A Study of the TDO missions on Atlas V (551)

AE 240: Spaceflight Mechanics

Aaron John Sabu

1 Introduction

This assignment investigates the mission characteristics of the TDO missions in 2019 and 2020 on the Atlas V (551). Simulations are performed to demonstrate the trajectory of the rocket and the position of delivery of the above mentioned payload. Finally, the burn profile and related information of the rocket are computed using information obtained from the World Wide Web.

2 Mission Characteristics

2.1 TDO

2.1.1 TDO-1

TDO-1 (NORAD ID - 44482) is a mission to test orbital debris tracking technologies [1], launched from the Air Force Eastern Test Range, Florida on 2019 August 08 [2]. This mission was executed by the United States Air Force (USAF) Space and Missile Systems Center (SMC) [presently acquired by the United States Space Force], a development center headquartered at the Los Angeles Air Force Base, California [3]. The satellite plays a significant role in space situational awareness (SSA); i.e., keeping track of objects in orbit and predicting where they will be at any given time. This helps monitor the possibility for threatening situations to satellites and launches from the tens of thousands of objects in orbit around the Earth [4]. Being a 12U CubeSat, this satellite had dimensions of 30cm $\times$ 20cm $\times$ 20cm [5] as depicted in Fig. 1.

Figure 1: 12U CubeSat [6]

TDO-1 flew on an Atlas V (551) rocket as a secondary payload on the aft buckhead carrier (ABC, depicted in Fig. 2) of the Atlas V (551) rocket launching the primary satellite AEHF 5 (Advanced Extreme High Frequency Satellite 5), a military communications satellite of the Department of Defense.

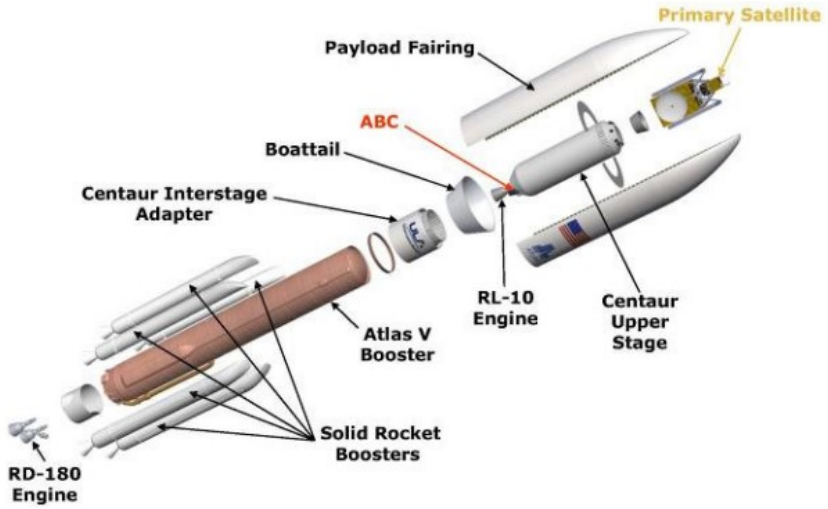


Figure 2: Aft Buckhead Carrier (ABC) in Atlas V 500 Series [7]

2.1.2 TDO-2

TDO-2 (NORAD ID - 45464) is another 12U Cubesat provided by the United States Space Force and is designed to test new capabilities of small satellites used by US government agencies. This mission supports space domain awareness through optical calibration and satellite laser ranging [8]. It was launched from the Air Force Eastern Test Range, Florida, similar to TDO-1, on 2020 March 26 [9] and was jettisoned from the Centaur at T+0:28:52.7 after the second Centaur firing but before the third. This Atlas V flight is the first mission for the United States Space Force [10] and represents the 500th flight of the RL10 engine.

Figure 3: TDO-2 Satellite [6]

2.2 Atlas V (551)

The Atlas V is an expendable launch system originally designed by Lockheed Martin and now operated by United Launch Alliance. The rocket consists of two main stages, the first powered by a Russian RD-170 engine (burning kerosene and liquid oxygen) and the second powered by one or two United States RL10 engine(s) (burning liquid hydrogen and liquid oxygen). The rocket has proven to be a huge success with a streak of 76 consecutive successful launches as of November 13, 2020 since the first ULA Atlas V launch on October 11, 2007.

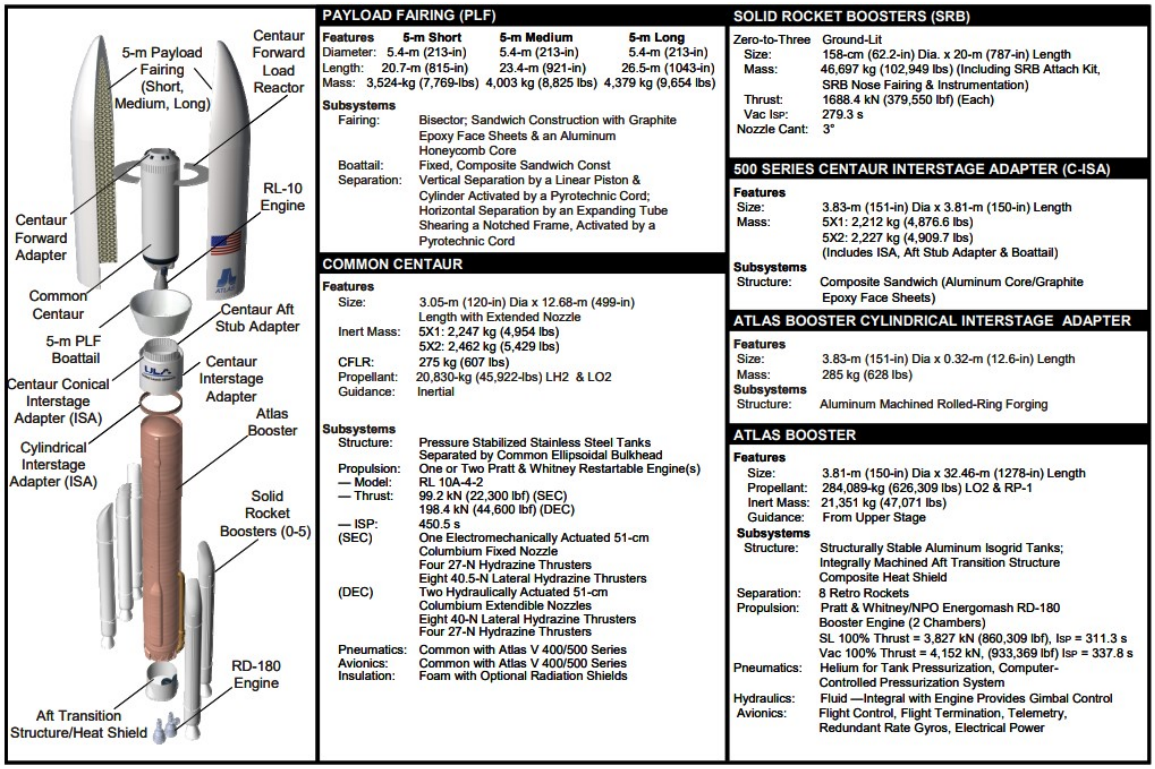


Figure 4: Atlas V 500 Series - Launch System [11]

The upper stage of the rocket, Centaur, uses a pressure-stabilized propellant-tank design and cryogenic propellants. Its inertial navigation unit (INU) provides guidance and navigation for both the Atlas and Centaur. The Centaur engines are capable of multiple in-space starts, making possible insertion into low Earth parking orbit, followed by a coast period and then insertion into geostationary transfer orbit. The engine may further burn to permit direct injection of payloads into geostationary orbit.
The Atlas V (551) rocket, at a base price of US$153 million, has a fairing of 5.4 m similar to the Atlas V (501) rocket in comparison to 4 m for the Atlas V (401) rocket. The rocket has been launched 10 times to date. It consists of a single Common Core Booster (CCB) but has 5 Solid Rocket Boosters (SRBs) in comparison to the Atlas V (501) rocket that has no SRBs or even the Atlas V (541) rocket that has 4 SRBs. The rocket hosts a single engine Centaur in comparison to the dual rocket Centaur in the Atlas V (552) rocket and it can carry a payload of 18,814 kg to Low Earth Orbit (8,900 km to Geosynchronous Transfer Orbit) [12].

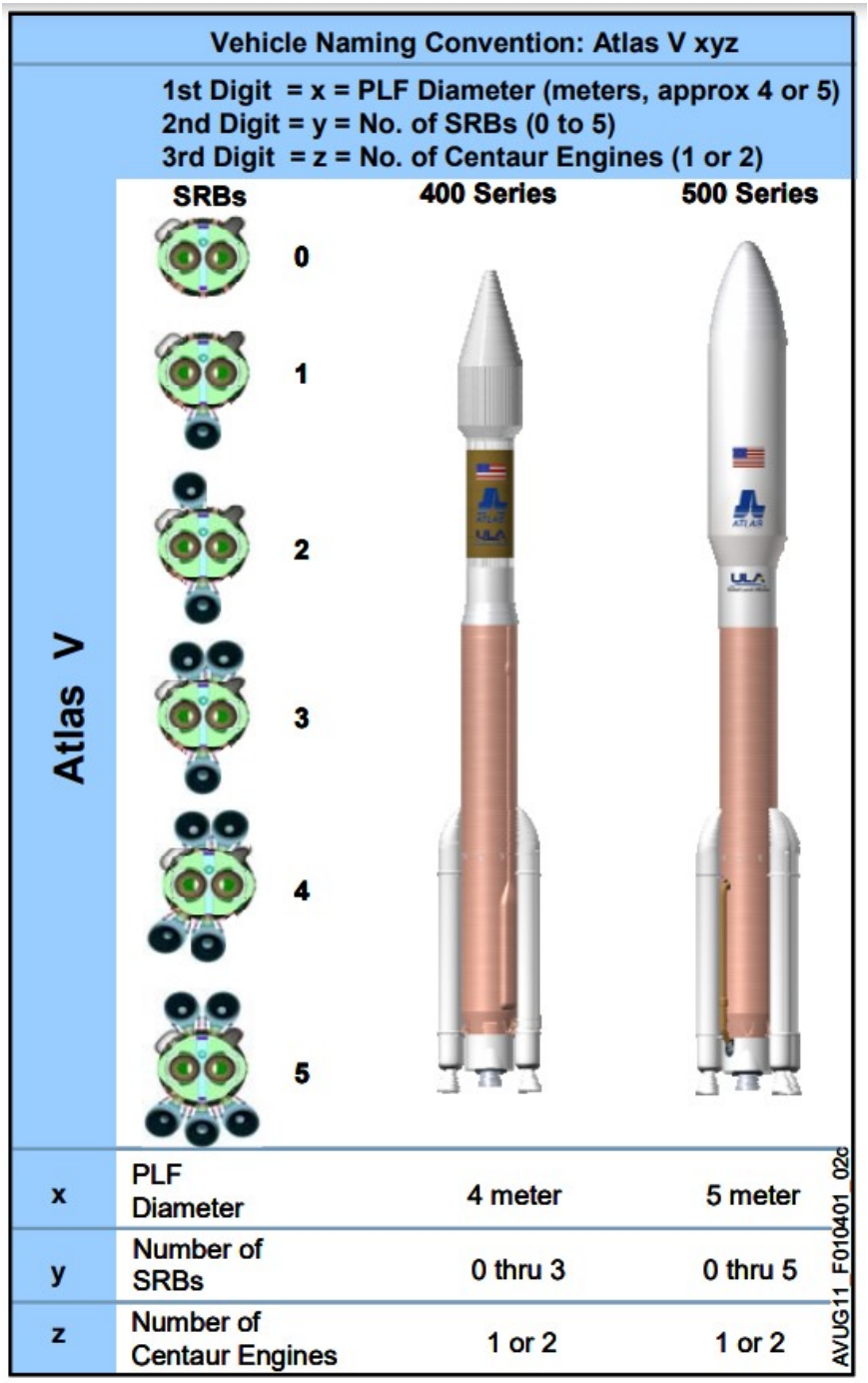


Figure 5: Atlas V Rockets - Vehicle Naming Convention [11]

3 Orbital Parameters

On investigating the orbital parameters as provided by `in-the-sky.org`, we notice the presence of two datetime parameters: 'Date fetched' representing the datetime of measurement of the orbital parameters and 'Epoch of osculation' representing the datetime at which the real orbit and the perturbed orbit touch each other. The orbital parameters provided in the following table are later used along with the epoch osculation to develop simulations of the orbit. These parameters are defined as:

Space body	Date Fetched	Epoch Osculation	Inc. (°)	Ecc.	RA Asc Node (hr)	Arg Peri (°)	Mean Anomaly (°)	Mean motion (rev/day)
TDO-1	2019-08-20 21:47	2019-08-08 15:48:43	26.4221	0.72690	19.9751	180.1991	123.1747	2.31717
Altas V Centaur	2019-08-20 21:47	2019-08-08 17:49:47	9.7973	0.34585	19.9786	180.5643	174.4088	1.59360
TDO-2	2020-04-04 15:03	2020-03-27 07:01:04	26.4980	0.72823	21.2656	179.9623	357.3555	2.30489
Altas V Centaur	2020-03-27 18:52	2020-03-27 14:01:43	13.4303	0.41484	21.2445	180.3608	170.7147	1.71717

Table 1: Orbital Parameters

According to [1], TDO-1 follows an orbit of periapsis 208km, apoapsis 35264km, and an eccentricity of 26.17° and TDO-2 follows an orbit of periapsis 200km, apoapsis 35459km, and an eccentricity of 26.50°. We will follow the parameters provided in the table for simulation due to their higher levels of accuracy.

4 Orbit Simulations

The following snapshots depict the transfer orbits and final orbit of each of the above missions: TDO-1 and TDO-2. The simulations are not perfect due to the lack of consideration of all orbital parameters due to the lack of resources to obtain them. Instead the simulations have been developed solely from the perigee distance, apogee distance, and the inclination of the orbit, effectively providing three orbital parameters: semi-major axis, eccentricity, and inclination.

4.1 TDO-1 Orbit Insertion

The information required for this simulation has been procured from the mission profile video by the United Launch Alliance [13]. The snapshots used from the video are provided in Appendix A.

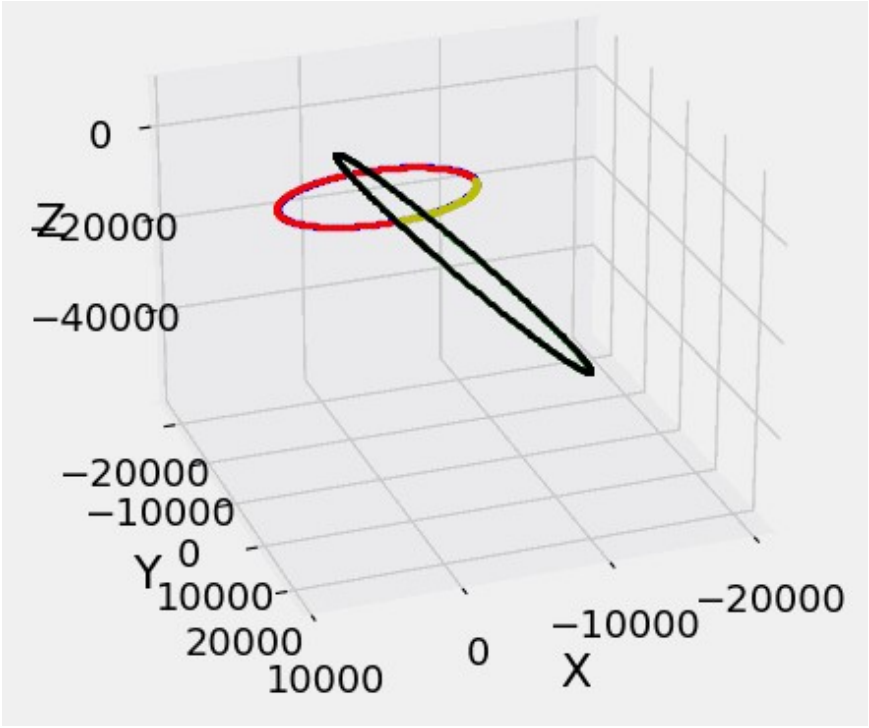


Figure 6: TDO-1 Simulation

4.2 TDO-2 Orbit Insertion

The information required for this simulation has been procured from the mission profile video by the United Launch Alliance [14]. The snapshots used from the video are provided in Appendix B. The orbit achieved by this mission is similar to that of TDO-1.

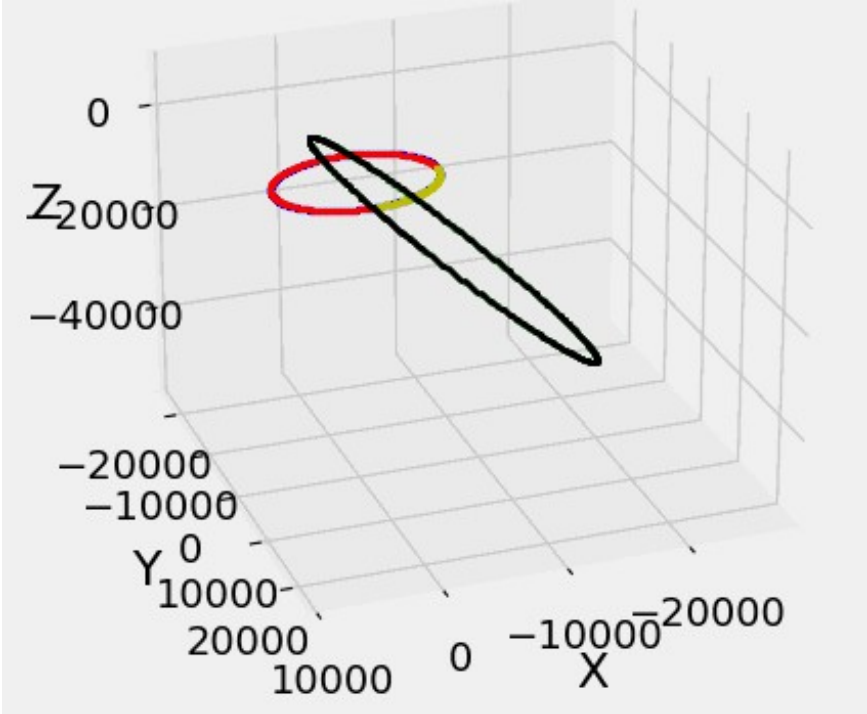


Figure 7: TDO-2 Simulation

5 Burn Profiles

Due to the limited information available to the public, we have developed approximate models for the burn profiles of the rockets. Based on the information from the snapshots provided in Appendices A and B, we assume unknown information and develop inferences for each mission as described below. Since the two missions achieved similar orbits and followed similar mission profiles, we will demonstrate the burn profile of the TDO-1 mission.

Due to the nature of the mission, we will consider it to have been split into three parts: a vertical ascent with the inclusion of the effect of gravity, a constant pitch-rate gravity turn, and a constant velocity gravity turn. We obtain the values for empty mass, propellant mass, specific impulse, and burn time for each stage and the main payload (AEHF) from [12] [15] [16] [17]. The mass of the TDO satellite is ignored since it is unknown (possibly classified by the US Dept. of Defense) and, moreover, does not contribute significantly to the total mass (the dimensions of the satellite are small).

5.1 Vertical Ascent against Gravity

We consider the rocket to have ascended vertically against the force of gravity until the point of depleting the fuel in all five boosters. In this part, the boosters would have burnt along with the first stage (CCB) of the Atlas V rocket. Hence, we compute the specific impulse as:

$$I_{sp} = \frac{5\dot{m}_{boost} I_{sp,boost} + \dot{m}_{stg01} I_{sp,stg01}}{5\dot{m}_{boost} + \dot{m}_{stg01}} \tag{1}$$

The mass rate (β) is considered to be constant and equal to:

$$\beta = 5\beta_{boost} + \beta_{stg01} = 5\frac{m_{p,boost}}{t_{b,boost}} + \frac{m_{p,stg01}}{t_{b,stg01}} \tag{2}$$

The equations for burn profile (mass, m), velocity (v), vertical distance (h), horizontal distance (x) and pitch (θ) are given as:

$$m = m_0 - \beta t \tag{3}$$

$$v = g_0 I_{sp} \ln \frac{m_0}{m} - g_0 t \tag{4}$$

$$h = \frac{m_0 g_0 I_{sp}}{\beta} \left(\left(1 - \frac{\beta t}{m_0} \right) \ln \left(1 - \frac{\beta t}{m_0} \right) + \frac{\beta t}{m_0} \right) - \frac{1}{2} \tilde{g} t^2 \tag{5}$$

$$x = 0.0 \tag{6}$$

$$\theta = 0.0 \tag{7}$$

5.2 Constant Pitch-Rate Turn

We consider the rocket to have performed a constant pitch rate turn from the point of ejecting the boosters until the point of depleting the fuel in the first stage. For the proper functionality of the maneuver, we consider the rocket to have been given a pitch kick prior to the maneuver. Given the pitch rate q_0, initial pitch θ_0, initial velocity v_0, and initial vertical distance h_0, the equations for burn profile (mass, m), velocity (v), vertical distance (h), horizontal distance (x) and pitch (θ) are given as:

$$\theta = q_0 t + \theta_0 \tag{8}$$

$$m = m_0 \exp \left(-\frac{2\tilde{g}}{q_0 g_0 I_{sp}} (\sin(\theta) - \sin(\theta_0)) \right) \tag{9}$$

$$v = \frac{\tilde{g} \sin(\theta)}{q_0} + v_0 \tag{10}$$

$$h = \frac{\tilde{g}}{4 q_0^2} (\cos(2\theta_0) - \cos(2\theta)) + h_0 \tag{11}$$

$$x = \frac{\tilde{g}}{2 q_0^2} \left((\theta - \theta_0) - \frac{\sin(2\theta) - \sin(2\theta_0)}{2} \right) \tag{12}$$

5.3 Constant Velocity Turn

We consider the rocket to have performed a constant velocity turn from the point of ejecting the first stage until the point of depleting the fuel in the second stage. Given the initial (and consistent) velocity v_0, initial pitch θ_0, initial vertical distance h_0, and initial horizontal distance x_0, the equations for burn profile (mass, m), velocity (v), vertical distance (h), horizontal distance (x) and pitch (θ) are given as:

$$v = v_0 \tag{13}$$

$$\theta = 2 \tan^{-1} \left(\exp \left(\frac{\tilde{g} t}{v_0} \right) \tan \left(\frac{\theta_0}{2} \right) \right) \tag{14}$$

$$m = m_0 \left(\frac{\sin(\theta)}{\sin(\theta_0)} \right)^{-\frac{v_0}{g_0 I_{sp}}} \tag{15}$$

$$h = \frac{v_0^2}{\tilde{g}} \ln \left(\frac{\sin(\theta)}{\sin(\theta_0)} \right) + h_0 \tag{16}$$

$$x = \frac{v_0^2}{\tilde{g}} (\theta - \theta_0) + x_0 \tag{17}$$

From these equations, we may plot the burn profiles for the three parts along with the reduction in mass due to the ejections of stages as follows:

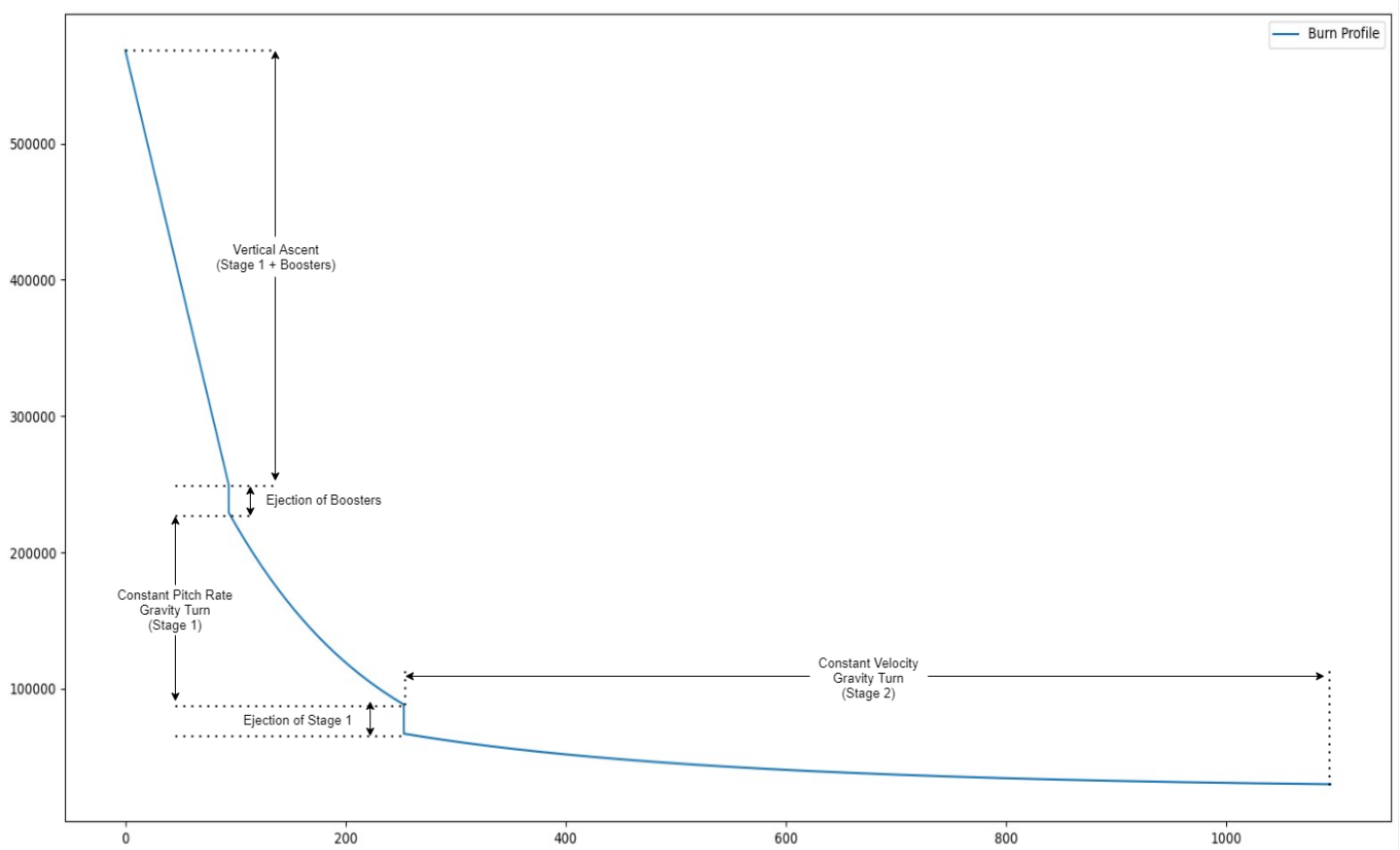


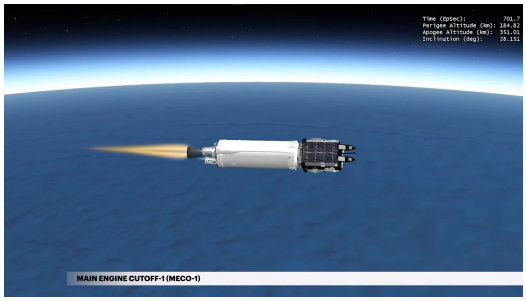
Figure 8: Burn Profile

6 Conclusion

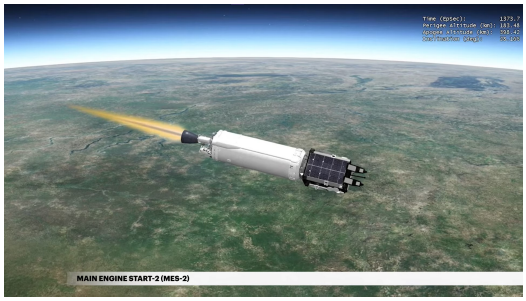
From the study performed on the TDO missions by USAF, we understand the importance and effect of space situational awareness in preventing threats to future missions while traveling to and remaining in orbit. We have performed a study of the Atlas V (551), a powerful 5-booster 2-stage rocket from ULA, which launched both the TDO missions. Furthermore, we have simulated the orbits taken by the TDO missions as well as the burn profiles of the rocket.

The code base for this assignment has been open-sourced at Space-Mission-AE240. All codes, images, and simulations have been made available in the GitHub repository. Moreover, the simulations are separately available as YouTube videos at TDO-1 and TDO-2.

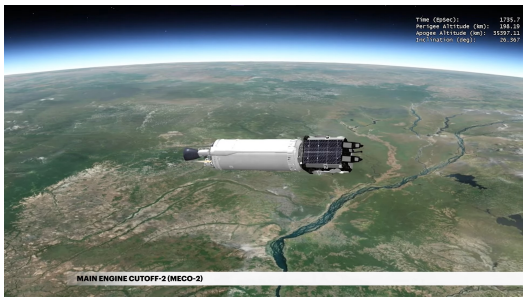
A Atlas V AEHF-5 Mission Profile - Snapshots



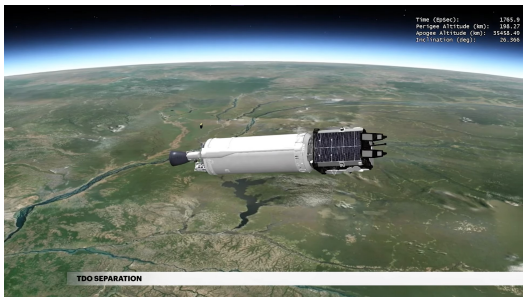
(a) MECO-1



(b) MES-2



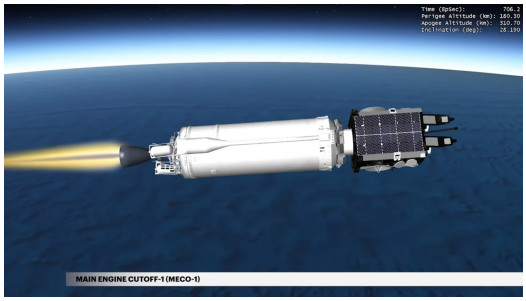
(c) MECO-2



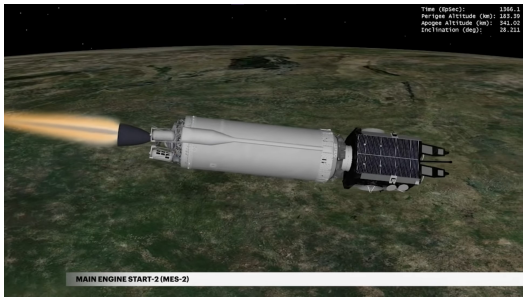
(d) TDO Separation

Figure 9: Atlas V AEHF-5 Mission Profile

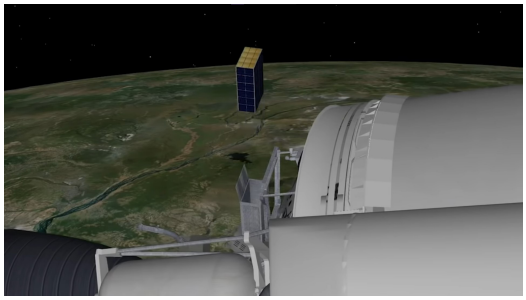
B Atlas V AEHF-6 Mission Profile - Snapshots



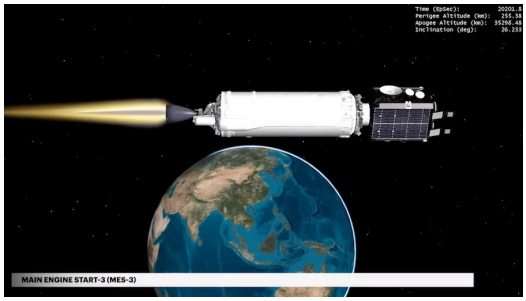
(a) MECO-1



(b) MES-2



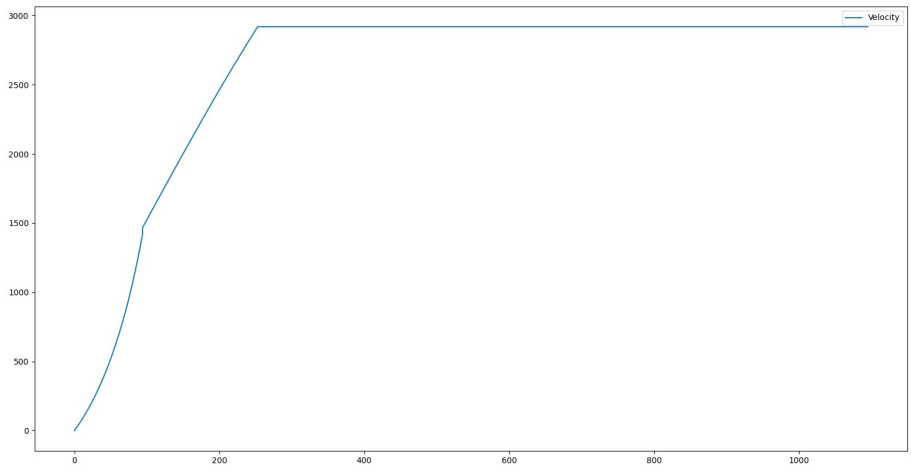
(c) TDO Separation



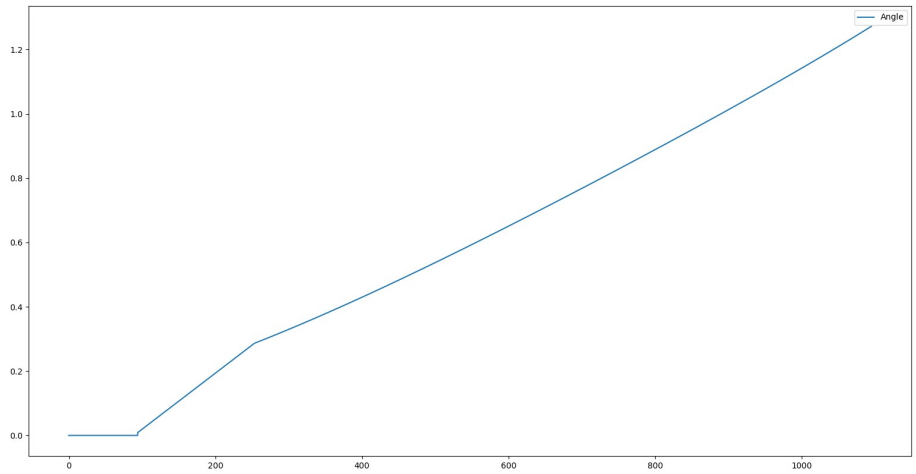
(d) Orbit Details after Separation

Figure 10: Atlas V AEHF-6 Mission Profile

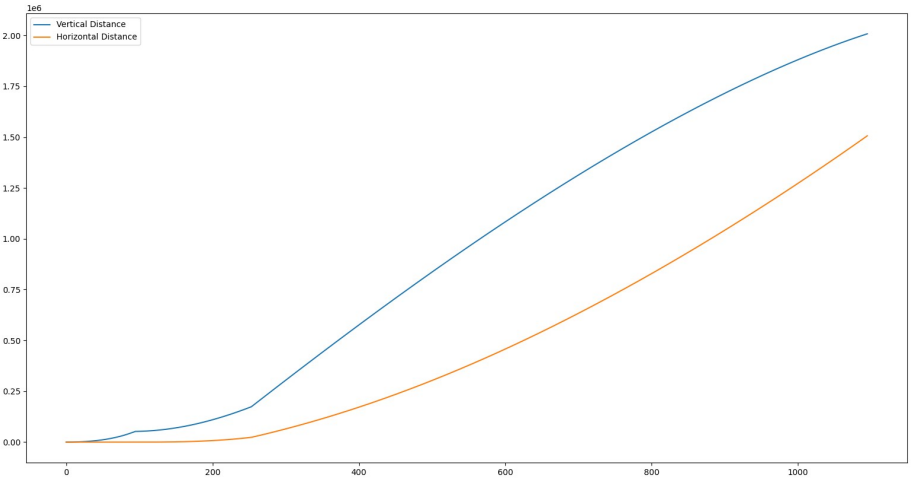
C Simulation Plots



(a) Velocity



(b) Pitch



(c) Distances

Figure 11: Atlas V (551) TDO-1 Simulation Plots

D Orbit Simulation Codes

D.1 main.py

```
import sys
import time
import matplotlib.pyplot as mplplt
from   math              import pi, cos, sin, sqrt
from   numpy             import arange
from   matplotlib        import animation
mplplt.style.use('fivethirtyeight')

tdoNum = int(sys.argv[1])
step   = pi/200
Re     = 6378.137
Perigees     = [[184.82+Re, 183.48+Re, 183.48+Re,   198.19+Re,   198.27+Re,   198.27+Re,   198.27+Re],
                [180.30+Re, 183.39+Re, 183.39+Re,   255.38+Re,   255.38+Re,   255.38+Re]]
Apogees      = [[351.01+Re, 398.42+Re, 398.42+Re, 35397.11+Re, 35458.49+Re, 35458.49+Re, 35458.49+Re],
                [310.70+Re, 341.02+Re, 341.02+Re, 35298.48+Re, 35298.48+Re, 35298.48+Re]]
Inclinations = [[   28.151,    28.166,    28.166,      26.367,      26.366,      26.366,      26.366],
                [   28.190,    28.211,    28.211,      26.233,      26.233,      26.233]]

def simulate(title, X, Y, Z, labels, nframes, markers, colors, orbitNum):
  def init():
    return(ax)
  def animate(frame):
    plots = [ax.plot(X[frame], Y[frame], Z[frame],
                     marker = markers[orbitNum[frame]],
                     color = colors[orbitNum[frame]], markersize = 2,
                     label = labels[orbitNum[frame]])]
    return plots

  xlim = (min(X)-abs(min(X)*0.5), max(X)+abs(max(X)*0.5))
  ylim = (min(Y)-abs(min(Y)*0.5), max(Y)+abs(max(Y)*0.5))
  zlim = (min(Z)-abs(min(Z)*0.5), max(Z)+abs(max(Z)*0.5))
  fig  = mplplt.figure()
  ax   = fig.add_subplot(111, projection = '3d', xlim = xlim, ylim = ylim, zlim = zlim)
  ax.legend(loc = 'upper right', shadow = True)
  ax.set_xlabel('X')
  ax.set_ylabel('Y')
  ax.set_zlabel('Z')
  ax.azim = 70
  ax.elev = 25
  anim = animation.FuncAnimation(fig, animate, init_func = init, frames = nframes)
  anim.save(title, fps = 30, writer = 'Pillow', progress_callback = lambda i, n: print(i))

X, Y, Z, orbitNum = [], [], [], []
startAngle = [   0.0,    0.0,    0.0, pi*0.2,    0.5,    0.0,    0.0]
endAngle   = [pi*2.0, pi*2.0, pi*0.5, pi*2.2, pi*2.0, pi*2.0, pi*2.0]
for orbit in range(len(Perigees[1])):
  peri = Perigees[tdoNum-1][orbit]       # Perigee
  apog = Apogees [tdoNum-1][orbit]       # Apogee
  incl = Inclinations[tdoNum-1][orbit]   # Inclination
  smaj = (peri+apog)/2.0          # Semi-Major Axis
  ecce = (peri-smaj)/smaj
  smin = sqrt(smaj**2*(1.0-ecce**2))
  orbitNum.extend([orbit for angle in arange(startAngle[orbit], endAngle[orbit], step)])
  X.extend([smaj*(ecce+cos(angle))*cos(incl) for angle in arange(startAngle[orbit], endAngle[orbit], step)])
  Y.extend([smin*sin(angle)                  for angle in arange(startAngle[orbit], endAngle[orbit], step)])
  Z.extend([smaj*(ecce+cos(angle))*sin(incl) for angle in arange(startAngle[orbit], endAngle[orbit], step)])

labels  = ['Orbit '+str(i) for i in range(len(Perigees[0]))]
markers = ['*', '.', '*', '.', '.', '.', '.']
colors  = ['b', 'r', 'y', 'g', 'k', 'k', 'k', 'k']
simulate('TDO'+str(tdoNum)+'.gif', X, Y, Z, labels, len(X), markers, colors, orbitNum)
```

D.2 run.sh

```
#!/bin/bash
for tdoNum in 1 2
do
  python main.py $tdoNum
done
```

E Burn Profile Simulation Code

E.1 main.py

```
import numpy                  as np
from math       import pi, cos, exp, log, sin, tan, atan
from matplotlib import pyplot as plt

G  = 6.67259*(10**(-11))
Me = 5.97219*(10**24)
Re = 6378100.0
mu = G*Me
g0 = 9.81
deg2rad = pi/180.0

emptyM = {'booster':  4067.0, 'stage1':  21351.0, 'stage2':  2316.0, 'AEHF': 6168.0}
proppM = {'booster': 42630.0, 'stage1': 284089.0, 'stage2': 20830.0, 'AEHF':    0.0}
Isp    = {'booster':   279.3, 'stage1':    311.3, 'stage2':   450.5, 'AEHF':    0.0}
bTime  = {'booster':    94.0, 'stage1':    253.0, 'stage2':   842.0, 'AEHF':    0.0}
M0     = 0.0
for component in emptyM.keys():
  if component == 'booster':
    M0 +=  5*emptyM[component] + 5*proppM[component]
  else:
    M0 += emptyM[component] + proppM[component]

## Calculate g at Height h
def gravity(h):
  return mu/((Re+h)**2)

## Vertical Ascent - Constant Mass Rate
def vertAsc(t, m0, Isp, v0, h0, x0, j0, ht1, beta):
  m = m0 - beta*t
  v = g0*Isp*log(m0/m) - g0*t + v0
  j = j0
  h = (m0*g0*Isp/beta)*((1.0-(beta*t/m0))*log(1.0-(beta*t/m0))+(beta*t/m0)) - 0.5*gravity(ht1)*(t**2) + v0*t +
    h0
  x = x0
  return m, v, j, h, x

## Constant Pitch Rate Gravity Turn
def constQ(t, m0, Isp, v0, h0, x0, j0, ht1, q0):
  j = q0*t + j0
  m = m0*exp(-2*gravity(ht1)*(sin(j)-sin(j0))/(q0*g0*Isp))
  v = gravity(ht1)*sin(j)/q0 + v0
  h = gravity(ht1)*(cos(2*j0)-cos(2*j))/(4*q0*q0) + h0
  x = gravity(ht1)*((j-j0)-((sin(2*j)-sin(2*j0))/2.0))/(2*q0*q0) + x0
  return m, v, j, h, x

## Constant Velocity Gravity Turn
def constV(t, m0, Isp, v0, h0, x0, j0, ht1):
  v = v0
  j = 2*atan(exp(gravity(ht1)*t/v)*tan(j0/2.0))
  m = m0*((sin(j)/sin(j0))**(-v/(g0*Isp)))
  h = (v**2/gravity(ht1))*log(sin(j)/sin(j0)) + h0
  x = (v**2/gravity(ht1))*(j-j0) + x0
  return m, v, j, h, x

## Plot all values
def plotter(tArr, mArr, vArr, jArr, hArr, xArr):
  plt.plot(tArr, mArr, label = 'Burn Profile')
  plt.legend()
  plt.show()
  plt.plot(tArr, vArr, label = 'Velocity')
  plt.legend()
  plt.show()
  plt.plot(tArr, jArr, label = 'Angle')
  plt.legend()
  plt.show()
  plt.plot(tArr, hArr, label = 'Vertical Distance')
  plt.plot(tArr, xArr, label = 'Horizontal Distance')
  plt.legend()
  plt.show()

```

```
if __name__ == "__main__":
  tArr, vArr, jArr, hArr, xArr = [[0.0] for i in range(5)]
  mArr = [M0]

  ### Launch to Ejection of Boosters
  T1   = bTime['booster']
  m0   = M0
  Isp1 = (5*Isp['booster']*proppM['booster']/bTime['booster'] + Isp['stage1']*proppM['stage1']/bTime['stage1'])
    /(5*proppM['booster']/bTime['booster'] + proppM['stage1']/bTime['stage1'])
  mpB  = 5*proppM['booster']
  mpS1 = proppM['stage1']*T1/bTime['stage1']
  mp   = mpB + mpS1
  beta = mp/T1
  for t in np.arange(0, T1, 0.01):
    m, v, j, h, x = vertAsc(t, m0, Isp1, 0.0, 0.0, 0.0, 0.0, hArr[-1], beta)
    tArr.append(t)
    mArr.append(m)
    vArr.append(v)
    jArr.append(j)
    hArr.append(h)
    xArr.append(x)

  ### Ejection of Boosters to Ejection of Stage 1
  t0   = tArr[-1]
  T2   = bTime['stage1'] - bTime['booster']
  m0   = mArr[-1] - 5*emptyM['booster']
  Isp2 = Isp['stage1']
  v0   = vArr[-1]
  h0   = hArr[-1]
  x0   = xArr[-1]
  j0   = 0.5*deg2rad  # Pitch Kick
  q0   = 0.1*deg2rad
  for t in np.arange(0, T2, 0.01):
    m, v, j, h, x = constQ(t, m0, Isp2, v0, h0, x0, j0, hArr[-1], q0)
    tArr.append(t+t0)
    mArr.append(m)
    vArr.append(v)
    jArr.append(j)
    hArr.append(h)
    xArr.append(x)

  ### Ejection of Stage 1 to Satellite Deployment
  t0   = tArr[-1]
  T3   = bTime['stage2']
  m0   = mArr[-1] - emptyM['stage1']
  Isp3 = Isp['stage2']
  v0   = vArr[-1]
  h0   = hArr[-1]
  x0   = xArr[-1]
  j0   = jArr[-1]
  for t in np.arange(0, T3, 0.01):
    m, v, j, h, x = constV(t, m0, Isp3, v0, h0, x0, j0, hArr[-1])
    tArr.append(t+t0)
    mArr.append(m)
    vArr.append(v)
    jArr.append(j)
    hArr.append(h)
    xArr.append(x)

  plotter(tArr, mArr, vArr, jArr, hArr, xArr)
```

References

[1] Gunter's Space Page: TDO 1, 2

[2] In-The-Sky.org: TDO-1

[3] Wikipedia: Space and Missile Systems Center

[4] Space Briefing Book: Space Situational Awareness

[5] NASA: What are SmallSats and CubeSats?

[6] LAAFB: Space and Missile Systems Center's multi-manifest satellite vehicle ready for integration on AEHF-6 mission

[7] The Atlas V Aft Buckhead Carrier - Requirements for the Small Satellite Designer

[8] Spaceflight Now: Atlas 5 launch timeline for the AEHF 6 mission

[9] In-The-Sky.org: TDO-2

[10] YouTube: Atlas V AEHF-6 Mission Profile

[11] Atlas V Users Guide 2010

[12] Wikipedia: Atlas V

[13] Atlas V AEHF-5 Mission Profile

[14] Atlas V AEHF-6 Mission Profile

[15] Wikipedia: Advanced Extremely High Frequency

[16] Wikipedia: Common Core Booster (Atlas V Stage 1)

[17] Wikipedia: Centaur (Atlas V Stage 2)