HW 3

Trajectory Optimization I: Brute Force Optimization (Pork Chop Plots), Acceleration Function, Optimizing Performance Index

Justin Self

AERO557: Advanced Orbital Mechanics

February 29, 2024

Problem 1

Pork Chop Plot

Generate a pork chop plot for the 2005 Earth to Mars opportunity. Use Earth departure dates from JD = 2453528.0 (June 6th , 2005) to 2453682.0 (Nov. 7 th , 2005) and Mars arrival dates from JD = 2453706.0 (Dec. 1 st , 2005) to 2454156.0 (Feb. 24th , 2007). Plot C3 at Earth (km2/s2), V_∞ at Mars (km/s), and time of transfer (days). Use the plot to answer the following questions. Please turn in the plot with the homework.

Solution.

See Appendix A for results and code for Problem 1

Part A: Robotic Mission to Mars

`What do you think is the optimal departure/arrival combination for a robotic mission that wishes to minimize total delta-v without using any aerobraking? (Just eyeball it - a range of answers is certainly acceptable.)`
`Is this a Type I or a Type II trajectory?`

Using pork chop plots as a brute-force optimization tool is a quick way to determine the best ballpark estimate for departure and arrival dates given known set of basic parameters. **In this case, the lowest V_∞ occurs at two departure/arrival combinations, shown in Fig. 1 and listed in Table 1.** The best departure / arrival combination times for a robotic mission that wishes to minimize total Δv are found in Table 1. If a sooner mission is desired, the Type II sets should be utilized, thus leaving the Type I (right side of Fig. 1) available for backup mission times.

Table 1: Brute Force Optimization for Robotic Mission

Timeline	Type I	Type II
Departure Dates	Aug. 23, 2005 - Sept. 22, 2005	June 8, 2005 - July 4, 2005
Arrival Dates	Mar 31, 2006 - May 20, 2006	Mar 11, 2006 - April 30, 2006
Days past epoch (Departure)	80-110	4-30
Days past epoch (Arrival)	120-170	100-150

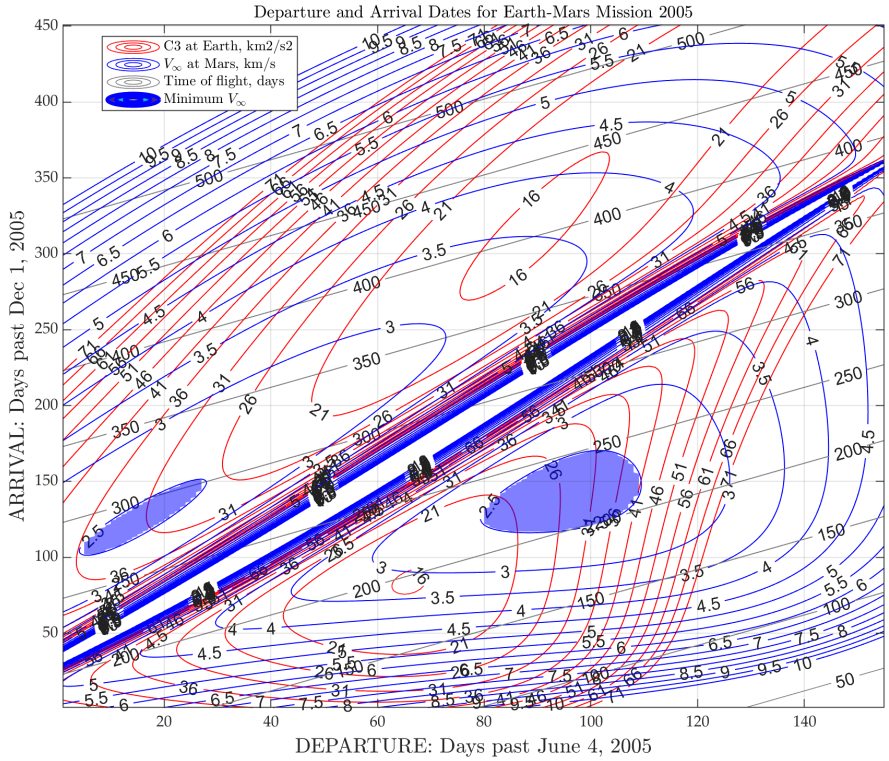


Figure 1: Porkchop plot showing departure and arrival times based on earth-mars trajectory in 2005. Maximum V_∞ shown is 10 km/s. Optimal (lowest V_∞) occurs at two different departure/arrival groups based on *short way* (Type I, right) or *long way* (Type II, left), shown highlighted.

Part B: Human Mission to Mars

`What is the optimal departure/arrival combination for a human mission to Mars, given the maximum flight duration of 150 days? Assume that the mission will implement a direct transfer descent to the Martian surface, i.e., it will not enter into a Mars orbit. Also assume that the entry vehicle can only withstand the heat generated from a trajectory with a V_inf < 4.5 km/s.`

Similar to the robotic mission, the pork chop plot shown in Fig. 2 was developed using the same mission parameters. In this case, however, constraints due to the human crew were applied: namely, maximum flight time is 150 or less and the arrival V_∞ must be ≤ 4.5 km/s. Only values that fit within the prescribed constraints are highlighted in Fig. 2. The only window of opportunity that falls within all imposed constraints is given in Table 2. Lowest flight time is about 130 days, and lowest V_∞ is about 3.5 km/s and is achievable near the 150-day flight time limitation. Lowest C3 value for this flight is about 30 km2/s2, visible in Fig. 3, a zoomed-in view of the feasible flights shown in Fig. 2.

Table 2: Brute Force Optimization for Human Mission

Timeline	Type I	Type II
Departure Dates	Sept. 12, 2005 - Oct. 20, 2005	N/A
Arrival Dates	Jan 30, 2006 - Mar 21, 2006	N/A
Days past epoch (Departure)	100-138	N/A
Days past epoch (Arrival)	60-110	N/A

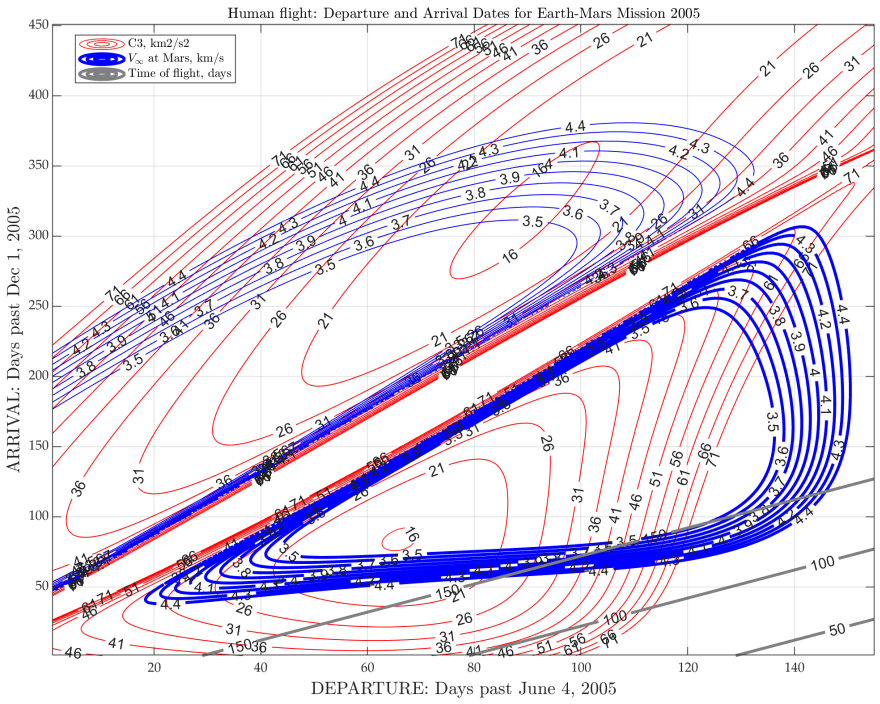


Figure 2: Same earth-mars trajectory plot as in Fig. 1, but with crewed flight mission constraints: maximum flight time < 150 days and V_∞ at arrival must be < 4.5 km/s. The optimal departure date window is roughly Sept. 2, 2005 - Oct. 22, 2005, and arrival window is roughly Feb. 4, 2006 - Mar. 16, 2006 with a minimum flight time of about 140 days.

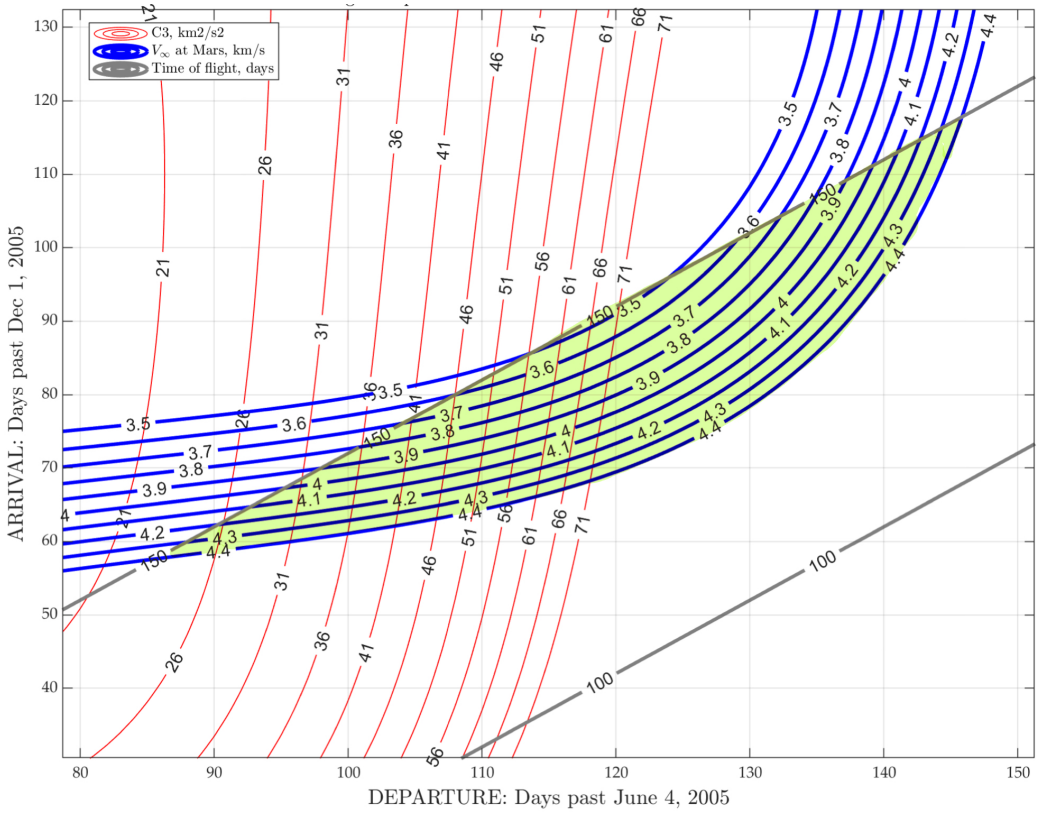


Figure 3: Inset showing viable transfer departure/arrival windows for human Mars mission with given constraints; feasibility region highlighted.

Problem 2

Acceleration Function

Determine the acceleration function a(t) that will minimize the performance index/cost function:

$$J = \dot{x_f}^2 + \frac{1}{2}\int_{t_0}^{t_f} a^2 dt$$

subject to the following boundary conditions: $t_0 = 0$, $x(t_0) = 0$, $\dot{x}(t_0) = 0$ and $t_f = 1$, $x(t_f) = 1$, and $\dot{x}(t_f) =$ free.

Give the optimal acceleration function, functions for position and velocity, and evaluate the cost/performance index.

Solution.

The final outputs derived from the work by hand are:

Optimal acceleration:

$$a = -2(3t - 2)$$

Position function:

$$x(t) = t(2t - t^2)$$

Velocity function:

$$\dot{x}(t) = t(4 - 3t)$$

Cost index evaluated at $t = [0, 1]$

$$J = 1 + 2\int_{t_0}^{t_f} (3t - 2)^2 \text{ dt}$$

$$= 3$$

See following pages for work by hand.

Problem 2: Acceleration Function

Determine the acceleration function a(t) that will minimize the performance index/cost function:

$$J = \dot{x}_f^2 + \frac{1}{2}\int_{t0}^{tf} a^2 dt$$

subject to the following boundary conditions:

t0 = 0, x(t0)=0, xdot(t0)=0 and tf =1, x(tf)=1, and xdot(tf) = free

Give the optimal acceleration function, functions for position and velocity, and evaluate the cost/performance index.

BOUNDARY CONDITIONS:

$t_0 = 0$ $\quad$ $t_f = 1$

$x(0) = 0$ $\quad$ $x(1) = 1$ $\quad \Rightarrow t_{span} = [0, 1]$

$\dot{x}(0) = 0$ $\quad$ $\dot{x}(1) =$ FREE

FIND

- OPTIMAL ACCELERATION FUNCTION, $a(t)$
- $x(t)$, $y(t)$
- FIND COST / PERF. INDEX.

MINIMIZE J!

i) FIND COST FUNCTION / PERFORMANCE INDEX.

PERFORMANCE INDEX: $\dot{x}_f^2$ ≡ MINIMIZE (SINCE ⊕) FINAL VELOCITY (x-DIRECTION) SQUARED.

COST FUNCTION: $\frac{1}{2}\int_{t_0}^{t_f} a^2 dt$

ii.) CREATE FUNCTION f

$$\bar{S} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

$$\bar{c} = [c_1] = a$$

$$\bar{f} = \dot{\bar{S}} = \begin{bmatrix} \dot{s}_1 \\ \dot{s}_2 \end{bmatrix} = \begin{bmatrix} s_2 \\ c_1 \end{bmatrix}$$

iii.) • DETERMINE CONSTRAINTS
• INIT. COND.
• FINAL Cond.

FINAL CONSTRAINTS: $x_f = 1$, $\dot{x}_f$ = FREE $\Rightarrow$ $s_{1f} = 1$, s_{2f} = FREE

$\therefore \bar{\Omega} = [\, s_{1f} - 1 \,]$ ONLY FINAL CONSTRAINT SINCE WE ARE OPTIMIZING s_{f2}!

INITIAL CONSTRAINTS:

$$\textcircled{x} = \begin{bmatrix} s_{10} \\ s_{20} \end{bmatrix} = \bar{0}$$

iv) BOLZA FUNCTION.

$$g = K + \bar{\omega}^T \bar{\Omega} + \theta^T \textcircled{x}$$ (SINCE $\bar{0}$)

AND SINCE $\dot{x}_f \equiv s_{2f}$

$$g = s_{2f}^2 + \omega_1 (s_{1f} - 1) \leftarrow$$

HAMILTON.

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

$$= \tfrac{1}{2} a^2 + \lambda^T f$$

$$= \tfrac{1}{2} c_1^2 + \lambda_1 (s_2) + \lambda_2 (c_1) \leftarrow$$

v.) NECESSARY CONDITIONS ($\dot{\lambda} = -H_s^T$) AND NATURAL B.C.S.

$\bar{\lambda}_f = g_{sf}$

COSTATES. $\dot{\lambda}$, $\bar{\lambda}_f$

CONTROL LAW: $\frac{\delta H}{\delta c} = 0$

$$\therefore \frac{\delta H}{\delta c_1} = \frac{\delta}{\delta c_1}\left\{\tfrac{1}{2}c_1^2 + \lambda_1(s_2) + \lambda_2(c_1)\right\}$$

$$\frac{\delta H}{\delta c_1} = c_1 + \lambda_2 = 0$$

$$\therefore c_1 = -\lambda_2 \leftarrow$$

SOLVE FOR COSTATES.

$$\dot{\lambda} = -H_s^T \begin{cases} \dot{\lambda}_1 = -\frac{\delta H}{\delta s_1} = 0 \\ \dot{\lambda}_2 = -\frac{\delta H}{\delta s_2} = -(\lambda_1) \end{cases}$$

COSTATE EQNS $\Rightarrow$

$$\therefore \dot{\lambda}_1 = 0 \leftarrow$$

$$\dot{\lambda}_2 = -\lambda_1 \leftarrow$$

REWRITE THE COSTATE FUNCTION:

$$\bar{f} = \dot{\bar{s}} = \begin{bmatrix} \dot{s}_1 \\ \dot{s}_2 \end{bmatrix} = \begin{bmatrix} s_2 \\ c_1 \end{bmatrix} = \begin{bmatrix} s_2 \\ -\lambda_2 \end{bmatrix} \leftarrow$$

AND $\begin{bmatrix} \dot{\lambda}_1 \\ \dot{\lambda}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -\lambda_1 \end{bmatrix} \leftarrow$

NATURAL BOUNDARY CONDITIONS.

$$\lambda_f = G_{sf}$$

$$\therefore \lambda_{1f} = \frac{\delta G}{\delta s_1} = \frac{\delta}{\delta s_1}\left\{ s_{2f}^2 + \omega_1 (s_{1f} - 1) \right\}$$

$$\lambda_{1f} = \omega_1 \leftarrow$$

$$\lambda_{2f} = \frac{\delta G}{\delta s_2} = \frac{\delta}{\delta s_2}\left\{ s_{2f}^2 + \omega_1 (s_{1f} - 1) \right\}$$

$$\lambda_{2f} = 2 s_{2f} \leftarrow$$

(c.) SOLVE $\lambda \Rightarrow$ STATES.

$\dot{\lambda}_1 = 0$, $\therefore \lambda_1$ IS A CONSTANT.

AND $\lambda_{1f} = \omega_1$

$$\therefore \lambda_{10} = \lambda_{1f} = \lambda_1 = \omega_1 \leftarrow$$

$$\dot{\lambda}_2 = -\lambda_1$$

$$\dot{\lambda}_2 = -\omega_1 \leftarrow$$

SOLVE FOR λ_2.

$$\int \dot{\lambda}_2 = \int -\omega_1 \, dt$$

$$\lambda_2 = -\omega_1 t + \alpha$$

CONST. OF INTEGRATION

FIND α

AT t_f :

$$\lambda_{2f} = 2s_{2f}$$

$$\therefore \quad \lambda_2(t_f) = -\omega_1 t + a = 2s_{2f}$$

$$-\omega_1 + a = 2s_{2f}$$

$$a = 2s_{2f} + \omega_1 \leftarrow$$

THUS,

$$\lambda_2 = -\omega_1 t + 2s_{2f} + \omega_1 \leftarrow$$

$$\dot{s_2} = c_1 = -\lambda_2$$

$$\therefore \quad \dot{s_2} = \omega_1 t - 2s_{2f} - \omega_1$$

$$\int \dot{s_2} = \int \omega_1 t - 2s_{2f} - \omega_1$$

CONST. OF INTEGRATION ↓

$$s_2 = \tfrac{1}{2}\omega_1 t^2 - 2s_{2f} t - \omega_1 t + \beta$$

FIND β

At t_0 $\quad s_2(0) = \dot{x}(0) = 0$

$$s_2(0) = 0 = \tfrac{1}{2}\omega_1 t^2 - 2s_{2f} t - \omega_1 t + \beta$$

$$\therefore \beta = 0$$

SO $\quad s_2 = \tfrac{1}{2}\omega_1 t^2 - 2s_{2f} t - \omega_1 t \leftarrow$

$$s_2 = \dot{s_1}$$

$$\therefore \quad \dot{s_1} = \tfrac{1}{2}\omega_1 t^2 - 2s_{2f}t - \omega_1 t$$

INTEGRATE

$$\int \dot{s_1} = \int \tfrac{1}{2}\omega_1 t^2 - 2s_{2f}t - \omega_1 t$$

CONST. OF INTEGR.

$$s_1 = \tfrac{1}{2}\omega_1 \cdot \tfrac{1}{3}t^3 - 2s_{2f}\cdot\tfrac{1}{2}t^2 - \tfrac{1}{2}\omega_1 t^2 + \gamma$$

FIND γ.

AT $t=0$, $s_1 = \gamma = 0$ (BY INSPECTION)

$$\therefore \quad s_1 = \tfrac{1}{6}\omega_1 t^3 - s_{2f}t^2 - \tfrac{1}{2}\omega_1 t^2 \longleftarrow$$

NOW WE HAVE EXPRESSIONS FOR s_1 AND s_2 IN TERMS OF ω_1 AND s_{2f}.

FREE

TWO EQNS, TWO UNKNOWNS (ω_1, s_{2f})

$$\begin{cases} s_1 = \tfrac{1}{6}\omega_1 t^3 - s_{2f}t^2 - \tfrac{1}{2}\omega_1 t^2 & E_1 \\ s_2 = \tfrac{1}{2}\omega_1 t^2 - 2s_{2f}t - \omega_1 t & E_2 \end{cases}$$

SOLVE AT $t = t_f = 1$

E_1) $\quad s_1' = \tfrac{1}{6}\omega_1 t^3 - s_{2f}t^2 - \tfrac{1}{2}\omega_1 t^2$

$$s_{2f} = \tfrac{1}{6}\omega_1 - 1 - \tfrac{1}{2}\omega_1$$

$$s_{2f} = -\frac{1}{3}\omega_1 - 1 \leftarrow$$

$E_2)$ $\quad s_2 = \frac{1}{2}\omega_1 t^2 - 2s_{2f}t - \omega_1 t$

$$-\frac{1}{3}\omega_1 - 1 = \frac{1}{2}\omega_1 - 2\left(-\frac{1}{3}\omega_1 - 1\right) - \omega_1$$

$$-\frac{1}{3}\omega_1 - 1 = \frac{1}{2}\omega_1 + \frac{2}{3}\omega_1 + 2 - \omega_1$$

$$-\frac{1}{3}\omega_1 - \frac{1}{2}\omega_1 - \frac{2}{3}\omega_1 + \omega_1 = 2 + 1$$

$$\omega_1\left(-\frac{1}{2}\right) = 3$$

$$\omega_1 = -6 \leftarrow$$

$$\therefore \quad s_{2f} = -\frac{1}{3}\omega_1 - 1$$

$$= -\frac{1}{3}(-6) - 1$$

$$s_{2f} = 1 \leftarrow$$

AND $\quad c_1 = -\eta_2$

$$c_1 = \omega_1 t - 2s_{2f} - \omega_1$$

$$c_1(t) = -6t - 2(1) - (-6)$$

$$= -6t + 4$$

$$\boxed{c_1(t) = -2(3t - 2) \\ = a}$$

OPTIMAL ACCELERATION FUNCTION

THUS, THE COST FUNCTION IS:

$$J = \dot{x}_f^2 + \frac{1}{2}\int_{t0}^{tf} a^2 dt$$

$$J = s_2 f^2 + \frac{1}{2}\int_{t_0}^{t_f} (-6t+4)^2 \, dt$$

$$= 1 + \frac{1}{2}\int_{t_0}^{t_f} \left[-2(3t-2)\right]^2 dt$$

$$J = 1 + 2\int_{t_0}^{t_f} (3t-2)^2 \, dt \longleftarrow$$

SOLVE FOR J, $x(t)$, $\dot{x}(t)$.

$$J = 1 + 2\int_{t_0}^{t_f} (9t^2 - 12t + 4)\, dt$$

$$= 1 + 2\left[3t^3 - 6t^2 + 4t \right]_{t_0}^{t_f}$$

$$= 1 + 2\left[3t_f^3 - 6t_f^2 + 4t_f \right]$$

FOR $t_0 = 0$, $t_f = 1$

$$J = 1 + 2(3 - 6 + 4)$$

$$= 1 + 2$$

$$J = 3 \longleftarrow$$

COST INDEX

$$x(t) = s_1 = \frac{1}{6}\omega_1 t^3 - s_{2f}t^2 - \frac{1}{2}\omega_1 t^2$$

$\omega_1 = -6$

$s_{2f} = 1$

$$\therefore \quad x(t) = -t^3 - t^2 + 3t^2$$

$$= -t^3 + 2t^2$$

$$\boxed{x(t) = t(2t - t^2)}$$

← POSITION FUNCTION

$$\dot{x}(t) = s_2 = \frac{1}{2}\omega_1 t^2 - 2s_{2f}t - \omega_1 t$$

$$= -3t^2 - 2t + 6t$$

$$= -3t^2 + 4t$$

$$\boxed{\dot{x}(t) = t(4 - 3t)}$$

← VELOCITY FUNCTION

Problem 3

Optimal Orbit to Maximize the Final horizontal Velocity
Consider a transfer in the absence of external forces with a constant 2D acceleration. Let β = steering angle. The equations of motion are:

$$\ddot{x} = cos\beta$$

$$\ddot{y} = sin\beta$$

Find the maximum final horizontal velocity ($\dot{x}$) with an initial position (x and y) and velocity ($\dot{x}$ and $\dot{y}$= 0) and a transfer time = 3. The final constraints are that the final height (y) = 1 and final vertical velocity ($\dot{y}$) = 0. Since the problem formation was done in class, code up using `fsolve / fmincon` and provide the final position and velocity, the initial value of the steering angle, and a plot of the steering angle over the course of the transfer.

Solution.

See Appendix B for results and code for Problem 3

The work done in class is included in Appendix B, but the important findings are given below:

$$\text{Performance index: } J = -\dot{x}_f$$

$$\bar{c} = c_1 = \beta$$

$$\bar{f} = \dot{\bar{s}} = \begin{bmatrix} s_3 \\ s_4 \\ cos\beta \\ sin\beta \end{bmatrix} = \begin{bmatrix} s_3 \\ s_4 \\ cos(c_1) \\ sin(c_1) \end{bmatrix}$$

From solving the partial derivative of the Hamiltonian with respect to the control, we solve for the control:

$$\frac{\partial H}{\partial c_1} \Rightarrow tan(c_1) = \frac{\lambda_3}{\lambda_4}$$

The costates are:

$$\dot{\lambda} = -H_s^T = \begin{bmatrix} \dot{\lambda_1} \\ \dot{\lambda_2} \\ \dot{\lambda_3} \\ \dot{\lambda_4} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\lambda_1 \\ -\lambda_2 \end{bmatrix}$$

Natural boundary conditions:

$$\bar{\lambda_f} = G_{s_f} = \begin{bmatrix} 0 \\ \omega_1 \\ -1 \\ \omega_2 \end{bmatrix}$$

Next, we solve iteratively using MATLAB's `fsolve` function. First, initial known values for the state and educated guesses for the costates were defined:

```
%..... Initial guesses
state = [0;0;0;0];       % known position, velocities (x;y;xd;yd)
lambda = [0;-1;-1;-1];   % 1st and 3rd entries known; others are -1
    by default guess
```

Propagating these initial guesses forward in time using `ode45` yields the plots in Figs. 4 and 5. The last values obtained from the ODE propagator were used as (better) initial guesses and run through the `fsolve` function and the best estimate for the initial state was obtained (see Fig. 6). This initial state, s_0, was then propagated forward to obtain the final state, s_f.

After running `fsolve`, the initial state is:

$$s_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -0.0000 \\ -0.0000 \\ -0.5153 \\ -1.0000 \\ -0.7730 \end{bmatrix}$$

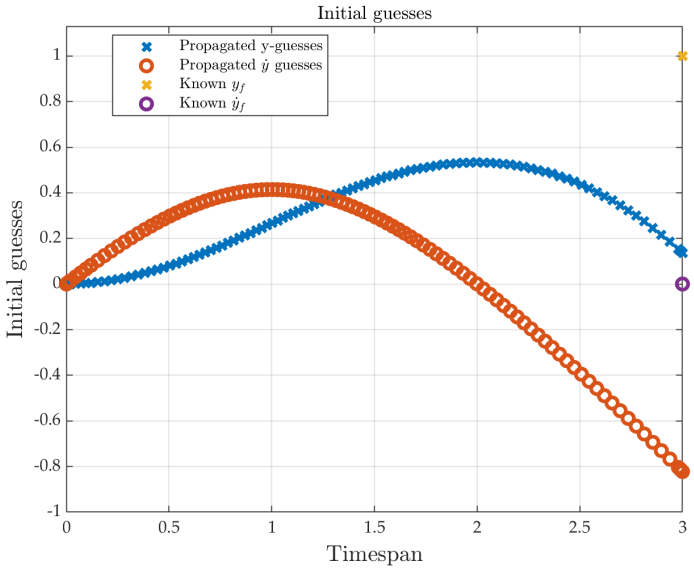


Figure 4: Initial guesses propagated forward to check for general look and feel–do the guesses generally approach the desired final points? They do, so we continue with these guesses.

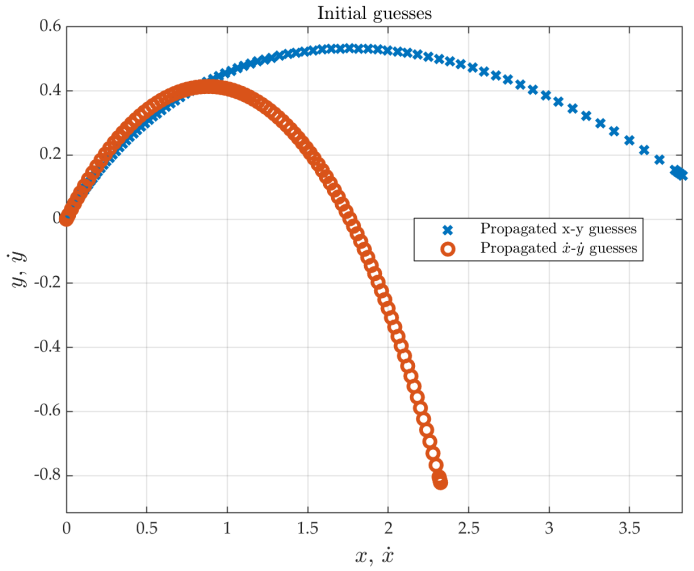


Figure 5: Initial guesses plotted by x and y coordinates. They start at zero and do not go negative or behave otherwise erratically. Good to proceed to the next step.

After propagating the initial state we obtain the final states:

$$s_f = \begin{bmatrix} 4.1417 \\ 1.0000 \\ 2.7611 \\ 0.0000 \end{bmatrix}$$

Iteration	Func-count	||f(x)||^2	Norm of step	First-order optimality	Trust-region radius
0	9	91.5811		27.2	1
1	18	45.3227	1	15.7	1
2	27	21.0728	2.5	8.67	2.5
3	36	2.07426	2.5	1.12	2.5
4	45	0.0370534	1.40091	1.06	6.25
5	54	3.29644e-05	0.0640213	0.0269	6.25
6	63	2.72242e-10	0.00580442	6.75e-05	6.25
7	72	3.17941e-20	2.18675e-05	6.3e-10	6.25

Equation solved.

Figure 6: Final outputs from the `fsolve` iterative scheme.

$$\lambda_f = \begin{bmatrix} -0.0000 \\ -0.5153 \\ -1.0000 \\ 0.7730 \end{bmatrix}$$

The parameter we have been maximizing, $\dot{x}_f$, is:

$$\dot{x}_f = 2.7611 \text{ D/T units}$$

The final state position vector is shown in Fig. 7 and the steering angle is plotted against time and is shown in Fig. 8.

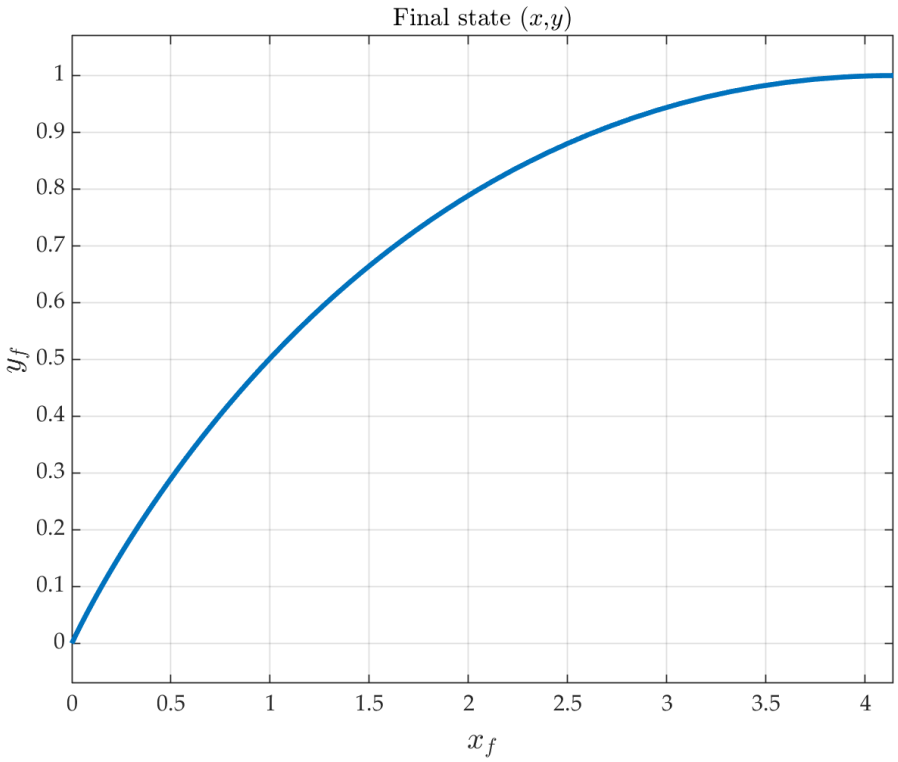


Figure 7: Final orbit (position)

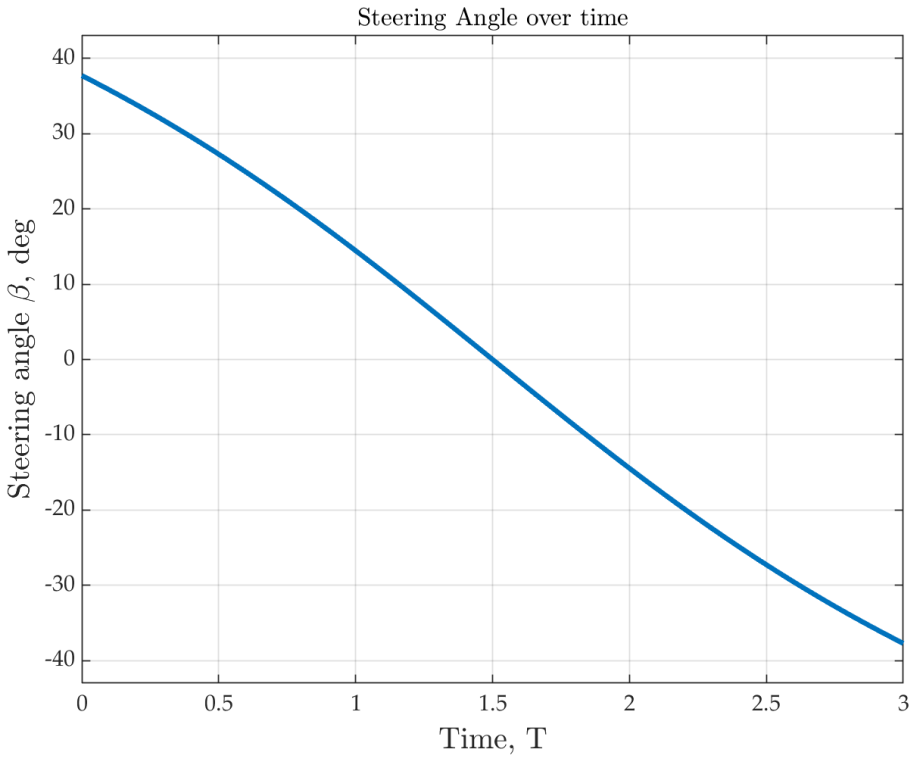


Figure 8: Steering angle over time

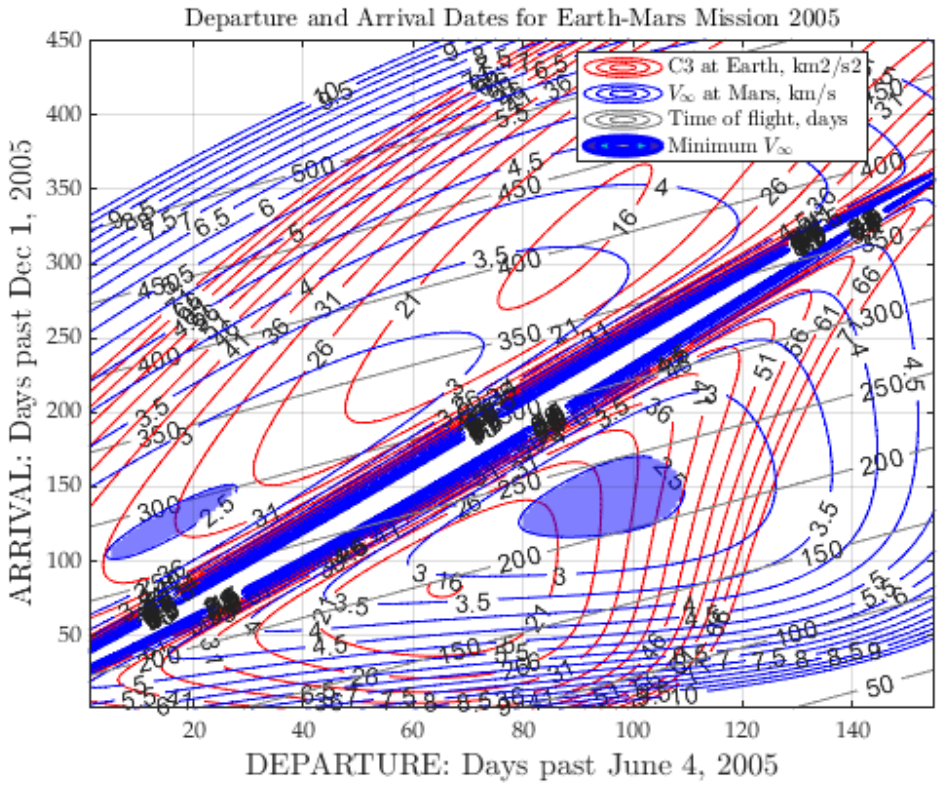
Appendix A: Problem 1 Results and Code

Table of Contents

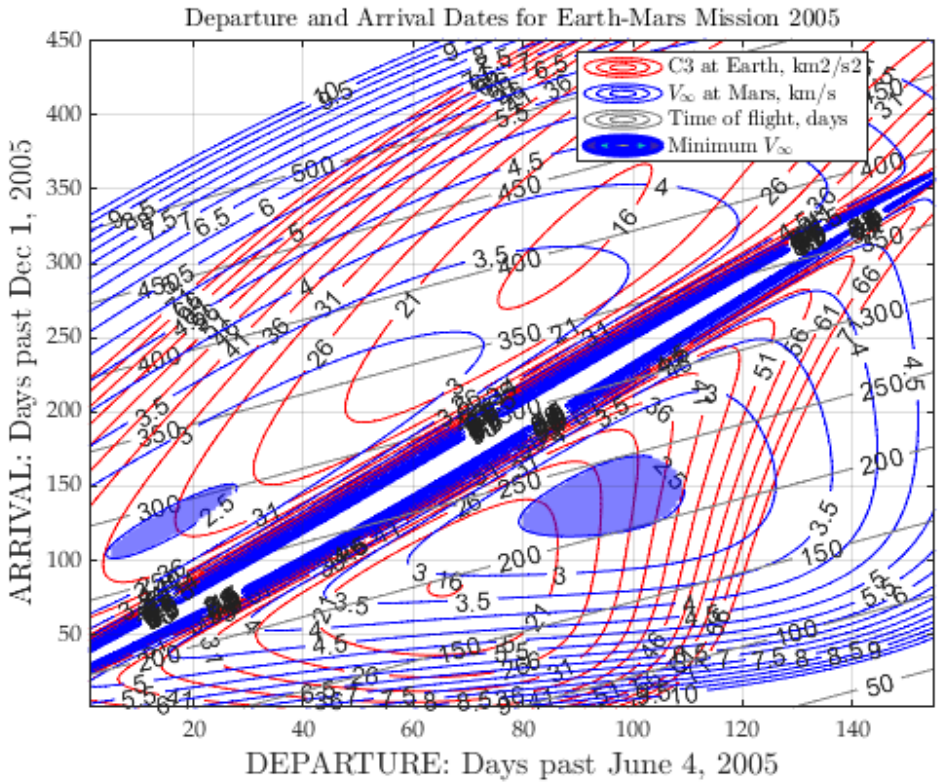
Problem 1: Pork Chop Plot

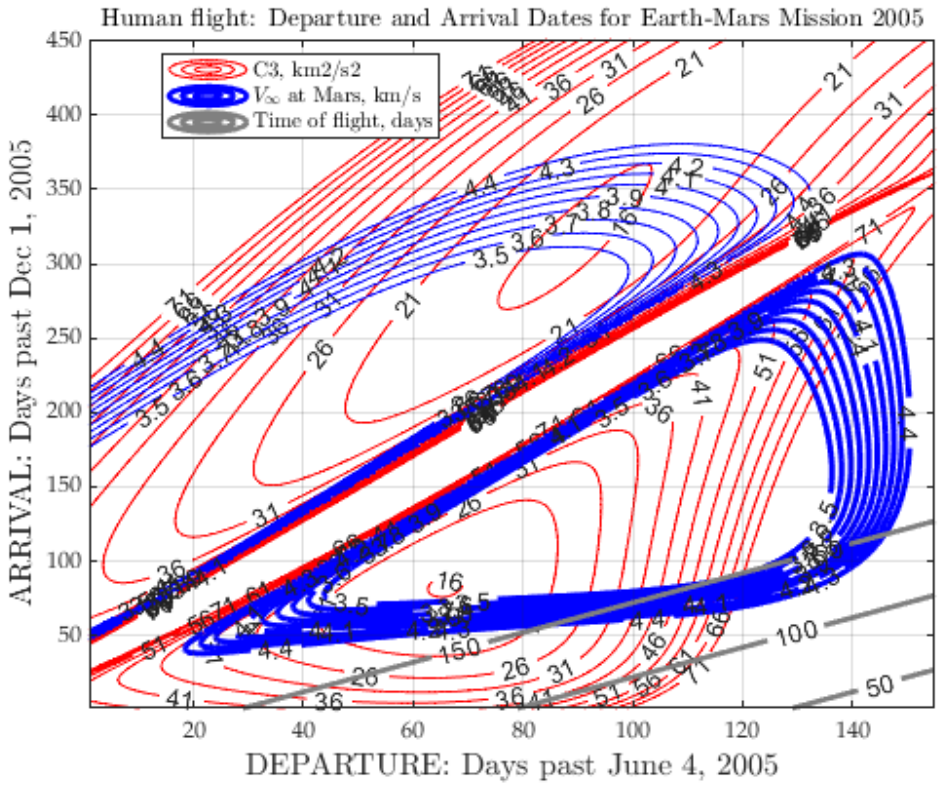
Elapsed time is 10.114989 seconds.

Porkchop plots



P1.b: Humans to Mars





Published with MATLAB® R2023b

Table of Contents


```
%{
Justin Self
California Polytechnic State University
AERO557 | Advanced Orbital Mechanics
Dr. Abercromby, WINTER 2024
HW #3
%}

% Housekeeping
clear all; close all; clc;

% Add Path
addpath('C://MATLAB_CODE/Orbits/')
tic
```

Problem 1: Pork Chop Plot

```
%{
Problem 1: Pork Chop Plot

Generate a pork chop plot for the 2005 Earth to Mars opportunity.
Use Earth departure dates from JD = 2453528.0 (June 6th , 2005) to 2453682.0
(Nov. 7 th , 2005)
and Mars arrival dates
from JD = 2453706.0 (Dec. 1 st , 2005) to 2454156.0 (Feb. 24th , 2007).

Plot C3 at Earth (km 2 /s 2 ), V_infinity at Mars (km/s), and time of
transfer (days).

The plot should look like the figure I showed you in class.

Use the plot to answer the following questions. Please turn in the plot with
the homework.

a) What do you think is the optimal departure/arrival combination for a
robotic mission that
wishes to minimize total deltaV without using any aerobraking? (Just eyeball
it - a range of
answers is certainly acceptable.) Is this a Type I or a Type II trajectory?

b) What is the optimal departure/arrival combination for a human mission to
Mars, given the
maximum flight duration of 150 days? Assume that the mission will implement
```

```
a direct
transfer descent to the Martian surface, i.e., it will not enter into a Mars
orbit. Also assume
that the entry vehicle can only withstand the heat generated from a
trajectory with a V ∞ <
4.5 km2 /s 2
%}

% Constants
mu = 132712440018; % km3/s2; mu sun

% Define dates (yyyy mm dd hh:mm:ss); all in UTC
date.depart.first = 2453528.0;
date.depart.last = 2453682.0;

date.arrive.first = 2453706.0;
date.arrive.last = 2454156.0;

date.depart.vector = date.depart.first : date.depart.last;
date.arrive.vector = date.arrive.first : date.arrive.last;

% Preallocate
v.depart.short = zeros(3,length(date.depart.vector));
v.depart.long = zeros(3,length(date.depart.vector));
v.arrive.short = zeros(3,length(date.arrive.vector));
v.arrive.long = zeros(3,length(date.arrive.vector));
dtArray = zeros(length(date.depart.vector),length(date.arrive.vector));
vinf.array.short =
zeros(length(date.depart.vector),length(date.arrive.vector));
vinf.array.long =
zeros(length(date.depart.vector),length(date.arrive.vector));

for i = 1:length(date.depart.vector)

        %.... Determine location of earth at departure
        jd.depart = date.depart.vector(i);
        t.depart = datetime(jd.depart,"ConvertFrom","juliandate");
        d = datevec(t.depart);
        earth = 3;
        [~, r.e, v.e, ~] = AERO557planetcoe_and_sv(earth, d(1), d(2), d(3),
d(4), d(5), d(6));
    for j = 1:length(date.arrive.vector)

            %.... Determine location of Mars at arrival
            jd.arrive = date.arrive.vector(j);
            t.arrive = datetime(jd.arrive,"ConvertFrom","juliandate");
            d = datevec(t.arrive);
            mars = 4; % mars
            [~, r.m, v.m, ~] = AERO557planetcoe_and_sv(mars, d(1), d(2),
d(3), d(4), d(5), d(6));

            % Use Lambert with R departure and R arrival and dt to get V for
the orbit at arrival and departure.
            dt = (jd.arrive - jd.depart); % in days
```

```
            m = 0; % no rev solution
            % SHORT WAY
            [v.depart.short,v.arrive.short,~,~] =
lambert_izzo(r.e',r.m',dt,m,mu);
            % LONG WAY
            [v.depart.long,v.arrive.long,~,~] = lambert_izzo(r.e',r.m',-
dt,m,mu);

            % Calculate time of flight (part A)
            dtArray(i,j) = dt;

            % Calculate v_inf departure (part A)
            vinf.array.short(i,j) = norm(v.depart.short' - v.e); % km/s
            vinf.array.long(i,j) = norm(v.depart.long' - v.e); % km/s

            % Calculate v_inf arrival (part B)
            vinfArrive.array.short(i,j) = norm(v.arrive.short' - v.m); % km/s
            vinfArrive.array.long(i,j) = norm(v.arrive.long' - v.m); % km/s

            % Calculate C3
            c3.array.short(i,j) = vinf.array.short(i,j)^2;
            c3.array.long(i,j) = vinf.array.long(i,j)^2;

    end % departure loop
end % arrival loop
toc
```

Porkchop plots

```
figure
contour(c3.array.short',1:5:75,'ShowText','on','EdgeColor','r')
hold on
contour(vinfArrive.array.short',0:0.5:10,'ShowText','on','EdgeColor','b')
contour(dtArray',0:50:500,'ShowText','on','EdgeColor','#808080')
contour(c3.array.long',1:5:75,'ShowText','on','EdgeColor','r')
contour(vinfArrive.array.long',0:0.5:10,'ShowText','on','EdgeColor','b')

% Calc and show mins
vinfminS = vinfArrive.array.short;
vinfminL = vinfArrive.array.long;

for i = 1:length(date.depart.vector)
        for j = 1:length(date.arrive.vector)

        % Vinf short way
            if vinfminS(i,j) > 2.5
                vinfminS(i,j) = nan;
            end

        % Vinf short way
            if vinfminL(i,j) > 2.5
                vinfminL(i,j) = nan;
            end
```

```
        end
end

contour(vinfminS',0:0.5:2.5,'FaceColor','b','FaceAlpha',0.5,'ShowText','on','
EdgeColor','b','LineWidth',2)
contour(vinfminL',0:0.5:2.5,'FaceColor','b','FaceAlpha',0.5,'ShowText','on','
EdgeColor','b','LineWidth',2)

% Graph pretty
ylim tight
xlim tight
xLab = xlabel('DEPARTURE: Days past June 4, 2005','Interpreter','latex');
yLab = ylabel('ARRIVAL: Days past Dec 1, 2005','Interpreter','latex');
plotTitle = title('Departure and Arrival Dates for Earth-Mars Mission
2005','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 12)
grid on
legend('C3 at Earth, km2/s2','$V_\infty$ at Mars, km/s','Time of flight,
days','','','Minimum $V_\infty$', 'interpreter','latex','Location', 'best')

% Plot 1.a discussion: Overleaf doc
```

P1.b: Humans to Mars

```
%{
b) What is the optimal departure/arrival combination for a human mission to
Mars, given the
maximum flight duration of 150 days? Assume that the mission will implement
a direct
transfer descent to the Martian surface, i.e., it will not enter into a Mars
orbit. Also assume
that the entry vehicle can only withstand the heat generated from a
trajectory with a V ∞ <
4.5 km2 /s 2
%}

% Modify parameters per constraints
    % Max flight duration
    for i = 1:length(date.depart.vector)
        for j = 1:length(date.arrive.vector)

        % Vinf short way
            if vinfArrive.array.short(i,j) > 4.5
                vinfArrive.array.short(i,j) = nan;
            end

        % Vinf short way
            if vinfArrive.array.long(i,j) > 4.5
```

```
                vinfArrive.array.long(i,j) = nan;
            end

        % Flight time
            if dtArray(i,j) > 150
                dtArray(i,j) = nan;
            end
        end
    end

%..... Plot human mission
figure
contour(c3.array.short',1:5:75,'ShowText','on','EdgeColor','r')
hold on
contour(vinfArrive.array.short',3.5:0.1:4.5,'ShowText','on','EdgeColor','b','
LineWidth',2)
contour(dtArray',0:50:150,'ShowText','on','EdgeColor','#808080','LineWidth',2
)
contour(c3.array.long',1:5:75,'ShowText','on','EdgeColor','r')
contour(vinfArrive.array.long',3.5:0.1:4.5,'ShowText','on','EdgeColor','b')

% Graph pretty
ylim tight
xlim tight
xLab = xlabel('DEPARTURE: Days past June 4, 2005','Interpreter','latex');
yLab = ylabel('ARRIVAL: Days past Dec 1, 2005','Interpreter','latex');
plotTitle = title('Human flight: Departure and Arrival Dates for Earth-Mars
Mission 2005','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 12)
grid on
legend('C3, km2/s2','$V_\infty$ at Mars, km/s','Time of flight, days',
'interpreter','latex','Location', 'best')
```

Published with MATLAB® R2023b

Appendix B: Problem 3 Results and Code

Table of Contents

Front matter

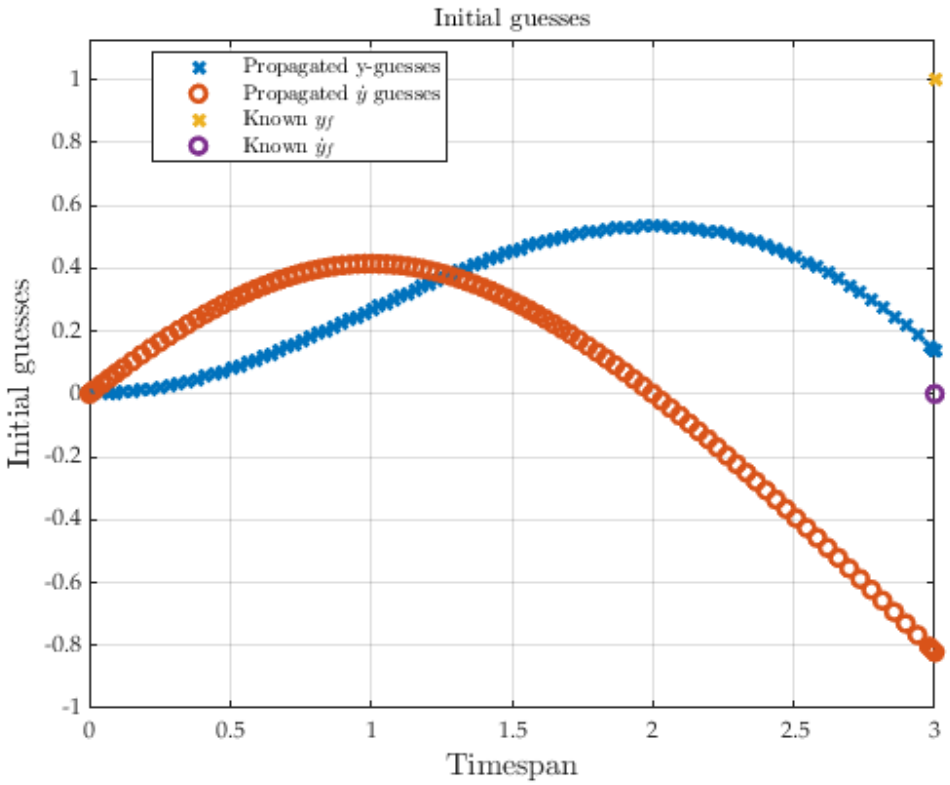
Initialization

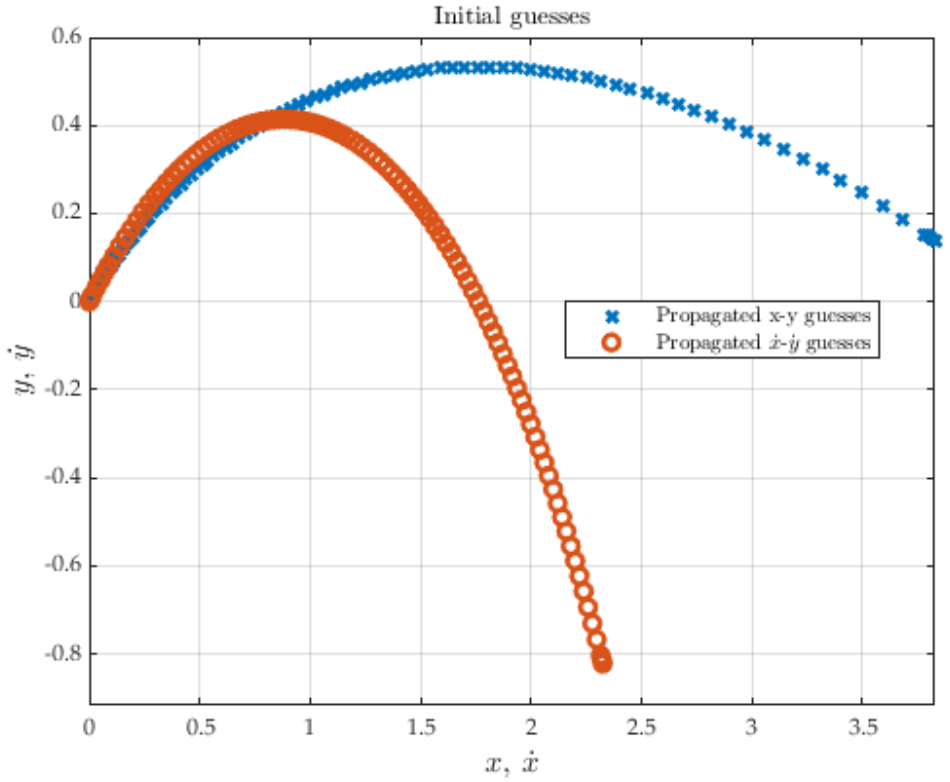
Housekeeping

```
***HW 3 Part 3: Optimal Orbit to maximize final horizontal velocity***
```

Check guesses

```
HEART CHECK: The guesses generally approach the final values. Good to
continue.
HEART CHECKS: These don't go negative or do anything weird; good. <<<
-------
```


Initial guesses
Propagated y-guesses
Propagated $\dot{y}$ guesses
Known y_f
Known $\dot{y}_f$
Initial guesses
Timespan


Initial guesses
Propagated x-y guesses
Propagated $\dot{x}$-$\dot{y}$ guesses
$y,\ \dot{y}$
$x,\ \dot{x}$

Use good guesses to run FSOLVE

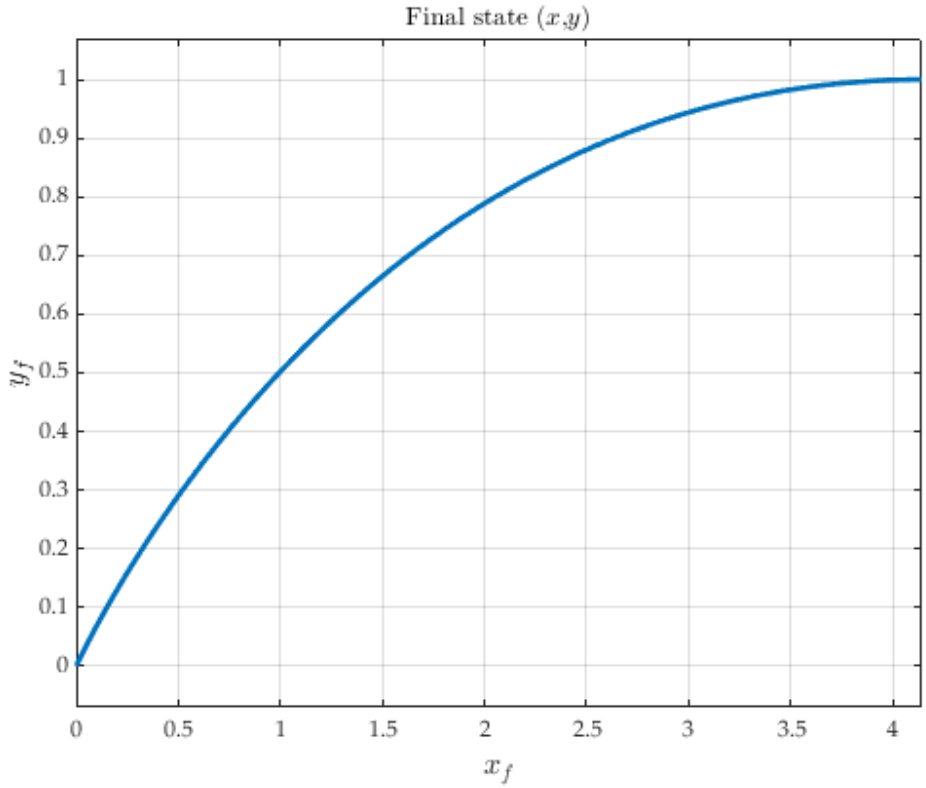
```
                                                Norm of      First-order
Trust-region
 Iteration  Func-count     ||f(x)||^2            step
optimality         radius
     0          9              91.5811
27.2               1
     1         18              45.3227               1
15.7               1
     2         27              21.0728             2.5
8.67             2.5
     3         36              2.07426             2.5
1.12             2.5
     4         45            0.0370534         1.40091
1.06            6.25
     5         54           3.29644e-05       0.0640213
0.0269            6.25
     6         63           2.72242e-10      0.00580442
6.75e-05             6.25
     7         72           3.17941e-20     2.18675e-05
6.3e-10             6.25

Equation solved.

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

Results

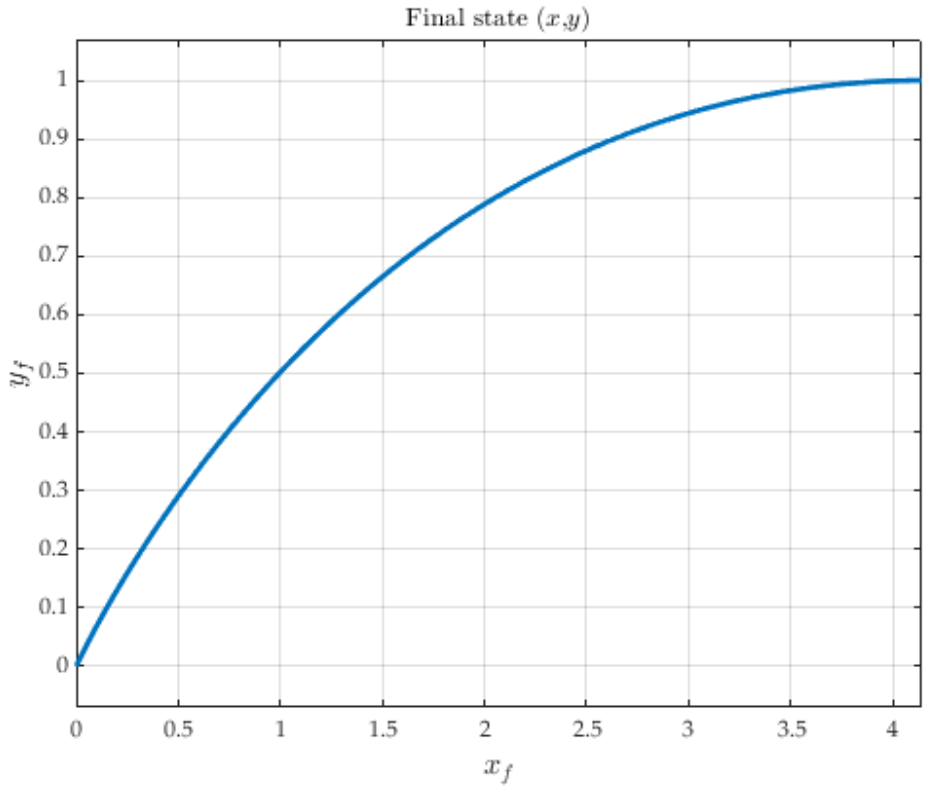
```
S0 is:
         0
         0
         0
   -0.0000
   -0.0000
   -0.5153
   -1.0000
   -0.7730

Final state (sf) is:
    4.1417
    1.0000
    2.7611
    0.0000

Final lambdas (lf) are:
   -0.0000
   -0.5153
```

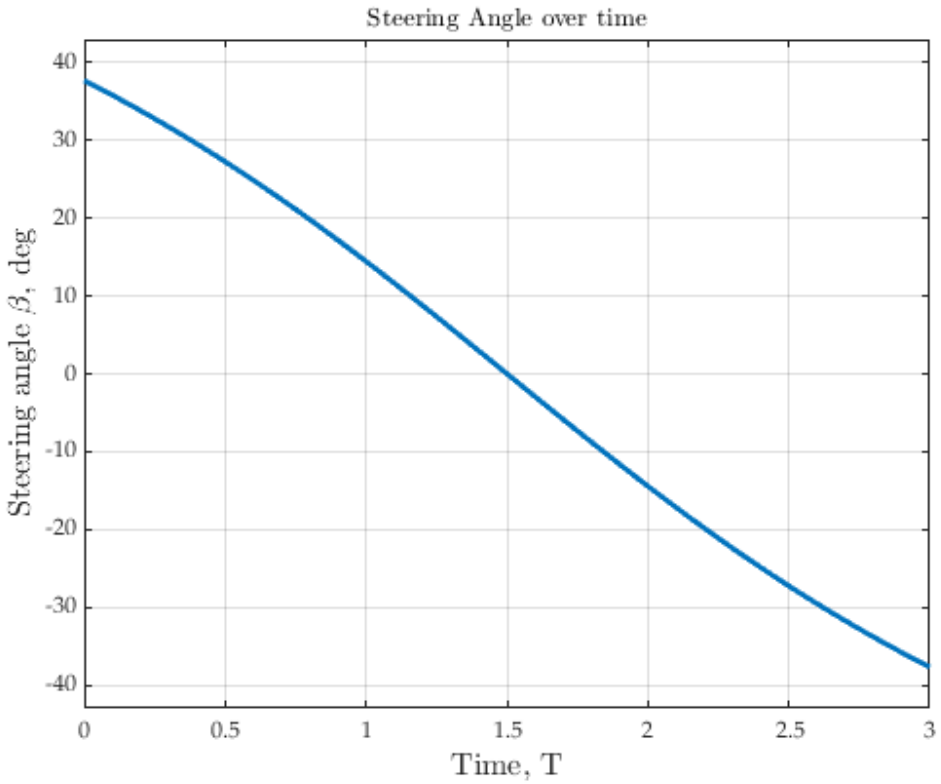
```
   -1.0000
    0.7730

Final horizontal velocity (xdot_f) is: 2.7611 D/T units <<< -------
```

Plot final orbit



Plot steering angle





Published with MATLAB® R2023b

Table of Contents

Front matter

```
%{
Justin Self
California Polytechnic State University
AERO557 | Advanced Orbital Mechanics
Dr. Abercromby, WINTER 2024
HW #3, part 3

PROBLEM 3: Optimal Orbit to maximize the final horizontal velocity (set up
worked through
in class)

Consider a transfer in the absence of external forces with a constant 2D
acceleration.
Let beta =  steering angle. The equations of motion are:

xddot = cosBeta
yddot = sinBeta

Find the maximum final horizontal velocity (xdot) with an initial position
(x and y) and velocity
(xdot = ydot = 0) and a transfer time = 3.

The final constraints are that the final height (y) = 1 and final vertical
velocity ydot = 0.

Since the problem formation was done in class, code up using FSOLVE/FMINCON
and provide
the final position and velocity, the initial value of the steering angle,
and a plot of the steering
angle over the course of the transfer.

%}
```

Initialization

Housekeeping

```
clear all; close all; clc;
```

```
% Add Path
addpath('C://MATLAB_CODE/Orbits/')

disp("***HW 3 Part 3: Optimal Orbit to maximize final horizontal
velocity***")
disp(" ")

%..... Initial guesses (0 = initial; f = final)
state = [0;0;0;0];      % known position, velocities (x;y;xd;yd)
lambda = [0;-1;-1;-1];  % 1st and 3rd entries known; others are -1 by
default guess
y0 = [state;lambda];

%..... Final states (known ones)
yf = 1;              % AKA sf2
ydf = 0;             % AKA sf4
tf = 3; % final time

%..... Find correct initial lambda values
ode_options = odeset("RelTol",1e-8,"AbsTol",1e-8);

tspan = [0 tf];
[timenew,statenew] = ode45(@optimizationODE,tspan,y0,ode_options);
```

Check guesses

```
figure
% Check guesses
plot(timenew,statenew(:,2),'x','LineWidth',2)       % y
hold on
plot(timenew,statenew(:,4),'o','LineWidth',2)       % ydot

% Plot final state
plot(tspan(2),yf,'x','LineWidth',2)                 % final position, y
plot(tspan(2),ydf,'o','Linewidth',2)                % final velocity, y

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

disp("HEART CHECK: The guesses generally approach the final values. Good to
```

```
continue.")

% .... Check x, y plots (don't want this going negative or anything weird)
figure
% Check guesses
plot(statenew(:,1),statenew(:,2),'x','LineWidth',2)        % y vs x
hold on
plot(statenew(:,3),statenew(:,4),'o','LineWidth',2)        % ydot vs x dot

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

disp("HEART CHECKS: These don't go negative or do anything weird; good. <<<
------- ")
```

Use good guesses to run FSOLVE

```
s0_guess = statenew(end,1:4);     % state vectors s1, s2, s3, s4
l0_guess = statenew(end,5:8);     % costates: lambda1, lambda2, etc.
state = [s0_guess';l0_guess'];    % best initial guesses

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

Results

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

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

% Extract vals
sf1 = statenew(:,1); % x-position
sf2 = statenew(:,2); % y-position
sf3 = statenew(:,3); % x-velocity < -- maximize this
sf4 = statenew(:,4); % y-velocity
lamf1 = statenew(:,5);
```

```
lamf2 = statenew(:,6);
lamf3 = statenew(:,7);
lamf4 = statenew(:,8);

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

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

% COST FUNCTION: J = -xdotf = -sf3 (maximize xdot_final)
disp("Final horizontal velocity (xdot_f) is: " + sf3(end) + " D/T units <<<
------- ")
```

Plot final orbit

```
figure
plot(sf1,sf2,'LineWidth',2) % x,y final
% Graph pretty
ylim padded
xlim tight
xLab = xlabel('$x_f$','Interpreter','latex');
yLab = ylabel('$y_f$','Interpreter','latex');
plotTitle = title('Final state ($x$,$y$)','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 12)
grid on
```

Plot steering angle

```
beta = atand(lamf4./lamf3);
figure
plot(timenew,beta,'LineWidth',2)
% Graph pretty
ylim padded
xlim tight
xLab = xlabel('Time, T','Interpreter','latex');
yLab = ylabel('Steering angle $\beta$, deg','Interpreter','latex');
plotTitle = title('Steering Angle over time','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 12)
grid on
```

Published with MATLAB® R2023b

Functions Used (all problems)

Table of Contents


```
%{
Justin Self
California Polytechnic State University

FSOLVE solver for optimization problems. Developed for
AERO557 | Advanced Orbital Mechanics
Dr. Abercromby, WINTER 2024
HW #3, part 3

This function performes trajectory optimization for 2D orbit state with
known constraints. VERY SPECIFIC TO HW 3 PART 3 IN A557.
%}

function F = optimizationFun(state,tf)

% Call ode
tspan = [0,tf];
options = odeset("RelTol",1e-8,"AbsTol",1e-8);
[~,statenew] = ode45(@optimizationODE,tspan,state,options);
sf = [statenew(end,1),statenew(end,2),statenew(end,3),statenew(end,4)];
lambdaf = [statenew(end,5),statenew(end,6),statenew(end,7),statenew(end,8)];

% Set up equations to check misclosures (THIS IS SPECIFIC TO THIS PROB)
F(1,1) = sf(2) - 1;
F(2,1) = sf(4);
F(3,1) = lambdaf(1);
F(4,1) = lambdaf(3) + 1;

% Initial conditions (forces these values to never change; that is,
% x0 = y0 = xdot0 = ydot0 = 0

F(5,1) = statenew(1,1);
F(6,1) = statenew(1,2);
F(7,1) = statenew(1,3);
F(8,1) = statenew(1,4);

end % fsolve function
```

Published with MATLAB® R2023b

Table of Contents


```
%{
Justin Self
California Polytechnic State University

ODE solver for optimization problems. Developed for
AERO557 | Advanced Orbital Mechanics
Dr. Abercromby, WINTER 2024
HW #3, part 3

This function propagates 4 state variables and 4 costates.
%}

function dstate = optimizationODE(time,state)

lambda = [state(5);state(6);state(7);state(8)];

dstate(1) = state(3);
dstate(2) = state(4);
dstate(3) = -lambda(3) / sqrt(lambda(3)^2 + lambda(4)^2);
dstate(4) = -lambda(4) / sqrt(lambda(3)^2 + lambda(4)^2);

dlam = [0;0;-lambda(1);-lambda(2)];
dstate = [dstate(1);dstate(2);dstate(3);dstate(4);dlam];

end % function
```

Published with MATLAB® R2023b