AE4879 Mission Geometry and Orbit Design

Assignment 10: Earth coverage II
EARTHCOV-7

Simon Billemont

December 17, 2010

Assignment 10: Earth coverage II

This assignment deals with determining the ground track for satellites in a circular low earth orbit. This is done using the dual-axis spiral as elaborated in Assignment 8.

1. Ground track determination

The path of the satellite over the earth, can be composted of two rotations. The first being rotation the of the orbital plane wrt the ECI reference frame. The second is the rotation of the satellite in the orbital plane around the earth. In essence, the problem is thus a dual-axis problem.

The first rotation, how the earth moves under the satellite, has a characteristic rotation pole and rotation speed. The pole of the rotation is simply the Earth north pole. For the rotation speed, this is basically the rotation speed of the earth with superimposed on this, additional orbital plane perturbations. The major perturbation is the J2 effect of the earth mass distribution. The rotation of the earth and the J2 contribution can be found using OCDM[4] eq 9-66 (or see eq 2). Note φ is latitude or elevation and θ is longitude or azimuth.

$$\varphi_C = 90^\circ; \theta_C = 0^\circ \tag{1}$$

$$\begin{aligned} \omega_C &= -\omega_{day} - \omega_{J2} \\ &= -0.004178 - 2.396288 \cdot 10^9 a^{-7/2}(1-e^2)^{-2}\cos i \qquad [deg/s] \end{aligned} \tag{2}$$

The second rotation of the satellite, has an orbit pole perpendicular to the orbit plane. This means that latitude of the orbit pole is the latitude of the obit plane (inclination i) plus 90°. The longitude of the pole varies as it rotates around C, the primary rotation axis. The rotation speed around this pole is the mean angular motion of the satellite orbit. Since the orbit is asumed to be a circular Keplerian orbit, the mean angular motion is defined by eq 4 (see OCDM[4] Table 2-4) .

$$\varphi_S = i + 90^\circ, \theta_S = \theta_{S,0} + \omega_C \cdot t \tag{3}$$

$$\omega_S = n = \sqrt{\frac{\mu}{a^3}} \qquad [rad/s] \tag{4}$$

Where i is the orbit inclination, $\theta_{S,0}$ is a constant defining the orbit pole start position, t is the time considered, a is the orbit semi-major axis and μ is the standard gravitational parameter of the orbited object ($\mu_{earth} = 3.986 \cdot 10^{14} m^3/s^2$). This means that the angular distance ($\rho_C S$) from C to S is $90 - \varphi_S = i$. The angular distance from S to P is 90°as S is the orbit pole and P is in the orbital plane. Summarizing, the parameters needed to plot the ground track are in table 1 (Orbit parameters of ENVISAT see [1]).

Table 1: Dual-axis spiral parameters

parameter	value	parameter	value
μ	μ_{earth}	$\rho_C S$	i
R_E	6371 km	$\rho_S P$	90°
H	800 km	$\theta_{S,0}$	0°
i	98.5°	$\theta_{P,0}$	0°
n	$n(\mu, R_E, H)$	ω_C	ω_C(Re, H, i)
P	$2\pi/n$	ω_S	n

2. EVNICSAT orbit

Consider the satellite ENVISAT (altitude = 800 km, i = 98.5°)

- **Compute the ground track of this satellite during one complete revolution, using the dual-axis spiral technique. Use a step-size of 1 minute.**
- **Compute the velocity as a function of time, during that revolution.**
- **Compute the location of the Euler pole as well as the angular distance ρ_E during that revolution (both as a function of time).**
- **Show that the maximum velocity is equal to ω_E, and that the minimum velocity is equal to $\omega_E \sin(90 + \delta_0 - \delta_E)$.**

The dual-axis spiral can be used determine a ground track segment the ESA satellite ENVISAT (details see [1]). This is done for a single orbital revolution ($\pm 6000s \approx 100.7min$) with a time-step of 1 min. This is visualized in figure 1. It shows both the 3D view of the orbit and a 2D equirectangular projection ($\varphi; \theta$). The color is a measure for the time in the orbit (t=0: blue, t=P: red). Furthermore it shows the location of the secondary orbit pole (S) and the instantaneous Euler pole (E) all with the same time color coding.

What can be seen is that when the satellite rises (from S to N, red section), the longitude decreases, confirming the retrograde orbit of ENVISAT. And as the satellite orbits the Earth, the earth rotates to the East. This means that the satellie tracks must of the next orbit must be to the West of the existing track. This also happens in the simulation (top of the end track [red] crosses the start point[blue]). Also notice that the Euler pole latitude does not change.

From the dual-axis spiral Euler pole, one can also determine the instantaneous velocity of the satellite over the earth. This can be determined using eq 5 (see OCDM[4] eq8-26 and Assignment 8).

$$v = \omega_E \sin \rho_E \tag{5}$$

Because this equation depends on ρ_E which is variable due to its dependency on the elevation of P, the velocity of the satellite is variable. The variations of both the velocity of the ground track (v) and the angular distance between E and P (ρ_E) are shown in fig 2. What can be seen is that the velocity of the satellite on the ground (in

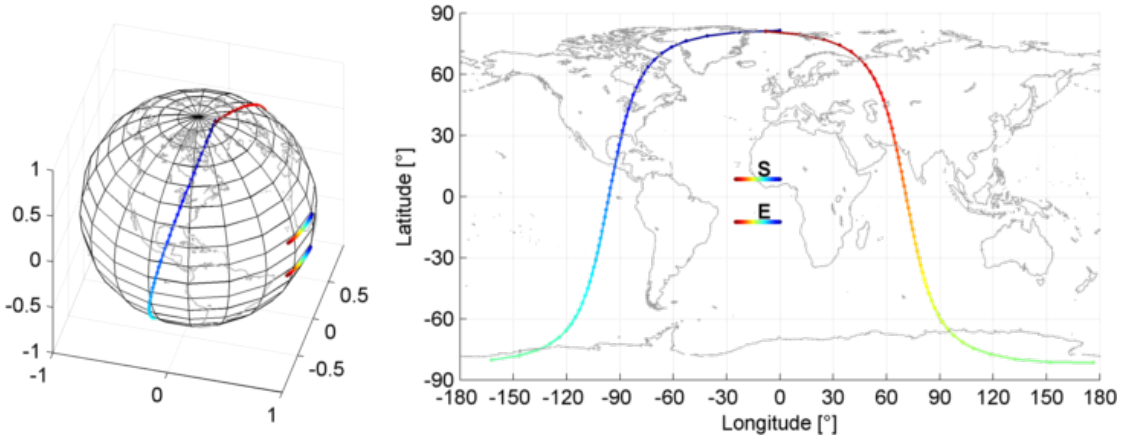

Figure 1:

[deg/s]) has about double the frequency of the distance with respect to the Euler pole. This happens because ρ_E is centered around 90°. Furthermore, it can be confirmed that the maximum ground track velocity that is reached is ω_E and the minimum velocity $\omega_E \sin(90° + (\varphi_S - \varphi_E))$. The minimum velocities are reached when the satellite is at the top and bottom of the orbit, while it reaches maximum velocities are near the equator.

References

[1] R. Noomen, *AE4-879 Earth coverage V3.2*, TUDelft Lecture Slides, 2010.

[2] D. Austin and W. Dickinson. (2009) Spherical easel. A spherical drawing program. [Online]. Available: http://merganser.math.gvsu.edu/easel/

[3] MathWorks. (2010a) Matlab 7.11. Natick, MA.

[4] J. R. Wertz, *Orbit & Constellation Design & Management*, second printing ed. El Segundo, California: Microcosm Press, 2009.

[5] E. W. Weisstein. (2010) Trigonometry. MathWorld. [Online]. Available: http://mathworld.wolfram.com/Trigonometry.html

[6] S. Eddins. (2010, Nov) Matlab xunit test framework. MATLAB Central File Exchange. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/22846

Additional information

Estimated work time:

$\sim$ 1h Studying theory + $\sim$ 3h making assignment + $\sim$ 3h writing report = $\sim$ 7h

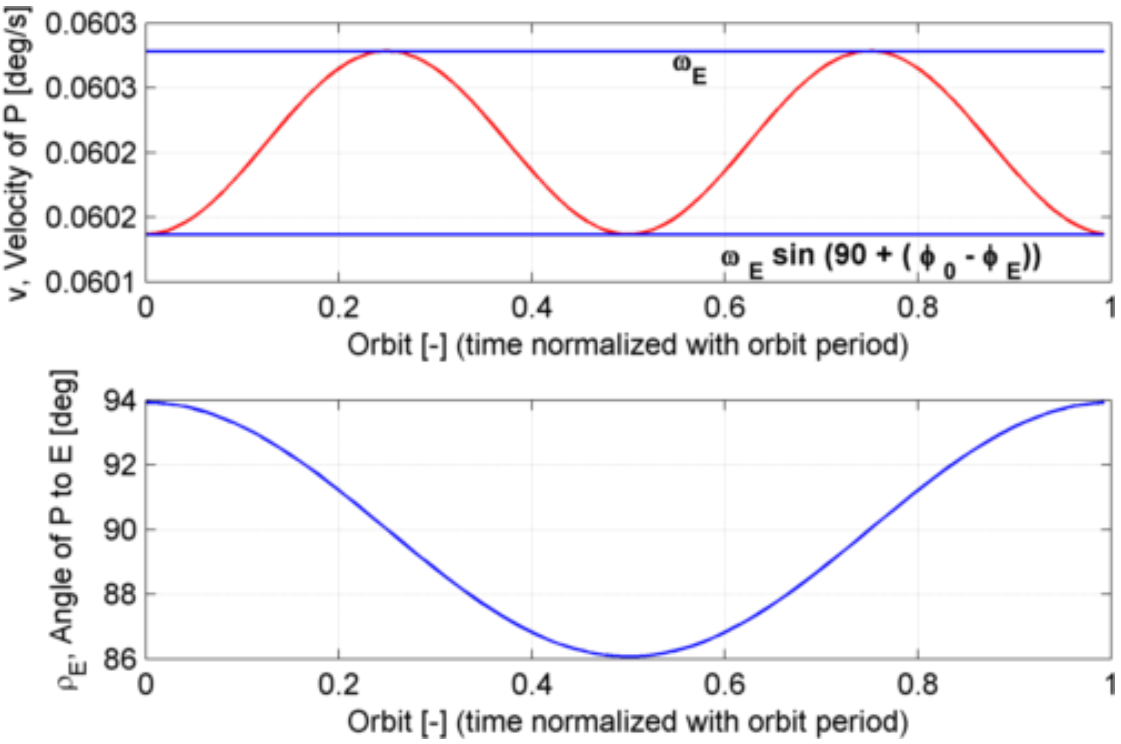

Figure 2:

Made by

Simon Billemont

Stud Nr: 1387855

s.billemont@student.tudelft.nl

License and notices

Version history

Version 1: Initial document

A. Matlab source code

The code written to implement the three described optimizers was written in MATLAB 7.11 (2010b)[3]. A structured overview of the dependencies is given below:

- EARTHCOV7.m
  - DualAxisProblem.m
    - H.m
    - acos2.m
  - globePlot2.m

The DualAxisProblem, H, and acos2 functions and classes have not been changed for this assignment (see Assignment 8). The globePlot2 function is not included as it is only a plotting function. It is available on http://angelcorp.be .

Listing 1: EARTHCOV7.m: Compute the ENVISAT orbit

```
%% By: Simon Billemont, sbillemont, 1387855
%   Contact: aodtorusan@gmail.com or s.billemont@student.tudelft.nl
%       Solve the dual-axis spiral problem for the ENVISAT orbit
%       Equations from Orbit & Constellation Design & Management (ocdm)
%   Made on: 11-12-2010 (dd-mm-yyyy)
%   This work is licensed under the
%       Creative Commons Attribution-NonCommercial 3.0 Unported License.
%       To view a copy of this license, visit http://creativecommons.org/licenses/by-nc/3.0/
%% Setup environment
clc
clearvars
close all

% Make a new utility for saving pictures
saver = ImSav();
saver.plotsDir = '../images/';            % Where the plots will go
saver.appendFigureNr = 0;                 % Do not append stuff the the name
saver.enabled = 0;

%% Only procede if all the unit tests are succesfull
if (~runtests)
    error('There are errors in the test cases. Fix these first')
end

%% Constants
% Earth constants
Re   = 6371E3;              % m
mu_E = 3.98600441E14;       % m^3/s^2

%% Compute

% ENVISAT Orbit param
h = 800E3;                  % m
i = deg2rad(98.5);          % rad
a = Re + h;                 % Semi-major axis [m]
e = 0;                      % Eccentricity [-]
P = 2*pi*sqrt(a.^3./mu_E);  % Period [s]

% rotation speed enivsat
%wR = -wDay -wJ2 = -0.004178 - 2.396288 10^9 a^(-7/2) (1-e^2)^(-2) cos(i) [deg/s]
wR = -0.004178-2.396288E9*a^(-7/2)*((1-e^2)^(-2))*cos(i); % [deg/s]
wR = deg2rad(wR);           % [rad/s]
n = (2*pi)/P;               % [rad/s]

% dual axis spiral param
r1 =     i;         % rad
r2 =     pi/2;      % rad
phi1_0 = 0;         % rad
phi2_0 = 0;         % rad
omega1 = wR;        % rad/s
omega2 = n;         % rad/s

time = 0:60:1*P;      % Time for the dual axis spiral, 1 orbit [s]

% Find all the dual axis spiral parameters
dap = DualAxisProblem(r1, r2, phi1_0, phi2_0, omega1, omega2);
s = dap.eval(time);

%% Fancy 3d / 2d unwrap plot
globePlot2(s.az, s.el, 1:length(s.az), 'drawCoast', true);          % ENVISAT track
```

```
hold on
    globePlot2(s.phi1, (i-pi/2)*ones(size(s.phi1)), 1:length(s.az));% Point S
hold on
    globePlot2(s.phi1, pi/2 - s.dE_, 1:length(s.az));               % Point E
hold on
    % Fancy labels
    text(mean(rad2deg(s.phi1)), 5+rad2deg(mean((i-pi/2)*ones(size(s.phi1)))), 'S', 'FontWeight', 'bold')
    text(mean(rad2deg(s.phi1)), 5+rad2deg(mean(pi/2 - s.dE_)), 'E', 'FontWeight', 'bold')
hold off
saver.saveImage('globe'); % Save to disk

%close

%% Plot rE and v of the orbit
clf % clear the figure

% Plot velocity, min and max
subplot(211)
plot(s.t(:)/P, rad2deg(s.v), 'r', 'linewidth', 1) % v
hold on
    % max, omega E
    plot(s.t(:)/P, rad2deg(s.wE), 'linewidth', 1)
    text(mean(s.t/P)+.05, mean(rad2deg(s.wE))-1.1E-5, '\omega_E', 'FontWeight', 'bold');
    % min omega_E sin(90+(lat_s-lat_e))
    plot(s.t(:)/P, rad2deg(s.wE.*sin(pi/2-i+s.dE_)), 'linewidth', 1)
    text(mean(s.t/P)+.1, mean(rad2deg(s.wE.*sin(pi/2-i+s.dE_)))-2E-5, ...
           '\omega _E sin (90 + ( \phi _0 - \phi _E))', 'FontWeight', 'bold');
hold off
ylabel('v, Velocity of P [deg/s]')
xlabel('Orbit [-] (time normalized with orbit period)')
grid on

% Plot of the distance E P (rho_E)
subplot(212)
plot(s.t(:)/P, rad2deg(s.rE), 'linewidth', 1)
xlabel('Orbit [-] (time normalized with orbit period)')
ylabel('\rho_E, Angle of P to E [deg]')
grid on

saver.saveImage('v_rE');
```

Listing 2: DualAxisProblem.m: Dual-axis spiral solver

```
classdef DualAxisProblem < handle
%DUALAXISPROBLEM Solve the Dual-Axis spiral problem
%   Find the angles and rates of point P in function of the given times
%   For details see Orbit & Constellation Design & Management (ocdm) chapter 8
% By: Simon Billemont (s.billemont@student.tudelft.nl), on: 16-11-2010 (dd-mm-yyyy)
% Licence: Creative Commons Attribution-NonCommercial 3.0 Unported License.
%   12-12-2010  Changed names in output
    properties (Access=public)
        r1      % Initial condition; distance C => S [rad]
        r2      % Initial condition; distance S => P [rad]
        phi1_0  % Initial condition; start angle around C between North and S [rad]
        phi2_0  % Initial condition; start angle around S between North and P [rad]
        omega1  % Rate of ratation of S around C [rad/s]
        omega2  % Rate of ratation of P around S [rad/s]
    end

    methods (Access=public)
        function obj = DualAxisProblem(r1, r2, phi1_0, phi2_0, omega1, omega2)
            % Constructor, sets the IC, based on the given values
            if nargin == 1 % Seperate values given
                obj.r1 = r1(1);             obj.r2 = r1(2);
                obj.phi1_0 = r1(3);         obj.phi2_0 = r1(4);
                obj.omega1 = r1(5);         obj.omega2 = r1(6);
            else % Values given as a vector
                obj.r1 = r1;                obj.r2 = r2;
                obj.phi1_0 = phi1_0;        obj.phi2_0 = phi2_0;
                obj.omega1 = omega1;        obj.omega2 = omega2;
            end
        end
        function s = eval(obj, t)
            % Find the angles and rates of P for the given timesamples
            if nargin==1
                t=0; % Default time sample
            end

            s = struct('t', t); % Stores all the results
            % Compute the angles and rates in the order descibed in slide 36
            s.phi1 = obj.azimuthS_C(s.t);
            s.phi2 = obj.azimuthP_S(s.t);
            s.el =   obj.elevationP_C(s.phi2);
            s.da =   obj.changeAzimuthP_C(s.el, s.phi2);
            s.az =   obj.azimuthP_C(s.phi1, s.da);
            s.dE_ =  obj.angleC_E(s.phi1);
            s.rE =   obj.angleP_E(s.dE_, s.el, s.da);
            s.wE =   obj.rotationE(s.phi1);
            s.v =    obj.velocityP(s.wE, s.rE);
            s.dPsi = obj.changeMotionP(s.dE_, s.rE, s.el, s.da);
            s.psi =  obj.directionP(s.dPsi);
        end
```

```
    end

    methods (Access=private)
        %% Intermediate values
        function phi1 = azimuthS_C(obj, t)
            % Azimuth of S around C relative to alpha = 0; ocdm eq 8-27
            phi1 = obj.phi1_0 + obj.omega1 .* t;
        end
        function phi2 = azimuthP_S(obj, t)
            % Azimuth of P around S realtive to C; ocdm eq 8-27
            phi2 = obj.phi2_0 + obj.omega2 .* t;
        end
        function da = changeAzimuthP_C(obj, d, phi2)
            % Delta alpha: Change in azimuth of P around C; ocdm eq 28-a
            da = acos2( ...
                (cos(obj.r2)-cos(obj.r1).*sin(d))./(sin(obj.r1).*cos(d)), ...
                -H(phi2));
        end
        function rE = angleP_E(obj, dE_, d, da)
            % rho E: Angle from P to E; ocdm eq  8-24
            rE = acos(cos(dE_).*sin(d)+sin(dE_).*cos(d).*cos(da));
        end
        function wE = rotationE(obj, phi1)
            % omega E: Rate of ratation about E; ocdm eq  8-23b
            wE = ones(size(phi1)) * ...
                    sqrt(obj.omega1.^2 +obj.omega2.^2 + 2.*obj.omega1.*obj.omega2.*cos(obj.r1));
        end
        function dE_ = angleC_E(obj, phi1)
            % Angle from C to E; ocdm eq  8-23a
            dE_ = ones(size(phi1)) * mod( ...
                    atan((obj.omega2.*sin(obj.r1))./(obj.omega1+obj.omega2.*cos(obj.r1))) ...
                    , pi);
        end
        function dPsi = changeMotionP(obj, dE_, rE, d, da)
            % Change in direction of motion of P; ocdm eq  8-25a
            dPsi = acos2( ...
                (cos(dE_)-cos(rE).*sin(d))./(sin(rE).*cos(d)), ...
                H(da));
        end
        %% Results
        function a = azimuthP_C(obj, phi1, da)
            % Azimuth of P around C (final  azimuth); ocdm eq  8-28b
            a = mod(phi1 + da, 2.*pi);
        end
        function d = elevationP_C(obj, phi2)
            % Elevation of P relative to C (final elevation); ocdm eq 8-28c
            d = pi./2 - acos(cos(obj.r1).*cos(obj.r2)+sin(obj.r1).*sin(obj.r2).*cos(phi2));
        end
        function v = velocityP(obj, wE, rE)
            % Velocity of P; ocdm eq  8-26
            v = wE .* sin(rE);
        end
        function psi = directionP(obj, dPsi)
            % Direction of motion of P; ocdm eq  8-25b
            psi = mod(dPsi - pi./2, 2.*pi);
        end
    end

end
```

Listing 3: H.m: OCDM[4] Hemisphere function

```
function [ H ] = H( phi )
    %H Hemisphere function
    %       H(phi) = +1 if (0   <= ?modulo360 < 180 [deg])
    %       H(phi) = -1 if (180 <= ?modulo360 < 360 [deg])
    %       Based on OCDM p389 eq 8-1
    % By: Simon Billemont (s.billemont@student.tudelft.nl), on: 12-2010 (mm-yyyy)
    % Licence: Creative Commons Attribution-NonCommercial 3.0 Unported License.
    phi = mod(phi, 2*pi);

    H = ones(size(phi));
    H(pi <= phi & phi < 2*pi) = -1;
end
```

Listing 4: acos2.m: OCDM[4] quadrant sensitive arccos function

```
function [ acos2 ] = acos2( cosPhi, H )
    %ACOS2 Quadrant sensative arcCosine, uses hemisphere function H
    %     acos2[cos(phi),H(phi)] = {H(phi) acos(cos(phi)}modulo360
    %     Based on OCDM p389 eq8-2
    % By: Simon Billemont (s.billemont@student.tudelft.nl), on: 12-2010 (mm-yyyy)
    % Licence: Creative Commons Attribution-NonCommercial 3.0 Unported License.
    if (any(cosPhi > 1))
        warning('MATLAB:acos2:bounds', ...
            'cosPhi values where clamped to [0, 1] <min:%f max:%f>\n', min(cosPhi), max(cosPhi));
        cosPhi(cosPhi > 1) = clamp(cosPhi(cosPhi > 1),0,1);
    end
    acos2 = mod(H .* real(acos(cosPhi)), 2*pi);
end
```