doi: 10.82124/CEAS-GNC-2026-045 CEAS-GNC-2026-045

Machine Learning for Parameter Estimation in CubeSats: A preliminary study for ST3LLARsat1

Jorge J. Ilarraza-Zuazo Department of Aerospace Engineering, Universidad Carlos III de Madrid (UC3M), Leganés, Spain. jorgejavier.ilarraza@alumnos.uc3m.es

Andrés Marcos Department of Aerospace Engineering, Universidad Carlos III de Madrid (UC3M), Leganés, Spain. anmarcos@ing.uc3m.es

ABSTRACT

Machine Learning methods are transforming many aspects of engineering by enabling the design and operation of systems that feature more advanced objectives thanks to the methods' predictive, autonomous, and data-driven capabilities. Cube/Nano-satellites are a type of aerospace system especially positioned to benefit from the use of these methods, given their very limited budget and onboard computational power. This article presents the application of the technique known as Compressed Sensing to the estimation of the inertial parameters of UC3M's ST3LLARsat1 "Boira" CubeSat during the detumbling phase. Its physical characteristics, detumbling controller, sensor/actuator models, and orbital parameters are implemented and simulated on an open-source CubeSat simulator, and a Monte Carlo campaign is performed randomly varying the initial conditions. The results are very promising, with a maximum estimation error of 1%.

Keywords: Machine Learning, Compresses Sensing, Attitude Estimation, CubeSats

1 Introduction

Recent developments in the field of Artificial Intelligence (AI) have greatly encouraged the use of Machine Learning (ML) and data-driven techniques in all technical fields. Engineering has not been different in this regard, as ML has allowed for improved efficiency, autonomy, and adaptability in many applications. This comes as a consequence of its predictive ability, which can be used to make decisions or find intricate patterns from very complex sets of data, even in real time.

Within engineering, control systems engineering is one of the disciplines that can benefit the most from these advances. There are many possible approaches to the design of controllers, their architectures, and their implementations, and it is often not trivial to determine which, if any, is the ideal/optimal configuration for the real, nonlinear system. Part of the problem is that accurate modeling of the system, and of the physics that concern it, is often times very difficult, but paramount in the development of successful controllers. In addition, there is a need to have tools and methods that allow to optimize performance versus robustness as measured on board.

It is precisely in these areas where it is clear that the use of ML could have a significant impact. It could allow to develop more accurate and flexible models of complex systems, which often involve nonlinearities and uncertainties that may be challenging to describe using traditional methods. Its optimization capabilities might, meanwhile, make it possible to tune control parameters for improved efficiency or performance, and its ability to find patterns in large datasets could be useful in informing the

design of controllers by analyzing prior behavior. ML could also enable more adaptive control systems that can continuously learn and adjust their functioning based on real-time sensor data and feedback. Such systems could autonomously adapt to changes in operating conditions, disturbances, or their own dynamics, leading to improved robustness and performance.

Due to the limited economic budget that commonly accompanies the development of CubeSats, the limits in computational capabilities as a result of the use of COTS, and their restricted size/weight/power, they are prime candidates to make use of ML. These methods could be applied for the design of the attitude control system, for the detection of anomalies during operation, or for the processing of (operational or scientific) data collected as part of their mission. It is also expected that ML and data-driven techniques could enable better online performance-vs-robustness trade-offs or open up new capabilities, which were previously not attainable for CubeSats.

This work presents the application of the ML technique known as Compressed Sensing (CS) to a CubeSat, specifically to ST3LLARsat1 "Boira", which is UC3M's first student-professor CubeSat. The layout of the article is as follows. Section 2 presents the basic dynamical equations for satellite dynamics and detumbling control; Section 3 covers the fundamentals of CS and signal sparsity; Section 4 presents ST3LLARsat1's CubeSat general characteristics and orbit parameters, as well as the open-source simulator used; Section 5 describes the implementation of the CS method as developed for the chosen study case and the results; and finally, Section 6 the conclusions.

2 Satellite Dynamics

2.1 Orbital and Rotational Dynamics

All of the equations, as well as the orbital mechanics theory discussed in this section have been extracted from [1], up to the modified formulation for the use of quaternions.

On a first approach, the orbiting of a satellite around Earth can be modelled as a two-body problem, where the only forces present are those generated by the gravitational interaction between Earth and the satellite. Written in the Earth-Centered Inertial (ECI) reference frame, the equation that governs the movement of the satellite around Earth, as derived from an equilibrium of forces, is given by Eq. (1):

$$ {}_0\ddot{r}_{01} = -\frac{\mu \cdot {}_0r_{01}}{\|{}_0r_{01}\|^3} \tag{1} $$

In the notation used, e.g. ${}_0a_{01}$, the two subscripts on the right ("0" and "1") indicate respectively, that a is the vector originating at the center of the ECI frame (termed as "0") and ending at the origin of some other reference frame (termed as "1"). The left subscript indicates the basis of the reference frame used to write the vector's elements (in the exemplified case, the ECI frame "0"). Thus, with respect to Eq. (1), r is the ECI position of the satellite, while μ the well-known gravitational parameter defined as in Eq. 2 with m_1 and m_2 respectively indicating the mass of Earth and of the satellite (with the latter negligible with respect to the former):

$$ \mu = G \cdot (m_1 + m_2) \tag{2} $$

The most common representation for the orbit of a spacecraft around Earth, and its position within that orbit, is done through the six classical orbital parameters, also known as Keplerian elements. These consist of the semi-major axis, denoting the average distance between the centers of mass of the bodies; eccentricity, measuring the deviation of the orbit's shape from a circle; inclination, defining the tilt of the orbital plane relative to a reference plane (often the equatorial or ecliptic); longitude of the ascending

node, specifying the position where the orbit crosses the reference plane (if using Earth's equatorial, it is known as right ascension of the ascending node, RAAN); argument of periapsis, indicating the orientation of the orbit's closest approach to the central body; and mean anomaly, representing the angular distance of the object from the periapsis at a given time.

As for the attitude of the spacecraft, the main law determining its dynamic evolution in time is known as Euler's Rotation Equation or Rigid Body Dynamics Equation. In its compact form is given by:

$$I\dot{\omega} + \omega \times I\omega = T \tag{3}$$

where I and T are respectively, matrices of moments of inertia and applied torques, while ω is the angular rate (velocity) vector. The kinematics are accounted for by the Kinematics Differential Equation, which from ECI to another reference frame are given by equation (4):

$$\dot{R}_{03} = -{}_3\omega_{03} \times R_{03} = -{}_3\hat{\omega}_{03} R_{03} \tag{4}$$

Here, R_{03} would be the rotation matrix from ECI to what is defined as the body-centered reference frame (subscript "3"), whose axes are aligned with its principal moments of inertia. The term $\hat{a}$ represents the matrix that gives the cross product of a with another vector.

Three rotations are needed in order to fully define the body frame from ECI. Hence, the angular rates can be decomposed into the following vector sum:

$${}_3\omega_{03} = {}_3\omega_{01} + {}_3\omega_{12} + {}_3\omega_{23} \tag{5}$$

If the rotations are chosen arbitrarily to be in the ZXZ form, meaning that from reference frames "0" to "1" a rotation is performed around Z, then from "1" to "2" by a rotation around X, and finally from "2" to "3" around Z, then by substituting the corresponding rotation matrix for every step, equation (5) becomes:

$${}_3\omega_{03} = \begin{bmatrix} \sin(\theta_2)\sin(\theta_3) & \cos(\theta_3) & 0 \\ \sin(\theta_2)\cos(\theta_3) & -\sin(\theta_3) & 0 \\ \cos(\theta_2) & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \tag{6}$$

where each θ is one of the three rotation angles. To find their derivatives from the rate vector, this can be rearranged into:

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = \frac{1}{\sin(\theta_2)} \begin{bmatrix} \sin(\theta_3) & \cos(\theta_3) & 0 \\ \sin(\theta_2)\cos(\theta_3) & -\sin(\theta_2)\sin(\theta_3) & 0 \\ -\cos(\theta_2)\sin(\theta_3) & -\cos(\theta_2)\cos(\theta_3) & \sin(\theta_2) \end{bmatrix} {}_3\omega_{03} \tag{7}$$

However, it is apparent that this results in a singularity whenever $sin(\theta_2) = 0$. This is always the case when using Euler angles, no matter if a rotation sequence different to ZXZ is used. To avoid this problem (known as Gimbal Lock), most spacecraft attitude computations are done with quaternions, which avoid the problem by having four terms for the three angles.

$$Q = (q_0, q) = (q_0, q_1, q_2, q_3) \tag{8}$$

Without getting into their strict mathematical definition, they are related to the rotation matrix by the following equations (9):

$$\begin{cases} q_0 = \pm\frac{1}{2}\sqrt{\text{trace}(R)+1} \\ q = \frac{1}{4q_0}\begin{bmatrix} R(2,3)-R(3,2) \\ R(3,1)-R(1,3) \\ R(1,2)-R(2,1) \end{bmatrix} \end{cases} \tag{9}$$

Using these definitions, and rearranging equation (4) into (10), an expression for the angular rate vector as a function of the quaternion can be found.

$${}_3\hat{\omega}_{03} = \begin{bmatrix} 0 & {}_3\omega_{03}(3) & {}_3\omega_{03}(2) \\ {}_3\omega_{03}(3) & 0 & {}_3\omega_{03}(1) \\ -{}_3\omega_{03}(2) & {}_3\omega_{03}(1) & 0 \end{bmatrix} = -\dot{R}_{01}R_{01}^T = f(Q,\dot{Q}) \tag{10}$$

Finally:

$$\begin{cases} \dot{q_0} = -\frac{1}{2}{}_3\omega^T{}_{03}q \\ \dot{q} = \frac{1}{2}{}_3\omega_{03}q_0 - {}_3\hat{\omega}_{03}q) \end{cases} \tag{11}$$

Which coupled with (3) through (9) solves the orientation dynamics of a spacecraft for a given torque.

2.2 B-Dot Control for Detumbling

Typically, to determine its attitude, a spacecraft will incorporate an angular-rate sensor. The most common for CubeSats are gyroscopes, also known as Inertial Measurement Units (IMUs), or magnetometers (MTMs). The latter measure Earth's magnetic field, but as it is locally constant and there is knowledge of its vector change in the body reference frame, then this is equivalent to knowing the body's angular-rates. However, it is worth noting that MTMs are affected by any residual magnetic dipole in the spacecraft. An alternative is to use photodiodes, which can serve as a rough Sun-sensor for orientation when in daylight.

When relaying on MTMs as sensors and magnetorquers (MTQs) as actuators, as is the case for most CubeSats, the main form of satellite control is known as B-dot control. It allows to perform detumbling after deployment from the launcher in a very simple and cost-effective manner.

Detumbling is necessary as most often than not, the spacecraft is deployed with a (relatively high) spinning rate around all axes, which must be reduced to within specific tolerances before the spacecraft can commence nominal operations. MTQs consist of a set of magnetic coils that produce a magnetic field when electrical current is passed through them. This field interacts with Earth's magnetic field resulting in a torque on the satellite that is able to alter its angular rates. [2]

If a CubeSat is equipped with three MTQs, one on each axis, then it has full control of the direction and magnitude of their total induced dipole. The resulting torque will always be normal to the plane containing this dipole and the local Earth magnetic field, as per the cross product in equation (12), where T is the torque, m_d the magnetorquer-induced dipole, and B the local magnetic field. All vectors are given in the body reference frame.

$$T = m_d \times B \tag{12}$$

This has the effect of prohibiting a spacecraft in, for instance, an equatorial orbit from ever generating a torque in the North to South direction, since the Earth's magnetic field lines will always lie along it.

In B-dot control, a proportional law is introduced to manage the magnetic dipole created by the MTQs. According to it, the dipole should be proportionally opposite to the derivative of the magnetic field measured in the body frame, as in equation (13):

$$m_d = -K_p \dot{B} \tag{13}$$

The derivative of the Earth's magnetic field in the body frame can be derived from the measurements by the MTMs or from pre-loaded look-up tables plus orbital position knowledge. The calculation of the derivative based on knowledge of the magnetic field and measurement of the angular speed is given by (14):

$$\dot{B} = \frac{B \times \omega}{\|B\|^2} \tag{14}$$

Substituting into equation (13), it yields the following expression for the magnetorquer dipole:

$$m_d = -\frac{K_p}{\|B\|^2}(B \times \omega) \tag{15}$$

The current passing through each of the magnetorquers present on the satellite can then be adjusted accordingly so as to create this dipole. The resulting torque is given by equation (16) [3]:

$$T = -\frac{K_p}{\|B\|^2}(B \times \omega) \times B \tag{16}$$

Expanded with the triple vector product identity, the latter becomes Eq. (17):

$$T = -\frac{K_p}{\|B\|^2}\left[-(B \cdot \omega)\, B + (B \cdot B)\, \omega\right] = K_p \left(\frac{B \cdot \omega}{\|B\|^2} B - \omega\right) \tag{17}$$

This expression can be substituted into equation (3) to find an explicit form of the angular accelerations, as a function of the angular rates, moments of inertia, and magnetic field under B-dot control. First, the projection of equation (3) into the principal axes of inertia results in equations (18):

$$\begin{cases} I_x \dot{\omega}_x + (I_z - I_y)\omega_y \omega_z = T_x \\ I_y \dot{\omega}_y + (I_x - I_z)\omega_z \omega_x = T_y \\ I_z \dot{\omega}_z + (I_y - I_x)\omega_x \omega_y = T_z \end{cases} \tag{18}$$

And by substitution of the B-dot torques, this develops into the following set of explicit equations (19):

$$\begin{cases} \dot{\omega}_x = \frac{K_p}{I_x}\left(\frac{B \cdot \omega}{\|B\|^2} B_x - \omega_x\right) - \frac{I_z - I_y}{I_x}\omega_y \omega_z \\ \dot{\omega}_y = \frac{K_p}{I_y}\left(\frac{B \cdot \omega}{\|B\|^2} B_y - \omega_y\right) - \frac{I_x - I_z}{I_y}\omega_z \omega_x \\ \dot{\omega}_z = \frac{K_p}{I_z}\left(\frac{B \cdot \omega}{\|B\|^2} B_z - \omega_z\right) - \frac{I_y - I_x}{I_z}\omega_x \omega_y \end{cases} \tag{19}$$

3 Compressed Sensing

This section explains the fundamentals behind Compressed Sensing (CS) and signal sparsity following reference [4], while the specific implementation used is described in Section 5.

CS challenges traditional signal processing theory by asserting that a signal can be accurately reconstructed from fewer samples than the Nyquist-Shannon sampling theorem states, under the condition that the signal is sparse or compressible in a certain domain. Being able to recreate a signal with less samples is useful in that it enables more efficient data acquisition and storage, without sacrificing information quality. This capability is particularly advantageous in situations where data collection is resource-expensive or time-consuming as is the case in CubeSats.

At its heart, the problem of trying to reconstruct a signal is a problem of solving an under-determined system. Thar is, to find a partial measurement y from a full signal x, via a compression matrix C:

$$y = Cx \tag{20}$$

Since y has less rows than x due to the compression, there exists a basis Θ such that:

$$y = \Theta s \tag{21}$$

if s is a sparse vector, meaning that it has only a few non-zero elements.

This is true when the signal in question is known to be constrained to be of a certain type, and in these cases, an appropriate "library" of functions Θ can be constructed so that the problem is solvable. For instance, if the signal were a picture taken with a camera, it is known that it will meet certain constraints on continuity, smoothness, etc. as opposed to an equivalent signal generated from a set of random pixels. This is what allows most images to be highly compressed while retaining most of their quality.

Further, since the system is under-determined, there are many combinations of functions that could reconstruct the measured compressed signal, and the objective becomes that of finding the fewest number of functions that do so for a desired accuracy.

Sparsity in the solution is desirable for multiple reasons. It facilitates interpretability, since only a small subset of predictors or functions are empathized (only those that have the most effect on the outcome or signal). It also makes for a more efficient reconstruction, requiring less storage and computing power since only a small subset is used. And it also avoids overfitting while improving generalization, making the model more robust and adaptable. Ideally, s being the best sparse solution would imply the following:

$$s \mid \min \left(\|\Theta s - y\|_2 + \lambda \|s\|_0 \right) \tag{22}$$

Here, s is such that it accurately reconstructs the measured signal y, since the standard (L2, Euclidean) norm is minimized. And it does so while being sparse by a regularization factor λ, since this last term minimizes the L0 norm of s. This L0 norm simply counts the total number of non-zero elements in the vector, and hence it is small for a sparse vector.

An issue is encountered in trying to solve the minimization problem of (22). This turns out to be an NP-hard, or combinatorially-hard, problem because it involves an exhaustive search over all possible subsets of variables in order to find the sparsest solution. The time it takes to solve this problem grows exponentially with the dimensionality of the data, resulting in a computationally intractable problem.

Instead, the L0 norm behavior can be approximated by the L1 norm. L1 regularization, which is also known as Lasso (Least Absolute Shrinkage and Selection Operator), is a good proxy for L0-regularization because it promotes sparsity while remaining computationally tractable. Although the L1 norm, which

in fact gives the sum of absolute values of the elements, does not directly account for vector sparsity (like the L0 norm does), it has the similar effect (to the L0 norm) of encouraging sparsity in the solution. This is due to the fact that, unlike the L2 norm, which is quadratic, L1 has an absolute-value geometry that encourages many coefficients during minimization to be exactly equal to zero and not just smaller. This enables to achieve sparsity.

Formulated using the L1 norm, the above equation (22) becomes (23):

$$s \mid \min \left(\|\Theta s - y\|_2 + \lambda \|s\|_1 \right) \tag{23}$$

This problem is now a convex optimization, which is much easier to solve than that for the L0 case. Many algorithms exist to solve such types of problems. Furthermore, this formulation can also be implemented in cases where y is not just a compressed measurement, but also a noisy one, since a sparse representation still exists with sufficient signal-to-noise-ratio (SNR). L1 regularization also possesses desirable theoretical properties such as stability, consistency, and the ability to perform variable selection.

Nevertheless, there are some requirements imposed upon the measured signal y in order to make use of the CS approach. Random sub-sampling must be used in the compression while retaining precise knowledge of the position of the measurements –this is equivalent to knowing the timing of the samples in a dynamical system. This is necessary so that the regression can make use of the denser-sampled regions to gain insight on y. A periodic, uniform distribution of measurement samples, as is usually done for many practical system/parameter identification approaches, will not be compatible with CS.

However, it is precisely this random sub-sampling step the one that enables CS to surpass the restrictions of the Nyquist–Shannon sampling theorem. In traditional, uniformly measured, discrete-time signals, this theorem imposes that the frequency of the sampling should be performed at twice the rate of the largest frequency that will later be desired to reconstruct, equation (24):

$$f_{sampling} \geq 2 f_{signal_{max}} \tag{24}$$

If this is not done, aliasing can occur, making the reconstructed signal feature a prominent lower-frequency component than the real one. CS meanwhile requires only that the samples be dispersed randomly in time, with the only condition being that the number of samples should be of a certain order of magnitude depending on the sparsity desired, and according to equation (25):

$$p \propto O\left(k \log\left(\frac{n}{k}\right)\right) \quad \text{for k-sparse } s \tag{25}$$

where p is the number of random samples, k the number of nonzero elements in s, and n the discrete dimension of the original full signal.

4 CubeSat Model and Detumbling SW Simulator

4.1 CubeSat Model: UC3M's ST3LLARsat1 "Boira"

ST3LLARsat1 "Boira" is the first student-professor CubeSat program established at UC3M. Its development is vertebrated within UC3M's Master in Space Engineering (MISE) program and the CubeSat itself is a two-units (2U, i.e. a 20x10x10cm) spacecraft whose mission consists of measuring the water vapor content along its ground track using a COTS spectrometer. These scientific observations will contribute to existing climate-change-monitoring databases, compensating its lack of scientific data

quality (compared to larger Earth Observation (EO) spacecraft) with a higher revisit time (due to the mission's lower Earth orbits, in the range of 400-600 kms).

The ST3LLARsat1 program started in September 2022 and a few months later was one of five European university teams selected by the European Space Agency's (ESA) for its 1.5 years' "Fly Your Satellite!" (FYS) pilot Design Booster program. Currently, ST3LLARsat1 is entering the Assembly, Integration, Validation, and Testing (AIVT) phase, with an expected launch-ready date by late 2026/27.

Regarding its Attitude Determination and Control System (ADCS), ST3LLARsat1 includes: (i) a miniature MEMS IMU gyroscope sensor used for angular rate determination; (ii) 15 photodiodes, located around the solar panels, that serve as a rough attitude Sun-based sensor; (iii) a magnetometer sensor able to measure the magnitude and direction of the local magnetic field; and (iv) a three-axis set of magnetorquers (two rods and one coil) that enable the CubeSat to control its angular rates and orientation. A B-dot control algorithm is used to actuate the MTQs during the detumbling phase of the mission, with two main control modes: a "simple" B-dot, which takes only readings of the magnetometer, and a "modified" B-dot, which, when enough power is available, also uses the measurements from the gyroscope/IMU sensor. The ADCS subsystem runs on a 10 Hz operating update frequency (sample rate).

The values for the nominal Moments of Inertia (MoI) and mass of ST3LLARsat1 at ESA's FYS! Design Booster's Final Design Review (FDR, on June 2024) are given in Table 1:

MoI	I_{xx}	I_{yy}	I_{zz}	**Mass**	m
$[kg \cdot m^2]$	0.0046	0.0046	0.00145	$[kg]$	1.366

Table 1 ST3LLARsat1: moments of Inertia and mass at ESA FYS-DB FDR

4.2 CubeSat Detumbling Simulator (CDSim)

While developing the actual high-fidelity simulator for ST3LLARsat1, and in order to facilitate undergraduate and master thesis' studies, an alternative CubeSat, open-source simulator was selected. Various options were considered and evaluated according to their complexity, number of orbital sources, magnetic perturbations, flexibility to introduce changes, modelling capabilities, and integrability with external (MATLAB) code –e.g. for the CS algorithm.

The selected software is called "CubeSat Detumbling Simulator" (termed from now on as CDSim), and was developed by Samir A. Rawashdeh [5]. CDsim is an open-source, purpose-built, CubeSat simulator dedicated to the detumbling phase that was originally intended for use by the University of California, Irvine (UCI) CubeSat team. It models a LEO, 2U, 3-axis magnetorquer-equipped CubeSat, featuring also magnetometer and gyroscope systems, and using B-dot control. This coincides in great part with ST3LLARsat1 configuration (as well as most 1U-3U CubeSats, which for reasons of space and cost, have this type of actuation/sensing suites and typically fly in LEO orbits between 300 to 600 kms of altitude). CDSim also includes a real-time user interface to visualize the detumbling process in-orbit, and a pre-calculating mode that avoids delays due to graphics updates and only displays results after the simulation has been performed. Although CDsim uses some simplifying assumptions and implements a basic integration method (RK-1) to prioritize computational speed over accuracy, it was deemed sufficient for the purposes of this preliminary application work.

In CDSim the user must define the CubeSat's physical, equipment, and orbital parameters, as well as initial conditions. The nominal principal moments of inertia are those of table 1 for ST3LLARsat1. Those for the equipment used the default values for UCI's CubeSat (see code in CDSim for the exact values), which include: for each axis of the MTQ unit (the gain, number of coils, section area, and maximum current); for the magnetometer and gyroscope sensors (the gain, sample rate, bias, and noise

amplitudes). The gains for the MTQ were modified to match the performance of ST3LLARsat1, which is expected to require about two orbital periods to detumble.

In this work, a generic orbit was used defined by the nominal orbital parameters given in Table (where h is the orbital altitude, e the eccentricity, i the inclination, ω the argument of periapsis, and Ω the right ascension of the ascending node):

4.2

Parameter	$h\ [km]$	$e\ [-]$	$i\ [deg]$	$\omega\ [deg]$	$\Omega\ [deg]$
Value	600	1	97.787	144.8873	59.4276

Table 2 Nominal orbital parameters

Finally, the initial conditions used for the angular rates are given in table 3:

Angular rates	ω_x	ω_y	ω_z
$[deg/s]$	30	30	30

Table 3 Nominal Initial Angular Rates

With the above configuration, and after the user defines the number of orbits to be simulated, CDSim calculates the number of iterations needed and fixes this value from the start. For each iteration, it calculates the orbital position, spacecraft orientation, and angular rates.

It is important (for the subsequent application of CS) to understand how the estimation of the magnetic field vector in the body reference frame, $\hat{B}$, and of the gyroscope readings, $\hat{\omega}$, are performed.

The magnetometer makes its reading using Eq. (26) where $G_{LP_{mag}}$ is a non-dimensional low-pass filter gain, B_{real} is the actual body-frame field, $e_{bias_{mag}}$ represents a fixed additive error, and $e_{noise_{mag}}$ is a random error (generated at each iteration):

$$\hat{B} = (1 - G_{LP_{mag}})(B_{real} - e_{bias_{mag}} + e_{noise_{mag}}) \tag{26}$$

The value of B_{real} is obtained by rotating the output of the magnetic field in the ECI frame obtained using the World Magnetic Model (WMM) 2020 standard [6].

The gyroscope reading is similarly given, see equation (27):

$$\hat{\omega} = (1 - G_{LP_{gyro}})(\omega_{real} - e_{bias_{gyro}} + e_{noise_{gyro}}) \tag{27}$$

Using the estimated magnetic field the magnetorquer actuation is calculated using first equation (14) to get the dipole and then equation (12) to compute the resulting torque. However, the current needed for this operation is also calculated, and if it is over the maximum defined for the CubeSat, the magnetorquer dipole is saturated accordingly.

Using the above nominal configuration, the evolution of the angular accelerations on each of the principal axes of the spacecraft is shown in figure 1.

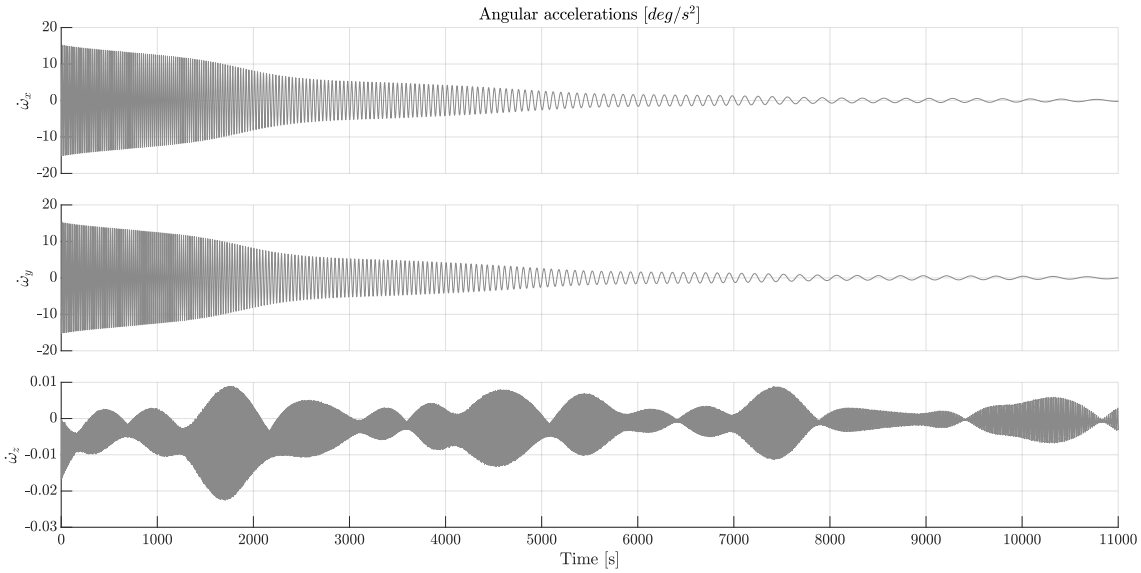

Fig. 1 Angular accelerations of the CubeSat in the nominal case

Note that the resulting accelerations are remarkably similar in the x and y axes, whereas the z axis remains near zero throughout the entire detumbling process. This coincides with the behavior that is expected of the spacecraft, as it is symmetric and the body-frame is chosen for the principal inertial axes.

CDsim was modified to be able to run Monte Carlo campaigns, the parameters considered are namely the CubeSat's moments of inertia, initial angular rates, and orbital parameters. As implemented, the code generates an array that contains in each entry the randomized values for the n chosen uncertainty parameters, $\tilde{a}_i$ with $i = 1 \ldots n$, each constructed by adding a random amount of uncertainty to the nominal value, a, following equation (28):

$$\tilde{a}_i = a_i \cdot (1 + \delta_{a_i} \cdot \sigma_{a_i}) \tag{28}$$

Note that δ_{a_i} is a randomly-generated value in the range $[-1, 1]$ and σ_{a_i} is a factor determining the range of the added uncertainty for each i parameter.

5 Compressed Sensing Estimation - Implementation and Results

The nominal CubeSat simulation configuration given in the tables above was used to train the CS model and subsequently, 20 random cases from the Monte Carlo SW utility were used to assess the resulting CS estimation performance and robustness.

Following the CS description from section 3, the first step in applying CS is to select the input data vector y, and the 2^{nd} step is to define a library of functions for this vector (the basis Θ from Eq. 3). The purpose of this library is to be able to relate, via the s sparse representation of the compressed signal, the (typically) nominal input data used for the CS training with the data from the measurements used later on. The matrix Θ is often defined using all possible permutations of the variables in the system under consideration following physical engineering understanding (for instance, up to second order as in [7]), but since the full behavior of the dynamics is not known a priori, other possibilities can be explored, such as increasing the order of the permutations, adding trigonometric or exponential functions . . .. The end result is a matrix whose columns correspond to a specific function and the rows contain the observed data (in varying step sizes) ranging from an initial to a final measurement time. Note that the functions may also be centered and normalized in order to non-dimensionalize the library, which typically helps with numerical issues and also facilitates the identification of the critical functions.

For ST3LLARsat1, the data vector y was initially chosen based on the time histories of the Euler angles (or quaternions), the angular rates, the estimated magnetic field in Body-frame and its derivative. And to define the function library Θ, since initially the objective was to reconstruct the CubeSat dynamics as given in equation (19) (and from there, the physical parameters such as the moments of Inertia), it was formed using the terms shown in f_i on top of figure 2. And since the desired outputs to be regressed are the angular rates, the angular accelerations $\dot{\omega}$ had to be computed by differentiation of the rates after first randomly subsampling them, as explained in section 3. At this point a regularized regression solver can be used to solve equation (23). Several were tested, but finally the LASSO [8] implementation found in MATLAB's Statistics and Machine Learning Toolbox [9] was used. The schematic view for the described operation is shown in figure 2.

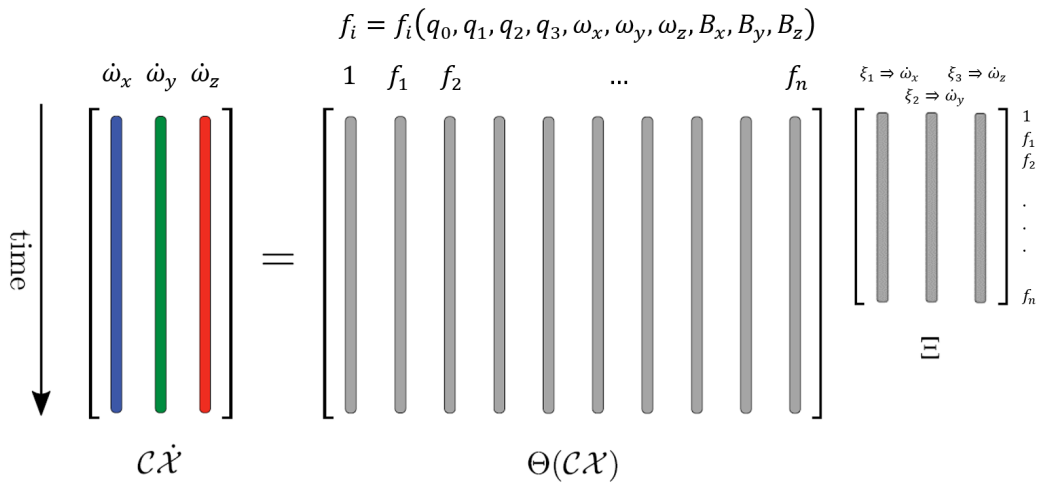

Fig. 2 Schematic of Compressed Sensing for ST3LLARsat1, adapted from [7]

As is typical in parameter estimation, the full time history is not recommended to be used for the estimation, so it is always recommended to parcel it into window data subsets [10]. Therefore, instead of performing a single regression over the entire detumbling simulation as it is done for the training part, a CS-windowed approach is followed for the verification where multiple sequential regressions are performed, each on a comparatively smaller simulation time frame (using only the rows of the function library corresponding to that frame). The data windows are defined by their period t_T and length t_L, and they should be defined to overlap in order to provide full time-history coverage. The two windowing variables t_* are part of the CS tuning hyperparameters that have to be optimized during the nominal training of the CS model.

An initial assessment was performed to ascertain the feasibility of the CS method to estimate the attitude dynamics of the CubeSat. This assessment yielded quite insightful lessons on the selection of the compressed input data y, the selection of the function library Θ, the selection of the solver for the regression, and also helped understand which CS hyperparameters needed to be tuned and how (e.g. the above windowing parameters).

Selection of the input data set y. The relevant variables included the spacecraft orientation quaternion, its angular rates, the magnetic field in the body frame, and the readings from the magnetometer and gyroscope/IMU. In addition, since the initial phase of the detumbling features fast oscillations in the angular rates, whereas after the detumbling is completed, these oscillations have subsided and the angular rates remain nearly constant around zero, but the sampling rate of the sensors remains fixed throughout the phase, there is a lack of continuous measurement points at the beginning and an excess at the end (where actually less information will be needed as the variables are nearly constant). Thus, the time period chosen for the CS estimation did not consider the initial detumbling and ended prior to the phase where the accelerations have mostly subsided.

Selection of the library matrix Θ. Numerous combinations of functions were assessed until a library featuring the quaternion, angular rates, and their sines and cosines, together with their permutations up to second order was chosen. This choice proved to be remarkably good at fitting the regression (that is, minimizing the error between the actual angular accelerations and the ones resulting from the CS solution), but its sparse representation s was hard to interpret and the resulting dominant functions had no resemblance to any terms in equation (19). Therefore, this approach did not allow for the estimation of any attitude dynamics parameters. In order to solve this issue, a different implementation was used where equation (19) was rewritten as in equation (29):

$$\begin{cases} \dot{\omega}_x = I_1 \cdot \omega_y\omega_z + K_1 \cdot \left(\frac{B\cdot\omega}{\|B\|^2}B_x - \omega_x\right) \\ \dot{\omega}_y = I_2 \cdot \omega_z\omega_x + K_2 \cdot \left(\frac{B\cdot\omega}{\|B\|^2}B_y - \omega_y\right) \\ \dot{\omega}_z = I_3 \cdot \omega_x\omega_y + K_3 \cdot \left(\frac{B\cdot\omega}{\|B\|^2}B_z - \omega_z\right) \end{cases} \quad \text{with} \quad \begin{cases} I_1 = -\frac{I_z - I_y}{I_x} \\ I_2 = -\frac{I_x - I_z}{I_y} \\ I_3 = -\frac{I_y - I_x}{I_z} \end{cases} \quad \begin{cases} K_1 = \frac{K_p}{I_x} \\ K_2 = \frac{K_p}{I_y} \\ K_3 = \frac{K_p}{I_z} \end{cases} \tag{29}$$

Now, the function library can be defined by including the measured angular rates $\{\omega_x, \omega_y, \omega_z\}$, their single-order combinations $\{\omega_x\omega_y, \omega_x\omega_z, \omega_y\omega_z\}$, and three mathematical functions combining the measured Body-frame magnetic field with the angular rate vector normalized with the square of its norm, i.e. $\left\{\frac{B\cdot\omega}{\|B\|^2}B_x, \frac{B\cdot\omega}{\|B\|^2}B_y, \frac{B\cdot\omega}{\|B\|^2}B_z\right\}$. It is noted that the values of the angular accelerations and of the Body-frame magnetic field come from respectively, the gyroscope and magnetometer sensors, which include measurement noise as per equations (27) and (26).

In accordance with CS theory, using the above matrix Θ for the regularized regression will give as solution a sparse representation of the dynamics parameters, whose coefficients will correspond to the values defined in equation (29). The resulting library can be visualized in figure 3, where each column is labeled according to the 9 aforementioned functions and the time period goes from a bit more than 1000 seconds (about 30 minutes after deployment when no maneuvering can be done) to over 2 hours.

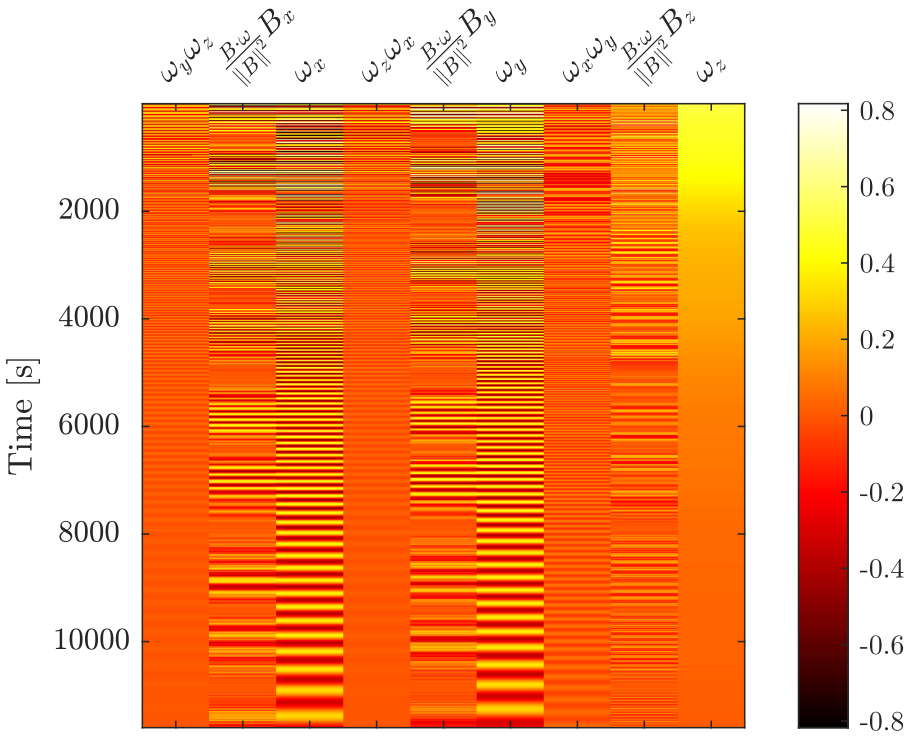

Fig. 3 Compressed Sensing function library of ST3LLARsat1 - nominal case

CS hyperparameter tuning. Next, 20 random simulations were generated using the Monte Carlo utility implemented in CDSim with the following values: 5% for all the σ_i and random δ_i values uniformly distributed in the range $[-1, 1]$. These simulation runs utilized the full precision allowed by CDS, which means that an iteration is performed every 0.1 s, matching the 10 Hz sampling rate of the CubeSat sensors. The CS algorithm was then applied to each of the resulting simulations and the CS hyperparameters manually tuned until the values in table 4 were chosen as giving the best results:

CS Hyperparameters	c $[-]$	λ $[-]$	$RelTol$ $[-]$	$DFmax$ $[-]$	t_i $[s]$	t_f $[s]$
values	3	10^{-6}	10^{-4}	∞	1500	7000

Table 4 CS Tuned Hyperparameters for ST3LLARsat1 - 20 random cases

Where c is the compression factor for the random subsampling of the observations used in the regression (meaning that only one out of every c measurements is kept); λ is the LASSO regression regularization factor; $RelTol$ is its relative tolerance; $DFmax$ the maximum number of non-zero coefficients in the solution (parameters to be estimated in the sparse representation); and finally, t_i and t_f are the initial and final simulation output times in between which CS is applied.

The application of the CS algorithm to each simulation run results in a sparse representation matrix whose coefficients give the estimated values of the parameters defined on the right of equation (29). As an example, Figure 4 shows the values corresponding to the nominal case, which serves to illustrate the correspondence between the coefficients and the dynamics parameters. Note that most of the values are zero or non-zero except for the coefficients corresponding to I_1 and I_2, which for the nominal case are respectively given by 0.6847826 and -0.6847826 ($I_3 = 0$ due to the CubeSat z-axis symmetry).

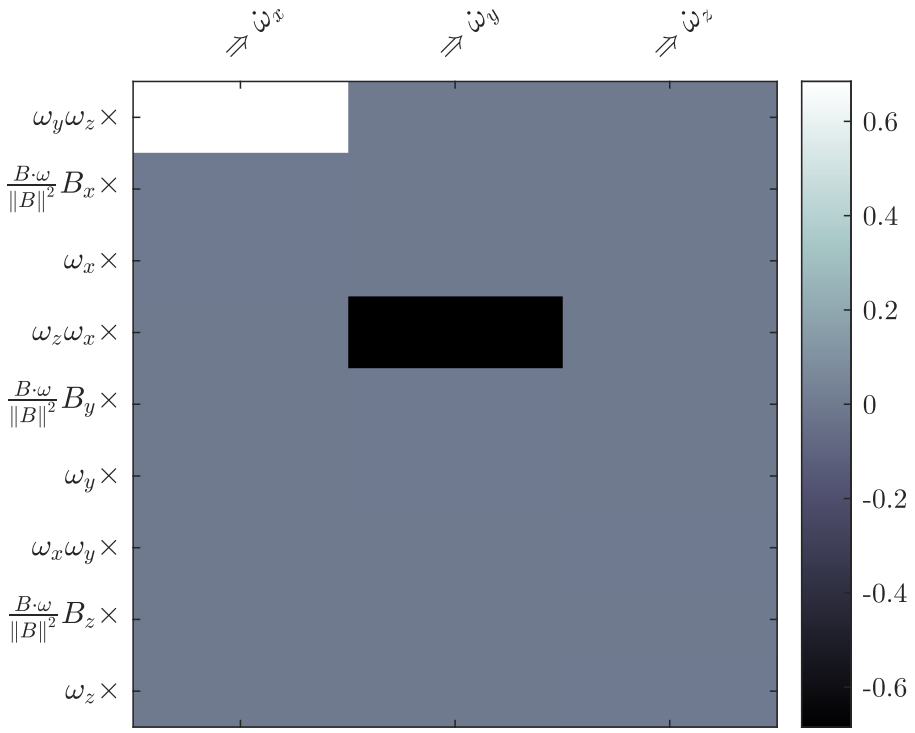

Fig. 4 CS sparse representation solution for ST3LLARsat1 - nominal case

Windowing tuning. Finally, as explained before, a windowing approach was used for the application of the CS estimation algorithm. This implies that the sensor observations are grouped in regular intervals on which CS is performed (together with the section of the function library that matches the window's time period). To do this, a window length t_L and period t_T must be chosen where the first parameter determines how many measurements are present in every window, while the second defines the number of windows. Using the same 20 random cases as before, the tuned values for these hyperparameters resulting in the best performance are given in table 5. Note that since $t_T = t_L/2$, these result in 35 completely overlapping windows in the analyzed time period. This means that every simulation instance belongs to exactly two windows, and hence, that there are two parameter estimates at every point. In the results shown next, the CS estimates correspond to the period of length t_T that lies centered around the middle of each window, which avoids the overlapping estimations.

Windowing Hyperparameters	t_L	t_T
values [s]	300	150

Table 5 Windowing Hyperparameters for ST3LLARsat1

5.1 ST3LLARsat1 CS Estimation results

The CS estimation assessment was performed including sensor noise, random variations of 5% in the orbital and initial parameters, and the nominal moments of inertia from table 1.

First, 20 cases were used considering constant the nominal inertial parameters $I_{I=1,2,3}$ and randomly changing all the other parameters. Figure 5 depicts the evolution of the estimates for I_1 (those for I_2 were similar). It can be observed that the CS algorithm is able to correctly estimate the value, with a maximum error of around 1%. The error grows larger towards the end of the selected detumbling phase segment, and indeed, this behavior would continue beyond the chosen time period for estimation, which is consistent with what was commented previously that when the acceleration oscillations decrease, it is more challenging for the algorithm to identify the dynamics since there is less excitation.

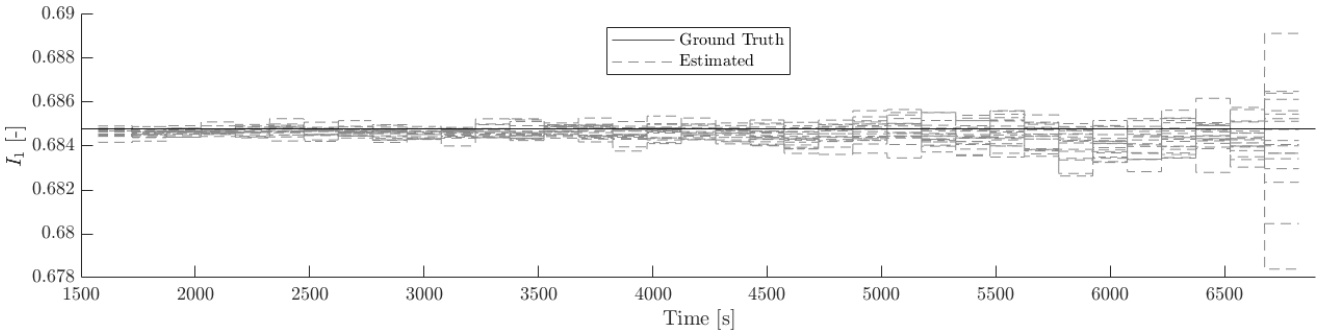

Fig. 5 Windowed Compressed Sensing estimation of inertial parameter I_1 in the nominal case

Additionally, inertial-parameters' vertex cases were tested. They were generated by varying the principal moments of inertia by 5% (i.e. $\sigma_{I_x} = \sigma_{I_y} = 5\%$) while keeping fixed $\delta = \pm 1$ (instead of random) according to whichever combination maximized or minimized the inertial parameters. The resulting vertex cases differ in their real I_1 and I_2 values by over 10% compared to the nominal ones. As before, 20 randomly-varied simulations were run for each vertex case, and the resulting windowed CS estimation is depicted in figure 6 (and again, it was similar for I_2).

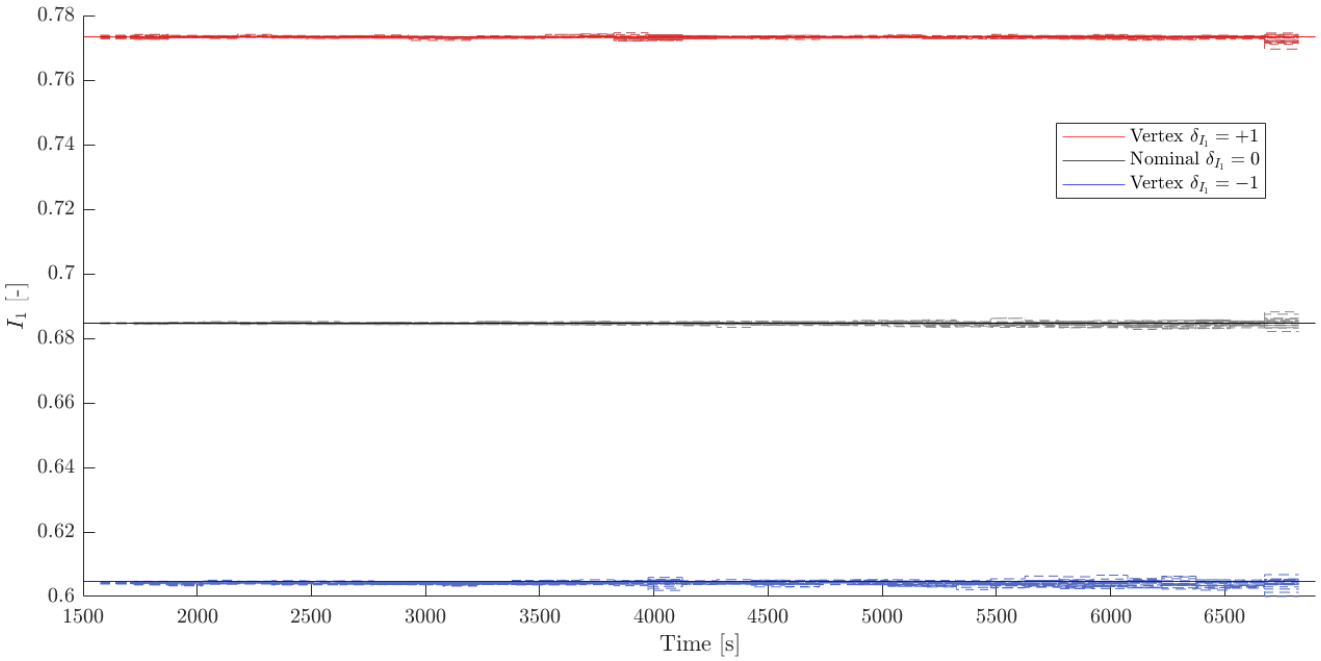

Fig. 6 Windowed-CS estimation for ST3LLARsat1 - inertial parameter I_1 vertex cases

Further assessment where performed (not shown for space reasons) assuming that the inertial parameters suffered a change during the detumbling, for example that some internal component became dislodged after launcher deployment, and the results yielded similar results.

6 Conclusions

In this article it has been shown the application of a Machine Learning technique, that of Compressed Sensing (CS), to the ST3LLARsat1 "Boira" CubeSat. The key elements of selection of the compressed input data y, selection of the function library Θ, selection of the solver for the regression, and tuning of the CS and windowing hyperparameters have been discussed on the application to ST3LLARsat1.

Overall, the windowed-CS approach proved its capacity to successfully identify and estimate the inertial parameters of the CubeSat under constant dynamics, and to do so with an onboard-achievable implementation, which would enable real-time parameter estimation and detection of a change in the attitude dynamics.

Acknowledgments

The work presented was part of the aerospace engineering undergraduate thesis of the first author under supervision of the second author. This thesis was performed at the University of Carlos III de Madrid (UC3M), and the work was funded by a collaborative scholarship from the Spanish Ministry for Education and Professional Development.

Declaration of Use of Artificial Intelligence

Artificial intelligence was not used to produce this article nor the work presented (except of course, that the work performed was to apply an AI/ML method to a CubeSat).

References

[1] W.E. Wiesel. *Spaceflight Dynamics*. Aphelion Press, 2010. ISBN: 9781452879598.

[2] Marco Lovera. Magnetic satellite detumbling: The b-dot algorithm revisited. In *2015 American Control Conference (ACC)*, pages 1867–1872, 2015. doi: 10.1109/ACC.2015.7171005.

[3] Giulio Avanzini and Fabrizio Giulietti. Magnetic detumbling of a rigid spacecraft. *Journal of Guidance, Control, and Dynamics*, 35(4):1326–1334, 2012. doi: 10.2514/1.53074.

[4] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019. doi: https://doi.org/10.1017/9781108380690.

[5] Edwin Christhuraj. Cubesat detumbling simulator. `https://github.com/echristhuraj/CubeSatDetumblingSimulator`, 2021. Commit: b6cd3901890b18398c6aee4e338457f3f228b57e. `https://github.com/echristhuraj/CubeSatDetumblingSimulator`.

[6] The MathWorks, Inc. wrldmagm: Use world magnetic model, 2020. `https://www.mathworks.com/help/aerotbx/ug/wrldmagm.html`.

[7] Diego Navarro Tapia, Andrés Marcos, and Joris Belhadj. Identification of ascent phase launcher dynamics using machine learning. 06 2023. doi: 10.5270/esa-gnc-icatt-2023-104.

[8] The MathWorks, Inc. Lasso or elastic net regularization for linear models, 2023. `https://www.mathworks.com/help/stats/lasso.html`.

[9] The MathWorks, Inc. Statistics and machine learning toolbox, 2023. `https://www.mathworks.com/help/stats/`.

[10] Shanwu Li, Eurika Kaiser, Shujin Laima, Hui Li, Steven L. Brunton, and J. Nathan Kutz. Discovering time-varying aerodynamics of a prototype bridge by sparse identification of nonlinear dynamical systems. *Phys. Rev. E*, 100:022220, Aug 2019. doi: 10.1103/PhysRevE.100.022220.