HW 4

Trajectory Optimization II: Optimal Orbit Altitude in Polar Coordinates and Determining Optimal Orbit for a Fuel-Minimizing Burn-Coast-Burn Maneuver

Justin Self

AERO557: Advanced Orbital Mechanics

March 12, 2024

Problem 1

Optimal Orbit to Maximize Final Altitude

Consider the case of a transfer in the two-body force field with polar coordinates where a is the non-dimensional applied acceleration and beta is the thrust angle. The spacecraft is initially in a circular orbit with radius of 1.05. A constant acceleration of 0.1 is then applied for a period of 4.0. The objective is to place the spacecraft into a final circular orbit with maximum possible altitude. Determine the orbit transfer. Equations of motion are:

$$\ddot{r} - r\dot{\theta}^2 = -\frac{1}{r^2} + a sin\beta$$

$$r\ddot{\theta} + 2\dot{r}\dot{\theta} = a cos\beta$$

Provide:

- The problem formulation including the state and co-state, boundary conditions, and the control law.
- The plot of the thrust angle history
- A plot of the transfer trajectory
- The initial and final values of the state and co-state variables.
- Value of the performance index

Solution.

See Appendix A for results and code for Problem 1

See following pages for the problem formulation carried out by hand, followed by `MATLAB` outputs.

AERO 557
Homework #4

Problem 1: Optimal Orbit to maximize the final altitude

Consider the case of a transfer in the two-body force field with polar coordinates where a is the non-dimensional applied acceleration and beta is the thrust angle. The spacecraft is initially in a circular orbit with radius of 1.05. A constant acceleration of 0.1 is then applied for a period of 4.0. The objective is to place the spacecraft into a final circular orbit with maximum possible altitude. Determine the orbit transfer. Equations of motion are:

$$\ddot{r} - r\dot{\theta}^2 = -\frac{1}{r^2} + a\sin\beta$$
$$r\ddot{\theta} + 2\dot{r}\dot{\theta} = a\cos\beta$$

Provide:
-The problem formulation including the state and co-state, boundary conditions, and the control law.
-The plot of the thrust angle history
-A plot of the transfer trajectory
-The initial and final values of the state and co-state variables.
-Value of the performance index

i.) K OR L ?

↳ MAXIMIZE ALTITUDE. POLAR COORDINATES!

$J = -r$ * MAXIMIZING ORBITAL RADIUS IN CIRCULAR ORBIT MAXIMIZES ALTITUDE

GIVEN: $a_0 = 1.05$ DU

APPLIED FOR: 4.0 TU

EQUATIONS OF MOTION:

$$\ddot{r} - r\dot{\theta}^2 = -\frac{1}{r^2} + a_{cc}\sin\beta \qquad (1)$$

($\ddot{r}$: ACCELERATION; $r\dot{\theta}^2$: TA; $\frac{1}{r^2}$: ORBIT RADIUS (CIRCLE); $a_{cc}\sin\beta$: STEERING ANGLE)

$$r\ddot{\theta} + 2\dot{r}\dot{\theta} = a_{cc}\cos\beta \qquad (2)$$

ii.) CREATE $\bar{f}$

$$\bar{s} = \begin{bmatrix} r \\ \theta \\ \dot{r} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix}$$

← STATE

CONTROL = $\bar{c} = [c_1] = [\beta]$ STEERING ANGLE

FROM EOMS AND $\bar{s}$ ↘

$$\bar{f} = \dot{\bar{s}} = \begin{bmatrix} \dot{r} \\ \dot{\theta} \\ \ddot{r} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} s_3 \\ s_4 \\ r\dot{\theta}^2 - \frac{1}{r^2} + a_{cc}\sin\beta \\ \frac{1}{r}\left[-2\dot{r}\dot{\theta} + a_{cc}\cos\beta \right) \end{bmatrix}$$

$$\bar{f} = \dot{\bar{s}} = \begin{bmatrix} s_3 \\ s_4 \\ s_1 s_4^2 - \frac{1}{s_1^2} + a_{cc}\sin c_1 \\ \frac{1}{s_1}\left[a_{cc}\cos c_1 - 2 s_3 s_4 \right] \end{bmatrix}$$

:::.) BUILD CONSTRAINTS

SINCE CIRCULAR, TA = ANYTHING

$\therefore TA_0 = 0^o$

KEPLERIAN ELEMENTS:

$$r = \frac{a\left(1 - ecc^2\right)}{1 - ecc(\cos\theta)} = a$$

$$\dot{r} = \frac{ecc\sin\theta}{a^{1/2}\left(1 - ecc^2\right)^{1/2}} = 0$$

$$\dot{\theta} = \frac{\left(1 + ecc\cos\theta\right)^2}{\left(a\left[1 - ecc^2\right]\right)^{3/2}} = a^{-3/2}$$

FINAL CONSTRAINTS.

GOING TO A CIRCULAR ORBIT. ←

$\therefore \dot{r}_f = 0$

$\dot{\theta}_f = a^{-3/2}$ * CODE UP $\dot{\theta}^2 a^3 = 1$

$$\bar{\Omega} = \begin{bmatrix} s_{3f} \\ s_{4f} \end{bmatrix} = \begin{bmatrix} \dot{r} \\ a^{-3/2} \end{bmatrix} = \begin{bmatrix} s_{3f} \\ s_{4f}^2 s_{1f}^3 \end{bmatrix}$$

$\dot{\theta}_f \quad r_f$

* $s_{1f} \equiv r_f$. TO OPTIMIZE

$s_{2f} \equiv \theta_f$. FREE.

$\therefore$ ONLY CONSTRAINTS ARE s_{3f}, s_{4f} FINAL

iv.) BUILD BOLZA AND HAMILTONIAN.

$$H = \bar{\lambda}^T \bar{f}$$

$$H = \lambda_1 s_3 + \lambda_2 s_4 + \lambda_3 \left(s_1 s_4^2 - \frac{1}{s_1^2} + a_{cc} \sin c_1 \right) + \cdots$$

$$\lambda_4 \left(\frac{1}{s_1} \left[a_{cc} \cos c_1 - 2 s_3 s_4 \right] \right)$$

CONTROL LAW

$$\frac{\delta H}{\delta c_1} = a_{cc} \cos c_1 \lambda_3 - \lambda_4 \left(\frac{1}{s_1} a_{cc} \sin c_1 \right) = 0$$

$$a_{cc} \cos c_1 \lambda_3 = \lambda_4 \left(\frac{1}{s_1} a_{cc} \sin c_1 \right)$$

$$\tan c_1 = \frac{s_1 \lambda_3}{\lambda_4} \longleftarrow \quad \therefore \boxed{c_1 = \tan^{-1}\left[\frac{s_1 \lambda_3}{\lambda_4}\right]}$$

CONTROL LAW

GET $\cos c_1$ AND $\sin c_1$ IN TERMS OF s_1, λ_3, AND λ_4 ONLY.

$$\cos c_1 = \frac{\lambda_4 \sin c_1}{\lambda_3 s_1}$$

$$\frac{\sin^2\theta}{\sin^2\theta} + \frac{\cos^2\theta}{\sin^2\theta} = \frac{1}{\sin^2\theta}$$

$$1 + \frac{\cos^2\theta}{\sin^2\theta} = \frac{1}{\sin^2\theta}$$

$$\sin\theta = \left(\frac{1}{1 + [\tan^2\theta]^{-1}}\right)^{1/2}$$

$$\sin c_1 = \frac{1}{\left(1 + \frac{1}{\tan^2 c_1}\right)^{1/2}}$$

$$\tan c_1 = \frac{s_1 \lambda_3}{\lambda_4} = \frac{1}{\left(\frac{\tan^2 c_1}{\tan^2 c_1} + \frac{1}{\tan^2 c_1}\right)^{1/2}}$$

$$= \frac{\tan c_1}{(\tan^2 c_1 + 1)^{1/2}}$$

$$= \frac{\frac{s_1 \lambda_3}{\lambda_4}}{\left(\frac{s_1^2 \lambda_3^2}{\lambda_4^2} + \frac{\lambda_4^2}{\lambda_4^2}\right)^{1/2}}$$

$$= \frac{\frac{s_1 \lambda_3}{\lambda_4} \cdot \left(\pm \lambda_4 \right)}{\left(s_1^2 \lambda_3^2 + \lambda_4^2 \right)^{1/2}}$$

USE NEGATIVE ROOT.

$$\therefore \sin c_1 = \frac{- s_1 \lambda_3}{\left(s_1^2 \lambda_3^2 + \lambda_4^2 \right)^{1/2}} \leftarrow$$

SIMILARLY,

$$\frac{\sin^2\theta}{\cos^2\theta} + \frac{\cos^2\theta}{\cos^2\theta} = \frac{1}{\cos^2\theta}$$

$$\tan^2\theta + 1 = \frac{1}{\cos^2\theta}$$

$$\cos^2\theta = \left(\tan^2\theta + 1 \right)^{-1}$$

$$\cos\theta = \left(\tan^2\theta + 1 \right)^{-1/2}$$

$$\cos c_1 = \frac{1}{\left(\frac{s_1^2 \lambda_3^2}{\lambda_4^2} + 1 \right)^{1/2}}$$

$$= \frac{1}{\left(\frac{s_1^2 \lambda_3^2 + \lambda_4^2}{\lambda_4^2} \right)^{1/2}}$$

USE NEGATIVE ROOT.

$\pm \lambda_4$

$$\cos c_1 = \frac{- \lambda_4}{\left(s_1^2 \lambda_3^2 + \lambda_4^2 \right)^{1/2}}$$

PUT THESE EXPRESSIONS BACK INTO THE HAMILTONIAN:

$$H = \lambda_1 s_3 + \lambda_2 s_4 + \lambda_3 \left(s_1 s_4^2 - \frac{1}{s_1^2} + a_{cc} \sin c_1 \right) + \cdots$$

$$\lambda_4 \left(\frac{1}{s_1} \left[a_{cc} \cos c_1 - 2 s_3 s_4 \right] \right)$$

$$H = \lambda_1 s_3 + \lambda_2 s_4 + \lambda_3 \left(s_1 s_4^2 - \frac{1}{s_1^2} + a_{cc} \frac{-s_1 \lambda_3}{\left(s_1^2 \lambda_3^2 + \lambda_4^2 \right)^{1/2}} \right) + \cdots$$

$$\lambda_4 \left(\frac{1}{s_1} \left[a_{cc} \frac{-\lambda_4}{\left(s_1^2 \lambda_3^2 + \lambda_4^2 \right)^{1/2}} - 2 s_3 s_4 \right] \right)$$

NOW, FIND $\dot{\lambda} = -\frac{\partial H}{\partial s}$

* USING WOLFRAM MATHEMATICA, WE OBTAIN:

MATHEMATICA

$$\dot{\lambda} = \begin{bmatrix} -\frac{2\lambda 3 + s1^3\, s4^2\, \lambda 3 + 2\, s1\, s3\, s4\, \lambda 4}{s1^3} + \frac{acc\left(\lambda 3^2 - \frac{\lambda 4^2}{s1^2}\right)}{sqrt\left[s1^2 \lambda 3^2 + \lambda 4^2\right]} - \frac{2\, acc\, \lambda 3^2 \left(s1^2 \lambda 3^2 + \lambda 4^2\right) sqrt'\left[s1^2 \lambda 3^2 + \lambda 4^2\right]}{sqrt\left[s1^2 \lambda 3^2 + \lambda 4^2\right]^2} \\ 0 \\ -\lambda 1 + \frac{2\, s4\, \lambda 4}{s1} \\ -\lambda 2 - 2\, s1\, s4\, \lambda 3 + \frac{2\, s3\, \lambda 4}{s1} \end{bmatrix}$$

* a_{cc} = ACCELERATION

MATLAB.

$$\dot{\lambda} = \begin{bmatrix} -\dfrac{\overbrace{\left(2\lambda_3\left[\lambda_3^2 s_1^2 + \lambda_4^2\right]^{1/2}}^{A} + \overbrace{c_{rc}\lambda_4^2 s_1}^{B} + \overbrace{\lambda_3 s_1^3 s_4^2\left[\lambda_3^2 s_1^2 + \ldots}^{C} \overbrace{\lambda_4^2\right]^{1/2} + 2\lambda_4 s_1 s_3 s_4\left(\lambda_3^2 s_1^2 + \lambda_4^2\right)^{1/2}}^{D}\right)}{s_1^3\left(\lambda_3^2 s_1^2 + \lambda_4^2\right)^{1/2}} \\ 0 \\ \dfrac{2\lambda_4 s_4}{s_1} - \lambda_1 \\ \dfrac{2\lambda_4 s_3}{s_1} - \lambda_2 - 2\lambda_3 s_1 s_4 \end{bmatrix}$$

THESE TWO METHODS AGREE. GOOD.

NEXT, CONSTRAINTS:

$$G = K + \underbrace{\omega^T \Omega}_{\text{FINAL CONSTRAINTS}} + \underbrace{\theta^T \mathcal{H}}_{\text{INITIAL CONSTRAINTS}}$$

$$\bar{\Omega} = \begin{bmatrix} s_3 f \\ s_4 f^2 s_1 f^3 - 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$\dot{\theta}_f$ ↙ ↘ r_f

RECALL
KEPLERIAN ELEMENTS:

FOR CIRCULAR, $ecc = 0$, $a = r = s_{1_0}$

$\theta = TA$; $\theta_0 = 0°$

$$r = \frac{a(1-ecc^2)}{1-ecc(\cos\theta)} = a \quad \downarrow s_{1_0}$$

$$\dot{r} = \frac{ecc\sin\theta}{a^{1/2}(1-ecc^2)^{1/2}} = 0 \quad \downarrow s_{3_f}$$

$$\dot{\theta} = \frac{(1+ecc\cos\theta)^2}{(a[1-ecc^2])^{3/2}} = a^{-3/2} \quad \rightarrow s_{4_f}^2 s_{1_f}^3$$

@ $t = 0$

$$\therefore \quad \textcircled{H} = \begin{bmatrix} s_{1_0}(1-ecc\cos(s_{2_0})) - s_{1_0} \\ s_{3_0}(s_{1_0}^{1/2}) \\ s_{4_0}(s_{1_0}^{3/2}) - 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

FROM

BOLZA FUNCTION

MAXIMIZE THIS K.

$$G = -s_{1_f} + \cancel{\theta_1 \textcircled{H}_1^{\,0}} + \cancel{\theta_2 \textcircled{H}_2^{\,0}} + \cancel{\theta_3 \textcircled{H}_3^{\,0}} + \dots$$

$$\omega_1(s_{3_f}) + \omega_2(s_{1_f}^3 s_{4_f}^2 - 1)$$

$$G = -s_{1_f} + \omega_1 s_{3_f} + \omega_2(s_{1_f}^3 s_{4_f}^2 - 1) \leftarrow$$

FIND INITIAL AND FINAL LAGRANGIANS.

$$\bar{\lambda}_0 = -G_{s_0}{}^T = -\left(\frac{\delta G}{\delta s_0}\right)^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leftarrow \bar{\lambda}_0$$

$$\bar{\lambda}_f = G_{Sf}^T = \left(\frac{\delta G}{\delta S_f}\right)^T = \begin{bmatrix} -1 + \omega_2 s_{qf}^2 \cdot 3 s_{1f}^2 \\ 0 \\ \omega_1 \\ \omega_2 s_{1f}^3 \cdot 2 s_{qf} \end{bmatrix} \leftarrow \bar{\lambda}_f$$

SOLVE COSTATES.

i) $\dot{\lambda}_2 = 0$, $\therefore \lambda_2 = \text{CONSTANT.}$

ii) $\lambda_{2f} = 0$ THUS $\lambda_{2f} = \lambda_{20} = \lambda_2 = 0$

iii) $\lambda_{3f} = \omega_1$

COSTATE EQNS →

$$\therefore \begin{cases} \lambda_{1f} = -1 + \omega_2 s_{qf}^2 \cdot 3 s_{1f}^2 \\ \lambda_{2f} = 0 \\ \lambda_{3f} = \omega_1 \\ \lambda_{4f} = \omega_2 s_{1f}^3 \cdot 2 s_{qf} \end{cases}$$

$$\omega_1 = \lambda_{3f} \leftarrow$$

$$\omega_2 = \frac{\lambda_{4f}}{2 s_{1f}^3 s_{qf}} \leftarrow$$

$$\lambda_{1f} = -1 + \omega_2 s_{qf}^2 \cdot 3 s_{1f}^2$$

$$= -1 + \frac{\lambda_{4f}}{2 s_{1f}^{3'} s_{qf}} \cdot s_{qf}^{2'} \cdot 3 s_{1f}^2$$

$$\lambda_{1f} = -1 + \frac{3 \lambda_{4f} s_{qf}}{2 s_{1f}}$$

SET $= 0$

$$\therefore \quad -1 + \frac{3 \lambda_{4f} s_{qf}}{2 s_{1f}} - \lambda_{1f} = 0 \leftarrow$$

MATLAB TIME.

INITIAL GUESSES:

$$\bar{S}_0^T = \begin{bmatrix} S_{10} & S_{20} & S_{30} & S_{40} \end{bmatrix};$$

$$= \begin{bmatrix} 1.05 & 0 & 0 & 1.05\text{^}(3/2) \end{bmatrix};$$

$$\bar{\lambda}_0^T = \begin{bmatrix} -1 & 0 & -1 & -1 \end{bmatrix};$$

(0 ← FROM $\dot{\lambda}$; −1, −1 ← DEFINED IN $\dot{\lambda}$)

→ PROPAGATE FROM $[0, t_f]$ TO DETERMINE BEST INITIAL GUESS.

CHECK FOR MISCLOSURES (INSIDE fsolve FUNCTION)

KNOWNS →

$F(1,1) = \Omega_1$

$F(2,1) = \Omega_2$

$F(3,1) = \lambda_{f2}$

$F(4,1) = \lambda_{f1}$

INITIAL CONDITIONS →

$F(5,1) = r_0 = S_{01} = 1.05 \Rightarrow S_{01} - 1.05 = 0$ ←

$F(6,1) = \dot{r}_0 = S_{02}$ ←

$F(7,1) = \theta_0 = S_{03}$ ←

$F(8,1) = \dot{\theta}_0 = S_{04} = 1.05^{3/2}$

$$\frac{S_{04}}{1.05^{3/2}} = 1$$

$$\frac{S_{04}}{1.05^{3/2}} - 1 = 0$$ ←

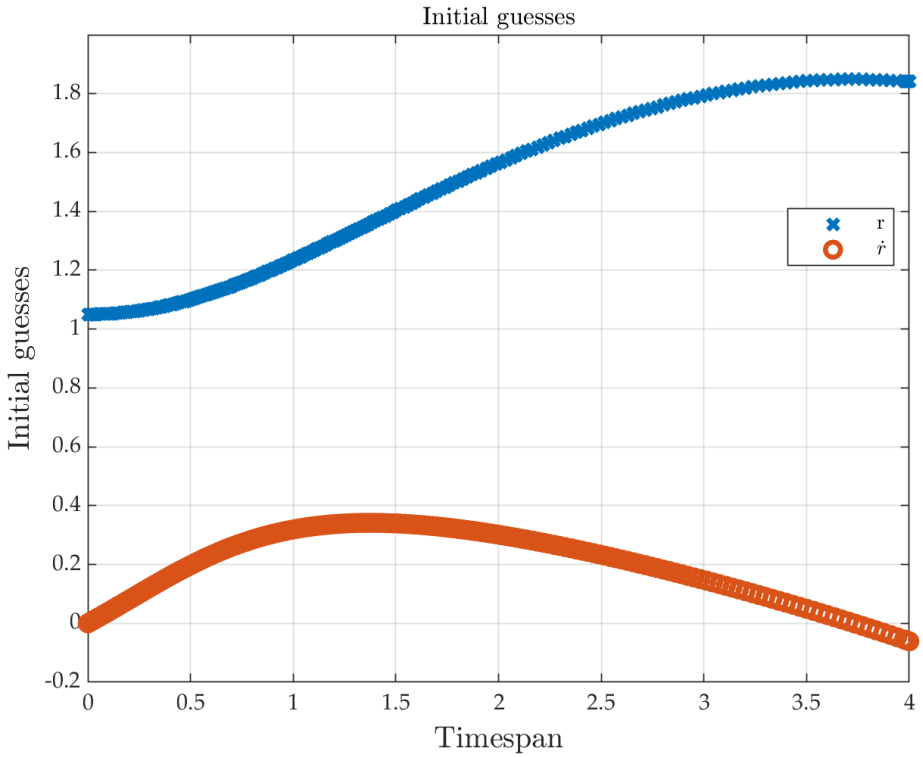


Figure 1: Initial guesses propagated for the full timespan before solving the equations for state. The orbital radius generally increases and $\dot{r}$ goes near zero, as intended. We may proceed to the `fsolve` solver scheme.

Iteration	Func-count	||f(x)||^2	Norm of step	First-order optimality	Trust-region radius
0	9	1.61515		11.7	1
1	18	1.17327	1	2.62	1
2	19	1.17327	1	2.62	1
3	28	0.123891	0.25	0.849	0.25
4	37	0.0413369	0.625	0.0591	0.625
5	46	0.025861	1.5625	1.17	1.56
6	55	0.00348933	1.5625	1.02	1.56
7	64	5.195e-05	0.522311	0.0998	1.56
8	73	2.50501e-08	0.0137167	0.00402	1.56
9	82	1.47327e-15	0.000125089	5.39e-07	1.56
10	91	2.47485e-28	5.24408e-08	4.8e-13	1.56

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>

Figure 2: Fsolve outputs.

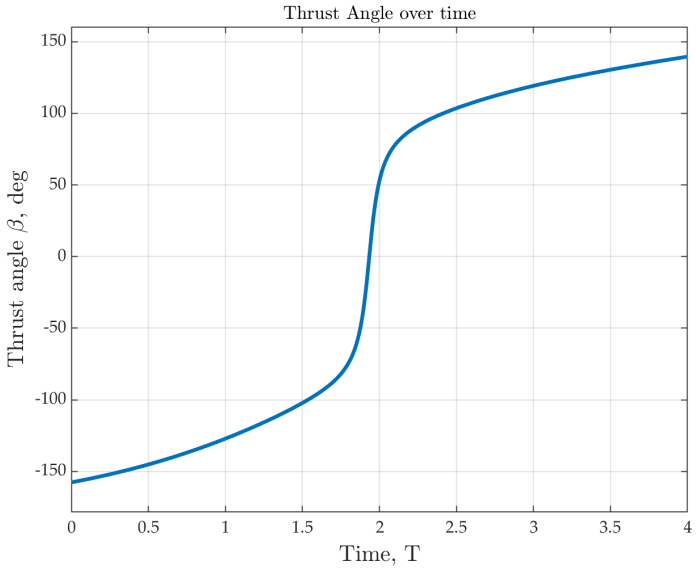


Figure 3: Thrust angle over the timespan of the trajectory.

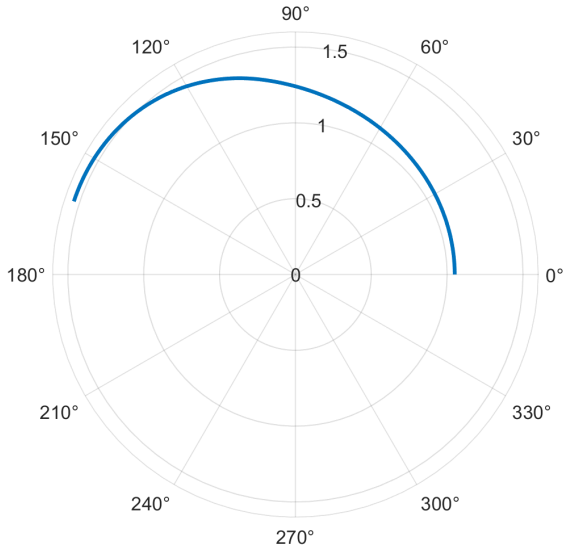


Figure 4: Polar plot showing the trajectory. The mission begins at r = 1.05 DU and $\theta = 0°$

Initial and Final States

After running the `fsolve` scheme, the initial state becomes:

$$s_0 = \begin{bmatrix} 1.0500 \\ 0 \\ -0.0000 \\ 0.9294 \\ -4.1867 \\ 0.0000 \\ -0.9804 \\ -2.4997 \end{bmatrix}$$

Propagating s_0 in $t = [0, 4]$ gives the final state, s_f :

$$s_f = \begin{bmatrix} 1.5376 \\ 2.8230 \\ -0.0000 \\ 0.5245 \\ -2.5424 \\ 0.0000 \\ 1.6689 \\ -3.0143 \end{bmatrix}$$

Final maximized orbital radius is:

$$r_f = 1.5376 \text{ DU}$$

Circularity check of final orbit ($\dot{r}$ should be near zero):

$$\dot{r}_f = -9.4815e - 15$$

This indicates the solver closed and optimized the trajectory according to the user-specified constraints with satisfactory results.

Problem 2

Optimal Orbit with Minimum Fuel

Given the equations of motion for two-body force field, determine the orbit that accomplishes a transfer from an initially circular orbit of 1.05 and a final circular orbit of 2.0 using minimum fuel (assuming the transfer is a burn-coast-burn). The maximum thrust is 0.1 and the exhaust velocity is 0.9. The arrival point on the final circular orbit is unconstrained. EOMs are:

$$\ddot{x} = -\frac{x}{r^3} + \frac{T}{m}u_x$$

$$\ddot{y} = -\frac{y}{r^3} + \frac{T}{m}u_y$$

$$\dot{m} = -\frac{T}{v_e}$$

Provide:

- The problem formulation including the state and co-state, boundary conditions, and the control law.
- The plot of the mass history
- A plot of the transfer trajectory
- The initial and final values of the state and co-state variables.
- The switching times
- The value of the performance index

Solution.

See Appendix B for results and code for Problem 2

See following pages for the problem formulation completed by hand. Following the work by hand are the final problem outputs and `MATLAB` plots.

Problem 2: Optimal Orbit with Minimum Fuel (-m_f)

Given the equations of motion for two-body force field, determine the orbit that accomplishes a transfer from an initially circular orbit of 1.05 and a final circular orbit of 2.0 using minimum fuel (assuming the transfer is a burn-coast-burn). The maximum thrust is 0.1 and the exhaust velocity is 0.9. The arrival point on the final circular orbit is unconstrained. EOMs are:

$$\ddot{x} = -\frac{x}{r^3} + \frac{T}{m}u_x$$

$$\ddot{y} = -\frac{y}{r^3} + \frac{T}{m}u_y$$

$$\dot{m} = -\frac{T}{v_e}$$

Provide:
-The problem formulation including the state and co-state, boundary conditions, and the control law.
-The plot of the mass history
-A plot of the transfer trajectory
-The initial and final values of the state and co-state variables.
-The switching times
-The value of the performance index

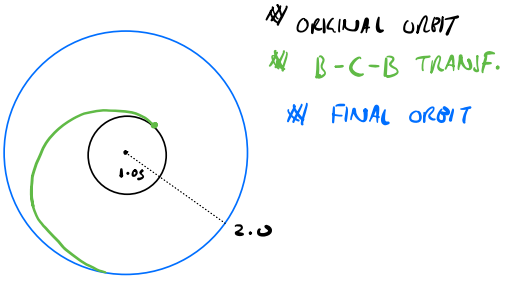
$r_0 = 1.05$ DU
$\theta_0 = 0°$
$\dot{r}_0 = 0$ (CIRCULAR)
$\dot{\theta}_0$ = FREE

$r_f = 2.0$ DU
θ_f = FREE
$\dot{r}_f = 0$ (CIRCULAR)
$\dot{\theta}_f$ = FREE

TRANSFER TYPE = BURN - COAST - BURN.

$T_{MAX} = 0.1$
$v_e = 0.9$

SKETCH



- ORIGINAL ORBIT
- B-C-B TRANSF.
- FINAL ORBIT

EOMS:

$$\ddot{x} = -\frac{x}{r^3} + \frac{T}{m}u_x$$

$$\ddot{y} = -\frac{y}{r^3} + \frac{T}{m}u_y$$

$$\dot{m} = -\frac{T}{v_e}$$

PROBLEM FORMULATION

i) PERFORMANCE INDEX: $K = J = -m_f$ "MAXIMIZE FINAL FUEL MASS"

ii) SINCE TIME IS FREE, NEED TO FORMULATE DEPENDENCY FUNCTION

FROM LECTURE, DEPENDENCY FUNCTION IS: (PUT INTO F(S,1))

$$-\lambda_1\left(\frac{s_2}{s_1}\right) + \lambda_2 - \lambda_3\left(\frac{s_4}{s_3}\right)\left(\frac{s_3 + \frac{s_2}{s_1}s_4}{s_1 + \frac{s_4}{s_3}s_2}\right) + \lambda_4\left(\frac{s_3 + \frac{s_2}{s_1}s_4}{s_1 + \frac{s_4}{s_3}s_2}\right) = \phi$$

ALSO, FROM LECTURE DERIVATION: $\lambda_{5f} = -1$ ←

$$\bar{s} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{bmatrix} = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ m \end{bmatrix} \qquad \therefore \ \dot{\bar{s}} = \bar{f} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \\ \dot{m} \end{bmatrix}$$

IN CANONICAL UNITS.

$$m_{S/C,0} = 1 \Rightarrow$$

CONTROL = $c = u_x, u_y, T$

$$c = \begin{bmatrix} u_x \\ u_y \\ T \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

i) PERFORMANCE INDEX: $K = -m_f$

ii) ESTABLISH $\bar{f}$

$$\bar{f} = \dot{\bar{s}} = \begin{bmatrix} s_3 \\ s_4 \\ -\frac{x}{r^3} + \frac{T}{m} u_x \\ -\frac{y}{r^3} + \frac{T}{m} u_y \\ -\frac{T}{v_e} \end{bmatrix} = \begin{bmatrix} s_3 \\ s_4 \\ -\frac{x}{r^3} + \frac{T}{m} c_1 \\ -\frac{y}{r^3} + \frac{T}{m} c_2 \\ -\frac{c_3}{v_e} \end{bmatrix}$$

* WHERE

$r \equiv \sqrt{x^2 + y^2}$

$x \equiv s_1$ ←

$y \equiv s_2$ ←

$$\bar{f} = \dot{\bar{s}} = \begin{bmatrix} s_3 \\ s_4 \\ \frac{-s_1}{(s_1^2 + s_2^2)^{3/2}} + \frac{c_3 c_1}{s_5} \\ \frac{-s_2}{(s_1^2 + s_2^2)^{3/2}} + \frac{c_3 c_2}{s_5} \\ \frac{-c_3}{v_e} \end{bmatrix}$$

←

iii) BUILD BOLZA AND HAMILTONIAN

MAXIMIZING

FINAL CONSTRAINTS

$$G = -s_{5_f} + \bar{\theta}^T \cancel{(H)}^{0} + \bar{\omega}^T \bar{\Omega}$$

NONE

↑ DERIVE $\bar{P}$ H.

$$H = \cancel{L}^{0} + \lambda^T \bar{f}$$

$$H = \lambda_1 s_3 + \lambda_2 s_4 + \lambda_3 \left(\frac{-s_1}{r^3} + \frac{c_3}{s_5} c_1 \right) + \cdots$$

$$\lambda_4 \left(\frac{-s_2}{r^3} + \frac{c_3}{s_5} c_2 \right) + \lambda_5 \left(\frac{-c_3}{v_e} \right)$$

iv.) N.B.C AND CONTROL LAW.

- FIND SWITCHING FUNCTION, Φ.

* NOTICE H IS MINIMIZED FOR:

$c_3 = 0$ (ALL c_3 TERMS)

OR $c_3 = T_{max}$ (LAST TERM)

$$\frac{\delta H}{\delta c} = 0$$

$$\therefore \frac{\partial H}{\delta c_1} = \frac{\lambda_3 c_3}{s_5}$$

$$\frac{\delta H}{\delta c_2} = \frac{\lambda_4 c_3}{s_5}$$

$$\frac{\partial H}{\delta c_3} = \frac{\lambda_3 c_1}{s_5} + \frac{\lambda_4 c_2}{s_5} - \frac{\lambda_5}{v_e}$$

* FROM LECTURE: ↓ ↓ SKIP MATH.

$$c_1 = -\frac{\lambda_3}{\lambda_v} \quad \text{AND} \quad c_2 = -\frac{\lambda_4}{\lambda_v} \longleftarrow$$

WHERE $\lambda_v = \sqrt{\lambda_3^2 + \lambda_4^2}$ ←

RECALL:

$$H = \lambda_1 s_3 + \lambda_2 s_4 + \lambda_3 \left(\frac{-s_1}{r^3} + \frac{c_3}{s_5} c_1 \right) + \lambda_4 \left(-\frac{s_2}{r^3} + \frac{c_3}{s_5} c_2 \right) + \lambda_5 \left(-\frac{c_3}{v_e} \right)$$

REARRANGE

$$= \lambda_1 s_3 + \lambda_2 s_4 - \frac{\lambda_3 s_1}{r^3} + \lambda_3 \frac{c_3 c_1}{s_5} - \lambda_4 \frac{s_2}{r^3} + \lambda_4 \frac{c_3 c_2}{s_5} - \frac{\lambda_5 c_3}{v_e}$$

$$= \lambda_1 s_3 + \lambda_2 s_4 - \frac{\lambda_3 s_1}{r^3} - \frac{\lambda_4 s_2}{r^3} + \frac{c_3}{s_5 v_e} \left(\lambda_3 c_1 v_e + \lambda_4 c_2 v_e - \lambda_5 s_5 \right)$$

$$= \lambda_1 s_3 + \lambda_2 s_4 - \frac{\lambda_3 s_1}{r^3} - \frac{\lambda_4 s_2}{r^3} + \frac{c_3}{s_5 v_e} \left[\left(\lambda_3 c_1 + \lambda_4 c_2 \right) v_e - \lambda_5 s_5 \right]$$

INSERT c_1 & c_2

$$= \lambda_1 s_3 + \lambda_2 s_4 - \frac{\lambda_3 s_1}{r^3} - \frac{\lambda_4 s_2}{r^3} + \frac{c_3}{s_5 v_e} \left[\left(\frac{\lambda_3 \cdot (-\lambda_3)}{\lambda_v} + \frac{\lambda_4 (-\lambda_4)}{\lambda_v} \right) v_e - \lambda_5 s_5 \right]$$

$$H = \lambda_1 s_3 + \lambda_2 s_4 - \frac{\lambda_3 s_1}{r^3} - \frac{\lambda_4 s_2}{r^3} - \frac{c_3}{s_5 v_e} \left[v_e \left(\frac{\lambda_3^2 + \lambda_4^2}{\lambda_v} \right) + \lambda_5 s_5 \right]$$

(λ_v^2 over $\lambda_3^2 + \lambda_4^2$)

$$H = \lambda_1 s_3 + \lambda_2 s_4 - \frac{\lambda_3 s_1}{r^3} - \frac{\lambda_4 s_2}{r^3} - \frac{c_3}{s_5 v_e} \left[\lambda_v v_e + \lambda_5 s_5 \right]$$

* LOOKING ONLY AT c_3 TERMS

* MINIMIZE H FOR $c_3 \neq$ TRIVIAL SOLN.

$$0 = \frac{-c_3}{s_5 v_e} \left[\lambda_v v_e + \lambda_5 s_5 \right]$$

$c_3 = 0$, TMAX.

$\lambda_v v_e + \lambda_5 s_5 = 0$ ← CONSTRAINT FOR fsolve.

THUS,

$$\begin{aligned} \mathcal{H}(t_0) &= \lambda_{v_0} v_e + \lambda_{s_0} s_{5_0} = 0 \\ \mathcal{H}(t_1) &= \lambda_{v_1} v_e + \lambda_{s_1} s_{5_1} = 0 \\ \mathcal{H}(t_2) &= \lambda_{v_2} v_e + \lambda_{s_2} s_{5_2} = 0 \\ \mathcal{H}(t_3) &= \lambda_{v_3} v_e + \lambda_{s_3} s_{5_3} = 0 \end{aligned}$$

NOW WE HAVE THIS FOR LATER.

BACK TO THE PROBLEM FORMULATION.

- NEED $\dot{\lambda}$.

* $r \equiv \sqrt{s_1^2 + s_2^2}$

$$\dot{\lambda} = -\frac{\delta H}{\delta s}$$

RECALL:

$$H = \lambda_1 s_3 + \lambda_2 s_4 - \frac{\lambda_3 s_1}{r^3} - \frac{\lambda_4 s_2}{r^3} - \frac{c_3}{s_7 v_e}\left[\lambda_v v_e + \lambda_7 s_5\right]$$

$$H = \lambda_1 s_3 + \lambda_2 s_4 - \frac{\lambda_3 s_1}{(s_1^2+s_2^2)^{3/2}} - \frac{\lambda_4 s_2}{(s_1^2+s_2^2)^{3/2}} - \frac{c_3}{s_7 v_e}\left[\lambda_v v_e + \lambda_7 s_5\right]$$

$-\frac{\delta H}{\delta s}$ COMPUTED BY MATLAB:

$$\therefore \quad \dot{\lambda} = \begin{bmatrix} -\frac{1}{(s_1^2+s_2^2)^{5/2}}\left[2\lambda_3 s_1^2 - \lambda_3 s_2^2 + 3\lambda_4 s_1 s_2\right] \\ -\frac{1}{(s_1^2+s_2^2)^{5/2}}\left[-\lambda_4 s_1^2 + 2\lambda_4 s_2^2 + 3\lambda_3 s_1 s_2\right] \\ -\lambda_1 \\ -\lambda_2 \\ -c_3 \\ \hline s_5^2\left(\lambda_3^2+\lambda_4^2\right)^{1/2} \end{bmatrix}$$

COSTATES ↗

FINAL CONSTRAINTS

i.) $r_f = 2$ (FINAL ORBITAL RADIUS = 2 DU)

RECALL FROM LECTURE:

$$s_{1f}^2 + s_{2f}^2 - r_f^2 = 0$$

$$\therefore \quad (s_{1f}^2 + s_{2f}^2)^{1/2} - 2 = 0 \leftarrow$$

ii.) CIRCULAR FINAL ORBIT.

$$s_{3f}^2 + s_{4f}^2 - v_f^2 = 0$$

$\downarrow$ $\dot{x}$ $\quad$ $\downarrow$ $\dot{y}$ $\quad$ $\downarrow$ CIRCULAR VELOCITY $= \sqrt{\frac{\mu}{r}} = \sqrt{\frac{1}{r_f}} = \sqrt{\frac{1}{2}}$

$$\therefore \quad s_{3f}^2 + s_{4f}^2 - \frac{1}{2} = 0 \leftarrow$$

THIRD CONSTRAINT: (DOT PRODUCT)

$$s_{4f} s_{1f} - s_{3f} s_{2f} - r_f v_f = 0$$

$$s_{4f} s_{1f} - s_{3f} s_{2f} - 2\sqrt{\tfrac{1}{2}} = 0$$

$$s_{4f} s_{1f} - s_{3f} s_{2f} - \sqrt{2} = 0 \leftarrow$$

FINAL CONSTRAINTS ARE:

$$\bar{\Omega} = \begin{bmatrix} (s_{1f}^2 + s_{2f}^2)^{1/2} - 2 \\ s_{3f}^2 + s_{4f}^2 - \frac{1}{2} \\ s_{4f} s_{1f} - s_{3f} s_{2f} - \sqrt{2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \leftarrow$$

$\Downarrow$ PUT INTO $F(1,1) \rightarrow F(1,3)$

REITERATE CONTROL LAW:

$$\bar{c} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} -\lambda_3 \lambda_7^{-1} \\ -\lambda_4 \lambda_7^{-1} \\ T \end{bmatrix}$$

FIND INITAL AND FINAL LAGRANGE MULTIPLIERS.

RECALL: $G = -s_{5_f} + \bar{\theta}^T \bar{H} + \bar{\omega}^T \bar{\Omega}$

SINCE FULLY DEFINED.

NONE

$$G = -s_{5f} + \bar{\omega}^T \bar{\Omega} \leftarrow$$

AND $\bar{\lambda}_f = G_{sf}{}^T = \left(\frac{\delta G}{\delta s_f}\right)^T$

$$\bar{\lambda}_f = \frac{\delta}{\delta s_f}\left\{ -s_{5f} + \left[(s_{1f}{}^2 + s_{2f}{}^2)^{1/2} - 2\right]\omega_1 + \cdots \right.$$
$$\left. \left[s_{3f}{}^2 + s_{4f}{}^2 - \tfrac{1}{2}\right]\omega_2 + \left[s_{4f}\, s_{1f} - s_{3f}\, s_{2f} - N\sqrt{2}\right]\omega_3 \right\}$$

$$\therefore \bar{\lambda}_f = \begin{bmatrix} \frac{1}{2}\left(s_{1f}{}^2 + s_{2f}{}^2\right)^{-1/2} \cdot 2 s_{1f}\, \omega_1 + s_{4f}\, \omega_3 \\ \frac{1}{2}\left(s_{1f}{}^2 + s_{2f}{}^2\right)^{-1/2} \cdot 2 s_{2f}\, \omega_1 - s_{3f}\, \omega_3 \\ 2 s_{3f}\, \omega_2 - s_{2f}\, \omega_3 \\ 2 s_{4f}\, \omega_2 + s_{1f}\, \omega_3 \\ -1 \end{bmatrix} \leftarrow$$

* $\lambda_{5f} = -1$

$\lambda_{5f} + 1 = 0$

INTO $F(q,1)$

SINCE NOT OPTIMIZING TIME, NEED ANOTHER NATURAL B.C.

$$H(t_f) = 0$$

↳ PUT INTO F(6,1)

$c_3 = T = 0$ @ t_f.

$$H(t_f) = \lambda_{1_f} s_{3_f} + \lambda_{2_f} s_{4_f} - \frac{\lambda_{3_f} s_{1_f}}{\left(s_{1_f}^2 + s_{2_f}^2\right)^{3/2}} - \frac{\lambda_{4_f} s_{2_f}}{\left(s_{1_f}^2 + s_{2_f}^2\right)^{3/2}} - \frac{c_3}{s_{5_f} v_e}\left[\lambda_V v_e + \lambda_{5_f} s_{5_f}\right] = 0$$

FINALLY, INITIAL CONDITION EQNS.

$$F(7,1) = s_{0_1} - 1.05 \; ;$$
$$F(8,1) = s_{0_2} \; ;$$
$$F(9,1) = s_{0_3} \; ;$$
$$F(10,1) = s_{0_4} - \sqrt{\frac{1}{1.05}} \; ; \qquad \dot{r}_0 = \text{velocity} = s_{0_4} = \sqrt{\frac{\mu}{r_0}}$$
$$F(11,1) = s_{0_5} - 1 ; \qquad \text{MASS}|_{t_0} = 1.0$$

MATLAB TIME!

THE REST OF THIS PROBLEM WAS COMPLETED USING MATLAB

END OF WORK BY HAND.

Thrust Always On Configuration

```
%..... Initial guesses for THRUST ALWAYS ON config.
state = [1.05;0;0;sqrt(1/1.05);1]; % Five state vars: x, y, xdot,
    ydot, mass
lambda = [-1;-1;-1;-1;-1];  % Five lambda guesses (totally unknown
     except for lam5 = -1)
```

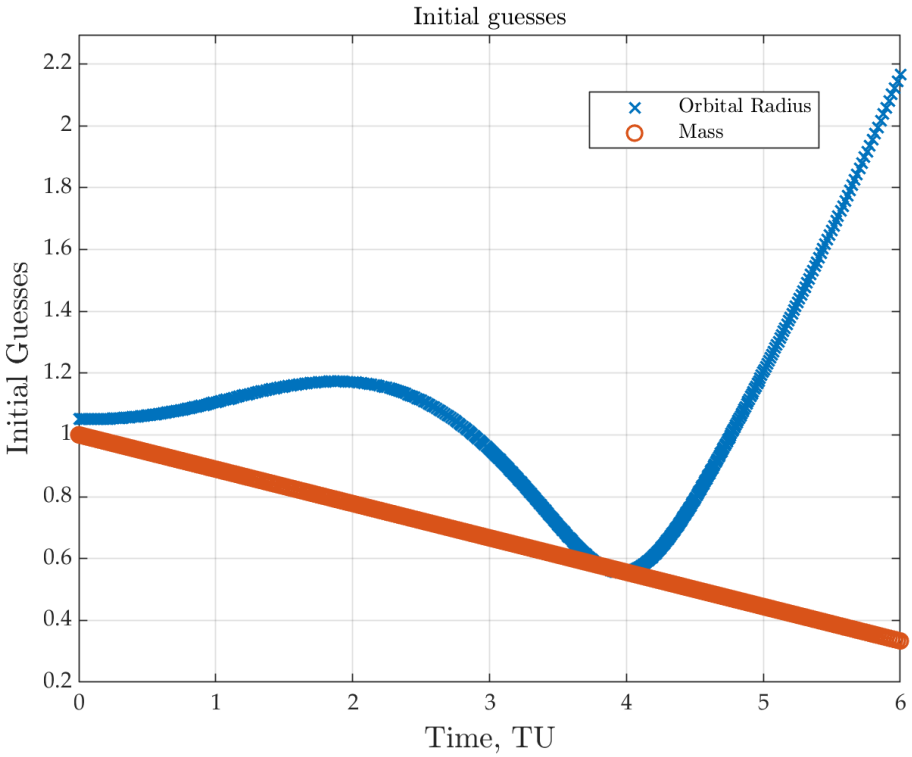


Figure 5: Initial guesses propagated using `ode45` for visualization. In this case, orbital radius is expected to increase from 1.05 to 2.0 DU, while mass is expected to decrease. These guesses confirm the general trend, so we continue to the solving scheme.

The time guesses going into the BCB solver part of the code are (from Fig. 6): 2.6, 4-2.6, and 4*0.2 (TU). These provide the initial guesses for elapsed time between the places where the final switching function should cross the x-axis.

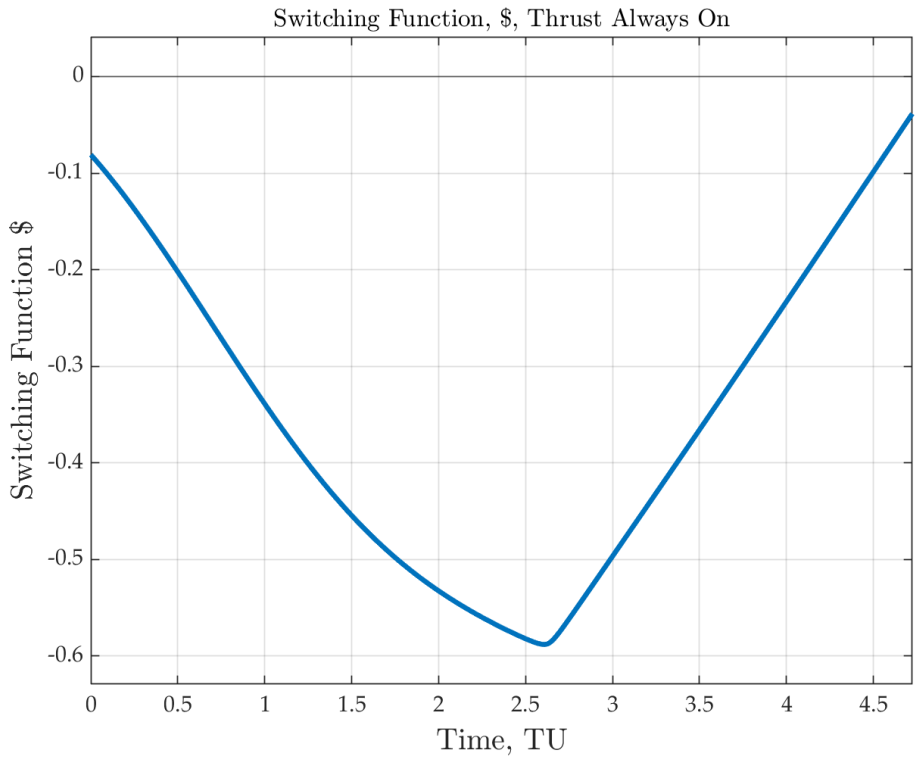


Figure 6: Initial switching function plot after the `fsolve` scheme converged for the thrust-on-full-time configuration. This plot provided good input guesses for the burn-coast-burn trajectory switching function times.

Burn-Coast-Burn Trajectory

```
%..... Initial guesses (for Burn-Coast_Burn traj)
state = [1.05;0;0;sqrt(1/1.05);1]; % Five state vars: x, y, xdot,
    ydot, mass
lambda = [-1;0;0;-1;-1]; % these work well (Thanks Dr A!)
tguess = [2.6; 4-2.6;4*0.2]; % peak of $ plot; end; end*0.2
```

Initial State and Co-State Variables

$$\bar{s_0} = \begin{bmatrix} 1.0500 \\ -0.0000 \\ -0.0000 \\ 0.9759 \\ 1.0000 \end{bmatrix} \qquad \bar{\lambda_0} = \begin{bmatrix} -0.7500 \\ 0.0143 \\ 0.0154 \\ -0.8287 \\ -0.7460 \end{bmatrix}$$

Final State and Co-State Variables

$$\bar{s}_f = \begin{bmatrix} -1.3972 \\ -1.4310 \\ 0.5059 \\ -0.4940 \\ 0.7469 \end{bmatrix} \qquad \bar{\lambda}_f = \begin{bmatrix} 0.1964 \\ 0.2036 \\ -0.5904 \\ 0.5832 \\ -1.0000 \end{bmatrix}$$

Switching Times

Initial guesses for switching function:

```
% Best time guesses from (thrust on) $ plot
t1guess = 2.6;          % peak of plot
t2guess = 4 - t1guess;  % time between t1, t2
t3guess = 4*0.2;        % 20% more than final guess
```

The final switching times are (in TU):

$$t_0 = 0$$

$$t_1 = 1.3425$$

$$t_2 = 6.1699$$

$$t_3 = 7.1054$$

Fig. 7 shows the final switching function with switching times highlighted.

Mass History and Final Performance Indices

As expected, the final mass for the fuel-optimizing trajectory using a burn-coast-burn maneuver consumed less fuel (resulting in a higher final mass value) than the thrust-always-on configuration (see Table 1. Mass history during the trajectory is shown in Fig. 8.

Table 1: Final Mass Values for Both Simulations

Thrust-always-on	Burn-Coast-Burn
0.4751 MU	0.74689 MU

Transfer Trajectory

The final transfer trajectory is presented in Fig. 9 and reflects the optimal trajectory with thrust firing for a Burn-Coast-Burn maneuver that minimizes fuel consumption and satisfies all given constraints.

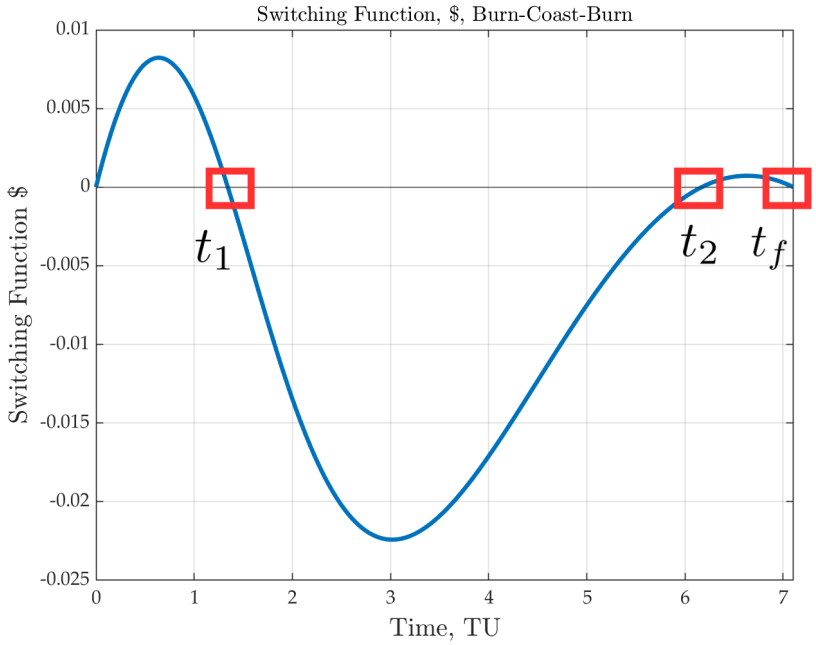


Figure 7: Switching function with time for the BCB maneuver.

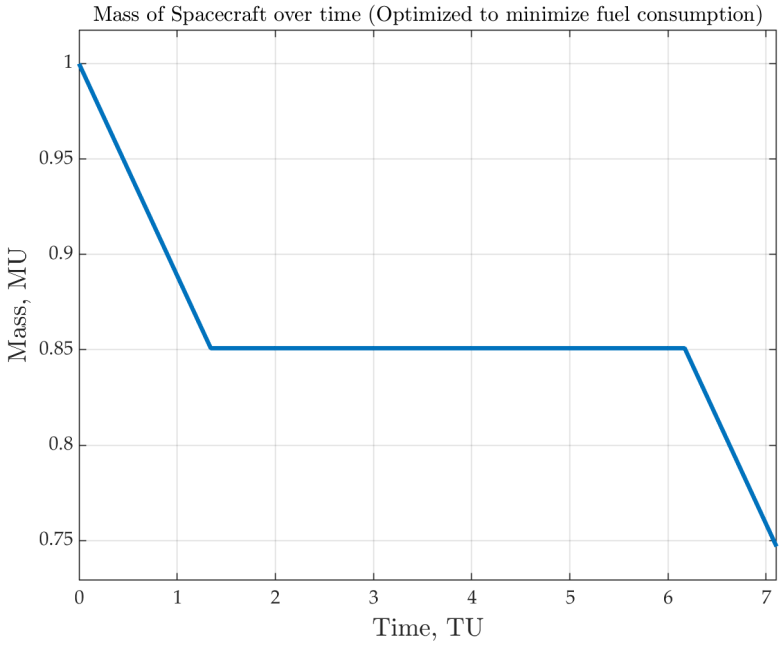


Figure 8: Spacecraft mass over time during the optimized maneuver. Sections in the plot clearly show the burn-coast-burn segments of the maneuver.

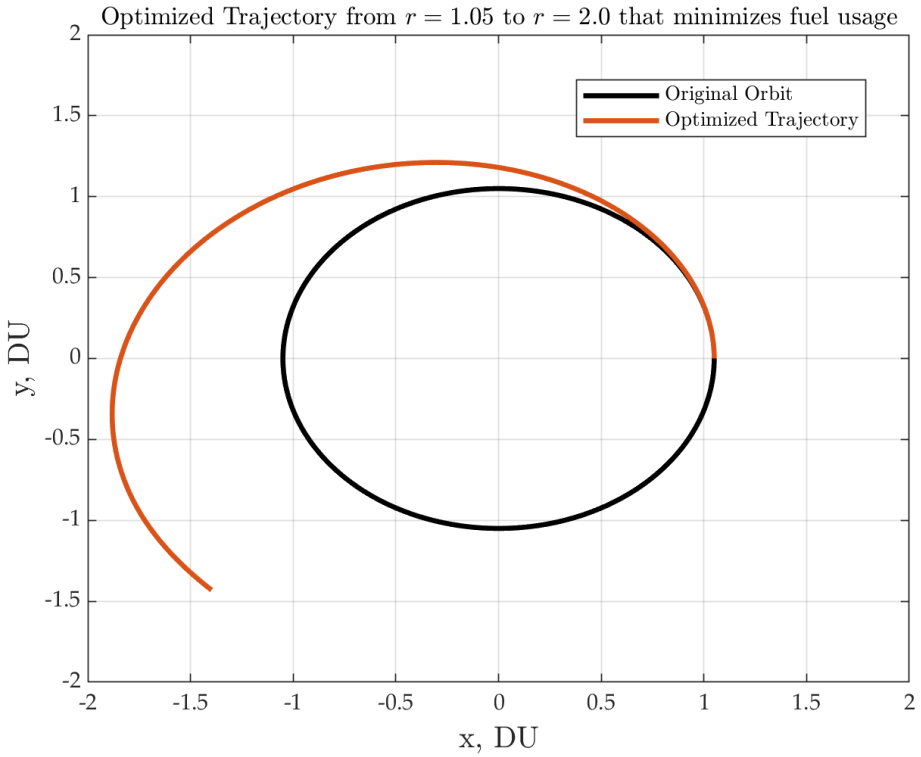


Figure 9: Final trajectory of circular-to-circular transfer trajectory using a Burn-Coast-Burn maneuver optimized to minimize fuel consumption with given constraints.

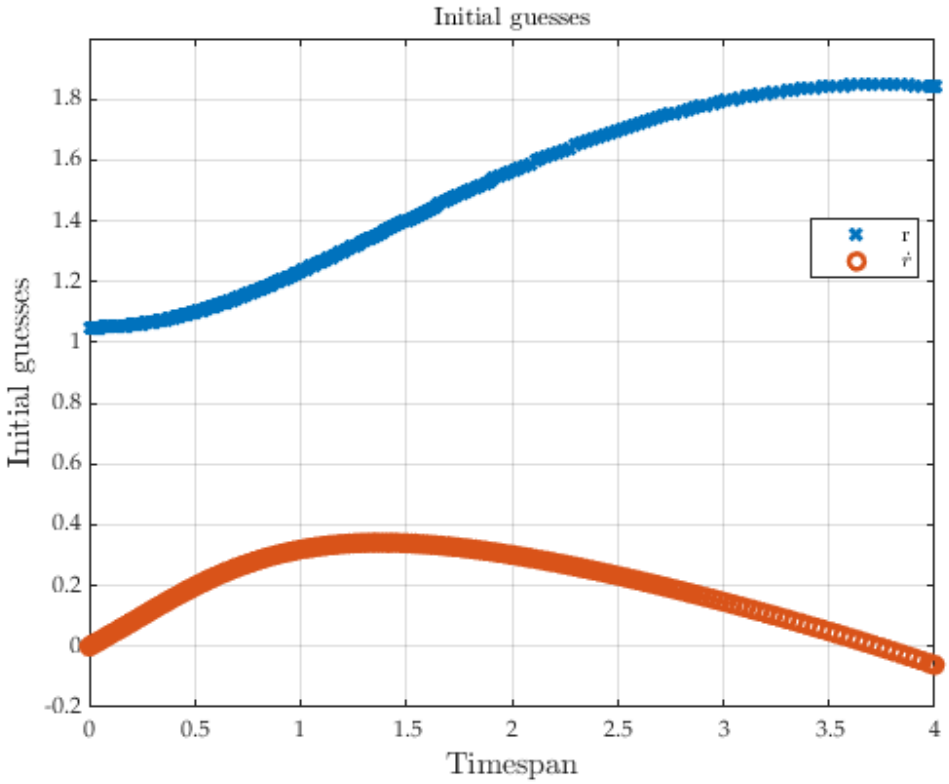
Appendix A: Problem 1 Results and Code

Table of Contents

Propagate initial guesses forward to obtain better estimates

Check guesses

HEART CHECK: r should be getting bigger, rdot should be zeroish. GOOD. Proceed to fsolve scheme.



Run fsolve

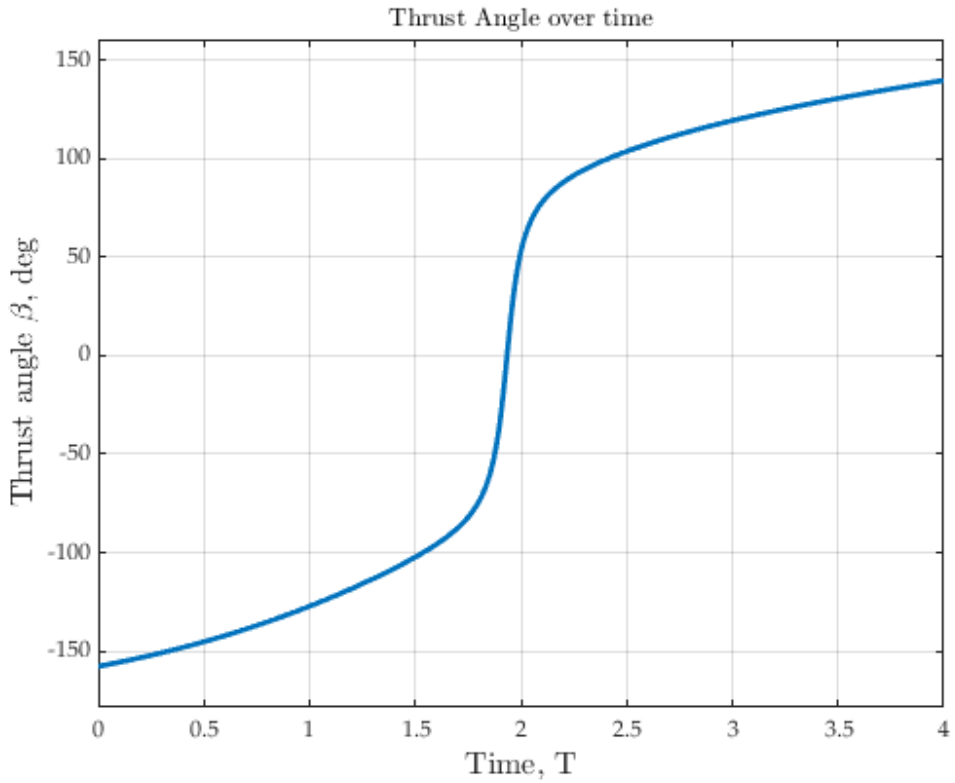
```
                                                Norm of      First-order
Trust-region
 Iteration  Func-count     ||f(x)||^2            step
optimality         radius
     0          9              1.61515
11.7               1
     1         18              1.17327               1
2.62               1
     2         19              1.17327               1
2.62               1
     3         28             0.123891            0.25
0.849            0.25
     4         37            0.0413369           0.625
0.0591           0.625
     5         46             0.025861          1.5625
1.17             1.56
     6         55           0.00348933          1.5625
1.02             1.56
     7         64            5.195e-05        0.522311
0.0998             1.56
     8         73          2.50501e-08       0.0137167
0.00402            1.56
     9         82          1.47327e-15     0.000125089
5.39e-07             1.56
    10         91          2.47485e-28     5.24408e-08
4.8e-13            1.56

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.
```

Results

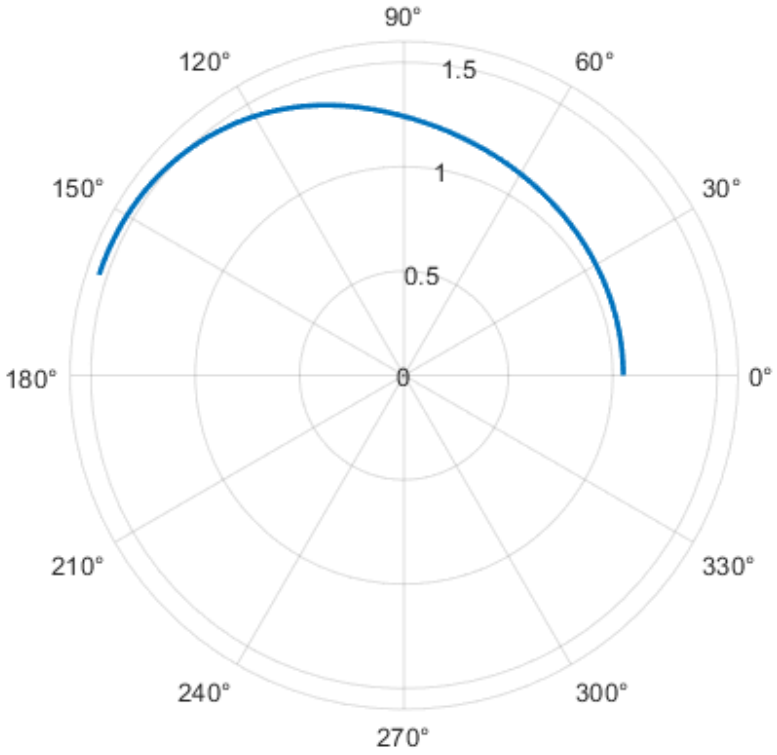
```
S0 is:
    1.0500
         0
   -0.0000
    0.9294
   -4.1867
    0.0000
   -0.9804
   -2.4997

Final state (sf) is:
    1.5376
```

```
    2.8230
   -0.0000
    0.5245

Final lambdas (lf) are:
   -2.5424
    0.0000
    1.6689
   -3.0143

Final orbital radius is: 1.5376 D units <<< -------
Check for circularity of final orbit. Rdot_final (should be near zero) is:
-9.4815e-15
```

Plot thrust angle



Plot trajectory in polar



Published with MATLAB® R2023b

Table of Contents


```
%{
Justin Self
AERO 557 | Advanced Orbital Mechanics
Winter 2024
California Polytechnic State University SLO

HW #4: Orbit Optimization Problems
%}

% Housekeeping
clear all; close all; clc;
addpath("C://MATLAB_CODE/Orbits/")

% Find lagrangian (lambda) dot for HW 4.1.
% Solved by hand up to this derivative (checking with Mathematica output)

% Define syms
syms lam1 lam2 lam3 lam4 s1 s2 s3 s4 acc

% Write equation derived by hand
lamdot =  lam1*s3 + lam2*s4 + lam3*(s1*s4^2 - s1^-2 + acc*(-s1*lam3) /
(sqrt(s1^2*lam3^2 + lam4^2))) + ...
        lam4*(s1^-1*(acc*(-lam4) /  (sqrt(s1^2*lam3^2 + lam4^2)) - 2*s3*s4));

% Have MATLAB differentiate
Dlam1 = simplify(-diff(lamdot,s1));
Dlam1 = matlabFunction(Dlam1); % so I can copy it into the ODE function
Dlam2 = -diff(lamdot,s2);
Dlam3 = -diff(lamdot,s3);
Dlam4 = -diff(lamdot,s4);
% Validated these against MATHEMATICA; good.
```

Propagate initial guesses forward to obtain better estimates

```
%..... Find best initial lambda values
ode_options = odeset("RelTol",1e-8,"AbsTol",1e-8);
tf = 4;
tspan = [0 tf];
```

```
%..... Initial guesses (0 = initial; f = final)
state = [1.05;0;0;1.05^(3/2)];      % knowns [r,theta,rdot,thetadot] @ t0
lambda = [-1;0;-1;-1];              % try these
y0 = [state;lambda];
[timenew,statenew] = ode45(@optimizationODE,tspan,y0,ode_options);
```

Check guesses

```
figure
% Check guesses
plot(timenew,statenew(:,1),'x','LineWidth',2)       % r
hold on
plot(timenew,statenew(:,3),'o','LineWidth',2)       % rdot

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('Timespan','Interpreter','latex');
yLab = ylabel('Initial guesses','Interpreter','latex');
plotTitle = title('Initial guesses','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 12)
grid on
legend('r','$\dot{r}$','interpreter','latex','Location', 'best')

% r should be getting bigger
% rdot should be zero
disp("HEART CHECK: r should be getting bigger, rdot should be zeroish. GOOD.
Proceed to fsolve scheme.")
```

Run fsolve

```
fs_options =
optimset('Display','iter','TolFun',1e-8,'tolx',1e-8,'MaxIter',2e3);
[x,fval,outputs] = fsolve(@optimizationFun,y0,fs_options,tf);
```

Results

```
s0 = x;
disp(" ")
disp("S0 is: ")
disp(s0)

% S0  = known; now propagate forward to find J.
tspan = [0 tf];
[timenew,statenew] = ode45(@optimizationODE,tspan,s0,ode_options);

% Extract vals
```

```
sf1 = statenew(:,1); % x-position
sf2 = statenew(:,2); % y-position
sf3 = statenew(:,3); % x-velocity < -- maximize this
sf4 = statenew(:,4); % y-velocity
lamf1 = statenew(:,5);
lamf2 = statenew(:,6);
lamf3 = statenew(:,7);
lamf4 = statenew(:,8);

% Display final state
disp("Final state (sf) is: ")
disp([sf1(end);sf2(end);sf3(end);sf4(end)])

disp("Final lambdas (lf) are: ")
disp([lamf1(end);lamf2(end);lamf3(end);lamf4(end)])

% COST FUNCTION: J = -rf --> maximize final orbital radius; circular
disp("Final orbital radius is: " + sf1(end) + " D units <<< ------- ")
disp("Check for circularity of final orbit. Rdot_final (should be near zero)
is: " + sf3(end))
```

Plot thrust angle

```
beta = atan2d((sf1.*lamf3),lamf4);

figure
plot(timenew,beta,'LineWidth',2)
% Graph pretty
ylim padded
xlim tight
xLab = xlabel('Time, T','Interpreter','latex');
yLab = ylabel('Thrust angle $\beta$, deg','Interpreter','latex');
plotTitle = title('Thrust Angle over time','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 12)
grid on
```

Plot trajectory in polar

```
figure
theta = sf2;
r = sf1;
polarplot(theta,r,'LineWidth',2)
```

Published with MATLAB® R2023b

Appendix B: Problem 2 Results and Code

Table of Contents

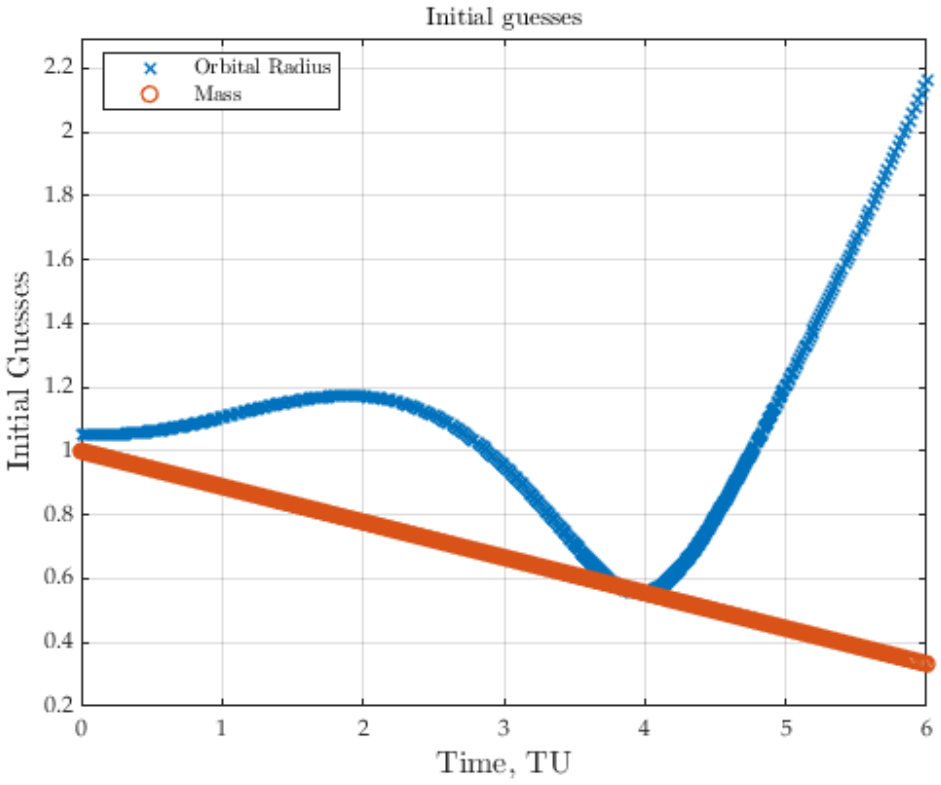
Problem Formulation

Check initial guesses with ODE propagation

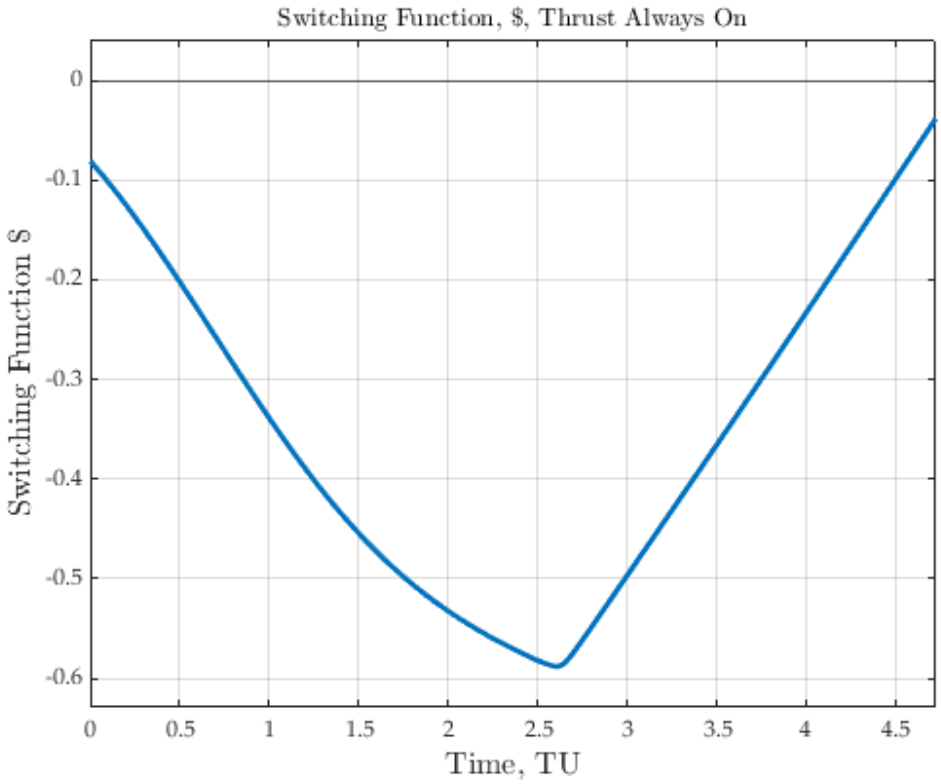
```
HEART CHECK: Initial guesses: orbital radius should increase, [fuel] mass
should decrease. Good.
Warning: Trust-region-dogleg algorithm of FSOLVE cannot handle non-square
systems; using Levenberg-Marquardt algorithm instead.

                                        First-order                    Norm
of
 Iteration  Func-count    ||f(x)||^2      optimality       Lambda
step
     0          12          20.4278             211         0.01
     1          24          3.35702            22.4        0.001
1.00778
     2          37          1.94407            7.76         0.01
1.28712
     3          49         0.192744            1.24        0.001
0.595787
     4          62        0.0852065           0.238         0.01
0.585736
     5          75        0.0714606           0.593          0.1
0.176922
     6          88        0.0616188          0.0548            1
0.0380066
     7         100        0.0560054           0.378          0.1
0.045962
     8         113        0.0518083           0.125            1
0.0375818
     9         125        0.0499339           0.025          0.1
0.029131
    10         137        0.0406157           0.449         0.01
0.203344
    11         150        0.0346403           0.019          0.1
0.16015
```

12	162	0.0303481	0.936	0.01	0.14422
13	175	0.0248257	0.0935	0.1	0.14159
14	188	0.0213087	1.97	1	0.029695
15	200	0.0197674	0.272	0.1	0.0313097
16	212	0.0190062	2.75	0.01	0.168906
17	225	0.0126619	4.43	0.1	0.118553
18	237	0.00687229	1.03	0.01	0.0957315
19	250	0.00515683	0.416	0.1	0.0769667
20	262	0.00416256	0.435	0.01	0.0635271
21	275	0.00344439	0.324	0.1	0.0540144
22	287	0.00292299	0.233	0.01	0.0470941
23	300	0.00252532	0.172	0.1	0.0418963
24	312	0.00220919	0.13	0.01	0.0378028
25	325	0.00195083	0.1	0.1	0.0344372
26	337	0.0017358	0.079	0.01	0.0315775
27	350	0.00155464	0.0628	0.1	0.0290914
28	362	0.00140066	0.0504	0.01	0.0268967
29	375	0.00126889	0.0407	0.1	0.0249409
30	387	0.00115549	0.0329	0.01	0.0231868
31	399	0.00104556	0.52	0.001	0.146496
32	412	0.000458106	0.192	0.01	0.101013
33	424	0.000266874	0.07	0.001	0.074894
34	437	0.000183402	0.0535	0.01	0.0587401
35	449	0.00013268	0.0341	0.001	0.047547
36	462	9.88869e-05	0.0218	0.01	0.0393371
37	474	7.52802e-05	0.0141	0.001	0.0330977
38	486	5.94822e-05	0.215	0.0001	0.131711

```
    39          498       1.32892e-05          0.0771         1e-05
0.109904
    40          510       1.34353e-08         0.00281         1e-06
0.0191455
    41          522       4.73086e-16        4.13e-07         1e-07
0.000235434
    42          534       2.97994e-17        5.01e-09         1e-08
2.58877e-07
    43          546       1.56758e-17        3.18e-09         1e-09
9.35338e-09

Equation solved, solver stalled.

fsolve stopped because the relative size of the current step is less than the
value of the step size tolerance and the vector of function values
is near zero as measured by the value of the function tolerance.

Thrust always on configuration final mass is: 0.4751 mass units
```



Plot switching function, first time.



Burn-Coast-Burn

Best time guesses from $ plot

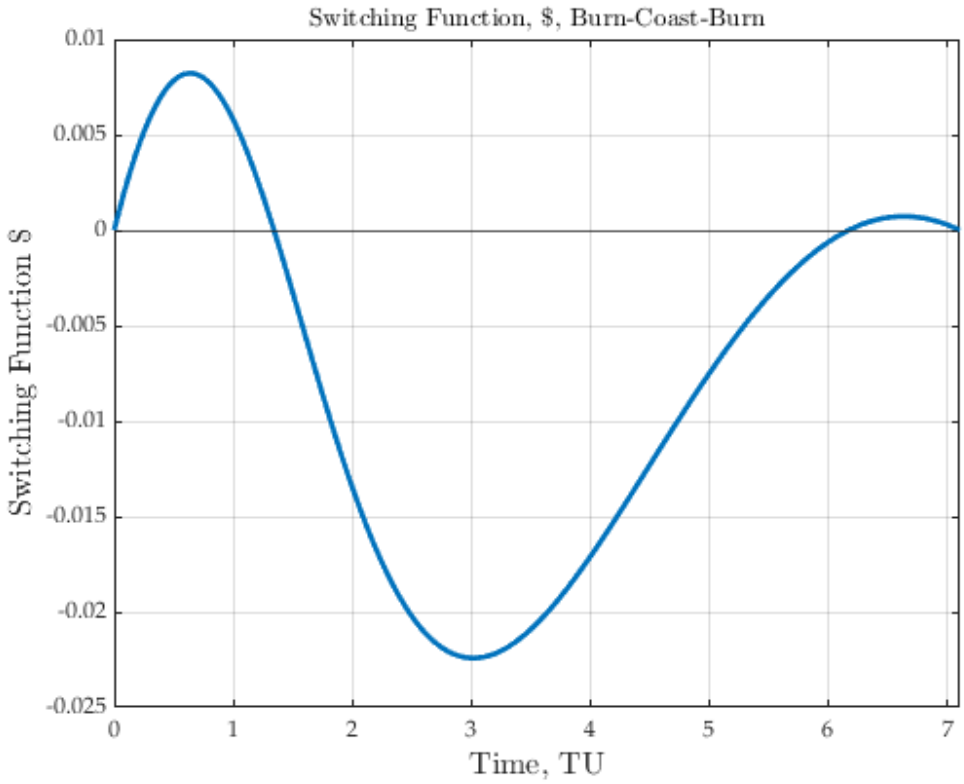
```
Warning: Trust-region-dogleg algorithm of FSOLVE cannot handle non-square
systems; using Levenberg-Marquardt algorithm instead.

                                        First-order                    Norm
of
 Iteration  Func-count    ||f(x)||^2      optimality       Lambda
step
     0          14          0.95805            6.79         0.01
     1          28        0.0290018           0.545        0.001
0.679529
     2          42        0.0187387           0.918       0.0001
1.14036
     3          56       0.00150329             0.3        1e-05
0.889158
     4          70      0.000392205           0.165        1e-06
0.618556
     5          84      2.61854e-05          0.0511        1e-07
0.383449
     6          98      2.70549e-06          0.0171        1e-08
0.22012
```

```
     7         112      2.13414e-07         0.00492        1e-09
0.119894
     8         126      1.09597e-09        0.000358        1e-10
0.0333457
     9         140      3.47352e-16        2.03e-07        1e-11
0.00089281
    10         154      2.88518e-18        3.45e-13        1e-12
2.38877e-07

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.
```

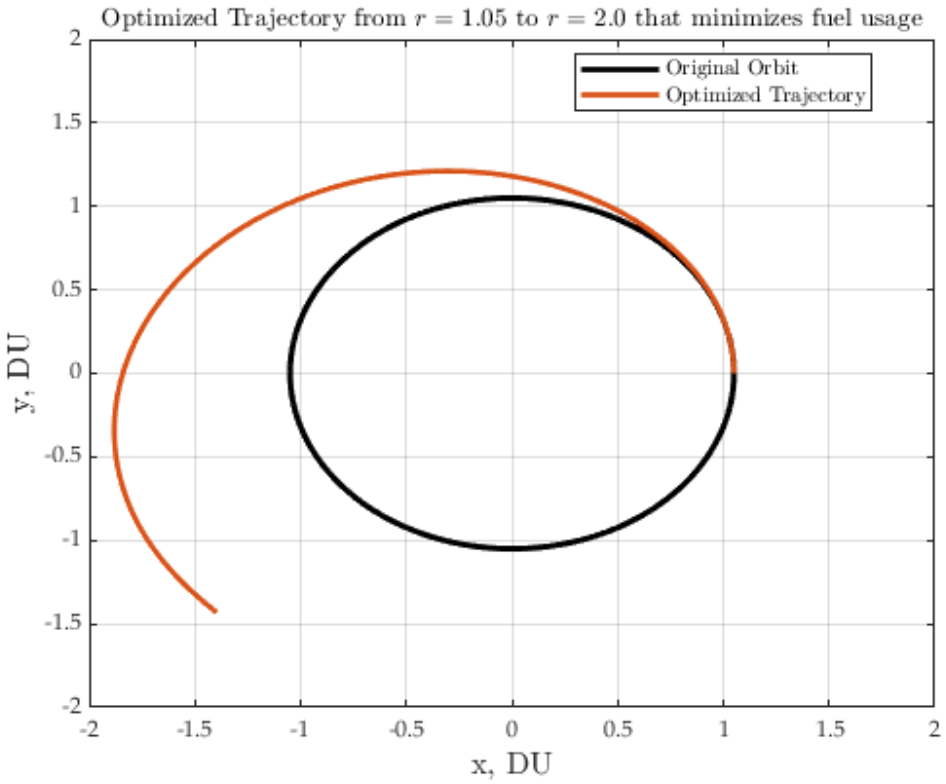
Report initial and final state values

```
s0 is:
    1.0500
   -0.0000
   -0.0000
    0.9759
    1.0000

lambda0 is:
   -0.7500
    0.0143
    0.0154
   -0.8287
   -0.7460

sf is:
   -1.3972
   -1.4310
    0.5059
   -0.4940
    0.7469

lambdaf is:
    0.1964
    0.2036
   -0.5904
    0.5832
   -1.0000
```

Plot Switching function

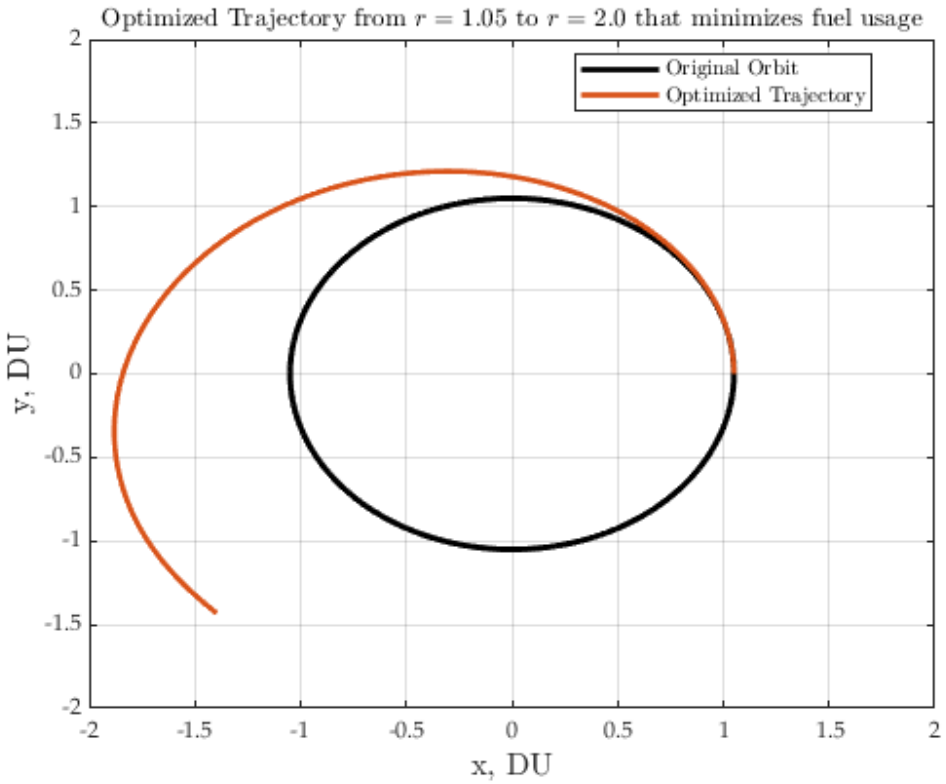


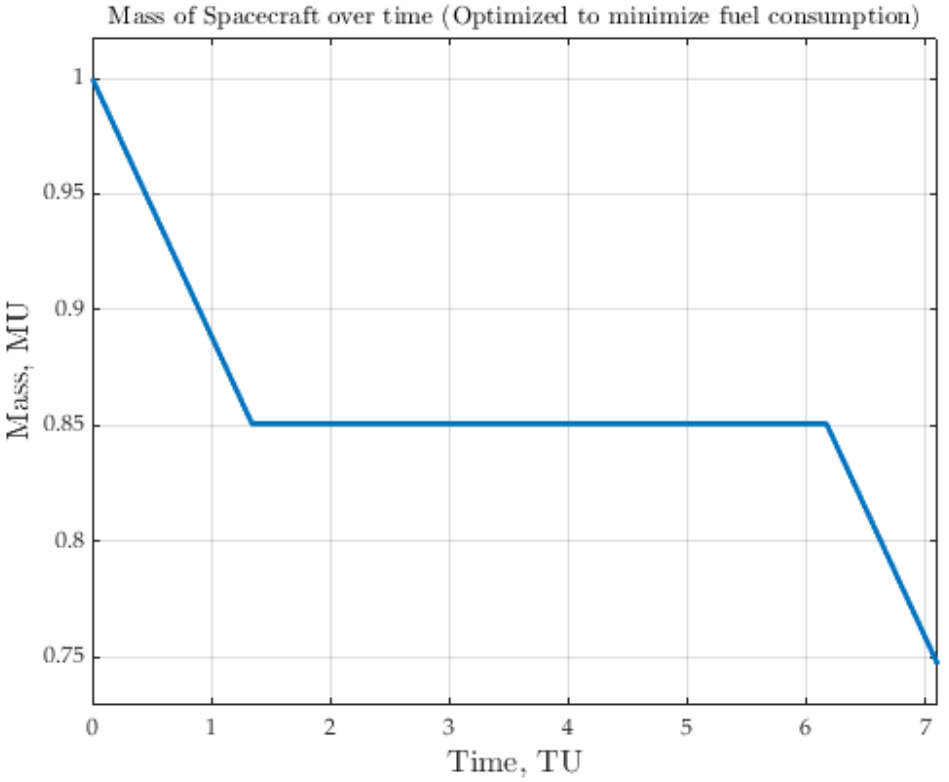
Plot trajectory



Report final mass and plot mass history

```
Final mass is: 0.74689 mass units
Compared to the 'thrust-always-on' final mass of: 0.4751 mass units
HEART CHECK: Final mass for BCB trajecetory is MORE than the thrust-always-
on; good.
```


Optimized Trajectory from $r = 1.05$ to $r = 2.0$ that minimizes fuel usage
Original Orbit
Optimized Trajectory
y, DU
x, DU
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Mass of Spacecraft over time (Optimized to minimize fuel consumption)
Mass, MU
Time, TU
1
0.95
0.9
0.85
0.8
0.75
0
1
2
3
4
5
6
7

Report Switching Times

```
ans =

         0
    1.3425
    1.3425
    6.1699
    6.1699
    7.1054

Switching times are (in TU):
         0
    1.3425
    1.3425
    6.1699
    6.1699
    7.1054
```

Published with MATLAB® R2023b

Table of Contents


```
%{
Justin Self
AERO 557 | Advanced Orbital Mechanics
Winter 2024
California Polytechnic State University SLO

HW #4, Part 2
Optimal Orbit with Minimum Fuel
%}

% Housekeeping
clear all; close all; clc;
%addpath("C://MATLAB_CODE/Orbits/")

% Performance Index = k = -mf (maximize final fuel / minimize fuel usage)

% Plan: run the code with full thrust on whole time; this will be a good
% baseline. Then run the B-C-B traj; it should TAKE MORE TIME and use MORE
% FUEL than the optimal version.
```

Problem Formulation

```
%..... Have parameters for lambda dot; find it
    % take derivative of H with respect to each val.
syms lam1 lam2 lam3 lam4 lam5 s1 s2 s3 s4 s5 c3

% Expressions derived by hand; see pdf handwork
lamv = sqrt(lam3^2 + lam4^2);
c1 = -lam3 / lamv;
c2 = -lam4 /lamv;
ve = 0.9; % given in prob statement

% Write equation derived by hand (see PDF)
H =  lam1*s3 + lam2*s4 - (lam3*s1 / (s1^2 + s2^2)^(3/2) ) - (lam4*s2 / (s1^2
+ s2^2)^(3/2) ) - (c3 / (s5*ve) ) * (lamv*ve + lam5*s5);

% Have MATLAB differentiate (NEGATIVE partial H / partial s)
```

```
Dlam1 = simplify(-diff(H,s1));
Dlam2 = simplify(-diff(H,s2));
Dlam3 = simplify(-diff(H,s3));
Dlam4 = simplify(-diff(H,s4));
Dlam5 = simplify(-diff(H,s5));

% Convert to matlabFunction so I can copy/paste into fsolve later
Dlambda{1,1} = matlabFunction(Dlam1);
Dlambda{2,1} = matlabFunction(Dlam2);
Dlambda{3,1} = matlabFunction(Dlam3);
Dlambda{4,1} = matlabFunction(Dlam4);
Dlambda{5,1} = matlabFunction(Dlam5);
```

Check initial guesses with ODE propagation

```
%..... INITIAL GUESSES
ode_options = odeset("RelTol",1e-8,"AbsTol",1e-8);
tf = 6; % By guess and check; seems enough
tspan = [0 tf];
T = 0.1; % thrust max

%..... Initial guesses (0 = initial; f = final)
state = [1.05;0;0;sqrt(1/1.05);1];     % Five state vars: x, y, xdot, ydot,
mass
lambda = [-1;-1;-1;-1;-1];             % Five lambda guesses (totally
unknown except for lam5 = -1)
y0 = [state;lambda];
[timenew,statenew] = ode45(@optimizationODE_p2,tspan,y0,ode_options,T);

% .... CHECK GUESSES
figure
rCheck = sqrt(statenew(:,1).^2 + statenew(:,2).^2); % orbital radius to
check (increasing)
plot(timenew,rCheck,'x','LineWidth',1)
hold on
plot(timenew,statenew(:,5), 'o','LineWidth',1) % mass (thrust is on full
time; should be decreasing)

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('Time, TU','Interpreter','latex');
yLab = ylabel('Initial Guesses','Interpreter','latex');
plotTitle = title('Initial guesses','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 12)
grid on
legend('Orbital Radius','Mass','interpreter','latex','Location', 'best')

disp("HEART CHECK: Initial guesses: orbital radius should increase, [fuel]
```

```
mass should decrease. Good.")

% Run fsolve
t1guess = 6;
y0 = [state;lambda;t1guess];
fs_options =
optimset('Display','iter','TolFun',1e-8,'tolx',1e-8,'MaxIter',2e3);
[x,~,~] = fsolve(@optimizationFun_p2,y0,fs_options);

%..... Propagate forward
tf = abs(x(11));

tspan = [0,tf];
[timenew,statenew] = ode45(@optimizationODE_p2,tspan,x(1:10),ode_options,T);

% Extract values
sf1 = statenew(:,1);
sf2 = statenew(:,2);
sf3 = statenew(:,3);
sf4 = statenew(:,4);
sf5 = statenew(:,5);

disp("Thrust always on configuration final mass is: " + sf5(end) + " mass
units")

lamf1 = statenew(:,6);
lamf2 = statenew(:,7);
lamf3 = statenew(:,8);
lamf4 = statenew(:,9);
lamf5 = statenew(:,10);
```

Plot switching function, first time.

```
lamfv = sqrt(lamf3.^2 + lamf4.^2);
switchF.t0 = lamfv.*ve + lamf5.*sf5;

% Plot $
figure
plot(timenew,switchF.t0,'LineWidth',2)
hold on
yline(0)
% Graph pretty
ylim padded
xlim tight
xLab = xlabel('Time, TU','Interpreter','latex');
yLab = ylabel('Switching Function \$','Interpreter','latex');
plotTitle = title('Switching Function, \$, Thrust Always
On','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
```

```
set([xLab, yLab],'FontSize', 12)
grid on
```

Burn-Coast-Burn

Best time guesses from $ plot

```
t1guess = 2.6;          % peak of plot
t2guess = 4 - t1guess;
t3guess = 4*0.2;

% Run fsolve with BURN COAST BURN on
y0 = [state;-1;0;0;-1;-1;t1guess;t2guess;t3guess]; % note the initial
guesses! Thanks Dr. A...!
fs_options =
optimset('Display','iter','TolFun',1e-8,'tolx',1e-8,'MaxIter',2e3);
[x,fval,outputs] = fsolve(@optimizationFun_p2_BurnCoastBurn,y0,fs_options);

%..... Propagate forward each time group (burn coast burn)
Ton = 0.1;
Toff = 0;

% BURN
t1 = x(11);
tspan = [0 t1];
y01 = x(1:10);
[timenew1,statenew1] = ode45(@optimizationODE_p2,tspan,y01,ode_options,Ton);

% COAST
t2 = t1 + x(12);
tspan = [t1 t2];
y02 = statenew1(end,:);
[timenew2,statenew2] = ode45(@optimizationODE_p2,tspan,y02,ode_options,Toff);

% BURN
t3 = t2 + x(13);
tspan = [t2 t3];
y03 = statenew2(end,:);
[timenew3,statenew3] = ode45(@optimizationODE_p2,tspan,y03,ode_options,Ton);

% Extract values for plotting
sf1 = [statenew1(:,1);statenew2(:,1);statenew3(:,1)];
sf2 = [statenew1(:,2);statenew2(:,2);statenew3(:,2)];
sf3 = [statenew1(:,3);statenew2(:,3);statenew3(:,3)];
sf4 = [statenew1(:,4);statenew2(:,4);statenew3(:,4)];
sf5 = [statenew1(:,5);statenew2(:,5);statenew3(:,5)];

lamf1 = [statenew1(:,6);statenew2(:,6);statenew3(:,6)];
lamf2 = [statenew1(:,7);statenew2(:,7);statenew3(:,7)];
lamf3 = [statenew1(:,8);statenew2(:,8);statenew3(:,8)];
lamf4 = [statenew1(:,9);statenew2(:,9);statenew3(:,9)];
lamf5 = [statenew1(:,10);statenew2(:,10);statenew3(:,10)];
```

```
time = [timenew1;timenew2;timenew3]; % For plotting
```

Report initial and final state values

```
disp("s0 is: ")
disp(x(1:5))

disp("lambda0 is: ")
disp(x(6:10))

disp("sf is: ")
disp([sf1(end);sf2(end);sf3(end);sf4(end);sf5(end)])

disp("lambdaf is: ")
disp([lamf1(end);lamf2(end);lamf3(end);lamf4(end);lamf5(end)])

% .... Plot switching function; final time
lamfv = sqrt(lamf3.^2 + lamf4.^2);
switchF.bcb = lamfv.*ve + lamf5.*sf5;
```

Plot Switching function

```
figure
plot(time,switchF.bcb,'LineWidth',2)
hold on
yline(0)
% Graph pretty
ylim padded
xlim tight
xLab = xlabel('Time, TU','Interpreter','latex');
yLab = ylabel('Switching Function \$','Interpreter','latex');
plotTitle = title('Switching Function, \$, Burn-Coast-
Burn','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 12)
grid on
```

Plot trajectory

```
figure
% Initial orbit
theta = linspace(0, 2*pi, 720); % 720 is the total number of points
r = 1.05;
xCenter = 0;
yCenter = 0;
x = r * cos(theta) + xCenter;
y = r * sin(theta) + yCenter;
% Plot circle.
```

```
plot(x, y, 'k-', 'LineWidth', 2);    % initial orbit
hold on
plot(sf1,sf2,"LineWidth",2)          % transfer

% Graph pretty
ylim([-2 2])
xlim([-2 2])
xLab = xlabel('x, DU','Interpreter','latex');
yLab = ylabel('y, DU','Interpreter','latex');
plotTitle = title('Optimized Trajectory from $r = 1.05$ to $r = 2.0$ that
minimizes fuel usage','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 12)
grid on
legend('Original Orbit','Optimized Trajectory',
'interpreter','latex','Location', 'best')
```

Report final mass and plot mass history

```
disp(" ")
disp("Final mass is: " + sf5(end) + " mass units")
disp("Compared to the 'thrust-always-on' final mass of: 0.4751 mass units")
disp("HEART CHECK: Final mass for BCB trajecetory is MORE than the thrust-
always-on; good.")

figure
plot(time,sf5,'LineWidth',2)
% Graph pretty
ylim padded
xlim tight
xLab = xlabel('Time, TU','Interpreter','latex');
yLab = ylabel('Mass, MU','Interpreter','latex');
plotTitle = title('Mass of Spacecraft over time (Optimized to minimize fuel
consumption)','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 12)
grid on
```

Report Switching Times

```
k = find(switchF.bcb < 1e-8 & switchF.bcb > -1e-8);
time(k)

disp("Switching times are (in TU): ")
disp(time(k))
```

```
% From plot, it will be 0, 1.3, 6.15, 7
```

Published with MATLAB® R2023b

Functions Used (all problems)

```
%{
Justin Self
California Polytechnic State University

ODE solver for optimization problems. Developed for
AERO557 | Advanced Orbital Mechanics
Dr. Abercromby, WINTER 2024

This function propagates 4 state variables and 4 costates.
THIS IS SPECIFIC TO HW 4 PART 1
%}

function dstate = optimizationODE(time,state)

a = 0.1; % acceleration
acc = a;
lambda = [state(5);state(6);state(7);state(8)];

sc1 = -state(1)*lambda(3) / sqrt( state(1)^2*lambda(3)^2 + lambda(4)^2 ) ;
% sin(c1)
cc1 = -lambda(4) / sqrt( state(1)^2*lambda(3)^2 + lambda(4)^2);
% cos(c1)

% s_bar dot
dstate(1) = state(3);
dstate(2) = state(4);
dstate(3) = state(1)*state(4)^2 - state(1)^-2 + a*sc1;
dstate(4) = state(1)^-1 * ( a*cc1 - 2*state(3)*state(4));

% lambda bar dot
lam1 = state(5);
lam2 = state(6);
lam3 = state(7);
lam4 = state(8);
s1 = state(1);
s2 = state(2);
s3 = state(3);
s4 = state(4);

% Copied from script "matlabFunction"
dlam = [-1.0./s1.^3.*1.0./
sqrt(lam4.^2+lam3.^2.*s1.^2).*(lam3.*sqrt(lam4.^2+lam3.^2.*s1.^2).*2.0+acc.*l
am4.^2.*s1+lam3.*s1.^3.*s4.^2.*sqrt(lam4.^2+lam3.^2.*s1.^2)+lam4.*s1.*s3.*s4.
*sqrt(lam4.^2+lam3.^2.*s1.^2).*2.0);
        0;
        2*lambda(4)*state(4)/state(1) - lambda(1);
        2*lambda(4)*state(3) / state(1) - lambda(2) -
2*lambda(3)*state(1)*state(4)];

% Combine
dstate = [dstate(1);dstate(2);dstate(3);dstate(4);dlam];
```

```
end % function
```

Published with MATLAB® R2023b

```
%{
Justin Self
California Polytechnic State University

FSOLVE solver for optimization problems. Developed for
AERO557 | Advanced Orbital Mechanics
Dr. Abercromby, WINTER 2024
HW #4, part 1

This function performes trajectory optimization for 2D orbit state with
known constraints. VERY SPECIFIC TO HW 4.1 IN A557.
%}

function F = optimizationFun(state,tf)

% Call ode
tspan = [0,tf];
options = odeset("RelTol",1e-8,"AbsTol",1e-8);
[~,statenew] = ode45(@optimizationODE,tspan,state,options);

% Extract s0, sf, and lambdaf from propagated state
s0 = [statenew(1,1:4)];
sf = [statenew(end,1),statenew(end,2),statenew(end,3),statenew(end,4)];
lambdaf = [statenew(end,5),statenew(end,6),statenew(end,7),statenew(end,8)];

% Set up equations to check misclosures *NOT GENERAL; THIS IS SPECIFIC*
F(1,1) = sf(3);                                                          %
Omega1
F(2,1) = sf(4)^2*sf(1)^3 - 1;                                            %
Omega 2 = thetadotf^2 * rf^3 - 1
F(3,1) = lambdaf(2);                                                     %
lambda_f 2 = 0
F(4,1) = -1 + ( (3*lambdaf(4) * sf(4) ) / (2*sf(1))  ) - lambdaf(1);     %
lambda_f 1

% Initial conditions (r,rdot,theta,thetadot) @ t0

F(5,1) = s0(1) - 1.05;  % r0 = 1.05
F(6,1) = s0(2);         % rdot0 = 0
F(7,1) = s0(3);         % theta0 = 0
F(8,1) = s0(4) - 1.05^(-3/2);

end % fsolve function
```

Published with MATLAB® R2023b

Table of Contents


```
%{
Justin Self
California Polytechnic State University

ODE solver for optimization problems. Developed for
AERO557 | Advanced Orbital Mechanics
Dr. Abercromby, WINTER 2024

This function propagates 4 state variables and 4 costates.
THIS IS SPECIFIC TO HW 4 PART 2
%}

function dstate = optimizationODE_p2(time,state,T)

s1 = state(1);
s2 = state(2);
s3 = state(3);
s4 = state(4);
s5 = state(5);

% Five lambdas
lam1 = state(6);
lam2 = state(7);
lam3 = state(8);
lam4 = state(9);
lam5 = state(10);

lambdaV = sqrt(lam3^2 + lam4^2);

ve = 0.9;
c3 = T;
c2 = -lam4 / lambdaV;
c1 = -lam3 / lambdaV;

% s_bar dot
dstate = [  s3;
            s4;
            -s1 / (s1^2 + s2^2)^(3/2) + (c3*c1 / s5);
            -s2 / (s1^2 + s2^2)^(3/2) + (c3*c2 / s5);
            -c3 / ve];


dlam = [-1.0./(s1.^2+s2.^2).^(5.0./2.0).*(lam3.*s1.^2.*2.0-
lam3.*s2.^2+lam4.*s1.*s2.*3.0);
        -1.0./(s1.^2+s2.^2).^(5.0./2.0).*(-
lam4.*s1.^2+lam4.*s2.^2.*2.0+lam3.*s1.*s2.*3.0);
        -lam1;
```

```
        -lam2;
        -c3.*1.0./s5.^2.*sqrt(lam3.^2+lam4.^2)];

% Combine
dstate = [dstate;dlam];

end % function
```

Published with MATLAB® R2023b

Table of Contents


```
%{
Justin Self
California Polytechnic State University

FSOLVE solver for optimization problems. Developed for
AERO557 | Advanced Orbital Mechanics
Dr. Abercromby, WINTER 2024
HW #4, part 1

This function performes trajectory optimization for 2D orbit state with
known constraints. VERY SPECIFIC TO HW 4 IN A557.

FIRST TIME THROUGH PROBLEM. THRUST IS ALWAYS ON.
%}

function F = optimizationFun_p2(state)

c3 = 0.1; % thrust max

tf = abs(state(11));

%..... Call ode
tspan = [0,tf];
T = c3;
options = odeset("RelTol",1e-8,"AbsTol",1e-8);
[~,statenew] = ode45(@optimizationODE_p2,tspan,state(1:10),options,T);

% Extract s0, sf, lambda0, and lambdaf from propagated state
s0 = statenew(1,1:5);
sf = statenew(end,1:5);
lambdaf = statenew(end,6:10);

% Set up equations to check misclosures
% Omega (final constraints)
F(1,1) = sqrt(sf(1)^2 + sf(2)^2) - 2;
F(2,1) = sf(3)^2 + sf(4)^2 - 0.5;
F(3,1) = sf(4)*sf(1) - sf(3)*sf(2) - sqrt(2);

% Lambda 5f = -1
F(4,1) = lambdaf(5) + 1;

% Dependency Equation
a = sf(3) + (sf(2)/sf(1) * sf(4));
b = sf(1) + (sf(4)/sf(3) * sf(2));
F(5,1) = -lambdaf(1)*(sf(2) / sf(1)) +...
    lambdaf(2) -...
    lambdaf(3)*(sf(4)/sf(3)) * ( a / b) + ...
```

```
    lambdaf(4)* (a / b);

% NBC for free time; H(tf) = 0
F(6,1) = lambdaf(1)*sf(3) + ...
         lambdaf(2)*sf(4) - ...
         lambdaf(3)*sf(1) / (sf(1)^2 + sf(2)^2)^(3/2) - ...
         lambdaf(4)*sf(2) / (sf(1)^2 + sf(2)^2)^(3/2);

% Five initial condition equations
F(9,1) = s0(1) - 1.05;
F(10,1) = s0(2);
F(11,1) = s0(3);
F(12,1) = s0(4) - sqrt(1/1.05); % velocity = s04 = sqrt(mu/r)
F(13,1) = s0(5) - 1;            % mass t0 = 1

end % fsolve function
```

Published with MATLAB® R2023b

Table of Contents


```
%{
Justin Self
California Polytechnic State University

FSOLVE solver for optimization problems. Developed for
AERO557 | Advanced Orbital Mechanics
Dr. Abercromby, WINTER 2024
HW #4, part 1

This function performes trajectory optimization for 2D orbit state with
known constraints. VERY SPECIFIC TO HW 4.1 IN A557.

BURN COAST BURN TRAJECTORY
%}

function F = optimizationFun_p2_BurnCoastBurn(state)
ve = 0.9; % given
c3 = 0.1; % thrust max

t1guess = state(11);
t2guess = state(12);
t3guess = state(13);

t1 = abs(t1guess);
t2 = t1 + abs(t2guess);
t3 = t2 + abs(t3guess);

%..... Call ode (t1) - THRUST ON
T = c3;
tspan = [0,t1];
options = odeset("RelTol",1e-8,"AbsTol",1e-8);
[~,statenew1] = ode45(@optimizationODE_p2,tspan,state(1:10),options,T);

%..... Call ode (t2) - THRUST OFF
T = 0;
tspan = [t1,t2];
options = odeset("RelTol",1e-8,"AbsTol",1e-8);
[~,statenew2] = ode45(@optimizationODE_p2,tspan,statenew1(end,:),options,T);

%..... Call ode (t3) - THRUST ON
T = c3;
tspan = [t2,t3];
options = odeset("RelTol",1e-8,"AbsTol",1e-8);
[~,statenew3] = ode45(@optimizationODE_p2,tspan,statenew2(end,:),options,T);

% Extract values for fsolve to use
s0 = statenew1(1,1:5);
```

```
lambda0 = statenew1(1,6:10);
s1 = statenew1(end,1:5);
lambda1 = statenew1(end,6:10);
s2 = statenew2(end,1:5);
lambda2 = statenew2(end,6:10);
sf = statenew3(end,1:5);
lambdaf = statenew3(end,6:10);

lamv0 = sqrt(lambda0(3)^2 + lambda0(4)^2);
lamv1 = sqrt(lambda1(3)^2 + lambda1(4)^2);
lamv2 = sqrt(lambda2(3)^2 + lambda2(4)^2);
lamvf = sqrt(lambdaf(3)^2 + lambdaf(4)^2);

% Set up equations to check misclosures
% Omega (final constraints)
F(1,1) = sqrt(sf(1)^2 + sf(2)^2) - 2;
F(2,1) = sf(3)^2 + sf(4)^2 - 0.5;
F(3,1) = sf(4)*sf(1) - sf(3)*sf(2) - sqrt(2);

% Lambda 5f = -1
F(4,1) = lambdaf(5) + 1;

% Dependency Equation
a = sf(3) + (sf(2)/sf(1) * sf(4));
b = sf(1) + (sf(4)/sf(3) * sf(2));
F(5,1) = -lambdaf(1)*(sf(2) / sf(1)) +...
    lambdaf(2) -...
    lambdaf(3)*(sf(4)/sf(3)) * ( a / b) + ...
    lambdaf(4)* (a / b);

% NBC for free time; H(tf) = 0
F(6,1) = lambdaf(1)*sf(3) + ...
        lambdaf(2)*sf(4) - ...
        lambdaf(3)*sf(1) / (sf(1)^2 + sf(2)^2)^(3/2) - ...
        lambdaf(4)*sf(2) / (sf(1)^2 + sf(2)^2)^(3/2);

% Switching function at all times
F(7,1) = lamv0 * ve + lambda0(5) * s0(5); % switching function at t0
F(8,1) = lamv1 * ve + lambda1(5) * s1(5);
F(9,1) = lamv2 * ve + lambda2(5) * s2(5);
F(10,1) = lamvf * ve + lambdaf(5) * sf(5); % switching function at tf


% Five initial condition equations
F(11,1) = s0(1) - 1.05;
F(12,1) = s0(2);
F(13,1) = s0(3);
F(14,1) = s0(4) - sqrt(1/1.05); % velocity = s04 = sqrt(mu/r)
F(15,1) = s0(5) - 1;            % mass t0 = 1

end % fsolve function
```

Published with MATLAB® R2023b