RS-WISP-04-01: Two Impuse Rendezvous

WorkerInSpace

Hongseok Kim

2/5/2025

I. Scope

In RS-WISP-02, we have demonstrated the C-W equation for Two-Impuse Rendezvous. For defining operation modes, we have to define multiple functions for these delta-v maneuvers. Furthermore, we should structurize the process of rendezvous process in multiple block diagrams. In this document, the role of Two-Impuse Rendezvous and I/O structure is defined, and corresponding function is described. From these function, we can proceed the simulation in multiple environments and multiple simulation setups.

II. Two Impulse Rendezvous Procedure

II.1 I/O Structure

Input Parameter

– **Position and velocity of Taget Satellite** : $r_{\text{target}} = \boldsymbol{r}_0$, $v_{\text{target}} = \boldsymbol{v}_0$

– **Kepler parameter of Chaser Satellite** : $\boldsymbol{r}_{\text{chaser}} = \boldsymbol{r}, \boldsymbol{v}_{\text{chaser}} = \boldsymbol{v}$

– **Required Elapsed Time** t_f

(note : Target Satellite shall maintain circular orbit)

Output Parameter

– **initial** $[\Delta \boldsymbol{v}]_{\mathbf{LVLH}}$ **and final** $[\Delta \boldsymbol{v}]_{\mathbf{LVLH}}$ with respect to LVLH frame

– $\delta \boldsymbol{r}(\boldsymbol{t})$ **and** $\delta \boldsymbol{v}(\boldsymbol{t})$ **data** with respect to LVLH frame

– **initial** $\boldsymbol{v}_{\mathbf{ECI}}^{+}$ with respect to ECI frame for orbit propagation

(**Note : the actual path should be drawn in ECI frame for cross check**)

II.2 Two Impulse Rendezvous Algorithm

1. Transformation matrix from ECI to Target LVLH frame

$$\hat{\mathbf{i}} = \frac{\boldsymbol{r}_0}{|\boldsymbol{r}_0|}, \hat{\mathbf{k}} = \frac{\boldsymbol{r}_0 \times \boldsymbol{v}_0}{|\boldsymbol{r}_0 \times \boldsymbol{v}_0|}, \hat{\mathbf{j}} = \hat{\mathbf{k}} \times \hat{\mathbf{i}} \Rightarrow [Q]_{\text{LVLH,ECI}} = \begin{bmatrix} - & [\hat{\mathbf{i}}]^T & - \\ - & [\hat{\mathbf{j}}]^T & - \\ - & [\hat{\mathbf{k}}]^T & - \end{bmatrix}$$

2. Calculate mean motion of Target and corresponding angular veocity

mean motion of Target $n = \dfrac{v_0}{r_0}$

$\Omega_{\text{Target}} = n\hat{\mathbf{k}}$

3. Calculate initial relative position and velocity vector of the Chaser wrt ECI frame

$\boldsymbol{r}_{\text{rel}} = \delta \boldsymbol{r}, \boldsymbol{v}_{\text{rel}} = \delta \boldsymbol{v}$

$\boldsymbol{v} = \boldsymbol{v}_0 + \Omega_{\text{Target}} \times \boldsymbol{r}_{\text{rel}} + \boldsymbol{v}_{\text{rel}}$

$\delta \boldsymbol{r} = \boldsymbol{r} - \boldsymbol{r}_0$

$\delta \boldsymbol{v} = \boldsymbol{v} - \boldsymbol{v}_0 - \Omega_{\text{Target}} \times \delta \boldsymbol{r}$

4. Calculate initial reletive position and velocity vector of the Chaser wrt Target LVLH frame

$[\delta \boldsymbol{r}_0] = [Q]_{\text{LVLH,ECI}}\, \delta \boldsymbol{r}$

$\left[\delta \boldsymbol{v}_0^-\right] = [\boldsymbol{Q}]_{\text{LVLH,ECI}}\, \delta \boldsymbol{v}$

5. Generate Clohessy – Wiltshire matrices function for given mean motion

$$[\Phi_{\text{rr}}(t)] = \begin{bmatrix} 4 - 3\cos nt & 0 & 0 \\ 6(\sin nt - nt) & 1 & 0 \\ 0 & 0 & \cos nt \end{bmatrix} \quad [\Phi_{\text{rv}}(t)] = \begin{bmatrix} \dfrac{\sin nt}{n} & \dfrac{2}{n}(1 - \cos nt) & 0 \\ \dfrac{2}{n}(\cos nt - 1) & \dfrac{4}{n}\sin nt - 3t & 0 \\ 0 & 0 & \dfrac{1}{n}\sin nt \end{bmatrix}$$

$$[\Phi_{\text{vr}}(t)] = \begin{bmatrix} 3n \sin nt & 0 & 0 \\ 6n(\cos nt - 1) & 0 & 0 \\ 0 & 0 & -n \sin nt \end{bmatrix} \quad [\Phi_{\text{vv}}(t)] = \begin{bmatrix} \cos nt & 2 \sin nt & 0 \\ -2 \sin nt & 4\cos nt - 3 & 0 \\ 0 & 0 & \cos nt \end{bmatrix}$$

6. From given t_f, find $\left[\delta \boldsymbol{v}_0^+\right]$ and corresponding $[\Delta \boldsymbol{v}_0]$ wrt LVLH frame

$\left[\delta \boldsymbol{v}_0^+\right] = -[\Phi_{\text{rv}}(t_f)]^{-1}[\Phi_{\text{rr}}(t_f)][\delta \boldsymbol{r}_0]$

$[\Delta \boldsymbol{v}_0]_{\text{LVLH}} = \left[\delta \boldsymbol{v}_0^+\right] - \left[\delta \boldsymbol{v}_0^-\right]$

7. At time t_f, find $\left[\delta \boldsymbol{v}_f^-\right]$ and corresponding $[\Delta \boldsymbol{v}_f]$ wrt LVLH frame

$\left[\delta \boldsymbol{v}_f^-\right] = [\Phi_{\text{vr}}(t_f)][\delta \boldsymbol{r}_0] + [\Phi_{\text{vv}}(t_f)]\left[\delta \boldsymbol{v}_0^+\right]$

$[\Delta \boldsymbol{v}_f]_{\text{LVLH}} = -\left[\delta \boldsymbol{v}_f^-\right]$

8. From $t = 0 \sim t_f$, we can calculate $[\delta \boldsymbol{r}]_{\text{LVLH}}$ and $[\delta \boldsymbol{v}]_{\text{LVLH}}$ from given equation

$[\delta \boldsymbol{r}(t)] = [\Phi_{\text{rr}}(t)][\delta \boldsymbol{r}_0] + [\Phi_{\text{rv}}(t)][\delta \boldsymbol{v}_0]$

$[\delta \boldsymbol{v}(t)] = [\Phi_{\text{vr}}(t)][\delta \boldsymbol{r}_0] + [\Phi_{\text{vv}}(t)][\delta \boldsymbol{v}_0]$

9. From LVLH to ECI coordinate transfer, we can calculate $\boldsymbol{v}_{\text{ECI}}^+$ from $\left[\delta \boldsymbol{v}_0^+\right]$

from $\boldsymbol{v} = \boldsymbol{v}_0 + \Omega_{\text{Target}} \times \boldsymbol{r}_{\text{rel}} + \boldsymbol{v}_{\text{rel}}$

$\boldsymbol{v}_{\text{ECI}}^+ = \boldsymbol{v}_0 + \Omega_{\text{Target}} \times \delta \boldsymbol{r} + [\boldsymbol{Q}]_{\text{LVLH,ECI}}^{-1}\left[\delta \boldsymbol{v}_0^+\right]$

(note : $\boldsymbol{v}_0$ = Target's veclotiy wrt ECI, $\left[\delta \boldsymbol{v}_0^+\right]$ = Chaser's relative velocity wrt target's LVLH)

III. Matlab function demonstration

```
function [Delta_v_0, Delta_v_f, delta_r_t_mat, delta_v_t_mat, v_plus_ECI] =
two_impulse_rendezvous(r_target, v_target, r_chaser, v_chaser, tf)

r_0 = r_target;
v_0 = v_target;
r = r_chaser;
v = v_chaser;

% Reference Frame of the Target
i_hat = r_0/norm(r_0);
j_hat = v_0/norm(v_0);
k_hat = cross(i_hat, j_hat);

% Transformation Matrix from ECI to Space Station Frame
Q_LVLH_ECI = [i_hat';j_hat';k_hat'];

% Position vector of the spacecraft relative to the space station (ECI)
delta_r = r - r_0;
n = norm(v_0)/norm(r_0);
Omega_target = n * k_hat;
delta_v = v - v_0 - cross(Omega_target, delta_r);

% Relative position vector at the beginning of the rendezvous maneuver
delta_r_0 = Q_LVLH_ECI * delta_r;

% Relative velocity just before launch into the rendezvous trajectory is
delta_v_0_minus = Q_LVLH_ECI * delta_v;

% Calculate Clohessy-Whiltshire matrix for t = t_f = 28800s and n
nt = n*tf;

Phi_rr_tf = [4 - 3*cos(nt),   0, 0;
             6*(sin(nt) - nt),1,0;
             0,0, cos(nt)];

Phi_rv_tf = [sin(nt)/n, 2/n * (1 - cos(nt)), 0;
          2/n*(cos(nt)-1), 4/n*sin(nt)-3*tf,0;
          0,0,1/n*sin(nt)];

Phi_vr_tf = [3*n*sin(nt),0,0;
          6*n*(cos(nt)-1),0,0;
          0,0,-n*sin(nt)];
Phi_vv_tf = [cos(nt), 2*sin(nt),0
         -2*sin(nt), 4*cos(nt)-3, 0;
          0,0,cos(nt)];

delta_v_0_plus = -inv(Phi_rv_tf) * Phi_rr_tf * delta_r_0;
delta_v_f_minus = Phi_vr_tf * delta_r_0 + Phi_vv_tf * delta_v_0_plus;

% Delta-v at the beginning of the rendezvous maneuver
```

```
Delta_v_0 = delta_v_0_plus - delta_v_0_minus;
% Delta-v at the conclusion of the maneuver
Delta_v_f = [0;0;0] - delta_v_f_minus;



% Drawing Reference Chaser Trajector wrt Target
t_vector = linspace(0,tf);
delta_r_t_mat = zeros(length(t_vector),3);
delta_v_t_mat = zeros(length(t_vector),3);

for timestep = 1:length(t_vector)

    t = t_vector(timestep);

    nt = n*t;

    Phi_rr_t = [4 - 3*cos(nt),   0, 0;
                6*(sin(nt) - nt),1,0;
                0,0, cos(nt)];

    Phi_rv_t = [sin(nt)/n, 2/n * (1 - cos(nt)), 0;
                2/n*(cos(nt)-1), 4/n*sin(nt)-3*t,0;
                0,0,1/n*sin(nt)];

    Phi_vr_t = [3*n*sin(nt),0,0;
                6*n*(cos(nt)-1),0,0;
                0,0,-n*sin(nt)];
    Phi_vv_t = [cos(nt), 2*sin(nt),0
               -2*sin(nt), 4*cos(nt)-3, 0;
                0,0,cos(nt)];

    delta_r_t = Phi_rr_t * delta_r_0 + Phi_rv_t * delta_v_0_plus;
    delta_v_t = Phi_vr_t * delta_r_0 +  Phi_vv_t * delta_v_0_plus;

    delta_r_t_mat(timestep,:) = delta_r_t';
    delta_v_t_mat(timestep,:) = delta_v_t';
end

v_plus_ECI = v_0 + cross(Omega_target, delta_r) + delta_v_0_plus\Q_LVLH_ECI;

end
```

IV. Matlab Function Test

```
%% Input Parameters

% Target Information
Target.a = 300+6378;
Target.e = 1e-5;
```

```
Target.i = 40;
Target.RAAN = 20;
Target.AOP = 0;
Target.TA = 60;


% Chaser Information
Chaser.a = 6378 + (318.5 + 515.51)/2;
Chaser.e = (6378+515.51)/Chaser.a-1;
Chaser.i = 40.130;
Chaser.RAAN = 19.819;
Chaser.AOP = 70.662;
Chaser.TA = 349.65;

% Timing of Docking
t_f = 28800;

%% Main Algorithm

% Change to ECI frame

% rv data of the target wrt ECI
[r_target, v_target] = kepler2ijk_hs(Target.a, Target.e, Target.i, Target.RAAN, Target.AOP,
Target.TA);

% rc data of the chaser wrt ECI
[r_chaser, v_chaser] = kepler2ijk_hs(Chaser.a, Chaser.e, Chaser.i, Chaser.RAAN, Chaser.AOP,
Chaser.TA);

[Delta_v_0, Delta_v_f, delta_r_t, delta_v_t, v_plus_ECI] = two_impulse_rendezvous(r_target,
v_target, r_chaser, v_chaser, t_f)
```

```
Delta_v_0 = 3×1
    0.0294
   -0.0667
    0.0130
Delta_v_f = 3×1
    0.0258
    0.0005
    0.0244
delta_r_t = 100×3
   20.0303   20.2865   19.9531
   21.5214    6.0269   21.1097
   20.5068   -8.3946   19.8978
   17.1003  -21.3140   16.4532
   11.6842  -31.2357   11.1624
    4.8661  -37.0005    4.6191
   -2.5889  -37.9157   -2.4425
   -9.8442  -33.8328   -9.2300
  -16.0859  -25.1640  -14.9818
  -20.6135  -12.8361  -19.0526
```

```
       ⋮
delta_v_t = 100×3
    0.0093   -0.0468    0.0080
    0.0008   -0.0503   -0.0001
   -0.0077   -0.0479   -0.0082
   -0.0155   -0.0400   -0.0153
   -0.0214   -0.0275   -0.0207
   -0.0250   -0.0117   -0.0238
   -0.0258    0.0055   -0.0243
   -0.0236    0.0223   -0.0220
   -0.0189    0.0368   -0.0172
   -0.0120    0.0472   -0.0105
       ⋮
v_plus_ECI = 3×3
   13.2732   -7.4962   -8.9281
   21.0754    0.3060   -1.1259
   23.0776    2.3083    0.8764
```

Appendix 1: Bundle functions (kepler2ijk, ijk2kepler)

```
function [r,v] = kepler2ijk_hs(a,e,inc,RAAN,AOP,theta)

mu = 398600.4415;

RAAN = RAAN/180*pi;
inc = inc/180*pi;
AOP = AOP/180*pi;
theta = theta/180*pi;


R_3_AOP = [cos(AOP), sin(AOP),0;
          -sin(AOP), cos(AOP),0;
                 0,        0,1];

R_1_inc = [1,       0,         0;
          0, cos(inc),  sin(inc);
          0,-sin(inc),  cos(inc)];

R_3_RAAN = [cos(RAAN), sin(RAAN),0;
           -sin(RAAN), cos(RAAN),0;
                  0,        0,1];



Q_rotation = R_3_AOP * R_1_inc * R_3_RAAN;

p = Q_rotation(1,:);
q = Q_rotation(2,:);
w = Q_rotation(3,:);
```

```
p_value = a*(1-e^2);
r = p_value* (cos(theta)*p + sin(theta) *q)/(1+e*cos(theta));
r = r';
v = sqrt(mu/p_value)*(-sin(theta)*p+(e+cos(theta))*q)';
end


function [a, e, inc, RAAN, AOP, theta] = ijk2kepler_hs(r,v)

mu = 398600.4415;

h = cross(r,v);                                   % Angular Velocity Vector
z = [0,0,1]';
n = (cross(z,h))/(norm(cross(z,h)));              % Normal Vector
e_vector = ((cross(v,h))/mu) - r/(norm(r));       % Eccentricity vector


a = (-mu)/((norm(v))^2 - 2*mu/norm(r));           % 1. Semi-major axis
e = norm(e_vector);                               % 2. Eccentricity
inc = acos(dot(z,h)/norm(h)) / pi *180;           % 3. Inclination

RAAN = atan2(n(2),n(1));                          % 4. RAAN
RAAN = RAAN / pi * 180;

AOP = acos((dot(n,e_vector))/norm(e_vector));                  % 5. Argument of periapse
if(e_vector(3)<0)
    AOP = 2*pi - AOP;
end
AOP = AOP/pi*180;

theta = acos((dot(r,e_vector))/(norm(r)*norm(e_vector)));      % 6. True anomaly
if(dot(r,v) < 0)
    theta = 2*pi-theta;
end
theta = theta / pi * 180;

end
```