Mini-course:

Optimal Control of Space Trajectories using GEKKO

Lecture 2

Jhonathan Murcia-Piñeros

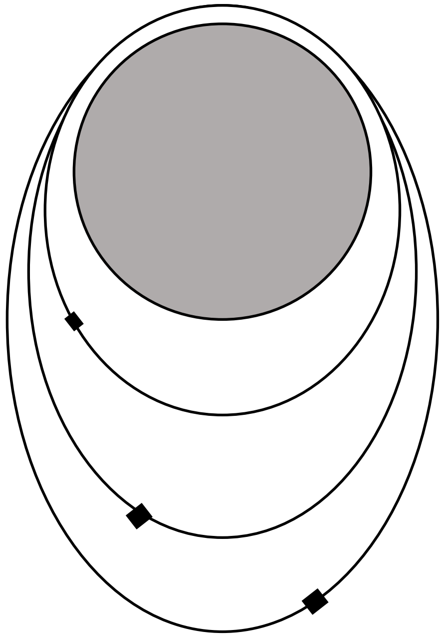
jhonathan.pineros@unifesp.br

CBDO - 2024

05/12/2024

Summary

I. Example – rocket launch.
II. Description of the script.
III. Low thrust maneuver.



I. Example to verification – rocket launch

- Source:

https://apmonitor.com/wiki/index.php/Apps/RocketLaunch

I. Example to verification – rocket launch

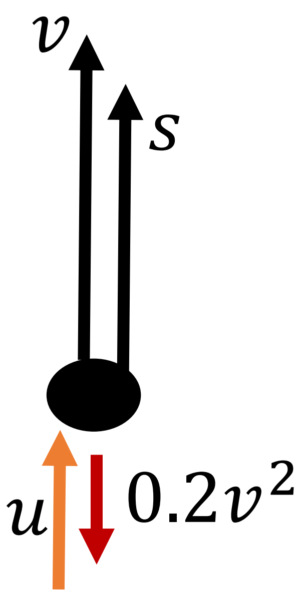
$$\boldsymbol{min(t_f)}$$

Subject to:

$$\dot{s} = v$$

$$\dot{v} = \frac{u - 0.2v^2}{m}$$

$$\dot{m} = -0.01u^2$$



$$0.0 \leq v(t) \leq 1.7$$

$$-1.1 \leq u(t) \leq 1.1$$

$$0.2 \leq m(t) \leq 1.0$$

$$s(t_0) = 0.0$$

$$v(t_0) = 0.0$$

$$m(t_0) = 1.0$$

$$s(t_f) = 10.0$$

$$v(t_f) = 0.0$$

II. Structure of the script

```python
import numpy as np
from gekko import GEKKO

# Call GEKKO
m = GEKKO()

# 1 unit of time in 101 points
m.time = np.linspace(0,1,101)

# optimal final time problem, initial value 1, lower 0.1, upper 100
tf = m.FV(value=1.0,lb=0.1,ub=100)
tf.STATUS = 1 #controlled
```

```python
# optimal final time problem, initial value 1, lower 0.1, upper 100
tf = m.FV(value=1.0,lb=0.1,ub=100)
tf.STATUS = 1 #controlled

# control variable
u = m.MV(value=0,lb=-1.1,ub=1.1)
u.STATUS = 1 #controlled by the program
u.DCOST = 1e-5 #Delta penalty control mov.

# variables
s = m.Var(value=0) # position
v = m.Var(value=0,lb=0,ub=1.7) # velocity
mass = m.Var(value=1,lb=0.2) # mass
```

```python
# differential equations
m.Equation(s.dt()==tf*v)
m.Equation(mass*v.dt()==tf*(u-0.2*v**2))
m.Equation(mass.dt()==tf*(-0.01*u**2))

# final conditions
m.fix(s, pos=len(m.time)-1,val=10.0)
m.fix(v, pos=len(m.time)-1,val=0.0)

# cost function
m.Obj(tf)
```

```python
# setup options for solver
m.options.NODES = 6 # Collocations points for poly
m.options.SOLVER = 3 #IPOPT
m.options.IMODE = 6 #OCP DYN OPT
m.options.MAX_ITER = 500
m.options.MV_TYPE = 0 #INTERPOLATION BETW END POINTS

# Solving
m.solve()
```

```
apm 193.19.205.166_gk_model0 <br><pre> ----------------------------------
 APMonitor, Version 1.0.3
 APMonitor Optimization Suite
 ----------------------------------------------------------------



 --------- APM Model Size ------------
 Each time step contains
   Objects      :            0
   Constants    :            0
   Variables    :            5
   Intermediates:            0
   Connections  :            4
   Equations    :            4
   Residuals    :            4
```

```
 Number of state variables:            3897
 Number of total equations: -          3800
 Number of slack variables: -             0
 ---------------------------------------
 Degrees of freedom       :              97

 **********************************************
 Dynamic Control with Interior Point Solver
 **********************************************
...
 Objective      :    3747.13119010218
 Successful solution
 ---------------------------------------------------
```

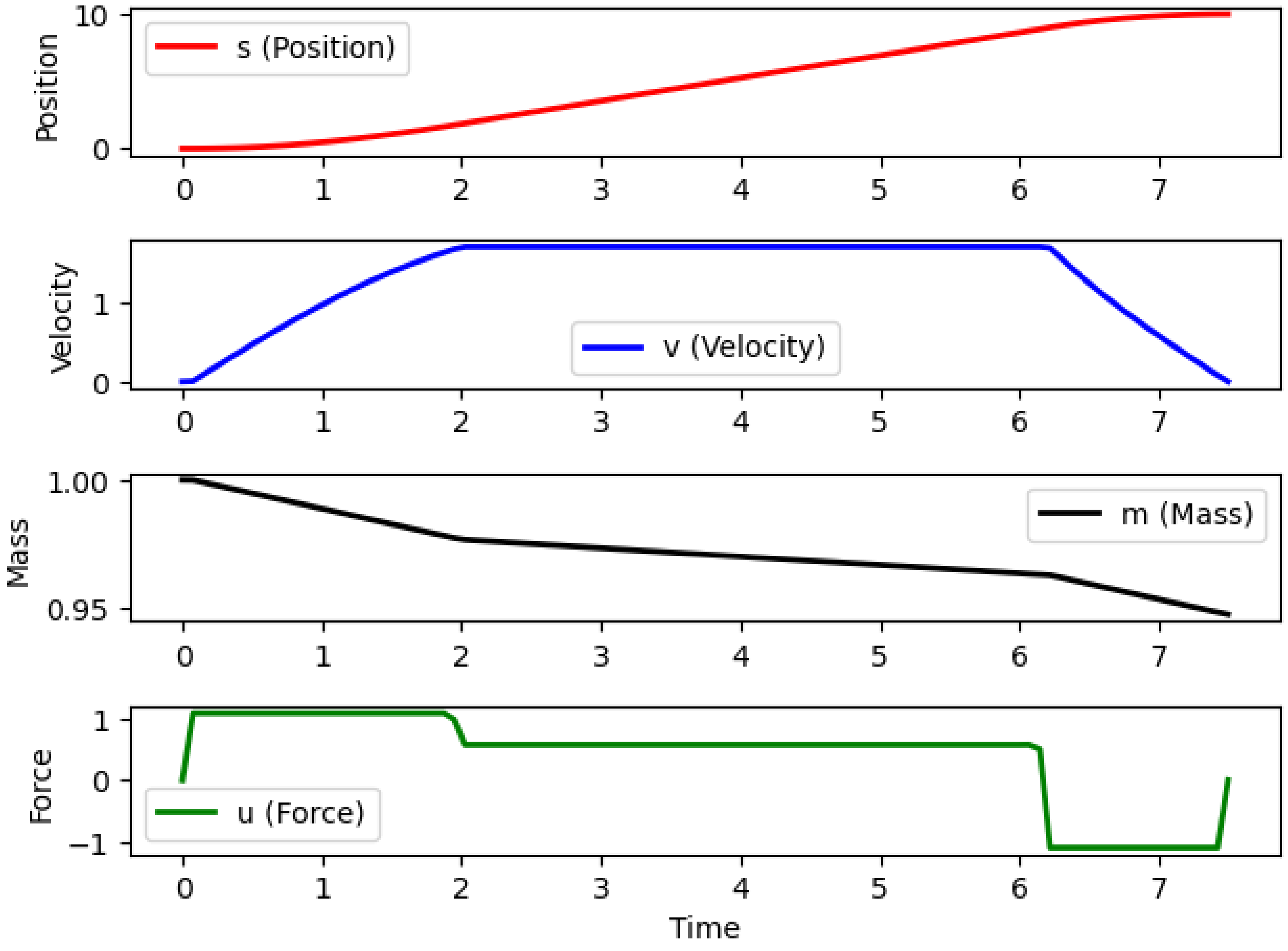
```python
print('Minimun final time: ' + str(tf.value[0]))
```

✓ 0.0s

```
Minimun final time: 7.4942617694
```

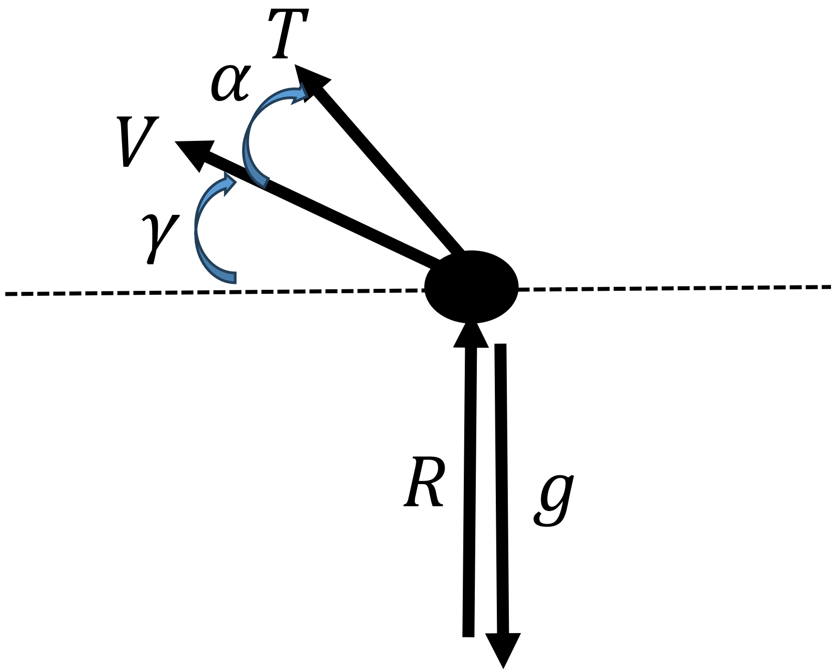
```python
import matplotlib.pyplot as plt

# scaled time
ts = m.time * tf.value[0]

# plot results
plt.figure(1)
plt.subplot(4,1,1)
plt.plot(ts,s.value,'r-',linewidth=2)
plt.ylabel('Position')
plt.legend(['s (Position)'])
```

```
plt.subplot(4,1,2)
plt.plot(ts,v.value,'b-',linewidth=2)
plt.ylabel('Velocity')
plt.legend(['v (Velocity)'])

plt.subplot(4,1,3)
plt.plot(ts,mass.value,'k-',linewidth=2)
plt.ylabel('Mass')
plt.legend(['m (Mass)'])

plt.subplot(4,1,4)
plt.plot(ts,u.value,'g-',linewidth=2)
plt.ylabel('Force')
plt.legend(['u (Force)'])

plt.xlabel('Time')
plt.tight_layout()
plt.show()
```



Position
s (Position)
10
0
Velocity
v (Velocity)
1
0
Mass
m (Mass)
1.00
0.95
Force
u (Force)
1
0
−1
0
1
2
3
4
5
6
7
Time

III. Low thrust orbital transfer



$$\dot{R} = V \sin\gamma \qquad (1)$$

$$\dot{\theta} = \frac{V \cos\gamma \cos A}{R \cos\varphi} \qquad (2)$$

$$\dot{\varphi} = \frac{V \cos\gamma \sin A}{R} \qquad (3)$$

$$\dot{V} = \frac{T \cos\alpha}{m} - g \sin\gamma \qquad (4)$$

$$\dot{\gamma} = \frac{T \sin\alpha}{mV} - \frac{g \cos\gamma}{V} + \frac{V \cos\gamma}{R} \qquad (5)$$

$$\dot{A} = -\frac{V \tan\varphi \cos A \cos\gamma}{R} \qquad (6)$$

$$\dot{m} = -\frac{T}{V_{sp}} \qquad (7)$$

$$\boldsymbol{U}\{T(t), \alpha(t)\} \qquad (8)$$

The Optimal Control Problem - Constraints

$$t_0 < t \le t_f$$

$$500\ s \le t_f \le 2000\ s$$

$$400\ km \le h(t) \le 402\ km$$

$$V_{cir} \le V(t_f)$$

$$-180\ deg \le \theta(t) \le 180\ deg$$

$$-90\ deg \le \varphi(t) \le 90\ deg$$

$$-20\ deg \le \gamma(t) \le 20\ deg$$

$$0\ deg \le A(t) \le 359.9\ deg$$

$$m_{min} \le m(t) \le m_{max}$$

$$-180\ deg \le \alpha(t) \le 180\ deg$$

$$0\ N \le T(t) \le 0.5\ N$$

- Initial conditions:
  - Altitude: 400 km
  - Velocity: 7802.88 m/s
  - FPA: 0°
- Final conditions:
  - Altitude: 402 km

Objective: Final time minimization.

$$\boldsymbol{J} = \min_{t_f}(t)$$

Script

```python
from gekko import GEKKO
import numpy as np
import matplotlib.pyplot as plt
import math
```

```python
# GEKKO Initialization --------------------------
m = GEKKO()

# Time parameters ------------------------------
nt = 151
tm = np.linspace(0,1,nt)
m.time = tm

p = np.zeros(nt)
p[-1] = 1.0
final = m.Param(value=p)
```

```python
# Parameters and Const ------------------------------------------------------
pi    = math.pi
pi2   = pi/2.0
deg2rad = pi/180.0

# Planet info
mu    = 3.986031954093051e14         # earth gravitational param (m^3/s^2)
Re    = 6378135.                    # mean radius of the earth (m)
g0    = 9.8                          # mean gravity at the SML (m/s^2)

# Atm info
rho0 = 1.217     # msl atmospheric density (kg/m^3)
H     = 8500     # atm scale height (m)

# Vehicle info
Surf  = 332.13              # Surface area m**2
mass  = 180.0 #21800.0      # spacecraft mass (kg)
A2m   = Surf/mass
propmass = 20.0
```

```
# Propulsion info
isp  = 230                             # Isp (s)
Ve   = isp*g0                          # Rocket motor especific velocity
mdot = 0.0
```

```
# Initial boundary conditions at (t0)   ---------------------------------
h0    = 400000                            # initial altitude (m)
R0    = Re+h0
LONG0 = 0
LAT0  = 0
V0    = 7668.585794677108                 # inital velocity in m/s
FPA0  = 0.0*deg2rad
AZI0  = pi2
msp0  = mass
```

```
# Final boundary conditions at (tf)  ---------------------------------
hf    = 402000                              # final desired altitude (m)
Rf    = Re+hf
Vf    = 7667.454673925565                # final velocity in m/s
FPAf  = 0.0*deg2rad                      # final FPA
mspf   = mass-propmass
```

```python
# Variables constraints -----------
# State vector
Ru = Rf
Rl = Re+75000

LONGl = 0
LONGu = pi

LATl = -pi2
LATu = pi2

Vl = Vf
Vu = V0

FPAl = -20*deg2rad
FPAu = -FPAl

AZIl = 0
AZIu = pi

mspu = msp0
mspl = mspf

#Manipulated variables ------------
Thrul = 0
Thruu = 0.5 #N
```

```python
#Time guess -------------------------------------------------------
Timel = 500.0 #(s) 50
Timeu = 2000.0 #(s) #800
```

```python
# Initialization and path constraints ------------------------------------
r    = m.Var (value=R0, lb=Rl, ub=Ru)          # Radio or altitude (m)
long = m.Var (value=LONG0, lb=LONGl, ub=LONGu) # Longitude angle (rad)
lat  = m.Var (value=LAT0, lb=LATl, ub=LATu)    # Latitude (rad)
v    = m.Var (value=V0, lb=6000)               # Velocity (m/s)
fpa  = m.Var (value=FPA0, lb=FPAl, ub=FPAu)    # Flight Path Angle
azi  = m.Var (value=AZI0, lb=AZIl, ub=AZIu)    # Azimuth
msp  = m.Var (value=msp0, lb=mspl, ub=mspu)
```

```python
# Final time
Tf = m.FV(lb=Timel,ub=Timeu); Tf.STATUS = 1

# Manipulated variables  -------------------------
Thru = m.MV (lb=Thrul, ub=Thruu)
Thru.STATUS = 1

AT = m.MV (value=0.0, lb=-pi, ub=pi)
AT.STATUS = 1
```

```python
# Aditional variables  ******************************************************
# Use m.Intermediate() to save and NOT USE for control and/nor MV
A2m  = m.Intermediate(Surf/msp)
# Gravity ----------------------------------------
g    = m.Intermediate(mu/r**2)                        # local gravity (m/s^2)
alt  = m.Intermediate(r-Re)                           # Local altitude (m)
# Atm -------------------------------------------
rho  = m.Intermediate(rho0*m.exp(-alt/H))             # local density (kg/m^3)
pdyn = m.Intermediate(0.5*rho*v**2)                   # dynamic pressure/mass
# Aero ------------------------------------------
cl = 0.0
cd = 0.0

L2m  = m.Intermediate(cl*pdyn*A2m)                    # lift acceleration (m/s^2)
D2m  = m.Intermediate(cd*pdyn*A2m)                    # drag acceleration (m/s^2)
# T to mass -------------------------------------
T2m  = m.Intermediate(Thru/msp)                       # Thrust to mass ratio (m/s^2)


gf   = m.Intermediate(L2m/g)                          #n load or g forces
Qhc = m.Intermediate(1.83e-8*m.sqrt(rho/0.5)*v**3)
a_qr = 1.072e6*(v**-1.88)*(rho**-0.325)

Qhr = m.Intermediate(4.736e4*(0.5**a_qr)*(rho**1.22)*359)
c_acc = m.Intermediate(v**2/r)                        #centrifugal acc
v_esc = m.Intermediate(m.sqrt(2*g*r))
BA = 0.0
```

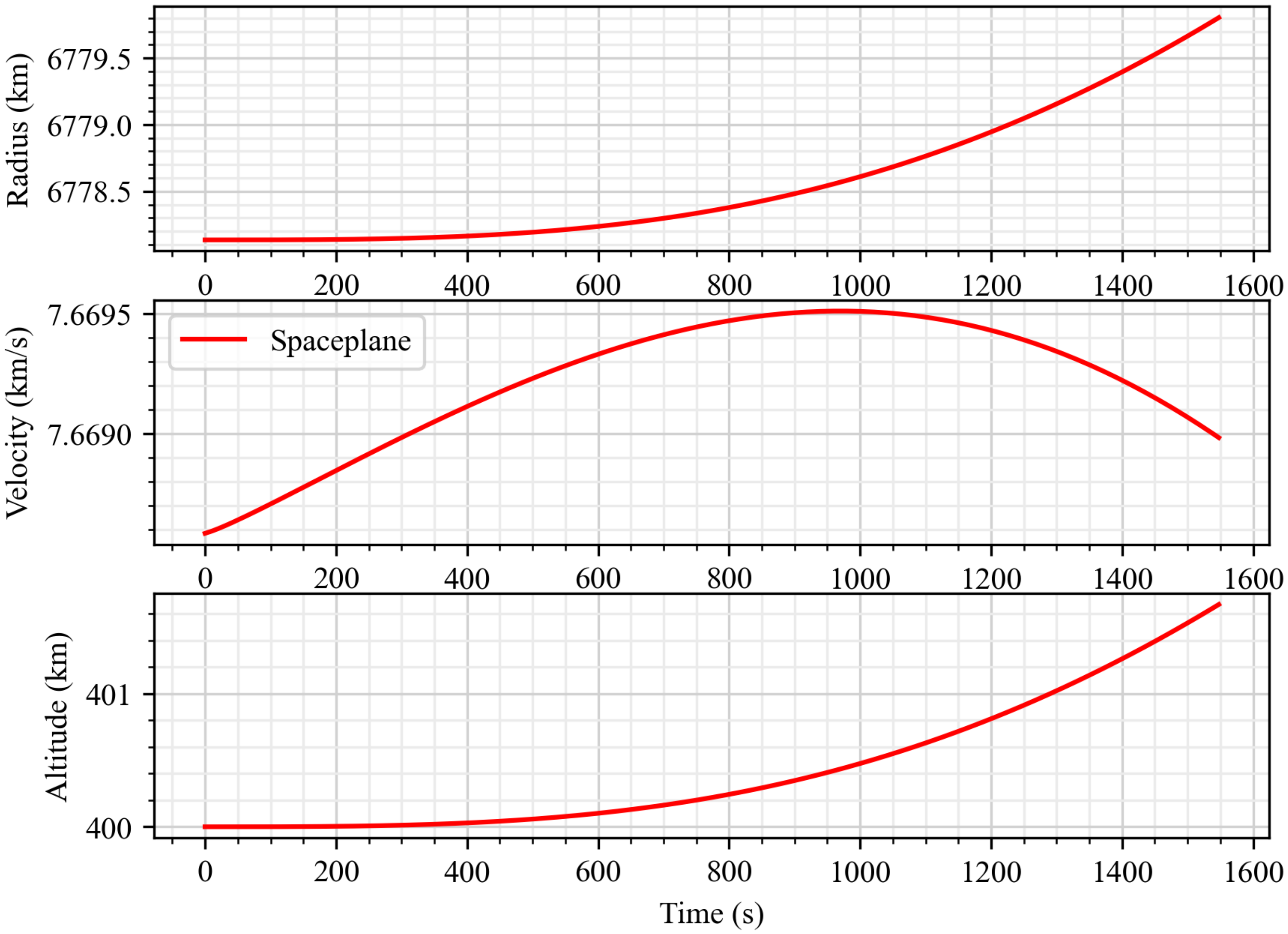
```
# Process model / EDOs/ DAEs ****************************************************************************
m.Equation(r.dt()/Tf == v*m.sin(fpa))
m.Equation(r*m.cos(lat)*long.dt()/Tf == v*m.cos(fpa)*m.sin(azi))
m.Equation(r*lat.dt()/Tf == v*m.cos(fpa)*m.cos(azi))
m.Equation(v.dt()/Tf == T2m*m.cos(AT)-D2m-g*m.sin(fpa))
m.Equation(v*fpa.dt()/Tf == T2m*m.sin(AT)+L2m*m.cos(BA)-(g-v**2/r)*m.cos(fpa))
m.Equation(v*r*m.cos(fpa)*azi.dt()/Tf == r*L2m*m.sin(BA) + (v*m.cos(fpa))**2*m.sin(azi)*m.tan(lat))
m.Equation(Ve*msp.dt()/Tf == -Thru)

# Objetive funtion ---------------------------------------------------------------------------------------
m.Minimize(final)

# Setup solution -----------------------------------------------------------------------------------------
m.options.IMODE    = 6     #  4 for SIM and 6 for Optimal Control
m.options.MAX_ITER = 2000  # Control of MAX ITER
m.options.NODES    = 2     # WITH 2 IS OK Collocation nodes
##m.options.OTOL      = 1e-3  # Iter Tolerance obje, same as GPOPS
##m.options.RTOL      = 1e-3  # Real tol eq
#m.options.SOLVER    = 3     # solver (IPOPT)

m.solve(disp=True) #True

# get additional solution information ------------------------------------------
import json
with open(m.path+'//results.json') as f:
    results = json.load(f)
```

```
apm 185.153.176.175_gk_model0 <br><pre> ------------------------
 APMonitor, Version 1.0.3
 APMonitor Optimization Suite
 ----------------------------------------------------------------



 --------- APM Model Size ------------
 Each time step contains
   Objects      :           0
   Constants    :           0
   Variables    :          11
   Intermediates:          13
   Connections  :           0
   Equations    :          21
   Residuals    :           8

 Number of state variables:          3601
 Number of total equations: -        3300
```

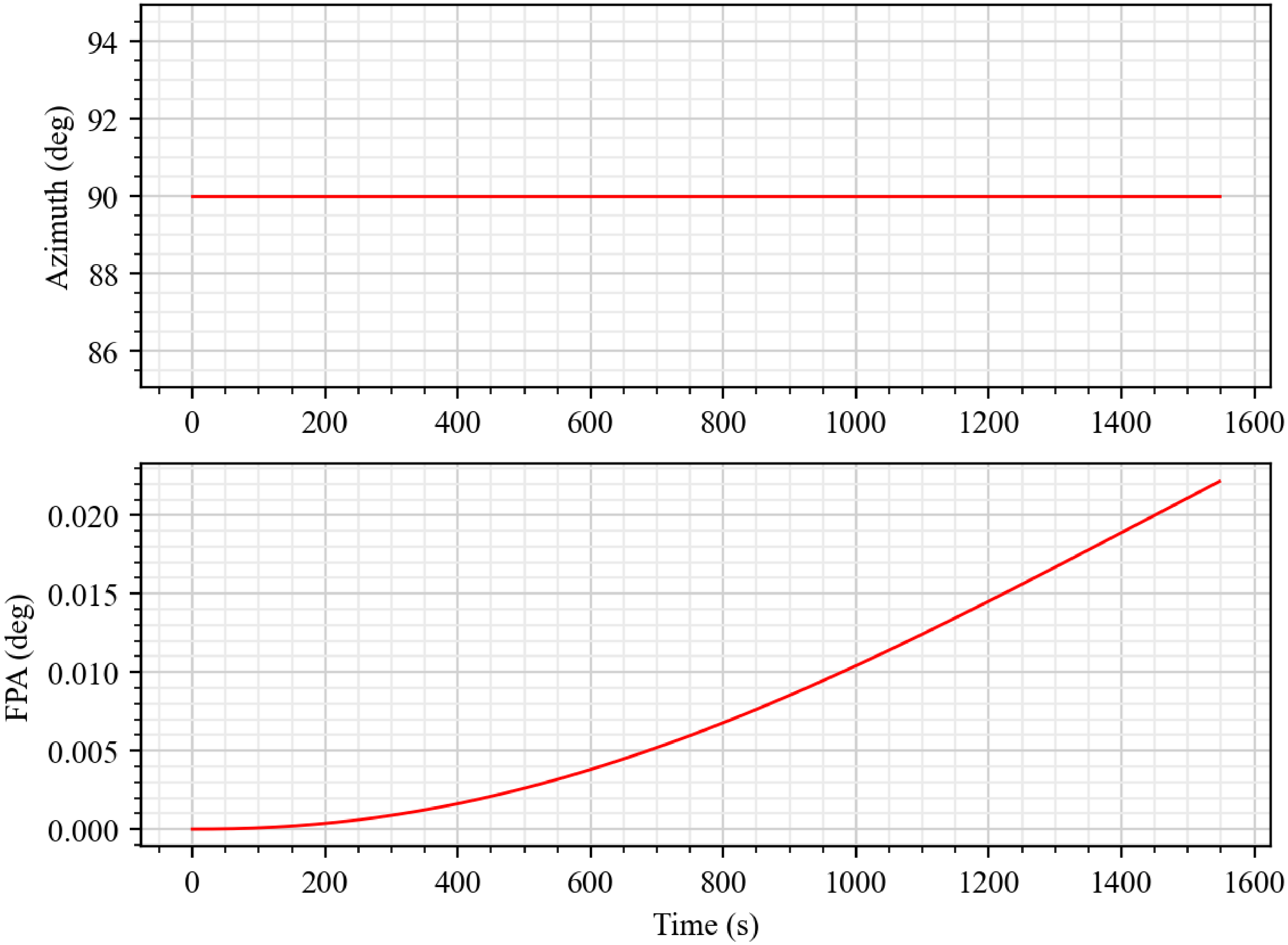
```
Number of Iterations....: 53

                                   (scaled)                 (unscaled)
Objective...............:   1.0000592927938146e+00    1.0000592927938146e+00
Dual infeasibility......:   2.8295922815122290e-11    2.8295922815122290e-11
Constraint violation....:   8.9391858100737346e-10    1.2118201642152826e-07
Complementarity.........:   4.7367067492974097e-08    4.7367067492974097e-08
Overall NLP error.......:   4.7367067492974097e-08    1.2118201642152826e-07
```

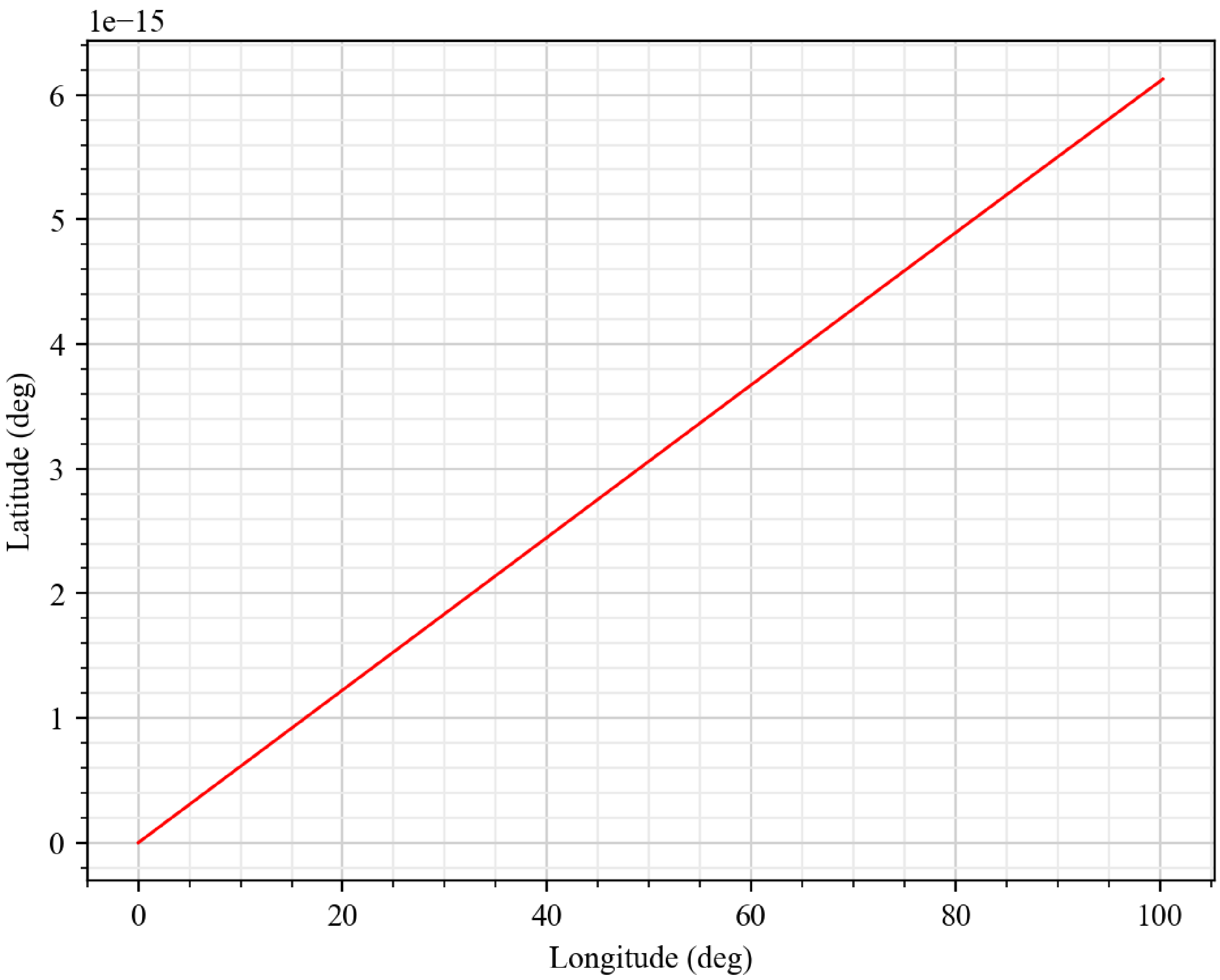
```
Number of inequality constraint evaluations          = 54
Number of equality constraint Jacobian evaluations   = 54
Number of inequality constraint Jacobian evaluations = 54
Number of Lagrangian Hessian evaluations             = 53
Total CPU secs in IPOPT (w/o function evaluations)   =      0.723
Total CPU secs in NLP function evaluations           =      2.362

EXIT: Optimal Solution Found.

 The solution was found.

 The final value of the objective function is    1.00005929279381

 ---------------------------------------------------
 Solver         :  IPOPT (v3.12)
 Solution time  :    3.19769999998971      sec
 Objective      :    1.00005929279381
 Successful solution
```

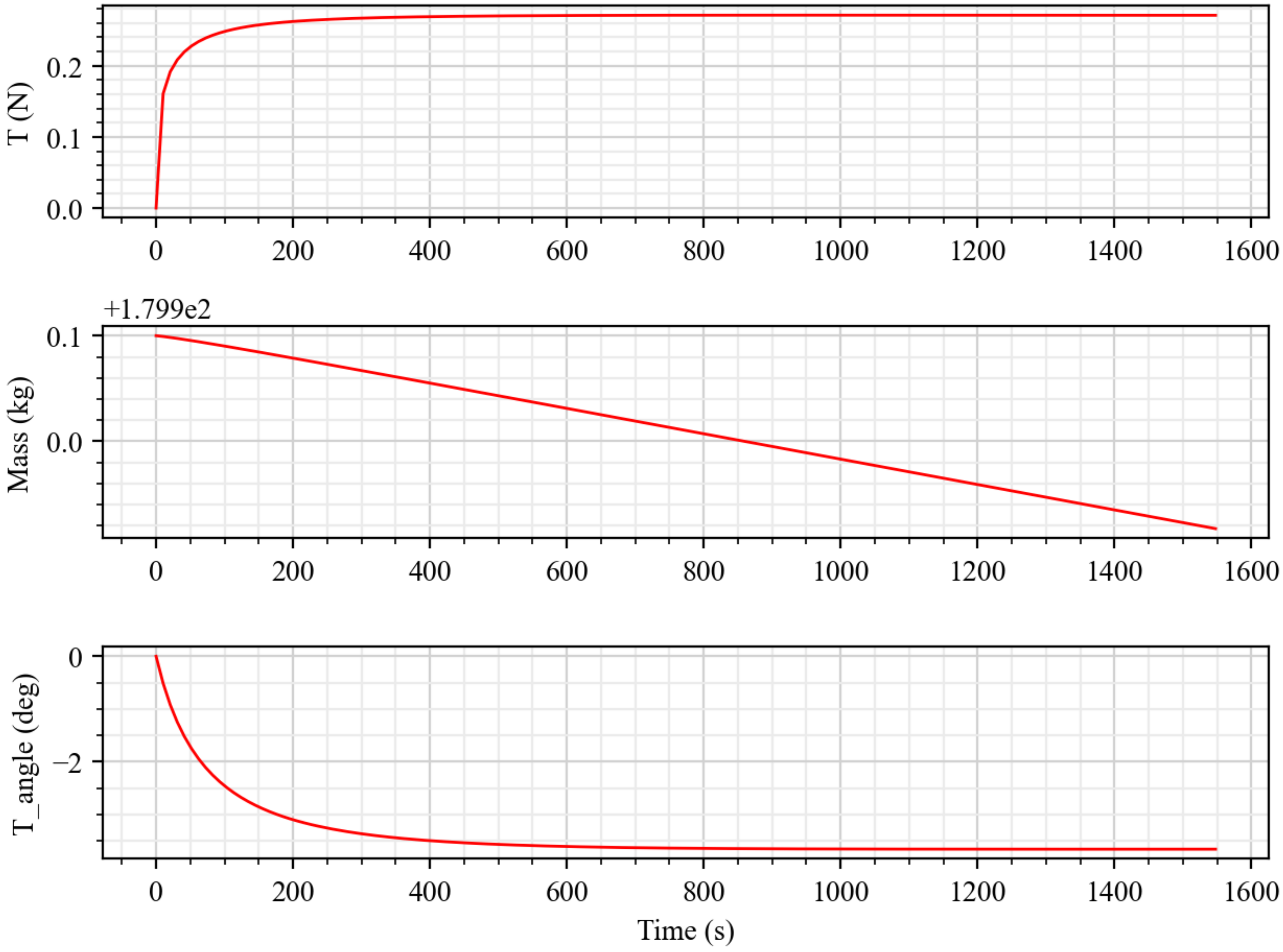
Results



Radius (km)
6779.5
6779.0
6778.5
0
200
400
600
800
1000
1200
1400
1600
Velocity (km/s)
7.6695
7.6690
Spaceplane
Altitude (km)
401
400
Time (s)


Azimuth (deg)
94
92
90
88
86
0
200
400
600
800
1000
1200
1400
1600
FPA (deg)
0.020
0.015
0.010
0.005
0.000
Time (s)


1e−15
6
5
4
3
2
1
0
Latitude (deg)
0
20
40
60
80
100
Longitude (deg)


T (N)
0.0
0.1
0.2
+1.799e2
Mass (kg)
0.1
0.0
T_angle (deg)
0
−2
0
200
400
600
800
1000
1200
1400
1600
Time (s)

Fixing final conditions

- Final conditions:
  - Altitude: 401 km
  - Velocity: 7667.45 m/s
  - FPA: 0°

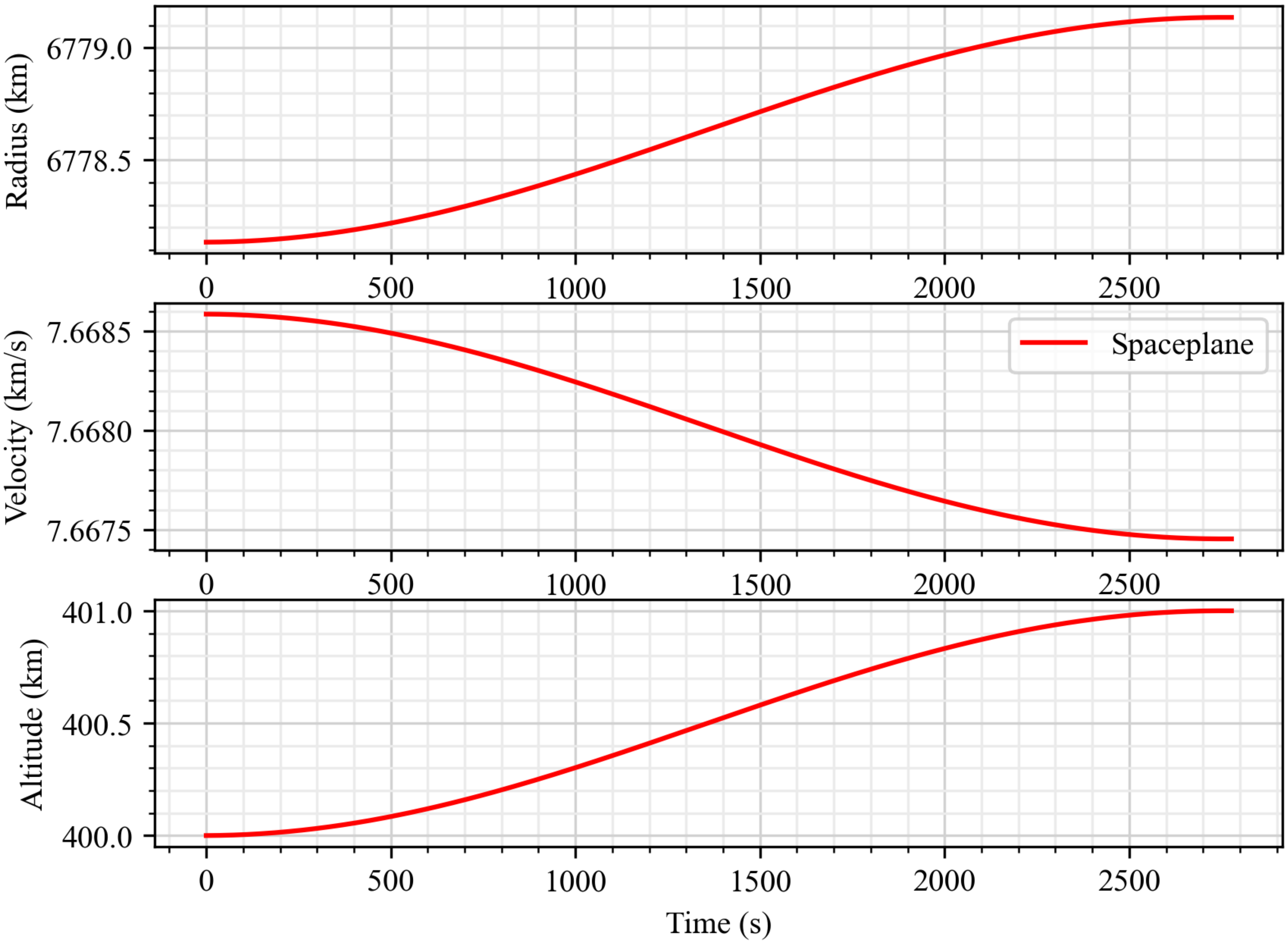
$$t_0 < t \leq t_f$$

$$2000\ s \leq t_f \leq 4000\ s$$

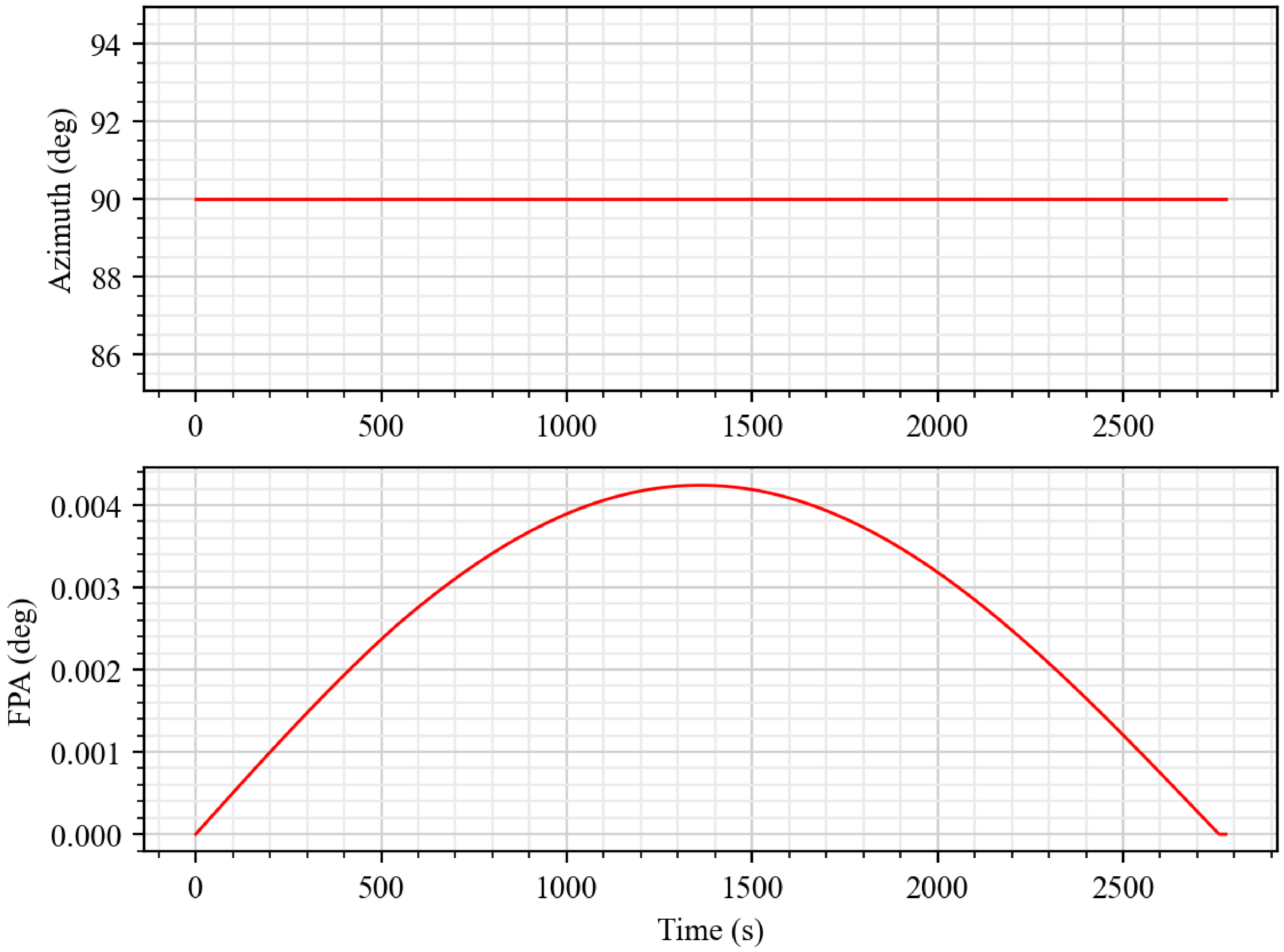
Objective: Final time minimization.

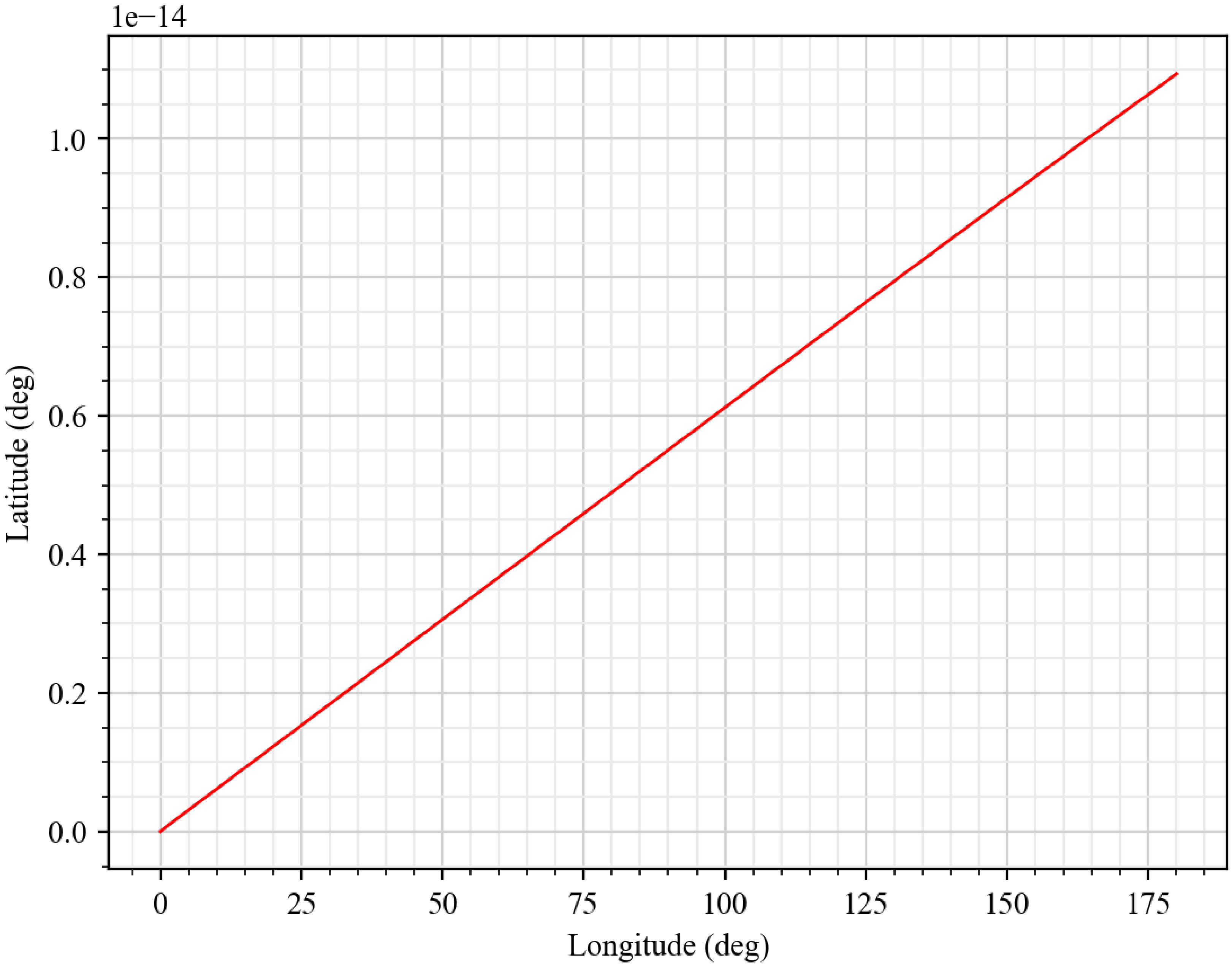
$$\boldsymbol{J} = \min_{t_f}(t)$$

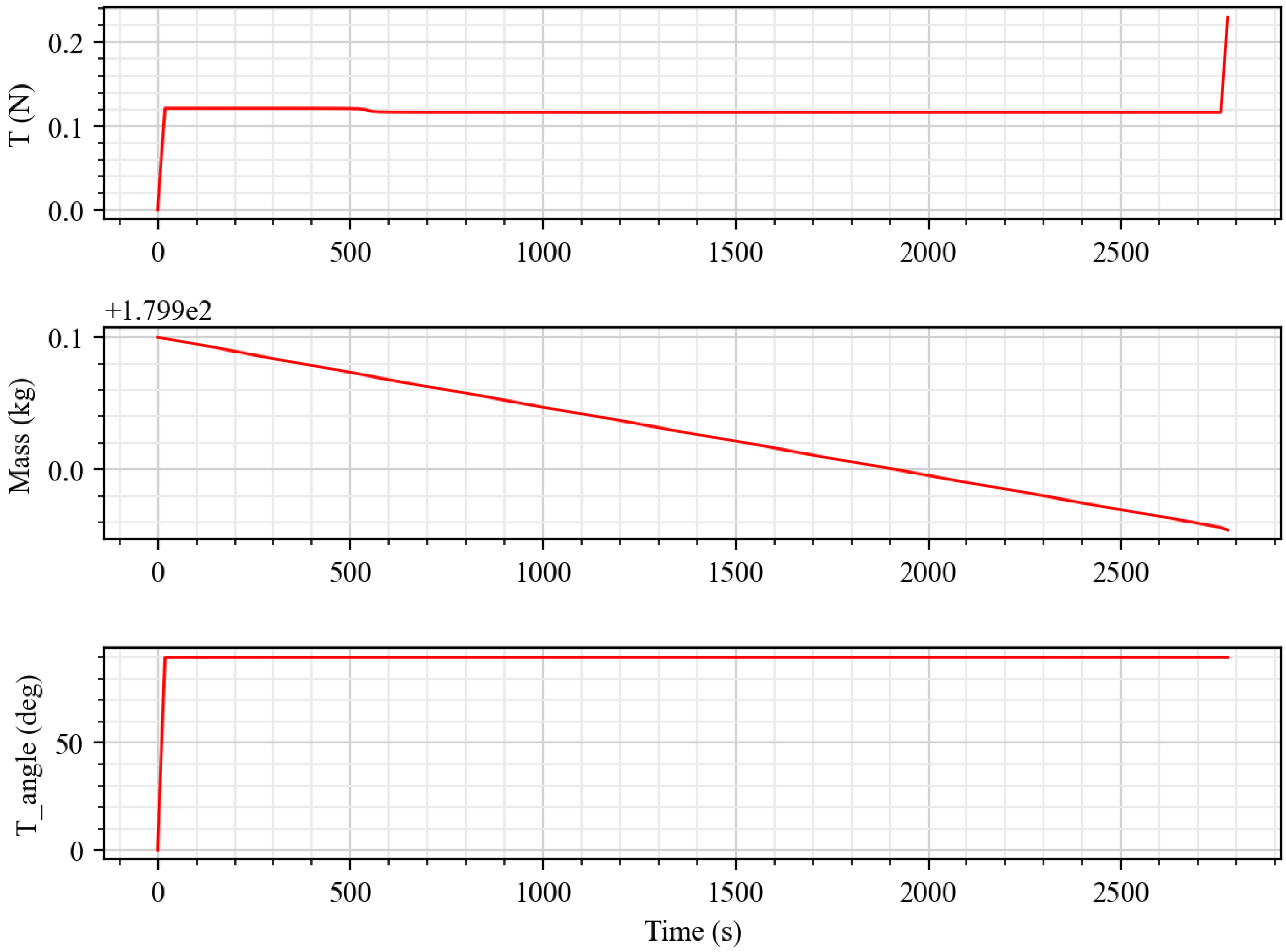
```
# Fixed variables at tf ----------
m.fix_final(r, Rf)
m.fix_final(v, Vf)
m.fix_final(fpa, FPAf)
```



Radius (km)
6779.0
6778.5
Velocity (km/s)
7.6685
7.6680
7.6675
Spaceplane
Altitude (km)
401.0
400.5
400.0
0
500
1000
1500
2000
2500
Time (s)


94
92
90
88
86
Azimuth (deg)
0
500
1000
1500
2000
2500
0.004
0.003
0.002
0.001
0.000
FPA (deg)
0
500
1000
1500
2000
2500
Time (s)


1e−14
1.0
0.8
0.6
0.4
0.2
0.0
Latitude (deg)
0
25
50
75
100
125
150
175
Longitude (deg)



T (N)
0.2
0.1
0.0
0
500
1000
1500
2000
2500
+1.799e2
Mass (kg)
0.1
0.0
0
500
1000
1500
2000
2500
T_angle (deg)
50
0
0
500
1000
1500
2000
2500
Time (s)

References

[1] Armellin, R., Lavagna, M., and Ercoli-Finzi, A.,(2006) Aero-gravity assist maneuvers: controlled dynamics modeling and optimization, Celestial Mechanics and Dynamical Astronomy, Vol. 95,, pp. 391–405. doi: 10.1007/s10569-006-9024-y

[2] Betts J. (2010) Practical Methods for Optimal Control Using Nonlinear Programming. 3 Ed..

[3] Patterson, M., Rao, A. (2016) GPOPS II Manual. V 2.3. Available at: https://gpops2.com/resources/gpops2UsersGuide.pdf

[4] Murcia-Piñeros, J., Prado, A. F., Dimino, I., & de Moraes, R. V. (2024). Optimal gliding trajectories for descent on Mars. Applied Sciences, 14(17), 7786. Doi: 10.3390/app14177786

[5] Piñeros, J. O. M., Bevilacqua, R., Prado, A. F., & de Moraes, R. V. (2024). Optimizing aerogravity-assisted maneuvers at high atmospheric altitude above Venus, Earth, and Mars to control heliocentric orbits. Acta Astronautica, 215, 333-347. Doi: 10.1016/j.actaastro.2023.12.017

[6] Murcia-Piñeros, J., Bevilacqua, R., Gaglio, E., Prado, A. B., & De Moraes, R. V. (2024). Optimization of Aero-gravity assisted maneuvers for spaceplanes at high atmospheric flight on Earth. In AIAA SCITECH 2024 Forum (p. 1459). Doi: 10.2514/6.2024-1459.

[7] Beal, L.D.R., Hill, D., Martin, R.A., and Hedengren, J. D., “GEKKO Optimization Suite”, Processes, Vol. 6, No. 8, 2018. doi: 10.3390/pr6080106.

[8] Hedengren, J. D., Asgharzadeh Shishavan, R., Powell, K.M., and Edgar, T.F., “Nonlinear Modeling, Estimation and Predictive Control in APMonitor”, Computers and Chemical Engineering, Vol. 70, 2014, pp. 133–148. doi: 10.1016/j.compchemeng.2014.04.013.

Thank you!

Questions?