



Delft University of Technology

## Neuromorphic Control using Input-Weighted Threshold Adaptation

Stroobants, Stein; De Wagter, Christophe; De Croon, Guido

DOI

[10.1145/3589737.3605963](https://doi.org/10.1145/3589737.3605963)

Publication date

2023

Document Version

Final published version

Published in

ICONS 2023 - Proceedings of International Conference on Neuromorphic Systems 2023

### Citation (APA)

Stroobants, S., De Wagter, C., & De Croon, G. (2023). Neuromorphic Control using Input-Weighted Threshold Adaptation. In *ICONS 2023 - Proceedings of International Conference on Neuromorphic Systems 2023* Article 2 (ACM International Conference Proceeding Series). ACM.  
<https://doi.org/10.1145/3589737.3605963>

### Important note

To cite this publication, please use the final published version (if applicable).  
Please check the document version above.

### Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

### Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.  
We will remove access to the work immediately and investigate your claim.



# Neuromorphic Control using Input-Weighted Threshold Adaptation

Stein Stroobants  
s.stroobants@tudelft.nl  
Delft University of Technology  
Delft, The Netherlands

Christophe De Wagter  
Delft University of Technology  
Delft, The Netherlands

Guido C.H.E. de Croon  
Delft University of Technology  
Delft, The Netherlands

## ABSTRACT

Neuromorphic processing promises high energy efficiency and rapid response rates, making it an ideal candidate for achieving autonomous flight of resource-constrained robots. It can be especially beneficial for complex neural networks as are used for high-level visual perception. However, fully neuromorphic solutions also need to tackle low-level control tasks. Remarkably, it is currently still challenging to replicate even basic low-level controllers such as proportional-integral-derivative (PID) controllers. Specifically, it is difficult to incorporate the integral and derivative parts. To address this problem, we propose a neuromorphic controller that incorporates proportional, integral, and derivative pathways during learning. Our approach includes a novel input threshold adaptation mechanism for the integral pathway. This Input-Weighted Threshold Adaptation (IWTA) introduces an additional weight per synaptic connection, which is used to adapt the threshold of the post-synaptic neuron. We tackle the derivative term by employing neurons with different time constants. We first analyze the performance and limits of the proposed mechanisms and then put our controller to the test by implementing it on a microcontroller connected to the open-source tiny Crazyflie quadrotor, replacing the innermost rate controller. We demonstrate the stability of our bio-inspired algorithm with flights in the presence of disturbances. The current work represents a substantial step towards controlling highly dynamic systems with neuromorphic algorithms, thus advancing neuromorphic processing and robotics. In addition, integration is an important part of any temporal task, so the proposed Input-Weighted Threshold Adaptation (IWTA) mechanism may have implications well beyond control tasks.

## KEYWORDS

Neuromorphic control, Spiking Neural Networks (SNNs), Micro-Air-Vehicles (MAVs), Rate Coding, Threshold Adaptation

### ACM Reference Format:

Stein Stroobants, Christophe De Wagter, and Guido C.H.E. de Croon. 2023. Neuromorphic Control using Input-Weighted Threshold Adaptation. In *International Conference on Neuromorphic Systems (ICONS '23)*, August 1–3, 2023, Santa Fe, NM, USA. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3589737.3605963>



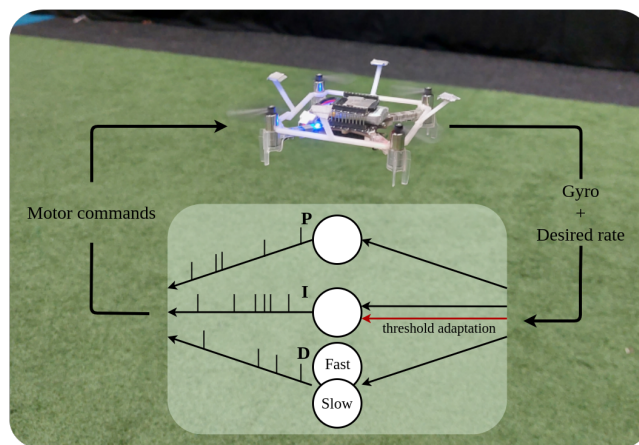
This work is licensed under a Creative Commons Attribution International 4.0 License.

ICONS '23, August 1–3, 2023, Santa Fe, NM, USA  
© 2023 Copyright held by the owner/author(s).  
ACM ISBN 979-8-4007-0175-7/23/08.  
<https://doi.org/10.1145/3589737.3605963>

## 1 INTRODUCTION

Autonomous drones are envisaged for a wide range of applications [22]. Many of these applications require a high computational capability, which enables them to accomplish tasks solely based on data acquired from their onboard sensors, such as cameras and GPS-sensors. Currently, this is out of reach for many drones, since they are very limited in terms of size, weight, and processing [19]. Deep neural networks require heavy, power-hungry processors. That is why there is a surge of interest in bio-inspired, neuromorphic processing, which carries the promise of low-latency, energy-efficient processing of deep neural networks [18]. Although there is a lot of focus on complex neural networks for high-level visual perception [11], a fully neuromorphic solution needs to encompass low-level control [1]. Remarkably, it is currently still highly challenging to replicate even simple low-level controllers such as PID controllers with spiking neural networks. An example of such a low-level controller is the fascinatingly elegant biological system that affects the haltere reflexes of the *drosophila melanogaster* [8].

Recently, an increasing amount of robotics research has been focused on developing spiking neural networks (SNNs) for control. Specifically for controlling flying robots, examples include Clawson et al. [4], which uses reward-modulated synaptic plasticity to track a Linear-Quadratic Regulator (LQR) for a flapping wing drone. In



**Figure 1:** We propose a novel spiking neural network mechanism for realizing the integral term in a spiking PID controller and analyze the use of different time constants for the derivative term. For the integral term, we introduce Input-Weighted Threshold Adaptation, leading to a second weight per synapse. These mechanisms are demonstrated with onboard attitude rate control of a tiny Crazyflie drone.

Qiu et al. [17], a neuro-evolution strategy is utilized to learn a controller for a drone and is shown to outperform a PID in simulation. Closer to our work are studies that mimic the behavior of conventional controllers with SNNs. Among those, the benefits of a spiking end-to-end control pipeline are especially clear in Vitale et al. [21]. In this work, the rotation of a bi-rotor was controlled by combining a neuromorphic implementation of the Hough transform with a population-coded spiking implementation of the conventional PID controller. They showed that due to the high update rates and asynchronous data flow from the event camera and neuromorphic chip, much faster responses could be obtained than with a conventional control setup. The accuracy of this network scaled quadratically with the number of neurons, putting a limit on the resolution. In our previous work, the complexity of such a PID network was reduced to make it scale linearly with the number of neurons, and the network was used to control the altitude of a free-flying drone in a real-world test [20]. However, the integral and derivative paths in both these works showed clear limitations, imposed by the number of neurons used to represent the signals.

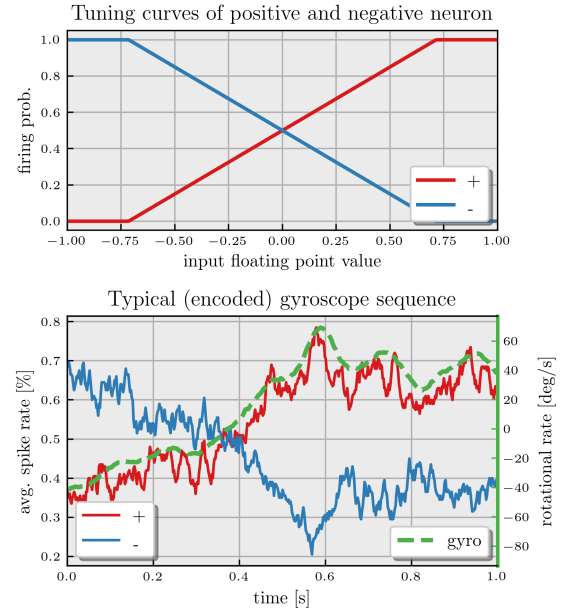
Besides population coding, also rate coding was used to recreate conventional control. For example, in Zaidel et al. [23] the joints of a 6-DOF robotic arm were controlled by a rate-encoded PID controller, creating the integral pathway by having self-recurrent connections and the derivative by adopting different time constants. Despite using this self-recurrency in the integral pathway, there still remained a steady-state error, showing the incapability of error integration over time.

In all these examples the seemingly simple tasks of calculating the integral and derivative over time have proven to be difficult for the current-based Leaky Integrate-and-Fire (LIF) model. This may be due to the simplicity of the LIF neuron used in most robotics research. Popular for its simplicity, it fails to deal with the phasic-tonic response exhibited by biological neurons [13]. Lastly, although all these works show clear steps toward a full end-to-end neuromorphic pipeline, none of them has demonstrated the ability to control the lowermost loop of a real, free-flying drone.

We present a neuromorphic controller that can more closely mimic PID controllers. We make the following contributions: (1) We introduce Input-Weighted Threshold Adaptation (IWTA) to achieve more accurate integration. (2) We systematically study the capabilities and limitations of neurons with slow and fast time constants for obtaining the derivative term. (3) We analyze the advantages and limitations of the introduced mechanisms. (4) We demonstrate the novel neuromorphic controller with the onboard attitude rate control of a tiny ( $\approx 30$  gram) flying Crazyflie robot (shown in Figure 1). Using only 320 neurons and 800 synapses per PID line, it is designed to take advantage of the unique characteristics of neuromorphic computing, such as high processing speed and fault tolerance, while maintaining the promise of low power consumption when implemented in specialized hardware. The network is automatically trained to closely track the output of a conventional PID using backpropagation-through-time (BPTT), removing the necessity for manual tuning of network parameters.

## 2 METHODS

The entire SNN consists of multiple groups of LIF neurons, each resembling one of the separate parts of a conventional PID controller. All groups consist of an encoding layer, transforming floating point inputs to rate encoded spike-trains representing positive and negative values. These spikes propagate to neurons that respond proportionally to the input, to the accumulated signal over time or to the rate of change of the input. These independent systems are discussed in detail below.



**Figure 2: Overview of the encoding layer.** The top graph shows the firing probability for a certain stimulus for positive and negative encoding neurons. The bottom graph depicts a typical sequence of encoded gyroscope measurements. The average spike rate of the encoded positive and negative spikes is shown in blue and red and the input sequence is shown in dashed green.

### 2.1 Encoding - floating points to spikes

For our network, a rate encoding scheme has been chosen that has separate channels for positive and negative values. To ensure a certain spiking frequency at an input stimulus, encoding is done according to two (symmetrical) tuning curves. These tuning curves represent the spike probability of an encoding neuron to a given stimulus, and by comparing this to a random-generated number, either a spike (1) or not (0) is produced by the neuron. Although more complex functions can be chosen for these tuning curves, in this work a linear relation between spiking frequency and stimulus was chosen. This linear relation between input  $i(t)$  and output spike probability  $P(s(t))$  at time  $t$  is dictated by the parameters  $\alpha$

and  $\beta$  as follows:

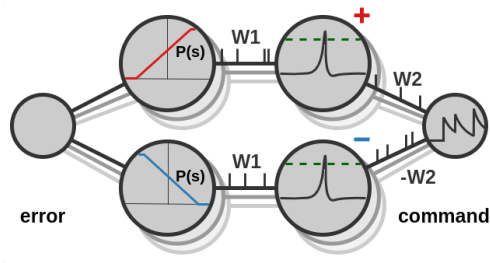
$$P(s(t)) = \begin{cases} 0 & r\beta i(t) + \alpha \leq 0 \\ r\beta i(t) + \alpha & 0 \leq r\beta i(t) + \alpha \leq 1 \\ 1 & r\beta i(t) + \alpha \geq 1 \end{cases} \quad (1)$$

$$r = \begin{cases} 1 & \text{positive neuron} \\ -1 & \text{negative neuron} \end{cases} \quad (2)$$

The value of  $r$  depends on whether the encoding neuron is positive or negative. This is visualized in Figure 2 where the set of tuning curves used in this work and the resulting output spikes are shown for a typical input sequence measured with gyroscopes.

## 2.2 Proportional - steering towards setpoint

The output of the proportional layer should drive the system from its current state to a setpoint by responding linearly to the error. In this network, this is done by feeding the output of an encoding group, as described above, to two current-based Leaky Integrate-and-Fire (LIF) neurons, again each representing either positive or negative control commands (see Section S1.1 for the model used). The spikes from this layer are sent to an output leaky integrator, acting as an exponential filter to smooth the response. To ensure a balanced response, the synaptic weights in the network must be symmetrical for the positive and negative inputs. By using more than one group, the stochastic effects induced by the encoding layer can be reduced and the accuracy increased.



**Figure 3: Structure of the proportional neuron groups.** The value is encoded into "positive" and "negative" spikes and sent via symmetrical weights to a layer of Leaky Integrate and Fire neurons. The spikes emitted by these neurons are sent to a Leaky Integrate output neuron, resulting in an exponential filtered output.

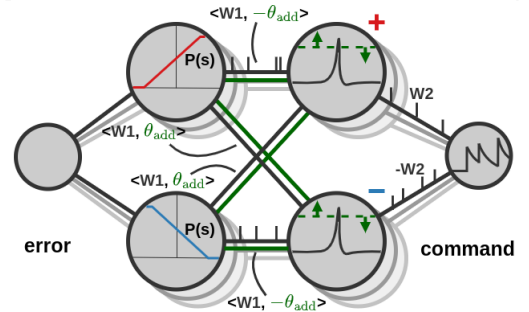
## 2.3 Integral - ensuring zero steady-state error

The integral neurons should remove the steady-state error, unresolved by the proportional control. Initially, one may think that LIF neurons could integrate by setting the decay parameter to one (no decay). However, as soon as a spike occurs the integrated membrane potential is reset. Hence, if the membrane potential is not read out directly and the integrator has to be encoded by spikes, a different mechanism is required. We propose a different solution in this work inspired by threshold adaptation mechanisms and the modulatory effects certain receptors exhibit. There is a synaptic connection between the positive and negative encoding neurons to

both neurons, as opposed to the proportional connections where there is only a connection between the positive neurons on the one hand and between the negative neurons on the other. Besides the synaptic weights, there is also a signal flowing from the encoding neurons to the thresholds of the neurons in the integration layer, increasing or decreasing the threshold based on the sign. Previous work has studied various mechanisms for adapting the threshold based on inputs, often to prevent spike saturation. For example, the regular Adaptive LIF (ALIF), adapts its threshold based on its own spike activity [3]. As a variation of the ALIF, Paredes-Vallés et al. [16] proposes to deduct the pre-synaptic spike trace from incoming spikes, to discern features under varying input statistics, such as the per-pixel firing rate of event camera data. In our method, however, the threshold is regulated based on weighing the incoming activity. A spike in the *positive* integration neuron coming from the *positive* encoding neuron decreases the threshold with a weight of  $\theta_{add}$  (therefore increasing the spiking rate), while the *negative* encoding neuron causes an increase with  $\theta_{add}$  (thus decreasing the spiking rate), and vice-versa for the *negative* integration neuron. This results in the following update rule for the threshold:

$$\theta^{thr}(t+1) = \theta^{thr}(t) + \theta_{add}(s_-(t) - s_+(t)), \quad (3)$$

with  $s_-(t)$  and  $s_+(t)$  the negative and positive incoming spikes, respectively. Now, the encoding neurons act as a constant driving synaptic signal to maintain a certain activity in the integration layer, while the actual spiking rate is governed by the variation in the threshold. A common problem with PID regulators is integral windup, where actuator saturation or large changes in setpoint might lead to large amounts of accumulated error [2]. In our LIF model, we solve this by limiting the amount of change in the threshold, keeping the threshold in the range of  $[0, 2\theta^{thr}]$  where  $\theta^{thr}$  is the base threshold. If the threshold is zero, the integrator will spike with the maximal spiking frequency, which is determined by the time step and refractory period. It could be noted that multiple



**Figure 4: Structure of the integral neuron groups, bearing the Input-Weighted Threshold Adaptation (IWTA) mechanism.** Next to the regular synapses, additional connections have been added that adapt the threshold of the LIF neurons according to their weights.

changes to this model are imaginable. For instance, if a decay term is added to the threshold it more closely resembles the ALIF, where the threshold converges back to a base value. In biology, this kind of input adaptation is similar to certain neurotransmitters with modulatory effects [7]. Also, every input group now uses only a single



positive and negative encoding neuron, with one update parameter  $\theta_{\text{add}}$ . One could imagine using a larger group of encoding neurons, separate update weights  $\theta_{\text{add}}$  per input connection and including these weights in the training procedure. In this work, this remains unexplored since it focused on the task of integrating errors for which the proposed set of connections was sufficient.

## 2.4 Derivative - decreasing overshoot

The derivative action should be proportional to the change over time, countering any potential overshoot. To obtain this, a similar structure to the proportional groups is used (as in Figure 3), but now two of these groups are used in unison, instead of one. One of these groups has very fast time constants (higher weights, but faster decay), allowing it to react quickly proportionally to the input. The other has slower time constants (lower weights, but slow decay), resulting in an output that is a slightly delayed version of the input. By taking the difference between these two groups, we get a measure of the change over time which can be used as the derivative term in our PID controller. Using multiple of these derivative groups again increases the accuracy of the overall estimate. For the derivative, this is especially important, as the derivative is usually already quite noisy due to noisy sensor measurements.

## 2.5 Training and tuning of the network

Since the network has a substantial number of parameters that all influence the performance of the controller (such as synaptic weights, decay parameters and encoding parameters) and the parameters all depend on the time constants and gains of a specific controller, manual tuning is undesirable. Therefore, it was chosen to use the architecture of the network as described above as a starting point and fine-tune the parameters using Backpropagation-Through-Time (BPTT). Because the LIF threshold function is non-differentiable, it was chosen to apply a surrogate gradient in the backwards-pass of BPTT [14]. Specifically, the surrogate gradient used in this work is the derivative of the arc-tangent as was proposed in [9].

To force a response that is close to that of the target, the Mean Squared Error (MSE) was used as the dominant term in the cost function. Since during training, especially the derivative term was very sensitive to converging to local minima, it was chosen to add a second cost term for the derivative based on the Pearson correlation coefficient  $\rho(x, \hat{x})$  [5]. Since we have a minimization problem and the perfect coefficient  $\rho$  is 1, we use  $1 - \rho(x, \hat{x})$  as the cost, further referring to it as the Pearson loss. For derivative control, it is very important that the control action is at least of the correct *sign* because failing so might mean instabilities can arise. The Pearson loss promotes a linear relationship between both inputs and therefore supports the network to produce the correct sign. Finally, the parameters of the network need to stay within certain bounds, decay parameters for instance cannot be larger than 1 or smaller than 0. To force parameters to stay within these bounds, a linear exterior penalty function  $p_{\text{err}}(\phi)$  is added to the cost function equal to the distance to the boundary. This results in the following cost function used in the BPTT training algorithm:

$$J(\phi) = \text{MSE}(x, \hat{x}) + (1 - \rho(x, \hat{x})) + p_{\text{err}}(\phi). \quad (4)$$

In this cost function,  $x$  and  $\hat{x}$  are the measured- and estimated values respectively,  $1 - \rho(x, \hat{x})$  the Pearson loss.  $p_{\text{err}}(\phi)$  is the error based on the parameters  $\phi$ , which is zero for all values of  $\phi$  inside their specific range but of size  $|\phi|$  for those outside. A table of all parameters included in the training, together with their valid ranges, can be found in the supplementary information.

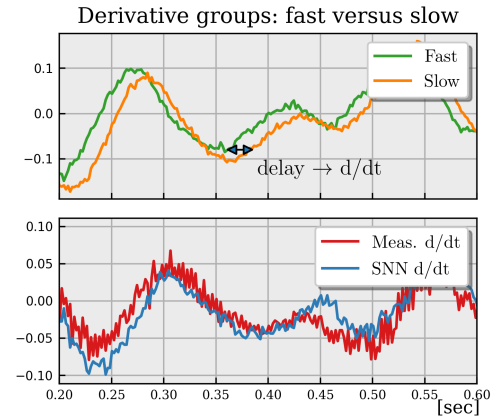
The data used for training was accumulated by logging the PID values of the regular Crazyflie controller on an onboard  $\mu\text{SD}$  card during manual flight. Care was taken to excite the system enough to gather a broad range of possible values a controller might encounter.

## 3 ANALYSIS SNN CONTROLLER

First, we look at the suitability of the multiple time constants for differentiation and afterward, we evaluate the IWTA mechanism for integral control.

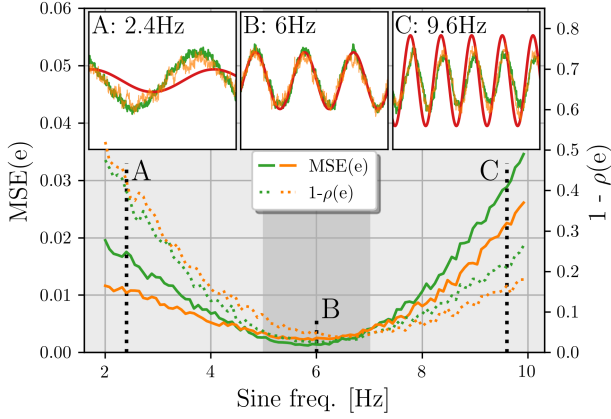
### 3.1 Derivative

To assess the ability of a network of LIFs with different time constants to estimate the derivative we start by looking at the response of both the fast and slow groups to an illustrational gyroscope sequence, obtained with the Crazyflie, after training. As can be seen in Figure 5, the average spiking rate of the slow groups is approximately a delayed version of the fast groups. By subtracting the delayed version we obtain a result similar to first-order backward difference, usually used in robotics to calculate the derivative of sensor data. We noticed, however, that a particular set of time constants



**Figure 5: Example response to a common gyroscope input sequence recorded with the Crazyflie of both derivative groups. The top graph shows the average spiking rate of the fast and slow groups. As can be seen, the slow group is slightly delayed and the difference between both can be seen as a measure of the rate of change over time. In the bottom figure, this is visible as the derivative estimated with the SNN is compared to the measured derivative.**

fits chiefly to the data it was trained upon. To further investigate this behavior, the response to sine waves with different frequencies was studied. Figure 6 shows the MSE and Pearson loss for a range of sine waves with different frequencies after training on two sets of frequencies. The network was trained on a smaller range of sine



**Figure 6: MSE and Pearson loss for a range of derivatives of sine waves after training on two sets of frequencies, 5-7Hz and 2-10Hz. Green lines correspond to the smaller range, and orange to the larger while the solid lines show MSE and dotted Pearson loss. The darker-colored area depicts the range on which the smaller set was trained. Three sub-figures are also included, demonstrating the response to a frequency A) below, B) inside and C) above the smaller training range with the SNN response in green and orange and real derivative in red.**

waves between 5 and 7Hz, and also on a much wider range of 2 to 10Hz. Afterwards, the response to the entire frequency range was compared for both trained networks. Although the network can accurately determine the derivative for the middle frequencies (B), it is too fast for lower frequencies (A), where the network in blue rises to its peak faster than the real derivative in red, and too slow for higher ones (C) where the network reaches its tipping point later. It is also visible that even with the larger range of input frequencies during training, the different time constants cannot correctly represent the entire band. Although the overall error gets reduced when training on the large band, the response for the lower and higher frequencies is still inaccurate. This suggests that a different mechanism would need to be introduced in order to obtain perfect differentiation independent of input frequency.

The importance of both cost terms in the loss function is also visible; the MSE cost for lower frequencies is relatively lower than the Pearson loss and for higher frequencies vice versa. Also, the amplitude of the SNN does not scale with the frequency, as is the case in the real derivative. This asserts the need for a reliable training procedure to tune the network to the application used, as was done in this work.

### 3.2 Integral

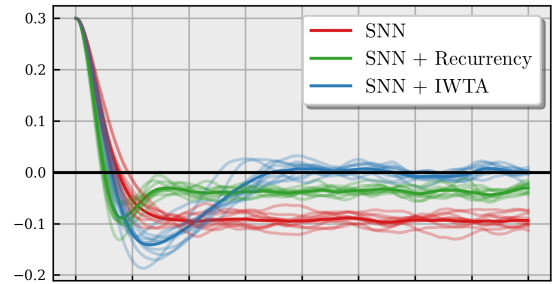
Furthermore, we have looked into the potential of the proposed Input-Weighted Threshold Adaptation (IWTA) mechanism. To present the importance of a functioning integrator, the classical control problem of the double integrator is used as an example. We have implemented the discrete-time equations of the double-integrator

as

$$x(t+dt) = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} \frac{1}{2}dt^2 \\ dt \end{bmatrix} (u(t) - g), \quad (5)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t), \quad (6)$$

where  $u(t)$  is the control input and  $g$  is a constant input disturbance. In Figure 7 we show the response of this system to three different SNNs; the LIF SNN with only PD components, an SNN with PD components and an integrator group that has fully recurrent connections (as was used in [23]) and lastly our SNN with IWTA. The system starts from an initial state of  $x_0 = [0.3, 0.0]^T$  and gets a setpoint 0.0. The constant disturbance force  $g$  was chosen to be 4.0. First, the SNN without any integrating mechanism settles at a steady-state error of  $-0.1$ . Second, the SNN with recurrent connections manages to reduce this steady-state offset, but not remove it completely. Due to representation errors, the recurrent connections cannot perfectly describe the integrated error. This means that the accumulated error is either underrepresented or overrepresented. In the former case, the integrator leaks information each time step and thus never completely removes the steady-state error. The green lines in Figure 7 show this, since it clearly reduces the steady-state error, but does not remove it completely. In the latter, the feedback amplifies the error at each time step which makes the system unstable and leads to all integrator neurons spiking at each step. Finally, the network that uses IWTA reaches a zero steady-state error.



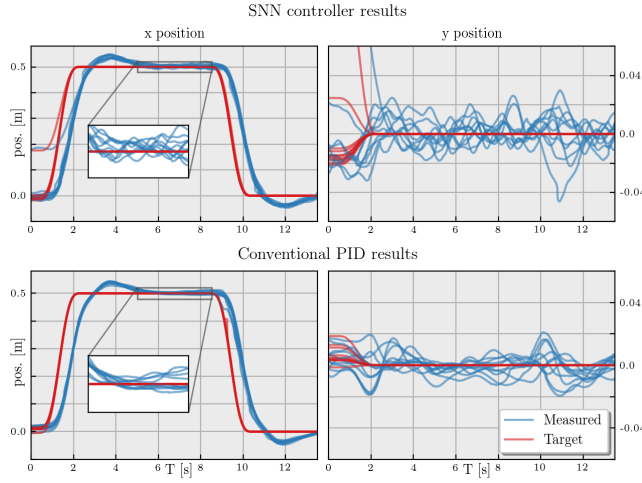
**Figure 7: Comparison of three controllers to a setpoint-control task with an unknown input disturbance. The SNN with PD control only shows a constant offset from the setpoint after converging. The SNN with recurrency is clearly able to reduce the steady-state error, but due to representation error retains a small offset. The network equipped with Input-Weighted Threshold Adaptation effectively removes the steady-state error.**

## 4 REAL-WORLD EXPERIMENTS

After the components are evaluated separately in simulation on toy problems, the use of the complete network was verified on a real-world problem.

#### 4.1 Hardware implementation

To demonstrate the capabilities of our approach, we have implemented it as the lowest control layer of the tiny open-source quadrotor Crazyflie [10] (See Figure 1). The Crazyflie was enriched with enough computational power by the development of a deck module based on a Teensy 4.0 development board allowing the SNN to run in real-time (500Hz) in C++ on the ARM Cortex-M7 microprocessor. For the real-world tests, two scenarios were discerned; 1) manual flight and 2) position control. For manual control the Crazyflie receives attitude (roll/pitch/yaw) commands from a (manually controlled) radio transmitter and the higher-level attitude controller transforms these into rate setpoints. In the case of position control, the Crazyflie receives position measurements obtained with a Motion Capture system along with position commands via radio and via the same high-level controller these are converted to rate setpoints. These setpoints are sent, along with the gyroscope measurements from the onboard Inertial Measurement Unit (IMU), via UART to the deck where the SNN controller is evaluated at 500Hz. The torque command outputs of the neural controller are in turn sent back to the Crazyflie via the same UART connection, where they are inserted directly in the motor mixer. The total take-off weight of the Crazyflie including the Teensy 4.0 is only 35 grams, allowing for approximately 5 minutes of flight time.



**Figure 8: Position setpoint responses of both the SNN controller (top) and the conventional PID (bottom). For both, the positions as measured by the Motion Capture system are shown in blue, and setpoints in red. Although the SNN controller has a slightly noisier response, its trajectory is very similar to the PID.**

#### 4.2 Flight tests

To demonstrate the capabilities of the network, a position-control test was performed along with manual flight. First, the Crazyflie was ordered to take off at its current position and after 2 seconds it was ordered to move to  $[0.5, 0.0]$  in 2 seconds. This test was repeated 10 times for both the SNN controller and the regular PID controller as a benchmark. The results of all these 10 tests are shown in Figure 8. In the top two plots, the response of the SNN

controller is shown. Among all ten tests, the controller remained stable and was able to follow the setpoint. The trajectory followed is very similar to that of the regular PID controller. However, on the  $y$ -results, as well as the inset axes for  $x$ , it is visible that the SNN controller has slightly larger deviations around the setpoint. These can be caused by the stochasticity in the encoder. Since the input floating point value is reduced to a binary spiking input, the accuracy of the encoding is influenced by the encoding parameters but mostly by the number of input groups used. For this work, only 40 groups were chosen per P, I and D pathway for each control axis. Since the P and I terms require 2 neurons per group and the D term 4, this sums up to a total of 320 neurons per controller. This small number of neurons was chosen to showcase the possibilities of using small-scale systems for neuromorphic control. However, increasing the number of input groups would make the spiking representation of the input signals more accurate.

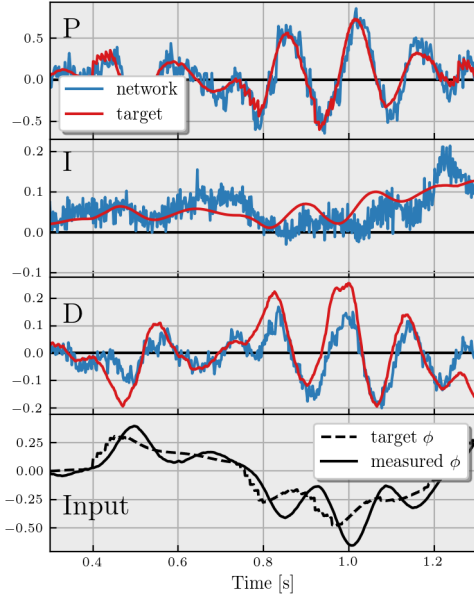
Besides the position-control tests, the response to manual flight was performed with which the response of the different parts of the controller on a real-world system could be analyzed. In Figure 9, one second of such a test is shown. It is evident that the proportional part is very accurate, and the integral pathway is able to effectively deal with prolonged errors. The response of the derivative pathway is less accurate. This is most likely caused by the mechanisms shown in the derivative analysis, earlier in this work (Section 3.1), where it was shown that the derivative pathway tunes to specific delays. Even though the network is trained on real flight data, the range in delays might be too large which results in larger errors for the lowest and highest frequencies. For control of the Crazyflie, this is still acceptable since the derivative control path still effectively dampens the response by countering large derivatives in the input.

### 5 CONCLUSION

In this work, we have proposed a novel input threshold adaptation mechanism, Input-Weighted Threshold Adaptation (IWTa). This mechanism adds extra weights per input connection that regulate the spiking threshold of the LIF neuron. By doing so, it enhances the network with the ability to integrate information over time, something the regular LIF model is unable to do.

Also, we have shown that neuromorphic controllers using rate-based encoding can be used to control highly unstable underactuated systems. To demonstrate this, we have shown control of the innermost loop of a real flying tiny quadrotor, the Crazyflie. Using only 320 neurons per control axis, the network showed to be capable of stable and robust control, with the potential of extremely low delays due to the high inference speed of neuromorphic hardware. By a straightforward training method using surrogate gradients and Backpropagation Through-Time, the network can be fine-tuned to a very limited amount of data from a real-world flying drone. Due to the sparse connections, the network is able to optimally benefit from the advantages of neuromorphic hardware.

In future work, we intend to apply the IWTa mechanism to different tasks and benchmarks to further establish its potential. Even though the different time constants for the derivative neurons allow us to dampen the control response, we have shown that they are limited to a specific frequency. We will also investigate the application of IWTa on the derivative neurons to improve the



**Figure 9: Output of the SNN controller in blue versus target values computed with a conventional PID controller in red for a manually controlled flight. The three uppermost figures depict the individual components of the controller. As can be seen, the SNN controller produces very similar results to the reference controller. The lowermost figure shows the input to the network; the target rotational rate  $\phi$  and the rate measured by the gyro.**

accuracy over a much broader range of frequencies. The availability of a neuromorphic controller, such as a PID, that can be easily implemented on neuromorphic hardware plays a crucial role in completing the neuromorphic control loop in robotics. These controllers can be readily integrated into pipelines utilizing event-based algorithms, like a vision-based control system using an event camera as in [15]. Lastly, the fast-paced and unpredictable movements of drones demand high-performance computing that traditional hardware struggles to provide, making neuromorphic processing an attractive alternative.

## SUPPLEMENTARY INFORMATION

### 1.1 Current-based Leaky-Integrate-and-Fire neuron

The current-based Leaky-Integrate-and-Fire (CUBA-LIF) is widely used in literature, available in most SNN simulators and commonly used in neuromorphic hardware, such as Intel’s Loihi [6]. The discrete-time dynamic equations of the LIF neuron are as follows:

$$v_i(t+1) = \tau_i^{\text{mem}} v_i(t) + i_i(t), \quad (7)$$

$$i_i(t+1) = \tau_i^{\text{syn}} i_i(t) + \sum w_{ij} s_j(t), \quad (8)$$

where  $v_i(t)$  is the membrane potential at time  $t$ ,  $\tau_i^{\text{mem}} \in [0, 1]$  and  $\tau_i^{\text{syn}} \in [0, 1]$  the membrane and synaptic time constants,  $i(t)$  the synaptic current at time  $t$ ,  $w_{ij}$  the synaptic weight between neurons

$i$  and  $j$ , and  $s_j$  a binary value representing either a spike or no spike coming from the pre-synaptic neuron  $j$ . To determine whether a neuron emits a spike, the membrane potential is reduced with the neurons firing threshold  $\theta_i^{\text{thr}}$  and passed through the Heaviside step function to determine the output of the neuron:

$$s_i(t) = H(v_i(t) - \theta_i^{\text{thr}}) = \begin{cases} 0, & v_i(t) - \theta_i^{\text{thr}} \leq 0 \\ 1, & v_i(t) - \theta_i^{\text{thr}} > 0 \end{cases} \quad (9)$$

When the Heaviside function resolves to 1 and the neuron emits a spike, the membrane potential  $v_i(t)$  is reduced by the threshold value (in literature this is known as a soft reset [12]).

### 1.2 Trained parameters and ranges

In Table S2, all the parameters used in the network are given, along with the specified range and the number that was used in the Crazyflie application.

**Table S2: All parameters trained in the network, their given range and how many are used for the real-world tests.**

|          | Parameter                | Range    | Count |
|----------|--------------------------|----------|-------|
| <b>P</b> | $\tau_i$ (current decay) | [0, 1]   | 80    |
|          | $\tau_v$ (voltage decay) | [0, 1]   | 80    |
|          | $w_i$ (input weight)     | [0, inf] | 40    |
|          | $w_o$ (output weight)    | [0, inf] | 40    |
| <b>I</b> | $\tau_i$ (current decay) | [0, 1]   | 80    |
|          | $\tau_v$ (voltage decay) | [0, 1]   | 80    |
|          | $w_i$ (input weight)     | [0, inf] | 40    |
|          | $w_o$ (output weight)    | [0, inf] | 40    |
| <b>D</b> | $\tau_i$ (current decay) | [0, 1]   | 160   |
|          | $\tau_v$ (voltage decay) | [0, 1]   | 160   |
|          | $w_i$ (input weight)     | [0, inf] | 80    |
|          | $w_o$ (output weight)    | [0, inf] | 80    |

## ACKNOWLEDGMENTS

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA8655-20-1-7044.

## REFERENCES

- [1] Ignacio Abadia, Francisco Naveros, Eduardo Ros, Richard R Carrillo, and Niceto R Luque. 2021. A cerebellar-based solution to the nondeterministic time delay problem in robotic control. *Science Robotics* 6, 58 (2021).
- [2] Karl Johan Astrom and Lars Rundqwist. 1989. Integrator windup and how to avoid it. In *1989 American Control Conference*. IEEE, IEEE, Pittsburgh, PA, USA, 1693–1698. <https://doi.org/10.23919/ACC.1989.4790464>
- [3] Romain Brette and Wulfram Gerstner. 2005. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of neurophysiology* 94, 5 (2005), 3637–3642.
- [4] Taylor S Clawson, Silvia Ferrari, Sawyer B Fuller, and Robert J Wood. 2016. Spiking neural network (SNN) control of a flapping insect-scale robot. In *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, IEEE, Las Vegas, NV, USA, 3381–3388. <https://doi.org/10.1109/CDC.2016.7798778>
- [5] Israel Cohen, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. *Pearson Correlation Coefficient*. Springer, Berlin, Heidelberg, 1–4. [https://doi.org/10.1007/978-3-642-00296-0\\_5](https://doi.org/10.1007/978-3-642-00296-0_5)



- [6] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham China, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 1 (2018), 82–99.
- [7] Gaetano Di Chiara, Micaela Morelli, and Silvana Consolo. 1994. Modulatory functions of neurotransmitters in the striatum: ACh/dopamine/NMDA interactions. *Trends in neurosciences* 17, 6 (1994), 228–233.
- [8] Michael H Dickinson. 1999. Haltere-mediated equilibrium reflexes of the fruit fly, *Drosophila melanogaster*. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 354, 1385 (1999), 903–916.
- [9] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. 2021. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision. CVPR, Virtual*, 2661–2671.
- [10] Wojciech Giernacki, Mateusz Skwierczyński, Wojciech Witwicki, Paweł Wroński, and Piotr Kozierski. 2017. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. In *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*. IEEE, IEEE, Miedzyzdroje, Poland, 37–42. <https://doi.org/10.1109/MMAR.2017.8046794>
- [11] Alessandro Giusti, Jérôme Guzzi, Dan C Cireşan, Fang-Lin He, Juan P Rodriguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. 2015. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters* 1, 2 (2015), 661–667.
- [12] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. 2020. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. CVPR, Virtual*, 13558–13567.
- [13] Marie Levakova, Lubomir Kostal, Christelle Monsempès, Philippe Lucas, and Ryota Kobayashi. 2019. Adaptive integrate-and-fire model reproduces the dynamics of olfactory receptor neuron responses in a moth. *Journal of The Royal Society Interface* 16, 157 (2019), 20190246. <https://doi.org/10.1098/rsif.2019.0246> arXiv:<https://royalsocietypublishing.org/doi/pdf/10.1098/rsif.2019.0246>
- [14] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. 2019. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine* 36, 6 (2019), 51–63.
- [15] Federico Paredes-Vallés, Jesse Hagenaars, Julien Dupeyroux, Stein Stroobants, Yingfu Xu, and Guido de Croon. 2023. Fully neuromorphic vision and control for autonomous drone flight. *arXiv preprint arXiv:2303.08778* (2023), 18 pages.
- [16] Federico Paredes-Vallés, Kirk YW Scheper, and Guido CHE De Croon. 2019. Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception. *IEEE transactions on pattern analysis and machine intelligence* 42, 8 (2019), 2051–2064.
- [17] Huanneng Qiu, Matthew Garratt, David Howard, and Sreenatha Anavatti. 2020. Evolving spiking neurocontrollers for UAVs. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, IEEE, Canberra, ACT, Australia, 1928–1935. <https://doi.org/10.1109/SSCI47803.2020.9308275>
- [18] Catherine D Schuman, Thomas E Potok, Robert M Patton, J Douglas Birdwell, Mark E Dean, Garrett S Rose, and James S Plank. 2017. A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963* (2017), 88 pages.
- [19] Silvia Sekander, Hina Tabassum, and Ekram Hossain. 2018. Multi-tier drone architecture for 5G/B5G cellular networks: Challenges, trends, and prospects. *IEEE Communications Magazine* 56, 3 (2018), 96–103.
- [20] Stein Stroobants, Julien Dupeyroux, and Guido De Croon. 2022. Design and implementation of a parsimonious neuromorphic PID for onboard altitude control for MAVs using neuromorphic processors. In *Proceedings of the International Conference on Neuromorphic Systems 2022*. Association for Computing Machinery, Knoxville, TN, USA, 1–7. <https://doi.org/10.1145/3546790.3546799>
- [21] Antonio Vitale, Alpha Renner, Celine Nauer, Davide Scaramuzza, and Yulia Sandamirskaya. 2021. Event-driven vision and control for UAVs on a neuromorphic chip. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, IEEE, Xi'an, China, 103–109. <https://doi.org/10.1109/ICRA48506.2021.9560881>
- [22] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. 2008. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine* 29, 1 (2008), 9–9.
- [23] Yuval Zaidel, Albert Shalunov, Alex Volinski, Lazar Supic, and Elishai Ezra Tsur. 2021. Neuromorphic NEF-based inverse kinematics and PID control. *Frontiers in neurorobotics* 15, 631159 (2021), 12 pages. <https://doi.org/10.3389/fnbot.2021.631159>