Objective - The goal of this problem is to formulate and solve an optimal control problem arising in spacecraft dynamics using the given knowledge of the problem.

Considering a spacecraft, modeled as a point mass *P* of mass *m*, moving relative to an inertial reference frame I. Furthermore, assume that the particle is moving in a plane that is itself fixed in I. Next, let $\{e_x, e_y, e_z\}$ be a right-handed orthonormal basis fixed in I.
Assuming now that the position of the spacecraft is denoted $r_{P/O}$, where *O* is fixed in I and denotes the location of the Sun (that is, the position of the spacecraft is measured relative to the Sun). Suppose now that $r_{P/O}$ is parameterized in terms of a basis $\{u_r, u_\theta, u_z\}$ where $\{u_r, u_\theta, u_z\}$ rotates about the e_z direction such that θ is the angle from e_x to u_r. Finally, assume that the distance from *O* to *P* is denoted r. A schematic of the geometry of the problem is given in Fig. 1.

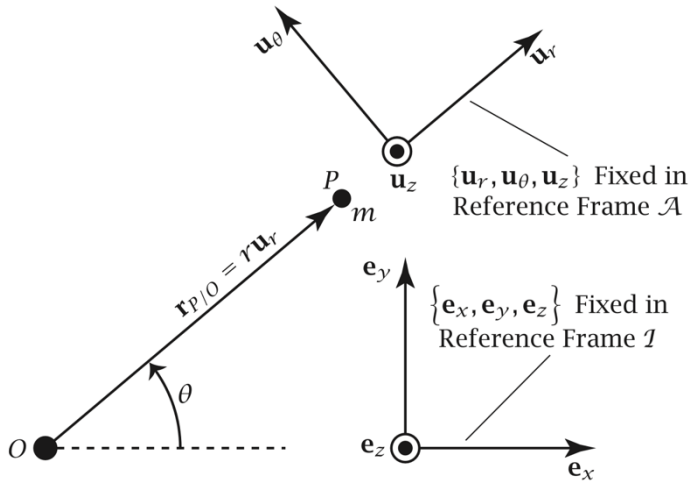


Figure 1: Schematic of particle moving in an inertially fixed plane.

Assuming now that the following two forces act on the spacecraft: (1) gravitational force and (2) thrust force. The gravitational force is given as

$$G = -m\mu \frac{r_{P/O}}{\left\| r_{P/O} \right\|^3}$$

while the thrust force is given as

$$T = Tw$$

where w is a unit vector that lies an angle β from the direction u_θ (in other words, β is the angle between u_θ and w).

Formulation of second-order differential equations

(a) Using Newton's second law for a particle

We know,

$$r_{P/O} = ru_r$$

Therefore, velocity is given by

$$^I v = \frac{^I d \vec{r}}{dt} = \frac{^A d \vec{r}}{dt} + \left(w_A^I \times r_{P/O}\right) = \frac{dr}{dt} u_r + (\dot{\theta} u_z + r u_r)$$

$$^I v = r u_r \dot{+} \dot{\theta} r u_\theta$$

And acceleration can be calculated as

$$^I a = \frac{^I d \vec{v}}{dt} = \frac{^A d \vec{v}}{dt} + (w_A^I \times {}^I v)$$

$$^I a = \frac{d}{dx}\left(\dot{r} u_r + \dot{\theta} r u_\theta\right) + \left(\dot{\theta} u_z \times \left(\dot{r} u_r + \dot{\theta} r u_\theta\right)\right)$$

$$^I a = \ddot{r} u_r + \left(\dot{\theta}\dot{r} u_\theta + r\ddot{\theta} u_\theta\right) + (\dot{\theta}\dot{r} r_\theta - \dot{\theta}^2 r u_r)$$

$$^I a = \left(\ddot{r} + \dot{\theta}^2 r\right) u_r + (r\ddot{\theta} + 2\dot{\theta}\dot{r}) u_\theta$$

According to Newton's second law,

$$F = ma$$

We know that the two forces acting on the spacecraft are the gravitation force along u_r and thrust force along w.

$$F = G + T = \frac{-m\mu(r u_r)}{r^2} + Tw$$

Unit vector w can be decomposed in terms of a basis $\{u_r, u_\theta, u_z\}$ as,

$$Tw = T\cos\beta\, u_\theta + T\sin\beta\, u_r$$

$$F = \frac{-m\mu u_r}{r^2} + T\cos\beta\, u_\theta + T\sin\beta\, u_r$$

$$F = \left(T\sin\beta - \frac{m\mu}{r^2}\right) u_r + T\cos\beta\, u_\theta$$

We also know that,

$$F = m\, {}^I a = m\left(\ddot{r} - \dot{\theta}^2 r\right) u_r + m(r\ddot{\theta} + 2\dot{\theta}\dot{r}) u_\theta$$

This implies,

$$\left(T\sin\beta - \frac{m\mu}{r^2}\right) u_r + T\cos\beta\, u_\theta = m\left(\ddot{r} - \dot{\theta}^2 r\right) u_r + m(r\ddot{\theta} + 2\dot{\theta}\dot{r}) u_\theta$$

Projecting this equation on u_r basis we get,

$$F.u_r = m\left(\ddot{r} - \dot{\theta}^2 r\right) = \left(T\sin\beta - \frac{m\mu}{r^2}\right)$$

$$\left(m\ddot{r} - m\dot{\theta}^2 r\right) - \left(T\sin\beta - \frac{m\mu}{r^2}\right) = 0$$

Projecting the above equation on u_θ we get,

$$F.u_\theta = m\left(r\ddot{\theta} + 2\dot{\theta}\dot{r}\right) = T\cos\beta$$

$$mr\ddot{\theta} + 2m\dot{\theta}\dot{r} - T\cos\beta = 0$$

Rewriting the two second-order differential equations into a system of four first-order equations. By making the following substitution,

$$\dot{r} = v_r$$

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

From the first second-order differential equation,

$$\left(m\ddot{r} - m\dot{\theta}^2 r\right) - \left(T\sin\beta - \frac{m\mu}{r^2}\right) = 0$$

$$m\dot{v}_r - m\dot{\theta}^2 = T\sin\beta - \frac{m\mu}{r^2}$$

$$m\dot{v}_r - \frac{m{v_\theta}^2}{r} = T\sin\beta - \frac{m\mu}{r^2}$$

$$m\dot{v}_r - \frac{m{v_\theta}^2}{r} - T\sin\beta + \frac{m\mu}{r^2} = 0$$

$$\dot{v}_r - \frac{{v_\theta}^2}{r} - \frac{T}{m}\sin\beta + \frac{\mu}{r^2} = 0$$

Using the second second-order differential equation,

$$mr\ddot{\theta} + 2m\dot{\theta}\dot{r} - T\cos\beta = 0$$

We know that,

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

Differentiating this equation we get,

$$\frac{d(r\dot{\theta})}{dx} = \dot{v}_\theta = r\ddot{\theta} + \dot{\theta}\dot{r}$$

And

$$\dot{\theta}\dot{r} = \frac{v_r v_\theta}{r}$$

this implies that,

$$m(\dot{v}_\theta) + m\left(\frac{v_r v_\theta}{r}\right) - T\cos\beta = 0$$

$$\dot{v}_\theta - \frac{T\cos\beta}{m} + \frac{v_r v_\theta}{r} = 0$$

The four highlighted equations represent the four first order differential equations derived using Newton's second law for a particle.

(b) Using Lagrange's equations

We know that the Lagrange's equation is given by

$$L = T - U$$

Where T is the kinetic energy and V is the potential energy of the system.
The kinetic energy is given by,

$$T = \frac{m\,v\,v}{2} = \frac{m}{2}\left(\dot{ru_r} + \dot{\theta}ru_\theta\right)\left(\dot{ru_r} + \dot{\theta}ru_\theta\right)$$

$$T = \frac{m}{2}\left(\dot{r}^2 + \left(\dot{\theta}r\right)^2\right) = \frac{m}{2}\left(\dot{r}^2 + \dot{\theta}^2\dot{r}^2\right)$$

$$T = \frac{m}{2}\left(\dot{r}^2 + \dot{\theta}^2\dot{r}^2\right)$$

And it is known that the generalized force associated G defined as a given is derivable form the scalar potential function U given as,

$$U = -\frac{m\mu}{r}$$

Hence, the Lagrange's equation can be written in terms of the generalized coordinates r and θ as follows,

$$L = T - U = \frac{m}{2}\left(\dot{r}^2 + \dot{\theta}^2 r^2\right) + \frac{m\mu}{r}$$

$$L = \frac{m\dot{r}^2}{2} + \frac{m\dot{\theta}^2 r^2}{2} + \frac{m\mu}{r}$$

We know, with respect to θ

$$\frac{d}{dt}\frac{dL}{d\dot{\theta}} - \frac{dL}{d\theta} = \acute{Q_\theta} = \grave{F}\frac{dr}{d\theta}$$

So,

$$\frac{dL}{d\dot{\theta}} = \frac{mr^2}{2}\left(2\dot{\theta}^2\right) = mr^2\dot{\theta}$$

Differentiating again we get,

$$\frac{d}{dt}\frac{dL}{d\dot{\theta}} = m\left(r^2\ddot{\theta} + \dot{\theta}(2r\dot{r})\right) = mr^2\ddot{\theta} + 2mr\dot{\theta}\dot{r}$$

On differentiating L with respect to θ we get,

$$\frac{dL}{d\theta} = 0$$

(Because L does not depend on θ)

$\grave{F}$ represents the force that cannot be estimated as a scalar potential function and is given by,

$$\grave{F} = Tw$$

So,

$$\grave{Q}_\theta = Tw\,\frac{du_r r}{d\theta}$$

We know

$$\frac{du_r r}{d\theta} = ru_\theta$$

So,

$$\grave{Q}_\theta = Twu_\theta = T\,r(\cos\beta\,u_\theta + \sin\beta\,u_r)u_\theta$$

$$\grave{Q}_\theta = Tr\cos\beta$$

On substitution we get,

$$\frac{d}{dt}\frac{dL}{d\dot{\theta}} - \frac{dL}{d\theta} = \acute{Q_\theta} = \grave{F}\frac{dr}{d\theta}$$

$$mr^2\ddot{\theta} + 2mr\dot{\theta}\dot{r} - 0 = Tr\cos\beta$$

$$mr\ddot{\theta} + 2m\dot{\theta}\dot{r} = T\cos\beta$$

$$\dot{v}_\theta + \frac{v_r v_\theta}{r} - \frac{T\cos\beta}{m} = 0$$

Lagrange's equation with respect to r,

$$\frac{d}{dt}\frac{dL}{d\dot{r}} - \frac{dL}{dr} = \acute{Q_\theta} = \grave{F}\frac{dr}{dr}$$

$$\frac{dL}{d\dot{r}} = \frac{2m\dot{r}}{2} = m\dot{r}$$

$$\frac{d}{dt}\frac{dL}{d\dot{r}} = m\ddot{r}$$

$$\frac{dL}{dr} = \frac{2m\dot{\theta}^2 r}{2} - \frac{m\mu}{r^2} = m\dot{\theta}^2 r - \frac{m\mu}{r^2}$$

Since,

$$\frac{dr}{dr} = \frac{dru_r}{dr} = u_r$$

$$\dot{Q}_\theta = Twu_r = T\, r(\cos\beta\, u_\theta + \sin\beta\, u_r)u_r$$
$$= T\sin\beta$$

On substitution we get,

$$\frac{d}{dt}\frac{dL}{d\dot{r}} - \frac{dL}{dr} = \acute{Q}_\theta$$

$$m\ddot{r} - \left(m\dot{\theta}^2 r - \frac{m\mu}{r^2}\right) = T\sin\beta$$

$$\dot{v}_r - \frac{{v_\theta}^2}{r} + \frac{\mu}{r^2} = T\sin\beta$$

$$\dot{v}_r - \frac{{v_\theta}^2}{r} + \frac{\mu}{r^2} - T\sin\beta = 0$$

Hence the equations,

$$\dot{r} = v_r$$

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

$$\dot{v}_r - \frac{{v_\theta}^2}{r} + \frac{\mu}{r^2} - \frac{T\sin\beta}{m} = 0$$

$$\dot{v}_\theta + \frac{v_r v_\theta}{r} - \frac{T\cos\beta}{m} = 0$$

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

Represent the five first order differential equations in the generalized coordinates r and θ with T and β as inputs to the system and μ being a parameter of the system. $\dot{m}$ represents the mass flow rate of the system where $v_e = 1.8758344$ is the exhaust speed of the engine.

(1.2) Formulation of the Optimal Control Problem

First, the spacecraft starts in a heliocentric circular orbit of radius $r_0 = 1$ and terminates in a heliocentric circular orbit of radius $r_f = 1.5$. It is assumed that the initial longitude (where θ denotes the longitude) is zero while the terminal longitude is free. Furthermore, the initial mass is unity (that is, $m_0 = 1$) while the terminal mass is free. The objective is to minimize the time taken to transfer the spacecraft from the initial orbit to the terminal orbit. (Or in other words to maximize the final mass as the thrust T is assumed to be 'ON' thought the process and hence maximizing final mass m_f will intern mean that we are minimizing final time). In order to minimize the total time required to place a space vehicle in a higher energy orbit the optimal control formulation leads to the following set of necessary conditions of optimality.

$$\dot{X} = f(t, X, u) \quad \text{(i)}$$

$$\dot{\lambda} = \frac{dH}{dX} = g(t, X, u, \lambda) \quad \text{(ii)}$$

$$\frac{dH}{du} = 0 = \varphi(t, X, u, \lambda) \quad \text{(iii)}$$

Where X and λ represents the 4x1 state and the co-state vectors respectively and u, is the thrust directional angle. Equation (i) and (ii) are differential equations, and equation (iii) is just an algebraic equation that leads to the necessary conditions for optimal control.

The set of 5 first order differential equations that represent equation (i) are,

$$\dot{r} = v_r \quad (1)$$

$$\dot{\theta} = \frac{v_\theta}{r} \quad (2)$$

$$\dot{v}_r - \frac{{v_\theta}^2}{r} + \frac{\mu}{r^2} - \frac{T \sin\beta}{m} = 0 \quad (3)$$

$$\dot{v}_\theta + \frac{v_r v_\theta}{r} - \frac{T \cos\beta}{m} = 0 \quad (4)$$

$$\dot{m} = -\frac{T}{v_e} \quad (5)$$

The below equations (5) to (6) represent the 5 first order differential equations of the co-states that represent equation (ii) are derived from the Hamiltonian as follows,

Formulation of the Hamiltonian

$$H = L + \lambda^T f$$

Where f is a function of the dynamics obtained from the five formulated first-order differential equations when represented in the form as shown below, where x represents the states.

$$\dot{x}(t) = f(x(t), u(t), t)$$

This implies, the Hamiltonian is equal to

$$H = -1 + \lambda_r v_r + \lambda_\theta \frac{v_\theta}{r} + \lambda_{vr}\left(\frac{T \sin\beta}{m} + \frac{{v_\theta}^2}{r} - \frac{\mu}{r^2}\right) + \lambda_{v\theta}\left(\frac{T}{m}\cos\beta - \frac{v_r v_\theta}{r}\right) - \frac{\lambda_m T}{v_e}$$

The necessary conditions for optimality are as follows,

$$\frac{dH}{du} = 0; \quad \frac{dH}{dx} = -\dot{\lambda}^T; \quad \frac{dH}{d\lambda} = \dot{x}^T$$

We know,

$$\frac{dH}{du} = \frac{dH}{d\beta} = \frac{\lambda_{vr} T \cos\beta}{m} - \frac{\lambda_{v\theta} T \sin\beta}{m} = 0$$

$$\lambda_{vr} \cos\beta = \lambda_{v\theta} \sin\beta$$

$$\tan\beta = \frac{\lambda_{vr}}{\lambda_{v\theta}}$$

Using $\frac{dH}{dx} = -\dot{\lambda}^T$ we get,

(With respect to r)

$$\frac{dH}{dr} = -\dot{\lambda_r}^T = -\frac{\lambda_\theta v_\theta}{r^2} - \frac{\lambda_{vr} {v_\theta}^2}{r^2} + \frac{2\lambda_{vr}\mu}{r^3} + \frac{\lambda_{v\theta} v_r v_\theta}{r^2}$$

$$\dot{\lambda_r}^T = \frac{\lambda_\theta v_\theta}{r^2} + \frac{\lambda_{vr} {v_\theta}^2}{r^2} - \frac{2\lambda_{vr}\mu}{r^3} - \frac{\lambda_{v\theta} v_r v_\theta}{r^2}$$

(With respect to θ)

$$\frac{dH}{d\theta} = -\dot{\lambda_\theta}^T = 0$$

(With respect to v_r)

$$\frac{dH}{dv_r} = \dot{\lambda_{vr}}^T = \frac{\lambda_{v\theta} v_\theta}{r} - \lambda_r$$

(With respect to v_θ)

$$\frac{dH}{dv_\theta} = \dot{\lambda_{v\theta}}^T = -\frac{\lambda_\theta}{r} - \frac{2\lambda_\theta \lambda_{vr}}{r} + \frac{\lambda_{v\theta} v_r}{r}$$

(With respect to m)

$$\frac{dH}{dm} = \dot{\lambda_m}^T = \frac{T\cos\beta\,\lambda_{v\theta}}{m^2} + \frac{\lambda_{vr} T \sin\beta}{m^2}$$

Rewriting the co-state equations,

$$\dot{\lambda_r}^T = \frac{\lambda_\theta v_\theta}{r^2} + \frac{\lambda_{vr} {v_\theta}^2}{r^2} - \frac{2\lambda_{vr}\mu}{r^3} - \frac{\lambda_{v\theta} v_r v_\theta}{r^2} \quad (6)$$

$$\dot{\lambda_\theta}^T = 0 \quad (7)$$

$$\dot{\lambda_{vr}}^T = \frac{\lambda_{v\theta} v_\theta}{r} - \lambda_r \quad (8)$$

$$\dot{\lambda_{v\theta}}^T = -\frac{\lambda_\theta}{r} - \frac{2\lambda_\theta \lambda_{vr}}{r} + \frac{\lambda_{v\theta} v_r}{r} \quad (9)$$

$$\dot{\lambda_m}^T = \frac{T\cos\beta\,\lambda_{v\theta}}{m^2} + \frac{\lambda_{vr} T \sin\beta}{m^2} \quad (10)$$

1.2 Formulation of Optimal Control Problem

v

To solve a system of 10 differential equations 10 boundary conditions are needed, but only 5 initial boundary conditions, $X(t_0)$, are known:

$$r_0 = 1; \quad (11)$$

$$\theta_0 = 0; \quad (12)$$

$$v_r(0) = 0; \quad (13)$$

$$v_\theta(0) = \sqrt{\frac{\mu}{r(0)}}; \quad (14)$$

$$m_0 = 1; \quad (15)$$

The other 3 boundary condition are known at the final time according to the desired final characteristics for the spacecraft. In order to achieve the desired energy orbit, the radial

velocity of the vehicle needs to be zero, and the circumferential velocity needs to be that defined by $\sqrt{\frac{\mu}{r_f}}$ where r_f denotes the desired final radial position. This provides 3 of the necessary 5 boundary conditions, which are:

$$r_f = 1.5; \tag{16}$$

$$\theta_f = free; \tag{17}$$

$$v_r(f) = 0; \tag{18}$$

$$v_\theta(f) = \sqrt{\frac{\mu}{r(f)}}; \tag{19}$$

$$m_f = free; \tag{20}$$

The other boundary conditions are obtained from the final transversality conditions.
Since $\theta(t_f) = free$, the transversality condition on the final state applies, which implies

$$\frac{dM}{d\theta(t_f)} - v^T \frac{db}{d\theta(t_f)} - \lambda_\theta^T(t_f) = 0$$

Since M is not a function of $\theta(t_f)$ and the boundary condition on θ does not exist at the final time,

$$\lambda_\theta^T(t_f) = 0 \tag{21}$$

And since $m_f = free$, the transversality condition on the final state applies, which implies

$$\frac{dM}{dm(t_f)} - v^T \frac{db}{dm(t_f)} - \lambda_m^T(t_f) = 0$$

Since boundary condition on m does not apply as it is free,

$$\lambda_m^T(t_f) - 1 = 0 \tag{22}$$

Since we are trying to minimize the final time for the orbit transfer, time is a free variable. Therefore, an extra equation is necessary in order to solve for the 11 unknowns.

Since $t_f = free$, the transversality conditions on the final time applies, which implies

$$H_{t_f} + \frac{dM}{dt_f} - v^T \frac{db}{dt_f} = 0$$

Since the mayor cost M of the objective function does not dependent on t_f

$$H_{t_f} = 0 \tag{23}$$

The 12 equations form equation (11) to equation (23) (except equation (17) and equation (20)) are the necessary boundary conditions.

Form the optimal control, we can obtain the equation to calculate the throttle angle (control input) β.

$$\frac{dH}{d\beta} = \frac{\lambda_{vr} T \cos\beta}{m} - \frac{\lambda_{v\theta} T \sin\beta}{m} = 0$$

$$\frac{\lambda_{vr}\cos\beta}{1} = \frac{\lambda_{v\theta}\sin\beta}{1}$$

$$\frac{\sin\beta}{\cos\beta} = \tan\beta = \frac{\lambda_{vr}}{\lambda_{v\theta}}$$

(2) Numerical Solution of Optimal Control Problem

In this section various methods will be used to solve the optimal control problem formulated in Section 1.

(a) Indirect Shooting

Solution and Plots –

```
clear all; close all; clc;
%OTMain
%Known parameters
t_0     = 0;
t_f_guess=3.5;
betaj   =[];
T       = 0.1405;
Ve      = 1.8758344;
mu      = 1;

%Know Boundary conditions
comp_t0 = tic;
r_0     = 1;
r_f     = 1.5;
theta_0 = 0;
vr_0    = 0;
vr_f    = 0;
vt_0    = sqrt(mu/r_0);
vt_f    = sqrt(mu/r_f);
m_0     = 1;



%Initial Lambda guesses
lambda_r_0     = -2;
lambda_theta_0 = 0;
lambda_vr_0    = 2;
lambda_vt_0    = 2;
lambda_m_0     = 2;
%Initial guesses for fsolve
lambda_0_guess = [lambda_r_0; lambda_theta_0; lambda_vr_0; lambda_vt_0; lambda_m_0; t_f_guess];

options = optimset('Display','Iter','TolX',1e-8,'TolFun',1e-8);
z = fsolve(@OTError, lambda_0_guess, options, r_0, theta_0, vr_0, vt_0, m_0, r_f, vr_f, vt_f, T,
mu, Ve);
```

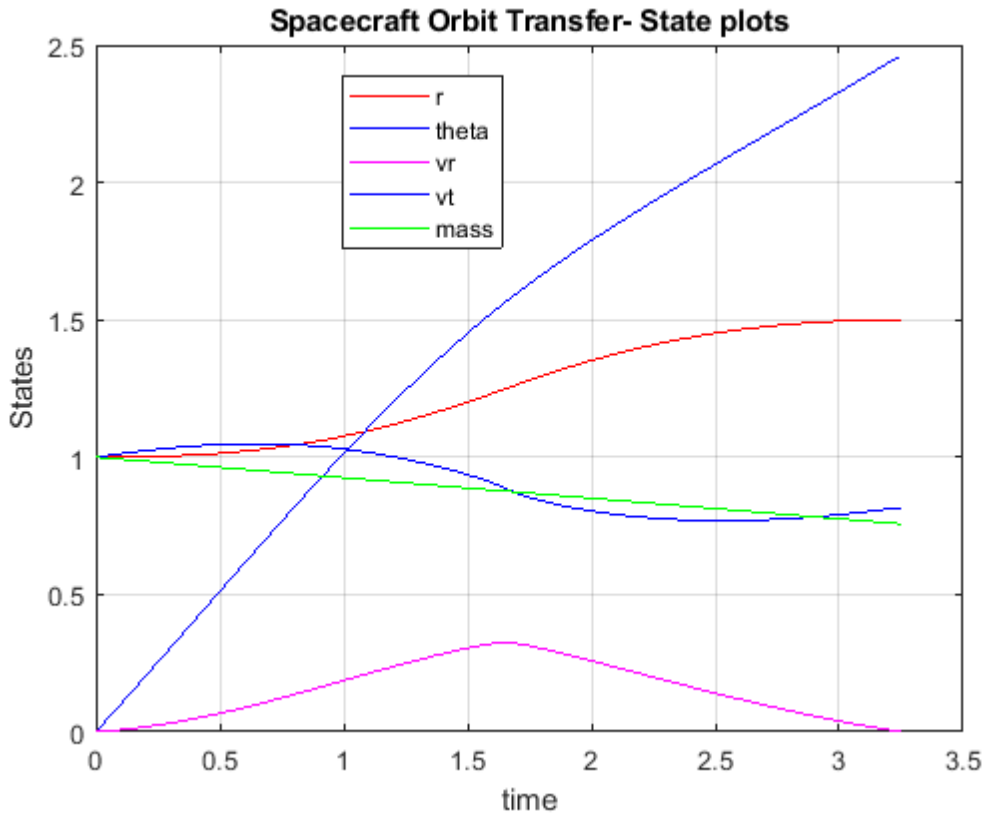
```
[E,t,p] = OTError(z, r_0, theta_0, vr_0, vt_0, m_0, r_f, vr_f, vt_f, T, mu, Ve);

%Plots
%State Plots
figure (1)
plot(t,p(:,1),'r',t,p(:,2),'b',t,p(:,3),'m',t,p(:,4),'bl', t, p(:,5),'g');
title('Spacecraft Orbit Transfer- State plots')
xlabel('time')
ylabel('States')
legend('r','theta','vr','vt','mass','Location','best');
grid on
hold on

%Calculating input for plot
for i=1:length(t)
    betaj(i) = atan2(p(i,8),p(i,9));
    betaj = unwrap(betaj);

end
%Input plot
figure (2)
plot(t,betaj);
title('Spacecraft Orbit Transfer- Input plot')
xlabel('time')
ylabel('Input')
legend('beta','Location','best');
grid on
hold on

%Polar plot of the position of the spacecraft
figure (3)
title('Spacecraft Orbit Transfer- Polar plot')
polarplot(p(:,2),p(:,1));
hold on;
comp_tf = toc(comp_t0);

%Orbit_transfer_myODE
function [P_dot]=OTODE(~,P,T,mu,Ve)
r       = P(1);
theta   = P(2);
vr      = P(3);
vt      = P(4);
m       = P(5);
lambda_r=P(6);
lambda_theta = P(7);
lambda_vr  = P(8);
lambda_vt = P(9);
lambda_m = P(10);

%solve for beta
beta    = atan2(lambda_vr,lambda_vt);
%first-order differential equations
r_dot   = vr;
```

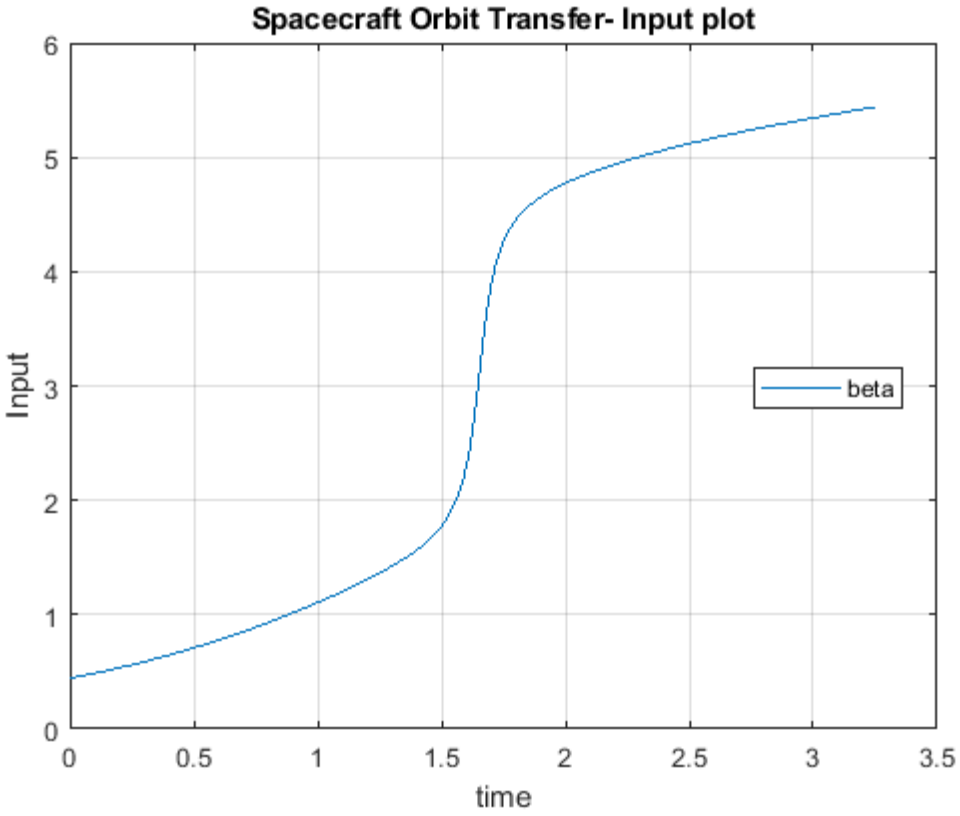
```
theta_dot = vt/r;
vr_dot  = (T/m)*sin(beta) + ((vt^2)/r) - mu/r^2;
vt_dot  = (T/m)*cos(beta) - vr*vt/r;
m_dot   = -T/Ve;
lambda_r_dot =lambda_theta*vt/(r^2) + lambda_vr*(vt^2)/(r^2) - 2*lambda_vr*mu/(r^3) -
lambda_vt*vr*vt/(r^2);
lambda_theta_dot = 0;
lambda_vr_dot = -lambda_r + lambda_vt*vt/r;
lambda_vt_dot = -lambda_theta/r - (2*vt*lambda_vr/r) + lambda_vt*vr/r;
lambda_m_dot = lambda_vt*T*cos(beta)/(m^2) + lambda_vr*T*sin(beta)/m^2;

P_dot   = [r_dot; theta_dot; vr_dot; vt_dot; m_dot; lambda_r_dot; lambda_theta_dot;
lambda_vr_dot; lambda_vt_dot; lambda_m_dot];

end

%OTError
function [E,t,P] = OTError(z_0, r_0, theta_0, vr_0, vt_0, m_0, r_f, vr_f, vt_f, T, mu, Ve)

P_0     = [r_0; theta_0; vr_0; vt_0; m_0; z_0(1:end-1)]; %constructing the initial guesses for
ode113
options = odeset('RelTol', 1e-8);
t_f_guess = z_0(end);
tspan   = [0, t_f_guess];
[t,P]   = ode113(@OTODE, tspan, P_0, options, T, mu, Ve);

Pth     = P(end,:).';
r_tf    = Pth(1);
theta_tf = Pth(2);
vr_tf   = Pth(3);
vt_tf   = Pth(4);
m_tf    = Pth(5);
lambda_r_tf = Pth(6);
lambda_theta_tf = Pth(7);
lambda_vr_tf = Pth(8);
lambda_vt_tf = Pth(9);
lambda_m_tf = Pth(10);
%input at final time to calucalate H_tf
beta_tf = atan2(lambda_vr_tf,lambda_vt_tf);
%Hamiltonian at final time
H_tf    = -1 + lambda_r_tf*vr_tf +lambda_theta_tf*vt_tf/r_tf
+lambda_vr_tf*(sin(beta_tf)*(T/m_tf)+(vt_tf^2)/r_tf-
mu/r_tf^2)+lambda_vt_tf*(cos(beta_tf)*(T/m_tf)-vr_tf*vt_tf/r_tf)-lambda_m_tf*(T/Ve); %beta
uncertainty
%Errors
E(1,1)  = r_tf-r_f;
E(2,1)  = vr_tf-vr_f;
E(3,1)  = vt_tf-vt_f;
E(4,1)  = H_tf;
E(5,1)  = lambda_theta_tf;
E(6,1)  = lambda_m_tf-1;

end
```

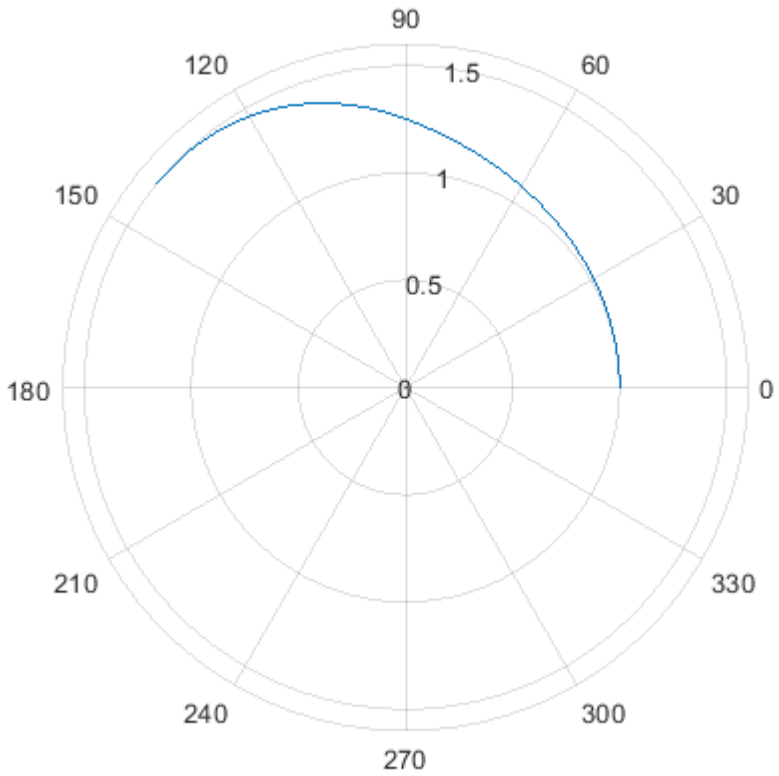
Iteration	Func-count	f(x)	Norm of step	First-order optimality	Trust-region radius
0	7	97.8204		65.4	1
1	14	22.3539	1	14.5	1
2	21	4.52851	1	3.23	1
3	28	0.780767	1	0.503	1
4	35	0.536553	2.5	0.136	2.5
5	36	0.536553	6.25	0.136	6.25
6	43	0.50316	1.5625	0.0924	1.56
7	44	0.50316	3.90625	0.0924	3.91
8	51	0.485985	0.976563	0.121	0.977
9	58	0.362908	0.976562	0.136	0.977
10	65	0.228069	0.976563	0.0727	0.977
11	72	0.0626226	2.44141	0.13	2.44
12	79	0.0344605	1.70839	0.377	2.44
13	86	0.00049303	0.221202	0.029	2.44
14	93	1.16193e-06	0.0174211	0.00124	2.44
15	100	1.54958e-12	0.00162618	1.5e-06	2.44
16	107	1.46556e-23	1.43948e-06	4.26e-12	2.44

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.







Published with MATLAB® R2021a

Tabular summary of Performance

Computation Time (in sec)	Number of Iterations	Objective function J (mass at t_f)	F(x) at final iteration
1.6233	16	0.7567	1.46556e-23

ANALYSIS –

Fsolve was able to solve the OTError function, implying that the equations define in the error were met, resulting in the optimal solution. The final mass of the spacecraft was calculated to be 0.7567 units. The polar plot indicates that the final orbit was reached. The mass curve in the state plot indicates a linear decay of mass. The input plot is almost linear for $\approx 1.6\ seconds$ after which there is a switch in the input.

(b) Indirect Multiple shooting

Solutions and Plots
(b.1) For Number of Intervals, K = 2

```
clear all; close all; clc;
%OTMain
%Know conditions
comp_t  =tic;
T       = 0.1405;
Ve      = 1.8758344;
mu      = 1;
nx      = 5; %No of States
K       = 2; %No of intervals
%boundary conditions
r_0     = 1;
r_f     = 1.5;
theta_0 = 0;
vr_0    = 0;
vr_f    = 0;
```

```
vt_0    = sqrt(mu/r_0);
vt_f    = sqrt(mu/r_f);
m_0     = 1;
t_0     = 0;
t_f_guess = 3.5;

betaj=[];
%Initial Lambda guesses
lambda_r_0      = -2;
lambda_theta_0  = 0;
lambda_vr_0     = 2;
lambda_vt_0     = 2;
lambda_m_0      = 2;

P_0_guess = ones(nx*2, K-1);
lambda_0_guess = [lambda_r_0; lambda_theta_0; lambda_vr_0; lambda_vt_0; lambda_m_0; t_f_guess;
P_0_guess(:)];
tau = linspace(-1,1,K+1);
options = optimset('Display','Iter','TolX',1e-8,'TolFun',1e-8);
z = fsolve(@OTError, lambda_0_guess, options, r_0, theta_0, vr_0, vt_0, m_0, r_f, vr_f, vt_f, T,
mu, Ve, nx, K, tau, t_0);
%plot using integrated dynamics
[E,t,p] = OTError(z, r_0, theta_0, vr_0, vt_0, m_0, r_f, vr_f, vt_f, T, mu, Ve, nx, K, tau, t_0);

%Plots
figure (1)
plot(t,p(:,1),'r',t,p(:,2),'b',t,p(:,3),'m',t,p(:,4),'k', t, p(:,5),'g');
title('Spacecraft Orbit Transfer- State plots')
xlabel('time')
ylabel('States')
legend('r','theta','vr','vt','mass','Location','best');
grid on
hold on

%Recalculating beta for plot
for i=1:length(t)
    betaj(i)= atan2(p(i,8),p(i,9));
    betaj = unwrap(betaj);

end
figure (2)
plot(t,betaj);
title('Spacecraft Orbit Transfer- Input plot')
xlabel('time')
ylabel('Input')
legend('beta','Location','best');
grid on
hold on

figure (3)
polarplot(p(:,2),p(:,1));
hold on;
```

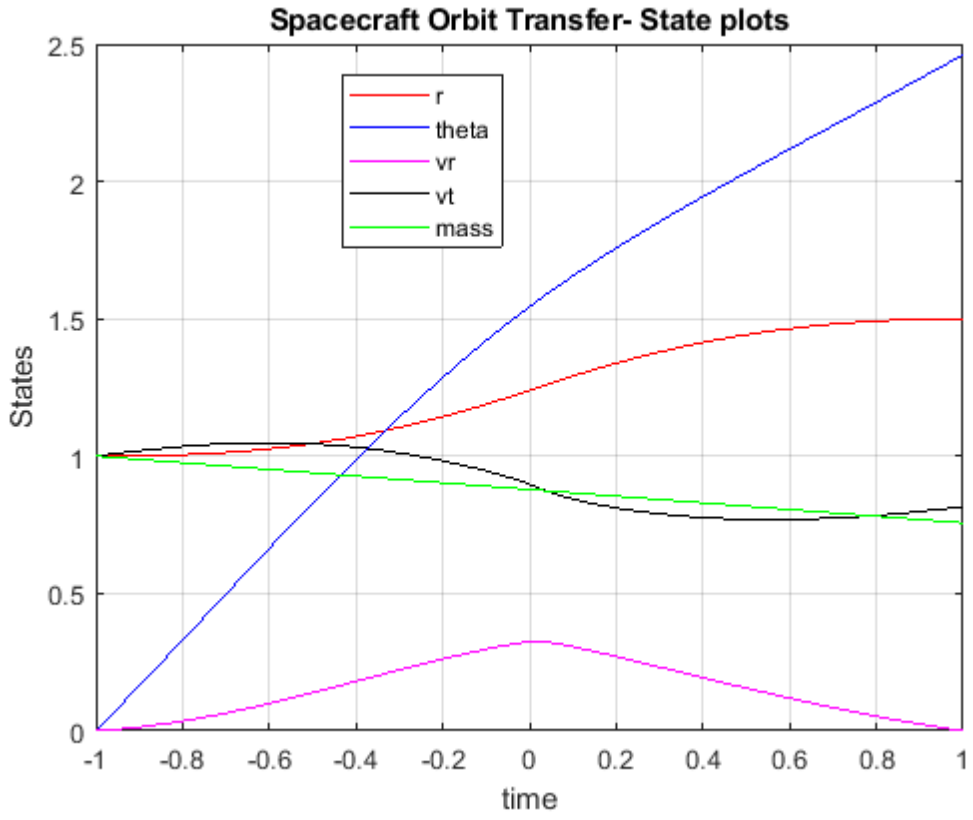
```
comp_tf=toc(comp_t);

%Orbit_transfer_myODE
function [P_dot]=OTODE(~,P,T,mu,Ve,t_0, t_f)
r       = P(1);
theta   = P(2);
vr      = P(3);
vt      = P(4);
m       = P(5);
lambda_r =P(6);
lambda_theta = P(7);
lambda_vr   = P(8);
lambda_vt   = P(9);
lambda_m    = P(10);


%solve for beta
beta    = atan2(lambda_vr,lambda_vt);

%first-order differential equations
r_dot   = vr;
theta_dot = vt/r;
vr_dot  = (T/m)*sin(beta) + ((vt^2)/r) - mu/r^2;
vt_dot  = (T/m)*cos(beta) - vr*vt/r;
m_dot   = -T/Ve;
lambda_r_dot =lambda_theta*vt/(r^2) + lambda_vr*(vt^2)/(r^2) - 2*lambda_vr*mu/(r^3) -
lambda_vt*vr*vt/(r^2);
lambda_theta_dot = 0;
lambda_vr_dot = -lambda_r + lambda_vt*vt/r;
lambda_vt_dot = -lambda_theta/r - (2*vt*lambda_vr/r) + lambda_vt*vr/r;
lambda_m_dot = lambda_vt*T*cos(beta)/(m^2) + lambda_vr*T*sin(beta)/m^2;

P_dot   = [r_dot; theta_dot; vr_dot; vt_dot; m_dot; lambda_r_dot; lambda_theta_dot;
lambda_vr_dot; lambda_vt_dot; lambda_m_dot];
P_dot   = (t_f-t_0)/2 * P_dot; %Scaling from tau to t

end

%OTError
function [E,t,P] = OTError(z_0, r_0, theta_0, vr_0, vt_0, m_0, r_f, vr_f, vt_f, T, mu, Ve, nx, K,
tau, t_0)
t_f         = z_0(6);
P_0_guess   = z_0(7:end);
P_0_guess   = reshape(P_0_guess, 2*nx, K-1);
options     = odeset('RelTol', 1e-8);
E=[];
t=[];
P=[];

for k=1:K
    if k==1
        %for the first interval since the inital boundary conditions are
        %known
```

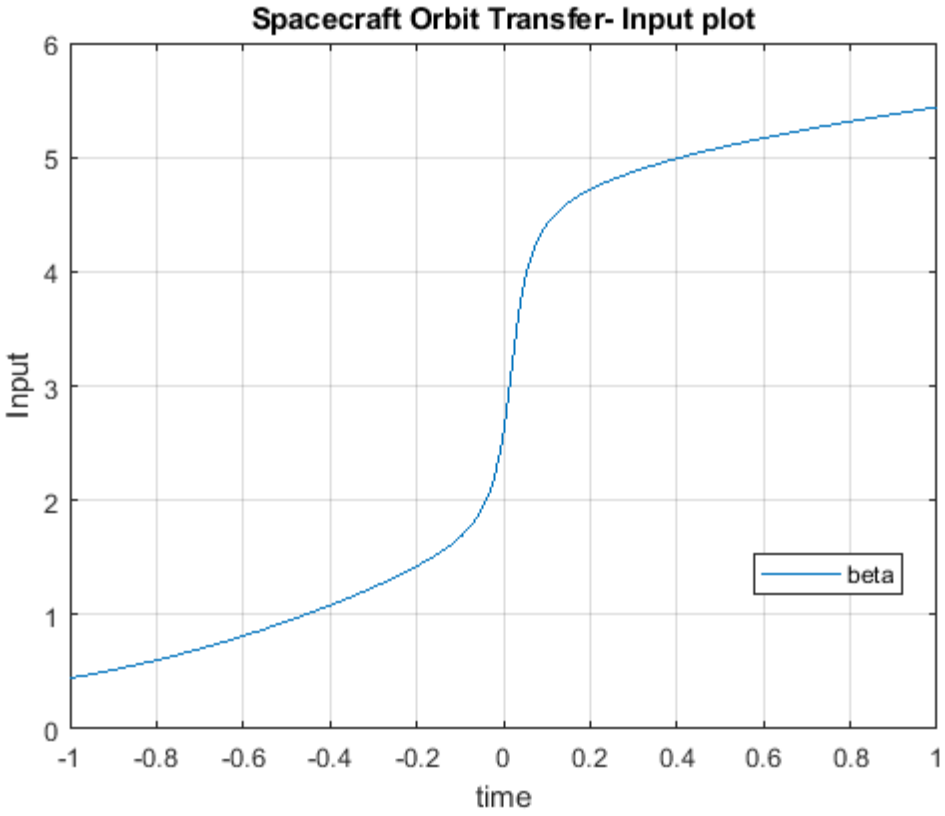
```
        P_0 = [r_0; theta_0; vr_0; vt_0; m_0; z_0(1:5)];
    else
        P_0 = P_0_guess(:,k-1);
    end

    tspan =[tau(k), tau(k+1)];
    [tout,Pout] = ode113(@OTODE, tspan, P_0, options, T, mu, Ve, t_0, t_f);
    Pth = Pout(end,:).';

    if k<K
        E = [E;Pth-P_0_guess(:,k)];
    end
    t=[t;tout];
    P=[P;Pout];
end

r_tf        =Pth(1);
theta_tf    = Pth(2);
vr_tf       = Pth(3);
vt_tf       = Pth(4);
m_tf        = Pth(5);
lambda_r_tf = Pth(6);
lambda_theta_tf = Pth(7);
lambda_vr_tf = Pth(8);
lambda_vt_tf = Pth(9);
lambda_m_tf = Pth(10);

beta_tf     = atan2(lambda_vr_tf,lambda_vt_tf);
H_tf        = -1 + lambda_r_tf*vr_tf +lambda_theta_tf*vt_tf/r_tf
+lambda_vr_tf*(sin(beta_tf)*(T/m_tf)+(vt_tf^2)/r_tf-
mu/r_tf^2)+lambda_vt_tf*(cos(beta_tf)*(T/m_tf)-vr_tf*vt_tf/r_tf)-lambda_m_tf*(T/Ve); %beta
uncertainty

E = [ E; r_tf-r_f; vr_tf-vr_f; vt_tf-vt_f; H_tf; lambda_theta_tf; lambda_m_tf-1];
end
```

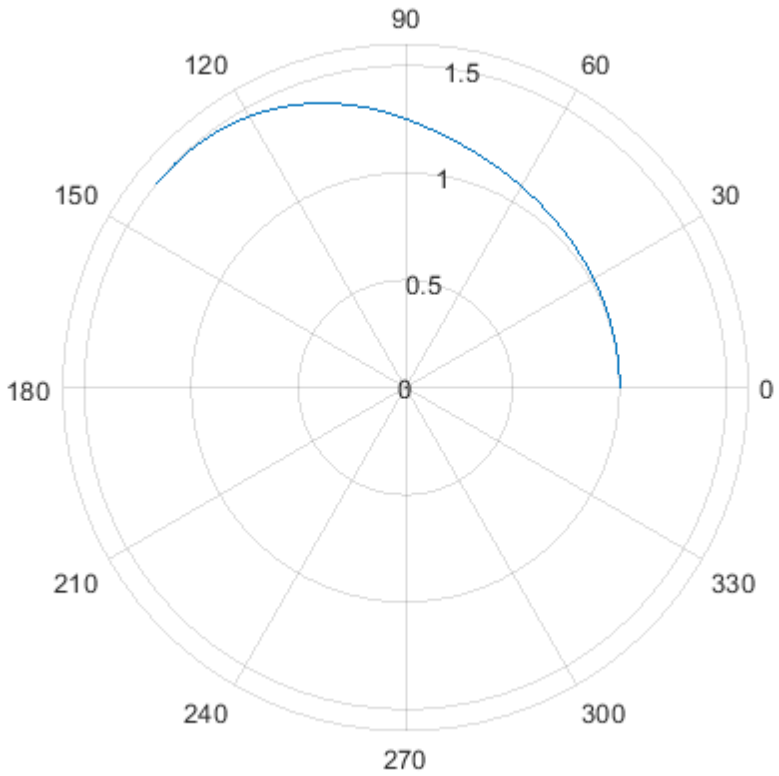
Iteration	Func-count	f(x)	Norm of step	First-order optimality	Trust-region radius
0	17	249.599		67.2	1
1	34	100.406	1	29.8	1
2	51	32.9232	1	11.7	1
3	68	3.38718	2.5	1.91	2.5
4	85	0.630491	3.27664	0.762	6.25
5	102	0.288667	4.093	1.09	6.25
6	119	0.0282123	2.76868	0.126	6.25
7	136	0.00533832	1.03747	0.149	6.92
8	153	6.75639e-07	0.0905774	0.000896	6.92
9	170	3.04423e-11	0.00628885	1.77e-05	6.92
10	187	2.11857e-22	1.14518e-05	2.85e-11	6.92

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.
```





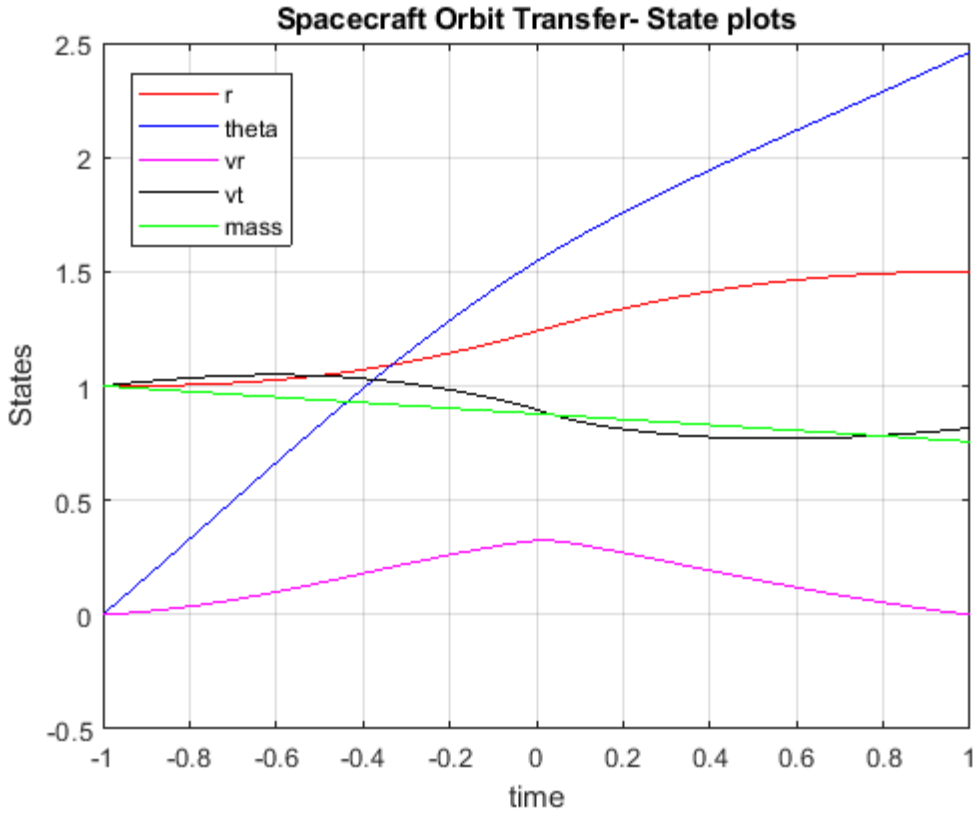


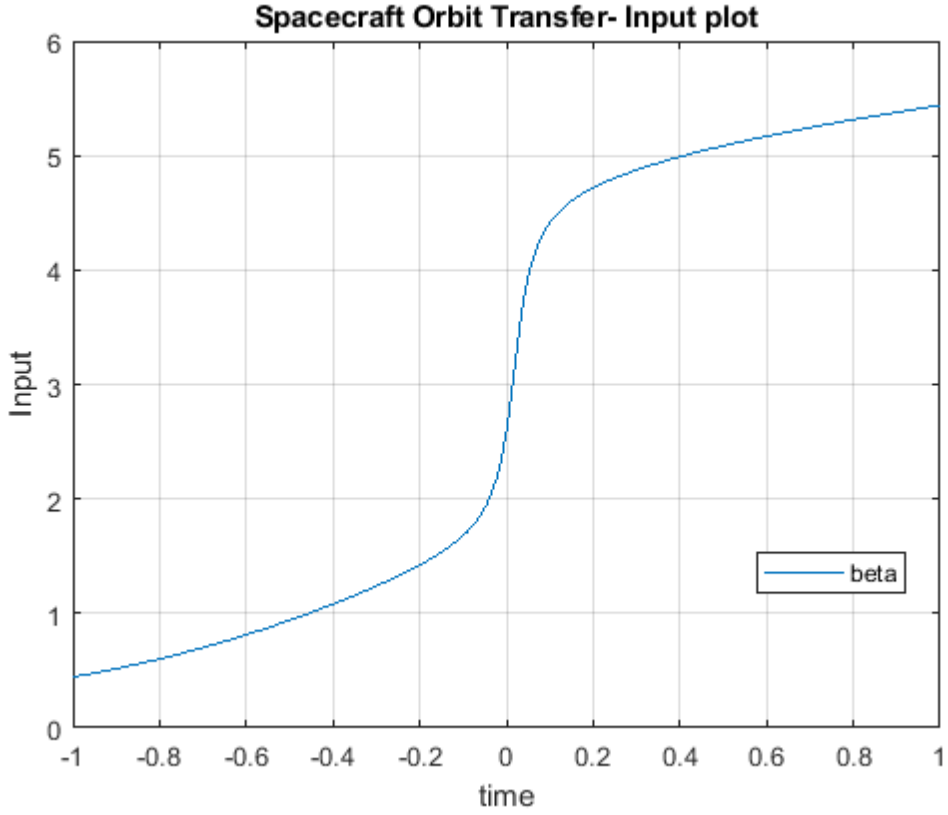
Published with MATLAB® R2021a

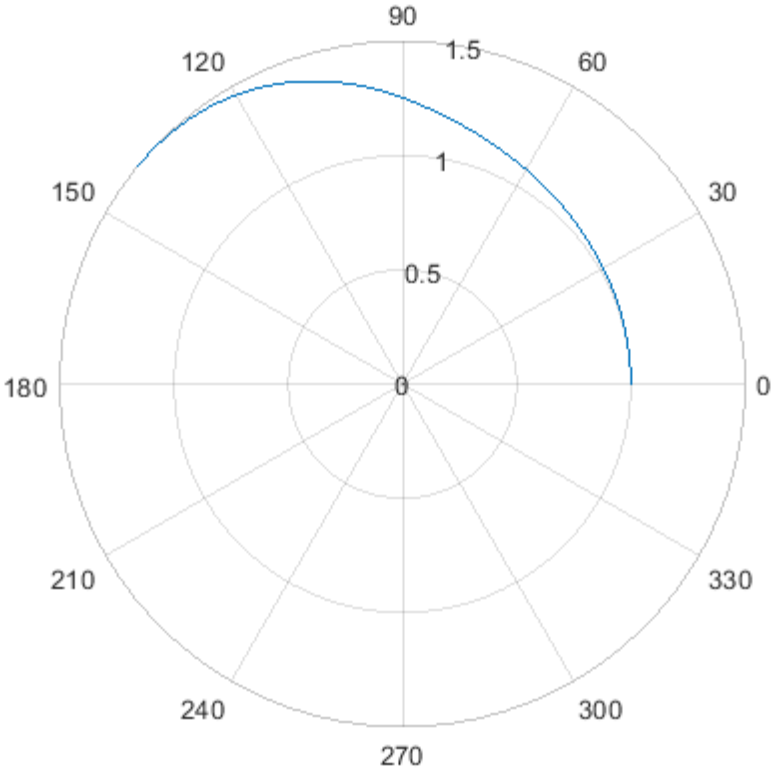
(b.2) For Number of Intervals, K = 4

Iteration	Func-count	f(x)	Norm of step	First-order optimality	Trust-region radius
0	37	79.581		17.9	1
1	74	36.193	1	8.98	1
2	111	5.46439	2.5	1.24	2.5
3	148	3.61206	5.35883	2.25	6.25
4	185	1.51283	6.25	1.71	6.25
5	186	1.51283	15.625	1.71	15.6
6	223	0.874596	3.90625	0.459	3.91
7	224	0.874596	9.76563	0.459	9.77
8	261	0.681969	2.44141	0.253	2.44
9	298	0.577965	6.10352	0.296	6.1
10	299	0.577965	6.10352	0.296	6.1
11	336	0.270795	1.52588	0.236	1.53
12	373	0.121441	1.52588	0.194	1.53
13	374	0.121441	3.8147	0.194	3.81
14	411	0.0687455	0.953674	0.0891	0.954
15	448	0.0322117	2.38419	0.212	2.38
16	485	0.00215257	1.4784	0.0632	2.38
17	522	1.3981e-08	0.101814	0.000185	3.7
18	559	9.10291e-18	0.000419608	4.01e-09	3.7

```
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.
```


Spacecraft Orbit Transfer- State plots
r
theta
vr
vt
mass
States
time
2.5
2
1.5
1
0.5
0
-0.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


Spacecraft Orbit Transfer- Input plot
beta
Input
time
6
5
4
3
2
1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

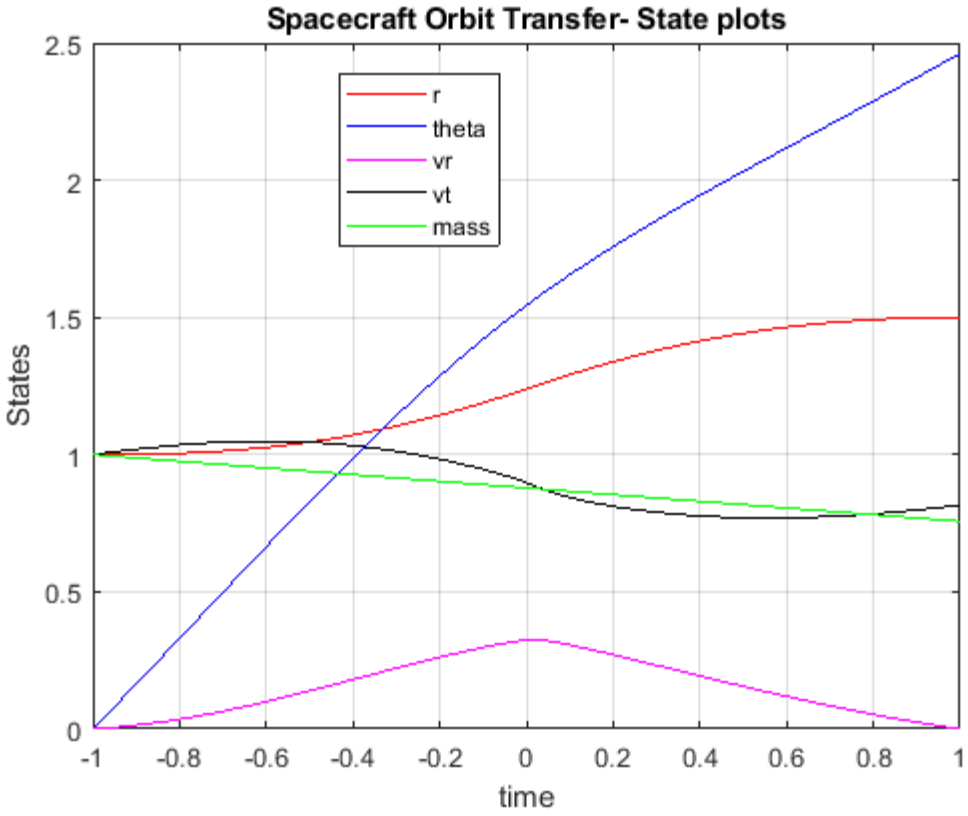


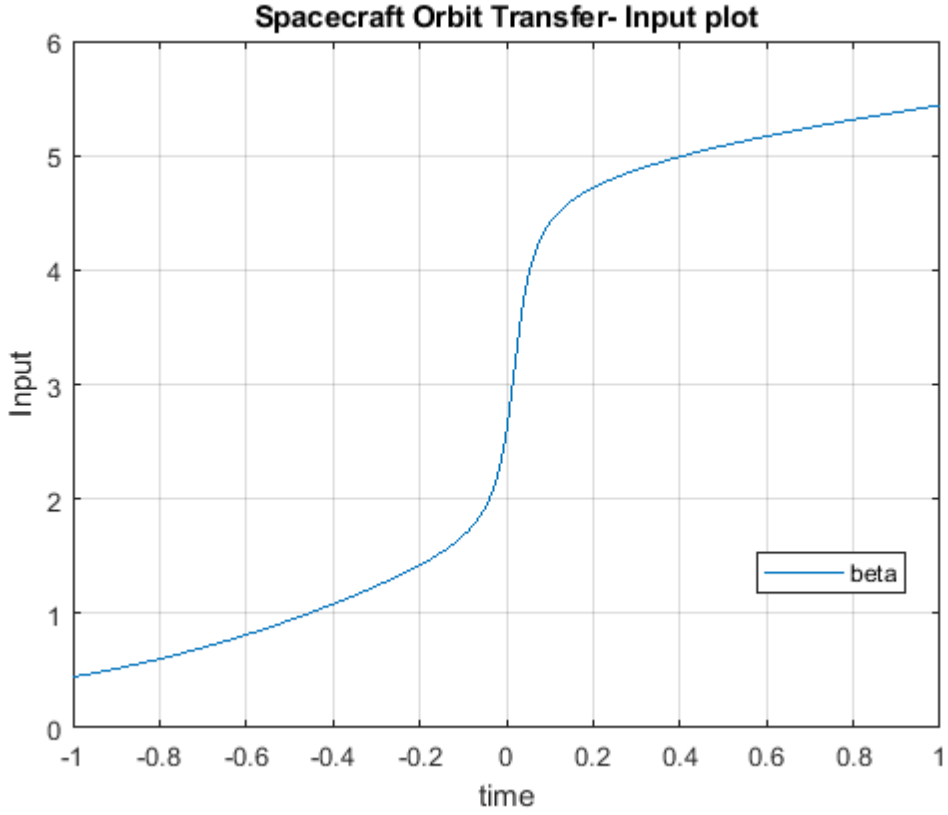
Published with MATLAB® R2021a

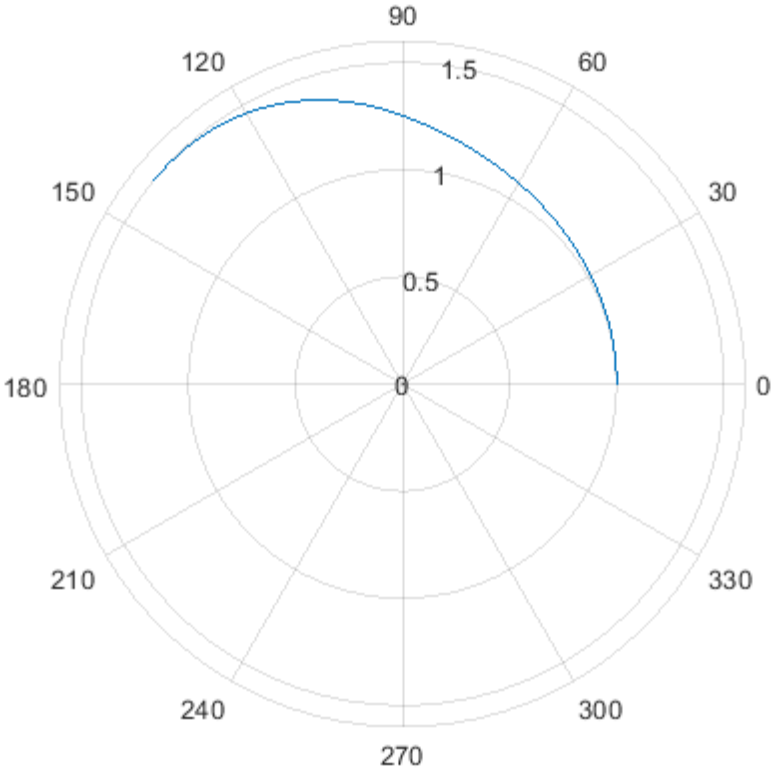
(3) No of intervals, K = 8

Iteration	Func-count	f(x)	Norm of step	First-order optimality	Trust-region radius
0	77	37.2476		5.91	1
1	154	19.1457	1	3.72	1
2	231	5.54091	2.5	1.05	2.5
3	308	3.17446	6.25	0.669	6.25
4	385	1.85442	6.25	0.517	6.25
5	462	0.724318	6.25	0.335	6.25
6	463	0.724318	10.2834	0.335	15.6
7	540	0.336834	2.57085	0.426	2.57
8	617	0.153961	6.42713	0.645	6.43
9	694	0.0301499	3.64378	0.247	6.43
10	771	5.86887e-05	0.79105	0.00842	6.43
11	848	1.24855e-08	0.0533301	6.39e-05	6.43
12	925	1.88539e-15	0.00101665	2.67e-08	6.43
13	1002	4.89442e-26	4.59907e-07	3.08e-13	6.43

```
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.
```


Spacecraft Orbit Transfer- State plots
r
theta
vr
vt
mass
States
time
2.5
2
1.5
1
0.5
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


Spacecraft Orbit Transfer- Input plot
beta
Input
time
6
5
4
3
2
1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

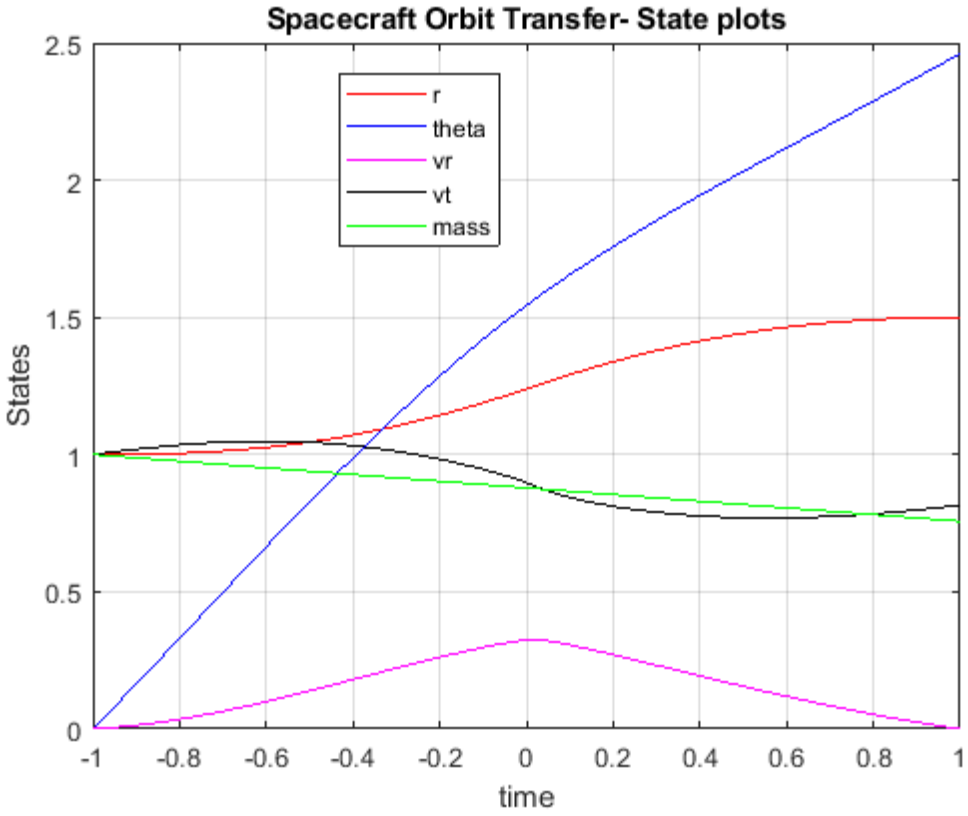


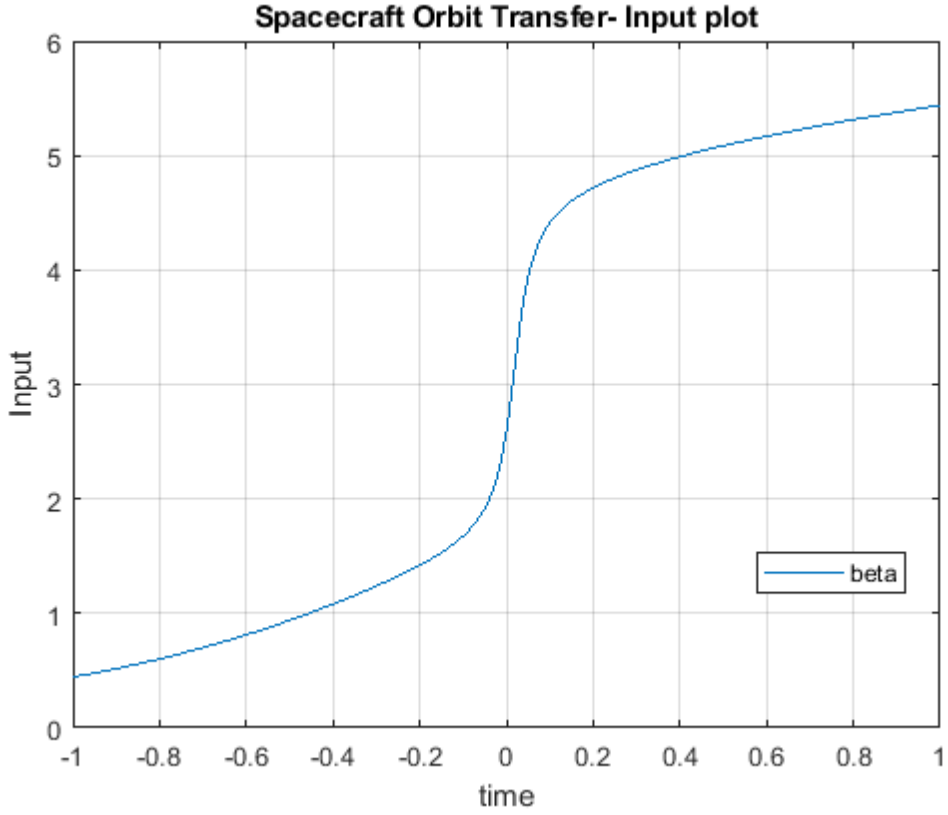
Published with MATLAB® R2021a

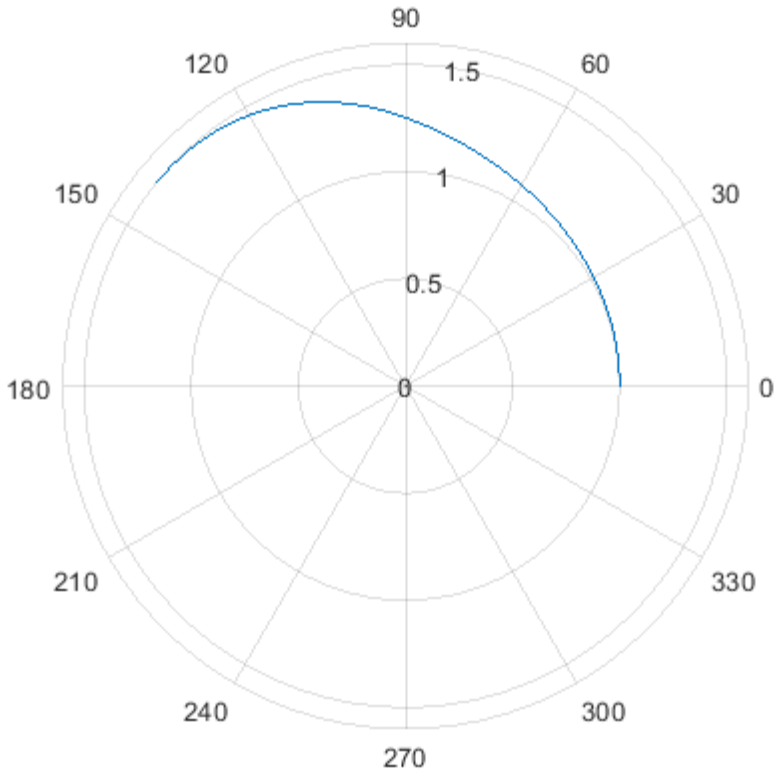
(4) For number of intervals, K =16

Iteration	Func-count	f(x)	Norm of step	First-order optimality	Trust-region radius
0	157	25.6179		4.01	1
1	314	13.9707	1	2.61	1
2	471	5.13981	2.5	0.979	2.5
3	628	3.49743	6.25	0.598	6.25
4	785	1.57958	6.25	0.381	6.25
5	942	0.545178	15.625	0.976	15.6
6	1099	0.0666451	6.91164	0.167	15.6
7	1256	0.000883614	2.39448	0.0268	15.6
8	1413	8.92893e-08	0.169243	0.000208	15.6
9	1570	8.74392e-14	0.00419646	1.48e-07	15.6
10	1727	1.06627e-22	1.1338e-05	8.45e-12	15.6

```
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.
```


Spacecraft Orbit Transfer- State plots
r
theta
vr
vt
mass
States
time
2.5
2
1.5
1
0.5
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


Spacecraft Orbit Transfer- Input plot
beta
Input
time
6
5
4
3
2
1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1



Published with MATLAB® R2021a

Tabular summary of Performance –

Number of intervals (K)	Computation Time (in sec)	Number of Iterations	Objective function J (mass at t_f)	F(x) at final iteration
2	2.5154	10	0.7567	2.11857e-22
4	7.8792	18	0.7567	9.10291e-18
8	15.2118	13	0.7567	4.89442e-26
16	26.1963	10	0.7567	1.06627e-22

Analysis –

The optimal solution was obtained through multiple shooting methods with different intervals $K = \{2, 4, 8, 16\}$. The same final value of the objective was reached by all the iterations with different intervals but however the computational time increased with the increase in the number of intervals as indicated in the table. As the number of intervals increases, the number of functions in the error increases, each of which must be satisfied to find the optimal solution. This increases the complexity of the problem and hence fsolve takes longer computational time as the number of intervals increases. The indirect multiple shooting method and single shooting method reached the same final mass 0.7567 units.

(c) Direct Shooting

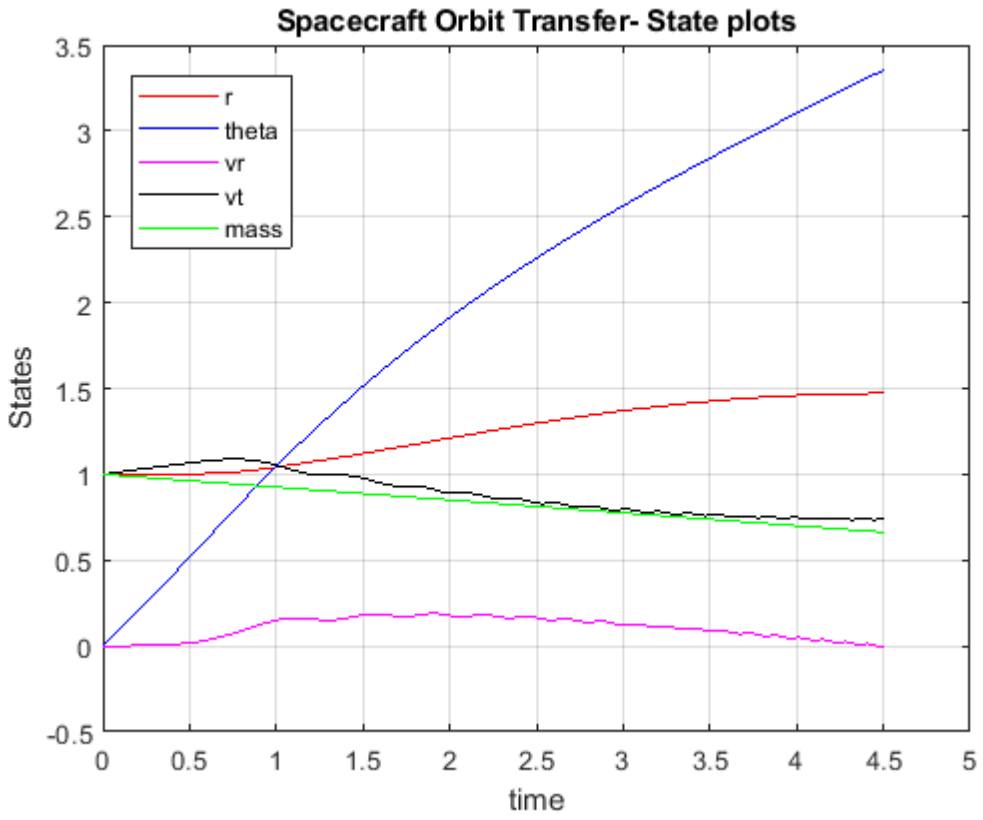
Solution and Plots –

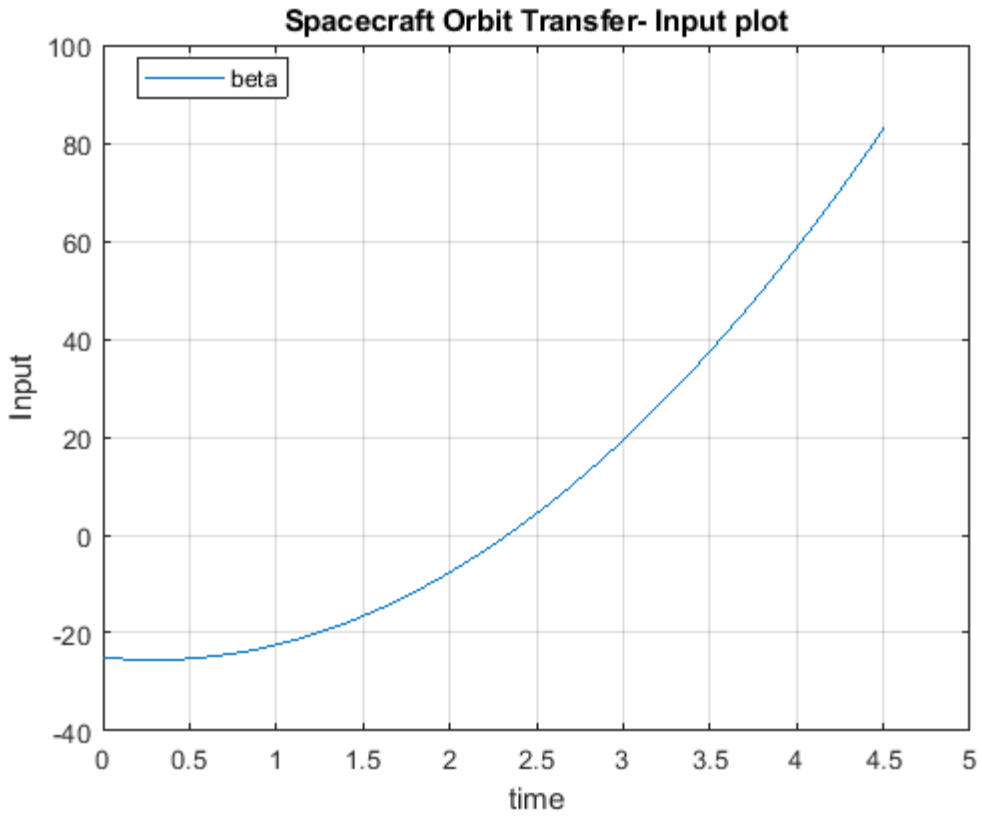
(c.1) For polynomial degree, n=2

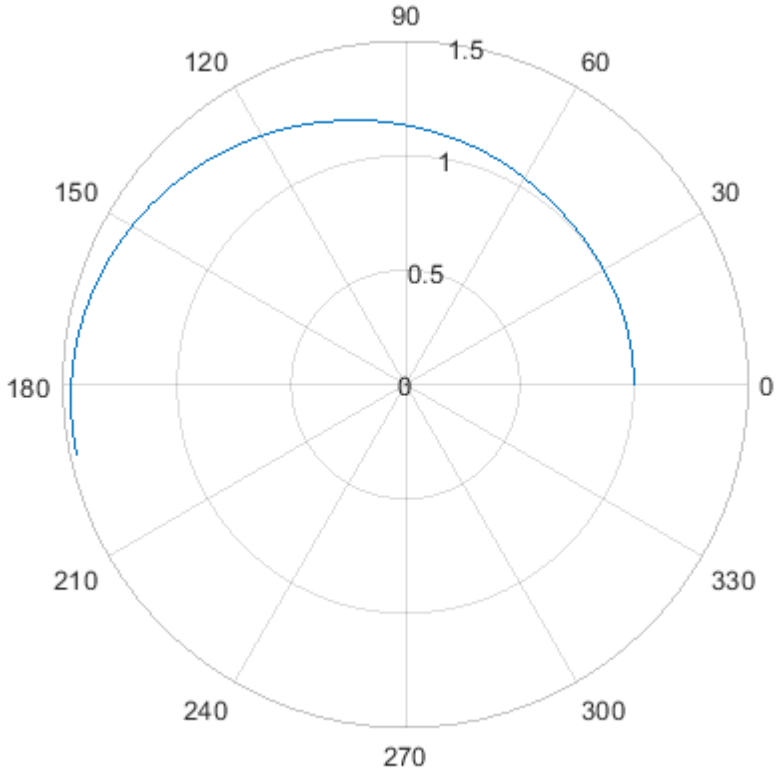
```
clear all; close all; clc;
comp_t0 = tic;
%Spacecraft_Orbit_Transfer_Main
%Know conditions
T   = 0.1405;
Ve  = 1.8758344;
mu  = 1;
n   = 2; %degree of the polynomial used to parametarize the control
tfmin = 0;
tfmax = 100;
%bounds for fmincon
zmin    = [-25*ones(n+1,1);tfmin];
zmax    = [25*ones(n+1,1); tfmax];
cguess  = 0.01*ones(n+1,1);
tfguess = 8;
zguess  = [cguess; tfguess];
A =[];
B =[];
Aeq =[];
Beq =[];

options = optimset('Display','Iter','TolX',1e-8,'TolFun',1e-8, 'MaxfunEvals', 10000);

z = fmincon(@OTObj, zguess, A, B, Aeq, Beq, zmin, zmax,@OTError,options,mu,T,Ve);
c = z(1:end-1);
[Eineq,Eeq,t,p] = OTError(z,mu,T,Ve);
beta = polyval(c,t);

%Plots
figure (1)
plot(t,p(:,1),'r',t,p(:,2),'b',t,p(:,3),'m',t,p(:,4),'k', t, p(:,5),'g');
title('Spacecraft Orbit Transfer- State plots')
xlabel('time')
ylabel('States')
legend('r','theta','vr','vt','mass','Location','best');
grid on
hold on

figure (2)
plot(t,beta);
title('Spacecraft Orbit Transfer- Input plot')
```

```
xlabel('time')
ylabel('Input')
legend('beta','Location','best');
grid on
hold on

figure (3)
polarplot(p(:,2),p(:,1));

comp_tf = toc(comp_t0);

%Spacecraft_Orbit_Transfer_ODE
function [P_dot]=OTODE(t,P,mu,T,Ve,c)
r       = P(1);
theta   = P(2);
vr      = P(3);
vt      = P(4);
m       = P(5);

%solve for beta
beta = polyval(c,t);

%first-order differential equations
r_dot   = vr;
theta_dot = vt/r;
vr_dot  = (T/m)*sin(beta) + ((vt^2)/r) - mu/r^2;
vt_dot  = (T/m)*cos(beta) - vr*vt/r;
m_dot   = -T/Ve;

P_dot = [r_dot; theta_dot; vr_dot; vt_dot; m_dot];

end

%Spacecraft_Orbit_Transfer_Obj
function J = OTObj(z,mu,T,Ve)
m = z(end);
%objective fuction to minmize final time
J = m;
end

%Spacecraft_Orbit_Transfer_Error
function [Eineq,Eeq,t,P] = OTError(z,mu,T,Ve)

t_f     = z(end); %terminal time
c       = z(1:end-1); %coefficients fo the polynomial

%boundary conditions
r_0     = 1;
r_f     = 1.5;
theta_0 = 0;
%theta_f = free;
vr_0    = 0;
vr_f    = 0;
```

```
vt_0    = sqrt(mu/r_0);
vt_f    = sqrt(mu/r_f);
m_0     = 1;
%m_f    = free;

P_0 = [r_0; theta_0; vr_0; vt_0; m_0];

options = odeset('RelTol', 1e-8);
tspan   = [0,t_f];
[t,P]   = ode113(@OTODE, tspan, P_0, options, mu, T, Ve, c);
r_tf    = P(end,1);
theta_tf = P(end,2);
vr_tf   = P(end,3);
vt_tf   = P(end,4);
m_tf    = P(end,5);

Eeq     = [r_tf-r_f; vr_tf-vr_f; vt_tf-vt_f];
Eineq   =[];

end
```



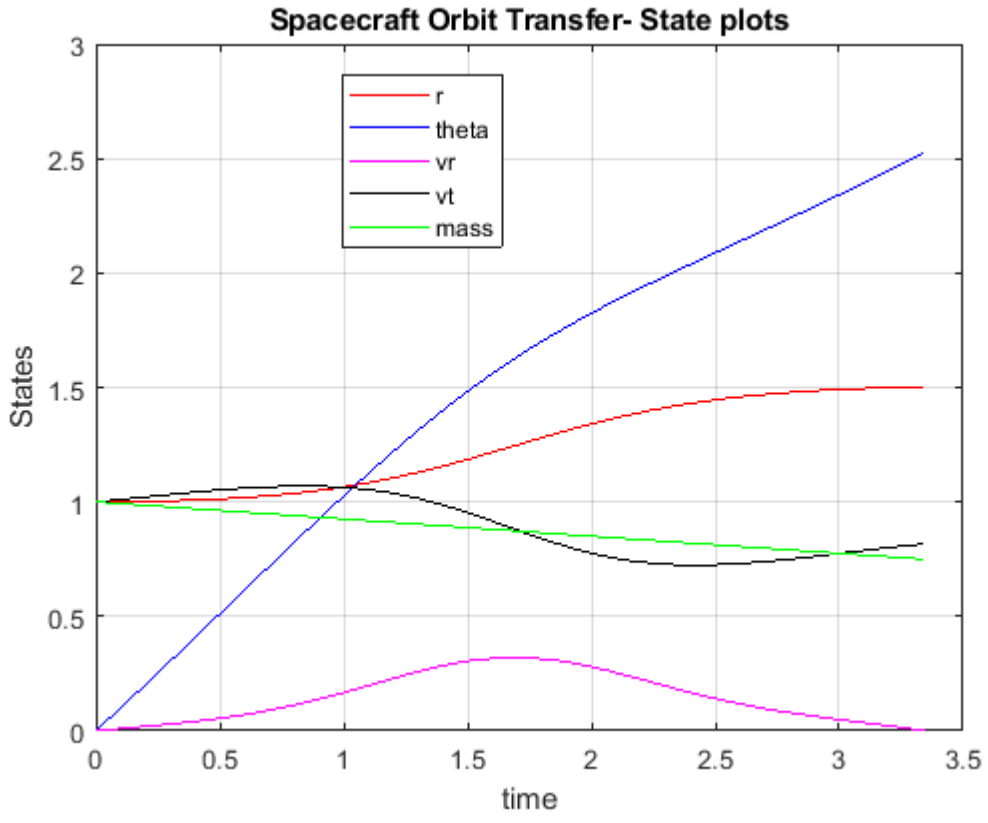


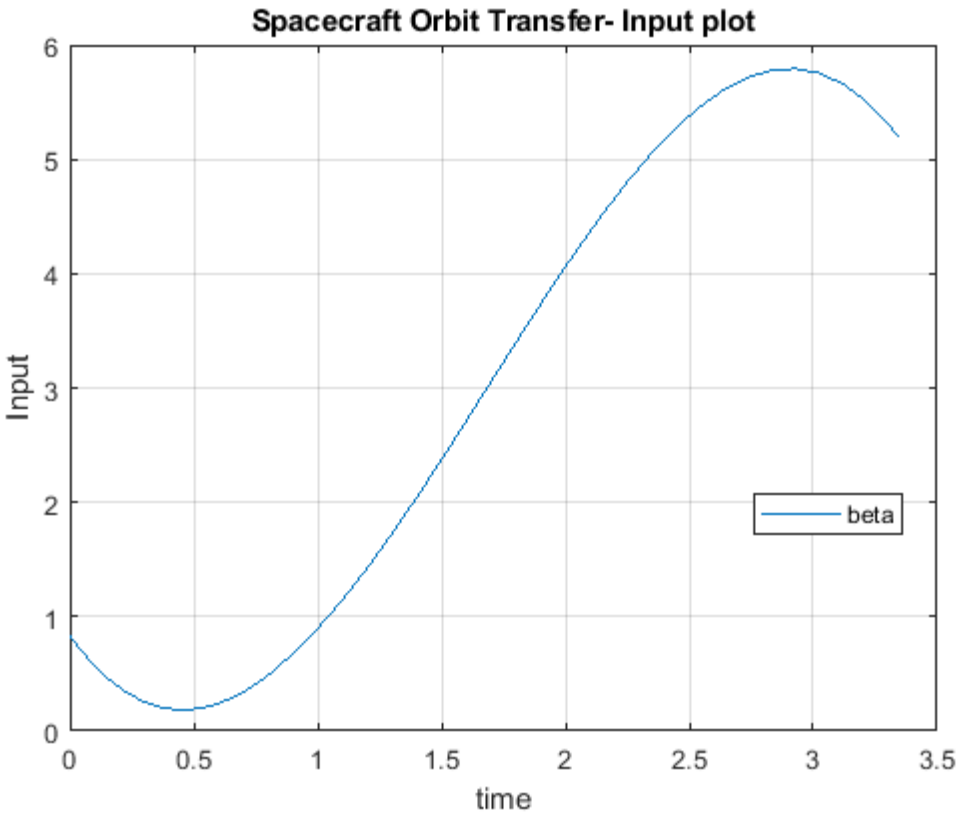


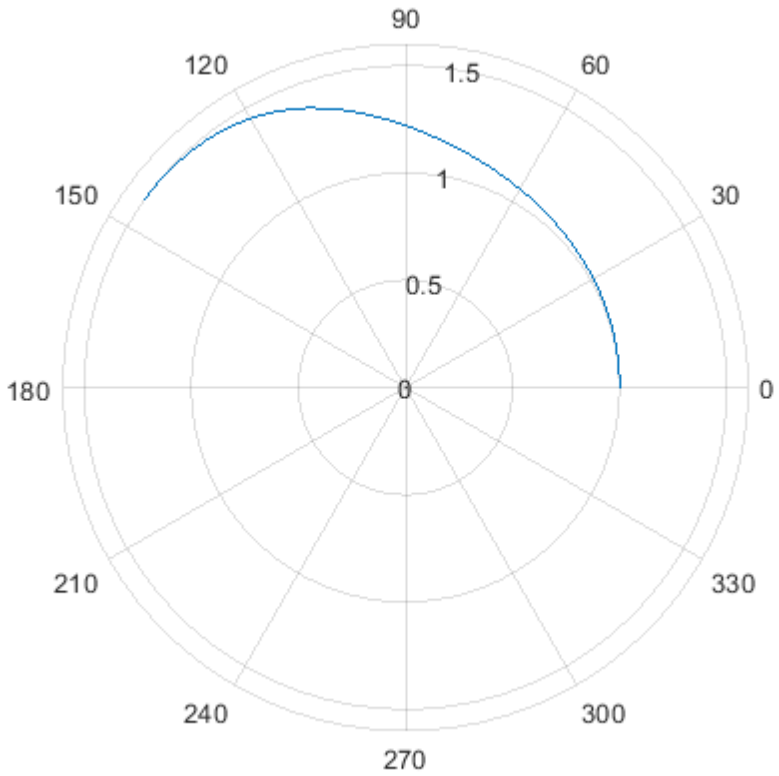
Published with MATLAB® R2021a

(c.2) For Polynomial Degree, n=3

```
Iter F-count            f(x)  Feasibility   optimality         step
   31     259    3.344200e+00    8.230e-09    5.422e-03    9.353e-04
   32     265    3.344200e+00    7.432e-09    5.722e-05    9.220e-04
   33     271    3.344200e+00    5.209e-12    3.475e-05    1.214e-05
   34     285    3.344200e+00    1.849e-12    2.460e-06    6.071e-06
   35     298    3.344200e+00    9.926e-13    6.737e-07    6.513e-07
   36     305    3.344200e+00    4.403e-13    6.999e-06    9.651e-07
   37     311    3.344200e+00    3.733e-13    2.295e-06    8.031e-07
   38     317    3.344200e+00    2.993e-13    3.904e-07    7.753e-07
   39     324    3.344200e+00    6.306e-14    1.572e-06    1.362e-07
   40     330    3.344200e+00    1.112e-14    7.300e-06    2.738e-08
   41     346    3.344200e+00    3.110e-15    6.705e-07    6.845e-09

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.
```

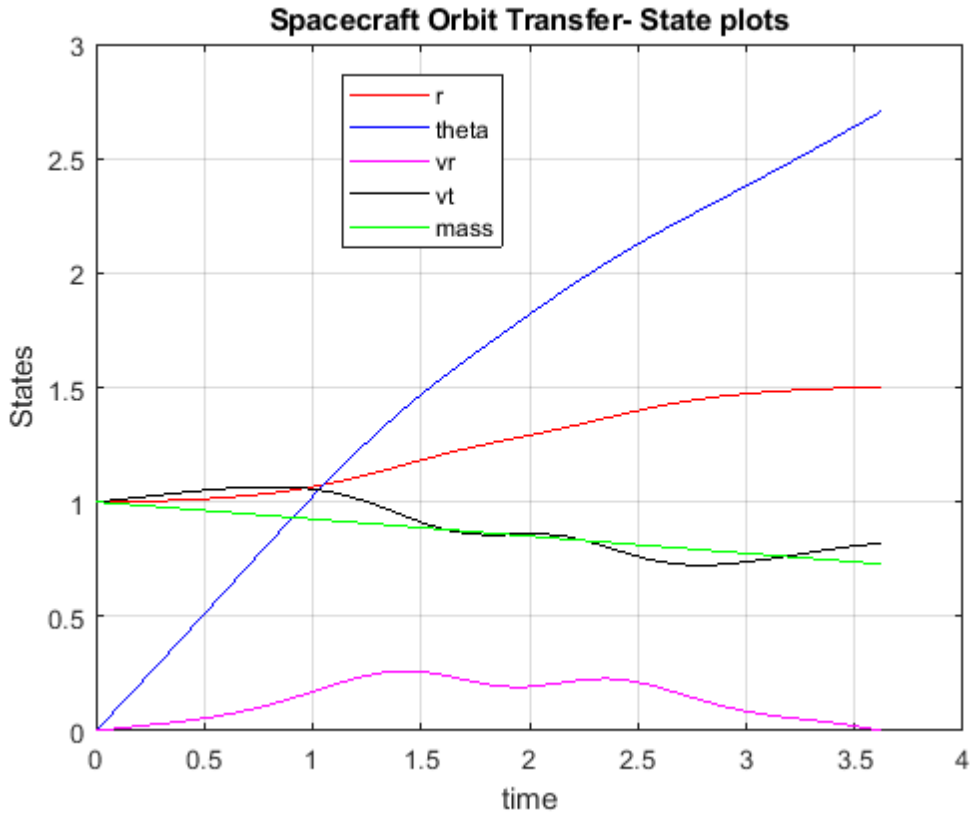


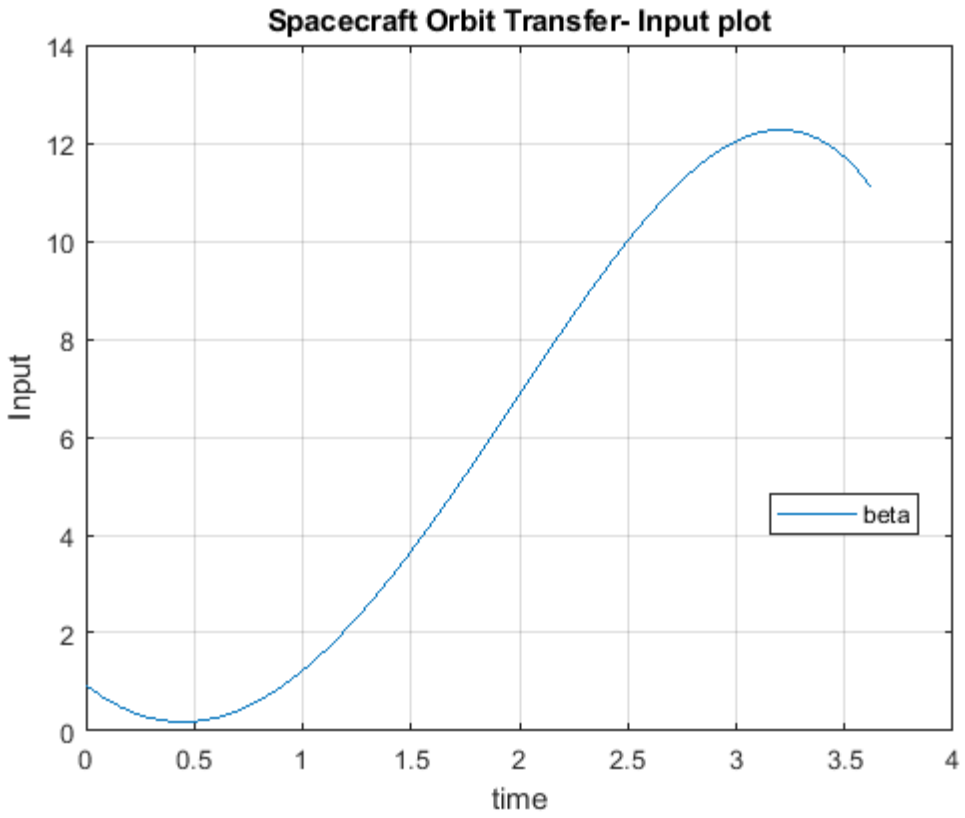


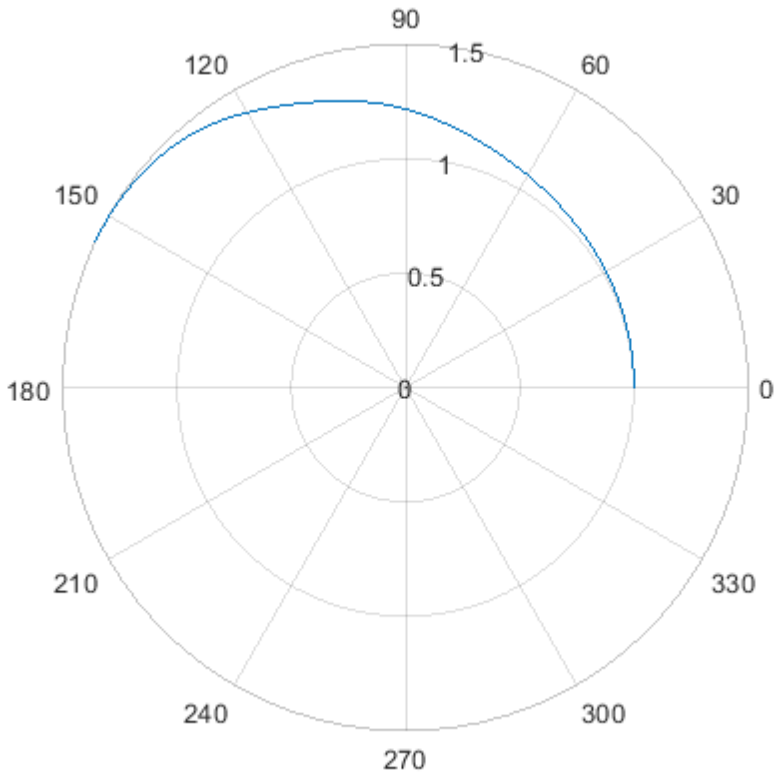


(c.3) For polynomial degree, n=4

```
    73      741     3.622673e+00    1.989e-09    1.293e-03    1.126e-03
    74      748     3.622673e+00    2.409e-10    2.025e-04    6.592e-04
    75      755     3.622673e+00    5.336e-10    1.311e-04    4.063e-04
    76      762     3.622673e+00    7.877e-11    1.458e-05    2.081e-04
    77      769     3.622673e+00    1.007e-10    3.693e-06    1.030e-04
    78      776     3.622673e+00    1.632e-12    2.077e-05    1.084e-05
    79      794     3.622673e+00    3.880e-12    3.651e-06    2.711e-06
    80      803     3.622673e+00    4.247e-12    9.099e-05    2.711e-06
    81      811     3.622673e+00    1.189e-14    2.465e-04    7.278e-06

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.
```

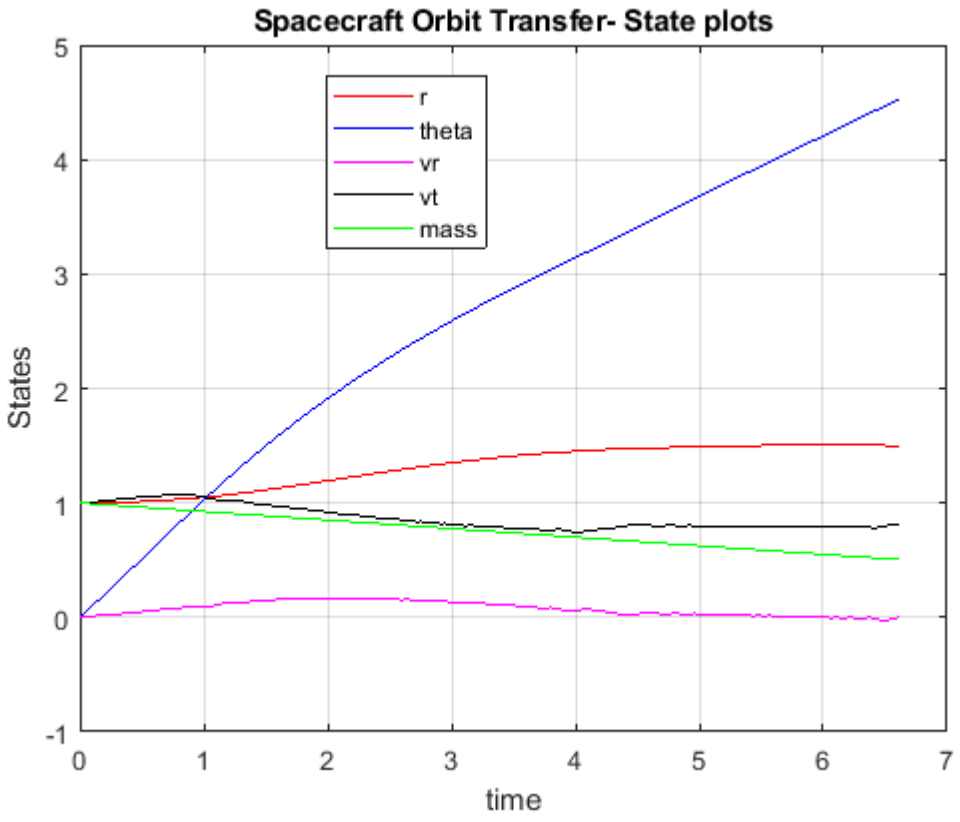


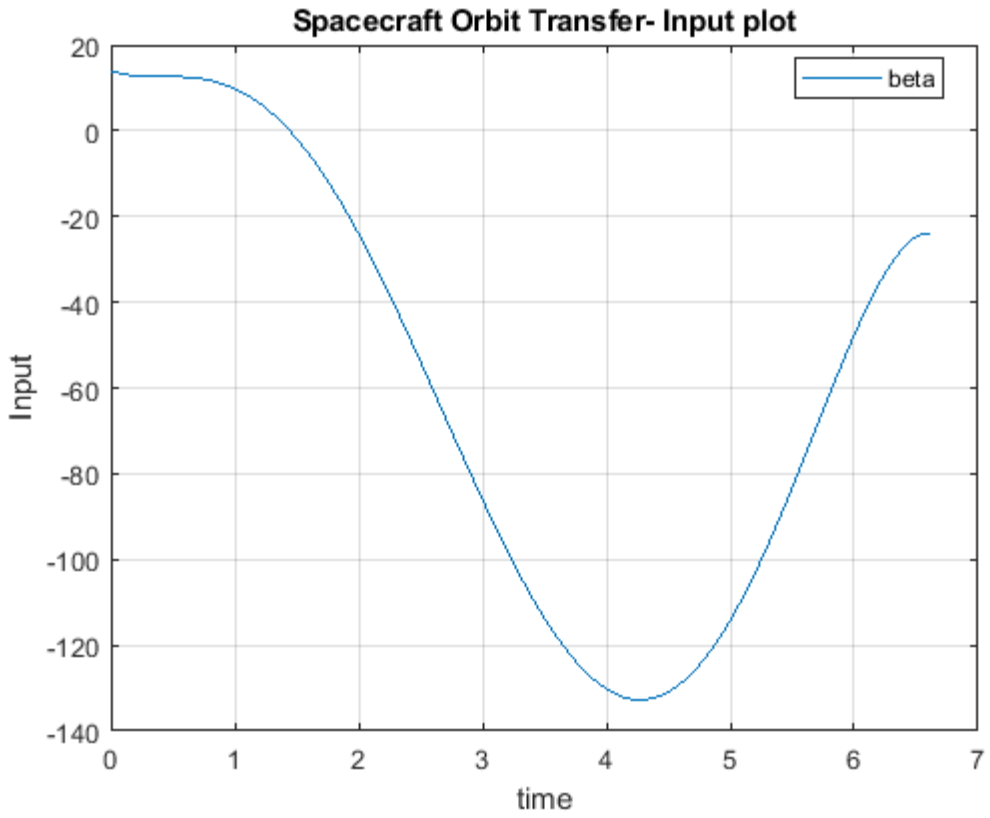


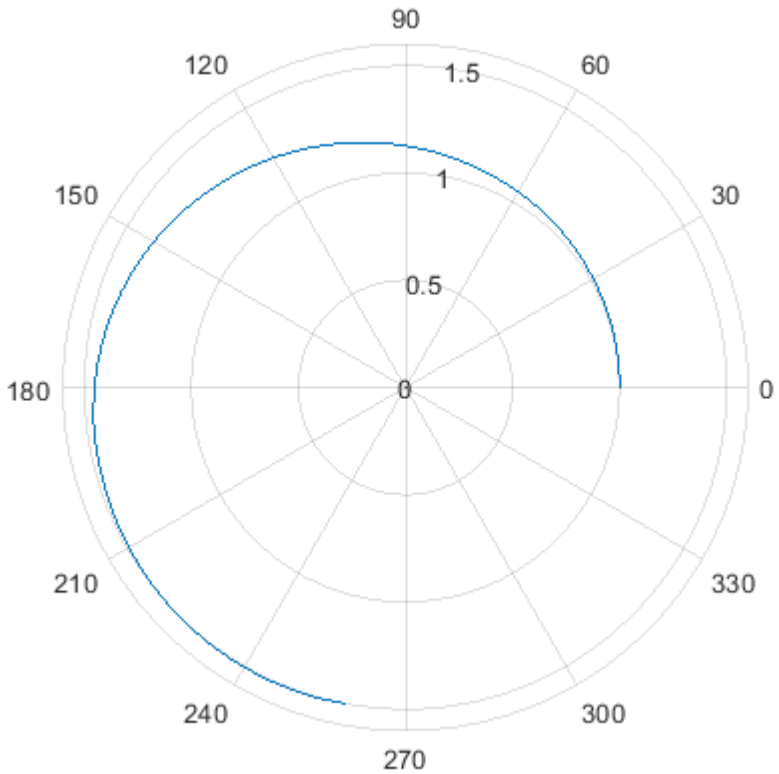


(c.4) For polynomial degree, n=5

```
  405     4607     6.612442e+00     6.072e-07     6.678e-01     5.131e-04
  406     4627     6.611970e+00     4.789e-08     8.952e-02     6.758e-03
  407     4649     6.611779e+00     2.110e-08     8.792e-02     2.957e-03
  408     4665     6.611587e+00     6.981e-09     8.628e-02     2.957e-03
  409     4685     6.611540e+00     1.589e-10     8.587e-02     7.391e-04
  410     4707     6.611519e+00     6.000e-11     8.568e-02     3.234e-04
  411     4725     6.611509e+00     2.791e-11     8.560e-02     1.617e-04
  412     4745     6.611507e+00     6.274e-12     8.554e-02     4.042e-05
  413     4769     6.611506e+00     7.718e-13     8.553e-02     2.526e-06
  414     4786     6.611506e+00     5.908e-13     8.553e-02     1.263e-06
  415     4802     6.611506e+00     3.267e-13     7.027e-01     6.316e-07

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.
```

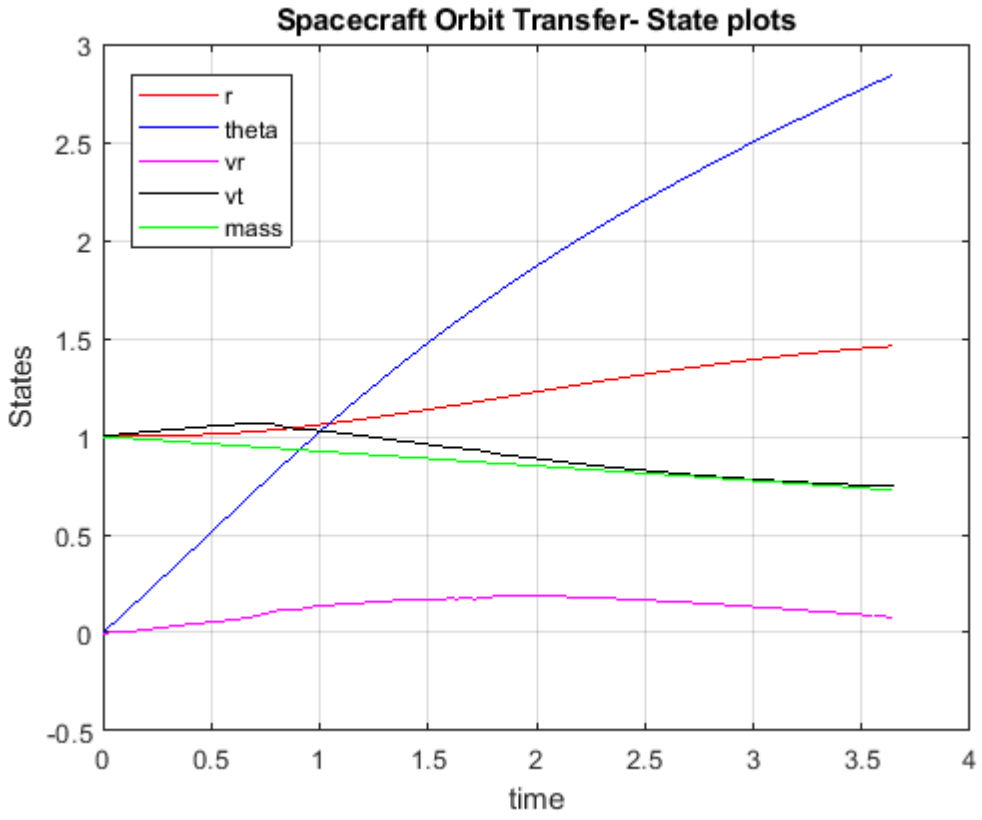


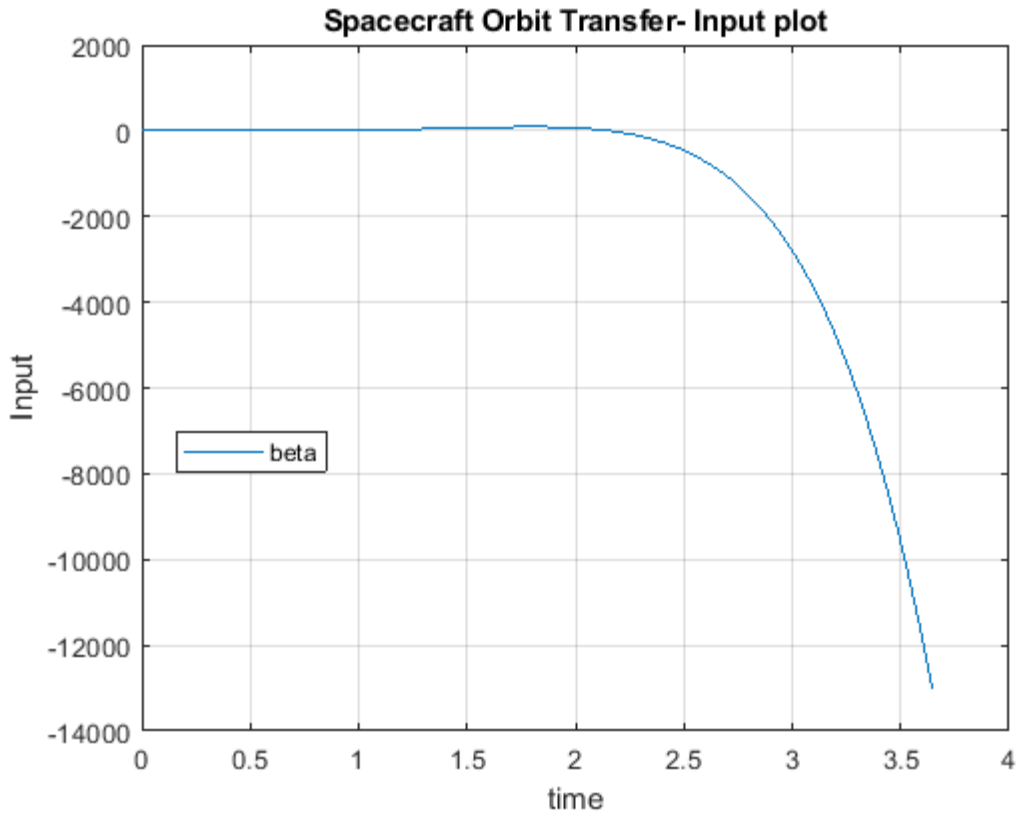


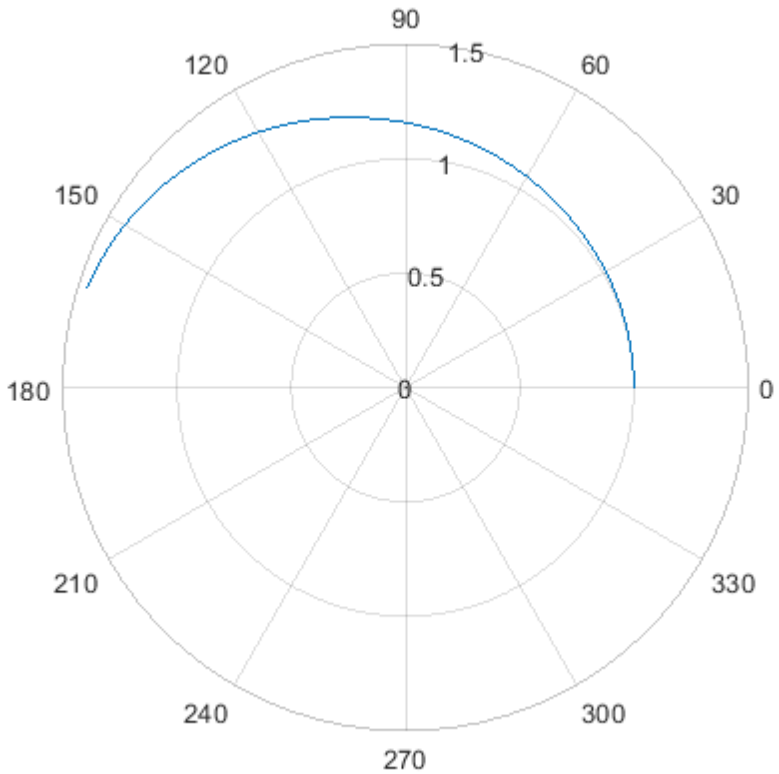


(c.5) For polynomial degree, n=6

```
107     1048     2.924083e+00     2.462e-01     2.752e+00     7.014e-02
  108     1057     2.924072e+00     2.462e-01     8.543e-01     9.647e-04
  109     1067     2.910796e+00     2.462e-01     6.401e-01     4.096e-01
  110     1078     2.937550e+00     2.456e-01     3.407e+01     8.309e-01
  111     1090     2.982011e+00     2.455e-01     1.650e+00     1.469e+00
  112     1102     3.158251e+00     2.222e-01     3.995e+01     3.331e+00
  113     1112     3.178031e+00     2.145e-01     5.059e+00     2.438e+00
  114     1121     3.622250e+00     8.439e-02     6.108e+01     3.207e+01
  115     1130     3.699521e+00     8.624e-02     3.309e+01     6.624e-01
  116     1159     3.699521e+00     8.624e-02     8.202e-01     1.424e-06
  117     1169     3.646214e+00     7.896e-02     5.909e+00     2.113e+00
  118     1191     3.646216e+00     7.897e-02     7.336e-01     1.544e-05

Converged to an infeasible point.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance but constraints are not
satisfied to within the value of the constraint tolerance.
```







Tabular summary of Performance –

Degree of the polynomial	Computation Time (in sec)	Number of Iterations	Objective function J (Final time, t_f)	Fmincon
2	39.3697	642	4.508051e+00	Converged to an infeasible point.
3	3.7837	41	3.344200e+00	Local minimum possible. Constraints satisfied.
4	9.3277	81	3.622673e+00	Local minimum possible. Constraints satisfied.
5	64.9306	4802	6.611506e+00	Local minimum possible. Constraints satisfied.
6	49.3293	1191	3.646216e+00	Converged to an infeasible point.

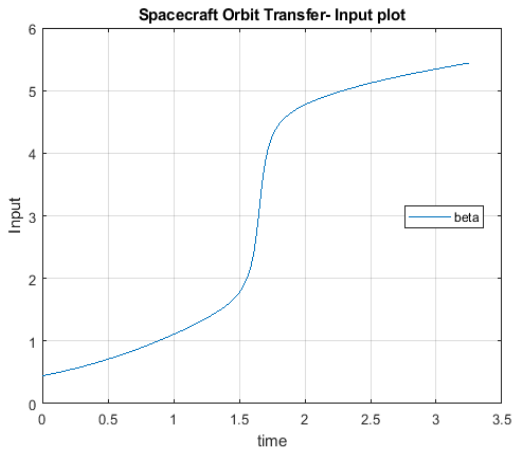
Analysis –

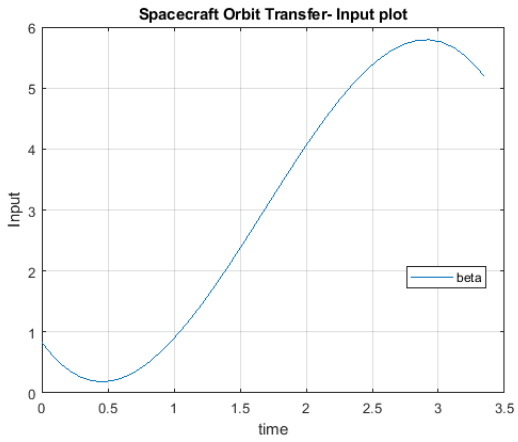
The tabular summary presented above represents the simulation results using different degrees of the polynomial used for parametrization. The polynomial degree 3 gives the closest approximate of optimal time. Since the optimal control is not a polynomial function, estimating the input with a polynomial can only give a good estimate but not actual optimal solution. The polynomial of degree 2 and 6 are converging to an infeasible point. Both degrees 3 and 4 give a close approximate to the actual optimal time at a reasonable computational time. While, n=5 took longer and was reasonably off. These results indicate that the polynomial of degree 2,5 and 6 were not a good approximate of the input function.

Direct Single Shooting vs Indirect Single Shooting

Indirect method solved for optimal solution instantly and the polar plots indicate the final position of the spacecraft is at (1.5, 141.038) while in the indirect method, the position of the spacecraft is at (1.5, 144.672) which indicates that the spacecraft reaches the final energy orbit faster using the indirect method compared to the direct one.

The figure on the left is the input plot obtained from indirect shooting method and the figure to the right is the input plot obtained from direct shooting method using a polynomial of degree 3.





The input plot obtained using direct method shows a almost linear change in input between time 0.5sec to 3 sec. While in the input plot obtained from the indirect method, there appears to be a switch in the input.

(d) Direct Multiple Shooting

Solution and Plots –

(d.1) For number of intervals, K=2 and polynomial degree n=2

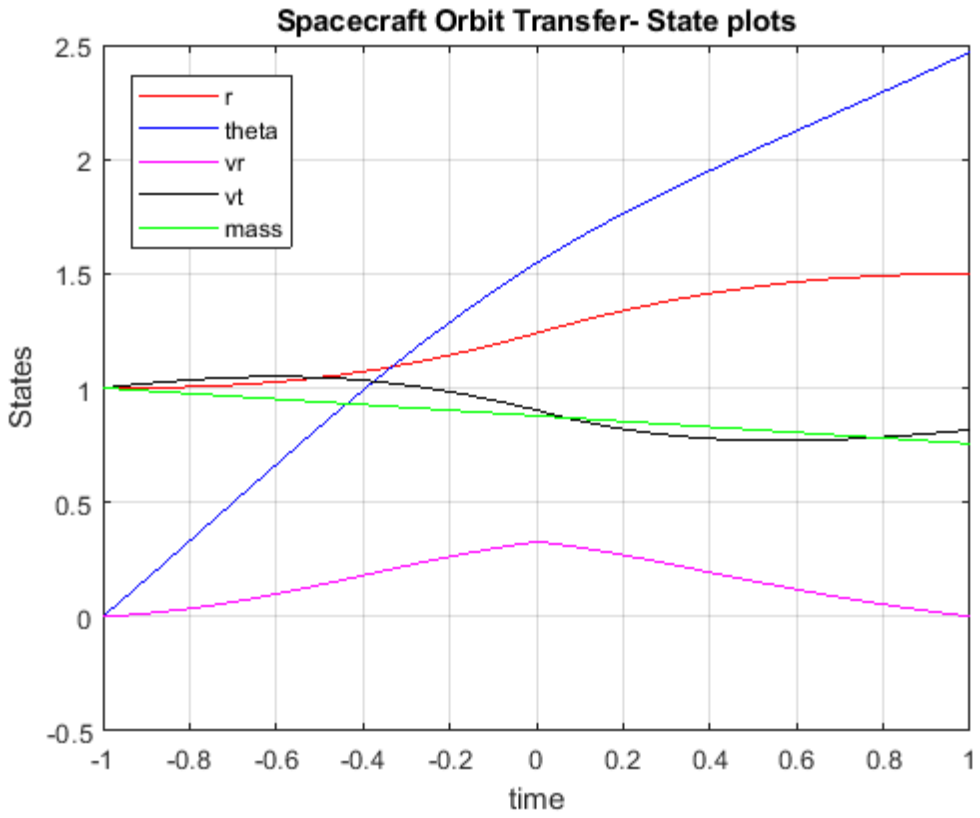
```
clear all; close all; clc;
%Spacecraft_Orbit_Transfer_Main
%Know conditions
comp_t0 = tic;
T       = 0.1405;
Ve      = 1.8758344;
mu      = 1;
t_0     = 0; %initial time/start time
nx      = 5; %no of states
K       = 2; %no of intervels
n       = 2; %degree of the polynomial used to parametarize the control
tfmin   = 0;
tfmax   = 10;
zmin    = [-50*ones((K*(n+1)+(K-1)*nx),1); tfmin]; %setting limits for fmincon
zmax    = [50*ones((K*(n+1)+(K-1)*nx),1); tfmax];
cguess  = ones(n+1,K);
P_guess = ones(nx,K-1); %guesses for intervels
tfguess = 3;
tau     = linspace(-1,1,K+1);
zguess  = [cguess(:); P_guess(:); tfguess];
A       = [];
B       = [];
Aeq     = [];
Beq     = [];
```

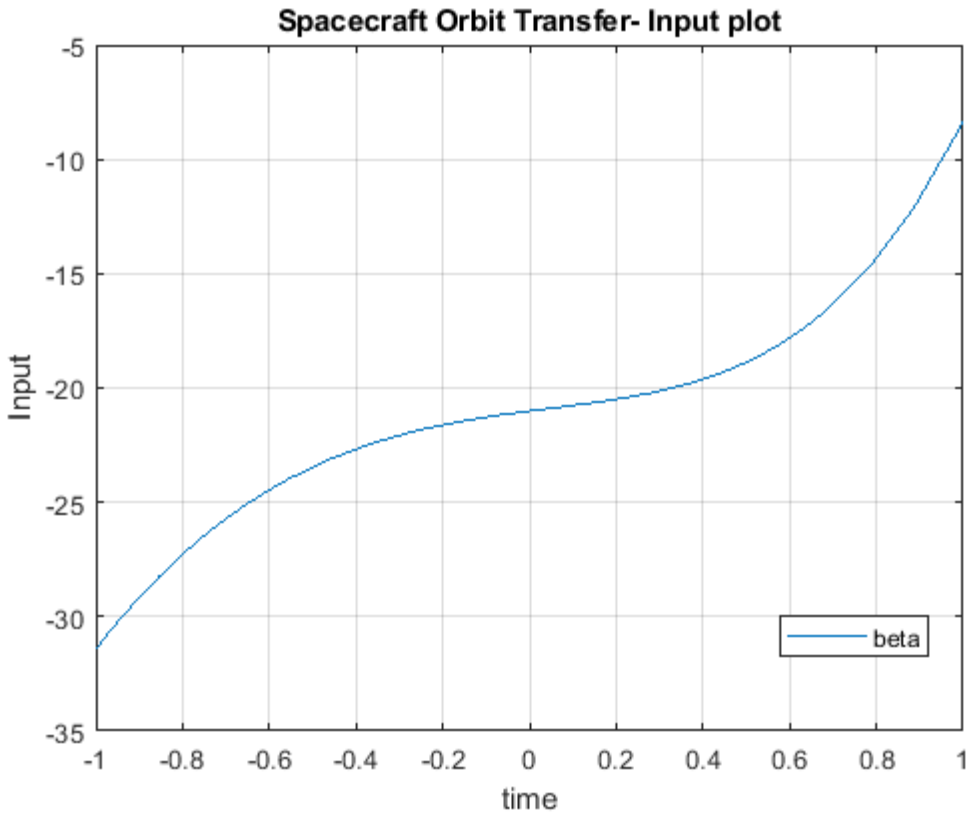
```
options = optimset('Display','Iter','TolX',1e-8,'TolFun',1e-8, 'MaxFunEvals', 100000);

z = fmincon(@OTObj, zguess, A, B, Aeq, Beq, zmin, zmax, @OTError, options, mu, T, Ve, nx, K, tau,
n, t_0);
c=z(1:K*(n+1));
[Eineq,Eeq,t,p] = OTError(z,mu,T,Ve, nx, K, tau, n, t_0);
beta = polyval(c,t);

%Plots
figure (1)
plot(t,p(:,1),'r',t,p(:,2),'b',t,p(:,3),'m',t,p(:,4),'k', t, p(:,5),'g');
title('Spacecraft Orbit Transfer- State plots')
xlabel('time')
ylabel('States')
legend('r','theta','vr','vt','mass','Location','best');
grid on
hold on

figure (2)
plot(t,beta);
title('Spacecraft Orbit Transfer- Input plot')
xlabel('time')
ylabel('Input')
legend('beta','Location','best');
grid on
hold on

figure (3)
polarplot(p(:,2),p(:,1));
hold on;

comp_tf = toc(comp_t0);

%Spacecraft_Orbit_Transfer_ODE
function [P_dot]=OTODE(t,P,mu,T,Ve,c, t_0, t_f)
r       = P(1);
theta   = P(2);
vr      = P(3);
vt      = P(4);
m       = P(5);

%solve for beta
beta    = polyval(c,t);

%first order differential equations
r_dot   = vr;
theta_dot = vt/r;
vr_dot  = (T/m)*sin(beta) + ((vt^2)/r) - mu/r^2;
vt_dot  = (T/m)*cos(beta) - vr*vt/r;
m_dot   = -T/Ve;

P_dot   = [r_dot; theta_dot; vr_dot; vt_dot; m_dot];
P_dot   = (t_f-t_0)/2 * P_dot;
```

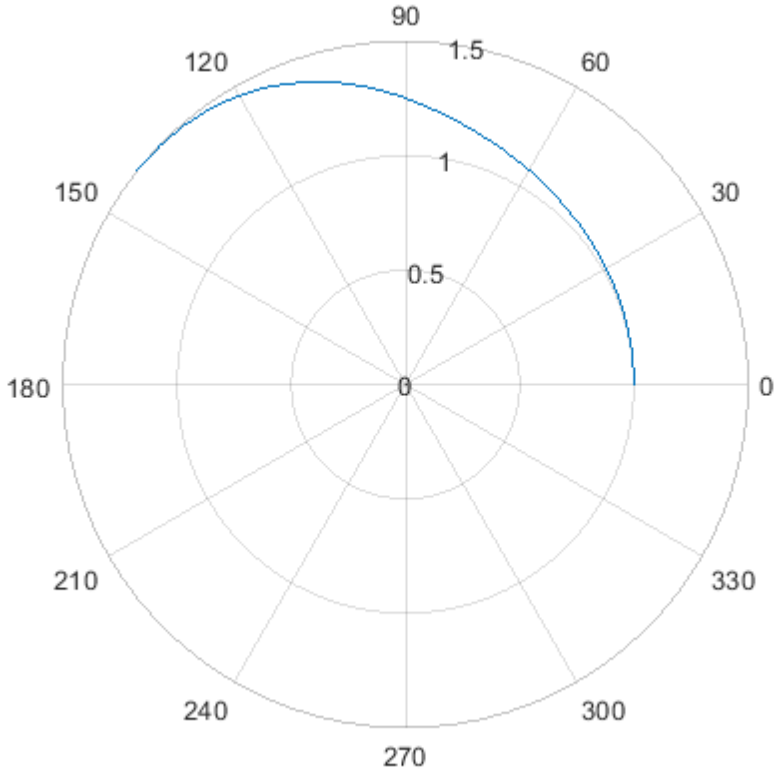
```
end

%Spacecraft_Orbit_Transfer_Obj
function J = OTObj(z,mu,T,Ve,nx,K,tau,n,t_0)
m = z(end);
J = m;
end

%Spacecraft_Orbit_Transfer_Error
function [Eineq,Eeq,t,P] = OTError(z,mu,T,Ve,nx,K,tau,n,t_0)

t_f = z(end); %terminal time
c   = z(1:(n+1)*K); %coefficient guesses of the polynomial
c   = reshape(c,n+1,K);
P_guess = z((n+1)*K+1 : end-1);
P_guess = reshape(P_guess, nx, K-1);

%boundary conditions
r_0     = 1;
r_f     = 1.5;
theta_0 = 0;
%theta_f = free;
vr_0    = 0;
vr_f    = 0;
vt_0    = sqrt(mu/r_0);
vt_f    = sqrt(mu/r_f);
m_0     = 1;
%m_f = free;
options = odeset('RelTol', 1e-8);
E       = [];
t       = [];
P       = [];

for k=1:K
    if k==1
        P_0 = [r_0; theta_0; vr_0; vt_0; m_0];
    else
        P_0 = P_guess(:,k-1);
    end

    tspan = [tau(k), tau(k+1)];
    [tout,Pout] = ode113(@OTODE, tspan, P_0, options, mu, T, Ve, c(:,k), t_0, t_f);
    Ptf = Pout(end,:).';

    if k<K
        E = [E; Ptf-P_guess(:,k)];
    end
    t = [t;tout];
    P = [P;Pout];
end
r_tf    = Ptf(1,:);
vr_tf   = Ptf(3,:);
vt_tf   = Ptf(4,:);
```

```
Eeq     = [E; r_tf-r_f; vr_tf-vr_f; vt_tf-vt_f];
Eineq   = [];

end
```

```
                                                First-order      Norm of
 Iter F-count            f(x)  Feasibility   optimality         step
  121    1977    3.249986e+00    3.508e-14    4.124e-06    7.522e-06
  122    1991    3.249986e+00    3.175e-14    2.170e-06    2.575e-05
  123    2005    3.249986e+00    7.949e-14    3.383e-06    6.972e-05
  124    2018    3.249986e+00    1.955e-13    7.621e-06    5.970e-06
  125    2039    3.249986e+00    8.660e-15    4.332e-06    5.970e-06
  126    2058    3.249986e+00    1.199e-14    7.145e-06    1.194e-05
  127    2077    3.249986e+00    2.354e-14    1.596e-05    2.388e-05
  128    2104    3.249986e+00    7.994e-15    1.125e-05    1.493e-06
  129    2123    3.249986e+00    1.599e-14    2.155e-06    1.493e-06
  130    2142    3.249986e+00    7.105e-15    1.658e-06    1.493e-06
  131    2173    3.249986e+00    7.716e-15    2.915e-06    2.332e-08

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.
```



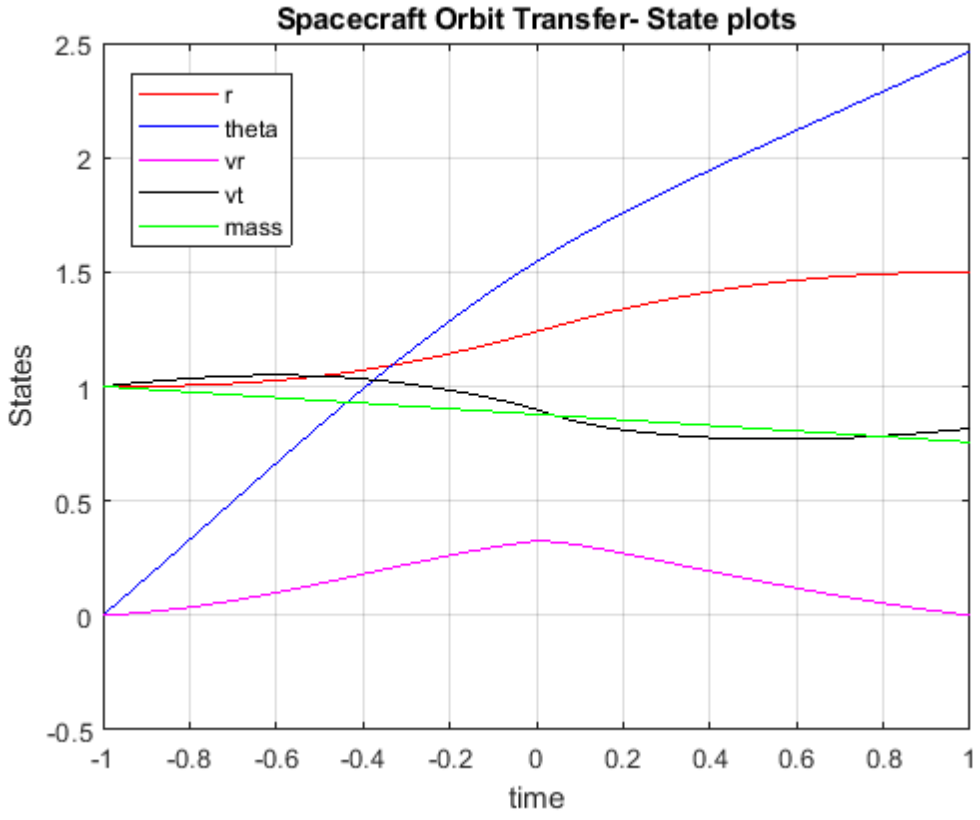


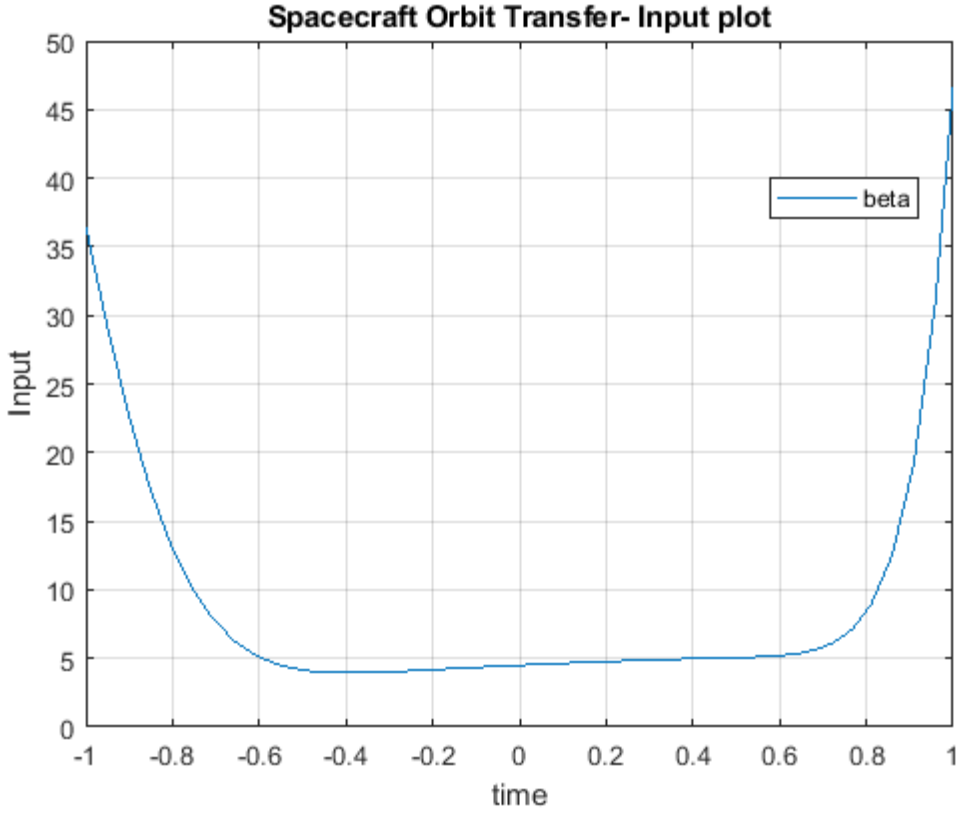


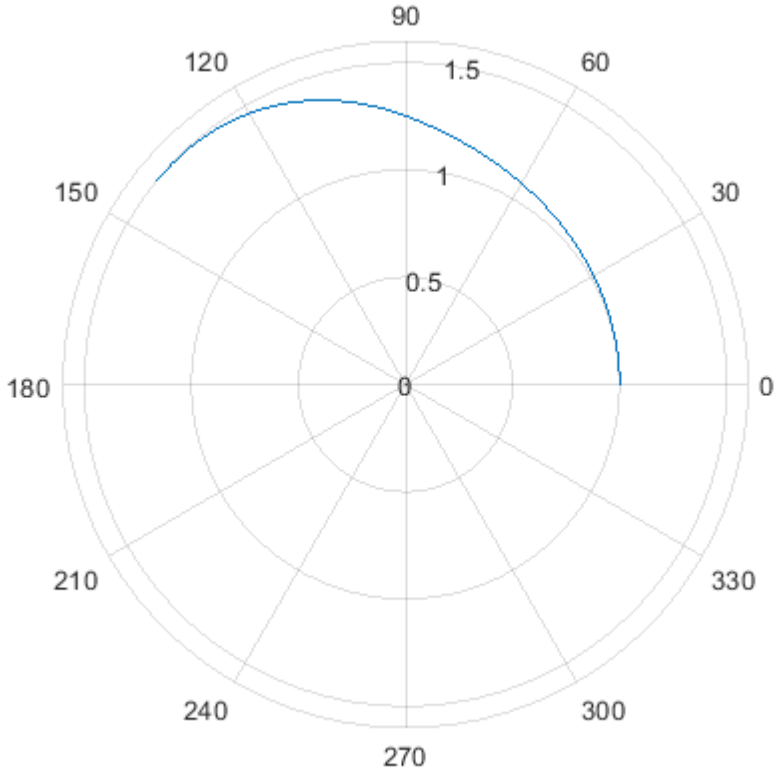
Published with MATLAB® R2021a

(d.2) For number of intervals, K=4 and degree of the polynomial, n=4

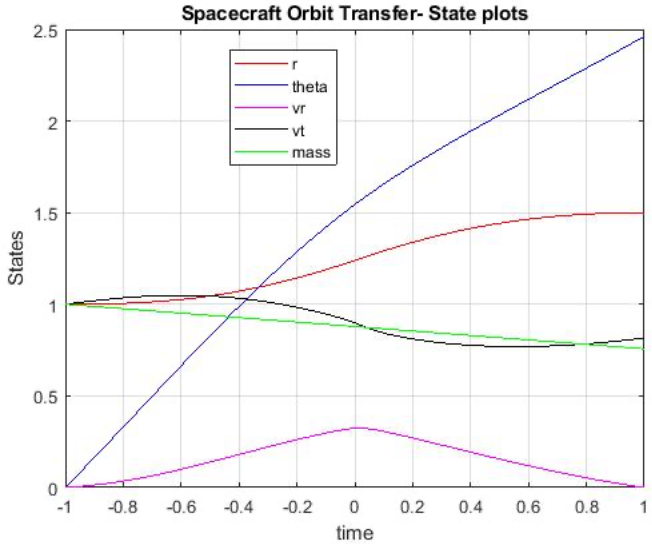
```
   343    13260    3.248410e+00    2.283e-10    1.452e-05    6.460e-03
   344    13312    3.248410e+00    4.105e-12    1.417e-05    4.192e-06
   345    13351    3.248410e+00    3.758e-12    1.363e-05    2.933e-05
   346    13401    3.248410e+00    2.735e-12    2.053e-05    3.668e-06
   347    13441    3.248410e+00    2.886e-12    1.644e-05    1.283e-05
   348    13499    3.248410e+00    1.216e-13    1.160e-01    1.003e-07
   349    13538    3.248410e+00    1.070e-13    4.345e-01    7.018e-07

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.
```

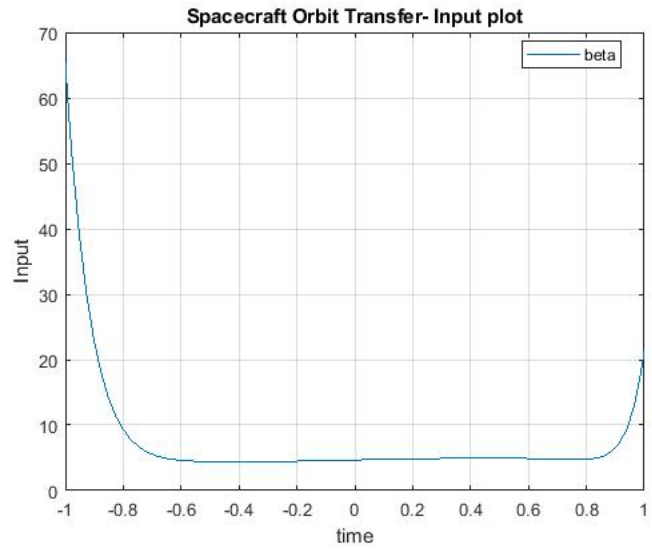


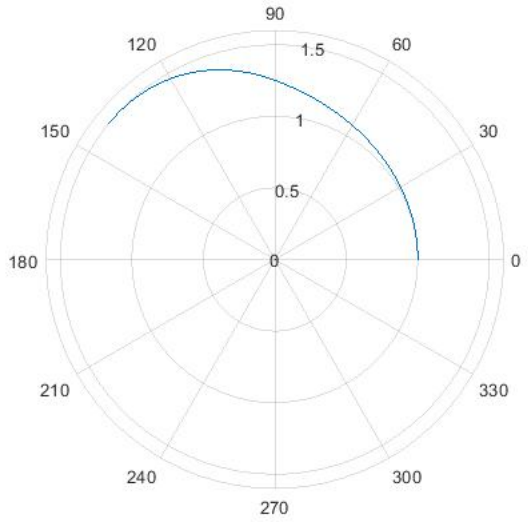

Spacecraft Orbit Transfer- Input plot
beta
Input
time
50
45
40
35
30
25
20
15
10
5
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


90
120
60
1.5
150
30
1
0.5
180
0
210
330
240
300
270

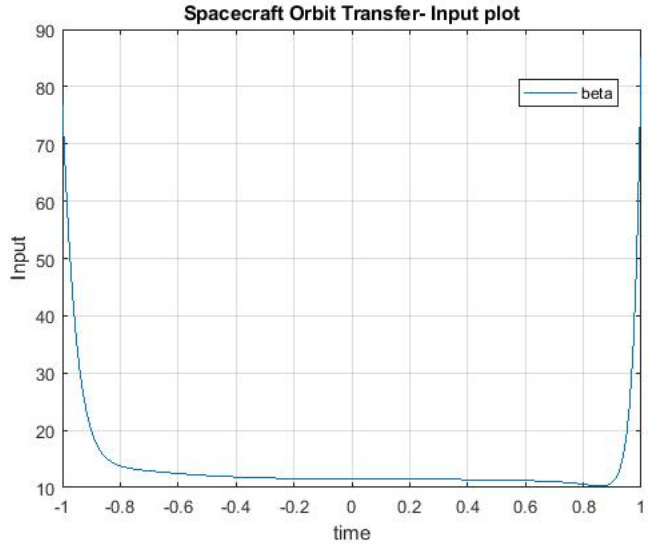
(d.3) For number of intervals, K=8 and degree of the polynomial, n=2

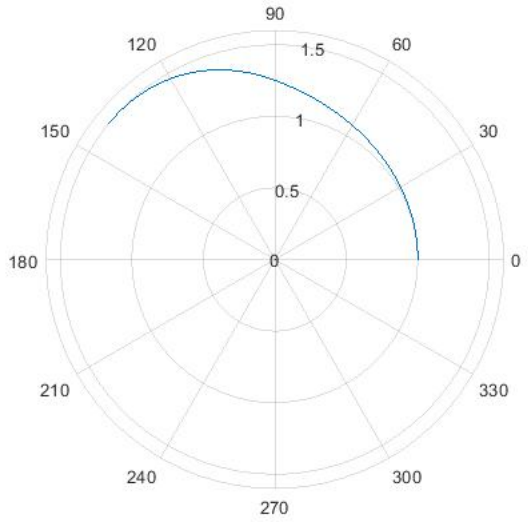


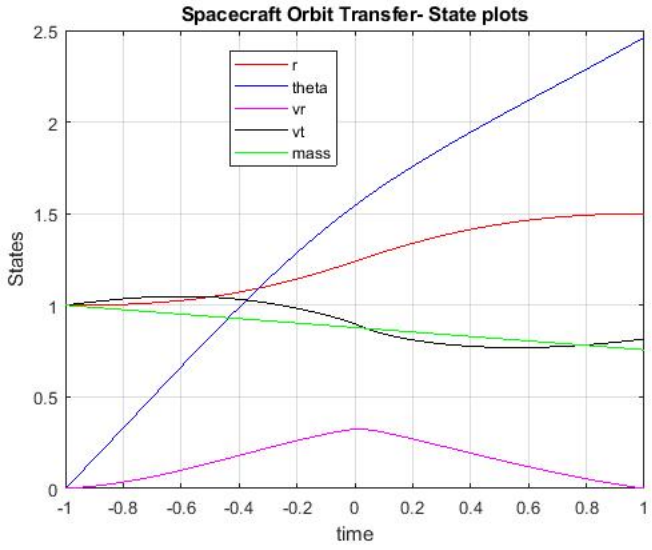




(d.4) For number of intervals, K=16 and degree of the polynomial, n=2







Tabular summary of Performance –

Degree of the polynomial	Number of Intervals	Computation Time (in sec)	Number of Iterations	Objective function J (Final time, t_f)	Fmincon
2	2	20.2085	131	3.249986e+00	Local minimum possible. Constraints satisfied.
3	2	21.8882	163	4.086292e+00	Local minimum possible. Constraints satisfied.
4	2	27.6102	207	3.855531e+00	Local minimum possible. Constraints satisfied.
5	2	24.0087	164	3.731638e+00	Local minimum possible. Constraints satisfied.
6	2	28.4342	203	3.760304e+00	Local minimum possible. Constraints satisfied.
2	4	28.8390	121	3.701893e+00	Local minimum possible. Constraints satisfied.
3	4	42.1830	205	3.628323e+00	Local minimum possible. Constraints satisfied.
4	4	57.2728	349	3.248410e+00	Local minimum possible. Constraints satisfied.
5	4	41.8415	174	3.513316e+00	Local minimum possible. Constraints satisfied.
6	4	52.8117	261	3.248454e+00	Local minimum possible. Constraints satisfied.
2	8	64.7269	175	3.248237e+00	Local minimum possible. Constraints satisfied.
3	8	172.9541	451	3.248248e+00	Local minimum possible. Constraints satisfied.
2	16	577.2558	510	3.248137e+00	Local minimum possible. Constraints satisfied.
3	16	648.4731	512	3.248135e+00	Local minimum possible. Constraints satisfied.

Analysis –

The above table presents the simulation results of direct multiple shooting method for intervals $K = \{2, 4, 8, 16\}$ using different degrees of the polynomial used for parametrization. The polynomial of degree 2 gave a very close approximation of the optimal solution for all intervals. The computational time increased with the increase in the number of intervals. As the degree of the polynomial increases the number of points at which the polynomial crosses the

horizontal axis increases implying that has more oscillations. Therefore, the polynomial with higher degree do not yield the best approximation. The input plot indicates a switch which indicates that the thrusters are switching in the opposite direction in order to slow the spacecraft as it approaches the target orbit.