Quadcopter Flight Control using Modular Spiking Neural Networks

Dissertation

submitted in partial fulfillment of the requirements
for the degree of

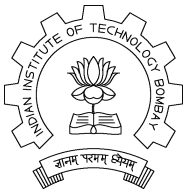
Bachelor of Technology

by

Sushrut R. Thorat
Roll No: 110260017

under the guidance of

Prof. Bipin Rajendran



Department of Physics
Indian Institute of Technology, Bombay
Mumbai - 400076.

2015

Declaration of Authorship

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Sushrut R. Thorat
IIT Bombay
April 17, 2015

Dissertation Approval Sheet

This is to certify that the dissertation titled

Quadcopter Flight Control using Modular Spiking Neural Networks

By

Sushrut R. Thorat
(110260017)

is approved as his **BTech Project II** thesis

Prof. Bipin Rajendran
(Guide)

Prof. Pradeep Sarin
(Co-Guide)

Date : _____

Dedicated to
my dearest
Miss. Shilpa Mohite
for weathering me through the ebbs and flows of life

Foreword

The dynamics of a quadcopter are unstable and non-linear. As a result, the quadcopters flight relies heavily on the Flight controller. Here we present a robust control scheme which can act as the flight controller for the quadcopter. We then describe a scheme to translate this scheme into a Spiking Neural Network using a modular approach, and analyse the performance of the SNN while it controls the quadcopter flight in realistic environmental conditions (presence of noisy wind, IMU noise, and delayed signals).

The dynamics of the quadcopter and the classical control scheme were dealt with, in detail, in the BTP-1 report. I have discussed them in short in the first paper - *Quadcopter Flight Control using Modular Spiking Neural Networks*, which is being written for the conference Engineering Application of Neural Networks (EANN'15).

The modular approach of converting arithmetic operations to spiking neural networks is dealt with in the second paper - *Arithmetic Computing via Rate Coding in Neural Circuits with Spike-triggered Adaptive Synapses*, which has been accepted at the International Joint Conference on Neural Networks (IJCNN'15).

We are working on constructing the entire SNN for quadcopter control as of now, and will be analysing its performance thereafter.

Quadcopter Flight Control using Modular Spiking Neural Networks*

Sushrut Thorat, Sukanya Patil, and Bipin Rajendran

Indian Institute of Technology - Bombay,
Powai, Mumbai - 400076, India

Abstract. The dynamics of a quadcopter are unstable and non-linear. As a result, the quadcopter's flight relies heavily on the Flight controller. Here we present a robust control scheme which can act as the flight controller for the quadcopter. We then describe a scheme to translate this scheme into a Spiking Neural Network using the modular approach described in [8], and analyse the performance of the SNN while it controls the quadcopter flight in realistic environmental conditions (presence of noisy wind, IMU noise, and delayed signals).

Keywords: Spiking Neural Networks, Adaptive Synapses, Flight Control

1 Introduction

Quadcopter design is popular in Unmanned Aerial Vehicle (UAV) research. The need for aircraft with greater maneuverability and hovering ability has led to the current rise in quadcopter research. They have been recently used in acquiring aerial imagery, assisting in disaster assessment, surveillance by the military and delivering parcels.

A quadcopter is a four-rotor rotorcraft. As a quadcopter is highly susceptible to disturbances, it requires a robust controller. The simplest controller is the PID controller [1]. As can be seen in [2], a PID cannot recover a quadcopter suffering from huge angular disturbances. Its recovery and noise-resilience capabilities are poor. Complex controllers with self-tuning PIDs [3] and model-dependent control [4] have been proposed, but as quadcopters are susceptible to disturbances and unknown dynamics, making an accurate model is difficult. An accurate model is computationally intensive too. So, adaptive methods were explored.

Neural Networks are excellent adaptive systems owing to their multitude of adaptive connections. There have been many successful approaches to incorporate neural networks in quadcopter control [2, 5, 6]. But none of these have used the third generation of neural networks (NNs) - the Spiking Neural Networks (SNNs). SNNs are computationally more powerful than the first two generations of NNs, and are highly noise-resilient [7]. Moreover, they are biologically realistic. Animals and birds can navigate efficiently in diverse environments. Their

* Ongoing project. This paper is being written for submission towards EANN 2015.

nervous systems are responsible for these traits. So, the properties SNNs possess are well-suited to navigation. Thus, we would like to analyse the performance of a SNN in controlling quadcopter flight.

In this paper, we first discuss the physics of quadcopter motion. Second, we propose a model-dependent classical algorithm to control quadcopter flight. We then begin to translate the arithmetic operations in the classical algorithm to SNNs by using the modular approach suggested in [8]. Finally, we discuss the construction of a SNN to control quadcopter flight and test its performance in realistic environmental settings as compared to the classical algorithm.

2 Quadcopter Mechanics

A quadcopter has 4 rotor blades which produce thrust. The frame of the quadcopter holds the Inertial Measurement Unit (IMU), the microprocessor, the sensors, and other payload. The *state* of a quadcopter can be described by 6 coordinates - the three Cartesian coordinates [X, Y, and Z] of the Center of Mass of the quadcopter in the ground frame, and the three Euler angles [θ(pitch), ϕ(roll), and ψ(yaw)] in the vehicle frame (a cartesian frame which moves with the quadcopter, without any rotations w.r.t the ground frame).

The external forces that act on the quadcopter are the gravitational force, air pressure, and drag. The difference in air pressure creates a thrust at each rotor (Similar to how airplanes generate lift). The air also exerts a drag force on the frame of the quadcopter.

The lifts generated by the motors can be considered to be proportional to the squared motor speeds [9], hence,

$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2), \tag{1a}$$
$$U_2 = lb(-\Omega_2^2 + \Omega_4^2), \tag{1b}$$
$$U_3 = lb(-\Omega_1^2 + \Omega_3^2), \tag{1c}$$
$$U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2), \tag{1d}$$
$$\Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \tag{1e}$$

where, Ω_n is the speed of motor n, U_1 is the overall thrust produced, U_2 is the roll torque, U_3 is the pitch torque, U_4 is the yaw torque, b is the Thrust factor, d is the Drag factor, and l is the length of an arm of the quadcopter. The function of Ω will become clear in a moment.

If we consider that the wind velocity is $\mathbf{v}_w$, and the torque due to the gradient in the wind speed over the face of the quadcopter is $\mathbf{T}$, then the coordinate

accelerations are given by [10],

$$\ddot{X} = (sin\psi.sin\phi + cos\psi.sin\theta.cos\phi)\frac{U_1 + A(\dot{X} - v_{w;x})}{m} \tag{2a}$$

$$\ddot{Y} = (-cos\psi.sin\phi + sin\psi.sin\theta.cos\phi)\frac{U_1 + A(\dot{Y} - v_{w;y})}{m} \tag{2b}$$

$$\ddot{Z} = -g + (cos\theta.cos\phi)\frac{U_1 + A(\dot{Z} - v_{w;z})}{m} \tag{2c}$$

$$\ddot{\phi} = \frac{I_{YY} - I_{ZZ}}{I_{XX}}\dot{\theta}\dot{\psi} - \frac{J_{TP}}{I_{XX}}\dot{\theta}\Omega + \frac{U_2 + T_\phi}{I_{XX}} \tag{2d}$$

$$\ddot{\theta} = \frac{I_{ZZ} - I_{XX}}{I_{YY}}\dot{\phi}\dot{\psi} - \frac{J_{TP}}{I_{XX}}\dot{\phi}\Omega + \frac{U_3 + T_\theta}{I_{XX}} \tag{2e}$$

$$\ddot{\psi} = \frac{I_{XX} - I_{YY}}{I_{ZZ}}\dot{\phi}\dot{\theta} + \frac{U_4 + T_\psi}{I_{ZZ}} \tag{2f}$$

where, A is the cross-section area of the quadcopter as viewed along the Z-direction in the hovering state, and Ω results due to Gyroscopic torque.

The motor speed response to the input motor voltage (v_n) is modelled by the equation,

$$\dot{\Omega}_n = -\frac{K_E K_M}{R J_{TP}}\Omega_n - \frac{d}{J_{TP}}\Omega_n^2 + \frac{K_M}{R J_{TP}}v_n \tag{3}$$

The values of all the constants are equal to those stated in [10].

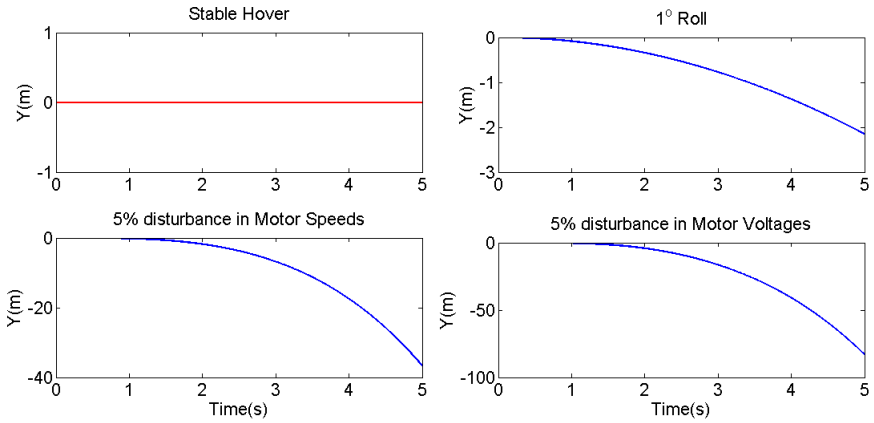


Fig. 1: Effect of small disturbances in the hovering state. Disturbances were provided in the initial roll angle, the initial motor speeds, and the input voltages, as stated in the figure. Even small disturbances have large effects, in this case a drift in the Y-direction.

3 Proposed Classical Control Algorithm

3.1 Necessity of Control

The state of hovering, of a quadcopter, can be easily disrupted by minute disturbances as seen in Fig. 1. To counter the force of gravity and keep its vertical position, the quadcopter has to generate a huge amount of thrust. So, the slightest deviation in the Euler angles or in the set motor speeds will make the quadcopter spin and crash. There is no restoring force or torque that is inherent in the system. Thus, we require a control scheme which can transform any desired velocity setpoint into a point of global, or local, stable equilibrium.

3.2 The Scheme

We consider *near-hover* flight, where the deviation in the euler angles is minute and the angular speeds are small. We neglect the drag force and torque for now as they cannot be measured by the IMU. We will see if this simplification can work even in the cases where the euler angles and angular speeds are large, and in the presence of wind. As we are designing a mechanism to overcome disturbances, we expect that it would work in some extreme cases too.

We can navigate through space by changing two Euler Angles, and keeping the third one constant. Here, we keep the yaw angle constant. This is because it isn't affected by the Gyroscopic torque, and so we can relax the control requirements on it as compared to the other two Euler angles. The roll and pitch motion have similar governing dynamics, and it is easier to design a control scheme around them. So, we would maintain $\psi = 0$, and control the roll and the pitch.

We simplify the equations (2) to,

$$\ddot{X} = \theta\frac{U_1}{m} \qquad \ddot{Y} = -\phi\frac{U_1}{m} \qquad \ddot{Z} = -g + \frac{U_1}{m}, \tag{4a}$$

$$\ddot{\phi} = \frac{U_2}{I_{XX}} \qquad \ddot{\theta} = \frac{U_3}{I_{XX}} \qquad \ddot{\psi} = \frac{U_4}{I_{ZZ}} \tag{4b}$$

The equations (1) and (3) cannot be simplified for Near-Hover flight.

Then, the equations involved in the control scheme are:

$$\Delta\boldsymbol{v} = \boldsymbol{v}_{inst} - \boldsymbol{v}_{set} \qquad \boldsymbol{a}_{req} = -k_a\Delta\boldsymbol{v}, \tag{5a}$$

$$\theta_{req} = \frac{\ddot{X}_{req}}{\ddot{Z}_{req} + g} \qquad \phi_{req} = -\frac{\ddot{Y}_{req}}{\ddot{Z}_{req} + g}, \tag{5b}$$

$$U_{1req} = (\ddot{Z}_{req} + g)m \qquad \boldsymbol{A}_{req} = [\theta_{req} \quad \phi_{req}]^T, \tag{5c}$$

$$\Delta\boldsymbol{A} = \boldsymbol{A}_{inst} - \boldsymbol{A}_{req} \qquad \boldsymbol{\omega}_{req} = -k_o\Delta\boldsymbol{A}, \tag{5d}$$

$$\omega_{\psi;req} = -k_{o\psi}\psi_{inst} \qquad \Delta\boldsymbol{\omega} = \boldsymbol{\omega}_{inst} - \boldsymbol{\omega}_{req}, \tag{5e}$$

$$\Delta\omega_\psi = \omega_{\psi;inst} - \omega_{\psi;req} \qquad \boldsymbol{\alpha} = -k_{ac}\Delta\boldsymbol{\omega}, \tag{5f}$$

$$\alpha_\psi = -k_{ac\psi}\Delta\omega_\psi \qquad \boldsymbol{V} = f.\boldsymbol{\Omega} \circ \boldsymbol{\Omega} + h\boldsymbol{\Omega} \tag{5g}$$

where,

$$\boldsymbol{\Omega} = [\Omega_1 \quad \Omega_2 \quad \Omega_3 \quad \Omega_4]^T \qquad \boldsymbol{\Omega} \circ \boldsymbol{\Omega} = abs(\Gamma) \tag{6a}$$

$$\boldsymbol{\Gamma} = \begin{bmatrix} b & b & b & b \\ -lb & 0 & lb & 0 \\ 0 & -lb & 0 & lb \\ -d & d & -d & d \end{bmatrix}^{-1} \begin{bmatrix} U_{1req} \\ I_{XX}\boldsymbol{\alpha} \\ I_{ZZ}\alpha_{\psi} \end{bmatrix} \qquad \boldsymbol{V} = \begin{bmatrix} V_1 \; V_2 \; V_3 \; V_4 \end{bmatrix}^T \tag{6b}$$

where, $abs(\boldsymbol{v})$ maps the elements of the vector $\boldsymbol{v}$ to their absolute values. The k's are tuning parameters which were optimised manually. The values of the constants k's, f, and h are given in the Appendix. We consider ψ separately because the moment of inertia along the Z'-axis is twice of that along the X' and Y' axes in the body frame. Let us now analyse the overall behavior of the Control Scheme.

3.3 Simulation conditions

We simulate the dynamics of the state in MATLAB, using the Euler method, with a step-size of 0.1 ms. The Control module is executed with the model of the quadcopter dynamics. The update rate of the control module is 100 Hz. The noise in the IMU output is modelled as a Gaussian white noise, filtered with a 5-point moving average. We model the delay in the IMU output by executing the control module at a random time before the normal set time of execution, but outputting the voltage at the normal set time.

Simulations are run for hovering, velocity waypoint navigation, and recovery from angular disturbances. In all the cases, the environment has a noisy wind with mean speed of 3.5 m/s (normal wind speed), and standard deviation of 0.5 m/s. The noisy drag torque has a mean of 0, and standard deviation of 1.35×10^{-3} Nm. The delay in the IMU data is around 0.47 ms [12]. The noise in IMU data is 15% of the signal. The results are explained in the next subsection.

3.4 Results

In Fig. 2, the Trajectory Module provides Velocity Waypoints to the Control Module, and they are implemented perfectly if the accelerations involved do not exceed 0.5 m/s^2 for a long duration. The quadcopter also recovers from any disturbances.

We have developed a non-spike based control scheme which follows velocity waypoints in realistic environmental conditions. Now we will discuss the translation of this control scheme to Spiking Neural Networks.

4 Arithmetic computation using Spiking Neural Networks

AEIF Spiking Neurons: We use the Adaptive Exponential Integrate and Fire model (AEIF) [13] to simulate the dynamics of the neurons in our SNNs, with

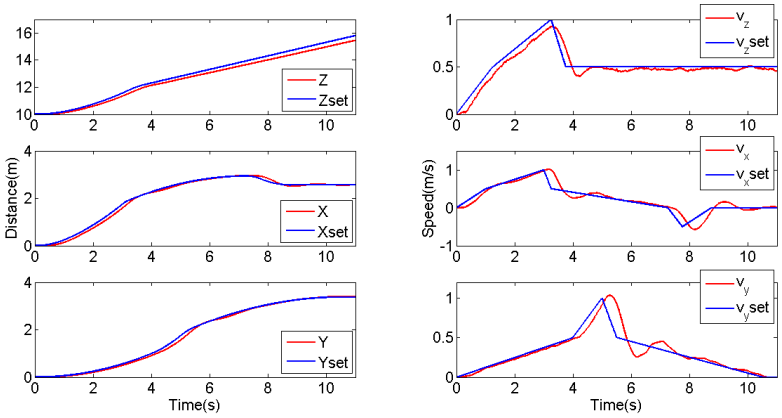


Fig. 2: Velocity Waypoint control. We can robustely follow accelerations upto $0.5\,\mathrm{m/s^2}$. The displecement profile shows that the quadcopter can follow velocity profiles without much error in displacement.

the parameters chosen to mimic regular spiking (RS) neurons. The dynamics of the membrane potential $V(t)$, in response to synaptic current I_{syn} and applied current I_{ext} is described by the equations,

$$C\frac{dV}{dt} = -g_L(V(t) - E_L) + g_L\Delta_T \exp\left(\frac{V(t) - V_T}{\Delta_T}\right) - U(t) + I_{ext}(t) + I_{syn}(t),$$
$$\tau_w \frac{dU}{dt} = a(V(t) - E_L) - U(t), \tag{7a}$$

and when $V(t) \geq 0, V(t) \rightarrow V_r$ and $U(t) \rightarrow U(t) + b$.

Synaptic current due to a spike in the pre-synaptic neuron at time t^f is given by the equation,

$$I_{syn}(t) = I_s w \left[exp\left(-\frac{t - t^f}{\tau_m}\right) - exp\left(-\frac{t - t^f}{\tau_s}\right)\right] h(t - t^f), \tag{8}$$

where w is the weight of the synapse and h is the Heaviside step function.

The instantaneous spike rates are calculated by using the Adaptive Kernel Density Estimation method, as discussed by Shimazaki & Shinomoto [14].

Linearisation of neuronal spike response The spike response of an AEIF neuron to input DC current is non-linear in the region of low spike rate. So, we encode information in a quantity termed as the Spike Info, s, which is given by $s = f - 14.55$, where f is the Spike Rate. Our linearised neuron has a bias input current and a self-inhibitory connection. The output spike info is a linear function of the average input current, and the output synaptic current is a linear function of the output spike info.

A scheme to translate any arithmetic operations into a spiking neural network (SNN) has been discussed earlier by the authors [8]. We are able to carry out the arithmetic operations of addition, subtraction, exponentiation, logarithm, multiplication and division. The linear operations of addition and subtraction involve cascading multiple *linearised* neurons. The non-linear operations are carried out using adaptive synapses with spike-triggered weight update rules.

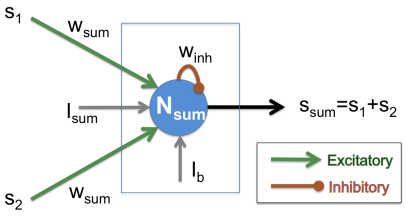


Fig. 3: SNN for Addition

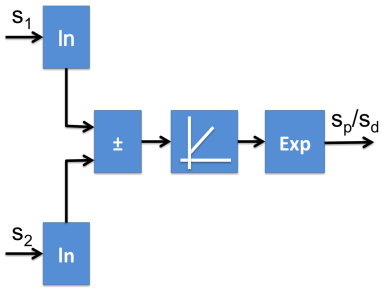


Fig. 4: SNN for Multiplication

The SNNs for addition and multiplication/division are shown in Figs. 3 & 4 respectively. s_1 and s_2 are the input spike infos. s_{sum} denotes the addition, s_p denotes the product, and s_d denotes the division of s_1 and s_2. Examples of addition and multiplication of spike infos are shown in Figs. 5 & 6 respectively. Refer to [8] for further details.

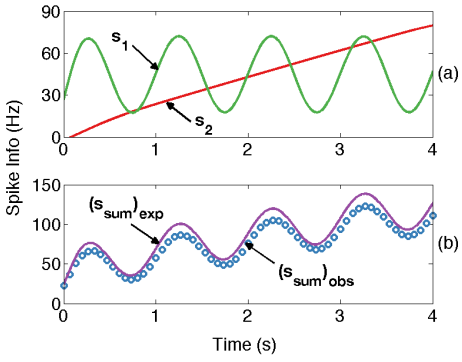


Fig. 5: Addition of Spike Infos

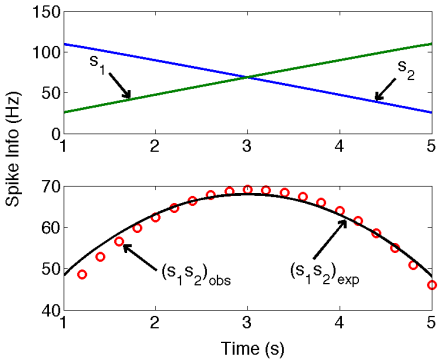


Fig. 6: Multiplication of Spike Infos

5 SNN based Control System

We will now transform the arithmetic operations stated in Eqns. 5 & 6 into SNNs. We have to identify the domains of the inputs $\boldsymbol{v_{set}}, \boldsymbol{v_{inst}}, \boldsymbol{\theta_{inst}}$, and $\boldsymbol{\omega_{inst}}$, and decide appropriate scaling to the spike info domain. *Work in progress.*.

6 Conclusion

We proposed a robust classical control scheme for quadcopter flight control. We built a modular spiking neural network based on that scheme, and analysed its performance as compared to the classical scheme. *Performance details, conclusions - TBD*

References

1. Salih, A.L., Moghavvemi, M., Mohamed, H.A.F., Gaeid, K.S.: Flight PID controller design for a UAV quadrotor. Scientific Research and Essays Vol. 5(23), pp. 3660-3667 (2010)
2. Shepherd, J., Tumer, K.: Robust Neuro-Control for a Micro Quadrotor. GECCO'10, Oregon, USA (2010)
3. Gautam, D., Ha, C.: Control of a Quadrotor Using a Smart Self-Tuning Controller Fuzzy PID Controller. Int. J. Adv. Robot. Syst., Vol. 10, 380:2013 (2013)
4. Bangura, M., Mahony, R.: Nonlinear Dynamic Modeling for High Performance Control of a Quadrotor. ACRA'12, Victoria University of Wellington, New Zealand (2012)
5. Calise, A., Rysdyk, R.: Nonlinear adaptive flight control using neural networks. IEEE Control Systems Magazine, 18(6):1425 (1998)
6. Madani, T., Benallegue, A.: Adaptive control via backstepping technique and neural networks of a quadrotor helicopter. Proceedings of the 17th World Congress of The International Federation of Automatic Control (2008)
7. Maass, W.: Networks of Spiking Neurons: The Third Generation of Neural Network Models. Neural Networks, Vol.10, No.9, Elsevier Science Ltd., Great Britain (1997)
8. Thorat, S., Rajendran, B.: Arithmetic Computing via Rate Coding in Neural Circuits with Spike-triggered Adaptive Synapses. IJCNN'15, Killarney, Ireland (*in-press*)
9. Hoffmann, G., Huang, H., Waslander, S. Tomlin, C.: Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment. AIAA Guidance, Navigation and Control Conference and Exhibit, South Carolina (2007)
10. Bresciani, T.: Modelling, Identification and Control of a Quadrotor Helicopter. Department of Automatic Control, Lund University (2008)
11. Smith, S.: The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Publishing, 2nd edition (1999)
12. Looney, M., Analyzing Frequency Response of Inertial MEMS in Stabilization Systems. Analog Dialogue 46-07 (2012)
13. Brette, R., Gerstner, W.:Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity. J. Neurophysiol., 94:3637-3642 (2005)
14. Shimazaki, H., Shinomoto, S.: Kernel bandwidth optimization in spike rate estimation. J. Comput. Neurosci., 29:171182 (2010)

Arithmetic Computing via Rate Coding in Neural Circuits with Spike-triggered Adaptive Synapses

Sushrut Thorat
Department of Physics
Indian Institute of Technology, Bombay
Maharashtra, India 400076
Email: sushrut.thorat94@gmail.com

Bipin Rajendran
Department of Electrical Engineering
Indian Institute of Technology, Bombay
Maharashtra, India 400076
Email: bipin@ee.iitb.ac.in

***Abstract*—We present spiking neural circuits with spike-time dependent adaptive synapses capable of performing a variety of basic mathematical computations. These circuits encode and process information in the spike rates that lie between $40-140$ Hz. The synapses in our circuit obey simple, local and spike-time dependent adaptation rules. We demonstrate that our circuits can perform the fundamental operations – addition, subtraction, multiplication and division, as well as other non-linear transformations such as exponentiation and logarithm for time dependent signals in real-time. We show that our spiking neural circuits are tolerant to a high degree of noise in the input variables, and illustrate its computational capability in an exemplary signal estimation problem. Our circuits can thus be used in a wide variety of hardware and software implementations for navigation, control and computation.**

I. Introduction

One of the fundamental questions of neuroscience is how basic computational operations could be performed by neural circuits that encode information in the temporal domain. Mathematical operations such as addition and multiplication are central to signal processing, and hence to neural computations. Addition and subtraction are essential for pattern recognition. These operations can alter the number of neurons required to reach threshold and influence the ability to distinguish different patterns of activation [1]. Multiplicative operations occur in a wide range of sensory systems. Computations such as looming stimulus detection in the locust visual system [2], and auditory processing in the barn-owl nervous system [3], rely on multiplication operations. In this paper, we model these operations using Spiking Neural Networks (SNNs) [4], to unravel potential mechanisms underlying these computations in biological systems. The goal of our study is to mathematically describe the connections in a system, and build SNNs to perform the required mathematical operations. These circuits can then be used to build complex networks to model higher order neural functions, and intricate control systems.

There have been various attempts at building SNNs to perform basic mathematical operations. Koch and Seveg detailed the role of single neurons in information processing [8]. Srinivasan and Bernard proposed a mechanism for multiplication which involved detecting coincident arrival of spikes [5], but coincidence detection cannot be used to perform division. Tal and Schwartz used a Log transfer function, generated by creating a refractory period in the Leaky-Integrate-and-Fire neurons, and correlated multiplication with the addition of the logarithms, but they did not calculate the exact multiplication [6]. Inspired by the barn-owl auditory processing capabilities, Nezis & van Rossum built a SNN for multiplication by approximating multiplication by the minimum function, and using a power-law transfer function [7]. All the approaches used voltage to spike-rate (pre-synaptic) non-linear transfer functions to achieve multiplication.

In this paper, we adopt a novel strategy to perform mathematical operations using SNNs. We develop small SNNs with linear transfer functions, and spike-time dependent plastic synapses using simple weight adaptation rules to perform these operations. We introduce a new token of information which we call *Spike Info, s,* to be used to encode information in place of Spike Rate, as required towards the linearisation of spike response to stimuli. We will use the linearized neuron model to develop circuits to perform linear operations such as addition and subtraction. For multiplication and division operations, we use $\exp(\log s_1 \pm \log s_2)$, where s_1 and s_2 are the input spike infos. A similar computational scheme exists in the locust visual system [2], although it uses pre-synaptic non-linear transfer functions for the logarithm and exponentiation operations. We then detail SNNs for performing transformations such as logarithm, exponentiation, multiplication, and division. Note that all these operations are in the spike rate domain, and they work in real-time for biologically plausible signals. We then assess the performance of the developed SNNs, especially its noise resilience, and illustrate its applicability in an exemplary signal estimation problem related to echolocation in real-time. The formalism and the circuits we have developed here can be used to compose large spiking neural circuits for software as well as power efficient hardware implementations.

II. Spiking Neural Network Implementation

We use the Adaptive Exponential Integrate and Fire (AEIF) neuron model [9][14] for modeling the dynamics of the neurons, with the parameters chosen to mimic regular spiking (RS) neurons. The dynamics of the membrane potential $V(t)$, in response to synaptic current I_{syn} and applied current I_{ext} is described by the equations,

$$C\frac{dV}{dt} = -g_L(V(t)-E_L) + g_L\Delta_T \exp\left(\frac{V(t)-V_T}{\Delta_T}\right) - U(t) + I_{ext}(t) + I_{syn}(t), \quad (1a)$$

$$\tau_w\frac{dU}{dt} = a(V(t)-E_L) - U(t), \quad (1b)$$

and when $V(t) \geq 0, V(t) \rightarrow V_r$ and $U(t) \rightarrow U(t) + b$.

Synaptic current due to a spike in the pre-synaptic neuron at time t^f is given by the equation,

$$I_{syn}(t) = I_s w \left[exp\left(-\frac{t-t^f}{\tau_m}\right) - exp\left(-\frac{t-t^f}{\tau_s}\right)\right] h(t-t^f), \quad (2)$$

where w is the weight of the synapse and h is the Heaviside step function.

The values of the parameters of the neuron are listed in Appendix A. We simulate the dynamics of our circuits in MATLAB and we obtain convergent results by using Euler method with a step size of 0.01 ms. The instantaneous spike rates are calculated by using the adaptive Kernel Density Estimation method, as discussed by Shimazaki & Shinomoto [10]. When we do not deal with the pictorial representation of spike rate in real-time (which requires knowledge of the instantaneous spike rate), we identify the time window during the simulation when the inverse of the inter-spike intervals approximately converge, and we average over the time window to find the average spike rate.

III. Linearized Neuron

The spike rate of a RS (AEIF) neuron in response to the applied DC current is shown in Fig. 1. The response is non-linear, starkly more so in the region of low spike rate. To make the response linear, we design a 'linearized neuron', by applying a bias current ($I_b = 265$ nA), and introducing a self-inhibitory synapse ($w_{inh} = -130$) to the AEIF neuron as shown in Fig. 2. The self-inhibitory connection also lets us access a wider domain of input current while keeping the spike rate under 200 Hz.

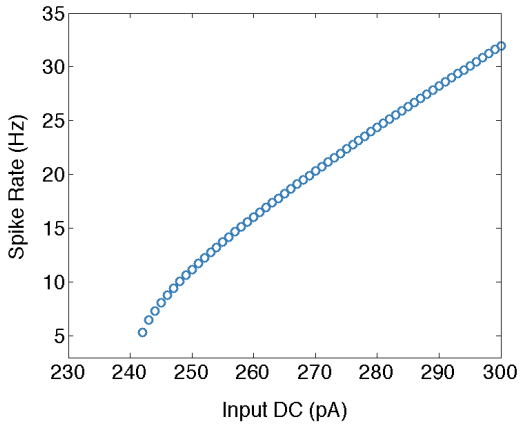


Fig. 1: The spike rate response at lower DC input currents of an AEIF neuron is non-linear. Since the spike rate response at higher DC input currents is almost linear, this regime is better suited for computations.

The spike rate of the linearized neuron (Fig. 3a) at zero input current is now $f_0 = 14.55$ Hz. Considering the information coded in the spike rate of the neuron to be offset by this value, we define a quantity called *Spike Info*, denoted as s, by the relation

$$s = f - f_0, \quad (3)$$

where f is the observed spike rate. Since the output spike rate is more or less constant, the average value of the synaptic

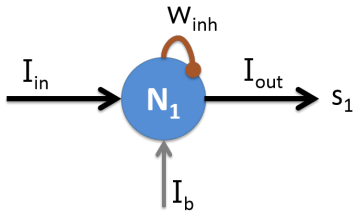


Fig. 2: Our linearized neuron is an AEIF neuron with a self-inhibitory synapse $w_{inh} = -130$ and with a bias current $I_b = 265$ nA.

current flowing out of the output neuron for unit weight is also found to increase linearly with spike rate (Fig. 3b). Hence, the cardinal relations obeyed by the linearized neuron are

$$s_{out} = \alpha I_{in} \quad (4)$$

$$\langle I_{out} \rangle = w\left(\beta s_{out} + \gamma\right) \quad (5)$$

where, $\alpha = 0.2237$ Hz/pA, $\beta = 0.01105$ pA/Hz, and $\gamma = 0.1605$ pA.

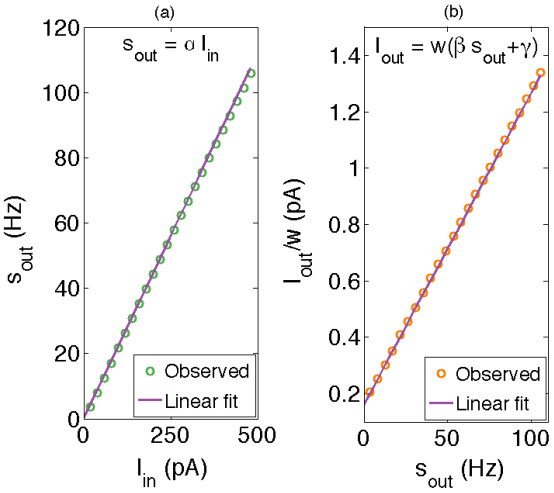


Fig. 3: By using a bias current to avoid the low current region, and introducing a self-inhibitory connection to reduce the residual non-linearity, a linear dependence can be obtained. The bias current also creates a positive offset of $f_0 = 14.55$ Hz in the spike rate. So, we introduce a token of information *Spike Info, s,* defined as $s = f - f_0$.

A. Offseting and Scaling Spike Info

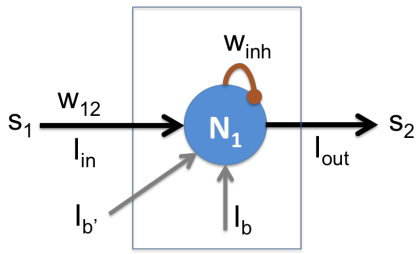


Fig. 4: By adjusting the value of w_{12} and $I_{b'}$, we can independently obtain the transformations $s_2 = s_1 + s_0$ and/or $s_2 = \eta s_1$.

Now we discuss how to design a SNN that offsets a given spike info (by a factor s_0) or scales it (by a factor η) according to the transformation

$$s_{out} = \eta s_{in} + s_0 \quad (6)$$

Referring to the SNN depicted in Fig. 4, s_1 denotes the input spike info and s_2 denotes the output spike info. We will show that by adjusting the values of w_{12} and $I_{b'}$, we can obtain this transformation independently. Let the average current flowing into N_1 due to s_1 be denoted as I_{in}. Then

$$\langle I_{in} \rangle = w_{12}(\beta s_1 + \gamma) \tag{7a}$$

$$s_2 \approx \alpha(\langle I_{in} \rangle + I_{b'}) \tag{7b}$$

Hence,

$$s_2 \approx w_{12}\alpha\beta s_1 + \alpha(I_{b'} + \gamma w_{12}) \tag{8}$$

Note that (7b) is an approximate relation, as $\langle I_{in} \rangle$ is not a DC current. By appropriately tuning the values w_{12} and $I_{b'}$, we can independently add an offset or scale the input spike info (Figs. 5 & 6). The values of these parameters needed for some of these transformations we will use later in the paper are listed in Table I.

TABLE I: Parameters for Scaling and Offset, $s_{out} = \eta s_{in} + s_0$

w_{12}	$I_{b'}$ (pA)	η	s_0 (Hz)	Fig.
195	−46	0.5	0	5
400	−75	1	0	5
865	−188	2	0	5
430	28	0	30	6
445	256	0	80	6
201	44	0.5	21	20, 21
207	123	0.5	40	6, 22
401	115	0.93	48	15

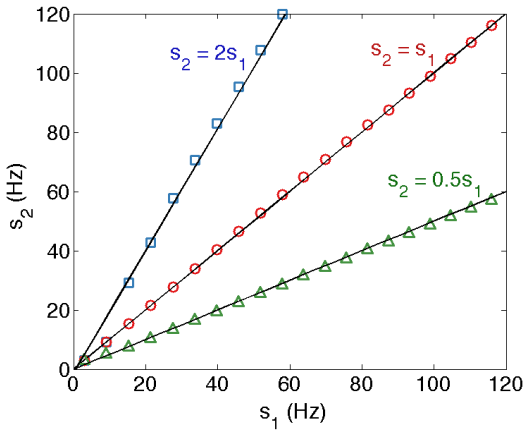


Fig. 5: In the range of $0 - 120$ Hz spike info, it is possible to linearly scale the spike info by tuning the parameters w_{12} and $I_{b'}$ (Table I).

IV. Linear Operations

A. Addition

We now design a SNN which will perform the addition operation in the 'spike info' domain. The circuit is shown in Fig. 7; the input spike infos are fed into a linearized neuron through two identical excitatory synapses w_{sum}. This neuron also receives an additional bias current I_{sum} to correct for offset errors. Hence, the total average current flowing into the adder neuron is

$$\langle I_{inp} \rangle \approx w_{sum}(\beta s_1 + \gamma) + w_{sum}(\beta s_2 + \gamma) + I_{sum} \tag{9}$$

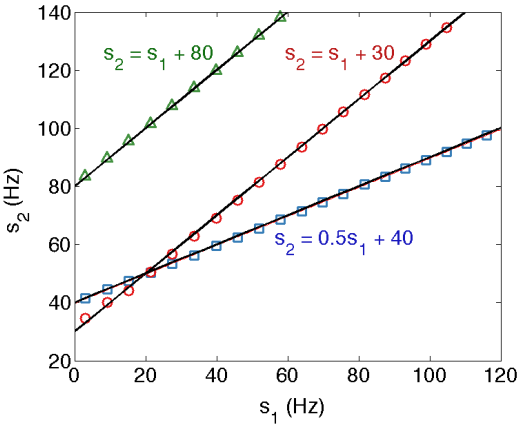


Fig. 6: It is possible to introduce an offset, and/or scale the spike info in the range of $0 - 120$ Hz by adjusting the parameters w_{12} and $I_{b'}$ (Table I).

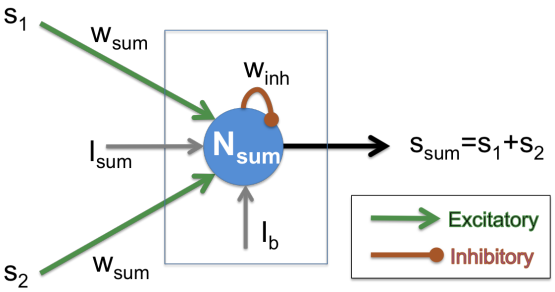


Fig. 7: Sum of spike infos can be computed by feeding the input spikes into a linearized neuron with equal excitatory synapses with $w_{sum} = 405$. The bias current $I_{sum} = -190$ pA ensures that $s_{sum} = s_1 + s_2$.

As before, this is the average current flowing into a linearized neuron. Hence,

$$\begin{aligned} s_{sum} &\approx \alpha I_{inp} \\ &= \alpha\beta w_{sum}(s_1 + s_2) + \alpha(2\gamma w_{sum} + I_{sum}) \end{aligned} \tag{10}$$

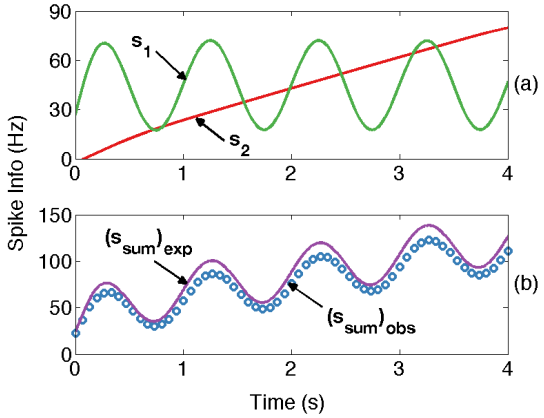


Fig. 8: (a) Exemplary spike infos s_1 and s_2 were generated by feeding sinusoidal and ramp input currents to two linearized neurons. (b) The output generated by the adder has a time dependent spike rate that matches the expected $s_1 + s_2$.

By appropriately choosing the parameter values, (here, $w_{sum} = 405$ and $I_{sum} = -190\,\text{pA}$) we can ensure that $s_{sum} = s_1 + s_2$. The circuit faithfully computes the sum of spike infos in real-time. When a slowly varying sinusoidal spike info and a ramp spike info are applied to the adder, it generates a spike stream in real-time, whose spike info closely matches the expected variation (Fig. 8). The input spike infos used in this study were generated by applying appropriate currents to linearized neurons not shown here.

B. Subtraction

On similar lines, a SNN to determine the difference of spike infos can be obtained as shown in Fig. 9 by feeding the input spikes to a linearized neuron, but one through an excitatory synapse and the other through an inhibitory synapse with identical strengths (w_{diff}). Then,

$$\begin{aligned} s_{diff} &\approx \alpha I_{inp} \\ &= \alpha\beta w_{diff}(s_1 - s_2) + \alpha I_{diff} \end{aligned} \tag{11}$$

A small value of I_{diff} is required to correct for offset errors that arise due to the synaptic currents being time dependent. The SNN shown with $w_{diff} = 405$ and bias current $I_{diff} = -13\,\text{pA}$ computes $(s_1 - s_2)u(s_1 - s_2)$, where u is the Heaviside function. (Fig. 10).

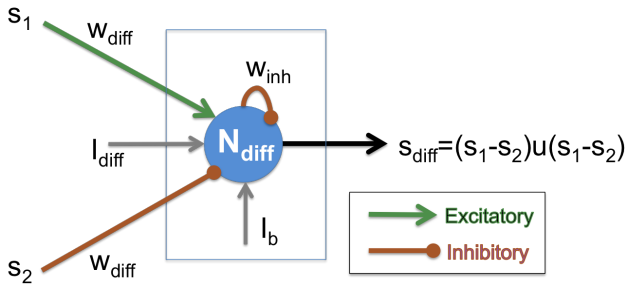


Fig. 9: The SNN circuit for calculating the difference in spike info uses an inhibitory synapse with equal magnitude as the excitatory synapse, with $w_{diff} = 405$. $I_{diff} = -13\,\text{pA}$.

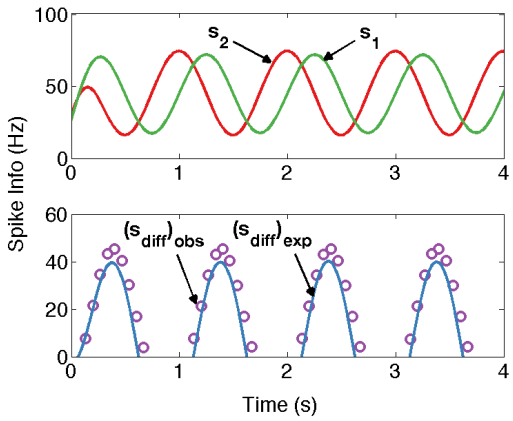


Fig. 10: (a) Exemplary spike infos s_1 and s_2 were generated by feeding sinusoidal input currents to two linearized neurons. (b) The output spike info of the difference SNN matches the expected value $(s_1 - s_2)$ when $s_1 > s_2$. Since all circuits in our scheme process only positive spike info, we have not optimized the circuit to match the response when $s_1 < s_2$.

V. Non-Linear Operations

To complete the repertoire of basic mathematical operations, we now proceed to build SNNs for multiplication and division. One way to implement these operations is by using exponentiation and logarithm as $\exp(\log s_1 \pm \log s_2)$, where s_1 and s_2 are the input spike infos.

A. Logarithm

We will now develop a SNN whose output spike info will be the natural logarithm of the input spike info. We propose that this can be achieved by feeding the input spikes to a linearized neuron with a spike dependent adaptive synapse $w_{ln}(t)$ as shown in Fig. 11. Also note that this linearized neuron is receiving an additional DC bias current I_{ln}.

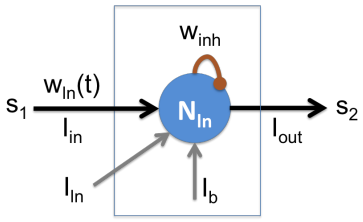


Fig. 11: The adaptive synapse $w_{ln}(t)$ responds dynamically to the input spikes. By choosing appropriate parameters for the update, output spike info can be made proportional to the logarithm of the input spike info.

We require the spike activity of the output neuron to be sub-linearly proportional to the synaptic current. This can be done by imposing the following weight update rule

$$\frac{dw_{ln}}{dt} = -k_0(w_{ln} - w_0) - k_1(I_{in} - I_0), \tag{12}$$

where w_0, k_0, k_1 and I_0 are some constants.

Since the bias current I_{ln} is a constant, (7a) implies that $I_{in} = w_{ln}(t)(\beta s_1 + \gamma)$. Thus, (12) can be written as

$$\frac{dw_{ln}(t)}{dt} = -aw_{ln}(t) - bs_1 w_{ln}(t) + c, \tag{13}$$

where a, b and c are constants. To perform computations in the spike domain, we propose the replacement of $s_1 dt$ with $\sum_{t^s} \delta(t - t^s)dt$, where t^s is the time of arrival of a spike. Now, we employ the weight-adaptation rule

$$w_{ln}(t+\Delta t) - w_{ln}(t) = (c - aw_{ln}(t))\,\Delta t - bw_{ln}(t) \sum_{t^s} \delta(t - t^s)\Delta t \tag{14}$$

The weight adaptation for three different input spike infos is shown in Fig. 12. The weights do not settle down to particular values, but their averages do. It is also seen that the temporal dynamics of the circuit settles down within $0.1 - 0.2\,\text{s}$.

Thus, as the input spike info increases, the rate of increase of average current into N_{ln} decreases, resulting in the logarithmic dependence (Fig. 14). We have designed our logarithmic SNN such that its domain is $20 - 120\,\text{Hz}$ spike info, and in this regime, our logarithmic translator is

$$s_{out} = c_{ln} \log(s_{in}) - f_l \tag{15}$$

where $c_{ln} = 10.73$ and $f_l = 13.73\,\text{Hz}$.

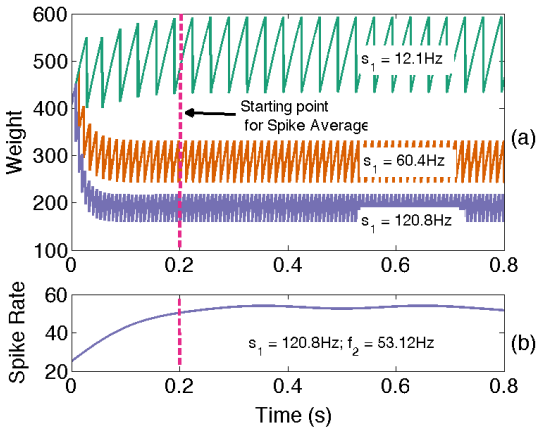


Fig. 12: The synaptic weight w_{ln} dynamically changes in response to the spike info s_1 in Fig. 11. The temporal dynamics of the spike rate at the output neuron N_{ln} is also shown for $s_1 = 120.8\,\text{Hz}$.

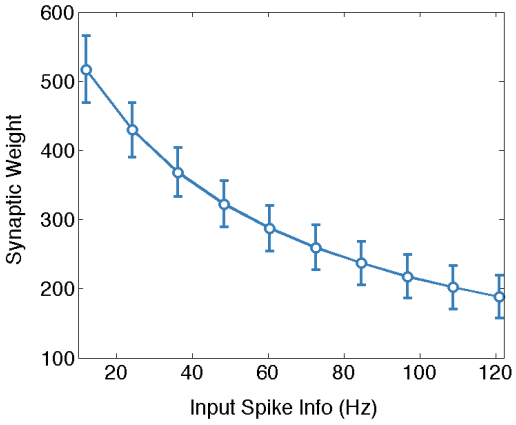


Fig. 13: The mean and standard deviation of the synaptic weight after the logarithmic circuit settles is shown as a function of s_1. The value of w_{ln} decreases with increasing input spike info, and hence the rate of increase of current into N_2 decreases with increasing input spike info, as the current is proportional to the product of the weight and the input spike info.

B. Exponentiation

The SNN to generate an output spike info proportional to the exponential of the input spike info can be designed on similar lines as that of the logarithm, except that we now require the spike activity of the output neuron to be super-linearly proportional to the synaptic current. The weight change rule is hence

$$\frac{dw_{exp}}{dt} = -k_2(w_{exp} - w_1) + k_3(I - I_1), \qquad (16)$$

where k_2, k_3, w_1 and I_1 are constants. The spike-triggered weight update rule we use, is given by

$$w_{exp}(t + \Delta t) - w_{exp}(t) = (c' - a' w_{exp}(t))\,\Delta t + b' w_{exp}(t) \sum_{t^s} \delta(t - t^s) \qquad (17)$$

However, we found that to accurately reproduce the exponential translation, it is necessary to offset the input spike

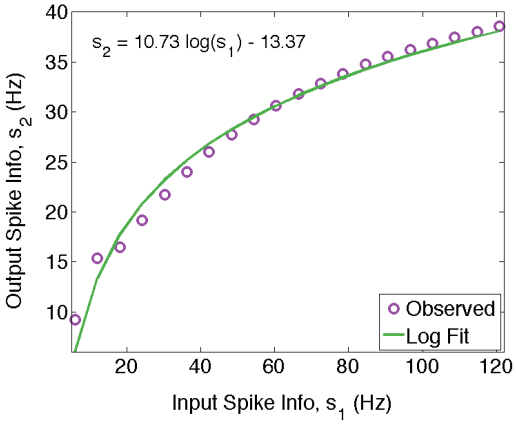


Fig. 14: The logarithmic circuit generates a spike stream, whose spike info is proportional to the logarithm of the spike info of the input spike stream (ranging from $20 - 120\,\text{Hz}$).

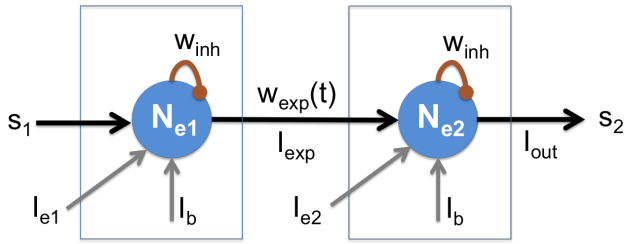


Fig. 15: The adaptive synapse $w_{exp}(t)$ responds dynamically to the input spikes. By choosing appropriate parameters for the update, output spike info can be made proportional to the exponential of the input spike info.

info by $48\,\text{Hz}$ and scale it by a factor 0.93. Hence, the exponentiation operation requires two neurons as shown in Fig. 15.

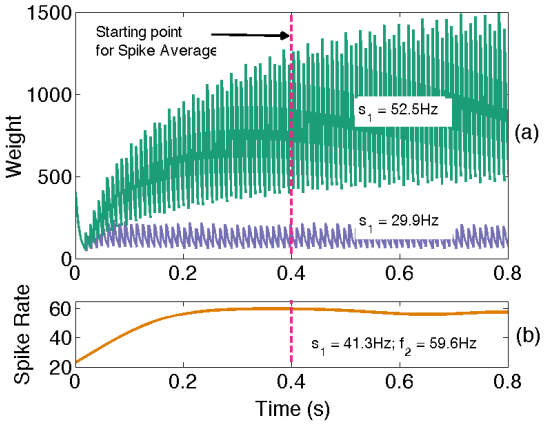


Fig. 16: The synaptic weight w_{exp} dynamically changes in response to the spike info s_1 in Fig.15 (top). The temporal dynamics of the spike rate at the output neuron N_{e2} is also shown for $s_1 = 41.3\,\text{Hz}$.

Dynamics in the weight update follow the expected trend (Fig. 16), and the mean value of the synaptic strength now increases super-linearly with input spike info (Fig. 17). The output spike info at N_{e2} is exponentially proportional to the the input spike info in the domain of $30 - 50\,\text{Hz}$ (Fig. 18). We also verified that the norm of residuals obtained by fitting the

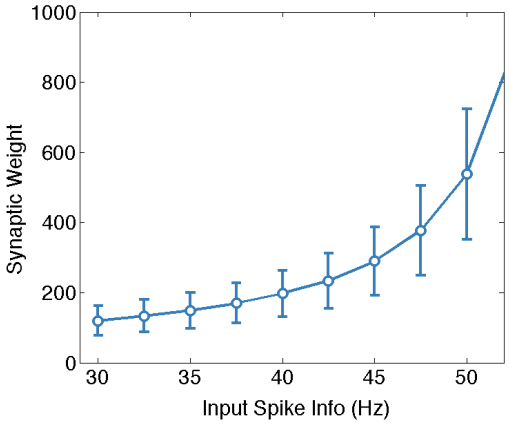


Fig. 17: The mean and standard deviation of the synaptic weight after the exponential circuit settles is shown as a function of s_1. The value of w_{exp} increases with increasing input spike info, and hence the rate of increase of current into N_2 increases with increasing input spike info.

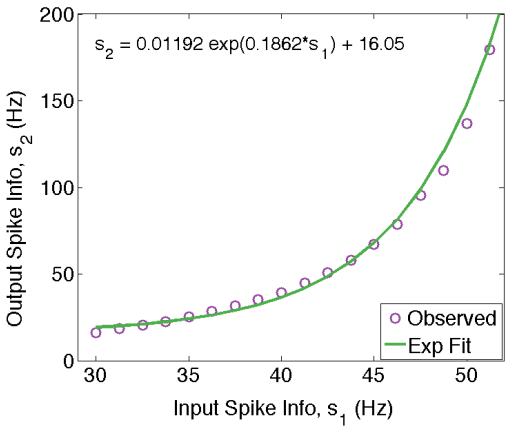


Fig. 18: The exponentiation circuit generates a spike stream, whose spike info is proportional to the exponential of the spike info of the input spike stream (ranging from $30-50$ Hz).

observed data with a square dependence is about a factor of 10 higher than the corresponding value for the exponential fit shown in Fig. 18.

We have designed our exponential SNN such that its domain is $30-50$ Hz spike info, and in this regime, our exponential translator is

$$s_{out} = \alpha_{exp} \exp(c_{exp} s_{in}) + s_{exp} \tag{18}$$

where $c_{exp} = 0.1862$, $\alpha_{exp} = 11.92 \times 10^{-3}$ and the offset $s_{exp} = 16.05$ Hz.

C. Multiplication & Division

The blocks to compute ln and exp functions can now be combined with the adder circuit, in principle, to generate circuits to perform multiplication and division. However, when cascading these circuits, since $c_{exp} \times c_{ln} = 1.99$, we have to incorporate a SNN circuit to scale the spike info of the output of the dder/difference circuit by a factor of 2. Also, since the range of the adder/difference circuit has to lie within the domain of the exponential circuit, we use a spike info translator. A block diagram of the complete circuit is given in Fig. 19.

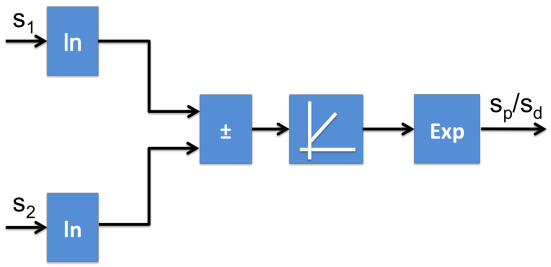


Fig. 19: The SNN for multiplying/dividing two spike infos involve the logarithmic converter and the adder/difference SNN. This output is then scaled and offset before being passed to the exponential SNN. Hence, six neurons are required for multiplication and division.

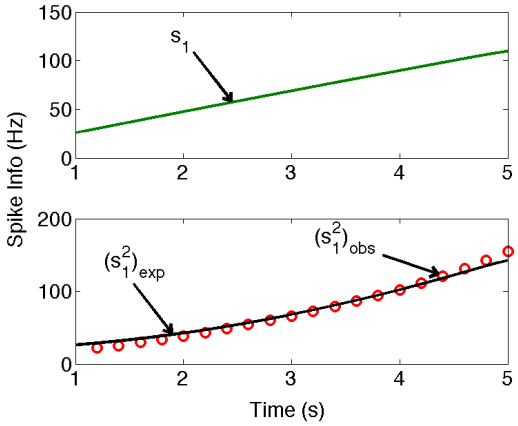


Fig. 20: When the multiplier circuit is fed with two identical spike trains whose frequency increases linearly with time, it generates an output spike train whose spike info increases quadratically, in the range of $20-120$ Hz.

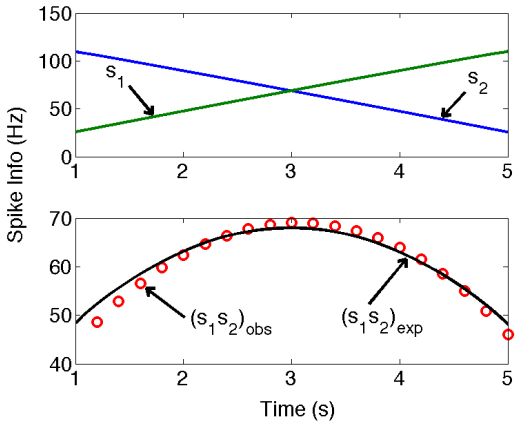


Fig. 21: When the multiplier circuit is fed with an increasing and decreasing ramp spike info, it generates an output spike that closely follows the expected parabolic product.

The response of the multiplier circuit for two slowly varying spike info signals is shown in Figs. 20 and 21. In both cases, the spike info of the output spike train changes quadratically with time and there is excellent match between the computed and expected values. The offset/scaling circuit used in the multiplier employs $\eta = 0.5$ and $s_0 = 21$ Hz; the circuit parameters are given in Table I.

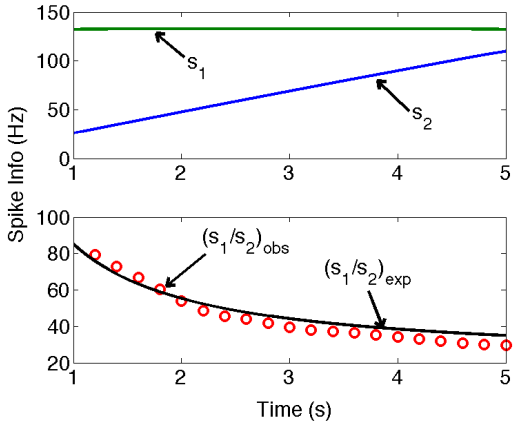


Fig. 22: Using the SNN for division, we demonstrate that it is possible to generate a spike train whose spike info varies inversely proportional to the spike info of the input spike train.

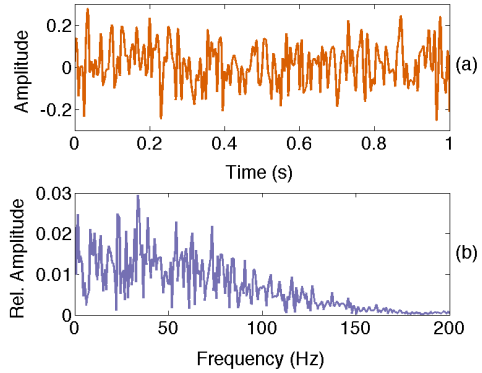


Fig. 23: An example of slowly-varying noisy signal (with std. dev: 0.1) and its DFT. This is used for noise-resilience tests with the SNNs.

Similarly, we demonstrate the performance of the SNN for division, by showing that it is possible to generate a spike train whose spike info varies inversely proportional to the spike info of the input spike train (Fig. 22). One of the inputs used in this experiment was a spike train with constant spike info, and the other had a linearly increasing spike info. The offset/scaling circuit now employs $\eta = 0.5$ and $s_0 = 40$ Hz.

VI. Processing Noisy Signals

We now study the performance of our SNN in the presence of noise in the input signals. We generated noisy signals with frequency components no bigger than 200 Hz (the maximum frequency we might encounter in the listed SNNs), by generating random values every 5 ms, and interpolating between them using piece-wise cubic hermite interpolation (pchip) for the time-step of 0.01 ms. An example of a slowly-varying noisy signal with standard deviation of 0.1 is shown with its Discreet Fourier Transform (DFT) [11], in Fig. 23. We characterize the noisy signal with a Coefficient of Variation (CV) which is given by $c_v = \sigma/\mu$, where σ is the standard deviation, and μ is the mean of the analog current used to generate the input spike info.

We generated a slowly-varying noisy current with a CV of 0.1, and provided it as an input to the SNNs for determining

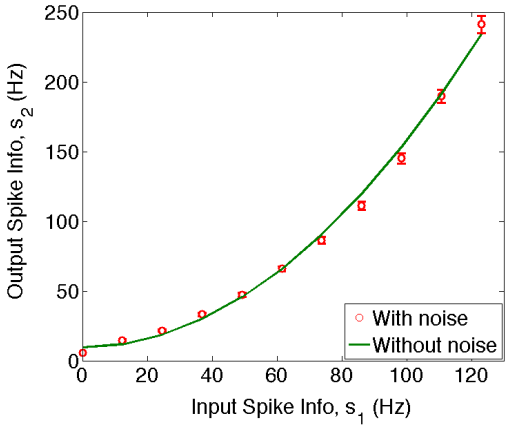


Fig. 24: Variation in the average output spike info when the multiplier SNN is fed with a spike stream generated using noisy analog current (CV: 0.1), as compared with the noise-less scenario.

the square of the input spike info, and simulated the response of the multiplier SNN for 100 experiments. The mean value of the output spike info in response to the noisy input waveform closely follows the expected spike trains without noise to a high degree of accuracy, as shown in Fig. 24. The SNN can tolerate noisy inputs with CV less than 0.2.

VII. Application to Signal Estimation

As an exemplary application of these SNN circuits, we demonstrate the use of the multiplier SNN for range detection. We are inspired by the signal processing involved in echolocation that is widely used by a variety of birds and animals to estimate the distance to preys and obstacles. For instance, its known that echolocating bats send out time varying sound signals which are reflected by preys or obstacles, and there are specific neural circuits that can detect the differences in the interference patterns caused by the emitted and received sounds [12]. However, it is not clear how spiking neural circuits can determine the differences in interference patterns between the emitted and received signal in real time.

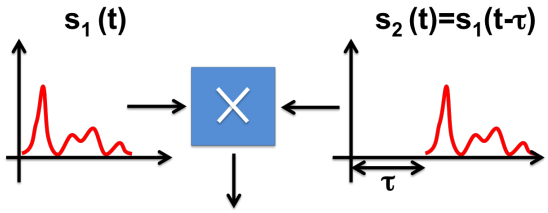


Fig. 25: A multiplier circuit that takes as its inputs a time varying spike train and its delayed version can produce an output spike train whose maximum frequency will encode τ.

We propose a simple model that uses time varying spike frequency signals and the multiplier SNN circuit that can detect the time delay between emitted and received signal by real-time signal processing. The multiplier SNN is fed by a time varying spike signal and a time delayed version of the same signal, as shown in Fig. 25. We assume that $s_1(t)$ is responsible for the creation of the emitted signal (for example, the neuron stimulating the larynx) and $s_2(t)$ is response of the neuron at the detection end (for instance, the neuron in the ear). We

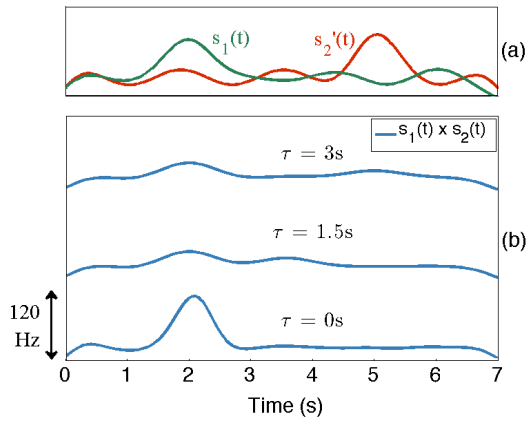


Fig. 26: The emitted signal, s_1, and the signal received after a delay of 3 s, s_2', are shown in (a). The products of s_1 and s_2 for various values of delay, τ, are presented in (b). As the time delay decreases, the overlap of the signal increases, and the spike frequency of the product increases.

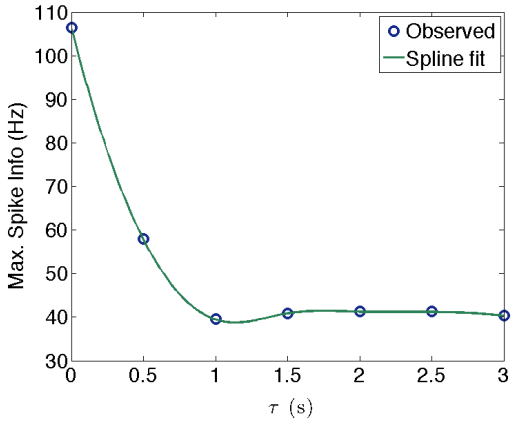


Fig. 27: The maximum spike frequency at the output of the multiplies depends strongly on the time delay between the emitted and reflected signal.

assume that there is some jitter and noise in the channel, hence the reflected signal is not the exact replica of the emitted signal. We monitor the peak spike rate at the output of the multiplier for various values of time delay between the emitted and received spike trains. The multiplication of the emitted signal and received signal for various time delays can be seen in Fig. 26. As can be seen in Fig. 27, the maximum spike frequency at the output of the multiplier SNN is a strong function of the time delay. So a network containing neurons with different axonal delays at the transmitter and a multiplier SNN could be used to determine the distance to the reflecting object in real time with spike based processing.

VIII. Conclusion

We presented a methodology to perform spike based arithmetic computations in neural circuits with spike-time dependent adaptive synapses based on rate coding. Our circuits perform the basic arithmetic operations on analog signals which result in time dependent spike rates in the range of $40-140$ Hz when fed to spiking neurons. The synapses in our circuit obey simple, local and spike time dependent adaptation rules. The building blocks we have designed in this paper can perform the fundamental operations – addition, subtraction, multiplication and division, as well as other non-linear transformations such as exponentiation and logarithm for such time dependent signals in real-time. We have demonstrated that these circuits can reliably compute even in the presence of highly noisy signals. We have illustrated the power of these circuits to perform complex computations based on the frequency of spike trains in real-time, and thus they can be used in a wide variety of hardware and software implementations for navigation, control and computing. Though our circuits use the AEIF neuron model, the outlined design methodology can be readily used to design SNN circuits that use other spiking neuron models.

Acknowledgment

This work was partially supported by the Department of Science and Technology, Government of India.

Appendix

A. Parameters used for simulation

1) Adaptive Exponential Integrate-and-Fire neuron model: Regularly spiking AEF neuron parameters used in this study are $C_m = 200\,\text{pF}$; $g_L = 10\,\text{nS}$; $E_L = -70\,\text{mV}$; $V_T = -50\,\text{mV}$; $\Delta_T = 2\,\text{mV}$; $a = 2\,\text{nS}$; $b = 0\,\text{pA}$; $\tau_w = 30\,\text{mS}$ and $V_r = -58\,\text{mV}$.

References

[1] SIlver RA, *Neuronal Arithmetic*, 2010, Nature Reviews Neuroscience, 11, 474-489.

[2] Gabbiani F, Krapp HG, Koch C, Laurent G, *Multiplication and stimulus invariance in a looming-sensitive neuron*, 2004, J Physiol Paris, 98(1-3):19-34.

[3] Peña JL, Konishi M, *Auditory spatial receptive fields created by multiplication*, 2001, Science, 292(5515):249-52.

[4] Maass W, *Networks of Spiking Neurons: The Third Generation of Neural Network Models*, 1976, Neural Networks, Vol. 10, No. 9, pp. 1659-1671.

[5] Srinivasan MV, Bernard GD, *A Proposed Mechanism for Multiplication of Neural Signals*, 1997, Biol. Cybernetics, 21, 227–236.

[6] Tal D, Schwartz EL, *Computing with the leaky integrate-and-fire neuron: logarithmic computation and multiplication*, 1997, Neural Comp, 9:305318.

[7] Nezis P, van Rossum MCW, *Accurate multiplication with noisy spiking neurons*, 2011, J. Neural Eng, 8, 034005.

[8] Koch C, Segev I, *The role of single neurons in information processing*, 2000, Nature Neuroscience, 3, 1171-1177.

[9] Brette R, Gerstner W, *Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity*, 2005, J Neurophysiol, 94:3637-3642.

[10] Shimazaki H, Shinomoto S, *Kernel bandwidth optimization in spike rate estimation*, 2010, J Comput Neurosci, 29:171182.

[11] Smith S, *The Scientist and Engineers Guide to Digital Signal Processing*, 1999, California Technical Publishing, 2nd edition.

[12] Ulanovsky N, Moss C, *What the bat's voice tells the bat's brain*, 2008, PNAS USA 105(25):8491-8.

[13] Ahmed FYH, Yusob B, Hamed HNA, *Computing with Spiking Neuron Networks A Review*, 2014, Int. J. Advance. Soft Comput. Appl., Vol. 6, No. 1.

[14] Clopath C, Jolivet R, Rauch A, Lüscher HR, Gerstner W, *Predicting neuronal activity with simple models of the threshold type: Adaptive Exponential Integrate-and-Fire model with two compartments*, 2006, Neurocomputing, doi:10.1016/j.neucom.2006.10.047