75th International Astronautical Congress (IAC), Milan, Italy, 14-18 October 2024.

IAC-24,C1,IP,15,x83073

Orbit Determination for X-ray Pulsar Navigation with the Aid of Artificial Neural Networks

Sui Chen[a,*] **and Francesco Topputo**[a]

[a,*]*Politecnico di Milano, Via La Masa 34, 20156 Milan, Italy,* sui.chen@polimi.it, fancesco.topputo@polimi.it
[*]Corresponding author

Abstract

Current space missions mainly rely on Earth-based Guidance, Navigation, and Control (GNC) systems, which involve human-in-the-loop operations. While reliable, these ground-based methods suffer from communication delays, limiting real-time responsiveness and autonomy. The increasing demand for space missions is also expected to overwhelm ground communication slots, posing challenges to future exploration. This paper introduces a novel artificial neural network (ANN)-aided X-ray pulsar autonomous navigation system. A radial basis function neural network (RBFNN) is used to predict the spacecraft's state error under high-fidelity perturbations, which is incorporated into a navigation filter's prediction step. A nonlinear Gaussian function is utilised in the network's hidden layer, with the K-means algorithm for hidden layer training and the recursive least squares (RLS) algorithm for output layer training. The navigation system relies solely on X-ray pulsar signals from the Crab pulsar for measurements. Training results show that the RBFNN is able to capture the general trends in the output's behaviours with a reasonable degree of effectiveness. Filter simulation results show position estimation errors below 3 km. By embedding the neural network directly in the filter's prediction step, our approach eliminates the need to define explicitly perturbation terms during network training and reduces computational burden of integrating highly nonlinear perturbed dynamic equations in the filter. This system also demonstrates a viable path towards fully autonomous spacecraft navigation using only X-ray pulsar signals, marking a significant step towards future space mission autonomy.

Keywords: X-ray Pulsar Navigation, Orbit Determination, Radial Basis Function Neural Network Aided Filter

1. Introduction

The rapid advancement in technology and the expanding commercial landscape within the space sector have driven a significant surge in space exploration missions. Currently, these missions depend heavily on ground-based Guidance, Navigation and Control (GNC) systems, often requiring human intervention. While this method is generally reliable, it has its drawbacks. Issues such as communication delays, limited real-time responsiveness, and the potential for ground infrastructure overload due to the rising number of space users highlight the need for innovative solutions. To address these challenges, there is a growing focus on developing onboard autonomous navigation techniques. This shift aims to reduce reliance on ground-based operations and offers a promising solution for future space missions.

One attractive autonomous navigation technique is the X-ray pulsar navigation (XNAV) which utilises X-ray pulsar timing signals as measurement sources. Pulsars are fast-rotating neutron stars that emit stable, periodic and predictable signals characterised by unique shapes. These features give pulsar-based navigation an extremely high achievable accuracy. The concept of leveraging these celestial sources as a navigation tool traces back to the 1970s, when Downs [1] explored the potential of utilising pulsating radio sources for interplanetary navigation. Across the electromagnetic spectrum, X-ray pulsars are of particular interest and often preferred for navigation applications because their higher signal-to-noise ratio allows for the use of smaller and lighter detectors compared to optical or radio sources [2]. The theoretical framework of pulsar-based navigation has been explored by several researchers [3] [4] [5] [2], and most of them demonstrated the XNAV system positioning accuracy below 10 km based on computational simulation. In 2016, XPNAV-1 [6] was launched to conduct a ground pulsar-based navigation experiment, utilising pulsar observation data collected onboard. This experiment resulted in an average navigation accuracy of 38.4 km solely through observations of the Crab pulsar. In 2017, NASA demonstrated a fully autonomous, real-time pulsar navigation experiment in space for the first time, attaining an average accuracy of 16 km [7].

For spacecraft navigation problems, a navigation filter is usually used for the spacecraft state estimation. The filter typically consists of a time-update process to propagate the state, followed by a measurement-update process to correct the propagated state. In most cases,

the time-update step consumes majority of the computational resources in a filter, primarily because numerical integration is typically used to propagate orbital state equations which are highly nonlinear with perturbation terms. Scientists and engineers have been searching for robust methods to reduce the computational burden while maintaining system accuracy. Among the various methods, the use of machine learning has proven to be an effective technique in supporting spacecraft navigation applications.

Recent studies have increasingly focused on using artificial neural networks for orbit determination. Specifically, these networks are employed to learn complex nonlinear dynamic models with high uncertainty, allowing for accurate predictions with much lower computational effort compared to other orbit propagation methods. For instance, Pesce et al. (2019) [8] utilised a Radial Basis Function Neural Network (RBFNN) to estimate disturbances in the prediction step of an adaptive extended Kalman filter. Mortlock and Kassas (2021) [9] introduced a Time Delay Neural Network (TDNN) for LEO satellite orbit determination, demonstrating improved tracking performance compared to an extended Kalman filter-based approach. Zhou et al. (2023) presented an adaptive estimation algorithm for orbit determination in their recent paper [10]. The algorithm is designed to maintain estimation accuracy while reducing computational cost by utilising a deep neural network (DNN)-based nonlinearity detector alongside an adaptive order-switching procedure.

Most existing studies have focused on using artificial neural networks to learn and predict environment perturbation accelerations in orbital dynamic equations. However, because these perturbations are often inherently unknown, the empirical expressions found in the literature only provide an approximate representation of their behaviour. Moreover, obtaining true perturbation accelerations directly is challenging because such data typically come from real missions, with some not being publicly accessible. This makes it challenging to create accurate output datasets, in the form of spacecraft accelerations, for training the neural networks. Additionally, while numerical perturbation methods involving integration offer greater accuracy, they are computationally intensive to be used within onboard spacecraft filters. Therefore, this study will instead adopt an alternative approach by implementing the neural network directly in the state prediction step of the filter, where it predicts spacecraft state errors at the next time step under the influence of high-fidelity perturbations. This approach eliminates the need to explicitly define perturbation acceleration terms during network training process and avoids the complexity of integrating perturbed highly nonlinear dynamic equations within the filter.

The rest of the paper is organised as follows. Section 2. outlines the methodology, covering spacecraft dynamics, neural network data preparation and training, the concept of X-ray pulsar navigation, and the navigation filter used for orbit determination. Section 3. presents the results of the network training performance and the overall orbit determination performance, followed by some discussion. Finally, Section 4.concludes the paper and provides suggestions for future work.

2. Methodology

2.1 Dynamics

In this paper, we consider an Earth-orbiting satellite with key properties summarised in Table 1. The spacecraft dynamics is described using Equation 1.

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t)) + \mathbf{w}(t) \tag{1}$$

where $\mathbf{x}(t)$ is the process state consisting of the spacecraft position and velocity $\mathbf{x}(t) = [\mathbf{r}(t), \mathbf{v}(t)]^T$, $\mathbf{w}$ is the process noise, and the dynamics $\mathbf{f}(t, \mathbf{x}(t))$ is defined as:

$$\mathbf{f}(t, \mathbf{x}(t)) = \begin{bmatrix} \mathbf{v}(t) \\ \mu_{\text{Earth}} \frac{\mathbf{r}}{r^3} + \mathbf{a}_{\text{pert}} \end{bmatrix} \tag{2}$$

The term $\mathbf{a}_{\text{pert}}$ is a summation of all relevant perturbation accelerations acting on the spacecraft. For an Earth-orbiting satellite, the main perturbations are the Earth J2 effect, atmospheric drag, solar radiation pressure etc.

Table 1: Key properties of the spacecraft.

Mass (kg)	100
Cross-sectional area (m^2)	5
X-ray detector area (m^2)	0.1
Drag coefficient C_D	2.2
Reflectivity coefficient C_R	1.3
Operational altitude (km)	600

2.2 Artificial Neural Network (ANN)

As mentioned in the introduction, we will employ an artificial neural network (ANN) to directly predict the spacecraft state $\mathbf{x}$ under the effect of perturbations, rather than explicitly predict the perturbation accelerations $\mathbf{a}_{\text{pert}}$ inside the dynamic equation $\mathbf{f}(t, \mathbf{x}(t))$. Following the work of [8] and [11], a Radial Basis Function Neural Network (RBFNN) is adopted in this paper, given its simple structure and fast training process.

A RBFNN is structured with three layers: an input layer, a hidden layer, and an output layer. The input layer contains source nodes which, in our case, are spacecraft state vectors $\mathbf{x}^+_{k-1}$ at time $k-1$. The hidden layer performs

a nonlinear transformation of the input via a radial basis function. The output layer linearly combines these hidden neurons to produce the final output, which is the spacecraft state error $\delta\mathbf{x}^-_{k_{\text{ANN}}}$ between the perturbed state and the unperturbed Keplerian state at time k. Figure 1 presents the overall structure of the proposed RBFNN.

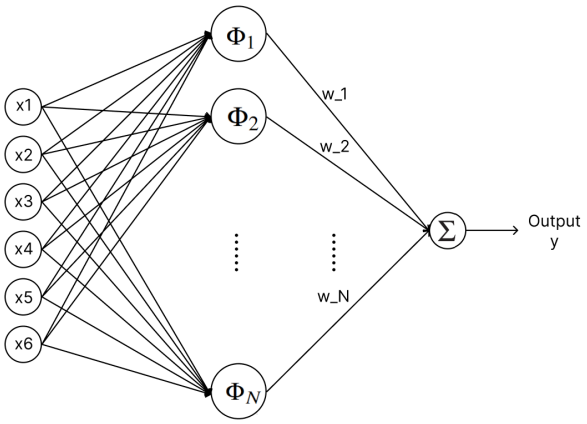


Fig. 1: Overall structure of the proposed RBFNN, with the input layer representing the spacecraft's six states at time $k-1$, and the output corresponding to the predicted state error at time k.

2.2.1 Dataset Generation

Before training the network, a substantial dataset needs to be generated which captures the input-output relationships. The input data consists of the spacecraft state $\mathbf{x}^+_i$ at the current time t_i, while the output data represents the predicted state error $\delta\mathbf{x}^-_{i+1_{\text{ANN}}}$, which is the difference between the high-fidelity "truth" orbital state and the Keplarian state at the next epoch t_{i+1}.

During the data collection process, each state is propagated for 3 hours using a high-fidelity orbit propagator to simulate the "true" dynamics. The parameters of this "truth" dynamic model, along with the initial orbital conditions, are detailed in Tables 2 and 3 respectively. It is important to note that various atmospheric models are available in the literature. Bruinsma et al. [12] have indicated that NRLMSISE-00 and DTM2009 are the most accurate models for altitudes above 500 km. Given that the operational altitude of the spacecraft in our study is set at 600 km, NRLMSISE-00 is therefore used to simulate the "true" dynamics as accurately as possible. At each time t_i, the propagated state $\mathbf{x}^+_i$ and the corresponding predicted state error $\delta\mathbf{x}^-_{i+1_{\text{ANN}}}$ are collected as a data point $(\mathbf{x}^+_i, \delta\mathbf{x}^-_{i+1_{\text{ANN}}})$. With a propagation time step of $\Delta t = 4$ s, a total of 2700 data points are generated. This entire dataset is used to train the RBFNN model to approximate $\delta\mathbf{x}^-_{i+1_{\text{ANN}}}$ based on $\mathbf{x}^+_i$. Since the state errors consist of six components, six separate RBFNN models will be trained, with each network producing the output for one component.

Table 2: Parameters used within the "truth" dynamic model.

Parameters	"Truth" dynamic model
Harmonic gravity field	50×50
Atmospheric model	NRLMSISE-00
Solar activity	Variable F10.7 index
Solar radiation pressure	Conical shadow model
Third-body perturbation	Sun + Moon + other solar planets

Table 3: Initial conditions of the spacecraft state.

x_0 (km)	3520.41	v_{x0} (km/s)	0.351373
y_0 (km)	5938.51	v_{y0} (km/s)	-1.466164
z_0 (km)	1007.11	v_{z0} (km/s)	7.406607

2.2.2 Network Training Algorithm

Following the work of [13], the K-means algorithm will be used to train the hidden layer, while the recursive least square (RLS) algorithm will be employed for training the output layer. In this section, we will provide a brief overview of the key algorithms. For more comprehensive details, readers are referred to [13].

Within the hidden layer, each neuron is mathematically characterised by a radial-basis function centered on a point. Various types of radial-basis functions are commonly applied in space applications, such as Gaussian, linear, thin-plate spline, and logistic function [14]. In this study, a nonlinear Gaussian function is selected as the radial-basis function for the hidden layer. The nonlinear Gaussian function is defined as:

$$\Phi_i(\mathbf{x}) = \exp - \frac{(\|\mathbf{x} - \boldsymbol{\mu}_i\|)^2}{2\sigma^2} \tag{3}$$

The cluster mean $\boldsymbol{\mu}_i$ among the input vectors defines the centre of the Gaussian function associated with each hidden neuron. We use the MATLAB built-in function `kmeans` to calculate $\boldsymbol{\mu}_i$. In K-means clustering, the squared Euclidean norm is used to define the measure of similarity d between every pair of vectors $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_{i'}$, i.e., the distance between centres:

$$d = \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_{i'}\|^2 \tag{4}$$

For simplicity, a constant width σ is assigned to all Gaussian functions, which is derived from the maximum distance, or the spread of the centres, $d_{\max}$ among all $\boldsymbol{\mu}$. The value of σ is computed using Equation 5:

$$\sigma = \frac{d_{\max}}{\sqrt{2K}} \tag{5}$$

where K is the number of radial centres used in the hidden layer and it is a user-defined parameter.

Upon completion of training the hidden layer, the training of the output layer can begin, which focuses on selecting the weight vector $\mathbf{w}$. To compute the weight vector in the output layer, a recursive least-square (RLS) algorithm is used. For discrete time steps $n = 1, 2, 3...$, we first define the prior estimation error $\alpha(n)$, based on the previous estimate $\mathbf{w}(n-1)$ of the weight vector:

$$\alpha(n) = d(n) - \mathbf{w}^T(n-1)\mathbf{\Phi}(n) \tag{6}$$

where the vector $\mathbf{\Phi}$ comprises the Gaussian functions obtained from Equation 3 for all hidden neurons.

The old weight estimate is subsequently updated to a new value using:

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{P}(n)\mathbf{\Phi}(n)\alpha(n) \tag{7}$$

The term $\mathbf{P}(n)\mathbf{\Phi}(n)$ is called the *gain vector* of the RLS algorithm. The matrix $\mathbf{P}(n)$ is computed recursively from:

$$\mathbf{P}(n) = \mathbf{P}(n-1) - \frac{\mathbf{P}(n-1)\mathbf{\Phi}(n)\mathbf{\Phi}^T(n)\mathbf{P}(n-1)}{1 + \mathbf{\Phi}^T(n)\mathbf{P}(n-1)\mathbf{\Phi}(n)} \tag{8}$$

The initial values of $\mathbf{w}$ amd $\mathbf{P}$ are set to be:

$$\mathbf{w}(0) = 0 \quad \text{and} \quad \mathbf{P}(0) = \lambda^{-1}\mathbf{I} \tag{9}$$

The entire training process for the RBFNN involves determining the parameters $\boldsymbol{\mu}_i$, σ, and $\mathbf{w}$. Once the network training is complete, the trained RBFNN is capable of predicting the output according to Equation 10:

$$\hat{y}_{\text{RBFNN}}(\mathbf{x}) = \mathbf{w}^T\mathbf{\Phi}(\mathbf{x}) \tag{10}$$

2.3 X-ray Pulsar Measurements

In this work, the Crab pulsar is chosen as the observational source. The signal from the Crab pulsar is detected as pulse photons collected by the X-ray detector onboard the spacecraft. The fundamental measurement of the XNAV system is the photon time of arrivals (TOAs). The complete signal processing pipeline is outlined as follows.

First, the photon TOA timestamps are collected at the spacecraft. These TOAs are folded into one pulse period using the epoch folding technique to create an observed pulse profile. A standard pulse profile is constructed at an inertial space point, usually the solar system barycentre (SSB), based on the pulsar timing model, the pulsar ephemeris and the pulse shape function. This standard pulse can be transferred from the SSB to a prior-estimated spacecraft position. Following that, a nonlinear least square (NLS) algorithm is used to compare the phase shift between the observed profile at the true spacecraft position and the standard pulse profile at the estimated spacecraft position, and any phase shift $\Delta\phi$ will correspond to the distance shift between the true and the estimated spacecraft position projected along the line of sight of the pulsar. The resultant d_{updated} is then fed into a navigation filter to aid spacecraft state estimation and correction.

$$y = d_{\text{updated}} = \hat{\mathbf{n}} \cdot \mathbf{r}_{\text{est}} + cP\Delta\phi \tag{11}$$

Equation 12 describes the time transfer between the SSB and the estimated spacecraft position, accounting for some higher-order time delay terms as outlined in [3]:

$$\begin{aligned} t_{\text{SSB}} - t_{\text{SC}} = {} & \frac{\hat{\mathbf{n}} \cdot \mathbf{r}}{c} + \\ & \frac{1}{2cD_0}\left[(\hat{\mathbf{n}} \cdot \mathbf{r})^2 - r^2 + 2(\hat{\mathbf{n}} \cdot \mathbf{b})(\hat{\mathbf{n}} \cdot \mathbf{r}) - 2(\mathbf{b} \cdot \mathbf{r})\right] + \\ & \frac{2\mu_{\text{Sun}}}{c^3}\ln\left|\frac{\hat{\mathbf{n}} \cdot \mathbf{r} + r}{\hat{\mathbf{n}} \cdot \mathbf{b} + b} + 1\right| \end{aligned} \tag{12}$$

where $\hat{\mathbf{n}}$ is the unit direction vector from the SSB to the pulsar, $\mathbf{r}$ is the spacecraft position vector from the SSB, c is the speed of light, D_0 is the distance between the pulsar and the SSB, $\mathbf{b}$ is the vector from the Sun centre to the SSB, and μ_{Sun} is the gravitational constant of the Sun. The first term on the right-hand side of Equation 12 is the first-order *Doppler delay*, representing the direct geometric time delay between the spacecraft and the SSB. The second term arises from parallax effects. Collectively, the first two terms are called the *Roemer delay*. The third term is the Sun's *Shapiro delay*, accounting for the additional time delay resulting from curved path of light influenced by the gravitational field of the Sun. If only the first term is considered while discarding the remaining higher order terms, Equation 12 is reduced to the simple geometrical representation of the projection distance between the spacecraft and the SSB along the pulsar direction, as illustrated in Figure 2.

As described earlier in this section, the final outcome of processing the X-ray pulse signal is the updated projection range between the spacecraft and the SSB along the pulsar direction. Therefore, the measurement model can be defined as follows:

$$h(\mathbf{x}) = \hat{\mathbf{n}} \cdot \mathbf{r} = \hat{n}_x x + \hat{n}_y y + \hat{n}_z z \tag{13}$$

The measurement noise is computed from the signal-to-noise ratio (SNR) of a specific pulsar:

$$SNR = \frac{F_x A_x p_f T_{\text{obs}}}{\sqrt{\left[B_x + F_x(1-p_f)\right]\left(A_x T_{\text{obs}} \frac{W}{P}\right) + F_x A_x p_f T_{\text{obs}}}} \tag{14}$$

$$\sigma_{\text{TOA}} = \frac{1}{2}\frac{W}{SNR} \tag{15}$$

In the above equations, W is the pulse width, F_x is the observed X-ray photon flux, A_x is the detector area, p_f is

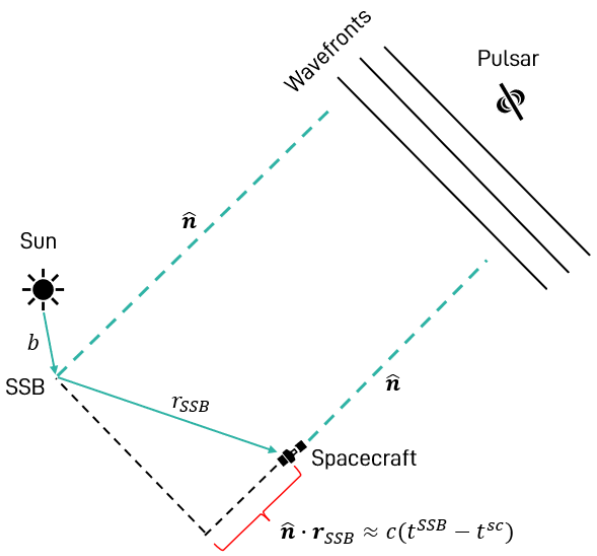


Fig. 2: Geometrical representation of the XNAV scheme.

the pulsed fraction, T_{obs} is the total observation time, B_x is the X-ray background radiation flux, and P is the pulse period.

2.4 Navigation Filter

An extended Kalman filter (EKF) is implemented for orbit determination. It consists of a time update step to propagate the spacecraft state, followed by a measurement update step which incorporates pulsar measurements to correct the state estimate. The spacecraft dynamics is modelled as a continuous-time system, while the measurements occur in discrete time. Consequently, the process and measurement equations are defined as follows:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t)) + \mathbf{w}(t) \tag{16}$$

$$y_k = h_k(x_k) + v_k \tag{17}$$

The filter is initialised by setting:

$$\mathbf{x}_0^+ = E[\mathbf{x}_0] \tag{18}$$

$$\mathbf{P}_0^+ = E\left[(\mathbf{x}_0 - \mathbf{x}_0^+)(\mathbf{x}_0 - \mathbf{x}_0^+)^T\right] \tag{19}$$

The EKF prediction block (time update) is detailed as follows:

$$\mathbf{x}_k^- = \mathbf{x}_{k-1}^+ + \int_{t_{k-1}}^{t_k} \mathbf{f}(t, \mathbf{x}(t))\, dt \tag{20}$$

$$\mathbf{P}_k^- = \mathbf{P}_{k-1}^+ + \int_{t_{k-1}}^{t_k} \mathbf{F}\mathbf{P} + \mathbf{P}\mathbf{F}^T + \mathbf{Q}\, dt \tag{21}$$

At time t_k, once the measurement y_k is obtained, it is incorporated in the EKF correction block (measurement update) to update both the state and covariance estimates:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} \tag{22}$$

$$\mathbf{x}_k^+ = \mathbf{x}_k^- + \mathbf{K}_k (y_k - h_k) \tag{23}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \tag{24}$$

In the above expressions, the Jacobian matrices $\mathbf{F}$ and $\mathbf{H}$ are defined as $\mathbf{F}_k = \left.\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_{k-1}^+}$ and $\mathbf{H}_k = \left.\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_k^-}$ respectively. $\mathbf{Q}$ is the process noise covariance matrix and $\mathbf{R}$ is the measurement noise covariance matrix.

The ANN proposed in Section 2.2 is integrated into Equation 20 during the prediction step. With the aid of ANN, the predicted state $\mathbf{x}_k^-$ is computed in two stages. First, the state is propagated assuming only two-body Keplerian motion ($\mathbf{x}_{k_{\text{Keplerian}}}^-$). By solving the perturbation-free Keplerian equation of motion, computational efficiency is improved as higher-order nonlinear perturbation accelerations are neglected. Next, the ANN-predicted state error ($\delta\mathbf{x}_{k_{\text{ANN}}}^-$), representing the difference between the Keplerian state and the fully perturbed state, is directly added to $\mathbf{x}_{k_{\text{Keplerian}}}^-$:

$$\mathbf{x}_k^- = \mathbf{x}_{k_{\text{Keplerian}}}^- + \delta\mathbf{x}_{k_{\text{ANN}}}^- \tag{25}$$

After signal processing, as outlined in Section 2.3, the X-ray pulsar measurements y_k are incorporated into the filter correction step, as described in Equation 23.

3. Results and Discussion

3.1 Simulation Setup

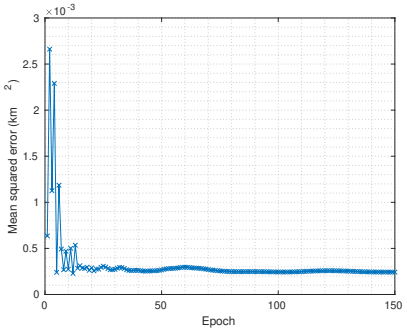
To evaluate the effectiveness of the proposed ANN and the overall performance of the navigation system, simulations were conducted for an extended duration of 6 hours using a spacecraft with the same characteristics detailed in Table 1. The subsequent sections present the simulation results along with discussion of the findings.

3.2 ANN Training Performance

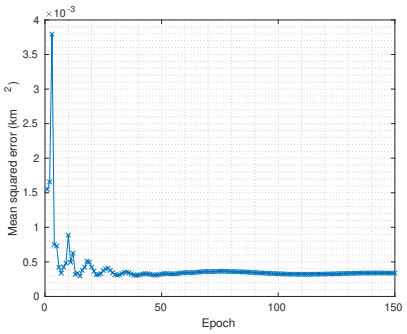
As the state errors consist of six components (position and velocity), six individual RBFNNs are trained, with each network producing the output for one component. During the training process, a loss function is typically used to evaluate the neural network's performance in predicting the output behaviour. In this work, the loss function we use during the training process is the mean squared error (MSE) between the RBFNN-predicted output and the expected output:

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \tag{26}$$

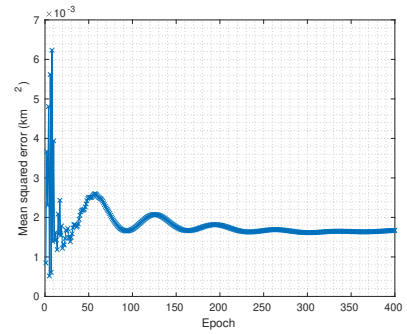
where y is the desired output, $\hat{y}$ is the actual output obtained from the RBFNN, and m is the total number of training samples. Figures 3 and 4 show the evolution of the loss function as a function of the training epoch.



(a) x component

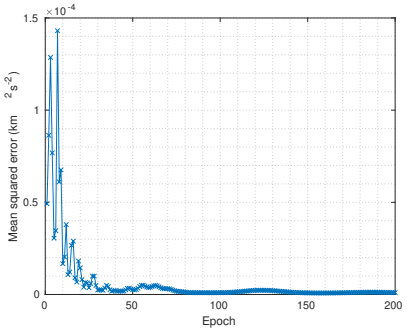


(b) y component

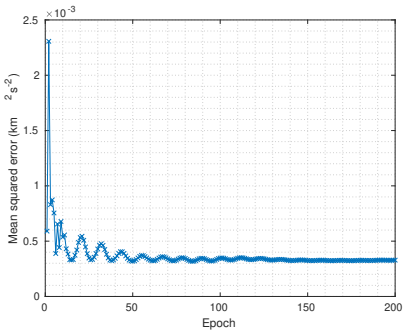


(c) z component

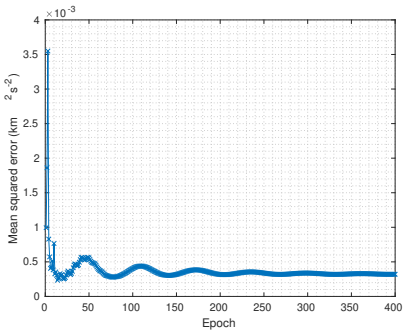
Fig. 3: RBFNN loss function evolution during the training process (position).



(a) v_x component



(b) v_y component



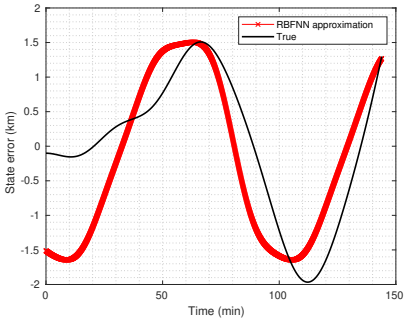
(c) v_z component

Fig. 4: RBFNN loss function evolution during the training process (velocity).

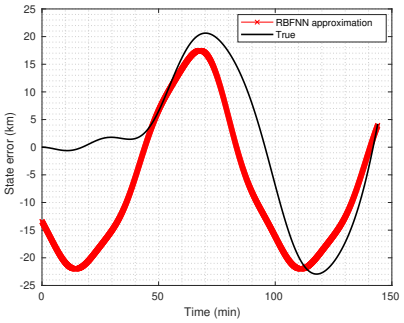
As shown in Figures 3 and 4, the loss function stabilises after approximately 50 epochs for both the x- and y-components, while the z-component requires around 200 epochs to reach stability. Once the loss function has stabilised, it indicates that the network is well-trained, and the resulting parameters $\boldsymbol{\mu}_i$, σ, and $\mathbf{w}$ can be used to predict the output.

To demonstrate the effectiveness of the trained RBFNN in predicting the output behaviour, a testing dataset is used, consisting of spacecraft states as inputs and state errors between the Keplerian state and the fully perturbed state as the expected outputs. The RBFNN-predicted outputs are generated using the trained network's parameters $\boldsymbol{\mu}_i$, σ, and $\mathbf{w}$. Figures 5 and 6 present a comparison between the RBFNN-predicted outputs and the expected outputs across all six components.

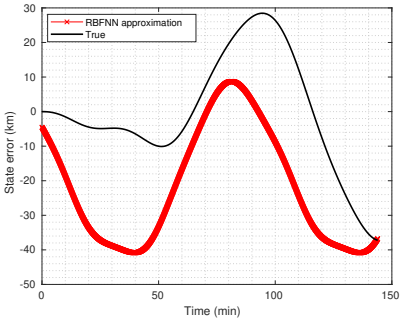
As illustrated in Figures 5 and 6, although the predicted outputs do not perfectly match the expected values, the trained RBFNN is able to capture the overall trends in the output behaviours, demonstrating a reasonable degree of effectiveness in predicting these behaviours. To improve the accuracy of the predictions, future work could explore the inclusion of additional input parameters, such as spacecraft properties, onboard clock errors, and other space environmental factors related to perturbation effects. This is because the current inputs are limited to



(a) x component

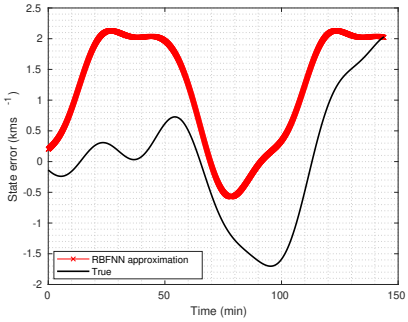


(b) y component

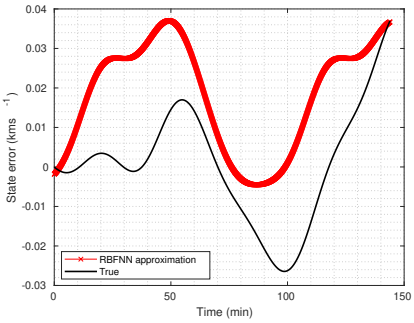


(c) z component

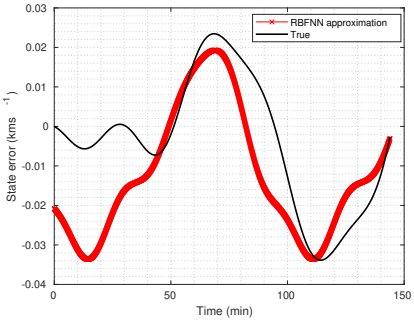
Fig. 5: Comparison of the RBFNN-predicted output and the desired output (position).



(a) v_x component



(b) v_y component



(c) v_z component

Fig. 6: Comparison of the RBFNN-predicted output and the desired output (velocity).

the spacecraft's position and velocity state vectors at the current epoch.

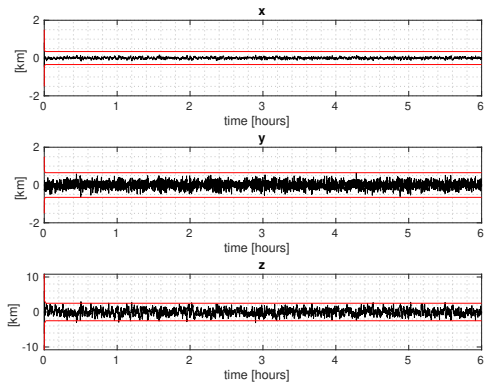
3.3 Navigation Performance

The proposed EKF, which incorporates the trained RBFNN in the prediction step and pulsar measurements in the correction step, was run for a 6-hour duration. Figures 7a and 7b show the navigation filter's performance in estimating both position and velocity. In these figures, the black line represents the error profile between the filter-estimated state and the true reference state, while the red line represents the 3σ covariance bounds. Table 4 provides a quantitative summary of the achievable accuracy of the proposed navigation system, indicating that position estimation errors can be below 3 km.

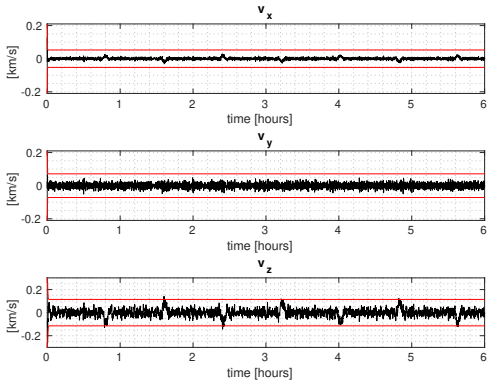
Table 4: Navigation accuracy of the proposed ANN-aided X-ray pulsar navigation system.

Position (km)		Velocity (km/s)	
δx	0.3439	δv_x	0.0527
δy	0.6524	δv_y	0.0714
δz	2.5018	δv_z	0.1145

The filter simulation results highlight the superior accuracy achieved by the proposed ANN-aided X-ray pulsar navigation system. By directly incorporating the neural network into the state prediction step of the filter, this approach removes the necessity of explicitly defining per-



(a) Position estimation errors



(b) Velocity estimation errors

Fig. 7: RBFNN-aided filter performance (6 hours).

turbation acceleration terms during the network training and bypasses the computational burden of integrating the highly nonlinear perturbed dynamic equations within the filter. Additionally, it demonstrates the feasibility of a fully autonomous spacecraft navigation system that operates solely on X-ray pulsar signals, without the support of ground stations. Given the limitations of ground-based navigation systems, such a fully autonomous approach is anticipated to emerge as a major trend in future space missions.

4. Conclusion

A novel concept of an ANN-aided X-ray pulsar navigation system is proposed in this study for an Earth-orbiting spacecraft. The system utilises a RBFNN to directly predict the spacecraft's state under high-fidelity perturbations, which is then applied in the prediction step of the navigation filter. A nonlinear Gaussian function is selected as the radial-basis function in the network's hidden layer, with the K-means algorithm used to train the hidden layer and the recursive least squares (RLS) algorithm for training the output layer. The X-ray pulsar signals from the Crab pulsar are used as the sole measurement source for the navigation system.

Training results indicate that the RBFNN is able to capture the general trends in the output behaviours, demonstrating a reasonable degree of effectiveness in predicting these behaviours. Filter simulation results highlight the superior accuracy of the navigation system, achieving position estimation errors below 3 km. By embedding the neural network directly into the filter's prediction step, our approach eliminates the need to explicitly define perturbation acceleration terms during network training and reduces the computational burden of integrating highly nonlinear dynamic equations inside the filter. Moreover, this system demonstrates the potential for a fully autonomous spacecraft navigation solution relying exclusively on X-ray pulsar signals, without ground station support - which is a promising direction for future space missions.

In future work, additional parameters, such as spacecraft properties, could be included as inputs to the neural network, allowing for greater flexibility and adaptability to different spacecraft and missions. The approach presented in this paper could also be readily applied to other space missions, including those involving planetary bodies or asteroids with highly uncertain gravitational perturbations and unknown physical influences.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 864697).

References

[1] GS Downs. Interplanetary navigation using pulsating radio sources. Technical report, 1974.

[2] Francesco Cacciatore, Víctor Gómez Ruiz, Gonzalo Taubmann, Jacinto Muñoz, Pablo Hermosín, Marcello Sciarra, Martiño Saco, Nanda Rea, Margarita Hernanz, Emilie Parent, et al. Podium: A pulsar navigation unit for science missions. *arXiv preprint arXiv:2301.08744*, 2023.

[3] Suneel I Sheikh, Darryll J Pines, Paul S Ray, Kent S Wood, Michael N Lovellette, and Michael T Wolff. Spacecraft navigation using x-ray pulsars. *Journal of Guidance, Control, and Dynamics*, 29(1):49–63, 2006.

[4] Po-Ting Chen, Jason L Speyer, David S Bayard, and Walid A Majid. Autonomous navigation us-

ing x-ray pulsars and multirate processing. *Journal of Guidance, Control, and Dynamics*, 40(9):2237–2249, 2017.

[5] Sui Chen and Francesco Topputo. Autonomous navigation using integrated x-ray pulsar and optical measurements. In *29th International Symposium on Space Flight Dynamics (ISSFD)*, pages 1–7, 2024.

[6] Liangwei Huang, Ping Shuai, Xinyuan Zhang, and Shaolong Chen. Pulsar-based navigation results: data processing of the x-ray pulsar navigation-i telescope. *Journal of Astronomical Telescopes, Instruments, and Systems*, 5(1):018003–018003, 2019.

[7] Alexandra Witze. Nasa test proves pulsars can function as a celestial gps. *Nature*, 553(7688):261–263, 2018.

[8] Vincenzo Pesce, Stefano Silvestrini, and Michèle Lavagna. Radial basis function neural network aided adaptive extended kalman filter for spacecraft relative navigation. *Aerospace Science and Technology*, 96:105527, 2020.

[9] Trier Mortlock and Zaher M Kassas. Assessing machine learning for leo satellite orbit determination in simultaneous tracking and navigation. In *2021 IEEE aerospace conference (50100)*, pages 1–8. IEEE, 2021.

[10] Xingyu Zhou, Dong Qiao, and Xiangyu Li. Adaptive order-switching kalman filter for orbit determination using deep-neural-network-based nonlinearity detection. *Journal of Spacecraft and Rockets*, 60(6):1724–1741, 2023.

[11] Yue Wu, Hui Wang, Biaobiao Zhang, and K-L Du. Using radial basis function networks for function approximation and classification. *International Scholarly Research Notices*, 2012(1):324194, 2012.

[12] Sean L Bruinsma, Noelia Sánchez-Ortiz, Estrella Olmedo, and Nuria Guijarro. Evaluation of the dtm-2009 thermosphere model for benchmarking purposes. *Journal of Space Weather and Space Climate*, 2:A04, 2012.

[13] Simon S Haykin. *Neural networks and learning machines: International Version*. Pearson Education (US), 2009.

[14] Stefano Silvestrini and Michèle Lavagna. Deep learning and artificial neural networks for spacecraft dynamics, navigation and control. *Drones*, 6(10):270, 2022.