GPU-based Fast Parameter Optimization for Phenomenological Spiking Neural Models

Zafeirios Fountas and Murray Shanahan

***Abstract*—A significant obstacle in using phenomenological models of spiking neurons for large-scale simulations is the approximation of the optimal parameters for a type of neuron, given the available experimental data. Here we show a method for optimizing the parameters of such models, based on a combination of different frequency-current and voltage-current relations of a neuron as well as known physiological properties. We also present a python toolbox which uses NeMo spiking neural network simulator and provides a fast GPU-based implementation of our method. As a benchmark, our toolbox was used to fit Izhikevich equations to neurological data obtained from a cat's thalamic relay cell. Our resulting model was able to predict the firing patterns of known membrane potential traces of this neuron, although they were not explicitly defined during training. A further comparison between this neuron model and a previous approach, when both models are used in the simulation of a generic thalamic nucleus, revealed that the distribution of neuronal avalanches is significantly different and conforms better to power law-like distributions, thus increasing the likelihood of a critical regime and the biological plausibility of the simulation.**

I. Introduction

Phenomenological models of spiking neurons can be very useful in computational neuroscience, since they provide a powerful way of capturing the dynamical behaviour of real cells and replicating their exact membrane potential traces with a reduced computational cost. For this reason, the popularity of this approach is continuously growing, especially in large-scale networks that include from real-time simulations of abstract models [1], [2], [3] to slower simulations of complete brain structures [4], [5], [6]. However, since a large proportion of the parameters of these models do not have any biological meaning, their employment usually requires prior fine-tuning, to make them operate accurately in different regimes.

A number of different optimization methods have been proposed that may involve matching exact membrane potential trajectories in different stimulus conditions [7], spike trains [8], approximation of the phase plane of a neuron [9], frequency-current (F-I) or voltage-current (V-I) relations [10] or a combination of the above [11] (for a review of these methods, see [12]). In addition, a variety of recently developed software tools, that aim to tune spiking neurons, provide implementations for many of the aforementioned methods at a certain computational cost. Examples of such software systems include the software tools Neurofitter [12] and Optimizer [11] as well as tools that work at the network level [13].

Despite the wide range of available approaches, the literature still lacks a general methodology and, as a result, researchers often prefer to follow the safest but remarkably time-consuming solution of hand-tuning [14], [15], [16]. Also, it is not yet clear which of the neuron features that have been used for optimization are more important to capture realistic dynamical behaviours at the network level [17].

In [10], Hertag et al. showed that the process of fitting neuron models to frequency current (F-I) current clamp and sub-rheobase voltage-current (V-I) data constitutes a sufficient estimator of the firing patterns of a real neuron. Their method, however, based on analytical approximation of F-I/V-I curves, does not assess electrophysiological properties or statistical variations of a neuron, two important properties for large-scale simulations.

The purpose of this work is to build a robust automatic method for parameter optimization, able to overcome the problems posed by the previous approaches, as well as an implementation of this method that minimizes its significant computational cost. Our method, based on both global and local optimization algorithms, combines the ideas of F-I/V-I tuning with electrophysiological and statistical properties estimation. In addition, a new python toolbox provides a GPU-accelerated implementation of this method, based on [18], able to complete the optimization process in a matter of minutes, requiring minimized user intervention.

To assess the performance of the proposed method, we examined to what extent it is able to improve a model of a cat's thalamocortical (TC) relay cell, previously fitted to membrane potential traces. The resulting model was a closer fit to the real cell properties than previous approaches. We found that it was able to reproduce firing patterns which are unique for TC neurons [19] without any relevant training. Also the employment of this model as the building block of a general thalamic nucleus resulted in a more biologically plausible simulation than previous optimized neurons.

II. Methodology

A. Neuron models

Our methodology can be applied to any phenomenological neuron model. The default selection in our toolbox is the Izhikevich [20] simple model which is shown to have the simplest possible form that is able to reproduce the majority of the computational properties of brain cells and have indistinguishable behaviour of *in vitro* and *in vivo* recordings [14]. The membrane potential v of this model is governed by

$$C\frac{dv}{dt} = k(v - v_r)(v - v_t) - u + I \tag{1}$$

$$\frac{du}{dt} = a\big(b(v - v_r) - u\big) \tag{2}$$

Zafeirios Fountas and Murray Shanahan are both with the Department of Computing, Imperial College London, London, United Kingdom, email: {zfountas, m.shanahan}@imperial.ac.uk

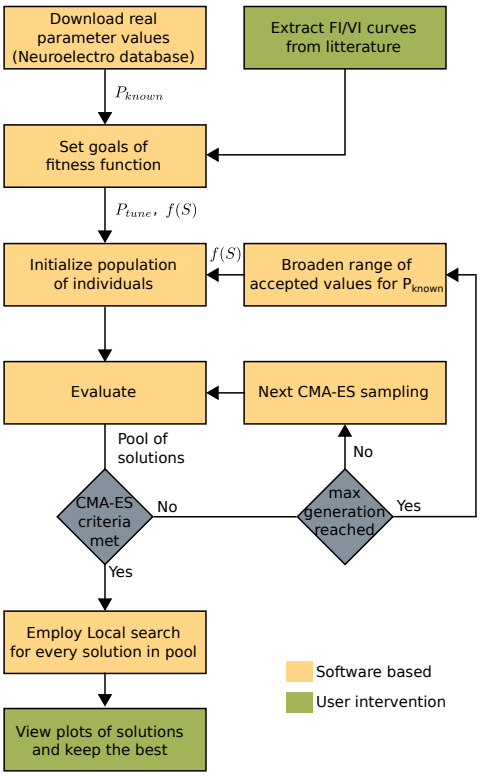

Fig. 1. Overview of the method used to find the mean values of the parameter set for the targeted neuron. P_{known}: The parameters whose real values were found in NeuroElectro database. P_{tune}: The parameters that need to be tuned. $f(S)$: The fitness function.

where u a phenomenological recovery variable, C the membrane capacitance, v_r the resting membrane potential, v_t the instantaneous threshold potential and finally a, b and k are three abstract parameters of the model. The neuron fires an action potential when the voltage exceeds the threshold value v_{peak} and the two variables of the model reset to

$$\begin{aligned} v &\rightarrow c \\ u &\rightarrow u + d \end{aligned} \tag{3}$$

where c and d are two further abstract parameters. Finally, I represents any input current that can be either dendritic or synaptic. For the purpose of this optimization algorithm, we do not model any chemical synapses and the input current of each neuron can be simply described as $I = I_{spon} + I_{injected}$, to take into account a general injected current that varies in different experiments as well as the default constant current I_{spon} that leads to the spontaneous activity of the cell.

For the rest of this paper we will use equations 1-3 to generate analytical solutions when needed but a similar process can be applied to the majority of the known phenomenological models.

B. Optimization algorithm

The design of an optimization algorithm for spiking neurons entails two main independent challenges [12]. First is the definition of a function that is able to evaluate robustly the fitness of a model to real biological data, and the second is the choice of a heuristic technique that can use this function to search for the optimal solution in the space of available parameters. Starting with the latter, we used a global search method to identify areas of potentially optimal solutions and a local search method to further optimize solutions in each of these areas [9]. This combination of global and local search provides a balance between exploration and exploitation [21] and facilitates the avoidance of local maxima convergence [12].

Figure 1 illustrates the high-level steps of the optimization method proposed. Once the fitness function is defined based on available experimental data, an evolutionary strategy called covariance matrix adaptation (CMA-ES) [22] is used to find areas of interest in the large and continuous parameter space. This method is an optimized evolutionary algorithm (EA) where the mutations represent real perturbations taken from a normal distribution, adapted to the local fitness landscape. Thus, this method is particularly suitable for ill-conditioned fitness functions, such as the multi-objective function described later in this section.

The best unique solutions of the EA are stored in a pool that divides the parameter space into grid cells, where each cell can contain up to one solution. This pool is updated after each generation according to the following algorithm

function UpdatePool(S)
if Pool $<$ PoolSize **then**
  Add S to Pool
else if $S_{score} <$ worst(Pool)$_{score}$ **then**
  for p **in** Pool **do**
    if $p \div$ GrSize $= S \div$ GrSize **and** $S_{score} < p_{score}$ **then**
      Replace p with S and **return**
    end if
  end for
  Replace worst(Pool) with S
end if

where S is the candidate solution, PoolSize the maximum size of the pool and GrSize a parameter that controls the size of the area enclosed by each grid cell.

After a pre-defined number of generations, or if the fitness criteria are met, the EA stops and the final potential solutions in the pool are returned. An exception is made for any solution S_i that $S_i \mod \text{GridSize} \approx 0$ and $\|S_i - S_j\| <$ GridSize for S_j being another solution. The former condition indicates that the the grid cell occupied by S_i might have a neighbour with a better fitness score and the latter condition verifies this assumption. Therefore, these solutions are treated as redundant and removed from the pool.

Next, each solution in the resulting pool becomes the subject of further local optimization based on the simplex algorithm [23]. This method, unlike faster and more efficient gradient-based alternatives [7], has been shown to overcome the problem of noisy parameter spaces, and thus it is a natural choice for the purpose of this algorithm.

The final step of the process is to approximate the variations in the observed electrophysiological properties of the real cell, which can be used for more realistic multi-neuron simulations. First, the neuron properties that are used as parameters of the model (e.g. AP threshold or resting potential) are sampled from a normal distribution $\mathcal{N}(\mu_i, \sigma_i)$ where μ_i is the value

of the optimal solution, σ_i is the standard deviation of the real recordings and i the parameter. Finally, if there are more known properties that do not correspond to model parameters, the CMA-ES algorithm is used again to optimize σ_i for the rest of the parameters. This time, the fitness function has a single objective which is the minimization of the difference between the real standard deviations and the ones sampled from 1000 instances of the model.

C. Derivation of initial parameters and parameter space

The initial values of the neuron parameters that need to be optimized, are sampled from a uniform distribution $\mathcal{U}(a_i, b_i)$, where a_i and b_i express the logical limitations of the parameter i. Also, parameters with values that can be obtained from experimental studies (e.g. from the NeuroElectro database) are sampled from a normal distribution, with standard deviation obtained from the same source.

Whenever possible, some of the parameters of the neuron model are inferred by the rest and their values are calculated before any other step of the fitness function. This, however, depends on the equations of the chosen neuron model that can be used to infer extra relations between parameters. If a found relation holds for every possible value of the rest "free" parameters and equation variables, the deduced parameter is not used in the ES chromosome. In the opposite case, these relations impose restrictions to the parameter space but do not reduce its dimension.

Examples of such relations for the simple integrated-and-fire (IF) and the more realistic adaptive exponential IF neuron models can be found in [10].

For the Izhikevich model given in (1-3), the parameters b and k can be derived from the equations

$$b = \frac{1}{R} + k(v_r - v_t) \tag{4}$$

$$I_{rheo} = \max\{b(v - v_r) - k(v - v_r)(v - v_t)\} \tag{5}$$

were R is the input resistance and I_{rheo} the rheobase current of the cell. These relations however, which are explained in [14], rely on the assumption that $b < 0$ and thus fall into the second category of relations described above.

Also, as shown in Section III, the default current of the neuron I_{spon} can be derived as the difference between the model's rheobase current and the real rheobase current of the neuron I_{rheo} obtained from any known F-I curve (Figure 3.B).

D. Fitness function

This is arguably the most crucial feature of the optimization method since it defines the criteria by which the optimal solution will be determined. The function proposed here is based on the approximation of F-I and V-I relations in different initial conditions of the neuron as well as any available electrophysiological parameter values. This multi-objective nature requires a careful consideration of each of these criteria, for which a brief description is given below.

F-I curves: A frequency-current (F-I) curve is used to describe relations between various amplitudes of injected current in a neural cell and the action potentials evoked as a response to this current. These relations have been used for the optimization of single-point neurons and shown to be sufficient for the reproduction of spike times in different cortical cells [10].

The proposed algorithm here is using F-I curves in different cases that include transient and steady state curves for different initial currents. Both categories are important capturing neuron's behaviour. Transient relations encode information regarding the initial response of a cell (such as rebound dynamics), while steady-state relations show the actual spike frequency and the overall behaviour of the cell. The fitness of the individual with respect to an F-I curve f_{FI} is given as root-mean-square difference between the experimental data points and the corresponding simulated neurons.

V-I curves: Similarly with above, V-I relations are also divided into instantaneous and steady-state, and they are a valuable source of information for the behaviour of the cell. Above the spiking threshold, (dynamic) V-I curves have been shown to be a self-contained method for accurate prediction [24]. Also, at currents bellow the rheobase level, they reflect the sub-threshold dynamics of the cell which are not easily approximated with F-I relations.

The V-I fitness of the individual f_{VI} can be obtained with the same methodology as above, by simulating a number of model neurons equal to the number of real data points. In some cases, depending on the neuron model that is used, sub-threshold V-I curves can be derived directly form its analytical equations (See section III).

Fitness of scalar values: The fitness level of electrophysiological properties is also an important objective of the function. Often, such properties represent crucial features of a targeted neuron that need to be emphasised in the tuning process. Additionally, in case of an insufficient amount of available F-I/V-I experimental data, these properties provide a valuable alternative for the approximation of instantaneous and steady-state dynamical phenomena.

Some of the most important properties that have been tested with this method include the resting potential of a neuron, action potential (AP) threshold, amplitude and width, after-hyperpolarization amplitude and width, cell capacitance, input resistance and adaptation ratio. The real values of these parameters can be obtained from centralized databases such as [25], where cross-study statistics of multiple neuron recordings can be also easily extracted.

Unlike the previous cases, the fitness formula of an individual here is using directly the real statistics via a normalized Gaussian function. That is, if x_j is the value of the parameter j of an individual, the fitness of this parameter is given by

$$f_{sc}\left(x_j\right) = e^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}} \tag{6}$$

where μ_j and σ_j are the mean and standard deviation of the real available data for the parameter j.

The value of σ_j encodes the range of accepted values and provides a good approximation of the corresponding fitness

weights. Hence, the final fitness function has the form

$$f(S) = \sum_{i \in C_{FI}} w_i f_{FI} + \sum_{i \in C_{FI}} w_i f_{VI} + \sum_{j \in P_{known}} f_{sc}(x_j) \quad (7)$$

All weights w_i, although can be defined by the user, are set to favour steady-state and transient F-I curves over V-I curves as a default. The reason is that, in our experiments, the former two cases were found to influence more the dynamical behaviour of a neuron, and be less likely to over-fit.

E. Termination

If the algorithm does not return a satisfying solution, after the completion of the above process, the range of the accepted values for the known neuron properties broadens (by increasing the value of σ_j) and the process starts again.

In the opposite case, the algorithm has to overcome the final problem of over-fitting in any of the above objectives. Frequently, the targeted neuron properties are obtained from recordings from different cells and, depending on the model used, there is no guarantee that a perfect individual that matches parameters and all V-I and F-I curves exists. Since there is no obvious solution to this issue, the algorithm keeps track of all unique potential solutions via the pool described earlier in this section. Hence, after the successful termination of the process, all individuals in the pool should be visualized (for an example see Figure 3) and shown to the user.

F. The toolbox

As a part of this study, a new optimization toolbox was developed, based on the above method and written in python. The most important feature of this toolbox is its high performance, which is achieved via the underlying spiking neural network simulator, NeMo [18]. NeMo is based on a C++/CUDA backend and delivers a high performance when simulating large-scale neural networks on graphics processing units (GPUs). In a state-of-the-art graphics card set up, NeMo is able to simulate up to 420.000 Izhikevich neurons in real time, connected with a realistic number of synapses, a number that corresponds to 4.2 billion spike events per second.

Using this framework, the calculation of the fitness function used by the above method can be almost fully parallelized for each generation of the CMA-ES (Figure 2). Before the beginning of the first generation, a new NeMo simulation is initialized. The network of this simulation contains N times the neurons needed for each calculation of the fitness function, where N is the number of individuals.

In case that the available system accommodates M GPUs instead of one, the toolbox automatically generates M NeMo simulations and distributes the calculations equally.

For the implementation of the evolutionary algorithm, the python library deap [26] was preferred, since it provides a significant number of alternative algorithms that can be also used, instead of CMA-ES, if required. Also, any local search method used in the toolbox is implemented via the standard optimization toolbox of the python library SciPy [27].

One of the major aims of this toolbox is to reduce the user intervention throughout the duration of the optimization process. As shown in Figure 1, the only requirement of the method is the location and extraction of the F-I and V-I relations that will be used by the fitness function. Following the same principle, the toolbox interfaces with NeuroElectro [25], an online database of experimentally found electrophysiological properties, that uses text mining to extract information from the published literature and thus updates on a daily basis. When the user initializes an instance of the toolbox and provides the name of the targeted neuron, every available statistic for this neuron will be downloaded, parsed and finally used during the optimization process.

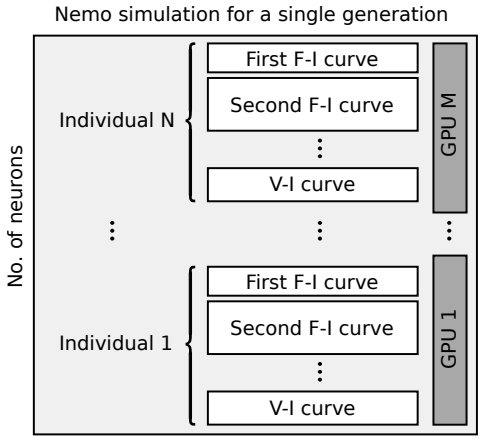

Fig. 2. A single neural network is initialized in every generation of the ES used and it is accommodated equally in all (M) available GPUs of the system. As a result, the fitness function for each individual is calculated concurrently taking advantage of NeMo's fast GPU calculation abilities.

The first finalized version of the toolbox will be released in http://nemosim.sourceforge.net.

III. Evaluation

A. Thalamocortical relay neuron

Dynamically, thalamocortical (TC) relay cells constitute an interesting category of spiking neurons since they integrate two spiking patterns with different dynamical behaviours. When the cell receives enough hyperpolarization, the existence of low-threshold activated Ca^{2+} current can cause a transient depolarization and, as a result, strong bursting activity, while at less hyperpolarized membrane potentials these neurons produce regular tonic spikes [29], [28]. Both these modes act as mechanisms to relay information that originates from the senses and other sub-cortical structures and is directed to the cortex. A unique characteristic that is shared among all TC relay cells is that the activation of the Ca^{2+} burst has a nearly "all-or-none" appearance [19]. Apart from this interesting behaviour, the fact that this neuron has been the subject of optimization with a different method [14] as well as the availability of a substantial amount of data in NeuroElectro database and in [28], made it a reasonable choice for the evaluation of the proposed algorithm.

In [14], Izhikevich suggested an adjustment to the simple model for the accurate simulation of TC relay neurons. Equation 2 splits into two parts depending on the level of membrane potential in the neuron.

$$\frac{du}{dt} = \begin{cases} a\big(b(v - (v_r - m)) - u\big) & \text{if } v \leq v_r - m \\ -au & \text{otherwise} \end{cases} \quad (8)$$

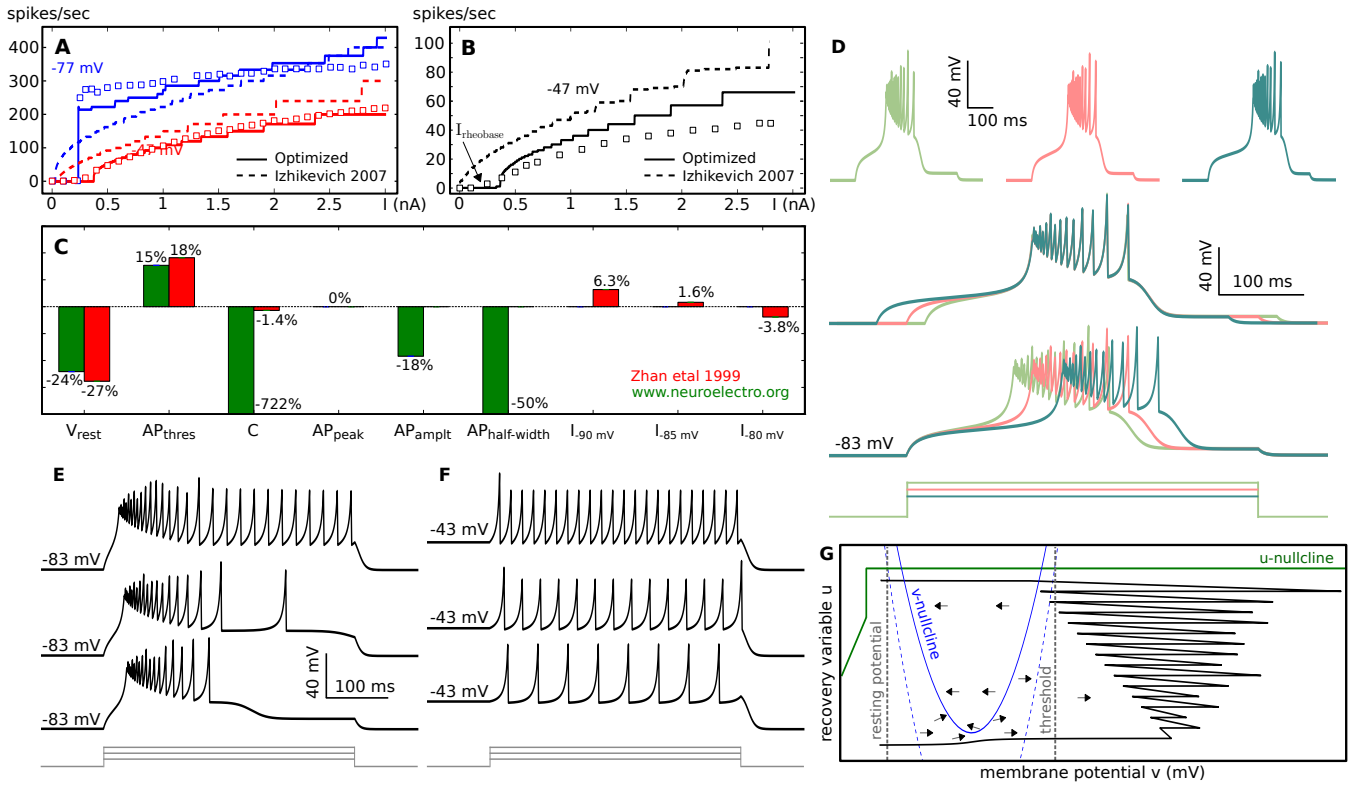

Fig. 3. Properties of the resulting TC neuron model that arise after $400ms$ current stimulation and replicate previously obtained data in [28]. **A:** Total number of spikes for current injections of different amplitudes and two different holding potentials. **B:** Firing frequency of the first 6 spikes for the same current injections. The symbols □ in A and B represent real recordings and the dashed lines represent the optimized TC model in [14]. **C:** Difference in various electrical properties of the model compared to data in NeuroElectro and [28]. **D:** Activation of the bursting mode shows the "all-or-none" nature of the Ca^{2+} activated spikes. After strong hyperpolarization, the membrane potential was stimulated with extra 240, 250 and $260pA$. The traces in the middle graph are time-shifted until the bursting pattern overlaps. The response time of the model is inversely proportional to the amplitude of the injected current. **E-F:** Response of the model for two different holding potentials. The injected currents in both cases are 486, 535.5 and $700pA$. **G:** Phase plane of the TC model for the same run as in E ($486pA$). The dashed line represents the v-nullcline for zero injected current.

The new parameter m is positive in order to prohibit the activation of the Ca^{2+} current during the resting state of the neuron. When the membrane potential exceeds the threshold value $v_{peak} + 0.1u$, the neuron fires a spike and the two variables of the model reset to

$$\begin{aligned} v &\rightarrow c - 0.1u \\ u &\rightarrow u + d \end{aligned} \tag{9}$$

With this adjustment, the purpose of the recovery variable u changes in order to capture the effects of the low-threshold Ca^{2+} current. Thus, with the proper parameter fitting, equations (1), (8) and (9) are able to reproduce firing traces of TC neurons in both modes [14]. However, it is not clear to what extent this model could be used to populate a biologically plausible model of a thalamic nucleus.

Taking the above into account, a new optimization attempt of the TC neuron was made, using the algorithm and the toolbox described in Section II. The simple model was used with the same adjustments described here, keeping only the value of parameter $m = 5mV$ from the model in [14].

To calculate I_{spon}, we assume that the membrane potential of the neuron is around the AP threshold, where the input current of the cell will be equal to the rheobase current $I = I_{rheo}$. Since $b = 0$ for large membrane potentials, the system (1)-(8) behaves as a quadratic integrate-and-fire neuron and it has two eigenvalues

$$\lambda_1 = a \tag{10}$$

$$\lambda_2 = \frac{k}{C}(2v - v_r - v_t) \tag{11}$$

The values of $\lambda_{1,2}$ are always real, hence the transition from the resting state to firing will take place via a saddle-node bifurcation [14]. By definition, a condition for the classification of a bifurcation to saddle-node is the existence of two real eigenvalues one of which must be equal to zero. Therefore, from Equation (11), the membrane potential close to the rheobase will have the value $v_{rheo} = \frac{v_r + v_t}{2}$. In addition, the equilibrium points of the system can be found from the intersection points of their two nullclines, by solving the system (1)-(8) for $\frac{dv}{dt} = \frac{du}{dt} = 0$. The resulting equation is

$$I(v) = k(v - v_r)(v - v_t) \tag{12}$$

and provides an analytical method for the calculation of all V-I relations of the neuron for voltages between $v_r - m$ and the rheobase, as well as the transient-state V-I relations below this interval. By applying the value of v_{rheo} to (12), we can get the rheobase current of the neuron

$$I_{rheo} = \frac{k}{4}(v_t - v_r)^2 \tag{13}$$

The difference in the value of the rheobase current obtained from (13) and the real rheobase current, that can be found in the steady-state F-I curve in Figure 3.B, represents the spontaneous current I_{spon} that the neuron needs to better fit the physiological recordings. However, because of the existence of this constant current, the parameters v_r and v_t do not represent the resting and threshold potentials any more. To change that, we can merge the current I_{spon} to parameters v_r, v_t and m. This can be done if the equation

$$k(v - v_r)(v - v_t) + I = 0 \tag{14}$$

has two real solutions. The solution with the lowest value represents the neuron's resting potential v_r^{new} while the second solution represent the threshold potential v_t^{new}. Finally, the parameter m in equation 8 can be transformed to $m^{new} = v_r^{new} - v_r + m$.

The above techniques were taken into account for the calculation of the fitness function described in methodology. After 50 generations of the evolutionary strategy, 66 significantly different solutions were gathered and further optimized with a local search method. The resulting parameter values of the optimal solution are $Ispon = -100pA$, $C = 294pF$, $v_{peak} = 35mV$, $k = 1.2$, $v_r = -78.99mV$, $v_t = -38.71mV$, and $m = 2.81mV$ while the abstract parameters $a = 0.0002$, $b = 20.55$, $c = -49.22$ and $d = 21.8$.

Figure 3 illustrates the properties of the resulting neuron model and a comparison with real data and a previously optimized set of parameters. Although the fitness function used took into account only F-I and V-I relations as well as a few basic electrophysiological properties, our simulations with this model matched data and features of cat's TC neurons that were never explicitly programmed. For instance, in most cases, activation of Ca^{2+} current evoked almost identical bursts, indicating the "all-or-none" behaviour of the cell.

The result of the optimization algorithm could be further improved if one additional recovery variable u_2 was used along with v and u. Since u is inactive in the regular spiking mode of the neuron, the model acts as a quadratic integrate-and-fire neuron with limited dynamical behaviours. Hence, with the introduction of u_2, the model would be able to exhibit rich dynamics in both modes and it might have more realistic F-I relations. However, for the purpose of this study which was the evaluation of the proposed technique, the resulting TC model needed to be comparable with previous optimized attempts, which have used only one recovery variable [14].

B. Simulation of a thalamic nucleus

For further validation of the resulting TC model, we attempted to simulate a neural ensemble, whose neuron types and connectivity follow the general rules of a thalamic nucleus. Two versions of this ensemble were created; the one using the current TC model and the second using the previous approach in [14] that was also used in the previous section. For clarity, we will refer to these ensembles as E_1 and E_2 respectively.

In both cases, the network comprises 800 TC relay neurons and 200 thalamic interneurons, keeping the balance that is usually found in experimental studies [30], [19] and used in other computational models [4]. The synaptic input to the neurons was modelled using a typical conductance-based

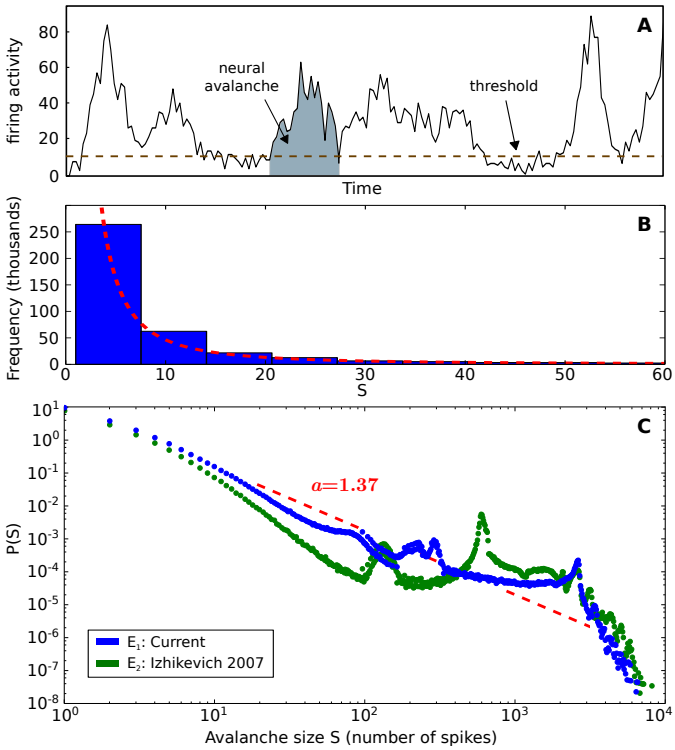

Fig. 4. **A:** Method of avalanche size extraction. **B, C:** Histogram of 0.5M avalanches for the two simulated thalamic ensembles. The dashed line represents the estimated power-law distribution for E_1, with the best fit.

approach [31] with default parameters for AMPA, NMDA and GABA neurotransmitters obtained from the same source. Each thalamic interneuron sends gabaergic connections to 25% of all other neurons in the group, while TC neurons have only excitatory afferents to 25% of the interneurons. The delays of all connections were randomized uniformly from 1 to 40 ms.

Furthermore, each neuron in the network was forced to spike with a rate taken from a Poisson distribution, and corresponds to 5 spikes/second, while TC neurons received an extra injected current I_{inj}, fixed at a single value. The distribution of weights, as well as the injected current I_{inj} were tuned in order to maintain a spontaneous firing rate of approximately 20 spikes/second [32].

Two different tuning scenarios were examined for more accurate comparison of the two TC models. First, the network connectivity and input current were kept fixed for both E_1 and E_2 and they were tuned to approximate the thalamic spontaneous firing rate. In a second experiment, connectivity remained again fixed but the injected current was tuned to make the two nuclei fire at exactly the same spontaneous frequency. To minimize the affected parameters in all cases, the inhibitory-to-excitatory weights of E_2 were normalized with respect to a ratio factor C_1/C_2, to eliminate the effect of different capacitance values to the synaptic current (see equation 1).

As the measure of comparison, the dynamical behaviour of E_1 and E_2 was assessed by means of the neural avalanche events in each ensemble. In biological neural systems, the size of neural avalanches over time has been found to follow a power law distribution ($P(x) \sim x^{-a}$), both in cortical

slices [33] and in vivo [34], with an exponent a around 1.5 [33]. The degree of approximation of this distribution is a good indicator of scale-free behaviour [34], [35] of the neural system and of whether it is operating near a critical regime[36].

The size of avalanches was measured by taking the area of neural firing activity, sampled in 1 ms bins, that exceeds the threshold of 10 spikes/ms for a number of continuously active bins (Figure 4.A). Figures 4.B and C illustrate the histograms of the avalanche sizes and the corresponding best-fit power law distribution.

In both simulated experiments, a Kolmogorov-Smirnoff (KS) test showed that, with p-values 0.26% and 0.007% respectively, the distributions of avalanche sizes in the two nuclei are different. By subtracting the final cutoff points, the result in the KS test changed to 0.42% and 0.16%, still rejecting the null hypothesis that the two samples come from the same distribution.

The next step was to run a test to see whether the two size distributions conform to a power law distribution. Due to the limited number of neurons in the system, these distributions loose accuracy towards the tail and start to bend downward, exhibiting an exponential cutoff that needs to be pruned before any fitting attempt (Figure 4.C).

Following the methodology in [37], the tests failed in both experiments and both models of TC neurons. However, this can be partially justified due to the rigorous nature of this test. In both simulated experiments, the KS distance between the histogram of the current model and the best-fit power law was found to be significantly smaller (in the order of 20% for the illustrated experiment) than the other approach. Finally, the distribution in Figure 4 could be also dramatically improved, perhaps in a degree above the requirements of the test, if the connectivity of the network was optimized by using, for instance, some form of synaptic plasticity [36]. Nonetheless, the results here suggest that networks using the optimized TC model in E_1 are closer to a critical state, a fact that increases the biological plausibility of the simulation.

C. Benchmarking

To assess the computational performance of the presented toolbox, we ran a number of simulations on different hardware platforms, optimizing the same neuron and recording the execution time of each generation. In order to calculate the fitness of each individual, the equations of 100 simulated neurons were numerically calculated for 2 seconds with 0.25 ms time step. The population of individuals was set to be 500, which corresponds to 50,000 neurons for each generation.

The average execution times are illustrated in Figure 5. In this figure, initialization time refers to all stages needed for the initialization of the optimization algorithm apart from the time of interaction with the online database NeuroElectro. All simulations were executed on 64bit machines with 32GB available memory and the same hard drive, while the evolutionary strategy was forced to continue for 40 generations.

The results reveal a significant speed up in GPU processing over CPU, even in the case of a high-performance state-of-the-art CPU processor with multi-threading support. This speedup ranges from 1.35 to 3.65 for the 8-threaded simulation based on

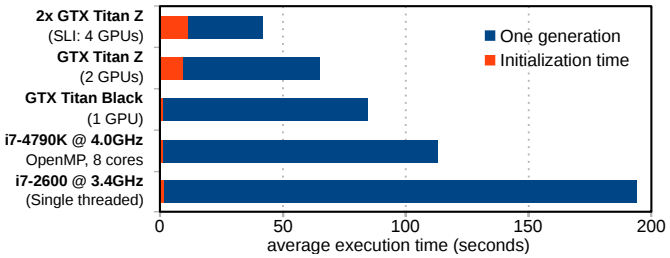

Fig. 5. Average execution time of one generation of the CMA-ES for different hardware. As the standard deviation was always < 2 seconds, it is omitted from the graph.

the OpenMP API and from 2.31 to 6.25 for a single threaded simulation. Initialization times were negligible in all cases (< 12 seconds) with small increases in the case of multiple GPU simulations. These results indicate that there is no obvious bottleneck in our implementation and that execution time depends highly on the spiking neuron simulation platform used.

IV. Concluding Remarks

The main contributions of this paper are as follows. First, the presentation of a fast optimization process that produces parameters and statistics for phenomenological neuron models which can be used for large-scale biological plausible simulations. Second, a python toolbox that implements the above process using a GPU-based high-performance spiking neural network simulator. The low requirements of this method along with the hardware acceleration allow the complete tuning process to be carried out in a manner of minutes and thus constitute the key feature of the toolbox. In addition, the initialization of this algorithm is automated, requiring a minimized user intervention. Any experimentally found neuron properties can be located in an online database and downloaded automatically.

The third and final contribution is a new model of the TC relay neuron, that is shown to fit better than a previous approach to experimental data, and to generate more realistic large-scale simulations. The detection of statistical differences in networks constructed with this new model and the previously optimized TC neuron is of high importance considering the already high accuracy of the latter.

One of the major aims of this work is to make the optimization procedure for this type of neuron models as automated and fast as possible. Nonetheless, the user of our toolbox still needs to detect a number of experimentally found F-I/V-I relations, to achieve high performance. This type of information is widely available in the literature of *in-vitro* cell recordings, either in the form of a curve or, implicitly, through other experiments. This issue can be overcome in the future, if more centralized databases for neurological data, such as NeuroElectro, arise.

We are currently extending the toolbox functionality in two ways. First, we introduce a method where the optimization includes the detection of the simplest possible model that is able to produce a good fit to the available data. In its current stage, this method starts to optimize a quadratic IF neuron as described in Section II. If no individual can be found with a

good fit, the process starts again incrementing the number of helping variables u_i as described in the last paragraph of the section III.A. Hence, the process is likely to return a solution with a better fit, as well as the least number of variables needed.

The second extension concerns optimization at the network level. The parameter sets currently produced by the toolbox are accompanied by estimates of their variation, that can be used to produce more realistic neural ensembles. Hence, based on the foundation of [13] and the processing capabilities of NeMo, we are developing a method that can optimize the connectivity parameters between and within ensembles, as well as other important network features, based on knowledge on their behaviour in different simulation scenarios.

ACKNOWLEDGMENT

This work was supported by an EPSRC studentship award. The authors would like to thank NVIDIA for donating the hardware that was used for the GPU computations and Mr. Pedro Martinez Mediano for his valuable recommendations and discussions.

REFERENCES

[1] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen, "A large-scale model of the functioning brain," *science*, vol. 338, no. 6111, pp. 1202–1205, 2012.

[2] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Frontiers in neuroscience*, vol. 7, 2013.

[3] D. Gamez, Z. Fountas, and A. K. Fidjeland, "A neurally controlled computer game avatar with humanlike behavior," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 5, no. 1, pp. 1–14, 2013.

[4] E. M. Izhikevich and G. M. Edelman, "Large-scale model of mammalian thalamocortical systems," *Proceedings of the national academy of sciences*, vol. 105, no. 9, pp. 3593–3598, 2008.

[5] M. D. Humphries, R. Wood, and K. Gurney, "Reconstructing the three-dimensional gabaergic microcircuit of the striatum," *PLoS computational biology*, vol. 6, no. 11, p. e1001011, 2010.

[6] Z. Fountas and M. Shanahan, "Phase offset between slow oscillatory cortical inputs influences competition in a model of the basal ganglia," in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 2407–2414.

[7] U. S. Bhalla and J. M. Bower, "Exploring parameter space in detailed single neuron models: simulations of the mitral and granule cells of the olfactory bulb," *Journal of Neurophysiology*, vol. 69, no. 6, pp. 1948–1965, 1993.

[8] C. Rossant, D. F. Goodman, B. Fontaine, J. Platkiewicz, A. K. Magnusson, and R. Brette, "Fitting neuron models to spike trains," *Frontiers in neuroscience*, vol. 5, p. 9, 2011.

[9] P. Achard and E. De Schutter, "Complex parameter landscape for a complex neuron model," *PLoS Computational Biology*, vol. 2, no. 7, p. e94, 2006.

[10] L. Hertäg, J. Hass, T. Golovko, and D. Durstewitz, "An approximation to the adaptive exponential integrate-and-fire neuron model allows fast and predictive fitting to physiological data," *Frontiers in computational neuroscience*, vol. 6, 2012.

[11] P. Friedrich, M. Vella, A. I. Gulyás, T. F. Freund, and S. Káli, "A flexible, interactive software tool for fitting the parameters of neuronal models," *Frontiers in neuroinformatics*, vol. 8, 2014.

[12] W. Van Geit, E. De Schutter, and P. Achard, "Automated neuron model optimization techniques: a review," *Biological cybernetics*, vol. 99, no. 4-5, pp. 241–251, 2008.

[13] K. D. Carlson, J. M. Nageswaran, N. Dutt, and J. L. Krichmar, "An efficient automated parameter tuning framework for spiking neural networks," *Frontiers in neuroscience*, vol. 8, 2014.

[14] E. M. Izhikevich, *Dynamical systems in neuroscience*. MIT Press, Cambridge, MA, 2007.

[15] A. A. Prinz, "Neuronal parameter optimization," *Scholarpedia*, vol. 2, no. 1, p. 1903, 2007.

[16] V. Booth, "Neuronal model hand-tuning," in *Encyclopedia of Computational Neuroscience*. Springer, 2014, pp. 1–3.

[17] P. Li and Q. D. Vu, "Identification of parameter correlations for parameter estimation in dynamic biological models," *BMC systems biology*, vol. 7, no. 1, p. 91, 2013.

[18] A. K. Fidjeland and M. P. Shanahan, "Accelerated simulation of spiking neural networks using GPUs," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, 2010, pp. 1–8.

[19] S. M. Sherman and R. W. Guillery, *Exploring the thalamus and its role in cortical function*. MIT Press, Cambridge, MA, 2006.

[20] E. M. Izhikevich *et al.*, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[21] A. E. Eiben and C. Schippers, "On evolutionary exploration and exploitation," *Fundamenta Informaticae*, vol. 35, no. 1, pp. 35–50, 1998.

[22] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[23] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[24] L. Badel, S. Lefort, T. K. Berger, C. C. Petersen, W. Gerstner, and M. J. Richardson, "Extracting non-linear integrate-and-fire models from experimental data using dynamic i–v curves," *Biological cybernetics*, vol. 99, no. 4-5, pp. 361–370, 2008.

[25] S. J. Tripathy, J. Savitskaya, S. D. Burton, N. N. Urban, and R. C. Gerkin, "Neuroelectro: a window to the world's neuron electrophysiology data," *Frontiers in neuroinformatics*, vol. 8, 2014.

[26] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.

[27] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–2015. [Online]. Available: http://www.scipy.org/

[28] X. Zhan, C. Cox, J. Rinzel, and S. M. Sherman, "Current clamp and modeling studies of low-threshold calcium spikes in cells of the cats lateral geniculate nucleus," *Journal of neurophysiology*, vol. 81, no. 5, pp. 2360–2373, 1999.

[29] H. Jahnsen and R. Llinas, "Electrophysiological properties of guinea-pig thalamic neurones: an in vitro study." *The Journal of physiology*, vol. 349, no. 1, pp. 205–226, 1984.

[30] J. A. Winer and D. T. Larue, "Evolution of gabaergic circuitry in the mammalian medial geniculate body," *Proceedings of the National Academy of Sciences*, vol. 93, no. 7, pp. 3083–3087, 1996.

[31] P. Dayan and L. F. Abbott, *Theoretical neuroscience*. Cambridge, MA: MIT Press, 2001.

[32] C. J. Wilson and P. M. Groves, "Spontaneous firing patterns of identified spiny neurons in the rat neostriatum," *Brain research*, vol. 220, no. 1, pp. 67–80, 1981.

[33] J. M. Beggs and D. Plenz, "Neuronal avalanches in neocortical circuits," *The Journal of neuroscience*, vol. 23, no. 35, pp. 11 167–11 177, 2003.

[34] T. Petermann, T. C. Thiagarajan, M. A. Lebedev, M. A. Nicolelis, D. R. Chialvo, and D. Plenz, "Spontaneous cortical activity in awake monkeys composed of neuronal avalanches," *Proceedings of the National Academy of Sciences*, vol. 106, no. 37, pp. 15 921–15 926, 2009.

[35] B. J. He, "Scale-free brain activity: past, present, and future," *Trends in cognitive sciences*, vol. 18, no. 9, pp. 480–487, 2014.

[36] F. P. P. Teixeira and M. Shanahan, "Does plasticity promote criticality?" in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 2383–2390.

[37] A. Clauset, C. R. Shalizi, and M. E. Newman, "Power-law distributions in empirical data," *SIAM review*, vol. 51, no. 4, pp. 661–703, 2009.