Model-based online implementation of spike detection algorithms for neuroengineering applications

M. Di Florio, *Student Member, IEEE,* V. Iyer, A. Rajhans, *Senior Member, IEEE,* S. Buccelli°, *Member, IEEE,* M. Chiappalone°, *Member, IEEE*

***Abstract*— Traditional methods for the development of a neuroprosthesis to perform closed-loop stimulation can be complex and the necessary technical knowledge and experience often present a high barrier for adoption. This paper takes a novel Model-Based Design approach to simplifying such closed-loop system development, and thereby lowering the adoption barrier. This work implements a computational model of different spike detection algorithms in Simulink® and compares their performances by taking advantage of synthetic neural signals to evaluate suitability for the intended embedded implementation.**

***Clinical Relevance*— Closed-loop systems have been demonstrated to be suitable for brain repair strategies. Coupling two different brain areas by means of a neuroprosthesis can potentially lead to restoration of communication by inducing activity-dependent plasticity.**

I. Introduction

Existing and upcoming in vivo intracranial recording systems provide high temporal and spatial resolution from thousands of sites during behavioral tasks in animals [1]. Other technologies for in vivo recording, including active CMOS probes, are able to isolate a single neuron from a large brain area [2]. This current capability to acquire signals from such a large number of electrodes deserves a corresponding proficiency in online signal processing for a wide spectrum of applications, e.g. BCI, adaptive neuromodulation, neuromorphic systems, neuroprosthetics, etc. [3]–[5].

The design and development of a real-time system, however, can be daunting in terms of complexity for a non-expert. Indeed, learning how to implement a signal processing algorithm on hardware often prevents many neuroscientists from realizing their novel and innovative systems. In this context, providing researchers with solutions that can reduce the entry barrier is of fundamental significance.

Spike detection is the first, basic, and essential step of neuronal data analysis for neuroengineering applications requiring real-time data processing [6], [7]. In this work, our goal is to design and test different spike detection algorithms in Simulink® [12] to evaluate the best candidate for the intended embedded (μC or Field Programmable Gate Array - FPGA) implementation. This Model-Based Design approach enables us to (i) simulate, analyze, and forecast pros and cons

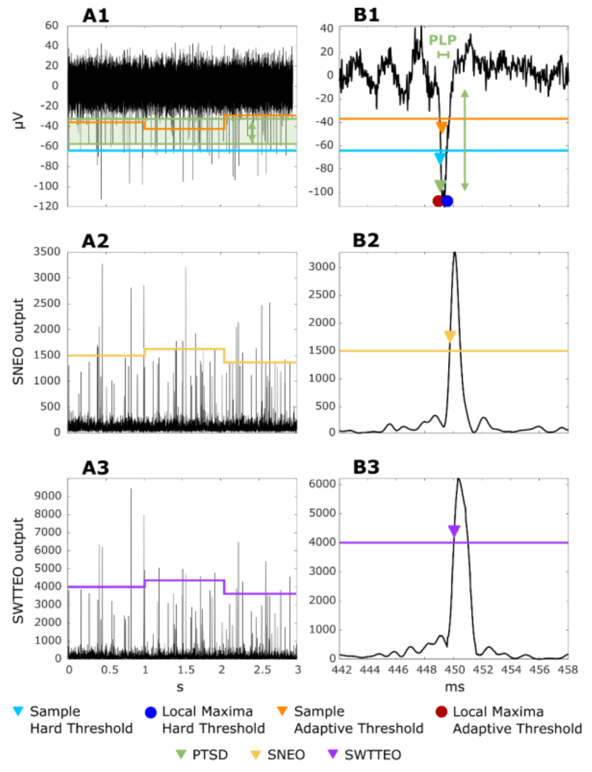

Figure 1. Main elements and concepts in spike detection algorithm s. Panels A1, A2 and A3 show, from the top to the bottom, portions of a filtered signal, a SNEO output and a SWTTEO output. Different colors highlight the different thresholding method used for identifying a spike: (i) light blue for the hard threshold, (ii) orange for the adaptive threshold from the Sample Thresholding category, (iii) green for the PTSD, (iv) yellow for the adaptive threshold of the SNEO, and (v) orange for the adaptive threshold of the SWTTEO. The figures B1, B2 and B3 show how and where a spike is identified. Local Maxima Hard Threshold and, Local Maxima Adaptive Threshold identify the first peak after the threshold crossing (blue and red circles). For the PTSD a spike is detected when the two conditions (differential threshold and peak lifetime period) are met (green triangle). For the remaining, the first sample above threshold is identified as a spike.

of the code execution on target hardware, and (ii) conduct preliminary checks about how the algorithms would behave in real-time. The results of our exemplar computational implementation thus represent a fundamental step for lowering

° Equal Senior contribution

*This work is supported by MathWorks. The opinions and views expressed in this publication are those of the authors, and not necessarily those of MathWorks.

M. Di Florio is with the Department of Informatics, Bioengineering, Robotics, System Engineering (DIBRIS), University of Genova, Via all'Opera Pia 13, 16145, Genova, Italy (e-mail: s4529318@studenti.unige.it).

V. Iyer is with MathWorks, 3 Apple Hill Drive, Natick, MA 01760, USA. (e-mail: vijayi@mathworks.com).

A. Rajhans is with MathWorks, 3 Apple Hill Drive, Natick, MA 01760, USA. (e-mail: arajhans@mathworks.com).

S. Buccelli is with the Rehab Technologies IIT-INAIL Lab, Istituto Italiano di Tecnologia, Via Morego 30, 12 16163 Genova, Italy (corresponding author, e-mail: stefano.buccelli@iit.it).

M. Chiappalone is with the Department of Informatics, Bioengineering, Robotics, System Engineering (DIBRIS), University of Genova, Via all'Opera Pia 13, 16145, Genova, Italy (corresponding author, e-mail: michela.chiappalone@unige.it).

the adoption barrier by neuroscientists and neuroengineers for the development of embedded applications.

The use of high-level programming tools that generate embedded code for µC and FPGAs can simplify the design workflow and speed up the deployment of closed-loop applications in the neuroscientific and neuroengineering fields. The Simulink models and MATLAB® source code, along with Python code used to generate the dataset, are available on GitHub (https://github.com/MattiaDif/model-based-spike-detection).

II. Implemented Spike Detection Algorithms

According to the literature [8], spike detection algorithms can be divided into three main classes based on the methodological approach: (i) Sample Thresholding, (ii) Energy operator, and (iii) Template Matching. In this study, algorithms from methodological approaches (i) and (ii) have been implemented and compared.

A. Sample Thresholding

These algorithms assume that the amplitude of a spike is greater than the amplitude of the noise, therefore, a spike can be detected by thresholding the signal. As part of this class, six different algorithms have been designed, briefly detailed here below. A graphical representation of the functioning of these algorithms is reported in Figure 1 – A1:

- *Sample Hard Threshold*. A sample above threshold is identified as a potential spike. If this sample appears at least one refractory period after the previous spike, a new spike is detected [8].
- *Local Maxima Hard Threshold*. A peak in the signal has to overcome the threshold in order to detect a potential spike [8].
- *Sample Adaptive Threshold*. A metric such as RMS, STD, Median Absolute Value (MAV), etc. is computed and a threshold is set multiplying the metric value by a constant. If a sample is above the adaptive threshold, it is identified as a putative spike [8].
- *Local Maxima Adaptive Threshold*. A peak in the signal has to overcome the adaptive threshold in order to detect a potential spike [8].
- *Precision Timing Spike Detection (PTSD)*. The PTSD logic analyzes consecutive portions of the signal and includes a differential threshold (DT) and a peak lifetime period (PLP) evaluation of the signal to detect a spike [6].

B. Energy Operator

Whenever an abrupt amplitude transient occurs, a high-frequency pattern different from the noise is introduced in the signal. Nonlinear energy operator (NEO), also known as Teager energy operator (TEO), is computed, estimating the instantaneous product of amplitude and frequency leading to an enhancement in the high-frequency content of the signal (i.e. a spike). We implement two algorithms from this category (see Figure 1 – A2, A3):

- *Smoothed Nonlinear Energy Operator (SNEO)*. Smoothed NEO (SNEO) is computed by a convolution between a window and the NEO. The threshold is automatically set by the algorithm according to the MAV of the SNEO [9].
- *Stationary Wavelet Transform Teager Energy Operator (SWTTEO)*. SWTTEO consists in a low-pass filter of the TEO (or NEO) using n level of approximation coefficients of the Discrete Wavelet Transform (DWT). However, the DWT is substituted by the Stationary Wavelet Transform (SWT) for avoiding the down sampling of the approximation coefficients. The threshold is automatically set by the algorithm according to the MAV of the SWTTEO output [8].

III. Synthetic Dataset

Validation of spike detection requires a reliable ground-truth. Recordings that combine acquisitions of extracellular and juxtacellular/patch clamp methods cannot provide a consistent ground-truth for our models, because the activity of only a few neurons can be simultaneously measured. Therefore, synthetic data was used allowing for the generation of sufficient statistics for comparative assessment.

A. Approach

To generate synthetic recordings, we employed MEArec. MEArec is an open-source Python-based simulator that provides fast, intuitive, biophysically accurate extracellular recordings for testing and optimization of spike sorting algorithms [10]. This tool allows exploration of different aspects that characterize in vivo recordings (e.g. bursting, drifting, far neuron noise, overlap, etc.), thus reproducing reliable MEA signals comparable to the reality. Moreover, MEArec includes the possibility to encompass different neuron models and types (e.g. from Allen Brain Institute database and from the Blue Brain Project database) and to generate recordings based on various probe models [10]. In this work, the dataset (see Figure 2) was generated using the following main features: (i) tetrode probe (ii) 2 neurons: 2 excitatory units, (ii) 1 minutes of recording, (iv) $F_s = 30\ kHz$, (v) drift and (vi) additive colored noise level STD: $10\mu V, 20\mu V, 30\mu V$.

IV. Design Considerations in Real-Time Simulation of Spike Detection Algorithms

Each spike detection algorithm has been designed for both single- and multi-channel detection. In this Section, we discuss the most relevant challenges regarding the development of spike detection algorithms in Simulink with their target embedded implementations in mind.

A. Filter

Generally, concerning neural recordings, a bandpass filter is required [7]. To this end, we implemented both a lowpass IIR filter (Butterworth, cutoff frequency of 3 kHz, 1st order) and a highpass IIR filter (Butterworth, cutoff frequency of 3kHz, 3rd order) by means of an automated tool (the Filter Designer App in the Signal Processing Toolbox [13])). For an online implementation of a filter, a given number of previous samples dependent on the filter order and architecture needs to be stored for the processing of the incoming samples. In Simulink, this task is automatically accomplished by the generated code.

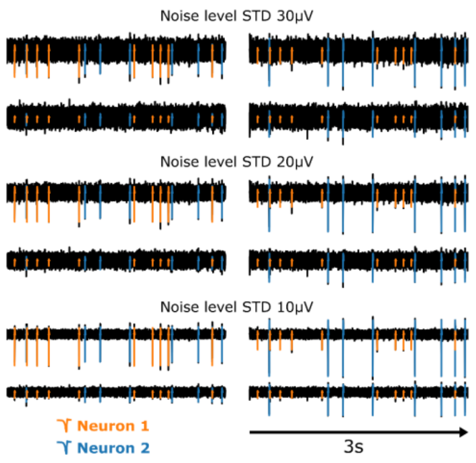

Figure 2. Synthetic data generated with MEArec for all the noise level STD.

B. Refractory Period

Computation of the refractory period online, requires a counter. Every time a spike is detected as per the logic of the algorithm, the refractory period starts counting. If a spike is detected at least one refractory period after the previous one, a new spike is detected, and the counter is reset to zero.

C. Local Maxima Check

Performing an online detection of a peak in the signal necessitates storing a window of the samples at time t_n, t_{n-1} and t_{n-2}. The peak detection occurs when the sample at t_{n-1} is above (or below for the local minimum) the ones recorded before (t_{n-2}) and after (t_n).

D. Adaptive Threshold

The adaptive threshold consists of an automatic update of the threshold according to the level of noise based on a specific metric such as STD, RMS, or MAV. As per [11], the MAV has been implemented due to its nature to be less conditioned by the spiking activity. To compute an adaptive threshold, N previous consecutive samples must be stored to calculate the MAV for estimating the threshold for the next segment of signal. This could be done using a buffer of length N (and different buffer overlaps depending on the frequency at which it is considered acceptable to modify the threshold).

E. PTSD implementation

The PTSD algorithm requires two specific parameters different from the other algorithms: (i) the DT and (ii) the PLP [6]. A buffer keeps in memory the last X samples of the signals. X needs to be comparable with the spike duration and it strongly depends on the sampling frequency. The first and the last sample of the buffer are read, and the two following conditions are checked. If the absolute value of the difference between the first and the last sample of the buffer is greater than or equal to DT, and the absolute value of the difference between the index of the first sample and the index of last sample is less than the PLP, a new spike is detected.

F. NEO

The discrete-time NEO is defined as:

$$\Psi[x(n)] = x^2(n) - x(n+1)x(n-1) \text{ [9].}$$

As indicated in the formula, a real-time implementation of the operator for a real-time application, would necessitate two previous samples to be stored. Indeed, to compute the current SNEO sample at time t_n (corresponding to $x(n+1)$) the square of the signal sample at time t_{n-1} and the product between the current signal sample t_n and the sample at time t_{n-2} are required. Note that the square operation for an embedded implementation can be demanding and could require a nontrivial amount of hardware resources.

G. Stationary Wavelet Transform

The SWT consists of applying two orthogonal filters (low-pass and high-pass) by a convolution between the signal and the wavelet coefficients. For online implementation, a signal segment needs to first be stored in a buffer, and then the convolution between the stored segment and the coefficients can be performed. Afterwards, a new signal segment is saved in a buffer and the convolution is performed again.

V. Results

This section reports the preliminary results of the performance analysis of the single channel algorithms designed in Simulink. The results account for all the four channels of the simulated tetrode, considering all the three noise levels (cf. Figure 2).

A. Parameter Choice

To compute the performances, we varied one parameter (typically the threshold) of each algorithm, according to the ranges reported in Table 1. The refractory period has been set to 1ms. The hard threshold and the threshold gain have been selected according to the noise levels. Moreover, the length of the smoothing windows and the wavelet for the SWTTEO algorithm have been chosen consistently with previous evidence [8].

Table 1: Spike detection parameters.

Algorithms	**Threshold**	**MAV**	**Others**
Sample Hard Threshold	-80µV to -40µV Step 1µV	-	-
Local Maxima Hard Threshold	-80µV to -40µV Step 1µV	-	-
Sample Adaptive Threshold	Threshold gain from 2 to 7 Step 0.1	Buffer 1s	-
Local Maxima Adaptive Threshold	Threshold gain from 2 to 7 Step 0.1	Buffer 1s	-
PTSD	DT from 20µV to 100µV Step 1.25µV	-	PLP of 0.7ms
SNEO	Threshold gain from 2 to 7 Step 0.1	Buffer 1s	Hamming windows of 1ms
SWTTEO	Threshold gain from 2 to 7 Step 0.1	Buffer 1s	Hamming windows of 1ms, sym5 wavelet, 2 level of decomposition

B. Spike Detection Performance

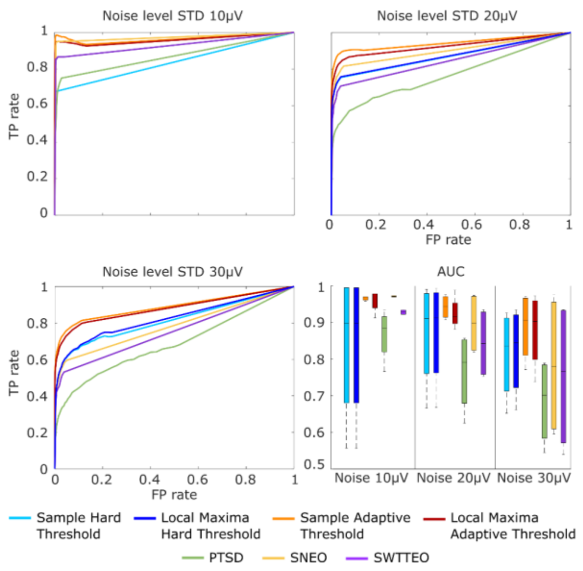

Figure 3: Performance analysis of the algorithms according to the different level of noise of the dataset. The ROC curves and the AUC are obtained by the average of FP_{rate} and TP_{rate} between the four channels of the tetrode. The 75th and 25th percentile of the data are represented by the top and the bottom of the boxes respectively. The whiskers range from the end of the interquartile to the maximum and minimum observation of the sample. The middle line represents the median.

Figure 3 reports the ROC curves and the boxplots of the area under the curve (AUC) according to the three different noise levels. For low noise level (STD 10 μV), the Sample Hard Threshold, Local Maxima Hard Threshold and PTSD algorithms suffers from their little capability to adapt the thresholding according to the signal features of each channel. Indeed, each channel of the tetrode is characterized by different spike amplitudes depending on the distance between the electrode and the neuron, therefore, the adaptive threshold-based algorithms perform better. For a noise level of 20 μV the algorithms behave similarly, with an overall decrease in the performance and a general increase in the variability of the AUC. For the highest noise level, as expected, the overall performances worsen, and the overall variability of the AUC is increased. The PTSD shows the worse results. Moreover, the SNEO and the SWTTEO performances abruptly decreases. Particularly, the loss of performance of the SWTTEO could depend on the type of the chosen wavelet and the number of the level of decomposition. The energy-based algorithms are more affected in case of noise as the high-frequency content of the spike blends with the high-frequency of the noise and the spikes exhibit lower amplitudes. Overall, our preliminary analysis indicates that the hard - and adaptive threshold-based algorithms show, on average, a more stable behavior than the others and the detection of the peak instead of the first sample above/below threshold does not improve the outputs.

VI. Discussion and Conclusion

This work represents the first, important step towards the adoption of a Model-Based Design approach for neuroengineering applications. The use of Simulink allowed us to simulate the functioning of a set of online spike detection algorithms before embedded/hardware implementation, thus providing a lightweight mechanism for its use even by non-experts. Indeed, Model-Based Design allows the developer to focus on the algorithm with a minimal effort on the final implementation. Depending on the final platform and the required performances, C/C++ or VHDL/Verilog code can be automatically generated. Our preliminary results point out the capability of the majority of the algorithms to provide a suitable spike detection for the different conditions. The energy-based operator models are promising and deserve a deeper analysis. A further investigation of the algorithm behavior for higher level of noise is required. Furthermore, neuroscientists and neuroengineers can take advantage of a results suite to analyze in order to determine the best algorithm for a specific real-time application.

It is worth to underlie that this project is under active development and will be available for the community, therefore new features will be introduced over time to expand and improve the current capability, such as: (i) new algorithm design, (ii) algorithm optimization, and (iii) GUI development. We believe this approach will speed up the implementation of complex closed-loop architectures with demanding timing requirements for an increasing number of recording channels.

References

[1] N. A. Steinmetz *et al.*, “Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings,” *Science (80-.).*, vol. 372, no. 6539, pp. 139–148, 2021.

[2] G. N. Angotzi *et al.*, “SiNAPS: An implantable active pixel sensor CMOS-probe for simultaneous large-scale neural recordings,” *Biosens. Bioelectron.*, vol. 126, no. August 2018, pp. 355–364, 2019.

[3] S. Buccelli *et al.*, “A Neuromorphic Prosthesis to Restore Communication in Neuronal Networks,” *iScience*, vol. 19, pp. 402–414, 2019.

[4] R. George *et al.*, “Plasticity and Adaptation in Neuromorphic Biohybrid Systems,” *iScience*, vol. 23, no. 10, pp. 1–26, 2020.

[5] D. J. Guggenmos *et al.*, “Restoration of function after brain damage using a neural prosthesis,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 110, no. 52, pp. 21177–21182, 2013.

[6] A. Maccione, M. Gandolfo, P. Massobrio, A. Novellino, S. Martinoia, and M. Chiappalone, “A novel algorithm for precise identification of spikes in extracellularly recorded neuronal signals,” *J. Neurosci. Methods*, vol. 177, no. 1, pp. 241–249, 2009.

[7] M. Murphy *et al.*, “Improving an open-source commercial system to reliably perform activity-dependent stimulation,” *J. Neural Eng.*, vol. 16, no. 6, p. ab3319, 2019.

[8] F. Lieb, H. G. Stark, and C. Thielemann, “A stationary wavelet transform and a time-frequency based spike detection algorithm for extracellular recorded data,” *J. Neural Eng.*, vol. 14, no. 3, p. aa654b, 2017.

[9] S. Mukhopadhyay and G. C. Ray, “A new interpretation of nonlinear energy operator and its efficacy in spike detection,” *IEEE Transactions on Biomedical Engineering*, vol. 45, no. 2. pp. 180–187, 1998.

[10] A. P. Buccino and G. T. Einevoll, “MEArec: A Fast and Customizable Testbench Simulator for Ground-truth Extracellular Spiking Activity,” *Neuroinformatics*, vol. 19, no. 1, pp. 185–204, 2021.

[11] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, “Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering,” *Neural Comput.*, vol. 16, no. 8, pp. 1661–1687, 2004.

[12] MathWorks®, "Simulink® R2021b," September 2019. [Online]. Available: https://www.mathworks.com/products/simulink.html

[13] MathWorks®, "Signal Processing Toolbox® R2021b," September 2019. [Online]. Available: https://www.mathworks.com/products/signal.html