Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

NEURAL NETWORKS FOR GNSS MATRIX ATTITUDE DETERMINATION IN AEROSPACE TRANSPORTATION

Raul de Celis
Jose Gonzalez-Barroso
Pablo Solano-Lopez
Luis Cadarso

Aerospace Systems and Transport Research Group
Rey Juan Carlos University
Camino del Molino 5
Fuenlabrada, Madrid 28942, SPAIN

ABSTRACT

Accurate navigation and control of Aerial Vehicles requires precise estimations of their position and attitude. Measuring an aircraft's rotation involves comparing two vectors in different reference frames, such as inertial and body axes. Typically, a GNSS sensor-based matrix with at least three sensors is utilized for this purpose, taking advantage of the carrier phase measurements. However, factors such as multipath, frequency lock loss, cycle slips, and severe clock drifts can impede accurate integer ambiguity resolution. To address these challenges, a new neural network-based technique has been developed to optimize the management of large amounts of data and increase carrier phase ambiguity resolution reliability. By using carrier phase difference and pseudorange information, various neural network configurations can be trained to solve the ambiguity and estimate the precise attitude of the GNSS sensor matrix. The provided solution can be used alone or hybridized with other attitude sensor such as gyroscope information.

1 INTRODUCTION

Using Global Navigation Satellite Systems (GNSS) as an alternative or complement to inertial navigation systems (INS) for attitude determination is an attractive option since a relatively common, and economically affordable system in aircraft, such as GNSS, can be redundantly used for other tasks apart from determining position, just introducing some relatively simple algorithms in terms of processing power into the on-board computers. Although INS can provide accurate positioning and attitude determination during brief GNSS outages, they are susceptible to drifts, biases, and colored noise that are difficult to model with precision. Therefore, GNSS is often used to calibrate INS and obtain a position estimate free of drift, and also can be used to determine attitude. This article focuses solely on determining attitude using GNSS measurements from low-cost, single-frequency receivers and antennas. However, these receivers have their own limitations, such as frequent phase jumps caused by unreliable tracking, severe code multi-path, significant antenna phase center variations, and clock oscillations that are too large to be canceled by double differencing. The use of a Neural Network-based estimator will also help to minimize (or nearly eliminate) these effects as it will be demonstrated along this paper.

Accurately determining attitude for low-cost aircraft applications requires a baseline length of approximately 1 meter, which makes carrier phase difference measurements optimal for attitude this purposes. However, carrier phase measurements are subject to uncertainty, and in recent years, significant effort has been devoted to developing reliable methods for resolving carrier phase ambiguities. These methods include merging GNSS and INS measurements, using multi-frequency linear combinations to improve ambiguity discrimination (Henkel and Günther 2012; Geng and Bock 2013), incorporating prior knowledge (Teunissen et al. 1997; Teunissen 2006) and utilizing multi-antenna systems (Li 2018).

To resolve carrier phase ambiguities, both code and carrier phase measurements from delay locked loop and phase locked loop are utilized. This article does not cover the classical integer search method, which is described in detail in Henkel and Günther (2012), Geng and Bock (2013), Xu et al. (2022), and using different methodologies as LAMBDA method in Joosten et al. (1999), Wang et al. (2022). The main innovation of the methodology presented in this paper is the absence of the need to know in detail the formulation or physics of the problem, since the neural network is in charge of adjusting the solution based on the experimental measurement data. The ability to measure phase differences between signals received from two or three baselines formed by multiple antennas is crucial for the performance of a GPS carrier-based attitude determination system. Currently, in GPS systems, phase differences can only be calculated within one wavelength, resulting in indeterminate phase differences corresponding to integer multiples of the wavelength. This creates ambiguity in attitude determination due to the unknown integer number of wavelengths of the carrier phase. Mathematical algorithms are required to resolve this integer ambiguity, relying on precise phase difference measurements from different satellites visible to all receivers at the same epoch.

When using non-dedicated low-cost GPS receivers for this purpose, the error in phase measurements increases significantly (Nie et al. 2020). This is caused by factors such as the receiver clock error or noise induced by the multi-path effect, eventually leading to failures in the ambiguity resolution algorithm. As non-dedicated low-cost GPS receivers lack a common oscillator, the carrier phase double-difference technique must be employed, which leads to noise amplification and an increase in the number of integer solutions (Hu et al. 2020). This results in excessive search time and high noise levels, decreasing reliability and availability, respectively, in attitude determination. Therefore, it is crucial to introduce an additional source of information in the system for real-time navigation aiding, suitable for most vehicle types.

Machine Learning (ML) is an emerging field of study that utilizes statistical algorithms to analyze and learn from data, with the goal of making predictions or decisions based on that data. In the context of aerospace engineering, ML algorithms can be applied to GNSS data to determine the integer in carrier phase measurements, which is essential for accurate attitude determination. One of the main advantages of using ML for this purpose is that it eliminates the need to have a deep understanding of the physical-mathematical foundations of aerospace vehicles (de Celis et al. 2021). This is because the ML algorithm can be trained or calibrated using input data, such as simulated or experimental data, to learn the relationship between the GNSS measurements and the integer ambiguity. Once the algorithm is trained, it can be used to determine the integer ambiguity in real-time, providing information that can be used in a guidance, navigation, and control (GNC) system (de Celis and Cadarso 2023). This can greatly improve the accuracy and reliability of the GNC system, which is essential for the safe operation of aerospace vehicles. Overall, the use of ML for determining the integer ambiguity in carrier phase measurements has the potential to revolutionize the field of aerospace engineering by providing a more efficient and effective method for attitude determination.

In order to achieve the goal of improving the current approaches for determining attitude using GNSS sensors and a neural network-based single-phase delay estimator, this study will employ an innovating variety of techniques and methods. Firstly, the researchers will review and analyze existing methods for attitude determination and identify areas for improvement. They will then develop and implement a neural network-based algorithm for estimating single-phase delay, which will improve the accuracy of angular measurements obtained from GNSS sensors. To evaluate the performance of the proposed method, the researchers will conduct experiments and simulations using both synthetic and real-world data. They will compare the accuracy of the proposed method with existing approaches for attitude determination, as well as evaluate its effectiveness when integrated into a guidance, navigation, and control system alongside other sensor measurements. The results of this study will have important implications for the development of more accurate and reliable systems for aerial vehicle navigation and control. By improving the accuracy of angular measurements obtained from GNSS sensors, the proposed method will enable more precise control of aerial vehicles, enhancing their safety and reliability. It will also contribute to the development of more advanced autonomous aerial vehicles, which are increasingly important in a variety of applications such as search and rescue, environmental monitoring, and transportation.

2 DESCRIPTION OF THE MODEL

This section aims to provide a detailed explanation of the model and sensor utilized to produce data for the neural networks. The determination of attitude can be achieved through mathematical calculations that identify two non-aligned vectors ($\overrightarrow{v_1} = [v_{1x}, v_{1y}, v_{1z}]$ and $\overrightarrow{v_2} = [v_{2x}, v_{2y}, v_{2z}]$) in two distinct triads. For this study, we consider the North-East-Down earth axes, which are denoted by the sub-index 'e' and have their center located at an arbitrary point on land. This triad has x_e pointing North, y_e pointing East, and z_e forming a right triad. Furthermore, we also consider the body fixed axes, denoted by the sub-index 'b', which have their center located at the center of gravity of the airship. This triad has x_b pointing forward, y_b pointing right wing, and z_e forming a right triad. A graphical representation of the two triads is provided in Figure 1 for clarity.

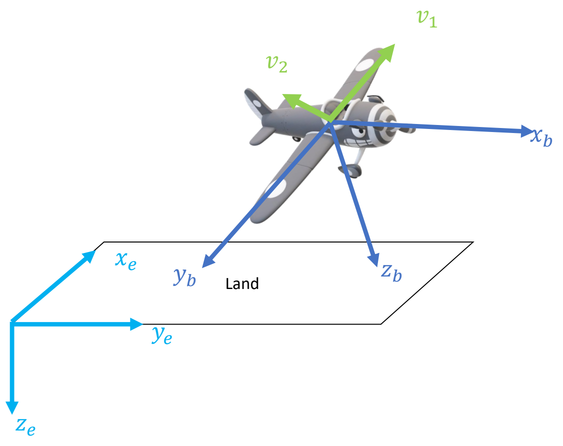


Figure 1: Earth and body axes.

The geometric plane defined by the two non-aligned vectors $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ is characterized by its associated normal vector $\overrightarrow{v_3} = \overrightarrow{v_1} \times \overrightarrow{v_2}$. To determine the rotation between the North-East-Down earth axes and the body fixed axes, a rotation matrix $R_{e,b}$ is used. To simplify the mathematical operations, the vectors $\overrightarrow{v_1}$, $\overrightarrow{v_2}$, and $\overrightarrow{v_3}$ are normalized and renamed as $\overrightarrow{u_1}$, $\overrightarrow{u_2}$, and $\overrightarrow{u_3}$, respectively. The rotation matrix can be calculated using equation (1), where the normalized vectors are arranged as columns. This method of determining attitude is crucial in aerospace vehicles as it allows for the precise determination of the vehicle's orientation in space. By using a rotation matrix, the vehicle's sensors can be aligned correctly, allowing for accurate measurements to be taken. This information can then be used in the guidance, navigation, and control system to make adjustments and ensure the vehicle's trajectory is on track. Overall, this approach provides a reliable and efficient way to determine attitude in aerospace vehicles.

$$R_{e,b} = \begin{bmatrix} u_{1x_e} & u_{2x_e} & u_{3x_e} \\ u_{1y_e} & u_{2y_e} & u_{3y_e} \\ u_{1z_e} & u_{2z_e} & u_{3z_e} \end{bmatrix} \cdot \begin{bmatrix} u_{1x_b} & u_{2x_b} & u_{3x_b} \\ u_{1y_b} & u_{2y_b} & u_{3y_b} \\ u_{1z_b} & u_{2z_b} & u_{3z_b} \end{bmatrix}^{-1} \tag{1}$$

In the context of an airship, the pair of vectors that determine attitude is represented by an array of GNSS sensors located on the body. The position of the body is determined by using at least three (and up to eight in this study) sensors, which are labeled as a_n for $n = 1,2,...,N$. These sensors collect position data from the Global Navigation Satellite System (GNSS) and transmit it to the onboard computer. Then one of the sensors (namely a_1) can be taken as a reference and use the others in order to calculate $\overrightarrow{u_1} = \frac{\overrightarrow{v_{a_1,a_2}}}{||\overrightarrow{v_{a_1,a_2}}||}$ to $\overrightarrow{u_{N-1}} = \frac{\overrightarrow{v_{a_1,a_N}}}{||\overrightarrow{v_{a_1,a_N}}||}$ in both systems of axes. This can be easily expressed as follows. The use of multiple GNSS sensors provides redundancy and improves the accuracy of the position determination. The number of sensors used in this study was chosen based on a trade-off between accuracy and cost. Increasing the number of sensors improves the accuracy of the position determination but also increases the cost and complexity of the system. According to this fact, equation (1) can be generalized for an infinite number of vectors as it is shown in equation (2):

$$R_{e,b} = \left(V_e \cdot V_e{}^T\right)^{-1} \cdot V_e \cdot V_b{}^T \text{ with } V_{axis} = \left[\left(\frac{\overrightarrow{v_{a_1,a_2}}}{||\overrightarrow{v_{a_1,a_2}}||} \right)^T \middle| \ldots \middle| \left(\frac{\overrightarrow{v_{a_1,a_N}}}{||\overrightarrow{v_{a_1,a_N}}||} \right)^T \right]_{axis}, \ axis = [e,b] \tag{2}$$

2.1 Data Generation

GNSS sensors are capable of measuring the carrier phase difference (ϕ) in GNSS wavelength, providing an approximate navigation precision of 0.003 m. However, determining the Carrier Phase Ambiguity or 'N', which is the product of GNSS wavelength and the integer number of cycles the wave travels from the satellite to the receiver, is a challenging task. Furthermore, several sources of errors can affect the navigation process, including satellite clock error (dT^{S_j}), sensor clock error (dT_{a_i}), random white noise from the sensor and satellite ($E_{a_i}^{S_j}$), and errors from the ionosphere and troposphere (E_{TI}) that affect each sensor and satellite. The primary objective of this navigation technique is to determine the pseudo-range ($\rho_{a_i}^{S_j}$), which represents the distance between sensor 'a_i' and satellite 'S^j'. Pseudo-range can be calculated as a function of several parameters, including ϕ, N, dT^{S_j}, dT_{a_i}, the speed of light 'c', wavelength 'λ', $E_{a_i}^{S_j}$, and E_{TI}. The mathematical expression for determining pseudo-range is given by equation (3).

$$\rho_{a_i}^{S_j} = \lambda\phi_{a_i}^{S_j} + \lambda N_{a_i}^{S_j} + c \cdot \left(dT^{S_j} - dT_{a_i}\right) + E_{a_i}^{S_j} + E_{TI} \tag{3}$$

In orientation problems, it may not be possible to determine certain errors. However, methods can be implemented to avoid such errors since they often have common sources. In the case of determining attitude, one such method involves defining the unit vector that points from the sensor a_i to the satellite S_j as $\overrightarrow{e}_{a_i}^{S_j}$, as it is shown in Figure 2. This vector is directly provided by the navigation data. As the distance between the sensors is much smaller compared to the distance to the satellite, the pointing vector can be considered independent of each sensor and can be expressed as a single vector $\overrightarrow{e}^{S_j}$. Using this, the vectors required for the attitude determination process can be calculated using equation (4).

$$\overrightarrow{v_{a_i,a_n}} = \rho_{a_i}^{S_j}\overrightarrow{e}_{a_i}^{S_j} - \rho_{a_n}^{S_j}\overrightarrow{e}_{a_n}^{S_j} = \left(\rho_{a_i}^{S_j} - \rho_{a_n}^{S_j}\right)\overrightarrow{e}^{S_j} \tag{4}$$

The difference between the pseudo-range measurements of two sensors observing the same satellite is known as a single difference. This difference can be expressed as $\left(\rho_{a_i}^{S_j} - \rho_{a_n}^{S_j}\right)$ and can be contracted as $\Delta\rho_{a_i,a_n}^{S_j}$. The expression for the single difference is given by equation (5), which is a function of the differences between the previously defined variables. It is important to note that the application of single differences eliminates errors caused by the ionosphere and troposphere, as well as satellite clock errors.

$$\Delta\rho_{a_i,a_n}^{S_j} = \Delta\phi_{a_i,a_n}^{S_j} + \Delta N_{a_i,a_n}^{S_j} - c \cdot \Delta dT_{a_i,a_n} + \Delta E_{a_i,a_n}^{S_j} \tag{5}$$

The same concept can be applied now by subtracting the single difference of satellite 'S_m from that of satellite 'S_j' in what is called double difference $\nabla\rho_{a_i,a_n}^{S_j,S_m}$. Note that applying double difference, GNSS sensor clock error is eliminated as it can be seen in expression (6).

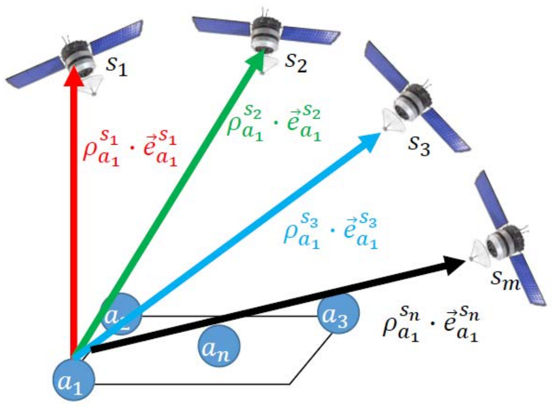

s_1
s_2
s_3
s_m
$\rho_{a_1}^{s_1} \cdot \vec{e}_{a_1}^{s_1}$
$\rho_{a_1}^{s_2} \cdot \vec{e}_{a_1}^{s_2}$
$\rho_{a_1}^{s_3} \cdot \vec{e}_{a_1}^{s_3}$
$\rho_{a_1}^{s_n} \cdot \vec{e}_{a_1}^{s_n}$
a_1
a_2
a_3
a_n

Figure 2: Example of pointing vector to each satellite.

$$\nabla\rho_{a_i,a_n}^{S_j,S_m} = \nabla\phi_{a_i,a_n}^{S_j,S_m} + \nabla N_{a_i,a_n}^{S_j,S_m} + \nabla E_{a_i,a_n}^{S_j,S_m} \tag{6}$$

To generate data for the neural network, a GNSS signal simulator such as the one included in Matlab can be used to obtain each of the terms of Equation (6) separately. These terms can then be used as data to train a neural network that will determine the double differences of the Carrier Phase Ambiguity, denoted as $\nabla N_{a_i,a_n}^{S_j,S_m}$. In other words, the input of the neural network consists of the combination of the double differences of the pseudo-range and error terms, given by '$\nabla\rho_{a_i,a_n}^{S_j,S_m} - \nabla E_{a_i,a_n}^{S_j,S_m}$', while the output of the neural network is the double differences of the Carrier Phase Ambiguity, represented as '$\nabla N_{a_i,a_n}^{S_j,S_m}$'. This neural network can then be used to estimate the values of $\nabla N_{a_i,a_n}^{S_j,S_m}$ for previously unseen data, finally, providing an accurate method for determining the attitude of the airship.

Finally, in the case under study, the set of inputs is generated for a combination of 8 sensors, taking one sensor as reference, and the other seven to generate the reference vectors, and 6 satellites present in the field of view. This configuration generates a set of 36 inputs (including the configuration for the satellites in the field of view as an input), and 35 outputs. It must be noted that, in order to simplify the amount of data, and reduce the training time, as this configuration will be sufficient for demonstrate the validity of the methodology, the simulation training data is particularized for a fixed geographic location (centered at URJC campus in Madrid) and for a determined period of time of 24 hours, but these assumptions will be generalized in further works.

The set of inputs is generated in 100 different simulations, with 5760 timesteps on each simulation, which means each timestep defined every 15 s of the simulated trajectory of the vehicle. The number of satellites is chosen based on how many of them are in sight, being the minimum available for the GNSS constellation 4 in a harsh location and 6 in a populated one. This facts are translated into a set of 36 different inputs during 5760 different timesteps and 100 different simulations. The 60% of the simulations is used for training, 20% for simulations and 20% for validation.

2.2 Machine Learning Architectures. A model for the Neural Networks.

Once the data-set has been described and introduced, the different Machine Learning (ML) algorithms will be introduced for this approach. One of the most important and most widely used machine learning techniques are neural networks. They are so varied that they can solve a large number of different problems. These networks would receive $\nabla\rho_{a_i,a_n}^{S_j,S_m} - \nabla E_{a_i,a_n}^{S_j,S_m}$ as input and return and unique $\nabla N_{a_i,a_n}^{S_j,S_m}$ as output. Therefore, as many neural networks as $\nabla N_{a_i,a_n}^{S_j,S_m}$ would be needed.

From now on, the set of trained neural networks with the same architecture and configuration of parameters specialized in outputting each $\nabla N_{a_i,a_n}^{S_j,S_m}$ will be called *model*. Since the possible values of $\nabla N_{a_i,a_n}^{S_j,S_m}$ are integers located within a finite range, the problem could be considered as both a regression (an output layer with a single neuron) or a classification task (an output layer with as many neurons as $\nabla N_{a_i,a_n}^{S_j,S_m}$ values, in case under study, 33 output neurons in High Quality, $\nabla N_{a_i,a_n}^{S_j,S_m} \in [-16, 16]$).

The simplest neural network consists on a multi-layer perceptron (MLP) (Rosenblatt 1958), that is, a neural network with multiple fully-connected/dense layers. Since consecutive inputs and outputs in an historical time are interrelated, it makes sense to apply a *slide time window* in each prediction. As an historical of data is received as input, it is also interesting to utilize RNN (recurrent neural networks). Figure 3 shows the expected performance of one network of the model (to predict all $\nabla N_{a_i,a_n}^{S_j,S_m}$ for a given pair of sensors and satellites). This network can do predictions in parallel with the rest of networks to get all $\nabla N_{a_i,a_n}^{S_j,S_m}$ in each time-step in real time. M is the number of instances per simulation, and N the number of simulations. In a simulation n, at each time-step, the network R receives an input in_m^n and the previous hidden state s_m^n of the recurrent neural network, and returns an output out_m^n and the updated hidden state s_{m+1}^n. Those hidden states s_m^n contain information about the previous inputs in_j^n, where $j < m$. In addition, s_1^i, where $1 \leq i \leq N$, is initialized to zero at the beginning of each simulation.

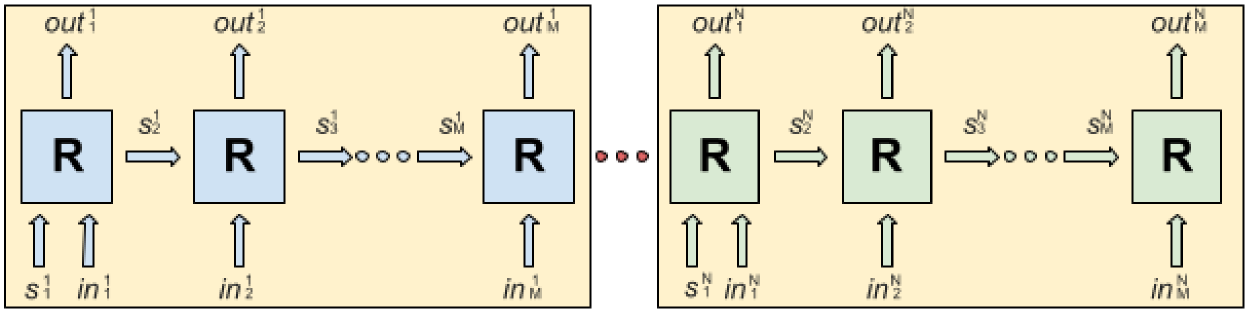


Figure 3: Recurrent neural network performance for a given pair of sensors and satellites.

The use of LSTM (Long short-term memory) (Hochreiter and Schmidhuber 1997) has been studied as it is the most widely used recurrent neural network, since it solves the short-term memory problem, which appears in simpler recurrent networks. This problem means that the network progressively forgets information about the inputs throughout the iterations, giving more importance to the last ones. However, for the current problem, it is more interesting the most recent information in the timeline, so the short-term memory problem is not and impediment for this concrete case. For these reasons, such recurrence layer could be based on a simple recurrent network. The rest of the network layers are fully connected (dense) and a dropout is also used as regularization before each dense layer to avoid over-fitting in both Simple RNN and MLP. Figure 4 shows our recurrent neural network structure where the number of Dense layers is 2 (for instance), including the output layer. Commonly, neural networks with more than three layers (including input and output layers) are called deep learning systems. Increasing the complexity of the neural network by adding more layers usually involves better results. The rest of parameters and architecture configurations (learning rate, number of layers, etc.) will be studied in 2.2.3.

2.2.1 Pre/Post- Processing

All inputs received by a neural network should be normalized beforehand in order to improve the performance of the resulting models (Sola and Sevilla 1997). This normalization transforms each input data into a value between 0 and 1, using the minimum and maximum value from among all the data obtained.

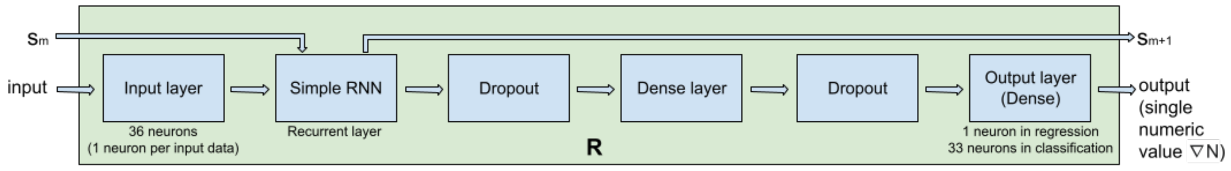


Figure 4: Recurrent neural network structure R with 36 inputs per time-step and 2 Dense layers.

For the training process it is also needed the corresponding output for each input. These outputs ($\nabla N_{a_i,a_n}^{S_j,S_m}$) must also be pre-processed (Sola and Sevilla 1997). Since $\nabla N_{a_i,a_n}^{S_j,S_m} \in [-16,16]$ in High Quality and $\nabla N_{a_i,a_n}^{S_j,S_m} \in \mathbb{Z}$, the neural network can be trained to solve both a classification or a regression task. If the task to be solved is a classification task, one-hot coding is used: output $o = \left(a_0...a_{32}\right) : a_i = 1$ if $i = \nabla N_{a_i,a_n}^{S_j,S_m} - min(\nabla N_{a_i,a_n}^{S_j,S_m})$ and $a_i = 0$ otherwise. Similarly, when prediction is obtained, it is necessary to interpret that value. The actual prediction can be expressed as $\widetilde{\nabla N_{a_i,a_n}^{S_j,S_m}} = argmax\left(a_0...a_{32}\right) + min(\nabla N_{a_i,a_n}^{S_j,S_m})$. During the regression task, the output is normalized as it is shown in (7). However, the normalization expression is different from the standard normalization in order to use the rounding function when de-normalizing without breaking equiprobability between $\nabla N_{a_i,a_n}^{S_j,S_m}$ values.

$$o = \frac{2(\nabla N_{a_i,a_n}^{S_j,S_m} - min(\nabla N_{a_i,a_n}^{S_j,S_m})) + 1}{2(max(\nabla N_{a_i,a_n}^{S_j,S_m}) - min(\nabla N_{a_i,a_n}^{S_j,S_m}) + 1)} \tag{7}$$

Similarly, the output o in each prediction must be de-normalized as it is shown in expression (8).

$$\widetilde{\nabla N_{a_i,a_n}^{S_j,S_m}} = \lfloor o * (max(\nabla N_{a_i,a_n}^{S_j,S_m}) - min(\nabla N_{a_i,a_n}^{S_j,S_m}) + 1) + min(\nabla N_{a_i,a_n}^{S_j,S_m}) - \frac{1}{2} \rceil \tag{8}$$

Figure 5 shows an example in case $\nabla N_{a_i,a_n}^{S_j,S_m} \in [0,1]$. The arrows indicate the value of $\nabla N_{a_i,a_n}^{S_j,S_m}$ to which each output converges when de-normalizing (applying rounding). This method of normalizing does not introduce an artificial precision that favors the success criterion defined in the next section, since any $\nabla N_{a_i,a_n}^{S_j,S_m}$ will have the same probability of being selected by generating a prediction uniformly randomly between 0 and 1.

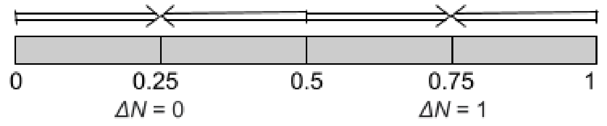


Figure 5: Output normalization in case $\nabla N_{a_i,a_n}^{S_j,S_m} \in [0,1]$.

2.2.2 Error Definition and Success Criteria

First of all, it is necessary to remember that each model is composed of a set of neural networks with the same architecture and parameters, but each one of them trained to predict a given $\nabla N_{a_i,a_n}^{S_j,S_m}$. However, these networks are analogous to each other. Therefore, to compare models, it will be only only taken a single network from each model as a reference (in the case under study it is the $\nabla N_{a_1,a_2}^{S_1,S_2}$. Therefore, the measures taken to establish the success criteria for each model are:

- **Accuracy**. Percentage of instances correctly predicted by the neural network.

- **Mean Square Error (*MSE*)**. In case of failure in the prediction. Small deviations from the exact value of the output.

The problem solved can be posed as both a classification and regression problem. After predicting and denormalizing we get numerical values that can be used to calculate the MSE, MSE in case of failure or the accuracy rate. However, during training, it must be used the corresponding metric (accuracy for classification and MSE for regression). The Mean Squared Error (MSE) is solely measured in the event of a failure. This metric assigns a value of 1 to the best-case scenario, representing the minimum deviation in predictions when they fail. The rationale behind this choice is our focus on models that exhibit high accuracy and, even in the event of failure, demonstrate minimal deviation from their true values.

2.2.3 Experimental Methodology

In this section the experimental methodology to obtain a model able to determine $\nabla N_{a_1,a_2}^{S_1,S_2}$ values will be described. Therefore, machine learning tasks, neural network types, architecture configurations and parameters will be listed and extended. It is important to mention that the main purpose is not to get the best model, but to demonstrate that the use of neural networks approaches are a good methodology to solve this particular problem. The optimization of the neural network selection process will be part of a further work. In addition, it is necessary to divide the experimentation process into several phases. In each phase, there will be set the rest of architecture configurations and parameters to specific values. Once a phase has determined which options are the best, subsequent phases will default to those options.

Some of those specific values by default are:

- Machine learning tool: *TensorFlow* in *Keras*.
- Amount of data in train, validation and test: 60%, 20% and 20% respectively.
- Number of Dense layers: 5.
- Number of neurons per Dense layer: 800, 400, 200, 100 and output layer size (33 in classification and 1 in regression) (from left to right).
- Activation function in intermediate layers: *ReLU*.
- Activation function in output layer in regression: *sigmoid*.
- Activation function in output layer in classification: *softmax*.
- Number of neurons in recurrent layer: 125.
- Activation function in recurrent layer: *ReLU*.
- Dropout rate: 0.1.
- Optimizer: *Adam*.
- Learning rate: 0.001.
- Batch size: 200 in MLP and 16 in RNN.
- Shuffle data in training: True.
- Epochs: 600.

There are 4 phases of experimentation. The first 3 phases will be focused on predicting only $\nabla N_{a_1,a_2}^{S_1,S_2}$, because the rest of the networks of the same model trained to predict $\nabla N_{a_i,a_n}^{S_j,S_m}$ would be analogous to this one. In addition, in these first three phases a checkpoint will be applied to save the best model found during training using the validation data set.

In **phase 1** there will be determined which machine learning task (regression or classification) and neural network type (recurrent network or multi-layer perceptron) are the best options to work in this problem. Each configuration will be run 3 times with different seeds. Since 3 different values of slide time window k will be tested (1, 5 and 10), and also one type of recurrent neural network (*SimpleRNN*), the number of different neural networks in this section will be 24. The baseline is a multi-layer perceptron with $k = 1$.

In **phase 2** there will be determined the best architecture and parameters among all those tested. The different dense layer architectures tested will be $(800,400,200,100)$, $(400,200,100,50)$ and $(200,100,50,25)$, indicating the number of neurons per dense layer (except output layer) from left to right. There will only be used 5 dense layers to apply deep learning (more than three layers including input and output layers) although the problem to be solved does not seem too complex. Since in phase 1 it will be chosen as the best network the recurrent one, it will be tested with different number of neurons (100, 125, 150, 175 and 200) and activation functions (*ReLU* and *tanh*) in the recurrent layer. As in the previous phase, it will be run each configuration 3 times with different seeds. Therefore, the number of different neural networks in this section will be 90.

In **phase 3** there will be determined the number of epochs and the learning rate to find the best neural network from the best architecture configuration and parameters found so far. It will be tested with 5 learning rates (0.01, 0.005, 0.001, 0.0005 and 0.0001) and with many epochs (2000). Again, it will be run each configuration 3 times with different seeds, and the number of different neural networks in this section will be 15.

Once the number of epochs and the learning rate have being adjusted, in **phase 4** it will be got the final model. In order to obtain the final result the validation set will be embedded into the training set, so the 80% will be used as training set and 20% as test set. By having more training data it will be expected to obtain better results, even without using checkpoint, just by saving the last model obtained in the training stage. After that, it will be gotten a neural network for each $\nabla N_{a_i,a_n}^{S_j,S_m}$. The networks that reach the reliability threshold (which has been set at 99% in accuracy rate in classification networks and 1.1 in *MSE* in case of failure in regression ones) will become part of the final model. Those that do not pass this threshold will be retrained from the beginning with the same configuration but using another seed. When all the networks in the model reach the reliability threshold, the experimentation ends.

2.2.4 Angular Restitution

The value for the double differences ($\nabla\rho_{a_i,a_n}^{S_j,S_m}$) can be determined from the use of GNSS carrier phase difference sensor measurements ($\nabla\phi_{a_i,a_n}^{S_j,S_m}+\nabla E_{a_i,a_n}^{S_j,S_m}$) and from the results of NN outputs ($\nabla N_{a_i,a_n}^{S_j,S_m}$) as it is deduced from equation (6). Applying double differences to expression (4) it can be obtained the following equation (9):

$$\nabla\rho_{a_i,a_n}^{S_j,S_m} = \overrightarrow{v_{a_i,a_n}}\cdot\left(\overrightarrow{e}^{S_j}-\overrightarrow{e}^{S_m}\right) \tag{9}$$

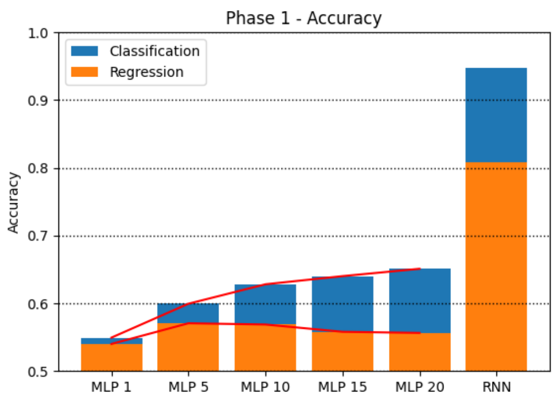
Equation (9) can be reformulated as a matrix equation (10), taking in account the signals from every satellite available:

$$\overrightarrow{v_{a_i,a_n}} = \left(H^T\cdot H\right)^{-1}\cdot H^T\cdot\nabla\overrightarrow{\rho},\ \nabla\overrightarrow{\rho} = \begin{bmatrix}\nabla\rho_{a_i,a_n}^{S_j,S_p}\\ \dots\\ \nabla\rho_{a_i,a_n}^{S_j,S_m}\end{bmatrix} \text{ and } H = \begin{bmatrix} e_x^{S_j}-e_x^{S_p} & e_y^{S_j}-e_y^{S_p} & e_z^{S_j}-e_z^{S_p}\\ \dots & \dots & \dots\\ e_x^{S_j}-e_x^{S_m} & e_y^{S_j}-e_y^{S_m} & e_z^{S_j}-e_z^{S_m}\end{bmatrix} \tag{10}$$

with $\overrightarrow{v_{a_i,a_n}}$ as a column vector, $\nabla\overrightarrow{\rho}$ as the column vector composed of the double differences of the reference satellite (j) with respect to the rest of available satellites ($p=[1,2,...,m]$ with $p\neq j$), and H as a matrix composed of the subtraction of the pointing row vectors of the reference satellite and the rest of available satellites.

3 RESULTS

This section will first show the results for the Neral Network selection problem and finally the final results applied to an angular restitution problem.



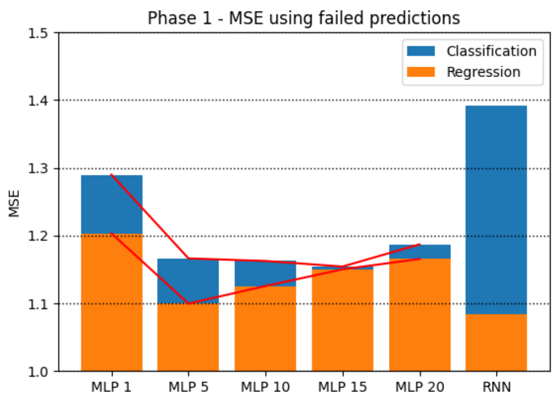


Figure 6: Left - Accuracy in MLP and RNN in classification and regression tasks. Right - *MSE* (in those predictions where the networks fails) in MLP and RNN in classification and regression tasks.

3.1 Comparison and Selection of the Final ML Model

Figure 6 (left) displays the results for the regression and classification accuracy of multi-layer perceptrons (MLP) with various slide time windows and for RNN. The classification networks exhibit superior performance compared to the regression counterparts, as they were trained to reduce categorical cross-entropy. The accuracy demonstrates an upward trend as the slide time window size increases, until it reaches a peak value. This is in line with the notion that past entries become less significant as time progresses, and the more inputs there are, the more difficult it becomes for the network to learn. However, increasing the time window excessively leads to a decline in accuracy. Regardless of the time window used, MLP accuracy will not reach the same level as RNN accuracy. Therefore, there is no reason to prefer MLP over RNN based on accuracy.

Figure 6 (right) illustrates the Mean Squared Error (MSE) in cases where the networks fail. The regression networks outperform their classification counterparts, as they were trained to minimize the MSE. Similar to the accuracy graph, the MSE decreases as the slide time window size increases until it reaches a minimum value. In the best-case scenario, the MSE would be 1, but none of the networks tested exceeded 1.4. The difference in performance between networks is minor. Regardless of the time slide window used, RNN for regression is the best network in terms of MSE, while RNN for classification is the worst.

Due to the great redundancy in the number of sensors, the classification networks have been chosen to carry out the angle restitution tests, since it is expected that with the redundancy in the available data, a vast majority of the data will have great precision, and the filtering algorithm is in charge of eliminating those results that have the worst MSE. For the density of the networks, the upper values listed in section 2.2.3 have been selected, with the aim of being tested in the angular restitution. Finally, the training have been carried out during 2000 epochs.

3.2 Angular Restitution Results

Monte Carlo simulations are utilized to assess the performance of the closed-loop system under various uncertainties, such as initial conditions, sensor data acquisition, atmospheric conditions, thrust features, and aerodynamic coefficients, where the coefficient uncertainties have been taken into account.

To demonstrate the effectiveness of the proposed attitude determination algorithm, simulation results are presented utilizing a non-linear flight dynamics model developed by (de Celis and Cadarso 2023). A set of nominal regular Euler angles are executed to compare the estimated and actual attitudes. The Euler angles are configures as a sine signal with a frequency of 1 rad/s and an amplitude of 20 deg. The simulations are performed using MATLAB/Simulink R2021a on a desktop computer with a processor of 2.8

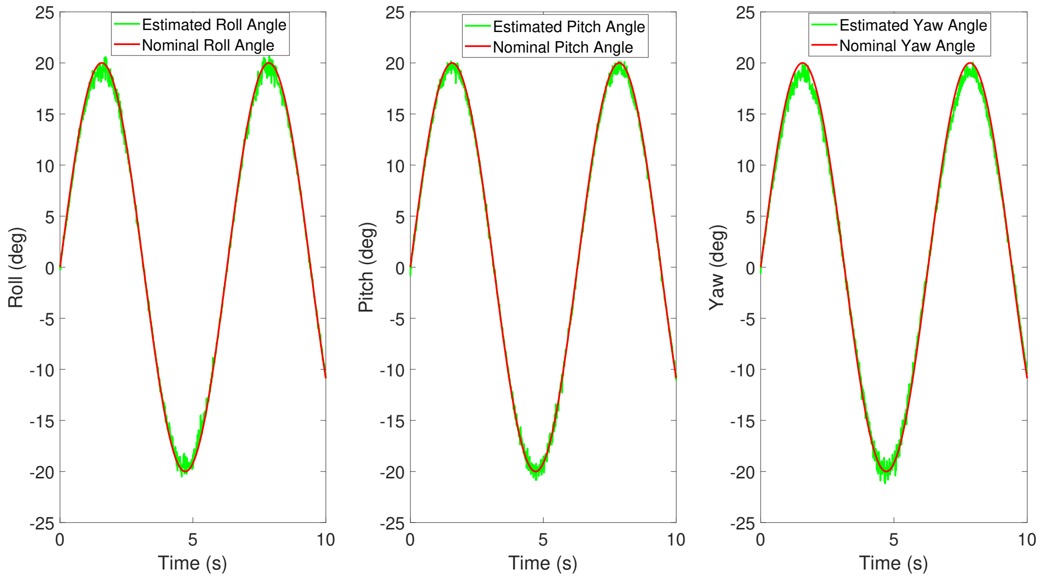


Figure 7: Roll (left) pitch (center) and yaw (right) angles comparison between nominal and estimation.

GHz and 8 GB RAM. The comparison between the estimated and actual attitudes for a nominal parabolic trajectory is shown in Figure 7.

The results consistently indicate accuracy levels below 2.5 degrees, which is considered sufficient based on current standards for utilizing the algorithm in the guidance, navigation, and control of low-cost aerial platforms.

4 CONCLUSIONS

A neural network-based approach has been developed for determining the N ambiguity in GNSS phase attitude calculation, which offers the potential for reduced costs and additional sources of information during GNC in aerospace platforms. Compared to other methods, this novel approach has been demonstrated to improve or match levels of accuracy. To validate the proposed approach, trajectory simulations were conducted to obtain "real" attitude, and the results were compared with the attitude obtained using the neural network method. The computational results show that the differences are negligible, making the proposed approach a viable option for use in a control algorithm, with total errors below 2.5 degrees for all three attitude angles in all cases. Moreover, the use of low-cost components makes this approach highly attractive for a range of low-cost professional applications.

ACKNOWLEDGMENTS

This research was supported by the “Rey Juan Carlos University” (Project Grant F919—TIAGVA) and by the “Agencia Estatal de Investigacion” (Project Grants PID2020-112967GB-C33 and TED2021-130347B-I00).

REFERENCES

de Celis, R., and L. Cadarso. 2023. “Neural Network-Based Controller For Terminal Guidance Applied In Short-Range Rockets”. *IEEE Aerospace and Electronic Systems Magazine* 1:1–11.

de Celis, R., P. S. López, and L. Cadarso. 2021. “Sensor Hybridization Using Neural Networks for Rocket Terminal Guidance”. *Aerospace Science and Technology* 111:106527.

Geng, J., and Y. Bock. 2013. “Triple-frequency GPS Precise Point Positioning with Rapid Ambiguity Resolution”. *Journal of Geodesy* 87:449–460.

Henkel, P., and C. Günther. 2012. “Reliable Integer Ambiguity Resolution: Multi-frequency Code Carrier Linear Combinations and Statistical a priori Knowledge of Attitude”. *Navigation* 59(1):61–75.

Hochreiter, S., and J. Schmidhuber. 1997. “Long Short-term Memory”. *Neural Computation* 9(8):1735–1780.

Hu, G., B. Gao, Y. Zhong, and C. Gu. 2020. “Unscented Kalman Filter with Process Noise Covariance Estimation for Vehicular INS/GPS Integration System”. *Information Fusion* 64:194–204.

Joosten, P., P. J. Teunissen, and N. Jonkman. 1999. “GNSS Three Carrier Phase Ambiguity Resolution Using the LAMBDA-method”. In *Proceedings GNSS*, Volume 99, 5–8.

Li, B. 2018. “Review of Triple-frequency GNSS: Ambiguity Resolution, Benefits and Challenges”. *The Journal of Global Positioning Systems* 16:1–11.

Nie, Z., F. Liu, and Y. Gao. 2020. “Real-time Precise Point Positioning with a Low-cost Dual-frequency GNSS Device”. *GPS Solutions* 24:1–11.

Rosenblatt, F. 1958. “The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain”. *Psychological Review* 65(6):386.

Sola, J., and J. Sevilla. 1997. “Importance of Input Data Normalization for the Application of Neural Networks to Complex Industrial Problems”. *IEEE Transactions on Nuclear Science* 44(3):1464–1468.

Teunissen, P. 2006. “The LAMBDA Method for the GNSS Compass”. *Artificial Satellites* 41(3):89–103.

Teunissen, P., P. D. Jonge, and C. Tiberius. 1997. “The Least-squares Ambiguity Decorrelation Adjustment: its Performance on Short GPS Baselines and Short Observation Spans”. *Journal of Geodesy* 71:589–602.

Wang, S., Z. You, and X. Sun. 2022. “A Partial Carrier Phase Integer Ambiguity Fixing Algorithm for Combinatorial Optimization between Network RTK Reference Stations”. *Sensors* 22(1):165.

Xu, W., W. Shen, C. Cai, L. Li, L. Wang, A. Ning, and Z. Shen. 2022. “Comparison and Evaluation of Carrier Phase PPP and Single Difference Time Transfer with Multi-GNSS Ambiguity Resolution”. *GPS Solutions* 26(2):58.

AUTHOR BIOGRAPHIES

RAUL DE CELIS is an associate professor in aerospace area at Rey Juan Carlos University. He received his Ph.D. degree from Universidad Rey Juan Carlos in December 2017. His research interests are model development of aeronautic systems and navigation and control for aerial platforms. His email address is raul.decelis@urjc.es.

JOSE GONZALEZ BARROSO is a researcher in aerospace area at Rey Juan Carlos University. He received a his MSc. degree in Computer Engineering from the UC3M, Spain. His research interests revolve around numerical methods to solve complex non-linear problems and machine learning applied to aerospace engineering. His email address is jose.barroso@urjc.es.

PABLO SOLANO LÓPEZ is an assistant professor in aerospace area at Rey Juan Carlos University. He received a Double MSc. degree in Aerospace Engineering from the Universidad Politécnica de Madrid, Spain and from the Institut Supérieur de l’Aéronautique et de l’Espace, France. His research interests revolve around numerical methods to solve complex non-linear problems and machine learning. His email address is pablo.solano@urjc.es.

LUIS CADARSO is an associate professor in aerospace area at Rey Juan Carlos University. He received the Ph.D. degree in Aerospace Engineering from the Technical University of Madrid, Spain. His research interests include operations research, navigation, and control for aerial platforms. His email address is luis.cadarso@urjc.es.