Learning-based Mitigation of Soft Error Effects on Quaternion Kalman Filter Processing

Tarso Kraemer Sarzi Sartori, Hassen Fourati, Rodrigo Possamai Bastos

IEEE Sensors Journal
Vol 24, Issue: 1, 01 Jan 2024, 1079 - 1089
ISSN: 1558-1748
DOI: 10.1109/JSEN.2023.3336054

LEARNING-BASED MITIGATION OF SOFT ERROR EFFECTS ON QUATERNION KALMAN FILTER PROCESSING

TARSO KRAEMER SARZI SARTORI[1,2] AND HASSEN FOURATI[2] AND RODRIGO POSSAMAI BASTOS[1]

Abstract. This paper proposes a novel mitigation technique for soft error effects on the attitude estimation (AE) processing for spacecrafts, especially for satellites' application. Specially, we are focused on the soft errors that occur in space and affect, for example, the quaternion Kalman filter, running on the processor of control system of satellite, which leads to invert bits of the estimated states, miscalculations and a decreased performance. The mitigation technique detects first the presence of soft error effects on the AE algorithm output using some residuals. Then the residuals are passed to a trained Machine Learning (ML) models to estimate the quaternion error that will be used to correct the estimations. A supervised regression solution was proposed to correct the soft error effects, in which a methodology for creating a dataset for training classical ML models was developed. The results from the case-study scenario show a high reduction of soft error effects, while adding little overhead to the Kalman filter processing.

1. Introduction

Electronic components can be upset by radiation particles in space. It creates transient faults able to invert memory bit elements of their circuits, characterizing single-event upset (SEU), or even interrupt their operation requiring a software reboot, defining a single event-functional interrupt (SEFI). Both of these events are considered as soft single-event effects or simply soft errors [1]. High-energy particles such as protons, electrons, and heavy ions, coming mainly from the Van Allen radiation belt [2] are responsible for inducing soft errors in space environment.

Spacecrafts, such as satellites, utilize Attitude Determination and Control Systems (ADCS) to determine the vehicle's attitude, throughout sensors' measurements, and subsequent control using actuators. The ADCS components are essentially electronic systems, sometimes upset by soft errors, as shown in Figure 1. As an example, the TDRS-1 is a satellite launched by NASA in April 1983 [3]. SEU events were observed in the ADCS Random Access Memory (RAM), during its trajectory to reach geosynchronous orbit. Some SEUs caused serious anomalies, which required ground control to keep the satellite's desired orientation.

Low Earth Orbit (LEO) spacecraft missions are also exposed to large radiation risks, due to their orbital motion inside the Van Allen inner radiation belt (600-10,000km of altitude), mainly when they pass over the South Atlantic Anomaly

(SAA), a localized region where the radiation fluxes reach their highest intensities [4]. Several factors can affect the SAA environment, such as geomagnetic storms, which are able to increase the SEU rates up to 40% in the recovery phase in relation to the initial phase of the storm [4]. In [5] a study for a circular orbit with inclination of 63° and 1111km of altitude, which passes through the SAA, was made to calculate the soft error rate caused by proton particles in the radiation belts in a representative satellite. It was found that for a memory chip of 64 Mbits, a soft error rate of 10^4 bit/day could be observed. The effects of the majority of the soft errors would not cause significant impact, acting just as an extra noise. However, if a soft error occurs in a critical function, such as in programs controlling actuators or in the data analyses, it could lead to dangerous scenarios [5].

The Attitude Estimation (AE), one of the main components of the ADCS, consists of determining the orientation of the spacecraft relative to a reference frame using sensors' measurements. The extended Kalman filter (EKF) is one of the most applied algorithms for real-time spacecraft AE [6], [7], [8]. However, this algorithm demands the linearization of the non-linear equations that relate the sensors' measurements with the current attitude (the measurement equation). This linearization procedure can provoke undesirable effects such as sensitivity to initial conditions, and also an increase in the computational load. Alternatively, the authors in [9] propose a novel quaternion Kalman filter (NQKF) algorithm, employing a special manipulation on the measurement equation, eliminating thus the linearization step. The AE is a critical task that must be accomplished by

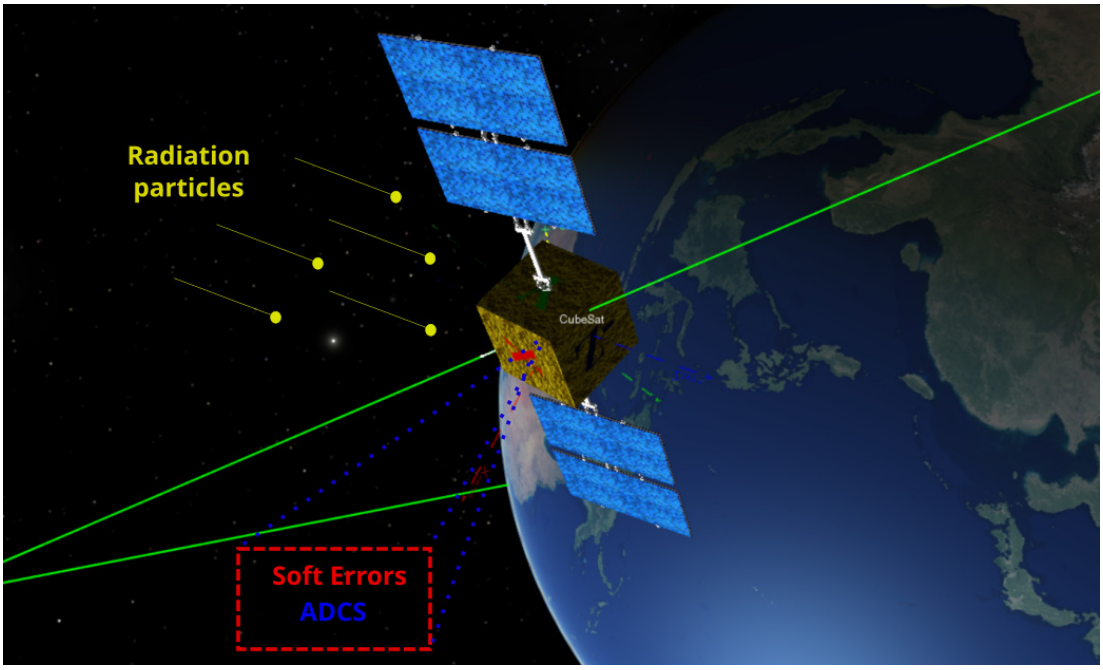

FIGURE 1. Satellite's ADCS under soft errors caused by radiation (VTS Timeloop software from Centre National d'Études Spatiales (CNES)).

the satellite's ADCS. As aforementioned, this system can be upset by soft errors caused by the radiation environment, especially for the processor where the Kalman filter-based AE algorithms run. In our previous works [10, 11, 12] we have already investigated the soft error effects on AE processing and exposed results obtained during experimental radiation campaigns. Therefore, the soft error effects should be mitigated for ensuring the system reliability.
A variety of works have addressed Kalman filter-based algorithms reliability [13, 14, 15]. However these works deal with faults, outliers, external perturbations, such as external accelerations[16, 17] or magnetic distortions [18], and different kind of noises that may occur in the sensors' measurements over time, then are not appropriate to be used in this framework. Unlike, soft errors, more specifically SEUs, are punctual events that occur when high energy particles hit memory elements inverting its states, i.e. bit-flips. The SEU effects in a processing system running an AE algorithm may result in miscalculations, hence providing a wrong attitude estimation, as shown in [10]. Plenty of known mitigation techniques can be employed for addressing this issue. Redundancy is one of the most common techniques, and can be applied replicating hardware, adding check bits, and even running multiple processes of the same program [19]. Error detection and correction codes are also common methods applied in current processors and memories architectures. However, these techniques are not completely fail-safe, they can increase the cost, processing time, and application's weight.

Machine Learning (ML) techniques have been successfully applied for improving the AE performance, or even replace the classical AE algorithms (estimation without dynamic model). In [20] a reinforcement learning method was applied to improve the performance of an EKF algorithm, addressing inaccurate initial estimation, filter gains, and noise model. A method for training deep neural networks was developed in [21] for enhancing the AE by a Kalman filter. In [22] a novel attitude estimator model was conceived based on recurrent neural networks able to eliminate sensor errors, while implementing dynamic AE. A long short-term memory neural network was trained in [23] using real quadrotor sensors' data for AE, and high accuracy results were obtained. In [24] a magnetic field gradient-based EKF is used along with a BiLSTM network to achieve better estimates of the velocity of a moving body using inertial sensors.
To the best of our knowledge, the mitigation of SEU on AE processing is not well addressed until now in the literature. More specifically, we deal with bit-flips in the algorithm's variables occurring in the processor during the computation of quaternion from a set of measurements, that can lead to miscalculations and consequently a wrong estimated attitude. The impact of learning approaches on AE processing to mitigate soft error effects (SEU) has the merit to be studied deeply, using residuals and a correction phase. A specific and realistic soft errors injection on AE processing was proposed, followed by an effective methodology to create a training dataset for mitigation, in which a satellite's orbit and the characteristics of sensors used for the mission need to be taken into account. Finally, the paper gives a deep analysis of the effects of ML models on AE processing and the improvement rates for mitigation of SEU effects in satellite applications.

This paper is organized as follows: Section II explains the framework of the novel technique developed for soft error effects mitigation, with some details on the considered quaternion Kalman filter and the methodology for creating a training dataset for ML models. Section III contains the ML models dataset generation and training results, as well as the simulation results considering the AE algorithm combined with the ML mitigation technique. Section IV presents a brief discussion about the impacts of the proposed solution on the control strategy inside the ADCS. Section V gives some conclusions of the work and possible perspectives. Finally, the Appendix I presents a brief description of three supervised ML models we used as component of the mitigation technique.

2. Framework for Soft Error Effects Mitigation approach on Quaternion Kalman filter for attitude estimation

This section describes the framework of the proposed ML-based technique to mitigate soft error effects on the processor of ADCS, where the quaternion Kalman filter is implemented. This framework is presented in Figure 2 and can be summarized as follows:

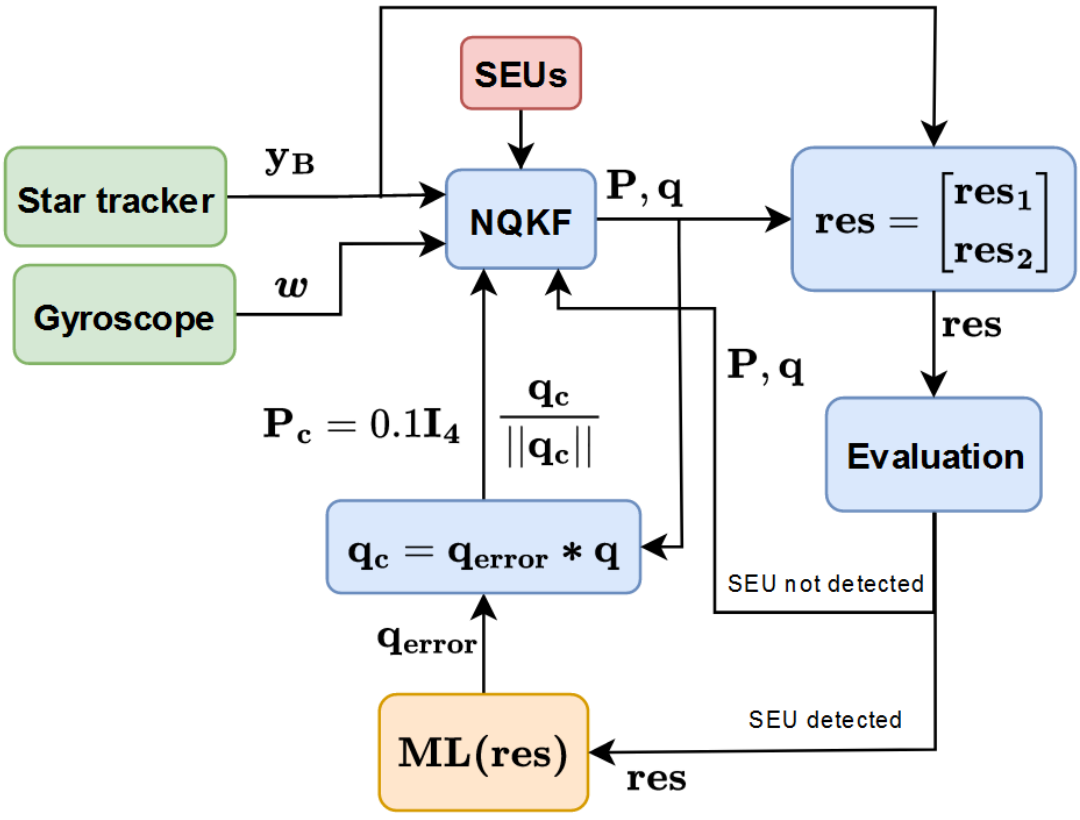

Figure 2. ML-based AE processing for soft error mitigation.

(1) The NQKF, for AE processing, receives a set of measurements from the used sensors. In this work we considered a 3-axis gyroscope and a star tracker as main sensors. Their measurements are assumed not disturbed by soft errors.

(2) The NQKF runs with this set of measurements using the processing resources of the ADCS. It is assumed that a SEU occurs in any variable of the NQKF during its processing, leading to miscalculations. The attitude quaternion $\mathbf{q}$ is then estimated, then the residuals are calculated using $\mathbf{q}$ and $\mathbf{y_B}$. We consider the star tracker will track the position of two stars, hence providing two vectors, $\mathbf{y_{B_1}}$ and $\mathbf{y_{B_2}}$ that compose $\mathbf{y_B}$. Therefore, the **res** vector is calculated as:

$$\mathbf{res} = \begin{bmatrix} \mathbf{res_1} \\ \mathbf{res_2} \end{bmatrix} = \begin{bmatrix} \mathbf{y_{B_1}} - \mathbf{A}[\mathbf{q}]\mathbf{r_{N_1}} \\ \mathbf{y_{B_2}} - \mathbf{A}[\mathbf{q}]\mathbf{r_{N_2}} \end{bmatrix} \tag{2.1}$$

in which $\mathbf{r_{N1}}$ and $\mathbf{r_{N2}}$ are the reference vectors in $\mathbf{N}$ (see section 2.1).
(4) Each residual in **res** pass through an evaluation step, where its norm is compared with a predefined threshold. If both residual norms are bigger than the threshold (SEU detected), the **res** vector will be passed to a ML model, already trained, to estimate the orientation error ($\mathbf{q_{error}}$) caused by the soft error. Otherwise, the ML model will not be activated (SEU not detected), hence no correction will be applied. In that case, the NQKF will continue to run normally and the next iteration starts.
(5) If the SEU is detected, $\mathbf{q}$ is multiplied by $\mathbf{q_{error}}$ throughout a quaternion multiplication producing a corrected quaternion $\mathbf{q_c}$. The normalized $\mathbf{q_c}$ will then replace $\mathbf{q}$, and also the error covariance matrix $\mathbf{P}$ is re-initialized as $\mathbf{P_c} = 0.1 \cdot \mathbf{I_4}$.

Detailed explanations of the variables in Figure 2, the NQKF algorithm, and the ML methodology can be found in Sections 2.1 and 2.2.

2.1. Quaternion Kalman Filter for AE.

2.1.1. *Preliminaries Necessary for AE.* The objective of the AE is to determine the orientation of a moving frame fixed to the body in the application (body frame $\mathbf{B}$) in relation to a reference frame ($\mathbf{N}$) normally celestial or Earth-fixed, based on at least two observation vectors (measurements). The observation equation based on these vectors can be used in dynamic estimation approaches (NQKF for example). It is defined as follows:

$$\mathbf{y_B} = \mathbf{y_B^0} + \delta\mathbf{y_B} = \mathbf{A}[\mathbf{q}]\mathbf{r_N} + \delta\mathbf{y_B}, \tag{2.2}$$

where vectors $\mathbf{y_B^0} \in \mathbb{R}^{3\times1}$ and $\mathbf{r_N} \in \mathbb{R}^{3\times1}$ are the normalized projections of physical vectors along the axes of $\mathbf{B}$ and $\mathbf{N}$, respectively. The vector $\mathbf{y_B} \in \mathbb{R}^{3\times1}$ represents the output of sensors fixed on $\mathbf{B}$, whereas $\mathbf{r_N}$ is known from a physical model. The vector $\delta\mathbf{y_B} \in \mathbb{R}^{3\times1}$ is an additive measurement noise. The attitude is represented in form of quaternion $\mathbf{q} \in \mathbb{R}^{4\times1}$, which is a hypercomplex vector as follows:

$$\mathbf{q} = \begin{bmatrix} \hat{\mathbf{e}} \\ q_4 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{n}} \cdot sin(\alpha/2) \\ cos(\alpha/2) \end{bmatrix} \tag{2.3}$$

where q_4 and $\hat{\mathbf{e}} = [q_1 \ q_2 \ q_3]^T \in \mathbb{R}^{3\times1}$ are the quaternion real and complex components, respectively. Also, the quatrenion can be decomposed in an angle of rotation α and an unitary axis $\hat{\mathbf{n}} \in \mathbb{R}^{3\times1}$, that indicates the direction of rotation. The attitude matrix $\mathbf{A}[\mathbf{q}] \in \mathbb{R}^{3\times3}$, function of the quaternion, is represented by:

$$\mathbf{A}[\mathbf{q}] = (q_4^2 - \hat{\mathbf{e}}^T\hat{\mathbf{e}})\mathbf{I}_3 + 2\hat{\mathbf{e}}\hat{\mathbf{e}}^T - 2q_4\mathbf{S}[\hat{\mathbf{e}}]. \tag{2.4}$$

This matrix is able to convert a vector represented in the reference frame $\mathbf{N}$ into the body frame $\mathbf{B}$. The matrix $\mathbf{S}[\mathbf{e}] \in \mathbb{R}^{3\times3}$ is the skew-symmetric matrix applied to the quaternion complex part, as shown in Equation 2.5, and $\mathbf{I_3} \in \mathbb{R}^{3\times3}$ is an identity matrix. Equation 2.4 is non-linear, thereby 2.2 as well.

$$\mathbf{S}[\mathbf{e}] = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{2.5}$$

2.1.2. *Overview of the Novel Quaternion Kalman Filter (NQKF).* The NQKF algorithm studied in this paper was developped in [9]. The state vector $\mathbf{x} \in \mathbb{R}^{4\times1}$ is composed only by the attitude quaternion, the sensors' biases are not estimated, i.e. $\mathbf{x} = \mathbf{q}$. Manipulating Equation 2.2 is possible to establish a linear pseudo-measurement equation that can be used in the Kalman filter algorithm, and then the linearization process is skipped. The summary of the case-study NQKF algorithm implemented is as follows:

- **Time Propagation**:
  - Initialize the state vector $\mathbf{x_{0/0}} = \mathbf{q_{0/0}}$ and the error covariance matrix $\mathbf{P_{0/0}}$.
  - Given $\mathbf{x_{k/k}}$ calculate the predicted state vector:

$$\mathbf{x_{k+1/k}} = \mathbf{\Phi_k x_{k/k}} = e^{(1/2)\mathbf{O}[\boldsymbol{w_k}]T_s}\mathbf{x_{k/k}},$$

    where $\mathbf{O}[\boldsymbol{w_k}] = [-\mathbf{S}[\boldsymbol{w_k}] \;\; \boldsymbol{w_k}; -\boldsymbol{w_k}^{\mathbf{T}} \;\; \mathbf{0}]$ is a matrix operator, function of the angular velocity vector $\boldsymbol{w_k} \in \mathbb{R}^{\mathbf{3\times1}}$ measured at the instant k, and the T_s is the sensors' sampling time. $\mathbf{S}[\boldsymbol{w_k}]$ is the skew-symmetric matrix applied to $\boldsymbol{w_k}$.
  - Given $\mathbf{P_{k/k}}$ calculate the predicted error covariance matrix:

$$\mathbf{P_{k+1/k}} = \mathbf{\Phi_k P_{k/k} \Phi_k^T} + 0.25T_s^2\mathbf{E}[\mathbf{x_{k/k}}]\mathbf{Q}\mathbf{E}[\mathbf{x_{k/k}}]^{\mathbf{T}},$$

    where $\mathbf{E}[\mathbf{x_{k/k}}] = [\mathbf{S}[\mathbf{e_{k/k}}] + q_{4_{k/k}}\mathbf{I_3}; -\mathbf{e_{k/k}}^T]$ is a quaternion-matrix operator, and $\mathbf{Q} = \sigma_w^2\mathbf{I_3}$ is the noise covariance matrix of the state $\mathbf{q_k}$, following a normal distribution $\sim N(0, \sigma_w)$.
- **Measurement Update**:
  - Given $\mathbf{y_{B_{k+1}}}$ and $\mathbf{r_{N_{k+1}}}$ compute:

$$\mathbf{s_{k+1}} = 0.5(\mathbf{y_{B_{k+1}}} + \mathbf{r_{N_{k+1}}}).$$

$$\mathbf{d_{k+1}} = 0.5(\mathbf{y_{B_{k+1}}} - \mathbf{r_{N_{k+1}}}).$$

  - Build the observation matrix:

$$\mathbf{H_{k+1}} = \begin{bmatrix} -\mathbf{S}[\mathbf{s_{k+1}}] & \mathbf{d_{k+1}} \\ -\mathbf{d_{k+1}}^T & 0 \end{bmatrix}.$$

  - Compute the Kalman gain:

$$\mathbf{K_{k+1}} = \mathbf{P_{k+1/k}H_{k+1}^T}(\mathbf{H_{k+1}P_{k+1}H_{k+1}^T} + \mathbf{R_{k+1}})^{-1},$$

where $\mathbf{R_{k+1}} = 0.25\ T_s^2\ \mathbf{E}[\mathbf{x_{k+1/k}}]\ \mathbf{R}\ \mathbf{E}[\mathbf{x_{k+1/k}}]^{\mathbf{T}}$ is the measurement noise covariance matrix, and $\mathbf{R} = \sigma_v^2\mathbf{I_3}$ is the covariance matrix associated to the measurement noise, assumed to have a normal distribution $\sim N(0, \sigma_v)$.

– Calculate the estimated state vector:

$$\mathbf{x_{k+1/k+1}} = \mathbf{x_{k+1/k}} - \mathbf{K_{k+1}}\mathbf{H_{k+1}}\mathbf{x_{k+1/k}}.$$

– Calculate the estimated error covariance matrix:

$$\begin{aligned}\mathbf{P_{k+1/k+1}} = (\mathbf{I_4} - \mathbf{K_{k+1}}\mathbf{H_{k+1}})\mathbf{P_{k+1/k}} \\ (\mathbf{I_4} - \mathbf{K_{k+1}}\mathbf{H_{k+1}})^{\mathbf{T}} \\ +\mathbf{K_{k+1}}\mathbf{R_{k+1}}\mathbf{K_{k+1}^{T}}.\end{aligned}$$

– Normalize the state vector (the quaternion):

$$\mathbf{x_{k+1/k+1}} = \frac{\mathbf{x_{k+1/k+1}}}{||\mathbf{x_{k+1/k+1}}||}.$$

For the sake of simplicity, from now on the estimated state vector $\mathbf{x_{k+1/k+1}}$ will be referred as just the quaternion $\mathbf{q}$, the state predicted $\mathbf{x_{k+1/k}}$ as $\mathbf{q_m}$, the predicted covariance matrix $\mathbf{P_{k+1/k}}$ as $\mathbf{P_m}$, the Kalman gain $\mathbf{K_{k+1}}$ as $\mathbf{K}$, and the estimated covariance matrix $\mathbf{P_{k+1/k+1}}$ as $\mathbf{P}$.

2.2. **ML Models and the Methodology for Creating a Training Dataset.**

The ML model needs to accomplish the following task: given a set of residuals (input), find the respective quaternion error (output) associated to these residuals. This scenario can be modeled as a supervised ML regression problem. Three classical ML models used for this task were implemented and tested: Artificial Neural Network (ANN), Decision Trees (DT), and Random Forest (RF) (see Appendix6 for detailed description of these models).

For implementing the proposed mitigation approach, it is necessary to train the ML models with a training dataset. Due to the non-availability of real satellite data under radiation effects and the complexity to obtain such dataset, a methodology for creating a training dataset was conceived as follows: (1) Case-study scenario definition; (2) Sensor measurements acquisition (3) Soft errors injection on AE processing; (4) Training dataset samples generation.

2.2.1. *Case-Study Scenario Definition.*

As aforementioned, the case-study scenario we are considering is a satellite orbit. In Figure 3, three important reference frames for defining a satellite's orbit are represented. The celestial frame is an Earth-centered stationary reference frame. Usually, this frame has its X-axis pointing to the fixed direction of the vernal equinox, Z-axis normal to the equatorial plane, and Y-axis completing the the orthogonal triad (XY contained in the equatorial plane). The perifocal frame ($\mathbf{N}$) lies on the orbital plane, having its x_N-axis pointing towards the periapsis point (see Figure 3), the location in the orbit where the satellite is the closest to the Earth, z_N-axis is normal to the orbital plane, and y_N-axis completes the orthogonal system. Finally, the body frame ($\mathbf{B}$) is fixed on the satellite center of mass.

The satellites' orbits are actually ellipses, having one of its focuses located on the

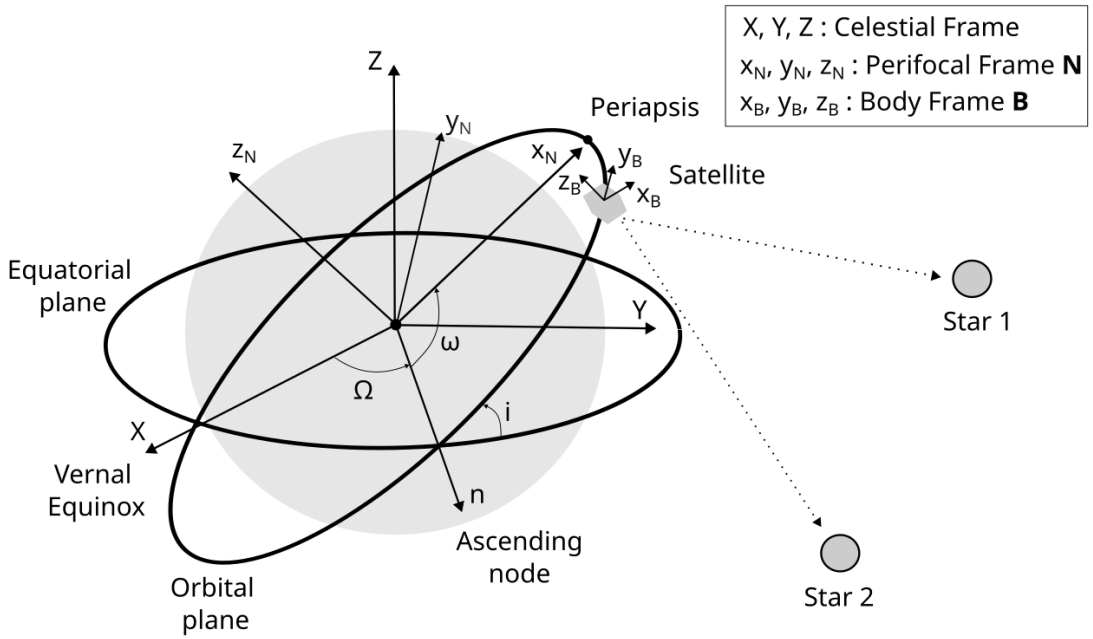

FIGURE 3. Satellite reference frames and orbital elements for dataset generation.

planet's center of mass. An orbit can be defined by six classical orbital elements [25]. The ellipsoidal shape is defined by its eccentricity e and its semi-major axis a. Moreover, the Euler angles Ω, i, ω, called right ascension, inclination, and perigee argument respectively, indicate the orientation of N in relation to the celestial frame. The last orbital element is called time of periapsis τ, which represents either the time that the satellite last passed by the periapsis point ($\tau < 0$) or the remaining time to reach the periapsis ($\tau \geq 0$). The satellite desired attitude in the present study depends on the satellite's position over time in the orbit. The defined attitude objective is: maintain the negative x_B-axis pointing towards the center of the Earth, while keeping thez_B-axis aligned with the z_N-axis during all the orbit.

2.2.2. *Sensor's Measurements Acquisition.* In this work the sensor measurements were obtained throughout simulation. The satellite's positions and velocities in a given orbit in relation to the celestial frame could be obtained by implementing the equations of motion for the two-body problem exposed in [25]. Only the Earth-satellite system was considered, being the Earth's gravitational force the only force acting on the satellite. For keeping the desired orientation, a control system was designed. Knowing the satellite's angular velocity and initial orientation ($\mathbf{q_0}$), the true quaternion representing the orientation of **B** with respect to **N** can be calculated by solving:

$$\mathbf{q_{true}} = e^{(1/2)\mathbf{O}[\boldsymbol{w}_{true}]\mathbf{t}}\mathbf{q_0}, \tag{2.6}$$

where $\boldsymbol{w}_{true}$ is the satellite's true angular velocity (without any noise added), $\mathbf{t}$ is the time, and the matrix operator $\mathbf{O}$ is defined in Section 2.1. The satellite's true

positions and orientations were used for generating the sensors' measurements. The gyroscope measurements were modeled as:

$$\boldsymbol{w} = \boldsymbol{w_{true}} + \delta_w \tag{2.7}$$

where δ_w is an additive white Gaussian noise $\sim N(0, \sigma_{\boldsymbol{w}})$. The star tracker's measurement model was implemented for tracking the position of two stars, thus generating two other references (as observations) for the NQKF. These references are vectors pointing towards the chosen reference stars. Figure 4 illustrates the process for obtaining the position vectors of the stars with respect to the satellite in the perifocal frame ($\mathbf{r_N}$). Having the satellite's and stars' positions in the

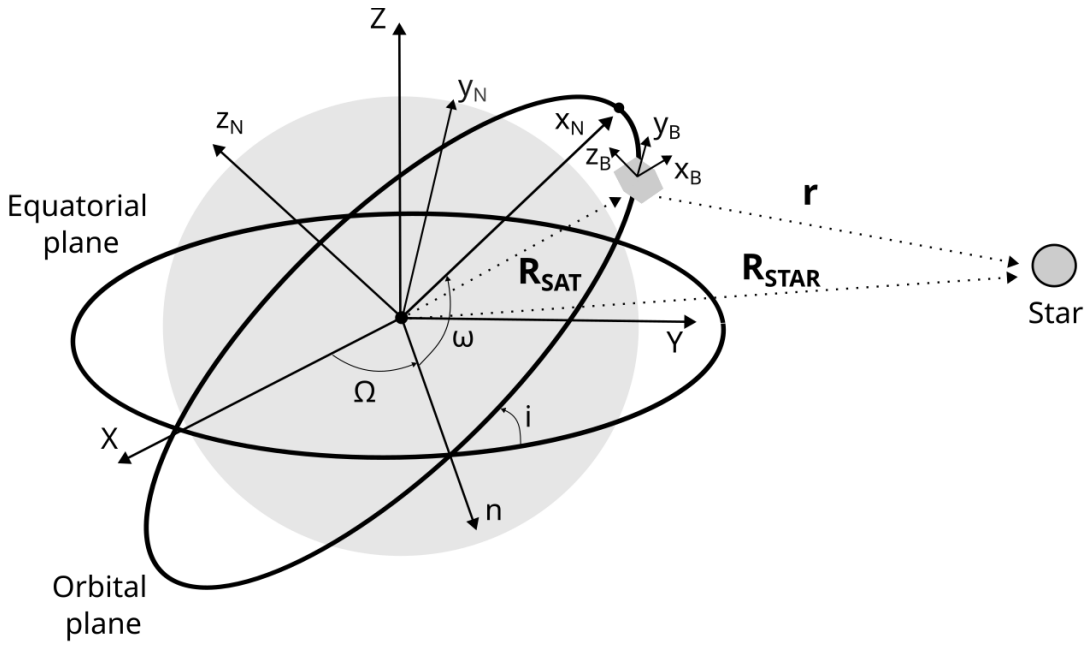

FIGURE 4. Satallite's and star's position vectors.

celestial reference frame, $\mathbf{R_{SAT}}$ and $\mathbf{R_{STAR}}$ respectively, it is possible to obtain the relative position of the star with respect to the satellite in the perifocal frame ($\mathbf{r_N}$) as:

$$\mathbf{r} = \mathbf{R_{STAR}} - \mathbf{R_{SAT}}$$
$$\mathbf{r_N} = \mathbf{C_3}(\omega)\mathbf{C_1}(i)\mathbf{C_3}(\Omega)\mathbf{r}/||\mathbf{r}||,$$

where $\mathbf{C_1}$ and $\mathbf{C_3}$ are the fundamental rotation matrices for rotating a vector in the x and z axis respectively [25].
Finally for generating the star tracker measurements $\mathbf{r_B}$, the following model was used:

$$\mathbf{r_B} = \mathbf{A}[\mathbf{q_{true}}]\mathbf{r_N} + \delta_s \tag{2.8}$$

where δ_s is an additive white Gaussian noise $\sim N(0, \sigma_s)$. The gyroscope measurements are used in the time propagation step of the NQKF, whereas the star tracker signals in the measurement update step as:

$$\mathbf{y_B} = \begin{bmatrix} \mathbf{r_{B_1}} \\ \mathbf{r_{B_2}} \end{bmatrix} = \begin{bmatrix} \mathbf{A}[\mathbf{q_{true}}]\mathbf{r_{N_1}} + \delta_s \\ \mathbf{A}[\mathbf{q_{true}}]\mathbf{r_{N_2}} + \delta_s \end{bmatrix} \tag{2.9}$$

in which $\mathbf{r_{B_1}}$ and $\mathbf{r_{B_2}}$ are the positions of the two stars provided by the star tracker in $\mathbf{B}$, whereas $\mathbf{r_{N_1}}$ and $\mathbf{r_{N_2}}$ are the position vectors of the same stars represented in $\mathbf{N}$.

2.2.3. *Soft Errors Injection on AE processing.* Among the different types of soft errors that can be provoked by radiation particles in electronic systems, the SEUs will be object of study. This kind of soft error is characterized by the inversion of a single or multiple bits in a memory element at a time [26]. Modern processor architectures usually handle variables with 32 or 64 bits. Arithmetic and logic unities and floating-point unities are present in modern CPUs to perform floating point operations, in which real numbers are usually expressed in the IEEE-754 standard [27]. Figure 5 shows the binary representation of a double precision floating-point number in the IEEE-754 standard.

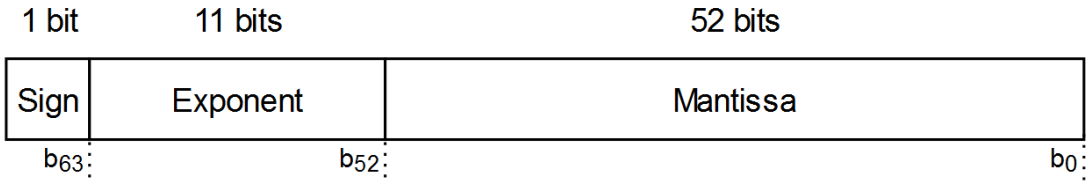

FIGURE 5. IEEE-754 double precision floating-point number binary representation.

The number is represented in a sequence of 64 bits, 1 bit dedicated to the sign, 11 bits to the exponent and 52 bits to the mantissa. Equation 2.10 can be used to convert the binary number to decimal representation.

$$n_{decimal} = (-1)^s \cdot (1 + \sum_{i=1}^{i=52} b_{52-i} 2^{-i}) \cdot 2^{E-1023} \tag{2.10}$$

In Equation 2.10, s represents the sign bit, b inside the summation represents the respective mantissa bit, and E is the decimal format of the exponent binary representation. The value stored in the bits relative to the exponent has a bias added to be possible storing both positive and negative exponents. For this representation, a bias of 1023 is added. Consequently, when converting back to decimal, a factor of 1023 needs to be subtracted from E, as Equation 2.10 shows.

It was necessary to observe the SEU effects directly on the algorithm's response, the faults could not be masked, and should be produced frequently. Therefore, in this work a high level method for injecting soft errors on the AE algorithm's execution was adopted. The sensor's measurements were split in windows of 5 sec and then processed by the NQKF. During the computation, a SEU was injected in a NQKF variable (such as one component of the Kalman gain or the error covariance matrices, or even the estimated quaternion for example, cf. Section 2.1) right in the middle of the time window. The selected variable was firstly converted into its binary double precision representation (cf. Figure 5), and then a single bit was inverted. After, the variable was converted back to its decimal representation using Equation 2.10, and passed to the algorithm to continue the computation.

2.2.4. *Training Dataset Samples Generation.* Finally, with the quaternions estimated by the AE algorithm under SEU effects, the quaternion error could be calculated through:

$$\mathbf{q_{error}} = \mathbf{q_{true}} * \mathbf{q}^{-1} \tag{2.11}$$

where $*$ represents the quaternion multiplication.
Furthermore, the residuals **res** (cf. Figure 2) are then computed using the estimated quaternion **q** and the star tracker's measurements from the two tracked stars considered.

In conclusion, the training dataset is composed of samples made with the inputs $\mathbf{res} \in \mathbb{R}^{6\times1}$ and the outputs $\mathbf{q_{error}} \in \mathbb{R}^{4\times1}$. It will be used for training the ML models.

3. Main Results of the Mitigation Approach

In this section, we give the parameters defined for creating the training dataset, show some results of the ML models selection/training, and finally propose a performance evaluation of the ML-based AE technique for soft error effects mitigation.

3.1. **Training Dataset Generation Results.** The orbit used for generating the training dataset samples for the ML models is defined by the following orbital elements:$e = 0.0001$, $a = Re{+}1111$ km (Re is the Earth's equatorial radius ~ 6371 km), $\tau = 1000$ sec, $\Omega = 0°$, $i = 63°$, $\omega = 0°$. A Python program was implemented for generating the orbital data, the sensors' measurements and further soft error injection, as described in Section 2. Figure 6 shows the attitude quaternions obtained for a section of the reference orbit. The abrupt variation of quaternion at the beginning of the simulation is due to the control's action trying to direct the satellite. Alpha Centauri and Sirius were the chosen stars for the star tracker monitoring, considering their positions in the J2000 celestial referential, in which the axis are oriented following the Earth's mean vernal equinox and mean rotation axis in the year 2000. The sampling time for the used sensors was chosen as 0.1 sec, and the noises for the gyroscope and star tracker following $\sim N(0, \sigma_{\boldsymbol{w}} = 10^{-3}$ rad/s) and $\sim N(0, \sigma_s = 10^{-4})$, respectively. Moreover, the measurement error of the attitude from the utilized star sensors was calculated theoretically to be about 20 arcsec, which is in accordance with the precision of commercial star trackers. If different sensors are used in such application with different orbit parameters, new data needs to be acquired and added to the database to train the ML models.

For executing the soft errors injection procedure described in Section 2.2.3, different NQKF variables were assessed. A testing dataset with 30 sec of simulation for the same sensors previously defined and the same orbit was created, with $\tau = 0$. SEUs were injected following the procedure described each 5 sec, i.e. a single bit of the different assessed variables was changed. For each variable, 500 simulations were performed, in which the bit (only signal and exponent) and the variable component were chosen randomly. Table 1 shows the results for simulations made to find the most critical variables to be disturbed by the SEU effects. The performance of the NQKF algorithm under SEU effects was

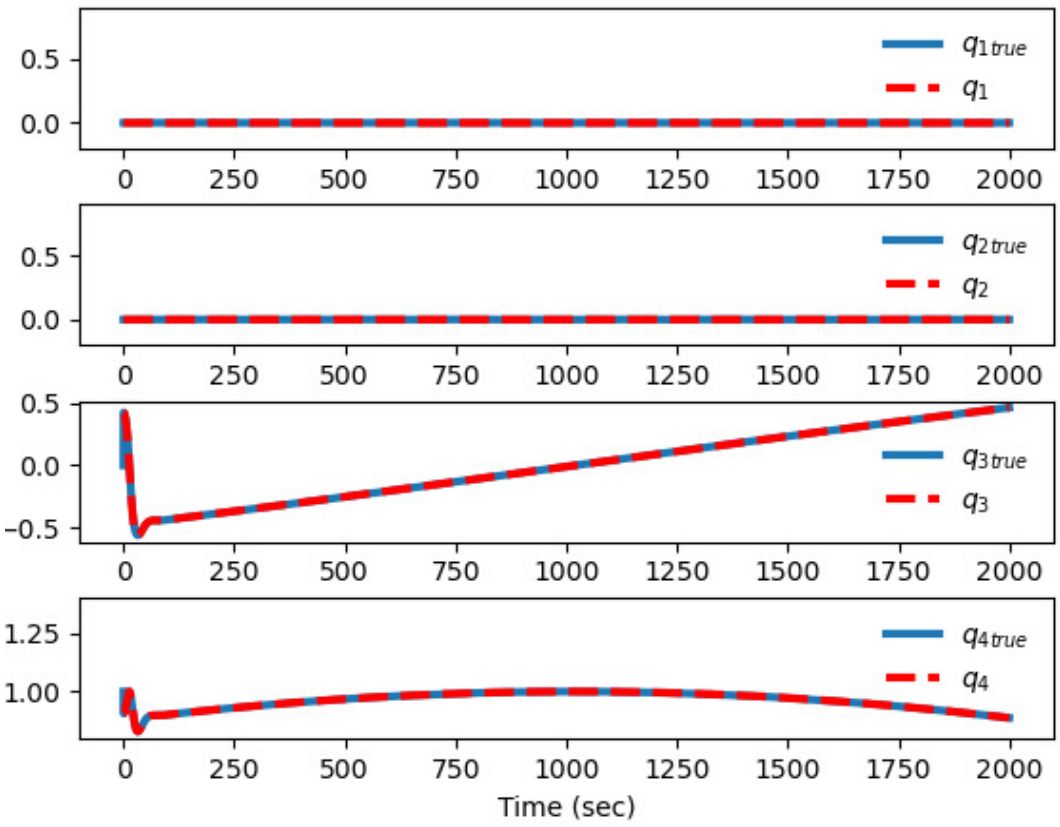

FIGURE 6. True quaternion and estimated one by the NQKF algorithm without soft error effects for partial satellite's orbit.

measured throughout the root mean square error (RMSE) defined as:

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(y_i^{true} - y_i)^2}{N}}, \tag{3.1}$$

where y_i^{true} and y_i represent the Euler angles obtained converting the estimated quaternion $\mathbf{q}$ without and with the SEU effects, respectively, following the Yaw-Pitch-Roll sequence. N is the number of points in the simulations. The RMSE for each angle in each simulation was then averaged.

TABLE 1. NQKF RMSE results for SEUs injected in different variables.

Variables of NQKF	RMSE $\phi(^o)$	RMSE $\theta(^o)$	RMSE $\psi(^o)$
$\mathbf{q}$	0.058624	0.035259	0.049740
$\mathbf{K}$	0.006429	0.001981	0.006429
$\mathbf{q_m}$	0.051221	0.033385	0.038866
$\mathbf{P_m}$	0.006440	0.002054	0.006436
$\mathbf{P}$	0.006435	0.001992	0.006438
q_4	0.167030	0.092891	0.150951
q_{m4}	0.143919	0.088078	0.109431

The most impacting effects were observed when injecting soft errors in the estimated quaternion $\mathbf{q}$, more specifically in the real component q_4, in which the largest RMSEs were observed. Finally, the training dataset contains the data in which SEUs were injected in the real components of the estimated quaternion, in the bits b_{63}, b_{61}, b_{60}, b_{59}, and b_{58} (See Figure 5). A dataset 1 with 2000 sec of simulation and a dataset 2 with multiple simulations of 70 sec each, were created. Both are simulations for the same orbit, with the same sensors, and same bits chosen for injecting SEUs. Nevertheless, Table 2 shows the parameters that differentiate the two datasets. All the parameters combinations were simulated. The final training dataset is a merge of these two, therefore containing the residuals and quaternion errors for training the ML models. In total, the final training dataset contained 310,000 samples, 100,000 from the dataset 1, and 210,000 from the dataset 2.

TABLE 2. Different parameters for generating the training dataset.

Datasets	**Initial orientation ($\mathbf{q_0}$)**	**τ (sec)**	**Simulation time (sec)**
Dataset 1	[0,0,0,1]	1000	2000
Dataset 2	$\hat{\mathbf{n}}$ = [[0,0,1], [0,1,0], [1,0,0]]	1000, 500, 0, -500, -800	70
	α = [-90^o, -45^o, 45^o, 90^o]		

3.2. **ML Models Training and Selection Results.** In this work, Python Keras 2.4.3/Tensorflow 2.3 alongside with scikit-learn 1.0.2 were used for training, testing and validation the ML models. Nested Cross-Validation (CV) (see 6) was used for selecting the models' hyper-parameters, and subsequently evaluate the performance of each approach. The hyper-parameters tested are summarized as follows:

- **Artificial Neural Network (ANN):**
  - Number of hidden layers: [1, 2, 3]
  - Batch size: [50, 100]
  - Number of neurons in each hidden layer: [150, 200, 300]
  - Dropout rate in between each layer: [10%, 20%]
- **Decision Tree (DT):**
  - Max depth: [50, 100, 300, 500]
  - Min samples leaf: [10, 25, 50]
- **Random Forest (RF):**
  - Number of estimators: [25, 50, 100]
  - Max depth: [50, 100, 300]
  - Min samples leaf: [10, 25, 50]

The "batch size" for the ANN training is the number of inputs that will be considered before the ANN weights are updated. The "dropout" layers are used only while training, and basically they randomly eliminate at each step a percentage

of neurons to be considered, which helps preventing overfitting. The optimization algorithm used was "Adam", and the rectified linear ("ReLU") was set as activation function for the neurons. For the DT model, the "max depth" limits the depth of the tree, whereas "min samples leaf" defines the minimum number of samples required to be at a leaf node. These hyper-parameters help to smooth the model, preventing overfitting. The hyper-parameters tested for the RF model were the same as for the DT with addition of the "number of estimators", that defines the number of trees in the model. This range of hyper-parameters was chosen through experimentation, trying to achieve the best performances with a reasonable computation time, that will be further discussed.

Table 3 shows the results of the nested CV method. An outer loop using a 5-fold CV was used for measuring the ML models' performances, considering the mean squared error (MSE) as metric, whereas an inner-loop consisting of a 3-fold CV was used to select the best hyper-parameters for each model. The scikit-learn function "GridSearchCV" was used to accomplish this task. This function performs the CV, testing exhaustively the combinations of hyper-parameters under test. Due to the large training dataset obtained, and the limited hardware resources available, just half of the samples were used in the nested CV process, i.e. the training dataset was copied taking one sample and ignoring the next.

TABLE 3. Cross-validation results of the implemented ML models.

ML model	**Average MSE**	**Best hyper-parameters**
ANN	0.012	Hidden layers: 3 Neurons: 200 Batch size: 100 Dropout: 10%
DT	0.008	Max depth: 50 Min samples leaf: 10
RF	0.008	Number of estimators: 50 Max depth: 100 Min samples leaf: 10

As Table 3 shows, the average MSE of the three models were similar, however the RF and DT performed better. Using the best hyper-parameters obtained in the nested CV process, the models were trained with the full training dataset. As the CV was made shuffling the training dataset, the samples used in each fold to evaluate each ML model were not the same. Because of this, the performances given by the average MSE may be biased. For approaching this issue, a testing dataset was created to evaluate the generalization performances of the NQKF coupled to the ML models.

3.3. **ML-based NQKF for Soft Error Mitigation Results.** The trained ML models were used in the framework of the NQKF, following the schema illustrated in Figure 2. For testing the generalization of the resulting algorithms obtained through the CV procedure, a testing dataset was created. This testing dataset is based on the same orbit and sensors' models previously defined, however, now

the sensors are sampled with a frequency of 50 Hz for estimation purpose. Different combinations of initial satellite's positions in the orbit and orientations are considered. Remembering that the objective of the orientation is always pointing the negative x_B to the center of the Earth, keeping z_B perpendicular to the orbit's plane. Table 4 shows the parameters for the testing dataset.

TABLE 4. Parameters for creating the testing dataset.

Initial orientation ($\mathbf{q_0}$)	$\boldsymbol{\tau}$(sec)
$\hat{\mathbf{n}}$ = [[1,0,0], [0,1,0], [0,0,1], $[1,1,1]/\sqrt{3}$, $[0,1,1]/\sqrt{2}$ $[1,0,1]/\sqrt{2}$, $[1,1,0]/\sqrt{2}$]]	600, 300, 0, -300, -600
$\alpha = 90^o$	

In total 35 simulations of 60 sec composed the testing dataset, which were executed 50 times for each ML-based NQKF while SEUs were injected in a component of the estimated quaternion. The time instant, the quaternion component, the amount of bit flips, and the target bits (considering only signal and exponent bits) were chosen randomly. However, for the three ML models, SEUs were identical, i.e. they were injected at the same time, quaternion component, and bits in the simulation. Moreover, it was ensured that the ML models were activated. For the sake of having a better interpretation, the quaternions were converted into Euler angles (following the Yaw-Pitch-Roll sequence), and the performance was measured throughout the RMSE (Equation 3.1) and the mean absolute error (MAE) calculated as:

$$MAE = \sum_{i=1}^{N} \frac{|y_i^{true} - y_i|}{N}, \tag{3.2}$$

where y_i^{true} represents the Euler angle obtained by converting $\mathbf{q_{true}}$. The term y_i either represent the Euler angle obtained by converting $\mathbf{q}$ from the simulation of the ML-based NQKF under SEU effects or the Euler angle calculated only with the NQKF under the same effects. N represents the number of points in the simulation. The simulations have 60 sec each, with a sampling time of 0.02 sec (50Hz), $N = 3000$. Moreover, the RMSE and MAE were averaged considering the total number of executions of the simulations in the testing dataset (35 different simulations $\times$ 50 executions of each simulation = 1750 total executions for each learning-based NQKF). The results are shown in Table 5.

Each column in Table 5 presents a different combination of the NQKF with a ML technique for the metrics relative to each respective Euler angle in each row. The values shown inside the parenthesis represent the error reduction of the ML-based NQKF compared to the second column, where only the NQKF was acting under the SEU effects. The three techniques acted very well, reducing both the MAE and the RMSE in a similar percentage compared to the NQKF alone. The MAE reduction for the ML-based NQKF reached a maximum of

TABLE 5. Testing results of the ML models-based NQKF under SEU effects.

Errors(^{o})	**NQKF**	**NQKF+NN**	**NQKF+DT**	**NQKF+RF**
RMSE ϕ	5.65	2.961(-47.6%)	2.999(-46.9%)	2.941(-47.9%)
RMSE θ	2.65	1.442(-45.6%)	1.37(-48.3%)	1.365(-48.5%)
RMSE ψ	5.78	2.516(-56.5%)	2.375(-58.9%)	2.363(-59.1%)
MAE ϕ	0.64	0.158(-75.4%)	0.157(-75.5%)	0.155(-75.7%)
MAE θ	0.54	0.097(-82.0%)	0.095(-82.5%)	0.095(-82.5%)
MAE ψ	0.65	0.134(-79.5%)	0.127(-80.5%)	0.127(-80.5%)

75.7%, 82.5%, and 80.5% for the angles ϕ, θ, and ψ, respectively, considering all the ML models. For the sake of comparison and to demonstrate the efficiency of the technique, another recent AE algorithm was also implemented within the ML-based framework. The algorithm described in [28] is an invariant extended Kalman filter (IEKF), which provides the attitude in the form of a rotation matrix. The implementation of the IEKF can be found in [29]. As the mitigation technique needs a quaternion attitude representation, the IEKF was normally executed, and its outputs were converted into quaternions. The same framework described in Figure 2 was implemented, as well as the same soft error injection performed for the NQKF (see Table 5) was applied to generate the results in Table 6.

TABLE 6. Testing results of the ML models-based IEKF under SEU effects.

Errors (^{o})	**IEKF**	**IEKF+NN**	**IEKF+DT**	**IEKF+RF**
RMSE ϕ	3.924335	2.07(-47.2%)	2.05(-47.8%)	2.22(-43.5%)
RMSE θ	1.875859	1.12(-40.1%)	1.07(-42.9%)	1.04(-44.5%)
RMSE ψ	3.734788	2.22(-40.4%)	1.79(-52.0%)	1.84(-50.7%)
MAE ϕ	0.503304	0.16(-68.5%)	0.15(-71.0%)	0.15(-70.2%)
MAE θ	0.252329	0.08(-67.1%)	0.08(-69.0%)	0.07(-70.3%)
MAE ψ	0.447838	0.16(-64.3%)	0.13(-71.5%)	0.13(-70.6%)

In the first column of Tables 5 and 6, only the AE algorithms were executed under SEUs (without mitigation), we can see that the IEKF achieved lower RMSE and MAE compared to the NQKF. But the final errors, when both algorithms were implemented within the mitigation technique, were similar. The NQKF is a classic linear AE algorithm, being simple to implement and fast to compute, since the traditional linearization of the measurement equations are not necessary. The IEKF is a recent algorithm successfully applied for AE, gyrocompass, and camera localization [28], for example. However, its implementation and the concepts behind the algorithm, which rely on Lie groups and the SO(3) orthogonal groups, are quite complex and request higher computation time and cost, than the NQKF. It is clear from this comparison that the NQKF, even if it becomes old in the domain, is still competitive and efficient (especially when coupled to the mitigation technique) and can continue to be used successfully within this framework.

The effects of the SEUs on the NQKF and IEKF are characterized as a peak when the soft error occurs, and a settling time where the algorithm tries to recover. Figure 7 shows an example of the SEU effects. In the schema of the proposed work (see Fig. 2), the star sensors were used for correction and estimation of the state vector, followed by the injection of the soft errors in the real component q_4. Then the accuracy brought by these sensors in the estimation/correction step is degraded by the injected soft errors after. Since the RMSE in the two Tables are calculated including the observed peaks, the attitude errors from the learning-based NQKF or the learning-based IEKF are little larger than the accuracy of the star sensors. Without the injection of soft errors, the attitude errors are found to be in the same range of star sensors' accuracy.

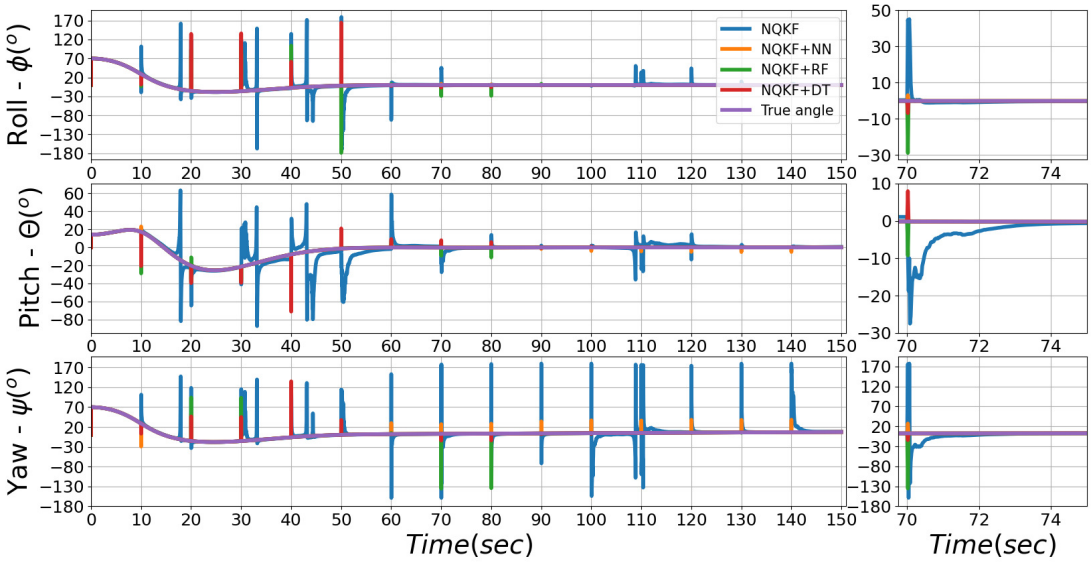

FIGURE 7. Illustration of the SEU effects on the different ML models-based NQKF.

Figure 7 is an example in which multiple SEUs were injected. It is important to highlight that is very unlikely that multiple sequential SEUs will occur like in this example, however this is just a theoretical simulation to illustrate the SEU effects on different time instants of the orbit, and how the proposed technique acts mitigating these effects. The true angle represents the ideal result of the NQKF algorithm, without any compensation of the soft error effects, and the other curves represent the ML trained models-based NQKF. The SEU injected in the real component of the estimated quaternions q_4, leads to change different exponent or signal bits, which effects were reflected in the Euler angles obtained by converting the estimated quaternions $\mathbf{q}$. The time between each injected SEU was 10 sec. Observing the blue curve, where just the NQKF was acting, each SEU provoked a huge peak, resulting in large instantaneous errors. Most errors lasted about 3 sec or even less before converging to zero, what explains why the values of the MAE and RMSE are relatively low. For generating Table 5, simulations of 60 sec were considered, in which just one SEU was applied. As the errors naturally

do not last more than 5 sec, the average error will be low. However the error magnitudes surpassed 100° in many cases in Figure 7, which cannot be neglected. Considering the other curves representing the ML models-based NQKF, all the combinations practically mitigated completely the SEU effects after 50 sec of simulation, period where the satellite's orientation varied very slowly. During the transitory phase, where the satellite was moving from a random orientation trying to point to the center of the Earth, the performance of ML techniques is less, specially the RF and DT, however in most cases better than the NQKF alone. Here lies the importance of the block "Evaluation" in Figure 2. The function of this block is to activate or deactivate the ML model, for reducing the overload, and also possible deterioration of the AE performance in case of multiple activation of the ML models. This block monitors the residuals magnitude, calculated using the star tracker information of the two stars considered. It was found throughout simulations that when any of the two norms of residual's vectors was bigger than 0.1, the ML models should be activated. This threshold provided was used for producing the results in Table 5 and in Figure 7.

4. Implications of the Soft Error Mitigation Technique

An AE system does not act independently. Figure 8 shows a schema of an ADCS, considering the AE and the control system.

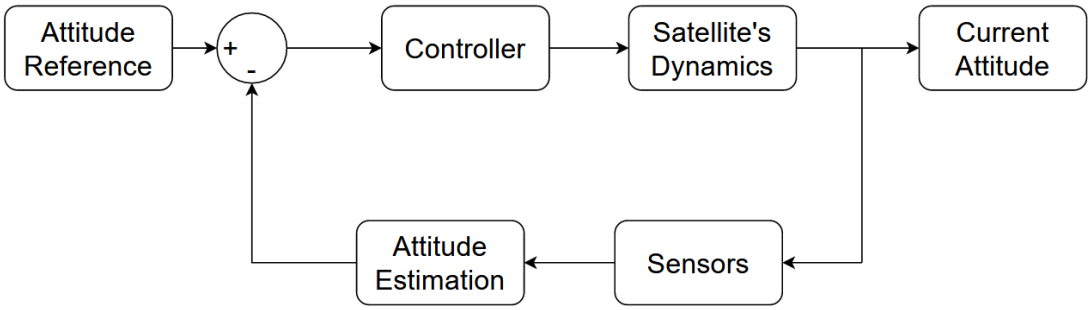

Figure 8. Block diagram of ADCS.

Basically the objective of the ADCS is to determine and control the satellite's attitude based on a target/reference. The AE block provides outputs, calculated based on the sensor's measurements, that will be compared to the reference and the error between them will serve as input for the control system. The control system will send commands to the actuators, finally controlling the satellite's orientation. In this paper, only the AE algorithm under soft error effects was studied. When the SEU disturbs directly the AE algorithm, it can cause large errors that will be passed to the control system. The effects of these errors on the control part need to be investigated, because the information that the control system receives may be largely wrong, during the SEU effect periods, capable of leading to dangerous scenarios in safety critical applications, such as satellites. Another point important to be mentioned is the overhead caused by the ML techniques in the AE algorithms. The impact of the ML in the processing time

of the AE algorithms will be noticed just when the residuals norm surpassed the threshold. The average execution time of the trained models is about 0.12 ms for the NN, 7.7 ms for the RF, and 0.06 ms for the DT, whereas one iteration of the NQKF is about 1.47 ms in the machine used for generating the results. The NN and DT techniques are much faster than the NQKF itself, about 11 and 23 times respectively, unlike the RF. One solution would be to simply run again the NQKF iteration in which a large residual error was observed. However, for this it would be necessary to use values stored in the RAM memory from the previous iteration. The SEUs are completely random, so it is difficult to identify which variable is corrupted, being possible to use again a corrupted parameter in the new NQKF iteration. The advantage of the proposed soft error mitigation technique is that it only needs the orientation error information contained in the residuals, not being necessary to monitor or protect the Kalman filter's variables.

The chances of a particle strike provoke a soft error in the AE algorithm's calculation are much higher than in the residuals or in the ML calculation, since the execution time of the Kalman filter-based algorithm is much higher, hence its variables remain more time exposed to the radiation environment. However, if a soft error occurs during the residuals calculation, a simple solution would be to perform this step twice, and then to compare the residuals to decide if the residuals are corrupted. Otherwise, if a soft error occurs in the ML calculations, the new residual vector can be calculated with the corrected quaternion, provided by the ML model, and compared with the first residuals (before the ML correction). Some actions could be taken, for example, if the new residuals' norm are higher than the previous one, the old quaternions could feed the next AE algorithm's iteration, or the ML correction could be performed a second time.

5. Conclusions

In this work three machine learning models were assessed in a framework for mitigation of soft error effects in the AE processing, more specifically SEU effects. The results show that the learning-based technique combined with an AE algorithm based on the Kalman filter, accomplished very well the task of minimizing the SEU effects, thus reducing the risks of passing large errors to the control system. The performances of the trained neural network, decision tree, and random forest models were quite similar, while adding little overhead to the AE algorithm's computation. Increasing the variety of samples data in the training dataset, including different initial orientations and positions in the orbit, probably will enhance the generalization capacity of the models, hence improving the SEU effects mitigation mainly in the phase in which the attitude varies more abruptly. Furthermore, the re-initialization of the error covariance matrix in this paper was made by only setting this matrix as a constant multiplied by an identity matrix. Re-initializing the error covariance matrix with values that reflect better the current states of the system is another point of improvement of the mitigation framework.

6. Acknowledgements

This work was supported in part by: the MultiRad project of the région Auvergne-Rhône-Alpes's international ambition pack (PAI); the French national research agency within the France-2030 program (ANR-15-IDEX-0002); the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01); the IRT Nanoelec (ANR-10-AIRT-05) of the French national program PIA ("Programme d'Investissements d'Avenir).

Appendix I: Overview on the selected Machine Learning (ML) Models and Cross-Validation (CV)

This section discusses three ML models commonly used for both supervised classification and regression tasks: Artificial Neural Networks, Decision Trees, and Random Forests. Moreover, an overview of Cross-Validation for ML model selection and evaluation is also presented.

Artificial Neural Networks (ANN). An ANN, more specifically a multi-layer perceptron (MLP) consists of neurons arranged in layers and connected by weights. The output of each layer serves as input to the next one. The inputs are multiplied by the weights and passed through activation functions. The MLP architecture, which combines nonlinear activation equations in each neuron, allows the approximation of extremely complex functions [30]. To accomplish this task it is necessary to train the network, i.e. adjust the weights and biases of each layer so that the ANN is able to connect a set of input vectors (input dataset) to a set of associated output vectors (output dataset). Backpropagation is the most used algorithm for training neural networks [31]. In simple terms, the weights are initialized, then a set of input vectors are forward propagated through the neural network generating predictions. The error between the predictions and the actual output vectors is accumulated in a cost function. It is then backward propagated, obtaining the gradients of the cost function with respect to the weights and biases. The weights are adjusted trying to minimize the overall error, throughout an optimization algorithm. ANNs may present a problem called overfitting, that is when the model fits very well the training dataset, however it cannot perform well on new data. Moreover, underffiting problems may occur, that unlike overfitting, the model does not fit well even the training dataset. Regularization techniques, different model structures, and more data might be required for solving this kind of problems.

Decision Trees (DT). The DT is a ML algorithm that can perform both classification and regression tasks, and it is capable of training using complex datasets [32]. It is composed of nodes and leaves. Each node has a decision boundary for a determined feature. The DT structure, e.g. amount of nodes, tree height, and leaves, is determined during the DT training. Usually the Classification And Regression Tree (CART) algorithm [32] is used for training DT. Basically it starts splitting the training dataset in two, based on the feature and threshold that minimizes a cost function, that is usually the mean squared error for regression tasks. The split subsets samples are assigned to its respective nodes, and the same logic is applied recursively until a leaf is produced (due to constraints or

the algorithm cannot reduce the cost). This algorithm is prone to overfitting, and because of this, the constraints in the training may be applied, such as limiting the tree maximum depth, and setting the minimum number of samples for splitting the nodes. For making a prediction with the training tree, the input needs to traverse the DT nodes, being conducted by the decision rules of each node, until it reaches a leaf. The resulting output, for regression tasks, will be the average of the samples in the leaf.

Random Forests (RF). The RF model uses a technique called model ensembling, that consists of combining multiple fundamental models in order to have better predictions [33]. In the case of RF regression, multiple decision trees are used as base models. Each tree is trained independently, as previously described, and a method called bootstrap aggregating (bagging) is applied. In this method, random sub-samples with replacement of the input dataset are used to build the decision tree structure (creation of decision nodes and leaves). For RF regression, the predictions of the independent trees are averaged, providing the final prediction. The bagging method provides to the RF training datasets with more randomness, which can improve the generalization.

Cross-Validation for ML model selection. When training a ML model, a set o parameters and hyper-parameters need to be adjusted so that the model will be able to better fit the data. The training should be adapted in such a way the model has enough examples to train on, but at the same time avoid overfitting the training data, being able to make predictions on unseen data. Cross-validation (CV) is the most common method used for evaluate the performance of ML models [34]. Furthermore, CV can be used for model selection, in which several ML models are compared and the one with the best performance is selected [35]. The CV consists of splitting the data, in which part of it is be used for training and the remaining for testing and performance measurement. Many different CV procedures exist, being one of the most common called k-fold CV. In this method the data is divided into k folds of the same size. Training and testing are done in k iterations, in which k-1 folds are used for training, leaving one fold for testing. The performance in each iteration is calculated and the results are averaged, providing the average model performance.

Nested CV is a method that consists on using two CV loops, one inside the other. The inner loop is used to select the best model hyper-parameters in a given set, whereas the outer loop computes the error. Instead of using CV for estimating the performance of a single ML model, nested CV estimates the performance of an optimized model, and it is a method that reduces the bias considerably [36], providing a better estimate of the true error of a ML model. After selecting the model through CV, the model parameters are then calculated using the entire dataset for training.

References

[1] "Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray Induced Soft Errors in Semiconductor Devices," Jan. 2012. [Online]. Available: https://www.jedec.org/standards-documents/docs/jesd-89a

[2] Y. Lu, Q. Shao, H. Yue, and F. Yang, "A review of the space environment effects on spacecraft in different orbits," *IEEE Access*, vol. 7, pp. 93 473–93 488, 2019.

[3] D. Wilkinson, S. Daughtridge, J. Stone, H. Sauer, and P. Darling, "TDRS-1 single event upsets and the effect of the space environment," *IEEE Transactions on Nuclear Science*, vol. 38, no. 6, pp. 1708–1712, 1991.

[4] K. M. Girgis, T. Hada, S. Matsukiyo, and A. Yoshikawa, "Radiation analysis of leo mission in the south atlantic anomaly during geomagnetic storm," *IEEE Journal of Radio Frequency Identification*, vol. 6, pp. 292–298, 2022.

[5] E. Petersen, "Soft errors due to protons in the radiation belt," *IEEE Transactions on Nuclear Science*, vol. 28, no. 6, pp. 3981–3986, 1981.

[6] J. Crassidis, L. Markley, and Y. Cheng, "Survey of nonlinear attitude estimation methods," *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM*, vol. 30, pp. 12–28, 2007.

[7] A. Sabatini, "Quaternion-based extended Kalman filter for determining orientation by inertial and magnetic sensing," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 7, pp. 1346–1356, 2006.

[8] X. Tong, Z. Li, G. Han, N. Liu, Y. Su, J. Ning, and F. Yang, "Adaptive EKF Based on HMM Recognizer for Attitude Estimation Using MEMS MARG Sensors," *IEEE Sensors Journal*, vol. 18, no. 8, pp. 3299–3310, 2018.

[9] D. Choukroun, I. Bar-Itzhack, and Y. Oshman, "Novel quaternion Kalman filter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 1, pp. 174–190, 2006.

[10] T. Kraemer Sarzi Sartori, H. Fourati, M. Letiche, and R. Possamai Bastos, "Assessment of radiation effects on attitude estimation processing for autonomous things," *IEEE Transactions on Nuclear Science*, vol. 69, no. 7, pp. 1610–1617, 2022.

[11] T. Kraemer Sarzi Sartori, H. Fourati, M. Garay Trindade, and R. Possamai Bastos, "Assessment of attitude estimation processing system under neutron radiation effects." in *Conference on Radiation Effects on Components and Systems (RADECS)*, 2021.

[12] T. Kraemer Sarzi Sartori, H. Fourati, A. Justus Rajappa, P. Reiter, and R. Possamai Bastos, "Effectiveness of attitude estimation processing approaches in tolerating radiation soft errors." in *Conference on Radiation Effects on Components and Systems (RADECS)*, 2022.

[13] X. Han, Y. Hu, A. Xie, X. Yan, X. Wang, C. Pei, and D. Zhang, "Quadratic-kalman-filter-based sensor fault detection approach for unmanned aerial vehicles," *IEEE Sensors Journal*, vol. 22, no. 19, pp. 18 669–18 683, 2022.

[14] X. Lyu, B. Hu, K. Li, and L. Chang, "An adaptive and robust ukf approach based on gaussian process regression-aided variational bayesian," *IEEE Sensors Journal*, vol. 21, no. 7, pp. 9500–9514, 2021.

[15] P. Li, W.-A. Zhang, and J.-H. Zhang, "Hmm based adaptive kalman filter for orientation estimation," *IEEE Sensors Journal*, vol. 22, no. 17, pp. 17 065–17 074, 2022.

[16] A. Makni, H. Fourati, and A. Y. Kibangou, "Energy-aware adaptive attitude estimation under external acceleration for pedestrian navigation," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 3, pp. 1366–1375, 2016.

[17] A. Makni, A. Kibangou, and H. Fourati, "Data Fusion-Based Descriptor Approach for Attitude Estimation underaccelerated maneuvers," *Asian Journal of Control*, vol. 21, no. 4, pp. 1433–1442, Jul. 2019. [Online]. Available: https://hal.science/hal-01982463

[18] J. Wu, Z. Zhou, J. Chen, H. Fourati, and R. Li, "Fast complementary filter for attitude estimation using low-cost marg sensors," *IEEE Sensors Journal*, vol. 16, no. 18, pp. 6997–7007, 2016.

[19] L. Dominik, "System mitigation techniques for single event effects," in *2008 IEEE/AIAA 27th Digital Avionics Systems Conference*, 2008, pp. 5.C.2–1–5.C.2–12.

[20] Y. Tang, L. Hu, Q. Zhang, and W. Pan, "Reinforcement Learning Compensated Extended Kalman Filter for Attitude Estimation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 6854–6859.

[21] M. K. Al-Sharman, Y. Zweiri, M. A. K. Jaradat, R. Al-Husari, D. Gan, and L. D. Seneviratne, "Deep-learning-based neural network training for state estimation enhancement: Application to attitude estimation," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 1, pp. 24–34, 2020.

[22] L. Xiang, L. Xiaoqin, and L. Yaohua, "Attitude estimation based on recurrent neural network and vector observations for attitude and heading reference system," in *2019 14th IEEE International Conference on Electronic Measurement & Instruments (ICEMI)*, 2019, pp. 1382–1387.

[23] Y. Liu, Y. Zhou, and X. Li, "Attitude estimation of unmanned aerial vehicle based on LSTM neural network," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, ISSN: 2161-4407.

[24] M. Zmitri, H. Fourati, and C. Prieur, "Bilstm network-based extended kalman filter for magnetic field gradient aided indoor navigation," *IEEE Sensors Journal*, vol. 22, no. 6, pp. 4781–4789, 2022.

[25] A. Tewari, *Atmospheric and Space Flight Dynamics: Modeling and Simulation with MATLAB and Simulink*, 1st ed. Birkhäuser Basel, 2007.

[26] JEDEC, "Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray Induced Soft Errors in Semiconductor Devices," Sep. 2021. [Online]. Available: https://www.jedec.org/standards-documents/docs/jesd-89a

[27] M. Maniatakos, P. Kudva, B. M. Fleischer, and Y. Makris, "Low-cost concurrent error detection for floating-point unit (fpu) controllers," *IEEE Transactions on Computers*, vol. 62, no. 7, pp. 1376–1388, 2013.

[28] A. Barrau and S. Bonnabel, "Three examples of the stability properties of the invariant extended kalman filter **this work is supported by the company safran." *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 431–437, 2017, 20th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896317300782

[29] "Ghadeer-SHAABAN/AttitudeEstimation-IEKF," https://github.com/Ghadeer-SHAABAN/AttitudeEstimation-IEKF/tree/main, Accessed on: 2023-08-30.

[30] M. W. Gardner and S. R. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," vol. 32, no. 14, pp. 2627–2636.

[31] N. Paeedeh and K. Ghiasi-Shirazi, "Improving the backpropagation algorithm with consequentialism weight updates over mini-batches," *Neurocomputing*, vol. 461, pp. 86–98, 2021.

[32] A. Géiron, *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems*, 2nd ed. O'Reilly, 2019.

[33] M. S. Acharya, A. Armaan, and A. S. Antony, "A comparison of regression models for prediction of graduate admissions," in *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, pp. 1–5.

[34] S. Yadav and S. Shukla, "Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification," in *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, pp. 78–83.

[35] Y. Zhang and Y. Yang, "Cross-validation for selecting a model selection procedure." *Journal of Econometrics*, vol. 187, pp. 95–112, 2015.

[36] S. Varma and R. Simon, "Bias in error estimation when using cross-validation for model selection." *BMC bioinformatics*, vol. 7, p. 91, 2006.

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, TIMA, 38000, Grenoble, France.

[2] Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, GIPSA-Lab, 38000, Grenoble, France.

Email address: tarso.kraemer-sarzi-sartori@univ-grenoble- alpes.fr, hassen.fourati@gipsa-lab, rodrigo.bastos@univ-grenoble-alpes