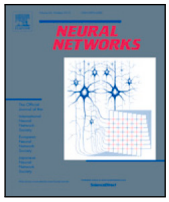
Neural Networks 180 (2024) 106656

Contents lists available at ScienceDirect

Neural Networks

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



Full Length Article

A new hybrid learning control system for robots based on spiking neural networks

Vahid Azimirad [a,*], S. Yaser Khodkam [b], Amir Bolouri [c]

[a] School of Engineering, University Of Kent, UK
[b] Faculty Of Mechanical Engineering, University Of Tabriz, Tabriz, Iran
[c] Faculty Of Engineering, University Of the West of England, Bristol, UK

ARTICLE INFO

Keywords:
Spiking neural networks
Reinforcement learning
Robot controller
Dopamine modulated spike timing depending plasticity
Fractional Order PID (FOPID)
Feedback linearization

ABSTRACT

This paper presents a new hybrid learning and control method that can tune their parameters based on reinforcement learning. In the new proposed method, nonlinear controllers are considered multi-input multi-output functions and then the functions are replaced with SNNs with reinforcement learning algorithms. Dopamine-modulated spike-timing-dependent plasticity (STDP) is used for reinforcement learning and manipulating the synaptic weights between the input and output of neuronal groups (for parameter adjustment). Details of the method are presented and some case studies are done on nonlinear controllers such as Fractional Order PID (FOPID) and Feedback Linearization. The structure and the dynamic equations for learning are presented, and the proposed algorithm is tested on robots and results are compared with other works. Moreover, to demonstrate the effectiveness of SNNFOPID, we conducted rigorous testing on a variety of systems including a two-wheel mobile robot, a double inverted pendulum, and a four-link manipulator robot. The results revealed impressively low errors of 0.01 m, 0.03 rad, and 0.03 rad for each system, respectively. The method is tested on another controller named Feedback Linearization, which provides acceptable results. Results show that the new method has better performance in terms of Integral Absolute Error (IAE) and is highly useful in hardware implementation due to its low energy consumption, high speed, and accuracy. The duration necessary for achieving full and stable proficiency in the control of various robotic systems using SNNFOPD, and SNNFL on an Asus Core i5 system within Simulink's Simscape environment is as follows:
– Two-link robot manipulator with SNNFOPID: 19.85656 hours
– Two-link robot manipulator with SNNFL: 0.45828 hours
– Double inverted pendulum with SNNFOPID: 3.455 hours
– Mobile robot with SNNFOPID: 3.71948 hours
– Four-link robot manipulator with SNNFOPID: 16.6789 hours.
This method can be generalized to other controllers and systems like robots.

1. Introduction

One of the challenges in traditional control systems is the lack of compatibility with new AI-based methods like Spiking neural networks (SNNs). They are energy-efficient processing systems that have applications in AI and reinforcement learning (Yamazaki, V., & D., 2022). Here, we propose a method for approximating any function such as a nonlinear controller algorithm through SNNs, and apply reinforcement learning to obtain optimum parameters. The proposed method enables us to integrate any traditional control system with reinforcement learning algorithms that are implemented in the SNNs context. It results in a hybrid learning-control algorithm like the critic–actor system which is applicable in robotics.

In recent years, there has been a convergence between the fields of neuroscience, engineering, and artificial intelligence, resulting in the development of novel control methods. One such method is reinforcement learning, a biologically inspired algorithm that is compatible with complex real-world engineering systems (Azimirad & M., 2020). In contrast, traditional nonlinear control systems are effective in controlling complex systems such as robots but cannot learn and update parameters. This paper presents a new algorithm for replacing nonlinear controllers through SNNs with a reinforcement learning perspective.

* Corresponding author.
E-mail addresses: v.azimirad@kent.ac.uk (V. Azimirad), y.khodkam98@ms.tabrizu.ac.ir (S.Y. Khodkam), amir.bolouri@uwe.ac.uk (A. Bolouri).

https://doi.org/10.1016/j.neunet.2024.106656
Received 3 December 2023; Received in revised form 23 April 2024; Accepted 19 August 2024
Available online 22 August 2024

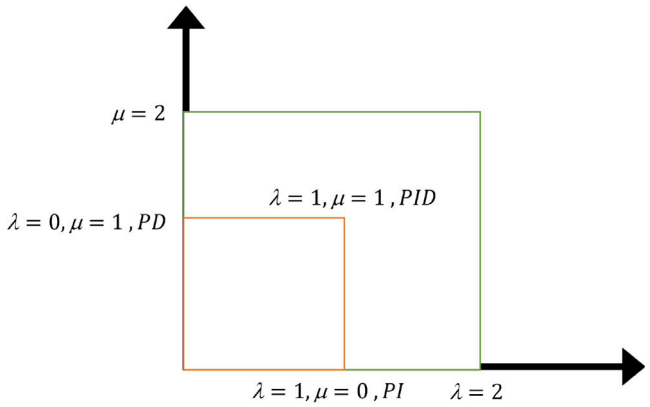


Fig. 1. FOPID controller converge.

It offers numerous engineering applications (e.g. control of nonlinear robotic systems) and will assist in overcoming common challenges (such as parameter adjustment and high energy consumption).

Finding optimal parameter values in designing a nonlinear controller such as Fractional Order PID or feedback linearization is a significant challenge. Although there are some methods for tuning controllers such as the Ziegler–Nichols method, for nonlinear controllers such as Fractional Order PID (FOPID), it is a more complex subject. Valerio and J. Costa have presented a FOPID regulator based on Ziegler–Nichols criteria (Cao & G., 2005), and more recently, Shalaby et al. propose a critic–actor optimization method for the FOPID controller (Shalaby, B., & T., 2023). However, these methods are not energy efficient and among the above works, none of them have utilized reinforcement learning through spiking neural networks (SNNs) for parameter approximation.

In Vinagre and I (2007), an automatic tuning method for FOPID based on the relay test has been introduced. Optimization algorithms such as genetic algorithm (Cao & G., 2005) and particle swarm optimization (Cao & G., 2006) have also been proposed for tuning FOPID controller parameters. In 2022, Lin Xu and colleagues utilized a combination of backstepping and FOPID techniques for tracing the trajectory of a differential drive mobile robot. To enhance the FOPID parameter values, they employed the beetle swarm optimization algorithm (Xu, D., S., & C., 2022). A FOPID fuzzy controller developed by Jiawen Zhang was utilized to regulate a two-wheeled self-balancing robot (TWSBR) system in an inclined environment (Zhang, Z., G., & D., 2021). A strategy has been presented in this regard, which utilizes a modified firefly algorithm (MFA) and particle swarm optimization to adjust torque and effective speed for BLDC motors. This approach effectively resolves the issue of uncertainty caused by load changes (Kommula & K., 2022). Furthermore, M. Abed used the fruit fly swarm algorithm to optimize the FOPID parameters of the nonlinear neural network for controlling the mobile robot, which has effective results (Abed, A., & k. A., 2022), but he did not study SNNs.

Some methods for parameter tuning in PID controllers have also utilized supervised learning based on Artificial Neural Networks (ANNs). For instance, in Lee and D.-W. (2021), a combination of two networks, Long Short-Term Memory (LSTM) and an Artificial Neural Network, was used to tune the PID parameters. Fuzzy FOPID controllers and PSO algorithms have been combined to regulate the position of a pneumatic cylinder in a nonlinear state (Muftah & M., 2022). In another work, supervised learning based on ANN has been used for face recognition and FOPID controller to regulate robot engine position (Hsu, C., C., F., & C., 2022). Webb et al. designed a three-in-one network using the Izhikevich model and the Hodgkin–Huxley type, where the PID parameters were set equal to the synaptic weights, but they only studied supervised learning (Webb & S., 2011). However, none of these works has studied reinforcement learning in the SNNs platform for the application of AI in the parameter tuning of nonlinear controllers like FOPID and feedback linearization.

Mohit Mehindir et al. (2019), to facilitate accurate tracking, offered a strategy of Simple Learning for the linearization of feedback in aerial robots. It was based on the gradient descent method and the aim was to obtain the rules for feedback controller gains and disturbance estimates in the feedback control law (Mehndiratta, E., M., & E., 2019). But his method was not energy efficient. In Hoang, J., and S. (2021), the combined method of feedback linearization and sliding mode controller was presented for controlling the vibration of an excavator based on a dynamic model and did not study reinforcement learning. In 2020, Anxing Liu et al. presented a smooth switching control method of feedback linearization and the Port-Controlled Hamiltonian combination, which was designed based on position error, to solve the conflict between dynamic performance and the steady state of a robot system (L. & Yu, 2020). However, none of the previous works in this area studied SNNs and reinforcement learning in the parameter adjustment of controllers.

The main contribution of this research is to present a new SNN-based algorithm for hybrid learning and control of robots. The proposed reinforcement learning system acts as a critic system and is used for tuning the parameters of the nonlinear controllers (as actor systems). The algorithm of the new proposed method has been introduced and its integration with other traditional controllers is discussed. E.g., it is integrated with the Fractional Order PID (FOPID) and feedback linearization controllers. The application of the new algorithm in robotics is studied and the results are compared with previous works. The new method shows a better performance in comparison with them. Then it is extended to optimize the parameters of the Feedback Linearization method (as a basic nonlinear controller in robotics).

1.1. Reinforcement learning in robotics

Reinforcement learning is a type of machine learning that enables robots to learn tasks through trial and error by receiving feedback in the form of rewards or penalties based on their actions. This feedback helps the robot to adjust its behavior and improve its performance over time. Reinforcement learning can enable robots to learn tasks by providing a framework for the robot to learn from its interactions with the environment. The primary objective of reinforcement learning in training a robot to acquire new skills is to enable it with three essential capabilities:

1. Learning tasks that are beyond the physical capabilities of a human teacher, such as lifting heavy weights. This allows the robot to perform tasks that require superior strength or endurance.

2. Acquiring the ability to optimize goals, which is a complex problem lacking any known analytical formula or solution. Even a human teacher may not possess knowledge of the optimal solution. By utilizing a function, the robot can minimize errors and energy consumption, thus achieving optimization.

3. Demonstrating adaptive learning in previously encountered tasks that may have variations or unseen challenges. For instance, the robot can learn to navigate smoothly along a curved path, even if it was initially trained on a straight path. This adaptability enables the robot to handle diverse scenarios effectively (Miljković, M., L., & B., 2013). One specific example of reinforcement learning enabling robots to learn tasks is in robotic manipulation. Robots can learn to grasp objects, manipulate them, and perform complex tasks by using reinforcement learning algorithms to optimize their actions based on feedback from the environment (Kormushev, C., & C., 2013; Pane, N., & B., 2019; Yamada, Englert, & J., 2021).

Another example is autonomous navigation, where robots can learn to navigate through complex environments by using reinforcement learning to learn optimal paths and avoid obstacles (Liu, D., C., S., & D., 2020; Quan, Y., & Y., 2020).

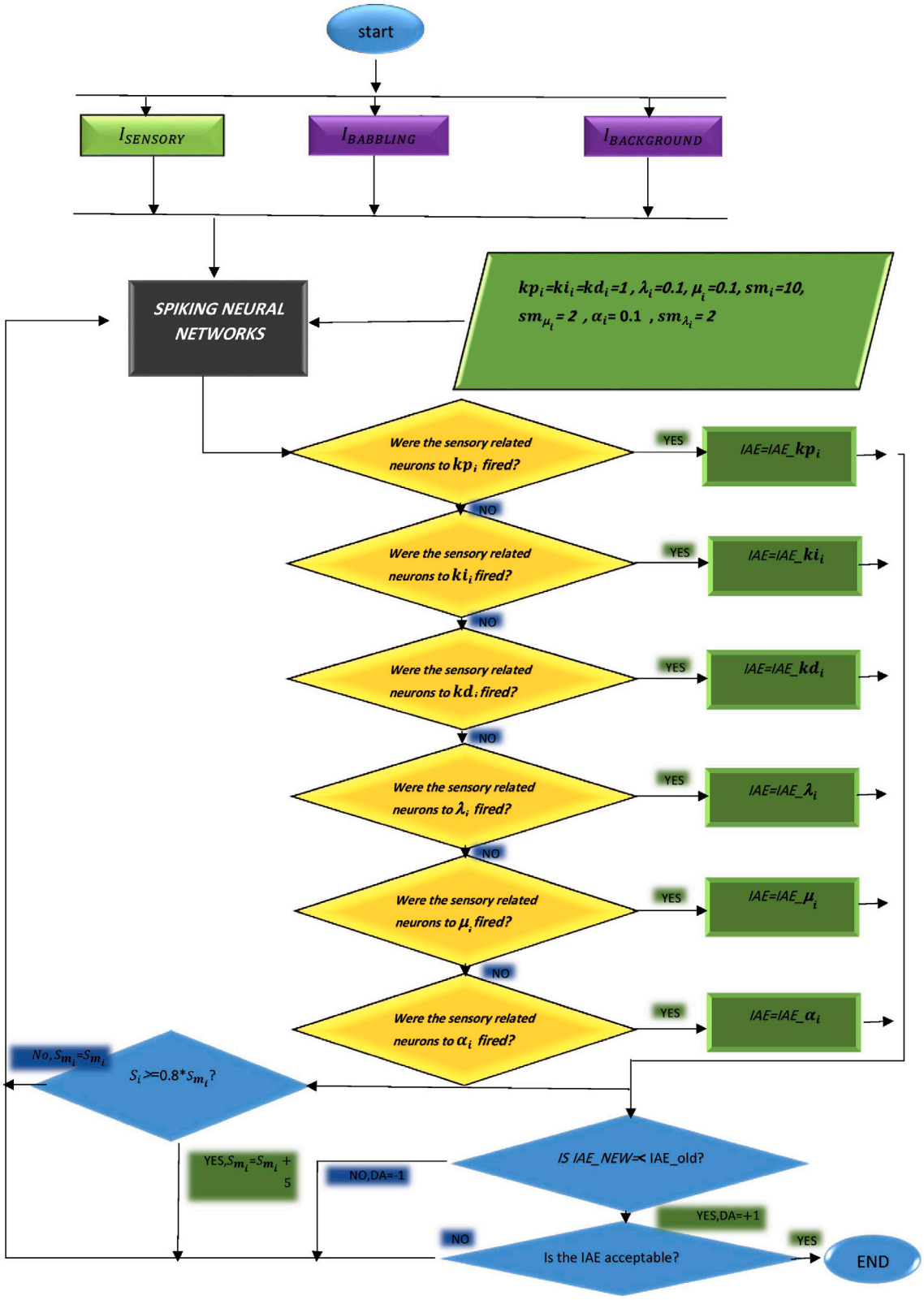


Fig. 2. The SNNFOPID Flowchart.

1- Initialize the parameters.
2- SNN design for each degree of freedom.
3- Inject the sensory current to input neurons.
4- Inject babbling and background currents to the motor neurons.
5- Biologically, it is widely observed and accepted that sensory neurons consistently exhibit spike activity prior to motor neurons. this fundamental principle should be duly considered
and implemented.
6- If v≥30 mv the neurons will spike.
7.- If the neurons related to kp_i spike,
7.1- If $IAE_{kp_i\ new} < IAE_{kp_i\ old}$

7.2- else,
DA= -1
8- If the neurons related to ki_i spike,
8.1- If $IAE_{kI_i} < IAE_{ki_i\ old}$
DA=+1
8.2 else,
DA= -1
9- If the neurons related to kd_i spike,
9.1- If $IAE_{kd_i\ new} < IAE_{kd_i\ old}$
DA=+1
9.2 else,
DA= -1
10- If the neurons related to μ_i spike,
10.1- If $IAE_{\mu_i\ new} < IAE_{\mu_i\ old}$
DA=+1
10.2- else,
DA= -1
11- If the neurons related to λ_i spike,
11.1- If $IAE_{\lambda_i\ new} < IAE_{\lambda_i\ old}$
DA=+1
11.2- else,
DA= -1
12- If the neurons related to α_i spike,
12.1- If $IAE_{\alpha_i\ new} < IAE_{\alpha_i\ old}$
DA=+1
12.2- else,
DA= -1
13- Repeat the process of 7-12
14- If IAE value is acceptable
15- end.

Fig. 3. Peusecode of SNNFOPID.

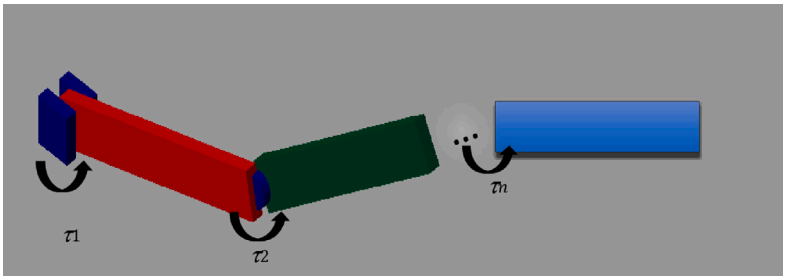


Fig. 4. Schematic of n-degrees of freedom serial manipulator.

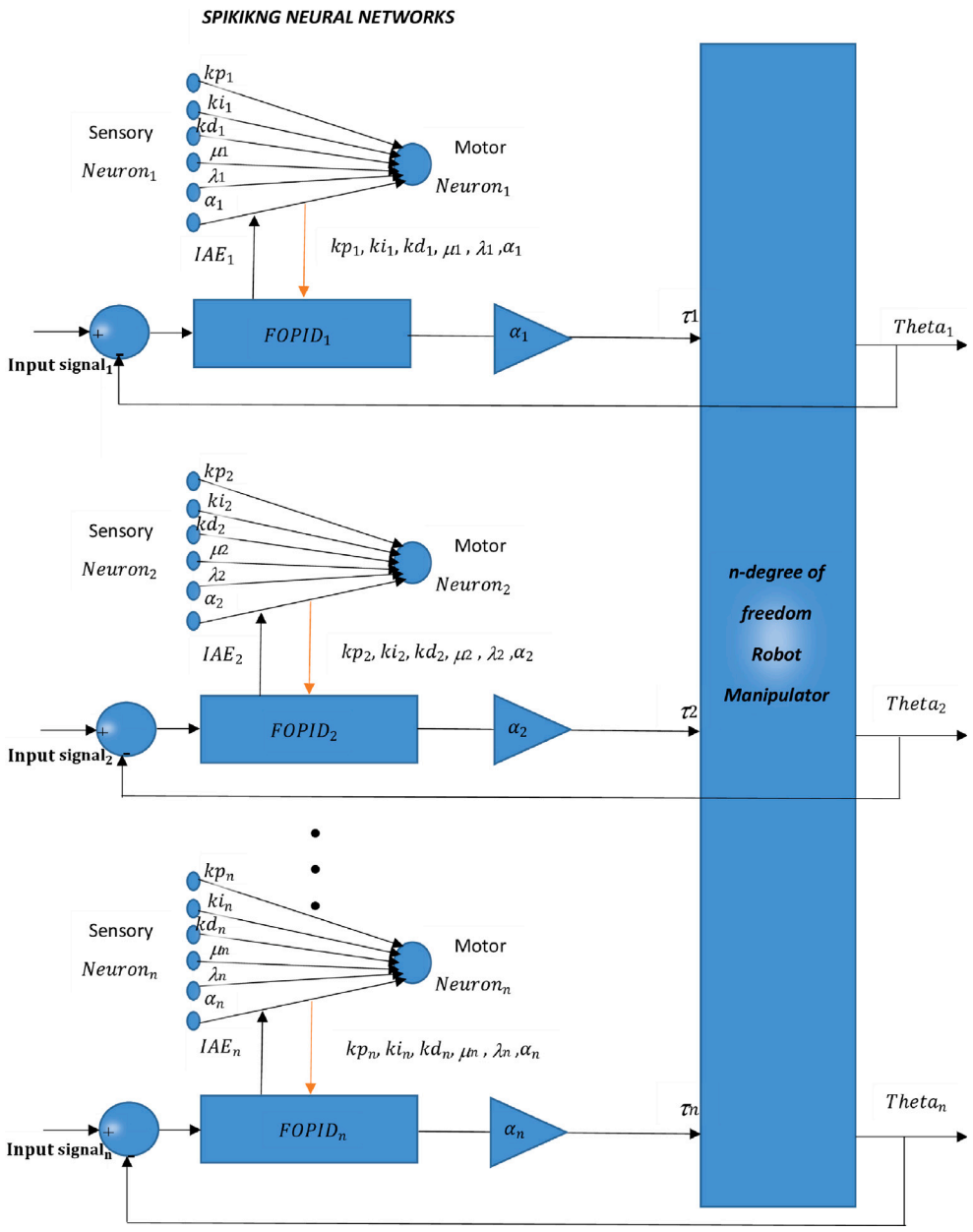


Fig. 5. The structure of connecting SNN-based learning system for controlling an n-DOF robot manipulator with n FOPID controllers.

1.2. Reinforcement learning for collaborative tasks

Empirical control based on reinforcement learning for collaborative tasks in the real world is already being used in various applications. These include autonomous vehicles, robotics, and industrial automation, among others. Researchers and companies are actively working on developing and implementing reinforcement learning algorithms for collaborative tasks to improve efficiency and performance in real-world settings. for instances: Uri Kartoun et al. introduce a new reinforcement learning algorithm, CQ(λ), that facilitates collaborative learning between a robot and a human. Based on the Q(λ) approach, the algorithm leverages human intelligence and expertise to expedite the learning process. It provides the robot with self-awareness, allowing it to adaptively switch its collaboration level from autonomous to semi-autonomous based on its learning performance. The approach is demonstrated and evaluated using a fixed-arm robot for finding the optimal shaking policy to empty the contents of a plastic bag, showing faster convergence compared to traditional Q(λ) reinforcement learning (Kartoun, Stern, & Edan, 2010). Wenshuai Zhao et al. specifically analyzed how multi-agent reinforcement learning can bridge the reality gap in distributed multi-robot systems with non-homogeneous operations. In this work, the effect of sensing, calibration, and accuracy mismatches in distributed reinforcement learning using proximal policy optimization (PPO) is introduced and discusses how different types of perturbances and the number of agents experiencing them affect collaborative learning efforts (Zhao & W., 2020). Ali Shafti et al. conducted a real-world collaborative maze game with humans and robots, focusing on implicit interaction. Using deep reinforcement learning, the robotic agent achieved results in 30 min without pretraining. Then it is studied how humans and robots colearned a policy for the game and found that each participant's agent represented their playing style (Shafti & F., 2020).

The following sections detail the contents of this paper. In Section 2, the applications of ANN and SNN are compared. Section 3 discusses the structure of spiking neural networks while Section 4 highlights the significance of FOPID. In Section 5, the SNNFOPID algorithm and

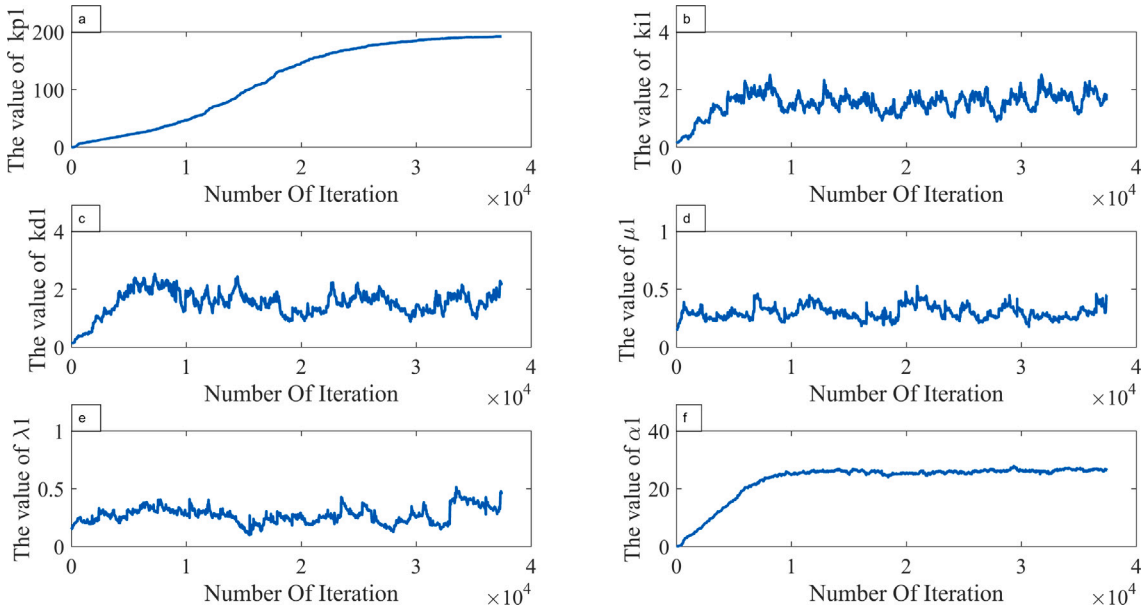


Fig. 6. The stability of parameters of the SNNFOPID controller for $link_1$.

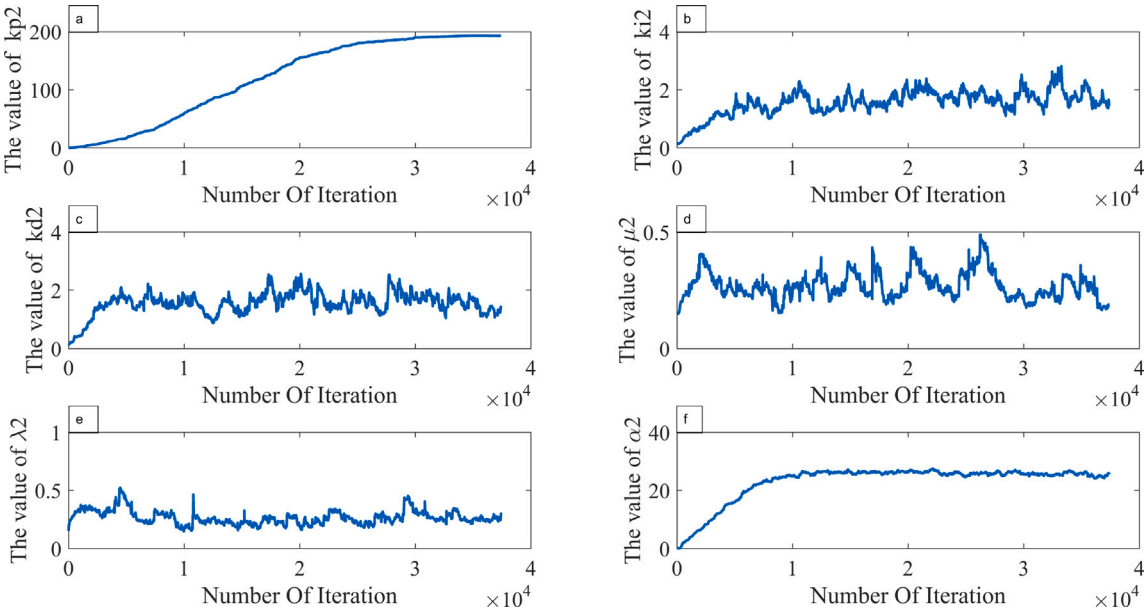


Fig. 7. The stability of parameters of the SNNFOPID controller for $link_2$.

the structure of SNNs are described. Moreover, it also delves into the implementation of the SNNFOPID algorithm, dynamic equations of (two links robot manipulator, double inverted pendulum, and mobile robot), and simulation results. The application of reinforcement learning in Feedback Linearization (FL) is presented in Section 6. Lastly, the significance and outcomes of this study are discussed.

2. SNNs or ANNs?

1. Computational Model:

ANNs: they are based on the concept of artificial neurons or perceptrons that process inputs and produce outputs using weighted connections and activation functions. They typically use continuous-valued activations and operate in discrete time steps. SNNs: they are more biologically inspired and model the behavior of real neurons in the brain more closely. They use a spiking or event-based representation, where information is encoded in the timing of discrete spikes or action potentials.

2. Information Representation:

ANNs: they represent information using continuous-valued activations, where the strength of the activation represents the importance or relevance of a feature. SNNs: SNNs represent information using discrete spikes or events. The timing of these spikes carries information, and the frequency or pattern of spikes can encode different features.

3. Temporal Processing:

ANNs: ANNs typically do not explicitly model temporal dynamics, as they operate in discrete time steps and process inputs independently at each step.

SNNs: SNNs naturally capture temporal dynamics due to their event-based representation. The timing of spikes allows for precise temporal processing and synchronization of information.

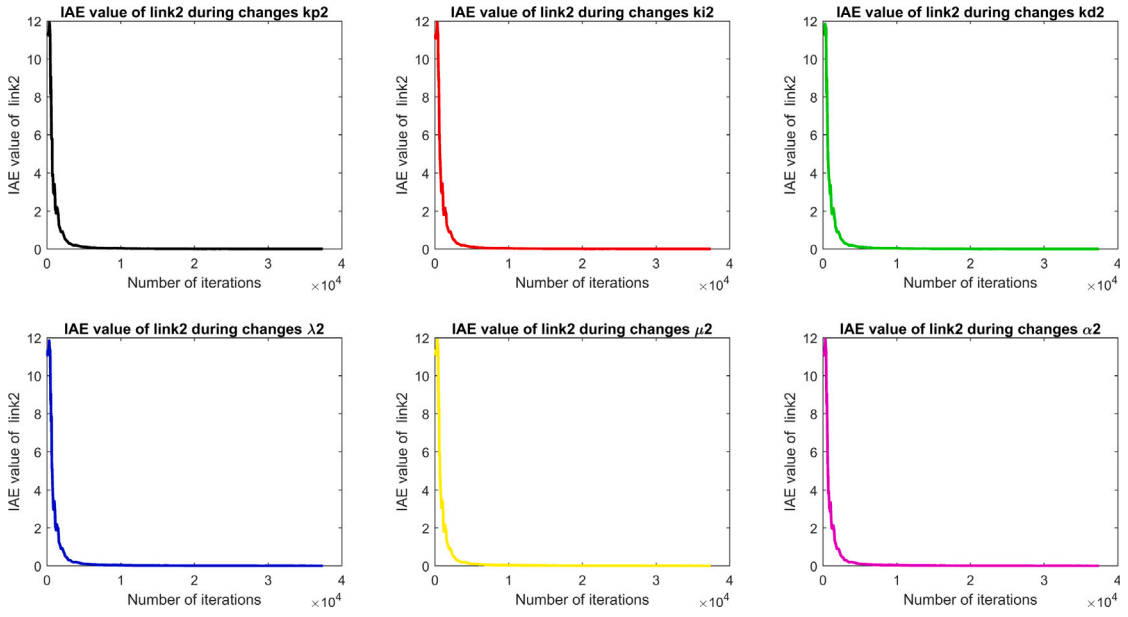


Fig. 8. (a) IAE value of link1 during changes kp1, (b) IAE value of link1 during changes ki1, (c) IAE value of link1 during changes kd1, (d) IAE value of link1 during changes λ_1, (e) IAE value of link1 during changes μ_1, (f) IAE value of link1 during changes α_1.

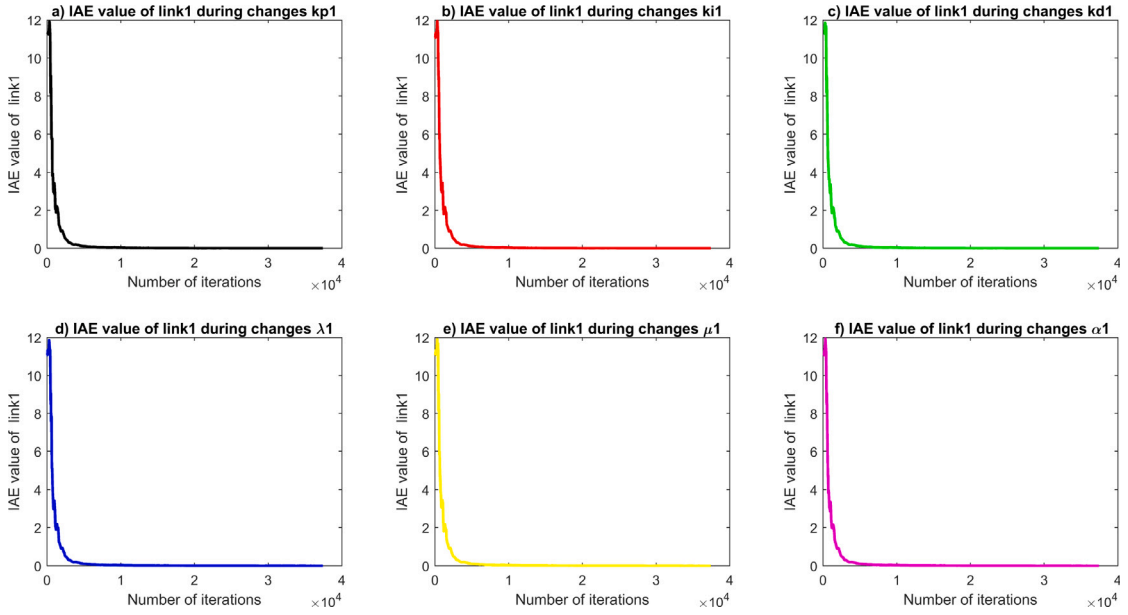


Fig. 9. (a) IAE value of link2 during changes kp2, (b) IAE value of link2 during changes ki2, (c) IAE value of link2 during changes kd2, (d) IAE value of link2 during changes λ_2, (e) IAE value of link2 during changes μ_2, (f) IAE value of link2 during changes α_2.

4. Energy Efficiency:

ANNs: ANNs are computationally intensive and often require high power consumption due to continuous-valued operations and parallel processing. SNNs: SNNs have the potential to be more energy-efficient because they operate in an event-driven manner, where spikes are only generated when necessary. This can reduce power consumption in certain applications.

It is important to note that the choice between ANNs and SNNs depends on the specific task and requirements. While ANNs have been extensively used and studied, SNNs are gaining attention for their potential in modeling spatiotemporal information processing and achieving energy-efficient computing. Here are some examples of tasks where the temporal processing capabilities of Spiking Neural Networks (SNNs) would be crucial:

1- Spike-based pattern recognition: SNNs can effectively process temporal patterns and sequences, making them well-suited for tasks such as speech recognition, gesture recognition, classification, and event prediction (Jun Hu, K., T., L., & S., 2013; Kasabov, B., D., & W., 2023).

2-Spike-based learning algorithms: SNNs can learn and adapt to temporal patterns in data, enabling tasks such as online learning,

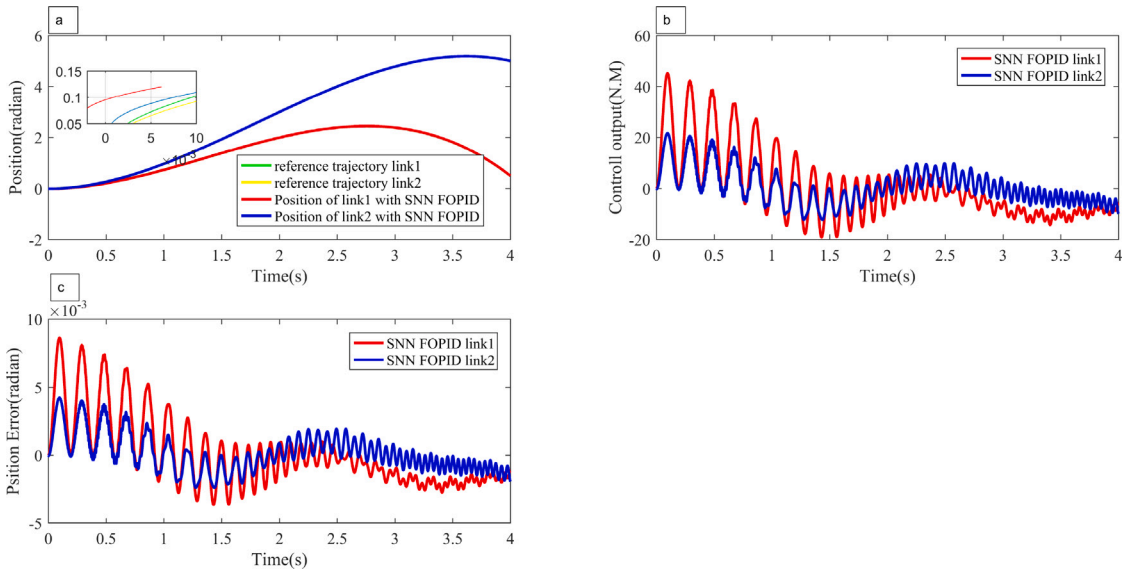


Fig. 10. (a) Trajectory tracking, (b) Control outputs, (c) Position error of $Link_1$ and $Link_2$ in the presence of no disturbance and noise for the SNNFOPID method.

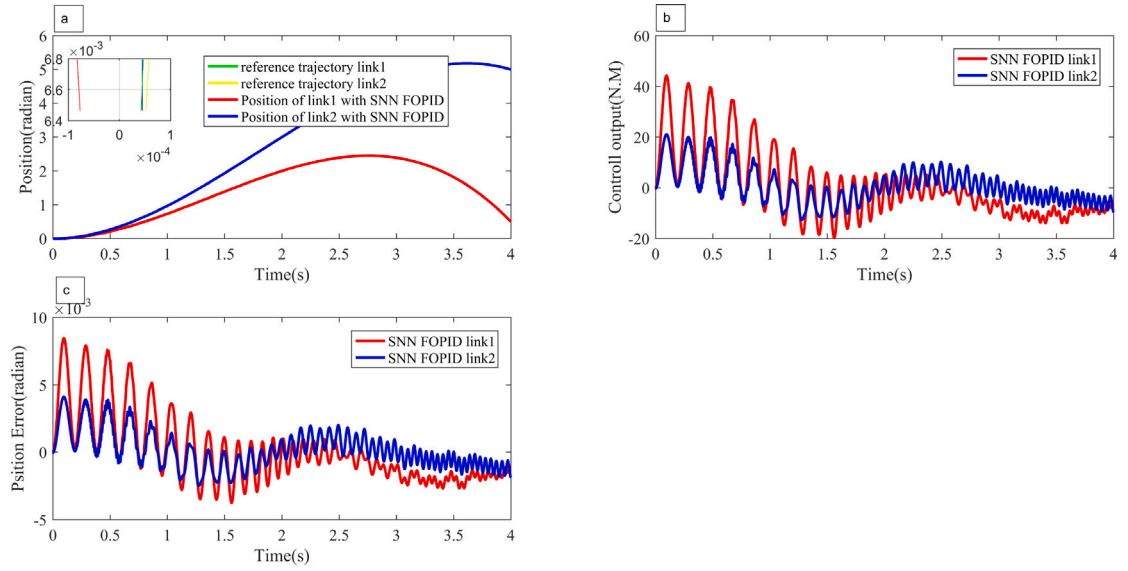


Fig. 11. (a) Trajectory tracking, (b) Control outputs, (c) Position error of $link_1$ and $link_2$ in the presence of disturbance $0.5 * sin(25t)Nm$ by the SNNFOPID method.

reinforcement learning, and unsupervised learning (Ning, D., X., T., & T., 2023; Zhang & L., 2021).

3- Neuromorphic computing: SNNs are often used in neuromorphic hardware for efficient and low-power implementation of cognitive tasks that require real-time processing of spatiotemporal data (Liu, Y., & C., 2020; Rathi, C., K., S., A., P., & R., 2023). Spiking neural networks can be integrated with other sensor modalities and perception systems in robots to enhance their autonomy in several ways:

(a) Multi-modal integration: Spiking neural networks can be used to integrate information from different sensor modalities, such as vision, touch, and sound, to provide a more comprehensive understanding of the robot's environment. This allows the robot to make more informed decisions based on a combination of sensory inputs.

(b) Sensor fusion: Spiking neural networks can be used to fuse information from multiple sensors to improve the accuracy and reliability of perception systems. For example, combining data from a camera and a LIDAR sensor can provide a more detailed and robust representation of the environment.

(c) Adaptive learning: Spiking neural networks can learn and adapt to new sensory inputs, allowing robots to continuously improve their perception capabilities over time. This adaptive learning process enables robots to better navigate complex and dynamic environments.

(d) Hierarchical processing: Spiking neural networks can be structured hierarchically to mimic the organization of the human brain. This allows robots to process sensory information at different levels of abstraction, enabling them to perform complex tasks that require

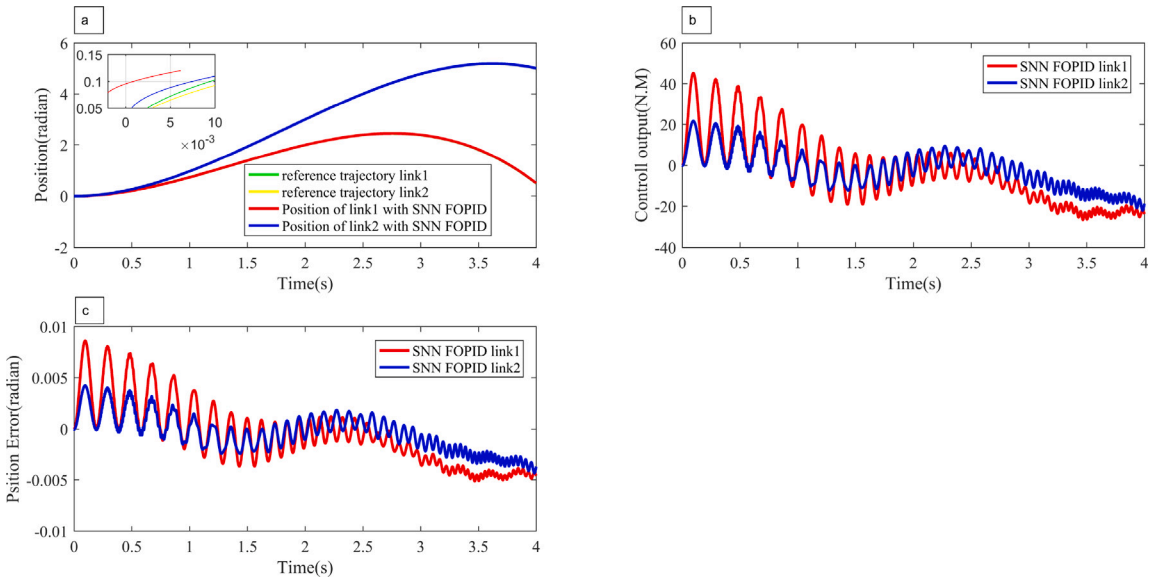


Fig. 12. (a) Trajectory tracking, (b) Control outputs, (c) Position error of $link_1$ and $link_2$ for different m_p values according to the third case of Table 7 by the SNNFOPID method.

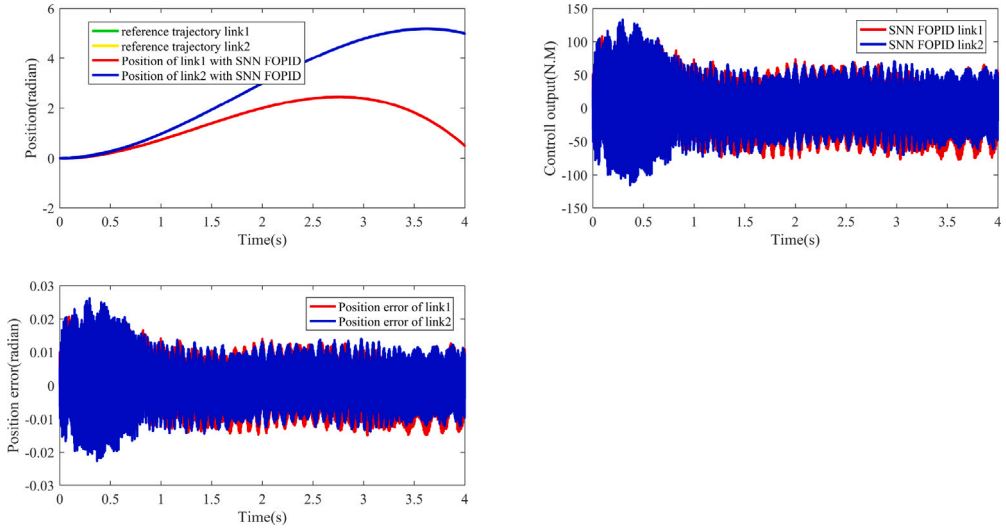


Fig. 13. (a) Trajectory tracking, (b) Control outputs, (c) Position error of $Link_1$ and $Link_2$ in the presence of noise for SNNFOPID method.

a combination of low-level sensor data and high-level cognitive reasoning. Overall, integrating spiking neural networks with other sensor modalities and perception systems can significantly enhance robot autonomy by improving their ability to perceive and interact with their environment more intelligently and adaptively.

3. SNNFOPID

Spiking neural networks, which are the third generation of artificial neural networks, are inspired by biological neural networks in the brain (Yan, Z., & W., 2021). They have high processing speed and temporal dynamics that enable them to efficiently encode and process information, mimicking the human brain's functionality. This unique feature allows the design of large-scale networks that set them apart from previous neural network models (Izhikevich, 2003). The Izhikevich model is used to simulate the spiking and bursting behavior of neurons. It combines the biological plausibility of the Hodgkin–Huxley model with the computational efficiency of the integrate-and-fire model. The Izhikevich model is known for its higher complexity, processing speed, and computational power compared to other spiking neuron models. The single-neuron differential equations of the Izhikevich model are given as Eqs. (1)–(3):

$$\frac{dv}{dt} = (0.04v^2 + 5v + 140 - u + I) \tag{1}$$

$$\frac{du}{dt} = a(bv - u) \tag{2}$$

$$if\, v >= 30mv(Millivolt), Then \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{3}$$

In which and represent the membrane potential and membrane recovery parameters, respectively. a, b and are dimensionless parameters and are set according to physiological data (Lee, P., G., & K., 2018). If the membrane potential is above 30, the motor neurons will spike (Valerio & J., 2010).

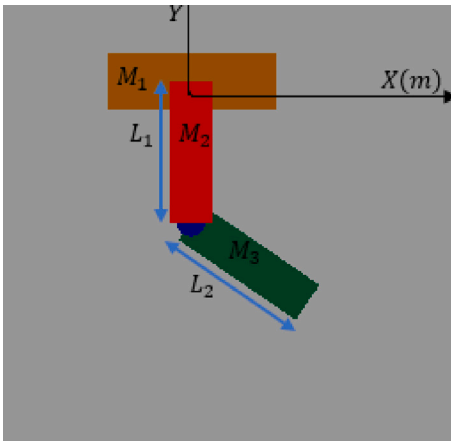


Fig. 14. Double inverted pendulum in SIMULINK.

3.1. FOPID structure

In Fractional Order PID (FOPID) controllers, the weighted sum of three actions regarding, I and elements are used to adjust the process via input. FOPID is preferable to the classic PID (Pritesh Shah, 2016):

1. FOPID will perform better than PID on higher-order systems (Das, S., & D., 2011; Shah & S., 2013).

2. FOPID provides better results on systems with long delays (Feliu-Batlle, R.-P., & C.-G., 2009; Das, D., & G., 2015).

3. For controlling nonlinear systems, FOPID performs better than PID (Das, S., & D., 2011).

4. A classical PID controller provides lower robust stability, whereas a Fractional Order PID controller has more robustness and stability (Mohammad Saleh Tavazoei, 2009; Petras, 2009).

The FOPID standard term is given by Eq. (4) (Xue & C., 2006):

$$C(s) = K_p + \frac{K_i}{S^{\lambda}} + K_d S^{\mu}, \quad (\lambda, \mu >= 0) \tag{4}$$

where C(s) is the transfer function, K_p is the proportional constant gain, K_i is the integration constant gain, K_d is the derivative constant gain, λ is the order of integrator, and is the order of differentiator (Fig. 1).

Generally, the range of fractional order is considered to be between (0,2). As it is shown in Fig. 1 (Hamamci, 2007):

- If $\lambda = 1$ and $\mu = 1$, then it is classical PID controller.
- If $\lambda = 0$ and $\mu = 1$, then it is classical PD controller.
- If $\lambda = 1$ and $\mu = 0$, then it is classical PI controller.
- If $\lambda = 0$ and $\mu = 0$, then it is classical P controller.

4. SNNs structure

Here, the SNNs are implemented through the Izhikevich model, and we incorporated the dynamic equation of a single neuron. The SNNs have two layers: the first layer is called the multi-input layer, and the other one is called the multi-output layer. The multi-input layer has neuronal groups to get data from the environment, and the multi-output layer is in contact with the environment by sending data to it. The key point is how they are connected. Each neuron in the multi-input layer is connected to the other neurons in the other layer. When the system starts to learn, the strength of connections between these neurons, which are called synaptic weights, change. The reinforcement learning mechanism of the networks is based on the integration of the spike-timing-dependent plasticity (STDP), and reward/punishment algorithm. Pure STDP is an unsupervised learning mechanism that adjusts synaptic weights based on temporal correlations between pre- and post-synaptic spike events (Hamamci, 2007). Here, reward/punishment-modulated STDP is used and the time-based formula for the dopamine-modulated STDP is as follows:

$$\begin{cases} I = S \\ \dot{S} = c.d \\ \dot{c} = -\frac{c}{\tau_c} + STDP(\tau)\delta(t - tpre/post) \\ \dot{d} = -\frac{d}{\tau_d} + DA \end{cases} \tag{5}$$

where I is the current passing through each synapse, S is the synaptic weight, τ is the decay parameter of $STDP$, C is the eligibility trace, d is the concentration of extracellular dopamine in the synaptic junction, and DA is the release of dopamine in neural networks $\delta(t)$ is the Dirac delta function that increases the variable C. Here we considered $a = 0.02$, $b = 0.2$, $c = -65$, $d = 8$ (Azimirad, M., S., V., & F., 2022; Azimirad, S., & N., 2021). This process has three types of currents: (i) Sensory (input) current: this current is given to a block of 50 input neurons each time we have input data. It activates a series of input neurons in the presence of the input data. (ii) Background current: it is randomly given to all the motor (output) neurons. (iii) Babbling current: it is given a block of 50 output neurons randomly each time we have input data. It activates a series of output neurons when input data are available.

The integration of these three currents guarantees proper learning. If only $STDP$ is used and dopamine is not released every time the correct action is performed, unsupervised learning will occur. According to some case studies, the speed of the learning process, when there is no reward and punishment, is low. Therefore, to increase the speed of the learning process, we decided to add reward and punishment, in which the mechanism of releasing dopamine (DA) is added to the STDP system. We considered $DA = 1$ (positive dopamine or reward) for each correct action and $DA = -1$ (negative dopamine or punishment) for each incorrect action. There are numerous methods for regulating dopamine levels, however, the focus of this research is to explore the biological function of spiking neural networks in order to achieve this objective.

5. SNNFOPID flowchart

To optimize the FOPID parameters and produce the hybrid learning control, SNNs are employed due to their impressive energy-saving capabilities. The network comprises various neuron groups, and the connections between these neurons exhibit variable synaptic strengths. Moreover, there is a system of reward and punishment: correct actions are rewarded, whereas incorrect actions are penalized, so altering the synaptic weight. The structure of this flowchart is illustrated in Fig. 2. As the network adapts and learns to achieve a minimal Integral Absolute Error (IAE), the dopamine dose is balanced towards zero, with both positive and negative feedback being equally considered.

Based on Figs. 2, 3, and 5:

1- A distinct Spiking Neural Network (SNN) is specifically crafted for each robot link based on the number of degrees of freedom of the robot. These networks consist of six types of input or sensor neurons and one type of output or motor neuron.

2- The neuronal spiking behavior in these networks is governed by Spike-Timing-Dependent Plasticity (STDP), also known as the local synapse role. This principle is inspired by the biological concept that input neurons consistently precede output neurons in firing patterns.

3- Each set of input and output neurons comprises 50 units, mirroring the specialized functionality observed in the brain and mammalian nervous systems(a set of neurons act for a special purpose). To optimize controller parameters, 50 input neurons establish synaptic connections with 50 output neurons, resulting in a total of 2500 synaptic weights. The average of these synaptic weights directly influences the desired controller parameter value.

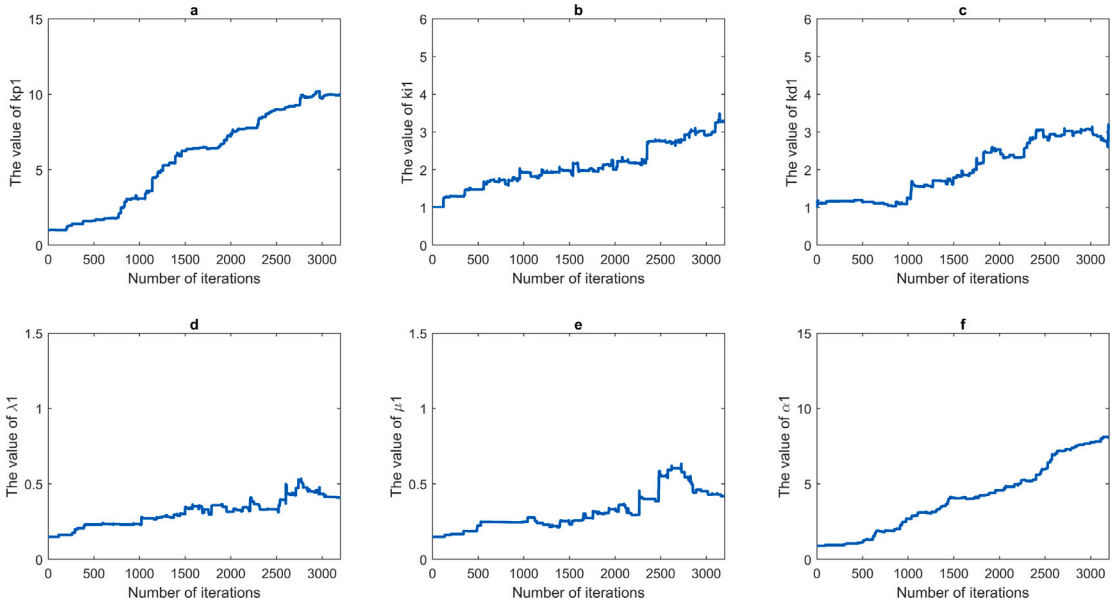


Fig. 15. $(a)\ kp1, (b)\ ki1, (c)\ kd1, (d)\ \lambda 1, (e)\ \mu 1, (f)\ \alpha 1$.

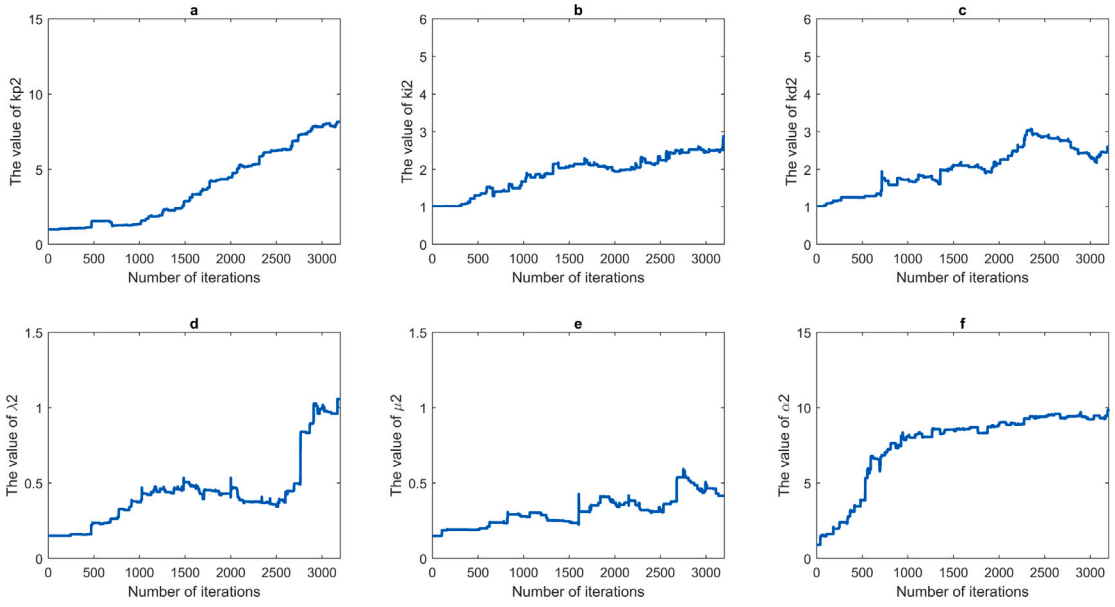


Fig. 16. $(a)\ kp2, (b)\ ki2, (c)\ kd2, (d)\ \lambda 2, (e)\ \mu 2, (f)\ \alpha 2$.

4- The network architecture is designed to introduce current randomly to a specific group of neurons during each iteration. Consequently, only the synaptic weights corresponding to that particular group of parameters are modified at any given time. These adjusted values are then used to determine the robot controller's performance, as indicated by the Integral of Absolute Error (IAE) value for the desired joint in the Spiking Neural Network (Eq. (6)).

$$Integral Absolute Error(IAE) = \int \mid e(t) \mid dt \tag{6}$$

5- When synaptic weight changes are tailored to the desired parameter type to optimize evaluation criteria, such as Integral Absolute Error (IAE) focusing on a specific robot link for correct actions, positive dopamine is introduced into these synapses by the network. This amount of dopamine accumulates with synaptic weights related to the desired parameter, enhancing its value. Conversely, if altering the synaptic weight of the desired parameter leads to increased IAE, signaling an incorrect action, negative dopamine linked to the desired parameter is released to decrease its value. ($\mid e(t) \mid$ is the absolute error). For each neuron, we consider $IAE_{\alpha}\ IAE_{\mu}, IAE_{\lambda}, IAE_{p}, IAE_{i} and\ IAE_{d}$. The activity of each neuron was compared with its previous activity.

6- The optimal synaptic weight values for μ and λ parameters in each robot link controller, crucial for system stability as per

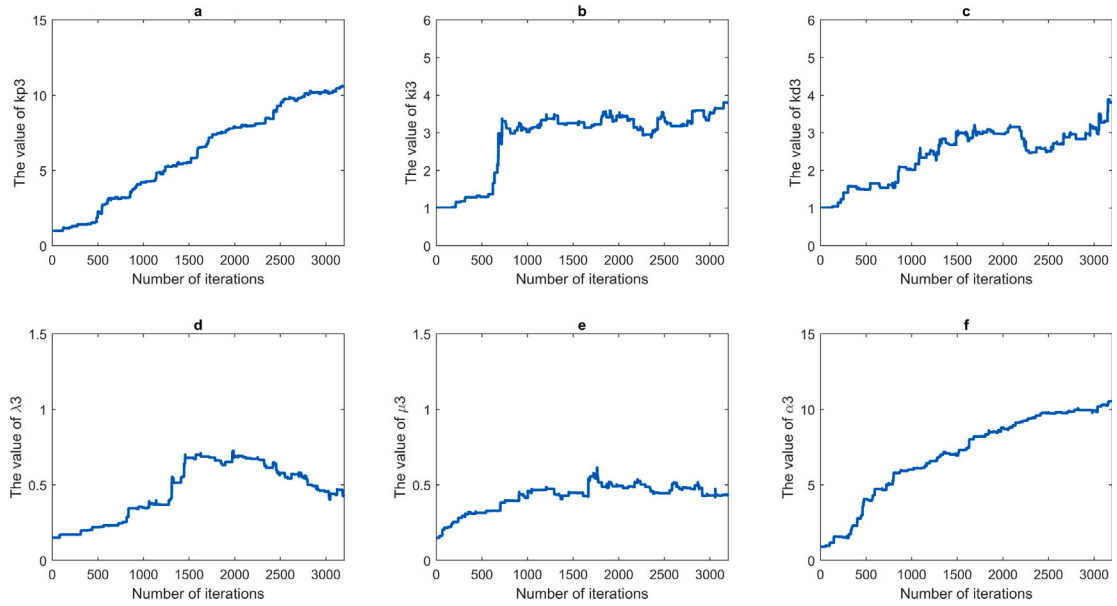


Fig. 17. $(a)\ kp3, (b)\ ki3, (c)\ kd3, (d)\ \lambda 3, (e)\ \mu 3, (f)\ \alpha 3$.

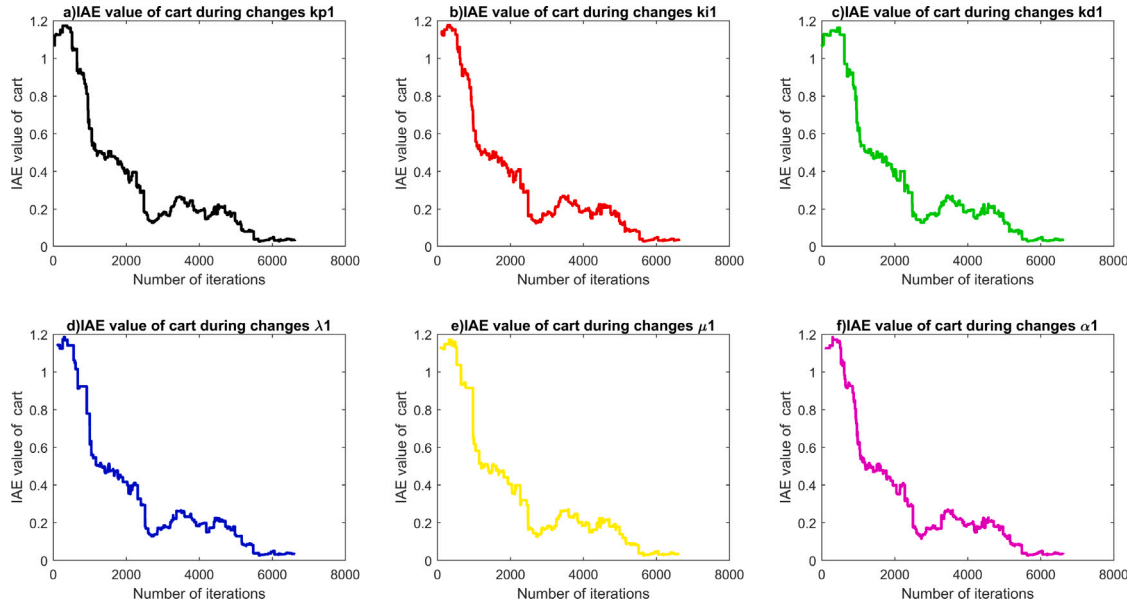


Fig. 18. (a) IAE value of cart during changes kp1, (b) IAE value of cart during changes ki1, (c) IAE value of cart during changes kd1, (d) IAE value of cart during changes λ_1, (e) IAE value of cart during changes μ_1, (f) IAE value of cart during changes α_1.

Ref. Iakymchuk, R.-M., F.G.-M., B.-M., V., and V. (2015), Pritesh Shah (2016), fall within the range of 0 to 2, with a recommended maximum of Sm = 2.

7- While parameters like kp, ki, kd, and α are not constrained by specific value limits, ensuring all synaptic weights in the Spiking Neural Networks (SNNs) reach a maximum value is essential for network convergence and learning. To facilitate this process, an initial arbitrary value Sm is assigned to these parameters, with the network updating it to $Sm_{new} = Sm_{old} + 5$, if values exceed $0.8 * Sm$ during learning. 8- The coefficient α serves as a multiplier in the controller of each robot link, strategically incorporated to expedite the network's convergence speed amidst the time-intensive reinforcement learning process, proving effective in practice.

9- This iterative process persists until achieving the lowest possible IAE values for each robot link, facilitating parameter convergence and optimizing system performance. The pseudo-code presented in Fig. 3 provides a comprehensive and clear representation of the functioning of SNNFOPID. Fig. 4 shows the schematic of n-degrees of freedom serial manipulator which is used to test the SNNFOPID method.

5.1. Application of the SNNFOPID algorithm in controlling n-degrees of freedom robot

Suppose that there is an n-degrees of freedom (n-DOF) robot (Fig. 3). To control it, the following algorithm is used for every joint:

$$Ui = \alpha_i(k_{Pi}e(t) + K_{Ii}D^{-\lambda_i}e(t) + K_{D}iD_i^{\mu}e(t)) \tag{7}$$

in which, U is the control signal, $e(t)$ shows error, α is the gain for changing the speed of convergence, and $i = 1, 2, ..n$. For each joint, a separate control system is considered and each control system will be designed through a separate spiking neural network. Fig. 5 shows how

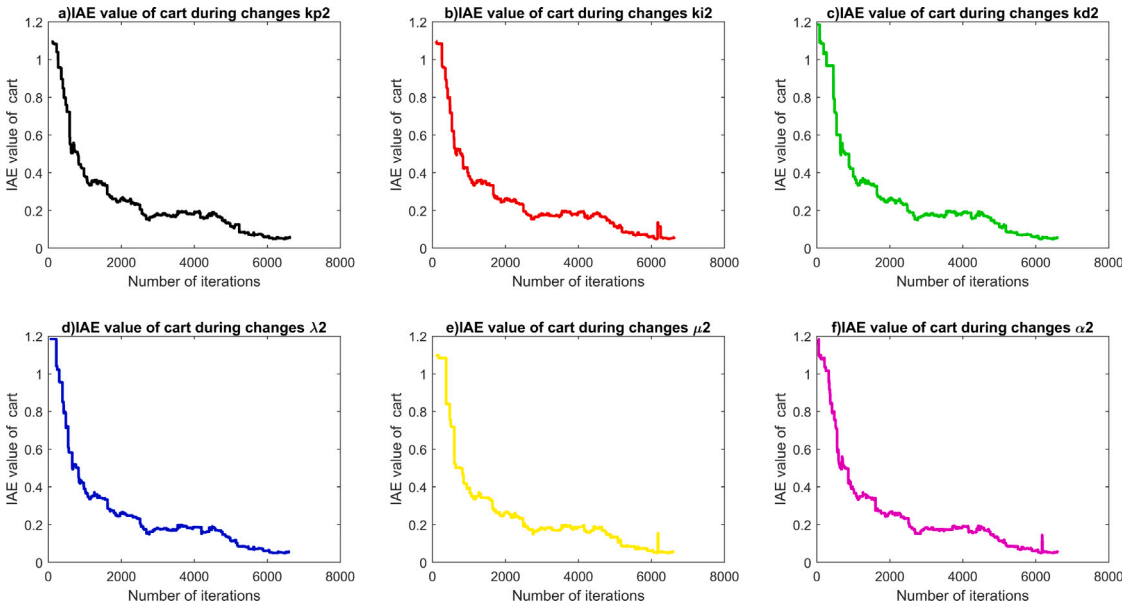


Fig. 19. (a) IAE value of link1 during changes kp2, (b) IAE value of link1 during changes ki2, (c) IAE value of link1 during changes kd1, (d) IAE value of link1 during changes λ_2, (e) IAE value of link1 during changes μ_2, (f) IAE value of link1 during changes α_2.

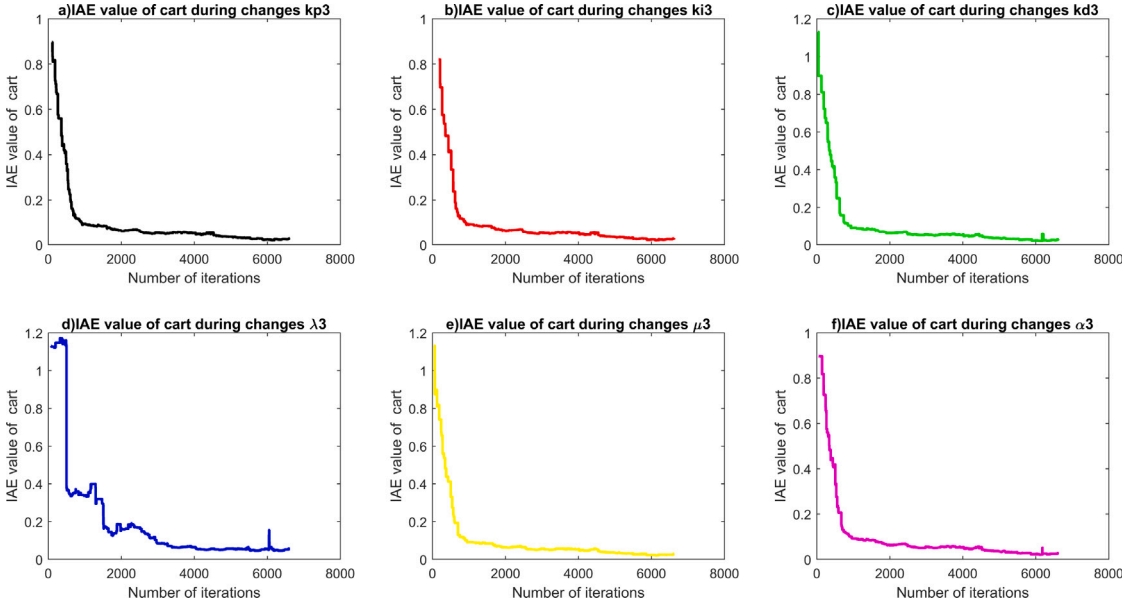


Fig. 20. (a) IAE value of link2 during changes kp3, (b) IAE value of link2 during changes ki3, (c) IAE value of link2 during changes kd3, (d) IAE value of link2 during changes λ_3, (e) IAE value of link2 during changes μ_3, (f) IAE value of link2 during changes α_3.

the SNN-based learning mechanism connects with the FOPID controller to control an n-DOF robotic system.

According to Fig. 5, in every iteration, the FOPID parameters are generated and passed to the control system by the reinforcement learning system, and the IAE is returned to the SNNs. The current IAE value is then compared to the previous one. If the amount of IAE decreases, positive dopamine is released ($DA = +1$), and if the IAE increases, negative dopamine is injected into the activated synapse ($DA = -1$). This process continues until the IAE value of each joint reaches its minimum value. To prove the proper performance of the new control system, we tested it on a robot with two degrees of freedom. The full dynamic model of a two-DOF robot is considered in the case study. The nonlinear dynamic equations of the simulated two-DOF robot are shown in the appendix (Lee & Lee, 2003). The values of the model parameters are shown in Table 1.

Table 1
Dynamic parameters of the simulated robot.

Parameter	$Link_1$	$Link_2$
Mass (kg)	1	1
Length (m)	1	1
$g(m/s^2)$	9.81	9.81
length from the joint to its center of gravity (m)	0.5	0.5
Lengthwise centroid inertia of link (kg m2)	0.2	0.2
Coefficient of viscous friction	0.1	0.1
Coefficient of dynamic friction	0.1	0.1

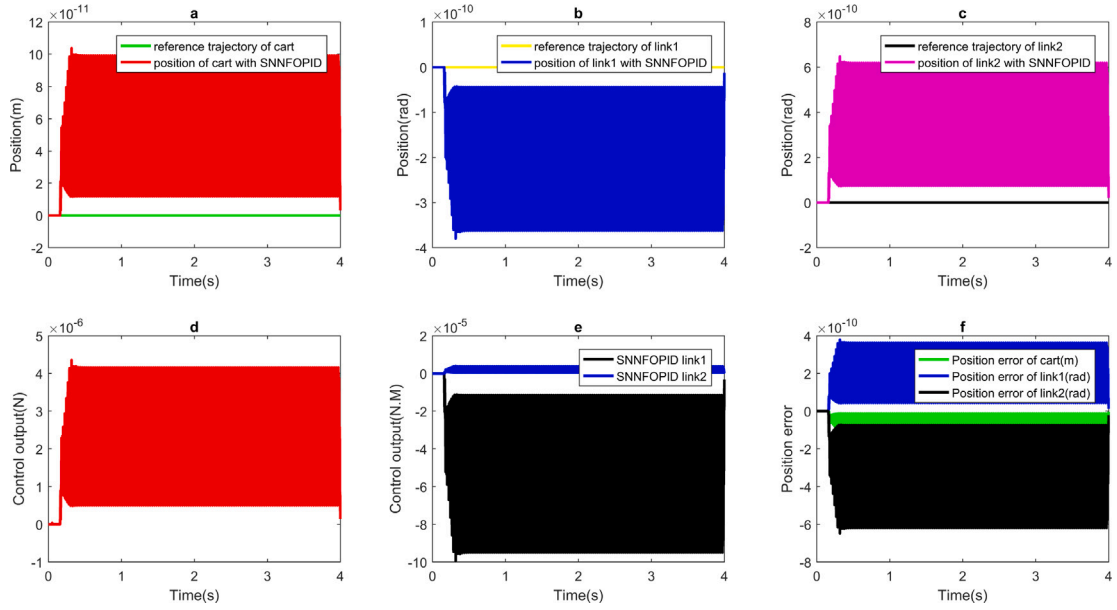


Fig. 21. (a) Trajectory tracking of cart, (b) Trajectory tracking of link1, (c) Trajectory tracking of link2, (d) Control output of cart (N), (e) Control output of link1, and link2 (N m), (f) Position error of cart, $Link_1$ and $Link_2$.

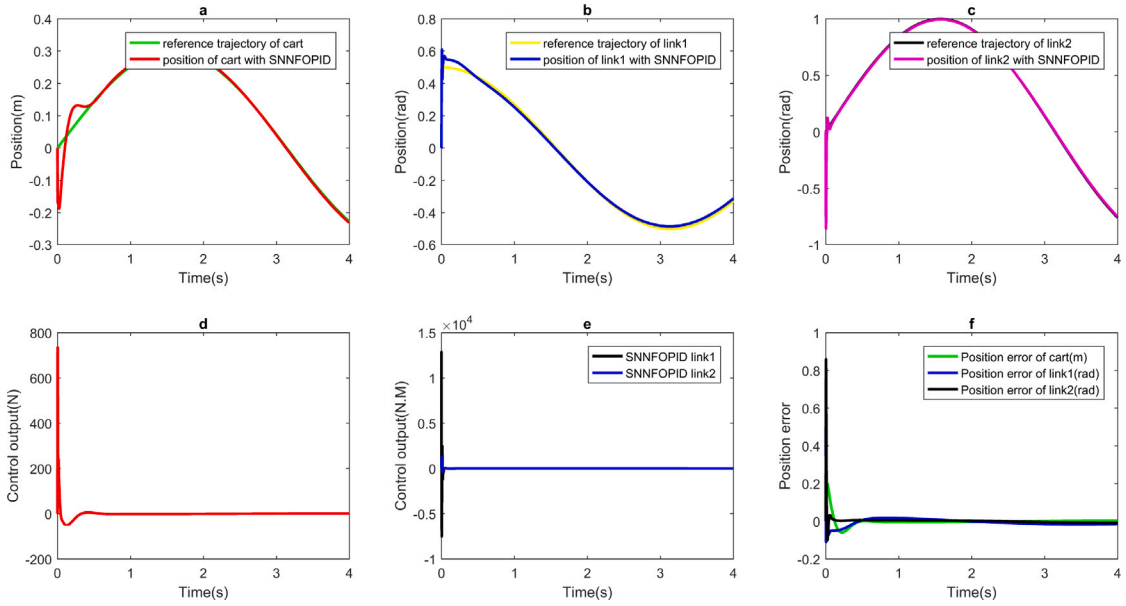


Fig. 22. (a) Trajectory tracking of cart, (b) Trajectory tracking of link1, (c) Trajectory tracking of link2, (d) Control output of cart (N), (e) Control output of link1, and link2 (N m), (f) Position error of cart, $Link_1$ and $Link_2$ in the case of no disturbance.

To validate the results, the reference trajectories for joints are taken into account the same as the ones in Iakymchuk et al. (2015). So for the first joint, the reference trajectory is considered as Eq. (8), and for the second joint the reference trajectory is given as Eq. (9) (Richa Sharma & M., 2015):

$$Reference\ trajectory_1 = 0.96875 * t^2 - 0.234375 * t^3 \tag{8}$$

$$Reference\ trajectory_2 = 1.1875 * t^2 - 0.21875 * t^3 \tag{9}$$

in which, t is the time of simulation (t = 0:4 s). The values of the FOPID control parameters obtained from the SNNFOPID algorithm (for the two-DOF robot) are given in Table 2.

In all cases, we have compared the results of the current work with the results of implementing FOPID method (Das et al., 2011) under identical conditions. The standard deviations (SD) of the synaptic weights for each class of 50 neurons are listed in Tables 3 and 4.

Tables 3 and 4 show that considering the average amount of synaptic weights as the weight of each class of neurons could be done with a good approximation (because of the low Standard deviation according to Richa Sharma and M. (2015)). Neurons perform a specific action in groups, and to imitate this point of view, we assigned 50 neurons to adjust each parameter, which shows the average synaptic weight of each group of 50 neurons. The standard deviation provides a good indication of how close the actions of each class of neurons are to each other. According to the standard deviation values of each class, the average weight of each of the 50 neurons is considered the value of each control parameter. Figs. 6 and 7 show the value of the controller parameters during the training process.

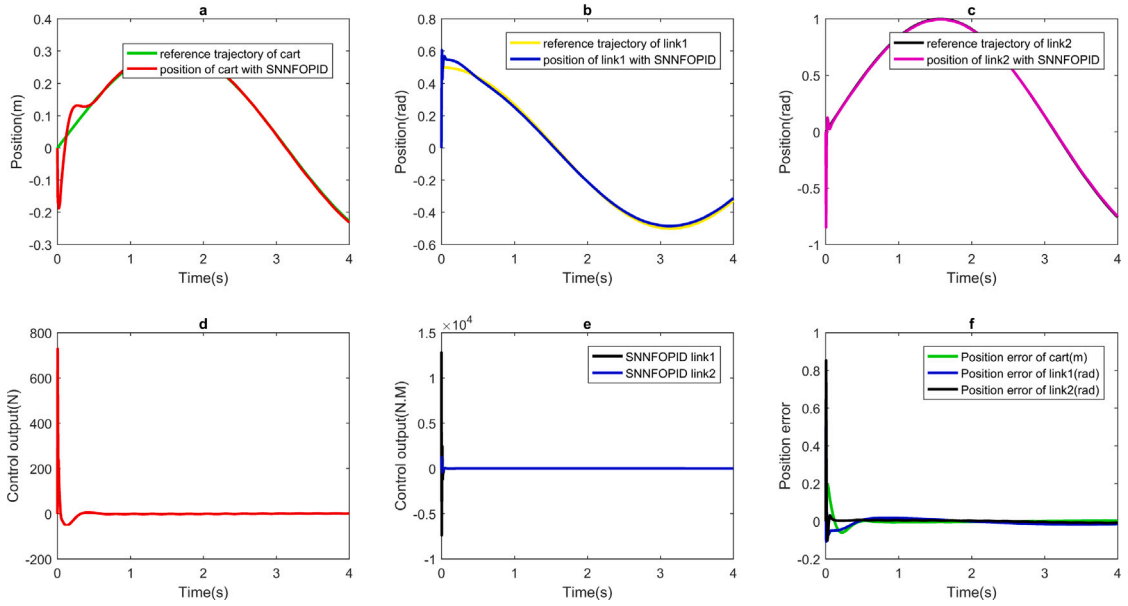


Fig. 23. (a) Trajectory tracking of cart, (b) Trajectory tracking of link1, (c) Trajectory tracking of link2, (d) Control output of cart (N), (e) Control output of link1, and link2 (N m), (f) Position error of cart, $Link_1$ and $Link_2$ in the presence of disturbance.

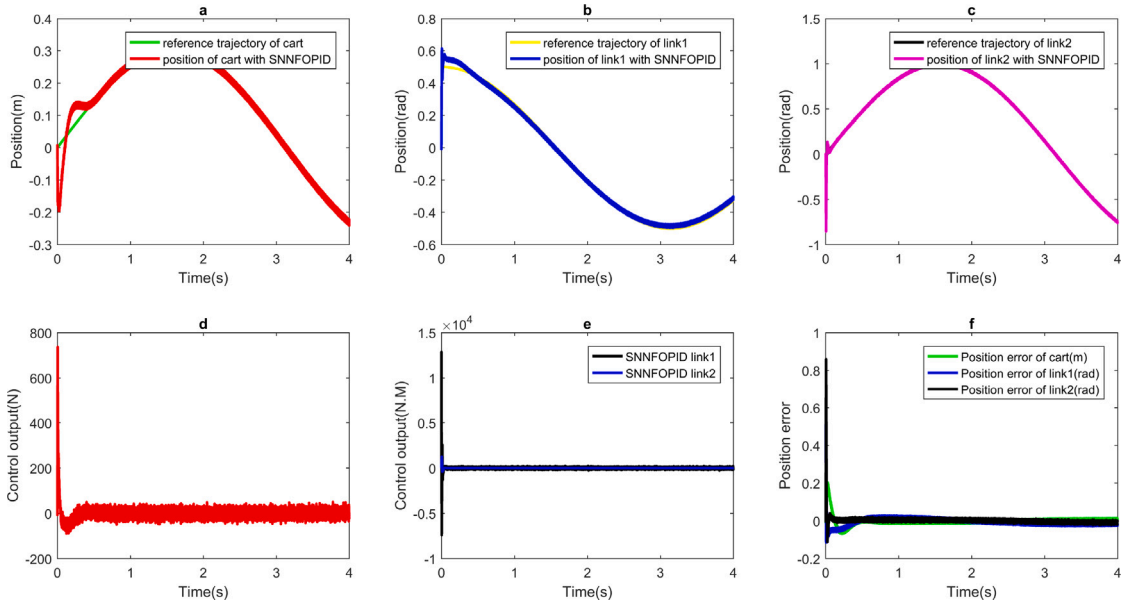


Fig. 24. (a) Trajectory tracking of cart, (b) Trajectory tracking of $Link_1$, (c) Trajectory tracking of $Link_2$, (d) Control output of cart (N), (e) Control output of $Link_1$, and $Link_2$ (N m), (f) Position error of cart, $Link_1$ and $Link_2$ in the presence of noise.

Table 2
The final values of SNNFOPID obtained after learning for the two-DOF robot.

K_{p1}	K_{i1}	K_{d1}	μ_1	λ_1	α_1	K_{p2}	K_{i2}	K_{d2}	μ_2	λ_2	α_2
192.0686	1.7968	2.1729	0.4491	0.4644	26.4431	192.8363	1.4489	1.4206	0.1885	0.2958	26.1481

As shown in Figs. 6 and 7, the values of k_{p1} and k_{p2} after 30,000 iterations reached values close to 190 and became stable, and α_1 and α_2 reached constant values of 26 in 10,000 iterations and then remained in this constant range. k_{d1}, k_{d2}, k_{i1} and k_{i2} fluctuated in the range of 1.7 to 2 after 3000 iterations. As expected, in the range below 2, μ_1, μ_2, λ_1 and λ_2 have reached constant values. When all the parameters become stable, the algorithm is stopped. The purpose of adding α_i into the control system is to reduce the number of iterations. This signifies the network's ability to learn and maintain stability over time. Figs. 8 and 9 show the minimization of IAE for links 1 and 2 during the changes in parameters by the SNNFOPID algorithm in different iterations.

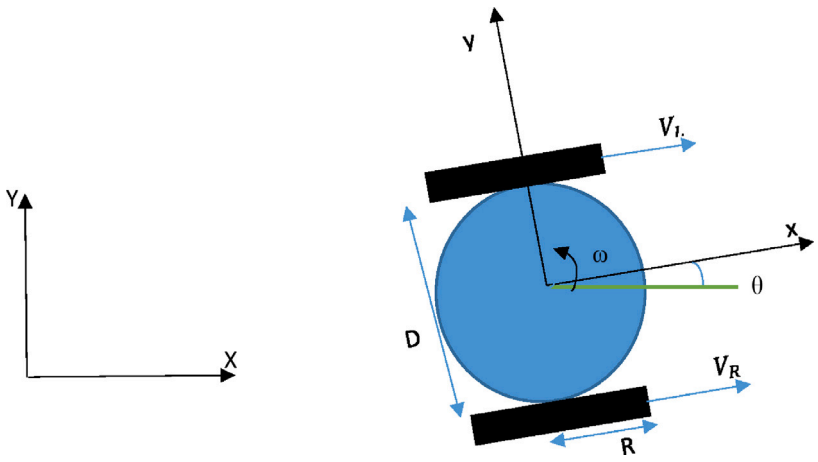


Fig. 25. Two-wheeled mobile robot.

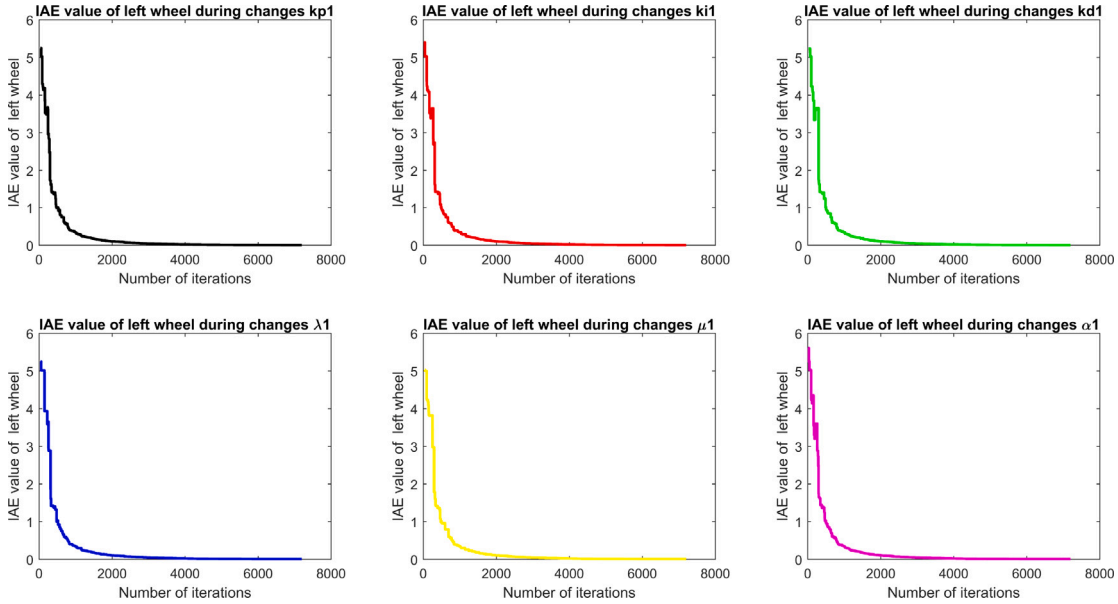


Fig. 26. (a) IAE value of left wheel during changes kp1, (b) IAE value of left wheel during changes ki1, (c) IAE value of left wheel during changes kd1, (d) IAE value of left wheel during changes λ_1, (e) IAE value of left wheel during changes μ_1, (f) IAE value of left wheel during changes α_1.

Table 3
Standard deviation (SD) of the synaptic weights of each class of FOPID Parameters for $link_1$.

STD_{Kp1}	STD_{Ki1}	STD_{Kd1}	$STD_{\mu 1}$	$STD_{\lambda 1}$	$STD_{\alpha 1}$
2.6565	0.1299	0.181	0.0789	0.0678	0.67

Table 4
Standard deviation (SD) of the synaptic weights of each class of FOPID Parameters for $link_2$.

STD_{Kp2}	STD_{Ki2}	STD_{Kd2}	$STD_{\mu 2}$	$STD_{\lambda 2}$	$STD_{\alpha 2}$
2.4937	0.1123	0.1081	0.06544	0.06957	0.7323

Based on the analysis of Figs. 8 and 9, it is evident that the IAE value is influenced by changes in the control parameters during the SNN learning process. As the learning process progresses, we observe a significant reduction in the IAE value, approaching zero, indicating the effective functioning of the dopamine system in guiding the network towards the desired target. The trajectory, control outputs, and position error of variables $Link_1$ and $Link_2$ resulting from the SNNFOPID operation without disturbance are shown in Fig. 10.

As shown in Fig. 10, when there is no disturbance and noise, the position error values of $link_1$ and $Link_2$ were less than 0.007 rad and 0.005 rad, respectively. The ranges of control outputs for $link_1$ and $Link_2$ are $(-20,45)$ Nm and $(-20,25)$ Nm, respectively.Given the desired path assigned to each link of the robot, the robot is trained to track this path accurately. Fig. 10 (c) illustrates the robot's precise learning along the desired paths. The position error for both robot links is less than 0.007 radians. To study the performance of the new SNNFOPID method, some simulation studies are done and the integral absolute error of the robot links in the presence of friction is considered (Lee & Lee, 2003). The new method's performance has been investigated for three different coefficients of viscous friction and results are compared with Sharma FOPID methods in Table 5 (Richa Sharma & M., 2015).

According to Table 5, in the presence of friction for links 1 and 2, the IAE value in the SNNFOPID method is lower than the IAE value in the Sharma method (Richa Sharma & M., 2015), and results show that

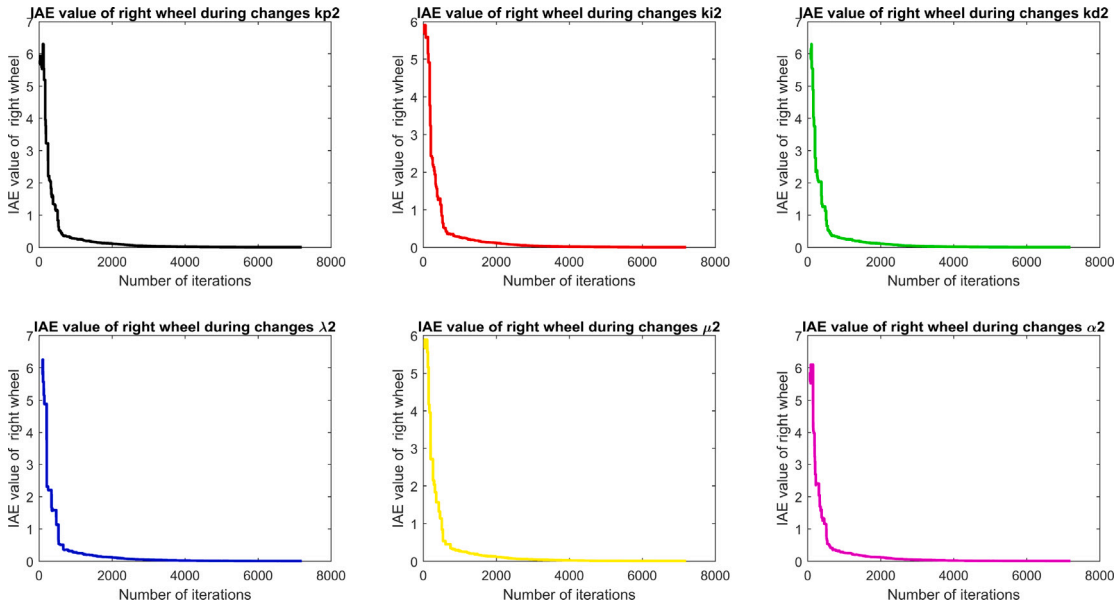


Fig. 27. (a) IAE value of right wheel during changes kp2, (b) IAE value of right wheel during changes ki2, (c) IAE value of right wheel during changes kd2, (d) IAE value of right wheel during changes λ_2, (e) IAE right wheel during changes μ_2, (f) IAE value of right wheel during changes α_2.

Table 5
IAE values of SNNFOPID for $Link_1$ and $Link_2$ in three different functions of viscous friction (v).

Parameter variation	Sharma FOPID Petrovic, V., I., M., and S. (2016)		SNNFOPID	
	$Link_1$	$link_2$	$Link_1$	$link_2$
case1 $v_1 = 0.5\ Sgn(\dot{\theta}_1)$ $v_2 = 0.1$	0.008775	0.01496	0.007665	0.004036
case2 $v_1 = 0.1$ $v_2 = 0.5\ Sgn(\dot{\theta}_2)$	0.008788	0.01418	0.007651	0.00406
case3 $v_1 = 0.5\ Sgn(\dot{\theta}_1)$ $v_2 = 0.5\ Sgn(\dot{\theta}_2)$	0.008775	0.01418	0.007631	0.004124

SNNFOPID acts better when there is viscous friction in joints of the robot. To evaluate the performance of the new SNNFOPID controller, a series of simulation studies are done in the presence of disturbance. In the first case study, ten different values of the disturbance are considered for the first link of the robot and there is no disturbance on the second link of the robot. In the second case study, ten different amounts of the maximum disturbance are considered for the second link and the disturbance on the first link is zero. In the third series of case studies, the performance of the system is studied in the presence of disturbances for both links of the robot. In each one, ten different values of sinusoidal disturbance are applied to the robot inputs, and the IAE values for each case and the value of the sinusoidal disturbance are presented for SNNFOPID and Sharma method (Richa Sharma & M., 2015). Results are shown in Table 6.

According to Table 6, in all three cases, SNNFOPID performs better when the disturbance is applied. As an example, the trajectory tracking, control outputs, and position error of $link_1$ and $link_2$ resulting from the SNNFOPID operation in the presence of disturbance $(0.5 * sin(25t)Nm)$ are shown in Fig. 11.

As shown in Fig. 9, in the presence of disturbance with a maximum value of 0.5 Nm, the error value for each link of the robot is less than 0.009 rad and the error for the first link is higher than the error for the second link. In addition, all error values converge to zero after about 3 s. In another case study, the carrying payload mass (m_p) of the robot is considered to have an altering value concerning time, and the performances of SNNFOPID and Sharma FOPID (Petrovic et al., 2016) are compared. Hence, m_p is considered to have three different values, according to Table 7, and for these three payload amounts, the IAE values are obtained (Table 7).

Table 7 shows that in all three cases, SNNFOPID has a lower IAE value for both robot links. Fig. 10 shows the results of the third case of Table 7 in terms of the position error and control outputs.

As shown in Fig. 12, for an altering value of m_p, the amount of error for each link of the robot is less than 0.009 rad and the error for the first link is higher than the error for the second link. Finally, we studied the performance of the new method when random noise is introduced into the system. In a series of case studies, the amount of $-0.01 \leq t \leq 0.01\ rad$ is added to the output of the robot as noise, and the IAE values for each case are obtained. In the first case study, the noise is only applied to the output of the first link, in the second case study, the noise is only applied to the output of the second link, and in the third case study, the noise is applied to the outputs of both links. The Integral Absolute Errors of $Link_1$ and $Link_2$ for these case studies are presented in Table 8.

According to the results of adding noise to the system output in Table 8, SNNFOPID in case 1 and case 3 has a lower IAE value for both links and in Sharma method for case 2, the IAE value is lower than the SNNFOPID method. For illustration, the trajectories, control inputs, and position error of adding noise to both links are shown in Fig. 13.

As shown in Fig. 11, in the presence of random noise $-0.01 \leq t \leq 0.01\ rad$, the error value for each robot link is less than 0.03 rad. Results in Figs. 12 and 13 show the robustness of the proposed system in the presence of disturbance and noise.

Table 6
IAE values of $Link_1$ *and* $Link_2$ for SNNFOPID method in presence of disturbances.

Disturbances (Nm)	Sharma FOPID in Petrovic et al. (2016)		SNN FOPID	
	$Link_1$	$Link_2$	$Link_1$	$Link_2$
$Link_1$				
0.1 sin(25t)	0.008784	0.01496	0.007631	0.004132
0.2 sin(25t)	0.008783	0.01496	0.007627	0.004153
0.3 sin(25t)	0.008783	0.01496	0.00724	0.004178
0.4 sin(25t)	0.008787	0.01496	0.007628	0.004172
0.5 sin(25t)	0.008797	0.01496	0.007636	0.004148
0.6 sin(25t)	0.008810	0.01496	0.007646	0.004125
0.7 sin(25t)	0.008825	0.01496	0.007658	0.0041
0.8 sin(25t)	0.008842	0.01496	0.007664	0.004098
0.9 sin(25t)	0.008860	0.01496	0.00767	0.004097
sin(25t)	0.008890	0.01496	0.007679	0.00409
$Link_2$				
0.1 sin(25t)	0.008791	0.01495	0.007629	0.00414
0.2 sin(25t)	0.008796	0.01494	0.007622	0.004177
0.3 sin(25t)	0.0088	0.01493	0.007627	0.004151
0.4 sin(25t)	0.008804	0.01492	0.007633	0.004134
0.5 sin(25t)	0.008808	0.01492	0.007633	0.004137
0.6 sin(25t)	0.008813	0.01491	0.007628	0.004164
0.7 sin(25t)	0.008817	0.01491	0.007626	0.004178
0.8 sin(25t)	0.008821	0.0149	0.007631	0.004164
0.9 sin(25t)	0.008826	0.0149	0.007639	0.004147
sin(25t)	0.008830	0.0149	0.007638	0.004159
Both Link				
0.1 sin(25t)	0.008788	0.01496	0.007634	0.004124
0.2 sin(25t)	0.00879	0.01495	0.007635	0.004126
0.3 sin(25t)	0.008793	0.01494	0.007637	0.004137
0.4 sin(25t)	0.008797	0.01493	0.007636	0.00416
0.5 sin(25t)	0.008802	0.01492	0.007643	0.004155
0.6 sin(25t)	0.008809	0.01492	0.007654	0.004139
0.7 sin(25t)	0.008815	0.01491	0.007667	0.004117
0.8 sin(25t)	0.008823	0.01491	0.007681	0.004116
0.9 sin(25t)	0.008832	0.0149	0.007692	0.00412
sin(25t)	0.008843	0.0149	0.007701	0.004126

5.2. Application of SNNFOPID to control a double inverted pendulum

To provide additional evidence of the effectiveness of the SNNFOPID method, we have extended its application to a double inverted pendulum, which incorporates an arm and a mobile platform. The three-degree-of-freedom robot, as depicted in Fig. 14, has been meticulously designed within the SIMSCAPE section of SIMULINK, MATLAB.

Table 9 presents the parameters of this robot:

Considering that this robot has three degrees of freedom, according to the SNNFOPID method, we will have three SNN networks, that is, three types of FOPID controllers. All steps of network learning are similar to the two-link robot. Tables 10, 11, and 12 show the value of controller parameters for each degree of freedom:

Figs. 15, 16, and 17 give the values of each parameter during the learning process for the three degrees of freedom of the robot.

According to Figs. 15, 16, and 17, almost after 6000 iterations, the parameters reach a stable range. Also, Figs. 18, 19, and 20 present SNNFOPID performance during SNN learning to minimize IAE in all three robot links, respectively.

In the case of the equilibrium state, the IAE value for the cart, link1, and link2 of the double inverted pendulum is shown in Table 13.

Fig. 21 shows the path tracking, controller output, and position error for the cart, links 1 and 2 of the robot for the equilibrium state, respectively.

Based on the data presented in Fig. 21, it is evident that the output values of the controllers and the error values for all three links are exceptionally low. This implies that the energy required to maintain the balance of this robot is significantly minimal.

We also conducted a professional evaluation of the performance of controllers for the double-inverted pendulum when subjected to variable inputs. The equations representing these inputs are as follows:
Reference trajectory of cart = 0.3 sin(t)
Reference trajectory of $link_1$ = 0.5 sin(t+ $\pi/2$)
Reference trajectory of $link_2$ = sin(t)
In the case of no disturbance, the IAE value for the cart, $link_1$, and $link_2$ of the double inverted pendulum is shown in Table 14. Fig. 22 presents the Trajectory tracking, controller output, and position error for the cart, links 1, and 2 of the robot in the presence of disturbance in the system, respectively.

In the other cause, some disturbance has been applied to all the joints of the robot: Disturbance = 0.5 sin(25t).

Table 15 gives us the value of IAE of the cart, and links of the robot in the presence of disturbances in the robotic system.

In Fig. 23, the trajectory tracking, controller output, and position error can be well displayed for the cart, links 1 and 2 of the double inverted pendulum, respectively. In the last case, the amount of noise $-0.01 \leq t \leq 0.01$ rad radians is given to the cart, and joints of the double inverted pendulum. Table 16 shows the IAE value of each link. Fig. 24 shows the trajectory tracking, controller output, and position error for the cart, $Link_1$, and $Link_2$ of the double inverted pendulum in the presence of noise, respectively.

According to Figs. 22, 23, 24, and 25, we reached acceptable results for controlling the double inverted pendulum. which brings the error value to less than 0.03 radians.

5.3. Application of SNNFOPID for two-wheeled mobile robot

The performance of the SNNFOPID has been evaluated in another case study involving a mobile robot, as discussed in the article authored by Petrovich et al. in 2016 (Petrovic et al., 2016). Here, the FOPID controller is employed to effectively control the x and y positions of the robot. Fig. 25 displays a comprehensive view of the two-wheeled mobile robot. The kinematic relationship is derived between the posture vector p, expressed in the X-Y coordinate system, and the corresponding velocity vector. velocities vector is derived as:

$$V_L = R\omega_L \ , V_R = R\omega_R \ , \dot{x} = V COS(\theta) \ , \dot{y} = V sin(\theta) \ ,$$
$$\dot{\theta} = \omega \ , w = \frac{V_L - V_R}{D} \ , \dot{P} = \begin{bmatrix} COS(\frac{1}{2}\theta) & COS(\frac{1}{2}\theta) \\ sin(\frac{1}{2}\theta) & sin(\frac{1}{2}\theta) \\ \frac{1}{D} & \frac{-1}{D} \end{bmatrix} \quad (10)$$

where V_L is the linear speed of the left wheel, V_R is the linear speed of the right wheel, Ω is the angular speed of the robot, R = 0.2 m is the radius of the wheels, D = 0.1 m, the distance between the two wheels, and θ is defined as the angle between the X-coordinate and the heading direction. Figs. 26 and 27 demonstrate the minimization of the IAE value in both wheels' positions throughout the learning process.

Based on data from Figs. 26 and 27, it can be observed that the value of IAE has reached its minimum value. Figs. 28 and 29 depict the controller parameter values throughout the learning process.

The parameters achieve stability following during the final 100 iterations. Tables 17 and 18 show the value of controller parameters for each degree of freedom:

The performance of the controllers has been assessed in three cases: no disturbance, in presence of disturbance, and noise. The amount of disturbance applied to both wheels of the robot is equal: 0.5 sin(25t), and the range of applied $-0.001 \leq t \leq 0.001$. Additionally, the IAE value for each case is provided in Tables 19–21.

According to Figs. 30, 31, and 32, the controlled trajectory tracking in the x and y directions with control inputs V_R and V_L shows that position errors are less than 0.01 m.

5.4. Application of SNNFOPID for four-link manipulator

The SNNFOPID was tested on the four-link robotic manipulator, as illustrated in Fig. 33.

The robot's parameters are detailed in Table 22.:

Table 7
Comparison of IAE for $Link_1$ and $Link_2$ in different m_p values.

Payload variation (Kg/s)	Sharma FOPID (Richa Sharma & M., 2015)		SNNFOPID	
	$Link_1$	$Link_2$	$Link_1$	$Link_2$
case 1 m_p = 0.125 t	0.008793	0.01524	0.006472	0.003374
case 2 m_p = −0.125 t+0.5	0.009061	0.01482	0.006814	0.003466
case 3 $m_p = \begin{cases} 0.5, & t \le 1.5 \\ \text{t-1}, & 1.5 \le t \le 2.5 \\ \text{0.5t+0.125}, & 2.5 \le t \le 3.5 \\ 2, & 3.5 \le t \le 4 \end{cases}$	0.009278	0.01632	0.01025	0.005973

Table 8
IAE values of $Link_1$ and $Link_2$ in the presence of noise.

Added noise (rad)	Sharma FOPID Richa Sharma and M. (2015)		SNNFOPID	
	$Link_1$	$Link_2$	$Link_1$	$Link_2$
case 1 $Link_1$	0.02139	0.01496	0.02182	0.01046
case 2 $Link_1$	0.008787	0.02413	0.02413	0.04823
case 3 Both links	0.02139	0.02413	0.0221	0.02245

Table 9
The value of parameters of double inverted pendulum.

Parameter	$Link_1$	$Link_2$	
Mass (kg)	0.5	1	1
g(m/s^2)	9.81	9.81	9.81
length from the joint to its center of gravity (m)	0.3	0.5	0.5
Dimensions (m)	[0.2 0.04 0.6]	[0.5 0.15 0.05]	[0.5 0.15 0.05]

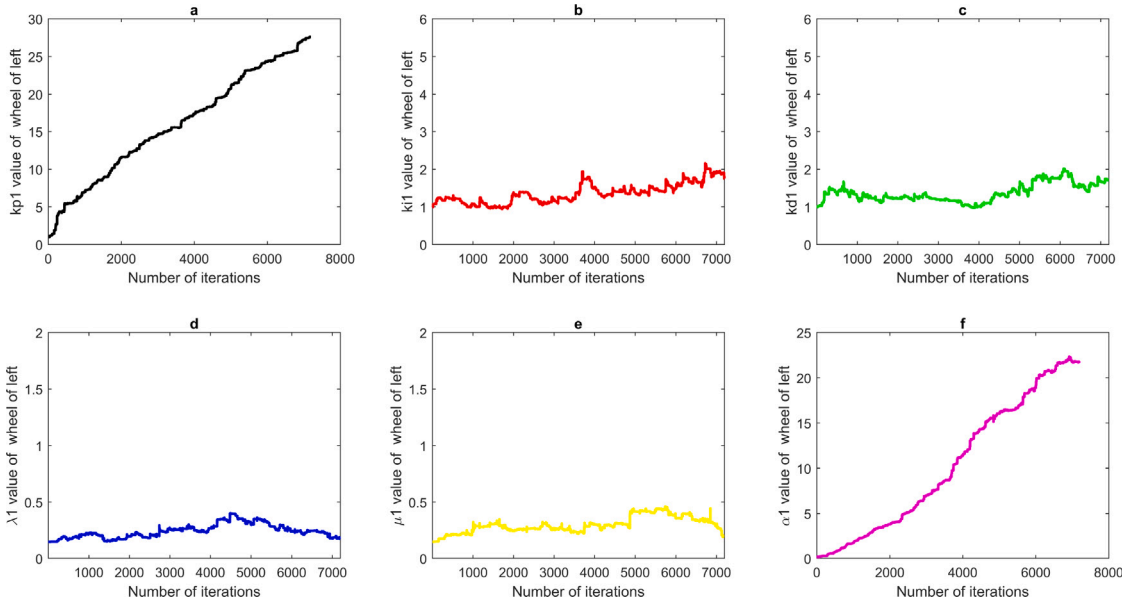


Fig. 28. (a) $kp1, (b)$ $ki1, (c)$ $kd1, (d)$ $\lambda 1, (e)$ $\mu 1, (f)$ $\alpha 1$.

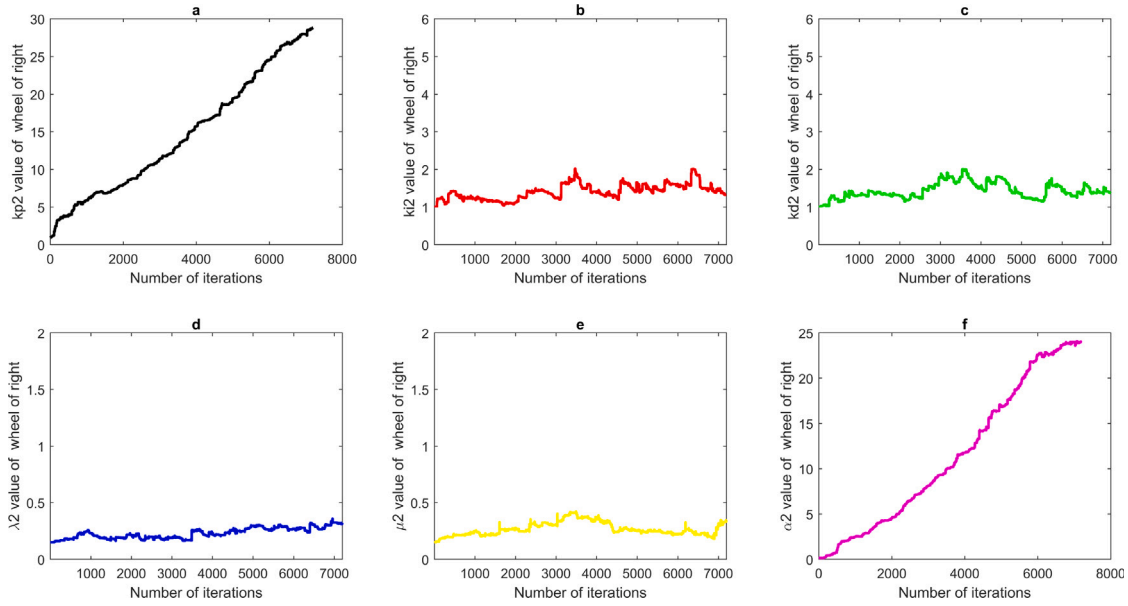


Fig. 29. $(a)\ kp2, (b)\ ki2, (c)\ kd2, (d)\ \lambda 2, (e)\ \mu 2, (f)\ \alpha 2$.

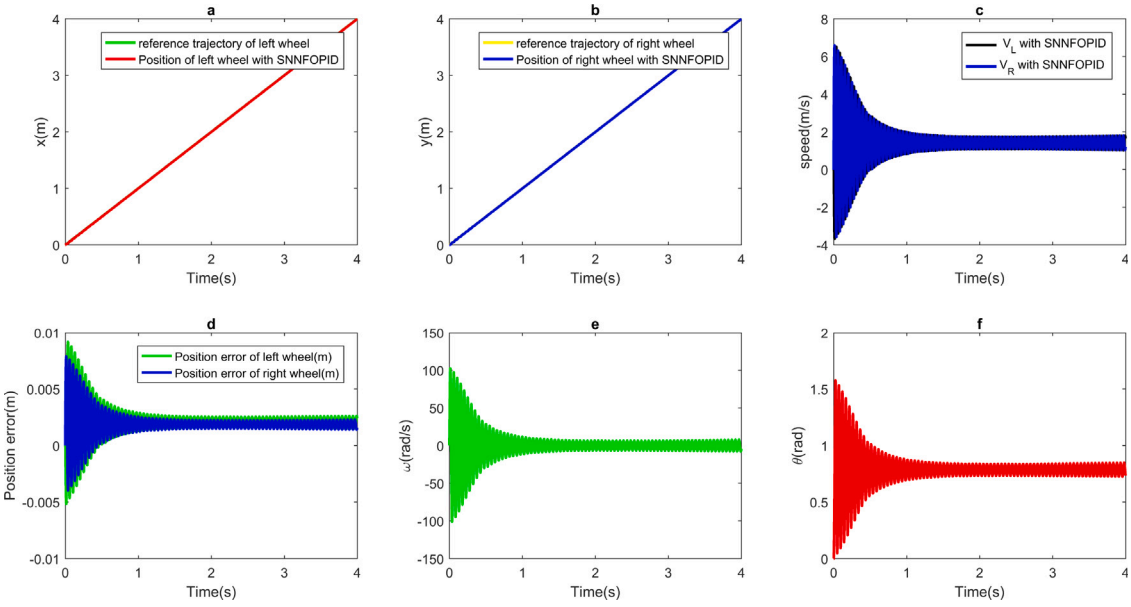


Fig. 30. (a) Trajectory tracking of the left wheel, (b) Trajectory tracking of the right wheel, (c) Controlled speed of left (V_L) and right (V_R) wheels (m/s), (d) Position error of left wheel, and right wheel (m), (e) Angular velocity of the mobile robot (rad/s), (f) θ (rad) is the angle between the X-coordinate and the heading direction in the case of no disturbance.

Table 10
The final value of the control parameters obtained from the SNNFOPID algorithm for the cart in the double inverted pendulum.

K_{p1}	K_{i1}	K_{d1}	μ_1	λ_1	α_1
16.729	3.9569	3.2365	0.7936	0.8944	15.7529

Table 11
The final value of the control parameters obtained from the SNNFOPID algorithm for link1 in the double inverted pendulum.

K_{p2}	K_{i2}	K_{d2}	μ_2	λ_2	α_2
15.0279	3.8647	5.083	1.0229	0.3752	16.0066

Table 12
The final value of the control parameters obtained from the SNNFOPID algorithm for link2 in the double inverted pendulum.

K_{p3}	K_{i3}	K_{d3}	μ_3	λ_3	α_3
14.6057	4.9359	3.2976	0.5223	0.4990	15.3729

Table 13
The final value of IAE obtained from the SNNFOPID algorithm for the double inverted pendulum for the equilibrium state.

$cart$	$link_1$	$link_2$
2.347*10e−15	2.352*10e−15	−1.61*10e−14

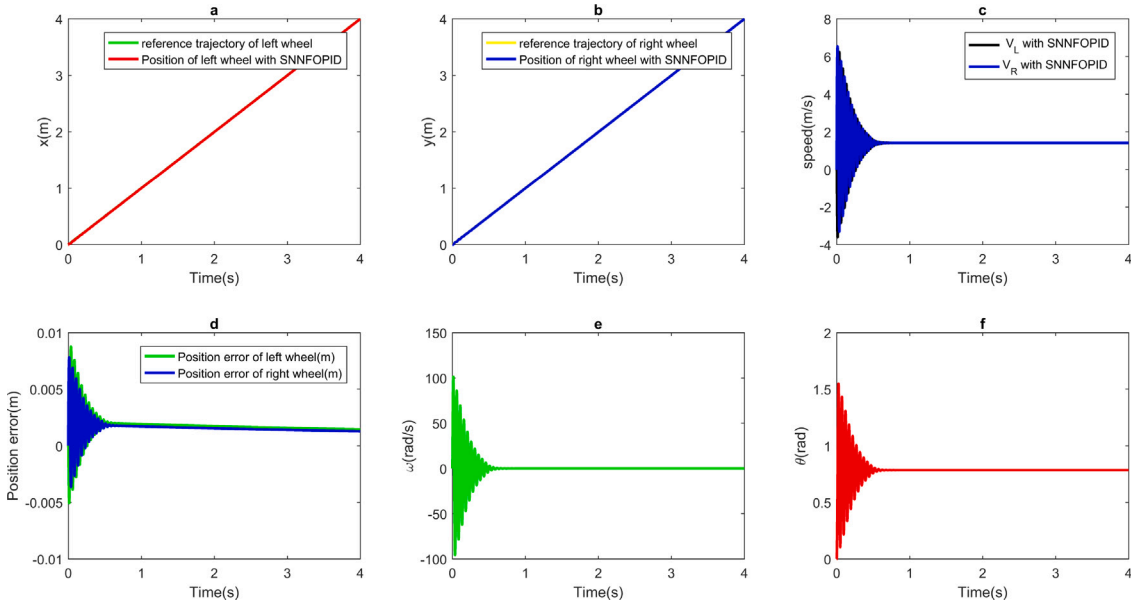


Fig. 31. (a) Trajectory tracking of the left wheel, (b) Trajectory tracking of the right wheel, (c) Controlled speed of left (V_L) and right (V_R) wheel (m/s), (d) Position error of left wheel, and right wheel (m), (e) Angular velocity of the mobile robot (rad/s), (f) θ (rad) is the angle between the X-coordinate and the heading direction in the presence of disturbance.

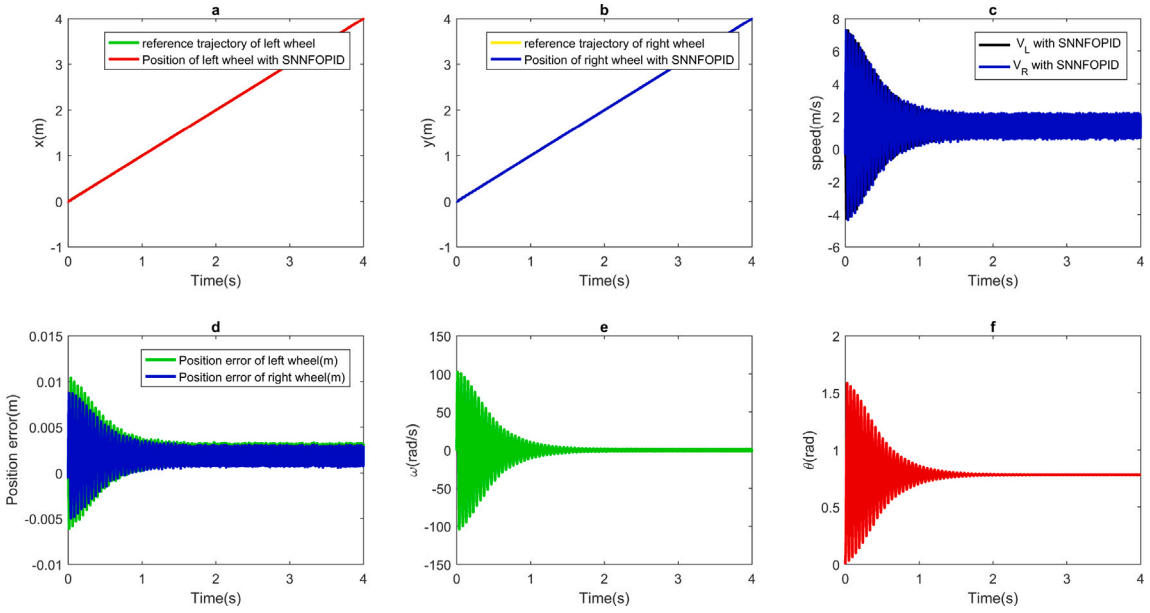


Fig. 32. (a) Trajectory tracking of the left wheel, (b) Trajectory tracking of the right wheel, (c) Controlled speed of left (V_L) and right (V_R) wheels (m/s), (d) Position error of wheel of the left, and right wheel (m), (e) Angular velocity of the mobile robot (rad/s), (f) θ (rad) is the angle between the X-coordinate and the heading direction in the case of the presence of noise.

Table 14
The final value of IAE obtained from the SNNFOPID algorithm for the double inverted pendulum in the case of no disturbance.

cart	$link_1$	$link_2$
0.03382	0.05859	0.02793

Table 15
The final value of IAE obtained from the SNNFOPID algorithm for the double inverted pendulum in the presence of disturbance.

cart	$link_1$	$link_2$
0.03395	0.05842	0.02786

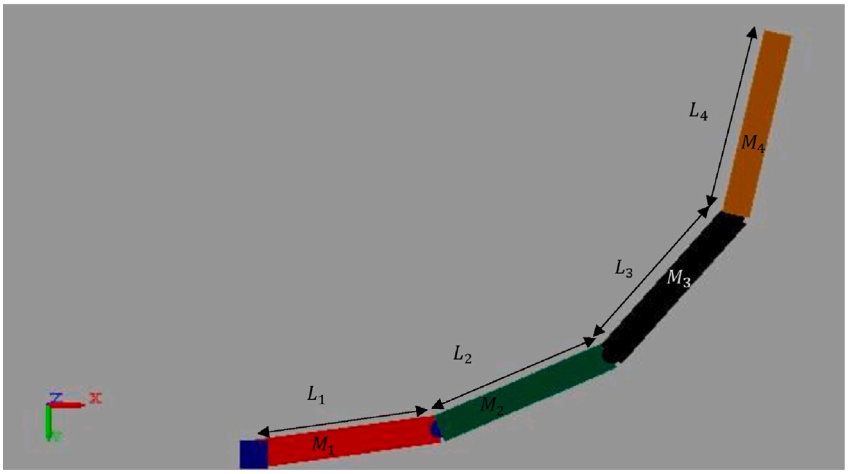


Fig. 33. Four-link robotic manipulator in SIMULINK.

Table 16
The final value of IAE obtained from the SNNFOPID algorithm for the double inverted pendulum in the presence of noise.

$cart$	$link_1$	$link_2$
0.04341	0.06032	0.03455

Table 17
The final value of the control parameters obtained from the SNNFOPID algorithm for the left wheel of the mobile robot.

K_{p1}	K_{i1}	K_{d1}	μ_1	λ_1	α_1
27.5958	1.7488	1.7133	0.2026	0.1775	21.7967

Table 18
The final value of the control parameters obtained from the SNNFOPID algorithm for right wheel of the mobile robot.

K_{p2}	K_{i2}	K_{d2}	μ_2	λ_2	α_2
28.7968	1.3235	1.3767	0.3459	0.3233	24.0087

Table 19
The final value of IAE obtained from the SNNFOPID algorithm for the two-wheeled mobile robot in the case of no disturbance.

IAE_{right}	IAE_{right}
0.009	0.008011

Table 20
The final value of IAE obtained from the SNNFOPID algorithm for the two-wheeled mobile robot in the presence of disturbance.

IAE_{left}	IAE_{right}
0.00902	0.00798

Table 21
The final value of IAE obtained from the SNNFOPID algorithm for two-wheeled mobile robot in the presence of noise.

IAE_{left}	IAE_{right}
0.009209	0.008189

As previously discussed, dedicated SNNs are designed for each degree of freedom to optimize the parameters of the corresponding robot joint. This results in a total of four SNNs being utilized. Tables 23–26 present the final control parameter values for each link of the robot, while Figs. 34–37 illustrate the parameter values of each controller throughout the learning process.

To delve further into the stability of the control parameters, it is essential to note that each parameter fluctuates within a restricted range relative to its corresponding maximum synaptic weight (S_m) and eventually attains a state of relative stability. For a comprehensive insight into this aspect, Tables 27–30 present the standard deviation values of each parameter during the final 100 iterations of the learning process.

Based on the data presented in the tables, it is evident that the standard deviation of the parameter values in the final 100 iterations is significantly low. This suggests that the parameter values have converged to a stable level during these iterations, indicating that the learning process has reached a halt.

Figs. 38 to 41 provide detailed insights into the Integral Absolute Error (IAE) values of every link within the robot, showcasing the impact of variations in each control parameter on the system's performance.

Based on these figs, it is evident that the learning process is effectively targeted towards minimizing the IAE for each link.

For this robot, we conducted a thorough analysis of the SNNFOPID performance across three scenarios: no disturbance, presence of disturbance, and noise. The input signals for each link, disturbance, and noise are outlined below: $input_1 = 0.3\ \sin(t)$

$input_2 = 0.5\ \sin(t+\pi/2)$

$input_3 = \sin(t)$

$input_4 = 0.5\ \sin(t+\pi/3)$

Disturbance = 0.5sin(25t) (N M/s)

$-0.01 \leq t \leq 0.01\ rad$

Tables 31–33 provide the IAE values, while Figs. 42, 43, and 44 show trajectory tracking, controller output, and position error under different cases respectively: no disturbance, presence of disturbance, and noise.

The data show in all scenarios IAE values remain below 0.07 and the position error is less than 0.03 radian.

6. SNNFL

In this section, the method of hybrid learning control is generalized to feedback linearization. It is a nonlinear controller (called computed torque in robotics), and the control law is shown in Eq. (11):

$$V = \ddot{r} + \alpha * (KP * e(t) + KD * \dot{e}(t)),\ \tau = M * V + C(q,\dot{q}) + G(q) \qquad (11)$$

where, V is the output of the linear controller, $\ddot{r}$ is the second derivative of the input signal to the system, KP is the proportional constant gain,

Table 22
The value of parameters of four-link robotic manipulator.

Parameter	$Link_1$	$Link_2$	$Link_3$	$Link_4$
Mass (kg)	2	0.8	0.6	0.5
$g(m/s^2)$	9.81	9.81	9.81	9.81
length from the joint to its center of gravity (m)	0.5	0.5	0.5	0.5
Dimensions (m)	[0.5 0.15 0.05]	[0.5 0.15 0.05]	[0.5 0.15 0.05]	[0.5 0.15 0.05]

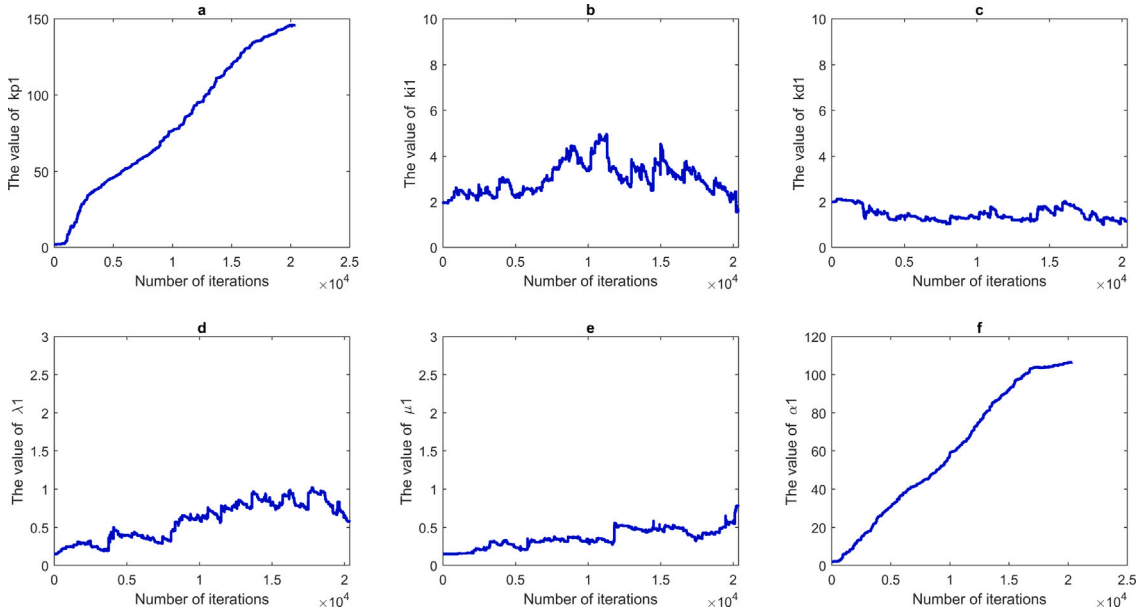


Fig. 34. $(a)\ kp1, (b)\ ki1, (c)\ kd1, (d)\ \lambda 1, (e)\ \mu 1, (f)\ \alpha 1$.

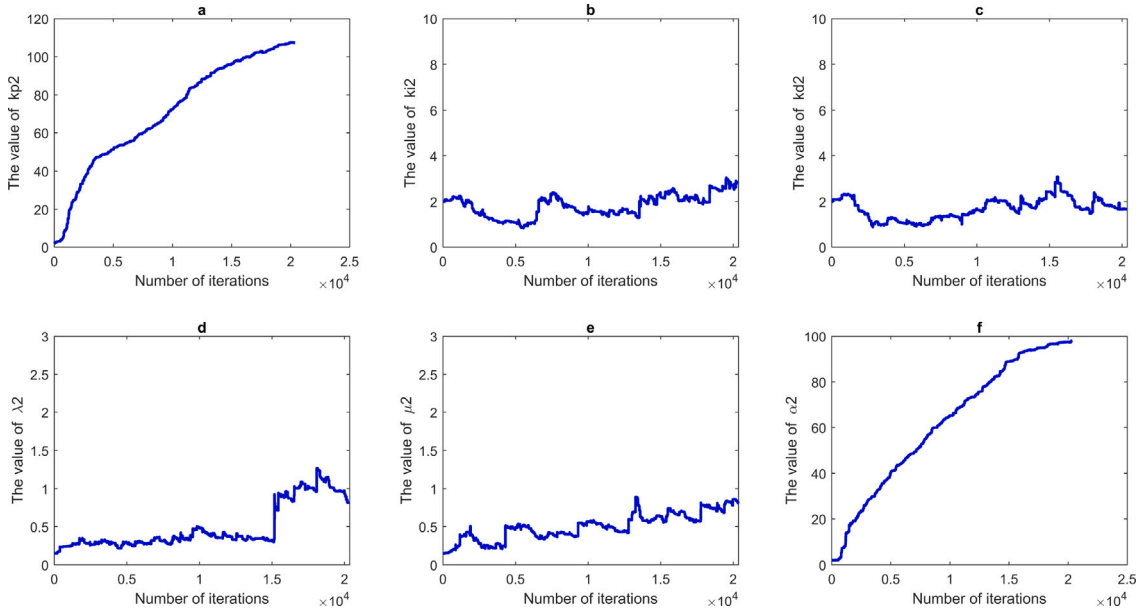


Fig. 35. $(a)\ kp2, (b)\ ki2, (c)\ kd2, (d)\ \lambda 2, (e)\ \mu 2, (f)\ \alpha 2$.

Table 23
The final value of the control parameters obtained from the SNNFOPID algorithm for $link_1$ in the four-link robotic manipulator.

K_{p1}	K_{i1}	K_{d1}	μ_1	λ_1	α_1
145.7512	1.56	1.1597	0.7791	0.5755	106.4618

Table 24
The final value of the control parameters obtained from the SNNFOPID algorithm for $link_2$ in the four-link robotic manipulator.

K_{p2}	K_{i2}	K_{d2}	μ_2	λ_2	α_2
107.2657	2.8126	1.6731	0.8033	0.8123	97.8945

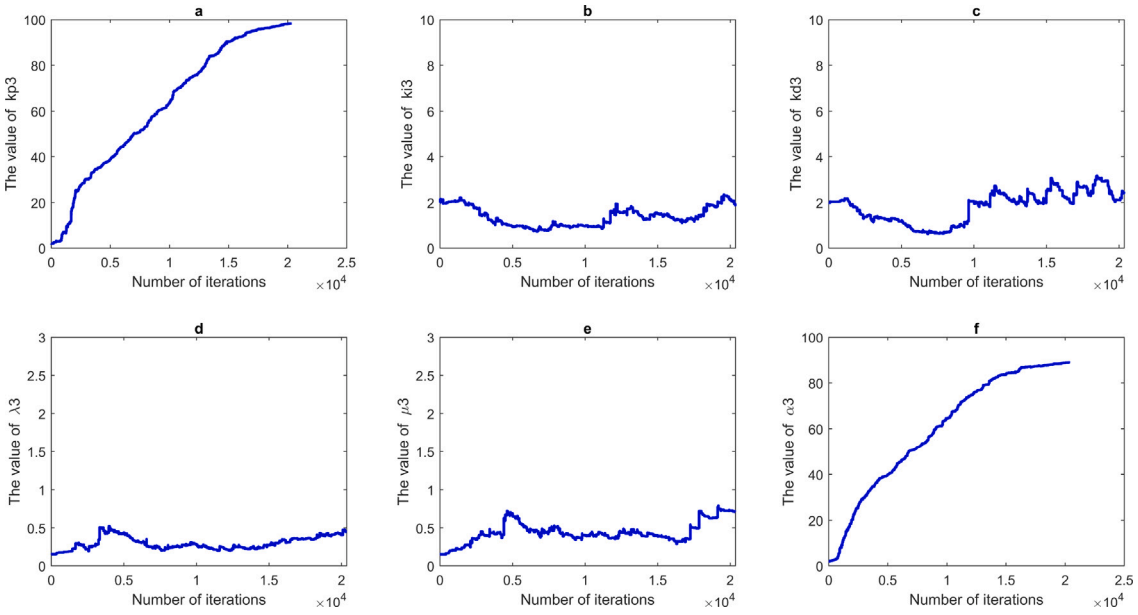


Fig. 36. $(a)\ kp3, (b)\ ki3, (c)\ kd3, (d)\ \lambda 3, (e)\ \mu 3, (f)\ \alpha 3$.

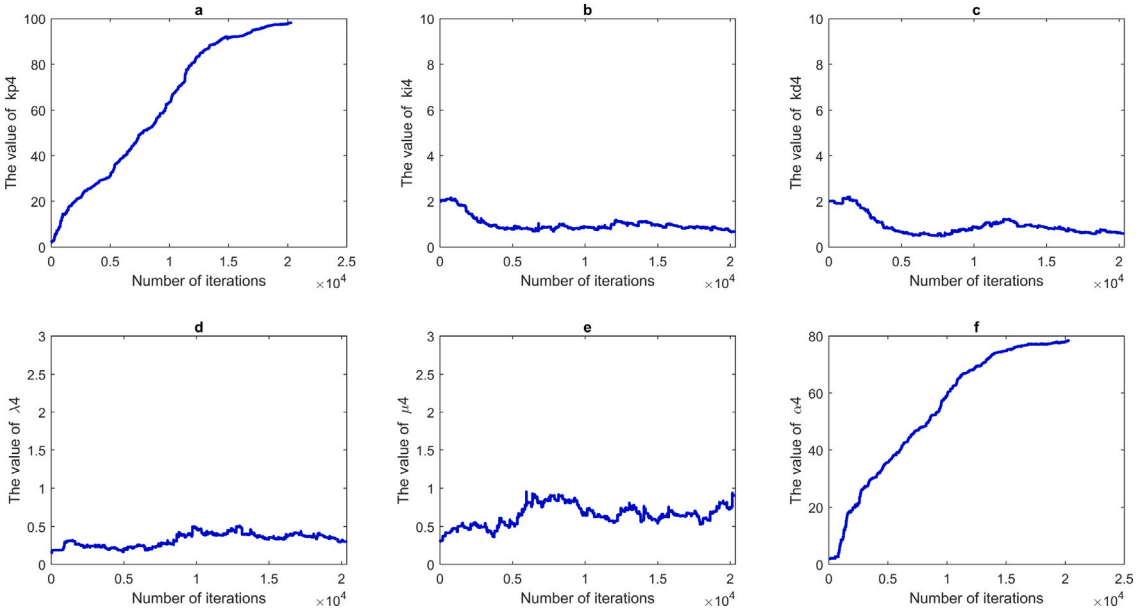


Fig. 37. $(a)\ kp4, (b)\ ki4, (c)\ kd4, (d)\ \lambda 4, (e)\ \mu 4, (f)\ \alpha 4$.

Table 25
The final value of the control parameters obtained from the SNNFOPID algorithm for $link_3$ in the four-link robotic manipulator.

K_{p3}	K_{i3}	K_{d3}	μ_3	λ_3	α_3
98.3313	1.9365	2.41625	0.7103	0.4526	88.8666

Table 26
The final value of the control parameters obtained from the SNNFOPID algorithm for $link_4$ in the four-link robotic manipulator.

K_{p4}	K_{i4}	K_{d4}	μ_4	λ_4	α_4
98.1995	0.6726	0.5999	0.9056	0.3043	78.380

Table 27
Standard deviation (SD) of the controller's $Link_1$ control parameters during the final 100 iterations to demonstrate the stability of the parameters.

SD_{Kp1}	SD_{Ki1}	SD_{Kd1}	$SD_{\mu 1}$	$SD_{\lambda 1}$	$SD_{\alpha 1}$
0.0625	0.0725	0.0019	0.0057	9.8564 10e−4	0.0103

Table 28
Standard deviation (SD) of the controller's $Link_2$ control parameters during the final 100 iterations to demonstrate the stability of the parameters.

SD_{Kp2}	SD_{Ki2}	SD_{Kd2}	$SD_{\mu 2}$	$SD_{\lambda 2}$	$SD_{\alpha 2}$
0.0669	0.0185	0.000047891	0.0104	0.00023222	0.0677

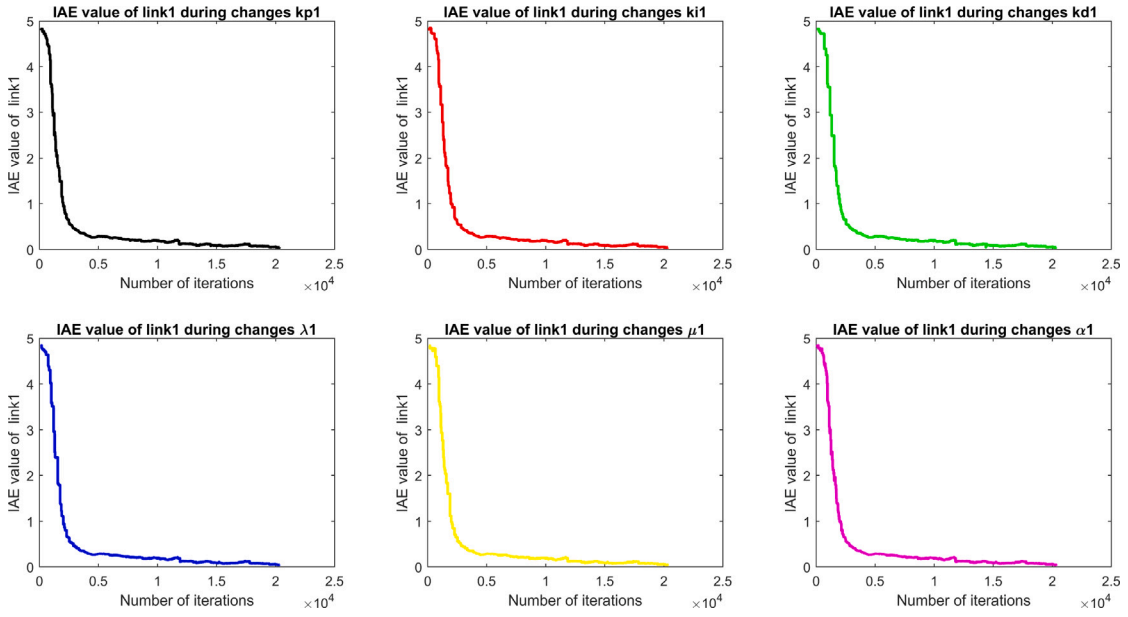


Fig. 38. (a) IAE value of $link_1$ during changes kp1, (b) IAE value of $link_1$ during changes ki1, (c) IAE value of $link_1$ during changes kd1, (d) IAE value of $link_1$ during changes λ_1, (e) IAE value of $link_1$ during changes μ_1, (f) IAE value of $link_1$ during changes α_1.

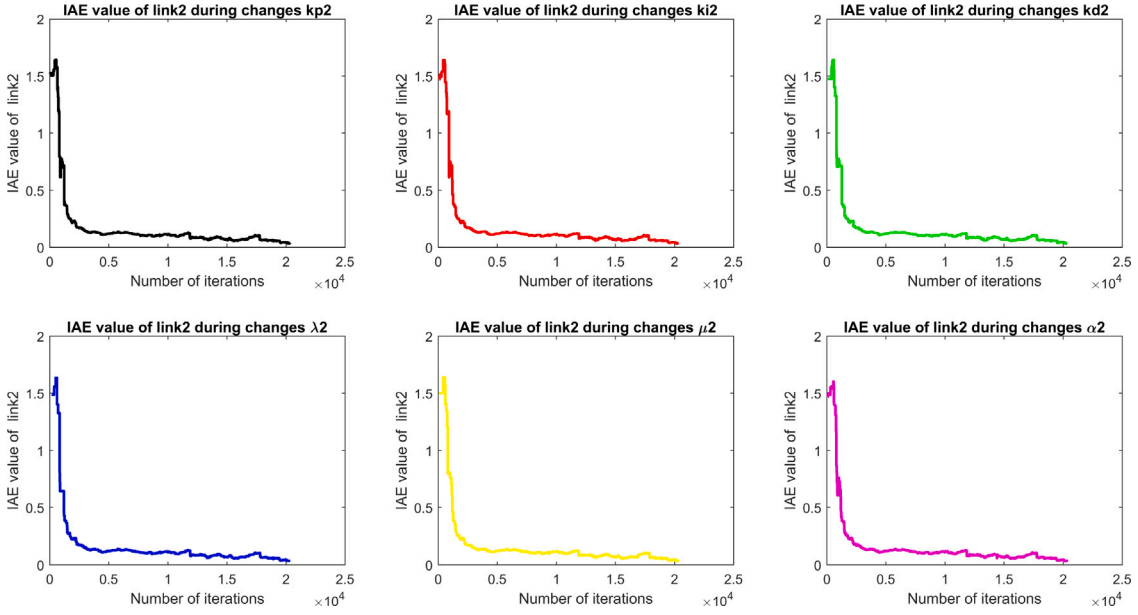


Fig. 39. (a) IAE value of $link_2$ during changes kp2, (b) IAE value of $link_2$ during changes ki2, (c) IAE value of $link_2$ during changes kd2, (d) IAE value of $link_2$ during changes λ_2, (e) IAE value of $link_2$ during changes μ_2, (f) IAE value of $link_2$ during changes α_2.

Table 29
Standard deviation (SD) of the controller's $Link_3$ control parameters during the final 100 iterations to demonstrate the stability of the parameters.

SD_{Kp3}	SD_{Ki3}	SD_{Kd3}	$SD_{\mu 3}$	$SD_{\lambda 3}$	$SD_{\alpha 3}$
0.0011	0.0528	0.0258	0.0034	0.0018	0.0429

Table 30
Standard deviation (SD) of the controller's $Link_4$ control parameters during the final 100 iterations to demonstrate the stability of the parameters.

SD_{Kp4}	SD_{Ki4}	SD_{Kd4}	$SD_{\mu 4}$	$SD_{\lambda 4}$	$SD_{\alpha 4}$
0.0337	0.0019	0.0000003533	0.00017	0.00064	0.009

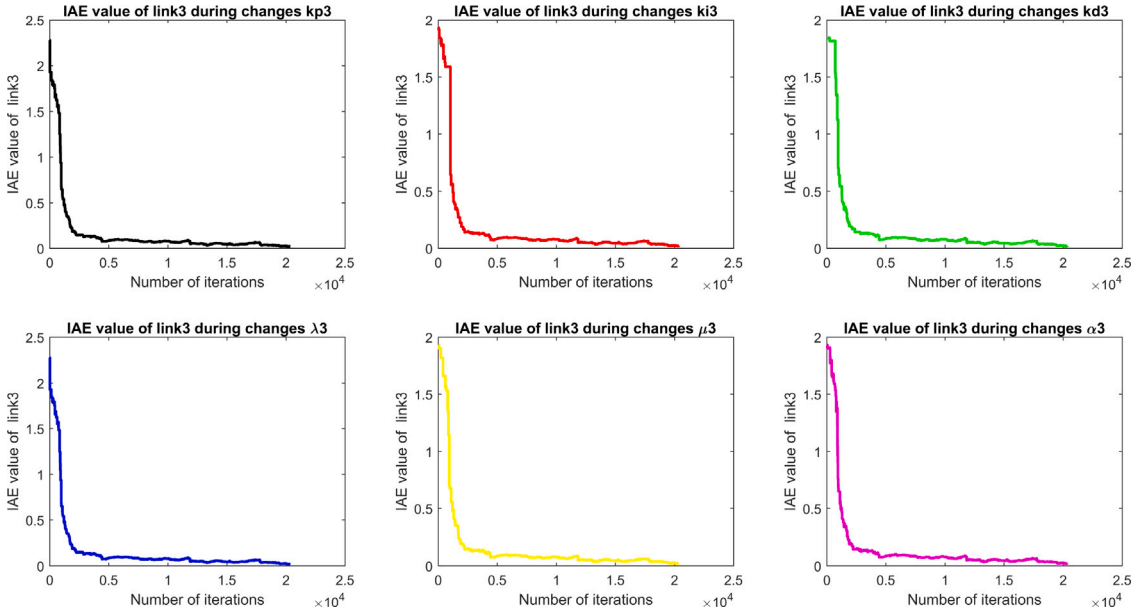


Fig. 40. (a) IAE value of $link_3$ during changes kp3, (b) IAE value of $link_3$ during changes ki3, (c) IAE value of $link_3$ during changes kd3, (d) IAE value of $link_3$ during changes λ_3, (e) IAE value of $link_3$ during changes μ_3, (f) IAE value of $link_3$ during changes α_3.

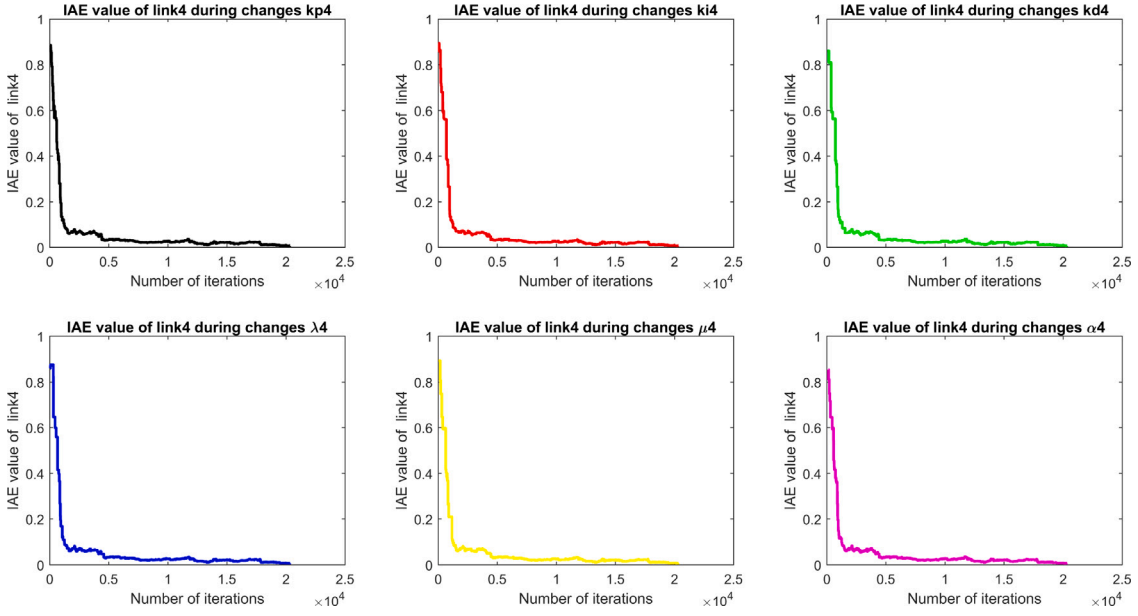


Fig. 41. (a) IAE value of $link_4$ during changes kp4, (b) IAE value of $link_4$ during changes ki4, (c) IAE value of $link_4$ during changes kd4, (d) IAE value of $link_4$ during changes λ_4, (e) IAE value of $link_4$ during changes μ_4, (f) IAE value of $link_4$ during changes α_4.

Table 31
The final value of IAE obtained from the SNNFOPID algorithm for the four-link robotic manipulator in the case of no disturbance.

$link_1$	$link_2$	$link_3$	$link_4$
0.04987	0.03839	0.0229	0.007869

Table 32
The final value of IAE obtained from the SNNFOPID algorithm for the four-link robotic manipulator in the presence of disturbance.

$link_1$	$link_2$	$link_3$	$link_4$
0.0745	0.05925	0.03607	0.01578

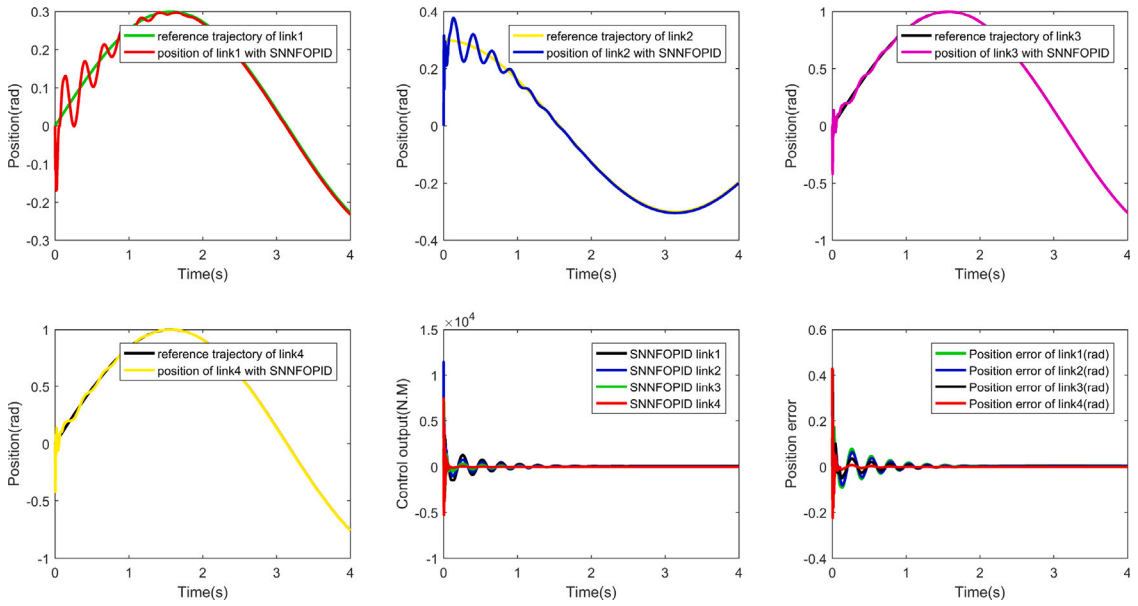


Fig. 42. (a) Trajectory tracking of $link_1$, (b) Trajectory tracking of $link_2$, (c) Trajectory tracking of $link_3$, (d) Trajectory tracking of $link_3$, (e) Control output of $link_1$, $link_1$, $link_3$, and $link_4$ (N m), (f) Position error of $link_1$, $link_1$, $link_3$, and $link_4$ in the case of no disturbance.

Table 33
The final value of IAE obtained from the SNNFOPID algorithm for the four-link robotic manipulator in the presence of noise.

$link_1$	$link_2$	$link_3$	$link_4$
0.05462	0.04597	0.03492	0.02492

KD is the derivative constant gain, e(t) is the error, $\dot{e}(t)$, τ is the control signal, M is the inertia matrix.$C(q, \dot{q})$ represents the vector of Coriolis and centrifugal forces, and $G(q)$ is a vector of gravity torques.

6.1. SNNFL structure

SNNs are used for simultaneous optimization of the parameters of this controller (KP, KD, and α) when the controller is running on a robot. α is added to speed up the stability of the parameters multiplied by KP and KD. In the new SNN Feedback Linearization (SNNFL) method, the sum of IAE1 and IAE2 is evaluated. If the changes in synaptic weights are in the direction of reducing the sum of IAE1, and IAE2, then the system will receive dopamine; otherwise, it will receive a penalty. Fig. 45 shows the structure of Feedback Linearization, and how it is related to SNNs. With a slight change in the dopamine conditions compared to the SNNFOPID algorithm, for SNNFL we checked the condition of the sum of both IAE1 and IAE2 so that if IAE1 + IAE2 reaches a lower value than the previous state, the reward will be released and dopamine amount will be considered as +1 ($DA = +1$), otherwise the punishment will be released and amount of dopamine will be considered as −1 ($DA = -1$) for the active synapse. Injection of Dopamine into neurons changes the synaptic weights and it has an effect on the way of connecting input–output neurons. It results in changing the parameters IAE1 + IAE2 and then they are checked again compared to the previous state, and this process continues until IAE1 + IAE2 reaches the minimum possible value .

6.2. Application of SNNFL on controlling a two-DOF robot

Here, in order to verify the performance of SNNFL, its application in controlling a two-DOF robot is studied. Fig. 46 shows the value of the feedback linearization controller parameters during the iteration. After finishing the learning process, the feedback linearization controller parameters are obtained. In Fig. 47, IAE1 and IAE2 values are shown. They gradually and quite regularly tend to the lowest possible.

As is seen in Fig. 46, the parameters reach their stable values after about 1200 iterations. The final parameters of feedback linearization KP, KD, and α obtained from the reinforcement learning for controlling the two-DOF robot are 11.7336, 9.6222, and 8.5059, respectively.

According to Fig. 46, α, KP, and KD parameters are stable after 1200 epochs, and after that, they fluctuate in a very limited range. To Study the performance of the new controller in the presence of disturbance and noise, two case studies were considered; $Sin(t)$ function for disturbance and $-0.01 \leq noise \leq 0.01$ (rad) for random noise were used. Figs. 48 and 49 show the results of considering disturbance and random noise in position error and control output values, respectively. The values of IAE1 and IAE2, in the presence of disturbance, are shown in Table 34, and their value in the presence of random noise are shown in Table 35.

The results in Fig. 49 show the range of the torque for $link_1$ and $link_2$ is (−10,40) Nm and (−5,20) Nm, respectively and the error value for both links is (−0.01,0.01) rad in presence of disturbance.

The results in Fig. 49 indicate that the torque range for $link_1$ and for $link_2$ is (−10, 40) Nm and (−5,20) Nm, respectively, and the error value for both links is (−0.02,0.02) rad in presence of random noise.

Table 34
IAE values of $Link_1$ and $Link_2$ in presence of disturbance (SNNFL method).

Disturbance $(Sin(t)Nm)$	IAE values	
	$Link_1$	$link_2$
Disturbance on $Link_1$	0.01348	0.01826
Disturbance on $Link_2$	0.01546	0.02013
Disturbance on both Link	0.01198	0.01497

Table 35
IAE values of $Link_1$ and $Link_2$ in the presence of random noise (SNNFL method).

Random noise $(-0.01 \leq$ noise ≤ 0.01 rad)	IAE values	
	$Link_1$	$link_2$
Random noise on $Link_1$	0.02266	0.01389
Random noise on $Link_2$	0.001148	0.02233
Random noise on both Link	0.02265	0.02233

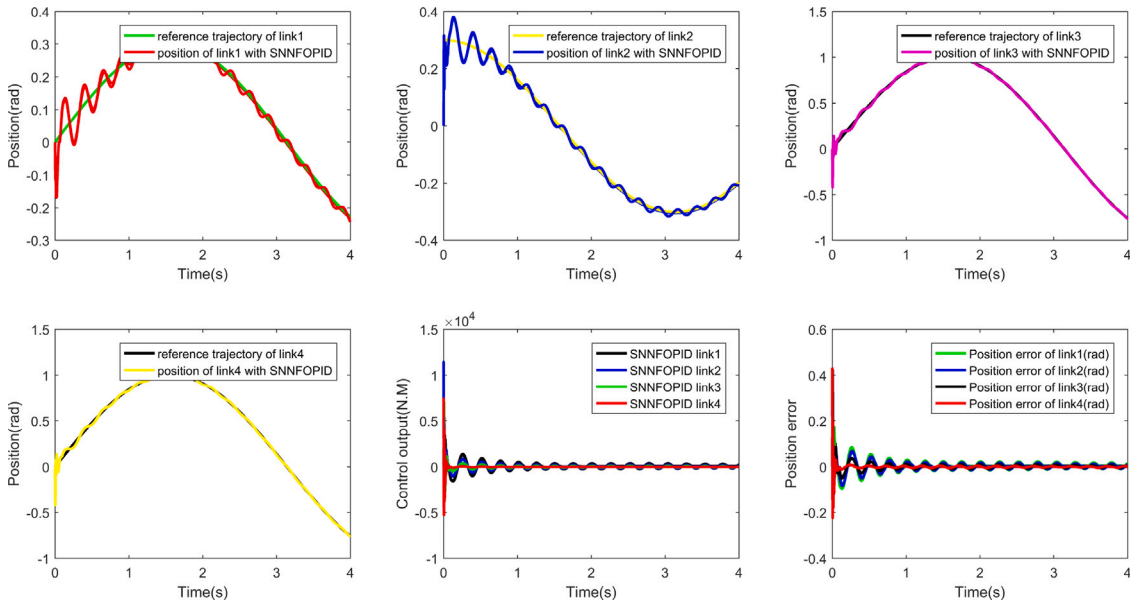


Fig. 43. (a) Trajectory tracking of $link_1$, (b) Trajectory tracking of $link_2$, (c) Trajectory tracking of $link_3$, (d) Trajectory tracking of $link_3$, (e) Control output of $link_1$, $link_1$, $link_3$, and $link_4$ (N m), (f) Position error of $link_1$, $link_1$, $link_3$, and $link_4$ in the presence of disturbance.

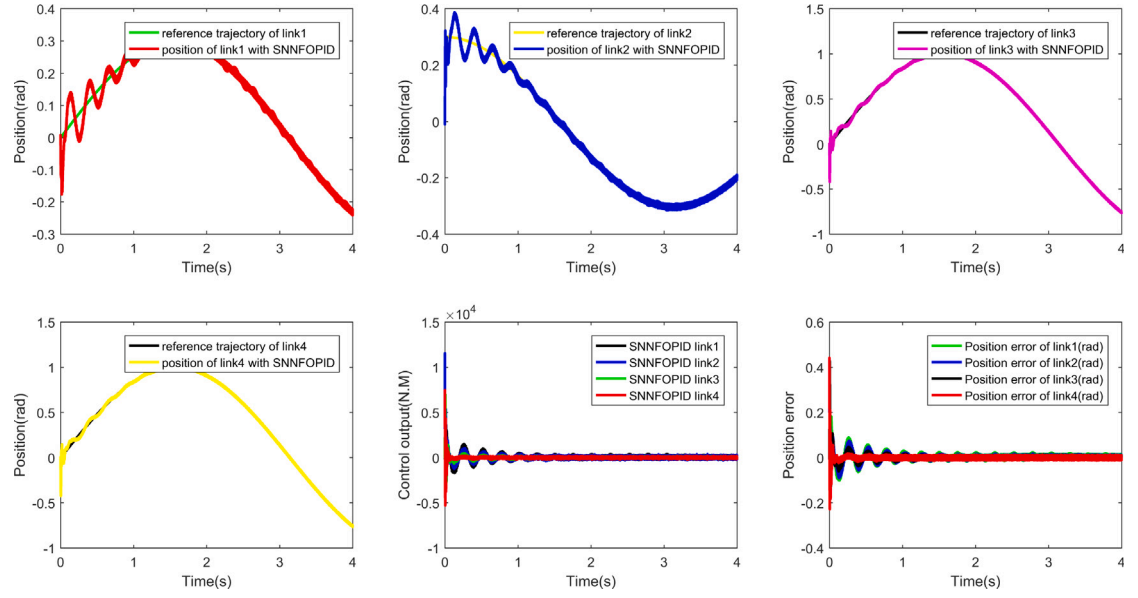


Fig. 44. (a) Trajectory tracking of $link_1$, (b) Trajectory tracking of $link_2$, (c) Trajectory tracking of $link_3$, (d) Trajectory tracking of $link_3$, (e) Control output of $link_1$, $link_1$, $link_3$, and $link_4$ (N m), (f) Position error of $link_1$, $link_1$, $link_3$, and $link_4$ in the presence of noise.

Tables 34 and 35 show the value of IAE for the two-link robot is less than 0.03 in any case of applying disturbance and random noise.

7. Convergence time of the robots

The SNNFOPID code execution time (on MATLAB software package) for each robot in each iteration and the total run time to reach parameter convergence is shown in Table 36 for an Asus Core i5 system.

Although the complexity of SNNs is higher than the complexity of ANNs, their computational costs in practice are lower because of communication through spikes. Based on the information in Table 36, it is evident that higher control parameter values and a greater number of degrees of freedom take longer to converge and lead to a more complex system. However, if we utilize a more robust system to run the code in MATLAB, we can achieve parameter convergence much faster.

8. Discussion

Parameter adjustment of controllers has been widespread in recent years. In the new hybrid learning-control system, at first, a method of adjusting any function through SNNs is presented and then, its application in the parameter adjustment of two nonlinear controllers (FOPID and feedback linearization) is studied. This method is mentioned as a critic–actor system, in which the reinforcement learning in the SNNs context is the critic and the controller is the actor system. These controllers are widely used due to their high efficiency in

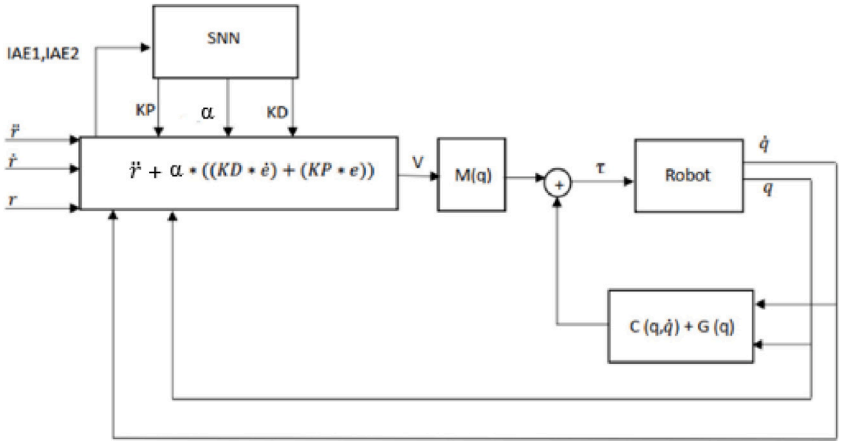


Fig. 45. The structure of SNNFL.

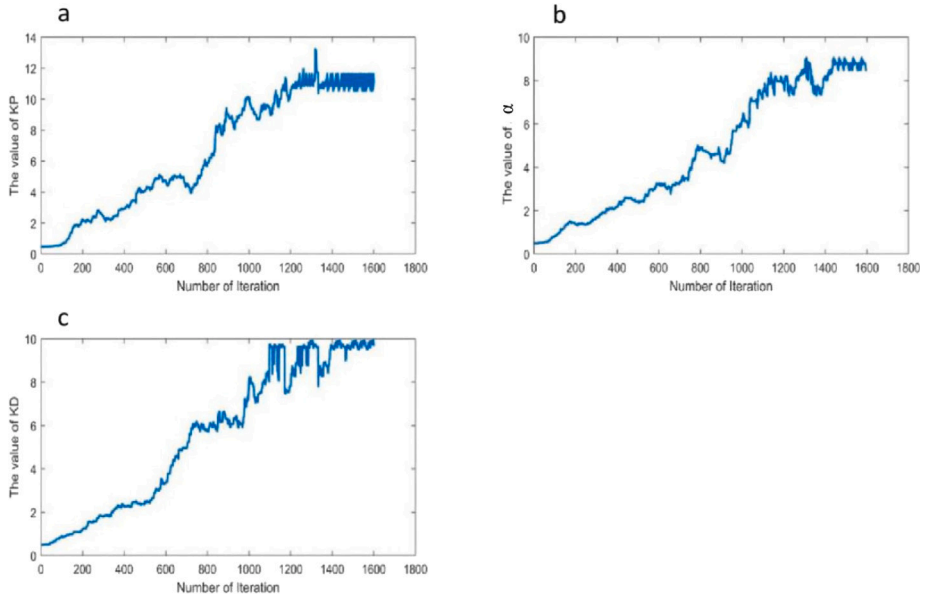


Fig. 46. The stability of SNNFL controller parameters.

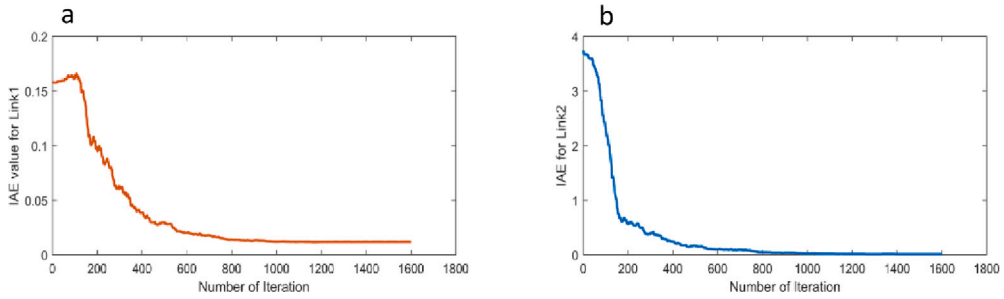


Fig. 47. (a) IAE value versus iterations for $Link_1$ (b) IAE value versus iterations for $Link_2$ in SNNFL method.

Table 36
Convergence time of control parameters for each robot during learning.

Robot	Time for each iteration(s)	Time for convergence(hours)
Two-link robot manipulator with SNNFOPID	1.91	19.85656
Two-link robot manipulator with SNNFL	0.48	0.45828
Double inverted pendulum with SNNFOPID	1.8768	3.455
Mobile robot with SNNFOPID	1.86	3.71948
Four-link robotic manipulator with SNNFOPID	2.95	16.6789

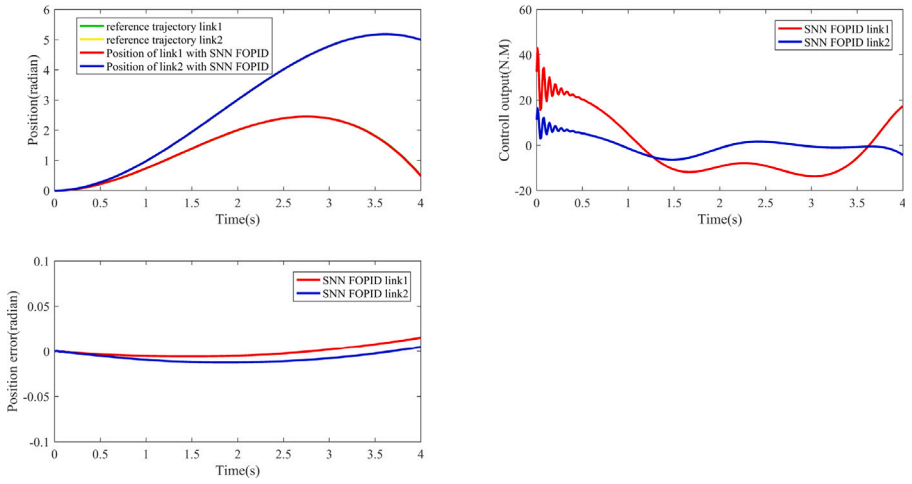


Fig. 48. (a) Trajectory tracking, (b) Control outputs, (c) Position error of $Link_1$ and $Link_2$ in the presence of disturbance for SNNFL method.

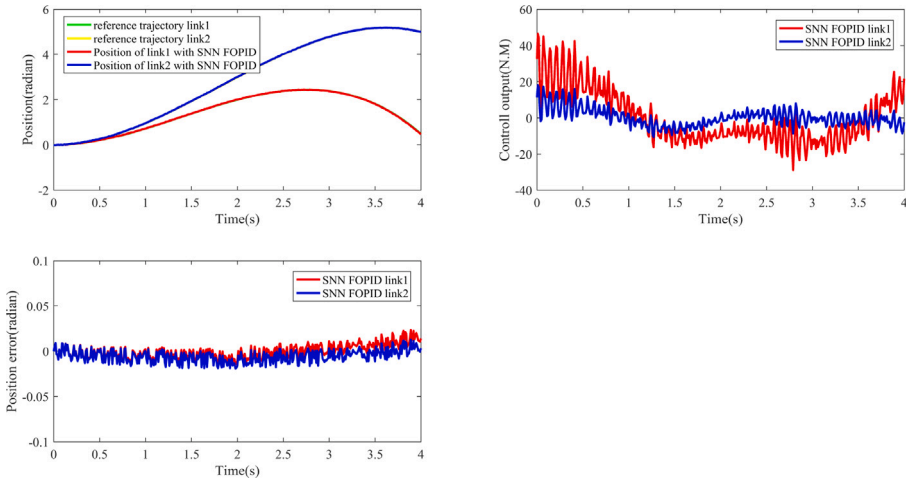


Fig. 49. (a) Trajectory tracking, (b) Control outputs, (c) Position error of $Link_1$ and $Link_2$ in the presence of random noise in the third case of Table 35 for SNNFL method.

various engineering branches, especially in robotics. Some optimization methods are used to adjust these parameters. SNNs have a bright future due to their simplicity in structure, high energy efficiency, and high processing speed. Reinforcement learning and the existence of the time dimension are the strengths that have made these networks practical in dynamic systems like robots. In addition, SNN-based hardware could be used for its simplicity, high speed and accuracy, and energy saving. It also contributes significantly to building fast and energy-saving hardware based on a combination of neural networks, especially RNN and LSTM neural networks (which are dynamic and more complex networks) with SNNs. Furthermore, this technique offers the advantage of enabling continuous control of robots through the augmentation of neuron count. Additionally, the combination of SNNFOPID and a multi-agent system allows for the collaborative control of multiple robots.

9. Conclusion

In this study, we presented a new algorithm for hybrid learning control systems by reinforcement in a spiking neural network platform based on the Izhkevich model of a single neuron. For this purpose, the controller was assumed to be a nonlinear function and then its parameters were adjusted by the new proposed reinforcement learning algorithm which is more energy efficient. The methodology was presented and to prove the applicability of the new method, it was implemented to two nonlinear control methods named FOPID and feedback linearization. Results showed that the algorithm can be implemented on any nonlinear controller because we use the function of the controller and convert it into a function in the SNN context. Some case studies were designed, and the performance of the proposed algorithm was validated and compared with another similar FOPID method. The results showed that SNNFOPID performs better against disturbances, friction, and robot payload changes. We showed final position error values of link1 and Link2 were less than 0.007 rad and 0.005 rad respectively. The only disadvantage of SNNFOPID compared with Richa Sharma and M. (2015) is the wider range of control torque applied to the robot links. Our study involved examining the performance of the SNN FOPID in controlling the double inverted pendulum. The results showed an impressive error position of less than 0.03 rad in four cases. We conducted a test on SNN FOPID using a 4-degree-of-freedom manipulator, and the error in each joint of the robot was less than 0.03 rad. We also demonstrated the remarkable efficiency and performance of SNNFOPID in controlling the velocities and positions of the mobile robot, resulting in an error of fewer than 0.01 m. Additionally, we used the algorithm to optimize the parameters of the feedback linearization controller, and the results showed that the error value was close to 0.02 rad when applying noise and disturbance. In the future, we plan to extend our work to more complex controllers and apply it to robotic systems using SNN hardware.

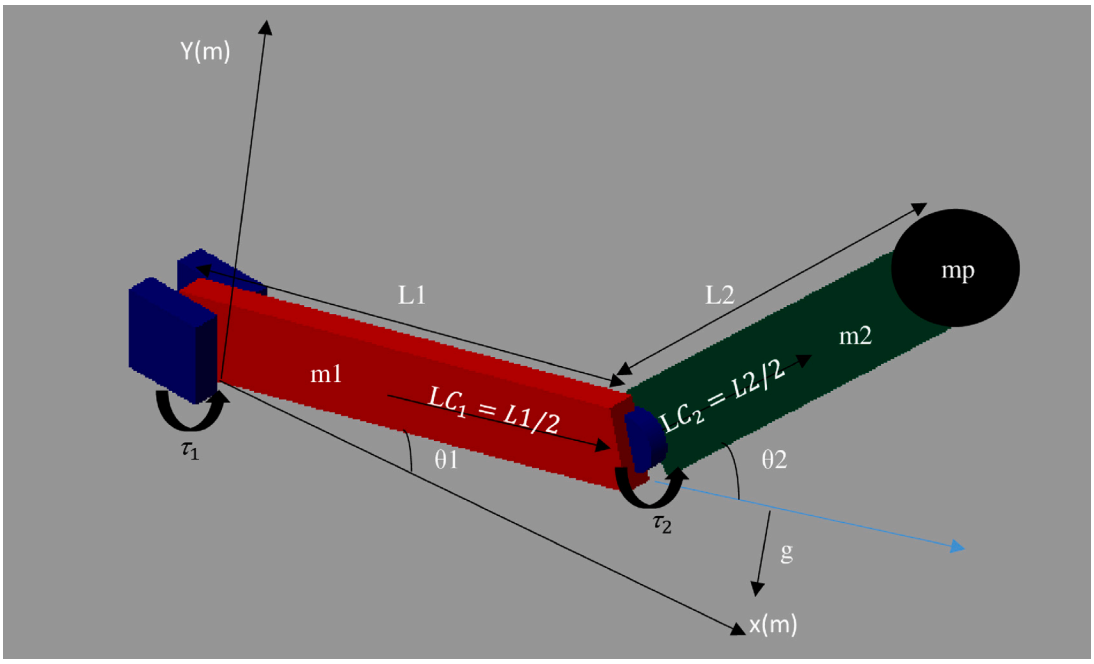


Fig. 50. Mathematical model of a two-link planar rigid robotic manipulator with payload.

CRediT authorship contribution statement

Vahid Azimirad: Conceptualization, Investigation, Methodology, Project administration, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **S. Yaser Khodkam:** Data curation, Formal analysis, Writing – original draft, Writing – review & editing. **Amir Bolouri:** Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Funding

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Appendix A

The dynamic model of a two-link manipulator is as follows (Fig. 50):

$$\tau = M(\ddot{q}) + C(q,\dot{q}) + G(q) + F_{friction}(\dot{q}) \tag{12}$$

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} M11 & M12 \\ M21 & M22 \end{bmatrix} + \begin{bmatrix} -c\dot{\theta}_2 & -c\dot{\theta}_1 - c\dot{\theta}_2 \\ -c\dot{\theta}_1 & 0 \end{bmatrix} + \begin{bmatrix} V_1\dot{\theta}_1 \\ V_2\dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} d_1 sgn(\dot{\theta}_1) \\ d_2 sgn(\dot{\theta}_2) \end{bmatrix} + \begin{bmatrix} G1 \\ G2 \end{bmatrix} \tag{13}$$

$$M(\ddot{q}) = \begin{bmatrix} M11 & M12 \\ M21 & M22 \end{bmatrix} \tag{14}$$

$$M11 = I_1 + I_2 + m_1 L_{C1}^2 + M_2(L1^2 L_{C1}^2 + 2L_1 L_{C2} COS(\theta_2)) + m_p(L_1^2 + L_2^2 + 2L_1 L_2 COS(\theta_2)) \tag{15}$$

$$M12 = I_2 + m_2(L_{C2}^2 + L_1 L_{C2} COS(\theta_2)) + m_p(L_2^2 + L_1 L_2 COS(\theta_2)) \tag{16}$$

$$M12 = M21 \tag{17}$$

$$M22 = I_2 + m_2 L_{C2}^2 + m_p L_2^2 \tag{18}$$

$$C(q,\dot{q}) = \begin{bmatrix} -c\dot{\theta}_2 & -c\dot{\theta}_1 - c\dot{\theta}_2 \\ -c\dot{\theta}_1 & 0 \end{bmatrix} \tag{19}$$

$$c = m_2 L_1 L_{C2} Sin(\theta_2) \tag{20}$$

$$G(q) = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} = \begin{bmatrix} (m_1 + m_2) L_{C1} g COS\theta_1 + m_2 g(L_{C2} COS(\theta_1 + \theta_2) + L_1 COS(\theta_1)) \\ m_2 L_{c2} g COS(\theta_1 + \theta_2) \end{bmatrix} \tag{21}$$

$$F_{friction} = F_v + F_D \tag{22}$$

$$F_v = \begin{bmatrix} V_1\dot{\theta}_1 \\ V_2\dot{\theta}_2 \end{bmatrix} \& F_D = \begin{bmatrix} d_1 sgn(\dot{\theta}_1) \\ d_2 sgn(\dot{\theta}_2) \end{bmatrix} \tag{23}$$

$$q(q) = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \tag{24}$$

L_1, and L_2 are the length of the $Link_1$, and $Link_2$; L_c1, and L_c2 indicate the distance from joint to center of gravity $Link_1$, and $Link_2$; m_1, and m_2 present mass of $Link_1$, and $Link_2$; v_1, and v_2 are viscous coefficient friction.d_1, and d_2 express dynamics coefficient friction of $Link_1$, and $Link_2$; θ_1, and θ_2 are the position of $Link_1$, and $Link_2$; τ_1,

and τ_2 are the torques given by the control system to joints 1 and 2, respectively. m_p is the mass of payload, and value is 0.5 kg (see Fig. 50).

References

Abed, A. M., A., R. Z. N. A. F. Z. S. R. S. M. A. M. J. A. J., & k. A., A. (2022). Trajectory tracking of differential drive mobile robots using fractional-order proportional-integral-derivative controller design tuned by an enhanced fruit fly optimization. *Measurement and Control*, *55*(3–4), 209–226.

Azimirad, V., & M., F. S. (2020). Experimental study of reinforcement learning in mobile robots through spiking architecture of Thalamo-Cortico-Thalamic circuitry of mammalian brain. *Robotica*, *38*(9), 1558–1575.

Azimirad, V., M., R., S., T., V., S., & F., J.-S. (2022). A consecutive hybrid spiking-convolutional (CHSC) neural controller for sequential decision-making in robots. *Neurocomputing*, *40*, 319–336.

Azimirad, V., S., S. V., & N., A. (2021). Vision-based learning: A novel machine learning method based on convolutional neural networks and spiking neural networks. In *9th RSI international conference on robotics and mechatronics* (pp. 192–197).

Cao, J. Y., & G., L. J. C. B. (2005). Optimization of fractional order PID controllers based on genetic algorithms. In *International conference on machine learning and cybernetics, vol. 9* (pp. 5686–5689).

Cao, J. Y., & G., C. B. (2006). Design of fractional order controllers based on particle swarm optimization. In *1st IEEE conference on industrial electronics and applications* (pp. 1–6).

Das, S., D., S., & G., A. (2011). Fractional order modeling of a PHWR under step-back condition and control of its global power with a robust $PI^{\lambda}D^{\mu}$ controller. *IEEE Transactions on Nuclear Science*, *58*(5), 2431–2441.

Das, S., S., S., & D., S. (2011). On the selection of tuning methodology of FOPID controllers for the control of higher order processes. *ISA Transactions*, *50*(3), 376–388.

Feliu-Batlle, V., R.-P., R., & C.-G., F. (2009). Fractional order controller robust to time delay variations for water distribution in an irrigation main canal pool. *Computers and Electronics in Agriculture*, *69*(2), 185–197.

Hamamci, S. E. (2007). An algorithm for stabilization of fractional-order time delay systems using fractional-order PID controllers. *IEEE Transactions on Automatic Control*, *52*(10), 1964–1969.

Hoang, Q., J., P., & S., L. (2021). Combined feedback linearization and sliding mode control for vibration suppression of a robotic excavator on an elastic foundation. *Vibration and Control*, *27*(3–4), 251–263.

Hsu, C.-H., C., S.-J., C., T.-J., F., Y.-M. H. and C.-P., & C., S.-F. (2022). Low-cost and high-efficiency electromechanical integration for smart factories of IoT with CNN and FOPID controller design under the impact of COVID-19. *Applied Sciences*, *12*(7), 3231.

Iakymchuk, Taras, R.-M., A., F.G.-M., J., B.-M., M., V., J., & V., F. (2015). Simplified spiking neural network architecture and STDP learning algorithm applied to image classification. *EURASIP Journal on Image and Video Processing*.

Izhikevich, M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, *14*(6), 1569–1572.

Jun Hu, H., K., T., T., C., L., H., & S., L. (2013). A spike-timing-based integrated model for pattern recognition. *Neural Computation*, *25*(2), 450–472.

Kartoun, Uri, Stern, Helman, & Edan, Yael (2010). *A human-robot collaborative reinforcement learning algorithm, intelligent, and robotic systems, vol 60*.

Kasabov, Nikola K., B., H., D., M., & W., A. (2023). Brain-inspired spatio-temporal associative memories for neuroimaging data classification: EEG and fMRI. *Bioengineering*, *10*(12), 450–472.

Kommula, Bapayya Naidu, & K., V. R. (2022). Design of MFA-PSO based fractional order PID controller for effective torque controlled BLDC motor. *Sustainable Energy Technologies and Assessments*, *49*.

Kormushev, Petar, C., S., & C., D. G. (2013). Neural network reinforcement learning for visual control of robot manipulators. *Expert Systems with Applications*, *40*(5), 1721–1736.

L., A., & Yu, H. (2020). Smooth-switching control of robot-based permanent-magnet synchronous motors via port-controlled Hamiltonian and feedback linearization. *Energies*, *13*(21), 5731.

Lee, Y.-S., & D.-W., J. (2021). Optimization of neural network-based self-tuning PID controllers for second order mechanical systems. *Applied Sciences*, *11*(17).

Lee, C.-Y., & Lee, J.-J. (2003). Adaptive control of robot manipulators using multiple neural networks. In *IEEE international conference on robotics and automation* (pp. 1074–1079).

Lee, C, P., P., G., S., & K., R. (2018). Training deep spiking convolutional neural networks with STDP-based unsupervised pre-training followed by supervised fine-tuning. *Frontiers in Neuroscience*, *12*.

Liu, L., D., D., C., G., S., R., & D., R. (2020). Robot navigation in crowded environments using deep reinforcement learning. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 5671–5677).

Liu, Dingbang, Y., H., & C., Y. (2020). Low-power computing with neuromorphic engineering. *Advanced Intelligence Systems*, *3*(2).

Mehndiratta, M., E., K., M., R., & E., K. (2019). Robust tracking control of aerial robots via a simple learning strategy-based feedback linearization. *EEE Access*, *8*, 1653–1669.

Miljković, Zoran, M., M., L., M., & B., B. (2013). Reinforcement learning in robotics: Applications and real-world challenges, robotics. *Robotics*, *2*(3), 122–148.

Mohammad Saleh Tavazoei, M. H. (2009). A note on the stability of fractional order systems. *Mathematics and Computers in Simulation*, *79*(5), 1566–1576.

Muftah, M. N., & M., F. A. A. M. S. S. S. (2022). Modeling and fuzzy FOPID controller tuned by PSO for pneumatic positioning system. *Energies*, *10*(15), 3757.

Muresan, Cristina I., D., E. H., & B., R. (2015). A novel tuning algorithm for fractional order IMC controllers for time delay processes. *Mechanical Engineering and Robotics Research*, *4*(3), 218–221.

Ning, Limiao, D., J., X., R., T., K. C., & T., H. (2023). Event-driven spiking neural networks with spike-based learning. *Memetic Computing*, *15*, 205–217.

Pane, Yudha P., N., S. P., & B., J. K. R. (2019). Reinforcement learning based compensation methods for robot manipulators. *Engineering Applications of Artificial Intelligence*, *78*, 236–247.

Petras, I. (2009). Stability of fractional-order systems with rational orders. *Dynamical Systems (Math. DS)*, *12*(3).

Petrovic, E., V., N., I., I., M., S., & S., P. (2016). Kinematic model and control of mobile robot for trajectorytracking. *Computer Science, Engineering*.

Pritesh Shah, S. A. (2016). Review of fractional PID controller. *Mechatronics*, *38*, 29–41.

Quan, H, Y., L., & Y., Z. (2020). A novel mobile robot navigation method based on deep reinforcement learning. *International Journal of Advanced Robotic Systems, International Journal of Advanced Robotic Systems*, *17*(3), 1289–1307.

Rathi, Nitin, C., I., K., A., S., A., A., A., P., P., et al. (2023). Exploring neuromorphic computing based on spiking neural networks: Algorithms to hardware. *ACM Computing Surveys*, *55*(12), 1–4.

Richa Sharma, P. G., & M., A. (2015). Freedom fractional order PID controllers for robotic manipulator with payload. *ISA Transactions*, *58*, 279–291.

Shafti, Ali, & F., J. T. W. D. A. A. (2020). Real-world human-robot collaborative reinforcement learning. In *IEEE/RSJ international conference on intelligent robots and systems, vol. 2* (pp. 11161–11166).

Shah, P, & S., A. (2013). Design and optimization of fractional PID controller for higher order control system. *Engineering*.

Shalaby, R., B., E.-H. M. A.-Z., & T., M. (2023). Optimal fractional-order PID controller based on fractional-order actor-critic algorithm. *Neural Computing and Applications*, *35*(3), 2347–2380.

Valerio, D., & J., C. (2010). A review of tuning methods for fractional PIDs. In *4th IFAC workshop on fractional differentiation and its applications, FDA, vol. 10, no. 5*.

Vinagre, B. M., & I, M. C. A. C. A. J. S. J. (2007). Fractional PID controllers for industry application. A brief introduction. *Journal of Vibration and Control*, *13*(9–10), 1419–1429.

Webb, A., & S., D. (2011). Spiking neural PID controllers. *Neural Information Processing*, *7064*, 259–267.

Xu, Lin, D., J., S., B., & C., M. (2022). A combined backstepping and fractional-order PID controller to trajectory tracking of mobile robots. *Systems Science & Control Engineering*, *10*(1), 134–141.

Xue, Dingyu, & C., C. Z. A. Y. (2006). Fractional order PID control of a DC-motor with elastic shaft: a case study. In *American control conference*.

Yamada, Englert, J., & J., L. Y. S. G. P. K. P. M. S. G. L. (2021). Motion planner augmented reinforcement learning for robot manipulation in obstructed environments. In *Proceedings of the 2020 conference on robot learning, vol. 155* (pp. 589–603). PMLR.

Yamazaki, L. N., V., K.-H., & D., B. (2022). Spiking neural networks and their applications: A review. *Brain Science*, *12*(7).

Yan, Zhanglu, Z., J., & W., W.-F. (2021). *Energy efficient ECG classification with spiking neural network: vol. 63*.

Zhang, Yun, & L., H. Q. X. L. Y. C. Y. W. M. Z. Z. (2021). A new recursive least squares-based learning algorithm for spiking neurons. *Memetic Computing*, *15*, 110–125.

Zhang, Jiawen, Z., T., G., B., & D., S. (2021). Fuzzy fractional-order PID control for two-wheeled self-balancing robots on inclined road surface. *Systems Science & Control Engineering*, *10*(1), 289–299.

Zhao, Wenshuai, & W., J. P. Q. L. Q. T. (2020). Towards closing the sim-to-real gap in collaborative multi-robot deep reinforcement learning. In *5th international conference on robotics and automation engineering, vol. 1* (pp. 7–12).