AE4879 Mission Geometry and Orbit Design

Assignment 4: Design
DESIGN-5

Simon Billemont

May 20, 2011

Assignment 4: Design

This assignment deals with investigating how the error budget of a satellite is build up. This is done by investigating tabulated values in [1].

- **Assuming that the individual error sources have errors as mentioned in the 2nd column and that the error propagation is done in an analytical way, find out which relations have been used to propagate these individual errors to a mapping error, and reconstruct these mapping budget contributions as given in the 3rd column. What are the satellite altitude and viewing angle needed to reconstruct these numbers, assuming that we have a flat Earth model? Verify the total geolocation error (bottom value in 3rd column).**
- **Do the same computation, but now introduce an additional systematic (i.e. constant) error of 0.005 degree in the payload sensor mounting in addition to the random errors that are given in OCDM Table 5-18. Propagate this error with the relevant mathematical relation, and treat it in the way discussed in this ppt file (sheets 18, 29,..). What is the new error estimate?**
- **Repeat the two computations as mentioned above, but now with a Monte Carlo technique (i.e. error propagation option 2). Assume that the problem is planar (i.e. 2-dimensional, all errors act in the orbital plane only).**
- **Compare the results of the analytical approach and Monte Carlo approach**

1. Analytical model

The location of the target coordinate "x", can be determined via simple geometric relations:

$$X = SSP + h \tan \eta \tag{1}$$

Errors can occur in each term of this equation. If there are errors in the SSP (and thus the position of the satellite), this error will directly translate in to an equal unknown position of the target. There is thus a one-to-one relation with the position unknowns.

$$\Delta x = \Delta p \tag{2}$$

Related to the positional error, are the timing errors. When the timing of the satellite is off, the timestamp is incorrect, and the satellite has moved somewhat with respect to where it was on the reference timestamp. The offset is based on the satellite satellite velocity and the velocity of the earth. As a worst case scenario these would be aligned (depends on the satellite orbit). Since most satellites fly in polar orbit, these velocities can be assumed to be perpendicular. The following relation between the timing error and mapping error now exists:

$$\Delta x = \Delta t V = \Delta t \sqrt{V_{sat}^2 + V_e^2} \tag{3}$$

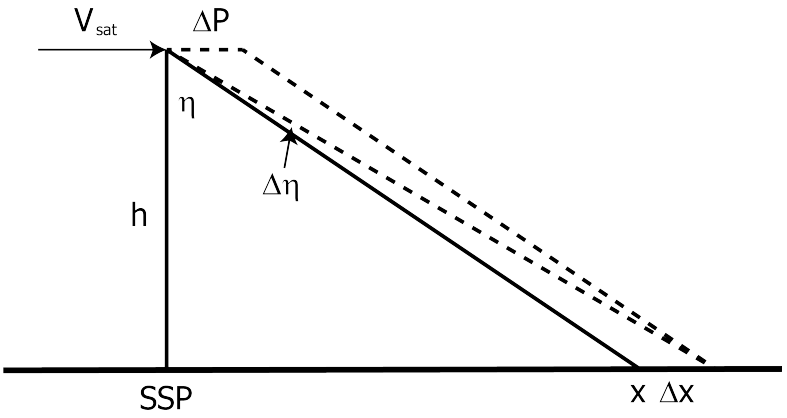

Figure 1: Visualization of the error sources

Where $V_e \approx 464m/s$, see [1]. For the angular error, the basis is the angular term $h \tan \eta$, to find the error contribution, simply multiply by the unknown angle $\Delta\eta$ to find:

$$\Delta x = \Delta\eta \cdot h \tan \eta \tag{4}$$

To retrieve the actual altitude and looking angle of Table 5-8 in [1], two sets of error contributions that contain h and η are needed. Since the positional errors don't include either of these terms they cannot be used. All the angular errors are linearly dependent on each other (dividing the impact on the mapping budget with the assumed error yields a constant), so only one of these can be used. The final equation comes from the timing error. That error source contains an implicit reference to the altitude of the satellite. The set of equations now yield:

$$\begin{cases} \left(0.0020\frac{\pi}{180}\right) h \tan[\eta] = 257.5 & \Rightarrow \eta = 82.2235^\circ \\ \left(50 \cdot 10^{-3}\right) \sqrt{\frac{\mu}{R_e+h}} + 464 \text{ m/s} = 367.5 & \Rightarrow h = 1007.47\text{km} \end{cases} \tag{5}$$

When all the errors of Table 5-8 in [1] are propagated with the computed altitude and looking angle, the following table of results is computed. Since all the errors are assumed to be independent, for the accumulation of the error, the root-sum square has been used. The looking angle η seems large, but this is for a flat earth. When accounting for the curvature of the earth, for a constant elevation, the looking angle goes down dramatically when moving away from the SSP.

Note: the total values where computed using the Root-sum-square: $\sqrt{\sum x^2}$

Furthermore also an additional error can be introduced into the system, namely an additional payload sensor mounting error of 0.005 °. When the modified error sources are computed, table 2 is generated.

It is clear that a additional bias in the payload sensor mounting, increases the error of that system with ~ 750m. When this is combined with the other possible errors, the total error has increased by ~ 205m.

Error source	Error assumed [deg |m |s]	Impact on mapping budget [m]
Star sensor measurement error	0.0015	193.0449
Star sensor mounting error	0.0020	257.3932
Star catalog accuracy	0.0001	12.8697
Attitude computation error	0.0001	12.8697
Payload sensor measerment error	0.0010	128.6966
Target centroiding error	0.0020	257.3932
Payload sensor mounting error	0.0010	128.6966
Transformation of target location to inertial coordinates	0.0001	12.8697
Orbit determination error	100.0000	100.0000
Timing error	0.0500	367.3327
Total pointing error wrt nadir	NaN	590.1903
Projection error due to altitude	700.0000	700.0000
Error in subsatellite point	450.0000	450.0000
Total geolocation error	NaN	1020.2081

Table 1: Resulting mapping error for Case 1 (unbiased)

Error source	Error assumed [deg |m |s]	Impact on mapping budget [m]
Star sensor measurement error	0.0015	193.0449
Star sensor mounting error	0.0020	257.3932
Star catalog accuracy	0.0001	12.8697
Attitude computation error	0.0001	12.8697
Payload sensor measerment error	0.0010	128.6966
Target centroiding error	0.0020	257.3932
Payload sensor mounting error	0.0060	772.1795
Transformation of target location to inertial coordinates	0.0001	12.8697
Orbit determination error	100.0000	100.0000
Timing error	0.0500	367.3327
Total pointing error wrt nadir	NaN	963.3395
Projection error due to altitude	700.0000	700.0000
Error in subsatellite point	450.0000	450.0000
Total geolocation error	NaN	1272.9976

Table 2: Resulting mapping error for Case 2 (biased)

2. Monte Carlo simulation

In order to see how the final error is distributed, and what the main components are, a Monte Carlo simulation was also performed. In this simulation, each unbiased error is randomly picked from a distribution. For the biased parameters (mounting errors and the star catalog accuracy), a constant value was used. In total, per simulation, $5 \cdot 10^4$ samples where used.

The error distribution was assumed to be Gaussian, with a zero mean value and the 1σ value of that tabulated in [1] Table 5-8 as the expected error. For the biased errors, the error was constant over all samples with again the tabulated value. To compute the error due to the offset in the angle η, both the original position and that with the delta are computed. The difference between the two is then the error. For the positional error and the timing error, eq 2-3 where used.

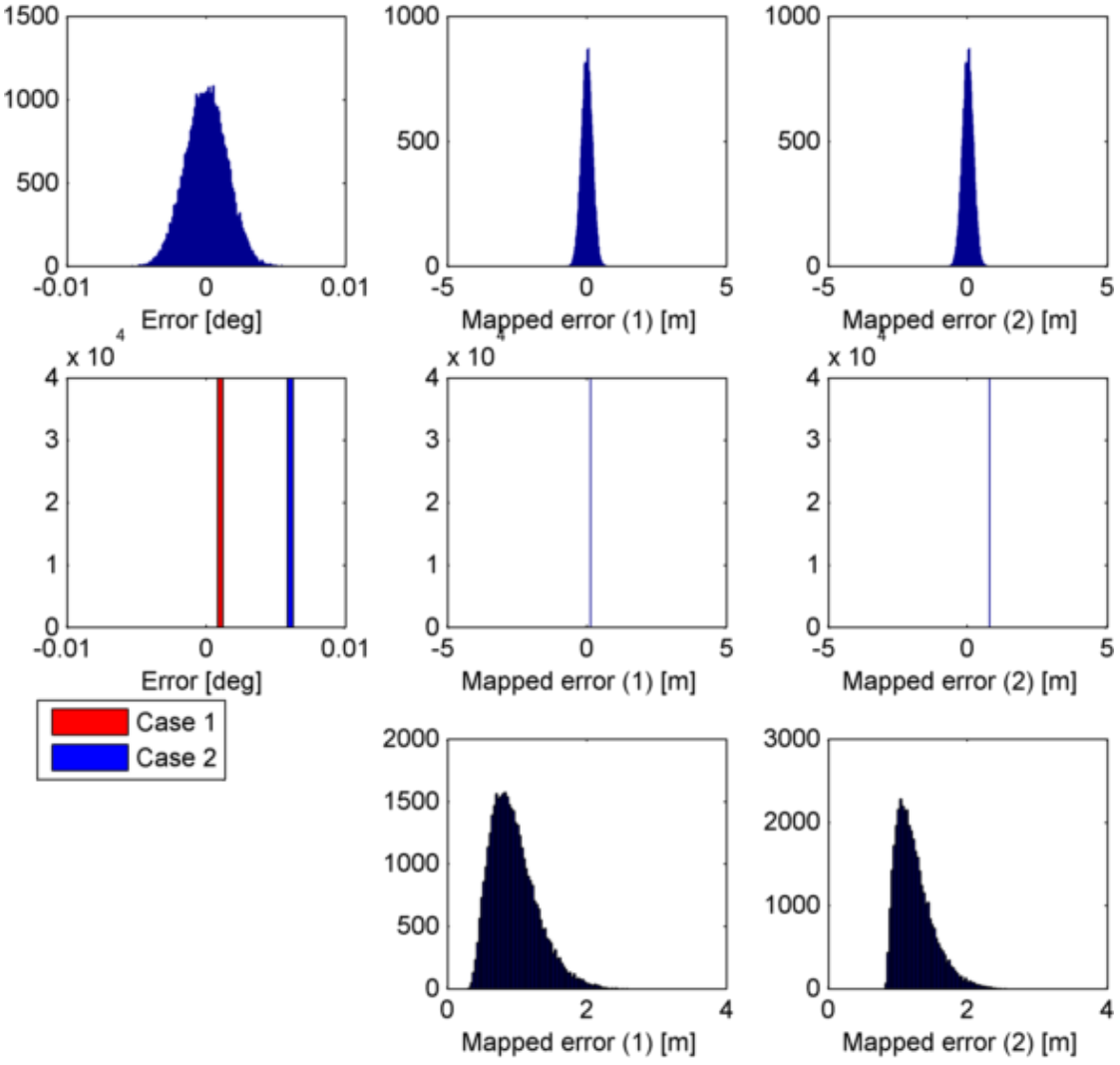

Figure 2:

Figure 2 shows a summary of the results. The first row is associated with the Star sensor measurement error, the second with the Payload sensor mounting error and the last

with the total geolocation error. The first column are the assumed error, the second column is the resulting mapping error distribution without the bias in the payload mounting and the last column is with the bias. All the unbiased parameters behave like the first row, only the spread/size of the error varies per parameter. The biased parameters show a similar behavior to the second row.

What is clear is that due to the RSS, the total resulting error distribution is no longer Gaussian, but more like a shifter Chi-square distribution. This is because the errors are not linearly added, but squared. When comparing with and without the bias in the payload, again the same shift in the results can be seen (of approximately 250 m).

Next we comparing both the analytical and numerical solutions. The top of the distribution lies at about 1km of error. This is the same result as for the analytical solution, however the probability that the error is larger than this value is still very high. It is also feasible that in real life, the error is smaller than this value, as also a large amount of samples is under the computed analytical solution. There is still a significant risk of an error of over 1.5 km. What could be done is state the required accuracy, and compute the % of samples over this value. The resulting value can be used to verify if the system has a high propability of meeting the given requirement. To have a high probability on a very accuracte sytem is hard, because of the long tail in the distribtion, allowing of occational extremely large errors.

Because the Monte Carlo simulation uses equations that describe the system more directly, this approach is more accurate. However, one should take care to take enough samples to completely describe the resulting distribution, what can take a significant amount of resources. The analytical solution is still a very good first order approximation to the error, and has the advantage of giving a direct solution, without all the simulations.

References

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

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

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

Additional information

Estimated work time:

$\sim$ 2h Studying theory + $\sim$ 6h making assignment + $\sim$ 2h writing report = $\sim$ 10h

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)[2] and the source files are given below:

Listing 1: DESIGN5.m: Compute all the steps for the assignment

```
%% By: Simon Billemont, sbillemont, 1387855
%   Contact: aodtorusan@gmail.com or s.billemont@student.tudelft.nl
%   Equations: slides AE4879 Space Mission Design: Design v3.1
%              by Ron Noomen, TUDelft
%   Made on: 19-04-2010 (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 = 1;

%% Projection error budget
table = [0.0015,  193.1, 2;     % 1
         0.0020,  257.5, 2;     % 2
         0.0001,  12.9,  2;     % 3
         0.0001,  12.9,  2;     % 4
         0.0010,  128.8, 2;     % 5
         0.0020,  257.5, 2;     % 6
         0.0010,  128.8, 2;     % 7
         0.0001,  12.9,  2;     % 8
         0.100,   100,   8;     % 9
         0.050,   367.5, 7;     % 10
         NaN,     590.5, NaN;   % 11
         0.700,   700,   8;     % 12
         0.450,   450,   8;     % 13
         NaN,     1020.4,NaN];  % 14
table(1:8, 1) = deg2rad(table(1:8, 1));
table(:, 2)   = table(:, 2) / 1E3;

earth       = constants.bodies.earth; % Properties of the earth
values      = struct();
values.h    = 1007;                   % km
values.eta  = deg2rad(82.2235);       % rad
values.V    = norm(sqrt(earth.mu/(earth.radius+values.h*1E3))*1E-3);% km/s Rotation speed wrt earth surface


%% Questtion a
errors1 = computeErrors(table, values); % Simply compute all the errors
makeTable(errors1, 'table-a', table);   % And make a fancy table

%% Question b
table2 = table; %copy all the errors
table2(7,1) = table2(7,1) + deg2rad(0.005); % add an additional error
errors2 = computeErrors(table2, values); % copute the mapping budget
makeTable(errors2, 'table-b', table2); % tabulate

%% Questtion c

instances = 5E4;                                % Nr of samples
results1 = zeros(instances, size(table,1));     % pre-allocate
results2 = zeros(instances, size(table,1));     % pre-allocate
tables   = zeros(instances, size(table,1));     % pre-allocate

% Compute the error in each parameter
for k=1:14
    switch k
        case {2, 3, 7} % Bias error
            tables(:,k) = ones(instances,1)*table(k,1);
        case {11, 14} % RSS
            tables(:,k) = nan(instances,1);
        otherwise % Random error
            tables(:,k) = normrnd(0,table(k,1),instances,1);
    end
end

% Compute the error for all the instances
for k=1:instances
    % unbiased (case 1)
    currentTable = tables(k,:);
    impacts = computeErrors(currentTable(:), values);
    results1(k,:) = impacts;

    % biased (case 2)
    currentTable(7) = currentTable(7) + deg2rad(0.005);
```

```
    impacts = computeErrors(currentTable(:), values);
    results2(k,:) = impacts;
end

%% Make a fancy plot
plots = [1,7,14];
for i=plots,
    switch i
        case {11, 14}
        case {7}
            subplot(length(plots),3,(find(i==plots)-1)*3+1)
            [n,xout] = hist(rad2deg(tables(:,7)), (-0.008:0.0005:0.008));
            bar(xout, n, 'r')
            hold on
                [n,xout] = hist(rad2deg(tables(:,7))+0.005, (-0.008:0.0005:0.008));
                bar(xout, n, 'b')
            hold off
            xlabel('Error [deg]')
            legend('Case 1', 'Case 2')
        otherwise
            subplot(length(plots),3,(find(i==plots)-1)*3+1)
            hist(rad2deg(tables(:,i)),(-0.008:0.0001:0.008))
            xlabel('Error [deg]')
    end
    subplot(length(plots),3,(find(i==plots)-1)*3+2)
    switch i
        case {11, 14}
            hist(results1(:,i),0.2:0.03:3.2);
        otherwise
            data = results1(:,i); data = data(abs(data) < 2.45); % clamp the plot
            hist(data,-2.45:0.01:2.45);
    end
    xlabel('Mapped error (1) [m]')
    subplot(length(plots),3,(find(i==plots)-1)*3+3)
    switch i
        case {11, 14}
            hist(results2(:,i),0.2:0.03:3.2);
        otherwise
            data = results2(:,i); data = data(abs(data) < 2.45); % clamp the plot
            hist(data,-2.45:0.01:2.45);
    end
    xlabel('Mapped error (2) [m]')
end
```

Listing 2: DESIGN5.m: Compute the mapping error liked to a given error in the parameters

```
function [ impact ] = computeErrors( table, values )
%COMPUTEERRORS Compute all errors from the parameteris in the given table
%   By: Simon Billemont, on: 19-04-2010 (dd-mm-yyyy)
%   License: Creative Commons Attribution-NonCommercial 3.0 Unported License.

    impact = zeros(14,1);
    for i=1:14
        delta = table(i,1);
        switch i
            case num2cell(1:8) % Angle
                impact(i) = delta * values.h * tan(values.eta);
                %impact(i) = values.h * (tan(values.eta + delta) - tan(values.eta));
            case {9, 12, 13} % Position
                impact(i) = delta;
            case 10 % Timing
                impact(i) = delta * values.V;
            case 11 % Rss of the pointing
                impact(i) = rss(impact(1:10));
            case 14 % Total rss
                impact(i) = rss(impact(11:13));
        end
    end

end
```

Listing 3: rss.m: Compute the root sum square of a series of numbers

```
function [ x ] = rss( x )
%RSS Root sum square
%   x = sqrt(sum(x.^2))
%   By: Simon Billemont, on: 19-04-2010 (dd-mm-yyyy)
%   License: Creative Commons Attribution-NonCommercial 3.0 Unported License.

x = sqrt(sum(x.^2));

end
```

Listing 4: makeTable.m: Creates a fancy LaTeXtable

```
function [  ] = makeTable( matrix, file, table )
%MAKETABLE Make a fancy latex table
%   By: Simon Billemont, on: 19-04-2010 (dd-mm-yyyy)
%   License: Creative Commons Attribution-NonCommercial 3.0 Unported License.

latexTable = struct();
latexTable.table.start =     '\\begin{tabular}';
latexTable.table.end =       '\\end{tabular}';
latexTable.table.options =   '{|p{6cm}|p{3cm}|}\\hline';
latexTable.cell.start =      '';
latexTable.cell.end =        ' & ';
latexTable.row.start =       '';
latexTable.row.end =         '\b\b \\\\ \\hline';

if (nargin == 2)
    colLabels = {'Error source', 'Impact on mapping budget [m]'};
else
    latexTable.table.options =   '{|p{6cm}|p{3cm}|p{3cm}|}\\hline';
    colLabels = {'Error source', 'Error assumed [deg \\textbar  m \\textbar  s]', 'Impact on mapping budget [m]'};
end
labels = {'Star sensor measurement error';
          'Star sensor mounting error';
          'Star catalog accuracy';
          'Attitude computation error';
          'Payload sensor measerment error';
          'Target centroiding error';
          'Payload sensor mounting error';
          'Transformation of target location to inertial coordinates';
          'Orbit determination error';
          'Timing error';
          'Total pointing error wrt nadir';
          'Projection error due to altitude';
          'Error in subsatellite point';
          'Total geolocation error'};

matrix = matrix *1E3;
if (nargin == 3)
    matrix = [table(:,1),matrix];
    matrix(1:8, 1) = rad2deg(matrix(1:8, 1));
    matrix([9,12,13], 1) = matrix([9,12,13], 1)*1E3;
end

fid = fopen(['../',file,'.tex'],'w');
fprintf(fid, '%s', sprintf(matrix2ConTeXt(matrix,'$%.4f$', 'table', latexTable, 'colHeaders', colLabels ,'rowHeaders', labels)));
fclose(fid);

end
```