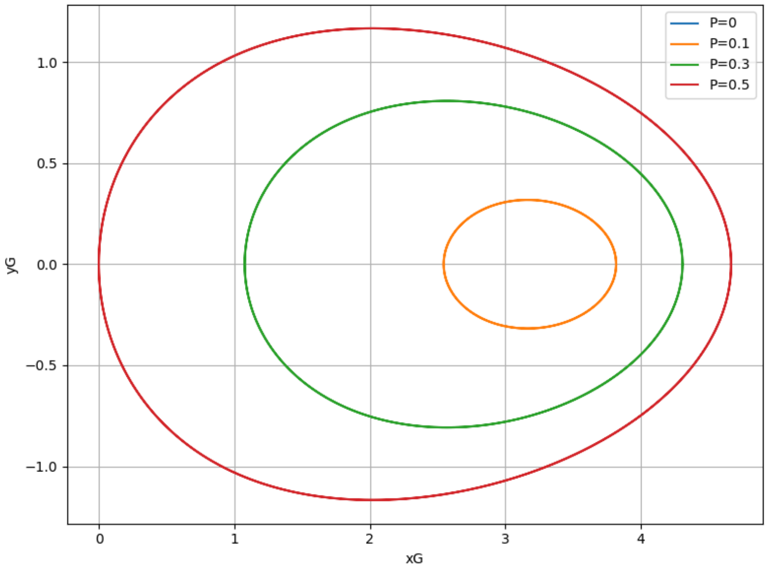


Figure 3: Plot of the motion of point G vertical position vs horizontal position ($Y_G$vs X_G).

For the P=0 case, there is only one fixed geometry solution available. Therefore, it does not show on the graph as there is no variation in position observable. This same logic applies to all other figures for the position, the velocity, and acceleration of points B and G. Since there is no change in geometry due to the constraints, there is no motion and no observable changes, creating a flat slope at 0.

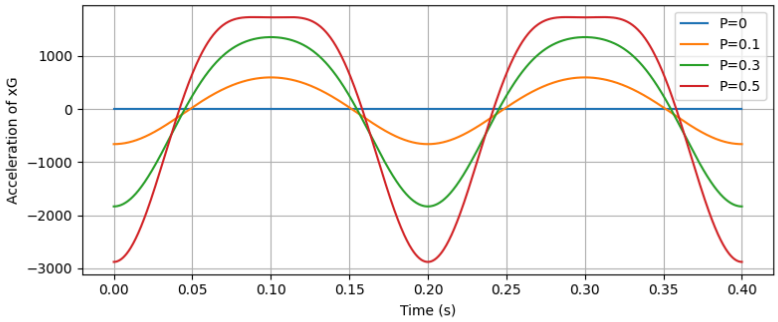


Figure 4: Plot of point G horizontal acceleration as a function of time with varying P values.

Newton’s second law can be applied to the problem based on the acceleration. Since force is mass times acceleration, point G and point B experience the greatest force in the x direction, where the magnitude of the acceleration is the greatest, at times indicated by the solution. **Figures 4 and 5** indicate that the acceleration of point G in the horizontal direction has the greatest magnitude where the links reach their maximum or minimum x position (see **Figure 6** in the appendix for the plot of the position and velocity of point G). Therefore, link AB experiences the greatest force at its center of gravity when it changes directions along the x-axis.

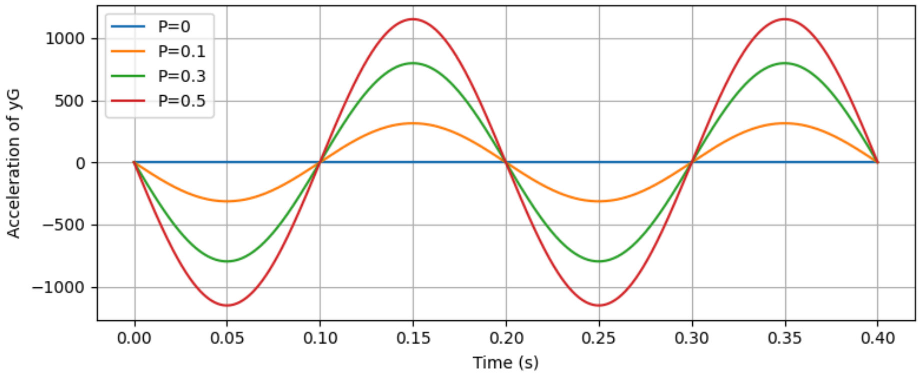


Figure 5: Plot of point G vertical acceleration as a function of time with varying P values.

Figure 5 illustrates the vertical acceleration of point G over time for varying P values. The vertical position in **Figure 7** in the appendix exhibits a sinusoidal pattern that reflects the crank's rotational motion, while the velocity and acceleration curves show how rapidly this vertical motion changes throughout the stroke. As P increases, the vertical velocity and acceleration both grow in magnitude, indicating a stronger vertical influence from the crank's rotation. The peaks in acceleration occur near the midpoints of each stroke, where the vertical direction reverses and inertial forces peak. This vertical dynamic is particularly important in applications like internal combustion engines, where vertical forces translate into vibrations and stresses that impact structural design and operational smoothness.

CONCLUSIONS

In conclusion, the kinematic analysis of the crank-slider mechanism reveals significant dependence on the crank-to-rod length ratio (P). As P increases, the dynamic effects on both point B and point G become more pronounced, with greater velocity and acceleration magnitudes. These findings have direct implications for the design and performance of reciprocating systems, where balancing force transmission and inertial effects is crucial. Newton's Second Law reinforces that maximum force corresponds to maximum acceleration, which occurs at predictable angular positions (fully expanded or contracted). This analysis provides valuable insight into optimizing component dimensions to reduce wear, minimize vibrations, and ensure efficient mechanical performance. Ultimately, this makes the mechanism ideal for the use case given of the crankshaft two-stroke engine.

APPENDIX

Point B Maximum and Minimum Velocities	
Time (s)	**Velocity (ft/s)**
0.0376	-82.3
0.163	82.3
0.237	-82.3
0.362	82.3

Table 1: Maximum and minimum velocity of point B and the time it occurs.

Point B Maximum and Minimum Acceleration	
Time (s)	**Acceleration (ft/s^2)**
0	-3454
0.119	1150
0.080	1150
0.200	-3454
.280	1150
0.319	1150
0.400	-3454

Table 2: Maximum and minimum acceleration of point B and the time they occur.

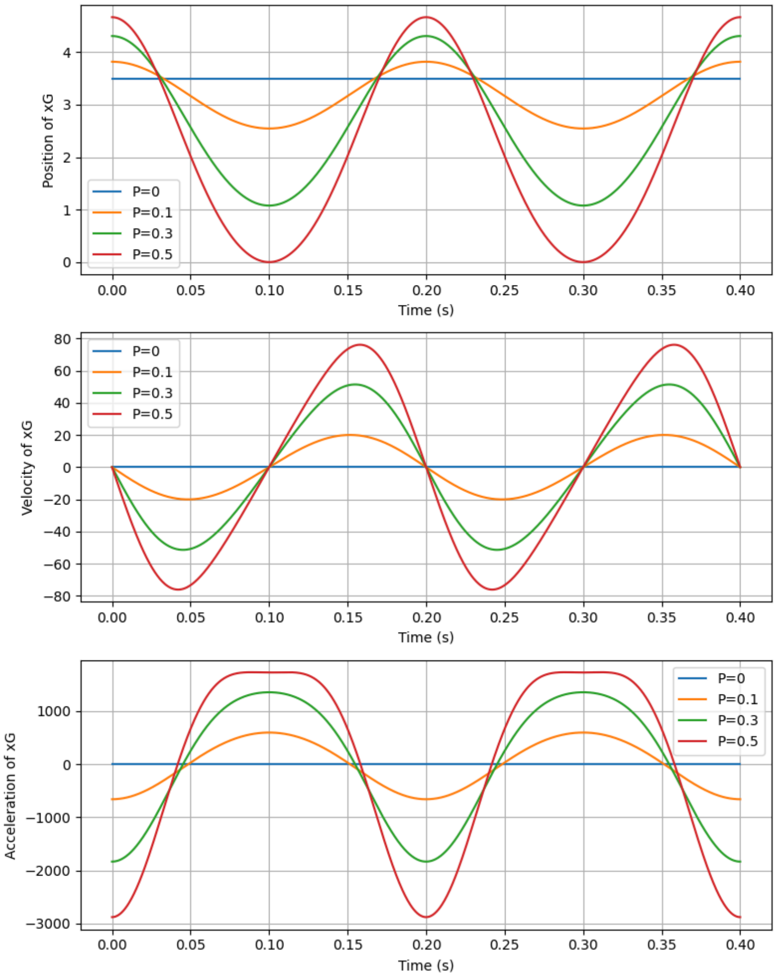


Figure 6: Plot of point G horizontal acceleration as a function of time with varying P values.

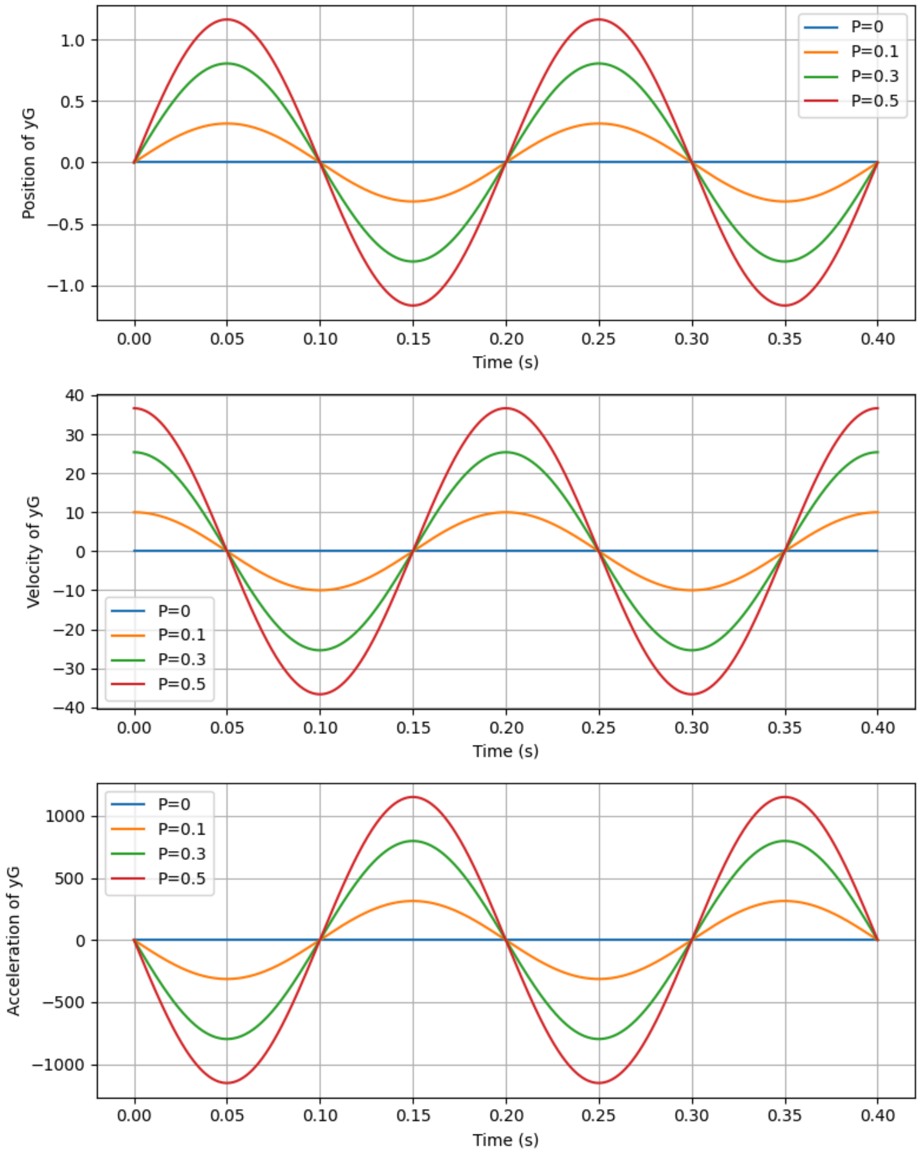


Figure 7: Plot of point G vertical position, velocity, and acceleration as a function of time with varying P values.

Alternative proof for point G, used to verify equations, and the simulation used:
For point G, an average between points A and B was taken:

Point A:
Position (X, Y Components)

$$l_1 cos(\theta) \ = \ x_A \tag{A.1}$$

$$l_1 sin(\theta) = y_A \tag{A.2}$$

Velocity (X, Y Components)

$$- l_1 sin(\theta) \ * \ \theta' \ = \dot{x}_A \tag{A.3}$$

$$l_1 cos(\theta) \ * \ \theta' \ = \dot{y}_A \tag{A.4}$$

Acceleration (X, Y Components)

$$-\ l_1 sin(\theta)\ *\ \theta''\ \ -\ l_1 cos(\theta)\ *\ \theta'^2 = \ddot{x}_A \tag{A.5}$$

$$l_1 cos(\theta)\ *\ \theta''\ -\ l_1 sin(\theta)\ *\ \theta'^2 =\ \ddot{y}_A \tag{A.5}$$

Average of points A and B, for point G:

Position

$$\frac{x_A+x_B}{2} = x_G = (l_1 cos(\theta)\ +\frac{1}{2}l_2 cos(\phi)) \tag{A.6}$$

$$\frac{y_A}{2} = y_G = \frac{1}{2} * l_1 sin(\theta) \tag{A.7}$$

Velocity

$$\frac{\dot{x}_A+\dot{x}_B}{2} = \dot{x}_G \tag{A.8}$$

$$\frac{\dot{y}_A}{2}\ = \dot{y}_G = \frac{1}{2} * l_1 cos(\theta)\ *\ \omega \tag{A.9}$$

Acceleration

$$\frac{\ddot{x}_A+\ddot{x}_B}{2} = \ddot{x}_G \tag{A.10}$$

$$\frac{\ddot{y}_A}{2} =\ \ddot{y}_G = \frac{1}{2}(l_1 cos(\theta)\ *\ \theta''\ -\ l_1 sin(\theta)\ *\ \omega^2) \tag{A.11}$$

θ'' term goes to 0 due to constant velocity, zero acceleration condition given

REFERENCES

[1] Dynamics in Engineering Practice, 11th Ed (only), D.W. Childs & Ap. Conkey, CRC Pubs https://www.crcpress.com/Dynamics-in-Engineering-Practice-Eleventh-Edition/Childs-Conkey/9781482250251

Computational Assignment 3 - Modeling a Mechanical System

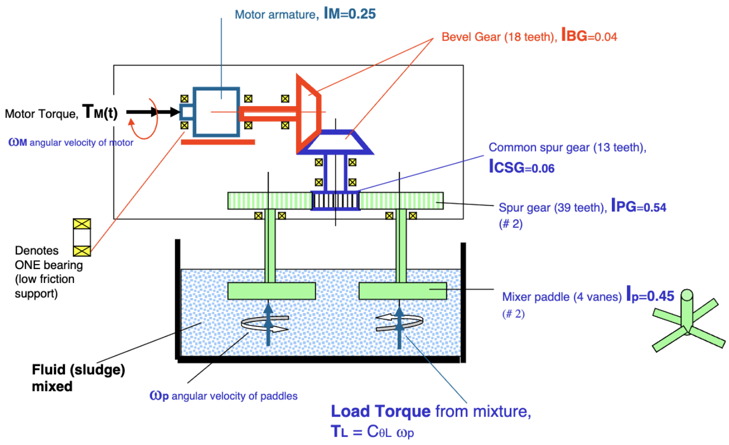
TO: Adolfo Delgado
FROM: Group 7 of MEEN 363 - 502 (Tori Abell, Alois Campbell, Jake Smith, Eddy Silva, Ian Wilhite)
Subject: Assignment 3, Modeling a Mechanical System
Date: 04/28/2025

EXECUTIVE SUMMARY

The objective is to derive equations of motion and find the total energy and dissipative power of the system. The equation of motion can be found through two methods. First, an equation of motion for the system is established by summing forces and moments for the five components and implementing the kinematic constraints of the system. The second method for finding the equation of motion is by analyzing the potential, kinetic, and dissipated energy of the system and using the concept of conservation of mechanical energy. Additionally, given specific damping coefficients and the motor speed, the motor torque and paddle speed are calculated. They will require a minimum of 323 newton meters of torque to drive the system, and will take 3.27 seconds from beginning at rest to reach 99% of operating speed. Due to power losses in the system, such as in the bearings, the efficiency of the system is 84%, and it costs $438 per day, $75 of which was lost due to friction.

METHOD

The mixer system presented had many moving parts. Under further inspection, they can be broken up into five distinct free bodies:


Motor armature, IM=0.25
Bevel Gear (18 teeth), IBG=0.04
Motor Torque, TM(t)
ωM angular velocity of motor
Common spur gear (13 teeth), ICSG=0.06
Spur gear (39 teeth), IPG=0.54 (# 2)
Denotes ONE bearing (low friction support)
Mixer paddle (4 vanes) Ip=0.45 (# 2)
Fluid (sludge) mixed
ωp angular velocity of paddles
Load Torque from mixture, TL = CθL ωp

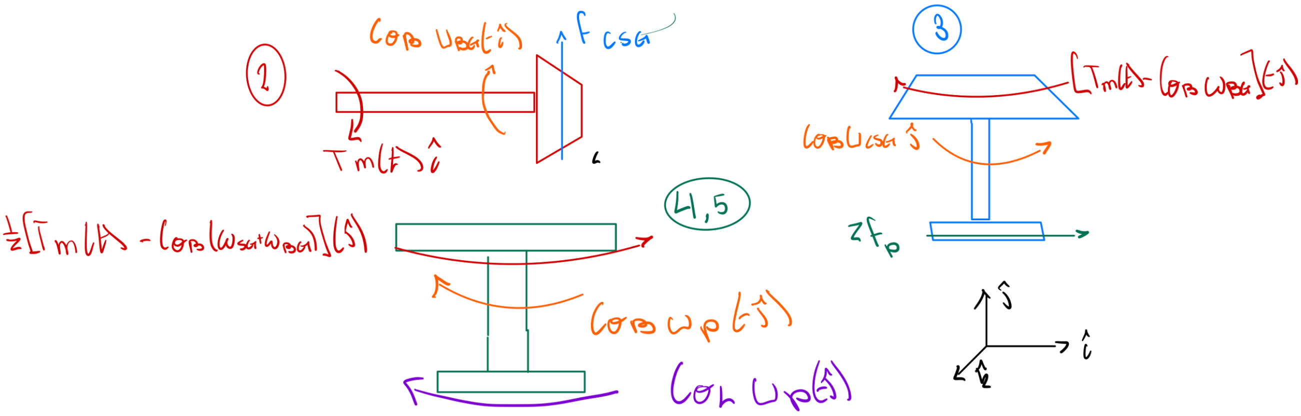


Figure 1: Free Body Diagrams for Components

Using this approach, the system can be described using 5 equations - one for each of the bodies:

$$I_M\theta_1'' = T_M - T_{12} - \theta_1' C_{\theta B} \tag{1}$$

$$I_{BG}\theta_2'' = T_{12} - T_{23} - 2\theta_2' C_{\theta B} \tag{2}$$

$$I_{BG}\theta_3'' = T_{23} - T_{34} - 2\theta_3' C_{\theta B} \tag{3}$$

$$I_{PG}\theta_4'' = T_{34} - T_{45} \tag{4}$$

$$I_P\theta_5'' = T_{45} - T_L = T_{45} - C_{\theta L}\omega_P - \theta_5' C_{\theta B} \tag{5}$$

Furthermore, some kinematic constraints are used to simplify these equations. The defined gear ratios and relationships between the bodies were used to explain how the motor's input torque is converted into output torque at the paddles.

$$\theta_1 = \theta_2 = \theta_3 = \theta_M \tag{6}$$

$$\theta_4 = \theta_5 = \theta_P \tag{7}$$

$$3\theta_M = \theta_P, 3\theta_M' = \theta_P', 3\theta_M'' = \theta_P'' \tag{8}$$

The five motion equations can be simplified to two, where θ_M is the angle of the motor, θ_P is the angle of the paddle, T_{CSG} is the torque of the center spur gear, T_{SG} is the torque of the additional spur gears, and the relationship of the torques are applied:

$$\theta_M''[I_M + 2I_{BG} + I_{CSG}] = T_M - 5\theta_M' C_{\theta B} - T_{CSG} \tag{9}$$

$$T_{CSG} = \frac{13}{39} T_{SG} \tag{10}$$

$$\theta_P''[2(I_{PG} + I_P)] = T_{SG} - 2C_{\theta B}\theta_P' - 2C_{\theta L}\omega_P \tag{11}$$

After this step, the equations are simplified even further, into a single equation.

$$\Rightarrow 9\theta_P''[I_M + 2I_{BG} + I_{CSG}] + \theta_P''[2(I_{PG} + I_P)] = 3T_M - 15\theta_P' C_{\theta B} - 2C_{\theta B}\theta_P' - 2C_{\theta L}\omega_P$$

$$\Rightarrow [9(I_M + 2I_{BG} + I_{CSG}) + 2(I_{PG} + I_P)]\theta_P'' = 3T_M + [-47C_{\theta B} - 2C_{\theta L}]\theta_P'$$

$$\Rightarrow 3T_M = \theta_P''[9(I_M + 2I_{BG} + I_{CSG}) + 2(I_{PG} + I_P)] + [47C_{\theta B} + 2C_{\theta L}]\theta_P' \tag{12}$$

Therefore,

$$I_{eq} = 9(I_M + 2I_{BG} + I_{CSG}) + 2(I_{PG} + I_P) \tag{13}$$

$$C_{eq} = 47C_{\theta B} + 2C_{\theta L} \tag{14}$$

Using the conservation of mechanical energy equation, P_{diss} (dissipative power) and P_{drive} (external drive power) can be used to describe the non-conservative forces in the system (from the damping and driving torque):

$\frac{d}{dt}(E_k + E_p) + P_{diss} = P_{drive}$, where E_k is the kinetic energy and E_p is the potential energy. (15)

$$\frac{d}{dt}E_k = [9(I_M + 2I_{BG} + I_{CSG}) + 2(I_P + I_{PG})]\theta_P'' * \theta_P' \tag{16}$$

$$P_{diss} = C_{\theta B}(\frac{d}{dt}\theta_1)^2 + 2C_{\theta B}(\frac{d}{dt}\theta_2)^2 + 2C_{\theta B}(\frac{d}{dt}\theta_3)^2 + 2C_{\theta B}(\frac{d}{dt}\theta_5)^2 + 2C_{\theta L}(\frac{d}{dt}\theta_5)^2 \tag{17}$$

$$P_{diss} = [47C_{\theta B} + 2C_{\theta L}](\frac{d}{dt}\theta_P)^2 = [47C_{\theta B} + 2C_{\theta L}](\theta_P') * (\theta_P') \tag{18}$$

$$P_{drive} = (T_M) * 3\theta_P' \tag{19}$$

The simplified energy equation is as follows, using (16), (18), and (19):

$$[9(I_M + 2I_{BG} + I_{CSG}) + 2(I_P + I_{PG})]\theta_P'' + [47C_{\theta B} + 2C_{\theta L}](\theta_P') = 3(T_M) \tag{20}$$

Therefore, the same I_{eq} and C_{eq} were determined as in the Newtonian approach:

$$I_{eq} = 9(I_M + 2I_{BG} + I_{CSG}) + 2(I_{PG} + I_P) \tag{13}$$

$$C_{eq} = 47C_{\theta B} + 2C_{\theta L} \tag{14}$$

PROCEDURE

Using the results obtained from the kinematic constraints, free body diagrams, and equations of motion, the mixer can be analyzed under steady state and transient conditions. Using the values that were provided, this is the current state of the system:

- Motor torque T_M for a constant motor speed of 60 Hz (377 rad/s):
  - $T_M * 3 = \theta_P''[9(I_M + 2I_{BG} + I_{CSG}) + 2(I_{PG} + I_P)] + [47C_{\theta B} + 2C_{\theta L}]\theta_P'$
  - $= 0 + [47(0.028) + 2(3.2)](125.66)$
  - $T_M = 323.2$ Nm

- Power from the motor:
  - $P = T * \omega$
  - $P = 323.2 * 377$
  - $P = 121.8$ kW
- Power lost from bearings:
  - $\omega_{CSG} = \omega_M = 377\, rad/s$
  - $\omega_P = \omega_M * \frac{N_{CSG}}{N_{PG}} = 377 * \frac{1}{3} = 126\, rad/s$
  - $P_{diss} = [47C_{\theta B}](\theta_P')^2 = [47(0.028)](126)^2$
  - Total Power Lost, $P = 20.8kW$
- Power from the load:
  - $P_L = T_L * \omega = C_{\theta L} * \omega^2$
  - $P_L = 2 * 3.2 * 126^2$
  - $P_L = 101.1kW$
- Mechanical efficiency:
  - $\eta = \frac{P_L}{P_M} * 100 = \frac{102}{122} * 100$
  - $\eta = 84\%$
- Cost to operate:
  - $C = 0.15 \frac{\$}{kW*h}$
  - $Energy = Power * t = 121.8 * 24 = 2923\, kW * h$
  - $Cost\ Per\ Day = E * C$
  - $Cost\ Per\ Day = \$438$
- Money lost:
  - $Energy\ Lost = Bearing\ Power * t = 20.8 * 24 = 499\, kW * h$
  - $Cost\ Per\ Day = Energy\ Lost * C$
  - $Money\ Lost = \$75$
- System time constant:
  - $I_{eq} = 9(I_M + 2I_{BG} + I_{CSG}) + 2(I_{PG} + I_P)$
  - $\quad = 9(0.25 + 2(0.04) + 0.06) + 2(0.54 + 0.45)$
  - $\quad = 5.49\, kg * m^2$
  - $C_{eq} = 17C_{\theta B} + 2C_{\theta L} = 47(0.028) + 2(3.2) = 7.72 \frac{Nm}{rad/s}$
  - $\tau = \frac{I_{eq}}{C_{eq}} = \frac{5.49}{7.72}$
  - $\tau = 0.711\, s$
- Time Response:
  - $\omega_p = \omega_{steady}(1 - e^{-t/\tau})$
  - $\omega_p = 126(1 - e^{-t/0.711})$
- Time to reach operating speed:
  - $124.74 = 126(1 - e^{-t/0.711})$
    - 124.74 is 99% of the operating speed. This value is used because ln(0) does not exist.

- $0.99 = 1 - e^{-t/0.711}$
- $ln(0.01) =- t/0.711$
- $t = 3.27s$

RESULTS and DISCUSSION

The gear ratios are a significant part of the system and are used to simplify the equations. By applying the gear ratio kinematic constraints and using the Newtonian method, the system is modeled in **Equation 12**. The equations of motion are found by summing forces and torques between the gears, as indicated by the free-body diagrams in **Figure 1**. The system's equivalent inertia and viscous damping coefficient are noted in **Equations 13** and **14**.

Additionally, the energy method is applied to the system to verify the first approach. The two main components in the energy equations are the power lost from the bearings and load, the torque from the motor, and the power due to kinetic energy. The energy method also uses the gear ratio kinematic constraints to simplify the result, and the final equation (**Equation 20**) results in the same equation of motion as the Newtonian method, as expected. The system's equivalent inertia and viscous damping coefficients are equivalent to those calculated using the Newtonian method, which verifies the results of both methods.

Considering the given values, several key quantities from the system are calculated in the procedure, and recorded in **Table 1**. It is important to note that the power in the motor can be calculated from the torque of the system and verified by summing both the dissipated power and the power of the load.

Table 1: Key values from the system.

T_{Motor}	323Nm
P_{Motor}	121.8kW
P_{diss}	20.8kW
P_{Load}	101.1kW
η	84%
$Cost\ Per\ Day$	$438
$Money\ Lost$	$75
τ	0.711s
$Time\ to\ reach\ \ operating\ speed$	3.27s

The money lost is due to the power losses in the system. To reduce the money lost each day, the frictional losses in the bearings should be reduced by implementing a maintenance routine on the system and applying lubricant to ensure smoother bearings. Furthermore, the paddles could be designed to be more efficient. If the losses are reduced, the efficiency of the system would be improved.

$$126(1 - e^{-t/0.724}) \qquad (21)$$

Equation 21 is the equation for the mixer speed as a function of time. **Figure 2** displays a graphical representation of the speed. The time constant, τ, is the value at which the system reaches 63% of its operational speed. **Figure 2** visually confirms the time constant of 0.724 seconds, showing that the speed reaches approximately 63% of its steady-state value. By observation of **Figure 2**, the time at which the system reaches its operating speed is around 3 seconds, validating prior calculations recorded in **Table 1**.

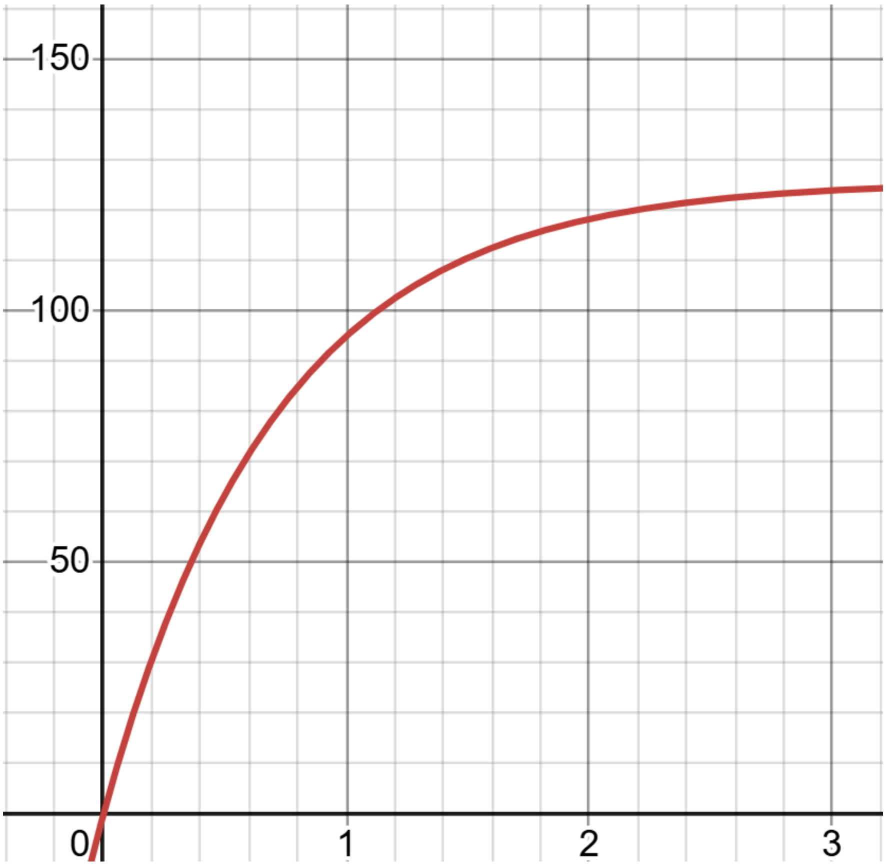


Figure 2. Mixer Speed (rad/s) as a Function of Time (s)

CONCLUSIONS

The mechanical system for the mixer was modeled successfully by applying principles of dynamics and energy conservation. The equivalent inertia and viscous damping parameters were determined consistently through independent approaches. Steady-state performance calculations established the required motor torque, operating speeds, and mechanical power losses within the system. The startup response analysis determined the system time constant and characterized the transient behavior of the paddles. As such, due to its mode of operation, this mixer can probably be classified as a “Paddle industrial mixer” [2]. To further improve its performance, an optimal angle can be set for the viscous fluid being mixed. From certain sources, it can be determined that a 45-degree angle [3] would provide the least amount of resistive torque from the fluid. The modeling effort clearly demonstrates the importance of accounting for mechanical inefficiencies, especially bearing and load damping, to accurately predict system performance. Overall, the work validates the modeling approach and highlights the need for precise dynamic characterization in the design and operation of mechanical systems.

APPENDIX

Detailed steps regarding the derivation of the EOM of the system using the power equation:

From (15) to (16), using the kinematic constraints to simplify:

$$\frac{d}{dt}E_k + 0 = (9I_M\theta_M'' + 2 * 9I_{BG}\theta_M'' + 9I_{CSG}\theta_M'' + 2I_{PG}\theta_P'' + 2I_P\theta_P'') * \theta_P'$$

To determine the left side of (15) in (16):

$$E_k = \frac{1}{2}I_M\theta_1'^2 + \frac{1}{2}I_{BG}\theta_2'^2 + \frac{1}{2}(I_{BG} + I_{CSG})\theta_3'^2 + 2\frac{1}{2}I_{PG}\theta_4'^2 + 2\frac{1}{2}I_P\theta_5'^2$$

$$E_p = 0$$

Using the kinematic constraints to simplify (17) to (18):

$$P_{diss} = C_{\theta B}(\frac{d}{dt}\theta_M)^2 + 2C_{\theta B}(\frac{d}{dt}\theta_M)^2 + 2C_{\theta B}(\frac{d}{dt}\theta_M)^2 + 2C_{\theta B}(\frac{d}{dt}\theta_P)^2 + 2C_{\theta L}(\frac{d}{dt}\theta_P)^2$$

$$P_{diss} = C_{\theta B}9(\frac{d}{dt}\theta_P)^2 + 2C_{\theta B}9(\frac{d}{dt}\theta_P)^2 + 2C_{\theta B}9(\frac{d}{dt}\theta_P)^2 + 2C_{\theta B}(\frac{d}{dt}\theta_P)^2 + 2C_{\theta L}(\frac{d}{dt}\theta_P)^2$$

To find the final EOM using energy in (20) from the previous equations:

$$\frac{d}{dt}(E_k + E_p) + P_{diss} = P_{drive}$$

$$[9(I_M + 2I_{BG} + I_{CSG}) + 2(I_P + I_{PG})]\theta_P'' * (\theta_P') + [47C_{\theta B} + 2C_{\theta L}](\theta_P') * (\theta_P') = (T_M) * \theta_M'$$

$$[9(I_M + 2I_{BG} + I_{CSG}) + 2(I_P + I_{PG})]\theta_P'' * (\theta_P') + [47C_{\theta B} + 2C_{\theta L}](\theta_P') * (\theta_P') = \frac{1}{3}(T_M) * \theta_p'$$

(Notice how the θ_P'/ω values cancel out to get a familiar equation of motion.)

REFERENCES

[1] Dynamics in Engineering Practice, 11th Ed (only), D.W. Childs & Ap. Conkey, CRC Pubs https://www.crcpress.com/Dynamics-in-Engineering-Practice-Eleventh-Edition/Childs-Conkey/9781482250251

[2] Anderson Process. 2022. "It's in the Mix: Industrial Mixers Defined, Types and Applications." Anderson Process. December 15. https://www.andersonprocess.com/its-in-the-mix-industrial-mixers-defined-types-and-applications/.

[3] Liu, Y., and Zhang, X. 2024. "Design and Analysis of a Novel Hybrid Energy Harvester for Smart Wearable Devices." Advances in Mechanical Engineering. https://doi.org/10.1177/16878132241269238.

Back to Contents

3. Numerical Methods (4th Place in Final Competition)

Summary

This section includes four project phases from the Numerical Methods course (MEEN 357), in which we placed 4th of 20 teams.

- Phase 1: Foundations & Subfunctions
- Phase 2: System Verification & Analysis
- Phase 3: Parameter Validation & Strategic Approach
- Phase 4: Final Competition Submission (Document)

Contributors: Jacob Hargreaves, Ian Wilhite, David Guess

Key Skills: Numerical Methods, Computational Analysis, Algorithm Development, Software Development, and Convex Optimization.

Relevance: This work demonstrates the development and management of large codebases, iterative verification and validation, and system design struture.

MEEN 357 - 501
Authors: Jacob Hargreaves, David Guess, Ian Wilhite

6.1 Coding

Question 1: We had you define the acceleration due to gravity as a field in a structure that you had to pass as an input argument to several functions. Instead, we could have had you type the value for the constant, 3.72 m/s2 , directly in those functions. Do you believe there is an advantage to how we had you do it? Explain. Would you have done it differently? Explain why or why not.

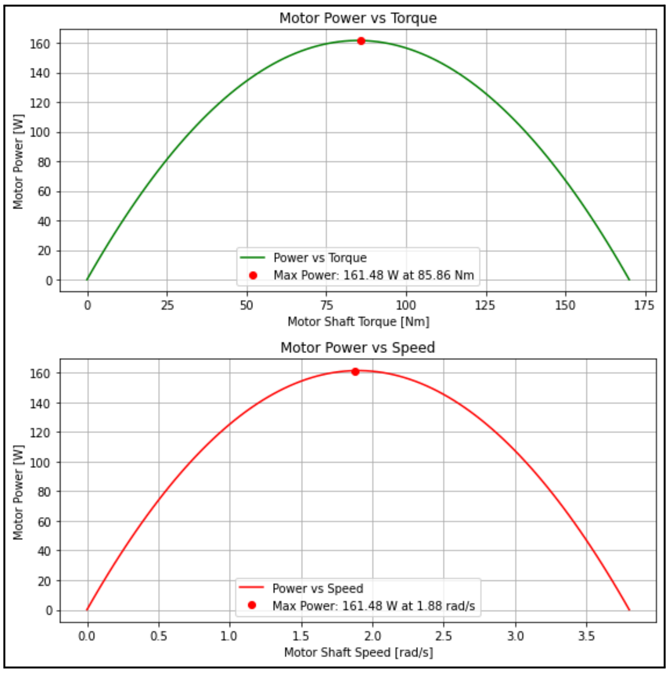
We believe that there are advantages to the way we define acceleration in this project. Because we did not code the gravity constant to simply be the value on Mars, we can update the gravity constant easily if we want to analyze the rover on other planets or moons. If we wanted to change the gravity constant, we could simply alter its structure in one place rather than having to change its value in every single function. It is for these reasons that we would not have done anything differently regarding the acceleration due to gravity.

Question 2: What happens if you try to call F_gravity using a terrain slope of 110 degrees? Is this desirable behavior? Explain why you think this.

When we try to call F_gravity using a terrain slope above 90 degrees, such as a slope of 110 degrees, F_gravity returns a force greater than the force of gravity in free fall. This is because the equation used in this function is $F_{gravity} = - m_{rover} * g * sin(\alpha)$ and when α (the angle of inclination) is above 90 degrees, the sin() value is greater than 1. This is not desirable behavior because the rover would never feel a force of gravity greater than the free fall force.

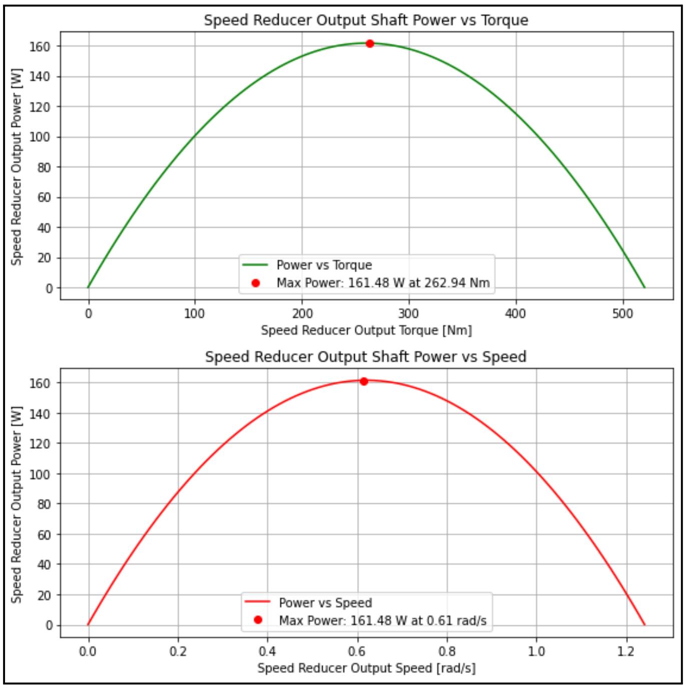
6.2 Motor and Speed Reducer Behavior

Question 3: What is the maximum power output by a single rover motor? At what motor shaft speed does this occur? Provide graphs or other data to support your answer.



As shown in the graph above, the maximum power output by a single rover motor is about 161.48 Watts and occurs at a motor shaft speed of 1.88 rad/s.

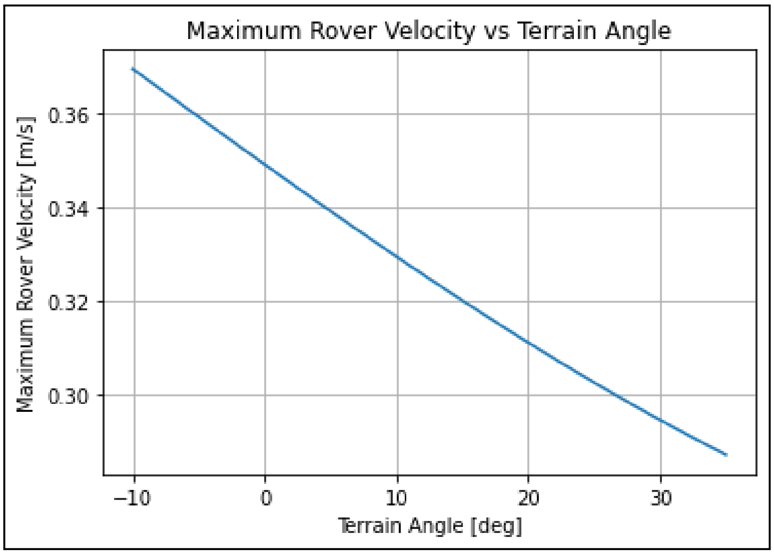
Question 4: What impact does the speed reducer have on the power output of the drive system? Again, provide any graphs or supporting data.



The speed reducer causes the maximum power output of the drive system to occur at a much lower motor shaft speed of 0.61 rad/s, as seen above. The speed reducer does this in order to help the rover overcome resistance such as rough terrain. The rover can better overcome this resistance as it has more torque at a lower speed. The speed reducer allows for more torque overall as compared to the figures in question #3.

6.3 Rover Behavior

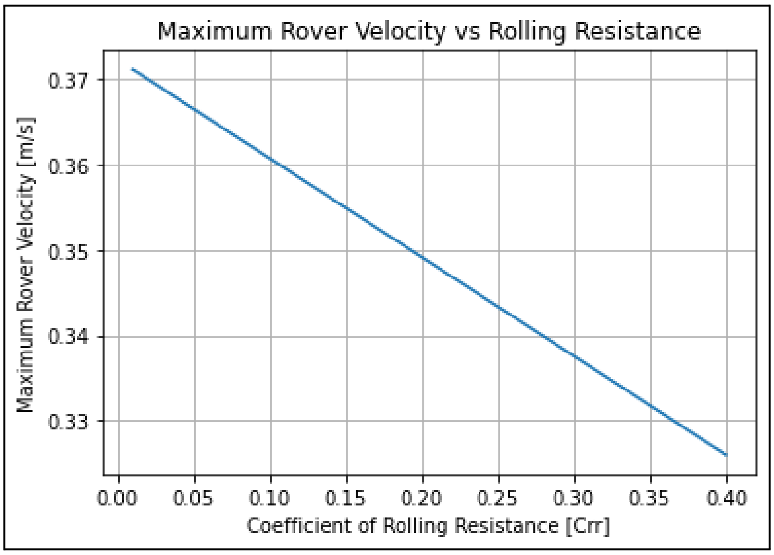
Question 5: Examine the graph you generated using analysis_terrain_slope.py. (Provide the graph in your response for reference.) Explain the trend you observe. Does it make sense physically? Why or why not? Please be precise. For example, if the graph appears linear or non-linear, can you explain why it should be the way you observed? Refer back to the rover model and how slope impacts rover behavior.



This trend from the graph above makes perfect sense. On downhill motion (Negative Terrain Angle) the rover experiences additional acceleration from gravity. Therefore, a higher maximum velocity is obtained. On flat and uphill motion (Positive Terrain Angle), the maximum velocity of the rover steadily decreases. This is because the rover has to overcome both gravity and rolling resistance pulling it backward.

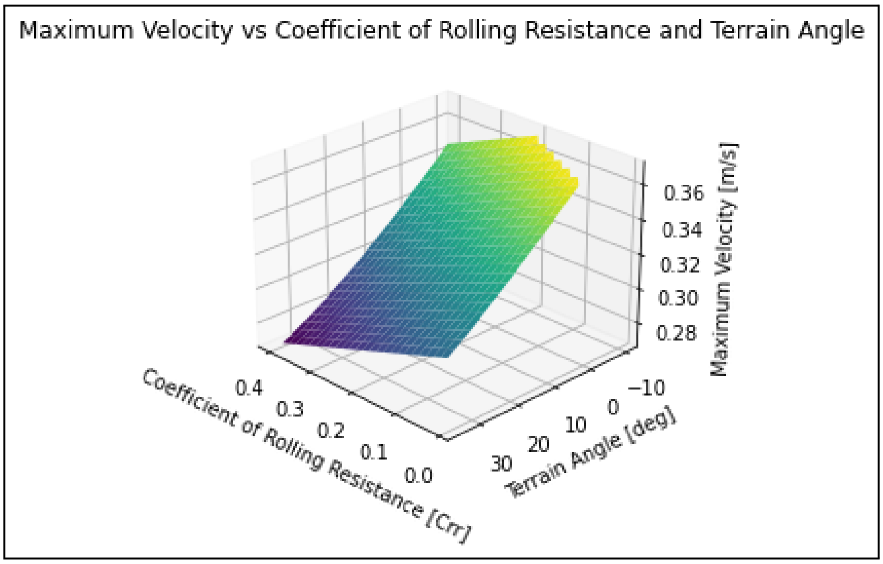
Additionally, the graph appears approximately linear, which suggests that the impact of slope on the rover's speed is proportional, which aligns with the rover model where gravitational forces and resistive forces change linearly with increasing slope.

Question 6: Examine the graph you generated using analysis_rolling_resistance.py. (Provide the graph in your response for reference.) Explain the trend you observe. Does it make sense physically? Why or why not? Please be precise. For example, if the graph appears linear or non-linear, can you explain why it should be the way you observed? Refer back to the rover model and how the coefficient of rolling resistance impacts rover behavior.



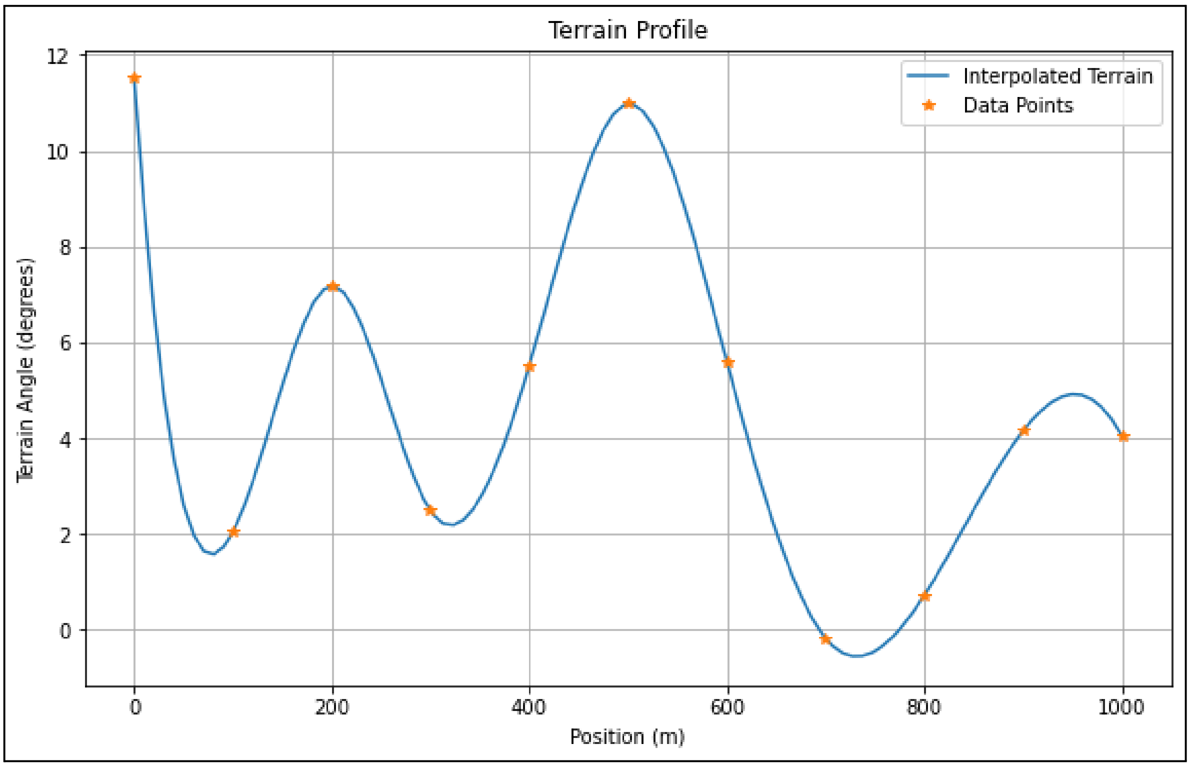
This graph shows a linear decrease in maximum rover velocity as the coefficient of rolling resistance (Crr) increases. This makes sense because rolling resistance is a force that opposes the rover's motion. As Crr increases, the motor has to exert more energy to overcome this resistance. Since the rolling resistance is directly proportional to Crr, the decrease in velocity follows a linear trend. This means that as Crr rises, the rover's maximum achievable speed steadily decreases. This is also consistent with the energy required to counteract the increasing resistance.

Question 7: Examine the surface plot you generated using analysis_combined_terrain.py. (Provide the graph in your response for reference.) What does this graph tell you about the physical conditions under which it is appropriate to operate the rover? Based on what you observe, which factor, terrain slope or coefficient of rolling resistance, is the dominant consideration in how fast the rover can travel? Please explain your reasoning.



The 3D-surface plot shows that both the terrain slope and the coefficient of rolling resistance (Crr) have a significant impact on the maximum velocity of the rover. As the terrain slope increases, the rover's maximum velocity decreases, and as the Crr increases, the velocity also decreases. However, the graph shows that terrain slope has a more noticeable effect on the rover's speed compared to the rolling resistance. This is evident from the steep gradient along the terrain angle axis, meaning that changes in slope result in larger variations in speed. Therefore, terrain angle is the dominant factor. This is because steeper angles significantly reduce the rover's maximum velocity while rolling resistance primarily adds resistance.

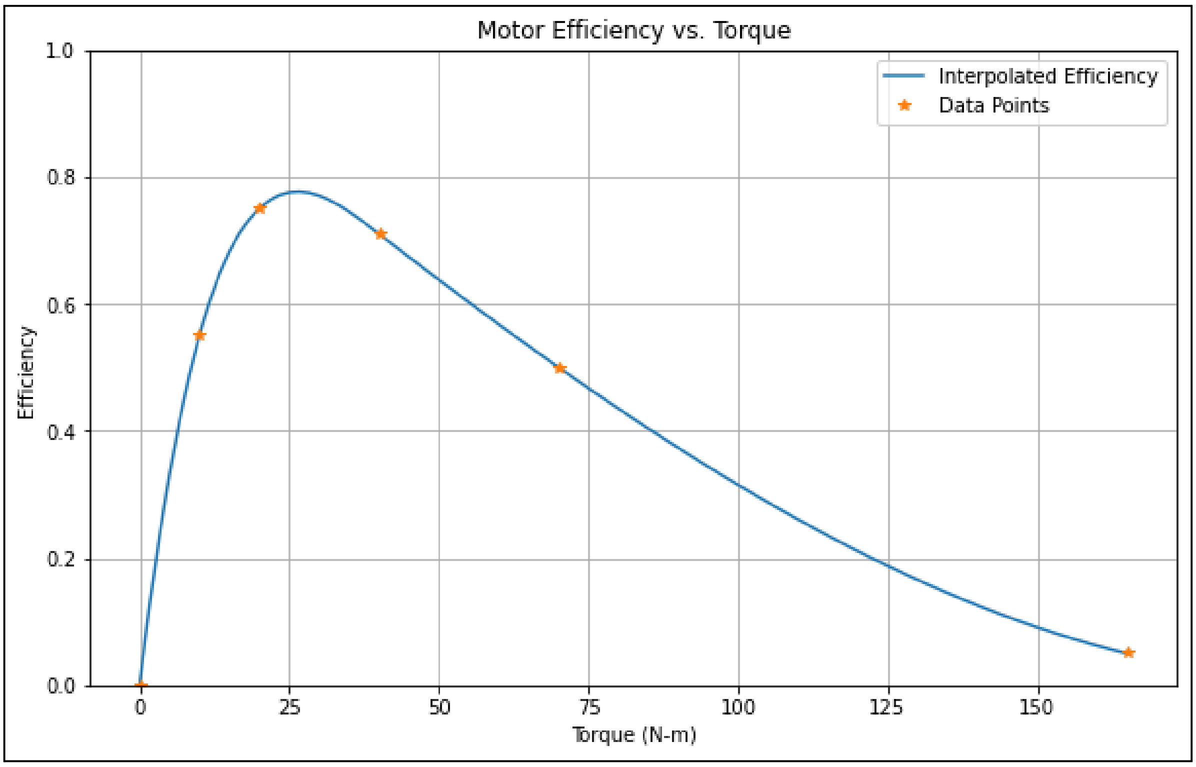
Task 2:



Experiment_Visualization Curve

The terrain profile generated through cubic spline interpolation exhibits smooth characteristics throughout the entire path. The graph shows that the blue interpolated line transitions naturally between the data points (marked by orange stars) without any jarring changes or breaks. This smoothness is from using cubic polynomials between points, which ensures continuous slopes and curvatures at the transitions. Additionally, the curve maintains accuracy to our input data, as evidenced by the interpolated line passing exactly through each orange star marker. This smooth behavior makes sense for modeling real-world terrain since actual landscapes typically don't have abrupt elevation changes.

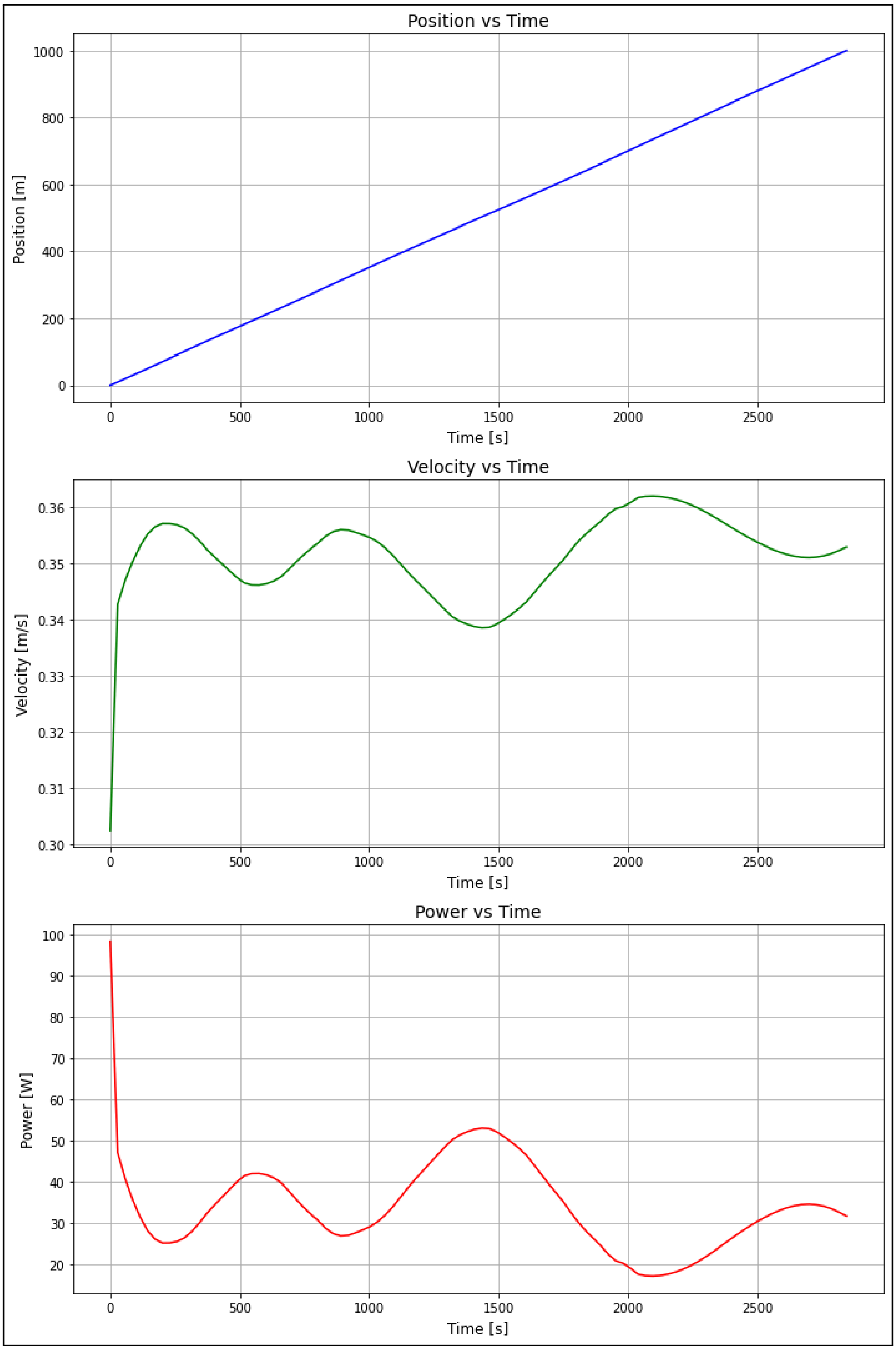
Task 5:



Efficiency_Visualization Curve

This motor efficiency curve shares a similar overall shape with Figure 2 from the handout - starting at zero, rising to a peak, then declining - though it operates at a much higher torque range. The curve is smooth throughout due to the cubic spline interpolation used to connect the data points. The interpolated line passes exactly through all six data points (marked with stars), which is a key property of cubic spline interpolation.

Task 8:


Position vs Time
Position [m]
1000
800
600
400
200
0
0
500
1000
1500
2000
2500
Time [s]
Velocity vs Time
Velocity [m/s]
0.36
0.35
0.34
0.33
0.32
0.31
0.30
0
500
1000
1500
2000
2500
Time [s]
Power vs Time
Power [W]
100
90
80
70
60
50
40
30
20
0
500
1000
1500
2000
2500
Time [s]

(a) Position vs Time: The position-time graph shows a slightly non-linear relationship. While there is an overall upward trend, the slope (which represents velocity) varies over time rather than being constant. This nonlinearity occurs because the rover's velocity changes as it encounters different terrain angles, requiring different amounts of power and resulting in varying speeds across the trajectory.

(b) Velocity-Power Relationship: Looking at the velocity and power plots, there's a clear correlation - when the rover requires more power (shown by spikes in the power graph), there are corresponding changes in velocity. The relationship appears inverse in many cases - when the rover encounters situations requiring more power (likely uphill sections), the velocity tends to decrease slightly. This makes physical sense as the rover needs to expend more energy to maintain motion against gravity on inclines, resulting in slightly reduced velocity.

(c) Velocity Profile and Terrain: The velocity shows periodic variations between approximately 0.34-0.36 m/s, which likely corresponds to changes in terrain elevation. The dips in velocity (around t=1500s) suggest encounters with uphill sections where the rover must work harder against gravity, while slight increases in velocity may correspond to downhill or level sections where less power is required to maintain motion.

(d) Velocity Smoothness: The velocity graph shows smooth transitions rather than abrupt changes. This smoothness is due to two factors:
1. The terrain profile was interpolated using cubic splines, creating smooth transitions between terrain points
2. The rover's inertia and motor characteristics prevent instantaneous velocity changes, leading to gradual transitions as terrain conditions change

Rover Telemetry Data

Variable	**Value**
completion_time	2842.92 s
distance_traveled	1000 m
max_velocity	0.36 m/s
average_velocity	0.35 m/s
battery_energy	1094725.05 J
batt_energy_per_distance	1094.725 J/m

Task 9:

During Experiment 1, the battery energy used is 1,094.725 kJ. This is greater than the energy provided by the Lithium Iron Phosphate battery pack, as this batter only provides 907.2 kJ. Therefore, no, the rover can not complete the case defined by Experiment 1 with the 0.9072e6 J Lithium Iron Phosphate battery pack. There is not sufficient energy capacity to handle the demands shown in the simulation. We arrived at this conclusion by analyzing the output of Experiment 1.

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/m^3); 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.

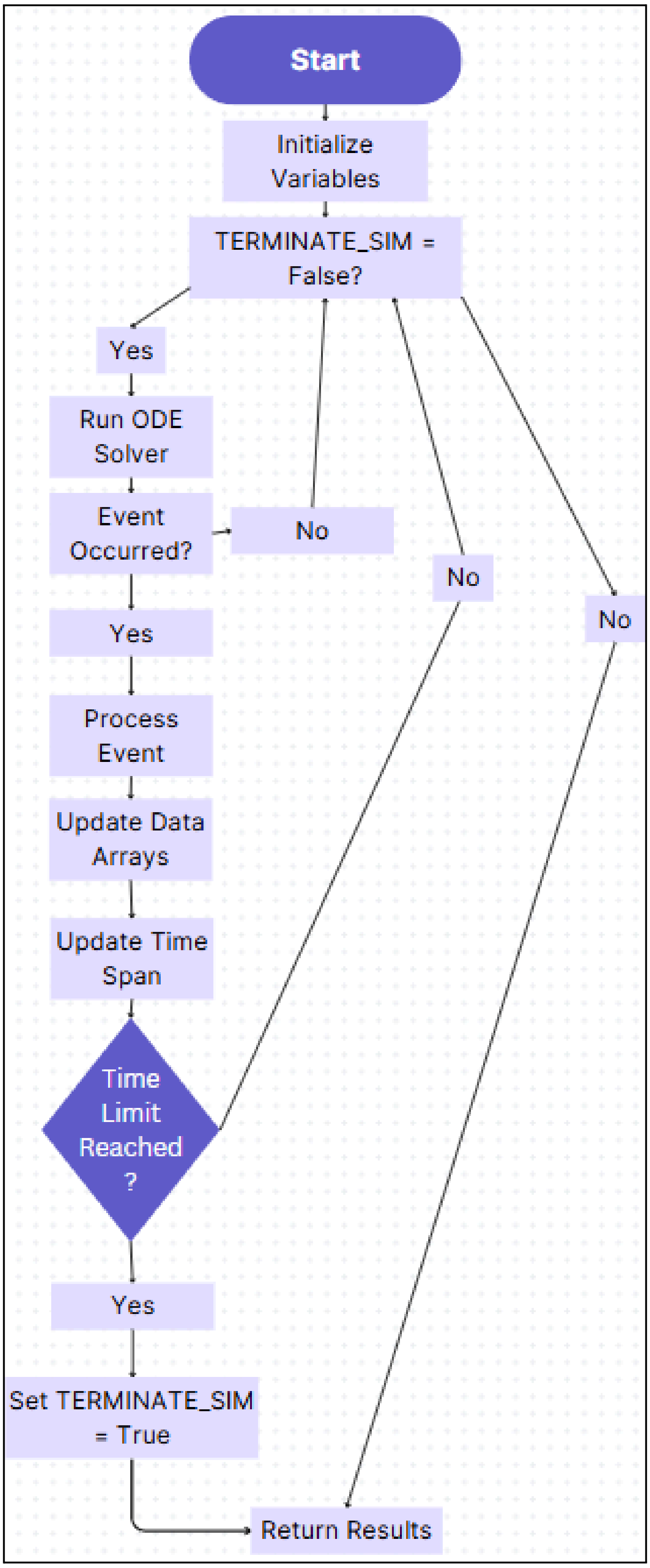
Key events handled by update_edl_state include:
1. Heat Shield Ejection: Heat shield is detached when a certain altitude is reached.
2. Parachute Ejection: Parachute is released based on altitude criteria.
3. Rockets Activation: Rockets are ignited for descent control at specific altitude.
4. Sky Crane 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

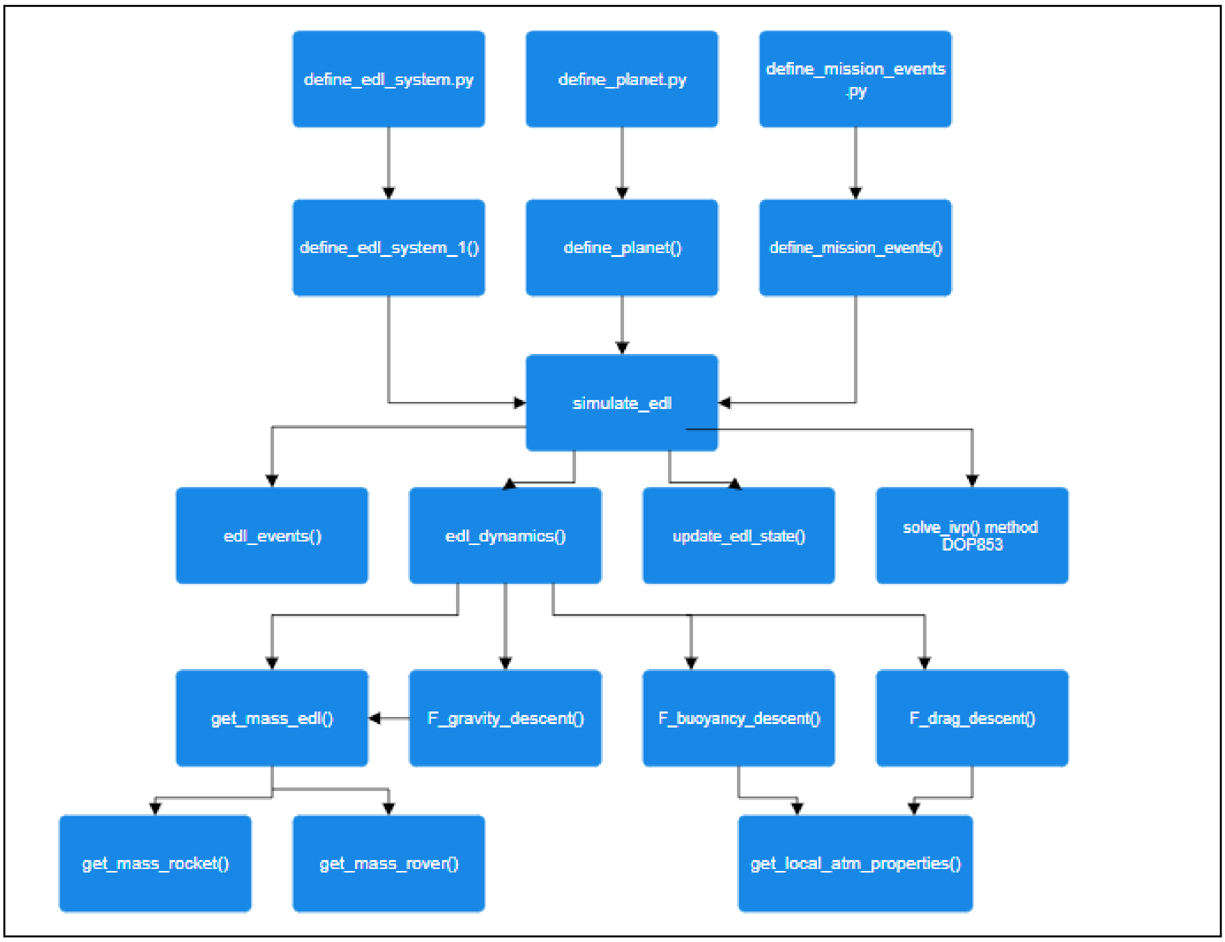
The while loop in simulate_edl repeats until TERMINATE_SIM is set to True. Termination occurs when time runs out, the rover safely lands, fuel depletes, or a crash is detected.

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.

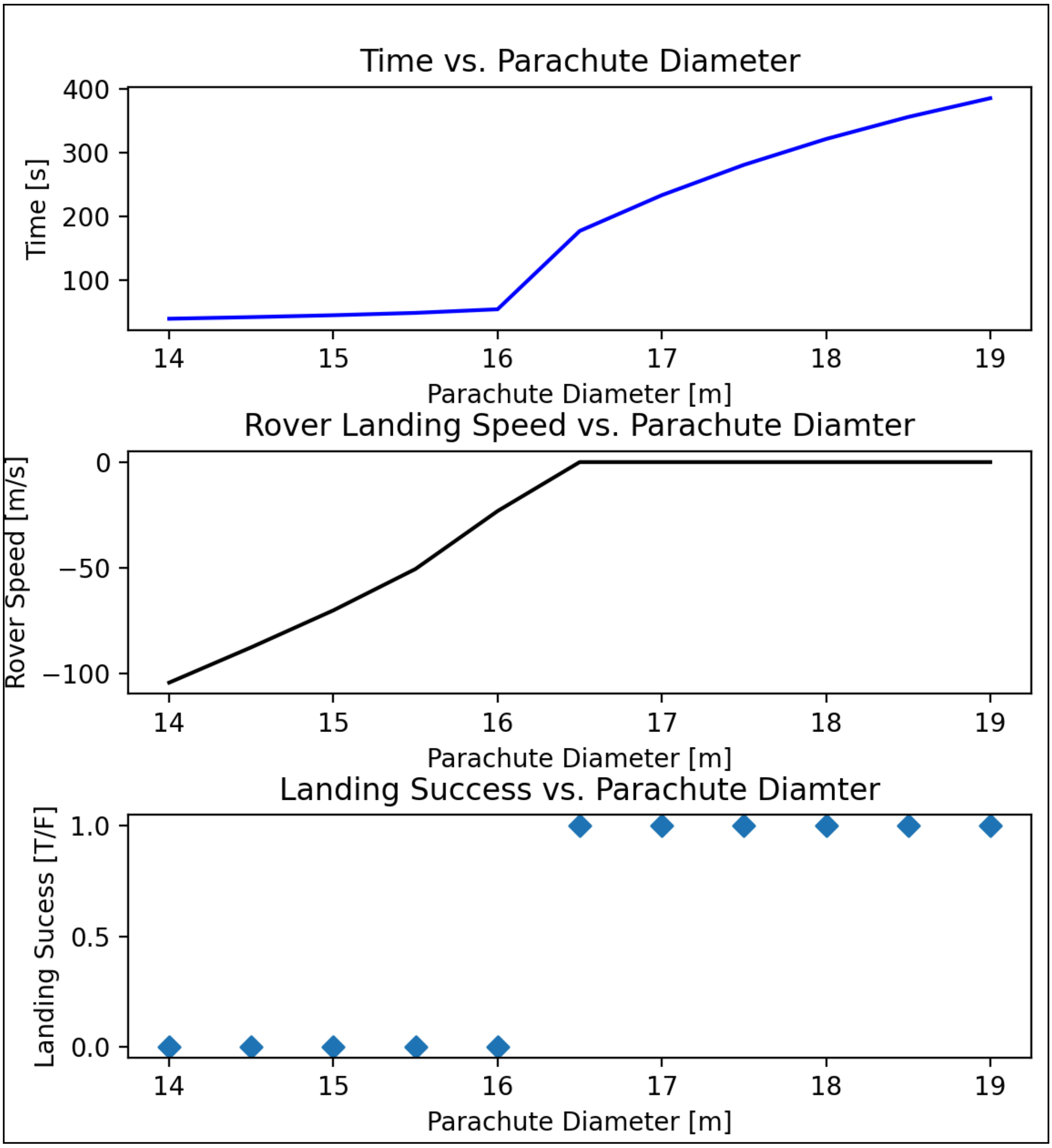

Start
Initialize Variables
TERMINATE_SIM = False?
Yes
Run ODE Solver
Event Occurred?
No
No
No
Yes
Process Event
Update Data Arrays
Update Time Span
Time Limit Reached ?
Yes
Set TERMINATE_SIM = True
Return Results

Task 4:



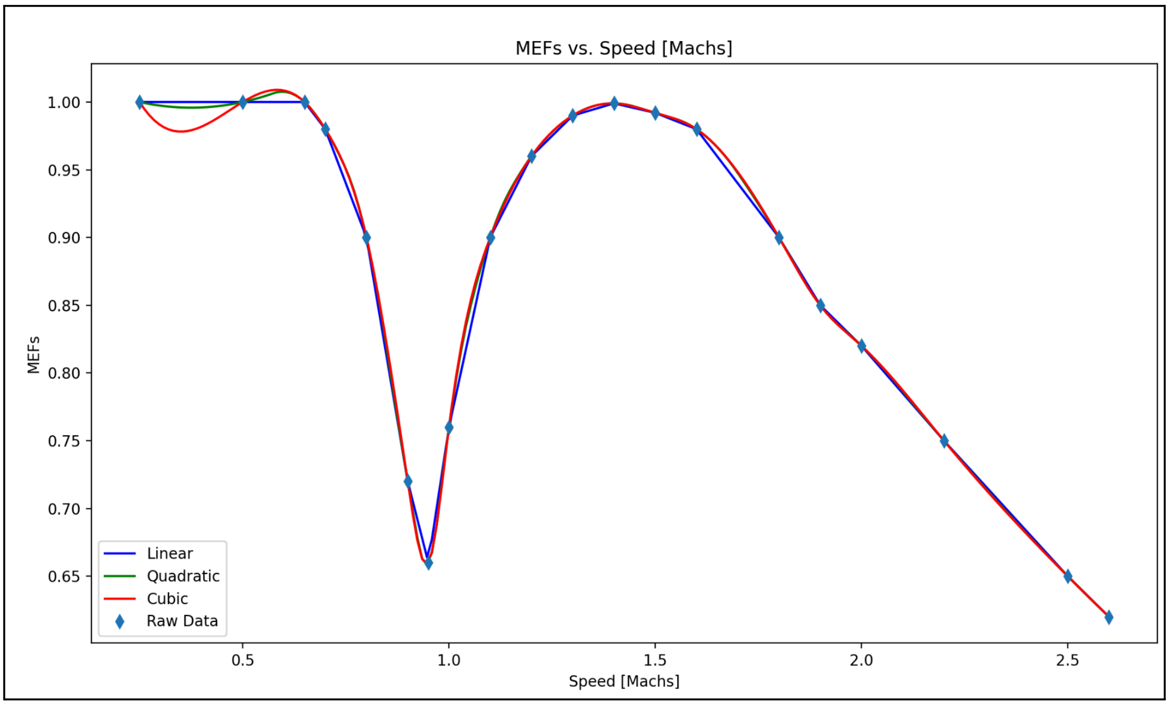
This flowchart begins with the imported files, followed by the loading of key dictionaries. These dictionaries help establish variables that are then passed into simulate_edl. Inside simulate_edl, we track each function call: the flow splits to cover edl_events, update_edl_state, and the solve_ivp function. 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.

PROJECT PHASE 4:

Optimization Challenge Report

Prepared for

MEEN 357 - Section 501

Team #10

By

Jacob Hargreaves

Ian Wilhite

David Guess

November 29, 2024

Aggie Honor Code: *“On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work."*

“None of the members of the team communicated in any way with other teams with regards to specific coding issues with this project.”

Signature:	*Jacob Hargreaves*	Date:	*11/29/2024*
	David Guess	Date:	*11/29/2024*
	Ian Wilhite	Date:	*11/29/2024*

Table of Contents

Introduction

The design of an Entry, Descent, and Landing (EDL) system and a rover for a Martian mission is a challenging problem that requires careful consideration of multiple factors.
The goal is to create a system that can safely land on Mars and allow the rover to travel at least 1 kilometer on a single battery charge. This involves optimizing key parameters like landing velocity, rover speed, and energy usage while staying under the $7.2 million budget. To meet these requirements, we had to choose the best combination of components, such as parachutes, motors, batteries, and chassis materials, while making trade-offs between cost, performance, and weight. This project highlights the complexity of designing systems for real-world space exploration missions.

Formulation of the Design Problem

We aim to design an Entry, Descent, and Landing (EDL) system coupled with a rover capable of traversing at least 1 km on a single battery charge over a predefined Martian terrain. The design must balance key performance metrics: speed, cost, and battery efficiency, while adhering to specific constraints. Among these, speed is prioritized, followed by cost considerations. The design solution should:

- **Maximize rover speed** to reduce the total mission duration.
- **Minimize total cost**, ensuring it stays within the $7.2 million budget.
- Optimize battery capacity to achieve sufficient energy per meter (J/m) without excessive weight or cost.

Ensure the system satisfies constraints such as:

- Landing velocity below 1 m/s.
- Rover chassis strength exceeding 40,000 units.
- Mass and geometry limits for all components.

Key decision variables include:

- Parachute diameter, fuel mass, wheel radius, and gear diameter (continuous variables).
- Motor type, battery type, number of modules, and chassis material (discrete variables).

Considered Design Solutions

To address the rover's design requirements, we examined multiple options for key components and materials, considering performance and cost.

1) **Motor Selection**: The six available motor types were evaluated, focusing on their efficiency, torque, and speed capabilities. High-efficiency motors offered a balance of performance and reasonable cost increases. Both torque_he and speed_he motors stood

out as superior options, with the speed_he motor chosen for its ability to maximize rover speed, a primary objective.

2) **Chassis Material**: The three chassis material options—steel, magnesium, and carbon fiber—were compared for cost, strength, and weight. While carbon fiber exceeded budget constraints due to its high cost, magnesium emerged as the optimal choice. Its high specific strength allowed for a lighter chassis, aligning with the weight limits, while maintaining structural integrity and meeting strength requirements.
3) **Battery Configuration**: Various battery types and configurations were analyzed for energy efficiency and cost. PbAcid-1 batteries provided a good energy-to-cost ratio, making them a favorable option. Initial testing started with 10 modules to evaluate performance before editing the configuration.

Final Design Selection

We developed a Python program called value_sweep.py designed to comprehensively analyze the EDL process by systematically compiling and evaluating constraint values across multiple iterations. Our approach involved collecting data for each constraint, sorting iterations by time, and strategically eliminating scenarios that did not work.

Unlike traditional methods that focus on optimizing a single parameter, our approach allowed for simultaneous multi-constraint analysis. This methodology provided a more holistic and nuanced optimization strategy, enabling us to maximize overall performance in ways that would have been impossible with a narrow, single-constraint approach. By examining the complex interplay between different constraints, we gained deeper insights into the system's potential.

The effectiveness of this strategy was evident in the progressive evolution of our value_sweep.py outputs. The data range from our initial analysis to the final iterations demonstrated significant improvements in our understanding and optimization of the EDL process, validating the power of our comprehensive analytical approach.

We used this approach to optimize our design and choose the best one, given the constraints and the parameters we wanted to emphasize.

Numerical Methods Used

Our Python script employs several numerical optimization methods for maximizing objectives and solving constraints. These methods are from the scipy.optimize library and include:

1. **Trust-Region Constrained Algorithm (trust-constr)**:

    - This method solves the optimization problem with constraints and bounds defined using a Nonlinear Constraint object. It iteratively updates the solution while respecting constraints like strength, cost, and battery energy limits.
2. **Sequential Least Squares Programming (SLSQP)**:
    - This method minimizes the objective function with equality and inequality constraints defined via the ineq_cons dictionary. It is also bound-aware.
3. **Differential Evolution**:
    - A global optimization algorithm that explores the search space more thoroughly compared to local optimizers. It handles constraints through a nonlinear constraint and allows population-based exploration.
4. **Constrained Optimization BY Linear Approximations (COBYLA)**:
    - It deals with constraints by linear approximations and can handle inequality constraints without requiring explicit gradient definitions.

Additionally, grid search is employed in the loops to systematically explore the parameter space before optimization, providing initial guesses (x0) for the optimization algorithms. This enhances the likelihood of finding a global maximum.

Description of Final Design

We arrived at our final design by optimizing key aspects of our rover in order to minimize the time while staying within the parameters. The following are key parameters found when designing our rover:

- **Parachute:** We found that minimizing the parachute diameter to 14 m allowed our rover to descend the quickest.
- **Wheel Radius:** Our wheel radius is maximized at almost 0.7 m in order to allow the rover to move the furthest that it can in the least amount of rotations.
- **Speed Reducer (d2):** Additionally, the speed reducer is the smallest that it can be at 0.05 m in order to maximize the number of rotations that the wheel has.
- **Fuel and Chassis Mass:** Our fuel mass and chassis mass are both a bit on the heavier side but light enough to allow the rover to land safely and quickly.

We found that the budget constraint was not one of the key factors to create tough decisions in our design. This is shown in our total cost as we are about $1.25 million under the $7 million budget.

Table 1: Final Optimized Rover Parameters

Parameter	Value	Units
Optimized Parachute Diameter	14.00	m
Optimized Fuel Mass	200.304121	kg
Time to complete EDL Mission	128.786949	s
Rover Velocity at Landing	-0.100512	m/s
Optimized Wheel Radius	0.699995	m
Optimized d2	0.05	m
Optimized Chassis Mass	375.299577	kg
Time to Complete Rover Mission	395.625	s
Time to Complete Mission	524.412	s
Average Velocity	2.436	m/s
Distance Traveled	1000.00	m
Battery Energy per Meter	541.524	J/m
Total Cost	5751017.33	$

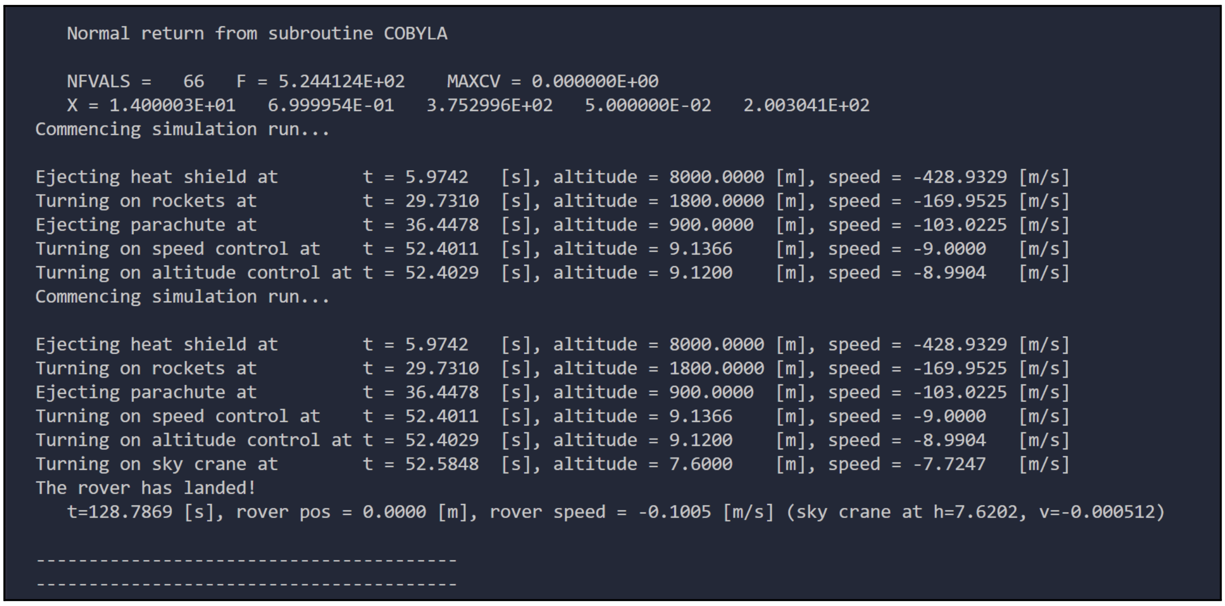
Performance Data

The final rover design, optimized by using the COBYLA method, meets all mission requirements while achieving high performance in the Entry, Decent, Landing, and traversal phases. Below is a detailed summary of the performance results.

- **Motor**: The high-efficiency speed_he motor was selected to maximize the rover's speed without significantly increasing energy consumption.
- **Chassis Material**: Magnesium was chosen for its lightweight and high strength-to-weight ratio. The cost was also a very big consideration for this choice
- **Battery**: PbAcid-1 batteries were selected for their reliability and cost efficiency. A configuration of 5 battery modules was used to achieve the necessary energy storage.

Table 2: Chosen Rover Specifications

Parameter	Chosen Specification
Motor	*speed_he*
Chassis Material	Magnesium
Battery	PbAcid-1
Number of battery Modules	5
Optimization Method	COBYLA



```
   Normal return from subroutine COBYLA

   NFVALS =   66   F = 5.244124E+02    MAXCV = 0.000000E+00
   X = 1.400003E+01   6.999954E-01   3.752996E+02   5.000000E-02   2.003041E+02
Commencing simulation run...

Ejecting heat shield at        t = 5.9742   [s], altitude = 8000.0000 [m], speed = -428.9329 [m/s]
Turning on rockets at          t = 29.7310  [s], altitude = 1800.0000 [m], speed = -169.9525 [m/s]
Ejecting parachute at          t = 36.4478  [s], altitude = 900.0000  [m], speed = -103.0225 [m/s]
Turning on speed control at    t = 52.4011  [s], altitude = 9.1366    [m], speed = -9.0000   [m/s]
Turning on altitude control at t = 52.4029  [s], altitude = 9.1200    [m], speed = -8.9904   [m/s]
Commencing simulation run...

Ejecting heat shield at        t = 5.9742   [s], altitude = 8000.0000 [m], speed = -428.9329 [m/s]
Turning on rockets at          t = 29.7310  [s], altitude = 1800.0000 [m], speed = -169.9525 [m/s]
Ejecting parachute at          t = 36.4478  [s], altitude = 900.0000  [m], speed = -103.0225 [m/s]
Turning on speed control at    t = 52.4011  [s], altitude = 9.1366    [m], speed = -9.0000   [m/s]
Turning on altitude control at t = 52.4029  [s], altitude = 9.1200    [m], speed = -8.9904   [m/s]
Turning on sky crane at        t = 52.5848  [s], altitude = 7.6000    [m], speed = -7.7247   [m/s]
The rover has landed!
   t=128.7869 [s], rover pos = 0.0000 [m], rover speed = -0.1005 [m/s] (sky crane at h=7.6202, v=-0.000512)

----------------------------------------
----------------------------------------
```

Figure 1: Rover Mission Timesheet

These results demonstrate the efficacy of the chosen design approach and optimization method. The design success highlights its capability to meet the heavy constraints of the Mars Mission.

Back to Contents

4. SEC Ignite Competition (3rd Place)

Summary

Our team competed in the Students Engineering Council's Ignite competiton Aerospace-Mechanical Track to develop a simulation of a landing rocket, where my team achieved 3rd place of 50+ teams. I worked to develop our control system simulation and organize much of the team collaberation.

Contributors: Eduardo Burciaga-Ichikawa, Kalen Jaroszewski, Julia Sopala, and Ian Wilhite

Key Skills: Computational Fluid Dynamics (CFD), Finite Element Analysis (FEA), Model Predictive Control (MPC), Numerical Methods, Simulation, Technical Writing, Team Collaboration, Project Management.

Relevance: This project showcases the ability to apply advanced engineering principles in a competitive environment, demonstrating skills in simulation, control system design, and technical communication.

Final Report:

Development of a Rocket Landing Simulation Using a Gimbal and Landing Gear

Eduardo Burciaga-Ichikawa, Kalen Jaroszewski, Julia Sopala and Ian Wilhite

Ignite Design Challenge

November 23, 2024

Table of Contents

Introduction

This paper explores the development of a rocket landing simulation that aims to enhance the landing efficiency and stability of the rocket landing vertically. Designing efficient and reliable landing systems for rockets remains a crucial challenge to safely and consistently land a rocket. The rocket explored in this paper has been modified to reduce mission costs and play a significant role in advancing reusable space technology. The design implemented gimbals for thrust vector control and robust landing gear systems, along with a redesign of the nose cone to realistically model the rocket that would go through both launching and landing. The design leverages gimbal systems to dynamically adjust thrust direction, ensuring precise control of the rocket's orientation during descent. Additionally, including landing gear improves stability upon touchdown, mitigating the risk of tip-over or damage. This solution addresses two fundamental challenges in rocket landings: maintaining attitude control during descent and ensuring structural integrity upon contact with the ground. By tackling these issues, the design enhances the reliability and efficiency of rocket recovery systems, paving the way for advancements in reusable rocket technology.

Problem

Background

The challenge of landing a rocket upright is both a technical and operational problem central to modern aerospace engineering. Traditional expendable launch systems discard hardware after a single use, driving up costs and limiting the efficiency of space exploration and transportation. In contrast, successful vertical landings of reusable rockets can drastically reduce costs and environmental impact. In industry, notable companies like SpaceX and Blue Origin have pioneered reusable rocket technology by implementing advanced systems such as thrust vector control (TVC) and deployable landing gear **[1]**. These technologies enable rockets to correct their attitude during descent and safely absorb impact upon touchdown. Thrust vector control, often achieved through gimbaled engines, allows for fine-tuned adjustments to the rocket's trajectory. Additionally, landing gear equipped with damping mechanisms ensures stability during landing, even on uneven surfaces.

Research highlights the complexities of balancing stability, accuracy, and robustness. For example, Reuben Ferrante illustrates how gimbal systems contribute to minimizing drift during descent while investigating computational methods for simulating controlled landings **[1]**. A second project that influenced the use of gimbals and a redesign of the nose cone is a launch report from the AIAA Regional Student Conferences which tested the launch of a rocket by replacing the fins with a gimbal **[2]**. These advancements inspire our approach to integrating gimbals and landing gear into a simulation that accurately models the dynamics of a rocket landing while implementing a better thrust-to-weight ratio **[3]**.

Purpose

Landing a rocket upright is a valuable problem to solve due to its implications for cost efficiency, safety, and the feasibility of sustainable space exploration. The difficulty lies in maintaining stability during descent and ensuring a secure touchdown on varying terrains. Without reliable control mechanisms, rockets risk instability, which can lead to mission failure, loss of valuable hardware, and environmental hazards.

This problem affects industries and organizations striving to make space travel more accessible. Reusable rockets can significantly reduce the cost of payload delivery, increasing accessibility to orbital applications for research institutions, private companies, and governments.

Design Criteria

To address the problem quantitatively, our team established the following design criteria:

1. **Stability During Descent:** The rocket must maintain an upright orientation with minimal angular drift, measured as the deviation from vertical (in degrees).

2. **Landing Controls:** The code for the rocket simulation must result in a zero velocity when altitude approaches zero, while maintaining stability to disturbances that can occur during landing.
3. **Impact Absorption:** The landing gear should effectively dampen forces upon touchdown to prevent structural damage, quantified by measuring landing impact forces (in Newtons).
4. **Minimized Mass:** No amount of material should be carried needlessly. All parts should be designed to perform their purpose with the minimum amount of additional material, and of the lightest possible material.

Solution

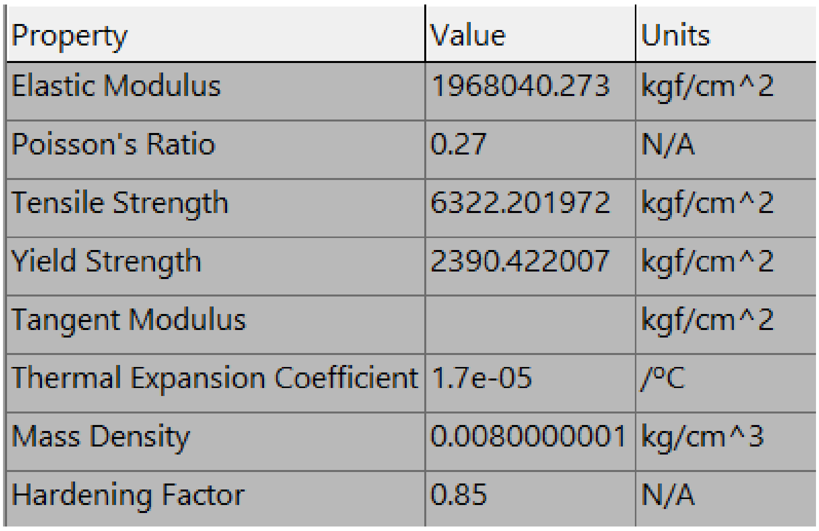
Overview

Our solution to the problem of landing a rocket upright integrates three key mechanical components: gimbal implementation, landing gear addition, and a nose cone redesign. The material these parts are manufactured from also plays an important role and has been considered carefully. Finally, vertical landing requires robust controllers in both the vertical axis and in the gimbal that can accommodate disturbances introduced from a variety of sources.

Together, these elements address the critical challenges of stability, precision, and impact absorption, as defined in our design criteria. Each component has been developed with an emphasis on efficiency, reliability, and adaptability to ensure successful landings under various conditions.

Topic 1: Material Selection

AISI 321 Annealed Stainless Steel SS



Property	Value	Units
Elastic Modulus	1968040.273	kgf/cm^2
Poisson's Ratio	0.27	N/A
Tensile Strength	6322.201972	kgf/cm^2
Yield Strength	2390.422007	kgf/cm^2
Tangent Modulus		kgf/cm^2
Thermal Expansion Coefficient	1.7e-05	/ºC
Mass Density	0.0080000001	kg/cm^3
Hardening Factor	0.85	N/A

Figure 1: Material Properties of AISI 321

Stainless steel was selected for numerous reasons. The most significant factor is the ability to maintain a high yield strength at increasing temperatures. Compared to 6061 Aluminum, AISI 321 stainless steel is significantly heavier and more expensive. The thrust from the engines was adequate to compensate for the additional mass, while the ability to withstand higher temperatures accounted for the additional price. The need to withstand high temperatures is very important as the rocket will travel above the speed of sound. The choice of stainless steel for large rockets has been proven with the most notable example being SpaceX Starship.

Topic 2: Gimbal Implementation

Gimbal systems are crucial for thrust vector control, enabling the rocket to adjust its trajectory during descent. By tilting the engine's thrust direction relative to the rocket's center of mass, gimbals allow for fine-tuned corrections to stabilize orientation and counteract disturbances such as wind or uneven terrain below.

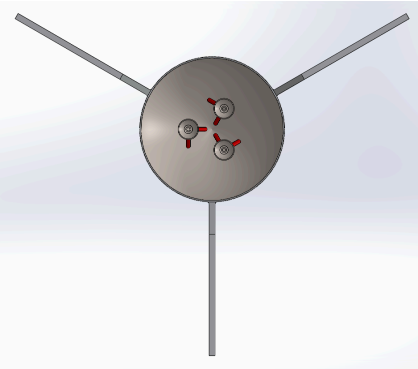


Figure 2: Bottom and side view of the gimbal engines at the base of the rocket

Design Approach

The design incorporates three thrusters mounted on gimbals, which adjust the orientation of each thruster to dynamically control the thrust vector based on real-time feedback from onboard sensors. A proportional-integral-derivative (PID) controller processes data from gyroscopes and accelerometers to maintain vertical alignment, minimizing angular drift even under external disturbances. Stability is further enhanced by strategically positioning the fins, as seen in **Figure 3**, and ensuring the center of pressure is located behind the center of gravity, promoting aerodynamic balance throughout descent. The three engines are designed around the specifications of Raptor V3 with a slightly heavier gimbal. The red linkages are hydraulic actuators for thrust vector control. The first of the orthogonal actuators is for the x position, while the other is for the y position. Three engines are used as this would allow for finer control in series with the gimbal to increase the precision of control.

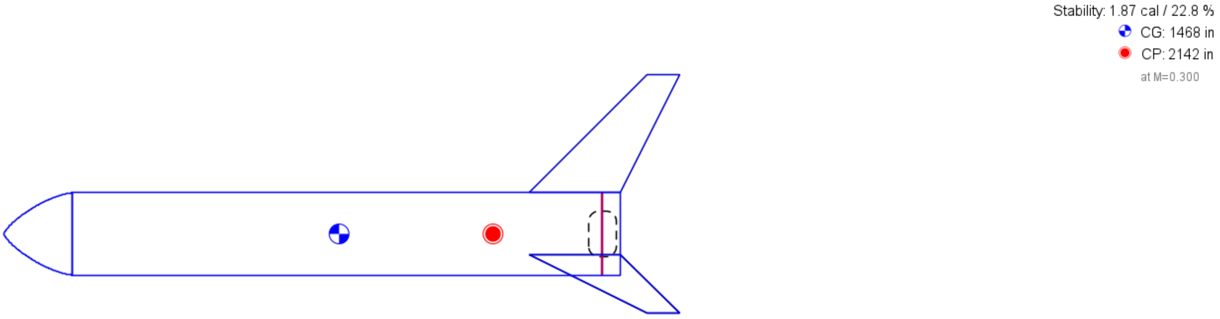


Figure 3: OpenRocket Simulator modeled rocket.

Topic 3: Landing Gear Addition and Topology Optimization

Landing gear provides critical support to absorb impact forces and stabilize the rocket upon touchdown. The design incorporates collapsible legs with shock-absorbing mechanisms to prevent tipping or structural damage.

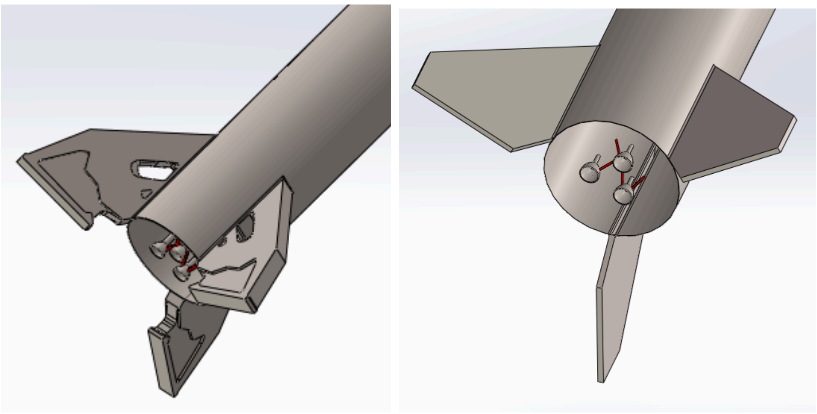


Figure 4: Isometric view of the original landing gear/fins vs final design

Design Approach

The landing gear consists of stainless steel fins equipped with pneumatic dampers that compress upon impact to dissipate energy. The legs are arranged around the rocket's base to maximize stability and are foldable to minimize drag during ascent. In the future, the system should also include footpads designed to distribute forces evenly, even on uneven terrain.

Implementation and Results

Using SolidWorks Topology Optimization, the design space was iteratively refined to remove unnecessary material while retaining the structural integrity required to support the calculated loading weight. The final design achieves a 28% weight reduction with respect to the base mold while meeting all performance criteria for impact absorption validated through FEA with a Factor of Safety of 8.3, as seen in **Figure 5**.

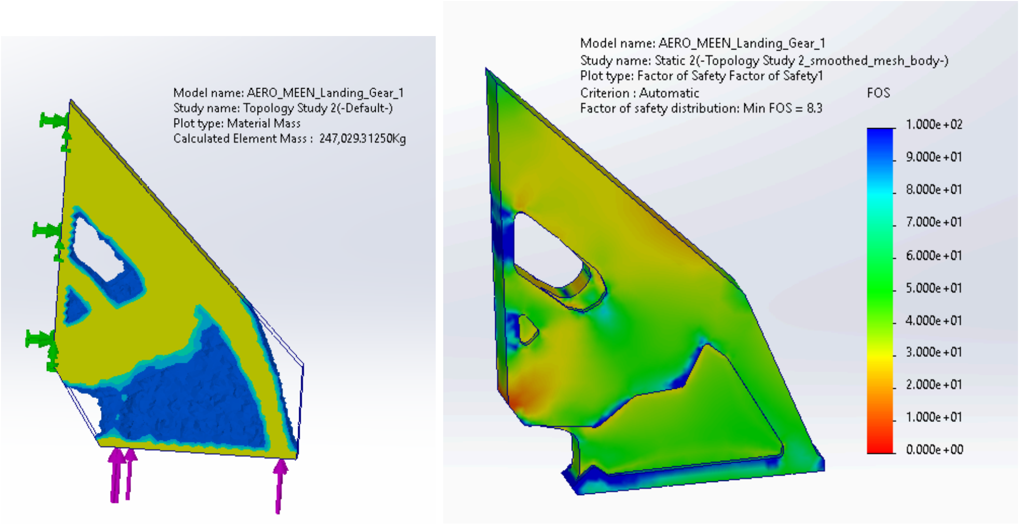

Model name: AERO_MEEN_Landing_Gear_1
Study name: Topology Study 2(-Default-)
Plot type: Material Mass
Calculated Element Mass : 247,02931250Kg
Model name: AERO_MEEN_Landing_Gear_1
Study name: Static 2(-Topology Study 2_smoothed_mesh_body-)
Plot type: Factor of Safety Factor of Safety1
Criterion : Automatic
Factor of safety distribution: Min FOS = 8.3
FOS
1.000e+02
9.000e+01
8.000e+01
7.000e+01
6.000e+01
5.000e+01
4.000e+01
3.000e+01
2.000e+01
1.000e+01
0.000e+00

Figure 5: Landing gear topology and FEA analysis

Topic 4: Nose Cone Re-design and Computational Fluid Dynamics (CFD)

The nose cone plays an essential role in aerodynamics during ascent and stability during descent. By optimizing its shape and material properties, we improved its performance in both phases of flight.

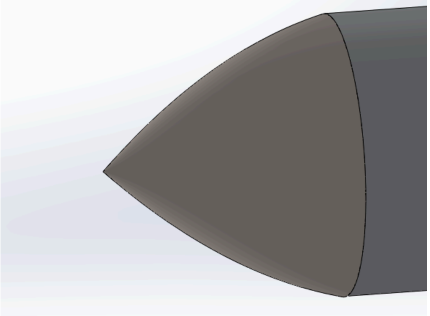


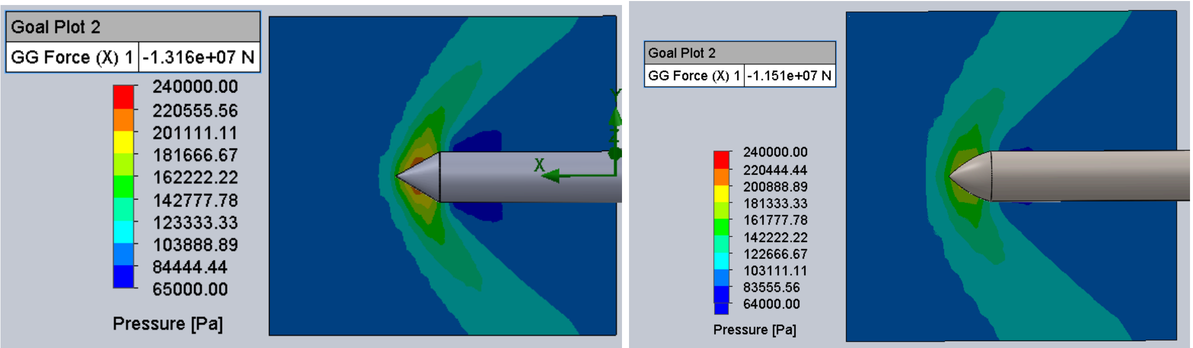
Figure 6: 3/4 Parabolic nose cone design, side view

Design Approach

The redesigned nose cone (**Figure 6**) features a 3/4 parabolic shape that minimizes drag during ascent and promotes aerodynamic stability during descent. The 3/4 parabolic shape is one of the most efficient shapes at supersonic speeds [7]. The nose cone also houses sensors that work in conjunction with the gimbal system to monitor wind patterns and atmospheric conditions.

Implementation and Results

Using SolidWorks' Flow Simulation, the nose cone design was tested at standard flight conditions during the ascent stage (500 m/s). Compared to the cone design, the new nose cone resulted in a 12.5% reduction in drag forces and a lower distribution of pressures at the nose cone caused by the supersonic flows, as seen in **Figure 7**.

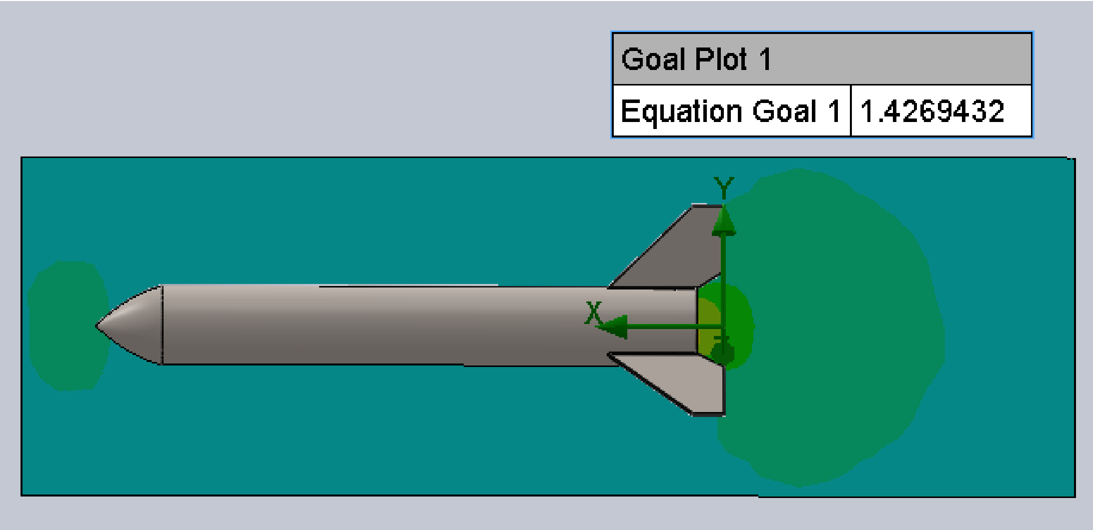


Goal Plot 2	
GG Force (X) 1	-1.316e+07 N

Goal Plot 2	
GG Force (X) 1	-1.151e+07 N

Figure 7: Nose cone CFD analysis of old vs final design at supersonic speeds

Additionally using CFD, the coefficient of drag was calculated from the model by using the coefficient of drag equation and the drag forces. The flow was directed upward from the bottom of the rocket to simulate the downward movement upon descent at 50 m/s. This resulted in a coefficient of drag of 1.427 (**Figure 8**), which was used for the trajectory simulations.

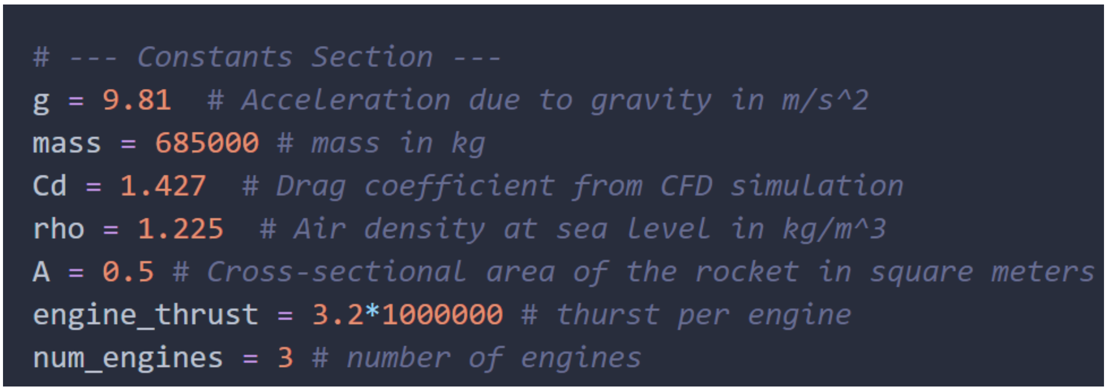


Goal Plot 1	
Equation Goal 1	1.4269432

Figure 8: CFD of the rocket in descent & drag coefficient

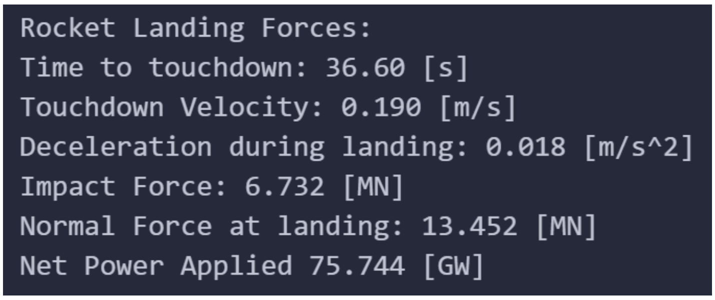
Topic 5: Landing Controls

The control algorithm for the descent and landing process is a crucial element of the success of a rocket's performance. The descent must be controlled, feasible, and minimize the normal force of landing applied to the rocket as it touches down. There are many factors that could be considered, including but not limited to fuel efficiency, landing time, peak force minimization, thruster performance accommodations or acceleration constraints. Regardless, a solution focusing on optimization would require an in-depth cost function analysis of these parameters. Solving for an optimal solution to this very complex problem **[6]**; however, for the scope of this report, only feasibility and force minimization will be considered.



```
# --- Constants Section ---
g = 9.81  # Acceleration due to gravity in m/s^2
mass = 685000 # mass in kg
Cd = 1.427  # Drag coefficient from CFD simulation
rho = 1.225  # Air density at sea level in kg/m^3
A = 0.5 # Cross-sectional area of the rocket in square meters
engine_thrust = 3.2*1000000 # thurst per engine
num_engines = 3 # number of engines
```

Figure 9: Rocket landing forces script inputs



```
Rocket Landing Forces:
Time to touchdown: 36.60 [s]
Touchdown Velocity: 0.190 [m/s]
Deceleration during landing: 0.018 [m/s^2]
Impact Force: 6.732 [MN]
Normal Force at landing: 13.452 [MN]
Net Power Applied 75.744 [GW]
```

Figure 10: Rocket landing forces script output

Design Approach

The landing controls system ensures the rocket maintains stability and lands upright by processing real-time sensor data to dynamically adjust thrust and orientation. This system combines a model representing a stable hover with an error-driven controller that integrates feedback from onboard sensors with control algorithms, enabling precise adjustments to gimbals and thrusters during descent. This system demonstrates effective landing in reasonable conditions with a minimal touchdown velocity **[Figure 10]**.

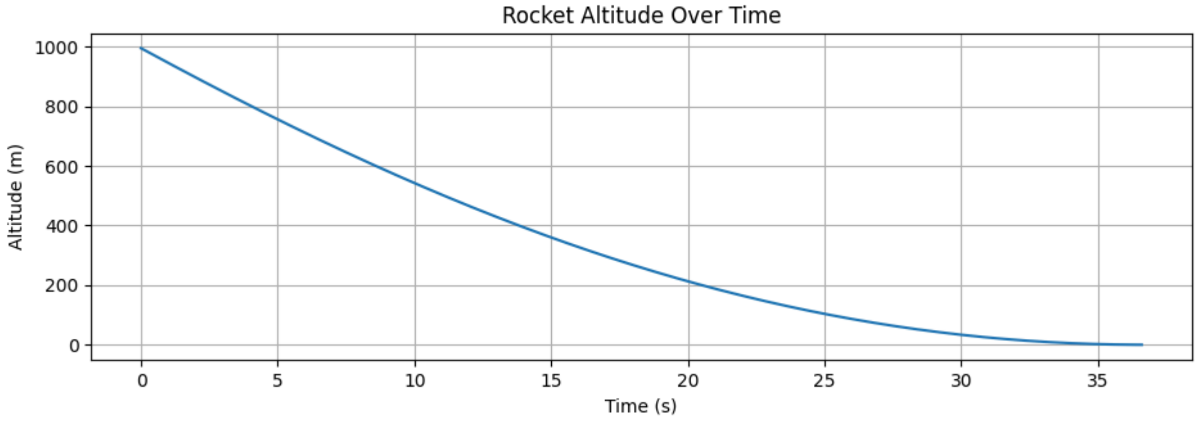


Figure 11: Simulated rocket altitude while landing

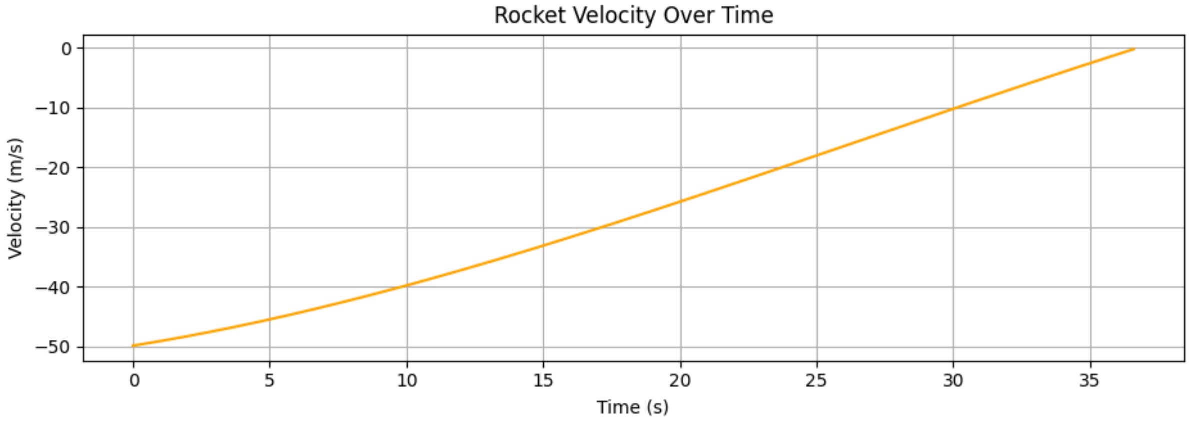


Figure 12: Simulated rocket velocity while landing

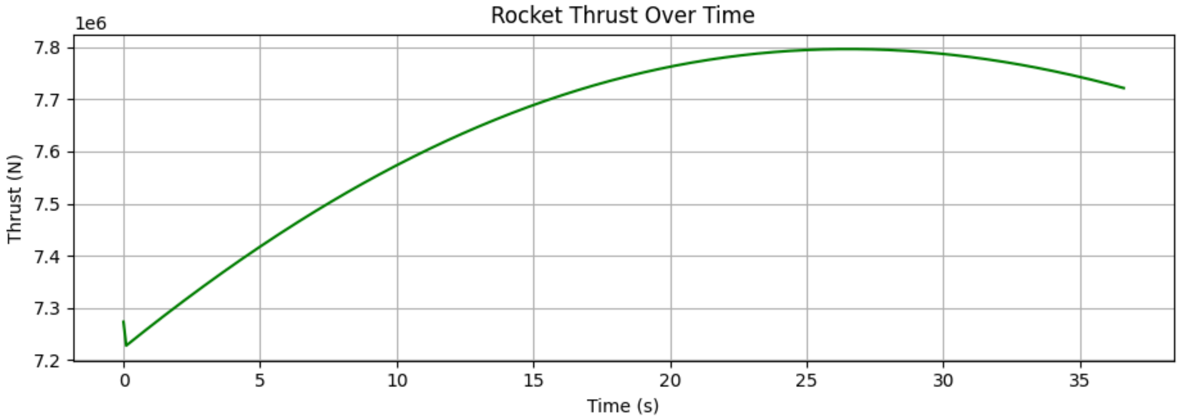


Figure 13: Simulated rocket thrust while landing

Altitude Control Algorithm Design

The control system is based on a Hybrid Controller or Model Predictive Controller (MPC) containing both a model term and an error-driven term. The model term is based on precalculated values known about the object being controlled and its current position, whereas the error-driven Proportional-Integral-Derivative (PID) controller element processes data from accelerometers, tachometers, and altimeters to correct the actual differences between setpoints and real-time measured outputs **[Figure 9]**, **[Figure 10]**. The model calculates the thrust required to maintain the current velocity by accounting for the weight and drag forces known, and the PID algorithm calculates the necessary adjustments to the thrust output to account for all variance from equilibrium. This system has advantages over purely Model Driven approaches as it can account for disturbances during flight to ensure that disturbances cannot accumulate into large errors. This system has advantages over purely error-driven approaches as the model accounts for a majority of the thrust required, and the PID controller can be more narrowly refined to account exclusively for the adjustments to the model **[Figure 11], [Figure 12], [Figure 13]**. This system will likely have a marginally slower refresh rate as compared to either of the previously discussed approaches, however, the increased reliability and performance would consistently outperform the alternatives **[4]**.

The starter code provided the foundations for solving the dynamics problem while leaving the controller open-ended, with a bias towards an open-loop controller with no feedback **[9]**. The solution implemented has the advantage of resilience towards disturbances.

Gimbal Control Algorithm Design

The gimbal control system would be based purely on an error-driven PID controller, which processes data from gyroscopes, accelerometers, magnetometers and an Internal Measurement Unit (IMU) to minimize the error between the current state of the rocket and the neutral vertical position. The PID algorithm calculates the necessary adjustments to the gimbals to maintain vertical alignment and counteract external forces such as wind or drift. This system does not require a model driven element in the current application as it is meant to purely maintain the stability of the flight. If a model were to be implemented in this system, it would be based on a two-dimensional inverted pendulum, for which each axis could be solved independently **[5]**. This kind of extended functionality would allow for horizontal displacement correction, allowing for precise control over the location of landing.

Conclusion

The challenge of safely and efficiently landing a rocket upright is pivotal in advancing reusable spaceflight technology. Our solution integrates three key innovations: gimbal implementation for thrust vector control, landing gear to absorb impact forces and maintain stability, and a redesigned nose cone for improved aerodynamics and sensor integration. Together, these components address the critical design criteria of stability during descent, precision of landing, and effective impact absorption, demonstrating a holistic approach to solving this complex engineering problem.

Challenges and Lessons Learned

Throughout the project, the team faced several challenges, including balancing lightweight designs to durability, ease of simulation and accuracy, and accurately simulating real-world conditions. The team also had to work on parallel development and understanding how one layer of analysis affected the others. For example, as we were performing the Computation Fluid Dynamics, we realized that an error meant that there was a slightly different drag coefficient, and we needed to retune the PID controller for this new value. Many times in industry, this kind of thing will happen, and this experience demonstrates the need for organized planning and coordination for dependent processes.

Overall, these challenges highlighted the fundamentally interdisciplinary nature of many classically-disciplined engineering challenges that occur in industry and research, while underscoring the importance of iterative testing, collaborative development and project planning.

From this experience, we learned to approach engineering problems systematically, breaking them into manageable components and validating each step through simulations. Additionally, we gained insights into how theoretical knowledge translates into practical design considerations.

Contributions

- **Ian Wilhite**: Designed the control algorithms for the landing, ensuring stable descent.
- **Eduardo Burciaga-Ichikawa**: Led the development of the landing gear, including material selection and impact testing, along with running topology and CFD analysis.
- **Kalen Jaroszewski**: Designed and tested the aerodynamic properties of the nose cone, integrating sensors for enhanced feedback.
- **Julia Sopala**: Consolidated data, and ensured alignment of all components with the design criteria and reporting it.

Room for Growth

Looking forward, next steps for the project could include a more in depth comparison of materials and consideration of design for manufacturing of parts, testing the design under more variable environmental conditions, and prototyping scaled physical models for validation. The controllers could be expanded to utilize the varied power between the three thrusters, or to include drag manipulators like flaps. Testing could include further validity testing with the current altitude controller, and implementation of the gimbal controller and full system simulations. In implementing the ideas discussed, this system could become a reliable method for assisting rocket landing accuracy and improving affordability of space missions.

References

[1] R. Ferrante, "A robust control approach for rocket landing," *Theses Univ. Edinburgh,* 2017.

[2] C. Fang *et al.*, "Gru & Vector Gimbal Rocket Design and Launch Report," in *2024 Regional Student Conferences*, 2024, p. 85702.

[3] C. Trom, "Development of a vertically landed rocket design challenge for expanding the pipeline and enhancing education of students pursuing careers in space," University of Illinois at Urbana-Champaign, 2022.

[4] Z. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3–35, 2013, doi: 10.1016/j.ins.2012.07.014.

[5] İ. Yiğit, "Model free sliding mode stabilizing control of a real rotary inverted pendulum," *International Journal of Control, Automation and Systems*, vol. 23, no. 10, 2015, doi: 10.1177/1077546315598031.

[6] C. Wang and Z. Song, "Trajectory optimization for reusable rocket landing," in *2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*, Xiamen, China, 2018, pp. 1-6, doi: 10.1109/GNCC42960.2018.9019105.

[7] V. K. Shah, "Determination of the optimum nose cone geometrical shape for supersonic missile," Aerospace Science and Technology, vol. 133, 2023, doi: 10.1016/j.ast.2022.106774.

[8] A. Aboelhassan, M. Abdelgeliel, E. E. Zakzouk, and M. Galea, "Design and implementation of model predictive control based PID controller for industrial applications," *Energies*, vol. 13, no. 24, p. 6594, 2020, doi: 10.3390/en13246594.

[9] B. Borovic, A. Q. Liu, D. Popa, H. Cai, and F. L. Lewis, "Open-loop versus closed-loop control of MEMS devices: choices and issues," *Journal of Micromechanics and Microengineering*, vol. 15, no. 10, p. 1917, 2005, doi: 10.1088/0960-1317/15/10/018.

Back to Contents

5. Mechanical Modeling (Best Prototype Award)

Summary

The need was a toy for the elderly which could help preserve dexterity. Our team recieved the ”Best Prototype” award out of 80 students.

- meen-210-Final_Project.pdf

Contributors: Astrid Garcia, Mckenzie McCain, Ryan Welty, Ian Wilhite

Key Skills: Mechanical Design, Computer Aided Design (CAD), SolidWorks, Finite Element Analysis (FEA), 3D Modeling, Technical Documentation.

Relevance: This project showcases the ability to design, model, and analyze mechanical components and assemblies using industry-standard software. It reflects a strong understanding of mechanical design principles.

FIDGET FUN

GROUP 13: MEEN 210-505
FALL '24 PROJECT

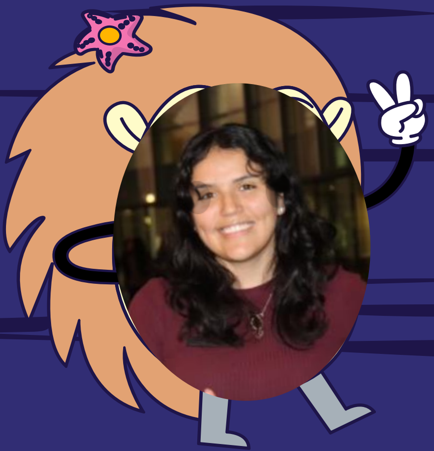
By Ryan Welty, Mckenzie McCain, Ian Wilhite and Astrid Garcia

THE TEAM

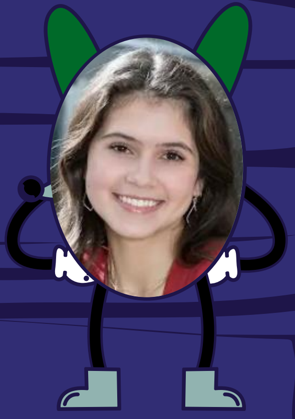
MEEN 210-505 FALL SEMESTER

IAN WILHITE

RYAN WELTY



ASTRID GARCIA



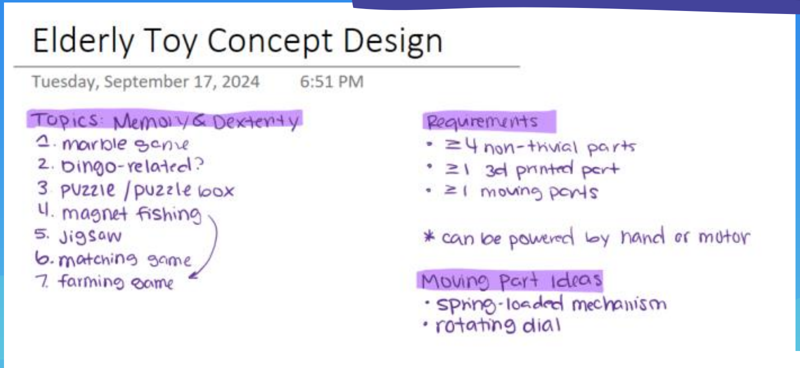
MCKENZIE MCCAIN

VIDEO ADVERTISEMENT

WHAT IS FIDGET FUN

A customizable, modular fidget toy that transforms into endless puzzle track configurations! Simply snap the pieces together, and let the magic unfold as a smooth ball glides along the creative paths you design.

CONCEPTS



Elderly Toy Concept Design

Tuesday, September 17, 2024 6:51 PM

Topics: Memory & Dexterity

1. marble game
2. bingo-related?
3. puzzle / puzzle box
4. magnet fishing
5. Jigsaw
6. matching game
7. farming game

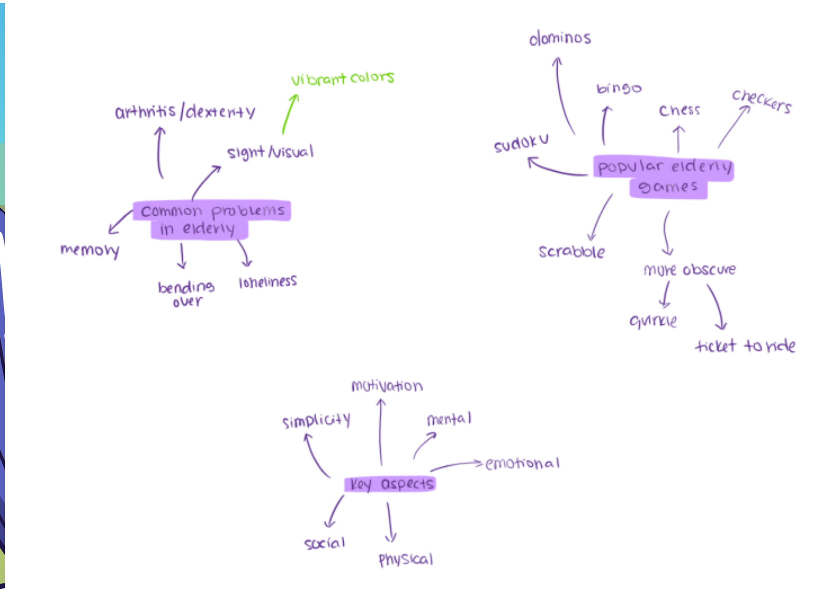
Requirements

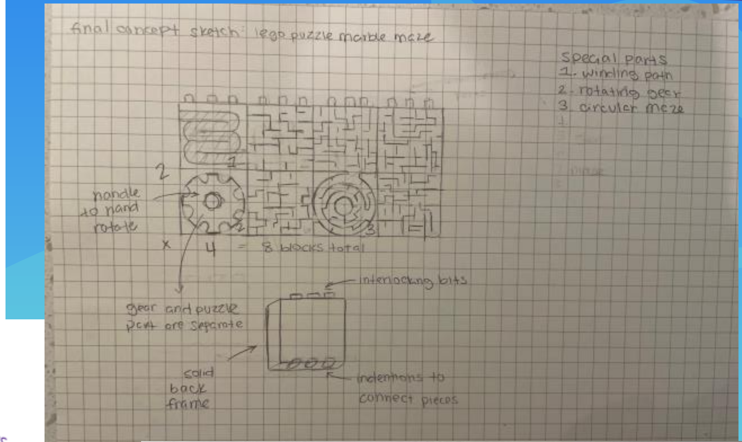
- ≥4 non-trivial parts
- ≥1 3d printed part
- ≥1 moving parts

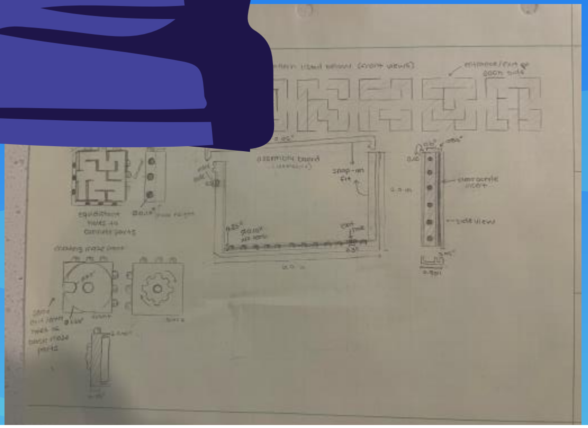
* can be powered by hand or motor

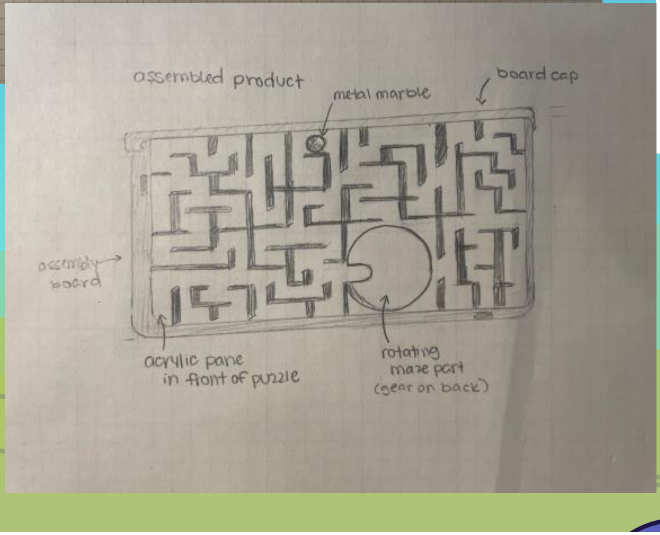
Moving Part Ideas

- spring-loaded mechanism
- rotating dial

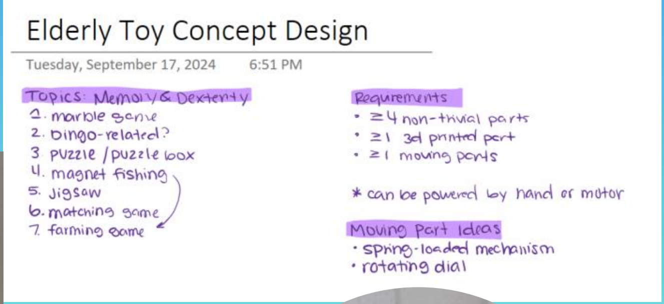








assembled product
board cap
metal marble
assembly board
acrylic pane in front of puzzle
rotating maze part (gear on back)



Elderly Toy Concept Design

Tuesday, September 17, 2024 6:51 PM

Topics: Memory & Dexterity

1. marble game
2. bingo-related?
3. puzzle / puzzle box
4. magnet fishing
5. Jigsaw
6. matching game
7. farming game

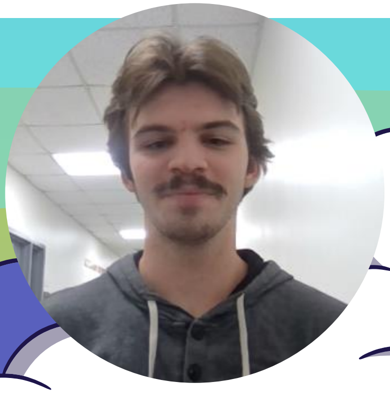
Requirements

- ≥4 non-trivial parts
- ≥1 3d printed part
- ≥1 moving parts

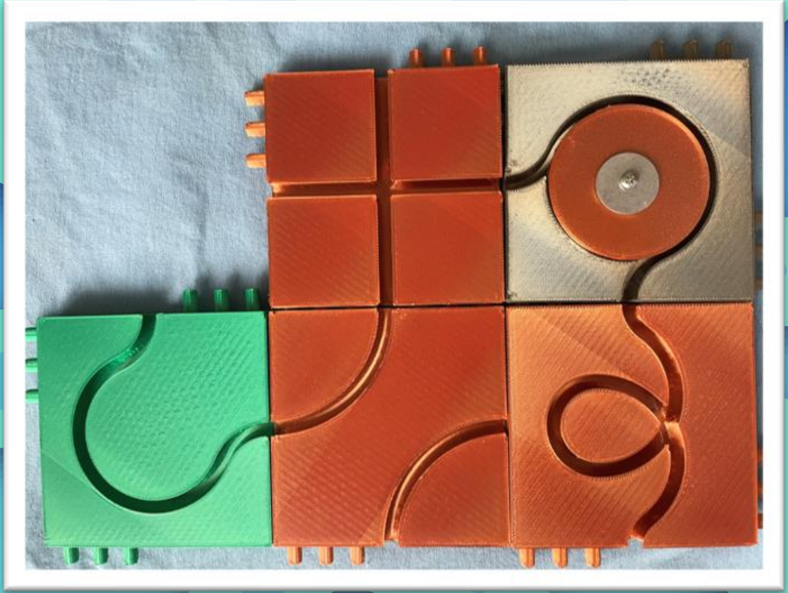
* can be powered by hand or motor

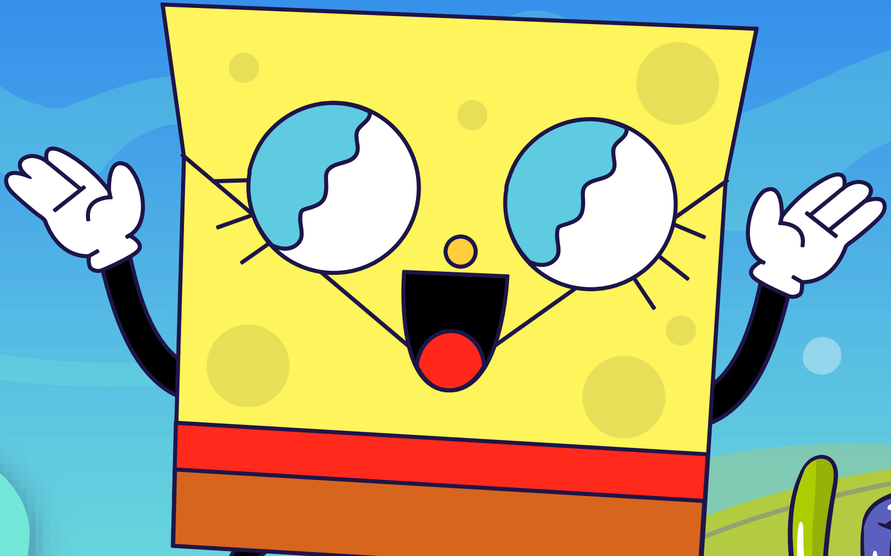
Moving Part Ideas

- spring-loaded mechanism
- rotating dial



THE PRODUCT





PROBLEM & NEEDS

"My grandma loves gardening, reading, and solving puzzles. Recently, she has started noticing changes. Her hands feel stiff when she tries to grab plates from the cabinets or even when she goes to grab her favorite book. She has also noticed that she often forgets where she places her keys and has found it hard to concentrate for long periods of time. This has begun to make her sad and frustrated because her body doesn't work the way it used to. She needs something that will engage her mind and helps her feel calm and in control"

PROBLEM & NEEDS

Target Audience

- Intended for anyone in the elderly age range group, someone 65 years or older. Although it is not restricted to just this age group. This product is also intended for people suffering from medical conditions such as Alzheimer's or dexterity /mobility issues.

Why & How would they use your product?

- Why: Fun, colorful, and easy to use!
- How: Just snap the pieces together and watch the ball glide through the path

Needs Statement

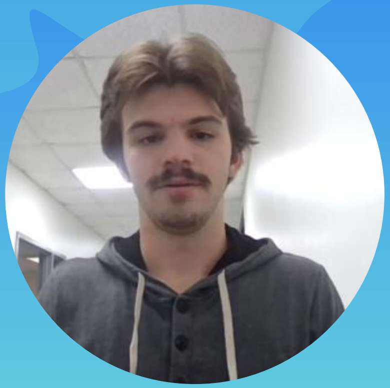
•Engaging mental stimulation
•Price-conscious
•The key aspects should include topics: social, physical, mental, or emotional
•Accommodating to elderly individuals
•Intuitive gameplay mechanics

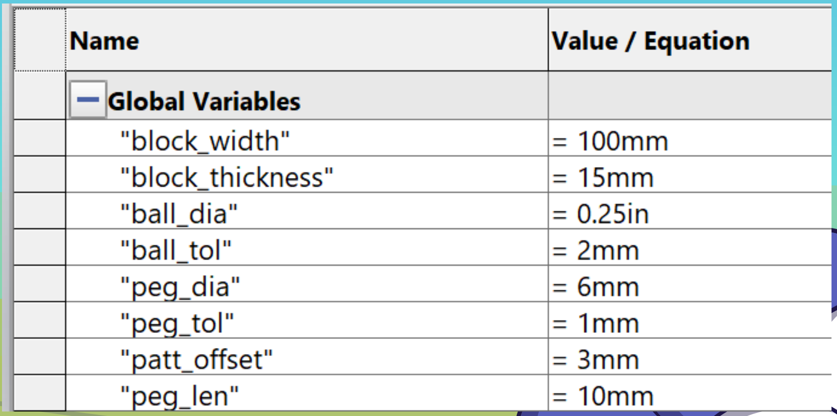
•Based off the project outlines from milestone 0, we were able to produce our needs statement

ENGINEERING REQUIREMENTS

Design Components:

- Block Width
- Block Thickness
- Ball Diameter (from stock ball bearing)
- Tolerance around ball
- Length of pegs
- Depth of path

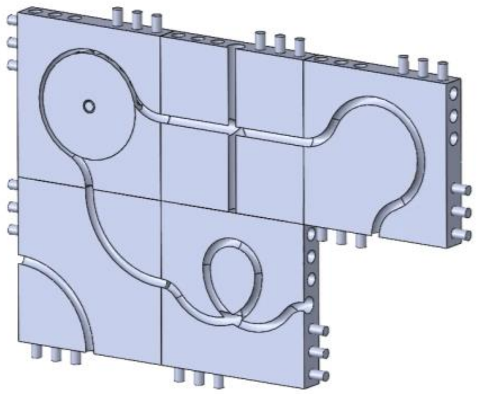




Name	Value / Equation
Global Variables	
"block_width"	= 100mm
"block_thickness"	= 15mm
"ball_dia"	= 0.25in
"ball_tol"	= 2mm
"peg_dia"	= 6mm
"peg_tol"	= 1mm
"patt_offset"	= 3mm
"peg_len"	= 10mm

MODELING

Animated Exploded View of Fidget Fun Assembly



FABRICATION PROCESS

MACHINE USED: Creality K1 3D printer

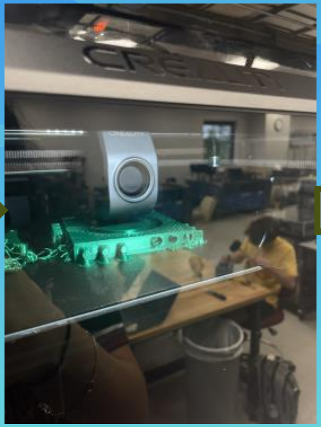
MATERIAL: Smooth PLA
Washer & Screw

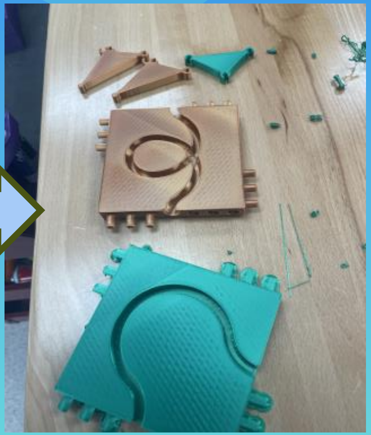
23 hr print time, 310.96 grams

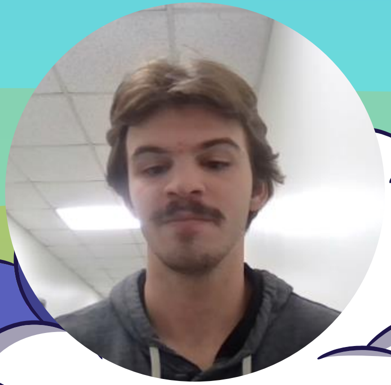
Same scale print

Conformity :

- The washer and screw
- The tolerance
- Size of ball bearing
- Scratched board design
- Orientation of the plates







THANKS AND GIG EM'

Back to Contents

6. Solid Mechanics

Summary

This section contains two projects from the Solid Mechanics course (MEEN 305).

- Project 1: Hollow-Tube Truss Analysis
- Project 2: Solar Flower Frame Analysis

Contributors: Nick Licon, Andres Pinzon, Ian Wilhite, and David Wood

Key Skills: Solid Mechanics, Stress Analysis, Strain Analysis, Material Behavior, Finite Element Analysis (FEA), Structural Analysis.

Relevance: These projects demonstrate a solid understanding of the principles of solid mechanics and their application to the analysis of structural components under various loading conditions.

Design and Analysis of a 2D Truss

MEEN 305-500 – Project 1

By Ian Wilhite, David Wood, Nick Licon, and Andres Pinzon

Introduction

In this project, we designed and analyzed a two-dimensional truss structure to determine an optimal layout with optimal cross-sectional geometry for each truss member to carry multiple point loads. To simplify our design process and analysis, we (1) generated truss designs with point loads applied exactly at joint locations, (2) assumed the weight of truss members to be negligible, and (3) assumed “smooth pins” at all truss member joints so that bending moments are neglected. These assumptions allowed us to treat each truss member as two-force members, either in tension or compression. Therefore, the only factors to consider when testing the structural integrity are the yield strength and critical buckling load.

The two-dimensional truss was required to fulfill the following specifications:

- Span a distance of 20 ft.
- Be made of A-36 steel.
- Support a 15-kip load at its center, a 10-kip load located 4 ft from one side, and a 5-kip load 4 ft from the other.
- Contain a minimum safety factor of 1.5 for all truss members under any potential failure mechanism (either yielding or buckling)
- For all truss members to have circular cross-sections

The first part of our design process was to brainstorm potential truss models. Each team member generated at least one unique truss design based on their analysis as seen in **Figures 1-5**. We then determined the minimal amount of mass required to satisfy the design constraints listed above for each truss design, accounting for failure due to either yielding or buckling. The design with the least amount of mass required to satisfy the design requirements was selected as the best model. Finally, we verified our results by calculating the individual factor of safety for each member of our final design, ensuring that they were all at least 1.5.

Truss Model

Individual Models:

As a part of the collaborative brainstorming process, multiple proposals were generated. They intended to take vastly different approaches to the problem to ensure a wide variety of possible solutions were considered. A common analysis process was then generated as a method to effectively determine the best model.

Method 1: Square Frame - Ian Wilhite

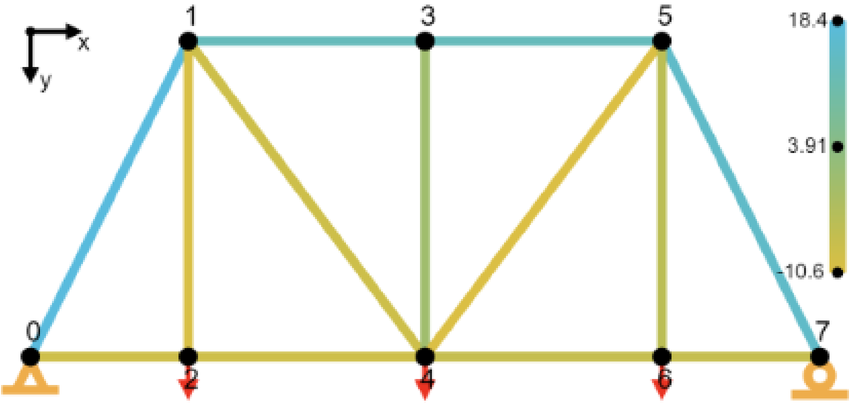


Figure 1. Visual analysis for square frame truss.

This method was intended to take inspiration from many train bridges across the country, and their very square or rectangular appearance by creating parallel beams and adding cross branches to support each segment. The segment widths were modified to allow for loading at the correct increments for analysis.

Method 2: Webbed Frame - David Wood

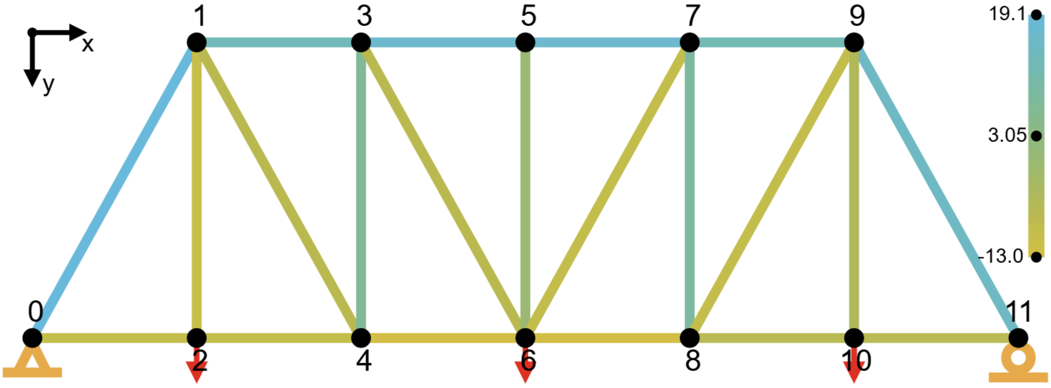


Figure 2. Visual analysis for webbed frame truss.

This model explored whether it would be worth increasing the number of webs in the truss to reduce the overall load for each truss member. While more webs correspond to more needed material, making the bridge heavier, it also contains members significantly shorter in length. This makes them much more resilient to buckling or bending. However, the actual tensile/compressive stress exerted per member remained approximately the same, compared to this same bridge above with 1 web.

Method 3: Parabolic-inspired Frame - Nicholas Licon

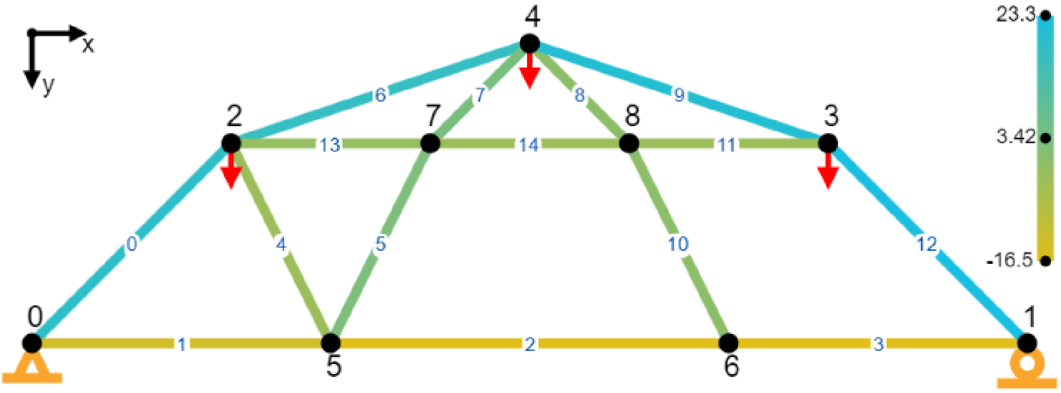


Figure 3. Visual analysis for the Parabolic-inspired Frame.

This shaped truss came from the idea that forces applied to a curved structure are often stronger than others. The shape of a “parabolic” structure distributes the load along the curve of the structure creating members of compression spaced out along the curve as shown with the blue two force members.

Method 4: Pentagonal Frame - Andres Pinzon

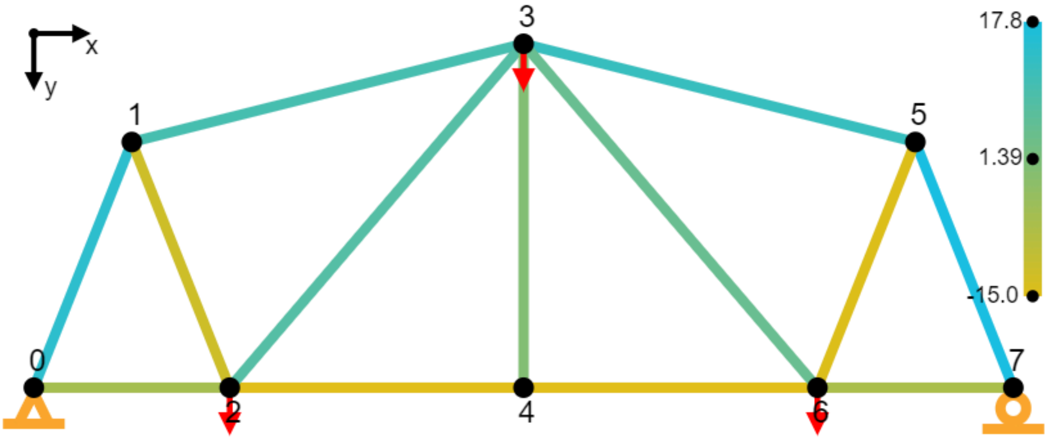


Figure 4. Visual analysis for pentagonal frame truss.

This model was created based on modifications to an already existing warren bridge. The main change is increasing the main member's height. This created a more pentagonal look to the frame and it was noticeably stronger than a simple warren bridge model. The axial force each member sustained was minimized to keep each member's area from growing too much.

Method 5: Triangular Frame - Ian Wilhite

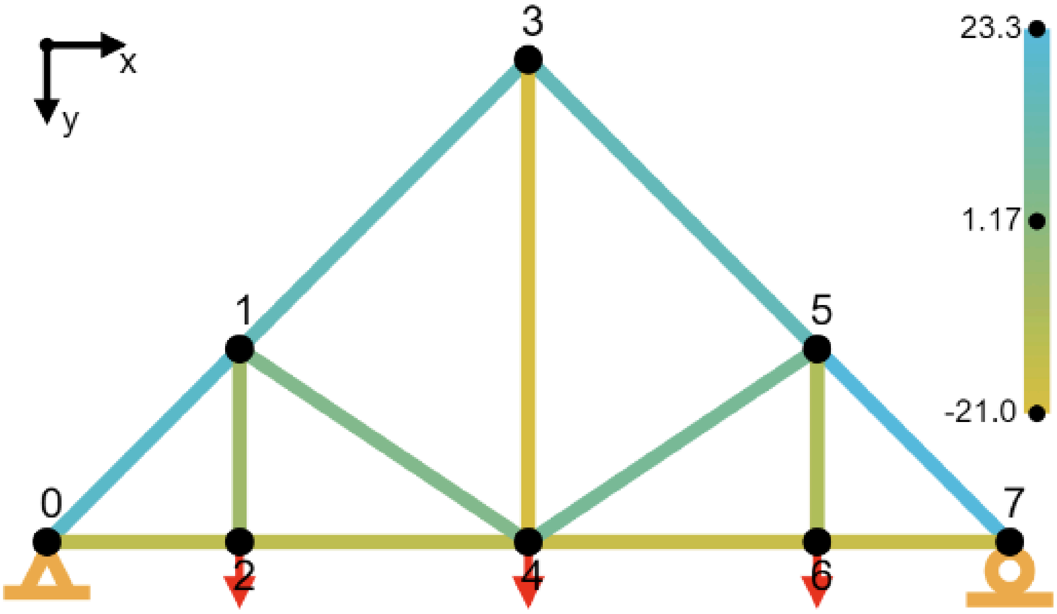


Figure 5. Visual analysis for triangular truss.

This model was meant to remain as simple as possible to theoretically take advantage of the roundup error in frames with many small beams all being rounded up to the nearest quarter inch. This resulted in a substantially weaker truss because the fewer members resulted in less optimal placement of each.

Chosen Model: Parabolic Inspired - Nicholas Licon

This model reduced the total volume of material needed to support the weight. This model leverages a proper balance between linkage placement and roundup radii to generate a truss that can effectively support the loading provided.

Justification of Choice

Each truss was analyzed to determine the total volume of material it would require. The total volume of each truss was then compared to determine the optimal design for the loading conditions.

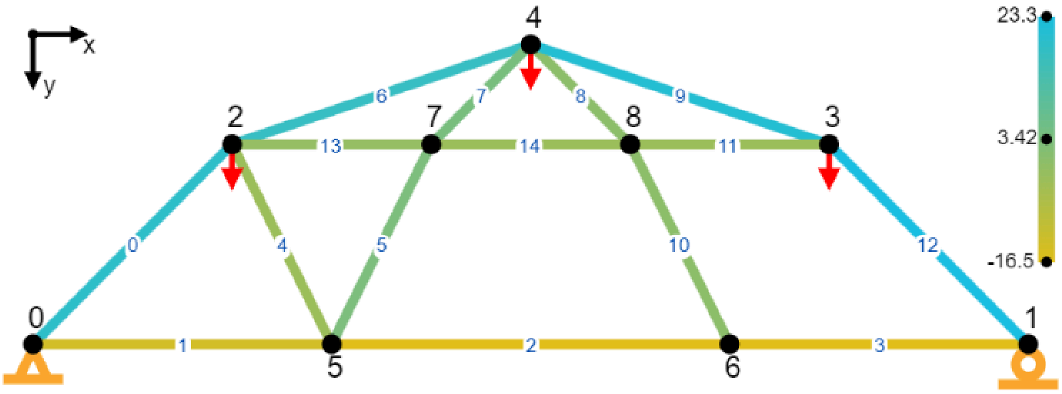
#	Name	Designer	Volume Required (in^3)	Weight Required (lb)
1	Square Frame	Ian Wilhite	2444.240	694.164
2	Webbed Frame	David Wood	1596.073	453.284
3	Parabolic Inspired	Nicholas Licon	1227.537	348.621
4	Pentagonal Frame	Andres Pinzon	2204.261	626.010
5	Triangular Frame	Ian Wilhite	3690.243	1048.029

Table 1. Comparison of bridge designs.

Analysis Process

Each truss presented was analyzed initially using an online calculator which performed the node forces for each truss and presented the forces per member given the loading conditions. These results were imported into a spreadsheet in which each truss could be compared to intake the force, safety factor, and failure stress, to output the total volume and weight of the truss.

Geometric Properties



Density of A-36 steel = 0.284 lb/in^3

RED = Compression	BLUE = Tension

Member	Length (in)	Area (in^2)	Volume (in^3)	Weight (lbs)	FOS
0	67.884	1.767	119.961	34.069	2.555
1	72	0.785	56.545	16.060	2.0943
2	96	0.785	75.398	21.413	1.714
3	72	0.785	56.549	16.060	1.714
4	53.664	0.196	10.537	2.992	2.108
5	53.664	1.227	65.856	18.703	3.551
6	75.9	3.142	238.447	67.719	2.244
7	33.936	0.785	26.653	7.570	2.875
8	33.936	0.0491	1.666	0.473	N/A
9	75.9	3.142	238.447	67.719	1.899
10	53.664	0.0491	2.634	0.748	N/A
11	48	0.196	9.425	2.677	2.356
12	67.884	2.405	163.280	46.372	2.091
13	48	0.196	9.425	2.677	4.712
14	48	0.196	9.425	2.677	2.356

Table 2. Analysis of parabolic inspired truss.

TOTAL VOLUME, TOTAL WEIGHT: **1227.537 in^3**, **348.621 lbs**

Tasks and Responsibilities

Each member was expected to participate in group brainstorming, generate one truss design, and a brief description of their approach and inspiration for their truss. There was also a strong collaborative effort in gaining a fundamental understanding of the equations, simulation tools, and theory behind what constitutes a good truss.

Ian Wilhite - Ian was responsible for creating the spreadsheet to analyze the trusses, the framework of the report, the chosen model, and Tasks and Responsibilities.

Nicholas Licon - Nicholas was responsible for the Introduction.

Andres Pinzon Diaz - Andres was responsible for the geometric properties, excel appendix.

David Wood - David was responsible for the Introduction, and the generation of Geometric Properties for the selected truss.

Appendix

Matlab code for solving forces:

```
% Number of nodes and members
n_nodes = 9;
n_members = 15;

% Node coordinates (x, y)
nodes = [0, 0; 20, 0; 4, 4; 16, 4; 10, 6; 6, 0; 14, 0; 8, 4; 12, 4];
% External forces (Fx, Fy) at each node
forces = [0, 0; 0, 0; 0, -5; 0, -10; 0, -15; 0, 0; 0, 0; 0, 0; 0, 0];
% Connectivity matrix (start node, end node)
members = [0, 2; 0, 5; 5, 6; 6, 1; 2, 5; 7, 5; 2, 4; 4, 7; 4, 8; 4, 3; 6, 8; 3,
8; 3, 1; 2, 7; 7, 8] + 1;

% Initialize system of equations
n_eqs = 2 * n_nodes; % 2 equations (Fx and Fy) per node
A = zeros(n_eqs, n_members + 3); % Matrix of coefficients
b = zeros(n_eqs, 1);             % Vector of constants
% Reaction forces (supports at nodes 0 and 1)
% Loop through each member and define equations based on equilibrium
for member_id = 1:n_members
   n1 = members(member_id, 1);
   n2 = members(member_id, 2);

   % Calculate direction cosines (l, m)
   dx = nodes(n2, 1) - nodes(n1, 1);
   dy = nodes(n2, 2) - nodes(n1, 2);
   L = sqrt(dx^2 + dy^2);
   l = dx / L;
   m = dy / L;

   % For node n1 (Fx and Fy)
   A(2*n1-1, member_id) = l;  % Fx equation for node n1
   A(2*n1, member_id) = m;    % Fy equation for node n1
   % For node n2 (Fx and Fy)
   A(2*n2-1, member_id) = -l; % Fx equation for node n2
   A(2*n2, member_id) = -m;   % Fy equation for node n2
end
A(1, n_members+1) = 1; A(2, n_members+2) = 1;
A(4, n_members+3) = 1;
for i = 1:n_nodes
   b(2*i-1) = forces(i, 1); % Fx
   b(2*i) = forces(i, 2);   % Fy
end
% Solve for forces in each member and support reactions
disp(A);
disp(b);
x = A \ b;
% Display the results
forces_in_members = x(1:n_members);
reaction_forces = x(n_members+1:end);
disp('Forces in members (kips):');
disp(forces_in_members);
disp('Reaction forces (kips):');
disp(reaction_forces);
```

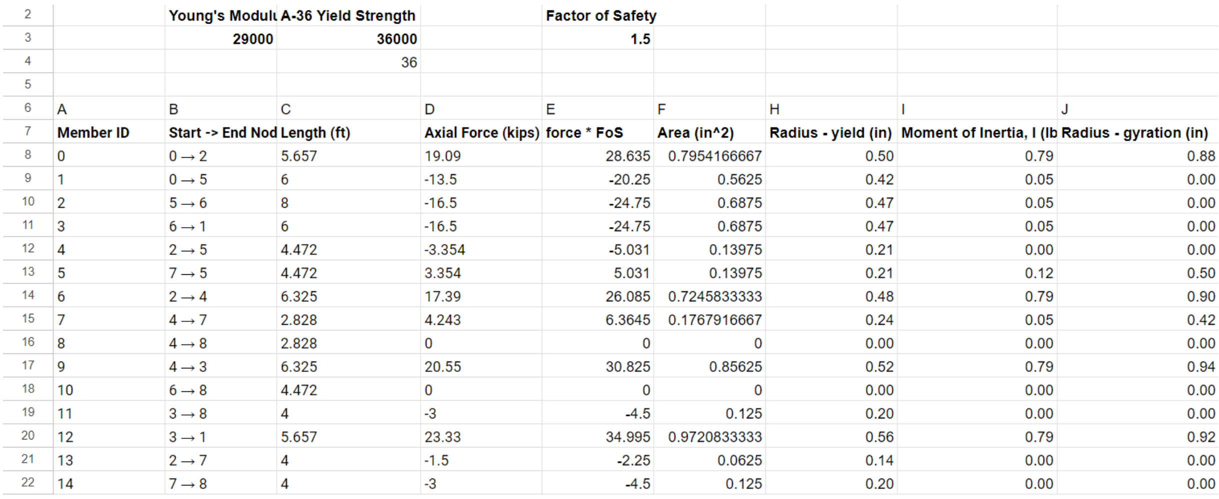
Output of MatLab code:

```
Forces in members (kips):
  19.0919
 -13.5000
 -16.5000
 -16.5000
  -3.3541
   3.3541
  17.3925
   4.2426
   0.0000
  20.5548
   0.0000
  -3.0000
  23.3345
  -1.5000
  -3.0000

Reaction forces (kips):
   0.0000
 -13.5000
 -16.5000
```

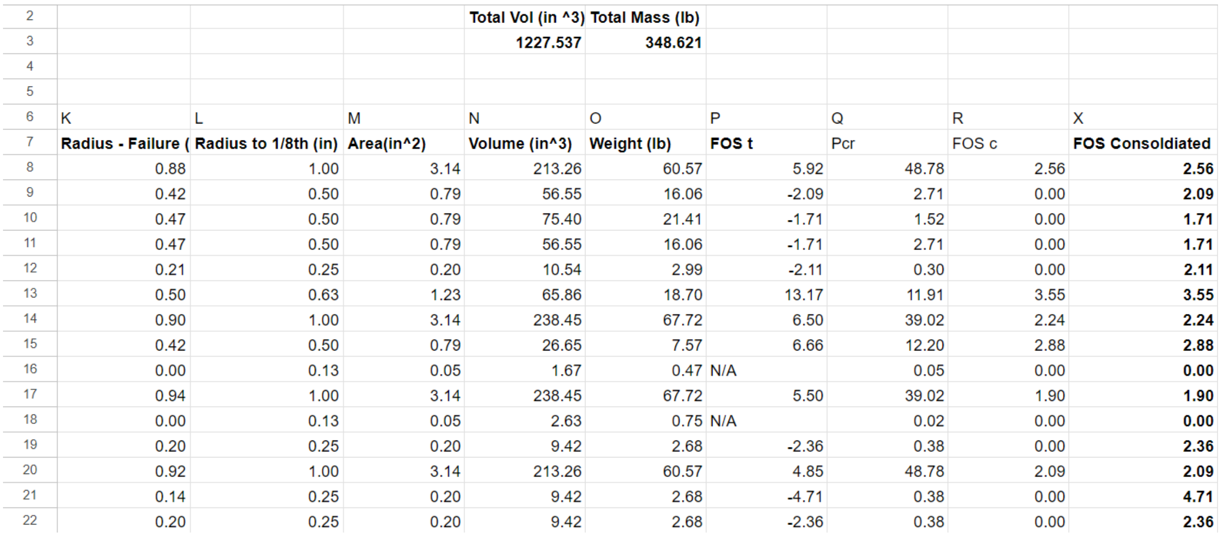
<u>Truss visualization:</u>

trussanalysis.com

<u>Excel sheets used to calculate geometric properties:</u>



2		**Young's Modulu**	**A-36 Yield Strength**		**Factor of Safety**				
3		**29000**	**36000**		**1.5**				
4			36						
5									
6	A	B	C	D	E	F	H	I	J
7	**Member ID**	**Start -> End Nod**	**Length (ft)**	**Axial Force (kips)**	**force * FoS**	**Area (in^2)**	**Radius - yield (in)**	**Moment of Inertia, I (lb**	**Radius - gyration (in)**
8	0	0 → 2	5.657	19.09	28.635	0.7954166667	0.50	0.79	0.88
9	1	0 → 5	6	-13.5	-20.25	0.5625	0.42	0.05	0.00
10	2	5 → 6	8	-16.5	-24.75	0.6875	0.47	0.05	0.00
11	3	6 → 1	6	-16.5	-24.75	0.6875	0.47	0.05	0.00
12	4	2 → 5	4.472	-3.354	-5.031	0.13975	0.21	0.00	0.00
13	5	7 → 5	4.472	3.354	5.031	0.13975	0.21	0.12	0.50
14	6	2 → 4	6.325	17.39	26.085	0.7245833333	0.48	0.79	0.90
15	7	4 → 7	2.828	4.243	6.3645	0.1767916667	0.24	0.05	0.42
16	8	4 → 8	2.828	0	0	0	0.00	0.00	0.00
17	9	4 → 3	6.325	20.55	30.825	0.85625	0.52	0.79	0.94
18	10	6 → 8	4.472	0	0	0	0.00	0.00	0.00
19	11	3 → 8	4	-3	-4.5	0.125	0.20	0.00	0.00
20	12	3 → 1	5.657	23.33	34.995	0.9720833333	0.56	0.79	0.92
21	13	2 → 7	4	-1.5	-2.25	0.0625	0.14	0.00	0.00
22	14	7 → 8	4	-3	-4.5	0.125	0.20	0.00	0.00



2				**Total Vol (in ^3)**	**Total Mass (lb)**				
3				**1227.537**	**348.621**				
4									
5									
6	K	L	M	N	O	P	Q	R	X
7	**Radius - Failure (**	**Radius to 1/8th (in)**	**Area(in^2)**	**Volume (in^3)**	**Weight (lb)**	**FOS t**	Pcr	FOS c	**FOS Consoldiated**
8	0.88	1.00	3.14	213.26	60.57	5.92	48.78	2.56	**2.56**
9	0.42	0.50	0.79	56.55	16.06	-2.09	2.71	0.00	**2.09**
10	0.47	0.50	0.79	75.40	21.41	-1.71	1.52	0.00	**1.71**
11	0.47	0.50	0.79	56.55	16.06	-1.71	2.71	0.00	**1.71**
12	0.21	0.25	0.20	10.54	2.99	-2.11	0.30	0.00	**2.11**
13	0.50	0.63	1.23	65.86	18.70	13.17	11.91	3.55	**3.55**
14	0.90	1.00	3.14	238.45	67.72	6.50	39.02	2.24	**2.24**
15	0.42	0.50	0.79	26.65	7.57	6.66	12.20	2.88	**2.88**
16	0.00	0.13	0.05	1.67	0.47	N/A	0.05	0.00	**0.00**
17	0.94	1.00	3.14	238.45	67.72	5.50	39.02	1.90	**1.90**
18	0.00	0.13	0.05	2.63	0.75	N/A	0.02	0.00	**0.00**
19	0.20	0.25	0.20	9.42	2.68	-2.36	0.38	0.00	**2.36**
20	0.92	1.00	3.14	213.26	60.57	4.85	48.78	2.09	**2.09**
21	0.14	0.25	0.20	9.42	2.68	-4.71	0.38	0.00	**4.71**
22	0.20	0.25	0.20	9.42	2.68	-2.36	0.38	0.00	**2.36**

Equations for Excel Columns:

Column	A	B	C
Equation	Member ID	Member Start->End	Length

Column	D	E	F
Equation	Axial Force (kips)	=D8*$E$3	=abs(E8*1000/$C$3)

Column	H	I	J
Equation	=sqrt(F8 / PI())	=PI()/4*L8^4	=if(D8>0,(E8*1000*C8*C8*12^2/(PI()^3*$B$3*1000*0.25))^(1/4),0)

Column	K	L	M
Equation	=max(J8,H8)	=floor(K8, 1/8) + 0.125	=pi()*L8^2

Column	N	O	P
Equation	=C8*pi()*L8^2*12	=N8*0.284	=$C$4/(D8/M8)

Column	Q	R	X
Equation	=(PI()^2 * 29000*I8) / (C8*12)^2	=if(D8>0,Q8/D8,0)	=if(P8<0,-P8,R8)

Meen 305 Project 2: Flower Petal Loadings

By David Wood, Nick Licon, Andres Pinzon, and Ian Wilhite

Section 1: Problem Summary

The client has a flower solar panel of 8 blades. The task provided was to design a truss and column structure that could support the mass of the blades and their loading during usage. The objective of this design is to minimize the mass of the column and truss with a given deflection while keeping reasonable factors of safety.

Section 2: Design Process

The column must not fail due to yielding/fracturing due to stresses, buckling due to applied load, or laterally deflect more than 0.3 in.

To get a radius that would create a maximum deflection of 0.3 in, Equations () were used. Then, the maximum stresses at critical points were calculated on either side of the column to make sure the column would not yield under the load. Additionally, the critical buckling load was calculated so that we could ensure the column would not buckle, since oftentimes a column will buckle before it yields under a compressive load.

There were a number of design constraints applied to ensure valid :

- Lateral deflection not to exceed 0.3 inches.
- Factor of safety on all calculated values of at least 2.
- Constant thickness of the column vertically (as opposed to varied thickness as a function of height) for ease of manufacturing and ability to use existing commercial products.
- Column may be hollow to allow for more efficient use of material.
- All column measurements must be in increments of ¼ inch for ease of manufacturing.

Failure due to Yielding:

The first thing we noticed for the column was the column would experience maximal stress when the solar panels were in the vertical (90°) position. This is because the sum of the panel components acts as a concentrated load 1 ft away from the column's central axis, generating a bending stress. Furthermore, the wind force, which is also modeled as concentrated load, is also highest when the panels are in the vertical position. If the wind force acts in the direction so that it hits the back of the panels, it will generate an additional bending moment and stress in the same direction. To illustrate this, three free body diagrams with different configurations are provided below to display our thought process and logically lead to the highest failure scenario for the column.

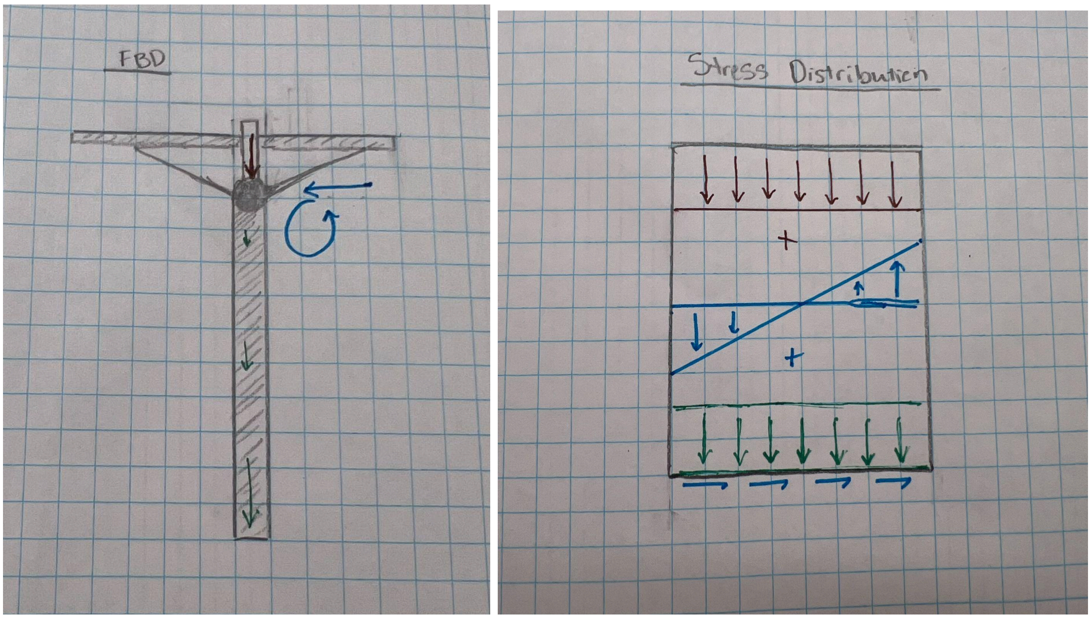


Figure 1. Force/Moment Diagram and Stress Distribution for Flat Position

In this configuration, there is no bending stress generated by the solar panels. Furthermore, the bending stress generated by the wind is smaller because the wind force data is provided and approximated at a lower value of 40 lbs. The column should not be designed to avoid failure under this scenario because there are other configurations that can generate much higher stresses. The same logic applied for the angled (45°) position. There is some bending stress generated by the weight of the solar panels, and the bending stress generated by the wind is also higher too (wind force modeled at 85 lbs), but they are both not as great as the vertical position.

A moment is calculated by multiplying the magnitude of a force and its perpendicular distance from an axis by which it revolves. In the case of the bending stress generated by the solar panels, the *radius* is maximal in the vertical position. For the bending stress generated by the wind, the *force* is maximal in the vertical position. Therefore, we decided to not design the column based on an angled configuration because, logically, there are other configurations that generate higher stresses.

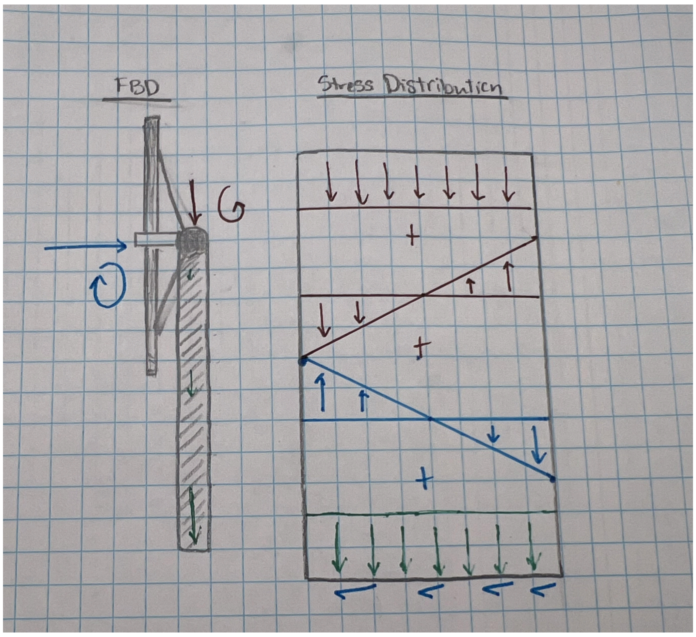


Figure 2. Force/Moment Diagram and Stress Distribution for Oncoming Wind

In this configuration, the bending moments generated by the weight of the solar panels and the wind partially offset each other because they are acting opposite in direction. As seen in **Figure 2**, on the left side of the column, the solar panels generate a maximum compressive stress, while the wind force generates a maximum tensile stress. On the right side of the column, the solar panels generate a tensile stress and the wind force generates a compressive stress. Regardless of where, the bending stresses are always cancelling each other out to some degree.

The last consideration is where the force is applied on the back side of the panels. This would generate much higher stresses because the bending moments would be acting in the same direction, with maximum compression on the left side and maximum tension on the right, as you can see below:

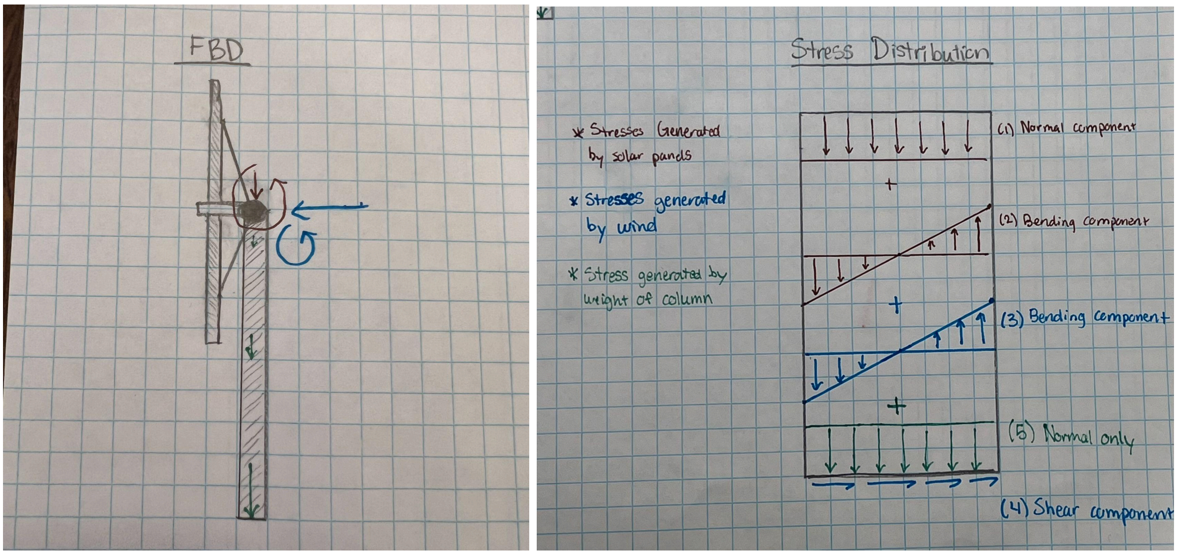


Figure 3. Force/Moment Diagram and Stress Distribution of Highest Failure Scenario

This is the configuration for the column by which the most stress is generated, and the one we need to design around. As you can see, the weight of the solar panels generates a stress that has both normal and bending components, while the wind force generates a stress that has shear and bending components. The weight of the column itself generates stress too, but unlike the other stresses, it varies with the length of the column. The normal stress due to the weight of the column is maximal at the bottom and zero at the top of the column.

The normal stress components have a uniform distribution, meaning that the magnitude and direction of the stress is constant at any individual location within the column's cross-section. Bending stresses, on the other hand, have a linear distribution based on the distance it is from the central axis of the column. The magnitude of this stress is the same in either direction from the center but opposite in direction, depending on whether you are closer or further away from where the moment-inducing force is being applied. On the left side, it generates a maximum compressive stress, while on the right it generates a maximum tensile stress.

The formulas used to calculate all stresses exerted on the column are summarized below:

Normal Stress	Shear Stress	Bending Stress	Torsional Stress
$\sigma = \frac{P}{A}$	$\tau = \frac{VQ}{It}$	$\sigma = -\frac{My}{I}$	$\tau = \frac{T\rho}{J}$

Brittle Materials

Brittle materials have different strengths in the tensile and compressive directions, being much stronger in compression than tension. This rules out our ability to use the **maximum normal stress theory**, which is derived based on the assumption that the material is equally strong in compressive/tensile directions. Instead, we will find the locations in the column where stress is maximum in either compressive or tensile directions, and compare them to the compressive/tensile stresses provided for each material.

We determined that there are two locations on the beam where failure is most likely to occur. Referencing the two-dimensional stress distribution layout in **Figure 3**, these points would be the lower left-hand corner and the upper-right hand corner. As the lower left location, the column is under **maximal compressive stress.** The compressive stresses add upon each other, and, due to the nature of each distribution, they accumulate to a maximum at this location. In the upper right location, the column is potentially under **maximal tensile stress**.

NOTE: The word "potentially" is used because the downward normal components acting on the right side of the column may be greater in magnitude than the upward components due to bending, but this is highly unlikely.

Ductile Materials

There is no data given on compressive/tensile strengths for the ductile materials. This is likely because yielding in ductile materials is dominated by shear mechanisms, and they are not super direction-dependent. Therefore, we will assume that the ductile materials are equally strong in compressive/tensile directions and use the **maximum shear stress theory** (or Tresca criterion) to determine if it will yield under compressive loads. Note that the magnitude of maximum compression at a location will be greater than the maximum tension at another. The yield strength is estimated based on the formula:

$$\tau_{\substack{abs\\ \max}} = \frac{\sigma_1 - \sigma_2}{2} = \frac{\sigma_y}{2}$$

The shear stress generated by the wind is neglected because it is relatively small (see **Appendix B.3**), so there is no need to determine the principal stresses. The maximum stress will simply be the lower-left corner (see **Figure 3)** where it is under maximum compression.

Failure Due to Buckling

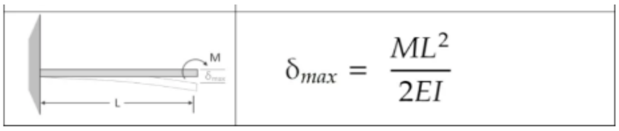
Buckling also had to be taken into consideration as the weight of the column could cause it to fail on itself. This occurs when the forces acting on the column overcome its ability to resist bending, and this depends on its stiffness (bending rigidity, EI), how it's supported, and its length. In this case, with a fixed-free setup, the column's weight itself can create enough stress to make it buckle when it reaches a critical value. It's like a battle between the column's rigidity and the load trying to push it into instability.

$$W_c = 7.84\frac{EI}{L^2}$$

This critical weight required to maintain the integrity of the column did not make much of an impact in the end. Since we were minimizing the weight of the column, our results yielded values that were much lower. Nonetheless, these calculations are important to make as they ensure this specific part of the design is secure and it adds an extra layer of credibility to our final design.

Minimizing Lateral Deflection to less than 0.3 inches

To make sure the column did not deflect more than 0.3 inches there were two scenarios that had to be considered. First we were tasked with finding the most suitable design without any wind forces, so the only thing causing the column to deflect was the bending moment caused by the blades.

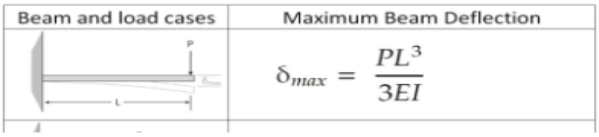


$$\delta_{max} = \frac{ML^2}{2EI}$$

Figure 4. Deflection due to bending stress

Due to this bending effect, the lateral deflection of the column was found using the equation seen in **Figure 4**. A design was then optimized to have a FOS that was less than 2.

Once wind was taken into consideration, another deflection was required to be calculated. This time the column was going to be deflected due to a force that pushes sideways and would make the column bow out over its length.



Beam and load cases	Maximum Beam Deflection
	$\delta_{max} = \frac{PL^3}{3EI}$

Figure 5. Deflection due to Force

With a new input causing a deflection on the column by the amount seen in the equation in **Figure 5**, our FOS changed from our minimum of 2 down to less than 1 for our chosen model. This would mean that the column would deflect more than the expected 3 inches. A new inner and outer radius were constructed to make sure that the sum of the deflections caused by both the wind and the bending would not exceed 3 inches.

Column Conclusions

For the column, it was clear that limiting the deflection to 0.3 inches was the primary limiting factor for design optimization. Calculations of failure due to buckling or yielding/fracturing did not even come close to their determined limits.

Section 2.2 Design Considerations for Truss:

The Truss must not fail due to buckling or normal stress under the load of the panel and must pass with a factor of safety greater than 2. The analysis of the truss problem is a statically indeterminate problem. Connected to a rigid wall there are 3 unknowns due to the cantilever beam-like properties it will have. Additionally, the force of the truss acts upward three feet from the wall. This means that we must use analysis of beam deflection to determine the fourth equation to solve for all four unknowns. To solve the beam deflection problem we will assume the compatibility to be no deflection at the end of the panel meaning that at what the truss is supporting the weight the displacement is 0 (L=3, v = 0).

Once the force acting on the truss is found a test for both brittle materials and ductile materials will be performed as well as for what the radius of the cylindrical solid truss should be.

Position one of the flowers allows us to ignore the truss in that position due to the compatibility of the orientation and the bar. With the weight going straight down the only compatibility for the panel that we know is that the displacement of the bar in the x direction is zero, meaning that the trust can exert any force upon the panel in that position leading no force to be placed on the truss in this position.

Volume = Area * length
Density(lbs/in^3) = Density(lbs/ft^3) / (12^3)

Material	Minimum radius	Volume	Weight
Gray cast iron 30	0.220in	5.7699in^3	1.402lbs
Gray cast iron 60	0.205in	5.01in^3	1.450lbs
Stainless steel 304	0.189in	4.258in^3	0.4189lbs
Aluminum Alloy	0.243in	7.0395in^3	0.387lbs

Table 1. Evaluation of Truss

Aluminum Alloy gives the same amount of support as the others for the minimized weight of the truss making it the best material to use for our given case.

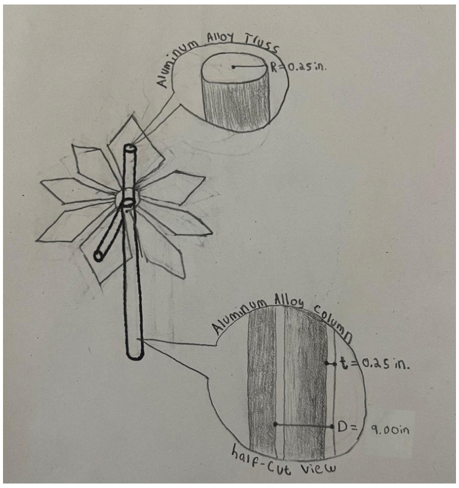


Figure 6. Sketch of Truss6

Section 3: Results

Section 3.1 Calculated Values

Parameter	Value
Maximum Lateral Deflection	0.3 [in]
Flower Weight	40 [lbs]
Column Height	10 [ft]
Number of Flowers	8
Minimum Factor of Safety	2
End condition factor	0.25

Table 2. Assumed parameters

Material	Material Type	Inner Radius (in)	Outer Radius (in)	Density (lb/ft^3)	Mass (lb)	FOS
Cast Iron 30	Brittle	3.75	4	420	177.5	2.127
Cast Iron 60	Brittle	3.5	3.75	500	197.7	2.322
Stainless steel 304	Ductile	3.00	3.25	170	58.0	2.047
Aluminum Alloy	Ductile	4.25	4.50	95	45.3	2.071

Table 3. Evaluation of Column

Section 3.1 Final Recommendations

Section	Material	Mass (lb)
Truss	Aluminum	0.387
Column	Aluminum	45.3
Total		45.7

Table 4. Chosen Masses

To minimize weight, both parts should be made of the aluminum alloy option, and the total structure will have a total weight of 45.7 lbs.

Conclusion

For the column, it was clear that limiting the deflection was the primary limiting factor for design optimization. Calculations of failure due to buckling or yielding/fracturing did not even come close to their determined limits. It could be inferred that in future analysis, this limit could be revisited as a design constraint. For the truss, the main limiting design factor was critical buckling load. The designs proposed meet the specified design constraints, and would perform as expected if constructed. Next steps for the project could include verification of calculated values via simulation, or scale tests to ensure the assumptions provided hold true. The design implemented effectively minimizes the weight of the material of the structure, minimizing the error from excluding their weight while calculating stresses.

Section 4: Member Contributions

The following is a summary of the contributions each member provided to the generation of the design and report presented:

- David Wood: David worked on developing the calculator for the column, specifically for the brittle materials. He also evaluated the columns compressive and tensile strength, and took the lead on debugging the calculator.
- Nick Licon: Nick developed the calculator for the truss for both ductile and brittle materials. He also introduced many of the analysis techniques that were crucial to evaluating the column.
- Andres Pinzon Diaz: Andres primarily worked on understanding the column buckling equations and their applications, then their implementation in the column.
- Ian Wilhite: Ian worked on developing the calculator for the column, specifically for the ductile materials. He also worked on preliminary reporting and approach structure.

Appendix

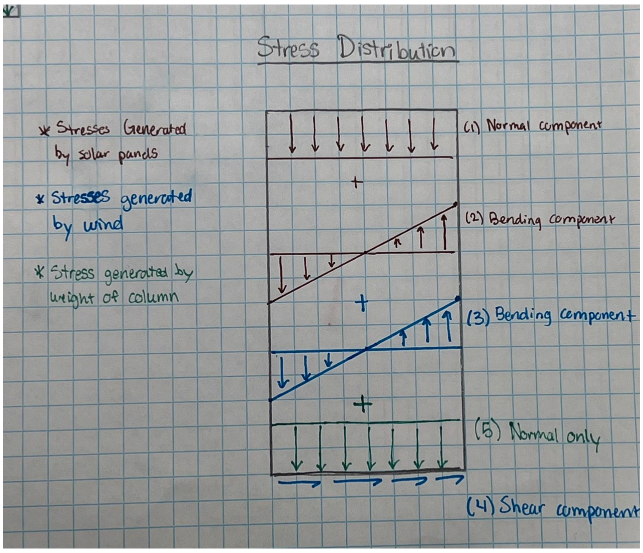
A. Tables

Table 1 Material Properties

Material	Density (lb/ft^3)	Elastic Modulus (ksi)	Tensile Yield Stress (ksi)	Tensile Ultimate Stress (ksi)	Compressive Ultimate Stress (ksi)	Shear Yield or Ultimate Stress (ksi)
Gray cast iron 30	420	15000		30	100	15
Gray cast iron 60	500	20000		60	180	30
Stainless steel 304	170	27500	40			20
Aluminum Alloy	95	10150	43			20

B. Calculations

B.1 Column Stress Analysis and Dimensional Optimization of Brittle Material Options



- Max compressive stress (lower left corner) = $\sigma_{\text{panels, normal}} + \sigma_{\text{panels, bending}} + \sigma_{\text{wind, bending}} + \sigma_{\text{column weight}}$
- Max Tensile Stress (upper right corner) = $\sigma_{\text{panels, normal}} + \sigma_{\text{panels, bending}} + \sigma_{\text{wind, bending}}$

SIGN CONVENTION: Negative sign is used for compression; positive sign is used for tension. So stresses must be greater than the maximum (negative) compressive stress, and less than the maximum (positive) tensile stress

$$\sigma_{max,\,compression} = -\frac{(320\ lbs)}{\pi\left(r_o^2-r_i^2\right)} - \frac{(320\ ft\cdot lbs)\cdot r_o}{\frac{\pi}{4}\left(r_o^4-r_i^4\right)} - \frac{(120lbs\cdot 120in)\cdot r_0}{\frac{\pi}{4}\left(r_o^4-r_i^4\right)} - \frac{\rho\cdot\pi\left(r_o^2-r_i^2\right)(120in)\cdot g}{\pi\left(r_o^2-r_i^2\right)}$$

$$\sigma_{max,\,tension} = -\frac{(320\ lbs)}{\pi\left(r_o^2-r_i^2\right)} + \frac{(320\ ft\cdot lbs)\cdot r_o}{\frac{\pi}{4}\left(r_o^4-r_i^4\right)} + \frac{(120lbs\cdot 120in)\cdot r_0}{\frac{\pi}{4}\left(r_o^4-r_i^4\right)}$$

Gray Cast Iron 30: r_0 = 4.0 in, r_i = 3.75 in

$$\sigma_{max,\,compression} = -\frac{(320\ lbs)}{\pi\left(4^2-3.75^2\right)} - \frac{(320\ ft\cdot lbs)\cdot 4}{\frac{\pi}{4}\left(4^4-3.75^4\right)} - \frac{(120lbs\cdot 120in)\cdot 4}{\frac{\pi}{4}\left(4^4-3.75^4\right)} - \frac{0.243\cdot\pi\left(4^2-3.75^2\right)(120in)\cdot 32.2}{\pi\left(4^2-3.75^2\right)}$$

= **-2.586619605 ksi > -100 ksi;** well within failure limit

**The density was converted to lbs/in^3*

$$\sigma_{max,\ tension} = -\frac{(320\ lbs)}{\pi\left(4^2-3.75^2\right)} + \frac{(320\ ft\cdot lbs)\cdot 4}{\frac{\pi}{4}\left(4^4-3.75^4\right)} + \frac{(120lbs\cdot 120in)\cdot 4}{\frac{\pi}{4}\left(4^4-3.75^4\right)}$$

= **1.542307995 ksi < 30 ksi;** well within failure limit

<u>Gray Cast Iron 60:</u> r_0 = 3.75in, r_i = 3.50 in

$$\sigma_{max,\ compression} = -\frac{(320\ lbs)}{\pi\left(3.75^2-3.5^2\right)} - \frac{(320\ ft\cdot lbs)\cdot 3.75}{\frac{\pi}{4}\left(3.75^4-3.5^4\right)} - \frac{(120lbs\cdot 120in)\cdot 3.75}{\frac{\pi}{4}\left(3.75^4-3.5^4\right)} - \frac{0.289\cdot\pi\left(3.75^2-3.5^2\right)(120in)\cdot 32.2}{\pi\left(3.75^2-3.5^2\right)}$$

= **-3.000360171 ksi > -180 ksi;** well within failure limit

$$\sigma_{max,\ tension} = -\frac{(320\ lbs)}{\pi\left(4^2-3.75^2\right)} + \frac{(320\ ft\cdot lbs)\cdot 4}{\frac{\pi}{4}\left(4^4-3.75^4\right)} + \frac{(120lbs\cdot 120in)\cdot 4}{\frac{\pi}{4}\left(4^4-3.75^4\right)}$$

= **1.769908297 ksi < 60 ksi;** well within failure limit

B.2 Column Stress Analysis and Dimensional Optimization of Ductile Material Options

- Stress, <u>is any direction</u>, must be less than $\sigma_{yield} = 2*\tau_{ultimate}$, per Tresca criterion. σ_{yield} = 2 * 20ksi (for both ductile materials) = **40 ksi.**
- The maximum stress in compression will always be greater than the maximum stress in tension, so there is no need to calculate maximum tension.

<u>Stainless Steel 304:</u> r_0 = 3.25 in, r_i = 3.0 in

$$\sigma_{max,\ compression} = -\frac{(320\ lbs)}{\pi\left(3.25^2-3^2\right)} - \frac{(320\ ft\cdot lbs)\cdot 3.25}{\frac{\pi}{4}\left(3.25^4-3^4\right)} - \frac{(120lbs\cdot 120in)\cdot 3.25}{\frac{\pi}{4}\left(3.25^4-3^4\right)} - \frac{0.0984\cdot\pi\left(3.25^2-3^2\right)(120in)\cdot 32.2}{\pi\left(3.25^2-3^2\right)}$$

= **-2.914629187 ksi > -40 ksi;** well within failure limit

**The density was converted to lbs/in^3*

<u>Aluminum Alloy:</u> r_0 = 4.5 in, r_i = 4.25 in

$$\sigma_{max,\ compression} = -\frac{(320\ lbs)}{\pi\left(4.5^2-4.25^2\right)} - \frac{(320\ ft\cdot lbs)\cdot 4.5}{\frac{\pi}{4}\left(4.5^4-4.25^4\right)} - \frac{(120lbs\cdot 120in)\cdot 4.5}{\frac{\pi}{4}\left(4.5^4-4.25^4\right)} - \frac{0.0549\cdot\pi\left(4.5^2-4.25^2\right)(120in)\cdot 32.2}{\pi\left(4.5^2-4.25^2\right)}$$

= **-1.505973093 > -40 ksi;** well within failure limit

**The density was converted to lbs/in^3*

B.3 Calculations for Shear Stress in Column (why it's neglected)

The shear stress for the cast iron 30 was determined using the formula:

$$\tau_{wind} = \frac{V_{wind}\cdot Q}{It} = \frac{120lbs\cdot\frac{2}{3}\left(r_o^3-r_i^3\right)}{\frac{\pi}{4}\left(r_o^4-r_i^4\right)\cdot 2r_o} = \mathbf{0.001939119528\ ksi.}$$

Since it was so small, we decided to exempt it in our stress analysis for simplicity.

B.4 Calculations for deflection

$M \ = n * w * 1ft * sin\theta$

Where n is the number of petals, w is the weight of each, and theta is the angle it makes with the x axis

$M_{90^{\circ}}$ = 3840 lbm*in

$\boldsymbol{\delta}_{max} = \frac{ML^2}{2EI}$

$\boldsymbol{\delta}_{max} = \frac{PL^3}{3EI}$

Gray Cast Iron 30: r_0 = 4.0 in, r_i = 3.75 in

$\boldsymbol{\delta}_{max} = \frac{3840*120^2}{2*15000*45.74} = 0.0403\ in$

$+ \ \text{wind}$

$\boldsymbol{\delta}_{max} = \frac{120*120^3}{3*15000*45.74} = 0.101 \text{ in}$

= 0.141in

Gray Cast Iron 60: r_0 = 3.75 in, r_i = 3.5 in

$\boldsymbol{\delta}_{max} = \frac{3840*120^2}{2*20000*37.46} = 0.0369\ in$

$+ \ \text{wind}$

$\boldsymbol{\delta}_{max} = \frac{120*120^3}{3*20000*37.46} = 0.0922 \text{ in}$

= 0.129in

Stainless Steel 304: r_0 = 3.25 in, r_i = 3.0 in

$\boldsymbol{\delta}_{max} = \frac{3840*120^2}{2*27500*24.01} = 0.0419\ in$

$+ \ \text{wind}$

$\boldsymbol{\delta}_{max} = \frac{120*120^3}{3*27500*24.01} = 0.105 \text{ in}$

= 0.147in

Aluminum Alloy: r_0 = 4.5 in, r_i = 4.25 in

$\boldsymbol{\delta}_{max} = \frac{3840*120^2}{2*10150*65.82} = 0.0413\ in$

$+ \ \text{wind}$

$\boldsymbol{\delta}_{max} = \frac{120*120^3}{3*10150*85.82} = 0.103 \text{ in}$

= 0.145in

*All below the max deflection that would give us a FOS of 2 (0.15in)

B.5 Calculations for buckling

$W_c \ = \ 7.84\frac{EI}{L^2}$

Gray Cast Iron 30: r_0 = 4.0 in, r_i = 3.75 in

$W_c = 7.84\frac{15000*45.7}{120^2}$

= 373217 lb

Gray Cast Iron 60: r_0 = 4.0 in, r_i = 3.75 in

$W_c = 7.84\frac{20000*37.46}{120^2}$

= 407898 lb

Stainless Steel 304: r_0 = 3.25 in, r_i = 3.0 in

$W_c = 7.84\frac{27500*24.01}{120^2}$

= 359483 lb

Aluminum Alloy: r_0 = 4.5 in, r_i = 4.25 in

$W_c = 7.84\frac{10150*65.82}{120^2}$

= 363729 lb

B.6 Truss Calculations

Use superposition to derive an expression for displacement of the cantilever beam.

Total deflection

| |

Weight of the bar

$$v = \frac{-PL^2}{48EI}(6x - L) \quad L/2 \le x \le L$$

+

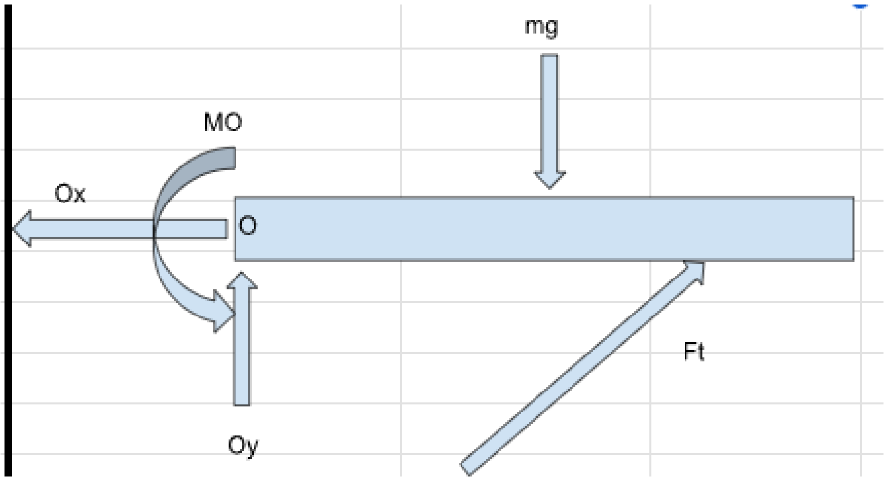
Force of the truss acting upward

$$y = \frac{Px^2}{6EI}(3a - x) \text{ for } 0 < x < a$$

L = 60 in, x = 36in, a = 36in

Find highest force on truss in all three positions:

Ideal position:



Knowns: mg = 40lbs, EI = 145000 kip/in^2

Unknowns: Ox, Oy, Mo, Ft

Compatibility: At x = 36 ; v = 0 (where the truss connects)

$\Sigma Fx \;=\; 0 \;=\; Ox \;+\; Ft\,cos(18.434)$

$\Sigma Fy \;=\; 0 \;=\; Oy \;+\; Ft\,sin(18.434) \;-\; mg$

$\Sigma Mo \;=\; 0 \;=\; Mo \;+\; Ft\,sin(18.434) \cdot 36in \;-\; mg \cdot 30in$

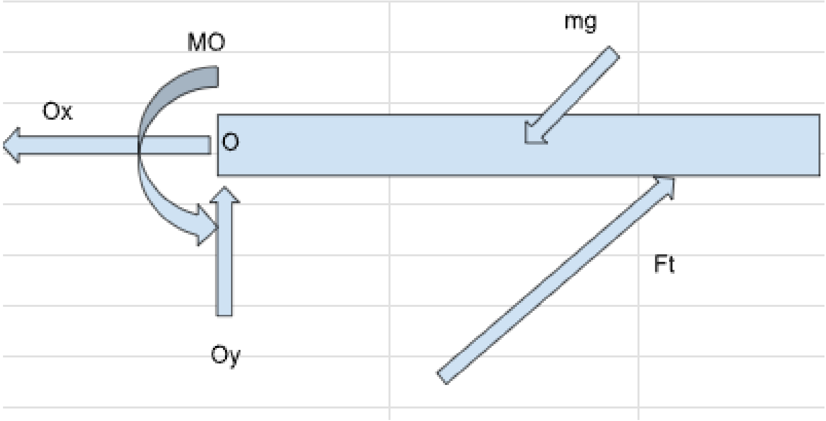
$$v \ = \ \frac{1}{EI}(\frac{mg\cdot(60in)^2}{48}(6 \cdot 36in - 60in) - \frac{Ftsin(18.434)\cdot(36in)^2}{6}(3 \cdot 36in - 36in))$$

Find Ft with v = 0

$$0 \ = \ (\frac{(40)\cdot(60in)^2}{48}(6 \cdot 36in - 60in) - \frac{Ftsin(18.434)\cdot(36in)^2}{6}(3 \cdot 36in - 36in))$$

$$Ft \ = \ \frac{\frac{(40)\cdot(60in)^2}{48}(6\cdot36in-60in)}{\frac{sin(18.434)\cdot(36in)^2}{6}(3\cdot36in-36in)} \ = 95.161lbs$$

Top side of position 2:



Knowns: mg = 40lbs, EI = 145000 kip/in^2

Unknowns : Ox, Oy, Mo, Ft

Compatibility: At x = 36 ; v = 0 (where the truss connects)

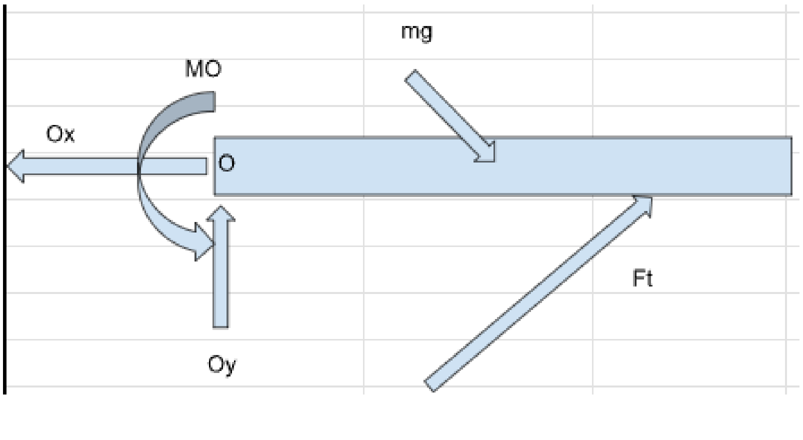
$$\Sigma Fx \ = \ 0 \ = \ Ox \ + Ft\, cos(18.434) \ - \ mgcos(45)$$

$$\Sigma Fy \ = \ 0 \ = \ Oy \ + Ft\, sin(18.434) \ - \ mgsin(45)$$

$$\Sigma Mo \ = \ 0 \ = \ Mo \ + Ft\, sin(18.434) \ \cdot \ 36in \ - \ mgsin(45) \ \cdot 30in$$

$$v \ = \ \frac{1}{EI}(\frac{mgsin(45)\cdot(60in)^2}{48}(6 \cdot 36in - 60in) - \frac{Ftsin(18.434)\cdot(3in)6^2}{6}(3 \cdot 36in - 36in))$$

Bottom side of position 2



Knowns: mg = 40lbs, EI = 145000 kip/in^2
Unknowns : Ox, Oy, Mo, Ft
Compatibility: At x = 36 ; v = 0 (where the truss connects)

$$\Sigma Fx \ = \ 0 \ = \ Ox \ + Ft\,cos(18.434) \ + \ mgcos(45)$$

$$\Sigma Fy \ = \ 0 \ = \ Oy + Ft\,sin(18.434) \ - \ mgsin(45)$$

$$\Sigma Mo \ = \ 0 \ = \ Mo + Ft\,sin(18.434) \ \cdot \ 36in - \ mgsin(45) \ \cdot 30in$$

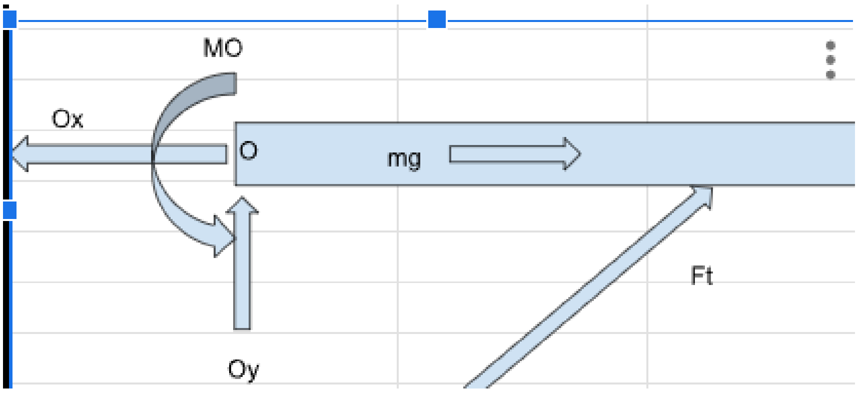
$$v \ = \ \frac{1}{EI}(\frac{mgsin(45)\cdot(6in)0^2}{48}(6 \cdot 36in - 60in) \ - \frac{Ftsin(18.434)\cdot(3in)6^2}{6}(3 \cdot 36in - 36in))$$

V equation same for both to find Ft
Find Ft with v = 0

$$0 \ = \ (\frac{(40sin(45))\cdot(60in)^2}{48}(6 \cdot 36in - 60in) \ - \frac{Ftsin(18.434)\cdot(3in)6^2}{6}(3 \cdot 36in - 36in))$$

$$Ft \ = \ \frac{\frac{(40sin(45))\cdot(60in)^2}{48}(6\cdot36in-60in)}{\frac{sin(18.434)\cdot(3in)6^2}{6}(3\cdot36in-36in)} \ = 67.289lbs$$

Vertical position



Knowns: mg = 40lbs, EI = 145000 kip/in^2
Unknowns: Ox, Oy, Mo, Ft
Compatibility: At x = 36in ; v = 0 (where the truss connects)

$$\Sigma Fx \ = \ 0 \ = \ Ox \ + Ft\,cos(18.434) \ + \ mg$$

$$\Sigma Fy \ = \ 0 \ = \ Oy + Ft\,sin(18.434)$$

$$\Sigma Mo \ = \ 0 \ = \ Mo + Ft\,sin(18.434) \ \cdot \ 36\ in$$

$$v \ = \ \frac{1}{EI}(- \frac{Ftsin(18.434)\cdot(3in)6^2}{6}(3 \cdot 36in - 36in))$$

When v = 0

$$0 \ = \ \frac{1}{EI}(- \frac{Ftsin(18.434)\cdot36^2}{6}(3 \cdot 36 - 36))$$

Ft must be 0

Since position 3 has the most force on it it will fail first so we will test materials and design in that orientation

Ft = 95.161lbs

Calculate radius of the truss for the force applied with all 4 materials

$P_{cr} = \frac{\pi^2 EI}{L^2}$

$\sigma = \frac{P}{A}$

$A = \pi r^2$

$L = 3.162ft = 37.947in$

$Pcr = F * F.O.S$

$Pcr = \frac{\pi\cdot(EI)}{(L)^2}$

$I = \frac{\pi r^4}{4}$

Brittle:

Gray cast iron 30

EI = 15000 ksi

$I = \frac{\pi r^4}{4}$

σult = 100 ksi

$\sigma allow$ = 50 ksi

$\sigma allow = F/\pi r^2$

$r = \sqrt{\frac{95.161lbs}{50000\ lbs/in^2\cdot\pi}} = 0.0246in$

$Pcr = F * F.O.S = 190.322lbs$

$Pcr = \frac{\pi\cdot(EI)}{(L)^2}$

$Pcr = \frac{\pi\cdot(15000000\ lbs/in^2\cdot\frac{\pi r^4}{4})}{(37.947in)^2}$

$r = \sqrt[4]{\frac{4\cdot(37.947in)^2\cdot190.322lbs}{\pi^3\cdot15000000\ lbs/in^2}} = 0.220in$

Gray cast iron 60

EI = 20000 ksi

σult = 180 ksi

$\sigma allow$ = 90 ksi

$\sigma allow = F/\pi r^2$

$r = \sqrt{\frac{95.161lbs}{90000\ lbs/in^2\cdot\pi}} = 0.0183in$

$Pcr = F * F.O.S = 190.322lbs$

$Pcr = \frac{\pi\cdot(EI)}{(L)^2}$

$Pcr = \frac{\pi\cdot(20000000\ lbs/in^2\cdot\frac{\pi r^4}{4})}{(37.947in)^2}$

$r = \sqrt[4]{\frac{4\cdot(37.947in)^2\cdot190.322lbs}{\pi^3\cdot20000000\ lbs/in^2}} = 0.205in$

Ductile:

Stainless steel 304

EI = 27500 ksi

$\sigma yield$ = 40 ksi

$\sigma allow$ = 20 ksi

$$\sigma allow = F/\pi r^2$$

$$r = \sqrt{\frac{95.161 lbs}{20000\ lbs/in^2 \cdot \pi}} = 0.0389in$$

$$Pcr = F * F.O.S = 190.322lbs$$

$$Pcr = \frac{\pi \cdot (EI)}{(L)^2}$$

$$Pcr = \frac{\pi \cdot (\ 27500000\ lbs/in^2 \cdot \frac{\pi r^4}{4})}{(37.947in)^2}$$

$$r = \sqrt[4]{\frac{4 \cdot (37.947in)^2 \cdot 190.322lbs}{\pi^3 \cdot 27500000\ lbs/in^2}} = 0.189in$$

Aluminum Alloy

EI = 10150 ksi

$\sigma yield$ = 43ksi

$\sigma allow$ = 21.5ksi

$$\sigma allow = F/\pi r^2$$

$$r = \sqrt{\frac{95.161 lbs}{21500\ lbs/in^2 \cdot \pi}} = 0.375in$$

$$Pcr = F * F.O.S = 190.322lbs$$

$$Pcr = \frac{\pi \cdot (EI)}{(L)^2}$$

$$Pcr = \frac{\pi \cdot (\ 10150000\ lbs/in^2 \cdot \frac{\pi r^4}{4})}{(37.947in)^2}$$

$$r = \sqrt[4]{\frac{4 \cdot (37.947in)^2 \cdot 190.322lbs}{\pi^3 \cdot 10150000\ lbs/in^2}} = 0.243in$$

Back to Contents

7. NASA L'SPACE Mission Concept Academy (MCA)

Summary

Through the NASA Lucy Student Pipeline Accelerator and Competency Enabler (L'SPACE) Mission Concept Academy (MCA) I have had the incredible privelige to serve as the Project Manager (PM) and lead a group of 20 scientists, engineers, and programmatics specialists across the country to develop a NASA Discovery-Class mission from scratch. We propose PHOENIX, an unmanned rover mission to Mars to support future human exploration. The links to the four deliverables are below, howver the Preliminary Design Review (PDR) document contains the full mission concept.

- Mission Concept Review (MCR)
- System Requirements Review (SRR)
- Mission Design Review (MDR)
- Preliminary Design Review (PDR)

Contributors: Mission Concept Academy Summer 2025 Team 01

Key Skills: Systems Engineering, Mission Design, NASA Project Lifecycle, System Requirements, Technical Writing, Team Collaboration.

Relevance: This experience provides direct insight into NASA's mission development process, how scientific needs drive system design, how programmatics enable their conduction. In addition to serving as the Project Manager, I also worked heavily on the Command and Data Handling (CDH) Engineering Subsystem.

Back to Contents

8. Subsea Remote Operated Vehicle (ROV) Capstone

Summary

The Interdisciplinary Engineering Capstone course (ITDE 401-402) encapsulated a year-long project to design, build, and test a subsea Remote Operated Vehicle (ROV). Our team is developing a functional prototype capable of performing underwater tasks, where I work on electrical systems inclduing the power subsystem, the telemtry subsystem, and lead the control software design.

- Four Thruster Gimbal Control Strategy Analysis
- Spherical Fixed Thruster Control Strategy Analysis

Contributors: Hudson Hurtig, Abraham Rice, Joshua Mendez, Juan Lopez, Luiz Lopez, Ian Wilhite

Key Skills: Systems Engineering, Top-down design, Procurement, Assembly, Integration, System Requirements, Project Engineering, Technical Writing, Team Collaboration.

Relevance: This was an end-to-end in-house engineering project where I developed the electrical systems and software control strategy.

ROV Four-Thruster Gimbaled Control Basis

Ian Wilhite

September 29, 2025

1 Introduction

The current proposed ROV design implements four thrusters and four gimbaled connections. On a mobile platform operating in six Degrees Of Freedom (6-DOF), this presents a system that is inherently underactuated. However, underactuated control is possible and is fundamentally limited by non-holonomic thruster constraints, gimbal angular rates, and reactionary moments from thrust vectoring.

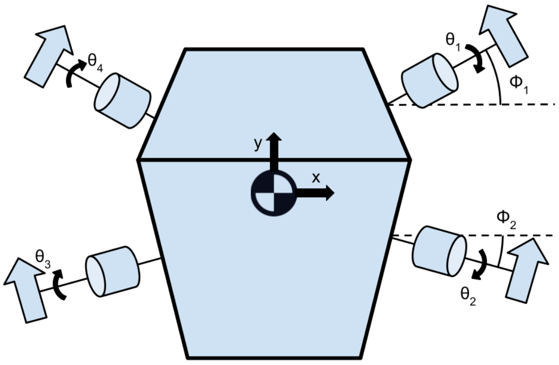


Figure 1: 4-Thruster 4-Gimbal ROV Convention

1.1 Modeling Assumptions

The model shown in Figure 1 presents the four-thruster convention with the origin set at the center of mass, x to the starboard side of the vehicle, y to the front of the vehicle, and z out of the page. Roll is measured around the y axis starting from x, pitch is measured around the x axis starting at y, and yaw is measured around the z axis starting at y.

To effectively compare the controllability of various designs, it is necessary to assume that

2 Dynamics and Control

2.1 System Reactions

The system dynamics can be found by finding the component of each input force applied in each component of the coordinate system. This serves to determine how each degree of freedom of the system is affected by each input force.

$$F_x = \sin\phi_1(F_4\cos\theta_4 - F_1\cos\theta_1) + \sin\phi_2(F_2\cos\theta_2 - F_3\cos\theta_3)\,, \tag{1}$$

$$F_y = \cos\phi_1(F_4\cos\theta_4 + F_1\cos\theta_1) + \cos\phi_2(F_2\cos\theta_2 + F_3\cos\theta_3)\,, \tag{2}$$

$$F_z = F_1\sin\theta_1 + F_2\sin\theta_2 + F_3\sin\theta_3 + F_4\sin\theta_4. \tag{3}$$

Moments are taken with respect to the center of mass of the vehicle using the coordinate (x_i, y_i) for each thruster to find the moment arm. In the 3D extension of the planar case, it is assumed that all forces are acting in the xy-plane of the center of mass.

$$M_x = F_1 \sin\theta_1\, y_1 + F_2 \sin\theta_2\, y_2 + F_3 \sin\theta_3\, y_3 + F_4 \sin\theta_4\, y_4, \tag{4}$$

$$M_y = -F_1 \sin\theta_1\, x_1 - F_2 \sin\theta_2\, x_2 - F_3 \sin\theta_3\, x_3 - F_4 \sin\theta_4\, x_4, \tag{5}$$

$$M_z = -k_{z,1}\, F_1 \cos\theta_1 + k_{z,2}\, F_2 \cos\theta_2 - k_{z,3}\, F_3 \cos\theta_3 + k_{z,4}\, F_4 \cos\theta_4, \tag{6}$$

$$k_{z,i} \;\equiv\; \rho_i \, \cos\big(\operatorname{atan2}(y_i, x_i) - \phi_j\big), \qquad \rho_i = \sqrt{x_i^2 + y_i^2},$$

where $j = 1$ for $i \in \{1, 4\}$ and $j = 2$ for $i \in \{2, 3\}$.

2.2 Control Basis Vectors

The components of the system reaction can be combined to find the wrench of the system applied with respect to each of the input forces. A wrench is a combined column vector of the forces and moments applied in a coordinate system. This 6x4 matrix represents the control basis of the system. The columns of this matrix are the basis vectors, which represent the wrench applied to the system for each input force.

$$[w] = \begin{bmatrix} F \\ \tau \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \underbrace{\begin{bmatrix} -\sin\phi_1 \cos\theta_1 & \sin\phi_2 \cos\theta_2 & -\sin\phi_2 \cos\theta_3 & \sin\phi_1 \cos\theta_4 \\ \cos\phi_1 \cos\theta_1 & \cos\phi_2 \cos\theta_2 & \cos\phi_2 \cos\theta_3 & \cos\phi_1 \cos\theta_4 \\ \sin\theta_1 & \sin\theta_2 & \sin\theta_3 & \sin\theta_4 \\ y_1 \sin\theta_1 & y_2 \sin\theta_2 & y_3 \sin\theta_3 & y_4 \sin\theta_4 \\ -x_1 \sin\theta_1 & -x_2 \sin\theta_2 & -x_3 \sin\theta_3 & -x_4 \sin\theta_4 \\ -k_{z,1} \cos\theta_1 & k_{z,2} \cos\theta_2 & -k_{z,3} \cos\theta_3 & k_{z,4} \cos\theta_4 \end{bmatrix}}_{B} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}.$$

$$k_{z,i} \;\equiv\; \rho_i \, \cos\big(\operatorname{atan2}(y_i, x_i) - \phi_j\big), \qquad \rho_i = \sqrt{x_i^2 + y_i^2},$$

2.3 Verification

$$\mathbf{b}_1 = \begin{bmatrix} \cos\theta_1 \cos\phi_1 \\ \cos\theta_1 \sin\phi_1 \\ \sin\theta_1 \end{bmatrix}.$$

$$\begin{aligned} \|\mathbf{b}_1\|^2 &= (\cos\theta_1 \cos\phi_1)^2 + (\cos\theta_1 \sin\phi_1)^2 + (\sin\theta_1)^2 \\ &= \cos^2\theta_1 \big(\cos^2\phi_1 + \sin^2\phi_1\big) + \sin^2\theta_1 \\ &= \cos^2\theta_1 \cdot 1 + \sin^2\theta_1 = \cos^2\theta_1 + \sin^2\theta_1 = 1. \end{aligned}$$

$$\therefore \quad \|\mathbf{b}_1\| = 1.$$

3 Conclusion – Underactuated Control

The four-thruster, four-gimbal system operates in 8DOF with four controllable basis vectors. This provides the system with full range of motion *if and only if the gimbals can reach any orientation.* Hardware implementation presents concerns about feasibility of the system practically.

Thruster Placement, Transformations, and Differentiable Evaluation for 6-DoF Wrench Authority

Ian Wilhite

September 29, 2025

1 Introduction

We consider N body-fixed thrusters mounted on or near a reference sphere of radius ρ. Each thruster produces a force along its own local axis, and we seek to (a) map these forces to a global wrench about the origin and (b) optimize placement/orientation for strong, well-conditioned 6-DoF authority. This document standardizes frames, derives the required transformations, builds the allocation matrix, and proposes a differentiable evaluation function for gradient-based design.

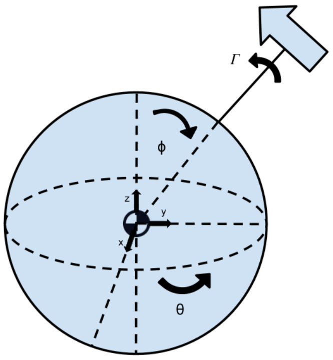


Figure 1: non-Gimbaled Thruster ROV Convention

2 Frames, Coordinates, and Conventions

Base frame. The global Cartesian frame $\{B\}$ has orthonormal axes (x, y, z) about the origin.

Spherical coordinates. We adopt (ρ, θ, ϕ) with azimuth θ about $+z$ from $+x$, and polar angle ϕ measured from $+z$ (north pole). The radial unit vector and tangent basis at (θ, ϕ) are

$$\boldsymbol{e}_r = \begin{bmatrix} \mathsf{c}_\theta \\ \mathsf{s}_\theta \\ c_\phi \end{bmatrix}, \quad \boldsymbol{e}_\theta = \begin{bmatrix} -s_\theta \\ c_\theta \\ 0 \end{bmatrix}, \quad \boldsymbol{e}_\phi = \begin{bmatrix} c_\phi c_\theta \\ c_\phi s_\theta \\ - \end{bmatrix}, \qquad (\boldsymbol{e}_\theta \times \boldsymbol{e}_\phi = \boldsymbol{e}_r). \tag{1}$$

3 Cartesian → Spherical: Coordinates and Basis

Point coordinates (nonlinear). For a Cartesian point $\boldsymbol{p}_B = (x, y, z)^\top$,

$$\rho = \sqrt{x^2 + y^2 + z^2}, \qquad \theta = \operatorname{atan2}(y, x), \qquad \phi = \arccos\left(\frac{z}{\rho}\right) \ (\rho > 0). \tag{2}$$

Basis change for vectors (linear). At (θ, ϕ),

$$\begin{bmatrix} F_r \\ F_\theta \\ F_\phi \end{bmatrix} = \underbrace{\begin{bmatrix} \mathsf{c}_\theta & \mathsf{s}_\theta & c_\phi \\ -s_\theta & c_\theta & 0 \\ c_\phi c_\theta & c_\phi s_\theta & - \end{bmatrix}}_{\boldsymbol{C}_{B\to\mathrm{sph}}(\theta,\phi)} \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}, \qquad \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{e}_r & \boldsymbol{e}_\theta & \boldsymbol{e}_\phi \end{bmatrix}}_{\boldsymbol{C}_{\mathrm{sph}\to B}} \begin{bmatrix} F_r \\ F_\theta \\ F_\phi \end{bmatrix}. \tag{3}$$

4 Surface and Thruster Transforms

Define the *surface frame* $\{S\}$ at position $\rho\,\boldsymbol{e}_r$ with axes $(x_s, y_s, z_s) = (\boldsymbol{e}_\theta, \boldsymbol{e}_\phi, \boldsymbol{e}_r)$. The rotation and translation from $\{B\}$ to $\{S\}$ are

$$\boldsymbol{R}_{BS} = \begin{bmatrix} \boldsymbol{e}_\theta & \boldsymbol{e}_\phi & \boldsymbol{e}_r \end{bmatrix} = \begin{bmatrix} -s_\theta & c_\phi c_\theta & \mathsf{c}_\theta \\ c_\theta & c_\phi s_\theta & \mathsf{s}_\theta \\ 0 & - & c_\phi \end{bmatrix}, \qquad \boldsymbol{p}_{BS} = \rho\,\boldsymbol{e}_r. \tag{4}$$

The homogeneous transform is

$$\boldsymbol{T}_{BS} = \begin{bmatrix} \boldsymbol{R}_{BS} & \boldsymbol{p}_{BS} \\ \boldsymbol{0}^\top & 1 \end{bmatrix}. \tag{5}$$

Let the *thruster frame* $\{T\}$ be obtained by a rotation γ about $z_s = \boldsymbol{e}_r$ within the tangent plane, then a standoff d along $+z_s$:

$$\boldsymbol{R}_{ST}(\gamma) = \begin{bmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{p}_{ST} = d \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_{\{S\}}. \tag{6}$$

The cumulative transform is

$$\boldsymbol{T}_{BT} = \boldsymbol{T}_{BS}\,\boldsymbol{T}_{ST}, \qquad \boldsymbol{R}_{BT} = \boldsymbol{R}_{BS}\boldsymbol{R}_{ST}, \qquad \boldsymbol{p}_{BT} = \boldsymbol{p}_{BS} + \boldsymbol{R}_{BS}\boldsymbol{p}_{ST} = (\rho + d)\,\boldsymbol{e}_r. \tag{7}$$

5 Force and Wrench Mapping

A pure force is rotated but not translated:

$$\boldsymbol{F}_B = \underbrace{\boldsymbol{R}_{BT}}_{\boldsymbol{M}_F(\theta,\phi,\gamma)} \boldsymbol{F}_T. \tag{8}$$

Carrying out the product $\boldsymbol{R}_{BS}\boldsymbol{R}_{ST}$ yields

$$\boldsymbol{M}_F(\theta, \phi, \gamma) = \begin{bmatrix} -c_\gamma\, s_\theta + s_\gamma\, c_\phi\, c_\theta & s_\gamma\, s_\theta + c_\gamma\, c_\phi\, c_\theta & \mathsf{c}_\theta \\ c_\gamma\, c_\theta + s_\gamma\, c_\phi\, s_\theta & -s_\gamma\, c_\theta + c_\gamma\, c_\phi\, s_\theta & \mathsf{s}_\theta \\ -s_\gamma & -c_\gamma & c_\phi \end{bmatrix}. \tag{9}$$

If the thruster produces a wrench $(\boldsymbol{F}_T, \boldsymbol{\tau}_T)$ at $\{T\}$, the base wrench is

$$\begin{bmatrix} \boldsymbol{F}_B \\ \boldsymbol{\tau}_B \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{R}_{BT} & \boldsymbol{0} \\ \boldsymbol{p}_{BT} mu \boldsymbol{R}_{BT} & \boldsymbol{R}_{BT} \end{bmatrix}}_{\boldsymbol{A}^*(\rho,\theta,\phi,\gamma,d)} \begin{bmatrix} \boldsymbol{F}_T \\ \boldsymbol{\tau}_T \end{bmatrix}, \tag{10}$$

where $\boldsymbol{p} muisthecross - productmatrix.$

6 Thruster Wrench Basis ($6 \times N$ Allocation)

For thruster i with parameters $(\rho_i, \theta_i, \phi_i, \gamma_i, d_i)$, position $\boldsymbol{r}_i = (\rho_i + d_i)\,\boldsymbol{e}_{r_i}$ and unit tangential direction $\hat{\boldsymbol{f}}_i = \cos\gamma_i\,\boldsymbol{e}_{\theta_i} + \sin\gamma_i\,\boldsymbol{e}_{\phi_i}$, define the unit wrench

$$\boldsymbol{b}_i = \begin{bmatrix} \hat{\boldsymbol{f}}_i \\ \boldsymbol{r}_i \times \hat{\boldsymbol{f}}_i \end{bmatrix} \in \mathbb{R}^6. \tag{11}$$

Stacking columns yields the allocation matrix

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_N \end{bmatrix} \in \mathbb{R}^{6\times N}, \qquad \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{\tau} \end{bmatrix} = \boldsymbol{B}\,\boldsymbol{u}, \tag{12}$$

where $\boldsymbol{u} \in \mathbb{R}^N$ are thrust magnitudes (signed if bidirectional).

7 Differentiable Evaluation of the Basis

For forces (N) and torques (N m) we adopt a balancing length $R_{\text{char}} > 0$ and define

$$\boldsymbol{W} = \operatorname{diag}\Big(1, 1, 1, \tfrac{1}{R_{\text{char}}}, \tfrac{1}{R_{\text{char}}}, \tfrac{1}{R_{\text{char}}}\Big), \qquad \widetilde{\boldsymbol{B}} = \boldsymbol{W}\,\boldsymbol{B}, \qquad \boldsymbol{M} = \widetilde{\boldsymbol{B}}\,\widetilde{\boldsymbol{B}}^\top. \tag{13}$$

With a small $\varepsilon > 0$, we combine three smooth terms into a maximization objective:

$$J_{\text{vol}}(\boldsymbol{B}) = \log\det\big(\boldsymbol{M} + \varepsilon \boldsymbol{I}_6\big), \tag{14}$$

$$P_{\text{iso}}(\boldsymbol{B}) = \Big\| \boldsymbol{M} - \tfrac{\operatorname{tr}(\boldsymbol{M})}{6}\boldsymbol{I}_6 \Big\|_F^2, \tag{15}$$

$$P_{\text{coh}}(\boldsymbol{B}) = \sum_{i \neq j} \left(\frac{\widetilde{\boldsymbol{b}}_i^\top \widetilde{\boldsymbol{b}}_j}{\|\widetilde{\boldsymbol{b}}_i\|\,\|\widetilde{\boldsymbol{b}}_j\|} \right)^2, \tag{16}$$

and the combined score

$$\boxed{J(\boldsymbol{B}) = \alpha\, J_{\text{vol}}(\boldsymbol{B}) - \beta\, P_{\text{iso}}(\boldsymbol{B}) - \gamma\, P_{\text{coh}}(\boldsymbol{B}),} \tag{17}$$

with weights $\alpha, \beta, \gamma > 0$. The gradient of J_{vol} admits the closed form

$$\frac{\partial J_{\text{vol}}}{\partial \widetilde{\boldsymbol{B}}} = 2\big(\boldsymbol{M} + \varepsilon \boldsymbol{I}_6\big)^{-1}\widetilde{\boldsymbol{B}}, \qquad \frac{\partial J_{\text{vol}}}{\partial \boldsymbol{B}} = \boldsymbol{W}^\top \frac{\partial J_{\text{vol}}}{\partial \widetilde{\boldsymbol{B}}}. \tag{18}$$

8 Optimization Statement and Practical Constraints

We optimize angular parameters $\{\phi_i, \theta_i, \gamma_i\}_{i=1}^{N}$ (and optionally d_i) subject to bounds and collision/clearance constraints:

$$\max_{\{\phi_i,\theta_i,\gamma_i\}} \quad J(\boldsymbol{B}(\{\phi_i, \theta_i, \gamma_i\})) \tag{19}$$

$$\text{s.t.} \quad \phi_i \in [0, \pi],\ \theta_i \in [0, 2\pi),\ \gamma_i \in [0, 2\pi), \tag{20}$$

$$d_i \in [d_{\min}, d_{\max}],\ \text{clearances, FOV, wiring, and mechanical limits.} \tag{21}$$

Choose R_{char} as a representative arm length, e.g. mean $(\rho_i + d_i)$.

9 Conclusion

This narrative consolidates geometry, transforms, wrench mapping, the allocation matrix, and a differentiable objective suitable for gradient-based thruster placement. It is intended to be self-contained and implementation-ready.