Experiment Design for System Identification on Satellite Hardware Demonstrator

Elias Krantz

Space Engineering, master's level
2018

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
Experiment Design for System Identification on Satellite Hardware Demonstrator

Author: Elias KRANTZ

Examiner: Prof. Reza EMAMI

Supervisor: Dr. Andrei KORNIENKO

Luleå University of Technology

Airbus Defence and Space (Friedrichshafen)

Thesis submitted in fulfilment of the requirements for a degree in Master of Science in Engineering, Space Engineering

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
Sweden

August 24, 2018
Abstract

The subject of this thesis covers the process of online parameter estimation of agile satellites. Accurate knowledge of parameters such as moment of inertia and centre of mass play a crucial role in satellite attitude control and pointing performance. Typically, identification of parameters such as these is performed on-ground using post-processing algorithms. This thesis investigates the potential of performing the identification procedures in real-time on-board operating satellites, using only measurements available from typical satellite attitude sensors.

The thesis covers the areas of system identification and modelling of spacecraft attitude dynamics. An algorithm based on the Unscented Kalman Filter is developed for online parameter estimation of spacecraft moment of inertia parameters. The proposed method is successfully validated, both through simulation environments, and in practice using Airbus’ satellite hardware demonstrator INTREPID, a three-axis air-bearing table equipped with CMG actuators and typical attitude sensors.
Contents

List of Abbreviations v
List of Symbols vi
Conventions vii

1 Introduction 1
  1.1 Motivation and Background .............................................. 1
  1.2 Objectives and Goals .................................................. 2
  1.3 Outline ................................................................. 3

2 System Identification 4
  2.1 About System Identification .............................................. 4
  2.2 Principles of System Identification ..................................... 4
    2.2.1 Tuneable Plant Models ........................................... 5
    2.2.2 Adaptation Algorithms ........................................... 5
  2.3 Methods of Online Parameter Estimation ............................. 6
    2.3.1 The Kalman Filter ................................................ 6
    2.3.2 The Extended Kalman Filter .................................... 8
    2.3.3 The Unscented Kalman Filter ................................... 8
    2.3.4 The Augmented UKF ............................................. 11
    2.3.5 Comparison between EKF and UKF ............................... 12

3 Modelling 14
  3.1 About Modelling ....................................................... 14
  3.2 Spacecraft Dynamics and Kinematics ................................ 14
    3.2.1 Spacecraft Attitude Kinematics ................................. 14
    3.2.2 Spacecraft Attitude Dynamics .................................. 15
  3.3 Hardware Modelling .................................................... 17
    3.3.1 Control Moment Gyroscopes ................................... 17
    3.3.2 INTREPID ....................................................... 18

4 Implementation 22
  4.1 About Implementation ................................................ 22
  4.2 Filter Algorithm ....................................................... 22
    4.2.1 Filter Modifications ............................................ 22
    4.2.2 UASPE for Rigid Spacecrafts .................................. 25
    4.2.3 UASPE for INTREPID ........................................... 26
  4.3 Filter Tuning ........................................................... 26
  4.4 Filter Implementation ................................................ 27

5 Simulation and Experiment Results 29
  5.1 About Simulation and Experiment Results ........................... 29
  5.2 Command Profile ....................................................... 29
Acknowledgements

I would like to thank my supervisor Andrei Kornienko for all support he has given me during my work. He has always been happy to help me and point me in the right direction. This thesis would not have been possible without his guidance.

I would also like to thank Airbus’ AOCS/GNC department for sharing their expertise and including me in their team during my stay with them. Finally, a special thank you to all the students who helped me and gave me a truly amazing time in Friedrichshafen, Germany.
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOCS</td>
<td>Attitude and Orbit Control Systems</td>
</tr>
<tr>
<td>CMG</td>
<td>Control Moment Gyroscope</td>
</tr>
<tr>
<td>CoG</td>
<td>Center of Gravity</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman Filter</td>
</tr>
<tr>
<td>GNC</td>
<td>Guidance and Navigation Control</td>
</tr>
<tr>
<td>MoI</td>
<td>Moment of Inertia</td>
</tr>
<tr>
<td>SysId</td>
<td>System Identification</td>
</tr>
<tr>
<td>S/c</td>
<td>Spacecraft</td>
</tr>
<tr>
<td>UASPE</td>
<td>Unscented Aocs State-Parameter Estimator</td>
</tr>
<tr>
<td>UKF</td>
<td>Unscented Kalman Filter</td>
</tr>
</tbody>
</table>
# List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>Rotation matrix</td>
</tr>
<tr>
<td>B</td>
<td>-</td>
<td>Body frame</td>
</tr>
<tr>
<td>(\vec{a})</td>
<td>-</td>
<td>General vector</td>
</tr>
<tr>
<td>(\vec{g})</td>
<td>-</td>
<td>Gibbs vector</td>
</tr>
<tr>
<td>(h)</td>
<td>kgm(^2)/sec</td>
<td>Angular momentum</td>
</tr>
<tr>
<td>(l)</td>
<td>-</td>
<td>Inertial (fixed) frame</td>
</tr>
<tr>
<td>(J)</td>
<td>kgm(^2)</td>
<td>Moment of inertia</td>
</tr>
<tr>
<td>(K)</td>
<td>-</td>
<td>Kalman gain</td>
</tr>
<tr>
<td>(k)</td>
<td>-</td>
<td>Discrete time step</td>
</tr>
<tr>
<td>(L)</td>
<td>-</td>
<td>Dimension of state-vector</td>
</tr>
<tr>
<td>(M/m)</td>
<td>kg</td>
<td>Mass (major/minor)</td>
</tr>
<tr>
<td>(P)</td>
<td>-</td>
<td>Covariance matrix</td>
</tr>
<tr>
<td>(q)</td>
<td>-</td>
<td>Quaternion</td>
</tr>
<tr>
<td>(r)</td>
<td>m</td>
<td>Distance</td>
</tr>
<tr>
<td>(T)</td>
<td>Nm</td>
<td>Torque</td>
</tr>
<tr>
<td>(\Delta t)</td>
<td>sec</td>
<td>Sampling time</td>
</tr>
<tr>
<td>(u)</td>
<td>-</td>
<td>Model input</td>
</tr>
<tr>
<td>(v)</td>
<td>-</td>
<td>Observation noise</td>
</tr>
<tr>
<td>(W)</td>
<td>-</td>
<td>Weight function</td>
</tr>
<tr>
<td>(w)</td>
<td>-</td>
<td>Process noise</td>
</tr>
<tr>
<td>(x)</td>
<td>-</td>
<td>Model state-vector</td>
</tr>
<tr>
<td>(y)</td>
<td>-</td>
<td>Model observed output</td>
</tr>
<tr>
<td>(\alpha)</td>
<td>-</td>
<td>UKF primary scaling parameter</td>
</tr>
<tr>
<td>(\beta)</td>
<td>-</td>
<td>UKF statistical tuning parameter</td>
</tr>
<tr>
<td>(\delta)</td>
<td>rad</td>
<td>Gimbal angle</td>
</tr>
<tr>
<td>(\theta)</td>
<td>-</td>
<td>Identification parameter</td>
</tr>
<tr>
<td>(\kappa)</td>
<td>-</td>
<td>UKF secondary scaling parameter</td>
</tr>
<tr>
<td>(\lambda)</td>
<td>-</td>
<td>UKF constructed tuning parameter</td>
</tr>
<tr>
<td>(\sigma^2)</td>
<td>-</td>
<td>Covariance (scalar)</td>
</tr>
<tr>
<td>(\Upsilon)</td>
<td>-</td>
<td>Observation sigma point</td>
</tr>
<tr>
<td>(\Phi)</td>
<td>rad</td>
<td>Euler angle</td>
</tr>
<tr>
<td>(\chi)</td>
<td>-</td>
<td>State sigma point</td>
</tr>
<tr>
<td>(\omega)</td>
<td>rad/sec</td>
<td>Angular velocity</td>
</tr>
</tbody>
</table>
# Conventions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_0$</td>
<td>Initial value of variable</td>
</tr>
<tr>
<td>$\hat{x}$</td>
<td>Estimated variable</td>
</tr>
<tr>
<td>$\hat{x}_0$</td>
<td>Initial estimate, $\hat{x}_0 = \mathbb{E}[x_0]$</td>
</tr>
<tr>
<td>$\hat{x}^-$</td>
<td>Priori estimate, i.e. before observation correction</td>
</tr>
<tr>
<td>$\hat{x}^+$</td>
<td>Posteriori estimate, i.e. after observation correction</td>
</tr>
<tr>
<td>$P_{xx}$</td>
<td>Covariance matrix of $x$, $P_{xx} = \mathbb{E}[(x - \hat{x})(x - \hat{x})^T]$</td>
</tr>
<tr>
<td>$P_{xy}$</td>
<td>Cross-covariance matrix of $x$ and $y$, $P_{xy} = \mathbb{E}[(x - \hat{x})(y - \hat{y})^T]$</td>
</tr>
<tr>
<td>$\mathbb{E}$</td>
<td>Expectation operator</td>
</tr>
<tr>
<td>$f()$</td>
<td>State function</td>
</tr>
<tr>
<td>$F()$</td>
<td>Jacobian of $f()$</td>
</tr>
<tr>
<td>$h()$</td>
<td>Observation function</td>
</tr>
<tr>
<td>$H()$</td>
<td>Jacobian of $h()$</td>
</tr>
<tr>
<td>$k$</td>
<td>Current time step</td>
</tr>
<tr>
<td>$k-1$</td>
<td>Previous time step</td>
</tr>
<tr>
<td>$i$</td>
<td>$i$th sigma point, $i = 0, 1, \ldots, 2L$</td>
</tr>
<tr>
<td>$\mathbf{q}$</td>
<td>Four-element quaternion, $\mathbf{q} = [q_0 , \vec{q}]^T$</td>
</tr>
<tr>
<td>$q_0$</td>
<td>Quaternion scalar part</td>
</tr>
<tr>
<td>$\vec{q}$</td>
<td>Quaternion vector part, $\vec{q} = [q_1 , q_2 , q_3]^T$</td>
</tr>
<tr>
<td>$\mathbf{J}$</td>
<td>Vector of moment of inertia elements, $\mathbf{J} = [J_{xx} , J_{yy} , J_{zz} , J_{xy} , J_{xz} , J_{yz}]^T$</td>
</tr>
<tr>
<td>$\chi^x$</td>
<td>&quot;$x$&quot;-part of sigma point</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation and Background

The attitude of a spacecraft describes its orientation in space. Whether its purpose is Earth observation, telecommunication, or space exploration, accurate knowledge and precise control of spacecraft attitude is crucial. It is the responsibility of the Attitude and Orbit Control System, AOCS, to achieve the highest possible performance in these regards.

In recent years, spacecrafts known as agile satellites are becoming increasingly common. These types of spacecrafts are able to reach comparatively high angular rates and accelerations. As a result, modern mission objectives can pose strict requirements of the AOCS’ pointing performance.

Most common AOCS systems follow a control loop similar to the one shown in figure 1.1 below. They make use of both feed-forward, $f_f$, and feed-back, $f_b$, control paths to calculate the spacecraft torque command. The feed-forward path is used to improve system response to pointing commands, while the feed-back, path on the other hand, is used to compensate for perturbations within the control loop. Among all kinds of perturbations affecting spacecrafts in-orbit, the most dominant one comes from modelling uncertainties, i.e. imprecise knowledge of the spacecraft dynamics and parameters. This is especially true when high agility satellites are considered. As a result, more precise modelling improves the accuracy of the feed-forward path and minimizes the burden of the feedback controller, leading to a great overall improvement of AOCS performance.

![Figure 1.1: General control loop used by most AOCS systems, using of both feed-back and feed-forward paths. Controller software (yellow), and physical components (blue).](image)
As most feed-forward controllers are based on a torque calculation to be applied to the spacecraft, the main parameter of interest for attitude control is the Moment of Inertia matrix, $\text{MoI}$. Proper knowledge of the inertia parameters not only improves the pointing accuracy, but also makes the controller more robust and efficient.

The spacecraft MoI is typically estimated on-ground before flight using various analytical and experimental techniques. Analytically, the inertia parameters can be estimated from CAD-drawings (Computer-Aided Design). However, this method rarely gives an accurate enough estimate as many drawings are simplified as volume models of uniform density.

In contrast, experimental techniques uses experiment data to estimate the model parameters. Most of these methods are based on a system identification, $\text{SysId}$, approach. With the increased interest in the field of system identification, the AOCS/GNC department of Airbus Defence and Space have been motivated to develop in-house software tools for SysId within various space domain applications. Current tools contain post-processing methods for SysId, and has already been used in several space missions. This thesis aims to broaden the capabilities with real-time processing methods, and investigate its potential for space applications.[5][7]

### 1.2 Objectives and Goals

The purpose of this thesis is to extend Airbus DS’ SysId capabilities to include methods of real-time in-orbit estimation of spacecraft parameters. The main parameter to be identified is be the moment of inertia matrix. However, other parameters will also be looked at.

For the development and verification of attitude control algorithms for agile satellites, a satellite hardware demonstrator, $\text{INTREPID}$, has been installed at the Airbus site in Friedrichshafen, Germany. This allows the developed method to not only be tested using computer simulations, but also experimentally using real satellite hardware and dynamics.

![Figure 1.2: INTREPID satellite hardware demonstrator.](image-url)
Chapter 1. Introduction

The goal of the thesis are summarized as

1. Literature research on system identification
2. Development of mathematical and simulation models of spacecraft attitude dynamics
3. Implementation of real-time system identification methods for parameter estimation using experimental data
4. Experiment design and experimental validation of developed methods
5. Documentation

1.3 Outline

The structure of this thesis has been considered to give the reader the best possible experience and understanding of the topic and its results. The outline is as follows

2. System Identification  Describes the field of system identification, some of the methods used for real-time estimation, and gives a detailed description of the method of choice in this thesis.
3. Modelling  Development of spacecraft attitude dynamics and kinematics equations, as well as equations concerning INTREPID hardware.
4. Implementation  Development and implementation of the chosen parameter identification method.
5. Simulation and Experiment Results  Summarizes results from simulations and practical experiments.
6. Discussion and Conclusions  Discussion on thesis results, final conclusions, and an outlook on future work in the subject area.
Chapter 2

System Identification

2.1 About System Identification

Constructing models from observed data is an integral part of science. Several methodologies exist for dealing with this very issue. Within control engineering, this topic is often referred to as System Identification. This chapter details the process of system identification and proposes a few methods for real-time identification of spacecraft parameters.

2.2 Principles of System Identification

The purpose of system identification is to find a model that best describes the “true” plant (Physical Plant in the figure 2.1). It usually is unrealistic to achieve a true description of the physical plant, instead, the identified model needs to meet some “goodness” criteria.

Generally described, system identification is the process of using statistical methods to identify dynamical systems from measured input and output data. This allows for parameter identification without directly measuring each system parameter. The general principle of system identification can be seen in figure 2.1 below.

Nominally, an identification experiment is performed where the physical plant is excited in some respect, and input-output data is measured. For our application, the physical plant will be the spacecraft attitude dynamics. It is generally desirable

![Figure 2.1: General principle of the system identification process](image-url)
to design the experiment such that the measured data contain as much information as possible, e.g. contain a wide range of frequencies and amplitudes.

The data is observed using various sensors, which introduce measurement noise. For space applications, these sensors are made up of actuator sensors (measuring applied forces, torques, angular momentums, etc.), and position and attitude sensors (GPS, accelerometers, rate sensors, star trackers, etc.). From noisy measurements an adaptation algorithm is applied, which in turn updates a tuneable plant model.\[8\]

### 2.2.1 Tuneable Plant Models

The choice of plant model that approximates the physical plant plays a fundamental role in the system identification process. It is a prerequisite to choose a suitable model structure before initializing the identification process, and requires some prior understanding of the physical system. The two most common models are

1. **Black-box**  If no prior knowledge of the system dynamics are available, some more general parametric model, known as black-box, as to be applied. These models describe the system in terms of differential equations and/or transfer functions. Many kinds of black-box models are available to pick from, such as ARMAX, Output-Error, and Box-Jenkins. It is usually advantageous to test a few different types to see which one fits the model at hand the best.

2. **Grey-box**  In some circumstances, some prior understanding of the underlying physical dynamics of the system are already known. In these cases a grey-box model can be applied. These models have a predefined structure with model inputs, outputs and states, as well as model parameters. Identifying the parameters of grey-box models is of primary interest in many engineering applications. This process is often referred to as parameter identification or parameter estimation.

For our application a grey-box model will be used as some knowledge of the spacecraft dynamics are already known. Consequently, our identification method will sometimes be referred to as parameter estimation. The dynamical model is developed and discussed in detail in chapter\[8\].

### 2.2.2 Adaptation Algorithms

The adaptation algorithm describes the processing method used for system identification. A large number of processing methods exists for different application purposes. For parameter estimation, these methodologies can be divided into two fields, offline and online parameter estimation.

Offline parameter estimation is performed in a post-processing mode, where all data has been collected during an experiment or training phase. This allows for a superior treatment of measured data, and more complex physical descriptions of the system dynamics. For AOCS applications, offline parameter estimation is performed on-ground before and during spacecraft operation. Examples of some common offline parameter estimation methods are Least-Squares Estimation (LSE), Maximum Likelihood Estimation (MLE), and Instrumental Variables (IV).

As more processing power is available on-ground, offline parameter estimation is often the preferred method. However, in some cases it is necessary to perform the parameter estimation in real-time on-board the spacecraft. This is of particularly high importance when rapidly changing parameters needs to be determined. For
example, during in-orbit operation the spacecraft the moment of inertia may change
as a result of fuel depletion, fuel sloshing, deployment of appendages, or during a
docking manoeuvre. In these cases an online parameter estimation method needs to
be applied.

Other advantages of using an online approach is that it allows for parameter esti-
mation in situations where it is infeasible to transmit measured data to be processed
on-ground. This may be the case for extraterrestrial spacecraft missions. It may even
find usage as a tool to identify anomalous performance, or characterize undeployed
or broken appendages during operation.[7]

2.3 Methods of Online Parameter Estimation

The aim of this thesis is to develop methods for online parameter estimation meth-
ods for space applications. As a result, offline parameter estimation methods fall
outside the scope of this paper. This section describes proposed methods for online
parameter estimation.

2.3.1 The Kalman Filter

One of the most powerful estimation techniques for online parameter estimation
is the Kalman Filter. Named after its inventor Rudolf E. Kálmán, the Kalman Filter
has been around for over half a century. One of its earliest applications was in
NASA’s Apollo programme where it was used for trajectory estimation problems.
Since then, a wide range of filters have been developed which fall under the cat-
egory of Kalman filters. Today it is the most popular estimation technique within
navigation, guidance and control of aircrafts and space vehicles.

As useful as the Kalman Filter is, it has a limitation; it is only applicable to linear
systems. As will be seen in[3] the spacecraft attitude dynamics are highly nonlinear.
For these types of problems, extensions to the standard Kalman filter have been
developed. The most common of these nonlinear Kalman filters are the Extended
Kalman Filter, EKF, and the Unscented Kalman Filter, UKF. Nevertheless, in order to
understand the EKF and UKF, it will be useful to first describe the standard Kalman
filter.

The Kalman filter uses a statistical approach to estimate both the state, \( x \), and
its covariance, \( P_x \). It takes errors produced by process, measurement and estima-
tion noise into account by assuming all noise is Gaussian distributed. Typically,
the Kalman filter is divided into two steps, prediction and correction. During the
prediction step, the state and state-covariance matrix are propagated through linear
dynamics (described by state-space linear transformation matrices, \( A,B,C \)). These
priori estimates are thereafter updated in the correction step by taking the latest
observations into account. The standard algorithm is described below.

Initialization

\[
\begin{align*}
\hat{x}_{\chi,-1}^+ &= \hat{x}_0 = \mathbf{E}[x_0] \\
P_{xx\chi,-1}^+ &= P_{xx,0} = \mathbf{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]
\end{align*}
\] (2.1)
Chapter 2. System Identification

Prediction

\[
\dot{x}_k^- = A \dot{x}_{k-1}^- + B u_{k-1}
\]
\[
P_{xx,k}^- = AP_{xx,k-1}^+ A^T + P_w
\]
\[
\dot{y}_k^- = C \dot{x}_k^-
\]
\[
P_{yy,k}^- = CP_{xx,k-1}^+ C^T + P_v
\]

(2.2)

Correction

\[
K_k = P_{xx,k}^- C^T (P_{yy,k}^-)^{-1}
\]
\[
\dot{x}_k^+ = \dot{x}_k^- + K_k (y_k - \hat{y}_k^-)
\]
\[
P_{xx,k}^+ = P_{xx,k}^- - K_k P_{yy,k}^- K_k^T
\]

(2.3)

Time step update

\[
k = k + 1
\]

Definitions

\[
k \quad \text{is the time step}
\]
\[
x_0 \quad \text{is the initial state}
\]
\[
\dot{x}_0 \quad \text{is the initial state estimate}
\]
\[
\dot{x}_k^- \quad \text{is the priori state estimate}
\]
\[
\dot{x}_k^+ \quad \text{is the posteriori state estimate}
\]
\[
\hat{y}_k^- \quad \text{is the priori observation estimate}
\]
\[
y_k \quad \text{is the current measurement}
\]
\[
u_{k-1} \quad \text{is the input}
\]
\[
P_{xx,0}^- \quad \text{is the initial state estimation error covariance}
\]
\[
P_{xx,k}^- \quad \text{is the priori state estimation error covariance}
\]
\[
P_{xx,k}^+ \quad \text{is the posteriori state estimation error covariance}
\]
\[
P_{yy,k}^- \quad \text{is the priori observation estimation error covariance}
\]
\[
P_w \quad \text{is process noise covariance matrix}
\]
\[
P_v \quad \text{is measurement noise covariance matrix}
\]
\[
K_k \quad \text{is the Kalman gain}
\]

For additional definitions see [List of Symbols] and [Conventions].

The closer the initial conditions, i.e. \( \dot{x}_0 \) and \( P_{xx,0}^- \), are to their real values, the faster the convergence of the filter. Better initial conditions also assures the filter does not diverge. \( P_w \) and \( P_v \), i.e. the process and measurement noise covariances, must also be set by the user before implementation and are usually based on some prior knowledge of the system dynamics.

So far the Kalman Filter is only estimating the states of the system. By extending the state-vector, \( x \), to include the unknown parameters, \( \theta \), the filter can be modified to a state-parameter estimator. This modified Kalman filter requires the state-space transformation matrices, \( A, B, C \), to be re-evaluated at each time step with the changing parameters.\[8\][9]

\[
x_k \rightarrow \begin{bmatrix} x_k \\ \theta_k \end{bmatrix}
\]
\[
A, B, C \rightarrow A_k(\theta_k), B_k(\theta_k), C_k(\theta_k)
\]
2.3.2 The Extended Kalman Filter

As the general Kalman Filter is restricted to linear problems only, some modified versions to the filter have been proposed. The Extended Kalman Filter, \( EKF \), is generally considered the standard filter for nonlinear dynamics, and is the most common online estimator for spacecraft attitude dynamics.

The main difference between the Extended Kalman Filter and the standard filter is in the prediction step. For the EKF, the \( A, B, C \) matrices are replaced by the Jacobian of the nonlinear system. Below follow the general EKF algorithm.

**Initialization**

\[
\hat{x}_{k-1} = 0 = \mathbb{E}[x_0] \\
P_{x_{k-1},x} = \mathbb{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]
\] (2.4)

**Prediction**

\[
\hat{x}_k^- = f(\hat{x}_{k-1}^+, u_{k-1}) \\
P_{x_{k-1},x} = F_{k-1}P_{x_{k-1},x}^+ F_{k-1}^T + P_w \\
\hat{y}_k^- = h(\hat{x}_k^-) \\
P_{y_{k-1},y}^- = H_{k-1}P_{x_{k-1},x}^+ H_{k-1}^T + P_v
\] (2.5)

where

\[
F_{k-1} = \left. \frac{\partial f(x,u_{k-1})}{\partial x} \right|_{\hat{x}_{k-1}^+}, \quad H_{k-1} = \left. \frac{\partial h(x)}{\partial x} \right|_{\hat{x}_{k-1}^+}
\] (2.6)

**Correction**

\[
K_k = P_{x_{k-1},x}^- C^T (P_{y_{k-1},y}^-)^{-1} \\
\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k - \hat{y}_k^-) \\
P_{x_{k-1},x}^+ = P_{x_{k-1},x}^- - K_k P_{y_{k-1},y}^- K_k^T
\] (2.7)

**Time step update**

\[ k = k + 1 \]

Return to **Prediction**

**Definitions**

\( f() \) and \( h() \) are the nonlinear state- and observation functions, and \( F() \) and \( H() \) are the Jacobians of \( f() \) and \( h() \). For additional definitions see [List of Symbols] and [Conventions][2].

2.3.3 The Unscented Kalman Filter

Despite its widespread use, the Extended Kalman Filter carries with it several disadvantages. One of its main drawbacks is its dependency on calculating the first order Jacobians. For highly nonlinear problems, this dependency can be both cumbersome to calculate, and cause the filter to diverge. The Unscented Kalman Filter, \( UKF \), is a derivative free alternative to the EKF, and provides superior performance with comparable computation complexity. The UKF was first proposed by Julier and Uhlmann in 1997, and later developed further by Wan and van der Merwe. The idea behind the filter, as presented by Julier and Uhlmann, is that “it is easier to approximate a probability distribution than it is to approximate and arbitrary nonlinear function or transformation”. [1]
Chapter 2. System Identification

Figure 2.2: Propagation of a Gaussian Random Variable using a first order Taylor approximation versus the Unscented Transformation of the UKF.

The basic difference between EKF and UKF is their treatment of the transformation of the state-vector. In both cases the state distribution is represented as a Gaussian Random Variable, GRV. For the EKF, the GRV is propagated analytically through a first order Taylor approximation of the nonlinear system. This can cause large errors in the propagated mean and covariance. The UKF addresses this issue by using a deterministic sampling approach of the GRV. A minimal set of sample points, commonly referred to as sigma points, are carefully selected such that they completely capture the true mean and covariance of the GRV. When propagated through the nonlinear system, these sigma points will capture the propagated mean and covariance accurately up to the 2nd order for any nonlinear system. Figure 2.2 demonstrates the basic difference in treatment of the GRV between EKF and UKF for a 2-dimensional state-vector.[1][2][5]

The Unscented Kalman Filter adds a few additional steps to the standard Kalman filter. The first step is to calculate the sigma points, \( \chi_i \). The minimal number of sigma points needed is \( 2L + 1 \), where \( L \) is the size of the state-vector. These sigma points are calculated as

\[
\begin{align*}
\chi_0 &= \hat{x}_{k-1}^+ \\
\chi_i &= \hat{x}_{k-1}^+ + \sqrt{(L + \lambda)P_{xx,k-1}^+}i, & i = 1, \ldots, L \\
\chi_i &= \hat{x}_{k-1}^+ - \sqrt{(L + \lambda)P_{xx,k-1}^+}i, & i = L + 1, \ldots, 2L
\end{align*}
\]

where \( (\sqrt{\cdot})i \) is the \( i \)th column of the matrix square root, e.g. a lower triangular Cholesky factorization, and \( \lambda \) is a scaling parameter of the spread of the sigma points around the estimated state-vector. \( \lambda \) is calculated as

\[
\lambda = \alpha^2(L + \kappa) - L
\]
where $\alpha$ control the spread of sigma points around $\hat{x}_{k-1}^+$, and $\kappa$ is a secondary scaling parameter.

Each of these sigma points is propagated through the nonlinear dynamics

$$Y_i = f(\chi_i), \quad i = 0, 1, \ldots, 2L$$

(2.10)

The propagated mean, $\hat{y}$, and covariance, $P_{yy}$, is estimated using weighted sample mean and covariance of the propagated sigma points.

$$\hat{y} = \sum_{i=0}^{2L} W_i^m Y_i$$

(2.11)

$$P_{yy} = \sum_{i=0}^{2L} W_i^c [Y_i - \hat{y}] [Y_i - \hat{y}]^T$$

Where the state weights, $W_i^m$, and covariance weights, $W_i^c$, are calculated as

$$W_0^m = \frac{\lambda}{L + \lambda}$$

$$W_0^c = \frac{\lambda}{L + \lambda} + (1 - \alpha^2 + \beta)$$

$$W_i^m = W_i^c = \frac{1}{2(L + \lambda)}, \quad i = 1, 2, \ldots, 2L$$

(2.12)

The constant $\beta$ is a tuning parameter for incorporating prior knowledge about the distribution of $x$.

Below follow a full description of the UKF algorithm. Figure 2.3 show a block diagram illustration of the algorithm.

**Initialization**

$$\hat{x}_{k-1}^+ = \hat{x}_0 = E[x_0]$$

$$P_{xx,k-1}^+ = P_{xx,0} = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]$$

(2.13)
Chapter 2. System Identification

Calculate sigma points

\[ \chi_{0,k-1} = \hat{x}_{k-1}^+ \]
\[ \chi_{i,k-1} = \hat{x}_{k-1}^+ + \left( \sqrt{(L+\lambda)P_{xx,k-1}^+} \right)_i, \quad i = 1, \ldots, L \]
\[ \chi_{i,k-1} = \hat{x}_{k-1}^+ - \left( \sqrt{(L+\lambda)P_{xx,k-1}^+} \right)_i, \quad i = L + 1, \ldots, 2L \] 

(2.14)

Prediction

\[ \chi_{i,k} = f(\chi_{i,k-1}, u_{k-1}) \]
\[ Y_{i,k} = h(\chi_{i,k}) \]
\[ \hat{x}_k^- = \sum_{i=0}^{2L} W_{im}^k \chi_{i,k} \]
\[ P_{xx,k}^- = \sum_{i=0}^{2L} W_{im}^k [\chi_{i,k} - \hat{x}_k^-] [\chi_{i,k} - \hat{x}_k^-]^T + P_w \]
\[ \hat{y}_k^- = \sum_{i=0}^{2L} W_{im}^k Y_{i,k} \]
\[ P_{yy,k}^- = \sum_{i=0}^{2L} W_{im}^k [Y_{i,k} - \hat{y}_k^-] [Y_{i,k} - \hat{y}_k^-]^T + P_v \]
\[ P_{xy,k}^- = \sum_{i=0}^{2L} W_{im}^k [\chi_{i,k} - \hat{x}_k^-] [Y_{i,k} - \hat{y}_k^-]^T \]

(2.15)

Correction

\[ K_k = P_{xy,k}^- (P_{yy,k}^-)^{-1} \]
\[ \hat{x}_k^+ = \hat{x}_k^- + K_k (y_k - \hat{y}_k^-) \]
\[ P_{xx,k}^+ = P_{xx,k}^- - K_k P_{yy,k}^- K_k^T \]

(2.16)

Time step update

\[ k = k + 1 \]

Return to Calculate sigma points

Definitions

\( \chi_i \) is the \( i \)th sigma point, and \( Y_i \) is the observation of \( \chi_i \). For additional definitions see List of Symbols and Conventions [1][2][3][6]

2.3.4 The Augmented UKF

If it is known how the process and measurement noise enters the system equations, \( f() \) and \( h() \), it is common to use the augmented form of the UKF. In these cases the state-vector and covariance matrix are augmented as

\[ x^a = \begin{bmatrix} x \\ w \\ v \end{bmatrix} \]
\[ P_{xx}^a = \begin{bmatrix} P_{xx} & 0 & 0 \\ 0 & P_w & 0 \\ 0 & 0 & P_v \end{bmatrix} \]

(2.17)

The prediction step is changed as of the UKF is changed to
Prediction (augmented UKF)

\[
\begin{align*}
\chi_{i,k} &= f(\chi_{i,k-1}^x, u_{k-1}, \chi_{i,k-1}^w) \\
Y_{i,k} &= h(\chi_{i,k}^x, \chi_{i,k}^v) \\
\hat{x}_k &= \sum_{i=0}^{2L} W^m_i \chi_{i,k} \\
P_{xx,k} &= \sum_{i=0}^{2L} W^m_i [\chi_{i,k} - \hat{x}_k] [\chi_{i,k} - \hat{x}_k]^T \\
\hat{y}_k &= \sum_{i=0}^{2L} W^m_i Y_{i,k} \\
P_{yy,k} &= \sum_{i=0}^{2L} W^m_i [Y_{i,k} - \hat{y}_k] [Y_{i,k} - \hat{y}_k]^T \\
P_{xy,k} &= \sum_{i=0}^{2L} W^m_i [\chi_{i,k} - \hat{x}_k] [Y_{i,k} - \hat{y}_k]^T
\end{align*}
\]

(2.18)

where \(\chi_{i}^x\) is the state sigma points, and \(\chi_{i}^w\) and \(\chi_{i}^v\) are the process and measurement sigma points.

Though this algorithm may result in a more optimal filter, it greatly increases the computational complexity as it more than doubles the number of sigma points needed. In special cases, where the process and measurement noise is purely additive, the non-augmented UKF can be used. Luckily, this has proven to often the case.[2]

2.3.5 Comparison between EKF and UKF

The Extended Kalman Filter is the most common nonlinear sequential filter within control engineering communities, and has been widely used for the past 40 years. As such it has a large catalogue of implementation examples and strategies. However, it does have some great disadvantages which the Unscented Kalman Filter solves.

The first order linearisation process of the EKF may lead to great estimation errors. This can cause the filter to diverge and ultimately fail. The linearisation process does also require the derivation of Jacobian matrices, which can create significant implementation difficulties. The UKF does not have these disadvantages. By not relying on Jacobians, the UKF can be applied to systems with discontinuous dynamics and black-box models.

Furthermore, the UKF gives accurate estimations for up to the 2nd order (sometimes higher) compared to the EKF which is only accurate to the 1st order. This leaves the UKF always performing equivalent or better than the EKF. Notably, the improvement in performance comes at no additional computational burden as the UKF has close to the same computational complexity as the EKF. This is a crucial point for AOCS applications, where the processing power is usually fairly limited.

Being a somewhat new implementation of the standard Kalman filter, the UKF comes with much less documentation. Consequently, setting the tuning parameters, \(\alpha\), \(\beta\) and \(\kappa\), can be difficult and is not yet fully understood. It has been shown that the choice is generally not critical for state estimation. However, for parameter estimation this can have greater impact and affect the convergence of the filter.

Both filters are limited to cases where the state-vector can be approximated as a Gaussian Random Variable. If this is not a valid approximation, another more
demanding filter has to be used, e.g. particle filters.

As the EKF has been around for a long time, some techniques for dealing with the first-order approximation issues have been developed. The most popular technique is the Iterative Extended Kalman Filter, \textit{IEKF}, which iterates the EKF equation at the current step by redefining the state and re-linearise the observation equations. Though the IEKF is able to perform better than the EKF, a similar technique can theoretically also be applied to the UKF.\[1\][2][5]

Because of the many advantages the UKF carries over the EKF, and without having any significant disadvantages, the chosen parameter estimator in this paper will be based on the UKF. The process and observation noise is assumed purely additive. To minimize the computational complexity, the standard non-augmented UKF will thus be used.
Chapter 3

Modelling

3.1 About Modelling

A crucial step in the system identification process is modelling of the physical plant. In this chapter the spacecraft attitude dynamics and kinematics are derived. As the parameter estimator will be implemented and tested experimentally on a satellite hardware demonstrator, *INTREPID*, the equations of motion will also be extended to include any additional dynamics this may bring.

3.2 Spacecraft Dynamics and Kinematics

3.2.1 Spacecraft Attitude Kinematics

The following is a summary of spacecraft attitude kinematics and quaternion conventions used throughout this paper.

A spacecraft’s orientation in space is most commonly expressed using attitude quaternions, \( q \), as

\[
q = \begin{bmatrix}
q_0 \\
\vec{q}
\end{bmatrix} = \begin{bmatrix}
\cos(\Phi/2) \\
\vec{a} \sin(\Phi/2)
\end{bmatrix}, \quad \vec{q} = \begin{bmatrix}
q_1 \\
q_2 \\
q_3
\end{bmatrix}
\]  

(3.1)

where \( \vec{a} \) is the axis of rotation, or Euler axis, and \( \Phi \) is the angle of rotation, or Euler angle. Note that the scalar part of the quaternion is placed first and named \( q_0 \). Many papers differ in this aspect, putting the scalar part last as \( q_4 \).

As a consequence of its definition (equation 3.1), the attitude quaternion has the constraint of always being of unit length \( q^T q = 1 \).

Rotation of a vector, \( \vec{a} \), from frame A to frame B is denoted as \( q_{BA} \). The rotation is performed using rotation matrix \( A(q_{BA}) \) as

\[
A(q_{BA}) \vec{a}_A = \vec{a}_B
\]

(3.2)

where the rotation matrix is calculated as

\[
A(q) = (q_0^2 - \vec{q}^2) I_3 + 2\vec{q}q^T - 2q_0 [\vec{a}] \times
\]

(3.3)

\( I_3 \) is the 3 \( \times \) 3 identity matrix, and \( [\vec{a}] \times \) denotes the cross-product matrix, defined as

\[
[\vec{a}] \times = \begin{bmatrix}
0 & -a_3 & a_2 \\
 a_3 & 0 & -a_1 \\
-a_2 & a_1 & 0
\end{bmatrix}
\]

(3.4)
Chapter 3. Modelling

Two successive quaternion rotations are found as the quaternion multiplication

$$q'' = q' \otimes q = \begin{bmatrix} q'_{0}q_{0} - 2\vec{q}' \cdot \vec{q} \\
-\vec{q}_{0} \times \vec{q}' + q_{0}\vec{q}' - \vec{q}' \times \vec{q} \end{bmatrix}$$ (3.5)

where $q$ is the first rotation, $q'$ the second, and $q''$ the the total combined rotation.

The quaternion multiplication can also be express as a 4x4 matrix multiplication

$$q' \otimes q = [q'] \otimes q$$ (3.6)

where

$$[q] = \begin{bmatrix} q_{0} \\
\vec{q} \\
q_{0}I_{3} - [\vec{q}]_{x} \end{bmatrix}$$ (3.7)

The inverse of a quaternion is

$$q^{-1} = \begin{bmatrix} q_{0} \\
-\vec{q} \end{bmatrix}$$ (3.8)

such that

$$q \otimes q^{-1} = q^{-1} \otimes q = I_{q}$$ (3.9)

where $I_{q}$ is the identity quaternion

$$I_{q} = \begin{bmatrix} 1 \\
0 \times 1 \end{bmatrix}$$ (3.10)

Finally, the time derivative of a quaternion is

$$\dot{q} = \frac{1}{2} [\omega] \otimes q$$ (3.11)

where

$$[\omega] = \begin{bmatrix} 0 & -\omega^{T} \\
\omega & -[\omega]_{x} \end{bmatrix}$$ (3.12)

Equation (3.11) together with (3.12) will be used by the estimator to propagate the attitude quaternions, i.e. predict the priori quaternions, $q_{k}$. [9][10][11][12]

3.2.2 Spacecraft Attitude Dynamics

The following is a derivation of rigid spacecraft attitude dynamics. Disturbances from non-rigid sources, such as flexible dynamics and fuel sloshing, are outside the scope of this paper.

The equations of angular momentum are

$$h = I\omega$$ (3.13)

$$\ddot{T} = \frac{d}{dt} h = I\dot{\omega}$$ (3.14)
where \( h \) is total angular momentum, \( \omega \) is angular rate, \( \mathbf{T} \) is applied torque, and \( J \) is the Moment of Inertia (MoI) matrix

\[
J = \begin{bmatrix}
J_{xx} & J_{xy} & J_{xz} \\
J_{yx} & J_{yy} & J_{yz} \\
J_{zx} & J_{zy} & J_{zz}
\end{bmatrix}
\] (3.15)

The basic kinematic equation states that the rate change of a vector, \( \mathbf{a} \), as observed in an inertial (fixed) frame equals the rate of change as observed in the rotating frame, plus the cross product \( \omega \times \mathbf{a} \).

\[
\frac{d}{dt} \mathbf{a} \bigg|_I = \frac{d}{dt} \mathbf{a} \bigg|_B + \omega \times \mathbf{a}
\] (3.16)

Applying this equation to the equations of angular momentum, equation 3.13 and 3.14 gives

\[
\mathbf{T} = J \dot{\omega} + \omega \times J \omega
\] (3.17)

This is often referred to as Euler’s moment equation.

For a spacecraft equipped with angular momentum actuators, such as reaction wheels or control moment gyroscopes, the equations of angular momentum, 3.13 and 3.14 gets extended to

\[
h_{tot} = J_s \omega_s/c + h_{act}
\] (3.18)

\[
\mathbf{T}_{ext} = J_s \dot{\omega}_s/c + \dot{h}_{act}
\] (3.19)

Applying equation 3.17 finally gives

\[
\mathbf{T}_{ext} = J_s \dot{\omega}_s/c + \dot{h}_{act} + \omega_s/c \times (J_s \omega_s/c + h_{act})
\] (3.20)

or rearranged

\[
\dot{\omega}_s/c = J_s^{-1} \left[ -\dot{h}_{act} - \omega_s/c \times (J_s \omega_s/c + h_{act}) + \mathbf{T}_{ext} \right]
\] (3.21)

This is the equation of motion of a rigid spacecraft and will be used by the estimator to propagate the angular rate, i.e. predict the priori angular rate, \( \dot{\omega}_s/c \). In this equation \( \omega_s/c \) is the s/c angular rate, \( J_s/c \) is the s/c MoI, \( h_{act} \) is the angular momentum of the s/c actuators, and \( \mathbf{T}_{ext} \) is the total torque from external sources (e.g. disturbances, thrusters, etc.). It will be assumed that no external torques are present, \( \mathbf{T}_{ext} = 0 \). From this equation it is clear that the torque applied by the actuators is

\[
\mathbf{T}_{act} = -\dot{h}_{act}
\] (3.22)

The term \( \omega \times h \) is known as the gyroscopic torque.

\[
\mathbf{T}_{gyro} = -\omega_s/c \times (J_s \omega_s/c + h_{act})
\] (3.23)

The equation of motion of a rigid spacecraft, eq. 3.21 has 6 unknown parameters

\[
\theta_{s/c} = [J_{xx} \ J_{yy} \ J_{zz} \ J_{xy} \ J_{xz} \ J_{yz}]^T
\] (3.24)
Concerning observability, expanding equation 3.17 as three scalar equations

\[ \begin{align*}
T_x &= J_{xx} \dot{\omega}_x + J_{xy} \dot{\omega}_y + J_{xz} \dot{\omega}_z + J_{yx} \dot{\omega}_y \dot{\omega}_y - J_{zy} \dot{\omega}_y \dot{\omega}_z \\
T_y &= J_{xy} \dot{\omega}_x + J_{yy} \dot{\omega}_y + J_{yz} \dot{\omega}_z + J_{yx} \dot{\omega}_x \dot{\omega}_y - J_{yz} \dot{\omega}_y \dot{\omega}_z \\
T_z &= J_{xz} \dot{\omega}_x + J_{yz} \dot{\omega}_y + J_{zz} \dot{\omega}_z + J_{yx} \dot{\omega}_x \dot{\omega}_y - J_{zx} \dot{\omega}_x \dot{\omega}_y 
\end{align*} \] (3.25)

it becomes clear that the major MoI parameters, \( J_{xx}, J_{yy} \) and \( J_{zz} \), are only observable when \( \dot{\omega}_x, \dot{\omega}_y \) and \( \dot{\omega}_z \) are non-zero respectively. That is, in order to identify \( J_{xx} \), the spacecraft has to perform a manoeuvre in \( \dot{\omega}_x \) (i.e. \( \dot{\omega}_x \neq 0 \)).

3.3 Hardware Modelling

3.3.1 Control Moment Gyroscopes

Typically, agile satellites are equipped with Control Moment Gyroscopes, CMG’s, as control torque actuators. These types of actuators are able to provide much higher torques than normal reaction wheel actuators. Reaction wheels provide torque by changing the spin rate of flywheels. CMG’s are in contrast using flywheels which are attached to a gimbal, allowing change in direction of the spin axis. The generally principle is

\[ T_{\text{out}} \propto \vec{h}_{\text{CMG}} \times \vec{\delta}_{\text{CMG}} \]

\[ |T_{\text{out}}| \gg |T_{\text{in}}| \]

where \( T_{\text{in}} \) and \( T_{\text{out}} \) are the input and output torque respectively, \( \vec{h}_{\text{CMG}} \) is the angular momentum of the CMG, and \( \vec{\delta} \) is the gimbal angle with \( \vec{\delta} \) being the gimbal axis. These vectors are defined in figure 3.1.

Typically, the CMG’s are equipped in an array configuration, where the combined angular momentum from all \( n \) CMG’s is

\[ \vec{h}_{\text{CMG}}(\delta) = \sum_{i}^{n} \vec{h}_i(\delta_i) \] (3.26)

and depends on the gimbal angle and the chosen array configuration.
The total torque supplied by all CMG’s is

$$\vec{T}_{\text{CMG}}(\delta, \dot{\delta}) = -\dot{\vec{h}}_{\text{CMG}}(\delta) = \sum_{i} \frac{\partial \vec{h}_{i}(\delta_{i})}{\partial \delta_{i}} \delta_{i} d\delta_{i} = -\mathbf{H}(\delta)\dot{\delta}$$  (3.27)

where $\mathbf{H}(\delta)$ is the $3 \times n$ Jacobian of the angular momentum CMG array.\[7\]

The main advantage of using CMG actuators is the great increase in output torque that can be supplied by the actuators. They are, however, restricted by their complexity and the possibility of reaching singularities, i.e. states where the CMG array cannot provide torque in all directions. These singularities add extra constraints on the AOCS controller.

### 3.3.2 INTREPID

The satellite hardware demonstrator INTREPID is an air-bearing table with three rotational degrees of freedom, equipped with typical satellite sensors, actuators, and processing units. The table can rotate unrestrictedly about the vertical axis, and tilt up to $\sim \pm 30$ degrees in the horizontal plane. The air-bearing allows for close to frictionless motion, simulating the motion of a satellite in-orbit.

The platform is equipped with four CMG actuators, an on-board computer, and attitude and rate sensors (i.e. a star tracker and inertia measurement units). The setup can be powered either by using an external power source or by using a battery pack mounted on the platform. The latter case allows the platform motion to be almost completely free from any external disturbances. The setup is seen in figure 3.2. The body reference frame is positioned at the centre of rotation, with the axes being aligned as demonstrated in the figure.

Being a ground based testbed, some additional dynamics are present. The Centre of Gravity, CoG, is not perfectly aligned with the centre of rotation of the air-bearing.
This results in an external gravitational torque, $\vec{T}_g(q_{BI})$, acting on the table.

$$\vec{T}_g(q_{BI}) = M \vec{r}_{CoG} \times \left( A(q_{BI}) \vec{g} \right)$$

(3.28)

where $M$ is the mass of the table, $\vec{r}_{CoG}$ is the CoG offset, $A(q_{BI})$ is the rotation matrix given by equation 3.3 from inertial (I) to body (B) frame, and $\vec{g}$ is the gravity vector

$$\vec{g} = \begin{bmatrix} -g_0 \\ 0 \\ 0 \end{bmatrix}$$

(3.29)

The CoG of INTREPID is unknown, resulting in three additional parameters that has to be identified, i.e. the elements of $\vec{r}_{CoG}$. However, from equation 3.28 it is clear that the effect of $\vec{r}_{CoG}$ on the gravitational torque is linearly dependant on the mass of INTREPID, $M$. Thus the additional estimation parameter becomes $M\vec{r}_{CoG}$.

The table is also equipped with a mass-balancing system. This system consists of four mass-balancing blocks on rails, allowing translational movement of the CoG of INTREPID. This allows for the gravitational torque to be controlled, and can simulate the effects of using magnetorquers. These rails are aligned with the lateral axes of the table, with an inclination of 17 degrees in respect to the horizontal plane.

The position of each mass-balancing block, $\vec{r}_i$, is given by

$$\vec{r}_i = \vec{r}_{i,0} + A_{BU,i} \begin{bmatrix} \Delta d_i \\ 0 \\ 0 \end{bmatrix} \quad i = 1, \ldots, 4$$

(3.30)

where $\vec{r}_{i,0}$ is the initial position block $i$, and $\Delta d_i$ is the moved distance of block $i$ along the track. $A_{BU,i}$ is the rotation matrix from unit frame (U) to body frame (B) as

$$A_{BU,i} = \left[ A_y(\beta) A_z \left( \frac{\pi}{2} (3 - i) \right) \right]^T \quad i = 1, \ldots, 4$$

(3.31)
where $\beta$ is the rail tilt angle ($\beta = 17$ deg), and $A_x(\theta)$, $A_y(\theta)$, $A_z(\theta)$ are the principal rotation matrices defined as

$$
A_x(\theta) = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(\theta) & -\sin(\theta) \\
0 & \sin(\theta) & \cos(\theta)
\end{bmatrix}
$$

$$
A_y(\theta) = \begin{bmatrix}
\cos(\theta) & 0 & \sin(\theta) \\
0 & 1 & 0 \\
-\sin(\theta) & 0 & \cos(\theta)
\end{bmatrix}
$$

$$
A_z(\theta) = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) & 0 \\
\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1
\end{bmatrix}
$$

The effect the mass-balancing system has on INTREPID’s total CoG will be excluded from the estimated CoG. The total CoG used in equation 3.28 is

$$
M_{tot}\vec{r}_{tot} = M\vec{r}_{CoG} + 4m \sum_i \vec{r}_i
$$

where $m$ is the mass of each mass-balancing block ($m = 0.5$ kg).

The CMG array of INTREPID can be in either ‘pyramid’ or ‘roof’ configuration. During the experiment the roof configuration will be used. Figure 3.4 shows the definition of the roof array configuration. Using this definition, the angular momentum, $\vec{h}_{CMG}$, and angular momentum rate, $\dot{\vec{h}}_{CMG}$, for each CMG in body frame (B) is

$$
\vec{h}_{CMG,i} = A_{BU,i}A_z(\delta_i) \begin{bmatrix}
h_{FW} \\
0 \\
0
\end{bmatrix}
$$

$$
\dot{\vec{h}}_{CMG,i} = A_{BU,i}A_z(\delta_i)\dot{\delta}_i \begin{bmatrix}
0 \\
h_{FW} \\
0
\end{bmatrix}
$$

where $\delta$ and $\dot{\delta}$ are the gimbal angle and gimbal rate respectively, $h_{FW}$ is the angular momentum of the CMG fly-wheel (assumed equal for all CMG’s). The rotation matrix from unit (U) to body (B) reference frame, $A_{BU,i}$, for each CMG is

$$
A_{BU,i} = (A_{UA,i}A_{AB})^T 
$$

where $A_{AB}$ is the rotational matrix from body (B) to array (A) frame

$$
A_{AB,i} = A_x\left(\frac{3\pi - \beta}{2}\right)A_z\left(\frac{3\pi}{2}\right)
$$

and $A_{UA,i}$ is the rotational matrix from array (A) to unit (U) frame

$$
A_{UA,i} = A_z\left(\frac{\pi}{2}\right)A_x(\pi)A_z\left(\frac{\pi}{2}\right), \quad i = 1, 4
$$

$$
A_{UA,i} = A_z\left(\frac{\pi}{2}\right)A_y(-\beta)A_z\left(\frac{\pi}{2}\right), \quad i = 2, 3
$$
Finally, the equation of motion for the INTREPID hardware demonstrator, expressed as in equation 3.21 becomes

\[ \dot{\omega}_{\text{Int}} = J_{\text{Int}}^{-1} \left[ -h_{\text{CMG}} - \omega_{\text{Int}} \times (J_{\text{Int}}\omega_{\text{Int}} + h_{\text{CMG}}) + \bar{T}_g(q_{BI}) \right] \]  

(3.39)

with a total of 9 unknown parameters

\[ \theta_{\text{Int}} = [J_{xx} \ J_{yy} \ J_{zz} \ J_{xy} \ J_{xz} \ J_{yz} \ M_{rx} \ M_{ry} \ M_{rz}]^T \]  

(3.40)
Chapter 4

Implementation

4.1 About Implementation

To identify the unknown parameters an estimator based on the Unscented Kalman Filter will be implemented. The estimator developed in this chapter will further be known as the Unscented Aocs State-Parameter Estimator, UASPE.

4.2 Filter Algorithm

4.2.1 Filter Modifications

The UASPE estimator is based on the Unscented Kalman Filter algorithm described in 2.3.3 with some modifications. First, the estimator will work in a joint state-parameter mode. That is, it will estimate both the states and the parameters simultaneously by treating the parameters as part of the state-vector. Figure 4.1 show a block diagram of the general estimation loop to be used by UASPE.

The combined state-parameter vector, $x$, is

$$x = \begin{bmatrix} q \\ \omega \\ \mathbf{J} \end{bmatrix}$$

(4.1)

with

$$\mathbf{J} = \begin{bmatrix} J_{xx} & J_{yy} & J_{zz} & J_{xy} & J_{xz} & J_{yz} \end{bmatrix}^T$$

(4.2)

The input, $u$, is the total actuator angular momentum, $h_{act}$, and torque, $T_{act}$, and is defined as

$$u = \begin{bmatrix} h_{act} \\ T_{act} \end{bmatrix}$$

(4.3)

For prediction of priori attitude quaternion, $\hat{q}_k$, and angular rate, $\hat{\omega}_k$, equation 3.11 and 3.21 are used. The time-derivative of estimation parameters is zero.

$$\dot{q} = \frac{1}{2} [\omega \otimes] q$$

$$\dot{\omega} = J^{-1} \left[ \hat{T}_{act} - \omega \times (J\omega + h_{act}) \right]$$

(4.4)

$$\dot{\mathbf{J}} = 0$$
Figure 4.1: Block diagram of the Unscented Aocs State-Parameter Estimator, UASPE.
Which gives the propagation equations in discrete time

\[
\mathbf{q}_k = f_q(x_{k-1}, \theta_{k-1}, u_{k-1}) = \mathbf{q}_{k-1} + \Delta t \frac{1}{2} [\omega_{k-1} \otimes \mathbf{q}_{k-1}]
\]

\[
\omega_k = f_\omega(x_{k-1}, \theta_{k-1}, u_{k-1}) = \omega_{k-1} + \Delta t f^{-1}_k \left[ T_{act,k-1} - \omega_{k-1} \times (J_{k-1} \omega_{k-1} + h_{act,k-1}) \right]
\]

\[
\mathbf{J}_k = f_J(x_{k-1}, \theta_{k-1}, u_{k-1}) = \mathbf{J}_{k-1}
\]

(4.5)

where \( \Delta t \) is the sampling time.

The observed output, \( y \), is the measured attitude and angular rate of the spacecraft.

\[
y = \begin{bmatrix} \mathbf{q} \\ \omega \end{bmatrix}
\]

(4.6)

Thus, the observation equations are simply

\[
y_k = h(x_k, \theta_k) = \begin{bmatrix} \mathbf{q}_k \\ \omega_k \end{bmatrix}
\]

(4.7)

However, there is an issue with directly implementing the UKF using attitude quaternions. The UKF predicts the states using a weighted average, which in the case of a four-element quaternion will not always produce a quaternion of unit length.[6] To work around this, a three-element Gibbs-vector representation of the error quaternion, \( \delta \mathbf{g} \), shall be used for the state-vector.

\[
x = \begin{bmatrix} \delta \mathbf{g} \\ \omega \\ \mathbf{J} \end{bmatrix}
\]

(4.8)

This workaround follows the algorithm described by Crassidis[3]. The benefit of using a Gibbs-vector representation rather than the attitude quaternion is that the state-vector will be free from constraints. However, this will require a number of extra steps in the UKF algorithm.

The error-quaternion, \( \delta \mathbf{q} \), describes the error in estimated quaternion and is defined as

\[
\delta \mathbf{q} = \hat{\mathbf{q}} \otimes \mathbf{q}^{-1}
\]

(4.9)

where \( \hat{\mathbf{q}} \) is the estimated quaternion, and \( \mathbf{q} \) is the true quaternion.

The Gibbs-vector is defined as

\[
\delta \mathbf{g} = 2 \frac{\delta \mathbf{q}}{\delta q_0}
\]

(4.10)

It will prove useful to also have the transform back to the error-quaternion

\[
\delta q_0 = \frac{2 \sqrt{4 - ||\delta \mathbf{g}||^2}}{4 + ||\delta \mathbf{g}||^2}
\]

\[
\delta \mathbf{q} = \frac{\delta q_0}{\delta \mathbf{g}}
\]

(4.11)
With the assumption that the estimated posteriori attitude quaternion, $\hat{q}_{k}^{+}$, is accurate, at each time step it will be necessary to do a state reset.

$$\delta_{\hat{q}_{k-1}} = 0 \quad \Rightarrow \quad \hat{x}_{k-1}^{+} = \begin{bmatrix} 0_{3 \times 1} \\ \hat{\omega}_{k-1}^{+} \\ \hat{f}_{k-1}^{+} \end{bmatrix}$$  \hspace{1cm} (4.12)

To calculate the propagated priori Gibbs-vector, the following additional steps are needed during the prediction.

1. Transform Gibbs-vector to attitude quaternion using equation 4.9 and 4.11 for each sigma point, $\chi_{i}$.

$$\chi_{\delta q_{0}}^{\delta q_{i,k-1}} = \frac{2\sqrt{4 - \|\chi_{\delta q_{i,k-1}}^{\delta q_{0}}\|^{2}}}{4 + \|\chi_{\delta q_{i,k-1}}^{\delta q_{0}}\|^{2}}$$  \hspace{1cm} (4.13)

2. Propagate attitude quaternion using equation 4.5

$$\chi_{i,k}^{q} = \chi_{i,k-1}^{q} + \Delta t \frac{1}{2} \left[ \chi_{i,k-1}^{\omega} \otimes \chi_{i,k-1}^{q} \right]$$  \hspace{1cm} (4.14)

3. Transform back to Gibbs-vector using equation 4.9 and 4.10

$$\chi_{\delta q_{0}}^{\delta q_{i,k}} = \chi_{\delta q_{i,k}}^{q} \otimes \left( \chi_{\delta q_{0}}^{q} \right)^{-1}$$  \hspace{1cm} (4.15)

Finally, in the correction step of the filter, the estimated posteriori Gibbs-vector, $\delta_{\hat{g}_{k}}^{+}$, need to be transformed back to the estimated attitude quaternion, $\hat{q}_{k}^{+}$, using equation 4.9 and 4.11

$$\delta_{q_{0,k}}^{+} = \frac{2\sqrt{4 - \|\delta_{\hat{q}_{k}}^{+}\|^{2}}}{4 + \|\delta_{\hat{q}_{k}}^{+}\|^{2}}$$

$$\delta_{\hat{q}_{k}}^{+} = \delta_{\hat{q}_{0,k}}^{+} \otimes Y_{0,k}^{q}$$  \hspace{1cm} (4.16)

### 4.2.2 UASPE for Rigid Spacecrafts

With these modifications, the algorithm for UASPE can be fully described. A complete description of the estimation algorithm for MoI estimation of a rigid spacecraft can be found in Appendix A.1
4.2.3 UASPE for INTREPID

To implement the filter with INTREPID for the experimental tests, a few more changes are needed to be made to the previously described algorithm. First, the CoG-parameters, $\vec{r}_{\text{CoG}}$, of INTREPID are added to the state-parameter vector.

$$x = \begin{bmatrix} \delta \vec{g} \\ \omega \\ M \vec{r}_{\text{CoG}} \end{bmatrix}$$  \hspace{1cm} (4.17)

Second, the gravitational torque need to be calculated and added to the propagation of $\chi^\omega$. From equation 3.28 and 3.39

$$\vec{T}_g(\chi^q_{k-1}, \chi^\text{MR}_{k-1}) = \chi^\text{MR}_{k-1} \times \left( A(\chi^q_{k-1}) \vec{g} \right)$$

$$\chi^\omega_k = \chi^\omega_{k-1} + \Delta t (\chi^J_{k-1})^{-1} \left[ \vec{T}_g(\chi^q_{k-1}, \chi^\text{MR}_{k-1}) + \vec{T}_{\text{CMG},k-1} - \chi^\omega_{k-1} \times (\chi^J_{k-1} \chi^\omega_{k-1} + h_{\text{CMG},k-1}) \right]$$  \hspace{1cm} (4.18)

Last, to account for the effect of the mass-balancing system, an additional input is needed; the combined CoG of the mass-balancing blocks, $\vec{r}_{\text{MB}}$.

$$u = \begin{bmatrix} h_{\text{CMG}} \\ \vec{T}_{\text{CMG}} \\ \vec{r}_{\text{MB}} \end{bmatrix}$$  \hspace{1cm} (4.19)

From equation 3.33 the total CoG of the system is

$$\chi^\text{MCoGtot} = \chi^\text{MR}_{\text{CoG}} + 4m \vec{r}_{\text{MB}}$$  \hspace{1cm} (4.20)

With this, the complete algorithm for MoI and CoG estimation of INTREPID can be described. The UASPE for INTREPID algorithm can be found in Appendix A.2.

4.3 Filter Tuning

Before implementing the algorithm, the filter must be parametrised. The following is a list of all UASPE tuning parameters, with some guidelines on tuning.

**Primary scaling parameter, $\alpha$**
Primary scaling parameter $\alpha$ determines the spread of the sigma points around the mean and is usually set to a small positive value, e.g. $1e^-4 \leq \alpha \leq 1$.

**Statistical tuning parameter, $\beta$**
Statistical tuning parameter $\beta$ integrates priori knowledge of the distribution of the state. For Gaussian distributions $\beta = 2$ is optimal.

**Secondary scaling parameter, $\kappa$**
Secondary scaling parameter $\kappa$ is a secondary scaling parameter. Usually set to 0 or $3 - L$, where $L$ is the size of the state-parameter vector.

**Process noise covariance, $P_w$**
Process noise covariance, $P_w$, incorporates additive disturbances to the propagation...
model as
\[ x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \]
where \( w_{k-1} \) is the process noise. These disturbances can come from external sources, actuators, or numerical errors, and may not be well-known. Generally larger covariance leads to slower state-parameter convergence, but also a more robust estimator.

**Measurement noise covariance, \( P_v \)**
Measurement noise covariance, \( P_v \), incorporates additive disturbances to the observation model as
\[ y_k = h(x_{k-1}) + v_{k-1} \]
where \( v_{k-1} \) is the measurement noise. It is thus recommended to simply set \( P_v \) to the expected sensor noise, which is usually more or less known.

**Initial state-parameter vector, \( \hat{x}_0 \)**
Initial state-parameter vector, \( \hat{x}_0 \), includes the initial states and parameters used by the estimator. More precise initial value leads to faster convergence of the estimator, but is in general less critical than other parameters.

**Initial estimation covariance matrix, \( P_{xx,0} \)**
Initial estimation covariance matrix, \( P_{xx,0} \), includes the initial estimation error covariance of the state-parameter vector. It is recommended to set it 3 times higher than the expected error to include 99.7% of the dynamics.
\[ P_{xx,0} = 3^2 \mathbb{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T] \]
Tuning recommendations of \( \alpha, \beta \) and \( \kappa \) are given by Wan and van der Merwe.\(^4\)

### 4.4 Filter Implementation

The filter is implemented in Matlab/Simulink as a masked Simulink block. This block is designed to be user-friendly and easy to implement in any AOCS-loop, with the possibility of being added to Airbus’ standard library for AOCS system identification. The estimator further contain user manual, help-files, initialisation scripts, and several demo examples of implementation and configuration. Figure 4.2 demonstrates a UASPE estimator block with an easy-to-understand dialogue box for filter tuning.

The only block inputs are measurements from actuators and relevant sensors, i.e. \( u_{\text{meas}} = [h_{\text{CMG}} \, \tilde{T}_{\text{CMG}} \, \tilde{r}_{\text{MB}}] \), and \( y_{\text{meas}} = [\mathbf{q} \, \omega] \). The estimator output is simply the estimated attitude, states, parameters and diagonal elements of the covariance matrix. The complete Simulink implementation can be found in Appendix 3.
To use the estimator with INTREPID, it is also needed to calculate the actuator inputs. This is performed outside the filter with two additional Simulink blocks. The first block calculates the combined CMG torque and angular momentum, $\vec{T}_{CMG}(\delta, \dot{\delta})$ and $\vec{h}_{CMG}(\delta)$, from measured gimbal angles and rates, $\vec{\delta}_{1:4}$ and $\dot{\vec{\delta}}_{1:4}$, using equation 3.22 and 3.34-3.38. The second block calculates the combined CoG of the mass-balancing blocks, $\vec{r}_{MB}$, from measured translated block distances, $\Delta \vec{d}_{1:4}$, using equation 3.30-3.31. The complete Simulink implementation of these two blocks can be found in Appendix C.
Chapter 5

Simulation and Experiment Results

5.1 About Simulation and Experiment Results

This chapter summarizes the results from simulations and experimental verification of the parameter estimator. Analysis and discussion is left for the next chapter.

5.2 Command Profile

To be able to correctly identify the unknown parameters, the spacecraft must be given a command profile to follow. This profile must be designed such that all parameters are observable, and must be sufficiently exciting to make them converge within a minimal time. Additionally, it must also comply with any limitations of the actuators (e.g. maximum torque output), and any pointing constraints the spacecraft may have (e.g. to protect sun sensitive instruments).

In the case of INTREPID, the guidance profile is limited to a maximum tilting angle of \( \pm 30 \) degrees of the horizontal plane. To identify the MoI parameters in all axes, INTREPID will sequentially tilt \( \pm 10 \) degrees around each body axis with relatively high angular rates (>3 deg/sec). This command profile will also be used for the spacecraft simulation. The profile used for experiments and simulation can be seen in figure 5.2.

![INTREPID Satellite Hardware Demonstrator](image)

**Figure 5.1:** INTREPID satellite hardware demonstrator.
5.3 Simulation

Before implementing the algorithm on the experimental case, it was validated in a simulated environment. An agile satellite equipped with CMG’s was simulated in Matlab/Simulink using a simulation environment provided by Airbus’ AOCS/GNC department. The simulation included measurement noise ($\sigma_q^v$ and $\sigma_\omega^v$), actuator noise ($\sigma_T^w$ and $\sigma_h^w$), CMG dynamics, transport delays, and discrete sampling time ($\Delta t$) equal to that of INTREPID (20Hz). The sensor noise was chosen to be similar to the expected noise of INTREPID.

The complete simulation ran for 1500 seconds. Table 5.1 show the spacecraft parameter values estimated by the filter at the end of the simulation. Figure 5.3 - 5.6 show how the estimated parameter values change over time during the first 300 seconds of the simulation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>True value</th>
<th>Assumed value</th>
<th>Estimated value</th>
<th>Estimation error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{xx}$ [kgm$^2$]</td>
<td>2600</td>
<td>3000</td>
<td>2599</td>
<td>0.058%</td>
</tr>
<tr>
<td>$J_{yy}$ [kgm$^2$]</td>
<td>1800</td>
<td>2000</td>
<td>1790</td>
<td>0.53%</td>
</tr>
<tr>
<td>$J_{zz}$ [kgm$^2$]</td>
<td>1800</td>
<td>2000</td>
<td>1802</td>
<td>0.11%</td>
</tr>
<tr>
<td>$J_{xy}$ [kgm$^2$]</td>
<td>0</td>
<td>100</td>
<td>-3.81</td>
<td>NaN</td>
</tr>
<tr>
<td>$J_{xz}$ [kgm$^2$]</td>
<td>0</td>
<td>100</td>
<td>-0.240</td>
<td>NaN</td>
</tr>
<tr>
<td>$J_{yz}$ [kgm$^2$]</td>
<td>0</td>
<td>100</td>
<td>1.09</td>
<td>NaN</td>
</tr>
</tbody>
</table>

Figure 5.2: Guidance profile used for INTREPID parameter estimation.
**Chapter 5. Simulation and Experiment Results**

**Figure 5.3:** Estimated diagonal elements of s/c MoI over time. Red line marks the start of the first manoeuvre around relevant body axis (top to bottom: $\omega_\alpha$, $\omega_\beta$, $\omega_\gamma$-manoeuvre).

**Figure 5.4:** Estimation error of diagonal elements of s/c MoI over time (error = $\hat{\theta} - \theta$), with $\pm 3\sigma$-bounds.
5.4 Experiment

Once the performance of the estimator had been shown to yield satisfying results in a simulated environment, it was tested in practice using the satellite hardware demonstrator INTREPID. INTREPID followed the guidance profile seen previously in figure 5.2. The experiment ran for a total of 850 seconds.

Table 5.2 show the parameter values estimated by the filter after a full experiment run on INTREPID, and table 5.3 summarises the parametrisation the filter used. Figure 5.7 - 5.12 show how the estimated parameter values changed over time. The final graph, figure 5.13, show how the norm of the estimation error evolved over time.
time, and is defined as

$$||error||_2 = \frac{||\hat{\theta} - \tilde{\theta}||_2}{||\tilde{\theta}||_2} \quad (5.1)$$

where $\hat{\theta}$ is the estimated parameter-vector and $\tilde{\theta}$ is the true parameter-vector.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Initial value</th>
<th>Estimated value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{xx}$ [kgm²]</td>
<td>20</td>
<td>26.2</td>
</tr>
<tr>
<td>$J_{yy}$ [kgm²]</td>
<td>20</td>
<td>25.2</td>
</tr>
<tr>
<td>$J_{zz}$ [kgm²]</td>
<td>30</td>
<td>35.1</td>
</tr>
<tr>
<td>$J_{xy}$ [kgm²]</td>
<td>0</td>
<td>-1.76</td>
</tr>
<tr>
<td>$J_{xz}$ [kgm²]</td>
<td>0</td>
<td>-1.38</td>
</tr>
<tr>
<td>$J_{yz}$ [kgm²]</td>
<td>0</td>
<td>0.929</td>
</tr>
<tr>
<td>$Mr_x$ [kgm]</td>
<td>0</td>
<td>-0.0029</td>
</tr>
<tr>
<td>$Mr_y$ [kgm]</td>
<td>0</td>
<td>0.0021</td>
</tr>
<tr>
<td>$Mr_z$ [kgm]</td>
<td>-0.5</td>
<td>-0.544</td>
</tr>
</tbody>
</table>

Figure 5.7: Estimated diagonal elements of INTREPID MoI over time. Red line marks the start of the first manoeuvre around relevant body axis (top to bottom: $\omega_x$, $\omega_y$, $\omega_z$ -manoeuvre). Dotted line marks final value after 850 seconds.
**Figure 5.8:** Estimation error of diagonal elements of INTREPID MoI over time (error = \( \hat{\theta} - \theta \)), with \( \pm 3\sigma \)-bounds.

**Figure 5.9:** Estimated off-diagonal elements of INTREPID MoI over time. Dotted line marks final value after 850 seconds.
Chapter 5. Simulation and Experiment Results

Figure 5.10: Estimation error of off-diagonal elements of INTREPID Mol over time (error = \( \hat{\theta} - \theta \)), with \( \pm 3\sigma \)-bounds.

Figure 5.11: Estimated INTREPID CoG over time. Dotted line marks final value after 850 seconds.
Chapter 5. Simulation and Experiment Results

Figure 5.12: Estimation error of INTREPID CoG over time (error = $\hat{\theta} - \theta$), with ±3σ-bounds.

Figure 5.13: Norm of estimation error of INTREPID parameters, defined by equation 5.1.
TABLE 5.3: Parametrisation of the UASPE estimator for the INTREPID experiment.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta t$</td>
<td>0.05</td>
<td>sec</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1e-3</td>
<td>-</td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$(\sigma^w_0)^2$</td>
<td>1e-7</td>
<td>-</td>
</tr>
<tr>
<td>$(\sigma^m_0)^2$</td>
<td>2e-7</td>
<td>(rad/sec)$^2$</td>
</tr>
<tr>
<td>$(\sigma^y_0)^2$</td>
<td>5e-6</td>
<td>-</td>
</tr>
<tr>
<td>$(\sigma^d_0)^2$</td>
<td>5e-6</td>
<td>(rad/sec)$^2$</td>
</tr>
<tr>
<td>$(\sigma^g_{xx,0})^2$</td>
<td>1e-3</td>
<td>-</td>
</tr>
<tr>
<td>$(\sigma^q_{xx,0})^2$</td>
<td>5e-4</td>
<td>(rad/sec)$^2$</td>
</tr>
<tr>
<td>$(\sigma^q_{xx,0})^2$</td>
<td>50</td>
<td>(kgm)$^2$</td>
</tr>
<tr>
<td>$(\sigma^m_{xx,0})^2$</td>
<td>10</td>
<td>(kgm)$^2$</td>
</tr>
<tr>
<td>$(\sigma^f_{xx,0})^2$</td>
<td>0.02</td>
<td>(kgm)</td>
</tr>
<tr>
<td>$(\sigma^\omega_{xx,0})^2$</td>
<td>5e-4</td>
<td>(rad/sec)$^2$</td>
</tr>
</tbody>
</table>
Chapter 6

Discussion and Conclusions

6.1 Discussion and Conclusions

The work of this thesis can be divided into three major steps, research, implementation and testing. During the research several methodologies for system identification were considered. Two main methods, the EKF and the UKF, stood out as especially promising contenders for parameter estimation. Both of these filters have been utilized for spacecraft state estimation, but less so for parameter estimation.

The algorithm and general principles of the two types of nonlinear Kalman filters have been investigated and is presented in this report. Comparison between the two filters shows a clear superiority of the UKF as an estimation method for nonlinear systems, both in performance, flexibility, and ease of implementation. Surprisingly, this advantage comes at no extra computational burden as the computational complexity of the UKF is comparable to that of the EKF. Hence, the UKF was chosen as the base for the parameter estimation algorithm without further considering the EKF.

The modelling of spacecraft dynamics and kinematics was a fairly straightforward process, as many extensive references exists within the matter. However, only rigid spacecraft dynamics were considered and the modelling process could have been much more cumbersome with the inclusion of flexible dynamics or fuel sloshing. Nevertheless, as the estimation algorithm was developed with agile satellites in mind, and as a result of implementation with INTREPID, the dynamics of CMG actuators had to be researched and added as a separate part. The inclusion of the INTREPID mass-balancing system also added extra modelling requirement that needed to be taken into account.

Once the dynamics had been modelled, they had to be implemented in the chosen identification algorithm. Much of this implementation was based on previous work by Crassidis[3], where a similar method was used for estimation of spacecraft rate sensor bias. This implementation is, however, not entirely trivial as the constraints of the attitude quaternion can be difficult to work around.

The estimation method was nevertheless successfully developed, and after extensive testing and tuning in a simulation environment shown to give good estimation results. During the simulation, a lot of experience was gained in filter tuning. Some parameters showed to have a much greater impact than others. For starters, the initial state estimate, \( \hat{x}_0 \), showed to only impact filter convergence time, and as long as a "bad" initial guess was accompanied with a sufficiently high initial state covariance matrix, \( P_{xx,0} \), the effect on the filter proved negligible.

Surprisingly, the effect of the UKF scaling parameters, \( \alpha \) and \( \kappa \), also showed a limited effect on the filter performance. In contrast, the process- and measurement noise covariance matrices, \( P_w \) and \( P_v \), showed to have a great impact on the filter,
both in terms of convergence time and stability. In most cases, the covariance of the sensor noise is known in some respect and is non-too difficult to implement with the filter. The process noise on the other hand can be quite difficult to approximate and account for.

Once the estimator had been thoroughly tested and tuned in simulation environments, it was implemented and tested in practice on INTREPID satellite hardware demonstrator. It showed promising results almost immediately. The results presented in this paper are from the longest running experiment with internal power supply (practically no disturbances).

The results show convergence of the filter within a relatively short amount of time. The main inertia parameters are converging to their real values almost immediately with the first manoeuvre around their axis. The estimated CoG parameters of INTREPID seem to move around much more. However, this is almost exclusively a result of the mass-balancing system which moved quite a lot during the experiments.

Unfortunately, the exact values of INTREPID parameters are not exactly known as the platform has been modified and extended since the last measurements. Nevertheless, the values identified by the estimator are in the range of the last known values where $J_{xx} \approx J_{yy} \approx 20$, and $J_{zz} \approx 30$. Additionally, the estimator converged to essentially the same values during each experiment run, further pointing to credible results.

### 6.2 Summary and Future Work

The subject of the thesis was to research system identification, develop models of satellite attitude dynamics, and apply a method of parameter estimation using experimental data. All these goals have been met successfully, and are presented in this paper.

The result of the work is a proposed method for real-time parameter identification for space applications. The developed method has been experimentally verified with realistic dynamics using a satellite hardware demonstrator. The algorithm has been included in a user-friendly toolbox in Matlab/Simulink, complete with documentation, help-files and demo examples.

On the subject of future work, further research need to be done in the subject of optimal input design. Theoretically, a more exciting command profile should improve the convergence of the estimator, minimizing the time and power cost of performing an estimation manoeuvre in-orbit.

Additionally, the filter may be modified to include estimation of other parameters, e.g. sensor bias. It may even be possible to extend the filter to include flexible dynamics or fuel sloshing.
References


Chapter 7

Appendices
A UASPE algorithms

A.1 UASPE for Rigid Spacecraft

State-vector, $x$, input, $u$, and observation, $y$, are defined as

$$
\begin{align*}
x &= \begin{bmatrix} \delta \hat{g} \\ \omega \\ f \end{bmatrix}, \\
u &= \begin{bmatrix} h_{act} \\ T_{act} \end{bmatrix}, \\
y &= \begin{bmatrix} q \end{bmatrix}
\end{align*}
$$

(7.1)

The estimation algorithm is as follows.

Initialization

$$
\begin{align*}
\hat{x}_{k-1}^+ &= \hat{x}_0 = \mathbb{E}[x_0] = \begin{bmatrix} 0_{3 \times 1} \\ \hat{\omega}_0 \\ \hat{f}_0 \end{bmatrix} \\
P_{xx,k-1}^+ &= P_{xx,0} = \mathbb{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]
\end{align*}
$$

(7.2)

State reset

$$
\begin{align*}
\hat{x}_{k-1}^- &= \begin{bmatrix} 0_{3 \times 1} \\ \hat{\omega}_{k-1}^- \\ \hat{f}_{k-1}^- \end{bmatrix}
\end{align*}
$$

(7.3)

Calculate sigma points

$$
\begin{align*}
\chi_{0,k-1} &= \hat{x}_{k-1}^+ \\
\chi_{i,k-1} &= \hat{x}_{k-1}^+ + \left(\sqrt{(L + \lambda)P_{xx,k-1}^+}\right)_{i}, \quad i = 1, \ldots, L \\
\chi_{i,k-1} &= \hat{x}_{k-1}^+ - \left(\sqrt{(L + \lambda)P_{xx,k-1}^+}\right)_{i-L}, \quad i = L + 1, \ldots, 2L
\end{align*}
$$

(7.4)

Prediction

$$
\begin{align*}
\chi_{i,k-1}^{\delta q_0} &= \frac{2\sqrt{4 - \|\chi_{i,k-1}^{\delta g}\|^2}}{4 + \|\chi_{i,k-1}^{\delta g}\|^2} \\
\chi_{i,k-1}^{\delta q} &= \frac{\chi_{i,k-1}^{\delta q_0} - \chi_{i,k-1}^{\delta g}}{\chi_{i,k-1}^{\delta g}} \\
\chi_{i,k-1}^{q} &= \chi_{i,k-1}^{q} \odot \chi_{i,k-1}^{\delta q} \\
\chi_{i,k}^{q} &= \chi_{i,k}^{q} + \Delta t \frac{1}{2} \left[ \chi_{i,k-1}^{\delta q} \otimes \chi_{i,k}^{\delta g} \right] \\
\chi_{i,k}^{q} &= \chi_{i,k}^{q} \otimes \left(\chi_{0,k}^{q}\right)^{-1} \\
\chi_{i,k}^{\delta g} &= 2 \frac{\chi_{i,k}^{\delta q}}{\chi_{i,k}^{\delta q_0}} \\
\chi_{i,k}^{\delta q} &= \chi_{i,k}^{\delta q} + \Delta t \left(\chi_{i,k-1}^{\delta q}\right)^{-1} \left[ \tilde{T}_{act,k-1} + h_{act,k-1} \right] \\
\chi_{i,k}^{\delta q} &= \chi_{i,k}^{\delta q} \odot \left(\chi_{i,k-1}^{\delta q} \chi_{i,k}^{\delta g} + h_{act,k-1} \right) \\
\chi_{i,k}^{q} &= \chi_{i,k}^{q} \\
\chi_{i,k}^{q} &= \chi_{i,k}^{q} \\
\chi_{i,k}^{q} &= \chi_{i,k}^{q}
\end{align*}
$$

(7.5)
Weighted mean
\[
\hat{x}_{k}^{-} = \sum_{i=0}^{2L} W_{m}^{i} \chi_{i,k}
\]
\[
P_{xx,k}^{-} = \sum_{i=0}^{2L} W_{m}^{i} [\chi_{i,k} - \hat{x}_{k}^{-}] [\chi_{i,k} - \hat{x}_{k}^{-}]^{T} + P_{w}
\]
\[
\hat{y}_{k}^{-} = \sum_{i=0}^{2L} W_{m}^{i} Y_{i,k}
\]
\[
P_{yy,k}^{-} = \sum_{i=0}^{2L} W_{m}^{i} [Y_{i,k} - \hat{y}_{k}^{-}] [Y_{i,k} - \hat{y}_{k}^{-}]^{T} + P_{v}
\]
\[
P_{xy,k}^{-} = \sum_{i=0}^{2L} W_{m}^{i} [\chi_{i,k} - \hat{x}_{k}^{-}] [Y_{i,k} - \hat{y}_{k}^{-}]^{T}
\]

Correction
\[
K_{k} = P_{xy,k}^{-} (P_{yy,k}^{-})^{-1}
\]
\[
\hat{x}^{+}_{k} = \hat{x}_{k}^{-} + K_{k} (y_{k} - \hat{y}_{k}^{-})
\]
\[
P_{xx,k}^{+} = P_{xx,k}^{-} - K_{k} P_{yy,k}^{-} K_{k}^{T}
\]
\[
\delta \hat{q}_{0,k}^{+} = \frac{2}{4 + \| \delta \hat{g}_{k} ^{+} \| ^{2}}\| \delta \hat{g}_{k} ^{+} \| ^{2}
\]
\[
\delta \hat{q}_{k} ^{+} = \frac{\delta \hat{q}_{0,k}^{+}}{\delta \hat{g}_{k} ^{+}}
\]
\[
\hat{q}^{+}_{k} = \delta \hat{q}_{k} ^{+} \otimes Y_{0,k}^{q}
\]

Time step update
\[
k = k + 1
\]

Return to State reset

Definitions

\( k \)
- is the time step

\( x_{0} \)
- is the initial state estimate

\( \hat{x}_{0} \)
- is the initial state estimate

\( \hat{x}_{k}^{-} \)
- is the priori state estimate (weighted mean)

\( \hat{x}_{k}^{+} \)
- is the posteriori state estimate

\( \hat{y}_{k}^{-} \)
- is the priori observation estimate (weighted mean)

\( y_{k} \)
- is the current measurement

\( u_{k-1} \)
- is the input

\( P_{xx,0} \)
- is the initial state estimation error covariance

\( P_{xx,k}^{-} \)
- is the priori state estimation error covariance (weighted covariance)

\( P_{xx,k}^{+} \)
- is the posteriori state estimation error covariance

\( P_{yy,k}^{-} \)
- is the priori observation estimation error covariance (weighted covariance)

\( P_{xy,k}^{-} \)
- is the priori state-observation cross-covariance (weighted covariance)

\( P_{w} \)
- is process noise covariance matrix

\( P_{v} \)
- is measurement noise covariance matrix
$K_k$ is the Kalman gain

$\chi_{i,k}$ is the $i$th sigma point

$Y_{i,k}$ is the observation of $\chi_{i,k}$

$W^m_i$ and $W^c_i$ are the state and covariance weights

For additional definitions see [List of Symbols](#) and [Conventions](#).
A.2 UASPE for INTREPID

State-vector, $x$, input, $u$, and observation, $y$, are defined as

$$x = \begin{bmatrix} \delta \tilde{g} \\ \omega \\ f \\ \tilde{M} \end{bmatrix}, \quad u = \begin{bmatrix} h_{\text{CMG}} \\ \tilde{T}_{\text{CMG}} \\ \tilde{r}_{\text{MB}} \end{bmatrix}, \quad y = \begin{bmatrix} q \\ \omega \end{bmatrix}$$

(7.9)

The estimation algorithm is as follows.

Initialization

$$\hat{x}_{k-1}^+ = \hat{x}_0 = E[x_0] = \begin{bmatrix} 0_{3 \times 1} \\ \omega_0 \\ f_0 \\ \tilde{M} r_0 \end{bmatrix}$$

(7.10)

$$P_{xx,k-1}^+ = P_{xx,0} = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]$$

State reset

$$\hat{x}_{k-1}^+ = \begin{bmatrix} 0_{3 \times 1} \\ \omega_{k-1}^+ \\ f_{k-1}^+ \\ \tilde{M} r_{k-1} \end{bmatrix}$$

(7.11)

Calculate sigma points

$$\chi_{0,k-1}^+ = \hat{x}_{k-1}^+$$

$$\chi_{i,k-1}^+ = \hat{x}_{k-1}^+ + \left( \sqrt{(L + \lambda) P_{xx,k-1}^+} \right)_i, \quad i = 1, \ldots, L$$

$$\chi_{i,k-1}^+ = \hat{x}_{k-1}^+ - \left( \sqrt{(L + \lambda) P_{xx,k-1}^+} \right)_{i-L}, \quad i = L + 1, \ldots, 2L$$

(7.12)

Prediction

$$\chi_{i,k-1}^{\delta q} = \frac{2}{\sqrt{4 - \|\chi_{i,k-1}^{\delta g}\|^2}} \chi_{i,k-1}^{\delta g}$$

$$\chi_{i,k-1}^{\delta q} = \frac{\chi_{i,k-1}^{\delta q}}{\chi_{i,k-1}^{\delta q}}$$

$$\chi_{i,k-1}^{q} = \chi_{i,k-1}^{q} \otimes \hat{q}_{k-1}^+$$

$$\chi_{i,k}^{q} = \chi_{i,k-1}^{q} + \Delta t \frac{1}{2} [\chi_{i,k-1}^{\omega} \otimes \chi_{i,k-1}] \chi_{i,k-1}^{q}$$

$$\chi_{i,k}^{q} = \chi_{i,k}^{q} \otimes (\chi_{0,k}^{q})^{-1}$$

$$\chi_{i,k}^{\delta q} = 2 \chi_{i,k}^{\delta q}$$

$$\chi_{i,k}^{\delta q} = 2 \chi_{i,k}^{\delta q}$$

(7.13)
\[ \chi_{i,k-1}^{M_{\text{tot}}} = \chi_{i,k-1}^{M} + 4m\vec{r}_{MB,k-1} \]
\[ \vec{T}_{g,k-1} = \chi_{i,k-1}^{M_{\text{tot}}} \times \left( A(\chi_{i,k-1}^{q}) \vec{g} \right) \]
\[ \chi_{i,k}^{q} = \chi_{i,k-1}^{q} + \Delta t(\chi_{i,k-1}^{I})^{-1} \left[ \vec{T}_{g,k-1}(\chi_{i,k-1}^{q}, \chi_{i,k-1}^{M_{\text{tot}}}) \right. \]
\[ \left. + \vec{T}_{CMG,k-1} - \chi_{i,k-1}^{q} \times (\chi_{i,k-1}^{I} \chi_{i,k-1}^{q} + h_{CMG,k-1}) \right] \] (7.14)

\[ \chi_{i,k}^{I} = \chi_{i,k-1}^{I} \]
\[ \chi_{i,k}^{M} = \chi_{i,k-1}^{M} \]
\[ Y_{i,k}^{q} = X_{i,k}^{q} \]
\[ Y_{i,k}^{o} = X_{i,k}^{o} \]

Weighted mean
\[ \hat{x}_{k} = \sum_{i=0}^{2L} W_{i}^{m} x_{i,k} \]
\[ P_{xx,k}^{-} = \sum_{i=0}^{2L} W_{i}^{c} [x_{i,k} - \hat{x}_{k}^{-}] [x_{i,k} - \hat{x}_{k}^{-}]^{T} + P_{w} \]
\[ \hat{y}_{k} = \sum_{i=0}^{2L} W_{i}^{m} y_{i,k} \]
\[ P_{yy,k}^{-} = \sum_{i=0}^{2L} W_{i}^{c} [y_{i,k} - \hat{y}_{k}^{-}] [y_{i,k} - \hat{y}_{k}^{-}]^{T} + P_{v} \]
\[ P_{xy,k}^{-} = \sum_{i=0}^{2L} W_{i}^{c} [x_{i,k} - \hat{x}_{k}^{-}] [y_{i,k} - \hat{y}_{k}^{-}]^{T} \]

Correction
\[ K_{k} = P_{xy,k}^{-}(P_{yy,k}^{-})^{-1} \]
\[ \hat{x}_{k}^{+} = \hat{x}_{k}^{-} + K_{k}(y_{k} - \hat{y}_{k}^{-}) \] (7.16)
\[ P_{xx,k}^{+} = P_{xx,k}^{-} - K_{k}P_{yy,k}^{-}K_{k}^{T} \]

\[ \delta q_{0,k}^{+} = \frac{2\sqrt{4 - ||\delta \hat{\vec{q}}_{k}^{+}||^2}}{4 + ||\delta \hat{\vec{q}}_{k}^{+}||^2} \]
\[ \delta \hat{\vec{q}}_{k}^{+} = \frac{\delta q_{0,k}^{+}}{\delta \hat{\vec{q}}_{k}^{+}} \] (7.17)

Time step update
\[ k = k + 1 \]

Return to State reset

Definitions

- \( k \) is the time step
- \( x_{0} \) is the initial state estimate
- \( \hat{x}_{0} \) is the initial state estimate
\( \hat{x}_k^- \) is the priori state estimate (weighted mean)
\( \hat{x}_k^+ \) is the posteriori state estimate
\( \hat{y}_k^- \) is the priori observation estimate (weighted mean)
\( y_k \) is the current measurement
\( u_{k-1} \) is the input
\( P_{xx,0} \) is the initial state estimation error covariance
\( P_{xx,k}^- \) is the priori state estimation error covariance (weighted covariance)
\( P_{xx,k}^+ \) is the posteriori state estimation error covariance
\( P_{yy,k}^- \) is the priori observation estimation error covariance (weighted covariance)
\( P_{xy,k}^- \) is the priori state-observation cross-covariance (weighted covariance)
\( P_w \) is process noise covariance matrix
\( P_v \) is measurement noise covariance matrix
\( K_k \) is the Kalman gain
\( \chi_{i,k} \) is the \( i \)th sigma point
\( Y_{i,k} \) is the observation of \( \chi_{i,k} \)
\( W_{i}^w \) and \( W_{i}^c \) are the state and covariance weights

For additional definitions see \textbf{List of Symbols} and \textbf{Conventions}.
B  UASPE Simulink Model

Figure 7.1: UASPE for INTREPID, as it is realized in Simulink.
LISTING 7.1: getSigmaPoints

```matlab
%% getSigmaPoints
% Computes sigma points, chi
%
% Parameters:
% alpha    UASPE tuning parameter alpha
% kappa    UASPE tuning parameter kappa
% numStates size of state vector
% numParams size of parameter vector
% covPxxEst0 initial state covariance matrix, Pxx0
%
% Inputs:
% stateEst previous state estimation vector
% covPxxEst previous estimated state covariance matrix, Pxx
%
% Outputs:
% chi    UASPE spread sigma points
%
% Syntax:
% function chi = getSigmaPoints(stateEst, covPxxEst, alpha, kappa,
%     numStates, numParams, covPxxEst0)
```

LISTING 7.2: propSigmaPoints

```matlab
%% propSigmaPoints
% Propagates sigma points using rigid body dynamics, and gibbs–
% parameterization of estimation error quaternion.
%
% Parameters:
% numStates size of state vector
% numMeas size of measurement vector
% numParams size of parameter vector
% massBalMass mass of single mass balancing block [kg]
% dtUaspe sample time of UASPE filter [sec]
%
% Inputs:
% chi    UASPE spread sigma points
% cmgAngMomTot_B total angular momentum from all CMG's in body
% frame [Nms]
% cmgTrqTot_B total torque acting on the s/c from all CMG's
% in body frame [Nm]
% massBalPos position of each mass balancing block in body
% frame [m]
% quatEst_BI current estimate of attitude quaternion
%
% Outputs:
% chiProp propagated sigma points, f(Xi)
% chiMeas propagated measurement points, h(Xi)
%
% Syntax:
```
% function [chiProp, chiMeas] = propSigmaPoints(chi, cmgAngMomTot_B, cmgTrqTot_B, massBalPos, quatEst_BI, numStates, numMeas, numParams, massBalMass, dtUaspe)

LISTING 7.3: kalmanUpdate

%% kalmanUpdate
% Performs Kalman Update
% Parameters:
% numStates size of state vector
% numMeas size of measurement vector
% numParams size of parameter vector
% alpha UASPE tuning parameter alpha
% beta UASPE tuning parameter beta
% kappa UASPE tuning parameter kappa
% covPw process noise covariance matrix, Pw
% covPv measurement noise covariance matrix, Pv
% Inputs:
% yMeas measurement vector
% chiProp propagated sigma points, f(Xi)
% chiMeas propagated measurement points, h(Xi)
% Outputs:
% stateParamEst estimated states and parameters
% covPxxEst estimated state covariance matrix, Pxx
% covPxxDiag square-root of diagonal Pxx elements (for plotting 3-sigma bounds)
% quatEst_BI estimated of attitude quaternion
% Syntax:
% function [stateParamEst, covPxxEst, covPxxDiag, quatEst_BI] = kalmanUpdate(yMeas, chiProp, chiMeas, numStates, numMeas, numParams, alpha, beta, kappa, covPw, covPv)
C  UASPE Additional Simulink Blocks

![Simulink Diagram](image)

**Figure 7.2:** Simulink blocks for calculating CMG torques and angular momentum (left), and combined CoG of mass-balancing blocks (right).

**Listing 7.4: cmgDynamics**

```matlab
%% cmgDynamics
% Computes total CMG angular momentum and torque in body frame

% Parameters:
% cmgMagAngMom absolute magnitude of angular momentum
% for each CMG [Nms]
% cmg1UnitAtt0_Dcm_BU DCM from unit to body frame for CMG #1
% cmg2UnitAtt0_Dcm_BU DCM from unit to body frame for CMG #2
% cmg3UnitAtt0_Dcm_BU DCM from unit to body frame for CMG #3
% cmg4UnitAtt0_Dcm_BU DCM from unit to body frame for CMG #4

% Inputs:
% gimbalAng_1 gimbal angle of CMG #1 [rad]
% gimbalRate_1 gimbal angular rate of CMG #1 [rad/sec]
% gimbalAng_2 gimbal angle of CMG #2 [rad]
% gimbalRate_2 gimbal angular rate of CMG #2 [rad/sec]
% gimbalAng_3 gimbal angle of CMG #3 [rad]
% gimbalRate_3 gimbal angular rate of CMG #3 [rad/sec]
% gimbalAng_4 gimbal angle of CMG #4 [rad]
% gimbalRate_4 gimbal angular rate of CMG #4 [rad/sec]

% Outputs:
```
Chapter 7. Appendices

Listing 7.5: getMassBalPos

% getMassBalPos
% Computation of mass balancing blocks' position in body-frame
%
% Inputs:
% massBalDist_1 distance block #1 has moved along its track [mm]
% massBalDist_2 distance block #2 has moved along its track [mm]
% massBalDist_3 distance block #3 has moved along its track [mm]
% massBalDist_4 distance block #4 has moved along its track [mm]
% massBalPos0 mass balancing blocks' initial position in body frame (3x4) [m]
% massBalAng tilt angle of mass balancing tracks [rad]
%
% Outputs:
% massBalPos mass balancing blocks' position in body frame (3x4) [m]
%
% Syntax:
% function massBalPos = getMassBalPos(massBalDist_1, massBalDist_2, massBalDist_3, massBalDist_4, massBalPos0, massBalAng)