

EML6934 Optimal Control Final Project

Elias Reyes

April 18, 2022

Contents

1	Problem Statement	1
2	Differential Equations of Motion	1
2.1	Position, Velocity and Acceleration of the Spacecraft	1
2.2	Newton's Second Law for A Particle	3
2.3	Conversion to First-Order Equations	3
3	Formulation of Optimal Control Problem	5
4	Numerical Solution of Optimal Control Problem	5
4.1	Minimize Terminal Time with Unconstrained Control	6
4.2	Maximize Terminal Mass with Unconstrained Control	22
4.3	Minimize Terminal Time with Constrained Control	38
4.4	Maximize Terminal Mass with Constrained Control	59
4.5	Overall Analysis	80
5	Future Work	80
6	Appendix	80

List of Figures

1	Schematic of particle moving in an inertially fixed plane	2
2	Reference frame rotation	2
3	Thrust Force at Angle β	2
4	Free Body Diagram	4
5	States for trajectory that minimized terminal time ($N : 3, K : 2$)	6
6	Control that minimized terminal time ($N : 3, K : 2$)	7
7	Trajectory from initial to final orbit ($N : 3, K : 2$)	7
8	States for trajectory that minimized terminal time ($N : 3, K : 4$)	8
9	Control that minimized terminal time ($N : 3, K : 4$)	8
10	Trajectory from initial to final orbit ($N : 3, K : 4$)	9
11	States for trajectory that minimized terminal time ($N : 3, K : 8$)	9
12	Control that minimized terminal time ($N : 3, K : 8$)	10
13	Trajectory from initial to final orbit ($N : 3, K : 8$)	10
14	States for trajectory that minimized terminal time ($N : 3, K : 16$)	11
15	Control that minimized terminal time ($N : 3, K : 16$)	11
16	Trajectory from initial to final orbit ($N : 3, K : 16$)	12
17	States for trajectory that minimized terminal time ($N : 3, K : 32$)	12
18	Control that minimized terminal time ($N : 3, K : 32$)	13
19	Trajectory from initial to final orbit ($N : 3, K : 32$)	13
20	States for trajectory that minimized terminal time ($N : 4, K : 2$)	14
21	Control that minimized terminal time ($N : 4, K : 2$)	14
22	Trajectory from initial to final orbit ($N : 4, K : 2$)	15
23	States for trajectory that minimized terminal time ($N : 4, K : 4$)	15
24	Control that minimized terminal time ($N : 4, K : 4$)	16
25	Trajectory from initial to final orbit ($N : 4, K : 4$)	16
26	States for trajectory that minimized terminal time ($N : 4, K : 8$)	17
27	Control that minimized terminal time ($N : 4, K : 8$)	17
28	Trajectory from initial to final orbit ($N : 4, K : 8$)	18
29	States for trajectory that minimized terminal time ($N : 4, K : 16$)	18
30	Control that minimized terminal time ($N : 4, K : 16$)	19
31	Trajectory from initial to final orbit ($N : 4, K : 16$)	19
32	States for trajectory that minimized terminal time ($N : 4, K : 32$)	20
33	Control that minimized terminal time ($N : 4, K : 32$)	20
34	Trajectory from initial to final orbit ($N : 4, K : 32$)	21
35	States for trajectory that maximized terminal mass ($N : 3, K : 2$)	23
36	Control that maximized terminal mass ($N : 3, K : 2$)	23
37	Trajectory from initial to final orbit ($N : 3, K : 2$)	24
38	States for trajectory that maximized terminal mass ($N : 3, K : 4$)	24
39	Control that maximized terminal mass ($N : 3, K : 4$)	25
40	Trajectory from initial to final orbit ($N : 3, K : 4$)	25
41	States for trajectory that maximized terminal mass ($N : 3, K : 8$)	26
42	Control that maximized terminal mass ($N : 3, K : 8$)	26
43	Trajectory from initial to final orbit ($N : 3, K : 8$)	27
44	States for trajectory that maximized terminal mass ($N : 3, K : 16$)	27

45	Control that maximized terminal mass ($N : 3, K : 16$)	28
46	Trajectory from initial to final orbit ($N : 3, K : 16$)	28
47	States for trajectory that maximized terminal mass ($N : 3, K : 32$)	29
48	Control that maximized terminal mass ($N : 3, K : 32$)	29
49	Trajectory from initial to final orbit ($N : 3, K : 32$)	30
50	States for trajectory that maximized terminal mass ($N : 4, K : 2$)	30
51	Control that maximized terminal mass ($N : 4, K : 2$)	31
52	Trajectory from initial to final orbit ($N : 4, K : 2$)	31
53	States for trajectory that maximized terminal mass ($N : 4, K : 4$)	32
54	Control that maximized terminal mass ($N : 4, K : 4$)	32
55	Trajectory from initial to final orbit ($N : 4, K : 4$)	33
56	States for trajectory that maximized terminal mass ($N : 4, K : 8$)	33
57	Control that maximized terminal mass ($N : 4, K : 8$)	34
58	Trajectory from initial to final orbit ($N : 4, K : 8$)	34
59	States for trajectory that maximized terminal mass ($N : 4, K : 16$)	35
60	Control that maximized terminal mass ($N : 4, K : 16$)	35
61	Trajectory from initial to final orbit ($N : 4, K : 16$)	36
62	States for trajectory that maximized terminal mass ($N : 4, K : 32$)	36
63	Control that maximized terminal mass ($N : 4, K : 32$)	37
64	Trajectory from initial to final orbit ($N : 4, K : 32$)	37
65	States for trajectory that minimized terminal time ($N : 3, K : 2$)	39
66	Control that minimized terminal time ($N : 3, K : 2$)	39
67	Path constrained control that minimized terminal time ($N : 3, K : 2$)	40
68	Trajectory from initial to final orbit ($N : 3, K : 2$)	40
69	States for trajectory that minimized terminal time ($N : 3, K : 4$)	41
70	Control that minimized terminal time ($N : 3, K : 4$)	41
71	Path constrained control that minimized terminal time ($N : 3, K : 4$)	42
72	Trajectory from initial to final orbit ($N : 3, K : 4$)	42
73	States for trajectory that minimized terminal time ($N : 3, K : 8$)	43
74	Control that minimized terminal time ($N : 3, K : 8$)	43
75	Path constrained control that minimized terminal time ($N : 3, K : 8$)	44
76	Trajectory from initial to final orbit ($N : 3, K : 8$)	44
77	States for trajectory that minimized terminal time ($N : 3, K : 16$)	45
78	Control that minimized terminal time ($N : 3, K : 16$)	45
79	Path constrained control that minimized terminal time ($N : 3, K : 16$)	46
80	Trajectory from initial to final orbit ($N : 3, K : 16$)	46
81	States for trajectory that minimized terminal time ($N : 3, K : 32$)	47
82	Control that minimized terminal time ($N : 3, K : 32$)	47
83	Path constrained control that minimized terminal time ($N : 3, K : 32$)	48
84	Trajectory from initial to final orbit ($N : 3, K : 32$)	48
85	States for trajectory that minimized terminal time ($N : 4, K : 2$)	49

86	Control that minimized terminal time ($N : 4, K : 2$)	49
87	Path constrained control that minimized terminal time ($N : 4, K : 2$)	50
88	Trajectory from initial to final orbit ($N : 4, K : 2$)	50
89	States for trajectory that minimized terminal time ($N : 4, K : 4$)	51
90	Control that minimized terminal time ($N : 4, K : 4$)	51
91	Path constrained control that minimized terminal time ($N : 4, K : 4$)	52
92	Trajectory from initial to final orbit ($N : 4, K : 4$)	52
93	States for trajectory that minimized terminal time ($N : 4, K : 8$)	53
94	Control that minimized terminal time ($N : 4, K : 8$)	53
95	Path constrained control that minimized terminal time ($N : 4, K : 8$)	54
96	Trajectory from initial to final orbit ($N : 4, K : 8$)	54
97	States for trajectory that minimized terminal time ($N : 4, K : 16$)	55
98	Control that minimized terminal time ($N : 4, K : 16$)	55
99	Path constrained control that minimized terminal time ($N : 4, K : 16$)	56
100	Trajectory from initial to final orbit ($N : 4, K : 16$)	56
101	States for trajectory that minimized terminal time ($N : 4, K : 32$)	57
102	Control that minimized terminal time ($N : 4, K : 32$)	57
103	Path constrained control that minimized terminal time ($N : 4, K : 32$)	58
104	Trajectory from initial to final orbit ($N : 4, K : 32$)	58
105	States for trajectory that maximized terminal mass ($N : 3, K : 2$)	60
106	Control that maximized terminal mass ($N : 3, K : 2$)	60
107	Path constrained control that maximized terminal mass ($N : 3, K : 2$)	61
108	Trajectory from initial to final orbit ($N : 3, K : 2$)	61
109	States for trajectory that maximized terminal mass ($N : 3, K : 4$)	62
110	Control that maximized terminal mass ($N : 3, K : 4$)	62
111	Path constrained control that maximized terminal mass ($N : 3, K : 4$)	63
112	Trajectory from initial to final orbit ($N : 3, K : 4$)	63
113	States for trajectory that maximized terminal mass ($N : 3, K : 8$)	64
114	Control that maximized terminal mass ($N : 3, K : 8$)	64
115	Path constrained control that maximized terminal mass ($N : 3, K : 8$)	65
116	Trajectory from initial to final orbit ($N : 3, K : 8$)	65
117	States for trajectory that maximized terminal mass ($N : 3, K : 16$)	66
118	Control that maximized terminal mass ($N : 3, K : 16$)	66
119	Path constrained control that maximized terminal mass ($N : 3, K : 16$)	67
120	Trajectory from initial to final orbit ($N : 3, K : 16$)	67
121	States for trajectory that maximized terminal mass ($N : 3, K : 32$)	68
122	Control that maximized terminal mass ($N : 3, K : 32$)	68

123	Path constrained control that maximized terminal mass ($N : 3, K : 32$)	69
124	Trajectory from initial to final orbit ($N : 3, K : 32$)	69
125	States for trajectory that maximized terminal mass ($N : 4, K : 2$)	70
126	Control that maximized terminal mass ($N : 4, K : 2$)	70
127	Path constrained control that maximized terminal mass ($N : 4, K : 2$)	71
128	Trajectory from initial to final orbit ($N : 4, K : 2$)	71
129	States for trajectory that maximized terminal mass ($N : 4, K : 4$)	72
130	Control that maximized terminal mass ($N : 4, K : 4$)	72
131	Path constrained control that maximized terminal mass ($N : 4, K : 4$)	73
132	Trajectory from initial to final orbit ($N : 4, K : 4$)	73
133	States for trajectory that maximized terminal mass ($N : 4, K : 8$)	74
134	Control that maximized terminal mass ($N : 4, K : 8$)	74
135	Path constrained control that maximized terminal mass ($N : 4, K : 8$)	75
136	Trajectory from initial to final orbit ($N : 4, K : 8$)	75
137	States for trajectory that maximized terminal mass ($N : 4, K : 16$)	76
138	Control that maximized terminal mass ($N : 4, K : 16$)	76
139	Path constrained control that maximized terminal mass ($N : 4, K : 16$)	77
140	Trajectory from initial to final orbit ($N : 4, K : 16$)	77
141	States for trajectory that maximized terminal mass ($N : 4, K : 32$)	78
142	Control that maximized terminal mass ($N : 4, K : 32$)	78
143	Path constrained control that maximized terminal mass ($N : 4, K : 32$)	79
144	Trajectory from initial to final orbit ($N : 4, K : 32$)	79

List of Tables

1	Results for minimizing t_f with unconstrained control	21
2	Results for maximizing m_f with unconstrained control	22
3	Results for minimizing t_f with constrained control	38
4	Results for maximizing m_f with constrained control	59

Listings

1	orbitTransferMain.m	80
2	orbitTransferFun.m	87
3	orbitTransferObj.m	89
4	orbitTransferCon.m	90
5	orbitTransferGrd.m	90
6	orbitTransferJac.m	91
7	orbitTransferJacPat.m	91

1 Problem Statement

2 Differential Equations of Motion

The differential equations of motion were derived using both Newton's second law. The schematic for the problem can be seen Figure 1. The spacecraft is modeled as point P of mass m . The spacecraft moves relative to an inertial reference frame l . The reference frame fixed in l is expressed as $\{e_x, e_y, e_z\}$. The position of the spacecraft is denoted as $r_{P/O}$, where O is modeled as the sun, fixed in l . The spacecraft is parameterized in the basis $\{u_r, u_\theta, u_z\}$, where the rotation is about $u_z = e_z$. The rotation creates an angle θ between e_x and u_r , which can be seen in Figure 2. Two forces are said to act on the spacecraft. The first is the gravitational force which is given as

$$G = -m\mu \frac{r_{P/O}}{\|r_{P/O}\|^3}, \quad (1)$$

while the second is the thrust force given as

$$T = Tw, \quad (2)$$

where w is the unit vector that lies an angle β from the direction u_θ as seen in Figure 3.

2.1 Position, Velocity and Acceleration of the Spacecraft

As seen in Figure 1, the position of the spacecraft represented in reference frame A , is given as

$${}^A\vec{r}_{P/O} = ru_r. \quad (3)$$

The velocity can then be represented in the inertial frame l by using equation 4, where ${}^l\vec{\omega}^A$ is the angular velocity between reference frame l and A .

$$\frac{{}^l d\vec{r}_{P/O}}{dt} = \frac{{}^A d\vec{r}_{P/O}}{dt} + {}^l\vec{\omega}^A \times {}^A\vec{r}_{P/O} \quad (4)$$

Using Equation 4, the velocity of the spacecraft in the inertial frame is then formulated as

$$\begin{aligned} {}^l\vec{v}_p &= \frac{{}^l d\vec{r}_{P/O}}{dt} \\ {}^l\vec{v}_p &= \dot{r}u_r + \dot{\theta}u_z \times ru_r \\ {}^l\vec{v}_p &= \dot{r}u_r + \dot{\theta}ru_\theta \end{aligned} \quad (5)$$

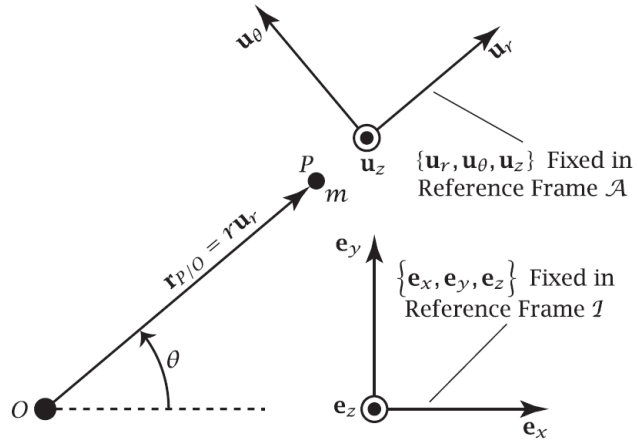


Figure 1: Schematic of particle moving in an inertially fixed plane

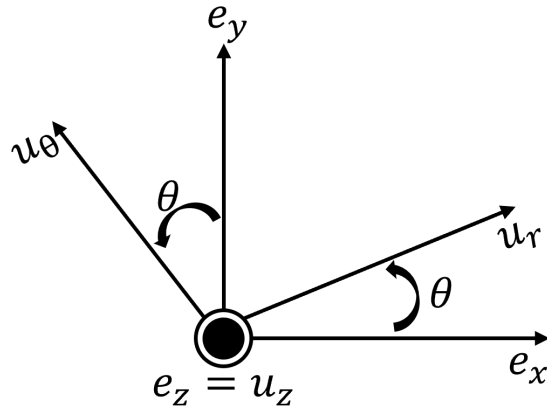


Figure 2: Reference frame rotation

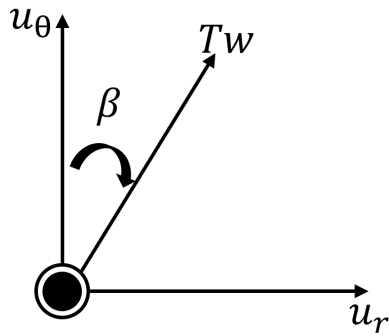


Figure 3: Thrust Force at Angle β

The acceleration of the particle can then be formulated as

$$\begin{aligned}
{}^l\vec{a}_p &= \frac{{}^l d\vec{v}_P}{dt} \\
{}^l\vec{a}_p &= \frac{{}^A d\vec{v}_P}{dt} + {}^l\vec{\omega}^A \times {}^A\vec{v}_P \\
{}^l\vec{a}_p &= \ddot{r}u_r + (\ddot{\theta}r + \dot{\theta}\dot{r})u_\theta + \dot{\theta}\dot{r}u_\theta - \dot{\theta}^2ru_r \\
{}^l\vec{a}_p &= (\ddot{r} - \dot{\theta}^2r)u_r + (\ddot{\theta}r + 2\dot{\theta}\dot{r})u_\theta
\end{aligned} \tag{6}$$

2.2 Newton's Second Law for A Particle

Newton's second law for a particle is represented by

$$\sum F_P = m_P * a_P. \tag{7}$$

Figure 4 represents the free body diagram of the particle system. F_G , represented by equation 1, is the gravitational force and acts along the u_r direction. F_T , represented by equation 2, is the thrust force and acts in the direction w . It can be seen in Figure 3 that the thrust force can be re-written as

$$F_T = T \sin(\beta)u_r + T \cos(\beta)u_\theta, \tag{8}$$

while the gravitational force can be written as

$$F_G = -m\mu \frac{1}{r^2}u_r. \tag{9}$$

We can now substitute equations 6, 8, and 9 into equation 7 to obtain

$$-m\mu \frac{1}{r^2}u_r + T \sin(\beta)u_r + T \cos(\beta)u_\theta = m[(\ddot{r} - \dot{\theta}^2r)u_r + (\ddot{\theta}r + 2\dot{\theta}\dot{r})u_\theta].$$

After equating terms, the two equations of motion using Newtons Seconds become:

$$(u_r) \quad \ddot{r} = \dot{\theta}^2r - \frac{\mu}{r^2} + \frac{T \sin(\beta)}{m} \tag{10}$$

$$(u_\theta) \quad \ddot{\theta} = -\frac{2\dot{r}\dot{\theta}}{r} + \frac{T \cos(\beta)}{mr} \tag{11}$$

2.3 Conversion to First-Order Equations

To re-write the two second-order equations into first four first-order equations, the following substitutions can be made:

$$\dot{r} = v_r, \tag{12}$$

$$r\dot{\theta} = v_\theta,$$

$$\dot{\theta} = \frac{v_\theta}{r}. \tag{13}$$

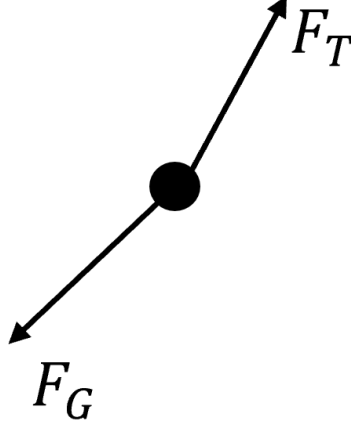


Figure 4: Free Body Diagram

Then, taking the derivatives of r and θ of equations 12 and 13, respectively, we obtain:

$$\ddot{r} = \dot{v}_r, \quad (14)$$

$$\ddot{\theta} = \frac{\dot{v}_\theta r - v_\theta \dot{r}}{r^2}. \quad (15)$$

After substituting equations 12, 13, 14, and 15 into equations 10 and 11, two first-order equations are derived as:

$$\begin{aligned} \dot{v}_r &= \frac{v_\theta^2 r}{r^2} - \frac{\mu}{r^2} + \frac{T \sin(\beta)}{m}, \\ \dot{v}_\theta &= \frac{v_\theta^2}{r} - \frac{\mu}{r^2} + \frac{T \sin(\beta)}{m}, \end{aligned} \quad (16)$$

and,

$$\begin{aligned} \frac{\dot{v}_\theta r - v_\theta v_r}{r^2} &= -\frac{2v_\theta v_r}{r^2} + \frac{T \cos(\beta)}{mr}, \\ \dot{v}_\theta r - v_\theta v_r &= -\frac{2v_\theta v_r r^2}{r^2} + \frac{T \cos(\beta) r^2}{mr}, \\ \dot{v}_\theta &= -\frac{2v_\theta v_r}{r} + \frac{T \cos(\beta)}{m} + \frac{v_\theta v_r}{r}, \\ \dot{v}_\theta &= -\frac{v_\theta v_r}{r} + \frac{T \cos(\beta)}{m}. \end{aligned} \quad (17)$$

Next, a fifth first-order equation is given as

$$\dot{m} = -\frac{T}{v_e}. \quad (18)$$

The five first-order differential equations are listed below:

$$\begin{aligned}\dot{r} &= v_r, \\ \dot{\theta} &= \frac{v_\theta}{r}, \\ \dot{v}_r &= \frac{v_\theta^2}{r} - \frac{\mu}{r^2} + \frac{T \sin(\beta)}{m}, \\ \dot{v}_\theta &= -\frac{v_\theta v_r}{r} + \frac{T \cos(\beta)}{m}, \\ \dot{m} &= -\frac{T}{v_e}.\end{aligned}$$

3 Formulation of Optimal Control Problem

There are two objectives for the optimal control problem. The first objective is to minimize the time to transfer from an initial circular orbit to a final circular orbit. The second objective is to maximize the terminal mass when transferring from an initial circular orbit to a final circular orbit. Below is an overview of the problem:

$$\begin{aligned}\text{Objective 1 : } & \min t_f \\ \text{Objective 2 : } & \max m(t_f) \\ \text{State : } & r(t), \theta(t), v_r(t), v_\theta(t), m(t) \\ \text{Control : } & \beta(t), T(t)\end{aligned}$$

4 Numerical Solution of Optimal Control Problem

The optimal control problem was solved using multiple interval Legendre-Gauss-Radau collocation. For simplicity, Adigator was used for automatic differentiation and IPOPT was used for the nonlinear optimization. For this study, four different cases were investigated. The four different discretizations of Legendre-Gauss-Radau collocation are:

1. Minimize t_f with with unconstrained control parameterized by polynomial degrees $N = (3,4)$ with intervals of $K = (2,4,8,16,32)$
2. Minimize $m(t_f)$ with with unconstrained control parameterized by polynomial degrees $N = (3,4)$ with intervals of $K = (2,4,8,16,32)$
3. Minimize t_f with with constrained control parameterized by polynomial degrees $N = (3,4)$ with intervals of $K = (2,4,8,16,32)$
4. Minimize $m(t_f)$ with with constrained control parameterized by polynomial degrees $N = (3,4)$ with intervals of $K = (2,4,8,16,32)$

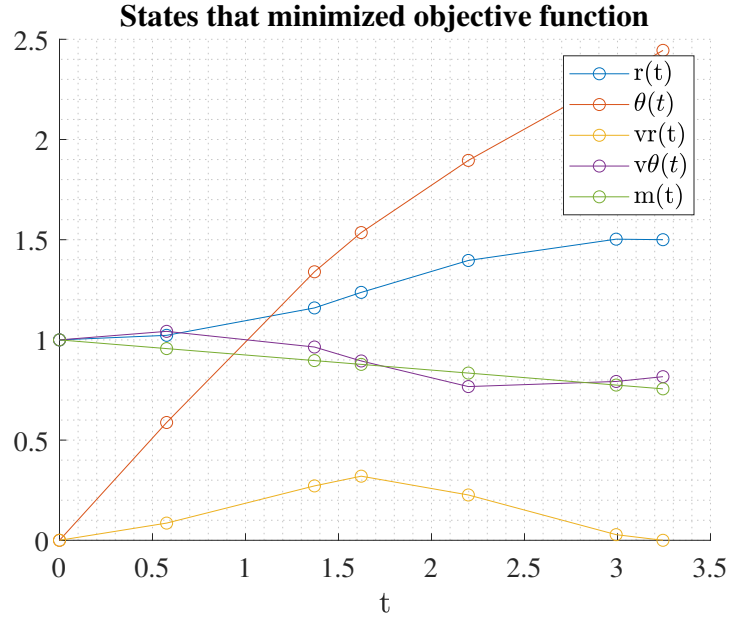


Figure 5: States for trajectory that minimized terminal time ($N : 3, K : 2$)

4.1 Minimize Terminal Time with Unconstrained Control

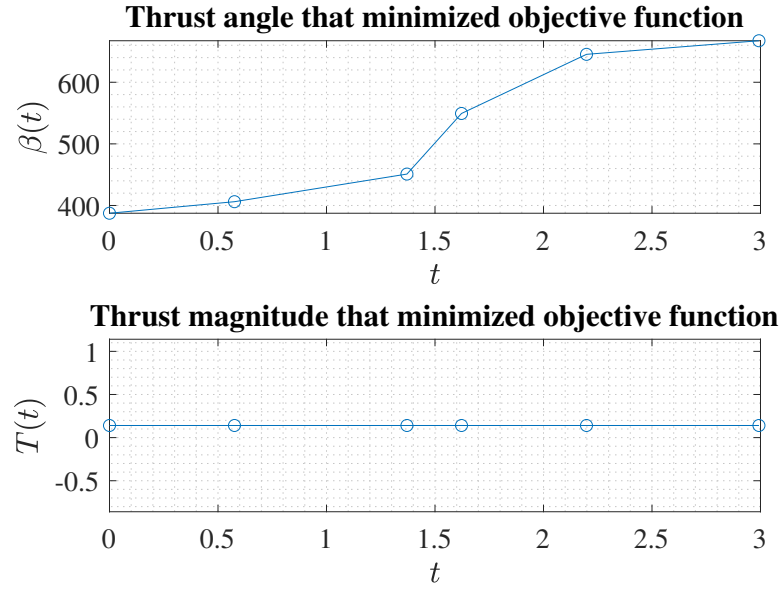


Figure 6: Control that minimized terminal time ($N : 3$, $K : 2$)

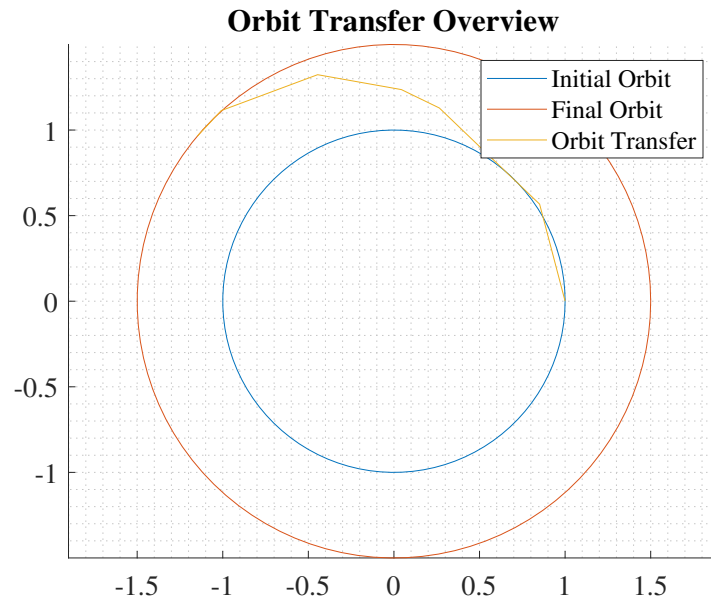


Figure 7: Trajectory from initial to final orbit ($N : 3$, $K : 2$)

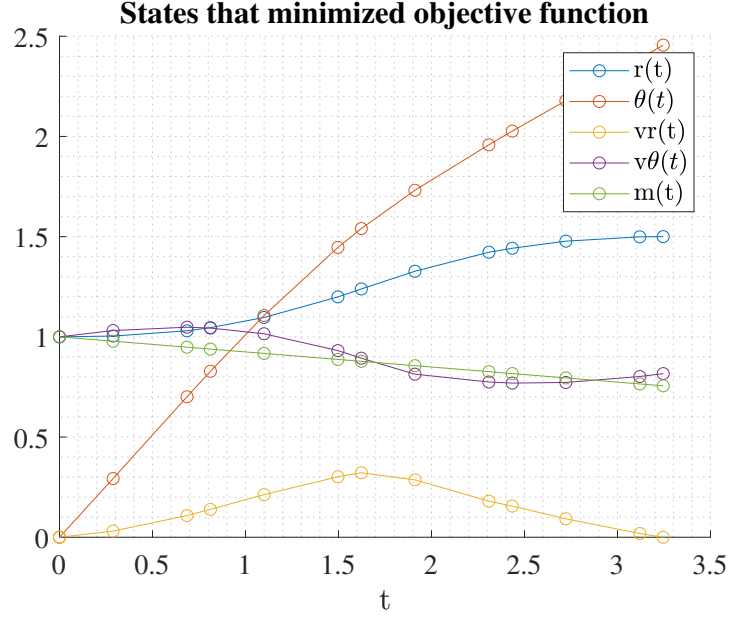


Figure 8: States for trajectory that minimized terminal time ($N : 3, K : 4$)

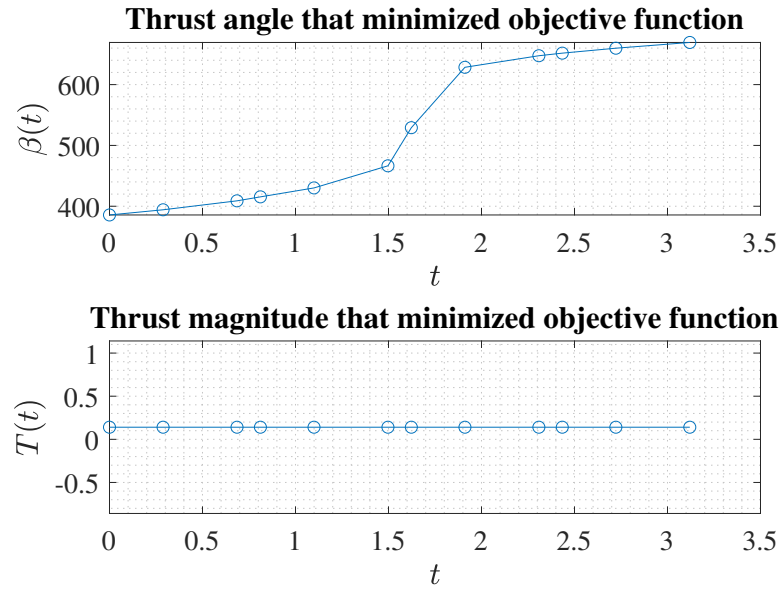


Figure 9: Control that minimized terminal time ($N : 3, K : 4$)

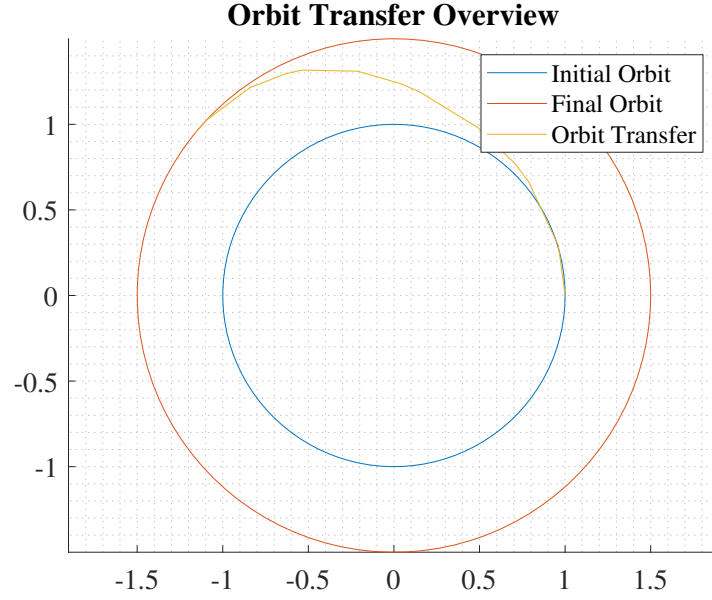


Figure 10: Trajectory from initial to final orbit ($N : 3, K : 4$)

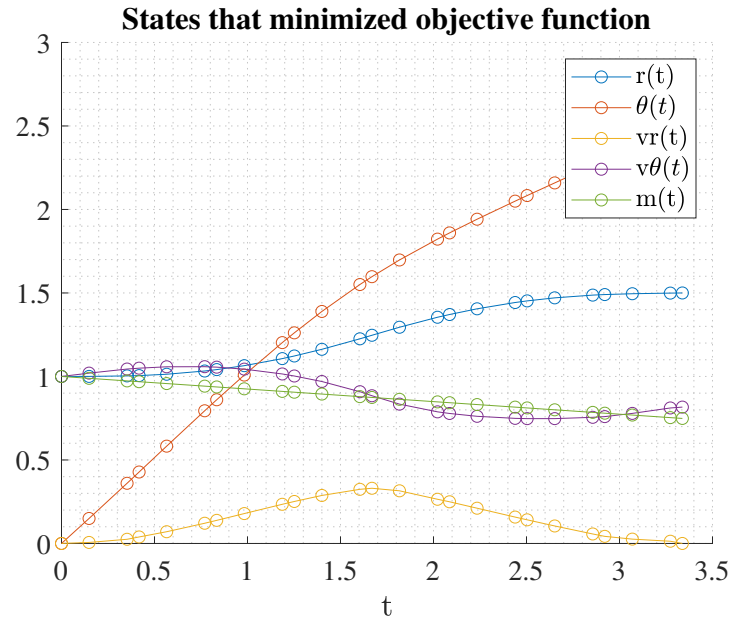


Figure 11: States for trajectory that minimized terminal time ($N : 3, K : 8$)

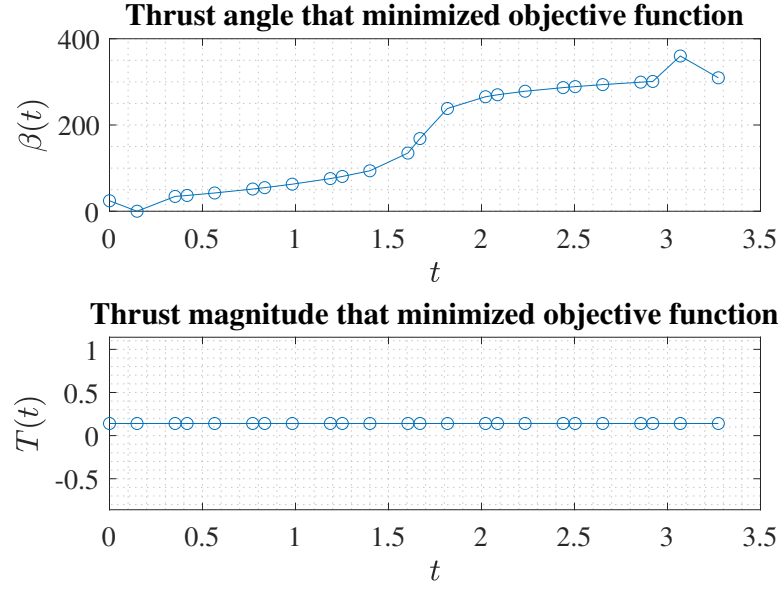


Figure 12: Control that minimized terminal time ($N : 3$, $K : 8$)

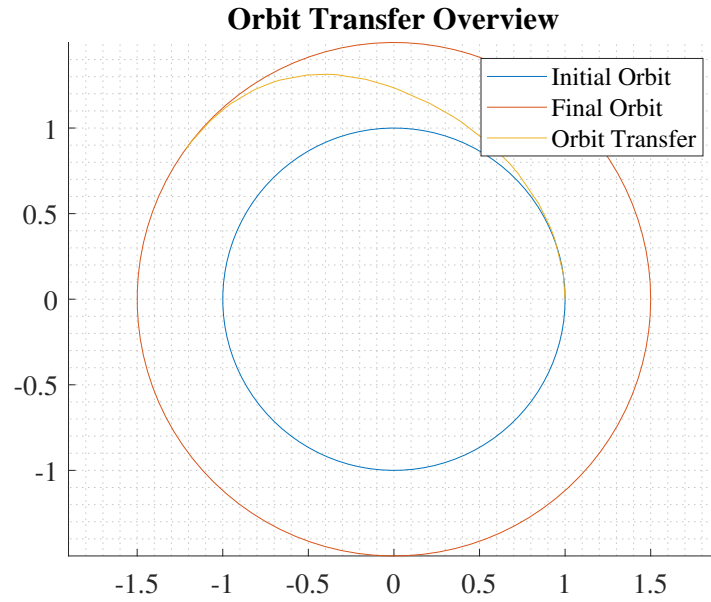


Figure 13: Trajectory from initial to final orbit ($N : 3$, $K : 8$)

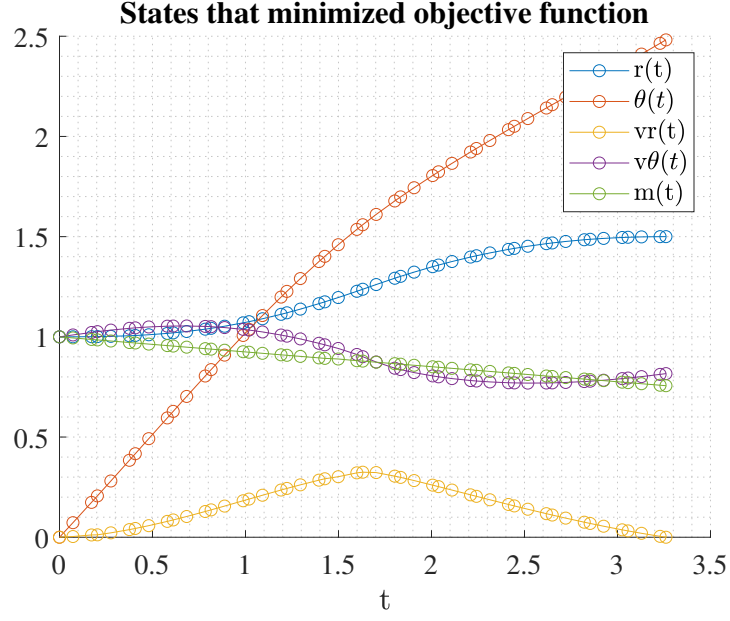


Figure 14: States for trajectory that minimized terminal time ($N : 3, K : 16$)

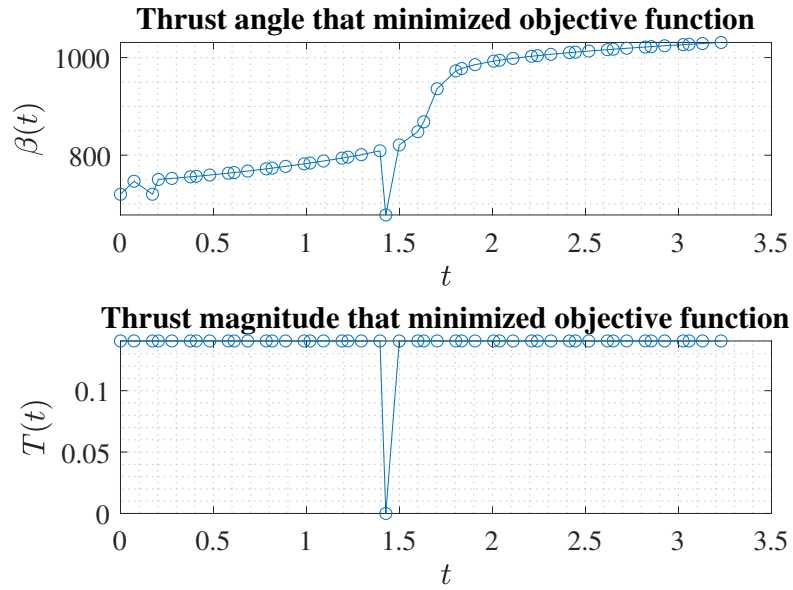


Figure 15: Control that minimized terminal time ($N : 3, K : 16$)

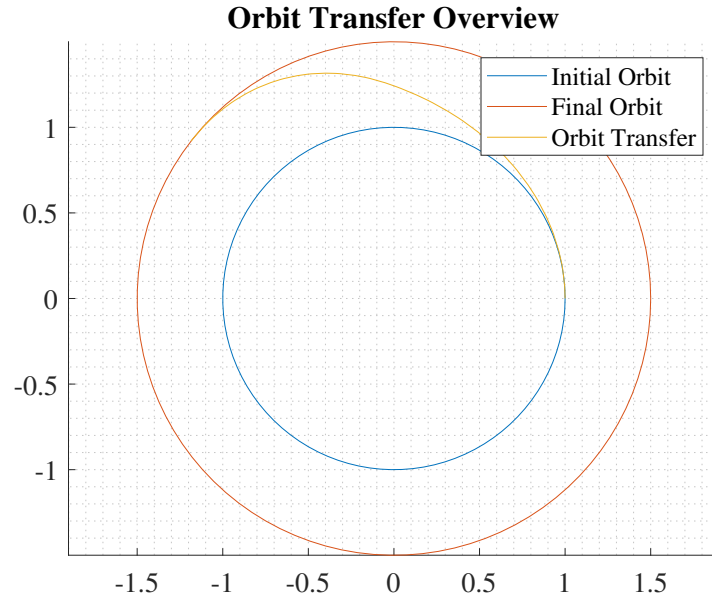


Figure 16: Trajectory from initial to final orbit ($N : 3$, $K : 16$)

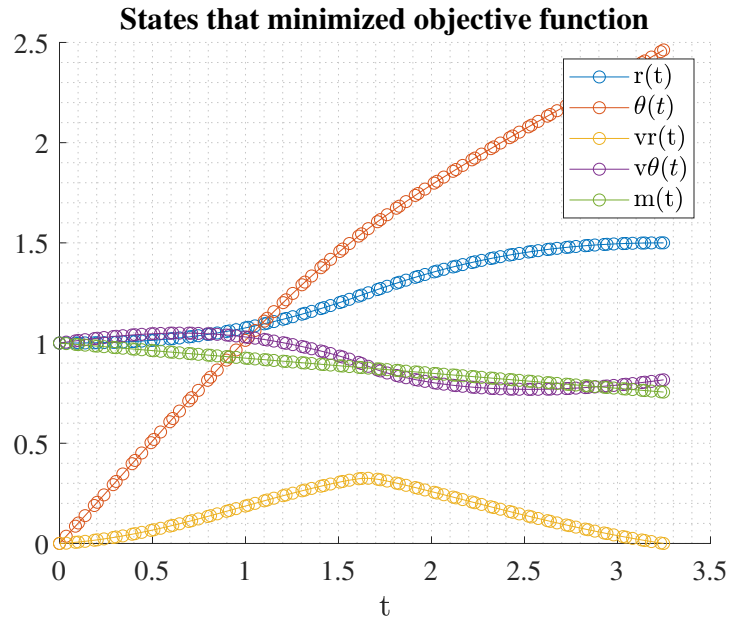


Figure 17: States for trajectory that minimized terminal time ($N : 3$, $K : 32$)

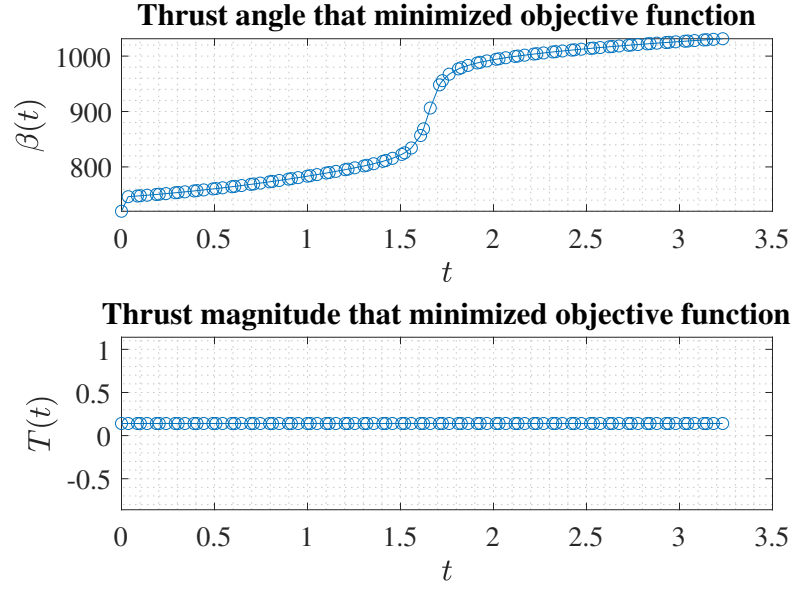


Figure 18: Control that minimized terminal time ($N : 3$, $K : 32$)

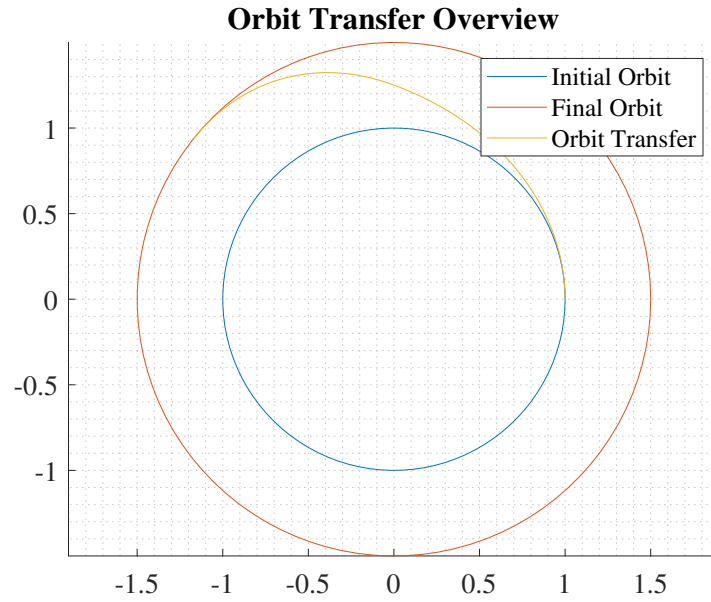


Figure 19: Trajectory from initial to final orbit ($N : 3$, $K : 32$)

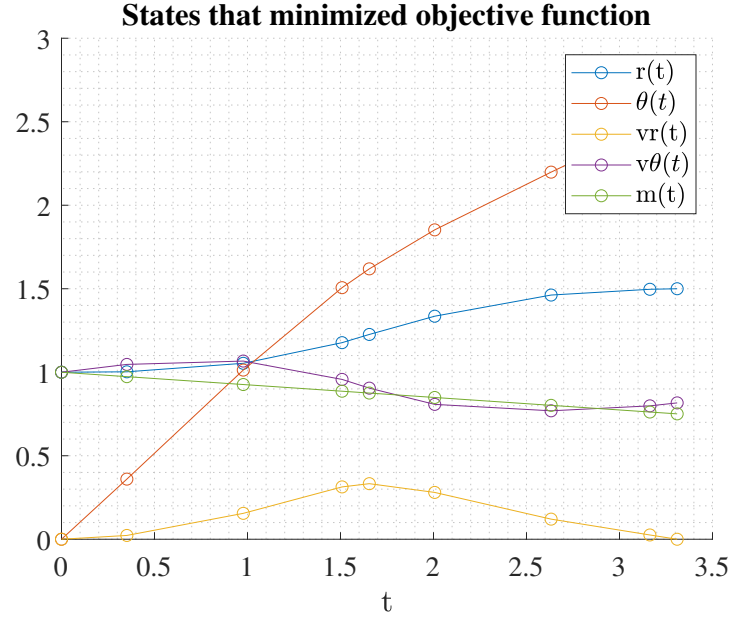


Figure 20: States for trajectory that minimized terminal time ($N : 4 , K : 2$)

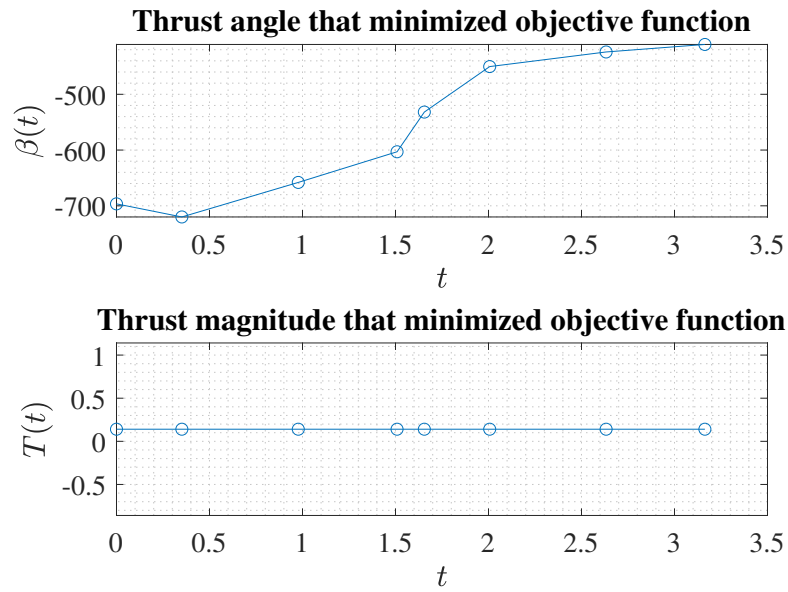


Figure 21: Control that minimized terminal time ($N : 4 , K : 2$)

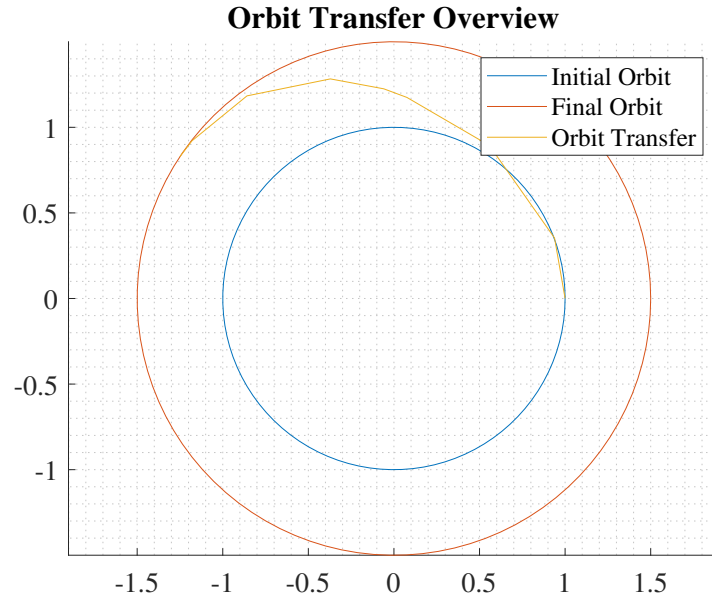


Figure 22: Trajectory from initial to final orbit ($N : 4, K : 2$)

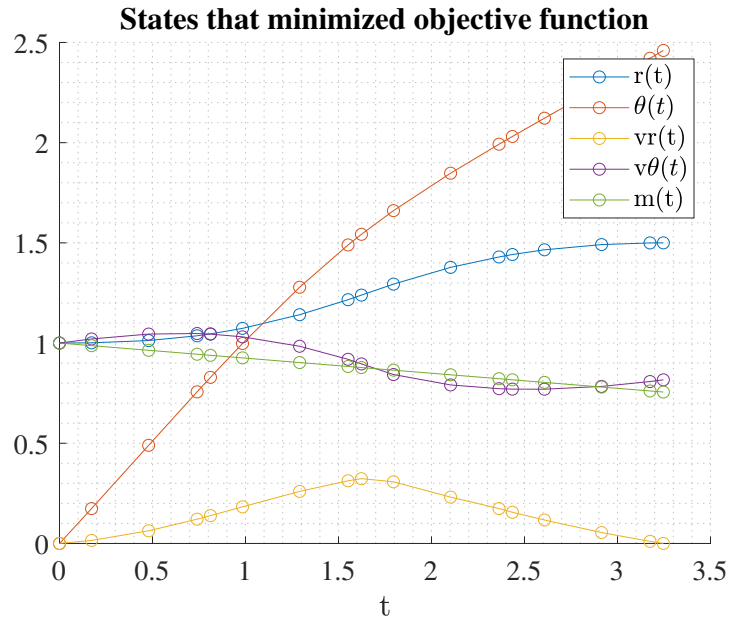


Figure 23: States for trajectory that minimized terminal time ($N : 4, K : 4$)

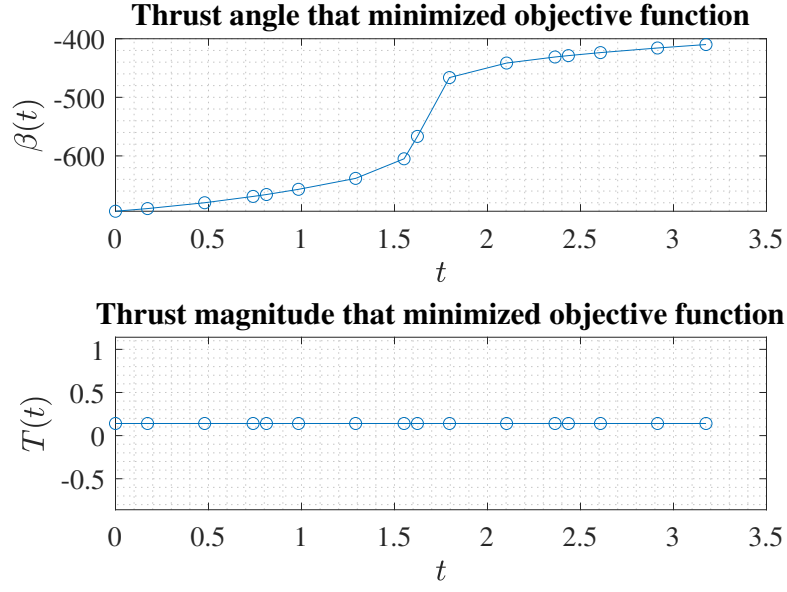


Figure 24: Control that minimized terminal time ($N : 4, K : 4$)

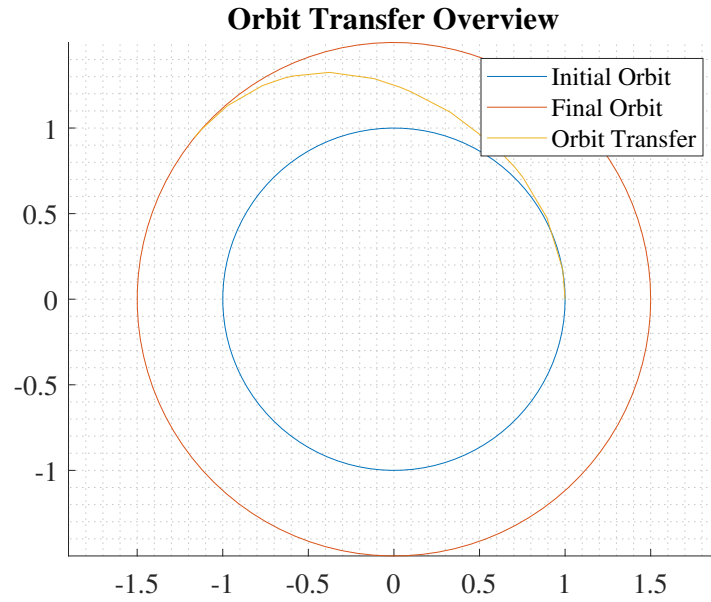


Figure 25: Trajectory from initial to final orbit ($N : 4, K : 4$)

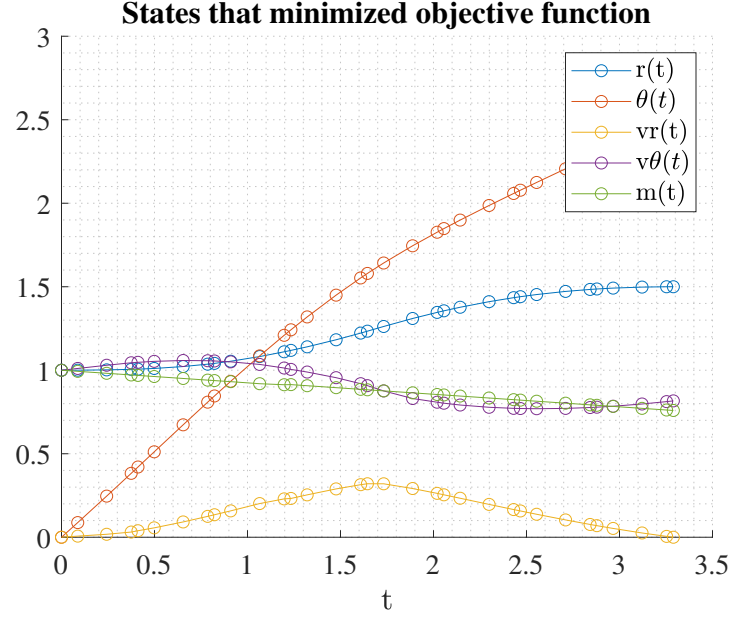


Figure 26: States for trajectory that minimized terminal time ($N : 4$, $K : 8$)

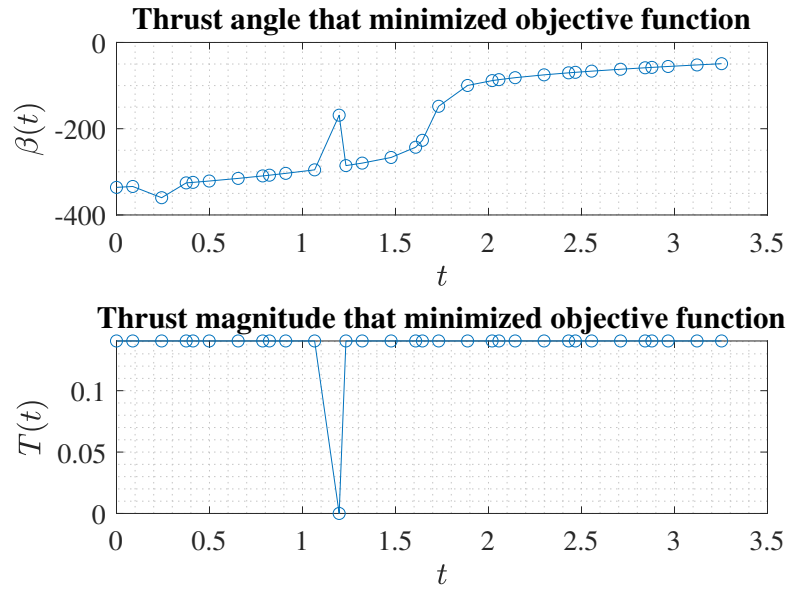


Figure 27: Control that minimized terminal time ($N : 4$, $K : 8$)

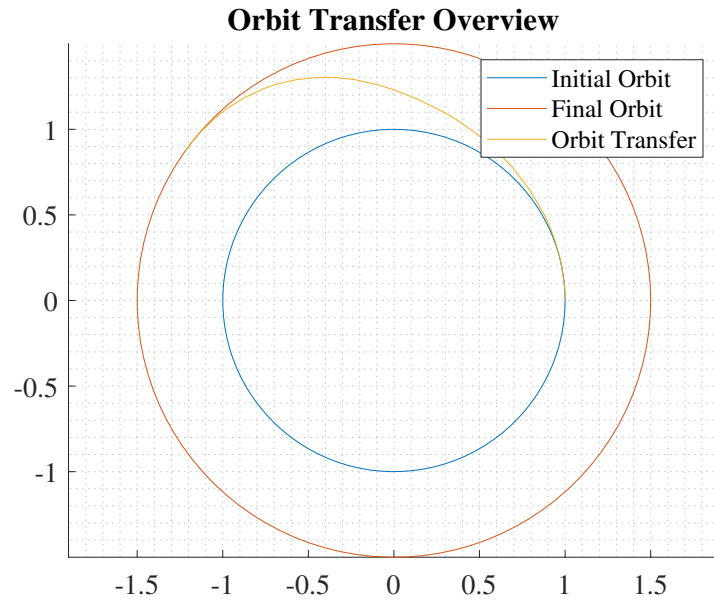


Figure 28: Trajectory from initial to final orbit ($N : 4, K : 8$)

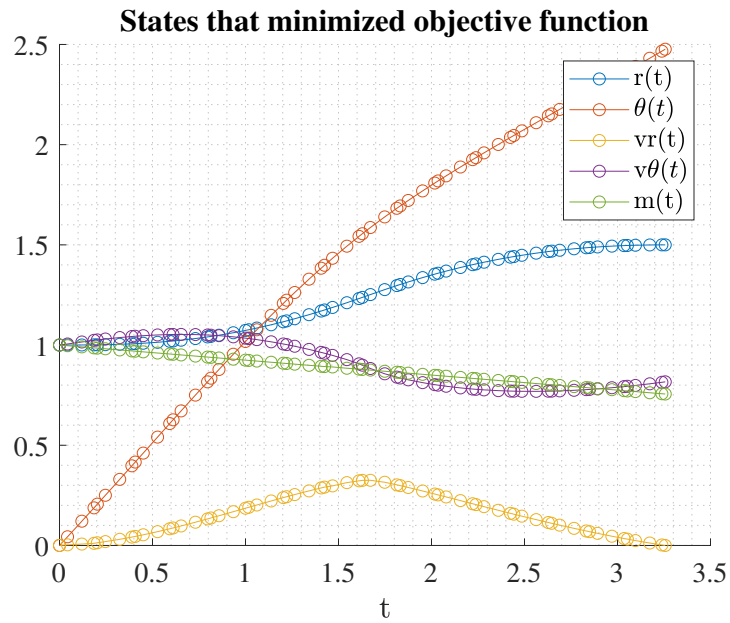


Figure 29: States for trajectory that minimized terminal time ($N : 4, K : 16$)

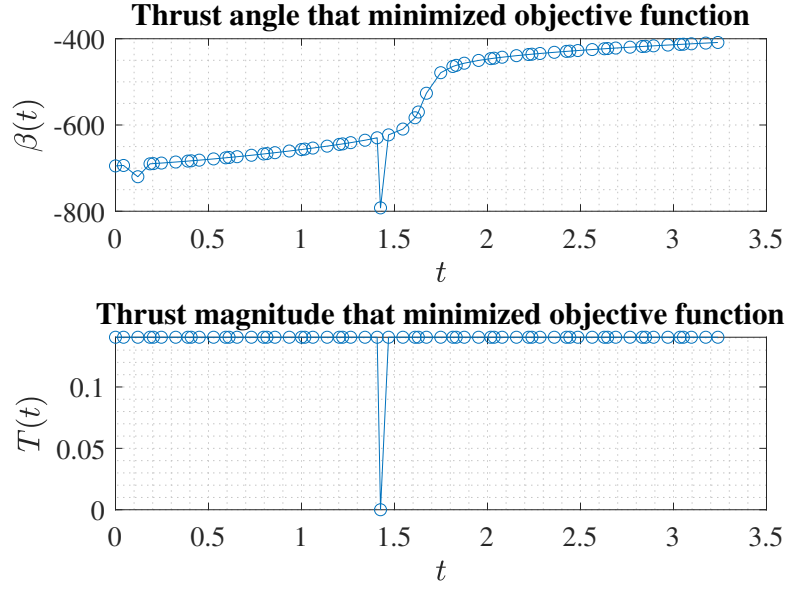


Figure 30: Control that minimized terminal time ($N : 4$, $K : 16$)

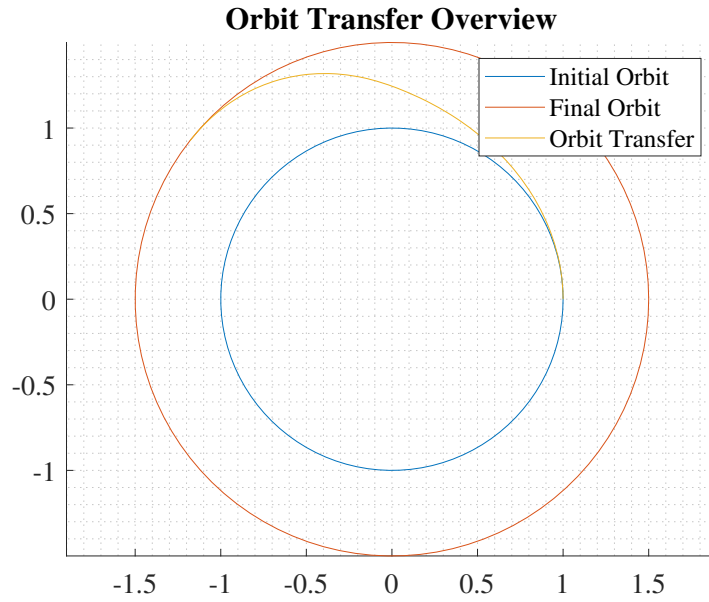


Figure 31: Trajectory from initial to final orbit ($N : 4$, $K : 16$)

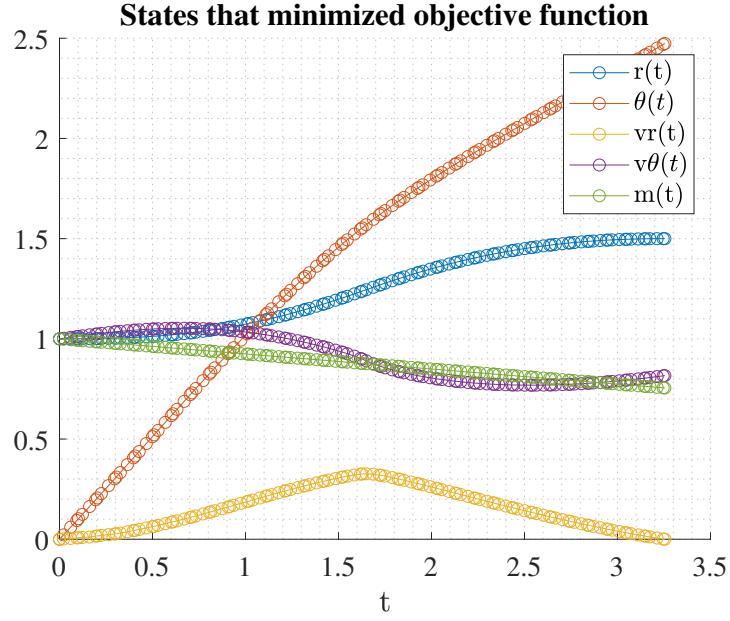


Figure 32: States for trajectory that minimized terminal time ($N : 4$, $K : 32$)

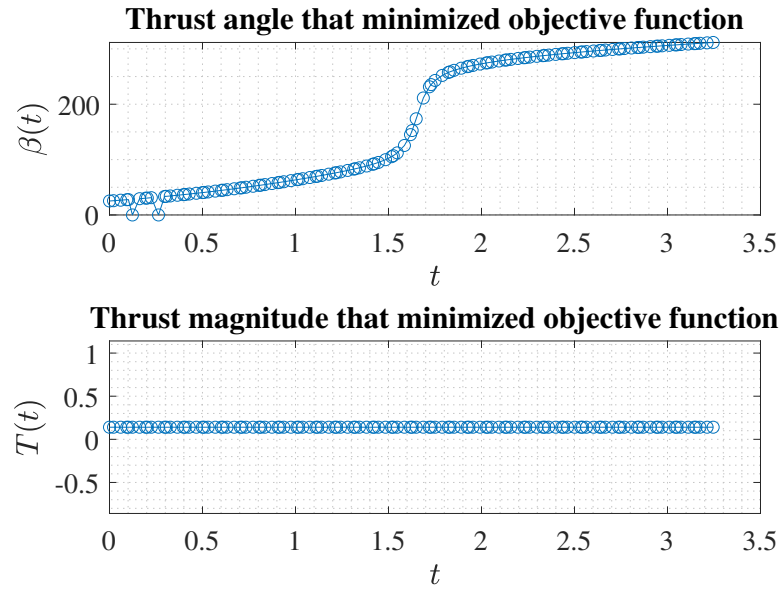


Figure 33: Control that minimized terminal time ($N : 4$, $K : 32$)

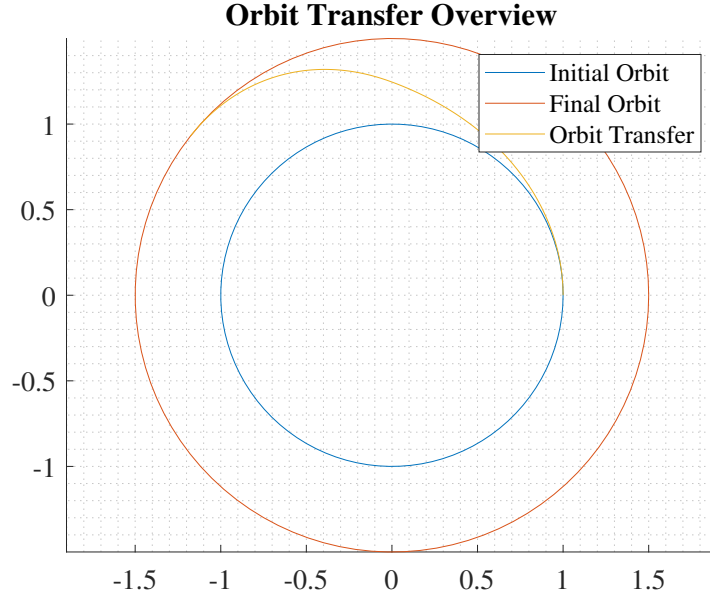


Figure 34: Trajectory from initial to final orbit ($N : 4$, $K : 32$)

Degree	Intervals	Iterations	CPU Time	tf	mf	Solved Status
3	2	35	0.185	3.2444	0.75569	0
3	4	66	0.427	3.2457	0.75559	0
3	8	155	0.623	3.3381	0.74864	0
3	16	110	0.499	3.2605	0.75619	0
3	32	187	0.893	3.2479	0.75543	0
4	2	80	0.337	3.31	0.75075	0
4	4	153	0.588	3.2466	0.75552	0
4	8	126	0.54	3.2896	0.75912	0
4	16	144	0.659	3.2567	0.75572	0
4	32	174	0.952	3.2546	0.75492	1

Table 1: Results for minimizing t_f with unconstrained control

4.2 Maximize Terminal Mass with Unconstrained Control

Degree	Intervals	Iterations	CPU Time	tf	mf	Solved Status
3	2	621	1.96	10.5501	0.90956	0
3	4	4271	13.906	21.0208	0.90633	-2
3	8	416	1.477	8.8194	0.90719	0
3	16	451	1.759	7.3181	0.9072	0
3	32	204	0.999	11.8567	0.9072	0
4	2	214	0.758	11.5211	0.90659	-2
4	4	1720	5.772	7.6934	0.90717	0
4	8	231	0.903	9.4231	0.90718	0
4	16	138	0.657	10.3297	0.90721	0
4	32	131	0.749	8.2286	0.90719	1

Table 2: Results for maximizing m_f with unconstrained control

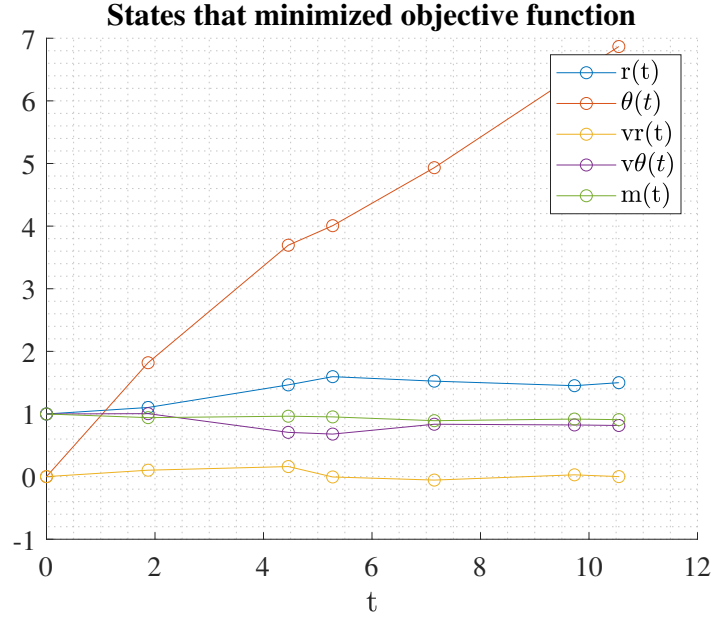


Figure 35: States for trajectory that maximized terminal mass ($N : 3, K : 2$)

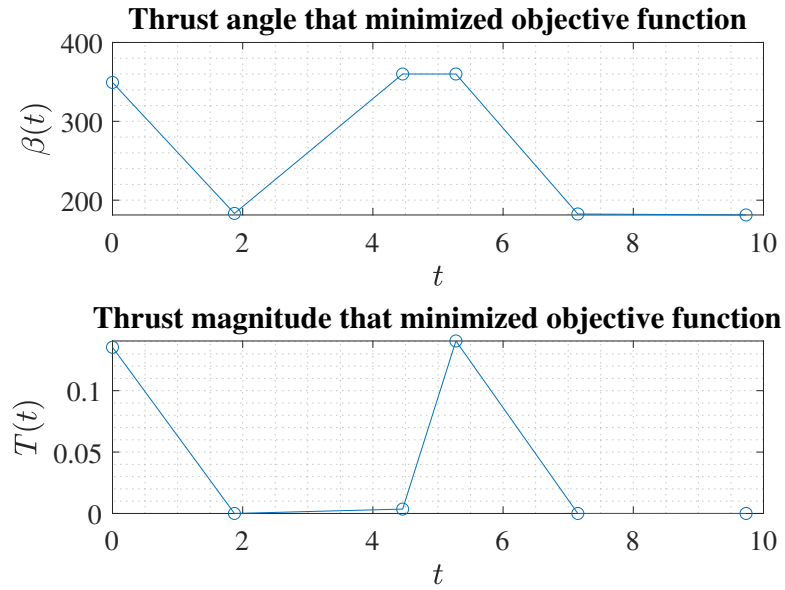


Figure 36: Control that maximized terminal mass ($N : 3, K : 2$)

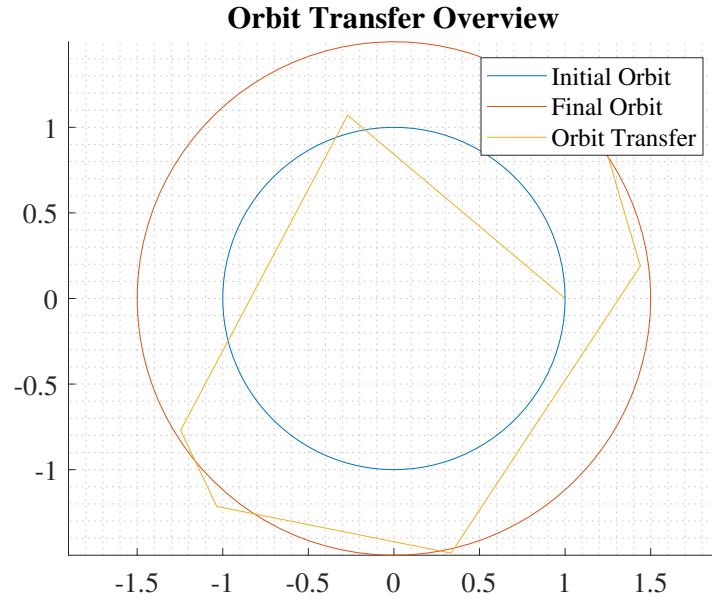


Figure 37: Trajectory from initial to final orbit ($N : 3, K : 2$)

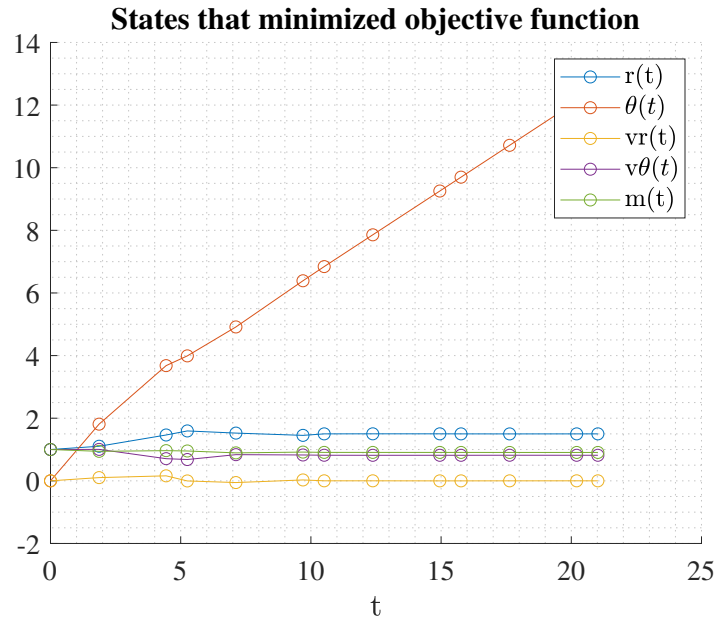


Figure 38: States for trajectory that maximized terminal mass ($N : 3, K : 4$)

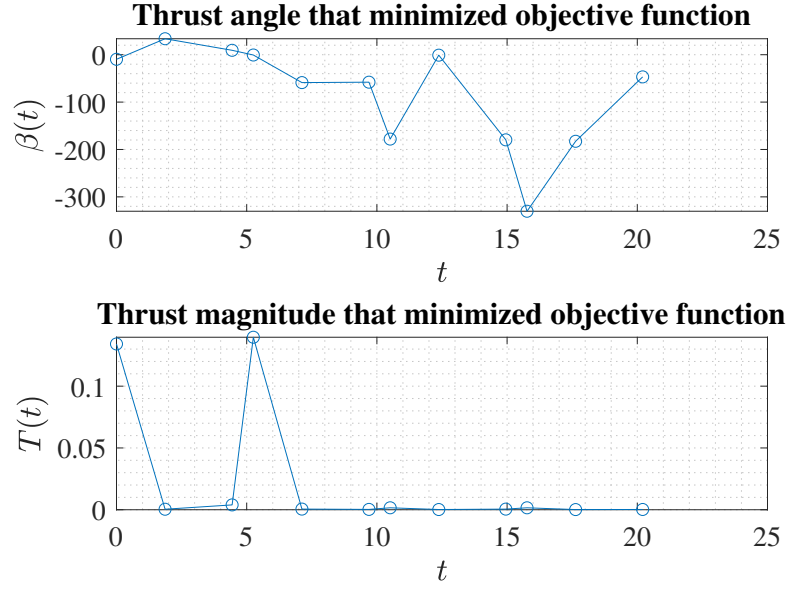


Figure 39: Control that maximized terminal mass ($N : 3$, $K : 4$)

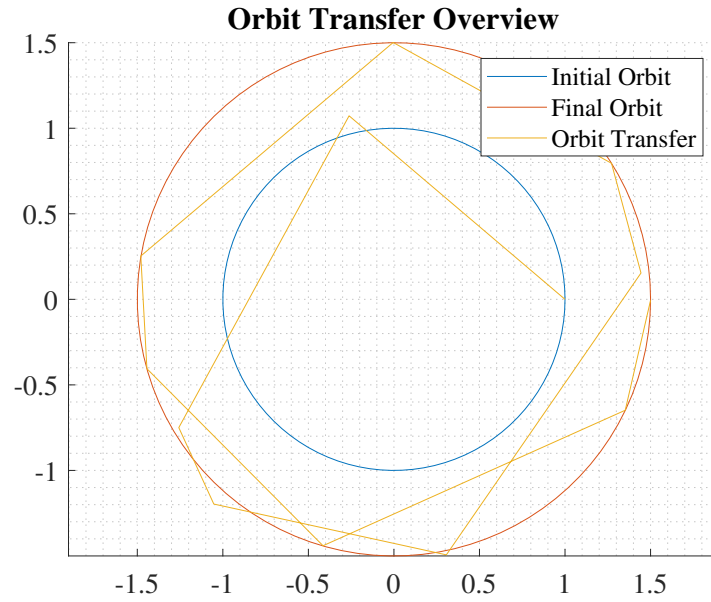


Figure 40: Trajectory from initial to final orbit ($N : 3$, $K : 4$)

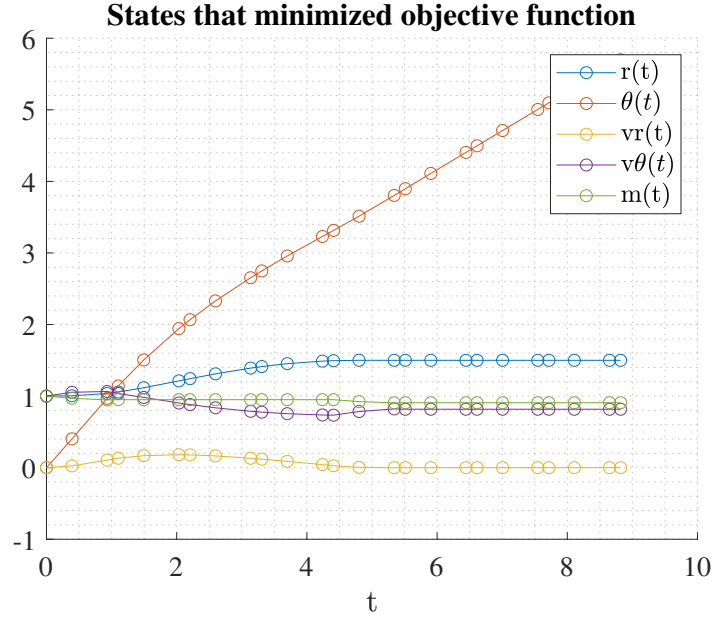


Figure 41: States for trajectory that maximized terminal mass ($N : 3, K : 8$)

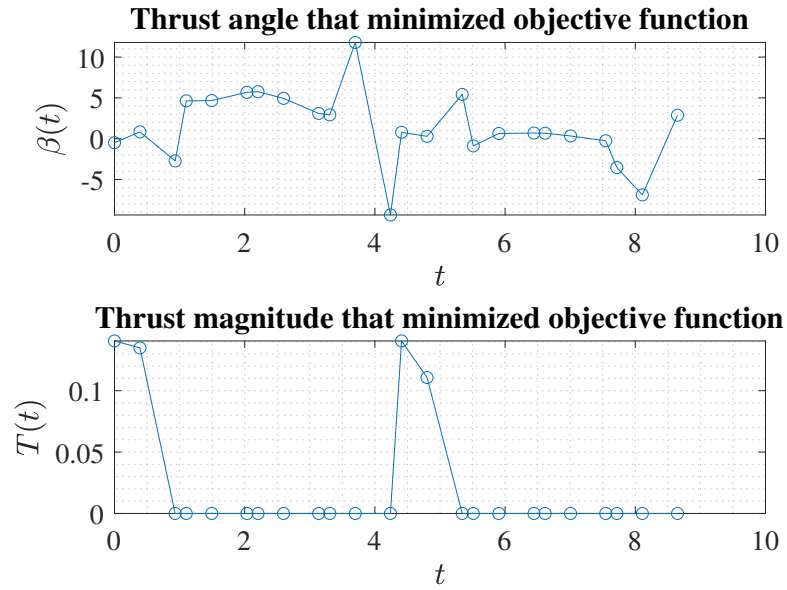


Figure 42: Control that maximized terminal mass ($N : 3, K : 8$)

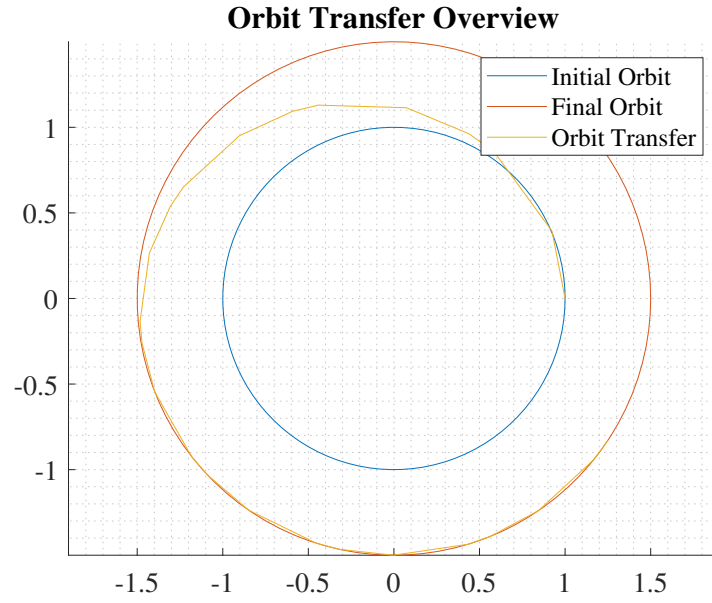


Figure 43: Trajectory from initial to final orbit ($N : 3, K : 8$)

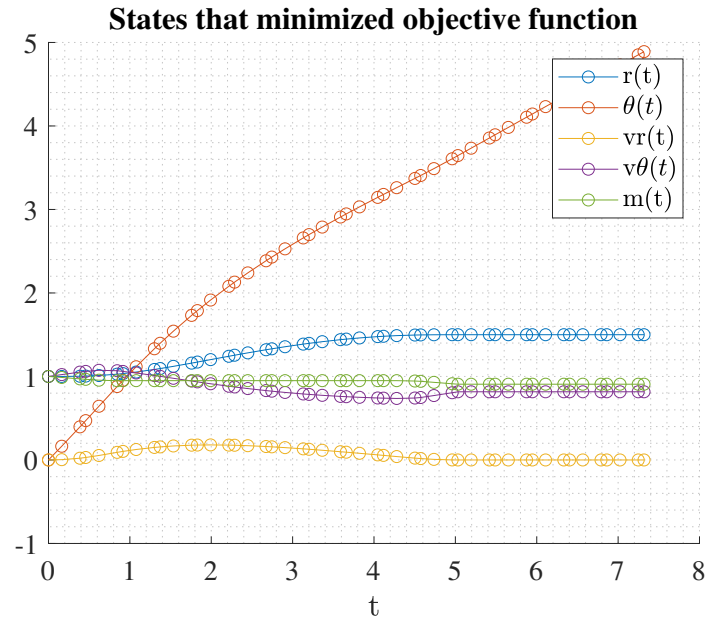


Figure 44: States for trajectory that maximized terminal mass ($N : 3, K : 16$)

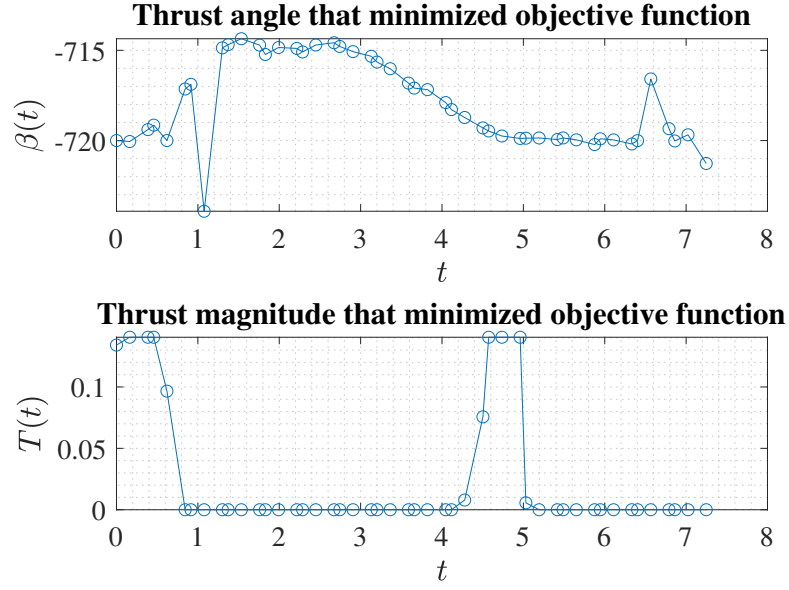


Figure 45: Control that maximized terminal mass ($N : 3$, $K : 16$)

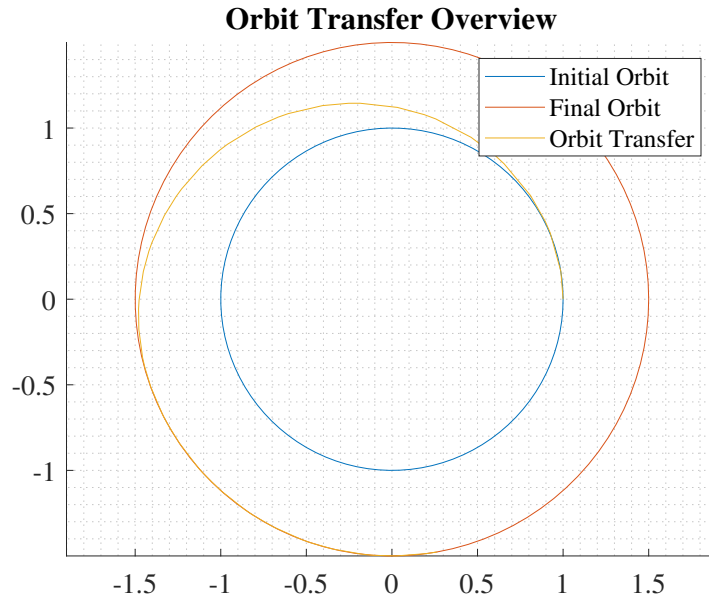


Figure 46: Trajectory from initial to final orbit ($N : 3$, $K : 16$)

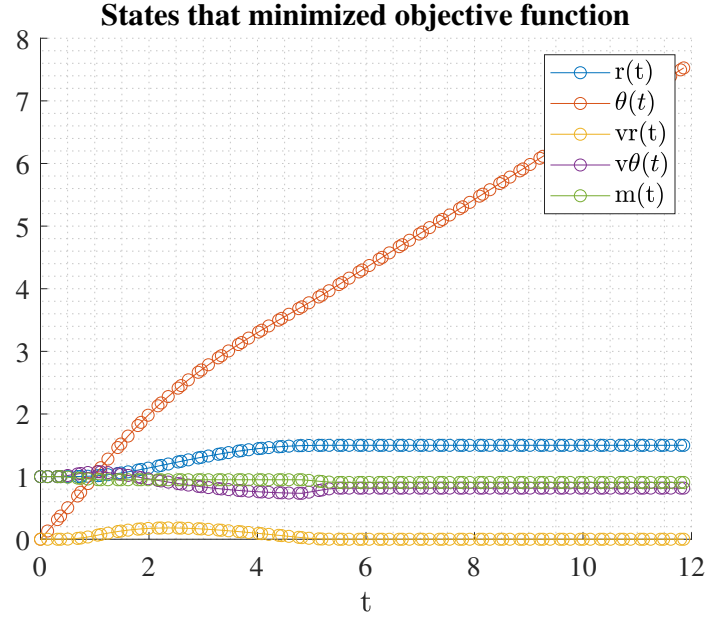


Figure 47: States for trajectory that maximized terminal mass ($N : 3$, $K : 32$)

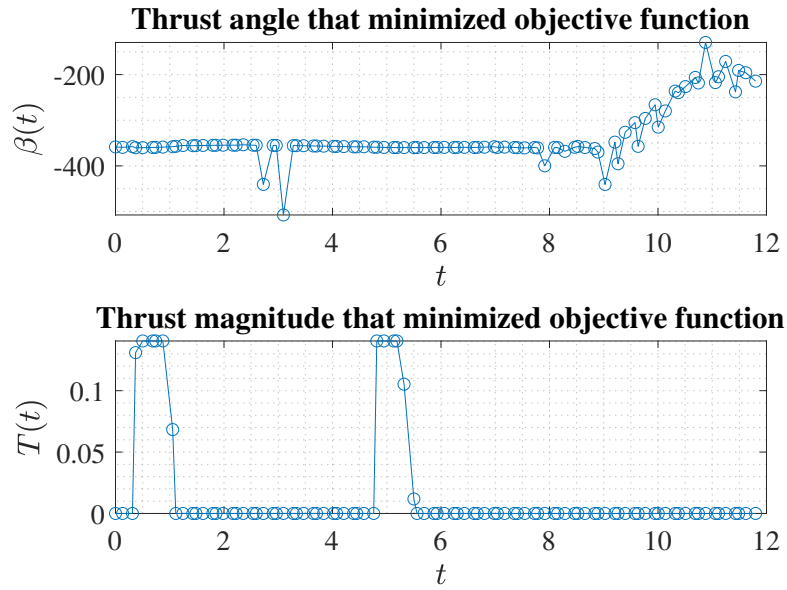


Figure 48: Control that maximized terminal mass ($N : 3$, $K : 32$)

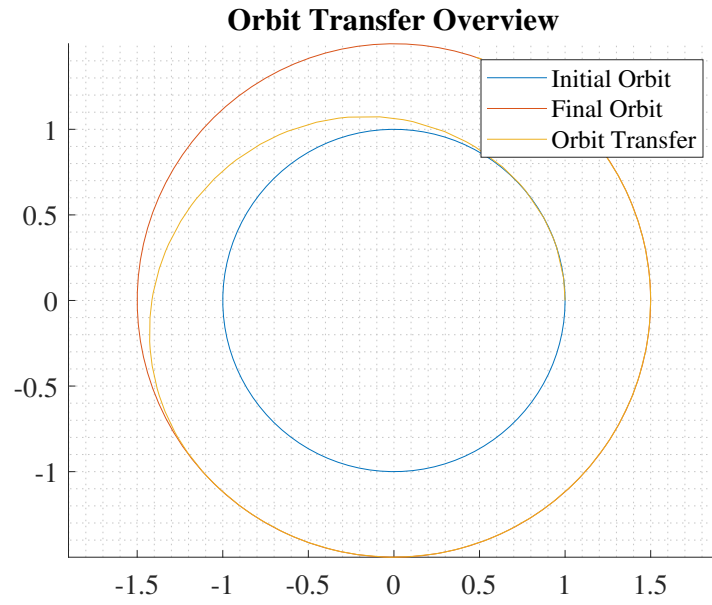


Figure 49: Trajectory from initial to final orbit ($N : 3$, $K : 32$)

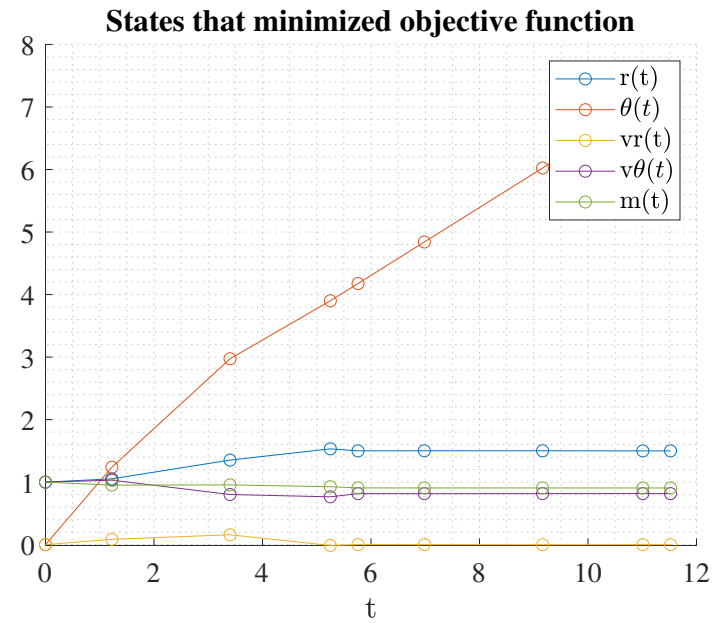


Figure 50: States for trajectory that maximized terminal mass ($N : 4$, $K : 2$)

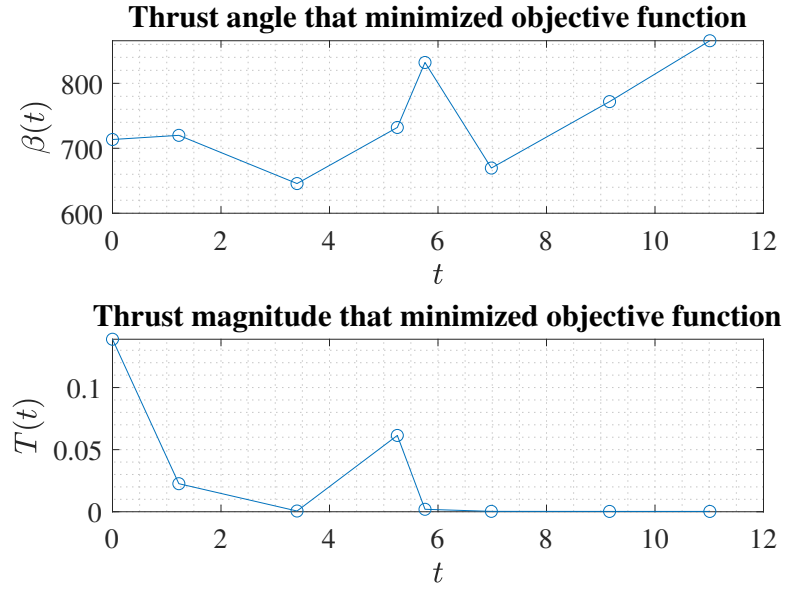


Figure 51: Control that maximized terminal mass ($N : 4 , K : 2$)

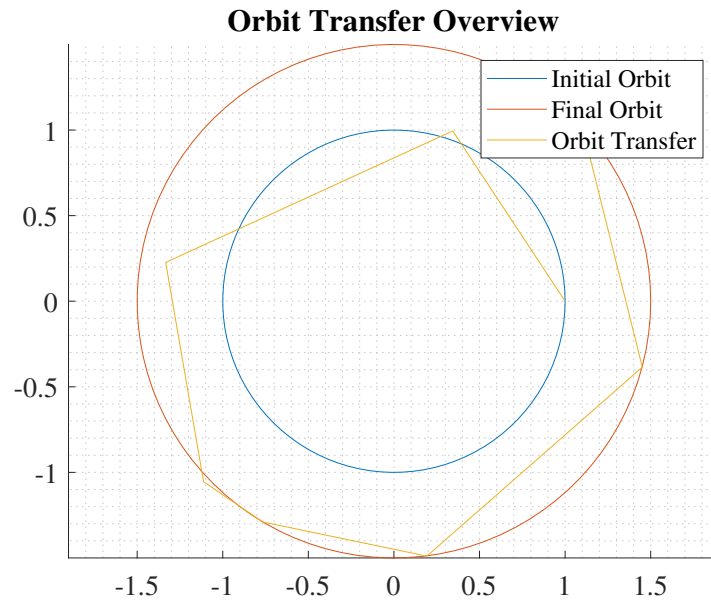


Figure 52: Trajectory from initial to final orbit ($N : 4 , K : 2$)

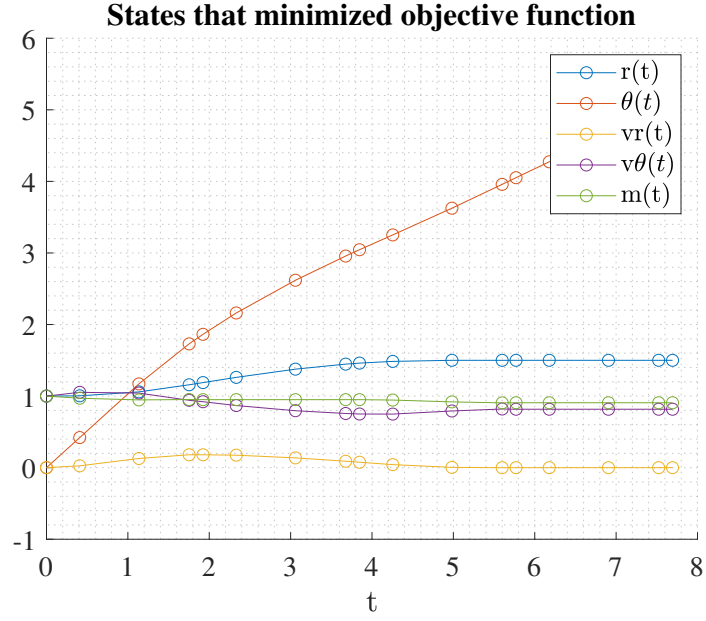


Figure 53: States for trajectory that maximized terminal mass ($N : 4, K : 4$)

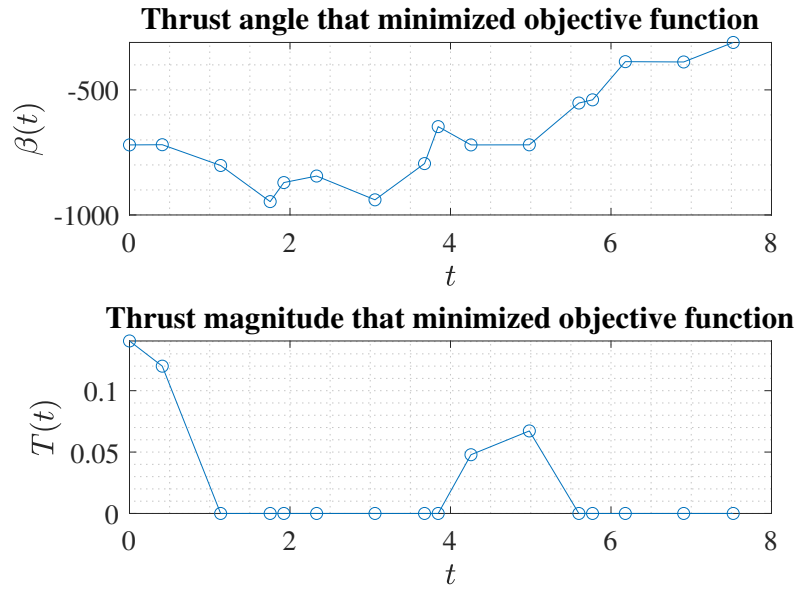


Figure 54: Control that maximized terminal mass ($N : 4, K : 4$)

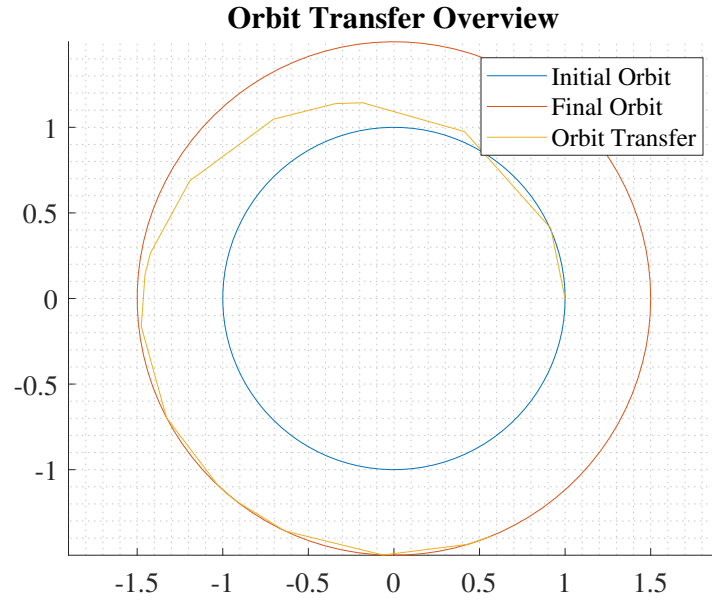


Figure 55: Trajectory from initial to final orbit ($N : 4, K : 4$)

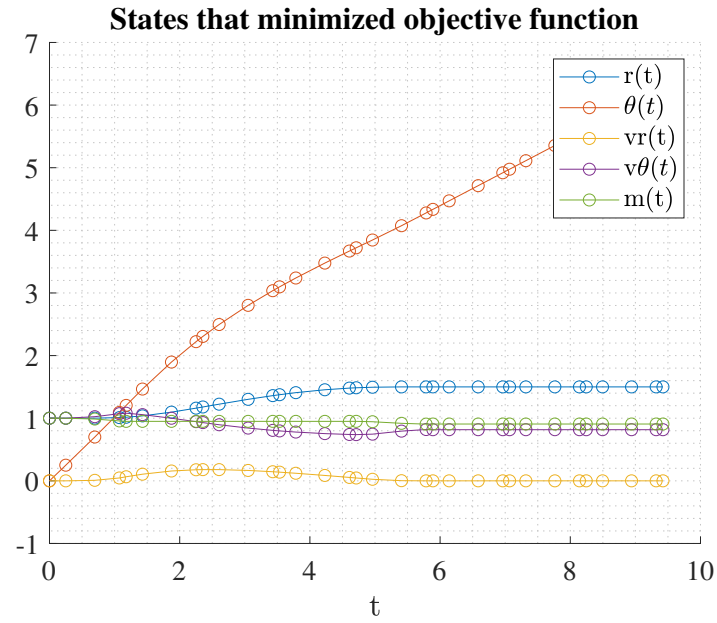


Figure 56: States for trajectory that maximized terminal mass ($N : 4, K : 8$)

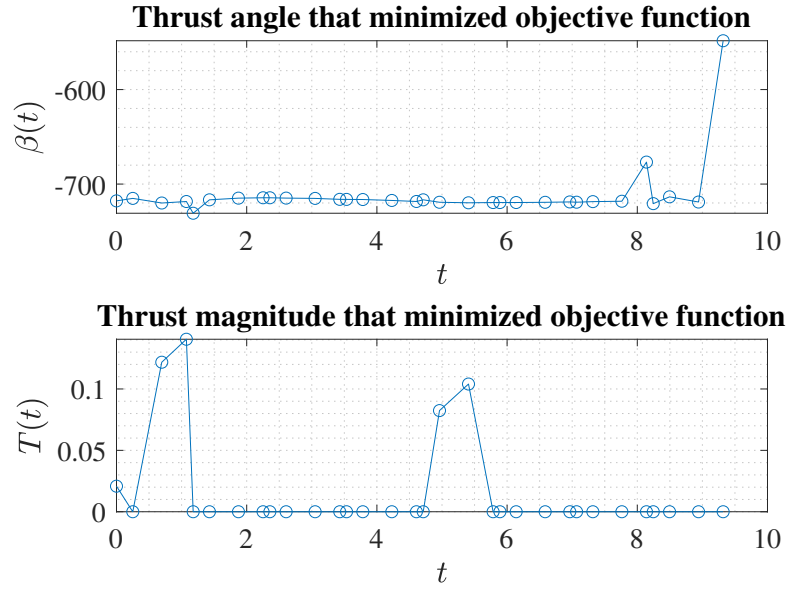


Figure 57: Control that maximized terminal mass ($N : 4$, $K : 8$)

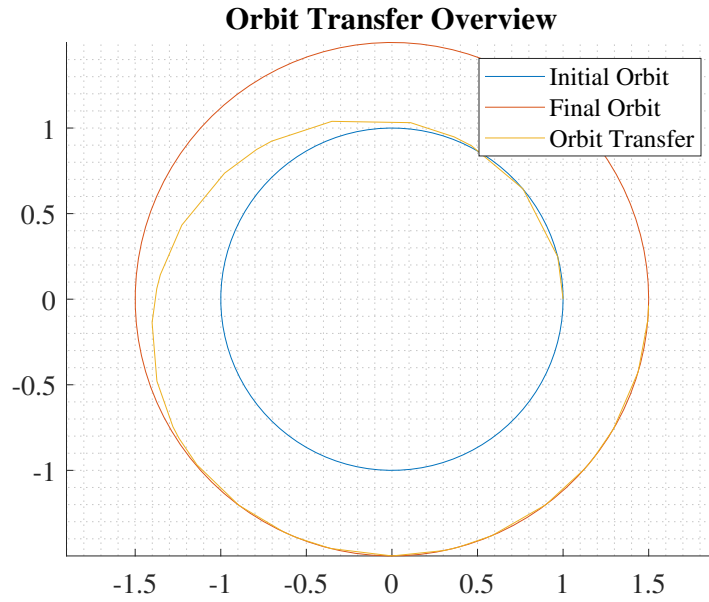


Figure 58: Trajectory from initial to final orbit ($N : 4$, $K : 8$)

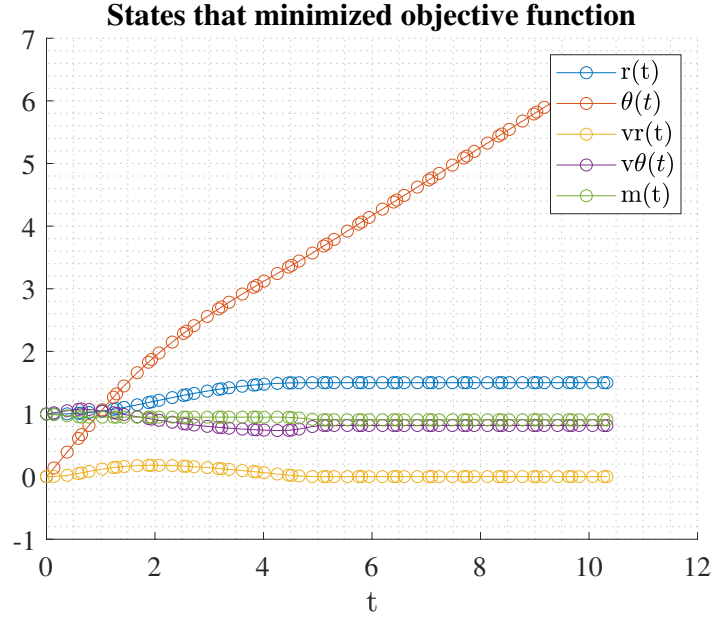


Figure 59: States for trajectory that maximized terminal mass ($N : 4, K : 16$)

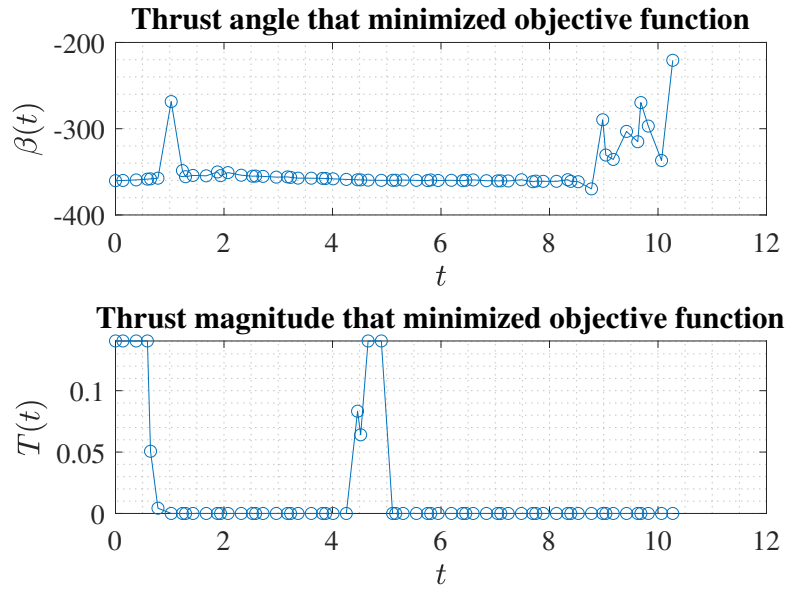


Figure 60: Control that maximized terminal mass ($N : 4, K : 16$)

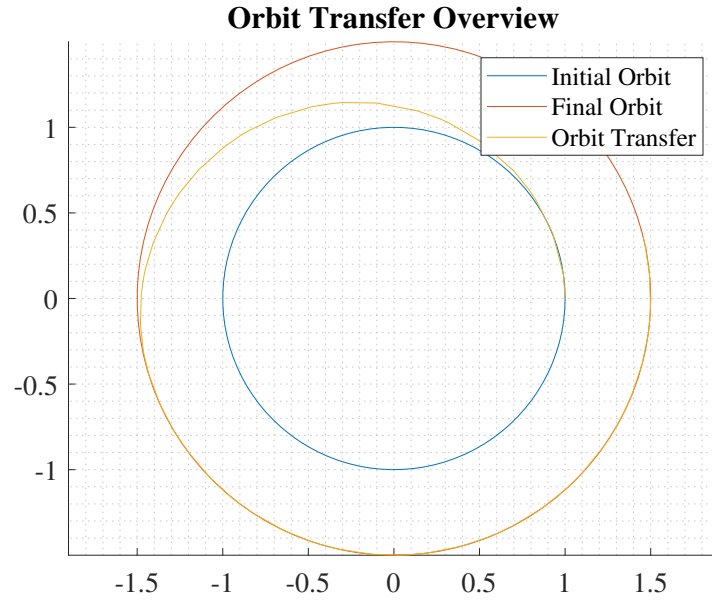


Figure 61: Trajectory from initial to final orbit ($N : 4$, $K : 16$)

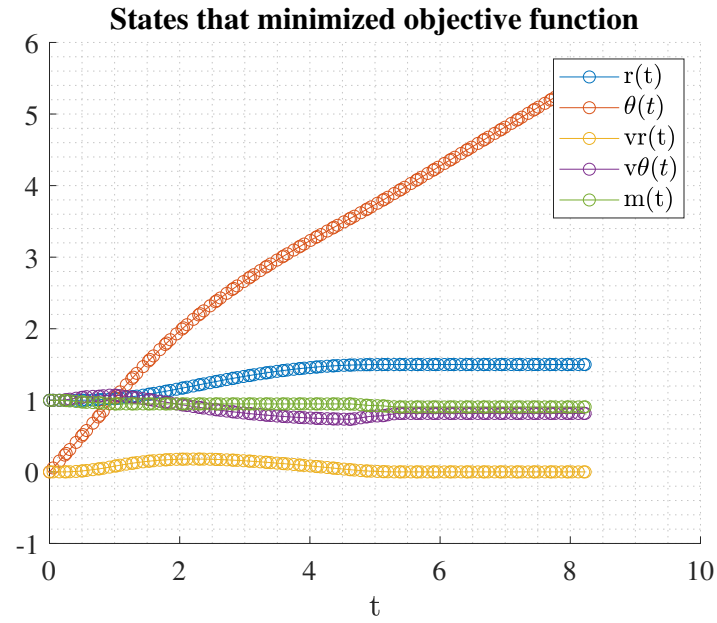


Figure 62: States for trajectory that maximized terminal mass ($N : 4$, $K : 32$)

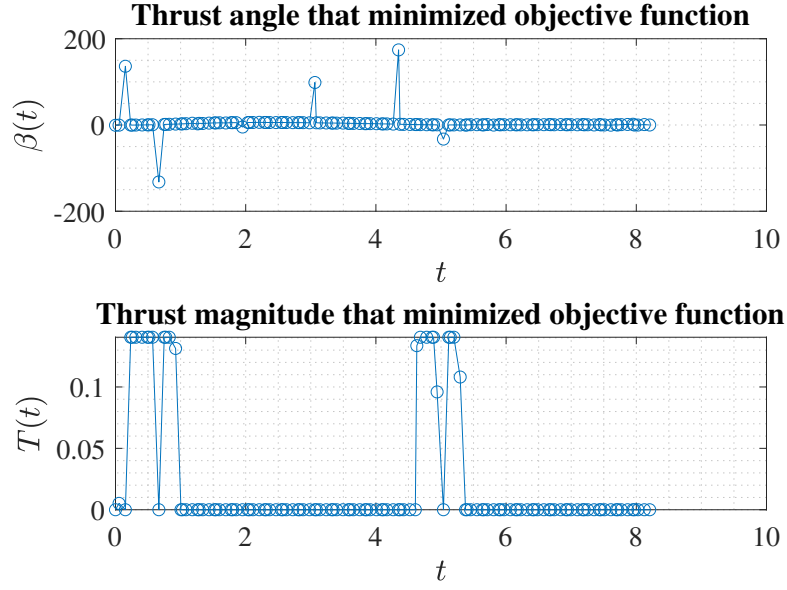


Figure 63: Control that maximized terminal mass ($N : 4, K : 32$)

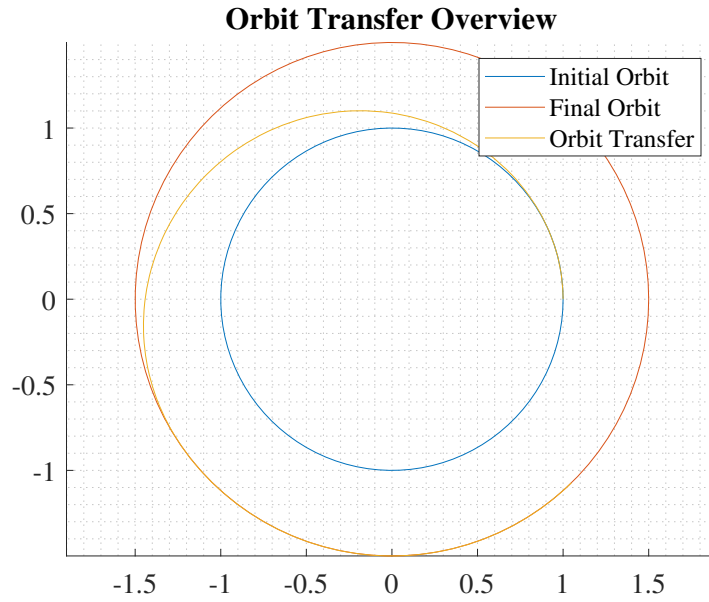


Figure 64: Trajectory from initial to final orbit ($N : 4, K : 32$)

4.3 Minimize Terminal Time with Constrained Control

Degree	Intervals	Iterations	CPU Time	tf	mf	Solved Status
3	2	47	0.225	3.2444	0.75569	0
3	4	69	0.446	3.2457	0.75559	0
3	8	77	0.369	3.2577	0.7581	0
3	16	119	0.537	3.247	0.7555	0
3	32	143	0.738	3.2623	0.75775	0
4	2	69	0.314	3.2456	0.7556	0
4	4	81	0.363	3.27	0.75761	0
4	8	102	0.464	3.2584	0.75655	0
4	16	143	0.677	3.247	0.7555	0
4	32	151	0.889	3.2482	0.75589	1

Table 3: Results for minimizing t_f with constrained control

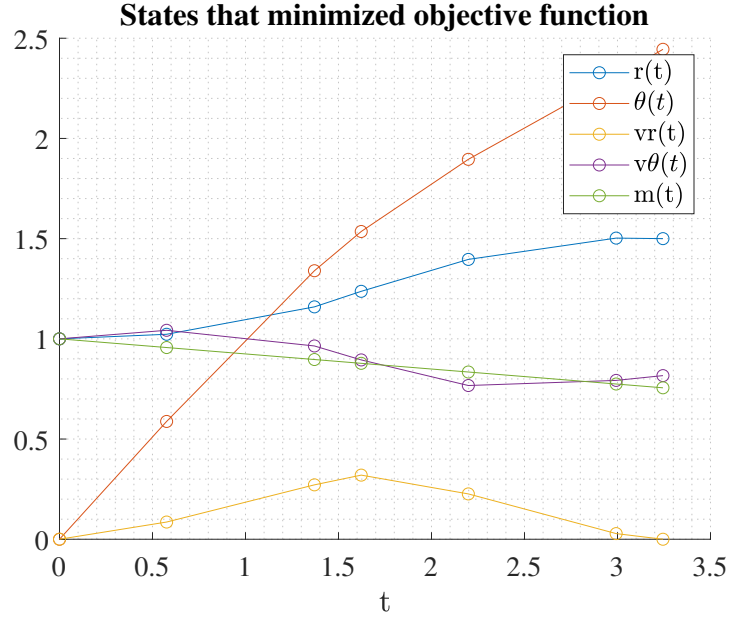


Figure 65: States for trajectory that minimized terminal time ($N : 3, K : 2$)

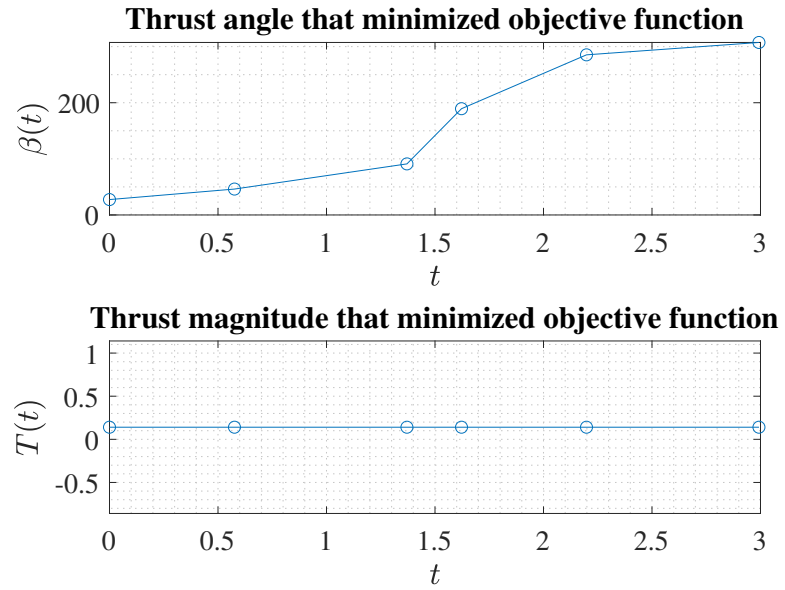


Figure 66: Control that minimized terminal time ($N : 3, K : 2$)

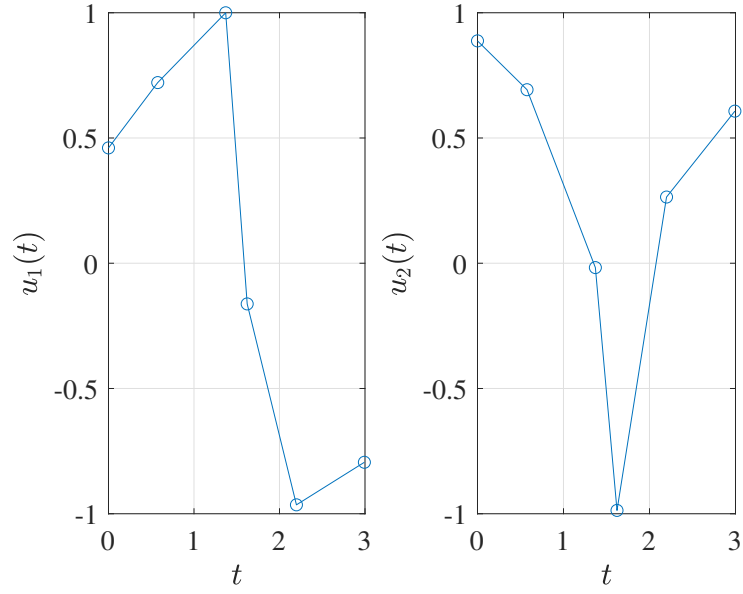


Figure 67: Path constrained control that minimized terminal time ($N : 3, K : 2$)

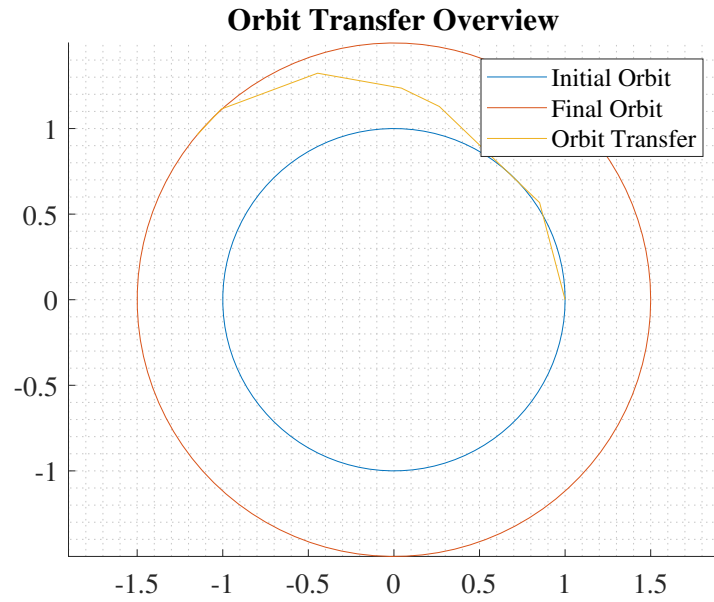


Figure 68: Trajectory from initial to final orbit ($N : 3, K : 2$)

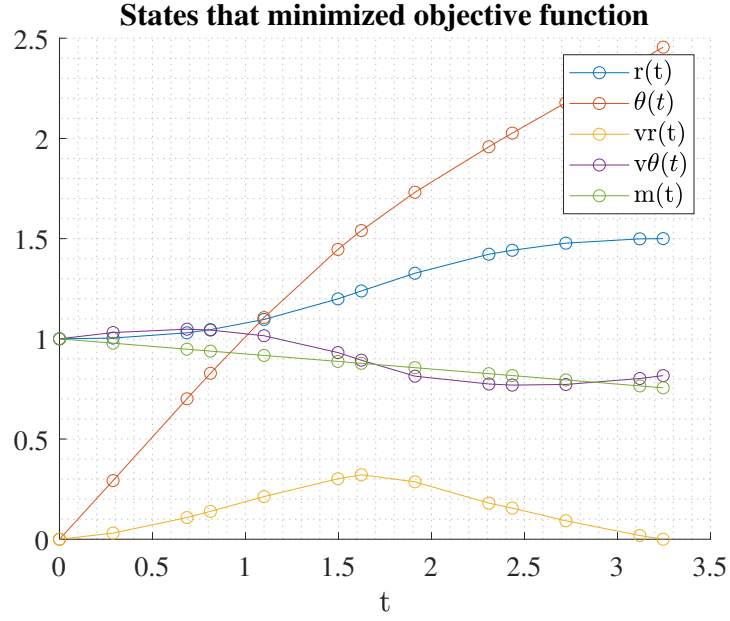


Figure 69: States for trajectory that minimized terminal time ($N : 3$, $K : 4$)

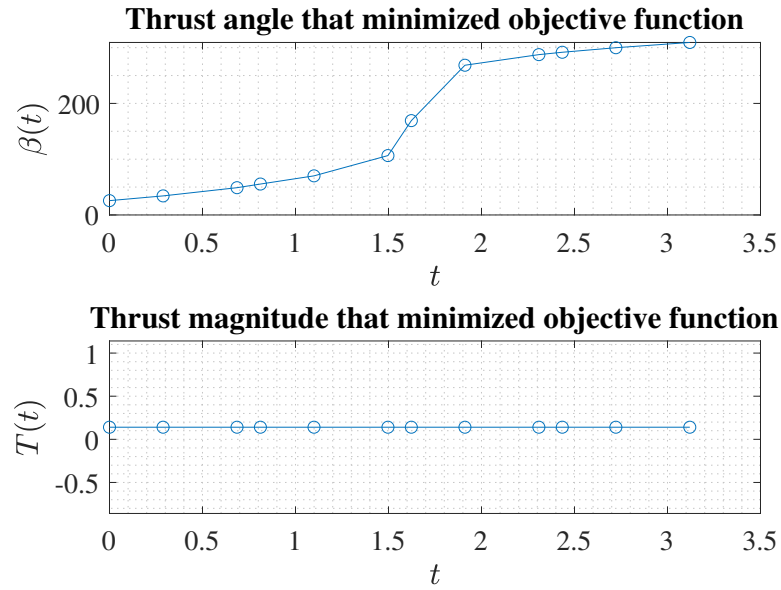


Figure 70: Control that minimized terminal time ($N : 3$, $K : 4$)

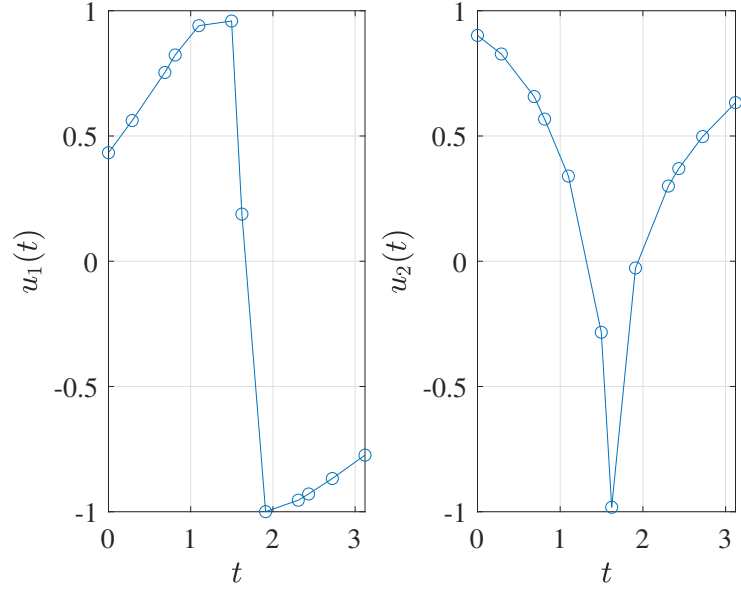


Figure 71: Path constrained control that minimized terminal time ($N : 3, K : 4$)

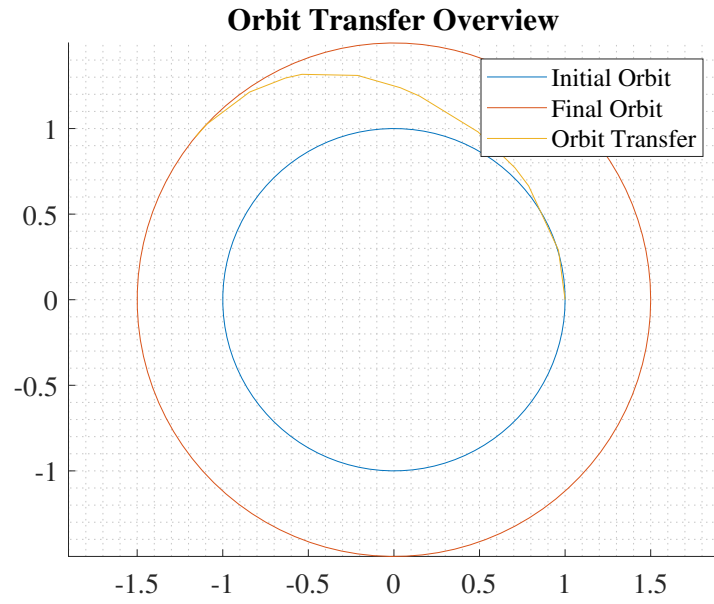


Figure 72: Trajectory from initial to final orbit ($N : 3, K : 4$)

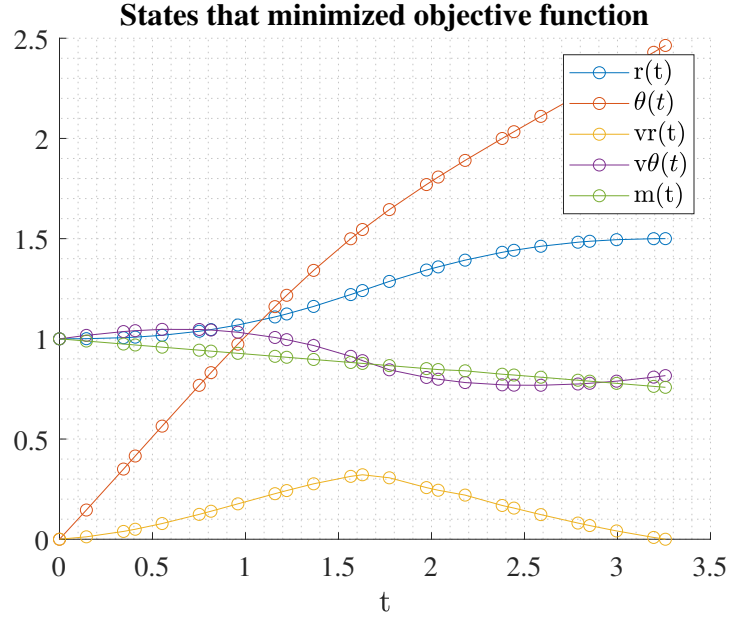


Figure 73: States for trajectory that minimized terminal time ($N : 3, K : 8$)

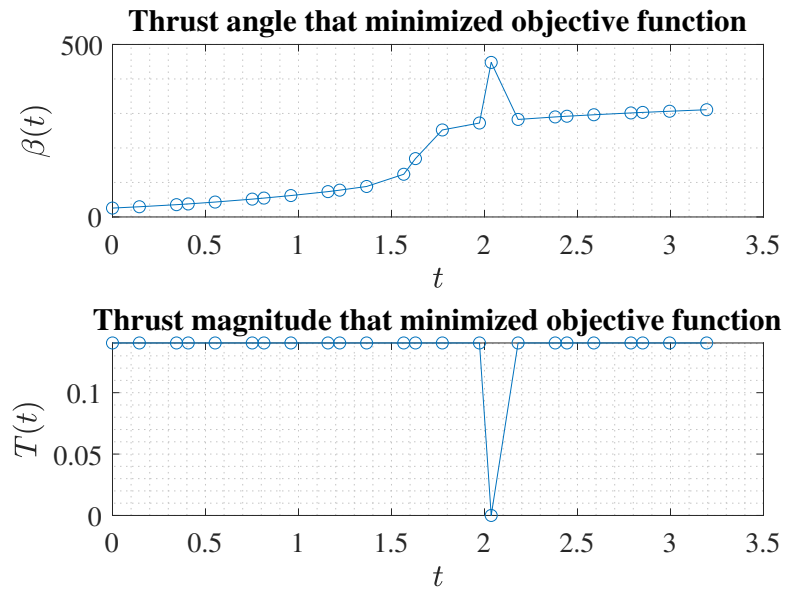


Figure 74: Control that minimized terminal time ($N : 3, K : 8$)

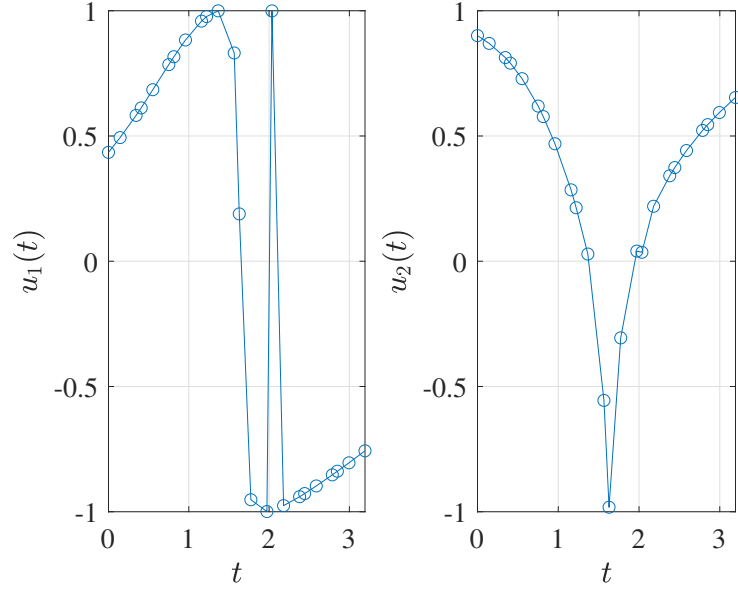


Figure 75: Path constrained control that minimized terminal time ($N : 3, K : 8$)

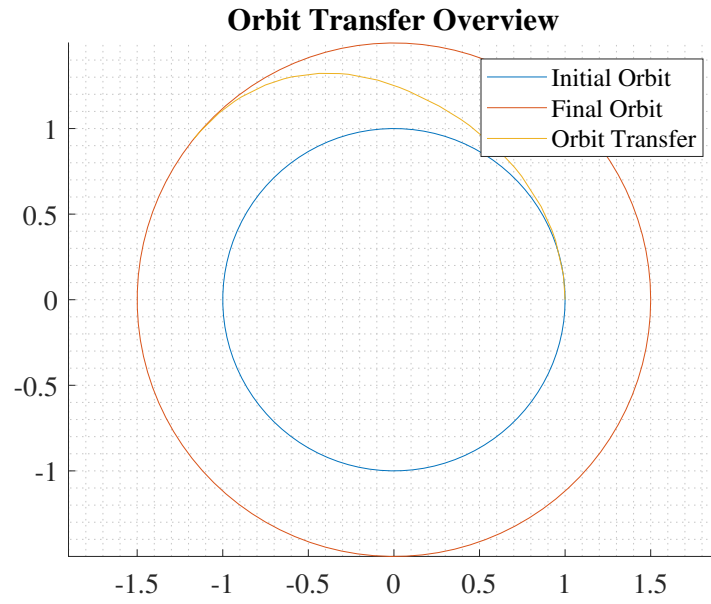


Figure 76: Trajectory from initial to final orbit ($N : 3, K : 8$)

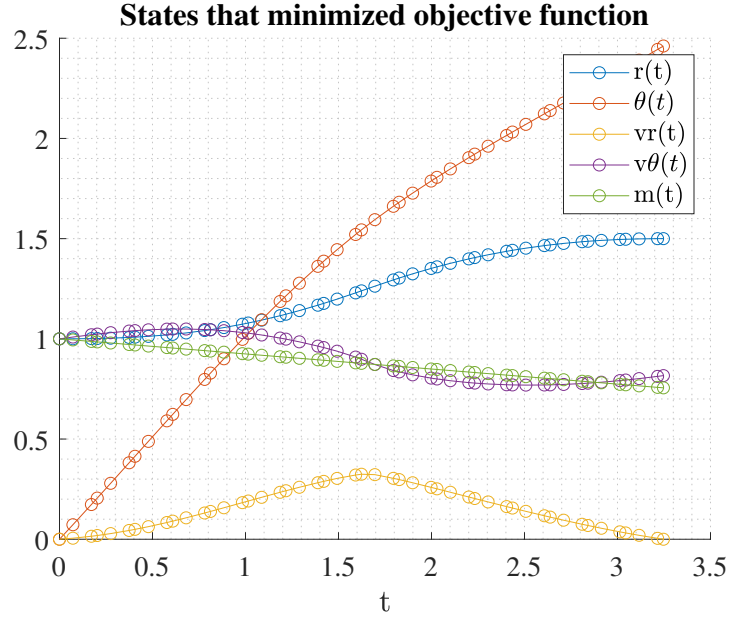


Figure 77: States for trajectory that minimized terminal time ($N : 3, K : 16$)

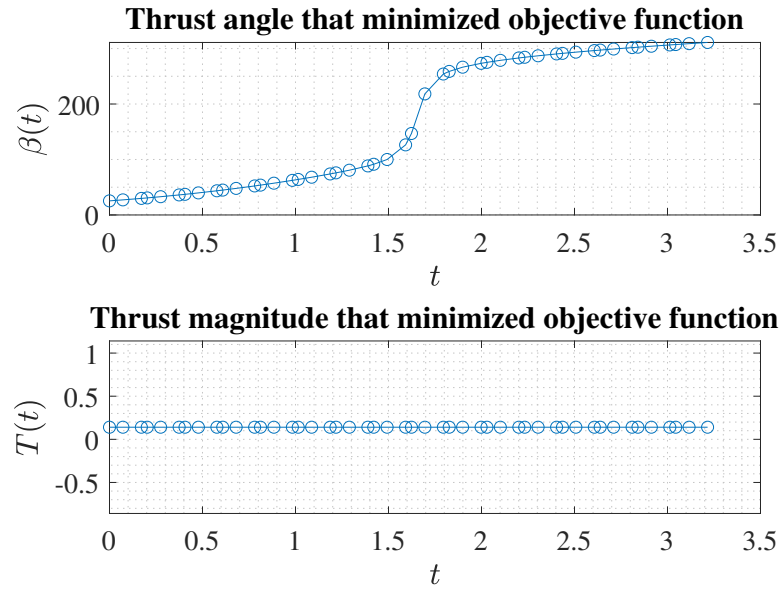


Figure 78: Control that minimized terminal time ($N : 3, K : 16$)

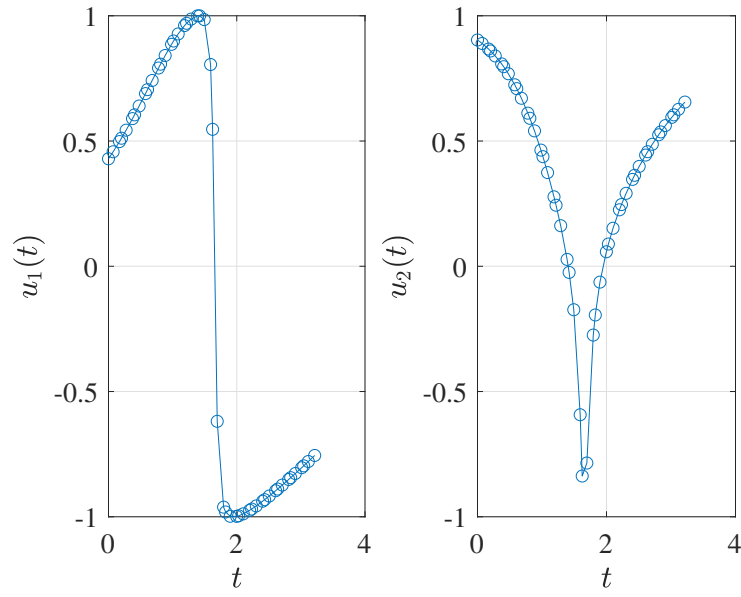


Figure 79: Path constrained control that minimized terminal time ($N : 3$, $K : 16$)

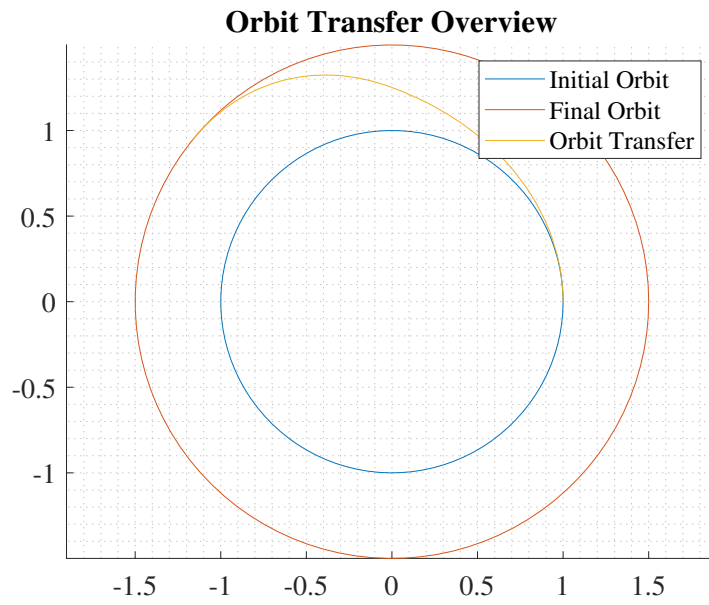


Figure 80: Trajectory from initial to final orbit ($N : 3$, $K : 16$)

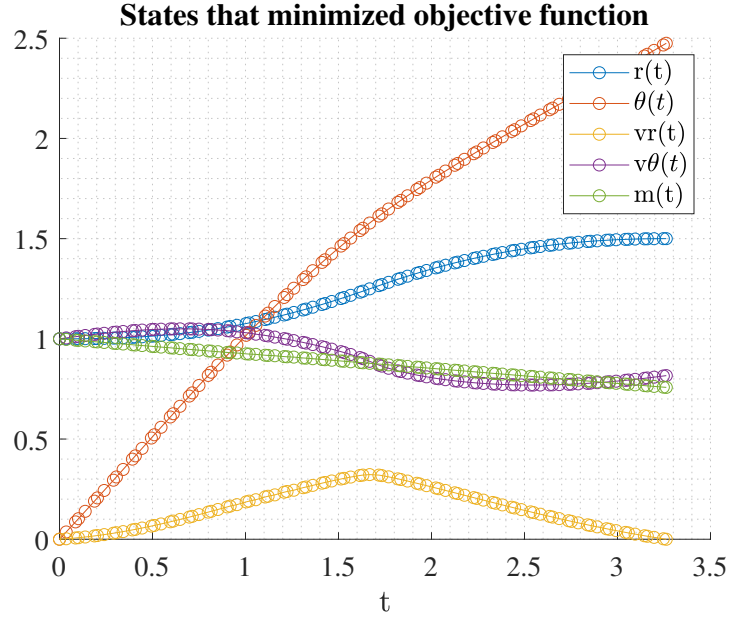


Figure 81: States for trajectory that minimized terminal time ($N : 3, K : 32$)

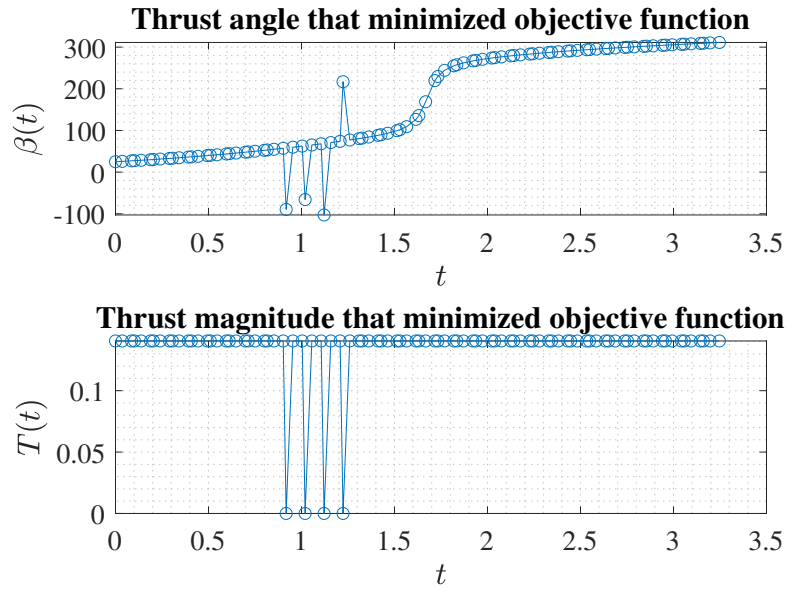


Figure 82: Control that minimized terminal time ($N : 3, K : 32$)

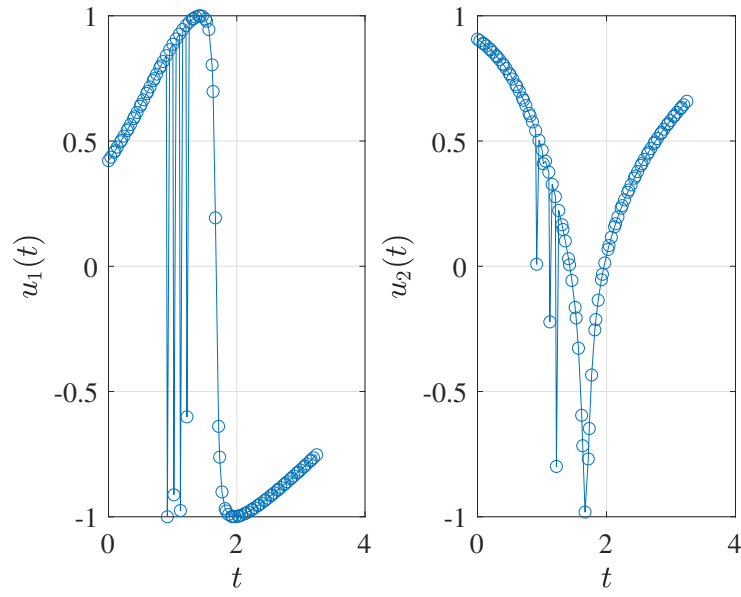


Figure 83: Path constrained control that minimized terminal time ($N : 3$, $K : 32$)

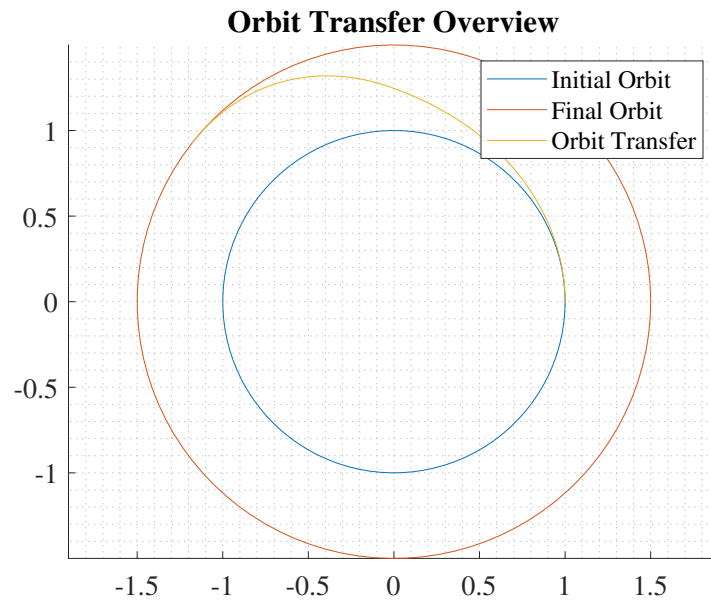


Figure 84: Trajectory from initial to final orbit ($N : 3$, $K : 32$)

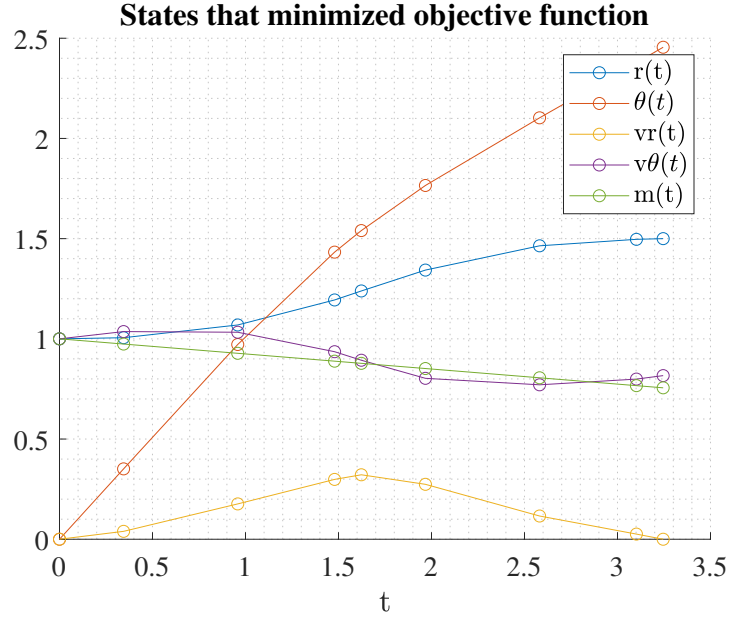


Figure 85: States for trajectory that minimized terminal time ($N : 4$, $K : 2$)

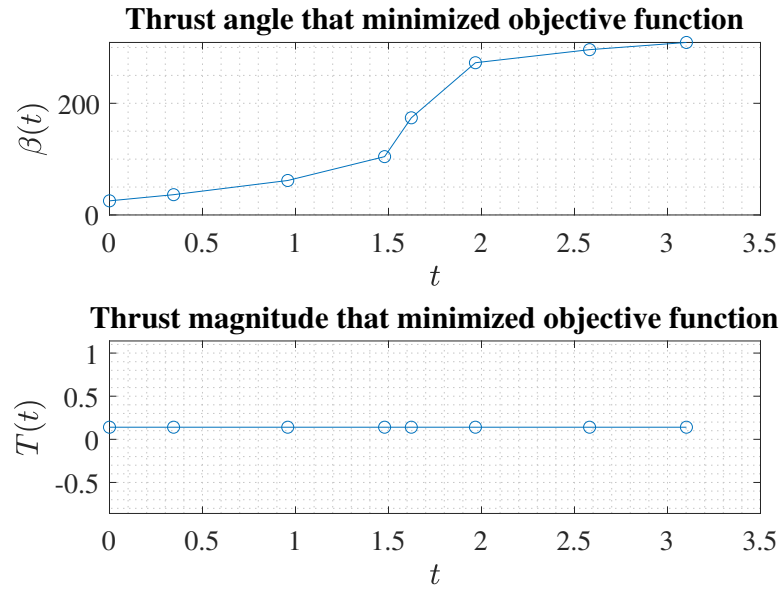


Figure 86: Control that minimized terminal time ($N : 4$, $K : 2$)

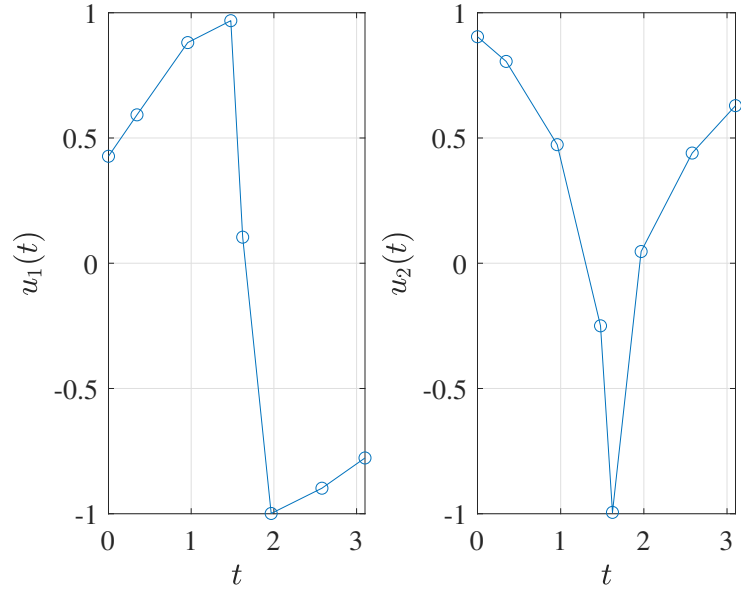


Figure 87: Path constrained control that minimized terminal time ($N : 4, K : 2$)

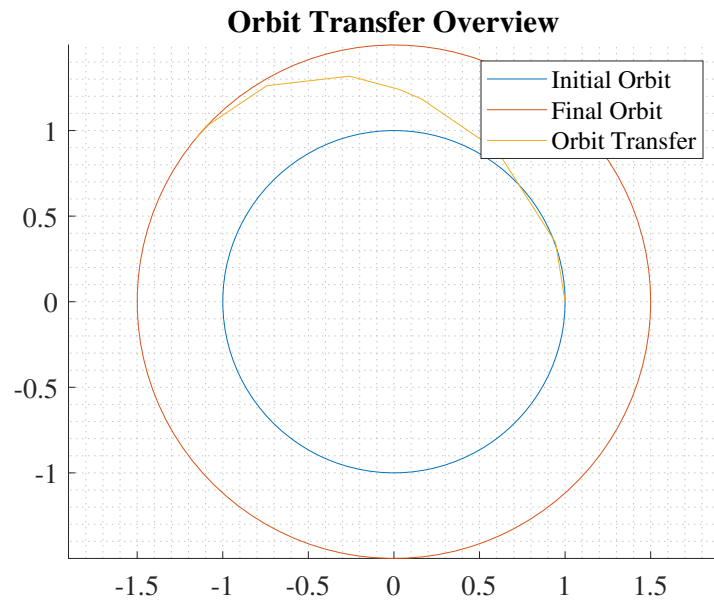


Figure 88: Trajectory from initial to final orbit ($N : 4, K : 2$)

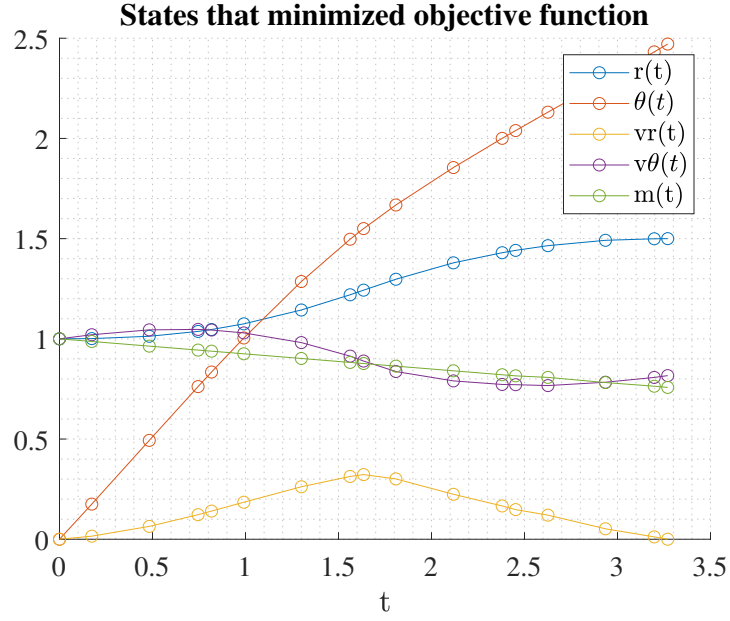


Figure 89: States for trajectory that minimized terminal time ($N : 4$, $K : 4$)

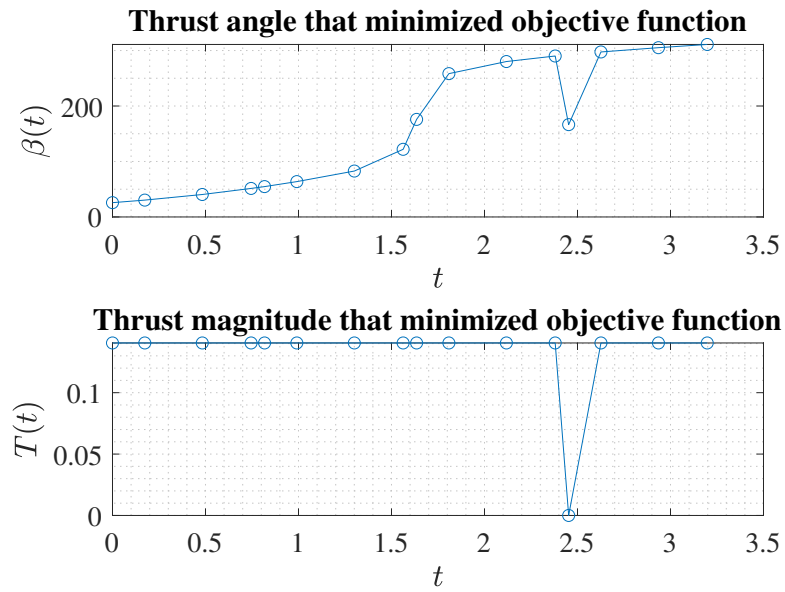


Figure 90: Control that minimized terminal time ($N : 4$, $K : 4$)

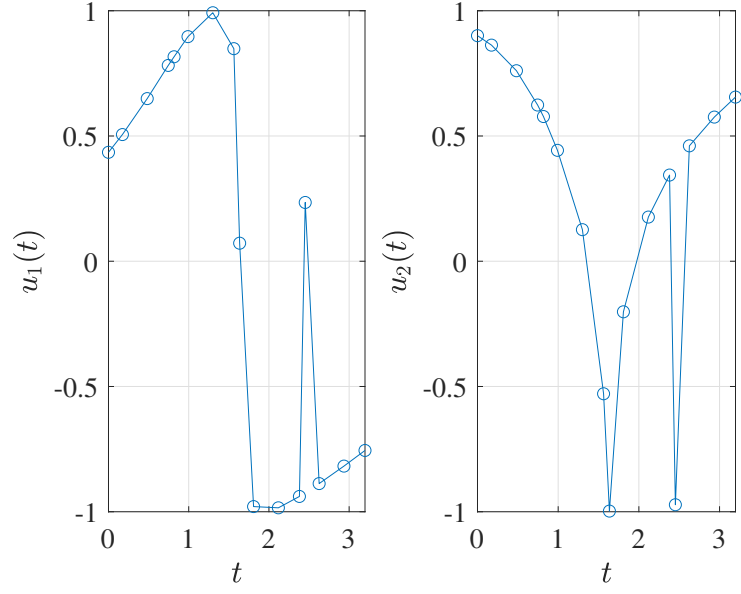


Figure 91: Path constrained control that minimized terminal time ($N : 4, K : 4$)

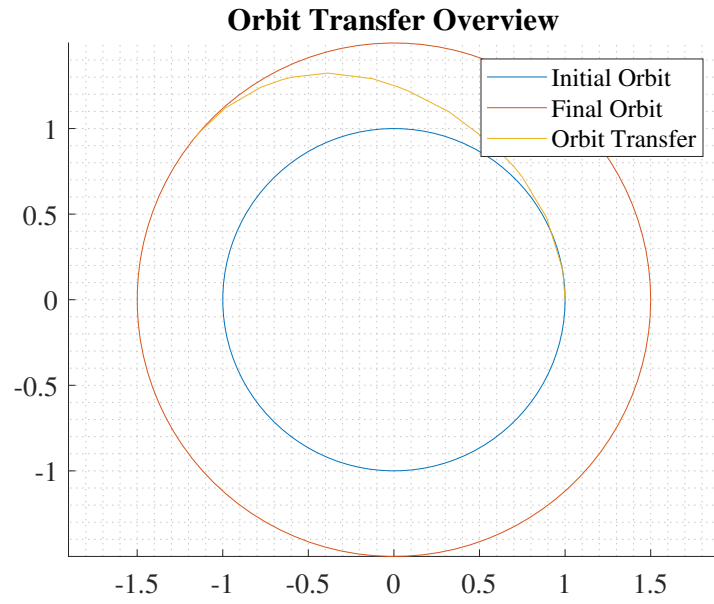


Figure 92: Trajectory from initial to final orbit ($N : 4, K : 4$)

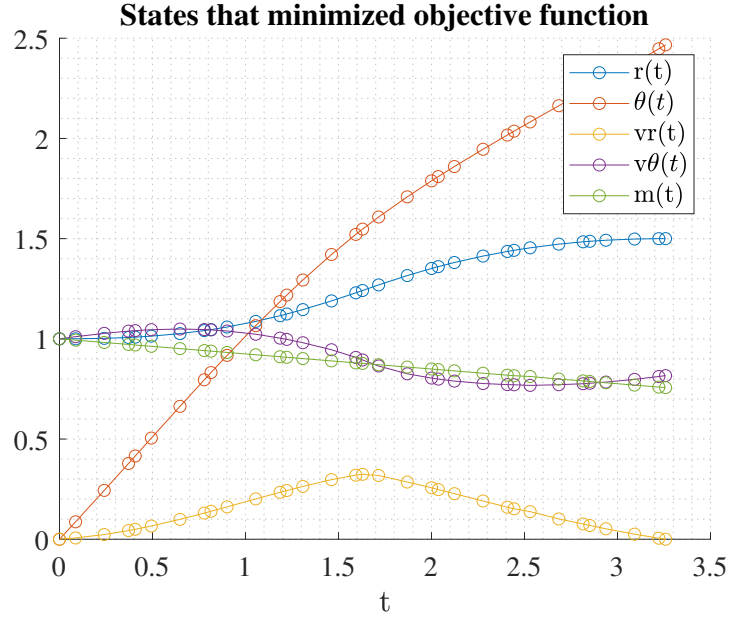


Figure 93: States for trajectory that minimized terminal time ($N : 4$, $K : 8$)

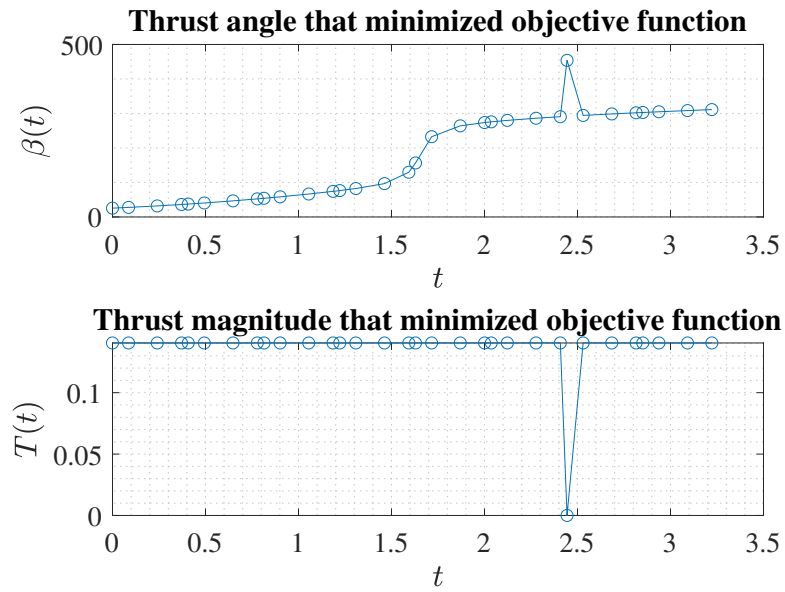


Figure 94: Control that minimized terminal time ($N : 4$, $K : 8$)

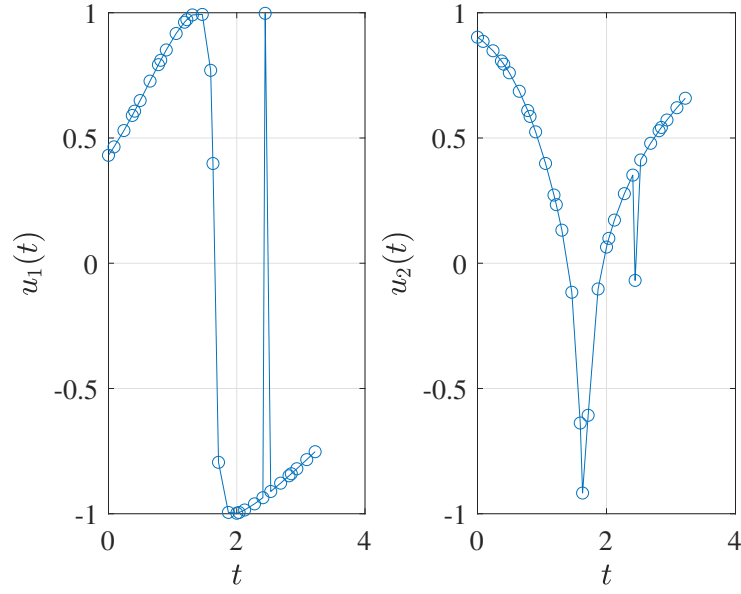


Figure 95: Path constrained control that minimized terminal time ($N : 4, K : 8$)

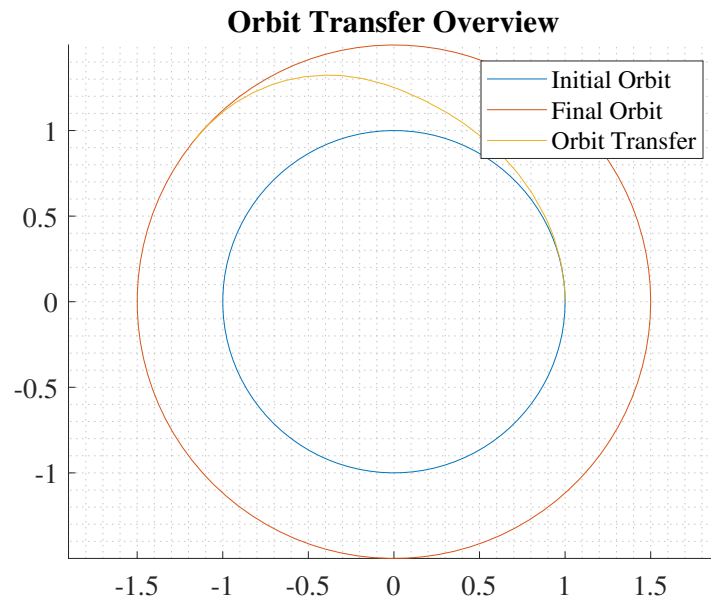


Figure 96: Trajectory from initial to final orbit ($N : 4, K : 8$)

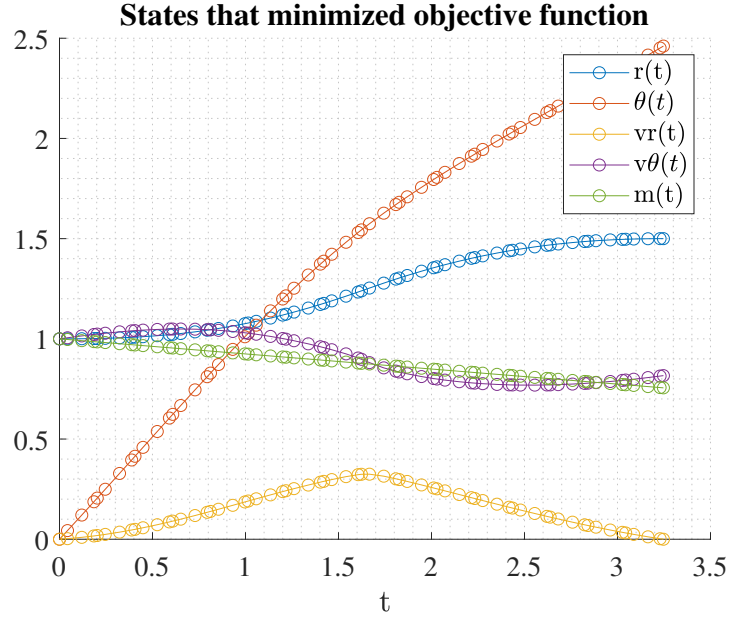


Figure 97: States for trajectory that minimized terminal time ($N : 4$, $K : 16$)

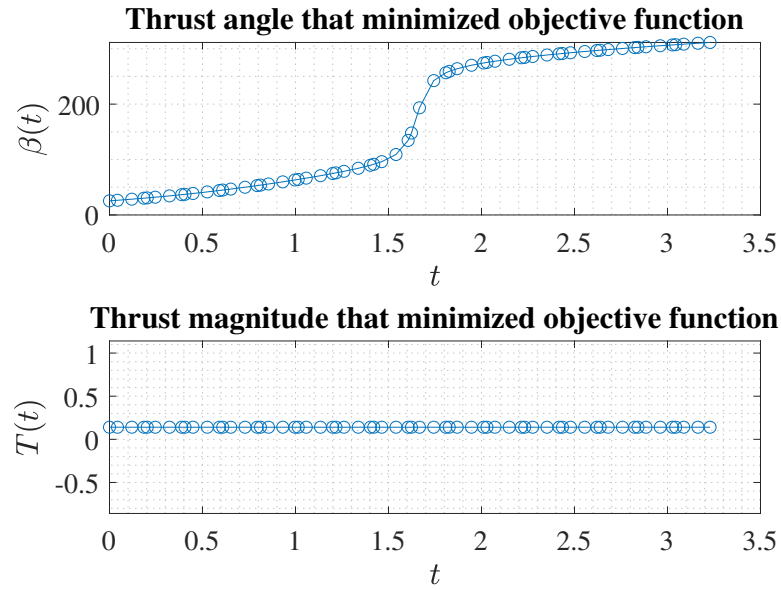


Figure 98: Control that minimized terminal time ($N : 4$, $K : 16$)

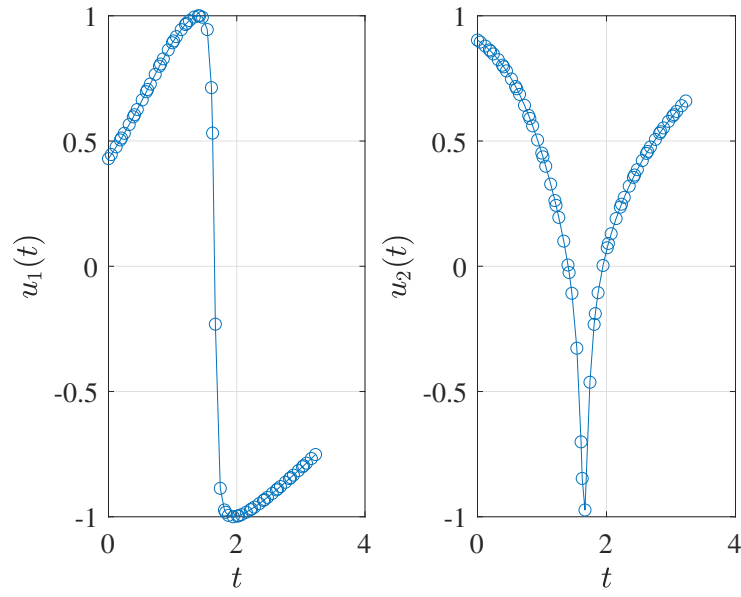


Figure 99: Path constrained control that minimized terminal time ($N : 4 , K : 16$)

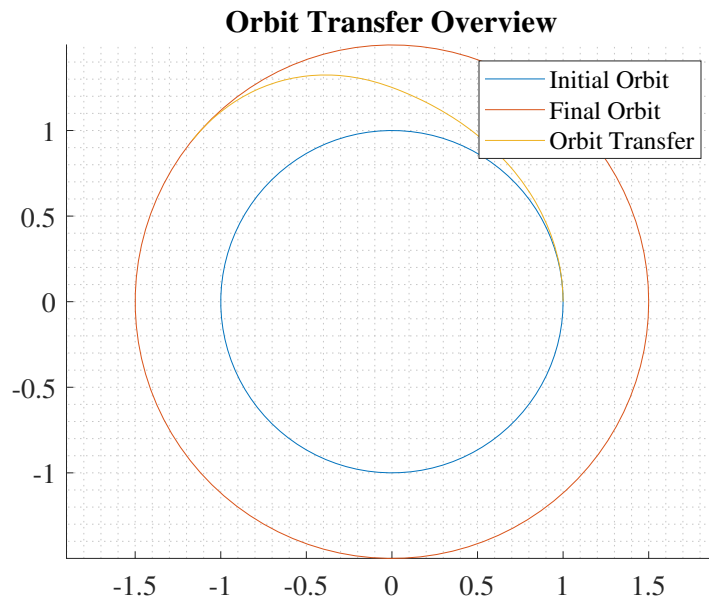


Figure 100: Trajectory from initial to final orbit ($N : 4 , K : 16$)

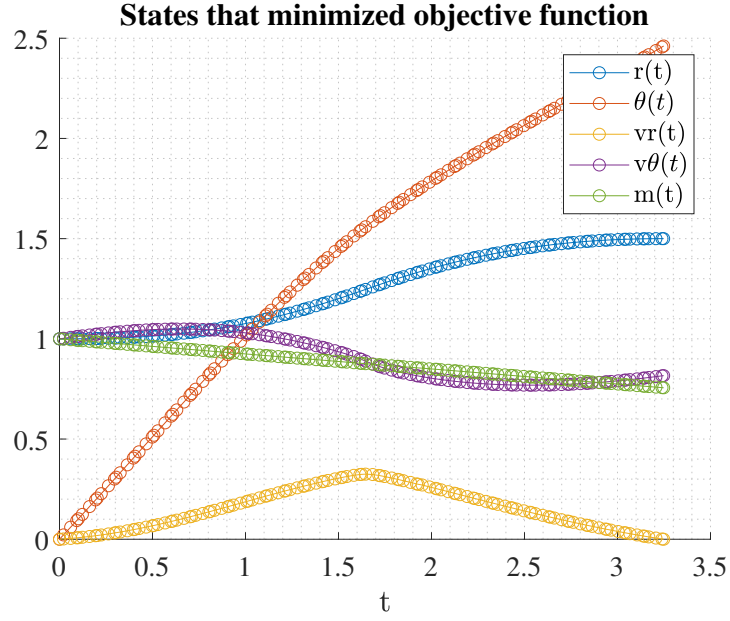


Figure 101: States for trajectory that minimized terminal time ($N : 4$, $K : 32$)

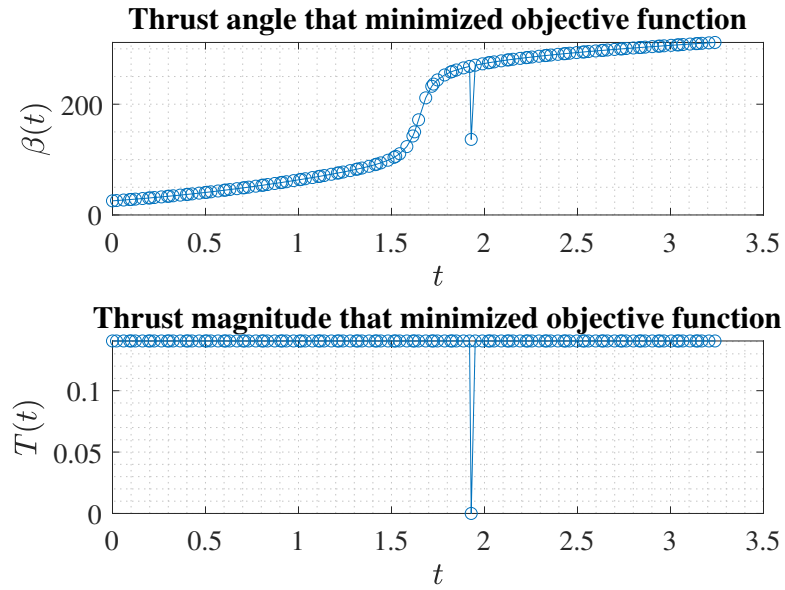


Figure 102: Control that minimized terminal time ($N : 4$, $K : 32$)

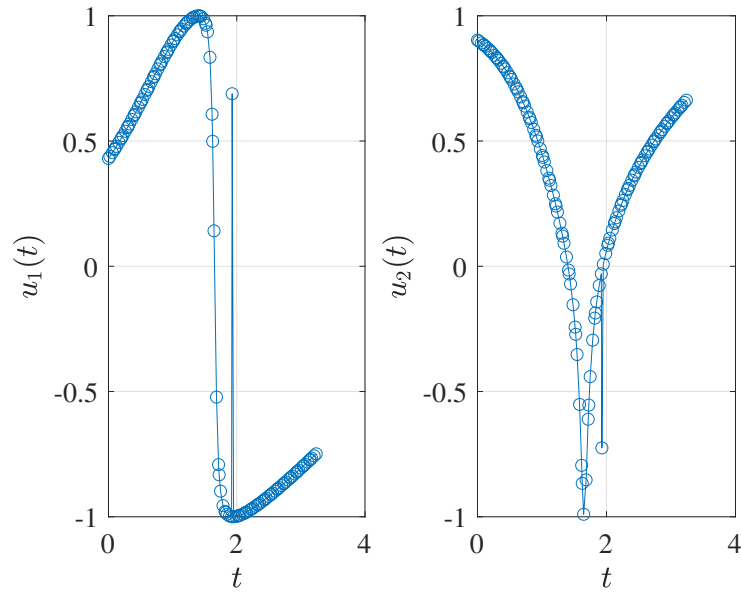


Figure 103: Path constrained control that minimized terminal time ($N : 4, K : 32$)

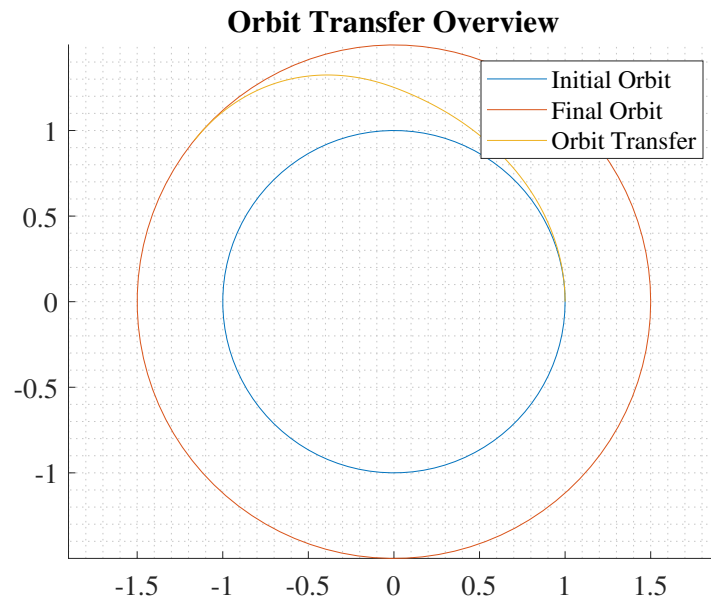


Figure 104: Trajectory from initial to final orbit ($N : 4, K : 32$)

4.4 Maximize Terminal Mass with Constrained Control

Degree	Intervals	Iterations	CPU Time	tf	mf	Solved Status
3	2	110	0.417	15.0512	0.90547	-2
3	4	2185	7.269	14.4119	0.9095	0
3	8	728	2.557	6.9851	0.90719	0
3	16	96	0.465	7.9129	0.9072	0
3	32	102	0.559	6.689	0.90721	1
4	2	531	1.765	12.3575	0.91207	0
4	4	415	1.465	6.1534	0.90718	0
4	8	397	1.511	7.0005	0.90721	0
4	16	141	0.688	6.3843	0.90721	1
4	32	1016	5.404	7.0926	0.90721	1

Table 4: Results for maximizing m_f with constrained control

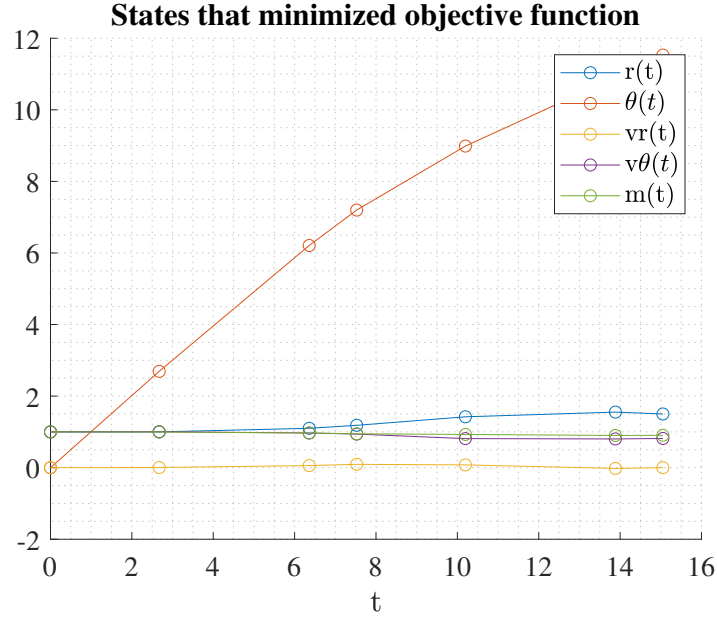


Figure 105: States for trajectory that maximized terminal mass ($N : 3, K : 2$)

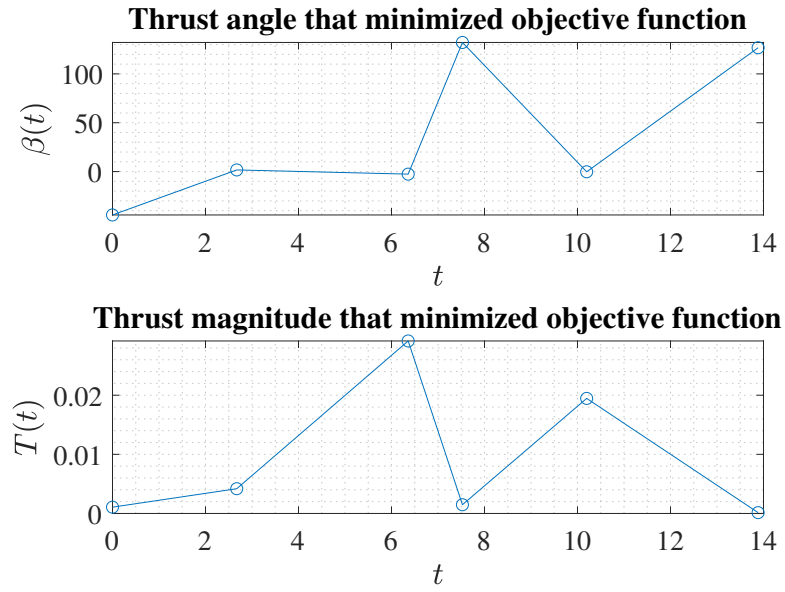


Figure 106: Control that maximized terminal mass ($N : 3, K : 2$)

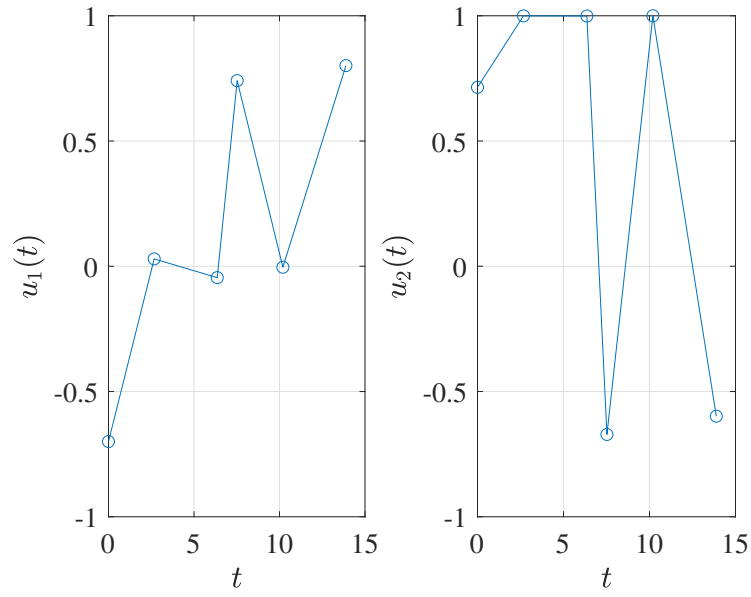


Figure 107: Path constrained control that maximized terminal mass ($N : 3, K : 2$)

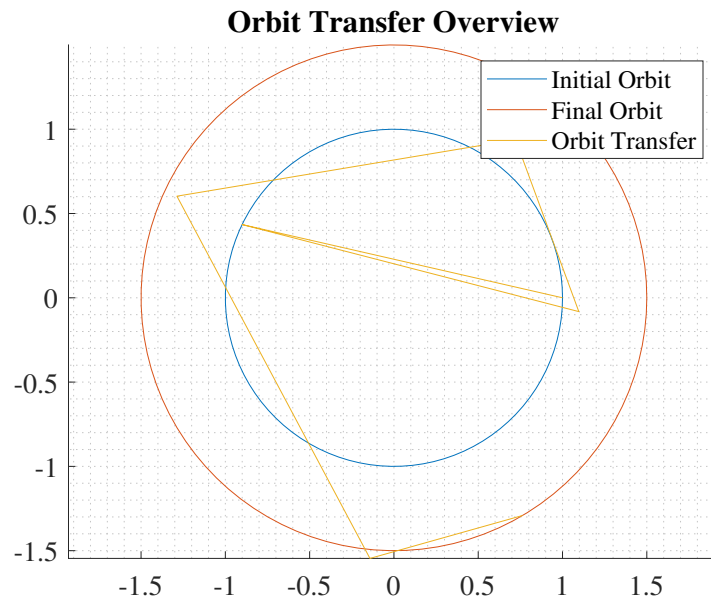


Figure 108: Trajectory from initial to final orbit ($N : 3, K : 2$)

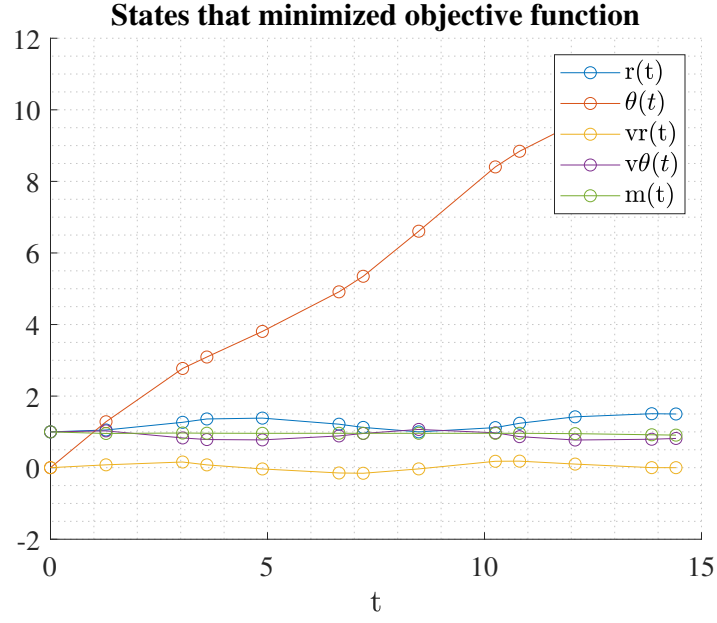


Figure 109: States for trajectory that maximized terminal mass ($N : 3, K : 4$)

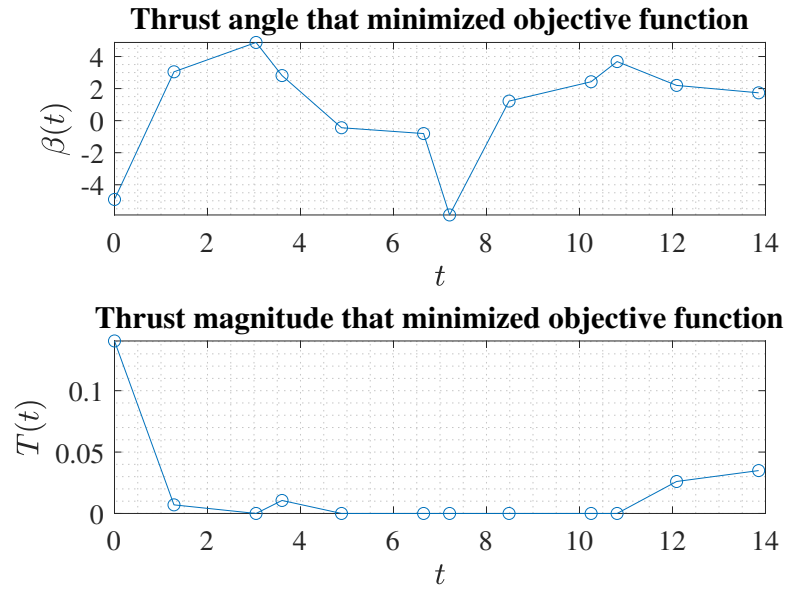


Figure 110: Control that maximized terminal mass ($N : 3, K : 4$)

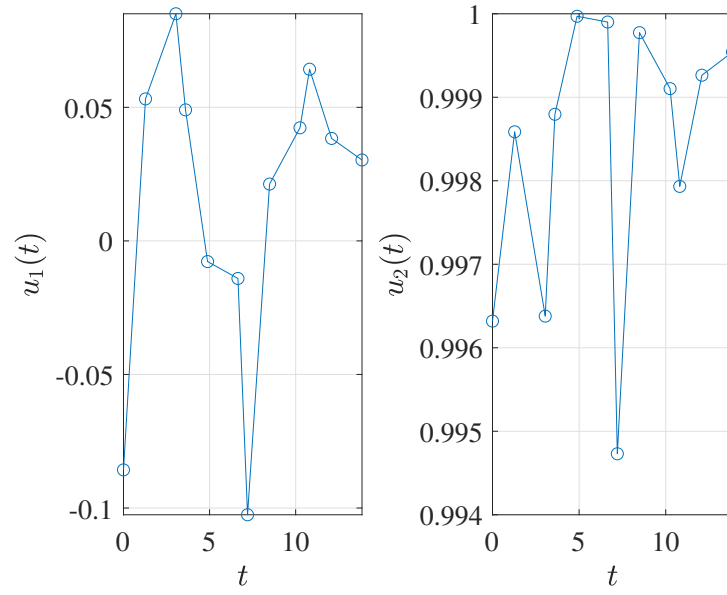


Figure 111: Path constrained control that maximized terminal mass ($N : 3, K : 4$)

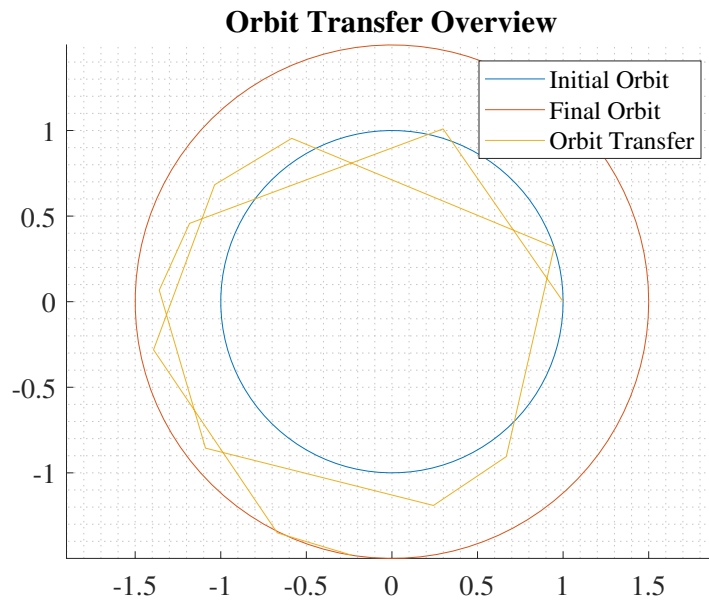


Figure 112: Trajectory from initial to final orbit ($N : 3, K : 4$)

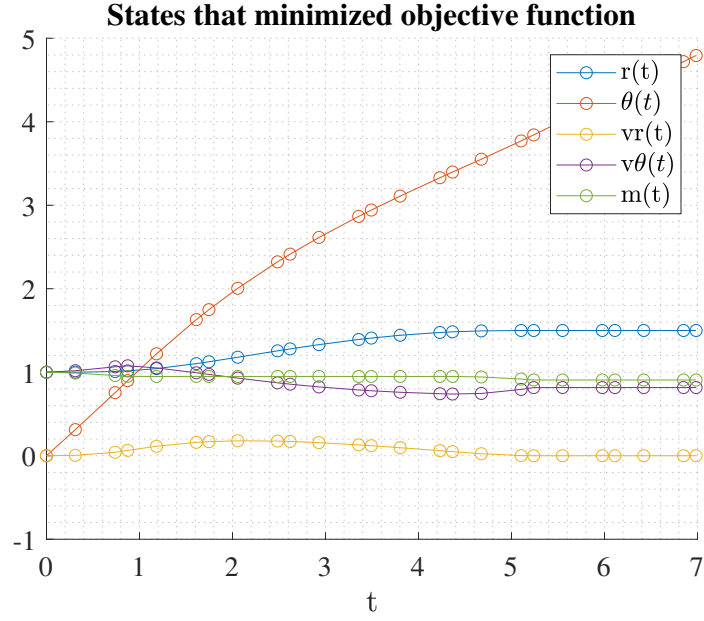


Figure 113: States for trajectory that maximized terminal mass ($N : 3, K : 8$)

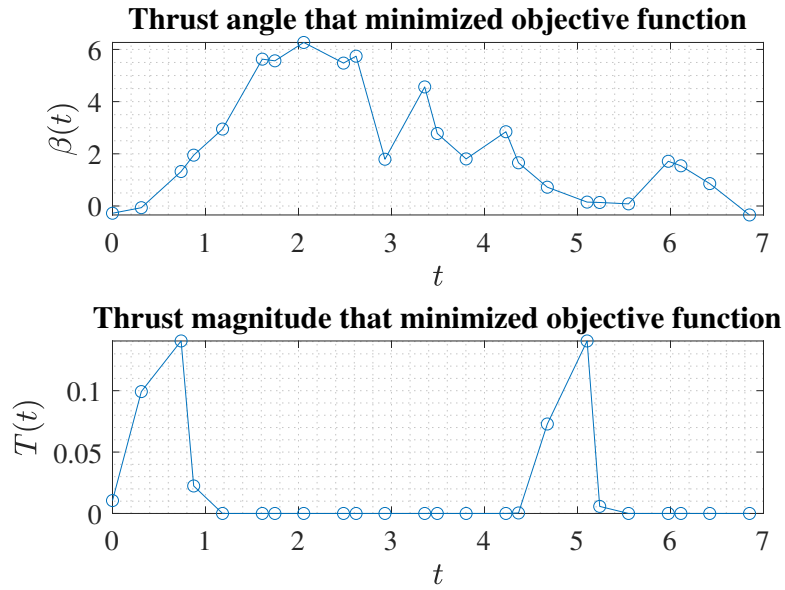


Figure 114: Control that maximized terminal mass ($N : 3, K : 8$)

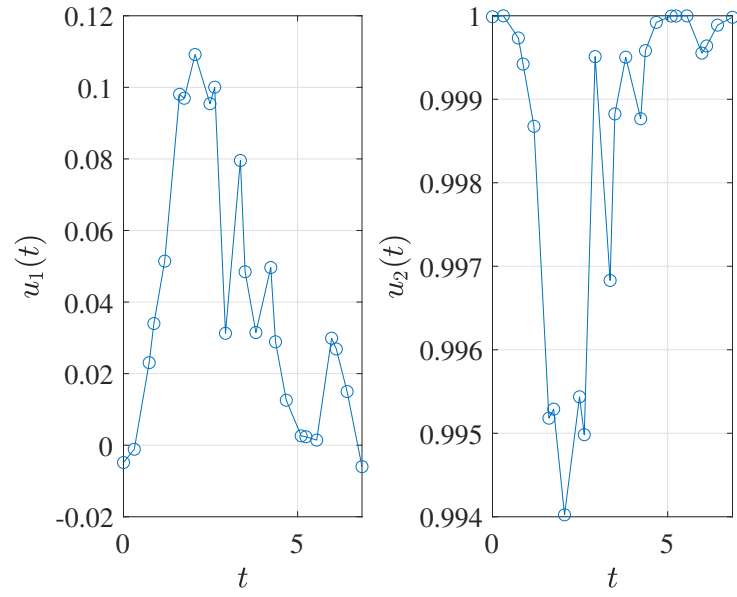


Figure 115: Path constrained control that maximized terminal mass ($N : 3, K : 8$)

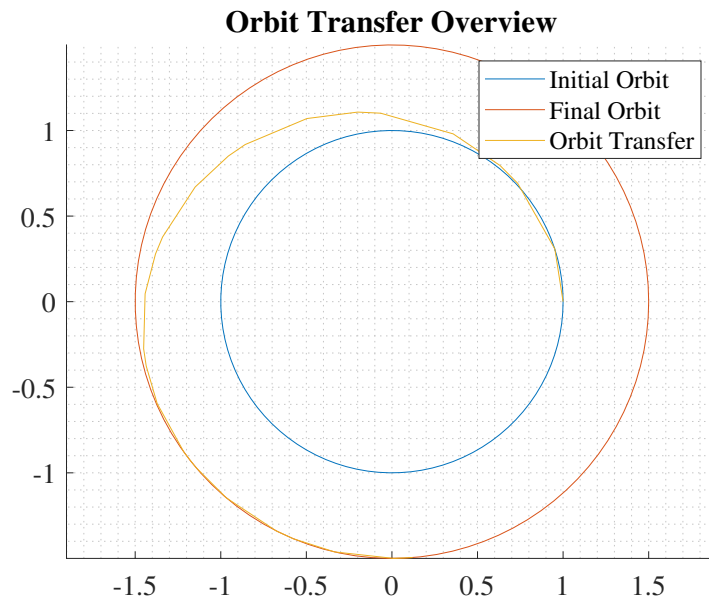


Figure 116: Trajectory from initial to final orbit ($N : 3, K : 8$)

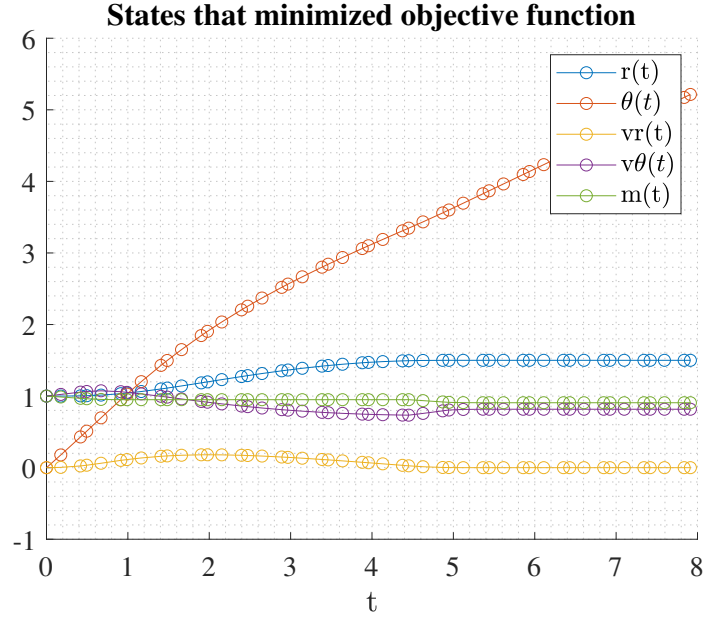


Figure 117: States for trajectory that maximized terminal mass ($N : 3$, $K : 16$)

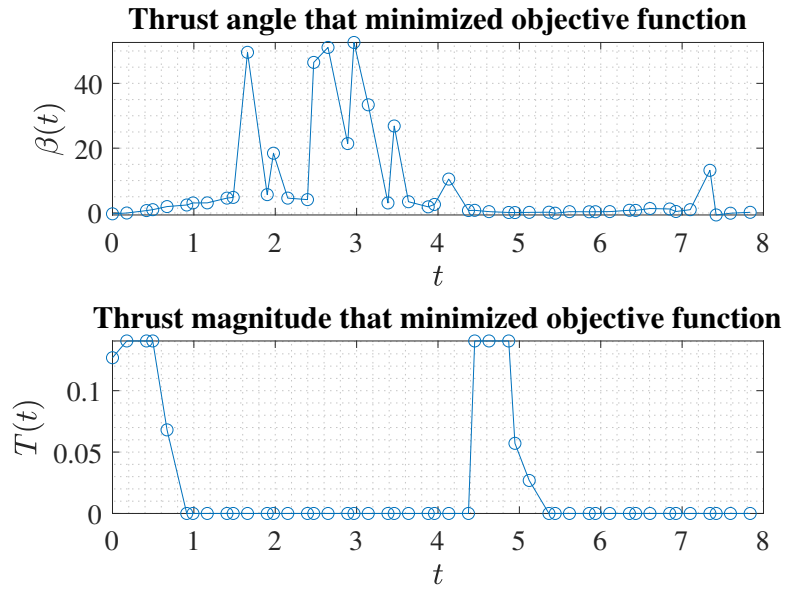


Figure 118: Control that maximized terminal mass ($N : 3$, $K : 16$)

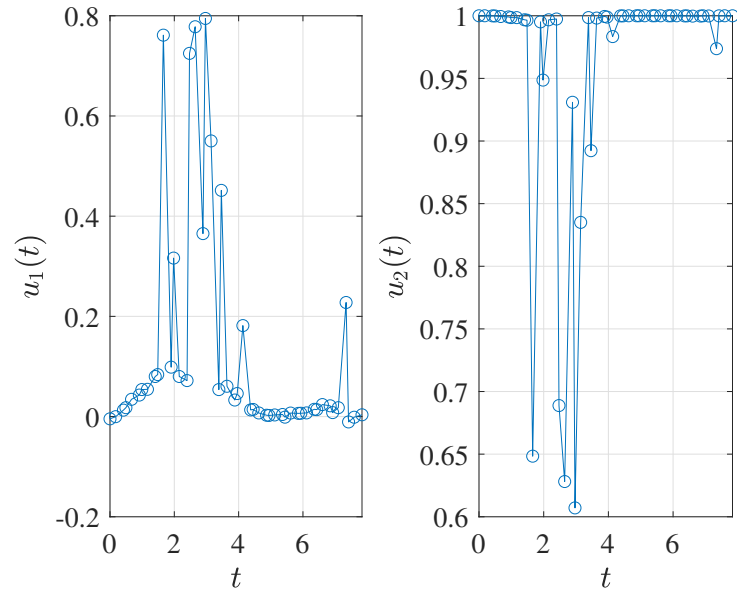


Figure 119: Path constrained control that maximized terminal mass ($N : 3, K : 16$)

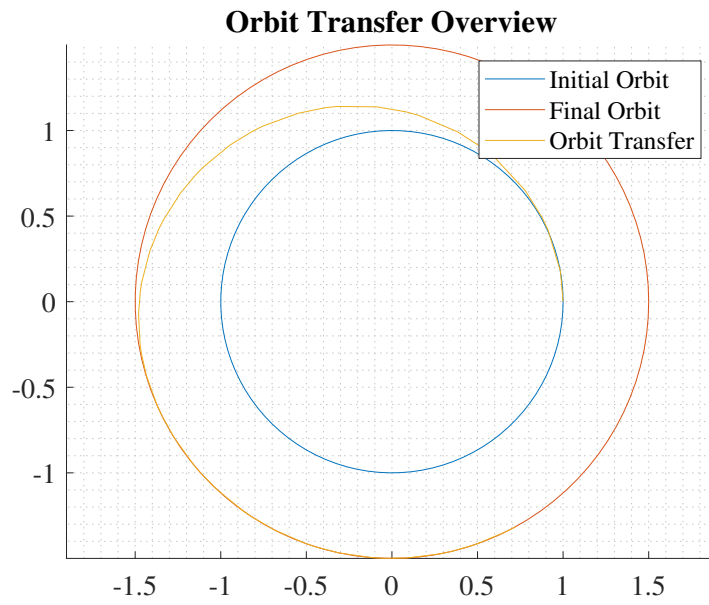


Figure 120: Trajectory from initial to final orbit ($N : 3, K : 16$)

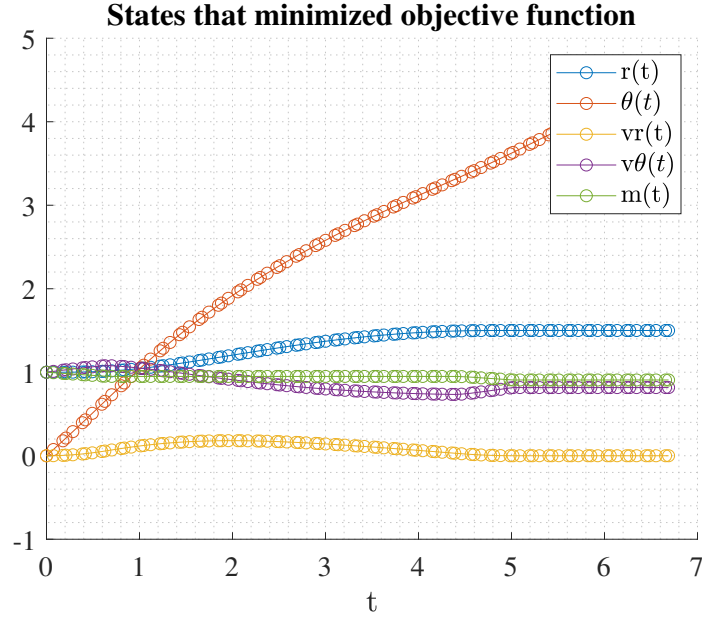


Figure 121: States for trajectory that maximized terminal mass ($N : 3$, $K : 32$)

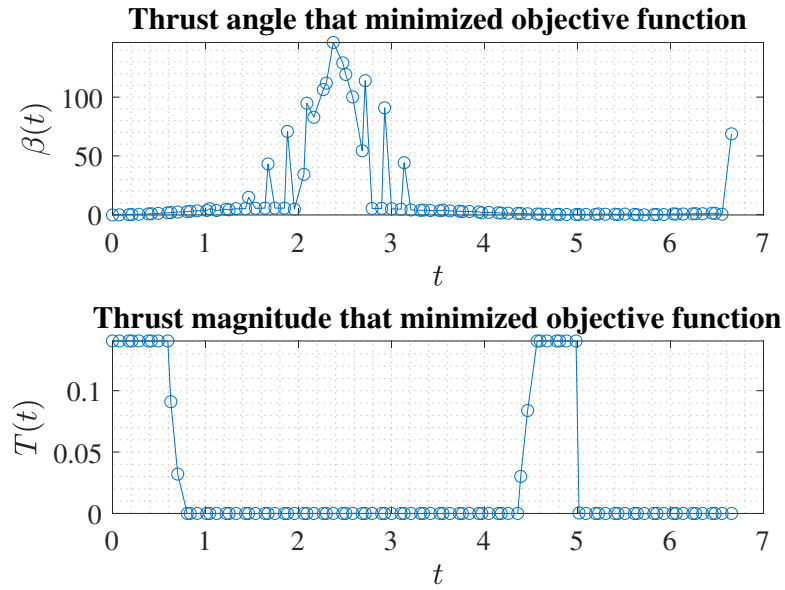


Figure 122: Control that maximized terminal mass ($N : 3$, $K : 32$)

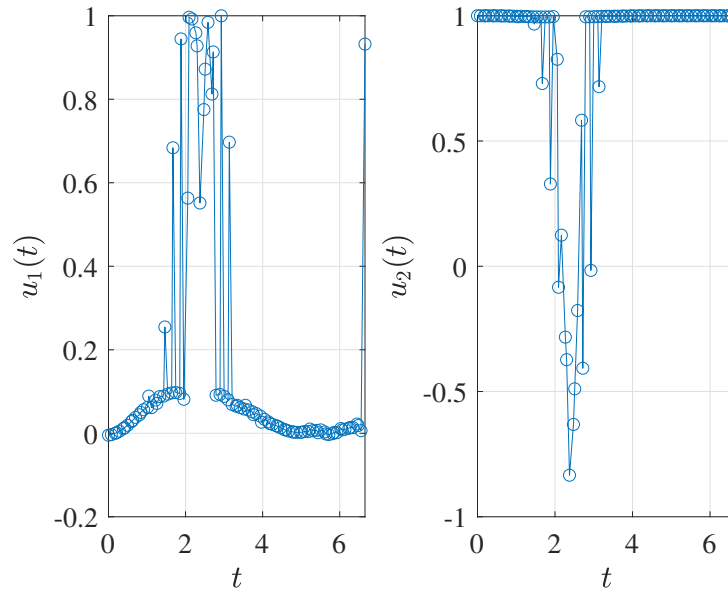


Figure 123: Path constrained control that maximized terminal mass ($N : 3 , K : 32$)

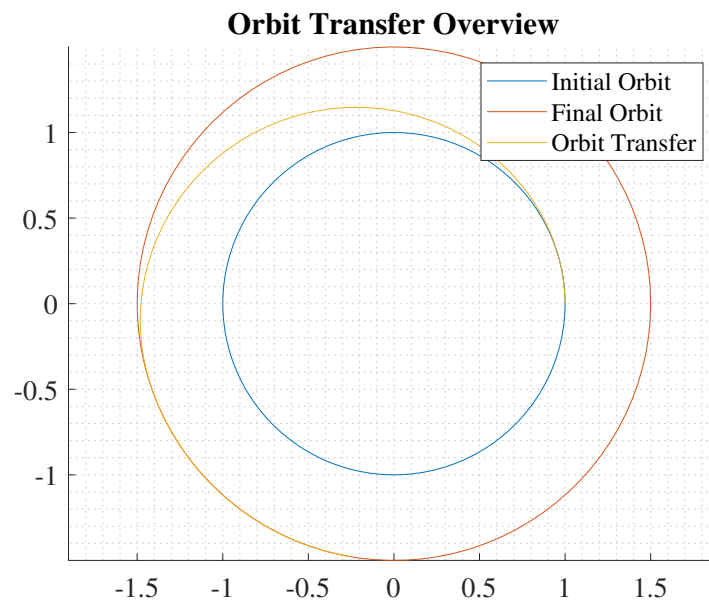


Figure 124: Trajectory from initial to final orbit ($N : 3 , K : 32$)

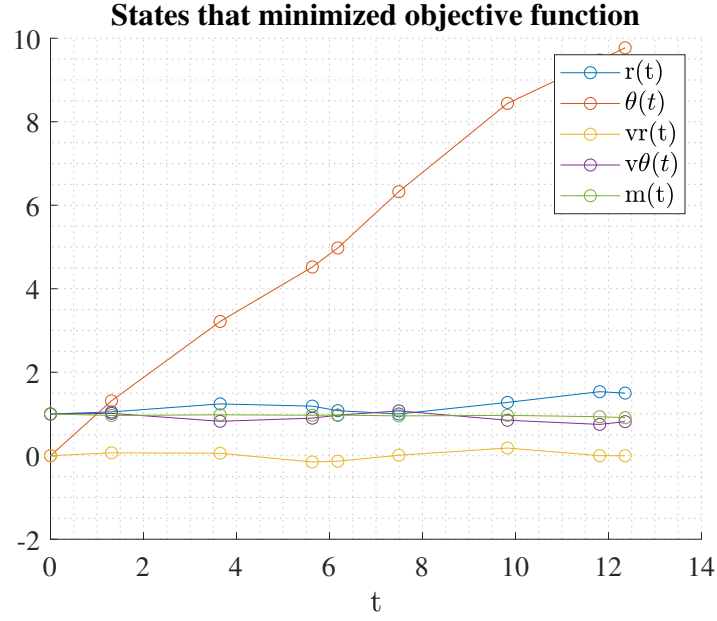


Figure 125: States for trajectory that maximized terminal mass ($N : 4, K : 2$)

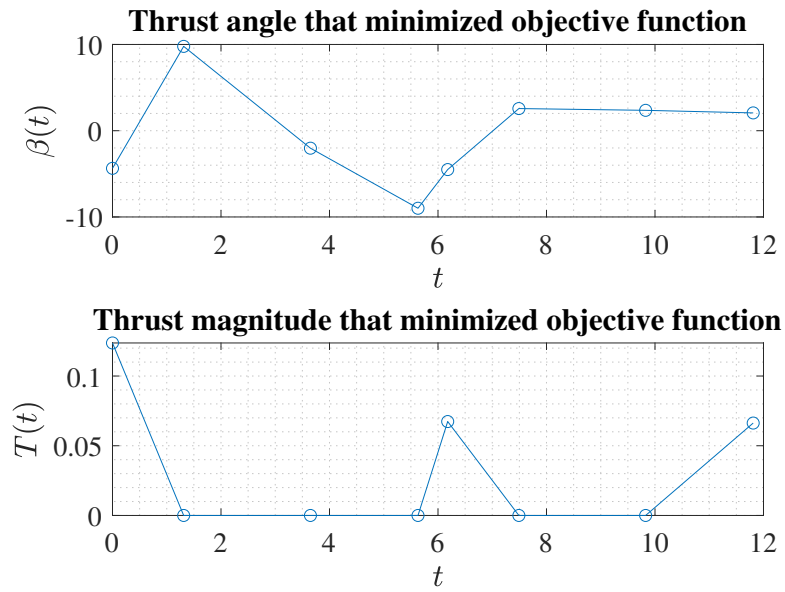


Figure 126: Control that maximized terminal mass ($N : 4, K : 2$)

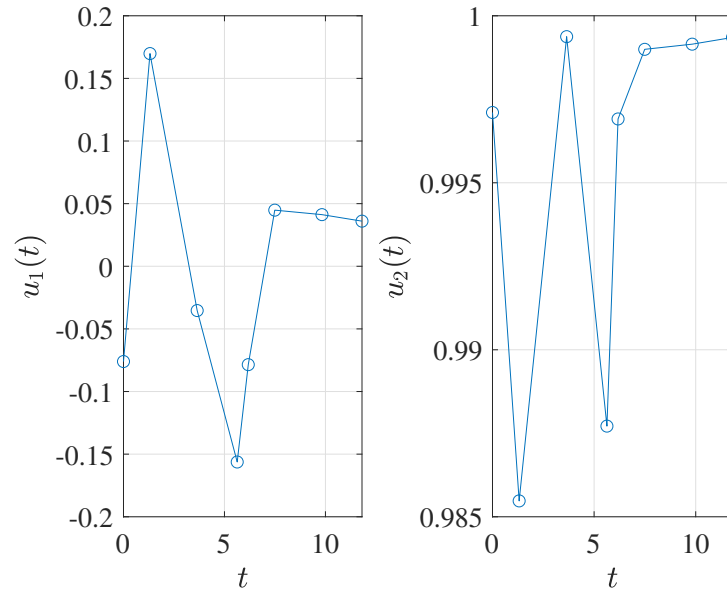


Figure 127: Path constrained control that maximized terminal mass ($N : 4, K : 2$)

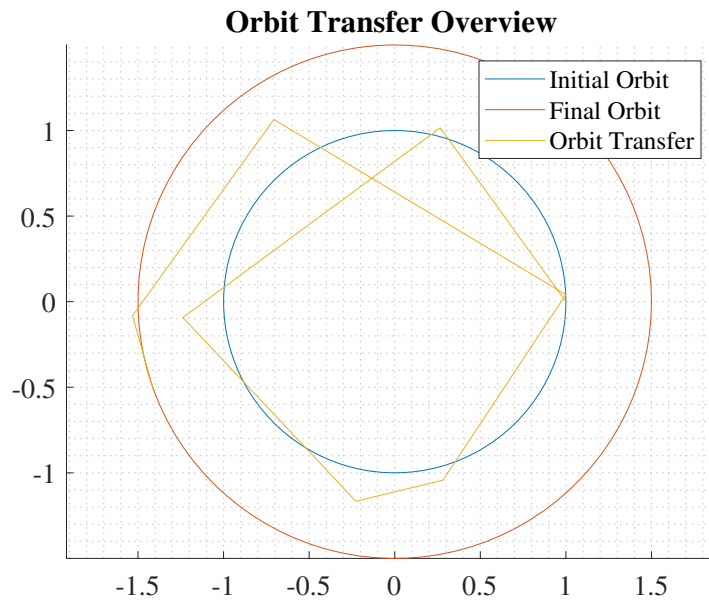


Figure 128: Trajectory from initial to final orbit ($N : 4, K : 2$)

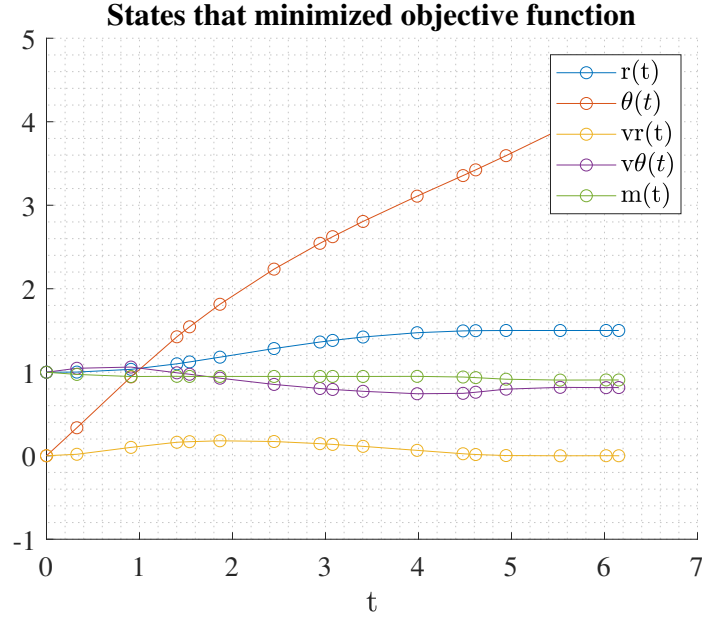


Figure 129: States for trajectory that maximized terminal mass ($N : 4, K : 4$)

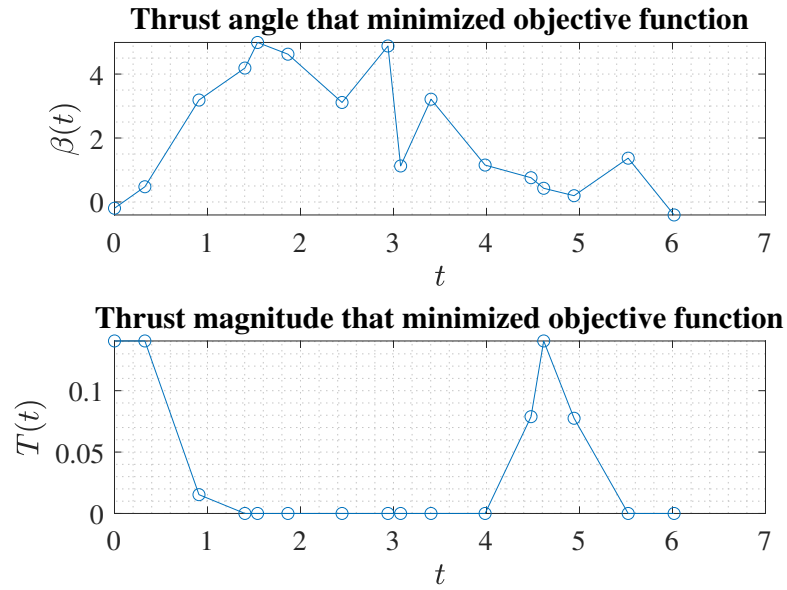


Figure 130: Control that maximized terminal mass ($N : 4, K : 4$)

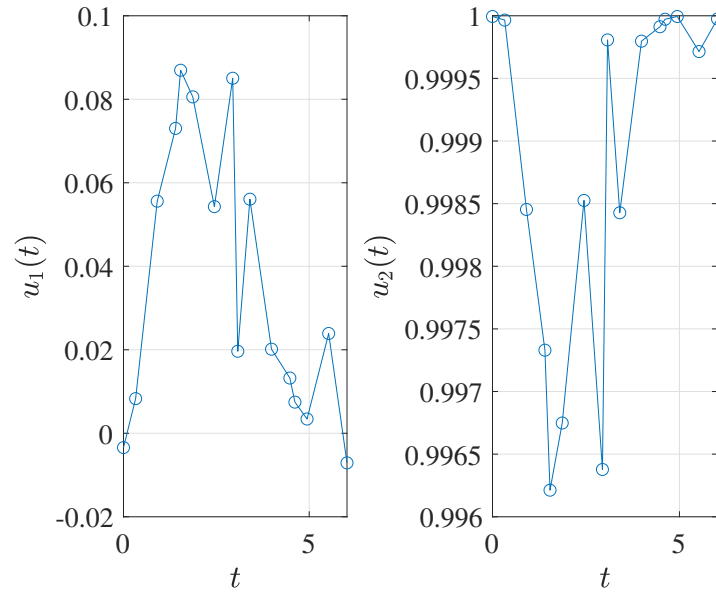


Figure 131: Path constrained control that maximized terminal mass ($N : 4, K : 4$)

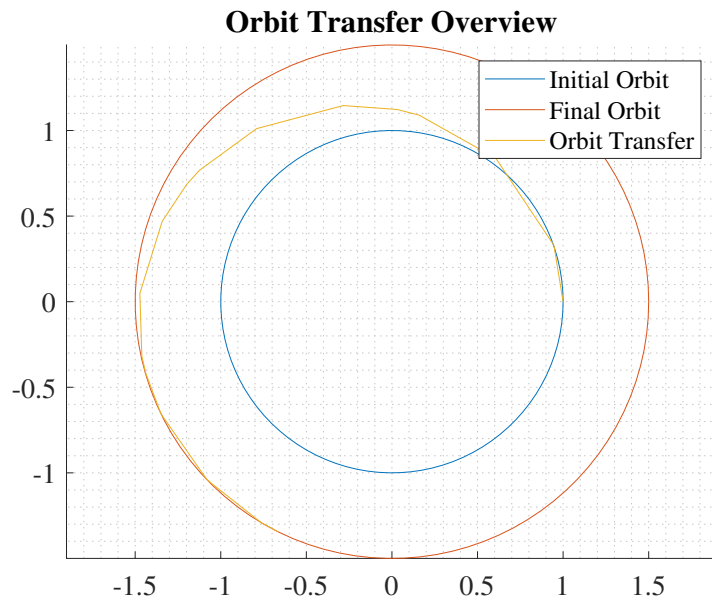


Figure 132: Trajectory from initial to final orbit ($N : 4, K : 4$)

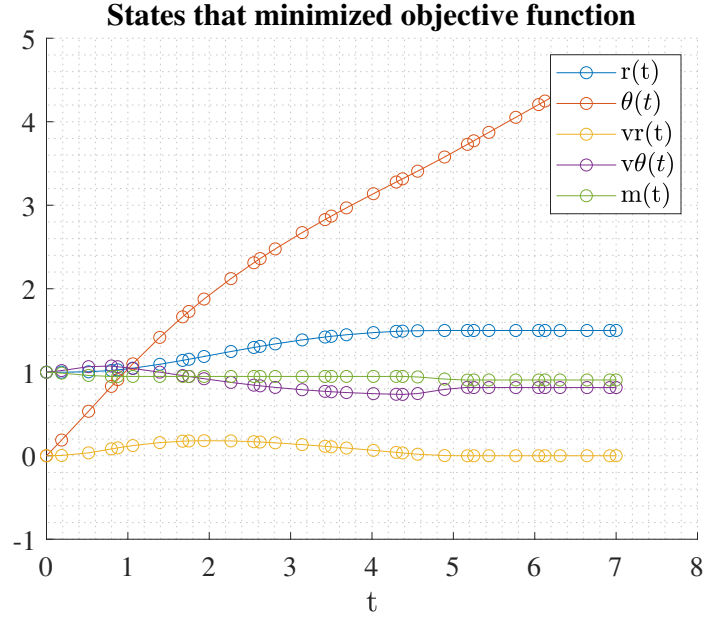


Figure 133: States for trajectory that maximized terminal mass ($N : 4, K : 8$)

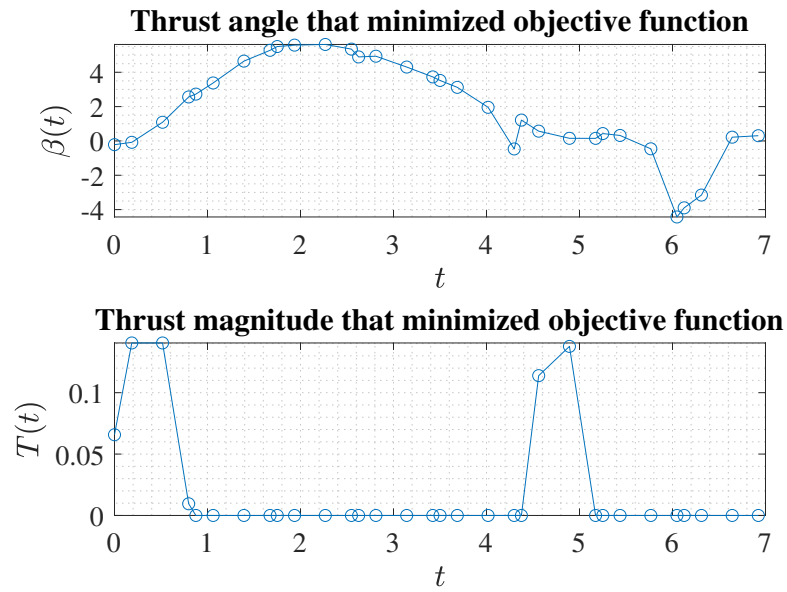


Figure 134: Control that maximized terminal mass ($N : 4, K : 8$)

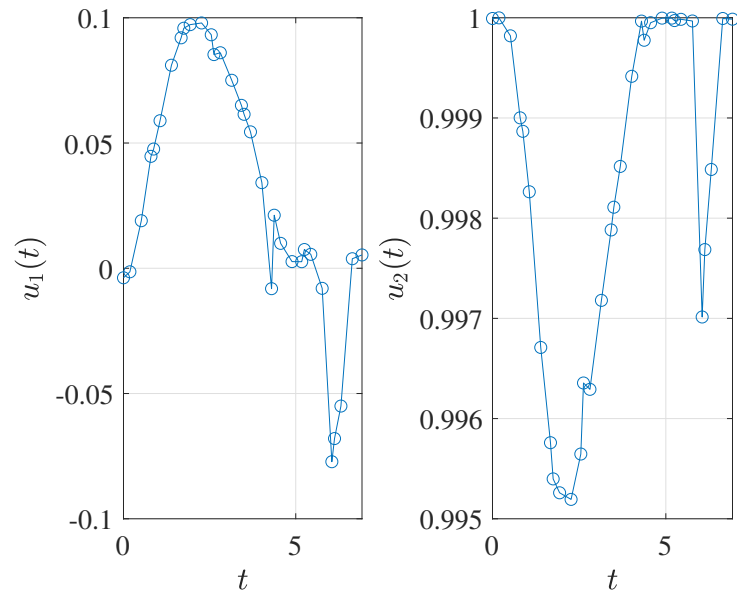


Figure 135: Path constrained control that maximized terminal mass ($N : 4, K : 8$)

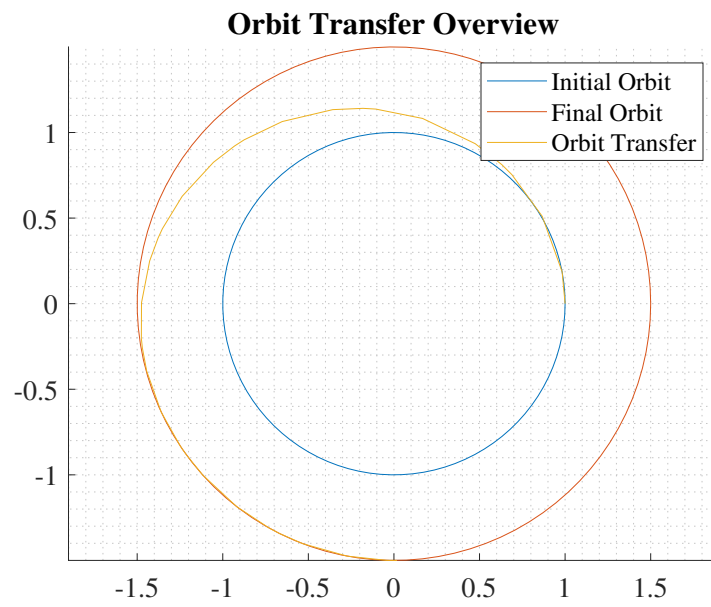


Figure 136: Trajectory from initial to final orbit ($N : 4, K : 8$)

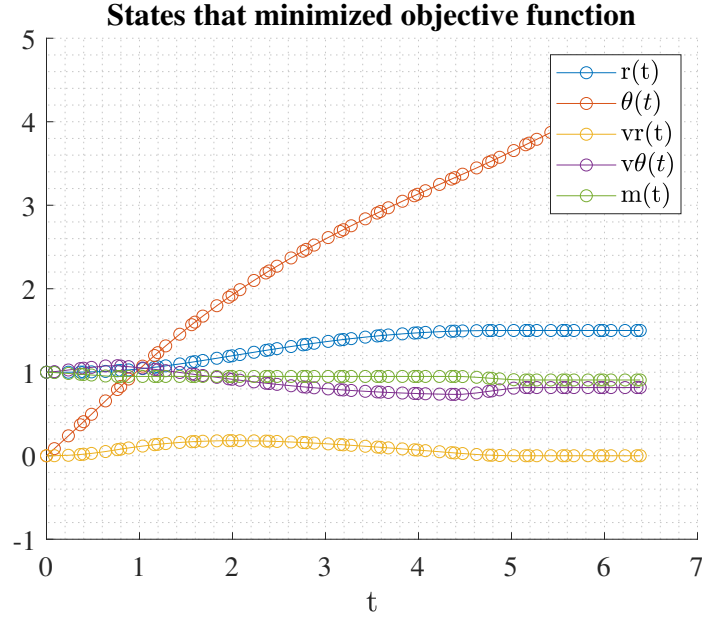


Figure 137: States for trajectory that maximized terminal mass ($N : 4$, $K : 16$)

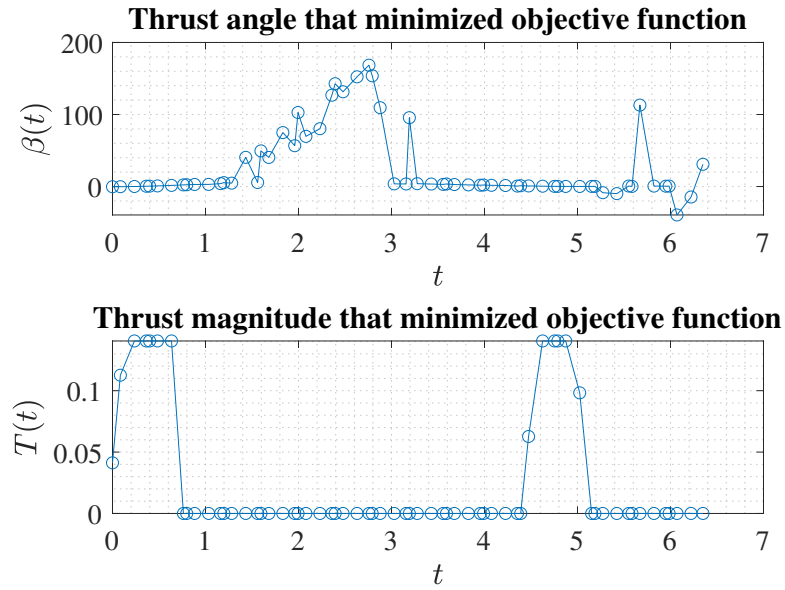


Figure 138: Control that maximized terminal mass ($N : 4$, $K : 16$)

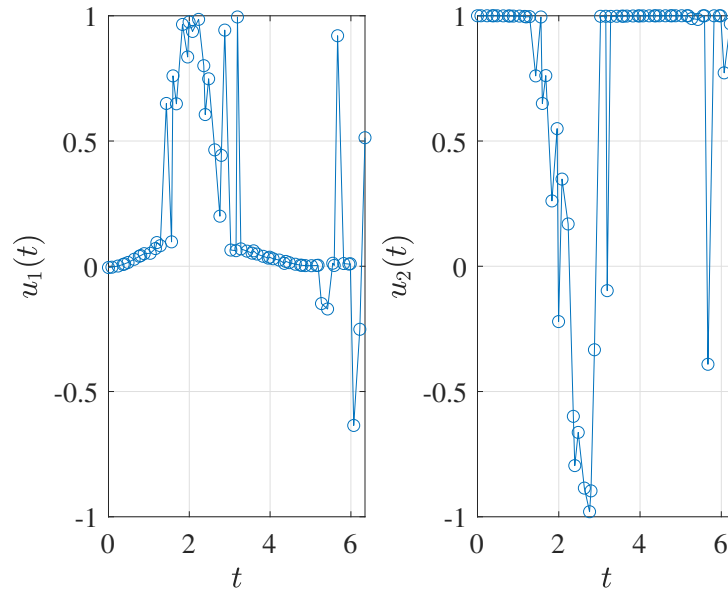


Figure 139: Path constrained control that maximized terminal mass ($N : 4, K : 16$)

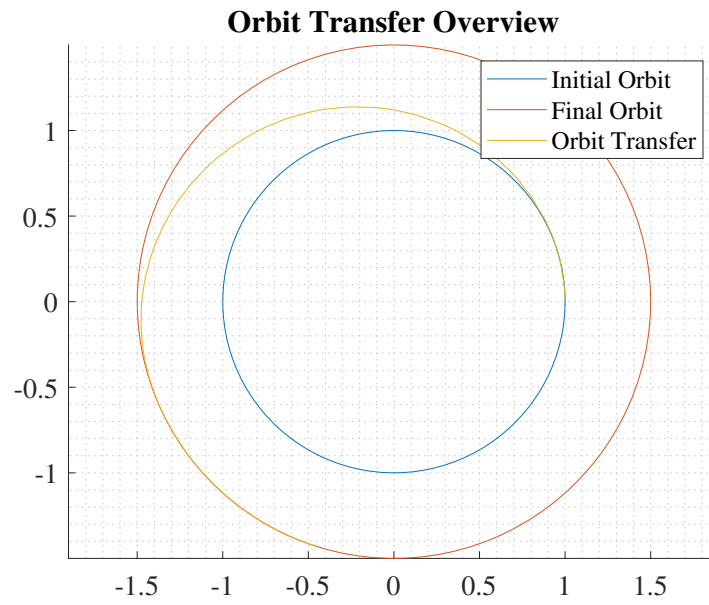


Figure 140: Trajectory from initial to final orbit ($N : 4, K : 16$)

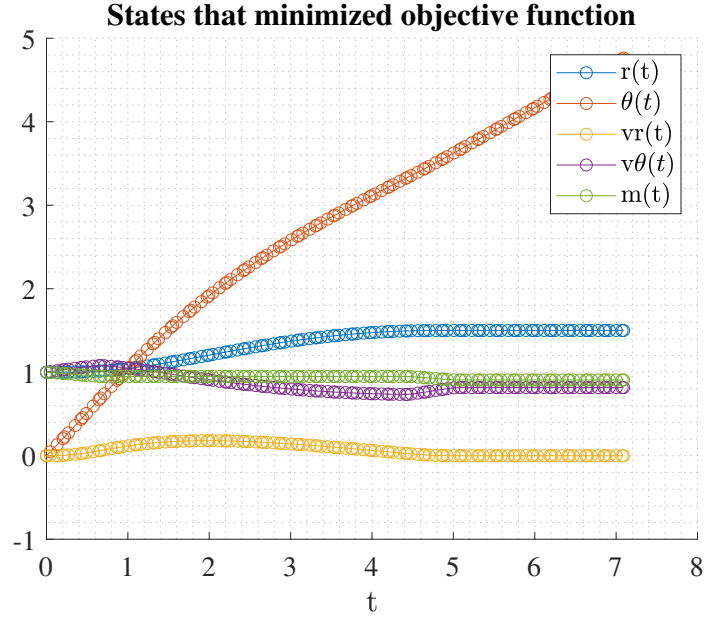


Figure 141: States for trajectory that maximized terminal mass ($N : 4$, $K : 32$)

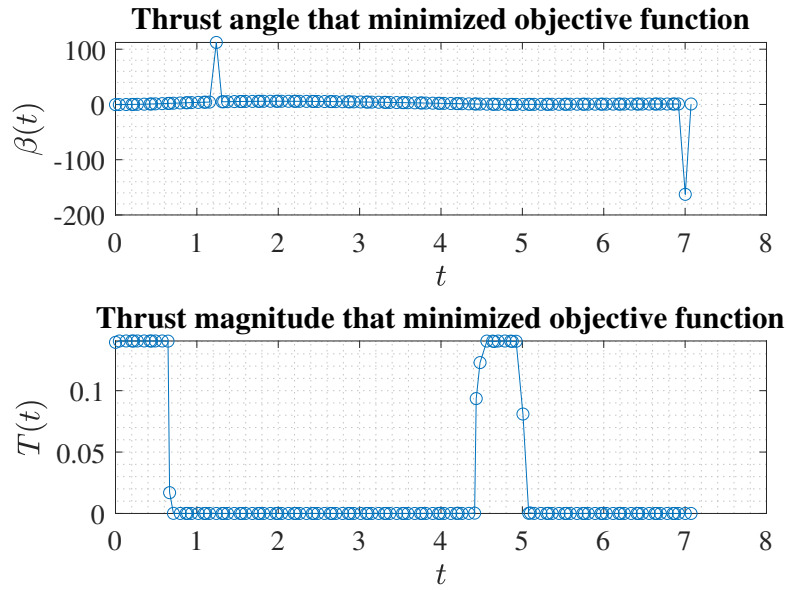


Figure 142: Control that maximized terminal mass ($N : 4$, $K : 32$)

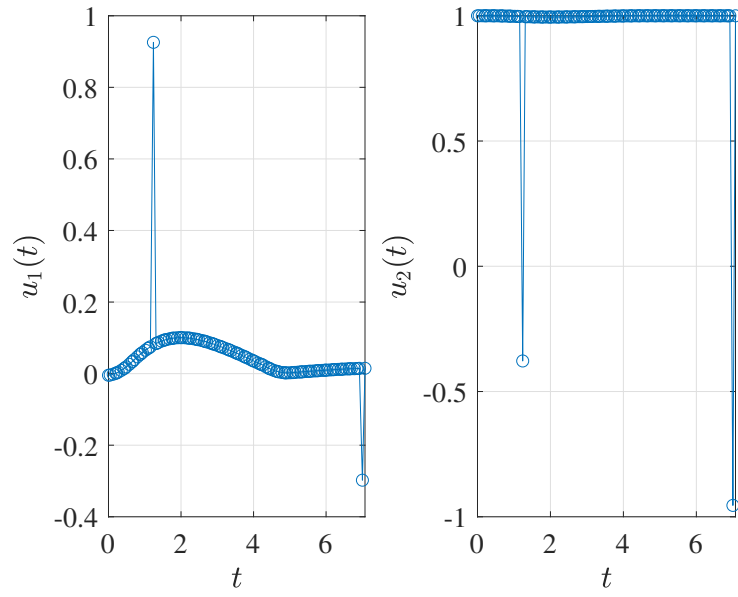


Figure 143: Path constrained control that maximized terminal mass ($N : 4, K : 32$)

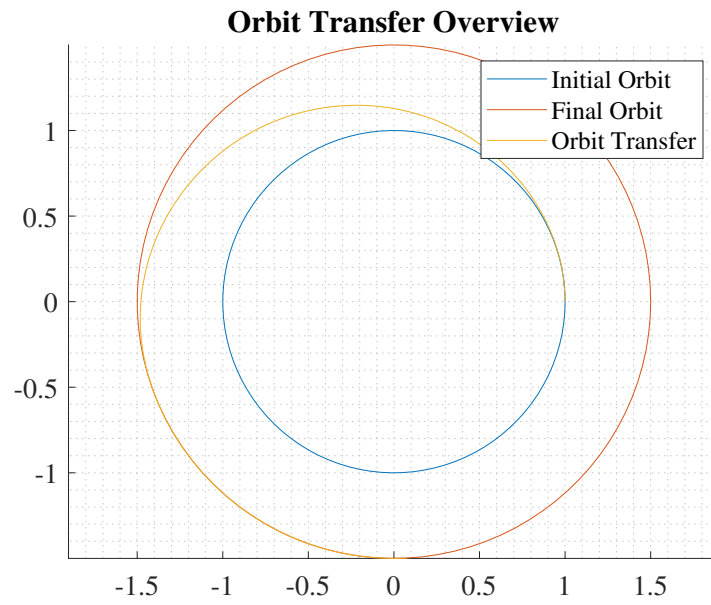


Figure 144: Trajectory from initial to final orbit ($N : 4, K : 32$)

4.5 Overall Analysis

5 Future Work

6 Appendix

Listing 1: orbitTransferMain.m

```
1 % ----- %
2 % Orbit-Transfer Problem %
3 % ----- %
4 % Solve the following optimal control problem: %
5 % Maximize t.f %
6 % subject to the differential equation constraints %
7 %  $dr/dt = v_r$  %
8 %  $d\theta/dt = v_\theta/r$  %
9 %  $dv_r/dt = v_\theta^2/r - \mu/r^2 + T*u_1/m$  %
10 %  $dv_\theta/dt = -v_r*v_\theta/r + T*u_2/m$  %
11 % the equality path constraint %
12 %  $u_1^2 + u_2^2 = 1$  %
13 % and the boundary conditions %
14 %  $r(0) = 1$  %
15 %  $\theta(0) = 0$  %
16 %  $v_r(0) = 0$  %
17 %  $v_\theta(0) = \sqrt{\mu/r(0)}$  %
18 %  $m(0) = 1$  %
19 %  $r(t.f) = 1.5$  %
20 %  $v_r(t.f) = 0$  %
21 %  $v_\theta(t.f) = \sqrt{\mu/r(t.f)}$  %
22
23 close all; clear all;
24 % ----- %
25 % BEGIN: DO NOT ALTER THE FOLLOWING LINES OF CODE!!! %
26 % ----- %
27 global igrid CONSTANTS psStuff nstates ncontrols npaths path_constraint maximize_mass
28 % ----- %
29 % END: DO NOT ALTER THE FOLLOWING LINES OF CODE!!! %
30 % ----- %
31 path = 'C:\Users\elias\Documents\UF_Classes\EML6934\Final.Project\EML6934_Final.Project';
32 addpath(genpath(path))
33
34 % save figures or create latex table?
35 save_figs = 1;
36 create_latex_table = 1;
37
38 % Set path constraint and objective function descision
39 path_constraint = 1; % is there a path constraint?
40 maximize_mass = 1; % maximize m(tf), else min tf
41
42 % Set polynomial degrees and intervals to loop through
43 n_list = [3 4];
44 k_list = [2 4 8 16 32];
45
46 % counter for table
47 count = 1;
48
49 for nldx = 1:numel(n_list)
50
51     for kldx = 1:numel(k_list)
52
53         % Set polynomial degree and number of intervals
54         N = n_list(nldx); % number of polynomial degree
55         numIntervals = k_list(kldx); % number of intervals
56     end
57 end
```

```

57 % set gloabl constants
58 CONSTANTS.MU = 1;
59 CONSTANTS.m0 = 1;
60 CONSTANTS.ve = 1.8658344;
61 % set number of states
62 nstates = 5;
63
64 % set number of controls and paths
65 if path_constraint
66     ncontrols = 3;
67     npaths = 1;
68 else
69     ncontrols = 2;
70     npaths = 0;
71 end
72
73 % Bounds on State and Control
74 % thetad and m are free
75 r0 = 1; theta0 = 0; vr0 = 0; vtheta0 = 1; m0 = 1;
76 rf = 1.5; vrf = 0; vthetad = sqrt(1/rf);
77
78 rmin = 1; rmax = 3;
79 thetamin = 0; thetamax = 4*pi;
80 vrmin = -10; vrmax = 10;
81 vthetamin = -10; vthetamax = 10;
82 mmin = 0.1; mmax = m0;
83 t0min = 0; t0max = 0;
84 tfmin = 0; tfmax = 100;
85 if path_constraint
86     u1min = -10; u1max = 10; % sin(beta)
87     u2min = -10; u2max = 10; % cos(beta)
88     u3min = 0; u3max = 0.1405; % thrust
89 else
90     u1min = -4*pi; u1max = 4*pi; % beta
91     u2min = 0; u2max = 0; % empty
92     u3min = 0; u3max = 0.1405; % thrust
93 end
94
95 % Create Leguandre Gauss Points
96 meshPoints = linspace(-1,1,numIntervals+1).';
97 polyDegrees = N*ones(numIntervals,1);
98 [tau,w,D] = lgrPS(meshPoints,polyDegrees);
99 psStuff.tau = tau; psStuff.w = w; psStuff.D = D; NLGR = length(w);
100
101 % Set the bounds on the NLP variables.
102 zrmin = rmin*ones(length(tau),1);
103 zrmax = rmax*ones(length(tau),1);
104 zrmin(1) = r0; zrmax(1) = r0;
105 zrmin(end) = rf; zrmax(end) = rf;
106
107 zthetamin = thetamin*ones(length(tau),1);
108 zthetamax = thetamax*ones(length(tau),1);
109 zthetamin(1) = theta0; zthetamax(1) = theta0;
110
111 zvrmin = vrmin*ones(length(tau),1);
112 zvrmax = vrmax*ones(length(tau),1);
113 zvrmin(1) = vr0; zvrmax(1) = vr0;
114 zvrmin(end) = vrf; zvrmax(end) = vrf;
115
116 zvthetamin = vthetamin*ones(length(tau),1);
117 zvthetamax = vthetamax*ones(length(tau),1);
118 zvthetamin(1) = vtheta0; zvthetamax(1) = vtheta0;
119 zvthetamin(end) = vthetad; zvthetamax(end) = vthetad;
120
121 zmmin = mmin*ones(length(tau),1);
122 zmmax = mmax*ones(length(tau),1);
123 zmmin(1) = m0; zmmax(1) = m0;
124

```



```

125 zu1min = u1min*ones(length(tau)-1,1);
126 zu1max = u1max*ones(length(tau)-1,1);
127
128 zu2min = u2min*ones(length(tau)-1,1);
129 zu2max = u2max*ones(length(tau)-1,1);
130
131 zu3min = u3min*ones(length(tau)-1,1);
132 zu3max = u3max*ones(length(tau)-1,1);
133
134 if path_constraint
135     zmin = [zrmin; zthetamin; zvrmin; zvthetamin; zmmmin; zu1min; zu2min; zu3min; t0min; tfmin];
136     zmax = [zrmax; zthetamax; zvrmax; zvthetamax; zmmmax; zu1max; zu2max; zu3max; t0max; tfmax];
137 else
138     zmin = [zrmin; zthetamin; zvrmin; zvthetamin; zmmmin; zu1min; zu3min; t0min; tfmin];
139     zmax = [zrmax; zthetamax; zvrmax; zvthetamax; zmmmax; zu1max; zu3max; t0max; tfmax];
140 end
141 % Set the bounds on the NLP constraints
142 % There are NSTATES sets of defect constraints.
143 defectMin = zeros(nstates*(length(tau)-1),1);
144 defectMax = zeros(nstates*(length(tau)-1),1);
145 if path_constraint
146     % There is a path constraint
147     pathMin = ones(length(tau)-1,1); pathMax = ones(length(tau)-1,1);
148 else
149     % No path constraint
150     pathMin = []; pathMax = [];
151 end
152 % I dont believe there is nonlinear event constraint
153 eventMin = []; eventMax = [];
154 objMin = -inf; objMax = inf;
155 Fmin = [objMin; defectMin; pathMin; eventMin];
156 Fmax = [objMax; defectMax; pathMax; eventMax];
157
158 % Supply an initial guess
159 rguess = linspace(r0,rf,NLGR+1).';
160 thetaguess = linspace(theta0,theta0,NLGR+1).';
161 vrguess = linspace(vr0,vrf,NLGR+1).';
162 vthetaguess = linspace(vtheta0,vtheta0,NLGR+1).';
163 mguess = linspace(m0,m0,NLGR+1).';
164 u3guess = linspace(0,0,NLGR).';
165 t0guess = 0;
166 tfguess = 3.5;
167
168 if path_constraint
169     u1guess = linspace(1,1,NLGR).';
170     u2guess = linspace(0,0,NLGR).';
171     z0 = [rguess;thetaguess;vrguess;vthetaguess;mguess;u1guess;u2guess;u3guess;t0guess;tfguess];
172 else
173     u1guess = linspace(0,0,NLGR).';
174     z0 = [rguess;thetaguess;vrguess;vthetaguess;mguess;u1guess;u3guess;t0guess;tfguess];
175 end
176
177 %-----%
178 % Generate derivatives and sparsity pattern using Adigator %
179 %-----%
180 % - Constraint Function Derivatives
181 xsize = size(z0);
182 x = adigatorCreateDerivInput(xsize,'z0');
183 output = adigatorGenJacFile('orbitTransferFun',{x});
184 S_jac = output.JacobianStructure;
185 [iGfun,jGvar] = find(S_jac);
186
187 % - Objective Function Derivatives
188 xsize = size(z0);
189 x = adigatorCreateDerivInput(xsize,'z0');
190 output = adigatorGenJacFile('orbitTransferObj',{x});
191 grd_structure = output.JacobianStructure;
192

```

```

193 %-----%
194 % set IPOPT callback functions
195 %-----%
196 funcs.objective = @(Z)orbitTransferObj(Z);
197 funcs.gradient = @(Z)orbitTransferGrd(Z);
198 funcs.constraints = @(Z)orbitTransferCon(Z);
199 funcs.jacobian = @(Z)orbitTransferJac(Z);
200 funcs.jacobianstructure = @(S)orbitTransferJacPat(S.jac);
201 options.ipopt.hessian_approximation = 'limited-memory';
202
203 %-----%
204 % Set IPOPT Options %
205 %-----%
206 options.ipopt.tol = 1e-8;
207 options.ipopt.linear_solver = 'ma57'; %'mumps';
208 options.ipopt.max_iter = 8000;
209 options.ipopt.mu_strategy = 'adaptive';
210 options.ipopt.ma57_automatic_scaling = 'yes';
211 options.ipopt.print_user_options = 'yes';
212 options.ipopt.output_file = ['orbitTransfer','IPOPTinfo.txt']; % print output file
213 options.ipopt.print_level = 5; % set print level default
214
215 options.lb = zmin; % Lower bound on the variables.
216 options.ub = zmax; % Upper bound on the variables.
217 options.cl = Fmin; % Lower bounds on the constraint functions.
218 options.cu = Fmax; % Upper bounds on the constraint functions.
219
220 %-----%
221 % Call IPOPT
222 %-----%
223 [z, info] = ipopt(z0,funcs,options);
224
225 %-----%
226 % extract lagrange multipliers from ipopt output, info
227 %-----%
228 Fmul = info.lambda;
229
230 % Extract the state and control from the decision vector z.
231 % Remember that the state is approximated at the LGR points
232 % plus the final point, while the control is only approximated
233 % at only the LGR points.
234 r = z(1:NLGR+1);
235 theta = z(NLGR+2:2*(NLGR+1));
236 vr = z(2*(NLGR+1)+1:3*(NLGR+1));
237 vtheta = z(3*(NLGR+1)+1:4*(NLGR+1));
238 m = z(4*(NLGR+1)+1:5*(NLGR+1));
239 u1 = z(5*(NLGR+1)+1:5*(NLGR+1)+NLGR);
240 if path_constraint
241     u2 = z(5*(NLGR+1)+NLGR+1:5*(NLGR+1)+2*NLGR);
242     u3 = z(5*(NLGR+1)+2*NLGR+1:5*(NLGR+1)+3*NLGR);
243     beta = atan2(u1,u2);
244     beta = unwrap(beta)*180/pi;
245 else
246     u3 = z(5*(NLGR+1)+NLGR+1:5*(NLGR+1)+2*NLGR);
247     beta = unwrap(u1)*180/pi;
248 end
249 t0 = z(end-1);
250 tf = z(end);
251 t = (tf-t0)*(tau+1)/2+t0;
252 tLGR = t(1:end-1);
253 %-----%
254 % Extract the Lagrange multipliers corresponding %
255 % the defect constraints. %
256 %-----%
257 multipliersDefects = Fmul(2:nstates*NLGR+1);
258 multipliersDefects = reshape(multipliersDefects,NLGR,nstates);
259 %-----%
260 % Compute the costates at the LGR points via transformation %

```

```

261 %------%
262 costateLGR = inv(diag(w))*multipliersDefects;
263 %------%
264 % Compute the costate at the tau=+1 via transformation %
265 %------%
266 costateF = D(:,end).'*multipliersDefects;
267 %------%
268 % Now assemble the costates into a single matrix %
269 %------%
270 costate = [costateLGR; costateF];
271 lamr = costate(:,1); lamtheta = costate(:,2);
272 lamvr = costate(:,3); lamvtheta = costate(:,4);
273
274 %------%
275 % Get planer coordinates
276 %------%
277 x_transfer = r.*cos(theta);
278 y_transfer = r.*sin(theta);
279 theta_orbit = linspace(0,2*pi,100);
280 x_orbit_1 = r0*cos(theta_orbit);
281 y_orbit_1 = r0*sin(theta_orbit);
282 x_orbit_2 = rf*cos(theta_orbit);
283 y_orbit_2 = rf*sin(theta_orbit);
284
285 %------%
286 % Plot Results
287 %------%
288 close all
289
290 fig_cntrl = figure;
291 h1 = subplot(2,1,1);
292 plot(tLGR,beta,'-o');
293 ylabel('$\beta(t)$','Interpreter','LaTeX');
294 xlabel('$t$','Interpreter','LaTeX');
295 set(gca,'FontName','Times','FontSize',14);
296 set(gcf,'color','white')
297 title('Thrust angle that minimized objective function')
298 grid minor;
299
300 h2 = subplot(2,1,2);
301 plot(tLGR,u3,'-o');
302 set(gca,'FontName','Times','FontSize',14);
303 set(gcf,'color','white')
304 ylabel('$T(t)$','Interpreter','LaTeX');
305 title('Thrust magnitude that minimized objective function')
306 xlabel('$t$','Interpreter','LaTeX')
307 grid minor
308 linkaxes([h1 h2],'x')
309
310 fig_states = figure; hold on; grid minor
311 plot(t,r,'-o'); plot(t,theta,'-o'); plot(t,vr,'-o');
312 plot(t,vtheta,'-o'); plot(t,m,'-o');
313 xlabel('t','Interpreter','LaTeX')
314 legend('r(t)', '$\theta(t)$', 'vr(t)', '$\theta(t)$', 'm(t)','Interpreter','LaTeX')
315 set(gcf,'color','white')
316 set(gca,'FontName','Times','fontSize',14)
317 title('States that minimized objective function')
318
319 fig_orbit = figure; hold on; grid minor
320 plot(x_orbit_1,y_orbit_1)
321 plot(x_orbit_2,y_orbit_2)
322 plot(x_transfer,y_transfer)
323 set(gca,'FontName','Times','fontSize',14)
324 set(gcf,'color','white')
325 legend('Initial Orbit','Final Orbit','Orbit Transfer')
326 axis equal
327 title('Orbit Transfer Overview')
328

```

```

329     if save_figs
330         if maximize_mass
331             str_obj = 'mf';
332         else
333             str_obj = 'tf';
334         end
335         plot_str = sprintf('_N%d_K%d_C%d_',N,numIntervals,ncontrols);
336         str_list = {'control','states','orbit'};
337         for idx = 1:numel(str_list)
338             name = [str_list{idx} plot_str str_obj];
339             print(ffigure(idx),name,'-depsc')
340         end
341     end
342     if path_constraint
343         fig_path = figure;
344         subplot(1,2,1);
345         plot(tLGR,u1,'-o');
346         xl = xlabel('$t$', 'Interpreter', 'LaTeX');
347         yl = ylabel('$u_1(t)$', 'Interpreter', 'LaTeX');
348         set(xl, 'FontSize', 14);
349         set(yl, 'FontSize', 14);
350         set(gca, 'FontName', 'Times', 'FontSize', 14);
351         grid on;
352
353         subplot(1,2,2);
354         plot(tLGR,u2,'-o');
355         xl = xlabel('$t$', 'Interpreter', 'LaTeX');
356         yl = ylabel('$u_2(t)$', 'Interpreter', 'LaTeX');
357         set(xl, 'FontSize', 14);
358         set(yl, 'FontSize', 14);
359         set(gca, 'FontName', 'Times', 'FontSize', 14);
360         set(gcf, 'color', 'white')
361         grid on;
362
363         if save_figs
364             str_path = ['path' plot_str str_obj];
365             print(fig_path, str_path, '-depsc')
366         end
367     end
368
369     % figure(1);
370     % subplot(2,2,1);
371     % plot(t,r,'-o');
372     % xl = xlabel('$t$', 'Interpreter', 'LaTeX');
373     % yl = ylabel('$r(t)$', 'Interpreter', 'LaTeX');
374     % set(xl, 'FontSize', 14);
375     % set(yl, 'FontSize', 14);
376     % set(gca, 'FontName', 'Times', 'FontSize', 14);
377     % set(gcf, 'color', 'white')
378     % grid on;
379     %
380     % subplot(2,2,2);
381     % plot(t,theta,'-o');
382     % xl = xlabel('$t$', 'Interpreter', 'LaTeX');
383     % yl = ylabel('$\theta(t)$', 'Interpreter', 'LaTeX');
384     % set(xl, 'FontSize', 14);
385     % set(yl, 'FontSize', 14);
386     % set(gca, 'FontName', 'Times', 'FontSize', 14);
387     % set(gcf, 'color', 'white')
388     % grid on;
389     %
390     % subplot(2,2,3);
391     % plot(t,vr,'-o');
392     % xl = xlabel('$t$', 'Interpreter', 'LaTeX');
393     % yl = ylabel('$v_r(t)$', 'Interpreter', 'LaTeX');
394     % set(xl, 'FontSize', 14);
395     % set(yl, 'FontSize', 14);
396     % set(gca, 'FontName', 'Times', 'FontSize', 14);

```

```

397 % set(gcf,'color','white')
398 % grid on;
399 %
400 % subplot(2,2,4);
401 % plot(t,vtheta,'-o');
402 % xl = xlabel('$t$', 'Interpreter', 'LaTeX');
403 % yl = ylabel('$v_{\theta}(t)$', 'Interpreter', 'LaTeX');
404 % set(xl,'FontSize',14);
405 % set(yl,'FontSize',14);
406 % set(gca,'FontName','Times','FontSize',14);
407 % set(gcf,'color','white')
408 % grid on;
409
410 % figure;
411 % subplot(2,2,1);
412 % plot(t,lamr,'-o');
413 % xl = xlabel('$t$', 'Interpreter', 'LaTeX');
414 % yl = ylabel('$\lambda_{r}(t)$', 'Interpreter', 'LaTeX');
415 % set(xl,'FontSize',14);
416 % set(yl,'FontSize',14);
417 % set(gca,'FontName','Times','FontSize',14);
418 % set(gcf,'color','white')
419 % grid on;
420 %
421 % subplot(2,2,2);
422 % plot(t,lamtheta,'-o');
423 % xl = xlabel('$t$', 'Interpreter', 'LaTeX');
424 % yl = ylabel('$\lambda_{\theta}(t)$', 'Interpreter', 'LaTeX');
425 % set(xl,'FontSize',14);
426 % set(yl,'FontSize',14);
427 % set(gca,'FontName','Times','FontSize',14);
428 % set(gcf,'color','white')
429 % grid on;
430 %
431 % subplot(2,2,3);
432 % plot(t,lamvr,'-o');
433 % xl = xlabel('$t$', 'Interpreter', 'LaTeX');
434 % yl = ylabel('$\lambda_{v_r}(t)$', 'Interpreter', 'LaTeX');
435 % set(xl,'FontSize',14);
436 % set(yl,'FontSize',14);
437 % set(gca,'FontName','Times','FontSize',14);
438 % set(gcf,'color','white')
439 % grid on;
440 %
441 % subplot(2,2,4);
442 % plot(t,lamvtheta,'-o');
443 % xl = xlabel('$t$', 'Interpreter', 'LaTeX');
444 % yl = ylabel('$\lambda_{v_{\theta}}(t)$', 'Interpreter', 'LaTeX');
445 % set(xl,'FontSize',14);
446 % set(yl,'FontSize',14);
447 % set(gca,'FontName','Times','FontSize',14);
448 % set(gcf,'color','white')
449 % grid on;
450
451 %-----%
452 % save results %
453 %-----%
454 degree(count,1) = N;
455 intervals(count,1) = numIntervals;
456 iterations(count,1) = info.iter;
457 cpu_time(count,1) = info.cpu;
458 final_time(count,1) = tf;
459 final_mass(count,1) = m(end);
460 solved_info(count,1) = info.status;
461 count = count + 1;
462 end
463 end
464 table_mat = horzcat(degree,intervals,iterations,cpu_time,final_time,final_mass,solved_info);

```

```

465 T = array2table(table_mat);
466 VarNames = {'Degree','Intervals','Iterations','CPU Time','tf','mf','Solved_Status'};
467 T.Properties.VariableNames = VarNames;
468 if create_latex_table
469     str_table = sprintf('table_C%d_',ncontrols);
470     table2latex(T,[str_table str_obj])
471 end

```

Listing 2: orbitTransferFun.m

```

1  function C = orbitTransferFun(z)
2
3  %------%
4  % Objective and constraint functions for the orbit-raising %
5  % problem. This function is designed to be used with the NLP %
6  % solver SNOPT. %
7  %------%
8  % DO NOT FOR ANY REASON ALTER THE LINE OF CODE BELOW! %
9  global psStuff nstates ncontrols npaths CONSTANTS path_constraint maximize_mass%
10 % DO NOT FOR ANY REASON ALTER THE LINE OF CODE ABOVE! %
11 %------%
12
13 %------%
14 % Extract the constants used in the problem. %
15 %------%
16 mu = CONSTANTS.MU;
17 ve = CONSTANTS.ve;
18 %------%
19 % Radau pseudospectral method quantities required: %
20 % - Differentiation matrix (psStuff.D) %
21 % - Legendre-Gauss-Radau weights (psStuff.w) %
22 % - Legendre-Gauss-Radau points (psStuff.tau) %
23 %------%
24 D = psStuff.D; tau = psStuff.tau; w = psStuff.w;
25
26 %------%
27 % Decompose the NLP decision vector into pieces containing %
28 % - the state %
29 % - the control %
30 % - the initial time %
31 % - the final time %
32 %------%
33 N = length(tau)-1;
34 stateIndices = 1:nstates*(N+1);
35 controlIndices = (nstates*(N+1)+1):(nstates*(N+1)+ncontrols*N);
36 t0Index = controlIndices(end)+1;
37 tfIndex = t0Index+1;
38 stateVector = z(stateIndices);
39 controlVector = z(controlIndices);
40 t0 = z(t0Index);
41 tf = z(tfIndex);
42
43 %------%
44 % Reshape the state and control parts of the NLP decision vector %
45 % to matrices of sizes (N+1) by nstates and (N+1) by ncontrols, %
46 % respectively. The state is approximated at the N LGR points %
47 % plus the final point. Thus, each column of the state vector is %
48 % length N+1. The LEFT-HAND SIDE of the defect constraints, D*X, %
49 % uses the state at all of the points (N LGR points plus final %
50 % point). The RIGHT-HAND SIDE of the defect constraints, %
51 % (tf-t0)F/2, uses the state and control at only the LGR points. %
52 % Thus, it is necessary to extract the state approximations at %
53 % only the N LGR points. Finally, in the Radau pseudospectral %
54 % method, the control is approximated at only the N LGR points. %
55 %------%
56 statePlusEnd = reshape(stateVector,N+1,nstates);
57 stateLGR = statePlusEnd(1:end-1,:);

```

```

58 control = reshape(controlVector,N,ncontrols);
59
60 %-----%
61 % Identify the components of the state column-wise from stateLGR. %
62 %-----%
63 r = stateLGR(:,1);
64 theta = stateLGR(:,1);
65 vr = stateLGR(:,3);
66 vtheta = stateLGR(:,4);
67 m = stateLGR(:,5);
68 if path_constraint
69     u1 = control(:,1);
70     u2 = control(:,2);
71     u3 = control(:,3);
72 else
73     u1 = control(:,1);
74     u2 = 0;
75     u3 = control(:,2);
76 end
77 %-----%
78 % The quantity STATEF is the value of the state at the final %
79 % time, tf, which corresponds to the state at  $\tau=1$ . %
80 %-----%
81 stateF = statePlusEnd(end,:);
82 %-----%
83 % The orbit-raising problem contains one nonlinear boundary %
84 % condition  $\sqrt{\mu/r(t.f)} - v_{\theta}(t.f) = 0$ . Because  $r(t)$  %
85 % and  $v_{\theta}(t)$  are the first and fourth components of the %
86 % state, it is necessary to extract stateF(1) and stateF(4) in %
87 % order to compute this boundary condition function. %
88 %-----%
89 rF = stateF(1);
90 vthetaF = stateF(4);
91 %
92 % a = T./m;
93
94 %-----%
95 % Compute the right-hand side of the differential equations at %
96 % the N LGR points. Each component of the right-hand side is %
97 % stored as a column vector of length N, that is each column has %
98 % the form %
99 % [ f.i(x.1,u.1,t.1) ] %
100 % [ f.i(x.2,u.2,t.2) ] %
101 % . %
102 % . %
103 % . %
104 % [ f.i(x.N,u.N,t.N) ] %
105 % where "i" is the right-hand side of the ith component of the %
106 % vector field f. It is noted that in MATLABB the calculation of %
107 % the right-hand side is vectorized. %
108 %-----%
109 rdot = vr;
110 thetadot = vtheta./r;
111 mdot = -u3./ve;
112 if path_constraint
113     vrdot = vtheta.^2./r - mu./r.^2 + u3.*u1./m;
114     vthetadot = -vtheta.*vr./r + u3.*u2./m;
115 else
116     vrdot = vtheta.^2./r - mu./r.^2 + u3.*sin(u1)./m;
117     vthetadot = -vtheta.*vr./r + u3.*cos(u1)./m;
118 end
119
120 diffeqRHS = [rdot, thetadot, vrdot, vthetadot, mdot];
121
122 %-----%
123 % Compute the left-hand side of the defect constraints, recalling %
124 % that the left-hand side is computed using the state at the LGR %
125 % points PLUS the final point. %

```

```

126 %------%
127 diffeqLHS = D*statePlusEnd;
128
129 %------%
130 % Construct the defect constraints at the N LGR points. %
131 % Remember that the right-hand side needs to be scaled by the %
132 % factor (tf-t0)/2 because the rate of change of the state is %
133 % being taken with respect to  $\tau \in [-1, +1]$ . Thus, we have %
134 %  $\frac{dt}{t} \frac{dau}{dau} = (tf-t0)/2$ . %
135 %------%
136 defects = diffeqLHS-(tf-t0)*diffeqRHS/2;
137
138 %------%
139 % Construct the path constraints at the N LGR points. %
140 % Reshape the path constraints into a column vector. %
141 %------%
142 if path_constraint
143     paths = u1.^2+u2.^2;
144     paths = reshape(paths,N*npaths,1);
145 else
146     paths = [];
147 end
148 % paths = u1.^2+u2.^2;
149
150 %------%
151 % Reshape the defect constraints into a column vector. %
152 %------%
153 defects = reshape(defects,N*nstates,1);
154
155 %------%
156 % Construct the objective function plus constraint vector. %
157 %------%
158 if maximize_mass
159     m = statePlusEnd(:,5);
160     J = -m(end);
161 else
162     J = tf;
163 end
164 C = [J;defects;paths];

```

Listing 3: orbitTransferObj.m

```

1 function obj = orbitTransferObj(z)
2 % Computes the objective function of the problem
3
4 global psStuff nstates ncontrols maximize_mass
5
6 %------%
7 % Extract the constants used in the problem. %
8 %------%
9 % MU = CONSTANTS.MU; mdot = CONSTANTS.mdot; T = CONSTANTS.T;
10
11 %------%
12 % Radau pseudospectral method quantities required: %
13 % - Differentiation matrix (psStuff.D) %
14 % - Legendre-Gauss-Radau weights (psStuff.w) %
15 % - Legendre-Gauss-Radau points (psStuff.tau) %
16 %------%
17 D = psStuff.D; tau = psStuff.tau; w = psStuff.w;
18
19 %------%
20 % Decompose the NLP decision vector into pieces containing %
21 % - the state %
22 % - the control %
23 % - the initial time %
24 % - the final time %
25 %------%

```



```

26 N = length(tau)-1;
27 stateIndices = 1:nstates*(N+1);
28 controlIndices = (nstates*(N+1)+1):(nstates*(N+1)+ncontrols*N);
29 t0Index = controlIndices(end)+1;
30 tfIndex = t0Index+1;
31 stateVector = z(stateIndices);
32 % controlVector = z(controlIndices);
33 % t0 = z(t0Index);
34 tf = z(tfIndex);
35
36 %-----%
37 % Reshape the state and control parts of the NLP decision vector %
38 % to matrices of sizes (N+1) by nstates and (N+1) by ncontrols, %
39 % respectively. The state is approximated at the N LGR points %
40 % plus the final point. Thus, each column of the state vector is %
41 % length N+1. The LEFT-HAND SIDE of the defect constraints, D*X, %
42 % uses the state at all of the points (N LGR points plus final %
43 % point). The RIGHT-HAND SIDE of the defect constraints, %
44 % (tf-t0)F/2, uses the state and control at only the LGR points. %
45 % Thus, it is necessary to extract the state approximations at %
46 % only the N LGR points. Finally, in the Radau pseudospectral %
47 % method, the control is approximated at only the N LGR points. %
48 %-----%
49 statePlusEnd = reshape(stateVector,N+1,nstates);
50 stateLGR = statePlusEnd(1:end-1,:);
51 % control = reshape(controlVector,N,ncontrols);
52
53 %-----%
54 % Identify the components of the state column-wise from stateLGR. %
55 %-----%
56 % r = stateLGR(:,1);
57 % theta = stateLGR(:,1);
58 % vr = stateLGR(:,3);
59 % vtheta = stateLGR(:,4);
60
61
62 % Cost Function
63 % minimizing time or maximizing mass
64 if maximize_mass
65     m = statePlusEnd(:,5);
66     J = -m(end);
67 else
68     J = tf;
69 end
70
71 obj = J;
72
73 end

```

Listing 4: orbitTransferCon.m

```

1 function constraints = orbitTransferCon(Z)
2 % computes the constraints
3
4 output = orbitTransferFun(Z);
5 constraints = output;
6
7 end

```

Listing 5: orbitTransferGrd.m

```

1 function grd = orbitTransferGrd(Z)
2 % computes the gradient
3
4 output = orbitTransferObj_Jac(Z);
5 grd = output;

```

```
6  
7 end
```

Listing 6: orbitTransferJac.m

```
1 function jac = orbitTransferJac(Z)  
2 % computes the jacobian  
3  
4 [jac,~] = orbitTransferFun_Jac(Z);  
5  
6 end
```

Listing 7: orbitTransferJacPat.m

```
1 function jacpat = orbitTransferJacPat(S_jac)  
2 % computes the jacobian structure  
3  
4 jacpat = S_jac;  
5  
6 end
```
