Between Two Worlds: Surrogate Gradients for Gradient-Based Parameter Estimation in Simplified Neuron Models

1st Paul Mayer
KTH Royal Institue of Technology
Stockholm, Sweden
pmayer@kth.se

***Abstract*—Simplified neuron models like AdEx need extensive parameter fitting to faithfully replicate experimental neuron spiking behaviour. Traditional approaches use derivative-free methods (grid search, genetic algorithms), while gradient-based frameworks like Jaxley have shown success for HH-type models but lack support for simplified models due to their non-differentiable spike mechanism. We address this by implementing surrogate gradient-enabled AdEx in Jaxley and investigate which loss functions enable successful parameter fitting in bio-physical simulations. Our evaluation on striatal projection neuron recordings reveals that loss function design is critical when adapting gradient-based optimization. MSE-based optimization fails due to spike timing sensitivity, converging to non-spiking solutions. Feature-based losses better capture firing patterns, though both approaches achieved $\Gamma \approx 0$ for spike timing precision. These results highlight that adapting gradient techniques for biophysical fitting requires domain-specific loss functions rather than standard approaches.**

***Index Terms*—Automatic Gradient Descent, Integrate-and-Fire Model, Simplified Neuron Model, Surrogate Gradients, Parameter Estimation**

I. Introduction

In the last century, studying and modelling neurons has been seen as the gateway of understanding the brain in its entirety. What first started with very simple Leaky Integrate-and-Fire (LIF)-models, quickly became proving ground for more complex and more sophisticated ideas. Lack of computational resources has been a primary driver for why simplified neuron models were still developed, because they promised to be the only way to really model parts of the brain on super computers 20 years ago. With the steady increase in computing power, the research community shifted focus from those models in favour of more sophisticated Hodgkin-Huxley (HH)-type models, which require more computational resources to simulate and to fit parameters to. However, the idea of simulating the brain as whole has remained an unreachable goal for many decades, until now.

Computational power has reached whole new levels, and using computationally less intensive simplified models might be the key step to perform whole brain simulations. Adaptive models like Adaptive Exponential Integrate-and-Fire (AdEx) can faithfully capture the firing behaviour of diverse neuron types and therefore promise to be a prime candidate for realistic large-scale simulations. However, this needs very fast and efficient, reliable, and effective optimization methods to fit neuron models to experimental data. Due to the non differentiable nature of simplified models, optimization methods like grid search or evolutionary algorithms are used instead of gradient methods.

In this work, we investigate whether surrogate gradient techniques can enable gradient-based parameter estimation for simplified neuron models. Specifically, we ask:

1) Can surrogate gradients produce useful parameter updates for the AdEx model? And
2) what loss function design is required to achieve successful optimization?

To investigate this, we make the following contributions:

1) We implement a surrogate gradient-enabled AdEx model within the Jaxley framework, enabling gradient-based parameter optimization.
2) We investigate loss function requirements and compare voltage-based Mean Squared Error (MSE) loss against feature-based losses.
3) We find that MSE-based optimization fails to recover correct spiking behavior. While feature-based losses better replicated overall firing behaviour, significant challenges remain. Spike timing remains a crucial challenge. Both approaches failed to achieve useful spike timings — we measured a coincidence factor $\Gamma \approx 0$.

Further work is needed to enable efficient and effective parameter optimization using gradient-based optimization.

In this work we will demonstrate how to bridge two worlds: combine gradient optimisation using Jaxley for using simplified models in bio-physical neuron simulations. We start giving a brief background introduction in Section II and an overview over the related work in III. In Section IV we introduce our methodology, Section VI contains our findings and Section VII concludes with a discussion of the findings and future work.

II. Background

A. Neuron Models

Large-scale brain models are limited in size mostly by computational feasibility. Detailed bio-physical models like the HH-model can precisely predict membrane behaviour of a vast variety of neuron types. However, they require huge

computational resources to estimate the required parameters to fit the models to experimental data. Simplified Neuron Models models overcome this challenge, capturing most of the sophisticated neuron behaviour with only a fraction of the computational cost, enabling large-scale simulations [1]. The Adaptive Exponential Integrate-and-Fire (AdEx) model [2] provides a particularly interesting balance — it reproduces a variety of firing patterns but drastically reduces the number of necessary parameters of the model. This makes it a favourable model for large-scale neuron simulations [3].

The AdEx model improves the simpler LIF model in two ways: first, it introduces exponential behaviour when initialising the spike. Second, the model allows for adaptation, meaning it can change spiking frequencies or post-spike refractoriness based on their firing history. It is defined by the equations given in [2], [4]:

$$C\frac{dV}{dt} = -g_L(V - E_L) + g_L\Delta_T \exp(\frac{V - V_T}{\Delta_T}) + I - w, \tag{1}$$

$$\tau_w \frac{dw}{dt} = a(V - E_L) - w, \tag{2}$$

with the following reset condition:

$$\text{if } V > 0 \text{ mV} \quad \text{then} = \begin{cases} V \to V_r, \\ w \to w_r = w + b. \end{cases} \tag{3}$$

It describes the membrane potential over time $V(t)$ given an injected current $I(t)$. Equation (1) describes the change of potential. C is the membrane capacitance, g_L the leak conductance and E_L the effective resting potential. We can understand $-g_L(V - E_L)$ as the leak current that slowly pulls the membrane potential towards its resting potential. $g_L\Delta_T \exp(\frac{V-V_T}{\Delta_T})$ models the depolarisation initialised by the fast reacting sodium channels using an exponential function, modelled using a driving force based on an effective threshold potential V_T and a slope factor Δ_T. I is the injected current, and w is the adaptation current, which is described in equation 2.

The adaptation mechanism is controlled via an adaptation current which opposes depolarisation. Unlike the fast membrane behaviour, this current is related to the time constant τ_w, which controls the decay time. It is controlled by a (subthreshold adaptation) as well as b (spike-triggered adaptation).

Even though the number of parameters is significantly reduced compared to bio-physical HH-type models, it still requires parameter fitting — especially because some parameters (e.g. a, b or τ_w) do not have direct physiological counterparts (like C or g_L). For AdEx to perform as an effective tool, efficient methods for parameter optimizations are essential.

B. Parameter Tuning for Neuron Models

A central challenge in neuroscience has been to 'identify the parameters of detailed biophysical models, such that they match physiological measurements at scale' [5] in reasonable simulation runtimes. The zoo of optimisation techniques is large, spanning from evolutionary algorithms [6] to gradient descent [7], however, deciding which algorithm to use may often depend on the underlying properties of the model. `Jaxley` [5], a newly developed framework, uses automatic differentiation to quickly fit parameters of sophisticated biophysical models to experimental data and perform large-scale neuron simulations. Specifically, `Jaxley` allows to simulate multi-compartment models with complex structures (dendrites, axons), cable theory equations and HH-type ion channels. To perform parameter optimisation, `Jaxley` uses automatic differentiation (via `Jax`). It shows very promising results, matching or outperforming other state-of-the-art (e.g. genetic algorithms) in performance for small and large neuron models respectively. However, as previously stated, in many circumstances it still remains unfeasible to use HH-type models for large-scale simulations. `Jaxley` fails to support simplified models to perform parameter tuning. The fundamental limitation is that simplified models are not by default automatically differentiable. This is a result of the reset condition in equation 3. Addressing this issues requires special treatment: surrogate gradients.

C. Surrogate Gradients

Surrogate Gradients (SGs) were developed to adapt spiking neurons (usually using LIF neuron models) for training, primarily on classification tasks. To fit these models to real data, the typical forward pass - backward pass strategy needed to be enabled. Because LIF-type models have a discontinuous derivatives (see reset in equation 3), automatic differentiation cannot be performed without adjustments. To overcome this issue, prior work replaced the discontinues (heavyside-) functions derivative with a (e.g. sigmoid) surrogate, but only for the backward pass [8]. For training weights in Spiking Neural Networks (SNNs), this performed sufficiently well.

III. Related Work

Simplified Neuron Models historically were a necessity to deal with the fact that computing resources were too limited to perform large scale brain simulations with sophisticated ion-channel based models. With the rise of cheap and widely available supercomputing, focus shifted back to more detailed HH-type model. In recent years however, interest in simplified models resurged, driven by mainly two reasons: researchers wanting to progress Artificial Neural Networks (ANNs) to use more detailed spiking neurons and computational neuroscientists getting closer to feasibly simulating whole mammal brains.

A. Parameter Estimation for AdEx Model

A neuron model has to replicate behaviour of a huge variety of neurons. In simplified neuron models like AdEx, some parameters lack a real (measurable) counterpart. Therefore parameters have to be estimated using optimisation techniques. Traditionally, a huge variety (from evolutionary strategies to particle swarm optimizations) of approaches were used [6]. A very recent paper uses grid search over a feature space, not

gradients [9]. Recently genetic algorithms, multi-state single-agent stochastic search (MSASS) or Teaching-Learning-Based Optimization (TLBO) have been used successfully to tune parameters for the AdEx model to real voltage-clamp recordings [10], [11]. However, these approaches tend to use a lot of computational resources.

Hertag et al. [12] developed an analytical approximation to the AdEx model to speed up computations. Using automatic differentiation could also address this issue. Jones et al. [7] and Deistler et al. [5] demonstrated that gradient-based methods can effectively parametrize HH-type models. However, they require the inherently differentiable nature of HH-type models. Simplified models require additional treatment to benefit from this technique.

B. Parameter Learning using Surrogate Gradients in Spiking Neural Networks

Previous work explored using surrogate gradients for training weights of SNNs in classification tasks [8], [13], [14]. Gerum and Schilling [13] explored different surrogates (like sigmoid or esser function) to train a LIF based Neural Network (NN) for image classification tasks. The authors used `Keras` to perform automatic differentiation and backpropagation. Perez-Nieves et al. [14] used a sigmoidal surrogate gradient in the backward pass to train a spiking NN in `PyTorch` to perform a variety of different classification tasks. However, their work focuses on training network weights on task-based classification tasks.

Recent theoretical work [15] clarifies why surrogate gradients succeed despite approximating the true gradient. But fitting biophysical parameters to experimental recordings poses a different challenge: the loss function must quantify similarity between simulated and recorded spike trains. Whether standard choices like mean squared error suffice — or whether domain-specific alternatives are required — remains unexplored. This work investigates this question.

C. Benchmark Sets for Neuron Models

Neuron models are highly specialised and are fit to very specific data. Therefore it proves to be difficult to compare models systematically. Jolivet et al. [16] introduce a coincidence factor, which allows to evaluate neuron models. We use this concept to evaluate the quality of our trained parameters.

IV. Methodology

This work introduces two contributions: firstly, we implement the AdEx model within `Jaxley`, providing both standard and surrogate gradient variants. The surrogate implementation can easily adopted to Jaxleys existing simplified models: LIF and Izhikevich. Secondly, we use `Jaxley` to perform gradient-based optimisation. We investigate two different loss functions:

- MSE
- Guarino et al. [9] feature-based loss function.

Using gradient-based optimisation requires a differentiable loss function. Therefore we will introduce a gradient version of the feature-based loss function.

A. Integration of Surrogate Gradients into Jaxley

`Jaxley` is built on top of `JAX` and leverages its automatic differentiation capabilities for gradient-based parameter optimisation. Neuron dynamics in `Jaxley` are defined through `Channel` classes, which specify three core methods: `update_states()` for state variable evolution, `compute_current()` for membrane current contributions, and `init_state()` for initialisation of state variables. During simulation, `JAX` traces these functions to construct a computational graph, enabling automatic gradient computation and backpropagation.

1) The Differentiability Problem: Simplified neuron models employ a discrete reset mechanism when the membrane potential crosses a threshold. This reset is mathematically expressed using the Heavyside step function H:

$$H(v - v_{\text{th}}) = \begin{cases} 1 & \text{if } v \geq v_{\text{th}} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The derivative of the Heavyside function is zero almost everywhere (and undefined at the threshold):

$$\frac{\partial H}{\partial v} = 0 \quad \text{for } v \neq v_{\text{th}} \tag{5}$$

This causes gradients to vanish during back-propagation through spike events, which prevents gradient-based optimisation from working.

2) Surrogate Gradient Approach: To overcome this limitation, we employ surrogate gradients [8]: the forward pass uses the original Heavyside function to maintain correct spike dynamics, while the backward pass substitutes the function with a surrogate. This is implemented in `JAX` using `jax.custom_vjp`, which allows defining custom vector-Jacobian products.

We implement three surrogate gradient functions, each providing different gradient characteristics:

Sigmoid surrogate:

$$\frac{\partial \tilde{H}}{\partial x} = \beta \cdot \sigma(\beta x) \cdot (1 - \sigma(\beta x)) \tag{6}$$

where σ is the sigmoid function and β controls the steepness.

Exponential surrogate:

$$\frac{\partial \tilde{H}}{\partial x} = \beta \cdot \exp(-\beta |x|) \tag{7}$$

SuperSpike surrogate [17]:

$$\frac{\partial \tilde{H}}{\partial x} = \frac{1}{(\beta |x| + 1)^2} \tag{8}$$

All three surrogates are centred at the threshold ($x = v - v_{\text{th}}$). The hyperparameter β controls the sharpness: higher values produce steeper gradients at the threshold, while lower values spread the gradient over a wider voltage ranges.

3) Implementation Details: We reformulate the conditional reset from a discrete selection:

$$v_{\text{new}} = \begin{cases} v_{\text{reset}} & \text{if spike} \\ v & \text{otherwise} \end{cases} \tag{9}$$

to a continuous, differentiable form:

$$v_{\text{new}} = s \cdot v_{\text{reset}} + (1 - s) \cdot v \tag{10}$$

where $s = \tilde{H}(v - v_{\text{th}})$ is the surrogate spike indicator, to ensure differentiability through the reset mechanism. This formulation allows gradients to flow through the reset operation.

This approach can easily be applied to the existing LIF (`FireSurrogate`) and Izhikevich models in `Jaxley`, not only to the newly implemented AdEx model (`AdExSurrogate`).

B. AdEx Model Implementation

We implement the AdEx model as a `Jaxley` channel, following the formulation by Brette and Gerstner [4] (see Equations 1–3). Unlike HH-type channels that contribute currents to a shared voltage solver, simplified models like AdEx handle voltage integration directly within their `update_states()` method. We follow the implementation pattern of [5].

1) Numerical Integration: The AdEx model consists of two coupled differential equations requiring different numerical treatment:

Adaptation current w: Equation 2 is a linear dynamic system in w, allowing the use of exponential Euler integration. It has the analytical solution

$$w(t+\Delta t) = w(t) \cdot e^{-\Delta t/\tau_w} + a(V - E_L)(1 - e^{-\Delta t/\tau_w}) \tag{11}$$

which gives increased numerical stability compared to e.g. forward euler methods.

Membrane potential V: The exponential term in Equation 1 makes this equation nonlinear, requiring forward Euler integration:

$$V(t + \Delta t) = V(t) + \Delta t \cdot \frac{dV}{dt} \tag{12}$$

To prevent numerical overflow from the exponential spike mechanism, we clamp the argument:

$$\exp\left(\frac{V - V_T}{\Delta_T}\right) \to \exp\left(\min\left(\frac{V - V_T}{\Delta_T}, 10\right)\right) \tag{13}$$

2) Spike Detection and Reset: After each integration step, we check for threshold crossing ($V > v_{\text{threshold}}$). If the threshold is crossed, we apply the reset conditions from Equation 3. For standard `LIF`, `AdEx` or `Izhikevich`, this uses `jax.lax.select()` for conditional assignment. In the differentiable variants, we use the continuous formulation described in Section IV for both V and w:

$$V_{\text{new}} = s \cdot V_r + (1 - s) \cdot V, \tag{14}$$

$$w_{\text{new}} = s \cdot (w + b) + (1 - s) \cdot w, \tag{15}$$

where $s : V \to [0, 1]$ is the surrogate of the heavyside function.

TABLE I
ADEX MODEL PARAMETERS AND DEFAULT VALUES.

Parameter	Symbol	Default	Unit
Membrane capacitance	C	200	pF
Leak conductance	g_L	10	nS
Leak reversal	E_L	-70	mV
Threshold potential	V_T	-50	mV
Slope factor	Δ_T	2	mV
Reset potential	V_r	-58	mV
Detection threshold	$v_{\text{threshold}}$	0	mV
Adaptation time constant	τ_w	30	ms
Subthreshold adaptation	a	2	nS
Spike-triggered adaptation	b	0	pA
Stimulation current	I	500	pA

3) Model Parameters: Table I summarises the AdEx parameters and their default values. Parameters without direct physiological counterparts (e.g., a, b, τ_w) are primary targets for optimisation.

AdEx is defined using absolute capacitance (pF). However, Jaxley is targeted at bio-physical HH-type simulations, and therefore works with cell morphology and gemoetries. It therefore expects specific capacitance ($\mu F/cm^2$). To circumvent this problem, we compute a fictitious cylindrical geometries whose surface area yields the desired total capacitance assuming jaxleys standard specific capacitance ($1\mu F/cm^2$). This scales injected currents to maintain correct dynamics.

C. Implementation Verification

To verify the correctness of our AdEx implementation, we compare simulation outputs against Brian2 [18], a widely-used reference simulator for spiking neural networks.

1) Comparison Setup: Both simulators are configured identically:

- Time step: $\Delta t = 0.01$ ms
- Simulation duration: 500 ms
- Stimulation duration: 400 ms
- Initial conditions: $V(0) = V_r$, $w(0) = 0$

2) Test Cases: We evaluate three parameter configurations from Naud et al. [2] that produce distinct firing patterns:

Tonic spiking:
Regular spiking without spike-frequency adaptation, serving as a baseline case. The parameters are the default parameters (see I).

Adaptation:
Strong spike-triggered adaptation with slow decay, producing decreasing firing rates over time. The parameters used are $C_m = 200$, $g_L = 12$, $E_L = -70$, $v_T = -50$, $\Delta_T = 2$, $v_r = -58$, $v_{\text{threshold}} = 0$, $\tau_w = 300$, $a = 2$, $b = 60$, $I = 500$.

Original parameters [4]:
The original parameter set from Brette and Gerstner, demonstrating combined adaptation mechanisms. The parameters used are $C_m = 281$, $g_L = 30$, $E_L = -70.6$, $v_T = -50.4$, $\Delta_T = 2$, $v_r = -70.6$, $v_{\text{threshold}} = 20$, $\tau_w = 144$, $a = 4$, $b = 80.5$, $I = 2500$.

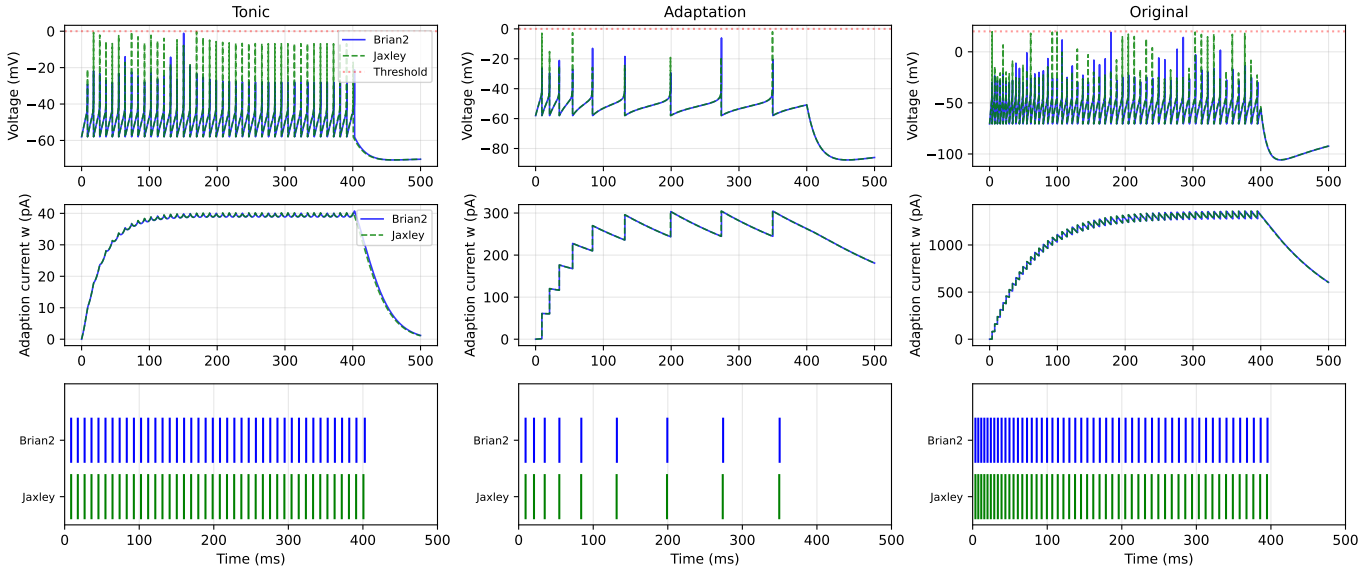

Fig. 1. Left to right: tonic, adaption and original parameters. Tonic and adaption parameters refer to the parameters provided by [2], original parameters are the parameters used in the original AdEx paper [4]. Top row shows the simulated voltage traces. Interesting to note is that when the threshold is reached, the voltage is reset in the same time step. This leads to the voltage never crossing the threshold line in the recordings. Second row displays the adaption current w. Last row compares the spike timings between `Brian2` and `Jaxley`. We expect minor numerical differences, mainly to the fact that different integration methods are used.

3) Comparison Metrics: We assess implementation correctness through:

- **Visual comparison:** Overlay of voltage traces from both simulators (Figure 1).
- **Spike count:** Both implementations produce identical spike counts for all test configurations.
- **Spike timing:** Maximum absolute difference in spike times $\max_i |t_i^{\text{Jaxley}} - t_i^{\text{Brian2}}| < 1$ ms.

D. Parameter Optimisation Evaluation

To evaluate the effectiveness of gradient-based optimisation with surrogate gradients, we fit AdEx model parameters to experimental voltage recordings.

TABLE II
INITIAL ADEX MODEL PARAMETERS AND TRAINABILITY

Parameter	Symbol	Initial Value	Trainable
Membrane capacitance	C_m	200.0 pF	No
Leak conductance	g_L	10.0 nS	Yes
Leak reversal potential	E_L	−68.0 mV	Yes
Spike threshold	V_T	−45.0 mV	Yes
Slope factor	Δ_T	2.0 mV	No
Spike detection threshold	$V_{\text{threshold}}$	−20.0 mV	No
Reset potential	V_{reset}	−55.0 mV	Yes
Adaptation time constant	τ_w	100.0 ms	Yes
Subthreshold adaptation	a	2.0 nS	Yes
Spike-triggered adaptation	b	1.0 pA	Yes

1) Dataset: We use intracellular recordings from striatal projection neurons provided by Johansson and Silberberg [19]. The dataset contains voltage traces from four distinct striatal neuron types, recorded under current-clamp conditions. However, we only used a single trace[1], mainly due to time constraint. This is a serious limitation and will be addressed in future work.

2) Optimisation Setup: We optimise the following subset of AdEx parameters: $\{g_L, E_L, V_T, a, b, \tau_w, v_{\text{reset}}\}$. Parameters $\{C_m, \Delta_T, v_{\text{threshold}}$ were fixed. Initial parameters are given in Table II.

We evaluated two different loss functions: MSE and a differentiable version of a feature based loss developed by Guarino et al. [9], which we will name Guarino loss for the remaining report. Detailed hyperparameters can be found in Table III, however, most parameters are chosen arbitrary. No extensive hyperparameter tuning was conducted. We plan on revising this in future work.

E. Loss Functions

1) MSE: Our initial approach was to use MSE as a loss function. It is defined as

$$\mathcal{L}_{\text{MSE}}(\theta) = \frac{1}{T}\sum_{t=1}^{T}\left(V_{\text{sim}}(t;\theta) - V_{\text{exp}}(t)\right)^2, \qquad (16)$$

where θ denotes the trainable parameters.

We were not able to achieve any sensible results using this loss function. Since other works mainly use feature-based loss functions, we implemented one as well.

2) Feature-Based Loss: Simplified neuron models are usually not required to perfectly imitate the voltage trace of a neuron. Their main requirement is to faithfully replicate the

[1]Files can be found here. Voltage file: `ECBL_IDthresh_ch5_553.dat` Current file: `ECBL_IDthresh_ch4_553.dat`

TABLE III
TRAINING HYPERPARAMETERS FOR ADEX OPTIMIZATION

Parameter	MSE	Guarino
Optimization		
Optimizer	Adam	Adam
Learning rate	0.1	0.1
Gradient clipping (global norm)	2.0	2.0
Number of epochs	500	500
Surrogate Gradient		
Surrogate type	sigmoid	sigmoid
Surrogate slope	25.0	25.0
Loss-Specific Parameters		
Normalize	False	—
Temperature (τ)	—	0.3
Beta (β)	—	10.0
Missing feature penalty	—	3.0
Guarino Feature Weights		
$w_{t_{\text{first}}}$	—	5.0
$w_{t_{\text{second}}}$	—	3.0
$w_{t_{\text{third}}}$	—	2.0
$w_{t_{\text{last}}}$	—	1.0
$w_{\text{inv\_first\_ISI}}$	—	2.0
$w_{\text{inv\_last\_ISI}}$	—	1.0
$w_{\text{firing\_freq}}$	—	1.0
$w_{V_{\text{stim\_end}}}$	—	0.5
Simulation		
Max stimulus duration	500 ms	
Spike threshold	−20 mV	

firing behaviour. Rather than comparing raw voltage traces, Guarino et al. [9] proposed a loss function based on electrophysiological features: To capture the essential characteristics of a neuron, they identified:

- **Spike timing:** time to first (t_1), second (t_2), third (t_3), and last spike (t_{last}),
- **Interspike intervals:** inverse of first ISI ($\frac{1}{\text{ISI}_1}$) and last ISI ($\frac{1}{\text{ISI}_{\text{last}}}$),
- **Firing rate:** mean firing frequency over the stimulus duration,
- **Subthreshold behaviour:** membrane voltage at stimulus end (V_{stimend}).

The loss is computed as a weighted sum of relative errors:

$$\mathcal{L}_{\text{Guarino}} = \sum_i w_i \cdot \frac{|f_i^{\text{sim}} - f_i^{\text{exp}}|}{|f_i^{\text{exp}}| + \epsilon} \tag{17}$$

where f_i^{sim} and f_i^{exp} denote simulated and experimental feature values respectively, and w_i are feature-specific weights. When a feature is present in the experimental data but absent in the simulation (e.g., the experimental trace has three spikes but the simulation produces only one), a fixed penalty is added to encourage the optimizer to produce the correct number of spikes.

This feature-based formulation addresses the spike timing sensitivity problem inherent to MSE: small temporal misalignments no longer produce catastrophic loss values, as long as the overall firing pattern—captured by the features—remains similar.

3) Differentiable Feature-Based Loss: The feature-based loss requires extracting spike times and counts from voltage traces. Standard implementations use hard threshold crossings, which are non-differentiable. To enable gradient-based optimization, we developed soft approximations of the feature extraction operations.

The core idea is to replace discrete spike detection with continuous values. The AdExSurrogate channel outputs a soft spike indicator $s(t) \in [0, 1]$ at each time step. The soft spike count becomes simply $N_{\text{spikes}}^{\text{soft}} = \sum_t s(t)$.

For spike timing, we use cumulative sums to identify when the n-th spike occurs and compute a weighted average over time. Similarly, ISI features are derived from the difference between consecutive soft spike times. Features that require a minimum number of spikes (e.g., ISI needs at least two spikes) are weighted by soft validity flags that smoothly transition based on the spike count.

The soft approximations introduce two hyperparameters: a temperature τ that controls how sharp the spike selection is, and a sigmoid steepness β for the masking operations. We used $\tau = 0.3$ and $\beta = 10.0$ based on initial experiments.

We note that this differentiable feature extraction is a working implementation that has not been thoroughly validated. Due to time constraints, we did not perform extensive testing of the soft approximations against their hard counterparts, nor did we systematically tune the hyperparameters. The approach produces gradients and enables optimization, but the accuracy of the soft features compared to exact values remains to be studied in future work.

4) Evaluation Metrics: We assess optimisation performance through:

- Visual comparison of fitted vs. experimental traces
- Coincidence Factor for Spike Train Evaluation

The **Coincidence Factor for Spike Train Evaluation** Γ measures spike timing prediction quality, normalized by chance level. It is given by

$$\Gamma = \frac{N_{coinc} - \langle N_{coinc} \rangle}{0.5(N_{data} + N_{model})} \cdot \frac{1}{1 - 2f\Delta} \tag{18}$$

where N_{coinc} describe coincidences within $\pm\Delta$ (default 2ms) and f is the model firing rate.

In other words, it describes how good it compares to a random firing pattern. $\Gamma = 1$ means perfect prediction whereas $\Gamma = 0$ is the same as chance level. $\Gamma < 0$ therefore means it behaves worse than chance.

Jolivet et al. achieved a Γ value of $\approx 0.82 - 0.83$ for their well-fitted AdEx models. We use this to compare our fitted parameters.

V. EXPERIMENTAL SETUP

All experiments were conducted on a 2020 Apple M1 Silicon Mac. Software is written in python and executed using python version 3.14.2. The implementation is built on Jaxley version 0.12.0. Comparisons were performed using the Brian2 simulator 2.10.1. Full list of software versions is found in the requirements.txt

TABLE IV
SPIKE TIMING COMPARISON BETWEEN JAXLEY AND BRIAN2 IMPLEMENTATIONS.

	Tonic	Adaptation	Original
Spike count	52	10	64
Max timing diff. (ms)	0.54	0.58	0.82
Mean timing diff. (ms)	0.28	0.27	0.42

VI. RESULTS

A. AdEx verification

Running verification[2] against `Brian2` gave the following results: Figure 1 demonstrates almost perfect overlap. The biggest differences are in spike magnitude. We note that both brian2 and jaxley results do not cross the detection threshold in the recording. This is an artefact due to how we record the simulated potentials. The integration step is computed and then the new voltage value is stored. When the integrated value crosses the threshold, the voltages are reset before the value is recorded, meaning in this all voltage recordings are below $v_{\text{threshold}}$.

We measured the difference between spike timings. Table IV shows the measured results. We identify over different spiking behaviour, both simulations produced identical amount of spikes. The measured maximum timing differences between spikes was < 1ms for all runs, the mean difference between spikes was $< 0.5ms$. Given the sensitivity of the exponential spiking initiation zone and the fact that different numerical integration was used for the adaption current, we were satisfied with the similarity of the results and concluded that the implementation is correct.

B. Surrogate Gradient verification

To verify the surrogate gradient implementation we first computed the gradient of a simple loss function (number of spikes) after stimulating the cell. The produced gradients are zero for the default implementation and non-zero for the surrogate model. We visualized (see Figure 2) the backward pass to verify compare different surrogate types (Sigmoid, Exponential, SuperSpike) — all peak around threshold of the heavyside-function.

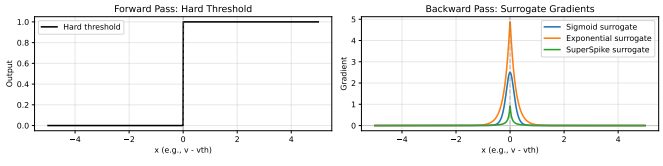

Fig. 2. Left: heavyside function in the forward pass. Right: different surrogate gradients, all centred around the threshold.

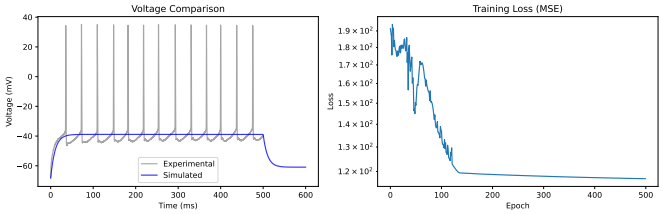

Fig. 3. Left: simulated voltage using parameters trained using MSE loss. Right: loss over training epoch, it converges into non spiking behaviour, trying to minimize sub-threshold loss

C. Parameter Estimation using MSE

The natural first approach for parameter estimation is to minimize the MSE16 between simulated and experimental voltage traces. However, our experiments indicated that this loss function is fundamentally unsuitable for spiking neurons. A core problem is *spike timing sensitivity*: even when simulated and experimental traces exhibit similar firing patterns, small temporal misalignments between spikes create disproportionately large MSE values. Consider two traces with identical spike counts and frequencies, but with spikes offset by 2 ms. During each spike, the voltage difference can exceed 100 mV (from resting potential to spike peak), resulting in enormous per-sample errors that dominate the loss. This timing sensitivity manifests in a highly non-convex loss landscape. The gradient at any point reflects primarily the instantaneous voltage mismatch rather than structural similarity of the spike trains. Consequently, gradient descent steps often move parameters in directions that reduce voltage error at specific time points while disrupting the overall firing pattern.

Additionally, perfect spike time matchings are punished. This is because AdEx values are reset before the jaxley simulator records the voltage trace — if experimental data and simulation data would spike at the exact same time, the loss would be huge (AdEx is at V_r, experiment is at V_{peak}). This enforces a mismatch between spike timing and simulation.

We attempted MSE-based optimization using the same experimental recordings and training setup described in Section IV-D. The optimizer converged to parameter configurations that minimized voltage deviation during subthreshold periods but failed to produce correct spiking behaviour. In many runs, the model either stopped spiking entirely (eliminating spike-related MSE contributions) or produced spike patterns uncorrelated with the experimental data.

This leads us to believe that MSE loss, while theoretically sound for continuous signals, is ill-suited for discrete event (spike) matching. The loss function must capture *what* features matter rather than demanding exact sample-by-sample voltage alignment.

D. Parameter Estimation using Feature-Based Loss

Since spiking timings are the most important aspect if a simplified neuron model, using a feature-based loss was

[2]Test can be run in Google Colab

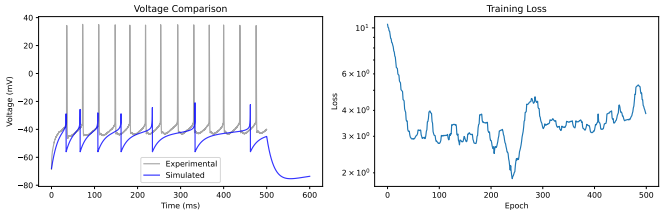

Fig. 4. Left: simulated voltage using parameters trained using guarino loss (best performing parameters, lowest loss). Even though parameters are not well-fitted enough to faithfully reproduce neuron behaviour, it shows better approximation behaviour than MSE loss altogether. Further improvement and verification needed to confidently evaluate this approach. Right: loss over training epoch. Loss does not decrease monotonically, indicating non-convex parameter space.

an attempt to shift the training behaviour in this direction. The training setup of the feature-based optimization is also described in Section IV-D. First experiments indicated that trained models did in fact better capture overall behaviour of the underlying experiments. For example, the number of spikes of the fitted model often matched (or got close) the number of spikes of the underlying experimental data. Spike timing of the first spike were consistently closer than in MSE trained models (or in arbitrary parameter constellations).

Training loss in different training runs did not decrease monotonically, as can be seen in Figure 4. This indicated to us a non-convex parameter space, at least in regards to this specific loss function. Training was also very susceptible to the starting parameters. Often parameters did not change drastically in result of the training, indicating that computed gradients were not able to leave local minima. However, as mentioned in Section IV-E3, we want to clarify that this part of the work has not been thoroughly tested and verified. A lot more quantification and testing has to be done for more reliable evaluation of this loss function.

E. Coincidence Factor

To evaluate the performance of the two loss functions, we measured spike timing prediction quality using the coincidence factor Γ from Jolivet et al. [16], with a coincidence window $\Delta = 2\,\text{ms}$. This means that according to Jolivet's metric, both fitted model perform equally good as firing at chance.

We attribute this to the following reasons:

- **Non-convex parameter-space:** The AdEx model has many interdependent parameters. This results in a non-convex optimization landscape with numerous local minima. Gradient-based optimization methods can easily become trapped in suboptimal regions that produce incorrect spike timing despite achieving low training loss.
- **Limited training data:** The models were trained on a single experimental trace with one stimulus condition. We don't think overfitting was a particular problem, but future work might want to build a more robust training system.
- **Surrogate gradients:** The use of a surrogate for the discontinuous spike mechanism may change the model dynamics and could have introduced bias to the optimization.
- **More careful loss function design:** Due to time constraints, there was no tweaking of feature weights etc. in the feature-based loss function. A lot more careful loss function tuning could greatly improve the results of the training.
- **Skewed voltage representation:** The recorded voltages in the simulation don't capture the true spike of the action potential. Especially non-feature based loss functions (e.g. MSE) are inherently limited in how well they can reproduce spiking behaviour.

Future work could address these limitations by carefully adjusting spike-timing-aware loss functions, training on multiple stimulus conditions or changing the recording behaviour of the simulation.

VII. Discussion and Conclusion

We successfully implemented the AdEx model in Jaxley and enabled automatic differentiation through surrogate gradients. We discovered that MSE is fundamentally unsuitable for training spiking behaviour. Feature-based losses show more promising optimization results, however, differentiability of the loss function as a fundamental requirement makes this a non-trivial problem to tackle. First evaluation showed that spike timing precision remains challenging, however it is a crucial requirement for useful bio-physical neuron simulations.

Loss function design remains a challenging problem and is critical when bridging SNN techniques to biophysical fitting. Therefore, SNN techniques can't directly be transferred into the neuroscience domain, it requires more domain-specific adaptations.

The main limitation of this work include that no accurate parameters could be produced. We only tested on a single neuron type and recording, making this work non-generalizable. Also, runtime- and convergence performance is not addressed because useful parameter results were the primary concern.

Future work can address all these issues by optimizing the loss functions, can increase the neuron types and training setup as well as introduce a more thorough comparison with other state-of-the-art genetic / feature-based algorithms. A thorough evaluation and comparison of different surrogate gradients is still missing, which in itself plays a big role in how the parameters are adjusted based on the computed loss.

We bridged two worlds: Gradient optimization methods using surrogate gradients and parameter fitting for simplified biophysical models. We discovered that loss function design matters more than the optimization algorithm and were able to produce a testing framework that showed exactly this. The main contribution was to implement a differentiable AdEx model in Jaxley and gained an understanding of loss function requirements.

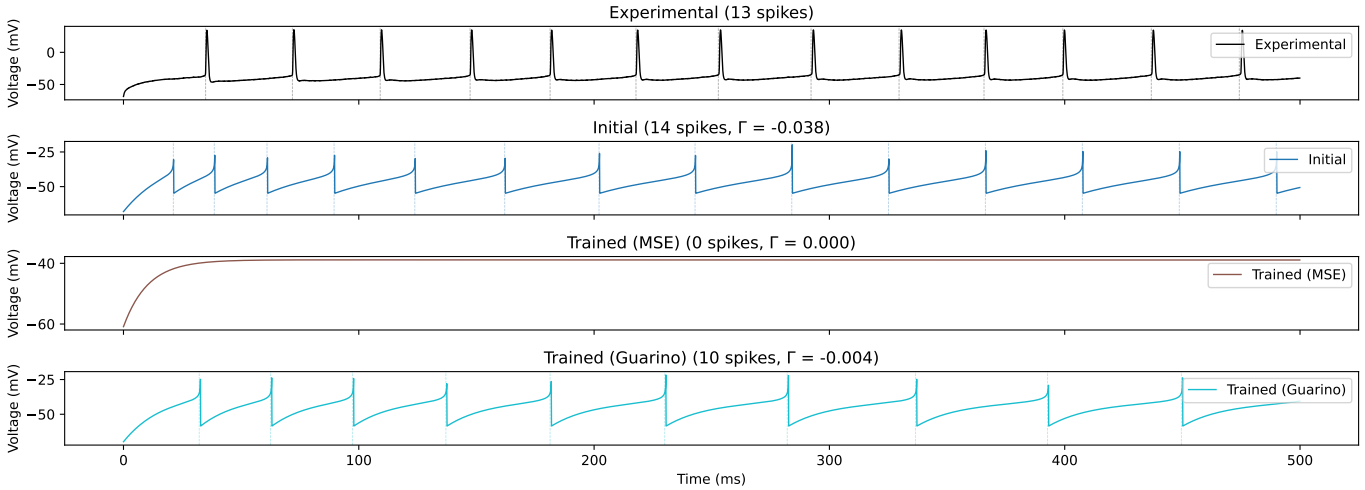

Fig. 5. Top row: experimental data. Shows voltage trace of striatal projection neuron in a current-clamp experiment. Neuron performs thirteen approx. equidistant spikes. Second row: simulation using arbitrary default parameters pre training. Third row: voltage trace of trained model using MSE-based loss. Bottom row: voltage trace of feature-based guarino loss. Approximates spike timings of first three spikes more precisely than default parameters, but fails to recreate correct number of spikes in this instance.

Acknowledgment

The author wants to thank Alexander Kozlov for suggesting this topic and providing valuable insight and resources. I also want to thank Johan Eklund for his supervising work.

AI assistance disclosure

AI assistance was used for this project. I used Anthropic's Opus 4.5 via Claude Code. Usage mainly included writing feedback, code refactoring, documentation of code and plotting scripts. For example, I used claude to rewrite my evaluation files into a coherent python module (see this prompt). It also included transcribing lists of values into tables (e.g. Table I) from code or text files. Transcripts of prompts can be found here. All results were validated and double checked by hand.

References

[1] W. Gerstner, Ed., *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. S.l.: Cambridge Univ. Press, 2014.

[2] R. Naud, N. Marcille, C. Clopath, and W. Gerstner, "Firing patterns in the adaptive exponential integrate-and-fire model," *Biological Cybernetics*, vol. 99, no. 4-5, Nov. 2008.

[3] F. Tesler, A. Kozlov, S. Grillner, and A. Destexhe, "A multiscale model of striatum microcircuit dynamics," Aug. 2024.

[4] R. Brette and W. Gerstner, "Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity," *Journal of Neurophysiology*, vol. 94, no. 5, Nov. 2005.

[5] M. Deistler, K. L. Kadhim, M. Pals, J. Beck, Z. Huang, M. Gloeckler, J. K. Lappalainen, C. Schröder, P. Berens, P. J. Gonçalves, and J. H. Macke, "Jaxley: Differentiable simulation enables large-scale training of detailed biophysical models of neural dynamics," *Nature Methods*, Nov. 2025.

[6] W. Van Geit, E. De Schutter, and P. Achard, "Automated neuron model optimization techniques: A review," *Biological Cybernetics*, vol. 99, no. 4-5, Nov. 2008.

[7] I. S. Jones and K. P. Kording, "Efficient optimization of ODE neuron models using gradient descent," 2024.

[8] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate Gradient Learning in Spiking Neural Networks," May 2019.

[9] D. Guarino, I. Carannante, and A. Destexhe, "A unified model library maps how neuromodulation reshapes the excitability landscape of neurons across the brain," *PLOS Computational Biology*, vol. 21, no. 12, Dec. 2025.

[10] M. Marín, M. J. Sáez-Lara, E. Ros, and J. A. Garrido, "Optimization of Efficient Neuron Models With Realistic Firing Dynamics. The Case of the Cerebellar Granule Cell," *Frontiers in Cellular Neuroscience*, vol. 14, Jul. 2020.

[11] N. C. Cruz, M. Marín, J. L. Redondo, E. M. Ortigosa, and P. M. Ortigosa, "A Comparative Study of Stochastic Optimizers for Fitting Neuron Models. Application to the Cerebellar Granule Cell," *Informatica*, 2021.

[12] L. Hertäg, J. Haß, T. Golovko, and D. Durstewitz, "An analytical approximation to the AdEx neuron model allows fast fitting to physiological data," *BMC Neuroscience*, vol. 12, no. S1, Dec. 2011.

[13] R. C. Gerum and A. Schilling, "Integration of leaky-integrate-and-fire neurons in standard machine learning architectures to generate hybrid networks: A surrogate gradient approach," *Neural Computation*, vol. 33, no. 10, Sep. 2021.

[14] N. Perez-Nieves, V. C. H. Leung, P. L. Dragotti, and D. F. M. Goodman, "Neural heterogeneity promotes robust learning," *Nature Communications*, vol. 12, no. 1, Oct. 2021.

[15] J. Gygax and F. Zenke, "Elucidating the Theoretical Underpinnings of Surrogate Gradient Learning in Spiking Neural Networks," *Neural Computation*, vol. 37, no. 5, Apr. 2025.

[16] R. Jolivet, R. Kobayashi, A. Rauch, R. Naud, S. Shinomoto, and W. Gerstner, "A benchmark test for a quantitative assessment of simple neuron models," *Journal of Neuroscience Methods*, vol. 169, no. 2, Apr. 2008.

[17] F. Zenke and S. Ganguli, "SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks," *Neural Computation*, vol. 30, no. 6, Jun. 2018.

[18] M. Stimberg, R. Brette, and D. F. Goodman, "Brian 2, an intuitive and efficient neural simulator," *eLife*, vol. 8, Aug. 2019.

[19] Y. Johansson and G. Silberberg, "The Functional Organization of Cortical and Thalamic Inputs onto Five Types of Striatal Neurons Is Determined by Source and Target Cell Identities," *Cell Reports*, vol. 30, no. 4, Jan. 2020.