ASEN 5050, HW 6
Fall 2024
Jash Bhalavat

Problem 1

- Part a

```
% Step 5 - iteratively solve for a
a_initial_guess = am + delta_a;

% Get function where the variable is a
fun = @(a)lamberts_eqn_elliptical(a, s, c, TOF, mu, TOF_min, greater_than_180);

% Use fsolve to output a where function goes to 0
a = fsolve(fun, a_initial_guess);
```

```
function fx = lamberts_eqn_elliptical(a, s, c, TOF, mu, TOF_min,
greater_than_180)
    % Output function to solve for fsolve
    % Inputs:
    %       a - Semi-major axis of transfer conic [km]
    %       s - semi-perimeter of transfer triangle [km]
    %       c - chord length of transfer triangle [km]
    %       TOF - time of flight of transfer [s]
    %       mu - gravitational parameter of central body [km^3/s^2\
    %       TOF_min - TOF for minimum energy transfer [s]
    %       greater_than_180 - bool that's true if chosen arc is >180 deg
    % Output:
    %       fx - function that's dependent on a to pass to fsolve

    % Mean anomaly for transfer ellipse
    n = sqrt(mu/a^3);

    % Calculate principal alpha, beta
    alpha_0 = 2 * asin(sqrt(s/(2*a)));
    beta_0 = 2 * asin(sqrt((s-c)/(2*a)));

    % Get correct alpha, beta dependent on TOF and (choice of)
    % greater_than_180 bool
    alpha_beta = lamberts_problem_alpha_beta_logic(TOF, TOF_min,
greater_than_180, alpha_0, beta_0);
    alpha = alpha_beta(1);
    beta = alpha_beta(2);

    % Output function

    fx = TOF - (alpha - beta - (sin(alpha) - sin(beta)))/n;
end
```

```
function out = lamberts_problem_alpha_beta_logic(TOF, TOF_min,
greater_than_180, alpha_0, beta_0)
    % Function that outputs appropriate alpha and beta depending on time of
    % flight (TOF) and choice of arc size (greater than 180 or no)
    % Inputs:
    %       TOF - Time of flight of transfer arc [s]
    %       TOF_min - Minimum energy arc TOF [s]
    %       greater_than_180 - bool that's true if chosen arc is >180 deg
    %       alpha_0, beta_0 - principal alpha, beta [rad]
    % Output:
    %       alpha, beta - actual alpha and beta [rad]

    % Logic from ASEN5050 Lambert's Problem alpha/beta table
    if TOF < TOF_min
        if greater_than_180
            % Purple arc
            alpha = alpha_0;
            beta = -beta_0;
        else
            % Blue arc
            alpha = alpha_0;
            beta = beta_0;
        end
    else
        if greater_than_180
            % Brown arc
            alpha = 2*pi - alpha_0;
            beta = -beta_0;
        else
            % Red arc
            alpha = 2*pi - alpha_0;
            beta = beta_0;
        end
    end

    out = [alpha, beta];
end
```

- Setup of numerical method - As seen above, the numerical method uses fsolve to solve for a where fsolve tries to solve for the lambert's equation by changing the semi-major axis (a) of the transfer arc.
- Stopping conditions - The fsolve default stopping condition is used which is 1e-6. Since, the transfer is between earth and venus and the magnitude of the semi-major axis is likely really large, this stopping condition is deemed fit. Additionally, even for smaller transfer arcs, this stopping condition is enough.
- Initial guess - an initial guess of a_min + delta_a is used where a_min is the semi-major axis of the minimum energy transfer ellipse and delta_a is a small number (0.1 km in this case).

ASEN5050
Jash Bhalerao
Fall 2024

HW6

Problem 1 → Given: Earth to Venus, less than 180°

$t_1 = 24577543$ TDB
$\vec{R}_1 = -2.688982\times10^{7}\hat{x} + 1.326980\times10^{8}\hat{y} + 5.752566\times10^{7}\hat{z}$ km
$\vec{V}_1 = -29.781722\hat{x} - 5.101774\hat{y} - 2.210394\hat{z}$ km/s

$t_2 = 2458771.5$ TDB
$\vec{R}_2 = -5.642901\times10^{7}\hat{x} - 8.571048\times10^{7}\hat{y} - 3.499446\times10^{7}\hat{z}$ km
$\vec{V}_2 = 29.658341\hat{x} - 16.091100\hat{y} - 9.116674\hat{z}$ km/s

Assumptions → 2 Body problem, $\mu = G(m_{sun} + \cancel{m_{s/c}}^{0}) - (\because m_{s/c} << m_{sun})$
$\mu = 1.327124428\times10^{11}$ km³/s², Only elliptical arcs for Lambert's problem

b) $TOF = (t_2 - t_1)\cdot 86400\frac{s}{day} = 10108800$ s

$r_1 = |\vec{R}_1| = 1.4711\times10^{8}$ km, $r_2 = |\vec{R}_2| = 1.0842\times10^{8}$ km

$\Delta\theta^* = \pm\cos^{-1}\left(\frac{\vec{R}_1\cdot\vec{R}_2}{r_1 r_2}\right) = 138.0957°$, ∵ Design choice is <180° → $\Delta\theta^* = 138.0957°$

$c = \sqrt{r_1^2 + r_2^2 - 2r_1r_2\cos(\Delta\theta^*)} = 2.3903\times10^{8}$ km

$s = \frac{1}{2}(r_1 + r_2 + c) = 2.4728\times10^{8}$ km

$TOF_p = \frac{1}{3}\sqrt{\frac{2}{\mu}}\left(s^{3/2} + (s-c)^{3/2}\right) = 5.0624\times10^{6}$ s

∵ $TOF > TOF_p$ → elliptical arc

$a_m = \frac{s}{2} = 1.2364\times10^{8}$ km $\quad n_m = \sqrt{\frac{\mu}{a_m^3}} = 2.6498\times10^{-7}\frac{rad}{s}$

$\alpha_m = \pi^{rad}_{rad}$, $\beta_{m,0} = 2\sin^{-1}\left(\sqrt{\frac{s-c}{s}}\right) = 0.3673$ rad

∵ Less than 180° → $\beta_{m,0} = \beta_m = 0.3673$ rad

$TOF_{min} = \frac{1}{n_m}(\alpha_m - \beta_m - \sin\alpha_m + \sin\beta_m) = 1.1825\times10^{7}$ s

∵ $TOF < TOF_{min}$ and less than 180° → $\alpha = \alpha_0$, $\beta = \beta_0$, Let $\Delta a = 0.1$ km

Initial guess → $a_0 = a_{min} + \Delta a = 1.2364\times10^{8}$ km

Have all inputs for Lambert's solver using fsolve() as seen in Part a

Output → $\boxed{a = 1.2581\times10^{8}\text{ km}}$, $n = \sqrt{\frac{\mu}{a^3}} = 2.5815\times10^{-7}\frac{rad}{s}$

Double check →
$\alpha_0 = 2\sin^{-1}\left(\sqrt{\frac{s}{2a}}\right) = 2.8781$ rad → $\alpha = \alpha_0 = 2.8781$ rad,
$\beta_0 = 2\sin^{-1}\left(\sqrt{\frac{s-c}{2a}}\right) = 0.3641$ rad → $\beta = \beta_0 = 0.3641$ rad
$TOF = \frac{1}{n}(\alpha - \beta - \sin\alpha + \sin\beta) = 10108800$ s ✓
$p = \frac{4a(s-r_1)(s-r_2)}{c^2}\sin^2\left(\frac{\alpha+\beta}{2}\right) = 1.2221\times10^{8}$ km, $e = \sqrt{1 - \frac{p}{a}} = \boxed{0.1693 = e}$

c] Min energy transfer is the lower bound for semi-major axis of transfer arc ($\because \epsilon = -\frac{\mu}{2a}$). $\therefore$ lower bound on semi-major axis is

$\boxed{a_m = 1.2364 \times 10^8 \text{ km}}$

d] $\theta_1^* = \pm \cos^{-1}\left(\frac{1}{e}\left(\frac{p}{r_1} - 1\right)\right) = 179.4087°$

$\theta_2^* = \pm \cos^{-1}\left(\frac{1}{e}\left(\frac{p}{r_2} - 1\right)\right) = 41.3130°$

$\Delta\theta^* = -\theta_2^* - (-\theta_1^*) = 138.0957°$

$\therefore \theta_1^* = -179.4087°, \quad \theta_2^* = -41.3130°$

$f = 1 - \frac{r_2}{p}(1 - \cos\Delta\theta^*) = -0.5475$

$g = r_1 \cdot \frac{r_2}{\sqrt{p\mu}} \sin(\Delta\theta^*) = 2.6541 \times 10^6 \text{ s}$

$\dot{f} = \sqrt{\frac{\mu}{p}} \tan\left(\frac{\Delta\theta^*}{2}\right) \frac{(1 - \cos\Delta\theta^*)}{p} - \frac{1}{r_2} - \frac{1}{r_1} = -1.5044 \times 10^{-7} \frac{1}{\text{s}}$

$\dot{g} = \left(1 - \frac{r_1}{p}\right)(1 - \cos\Delta\theta^*) = -1.0097$

$\vec{r}_2 = f\vec{r}_1 + g\vec{v}_1$

$\vec{v}_2 = \dot{f}\vec{r}_1 + \dot{g}\vec{v}_1$

$\vec{v}_{1,f} = (\vec{r}_2 - f\vec{r}_1)\frac{1}{g} = -26.8951\hat{x} - 4.9363\hat{y} - 1.3228\hat{z} \text{ km/s} = \vec{v}_{1,f}$

$\vec{v}_{2,i} = \dot{f}\vec{r}_1 + \dot{g}\vec{v}_{1,f} = 33.66175\hat{x} - 14.5349\hat{y} - 7.1996\hat{z} \text{ km/s} = \vec{v}_{2,i}$

$\Delta v_1 = |\vec{v}_{1,f} - \vec{v}_1| = \boxed{3.0246 \text{ km/s} = \Delta v_1}$

$\Delta v_2 = |\vec{v}_{2,f} - \vec{v}_2| = \boxed{4.6661 \text{ km/s} = \Delta v_2}$

```
function out = problem2_function(mu, R1, V1, R2, V2, TOF, delta_a,
greater_than_180)
    % Solve lambert's problem using fsolve using only elliptical arcs and
    % output v_infinity for porkchop plots
    % Inputs - gravitational parameter of central body,
    %          initial pos, vel, final pos, vel, Time of Flight,
    %          initial guess delta a, whether choice is greater/lower than
    %          180 deg
    % Output - v_inf_earth and v_inf_mars in km/s

    % Compute norms of initial and final positions
    r1 = norm(R1);
    r2 = norm(R2);

    % Step 1 - Transfer angle
    delta_theta_star = acos(dot(R1, R2) / (r1*r2));

    % If greater than 180, subtract original value by 360 deg to get a
    % delta theta star that's > 180 deg
    if greater_than_180
        delta_theta_star = 2*pi - delta_theta_star;
    end

    % Step 2 - calculate c and s
    c = sqrt(r1^2 + r2^2 - 2*r1*r2*cos(delta_theta_star));
    s = 0.5 * (r1 + r2 + c);

    % Step 3 is figuring out if arc is elliptical or hyperbolic but since
    % this only uses elliptical arcs, that step is eliminated.

    % Step 4 - Min energy transfer arc
    am = s/2;
    alpha_m = pi;
    nm = sqrt(mu/am^3);
    beta_m_0 = 2 * asin(sqrt((s-c)/s));
    beta_m = beta_m_0;
    if greater_than_180
        beta_m = -beta_m_0;
    end
    TOF_min = 1/nm * (alpha_m - beta_m - (sin(alpha_m) - sin(beta_m)));

    % Step 5 - iteratively solve for a
    a_initial_guess = am + delta_a;

    % Get function where the variable is a
    fun = @(a)lamberts_eqn_elliptical(a, s, c, TOF, mu, TOF_min,
greater_than_180);

    % Use fsolve to output a where function goes to 0
    options = optimoptions('fsolve', 'Display', 'off');
    a = fsolve(fun, a_initial_guess, options);
```

```
    % Compute final alpha, beta values to compare
    alpha_0_final = 2 * asin(sqrt(s/(2*a)));
    beta_0_final = 2 * asin(sqrt((s-c)/(2*a)));
    alpha_beta_final = lamberts_problem_alpha_beta_logic(TOF, TOF_min,
greater_than_180, alpha_0_final, beta_0_final);
    alpha_final = alpha_beta_final(1);
    beta_final = alpha_beta_final(2);
    n_final = sqrt(mu/a^3);

    % Step 6
    TOF_final = 1/n_final * (alpha_final - beta_final - (sin(alpha_final) -
sin(beta_final)));
    if abs(TOF_final - TOF) > 1e-4
        disp("TOFs don't match and the abs error is " + abs(TOF_final-TOF) +
" seconds.")
    end

    % Step 7 - find p and e
    p = (4*a*(s-r1)*(s-r2))/(c^2) * (sin((alpha_final + beta_final)/2))^2;
    e = sqrt(1 - p/a);

    % Step 8 - find true anomaly at each location
    theta_star_1_p = abs(acos(1/e * (p/r1 - 1)));
    theta_star_2_p = abs(acos(1/e * (p/r2 - 1)));
    theta_star_1_n = -theta_star_1_p;
    theta_star_2_n = -theta_star_2_p;

    % Test matrix
    % First column = difference between 2 and 1
    % Second column = sign of theta_star_2
    % Third column = sign of theta_star_1
    test_ones = ones(6, 2);
    test = [zeros(6, 1), test_ones];
    test(1, 1) = theta_star_2_p - theta_star_1_p;
    test(2, 1) = theta_star_2_p - theta_star_1_n;
    test(2, 3) = -1;
    test(3, 1) = theta_star_2_n - theta_star_1_p;
    test(3, 2) = -1;
    test(4, 1) = theta_star_2_n - theta_star_1_n;
    test(4, 2) = -1;
    test(4, 3) = -1;
    test(5, 1) = 2*pi - theta_star_2_p - theta_star_1_p;
    test(5, 2) = -1;
    test(6, 1) = 2*pi - theta_star_2_p - theta_star_1_n;
    test(6, 2) = -1;
    test(6, 3) = -1;

    % Parse through matrix to get correct transfer angle
    for i = 1:length(test)
        if abs(test(i) - delta_theta_star) < 1e-6
            theta_star_1 = theta_star_1_p * test(i, 3);
            theta_star_2 = theta_star_2_p * test(i, 2);
            break
        else
```

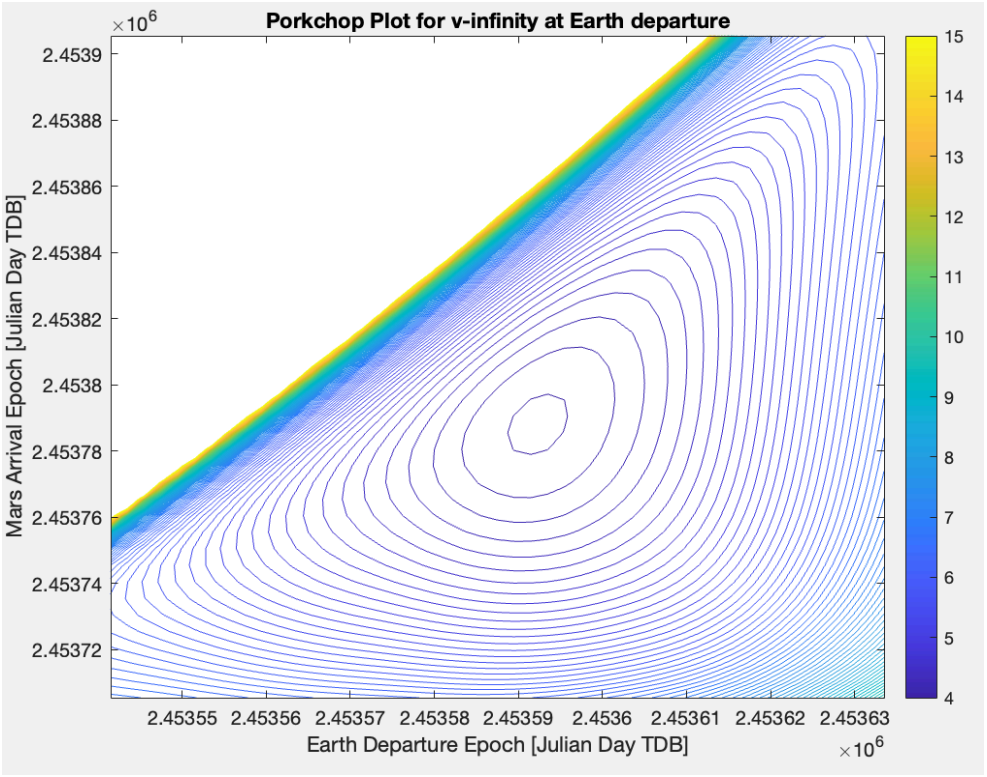
```
            if i == length(test)
                disp("Needs manual interference in delta_theta_star calc")
            end
        end

    end

    % Step 9 - find f, g to compute v1f, v2i
    f_g_fdot_gdot = fg_out(R1, R2, delta_theta_star, p, mu);
    f = f_g_fdot_gdot(1);
    g = f_g_fdot_gdot(2);
    fdot = f_g_fdot_gdot(3);
    gdot = f_g_fdot_gdot(4);

    v_1_f = (R2 - f*R1)./g;
    v_2_i = fdot*R1 + gdot*v_1_f;

    % Step 10 - v_inf for earth and mars
    v_inf_earth = norm(v_1_f - V1);
    v_inf_mars = norm(v_2_i - V2);

    out = [v_inf_earth, v_inf_mars];
end


function fx = lamberts_eqn_elliptical(a, s, c, TOF, mu, TOF_min,
greater_than_180)
    % Output function to solve for fsolve
    % Inputs:
    %       a - Semi-major axis of transfer conic [km]
    %       s - semi-perimeter of transfer triangle [km]
    %       c - chord length of transfer triangle [km]
    %       TOF - time of flight of transfer [s]
    %       mu - gravitational parameter of central body [km^3/s^2\
    %       TOF_min - TOF for minimum energy transfer [s]
    %       greater_than_180 - bool that's true if chosen arc is >180 deg
    % Output:
    %       fx - function that's dependent on a to pass to fsolve

    % Mean anomaly for transfer ellipse
    n = sqrt(mu/a^3);

    % Calculate principal alpha, beta
    alpha_0 = 2 * asin(sqrt(s/(2*a)));
    beta_0 = 2 * asin(sqrt((s-c)/(2*a)));

    % Get correct alpha, beta dependent on TOF and (choice of)
    % greater_than_180 bool
    alpha_beta = lamberts_problem_alpha_beta_logic(TOF, TOF_min,
greater_than_180, alpha_0, beta_0);
    alpha = alpha_beta(1);
    beta = alpha_beta(2);

    % Output function
```

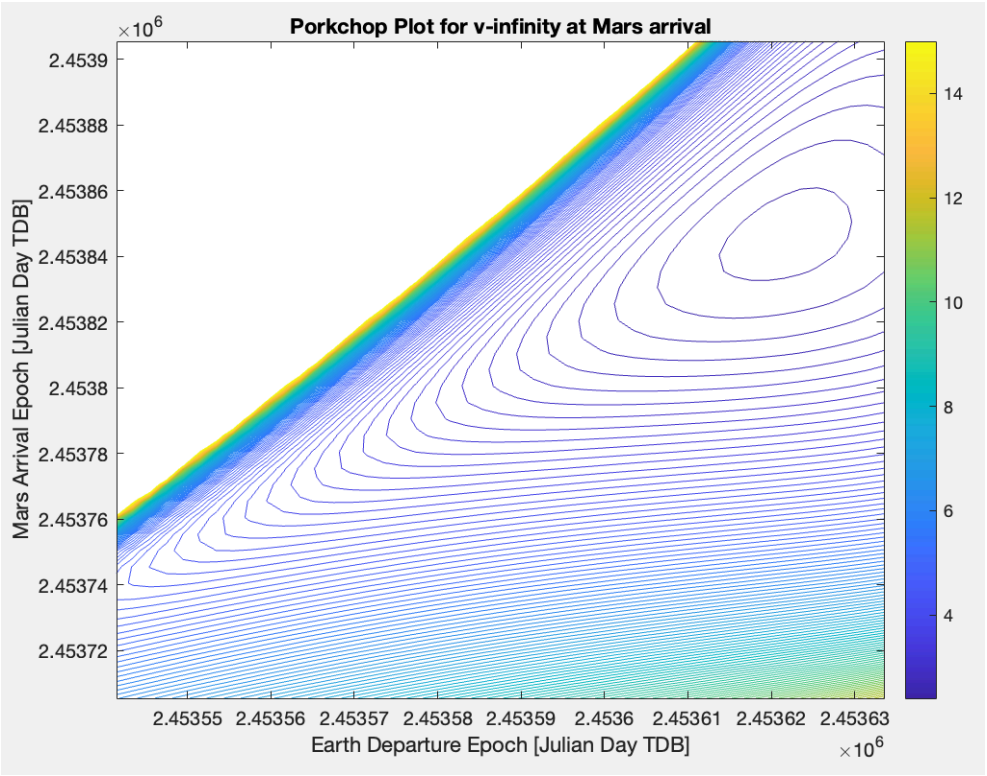
```
    fx = TOF - (alpha - beta - (sin(alpha) - sin(beta)))/n;
end

Not enough input arguments.

Error in problem2_function (line 11)
    r1 = norm(R1);
```

Published with MATLAB® R2024a

Problem 2

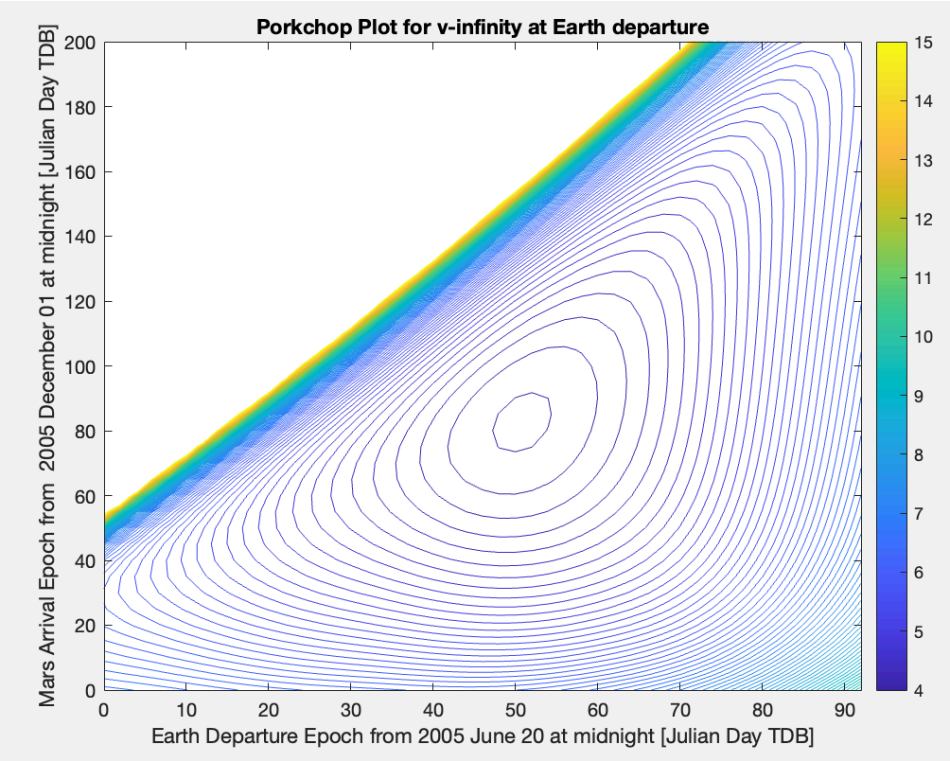
- Part b

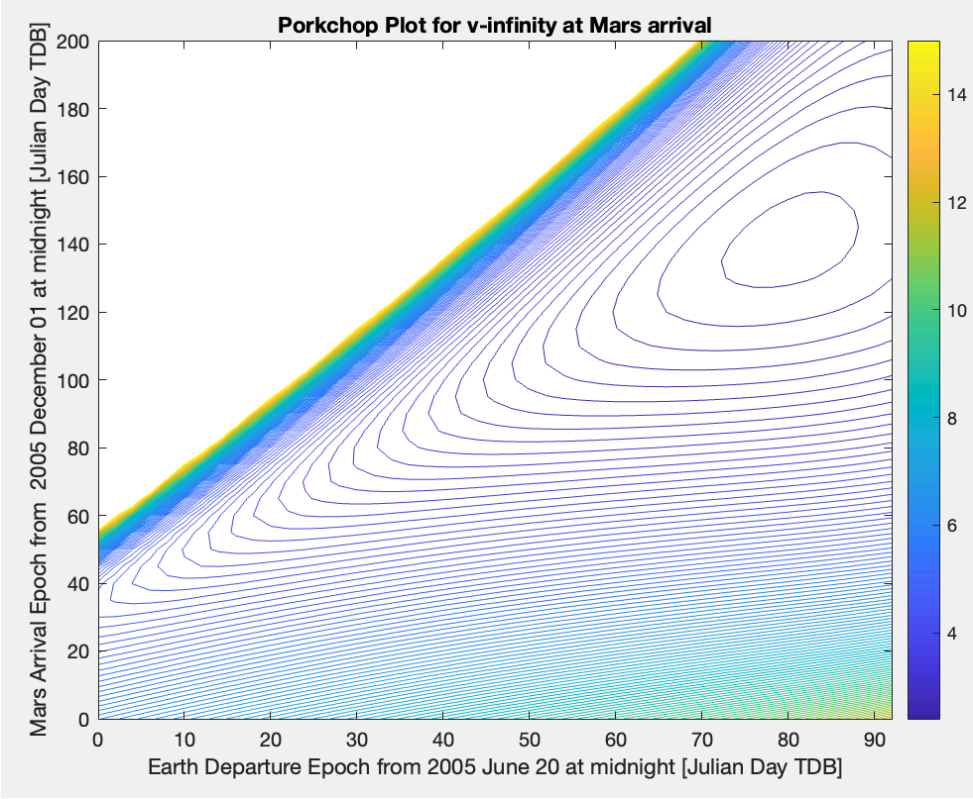




- Part c
    - The above plots tell us that in the minimum energy transfer is not (innermost circle) is not necessarily the lower TOF (fastest) transer. Also, a particular time of flight can require different v-inf based on the departure and arrival times. Since, this design space is not fully constructed (missing hyperbolic arcs and greater than 180 deg arcs), the white space indicates that certain flight times are outside the design space and cannot be achieved with the limitations posed by the problem.
    - For further analysis, greater than 180 deg arcs along with hyperbolic arcs should be added to the design space in order to expand this analysis. Other times of flight may also be considered in order to get an optimal transfer orbit.

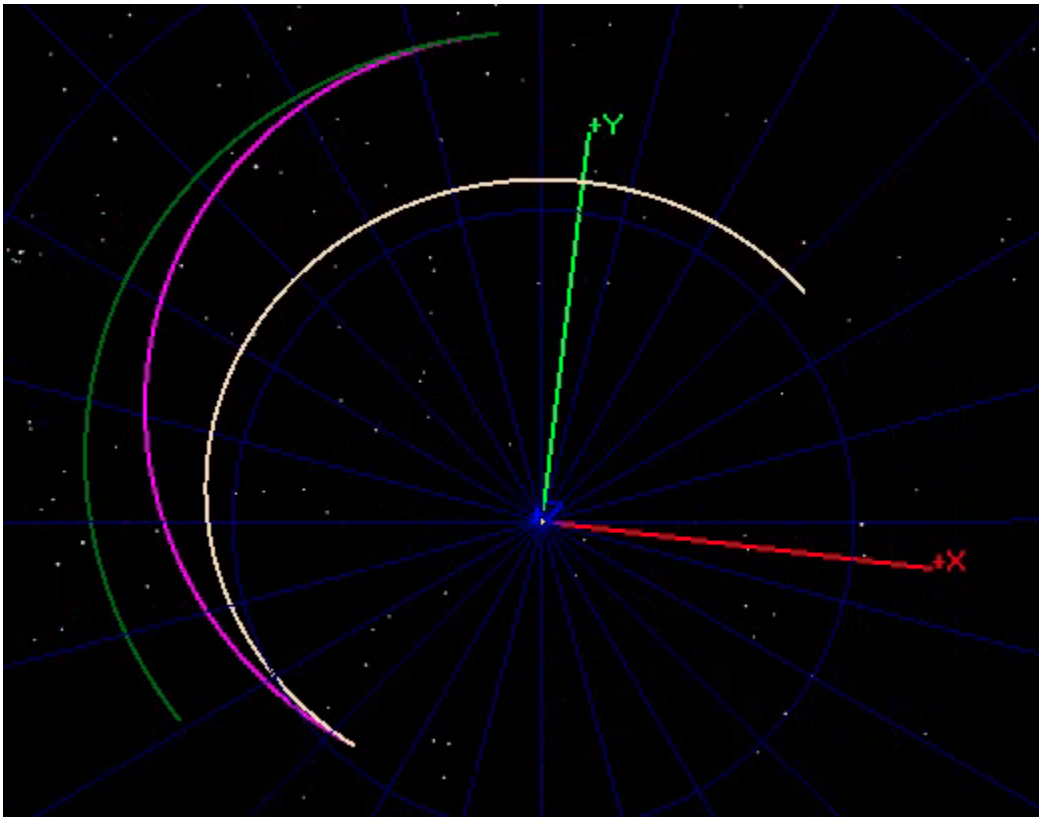
Normalizing dates in X and Y axes to make it easier to read plots:




Porkchop Plot for v-infinity at Mars arrival
Mars Arrival Epoch from 2005 December 01 at midnight [Julian Day TDB]
Earth Departure Epoch from 2005 June 20 at midnight [Julian Day TDB]
0
20
40
60
80
100
120
140
160
180
200
0
10
20
30
40
50
60
70
80
90
4
6
8
10
12
14

Problem 3

- Part a
    - The direction of motion of the spacecraft (magenta) and earth and venus is counterclockwise as seen in the image below where the sun is at the origin of the axes and Earth's orbit is in green (outermost) and Venus' orbit is in yellow (innermost)

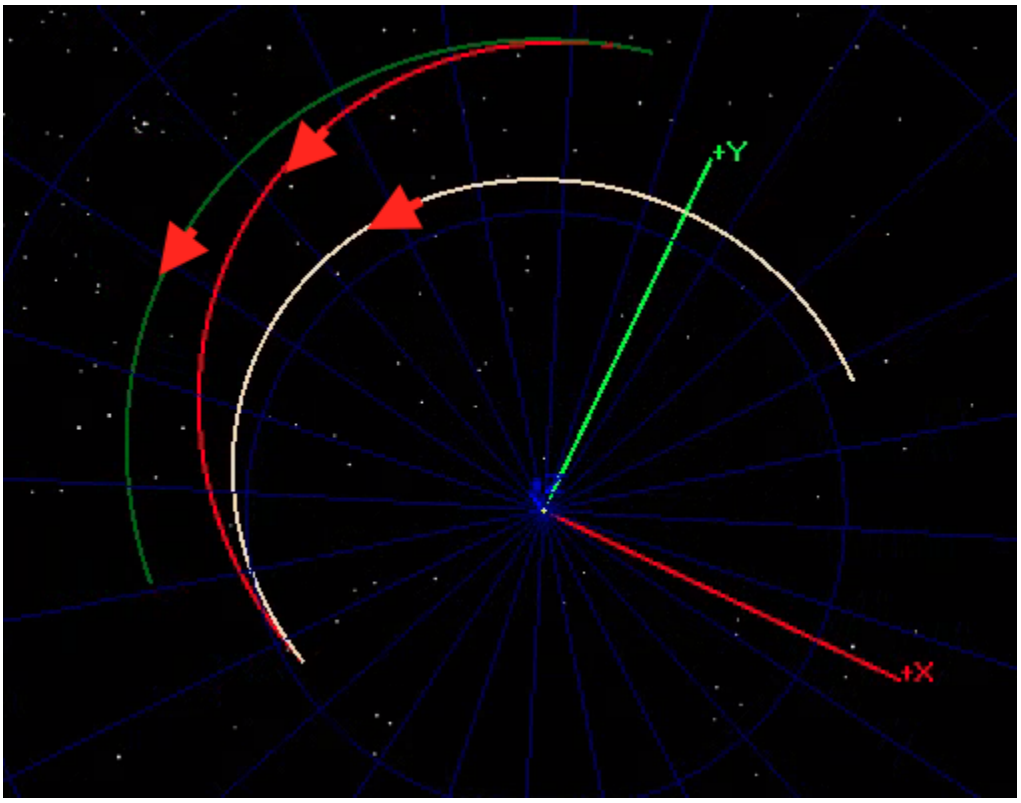


- Part b
    - The image below shows the output of the spacecraft after the second maneuver has been applied:

```
Cartesian State
---------------------------
X  =  -56429572.265292 km
Y  =  -85710788.980482 km
Z  =  -34994818.527999 km
VX =   29.658111011629 km/sec
VY =  -16.091170133921 km/sec
VZ =  -9.1167170173731 km/sec
```

    - The position and velocity vectors are very similar to the R2 and V2 vectors provided in the problem statement.

- Part c
    - The direction of motion of the spacecraft (red) and earth and venus is counterclockwise as seen in the image where the sun is at the origin of the axes and Earth's orbit is in green (outermost) and Venus' orbit is in yellow (innermost)




```
Cartesian State
-------------------------
X  =  -56429010.000477 km
Y  =  -85710479.999748 km
Z  =  -34994659.999863 km
VX =   29.658340999962 km/sec
VY =  -16.091100000116 km/sec
VZ =  -9.1166740000396 km/sec
```

    - The image above shows the position and velocity vectors of the spacecraft after the second maneuver has been applied. This state vector lies within the specified tolerance of the state vector as specified by the vary and achieve methods in GMAT.

```
Maneuver Summary
----------------
Impulsive Burn:     DV1
Spacecraft:         DefaultSC
Coordinate System:  SunInertial
Delta V Vector:
   Element 1:    2.8866439676735 km/s
   Element 2:    0.1654443249558 km/s
   Element 3:    0.8876338778209 km/s

No mass depletion
```

```
Maneuver Summary
----------------
Impulsive Burn:     DV2
Spacecraft:         DefaultSC
Coordinate System:  SunInertial
Delta V Vector:
   Element 1:   -3.9591769800198 km/s
   Element 2:   -1.5562398305259 km/s
   Element 3:   -1.9170801084901 km/s

No mass depletion
```

- The images above show the delta v vectors for the two maneuvers. The magnitude of delta V1 is 3.0246 km/s and magnitude of delta V2 is 4.6661 km/s. Comparing these to the two norms calculated in Problem 1 yields the result that both of these are almost identical.

```
' Targeting Completed in 7 iterations.
' The Targeter converged!
```

- Lastly, the above image shows the number of iterations it took for GMAT to calculate the solution - 7.