Preliminary Design and Development of MAST, a Multivehicle Analysis and Separation Tool

Björn Gäßler, Paul Acquatella B., Cèlia Yábar Vallès, Antonio Rinalducci, and Samir Bennani

The recent surge in demand for launch opportunities of multiple small satellites has stimulated further development of dedicated dispensers and new multi-payload concepts. A large market of adapters and deployment technologies has been created to compactly house future small satellites on existing launchers. However, these new launch scenarios come with a new set of challenges and requirements regarding satellite separation dynamics which are very particular to each dedicated mission or dispenser. Separation dynamics modeling in a multi-body simulation environment is therefore a critical capability to develop regarding those future mission scenarios. This motivates the objective of this paper which is to present the preliminary design and development of a dedicated multi-body dynamics simulation tool denominated MAST (Multivehicle Analysis and Separation Tool) to provide the means to verify the required separation requirements and to verify absence of collisions between the separated satellites and the upper stage during the separation processes. The implementation of this tool and a test–case are presented together with some results from a realistic use–case scenario involving a hypothetical VEGA SMSS flight mission. This preliminary design and development of MAST shows satisfying results in terms of the tool's requirements also briefly outlined in the paper as well as its potential to assist in future launch scenarios involving multivehicle separations.

Björn Gäßler, Paul Acquatella B., Cèlia Yábar Vallès, Antonio Rinalducci, and Samir Bennani; *Preliminary Design and Development of MAST, a Multivehicle Analysis and Separation Tool.* 8th ICATT (International Conference on Astrodynamics Tools and Techniques), 23 - 25 June 2021, Virtual.

PRELIMINARY DESIGN AND DEVELOPMENT OF MAST, A MULTIVEHICLE ANALYSIS AND SEPARATION TOOL

Björn Gäßler[(1)], Paul Acquatella B.[(1)], Cèlia Yábar Vallès[(2)], Antonio Rinalducci[(2)], and Samir Bennani[(2)]

[(1)] *DLR, German Aerospace Center, Institute of System Dynamics and Control Oberpfaffenhofen D-82234, Germany {bjoern.gaessler, paul.acquatella}@dlr.de*
[(2)] *European Space Agency ESA/ESTEC, Noordwijk, NL-2200, The Netherlands {celia.yabar.valles, antonio.rinalducci, samir.bennani}@esa.int*

ABSTRACT

The recent surge in demand for launch opportunities of multiple small satellites has stimulated further development of dedicated dispensers and new multi-payload concepts. A large market of adapters and deployment technologies has been created to compactly house future small satellites on existing launchers. However, these new launch scenarios come with a new set of challenges and requirements regarding satellite separation dynamics which are very particular to each dedicated mission or dispenser. Separation dynamics modeling in a multi-body simulation environment is therefore a critical capability to develop regarding those future mission scenarios. This motivates the objective of this paper which is to present the preliminary design and development of a dedicated multi-body dynamics simulation tool denominated MAST (Multivehicle Analysis and Separation Tool) to provide the means to verify the required separation requirements and to verify absence of collisions between the separated satellites and the upper stage during the separation processes. The implementation of this tool and a test–case are presented together with some results from a realistic use–case scenario involving a hypothetical VEGA SMSS flight mission. This preliminary design and development of MAST shows satisfying results in terms of the tool's requirements also briefly outlined in the paper as well as its potential to assist in future launch scenarios involving multivehicle separations.

1 INTRODUCTION

1.1 Background and Motivation

Launching and deploying multiple satellites at once by a single launch vehicle is a demonstrated capability that has been recently growing. The large amount of small spacecraft to be developed has also been rapidly increasing as evidenced in the recent years while current trends and forecasts keep showing that this shift towards launching multiple small spacecraft will continue in the future.
Recently, SpaceX has set a new record by launching and deploying 143 satellites in orbit with the Transporter 1 rideshare mission surpassing the record set by ISRO (the Indian Space Research Organization) in 2017 which carried and deployed a 714 kg satellite for Earth observation and another 103 additional nanosatellites weighing less than 10 kg each with the PSLV-C37 mission. Europe's VEGA small launcher recently demonstrated this capability by deploying 65 satellites (53 by the launcher and the remaining by a deployed satellite/module) using its new versatile Small Satellites Mission Service

(SSMS) dispenser during the VV16 flight [1, 2], see Figure 1. With this newly developed SMSS modular dispenser, ESA extended VEGA's capability to deploy multiple small satellites within one launch. In this regard, the surge in demand for these kind of launch opportunities for small satellites has stimulated further development of dedicated dispensers and new multi-payload concepts. In the meantime, a large market of adapters and deployment technologies has been created to compactly house multiple small spacecraft on existing launchers.
However, the VEGA new launch scenarios and modular dispensers come with a new set of challenges and requirements regarding satellite separation dynamics and clearance of potential collisions (with the upper stage during separation or between satellites themselves afterwards) which are very particular to each dedicated mission or dispenser. The importance of such capability to foresee plausible collision scenarios arises from the fact that after separation, the integrity of each stage and each deployed element must be kept in order to guarantee overall success of the mission at hand. In that sense, the development of an integrated tool for the analysis and simulation of such separation dynamics is desired. Separation dynamics modeling in a multi-body simulation environment is therefore a critical capability to develop regarding those future mission scenarios and a rapid mission preparation tool for multiple payload release is required. This motivates the development of a dedicated multi-body dynamics simulation tool to provide the means to verify the separation requirements and to ensure the absence of collisions between the separated satellites and the upper stage during the separation processes, and between the satellites themselves in the short– and medium term after separation.

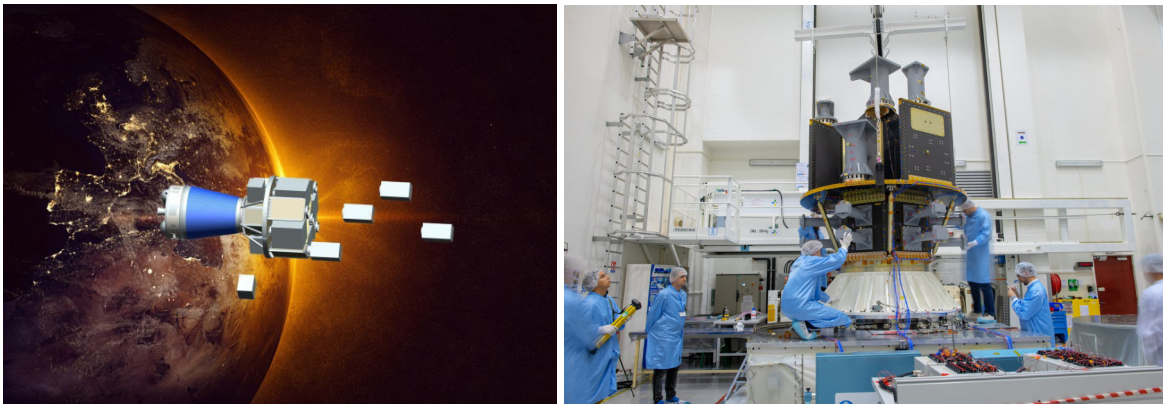

Figure 1: Deploying multiple small satellites virtually (left); SSMS vibration testing (right) Credit: European Space Agency – M. Pedoussaut

1.2 Previous work

Launch vehicle separation dynamics modeling tools were first carried out by NASA in partnership with the (former) Martin Marietta Corporation [3, 4, 5]. This was motivated given the importance to study launch vehicle stage separation; those efforts led to the development of the optimization tool for trajectory optimization and guidance design POST, a *Program to Optimize Simulated Trajectories* [6] and its more recent follow-up, POST2. Regarding multibody dynamics, several tools were also sought motivated by developing capabilities for simulating multiple flexible–body dynamics, separation analysis, and liftoff clearance analysis; among those tools were for instance TREETOPS [7, 8] and CLVTOPS [9].
Renewed interest in the subject of separation dynamics modeling during the 2000's led NASA towards the development of stage separation conceptual separation tools as *SepSim* and *ConSep* [10, 11, 12,

13]. Since stage separation dynamics modeling is clearly a very challenging task and a critical capability that must be considered in the preliminary studies of next generation launchers, a generalized approach to stage separation dynamics of launch vehicles was developed and coined as the *Constraint Force Equation* (CFE) methodology [14, 15, 16]. This methodology was first implemented into the *Program to Optimize Simulated Trajectories II* (*POST2*), mentioned before.
On the European side, early efforts on stage separation and multibody dynamics for space applications were also carried out for over 30 years by the European Space Agency (ESA) with the *Dynamic and Control Analysis Package* DCAP [17, 18, 19]. It provided capabilities to model, simulate, and analyze the dynamics and control performances of coupled rigid and flexible structural systems subject to structural and space environmental loads. More recent efforts for developing and consolidating knowledge in launcher dynamics led ESA to develop a launcher multibody dynamics simulators using DCAP as a backbone [20, 21, 22] and SimMechanics for multibody separation dynamics [23].
From DLR side, several efforts have been done to investigate an alternative approach to the methodologies found in recent literature by considering an *acausal* or *declarative* modeling approach with the MODELICA language [24, 25]. These efforts included a novel CFE implementation for a generic upper stage separation systems using the mentioned acausal modeling approach where internal joint loads are obtained automatically by the nature of such physics-based models [26]. In fact, these efforts proved very helpful regarding dynamics modeling as a critical capability for launch vehicle preliminary design studies where later improvements have allowed to consider such multidisciplinary modeling capability in combination with multi–objective optimization for preliminary guidance and control design of future space launchers [27, 28, 29, 30]. More recently, NASA is developing newer stage separation dynamics modeling tools using the Julia language [31, 32].
Introduced within the MATLAB® Release 2019b, Simscape Multibody® offers the possibility to disengage joints triggered by an *external mode* parameter [33]. This means that the original kinematic constraints imposed by a joint are removed and unrestricted motion between the originally coupled bodies are enabled. Making use of this new functionality, a *Weld Joint* model triggered by an external input can be used for the implementation of the CFE in MAST.

1.3 Objectives

The main objective of this paper to present the design and development of MAST, a *Multivehicle Analysis and Separation Tool*. The main purpose of the development of this tool is to simulate multi-payload (MPL) separation sequences and to provide the means to verify and to validate mission design separation sequences and requirements. For instance, the tool shall be able to verify not only that the separation requirements are fulfilled and that there are no collisions between the separated satellites and the LV upper stage during the separation process (“clearance”), but also to verify that this does not happen among the satellites themselves as well after a few orbits. Further detailing from the points above leads to the following sub-objectives: (a) definition, preliminary design and development of the tool; (b) implementation of pre–processing scripts for mission profile definition, MAST elements properties, and a procedure or script to automatically create new multi-payload separation models using a specific input file, and (c) implementation of post–processing scripts to analyze results, visualize mission scenarios (collision detection between payloads as well as between payloads and upper stage), perform Monte Carlo simulations with perturbed properties and eventually worst-case analysis to prove robustness.

1.4 Scope

The simulator has been developed and implemented using MathWorks' Simscape Multibody modeling toolbox which allows to physically model the composite of the scenario consisting of several separation elements (the upper stage, the payload adapter(s), the separation mechanisms, and the payloads themselves). Aside from a user–defined mission profile interface (pre-processing) that is capable to automatically assemble dynamical models, the tool also provides an interface for post-processing analyses and visualization within Simscape's Mechanic Explorer. The visualization setup with a pre-defined set of cameras allows to visualize the separation events sequence. Regarding post–processing, the tool is also capable to be combined with Monte–Carlo analyses targeting the analysis of different system requirements where models are used to simulate separation sequences with uncertain parameters. In that sense, this paper aims to give an overview of this recent activity regarding the preliminary design and development of MAST as a simulation tool for multi-payload separation dynamics. The implementation of this tool, some V&V unit tests, and Monte-Carlo results are presented for a realistic use–case of a hypothetical SSMS flight.

1.5 Outline

The remainder of the paper is organized as follows. Section 2 presents the preliminary requirements and architectural design of the tool. In Sections 3 present modeling and implementation aspects. Section 4 covers the tool workflow or *how to use MAST*, and Section 5 presents some simulation results of a realistic ESA use case. Finally, Section 6 presents brief conclusions followed by an outlook for future work.

2 ARCHITECTURAL DESIGN

2.1 Requirements

Within the scope of MAST, the requirements were divided into four categories: user requirements, functional requirements, design requirements, and system requirements. Within the scope of this paper, only a summary of the main requirements are presented in Table 1.

2.2 Basic Architectural Design

The main task of MAST is to simulate and analyze multi-payload separations from an upper stage. In this subsection the basic architectural design of the Simulink/Simscape model is described. As it can be seen in Figure 2 the MAST model consists of seven main blocks with their main functionalities being stated below:

- **Environment**: Provides reference frames and the implementation of a gravity model with additional perturbations due to non-spherical earth and residual atmosphere.
- **Launcher Dynamics**: Specify the orbital motion of the upper stage with its payloads.
- **Upper Stage**: Contains mechanical and geometrical properties of the upper stage, including predefined payload attachment points.
- **Separation Mechanism**: Holds two essential functionalities for multi-payload separation simulations - first the implementation of a Constrain Force Equation (CFE) to ensure rigid connection between a payload and the upper stage before separation and free motion afterwards - and second the implementation of actuators to generate a separation impulse.

Table 1: Summary of MAST Main Requirements

Main Requirements	Description
REQ-MAIN-01	The tool shall be able to simulate multi-payload separation sequences from a launcher upper composite dispenser.
REQ-MAIN-02	The tool shall be able to analyze the possible collisions between the separated payloads and the launcher upper composite, i.e., be able to validate or cross–check a mission design based on a sequence of separation events.
REQ-MAIN-03	The tool shall be able to provide a means of representing a multi-payload separation scenario where mission specific characteristics are defined.
REQ-MAIN-04	The tool shall be flexible enough to accommodate several configurations of launcher upper composite (upper stage, S/C adapter and dispenser) and multi-P/L (satellites).
REQ-MAIN-05	The tool shall contain an automatic routine to create a new multi-payload separation simulation model based on a user defined input file.
REQ-MAIN-06	The tool shall be able perform Monte-Carlo simulations for analysis of different separation scenarios.
REQ-MAIN-07	The tool shall be able to visualize the events sequence, payload separation, and collisions (if any).

- **Payloads**: Contains mechanical and geometrical properties of each payload.
- **Sensing**: Contains sensors to measure all quantities required for separation and collision analysis. This includes measurements of distances between the bodies' Center of Gravity and between their actual geometries/surfaces.
- **AOCS System**: Attitude- and Orbit Control System includes an AOCS Mode Management as well as sensors, controller and actuator implementations to follow attitude and orbit commands.

In the next chapter a closer description of the implementation of these elements will be given.

3 MODELING AND IMPLEMENTATION

MAST is based on MATLAB®'s physical modeling toolboxes Simscape® and Simscape Multibody®. Simscape allows to model multi-domain physical systems within the Simulink® environment. It is designed to emulate physical systems' appearance and provides the fundamental building blocks for different physical domains such as electrical, hydraulic, mechanical, thermal and more. In contrast to signal-flow based modeling in Simulink, it is based on acausal signal flows and physical connections exchanging energy through their bidirectional connection ports [34]. Simscape Multibody (formerly known as SimMechanics®) is a toolbox extending Simscape to multi-body simulations of 3-dimensional mechanical systems using predefined body blocks, joints, sensors, forces etc. from

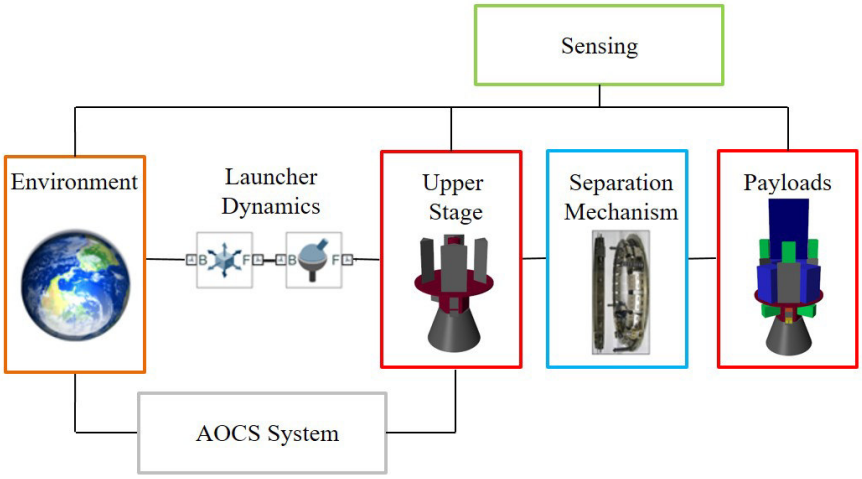

Figure 2: MAST Architectural Design

which the corresponding differential algebraic equations (DAEs) are self-generated [34]. The Multibody toolboxes comes with the *Mechanics Explorer*, a visualization tool that lets the user visually explore the model. For further information on Simscape or Simscape Multibody, the reader is referred to MathWorks® products and documentation [34, 35]. Due to its acausal multi-body simulation capabilities combined with direct integration of Simulink control toolboxes, modeling and design techniques, Simscape Multibody is a convenient and user-friendly framework for the application sought in this work which falls within the multi-vehicle separation dynamics and analysis context. In the following section the implementation of the MAST tool will be described. At the time of publication of this paper, the MAST tool was implemented in MATLAB Release 2019b. Models with more than one physical connection line were connected using the *Simscape Bus* which allows to bundle conserving connections such as non–directional connection lines between Simscape Multibody conserving ports into one Simscape Bus line [34] improving significantly the clarity of the MAST model.

3.1 Environment

This subsection introduces the MAST environment model. It contains reference frames, a gravity model including perturbations, and an atmosphere model which are necessary to simulate the orbital dynamics of an upper stage (US) configuration with several payloads (PLs) both attached and separated.

The first reference frame is the Earth-Centered-Inertial frame (ECI). Its origin is placed in Earth's center of gravity with the x–axis pointing to the vernal equinox, its z–axis points to the north pole and the y–axis completes a right-hand set. The Earth–Centered–Earth–Fixed frame (ECEF) shares the same origin and xy–plane with the x_{ECEF}–axis pointing towards the Greenwich meridian. It rotates around the z_{ECI}–axis with Earth's angular velocity $\boldsymbol{\Omega}_{\text{ECI}} = [\,0 \quad 0 \quad \omega_E\,]^T$ with $\omega_E = 7.2921158553^{-5}$ rad/s [36]. For control and pointing purposes, it is convenient to define a reference frame relative to the spacecraft's orbit. For this, the origin of a Local-Vertical-Local-Horizontal frame (LVLH) is placed in the spacecraft's center of gravity. Its z–axis is pointing towards the center of Earth, and its y–axis points along the negative orbit normal. The x–axis completes a right-hand set. In case of a circular orbit, the x_{LVLH}–axis is aligned with the orbital velocity of the spacecraft. The basis vectors

of the LVLH coordinate system, expressed in the ECI–frame can be determined as follows [36]:

$$\mathbf{z}_{\mathrm{LVLH}} = -\frac{\mathbf{r}_{\mathrm{ECI}}}{\|\mathbf{r}_{\mathrm{ECI}}\|} \tag{1a}$$

$$\mathbf{y}_{\mathrm{LVLH}} = -\frac{(\mathbf{r}_{\mathrm{ECI}} \times \mathbf{v}_{\mathrm{ECI}})}{\| (\mathbf{r}_{\mathrm{ECI}} \times \mathbf{v}_{\mathrm{ECI}}) \|} \tag{1b}$$

$$\mathbf{x}_{\mathrm{LVLH}} = \mathbf{y}_{\mathrm{LVLH}} \times \mathbf{z}_{\mathrm{LVLH}} \tag{1c}$$

where $\mathbf{r}_{ECI}$ and $\mathbf{v}_{ECI}$ are the spacecraft's position and velocity in the ECI frame. The direction cosine matrix (DCM) from LVLH to ECI frame can be expressed as:

$$\mathbf{R}_{\mathrm{ECI/LVLH}} = [\mathbf{x}_{\mathrm{LVLH}} \quad \mathbf{y}_{\mathrm{LVLH}} \quad \mathbf{z}_{\mathrm{LVLH}}] \tag{2}$$

Gravity was implemented using Simscape's *Inverse Square Law Force* block. This block calculates a force proportional to the inverse of the squared distance between the two frames attached. Therefore one *Inverse Square Law Force* block was necessary for the upper stage and one additional block for each payload. The magnitude of the gravity force depends on a force constant which is the earth standard gravitational parameter $\mu = G \cdot M_{\mathrm{Earth}}$ multiplied by the mass of the upper stage or corresponding payload. Perturbations due to Earth deviating from a perfect spherical shape were

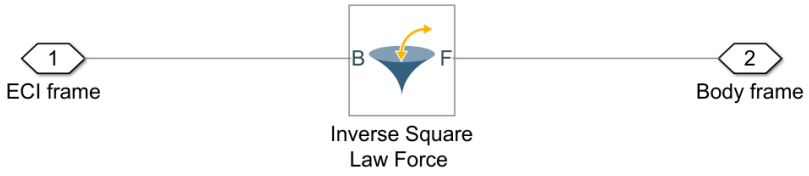

Figure 3: Gravity Implementation using Simscape's *Inverse Square Law Force* block.

taken into account by considering a second-order spherical harmonic geopotential model including only the zonal harmonics. These perturbing forces acting on the spacecraft in flight were calculated as:

$$\mathbf{F}_{\mathrm{pert}} = m \cdot \mathbf{a}_{\mathrm{pert}} \tag{3}$$

where m is the mass of the corresponding body (US or PL) and $\mathbf{a}_{pert}$ are perturbing accelerations due to the non-spherical shape of Earth. Due to the limited time–duration of the simulations, $\mathbf{a}_{\mathrm{pert}}$ are considered up to J2 and J3 effects since higher order perturbations can be deemed negligible. In that sense, $\mathbf{a}_{\mathrm{pert}}$ is given by [36]:

$$\mathbf{a}_{\mathrm{pert}} = \mathbf{a}_{\mathrm{J2}} + \mathbf{a}_{\mathrm{J3}} \tag{4}$$

with

$$\mathbf{a}_{\mathrm{J2}} = -\frac{3}{2} J_2 \left(\frac{\mu}{r^2}\right) \left(\frac{R_\oplus}{r}\right)^2 \begin{bmatrix} \left(1 - 5\left(\frac{z}{r}\right)^2\right)\frac{x}{r} \\ \left(1 - 5\left(\frac{z}{r}\right)^2\right)\frac{y}{r} \\ \left(3 - 5\left(\frac{z}{r}\right)^2\right)\frac{z}{r} \end{bmatrix} \tag{5}$$

$$\mathbf{a}_{\mathrm{J3}} = -\frac{1}{2} J_3 \left(\frac{\mu}{r^2}\right) \left(\frac{R_\oplus}{r}\right)^3 \begin{bmatrix} -5\left(7\left(\frac{z}{r}\right)^3 - 3\left(\frac{z}{r}\right)\right)\frac{x}{r} \\ -5\left(7\left(\frac{z}{r}\right)^3 - 3\left(\frac{z}{r}\right)\right)\frac{y}{r} \\ 3\left(10\left(\frac{z}{r}\right)^2 - \frac{35}{3}\left(\frac{z}{r}\right)^4 - 1\right) \end{bmatrix} \tag{6}$$

where $r = \|\mathbf{r}\|$, is the norm of the spacecraft position vector, μ is the earth standard gravitation parameter, $R_\oplus$ is the earth equatorial radius and J_2 and J_3 are zonal coefficients with $J_2 = 1.08262668355^{-3}$ and $J_3 = -2.53265648533^{-6}$. Furthermore, aerodynamic drag forces due to residual atmosphere were calculated as follows [36]:

$$\mathbf{F}_{\text{aero}} = -\frac{1}{2} \cdot \rho \cdot c_{\text{D}} \cdot S \cdot \|\mathbf{v}\| \cdot \mathbf{v} \tag{7}$$

where ρ is the atmospheric density, c_{D} is the spacecraft's drag coefficient, S is the spacecraft area projected along the direction of motion and $\mathbf{v}$ is the relative velocity of the spacecraft with respect to the atmosphere. To obtain the density ρ, the MSIS–86 atmosphere model was implemented [37] since it is a common model of temperature, density, and composition in the upper atmosphere. Since it is assumed that earth's atmosphere rotates with the planet without any shearing, $\mathbf{v}$ is equal to the velocity of the spacecraft with respect to the ECEF frame. Moreover, solar pressure might be also taken into account but it can be neglected since this perturbation is much less than the drag forces at the considered altitudes. Perturbing forces were calculated using standard Simulink blocks and introduced to the Simscape environment using Simscape's *External Forces and Torques* block as shown in Figure 4 for the aerodynamic drag.

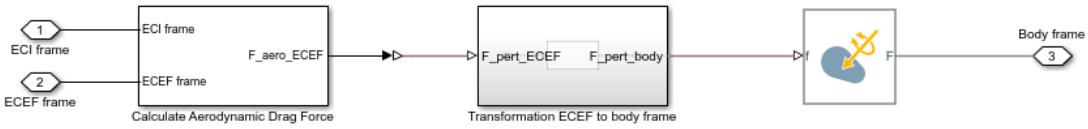

Figure 4: Implementation of Aerodynamic Drag using Simscape's *External Force and Torque* Block

3.2 Launcher Dynamics

To model the orbital dynamics of the upper stage configuration, Simscape's *Cartesian-* and *Spherical Joint* blocks were used, see Figure 5. These blocks specify the motion of a body, in this case the upper stage with respect to a reference frame. Within MAST, the Earth–Centered–Inertial frame as described in Chapter 3.1 was chosen as reference frame, whereby the *Cartesian Joint* enables three translational degrees of freedom (DoF) and the *Spherical Joint* three rotational DoF. Within these two blocks, the initial conditions of the upper stage for position, velocity, angular rate as well as initial attitude angles are defined as *state targets*. Orbital dynamics are governed by external accelerations (due to gravity, residual atmosphere, aerodynamic drag, etc.) acting on the corresponding body. Within MAST, these accelerations were implemented in the environment model as described in the previous subsection. Correct implementation of the orbital dynamics and environment model were verified by directly comparing orbital elements with results from the commercially available and well–established ASTOS tool [38].

Figure 5: Implementation of Launcher Dynamics using *Cartesian-* and *Spherical Joints*

3.3 Upper Stage

The implementation of the upper stage model can be divided in three main functionalities: mechanical properties, geometry model and definition of payload attachment points. The geometrical model consists of several Simscape body elements including *Extruded*, *Cylindrical*, and *Revolved Solids*, to approximately represent the physical SSMS dispenser geometry. A modular implementation was chosen such that different upper stage configurations can be easily realized. Figure 6 shows different configurations of the upper stage with the main modules being named.

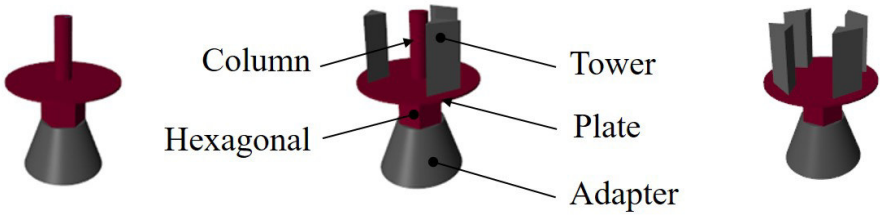

Figure 6: Upper Stage in different Configurations with main Modules

At each upper stage module, except the adapter, payloads can be attached to specific attachment points with the possibility of adding dispensers for CubeSats to the *Hexagonal*-element (HEX). For every upper stage module these attachment points are pre–defined and routing between the attachment point and the corresponding payload, respectively its separation system, takes place within the upper stage model.

3.4 Separation Mechanisms

3.4.1 Constraint Force Equation Methodology

The core part of multivehicle separation modeling is the separation dynamics and its implementation. This can be done for instance using the *Constraint Force Equation* (CFE) methodology [14, 15, 16]. In the context of payload separation, a CFE implementation is very useful to ensure that all payloads are rigidly connected to the upper stage during *joint motion* and that they follow their respective motion after separation. The CFE methodology is a highly intuitive method consisting of the computation of joint loads and constraints (internal forces and torques) together with their application as external forces and torques on each body independently, see [14, 15, 16]. In consequence, the CFE joint model simply augments the external loads of the system [15]. In general, the CFE methodology allows to consider any type of joint which allows (or forbids) any specific relative motion between bodies and redundancy of joints when necessary. Generally speaking, we consider for instance the constrained motion of two rigid bodies A and B connected by a *single joint* between two points $\overline{A}$ and $\overline{B}$ in their respective body–frames. Defining unit–vectors $\mathbf{e}$ for particular directions (depending of the considered constrained motion) of the corresponding body–frame (A or B), to constraint the relative motion between the bodies it is required that:

$$(\mathbf{r}_B - \mathbf{r}_A) \cdot \mathbf{e}_A = \mathbf{0}, \quad \text{for each direction constrained} \tag{8a}$$

$$\mathbf{e}_A \cdot \mathbf{e}_B = \mathbf{0}, \quad \text{for constrained (perpendicular) direction pairs} \tag{8b}$$

where $\mathbf{r}_A$ (resp., $\mathbf{r}_B$) is the position vector to point $\overline{A}$ (resp., $\overline{B}$). This means that the distance between the two points of a particular direction must remain fixed and that three properly selected two-unit-vector sets must remain perpendicular. Eq. (8) has to be differentiated twice with respect to time so

that the resulting relationships involve the unknown accelerations and angular accelerations, thus finally being able to couple them with the equations of motion. In other words [14], these equations are given by $\ddot{\mathbf{g}} = \mathbf{0}$ which are generalized constraint equations of the joint where $\mathbf{g}$ represents the nondifferentiated constraints in Eq. (8). An important aspect of the CFE methodology relevant to this work is the accuracy and computational effort of the joint loads solution; these are sensitive to computational error and initial joint missalignment [15] since $\ddot{\mathbf{g}} = \mathbf{0}$ allows perturbations to grow linearly with time. To handle such concern, the CFE algorithm is typically augmented with a stabilization technique known as Baumgarte stabilization [14, 39, 40]. This particular stabilization technique consists on replacing the generalized constraint equations by $\ddot{\mathbf{g}} + 2\eta\dot{\mathbf{g}} + \eta^2\mathbf{g} = \mathbf{0}$ which is an asymptotically stable ODE (for $\eta > 0$); however, at the expense of more computational effort. In [26] it was shown how using MODELICA a CFE implementation (including the Baumgarte stabilization) using an acausal modeling approach helps to obtain the internal joint loads automatically [26]. The SimMechanics implementation presented in [23] solved the CFE and stabilization issues by considering the nondifferentiated and once differentiated forms of $\mathbf{g}$ in closed–loop with a PD–regulator with tunable gains $\mathbf{K}_\mathrm{P}$ and $\mathbf{K}_\mathrm{D}$.

3.4.2 Simscape Implementation

Introduced within the MATLAB Release 2019b, Simscape Multibody offers the possibility to disengage joints controlled by an external *mode*-parameter [33]. This means that the original kinematic constraints imposed by a joint are removed and unrestricted motion between the originally coupled bodies are enabled. Making use of this functionality one *Weld Joint* block per Payload with an external input was used to implement the CFE in MAST (see Figure 7). Separation is triggered by switching the *mode*-parameter input from 0 (*joint motion*) to -1 (*free motion*) and thereby disengaging the joint.

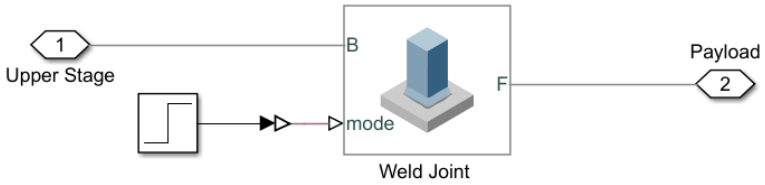

Figure 7: CFE Implementation using Simscape's *Weld Joint* Block

After separation, free translational and rotational motion between the two bodies needs to be ensured with (at least) an initial impulse to generate a separation. Within MAST, this functionality is divided into two blocks:

- CFE-block: Implementation to ensure a rigid connection before and free motion after separation (using a *Weld Joint* block or an implementation using Eq. (**??**))
- Separation System–block: Implementation of actuators to generate a separation impulse

Figure 8 shows on the left joint constraint forces and torques as well as the corresponding joint's *mode*-parameter for an exemplary application. It can be seen that constraint forces and torques are non-zero before separation at *t = 10* s (while the CFE is active) and become zero as soon as separation is triggered by switching the *mode*-parameter to -1, thereby disengaging the joint and enabling free motion. As mentioned above, the second functionality of the separation mechanism is to introduce a separation impulse [23]. This impulse is caused by several actuators such as compressed springs or

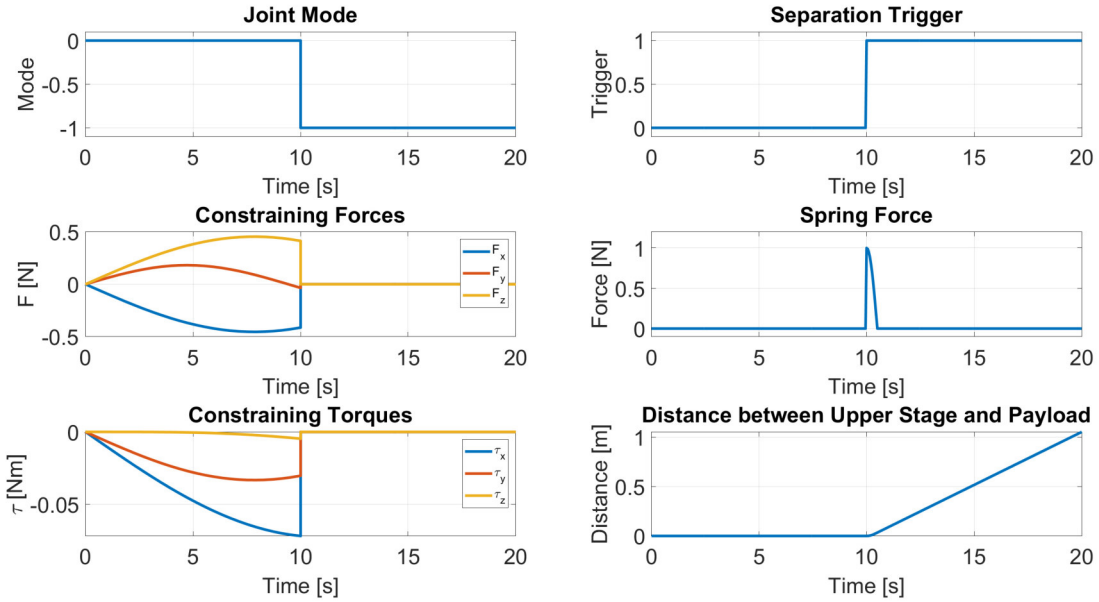

Figure 8: CFE Characteristics and Actuator Dynamics for a Separation at *t=10* s

spring–damper systems which are released when a payload is separated. Within MAST, an impulsive, non–damped spring was implemented as actuator where the spring-force is calculated as follows [23]:

$$F_{\text{Actuator}} = \begin{cases} F_{\max} \cdot \cos\left(\sqrt{\frac{k}{m}} \cdot (t - t_{\text{sep}})\right) & \text{, if } t \geq t_{\text{sep}} \text{ and } (t - t_{\text{sep}}) < t_{\text{impulse}} \\ 0 & \text{, otherwise} \end{cases} \tag{9}$$

with $F_{\max}$ being the maximum spring force, k is the spring constant, m the release mass and t_{sep} the time of separation. The actuator response can be seen in Figure 8 on the right. As soon as a separation is triggered an actuator force is generated (middle plot) and applied to the connected bodies. Since the joint in between the payload and upper stage is disengaged at this time their distance increases immediately (bottom plot). The spring model as implemented in Simscape is shown in Figure 9. The spring force is introduced to the system by making use of an actuated *6-DoF Joint* with the separation direction being parallel to the surface normal at the attachment point on the upper stage.

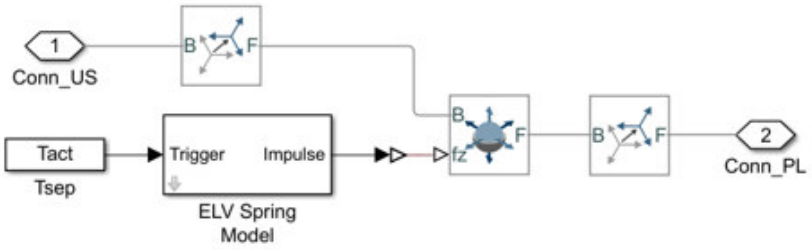

Figure 9: Actuator Implementation

For each physical actuator of the separation system one spring model has to be added to the separation system model. Besides regular payloads, the MAST tool should also be able to simulate the separation of CubeSats from their dispensers. In contrast to regular payloads the separation of a CubeSat from

its dispenser is limited to a 1-dimensional translational motion along the dispenser rail before it has completely left the dispenser. In MAST this was solved by connecting a *Weld-* and a *Prismatic Joint* in parallel. Same as with regular payloads the *Weld Joint* provides the CFE implementation and guarantees a rigid connection between upper stage and CubeSats before separation. After separation the *Prismatic Joint* remains engaged enabling a 1-dimensional translational motion along the joint's z–axis. The distance between the upper stage surface and the CubeSat is measured, and once the distance exceeds the dispenser rail length, the *Prismatic Joint* is disengaged – enabling the CubeSat to move freely. Figure 10 shows the implementation of a CubeSat dispenser in detail.

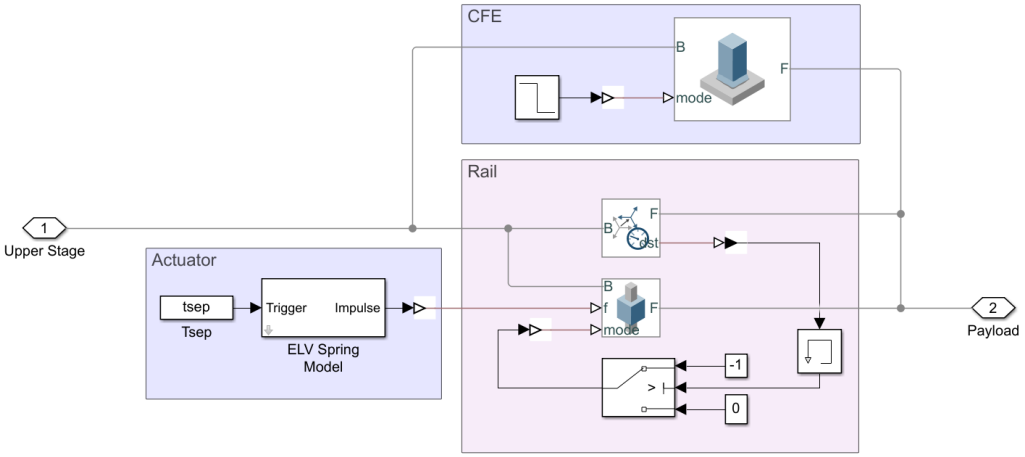

Figure 10: CubeSat Dispenser Implementation

3.5 Mode Management and AOCS System

This subsection briefly describes the implemented Attitude and Orbit Control System (AOCS). The need for Attitude and Orbit control can directly be derived from the tool requirements. It shall be possible to manipulate the upper stage orbit as well as its attitude and angular rates. As input to the AOCS system, the user has to define an AOCS input file containing the AOCS modes and commands as time–vectors. Simscape's *Transform Sensor* is used to measure quantities required within the controllers, such as attitude and angular rates. Mode management and controllers were implemented using standard Simulink blocks while Simscape's *External Force and Torque* block was used to apply thrust and torque computed by the controllers to the upper stage body. Table 2 summarizes the implemented AOCS modes and respective controllers.

3.6 Distance Measurement and Collision Detection

MAST is able to track collisions between all payloads and between the upper stage and the payloads. Two basic methods of collision detection between these bodies have to be differentiated:

- In–simulation collision detection between geometries
- Post–processing collision detection between geometries and collision spheres

During simulation, the distance between the geometries of the separated elements are tracked. In case their distance falls below a certain threshold, a collision is detected and the simulation is stopped; this also triggers an alert that is provided to the user. The results from the in–simulation collision detection are not further used for the post-processing collision detection. The main collision detection within

Table 2: AOCS Modes implemented in MAST

Mode	Description
Attitude Control Modes	
Coast	No forces or torques applied.
Detumbling	Angular rate is damped to zero.
Spin	Angular rate tracks a commanded value.
Attitude control (wrt. ECI)	Upper stage attitude tracks reference quaternion with respect to ECI. When the reference is the unit quaternion $q_{\text{ref}} = [0 \quad 0 \quad 0 \quad 1]^T$, the upper stage body frame aligns to the ECI frame.
Attitude control (wrt. LVLH)	Upper stage attitude tracks reference DCM wrt. LVLH (see 3.1 for information on LVLH frame). When the reference DCM equals the identity matrix, the upper stage body frame aligns to the LVLH frame.
Orbit Control Modes	
Engine Off	No force applied to the upper stage.
Boost	Applies a predefined constant thrust force to the upper stage during a specified time interval.
Delta–v	Changes the upper stage's orbital velocity according to the command.

the MAST tool takes place in post–processing scripts. During simulations, two types of sensors – *Transform Sensors* and *Spatial Contact Force* blocks are used for sensing. Simscape's *Transform Sensor* measures translational and rotational quantities between two physical frames. Within MAST, these frames are placed in the Center of Gravity of the upper stage and payloads. The *Spatial Contact Force* block allows to measure the distance between two bodies' geometries. In order to use Spatial Contract Force block, the export of a body's entire geometry has to be enabled. The implementation of these two sensors with the corresponding measurement method are shown in Figure 11. These measurement outputs are then used within the post-processing scripts for collision detection. Four different types of collisions are distinguished:

- Collision of geometries
- Collision of constant collision spheres
- Collision of increasing collision spheres
- Close fly–by (with a sign change of the relative velocity)

A *Collision Sphere* is a sphere that envelopes a payload's entire geometry. The first three types of collisions are sketched in Figure 12. The idea behind changing the collision detection method is to counteract inaccuracies in the model and numerics. An identifier is assigned to each collision type. In

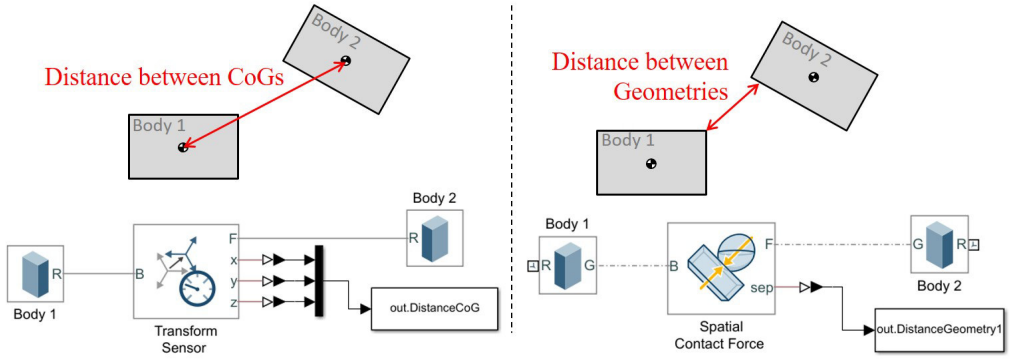

Figure 11: Simscape's *Transform Sensor* (left) and *Spatial Contact Force* block (right) with the corresponding Measurement Method

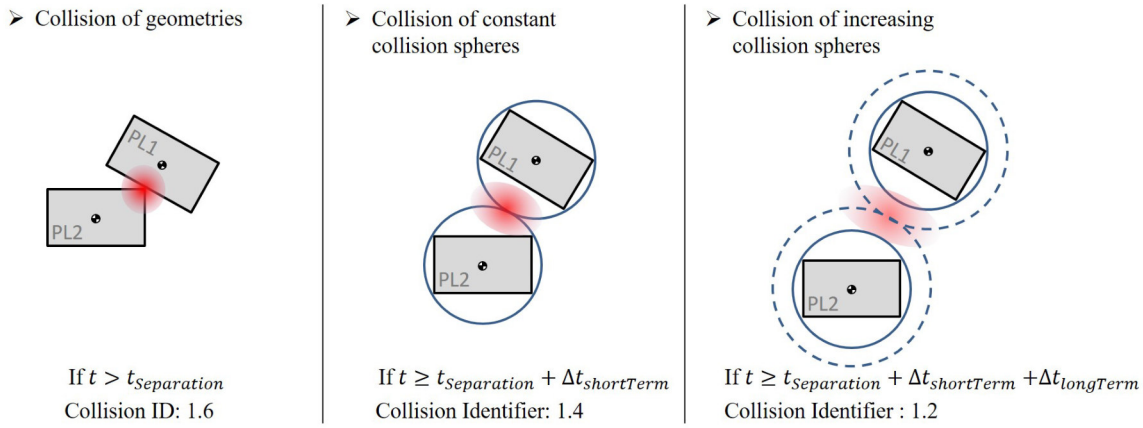

Figure 12: Collision Types distinguished within the Post-Processing Collision Detection

case a possible collision due to a close fly–by (with a sign change in the relative velocity) is detected (without another type of collision being identified), an additional small simulation with the involved bodies' parameters and properties is started with the additional simulation having a much smaller solver step size than the original simulation.

4 TOOL WORKFLOW – *HOW TO USE MAST*

4.1 General Workflow

As already mentioned, MAST is a MATLAB tool with the capability to automatically assemble Simulink and Simscape models in order to be able to simulate separation dynamics and sequences of several payloads deployed from a common upper stage. Additionally to the pre–processing capabilities and automatic assembly options, Monte Carlo simulations and post–processing routines were developed for the MAST tool; among these, a common scenario to analyze is to detect possible collisions between the payloads themselves and between payloads and upper stage ("clearance").
Figure 13 shows the typical workflow when using MAST while Figure 14 shows a sketch of the intended user experience. In a first step, the user has to prepare an input file containing all information required from the mission at hand and regarding the automatic assembly options. A second input file includes AOCS modes concerning mode management, and the attitude and orbit reference commands. In a next step, the automatic assembly script can be executed resulting in the automatic

assembly of the Simulink and Simscape models in accordance with the definitions and requirements from the user input file. The newly assembled model can be executed as standalone model or further used in a Monte Carlo routine. In both cases, Simscape's Mechanics Explorer can be used to visualize the simulation making use of predefined or additional cameras. After the definition of Monte Carlo settings, the automatically assembled model can be used to perform Monte Carlo simulations with perturbed parameters (using normal distribution). After all simulations are finished, possible collisions are detected and plotted by a post–processing script.

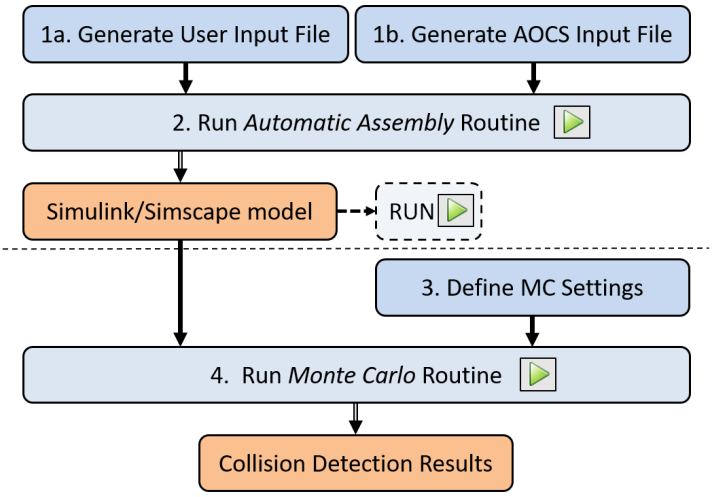

Figure 13: MAST Workflow

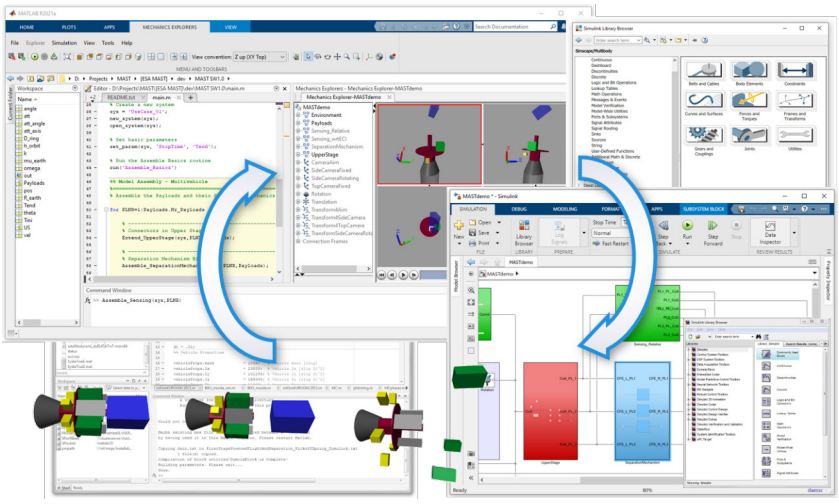

Figure 14: MAST User Interfaces

4.2 Pre–Processing and Automatic Assembly

The automatic assembly aims at simplifying the usage of MAST and at reducing the possibility for errors by manual assembly or modification of the model(s). It is based on a user–input file in which the user defines all mechanical properties and parameters needed for the respective multi–payload separation scenario. The user–input file needs to contain at least the following information in a structured manner:

- **Simulation Properties**: Simulation times, initial conditions, perturbations;
- **Upper Stage Properties**: Mass, CoG position, Moment of Inertia (MoI), drag coefficient, surface area, geometrical properties;
- **Payload Properties**: Mass, MoI, dimensions, CoG position, drag coefficient, surface area – for each payload;
- **Separation System Properties**: Separation time, actuation time, number of actuators, actuator positions, spring force and final stroke – for each payload's separation system;
- **Routing Information**: Payload attachment positions on the upper stage.

The automatic assembly script calls the user input file and assembles the model accordingly. The core of the assembly routine is a *for–loop* with the total number of payloads as the counter. Frequently recurring sub–models such as generic payloads or separation system actuators are defined in a library. For every step of the assembly routine and definition in the user input file, the corresponding Simulink, Simscape or library blocks are added to the basic model and their parameters are set based on the definition of the user input file. The automatic assembly script is mainly based on the following four MATLAB commands which can be used to manipulate models in Simulink: `add_block()`; `add_line()`; `set_param()`; and `get_param()`.
A routing script within the automatic assembly enables the payloads to be correctly and automatically assigned to their attachment point on the upper stage as defined within the routing section of the user–input file. Figure 15 shows the routing section of an exemplary user input file in which the attachment position of all payloads, including several CubeSats stored in CubeSat dispensers are defined. The right side of the Figure shows the visualization of the corresponding upper stage assembly.

```
%% Routing (Payloads to Attachment Positions)

% Enable CubeSat - Dispenser
% [Iso3, Duo1, Duo2, Quad1, Quad2, Quad3, Quad4]
US.CubeSatDispenser.HEX1.Nr = '2';
US.CubeSatDispenser.HEX1.UP_Type = 'Quad3';
US.CubeSatDispenser.HEX1.DWN_Type = 'Quad4';
US.CubeSatDispenser.HEX6.Nr = '1';
US.CubeSatDispenser.HEX6.Type = 'Quad1';


% ROUTING to Payload Attachment Positions
Payloads.Payload_1.Position = 'HEX5';
Payloads.Payload_2.Position = 'HEX1_UP_Quad3_Pos3';
Payloads.Payload_3.Position = 'HEX1_UP_Quad3_Pos2';
Payloads.Payload_4.Position = 'HEX1_UP_Quad3_Pos1';
Payloads.Payload_5.Position = 'HEX1_DWN_Quad4_Pos1';
Payloads.Payload_6.Position = 'HEX1_DWN_Quad4_Pos2';
Payloads.Payload_7.Position = 'HEX1_DWN_Quad4_Pos3';
Payloads.Payload_8.Position = 'HEX1_DWN_Quad4_Pos4';
Payloads.Payload_9.Position = 'HEX6_Quad1_Pos1';
```

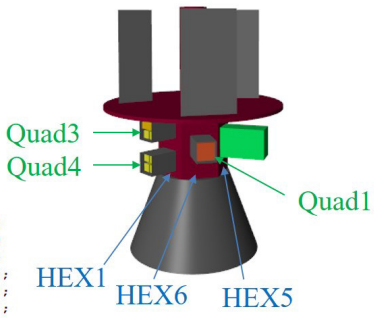

Figure 15: Routing section of the User–Input File (left) and Visualization of corresponding Model (right)

4.3 Post-Processing - Monte Carlo Routine

As depicted in Figure 13 the automatic assembly results in a ready-to-run Simulink/Simscape model. This model can either be used stand-alone or it can be embedded in a Monte Carlo routine. The latter requires the user to name perturbed parameters and specify their standard deviation assuming normal distribution. Table 3 summarizes all properties that can be perturbed for Monte Carlo simulations.
The Monte Carlo script then automatically creates a perturbed parameter set and adjusts the perturbed parameter for each simulation in all elements of the model that depend on the respective parameter. It

Table 3: Variable Parameters for Monte Carlo Simulations

Model	Parameter
Upper Stage	Mass, MoI, PoI, CoG vector
Payloads	Mass, MoI, PoI, CoG vector
Separation System	t_{sep}, t_{act}, final spring stroke L_{f}, spring force F_{k}

further executes all simulations and stores the results after each execution. When all simulations are finished the Monte Carlo script calls the post processing scripts for collision detection (see section 3.6). Possible collisions between the payloads themselves as well as between the upper stage and a payload are detected and results are plotted for all simulations performed.

5 VERIFICATION AND VALIDATION TEST CASE

5.1 Test–Case Setup

The main objective of this test–case is to verify the correct implementation of the automatic assembly script. This includes the model's functional behavior in a complex separation scenario and the correct implementation of the post–processing scripts. The post–processing scripts include the Monte Carlo routine and the collision detection routines. Upper stage and payloads shall represent a hypothetical VEGA SSMS mission on a sun–synchronous, circular orbit. All non–zero initial conditions are summarized in Table 4.

Table 4: Test–Case (non–zero) Initial Conditions

Parameter	Value	Units
Orbit height	800	km
Inclination	98.696	deg
Angular Rate	$[5 \quad 0.1 \quad 0.2]^{\mathrm{T}}$	deg/s

For the test–case the upper stage shall be composed of the lower Hexagonal module, the Plate and central Column modules plus four Tower modules. Payloads should be attached as summarized in Table 5.
After an initial coast and detumbling phase the upper stage's attitude shall be controlled with respect to the LVLH frame. To avoid collisions, an avoidance maneuver shall be performed after the lateral separation from each upper stage module. These avoidance maneuvers were implemented by rotating the upper stage by $60\ \mathrm{deg}$ around the LVLH y–axis and firing the main engine for a short–time boost. After separation of the first payload the orbit of the upper stage assembly shall be slightly changed by performing a delta–v maneuver. Figure 16 summarizes the separation sequence as well as the AOCS modes of the test case.
When the model is assembled Monte Carlo simulations with perturbed parameters shall be performed taking into account that system properties are defined with certain tolerance levels around their nominal values.

Table 5: Payload Allocation for Test Case

Upper Stage Module	Attached Payloads	Separation Type and Time
Column Module	1 Mini-Satellite	Longitudinal, single separation at 1300 s
Tower Modules	4 Nano-satellites	Longitudinal, sequential separation at $t_{sep} = [2700, 2750, 2800, 2850]$ s
Plate Module	4 Micro-satellites	Longitudinal, sequential separation at $t_{sep} = [3500, 3550, 3600, 3650]$ s
Hexagonal Module	3 Nano-Satellites	Lateral, simultaneous separation at $t_{sep} = 4500$ s
	3 CubeSat dispensers (total of 8 CubeSats)	Lateral, sequential separation between $t = 4700$ and $t = 5450$ s

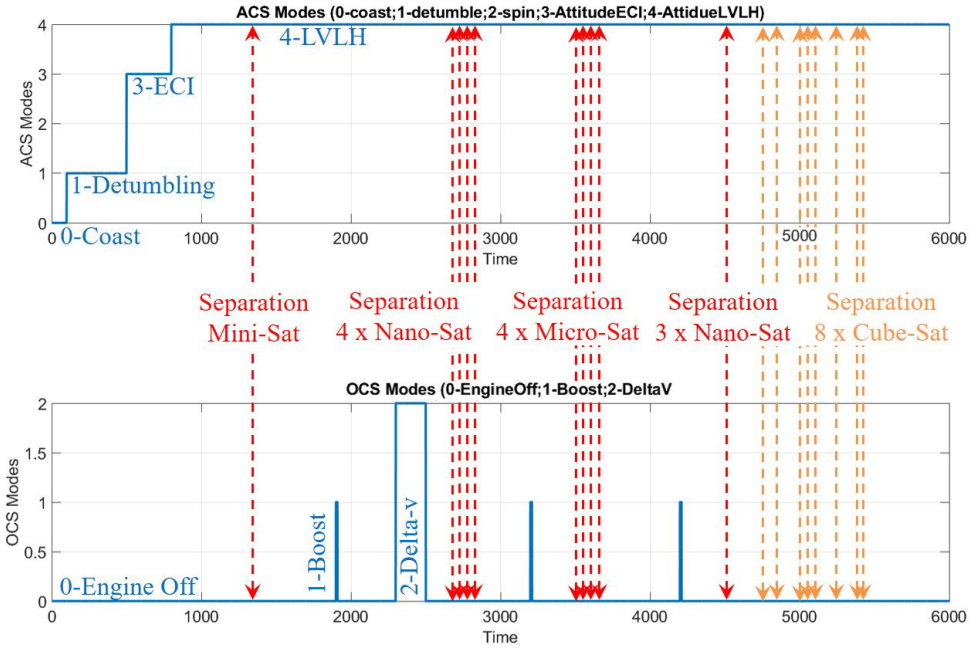

Figure 16: Test Case - Separation Sequence and AOCS Modes

5.2 Use–Case Model and Results

Figure 17 shows the Simscape/Simulink model resulting from the automatic assembly on the left with the corresponding visualization including all payloads on the right. Based on the user input file, the model was correctly assembled by the automatic assembly routine and the AOCS implementation worked as expected following all commands without getting unstable.
Separation of all payloads was also triggered correctly with the separation sequence shown in Figure 18 and no collisions were detected due to the long time intervals between separations.
Regarding the results of the test–case, correct implementation of all main functionalities according to user requirements and functional requirements could be verified. The model was correctly assembled

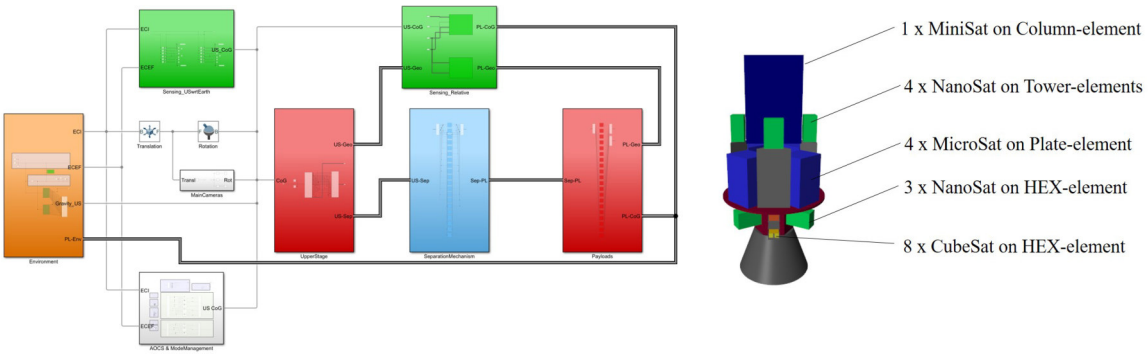

Figure 17: Test Case - SimScape Model (left) and Visualization (right)

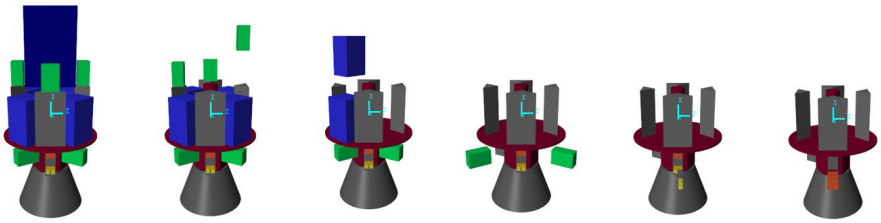

Figure 18: Test Case - Visualization of Payload Separation Sequence

by the automatic assembly routine based on the definitions in the user input file. Payloads were attached at the right positions, and the separation system characteristics were correctly assigned. The assembled model could be directly run without further effort. Visualization was enabled using Simscape's *Mechanic Explorer*. All payloads were separated at the defined times with the AOCS system keeping the upper stage controlled and stable. Perturbed parameter sets for Monte Carlo simulations were also generated, and perturbed parameters were manipulated within the model for each simulation and all simulations were performed. Due to high time intervals between payload separations, no collisions could be detected. Unit tests in the scope of a Verification and Validation (V&V) test plan showed the collision detection capability of the post-processing routines. In the next subsection one of these unit-tests based on the test-case configuration with a deliberately caused collision is shown.

5.3 Collision Test–Case

To evaluate a collision in the test case scenario, a collision of the mini–satellite (mounted on the central Column module) and a micro–satellite attached to the Plate module was intentionally caused by manipulating the separation times and separation system characteristics. Figure 19 shows the result of this test case. The plot shows the distance between the two payloads (top right) as well as the corresponding relative velocity (middle right) and collision flag (bottom right). The mini–satellite is separated at $t = 1300$ s causing the distance to increase until separation of the micro–satellite, which in turn experiences a higher acceleration due to modified mass and separation system characteristics. At $t = 1325$ s, the short time period after separation is reached and the collision spheres method is introduced (see chapter 3.6, especially Figure 12). At approximately $t = 1356$ s, the contact between collision spheres is detected resulting in the collision flag being raised to `1.4`. In the top plot of Figure 19 this can be seen by the relevant distance (red line) crossing the collision condition (black line; equals the sum of both collision shere radii). Shortly afterwards, the geometries touch and the collision flag is set to `1.6` resulting in the simulation to be stopped (see blue dashed line reaching zero

distance). This example clearly showcases one of the capabilities of MAST, which is being able to detect collisions in a meaningful way. The tool shall be used therefore in properly assessing mission separation sequences and eventually having an impact on design and configuration choices for the next generation of satellite deployment architectures.

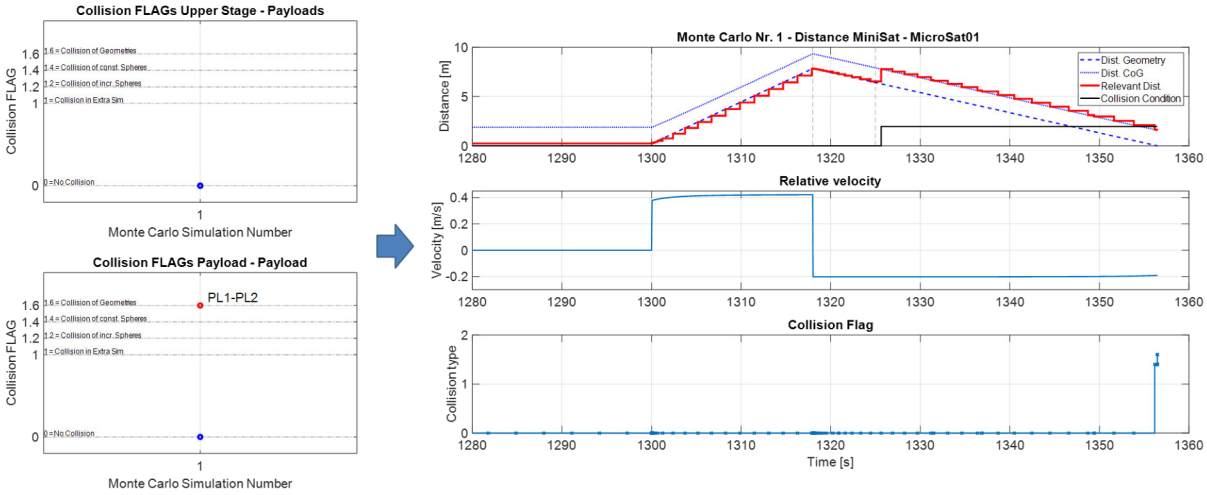

Figure 19: Collision Detection Results

6 CONCLUSIONS AND OUTLOOK

The objective of this paper was to present the preliminary design and development of MAST, a *Multi-vehicle Analysis and Separation Tool*. The main purpose of the development of this tool is to simulate multi-payload (MPL) separation sequences and to provide the means to verify and to validate such mission design separation sequences and requirements. The simulator is developed and implemented using the MathWorks Simscape Multibody modeling toolbox which allows to physically model the composite of the scenario consisting of the launch vehicle upper stage, the payload adapters, separation mechanisms, and the payloads themselves, while also considering separation and orbital dynamics. This is done within the simulation environment of MATLAB & Simulink, which in combination with Simscape, allows these physical–based, acausal modeling features to be combined and integrated for example with other disciplines like AOCS and GN&C. A use case representative of a realistic mission scenario for a hypothetical VEGA SSMS flight was defined and successfully tested using the MAST tool. This scenario implementation was accompanied with some V&V unit tests and Monte-Carlo routines. The results were able to verify not only that the required separation requirements are fulfilled and that there are no collisions between the separated elements during separation, but also to verify that collisions does not happen among the satellites after a few orbits.
For future implementations and improvements of the tool, we conclude the paper with the following outlook: (a) modification of the tool using new features from the latest Simscape release (R2021a) which include better collision models and interfaces between the foundational domain (1D) and the multibody domain; (b) improvements and modifications in the Environment model (common gravity model, for instance); (c) increasing the fidelity of actuator dynamics and their impact at separation time; (d) implementation of novel, more complex CubeSat dispensers; (e) integration of Monte Carlo routines for use with worst case analysis tools.

ACKNOWLEDGMENTS

This work was financed by the ESA study *MAST: Multivehicle Analysis and Separation Tool*, contract No. 4000128760/19/NL/CRS/hh. The authors would like to thank and acknowledge useful discussions and ideas with/from Pedro Simplício (ESA/ESTEC) and Tobias Posielek (DLR-SR). The view expressed in this paper can in no way be taken to reflect the official opinion of the European Space Agency.

REFERENCES

[1] F. Caramelli, F. Battie, A. Scaccia, A. De-Lillis, S. Corbo, A. Dalloneau, A. Fontana, A. Cramarossa, and A. Gabrielli, "Optimised flight preparation process for the first Vega Ride-Share mission," in *Proceedings of the AIAA/USU Annual Conference on Small Satellites*, 2018.

[2] F. Caramelli, F. Battie, A. Scaccia, S. Corbo, A. Dalloneau, F. Fabiani, A. Fontana, M. Mariani, P. Guerrieri, A. Cramarossa, and A. Gabrielli, "The First Vega Ride-Share Mission Flight," in *Proceedings of the AIAA/USU Annual Conference on Small Satellites*, 2019.

[3] J. P. Decker, "Experimental Aerodynamics and Analysis of the Stage Separation of Reusable Launch Vehicles," NASA, Tech. Rep. SP-148, 1967.

[4] J. P. Decker and J. Gera, "An Exploratory Study of Parallel-Stage Separation of Reusable Launch Vehicles," NASA, Tech. Rep. TN D-4765, October 1968.

[5] J. P. Decker and A. W. Wilhite, "Technology and Methodology of Separating Two Similar Size Aerospace Vehicles within the Atmosphere," in *AIAA 13th Aerospace Sciences Meeting*, no. 1975-29, January 1975.

[6] G. L. Bauer, D. E. Cornick, and R. Stevenson, "Capabilities and Applications of the Program to Optimize Simulated Trajectories (POST)," NASA, Tech. Rep. CR-2770, 1977.

[7] "User's Manual for TREETOPS, a Control System Simulation for Structures With a Tree Topology," NASA, Tech. Rep. Contract NAS-36287, Marshall Space Flight Center, April 1990.

[8] R. M. Yates, "TREETOPS Structural Dynamics Controls Simulation System Upgrade," NASA, Tech. Rep. Contract NAS8-40194, Marshall Space Flight Center, August 1996.

[9] J. S. Orr, "Space Launch System Flight Control," in *Aerospace Control and Guidance Systems Committee (ACGSC) Meeting 110*, October 2012.

[10] K. J. Murphy, P. G. Buning, B. N. Pamadi, W. I. Scallion, and K. M. Jones, "Overview of Transonic to Hypersonic Stage SeparationTool Development for Multi-Stage-To-Orbit Concepts," in *AIAA Aerodynamic Measurement Technology and Ground Testing Conference*, 2004.

[11] B. N. Pamadi, T. A. Neirynck, P. F. Covell, N. J. Hotchko, and D. M. Bose, "Simulation and Analyses of Staging Maneuvers of Next Generation Reusable Launch Vehicles," in *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, 2004.

[12] B. N. Pamadi, N. J. Hotchko, S. J. A., P. F. Covell, and P. V. Tartabini, "Simulation and Analyses of Multi-Body Separation in Launch Vehicle Staging Environment," in *AIAA/AHI Space Planes and Hypersonic Systems and Technologies Conference*, 2004.

[13] B. N. Pamadi, T. A. Neirynck, N. J. Hotchko, S. W. I., K. J. Murphy, and P. F. Covell, "Simulation and Analyses of Stage Separation of Two-Stage Reusable Launch Vehicles," *Journal of Spacecraft and Rockets*, vol. 44, no. 1, pp. 66–80, January 2007.

[14] M. Toniolo, P. Tartabini, B. Pamadi, and N. Hotchko, "Constraint force equation methodology for modeling multi-body stage separation dynamics," in *AIAA Aerospace Sciences Meeting and Exhibit*, 2008.

[15] P. V. Tartabini, C. M. Roithmayr, M. D. Toniolo, C. D. Karlgaard, and B. N. Pamadi, "Modeling multibody stage separation dynamics using constraint force equation methodology," *Journal of Spacecraft and Rockets*, vol. 48, no. 4, pp. 573–583, 2011.

[16] B. Pamadi, P. Tartabini, M. Toniolo, C. Roithmayr, C. Karlgaard, and J. A. Samareh, "Application of Constraint Force Equation Methodology for Launch Vehicle Stage Separation," *Journal of Spacecraft and Rockets*, vol. 50, no. 1, pp. 191–205, January-February 2013.

[17] R. Franco M., "Enhanced Dynamics and Control Analysis Package (DCAP)," in *Proceedings of ESA Workshop on Spacecraft Guidance, Navigation and Control*, September 1992.

[18] R. Franco M., L. Dumontel, S. Portigliotti, and R. Venugopal, "The Dynamics and Control Analysis Package (DCAP) - a Versatile Tool for Satellite Control," ESA Bulletin, Tech. Rep. 87, August 1996.

[19] S. Portigliotti, M. Dumontela, G. Baldesi, and D. Sciacovelli, *DCAP (Dynamics and Control Analysis Package) - an Effective Tool for Modelling and Simulating of Coupled Controlled Rigid Flexible Structure in Space Environment*, 2004.

[20] G. Baldesi, D. Sciacovelli, and A. Thirkettle, "Simulation Tool for Generic Launcher Flight Dynamics-Control Interaction Analysis," in *Proceedings of the 6th International Symposium on Launcher Technologies: Flight Environment Control for Future and Operational Launchers*, November 2006.

[21] G. Baldesi and M. Toso, "ESA Launcher Flight Dynamics Simulator used for System and Subsystem Level Analyses," in *Proceedings of the 11th Intl. Workshop on Simulation & EGSE facilities for Space Programmes (SESP 2010)*, September 2010.

[22] G. Baldesi and M. Toso, "European Space Agency's Launcher Multibody Dynamics Simulator used for System and Subsystem Level Analyses," *CEAS Space Journal*, vol. 3, pp. 27–48, 2012.

[23] P. Simplicio and S. Bennani, "Worst-case Launch Vehicle Stage Separation Analysis," 04 2015, 3rd CEAS Specialist Conference on Guidance, Navigation and Control, CEAS, 2015.

[24] H. Elmqvist, S. E. Mattsson, and M. Otter, "Modelica - An International Effort to Design an Object-Oriented Modeling Language," in *Proceedings of the SCSC'98 - Summer Computer Simulation Conference*, 1998.

[25] S. E. Mattson, H. Elmqvist, and M. Otter, "Physical System Modeling with Modelica," *Control Engineering Practice*, vol. 6, pp. 501–510, 1998.

[26] P. Acquatella B. and M. J. Reiner, "Modelica Stage Separation Dynamics Modeling for End-to-End Launch Vehicle Trajectory Simulations," in *Proceedings of the 10th International Modelica Conference*, 2014, pp. 589–598.

[27] P. Acquatella B., "Launch Vehicle Multibody Dynamics Modeling Framework for Preliminary Design Studies," in *6th International Conference on Astrodynamics Tools and Techniques, ICATT*, 2016.

[28] L. E. Briese, K. Schnepper, and P. Acquatella B., "Advanced Modeling and Trajectory Optimization Framework for Reusable Launch Vehicles," in *Proceedings of the 2018 IEEE Aerospace Conference*, 2018.

[29] L. E. Briese, P. Acquatella B, and K. Schnepper, "Multidisciplinary modeling and simulation framework for launch vehicle system dynamics and control," *Acta Astronautica*, vol. 170, pp. 652–664, 2020.

[30] P. Acquatella B., L. E. Briese, and K. Schnepper, "Guidance command generation and nonlinear dynamic inversion control for reusable launch vehicles," *Acta Astronautica*, vol. 174, pp. 334–346, 2020.

[31] J. Diegelman, "Modeling Spacecraft Separation Dynamics in Julia," in *SIAM Conference on Computational Science and Engineering*, 2021.

[32] S. K. J. Bezanson, A. Edelman and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, vol. 59, no. 1, p. 65–98, 2017.

[33] MathWorks. Simscape Multibody Release Notes. [Online]. Available: https://mathworks.com/help/physmod/sm/release-notes.html

[34] MathWorks. (2020) Simscape Documentation. [Online]. Available: https://mathworks.com/help/physmod/simscape/

[35] MathWorks. (2020) Simscape Multibody Documentation. [Online]. Available: https://mathworks.com/help/physmod/sm/

[36] F. L. Markley and J. L. Crassidis, *Fundamentals of Spacecraft Attitude Determination and Control*. Springer, Space Technology Library, 2014.

[37] A. E. Hedin, "Msis-86 thermospheric model," *Journal of Geophysical Research: Space Physics*, vol. 92, no. A5, pp. 4649–4662, 1987.

[38] Astos Solutions. Astos User Manual. [Online]. Available: https://www.astos.de/products/astos

[39] J. Baumgarte, "Stabilization of Constraints and Integrals of Motion in Dynamical Systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 1, no. 1, pp. 1–16, 1972.

[40] U. M. Ascher, H. Chin, L. R. Petzold, and S. Reich, "Stabilization of Constrained Mechanical Systems with DAEs and Invariant Manifolds," *Journal of Mechanics of Structures and Machines*, vol. 23, pp. 135–157, 1994.