electronics

MDPI

Article

Synthesis and Analysis of the Fixed-Point Hodgkin–Huxley Neuron Model

Valery Andreev [1], Valerii Ostrovskii [1], Timur Karimov [1], Aleksandra Tutueva [1], Elena Doynikova [2] and Denis Butusov [3,*]

[1] Department of Computer-Aided Design, St. Petersburg Electrotechnical University "LETI", 197376 St. Petersburg, Russia; vsandreev@etu.ru (V.A.); vyostrovskii@etu.ru (V.O.); tikarimov@etu.ru (T.K.); avtutueva@etu.ru (A.T.)

[2] Laboratory of Computer Security Problems, Saint Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, 199178 Saint Petersburg, Russia; doynikova@comsec.spb.ru

[3] Youth Research Institute, St. Petersburg Electrotechnical University "LETI", 197376 Saint Petersburg, Russia

* Correspondence: dnbutusov@etu.ru; Tel.: +7-950-008-7190

Received: 3 February 2020; Accepted: 3 March 2020; Published: 5 March 2020

Abstract: In many tasks related to realistic neurons and neural network simulation, the performance of desktop computers is nowhere near enough. To overcome this obstacle, researchers are developing FPGA-based simulators that naturally use fixed-point arithmetic. In these implementations, little attention is usually paid to the choice of numerical method for the discretization of the continuous neuron model. In our study, the implementation accuracy of a neuron described by simplified Hodgkin–Huxley equations in fixed-point arithmetic is under investigation. The principle of constructing a fixed-point neuron model with various numerical methods is described. Interspike diagrams and refractory period analysis are used for the experimental study of the synthesized discrete maps of the simplified Hodgkin–Huxley neuron model. We show that the explicit midpoint method is much better suited to simulate the neuron dynamics on an FPGA than the explicit Euler method which is in common use.

Keywords: nonlinear dynamics; scaling; discrete chaotic systems; fixed-point arithmetic; Hodgkin–Huxley model; FPGA

1. Introduction

The implementation of artificial neural networks (ANNs) in modern electronic devices requires different network topologies depending on the solving tasks, which vary from clustering and classification to pattern recognition. Based on the available training data, the learning model for the network can be supervised or unsupervised. The first principle, relying on labeled datasets, takes place in Convolutional Neural Networks (CNNs), which have recently shown impressive performance in cognitive tasks such as recognition [1,2] and prediction [3,4]. During supervised learning, the error between the input target and the output of a CNN is minimized by adjusting the synaptic weights of the network. High accuracy can only be achieved with a large number of training examples and algorithm cycles, which entails the application of massive-power-consuming computers. This imposes a significant restriction on the application of CNNs in embedded systems, motivating the development of neural networks of the third generation—Spiking Neural Networks (SNNs). SNNs, being unsupervised learners, provide the most realistic natural neural network emulation. Like neurons in living organisms, SNNs encode information as sequences of spikes, the precise timing between which is used to update the synaptic weights. The main advantages of SNNs over the previous-generation ANNs are their computational speed, superior classification abilities, and efficiency in control problems,

resulting from their ability to derive meaning from few pieces of information about the target. The most promising practical application of SNNs is the construction of an interface between silicon and biological neurons, which in future could bring the ability of direct human brain–computer interaction and the development of bionic prosthetic systems, such as thought-driven limbs and neural prostheses for restoring cognitive functions [5,6].

Meanwhile, the limits of a standard computer's performance for SNN simulation require special hardware for its acceleration [6]. A perfect platform for implementing neural networks must be massively parallel, algorithmically flexible, and of low power consumption. Massive parallelism can be achieved with either an analog or digital simulation approach. Analog simulations of neurons using silicon-based Very-Large-Scale Integration (VLSI) circuits were pioneered by Mead [7] in the early 1980s with a focus on their low power consumption when compared to digital systems [8,9]. Modern analog circuits for the implementation of neural oscillators and synaptic memory often include memristive [10,11] or other experimental CMOS-compatible devices. Despite all these advantages, a common drawback of analog neuromorphic circuits is the fundamental limitations on measurement: it is impossible to organize the monitoring of all state variables for each neuron and, therefore, flexible control of these variables. Thus, from the algorithmic flexibility point of view, neural network simulation based on digital devices remains relevant. Although advances in GPU-based acceleration of neural network simulation have been reported [12], most researchers consider FPGAs to be the better-fitting digital platform. The advantages of FPGAs, including great flexibility, low power consumption, the ability to work in real time, and small dimensions, have already facilitated the use of SNN algorithms for embedded systems, for example, for MIMO temperature management [13] or the detection of impurities in natural gas [14].

The use of FPGA implies two features: the use of discrete models instead of continuous and the use of fixed-point arithmetic. To conserve the neural model properties, it is necessary to choose a numerical method that will ensure good correspondence between the dynamics of a continuous model and those of a discrete model, including the time step, under conditions of limited number representation accuracy. Nowadays, researchers pay little attention to these implementation features. For example, in the FPGA-based SNN simulator made by Pani et al. [6], the simplest solver, that is, the explicit Euler method, is used to implement the Izhikevich neuron. Meanwhile, in a comprehensive study on SNN numerical simulation [15], strong evidence was presented indicating that the use of first-order numerical methods with large steps leads to totally incorrect neuron model dynamics. A very recent article [16] substantiated both the interest in fixed-point arithmetic and the need to perform fixed-point neuron modeling via a specific approach.

This paper provides the results of an investigation of the neuron model described by the simplified Hodgkin–Huxley equations [17] in a fixed-point implementation. In Section 2, we propose the neuron fixed-point model, as well as the data type conversion (scaling) technique. In Section 3, the numerical experiments, including resonance excitability analysis, chaotic spiking generation analysis, and examination of the neural refractory period and hysteresis, are described. Simulations were carried out in NI LabVIEW 2019 software. Section 4 concludes the paper. A comparative table and some recommendations are given here.

2. Numerical Simulation of the Simplified Hodgkin–Huxley Model Using Fixed-Point Arithmetic

The original system of Hodgkin–Huxley (HH) equations [18] is a classical phenomenological neuron model that determines the dynamical behavior of membrane ion gates. This dynamical system is of the fourth order and includes transcendental functions that make it time-consuming for large-scale computer simulations and complicated for pure mathematical analysis. Insightful simplifications of the HH model to two-dimensional systems were presented by Rinzel [19] and later by Wilson [17]; the second one is used in this study.

An equivalent electrical circuit for the simplified HH model is shown in Figure 1. The circuit comprises membrane capacitance C and two voltage-sensitive conductive elements G_{Na} and G_{K}, accordingly connected in series with batteries E_{Na} and E_{K}.

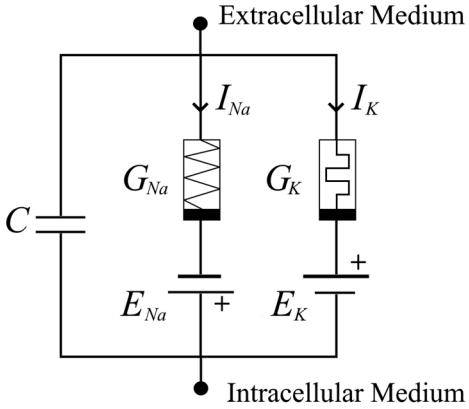


Figure 1. A compact equivalent circuit for the simplified Hodgkin–Huxley (HH) model.

The circuit dynamics is described by the following differential equations:

$$\begin{array}{c} C\frac{dV}{dt} = -(17.81 + 47.71V + 32.63V^2)(V - 0.55) - 26.0R(V + 0.92) + I \\ \frac{dR}{dt} = \frac{1}{\tau}(-R + 1.35V + 1.03) \end{array} \quad (1)$$

where V is the potential difference between the neuron's membrane and the environment, R is the recovery variable, I is the input current, C = 0.8 μF/cm2 is the membrane capacity, and τ = 1.9 ms is the recovery time constant. The right-hand side of the first equation is the sum of the input current and Na+ and K+ ion currents. The passive leakage current of the original HH model is absorbed into the Na+ current. The second equation represents the behavior of the recovery variable R, which describes the K+ channel as a memristive element.

To move from a continuous model of dynamical system (1) to the set of investigated ordinary differential equation (ODE) solvers, the following methods of numerical integration were used: the Explicit Euler method (EE), the Semi-Explicit Euler method (SEE), the Explicit Midpoint method (EMP), and the Modified Explicit Midpoint method with a smoothing step (MEMP). The choice of explicit methods of the first and second order was determined by the simplicity of their implementation in integer representation and the visibility of the observed numerical effects.

2.1. Floating to Fixed Point Model Conversion

Conversions of ODE solvers with floating points to integer solvers were implemented using the approach described in [20]. First, the minimum and maximum possible values of each state variable of system (1) were determined by preliminary simulation. After that, the largest modulus value was selected among all state variables and system parameters to determine the required number of bits to store the integer part of the fixed-point data type (FXP). It was shown that to store the integer part of the state variables and parameters required not less than six bits. Thus, all state variables and constant coefficients of system (1) were converted to the FXP data type, where one bit is allocated for a sign, seven bits are allocated for storing the integer part, and the remaining bits are allocated for the fractional part. It should be noticed that the number of bits of the integer part was increased by one in order to guarantee an overflow avoidance of the bit grid during calculations. In the research, integer models of 32-bit and 64-bit FXP data types were explored, the fractional parts of which take 24 and 56 bits, respectively. Investigation of models with a longer bit grid is not of interest at the moment, since modern computing platforms do not support arithmetic operations on a hardware level for machine words longer than 64 bits. For the same type, the software implementation of these

operations requires additional hardware costs that negate all the benefits of using an integer data type. It would be advantageous to get 16-bit FXP system models, which would allow the use of low-power and less-expensive hardware platforms for implementing large neural networks. However, preliminary simulation showed that system (1) cannot be adequately represented in a case of such limited bit length.

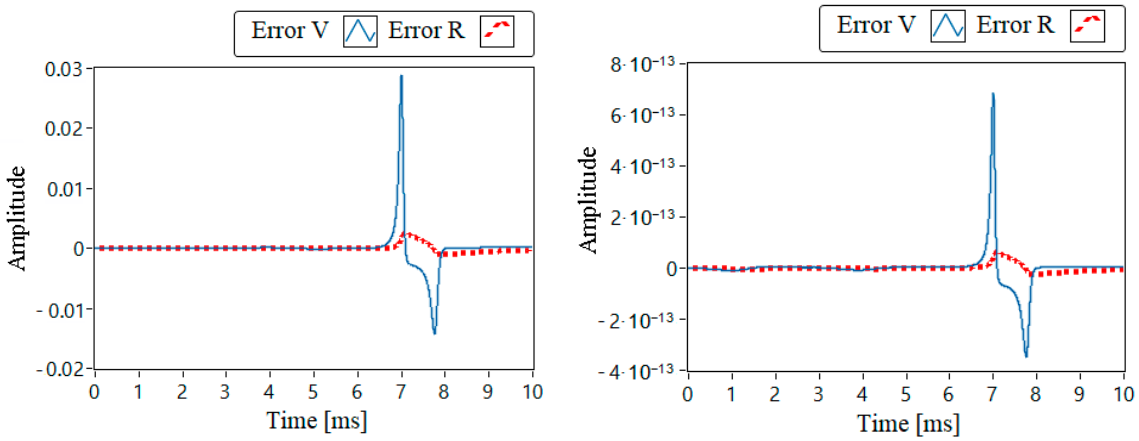
After converting all variables and constants to the FXP data type, adequate simulation is still not guaranteed. In the process of calculations, the values of state variables can overflow the bit grid. This situation is most probable while calculating the $(17.81 + 47.71V + 32.63V^2)$ part of the first equation of system (1). This was considered while generating the FXP solvers and compensated by organizing the correct order of calculation. For example, the algorithm suitable for the integer implementation of the ODE solver of system (1), constructed on the basis of the Euler method, looks as follows:

$$\begin{gathered} V_{(i+1)} = V_{(i)} + (-\tfrac{h}{C}17.81 - \tfrac{h}{C}47.71V - \tfrac{h}{C}32.63V^2)(V - 0.55) - \tfrac{h}{C}26.0R(V - 0.92) + I\tfrac{h}{C} \\ R_{(i+1)} = R_{(i)} - \tfrac{h}{\tau}R + 1.35\tfrac{h}{\tau}V + 1.03\tfrac{h}{\tau} \end{gathered} \quad (2)$$

where h is the constant integration step size, and i is the solution time.

2.2. Accuracy of the Fixed Point Simulation

Let us estimate the accuracy of various implementations of system (1) on the basis of the difference between the values of the state variables of the 32-bit and 64-bit FXP models in comparison to the floating-point model (double, DBL) in the time domain. Figure 2 provides charts of absolute error for the EMP-based solvers.



(**a**) (**b**)

Figure 2. The absolute error of (**a**) 32-bit fixed-point (FXP) and (**b**) 64-bit FXP models compared with that of the double (DBL) model.

The simulation was performed with the parameters given in equation (1). The initial conditions for all models are the same: $C_{(0)} = -0.65$, $R_{(0)} = 0.097$. The integration step size is $h = 0.001$. The input current was set according to the following law: $I = 0.075 + 0.007 \sin[2\pi\ 0.2646(i + h)]$. The models based on the EE, SEE, and MEMP methods demonstrate similar behavior and errors in the case of the same simulation parameters. With increasing simulation time, the effects of the accumulation of differences between the FXP and DBL models can be observed (Figure 3).

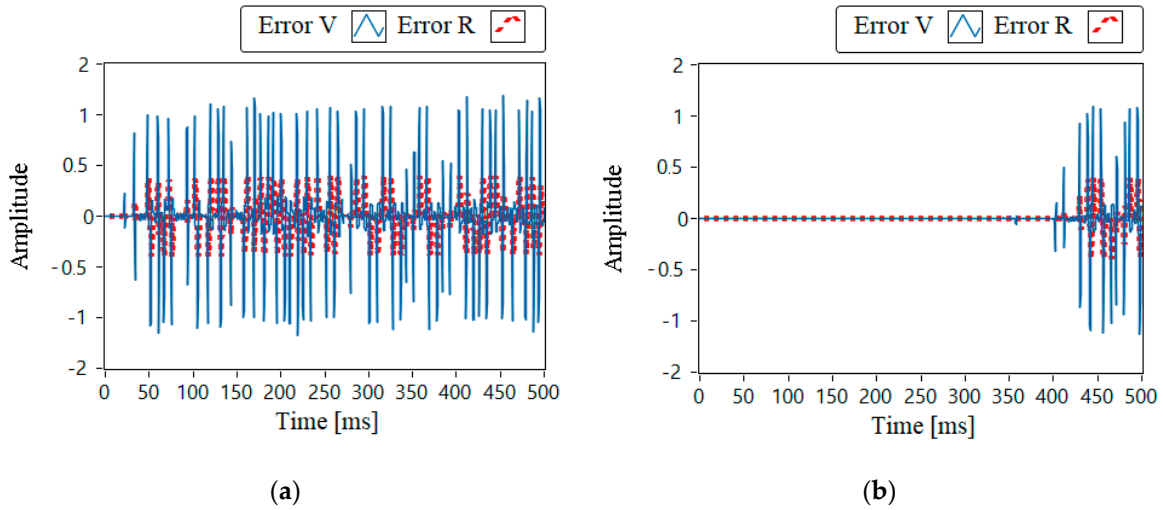


Figure 3. The absolute error of (**a**) 32-bit FXP and (**b**) 64-bit FXP models compared with that of the DBL model in the case of long simulation time.

Figure 3 shows that error accumulation leads to a complete divergence between the trajectories of the two models. For the 32-bit EMP-based solver, it occurs after 30 ms of simulation, while the 64-bit solver switches to a different operational mode after 400 ms. It should be noted that the process of distancing the trajectory of the integer solution from the trajectory of the solution of the original algorithm is inevitable not only due to the limitations of the bit grid but also because of the different order of arithmetic operations in the solvers. However, the stability of the solution is retained, as was confirmed in a series of experiments.

3. Results

3.1. Resonant Spike Generation

The simplified HH neuron model can demonstrate a stationary mode, damped subthreshold oscillations, and spike generation mode. Therefore, considering Izhikevich's classification [21], the HH model is a bistable resonator.

In Figure 4 the represented dynamical maps demonstrate the behavior of system (1), which was implemented using the solvers under investigation, when applying a pulse signal to its input with amplitude I = 4.5 μA and different values of period T and pulse width w. The period T ranged from 20 to 40 ms with step ΔT = 0.05, and the pulse width ranged from 10 to 30 ms with step Δw = 0.05. The white areas on the diagrams of Figure 4 correspond to the stationary mode and the mode of subthreshold oscillations, while blue areas correspond to the spiking mode for the considered 32-bit and 64-bit FXP and DBL solvers built using the following methods: EE, SEE, EMP, and MEMP. The integration step size for all experiments was the same h = 0.01 ms.

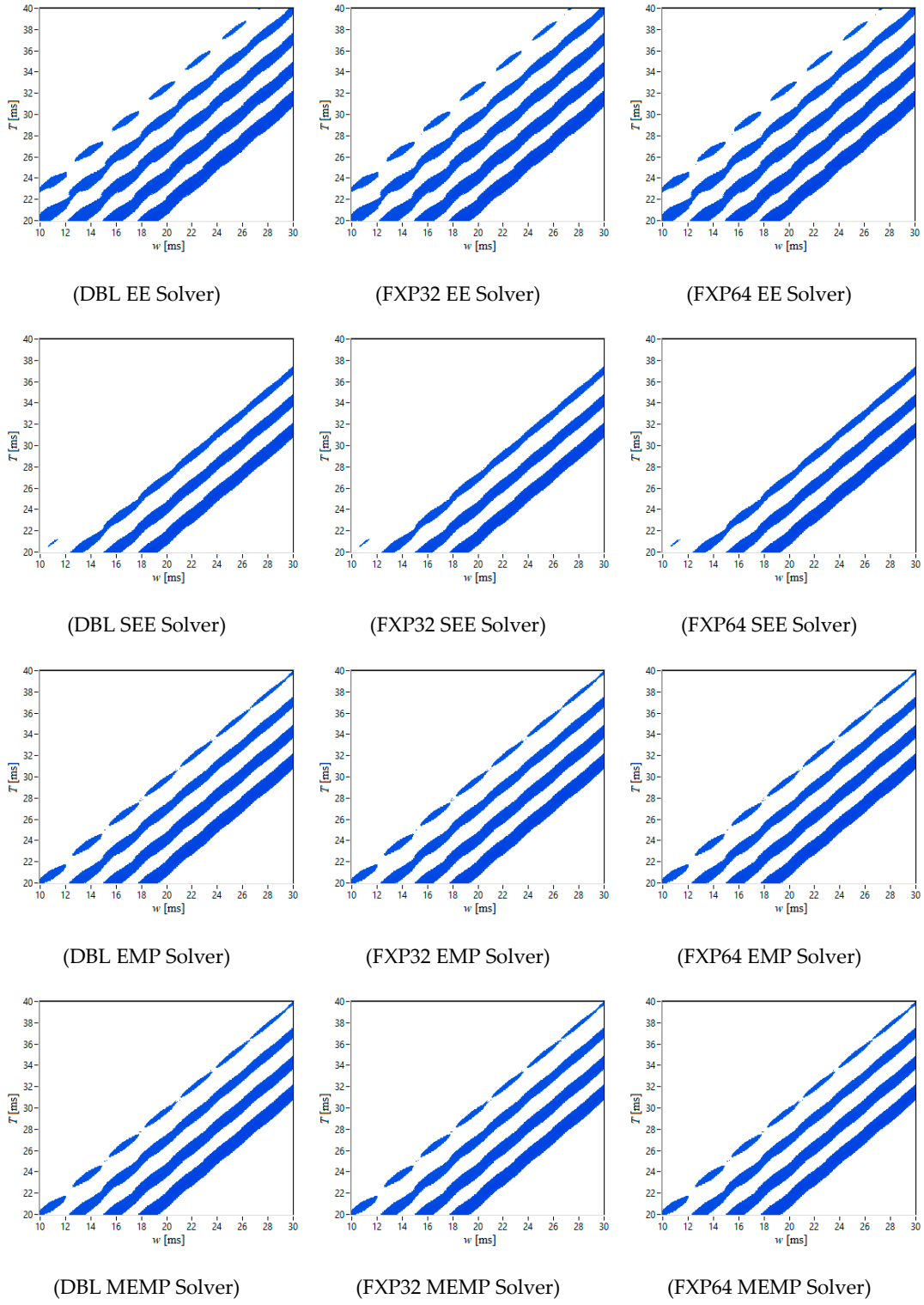


Figure 4. Areas of spiking mode for dynamical system (1) relative to period T and pulse width w when applying pulsed input current with amplitude I = 4.5 μA for 32-bit FXP, 64-bit FXP, and DBL solvers based on the EE, SEE, EMP, and MEMP methods.

The results showed that the excitability regions mainly form diagonal bands. The number of bands depends on the integration method used. The solvers based on the IE and SEE methods create the smallest excitability area: just three bands. The EE method leads to the highest number of bands in the excitability area, while the second-order methods EMP and MEMP give intermediate results. The transition to an integer data representation reduces the excitability area for each of the considered solvers. Table 1 provides a numerical assessment of this phenomenon. The number of lost excitability areas for the 32-bit and 64-bit FXP solvers relative to the DBL solvers for given simulation parameters was calculated as a percentage.

Table 1. The difference in area for the spiking mode for DBL and FXP solvers.

	EE Solver	**SEE Solver**	**EMP Solver**	**MEMP Solver**
FXP32 (10^{-3}%)	2.41	0.075	0.131	0.156
FXP64 (10^{-3}%)	2.35	0.018	0.012	0.006

As one can see from Table 1, the FXP solvers of system (1) based on the EE method most strongly lost sensitivity to the pulse signal. However, these losses were calculated in thousandths of a percent.

3.2. Chaotic Spike Generation

A series of computational experiments were carried out for the case when dynamical system (1) goes into a chaotic mode of generating action potentials in response to a signal $I = I_0 + A \sin(2\pi\omega t)$. The interspike interval histograms were used as an analysis tool. They represent the dependence between the number of intervals n among spikes and their duration t. In [22], it was concluded that the application of interspike interval histograms to solvers based on first-order methods is impractical because they do not maintain the chaotic mode of the system. The diagrams for the DBL, 32-bit FXP, and 64-bit FXP solvers built using the methods EMP and MEMP are represented in Figure 5. Simulations were conducted on the time interval $T = 105$ ms with integration step $h = 0.01$ for all solvers under consideration.

Figure 5 shows that the total number and duration of time intervals differ for all neuron models under consideration. However, it cannot be said that 32-bit FXP solvers are inferior to 64-bit FXP solvers in this series of experiments. It should be also noticed that differences in data type more strongly affect the solvers based on the EMP method: for the FXP64 EMP solver, intervals longer than 22 ms were not detected, although such intervals were observed on the prototype of the DBL solver.

3.3. Refractory Period

In this part of the research, we investigated the refractory period of the neuron model solvers. This period is the response time of a neuron model when two short-term current pulses are sent to its input. A common representation of input and output signals is provided in Figure 6.

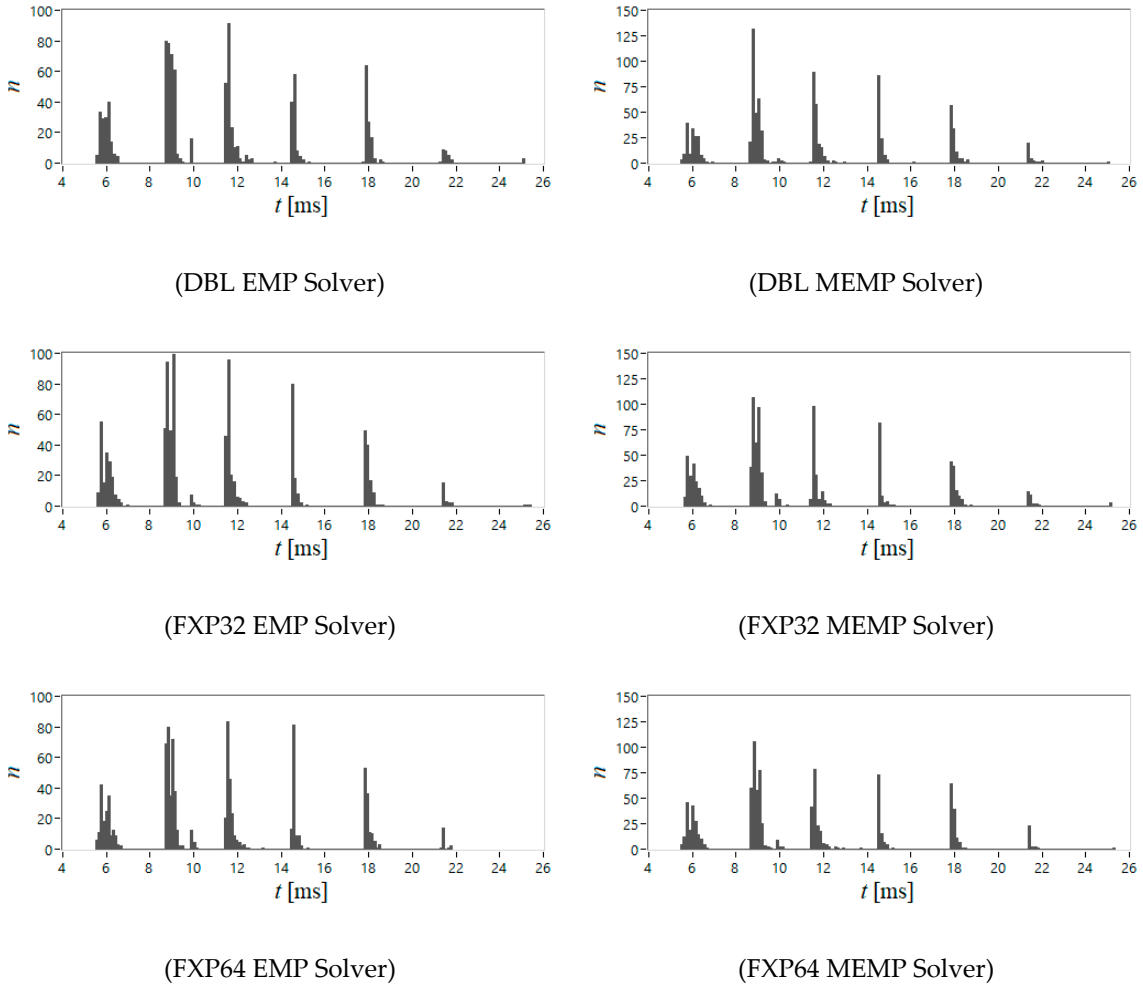


Figure 5. The interspike interval histograms (150 bins) for the DBL, 32-bit FXP, and 64-bit FXP solvers built using the methods EMP and MEMP.

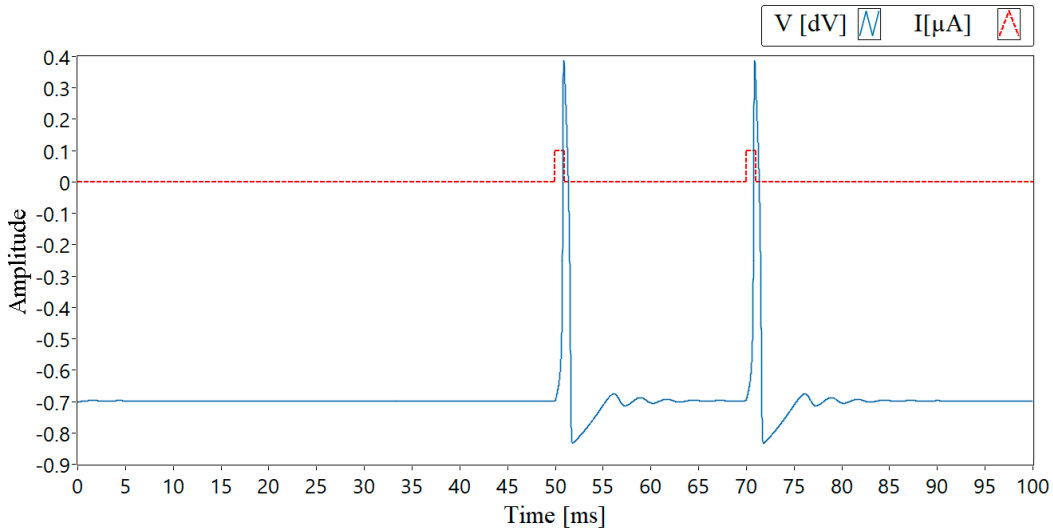


Figure 6. Input and output signals generated during refractory analysis of system (1).

The time interval between the first and the second pulses ranged from 50 to 70 ms. The pulse width was 1 ms. Simulations were conducted on the time interval $T = 100$ ms with integration step $h = 0.005$ ms. The results provided in Figure 6 were obtained for the input pulse amplitude $A = 20$ μA.

Figure 7a demonstrates that the occurrence times of spikes for system (1) almost coincide. The experiments showed that the data type of the EE, SEE, EMP, and MEMP solvers affects this characteristic significantly less than the numerical integration method that forms the basis of the solver. The maximum values of discrepancy in the time of occurrence of the second spike for the FXP solvers compared to the DBL solvers based on the same numerical integration method are given in Table 2. Figure 7b represents the differences between the DBL solvers on the basis of EE, SEE, EMP, and MEMP. The difference in time of occurrence of the second spike here reached tenths of a millisecond.

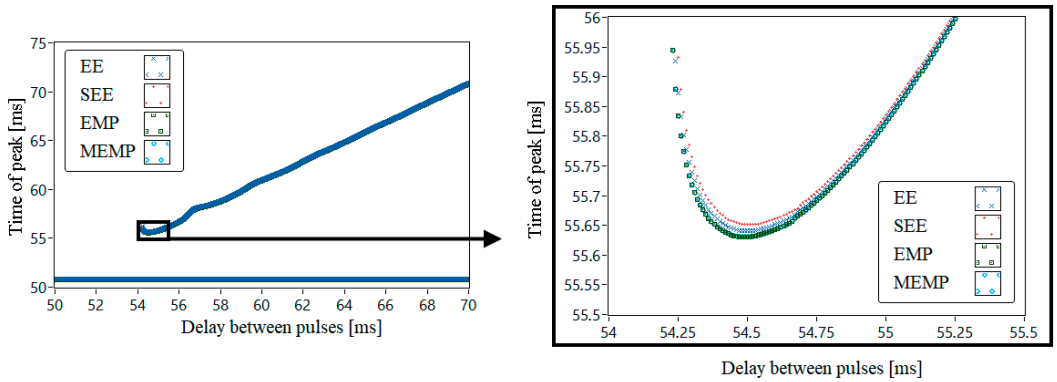


Figure 7. The refractory periods of the EE, SEE, EMP, and MEMP solvers of the system (1) while sending two pulse signals with amplitude $A = 20$ μA. The subgraph shows the observable difference between solutions obtained by various methods.

Table 2. The maximum values of discrepancy in the time of occurrence of the second spike for DBL and FXP solvers while sending two pulse signals with amplitude $A = 20$ μA.

	EE Solver	SEE Solver	EMP Solver	MEMP Solver
FXP32 (ms)	1.375×10^{-3}	2.771×10^{-3}	1.287×10^{-3}	1.841×10^{-3}
FXP64 (ms)	1.101×10^{-12}	8.458×10^{-13}	8.549×10^{-13}	1.001×10^{-13}

Table 2 demonstrates that all FXP solvers of system (1) give a similar discrepancy in occurrence time of the system (1) response regardless of the accuracy order of the integration method on which they are based.

In the case of input pulse amplitude $A = 12$ μA and the same simulation parameters, we obtained the results represented in Figure 8.

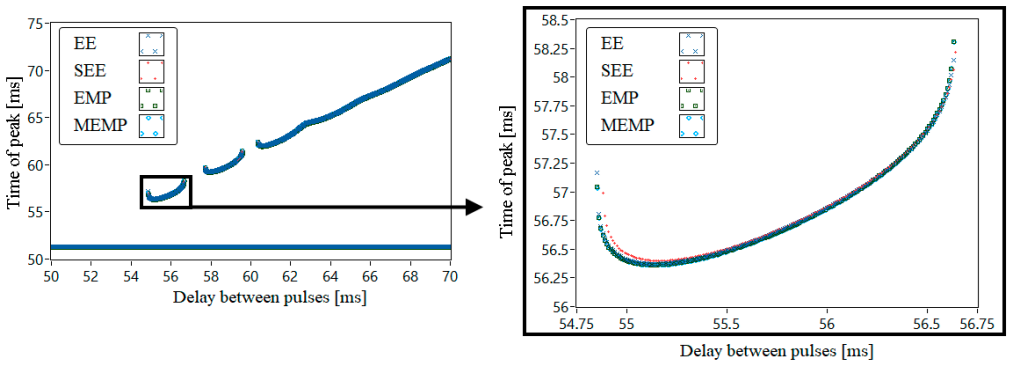


Figure 8. The refractory periods of the EE, SEE, EMP, and MEMP solvers of the system (1) while sending two pulse signals with amplitude $A = 12$ μA. The subplot shows the difference between the trajectories in a larger scale.

In the case of this amplitude, disappearance of the second spike at some intervals of time delay of the second input pulse was observed. The differences in behavior for the considered solvers of system (1) are retained. This can be seen in Figure 8b. However, compared to the previous experiment, the solvers on the basis of EE, SEE, and EMP match up to hundredths of a millisecond, and just the MEMP-based solver demonstrates a significantly different time of occurrence of the second spike on the system (1) output, calculated in tenths of a millisecond.

The FXP solvers give a greater discrepancy with DBL solvers in terms of response time with the input signal amplitude $A = 12$ μA. Similar to in the previous experiment, this difference is less significant than the difference between solvers based on various integration methods. This is represented in Table 3.

Table 3. The maximum values of discrepancy in the time of occurrence of the second spike for DBL and FXP solvers while sending two pulse signals with amplitude $A = 12$ μA.

	EE Solver	SEE Solver	EMP Solver	MEMP Solver
FXP32 (ms)	4.659×10^{-3}	2.775×10^{-3}	7.022×10^{-3}	10.916×10^{-3}
FXP64 (ms)	1.240×10^{-11}	3.711×10^{-12}	6.658×10^{-12}	5.866×10^{-12}

3.4. Hysteresis

In this part of the study, we explored the response of various neuron models to the half-period of the triangular wave. Such analysis allows for estimating the difference in hysteresis for system (1) solvers. Figure 9 illustrates an example of the input and output signals. The simulation was performed on the time interval $T = 400$ ms with integration step $h = 0.005$ ms. The period of the input signal was T = 800 ms, and the amplitude of the input signal was $A = 40$ μA.

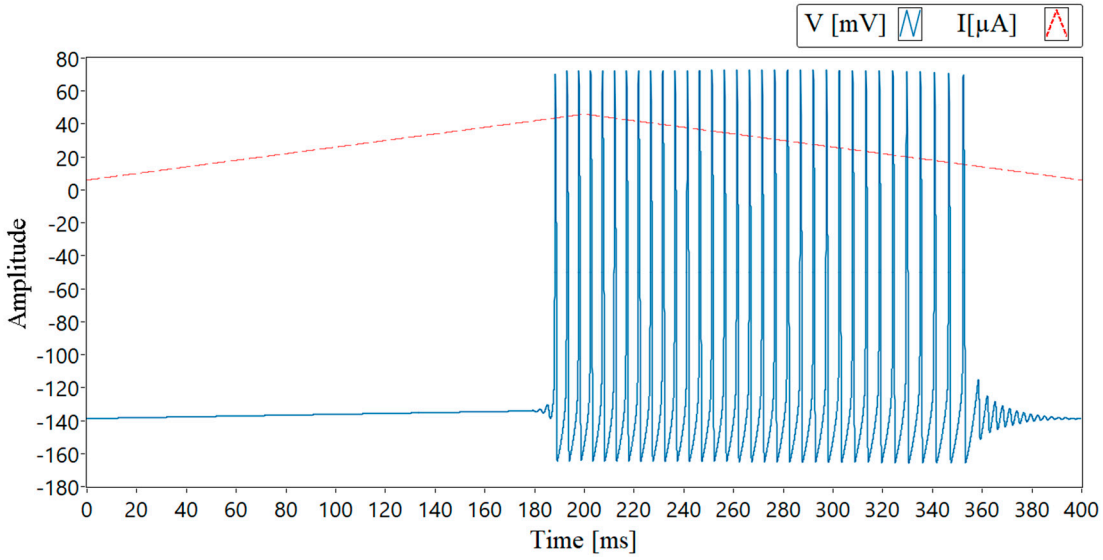
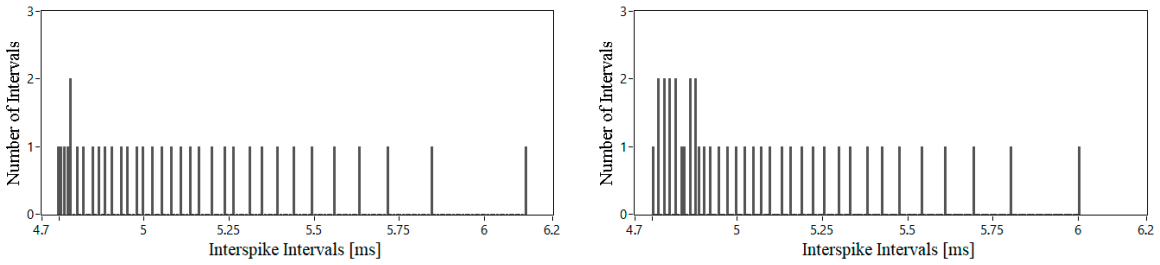


Figure 9. Hysteresis produced by system (1) in response to a triangular current ramp.

In order to evaluate the differences between the solvers in detail, the interspike interval histograms of system (1) were plotted. An example of these diagrams is shown in Figure 10. The time in milliseconds is plotted on the y axis and the number of interspike intervals on the x axis.



(**a**) (**b**)

Figure 10. Interspike intervals for dynamical system (1) when applying a triangular input signal of period T = 800 ms and amplitude I = 40 µA for DBL (**a**) and 32-bit FXP (**b**) Euler method solvers.

The processing of the results obtained during these two experiments is summarized in Tables 4 and 5.

Table 4. Spiking beginning time errors between FXP and DBL solvers, ms.

Data Type	EE Solver	SEE Solver	EMP Solver	MEMP Solver
FXP32	25	25	22	30
FXP64	0	7	7	8

Table 5. The largest interspike interval errors between FXP and DBL solvers, ms.

Data Type	EE Solver	SEE Solver	EMP Solver	MEMP Solver
FXP32	0.117	0.003	0.05	0.12
FXP64	0.001	0.097	0.053	0.03

According to the results of the hysteresis analysis, we can conclude that the MEMP method is the worst for fixed-point implementation. The EE method reproduces hysteresis properly even on 32 bits, but it poorly reproduces interspike intervals. The EMP method provides minor error compared to DBL for both bit lengths and can therefore be recommended for practical use.

3.5. Variation of the Integration Step Size

In discrete solvers of chaotic systems, the integration step size affects the system mode defining the rate of accumulation of rounding error. It leads to the emergence of a chaotic mode in points where it was not in the original system, and to the disappearance of this mode in the points where it exists in the prototype. To estimate the divergence of the DBL and FXP models using this criterion we built bifurcation diagrams that use the discretization step as a nonlinearity parameter. The charts of such dependency are called h-diagrams [23,24]. h-diagrams of the DBL and FXP solvers built on the basis of EMP are presented in Figure 11.

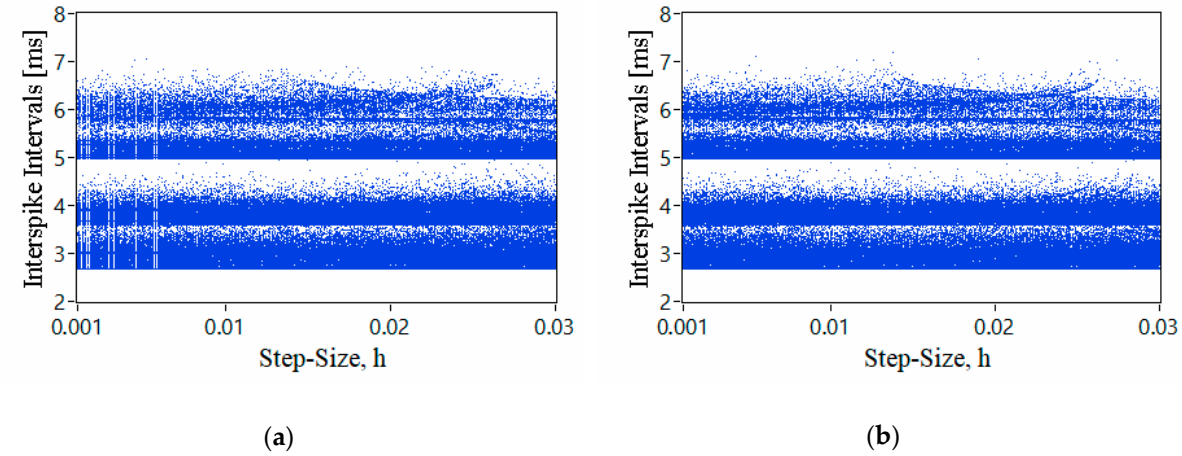


Figure 11. h-diagrams for system (1) built with EMP solvers: (**a**) 32-bit FXP, (**b**) DBL, and 64-bit FXP.

The DBL and FXP solvers demonstrate chaotic model behavior for parameters specified in system (1) and values of integration step size $h \in [0.001; 0.03]$. Differences between the DBL solver and 64-bit FXP solver are insignificant and almost invisible. However, different from them, the 32-bit solver demonstrates chaotic behavior at small integration step sizes. The models based on the EE, SEE, and MEMP methods give close results. A more detailed demonstration of the difference in dependency of chaotic mode on integration step size for the researched models can be seen in Figure 12.

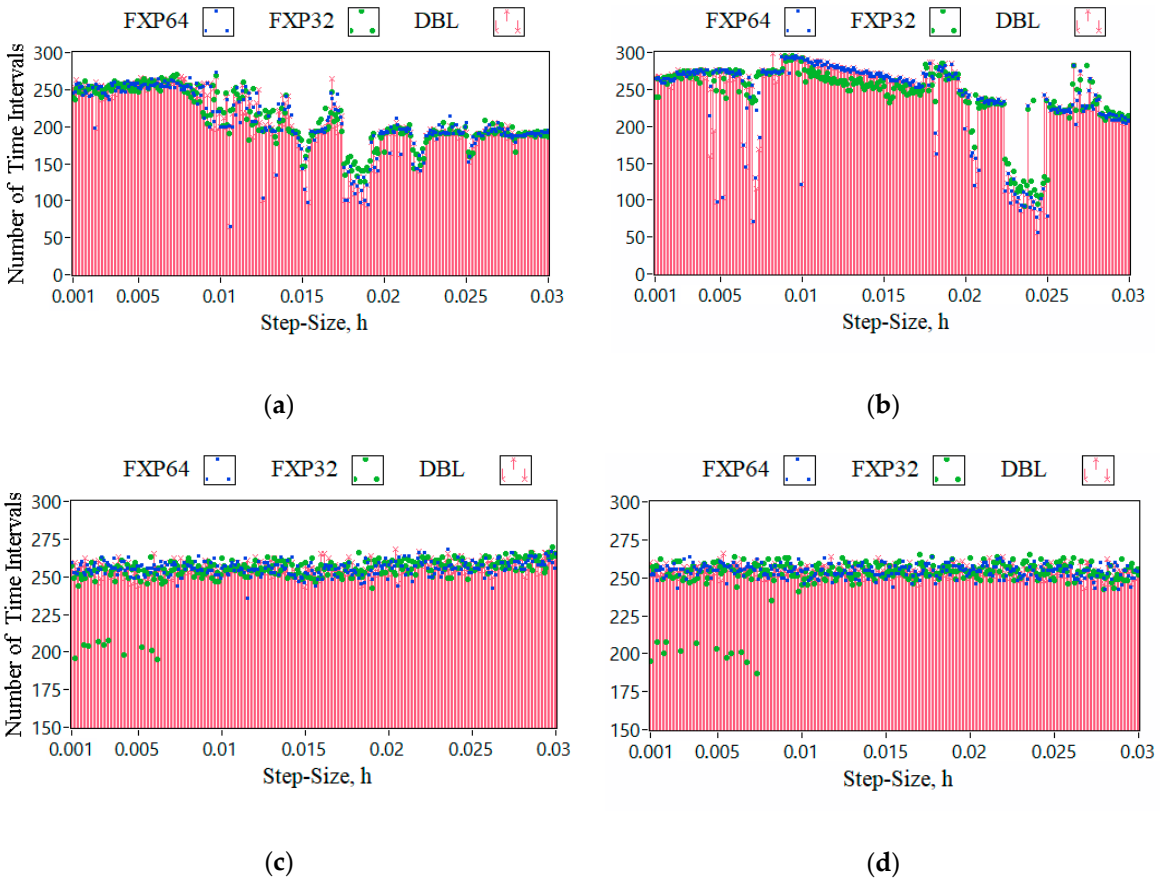


Figure 12. The numbers of interspike intervals for 32-bit FXP, 64-bit FXP, and DBL solvers obtained by the methods (**a**) EE, (**b**) SEE, (**c**) EMP, and (**d**) MEMP.

Figure 12 shows the dependency of the number of time intervals between spikes on integration step sizes. It can be noticed that methods of the second accuracy order (EMP, MEMP), compared to methods of the first order (EE, SEE), depend less on the integration step size and provide more stable solutions within the entire selected integration step size range. However, they are more sensitive to the data type used. On small integration step sizes $h \in [0.001; 0.007]$, a significant difference between the DBL solver and 32-bit FXP solver can be observed. This is not present in solvers built on the basis of first-order methods.

4. Discussion and Conclusions

Software models can be not applicable in many practical and scientific tasks because of their low performance and negative influence of the operating systems in real-time applications. In the last decade, scholars have intensively worked on high-speed realistic implementation of neuromorphic system in hardware. Meanwhile, these studies focused mainly on the interaction of neurons and the overall network architecture, while comparatively little attention has been paid to numerical solvers needed to synthesize finite-difference models reproducing single neuron dynamics. For example, in recent work by Pani et al. [2], a 32-bit FPGA implementation of the Izhikevich neuron using the explicit Euler method was shown. A good correspondence between the fixed-point model and the floating-point simulation was established; this may confuse followers, as for the various neuron models, different network architecture and different bit-length FPGA simulation results may not be that close to the software simulation results. In our work, we focused on the dynamics of individual neurons

and studied it in the case of various numerical methods and bit lengths to help developers correctly choose a numerical implementation.

Our present study is devoted to the possibility of a realistic spiking neuron model implementation (the simplified Hodgkin–Huxley model) using fixed-point arithmetic with bit lengths of 32 (FXP32) and 64 (FXP64) bits, which is relevant when implementing neuromorphic systems on FPGAs. For numerical simulation of the neuron dynamics, four methods were chosen: the Explicit Euler method (EE), the Semi-Explicit Euler method (SEE), the Explicit Midpoint method (EMP), and the Modified Explicit Midpoint method with a smoothing step (MEMP). These methods were tested on various problems often found in the scientific literature on natural and artificial neuron studies. Therefore, the obtained results can be considered valid for a wide class of problems associated with neuron simulation.

To summarize the results of all the experiments, Table 6 is given. In the left column for each bit length, the method providing the smallest simulation error compared to the double (DBL) data type is placed. Methods listed in the right-hand columns are the runners-up.

Table 6. Table summarizing the results of the study.

Test	FXP32		FXP64	
	Method Providing Lowest Error	**Second-Best Method(s)**	**Method Providing Lowest Error**	**Second-Best Method(s)**
Resonant spike generation	SEE	EMP	MEMP	EMP
Chaotic spike generation	EMP	EE	MEMP	SEE
Refractory period	EMP	EE	MEMP	EMP
Hysteresis (spiking phase error)	EMP	EE, SEE	EE	EMP, SEE
Hysteresis (largest interspike interval)	SEE	EMP	EE	MEMP

The following conclusions can be drawn from the results of the study:

- With a 32-bit word length, the EMP method having second-order algebraic accuracy turned out to be the best solver. Among the first-order methods, the SEE method is preferable. The MEMP method seems to be worse for a low-bit implementation due to the higher number of arithmetical operations.
- With a bit length of 64 bits, the MEMP method turned out to be the most accurate. The other three methods competed well with each other for accuracy in various tests. Taking into account its second-order algebraic accuracy, the EMP method is preferable.
- Figure 12 shows that when using the EMP and MEMP methods with different integration steps, the dynamics of the neuron remains relatively unchanged, while the EE and SEE methods strongly affect the dynamics. The difference in the number of time intervals at various steps can reach 2 times, so the use of the Euler method and its modifications in neural network emulators cannot be recommended.
- Summarizing the two last points, in the general case, one should choose the EMP method to reproduce the dynamics of neurons in fixed-point arithmetic.

In future studies, we will check intermediate bit lengths (such as 40 bits) to establish a smoother dependence of the method preference according to data type. The implementation of a neuron and neuromorphic system model on FPGA is also of interest.

Author Contributions: Data curation, V.O., T.K. and A.T.; formal analysis, A.T. and E.D.; investigation, V.A. and V.O.; methodology, V.A. and D.B.; project administration, D.B.; resources, V.O. and E.D.; software, V.A. and D.B.; supervision, A.T.; validation, V.O., T.K., E.D. and D.B.; visualization, T.K. and A.T.; writing—original draft, V.A., V.O. and D.B.; writing—review and editing, E.D., A.T. and D.B. All authors have read and agree to the published version of the manuscript.

Funding: The reported study was supported by RFBR, research project No. 19-07-00496.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ko, B.C. A Brief Review of Facial Emotion Recognition Based on Visual Information. *Sensors* **2018**, *18*, 401. [CrossRef] [PubMed]
2. Zhao, J.; Mao, X.; Chen, L. Speech Emotion Recognition Using Deep 1D & 2D CNN LSTM Networks. *Biomed. Signal Process. Control.* **2019**, *47*, 312–323.
3. Ma, X.; Dai, Z.; He, Z.; Ma, J.; Wang, Y.; Wang, Y. Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction. *Sensors* **2017**, *17*, 818. [CrossRef] [PubMed]
4. Chen, M.; Hao, Y.; Hwang, K.; Wang, L.; Wang, L. Disease Prediction by Machine Learning over Big Data from Healthcare Communities. *IEEE Access* **2017**, *5*, 8869–8879. [CrossRef]
5. Berger, T.W.; Hampson, R.E.; Song, D.; Goonawardena, A.; Marmarelis, V.Z.; Deadwyler, S.A. A cortical neural prosthesis for restoring and enhancing memory. *J. Neural Eng.* **2011**, *8*, 046017. [CrossRef] [PubMed]
6. Pani, D.; Meloni, P.; Tuveri, G.; Palumbo, F.; Massobrio, P.; Raffo, L. An FPGA Platform for Real-Time Simulation of Spiking Neuronal Networks. *Front. Neurosci.* **2017**, *11*, 90. [CrossRef] [PubMed]
7. Mead, C. *Analog VLSI and Neural Systems*; Addison-Wesley Longman Publishing Co Inc.: Boston, MA, USA, 1989.
8. Mead, C. Neuromorphic electronic systems. *Proc IEEE* **1990**, *78*, 1629–1636. [CrossRef]
9. Douglas, R.; Mahowald, M.; Mead, C. Neuromorphic analogue VLSI. *Ann Rev Neurosci* **1995**, *18*, 255–281. [CrossRef] [PubMed]
10. Prezioso, M.; Merrikh-Bayat, F.; Hoskins, B.D.; Adam, G.C.; Likharev, K.K.; Strukov, D.B. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **2015**, *521*, 61. [CrossRef] [PubMed]
11. Jang, J.T.; Ahn, G.; Choi, S.J.; Kim, D.M.; Kim, D.H. Control of the Boundary between the Gradual and Abrupt Modulation of Resistance in the Schottky Barrier Tunneling-Modulated Amorphous Indium-Gallium-Zinc-Oxide Memristors for Neuromorphic Computing. *Electronics* **2019**, *8*, 1087. [CrossRef]
12. Kasap, B.; van Opstal, A.J. Dynamic parallelism for synaptic updating in GPU-accelerated spiking neural network simulations. *Neurocomputing* **2018**, *302*, 55–65. [CrossRef] [PubMed]
13. Zhang, Z.; Ma, C.; Zhu, R. A FPGA-Based, Granularity-Variable Neuromorphic Processor and Its Application in a MIMO Real-Time Control System. *Sensors* **2017**, *17*, 1941. [CrossRef] [PubMed]
14. Jia, T.; Guo, T.; Wang, X.; Zhao, D.; Wang, C.; Zhang, Z.; Lei, S.; Liu, W.; Liu, H.; Li, X. Mixed Natural Gas Online Recognition Device Based on a Neural Network Algorithm Implemented by an FPGA. *Sensors* **2019**, *19*, 2090. [CrossRef] [PubMed]
15. Valadez, G.S.; Sossa, H.; Santiago-Montero, R. How the Accuracy and Computational Cost of Spiking Neuron Simulation are Affected by the Time Span and Firing Rate. *Comput. Sist.* **2017**, *21*, 841–861. [CrossRef]
16. Hopkins, M.; Mikaitis, M.; Lester, D.R.; Furber, S. Stochastic rounding and reduced-precision fixed-point arithmetic for solving neural ordinary differential equations. *Philos. Trans. R. Soc. Math. Physical Eng. Sci.* **2020**, *378*, 20190052. [CrossRef] [PubMed]
17. Wilson, H.R. *Spikes, Decisions, and Actions: The Dynamical Foundations of Neurosciences*; Oxford University Press: Oxford, UK, 1999.
18. Hodgkin, A.L.; Huxley, A.F. A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve. *J. Physiol.* **1952**, *117*, 500–544. [CrossRef]
19. Rinzel, J. Excitation Dynamics: Insights from Simplified Membrane Models. *Fed. Proc* **1985**, *44*, 2944–2946. [PubMed]
20. Andreev, V.S.; Goryainov, S.V.; Krasilnikov, A.V.; Sarma, K.K. Scaling Techniques for Fixed-Point Chaos Generators. In Proceedings of the IEEE Russia Section Young Researchers in Electrical and Electronic Engineering Conference (2017 ElConRus), Saint Petersburg, Russia, 1–3 February 2017.
21. Izhikevich, E.M. *Dynamical Systems in Neuroscience*; MIT press: Cambridge, MA, USA, 2007.
22. Ostrovskii, V.Y.; Karimov, T.I.; Solomevich, E.P.; Kolev, G.Y.; Butusov, D.N. Numerical Effects in Computer Simulation of Simplified Hodgkin–Huxley Model. In Proceedings of the 2nd International Conference on Mathematics and Statistics (ICoMS 2019), Prague, Czech Republic, 8–10 July 2019.
23. Butusov, D.N.; Ostrovskii, V.Y.; Karimov, A.I.; Andreev, V.S. Semi-Explicit Composition Methods in Memcapacitor Circuit Simulation. *Int. J. Embed. Real-Time Commun. Syst.* **2019**, *10*, 37–52. [CrossRef]

24. Kaplun, D.I.; Tutueva, A.V.; Butusov, D.N.; Karimov, A.I.; Toming, J. Memristive Circuit Simulation Using the Semi-Implicit Multistep Method. In Proceedings of the 42nd International Conference on Telecommunications and Signal Processing (TSP), Budapest, Hungary, 1–3 July 2019; pp. 19–20.