doi: 10.82124/CEAS-GNC-2026-103 CEAS-GNC-2026-103

Data-driven Learning of LPV Surrogate Models of Fuel Sloshing

E. Javier Olucha PhD candidate, Control Systems Group, Eindhoven University of Technology, Eindhoven, The Netherlands. e.j.olucha.delgado@tue.nl

Valentin Preda GNC System Engineer, Department of Guidance, Navigation and Control, European Space Agency, Noordwijk, The Netherlands. valentin.preda@esa.int

Amritam Das Assistant Professor, Control Systems Group, Eindhoven University of Technology, Eindhoven, The Netherlands. am.das@tue.nl

Roland Tóth Full Professor, Control Systems Group, Eindhoven University of Technology, Eindhoven, The Netherlands. r.toth@tue.nl
Senior Research Fellow, Systems and Control Laboratory, HUN-REN SZTAKI, Budapest, Hungary. toth.roland@sztaki.hun-ren.hu

ABSTRACT

This paper aims to enhance the efficiency of validation and verification campaigns involving fuel sloshing phenomena. Our first contribution is the development of an open-source, high-fidelity and computationally efficient two-dimensional smoothed-particle hydrodynamics-based fuel sloshing simulator that reproduces the dynamics of a spacecraft with a partially filled tank with liquid propellant. Implemented in Python using Jax, the simulator leverages GPU parallelization and supports automatic differentiation, enabling rapid generation of simulation data and system linearizations for general surrogate modelling purposes. Our second contribution is the demonstration of a practical methodology for constructing surrogate models of fuel sloshing from input–output data generated by the simulator, targeting rapid simulation and model-based control applications. The surrogate model employs a *linear parameter-varying* (LPV) state-space structure with affine dependence on the scheduling variables, providing an accurate yet computationally efficient approximation of the sloshing dynamics. The capabilities of the proposed approach are demonstrated through closed-loop simulations of a rigid spacecraft with a partially filled fuel tank for two manoeuvre profiles under zero-gravity conditions. The identified surrogate enables simulations that are two orders of magnitude faster than the high-fidelity model.

Keywords: Surrogate modelling, Fuel sloshing, Smoothed-Particle Hydrodynamics, Linear-Parameter Varying

Nomenclature

GPU	=	Graphics processing unit
SPH	=	Smoothed-Particle Hydrodynamics
SS	=	State-Space
LTI	=	Linear-Time Invariant

LPV	=	Linear-Parameter Varying
CoM	=	Centre of mass
$\mathcal{B}$	=	Rigid body representing the spacecraft
m, J	=	Mass and inertia of the spacecraft rigid body, respectively
$\mathcal{R}_\mathcal{W}, \mathcal{R}_\mathcal{B}$	=	Inertial world and body-fixed reference frames, respectively
r, θ	=	Linear and angular position of the spacecraft at the CoM with respect to $\mathcal{R}_\mathcal{W}$
$\mathcal{D}_N$	=	Data set
ϑ	=	Surrogate model parameters
m_i, ρ_i, A_i	=	Mass, density and field quantity of particle j, respectively
W	=	Radially symmetric kernel function with compact support
h	=	Smoothing length of a kernel function W
ρ_0, α	=	Rest density and viscous factor of the fluid, respectively
β, γ_1	=	Fluid-boundary viscous factor and boundary correcting factor, respectively

1 Introduction

In aerospace systems such as satellites, launchers, and upper stages, liquid propellants are used for vehicle propulsion when executing various manoeuvres such as flight-to-orbit, rendezvous, or station keeping. Due to propellant consumption, these systems typically operate with partially filled tanks. Consequently, during manoeuvres, the propellant experiences motion relative to the tank walls, a phenomenon known as *fuel sloshing* [1, 2]. The motion of the fluid and its interaction with the fuel tank boundaries induce a dynamic coupling between the fluid and the rigid body, effecting the overall spacecraft dynamics. These fluid–solid interactions can degrade pointing accuracy and may even lead to instability if not properly accounted for [3–5]. Moreover, as spacecrafts become more agile and lightweight, the impact of fuel sloshing becomes increasingly critical and must be carefully addressed during both controller design and *validation and verification* (V&V) campaigns.

Accurate representation of fuel sloshing requires *high-fidelity* models capable of capturing complex physical effects such as free-surface deformation, viscous damping and the interaction between the fluid and the solid boundaries. However, such models are computationally expensive, high-dimensional, and strongly nonlinear, and they are rarely available in an analytic *state-space* (SS) representation. As a result, they cannot be directly integrated into conventional control design or system analysis toolchains, and their use is typically restricted to offline analyses. Furthermore, their computational complexity limits the efficiency and cost-effectiveness of V&V campaigns and hinders rapid system-level evaluations during early design stages.

To circumvent these challenges, *surrogate models* can be employed to approximate the effects of fuel sloshing. A surrogate model is a simplified representation of a high-fidelity model with minimal complexity, but with sufficient system information and adequate model structure to reach a specific utilization objective. The development of suitable fuel sloshing surrogate models is therefore a key enabler for efficiently integrating accurate fluid–solid interactions into early-stage engineering processes.

In the context of fuel sloshing, two main classes of models are primarily used in the literature: (i) high-fidelity *computational fluid dynamics* (CFD) [6–9] or *smoothed particle hydrodynamics* (SPH) [10–12] representations that are accurate but suffer from the drawbacks aforementioned, and (ii) low-fidelity, control-oriented analytical or empirical models, often based on mechanical analogies [13–17] with low complexity but may fail to capture the sloshing phenomena with sufficient accuracy. An intermediate class of surrogate models that balances physical fidelity and computational tractability remains largely missing for early-stage engineering tasks and downstream V&V campaigns.

The systematic extraction of surrogate models typically relies on either analytical descriptions of the underlying system dynamics or on datasets composed of system inputs, measured outputs, state

trajectories, or linearized dynamics around selected operating points. For systems with fuel sloshing, analytical descriptions are rarely available, whereas data can be generated using suitable physics-based simulators. However, despite the availability of several fuel sloshing simulators, open-source tools capable of efficiently generating informative datasets in terms of system trajectories or linearizations are mostly unavailable. Such datasets are essential for constructing surrogate models that approximate the input–output behaviour, and potentially the internal dynamics, of high-fidelity simulations. Therefore, developing an efficient fuel sloshing simulator capable of generating these datasets and establishing a consistent methodology for surrogate model extraction are crucial steps towards improving the overall efficiency of aerospace engineering processes.

In this paper, two main contributions are presented. The first is the development of an ultra-fast, open-source, high-fidelity *two-dimensional* (2D) SPH-based fuel sloshing simulator that: (i) leverages *graphics processing units* (GPU) for parallel computation of particle interactions; (ii) exploits *just-in-time* (JIT) compilation provided by JAX [18]; and (iii) supports *automatic differentiation* [19] to efficiently obtain linearized dynamics around arbitrary operating points. The simulator provides a flexible framework for exciting a spacecraft containing a tank partially filled with fluid propellant and enables systematic dataset generation for surrogate modelling purposes. The second contribution is the demonstration of a practical methodology for learning *linear parameter-varying* (LPV) surrogate models of fuel sloshing dynamics using data generated by the simulator. In particular, we show how input–output data can be used to identify a self-scheduled LPV state-space model with affine dependence on the scheduling. The resulting surrogate model provides an accurate yet computationally efficient approximation, making it suitable for rapid simulation and integration into control design and V&V workflows.

This paper is structured as follows. First, we introduce the surrogate modelling problem for the fuel sloshing scenario in Section 2, and we detail the development of the high-fidelity SPH-based fuel sloshing simulator in Section 3. Section 4 presents the LPV surrogate modelling approach, and we demonstrate the capabilities of the SPH-based fuel sloshing simulator together with the performance of the identified LPV surrogate in Section 5. Finally, conclusions are drawn in Section 6.

2 Problem definition

2.1 Definition of the fuel sloshing scenario

In this paper, we consider a two-dimensional spacecraft containing a partially filled fuel tank, as illustrated in Fig. 1. The spacecraft is modelled as a rigid body, denoted by $\mathcal{B}$, with *centre of mass* (CoM) located at point G. The fuel tank is rigidly attached to the spacecraft and contains fluid propellant that can move freely within the tank boundaries. A body-fixed reference frame attached to the spacecraft CoM is defined as $\mathcal{R}_{\mathcal{B}} = (G; x_G, y_G)$, while an inertial reference frame is denoted by $\mathcal{R}_{\mathcal{W}}$. Unless otherwise specified, all positions, velocities, and accelerations are expressed in the inertial frame $\mathcal{R}_{\mathcal{W}}$.

The translational motion of the spacecraft at a point Q is described by the position vector $r(t) = [r_{\mathrm{x}}(t)\ r_{\mathrm{y}}(t)]^\top \in \mathbb{R}^2$, and its orientation by the rotation angle $\theta(t) \in \mathbb{R}$. Their time derivatives $\dot{r}(t)$, $\ddot{r}(t)$ and $\dot{\theta}(t)$, $\ddot{\theta}(t)$ denote the corresponding translational and rotational velocities and accelerations. The system is subjected to input forces and torque applied at a point P, representing the action of spacecraft thrusters, and the input vector is defined as $u(t) = [u_{\mathrm{x}}\ u_{\mathrm{y}}\ \tau]^\top(t) \in \mathbb{R}^3$. For simplicity, we assume that P and Q are located at the CoM of the spacecraft, i.e., $P = G$ and $Q = G$. The measured output of the system, expressed with respect to the spacecraft CoM, is then defined as $y(t) = [r_{\mathrm{x}}\ r_{\mathrm{y}}\ \theta\ \dot{r}_{\mathrm{x}}\ \dot{r}_{\mathrm{y}}\ \dot{\theta}]^\top \in \mathbb{R}^6$. The sloshing motion of the fluid induces hydrodynamic forces and torques on the spacecraft, thereby coupling the rigid-body dynamics with the internal fluid motion. The mass and temperature of both the fluid and the satellite are assumed constant.

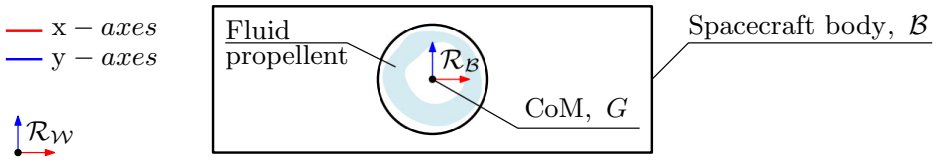

Fig. 1 Sketch of the fuel sloshing scenario considered in this paper, where $\mathcal{R}_\mathcal{W}$ denotes the inertial world reference frame, and $\mathcal{R}_\mathcal{B}$ denotes a local body frame attached at the CoM G of the spacecraft rigid body $\mathcal{B}$.

2.2 Problem formulation and objectives

Based on the fuel sloshing scenario described in Section 2.1, the two complementary challenges identified in Section 1 are formalized as the following objectives.

1) ***High-fidelity fuel sloshing simulator.*** The first objective is the development of a computational model Σ capable of reproducing the coupled spacecraft-fuel sloshing dynamics with high physical fidelity, of the form:

$$\Sigma : \begin{cases} \dot{x}(t) = f(x(t), u(t)), \\ y(t) = h(x(t), u(t)), \end{cases} \tag{1}$$

where $t \in \mathbb{R}$ is time, $x(t) \in \mathbb{X} \subseteq \mathbb{R}^{n_\mathrm{x}}$ with $n_\mathrm{x} \in \mathbb{Z}$ is the state variable, $u(t) \in \mathbb{U} \subseteq \mathbb{R}^3$ the input of the system, $y(t) \in \mathbb{Y} \subseteq \mathbb{R}^6$ is the measured output, corresponding to the translational and rotational motion of the system as defined in Section 2.1. The functions $f : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^3 \to \mathbb{R}^{n_\mathrm{x}}$ and $h : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^3 \to \mathbb{R}^6$ represent the (nonlinear) evolution of the coupled rigid-body and fluid dynamics. For any initial condition $x(t_0) = x_0$ and input trajectory $u(t)$, the solutions of (1) are forward complete and unique for all $t \geq t_0$. The model Σ serves as an efficient data-generation platform for the development and validation of surrogate modelling methods. Therefore, it must be able to rapidly generate data in terms of input, output and state trajectories of the satellite and/or the fluid, as well as local linearizations at arbitrary operating conditions.

2) ***Simulation-oriented fuel sloshing surrogate model.*** For the second objective, we assume that we have collected a data set $\mathcal{D}_N$ from the high-fidelity computational model (1):

$$\mathcal{D}_N = \{(u_k, y_k)\}_{k=0}^{N-1}, \tag{2}$$

where $\{y_k\}_{k=0}^{N-1}$ is the response sampled at time instances $t = kT_\mathrm{s}$, i.e., $y_k = y(kT_\mathrm{s})$, with $k \in \mathbb{Z}^+$ and T_s the sampling time, for a given excitation sequence $\{u_k\}_{k=0}^{N-1}$. We also assume the fluid is initialized in stationary conditions.
Then, based on $\mathcal{D}_N$, the goal is to identify a discrete-time surrogate model S that captures the fuel sloshing system dynamics, of the form:

$$S : \begin{cases} \hat{x}_{k+1} = \hat{f}(\hat{x}_k, u_k, \vartheta), \\ \hat{y}_k = \hat{h}(\hat{x}_k, u_k, \vartheta), \end{cases} \tag{3}$$

where $k \in \mathbb{Z}^+$ denotes the discrete time-step, $\hat{x}_k \in \mathbb{R}^{\hat{n}_\mathrm{x}}$ with $\hat{n}_\mathrm{x} \geq 1$ is the surrogate state, ϑ the surrogate model parameters and $\hat{f} : \mathbb{R}^{\hat{n}_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \times \mathbb{R}^{n_\vartheta} \to \mathbb{R}^{\hat{n}_\mathrm{x}}$ and $\hat{h} : \mathbb{R}^{\hat{n}_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \times \mathbb{R}^{n_\vartheta} \to \mathbb{R}^{n_\mathrm{y}}$ are functions parametrized by ϑ. The surrogate model S aims at: (i) minimizing the criterion

$$J_{\mathcal{D}_N}(\vartheta, \hat{x}_0) = \frac{1}{N} \sum_{k=0}^{N-1} \|y_k - \hat{y}_k\|_2^2, \tag{4}$$

with $\hat{x}_0 \in \mathbb{R}^{\hat{n}_\mathrm{x}}$ as the initial condition, and (ii) the parametrization of $\hat{f}$ and $\hat{h}$ ensures a computationally efficient model evaluation.

3 SPH-based high-fidelity simulator for fuel sloshing with GPU acceleration and automatic differentiation

This section addresses the first objective in Subsection 2.2, namely the development of a high-fidelity and computationally efficient model of the fuel sloshing dynamics.

3.1 Modelling fluid dynamics with the SPH framework

Smoothed-particle hydrodynamics, initially developed by Gingold and Monaghan [20] and Lucy [21] for the simulation of astrophysical problems, is a computational method for simulating the mechanics of continuum media. By construction, it is a mesh free Lagrangian method and therefore it is suited to approach problems like free surface flows. The lack of mesh simplifies the parallelization of computations. These features make the SPH approach ideal for developing efficient simulators of fuel sloshing.

The SPH method approximates continuous media with a set of discrete moving elements, referred to as particles. These particles interact through a radially symmetric kernel function W with compact support radius H and "smoothing length" h. Then, a physical quantity can be approximated at any position r based on the relevant quantities of the other particles. For this, the interpolation of a field quantity A is based on

$$A(r) = \int_{\Omega} A(r')W(\|r - r'\|; h)\mathrm{d}r', \tag{5}$$

where $\mathrm{d}r'$ denotes a differential volume element [22], and in two dimensions the integration domain is the compact support $\Omega = \{\, r' \in \mathbb{R}^2 \mid \exists\, r \in \mathbb{R}^2 \text{ s.t. } \|r' - r\| \leq H \,\}$. If the kernel is normalized, i.e.,

$$\int_{\Omega} W(r)\mathrm{d}r = 1, \tag{6}$$

the interpolation (5) is of second order accuracy [23]. The integral in Eq. (5) is then approximated using a Riemann summation over the particles:

$$A(r) = \sum_j \frac{m_j}{\rho_j} A_j W(\|r_{i,j}\|; h), \tag{7}$$

where m_j, ρ_j, and A_j denote the mass, the density and the field quantity for a particle j, $r_{i,j} = r_i - r_j$, and the summation over j includes all particles[1]. Based on this, the density ρ_i of particle i can be obtained as:

$$\rho_i = \sum_j m_j W_{i,j}, \tag{8}$$

where $W_{i,j}$ is a shorthand notation for $W(\|r_i - r_j\|; h)$. In general, for incompressible isothermal viscous fluids, the SPH acceleration equation is given by

$$m_i \ddot{r}_i = -F_i^{\text{pressure}} + F_i^{\text{viscous}} + F_i^{\text{external}}, \tag{9}$$

where $\ddot{r}_i$ is the acceleration of particle i, and the forces F_i^{pressure}, F_i^{viscous} and F_i^{external} are related to the conservation of momentum, the fluid viscosity and the external forces, respectively. There exist multiple approaches to model these forces in the literature. We choose the following approach for F_i^{pressure}, as, for a fixed h, the linear and angular momentum are exactly conserved [22]:

$$F_i^{\text{pressure}} = m_i \sum_j m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla_i W_{i,j}, \tag{10}$$

[1] In practice, it is sufficient to include the neighbouring particles that lie within the support of the kernel.

where P_i is the pressure of particle i that can be computed with the so-called *state equation* [24]:

$$P_i = k(\rho_i - \rho_0), \tag{11}$$

where k and ρ_0 represent stiffness and rest density constants, respectively, and $\nabla_i W_{i,j}$ denotes the gradient taken with respect to the coordinates of particle i:

$$\nabla_i W_{i,j} = \frac{\partial W_{i,j}}{\partial \|r_{i,j}\|} \frac{r_{i,j}}{\|r_{i,j}\|}, \qquad \nabla_i W_{i,j} = -\nabla_j W_{i,j}.$$

Note that fluids whose pressure can be accurately described by a state equation such as (11) are commonly referred to as *weakly compressible*. Alternative formulations that enforce incompressibility exist [25–28], but these involve computationally demanding iterative procedures.

Lastly, the inclusion of viscous forces involves the discretization of the Laplacian differential operator, but this discretization leads to a poor estimate [24]. Therefore, we make use of the "artificial viscosity" approach introduced in [29]:

$$F_i^{\text{viscous}} = m_i \sum_j m_j \frac{2\,\alpha\, h}{\rho_i + \rho_j} \frac{\dot{r}_{i,j} \cdot r_{i,j}}{\|r_{i,j}\|^2 + \epsilon\, h^2} \nabla_i W_{i,j} \tag{12}$$

where α represents the viscous factor of the fluid, and $\epsilon \sim 0.01$ is introduced to prevent a numerical singularity when $r_{i,j} = 0$.

3.2 Modelling the boundaries of the fuel tank with ghost particles

We model the boundaries of the fuel tank using a particle based strategy, where the geometry of the boundary is populated by a single layer of uniformly distributed *ghost* particles. These ghost particles are incorporated into the computation of field properties of fluid particles, and will serve to define the forces and effects resulting from the interaction between the fluid and the tank walls, as well as to prevent *tunnelling*, i.e., fluid particles penetrating the boundary.

Each ghost particle is then considered to be a static fluid particle with respect to the fuel tank frame $\mathcal{R}_\mathcal{B}$, i.e., their relative position w.r.t. $\mathcal{R}_\mathcal{B}$ does not change over time, replicating the geometry of the boundary. In addition, ghost particles inherit the properties of fluid particles, and the volume of ghost particles must be equal or lower than the fluid counterpart to prevent tunnelling. When including ghost particles, the density computation in Eq. (8) is modified to

$$\rho_i = m_i \left(\sum_{i_f} W_{i,i_f} + \sum_{i_g} W_{i,i_g} + \sum_{i_m} W_{i,i_m} \right), \tag{13}$$

where i_f, i_g and i_m are the indices that refer to fluid, ghost and missing particles, respectively. In particular, missing particles are the ones that lie within the kernel support, but are not included in the single layer realization of the boundary, as illustrated in Figure 2. Therefore, fluid particles close to a boundary miss the contribution from missing particles, consequently underestimating the field quantities. A practical approach to tackle this issue is to encode the contribution from missing ghost particles in a correcting factor γ_1:

$$\rho_i = m_i \left(\sum_{i_f} W_{i,i_f} + \gamma_1 \sum_{i_g} W_{i,i_g} \right), \tag{14}$$

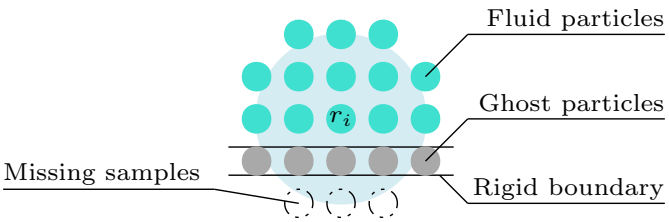

Fig. 2 Modelling of the tank boundary with a single layer of uniformly distributed ghost particles. For fluid particles near the boundary, the computation of field quantities may omit contributions from missing samples, i.e., neighbouring particles lying outside the simulated domain but within the kernel support.

where γ_1 can be analytically estimated [24] as

$$\gamma_1 = \frac{\frac{\rho_i}{m_i} - \sum_{i_f} W_{i,i_f}}{\sum_{i_f} W_{i,i_b}}. \tag{15}$$

Then, the pressure force that a ghost particle j_g exerts onto a fluid particle i_f can be expressed as

$$F^{\text{pressure,g2f}}_{i_f \leftarrow j_g} = 2 m_{i_f} m_{j_g} \frac{P_{i_f}}{\rho^2_{i_f}} \nabla_{i_f} W_{i_f, j_g}, \tag{16}$$

whereas ghost particles inherit the physical properties of the fluid, $m_{i_f} = m_{j_g}$, and the symmetric pressure force from a fluid particle i_f to a ghost particle j_g is

$$F^{\text{pressure,g2f}}_{i_f \leftarrow j_g} = -F^{\text{pressure,f2g}}_{j_g \leftarrow i_f}. \tag{17}$$

Moreover, as proposed in [30], the viscous force between the a fluid particle i_f to a ghost particle j_g can be approximated by

$$F^{\text{viscous,g2f}}_{i_f \leftarrow j_g} = m_{i_f} m_{j_g} \frac{2\beta}{\rho_{i_f} + \rho_{j_g}} \frac{\min\left(v_{i_f j_g} \cdot r_{i_f j_g}, 0\right)}{\|r_{i_f j_g}\|^2 + \epsilon\, h^2} \nabla_{i_f} W_{i_f, j_g}, \tag{18}$$

where β is a coefficient that represents the viscosity between the fluid and the solid boundary, $\rho_{i_f} = \rho_{j_g}$ for the reasons stated above, and the symmetric viscous force from a fluid particle i_f to a ghost particle j_g is

$$F^{\text{viscous,g2f}}_{i_f \leftarrow j_g} = -F^{\text{viscous,f2g}}_{j_g \leftarrow i_f}. \tag{19}$$

3.3 Rigid body dynamics and kinematic coupling between the ghost particles and the spacecraft

To satisfy our first objective, the fuel sloshing simulator must capture the rigid-body dynamics of the spacecraft, including the forces exerted by the fluid on it. The Newton–Euler equations of motion in two-dimensional space [31], expressed with respect to $\mathcal{R}_\mathcal{W}$, are given by

$$\begin{aligned} m\,\ddot{r} &= \sum_{j_g} \sum_{i_f} \left(F^{\text{pressure,f2g}}_{j_g \leftarrow i_f} + F^{\text{viscous,f2g}}_{j_g \leftarrow i_f} \right) + F^{\text{external}}, \\ J\,\ddot{\theta} &= \sum_{j_g} (r_{j_g} - r) \times \left(\sum_{i_f} F^{\text{pressure,f2g}}_{j_g \leftarrow i_f} + \sum_{i_f} F^{\text{viscous,f2g}}_{j_g \leftarrow i_f} \right) + T^{\text{external}}, \end{aligned} \tag{20}$$

Algorithm 1 State transition function $\dot{x}(t) = f(x(t), u(t))$ of the high-fidelity computational fuel sloshing model (1)

Inputs: The state $x(t) = [r \;\; \theta \;\; r_{i_f} \;\; \dot{r} \;\; \dot{\theta} \;\; \dot{r}_{i_f}]^\top(t)$ at any time instance t.
Output: $\dot{x}(t) = [\dot{r} \;\; \dot{\theta} \;\; \dot{r}_{i_f} \;\; \ddot{r} \;\; \ddot{\theta} \;\; \ddot{r}_{i_f}]^\top(t)$.

for each ghost particle j_g **do**
    update the position $r_{j_g,k}$ and velocity $\dot{r}_{j_g,k}$ w.r.t. $\mathcal{R}_\mathcal{W}$ with Eq. (21).
end for
for each fluid particle i_f **do**
    compute the density $\rho_{i_f}(t)$ with Eq. (14).
    compute the pressure $P_{i_f}(t)$ with Eq. (11).
    compute $F_{i_f}^{\text{pressure}}(t)$ and $F_{i_f}^{\text{viscous}}(t)$ with Eqs. (10) and (12), respectively.
    compute $F_{i_f}^{\text{g2f}}(t) = \sum_{j_g}(F_{i_f \leftarrow j_g}^{\text{pressure,g2f}}(t) + F_{i_f \leftarrow j_g}^{\text{viscous,g2f}}(t))$ using Eqs. (16) and (18).
    compute $\ddot{r}_{i_f}(t) = m_{i_f}^{-1}(-F_{i_f}^{\text{pressure}} + F_{i_f}^{\text{viscous}} + F_{i_f}^{\text{external}} + F_{i_f}^{\text{g2f}})(t)$
end for
Compute $\ddot{r}(t)$ and $\ddot{\theta}(t)$ with Eq. (20).
Return $\dot{x}(t) = [\dot{r} \;\; \dot{\theta} \;\; \dot{r}_{i_f} \;\; \ddot{r} \;\; \ddot{\theta} \;\; \ddot{r}_{i_f}]^\top(t)$.

where m and J are the respective mass and inertia of the spacecraft rigid body, and F^{external} and T^{external} are the external force and torque applied at the CoM. The interaction forces $F_{j_g \leftarrow i_f}^{\text{pressure,f2g}}$ and $F_{j_g \leftarrow i_f}^{\text{viscous,f2g}}$ are given by Eqs. (17) and (19), respectively.

While the forces acting on ghost particles due to the fluid are accounted for in (20), ghost particles are not automatically constrained to follow the rigid-body motion of the tank. Therefore, an explicit kinematic coupling is introduced. As described in Section 3.2, the position of ghost particles remain fixed in the body frame $\mathcal{R}_\mathcal{B}$. Let $r_{j_g}^{\mathcal{R}_\mathcal{B}}$ denote the position of the j-th ghost particle expressed in $\mathcal{R}_\mathcal{B}$. Then, the kinematic update of a ghost particle is given by

$$\begin{aligned} r_{j_g} &= R(\theta) r_{j_g}^{\mathcal{R}_\mathcal{B}} + r, \\ \dot{r}_{j_g} &= \dot{r} + \dot{\theta} \times (r_{j_g} - r), \end{aligned} \qquad \text{where} \qquad R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{21}$$

3.4 Fuel sloshing simulation engine

Building upon the formulation introduced in Subsections 3.1, 3.2 and 3.3, we now combine these ingredients to assemble the state transition function $\dot{x}(t) = f(x(t), u(t))$ of the high-fidelity computational fuel sloshing model (1). The state vector is defined as

$$x(t) = \begin{bmatrix} r(t) & \theta(t) & r_{i_f}(t) & \dot{r}(t) & \dot{\theta}(t) & \dot{r}_{i_f}(t) \end{bmatrix}^\top,$$

and the corresponding state transition function is detailed in Algorithm 1. The output mapping $y(t) = h(x(t), u(t))$ of the high-fidelity model (1) is obtained by retrieving the desired quantities of interest (e.g., pose of the tank, fluid-boundary forces or fluid density) directly from Algorithm 1. Lastly, in the SPH literature, symplectic integration schemes are preferred over non-symplectic alternatives for the time integration step of particle states, as they conserve angular momentum and ensure reversibility [22]. Accordingly, we employ a first-order symplectic Euler integration scheme [32, Chapter 6].

3.5 Software implementation details

We leverage the computational capabilities of the open-source software package JAX for PYTHON [33] to implement Algorithm 1 and the first-order symplectic Euler integrator, fulfilling the first objective defined in Subsection 2.2. The proposed implementation offers three main advantages:

1) Algorithm 1 requires iterative computations over all particles. The JAX-based implementation enables parallel execution of these operations on a GPU, such that the computational cost does not significantly increase with the number of fluid particles, up to the limits imposed by the available GPU memory.
2) The use of JAX enables exact and efficient computation of the Jacobian of the state transition function via automatic differentiation.
3) The implementation benefits from just-in-time (JIT) compilation, which translates high-level PYTHON code into high-performance low-level instructions using the *accelerated linear algebra* (XLA) compiler.

To balance numerical accuracy and computational efficiency, a multi-rate simulation scheme is adopted. The fluid and rigid-body dynamics are integrated using a faster sampling rate to accurately capture the evolution of the fluid. In contrast, higher-level operations such as control input updates, data logging, Jacobian evaluations and visualization are performed at a lower sampling rate.

The computation of the correcting factor γ_1 in (15) depends on several aspects, including the position of the fluid relative to the boundary and the local fluid density. In practice, a fixed value for γ_1 is used. This value can either be precomputed for a representative particle configuration reproducing the scenario shown in Fig. 2, or tuned empirically to prevent particle tunnelling during the simulation.

A wide range of kernel functions W has been proposed in the SPH literature. In this work, we employ the cubic spline kernel [22]:

$$W_{\mathrm{cb}}(x;h) = \frac{15}{14\pi h^2}\begin{cases}(2-q)^3 - 4(1-q)^3, & \text{for } 0 \le q \le 1,\\ (2-q)^3, & \text{for } 1 \le q \le 2,\\ 0, & \text{otherwise,}\end{cases} \qquad q = \frac{|x|}{h}, \tag{22}$$

for the computation of Eqs. (14), (10), and (12). Additionally, we use the spiky kernel introduced in [34], which has a non-vanishing gradient near the centre:

$$W_{\mathrm{s3}}(x;h) = \frac{10}{\pi h^5}\begin{cases}(h-x)^3, & \text{for } 0 \le x \le h,\\ 0, & \text{otherwise,}\end{cases} \tag{23}$$

for the computation of Eqs. (16) and (18). Note that both kernels (22) and (23) are normalized for $x \in \mathbb{R}^2$.

4 Learning of an LPV surrogate model

In this section, we address the second objective defined in Subsection 2.2. Specifically, the identification of the model parameters ϑ that minimize the objective (4), together with the selection of a suitable model structure for $\hat{f}$ and $\hat{h}$. To this end, we adopt an LPV-SS model structure with *affine* dependence on the scheduling. This choice is motivated by three main considerations. First, the affine LPV structure retains a computational complexity comparable to that of linear models, making it well suited for rapid simulation. Second, this model structure offers utilization potential for control through the wide range of convex analysis and controller synthesis techniques. Third, the nonlinear scheduling map enables the extraction of complex nonlinear relationships from simulation data. While scheduling variables can be

selected based on engineering insight, such relationships are difficult to derive for fuel sloshing due to the complex fluid–solid interactions. A data-driven scheduling approach is therefore adopted.

Then, the proposed parametrization of the surrogate model (3) is given as

$$S_\vartheta : \begin{cases} \hat{x}_{k+1} = A(p_k)\hat{x}_k + B(p_k)u_k, \\ \hat{y}_k = C(p_k)\hat{x}_k + D(p_k)u_k, \end{cases} \tag{24}$$

where $p_k \in \mathbb{R}^{n_\mathrm{p}}$ is the scheduling variable that is defined through the *scheduling map* $p_k = \eta(\hat{x}_k, u_k, \vartheta_\eta)$, where $\eta : \mathbb{R}^{\hat{n}_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \times \mathbb{R}^{n_{\vartheta_\eta}} \to \mathbb{R}^{n_\mathrm{p}}$ is a parametrized *feedforward neural network* (FNN) representing a nonlinear function , as proposed in [35]. This corresponds to a self-scheduled LPV model. Furthermore, the matrix functions $A, \ldots, D$, collected as

$$M(p_k, \vartheta_M) = \begin{bmatrix} A(p_k, \vartheta_M) & B(p_k, \vartheta_M) \\ C(p_k, \vartheta_M) & D(p_k, \vartheta_M) \end{bmatrix},$$

have an affine dependency on p_k:

$$M(p_k, \vartheta_M) = M_0(\vartheta_M) + \sum_{i=1}^{n_\mathrm{p}} p_k M_i(\vartheta_M), \tag{25}$$

which can also be interpreted as a linear output layer of the FNN η. Here, the parameters ϑ_M are the the elements of the matrices in (25). Therefore, the parameter vector to learn becomes $\vartheta = \mathrm{vec}([M_0, M_1, \ldots, M_{n_\mathrm{p}}], \vartheta_\eta) \in \mathbb{R}^{n_\vartheta}$.

As the given dataset $\mathcal{D}_N$ is directly obtained from the fuel sloshing simulator, the measured data is noise-free. To address the identification problem with the prediction error minimization objective (4), we solve the following optimization proposed in [35]:

$$\begin{aligned} \vartheta^*, \hat{x}_0^* = \arg\min_{\vartheta, \hat{x}_0^*} J_{\mathcal{D}_N}(\vartheta, \hat{x}_0) + R(\vartheta, \hat{x}_0) \quad \text{subject to} \\ \hat{x}_{k+1} = A(p_k, \vartheta_M)\hat{x}_k + B(p_k, \vartheta_M)u_k, \\ \hat{y}_k = C(p_k, \vartheta_M)\hat{x}_k + D(p_k, \vartheta_M)u_k, \\ p_k = \eta(\hat{x}_k, u_k, \vartheta_\eta), \ k \in \{0, \ldots, N\}, \end{aligned} \tag{26}$$

where the loss $J_{\mathcal{D}_N}$ is defined in (4), $\hat{x}_0$ is the initial state and $R(\vartheta, \hat{x}_0)$ is a regularization term given by

$$R(\vartheta, \hat{x}_0) = \frac{\sigma_2}{2}\|\vartheta\|_2^2 + \frac{\sigma_x}{2}\|\hat{x}_0\|_2^2, \tag{27}$$

where $\sigma_2 > 0$ and $\sigma_x > 0$ are the weights corresponding to ℓ_2 regularization of the model parameters and the initial state $\hat{x}_0$, respectively. The optimization (26) is solved with a fixed number of Adam [36] gradient-descend steps that is used to warm-start a limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) scheme [37], and the gradients are computed using automatic differentiation.

5 Results

This section demonstrates the capabilities of the proposed SPH-based fuel sloshing simulator and evaluates the performance of an identified LPV surrogate model in a closed-loop attitude control setting under zero-gravity conditions. First, a benchmark scenario[2] and attitude control setup are introduced.

[2]The parameter values used for the fuel sloshing system are chosen to be physically plausible but not representative of a specific spacecraft, and atmospheric disturbances are neglected.

Table 1 Physical parameters of the considered satellite model.

Parameter	Value	Parameter	Value
Satellite geometry	Rectangular	Fuel tank geometry	Circular
Satellite width (m)	1.0	Satellite CoM (m)	(0, 0)
Satellite height (m)	0.5	Fuel tank CoM (m)	(0, 0)
Satellite mass (kg)	1010.71	Fuel tank inner wall radius (m)	0.2
Satellite inertia (kg $\cdot$ m^2)	133.84	Fuel tank fill ratio (%)	60

Table 2 Physical parameters to model the considered fuel sloshing fluid.

Parameter	Value	Parameter	Value
Number of fluid particles	666	Number of ghost particles	236
Fuel particle length (mm)	6	Ghost particle length (mm)	5.4
Kernel smoothing length h (mm)	9.42	Stiffness coefficient k (-)	3.0
Base density ρ_0 (kg/m^2)	1017	Viscous factor α (-)	8.32×10^{-4}
Boundary viscous factor β (-)	4×10^{-4}	Correcting factor γ_1 (-)	0.5

Second, closed-loop simulations under two distinct manoeuvre profiles are performed. Lastly, based on a separate open-loop excitation dataset, an LPV surrogate model is identified and validated on the closed-loop manoeuvre profiles. The software implementation of the simulator is publicly available at https://gitlab.com/Javi-Olucha/fuel-sloshing-sph-simulator.

5.1 Benchmark scenario and control setup

We consider a spacecraft consisting of a rectangular rigid body containing a circular fuel tank located at the CoM. The geometric and physical parameters of the satellite are summarized in Table 1. The tank is partially filled to 60% with a fluid whose properties are selected to be representative of Hydrazine. The fluid is discretized using 666 particles with a diameter of 6 mm, and a smoothing length of 9.42 mm is used for the SPH kernel functions. The inner boundary of the tank is modelled using a single layer of 236 ghost particles, uniformly distributed along the tank wall. The corresponding fuel sloshing parameters are summarized in Table 2. No gravity or atmospheric disturbance forces are considered. The system inputs and outputs are defined as in Section 2.1, where the thruster forces and torque act at the CoM and the measured outputs correspond to the translational and rotational motion of the satellite CoM.

Both the satellite and the fluid are initialized at rest. To obtain a steady-state fluid configuration, a preliminary simulation is performed in which the fuel particles are spawned randomly distributed within the tank and allowed to evolve without external actuation until their velocities converge to zero. Lastly, for numerical simulation, a time step of $T_s = 0.001$ s is used for integrating the system dynamics, while a time step of $T_s = 0.05$ s is used for measuring, update the control action and update the visualization.

5.1.1 Attitude controller

The angular motion of the spacecraft is regulated by a feedback attitude controller, while the translational motion remains open-loop. A negative feedback controller K is designed following the guidelines in [38], with parameters tuned based on the static inertia of the satellite and the desired closed-loop response. Specifically, the control law is given by

$$\tau_k = K \begin{bmatrix} \tilde{\theta}_k - \theta_k & -\dot{\theta}_k \end{bmatrix}^\top, \quad K = \begin{bmatrix} J\omega^2 & 2\xi J\omega \end{bmatrix}, \tag{28}$$

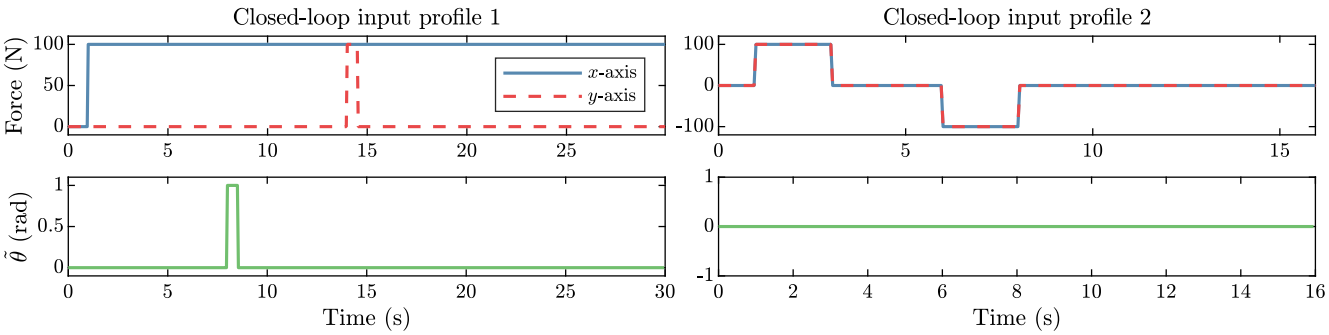

Fig. 3 Manoeuvre profiles used in the closed-loop simulations. The desired angular position $\tilde{\theta}$ of the spacecraft is regulated by the controller K.

where θ_k and $\dot{\theta}_k$ are the sampled angular position and velocity, respectively, $\tilde{\theta}_k$ is the sampled angular reference signal, and J is the static inertia of the satellite. The controller parameters are selected to achieve a closed-loop bandwidth of 0.1 Hz and a damping ratio of 0.7, corresponding to $\omega = 0.2\pi$ and $\xi = 0.7$. Lastly, the controller output is kept constant during the sampling period by a zero order hold.

5.1.2 Manoeuvrer profiles

Two distinct manoeuvre profiles are defined to evaluate the simulator performance and to excite different fuel sloshing dynamics. The first profile consists of a constant thrust along the x-axis, representing a translational acceleration of the spacecraft. The performance of the attitude controller is then evaluated through a reference change in the desired angular position. Next, a pulsed force is applied along the y-axis, representing an external disturbance (e.g., a micro-meteoroid impact), which induces a pendulum-like sloshing motion of the fluid. The second profile consists of pulsed thrust inputs applied along both the x and y-axes, followed by thrust in the opposite direction after a dwell time. This manoeuvre represents a diagonal translation typical of rendezvous operations, and excites a mass–spring–damper-like sloshing response. Both input profiles are illustrated in Fig. 3.

5.2 Closed-loop simulation of the spacecraft under fuel sloshing

The benchmark scenario described in the previous Subsection 5.1 is simulated in closed loop for the two manoeuvre profiles to analyse the performance of the proposed SPH-based fluid simulator and investigate the coupled rigid-body and fluid dynamics under representative operating conditions.

The simulation computation time[3] is reported in Table. 3, and the simulated closed-loop trajectories of the spacecraft are shown in Fig. 9. It can be observed that for both manoeuvre profiles the attitude controller stabilizes the spacecraft effectively. Specifically, the first manoeuvre profile induces a pendulum-like behaviour on the fluid, which is shown via snapshots of the fluid particle distribution in Fig. 4. In contrast, as shown in Fig. 5, the second manoeuvre profile induces a mass–spring–damper-like behaviour on the fluid, interacting with the tank walls during changes in the spacecraft acceleration.

To further analyse the nonlinear nature of the system dynamics, linearizations of the continuous-time nonlinear dynamic model are computed along the simulated trajectories using automatic differentiation. These linearizations are performed on the open-loop system dynamics, i.e., excluding the controller. The eigenvalues of the resulting linearized systems at different time instances are shown in Fig. 6 and Fig. 7 for the first and second manoeuvre profiles, respectively. The observed evolution of the eigenvalues over time highlights the inherently nonlinear nature of the system dynamics, which naturally motivates the use of a linear parameter-varying (LPV) framework.

[3]On a laptop with an i7-13850HX (2.10 GHz) CPU, an RTX2000 Ada (8 GB VRAM) GPU, and 64 GB RAM.

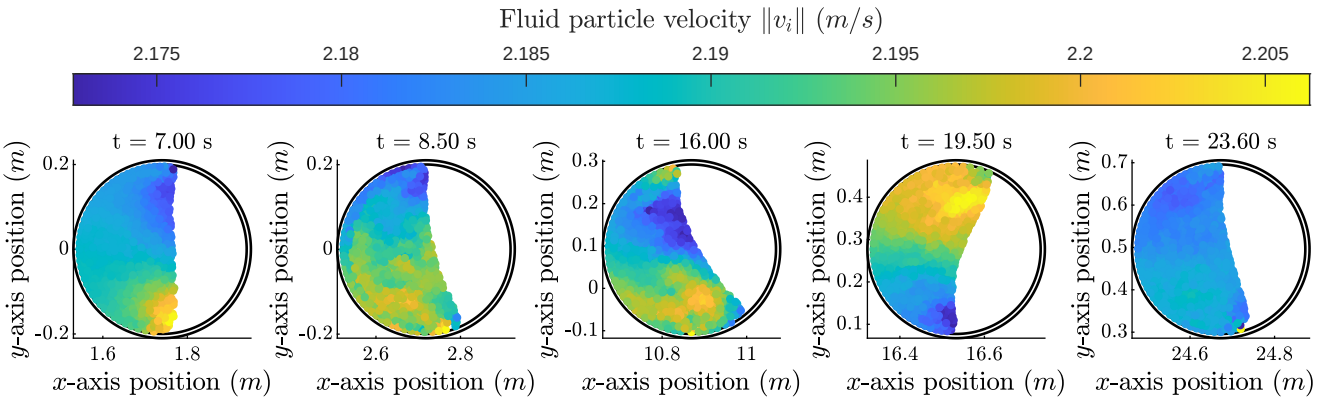

Fig. 4 Visualization of the fluid particle distribution at different time snapshots for the first manoeuvre profile. A pendulum-like sloshing mode is observed following the disturbance along the y-axis. The visualization of the spacecraft rigid body is omitted for clarity. A video of the corresponding simulation is available at https://youtu.be/n0erMbdLwgc.

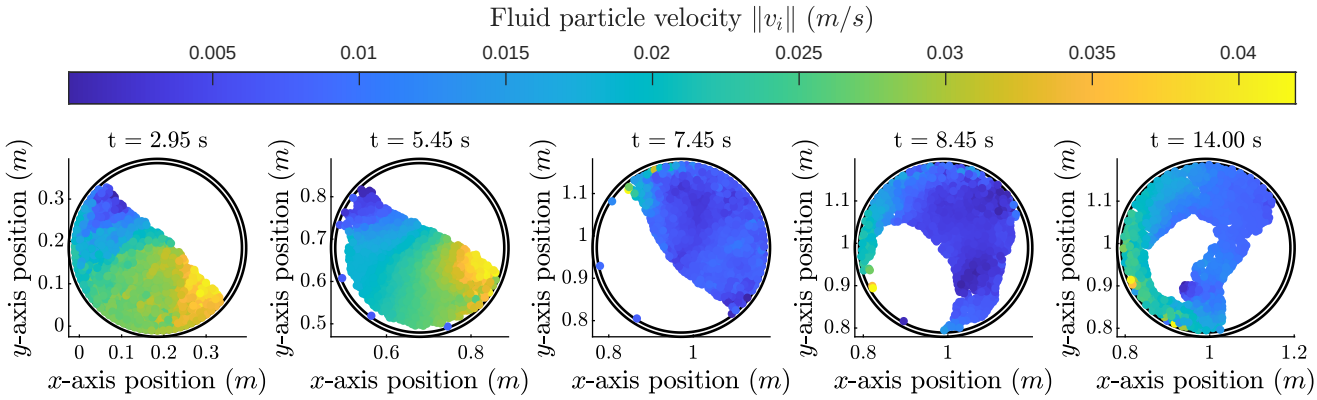

Fig. 5 Visualization of the fluid particle distribution at different time snapshots of the closed-loop simulation for the second manoeuvre profile. The fluid exhibits a mass–spring–damper-like response, interacting with the tank walls during changes in the spacecraft acceleration. The visualization of the spacecraft rigid body is omitted for clarity. A video of the corresponding simulation is available at https://youtu.be/SMiOcJaOFTk.

5.3 Learning an LPV surrogate model

5.3.1 Experiment design

To address the LPV system identification problem, a training dataset distinct from the previously defined manoeuvre profiles is generated. An input train signal, shown in Fig. 8, is designed with a length of 2200 samples and consists of a combination of pulsed inputs and multi-sine signals with random phase, applied independently to each input channel. The pulsed inputs are introduced to excite transient dynamics associated with mass–spring–damper-like sloshing behaviour. In addition, the multi-sine signals cover a frequency range of $[0,\ 2)$ Hz with a resolution of 0.05 Hz, exciting a wide spectrum of the system dynamics. The designed input train signal is applied to the system in open-loop, and the resulting input–output trajectories are collected as the training dataset $\mathcal{D}_N$.

5.3.2 Identification of an LPV surrogate model

Based on the training dataset $\mathcal{D}_N$, our second objective is to identify an LPV surrogate model that provides a computationally efficient approximation of the spacecraft dynamics. The surrogate model should capture the input–output behaviour of the system, relating the applied forces and torque to the

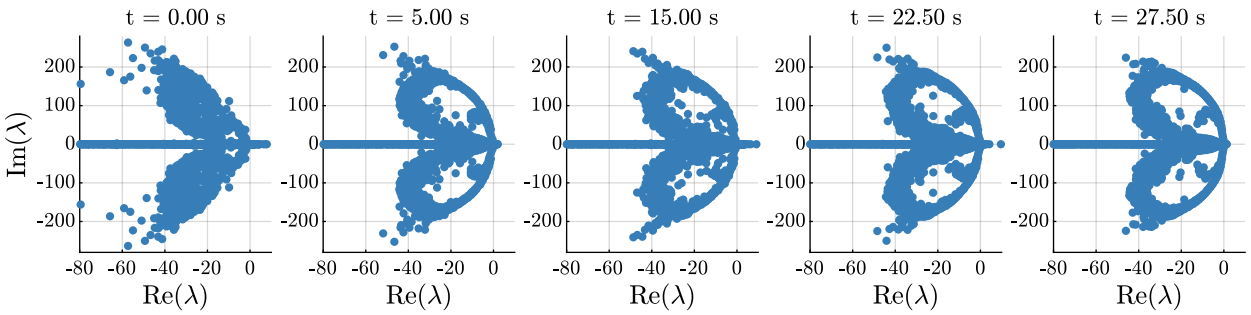

Fig. 6 Eigenvalue evolution of the linearized system at different time instances of the closed-loop simulation for the first manoeuvre profile.

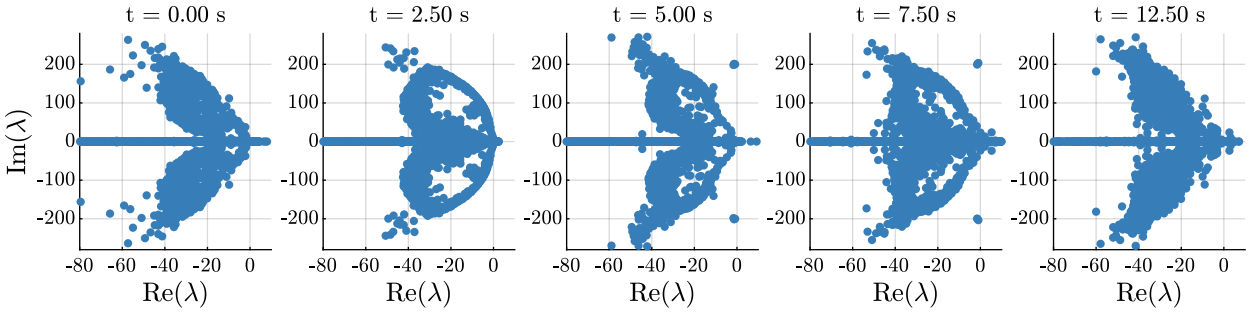

Fig. 7 Eigenvalue evolution of the linearized system at different time instances of the closed-loop simulation for the second manoeuvre profile.

translational and rotational motion of the spacecraft. A direct identification of the full input–output model, including positions and velocities, was found to be challenging. In particular, standard LTI identification approaches were not able to reliably capture the integrator dynamics associated with the position states. To address this issue, the identification problem is reformulated at the velocity level. Specifically, a model that maps the input forces $u(t) = [u_{\mathrm{x}}(t)\ u_{\mathrm{y}}(t)\ \tau(t)]^\top$ to the output velocities $y(t) = [\dot{r}_{\mathrm{x}}(t)\ \dot{r}_{\mathrm{y}}(t)\ \dot{\theta}(t)]^\top$ is identified. The full surrogate model, including position states, is subsequently recovered by augmenting the identified model with integrators.

For this purpose, we proceed to identify a DT LPV–SS model with affine dependence on the scheduling variable, as described in Section 4. The model structure takes the form (24) with sampling time $T_{\mathrm{s}} = 0.05$ s, and dimensions $\hat{n}_{\mathrm{x}} = 4$, $n_{\mathrm{u}} = 3$, $n_{\mathrm{y}} = 3$, and $n_{\mathrm{p}} = 1$ for the state, input, output, and scheduling variables, respectively. The feedthrough matrix is constrained to zero, and the scheduling map $\eta(\hat{x}_k, u_k, \vartheta_\eta)$ is parametrized by a FNN with two hidden layers of four neurons each and tanh activation functions. The resulting LPV model contains a total of $n_\vartheta = 130$ trainable parameters.

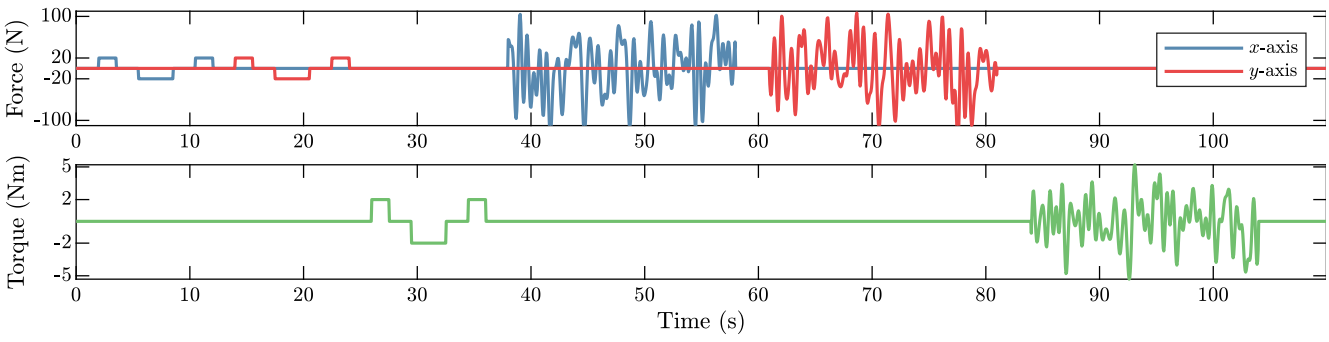

Fig. 8 Input train signal used for system identification. The signal combines pulsed inputs and multi-sine components with random phase, applied independently to each input channel.

Table 3 Closed-loop simulation results. The reported computation time corresponds exclusively to the simulation of the system dynamics, excluding data storage, gradient computation and visualization.

Model	Manoeuvre profile	Computation time (s)	BFR (%)					
			r_x	r_y	θ	$\dot{r}_x$	$\dot{r}_y$	$\dot{\theta}$
SPH simulator	1	9.9093						
	2	5.1876						
Identified LTI	1	0.0038	84.41	-146.95	96.40	71.85	-99.11	83.23
	2	0.0023	45.92	93.02	-436.92	72.66	88.38	-557.13
Identified LPV	1	0.0242	99.52	95.60	98.18	98.94	94.64	90.99
	2	0.0056	99.40	98.62	-66.01	97.17	96.61	-41.68

The LTI component of the model, corresponding to M_0, is initialized using an DT LTI–SS model identified with the SSEST function in MATLAB. For this, the feedthrough matrix is fixed to zero, the estimation of a noise model is disabled, and the identification focus is set to simulation. The identified LTI model achieved an average *best fit rate*[4] (BFR) of 82.16% on the training dataset.

Next, the scheduling-dependent matrices M_i are initialized from a zero-mean normal distribution, and the neural network weights are initialized using the Xavier method [39]. The regularization weights are set to $\sigma_2 = 10^{-4}$ and $\sigma_x = 10^{-6}$. The LPV model parameters are estimated by minimizing (26) with the scaled and normalized dataset, using 2000 iterations of the ADAM optimizer followed by up to 6000 iterations of L-BFGS. The optimization is repeated eight times from different random initial guesses, resulting in a total training time of approximately 48 s. The final model is selected as the one achieving the highest average BFR on the training dataset, yielding an average BFR of 98.66%. Finally, the full LPV surrogate model is obtained by augmenting the identified velocity-level model, resulting in

$$S : \begin{cases} \begin{bmatrix} \hat{x}^{\mathrm{e}}_{k+1} \\ \hat{x}_{k+1} \end{bmatrix} = \begin{bmatrix} I & T_{\mathrm{s}} C(p_k) \\ 0 & A(p_k) \end{bmatrix} \begin{bmatrix} \hat{x}^{\mathrm{e}}_{k} \\ \hat{x}_{k} \end{bmatrix} + \begin{bmatrix} T_{\mathrm{s}} D(p_k) \\ B(p_k) \end{bmatrix} u_k, \\ \begin{bmatrix} \hat{y}^{\mathrm{e}}_{k} \\ \hat{y}_{k} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & C(p_k) \end{bmatrix} \begin{bmatrix} \hat{x}^{\mathrm{e}}_{k} \\ \hat{x}_{k} \end{bmatrix} + \begin{bmatrix} 0 \\ D(p_k) \end{bmatrix} u_k, \end{cases} \tag{29}$$

where I and 0 are identity and zero matrices of appropriate dimensions, and the augmented states $\hat{x}^{\mathrm{e}}_k \in \mathbb{R}^3$ and outputs $\hat{y}^{\mathrm{e}}_k \in \mathbb{R}^3$ correspond to the reconstructed position of the spacecraft.

5.4 Closed-loop validation of the identified surrogate models

The identified LTI and LPV models are validated in closed-loop simulations using the two manoeuvre profiles introduced in Section 5.1. Table 3 summarizes the results in terms of average best fit ratio (BFR) and computation time. The reported computation time corresponds exclusively to the simulation of the system dynamics.

The simulation results, shown in Fig. 9, indicate that the LPV surrogate model consistently reproduces the dominant dynamics of the spacecraft with high accuracy across both manoeuvre profiles, with BFR values close to 100% for most outputs. The lower BFR observed for the angular motion in the second profile is attributed to the small angular rate, on the order of 10^{-4}, which results in a negligible contribution

[4] $\mathrm{BFR} = \left(1 - \sqrt{\frac{\sum_{k=0}^{N} \|y(k) - \hat{y}(k)\|_2^2}{\sum_{k=0}^{N} \|y(k) - y_{\mathrm{mean}}\|_2^2}}\right) \cdot 100\%$, where y is the data sequence, y_{mean} is the sample mean of y, and $\hat{y}$ is the predicted response of the model.

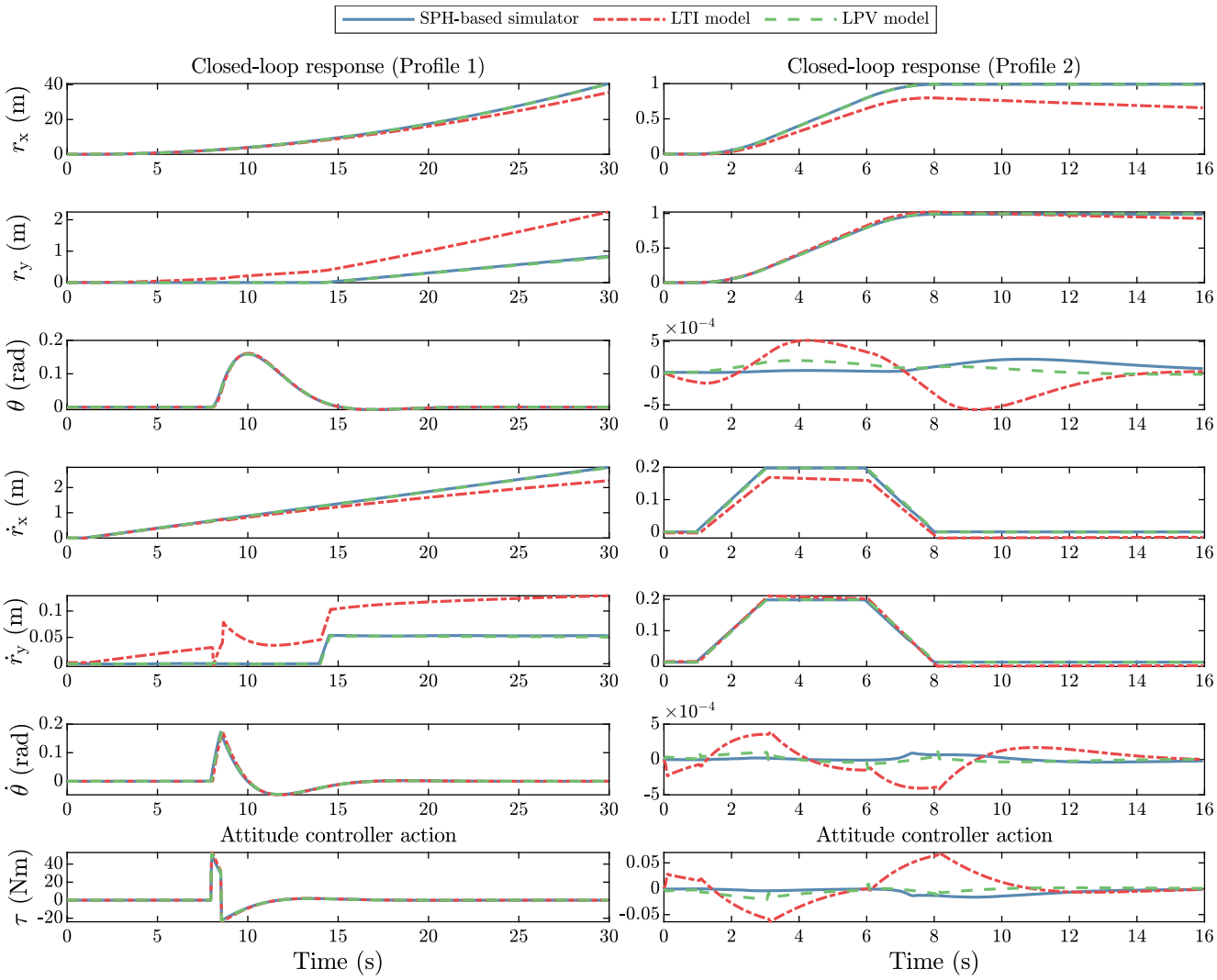

Fig. 9 Closed-loop simulation results between the SPH-based simulator and the identified LTI and LPV models for the two manoeuvre profiles.

to the overall system dynamics. In contrast, the performance of the LTI surrogate model is significantly lower, particularly for manoeuvre profile 2, highlighting the limitations of LTI models in capturing the inherently nonlinear nature of the fuel sloshing dynamics. In terms of computational efficiency, both surrogate models provide a substantial reduction in simulation time compared to the SPH-based simulator. In particular, the LPV surrogate achieves simulation times that are more than two orders of magnitude faster, while maintaining high accuracy. Although the LTI model is faster, this comes at the cost of reduced accuracy.

These results demonstrate that the LPV surrogate model provides a reliable and computationally efficient approximation of the dominant spacecraft dynamics, which makes the proposed LPV surrogate model particularly suitable for applications requiring repeated simulations. In particular, the LPV surrogate can be used to accelerate Monte Carlo simulation campaigns and to rapidly screen or discard unsuitable controller designs before validation with the high-fidelity model.

6 Conclusions

In this paper, we have presented an open-source, high-fidelity SPH-based fuel sloshing simulator capable of rapidly generating informative datasets, including system inputs, outputs, states, and linearized dynamics, to support surrogate model extraction. Leveraging the computational capabilities of JAX, the

proposed implementation exploits GPU parallelization, and the computational cost of the simulator scales favourably with the number of fluid particles. This enables efficient analysis and data generation across a wide range of simulation scenarios. As demonstrated in Subsection 5.2, the simulator reproduces the expected fuel sloshing behaviour of a partially filled tank in a spacecraft under closed-loop attitude control, achieving a computation time below 10 s for a 30 s simulation for a benchmark consisting of 666 fluid particles. Furthermore, we have demonstrated that LPV surrogate models can be used to effectively capture the fuel sloshing dynamics. The identified LPV surrogate model has been validated in closed loop, where it accurately reproduces both pendulum–like and mass–spring–damper-like sloshing modes, while achieving simulation times that are more than 100× faster than the high-fidelity model. For further research, we plan to investigate on leveraging linearized dynamic information during the LPV surrogate extraction process, and on characterizing the surrogate model uncertainty.

Declaration of Use of Artificial Intelligence

We have used learning based methods to estimate models from data with a PYTHON implementation. No other forms of artificial intelligence have been used in the work presented.

References

[1] H. Norman Abramson. The dynamic behavior of liquids in moving containers: With applications to space vehicle technology. Technical Report NASA SP-106, NASA, Washington, D.C., 1966.

[2] J. T. Neer and Jeremiah O. Salvatore. Fuel Slosh Energy Dissipation on a Spinning Body. Technical Report SCG-20047R, Defense Technical Information Center, California, 1972.

[3] Raouf A. Ibrahim. *Liquid Sloshing Dynamics: Theory and Applications*. Cambridge University Press, Cambridge, 2005.

[4] Feng Liu, Baozeng Yue, and Liangyu Zhao. Attitude dynamics and control of spacecraft with a partially filled liquid tank and flexible panels. *Acta Astronautica*, 143:327–336, 2018. doi: 10.1016/j.actaastro.2017.11.036.

[5] Alessia Simonini, Michael Dreyer, Annafederica Urbano, Francesco Sanfedino, Takehiro Himeno, Philipp Behruzi, Marc Avila, Jorge Pinho, Laura Peveroni, and Jean-Baptiste Gouriet. Cryogenic propellant management in space: Open challenges and perspectives. *npj Microgravity*, 10(1):34, 2024. doi: 10.1038/s41526-024-00377-5.

[6] Jan P. B. Vreeburg. Spacecraft maneuvers and slosh control. *IEEE Control Systems Magazine*, 25(3):12–16, 2005. doi: 10.1109/MCS.2005.1432593.

[7] M. Lazzarin, M. Biolo, A. Bettella, M. Manente, R. Da Forno, and D. Pavarin. EUCLID satellite: Sloshing model development through computational fluid dynamics. *Aerospace Science and Technology*, 36:44–54, 2014. ISSN: 12709638. doi: 10.1016/j.ast.2014.03.015.

[8] A. Dalmon, M. Lepilliez, S. Tanguy, A. Pedrono, B. Busset, H. Bavestrello, and J. Mignot. Direct numerical simulation of a bubble motion in a spherical tank under external forces and microgravity conditions. *Journal of Fluid Mechanics*, 849:467–497, 2018. doi: 10.1017/jfm.2018.389.

[9] Alexis Dalmon, Mathieu Lepilliez, Sébastien Tanguy, Romain Alis, Elena R. Popescu, Rémi Roumiguié, Thomas Miquel, Barbara Busset, Henri Bavestrello, and Jean Mignot. Comparison Between the FLUIDICS Experiment and Direct Numerical Simulations of Fluid Sloshing in Spherical Tanks Under Microgravity Conditions. *Microgravity Science and Technology*, 31(1):123–138, 2019. doi: 10.1007/s12217-019-9675-4.

[10] Manuel Hahn, Stefan Adami, and Roger Förstner. Computational modeling of nonlinear propellant sloshing for spacecraft AOCS applications. *CEAS Space Journal*, 10, 2018. doi: 10.1007/s12567-018-0216-6.

[11] Mohammed M. Atif, Sheng-Wei Chi, Emanuele Grossi, and Ahmed A. Shabana. Evaluation of breaking wave effects in liquid sloshing problems: ANCF/SPH comparative study. *Nonlinear Dynamics*, 97(1):45–62, 2019. doi: 10.1007/s11071-019-04927-5.

[12] K. Kotsarinis, M.D. Green, A. Simonini, O. Debarre, T. Magin, and A. Tafuni. Modeling sloshing damping for spacecraft: A smoothed particle hydrodynamics application. *Aerospace Science and Technology*, 133:108090, 2023. doi: 10.1016/j.ast.2022.108090.

[13] P. Enright and E. Wong. Propellant slosh models for the Cassini spacecraft. In *Astrodynamics Conference*, pages 186–195, Scottsdale, AZ, 1994. doi: 10.2514/6.1994-3730.

[14] Jiannwoei Jang. Mechanical Slosh Models for Rocket-Propelled Spacecrafts. In *AIAA Guidance, Navigation, and Control Conference*, 2013. doi: 10.2514/6.2013-4651.

[15] Paolo Gasbarri, Marco Sabatini, and Andrea Pisculli. Dynamic modelling and stability parametric analysis of a flexible spacecraft with fuel slosh. *Acta Astronautica*, 127:141–159, 2016. doi: 10.1016/j.actaastro.2016.05.018.

[16] E. Javier Olucha. High-accuracy pointing control during on-orbit satellite refuelling. Master's thesis, Eindhoven University of Technology, Oct. 2023.

[17] Ricardo Rodrigues, Francesco Sanfedino, Daniel Alazard, Valentin Preda, and E. Javier Olucha. Linear parameter-varying gain-scheduled attitude controller for an on-orbit servicing mission involving flexible large spacecraft and fuel sloshing. In *ESA GNC and ICATT: 12th International Conference on Guidance, Navigation & Control Systems*, 2023.

[18] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake v.d. Plas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018.

[19] Dougal Maclaurin, David Duvenaud, and Ryan P Adams. Autograd: Effortless gradients in numpy. In *Proceedings of the ICML 2015 AutoML Workshop*, volume 238, 2015.

[20] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: Theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375–389, 1977. doi: 10.1093/mnras/181.3.375.

[21] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82(12):1013–1024, 1997.

[22] J. J. Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68(8):1703, 2005. doi: 10.1088/0034-4885/68/8/R01.

[23] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 154–159, San Diego, California, 2003.

[24] Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. Smoothed particle hydrodynamics techniques for the physics based simulation of fluids and solids. In Wenzel Jakob and Enrico Puppo, editors, *Eurographics 2019 - Tutorials*. The Eurographics Association, 2019. doi: 10.2312/egt.20191035.

[25] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible SPH. In *ACM SIGGRAPH*, New York, NY, USA, 2009. Association for Computing Machinery. doi: 10.1145/1576246.1531346.

[26] Kenneth Bodin, Claude Lacoursiere, and Martin Servin. Constraint fluids. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):516–526, 2012. doi: 10.1109/TVCG.2011.29.

[27] Miles Macklin and Matthias Müller. Position based fluids. *ACM Trans. Graph.*, 32(4), 2013. doi: 10.1145/2461912.2461984.

[28] Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):426–435, 2014. doi: 10.1109/TVCG.2013.105.

[29] J. J. Monaghan and R.A Gingold. Shock simulation by the particle method SPH. *Journal of Computational Physics*, 52(2):374–389, 1983. ISSN: 0021-9991. doi: 10.1016/0021-9991(83)90036-0.

[30] Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. Versatile rigid-fluid coupling for incompressible SPH. *ACM Trans. Graph.*, 31(4), 2012. doi: 10.1145/2185520.2185558.

[31] Mark D. Ardema. *Newton-Euler Dynamics*. Springer New York, 1 edition, 2005.

[32] Ernst Hairer, Gerhard Wanner, and Christian Lubich. *Geometric Numerical Integration*. Springer Berlin, 2 edition, 2006. doi: 10.1007/3-540-30666-8.

[33] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[34] Mathieu Desbrun and Marie-Paule Gascuel. Smoothed Particles: A new paradigm for animating highly deformable bodies. In Ronan Boulic and Gerard Hégron, editors, *Computer Animation and Simulation '96*, pages 61–76, Vienna, 1996. Springer Vienna.

[35] Alberto Bemporad and Roland Tóth. Efficient identification of linear, parameter-varying, and nonlinear systems with noise models, 2025.

[36] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[37] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995. doi: 10.1137/0916069.

[38] Nicolas Guy, Daniel Alazard, Christelle Cumer, and Catherine Charbonnel. Dynamic Modeling and Analysis of Spacecraft With Variable Tilt of Flexible Appendages. *Journal of Dynamic Systems, Measurement, and Control*, 136(021020), Jan. 2014. ISSN: 0022-0434. doi: 10.1115/1.4025998.

[39] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proc. 13th Int. Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Italy, 2010. PMLR.