```
function [r_f,v_f, r, v, time] = Propagate(tf, r0, v0, Plot, dt)
%-------------------------------------------------------------------------
% Uses STM to propagate an orbit forward
%-------------------------------------------------------------------------
% Inputs are Variation but should be:
%   tf     - final time to prpogate to [s]
%   r0     - 3x1 initial position [m] in cartesian
%   v0     - 3x1 initial velocity [m/s] in cartesian
%   Plot   - True or False whether to plot the orbit
%   dt     - optional delta to propagate
% Output:
%   J  - 6x6 Jacobian Matrix
%
mu = 3.986004418e14; % m^3/s^2

if nargin < 5
    dt=10;
end

%J2=.001082; %Non spherical coefficient


%% Calculation of Oribt for Earth as a Sphere
counter=0; %used to count each itiration
t=0;  %initial time
r_temp=r0;  %initial position (as a row) [m]
v_temp=v0;  %initial velocity (as a row) [m/s]
time=[];
while t<=tf
    counter=counter+1;
    time(counter)=t;
    r(counter, :)=r_temp;   %next position iteratio
    v(counter, :)=v_temp;   %next velocit iteration

    %Runga Kutta m4th order method for both r and v
    k1=dt*Derivative(r(counter, :), v(counter, :), mu);
    k2=dt*Derivative(r(counter, :)+k1(1, :)/2, v(counter, :)+k1(2, :)/2, mu);
    k3=dt*Derivative(r(counter, :)+k2(1, :)/2, v(counter, :)+k2(2, :)/2, mu);
    k4=dt*Derivative(r(counter, :)+k3(1, :), v(counter, :)+k3(2, :), mu);
    t=t+dt; %increase iteration of time
    r_temp=r(counter, :)+(k1(1,:)+2*k2(1,:)+2*k3(1,:)+k4(1,:))/6; %set next↙
iteration of r to runga kutta approxiamtion
    v_temp=v(counter, :)+(k1(2,:)+2*k2(2,:)+2*k3(2,:)+k4(2,:))/6;  %set next↙
iteration of v to runga kutta approxiamtion
end

r_f=r(end,:)';
v_f=v(end,:)';

if Plot
    figure
    earth_sphere('km');
    axis equal
    hold on
```

```
    % Plot propagated orbit
    plot3(r(:, 1)/1000, r(:, 2)/1000, r(:, 3)/1000, 'b', 'LineWidth', 1.5);
    % Labels and title
    title('Orbit of Satellite Around (Spherical) Earth');
    xlabel('x in ECI [km]');
    ylabel('y in ECI [km]');
    zlabel('z in ECI [km]');
    grid on
    hold off
end
end

function M = Derivative(x, y, mu)
M(1, :)=y;   %x dot, y dot and v dot are already given in the velocity vector
M(2, :)=-mu*(x/(norm(x)^3));  %accoridng to equation r double dot = -u* r hat / r↙
magnitude ^3
end

% function state_der = SatelliteDynODE(t, state, Mu) %, Re, j2, A_D, C_D, m, C_sr,↙
A_sp, Date)
%     r_vec = [state(1) state(2) state(3)]';
%     v_vec = [state(4) state(5) state(6)]';
%     r = norm(r_vec);
%     %[a,e,i,w,OMEGA,Nu] = ijk2keplerian(r_vec, v_vec);
%     %F_d = Drag_Per(Re, A_D, C_D, m, r, v_vec);
%     %F_j = j2_Per(Re, Mu, r, i, Nu, w, OMEGA, j2);
%     %F_s = Solar_Per(Date, C_sr, A_sp, m, r_vec);  % Pass r_vec to Solar_c
%     F_tot=[0, 0, 0]; %F_tot = F_d + F_j + F_s;
%     X_2dot = -(Mu/r^3)*r_vec(1) + F_tot(1);
%     Y_2dot = -(Mu/r^3)*r_vec(2) + F_tot(2);
%     Z_2dot = -(Mu/r^3)*r_vec(3) + F_tot(3);
%     state_der = [v_vec; [X_2dot Y_2dot Z_2dot]'];
% end
```