AE4879 Mission Geometry and Orbit Design

Assignment 7: Mission geometry
GEOMETRY-2

Simon Billemont

November 19, 2010

Assignment 7: Mission geometry

This assignment deals with the familiarization of spherical geometry and the techniques used to compute the different angles, sides and areas in spherical trigonometry.

1. Background

Compute the angular distances, rotation angles and total area for each of the following spherical triangles, given by the following sets of points (azimuth, elevation)

1. **P1 = (0,0), P2 = (90,0), P3 = (0,90)**
2. **P1 = (0,0), P2 = (30,0), P3 = (0,90)**
3. **P1 = (0,20), P2 = (10,25), P3 = (5,30)**
4. **P1 = (5,0), P2 = (30,30), P3 = (5,90)**
5. **P1 = (0,-20), P2 = (45,-20), P3 = (0,90)**
6. **P1 = (0,20), P2 = (10,25), P3 = (5,30) (outer area, cf. OCDM[1] Fig 6-13F)**

First, the azimuth (φ) and elevation (θ) are converted to Cartesian vectors on the unit sphere (see eq 1). This is done because the equations to compute the spherical triangle sides and angles also use these vectors (alternative equations also exist using pure angles).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sin(\theta) \cdot \cos(\varphi) \\ \sin(\theta) \cdot \sin(\varphi) \\ \cos(\theta) \end{bmatrix} \tag{1}$$

The 'length' of the sides of the triangle (or rotations) can easily be found by evaluating the angle between the unit vectors to each point (see eq 2). Then for actual angles of the triangle, the eq 3 from OCDM[1] p298 was used. The area of a spherical triangle can be found using the eq 4 (also from OCDM[1] p298).

$$a = cos^{-1}\left(\mathbf{P_1} \cdot \mathbf{P_2}\right) \tag{2}$$

$$C = \tan^{-1}\left(\frac{\mathbf{P_3} \cdot (\mathbf{P_1} \times \mathbf{P_2})}{(\mathbf{P_1} \cdot \mathbf{P_2}) - (\mathbf{P_3} \cdot \mathbf{P_1}) \times (\mathbf{P_3} \cdot \mathbf{P_2})}\right) \tag{3}$$

$$\text{area} = \Sigma - (n-2) \cdot \pi; \tag{4}$$

Where P_1 to P_3 are the points of the triangle, a to c are the sides and A to C are the rotation angles. Furthermore Σ is the sum of the rotation angles and n is the number of sides (3 for a triangle).

For the last triangle, the outside area is considered the triangle. This means that the rotation angles are the outside angles and the surface area is also the outside area. Therefore, the rotation angles computed by the equations above must be corrected by subtracting them from 2π. Also the area is corrected, this is done by inverting the surface area (subtracting 4π from the inner triangle area).

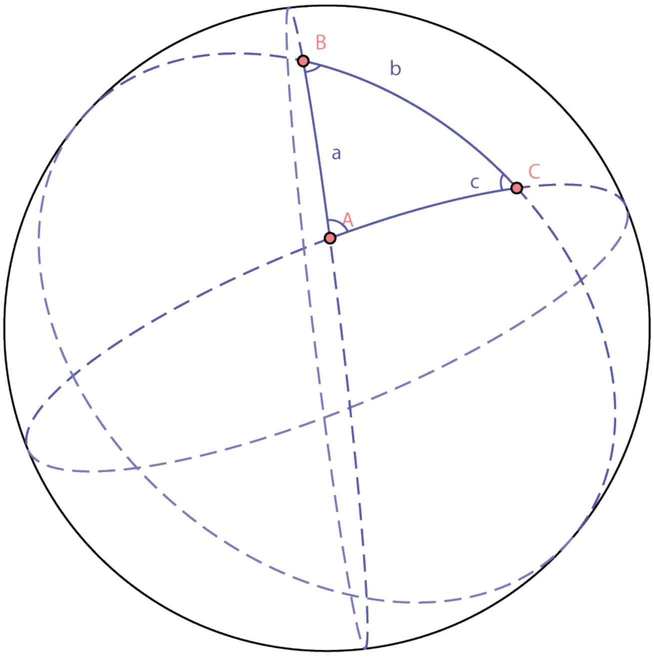

Figure 1: Triangle side and angle notations (made with [2])

Triangle#	side a[deg]	side b[deg]	side c[deg]	angle A[deg]	angle B[deg]	angle C[deg]	area
1	90.00	90.00	90.00	90.00	90.00	90.00	1.57
2	30.00	90.00	90.00	90.00	90.00	30.00	0.52
3	10.50	6.68	10.97	36.37	75.99	68.24	0.01
4	38.29	60.00	90.00	36.20	137.00	25.00	0.32
5	42.15	110.00	110.00	98.06	98.06	45.00	1.07
6	10.50	6.68	10.97	323.63	284.01	291.76	12.56

Table 1:

2. Results

For the resulting angles and sides, the notations as visualized in fig 1 is used. The resulting angles and areas are given in table 1. The triangle numbers refer the to the same numbers as given in the question (the 1)

References

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

[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] R. Noomen, *AE4-879 Mission Geometry V3.1*, TUDelft Lecture Slides, 2010.

Additional information

Estimated work time:

$\sim$ 2h Studying theory + $\sim$ 2h making assignment + $\sim$ 2h writing report = $\sim$ 6h

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:

- GEOMETRY2.m
  - SphereGeom.m

The script GEOMETRY2.m computes all the sides and rotation angles using general spherical geometry equation contained in SphereGeom.m.

Listing 1: GEOMETRY2.m: Computes all the angles and sides of a spherical triangle

```
%% By: Simon Billemont, sbillemont, 1387855
%   Contact: aodtorusan@gmail.com or s.billemont@student.tudelft.nl
%       Calculate the sides, angles and area of spherical triangles
%       Based on Orbit & Constellation Design & Management chapter 6
%
%   Made on: 10-10-2010 (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.c_plotsDir = '../images/matlab/';     % Where the plots will go
saver.cleanPlots;                           % Remove any plots already made
saver.c_changec_figSize = 0;                % Do not resize
saver.c_appendFigureNr = 0;                 % Do not append stuff the the name

%% Setup constants

allPoints(1,:,:) = [ 0,  0; 90,  0;  0, 90];
allPoints(2,:,:) = [ 0,  0; 30,  0;  0, 90];
allPoints(3,:,:) = [ 0, 20; 10, 25;  5, 30];
allPoints(4,:,:) = [ 5,  0; 30, 30;  5, 90];
allPoints(5,:,:) = [ 0,-20; 45,-20;  0, 90];
allPoints(6,:,:) = [ 0, 20; 10, 25;  5, 30];
allPoints = allPoints * (pi/180); % Convert to rad

%% Compute GEOMETRY-2

for i=1:size(allPoints,1)
    % Compute unitvectos to P1, P2, P3
    % points = [x1 y1 z1]
    %          [x2 y2 z2]
    %          [x3 y3 z3]
    points =   SphereGeom.sphere2cart(squeeze(allPoints(i,:,:)));

    % Arc lengths:
    sides(1) = SphereGeom.arcLength(points(1,:), points(2,:)); % Side a (A->B)
    sides(2) = SphereGeom.arcLength(points(2,:), points(3,:)); % Side b (B->C)
    sides(3) = SphereGeom.arcLength(points(3,:), points(1,:)); % Side c (C->A)

    % Rotation angles:
    angle(1) = SphereGeom.rotationAngle(points(2,:), points(3,:), points(1,:)); % B->C over A
    angle(2) = SphereGeom.rotationAngle(points(3,:), points(1,:), points(2,:)); % C->A over B
    angle(3) = SphereGeom.rotationAngle(points(1,:), points(2,:), points(3,:)); % A->B over C

    % Area of the sphere
    area = SphereGeom.area(angle);
    if (i==size(allPoints,1))
        angle = 2*pi-angle;
        area =  4*pi-area;
    end

    angle = angle * (180/pi); % Convert to degree for output
    sides = sides * (180/pi); % Convert to degree for output

    % Fancy latex output
    results(i,:) = [i, sides, angle, area];
    fprintf('\t %d & %.2f & %.2f & %.2f & %.2f & %.2f & %.2f & %.2f \\\\ \\hline \n', ...
        results(i,:) );
end
```

Listing 2: SphereGeom.m: Contains general equations for spherical geometry with unit vectors

```
classdef SphereGeom
% SPHEREGEOM Generic forumals for working with spherical triangles (with unit vectors)
%   All methods are static since they are generic forulas : )
% By: Simon Billemont, on: 10-10-2010 (dd-mm-yyyy)
% License: Creative Commons Attribution-NonCommercial 3.0 Unported License.

    methods (Static)
        function cart = sphere2cart(az, el)
            % Duplicate of the matlab cart2sph but for unit vectors
            % Converts aziuth and elevation to x y z
            if nargin==1 % assume vector input
                el = az(:,2);
                az = az(:,1);
            end
            x = cos(az) .* cos(el);
            y = sin(az) .* cos(el);
            z = sin(el);
            cart = [x, y, z];
        end
        function l = arcLength(A, B)
            % Find the angle between two unit vectors (sides of the triangle)
            if nargin==1 % Assume vector input
                B = A(:,2);
                A = A(:,1);
            end
            l = acos(dot(A, B));
        end
        function phi = rotationAngle(A, B, C)
            % Find the angle from A  to B over C
            if nargin==1 % Assume vector input
                C = A(:,3);
                B = A(:,2);
                A = A(:,1);
            end
            % See ocdm p298, table 298 (eq in error) or appendix A eq A-2b
            numerator =   dot(C, cross(A, B));
            denominator = dot(A, B) - dot(C, A) * dot(C, B);
            phi = atan2(numerator,denominator);
        end
        function [A1 A2]=area(sides)
            % Calculate the area of the triagle
            % Gives the inner and outer area
            A1 = sum(sides) - (length(sides)-2)*pi; % Inner area
            A2 = 4*pi - A1;                          % Outer area
        end
    end

end
```