POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

States and parameters estimation in the post-capture scenario of a spacecraft with flexible manipulator

Tesi di Laurea Magistrale in
Ingegneria Spaziale

Author: **Davide Degavi**

Student ID: 969230
Advisor: Prof. Mauro Massari
Co-advisors: Prof. Pierluigi Di Lizia, Prof. Riccardo Vescovini
Academic Year: 2021-2022

Abstract

The ever-increasing number of space debris may put at risk the accessibility of space in the future. The recent development of large constellations means that, even with the proper mitigation strategies, the number of unwanted objects orbiting Earth will rapidly increase. To invert the growth in fragments number, missions such as Active Debris Removal (ADR) and On-Orbit Servicing (OOS) are required. They strive to reduce the debris in space, and this is achieved in different ways; ADR plans to move large, mostly intact satellites to orbits with a fast natural decay, while OOS aims to increase the operational life of existing satellites. ESA's e.Deorbit mission has the aim of developing a multi-purpose vehicle capable of both ADR and OOS missions.

The post-capture phase for ADR and OSS has seen fewer developments than the pre-capture one, especially to what flexibility is concerned. Large solar panels and lightweight manipulators introduce flexibilities that should be considered in the attitude determination process.

This thesis investigates the effects of flexibility in the attitude and inertial parameter estimation performance in the post-capture scenario of ESA's e.Deorbit mission to the Envisat satellite. In this work, the flexible system was modelled with the use of acausal programming language, and the attitude estimation filters were implemented within the functional mock-up interface (FMI) standard. Two Kalman filters are implemented, the Extended Kalman Filter (EKF) and the Multiplicative Extended Kalman Filter (MEKF). At last, the performance of the filters is then evaluated and compared. Simulations show that the EKF is much more capable than the MEKF in the estimation process, mainly due to the EKF's use of the Functional Mock-up Unit (FMU) specific tools.

Keywords: Attitude estimation, EKF, MEKF, Parameter estimation, FMI

Abstract in lingua italiana

Il continuo aumento del numero di detriti spaziali potrebbe limitare l'accesso allo spazio nel futuro prossimo. Il recente sviluppo di grandi costellazioni comporta che, nonostante la corretta applicazione di strategie di mitigazione, il numero di frammenti in orbita aumenterà rapidamente. Affinchè ci sia una inversione di questo trend è necessario l'utilizzo di strategie di Active Debris Removal (ADR) e di On-Orbit Servicng (OOS). Il loro obbiettivo è quello di ridurre il numero di detriti nello spazio. Una missione ADR vuole spostare un satellite defunto in un'orbita che abbia un tempo di decadimento relativamente rapido, mentre una missione OOS vuole aumentare la vita operativa di un satellite già in orbita. L'Agenzial Spaziale Europea (ESA) ha all'attivo la missione e.Deorbit con l'obbiettivo di sviluppare un veicolo multi funzione che sia in grado di portare a compimento entrambe le missioni di ADR e di OOS.

Nelle missioni di ADR e OOS, la fase sucessiva alla cattura del satellite bersaglio è stata indagata con un minore dettaglio in particolare per quanto riguarda gli effetti della flessibilità. L'utilizzo di grandi strutture leggere, come pannelli solari e bracci robotici, introduce inevitabilmente flessibilità nel sistema. Tale effetto dovrebbe essere considerato nel processo di determinazione dell'attitude.

Questa tesi indaga gli effetti della flessibilità nella stima dell'attitude e nella stima dei parametri inerziali, in particolar modo della massa. La stima è effettuata nel contesto proposto dalla missione e.Deorbit dell'ESA, il cui obbiettivo è quello di rimuovere il satellite Envisat. In questo lavoro di tesi, il sistema è modellato attraveso l'utilizzo di un linguaggio di programmazione acausale e l'implementazione dei filtri è stata eseguita con lo standard functional mock-up interface (FMI). Sono state implementate due tipologie di filtri di Kalman, un Extended Kalman Filter (EKF) e un Multiplicative Extended Kalman Filter (MEKF). Infine, sono state valutate e confrontate le performance dei due filtri. Le simulazioni effettuate mostrano come il filtro EKF sia più abile rispetto al MEKF nel processo di stima; questo è dovuto principalmente all'utilizzo estensivo nella implementazione dell'EKF degli strumenti forniti dalla functional mock-up unit (FMU).

Parole chiave: Stima dell'attitude, EKF, MEKF, Stima dei Parametri, FMI

Contents

1 | Introduction

This initial chapter seeks to provide a broad overview of the active debris removal (ADR) and on-orbit servicing (OOS) missions that form the foundation of this thesis study. The ESA e.Deorbit mission is presented within the paradigm introduced. A discussion of the state of the art in the pre and post-capture of uncooperative spacecraft will precede it. The dissertation regarding the thesis work is provided in the chapter's final section.

1.1. Active debris removal

The term "space debris" refers to any human-made detritus that orbits Earth. Such elements can range in size from a few centimeters across to fully intact satellites or even entire rocket stages. In 2022, ESA published its yearly technical report on the state of the space environment [8]. ESA and other agencies have recorded and regularly tracked space debris pieces. Today, more than 30 000 objects larger than 10 cm across are being monitored. Figure 1.1 shows the evolution with time of the tracked elements. If an object's origin can be traced to a payload, it is labeled P. If it can be traced to a rocked body, it is given the name R. If it is unclassified, it is given the label UI.

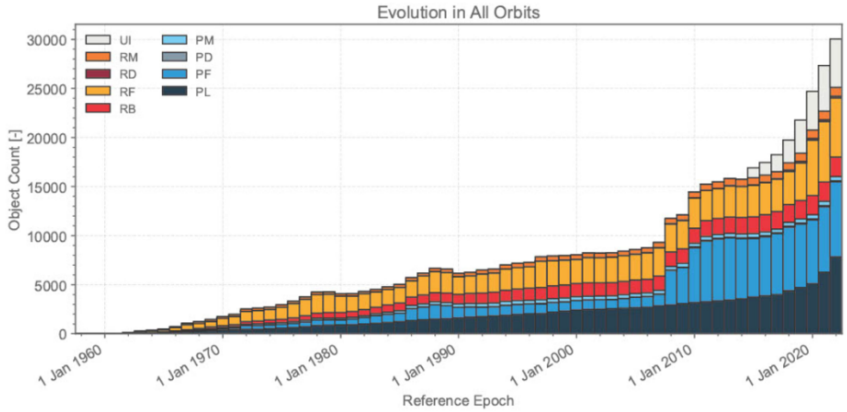


Figure 1.1: Evolution of number of objects in geocentric orbit by object class, source [8].

ESA estimates that the number of space debris pieces orbiting Earth larger than 1 cm is

in the order of millions.

In recent years, there has been a sharp increase in satellite launches. New commercial operators are primarily responsible for this trend as they work to build expansive satellite constellations. The number of satellites launched into orbit each year is depicted in Figure 1.2.

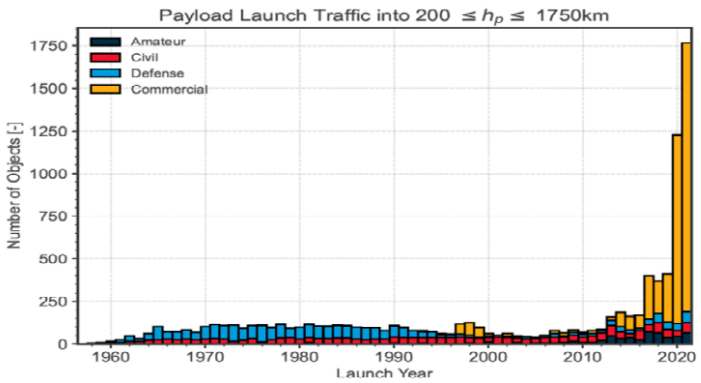


Figure 1.2: Evolution of the launch traffic near LEO per mass category, source [8].

Although the space industry has become more responsible than ever regarding how and what is being launched, the ESA report [8] emphasizes that the amount of space debris is destined to increase enormously in the future unless additional measures are taken. This implies that strict adherence to the IADC's recommendations for space debris mitigation is required to ensure the long-term viability of space activities. The following goals can be connected to the suggested mitigation measures:

- Limitation of space debris released during normal operation;
- Avoid collision and anti-satellite rocket tests;
- Post mission disposal.

The third goal includes actively removing debris (ADR). Studies reveal that reducing orbital collision events is necessary, along with new launches following mitigation procedures, to reverse the trend depicted in figure 1.1. ESA estimates that around 2500 decommissioned satellites in Low Earth Orbit (LEO) are mostly intact [7]. These relatively large objects could represent a risk for future space accessibility. Thus, it becomes necessary to implement ADR as a remediation program to reduce the number of large and mostly intact satellites orbiting Earth. Such action will be highly efficient in reducing space junk since it eliminates objects that could generate an enormous debris cloud.

1.2. On-orbit servicing

The OOS missions are an alternative to the ADR for reducing space debris. OOS seeks to prolong the operational life of the target satellites [17], as opposed to ADR, which strives to remove large spacecraft from dense orbits. The inability to execute servicing, repairs, or upgrades after deployments indirectly reduces the time a space system may maintain its capabilities. Recent studies have found that such untimely equipment losses result in billions of dollars in damages [6]. Due to a decrease in launch-related mishaps, in-orbit issues are progressively taking the lead in space system failures. On-orbit servicing is the general phrase for a spacecraft's post-launch modification. Prior missions have demonstrated the efficiency and capability of OSS, such as the repair of the Hubble Space Telescope or the upkeep of the International Space Station (ISS). The goal of the space industry is to completely switch to robotic servicing missions to cut costs and enable repairs even outside of low earth orbits.
OSS missions will require the development of the following capabilities:

- Robotic manipulation
- Rendezvous and proximity operations
- Refueling and repairs

A robotic manipulator provides the ability to grapple and manipulate payloads or other spacecraft. This capability is one of the most developed with respect to others. The Space Station Remote Manipulator System (SSRMS) and the Shuttle Remote Manipulator System (SRMS), both of which have supported human space exploration since 1981, are notable examples. New developments are being worked on within the robotic manipulation capability area. Planned manipulators designed for repair and assembly work in open space with lightweight robotic elements are well underway: ROKVISS [2] can be seen as an European space-tested example of recent innovations. The first fully unmanned mating of two spacecraft was performed in 1998; nonetheless, there was no demonstration of autonomous capture or mating with an uncooperative satellite. Since it will be necessary to carry out any refueling or repairs in orbit, this is one of the critical capabilities that must be developed to improve OOS feasibility. The most recent advancement in the OSS was accomplished in 2021 by Northrop Grumman's MEV-2, which performed the first space demonstration of docking to an operational geostationary satellite. Regarding the refueling capability, the Russian Progress vehicle represents the most mature fluid transfer system.

It is clear that OOS requires more developments to reach a sufficiently high technology

readiness level before an actual mission can occur.

1.3. ESA e.Deorbit mission

In the context of ADR and OOS, ESA worked on the e.Deorbit mission. The mission was developed by ESA's Clean Space initiative in 2013, with the deceptively simple aim of capturing and safely deorbiting a derelict ESA-owned spacecraft in highly frequented low-Earth orbit. The Envisat Earth-observing satellite, which had been operating for ten years, was the target. It abruptly failed in 2012. During the 1980s, Space Shuttle astronauts captured rogue satellites, but there was no precedent for the autonomous robotic acquisition of such a recalcitrant target.

With a total of ten instruments and a weight of eight tonnes, Envisat is one of the biggest Earth observation satellites ever constructed. Studying atmospheric chemistry and enabling more efficient resource monitoring and management were two of Envisat's goals. Similarly to other Earth observation satellites, it was placed in a Sun-synchronous orbit with an altitude of $782km$. On the 8th of April 2012, the communication link with the satellite was lost for still unknown reasons.

ESA is strongly committed to the reduction of space debris. With this objective in mind, six tasks were identified that would need to be fulfilled by the e.Deorbit mission [3]:

- Launch into space
- Perform commissioning
- Transfer and phase to target orbit
- Rendezvous with target
- Capture target
- De-orbit target

Phase A of the e.Deorbit program began in 2014. To carry out the earliest basic research, a contract was given to three companies: Airbus, OHB and Thales Alenia. The firms put out similar proposals for phase B1, which calls for approaching the target satellite from the launch adapter side and grabbing the adapter flange using a manipulator with seven degrees of freedom (DOFs). Figure 1.3 provides a visual representation of this process.

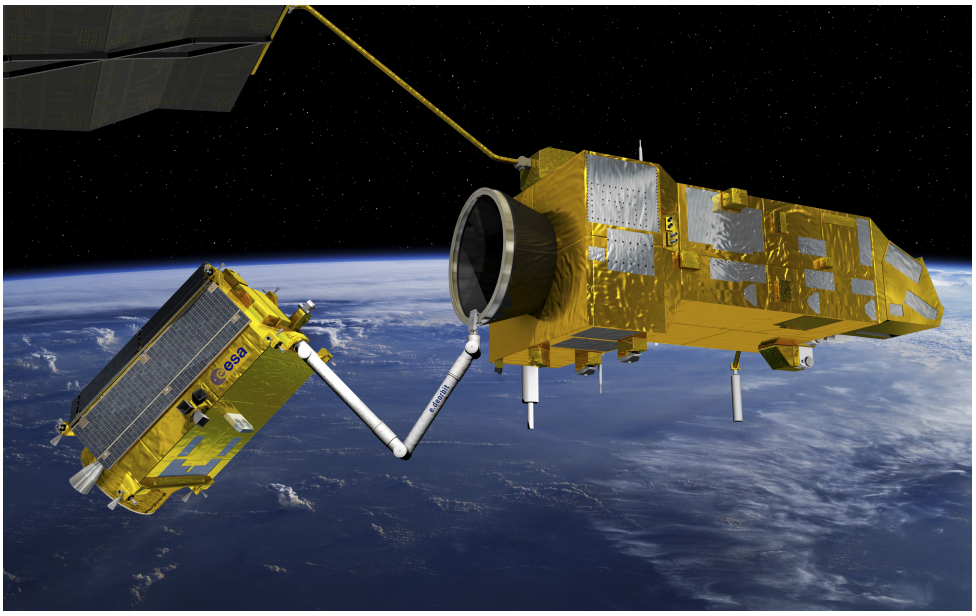


Figure 1.3: e.Deorbit mission visualization by ESA–David Ducros, 2016.

A comprehensive review of the e.Deorbit mission was carried out in 2018, following several delays. Since most of the capabilities needed to complete an ADR mission are best suited for an OSS vehicle, ESA increased the scope of the initial mission. The e.Deorbit program now seeks to create a more multifunctional satellite with the potential to undertake both spacecraft removal and servicing while it is in orbit.

1.4. State of the art

The surge in literature studies and the space industry's interest in the ADR and OOS missions are strongly related. Capturing the target satellite is one of the most important stages that both types of missions share. This task divides the proximity operation mission phase into three distinct phases: the pre-capture scenario, the grasping action, and the post-capture operations. The design of a manipulator is different for space applications compared to terrestrial ones. Space manipulator systems (SMS) are made to be lightweight and long-reaching due to the lack of gravity loading, which introduces link flexibilities. Additionally, deployable truss antennas, lightweight, flexible constructions, and solar arrays are used [5]. Often, their joints are driven by harmonic drives with large gear ratio and compact design, introducing joint flexibility [22]. During on-orbit service, these kinds of flexibilities could generate vibrations in the manipulator and in the spacecraft, especially when physical contact is involved [20]. Flexibilities introduce unique effects in each of the three phases of which the capture process is made of.

The biggest issue is with manipulator control when it comes to flexibility effects during the pre-capture of the target satellites. In reality, if such effects are ignored, poor performance and even control instabilities may happen. Meng et al. [16] derived the kinematics equation governing a free-floating satellite with a flexible manipulator and appendices. A free-floating model describes the system without any external action; thus, the center of mass does not move, and the attitude of the spacecraft is influenced only by the movement of the manipulator. In the same work, the coupling effect between the rigid movement of the end-effector and the solar array has been modelled. The coupled dynamics equation is required to conduct the pre-grasping motion planning.

While capturing high-speed orbiting objects, robotic arms undergo impact and require appropriate modeling of the system. The contact phase represents another challenging task in which the flexibility effects must be considered to determine exactly the forces acting on the manipulator. Raina et al. [19] proposed a unified framework for the modeling of impact dynamics. The model for the robotic arm is obtained using the Decoupled Natural Orthogonal Complements (DeNOC) with closed-loop constraints equations.

The post-capture scenario for an ADR or OOS mission has seen fewer analyses than the contact phase or the pre-capture one. It would be very helpful for the subsequent controller design if the target mass properties could be identified. Accordingly, particular control and parameter identification techniques must be developed to cope with the challenging issues. Wang et al. [23] developed a detumbling strategy and an impedance control scheme for the post-capture scenario. The model comprises two satellites linked with one another through a manipulator modelled with joint flexibility. In the same paper, an approach is proposed to estimate the target satellite's inertia properties. It is possible to identify the true values of the mass properties from the observations of the torque-free tumbling motion of a target satellite without excitation [1]. The inertial parameters are derived using a laser camera system (LCS). An algorithm is then used to match the points from the CAD model and the scanned data to estimate the object pose. The derived pose is then used with a Kalman filter to estimate the inertial parameters of the target object. Visual-based techniques to determine the inertia values of the target satellite have been thoroughly researched. Although such solutions are viable, they may result in poor performance when the target satellite has a high damage level that could impede the correct identification of the satellite features. If large flexible appendices or solar panels are present, it is possible that, with their motion, they can interfere with the pose determination algorithm.

Incorporating flexible modelling in the post-capture phase of an ADR or OOS mission is an important step to increase the knowledge of such a mission and to see in advance

any difficulties in the mission scenario. The research on this topic is less diverse than the one performed in other phases. Most researchers, like Raina et al. [19], preferred to start the modelling of the system with the manipulator as the only flexible element. Meng et al. [15] derived a dynamical model and developed a closed-loop simulation system for a post-capture phase of a large flexible spacecraft in order to supply a way to verify key path planning and control algorithms. This work represents one of the most complete analyses of the post-capture scenario with flexibility effects. It is important to underline that the majority of such works do not provide any verification for the attitude determination algorithm with large flexible satellites. An analysis of the performance of the attitude estimation process with flexibility effects is presented by Ghani et al. [10]. In addition to that, the results of the estimation of the flexible states of the satellite are provided. In their work, Ghani et al. showed that it is possible to estimate the deformation of the solar panel through the use of a Kalman filter.

1.5. Thesis dissertation

Research on flexibility effects in the aftermath of a target satellite's capture has been found to be generally lacking, especially when paired with the study of attitude and parameter estimation. This thesis aims to provide more information on the subject, improve understanding of attitude estimation algorithm performance, and increase the accuracy of inertial parameter identification. In particular, the modelled system will be mostly based on the research done in the phase B1 of the e.Deorbit mission [3]. The Envisat is the target satellite, and the chaser satellite is derived from the preliminary design work performed by OHB [18]. The overall system comprises the chaser satellite and Envisat with one flexible solar panel each and a flexible 7 DOFs manipulator that links the two spacecrafts together.

The modelling of the system is performed with the Modelica language, which allows for the use of acausal modelling techniques. Quaternion elements have been chosen as the attitude parametrization with which the chaser satellite is identified. The Extended Kalman Filter (EKF) and the Multiplicative Extended Kalman Filter (MEKF) are the estimation algorithms for the attitude quaternion. The EKF is also used for estimating the inertia properties. In particular, in this work, the mass of the Envisat is estimated.

The usage of the Functional Mock-up Interface (FMI) [9] for the attitude estimation procedure is another contribution of this study. The use of FMI represents a novel approach for the non-linear filter implementation in the attitude determination of a satellite.

The thesis is divided into four main chapters. First, the fundamental mathematical and

informatics instruments required to comprehend the work are explained. In particular, the Modelica language and the filter derivation are described. Then, a detailed explanation of the model derivation is given with a focus on the validation of the lumped parameter model with which the flexible solar panels are implemented. A brief summary of how the Kalman filters needed have been implemented within the Modelica association FMI [9] is given.

Chapter four presents the results obtained from the attitude estimation process. In particular, four different analyses are carried out. The attitude estimation process is performed with the EKF and MEKF filter, and it is done using a model with flexibility effects and a model with rigid bodies assumption. The performance of the filters in both scenarios is analyzed and compared. The final part of this thesis work summarizes the achieved results and provides suggestions for future developments.

2 | Fundamentals

The mathematical background and informatics instruments, needed to understand the work presented, are provided in this chapter. With the parts below, the contents are succinctly listed.

2.1. Euler Lagrange formalism

Hamilton's principle, which defines the motion of a mechanical system when all the forces can be determined from scalar potential, is used to construct the equations of motion in Lagrangian mechanics. Hamilton's principle will be followed as the system evolves.

Theorem 2.1. *(Hamilton's Principle). The motion of a dynamical system in a given time interval is such as to maximize or minimize the action integral defined by:*

$$J = \int_{t_0}^{T} \mathcal{L}(t, q, \dot{q}) dt \tag{2.1}$$

The Lagrangian is defined as $\mathcal{L} = K - U$, where K represents the kinetic energy and U represents the potential energy.

From Hamilton's principle, it is possible to derive the Lagrange equations of motion using the calculus of variations.

The variation of the integral J is:

$$\delta J = \delta \int_{t_0}^{T} \mathcal{L}(t, q, \dot{q}) dt \tag{2.2}$$

J is now considered as a function of parameter α that labels a possible set of curves $q_1(t, \alpha)$.

$$\begin{aligned} q_1(t, \alpha) &= q_1(t, 0) + \alpha \eta_1(t) \\ q_2(t, \alpha) &= q_2(t, 0) + \alpha \eta_2(t) \\ &\dots \end{aligned} \tag{2.3}$$

where $q_1(t,0)$, $q_2(t,0)$, etc. are the solutions of the extremum problem and η_1, η_2, etc. are independent functions of t that vanish at the endpoints and that are $\mathcal{C}^2$. The calculation proceeds by applying techniques of the calculus of variations:

$$\frac{\partial J}{\partial \alpha} d\alpha = \int_{t_0}^{T} \sum_i \left(\frac{\partial \mathcal{L}}{\partial q_i} \frac{\partial q_i}{\partial \alpha} d\alpha + \frac{\partial \mathcal{L}}{\partial \dot{q}_i} \frac{\partial \dot{q}_i}{\partial \alpha} d\alpha \right) dt \tag{2.4}$$

Then, by applying the integration by parts rule, it is possible to write that:

$$\int_{t_0}^{T} \frac{\partial \mathcal{L}}{\partial \dot{q}_i} \frac{\partial^2 q_i}{\partial \alpha \partial t} d\alpha = \frac{\partial \mathcal{L}}{\partial \dot{q}_i} \frac{\partial q_i}{\partial \alpha} \bigg|_{t_0}^{T} - \int_{t_0}^{T} \frac{\partial q_i}{\partial \alpha} \frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i} d\alpha \right) \tag{2.5}$$

where the first term vanishes because all curves pass through the fixed endpoints. Substituting in Eq (2.4), δJ becomes

$$\delta J = \int_{t_0}^{T} \sum_t \left(\frac{\partial \mathcal{L}}{\partial q_i} - \frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_i} \right) \delta q_i dt \tag{2.6}$$

Since the variables q are independent, the various δq_i are independent, hence the condition that δJ is zero requires that the coefficients of the δq_i separately vanish:

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = 0 \qquad i = 1, 2, \ldots, n \tag{2.7}$$

The Lagrange equations are obtained from Hamilton's principle for monogenic systems with holonomic constraints.

The generalized coordinates of non-holonomic systems are not independent of each other, and thus they have the following constraints equation:

$$\mathcal{L}(q_1, q_2, \ldots, q_n, t) = 0 \tag{2.8}$$

It is possible to extend Hamilton's principle to cover semi-holonomic systems. In a semi-holonomic system, the equation of the constraints may be expressed in the form:

$$f_\alpha(q_1, \ldots, q_n; \dot{q}_1, \ldots, \dot{q}_n) = 0 \qquad \alpha = 1, 2, \ldots, m \tag{2.9}$$

The *Lagrange multipliers* are used in the procedure to obtain Lagrange's equations for semi-holonomic systems.

$$\sum_{\alpha=1}^{m} \lambda_\alpha f_\alpha = 0 \tag{2.10}$$

The equation (2.2) becomes:

$$\delta \int_{t_0}^{T} \left(\mathcal{L} + \sum_{\alpha=1}^{m} \lambda_\alpha f_\alpha \right) dt = 0 \tag{2.11}$$

The variation can now be performed with the δq_i and λ_α for the $m + n$ independent variables. Assuming that $\lambda_\alpha = \lambda_\alpha(t)$, the resulting equation becomes:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = Q_k \quad \text{where} \quad Q_k = \sum_{\alpha=1}^{m} \left\{ \lambda_\alpha \left[\frac{\partial f_\alpha}{\partial q_i} - \frac{d}{dt}\left(\frac{\partial f_\alpha}{\partial \dot{q}_i}\right)\right] - \frac{d\lambda_\alpha}{dt}\frac{\partial f_\alpha}{\partial \dot{q}_i} \right\} \tag{2.12}$$

where the term Q_k represents the generalized force applied to the system.

2.2. Modelica Language

The Modelica Language and the Functional Mock-up Interface (FMI) are open-source standards developed and supported by the Modelica Association, a non-profit organization.

The Modelica Language is a programming language for modelling cyber-physical systems, supporting acausal connection of components governed by mathematical equations to facilitate modelling from first principles. Modelica language defines an object-oriented equation-based programming language.

2.2.1. Co-Simulation

Co-simulation is a technique that extends the classical notion of simulation, splitting a complex system into many smaller heterogeneous domain-specific systems (System-of-Systems) which interact among themselves with an input/output causal communication scheme. One of the most significant benefits of using co-simulation methodologies is the ability to deploy each subsystem into separate solvers, which are coordinated by the platform that maintains time synchronisation and data exchanges. The system may be broken down into separate subsystems thanks to the decentralized design, enabling plug-and-play operation and the progressive replacement of simulators and models with actual physical hardware. For instance, adding a new element or replacing an existing one may be implemented into the model with little effort since it is only necessary to edit a single component rather than the whole model.

A standard that enables the communication between the model's hardware and software components and a straightforward and standardized process for swapping out individual

components is required to create a co-simulation framework. The FMI (Functional Mock-up Interface) standard will be applied to this specific objective.

2.2.2. Acausal modelling

The modelling of a real system is performed with equations, as done in the Euler-Lagrange approach, that have no real indication of the directionality of the information exchange between the various components of the real system and thus it is better to avoid the use of predefined relation between the systems. Using an acausal software based on the Modelica language reflects the nature of the real system.

Models designed using causal software such as Simulink provide a clear graphic visualization of individual mathematical relationships. Signals flow in connections between the individual blocks, in which the processing of input information to output information takes place. Thus, the way the blocks are connected in such software represents the calculation process rather than the actual structure of the modeled reality. Acausal notation of models means that individual components of the model describe the equations directly and not the algorithm of their solution.

The equations in Modelica do not express assignment but rather a relationship between variables. The numerical solvers are intrinsically causal; because causality is missing from the model description, it must be determined by the simulator on the aggregate system.

2.2.3. FMI and FMU Standard

An open-source standard called the FMI was created to interchange dynamic simulation models. The initial iteration of the FMI standard was created as part of the MODELISAR ITEA2 project with the goal of creating a uniform interface and standard to make it easier for suppliers to interchange models, regardless of the simulation language and tools used. Since 2012, the Modelica Association has been responsible for the FMI project development.

The file format specified by the FMI standard that holds the simulation model is called a Functional Mock-up Unit (FMU). There are two types of FMUs that are specified by the FMI standard:

- **FMU for Model Exchange:** contains the model of the dynamical systems with a collection of discrete equations and differential algebraic equations with time state and step events. The solver is provided by the importing tool (OpenModelica, Dymola or any software found in the FMI-supported software list [9]) rather than

being incorporated by the model exchange FMU. Because of this, the user has the ability to select the solution and time step on his own. The FMU for model exchange is more suited to be used in embedded control systems on microprocessors since it only provides the model and the differential equation regulating it. This allows for greater freedom in the usage of the model. The importing tool must link the FMU to a numerical solver in order to simulate the system. The solver establishes the internal state of the FMU, requests the state derivatives, chooses the step size, and decides how to compute the state at the following time step.

- **FMU for Co-Simulation:** the exporting tool includes and supplies the numerical solver. The time step, inputs, and outputs are all set by the importing tool. Programs adoption of FMU Co-Simulation is greater than that of FMU model exchange, and the exporting software may also have a more effective solver than the importing tool.

The FMU consists of a single zip file extension `.fmu`, which contains all the necessary components to utilize the FMU:

- An XML-file that also contains other model information and the definitions of variables that are available to the outer environment. It is feasible to operate the FMU for the Model Exchange standard without this knowledge. Instead, for the sake of the Co-Simulation, the relevant information regarding the slaves is provided in a slave-specific XML-file together with a number of flags indicating the model's capacity to support complex master algorithms.
- For both the Model Exchange and the Co-Simulation FMUs, the necessary functions are provided as a set of C-functions in the form of source or binary code.
- Additional data such as the model icon, documentation files, and necessary resources, as well as all the DLLs and object libraries may be included in the FMU zip-file.

All information about a model and a co-simulation setup that is not needed during execution is stored in an XML-file called `modelDescription.xml` described by the XML-scheme called `fmiModelDescription`, presented in Figure 2.1.

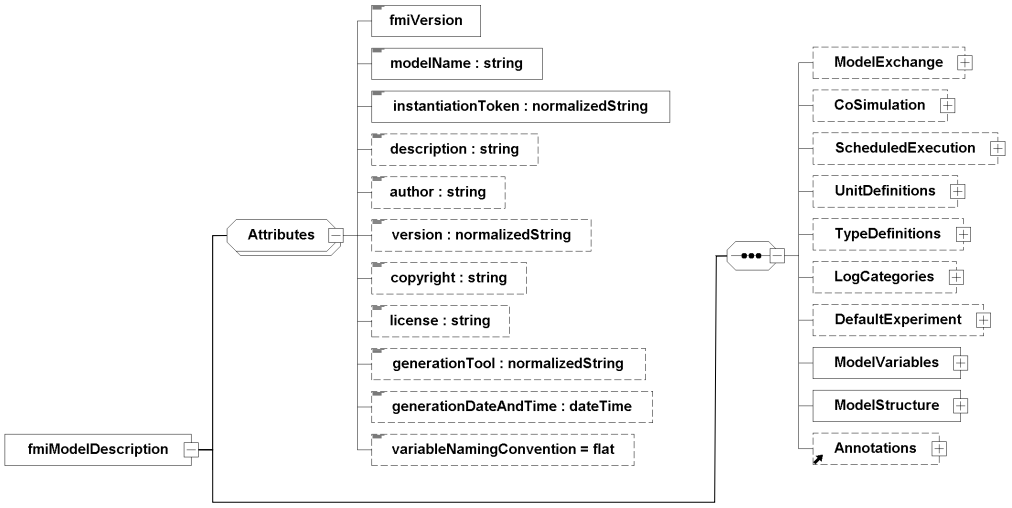


Figure 2.1: *fmiModelDescription* element - XML scheme

The `modelDescription.xml` file is relevant for the work done in this paper since it contains all the Ids with which the variables are called inside the FMU.

After having fully completed the execution of their operations at the predetermined time, both forms of FMU communicate with the simulation environment. Thus, there is no possibility of performing an exchange of information between FMU and the external environment before the integration is performed.

2.2.4. Simulation Platform

The Co-Simulation framework on which the platform will be based is the first choice to be made when creating the model environment. It is a significant choice that will have an impact on the performance, software compatibility, and implementation of the model and filters. Dymola has been selected as the platform for the model and as the FMU importing and exporting tool. It was chosen because it has great compatibility with the MATLAB program and because it has features that will make it simpler to create EKF and MEKF filters.

2.2.5. FMU file structure

After choosing *text visualization*, the user is presented with a Modelica language code that contains the instructions needed to execute the FMU properly once it has been loaded into Dymola. As the primary interface that initializes the model, it reads the files that are included with the FMU and instructs it to proceed with the integration. It does

not, however, contain the mathematical model by itself; that information is found in the binary code file. It is pertinent to this thesis' goal to briefly describe the Modelica language code's structure.

Algorithm 2.1 FMU Structure

```
1: model Name_of_FMU
2: protected parameter, protected constant
   {Definition of parameters, constants and variables that will be saved at each time step of the FMU}
3: package fmi_Functions
   {Section in which all the functions that will be called are defined, it is possible to add user-defined functions}
4: public
   {Section containing the parameters, constants and variables needed and defined by the user to implement the wanted modification inside the FMU}
5: Modelica.inputs, Modelica.outputs
   {Section containing inputs and outputs of the FMU}
6: initial algorithm
   {Section containing the algorithm and equation that are needed only at the first time step to initialize FMU}
7: algorithm
   {Section containing all the algorithm that needs to be performed before the FMU advances in time}
8: Step Foward with DOSTEP
   {Command that informs the FMU to advance in time with one step}
9: algorithm
   {Algorithm and equation performed after do step event}
10: equation
   {Equation needed to assign the user-defined outputs of the FMU}
```

Inside the algorithm 2.1 the user has the opportunity to modify the base file generated automatically during the import process in Dymola. Sections 4 through 10 of the code received the most changes in the work that followed.

2.3. Denavit-Hartenberg parameters

A systematic, general method must be derived to define the relative position and orientation of two successive links to compute the direct kinematics equations for an open-chain manipulator using the recursive expression $\mathbf{T}_n^0(q) = \mathbf{A}_1^0(q_1)\dots\mathbf{A}_n^{n-1}(q_n)$. The task involves identifying the frames attached to each link and determining a recursive formulation to derive the transformations between them.

As long as the frames are connected to the referenced link, they can generally be picked

randomly. However, it is practical to establish certain guidelines for defining the links frame as well [21]. The frame attached to each link is defined as:

- Select axis z of frame i along the axis of joint $i+1$, which is the joint connecting link i with the link $i+1$.
- Position origin O_i at the intersection of the minimum distance segment between z_{i-1} and z_i. The minimum distance segment between two lines can be defined as the common normal. Find Oi' where the common normal meets axis z_{i-1}.
- Choose axis x_i along the common normal to axes z_{i-1} and z_i with positive direction from Joint i to Joint $i+1$.
- The axis y_i is then defined to obtain a right-hand frame.

In the following scenarios, the Denavit-Hartenberg convention defines the link frame that is not unique:

- Only the direction of axis z_0 needs to be specified for Frame 0; the values for O_0 and x_0 are left up to the user's discretion.
- Regarding the last manipulator frame, since there is no $n+1$ link, the axis z_n can be arbitrarily chosen. Since the previous joint is revolute for most manipulator applications, the z axis of frame n is aligned with the direction of the same axis of frame $n-1$.
- The common normal between two consecutive axes cannot be precisely determined when the axes are parallel to one another.
- At the point where two successive axes meet, the direction of x_i can be picked randomly.

In any of these circumstances, indeterminacy can be utilized to simplify the approach; for instance, subsequent frames' axes can be aligned.

After the link frames have been constructed, the following parameters will completely specify the location and orientation of frame i concerning frame $i-1$:

a_i represent the distance between the origins O_i and $O_{i'}$,

d_i represent the distance between the origins $O_{i'}$ and O_{i-1},

α_i angle between axes z_{i-1} and z_i about axis x_i, which must be taken positive when rotation is made counter-clockwise,

θ_i angle between axes x_{i-1} and x_i about axis z_{i-1}, which must be taken positive when rotation is made counter-clockwise.

Two of the four parameters (a_i and α_i) are always constant and rely solely on the geometrical connection made by link i between succeeding joints. Only one of the remaining two characteristics varies based on the kind of joint that links link $i-1$ to link i. More specifically:

- if Joint i is revolute the variable is θ_i,
- if Joint i is prismatic the variable is d_i.

At this point in the process, it is possible to express the coordinate transformation that occurs between Frames i and $i-1$ following the steps that are listed below:

- Choose a frame that is in perfect alignment with the one called $i-1$.
- Move the newly defined frame by d_i along axis z_{i-1}, then rotate it by θ_i about axis z_{i-1}; this procedure allows to align the current frame with Frame i' and is described by the homogeneous transformation matrix

$$\mathbf{A}_{i'}^{i-1} = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

- Transport the frame aligned with Frame i' by a_i along axis $x_{i'}$ and rotate it by α_i about axis $x_{i'}$; thus the current frame is being aligned with the frame i, such operation is described by the following homogeneous transform

$$\mathbf{A}_{i}^{i'} = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

- The overall coordinate transformation can be defined with just one homogeneous transform obtained by post-multiplying the two transformation matrices introduced

above.

$$\mathbf{A}_i^{i-1}(q_i) = \mathbf{A}_{i'}^{i-1}\mathbf{A}_i^{i'} = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

Notice that the transformation matrix from frame i to frame $i-1$ is a function only of the joint variable q_i, that is, θ_i for a revolute joint or d_i for a prismatic joint.

According to the Denavit-Hartenberg convention, the direct kinematics function can be constructed by combining the distinct coordinate transformations indicated by eq. (2.15) into a single homogeneous transform matrix. Any open kinematic chain can use the process, and it is simple to rewrite it in the following operating form:

1. Identify and enumerate each joint orderly, at the same time, identify the direction of the axes $z_0, z_1, \ldots, z_{n-1}$.
2. The Frame 0 is defined by identifying the origin position along axis z_0, and then the other axes are chosen to obtain a right-handed frame. If possible, choosing frame 0 to coincide with the base Frame allows skipping one transformation matrix product.

Then, the subsequent step needs to be executed for $n-1$ times.

3. Locate the origin O_i at the common normal to axes z_{i-1} and z_i. If the two joint axes are parallel and the joint i is revolute, its origin is positioned in order to have $d_i = 0$. If joint i is prismatic, the origin is placed at the reference point from which the mechanical limit of the joint is derived, e.g., the maximum range.
4. Choose an axis x_i along the common normal to axes z_{i-1} and z_i with direction from Joint i to Joint $i+1$.
5. Choose axis y_i to obtain a right-handed frame.

At the end of this iterative process, the following steps are performed:

6. The frame n can be derived considering if the final joint is revolute; if it is true, the axis z_n is aligned with z_{n-1}; otherwise the axis z_n can be picked arbitrarily. Axis x_n is defined in the same way as for the other frames.
7. The Denavit-Hartenberg parameters $a_i, d_i, \alpha_i, \theta_i$ are used to compute the homogeneous transformation matrices $\mathbf{A}_i^{i-1}(q_i)$ for $i = 1, \ldots, n$.
8. Compute the homogeneous transformation $\mathbf{T}_n^0(q) = \mathbf{A}_1^0\mathbf{A}_n^{n-1}$ that yields the position and orientation of Frame n with respect to Frame 0.

9. Calculate the direct kinematics function as $\mathbf{T}_e^b(q) = \mathbf{T}_0^b \mathbf{T}_n^0 \mathbf{T}_e^n$ to determine the position and orientation of the end-effector frame in relation to the base frame given the values of $\mathbf{T}_0^b$ and $\mathbf{T}_e^n$.

The final step in computing the orientation is to evaluate the first two unit vectors of the end-effector frame using the simplest expression and then create the third vector by multiplying the first two.

2.4. Kalman filtering

The Kalman filter [12] is an implementation of the predictor-corrector type estimator that is *optimal* in the sense that it minimizes the estimated error covariance. The aim of a Kalman filter implementation is to estimate the state $x \in \Re^n$ of a discrete-time system, which can be described by the linear equation:

$$\begin{aligned} x_k &= \mathbf{A}x_{k-1} + \mathbf{B}u_k + w_{k-1} \\ z_k &= \mathbf{H}x_k + \nu_k \end{aligned} \tag{2.16}$$

Where the $n \times n$ matrix $\mathbf{A}$ relates the state at the time t_{k-1} to the state at time t_k. The $n \times l$ matrix $\mathbf{B}$ relates the input $u \in \Re^l$ to the states x and the matrix $H_{[m \times n]}$ derives the measurement vector z_k from the state. The variables w_k and ν_k are the process noise and the measurement noise, respectively. They are considered to be white Gaussian noises with zero mean and noise covariance matrices $\mathbf{Q}$ and $\mathbf{R}$, respectively.

It is required to introduce some helpful notation in order to explain a Kalman filter's operation correctly. As a form of a predictor-corrector estimator, the Kalman filter requires the definition of $\hat{x}_{k|k-1}$ as the predicted state estimate of the filter at step k and $\hat{x}_k$ as the corrected state estimate at step k. Now it is possible to proceed with the filter derivation. The estimate error and the estimate error covariance can be defined as:

$$\begin{aligned} e_k &= x_k - \hat{x}_k \\ \mathbf{P}_k &= E[e_k e_k^T] \end{aligned} \tag{2.17}$$

The formulation of the Kalman filter is derived from the Bayesian estimation theory, in particular, it relies on the concept of conditional probability eq. (2.18) and on the Bayes Theorem eq. (2.2).

$$P(A \setminus C) = \frac{P(A \cap C)}{P(C)} \tag{2.18}$$

Theorem 2.2. *For two events A and $B \in C$ with $P(A), P(B) \neq 0$ it holds that*

$$P(A \setminus B) = \frac{P(B \setminus A)P(A)}{P(B)}$$

The equation (2.18) states that the conditional probability of A given C is defined as the probability of the intersection of the two events divided by the probability of event C. It is possible to extend the Bayes theorem to obtain the conditional probability density:

$$f_p(q_1 \setminus q_2) = \frac{f_p(q_2 \setminus q_1) f_p(q_1)}{f_p(q_2)} \tag{2.19}$$

Using the Bayes estimation theory introduced above, it is possible to derive the optimal one-step-ahead predictor, considering the derivation of the state predictor:

$$\hat{x}_{k+1|k} = E\left[x(k+1) \setminus y^k\right] = E\left[x(k+1) \setminus y^{k-1}\right] + E\left[x(k+1) \setminus e(k)\right] \tag{2.20}$$

Now it is possible to continue the computation of the two elements constituting the RHS[1] of the equation (2.20). The first term is developed as:

$$E\left[x(k+1) \setminus y^{k-1}\right] = E\left[x(k) \setminus y^{k-1}\right] + E\left[w(k) \setminus y^{k-1}\right] = \mathbf{A}\hat{x}_{k|k-1} \tag{2.21}$$

where $E\left[w(k) \setminus y^{k-1}\right]$ is equal to zero. Then the second element of the RHS can be computed:

$$\begin{aligned} &E\left[x(k+1) \setminus e(k)\right] = E\left[x(k+1)e^T(k)\right]\left(E\left[e(k)e^T(k)\right]\right)^{-1} \\ &\text{where} \\ &E\left[x(k+1)e^T(k)\right] = \mathbf{A}\mathbf{P}_k\mathbf{H}^T \\ &E\left[e(k)e^T(k)\right] = \mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R} \end{aligned} \tag{2.22}$$

where the P_k is the covariance matrix of the state error and the innovation e, that differs from the error e_k, is defined as:

[1] RHS stands for the right-hand side of the equation

$$e = z(k) - \hat{z}(k \setminus k-1) = z(k) - \mathbf{H}\hat{x}_{k|k-1} \tag{2.23}$$

In equation (2.22), the vector Bayes rule has been applied to derive the results of $E\left[x(k+1)e^T(k)\right]$. Finally, the results from equations (2.21) and (2.22) are substituted inside eq. (2.20) to obtain :

$$\hat{x}_{k+1|k} = \mathbf{A}\hat{x}_{k|k-1} + \mathbf{A}\mathbf{P}_k\mathbf{H}^T\left(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R}\right)^{-1} e(k) \tag{2.24}$$

Deriving the optimal filtering equations is the final stage, and it is simple to do so because it just requires estimating the present step rather than forecasting the future one. Assuming that $\mathbf{A}$ is invertible, the optimal filter state equation is:

$$\hat{x}_k = A^{-1}\hat{x}_{k+1|k} \tag{2.25}$$

substituting in eq. (2.24):

$$\hat{x}_k = \hat{x}_{k|k-1} + \mathbf{P}_k\mathbf{H}^T\left(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R}\right)^{-1} e(k) \tag{2.26}$$

The covariance matrix needs to be updated at each step, the update equation is again derived from Bayes estimation theory and results in:

$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q} - \mathbf{A}\mathbf{P}_k\mathbf{H}^T\left(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R}\right)^{-1}\mathbf{H}\mathbf{P}_k\mathbf{A}^T \tag{2.27}$$

Equation (2.27) is also known as the Riccati equation.

To summarize, the *a posteriori* state estimate $\hat{x}_k$ can be calculated as a linear combination of the *a priori* state estimate and a weighted difference between the true and expected measurements. The *a posteriori* state estimate eq. (2.28) is derived from the Bayes estimation theory. The *a priori* estimate $\hat{x}_{k|k-1}$ is conditioned on all the prior measurements z_k.

$$\hat{x}_k = \hat{x}_{k|k-1} + \mathbf{L}_k(z_k - \mathbf{H}\hat{x}_{k|k-1}) \tag{2.28}$$

The $n \times m$ matrix L_k is the Kalman gain and the aim is to compute it in order to minimize the error covariance. The minimization is actually accomplished by substituting equation (2.28) into the covariance error formulation obtained in equation (2.17). The K that

minimizes error covariance results to be equal to:

$$\mathbf{L}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T \left(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R}\right)^{-1} \tag{2.29}$$

Finally, it is possible to describe the general discrete Kalman filter algorithm.

Algorithm 2.2 KF Algorithm

Input: real measurements z_k
Output: state estimate $\hat{x}_k$
{Time update: predictor section}
1: $\hat{x}_{k|k-1} = \mathbf{A}\hat{x}_{k-1} + \mathbf{B}u_{k-1}$
2: $\mathbf{P}_{k|k-1} = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}$
{Measurements update: corrector section}
3: $\mathbf{L}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T \left(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R}\right)^{-1}$
4: $\hat{x}_k = \hat{x}_{k|k-1} + \mathbf{L}_k \left(z_k - \mathbf{H}\hat{x}_{k|k-1}\right)$
5: $\mathbf{P}_k = \left(\mathbf{I} - \mathbf{L}_k\mathbf{H}\right)\mathbf{P}_{k|k-1}$

The discrete Kalman filter requires at each time step the measurement of the real process z_k and provides the estimated state $\hat{x}_k$. The time update section performs the projection of the previously computed state and covariance to the new time step. The first task in the measurement update section of 2.2 is to compute the Kalman gain, L_k, which minimizes the error covariance matrix. Then *a priori* estimate is updated by incorporating the new measurements, as described in eq. (2.28). The final stage is to obtain the error covariance estimate.

The Kalman filter has a main advantage over other solutions: the recursive procedure makes its implementation much easier and straightforward since, in contrast with Wiener's solution, it does not require the use of all previous measurements to perform each estimate at time k.

It is relevant to briefly mentioned under which assumption the Kalman filter converges. By convergence of the filter it is intended whether the gain reaches a steady value or not. Equation (2.29) shows that the convergence of the gain depends on the convergence of the error covariance matrix. Analyzing the form of the Riccati equation (2.27), it is derived that the system is stable under two conditions:

- **If the system is asymptotically stable** then the steady state predictor is asymptotically stable.
- **If the model is not asymptotically stable** then if the pair (F,H) is observable and the (F,G) pair is reachable, where $Q = GG^T$, then the corresponding steady-

state predictor is asymptotically stable.

2.4.1. EKF

As described above, the Kalman filter addresses the general problem of trying to estimate the state of a discrete-time controlled process that is governed by a linear stochastic difference equation. This severely restricts the use of such a filter because most processes are controlled by nonlinear relations. This issue is resolved by the extended Kalman filter (EKF), which is a Kalman filter that linearizes the system around the current mean and covariance.

The new nonlinear stochastic process that needs to be estimated is:

$$\begin{aligned} x_k &= f(x_{k-1,u_{k-1},w_{k-1}}) \\ z_K &= h(x_k, \nu_k) \end{aligned} \tag{2.30}$$

From such a system it is possible to compute the required Jacobians with a linearization.

$$\begin{aligned} \mathbf{A} &= \left.\frac{\partial f}{\partial x^T}\right|_{(x_{k-1},u_{k-1},0)} \\ \mathbf{W} &= \left.\frac{\partial f}{\partial w^T}\right|_{(x_{k-1},u_{k-1},0)} \\ \mathbf{H} &= \left.\frac{\partial h}{\partial x^T}\right|_{(x_{k-1},0)} \\ \mathbf{V} &= \left.\frac{\partial f}{\partial \nu^T}\right|_{(x_{k-1},0)} \end{aligned} \tag{2.31}$$

In equation (2.31), $\mathbf{A}$ represents the Jacobian matrix of partial derivatives of f with respect to x, $\mathbf{W}$ is the Jacobian matrix of partial derivatives of f with respect to w, $\mathbf{H}$ and $\mathbf{V}$ represent respectively the Jacobian matrix of partial derivatives of h with respect to x and ν.

The new EKF algorithm has the following structure:

Algorithm 2.3 EKF Algorithm

Input: real measurements z_k

Output: state estimate $\hat{x}_k$

{Time update: predictor section}

1: $\hat{x}_{k|k-1} = f(\hat{x}_{k-1}, u_{k-1}, 0)$

2: $\mathbf{P}_{k|k-1} = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^T + \mathbf{W}_k \mathbf{Q}_k \mathbf{W}_k^T$

{Measurements update: corrector section}

3: $\mathbf{L}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}^T + \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^T \right)^{-1}$

4: $\hat{x}_k = \hat{x}_{k|k-1} + \mathbf{L}_k \left(z_k - h(\hat{x}_{k|k-1}, 0) \right)$

5: $\mathbf{P}_k = \left(\mathbf{I} - \mathbf{L}_k \mathbf{H}_k \right) \mathbf{P}_{k|k-1}$

Differently from the KF, there is no guarantee on the convergence of the EKF. [2]

2.4.2. Multiplicative Extended Kalman Filter

The MEKF is an important modification of the EKF that makes it preferable in the implementation for quaternion based orientation estimation. As described in the section above in equation (2.28), the *a priori* state correction is performed by adding a non-zero quantity to the state vector $\hat{x}_{k|k-1}$. Since the state vector, for the purpose of this work, is a unit quaternion, the addition of the term $L_k y_k$ will make it non-unitary and thus it will not represent a rotation. This is because, by definition, rotations quaternions are not vectors; rather, they are described more precisely as groups. Kalman Filtering techniques were initially created for vector applications. Since groups are only closed under their particular group operations, which do not include vector addition, they are not recognized as acceptable Kalman filter states.

In order to recover the unit constraint, one straightforward approach to this issue is to normalize the *a posteriori* correction [14]. The error covariance won't have a clear physical meaning, which is the fundamental drawback of this method.

A simple but yet effective solution to this problem was first introduced in the 60s by NASA. The orientation estimation is no longer captured by a single entry in the state vector. The attitude is represented with two separate parameters, $\hat{q}$ representing the accumulated orientation estimate and a separate small angle error vector $\delta q(\alpha)$. From the quaternions rules:

$$\delta q = [\delta \rho^T; \delta q_4] = \hat{q}^{-1} \otimes q \tag{2.32}$$

[2] With a set of strong assumptions, some researcher was able to establish an analytical solution for the convergence of the EKF.

Because the error is expressed in terms of the group operation, the unitary constraints are maintained without the need for any unnecessary normalization, which is one major distinction between this error operation and native error correction.

In spacecraft missions, the angular velocity ω is measured by gyroscopes. The observation model of the gyroscope is modelled by:

$$\begin{cases} \tilde{\omega} = \omega + b + \nu_v \\ \dot{b} = \nu_u \end{cases} \tag{2.33}$$

where b represents the bias drift and the terms ν_v and ν_u are zero-mean white Gaussian noise. The attitude of the spacecraft is measured by a star tracker that provides directly a quaternion as measurement $\tilde{q}^{STT}$.

In the gyro-stellar attitude estimation problem the state and measurements are described by $X = [q^T; b^T]$ and $\tilde{Y}_k = \tilde{q}_k^{STT}$.

With the MEKF, the error state is redefined as $\delta X = [\delta\rho^T; \Delta b^T]$ along with $\Delta b = b - \hat{b}$. The associated state covariance matrix is defined as: $P = E\left\{\delta X \delta X^T\right\}$.

MEKF differs from the EKF algorithm in the computation of the innovation and in the corrected state computation. The innovation at each time step is derived by applying eq. (2.32) as:

$$\begin{aligned} \delta q_k &= \hat{q}_{k|k-1}^{-1} \otimes y_m \\ Z_k &= \delta\rho = [\delta q_1, \delta q_2, \delta q_3]_k^T \end{aligned} \tag{2.34}$$

The measurement update can be computed as:

$$\begin{cases} \delta q_k(\delta\rho_k) = \left[\delta\rho_k^T; \sqrt{1 - \delta\rho_k^T \rho_k}\right] \\ \hat{q}_k = \hat{q}_{k|k-1} \otimes \delta q_k(\delta\rho_k) \\ \hat{q}_k = \hat{q}_k / \|\hat{q}_k\| \\ \hat{b}_k = \hat{b}_{k|k-1} + \Delta b_k \end{cases} \tag{2.35}$$

By applying all the above modifications the EKF algorithm 2.3, the MEKF algorithm is derived:

Algorithm 2.4 MEKF Algorithm

Input: real measurements q_k^{STT}

Output: quaternion estimate $\hat{q}_k$

{Time update: predictor section}

1: $\hat{X}_{k|k-1} = \breve{\mathbf{\Phi}}_{k-1}\hat{X}_{k-1}$

2: $\mathbf{P}_{k|k-1} = \mathbf{\Phi}_k\mathbf{P}_{k-1}\mathbf{\Phi}_k^T + \mathbf{Q}_k$

{Measurements update: corrector section}

3: $\mathbf{H}_k = [\mathbf{I}_3, \mathbf{0}_{3\times3}]$

4: $\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k$

5: $\mathbf{L}_k = \begin{bmatrix}\mathbf{L}_{a,k}\\ \mathbf{L}_{b,k}\end{bmatrix} = \begin{bmatrix}\mathbf{S}_k^{-1}\mathbf{P}_{aa,k|k-1}\\ \mathbf{S}_k^{-1}\mathbf{P}_{bb,k|k-1}\end{bmatrix}$

6: $\hat{q}_{k|k-1}^{-1} \otimes \tilde{Y}_k \rightarrow Z_k$

7: $\delta X_k = \mathbf{L}_k Z_k$

8: $\hat{X}_k = \begin{bmatrix}\hat{q}_k\\ \hat{b}_k\end{bmatrix} = \begin{bmatrix}\hat{q}_{k|k-1} \otimes \delta q_k(\delta\rho_k)\\ \hat{b}_{k|k-1} + \Delta b_k\end{bmatrix}$

In algorithm 2.4, the matrices $\breve{\mathbf{\Phi}}$ and $\mathbf{\Phi}$ represent the state transition matrices used respectively to propagate the state and to correct the error covariance.

2.5. Constrained parameter estimation

This section discusses a general method for parameter estimation. This strategy employs Kalman filtering methods similar to those previously discussed. In particular, the EKF algorithm will be modified to enable the estimation of the relevant parameters. The MEKF is not appropriate for this application because its goal was to apply filtering techniques to quaternions where the unitary constraints are broken by the *a posteriori* computation. As a result, the parameter vector can be employed directly with equation (2.28).

By making the assumption that parameters θ are stable over time or change slowly, equation (2.36) models them as constants with a minor perturbation:

$$\theta_k = \theta_{k-1} + r_{k-1} \tag{2.36}$$

The noise r_{k-1} is *fictitious*, in the sense that it does not represent any real system noise nor measurement noise but its aim is to model the slow drift in the system parameters plus the inaccuracy of the model structure used. The noise is characterized by zero mean and an assigned auto-covariance $\mathbf{Q}_k^\theta$, which can vary with time.

There are two viable strategies to implement the parameter estimation process using EKF:

- **Dual estimation** consists of implementing two different Kalman filters to perform the state estimation and the parameter estimation respectively. The computational complexity of each filter is smaller and usually leads to better-conditioned matrices. The cross-correlation between changes in states and parameters is lost when parameters and state are separated, though.
- In **joint estimation**, the state vector and parameter vector are combined and a single Kalman filter simultaneously estimates their values. It has the disadvantages of working with large matrix operations and potentially poor numeric conditioning.

The choice to use two filters operating simultaneously, as mentioned for the dual estimation technique, was made since in this study the filters will be implemented inside an FMU with a preset structure and rules. This is mostly due to the fact that it is advantageous to build a solution with great flexibility so that it may be adjusted and deployed in a variety of situations. The dual estimate allows for the development of two distinct filters, one for the state and one for the parameters. Thus the filter for the state can be a different kind of predictor-corrector filter, an Unscented Kalman Filter [3] or, if required, it could be a different kind of filter all together. This method makes it simpler to modify the filter's parameter to achieve a steady behavior because it enables the testing of the filters separately.

The generic algorithm for the parameter estimation filter is:

Algorithm 2.5 EKF Parameter Estimation Algorithm

Input: real measurements z_k

Output: parameter estimate $\hat{\theta}_k$

{Time update: predictor section}

1: $\hat{\theta}_{k|k-1} = \hat{\theta}_{k-1} + r_k$

2: $\mathbf{P}^{\theta}_{k|k-1} = \mathbf{P}^{\theta}_{k-1} + \mathbf{Q}^{\theta}_{k}$

{Measurements update: corrector section}

3: $\mathbf{L}^{\theta}_{k} = \mathbf{P}^{\theta}_{k|k-1}\mathbf{H}^{\theta T}_{k}\left(\mathbf{H}^{\theta}_{k}\mathbf{P}^{\theta}_{k|k-1}\mathbf{H}^{\theta T} + \mathbf{R}^{\theta}_{k}\right)^{-1}$

4: $\hat{\theta}_{k} = \hat{\theta}_{k|k-1} + \mathbf{L}^{\theta}_{k}\left(z^{\theta}_{k} - h^{\theta}(\hat{x}_{k|k-1}, \theta_{k|k-1})\right)$

5: $\mathbf{P}_{k} = \left(\mathbf{I} - \mathbf{L}^{\theta}_{k}\mathbf{H}^{\theta}_{k}\right)\mathbf{P}^{\theta}_{k|k-1}$

Due to the nature of the FMU implementation, a small modification to the dual estimate

[3] The implementation of a UKF filter needs a different implementation in the FMU since it is required to perform multiple integration at time t_k with different initial conditions.

structure presented in figure 2.2 is required.

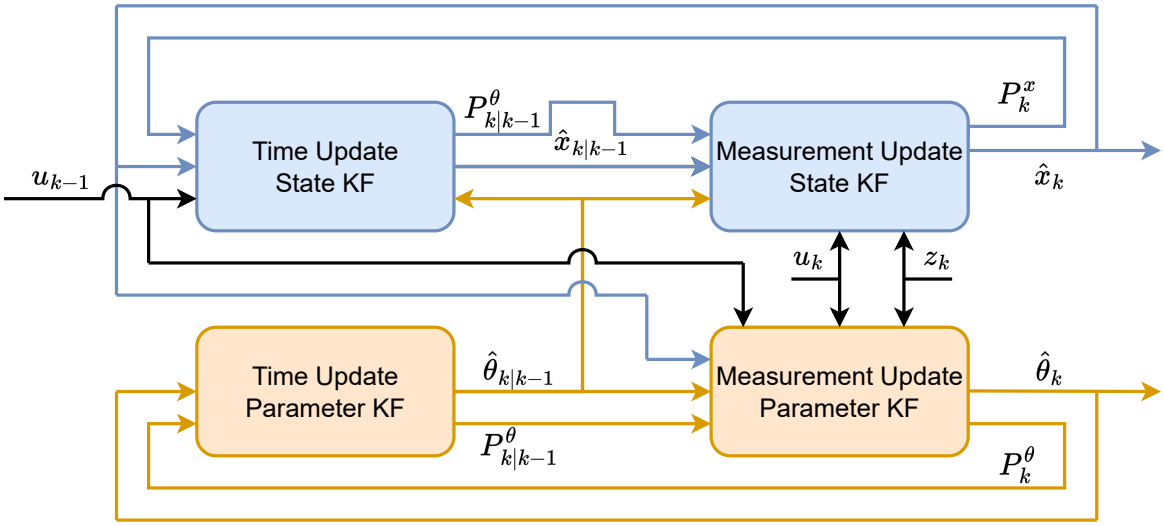


Figure 2.2: Dual Extended Kalman Filter (DEKF) structure

The blue filter represents the filter which aim is to estimate the state, while the filter working on the estimation of the parameter is reported in orange. The information's source is shown by the colored arrows; orange indicates the parameter filter and blue the state filter. As shown in figure 2.2, the two filters run in parallel and they exchange information between the time update and the measurements update steps. As explained in section 2.2.3, the FMU is capable of exchanging information with the simulation environments only at the end of the integration process performed during the call of the FMU [9]. This is relevant because it blocks the possibility of implementing the parallel filter for the estimation of the parameter as reported in Figure 2.2.

The proposed improvement is a straightforward alteration to the way the two filters communicate with one another. As shown in Figure 2.3, the predictor-corrector step can essentially be regarded as a single step, with the filter exchanging information only at the conclusion of the measurement update step. The following chapters will provide a more thorough explanation of how it is implemented within the FMU code.

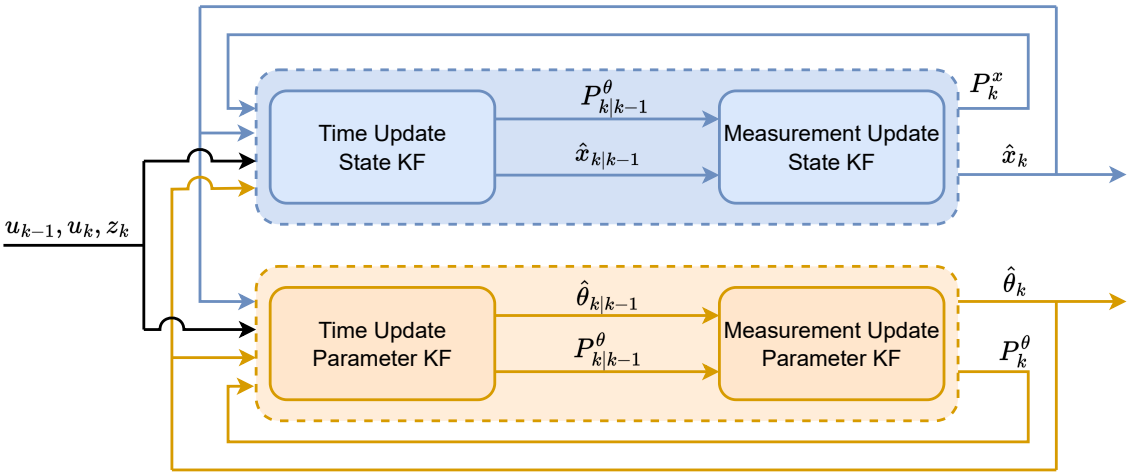


Figure 2.3: Modified dual estimate structure

The parameters of interest have a strict physical meaning, the aim is to estimate the Envisat mass. This parameter is defined as strictly positive, and thus the necessity to impose some constraints on the range in which such parameter is allowed to vary arises. This problem is simply known as constrained parameter optimization. Thus, it is necessary to add such constraints in the formulation of the EKF for parameter estimation. The use of a Sigmoid function [11], equation (2.37), allows for the integration of such constraints without modifying the Kalman filtering structure. The Sigmoid function, first equation in (2.37), is used as a mapping between the state variable of the filter x^{θ}_k to the inertial parameter θ_k.

$$\begin{aligned} \theta_k &= Sig(x^{\theta}_k) = \frac{a^{\theta}_k - b^{\theta}_k}{1 - e^{-cx^{\theta}_k}} = b^{\theta}_k \\ x^{\theta}_k &= Sig(\theta_k)^{-1} = \frac{ln(a^{\theta}_k - \theta_k) - ln(b_k - \theta_k)}{-c} \end{aligned} \tag{2.37}$$

In equations (2.37), a^{θ}_k and b^{θ}_k represent respectively the lower and the upper bounds that the inertial parameter could assume and c controls the slope of the exponential. To initialize the Kalman filter and to update at each step the state of the FMU, the inverse Sigmoid function is used, eq. (2.37). In sections 2 and 3 of 2.6, involving the transformation between inertia parameters and state parameters, different Sigmoid functions equal to the number of the parameters to be estimated are used.

The resulting algorithm is:

Algorithm 2.6 Constrained parameter estimation EKF

Input: real measurements z_k

Output: parameter estimate $\hat{\theta}_k$

{Time update: predictor section}

1: $\hat{\theta}_{k|k-1} = \hat{\theta}_{k-1} + r_k$

2: $\hat{x}^{\theta}_{k|k-1} = Sig(\hat{\theta}_{k|k-1})^{-1}$

3: $\mathbf{P}^{\theta}_{k|k-1} = \mathbf{P}^{\theta}_{k-1} + \mathbf{Q}^{\theta}_{k}$

{Measurements update: corrector section}

4: $\mathbf{H}^{x}_{k} = \mathbf{H}^{\theta}_{k} \frac{c(a^{\theta} - b^{\theta})e^{c\theta_x}}{(e^{c\theta_x}+1)^2}$

5: $\mathbf{L}^{\theta}_{k} = \mathbf{P}^{\theta}_{k|k-1}\mathbf{H}^{xT}_{k}\left(\mathbf{H}^{x}_{k}\mathbf{P}^{\theta}_{k|k-1}\mathbf{H}^{xT} + \mathbf{R}^{\theta}_{k}\right)^{-1}$

6: $\hat{x}^{\theta}_{k} = \hat{x}^{\theta}_{k|k-1} + \mathbf{L}^{\theta}_{k}\left(z^{\theta}_{k} - h^{\theta}(\hat{x}^{\theta}_{k|k-1}, \theta_{k|k-1})\right)$

7: $\mathbf{P}_{k} = \left(\mathbf{I} - \mathbf{L}^{\theta}_{k}\mathbf{H}^{x}_{k}\right)\mathbf{P}^{\theta}_{k|k-1}$

8: $\hat{\theta}_{k} = Sig(\hat{x}^{\theta}_{k})$

The step 4 of 2.6 computes the new $\mathbf{H}$ matrix, derived as:

$$\mathbf{H}^{x} = \frac{\partial h^{\theta}}{\partial x^{\theta}} = \frac{\partial h^{\theta}}{\partial \theta}\frac{\partial Sig(x^{\theta})}{\partial x^{\theta}} = \mathbf{H}^{\theta}\frac{c(a^{\theta} - b^{\theta})e^{c\theta_x}}{(e^{c\theta_x}+1)^2} \tag{2.38}$$

3 | Model implementation and validation

3.1. Euler Lagrange Approach

The model implementation is done using the Modelica language to exploit the usage of acausal connections, as described in sec. 2.2.2. The use of the Euler-Lagrange approach to implement the model was initially considered and then discarded. The motivations behind this choice are explained hereafter.

In order to use the Euler-Lagrange method, the kinetic and potential energies must be derived. Since the hand operation would be laborious and hardly scalable, a symbolic tool is utilized to compute the desired quantities.

To explain why the Euler approach was not used it is introduced a simplified model that just includes the main body of the chaser, the solar panel, and the 7 DoFs manipulator, completely ignoring the presence of Envisat. The kinetic energy has been divided into three parts, each corresponding to an element of the new system.

The kinetic energy of the rigid body of the chaser is reported in equation 3.1.

$$T_c = \frac{1}{2} m_c \dot{r}_c^{oT} \dot{r}_c^o + \frac{1}{2} \omega_c^{oT} \mathbf{R}_c^o \mathbf{I}_c^c \mathbf{R}_c^{oT} \omega_c^o \tag{3.1}$$

Where r_c^o represents the vector connecting the CoG of the chaser to the origin of the reference frame and $\mathbf{R}_c^o$ describes the orientation of the c frame, attached to the satellite body, as seen from the base frame.

Then, the kinetic energy relative to each element of the solar panel is computed as:

$$
\begin{aligned}
T_{sp}^{(i)} &= \frac{1}{2} m_i \dot{r}_c^{oT} \dot{r}_c^o + m_i \dot{r}_c^{oT} \mathbf{S}(\omega_c^o) \mathbf{R}_c^o r_a^c + m_i \dot{r}_c^{oT} \mathbf{S}(\omega_c^o) \mathbf{R}_c^o r_{cg,i}^c + m_i \dot{r}_c^{oT} \mathbf{R}_c^o \dot{r}_{cg,i}^c + \\
&+ \frac{1}{2} m_i r_a^{cT} \mathbf{R}_c^{oT} \mathbf{S}^T(\omega_c^o) \mathbf{S}(\omega_c^o) \mathbf{R}_c^o r_a^c + m_i r_a^{cT} \mathbf{R}_c^{oT} \mathbf{S}^T(\omega_c^o) \mathbf{S}(\omega_c^o) \mathbf{R}_c^o r_{cg,i}^c + \\
&+ m_i r_a^{cT} \mathbf{R}_c^{oT} \mathbf{S}^T(\omega_c^o) \mathbf{R}_c^o \dot{r}_{cg,i}^c + \frac{1}{2} \omega_c^{oT} \mathbf{R}_i^o \mathbf{I}_i^i \mathbf{R}_i^{oT} \omega_c^o + \omega_c^{oT} \mathbf{R}_c^o \mathbf{R}_i^c \mathbf{I}_i^i \mathbf{R}_i^{cT} \omega_i^c + \\
&+ \frac{1}{2} \omega_i^{cT} \mathbf{R}_i^c \mathbf{I}_i^i \mathbf{R}_i^{cT} \omega_i^c + \frac{1}{2} m_i r_{cg,i}^{cT} \mathbf{R}_c^{oT} \mathbf{S}^T(\omega_c^o) \mathbf{S}(\omega_c^o) \mathbf{R}_c^o r_{cg,i}^c + \\
&+ m_i r_{cg,i}^{cT} \mathbf{R}_c^{oT} \mathbf{S}^T(\omega_c^o) \mathbf{R}_c^o \dot{r}_{cg,i}^c + \frac{1}{2} m_i \dot{r}_{cg,i}^{cT} \dot{r}_{cg,i}^c
\end{aligned}
\tag{3.2}
$$

where the operator $\mathbf{S}(-)$ indicates the skew symmetric matrix, the matrix $\mathbf{R}_i^o$ represents the rotation between the base frame and the i-th link and the vector $r_{cg,i}^c$ is the position of the CoG of the i-th one with respect to the CoG of the chaser represented in the body fixed reference frame. The manipulator can be modelled using the DH convection similarly to what is done in equation 3.2. Finally, all the parts can be summed to obtain the total kinetic energy.

$$
T_{tot} = T_c + \sum_{i=1}^{n} T_{sp}^{(i)} + \sum_{j=1}^{7} T_m^{(j)} \tag{3.3}
$$

The kinetic energy is derived laboriously using a large number of rotation matrices, as seen from the equation above. As a result, a system is created with lengthy, complex equations that a symbolic manipulator must handle. The issue arises from the fact that the tool used to derive the Lagrangian and acquire the system equation slows down significantly until it reaches a point where it stops working. This behaviour was seen with a solar panel discretized into just three elements.

As a result, even though the Euler equation of motion represents the smallest number of equations that must be integrated, it is not the best option for modelling complex rigid body systems because the derivation process is time-consuming and needs the use of a symbolic manipulator.

3.2. Model implementation with acausal software

Given the fact that the use of Euler's approach to retrieve the dynamic equations governing the system can be discarded due to the problems highlighted above, a new implementation method is required. The Dymola program based on the Modelica language has been chosen for the use of acausal programming language with all the benefits listed in section 2.2.2. Another advantage of the Dymola program is its comprehensive suite of tools for multi-body modelling.

The model implemented inside Dymola is constituted by the following components:

- Envisat
- 7 Dofs manipulator
- Chaser
- Chaser solar panels

The following section provides a brief description and explanation of each component's construction process.

3.2.1. Envisat

The Dymola scheme reported in figure 3.1 represents the blocks and their connection constituting the Envisat satellite model.

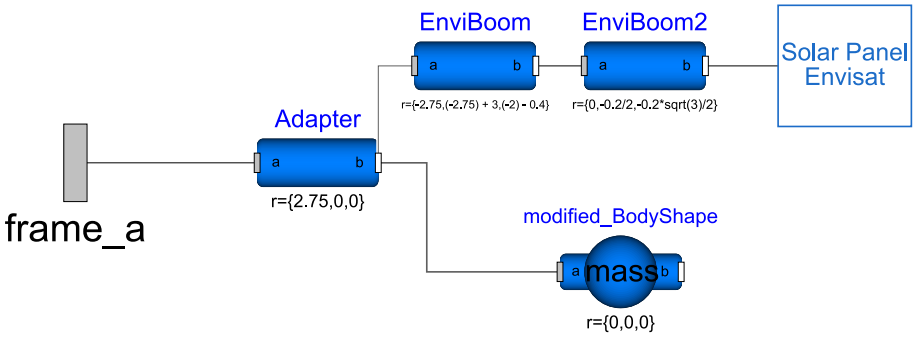


Figure 3.1: Envisat - Dymola scheme

The blocks *Adapter*, *EnviBoom* and *EnviBoom2* are cylindrical blocks of the Modelica multi-body standard library and they are inserted as it is with the specification of dimension and mass of each part. In particular, the *Adapter* element models the launch adapter of the satellite which is the predetermined point on which the manipulator of the chaser satellite will be attached [18]. Inside the Modelica multi-body library, the *BodyShape* element has its inertial properties initialized as parameters and thus they cannot change during the simulation process [9]. Since, for the aim of this work, the Envisat mass and inertia matrices needed to change in time to apply at each time step the predicted value coming from the parameter estimation filter, a new element (*modified_ BodyShape*) is created in which the inertial parameters are defined as variables. The parameters of the Envisat main body with the associated uncertainties are reported in Table 3.1 [18].

Envisat Body	Value	Uncertainty
Dimensions $[m]$	$4.57 \times 10, 5 \times 4.57$	\
Mass $[kg]$	7928	± 30
CoG $[mm]$	$[3905, -9, 3]$	$\pm\,[0.01, 0.004, 0.004]$
Inertia matrix $[kg\,m^2]$	$\begin{bmatrix} 17023 & 397.1 & 2171 \\ 397.1 & 124826 & 344.2 \\ 2171 & 344.2 & 129112 \end{bmatrix}$	$\pm \begin{bmatrix} 350 & 100 & 250 \\ 100 & 3000 & 150 \\ 250 & 150 & 3000 \end{bmatrix}$

Table 3.1: List of Envisat parameters with uncertainties

Inside the *Solar Panel Envisat* block, the iterative procedure to model the solar panel is performed. It has been chosen to write the code in the Modelica language directly (algorithm 3.1) rather than using the graphical interface because it simplifies the implementation and allows for changing of the number of solar panel elements. This approach allows for higher flexibility and an easier change in element size and stiffness properties. Note that since the FMU is pre-compiled in C-code, it is impossible to modify the solar array's discretisation after the compilation and exporting process. The FMU pre-allocates the memory for each required variable. The solar panel model implementation is described inside the algorithm:

Algorithm 3.1 Solar panel implementation in Dymola

Input: N

{Initialization of rigid body elements}

`Modelica.Mechanics.MultiBody.Parts.BodyBox`

{Initialization of Spring-Damper elements}

`Modelica.Mechanics.Rotational.Components.SpringDamper`

{Initialization of rotational joints}

`Modelica.Mechanics.Joints.Revolute`

`connect` base frame with first `joint.frame_a`

for $i = 1 : N - 1$ **do**

  `connect` i-th `revolute.frame_b` with i-th `bodyBox.frame_a`

  `connect` i-th `bodyBox.frame_b` with $(i{+}1)$-th `revolute.frame_a`

  `connect` i-th `springDamper` with i-th `revolute`

end for

`connect` N-th `revolute.frame_b` with N-th `bodyBox.frame_a`

`connect` N-th `springDamper` with N-th `revolute`

Each reference frame attached to each solar panel rigid elements follows the Denavit-Hartenberg parameters explained in section 2.3. In table 3.1, the dimensions, the mass and the inertia matrix of the Envisat with the corresponding uncertainty ranges are reported. In table 3.2 the mass and size of the main solar panel are reported.

Envisat solar array	value
weight $[kg]$	380
width $[m]$	4.97
length $[m]$	15
thickness $[mm]$	12

Table 3.2: Envisat solar panel properties

3.2.2. Chaser

The chaser is modelled as a single rigid body whose dimensions are derived from the e.Deorbit mission concept studies [18]. Table 3.3 shows the dimensions and masses. Since it is not the aim of this work the study of the chaser structure and architecture, a simple design has been modelled.

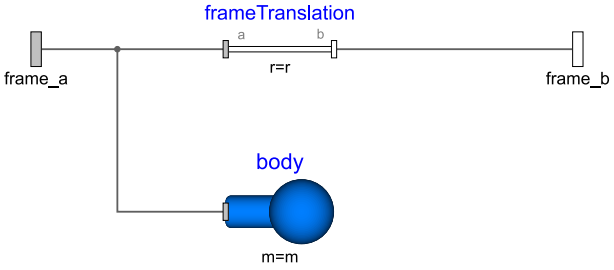


Figure 3.2: Chaser - Dymola scheme

Figure 3.2 shows the insides of the chaser body block. It is modelled as a rectangular parallelepiped. Since all the other blocks will be attached to the chaser body, it is important to take care in the positioning of `frame_a` and `frame_b` with respect to the chaser center of gravity (CoG). The chaser solar array is modelled with the exact same procedure as the one reported in 3.1 with the parameters listed in table 3.3.

Chaser Body	value	Chaser SP	value
weight $[kg]$	1594	weight $[kg]$	30
width $[m]$	1	width $[m]$	1
length $[m]$	1.2	length $[m]$	3
height $[m]$	1.2	thickness $[mm]$	8

Table 3.3: Chaser satellite properties

3.2.3. 7 Dofs Manipulator

The manipulator structure is derived according to previous works that studied the use of a 7 Dofs manipulator like ROKVISS [2]. The model is constructed through the use of revolute joint to which the cylindrical elements that constitute the manipulator links are attached. A spring damper element is assigned to each joint as shown in figure 3.3 with spring stiffness $k_{man} = 1000\ [Nm/rad]$ and damping constant $c_{man} = 0.1\ [Nms/rad]$.

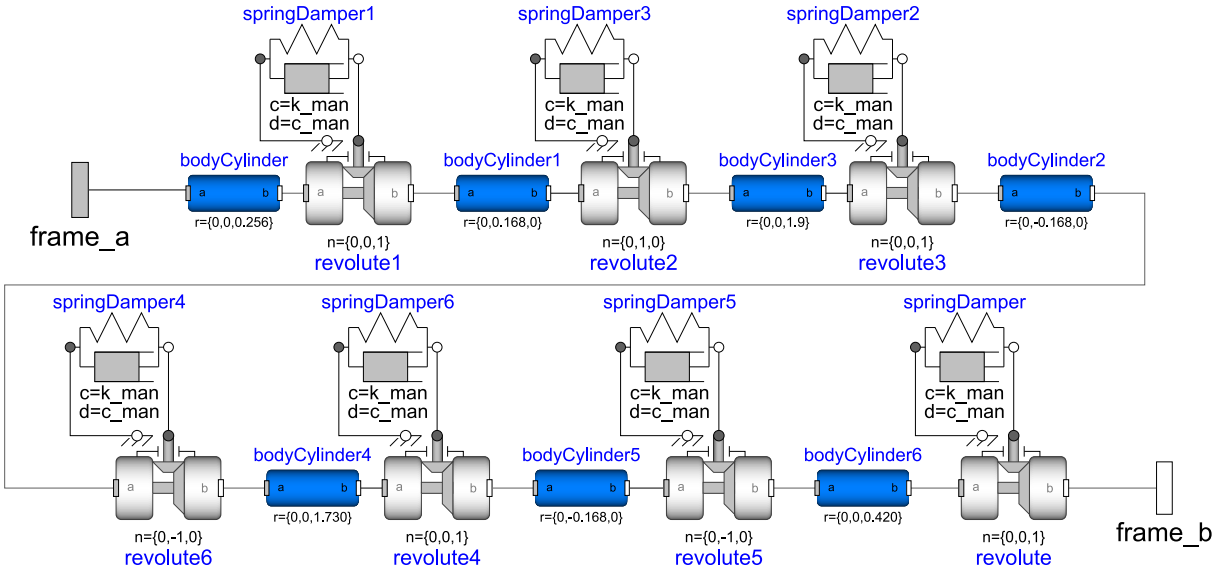


Figure 3.3: 7 Dofs Manipulator - Dymola scheme

The DH parameters listed in table 3.4 describes the manipulator structure. The e.Deorbit studies proposed a clamp system attached at the end of the robotic arm [18]. It is assumed that the manipulator is attached to the Envisat adapter ring with a rigid connection, and thus the modelling of the clamp mechanism was not performed.

Joint	n_1	n_2	n_3	n_4	n_5	n_6	n_7
a $[mm]$	0	0	0	0	0	0	0
α $[deg]$	0	-90	90	-90	90	90	-90
θ $[rad]$	1.57	0.87	6.01	1.83	1.12	1.67	1.70
d $[mm]$	256	168	1900	168	1730	168	420

Table 3.4: Manipulator's DH parameters

The robotic arm structure is made of aluminium and the links have an internal diameter equal to $d_i = 123\ [mm]$ and an external diameter equal to $d_e = 127\ [mm]$.

3.2.4. Complete system model

After explaining each component of the dynamical system, it is possible to build the complete model inside the Dymola environment. Its scheme is reported in figure 3.4. Each component is attached to the main chaser body, to which an external torque is applied to simulate the use of thrusters for attitude control. The block *world* is used to model the Earth's gravitational acceleration; disturbing external torques are not modelled.

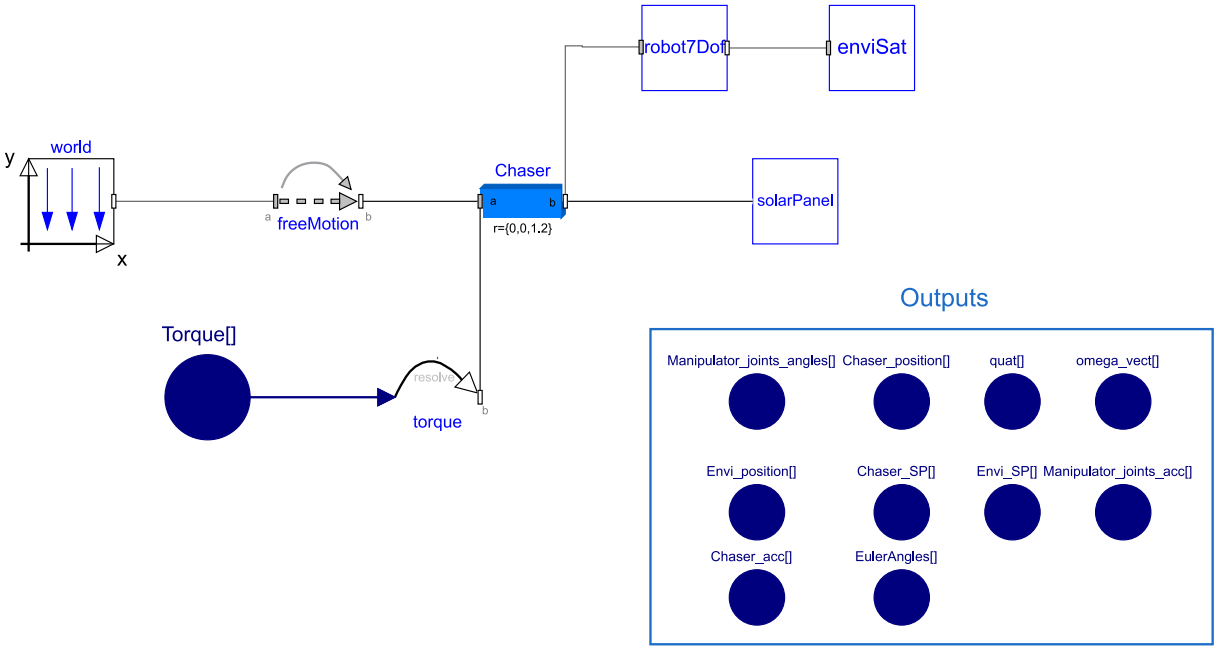
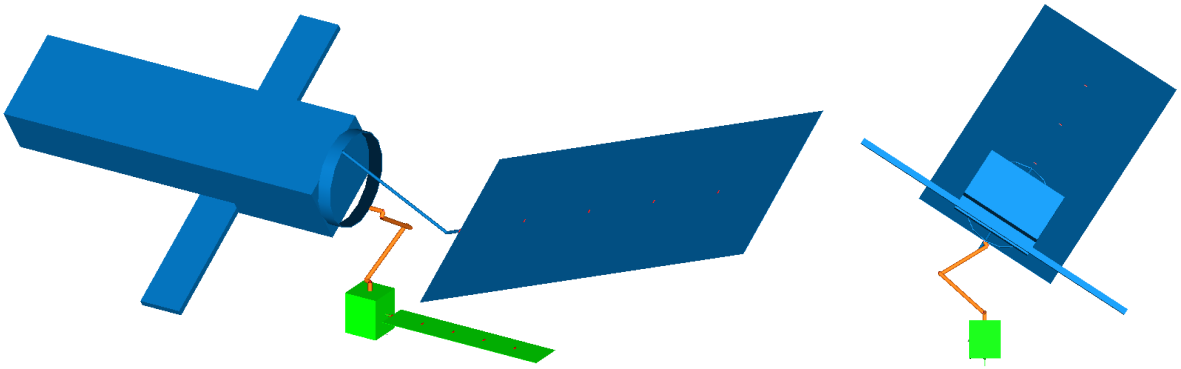


Figure 3.4: Complete system model - Dymola scheme

Inside figure 3.5 a visualization of the two spacecraft assembly is shown. The chaser satellite, represented in green, is connected with Envisat, shown in blue, through the

manipulator in orange. Envisat's solar panels and the chaser satellite's solar array are modelled as described in the section above. From figure 3.5a it is possible to see that the manipulator is attached to Envisat in correspondence with the adapter ring. The adapter ring is used as a grasping point due to its rigidity since it has been sized to withstand the launch loads.



(a) Frontal view (b) Lateral view

Figure 3.5: 3D Model Visualization

3.3. Spring stiffness derivation

Since the solar panels are modelled with a lumped parameter approach, a comparison against the results of a FEM of the solar panel is performed to ensure the validity of the model. The following sections describe the steps to perform such validation.

3.3.1. Linearization of the Solar Panel model

This section describes the derivation of the equations of motion of the solar panel model and their linearization. The following discussion uses the DH convection 2.3 to assign at each link a reference frame with an iterative procedure.

The equations of motion are derived using the Newton-Euler formulation. Considering J_{pi} as the Jacobian matrix for translational motion, J_{oi} as the Jacobian matrix for rotational motion, p_i as linear momentum, l_i as angular momentum, F_i as applied forces, M_i as applied moments for body i, the principles for linear and angular momentum are applied to get equation (3.4).

$$\sum_{i=1}^{n_s} \left[J_{P_i}^T (\dot{p}_i - F_i) + J_{O_i}^T (\dot{l}_i - M_i) \right] = 0 \tag{3.4}$$

To proceed with the mathematical derivation the following notation is used:

- r_s is the position vector from Newtonian frame O_I to body fixed frame S_i;
- r_{pi} is the position vector from Newtonian frame O_I to joint P_i;
- R_i is the rotation matrix representing rotation of frame S_i with respect to frame O_I about the Z axis;
- I_i is the inertia tensor of the link i;
- v_{si} is the linear velocity associated with the center of mass of link i;
- ω_i is the angular velocity of link i.

The derivative of the linear momentum p_i can be rewritten using the above introduced notation as:

$$\dot{p}_i = m_i \ddot{r}_{si} \tag{3.5}$$

Similarly the derivative of the angular momentum is equal to:

$$\dot{l}_i = I_i \dot{\omega}_i + \tilde{\omega}_i I_i \omega_i \tag{3.6}$$

Using the equations (3.5) and (3.6), equation (3.4) can be rewritten in the equation (3.7).

$$\sum_{i=1}^{n_s} J_{Pi}^T \left[m_i \ddot{r}_{s_i - F_i} \right] + J_{O_i}^T \left[I_i \dot{\omega}_i + \tilde{\omega}_i I_i \omega_i - M_i \right] = 0 \tag{3.7}$$

The bodies kinematic equations 3.8 are described by the generalized coordinates $q = (\delta, \phi_1, \phi_2, ..., \phi_{n-1})^T$. These coordinates represent the absolute rotation of each joint of the solar panel model from the original reference condition.

$$
\begin{cases}
R_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\delta) & -sin(\delta) \\ 0 & sin(\delta) & cos(\delta) \end{bmatrix} \\
R_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\delta) & -sin(\delta) \\ 0 & sin(\delta) & cos(\delta) \end{bmatrix} \begin{bmatrix} cos(\phi_i) & -sin(\phi_i) & 0 \\ sin(\phi_i) & cos(\phi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{for} \quad i \neq 1 \\
r_{pi} = r_{pi-1} + R_{i-1}\left[L_s; 0; 0\right], \qquad r_{p1} = [0;0;0] \\
r_{li} = r_{pi} + R_i[L_s/2; 0; 0] \\
J_{Pi} = \dfrac{\partial r_{li}}{\partial q} \qquad v_{si} = J_{Pi}\dot{q} \\
\omega_1 = [\dot{\delta}; 0; 0] \qquad \omega_i = [0; 0; \dot{\phi}_i] \quad \text{for} \quad i \neq 1 \\
J_{Oi} = \dfrac{\partial \omega_i}{\partial \dot{q}} \\
\ddot{r}_{si} = J_{Pi}\ddot{q} + \dfrac{\partial v_{si}}{\partial q}\dot{q} \\
\dot{\omega}_i = J_{Oi}\ddot{q} + \dfrac{\partial \omega_i}{\partial q}\dot{q}
\end{cases}
\tag{3.8}
$$

Using the kinematic equations, 3.7 can be rearranged to 3.9.

$$
\begin{aligned}
T_{li} &= \frac{1}{2} m_{li} \dot{q}^T J_{pi}^T J_{pi} \dot{q} + \frac{1}{2} \dot{q}^T J_{Oi}^T R_i I_i R_i^T J_{Oi} \dot{q} \\
T &= \sum_{i=1}^{n} T_{li} = \frac{1}{2} \dot{q}^T B(q) \dot{q} \\
B(q) &= \sum_{i=1}^{n} \left(m_{li} J_{pi}^T J_{pi} + J_{Oi}^T R_i I_i R_i^T J_{Oi} \right)
\end{aligned}
\tag{3.9}
$$

The only component of the potential energy, needed for the derivation of the spring stiffnesses, is the elastic one.

$$
U_{li} = \frac{1}{2} k_i (q_{i-1} - q_i)^2 \qquad U = \sum_{i=1}^{n} U_{li} \tag{3.10}
$$

From (3.10) and (3.9) the Lagrangian equation is derived.

$$
L(q, \dot{q}) = T(q, \dot{q}) - U(q) \tag{3.11}
$$

Applying the Lagrange formalism to (3.11) the dynamic equation is obtained:

$$\begin{aligned} g(\dot{q},q) &= \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} \\ g(\dot{q},q) &= B(q)\ddot{q} + \dot{B}(q)\dot{q} - \frac{1}{2}\left(\frac{\partial}{\partial q}(\dot{q}^T B(q)\dot{q})\right)^T + \left(\frac{\partial U(q)}{\partial q}\right)^T = 0 \end{aligned} \tag{3.12}$$

Then, through the use of symbolic manipulation, the following equation is derived:

$$f(\dot{q},q) = B(q)^{-1}\left(-\dot{B}(q)\dot{q} + \frac{1}{2}\left(\frac{\partial}{\partial q}(\dot{q}^T B(q)\dot{q})\right)^T - \left(\frac{\partial U(q)}{\partial q}\right)^T\right) \tag{3.13}$$

The equation is reduced to a differential equation of order 1 and then the linearized model is obtained:

$$\dot{x} = \begin{Bmatrix} f(\dot{q},q) \\ \dot{q} \end{Bmatrix} = f_1(\dot{q},q) \quad \text{where} \quad x = \begin{Bmatrix} \dot{q} \\ q \end{Bmatrix} \tag{3.14}$$

$$A = \left.\frac{\partial f_1}{\partial x}\right|_{x=x_0} \tag{3.15}$$

Matrix A in equation (3.15) represents the overall linearized system dynamics. The linearization is performed at the equilibrium point $\dot{q} = 0_{n\times 1}$ and $q = 0_{n\times 1}$. The eigenvalues and eigenvectors of matrix A represent the modal frequencies and modal shapes, respectively, of the linearized system.

3.3.2. Solar panel FEM model

A finite element model (FEM) of the solar panel is constructed and used to retrieve a reference solution for the validation process. As pre and post-processing software Femap is used, while the solver is Nastran.

The FEM model is constructed using plate elements. Regarding the mesh size, the width of the solar panel is discretized with 20×60 elements. These values guarantee convergence of the FEM results. The property of the elements is defined using the PCOMP, where the stacking sequence is defined by providing thickness, orientation and material id of each ply composing the stack. The layup is a sandwich of an aluminium honeycomb core with uni-directional carbon fiber layers for the upper and lower faces. The honeycomb core is implemented as a 2D-orthotropic material with properties reported in equation (3.16).

$$E_m = \begin{bmatrix} E_1 \\ E_2 \end{bmatrix} = \begin{bmatrix} 9900 \\ 10700 \end{bmatrix} [\text{MPa}] \quad G_m = \begin{bmatrix} G_{12} \\ G_{1z} \\ G_{2z} \end{bmatrix} = \begin{bmatrix} 2600 \\ 5500 \\ 5000 \end{bmatrix} [\text{MPa}] \quad \begin{matrix} \nu_m = 0.92 \\ \rho = 2e^{-10}\ [T/mm^3] \end{matrix} \tag{3.16}$$

In eq. (3.16), E indicates the stiffness value, G the shear, ν_m the Poisson ratio and ρ is the mass density of the material. The presence of the solar cell is modelled by increasing the density of the upper elements of the laminate. The layup of the solar panel is reported in figure 3.6.

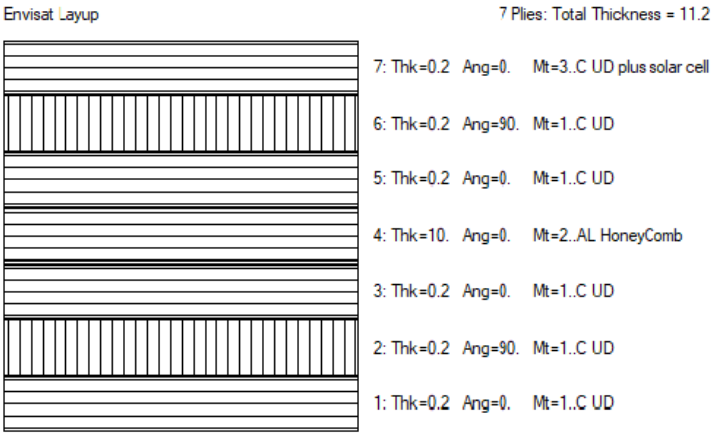


Figure 3.6: Solar panel layup

The solar panel model is constrained on one short edge by fixing all 6 DoFs of the nodes.

3.3.3. Optimization procedure

The eigenvalues and eigenvectors of matrix A, derived from the dynamic model of the solar panel, are a function of the spring stiffness of each joint. A physically correct model would require all stiffnesses to be equal because the bending properties do not change along the length direction. This option was considered initially, but it led to poor frequency derivation and modal shape results. Such a result was probably due to the low number of elements used in the modelling procedure. The number of elements is limited due to the high computational cost associated with the symbolic manipulation of equation (3.15). In particular, the length of each equation composing the matrix A grows rapidly with the number of elements used. The upper bound for the number of discretized sections

was established to be 5. This value is a good compromise between the computational time and the number of frequencies that can be matched. The poor accuracy achieved by using single value for the spring stiffnesses is a known aspect in the literature. For example, it is highlighted in numerous works involving the derivation of a pseudo-rigid body model (PRBM) for beam elements [24]. In particular, the PRBM method is used extensively to synthesize and analyze the behaviour of compliant mechanisms for non-linear deflections. The tuning of the parameters in the PRBM highly depends on the quantity to be optimized. Most works developed a solution for large deflection of cantilever beams to minimize the error between the model and the actual deformed shapes. A similar approach can be used to minimize the modal frequency errors and obtain modal shapes similar to the one achieved using NASTRAN.

In the linearized model derived in section 3.3.1, the elastic constant of the springs is the only parameter allowed to change, differently from the most common implementation of PRBM where both link lengths and stiffnesses are allowed to vary. The system becomes increasingly complex as the number of discrete elements and variables increases. This behaviour stems from the mathematical complexity of matrix A that inevitably slows down the symbolic manipulation.

The derivation and validation of the spring constants are carried out separately for the torsional spring. This procedure is justified by the uncoupled torsional and bending behaviour for the structure at hand.

From the FEM model, the first five flexural bending modes are extracted. An optimization procedure is then carried out to recover the stiffnesses of the lumped spring of the simplified model. The system identification procedure reads:

$$\min_{k \in \mathbb{R}^n} \left(\bar{Y} - \hat{Y}\right)^T \mathbf{W}_w \left(\bar{Y} - \hat{Y}\right) \tag{3.17a}$$

subject to

$$\hat{Y} = g(k) \tag{3.17b}$$

The cost criterion in equation (3.17a) is the weighted sum of squared errors. The weight is applied with the matrix $\mathbf{W}_w$, where each element is equal to the weight squared. The vector $\hat{Y}$ collects the linear frequencies and the shape of the corresponding modes. As described in equation (3.17b), the vector $\hat{Y}$ is only a function of the spring stiffness values. Instead, the vector $\bar{Y}$ vector contains the frequency and the nodal displacement obtained from the FEM analysis.

$$\hat{Y} = \left[f_{est}^{1}, \ldots, f_{est}^{N}, displ^{1}, \ldots, displ^{N}\right] \tag{3.18}$$

In equation (3.18) the element f_{est}^{1} is the first modal frequency derived from the linearized model, and the term $displ^{1}$ represents the nodal displacement of the associated mode. Considering N as the number of elements in which the solar panel is being discretized, the term $displ^{1}$ is a vector of size $N-1$ since the first node is fixed due to the boundary conditions of the problem. The nodal displacement is normalized with respect to the tip displacement and is computed by substitution of the eigenvector in equation (3.8) to obtain r_{pi}. Then each r_{pi} is used to construct $displ^{1}$ (3.19), which is repeated for each modal shape.

$$displ^{1} = \left[r_{p2}, \ldots, r_{pN}\right] / r_{pN} \tag{3.19}$$

The optimization is carried out via a Gauss-Newton method. It is possible to use the Gauss approximation since the cost function has a quadratic form. This allows to provide to the algorithm only the vector F, which is used in the computation of the line search direction and in the calculation of the required step size.

$$F = \mathbf{W}_{w}^{1/2}\left(\bar{Y} - \hat{Y}\right) \tag{3.20}$$

Then, it is possible to compare the results obtained from the optimization procedure with the FEM ones. The weight matrix is changed to get different results in the optimization process. The parameter $\mathbf{W}_{w}$ has the form described in equation (3.21). The value of w is used to modify the relative weights of the frequencies with respect to the modal shapes.

The above mentioned steps are performed three times for every weight change. The first optimization is done by setting all the weights to 1. From the FEM analysis, the modal shape is extracted from the nodes belonging to the longitudinal median line. The data acquired by the finite element method are referred to as target modes or target frequencies in the following discussion.

$$\mathbf{W}_{w} = diag([w_{1\times N}, 1_{1\times(5N-5)}]) \tag{3.21}$$

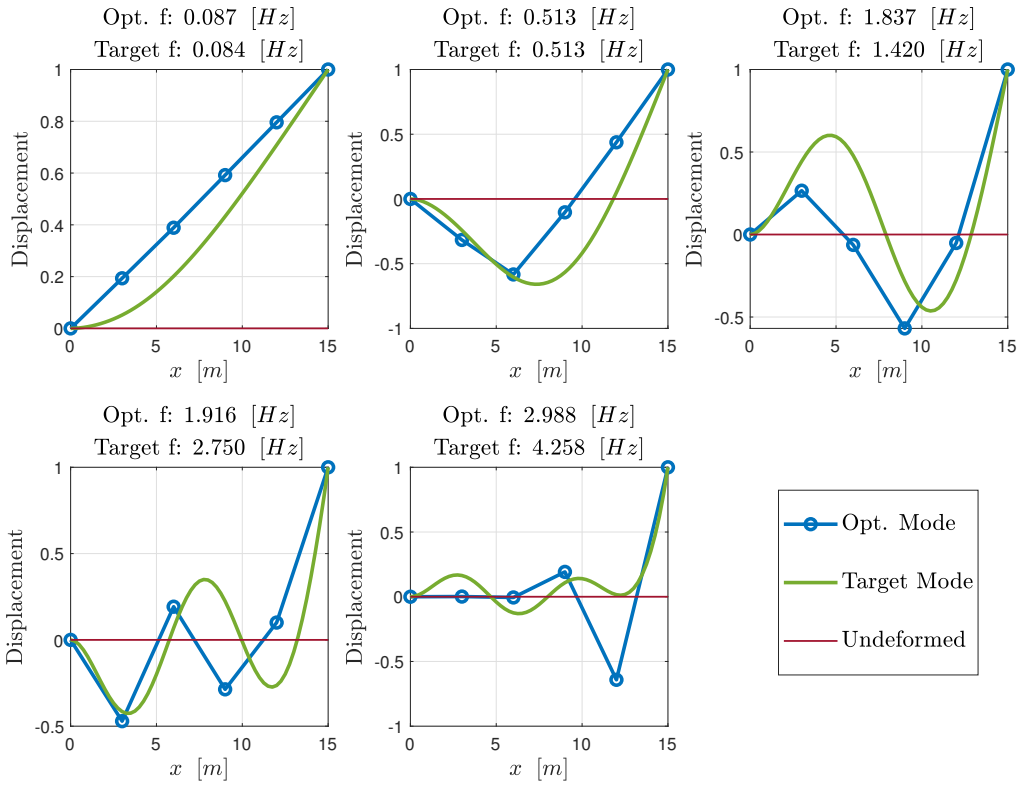


Figure 3.7: Modal shapes and Modal frequencies from optimization with $w = 1$

Run with $w = 1$	k_1	k_2	k_3	k_4	k_5
value $[Nm/rad]$	3080	218023	25123	127773	79382

Table 3.5: Stiffnesses obtained from optimization with $w = 1$

Figure 3.7 reports the modal shapes and frequencies obtained after the optimization procedure. The stiffnesses obtained by setting $w = 1$ are summarized in table 3.5. The results of this first run show that the discretized model match closely the first three modal frequencies, while the other eigenvalues differ with respect to the target one. The mean square error (MSE) of the distances between the node and the target position is determined for the modal shape in order to provide a quantitative evaluation of the results. Each modal shape's mean square error is calculated separately.

For the second optimization, the weight is set to 10. In doing so, the errors due to any frequency mismatch are overweighted. The results are shown in figure 3.8. The first three eigenvalues do not vary significantly compared to the target one since they were already sufficiently close to it. This can be observed by comparing the frequencies with

the ones derived from the initial optimization procedure. The error on the fifth frequency is drastically reduced; the percent error decreases from 30% to 1.5%. At the same time, there are no significant changes to the fourth eigenvalues.

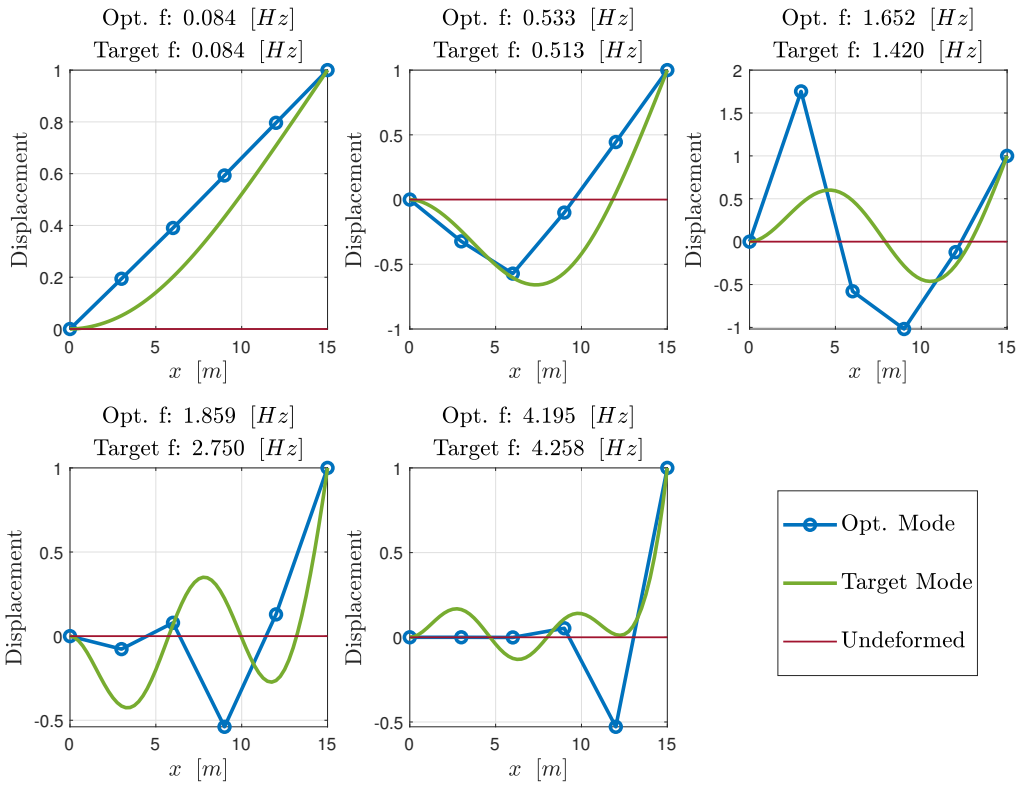


Figure 3.8: Modal shapes and Modal frequencies from optimization with $w = 10$

The results obtained from the second optimization run are reported in table 3.6.

Run with $w = 10$	k_1	k_2	k_3	k_4	k_5
value $[Nm/rad]$	2863	165105	27927	114564	170855

Table 3.6: Stiffnesses obtained from optimization with $w = 10$

The third and final run is performed by shifting the weight on the modal shapes. The results are reported in figure 3.9. With this optimization, the frequencies values are worse with respect to the first run. The percentage error on the first frequency increases from 4% to 57%, and the second differs by 38% compared to 0.3% of the initial optimization. The stiffness values relative to the third optimization step are reported in table 3.7.

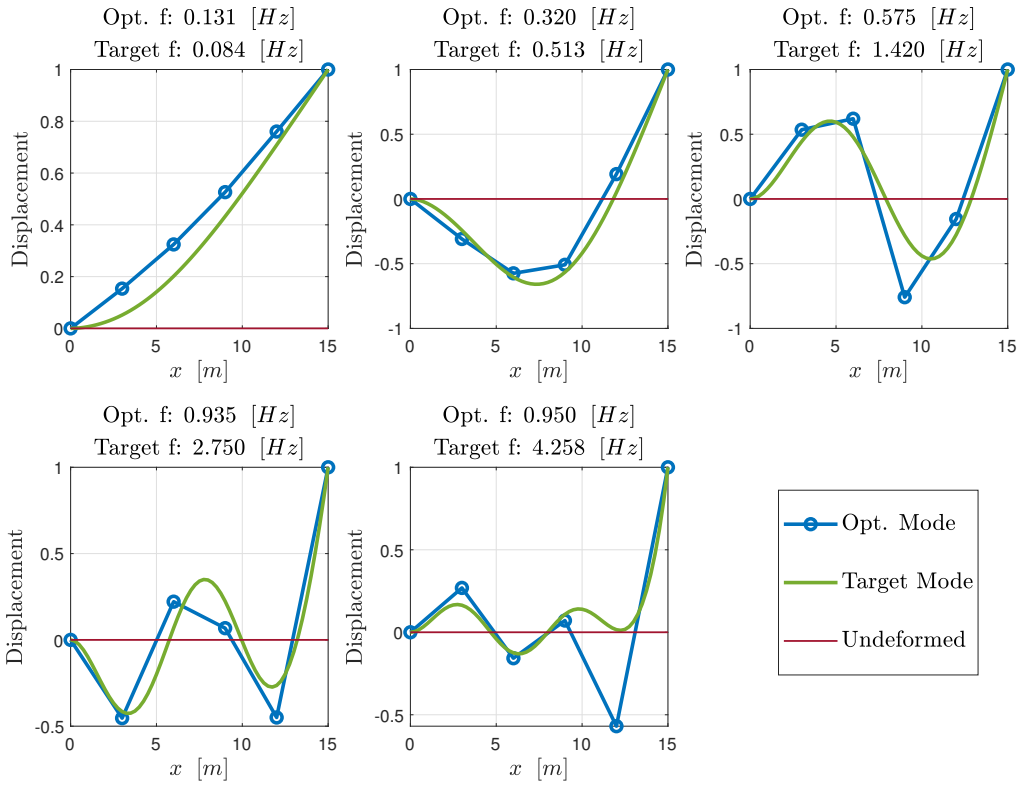


Figure 3.9: Modal shapes and Modal frequencies from optimization with $w = 0.1$

Run with $w = 0.1$	k_1	k_2	k_3	k_4	k_5
value $[Nm/rad]$	8051	47568	15648	6884	8565

Table 3.7: Stiffnesses obtained from optimization with $w = 0.1$

Utilizing the MSE, which serves as the key performance indicator (KPI), it is feasible to compare the correctness of the modal shape following the introduction of all optimization runs. For all modal shapes except the fourth, the optimization carried out with weight $w = 10$ has the highest KPI value, as shown in table 3.8. The lowest KPI for each modal shape is achieved using $w = 0.1$, which closely resembles the modal shapes. As anticipated, the optimization process with run $w = 1$ falls in the middle of the other runs, with frequency values quite near the target values and similar modal shapes, particularly for the first two.

Modal shape	weight $w = 1$	weight $w = 10$	weight $w = 0.1$
n_1	$2.27e-02$	$2.30e-02$	$1.14e-02$
n_2	$8.24e-02$	$8.45e-02$	$6.47e-03$
n_3	$1.18e-01$	$1.65e-01$	$4.00e-02$
n_4	$1.14e-01$	$2.21e-01$	$2.61e-02$
n_5	$1.29e-01$	$1.03e-01$	$8.74e-02$

Table 3.8: MSE of each modal shape for the three optimization runs

Regarding the torsional spring, this value is determined by matching the first torsional frequency, and it is equal to: $k_{tor} = 621.6\ [Nm]$.

3.4. Validation

The validation is performed as a final step in deriving the spring stiffnesses. The process consists of a transient load analysis performed separately on Dymola with the discretized model and on Nastran. The transient load is applied to the tip of the solar panel while one end is fixed, as represented in figure (3.10). The transient load test is run for 100 $[s]$ with a sample time equal to $\Delta t = 0.125\ [s]$ resulting in 800 samples. The resulting discrete Fourier transform has a frequency resolution $\Delta f = 0.1\ [Hz]$, and the maximum frequency is $f_{max} = 4\ [Hz]$. The transient load is a sine sweep function described in equation (3.22).

$$F_{in} = sin\left(\frac{2\pi t^2}{2t_{span}}\right) \quad [N] \tag{3.22}$$

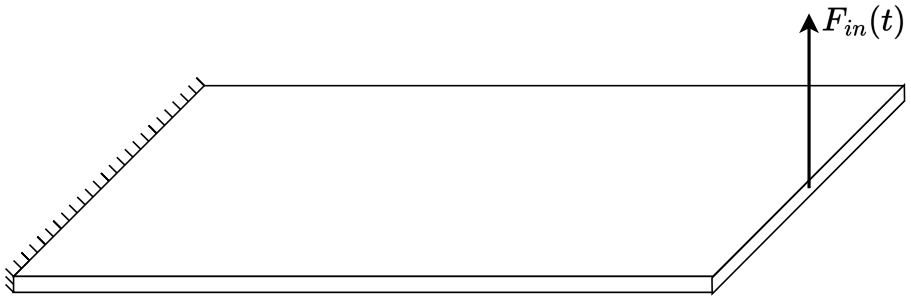


Figure 3.10: Transient load application

The solar panel model is the same as the one explained in 3.2.1. The three configurations obtained in the previous section are compared against the finite element simulation. All three sets of spring stiffnesses could be used inside the actual model.

The single-sided amplitude spectrum and the Transfer function are used to compare the various models. It is possible to evaluate the resulting displacements at five distinct places. The displacement information relative to the solar panel's tip has been chosen in order to assess the various stiffness options. Each run provides the nodes' and the input force's displacement with respect to time as raw data. A frequency analysis, applied to the normalized tip displacement, was used to get the graphs that are shown below.

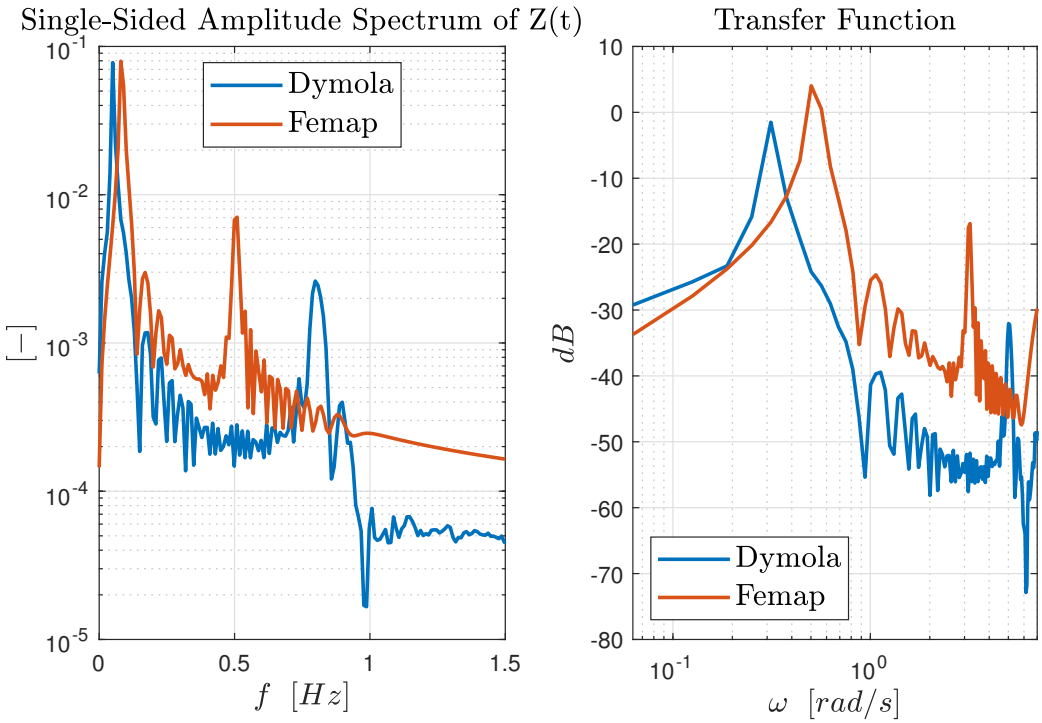


Figure 3.11: Validation results using stiffnesses from 3.5 with $w = 1$

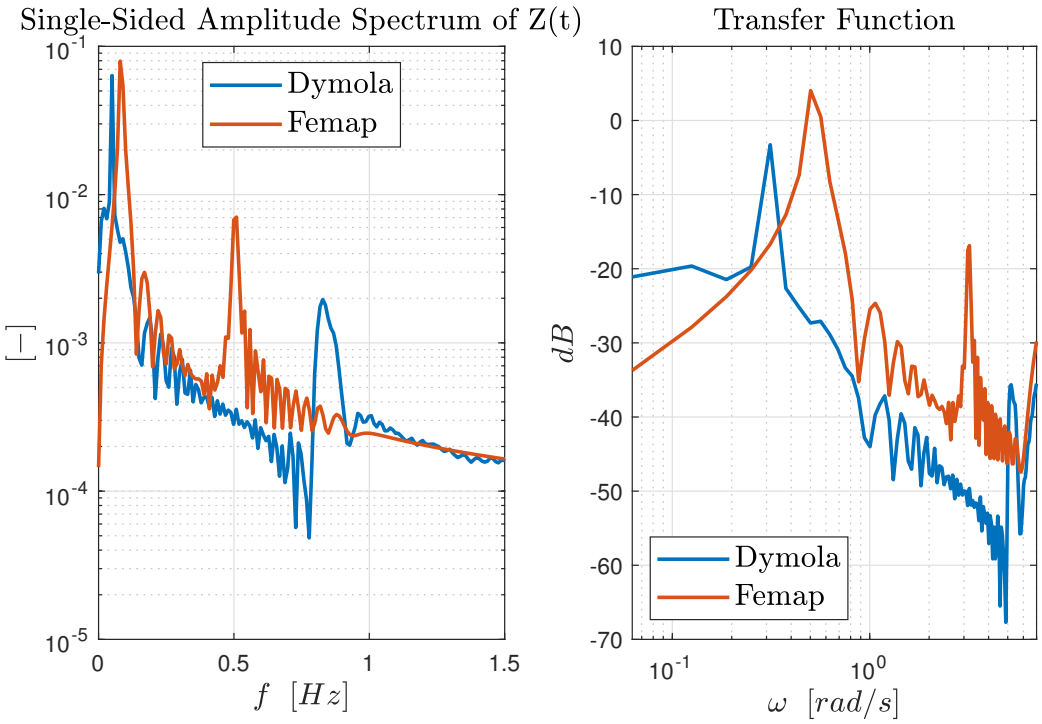


Figure 3.12: Validation results using stiffnesses from 3.6 with $w = 10$

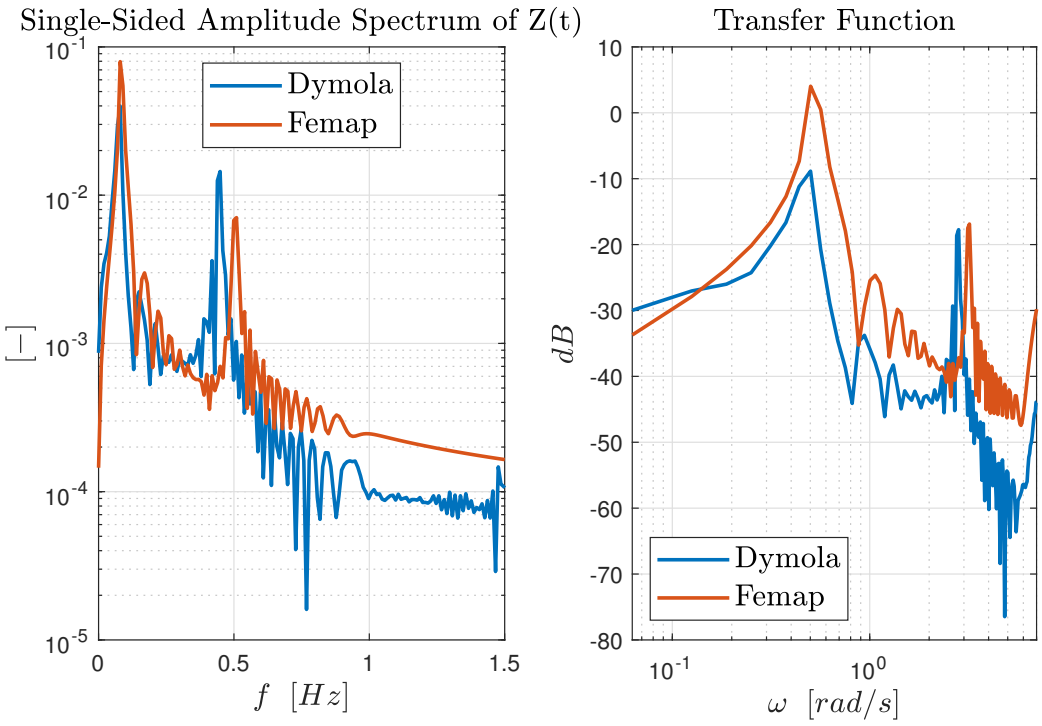


Figure 3.13: Validation results using stiffnesses from 3.7 with $w = 0.1$

As seen from figures 3.11 - 3.13, the choice of the weight has a profound effect on the

results.

The Dymola model has a lower height of the first peak with respect to the Femap model. All three frequency analyses show the presence of two distinct peaks in correspondence of the resonance frequencies of the model. The closest match to these peaks with the ones derived from the FEM data is achieved using $w = 0.1$. In the table below, the peak position for each model is given with their corresponding value and is compared with the original FEM data.

Dymola results		
Weight used in analysis	Peak position [Hz]	Peak height [-]
$w = 1$	0.0499	0.0779
$w = 10$	0.0499	0.0635
$w = 0.1$	0.0798	0.0399
Femap results	0.0799	0.0795

Table 3.9: Resonance frequencies for validation analysis

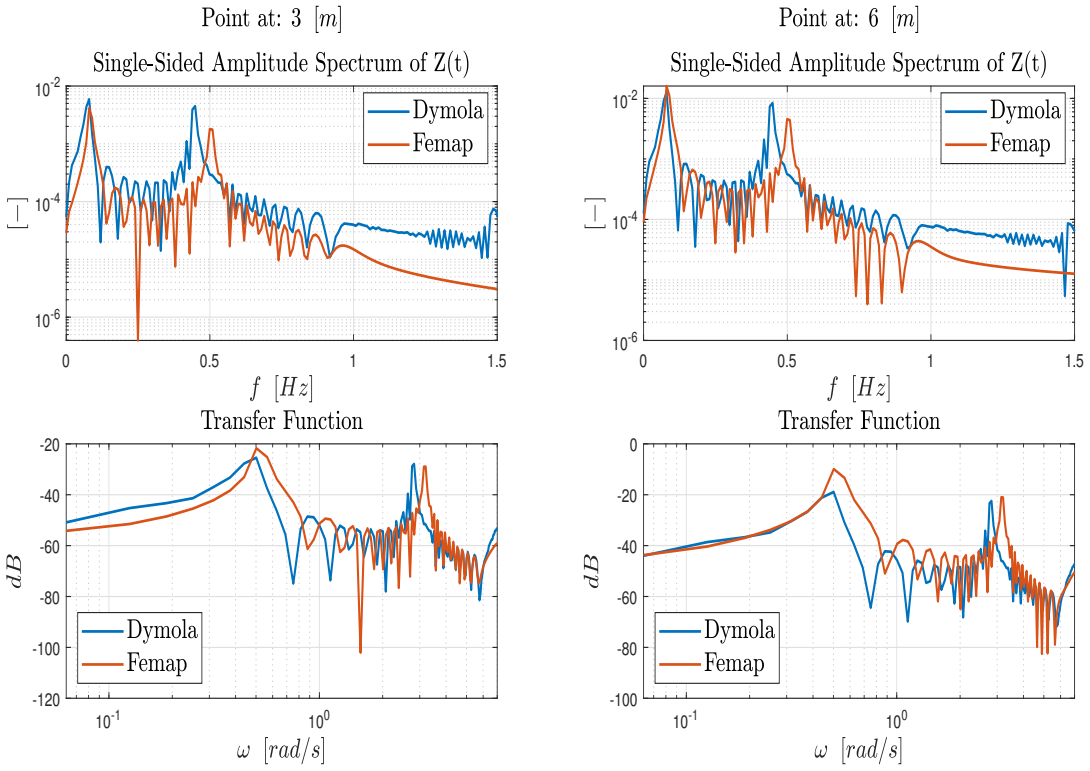
Table 3.9 reports the results derived from the single-sided amplitude spectrum graphs. It is essential to make the following consideration. Even though, the optimizations with weights $w = 10$ and $w = 1$ show that the frequencies of the linearized model are close to the modal frequencies derived from the FEM analysis, the results from the spectral analysis done in Dymola show that such frequencies are not the same. This is due mainly to the non-linear nature of the discretized solar panel dynamics. Since the non-linear dynamical system is optimized on a linearized system, the system's natural frequency will inevitably differ from the frequency obtained by the optimization.

It is possible that the concurrent action of the load type could provide an additional explanation for the results. A model that has exact modal shapes rather than modal frequencies may be preferred if the load is a force at the tip of the solar panel rather than a specified displacement at the base, which may have been an alternative for the validation procedure.

To summarize, the stiffnesses obtained with $w = 0.1$ should be the ones to be used in the model in subsequent chapters of this work.

3.4.1. Validation Results

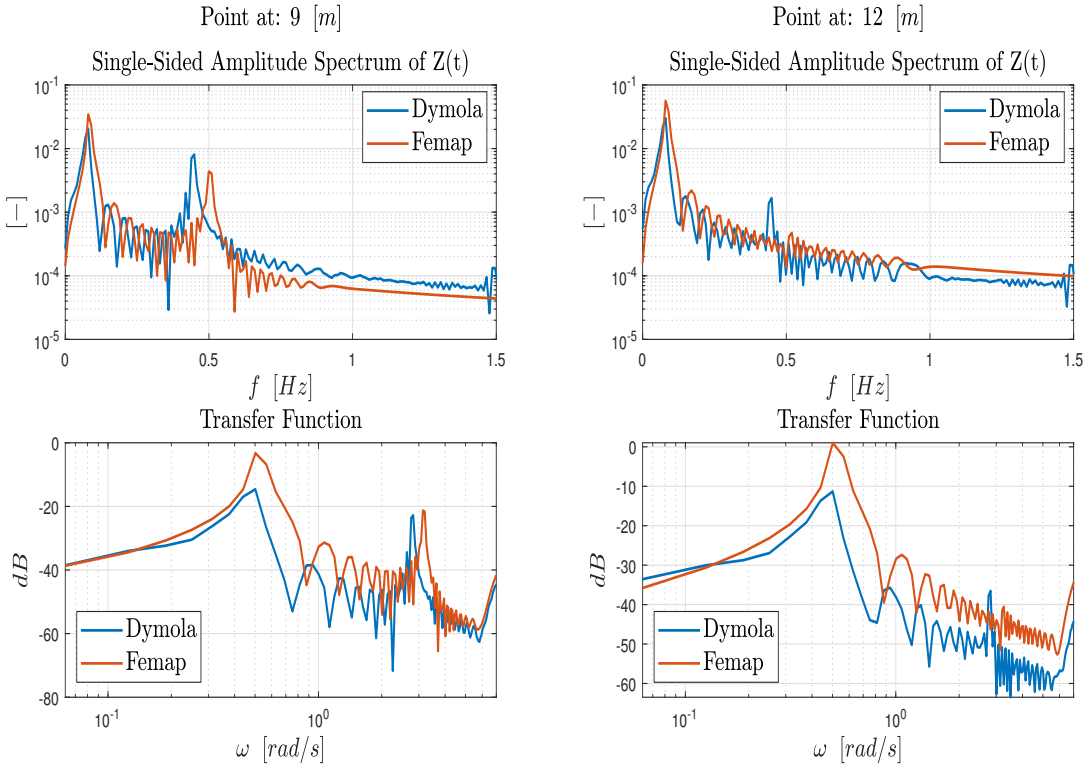
In this section, the validation is presented by referring to the spring element stiffnesses obtained through the optimization process with $w = 0.1$. As presented before, this is the weight selection leading to the best results of the transient load compared with the reference model. Figures 3.14 and 3.15 represent the results relative to the inner point of the solar panel. The inner nodes are equally spaced and are identified by their position along the longitudinal direction of the solar panel. They are found at $x = (3,\ 6,\ 9,\ 12)\ [m]$ respectively.



(a) Inner point 1

(b) Inner point 2.

Figure 3.14: Validation results relative to inner points with stiffness obtained with $w = 0.1$ at position 3 and 6 $[m]$



(a) Inner point 3.

(b) Inner point 4.

Figure 3.15: Validation results relative to inner points with stiffness obtained with $w = 0.1$ at position 9 and 12 $[m]$

Figures 3.14 and 3.15 show the results with upper-frequency limitations of 1.5 $[Hz]$, even though the dataset allows for a frequency analysis up to 4 $[Hz]$. This makes validation in the lower frequency range simpler to be appreciated. As seen, the model behaves similarly to the finite element model for the inner nodes and not just at the solar panel's tip.

3.4.2. Validation of the chaser's solar panel

Regarding the chaser satellite solar array, the same procedure for the validation is followed. Indeed the e.Deorbit studies [18] did not include a detailed description of the solar array; a similar layup structure is used. The thickness of the aluminium honeycomb element was reduced to $th = 2$ $[mm]$. In such a way, the chaser's solar array has modal frequencies in the same order of magnitude as the Envisat ones. The last difference with the validation procedure above is using a sine sweep with an amplitude of $a = 0.01$ $[N]$ rather than unitary.

In this case, the results relative to the best run obtained with the weight set at $w = 0.1$

are reported. The derived spring stiffnesses are reported in table 3.10.

Chaser SP	k_1	k_2	k_3	k_4	k_5	k_{tor}
value $[Nm/rad]$	53.6	99.7	63.1	36.4	15.2	9.93

Table 3.10: Chaser's solar panel paremeters

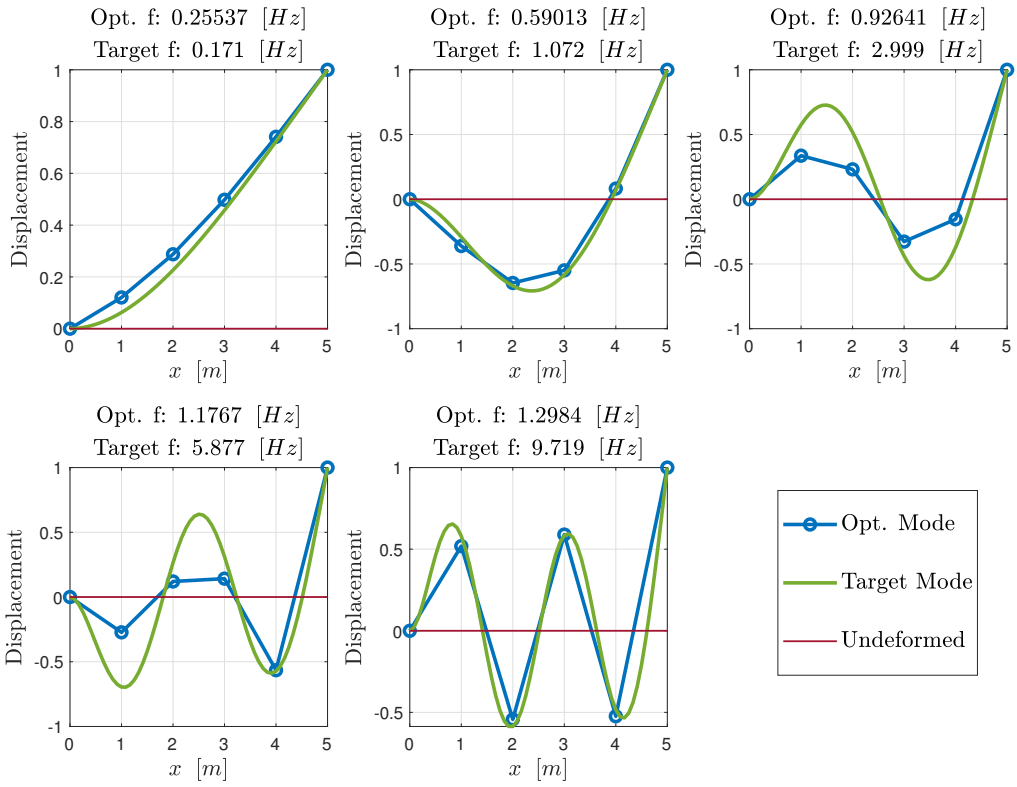


Figure 3.16: Modal shapes and Modal frequencies of the chaser's solar panel $w = 0.1$

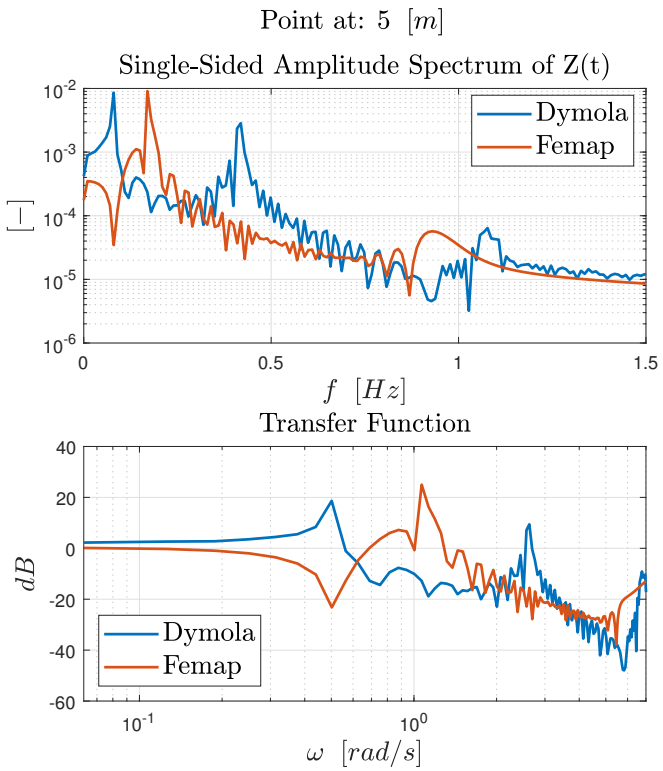


Figure 3.17: Validation results for the chaser's solar panel with $w = 0.1$

Figures 3.16 and 3.17 report the results of the modal analysis and of the transient response analysis of the solar panel. Specifically, the tip displacement and the analysis of the modal frequencies and shapes are presented. These results are worse than the previous ones, even though the same approach was used. This is primarily due to the non-linear behaviour of the system, nonetheless they can be considered acceptable for the scope of this work.

4 | Filter implementation

In this chapter, the filter implementation inside the FMU code will be explained. With the term FMU code, it is intended the Modelica code that Dymola presents to the user after the importing of the FMU takes place. As described in more detail in section 2.2.3, such code gives the user the possibility to write algorithms directly inside the FMU.

4.1. FMU filter implementation

Implementing the two filters described in the algorithms 2.4 and 2.3 follows the same procedure. First, it is necessary to get the identifiers of the state variables of the system. These identifiers are reported inside the model `ModelDescription.xml`. This file structure is described in section 2.2.3. Figure 4.1 shows how the state variable are reported in the section `ModelVariables` of the provided model description file.

```
<!-- Index for next variable = 13794 -->
<ScalarVariable
  name="stateSelect.set1.x[1]"
  valueReference="33554479">
  <Real/>
</ScalarVariable>
<!-- Index for next variable = 13795 -->
<ScalarVariable
  name="stateSelect.set1.x[2]"
  valueReference="33554480">
  <Real/>
</ScalarVariable>
<!-- Index for next variable = 13796 -->
<ScalarVariable
  name="stateSelect.set1.x[3]"
  valueReference="33554481">
  <Real/>
</ScalarVariable>
```

Figure 4.1: State variable described inside ModelDescription.xml

These few lines give some vital information for the attitude filter implementation. As already mentioned above, the identifiers are reported, and they can be found mentioned as `valueRefernce`. The variable's reference index is then provided. It is highlighted in green, and it is helpful to look for more details about the variable in the model description

file. Using this as an example, one may determine whether there is a correlation between two quantities. Note that despite representing a quaternion, the state inside the FMU consists of 3 elements. Only a minimal representation of the state is required for the FMU to function, and the orientation is represented by three scalar elements. Only three attitude quaternion elements are chosen by the Dymola exported FMU; the last element is retrieved by employing the group's unitary constraints feature.

However, there is an issue with the filter's application. In order to prevent singularities during the integration process, three of the quaternion elements are dynamically chosen as states inside the FMU. The user is aware of the names of the three state components but is unaware of which attitude quaternion element refers to which state. Inside the FMU, a straightforward algorithm has been implemented that checks for changes in the state's value and, if any are found, calls a function that compares the estimated quaternion to the state variable. At the same time, each component of the measured quaternion, coming from a star tracker, is sorted to match the new state selection.

Now it is possible to go through the filter implementation procedure. The predictor step of the Kalman filter is added inside section 7 in the FMU code, see 2.1. The use of the FMU allows skipping the computation of the predicted state; its values will be extracted later after the `DoStep` section of the FMU.

For the EKF implementation the computation of the linearized Jacobian A is required; see equation 2.31. It is derived as follows [13]:

Algorithm 4.1 State Jacobian Derivation

```
for i = 1 : N_x do
  dz := fmiF.fmiGetDirectionalDerivative(fmi,{Id_der},{Id_state});
  dF[:,i] := dz;
end for
A = exp(dF*stepSize);
```

In algorithm 4.1 the `fmiGetDirectionalDerivative` function takes the directional derivative of the integer labels corresponding to the derivative of the state `Id_der` with respect to the state labels `Id_state`. Both identifiers are derived, as described above, from the model description file. For both EKF and MEKF algorithms, the computation of the linearized observation operator is required. Its computation is described in algorithm 4.2.

Algorithm 4.2 Measurements Jacobian Derivation

for $i = 1 : N_x$ **do**

```
dh := fmiF.fmiGetDirectionalDerivative(fmi,{Id_in},{Id_state});
H[:,i] := dz;
```

end for

In algorithm 4.2 the term `Id_in` indicates the labels of the measured quantities.

Then, before the instruction `DoStep` is given to the FMU, the state correction computed in the previous time step is applied to the FMU as follows:

```
fmiF.fmiSetReal(fmi,{Id_state},XCorr);
```
(4.1)

The FMU integrates the system to the next time step. The use of FMUs allows for the direct derivation of the expected measurements without using any Jacobian as shown in equation (4.2). In the same equation, the predicted state is extracted.

```
yPred:=fmiF.fmiGetReal(fmi,{Id_in});
xPred:=fmiF.fmiGetReal(fmi,{Id_state});
```
(4.2)

Finally, in the second algorithm section of 2.1 it is possible to implement the predictor section in the corresponding attitude filter.

The two FMUs can now be both defined separately. The observed quaternion element provided by a star tracker and the measurements about the angular velocities produced from a gyroscope are needed as inputs for the EKF implementation. In the EKF algorithm, the angular velocity is estimated individually. According to equation (4.3), a new state is implemented for each angular velocity component. This has the benefit of using fewer matrices during computation.

$$\Omega = \begin{Bmatrix} \omega \\ \delta\omega \end{Bmatrix} \tag{4.3}$$

The state Jacobian used is equal to:

$$\mathbf{F}_\omega = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \tag{4.4}$$

Then, the FMU that implements the EKF is reported in figure 4.2. On the right-hand side of the FMU block the needed inputs are present. As explained before, the FMU is generated directly from Dymola and then modified to include the EKF algorithm. Thus, it requires the same input torque T_{in} as the chaser system model 3.2. The filter needs the observed quaternion and angular velocity vector to assess the system's attitude appropriately. The estimated inertial properties are the final quantities used before the integration. These are the output of the parameter estimation FMU. Inside both EKF and MEKF implementation, the string of code (4.1) is used to change the mass and the inertia matrix with the most up-to-date values.

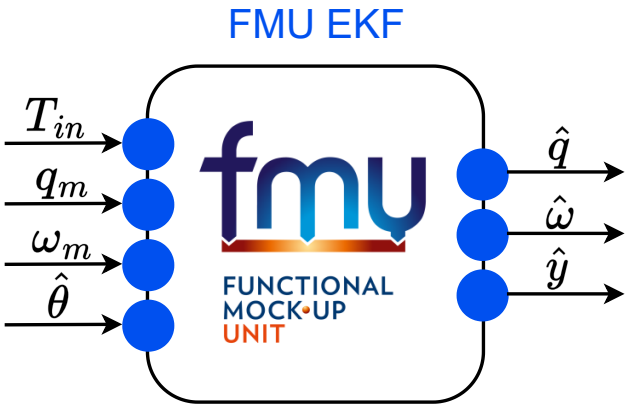


Figure 4.2: EKF implemented inside FMU

The outputs of the FMU are shown on the left-hand side of the block; see figure 4.2. The quantities $\hat{q}$ and $\hat{\omega}$ are, respectively, the estimated quaternion and angular velocity vectors. The parameter estimation technique uses the third and final output, known as $\hat{y}$, as an input. Such quantity will be explained in the next section.

Regarding the MEKF implementation, as explained in section 2.4.2, the angular velocity is estimated directly with the error quaternion. The output vectors for the FMU block will be the same and, if necessary, all system variables can be accessed in the `Results.mat` file that is created once each simulation is started.

4.2. Parameter estimation FMU

The parameter estimation is implemented inside the FMU block shown in figure 4.3. The main differences with respect to the previous functional mock-up units shown above are the inputs and outputs needed. As explained in section 2.5, the parameter estimation process utilizes the estimated quantities from the state estimation algorithm. The vari-

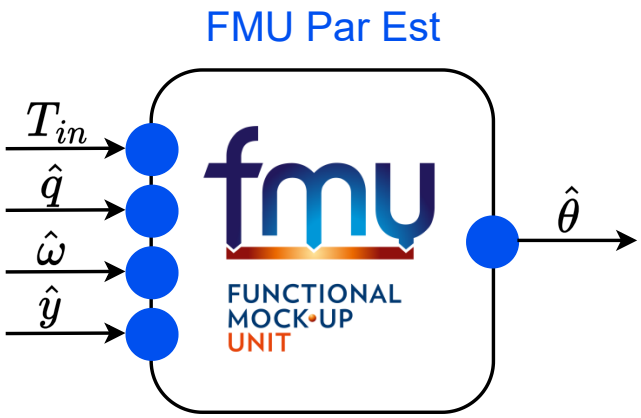


Figure 4.3: EKF for parameter estimation implemented inside FMU

ables required for the mass and inertia estimation operations can be determined from the `ModelDescription.xml` file. It contains the information on which quantities depend on the inertial properties. They are the angular accelerations of the main chaser body and the joint variable acceleration of the 7 DOF manipulator. Then, using the labels related to the accelerations mentioned above, the Jacobian of the measured quantities with respect to the inertial parameter is determined as explained in algorithm 4.2. The filter requires the use of the Sigmoid mapping, see section 2.5, so it is not possible to apply the filter state directly to the FMU since it does not directly represent the inertial properties. This is another difference between implementing the parameter estimation process in the FMU and the procedure shown above for the EKF. As a result, the state is transformed into the inertial parameter after the predictor step so that it can be applied to the FMU during the subsequent iteration.

The filter was initially tested independently of the other FMU; as a result, the acceleration data straight from the entire system model were used as inputs. This testing brought to light that the related covariance was growing over time as the filter estimated the value. This behaviour was the antithesis of what was desired. It was determined that the $\mathbf{H}_k^x$ term drove these observations; for more information, see algorithm 2.6. Equation 7 in 2.6 may be approximated as follows since the directional derivative, in particular, has an extremely low value, in the order of $1e-13$.

$$\mathbf{P}_k^\theta = \left(\mathbf{I} - \mathbf{L}_k^\theta \mathbf{H}_k^x\right) \mathbf{P}_{k|k-1}^\theta \simeq \mathbf{P}_{k|k-1}^\theta \rightarrow \mathbf{P}_k^\theta \simeq \mathbf{P}_{k-1}^\theta + \mathbf{Q}_k^\theta \tag{4.5}$$

Since $\mathbf{Q}_k^\theta$ represents the variance of the fictitious noise used for the dynamic modelling of the parameter, which is strictly positive, the covariance matrix term tends to grow con-

tinuously. An additional tuning parameter is used to improve the algorithm's derivation of the covariance matrix. The tuning parameter is used in equation 7 of algorithm 2.6, and the modified equation becomes:

$$\mathbf{P}_k^\theta = \left(\mathbf{I} - u\mathbf{L}_k^\theta \mathbf{H}_k^x\right) \mathbf{P}_{k|k-1}^\theta \tag{4.6}$$

The value assigned to the term u will be given in the following chapter within the initialization data for the simulation process. One last consideration on the parameter estimation algorithm can be done on the fictitious noise r_k and its auto-covariance. The noise itself changes the parameter value at each iteration process. Generally, it is possible to state that as the noise increases, the filter will modify the parameter value quickly. As the noise decreases, the parameter value will be modified gradually. Using such information, the noise with its auto-covariance will assume a different value as the time of the simulation progresses. Its value will be large at the start of the simulation since the aim is to obtain a fast convergence of the parameter we want to estimate. Then as simulation time increases, the noise will decrease in value up to a predetermined lower bound. In this way, when the filter is near the true value of the parameter, its estimate will be more stable.

4.3. Overall system structure

Finally, once all aspects have been implemented within the Dymola environment, they may be combined into the final model, which is utilized to obtain the results displayed in the following chapter.

Figure 4.4 depicts how each block is connected to one another, as the two FMUs responsible for state and parameter estimation are coupled according to Scheme 2.3. In addition to the FMUs and the system model, the sensors used for attitude determination are implemented. The star tracker is set up as follows:

Algorithm 4.3 Star Tracker modelling

Input: true quaternion q_{true}

Output: measured quaternion q_m

1: Convert quaternion into Euler's angles
2: Apply to each of the three Euler's angles a white noise
3: Convert Euler's angles to quaternion q_m

This approach preserves the unitary constraints of the true quaternion.

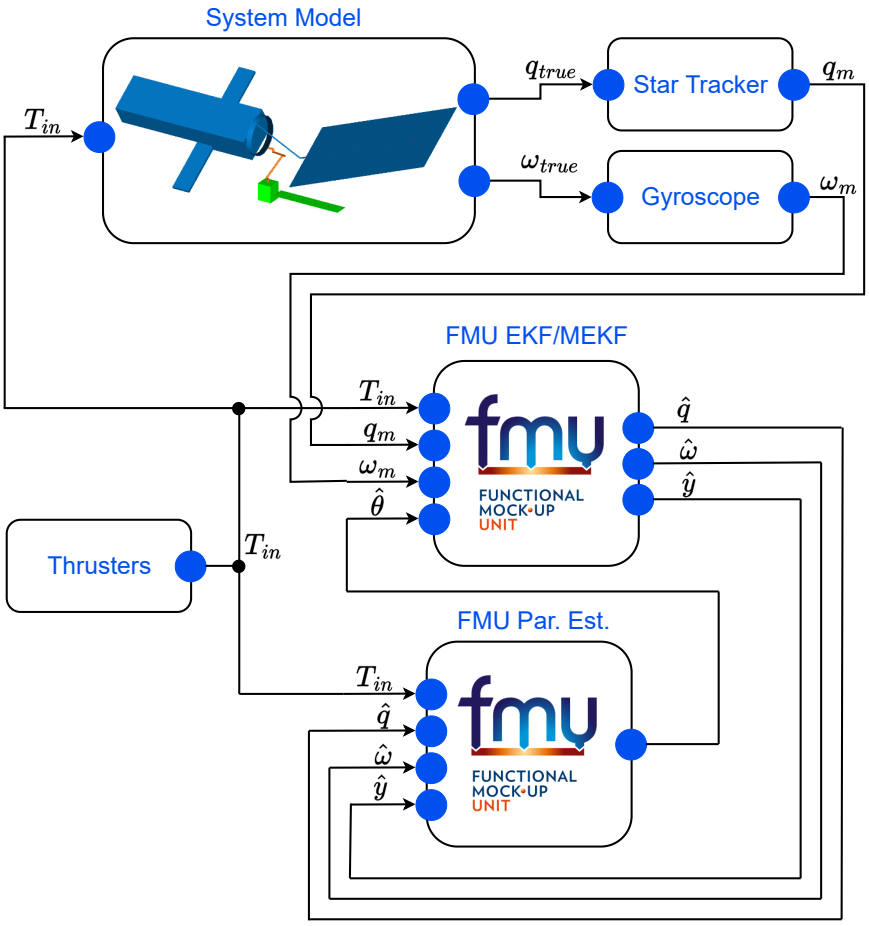


Figure 4.4: Overall system with filters scheme

In terms of angular velocity, the gyroscope model has been realized using equation (2.33). The thruster block lacks a model, but it is placed to show the existence of a specified input torque. The prescribed torque in the simulation has the shape shown in picture 4.5. Its goal is to simulate the impulsive behaviour of an attitude control thruster. Also, since the simulation aims to check if the attitude determination works and if the estimator can estimate at least part of the system's inertial properties, the integral of the torque in time is equal to zero. In this manner, the system is probed without interfering with other functions.

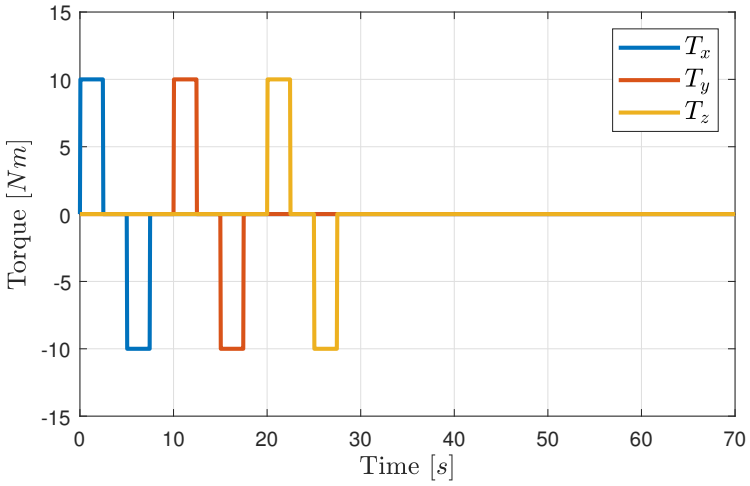


Figure 4.5: Input torque profile applied to chaser body

5 | Performance Analysis

In this chapter, the results from the simulations are presented and analyzed.

All the simulations are initialized with the same system parameters. As the initial state, it is assumed that the system of the two combined satellites has zero absolute angular velocity. The starting orientation value is unimportant to the goal of this work. As a result, the following orientation quaternion is assigned to the reference system:

$$q_i = \begin{bmatrix} 0.233 \\ 0.332 \\ 0.084 \\ 0.910 \end{bmatrix} \tag{5.1}$$

As previously stated, the input torque was modelled to simulate the behaviour of several impulsive attitude thrusters. The chaser's and Envisat's inertia and mass attributes are identical to those described in Section 3.2.

The noises associated with each sensor are the final elements that must be initialized. In terms of the star tracker, the noise added to each Euler angle prior to conversion is as follows:

$$\sigma_i = 0.017[\text{rad}] \tag{5.2}$$

The noise levels are acquired from Hyperion Technologies' star tracker ST400 [4]. The resulting star tracker model is significantly noisier than commercial products for the class of satellite used as a chaser; this is done to demonstrate the FMU's ability to estimate states from high noise measurements, making this application possible even for less expensive hardware or smallsat missions.

Similarly, the noise values for the gyroscope are derived from an actual gyroscope, the KVH DSP-4000 [4]. Its parameters are reported in table 5.1.

Gyroscope	Bias instability $[deg/hr]$	ARW $[deg/\sqrt{hr}]$
KVH DSP-4000	1	0.067

Table 5.1: Noises for gyroscope model

5.1. EKF results analysis

The following structure is used to analyze the results: the EKF filter is tested without the parameter prediction model first, assuming that the inertial parameters are known ahead of time, and then the EKF algorithm is tested with the parameter estimating procedure. Two simulations are run for each analysis, one with the flexible model parameter derived in section 3.3.1 and one with the spring stiffnesses set to $k = 1e+7$ $[Nm/rad]$ to simulate a model with the solar panels and manipulator assumed to be rigid. The state of the FMUs is set to:

$$q_i = \begin{bmatrix} 0.265 \\ 0.329 \\ 0.090 \\ 0.902 \end{bmatrix} \tag{5.3}$$

The tuning parameters used in the EKF FMU, during the analysis, are reported in equation (5.4) for both quaternion and angular velocity estimation.

$$\begin{array}{lll} \mathbf{R}_{3\times3} = 0.01^2\mathbf{I}_{3\times3} & Q_{3\times3} = 0.0007^2\mathbf{I}_{3\times3} & P_{0,3\times3} = 0.02^2\mathbf{I}_{3\times3} \\ \mathbf{R}_{\omega,2\times2} = 0.1^2\mathbf{I}_{2\times2} & Q_{\omega,2\times2} = 0.1^2\mathbf{I}_{2\times2} & P^{\omega}_{0,2\times2} = 0.1\mathbf{I}_{2\times2} \end{array} \tag{5.4}$$

The mass of Envisat is the primary inertial parameter that is estimated.

The first result refers to the simulation without the parameter estimation component. The quaternion estimate from the EKF filter with the true state of the system and with the noisy observations are reported in Figure 5.2.

The derivation of the fourth quaternion auto-covariance value is particularly noteworthy. The filters produce the covariance matrix related to the state, which contains three elements, as described in 4.1. To represent the 3σ confidence levels, the diagonal elements of the covariance matrix are used as $\sigma_i = \sqrt{P_{ii}}$. The fourth quaternion element can be computed simply from the other values using the unitary constraints. In contrast, its 3σ value is derived from the assumption that the uncertainties associated with the three estimated

quaternion elements are unrelated. This assumption can be confirmed by inspecting the covariance matrix extra-diagonal elements, see figure 5.1 .

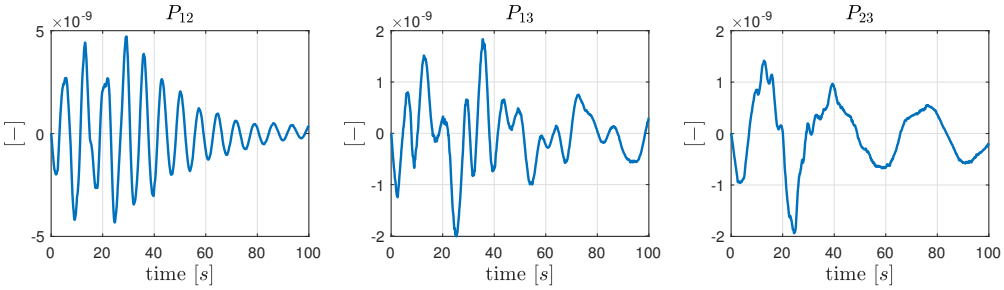


Figure 5.1: Outer-diagonal elements of Covariance matrix for EKF

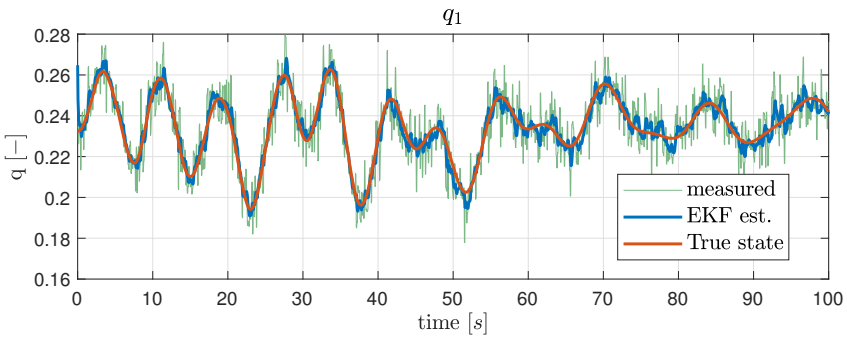
As can be seen, the outer-diagonal values are more than orders of magnitude lower than the diagonal ones that assume values above 10^{-5}. As a result, the auto-covariance of the fourth quaternion component can be calculated as follows:

$$P_4 = \left(\frac{\partial f(q)}{\partial q_1}\right)^2 P_1 + \left(\frac{\partial f(q)}{\partial q_2}\right)^2 P_2 + \left(\frac{\partial f(q)}{\partial q_3}\right)^2 P_3$$
$$\text{where} \tag{5.5}$$
$$f(q) = \sqrt{1 - (q_1^2 + q_2^2 + q_3^2)}$$

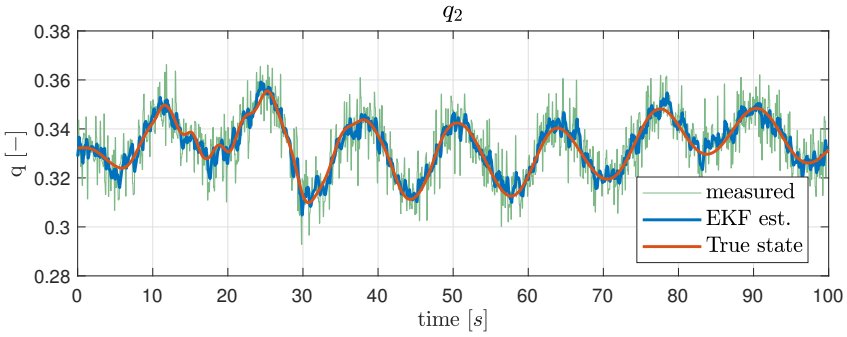
As illustrated in Figure 5.2, the EKF filter is capable of correctly calculating the physical system's orientation. Figure 5.3 depicts the quaternion error together with the 3σ confidence ranges produced from the FMU estimated covariance matrix. The EKF filter's covariance converges during the first few seconds of simulation, and it can be noted that the absolute error of each quaternion is constantly less than the 3σ value. The mean absolute integrated error is determined to provide a quantitative measure associated with the error in estimating the quaternion state; the results are displayed in Table 5.2.

e_1	e_2	e_3	e_4
$2.76e-03$	$2.48e-03$	$2.968e-03$	$9.36e-04$

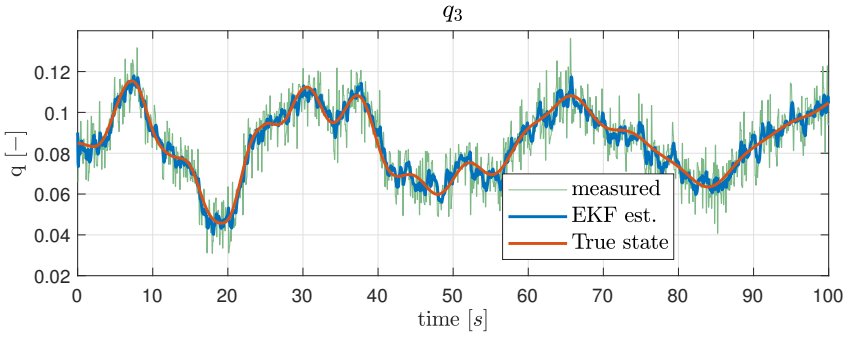
Table 5.2: Mean integrated absolute error of EKF



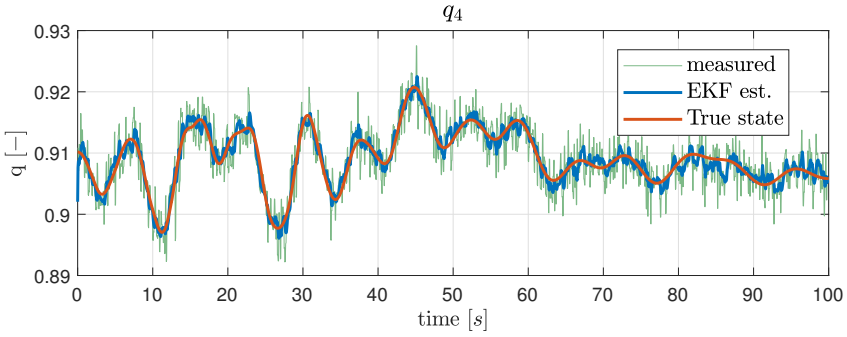
(a) Quaternion element 1



(b) Quaternion element 2

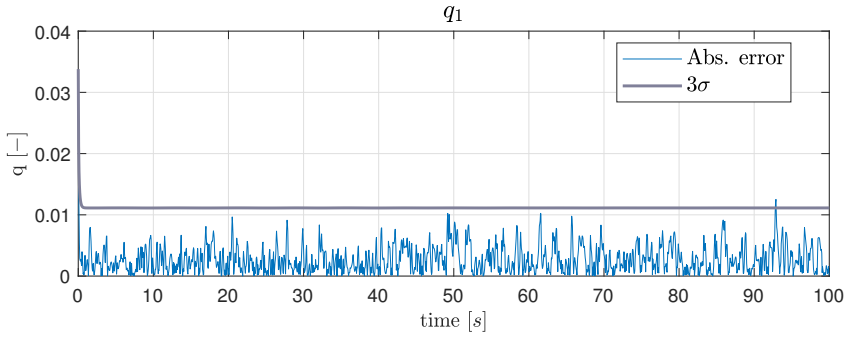


(c) Quaternion element 3

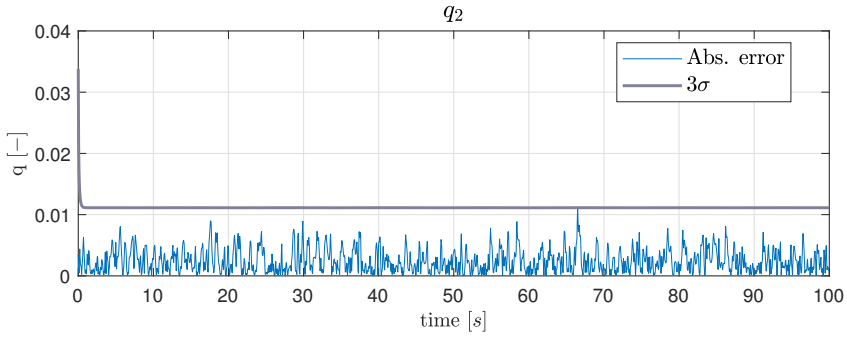


(d) Quaternion element 4

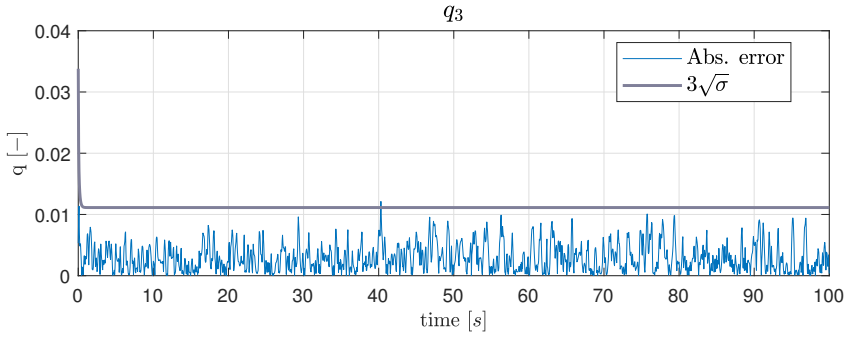
Figure 5.2: Attitude estimate obtained from EKF without parameter estimation.



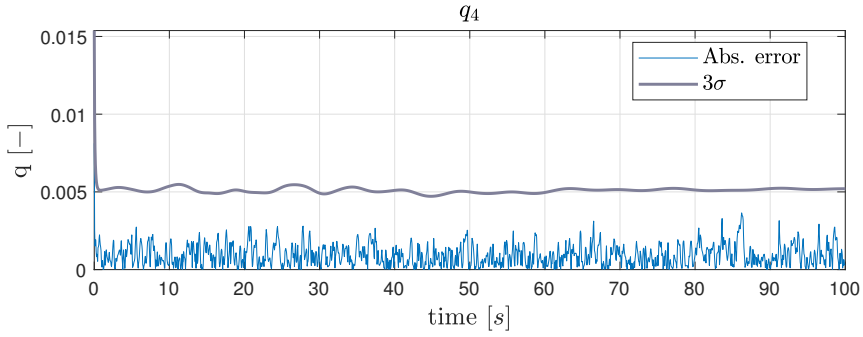
(a) Quaternion element 1



(b) Quaternion element 2



(c) Quaternion element 3



(d) Quaternion element 4

Figure 5.3: Attitude estimate error with 3σ confidence range.

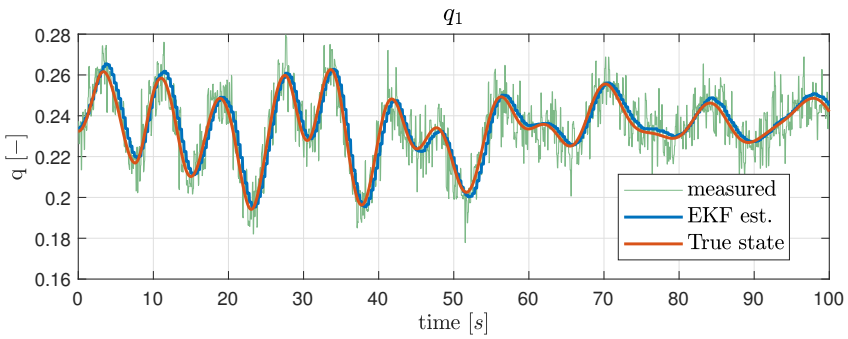
The simulation results with the FMU representing a rigid system are then illustrated in Figures 5.4 and 5.5. The two figures indicate the estimated state and the estimation process error, respectively. The mean integrated absolute error is computed once more to provide a measure of the error, and the results are displayed in Table 5.3. It is clear that, even if the FMU does not accurately represent the underlying system, it can still achieve a good approximation of the state.

e_1	e_2	e_3	e_4
$3.37e-03$	$1.71e-03$	$2.14e-03$	$7.91e-04$

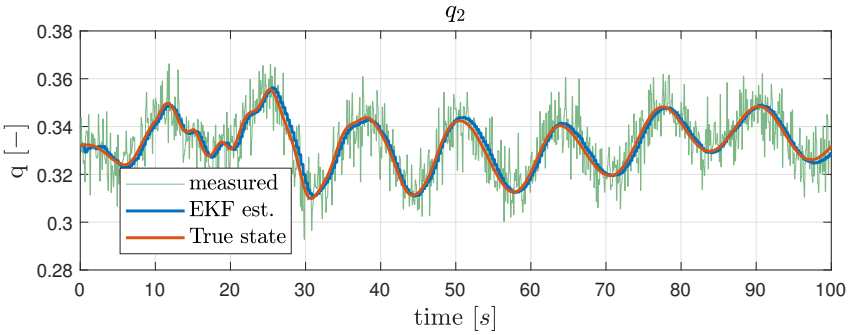
Table 5.3: Mean integrated absolute error of EKF with rigid assumption

The outcomes of the two simulations can now be compared. The errors relative to the rigid simulation are lower than the ones obtained from the simulation with the flexible model. Such analysis may be deemed insufficient to truly evaluate the two filters. In fact, as shown in Figure 5.5, the error of the rigid simulation exhibits a distinct behaviour that differs from white noise. This assertion is supported by a whiteness test performed on the error. Various approaches exist for performing such tests; the Anderson test is employed in this work. It consists of computing the sample covariance from the error data with a varying number of samples τ and counting the number of covariances that exceed the confidence level $\alpha = 0.05$ referenced to a Gaussian distribution. The flexible assumption filter passes the whiteness test, while the other does not. As a result, while the second filter implementation has a smaller mean absolute error for some components of q, it is incapable of accounting for the absence of flexible parts. This is also evident in Figures 5.6 and 5.7, which depict the sample covariance of the simulations run using the flexible and rigid assumptions, respectively. The covariance is calculated using Equation (5.6), where N is the number of samples in the signal. The information is acquired from the error concerning the first quaternion components. As can be seen, the 5.6 resembles a white noise, that has the same variance as the error, better than the rigid simulation results.

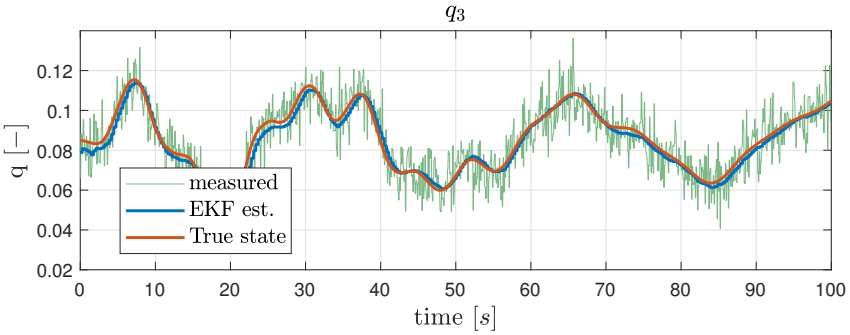
$$Cov(\tau) = \frac{1}{N} \sum_{i=1}^{N-\tau} e(i)e(i-\tau) \tag{5.6}$$



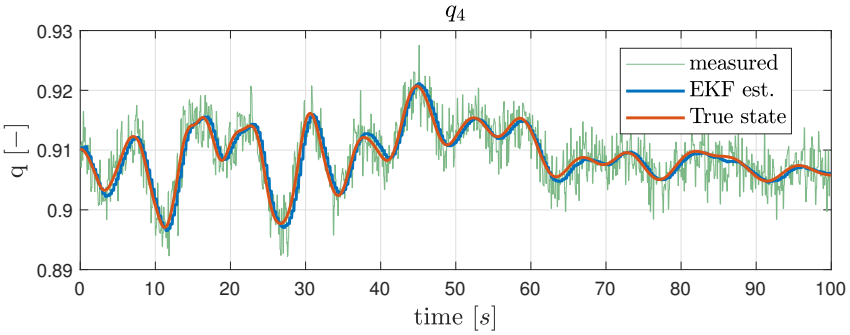
(a) Quaternion element 1



(b) Quaternion element 2

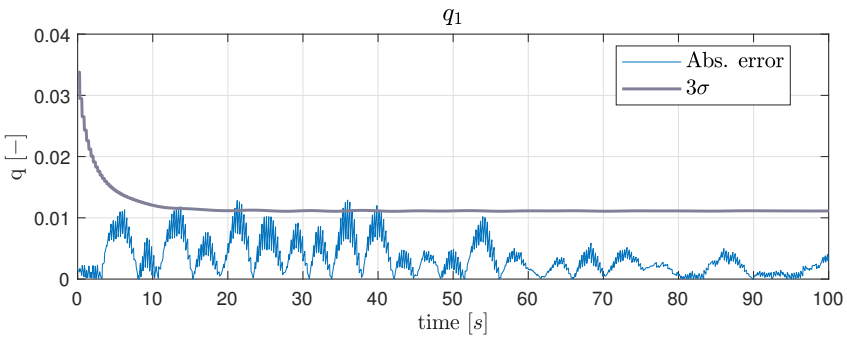


(c) Quaternion element 3

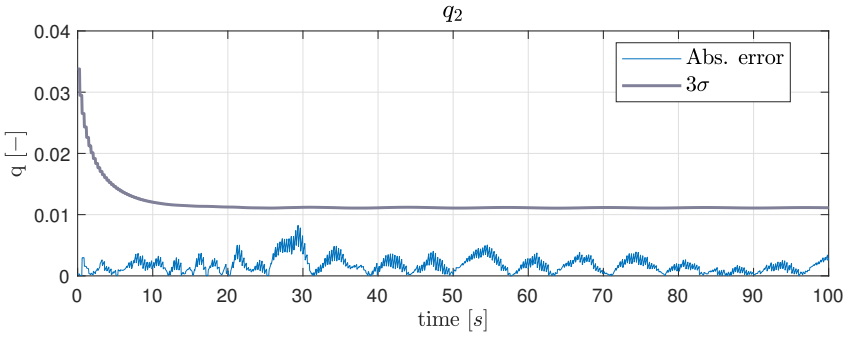


(d) Quaternion element 4

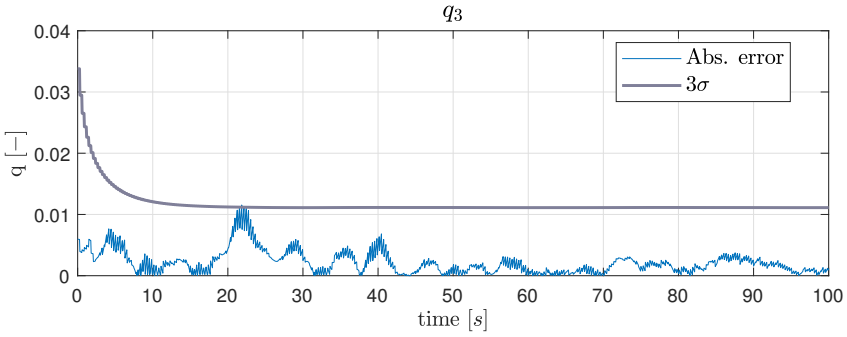
Figure 5.4: Attitude estimate obtained from EKF with rigid solar panels and manipulator arm.



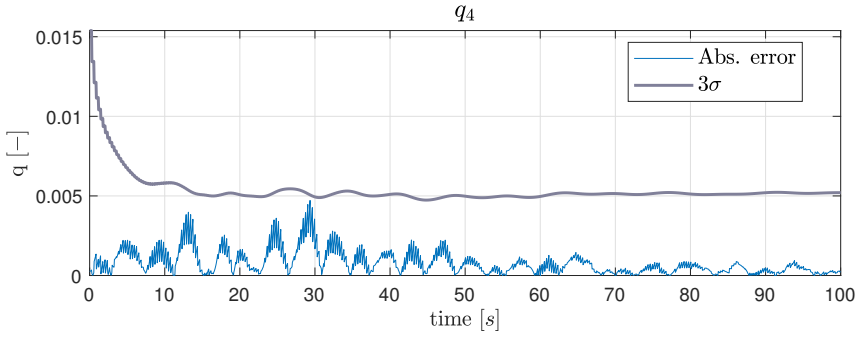
(a) Quaternion element 1



(b) Quaternion element 2



(c) Quaternion element 3



(d) Quaternion element 4

Figure 5.5: EKF attitude estimate error with 3σ confidence range with rigid solar panels and manipulator arm.

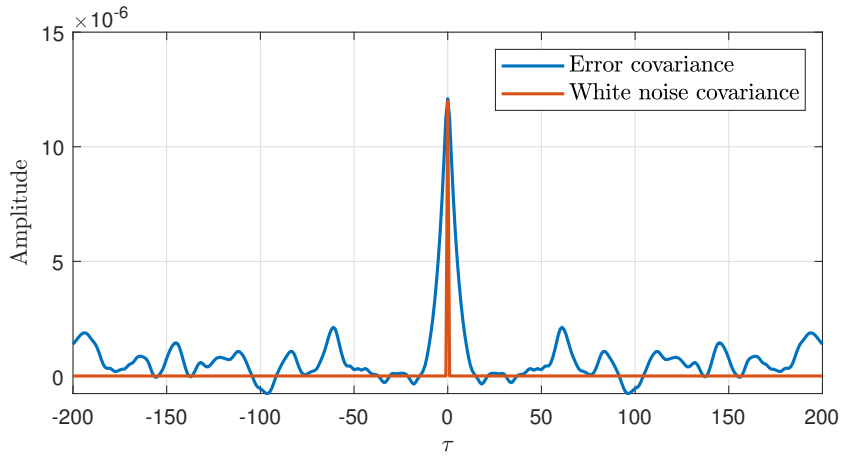


Figure 5.6: Covariance Analysis EKF

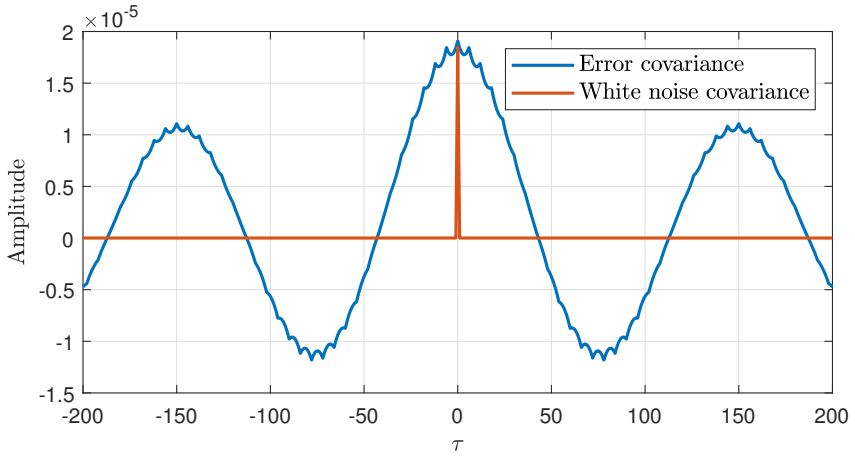


Figure 5.7: Covariance Analysis EKF rigid

5.1.1. EKF and parameter estimation

The parameter estimation results conducted with the EKF are provided and explored further below.

The parameter estimation FMU is set up with:

$$c = 0.001 \qquad a = 9000\ [kg] \qquad b = 5000\ [kg]$$
$$R_\theta = \begin{bmatrix} 10^{-4}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times7} \\ \mathbf{0}_{7\times3} & 10^{-2}\mathbf{I}_{7\times7} \end{bmatrix} [kg^2] \quad Q_\theta = 0.1 + \tfrac{20}{t+1}\ [kg^2] \qquad u = 10^{14} \tag{5.7}$$

These parameters are discussed in sections 2.5 and 4.2. The value t represents the current simulation time in equation (5.7). In contrast, the term R_θ represents the covariance of the noise added to the inertial parameter inside the filter. As discussed in section 4.2, the

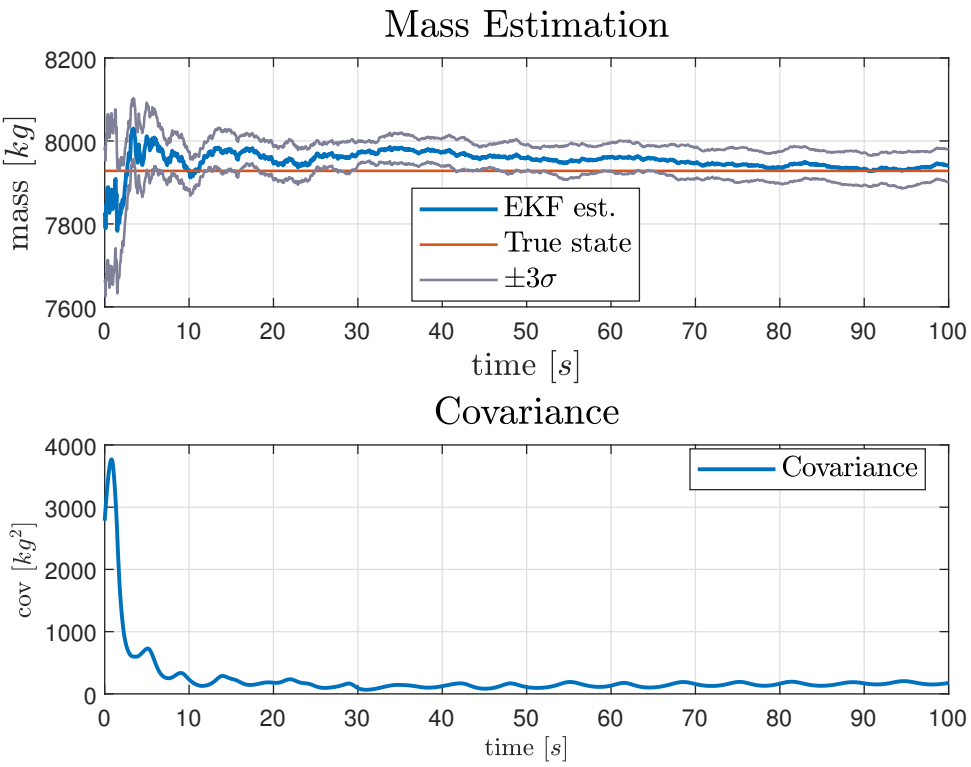


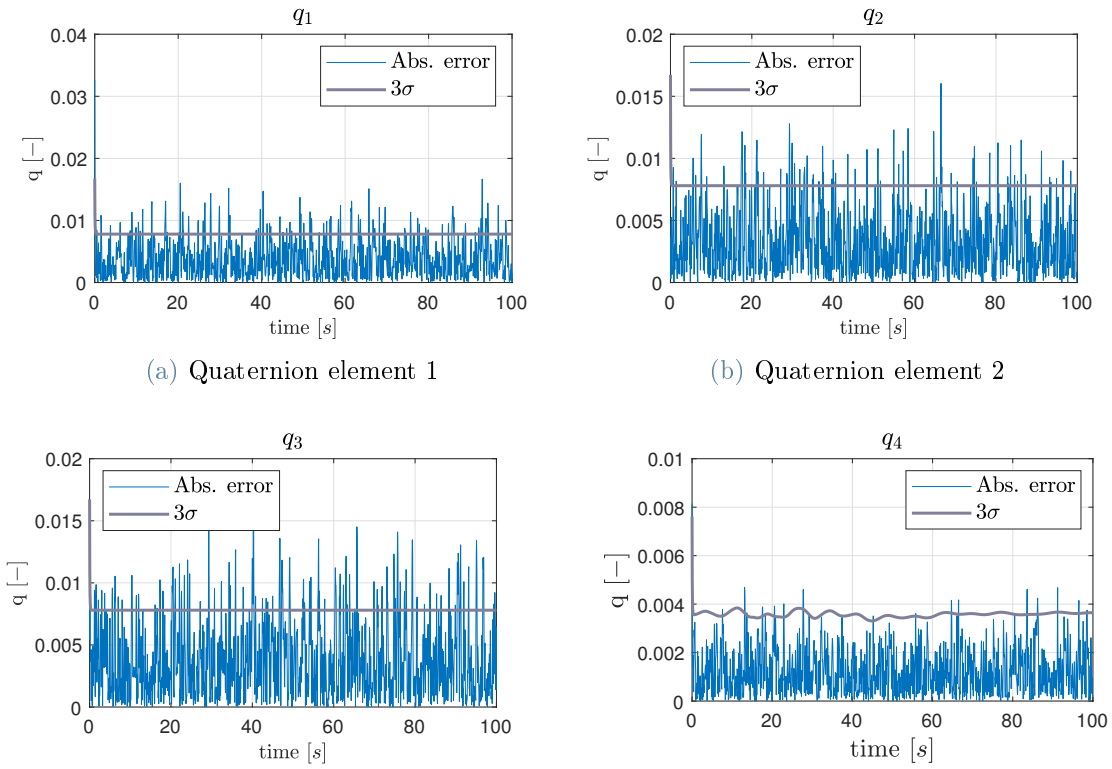
Figure 5.8: Mass estimate performed by EKF with flexible bodies assumption.

input noise to the FMU used for parameter estimation varies over time. In particular, for all simulations performed in this work, the noise covariance is fixed to 20.1 at time t_0 and gradually drops to $R_\theta = 0.1$.

Both simulations for flexible and stiff FMUs are done with an inaccurate Envisat satellite mass value. The initial guess for the mass is 100 $[kg]$ lower than the true value.

The findings for the EKF with flexible bodies assumption are shown in Figure 5.8. The figure demonstrates that the filter can estimate the mass. The estimation quickly approaches the true value, and after $t = 65$ $[s]$ it is within the 3σ range. At the end of the simulation, the estimated mass is equal to $m_{est} = 7940.9$ $[kg]$ with $3\sigma = 39.70$ $[kg]$.

In Figure 5.9, the absolute error in the attitude estimate procedure may be seen. When compared to the findings obtained without parameter estimation, the mass estimation procedure produces a greater error. This is to be expected because the filter needs to work with an inaccurate number for Envisat's mass at first. It is possible to notice that while the mass error decreases over time, the attitude error remains essentially constant. This is due to the DEKF implementation, which excludes any correlation between mass fluctuation and orientation, which, as can be seen, leads to larger errors.



(a) Quaternion element 1

(b) Quaternion element 2

(c) Quaternion element 3

(d) Quaternion element 4

Figure 5.9: Quaternion error EKF with mass estimation and flexible assumption.

The final step for the EKF analysis is to determine if the rigid assumption filter can still estimate the mass of Envisat. The parameter estimation filters utilize only the acceleration data relative to the chaser body since the manipulator's joints are considered rigid. The parameter estimation results are shown in figure 5.10. As it is possible to see, the filter cannot estimate the mass, and the outputs of the filters hover around the initial incorrect value. Without an accurate measure of the Envisat mass, this filter cannot estimate the attitude of the chaser satellite with the tuning parameters given above, as shown in figure 5.11.

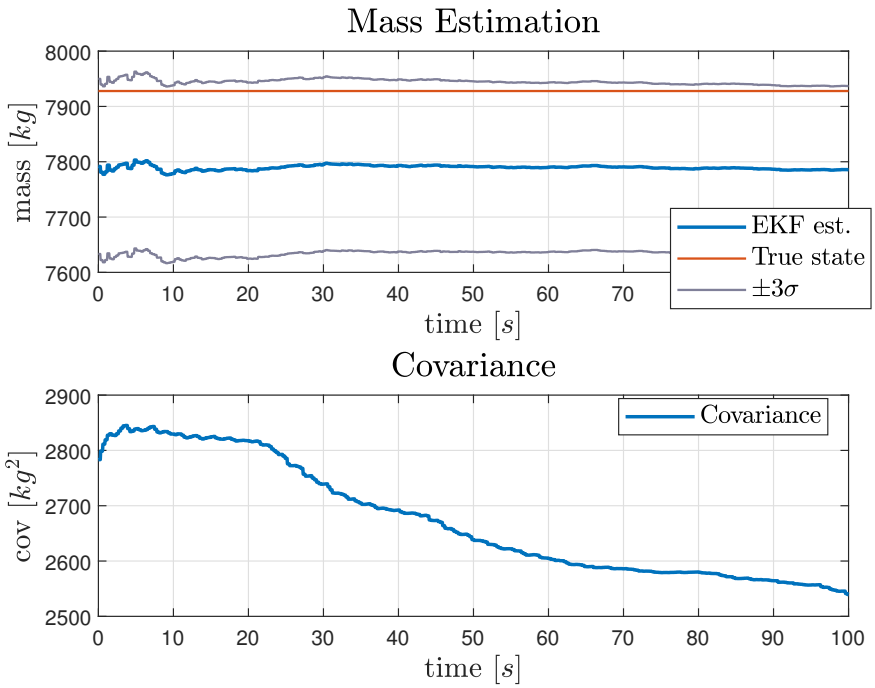
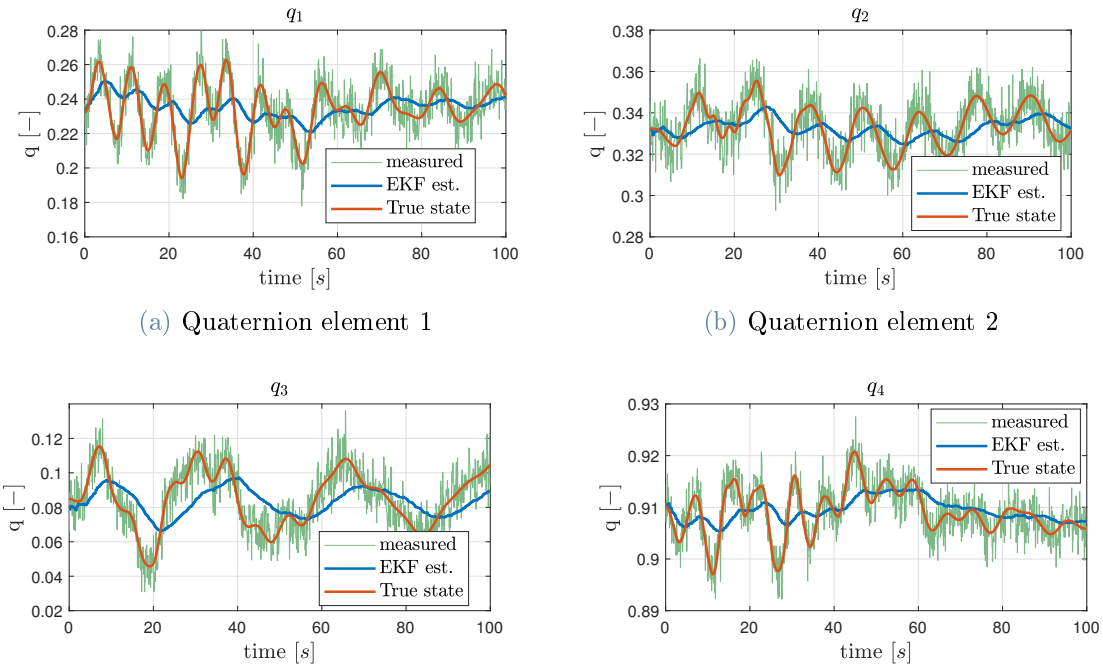


Figure 5.10: Mass estimation with EKF and rigid bodies.



(a) Quaternion element 1

(b) Quaternion element 2

(c) Quaternion element 3

(d) Quaternion element 4

Figure 5.11: Attitude estimate EKF with rigid bodies and incorrect Envisat mass.

5.2. MEKF results analysis

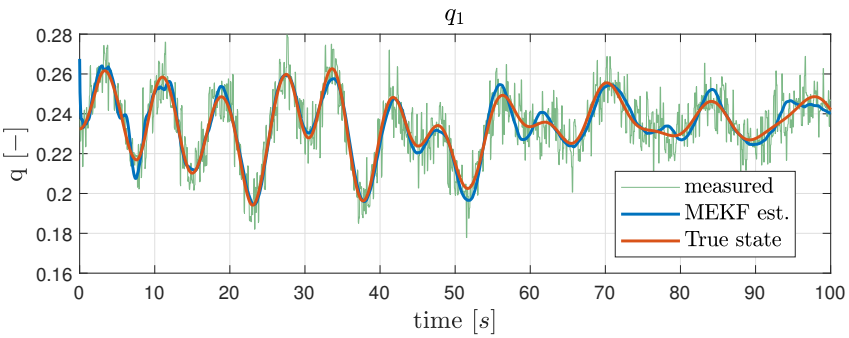
The MEKF performance analysis is carried out similarly to the EKF. The key distinction is in the tuning parameters of the MEKF, which uses a different state. Equation (5.8) shows the values of the tuning parameters.

$$\mathbf{Q} = 10^{-10}\mathbf{I}_{6\times6} \quad \mathbf{P}_0 = \begin{bmatrix} 0.8^2\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & 0.1^2\mathbf{I}_{3\times3} \end{bmatrix} \quad \mathbf{R} = 0.8^2\mathbf{I}_{3\times3} \tag{5.8}$$

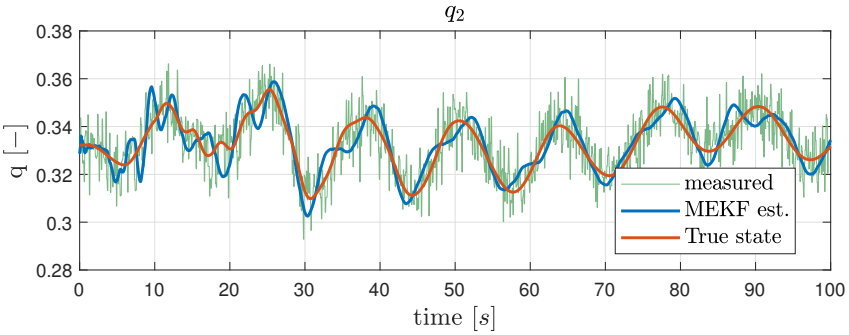
The same filter utilized for the EKF study is employed for parameter estimation. The collected data can then be analyzed. Figure 5.12 depicts the simulation results without the parameter estimation filter. It highlights that while the MEKF filter may estimate the true system state, it does so with considerable oscillatory behaviour, particularly for the third and fourth quaternion components. This behaviour can be better understood by looking at figure 5.13. In contrast to the EKF analysis, the auto-covariance values are not reported here. This is done because the MEKF computed covariance is linked to the quaternion error vector used as a state rather than the attitude quaternion itself. Table 5.4 shows the mean integrated errors relative to each component of the attitude representation. They will be compared afterwards to the modified model with rigid bodies assumption and with the EKF results.

e_1	e_2	e_3	e_4
$2.72e-03$	$4.65e-03$	$4.05e-03$	$1.68e-03$

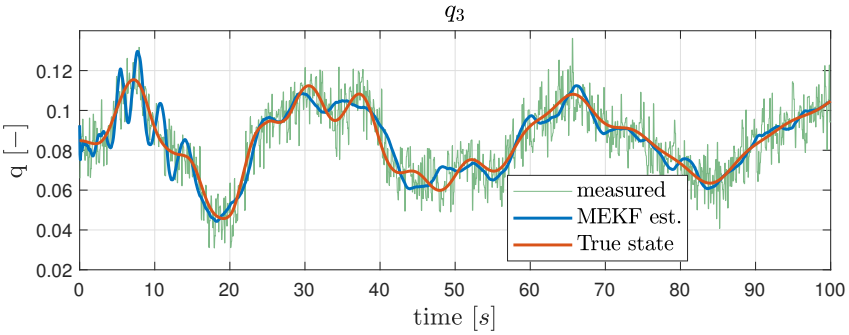
Table 5.4: Mean integrated absolute error of MEKF.



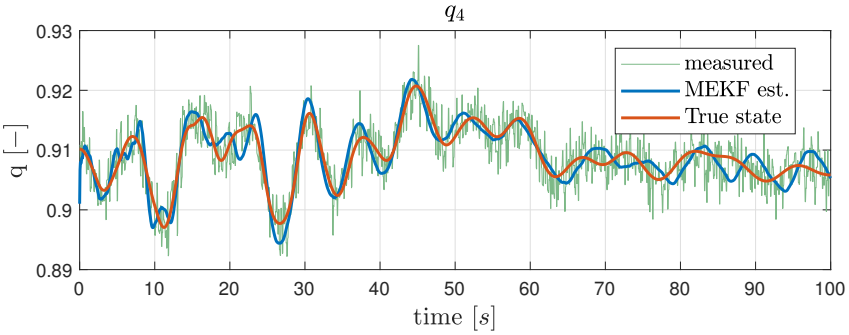
(a) Quaternion element 1



(b) Quaternion element 2

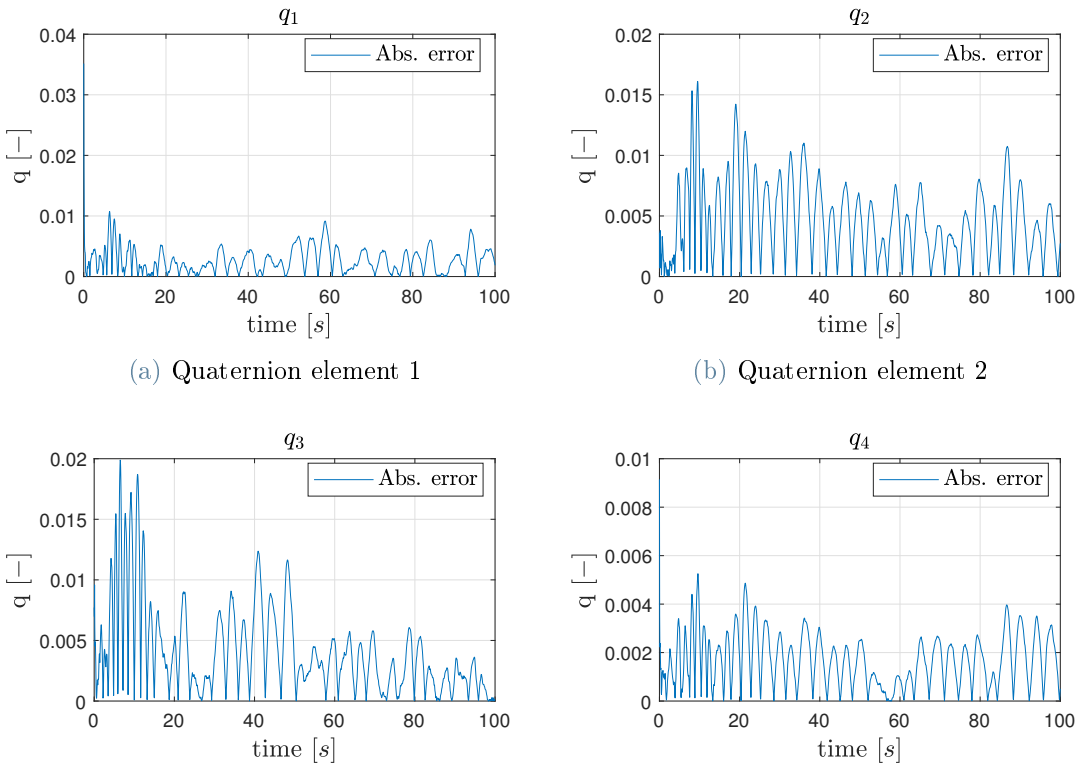


(c) Quaternion element 3



(d) Quaternion element 4

Figure 5.12: Attitude estimate obtained from MEKF without parameter estimation.



(a) Quaternion element 1

(b) Quaternion element 2

(c) Quaternion element 3

(d) Quaternion element 4

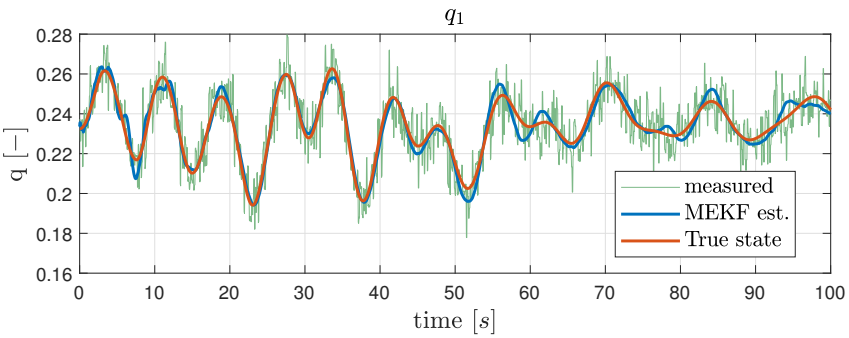
Figure 5.13: MEKF attitude estimate error.

Then, the simulation with all the spring stiffnesses set to $k = 10^7\ Nm/rad$, which aims to model a rigid implementation of the real system inside the FMU, is carried out. Figure 5.14 and figure 5.15 show respectively the estimation of the attitude and the attitude error. It can be clearly seen that the two results obtained from the flexible and rigid FMUs are very similar to one another. This observation is corroborated by the mean integrated absolute error results shown in table 5.4 and in table 5.5.

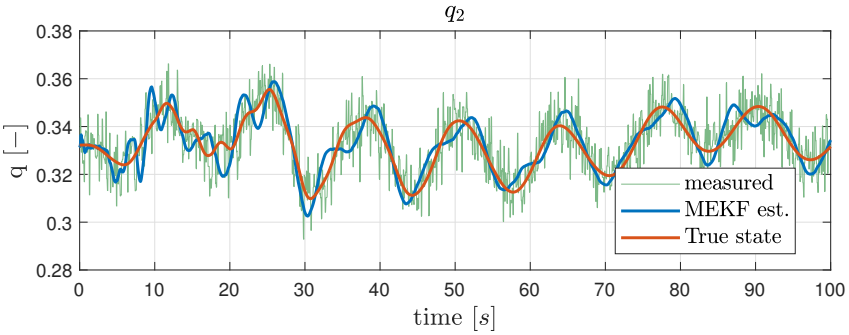
The whiteness test is performed over the error data derived from the MEKF simulations. Both error vectors did not pass the Anderson test.

e_1	e_2	e_3	e_4
$2.69e-03$	$4.63e-03$	$4.03e-03$	$1.68e-03$

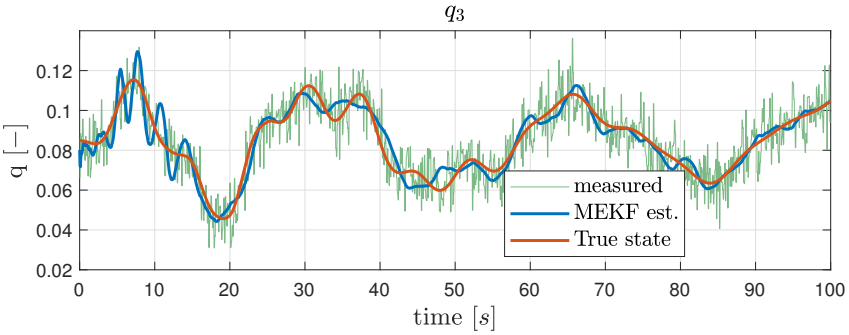
Table 5.5: Mean integrated absolute error of MEKF with rigid assumption



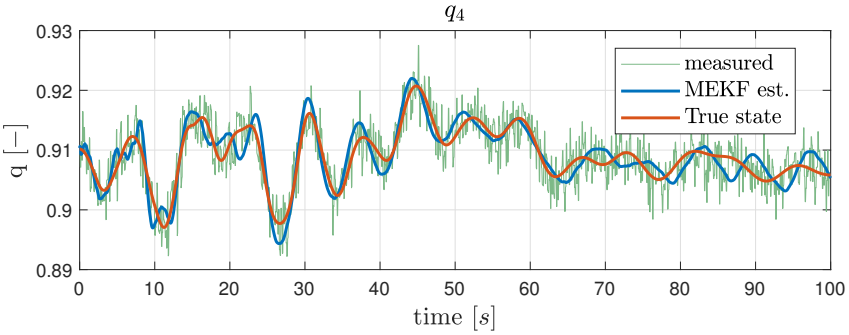
(a) Quaternion element 1



(b) Quaternion element 2

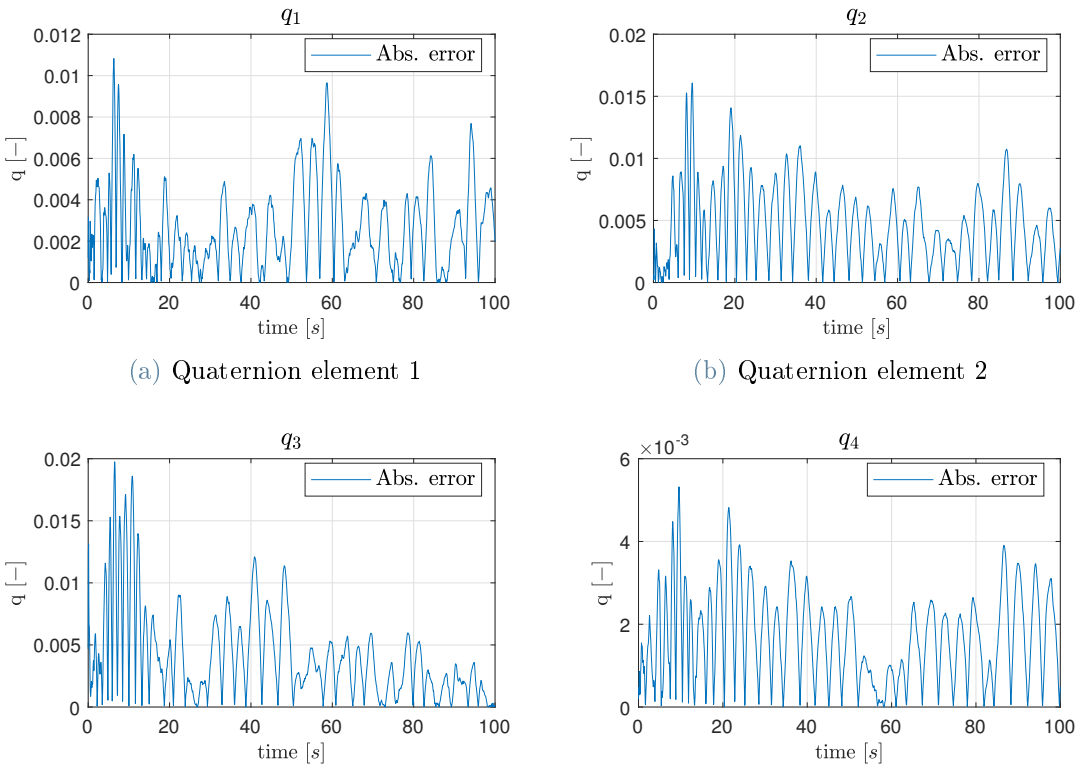


(c) Quaternion element 3



(d) Quaternion element 4

Figure 5.14: Attitude estimate obtained from MEKF without parameter estimation and with rigid bodies assumption.



(a) Quaternion element 1

(b) Quaternion element 2

(c) Quaternion element 3

(d) Quaternion element 4

Figure 5.15: MEKF attitude estimate error with rigid bodies assumption.

Similarly to what is done above for the EKF, the covariance is computed and plotted against a white Gaussian noise that has the same covariance as the error vector. The obtained graphs are shown in figure 5.16 and 5.17. The first figure depicts the covariance computation relative to the flexible MEKF implementation, and the second one is relative to the rigid implementation of the filter. The two graphs are again very similar. The main explanation for such behaviour is that the MEKF, as it is implemented, does not utilise the functionalities provided by the FMI standard. In particular, in the Kalman gain computation and the covariance update, it does not take advantage of the provided function `getDirectionalDerivatives` but instead uses a computed state transition matrix. It was not possible to use the build-in function mainly because the multiplicative filter does not use a state that is common with the FMU. The MEKF filter utilises the functionalities of the FMI standard only in the computation of the predicted state and of the predicted measurements. This consideration can further explain the similarities between the absolute error of the attitude of the rigid and flexible implementations. The MEKF, since it does not utilise major components provided by the FMU, is less dependent on the model implemented inside the FMU. The slight differences between the output provided

by the two different filters are not enough to significantly impact the estimation process.

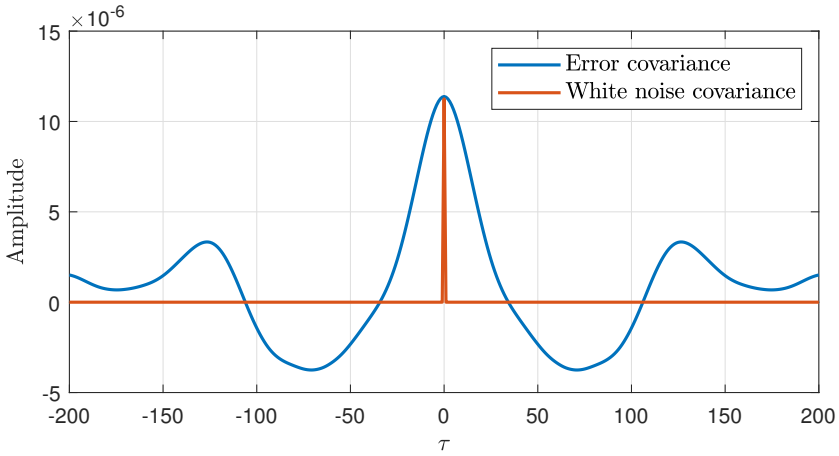


Figure 5.16: Covariance Analysis MEKF

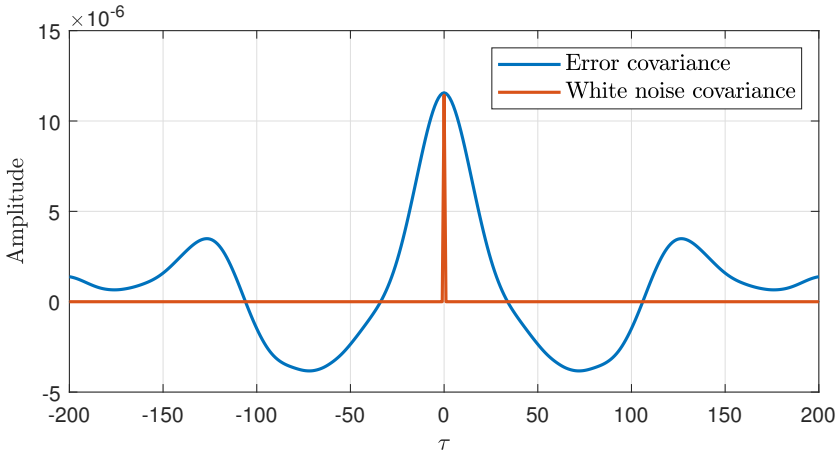


Figure 5.17: Covariance Analysis MEKF rigid

5.2.1. MEKF and mass estimation

The final step in the analysis of the MEKF performance is to integrate it inside the dual estimation process used to determine the mass of the true system. The mass estimation results are shown in figure 5.18. As it is possible to see, MEKF does not converge to the correct mass during the simulation process that spans 100 seconds.

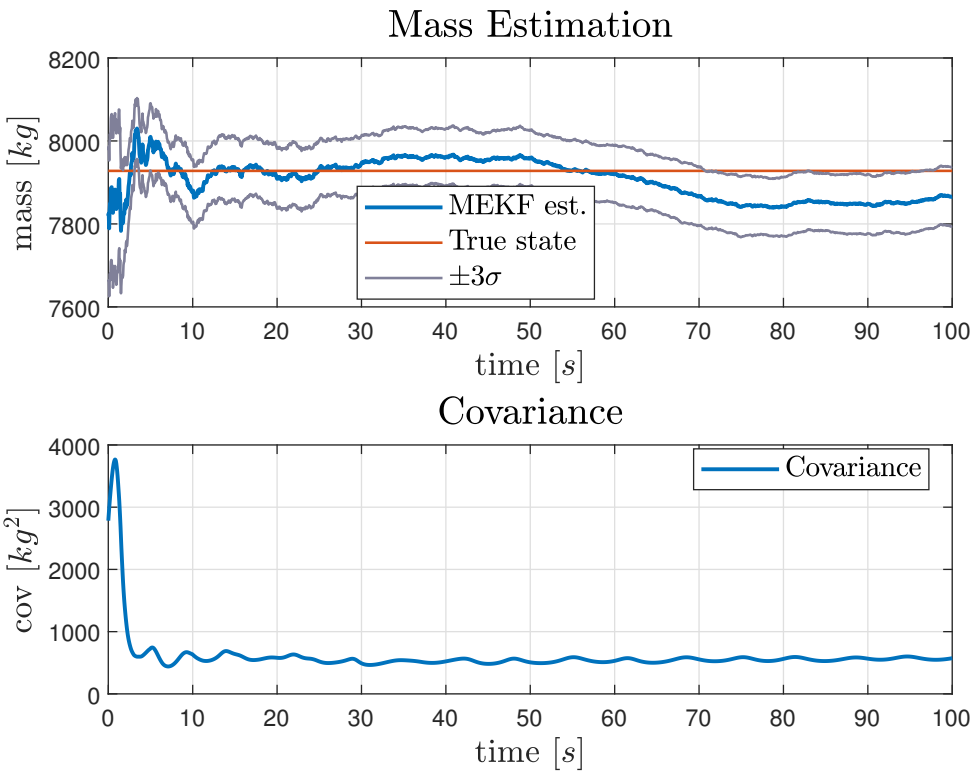
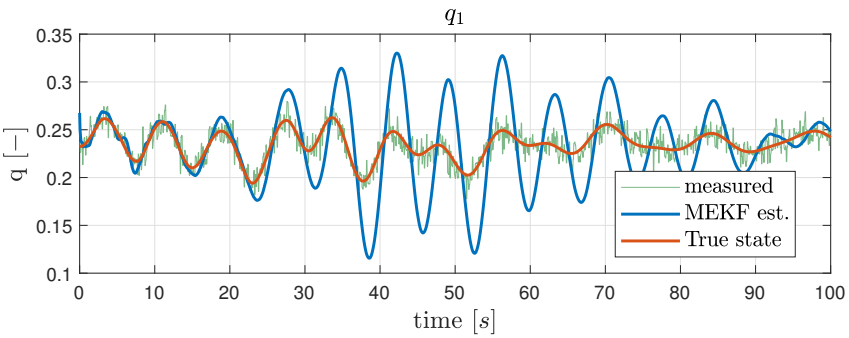
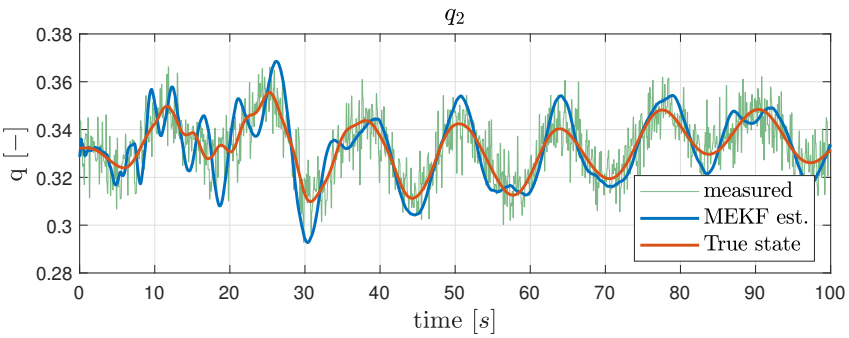


Figure 5.18: Mass estimate performed by MEKF with flexible bodies assumption

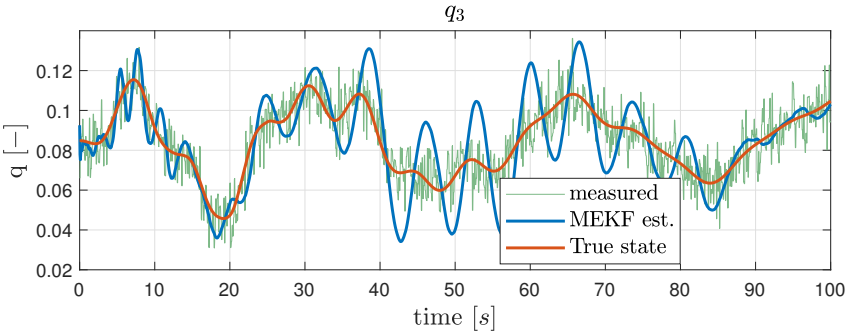
It is possible to see the results of the quaternion estimation process from figure 5.19. With the same tuning parameter used in the previous section, the filter cannot provide a reliable estimate of the satellite attitude. A strong oscillatory behaviour of the estimate of all the components of the attitude quaternion is highlighted. The MEKF estimator has difficulties when the estimated mass changes in time. This is most probably due to two concurrent factors. First, as explained before, the filter does not utilize the FMU function as EKF and thus has a poorer *comprehension* of the true system behaviour. Secondly, at each time step, the mass varies, modifying the estimated measurements coming from the FMU. With the inaccurate data coming from the MEKF, the parameter estimation algorithm struggles to reach convergence in the simulation time.



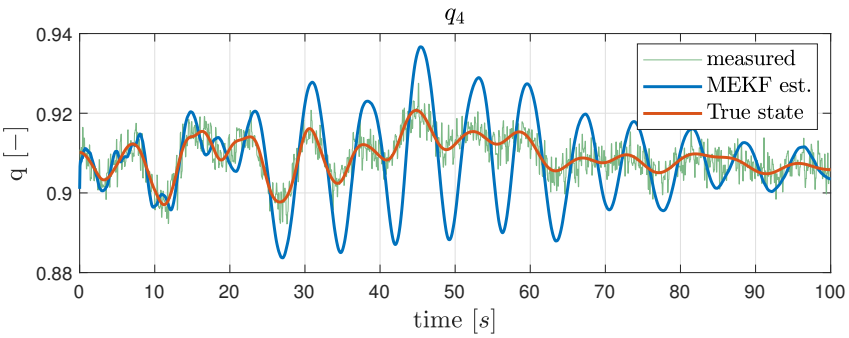
(a) Quaternion element 1



(b) Quaternion element 2



(c) Quaternion element 3



(d) Quaternion element 4

Figure 5.19: Attitude estimate obtained from MEKF with parameter estimation.

Lastly, the mass estimation results obtained from the MEKF with the rigid bodies assumption are reported in figure 5.20. These results are expected, given that the joint's accelerations are not used. The filter cannot perform an accurate estimation of the Envisat mass; on the contrary, the mass hovers around the initial guess.

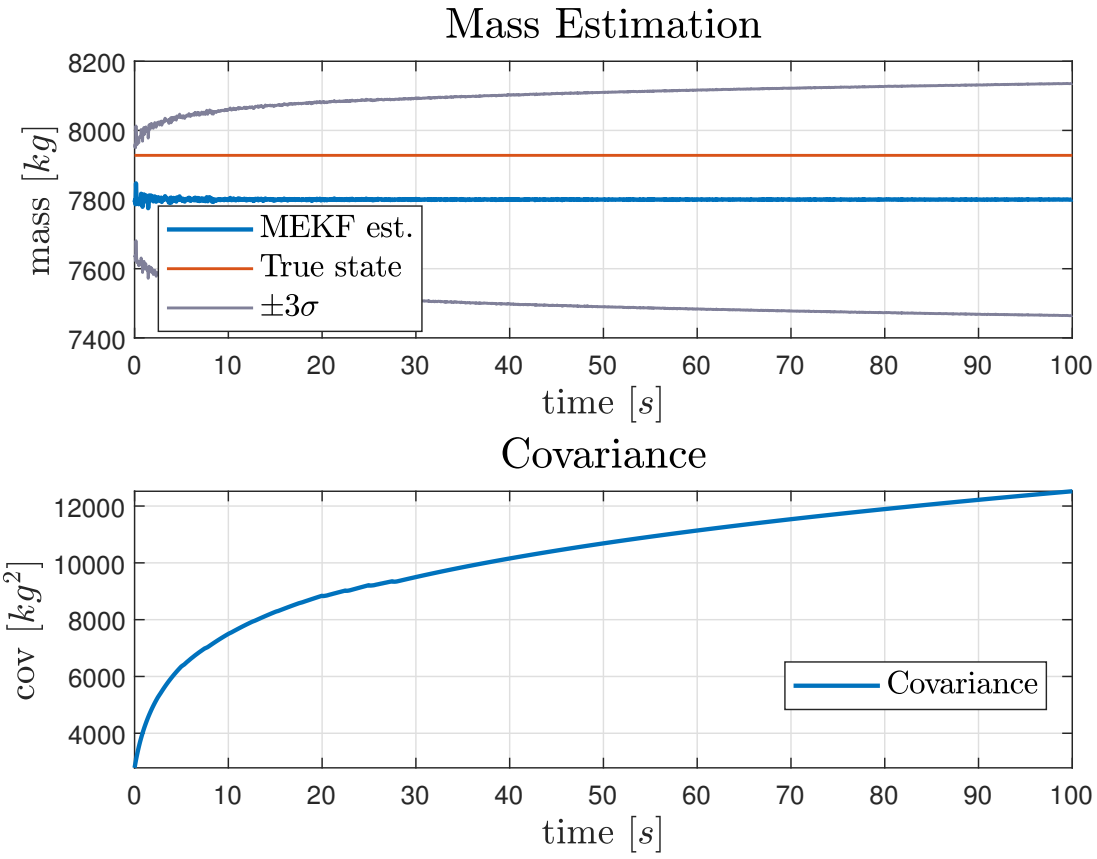


Figure 5.20: Mass estimate performed by MEKF with rigid bodies assumption

5.3. Comparision between EKF and MEKF

After all the data for the EKF and MEKF are presented, it is possible to compare the results obtained from the two different approaches.

Regarding the state estimation process, both the EKF and MEKF FMUs are capable of determining the attitude of the true system from the measurements with some minor differences in the errors, even if the MEKF implementation shows some oscillatory behaviours around the true state. Nonetheless, it is essential to highlight that the EKF with the flexibility assumption has the advantage of better representing the true system. This is derived from its error covariance being the most similar to the one of a white noise.

Regarding the mass estimation process, the EKF with the flexibility implementation perform the best and converges quickly to a correct estimate. Both rigid implementations failed in the estimation procedure; this is due to the differences between the true system and the one implemented inside the FMU, in addition to the exclusive use of the accelerations of the chaser satellite body. The MEKF with the flexible body assumption did not work either and, as explained above, this is because the multiplicative filter does not take advantage of the functions provided by the FMU. The MEKF does not need such functions to work since it is implemented and derived differently from the EKF, but in this way, it performs much worse with the dual filter implementation.

6 | Conclusion and Future Developments

This chapter deals with the conclusion and future developments. In particular, in section 6.1 the achieved objectives of this thesis are presented, whereas in section 6.2 some suggestions for future developments are proposed.

6.1. Conclusion

In chapter 3 the model implementation and validation procedure is presented. In particular, in section 3.1 an initial modeling approach with the Euler-Lagrange equation is shown. The derivation of such equations is complex, and it requires the use of symbolic manipulation. A simplified system consisting of a solar panel with three rigid elements connected with spring elements, the 7 DOFs manipulator, and the rigid chaser body, is constructed. The kinetic energy for each rigid element of the solar panel and the rigid chaser body is derived. The derived equations are complex due to the number of rotation matrices relating each solar array element to the neighboring one, resulting in a small set of large equations. The symbolic manipulation process can compute the dynamic equations by applying the Lagrange formalism. The process is mainly limited by the conversion of the symbolic equations into a function handle. The process with a further simplified system, without the manipulator, is time intensive, and it does not work at all with a more complex system. The Euler-Lagrange approach provides the smallest set of equations needed to model the dynamic system, but they are difficult to translate into a function that can be integrated. The translation procedure slows down mainly due to the simplification process in which the equation is written in a form that can be integrated. As a result, the Euler-Lagrange approach is not the most suited option for modeling complex rigid body systems.

In section 3.2 an alternative model implementation technique is explored. The approach consists in using the Modelica language inside the Dymola environment. A system is described by a set of equations that do not indicate the directionality of the exchange

of information between its various components. Using an acausal programming language such as Modelica allows for a model implementation that reflects the nature of the real system. Using a causal program such as Simulink provides a clear graphic visualization of the signal flow between blocks. Each block is connected with an arrow that symbolizes an assignment; one block's input is dependent on another's output. This procedure represents the calculation process rather than the modeled reality's structure. The acausal notation allows for the implementation of a model through the use of relationships between variables rather than expressing assignment.

The derivation procedure was straightforward: initially, a linearized model of the solar panel is derived, and then the obtained set of equations is used to match the eigenvalues and eigenvectors with the one obtained from a finite element analysis. The matching process is performed with a Gauss-Newton method, with which different weights are used to prioritize the minimization of the error on the eigenfrequencies or the minimization of the error on the modal shapes. Then, in the same chapter 3, in section 3.3, the spring stiffnesses of the solar panel model are derived.

The resulting set of spring stiffnesses is then validated in section 3.4. The validation process is carried out using a transient load analysis. The results from the validation process show that the model with the weights set to $w = 0.1$ more closely resembles the fem model. This behavior can mainly be attributed to the non-linearity of the true model dynamics.

Chapter 4 shows the implementation of the filters algorithm inside the FMU. The filter aims to provide a reliable estimate of the chaser orientation and of the Envisat mass, which is considered known but highly uncertain. This approach represents a new contribution to the attitude determination framework. Using the FMI standards allows for a flexible and easy-to-share implementation of the attitudes filters. In addition, the testing can be carried out in different environments than the one used for modeling the system.

In chapter 5, the results of the attitude filters are presented and divided for the EKF and MEKF implementations. Regarding the EKF, the filters can estimate the attitude with or without the flexibility effects. In both cases, the covariance of the filters converges. However, it is much faster for the Extended Kalman Filter with flexibility dynamics than the one with rigid body assumptions. For the mass estimation, only the EKF with the flexible model can provide a good estimate of the inertia parameter. This is due to two main reasons: the modeled rigid system does not correctly represent the true one, and the filter cannot take advantage of the acceleration measurements from the manipulator's joints. The inability to estimate the mass parameter correctly leads to a poor estimation

of the attitude parameter.

The MEKF filter implementation performed worse compared to the results obtained with the EKF. First, no major differences in attitude estimation are highlighted between rigid or flexible models. This can be explained by how the filter has been implemented. The filter does not use the provided functionalities of the FMU in the computation of the state Jacobian. The FMU is utilized mainly as a filter container and to compute the predicted state and measurements. The different implementation makes the MEKF less capable than the EKF in interpreting the proper system behaviour.

Regarding the mass estimation process, the measurements obtained with the MEKF are used as input to the parameter estimation filter, which is the same one utilized in the EKF implementation. Regarding the flexible model implementation, the results of the dual filtering implementation show that the MEKF has worse performance when compared to the EKF. This can be attributed to the different implementations of the MEKF, as explained above. In addition to the poor performance in the mass estimation, the filter does not provide a reasonable estimate even for the attitude parameters. A possible explanation can be found in the interaction process of the filters in the dual Kalman filtering scheme. Most probably, the ever-changing values for the Envisat mass break the stability achieved with the MEKF in the previous step, in addition to the wrong assumption on the Envisat mass. Finally, the MEKF with the rigid bodies assumption cannot provide an acceptable estimate for the mass or for the orientation quaternion.

To summarize, this work has shown the ability of the dual EKF implementation, paired with an FMU, to estimate the attitude of the chaser spacecraft and to measure the mass property of the target satellite. At the same time, the MEKF implementation shortcomings have been highlighted.

6.2. Future Developments

The conclusion of this work provided a full-picture view of the flexibility modelling and effects for the attitude and parameter estimation problem. This section gives some suggestions on how further developments may be carried out. In particular:

- This work focused on the determination of the mass parameter of the target satellite. Further analysis can be done in the estimation of the overall inertia matrix of Envisat.
- All simulations carried out in this thesis are done with a 100 second period. It would be beneficial to investigate the stability of the EKF filter for a longer simulation

time.

- A future research topic could be implementing a different filter than the EKF or MEKF within the FMU. An example could be the implementation of an Unscented Kalman Filter (UKF).
- Further developments can be done in modelling the system's flexible elements. While modelling the manipulator flexibilities with a lumped parameter approach is fine, it could be beneficial to model the flexible solar panel with a FEM-like code inside the Dymola environment.
- In future developments, it would be advisable to implement the parameter estimation filter directly inside the attitude EKF, in order to verify whether the enlargement of the state could provide any performance improvements.

Bibliography

[1] F. Aghili. A prediction and motion-planning scheme for visually guided robotic capturing of free-floating tumbling objects with uncertain dynamics. *IEEE Transactions on Robotics*, 28(3):634–649, June 2012. ISSN 1941-0468. doi: 10.1109/TRO.2011.2179581.

[2] H. Albu-Schäffer. Rokviss – robotics component verification on iss. *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, 2006.

[3] R. Biesbroek, L. Innocenti, A. Wolahan, and S. M. Serrano. E.deorbit – ESA's active debris removal mission. In *7th European Conference on Space Debris*, 2017.

[4] A. R. Center. State-of-the-art, small spacecraft technology. Technical report, National Aeronautics and Space Administration, October 2021.

[5] Y. Du and C. Wang. Dynamic coupling and control of flexible space robots. *International Journal of Structural Stability and Dynamics*, 20(09):2050103, 2020. doi: 10.1142/S0219455420501035. URL `https://doi.org/10.1142/S0219455420501035`.

[6] A. Ellery, J. Kreisel, and B. Sommer. The case for robotic on-orbit servicing of spacecraft: Spacecraft reliability is a myth. *Acta Astronautica*, 63(5):632–648, 2008. ISSN 0094-5765. doi: https://doi.org/10.1016/j.actaastro.2008.01.042. URL `https://www.sciencedirect.com/science/article/pii/S0094576508001197`.

[7] ESA. Active debris removal. Online, 2020. URL `https://www.esa.int/Space_Safety/Space_Debris/Active_debris_removal`. accessed: 2022-11-16.

[8] ESA. Esa's annual space environment report. Technical report, ESA Space Debris Office, 2022.

[9] M. A. P. "FMI". Functional mock-up interface specification. Online, May 2022. URL `https://fmi-standard.org/docs/3.0`. accessed: 21-11-2022.

[10] M. Ghani, N. Assadian, and R. Varatharajoo. Attitude and deformation coupled estimation of flexible satellite using low-cost sensors. *Advances in Space Research*, 69(1):677–689, jan 2022. doi: 10.1016/j.asr.2021.08.010.

[11] V. Joukov. Constrained dynamic parameter estimation using the extendedn kalman filter. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015.

[12] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, mar 1960. doi: 10.1115/1.3662552.

[13] S. A. Laughman, Christopher R.; Bortoff. Nonlinear state estimation with fmi: Tutorial and applications. In *American Modelica Conference*, 2020.

[14] F. L. Markley. Multiplicative vs. additive filtering for spacecraft attitude determination. *NASA's Goddard Space Flight Center*, 2003.

[15] D. Meng, W. Xu, C. Xu, and Z. Mu. Modeling and simulation study of flexible space robot for capturing large flexible spacecraft. In *Proceedings of the 32nd Chinese Control Conference*, pages 5837–5842, 2013.

[16] D. Meng, B. Liang, W. Xu, X. Wang, H. Liu, and X. Zhu. On the autonomous target capturing of flexible-base space robotic system. In *2014 13th International Conference on Control Automation Robotics and Vision (ICARCV)*, pages 1894–1899, 2014. doi: 10.1109/ICARCV.2014.7064605.

[17] NASA. On-orbit servicing, assembly, andmanufacturing (osam) state of play. Technical report, OSAM National Initiative, 2021.

[18] OHB. Target analysis report. *e.Deorbit Mission Phase B1*, 2016.

[19] D. Raina, S. Gora, D. Maheshwari, and S. V. Shah. Impact modeling and reactionless control for post-capturing and maneuvering of orbiting objects using a multi-arm space robot. *Acta Astronautica*, 182:21–36, may 2021. doi: 10.1016/j.actaastro.2021.01.034.

[20] S. Schneider and R. Cannon. Object impedance control for cooperative manipulation: theory and experimental results. *IEEE Transactions on Robotics and Automation*, 8 (3):383–394, jun 1992. doi: 10.1109/70.143355.

[21] B. Siciliano. *Robotics - Modelling,Planning and Control*. Springer, 2010. ISBN 978-1-84628-641-4.

[22] S. Ulrich, J. Z. Sasiadek, and I. Barkana. Modeling and direct adaptive control of a flexible- joint manipulator. *Journal of Guidance, Control, and Dynamics*, 35(1): 25–39, jan 2012. doi: 10.2514/1.54083.

[23] M. Wang, J. Luo, J. Yuan, and U. Walter. An integrated control scheme for space

robot after capturing non-cooperative target. *Acta Astronautica*, 147:350–363, jun 2018. doi: 10.1016/j.actaastro.2018.04.016.

[24] S. Šalinić. Determination of natural frequencies of a planar serial flexure-hinge mechanism using a new pseudo-rigid-body model (prbm) method. *The 6th International Congress of Serbian Society of Mechanics*, 2017.

List of Figures

List of Tables

List of Symbols

Variable	Description	SI unit
J	Jacobian	–
$\mathcal{L}$	Lagrangian	–
t	Time	s
U	Potential energy	J
$\mathbf{Q}_k$	Generalized non conservative forces	–
T	Kinetic energy	J
$\mathbf{T}_n^o(q)$	Homogeneus transformation from frame O to frame n	–
$\mathbf{A}_i^{i-1}(q_i)$	Homogeneus transformation from frame (i-1) to frame i	–
$\mathbf{A}$	State Jacobian	–
$\mathbf{B}$	Input Jacobian	–
$\mathbf{H}$	Measurements Jacobian	–
$\mathbf{Q}$	Process noise covariance matrix	–
$\mathbf{R}$	Measurements noise covariance matrix	–
$\mathbf{P}$	Error covariance	–
P	Probability	–
f_p	Probability density	–
E	Expectation operator	–

Continued on next page

Variable	Description	SI unit
e	State estimate error	–
$\hat{x}$	State estimate	–
x	True state	–
$\mathbf{L}$	Kalman gain	–
z	Measurements vector	–
$\mathbf{I}$	Identity matrix	–
$\mathbf{W}$	Process noise Jacobian	–
$\mathbf{V}$	Measurements noise Jacobian	–
f	Non-linear system equations	–
ν	Measurements noise	–
w	process noise	–
$\hat{q}$	Quaternion estimate	–
δq	Quaternion error	–
$\tilde{\omega}$	Measured angular velocity	rad/s
ω	Angular velocity	rad/s
b	ARW	$rad/\sqrt{hr}$
ν_v	Bias instability	rad/hr
Z	MEKF innovation	–
$\delta\rho$	First 3 components of δq	–
δX	MEKF state	–
$\breve{\mathbf{\Phi}}$	State transition matrix of MEKF	–
$\mathbf{\Phi}$	Error covariance transition matrix of MEKF	–
Θ	Mass parameter	kg
r	Parameter noise	kg
$\mathbf{Q}^{\theta}$	Parameter noise covariance	–

Continued on next page

Variable	Description	SI unit
$\hat{x}_k$	Corrected KF state	–
$\hat{x}_{k\|k-1}$	Predicted KF state	–
a^θ	Mass upper bound	kg
b^θ	Mass lower bound	kg
c	Sigmuoid steepness coefficient	–
$Sig(-)$	Sigmuoid function	–
$\mathbf{I}_c^c$	Inertia matrix of body c in ref. system c	kgm^2
$\mathbf{R}_c^o$	Rotation matrix frame c to frame o	–
r_c^o	Position vector of CoG of c in ref. frame O	m
$\mathbf{S}(-)$	Skew symmetric matrix operator	–
CoG	Center of gravity	–
d_i	Inner diameter	m
d_e	External diameter	m
E_m	Young modulus	MPa
G_m	Shear modulus	MPa
ν_m	Poisson ratio	–
ρ	Material density	–
$\mathbf{W}_w$	Weights matrix	–
$\hat{Y}$	Linear frequencies and modal shapes vector	–
$\bar{Y}$	FEM linear frequencies and modal shapes vector	–
k_i	Spring stiffness	Nm/rad
MSE	Mean square error	–
F_{in}	Transient load	N

Acknowledgements

This thesis represents the end of a journey that took me five years to accomplish. During this time, I've met wonderful people that helped me achieve this goal.

First and foremost, I have to thank my thesis supervisors, Professor Mauro Massari, Professor Riccardo Vescovini, and Professor Pierluigi Di Lizia. Without their assistance and the feedback I have received in most steps throughout the process, this paper would have never been accomplished. I would like to thank you very much for your support and for your suggestions over these past few months.

I would also like to show gratitude to my fellow university colleagues and friends; thanks to you, I will always remember the past five years as one of the funniest and happy periods of my life. I must also thank three friends and colleagues, Simo, Dani, and Buonfa, for giving me invaluable advice during this thesis work. I must show you all my gratitude because, without your support and friendly suggestions, I would not have been able to come so far in this work. I truly believe that you will be one of the most positive figures in my life in the coming years. Other than saying thank you, I only hope I can represent a good friend to you in the future.

Getting through my thesis work required more than academic support. I have many, many people to thank for listening and, at times, for tolerating my long monologues. I cannot begin to express my gratitude and appreciation for their friendship. Pietro and Alessandro have been unwavering in their personal support. They were the best company I could ever ask for during the long study hours, and with some jokes, they always cheered me up, especially when I needed it the most.

Most importantly, none of this could have happened without my family. My grandfather Franco with his practical sense and with his infinite wisdom, is and will always be a reference figure in my life. Finally, I must express my profound gratitude to my parents, Stefano and Roberta, and my brother Mattia for providing me with unfailing support and encouragement throughout my years of study. This accomplishment stands as a statement of your unconditional love, and I would not be able to arrive so far without them.
Thank you.