Path-following control using spiking neural networks associative maps

Fernández, Javier Pérez; Vargas, Manuel Alcázar; Carrillo, Juan A.Cabrera; Aguilar, Juan J.Castillo; Shyrokau, Barys

DOI
10.1016/j.robot.2025.105077

Publication date
2025

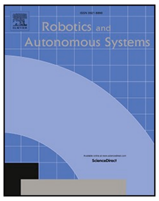
Document Version
Final published version

Published in
Robotics and Autonomous Systems

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Path-following control using spiking neural networks associative maps

Javier Pérez Fernández [a,*], Manuel Alcázar Vargas [a], Juan A.Cabrera Carrillo [a], Juan J.Castillo Aguilar [a], Barys Shyrokau [b]

[a] University of Málaga, Spain
[b] Delft University of Technology, the Netherlands

ARTICLE INFO

Keywords:
Linear quadratic regulator
Spiking neural networks
Associative maps
Automated vehicle

ABSTRACT

Bio-inspired control systems attract significant interest in the scientific community. The advantage of neural systems lies in their ability to adapt to control processes. Path-following tasks in automated vehicles and advanced driver assistance systems are an essential component related to vehicle safety and performance. It is known that model-based controllers, which integrate a vehicle model into the control logic, are more effective than geometry-based controllers. However, a disadvantage of model-based controllers is the lack of adaptation capability to changing vehicle dynamic conditions. To address this issue, an adaptive neural controller for path-following tasks is proposed based on neural networks, particularly Spiking Neural Networks and Associative Maps. Consequently, associative maps and neural interpolation via the modelling of non-linear synaptic connections are brought to a spiking neural network to perform adaptive control tasks. Neural associative maps are used to derive functional relationships between neural inputs and outputs, further enhancing inference capabilities. In addition, neural interpolation with non-linear synaptic connections enables efficient pairwise association. Thus, by reproducing a linear quadratic regulator with a learning-capable neural network, it is possible to adjust for discrepancies and changes in dynamics through spike-timing-dependent plasticity. Results demonstrate that the adaptive controller is effective in maintaining the initial tracking performance of the vehicle while adapting to changing dynamic conditions with a computational cost that allows real-time execution. The proposed strategy results in lower error levels in lateral tracking after the learning process, while providing similar performance on heading.

1. Introduction

The control of vehicle lateral dynamics is an active and constantly evolving research area, especially with the rise of automated vehicles and advanced driver assistance systems (ADAS). Efficient path-following control ensures driver safety and proper vehicle operation [1].

Various control techniques have been intensively applied to vehicle path-following control. Geometry-based controllers, such as the Stanley controller [2], have often been used in the past but have demonstrated a limited performance in a wider driving range. Instead, model-based controllers that integrate a vehicle model into the control logic have proven to be more effective. Some of the most common model-based controllers are the linear-quadratic regulator (LQR) [3], sliding mode control (SMC) and model predictive control (MPC) [4].

One of the main disadvantages of model-based controllers is the need for adaptation. Vehicle dynamics dramatically change due to weather conditions, road surface and component wear, e.g. tires. In addition, there are always discrepancies between the model and actual vehicle behaviour. To address this issue, various adaptive control approaches have been proposed. One option is to use a more complex model and update it over time to adapt to changing conditions, adding this functionality to the MPC [5] or SMC [6]. However, this requires significant computational costs. Alternatively, a less demanding control technique, such as LQR, can be used along with an adaptive algorithm to minimize discrepancies, resulting in a lower computational budget.

An emerging option for adaptive control is the use of neural networks [7]. Neural networks are able to learn from data, making them a promising approach to adapt to changing situations. Adaptive neural network applications in non-linear control include electromechanical systems [8], robotics [9], vehicles [10] or autonomous systems [11]. Spiking neural networks (SNN) are a type of neural network inspired by the structure and function of biological neural networks [12]. They have

* Corresponding author.
E-mail address: juancas@uma.es (J.P. Fernández).

https://doi.org/10.1016/j.robot.2025.105077
Received 30 September 2024; Received in revised form 9 May 2025; Accepted 22 May 2025
Available online 23 May 2025

successfully been used for non-linear control [13] including robotics [14–16], vehicles [17,18] or autonomous systems [19]. Other applications of spiking neural networks include computer vision [20,21], neuromorphic computing [22,23] and information processing [24,25].

This paper proposes an adaptive path-following algorithm for automated vehicles based on spiking neural networks and associative maps. First, an LQR controller is used as a reference [26]. Next, this controller is replicated using a neural network endowed with learning capabilities. This way, the complete system is capable of adapting to discrepancies and changes in dynamics through continuous learning. Associative maps allow the efficient integration of the LQR controller functionality. Additionally, associative maps are a learning technique used to create associations between inputs and outputs, so that inferences can be made from them.

Therefore, the LQR control function is subdivided into sub-functions, similar to an approach in genetic programming [27]. Thus, some neural sets function as lookup tables to integrate controller operations [28]. Neural structures with this functionality can be found in biological systems. In these cases, lookup table interpolation is performed through non-linear synaptic junctions [29,30]. This requires the implementation of "and-like" operations, where the connections between neurons perform some forms of multiplication [31]. The Radial Basis Function (RBF) is used to divide the input space and perform pairwise connections through axo-axonic synaptic junctions [32–34]. This study proposes using three synapse models to represent the main types of biological synapses: axo-dendritic, dendro-dendritic, and axo-axonic. Unlike conventional neural network models that rely only on axo-dendritic connections, this approach enables a more detailed and biologically accurate representation of neural dynamics.

To implement adaptation, it is necessary to define the learning process that intervenes during control. There are three basic learning paradigms [35,36]: supervised learning using error vectors [19,37], reinforcement learning using scalar reward signals [38,39] and unsupervised learning using statistics of the input signal itself.

Supervised learning was chosen over the other approaches because it only requires a known error vector, which is available in this context. On the contrary, reinforcement learning requires a scalar reward signal, which may not be available or difficult to define during control. In turn, unsupervised learning relies on the statistics of the input signal itself, which may not be sufficient to achieve the desired adaptation.

Making use of neuronal implementation based on SNN, a bio-inspired method based on the temporal difference between pre- and post-synaptic neuron spikes is used. This learning method, called spike-timing-dependent plasticity (STDP), is modulated by the error vector at each time step, similar to how dopamine modulates learning in biological systems. A more extensive description of this learning method can be found in [36].

In this paper, the proposed methodology has been applied to perform path tracking. The performance of the controller is evaluated by means of simulations in inter-urban driving environments. Key Performance Indicators (KPIs) are established to evaluate the performance of the adaptive controller, allowing for a comparison of the response with an LQR controller without adaptation. The results indicate that the proposed controller can maintain the initial tracking performance of the vehicle while adapting to changing dynamic conditions.

The main contributions of this study are as follows:

- Implementation of spiking neural networks to reproduce the behaviour of a linear quadratic regulator (LQR) and provide it with real-time adaptation capability.
- Definition of neural associative maps to derive functional relationships between neural inputs and outputs, thus enhancing inference capabilities.
- A framework inspired by genetic programming that decomposes the control problem into subfunctions, thus facilitating the development of novel controllers.
- To develop a methodology for performing neuronal interpolation with non-linear synaptic junctions. More specifically, the replication of axo-dendritic, dendro-dendritic and axo-axonic biological synapses allows efficient pairwise associations.
- Implementation of the proposed approach to the development of a path following LQR SNN controller with real-time learning capability.

One of the difficulties faced in this type of system is the need to execute the control loop in real time, which requires simplifying the complexity of the proposal to maintain this requirement without compromising the performance of the controller. The proposal presented here allows its execution in real time with a satisfactory result in terms of lateral error control and heading.

This paper is structured as follows: The neuronal network is described in Section 2. Section 3 is devoted to the learning procedure. Section 4 presents the LQR controller. A sensitivity and stability evaluation and a performance comparison are included in Section 5. The simulation setup is defined in Section 6. The results of simulations and experiments and a discussion are included in Section 7. Finally, conclusions are drawn in Section 7.

2. Spiking neural network

This section is devoted to describing the main components of the spiking neural network that will be used to develop a neural LQR controller with the learning capability to perform path tracking. Generally speaking, neural networks are formed by linking neurons through synaptic connections. Thus, by grouping different neurons, it is possible to create a structure capable of performing the desired task. Next, the three main elements that constitute the proposed spiking neural network, i.e. the neuron, the synapse and the neural structure, are described. Each subsection details the mathematical model of each component respectively.

2.1. Neuron model

The main component of a neural network is the neuron. Neurons are composed of dendrites, body, and axons (Fig. 1). Both dendrites and axons communicate with the environment [40]. The behaviour of these neural connections, called synapses, is described in Section 2.2. The primary function of the neuron's body, known as the soma, is to convert the concentration of neurotransmitters in the synapse (ρ) into electrical impulses. These impulses are produced by the current (c) generated in the dendrites and contribute to the potential of the neuron (u), which then travels along the axon as an electrical impulse.

In this paper, a type of neuron that encodes information in the form of spikes, as occurs in biological systems, is used. In particular, the Leaky Integrate-and-Fire (LIF) [41] neuron model is employed to reproduce the dynamic processes occurring inside biological neurons. This

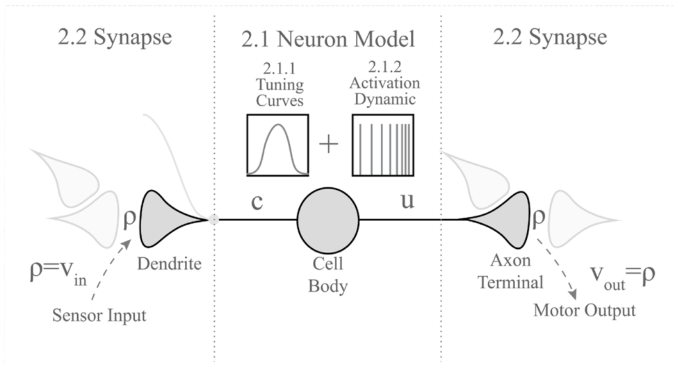


Fig. 1. Neuron model.

simplified model resorts to only one differential equation per neuron, significantly reducing its computational cost compared with more complex models. To computationally reproduce the biological process carried out in the neuron of conversion of neurotransmitter concentration into electrical impulses, two stages are implemented. First, tuning curves are used to relate the neuronal gain as a function of the spatial position of the impulses. Next, the activation dynamics model its temporal response. These processes are described next.

2.1.1. *Tuning curves (ρ-c)*

Each neuron of the network has an associated tuning curve [42] that relates the actual concentration of neurotransmitters with the current generated in the neuron body. This way, tuning curves allow for changes in the response of the neuron based on the concentration of neurotransmitters. To this end, Radial Basis Functions (RBFs) are used in this study [43]. This selection was made because they allow for the approximation of multivariable functions by adding them linearly. Consequently, an RBF tuning curve is assigned to each neuron in such a way that the linear combination of all neurons represents the process variable.

Fig. 2 shows RBF using a different selection of distribution centres (μ) and variances (σ^2). Eq. (1) represents the Gaussian tuning curve that reproduces the current for each neuron *n,* m being the total number of neurons. This current is a function of the concentration in the synapse (ρ) or an external variable (v_{in}). In the case of sensory neurons $\rho = v_{in}$. The position of each neuron in space is described by the centre (μ) of the Gaussian bell.

$$\begin{cases} c(n,\rho) = e^{\frac{-(\mu(n)-\rho)^2}{2\sigma^2}} \\ \mu(n) = \rho_{min} + \dfrac{n-1}{m-1}(\rho_{max} - \rho_{min}) \end{cases} \tag{1}$$

As can be seen in Fig. 2b, the RBF tuning functions of the neurons are uniformly distributed in the working range [ρ_{min}, ρ_{max}]. Each Gaussian bell represents the spatial position of a neuron in the input range. Therefore, depending on the neurotransmitter concentration (ρ), a certain number of neurons are activated, each of them generating the corresponding current value (*c*). This way, neural activity is distributed among a limited number of neurons, with those closer to the input level of neurotransmitter concentration responding more actively.

On the one hand, this methodology prevents very high neuronal activity, as can be seen in (Fig. 2c), which would imply high energy consumption in a biological system. On the other hand, it also avoids a very low activity which would negatively affect the input discretization, as can be observed in (Fig. 2a).

2.1.2. *Activation dynamics (c-u)*

The activation dynamics defines the firing activity of the neuron [44]. In this work, activation dynamics is modelled using Eq. (2), which reproduces the behaviour of the LIF neuronal model making use of a time constant (τ). This model converts the current c(n,ρ) obtained from the tuning curves into the neuron (u) potential over time. Additionally, when this potential reaches the threshold value (u_{th}), it resets to its bias value (u_0), which generates an electrical impulse, also called spike. The time at which this occurs is known as the time of the last spike (t_s).

$$\begin{cases} \dot{u}(n) = \tau\,(c(n,\rho) + u_0 - u(n)) \\ if\ u(n) > u_{th}\ then \begin{cases} u(n) = u_0 \\ t_s(n) = t \end{cases} \end{cases} \tag{2}$$

2.2. *Synapse model (u-ρ)*

The synapse is the component of the neuron that allows spikes to pass from an output neuron to an input neuron via neurotransmitters. Different synaptic connections can occur depending on the zone of the neuron involved, giving rise to three types of synapses: axo-dendritic, dendro-dendritic and axo-axonic [34].

The concentration (ρ) of these neurotransmitters is responsible for enhancing or decreasing the response of the post-synaptic neurons. This work proposes the use of the three types of synapse models. Each model reproduces the behaviour of the three main existing biological synapse types. On one hand, axo-dendritic and dendro-dendritic connections are considered linear. On the other hand, multiplication is assigned to axo-axonic synapses to reproduce non-linear connections. Therefore, an appropriate selection of synapse types is essential to establish the desired behavior of the neural network.

Fig. 3 shows the possible configurations between the axons and dendrites of two neurons. The most common synaptic connection is axo-dendritic. In this type of synapse, the axon of one neuron connects with the dendrites of another. This connection is usually considered linear and can include a delay in the release of neurotransmitters. Similarly, the dendro-dendritic connection between two dendrites of different neurons is usually modelled like the axo-dendritic connection. However, the axo-axonic neuron, produced between two axons, usually exhibits

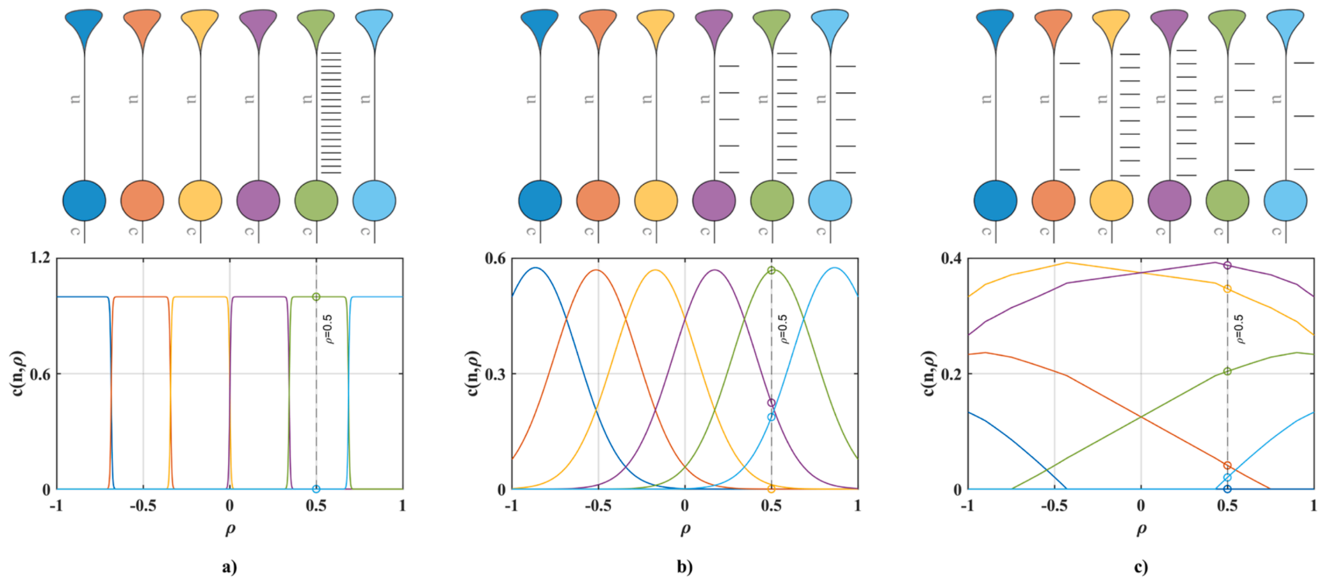


Fig. 2. Tuning curves: low a), medium b), and high c) activity.

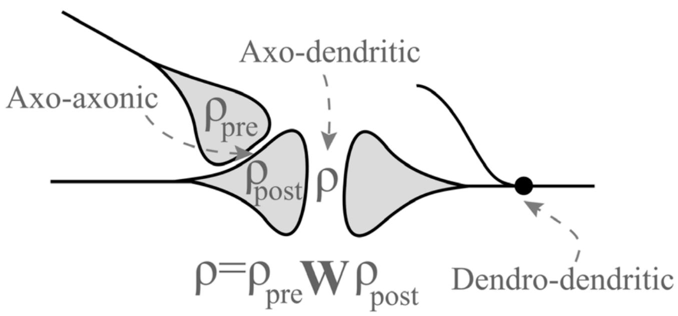


Fig. 3. Synapse configuration.

non-linear behaviour, where one of the axons directly modulates the other. This behaviour can be multiplication or a firing detector in a digital "and-like" operation.

Consequently, for linear connections (axo-dendritic and dendro-dendritic), Eq. (3) is used to convert the membrane potential into neurotransmitter concentration, which is proportional to the firing rate. To do so, the firing rate is defined as the inverse of the time elapsed since the last spike, calculated as the difference between the current time (t) and the time of the last spike (t_s). Additionally, a gain term (s_g) and bias value (s_0) have been added to compensate for residual neuron activity.

$$\rho(t_s) = \left(\frac{s_g}{(t - t_s^{pre}) - s_0}\right)\boldsymbol{W} \tag{3}$$

Where **W** represents the weight of the synaptic connection.

To reproduce the non-linear response of axo-axonic connections, it is necessary to include the neuronal activity of both axons. To this end, a multiplication operation is performed [45], according to:

$$\rho\left(t_s^{pre}, t_s^{post}\right) = \left(\frac{s_g}{(t - t_s^{pre}) - s_0}\right)\boldsymbol{W}\left(\frac{s_g}{(t - t_s^{post}) - s_0}\right) \tag{4}$$

where (post) is the post-synaptic neuron, and (pre) is the presynaptic neuron.

Motor neurons are connected to the environment; therefore, neurotransmitter concentration is assigned to an output variable ($v_{out} = \rho$). Similarly, in the case of sensory neurons, an input variable is assigned ($\rho = v_{in}$).

2.3. Neural structure

By combining neurons and synapses, it is possible to create different neural structures. In this proposal, an associative map is required to implement the LQR controller in a neural network. Poggio [29] proposed an approach to modelling "how the brain might work", using a Gaussian function as the only factorizable RBF. The Gaussian function is used to represent any function in a physiologically plausible manner by establishing relationships between pairs of variables (Fig. 4).

For this purpose, Poggio proposed that the brain is composed of 'modules' that can approximate any multivariable function. The organisation of the modules for a three-variable function is shown in Fig. 4. In this case, inputs x and y, symbolizing a receptive field, are discretized using Gaussian functions and an associative map. The output is obtained using function $G_2 = f(x,y)$. Variable z is also associated with function $G_1 = g(z)$. Finally, the function that associates the three variables by combining G_1 and G_2 through multiplication is defined by $G_3 = f(x,y) \cdot g(z)$. Poggio stated that the combination of multiple variables could be replicated by using only the multiplication operator.

In this study, Poggio's theory is applied to create associative maps between two variables and connect them to represent any desired function, such an LQR controller. As indicated in 2.2, this work proposes using axo-axonic connections to reproduce the non-linear relationship between two variables. Subsequently, more complex functions can be

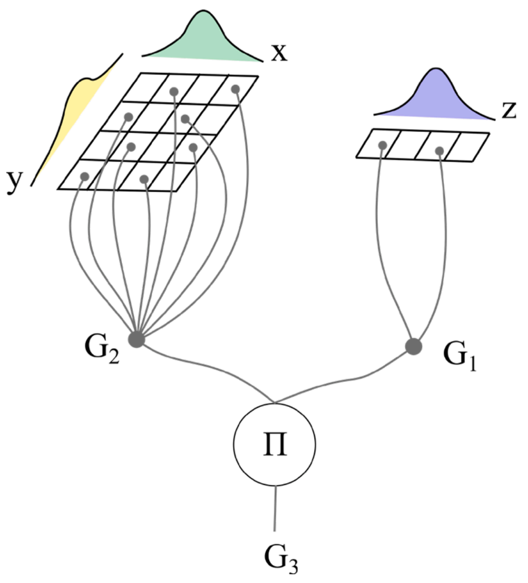


Fig. 4. Physiologically plausible multidimensional Gaussian structure (extract from [29]).

replicated by combining sets of neurons that perform the required linear and non-linear operations with the tuned weights provided by associative maps.

As shown in Fig. 5, each associative map consists of two input variables, one connected with an axo-dendritic layer and the other one assigned to an axo-axonic layer. Both layers contain neurons associated to Gaussian tuning curves uniformly distributed over the input space (see Eq. (1)), as was described in 2.1.1. The connections of these neurons through non-linear synapses determine the output value of the associative map.

For a given input, a specific group of neurons is activated according to the tuning curve associated with each neuron. This way, closer neurons are more excited, generating greater neuronal activity. Therefore, the weight of the synaptic connection with the highest activity determines the output at each moment in time (Fig. 6). This method is similar to coincidence detection [30,46], in which non-linear connections are used to select a neuronal region. Therefore, the activity in the neurons of the module or associative map depends on the synchronisation between the connected neurons (temporal summation) and the temporal distribution of the variables that define the associative map (spatial summation).

For instance, an associative map can be created between two variables (x,y) that represent function f(x,y) (Fig. 7). This map associates each variable to the axo-dendritic (x) and axo-axonic (y) unions. The weight matrix of the map (5) is directly calculated by evaluating the function at the centroid of the tuning curves (μ) for each input, as defined in Eq. (1).

$$\boldsymbol{W} = f(\boldsymbol{x}, \boldsymbol{y}) \begin{cases} x = \mu_{axo-dentritic} \\ y = \mu_{axo-axonic} \end{cases} \tag{5}$$

The function exhibits greater or lesser discretisation depending on the number of neurons (m) associated with each input set.

Fig. 7 shows an associative map that reproduces function f(x,y)=-(x^2+y^2), a colour scale is used to represent the weight of each connection between the axo-dendritic and axo-axonic layers . Sinusoidal signals are utilized as inputs for both the axo-axonic and axo-dendritic layers to evaluate the functionality of the associative map.

To sum up, a module that reproduces any non-linear function f(x,y) can be created using the proposed neural network. This module can be combined with others, creating more complex functions. This way, it is possible to emulate the behaviour of brain areas that carry out complex

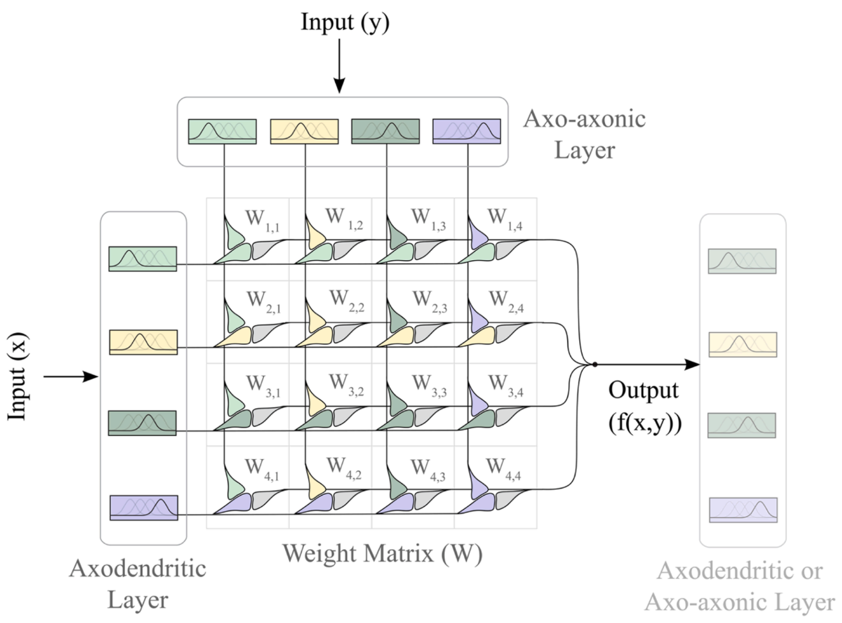


Fig. 5. Neural structure for the input-output associative map.

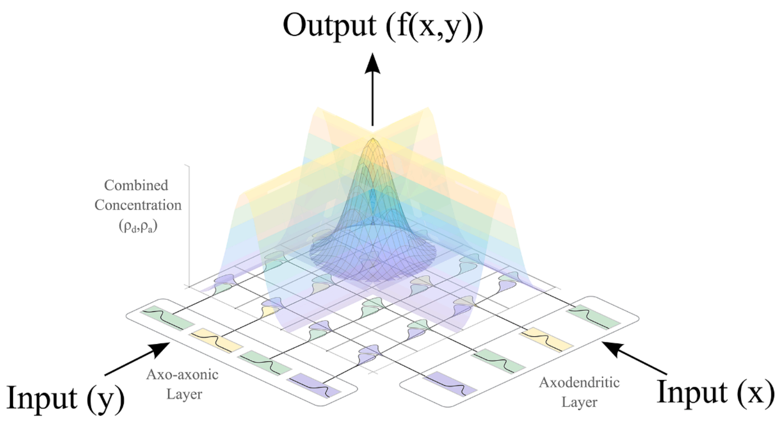


Fig. 6. Neural activity concentration in the synapses.

processing, such as system control of objects and face detection in the visual system. To facilitate adaptation or the learning of unknown behaviours, an appropriate learning method must be employed to define the network weights, as described in the following section.

3. Learning method

Learning allows the adaptation of the LQR controller to model discrepancies, changes in dynamics, and other unmodelled uncertainties. This study is aimed at performing lateral control of a four-wheeled vehicle to follow a predetermined path. In this example, the error vector (e) contains information about the lateral and heading deviations. As this error vector is determined at each time step, supervised learning is proposed as the learning methodology.

The main advantage of the proposed approach is that it can be actively implemented during vehicle control, having the ability to tune the reference controller (LQR) without compromising stability.

A biologically plausible mechanism that exploits neuronal plasticity is used to implement supervised learning with a spiking neural model. The chosen mechanism is spike-timing-dependent plasticity (STDP), which adjusts synaptic weights (W) depending on the firing times of pre- and post-synaptic neurons [47]. Additionally, this mechanism is modulated by an external signal that activates or deactivates learning, similar to dopamine in biological learning.

Therefore, the eligibility trace (c) is defined as the STDP signal. The eligibility trace value is obtained from the firing time of the pre- and post-synaptic neurons (6) as a function of time constant τ_c. This eligibility emphasizes weight potentiation to achieve synchronization between the two neurons, depending on the firing times (t_s) of the pre- and postsynaptic neurons (Fig. 8).

$$\dot{c} = \tau_c \left(\frac{1}{(t - t_s^{pre})(t - t_s^{post})} - c \right) \tag{6}$$

Additionally, the action of dopamine (d) is defined using the error vector (e), as a function of time constant τ_d, according to:

$$\dot{d} = \tau_d (e - d) \tag{7}$$

Both signals modify the synaptic weights of the associative maps through Eq. (8), which adds a learning rate factor (λ) to adjust the learning speed for each problem.

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \lambda\, \boldsymbol{c}(t)\, \boldsymbol{d}(t) \tag{8}$$

Neuronal connections have the capacity to modify their weight depending on the temporal synchronization of these, the error committed (dopamine (d)) and the spatial arrangement of the variables

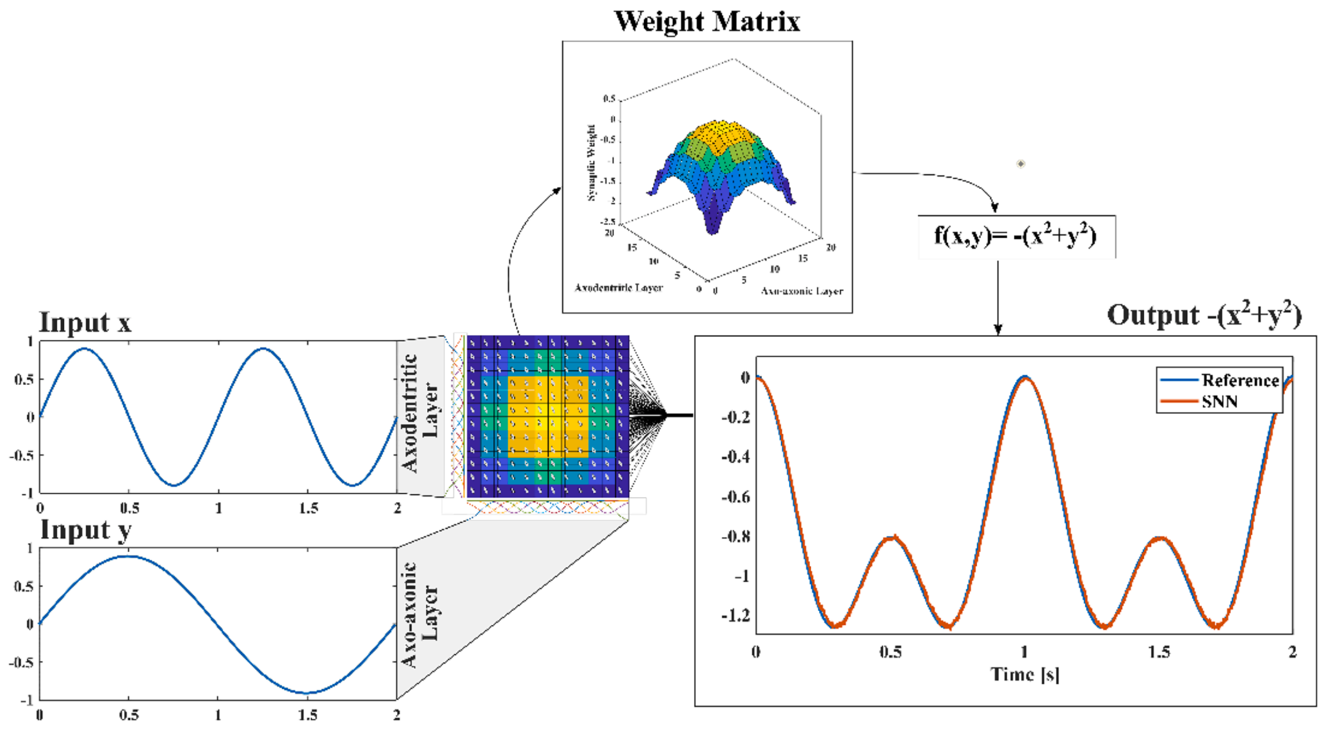


Fig. 7. 2D function f(x,y)=-(x^2+y^2) neural representation for sinusoidal input.

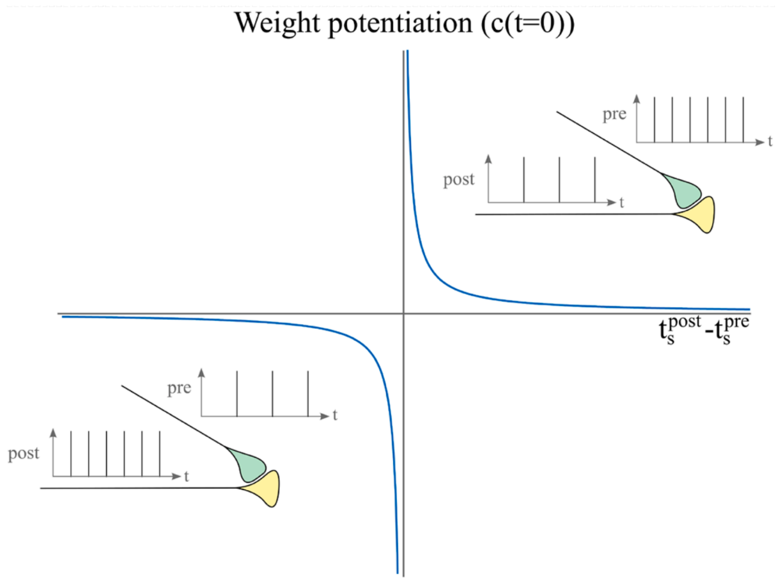


Fig. 8. Weight potentiation based on pre- and postsynaptic firing times.

that activate that part of the associative map that needs learning.

To illustrate the learning process, we present an example of a simple neural network, similar to the one shown in Fig. 7, which represents function -(x^2+y^2). In this case, the network has no prior knowledge. Therefore, all weights are initialized at zero at the beginning of the learning process ($\boldsymbol{W} = 0$).

As shown in Fig. 9, the weight matrix is updated in each iteration. In addition, Fig. 10 shows the response of the network compared to the reference value and the corresponding level of dopamine at each instant. It is worth noting that dopamine levels are directly related to the error in tracking the reference signal. This figure includes the response of the network after 5, 10 and 15 iterations. During this process, it can be verified that the neural network began to shape the weight matrix at its ends, prioritizing the area with more error and, consequently, higher dopamine assignment. After 15 iterations, the weight matrix reached the configuration shown in Fig. 7, thus demonstrating that the learning method functioned as intended. Notably, the learning rate of the network is reduced when low dopamine levels are reached.

4. Neural LQR controller

This section introduces the LQR controller, which is used as a model-based controller to implement a neural network with learning capabilities. Additionally, the controller is equipped with preview and gain-scheduling capabilities that enable real-time adjustment to improve its performance. The integration with the neural network, coupled with the

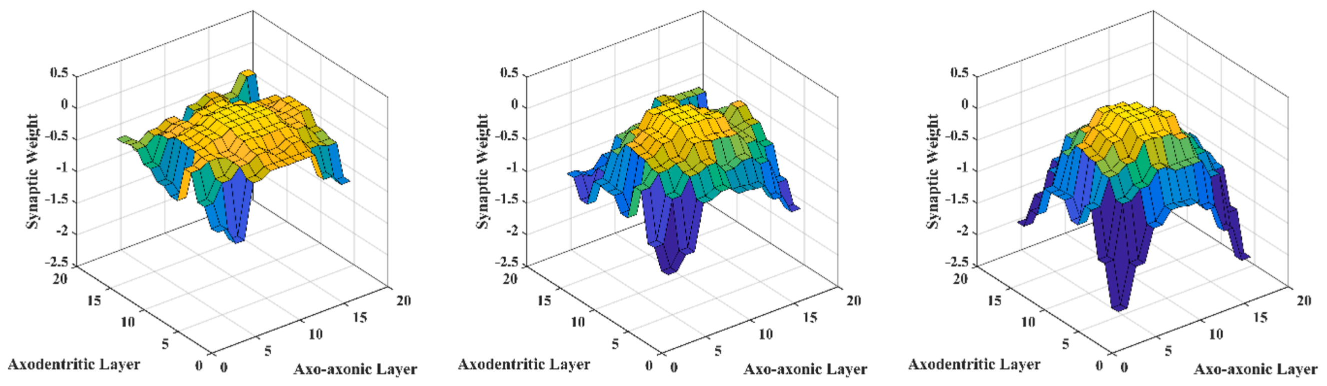


Fig. 9. Evolution of the weight matrix (W) during the learning process (5, 10, and 15 iterations).

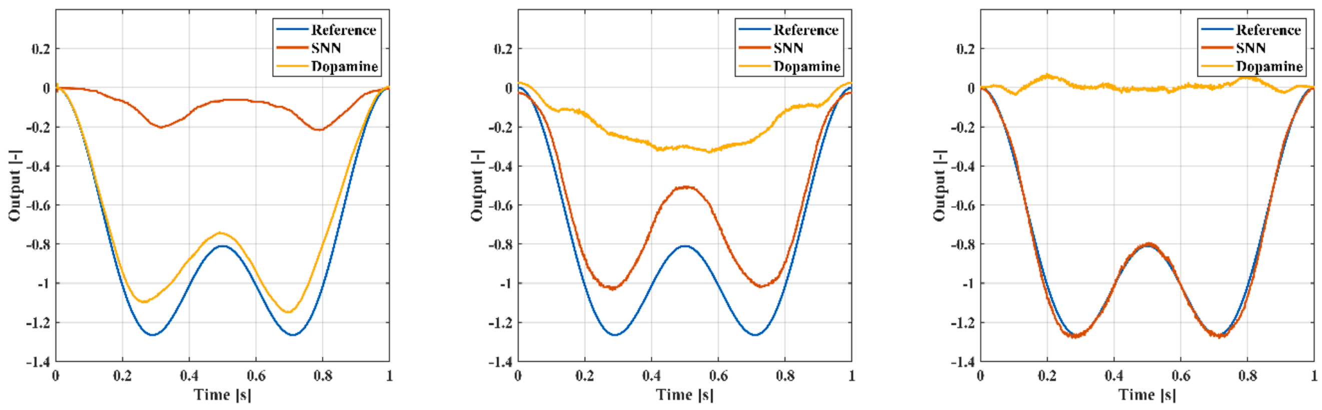


Fig. 10. Evolution of the output and dopamine during the learning process (5, 10, and 15 iterations).

use of STDP learning, allows the controller to continuously adapt to external conditions and model discrepancies.

4.1. Conventional LQR

LQR provides an alternative to selecting the closed-loop eigenvalue locations to minimize the cost function that combines the input energy and system state (x). Specifically, the LQR calculates the steering wheel angle (δ) (where δ = -K*x) to minimize the cost function (9). The controller gain is then calculated by solving the Continuous Algebraic Riccati Equation ($K = [K_y, K_{\dot{y}}, K_\Psi, K_{\dot{\Psi}}]$) (10).

$$J = \int_0^\infty \left(x^T(t) Q\, x(t) + \delta^T(t) R\, \delta(t)\right) dt \tag{9}$$

$$K = \left(R + B^T x\, B\right)^T B^T x\, A \tag{10}$$

where Q and R are the weights of the states and the control input, respectively, and must be adjusted accordingly. To determine the gains, a bicycle model was used with a lateral error (e_y) and heading error (e_Ψ) as well as their derivatives ($dote_y, dote_\Psi$) set as state variables (x) (11).

$$\boldsymbol{x} = \begin{bmatrix} e_y & \dot{e}_y & e_\psi & \dot{e}_\psi \end{bmatrix}^T \tag{11}$$

Using this state vector, state-space Eq. (12) is posed using A and B matrices of the bicycle model.

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\, \boldsymbol{x} + \boldsymbol{B}\, \delta \tag{12}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 & \frac{C_f}{m} & 0 & \frac{l_f C_f}{I_{zz}} \end{bmatrix}^T$$

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{C_f + C_r}{m\, V_x} & \frac{C_f + C_r}{m} & -\frac{C_f}{m\, V_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{C_f\, l_f - C_r l_r}{I_{zz} V_x} & \frac{C_f l_f - C_r l_r}{I_{zz}} & -\frac{C_f l_f^{\,2} + C_r l_r^{\,2}}{I_{zz} V_x} \end{bmatrix}$$

Table 1 lists the variables and parameters used in the simulations. The control function is presented in Eq. (13), where a gain vector is defined for each state variable and a global gain is associated with the feedback term (K_{FB}).

Additionally, a feedforward gain (K_{FF}) was added to utilize future information from the system using the curvature response of the bicycle model (14). The future information used is the road curvature radius (R).

$$\delta = K_{FB}\left(\left[K_y\ K_{\dot{y}}\ K_\psi\ K_{\dot{\psi}}\right]\boldsymbol{x}\right) + K_{FF}\delta_{FF} \tag{13}$$

$$\delta_{FF} = \frac{l_f + l_r + k_{us} V_x^2}{R} \tag{14}$$

Due to the strong dependence of the response of the system to the

Table 1
Icycle model parameters.

Parameter	Description	Value	Units
M	Mass of the vehicle	1620	kg
l_f	Distance from front Axle to CoG	1.075	m
l_r	Distance from rear Aaxle to CoG	1.725	m
C_f	Front axle stiffness	1.5e5	N/rad
C_r	Front axle stiffness	1.1e5	N/rad
I_{zz}	Inertia moment of vehicle (z-axis)	2253	kg*m^2
k_{us}	Understeer gradient	0.0098	rad*s^2/m

longitudinal speed (V_x), the control gain is determined for multiple discrete speed values. This way, as a way of gain scheduling, the controller parameters are adjusted according to the speed at each instant. Control function (13) is modified to include system longitudinal speed dependence, giving rise to the following equation:

$$\delta = K_{FB}\left(K_y\left(e_y, V_x\right) + K_{\dot{y}}\left(dote_y, V_x\right) + K_\psi\left(e_\psi, V_x\right) + K_{\dot{\psi}}\left(dote_\psi, V_x\right), V_x\right) + K_{FF}(R, V_x) \tag{15}$$

4.2. Neural LQR

This work presents an innovative implementation of the LQR controller using spiking neural networks. Control variables, i.e. heading and lateral error, are related through neural maps, each one consisting of two variables with a size of $m \times m$ neurons.

The use of two variables per map is possible because the control function (15) only has a non-linear relationship with the longitudinal velocity. This makes it possible to implement it by using only six associative maps, each associated with a gain ($[K_y, K_{\dot{y}}, K_\psi, K_{\dot{\psi}}, K_{FF}, K_{FB}]$)). Fig. 11 shows the variables used for each map, the connections between them and a representation of the weights in different colours. In total, 12 neural layers are required as interfaces for associative maps. The learning process is carried out locally only in associative maps that use an input error signal, with the same signal acting as a dopamine activation variable (d). This error adjusts STDP learning, modifying the associative map where there is more neuronal activity without affecting the other regions.

Table 2 summarizes the values used to implement the neural LQR, including the neural network parameters and the learning method. Discretization of the control function using neural associative maps requires defining the working range to establish the fitting curves. Table 2 also includes the discretization ranges (ρ) selected to obtain a good resolution in the map without compromising the operation of any variable that works outside the range and saturates the input.

These parameters have thus been selected based on similar data found in the literature and after a trial-and-error tuning process. More specifically, the number of neurons and the learning factor, which are the more critical parameters, were selected to ensure that all the necessary processes were accomplished with minimal computation cost and without adversely affecting the performance of the controller, as will be shown in the next section. Future work will address the development of an optimization process to adjust the network parameters.

Table 2
Neural network parameters.

Symbol	Description	Value
M	Number of neurons per layer	4 to 28
N	Number of neurons	240=12 (Layers)*m
u_0	Bias potential	0.2
T	Neuron time constant	2 ms
s_g	Synapse gain	21
τ_c	Eligibility time constant	10 ms
τ_d	Dopamine time constant	30 ms
Λ	Learning Factor	0 to 4
Δt	Time interval	1 ms
ρ_y	Range of lat. error	[−0.3.0.3] m
$\rho_{\dot{y}}$	Range of dlat. error	[−0.5,0.5] m/s
$\rho_{\psi y}$	Range of head. error	[−0.5.0.5]
$\rho_{\dot{\psi}}$	Range of dhead. error	[−0.5.0.5]
ρ_{Vx}	Speed range	[0160]

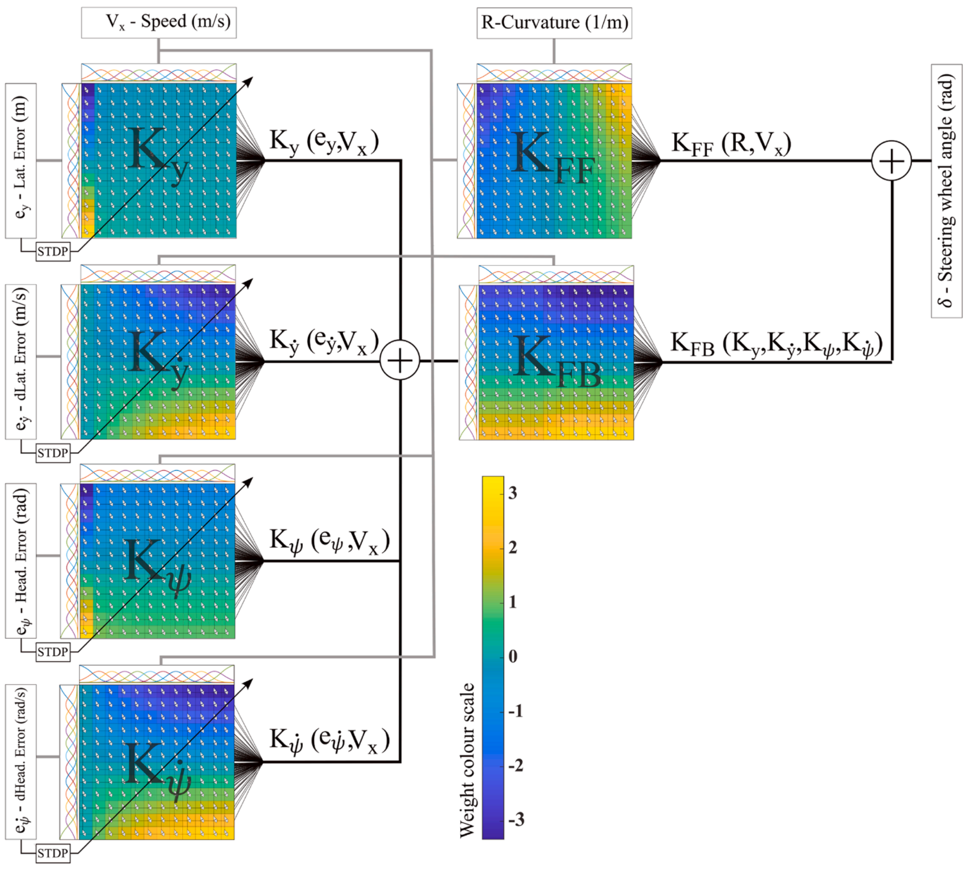


Fig. 11. LQR function neural representation.

5. Sensitivity, stability and performance comparison

In this section, three key aspects of the proposed methodology to develop SNN based are analyzed. First, a sensitivity analysis is carried out to assess how changes in the main SNN parameters affect the controller performance. Second, the stability of the controller is verified following an empirical approach. Third, the performance of an SNN controller is evaluated by comparing it with other conventional controllers, including an adaptative model predictive controller. These tests are performed using the model presented in Section 4 and with the vehicle conducting undergoing a double-lane change maneuver in two different road conditions, i.e. high adherence (μ=1) and low adherence (μ=0.2). The track is 4 m wide and the tests are performed at a longitudinal speed of 15 m/s.

5.1. Sensitivity analysis

Next, an evaluation of how variations in the main network parameters affect the SNN controller performance is conducted. The most influential parameters of the neural network, and therefore affecting the controller performance, are the number of neurons and the learning factor. Both must be carefully selected for the specific problem, which in this case is vehicle control. To perform this analysis, a double-lane change maneuver on a high-adherence road is reproduced.

Fig. 12 illustrates the sensitivity of the controller to these parameters. In the first case, the influence of selecting a number of neurons between 4 and 28 is evaluated. As can be seen in Fig. 12 (left plot), a low number of neurons results in insufficient discretization, which leads to a loss of information from the base controller and a degradation of its performance. On the contrary, if the number of neurons is too high, each neuron is assigned a very small portion of the control variable leading to a limited time to act, which also leads to poor performance. Consequently, the number of neurons per layer is set to 20 to carry out the simulations.

Second, as for the learning factor, low values result in behavior identical to the standard LQR ($\lambda = 0$), since the network does not adapt effectively when the road changes from high adherence to low adherence conditions. On the other hand, excessively high values cause the controller to aggressively try to minimize the error, resulting in severe vehicle oscillations, as shown in Fig. 12 (right plot). Consequently, a learning factor equal to 1 is selected for the simulations.

5.2. Controller stability

Analyzing the stability of a neural network-based controller is inherently challenging. Neural networks, due to their highly nonlinear and complex structures, act as black box approximators. This complexity makes it difficult to derive analytical stability conditions using classical methods such as Lyapunov theory. The nonlinearity and high dimensionality of the network, combined with the intricate interaction of its weights and activation functions, hinder the formulation of an appropriate Lyapunov function, making formal mathematical proofs of stability largely impracticable.

In this context, the empirical cell mapping technique offers a practical alternative for stability evaluation [48]. The method involves discretizing the state space into a finite number of cells and mapping the state transitions between these cells over time. By simulating the controller's response over a range of initial conditions, cell mapping provides insight into whether trajectories converge to stable attractors or diverge. This approach allows stability properties to be observed without the need for an explicit analytical model.

Fig. 13 shows the evolution of the lateral position and heading errors in simulations, evaluated using the cell mapping method on an 11×11 grid with a maximum lateral error of 5 m and a heading error of 40°. As can be seen, both algorithms tend to a stable response of zero error, being the SNN LQR faster due to the learning capabilities. In high adhesion conditions, both controllers behave similarly and are able to ensure vehicle stability. However, when the adhesion decreases, the LQR controller—although remaining stable—exhibits pronounced oscillations. In contrast, the adaptive properties of the neural implementation minimize these oscillations, thus ensuring stability across its entire operating range.

5.3. Controller comparison

Three controllers are used to compare the performance of the Neural LQR. To evaluate them, the double lane change maneuver is performed under both high and low adhesion conditions. This comparison evaluates the response of the controllers under design conditions (high adhesion), as well as their ability to handle dynamic changes caused by variations in road grip. For this purpose, seven key performance indicators are defined. These KPIs include the root-mean-square error (RMSE), the maximum value (MAX) and the standard deviation (SD) of the lateral position and heading angle errors. In addition, the execution time per iteration (DT) was added as another KPI to measure the maximum frequency at which the control loop can be executed and to evaluate the computational cost of the algorithms.

The first controller selected is the standard LQR, which is used as a reference. Although it initially behaves similarly to the neural LQR under high adhesion conditions, its performance deteriorates when adhesion decreases, leading to pronounced oscillations, especially in the heading angle. The second controller evaluated is a model predictive controller (MPC), chosen as a reference based on a well-established model. In this case, despite its predictive capabilities, it also leads to poor controller response when the adhesion changes, as its internal model does not fully reproduce the actual conditions. Furthermore, an adaptative model predictive controller (AMPC) is also included in this comparison [49]. This approach enables the path-following controller to adapt to changing operating conditions, thus allowing better adaptation to changes in adhesion levels.

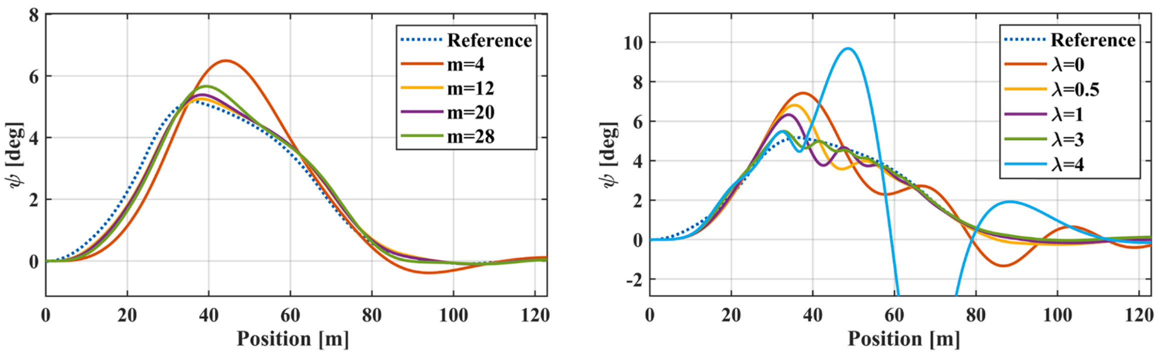


Fig. 12. Number of neurons (μ=1) (left) and learning rate influence (μ=0.2) (right).

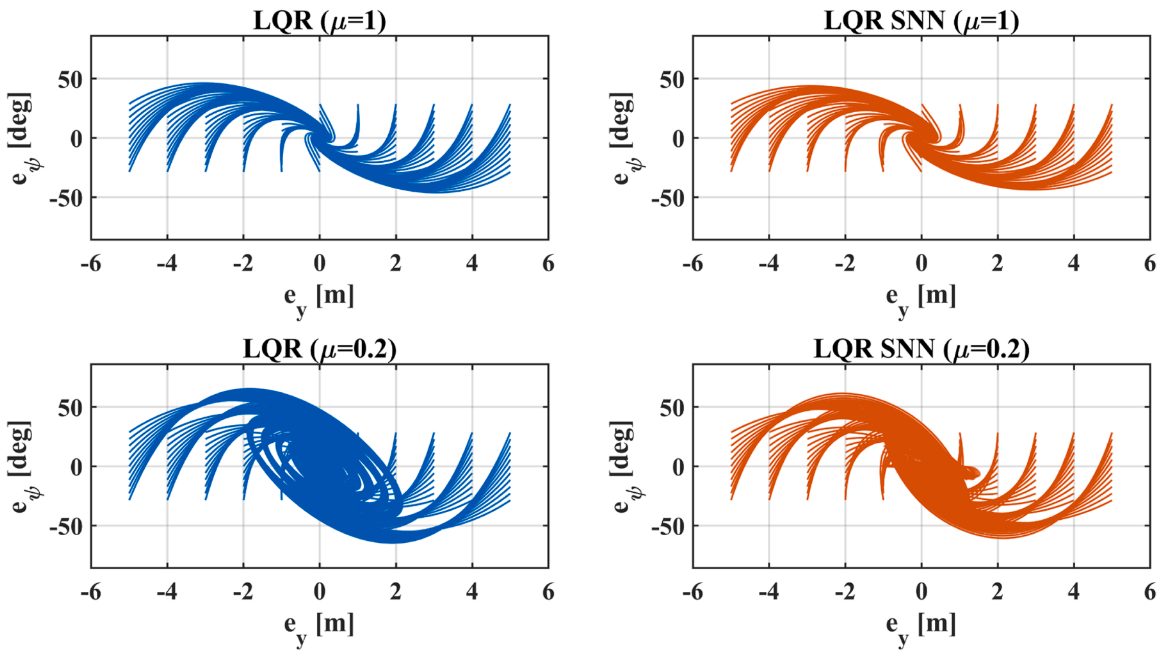


Fig. 13. LQR and LQR SNN error evolution in low and high adherence conditions.

As can be seen in Fig. 14, the performance of all controllers is similar under high adhesion conditions ($\mu = 1$), resulting in low position and heading angle errors. However, when adhesion decreases, the response of the controllers deteriorates significantly in some cases. First, both LQR and standard MPC give rise to high errors, as they lack the ability to adapt to dynamic changes. On the contrary, the AMPC and the Neural LQR are able to successfully reduce the error after adapting to the change in adhesion conditions. Table 3 summarizes the key performance indicators of the controllers' response in the low-adherence region of the test. The analysis of these results demonstrates the capability of the proposed SNN LQR controller to perform path-tracking tasks satisfactorily. As shown, the SNN LQR controller provides the best overall results in terms of lateral error and heading angle. On the contrary, the execution time per iteration of the proposed controller is the highest, but it still meets real-time requirements. In this regard, it is worth mentioning that the programming of the control algorithm on neuromorphic hardware will significantly reduce the execution time.

In this simulation, the learning factor is set to ($\lambda = 1$). During this maneuver, when $\mu = 1$, the error is very low, so learning is not activated. However, when the adhesion decreases, i.e., $\mu = 0.2$, the error committed is very high and the learning acts quickly to compensate for the error produced. This example is intended to showcase the controller's capabilities under extreme conditions. In a real environment, however, such discrepancies would not occur as frequently, and the learning process would be gradual, improving the vehicle's performance over time, similar to how a human driver would adapt. This behavior is demonstrated in the following section, where, on a track, the vehicle's performance gradually improves over time beyond the baseline controller.

Table 3
Controllers performance comparison.

	$RMSE_y$	$RMSE_\Psi$	MAX_y	MAX_Ψ	SD_y	SD_Ψ	DT (μs)
LQR	0.04	0.89	0.46	2.52	0.18	0.94	**2**
SNN	**0.01**	**0.17**	**0.16**	**1.2**	**0.06**	**0.42**	180
MPC	0.02	0.93	0.28	1.89	0.11	0.96	48
AMPC	0.02	0.23	0.34	1.37	0.14	0.48	98

As it has been shown, the key advantage of the proposed Neural LQR is that it does not require prior knowledge of road conditions to update the model, as is the case with AMPC. Significantly, when an increase in error is detected, the neural LQR activates learning and improves its response accordingly. This allows the controller to compensate for discrepancies between the model used in the design phase and the real

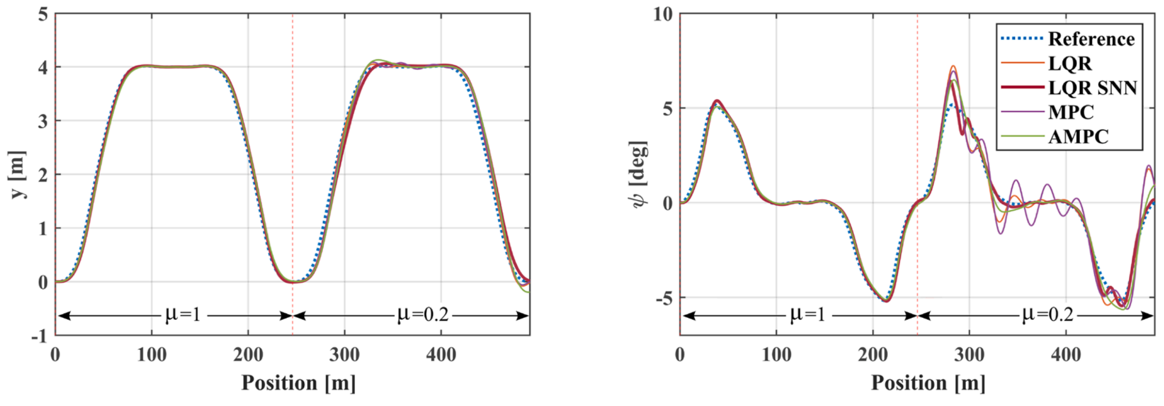


Fig. 14. Controller comparison in double lane change (Lateral position (left) and heading angle (right)).

vehicle being driven as well as to changes in the driving conditions, as will be demonstrated in the next section, where the performance of the controller is verified using a high-fidelity simulation of a long vehicle course.

6. Track simulation setup

To validate the performance of the neural LQR control algorithm, a simulation of a vehicle driving on a test track (Fig. 15) was performed. The simulation used a validated multi-body vehicle model in IPG-CarMaker, which was supplemented with Delft-Tyre 6.2 software representing tire dynamics. IPG Automotive's CarMaker simulation technology enables demonstrating the proposal's effectiveness in real-world dynamic scenarios. The test track was a 4.3-km-long, two-lane asphalt road with 20 curves of varying radii and distances between them, representing both urban and inter-urban driving environments. The IPG driver was set to cruise at the reference speed of 15 m/s.

The experiment setup was designed to assess both the performance and stability of the proposed system [17,50]. This study aimed to evaluate the stability of a neural network controller in an inter-urban environment. To evaluate robustness and performance, simulations and real-world tests under various conditions were conducted, comparing the results to those of a leading competitor. Specifically, simulations to replicate a long-path tracking test that included extreme conditions were carried out, incorporating sensor noise to reflect real-world scenarios. Performance indexes were obtained and compared to those of the competitor to verify the effectiveness of the proposal. Hence, a long-duration simulation was carried out, integrating curves of both large and small radii as well as 90- or 180-degree turns in a few meters to demonstrate the ability of the controller to maintain control in different situations. A measurement model was included to reproduce model uncertainties. The limitations of the actuator were also considered. The aforementioned features were included to yield results that showed that the proposed controller can maintain stability throughout its entire operating range. Throughout the testing, no stability issues were observed, leading to the conclusion that the controller's stability is ensured.

The control and learning neural network implementation was conducted using Matlab/Simulink, co-simulating with IPG software. To ensure real-time execution of the model and algorithm as well as the possibility to embed the control software, this co-simulation was carried out on hard real-time dSPACE Scalexio hardware, commonly used for rapid control prototyping.

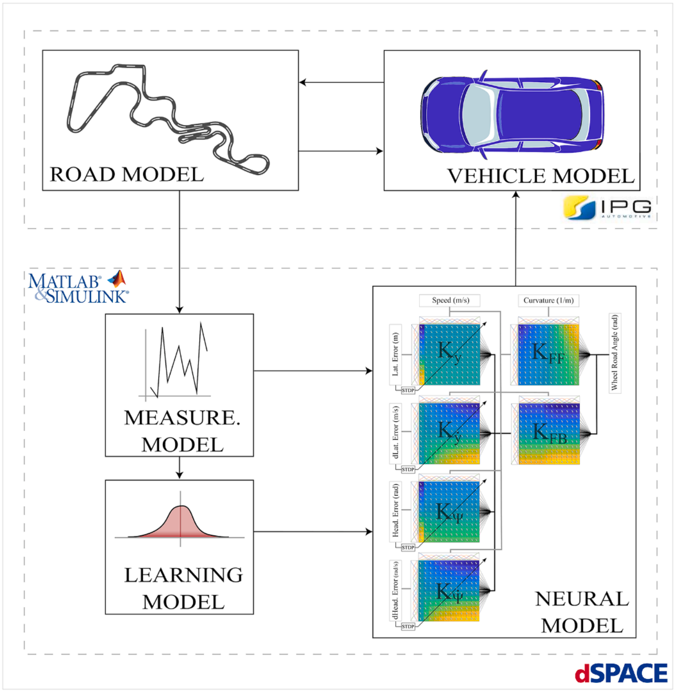


Fig. 15. Simulation control scheme.

7. Track simulation results

In this section, we present the results of control simulations using the configuration described in the previous section. It should be noted that the algorithm programmed in the simulation tool also included the learning strategy. Therefore, the whole proposal has been tested, validated and evaluated in a real-time environment. Fig. 16 shows the reduction in the lateral error (e_y) of the proposed controller compared with the LQR controller. In addition, Fig. 17 includes four graphs, each corresponding to one lap. The evolution of the error can be seen as a function of the vehicle's position on the circuit. It can be observed that the SNN-based adaptive controller reduces lateral error in each lap of the circuit without compromising stability.

However, the heading error (e_ψ) did not decrease because of the extremely low errors achieved (below 0.5°). After 40 laps (Fig. 18a), despite having a similar error value, a temporary deviation is observed between the benchmark and the proposed controller (Fig. 18b). This is because the target trajectory does not necessarily have to be the fastest or shortest one. Thus, the proposed adaptive controller requires slightly more time to complete the path but manages to provide fewer tracking errors.

Fig. 19 shows the percentage of time in which the controller is dealing with driving conditions that allow effective learning. As can be seen, the percentage decreases as the test progresses. Thus, in the first laps, effective learning is carried out only in around 10 % of the driving time. After 20 laps, the LQR SNN controller only learns for less than 5 s per lap.

Fig. 20 shows that the maximum lateral error decreases when the number of laps increases. After 40 iterations (laps), the convergence criterion of 1 mm/lap is reached and the simulation is stopped. At this point, after approximately three hours of driving, the simulation is considered to have been concluded. Regarding the convergence speed, it should be noted that at the learning process, the LQR SNN controller has no prior knowledge. Consequently, at the beginning of the test, the LQR SNN controller behaves like a conventional LQR controller, as can be seen in Fig. 17 (LQR vs LQR SNN in lap 1).

By analysing two consecutive curves (Fig. 21), it can be observed that the lateral error decreases significantly after only 10 laps. However, it was necessary to perform 30 more laps to reach the stopping criterion. The dopamine levels are so low that no further learning is achieved. As shown in lap 40, the driving becomes more aggressive to reduce the error, with high-frequency but low-amplitude oscillations observed. Depending on the type of controller sought, the convergence criterion can be adjusted and dopamine levels can be reduced to limit learning and obtain less aggressive control, such as the one achieved after 10 laps.

Finally, to compare the proposed algorithm with the LQR controller, Key Performance Indicators (KPIs) are again used to quantify the improvement achieved. Table 4 shows that, after learning, the proposed SNN controller reduced the error levels in the lateral and heading errors. However, it was observed that the heading error of the proposed controller is slightly higher than the one provided by the reference LQR controller. This is due to the fact that, to reduce the lateral error, the neural controller produces slightly higher yaw values.

Regarding the execution time, although the implementation of the neural network and learning method considerably increases the execution time of the algorithm, it still stays below the limit of 1000 us, which meets the real-time requirements at a frequency of 1 kHz.

As can be seen in Table 4, the results obtained are promising, demonstrating a significant reduction in lateral error and maintenance of system stability. Heading error did not decrease significantly, being

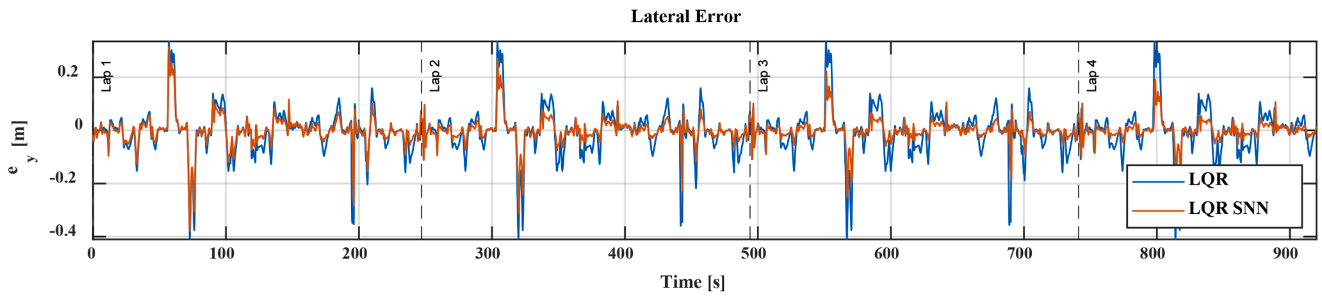


Fig. 16. Lateral error through the first four laps (over time).

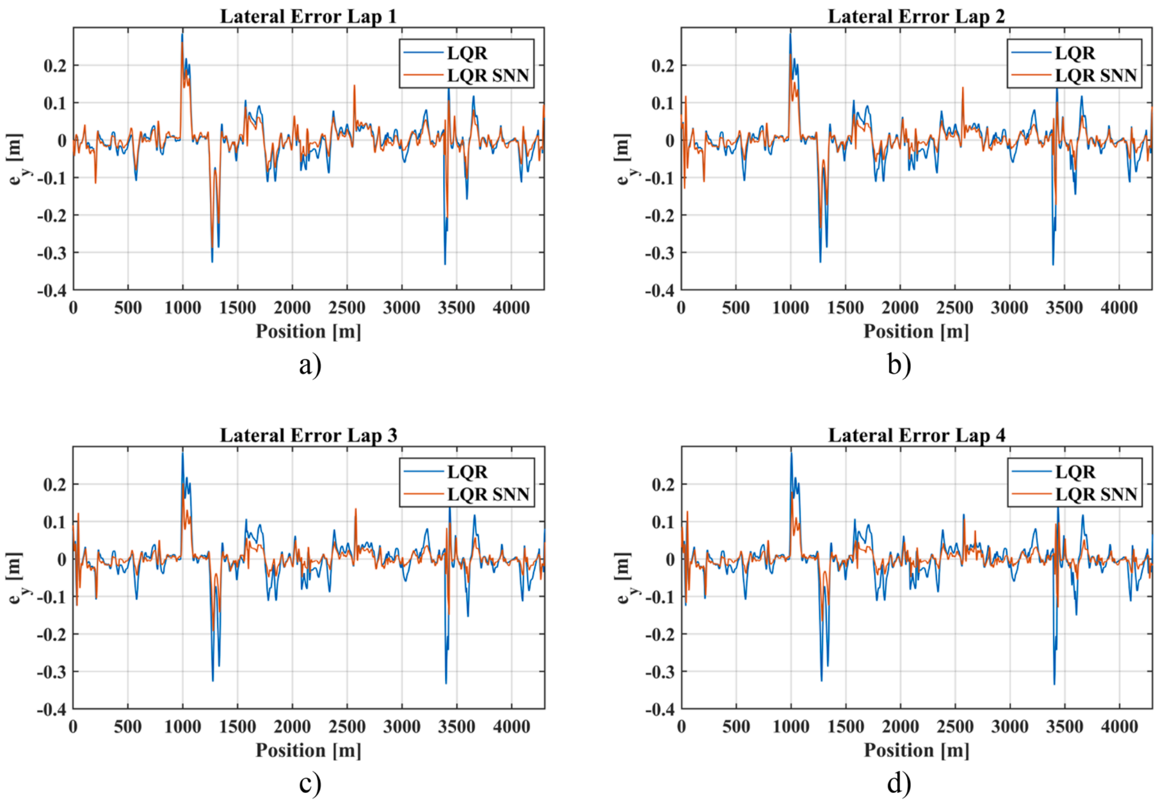


Fig. 17. Lateral error through the first four laps (relative to position).

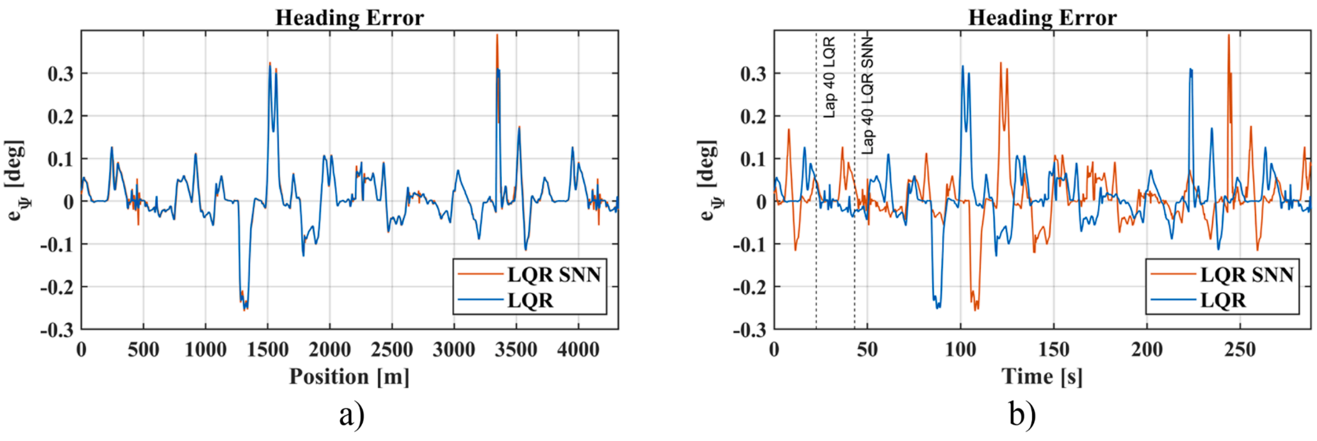


Fig. 18. Heading error through lap 40.

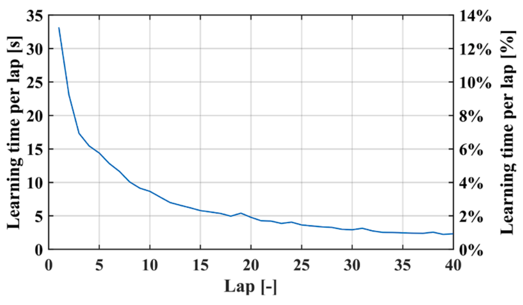


Fig. 19. Effective learning time per lap.

slightly higher than that of the conventional LQR controller in some cases. The implementation of the neural network increased the iteration execution time, which could be a potential limitation for systems with more stringent real-time requirements. This limitation can be addressed by implementing the neural controller on neuromorphic hardware.

8. Conclusion

This study proposes an adaptive path-following control algorithm combining an LQR controller with spiking neural networks and associative maps. By integrating these techniques, the proposed controller provides an efficient and optimal solution for controlling the lateral dynamics of a vehicle, which is critical for ensuring vehicle tracking performance.

The proposed approach provides an adaptation mechanism for a model-based controller. Using spiking neural networks, it is possible to perform STDP learning to adjust synaptic weights. This way, the controller has the capability to adapt to discrepancies between the simplified model used by the controller and the high-fidelity model or real-world conditions.

The simulations conducted using Matlab/Simulink and the well-known fully validated IPG CarMaker software to model the vehicle demonstrate the effectiveness and stability of the proposed controller under various curvature conditions on an inter-urban circuit. Key Performance Indicators were used to evaluate the performance of the controller.

In future work strategies to optimize the computational cost, such as eliminating low-activity neural connections, using bio-inspired neural structures and implementing the neural network on neuromorphic hardware, will be explored. In addition, this study focused on an under-actuated system in which only the steering angle could be controlled. To further improve vehicle performance, neural structures can be easily scaled up to act on fully actuated or over-actuated systems, enabling control over more variables, such as the traction and steering angle of different wheels. Methods for activating or deactivating the dopamine signal can also be researched, considering factors such as energy consumption or the time required to perform the required maneuver. Finally, the proposed neural structures can be adapted to perform other control tasks, such as braking, traction and longitudinal control. By initially replicating the response of a conventional controller, the neural network can subsequently modify its internal parameters to adapt its response to changing inputs, thanks to the learning capability. This adaptation can be very useful when dealing with different surfaces, load conditions and malfunctions.

CRediT authorship contribution statement

Javier Pérez Fernández: Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation,

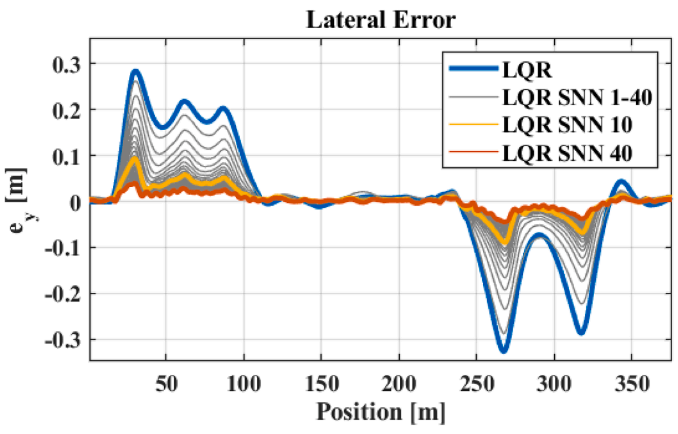


Fig. 21. Lateral error for the same corner.

Table 4
KPIs results.

	RMSE$_y$	RMSE$_\Psi$	MAX$_y$	MAX$_\Psi$	SD$_y$	SD$_\Psi$	DT (μs)
LQR	0.36	**0.48**	0.35	**0.31**	0.05	**0.06**	**2**
LQR$_{SNN}$ (LAP 10)	0.08	**0.48**	**0.28**	0.35	**0.02**	**0.06**	180
LQR$_{SNN}$ (LAP 40)	**0.04**	**0.48**	**0.28**	0.35	**0.02**	**0.06**	180

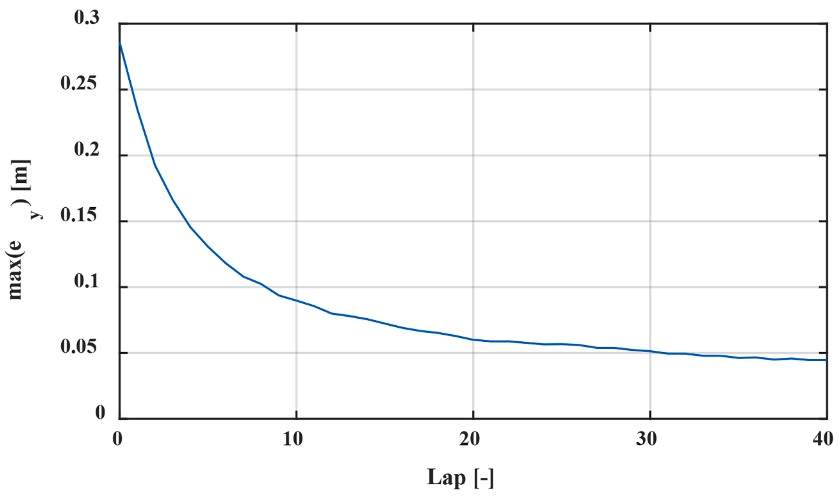


Fig. 20. Maximum lateral error for each lap.

Conceptualization. **Manuel Alcázar Vargas:** Writing – review & editing, Visualization, Validation, Methodology, Investigation, Formal analysis. **Juan A.Cabrera Carrillo:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Juan J.Castillo Aguilar:** Writing – review & editing, Supervision, Methodology, Investigation, Funding acquisition, Formal analysis. **Barys Shyrokau:** Writing – review & editing, Visualization, Validation, Supervision, Software, Resources, Methodology, Formal analysis.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

All authors report financial support was provided by Spain Ministry of Science and Innovation.

All authors report financial support was provided by University of Malaga.

All authors report financial support was provided by Regional Government of Andalusia. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was partly supported by the Ministry of Science and Innovation under grant PID2022-137095OB-I00, partly by the Regional Government of Andalusia under grant UMA18-FEDERJA-109 and partly by the University of Malaga as well as the Delft University of Technology.

Data availability

Data will be made available on request.

References

[1] O. Mokhiamar, M. Abe, Simultaneous optimal distribution of lateral and longitudinal tire forces for the model following control, J. Dyn. Syst. Meas. Control Trans. ASME 126 (2004) 753–763, https://doi.org/10.1115/1.1850533.

[2] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, et al., Stanley: the robot that won the DARPA grand challenge, J. Field Robot. 23 (2006) 661–692, https://doi.org/10.1002/rob.20147.

[3] T. Yang, Z. Bai, Z. Li, N. Feng, L. Chen, Intelligent vehicle lateral control method based on feedforward + predictive LQR algorithm, Actuators 10 (2021), https://doi.org/10.3390/act10090228.

[4] N. Chowdhri, L. Ferranti, F.S. Iribarren, B. Shyrokau, Integrated nonlinear model predictive control for automated driving, Control Eng. Pr. 106 (2021), https://doi.org/10.1016/j.conengprac.2020.104654.

[5] C. Sun, X. Zhang, Q. Zhou, Y. Tian, A model predictive controller with switched tracking error for autonomous vehicle path tracking, IEEE Access 7 (2019) 53103–53114, https://doi.org/10.1109/ACCESS.2019.2912094.

[6] J. Zhang, H. Wang, J. Zheng, Z. Cao, Z. Man, M. Yu, et al., Adaptive sliding mode-based lateral stability control of steer-by-wire vehicles with experimental validations, IEEE Trans. Veh. Technol. 69 (2020) 9589–9600, https://doi.org/10.1109/TVT.2020.3003326.

[7] L. Wiklendt, S. Chalup, R. Middleton, A small spiking neural network with LQR control applied to the acrobot, Neural Comput. Appl. 18 (2009) 369–375, https://doi.org/10.1007/s00521-008-0187-1.

[8] L. Liu, W. Zhao, Y.J. Liu, S. Tong, Y.Y. Wang, Adaptive finite-time neural network control of nonlinear systems with multiple objective constraints and application to electromechanical system, IEEE Trans. Neural Netw. Learn. Syst. 32 (2021) 5416–5426, https://doi.org/10.1109/TNNLS.2020.3027689.

[9] S. Xu, Z. Wu, Adaptive learning control of robot manipulators via incremental hybrid neural network, Neurocomputing 568 (2024), https://doi.org/10.1016/j.neucom.2023.127045.

[10] Z. Dong, T. Bao, M. Zheng, X. Yang, L. Song, Y. Mao, Heading control of unmanned marine vehicles based on an improved robust adaptive fuzzy neural network control algorithm, IEEE Access 7 (2019) 9704–9713, https://doi.org/10.1109/ACCESS.2019.2891106.

[11] T. Jiang, Y. Yan, D. Wu, S. Yu, T. Li, Neural network based adaptive sliding mode tracking control of autonomous surface vehicles with input quantization and saturation, Ocean Eng. 265 (2022) 112505, https://doi.org/10.1016/j.oceaneng.2022.112505.

[12] T. DeWolf, Spiking neural networks take control, Sci. Robot. 6 (2021) eabk3268, https://doi.org/10.1126/scirobotics.abk3268.

[13] J. Pérez, J.A. Cabrera, J.J. Castillo, J.M. Velasco, Bio-inspired spiking neural network for nonlinear systems control, Neural Netw. 104 (2018) 15–25, https://doi.org/10.1016/j.neunet.2018.04.002.

[14] R. Batllori, C.B. Laramee, W. Land, J.D. Schaffer, Evolving spiking neural networks for robot control, in: Proc. Comput. Sci., 6, 2011, pp. 329–334, https://doi.org/10.1016/j.procs.2011.08.060.

[15] X. Wang, Z.G. Hou, F. Lv, M. Tan, Y. Wang, Mobile robots' modular navigation controller using spiking neural networks, Neurocomputing 134 (2014) 230–238, https://doi.org/10.1016/j.neucom.2013.07.055.

[16] X. Wang, Z.G. Hou, A. Zou, M. Tan, L. Cheng, A behavior controller based on spiking neural networks for mobile robots, Neurocomputing 71 (2008) 655–666, https://doi.org/10.1016/j.neucom.2007.08.025.

[17] J. Pérez, M. Alcázar, I. Sánchez, J.A. Cabrera, M. Nybacka, J.J. Castillo, On-line learning applied to spiking neural network for antilock braking systems, Neurocomputing 559 (2023) 126784, https://doi.org/10.1016/j.neucom.2023.126784.

[18] Z. Bing, C. Meschede, G. Chen, A. Knoll, K. Huang, Indirect and direct training of spiking neural networks for end-to-end control of a lane-keeping vehicle, Neural Netw. 121 (2020) 21–36, https://doi.org/10.1016/j.neunet.2019.05.019.

[19] J. Liu, H. Lu, Y. Luo, S. Yang, Spiking neural network-based multi-task autonomous learning for mobile robots, Eng. Appl. Artif. Intell. 104 (2021), https://doi.org/10.1016/j.engappai.2021.104362.

[20] S. Barchid, J. Mennesson, J. Eshraghian, C. Djéraba, M. Bennamoun, Spiking neural networks for frame-based and event-based single object localization, Neurocomputing 559 (2023), https://doi.org/10.1016/j.neucom.2023.126805.

[21] Y. Li, Y. Liu, R. Li, L. Zhou, L. Dang, H. Mu, Q, Hyperspectral image classification based on faster residual multi-branch spiking neural network, Comput. Geosci. 197 (2025), https://doi.org/10.1016/j.cageo.2025.105864.

[22] S. Yang, Q. He, Y. Lu, B. Chen, Maximum entropy intrinsic learning for spiking networks towards embodied neuromorphic vision, Neurocomputing 610 (2024) 128535, https://doi.org/10.1016/j.neucom.2024.128535.

[23] S. Yang, H. Wang, B. Chen, SIBoLS: robust and energy-efficient learning for spike-based machine intelligence in information bottleneck framework, IEEE Trans. Cogn. Dev. Syst. 16 (5) (2024) 1664–1676, https://doi.org/10.1109/TCDS.2023.3329532.

[24] S. Yang, B. Chen, Effective surrogate gradient learning with high-order information bottleneck for spike-based machine intelligence, IEEE Trans. Neural Netw. Learn. Syst. 36 (1) (2025) 1734–1748, https://doi.org/10.1109/TNNLS.2023.3329525.

[25] C. Du, F. Liu, B. Kang, T. Hou, Speech emotion recognition based on spiking neural network and convolutional neural network, Eng. Appl. Artif. Intell. 147 (2025), https://doi.org/10.1016/j.engappai.2025.110314.

[26] A. Schmeitz, J. Zegers, J. Ploeg, M. Alirezaei, Towards a generic lateral control concept for cooperative automated driving, in: 2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems, IEEE, 2017.

[27] R. Poli, J. Koza, Genetic programming. Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, 2nd Edition, Springer US, 2014, pp. 143–186, https://doi.org/10.1007/978-1-4614-6940-7_6.

[28] H. Bagherinezhad, M. Rastegari, A. Farhadi, LCNN: lookup-based convolutional neural network, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[29] T. Poggio, A theory of how the brain might work, Cold Spring Harb. Symp. Quant. Biol. 55 (1990) 899–910.

[30] F. Schubert, C. Gros, Nonlinear dendritic coincidence detection for supervised learning, Front. Comput. Neurosci. 15 (2021), https://doi.org/10.3389/fncom.2021.718020.

[31] M. Christof, K.B.W.S Pi, Learning: on radial basis functions and cortical associative learning, Adv. Neural Inf. Process. Syst. 2 (2000) 474–481.

[32] E.R. Kandel, The molecular biology of memory storage : a dialogue between genes and synapses, Science 294 (2015) 1030–1039.

[33] B.D. Baptista Filho, E.L.L. Cabrai, A.J Soares, A new approach to artificial neural networks, IEEE Trans. Neural Netw. 9 (1998) 1167–1179, https://doi.org/10.1109/72.728360.

[34] K.K. Cover, B.N. Mathur, Axo-axonic synapses: diversity in neural circuit function, J. Comp. Neurol. 529 (2021) 2391–2401, https://doi.org/10.1002/cne.25087.

[35] K. Doya, M. Kawato, Neural mechanisms of learning and control, IEEE Control Syst. Mag. 21 (2001) 42–54, https://doi.org/10.1109/37.939943.

[36] Z. Bing, C. Meschede, F. Röhrbein, K. Huang, A.C. Knoll, A survey of robotics control based on learning-inspired spiking neural networks, Front. Neurorobot. 12 (2019), https://doi.org/10.3389/fnbot.2018.00035.

[37] J. Pérez Fernández, M. Alcázar Vargas, J.M. Velasco García, J.A. Cabrera Carrillo, J.J Castillo Aguilar, A biological-like controller using improved spiking neural networks, Neurocomputing 463 (2021) 237–250, https://doi.org/10.1016/j.neucom.2021.08.005.

[38] Y. Shan, B. Zheng, L. Chen, L. Chen, D. Chen, A reinforcement learning-based adaptive path tracking approach for autonomous driving, IEEE Trans. Veh. Technol. 69 (2020) 10581–10595, https://doi.org/10.1109/TVT.2020.3014628.

[39] L. Chen, Y. Chen, X. Yao, Y. Shan, L. Chen, An adaptive path tracking controller based on reinforcement learning with urban driving application. 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 2019, pp. 2411–2416, https://doi.org/10.1109/IVS.2019.8814130.

[40] C.C. Aggarwal, Neural Networks and Deep Learning, Springer, 2018.

[41] A.N. Burkitt, A review of the integrate-and-fire neuron model: II. Inhomogeneous synaptic input and network properties, Biol. Cybern. 95 (2006) 97–112, https://doi.org/10.1007/s00422-006-0082-8.

[42] C. Eliasmith, How to Build a Brain: a Neural Architecture for Biological Cognition, Oxford Scholarship, 2011.

[43] A. Zhao, S. Xing, X. Wang, J.Q. Sun, Radial basis function neural networks for optimal control with model reduction and transfer learning, Eng. Appl. Artif. Intell. 136 (2024) 108899, https://doi.org/10.1016/j.engappai.2024.108899.

[44] F. Pulvermüller, Neurobiological mechanisms for language, symbols and concepts: clues from brain-constrained deep neural networks, Prog. Neurobiol. 230 (2023), https://doi.org/10.1016/j.pneurobio.2023.102511.

[45] C. Koch, T. Poggio, Multiplying with synapses and neurons. Single Neuron Computation, Elsevier, 1992, pp. 315–345, https://doi.org/10.1016/b978-0-12-484815-3.50019-0.

[46] R. Liu, Y. He, S. Jiang, L. Wang, Q. Wan, Synaptic plasticity modulation and coincidence detection emulated in multi-terminal neuromorphic transistors, Org. Electron. 92 (2021), https://doi.org/10.1016/j.orgel.2021.106125.

[47] E.M. Izhikevich, Solving the distal reward problem through linkage of STDP and dopamine signaling, BMC Neurosci. 2 (2007) 1–2, https://doi.org/10.1093/cercor/bhl152.

[48] Sun, J.Q. & Xiong, F. & Schütze, O. & Hernández Castellanos, C.. (2018). Cell mapping methods, algorithmic approaches and applications. https://doi.org/10.1007/978-981-13-0457-6.

[49] A. Bemporad, M. Morari, Robust model predictive control: a survey, in: A. Garulli, A. Tesi (Eds.), Robustness in Identification and control. Lecture Notes in Control and Information Sciences, Robustness in Identification and control. Lecture Notes in Control and Information Sciences, 245, Springer, London, 1999, https://doi.org/10.1007/BFb0109870.

[50] F. Pretagostini, L. Ferranti, G. Berardo, V. Ivanov, B. Shyrokau, Survey on wheel slip control design strategies, evaluation and application to antilock braking systems, IEEE Access 8 (2020) 10951–10970, https://doi.org/10.1109/ACCESS.2020.2965644.