ISSN 1870-4069

Spiking Neural Network Implementation of LQR Control on an Underactuated System

Jorge A. Juárez-Lora, Humberto Sossa, Victor H. Ponce-Ponce,
Elsa Rubio-Espino, Ricardo Barrón-Fernández

Instituto Politécnico Nacional,
Centro de Investigación en Computación,
Mexico

{jjuarezl2020, hsossa, vponce, erubio, rbarron}@cic.ipn.mx

Abstract. This writing presents an architecture proposal designed to implement a control loop in a mobile-wheeled under-actuated inverted pendulum system, using spiking neural networks, linear quadratic regulator control technique, and a neural framework that allows us to define the neuron ensembles specification to represent specific control signals.

Keywords: Robotics, neural networks, spiking neural networks, machine learning, neurorobotics, neurocomputing.

1 Introduction

4.0 Industry has brought a massive proliferation of sensors and data acquisition devices for monitoring and analysis purposes. This situation has escalated quickly, as the amount of recollected data overpasses any computation and storage capacities needed to provide information solutions that allow intelligent decision-making using *Big Data* process techniques. Storing all the information is not feasible anymore, so new analysis techniques have appeared, such as artificial neural networks, which have proven to be very useful in online learning scenarios.

Spiking neural networks (SNN), also known as artificial third-generation neural networks, intend to emulate biological plausibility, physical, chemical, and biological mechanisms, allowing computation and Hebbian learning to occur in biological living systems. These models have optimal characteristics for hardware implementation [1, 2].

It promises enormous parallel computing capacity and low energetic usage, enabling feasible online learning platforms to implement size and power restriction applications, such as robotics. Neurorobotics [3, 4] is a discipline that takes as a challenge to design control mechanisms, hardware, and implementation techniques for robotic applications. These agents adapt their behavior up to changes in themselves or the environment dynamics.

This situation can be seen in biological systems with growing limbs or holding a heavy object, alien to its composition, or perhaps, aging, which modifies friction in arm or leg joints. One of the most evident obstacles in neurorobotics development is the shortage of physical platforms for neuromorphic computation. However, there are

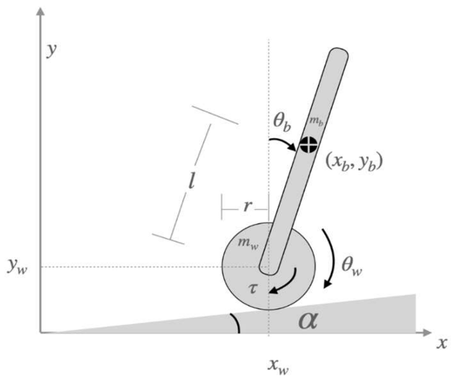

Fig. 1. 2D MWIP model, extracted from [17].

simulation platforms that allow designing the neural structures needed for future implementations.

Some efforts such as *Human Brain Project* [5], which has a neurorobotics platform [6], allow to simulate a brain or physical body and explore how to control movement, stimuli reaction and learn from a virtual or real environment. Another platform, primarily focused on SNN implementation, is called Nengo [7-9], a tool that allows to build and design SNNs architectures. It is quite flexible, as the user can define its neural models, its own learning rules, optimization methods, reuse of subnetworks, data input, and even has libraries for exporting these models to neuromorphic hardware or FPGA implementations.

As a small tour in literature, in [10], an adaptive control method proposed in [11] is used, allowing them to control a three-link arm in simulation, using a spiking neural network structure designed to estimate the inverse jacobian dynamics. Here, authors name part of their proposed neural network as their biological counterpart to match specific tasks made by natural brains. In [12], control of a simulated robotic arm, without path planning, is achieved using SNNs and *motor primitives*. In [13], a biologically inspired spiking neural network (SNN) for soft-grasping to control a robotic hand, used for robots interacting with objects shaped for humans, is presented.

Finally, in [14], a hardware adaptive control implementation of a Kinova Jaco robotic arm using the Loihi platform [15] is introduced. These three examples have something in common; these are complete actuated systems. In this work, an implementation of a linear quadratic regulator (LQR) strategy using SNN is presented as an introductory example of how Nengo and SNN can be used for under-actuated systems, showing which obstacles must be tackled to perform precise control signal representation.

The Mobile Wheeled Inverted Pendulum (MWIP) is an easily controllable system for a human with a bit of practice but a challenge in control theory.

Although some of these under-actuated systems show controllability under linearization around a certain equilibrium point, the control tasks entitle arbitrary output reference trajectory tracking, taking the system state away from the equilibrium point, thus overcoming a traditional obstacle to linearization-based control of nonlinear systems [16].

This document is organized as follows. Section 2 describes the MWIP dynamics and the LQR control technique used for controlling the system. Section 3 describes which methodology was used to create an SNN structure to represent the LQR controller and shows its implementation in Nengo software simulation. Section 4 shows the configuration parameter for the simulation and its results. Finally, section 5 is used for conclusions and proposed future work.

2 Dynamics and Control Strategy of the Robotic System

2.1 Dynamic Model of the System

Fig. 1 shows the graphical representation of the MWIP (Mobile Wheeled Inverted Pendulum). Here, x_w, y_w are the wheel coordinates, x_b, y_b are the mass center coordinates of the bar, α is the plane's angle inclination, $m_b,\ m_w$ stand for the bar and the wheel's mass, respectively, I_b, I_w are the moments of inertia from the bar and the wheel, L is the bar's length, r is the radius of the wheel, and θ_w, θ_b are the states of the system, which stand for the rotation angle of the wheel, and the bar's inclination, in that order.

The robotic system corresponds to a second-order underactuated system [18]. Starting from the modeling dynamics using Euler-Lagrange technique in [17], setting $\alpha = 0$ leads to a system with the following depiction:

$$M(q)\ddot{q} = C(q, \dot{q}, u), \tag{1}$$

where: $M(\cdot)$ is the inertia matrix, $C(\cdot)$ groups *the coriolis, gravity, and control terms,* and the extended form of this equation is:

$$\begin{pmatrix} (m_b + m_w)r^2 + I_w & m_b Lr\, cos(\theta_b) \\ m_b Lr\ \cos(\theta_b) & m_b L^2 + I_b \end{pmatrix} \begin{pmatrix} \ddot{\theta_w} \\ \ddot{\theta_b} \end{pmatrix} = \begin{pmatrix} u + m_b Lr\, \dot{\theta}_b^2\ \sin(\theta_b) \\ -u + m_b g\ L\ \sin(\theta_b) \end{pmatrix}. \tag{2}$$

Here, u is the control signal. It is worth mentioning the motor is mounted in the hub that connects the wheel and bar, so the used torque is the same but in the opposite direction. To obtain the accelerations of the system, we rewrite eq. (2) as:

$$\ddot{q} = M(q)^{-1} \cdot c(\dot{q}, q). \tag{3}$$

Both second-order differential equations can be represented in four first-order equations, this is, rewriting the system in a space state manner, such as $\ddot{x} = f(q, \dot{q}, u)$. Setting $x = [x_1, x_2, x_3, x_4] = [\theta_w, \dot{\theta}_w, \theta_b, \dot{\theta}_b]$, the system can be linearized in its equilibrium states $\theta_b = \dot{\theta}_w = \dot{\theta}_b = 0$, which is equivalent to a pendulum in an upright position. Therefore, the linearized system results in the form $\dot{x} = Ax + Bu$ with the matrices A, B given by:

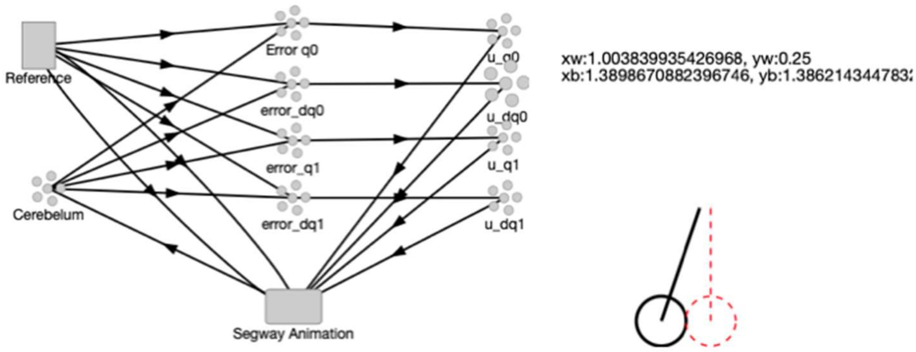

Fig. 2. SNN structure proposed for the control simulation.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{-gL^2 m_b^2 r}{z} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\dfrac{gLm_b^2 r^2 + gLm_w m_b r^2 + I_w gLm_b}{z} & 0 \end{bmatrix}, \tag{4}$$

$$B = \begin{bmatrix} 0 \\ \dfrac{m_b L^2 - m_b rL + I_b}{z} \\ 0 \\ -\dfrac{I_w + m_b r^2 + m_w r^2 - Lm_b r}{z} \end{bmatrix}. \tag{5}$$

With:

$$z = I_b I_w + I_w L^2 m_b + I_b m_b r^2 + I_b m_w r^2 + L^2 m_b m_w r^2. \tag{6}$$

It can be easily shown that the system is fully controllable and observable.

2.2 Linear Quadratic Regulator (LQR)

The goal then is to move the vector state x from an initial condition to a desired vector x_d. Finding a control law $u = -K_r(x - x_d)$ for the MWIP system, which moves the closed-loop eigenvalues of the system as far as required on the left half of the complex plane, in order to achieve optimal control, is a task of optimization.

Choosing over stable eigenvalues might cause the system to overreact to small perturbations or noise or require high control signals, which might overpass the actuator's capacity. On the other side, choosing eigenvalues as close as possible to the right half in the complex plane might result in long stabilization times and small control signals, leading to instability.

The Linear Quadratic Regulator [19] (LQR hereinafter) consists in to deliver a full-state feedback control method that minimizes the following cost function:

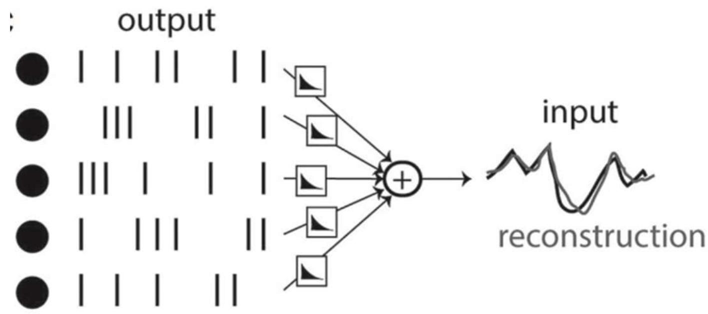

Fig. 3. Spike-based sparse coding. A reconstruction of the signal is obtained from combining filtered spike trains together, and spikes are timed to make the reconstruction accurate. Extracted from [26].

Table 1. MWIP model parameters used for simulation.

Parameter	Value
m_b	$1[kg]$
m_w	$2[kg]$
L	$1.2[m]$
r	$0.25[m]$
I_w	$10[N/m]$
I_b	$10[N/m]$

$$J(t) = \left(\int_0^\infty (x - x_d)^T Q(x - x_d) + u^T R u\right) dt. \tag{4}$$

The function in (4) pictures a balance between the energetic cost of an effective state regulation, which is intended to be minimized, and a quicker control response, which is intended to be fast.

These objectives are regulated by $Q = Q^T \geq 0$ and $R = R^T \geq 0$ respectively, and they can be selected as wished to prioritize control objectives. As bigger is Q, it will move the system to the desired vector state as soon as possible. As big as R might be, lower control signals will be the priority, while $J = \lim_{t\to\infty} x(t) = 0$.

As $J(\cdot)$ is a quadratic function, there exists an analytic solution for control weights in K_r given by:

$$K_r = R^{-1}BP, \tag{5}$$

where P is the Ricatti's algebraic equation solution:

$$PA + AP^T - PBR^{-1}B^TP + Q = 0. \tag{6}$$

In order to solve eq. (6) there are several software-implemented methods [20, 21], which start from a known A, B for a $\ddot{x} = f(q, \dot{q}, u)$ system dynamics.

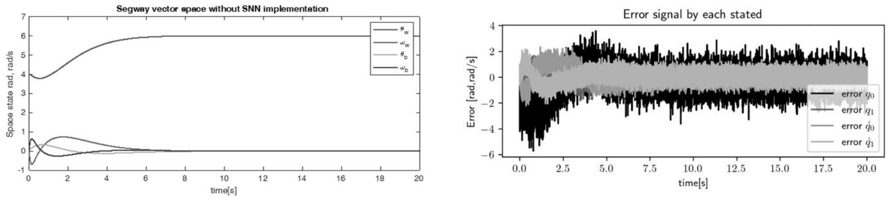

Fig. 4. Evolution of the Vector State (left: simple control loop simulation without SNN, right: using the proposed SNN structure).

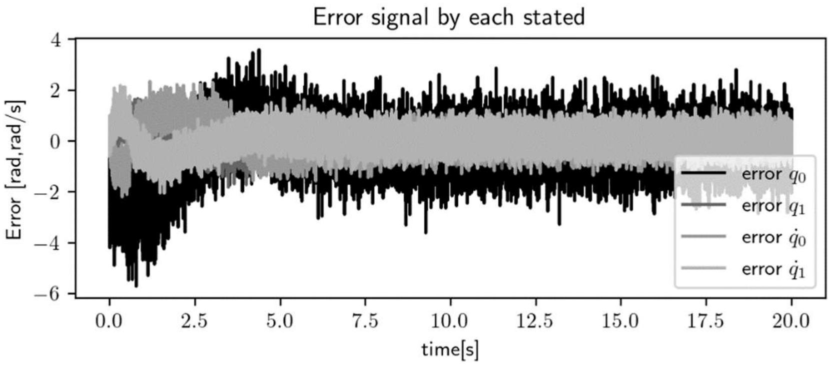

Fig. 5. Evolution of error signal for each state, simulating the proposed SNN structure.

3 Neural Network Simulation

In order to design and implement the neural network, the principles developed in *Neural Engineering Framework (NEF)* [27, 28] are used. NEF can be seen as a *'neural compiler'* that guarantees an optimal global approximation of the defined dynamic equations by the user-defined groups of neurons called ensembles. Given a signal, it is possible to use a nonlinear encoding matrix E to parse it into a spike domain.

Then, recover an approximation of the original signal through a linear decoding matrix D, obtaining then the synaptic weights $W = ED$ for the ensemble. Once connected, the resulting network approximates the ideal signal according to neural heterogeneity, stochasticity, and connectivity, affecting its performance.

This is called Spike-based sparse coding, and it can be seen in Fig. 3 [26] Fig. 2 shows the implemented SNN using the simulation software Nengo [22], which provides libraries for define and connect ensembles, which once connected between them, Nengo will find the appropriate synaptic weights using a learning rule (such as STDP) to represent any stimuli value proportioned, and represent it on the next ensemble.

This is achieved using the methodology described in NEF and BCM theory [23,24], which describes how synaptic plasticity on cortical neurons is stabilized by the average postsynaptic activity. All the neurons in this proposed model use *Leaky Integrate and Fire (LIF)* [25] model for its representation.

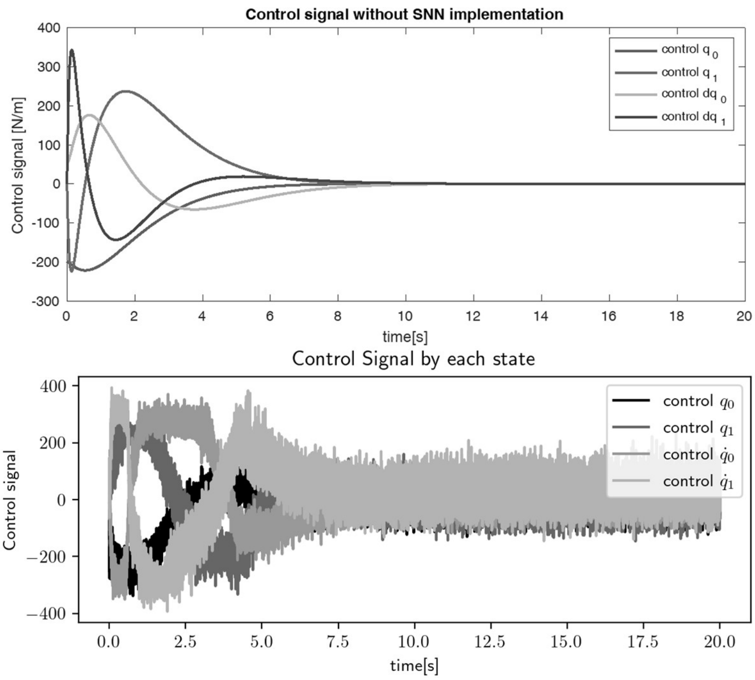

Fig. 6. Evolution of control signal for each state (Up, simple control loop simulation without SNN, down, simulating the proposed SNN structure).

As part of the encoding, each ensemble can encode a signal from a minimum to a maximum value. Nengo allows the user to modify this parameter in the ensemble construction, in order to define the represented function domain. This domain is called radius. By default, this radius is set between [-1, 1].

However, as soon we reset to a bigger domain, the number of neurons in the ensemble needs to be implemented. If this is not the case, the output signal will lose resolution, creating noisy output signals. A small radius then implies better precision. Up next, each of the elements of the proposed structure in Fig. 2 is described:

– Reference: A small function that returns the desired vector state x_d.
– Cerebellum: 1000×4 neuron ensemble, which encodes the state from the MWIP. Radius (-10, 10). Similarly called like this as [10].
– Error: 100×4 neuron ensemble encodes the signal reference and computes the difference between the actual and the reference vector state. Radius (-3.1416, 3.1416).
– Control U: 1000×4 neuron ensemble, which computes the control signal for each state variable. Range (-350,350).
– MWIP Animation: Node used for MWIP simulation, with one control input (u) and four outputs ($\theta_w, \dot{\theta}_w, \theta_b, \dot{\theta}_b$).

The ensemble's radius was selected according to the function domain for each state variable represented. Nevertheless, some states such as angular velocities $\dot{\theta}_w$, $\dot{\theta}_b$ have virtually open domains, while in practice, these are limited. Another example is the computed control signal. Appropriate radii were selected to achieve the maximal values required for the given initial conditions.

4 Simulation Scenario

4.1 Simulation Parameters

In order to evaluate the architecture performance, the MWIP starts from an initial state $x_i = [4,0,0.1,0]$, with a desired vector state $x_d = [6,0,0,0]$. Table 1 shows the MWIP model parameters used and $g = 9.81 m/s^2$. For the control loop, $u = -K(x - x_d)$, setting $Q = I$ and $R = 0.001$, the matrix $K = [-100, -323.3434, -542.0927, -541.08]$, using the described methods in [20, 21]. The simulation period has a duration of $t = 20s$ with a step integration of $1\ ms$.

4.2 Results

Fig. 4 shows the MWIP space state evolution. We can see the initial values evolve towards the desired vector state successfully, with a smooth transition and finishing with a relatively small error, as shown in Fig. 5. It is worth mentioning the stochastic and noisy nature of the control signals (See Fig. 6), due to spike-based sparse coding, oscillate but achieve to represent the desired control signal.

5 Conclusions and Future Work

During the development of this work, a neural architecture based in a biologically plausible model can be used to emulate system dynamics and control implementation. It has been shown how to declare control structures and implement them successfully in under-actuated robotic systems. It is shown that a correct radius specification for each ensemble reflects the precision of its output control signal.

However, while this control strategy reaches a desired vector state, the produced control signals seem to be noisy and stochastic. It reminds those produced by control strategies such as sliding modes [17]. While noise added by the neural dynamics might be problematic, it adds a small value, avoiding singular matrices during encoding and decoding processes used in NEF [7].

This also might be beneficial to prevent an overfitting case. We consider possible future works, like a *Kalman* filter implementation in SNN networks for signal cleaning, or online dynamics estimation of the plant to compute the control signal, exploring other control techniques such as ADRC [29] for unknown plant dynamics and its possible neuromorphic implementation in FPGA or ASIC devices.

Acknowledgments. This work was supported by the Instituto Politécnico Nacional (IPN) and Secretaría de Investigación y Posgrado (SIP-IPN) under the projects: SIP20200630, SIP20201397, SIP20200885, SIP20210788, SIP20210124, and SIP20211657, Comisión de Operación y Fomento de Actividades Académicas

(COFAA-IPN), also the Consejo Nacional de Ciencia y Tecnología (CONACYT-México) under projects 65 (Frontiers of Science 2015) and 6005 (FORDECYT-PRONACES). Juárez-Lora would like to thank to CONACYT for the grant proportioned for his PhD studies.

References

1. Yan, Y., Stewart, T., Choo, X., Vogginger, B., Partzsch, J., Hoeppner, S., Kelber, F., Eliasmith, C., Furber, S., Mayr, C.: Comparing Loihi with a SpiNNaker 2 prototype on low-latency keyword spotting and adaptive robotic control. Neuromorphic Computing and Engineering, vol. 1, no. 1 (2021) doi: 10.1088/ 2634-4386/abf150
2. Lobo, J. L., Del Ser, J., Bifet, A., Kasabov, N.: Spiking neural networks and online learning: An overview and perspectives. Neural Networks, vol. 121, pp. 88–100 (2020) doi: 10.1016/j.neunet.2019.09.004
3. Galindo, S., Toharia, P., Robles, Ó., Ros, E., Pastor, L., Garrido, J.: Simulation, visualization and analysis tools for pattern recognition assessment with spiking neuronal networks. Neurocomputing. vol. 400, pp. 309–321 (2020)
4. Bogdan, P., Marcinnò, B., Casellato, C., Casali, S., Rowley, A., Hopkins, M., Leporati, F., D'Angelo, E., Rhodes, O.: Towards a bio-inspired real-time neuromorphic cerebellum. Frontiers in Cellular Neuroscience, vol. 15, pp. 130 (2021)
5. Amunts, K., Knoll, A. C., Lippert, T., Pennartz, C., Ryvlin, P., Destexhe, A., Jirsa, V. K., D'Angelo, E., Bjaalie, J. G.: The human brain project-synergy between neuroscience, computing, informatics, and brain-inspired technologies. Public Library of Science Biology, vol. 17, no. 7 (2019) doi: 10.1371/journal .pbio.3000344
6. Falotico, E., Vannucci, L., Ambrosano, A., Albanese, U., Ulbrich, S., Vasquez-Tieck, J. C., Hinkel, G., Kaiser, J., Peric, I., Denninger, O., Cauli, N., Kirtay, M., Roennau, A., Klinker, G., von Armin, A., Guyot, L., Peppicelli, D., Martínez-Cañada, P., Ros, E., Maiers, P.: Connecting artificial brains to robots in a comprehensive simulation framework: The neurorobotics platform. Frontiers in Neurorobotics, vol. 11, no. 2 (2017) doi: 10.3389/fnbot.2017.00002
7. Voelker A. R., Eliasmith C.: Programming neuromorphics using the neural engineering framework. In: Thakor N.V. (eds) Handbook of Neuroengineering (2021) doi: 10.1007/978-981-15-2848-4_115-1
8. Rasmussen, D.: NengoDL: Combining deep learning and neuromorphic modelling methods. Neuroinformatics. vol. 17 (2019) doi: 10.1007/s12021-019-09424-z
9. The Nengo neural simulator-Nengo. Available at https://www.nengo.ai/
10. DeWolf, T., Stewart, T., Slotine, J. J., Eliasmith, C.: A spiking neural model of adaptive arm control. In: Proceedings of the Royal Society: Biological Sciences, vol. 283, no. 1843 (2016) doi: 10.1098/rspb.2016.2134
11. Cheah, C. C., Liu, C., Slotine, J. J.: Adaptive tracking control for robots with unknown kinematic and dynamic properties. The International Journal of Robotics Research, vol. 25, no. 3, pp. 283–296 (2006)
12. Tieck, J., Steffen, L., Kaiser, J., Roennau, A., Dillmann, R.: Controlling a robot arm for target reaching without planning using spiking neurons. In: IEEE 17th International Conference on Cognitive Informatics Cognitive Computing, pp. 111–116 (2018) doi: 10.1109/ICCI-CC.2018.8482049
13. Tieck, J., Secker, K., Kaiser, J., Roennau, A., Dillmann, R.: Soft-Grasping with an anthropomorphic robotic hand using spiking neurons. Robotics and Automation Letters, vol. 6, no. 2, pp. 2894–2901 (2021) doi: 10.1109/LRA.20 20.3034067

14. DeWolf, T., Jaworski, P., Eliasmith, C.: Nengo and Low-Power AI Hardware for Robust, Embedded Neurorobotics. Frontiers in Neurorobotics, vol. 14. (2020) doi: 10.3389/fnbot.2020.568359
15. Davies, M., Wild, A., Orchard, G., Sandamirskaya, Y., Guerra, G., Joshi, P., Plank, P., Risbud, S.: Advancing neuromorphic computing with Loihi: A survey of results and Outlook. In: Proceedings of the IEEE, vol. 109, no. 5, vol. 911–934 (2021) doi: 10.1109/JPROC.2021.3067593
16. Sira-Ramírez, H.: Active disturbance rejection control of dynamic systems: a flatness-based approach. Oxford: Butterworth-Heinemann (2017)
17. Ri, S., Huang, J., Wang, Y., Kim, M., An, S.: Terminal sliding mode control of mobile wheeled inverted pendulum system with nonlinear disturbance observer. Mathematical Problems in Engineering, vol. 2014, pp. 1–8 (2014) doi: 10.1155/2014/284216
18. Krafes, S., Zakaria, C., Saka, A.: A Review on the control of second order underactuated mechanical systems. Complexity, pp. 1–17 (2018) doi: 10.1155/20 18/9573514
19. Brunton, S., Kutz, J.: Linear control theory. Data-Driven Science and Engineering, Cambridge University Press, pp. 276–320 (2019) doi: 10.1017/9781108380 690.009
20. Benner, P., Li, J. R., Penzl, T.: Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems. Numerical Linear Algebra with Applications, vol. 15, no. 9, pp. 755–777 (2008)
21. Arnold, W. F., Laub, A.: Generalized eigen problems algorithms and software for algebraic Ricatti equations. In: Proceedings of the IEEE, vol. 72, no. 12, pp. 1746–1754 (1984)
22. Rasmussen, D. NengoDL: Combining deep learning and neuromorphic modelling methods. Neuroinform vol. 17, pp. 611–628 (2019) doi: 10.1007/s120 21-019-09424-z
23. Izhikevich, E., Desai, N.: Relating STDP to BCM. Neural computation, vol. 15, no. 7, pp. 1511–1523 (2003) doi: 10.1162/089976603321891783
24. Blais B. Shouval H., Cooper L. N.: The role of presynaptic activity in monocular deprivation: Comparison of homosynaptic and heterosynaptic mechanisms. In: Proceedings of the National Academy of Sciences, vol. 96, no. 3, pp. 1083–1087 (1999)
25. Gerstner, W., Kistler, W., Naud, R. Paninski, L.: Neuronal Dynamics. from single neurons to networks and models of cognition (2006)
26. Brette, R.: Philosophy of the Spike: Rate-Based vs. Spike-Based theories of the brain. Frontiers in Systems Neuroscience, vol. 9, no. 151 (2015)
27. Eliasmith C, Anderson, C.: Neural engineering: computation, representation, and dynamics in neurobiological systems. Cambridge, MA: MIT Press (2004)
28. Eliasmith C.: How to build a brain: A neural architecture for biological cognition. Oxford University Press (2013) doi: 10.1093/acprof:oso/9780199794546.001.000
29. Aguilar-Ibanez, C., Sira-Ramirez, H., Acosta, J. Á., Suarez-Castanon, M. S.: An algebraic version of the active disturbance rejection control for second-order flat systems. International Journal of Control, vol. 94, no. 1, pp. 215–222 (2019) doi: 10.1080/00207179.2019.1589651