John Clouse IMD HW5

Table of Contents

Initialize

```
clearvars -except hw_pub function_list

rSOI=[546507.344255845;
    527978.380486028;
    531109.066836708];

vSOI=[4.9220589268733;
    5.36316523097915;
    5.22166308425181];

mu = 398600.4415;
```

Problem 1: Compute the effects of perturbations on the velocity to the B Plane

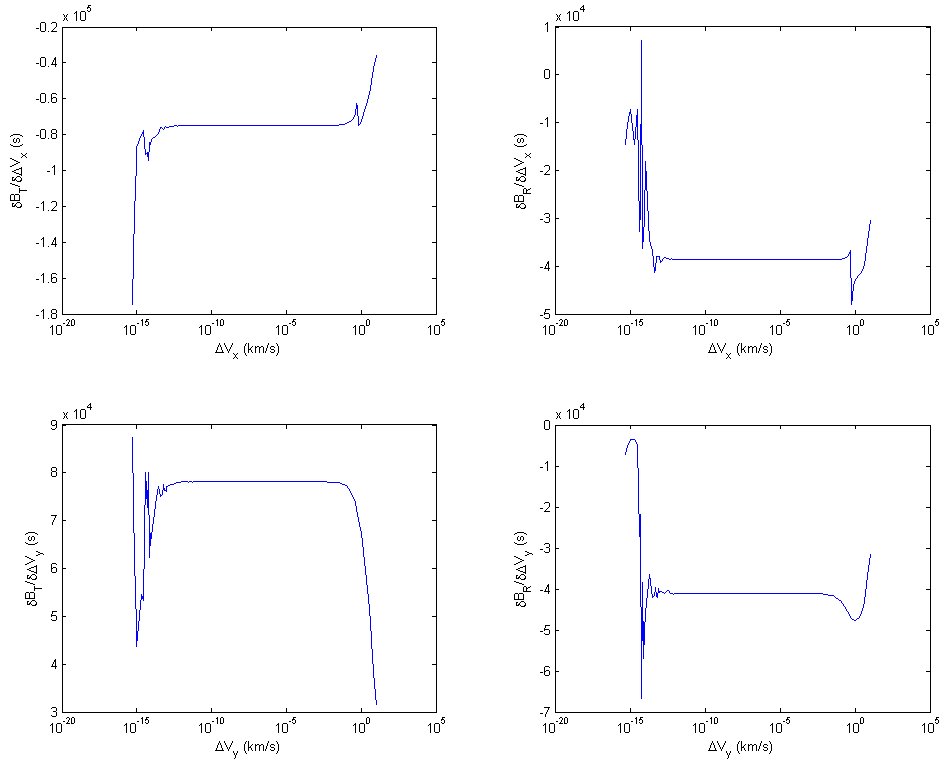
```
% The B-Plane as-is
[ b, B_hat, B_plane ] = Bplane_rv( rSOI, vSOI, mu );

BT = dot(b*B_hat, B_plane(:,2));
BR = dot(b*B_hat, B_plane(:,3));

% Set up the velocities and storage
dv_range = 10:-1:1;
blah = 1e-1;
% blah = 1e-4;
% dv_range = 9*blah;
while min(dv_range) > 5e-16
    dv_range = [dv_range (9:-1:1)*blah];
    blah = blah*1e-1;
end

dv_range = dv_range(dv_range > 5e-16);

dBT_dvx = zeros(1,length(dv_range));
```

```
dBT_dvy = zeros(1,length(dv_range));
dBR_dvx = zeros(1,length(dv_range));
dBR_dvy = zeros(1,length(dv_range));

% Iterate on the velocity perturbations
for ii = 1:length(dv_range)
    % X first
    [ b_x, B_hat_x, B_plane_x ] = ...
        Bplane_rv( rSOI, vSOI + [dv_range(ii);0;0], mu );

    BT_x = dot(b_x*B_hat_x, B_plane_x(:,2));
    BR_x = dot(b_x*B_hat_x, B_plane_x(:,3));

    dBT_dvx(ii) = (BT_x - BT)/dv_range(ii);
    dBR_dvx(ii) = (BR_x - BR)/dv_range(ii);

    % Y
    [ b_y, B_hat_y, B_plane_y ] = ...
        Bplane_rv( rSOI, vSOI + [0;dv_range(ii);0], mu );

    BT_y = dot(b_y*B_hat_y, B_plane_y(:,2));
    BR_y = dot(b_y*B_hat_y, B_plane_y(:,3));

    dBT_dvy(ii) = (BT_y - BT)/dv_range(ii);
    dBR_dvy(ii) = (BR_y - BR)/dv_range(ii);
end
```

Problem 1 conclusion

Numerical errors are pretty significant toward smaller perturbations, due to the very small dV being in the denominator. The nonlinearities are noticable both in very small and very large perturbations. I would say the linear cutoff point is in the tens of meters per second. My later computations will use 1 m/s as the velocity at which to calculate the partials.

```
figure('Position', hw_pub.figPosn)
subplot(2,2,1)
% loglog(dv_range,dBT_dvx)
semilogx(dv_range,dBT_dvx)
ylabel('\deltaB_T/\delta\DeltaV_x (s)')
xlabel('\DeltaV_x (km/s)')
subplot(2,2,2)
semilogx(dv_range,dBR_dvx)
ylabel('\deltaB_R/\delta\DeltaV_x (s)')
xlabel('\DeltaV_x (km/s)')
subplot(2,2,3)
semilogx(dv_range,dBT_dvy)
ylabel('\deltaB_T/\delta\DeltaV_y (s)')
xlabel('\DeltaV_y (km/s)')
subplot(2,2,4)
semilogx(dv_range,dBR_dvy)
ylabel('\deltaB_R/\delta\DeltaV_y (s)')
xlabel('\DeltaV_y (km/s)')
```



Problem 2: Find the dV needed to hit the desired target

```
BT_target = 13135.7982982557; %km
BR_target = 5022.26511510685; %km
vel = vSOI;

% Numerically calculate at dV = 1 m/s
calc_vel = 1e-3; %km/s

tol = 1e-10;

BT_curr = BT;
BR_curr = BR;
d_BT = BT_target - BT_curr;
d_BR = BR_target - BR_curr;

% Keep iterating until we hit the tolerance
while abs(d_BT) > tol && abs(d_BR) > tol
    dB = [d_BT; d_BR];
    % X first
    [ b_x, B_hat_x, B_plane_x ] = ...
        Bplane_rv( rSOI, vel + [calc_vel;0;0], mu );
```

```
    BT_x = dot(b_x*B_hat_x, B_plane_x(:,2));
    BR_x = dot(b_x*B_hat_x, B_plane_x(:,3));

    dBT_dvx = (BT_x - BT_curr)/calc_vel;
    dBR_dvx = (BR_x - BR_curr)/calc_vel;

    % Y
    [ b_y, B_hat_y, B_plane_y ] = ...
        Bplane_rv( rSOI, vel + [0;calc_vel;0], mu );

    BT_y = dot(b_y*B_hat_y, B_plane_y(:,2));
    BR_y = dot(b_y*B_hat_y, B_plane_y(:,3));

    dBT_dvy = (BT_y - BT_curr)/calc_vel;
    dBR_dvy = (BR_y - BR_curr)/calc_vel;

    dV = [dBT_dvx dBT_dvy; dBR_dvx dBR_dvy]\dB;

    vel = vel+[dV;0];

    % Calculate resultant b-plane target
    [ b, B_hat, B_plane ] = ...
        Bplane_rv( rSOI, vel, mu );
    BT_curr = dot(b*B_hat, B_plane(:,2));
    BR_curr = dot(b*B_hat, B_plane(:,3));
    d_BT = BT_target - BT_curr;
    d_BR = BR_target - BR_curr;
end
```

Problem 2 conclusion

```
fprintf('New velocity vector:\n');
disp(vel)
fprintf('\b\b km/s\n')
fprintf('\n')
fprintf('dV:\n')
disp(vel-vSOI)
fprintf('\b\b km/s\n')
```

```
New velocity vector:
   5.286077698289384
   5.298100544621716
   5.221663084251810 km/s

dV:
   0.364018771416085
  -0.065064686357434
                   0 km/s
```

Published with MATLAB® R2013b

```
function [ b, B_hat, B_plane ] = Bplane_rv( r, v, mu )
%Bplane_rv Compute B Plane from r, v, mu wrt target planet
%   Detailed explanation goes here
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

r_mag = norm(r);
v_mag = norm(v);

energy = v_mag*v_mag/2 - mu/r_mag;

v_inf_2 = energy*2;

h_hat = cross(r,v)/norm(cross(r,v));

e_vec = ((v_mag*v_mag-mu/r_mag)*r-dot(r,v)*v)/mu;
a = -mu/v_inf_2;
e = norm(e_vec);

b = -a*sqrt(e*e-1);

phi_s = acos(1/e);
k = [0;0;1];

S_hat = e_vec/e*cos(phi_s) ...
    + cross(h_hat,e_vec)/norm(cross(h_hat,e_vec))*sin(phi_s);
T_hat = cross(S_hat,k)/norm(cross(S_hat,k));
R_hat = cross(S_hat, T_hat);

B_plane = [S_hat T_hat R_hat];

B_hat = cross(S_hat,h_hat);
end
```

Published with MATLAB® R2013b

```
function fcnPrintQueue( filename )
global function_list;
if exist('function_list', 'var')
    file_in_list = 0;
    for idx = 1:length(function_list)
        if strcmp(function_list(idx), filename);
            file_in_list = 1;
            break
        end
    end
    if ~file_in_list
        function_list = [function_list, filename];
    end
end
end
```

Published with MATLAB® R2013b