Advanced GNC-oriented Modeling and Simulation of Vertical Landing Vehicles with Fuel Slosh Dynamics

Stefano Farì[a,*], David Seelbinder[a] and Stephan Theil[a]

[a]German Aerospace Center (DLR) - Institute of Space Systems, Navigation and Control Systems Department, Robert Hooke Straße 7, Bremen, Germany

ARTICLE INFO

Keywords:
Slosh dynamics
Modelica
Object-oriented modeling
Simulation
GNC
Vertical landing vehicle
Lunar lander
Reusable launch vehicle

ABSTRACT

This paper introduces a flexible framework to enable high-fidelity simulations of the fuel slosh dynamics for verification of the Guidance, Navigation and Control (GNC) algorithms. The sloshing phenomenon must be tackled during the control system design phase and the GNC software run and validated within appropriate simulation frameworks. Equivalent mechanical models can well approximate sloshing under specific assumptions, allowing the derivation of the multibody vehicle's equations of motion; however, while suitable for analysis and control synthesis purposes, this approach presents several limitations for implementation as a simulation model. In this work the multibody plant embedding slosh dynamics is modeled by means of the DLR's Vertical Landing Vehicles Library (VLVLib) written using the object-oriented Modelica modeling language. The advantages of using Modelica are explored throughout the paper. Given that control synthesis, analyses and simulation campaigns are usually performed in the Matlab/Simulink environment, a core point of the proposed solution is to achieve a good synergy within the latter and the Modelica plant model with the least readaptation burden for the developer. The strategies to achieve a full integration and verification tool chain are illustrated. The potential of this approach is demonstrated via two different test cases: a lunar landing scenario of the ALINA spacecraft developed by PTS, and CALLISTO reusable rocket demonstrator.

1. Introduction

Propellant slosh dynamics within space vehicles is a deeply studied phenomenon and a well known factor to consider in the Guidance, Navigation and Control (GNC) system design since the early days of large liquid-fuel rockets [1]. If the propellant contained in a tank has a free surface, parasitic interactions with its structure may arise as an effect of the liquid movements. When the ratio between fuel and overall vehicle mass becomes relatively high, these adverse dynamics cannot be neglected and should be appropriately considered to avoid critical performance degradation or instability, especially in conjunction with strict trajectory tracking requirements. Applications like Vertical Take-off Vertical Landing (VTVL) vehicles, such as reusable rockets, or planetary landing systems — both case studies of this work— are nowadays demanding for better performance, possibly leading to GNC algorithms with growing complexity. Thus, they must be verified appropriately in closed-loop high-fidelity simulators. Considering the presence of advanced modeling tools and the enhanced processors computational power, new frameworks able to capture sloshing from GNC perspective must be researched and implemented [2]. Similarly important is also the need for a responsive modeling methodology, which can quickly adapt to requirement changes during the project development process (or between different projects), in order to speed up timelines and reduce the associated costs.

Simulating the fluid motion as a whole is a very complex task: computational fluid dynamics and finite element methods can be considered to solve the fluid-structure interaction problem numerically, but they are not suitable for use within the GNC algorithms development, because of their strong computational requirement and the complexity of the required simulation setup. Nevertheless, liquid sloshing can be studied by separating the main effects they produce onto the rigid body dynamics under certain boundary conditions. For the craft types investigated in this work, slosh dynamics assume relevance in the accelerating phases, such as delta-V maneuvers and ascent, but especially during descent and landing. In these scenarios, the lateral dynamics (simply shown in Figure 1) can be severely affected and result in critical deviations from the nominal trajectory as a consequence of disturbed thrust pointing [3]. This paper discusses how is it possible to embed the main sloshing effects in a simulation setup meant to verify GNC algorithms performance for Vertical Landing (VL) vehicles.

1.1. State of the art

It has been demonstrated that pendulum and mass-spring models can approximate the fluid dynamics of interest [3, 5, 6]. Relying on this approach, many studies on dynamics and control of space vehicles with fuel slosh dynamics have been conducted, assuming that the tank external excitation does not coincide with the natural frequencies of the sloshing.

*Corresponding author
stefano.fari@dlr.de (S. Farì); david.seelbinder@dlr.de (D. Seelbinder); stephan.theil@dlr.de (S. Theil)
ORCID(s): 0000-0001-5595-6905 (S. Farì); 0000-0003-4080-3169 (D. Seelbinder); 0000-0002-5346-8091 (S. Theil)

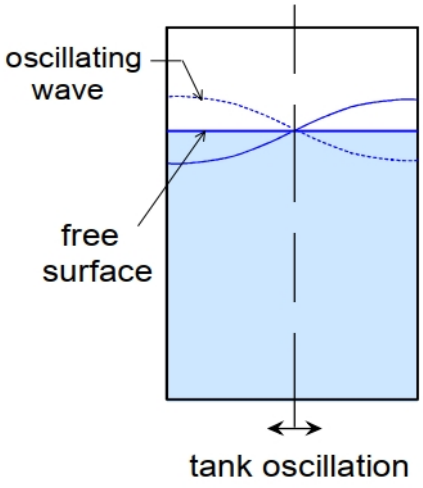

Figure 1: Lateral sloshing representation (from [4]).

Figure 2: ALINA spacecraft render, courtesy of PTS GmbH.

The equivalent model characteristic parameters correspond to one specific mode of liquid oscillation at a specific tank filling level, and depend on the liquid properties and tank geometry [4]. They can be obtained with CFD methods [7] or analytically for simple tank shapes [8], and then validated experimentally [9]. The vehicle motion is therefore represented by a multibody model, where a rigid body representing the dry vehicle is separated from the pendulum or spring-mass model describing fluid slosh dynamics, and from a rigid body representing the nonsloshing part of the liquid in its tank. As with any continuum dynamics, an infinite number of sloshing modes exists, but one to three are largely sufficient to characterize the main disturbances produced on the vehicle; higher sloshing modes have a small impact on the whole dynamics. This solution enables the computation of the vehicle's equations of motion, so to form the basis for system stability analyses and controller synthesis: for instance, in [10] it is used to investigate the stability properties of an accelerated spinning spacecraft. In [11] the effect of slosh-vehicle coupling on vehicle stability for a typical launch vehicle in planar atmospheric flight is studied and bifurcation analysis performed to show vehicle global stability properties in a two-parameters space. Reyhanoglu performed an extensive research on sloshing modeling and control: for example, in [12, 13, 14, 15] the controller design problems of a planar spacecraft motion with one tank and sloshing mode are approached; in [16] and [17] these methods are extended to a spacecraft with two sloshing modes; in [18] two sloshing modes with fuel draining (thus time-variant sloshing parameters) are considered; in [19] and [20] the equivalent modeling is used for control design in a three-dimensional space for one and two sloshing modes respectively. Other examples may include the development of a full-state feedback employing a reduced-order observer for the estimation of the slosh states to attenuate the sloshing modes [21]; adaptive [22] and fuzzy [23] control approaches have been researched as well. Baozeng *et al.* performed several studies regarding liquid-filled spacecraft maneuvers; in [24] the spherical pendulum employed to simulate the sloshing is controlled using a model dynamic inversion and input shaping hybrid approach. In [25] a pulsating ball model is used to simulate large sloshing amplitudes during a spacecraft re-orientation. For spherical tanks, these large amplitudes are also validated with respect to the analytical solution via a mesh moving strategy [26].

1.2. Problem statement

Existing research outcomes usually start by obtaining the vehicle equations of motion under some simplifications. The most recurrent ones are:

(i) the parameters for the equivalent models correctly preserve the actual mass, center of mass and moment of inertia of the fluid;

(ii) the vehicle motion is modeled on a plane (2D) only;

(iii) only one sloshing mode is accounted for;

(iv) the spacecraft has one propellant tank only;

(v) or, sloshing model parameters are not considered time-variant as consequence of fuel draining at commanded accelerations.

In general, operating in this way pays out for analysis or control synthesis purposes, but, to the best of the authors knowledge, no work has specifically addressed the implementation of simulation models able to accurately represent a vehicle affected with slosh dynamics that does not subordinate to any of the aforementioned points.

As sloshing acts as an internal force, the vehicle equations can either be written by adding algebraic constraints and obtaining a so-called Differential-Algebraic Equations (DAE) system, in order to link the oscillating and the "rigid" parts, or after removing these constraints via algebraic manipulation to obtaining a set of Ordinary Differential Equations (ODE). For the implementation of ODE systems, Simulink, a causal modeling environment, is the de-facto standard due to its inherent suitability for control laws

design and simulation. On the same software, validation and verification campaigns are likewise usually performed, before moving to Hardware-In-the-Loop (HIL) tests and actual experiments. For these reasons, Simulink will be considered here the software on which to solidly found the approach proposed in this paper.

Despite its advantages, Simulink presents some general drawbacks: (i) it may require a tedious and error-prone analytic work to obtain the whole set of equations to be implemented, which drastically grows as the vehicle configuration complexity increases; (ii) as known for causal modeling environments, it offers very poor reusability of elementary components, thus inducing a major rework of the whole set of underlining equations as soon as the vehicle configuration changes; (iii) similarly, it is not easily expandable as soon as new dynamics for more accurate simulations are to be added for study, as equations must be reworked as well. These statements are especially justified for vehicles with several tanks. As an example, the underlying multibody model for a spacecraft with eight tanks, like the one in Figure 2, would include at least: a body representing the dry craft; eight bodies representing the nonsloshing liquid per each tank; one pendulum per each tank representing for sloshing - but more are to be added if two or three sloshing modes shall be simulated.

Therefore, acausal environments can be used to overcome these limitations and give additional modeling flexibility. For maximum compatibility with Simulink environment, one can consider Mathworks SimMechanics payware add-on package. This scenario has been explored in [27] and [28]. Even in this latter way, some issues exist: above all, SimMechanics hides the models implementation to the user. This compromises flexibility, adaptability and, to some extent, comparisons with other models. Furthermore, depending on the available software release, the developer is bounded to the provided library blocks, which may not cover all the very specific modeling needs discussed before.

These strong arguments lead to the solution adopted here consisting in the use of Modelica modeling language [29], a free acausal object-oriented modeling language for large system descriptions. It enables a clear representation of physical meaning within an object-oriented structure; furthermore, its modularization and component reuse features bring to a greater model adaptiveness against changes. A free Modelica Standard Library (MSL) exists [30] and enables the creation of multibody models [31], entailing several benefits for our objectives, such as: (i) coordinate frames can be simply included in the model in terms of connectors, describing kinematic transformations from one coordinate system to another; (ii) easy extendability for creating custom classes; (iii) specific Modelica constructs are available to deal with the modeling of physical fields and environmental quantities; (iv) sensors and actuators can also be eventually represented in the Modelica paradigm; (v) packages of data sheets for each class can be constructed and components easily modified within each vehicle model, using Modelica's advanced features [29]. These advantages have been exploited, for example, in [32, 33, 34, 35, 36], where similar a methodologies are described. For similar reasons, several libraries have been created, like the DLR Flight Dynamics Library [37] or the Space Flight Dynamic library [38]. The language, therefore, can well suit the creation of a high-fidelity model of vehicles with slosh dynamics using mechanical equivalents.

1.3. Objectives

In this work, a new framework for GNC algorithms verification in presence of a vehicle affected by slosh dynamics is presented. Simulink environment is used for the structuring the whole simulation setup where control software is also included, whereas Dymola [39], a Modelica IDE[1], is exploited to produce the vehicle model. This choice grants extra flexibility to the modeler, that can address multiple vehicle's configuration by just editing appropriate flags or parameters and successively re-exporting the new model. Additional value is the possibility of "disabling" the sloshing motion, thus inhibiting the slosh motion whenever the vehicle is not experiencing an acceleration anymore, like, for instance, during reusable rocket boostback maneuvers. The Modelica Vertical Landing Vehicles Library (VLVLib) [40] has been implemented to support the creation of the specific vehicle models subject of this study. Created with Dymola software, it allows a quick development of ad-hoc spacecraft models, which can be thereafter exported and embedded in the full Simulink simulator [41]. Two study-cases are presented: a Moon landing [42, 43] and a full end-to-end mission of CALLISTO, a reusable rocket demonstrator [44]. The second scenario is shortly included, to prove the capability of the model to inhibit the effect of sloshing dynamics during non-accelerating phases in end-to-end simulations.

In the following Section, the requirements and the approach underlining this research are explained, and the sloshing phenomenon modeling addressed. In Section 3 the VLVLib is introduced. In Section 4, the Dymola-Simulink integration strategy is explained, while in Section 5 the simulation results are shown. Section 6 concludes the paper.

2. Modeling approach for a VTVL vehicle

The development requirements for sloshing modeling are here recalled:

1. the Modelica vehicle model shall address the sloshing phenomenon without altering the fluids MCI static properties;
2. the model model shall not make any planar motion assumption;
3. fuel draining effect shall be considered;
4. the possibility of including up to three sloshing mode per tank shall be given;

[1] Integrated Development Environment

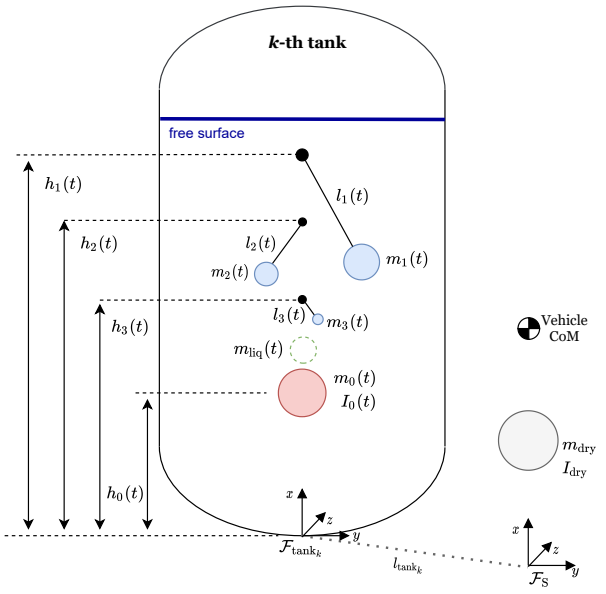

Figure 3: Representation of the employed pendulum equivalent model.

5. the possibility to stop the sloshing motion during simulation under non-accelerated phases shall be given;

6. GNC algorithms, environment properties (e.g. gravity), actuators and sensors modeling, and any simulation infrastructure shall be kept within Simulink.

The last requirement is very important because it avoids unnecessary duplication of functions within the Modelica model. For example, it is essential to retain any already existing and verified models for the gravity gradient, as well as advanced sensor and actuator dynamics, or any conditioning functions. Furthermore, already structured and functioning software for running Monte-Carlo campaigns and advanced analyses can be retained without extra effort. These aspects will become clearer in Section 2.2.

2.1. Sloshing equivalent mechanical models

There are two possible equivalent mechanical models that can properly represent the lateral slosh dynamics for small liquid oscillations: a pendulum model and a spring-mass model. While being both appropriate, the main difference between the two lies in the way the required system natural frequency is obtained, expressed as $\omega = \sqrt{a_x/l_i}$, where $a_x \geq 0$ is the tank longitudinal non-gravitational acceleration and l_i is the arm of the i-th pendulum. In a spring-mass model the stiffness of the spring must be varied as function of a_x in order to achieve the desired natural frequency, whereas in a pendulum model this is behavior is implicitly present already. For this reason, the pendulum model is used. This model is usually valid in the neighborhood of its stable equilibrium; it is possible to rule that a good controller must be able to keep pendulum oscillations bounded within ± 15 degrees. It also operates a distinction between the sloshing and the nonsloshing parts of the liquid. Therefore, the whole liquid mass m_{liq} is split between the single pendulum masses m_i, with $i \in [1, 2, 3]$ being the sloshing mode, and the mass m_0 representing the idle liquid.

Figure 3 illustrates the model together with the main reference frames. Denoted with h_0 and h_i are the nonsloshing body and the i-th pendulum hinge height with respect to the bottom of the tank, respectively; m_i and l_i are the mass and arm of the i-th pendulum. The set of parameters m_0, I_0, h_i, l_i and m_i are provided with respect to discrete tank filling levels and have to be interpolated during the simulation. The frame $\mathcal{F}_{\text{S}}$ is a vehicle-structure-fixed frame conveniently chosen, whereas $\mathcal{F}_{\text{tank}_k}$ is a frame placed at the bottom point of the k-th tank, with $k = 1, 2, ..., K$. Regarding the moment of inertia, no distinction is made between the idle and moving liquid, therefore the whole liquid inertia in a tank I_0 is considered and placed together with the nonsloshing mass m_0. The gray body in Figure 3 models the dry vehicle: the mass and moment of inertia has been labeled with m_{dry} and I_{dry}. Lastly, the overall vehicle center of mass is depicted; its position is influenced by the idle masses distribution, but also by the pendulum motion.

Please note that, although not shown, the position of the dry body and its inertia are allowed to change as consequence of a vehicle overall mass distribution change. This can happen for any actuator deployment, such as fins or landing devices.

To ensure the model validity, the static properties of the liquid must be preserved at every time instant for every k-th tank. Hence, the sum of all the nonsloshing and the pendulum masses must be the same as the whole liquid mass:

$$m_0 + \sum_i m_i = m_{\text{liq}}. \tag{1}$$

Secondly, the center of mass of the liquid in the unperturbed tank must be preserved:

$$m_0 h_0 + \sum_i m_i (h_i - l_i) = m_{\text{liq}} h_{\text{liq}}, \tag{2}$$

where h_{liq} is the height of the total mass m_{liq} over the $\mathcal{F}_{\text{tank}_k}$ origin, hence the static liquid center of mass height.

Figure 3 shows only a planar representation, though each modeled pendulum is enabled to move in three dimensions. Implementation details will be given in Section 3.1.2.

Inclusion of damping

During the slosh dynamics, some energy is dissipated each cycle, meaning that the liquid oscillations are damped. In this paragraph, the model is augmented with a viscous damper.

The damping ratio γ can be determined from the liquid properties and the tank shape via advanced fluid dynamics simulations. Otherwise, Dodge [4] addresses the damping coefficient computation in a simplified form, although more sophisticated shapes can be accounted for with different computations or by performing experimental identification. As an example, we consider here a cylindrical tank shape, assuming that its height predominates over its diameter. The

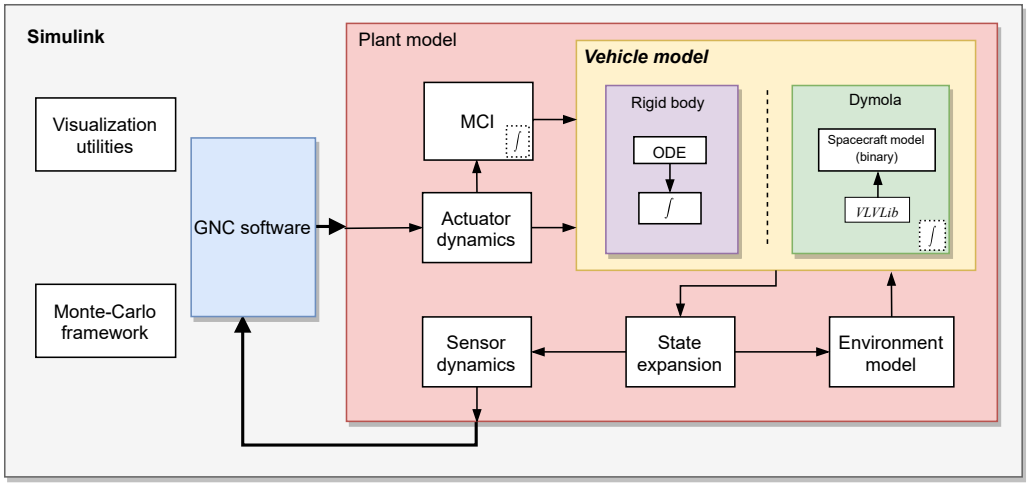

Figure 4: Framework functional scheme.

damping ratio is expressed as

$$\gamma = \sqrt{0.79}\mathrm{Re}_1, \qquad \text{with} \qquad \mathrm{Re}_1 = \frac{\nu}{\sqrt{gL^3}}. \tag{3}$$

In the last expression, Re_1 is the inverse Reynolds number, g the gravitational acceleration, ν the liquid viscosity, and L is the characteristic tank length, here corresponding to its radius.

Introducing damping to a pendulum means adding a linear rotational damper with coefficient C_θ at its hinge point. To compute C_θ, the simple pendulum planar equations of motion are used. With the Lagrange approach, in absence of external forces, one can write for the free-motion of the pendulum:

$$ml^2\ddot{\theta} + C_\theta\dot{\theta} + mla_x \sin\theta = 0. \tag{4}$$

The free coordinate of the pendulum system is just the pendulum angle θ. Variable l is adopted for the pendulum arm length (the pendulum index is omitted as well). Writing C_θ as funcion of ω, it follows that:

$$C_\theta = 2\gamma ml^2\omega = 2\gamma ml^2\sqrt{a_x/l}. \tag{5}$$

Note how damping is time-varying as function of a_x. The unit of C_θ is $[\mathrm{N\,m\,s}] = [\mathrm{kgm}^2/\mathrm{s}]$. The same damping action is exerted in the off-plane as well.

2.2. General framework structure

Figure 4 shows the idea behind the simulation framework used in this paper and applied to the test cases. The Simulink simulator runs using fixed-step *ODE4* integration method and is functionally separated in three different domains: the GNC (in blue), the plant (in red), and the utility functions (in white).

The GNC algorithms are retained in appropriate functions running at a specific frequency dictated by the GNC requirements. The control signals are generated based on the sensors and used to command the actuators, both functionally part of the plant domain where they are modeled in detail.

In the *Actuator dynamics* block, the fuel consumption is computed and propagated to the *MCI* (Mass-Centering-Inertia) block, where all tanks states are obtained. This includes, for each tank, the mass, center of mass, moment of inertia of the liquid at rest, as well as the overall mass properties as aggregated to the entire vehicle. The propellant moment of inertia I_0 in the tanks can be obtained by either: (i) appropriately scaling the moment of inertia of the solid propellant; (ii) interpolating the values provided within provided tables after specialized fluid dynamics simulations. Similarly, the sloshing parameters are either computed using simplified formulas, *e.g.* as in [4], or interpolated as well. Regardless of the method, they must ensure that equations (1) and (2) hold true. In this way, propellant consumption can dynamically alter the sloshing properties, when simulated by the Modelica model.

The *Vehicle model* (in yellow) allows simulating the vehicle dynamics based on the MCI states and the forces and torques of the actuators. The model can be swapped via the use of Simulink variants between a rigid-body model and the Modelica model (compiled and embedded via Dymola software utilities - see Section 4). The computed vehicle states are expanded into other reference frames and passed to the sensors and to the *Environment model*. The latter includes, for example, gravity gradient, atmospheric and wind models, whose effect is fed back into the vehicle dynamics.

Note that, within the plant, numerical integration happens in the *MCI* and *Vehicle model* subsystems, as indicated with the appropriate symbol in the Figure.

3. The Vertical Landing Vehicles Library

Acausal modeling is a declarative modeling style, meaning that it is based on equations instead of assignment statements. Equations do not specify which variables are inputs or

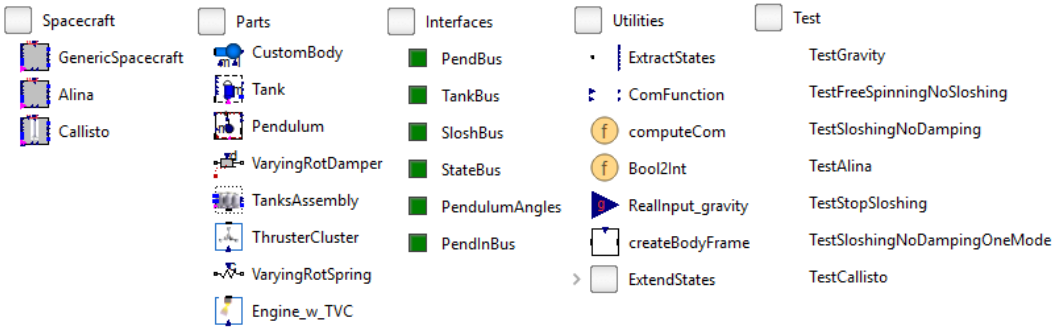

Figure 5: Listing of the current VLVLib packages and the contained classes.

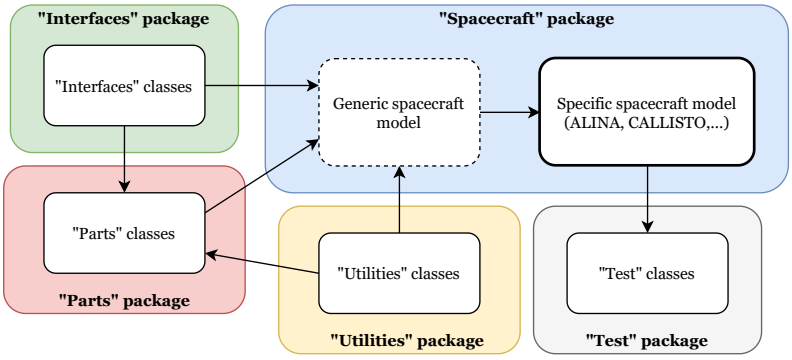

Figure 6: Graph highlighting the functional dependencies between the classes of the VLVLib.

outputs, whereas in assignment statements variables on the left-hand side are always the outputs (results) and variables on the right-hand side are the inputs. Thus, the causality of equation-based models is unspecified and becomes fixed only when the corresponding equation systems are solved. Acausal models are sometimes addressed to as physical models because they are very well suited for representing the physical structure and components of a system.

The class concept and inheritance property are the keys for reuse of modeling knowledge in Modelica. Additionally, in contrast to many other object-oriented languages, Modelica also provides the 'inheritance through modification' property via the so-called 'modifiers'. This makes reuse even easier and the implementation concise. In fact, the three operations of inheritance, specialization through modifications, and object instantiation, can be expressed within a single Modelica declaration. For a comprehensive coverage of the language specifications, the interested reader can refer to [29].

The Multibody library classes, part of the MSL, is used to enable the creation of the library. The VLVLib is based on five main subpackages: `Spacecraft`, `Parts`, `Interfaces`, `Utilities` and `Test` (Figure 5). Their logic dependencies are expressed in Figure 6. The `Spacecraft` package contains one main class, called `GenericSpacecraft`, which instantiates all the base classes necessary for vehicle modeling. By exploiting the mentioned Modelica features, it is possible to tailor the generic model to form a specialized class for the vehicle to be simulated.

3.1. Main model components

The main library components, here described in details, are:

1. a generic spacecraft model (`GenericSpacecraft`);
2. a class for the tanks assembly (`TanksAssembly`);
3. a `Tank` class for building up the whole craft's propulsion system;
4. the described `Pendulum` class for slosh oscillations modeling;
5. a `CustomBody` class to enable custom gravity inputs.

They are presented below in reversed order.

3.1.1. The `CustomBody` *class*

The `CustomBody` model is modified `Body` class that allows for mass and moment of inertia dynamic change during simulation. Another advantage is that it exploits a gravity vector inputted by the user, hence not defined directly by the `World` model. Therefore, `CustomBody` is used throughout all classes in place of the standard `Body` model. Further details are given in Section 3.1.5.

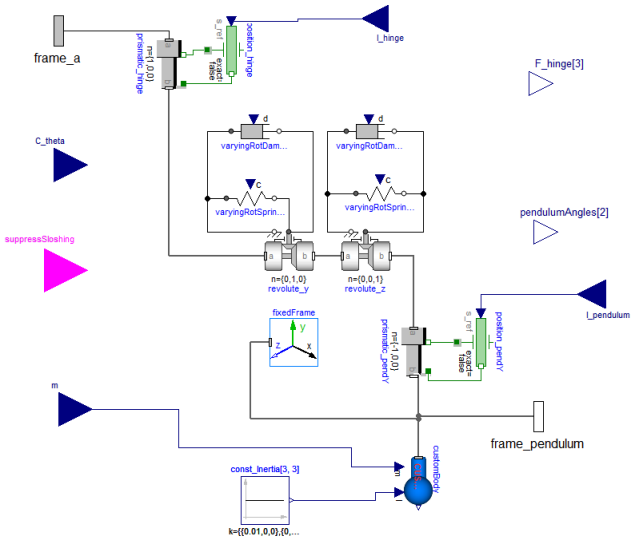

Figure 7: `Pendulum` class diagram view.

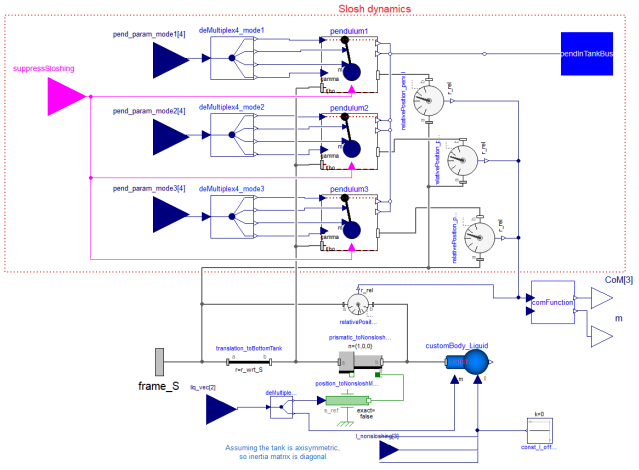

Figure 8: `Tank` class diagram view.

3.1.2. *The pendulum model*

The `Pendulum` class is responsible for modeling the pendulum for slosh dynamics representation (Figure 7). It is allowed to move in the $x - y$ and $x - z$ planes by means of a universal joint which allows two rotational degrees of freedom. As from requirements, the model can adapt to the sloshing parameters changes by means of prismatic joints: they allow for the translational motion of the model elements so to reflect the changes in hinge height and pendulum arm. A rotational damper is added at the pendulum hinge point in order to simulate energy dissipation. This component can also act in synergy with a rotational spring, so to stop the sloshing motion when a flag `suppressSloshing` is set to true and restore the pendulum neutral position with $\theta = 0$. When this happens, the damping coefficient and the spring stiffness pass, respectively, from C_θ and zero to suitable high values.

```
model Tank "Tank model"
 Sloshing.Pendulum pendulum1 if nSloshModes >= 1
 Sloshing.Pendulum pendulum2 if nSloshModes >= 2
 Sloshing.Pendulum pendulum3 if nSloshModes == 3
 // ...
end Tank;
```

Figure 9: Conditional declaration of component `pendulum1` within `Tank` class. The single pendulum models are instantiated (and its connections activated) for specific values of parameter `nSloshModes`.

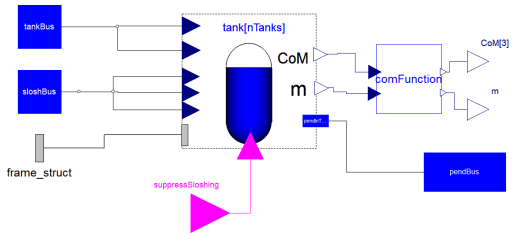

Figure 10: `TanksAssembly` class diagram view.

3.1.3. *The tank model*

The `Tank` class is in charge of modeling the fluid part of each tank by structuring the sloshing equivalent model instantiations (Figure 8). Firstly, the class uses `CustomBody` to represent the nonsloshing body with a variable mass and a variable inertia, in order to account for the idle liquid mass m_0 and for the whole liquid inertia I_0 at the prescribed filling level.

Three pendulum model instances have been used for embedding the three sloshing modes. The peculiarity of the instantiated `Pendulum` classes is that they are defined as *conditional components*. Conditional components constitute a specific feature of Modelica language, which allows the declared class to be included in the whole model at compilation time only if a specific logical condition is true [29]. If this is not the case, the class and its related connections (thus the underlying binding equations) will be entirely removed. Of course, the class instantiating a conditional component must be well-posed in case of component removal. This behavior is obtained with the simple code structure shown in Figure 9.

A specific number i of pendulum models is instantiated in each tank according to the specified number of sloshing modes (`nSloshMode`) parameter. Note that it is even possible to exclude sloshing effect, hence removing all pendula by setting `nSloshModes`=0. The parameters m_i, l_i, h_i and C_θ are provided as inputs to each pendulum model.

3.1.4. *The tanks assembly*

The `TanksAssembly` class is meant instantiate as many `Tank` components to model the complete vehicle tanks configuration. To do so, the `Tank` class has been declared as a `nTanks`-long vector, with $nTanks = K$ being the prescribed number of tanks. The tank- and sloshing-related parameters are accordingly inputted to every element of the classes

```
model TanksAssembly "Tanks assembly"
// All declarations
equation
// ...
 for i in 1:nTanks loop
 // Connecting the structural frame
 connect(frame_struct, tank[i].frame_S);
 // Connecting inputs/outputs
 connect(tankBus.liq_vec[:, i],tank[i].liq_vec);
 // List of all connections
 // statements below...
 end for;
//...
end TanksAssembly;
```

Figure 11: Recursive `connect()` command for each tank within `TanksAssembly` class.

```
model Alina "ALINA spacecraft model"
  extends GenericSpacecraft(
    //CoM position of the dry craft
    r_ComDry={0.2,0,0}, // confidential

    m_dry=1000, // confidential, dry mass
    tanksAssembly(
      nTanks=11, // number of tanks
      // position of each tank frame wrt frame_S
      r_wrt_S=[
            0.1,    -0.2,    0.7;
            0.1,     0.2,   -0.95;
            0.1,     0.2,    0.95;
            0.1,    -0.2,   -0.95;
            0.1,     0.95,   0.2;
            0.1,    -0.95,  -0.2;
            0.1,    -0.95,   0.2;
            0.1,     0.95,  -0.2;
            0,       0,      0;
            0,       0,      0;
            0,       0,      0], // confidential
      // Initial position of the pendula
      // per each mode
      anglesYz_t0_mode1 = zeros(nTanks,2),
      anglesYz_t0_mode2 = zeros(nTanks,2),
      anglesYz_t0_mode3 = zeros(nTanks,2),
      nSloshModes = {3,3,3,3,3,3,3,3,0,0,0}))
      // # sloshing modes per tank
end Alina;
```

Figure 12: `Alina` spacecraft class extends `GenericSpacecraft`. Lines marked with `confidential` comment have modified values for confidentiality reasons.

vector. With this implementation, there is no need to manually reinstantiate as many tank components as needed by the specific vehicle configuration. The diagram view of the `TanksAssembly` class is shown in Figure 10, whereas the code structure for recursive connections in Figure 11.

3.1.5. The spacecraft class

The `GenericSpacecraft` class includes all the components to assemble a complete vehicle model; specific spacecraft configurations are derived starting from this class. The main class inputs and outputs are summarized in Table 1.

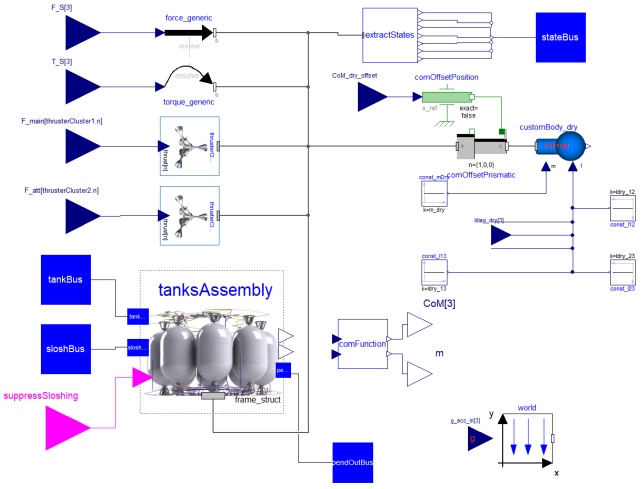

Figure 13: `GenericSpacecraft` class diagram view.

To work correctly the model must include the `World` class which defines an inertial frame (called 'World') for referencing all model states and the gravity field. The gravity field function, which acts onto every rigid body element below the hierarchy where the `World` component is instantiated, can be either disabled, set to uniform or point-wise. Nevertheless, the `World` gravity function is declared in the class as `replaceable`; this means that the user can redefine it and implement a custom gravity function (which might depend on some specific model states) and substitute it to the default ones. Despite this possibility, our purposes require the gravity vector to be a model input, to retain eventual already existing and tested functions in Matlab/Simulink. The solution consists in disabling entirely the gravity field in the `World` component, adopting a real input called `g_acc_in` and declaring it with the `inner` attribute in the `GenericSpacecraft` class. Variable `g_acc_in` is instantiated, as `outer` component, by `CustomBody` class and included into its dynamics without additional inputs or interfaces. Note that with this approach, it is implicit that the distance between each vehicle's components is neglected as compared with the distance from the source of the gravitational field. The most important connection element in Figure 13, showing `GenericSpacecraft` class, is the structural frame $\mathcal{F}_S$ represented with a gray line between the instantiated classes. From this single frame, all the others can be derived upon specific needs. The body `customBody_dry` represents the dry vehicle structure. It must be noticed how the inertia of the dry structure and its distance from the structural frame are inputted from outside of the model to enable center of mass or inertia variations due to any vehicle dry mass distribution change.

The propulsion part is integrated via the `tankAssembly` class; its inputs are two connectors, `tankBus` and `sloshBus`, acting as signal busses collecting the necessary sloshing model parameters, and the `suppressSloshing` boolean vector meant to eventually stop the sloshing motion and restore the pendula neutral position independently for each tank. The outputs are the dry vehicle states and auxiliary quantities,

like the pendula sloshing angles (via the `pendulumAngles` connector) or the overall center of mass with respect to $\mathcal{F}_S$. The latter may be used by other functions in order to obtain, among others, a body frame $\mathcal{F}_B$ centered at the vehicle's center of mass. This result is useful whenever it is necessary to account for the effect of some forces or torques expressed in $\mathcal{F}_B$ (e.g. aerodynamics). Two other inputs for generic forces and torques with respect to $\mathcal{F}_S$ can be used upon need. Vehicle's states with respect to the 'World' frame are collected within the `stateBus` connector and outputted.

3.1.6. Modeling specific vehicles

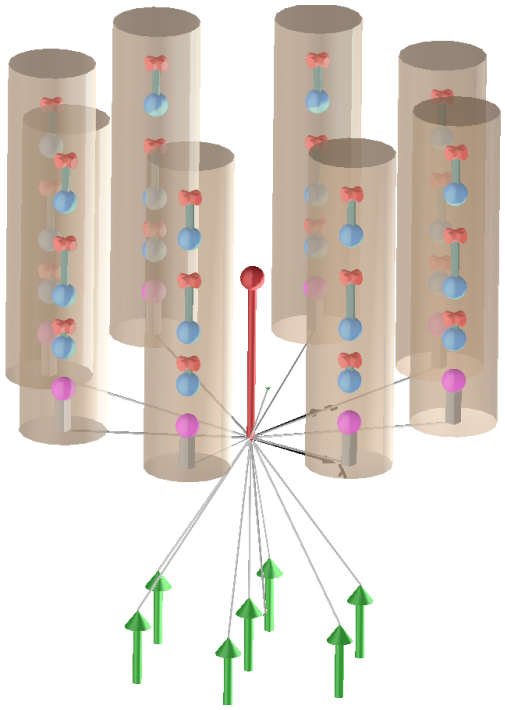

Figure 14: Visualization of a generic spacecraft with eight tanks obtained via Dymola software.

The spacecraft specific class is obtained by extending `GenericSpacecraft`, as shown in Figure 12, where `Alina` model extends `GenericSpacecraft` class and modifies the parameters of the its instantiated classes. It is invoked by other specific testing classes (in the `Test` subpackage) for verification purposes within Dymola. Most importantly it is the class to be compiled and exported into the Simulink simulator. In the example, the number of sloshing modes per tank has been defined using the integers vector `nSloshModes` and the pendula initial position about y- and z-axes via `anglesYz_t0_mode1`, `anglesYz_t0_mode2` and `anglesYz_t0_mode3`. The position of each $\mathcal{F}_{\text{tank}_i}$ with respect to $\mathcal{F}_S$ has been set via `r_wrt_S`. The dry mass is set via `m_dry`. The initial states with respect to the inertial frame can be set similarly as well.

Figure 14 shows the multibody configuration of an unspecified spacecraft with eight tanks: the dark red body represents the dry mass (and related moment of inertia) of the vehicle. Within the eight tanks, one can find three pendula for sloshing and a purple body representing the liquid nonsloshing mass. Grey elements are mass-less visual connectors; lastly, seven longitudinal thrusters exert a balanced force (green arrows) to the craft.

Inputs	**Description**
g_acc_in	gravity vector wrt inertial frame
F_S, T_S	external forces and moments expressed in $\mathcal{F}_S$ frame
tankBus	bus for position, mass and inertia of the nonsloshing body
sloshBus	bus containing all pendulum parameters
suppressSloshing	boolean vector for selective sloshing suppression
CoMdryOffset	deviation of the dry CoM position from nominal one
IdiagDry	inertia of the dry body
Outputs	**Description**
stateBus	vehicle states ($\mathcal{F}_S$ frame wrt inertial frame)
pendOutBus	pendula angles
m	overall vehicle mass
CoM	overall vehicle center of mass

Table 1
Inputs and outputs of `GenericSpacecraft` class.

3.2. Other library components

In the `Parts` subpackage, there are also other models meant to enable specific actuators to exert their force (and torque) directly onto the structural frame, instead of computing the sum of all forces and torques acting on the vehicle expressed in the structural frame beforehand. It is the case of `ThrusterCluster` and `Engine_w_TVC`. The first is meant to simulate a parametrized number of thrusters acting in a specific direction and specifically positioned with respect to the structural frame. It would therefore be possible to input directly the forces developed by each thruster from Simulink; likewise, the second acts similarly by modeling a single gimballed thruster. Some Modelica connectors have been defined for encapsulating the signals, like `PendBus`, `TankBus`, `SloshBus`, or `StateBus`. They are included in the `Interfaces` subpackage. The `Utilities` subpackage defines some auxiliary functions. Some of them would be necessary for the main components, others just for convenience; `ComFunction`, for instance, computes the center of mass of an arbitrary number of bodies. Lastly, the `Test` subpackage implements the classes for testing within the Dymola environment. The `GenericSpacecraft` or any vehicle-specific class is instantiated and fed with appropriate inputs, so to validate the correctness of the modeling (note that this sort of testing is open-loop). Other minor components exist and will not be further described.

4. Dymola-Simulink integration

For integration within Simulink, the Modelica model must be exported as a binary file. Two formats can be

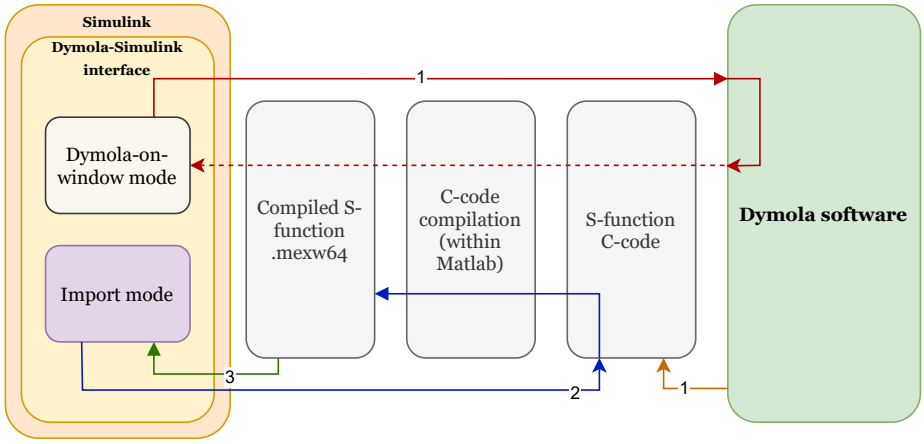

Figure 15: Process chain to integrate a compiled S-function in Simulink: with the "Dymola-on-windows" mode a single user interaction is required and triggered within Simulink environment (red arrow). The process is automatic and takes the compiled S-function to include it in the Dymola-Simulink interface directly; with the manual mode, the user has to generate the S-function C-code via Dymola, and afterwards compile it in Matlab for use in the Dymola-Simulink interface.

used: S-function or Functional Mockup Unit (FMU). An S-function can be seen as a "custom" Simulink block: it is defined as a language description of a Simulink block written in Matlab code, C, C++, or Fortran. It can then be compiled as MEX file, called and automatically loaded and executed. An S-function can handle very well continuous- and discrete-time dynamics, as well as parametrization and initialization routines. On the other hand, an FMU is a simulation model created to adhere to the Functional Mockup Interfaces (FMI) standard: FMI is an open standard for exchanging dynamic simulation models between different tools in a standardized format [45]. It has rapidly evolved in the last years especially in the automotive field.

Here, an S-function is used: compiling a Modelica model as S-function is a feature made available from Dymola software. The interface allows Dymola software to compile a specific class as S-function in C-code, which needs to be then recompiled using a C/C++ compiler into a binary MEX file and lastly embedded in a Simulink subsystem; Dymola provides a proprietary Simulink block which acts as compiled S-function wrapper to facilitate the user's inputs/outputs via a graphical interface or programmatically. This is shown in Figure 15. The whole compilation chain can be ran through in two different ways: with an "on-the-fly" compilation and integration of the Modelica model within Simulink, or manually. The first requires the use of the Dymola-Simulink interface in the so-called "Dymola-on-window mode"; it is convenient for rapid prototyping, as the Dymola engine is directly called using the Simulink interface, and the Dymola compilation triggered for C-code S-function compilation. Afterwards, a C-compiler executed within Matlab brings the model into its final binary form and linked with the Simulink block. Instead, the manual option requires the user to perform manually all the steps. An example is shown in Figure 16, where ALINA spacecraft model has been integrated in Simulink.

Nr.	Fluid type	Sloshing modes	Capacity
1-4	MON	3	> 450kg
5-8	MH_2	3	> 300kg
9	He	0	< 10kg
10	MON	0	< 0.5kg
11	MH_2	0	< 0.5kg

Table 2
ALINA propulsion system configuration.

5. Case studies

To demonstrate the potential of the approach described so far, two study cases have been considered: a Moon landing and a return-to-launch-site scenario of a VTVL reusable launch vehicle. In both cases a simulation environment able to perform high-fidelity simulations, unit testing and Monte-Carlo campaigns has been integrated and relies on DLR's Hybrid Simulation Platform for Space Systems (HPS) [46]. In the following, the vehicles configuration is explained and simulation results shown. For both scenarios, the control strategy involves the use of linear controllers designed with optimal control methods or with an $\mathcal{H}_\infty$-based synthesis. To date, both control systems are in their mid-stage development phase and will be further improved; guidance systems are also present to ensure the selection of the most optimal trajectory point based on the vehicle's states, as well as complex navigation algorithms.

5.1. ALINA spacecraft Moon landing

ALINA is a spacecraft meant to enable up to 100kg of payload deliveries on the surface of the Moon [42] (Figure 2). It has an overall mass of about 4400kg, composed by more than 70% of propellant. The actual propulsion system is summarized in Table 2 and shown as a schematic in Figure 17. ALINA's GNC architecture, design and test results are deeply explained in [33, 47].

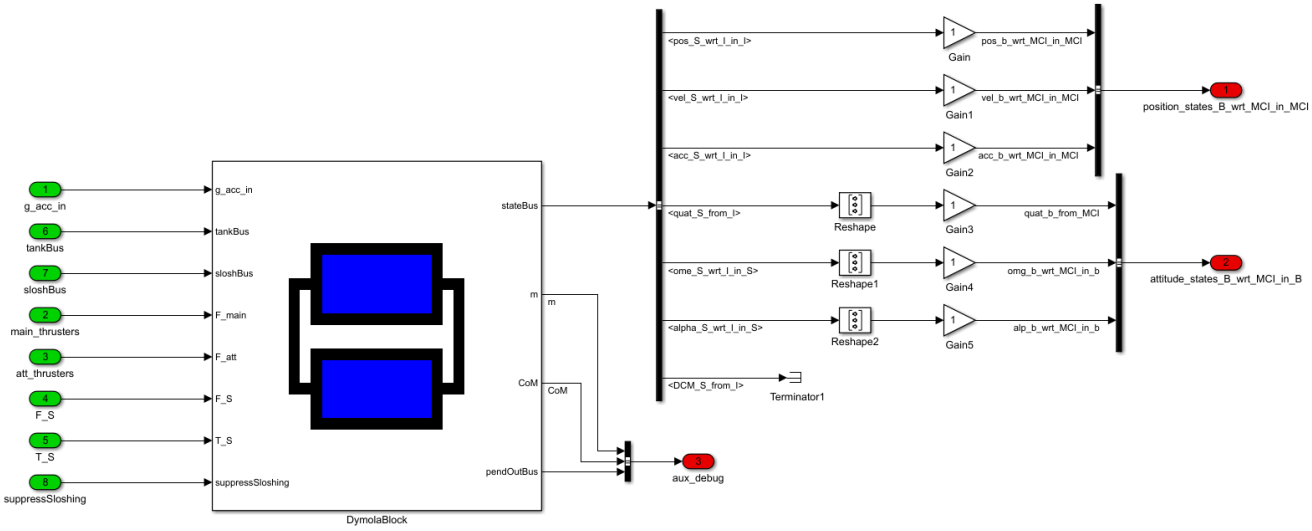

Figure 16: The `DymolaBlock` containing the vehicle model as integrated in the Simulink environment.

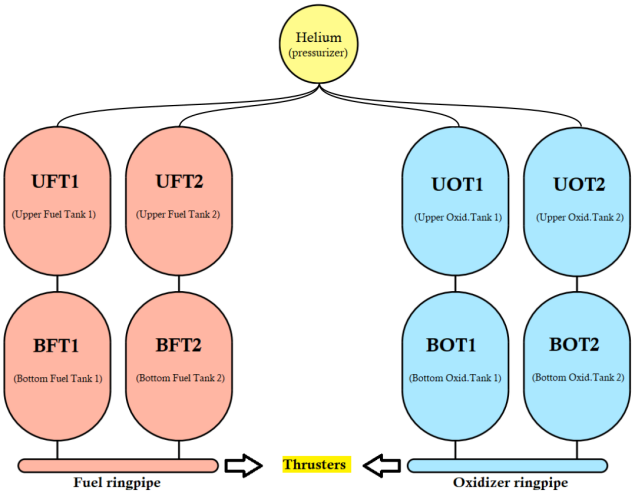

Figure 17: ALINA's propulsion system simplified schematics. Adapted from [48].

The configuration consists of a pressurizing tank, eight propellant tanks and two distribution rings. The fuel tanks are arranged in two pairs; within each pair, tanks are connected in series. To underline the serial connection, they are labeled as "Upper" (U) and "Bottom" (B). Oxidizer tanks work similarly, and the four fuel and oxidizer pair outlets feed into two apposite distribution rings. The ring pipes lead directly to several high-thrust engines and the thruster clusters for fine attitude control. To reduce development costs and ease the vehicle's layout, the tank geometry is chosen to be the same for all tanks, which implies that the oxidizer to fuel mass ratio of the ALINA spacecraft is chosen such as that the total oxidizer volume equals the total fuel volume.

The inertial frame is chosen to be a Moon Centered Inertial (MCI) frame. The simulation starts at a low altitude of about 15km from the surface. During the whole powered descent, the vehicle experiences a non-gravitational deceleration between 1.7 to 3.5m/s^2. In this dynamic regime, the sloshing frequencies will vary as function of the fuel draining: in fact, mass consumption causes vehicle's longitudinal acceleration to increase and the filling level of the tanks to decrease. This, in turn, leads to increasingly higher sloshing natural frequencies. The combined effects of these factors is shown in Figure 18. The color gradient represents the tank filling level; in the final approach part, the acceleration is reduced and the natural frequencies drop. Note that the control system does not encourage sloshing: in fact, the highest controller bandwidth is the one of the attitude closed-loop, designed sufficiently separated from the sloshing frequencies (see [33] for more details).

Figure 19 shows how sloshing affects the vehicle position error norm with respect to the planned trajectory. The plot highlights the position error norm of the vehicle simulated with the sloshing effect (Modelica model) as compared with a simple rigid body. It can be noticed how the position error is not negligible when the spacecraft requires accurate positioning during its descent and landing.

In Figure 20 the pendulum angles for BOT1, BOT2, BFT1 and BFT2 are shown: it can be noticed how they maintain well inside the model validity boundaries. The pendulum angles for the upper oxidizer and fuel tanks are not shown because of the probability to empty them first during the Earth-to-Moon cruise. Naturally, if the upper tanks still show some residual liquid, the same implementation and considerations apply.

5.2. CALLISTO RLV end-to-end mission

CALLISTO (Cooperative Action Leading to Launcher Innovation in Stage Toss back Operations) is a VTVL reusable launch vehicle demonstrator being developed by DLR, CNES and JAXA in a trilateral project [44]. Its main

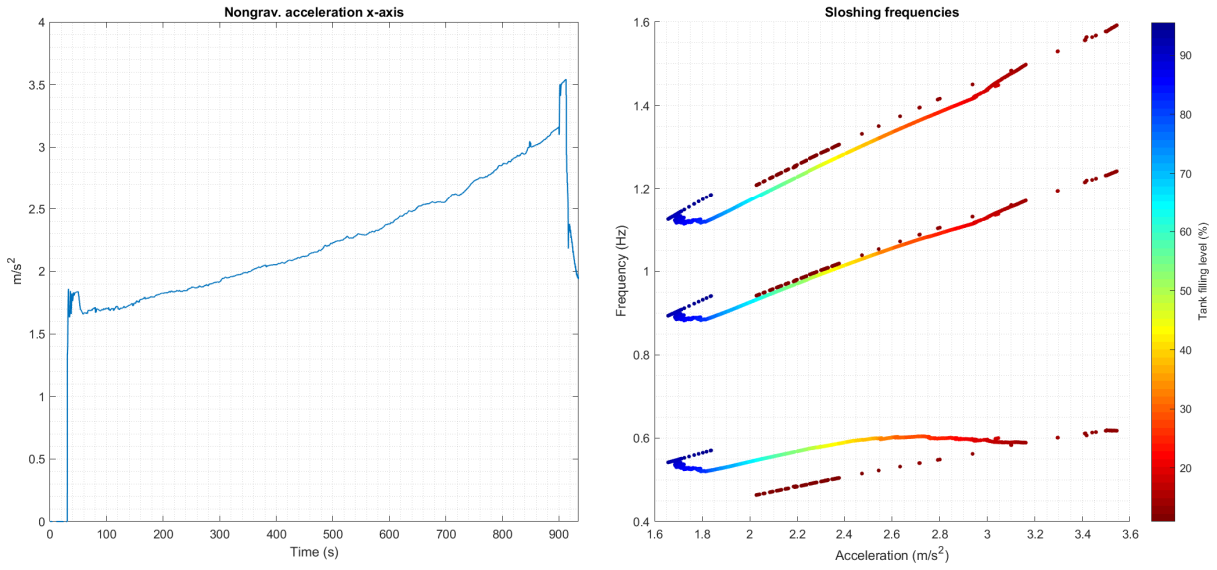

Figure 18: On the left, ALINA's non-gravitational longitudinal acceleration; on the right is sloshing frequencies as function of this acceleration and the tank filling level. The frequency is the same for both propellants, because it primarily depends on the tank geometry, which is the same by design.

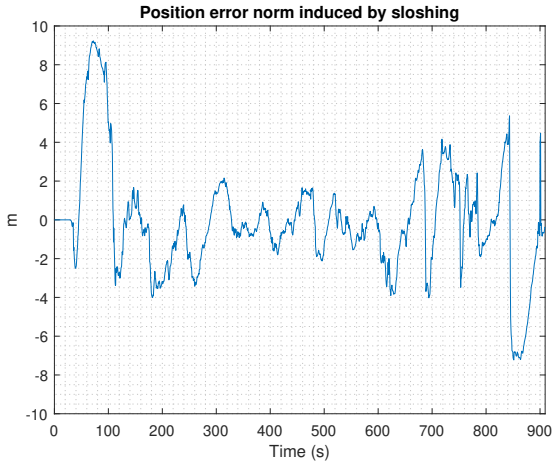

Figure 19: Position errors caused by sloshing during ALINA's powered descent and landing phase, as compared with a rigid body model.

objective is to demonstrate the capability to recover a vehicle with toss-back trajectory. This goal implies many sub-objectives for the propulsion system, such as in-flight engine reignition, propellant settling and sloshing management. An on-board computer has to run suitable robust algorithms for guidance, control and navigation during all phases for enabling vehicle stability and a safe landing. CALLISTO vehicle is foreseen to have more than 2 tons of propellant, mainly divided between liquid oxygen (LOX) for more than 1500kg and liquid hydrogen (LH_2) for more than 300kg.

The mission has four main phases: ascent, boostback, aerodynamic and landing. At ascent the rocket is meant to reach its apogee, before reorienting itself and proceed throughout the boostback maneuver. An aerodynamically controlled descent follows to guide the rocket towards the designed landing point. Lastly, the vehicle executes a landing burn for a soft touchdown. Insights of the control system design can be found in [49, 50].

For both main tanks, only one sloshing mode is accounted for. Pendulum angles for both tanks obtained from a closed-loop simulation. For instance, LH_2 tank pendulum angles with respect to vehicle's x- and y-axes are shown normalized in Figure 21. Even at this GNC system design early stage, the validity boundaries for the sloshing model are fulfilled, despite not directly visible in the plot for data confidentiality reasons.

At boostback, the vehicle experiences a null non-gravitational acceleration on its longitudinal axis; in this phase the liquid undergoes a chaotic motion and spreads in the tank. From a simulation viewpoint, this implies that the equivalent sloshing model is not valid anymore, and the pendulum motion must be inhibited via the boolean variable `suppressSloshing`. In fact, Figure 21 evidences how, in the highlighted area where `suppressSloshing=true`, the pendulum restores its neutral position and stops oscillating. Please note, however, that in this situation the center of mass must be artificially manipulated via the model inputs to obtain another center of mass location that best represents the liquid chaotic motion in the tanks. Furthermore, after boostback the liquid needs some time before settling completely: this means that the pendulum motion can also restart from a non-neutral position. This is possible using Modelica features and the new initial angle has to be accounted for as uncertain parameter in the Monte-Carlo campaigns. Future library releases will consider this aspect too.

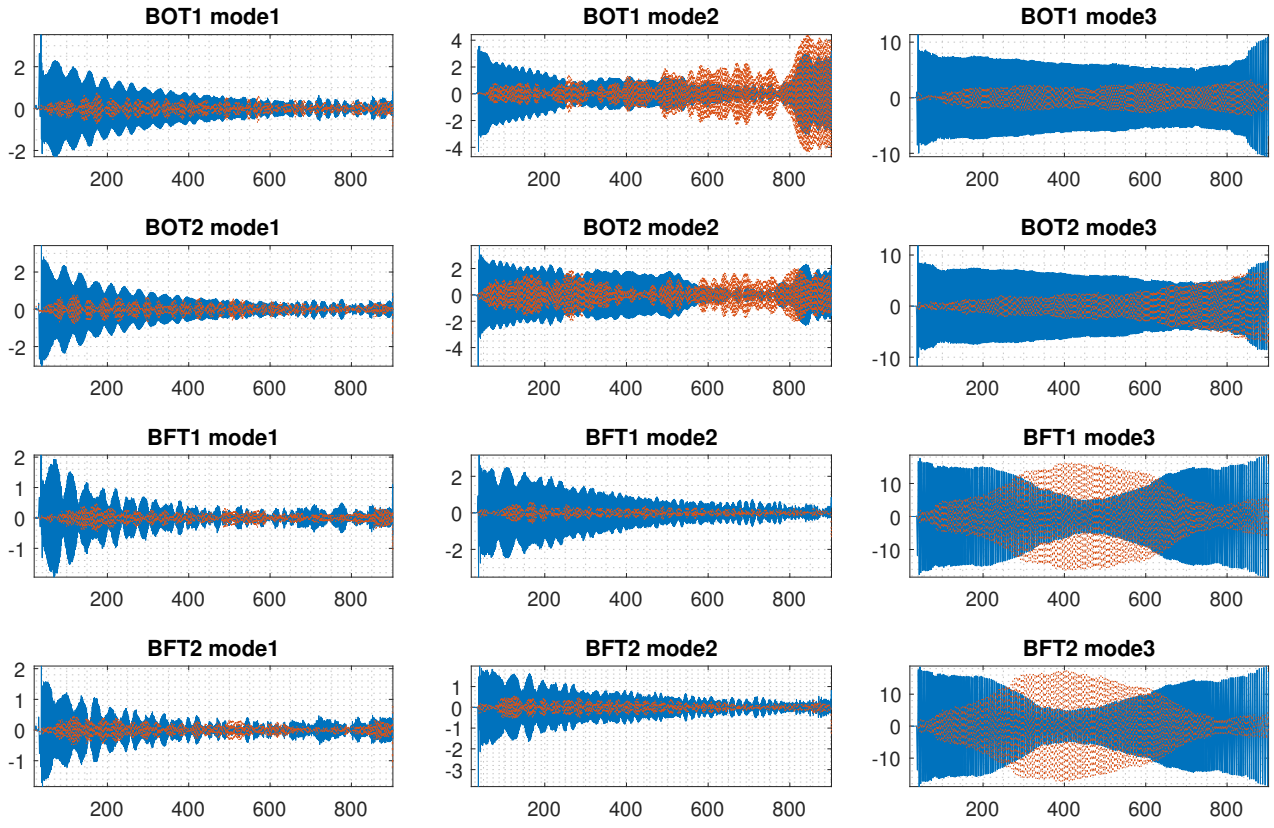

Figure 20: ALINA's tank pendulum angles about y- and z- axes, in degrees. Abscissa is time (seconds).

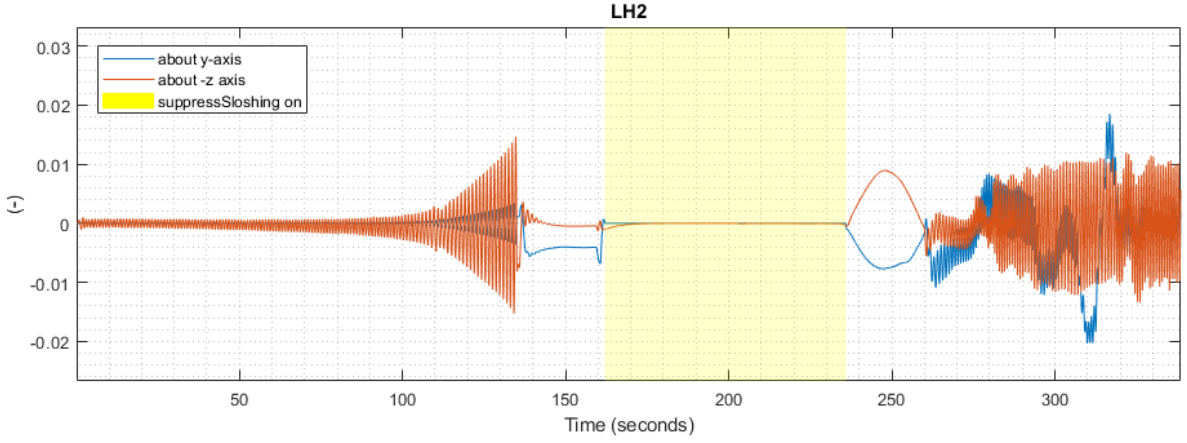

Figure 21: CALLISTO normalized LH_2 pendulum angles.

6. Conclusions

In this paper, the issues of implementing a spacecraft simulation model aimed at the verification of the GNC algorithms in presence of slosh dynamics, have been tackled. Mechanical equivalents are used to model sloshing with the purpose of investigating upon the resulting vehicle stability properties. Here the problem has been addressed from a simulation standpoint: by exploiting the combined potential of Simulink and Modelica language, a new framework to model and simulate vehicles with slosh dynamics has been introduced. The proposed modeling strategy allows changing of the propulsion configuration with minimal effort from the developer through Modelica inheritance and parameterization properties of a generic model.

Classic assumptions and limitations when simulating slosh dynamics, are overcome: multiple sloshing modes in three dimensions can be simulated, for any number of tanks,

as well as propellant draining. Additionally, the sloshing motion can be artificially stopped when no (non-gravitational) longitudinal acceleration is exerted on the vehicle, thus avoiding a complete nonphysical disruption of the simulation when pendula are left freely spinning.

The described methodology integrates the current industrial state-of-practice GNC modeling and simulation approach, which mainly employs Simulink environment. This avoids unnecessary replication of well-consolidated functions into another modeling language. Moreover, the solution guarantees that pre-developed simulation infrastructure can be re-used, and that a rigid-body vehicle model can be easily swapped with the more advanced Modelica one.

The presented results show that the approach can successfully integrate the GNC simulation, validation and verification work logic of fuel sloshing dynamics.

Declaration of competing interest

The authors declare that they have no competing interests.

Funding sources

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] J. A. Frosch, D. P. Vallely, Saturn AS-501/S-IC flight control system design., Journal of Spacecraft and Rockets 4 (1967) 1003–1009.

[2] H. Strauch, K. Luig, S. Bennani, Model Based Design Environment for Launcher Upper Stage GNC Development, in: Workshop on Simulation on for European Space Programmes (SESP), ESA/ESTEC Noordwijk, The Netherlands, 2018.

[3] Abramson, Dynamic Behavior of Liquids in Moving Containers with Applications to Space Vehicle Technology, Government Press, 1966.

[4] F. T. Dodge, et al., The new" dynamic behavior of liquids in moving containers", Southwest Research Inst. San Antonio, TX, 2000.

[5] T. A. Lance, Analysis of Propellant Slosh Dynamics and Generation of an Equivalent Mechanical Model for Use in Preliminary Voyager Autopilot Design Studies, Technical Memorandum 19680017770, NASA, 1966.

[6] O. Bayle, V. L'Hullier, M. Ganet, P. Delpy, J.-L. Francart, D. Paris, Influence of the ATV Propellant Sloshing on the GNC Performance, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, American Institute of Aeronautics and Astronautics, Monterey, California, 2002. doi:10.2514/6.2002-4845.

[7] H. Q. Yang, J. Peugeot, Propellant Sloshing Parameter Extraction from CFD Analysis, in: 46th AIAA Joint Propulsion Conference, 2010.

[8] R. A. Ibrahim, Liquid Sloshing Dynamics: Theory and Applications, Cambridge University Press, 2005.

[9] J. G. Pérez, R. A. Parks, D. R. Lazor, Validation of Slosh Model Parameters and Anti-Slosh Baffle Designs of Propellant Tanks by Using Lateral Slosh Testing, in: 28th Aerospace Testing Seminar, 2012.

[10] L. Mazmanyan, M. A. Ayoubi, Attitude Motion of a Spinning Spacecraft with Fuel Sloshing in High-G Maneuvers, in: Proc. AAS/AIAA Astrodyn. Spec. Conf., 2013, pp. 793–812.

[11] A. Shekhawat, C. Nichkawde, N. Ananthkrishnan, Modeling and Stability Analysis of Coupled Slosh-Vehicle Dynamics in Planar Atmospheric Flight, in: 44th AIAA Aerospace Sciences Meeting and Exhibit, 2006.

[12] S. Cho, M. McClamroch, M. Reyhanoglu, Feedback control of a space vehicle with unactuated fuel slosh dynamics, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, Guidance, Navigation, and Control and Co-Located Conferences, American Institute of Aeronautics and Astronautics, 2000. doi:10.2514/6.2000-4046.

[13] M. Reyhanoglu, S. Cho, N. H. McClamroch, Control Problems for Planar Maneuvers of a Spacecraft with Fuel Slosh, in: Proc. Int. Conf. on Nonlinear Problems in Aviation, 2000, pp. 579–584.

[14] M. Reyhanoglu, Maneuvering control problems for a spacecraft with unactuated fuel slosh dynamics, in: Proceedings of 2003 IEEE Conference on Control Applications, 2003. CCA 2003., volume 1, 2003, pp. 695–699 vol.1. doi:10.1109/CCA.2003.1223522.

[15] M. Reyhanoglu, Modelling and Control of Space Vehicles with Fuel Slosh Dynamics, in: J. Hall (Ed.), Advances in Spacecraft Technologies, InTech, 2011. doi:10.5772/13540.

[16] M. Reyhanoglu, J. R. Hervas, Nonlinear control of space vehicles with multi-mass fuel slosh dynamics, in: Proceedings of 5th International Conference on Recent Advances in Space Technologies - RAST2011, 2011, pp. 247–252. doi:10.1109/RAST.2011.5966834.

[17] M. Reyhanoglu, J. R. Hervas, Nonlinear control of a spacecraft with multiple fuel slosh modes, in: 2011 50th IEEE Conference on Decision and Control and European Control Conference, 2011, pp. 6192–6197. doi:10.1109/CDC.2011.6160660.

[18] J. R. Hervas, M. Reyhanoglu, Control of a spacecraft with time-varying propellant slosh parameters, in: 2012 12th International Conference on Control, Automation and Systems, 2012, pp. 1621–1626.

[19] J. Rubio Hervas, M. Reyhanoglu, Thrust-vector control of a three-axis stabilized upper-stage rocket with fuel slosh dynamics, Acta Astronautica 98 (2014) 120–127.

[20] M. Navabi, A. Davoodi, M. Reyhanoglu, Modeling and control of a nonlinear coupled spacecraft-fuel system, Acta Astronautica 162 (2019) 436–446.

[21] J. R. Hervas, M. Reyhanoglu, Observer-based nonlinear control of space vehicles with multi-mass fuel slosh dynamics, in: 2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE), 2014, pp. 178–182. doi:10.1109/ISIE.2014.6864607.

[22] J. M. Adler, M. S. Lee, J. D. Saugen, Adaptive control of propellant slosh for launch vehicles, in: P. D. Dean (Ed.), Orlando '91, Orlando, FL, Orlando, FL, 1991, pp. 11–22. doi:10.1117/12.46489.

[23] M. Navabi, A. Davoodi, Fuzzy control of fuel sloshing in a spacecraft, in: 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), 2018, pp. 123–126. doi:10.1109/CFIS.2018.8336651.

[24] Y. Baozeng, Z. Lemei, Hybrid Control of Liquid-Filled Spacecraft Maneuvers by Dynamic Inversion and Input Shaping, AIAA Journal 52 (2014) 618–626.

[25] M. Deng, Y. Baozeng, Nonlinear model and attitude dynamics of flexible spacecraft with large amplitude slosh, Acta Astronautica 133 (2017) 111–120.

[26] T. Yong, Y. Baozeng, Simulation of Large-Amplitude Three-Dimensional Liquid Sloshing in Spherical Tanks, AIAA Journal 55 (2017) 2052–2059.

[27] Y. C. S. Gangadharan, B. Marsell, K. Schlee, A. Corporation, J. Sudermann, C. Walker, J. Ristow, C. Hubert, H. Astronautics, Modeling, simulation, and parameter estimation of lateral spacecraft fuel slosh, in: 12th World Multi-Conference on Systemics, Cybernetics and Informatics 2008, 2008, p. 9.

[28] S. Keith, Modeling Spacecraft Fuel Slosh at Embry, https://de.mathworks.com/company/newsletters/articles/modeling-spacecraft-fuel-slosh-at-embry-riddle-aeronautical-university.html, 2007.

[29] P. Fritzson, Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach, John Wiley & Sons, 2014.

[30] Modelica Standard Library, Modelica Association, 2021. URL: https://github.com/modelica/ModelicaStandardLibrary.

[31] M. Otter, H. Elmqvist, S. E. Mattsson, The New Modelica MultiBody Library, in: 3rd International Modelica Conference, 2003, pp. 311–330.

[32] S. Farì, D. Grande, Vector Field-based Guidance Development for Launch Vehicle Re-entry via Actuated Parafoil, in: Proceedings of the International Astronautical Congress, volume D2, International Astronautical Federation, 2021.

[33] J. L. R. Gutierrez, S. Farì, M. Winter, Control System Design for the ALINA Lunar Lander, in: Proceedings of the International Astronautical Congress, volume A3, International Astronautical Federation, 2021.

[34] L. E. Briese, K. Schnepper, P. Acquatella B., Advanced modeling and trajectory optimization framework for reusable launch vehicles, in: 2018 IEEE Aerospace Conference, IEEE, Big Sky, MT, 2018, pp. 1–18. doi:10.1109/AERO.2018.8396704.

[35] L. E. Briese, P. Acquatella B, K. Schnepper, Multidisciplinary modeling and simulation framework for launch vehicle system dynamics and control, Acta Astronautica 170 (2020) 652–664.

[36] P. Acquatella, M. J. Reiner, Modelica stage separation dynamics modeling for End-to-End launch vehicle trajectory simulations, in: H. Tummescheit, K.-E. Årzén (Eds.), Proceedings of the 10th International Modelica Conference - Lund, Sweden - Mar 10-12, 2014, volume 96, LiU Electronic Press, Lund, Sweden, 2014, pp. 589–598.

[37] G. Looye, The new DLR flight dynamics library, in: Proceedings of the 6th International Modelica Conference, volume 1, 2008, pp. 193–202.

[38] T. Pulecchi, M. Lovera, A Modelica library for space flight dynamics, in: In Proceedings of the 5th International Modelica Conference, Citeseer, 2006.

[39] D. Brück, H. Elmqvist, S. E. Mattsson, H. Olsson, Dymola for Multi-Engineering Modeling and Simulation Dymola for Multi-Engineering Modeling and Simulation, in: 2nd International Modelica Conference, 2002, pp. 55–1, 55–8.

[40] S. Farì, The Vertical Landing Vehicles Library (VLVLib): A Modelica-based approach to high-fidelity simulation and verification of GNC systems for reusable rockets, in: 73rd International Astronautical Congress (IAC), International Astronautical Federation, Paris, France, 2022.

[41] C. Martin, A. Urquia, J. Sanchez, S. Dormido, Interactive Simulation of Object-Oriented Hybrid Models, by Combined Use of Ejs, Matlab/Simulink and Modelica/Dymola, in: 18th European Simulation Multiconference, 2004.

[42] T. Kriening, Mission to the Moon and the return to Apollo 17, European Planetary Science Congress 12 (2018) EPSC2018–819.

[43] R. Kelso, J. Eyer, Europe's commercial lunar lander ALINA: Innovations from planetary transportation systems (PTS), in: 51st Annual Lunar and Planetary Science Conference, 2326, 2020, p. 1018.

[44] E. Dumont, T. Ecker, C. Chavagnac, L. Witte, J. Windelberg, J. Klevanski, S. Giagkozoglou, CALLISTO - Reusable VTVL launcher first stage demonstrator, in: Space Spropulsion Conference 2018, Sevilla, Spanien, 2018.

[45] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauss, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J.-V. Peetz, S. Wolf, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, in: The 8th International Modelica Conference, Technical Univeristy, Dresden, Germany, 2011, pp. 105–114. doi:10.3384/ecp11063105.

[46] R. Schwarz, S. Bremer, D. Seelbinder, M. List, B. Rievers, T. Kato, F. Wöske, The Hybrid Simulation Platform for Space Systems (HPS): A Modular MATLAB/Simulink-Based Simulation Library for GNC Systems Development, in: 70th International Astronautical Congress (IAC), Washington, D.C., United States, 2019.

[47] M. Winter, H. Krüger, S. Farì, J. L. Redondo, B. Razgus, S. Woicke, D. Seelbinder, A. Wenzel, S. Theil, B. Maass, M. Sagliano, G. F. Trigo, A. Heidecker, R. Schwarz, S. Bremer, ALINA Moon Lander GNC – Architecture, Design and Test Results, in: 11th International ESA Conference on Guidance, Navigation & Control Systems, 2021.

[48] N. M. Dick, Development and Verification of a Continuous Estimation of Fuel Reserves and Dynamic Mass Properties of a Space Vehicle, Master's thesis, University of Bremen, 2020.

[49] J. A. M. Hernandez, M. Sagliano, A. Heidecker, D. Seelbinder, S. Farì, S. Theil, S. Woicke, E. Dumont, Ascent Flight Control System for Reusable Launch Vehicles: Full order and structured H_∞ designs, in: 11th International ESA Conference on Guidance, Navigation & Control Systems, 2021.

[50] M. Sagliano, T. Tsukamoto, A. Heidecker, J. M. Hernández, S. Farì, M. Schlotterer, S. Woicke, D. Seelbinder, S. Ishimoto, E. Dumont, Robust Control for Reusable Rockets via Structured H_∞ Synthesis, in: 11th International ESA Conference on Guidance, Navigation & Control Systems, 2021.