PSO1

Given: CR3BP, ⊕-☾ and position and velocity of s/c.

$$\bar{r}_0 = 0.488\,\hat{x} + 0.200\,\hat{y} \qquad \bar{v}_0 = -0.880\,\hat{x} + 0.200\,\hat{y}$$

Find a) IC in dimensional units

b) i) Simulate until it crosses x-axis. Include 4×4 Φ
Time non-dim / dim?
Φ at t_f?

ii) Check Φ_{13}, Φ_{23}, Φ_{42}, Φ_{16}. Use this info to predict changes in final state due to variation initial state.
Estimate change in x_f as a results of a +1% $\dot{y}_0$ in km/s

c) Refine $\delta\bar{r}_0 = \begin{pmatrix}\delta x_0\\ \delta y_0\end{pmatrix}$, $\delta\bar{v}_0 = \begin{pmatrix}\delta \dot{x}_0\\ \delta \dot{y}_0\end{pmatrix}$. Change IC to investigate impact.

i) Case #1 $\begin{cases}\delta x_0 = 1\%\ \text{of}\ x_0\\ \delta y_0 = \delta \dot{x}_0 = \delta \dot{y}_0 = \delta t_f = 0\end{cases}$

ii) Case #2 $\begin{cases}\delta \dot{y}_0 = 1\%\ \text{of}\ \dot{y}_0\\ \delta x_0 = \delta y_0 = \delta \dot{x}_0 = \delta t = 0\end{cases}$

What change in final state ($\delta\bar{r}_f$, $\delta\bar{v}_f$) in dimensioned is predicted? What change of final state actually results from numerical sim. Which elements of STM are accurate? Why? Why STM produced inaccurate results.

d) Introduce a change δt_f (1%, 10%.) What is the impact? Simulation?

Solution:

a) $\bar{r}_{dim} = \bar{r}_{non\text{-}dim} \times L^* = \boxed{1.8759\times10^5\,\hat{x} + 0.7688\times10^5\,\hat{y}\ \text{km}}$

$\bar{v}_{dim} = \bar{v}_{non\text{-}dim} \times \frac{L^*}{t^*} = \boxed{-0.9016\,\hat{x} + 0.2049\,\hat{y}\ \text{km/s}}$

b.i) $\delta\bar{x}(t) = \bar{x}(t) - \bar{x}_n(t)$ Note: we do not need z-direction

$$\therefore \delta\ddot{x} - 2\delta\dot{y} = U_{xx}\delta x + U_{yx}\delta y + \underline{U_{xz}\delta z}$$

$$\delta\ddot{y} + 2\delta\dot{x} = U_{xy}\delta x + U_{yy}\delta y + \underline{U_{yz}\delta z}$$

$$\underline{\delta\ddot{z} = U_{xz}\delta x + U_{yz}\delta y + U_{zz}\delta z}$$

Continued -

$$\therefore \delta\dot{x}(t) = A(t)\,\delta x$$

3 dimensional

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ U^*_{xx} & U^*_{xy} & U^*_{xz} & 0 & 2 & 0 \\ U^*_{yx} & U^*_{yy} & U^*_{yz} & -2 & 0 & 0 \\ U^*_{zx} & U^*_{zy} & U^*_{zz} & 0 & 0 & 0 \end{bmatrix} \qquad \delta x = \begin{bmatrix} \delta x_0 \\ \delta y_0 \\ \delta z_0 \\ \delta \dot{x}_0 \\ \delta \dot{y}_0 \\ \delta \dot{z}_0 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ U^*_{xx} & U^*_{xy} & 0 & 2 \\ U^*_{yx} & U^*_{yy} & -2 & 0 \end{bmatrix}$$

2 dimensional

Note. These U^*_{xx} change with time depending on the position

$$\delta x = \begin{bmatrix} \delta x_0 \\ \delta y_0 \\ \delta \dot{x}_0 \\ \delta \dot{y}_0 \end{bmatrix}$$

$t = 0.478$ (non-dim)

$t_{dim} = t_{non\text{-}dim} \times t^* = 4.076$ days

$$\dot{\phi} = A\phi$$

Integrate with non-linear equations

$$\phi_{tf} = \begin{array}{c} \\ x_f \\ y_f \\ \dot{x}_f \\ \dot{y}_f \end{array} \begin{array}{c} \begin{array}{cccc} x_0 & y_0 & \dot{x}_0 & \dot{y}_0 \end{array} \\ \begin{bmatrix} 0.1051 & -0.0762 & \underline{0.2764} & \underline{-0.1260} \\ 4.1996 & 3.7018 & \underline{0.0644} & 1.4087 \\ -27.1627 & -21.1302 & 0.4904 & -8.3722 \\ 8.3622 & \underline{8.2587} & -1.9807 & 4.1530 \end{bmatrix} \end{array}$$

(ii)

	Value	Initial State	Final State
ϕ_{13}	0.2764	$\dot{x}_0$	x_f
ϕ_{23}	0.0644	$\dot{x}_0$	y_f
ϕ_{42}	8.2587	y_0	$\dot{y}_f$
ϕ_{14}	-0.1260	$\dot{y}_0$	x_f

continued...

The first row corresponds to x_f (0.01)

$\therefore \delta x_f = \phi_{11}\delta x_0^{\,0} + \phi_{12}\delta y_0^{\,0} + \phi_{13}\delta \dot{x}_0^{\,0} + \phi_{14}\delta \dot{y}_0$

$\delta x_f = -0.0024$

	Non dim	Dimensional (km)
x_f with ϕ	-0.2516	-9.6699×10^4
x_f with numerical	-0.2494	-9.5857×10^4

} Not a bad prediction

c) i) Case #1 we perturb the x_0 position since all the other element are zero this is an easy matrix multiplication: $\delta x_f = \phi_{11}\delta x_0$, $\delta y_f = \phi_{21}\delta x_0 \rightsquigarrow \bar{r}_f$

$\delta \dot{x}_f = \phi_{31}\delta x_0$, $\delta y_f = \phi\,\delta x_0 \rightsquigarrow \bar{v}_f$

ii) Similarly for Case #1 we need 4th column:

$\delta x_f = \phi_{41}\delta \dot{y}_0$, $\delta y_f = \phi_{42}\delta \dot{y}_0$

$\delta \dot{x}_f = \phi_{43}\delta \dot{y}_0$, $\delta \dot{y}_f = \phi_{44}\delta \dot{y}_0$

	$\delta \bar{r}_f$ with ϕ (km)	$\delta \bar{r}_f$ with numerical (km)
Case #1	$0.1690 \times 10^3\,\hat{x}$ $+7.8864 \times 10^3\,\hat{y}$	$0.4149 \times 10^3\,\hat{x}$ $+7.8346 \times 10^3\,\hat{y}$
Case #2	$-0.1004 \times 10^3\,\hat{x}$ $+1.0843 \times 10^3\,\hat{y}$	$-0.0964 \times 10^3\,\hat{x}$ $+1.0837 \times 10^3\,\hat{y}$

ϕ from previous page used

	$\delta \bar{v}_f$ with ϕ (km/s)	$\delta \bar{v}_f$ numerical (km/s)
Case #1	$-0.1357\,\hat{x}$ $+0.0411\,\hat{y}$	$-0.1353\,\hat{x}$ $+0.0468\,\hat{y}$
Case #2	$-0.0172\,\hat{x}$ $+0.0084\,\hat{y}$	$-0.0171\,\hat{x}$ $+0.0085\,\hat{y}$

Continued...

In the two previous tables show that the predictions made by the STM matrix is reasonably good. Note that some position / velocity are better predicting in specific direction than others.
For example: In case #1 y direction is more accurately predicted compared to x direction.
example 2: The velocities in both direction are more accurately predicted for case #2 compared to case #1.

So. the specific STM is better at predicting certain parameters due to certain perturbation

Note. STM is an estimation made through linearization of the non-linear model. At times, or certain parameter may not behave in the linear fashion and so making the STM predictions inaccurate / less accurate.

d) When we introduce δt we need to modify equations a bit

$$\delta \bar{x}_t = [\Phi]\, \delta \bar{x}_0 + [\underbrace{\dot{x}, \dot{y}, \dot{z}}_{\text{numerical solution}}, \underbrace{\ddot{x}, \ddot{y}, \ddot{z}}_{\text{non-linear equations}}]^T \delta t_f$$

δt_f	$\delta \bar{r}_f$ from Φ (km)	$\delta \bar{r}_f$ from numerical (km)	$\delta \bar{v}_f$ from Φ (km/s)	$\delta \bar{v}_f$ from numerical (km/s)
1%	$-1.4744 \times 10^3\ \hat{x}$ $-3.9292 \times 10^3\ \hat{y}$	$-1.4182 \times 10^3\ \hat{x}$ $-3.9220 \times 10^3\ \hat{y}$	$0.0724\ \hat{x}$ $-0.3553\ \hat{y}$	$-0.7686\ \hat{x}$ $-1.6372\ \hat{y}$
10%	$-1.4744 \times 10^4\ \hat{x}$ $-3.9292 \times 10^4\ \hat{y}$	$-0.9877 \times 10^4\ \hat{x}$ $-3.7909 \times 10^4\ \hat{y}$	$0.7237\ \hat{x}$ $-3.5316\ \hat{y}$	$0.4979\ \hat{x}$ $0.1807\ \hat{y}$

Continued...

Change in Δt does not produce good predictions. It just holds the acceleration constant to predict position and velocity. This might be okay at very short / small Δt but otherwise it is inaccurate.

The position is reasonably well predicted but the velocity is poorly predicted

Table of Contents

PSD1

```
clear
close all
clc

SS = SolarS;
systems = {'-', 'Earth-Moon'};
param = {'l* (km)','m* (kg)','miu' , 't*'};
G = 6.6738*10^-20;
r = [.488 .2 0];
v = [-.88 .2 0];

dim_vals = num2cell(zeros(length(param),1));
dim_vals = [systems; param',dim_vals];

% System Constants
[dim_vals{2,2}, dim_vals{3,2}, dim_vals{5,2}] =
 charE(SS.dM_E,0,SS.mMoon/G,SS.mEarth/G);        % Earth Moon
dim_vals{4,2} = SS.mMoon/dim_vals{3,2}/G;   % miu

% Part a
r_dim = r*dim_vals{2,2};
v_dim = v * dim_vals{2,2} / dim_vals{5,2};
```

Part b i)

Non-linear propgation

```
phi_0 = eye(4);
phi_0 = phi_0(:)';

IC = [r(1:2),v(1:2),phi_0];
t_end = 1;
options=odeset('RelTol',1e-13, 'AbsTol',1e-15); % Sets integration
 tollerance

[t,y] = ode45(@cr3bp_STM_df2d,[0 t_end],IC,options,dim_vals{4,2});

t_end = t(find(y(:,2)<0,1));
```

```
r_t = y(:,1:2);
v_t = y(:,3:4);
phi_t = y(:,5:end);

phi_tf = reshape(phi_t(end,:),4,4)';
```

Part b ii)

```
dx_0 = [[r(1:2),v(1:2)].*[0 0 0.01 0]]';
dx_f = phi_tf*dx_0;
x_tf_phi = dx_f(1) + [r_t(end,1)];
x_tf_phi_dim = x_tf_phi*dim_vals{2,2};

IC = [r,v].*[1 1 1 1 1.01 1];

[~,y] = ode45(@cr3bp_df,[0 t_end],IC,options,dim_vals{4,2});

x_tf_num = [y(end,1)]';
x_tf_num_dim = x_tf_num*dim_vals{2,2};
```

Part c i)

```
dx_0 = [[r(1:2),v(1:2)].*[0.01 0 0 0]]';
dx_f = phi_tf*dx_0;
r_tf_phi_dim =  dx_f(1:2)*dim_vals{2,2};
v_tf_phi_dim = dx_f(3:4)*dim_vals{2,2}/dim_vals{5,2};

% Numerical
IC = [r,v].*[1.01 1 1 1 1 1];
[~,y] = ode45(@cr3bp_df,[0 t_end],IC,options,dim_vals{4,2});
r_tf_num_dim = [y(end,1:2)-r_t(end,:)]'*dim_vals{2,2};
v_tf_num_dim = [y(end,4:5)-v_t(end,:)]'*dim_vals{2,2}/dim_vals{5,2};
```

Part c ii)

```
dx_0 = [[r(1:2),v(1:2)].*[0 0 0 0.01]]';
dx_f = phi_tf*dx_0;
r_tf_phi_dim =  dx_f(1:2)*dim_vals{2,2};
v_tf_phi_dim = dx_f(3:4)*dim_vals{2,2}/dim_vals{5,2};

% Numerical
IC = [r,v].*[1 1 1 1 1.01 1];
[~,y] = ode45(@cr3bp_df,[0 t_end],IC,options,dim_vals{4,2});
r_tf_num_dim = [y(end,1:2)-r_t(end,:)]'*dim_vals{2,2};
v_tf_num_dim = [y(end,4:5)-v_t(end,:)]'*dim_vals{2,2}/dim_vals{5,2};
```

Part d i)

```
IC = [r,v];
[~,y] = ode45(@cr3bp_df,[0 t_end],IC,options,dim_vals{4,2});
```

```
dd = sqrt((y(end,1)+dim_vals{4,2})^2 + y(end,2)^2 + y(end,3)^2);
rr = sqrt((y(end,1)+dim_vals{4,2}-1)^2 + y(end,2)^2 + y(end,3)^2);
x_ddot_tf =  2*y(end,5) + y(end,1) - (1-dim_vals{4,2})*(y(end,1)-
dim_vals{4,2})/dd^3 - dim_vals{4,2}*(y(end,1)-1+dim_vals{4,2})/rr^3;
y_ddot = -2*y(end,4) + y(end,2) - (1-dim_vals{4,2})/dd^3 -
 y(end,2)*dim_vals{4,2}/rr^3;

phit_tf = [phi_tf , [y(end,4:5),x_ddot_tf, y_ddot]'];

dx_0 = [[r(1:2),v(1:2), t_end].*[0 0 0 0 0.01]]';
dx_f = phit_tf*dx_0;
r_tf_phi_dim =  dx_f(1:2)*dim_vals{2,2};
v_tf_phi_dim = dx_f(3:4)*dim_vals{2,2}/dim_vals{5,2};

% Numerical
IC = [r,v];
[~,yp] = ode45(@cr3bp_df,[0 t_end*1.01],IC,options,dim_vals{4,2});

r_tf_num_dim = [yp(end,1:2)-y(end,1:2)]'*dim_vals{2,2};
v_tf_num_dim = [yp(end,4:5)-y(end,4:5)]'*dim_vals{2,2}/dim_vals{5,2};
```

Part d ii)

```
dx_0 = [[r(1:2),v(1:2), t_end].*[0 0 0 0 0.1]]';
dx_f = phit_tf*dx_0;
r_tf_phi_dim =  dx_f(1:2)*dim_vals{2,2};
v_tf_phi_dim = dx_f(3:4)*dim_vals{2,2}/dim_vals{5,2};

IC = [r,v];
[~,yp] = ode45(@cr3bp_df,[0 t_end*1.1],IC,options,dim_vals{4,2});

r_tf_num_dim = [yp(end,1:2)-y(end,1:2)]'*dim_vals{2,2};
v_tf_num_dim = [yp(end,4:5)-y(end,4:5)]'*dim_vals{2,2}/dim_vals{5,2};
```

Published with MATLAB® R2018a

PSD2

Given: Same conditions as the previous problem

Find a) Numerically integrate trajectory for first time
What a time interval? Now target these positions:
(i) $x_f = -0.5$ $y_f = 0.05$ (ii) $x_f = -0.1$ $y_f = 0$ (iii) $x_f = -0.4$ $y_f = -0.1$
For each, δv_1. Fix the time. Plot results. Then every subsequent result. Reference iteration. Find $\Delta\bar{v}$, $|\Delta\bar{v}|$, list t_f, r_f, Δv in dim.
Which STM are you using? Which should you use? Why?

b) Complete same 3 cases again, $\delta\dot{x} = 0$ but $\delta t_f \neq 0$. Repeat process above. $\Delta\bar{v}$, $|\Delta\bar{v}|$, Δt

c) How far away can solution still converge? Pick far away point. Can targeter succeed? Iterations?
Do iterations move smoothly? Random? While prior to reaching local basin?

Solution

a) We can use the non-linear/exact equation to get time taken to pass x-axis

$t_{end} = \boxed{0.475 \text{ (non-dimensional)}}$

$t_{end} = t^* \times t_{end} = \boxed{2.0738 \text{ days}}$

Note: for (i), (ii), and (iii) the time is fixed

The steps we take are:

1) Find ϕ for the trajectory we have ← ϕ derivation shown in PSD1
2) Use ϕ to make a guess of Δv
3) Find new trajectory with Δv
4) Find error between target and new trajectory
   ↳ If error is too big re-iterate from (1) with new trajectory.

continued...

For step 1) You can use the differential equation

$\dot{\phi} = A\phi$

$A \rightarrow$ variational equation. Derived in previous problem

2) In our case, and most cases we cannot change initial positions.

difference in target and actual $\rightarrow \bar{e} = \bar{r}_d(t_f) - \bar{r}(t_f) = \delta\bar{r}(t_f)$

$$\begin{pmatrix} \delta\bar{r}(t_f) \\ \delta\bar{v}(t_f) \end{pmatrix} = \begin{bmatrix} \phi_{rr} & \phi_{rv} \\ \phi_{vr} & \phi_{vv} \end{bmatrix} \begin{pmatrix} \delta\bar{r}(t_0) \\ \delta\bar{v}(t_0) \end{pmatrix}$$

$\therefore \delta\bar{r}(t_f) = \phi_{rr}\,\delta\bar{r}(t_0) + \phi_{rv}\,\delta\bar{v}(t_0)$

$\therefore \delta\bar{r}(t_f) = \phi_{rv}\,\delta\bar{v}(t_0)$

target $\therefore \phi^{-1}_{rv}\,\delta\bar{r}(t_f) = \delta\bar{v}(t_0)$ ← ΔV at t_0

3) Incorporate the $\delta\bar{v}(t_0)$ into new $IC_{\Delta V}$ and simulate the exact solution

choose 10^{-8}

4) Error $= \bar{r}_d(t) - \bar{r}_{\Delta V}(t_f)$ ← from new IC condition

↳ If not satisfied → repeat with the $IC_{\Delta V}$

In this formulation you have to recalculate the ϕ. ΔV calculated on every iteration reduces error, so a new corresponding ϕ needs to be used to converge from the new solution. If you use the old ϕ it may not be able to predict the new S_{xf} for the new trajectory. These non-linear equations are sensitive to IC so you have to use new ϕ.

Continued...

	Target 1	Target 2	Target 3
	$x_f = -0.3$ $y_f = +0.05$	$x_f = -0.1$ $y_f = 0$	$x_f = -0.4$ $y_f = -0.1$
Iterations	3	6	4
$\Delta\bar{V}$ (km/s)	$-0.1705\,\hat{x}$ $0.0410\,\hat{y}$	$0.6144\,\hat{x}$ $-0.1195\,\hat{y}$	$-0.5573\,\hat{x}$ $-0.1054\,\hat{y}$
$\lvert\Delta\bar{V}\rvert$ (km/s)	0.1754	0.6259	0.5672
$\Delta\bar{V}$ (Normalized)	$-0.1664\,\hat{x}$ $0.04\,\hat{y}$	$0.5997\,\hat{x}$ $-0.1167\,\hat{y}$	$-0.5440\,\hat{x}$ $-0.1029\,\hat{y}$
$\lvert\Delta\bar{V}\rvert$ (Normalized)	0.1712	0.6109	0.5536
$\Delta\bar{r}_f$ (km)	$0.0289\times10^{-3}\,\hat{x}$ $0.1851\times10^{-3}\,\hat{y}$	$1.739\times10^{-7}\,\hat{x}$ $0.398\times10^{-7}\,\hat{y}$	$-3.951\times10^{-2}\,\hat{x}$ $-0.048\times10^{-2}\,\hat{y}$
$\Delta\bar{r}_f$ (non-dim)	$0.0753\times10^{-9}\,\hat{x}$ $0.4816\times10^{-9}\,\hat{y}$	$4.524\times10^{-13}\,\hat{x}$ $1.035\times10^{-12}\,\hat{y}$	$-1.028\times10^{-8}\,\hat{x}$ $-0.013\times10^{-8}\,\hat{y}$
t_f (days)	2.0738	2.0738	2.0738

See Figures:
Target 1 ⇒ 02.1, 02.2
Target 2 ⇒ 02.3, 02.4
Target 3 ⇒ 02.5, 02.6

b) Now we can vary time but $\dot{x}_0 = 0$ so the steps have to change slightly.

1) Find ϕ for the trajectory we have
2) Get a modified $\phi \to K$
3) Use K to find guess of ΔV
4) Find new trajectory with the ΔV
5) Find error between trajectory and desired position
   ⇒ If error is too big re-iterate from (1) with new trajectory.

Continued...

Note: All the same steps as before but for step (2) is different. ↙ get $\dot{x}$ and $\dot{y}$ from numerical simulation used to find STM/ϕ

2) $$\begin{Bmatrix} \delta x \\ \delta y \end{Bmatrix} = \begin{bmatrix} \phi_{15} & \phi_{16} & \dot{x} \\ \phi_{25} & \phi_{26} & \dot{y} \end{bmatrix} \begin{Bmatrix} \delta x \\ \delta y \\ \delta t \end{Bmatrix}$$

See Figures: Target 1 D2.7, D2.8 , Target 2 D2.9, D2.10 , Target 3 D2.11, D2.11

	$x_f = -0.3$ $y_f = 0.05$	$x_f = -0.1$ $y_f = 0$	$x_f = -0.4$ $y = -0.1$
Iterations	4	6	5
$\Delta\bar{V}$ (km/s)	$0.1005\ \hat{y}$	$-0.3652\ \hat{y}$	$0.1314\ \hat{y}$
$\lvert\Delta\bar{V}\rvert$ (km/s)	0.1005	0.3652	0.1314
$\Delta\bar{V}$ (Non-dim)	$0.0981\ \hat{y}$	$-0.3538\ \hat{y}$	$0.1282\ \hat{y}$
$\lvert\Delta\bar{V}\rvert$ (Non-dim)	0.0981	0.3538	0.1282
$\Delta\bar{r}_f$ (km)	$8.868\times10^{-7}\ \hat{x}$ $-4.402\times10^{-7}\ \hat{y}$	$0.280\times10^{-3}\ \hat{x}$ $1.229\times10^{-3}\ \hat{y}$	$-0.156\times10^{-6}\ \hat{x}$ $1.046\times10^{-6}\ \hat{y}$
$\Delta\bar{r}_f$ (Non-dim)	$2.307\times10^{-11}\ \hat{x}$ $-1.1145\times10^{-11}\ \hat{y}$	$0.728\times10^{-9}\ \hat{x}$ $3.198\times10^{-9}\ \hat{y}$	$-0.407\times10^{-11}\ \hat{x}$ $2.720\times10^{-11}\ \hat{y}$
δt_f (non-dim)	0.0493	-0.1418	0.1830
δt_f (hours)	5.138	-14.7785	19.072

As seen in PSDI STM is a poor predictor when time is involved, either the iterations increased or the accuracy fell. Showing this if you compare with previous table

continued...

c) I tried some points, If you really far away then method converges. Then numbers distances of 10^{10} ~ 10. but some But there are some point which have more trouble. This is not necessarily a function of distance but the non-linear behaviors of the system. I have provided figures which show that iterations increase based on specific locations rather than just distance. It just needs to find basin

See Figures: 02-13, 02-14, 02-15, 02-16, 02-17, 02-18

The targeter tries to look for basin and once it finds it, the convergence is smooth. But before finding basin targeter shoots all over the place

PSD2

Part ai)

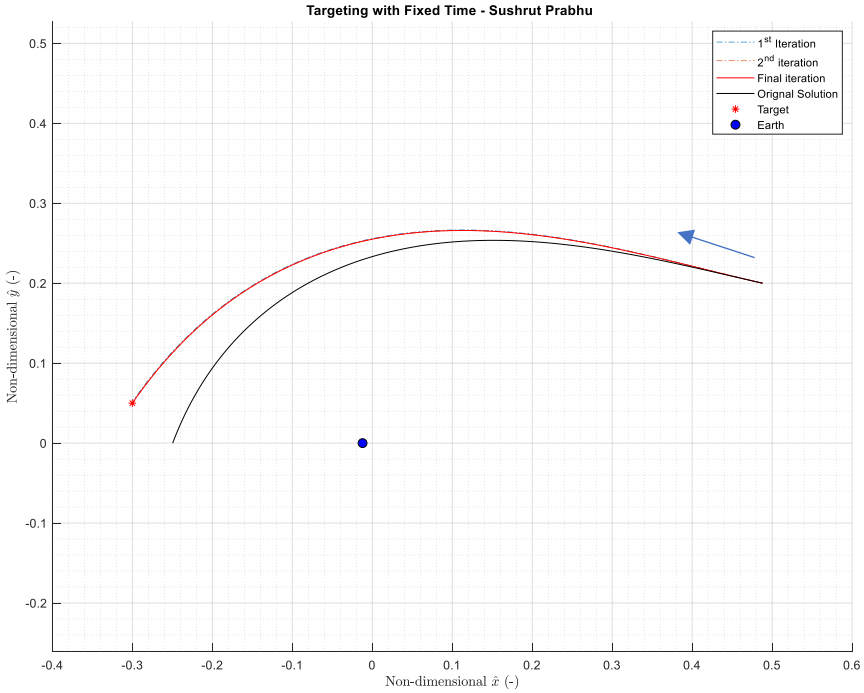


Figure D2.1: Targeting using an STM to point [x = -0.3 y = 0.05].

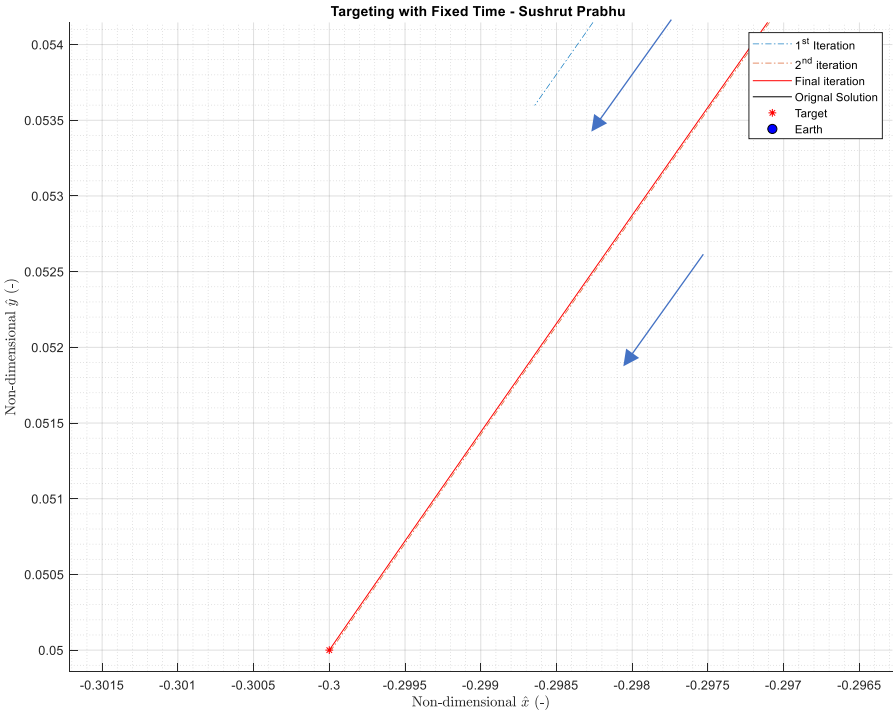


Figure D2.2: Targeting using an STM zoomed in to point [x = -0.3 y = 0.05].

Part a ii)

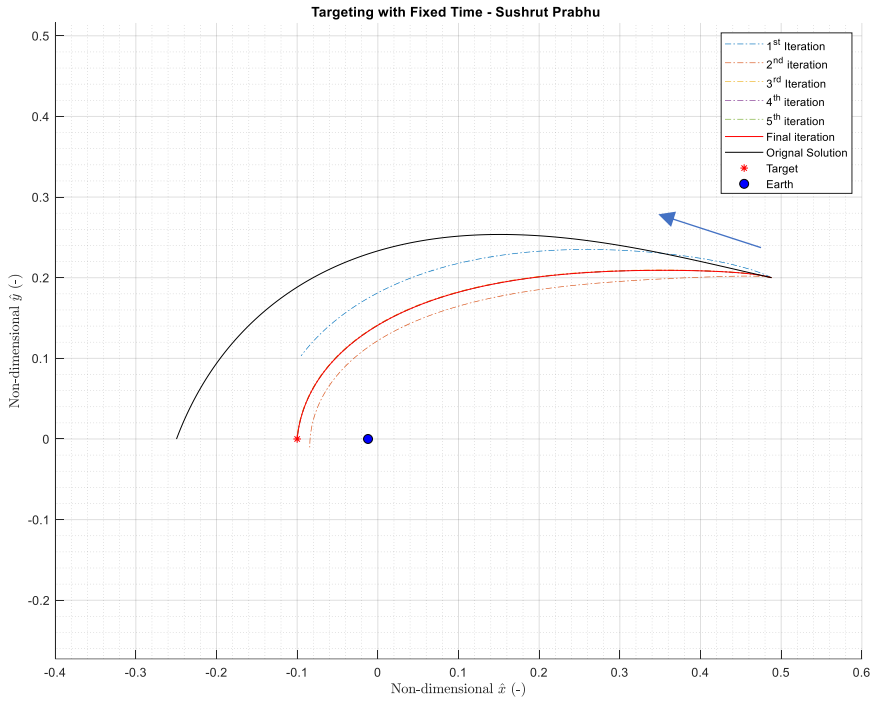


Figure D2.3: Targeting using an STM to point [x = -0.1 y = 0].

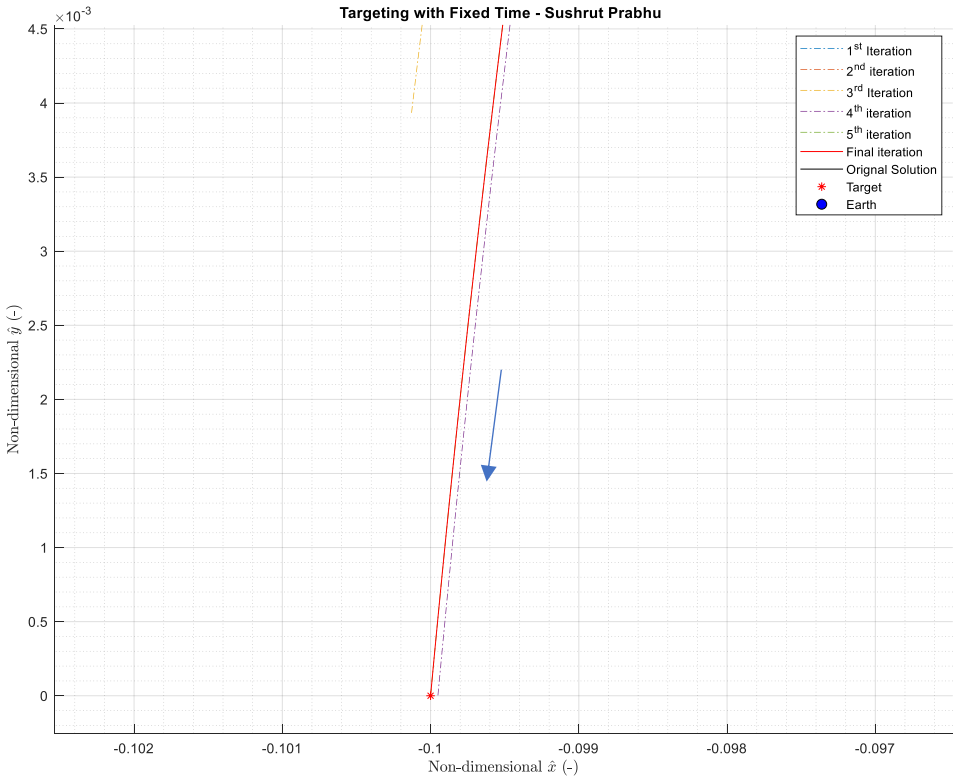


Figure D2.4: Targeting using an STM zoomed in to point [x = -0.1 y = 0].

Part a iii)

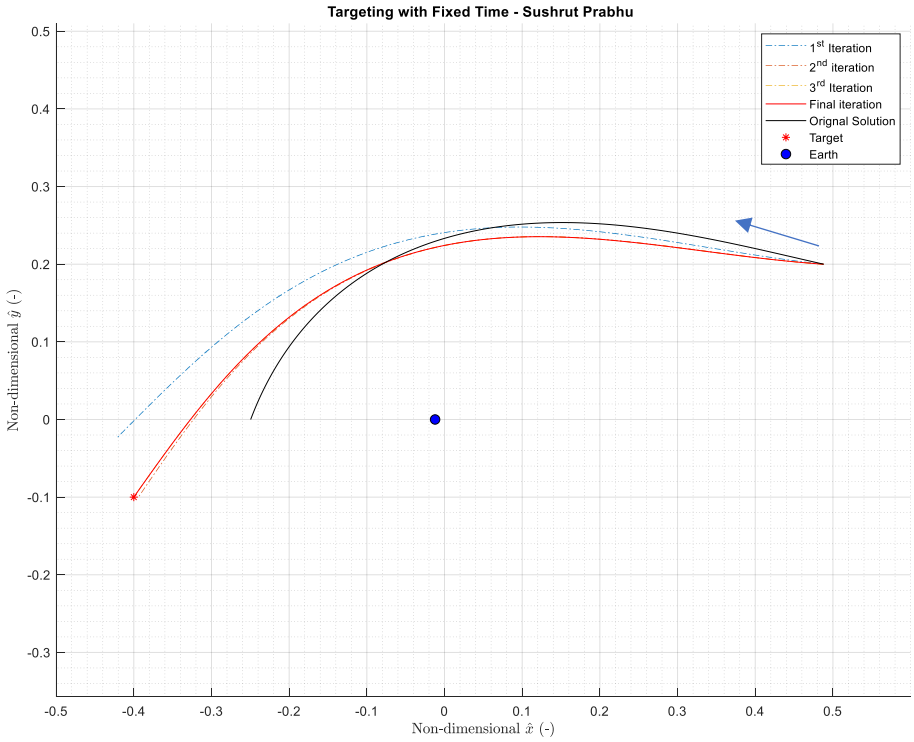


Figure D2.5: Targeting using an STM to point [x = -0.4 y = -0.1].

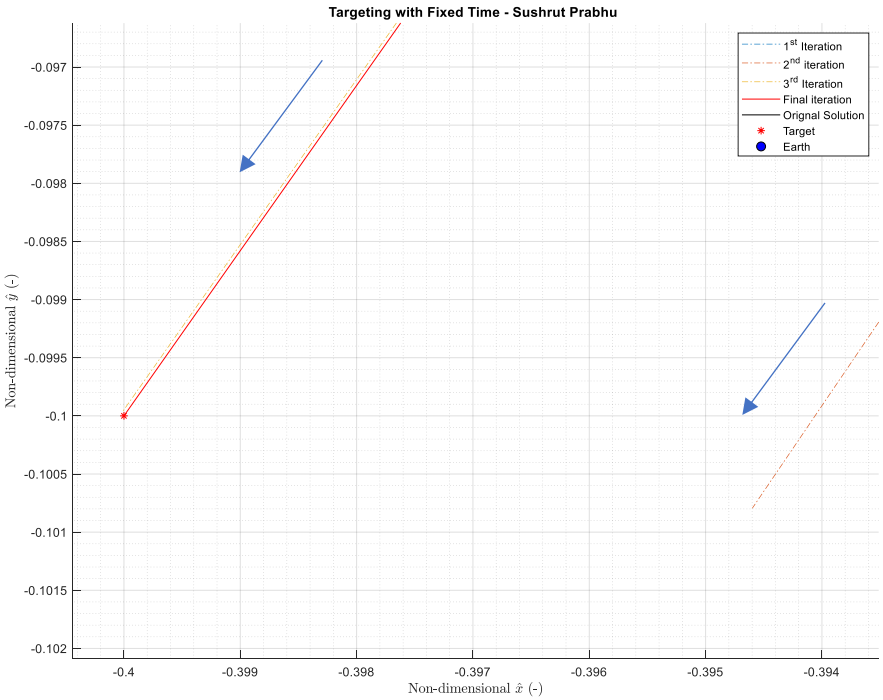


Figure D2.6: Targeting using an STM zoomed in to point [x = -0.4 y = -0.1].

Part b i)

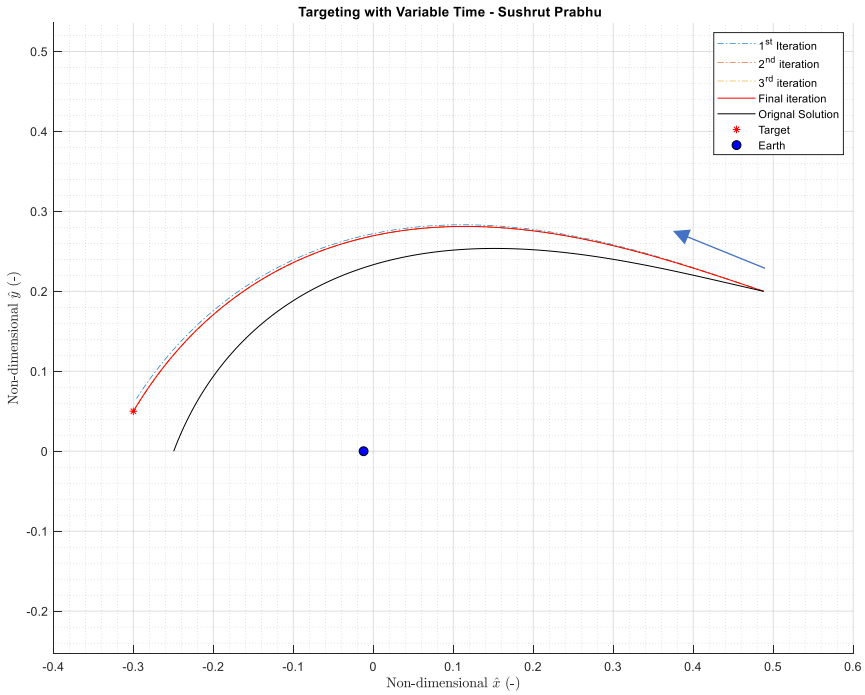


Figure D2.7: Targeting using an STM with variable time to point [x = -0.3 y = 0.05].

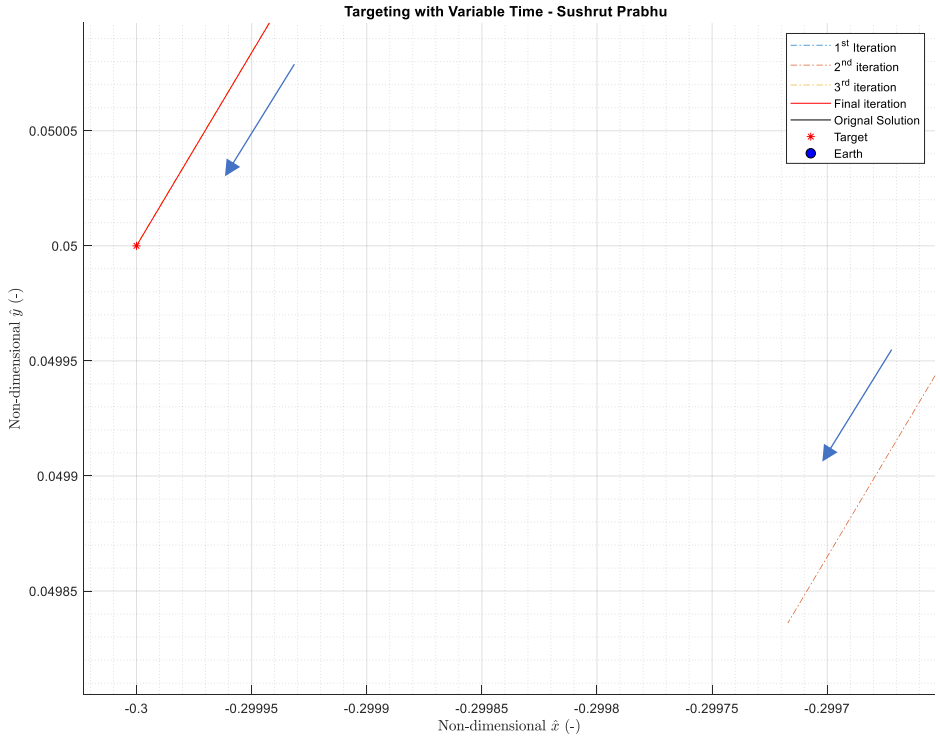


Figure D2.8: Targeting using an STM with variable time zoomed in to point [x = -0.3 y = 0.05].

Part b i)

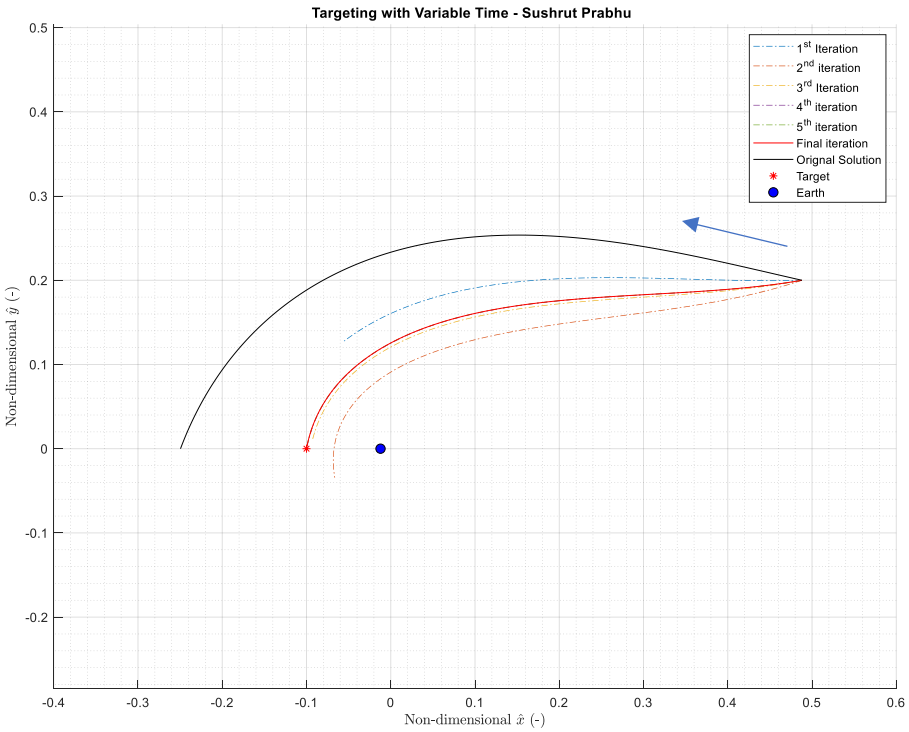


Figure D2.9: Targeting using an STM with variable time to point [x = -0.1 y = 0].

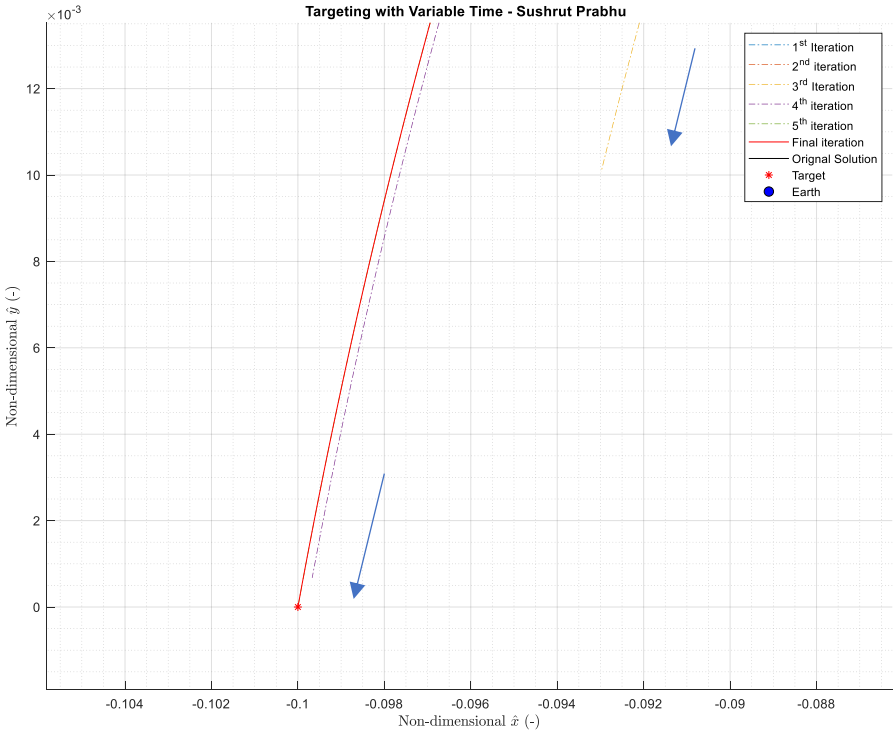


Figure D2.10: Targeting using an STM with variable time zoomed in to point [x = -0.1 y = 0].

Part b ii)

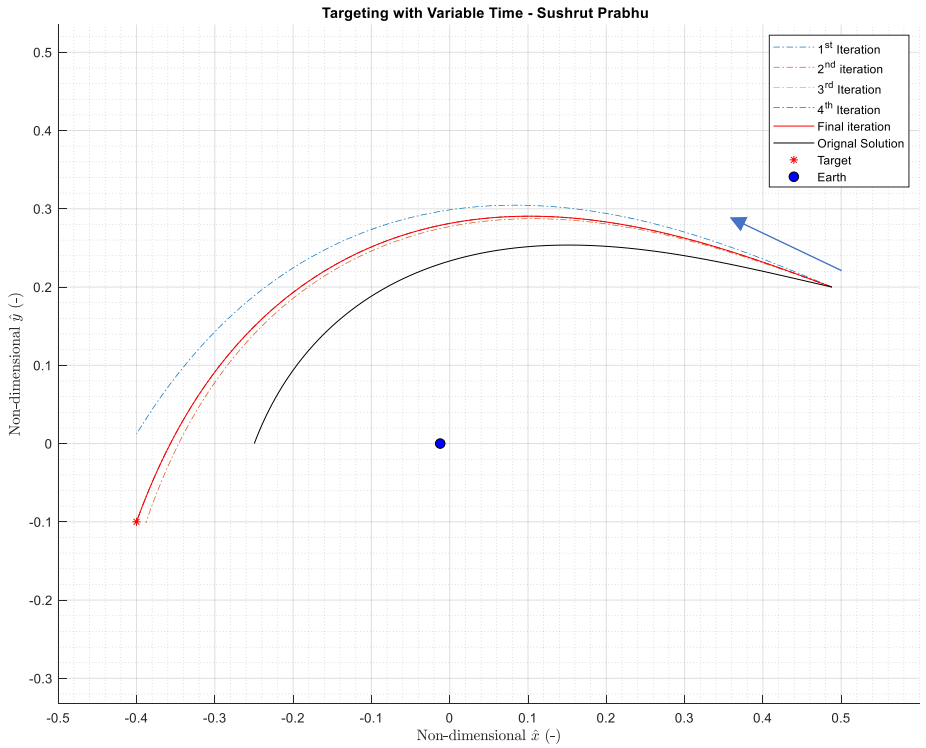


Figure D2.11: Targeting using an STM with variable time to point [x = -0.4 y = -0.1].

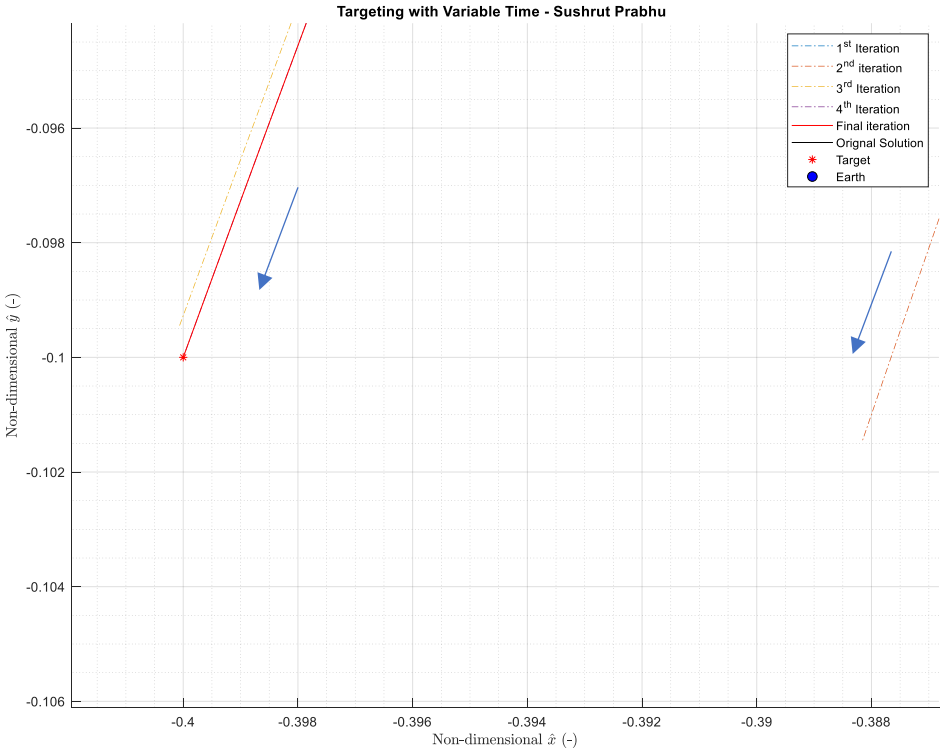


Figure D2.12: Targeting using an STM with variable time zoomed in to point [x = -0.4 y = -0.1].

Part c)

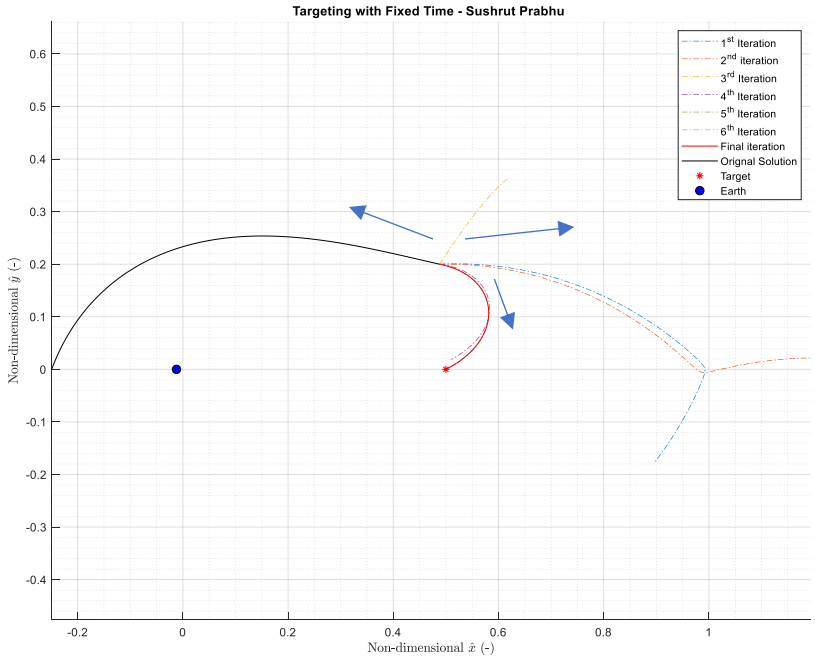


Figure D2.13: Targeting using an STM with variable time zoomed in to point [x = .5 y = 0].

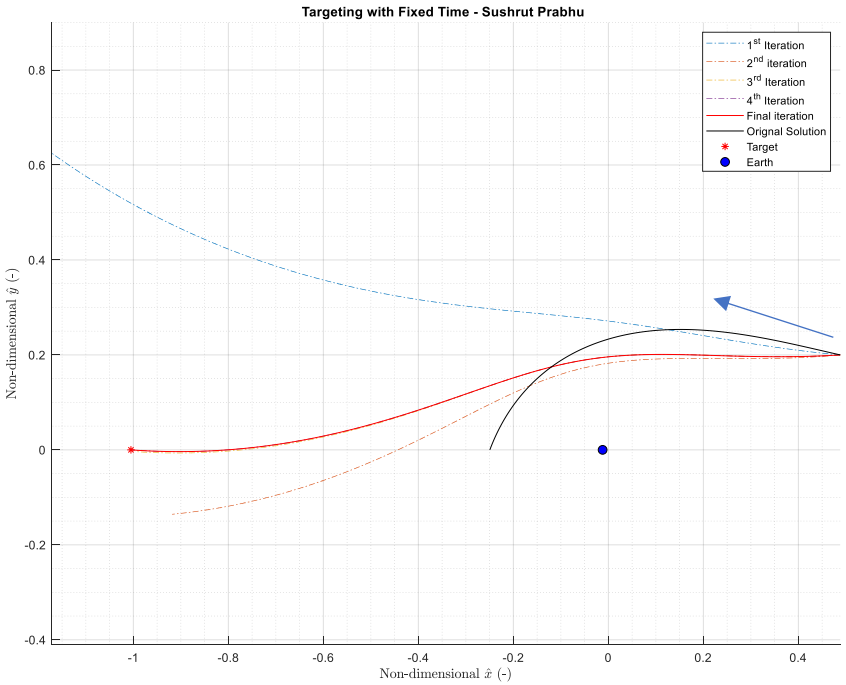


Figure D2.14: Targeting using an STM with variable time zoomed in to point L_3

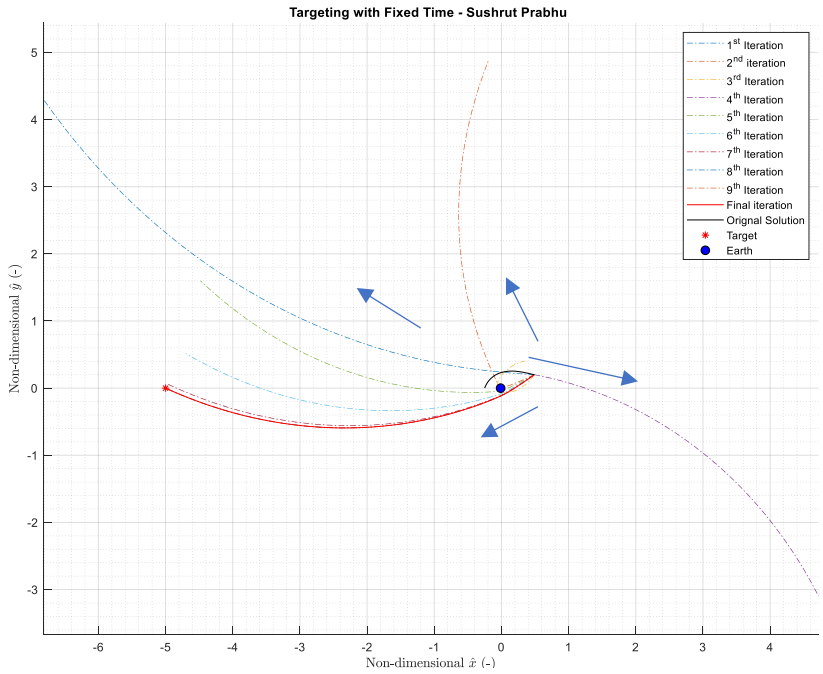


Figure D2.15: Targeting using an STM with variable time zoomed in to point [x = -5 y = 0].

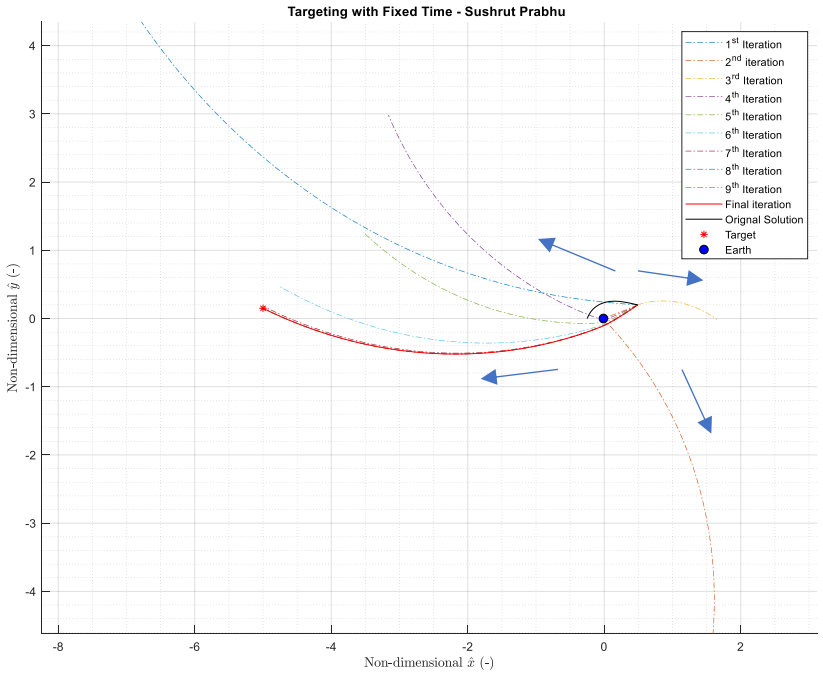


Figure D2.16: Targeting using an STM with variable time zoomed in to point [x = -5 y = 0.15].

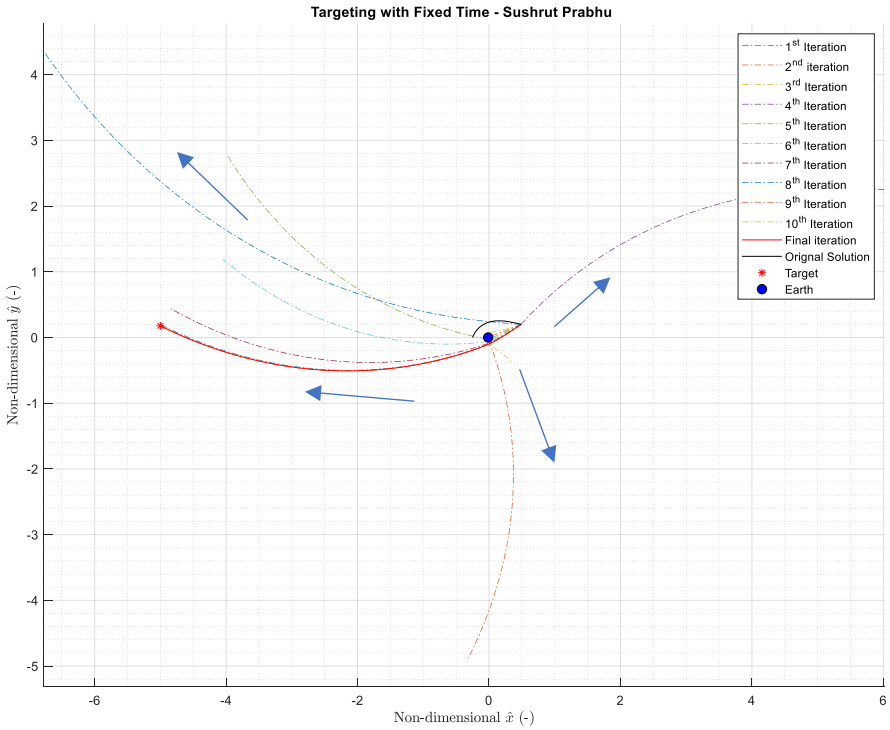


Figure D2.17: Targeting using an STM with variable time zoomed in to point [x = -0.5 y = 0.177].

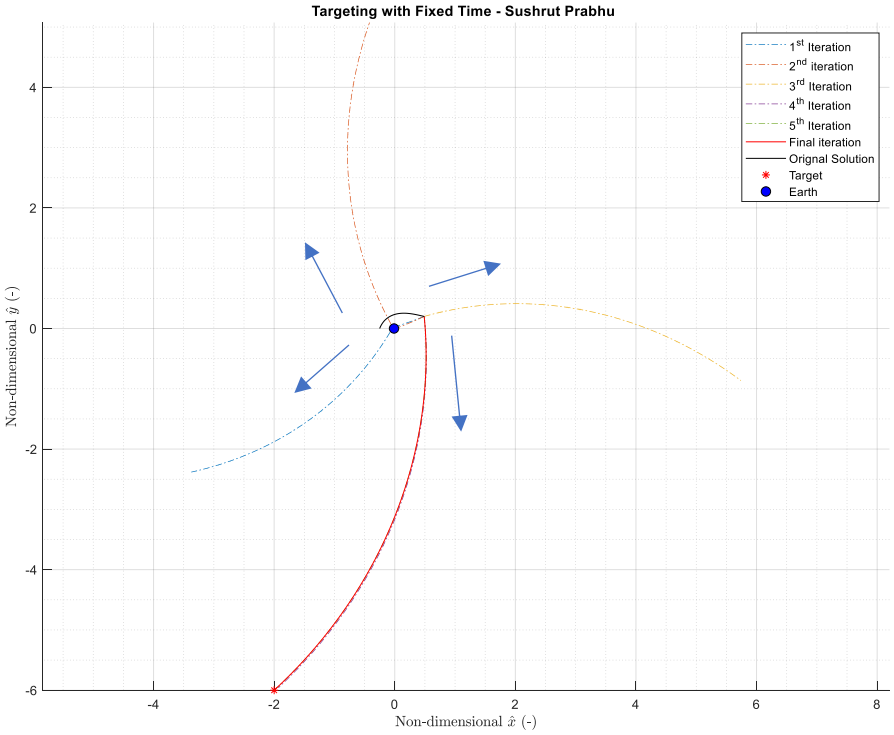


Figure D2.18: Targeting using an STM with variable time zoomed in to point [x = -2 y = -6].

Table of Contents

PSD2

```
clear
close all
clc
```

Initial Setup

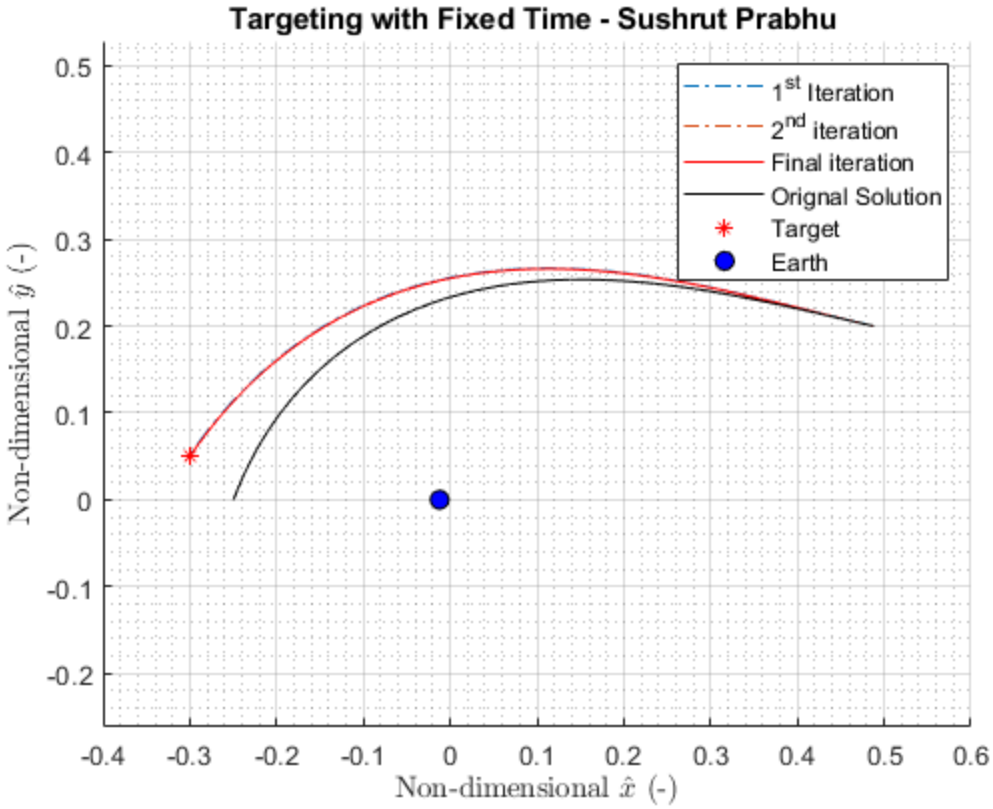
```
SS = SolarS;
systems = {'-', 'Earth-Moon'};
param = {'l* (km)','m* (kg)','miu' , 't*'};
G = 6.6738*10^-20;
options=odeset('RelTol',1e-13, 'AbsTol',1e-15); % Sets integration
 tollerance
r = [.488 .2 0];
v = [-.88 .2 0];

dim_vals = num2cell(zeros(length(param),1));
dim_vals = [systems; param',dim_vals];

% System Constants
[dim_vals{2,2}, dim_vals{3,2}, dim_vals{5,2}] =
 charE(SS.dM_E,0,SS.mMoon/G,SS.mEarth/G);        % Earth Moon
dim_vals{4,2} = SS.mMoon/dim_vals{3,2}/G;   % miu

t_end = .47755;

% Orignal
t = 0:.001:t_end;
IC = [r,v];
[~,yo] = ode45(@cr3bp_df,[0 t_end],IC,options,dim_vals{4,2});
```

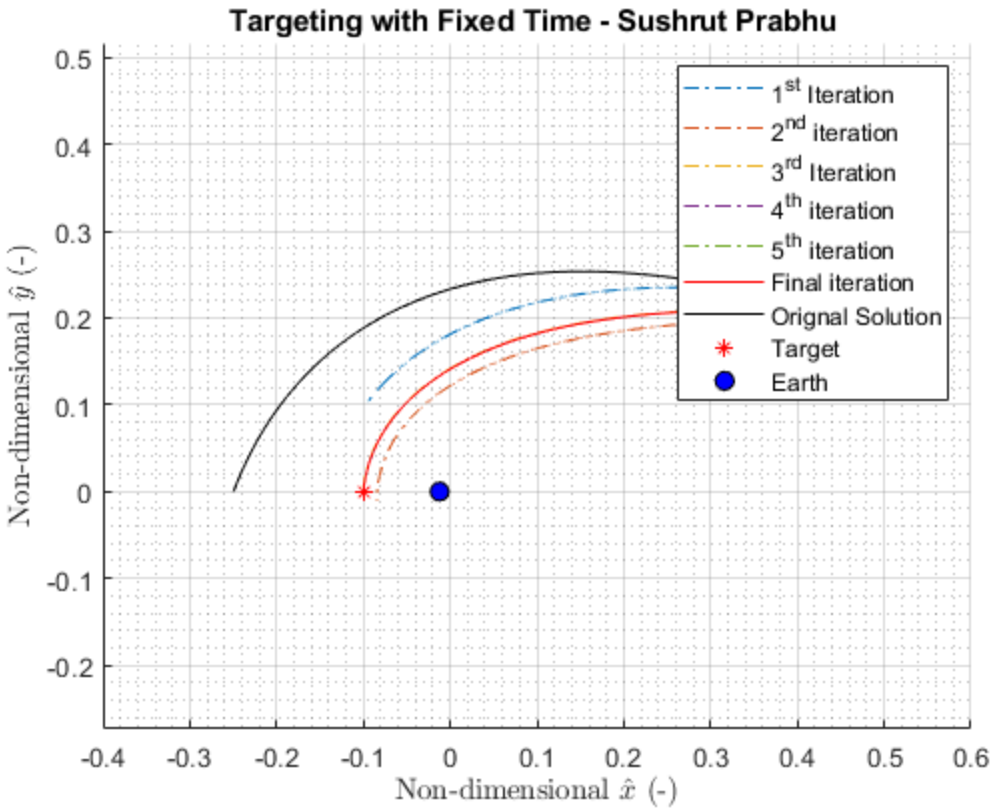
Part a

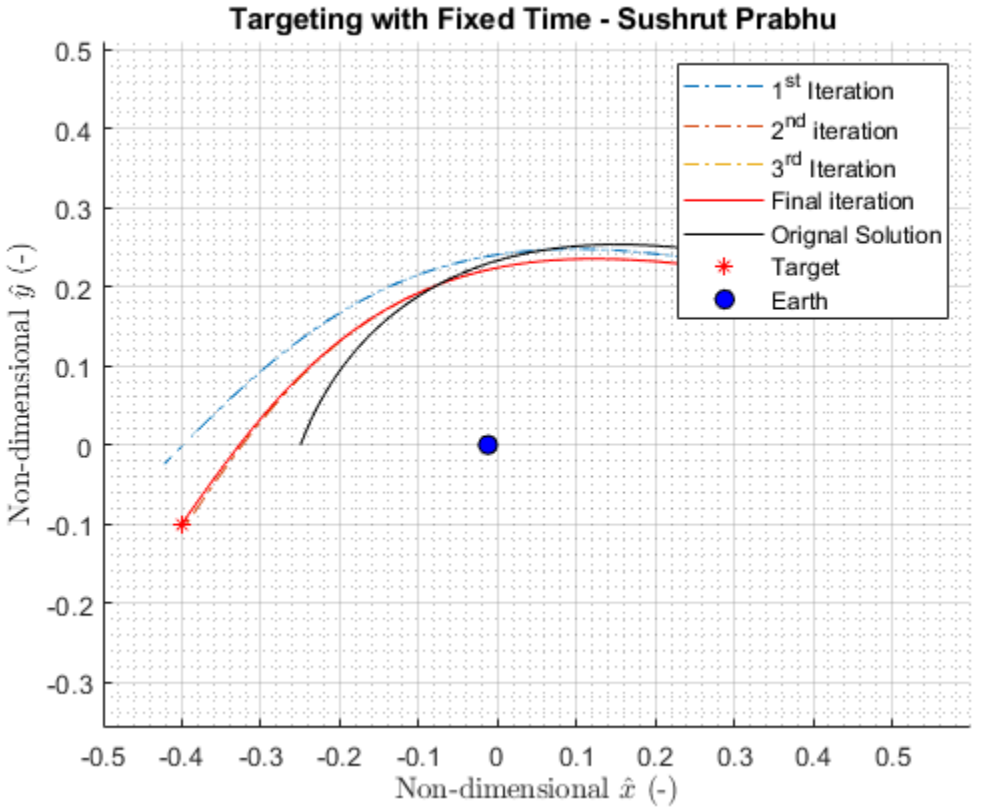
```
r1_des = [-.3 .05]';
yn1 = Target2d_tf_phi(r1_des,r,v,t_end,dim_vals{4,2},0,10^-8);
delv1 = yn1(1,4:6)-yo(1,4:6);

plot(yo(:,1),yo(:,2),'k')
plot(r1_des(1),r1_des(2),'*r')
plot(-dim_vals{4,2},0,'ko','MarkerSize',7,'MarkerFaceColor','b')
xlim([-.4 .6])
```

```
legend('1^{st} Iteration', '2^{nd} iteration','Final
 iteration','Orignal Solution', 'Target','Earth')
title('Targeting with Fixed Time - Sushrut Prabhu ')
xlabel("Non-dimensional $$\hat{x}$$ (-)","Interpreter", "latex")
ylabel("Non-dimensional $$\hat{y}$$ (-)","Interpreter", "latex")
grid on
grid minor
axis equal

% Part a ii)
r2_des = [-.1 0]';
yn2 = Target2d_tf_phi(r2_des,r,v,t_end,dim_vals{4,2},0,10^-8);
delv2 = yn2(1,4:6)-yo(1,4:6);

plot(yo(:,1),yo(:,2),'k')
plot(r2_des(1),r2_des(2),'*r')
plot(-dim_vals{4,2},0,'ko','MarkerSize',7,'MarkerFaceColor','b')
xlim([-.4 .6])
legend('1^{st} Iteration', '2^{nd} iteration','3^{rd}
 Iteration', '4^{th} iteration', '5^{th} iteration', 'Final
 iteration','Orignal Solution', 'Target','Earth')
title('Targeting with Fixed Time - Sushrut Prabhu ')
xlabel("Non-dimensional $$\hat{x}$$ (-)","Interpreter", "latex")
ylabel("Non-dimensional $$\hat{y}$$ (-)","Interpreter", "latex")
grid on
grid minor
axis equal

% Part a iii)
r3_des = [-.4 -.1]';
yn3 = Target2d_tf_phi(r3_des,r,v,t_end,dim_vals{4,2},0,10^-8);
delv3 = yn3(1,4:6)-yo(1,4:6);

plot(yo(:,1),yo(:,2),'k')
plot(r3_des(1),r3_des(2),'*r')
plot(-dim_vals{4,2},0,'ko','MarkerSize',7,'MarkerFaceColor','b')
xlim([-.5 .6])
legend('1^{st} Iteration', '2^{nd} iteration','3^{rd}
 Iteration', 'Final iteration','Orignal Solution', 'Target','Earth')
title('Targeting with Fixed Time - Sushrut Prabhu ')
xlabel("Non-dimensional $$\hat{x}$$ (-)","Interpreter", "latex")
ylabel("Non-dimensional $$\hat{y}$$ (-)","Interpreter", "latex")
grid on
grid minor
axis equal


t_end =

    0.4775


t_end =
```

```
    0.4775

t_end =

    0.4775
```




Targeting with Fixed Time - Sushrut Prabhu
1st Iteration
2nd iteration
3rd Iteration
4th iteration
5th iteration
Final iteration
Orignal Solution
Target
Earth
Non-dimensional $\hat{y}$ (-)
Non-dimensional $\hat{x}$ (-)


Targeting with Fixed Time - Sushrut Prabhu
1st Iteration
2nd iteration
3rd Iteration
Final iteration
Orignal Solution
Target
Earth
Non-dimensional $\hat{y}$ (-)
Non-dimensional $\hat{x}$ (-)

Part b i)

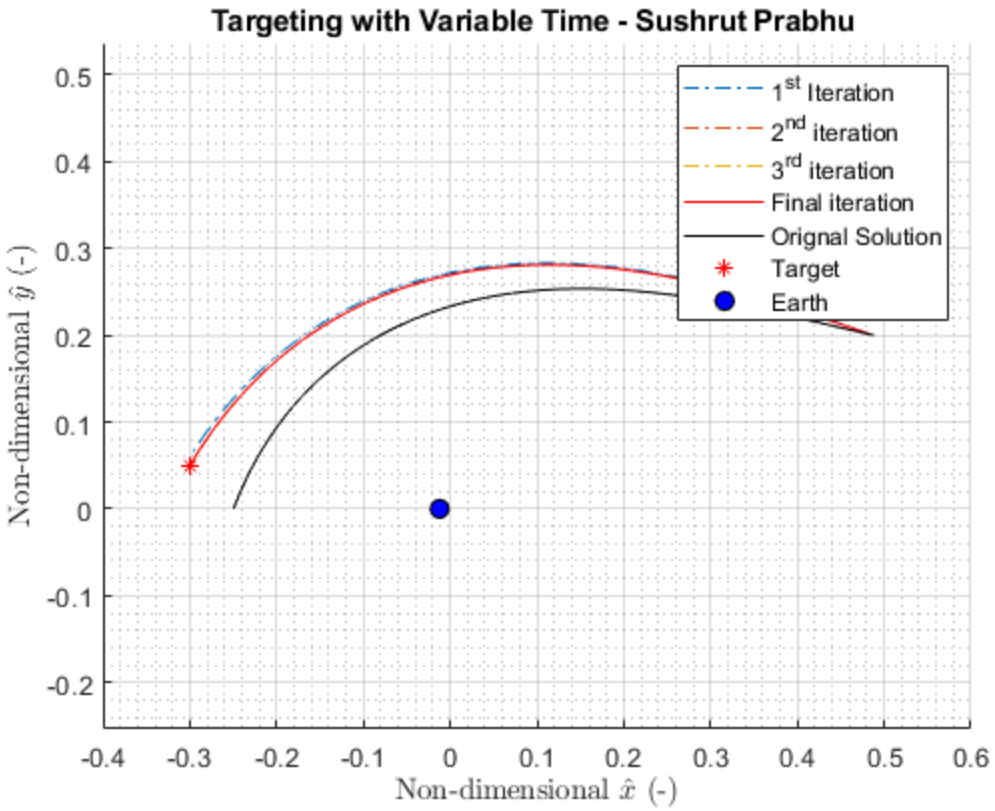
```
r1_des = [-.3 .05]';
yn1 = Target2d_tf_phi(r1_des,r,v,t_end,dim_vals{4,2},1,10^-8);
delv1t = yn1(1,4:6)-yo(1,4:6);

plot(yo(:,1),yo(:,2),'k')
plot(r1_des(1),r1_des(2),'*r')
plot(-dim_vals{4,2},0,'ko','MarkerSize',7,'MarkerFaceColor','b')
xlim([-.4 .6])
legend('1^{st} Iteration', '2^{nd} iteration','3^{rd}
 iteration','Final iteration','Orignal Solution', 'Target','Earth')
title('Targeting with Variable Time - Sushrut Prabhu ')
xlabel("Non-dimensional $$\hat{x}$$ (-)","Interpreter", "latex")
ylabel("Non-dimensional $$\hat{y}$$ (-)","Interpreter", "latex")
grid on
grid minor
axis equal

% Part b ii)
r2_des = [-.1 0]';
yn2 = Target2d_tf_phi(r2_des,r,v,t_end,dim_vals{4,2},1,10^-8);
delv2t = yn2(1,4:6)-yo(1,4:6);

plot(yo(:,1),yo(:,2),'k')
plot(r2_des(1),r2_des(2),'*r')
plot(-dim_vals{4,2},0,'ko','MarkerSize',7,'MarkerFaceColor','b')
xlim([-.4 .6])
legend('1^{st} Iteration', '2^{nd} iteration','3^{rd}
 Iteration', '4^{th} iteration', '5^{th} iteration', 'Final
 iteration','Orignal Solution', 'Target','Earth')
title('Targeting with Variable Time - Sushrut Prabhu ')
xlabel("Non-dimensional $$\hat{x}$$ (-)","Interpreter", "latex")
ylabel("Non-dimensional $$\hat{y}$$ (-)","Interpreter", "latex")
grid on
grid minor
axis equal

% Part b iii)
r3_des = [-.4 -.1]';
yn3 = Target2d_tf_phi(r3_des,r,v,t_end,dim_vals{4,2},1,10^-8);
delv3t = yn3(1,4:6)-yo(1,4:6);

plot(yo(:,1),yo(:,2),'k')
plot(r3_des(1),r3_des(2),'*r')
plot(-dim_vals{4,2},0,'ko','MarkerSize',7,'MarkerFaceColor','b')
xlim([-.5 .6])
legend('1^{st} Iteration', '2^{nd} iteration','3^{rd}
 Iteration', '4^{th} Iteration', 'Final iteration','Orignal
 Solution', 'Target','Earth')
title('Targeting with Variable Time - Sushrut Prabhu ')
xlabel("Non-dimensional $$\hat{x}$$ (-)","Interpreter", "latex")
ylabel("Non-dimensional $$\hat{y}$$ (-)","Interpreter", "latex")
```

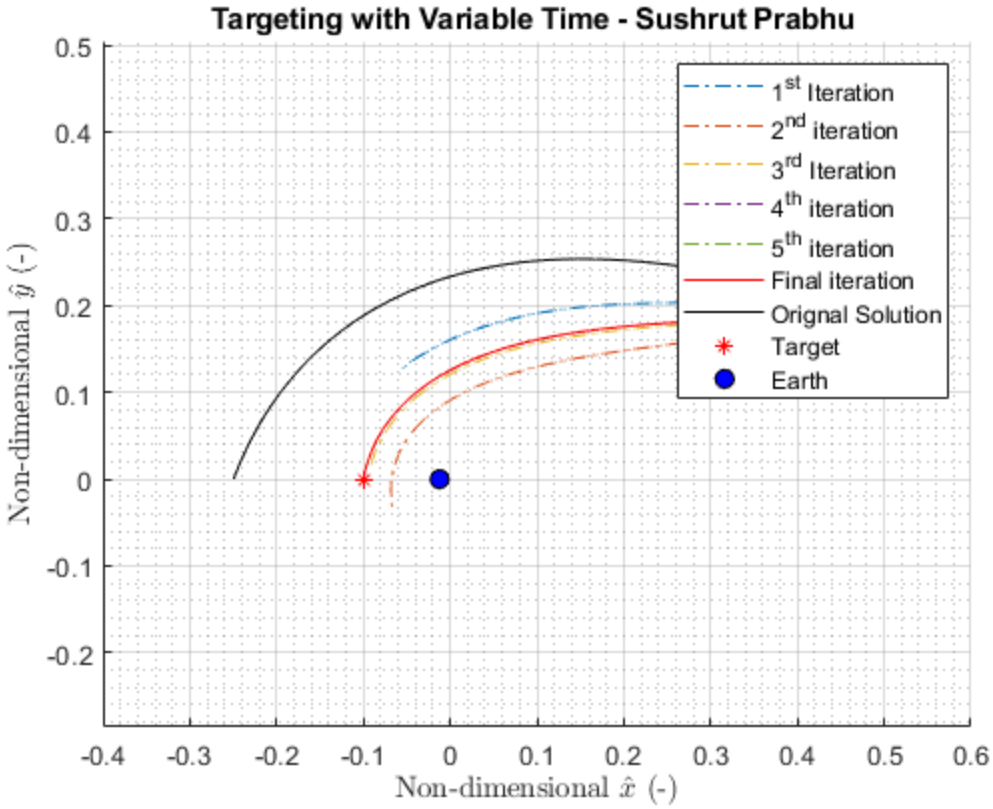
```
grid on
grid minor
axis equal
```

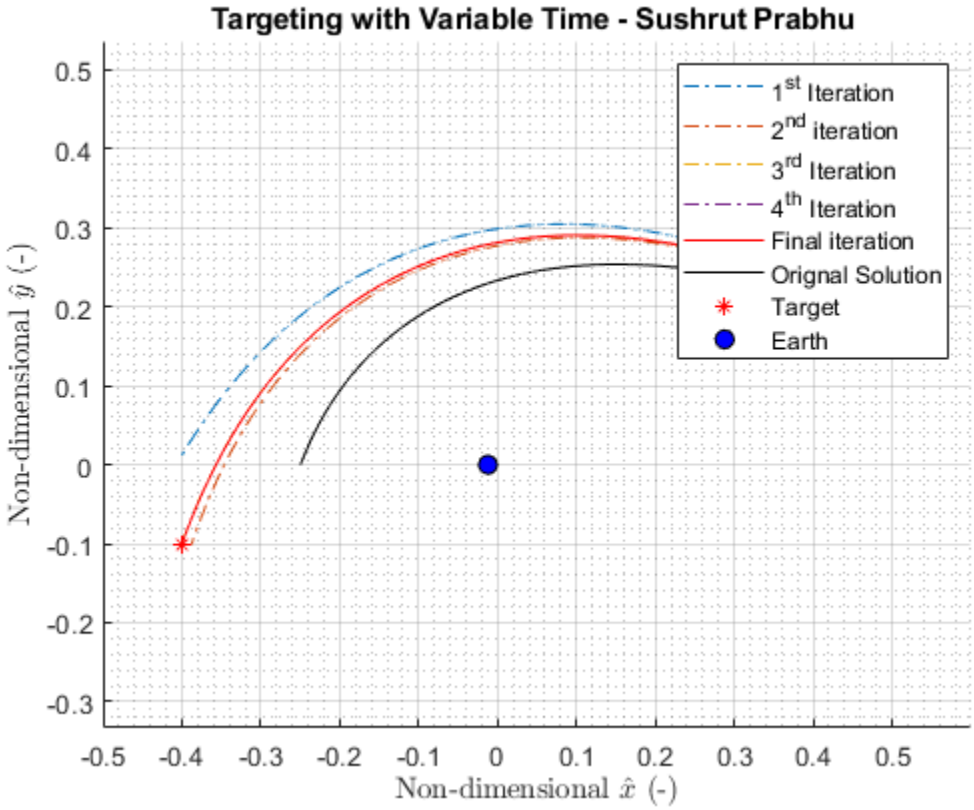
```
t_end =

    0.5268
```

```
t_end =

    0.3357
```

```
t_end =

    0.6605
```



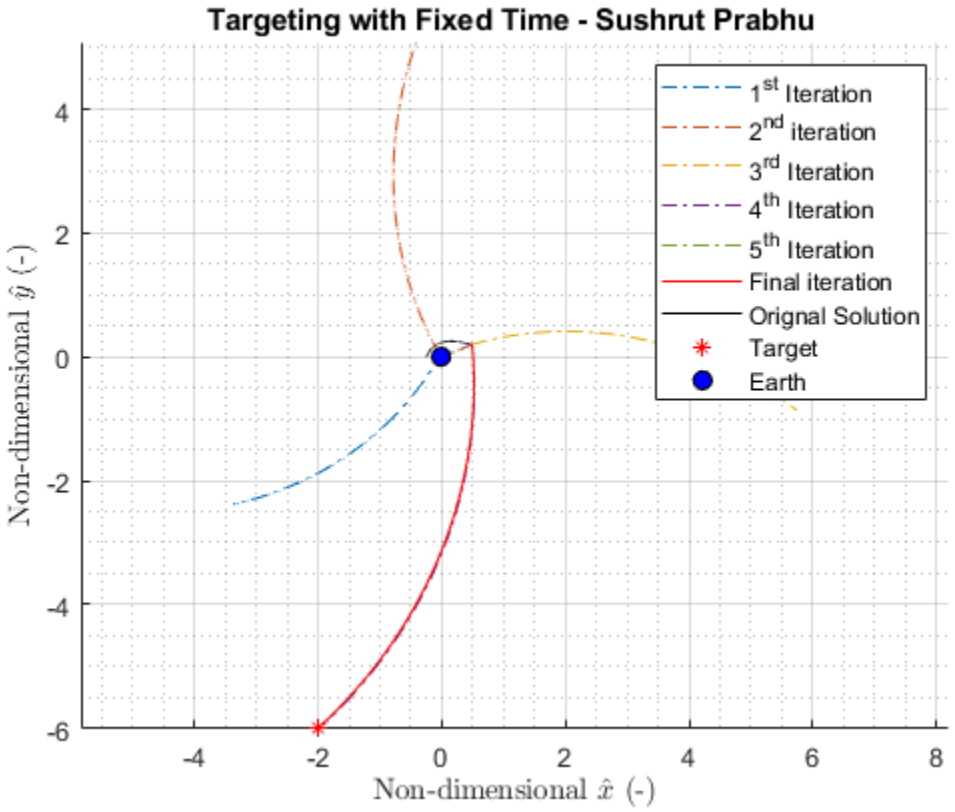

Targeting with Variable Time - Sushrut Prabhu
1st Iteration
2nd iteration
3rd Iteration
4th iteration
5th iteration
Final iteration
Orignal Solution
Target
Earth
Non-dimensional $\hat{y}$ (-)
Non-dimensional $\hat{x}$ (-)


Targeting with Variable Time - Sushrut Prabhu
1st Iteration
2nd iteration
3rd Iteration
4th Iteration
Final iteration
Orignal Solution
Target
Earth
Non-dimensional $\hat{y}$ (-)
Non-dimensional $\hat{x}$ (-)

Part c i)

```
r3_des = [-2 -6]';
yn3 = Target2d_tf_phi(r3_des,r,v,t_end,dim_vals{4,2},0,10^-10);
delv3 = yn3(1,4:6)-yo(1,4:6);

plot(yo(:,1),yo(:,2),'k')
plot(r3_des(1),r3_des(2),'*r')
plot(-dim_vals{4,2},0,'ko','MarkerSize',7,'MarkerFaceColor','b')
legend('1^{st} Iteration', '2^{nd} iteration','3^{rd}
 Iteration','4^{th} Iteration','5^{th} Iteration','Final
 iteration','Orignal Solution', 'Target','Earth')
title('Targeting with Fixed Time - Sushrut Prabhu ')
xlabel("Non-dimensional $$\hat{x}$$ (-)","Interpreter", "latex")
ylabel("Non-dimensional $$\hat{y}$$ (-)","Interpreter", "latex")
grid on
grid minor
axis equal
```

```
t_end =

    0.4775
```



Published with MATLAB® R2018a

PSD3

Given: Perturbation at L_1 of $\xi = 0.01$ and $\eta = 0$ for the Earth-Moon system

Find: a) When it crosses x-axis? Is crossing perpendicular?
b) Periodic orbit, IC? P? Dimensional and non-dimensional
Propagate for 2P, plot
c) Error in $\bar{x}_f - \bar{x}_c$? Error for the elements? Which is most accurate state? Can you improve error?

Solution:

a) We are given $\bar{x}$ so we need to find $\bar{v}$
For $\bar{v}$ we can use linear solution near Libration point to get $\bar{v}_0$

∴ Find U_{xx} and U_{yy} ~ $x = L_1 + \xi$ $y = 0$ $z = 0$

$$U_{xx}^* = 1 - \frac{(1-\mu)}{d^3} - \frac{\mu}{r^3} + \frac{3(1-\mu)(x+\mu)^2}{d^5} + \frac{3\mu(x-1+\mu)^2}{r^5}$$

$$U_{yy}^* = 1 - \frac{(1-\mu)}{d^3} - \frac{\mu}{r^3} + \frac{3(1-\mu)y^2}{d^5} + \frac{3\mu y^2}{r^5}$$

$$d = \left[(x+\mu)^2 + y^2 + z^2\right]^{1/2} \qquad r = \left[(x+\mu-1)^2 + y^2 + z^2\right]^{1/2}$$

$$\beta_1 = 2 - \frac{U_{xx}^* + U_{yy}^*}{2} \qquad \beta_2 = \sqrt{-U_{xx}^* U_{yy}^*}$$

$$s = \left[\beta_1 + (\beta_1^2 + \beta_2^2)^{1/2}\right]^{1/2} \qquad \beta_3 = \frac{s^2 + U_{xx}^*}{2s}$$

$$\therefore \dot{\xi}_0 = 1 \cdot s / \beta_3 \qquad \dot{\eta}_0 = -\beta_2 \xi_0 s = -0.095 \text{ (non-dim)}$$

Now propagate the exact/non-linear equation until it crosses x-axis

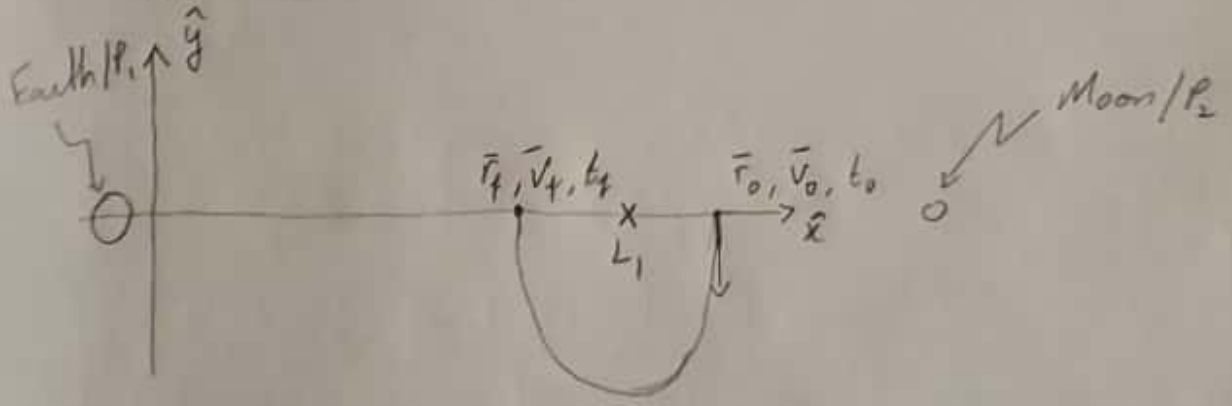
Continued...

Note: at this perturbation the crossing is not perpendicular, you can see it visually but also $\dot{x}_f \neq 0$

The end time is => $t_{end} = 1.1843$ (nondim)
$t_{end} = t^* t_{end} = 5.1426$ days

See Figure: 03.1 Not perpendicular crossing initially

Now let us create a targeting scheme which can get us the solution. We can use the reflection theorem to only propagate ½ an orbit



We know some of the conditions at t_f, we know that in the $\hat{y}$ direction $\bar{r}_f$ needs to 0 and $\bar{v}_f$ in $\hat{x}$ direction need to 0.

$\therefore \bar{r}_f = [x_f, 0, \cancel{z_f}^{0}]$ → planar problem

$\bar{v}_f = [0, \dot{y}_f, \cancel{\dot{z}_f}^{0}]$ → planar

Furthermore, we have fixed $\bar{r}_0$ at a position and $\bar{v}_0$ in $\hat{x}$ direction is 0 to maintain the reflection theorem So only thing we can change are t and $\dot{y}_0$

Continued:

Now we need the relation between the variable initial and final fixed parameter. We pick out the relevent element of the STM

$$\therefore \delta y_f = \frac{\partial y}{\partial x_0}\cancel{\delta x_0}^{0} + \frac{\partial y}{\partial \dot{x}_0}\cancel{\delta \dot{x}_0}^{0} + \frac{\partial y}{\partial y_0}\cancel{\delta y_0}^{0} + \frac{\partial y}{\partial \dot{y}_0}\delta \dot{y}_0 + \dot{y}\delta t$$

$$\therefore \delta y_f = \frac{\partial y}{\partial \dot{y}_0}\delta \dot{y}_0 + \dot{y}\delta t$$

$$\delta \dot{x}_f = \frac{\partial \dot{x}}{\partial x_0}\cancel{\delta x_0}^{0} + \frac{\partial \dot{x}}{\partial \dot{x}_0}\cancel{\delta \dot{x}_0}^{0} + \frac{\partial \dot{x}}{\partial y_0}\cancel{\delta y_0}^{0} + \frac{\partial \dot{x}}{\partial \dot{y}_0}\delta \dot{y}_0 + \ddot{x}\delta t$$

$$\therefore \delta \dot{x}_f = \frac{\partial \dot{x}}{\partial \dot{y}_0}\delta \dot{y}_0 + \ddot{x}\delta t$$

$\overline{rv}_{des}$ ← $\quad$ → Φ_{mod} $\quad$ → $\overline{rv}_{IC}$

$$\therefore \begin{bmatrix} \delta y_f \\ \delta \dot{x}_f \end{bmatrix} = \begin{bmatrix} \frac{\partial y}{\partial \dot{y}_0} & \dot{y} \\ \frac{\partial \dot{x}}{\partial \dot{y}_0} & \ddot{x} \end{bmatrix} \begin{bmatrix} \delta \dot{y}_0 \\ \delta t \end{bmatrix}$$

$$\Phi_{mod} = \begin{bmatrix} \Phi_{2,5} & \dot{y} \\ \Phi_{4,5} & \ddot{x} \end{bmatrix} \rightarrow \text{I am using } 6\times 6 \text{ STM}$$

$\dot{y}$ → obtained from the propogation

$\ddot{x}$ → Use the values of propagation and non-linear equation to get acceleration

$$\therefore \delta y_f = \cancel{y_{f,desired}}^{0} - y_{f,actual} = -y_{f,actual}$$

$$\delta \dot{x}_f = \cancel{\dot{x}_{f,desired}}^{0} - \dot{x}_{f,actual} = -\dot{x}_{f,actual}$$

Continued...

$$\therefore \begin{bmatrix} \delta \dot{y}_0 \\ \delta t \end{bmatrix} = \phi_{modified}^{-1} \begin{bmatrix} -y_{f,actual} \\ -\dot{x}_{f,actual} \end{bmatrix} \leftarrow \text{Targeting Scheme}$$

Use same steps as in PSD2

1) Find ϕ for the trajectory we have
2) Get a modified $\phi \rightarrow \phi_{modified}$
3) Use $\phi_{modified}$ to get $\delta \dot{y}_0$ and δt
4) Use new $\dot{y}_0$ and t_{end} to get new trajectory
5) Find error in trajectory compared to desired state
   ↳ If error is too big re-iterate from (1) with new trajectory

See Figure: D3.1 for targeting iterations

IC (non-dim):
$\bar{r}_0 = 0.8469\,\hat{x} + 0\hat{y} + 0\hat{z}$
$\bar{v}_0 = 0\,\hat{x} - 0.0782\,\hat{y} + 0\hat{z}$

IC:
$r_0 = 3.2555 \times 10^5\,\hat{x} + 0\hat{y} + 0\hat{z}$ km
$v_0 = 0\hat{x} - 0.0802\,\hat{y} + 0\hat{z}$ km/s

$P = 2t_{end} = 2.7092$ (non-dim)
$= P \times t^* = 11.7648$ days

error $(\dot{x}_f) = 2.7103 \times 10^{-9}$ (non-dim)
↳ $= 2.7769 \times 10^{-9}$ (km/s)
error $(y_f) = 1.3174 \times 10^{-9}$ (non-dim)
↳ $= 5.064 \times 10^{-4}$ km

Velocity is more accurate in dimensional quantities

Continued...

The error I was working was with a tolerance of 10^{-8} and was getting reasonable good solution in a reasonable time and steps. But above a tolerance of 10^{-10} it takes a lot of iterations and time to improve by a small amount and above 10^{-14} it doesn't seem to converge. There is a numerical upper limit

See Figure: D3.2 for a plot of 2T.
Note: after you zoom in you can see the effect of inaccurate j creeping in but it is minor
See Figure: D3.3

PSD3

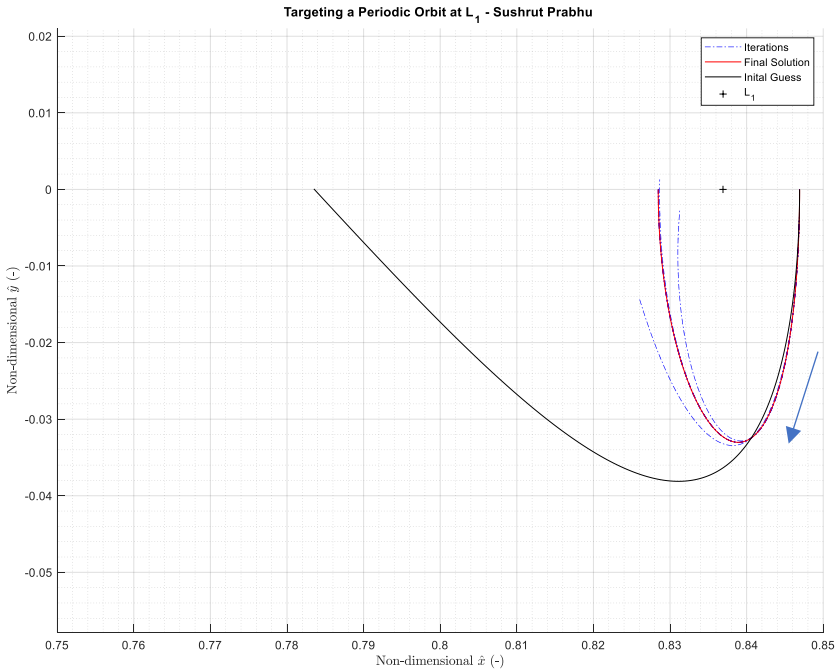


Figure D3.1: Targeting the periodic orbit at L_1.

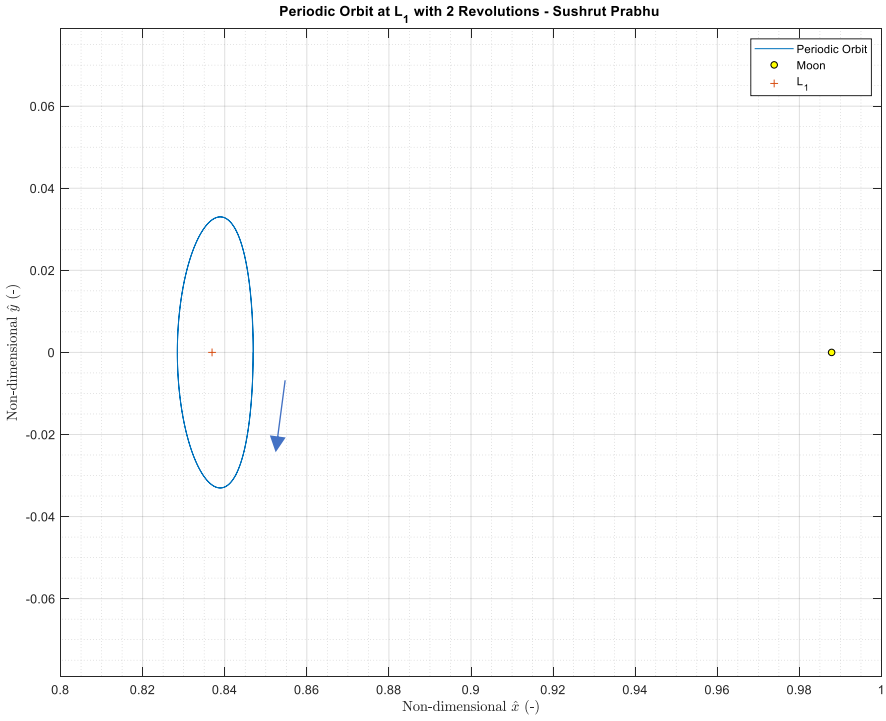


Figure D3.2: Periodic orbit around L_1 simulated for 2 periods.

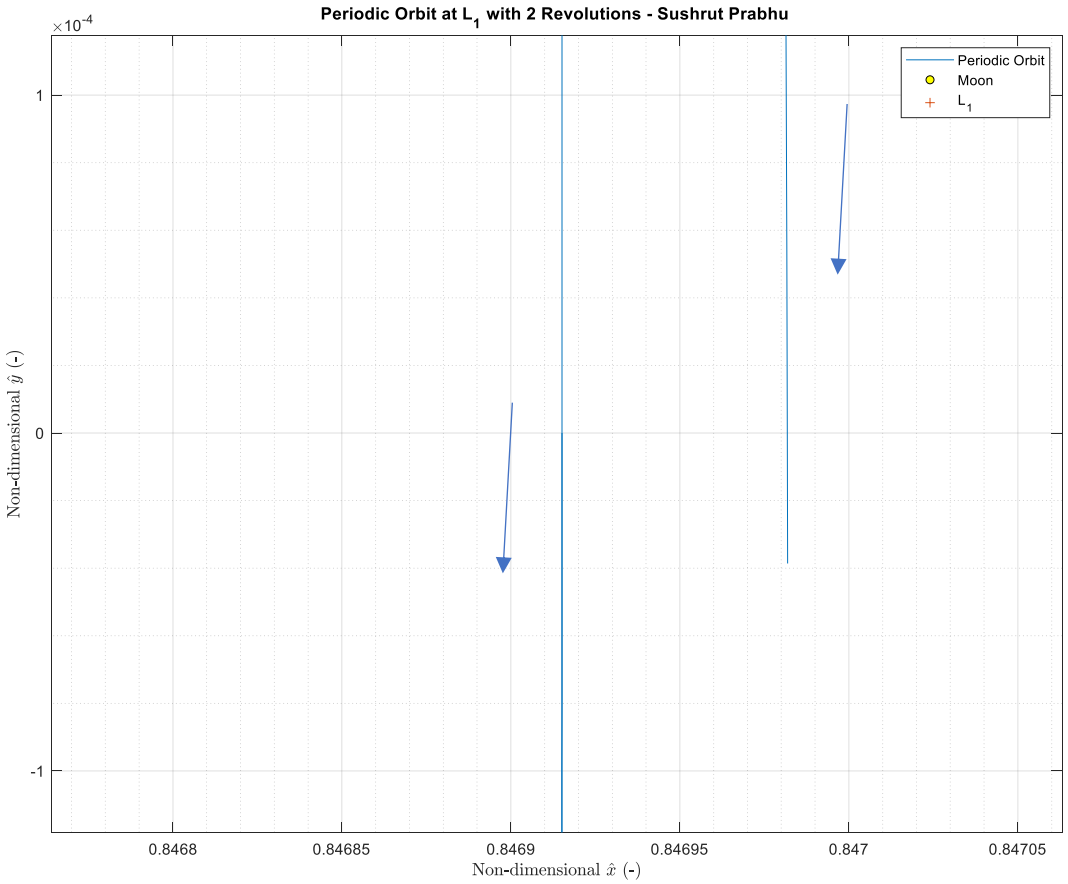
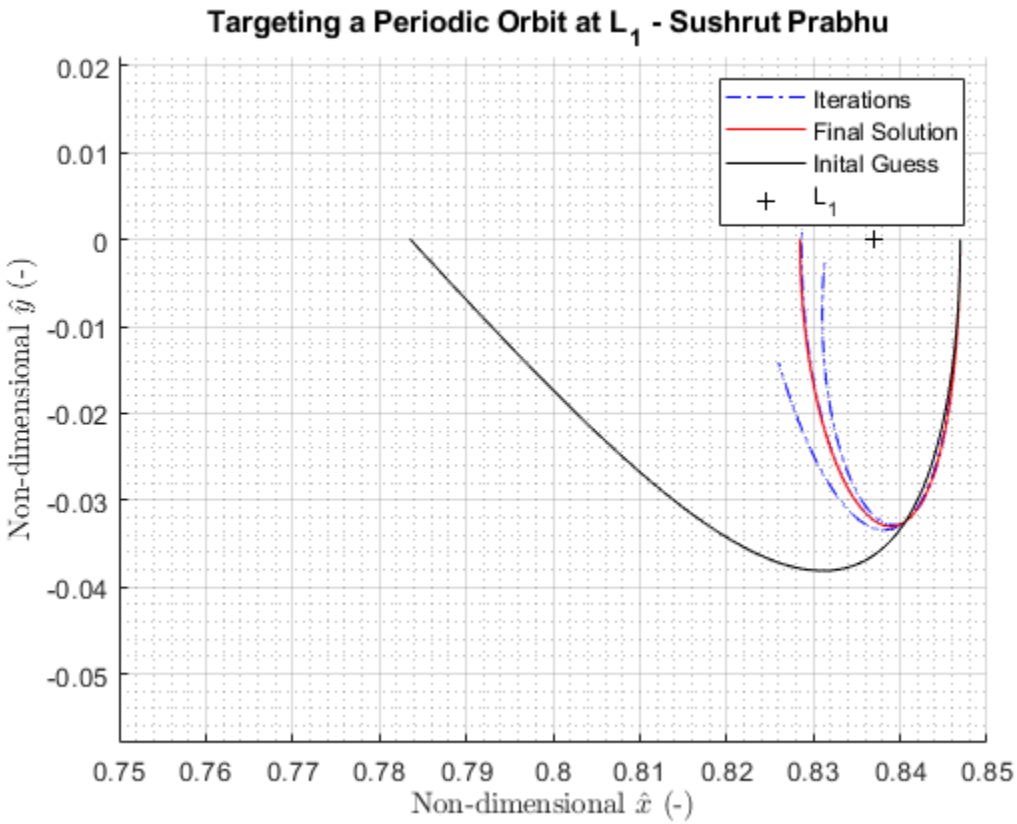


Figure D3.2: Periodic orbit around L_1 simulated for 2 periods zoomed at start.

PSD3

```
clear
close all
clc

SS = SolarS;
systems = {'-', 'Earth-Moon'};
param = {'l* (km)','m* (kg)','miu' , 't*', 'gamma_1', 'L_1', 'gamma_1
 (km)', 'L_1 (km)'};
G = 6.6738*10^-20;
options=odeset('RelTol',1e-12, 'AbsTol',1e-15); % Sets integration
 tollerance

dim_vals = num2cell(zeros(length(param),1));
dim_vals = [systems; param',dim_vals];

% System Constants
[dim_vals{2,2}, dim_vals{3,2}, dim_vals{5,2}] =
 charE(SS.dM_E,0,SS.mMoon/G,SS.mEarth/G);        % Earth Moon
dim_vals{4,2} = SS.mMoon/dim_vals{3,2}/G;   % miu

% Lagrange Point 1
dim_vals{6,2} = abs(L1_NRmethod(dim_vals{4,2}*.7,dim_vals{4,2},
 10^-8));
dim_vals{7,2} = 1-dim_vals{4,2} - dim_vals{6,2};
dim_vals{8,2} = dim_vals{6,2}*dim_vals{2,2};
dim_vals{9,2} = dim_vals{7,2}*dim_vals{2,2};

xi0 = 0.01;
eta0 =0;
r = [dim_vals{7,2}+xi0 eta0 0];

[Uxx,Uyy,~,~,~,~]= Unn(r(1),r(2),r(3),dim_vals{4,2});
beta1 = 2 - (Uxx+Uyy)/2;
beta2 = sqrt(-Uxx*Uyy);
s = sqrt(beta1 + sqrt(beta1^2 + beta2^2));
beta3 = (s^2 + Uxx)/2/s;

xi0_dot = eta0*s/beta3;
eta0_dot = -beta3*xi0*s;
v = [xi0_dot eta0_dot 0];

Per = 2*pi/s;
t_end = Per/2;

IC = [r,v];
[t,y] = ode45(@cr3bp_df,[0 t_end],IC,options,dim_vals{4,2});

t_end = t(find(y(:,2)>0,1));
clear t
[~,y] = ode45(@cr3bp_df,[0 t_end],IC,options,dim_vals{4,2});
```

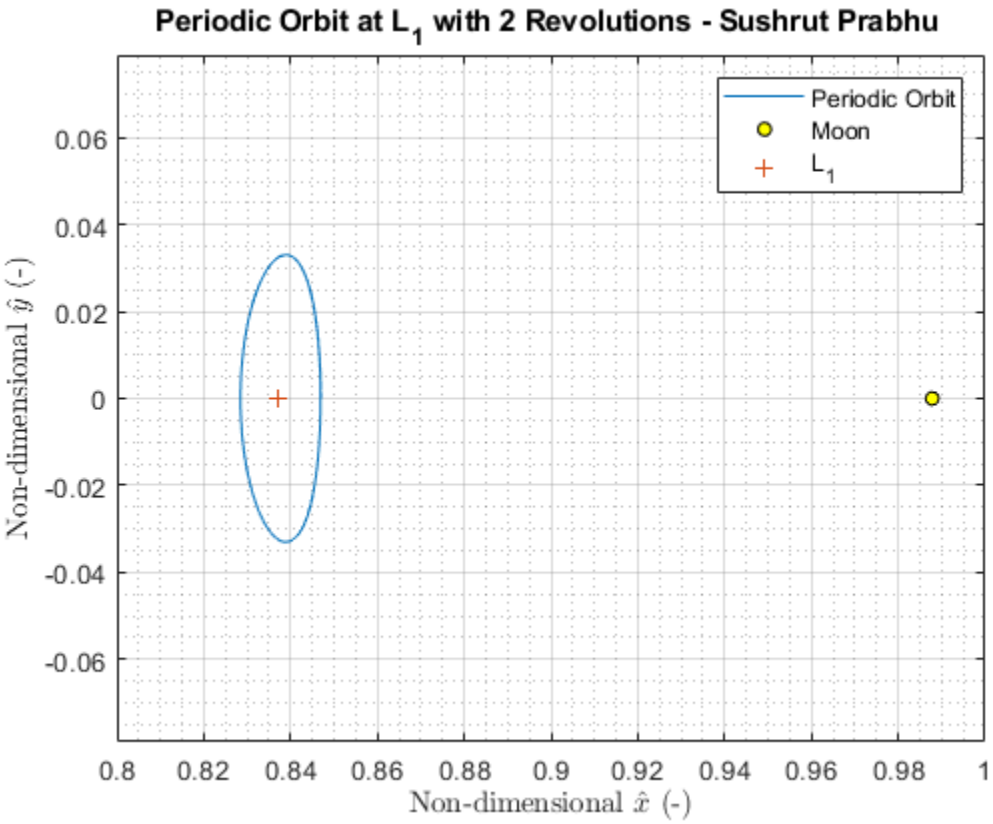
```
rv_des = [0 0]';  % y = 0 and xfdot = 0
[yn,t_end] = Target3d_per(rv_des,r,v,t_end,dim_vals{4,2},"planar",
 10^-8,"plot");

plot(y(:,1),y(:,2),'-k')
plot(dim_vals{7,2},0,'+k')
legend('Iterations','Final Solution','Inital Guess','L_1')
title('Targeting a Periodic Orbit at L_1 - Sushrut Prabhu ')
xlabel("Non-dimensional $$\hat{x}$$ (-)","Interpreter", "latex")
ylabel("Non-dimensional $$\hat{y}$$ (-)","Interpreter", "latex")
xlim([.75 .85])
axis equal
grid on
grid minor
```



Final Orbit

```
IC = yn(1,:);
t_end = t_end*4;
[~,y] = ode45(@cr3bp_df,[0 t_end],IC,options,dim_vals{4,2});

figure
plot(y(:,1),y(:,2))
hold on
plot(1-dim_vals{4,2},0,'ko','MarkerSize',5,'MarkerFaceColor','y')
```

```
plot(dim_vals{7,2},0,'+')
legend("Periodic Orbit", "Moon", "L_1")
title('Periodic Orbit at L_1 with 2 Revolutions - Sushrut Prabhu ')
xlabel("Non-dimensional $$\hat{x}$$ (-)","Interpreter", "latex")
ylabel("Non-dimensional $$\hat{y}$$ (-)","Interpreter", "latex")
axis equal
xlim([.8 1])
grid on
grid minor
```



Published with MATLAB® R2018a

PSD4

Given: The same periodic orbit from PSD3

Find a) Monodromy matrix associated with the orbit
Eigen values and eigenvectors of the matrix, are they real / complex?

b) Find the Lyapunov orbit (5 orbits)

i) Eigenvalues for all of the monodromy matrix. What is x_0 step size? Why? Plot all orbits

ii) Plot i_{y_0} as a function of x_0. Is the curve smooth with continuous derivatives? Could you now predict with more efficiency?

iii) Plot period and JC as a function of x_0.

Solution:

a) Use the same method as before to get the orbit

$\lambda_1 = 8.2309 \times 10^{3}$ $\quad V_1 = [0.9578, -0.0399, 0, 0.2679, 0.0964, 0]^T$

$\lambda_2 = 1.2124 \times 10^{-4}$ $\quad V_2 = [-0.9185, -0.2043, 0, 0.2528, -0.2252, 0]^T$

$\lambda_3 = -0.0005 + 0.0667i$ $\quad \lambda_4 = \bar{\lambda}_3 = -0.0005 - 0.0667i$

$$V_3 = \begin{bmatrix} -0.0113 - 0.1332i \\ 0.8772 + 0i \\ 0 + 0i \\ -0.0208 + 0.0254i \\ -0.2109 - 0.4088i \\ 0 + 0i \end{bmatrix} \qquad V_4 = \begin{bmatrix} -0.0113 + 0.1332i \\ -0.8772 + 0i \\ 0 + 0i \\ -0.0208 - 0.0254i \\ -0.2109 - 0.4088i \\ 0 + 0i \end{bmatrix}$$

$\lambda_5 = 0.9899 + 0.1421i$ $\quad \lambda_6 = \bar{\lambda}_5 = 0.9899 - 0.1421i$

$$V_5 = \begin{bmatrix} 0 + 0i \\ 0 + 0i \\ 0.8944 + 0i \\ 0 + 0i \\ 0 + 0i \\ 0 + 0.4472i \end{bmatrix} \qquad V_6 = \begin{bmatrix} 0 + 0i \\ 0 + 0i \\ 0.8944 + 0i \\ 0 + 0i \\ 0 + 0i \\ 0 - 0.4472i \end{bmatrix}$$

Continued...

There are 2 real and 4 complex root, the complex roots are a pair of complex conjugates.

6 i) To compute new periodic orbit use the previous velocity conditions as initial guess

1) $x_0 = 0.8475$

$\lambda_1 = 8.2029 \times 10^3$ $\lambda_2 = 1.2163 \times 10^{-4}$

$\lambda_3 = -0.001 + 0.0665i$ $\lambda_4 = \bar{\lambda}_3 = -0.001 - 0.0665i$

$\lambda_5 = 0.9905 + 0.1377i$ $\lambda_6 = \bar{\lambda}_5 = 0.9905 - 0.1377i$

2) $x_0 = 0.8481$

$\lambda_1 = 8.1739 \times 10^3$ $\lambda_2 = 1.2202 \times 10^{-4}$

$\lambda_3 = -0.016 + 0.0664i$ $\lambda_4 = \bar{\lambda}_3 = -0.016 + 0.0664i$

$\lambda_5 = 0.9911 + 0.133i$ $\lambda_6 = \bar{\lambda}_5 = 0.9911 - 0.133i$

3) $x_0 = 0.8487$

$\lambda_1 = 8.1438 \times 10^3$ $\lambda_2 = 1.2244 \times 10^{-4}$

$\lambda_3 = -0.022 + 0.0662i$ $\lambda_4 = \bar{\lambda}_3 = -0.022 - 0.0662i$

$\lambda_5 = 0.9918 + 0.1279i$ $\lambda_6 = \bar{\lambda}_5 = 0.9918 - 0.1279i$

4) $x_0 = 0.8493$

$\lambda_1 = 8.1128 \times 10^3$ $\lambda_2 = 1.2287 \times 10^{-4}$

$\lambda_3 = -0.0028 + 0.0661i$ $\lambda_4 = \bar{\lambda}_3 = -0.0028 - 0.0661i$

$\lambda_5 = 0.9925 + 0.1224i$ $\lambda_6 = \bar{\lambda}_5 = 0.9925 - 0.1224i$

Continued...

5) $x_0 = 0.8499$

$\lambda_1 = 8.0808 \times 10^{3}$ $\quad \lambda_2 = 1.2331 \times 10^{-4}$

$\lambda_3 = -0.0034 + 0.0659i$ $\quad \lambda_4 = \bar{\lambda}_3 = -0.0034 - 0.0659i$

$\lambda_5 = 0.9932 + 0.1164i$ $\quad \lambda_6 = \bar{\lambda}_5 = 0.9932 - 0.1164i$

See Figure: 04.1 for plot of Lyapunov orbits

6 i) $x_{0,step} = 0.0006$, this was the largest possible step. 0.0007 would not create a curve from the previous velocity as lunar gravity influence causes more non-linearities. The algorithm does not handle this well.

6 ii) Visually with just 5 points the relationship of $\dot{y}_0$ and x_0 is continuous and linear. We could predict the any orbits between these x_0 bounds but not outside. Note that this system is non-linear, just because a certain section is linear does not mean the next x_0 greater than 0.8499 will be linear.

See figure: 04.2 for plot

iii) We know how to calculate "C", have done it before. See figures: 04.3

For period relationship See figure: 04.4

PSD4

Part b)

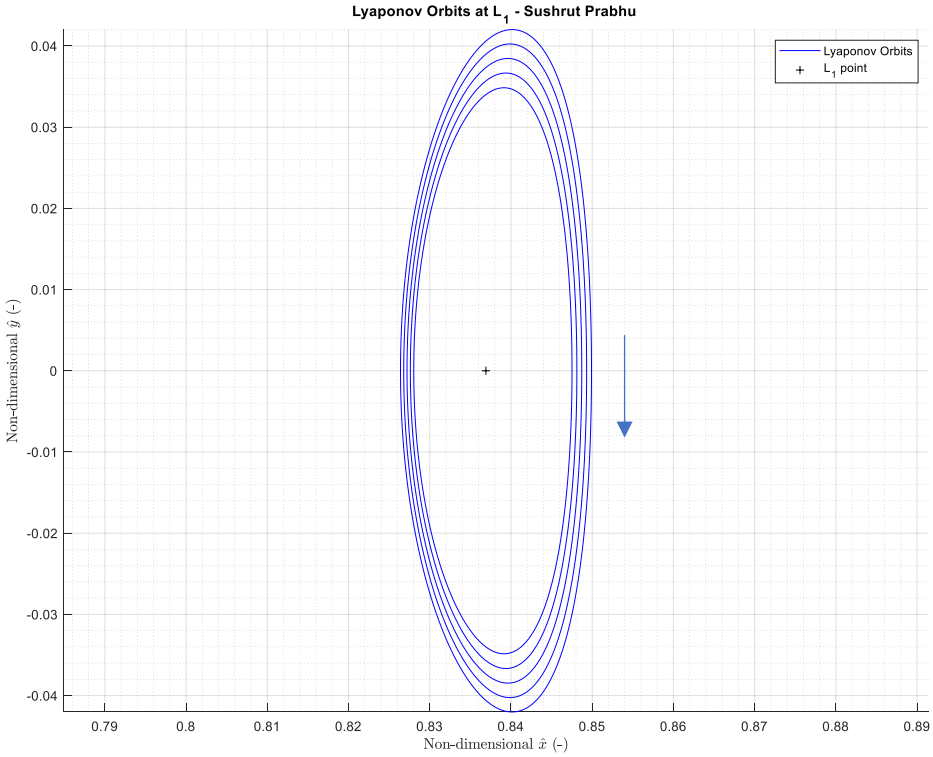


Figure D4.1: Lyapunov orbits around L_1 point.

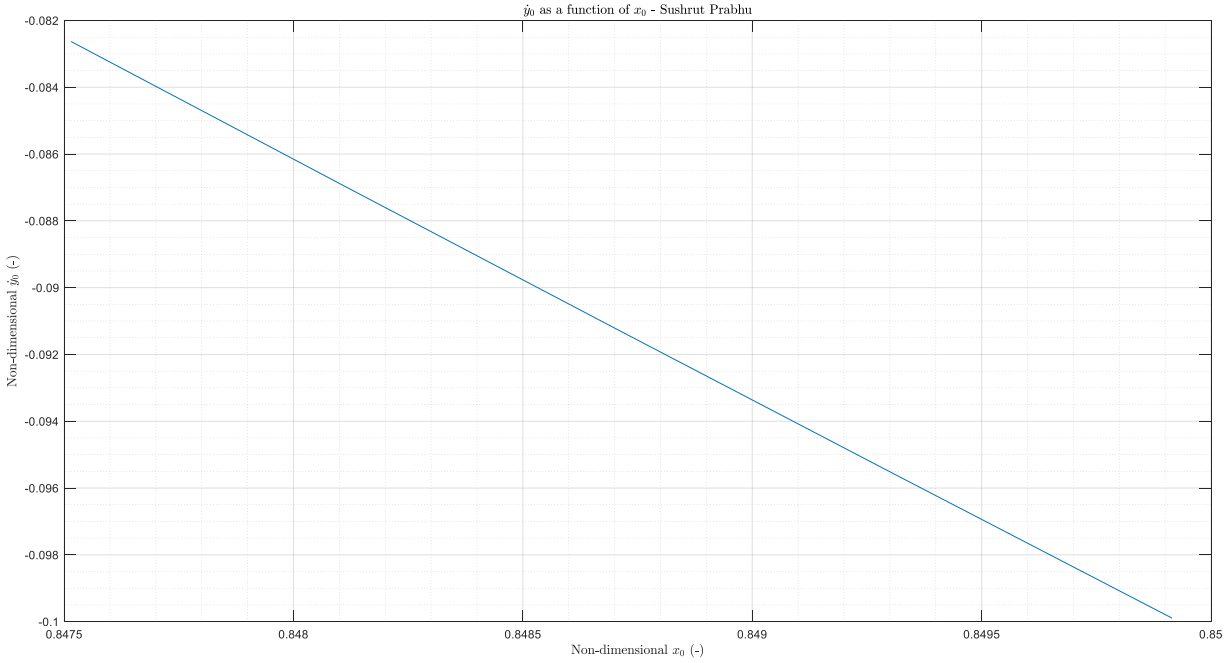


Figure D4.2: Relationship between initial position in x and initial velocity in y.

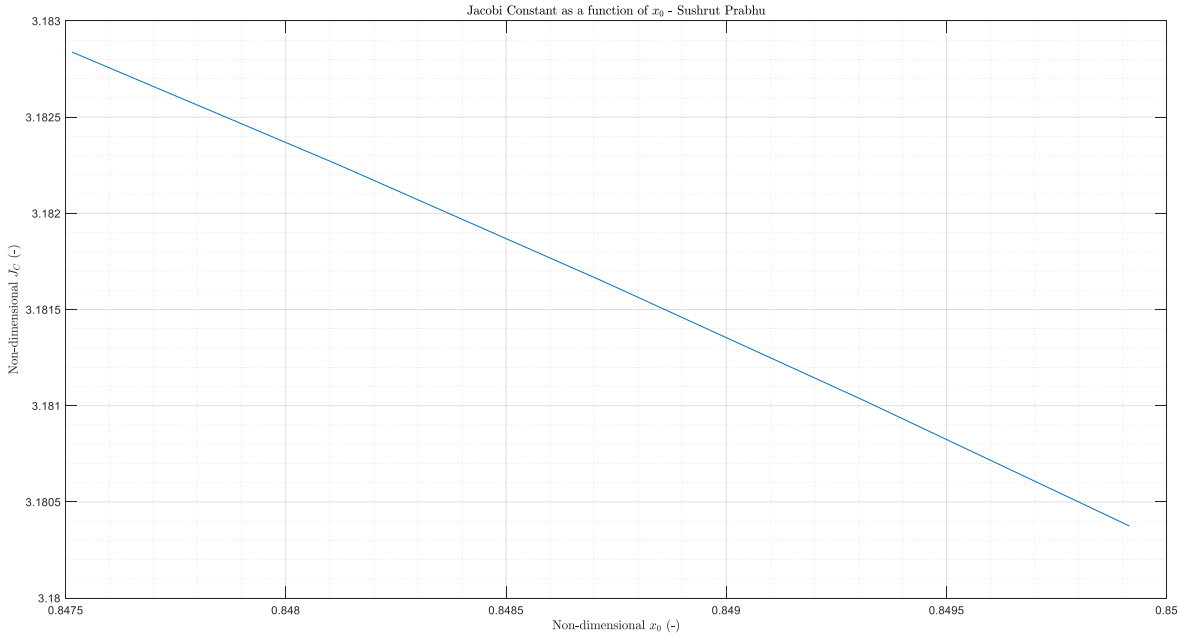


Figure D4.3: Relationship between initial position in x and Jacobi constant.

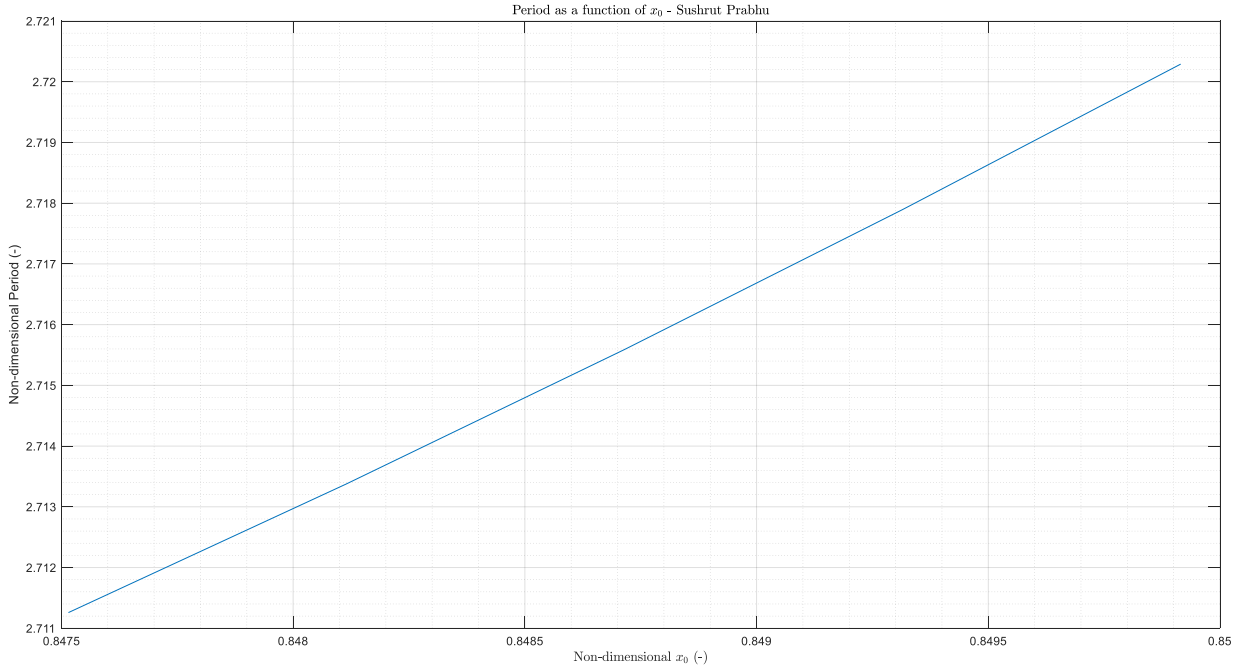


Figure D4.4: Relationship between initial position in x and period of orbit.

Table of Contents

PSD4

```
clear
close all
clc

SS = SolarS;
systems = {'-', 'Earth-Moon'};
param = {'l* (km)','m* (kg)','miu' , 't*', 'gamma_1', 'L_1', 'gamma_1
 (km)', 'L_1 (km)'};
G = 6.6738*10^-20;
options=odeset('RelTol',1e-12, 'AbsTol',1e-15); % Sets integration
 tollerance

dim_vals = num2cell(zeros(length(param),1));
dim_vals = [systems; param',dim_vals];

% System Constants
[dim_vals{2,2}, dim_vals{3,2}, dim_vals{5,2}] =
 charE(SS.dM_E,0,SS.mMoon/G,SS.mEarth/G);        % Earth Moon
dim_vals{4,2} = SS.mMoon/dim_vals{3,2}/G;   % miu

% Lagrange Point 1
dim_vals{6,2} = abs(L1_NRmethod(dim_vals{4,2}*.7,dim_vals{4,2},
 10^-8));
dim_vals{7,2} = 1-dim_vals{4,2} - dim_vals{6,2};
dim_vals{8,2} = dim_vals{6,2}*dim_vals{2,2};
dim_vals{9,2} = dim_vals{7,2}*dim_vals{2,2};

xi0 = 0.01;
eta0 =0;
r = [dim_vals{7,2}+xi0 eta0 0];

[Uxx,Uyy,~,~,~,~]= Unn(r(1),r(2),r(3),dim_vals{4,2});
beta1 = 2 - (Uxx+Uyy)/2;
beta2 = sqrt(-Uxx*Uyy);
s = sqrt(beta1 + sqrt(beta1^2 + beta2^2));
beta3 = (s^2 + Uxx)/2/s;

xi0_dot = eta0*s/beta3;
eta0_dot = -beta3*xi0*s;
v = [xi0_dot eta0_dot 0];

Per = 2*pi/s;
```

```
t_end = Per/2;

IC = [r,v];
[t,y] = ode45(@cr3bp_df,[0 t_end],IC,options,dim_vals{4,2});

t_end = t(find(y(:,2)>0,1));
clear t

rv_des = [0 0]';  % y = 0 and xfdot = 0
[yn,t_end] = Target3d_per(rv_des,r,v,t_end,dim_vals{4,2},"planar",
 10^-8,"");
```

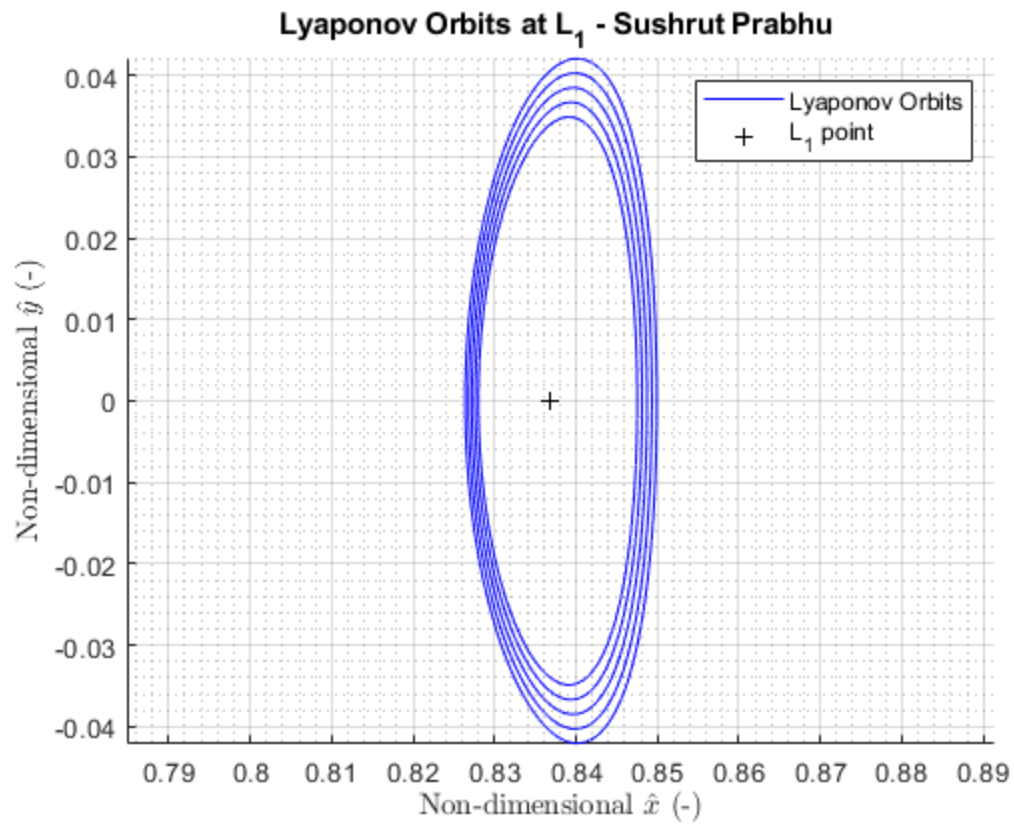
Final Orbit

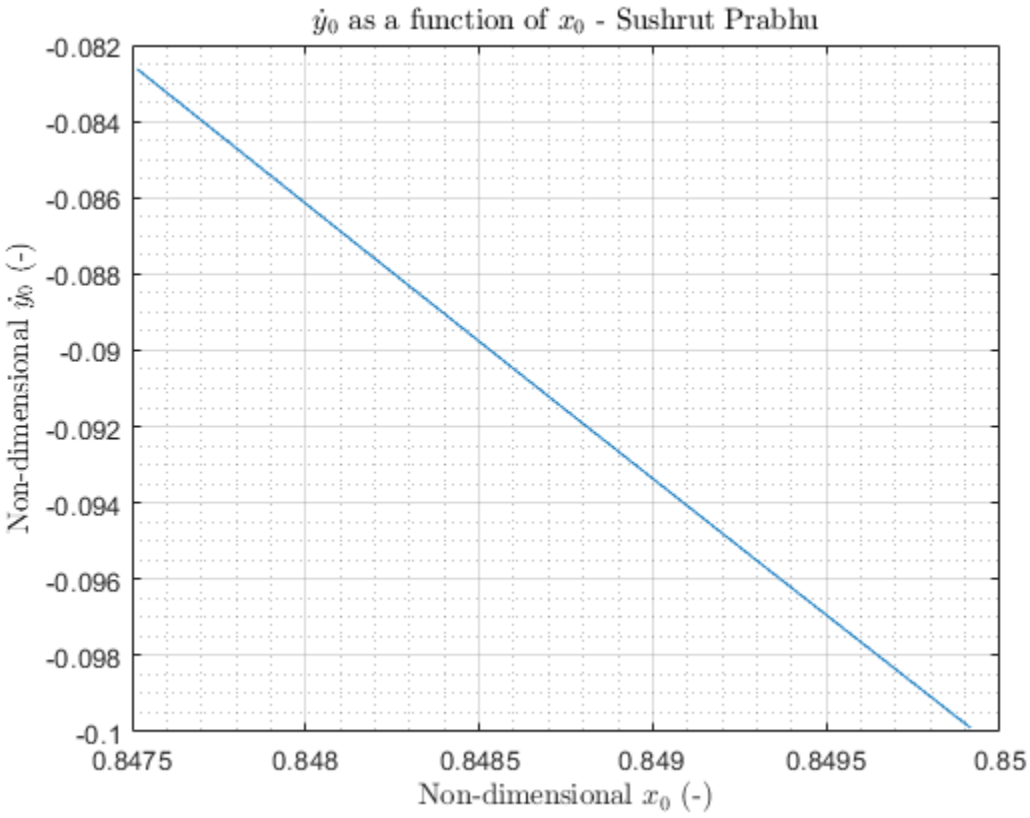
```
IC_stm = eye(6);
IC_stm = IC_stm(:)';
IC = [yn(1,1:6), IC_stm];
t_end = t_end*2;
[~,y] = ode45(@cr3bp_STM_df3d,[0 t_end],IC,options,dim_vals{4,2});

monodromy = reshape(y(end,7:end),6,6);
[V,D] = eig(monodromy);
D = diag(D);
```

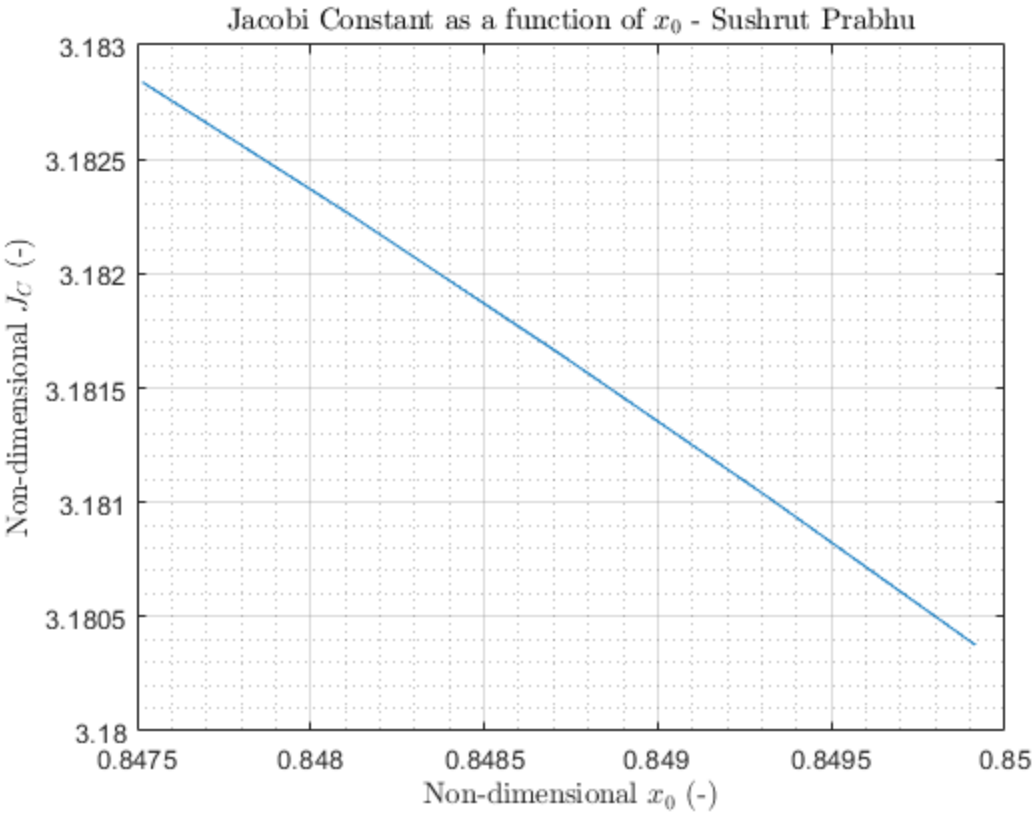
Part b)

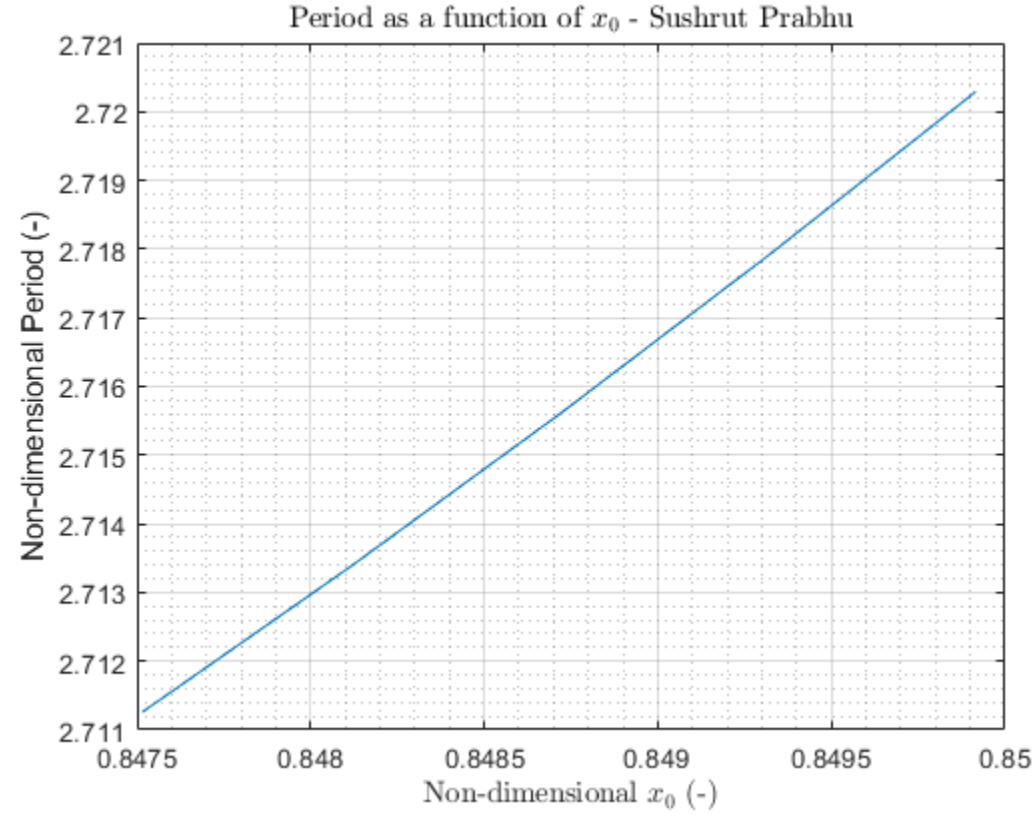
```
x0_step = 0.0006;
k = 1;
t_end = t_end*.8;
figure
hold on

while k<6
    % Inital Guess
    IC = [y(1,1:3),y(1,4:6)];
    IC = IC + [x0_step, 0, 0, 0, 0, 0];
    [t,y] = ode45(@cr3bp_df,[0 t_end],IC,options,dim_vals{4,2});

    t_end = t(find(y(:,2)>0,1));
    clear t

    rv_des = [0 0]';  % y = 0 and xfdot = 0
    [yn,t_end] =
 Target3d_per(rv_des,IC(1:3),IC(4:6),t_end,dim_vals{4,2},"planar",
 10^-8,"");

    % Final plot and solution
    IC_stm = eye(6);
    IC_stm = IC_stm(:)';
    IC = [yn(1,1:6), IC_stm];
    t_end = t_end*2;
    [~,y] = ode45(@cr3bp_STM_df3d,[0 t_end],IC,options,dim_vals{4,2});
```

```
    monodromy = reshape(y(end,7:end),6,6);
    [V,D] = eig(monodromy);
    D = diag(D);

    ly_plt = plot(y(:,1),y(:,2),'-b');
    if k >1

 set(get(get(ly_plt,'Annotation'),'LegendInformation'),'IconDisplayStyle','off');
    end
    x0_vec(k) = y(1,1);
    yd0_vec(k) = y(1,5);

    Per_vec(k) = t_end;
    J_vec(k) = Jacobi_C(y(1,1),y(1,2),0,norm(y(1,4:6)),dim_vals{4,2});

    k = k+1;
end

plot(dim_vals{7,2},0,'+k')
legend('Lyaponov Orbits','L_1 point')
title('Lyaponov Orbits at L_1 - Sushrut Prabhu ')
xlabel("Non-dimensional $$\hat{x}$$ (-)","Interpreter", "latex")
ylabel("Non-dimensional $$\hat{y}$$ (-)","Interpreter", "latex")
axis equal
grid on
grid minor

figure
plot(x0_vec,yd0_vec)
grid on
grid minor
title("$$\dot{y}_0$$ as a function of $$x_0$$ - Sushrut
 Prabhu","Interpreter", "latex")
xlabel("Non-dimensional $$x_0$$ (-)","Interpreter", "latex")
ylabel("Non-dimensional $$\dot{y}_0$$ (-)","Interpreter", "latex")

figure
plot(x0_vec,J_vec)
grid on
grid minor
title("Jacobi Constant as a function of $$x_0$$ - Sushrut
 Prabhu","Interpreter", "latex")
xlabel("Non-dimensional $$x_0$$ (-)","Interpreter", "latex")
ylabel("Non-dimensional $$J_C$$ (-)","Interpreter", "latex")

figure
plot(x0_vec,Per_vec)
grid on
grid minor
title("Period as a function of $$x_0$$ - Sushrut
 Prabhu","Interpreter", "latex")
xlabel("Non-dimensional $$x_0$$ (-)","Interpreter", "latex")
ylabel("Non-dimensional Period (-)")
```


Lyaponov Orbits at L1 - Sushrut Prabhu
Lyaponov Orbits
L1 point
Non-dimensional ŷ (-)
Non-dimensional x̂ (-)
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
0.79
0.8
0.81
0.82
0.83
0.84
0.85
0.86
0.87
0.88
0.89


ẏ0 as a function of x0 - Sushrut Prabhu
Non-dimensional ẏ0 (-)
Non-dimensional x0 (-)
-0.082
-0.084
-0.086
-0.088
-0.09
-0.092
-0.094
-0.096
-0.098
-0.1
0.8475
0.848
0.8485
0.849
0.8495
0.85


Jacobi Constant as a function of x_0 - Sushrut Prabhu
Non-dimensional J_C (-)
3.183
3.1825
3.182
3.1815
3.181
3.1805
3.18
0.8475
0.848
0.8485
0.849
0.8495
0.85
Non-dimensional x_0 (-)


Period as a function of x_0 - Sushrut Prabhu
Non-dimensional Period (-)
2.721
2.72
2.719
2.718
2.717
2.716
2.715
2.714
2.713
2.712
2.711
0.8475
0.848
0.8485
0.849
0.8495
0.85
Non-dimensional x_0 (-)

Published with MATLAB® R2018a

A_t

Intermediate caluclation for STM

```
function A = A_t(x,y,z,miu)

if isnan(z)
    [Uxx,Uyy,~,Uxy,~,~]= Unn(x,y,0,miu);
    A = [zeros(2,2), eye(2,2); Uxx, Uxy, 0, 2; Uxy, Uyy, -2, 0];
else
    [Uxx,Uyy,Uzz,Uxy,Uxz,Uyz]= Unn(x,y,z,miu);
    A = [zeros(3,3), eye(3,3); Uxx, Uxy,Uxz, 0, 2,0; Uxy, Uyy, Uyz, 0,
 -2, 0; Uxz, Uyz, Uzz, 0, 0, 0];
end

end
```

```
Not enough input arguments.

Error in A_t (line 5)
if isnan(z)
```

Published with MATLAB® R2018a

acc_nonlin

Non-linear acceleration for STM

```
function acc = acc_nonlin(r,v,miu)

dd = sqrt((r(1)+miu)^2 + r(2)^2 + r(3)^2);
rr = sqrt((r(1)+miu-1)^2 + r(2)^2 + r(3)^2);

acc(1) = 2*v(2) + r(1) - (1-miu)*(r(1)+miu)/dd^3 - miu*(r(1)-1+miu)/
rr^3;
acc(2) = -2*v(1) + r(2) - (1-miu)*r(2)/dd^3 - miu*r(2)/rr^3;
acc(3) = -(1-miu)*r(3)/dd^3 - miu*r(3)/rr^3;
```

```
Not enough input arguments.

Error in acc_nonlin (line 5)
dd = sqrt((r(1)+miu)^2 + r(2)^2 + r(3)^2);
```

Published with MATLAB® R2018a

cr3bp_STM_df2d

2d STM

```
function dx = cr3bp_STM_df2d(t,x,miu)
dx = zeros(20,1);

d = sqrt((x(1)+miu)^2 + x(2)^2);
r = sqrt((x(1)+miu-1)^2 + x(2)^2);
phi = reshape(x(5:end),4,4)';

dx(1) = x(3);
dx(2) = x(4);
dx(3) = 2*x(4) + x(1) - (1-miu)*(x(1)+miu)/d^3 - miu*(x(1)-1+miu)/r^3;
dx(4) = -2*x(3) + x(2) - (1-miu)*x(2)/d^3 - miu*x(2)/r^3;

phi_dot = [A_t(x(1),x(2),NaN,miu)*phi]';

dx(5:end) = phi_dot(:);
end
```

```
Not enough input arguments.

Error in cr3bp_STM_df2d (line 6)
d = sqrt((x(1)+miu)^2 + x(2)^2);
```

Published with MATLAB® R2018a

cr3bp_STM_df3d

3d STM

```
function dx = cr3bp_STM_df3d(t,x,miu)

dx = zeros(42,1);

d = sqrt((x(1)+miu)^2 + x(2)^2);
r = sqrt((x(1)+miu-1)^2 + x(2)^2);
phi = reshape(x(7:end),6,6)';

d = sqrt((x(1)+miu)^2 + x(2)^2 + x(3)^2);
r = sqrt((x(1)+miu-1)^2 + x(2)^2 + x(3)^2);

dx(1) = x(4);
dx(2) = x(5);
dx(3) = x(6);
dx(4) = 2*x(5) + x(1) - (1-miu)*(x(1)+miu)/d^3 - miu*(x(1)-1+miu)/r^3;
dx(5) = -2*x(4) + x(2) - (1-miu)*x(2)/d^3 - miu*x(2)/r^3;
dx(6) = -(1-miu)*x(3)/d^3 - miu*x(3)/r^3;

phi_dot = [A_t(x(1),x(2),x(3),miu)*phi]';

dx(7:end) = phi_dot(:);
end
```

```
Not enough input arguments.

Error in cr3bp_STM_df3d (line 7)
d = sqrt((x(1)+miu)^2 + x(2)^2);
```

Published with MATLAB® R2018a

Target2d_tf_phi

2D position targeter

```
function  yn = Target2d_tf_phi(r_des,r,v,t_end,miu,dt, tol)
% Initialization
% t = 0:.001:t_end;
options=odeset('RelTol',1e-13, 'AbsTol',1e-15); % Sets integration
 tollerance
error = 2*tol;
phi_0 = eye(4);
phi_0 = phi_0(:)';
i = 1;

figure
hold on

while error > tol
    % Non-linear propgation with phi
    IC = [r(1:2),v(1:2),phi_0];
    [~,y] = ode45(@cr3bp_STM_df2d,[0 t_end],IC,options,miu);

    % Phi at final time
    r_t = y(:,1:2); % Can remove later Simplify
    phi_t = y(:,5:end);
    phi_tf = reshape(phi_t(end,:),4,4)';

    if dt == 0
        phi_main_tf = phi_tf(1:2,3:4);
    elseif dt == 1
        phi_main_tf = [phi_tf(1:2,4),y(end,3:4)'];
    elseif dt == 2
        phi_main_tf = [phi_tf(1:2,3),y(end,3:4)'];
    end

    err = r_des - r_t(end,:)' ;
    delv1 = phi_main_tf^-1 * err;

    if dt == 0
        IC = [r,v+[delv1',0]];
    elseif dt == 1
        IC = [r,v+[0, delv1(1), 0]];
        t_end = t_end+delv1(2);
    elseif dt == 2
        IC = [r,v+[delv1(1) 0, 0]];
        t_end = t_end+delv1(2);
    end

    [~,yn] = ode45(@cr3bp_df,[0 t_end],IC,options,miu);

    r = yn(1,1:3);
    v = yn(1,4:6);
```

```
    error = max(abs(yn(end,1:2)-r_des'));
    i = i+1;

    if i > 20
        error = tol/2;
        fprintf("Did not Coverge")
    end

    if error > tol
        plot(yn(:,1),yn(:,2),'-.');
    else
       plot(yn(:,1),yn(:,2),'r');
    end
end


end
```

```
Not enough input arguments.

Error in Target2d_tf_phi (line 7)
error = 2*tol;
```

Published with MATLAB® R2018a

Target3d_per

Periodic orbit targeter

```
function  [IC_final,t_end] = Target3d_per(r_des,r,v,t_end,miu,fix,
 tol, pl)
% Initialization
% t = 0:.001:t_end;
options=odeset('RelTol',1e-13, 'AbsTol',1e-15); % Sets integration
 tollerance
error = 2*tol;
phi_0 = eye(6);
phi_0 = phi_0(:)';
i = 1;

if pl == "plot"
    figure
    hold on
end

while error > tol
    % Non-linear propgation with phi
    IC = [r,v,phi_0];
    [~,y] = ode45(@cr3bp_STM_df3d,[0 t_end],IC,options,miu);

    % Phi at final time
    phi_t = y(:,7:end);
    phi_tf = reshape(phi_t(end,:),6,6)';
    acc = acc_nonlin(y(end,1:3),y(end,4:6),miu);

    if fix == "planar"
        phi_main_tf = [phi_tf(2,5), y(end,5); phi_tf(4,5), acc(1)];
        err = r_des - [y(end,2), y(end,4)]';
    elseif fix == "x0"
        phi_main_tf = [phi_tf(4,3), phi_tf(4,5); phi_tf(6,3),
 phi_tf(6,5)] + 1/y(end,5)*[acc(1), acc(3)]'*[phi_tf(2,3),
 phi_tf(2,5)];
        err = r_des - [y(end,4), y(end,6)]';
    elseif fix == 2
        phi_main_tf = [phi_tf(1:2,4),y(end,3:4)'];
    elseif dt == 2
        phi_main_tf = [phi_tf(1:2,3),y(end,3:4)'];
    end

    delv1 = phi_main_tf^-1 * err;

    if fix == "planar"
        IC = [r,v+[0, delv1(1), 0]];
        t_end = t_end + delv1(2);
    elseif fix == "x0"
        IC = [r+[0,0,delv1(1)],v+[0,delv1(2),0]];
    elseif fix == 1
```

```
        IC = [r,v+[0, delv1(1), 0]];
        t_end = t_end+delv1(2);
    elseif fix == 2
        IC = [r,v+[delv1(1) 0, 0]];
        t_end = t_end+delv1(2);
    end

    [~,yn] = ode45(@cr3bp_df,[0 t_end],IC,options,miu);

    r = yn(1,1:3);
    v = yn(1,4:6);

    if fix == "planar"
        error = max(abs([yn(end,2),yn(end,4)]-r_des'));
    elseif fix == "x0"
        error = max(abs([yn(end,4),yn(end,6)]-r_des'));
    end

    i = i+1;

    if i > 500
        error = tol/2;
        fprintf("Did not Coverge")
    end
    if pl == "plot"
        if error > tol
            ite = plot(yn(:,1),yn(:,2),'-.b');
            if i > 2

 set(get(get(ite,'Annotation'),'LegendInformation'),'IconDisplayStyle','off');
            end
        else
           plot(yn(:,1),yn(:,2),'r');
        end
    end
end

IC_final = yn(1,:);
end
```

```
Not enough input arguments.

Error in Target3d_per (line 7)
error = 2*tol;
```

Published with MATLAB® R2018a

Characteristic Elements

```
function [lstar, mstar, tstar] = charE(D1,D2,m1,m2)
G = 6.6738*10^-20;

lstar = D1+D2;
mstar = m1 + m2;

tstar = sqrt(lstar^3/G/mstar);
```

```
Not enough input arguments.

Error in charE (line 5)
lstar = D1+D2;
```

Published with MATLAB® R2018a

horner_alg

```
function [alpha,beta] = horner_alg(n,a,z0)
alpha = a(1);
beta = 0;

for k = 2:n
   beta = alpha + z0*beta;
   alpha = a(k) + z0*alpha;
end


end
```

```
Not enough input arguments.

Error in horner_alg (line 3)
alpha = a(1);
```

Published with MATLAB® R2018a

Gamma1 Newton Rhapson

```
function g1_n1 = L1_NRmethod(g1_n, miu, acc)

err = acc*2;
pz = [-1,(3-miu),(2*miu-3),miu,-2*miu,miu];
while (err > acc)
    [fn,fn_p] = horner_alg(6,pz,g1_n);

    g1_n1 = g1_n - fn/fn_p;

    err = abs(g1_n1-g1_n)/abs(g1_n);
    g1_n = g1_n1;
end

end
```

```
Not enough input arguments.

Error in L1_NRmethod (line 4)
err = acc*2;
```

Published with MATLAB® R2018a

Unn

```
function [Uxx,Uyy,Uzz,Uxy,Uxz,Uyz]= Unn(x,y,z,miu)

xm = (x+miu);
xm1 = x+miu-1;

d = sqrt(xm^2 + y^2 +z^2);
r = sqrt(xm1^2 + y^2 +z^2);

Uxx = 1-(1-miu)/d^3 - miu/r^3 + 3*(1-miu)*xm^2 / d^5 + 3*miu*xm1^2 /
 r^5;
Uyy =  1-(1-miu)/d^3 - miu/r^3 + 3*(1-miu)*y^2 / d^5 + 3*miu*y^2 /
 r^5;
Uzz = -(1-miu)/d^3 - miu/r^3 + 3*(1-miu)*z^2 / d^5 + 3*miu*z^2 / r^5;

Uxy = 3*(1-miu)*xm*y/d^5 + 3*miu*xm1*y/r^5;
Uxz = 3*(1-miu)*xm*z/d^5 + 3*miu*xm1*z/r^5;
Uyz = 3*miu*y*z/d^5 + 3*miu*y*z/r^5;
end
```

```
Not enough input arguments.

Error in Unn (line 4)
xm = (x+miu);
```

Published with MATLAB® R2018a

Solar Systems Constants

Constants

```
classdef SolarS
    properties
        % Distances from Sun/ planet to planet/ Moon
        dMercury = 57909226.542
        dVenus = 108209474.537
        dEarth = 149597870.7
        dMars = 227943816.693
        dJupiter = 778340816.693
        dSaturn = 1426666414.180
        dM_E = 384400
        dMoon = 384400+149597870.7
        dPluto = 5906440596.5288
        dTitan_S = 1221865
        dPhobos_M = 9376;
        dEuropa_J = 671100
        dOberon_U = 583500
        dTriton_N = 354759
        dCharon_P = 17536

        % Mass of Planet/Star/ Moon (G*m)
        mSun = 132712440017.99
        mVenus = 324858.5988
        mEarth = 398600.4415
        mMars = 42828.3142
        mJupiter = 126712767.8578
        mSaturn = 37940626.0611
        mUranus = 5794549.0070719
        mNeptune = 6836534.06387
        mPluto = 981.600887707;

        mMoon = 4902.8011
        mTitan = 8979.766
        mPhobos = 7.11328968e-04
        mEuropa = 3203.31978
        mOberon = 192.4249
        mTriton = 1427.8589
        mCharon = 103.2187

        % Radius of Moon/Planet
        rEarth = 6378.136;
        rMars = 3397.00;
        rMoon = 1738.10;
        rSaturn = 60268.00;
        rJupiter = 71492.00;
        % Eccentricity of Planets
        eEarth = 0.01671022
        eSaturn = 0.05386179
```

```
    end
end
```

```
ans =

  SolarS with properties:

     dMercury: 5.7909e+07
       dVenus: 1.0821e+08
       dEarth: 1.4960e+08
        dMars: 2.2794e+08
     dJupiter: 7.7834e+08
      dSaturn: 1.4267e+09
         dM_E: 384400
        dMoon: 1.4998e+08
       dPluto: 5.9064e+09
     dTitan_S: 1221865
    dPhobos_M: 9376
    dEuropa_J: 671100
    dOberon_U: 583500
    dTriton_N: 354759
    dCharon_P: 17536
         mSun: 1.3271e+11
       mVenus: 3.2486e+05
       mEarth: 3.9860e+05
        mMars: 4.2828e+04
     mJupiter: 1.2671e+08
      mSaturn: 3.7941e+07
      mUranus: 5.7945e+06
     mNeptune: 6.8365e+06
       mPluto: 981.6009
        mMoon: 4.9028e+03
       mTitan: 8.9798e+03
      mPhobos: 7.1133e-04
      mEuropa: 3.2033e+03
      mOberon: 192.4249
      mTriton: 1.4279e+03
      mCharon: 103.2187
       rEarth: 6.3781e+03
        rMars: 3397
        rMoon: 1.7381e+03
      rSaturn: 60268
     rJupiter: 71492
       eEarth: 0.0167
      eSaturn: 0.0539
```

Published with MATLAB® R2018a