CLEMSON UNIVERSITY
Department of Mechanical Engineering

MEGN8710
Fall 2021
Final Exam

SCORE: ____________________/100 NAME: Vinayak Khade

DUE: 5PM December 10, 2021 via Canvas

ACADEMIC TEST MATERIAL

Instructions:
The following pages contain a description of the problem for the ME8710 final exam. The purpose of the exam is to assess you understanding of how to perform a real-world optimization study on a difficult, nonlinear, multiple objective, multimodal problem. You will be evaluated on the feasibility of the solution you propose, your ability to deal with the complexity of the problem, to justify your choice of approach using the methods you have encountered in ME8710, and your ability to explain and justify your result. As with many real problems, there is not a single answer. So, how you choose to solve the problem and how you explain and justify your approach are very important. Nor is it expected that every method taught in ME8710 will be useful in this problem.

This examination is **NOT** released from academic security until 5:00pm on Sunday, December 12, 2021. I agree not to reveal its contents to, or discuss it with, anyone but my instructor until then. ***Furthermore, I acknowledge that all work in this exam is my own and I have not sought, offered or received help from anyone else.*** By signing below, I agree to follow by the instructions of the exam, and that this is my own academic work.

Signature: ____________________

Content

Problem Background

Problem Description: Determine the best configuration for a single rocket that can meet the given requirements.

Information sources/files:

- Simulation for design evaluation: RocketSim.exe (Given)
- Data files: Rocket.txt, Batch.txt, Telemetry.txt, RocketData.txt (Given)
- Source files: Menu.cpp, FlightModels.cpp, RocketSim.exe
- Other Sources: Google Scholar, Wikipedia, etc.

Requirements:

- Given tasks:
  - Payload: 20,000 – 25000 kg
  - Target Velocity: 15, 565 m/s
  - Tolerance: +/- 100 m/s
  - Maximum Dynamic Pressure: 33,440
  - Maximum Acceleration: 8g
- Possible Configuration Considerations:
  - Number of stages (1-5)
  - Number of boosters (0+)
  - Type of booster (32 choices)
  - Type of First Stage Engines (46 choices)
  - Number of First Stage Engines (up to 50)
  - Type of Second, Third, Fourth and Fifth Stage Engines (40 choices of Engine)
  - Number of Second, Third, Fourth and Fifth Stage Engines
  - Throttle Profile of First Stage (and Boosters)
  - Burn Durations of the Second, Third, Fourth and Fifth Stages
  - Diameter of the Main Rocket Core
  - Mass of the Payload

Considerations Made (Constraints & Criteria):

1. Engines chosen based on the Manufacturer's location
   Since collaboration with various stakeholders would be important in the success of this project, to reduce possible conflicts arising due to collaborations over different countries, a consideration was made that **only those engines will be considered which are manufactured in the USA**. This reduced the total engine choices to 42 from 128 of them.

2. Initial values for rocket configuration made based on literature
   To select initial set of values, configurations of heavy-lift launch vehicle were referred from three different sources [1, 2, 3].

Version	CZ-5[1]	CZ-5-522[1]	Proposed[3]	Proposed[2]
No. of boosters	6	6	2	2
No. of 1st stage engines	2	2	3	2
No. of 2nd stage engines	2	2	3	3
No. of 3rd stage engines	Optional	-	-	-
Payload (kg)	14000	25000	25000	133568
Isp- 1st, 2nd, booster, 3rd	430s, 442s, 335s, -	-	277s, 452.5s, 242s, -	250s, 450s, 266s, -

Based on the above table, initial configuration was chosen, the type of engine was chosen based on the Isp values of the table above.

- Number of stages: 2
- Number of boosters: 4
- Type of booster: Northrup GEM63XL
- Number of 1st stage engines: 3
- Type of 1st stage engine: Blue Origin BE-4
- Number of 2nd stage engines: 2
- Type of 2nd stage engine: Blue Origin BE-3U

3. Diameter calculation done based on circle packing
   The Main stage diameter was calculated based on the packing problem 'Circle Packing in a Circle'.
   Main stage diameter = 1.1*1st stage engine diameter*CP

No. of unit circles	CP
1	1
2	2
3	2.154
…	…
19	4.863
20	5.122

Values from 21 to 50 were extrapolated based on the above table.

References:

- [1] https://en.wikipedia.org/wiki/Long_March_5
- [2] Ritter, Paul Andreas, "Optimization and Design for Heavy Lift Launch Vehicles. " Master's Thesis, University of Tennessee, 2012. https://trace.tennessee.edu/utk_gradthes/1197
- [3] Alber, Irwin E.. Aerospace Engineering on the Back of an Envelope. Germany, Springer Berlin Heidelberg, 2012. [Chapter 3]
- [4] https://en.wikipedia.org/wiki/Circle_packing_in_a_circle

Methodology

Methodology used here is based on the Simulated Annealing (SA) method. SAs are often used when the solution spaces are bit discrete. In this problem, the throttle map range is slightly discrete, and selection of type and configuration numbers are discrete. Thus, using SA could yield acceptable improvements. Methodology can be summarized in these steps,

1. Based on the given data and initial literature study, choose initial set of configurations
2. A code was generated on Python3.9 (Jupyter Notebook) for optimization of Throttle map timing based on the maximum velocity achieved. Considerations for other performance tasks were also made in the program. Here's how I planned the code to work,

    a. Read current batch file and setup initial variables
    b. Using initial variables run evaluation simulation, RocketSim.exe
    c. Read the result values from the Telemetry.txt file
    d. Define a function that takes the initial variables and result values
    e. Generate a random step from initial values based on the step size
    f. Evaluate the results of the step using RocketSim.exe
    g. Check if max. dynamic pressure <= 33440
    h. Check if acceleration <= 80
    i. Check if max. velocity is between 15456-15656 m/s
    j. If true, add to accepted models
    k. Check if max. velocity is between 11500-17000m/s
    l. If true, add to workable models
    m. Make the step as current/initial values
    n. Else, calculate delta = difference between max velocity of step to initial values
    o. If delta < 0, make the step as current/initial model
    p. Run the above steps for n iterations
    q. Compare the Initial result values to final result values

    Steps d-p basically describe an optimization process based on Simulated Annealing method

3. Using the Throttle map from throttle optimization, another set variables would be varied,
    - Number of 1st stage engines
    - Number of 2nd stage engines
    - Number of boosters

    This is done because after reaching certain iterations, the Throttle map doesn't seem to achieve much improvement based on the previous iterations.

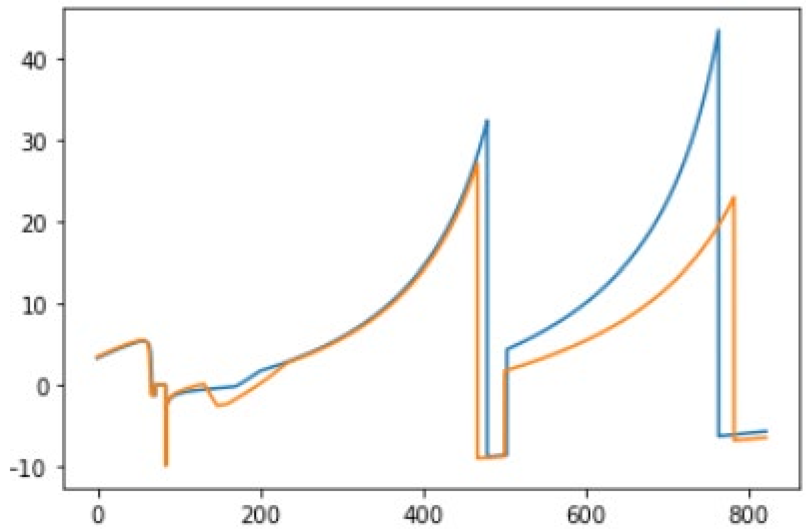
Results & Conclusions

Optimization Parameters:

- Iterations = 50 (Couldn't run more at once because of the time consumed to run RocketSim)
- Step Size = 1
- Temperature = 100
- Probability threshold for delta estimation = exp(-delta/(Temperature/(i+1))

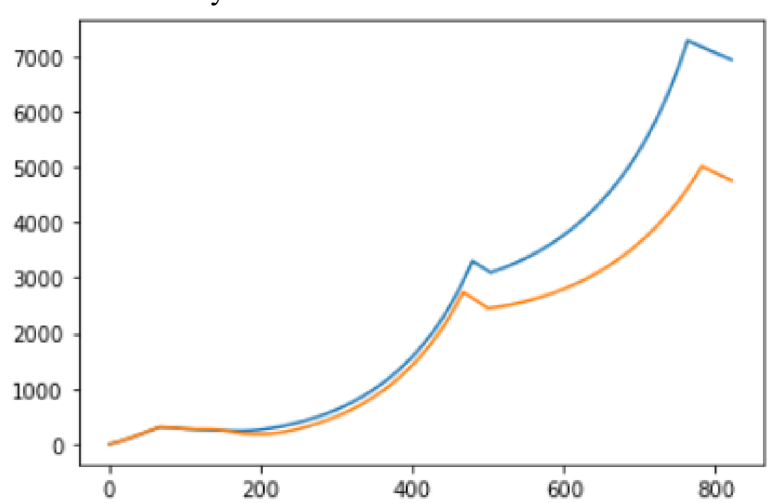
Plots achieved after performing Throttle optimization

- Time vs Acceleration:



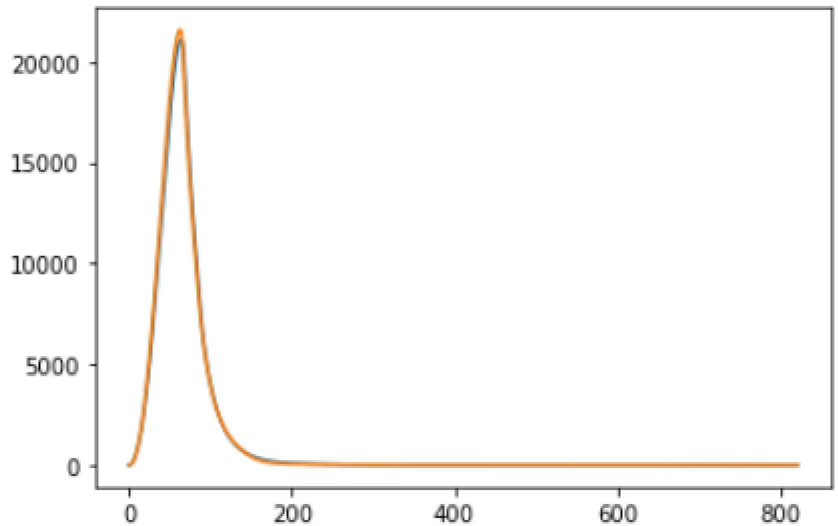
As can be observed, acceleration value is well below 8g and it can be seen that acceleration increases after performing throttle optimization (Orange – Initial values and Blue- Final Values

- Time vs Velocity:



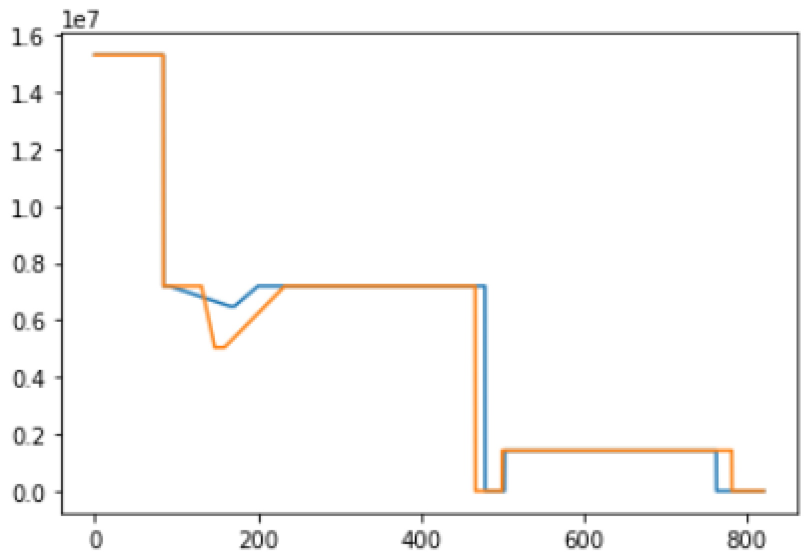
The max. velocity achieved by the rocket initially is around 5000 m/s and after Throttle optimization it increases to around 7200 m/s. There is improvement in the velocity performance, but it does not reach the target. It can be concluded that, further tuning of optimization parameters is needed but mainly running more iterations will certainly yield better results and tell us how well the algorithm can actually perform.

- Time vs Maximum Dynamic Pressure:



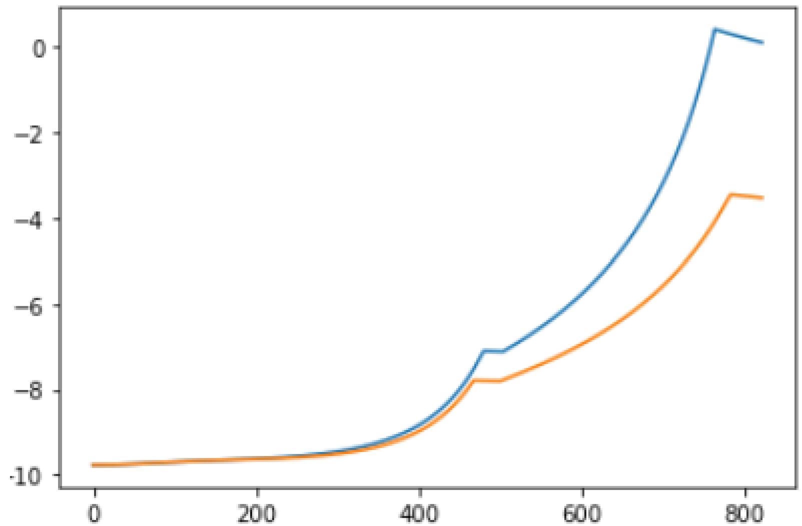
As can be seen, the maximum dynamic pressure profile remains almost same between the final and initial models. Also, the dynamic pressure always remains under 33440 Pa.

- Time vs Thrust:



As can be seen, the initial and final Thrust throttle maps are being compared. Interesting observations can be made here. First, throttle down initially starts after some time boosters are burnt out but in the final map which performs better, the throttle down starts as soon as boosters are out. Second, the throttle down and up in final model are much gradual compared to the initial model. Third, total burn time of first stage increases and burn time of 2nd stage reduces.

- Time vs Orbital Acceleration:



It can be observed that, based on the initial throttle map the rocket was not able to achieve acceptable value and was pretty far, reaching -4 instead of achieving positive value near 0 m/s^2 . After performing throttle optimization, significant improvement can be observed and the rocket was able to achieve positive orbital speed, thus reaching a stable orbit.

<u>Conclusions</u>:

Thus, based on the results and the observations it can be said that the methodology suggested here seems to be profitable and achieve significant improvements from initial set of values. Important points of discussion and further work are,

- Though the method seems sound and can achieve improvements, for proper evaluation a run with more iterations should be performed
- Setting up a python code - that could read batch file, run RocketSim simulation, read results, update batch based on step and the algorithm, and then loop this process – was taxing and time consuming but fruitful
- Based on the observations made in Thrust profile after performing optimization, further tuning of algorithm can be done
- The Throttle temperatures are selected randomly for every step, this can be further tuned based on their relations
- After obtaining acceptable throttle map values, further improvement can be achieved by optimizing the design variables - no. of 1^{st} and 2^{nd} stage engines and no. of boosters - and using similar Progressive optimization method
- Since SA methods work really good with discrete values, optimization based on various combinations of engine types can also be performed to see best combinations
- The algorithm can further be improved by understanding the physics better and by understanding the relations between different variables. This can be done by adding pieces of code as discussed above and by making this methodology more robust to uncertain factors
- The uncertain factors can be other possible constraints, requirements but factors such as operating systems, compatibility, interoperability are also important and can bring lot of uncertain issues.

Appendix

Language: Python (Jupyter Notebook)

Platform: Windows OS (necessary to run Simulation file through python)

Path: Same as the RocketSim.exe file so that it can have permission to run the file

Conditions:

- Batch.txt should have 1st value greater than 0 to automate the process

Code:

```
import numpy as np
import os
from scipy import interpolate
import matplotlib.pyplot as plt

# Reading the Batch file

#Batch_file = open(r"C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalExam\Test Run\Batch.txt")
#Batch_content = Batch_file.read()
#Batch_content = Batch_content.split('\n')
#Batch = np.asarray(Batch_content)
Batch = np.loadtxt(r"C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalExam\Test Run\Batch.txt")

# Reading Rocket data

Rocket_data = np.loadtxt(r"C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalExam\Test Run\RocketData.txt")
#print(Rocket_data[:, 0])

# Design Variables - Types & Number of Engines

n_stages = float(Batch[1])          # No. of stages
n_boosters = float(Batch[2])        # No. of boosters
booster_type = float(Batch[3])      # Booster type(1-32)
fsengine_type = float(Batch[5])     # First-stage engine type
n_fsengine = float(Batch[6])        # No. of First-stage engine
ssengine_type = float(Batch[8])     # 2nd stage engine type
n_ssengine = float(Batch[9])        # No. of 2nd stage engines
print(n_ssengine)

# Design Variables - Throttle map time

T_tD = float(Batch[21])             # Throttle down
T_tDC = float(Batch[23])            # Throttle down complete
```

```
throttle_Ttdc = float(Batch[24])  # Throttle value at T_tDC
T_tU = float(Batch[25])           # Throttle up
T_tUC = float(Batch[27])          # Throttle up complete
T_meco = float(Batch[29])         # Main engine cut-off time
T_2SI = float(Batch[31])          # 2nd stage ignition time
T_2ECO = float(Batch[33])         # 2nd stage cut-off time

# Circle packing formula setup to calculate Main Stage diameter

c_enc = np.array([[1.0, 1.0],[2.0, 2.0],[3.0, 2.154],
                  [4.0, 2.414],[5.0, 2.701],[6.0, 3.0],
                  [7.0, 3.0],[8.0, 3.304],[9.0, 3.613],
                  [10.0, 3.813],[11.0, 3.923],[12.0, 4.029],
                  [13.0, 4.236],[14.0, 4.328],[15.0, 4.521],
                  [16.0, 4.615],[17.0, 4.792],[18.0, 4.863],
                  [19.0, 4.863],[20.0, 5.122]])

x = c_enc[:, 0]
y = c_enc[:, 1]
f = interpolate.interp1d(x, y, fill_value='extrapolate')

# getting the diameter of First stage engine from Rocket Data

for i in range(len(Rocket_data[:, 0])):
    #print(Rocket_data[i, 7], Rocket_data[i, 0])
    if Rocket_data[i, 0] == fsengine_type:
        fsengine_dia = Rocket_data[i, 7]

dia_ms = 1.1*fsengine_dia*f(n_fsengine)
#print(dia_ms)
#print(Batch[47])


# Running RocketSim.exe
import subprocess
import time
from keyboard import press

os.chdir(r'C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalExam\T
est Run')
#p=subprocess.Popen(r'C:\Users\vinay\OneDrive - Clemson University\ME8710\
FinalExam\Test Run\RocketSim.exe')
#subprocess.check_output('Taskkill /PID %d /F' % pid)
#p.terminate()
```

```python
os.startfile(r'C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalEx
am\Test Run\RocketSim.exe')
time.sleep(1)
press('enter')
#time.sleep(1)
#os.system('taskkill /F /im RocketSim.exe')


# Read the telemetry data
import pandas as pd

df = pd.read_csv(r'C:\Users\vinay\OneDrive - Clemson University\ME8710\Fin
alExam\Test Run\Telemetry.txt', sep = ' ', index_col=False)
Telemetry = df.to_numpy()
#Telemetry = np.loadtxt(r"C:\Users\vinay\OneDrive - Clemson University\ME8
710\FinalExam\Test Run\Telemetry.txt", skiprows=1)

r_time = Telemetry[:, 0]
thrust = Telemetry[:, 1]
mass = Telemetry[:, 2]
altitude = Telemetry[:, 3]
velocity = Telemetry[:, 4]
acceleration = Telemetry[:, 5]
MaxQ = Telemetry[:, 12]
orbital_acc = Telemetry[:, 14]
np.amax(np.abs(acceleration))


import random
import pandas as pd
from math import exp

# Optimizing throttle map
def ThrotOpt(Batch, iters, ss, temp):

    # Throttle Map - Initial Values
    T_tD_initial = float(Batch[21])            # Throttle down
    T_tDC_initial = float(Batch[23])           # Throttle down complete
    throttle_Ttdc_initial = float(Batch[24])   # Throttle value at T_tDC
    T_tU_initial = float(Batch[25])            # Throttle up
    T_tUC_initial = float(Batch[27])           # Throttle up complete
    T_meco_initial = float(Batch[29])          # Main engine cut-off time
    T_2SI_initial = float(Batch[31])           # 2nd stage ignition time
    T_2ECO_initial = float(Batch[33])          # 2nd stage cut-off time

    # Throttle Map - Current Values
    T_tD_curr = T_tD_initial           # Throttle down
```

```
    T_tDC_curr = T_tDC_initial          # Throttle down complete
    throttle_Ttdc_curr = throttle_Ttdc_initial  # Throttle value at T_tDC
    T_tU_curr = T_tU_initial           # Throttle up
    T_tUC_curr = T_tUC_initial          # Throttle up complete
    T_meco_curr = T_meco_initial         # Main engine cut-off time
    T_2SI_curr = T_2SI_initial          # 2nd stage ignition time
    T_2ECO_curr = T_2ECO_initial         # 2nd stage cut-off time




    # Running RocketSim.exe
    os.chdir(r'C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalEx
am\Test Run')
    os.startfile(r'C:\Users\vinay\OneDrive - Clemson University\ME8710\Fin
alExam\Test Run\RocketSim.exe')
    time.sleep(1)
    press('enter')
    # Getting telemetry values
    up_file = open(r'C:\Users\vinay\OneDrive - Clemson University\ME8710\F
inalExam\Test Run\Telemetry.txt', 'r+')
    df = pd.read_csv(up_file, sep = ' ', index_col=False, error_bad_lines=
False)
    Telemetry = df.to_numpy()

    curr_m_vel = np.amax(Telemetry[:, 4])

    op_model = np.loadtxt(r"C:\Users\vinay\OneDrive - Clemson University\M
E8710\FinalExam\Test Run\Rocket.txt")
    timp_model = op_model

    prob = []

    for i in range(iters):

        curr_m_vel = np.amax(Telemetry[:, 4])
        #step = (ss * float(random.randint(-10, 10)))
        # Throttle Map - Step Values

        print('iter:', i)
        print('[', T_tD_curr, T_tDC_curr, throttle_Ttdc_curr, T_tU_curr, T
_tUC_curr, T_meco_curr, T_2SI_curr, T_2ECO_curr, ']')

        T_tD_step = T_tD_curr + (ss * float(random.randrange(-
10, 10, 1)))        # Throttle down
        temp = T_tD_step
```

```
        while (temp < (T_tD_step + 1)):
            temp = T_tDC_curr + (ss * float(random.randrange(-
10, 10, 1)))        # Throttle down complete
        T_tDC_step = temp

        throttle_Ttdc_step = (random.randint(6,9))/10
        # Throttle value at T_tDC

        while (temp < (T_tDC_step + 1)):
            temp = T_tU_curr + (ss * float(random.randrange(-10, 10, 1)))
        T_tU_step = temp
    # Throttle up

        while (temp < (T_tU_step + 1)):
            temp = T_tUC_curr + (ss * float(random.randrange(-10, 10, 1)))
        T_tUC_step = temp
    # Throttle up complete

        while (temp < (T_tUC_step + 1)):
            temp = T_meco_curr + (ss * float(random.randrange(-
10, 10, 1)))
        T_meco_step = temp
    # Main engine cut-off time

        while (temp < (T_meco_step + 1)):
            temp = T_2SI_curr + (ss * float(random.randrange(-10, 10, 1)))
        T_2SI_step = temp
    # 2nd stage ignition time

        while (temp < (T_2SI_step + 1)):
            temp = T_2ECO_curr + (ss * float(random.randrange(-
10, 10, 1)))
        T_2ECO_step = temp
    # 2nd stage cut-off time

        Batch[21] = T_tD_step
        Batch[23] = T_tDC_step
        Batch[24] = throttle_Ttdc_step
        Batch[25] = T_tU_step
        Batch[27] = T_tUC_step
        Batch[29] = T_meco_step
        Batch[31] = T_2SI_step
        Batch[33] = T_2ECO_step

        #print(Batch)
```

```python
        np.savetxt(r"C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalExam\Test Run\Batch.txt", Batch, fmt = '%s')

        # Running RocketSim.exe
        os.chdir(r'C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalExam\Test Run')
        os.startfile(r'C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalExam\Test Run\RocketSim.exe')
        time.sleep(3)
        press('enter')
        # Getting telemetry values
        up_file = open(r'C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalExam\Test Run\Telemetry.txt', 'r+')
        df = pd.read_csv(up_file, sep = ' ', index_col=False, error_bad_lines=False)
        Telemetry = df.to_numpy()
        #print(Telemetry)
        up_file.truncate(0)

        #Telemetry = np.loadtxt(r"C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalExam\Test Run\Telemetry.txt", delimiter=' ', skiprows=1)
        r_time = Telemetry[:, 0]
        thrust = Telemetry[:, 1]
        mass = Telemetry[:, 2]
        altitude = Telemetry[:, 3]
        velocity = Telemetry[:, 4]
        acceleration = Telemetry[:, 5].astype(float)
        MaxQ = Telemetry[:, 12]

        m_acc = np.amax(np.abs(acceleration))
        m_vel = np.amax(velocity)
        m_maxq = np.amax(MaxQ)

        if (m_maxq <= 33400):
            if (m_acc <= 80):
                if (m_vel >= 15456) and (m_vel <= 15656):
                    curr_model = np.loadtxt(r"C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalExam\Test Run\Rocket.txt")
                    op_model = np.vstack((op_model, curr_model))
                    # Updating initial values
                    T_tD_initial = T_tD_step           # Throttle down
                    T_tDC_initial = T_tDC_step           # Throttle down complete
                    throttle_Ttdc_initial = throttle_Ttdc_step  # Throttle value at T_tDC
                    T_tU_initial = T_tU_step           # Throttle up
```

```
                    T_tUC_initial = T_tUC_step          # Throttle up comp
lete
                    T_meco_initial = T_meco_step         # Main engine cut
-off time
                    T_2SI_initial = T_2SI_step          # 2nd stage igniti
on time
                    T_2ECO_initial = T_2ECO_step         # 2nd stage cut-
off time

                elif (m_vel >= 11500) and (m_vel <= 17000):
                    curr_model = np.loadtxt(r"C:\Users\vinay\OneDrive - Cl
emson University\ME8710\FinalExam\Test Run\Rocket.txt")
                    timp_model = np.vstack((timp_model, curr_model))

                    # Updating current values
                    T_tD_curr = T_tD_step            # Throttle down
                    T_tDC_curr = T_tDC_step          # Throttle down compl
ete
                    throttle_Ttdc_curr = throttle_Ttdc_step  # Throttle va
lue at T_tDC
                    T_tU_curr = T_tU_step            # Throttle up
                    T_tUC_curr = T_tUC_step          # Throttle up complet
e
                    T_meco_curr = T_meco_step         # Main engine cut-
off time
                    T_2SI_curr = T_2SI_step          # 2nd stage ignition
time
                    T_2ECO_curr = T_2ECO_step         # 2nd stage cut-
off time

                delta = (curr_m_vel - m_vel)
                t_step = temp/float(i+1)
                m = exp(-(abs(delta))/t_step)
                prob.append(m)

                if delta < 0 or np.random.rand() < m:

                    # Updating current values
                    T_tD_curr = T_tD_step            # Throttle down
                    T_tDC_curr = T_tDC_step          # Throttle down compl
ete
                    throttle_Ttdc_curr = throttle_Ttdc_step  # Throttle va
lue at T_tDC
                    T_tU_curr = T_tU_step            # Throttle up
                    T_tUC_curr = T_tUC_step          # Throttle up complet
e
```

```
                    T_meco_curr = T_meco_step         # Main engine cut-
off time
                    T_2SI_curr = T_2SI_step          # 2nd stage ignition
time
                    T_2ECO_curr = T_2ECO_step         # 2nd stage cut-
off time

                    curr_model = np.loadtxt(r"C:\Users\vinay\OneDrive - Cl
emson University\ME8710\FinalExam\Test Run\Rocket.txt")
                # Updating current values
                #T_tD_curr = T_tD_step           # Throttle down
                #T_tDC_curr = T_tDC_step          # Throttle down complete
                #throttle_Ttdc_curr = throttle_Ttdc_step  # Throttle value
 at T_tDC
                #T_tU_curr = T_tU_step           # Throttle up
                #T_tUC_curr = T_tUC_step          # Throttle up complete
                #T_meco_curr = T_meco_step         # Main engine cut-
off time
                #T_2SI_curr = T_2SI_step          # 2nd stage ignition tim
e
                #T_2ECO_curr = T_2ECO_step         # 2nd stage cut-
off time

    return op_model, timp_model, curr_model, prob

i = 50
ss = 1
t = 100
op, timp, curr, probs = ThrotOpt(Batch, i, ss, t)


Batch0 = Batch[0]
Batch = np.loadtxt(r"C:\Users\vinay\OneDrive - Clemson University\ME8710\F
inalExam\Test Run\Rocket.txt")
Batch = np.insert(curr, 0, 1.0, axis=0)
np.savetxt(r"C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalExam
\Test Run\Batch.txt", Batch, fmt = '%s')

os.startfile(r'C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalEx
am\Test Run\RocketSim.exe')
time.sleep(1)
press('enter')

df = pd.read_csv(r'C:\Users\vinay\OneDrive - Clemson University\ME8710\Fin
alExam\Test Run\Telemetry.txt', sep = ' ', index_col=False)
Telemetry = df.to_numpy()
```

```
r_time = Telemetry[:, 0]
thrust_curr = Telemetry[:, 1]
mass_curr = Telemetry[:, 2]
altitude_curr = Telemetry[:, 3]
velocity_curr = Telemetry[:, 4]
acceleration_curr = Telemetry[:, 5]
MaxQ_curr = Telemetry[:, 12]
orbital_acc_curr = Telemetry[:, 14]

Batch0 = Batch[0]
Batch = np.loadtxt(r"C:\Users\vinay\OneDrive - Clemson University\ME8710\F
inalExam\Test Run\Rocket.txt")
Batch = np.insert(op, 0, 1.0, axis=0)
np.savetxt(r"C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalExam
\Test Run\Batch.txt", Batch, fmt = '%s')

os.startfile(r'C:\Users\vinay\OneDrive - Clemson University\ME8710\FinalEx
am\Test Run\RocketSim.exe')
time.sleep(1)
press('enter')

df = pd.read_csv(r'C:\Users\vinay\OneDrive - Clemson University\ME8710\Fin
alExam\Test Run\Telemetry.txt', sep = ' ', index_col=False)
Telemetry = df.to_numpy()

r_time = Telemetry[:, 0]
thrust_in = Telemetry[:, 1]
mass_in = Telemetry[:, 2]
altitude_in = Telemetry[:, 3]
velocity_in = Telemetry[:, 4]
acceleration_in = Telemetry[:, 5]
MaxQ_in = Telemetry[:, 12]
orbital_acc_in = Telemetry[:, 14]


plt.figure(1)
plt.plot(r_time, acceleration_curr)
plt.plot(r_time, acceleration_in)

plt.figure(2)
plt.plot(r_time, velocity_curr)
plt.plot(r_time, velocity_in)

plt.figure(3)
plt.plot(r_time, MaxQ_curr)
plt.plot(r_time, MaxQ_in)
```

```
plt.figure(4)
plt.plot(r_time, altitude_curr)
plt.plot(r_time, altitude_in)

plt.figure(5)
plt.plot(r_time, thrust_curr)
plt.plot(r_time, thrust_in)

plt.figure(6)
plt.plot(r_time, mass_curr)
plt.plot(r_time, mass_in)

plt.figure(7)
plt.plot(r_time, orbital_acc_curr)
plt.plot(r_time, orbital_acc_in)
```