IJRCS

International Journal of Robotics and Control Systems
Vol. 1, No. 4, 2021, pp. 488-500
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

ASCEE

Spiking PID Control Applied in the Van de Vusse Reaction

Carlos A. Márquez-Vera [a,1], Zaineb Yakoub [b,2], Marco A. Márquez-Vera [c,3,*], Alfian Ma'arif [d,4]

[a] Universidad Veracruzana, Prolongación Venustiano Carranza, Col. Revolución, Poza Rica 93390, Veracruz, Mexico
[b] National Engineering School of Gabès, Rue Omar Ibn El Khattab, Zrig Eddakhlania, Gabès 6029, Tunisia
[c] Polytechnic University of Pachuca, Pachuca-Cd. Sahagún Km 20, Rancho Luna, Zempoala, Hgo., 43830, Mexico
[d] Universitas Ahmad Dahlan, Jl. Kapas No.9, Semaki, Kec. Umbulharjo, Yogyakarta, 55166, Indonesia
[1] carmarquez@uv.mx; [2] yakoubzaineb@yahoo.fr; [3] marquez@upp.edu.mx; [4] alfian.maarif@te.uad.ac.id
* Corresponding Author

ARTICLE INFO

Article history
Received 28 October 2021
Revised 23 November 2021
Accepted 25 November 2021

Keywords
Spiking Neuron;
PID Control;
Chemical Reaction;
Van de Vusse Reaction;
Artificial Neural Networks

ABSTRACT

Artificial neural networks (ANN) can approximate signals and give interesting results in pattern recognition; some works use neural networks for control applications. However, biological neurons do not generate similar signals to the obtained by ANN. The spiking neurons are an interesting topic since they simulate the real behavior depicted by biological neurons. This paper employed a spiking neuron to compute a PID control, which is further applied to the Van de Vusse reaction. This reaction, as the inverse pendulum, is a benchmark used to work with systems that has inverse response producing the output to undershoot. One problem is how to code information that the neuron can interpret and decode the peak generated by the neuron to interpret the neuron's behavior. In this work, a spiking neuron is used to compute a PID control by coding in time the peaks generated by the neuron. The neuron has as synaptic weights the PID gains, and the peak observed in the axon is the coded control signal. The neuron adaptation tries to obtain the necessary weights to generate the peak instant necessary to control the chemical reaction. The simulation results show the possibility of using this kind of neuron for control issues and the possibility of using a spiking neural network to overcome the undershoot obtained due to the inverse response of the chemical reaction.

1. Introduction

To work with nonlinear systems, the soft computing techniques are interesting alternatives; one advantage is their capability of learning using data in a similar way to adaptive systems. Soft computing can be seen as a metaphor for the natural world [1]; in particular, Artificial Neural Networks (ANN) try to mimic the connections among some neurons. The first generation of ANN can be described as some switches interconnected to have certain behavior; one example is to decide if is turned on a machine as a logic circuit can make in automation applications [2]. Once the activation function for a neuron was not limited to give 0 or 1 as in Boolean algebra, it is possible to compute the neuron action according to

the derivative of the error signal. The synaptic weights can be adapted to minimize the error found between the desired output of the neuron and its current response; this is the backpropagation training method. In this case, these kinds of neurons are known as the second generation of ANN [2].

The neurons of the second-generation work in the following way: some input nodes represent the dendrites of a real neuron, they collect the input information, then the inputs are modified using certain gains that are known are synaptic weights, if these gains are big, they produce the excitation of the neuron, and if they are small or negative, then they produce the inhibition of the neuron. Now, inside the neuron, the input information affected by the weights is added, and if they exceed an internal bias, the neuron is activated; the sum computed inside the neuron is similar to the soma in real neurons. Finally, the output signal is carried out through the axon; this idea is made by evaluating the addition of the information using a nonlinear function [3].

When some neurons are organized in parallel, i.e., there exists only one layer of neurons, the ANN is known as a perceptron, this kind of ANN is evaluated quickly, and its training is also fast, the perceptron is a good proposal for image recognition [4]. When they have connected some perceptrons, the ANN is known as a multilayer perceptron or multilayer ANN. In the control issue, they have used two neuron layers. The first one is known as the hidden layer [5]. The more common activation function for the hidden layer is the hyperbolic tangent because they can be handled by positive and negative values. The second later is the output layer that has as many neurons as degrees of freedom of the dynamic system, and its activation function used to be a linear function.

Recently, Deep Learning (DL) has appeared thanks to the ideas of Hinton and his working team [6]. They affront the problem of adapting a multilayer perceptron if there are used more than three layers because the gradient used in the backpropagation vanishes. DL uses ideas different from the ANN. It uses Boltzmann machines [7], auto-encoders [8], or convolution functions [9] for pre-training. When the system has an acceptable behavior, the new structure is quite similar to a classical ANN (second generation). Now a fine-tuning is made using the well-known backpropagation [10].

The DL can be considered into the fourth industrial revolution, even as the third generation of the ANN, but there exists another kind of artificial neuron. In control, issues are not common to find DL applications. With the ideas and functions proposed by Izhikevich [11], the spiking neurons were easy to compute, and a new paradigm appeared. The idea was to obtain activation signals more similar to the real behavior of biological neurons.

Now, some ideas as to building electronic devices capable of computing spiking neural networks have been developed [12]. This paper is shown an application of a PID controller where the gains are computed using spiking neurons. A codification is needed to interpret the spikes in order to obtain a continuous control signal. The classical codifications are in time, taking into consideration when the spike appears [13], or in frequency by averaging the pulses obtained into a time interval.

The system to control is the Van de Vusse reaction. This chemical process presents an inverse response. It is a common situation in chemical and biotechnological processes because the system input appears with a negative sign in all the differential equations that represent the evolution of the concentration in time according to the mathematical model. The Van de Vusse reaction has been treated as a benchmark to propose control strategies for systems with an inverse response or, in its linear version, systems with non-minimum phases [14][15]. Some control techniques have been used for this chemical reaction as the classical proportional, integral, and derivative (PID) control [15], state feedback regarding the unstable zero [16], internal mode control [17], fuzzy control, and its comparison with type-2 fuzzy control [18], and even ANN by approaching a neural model of the chemical reaction [19].

The spiking neural networks have also been used for control applications. To control a robot, reference [20] was developed an evolving spiking neural network. Reference [21] was designed a spiking PID controller for a UAV. To control nonlinear systems, reference [22] is shown a more general structure using a supervised learning stage with an evolutionary algorithm. In this article, a spiking neuron having as synaptic weights the PID gains are used to control the Van de Vusse chemical reaction. In this way, only one neuron is capable of obtaining the control signal. The synaptic weights are obtained using the spike prop algorithm, a similar technique to the backpropagation. An interesting topic is how to reduce the undershoot obtained in the simulation. One way to avoid this situation is by using an anti-windup term in the control signal. This work presented the spiking PID control application using only one spiking neuron to show how it works and to propose alternatives for enhancing the neuron behavior. For example, in [21], three neuron layers were used to compute different instances in the controller.

The research contribution is to update the synaptic weights by using a model reference in a similar way to the reported in [11], where the control signal is approached after training a spiking neuron with the control signal from a linear PID. In this paper is used the difference between the output of the chemical reaction and the output of the model is a reference to compute the control signal, and then the synaptic weights are updated. One problem is the signal coding because the time resolution used to compute the vector time affects the possible values decoded after the neuron computation. On the other hand, the undershoot shown by the Van de Vusse reaction does not appear in the model reference, and without an anti-windup term, it was not possible to reduce this undesirable effect due to the inverse response.

The rest of the paper is organized as follows: first, in Section 2, the spiking neurons are described. Section 3 shows the Van de Vusse reaction. Section 4 presents the training of the spiking neurons to control the chemical reaction. Section 5 shows the results and discussion. Finally, the conclusions are presented in Section 6.

2. Spiking Neurons

To describe the behavior of biological neurons, many biophysical interactions are needed, thus in [11] was shown a bifurcation methodology, and now two differential equations are capable of reproducing the signals obtained from real neurons. These signals show some spikes after a certain internal potential is reached. Fig. 1 shows some spikes in shape obtained when an external stimulus makes a rat respond.

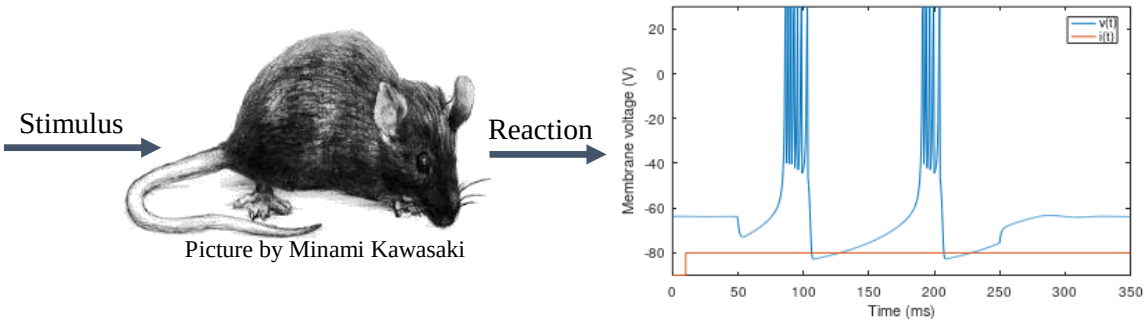

Fig. 1. Spikes are obtained from biological neurons.

The differential equations that can describe the spikes obtained are shown by (1) and (2):

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

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

where $v(t)$ represents the membrane potential of the neuron, $u(t)$ is the potential membrane recovery. These variables are reset when the membrane potential reaches 30mV. This value is approximated the same as biological neurons show. Depending on b value, the internal potential is around -65mV, and according to the membrane potential evolution, the threshold potential is around -50mV.

Once the spike is produced, a reset in the neuron is made, as the graphic in Fig. 1 shows; this reset is computed with the auxiliary condition (3):

$$\text{if } v(t) \geq 30\text{mV, then } \begin{cases} v(t) \leftarrow c, \\ u(t) \leftarrow u(t) + d. \end{cases} \tag{3}$$

An interesting result is shown in [11], where Izhikevich showed the possible spiking shapes when different values of parameters a, b, c, and d are used in the neuron model. Also, a program that simulates 1000 randomly coupled spiking neurons is presented in his paper. Fig. 2 shows a simulation of membrane potential described by (1). It is possible to see the peaks obtained in the simulation. After each peak, a reset is applied, but as the external stimulus $I(t)$ is maintained, the spikes continue.

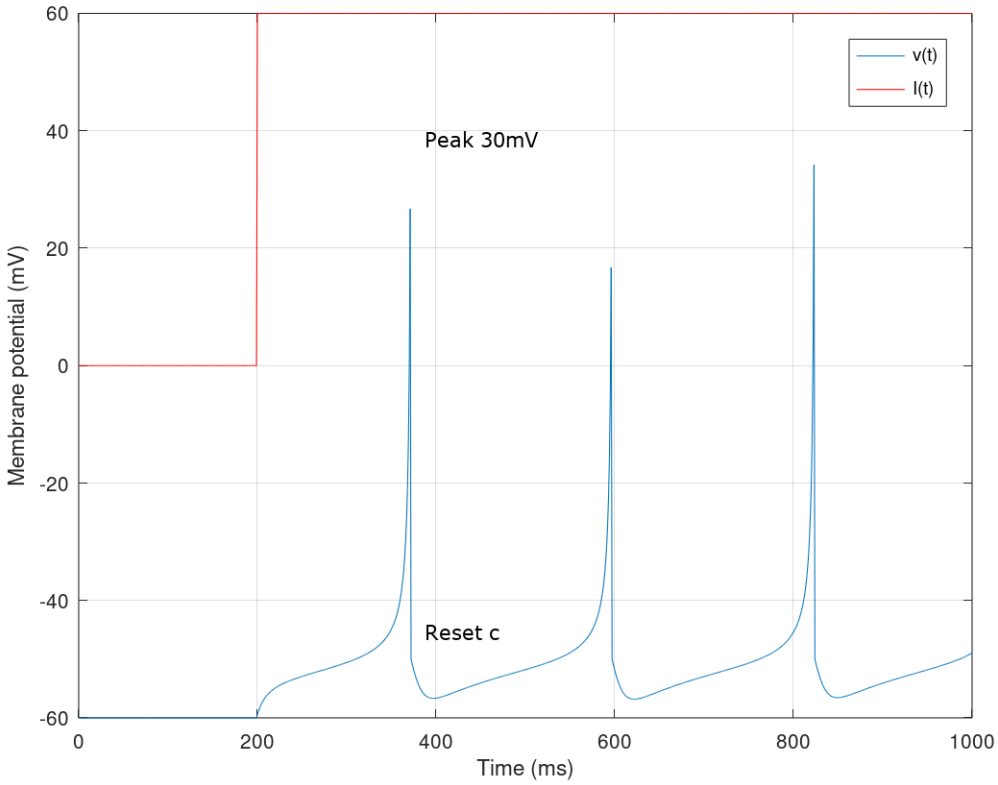

Fig. 2. Spikes of the membrane potential and its resets.

To communicate between neurons, peaks are necessary to simulate the response of a previous neuron. As was aforementioned, there are some techniques to code the information obtained by the peaks given by neurons. In this way, Fig. 3 shows the spiking neuron response when pulses are applied as input. The time when peaks are obtained can be interpreted as the neuron response for situations where more than one peak is generated. Usually, the first peak is only considered.

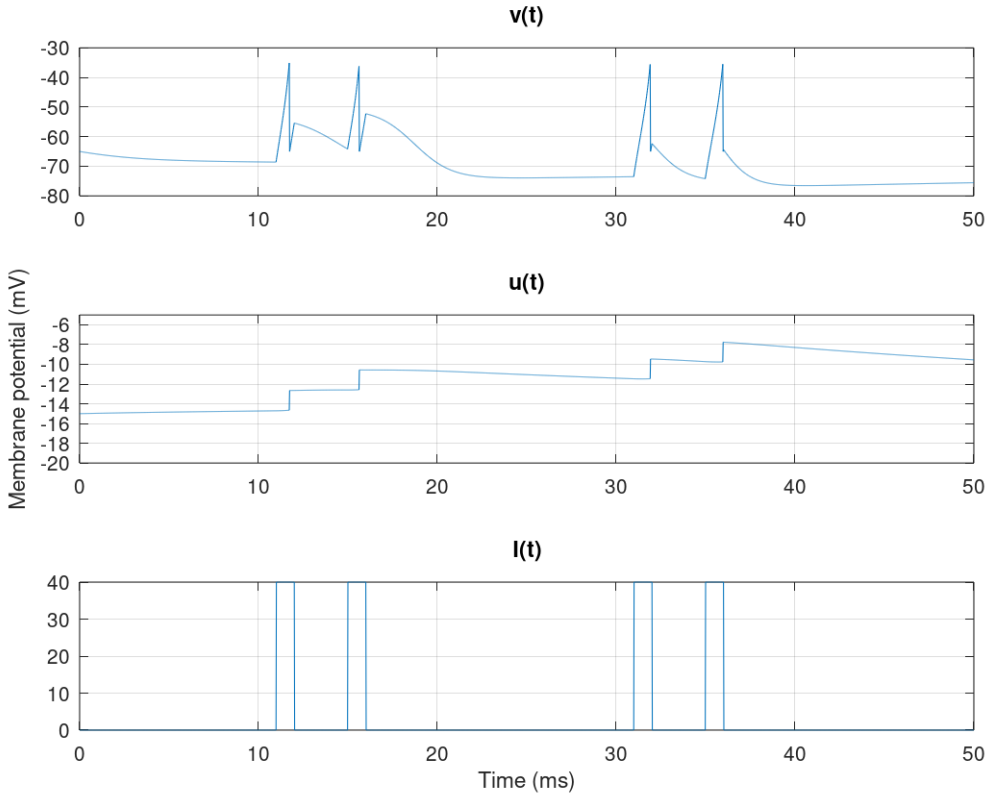

Fig. 3. Membrane potential using pulses as input.

The following step is to find when a peak takes place, and if different neurons are used, each one gives some information regarding the instant of the first pulse it obtained, talking about [11], a simulation or 1000 random connection between spiking neurons is shown in its Fig. 3, this result seems not to be relevant initially, only the fast computing to simulate the peak was important, but when medical researchers saw that graphic, they interpret alpha (10Hz) and gamma (40Hz) rhythms, and when they realized that a random ANN was organized alone, these results were very popular. Fig. 4 shows the simulation of two spiking neurons, and they are presented as the peak instants. Some points as presented are the 1000 points commented in [11].

2.1. Time to First Spike Coding

For control issues, for example, to propose spiking PID, it is possible to assign each component to a certain code. The coding consists in to give a time window, where the peak location corresponds to a certain position into the window that corresponds to a scaled output. First, it is necessary to determine some constants, x_{min} and x_{max} represent the cotes for the input, now t_{min} and t_{max} are cotes for the time window. Finally, to code in time some variables like the error $e(t)$, its derivative named $d_e(t)$ and its integral respect to time $i_e(t)$ is possible to use:

$$t_e = t_{min} + \frac{(e(t) - x_{min})(t_{max} - t_{min})}{(x_{max} - x_{min})}, \tag{4}$$

$$t_{de} = t_{min} + \frac{(d_e(t) - x_{min})(t_{max} - t_{min})}{(x_{max} - x_{min})}, \tag{5}$$

$$t_{ie} = t_{min} + \frac{(i_e(t) - x_{min})(t_{max} - t_{min})}{(x_{max} - x_{min})}. \quad (6)$$

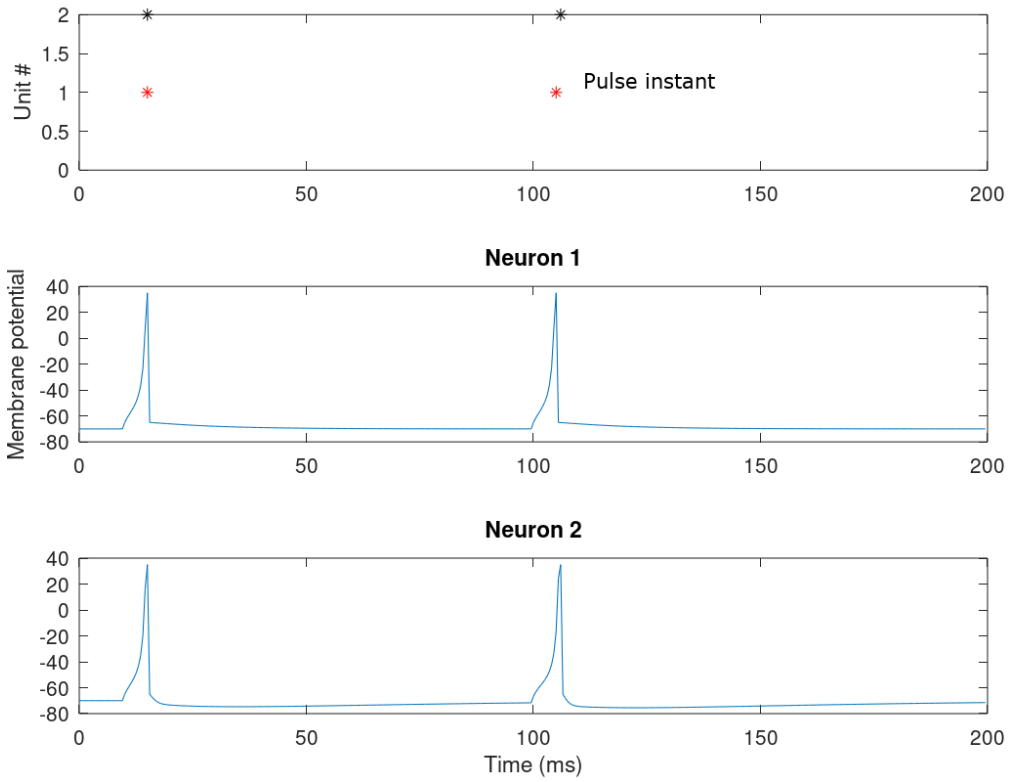

Fig. 4. Peak instants for two spiking neurons.

In order to have pulses when a signal increases, only some kind of spiking signals must be considered. The shape depends on the parameters selected, as was aforementioned. Fig. 2 of [11] shows some of them; in this work are proposed six peaks shapes that are shown in Fig. 5, where also the parameters used to have these responses are presented.

In a similar way, the reference signal can be coded in time (t_d), now the synaptic weights can be computed using a time window and a vector that contains the error coding $\tau = (t_e \quad t_{de} \quad t_{ie})$, the weights are defined by (7) in order to have a smooth function to compute its gradient:

$$w = \frac{\theta T}{N(t_{fin} - \tau)} e^{\frac{t_{fin} - \tau}{T-1}}, \quad (7)$$

where θ, T and N are parameters to be proposed, t_{fin} can be a new t_{max} for a new time window. Therefore, the spiking neuron can be seen as a neuron that has three inputs defined into the vector τ. The weights computing *were* obtained using element by element operations because (7) shows a division and uses an exponential function. These ideas are taken from [23]. Thus a gradient can be computed to update the synaptic weights used in the spiking neuron.

Once the initial weights are obtained, the input vector is obtained using:

$$y(\tau) = \frac{t - \tau}{T} e^{\frac{1-(t-\tau)}{T}}. \quad (7)$$

And the pondered input is $x = wy^T$. In (7), the time t is evaluated until $x >= \theta$. In that case, the cycle breaks, and y is not updated.

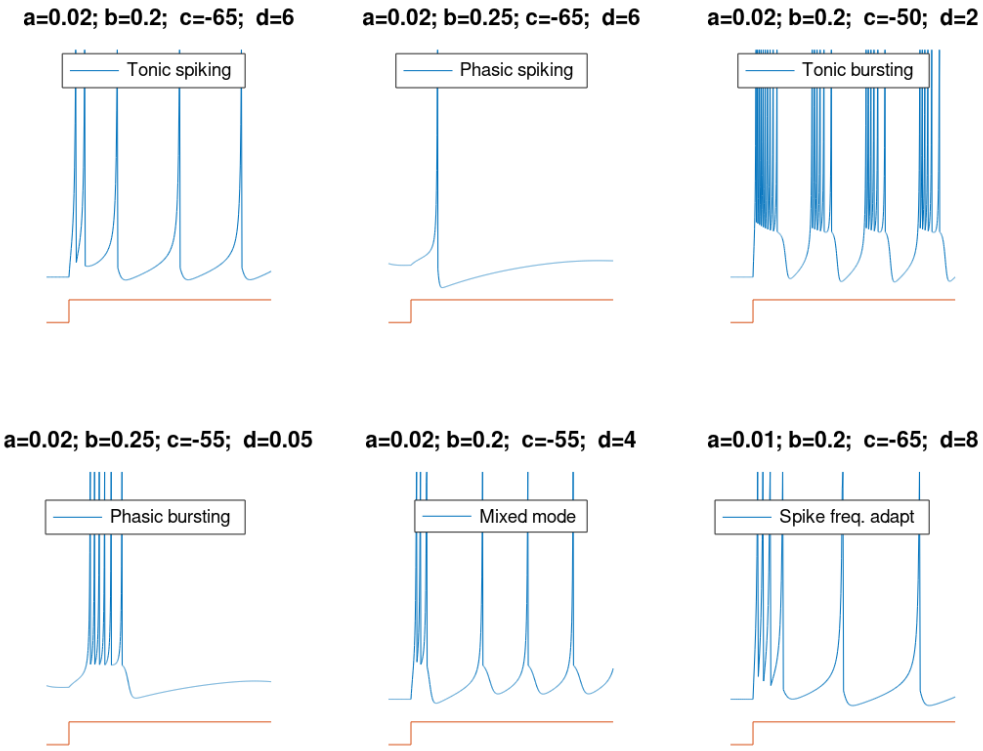

Fig. 5. Peaks shapes are recommended for time coding.

2.2. Weights Updating and Decoding

The function used to describe the weights is differentiable, now using the gradient is computed the change that must be added to the current weights by using (8),

$$\Delta w = \eta y(\tau)\frac{\theta - \tau}{\tau}, \tag{8}$$

and the updating of the weights is $w \leftarrow w + \Delta w$. This is the procedure always that $x < \theta$. Similar to the backpropagation algorithm used in ANN, for SNN, the procedure used to be called spike prop. In case of x exceeds the threshold used, the variation of the weights is made by using:

$$dy = -(1 + t_d - t_{in})\frac{1}{T}e^{\frac{1-t_d-\tau}{T}}, \tag{9}$$

$$\delta = \frac{(t_d - t_{in})}{w}dy, \tag{10}$$

$$\Delta w = \eta y \delta, \tag{11}$$

Being t_{in} the same or a different initial value for a time window, and η is the learning rate. Once the weights were updated. The control signal $u(t)$ can be decoded by proposing a signal interval with the cotes $(u_{min} \quad u_{max})$ and using:

$$u(t) = u_{min} + \frac{t_{in}-t_{min}}{t_{max}-t_{min}}(u_{max} - u_{min}). \tag{12}$$

3. Van de Vusse reaction

A benchmark to work with non-minimum-phase systems because they show an inverse response is the Van de Vusse chemical reaction. The process consists of the production of cyclopentenyl B from cyclopentadiene A by using acid-catalyzed electrophilic addition of water [24]. This reaction is described by (13) and (14)

$$\frac{dC_A(t)}{dt} = \big(C_{Ain} - C_A(t)\big)u(t) - k_1 C_A(t) - k_3 C_A^2(t), \tag{13}$$

$$\frac{dC_B(t)}{dt} = -C_B(t)u(t) + k_1 C_A(t) - k_2 C_B(t), \tag{14}$$

where B is the cyclopentenyl taken as the process output [23]. The process input is the dilution rate $u(t)$ which is the quotient $F(t)/V(t)$, being $F(t)$ the inlet flow and $V(t)$ the liquid volume inside the reactor, this input $u(t)$ use to be used for control simplification in chemical processes [25].

According to [17], this chemical reaction has the following characteristics:

- Input multiplicity, in this way, the Van de Vusse reaction is not controlled affine.
- The reaction has a sign change producing a non-minimum phase behavior.
- Asymmetric response due to nonlinearities in the model.
- A delay due to measuring instruments is often not taken into account.

A scheme that represents the Van de Vusse reaction is shown in Fig. 6, where the PID control is obtained by using a spiking neuron, being the error, its derivative, and its integral coded in time, and they represent peaks as Fig 2.1 in [23].

In this kind of system, the inverse response makes the system reacts in the opposite direction to the input at the beginning of the response, this can be seen as a delay in the response, and anti-windup action can be applied to reduce this undesired effect.

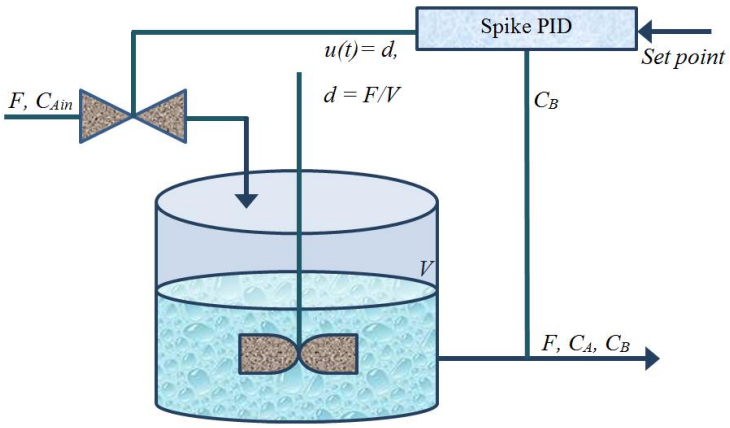

Fig. 6. Van de Vusse reaction scheme.

The control scheme is shown in Fig. 7, where the control gains are updated to have the desired response. Reference [26] were necessary 4ms to adapt the synaptic weights in a leaky integrate and fire spiking model. The application they showed was a haptic interface using quaternion spike neurons.

The spiking control is obtained by computing the PID gains in order to follow a model reference, the spike prop algorithm obtains information from the process output and the model reference, then a change for the synaptic weights is obtained using (8) or (11) according to the threshold value θ, with these new gains the spiking neuron computes the

control signal $u(t)$ using (12), and the signals like the derivative of error or the process output are coded in time to interpret the information into the spiking neuron.

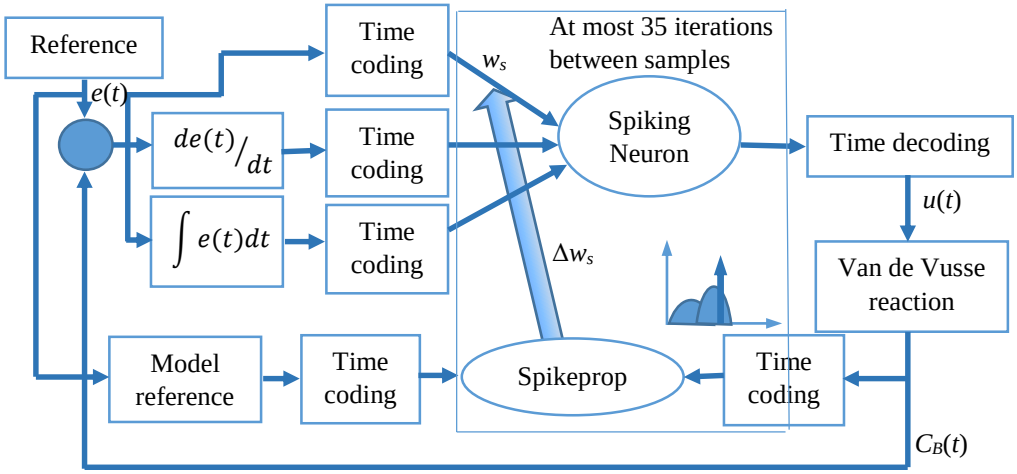

Fig. 7. Spike PID control scheme.

4. Results and Discussion

The spiking PID computed its gains between samples, it was determined as a limit of iterations to train the neuron in 35 iterations, after the simulation, it was obtained an average of 28.387 iterations to compute the PID gains, and the standard deviation was 4.2698 iterations. Fig. 8 plotted the membrane potential evolution through iterations until converge.

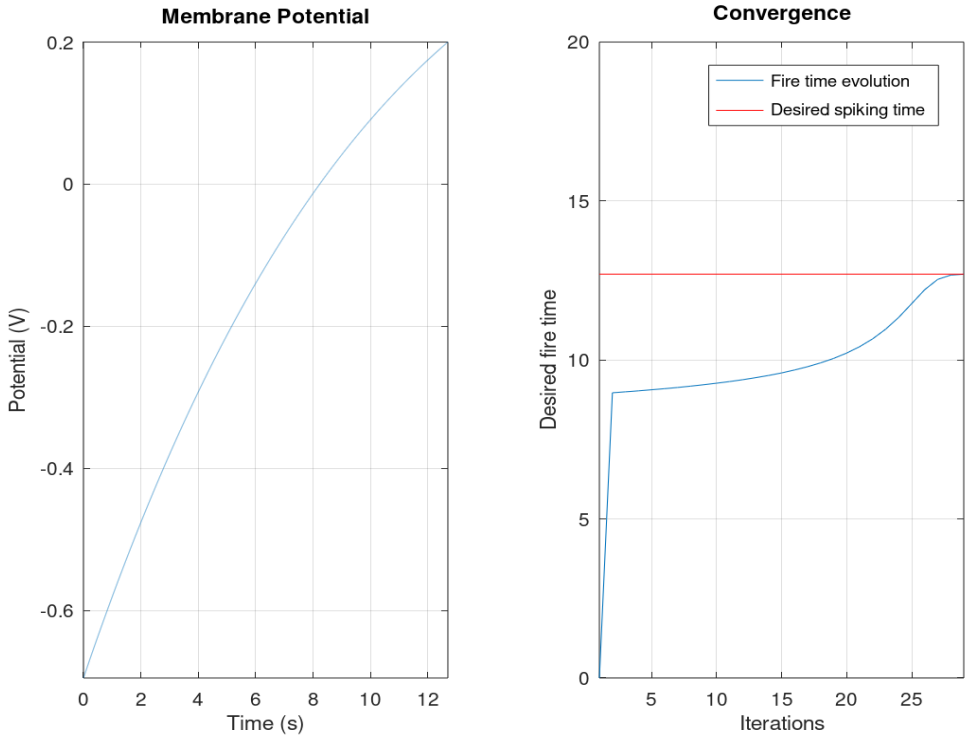

Fig. 8. Iterations convergence between samples.

The sampling time used was 0.02 min, i.e., 1.2 s, the time used to compute 35 iterations was 645ms with a standard deviation of 4.2 ms, the machine used was a computer with 4GB in RAM and a processor of intel core™ 2.3GHz. However, they were necessary around 28.387 iterations to compute the control gains instead of the limit of 35. The control simulation is

shown in Fig. 9, where it is possible to appreciate oscillations at the beginning of the simulation. They are due to the changes obtained in the gains because the spiking neuron tries to obtain a certain PID. As time increase, the gains converge in order to eliminate the steady-state error. The Van de Vusse reaction has an inverse response, and it can be seen that when the oscillations in the simulation are small, the gains are obtained to generate an undesired undershoot. One alternative is to saturate the control gains, the control signal, or even an early stop in the spiking neuron adaptation to avoid the overfitting. In this work, the time window in the peaks coding was used to regulate the undershoot. However, it was necessary to consider if the settling time or the undershoot is the most important characteristic in the simulation. The learning rate η also reduces the settling time but increases the initial oscillations.

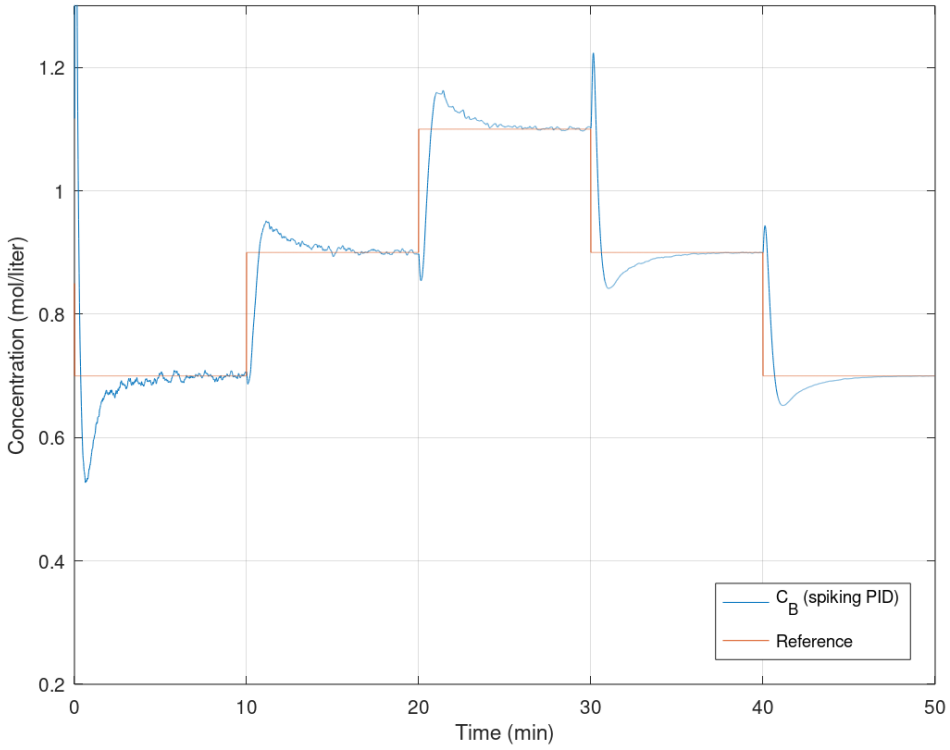

Fig. 9. The Spiking PID control simulation of the Van de Vusse reaction.

The PID gains are computed each sampling time, they are updated at most 35 times between samples, and their evolution is shown in Fig. 10. Table 1 are shown the initial conditions and parameters used in the simulation.

This work was used five synaptic connections were as inputs to the spiking neuron. They are the reference signal, the error as the difference between the process output and the reference signal, their derivative and integral, and a measure about the undershoot in order to adapt the time window. In [17], an interesting result about the control of the Van de Vusse reaction is shown in Fig. 6, where was used a nonlinear control strategy, and the different results were obtained by varying the parameter k_1 in the model equations (13) and (14).

5. Conclusions

Spiking neurons are an interesting topic because they simulate the real behavior depicted by biological neurons. To obtain peaks by stimulating the neurons was easy using the model shown in [11]; now, each response obtained by a real neuron is possible. An important issue is to decode the peaks given as neuron responses. This work was used time coding to interpret the signal obtained from the neuron.

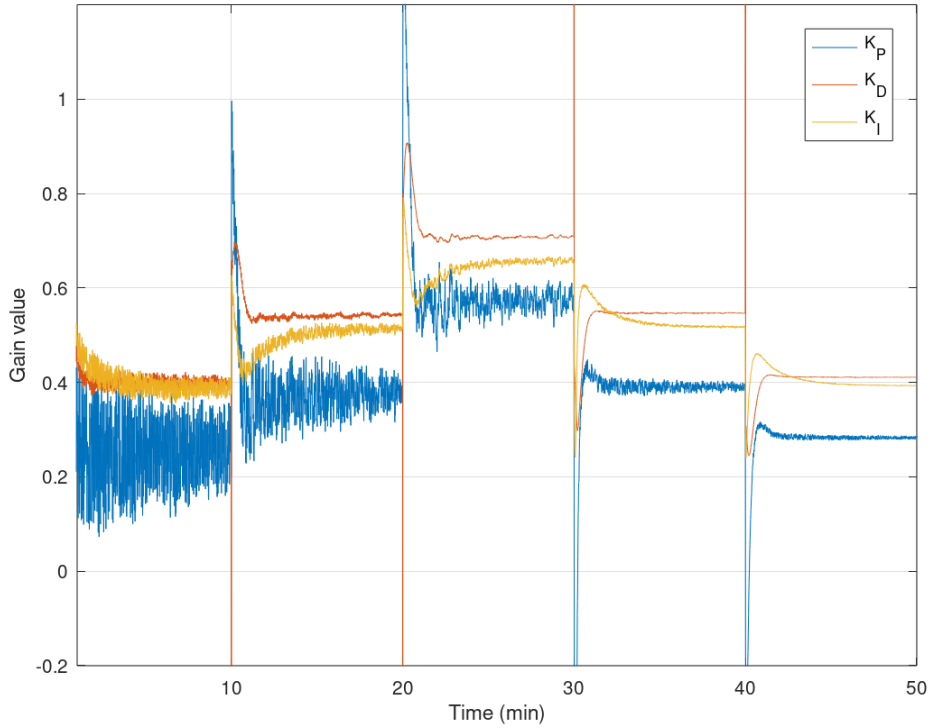

Fig. 10. PID gains evolution (synaptic weights) through time.

Table 1. Process initial conditions

Variable, parameter	Description	
	Value	***Unit***
k_1	5/6	min^{-1}
k_2	5/3	min^{-1}
k_3	1/6	$mol\ min^{-1}$
C_{Ain}	3	$mol\ l^{-1}$
C_A	1.117	$mol\ l^{-1}$
C_B	10	$mol\ l^{-1}$
H	0.002	---
Θ	0.2	---

In a similar way to the backpropagation algorithm used to train ANN, some algorithms use similar ideas as the spike prop and the quick prop. In this paper, the PID gains are proposed by reducing the output error inside a spiking neuron. To adapt the PID gains, which are the synaptic weights of the spiking neuron, a reference model was used to change the closed-loop behavior. A small learning rate can reduce the initial oscillation in the simulation, but sometimes are necessary more iteration for the weight adaptation. Thus the learning rate was selected in order to require less than 35 iterations.

As control application was used, the Van de Vusse reaction, a benchmark used to work with systems with an inverse response due to its non-minimum-phase characteristic. The results are possible to see the control simulation. The main drawbacks are the oscillation obtained until the convergence of the parameters is achieved and the undershooting due to the inverse response must be reduced by aggregating another control technique, for example an anti-windup term.

The main drawback was that the vector time used to code the information has a determined sampling time which determines the resolution of the control signal when it is decoded in the final stage. Future work is planned to use an SNN to process more information in order to have a neural model of the Van de Vusse reaction to propose a new control strategy, using the quick prop as an adaptation algorithm.

Acknowledgment

The authors thank Editor Aninditya Anggari Nuryono from Universitas Gadjah Mada for its support in the review process and for feedback on the necessary comments to improve this paper.

References

[1] S. R. Nandakumar, S. R. Kulkarni, A. V. Babu and B. Rajendran, "Building brain-inspired computing systems: Examining the role of nanoscale devices," *IEEE Nanotechnology Magazine*, vol. 13, no. 2, pp. 19-35, Sep. 2018. https://doi.org/10.1109/MNANO.2018.2845078

[2] J. C. Patterson, Managing a real-time massively-parallel neural architecture, Doctoral Thesis. University of Manchester, School of Computer Science, 232 pages, 2012.

[3] J. Feng and S. Lu, "Performance analysis of various activation functions in artificial neural networks," *J. Phys.: Conf. Series*, vol. 1237, 022030, 2019. https://doi.org/10.1088/1742-6596/1237/2/022030

[4] J. Ou, Y. Li and W. Liu, "TDP: Two-dimensional perceptron for image recognition," *Knowledge-Based Systems*, vol. 195, pp. 105615, May. 2020. http://dx.doi.org/10.1016/j.knosys.2020.105615

[5] Y. V. Tiumentsev and M. V. Egorchev, Chapter 4 - Neural network black box modeling of nonlinear dynamical systems: Aircraft controlled motion. Neural Network Modeling and Identification of Dynamical Systems, Academic Press, pp. 131-163, 2019. http://dx.doi.org/10.1016/B978-0-12-815254-6.00014-9

[6] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature*, vol. 521**,** pp. 436–444, May 2015. https://doi.org/10.1038/nature14539X

[7] F.C. Morabito, M. Campolo, C. Ieracitano and N. Mammone, Chapter 11 - Deep Learning Approaches to Electrophysiological Multivariate Time-Series Analysis, Artificial Intelligence in the Age of Neural Networks and Brain Computing, Academic Press, pp. 219-243, 2019. https://doi.org/10.1016/B978-0-12-815480-9.00011-6

[8] G. Liu, H. Bao and B. Han, "A stacked autoencoder-based deep neural network for achieving gearbox fault diagnosis," *Mathematical Problems in Engineering, Hindawi*, vol. 2018, article 5105709, 10 pages, July 2018. https://doi.org/10.1155/2018/5105709

[9] S. Indolia, A. K. Goswamim, S. P. Mishra and P. Asopa, "Conceptual understanding of convolutional neural network - A deep learning approach," *Procedia Computer Science*, vol. 132, pp. 679-688, 2018. https://doi.org/10.1016/j.procs.2018.05.069

[10] B. J. Wythoff, "Backpropagation neural networks: A tutorial," *Chemometrics and Intelligent Laboratory Systems*, vol. 18, pp. 115-155, 1993. https://doi.org/10.1016/0169-7439(93)80052-J

[11] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569-1572, Nov. 2003. https://doi.org/10.1109/TNN.2003.820440

[12] M. S. Asghar, S. Arslan and H. Kim, "A low-power spiking neural network chip based on a compact LIF neuron and binary exponential charge injector synapse circuits," *Sensors*, vol. 21, 4462, 17 pages, June 2021. https://doi.org/10.3390/s21134462

[13] S. Zhou, X. Li, Y. Chen, S. T. Chandrasekaran and A. Sanyal, "Temporal-coded deep spiking neural network with easy training and robust performance*," in proc. The Thirty-Fifth AAAI Conference on Artificial Intelligence*, Vancouver, Canada, Feb. 2021, pp. 11143-11151. https://ojs.aaai.org/index.php/AAAI/article/view/17329

[14] H. Perez, B. Ogunnaike and S. Devasia, "Output tracking between operating points for nonlinear processes: Van de Vusse example," *IEEE Transactions on Control Systems Technology*, vol. 10, 2002, pp. 611–617. https://doi.org/10.1109/TCST.2002.1014680

[15] P. Patil and C. S. Rao, "Enhanced PID controller for non-minimum phase second order plus time delay system," *Chemical Product and Process Modeling*, vol. 14, no. 3, pp. 20180059, 2019. https://doi.org/10.1515/cppm-2018-0059

[16] V. Alfaro, P. Balaguer and O. Arrieta, "Robustness considerations on PID tuning for regulatory control of inverse response processes," *in: Elsevier (Ed.), IFAC Conference on advanced PID Control*, Bresia, Italy, 2012, pp. 193–198. https://doi.org/10.3182/20120328-3-IT-3014.00033

[17] S. Kuntanapreeda and P. Marusak, "Nonlinear extended output feedback control for CSTRs with Van de Vusse reaction," *Computers & Chemical Engineering*, vol. 41, pp. 10–23, 2012. https://doi.org/10.1016/j.compchemeng.2012.02.010

[18] A. Bertone, R. da M. Jafelice and B. Goes, "Classic and fuzzy type-2 control for the Van de Vusse reactor: A comparative study*," in: S. de Matemática Aplicada e Computacional (Ed.), Proc. Series of the Brazilian Society of Computational and Applied Mathematics*, vol. 6, Sao Carlos, Brazil, Dec. 2018, pp. 1–7. https://doi.org/10.5540/03.2018.006.02.0258

[19] R. Malar and T. Thyagarajan, "Artificial neural networks based modeling and control of continuous stirred tank reactor," *American J. of Engineering and Applied Sciences*, vol. 2, pp. 229–235, 2009. https://thescipub.com/pdf/ajeassp.2009.229.235.pdf

[20] R. Batllori, C. B. Laramee, W. Land and J. D. Schaffer," Evolving spiking neural networks for robot control," *Procedia Computer Science*, vol. 6, pp. 329-334, 2011. https://doi.org/10.1016/j.procs.2011.08.060

[21] R. Stagsted, A. Vitale, J. Binz, A. Renner, L. Bonde, Leon and Y. Sandamirskaya, "Towards neuromorphic control: A spiking neural network based PID controller for UAV," *In: Robotics: Science and Systems, Virtual Conference*, 12-16 July 2020. https://doi.org/10.15607/rss.2020.xvi.074

[22] J. Pérez, J. A.CabreraJuan, J. Castillo and J. M. Velasco, "Bio-inspired spiking neural network for nonlinear systems control," *Neural Networks*, vol. 104, pp. 15-25, 2018. https://doi.org/10.1016/j.neunet.2018.04.002

[23] J. Wang, Spiking Neural P Systems, Doctoral Thesis. Faculty of Science, Leiden University, the Netherlands, IPA Dissertation Series, 169 pages, Dec. 2011. https://scholarlypublications.universiteitleiden.nl/handle/1887/18261

[24] S. Engell and K. Klatt, "Nonlinear control of a nonminimum-phase CSTR," *in: IEEE (Ed.), Proceedings if the American Control Conference, San Francisco, United States of America*, 1993, pp. 2941–2945. https://doi.org/10.23919/acc.1993.4793439

[25] M.A. Márquez-Vera, L.E. Ramos-Velasco and B.D. Balderrama-Hernández, "Stable fuzzy control and observer via LMIs in a fermentation process," *Journal of Computational Science*, vol. 27, pp. 192–198, 2018. https://doi.org/10.1016/j.jocs.2018.06.002

[26] E. Bayro-Corrochano, L. Lechuga-Gutiérrez and M. Garza-Burgos, "Geometric techniques for robotics and HMI: Interpolation and haptics in conformal geometric algebra and control using quaternion spike neural networks," *Robotics and Autonomous Systems*, vol. 104, pp. 72-84, 2018. https://doi.org/10.1016/j.robot.2018.02.015