IMD HW 1, Problem 1

Table of Contents

John Clouse

Initialize

```
clearvars -except hw_pub function_list

h_i = 400; %km
h_f = 400; %km

mu_sun = 1.327e11; %km3/s2
mu_earth = 3.986e5; %km3/s2
mu_mars = 4.305e4; %km3/s2

km_per_AU = 1.4959787e8; %km
a_earth = 1*km_per_AU; %km
a_mars = 1.52367934*km_per_AU; %km
r_earth = 6378.1363; %km
r_mars = 3397.2; %km
```

Calculate transfer orbit, velocities

```
a_xfer = (a_earth + a_mars)/2;
v_earth = sqrt(mu_sun/a_earth);
v_mars = sqrt(mu_sun/a_mars);

v_xfer_i = sqrt(2*mu_sun/a_earth - mu_sun/a_xfer);
v_xfer_f = sqrt(2*mu_sun/a_mars - mu_sun/a_xfer);
fprintf('Departure velocity: %.3f km/s\n',v_xfer_i);
fprintf('Arrival velocity: %.3f km/s\n',v_xfer_f);

v_esc_earth = sqrt(2*mu_earth/(r_earth + h_i));
v_park_earth = sqrt(mu_earth/(r_earth + h_i));

v_esc_mars = sqrt(2*mu_mars/(r_mars + h_f));
v_park_mars = sqrt(mu_mars/(r_mars + h_f));

v_inf_dep = v_xfer_i-v_earth;
v_inf_arr = v_mars-v_xfer_f;
v_hyp_dep = sqrt(2*mu_earth/(r_earth + h_i) + v_inf_dep^2);
v_hyp_arr = sqrt(2*mu_mars/(r_mars + h_f) + v_inf_arr^2);
dv_departure = v_hyp_dep - v_park_earth;
dv_arrival = v_hyp_arr - v_park_mars;
```

```
fprintf('Departure dV: %.3f km/s\n',dv_departure);
fprintf('Arrival dV: %.3f km/s\n',dv_arrival);
fprintf('\t Initial velocity = %.3f km/s\n', v_park_earth);
fprintf('\t Hyperbolic departure velocity = %.3f km/s\n', v_hyp_dep);
fprintf('\t Departure dV = %.3f km/s\n', v_hyp_dep-v_park_earth);
fprintf('\t Hyperbolic arrival velocity = %.3f km/s\n', v_hyp_arr);
fprintf('\t Final Mars velocity = %.3f km/s\n', v_park_mars);
fprintf('\t Arrival dV = %.3f km/s\n', v_hyp_arr-v_park_mars);
T = pi*sqrt(a_xfer^3/mu_sun)/3600/24;
fprintf('Transfer time: %.3f days\n', T);
```

```
Departure velocity: 32.728 km/s
Arrival velocity: 21.479 km/s
Departure dV: 3.569 km/s
Arrival dV: 2.082 km/s
  Initial velocity = 7.669 km/s
  Hyperbolic departure velocity = 11.238 km/s
  Departure dV = 3.569 km/s
  Hyperbolic arrival velocity = 5.449 km/s
  Final Mars velocity = 3.367 km/s
  Arrival dV = 2.082 km/s
Transfer time: 258.878 days
```

Published with MATLAB® R2013b

IMD HW 1 Problem 2

Table of Contents

John Clouse

Initialization

```
clearvars -except T v_xfer_i hw_pub function_list
close all
lineWidth = 2;

mu_sun = 1.327e11; %km3/s2
mu_earth = 3.986e5; %km3/s2
mu_mars = 4.305e4; %km3/s2

km_per_AU = 1.4959787e8; %km
a_earth = 1*km_per_AU; %km
a_mars = 1.52367934*km_per_AU; %km
% r_earth = 6378.1363; %km
% r_mars = 3397.2; %km

r_earth = [-578441.002878924;
    -149596751.684464;
    0]; % km
v_earth = [29.7830732658560;
    -0.115161262358529;
    0]; % km/s

r_mars_f = [-578441.618274359;
    227938449.869731;
    0];

v_mars_f = [-24.1281802482527;
    -0.0612303173808154;
    0];

r_sat = [0;-km_per_AU;0];
v_sat = [v_xfer_i;0;0];

% Compute the initial location of Mars.
% Since we assume Mars is in circ. orbit and coplanar, just rotate the
% vector back.
% Find mean motion of Mars
% Rotate Mars final r,v by -n*T
```

```
n_mars = sqrt(mu_sun/a_mars^3);
r_mars_i = Euler2DCM('3',n_mars*T*3600*24)*r_mars_f;
v_mars_i = Euler2DCM('3',n_mars*T*3600*24)*v_mars_f;
```

Propagate with just sun accel

```
prop_opts.mu = mu_sun;
times = linspace(0, T*3600*24, 3000);

ode_opts = odeset('RelTol', 1e-12, 'AbsTol', 1e-16);
ode_opts = odeset('RelTol', 1e-3, 'AbsTol', 1e-6);
[T_out, X_out] = ode45(@two_body_state_dot, times, [r_sat;v_sat], ode_opts, prop_o
X_x_store = X_out(:,1);
X_y_store = X_out(:,2);
X_vx_store = X_out(:,4);
X_vy_store = X_out(:,5);
```

Propagate with Earth and Mars perturbations.

Need to track the states of earth and mars for each integration step

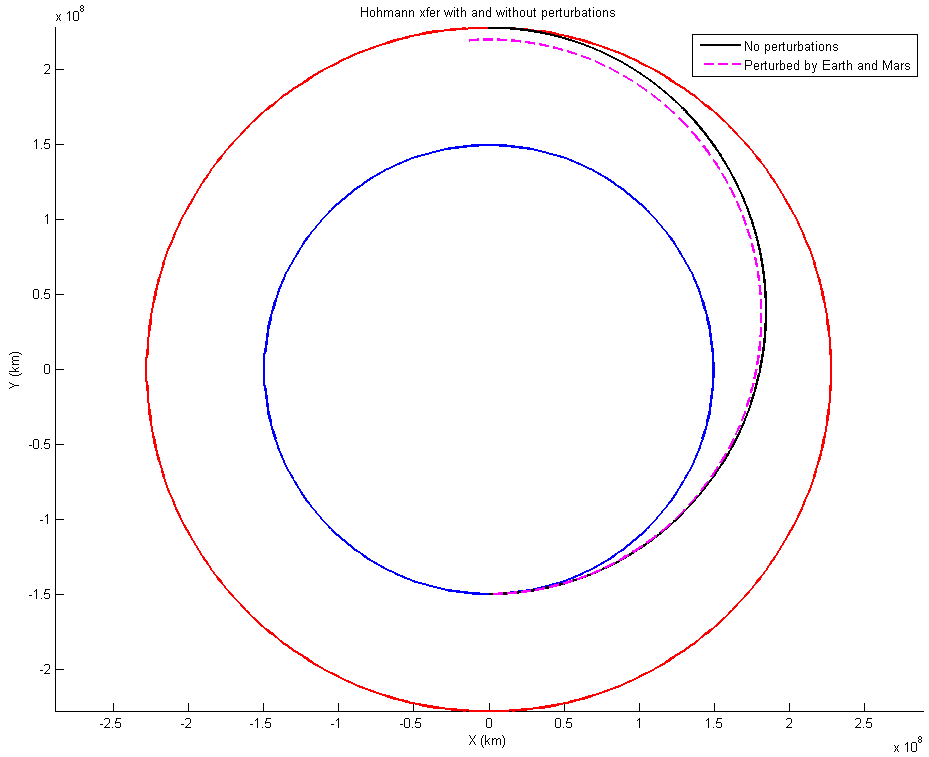
```
state = [r_earth; v_earth; r_mars_i; v_mars_i; r_sat; v_sat;];

% Anonymous function to calculate heliocentric acceleration
sun_accel = @(r) -mu_sun*r/(norm(r))^3;

% Anonymous function to calculate combined accel on satellite from earth,
% mars, and sun
total_accel = @(re, rm, rs) -mu_earth*(rs-re)/(norm(rs-re))^3 + ...
    -mu_mars*(rs-rm)/(norm(rs-rm))^3 + ...
    sun_accel(rs);

% This is the state derivative.
state_dot = @(t, state) [state(4:6); sun_accel(state(1:3));...
    state(10:12); sun_accel(state(7:9));...
    state(16:18); total_accel(state(1:3),state(7:9),state(13:15))];

[T_out, X_out] = ode45(state_dot, times, state, ode_opts);
pert_X_x_store = X_out(:,13);
pert_X_y_store = X_out(:,14);
pert_X_vx_store = X_out(:,16);
pert_X_vy_store = X_out(:,17);

earth_x_store = X_out(:,1);
earth_y_store = X_out(:,2);
mars_x_store = X_out(:,7);
mars_y_store = X_out(:,8);
```

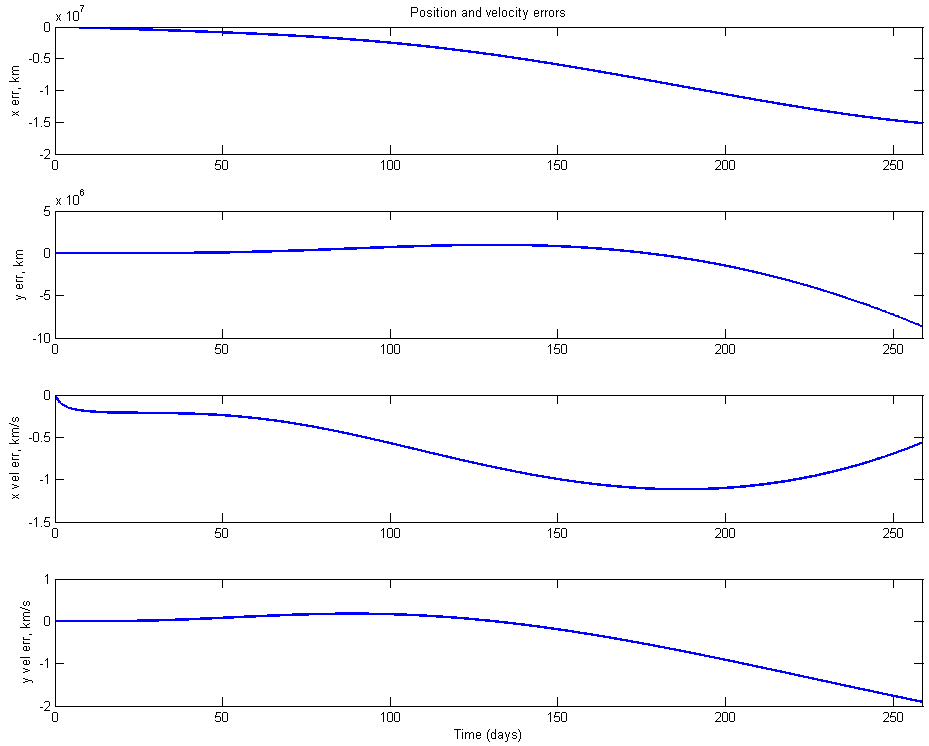
Plots

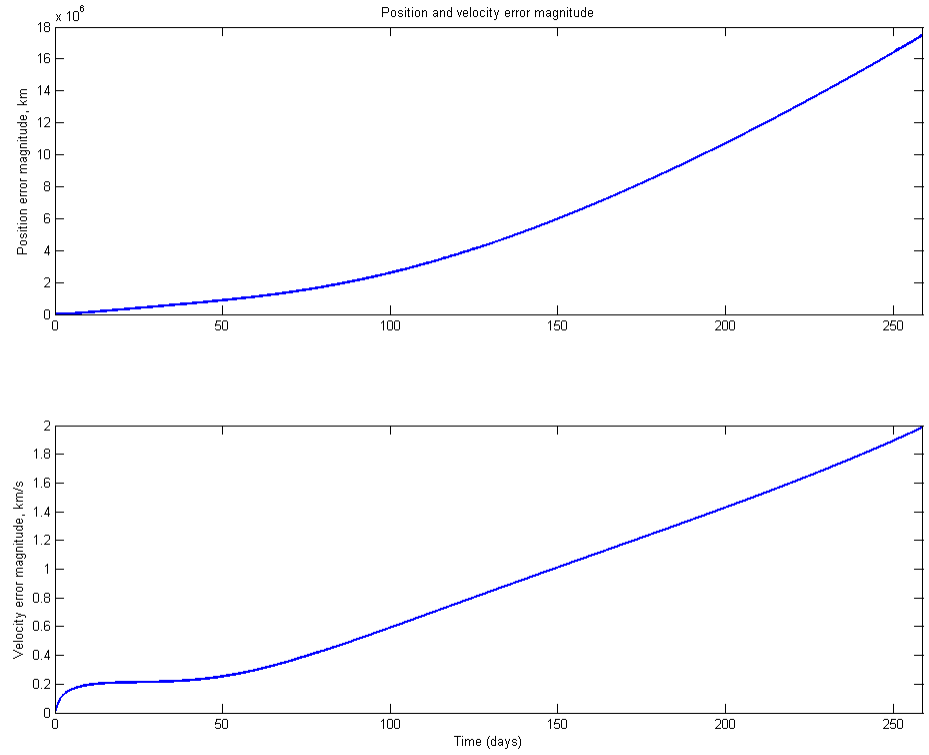
```
polar_plot = figure('Position', hw_pub.figPosn);
```

```
hold on
angles = 0:1:360;
plot(norm(r_earth)*cosd(angles),norm(r_earth)*sind(angles), ...
    'LineWidth', hw_pub.lineWidth)
plot(norm(r_mars_f)*cosd(angles),norm(r_mars_f)*sind(angles),'r', ...
    'LineWidth', hw_pub.lineWidth)
no_pert_plot = plot(X_x_store, X_y_store,'k', ...
    'LineWidth', hw_pub.lineWidth);
axis equal
pert_plot = plot(pert_X_x_store, pert_X_y_store,'m--', ...
    'LineWidth', hw_pub.lineWidth);
% plot(earth_x_store, earth_y_store,'m--') % Earth motion propagated
% plot(mars_x_store, mars_y_store,'b--') % Mars motion propagated
xlabel('X (km)');
ylabel('Y (km)');
title('Hohmann xfer with and without perturbations')
legend([no_pert_plot, pert_plot],{'No perturbations', ...
    'Perturbed by Earth and Mars'});

figure('Position', hw_pub.figPosn)
plot_times = times/3600/24;
lims = [0;plot_times(end)];
subplot(4,1,1)
plot(plot_times, pert_X_x_store-X_x_store, 'LineWidth', hw_pub.lineWidth)
title('Position and velocity errors')
ylabel('x err, km')
xlim(lims)
subplot(4,1,2)
plot(plot_times, pert_X_y_store-X_y_store, 'LineWidth', hw_pub.lineWidth)
ylabel('y err, km')
xlim(lims)
subplot(4,1,3)
plot(plot_times, pert_X_vx_store-X_vx_store, 'LineWidth', hw_pub.lineWidth)
ylabel('x vel err, km/s')
xlim(lims)
subplot(4,1,4)
plot(plot_times, pert_X_vy_store-X_vy_store, 'LineWidth', hw_pub.lineWidth)
ylabel('y vel err, km/s')
xlim(lims)
xlabel('Time (days)')

% norm of r_non_perturbed - r_perturbed
diff_r = sqrt((X_x_store-pert_X_x_store).*(X_x_store-pert_X_x_store) ...
    + (X_y_store-pert_X_y_store).*(X_y_store-pert_X_y_store));
% norm of v_non_perturbed - v_perturbed
diff_v = sqrt((X_vx_store-pert_X_vx_store).*(X_vx_store-pert_X_vx_store) ...
    + (X_vy_store-pert_X_vy_store).*(X_vy_store-pert_X_vy_store));

figure('Position', hw_pub.figPosn)
subplot(2,1,1)
plot(plot_times, diff_r, 'LineWidth', hw_pub.lineWidth)
title('Position and velocity error magnitude')
ylabel('Position error magnitude, km')
xlim(lims)
```

```
subplot(2,1,2)
plot(plot_times, diff_v, 'LineWidth', hw_pub.lineWidth)
ylabel('Velocity error magnitude, km/s')
xlim(lims)
xlabel('Time (days)')
```




Position and velocity errors
x 10^7
x err, km
x 10^6
y err, km
x vel err, km/s
y vel err, km/s
Time (days)


Position and velocity error magnitude
x 10^6
Position error magnitude, km
Velocity error magnitude, km/s
Time (days)

Conclusion

The perturbations that were modeled are slowing down the satellite with respect to the sun. At the beginning, Earth gravity is providing an acceleration against the satellite velocity vector. Toward the end, Mars is lagging in phase with the satellite and slowing it down in the velocity vector direction. These accelerations result in a lowered apoapsis.

Published with MATLAB® R2013b

```
function DCM = Euler2DCM( seq_string, angle_vector )
%Euler2DCM Turn an Euler Angle set into a DCM
%   Angle vector in radians
fcnPrintQueue(mfilename('fullpath'))

DCM = eye(3);
%get the trig functions
num_rot = length(seq_string);
c = zeros(num_rot,1);
s = zeros(num_rot,1);

for idx = 1:num_rot
c(idx) = cos(angle_vector(idx));
s(idx) = sin(angle_vector(idx));
end


for idx = num_rot:-1:1
    if strcmp(seq_string(idx),'1')
        M = [1 0 0; 0 c(idx) s(idx); 0 -s(idx) c(idx)];
        DCM = DCM*M;
    elseif strcmp(seq_string(idx),'2')
        M = [c(idx) 0 -s(idx); 0 1 0; s(idx) 0 c(idx)];
        DCM = DCM*M;
    elseif strcmp(seq_string(idx),'3')
        M = [c(idx) s(idx) 0; -s(idx) c(idx) 0; 0 0 1];
        DCM = DCM*M;
    else
        fprintf('%s is not a valid axis\n', seq_string(idx))
    end
end


end
```

Published with MATLAB® R2013b

```
function state_dot = two_body_state_dot(t, state, opts)
%two_body_state_dot   Return state_dot given state. Used for numerical
%integration
% The first 6 elements are assumed to be r and v. If a state transition
% matrix (STM) is to be calculated as well, opts.OD needs to be set up.
% Currently assumes r/v are the only state elements that have non-zero
% derivatives.
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

state_dot = zeros(length(state),1);
state_dot(1:3) = state(4:6);
mu = 3.986e5; % km3/s2
if isfield(opts, 'mu')
    mu = opts.mu;
end

r_vec = (state(1:3));
r = norm(r_vec);
state_dot(4:6) = -mu * r_vec/(r*r*r); % Simple 2-body

% Use J2
if isfield(opts, 'J2') && isfield(opts.J2, 'use')
    if opts.J2.use == 1
        if isfield(opts.J2, 'params')
            state_dot(4:6) = state_dot(4:6) + J2_accel(state(1:3), opts.J2.params)
        else
            state_dot(4:6) = state_dot(4:6) + J2_accel( state(1:3) );
        end
    end
end

% Use J3
if isfield(opts, 'J3') && isfield(opts.J3, 'use')
    if opts.J3.use == 1
        if isfield(opts.J3, 'params')
            state_dot(4:6) = state_dot(4:6) + J3_accel(state(1:3), opts.J3.params)
        else
            state_dot(4:6) = state_dot(4:6) + J3_accel( state(1:3) );
        end
    end
end

% Use drag
if isfield(opts, 'drag') && isfield(opts.drag, 'use')
    if opts.drag.use == 1
        state_dot(4:6) = state_dot(4:6) + drag_accel( state, opts.drag );
    end
end

% For orbit determination application
if isfield(opts, 'OD')
    if opts.OD.use == 1
```

```
        opts.OD.state_len;
        % The OD.state_len is the length of the estimation state. The rest
        % is the STM, numerically propagated with the A-Matrix
        A = opts.OD.A_mat_handle(state(1:opts.OD.state_len),opts.OD.A_params);

        % Block matrix multiplication
%         long_dim = 9;
%         STM = zeros(long_dim);
        STM = reshape(state(opts.OD.state_len+1:end),...
            opts.OD.A_params.important_block(1),...
            opts.OD.A_params.important_block(2));
        STM_dot = ...
            A(1:opts.OD.A_params.important_block(1),1:opts.OD.A_params.important_b
            *STM;
        % Pack up the important stuff
        state_dot(opts.OD.state_len+1:end) = reshape(STM_dot,...
            opts.OD.A_params.important_block(1)*opts.OD.A_params.important_block(2
    end
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