# Spike-Triggered Descent

Michael Kummer and Arunava Banerjee

Computer and Information Science and Engineering, University of Florida,

PO Box 116120, Gainesville, FL 32611, USA

E-mail: msk@cise.ufl.edu, arunava@ufl.edu

**Abstract**

The characterization of neural responses to sensory stimuli is a central problem in neuroscience. Spike-triggered average (STA), an influential technique, has been used to extract optimal linear kernels in a variety of animal subjects. However, when the model assumptions are not met, it can lead to misleading and imprecise results. We introduce a technique, called spike-triggered descent (STD), which can be used alone or in conjunction with STA to increase precision and yield success in scenarios where STA fails. STD works by simulating a model neuron that learns to reproduce the observed spike train. Learning is achieved via parameter optimization that relies on a metric induced on the space of spike trains modeled as a novel inner product space. This technique can precisely learn higher order kernels using limited data. Kernels extracted from a *Locusta migratoria* tympanal nerve dataset[1] demonstrate the strength of this approach.

# Introduction

A major goal of sensory neuroscience is to precisely characterize the mapping that specifies how a neuron responds to sensory stimuli. This response function accounts for intermediary physical processes along with activity of the entire network upstream from the target neuron. The complexity of this problem arises from variations in network connectivity and constituent ion channels which cause wildly differing behavior. There's additional difficulty in a direct component wise analysis resulting from discontinuity of the spiking behavior caused by Hodgkin Huxley ion channels [2]. The inaccessible and innumerable physical parameters create complex interactions which lead to intractable calculations necessitating model simplifications.

A key simplifying assumption comes from signal processing: Any time-invariant continuous nonlinear operator with fading memory can be approximated by a Volterra series operator [3]. The overall impact upon the membrane potential by the upstream network can then be described by a set of Volterra Weiner kernels [4] [5] [6]. Using the first order kernel: Spike-triggered average (STA), a technique which has seen widespread application, assumes a simple probabilistic model of spike generation. When the model assumptions are met, it returns an optimal first order kernel. We introduce a new technique, called spike-triggered descent (STD), which can learn higher order kernels and yield higher accuracy. These techniques approximate the desired kernel by constructing a relationship between sensory stimuli and the spike trains they cause.

Here we give an overview of how STA [7] [8] [9] [10] and STD work, describe their models, and point out a few key differences. STA is based on the linear non-linear Poisson cascade model (LNP) which convolves the signal with a linear kernel, applies a point nonlinearity to convert it into a firing rate, and then samples it using an inhomogeneous Poisson point process to generate spikes. Obtaining the kernel from the signal and resulting spikes requires the signal

to be a stationary Gaussian process so that Bussgang's theorem [11] can be applied. In contrast to STA, STD is based on the cumulative spike response model (CSRM) [12] and can approximate higher order kernels for any sufficiently complex signal. Replacing the nonlinear Poisson spike generation, the CSRM spikes occur when the convolution's resulting membrane potential exceeds threshold which inhibits future spikes by way of an after hyperpolarizing potential (AHP). STD works by comparing simulated and recorded spike trains to form a gradient that optimizes kernel parameters.

Neuroscientists use STA because it recovers the optimal linear kernel and is easy to use. It has been applied in a variety of applications including creating bionic interfaces[13]. However, the reliance on the stimulus being a Gaussian process and the restriction to first order kernels are weaknesses which STD does not share. The techniques can be used in tandem or STD's initial learning kernel can be randomly guessed. Surprisingly without over fitting, it can precisely approximate while reusing limited data. In the following sections, we introduce a spike train metric used in momentum based stochastic gradient descent (SGD) to update kernels that represent response functions.

# Results

Spike-triggered descent is a robust technique that can precisely approximate linear and higher order kernels along with an AHP time constant. After showing a case where STD excels beyond STA, we indicate why. Introduced next is a generalization of the cumulative spike response model (GCSRM). At the core of STD is the ability to minimize the distance (3) between the desired ($D$, experimentally measured) and output ($O$, simulated reconstruction) spike trains with respect to kernel parameters. As the simulated neuron's spike trains approach the desired, so too do its parameters. Convergence is demonstrated in a variety of experimental setups: first and second order kernels, first order kernels with accompanied AHP time constants, and

first order kernels from LNP spikes. Additionally demonstrated is the robustness against noise effects applied to the input, spike times, and spike addition/deletion. Finally, we demonstrate the effectiveness by applying it to a *Locusta migratora* dataset.

## STD compared to STA

Shown in Figure 1 are results from an experimental setup where STD achieves superior accuracy as compared to STA in two distinct experiments (before and after distortion). The input signal to this virtual neuron was uniform white noise sampled at 1kHz distorted by a second order kernel. The stimulus drove a neuron which had a first order kernel and spiked when the convolved signal crossed a fixed threshold where each spike had an impact according to an exponential decay function describing an AHP. The STA kernel yields an error of $50\%$ (cyan) as defined by $\frac{|k_{des}-k_{lrn}|}{|k_{des}|}$ where $|\cdot|$ is the $L_2$ norm.

Adding signal distortion via a second order kernel creates a signal (1a) that results in STA having high frequency oscillation about the origin with an error of $119\%$ (red) which when smoothed had an error of $98\%$ (magenta). For comparison purposes, instead of randomly initializing, STD is using the scaled and smoothed STA result (magenta) as a starting place for gradient updates until convergence. This proceeds by randomly selecting one of the $100$ input slices (1a) $10,000$ times and eventually yields an $8\%$ error (orange). To reiterate, STD achieved superior results while retraining on the same data and not over fitting! When trying to obtain a close approximation, to avoid compounding errors in complex analyses, it would be favorable to use STD.

## Why STA requires Gaussian input

A graphical summary of how the LNP model operates is shown in Figure 2. The linear component is the first order Volterra kernel $k$, the nonlinear component (N) is a function $v$ that maps

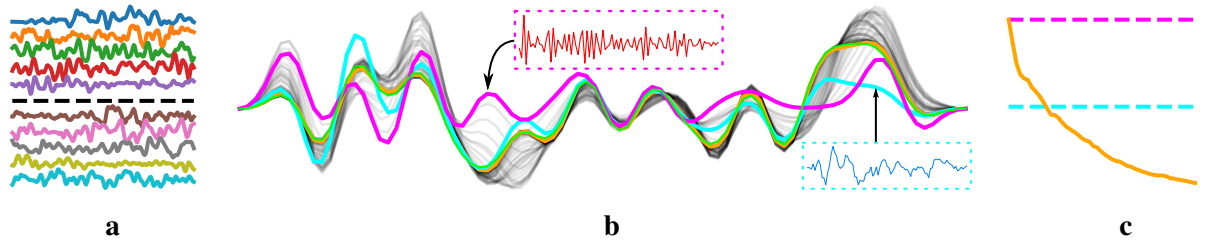



**Fig. 1.** **a**, The input is distorted by passing it through a second order kernel modeled by a grid made of $20 \times 20$ third order cardinal B-splines. This produces a new signal that is used instead as input $s = \int \int Kx(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2$. The first and last $5$ of $100$ input slices of $200ms$ are shown sequentially. **b**, The distorted input leads to a high frequency STA kernel (red) which is then smoothed (magenta). STD proceeds from here by iterating through a chain of intermediary learning kernels (black) to approximate (orange) the desired (green). The undistorted input's result before (blue) and after (cyan) smoothing show the effect the distortion has on STA. **c**, The error, as measured by $\frac{|k_{des}-k_{lrn}|}{|k_{des}|}$, decreases to $8\%$ (orange) as the STD kernel approaches the desired. Smoothing and scaling the undistorted and distorted STA results improves them from $58\%$ (blue) to $50\%$ (cyan) and from $119\%$ (red) to $98\%$ (magenta).

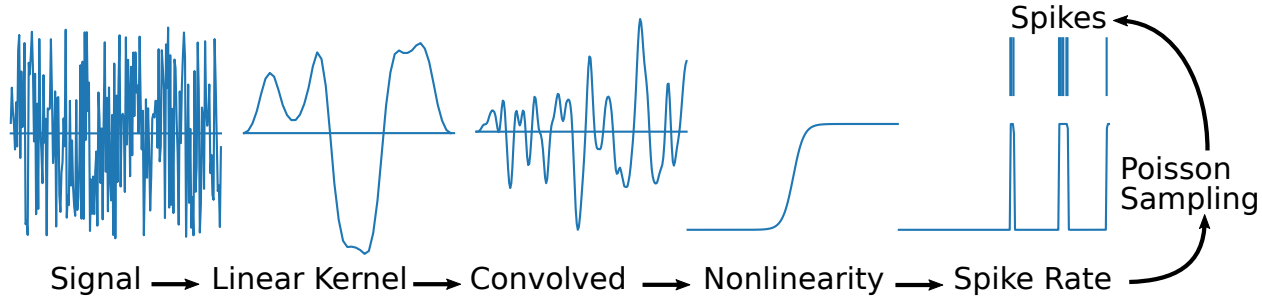



**Fig. 2.** A signal is convolved with a linear kernel (L), passed through a nonlinearity (N) to achieve a spike rate, and then Poisson sampled (P) to produce spikes.

the kernel response $y = k * x_w$ to a firing rate $z$, and an inhomogeneous Poisson (P) process generates spikes. At any point in time, the probability of firing is $z = v(k * x_w)$. The solution to the STA kernel can be expressed as $k = \phi_{xx}^{-1}\phi_{xy}$ where $\phi_{xx}$ is the signal's autocorrelation and $\phi_{xy}$ is the correlation of the signal with the potential. Instead of $\phi_{xy}$, we can instead only observe a correlation with the spike rate $\phi_{xz}$. Bussgang[11] showed $\phi_{xy} = C\phi_{xz}$ the resulting kernel differs only by a scaling factor after the nonlinearity is applied, when the input is a Gaussian process.

## Generalized cumulative spike response model

The cumulative spike response model (CSRM) [12] illustrated in Figure 3 can be generalized to include the full Volterra series of kernels (1). These kernels, approximated here by splines, represent the impact on the membrane potential by the stimuli's higher order autocorrelates. Increasing $n$ allows for increased pattern detection capabilities. The AHP function $\eta = -Ae^{(t_l - t_k)/\mu}$ models the refractory period which is a region where spikes are highly unlikely to occur right after firing. In contrast to Poisson sampling, the AHP approach is deterministic and imposes a prior state dependency. A spike is generated when a threshold ($\tilde{\Theta}$) is exceeded by the signal convolved with kernel(s) minus past spikes' AHPs. The n-order kernel $K_n$ is a spline function composed of the n-ary Cartesian product of third order cardinal B-splines $B_n$. The kernel is incrementally updated by the optimization process. At time $t = t_l^O$ the kernel is $K_{l,n} = \sum_i B_{i,l,n}\beta_{i,l,n}$. The current and prior spike times are $t_l^O$ and $t_k^O$. The AHP parameter $\mu$ modifies the refractory time.

$$\tilde{\Theta} = \sum_{n=1}^{\infty} \int \dots \int K_{l,n}(\tau_1 \dots \tau_n; \beta_{i,l,n}) \prod_i^n x(t_l^O - \tau_i) d\tau_i + \sum_k \eta(t_l^O - t_k^O; \mu) \tag{1}$$

## Theoretical overview of STD

Spike-triggered descent updates parameters based off of the distance between simulated and desired spike trains. To support using the distance (4) from [14] on the GCSRM, it is important to also generalize spike trains and show that they're a subset of a vector space with an inner product $\langle \cdot, \cdot \rangle$. Considering augmented spike trains with countably infinite ($\mathbb{N} = \{1, 2, ...\}$) spikes gives the framework the versatility to compare spike trains of any length. The augmentation turns spike trains into tuples of times and coefficients $\boldsymbol{t} = \{(t_i, \alpha_i)\}$. This forms a vector space and provides the foundation for creating an inner product (2) that induces a metric (Methods).

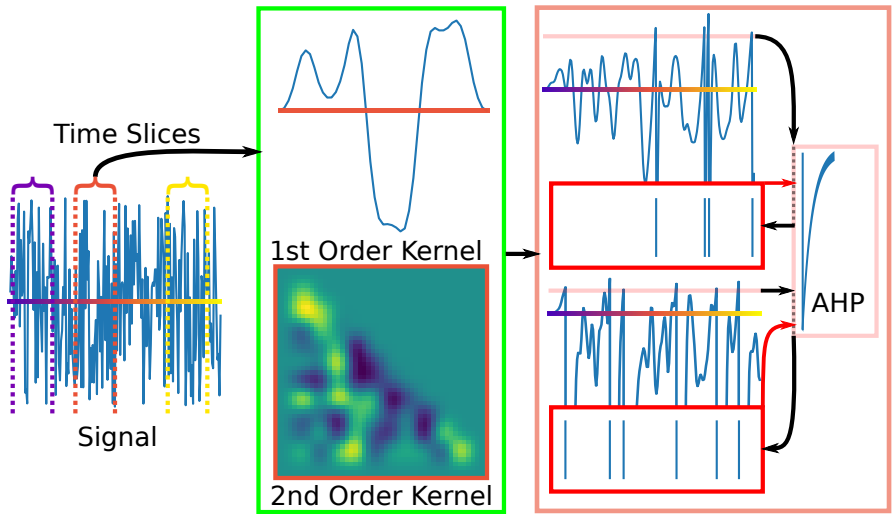



**Fig. 3.** Slices of the uniformly random generated signal are serially passed through a kernel to produce a simulated electrical potential. When above threshold (pink) an AHP, with time constant $\mu$ generally set to $1.2ms$, is applied and a spike is produced which collectively (red box) inhibit future firing.

Setting the $\alpha's$ to 1 for finitely many spikes reduces to the usual space of spike trains within a bounded past. This is a subset of the generalization with the same metric and is squared to simplify the algebra (3).

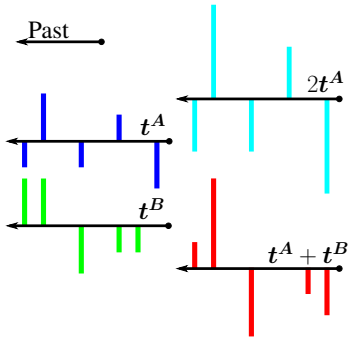
$$\langle \boldsymbol{t^A}, \boldsymbol{t^B} \rangle = \sum_{i,j=1}^{\infty} (\alpha_i^A \times \alpha_j^B) \frac{t_i^A \times t_j^B}{(t_i^A + t_j^B)^2} e^{-\frac{t_i^A + t_j^B}{\tau}} \tag{2}$$

$$E = d^2 = \langle \boldsymbol{t^D} - \boldsymbol{t^O}, \boldsymbol{t^D} - \boldsymbol{t^O} \rangle \tag{3}$$

Setting $\alpha$ values to 1 for $N, M$ spikes, and 0 otherwise, we get:

$$
\begin{aligned}
E(\boldsymbol{t^D}, \boldsymbol{t^O}) = &\sum_{i,j=1}^{M,M} \frac{t_i^D \times t_j^D}{(t_i^D + t_j^D)^2} e^{-\frac{t_i^D + t_j^D}{\tau}} \\
&+ \sum_{i,j=1}^{N,N} \frac{t_i^O \times t_j^O}{(t_i^O + t_j^O)^2} e^{-\frac{t_i^O + t_j^O}{\tau}} \\
&- 2 \sum_{i,j=1}^{M,N} \frac{t_i^D \times t_j^O}{(t_i^D + t_j^O)^2} e^{-\frac{t_i^D + t_j^O}{\tau}}
\end{aligned}
\tag{4}
$$

A diagram of the generalized version of spikes is shown in Figure 4a. It illustrates scalar multiplication (cyan) and vector addition (red). The vectors point positively into the past from the current time at zero. Multiplying the vector $\boldsymbol{t^A}$ by 2 has the effect of doubling the coefficient for each time (cyan). When vectors $\boldsymbol{t^A}$ and $\boldsymbol{t^B}$ are added, equal times cause their coefficients to be added, else concatenated. Coefficients which sum to zero cause the spike to be deleted.




**a**

$$
\begin{pmatrix}
\frac{Dt_1}{\partial\beta_{i,1}} & \frac{Dt_2}{\partial\beta_{i,2}} & \frac{Dt_3}{\partial\beta_{i,3}} & \frac{Dt_4}{\partial\beta_{i,4}} = \frac{\partial t_4}{\partial\beta_{i,4}} \\
0 & \frac{Dt_2}{\partial\beta_{i,1}} & \frac{Dt_3}{\partial\beta_{i,2}} & \frac{Dt_4}{\partial\beta_{i,3}} = \frac{Dt_3}{\partial\beta_{i,3}} \frac{\partial t_4}{\partial t_3} \\
0 & 0 & \frac{Dt_3}{\partial\beta_{i,1}} & \frac{Dt_4}{\partial\beta_{i,2}} = \frac{Dt_3}{\partial\beta_{i,2}} \frac{\partial t_4}{\partial t_3} + \frac{Dt_2}{\partial\beta_{i,2}} \frac{\partial t_4}{\partial t_2} \\
0 & 0 & 0 & \frac{Dt_4}{\partial\beta_{i,1}} = \frac{Dt_3}{\partial\beta_{i,1}} \frac{\partial t_4}{\partial t_3} + \frac{Dt_2}{\partial\beta_{i,1}} \frac{\partial t_4}{\partial t_2} + \frac{Dt_1}{\partial\beta_{i,1}} \frac{\partial t_4}{\partial t_1}
\end{pmatrix}
$$

**b**

**Fig. 4.** **a**, Two example spike trains $\boldsymbol{t^A}$ (blue) and $\boldsymbol{t^B}$ (green) under scalar multiplication $2\boldsymbol{t^A}$ (cyan) and vector addition $\boldsymbol{t^A} + \boldsymbol{t^B}$ (red). **b**, Example vectorization/memoization [15] for calculating the total derivatives of time changes with respect to changes in $\beta$ (8) for spline $i$ at each prior spike.

The GCSRM (1) describes when the threshold is equal to the sum of the convolution(s) minus past AHP effects. Perturbing its parameters, performing a first order Taylor approximation, and setting the perturbed and unperturbed expressions equal allows for the creation of partial derivatives with respect to the parameters $\beta$ (5), $\mu$ (6), and $t_k$ (7) (Methods). These express

how spike times change with respect to a change in spline coefficient, time constant, and prior spike time. These partial derivatives are used to calculate the total derivatives (8), an example for $4$ spikes is shown in 4b. The matrix for $\mu$ is similar except that $\mu$ only effects future spikes meaning $\frac{\partial t_l}{\partial \mu_l} = 0$. The total derivative for a spike time with respect to a parameter at a prior time is the sum of that parameter's total effect on each of the more recent prior spikes each multiplied by the relevant time partial to tie it to the current time.

$$\frac{\partial t_l^O}{\partial \beta_{i,l}} = \frac{-\int B_i(\tau) x(t_l^O - \tau) d\tau}{\int_0^{|K|} K(\tau; \beta_{i,l}) \frac{\partial x}{\partial t}\Big|_{t_l^O - \tau} d\tau + \sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O}} \tag{5}$$

$$\frac{\partial t_l^O}{\partial \mu_l} = \frac{-\frac{\partial \eta}{\partial \mu}\Big|_{t_l^O - t_k^O}}{\int_0^{|K|} K(\tau; \beta_{i,l}) \frac{\partial x}{\partial t}\Big|_{t_l^O - \tau} d\tau + \sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O}} \tag{6}$$

$$\frac{\partial t_l^O}{\partial t_k^O} = \frac{\frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O}}{\int_0^{|K|} K(\tau; \beta_{i,l}) \frac{\partial x}{\partial t}\Big|_{t_l^O - \tau} d\tau + \sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O}} \tag{7}$$

$$\frac{D t_{k+1}^O}{\partial \beta_{i,l}} = \sum_{t_j > t_l}^{k} \frac{D t_j^O}{\partial \beta_{i,l}} \frac{\partial t_{k+1}^O}{\partial t_j^O} \tag{8}$$

The total derivative expressions for each of the spikes with respect to a parameter are then connected via another chain rule to the squared distance E. This is performed by summing over the multiple of $\frac{\partial E}{\partial t_k^O}$ along a (colored) diagonal in 4b. The sum of all of these changes in E with respect to that parameter across all time steps is the gradient. The equations (9) and (10) are the gradient updates used to minimize the spike train distance with respect to the spline parameters $\beta$ and AHP time constant $\mu$. Using these gradients, small steps (with size $\alpha$) are taken in accordance with sgd methods such as momentum [16] and katyusha [17]. The updates are best triggered soon after spikes since E exponentially decays as spikes are pushed further into the past.

$$\beta_i = \beta_i - \alpha_\beta \sum_{k \in F_l} \frac{\partial E}{\partial \beta_{i,l}} \tag{9}$$

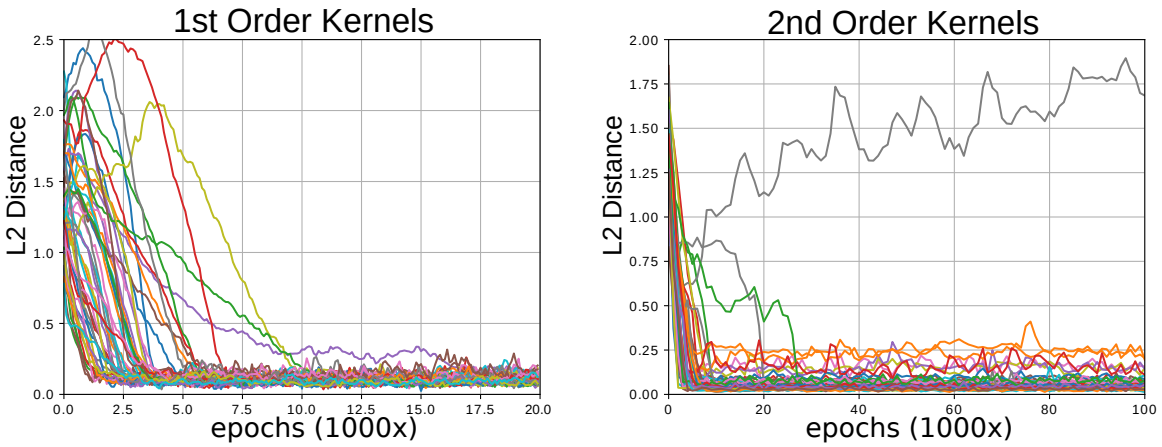
$$\mu = \mu - \alpha_\mu \sum_{k \in F_l} \frac{\partial E}{\partial \mu_k} \tag{10}$$

## Convergence of $\beta$ and $\mu$

Convergence of first and second order kernels on simulated data are shown in Figure 5. While holding $\mu$ fixed, the convergence for 50 trials of 1st (5a) and 2nd (5b) order kernels are demonstrated. An error, or $L_2$ distance, of zero means the learning and desired kernels are equal. A variety of SGD methods were used: vanilla was too slow, katyusha was too unstable, and momentum was fast and relatively robust.

One simulation, seen in (5b), does not converge. These cases can arise for a variety of reasons and is often the result of hyperparameters: momentum, learning rate, cap, AHP threshold, and number of epochs. Lower learning rates and momentum converge more stably but require more iterations. For 2nd order kernels, a mask is applied to keep it lower triangular. This was used as a means of preventing diagonally symmetric terms from conflicting. The cap, a crucial parameter due to the highly nonlinear nature of the problem, is an upper bound on the $L_2$ norm of the gradient step preventing unnecessarily large updates which lead to divergence.

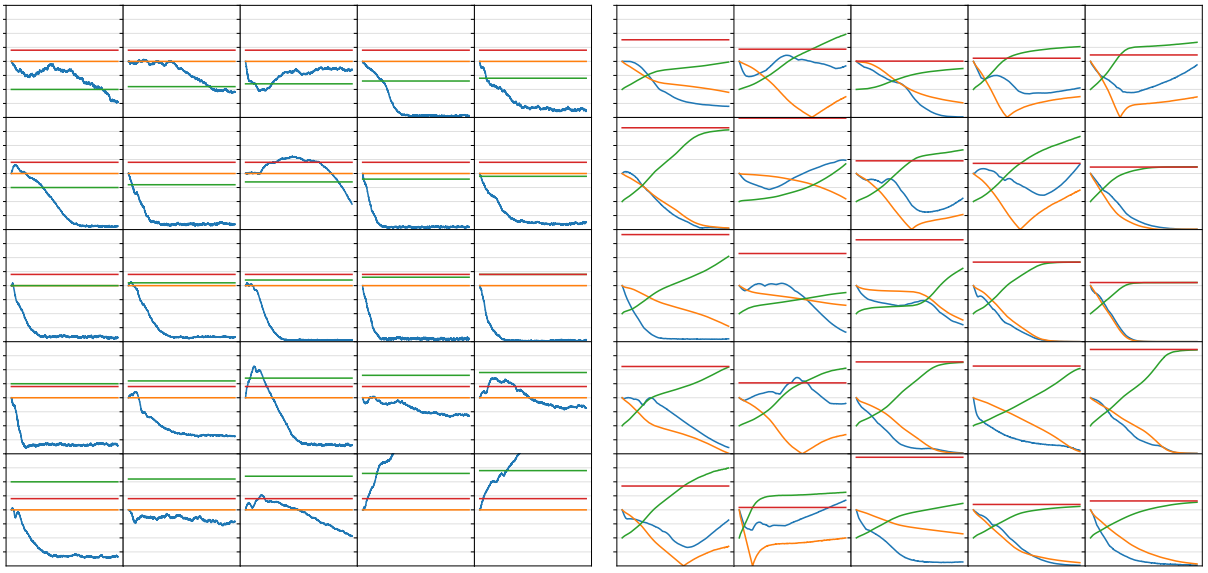
When the desired kernel is shorter than the learning kernel, the extra values become zero as would be expected. Convergence was best for these third order overlapping cardinal B-splines with $12$ abstract time units of nonzero support. Most tests were performed on $10$ splines. More splines took more training time and this was compounded by increasing the kernel's order. The computational and space requirements of higher order kernels restrict consideration to fewer parameters.




**a** **b**

**Fig. 5.** **a**,**b**, Convergence for $50$ trials of 1st order kernels composed of $10$ splines and 2nd order kernels composed of a grid of $8 \times 8$ splines. Each epoch represents an update and consists of a signal of $400$ time units set as $1ms$ meant to be consistent with a statically set AHP time constant $\mu = 1.2ms$.

In Figure 6a, the learning $\mu$ (green) is set incorrectly from the desired $\mu$ (red) and scanned for its effect on the convergence of the $\beta$ parameters. There is a larger region of stability below the desired $\mu$. The $\mu$ (orange) and $\beta$ (blue) errors are normalized with respect to their initial distance from the desired. Demonstrated in 6b, $\mu$ and $\beta$ are being learned in tandem. Once the learning $\mu$ grows larger than the desired, the simulations begin to diverge. This is because $\mu$'s effect is exponential and $\beta$'s splines are linear. They cannot compensate for this error and this causes a feedback loop. Seeing as there was a large range of stability it is something that can easily be scanned. We also tried using a spline representation for the AHP, but getting the parameters to match required spikes to overlap with the spline. Some splines, particularly immediate after a firing, were the least likely to occur but also had the greatest impact. Other basis functions might be an option.



**a** **b**

**Fig. 6.** **a**, $\mu$ (green) is scanned from $0.50$ (top left) to $1.7$ (bottom right) in steps of $0.05$. For a desired $\mu = 1.2$ (red) a range for accurate convergence for $\beta$ (blue) corresponds to $[0.65, 1.35]$ or alternatively $[54\%, 113\%]$ of the desired. **b**, Convergences are shown for $\mu$ (orange) and $\beta$ (blue) being simultaneously learned and plotted as ratios from their original $L_2$ distance.

## Sensitivity to Noise: shifting addition and deletion of spikes

After giving STA difficulty with a complex AHP scenario, it was important to similarly challenge the capability of STD by learning from LNP spikes. This exploration was initiated by analyzing the effect on convergence by the shifting, adding, or deleting of spikes. Upon perturbing the spike times by some fixed amount ($dt$) it quickly became apparent that more granularity was needed in the AHP spike generation process itself. This lead to calculating a more precise intersection of the potential with the threshold and was achieved by linear interpolation. Figure 7 demonstrates the effects of adding noise to the CSRM system and the effect it has on convergence and cosine similarity defined by $\frac{a \cdot b}{||a||\ ||b||}$ which is chosen for its invariance to scaling.

Moving far away from $dt = 0$ resulted in divergence and indicated that the learning kernel needed flexibility. Padding both the learning and desired kernels with splines having $\beta = 0$ allowed the learning kernel to appropriately shift (7a). Removing spikes (7c) randomly with some probability, normalized with respect to the number of spikes, had minimal effect on convergence until the deletion chance was $50\%$. Real neurons are highly connected and auxiliary processing could cause additional seemingly random spikes. Adding spikes to any location (7d) shows that learning is comparatively more sensitive to addition.

STD's success on spikes generated via an LNP process is explored in Figure 8. For the LNP model, a high exponential coefficient in the nonlinearity (a sigmoid) resulted in a firing rate which was mostly either $0$ or $1$. This made it crucial to center it around the threshold value. This resulted in too many spikes above threshold and none below. To obtain a corresponding amount of spikes in both the LNP and AHP models a mixture of modifications can be made: reducing $\mu$ the AHP time constant, reducing the peak probability of spiking, and constraining the LNP time domain. The peak probability was set at $50\%$ and the LNP spike time domain was binned down sampling by a factor of $4$. Among the $500$ cases, the average learning kernel

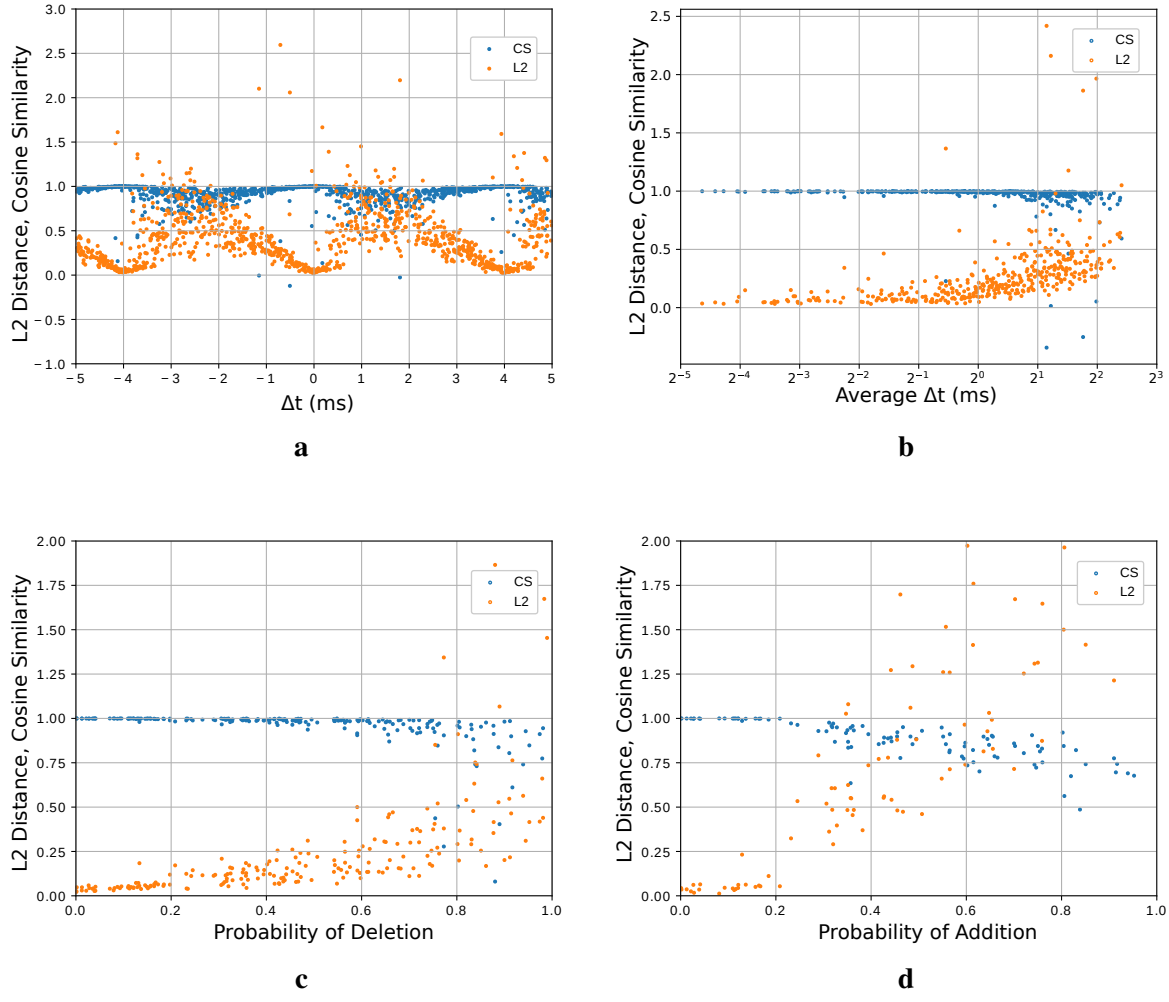



**Fig. 7.** **a**, Shifting spikes uniformly by a set amount resulted in a cyclic pattern as the 10 splines snapped into alignment with adjacent splines. **b**, Randomly perturbing each desired spike separately about a normal distribution. **c**, **d**, Picking a homogeneous Possion rate for deleting (or adding) spikes reveals that as many as half (or a fifth) of the spikes could be deleted (or added) while still extracting a kernel with a low $L_2$ distance.

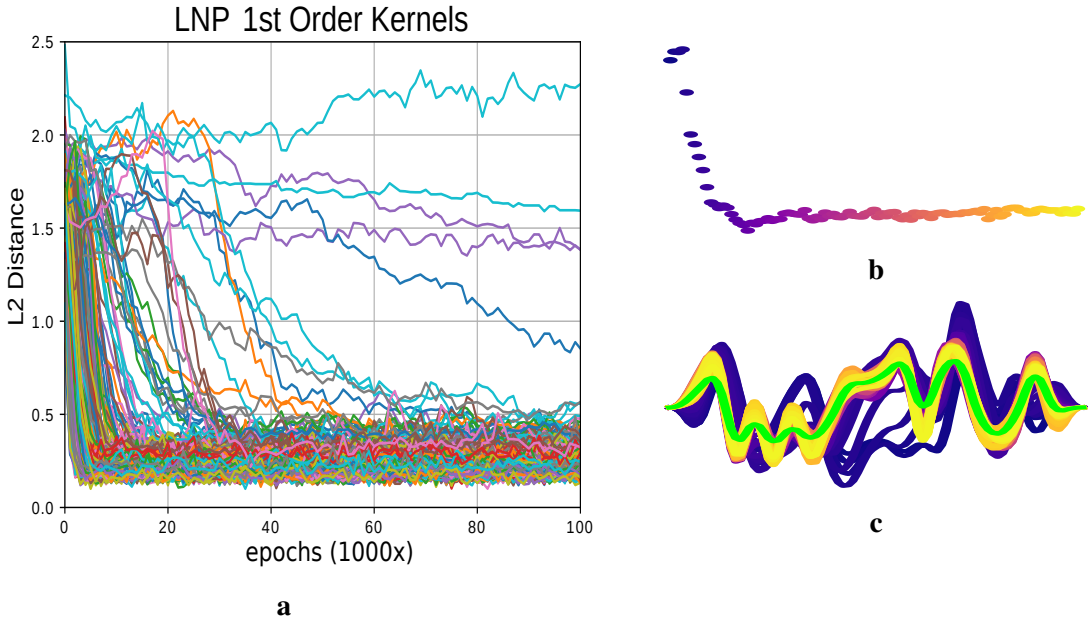



**Fig. 8.** **a**, Convergence for $500$ STD trials on a $4$-binned LNP process learning $20$ splines of $80ms$ over $100,000$ epochs of length $400ms$. **b**, Thirteenth worst convergence peaking at an $L_2$ distance of $178\%$ and ending at $42.3\%$. **c**, Intermediary learning kernels vs the desired (lime).

had a $20\%$ lower error.

## Kernel for *Locusta migratoria* tympanal nerve

Sensory neurons which, when given the same stimuli, reliably produce similar spike trains are good candidates to test STD. One such possibility is the tympanal nerve's auditory receptor axons in the *Locusta migratoria* grasshopper whose action potentials, recorded intracellularly, can have a $0.15ms$[18] inter trial jitter. This dataset, collected by Ariel Rokem[19] [20] at the lab of Andreas Herz, was graciously shared through the CRCNS program (http://crcns.org) [1]. The stimuli consisted of a carrier wave perturbed by random amplitude modulations and a cutoff frequency of up to $800Hz$. We used a particular subset of this dataset to demonstrate kernel extraction.

Repeated updates lead to a kernel which attempts to reconstruct the provided spike train.

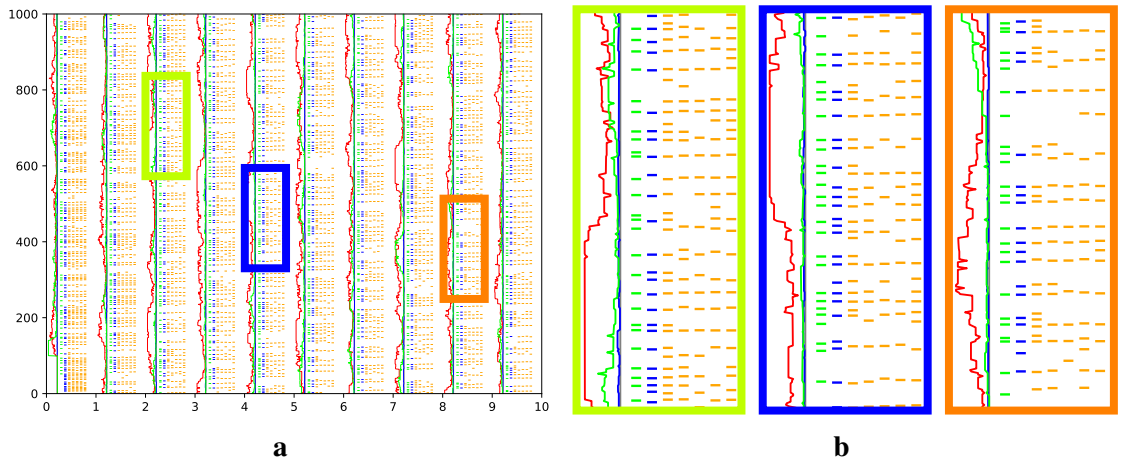



**Fig. 9.** **a**, The *Locusta migratoria* spike data (blue and orange dots) collected by A. Rokem in cell "./crcns-ia1/Data1/03-04-23-ad/" corresponding to the "gauss_st6_co200.dat" stimuli with a cutoff frequency of $200Hz$. The recording is sliced up with milliseconds on the y axis and seconds on the x for a total of $10$ seconds. Each spike train consists of approximately $1000$ spikes. The goal was to train an STD kernel that could reconstruct (green dashes) the desired (blue dashes) spike train. There are $5$ other recordings (orange dashes) for this stimuli and cell. The final kernel's simulated spike train closely approximates the desired. The green line represents error between the learning and desired spike trains from a preceding window of $100ms$ and is clipped to a range from $0$ to $6$. Similarly, the blue and red lines represent the minimum and maximum error between any pair of recorded spike trains. **b**, Zoomed sections of the reconstruction were randomly chosen. Notice that the learning error is often between the minimum and maximum errors between recordings. Additionally, a simulated spike will sometimes align with the other recordings even when the desired does not. These are indications that the kernel accurately represents underlying model.

When running a simulation with a known desired neuron the absolute distance between kernels can be measured. In absence of a known answer, and in lieu of a universally accepted metric, we revert to demonstrating the effectiveness by showcasing the simulated spike train reconstruction for a particular kernel. Figure 9 demonstrates this using a (stimuli, spike train) pair as input. The stimuli we used had a cutoff frequency of $200Hz$. The reconstruction (green dashes) was learned from (blue dashes) one of the $6$ recorded spike trains for a particular cell ("./crcns-ia1/Data1/03-04-23-ad/"). The reconstruction occasionally fires where the desired does not but the other (orange dashes) recordings do. This is an indication that the underlying model is being

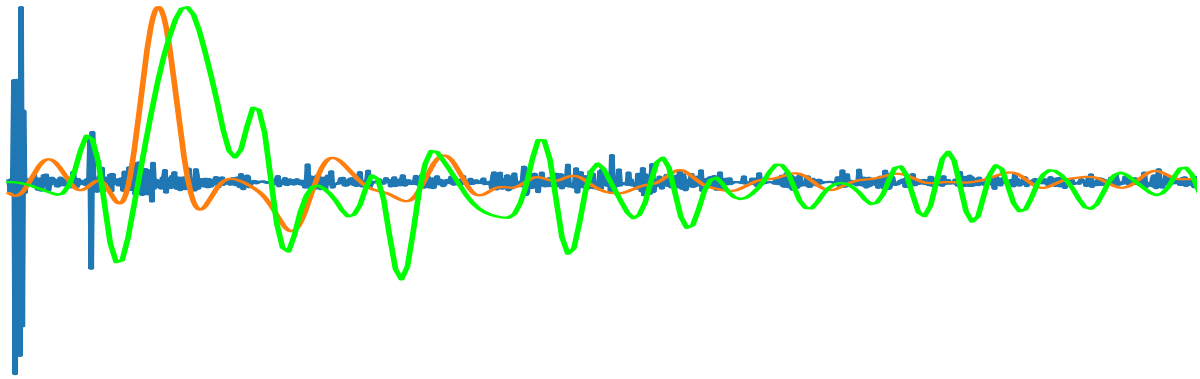


**Fig. 10.** The raw STA kernel (blue) spanning $1000$ time units of $0.05ms$ was smoothed (orange) and compared with the STD result (green). The smoothed STA kernel was the result of of convolving $5$ times against a box spline spanning $1ms$.

accurately represented.

Given that the data set was recorded at $0.05ms$ and borrowing inspiration from Nemenman's work[21], showing that a blowfly's H1 neuron represents information with sub-millisecond precision even for slow stimuli, finer time resolution STD kernels were explored. The best observed reconstruction was for an STD kernel (green) composed of $50$ splines and four $0.25ms$ units per knot interval shown in Figure 10 along with raw (blue) and smoothed (orange) STA kernels. The STA results were omitted from Figure 9 due to a large variety of low quality reconstructions. It was trained with all $6$ recorded spike trains, different sized kernels, various levels of smoothing, and multiple spike generation methods (LNP, CSRM). STD is a better technique for learning kernels that have the ability to reconstruct spike trains.

# Discussion

STD, unlike STA, can learn higher order kernels and generate spike trains to verify the accuracy of its approximation. STA is restricted to linear kernels and doesn't provide a nonlinearity for reconstruction. STD provides a better kernel approximation than STA in our simulated examples. Robustness to a variety of noise and a cross model comparison indicate its potential

to be widely applicable. The simulated reconstructions of the *Locusta migratoria* spike trains are closely aligned which indicates that STD can extract the kernel from many different neural systems.

The primary concern is that model mismatch might make convergence impossible. Unfortunately, this could be confused with incorrect hyperparameters such as the cap, AHP value, threshold, learning rate, momentum, and epochs. Convergence can be measured in terms of parameter stabilization, the predictive ability of the simulation, and with the spike train distance. Multiple simulations can be run from different initializations, but convergence to the same parameter set does not imply a correct solution. This was first seen when trying to learn LNP spike trains where settings stably converged to an incorrect parameter set. This was due to an incompatible model mismatch and was fixed by down sampling.

Slowly changing kernels could be learned from slices of the signal and interpolating the intermediary results. The implementation of the second order kernel operations are similar enough to first order such that core pieces of code can be left untouched for further generalization. The lowest hanging fruit for this technique would be to apply it to other existing datasets. Expanding the code to work for video inputs is another avenue for future research.

## Acknowledgments

Anik Chattopadhyay and Daniel Crews for their assistance with theory and troubleshooting, Kyle Altendorf (@altendky) and Aleksi Torhamo (@Alexer) via #python on IRC in freenode for help with profiling and vectorizing calculations. And thanks to Ariel Rokem for sharing the *Locusta migratoria* dataset and CRCNS for hosting it. This work was partially funded by AFOSR grant #FA9550-16-1-0135.

# Materials and Methods

## Spike Triggered Average as an optimization

Ignoring the nonlinearity due to the proportionality via Bussgang's equation 20 [11] leads to an optimization problem. In the discrete time framework, each successive $|k|$ sized slice of a signal $x$ forms a row of a matrix $X$. The convolution of the kernel with a corresponding section of the signal $k * x_w$ becomes $Xk$. The resulting simulated electrical potential from the convolution is labeled $y$. The optimization problem can then be phrased $min(||Xk - y||^2)$. To solve for $k$ let a partial derivative operator $D = \frac{\partial}{\partial k^T}$ operate on $||Xk - y||^2 = (Xk - y)^T(Xk - y) = k^T X^T Xk - k^T X^T y - y^T Xk + y^T y$ so that $D||Xk - y||^2 = 2X^T Xk - 2X^T Y = 0$ and $k = (X^T X)^{-1} X^T Y$. This formulation is termed whitened STA.

## Third order cardinal B-Splines

Central to STD is the ability to tune parametrized kernels to decrease the spike distance. These kernels are spline functions composed of third order Cardinal B-splines. This causes them to be everywhere differentiable with respect to time. Starting with the recurrence relation in equation (15) (page 90 [22], page 143 [23]), the B-Spline coefficients from [24] are shown in equation (11). The number of steps within each knot sequence was set to $4$ time units. The kernel is constructed by summing the multiple of all splines with their corresponding scaling parameters (12). The $n$ order kernels $K_n$ are formed by B-splines $B_{i,n}$ which are the n-ary Cartesian product of one dimensional splines. This is a deviation from the standard notation where $n$ refers instead to the order of the spline which was always $3$ in our experiments.

$$B_{0,3} = \begin{cases} x \in [0,1) & \frac{1}{2}x^2 \\ x \in [1,2) & -x^2 + 3x - \frac{3}{2} \\ x \in [2,3) & \frac{1}{2}x^2 - 3x + \frac{9}{2} \\ x \notin [0,3) & 0 \end{cases} \tag{11}$$

$$K_n = \sum_{i=0} B_{i,n}\beta_i \tag{12}$$

Verifying the coefficients from [24] using the recursion formulas (13) and (14) from [22] combine into (15). The final representation, displayed similarly to [25], represents what the value of the spline is for each section of its support. This is also commonly displayed as a matrix where elements represent polynomial coefficients [24]. Setting $k = 3$ leads to it being a third order and the knot sequence increasing as counting numbers makes these third order cardinal splines. Successive support regions increment $j$ by $1$ causing an overlap of the nonzero region with the adjacent splines. Adding additional evaluations within each knot range have the effect of smoothing the spline at the cost of taking up more time units.

$$B_{j,k} := w_{j,k}B_{j,k-1} + (1 - w_{j+1,k})B_{j+1,k-1} \tag{13}$$

$$w_{j,k}(x) := \frac{x - t_j}{t_{j+k-1} - t_j} \tag{14}$$

$$B_{j,k} := \frac{x - t_j}{t_{j+k-1} - t_j}B_{j,k-1} + \frac{t_{j+k} - x}{t_{j+k} - t_{j+1}}B_{j+1,k-1} \tag{15}$$

$$k = 3$$

$$B_{j,3} := \frac{x - t_j}{t_{j+2} - t_j}B_{j,2} + \frac{t_{j+3} - x}{t_{j+3} - t_{j+1}}B_{j+1,2} \tag{16}$$

$$B_{j,2} := \frac{x - t_j}{t_{j+1} - t_j}B_{j,1} + \frac{t_{j+2} - x}{t_{j+2} - t_{j+1}}B_{j+1,1} \tag{17}$$

$$B_{j,1} := \begin{cases} x \in [t_j, t_{j+1}) & 1 \\ x \notin [t_j, t_{j+1}) & 0 \end{cases} \tag{18}$$

$$j = 0 \qquad t = [0, 1, 2, 3] \tag{19}$$

$$\begin{aligned} B_{0,3} &:= \frac{x - t_0}{t_2 - t_0} B_{0,2} + \frac{t_3 - x}{t_3 - t_1} B_{1,2} \\ &= \frac{x - 0}{2 - 0} B_{0,2} + \frac{3 - x}{3 - 1} B_{1,2} \\ &= \frac{x}{2} B_{0,2} + \frac{3 - x}{2} B_{1,2} \end{aligned} \tag{20}$$

$$\begin{aligned} B_{0,2} &:= \frac{x - 0}{1 - 0} B_{0,1} + \frac{2 - x}{2 - 1} B_{1,1} \\ &= \frac{x}{1} B_{0,1} + \frac{2 - x}{1} B_{1,1} \end{aligned} \tag{21}$$

$$B_{0,1} := \begin{cases} x \in [0, 1) & 1 \\ x \notin [0, 1) & 0 \end{cases} \tag{22}$$

$$\begin{aligned} B_{1,2} &:= \frac{x - 1}{2 - 1} B_{1,1} + \frac{3 - x}{3 - 2} B_{2,1} \\ &= \frac{x - 1}{1} B_{1,0} + \frac{3 - x}{1} B_{2,0} \end{aligned} \tag{23}$$

$$\begin{aligned} B_{0,3} = &\frac{x}{2} \left( \frac{x}{1} B_{0,1} + \frac{2 - x}{1} B_{1,1} \right) \\ &+ \frac{3 - x}{2} \left( \frac{x - 1}{1} B_{1,1} + \frac{3 - x}{1} B_{2,1} \right) \end{aligned} \tag{24}$$

$$B_{0,3} = \begin{cases} x \in [0, 1) & \frac{x}{2} \frac{x}{1} \\ x \in [1, 2) & \frac{x}{2} \frac{2 - x}{1} + \frac{3 - x}{2} \frac{x - 1}{1} \\ x \in [2, 3) & \frac{3 - x}{2} \frac{3 - x}{1} \\ x \notin [0, 3) & 0 \end{cases} \tag{25}$$

$$B_{0,3} = \begin{cases} x \in [0,1) & \frac{1}{2}x^2 \\ x \in [1,2) & -x^2 + 3x - \frac{3}{2} \\ x \in [2,3) & \frac{1}{2}x^2 - 3x + \frac{9}{2} \\ x \notin [0,3) & 0 \end{cases} \tag{26}$$

## Spike trains as a vector space

Spike trains can be generalized and transformed into a vector space. This is achieved by augmenting them to include coefficients for each spike time along with defining appropriate addition and multiplication operations such that vector space properties are satisfied. Sequence spaces are similar constructions that have a simpler addition operator due to their natural ordering. These augmented spike trains are sequences of 2-tuples which have 5 primary properties: (27), (28), (29), (30), and (31). The times are exclusively in the past (28) and are ordered so that they are strictly increasing (29). The coefficients can be any non zero value in $\mathbb{R}$ (27). These sequences can be finite or countably infinite (30) and are confined to a subspace where the sum of the coefficients converges absolutely (31).

$$\alpha_i \in \mathbb{R} - \{0\} \tag{27}$$

$$t_i \in \mathbb{R}_{>0} \tag{28}$$

$$t_i < t_j \forall i < j \tag{29}$$

$$i, j \in \mathbb{N} \tag{30}$$

$$\sum_{i=0}^{\infty} |\alpha_i| < \infty \tag{31}$$

*Definition*: Let $V$ be the space of finite or countably infinite sequences of time ordered 2-tuples $\boldsymbol{t} = \{(t_i, \alpha_i)\} \in V$ such that (27), (28), (29), (30), and (31) hold.

Adding these augmented spike trains results in a new sequence that consists of adding temporally corresponding coefficients and otherwise concatenating. Defining two intermediary operators (32) and (33) allows for a simpler addition definition for these sequences (34). The spike train vectors $\boldsymbol{t^A}, \boldsymbol{t^B} \in V$ with times $t_i^A, t_j^B$ and coefficients $\alpha_i^A, \alpha_j^B$ correspond to different spike trains $A, B$. Selecting tuples where the time only occurs in the first spike train can be expressed by a modified version (32) of the set difference operator ($A \setminus B$). The set theory intersection operator $\cap$ can be redefined (33) to select where the same time exists in both while adding coefficients and deleting tuples with zero coefficients. The unioned sets in (34) are always disjoint so $\cup$ does not need to be modified.

$$\boldsymbol{t^A} \setminus \boldsymbol{t^B} = \{(t_i^A, \alpha_i^A) : (t_i^A, \alpha_i^A) \in \boldsymbol{t^A} \text{ and } (t_j^B, \cdot) \notin \boldsymbol{t^B} \text{ and } t_i^A = t_j^B\} \tag{32}$$

$$\boldsymbol{t^A} \cap \boldsymbol{t^B} = \{(t_i^A, \alpha_i^A + \alpha_j^B) : (t_i^A, \alpha_i^A) \in \boldsymbol{t^A} \text{ and } (t_j^B, \alpha_j^B) \in \boldsymbol{t^B}, t_i^A = t_j^B \text{ and } \alpha_i^A + \alpha_j^B \neq 0\} \tag{33}$$

$$\boldsymbol{t^A} + \boldsymbol{t^B} = \{(t_i, \alpha_i) \in \{\boldsymbol{t^A} \setminus \boldsymbol{t^B}\} \cup \{\boldsymbol{t^B} \setminus \boldsymbol{t^A}\} \cup \{\boldsymbol{t^A} \cap \boldsymbol{t^B}\}\} \tag{34}$$

$$a\boldsymbol{t^A} = \{(t_i^A, a\alpha_i^A) : (t_i^A, \alpha_i^A) \in \boldsymbol{t^A}, a\alpha_i^A \neq 0\} \tag{35}$$

Scalar multiplication with a vector (35), is only applied to the spike's coefficient and not the time. A doubling would then correspond to a sequence of spikes at the same times but with each coefficient being twice as large. Setting $a = 0$ removes all elements. We now show that for any $\boldsymbol{t^A}, \boldsymbol{t^B} \in V$ the operations $\boldsymbol{t^A} + \boldsymbol{t^B}$ and $a\boldsymbol{t^A}$ satisfy all of the properties (27)...(31) and thus the space is closed under vector addition and scalar multiplication.

Adding spike trains does not alter the set to which the coefficients belong (27) since zeros are deleted by construction. Times are unaffected by (34) except when removed, thus (28) and (29). The indices of spikes can change but are still countable (30). The spike train resulting from addition is just the concatenation of all spike trains $\boldsymbol{t^A} \setminus \boldsymbol{t^B}$, $\boldsymbol{t^B} \setminus \boldsymbol{t^A}$, and $\boldsymbol{t^A} \cap \boldsymbol{t^B}$ with non zero coefficients $\alpha_i^A + 0$, $0 + \alpha_i^B$, and $\alpha_i^A + \alpha_i^B$ respectively. Thus (31) is now a trivial

application of Minkowski's inequality where $p = 1$: $\sum |\alpha_i^A + \alpha_j^B| \leq \sum |\alpha_i^A| + \sum |\alpha_j^B| < \infty$ (page 103 [26], page 588 [27]).

Scalar multiplication results in a new vector with tuple elements in the same set (27) (28). As before with addition, zero elements are explicitly deleted so that multiplying a vector by zero has the effect of removing all spikes. The scalar has no effect on times, except when it is zero and that effect is to remove spikes so the temporal ordering is maintained (29). The indices are untouched, except when all spikes are removed, in either case (30) holds. In (31), the scalar simply factors out of the summation yielding $|a| \sum_{i=0}^{\infty} |\alpha_i^A| < \infty$ which is still clearly a finite sum.

A vector space is made of a field ($\mathbb{R}$), a set of vector objects ($V$), and the following rules: commutativity, associativity, identities, inverse, compatibility, and distributivity [28]. The operations we describe satisfy all of these properties, therefore $V$ is a vector space.

- Commutativity: $\boldsymbol{t^A} + \boldsymbol{t^B} = \boldsymbol{t^B} + \boldsymbol{t^A}$. (34) is symmetric by construction and commutes.
- Associativity: $\boldsymbol{t^A} + (\boldsymbol{t^B} + \boldsymbol{t^C}) = (\boldsymbol{t^A} + \boldsymbol{t^B}) + \boldsymbol{t^C}$. Both of these expressions describe the same sequence due to their set theory expansion.
- Additive Identity: $\boldsymbol{t^A} + \mathbf{0} = \boldsymbol{t^A}$. The zero vector is simply the empty sequence.
- Additive Inverse: This occurs when $a = -1$ in $\boldsymbol{t^A} + a\boldsymbol{t^A} = \mathbf{0}$ leading to all terms canceling and producing the empty vector $\{(t_i^A, \alpha_i^A + -\alpha_i^A)\} = \{\} = \mathbf{0}$.
- Multiplicative Identity: $1\boldsymbol{t^A} = \boldsymbol{t^A}$, where $1$ is the multiplicative identity of the field $\mathbb{R}$ and $\{(t_i^A, 1\alpha_i^A)\} = \{(t_i^A, \alpha_i^A)\}$.
- Compatibility: $a(b\boldsymbol{t^A}) = (ab)\boldsymbol{t^A}$ for $a, b \in \mathbb{R}$ and $\boldsymbol{t^A} \in V$ yields $\{(t_i^A, a(b\alpha_i^A))\} = \{(t_i^A, (ab)\alpha_i^A)\}$, which reduces to field scalar compatibility.

- Distributivity (scalar addition): $(a+b)\boldsymbol{t^A} = a\boldsymbol{t^A} + b\boldsymbol{t^A}$. $\{(t_i^A, (a+b)\alpha_i^A\} = \{(t_i^A, a\alpha_i^A + b\alpha_i^A\} = \{(t_i^A, a\alpha_i^A)\} + \{(t_i^A, b\alpha_i^A)\}$

- Distributivity (vector addition): $a(\boldsymbol{t^A} + \boldsymbol{t^B}) = a\boldsymbol{t^A} + a\boldsymbol{t^B}$. The left side is a scalar multiple applied to the three separate partitions in (34). The term $a\boldsymbol{t^A}$ is then the spikes with corresponding $\alpha_i^A$ components from both $\boldsymbol{t^A} \setminus \boldsymbol{t^B}$ and $\boldsymbol{t^A} \cap \boldsymbol{t^B}$ scaled by $a$. Likewise for $a\boldsymbol{t^B}$.

## Spike trains as an inner product space

The inspiration for this work originates from a similar technique [14] which uses a measure of disparity between spike trains. While other such measures exist, this measure was selected for its ability to model the vanishing and asymmetric aspect of past incoming spikes along with the feasibility of obtaining a closed form gradient. This starts by selecting an appropriately expressive function which can capture enough of the dynamics while also simplifying the analysis. The goal here is to show that an emerging inner product induces a metric in which the subset of finite spike trains is embedded.

This disparity measure leads to a gradient with respect to the parameters which is core to both techniques. The primary difference being that this work learns the parameters of a response function instead of synaptic weights. This results in different applications. A parameterized expression in terms of time models the shape along with the asymmetric and vanishing aspects of post synaptic potentials (36). Integrating over all parameters for two spikes gives an expression in terms of spike times (37).

$$f_{\beta,\tau}(t) = \frac{1}{\tau} e^{\frac{-\beta}{t}} e^{\frac{-t}{\tau}} \quad \text{for} \quad \beta, \tau \geq 0 \quad \text{and} \quad t > \epsilon > 0 \qquad \text{4.1 from [14]} \qquad (36)$$

$$\int_0^\tau \int_0^\infty \frac{1}{\tau} e^{\frac{-\beta}{t_1}} e^{\frac{-t_1}{\tau}} \times \frac{1}{\tau} e^{\frac{-\beta}{t_2}} e^{\frac{-t_2}{\tau}} d\beta d\tau = \frac{t_1 \times t_2}{(t_1+t_2)^2} e^{-\frac{t_1+t_2}{\tau}} \qquad \text{4.4 from [14]} \qquad (37)$$

The following properties needed to make an inner product are symmetry, linearity, and positive definiteness. It will be shown that (38) satisfies these properties and is therefore an inner product. The reordering of the integrals and the summations from (38) to (39) follows from Fubini's theorem since (2) will be shown to be less than infinity. The alphas are a multiple of two absolutely converging sequences (31) is thus bounded (by Theorem 3.50 [29]). Times are positive so terms $\frac{t_i^A \times t_j^B}{(t_i^A+t_j^B)^2}$ and $e^{-\frac{t_i^A+t_j^B}{\tau}}$ are bounded below by zero and above by one. The inclusion of each term has the effect of shrinking the sum and so equation (2) converges. This validates the use of Fubini's theorem and the simplification from (38) to (39).

$$\langle \boldsymbol{t^A}, \boldsymbol{t^B} \rangle = \int_0^\tau \int_0^\infty \left( \sum_{i=1}^\infty \alpha_i f_{\beta,\tau}(t_i^A) \right) \left( \sum_{i=1}^\infty \alpha_i f_{\beta,\tau}(t_i^B) \right) d\beta d\tau \qquad (38)$$

$$\langle \boldsymbol{t^A}, \boldsymbol{t^B} \rangle = \sum_{i,j=1}^\infty \alpha_i \times \alpha_j \int_0^\tau \int_0^\infty f_{\beta,\tau}(t_i^A) \times f_{\beta,\tau}(t_j^B) d\beta d\tau \qquad (39)$$

$$\langle \boldsymbol{t^A}, \boldsymbol{t^B} \rangle = \sum_{i,j=1}^\infty (\alpha_i^A \times \alpha_j^B) \frac{t_i^A \times t_j^B}{(t_i^A + t_j^B)^2} e^{-\frac{t_i^A+t_j^B}{\tau}} \qquad (2)$$

It is symmetric because switching the parameters yields an equivalent result since the nested operations are all individually symmetric. The scalar multiple factors out of the summation, thus: $\langle c\boldsymbol{t^A}, \boldsymbol{t^B} \rangle = c\langle \boldsymbol{t^A}, \boldsymbol{t^B} \rangle$. For additivity, let the vectors be altered to include virtual zeros where there's a corresponding non zero $\alpha$ in the other. These zeros have no effect except to allow for clearer separation. At each point in the summation $\alpha_i^A + \alpha_i^C$ can be split into two separate double summations such that $\langle \boldsymbol{t^A} + \boldsymbol{t^C}, \boldsymbol{t^B} \rangle = \langle \boldsymbol{t^A}, \boldsymbol{t^B} \rangle + \langle \boldsymbol{t^C}, \boldsymbol{t^B} \rangle$.

If $\boldsymbol{t^A} \neq \boldsymbol{0}$ and $\langle \boldsymbol{t^A}, \boldsymbol{t^A} \rangle > 0$ then it is positive definite. Equation (38) integrates the square of a nonlinear function $(F_{\beta,\tau}(\boldsymbol{t}))^2 = (\sum_i \alpha_i f_{\beta,\tau}(t_i))^2 \geq 0$. In order for it to be zero everywhere

it would have to be the zero function, but clearly it is not. Therefore, it is greater than zero at least somewhere causing (2) to be an inner product on the vector space of augmented spike trains. The usual spike trains are simply the subset of finite sequences with the coefficients set to $1$ and ignored. These need to be finite since the countable sequence is unbounded and not in the vector space. This subset is a metric space with the same metric [29] (3) where $E$ an error between spike trains to be minimized.

$$E = d^2 = \langle \boldsymbol{t^A} - \boldsymbol{t^B}, \boldsymbol{t^A} - \boldsymbol{t^B} \rangle \tag{3}$$

## Gradient calculation

Inspired by a similar technique[14], the full Volterra series operator for the GCSRM is perturbed and set equal to the first order Taylor approximation (40). A simplifying step (41) cancels terms. The terms where two perturbation terms appear are additionally removed. Solving for $\Delta t_l^O$ (42) reveals how the spike time changes with respect to changes in the past system. After detailing this general formulation, to clarify further, the simpler first order kernel derivation is shown. Starting again from the threshold equation, we proceed through the partial derivatives until the gradient update calculations.

$$\tilde{\Theta} = K_0 + \sum_{n=1}^{N} \int \cdots \int_n K(\tau_{1\to n}; \beta_{i,l}) \prod_{i=1}^{n} x(t_l^O - \tau_i) d\tau_i + \sum \eta(t_l^O - t_k^O; \mu_k)$$

$$\begin{aligned}
=&(K_0 + \Delta K_0) + \sum_{n=1}^{N} \int \cdots \int_n \Big[ K(\tau_{1\to n}; \beta_{i,l} + \Delta\beta_{i,l}) \Big] \prod_{j=1}^{n} \Big[ x(t_l^O + \Delta t_l^O - \tau_j) \Big] d\tau_j \\
&+ \sum \eta(t_l^O + \Delta t_l^O - t_k^O - \Delta t_k^O; \mu_k + \Delta\mu_k)
\end{aligned}$$

$$\begin{aligned}
=&(K_0 + \Delta K_0) \\
&+ \sum_{n=1}^{N} \int \cdots \int_n \Big[ K(\tau_{1\to n}; \beta_{i,l}) + \sum B_i(\tau_{1\to n}) \Delta\beta_{i,l} \Big] \prod_{j=1}^{n} \Big[ x(t_l^O - \tau_j) + \frac{\partial x}{\partial t}\Big|_{t_l^O - \tau_j} \Delta t_l^O \Big] d\tau_j \\
&+ \sum \Big[ \eta(t_l^O - t_k^O; \mu_k) + \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O} (\Delta t_l^O - \Delta t_k^O) + \frac{\partial \eta}{\partial \mu}\Big|_{t_l^O - t_k^O} (\Delta\mu_k) \Big]
\end{aligned} \tag{40}$$

$$\begin{aligned}
0 = \Delta K_0 &+ \sum_{n=1}^{N} \int \cdots \int_n \Big[ \sum B_i(\tau_{1\to n}) \Delta\beta_{i,l} \Big] \Big[ \prod_{j=1}^{n} x(t_l^O - \tau_j) d\tau_j \Big] \\
&+ \sum_{n=1}^{N} \int \cdots \int_n K(\tau_{1\to n}; \beta_{i,l}) \Big[ \sum_{m=1}^{n} \Big[ \prod_{j\neq m}^{n} x(t_l^O - \tau_j) \Big] \frac{\partial x}{\partial t}\Big|_{t_l^O - \tau_m} \Delta t_l^O \Big] \prod_{j=1}^{n} d\tau_j \\
&+ \sum \Big[ \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O} (\Delta t_l^O - \Delta t_k^O) + \frac{\partial \eta}{\partial \mu}\Big|_{t_l^O - t_k^O} (\Delta\mu_k) \Big]
\end{aligned} \tag{41}$$

$$\Delta t_l^O = \frac{\sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O} \Delta t_k^O - \sum \frac{\partial \eta}{\partial \mu}\Big|_{t_l^O - t_k^O} \Delta\mu_k - \Delta K_0 - \sum_{n=1}^{N} \int \cdots \int_n (\sum B_i(\tau_{1\to n}) \Delta\beta_{i,l}) \prod_{j=1}^{n} x(t_l^O - \tau_j) d\tau_j}{\sum_{n=1}^{N} \int \cdots \int_n K(\tau_{1\to n}; \beta_{i,l}) \Big[ \sum_{m=1}^{n} \Big[ \prod_{j\neq m}^{n} x(t_l^O - \tau_j) \Big] \frac{\partial x}{\partial t}\Big|_{t_l^O - \tau_m} \Big] \prod_{j=1}^{n} d\tau_j + \sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O}} \tag{42}$$

Selecting only the first order kernel ($N = 1$) and starting back at the beginning (43) we show the core derivation used in these experiments. As before, the parameters are perturbed.

$$\tilde{\Theta} = \int K(\tau; \beta_{i,l}) x(t_l^O - \tau) d\tau + \sum \eta(t_l^O - t_k^O; \mu_k) \tag{43}$$

$$
\begin{aligned}
&= \int K(\tau; \beta_{i,l} + \Delta\beta_{i,l}) x(t_l^O + \Delta t_l^O - \tau) d\tau \\
&\quad + \sum \eta(t_l^O + \Delta t_l^O - t_k^O - \Delta t_k^O; \mu_k + \Delta\mu_k)
\end{aligned} \tag{44}
$$

A first order taylor approximation is performed for each of the perturbed variables which emerges from the definition of a derivative:

$$
f'(x) = lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} \tag{45}
$$

At the limit this resolves to $f(x + \Delta x) = f(x) + \Delta x f'(x)$. Substituting this for our variables yields equation (46) where the notation $\Big|_t$ represents the location where the function is evaluated.

$$
\begin{aligned}
&= \int (K(\tau; \beta_{i,l}) + \sum B_i(\tau) \Delta\beta_{i,l}) (x(t_l^O - \tau) + \frac{\partial x}{\partial t}\Big|_{t_l^O - \tau} \Delta t_l^O) d\tau \\
&\quad + \sum (\eta(t_l^O - t_k^O; \mu_k) + \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O} (\Delta t_l^O - \Delta t_k^O) + \frac{\partial \eta}{\partial \mu}\Big|_{t_l^O - t_k^O} (\Delta\mu_k))
\end{aligned} \tag{46}
$$

Setting (43) equal to (46) and canceling out terms yields (47).

$$
\begin{aligned}
0 = &\int \Big( \sum B_i(\tau) \Delta\beta_{i,l} \Big) x(t_l^O - \tau) d\tau \\
&+ \int K(\tau; \beta_{i,l}) \frac{\partial x}{\partial t}\Big|_{t_l^O - \tau} \Delta t_l^O d\tau \\
&+ \sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O} (\Delta t_l^O - \Delta t_k^O) + \sum \frac{\partial \eta}{\partial \mu}\Big|_{t_l^O - t_k^O} \Delta\mu_k
\end{aligned} \tag{47}
$$

An intermediary step of isolating $\Delta t_l^O$ is shown in (48). This can be factored out and becomes the denominator in (49).

$$
\begin{aligned}
\int K(\tau; \beta_{i,l}) \frac{\partial x}{\partial t}\Big|_{t_l^O - \tau} \Delta t_l^O d\tau \\
+ \sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O} \Delta t_l^O
\end{aligned}
\quad = \quad
\begin{aligned}
&\sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O} \Delta t_k^O \\
&- \sum \frac{\partial \eta}{\partial \mu}\Big|_{t_l^O - t_k^O} \Delta\mu_k \\
&- \int \Big( \sum B_i(\tau) \Delta\beta_{i,l} \Big) x(t_l^O - \tau) d\tau
\end{aligned} \tag{48}
$$

Now that an expression for $\Delta t_l^O$ has been obtained, partial derivatives with respect to the parameters can be taken. The partial derivatives in (5), (6), and (7) represent how the timing of spike $t_l^O$ changes with respect to $\beta$, $\mu$, and previous spikes $t_k^O$ respectively. The limits of integration are from the current moment to the end of the kernel.

$$\Delta t_l^O = \frac{\sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O} \Delta t_k^O - \sum \frac{\partial \eta}{\partial \mu}\Big|_{t_l^O - t_k^O} \Delta \mu_k - \int \Big( \sum B_i(\tau) \Delta \beta_{i,l} \Big) x(t_l^O - \tau) d\tau}{\int K(\tau; \beta_{i,l}) \frac{\partial x}{\partial t}\Big|_{t_l^O - \tau} d\tau + \sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O}} \tag{49}$$

$$\frac{\partial t_l^O}{\partial \beta_{i,l}} = \frac{-\int B_i(\tau) x(t_l^O - \tau) d\tau}{\int_0^{|K|} K(\tau; \beta_{i,l}) \frac{\partial x}{\partial t}\Big|_{t_l^O - \tau} d\tau + \sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O}} \tag{5}$$

$$\frac{\partial t_l^O}{\partial \mu_l} = \frac{-\frac{\partial \eta}{\partial \mu}\Big|_{t_l^O - t_k^O}}{\int_0^{|K|} K(\tau; \beta_{i,l}) \frac{\partial x}{\partial t}\Big|_{t_l^O - \tau} d\tau + \sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O}} \tag{6}$$

$$\frac{\partial t_l^O}{\partial t_k^O} = \frac{\frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O}}{\int_0^{|K|} K(\tau; \beta_{i,l}) \frac{\partial x}{\partial t}\Big|_{t_l^O - \tau} d\tau + \sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O - t_k^O}} \tag{7}$$

The following equations (8), (50), (52), and (53) connect the partial derivatives through the chain rule to the error via (51) [14]. The definitions are self referencing and recursive for (8) and (50) where the first instance is just the corresponding partial from before. The expressions (52) and (53) represent how the spike train distance changes with respect to the $\beta$ and $\mu$ parameters respectively.

$$\frac{D t_{k+1}^O}{\partial \beta_{i,l}} = \sum_{t_j > t_l}^{k} \frac{D t_j^O}{\partial \beta_{i,l}} \frac{\partial t_{k+1}^O}{\partial t_j^O} \tag{8}$$

$$\frac{D t_{p+1}^O}{\partial \mu_k} = \sum_{t_j > t_p}^{p} \frac{D t_j^O}{\partial \mu_k} \frac{\partial t_{p+1}^O}{\partial t_j^O} \tag{50}$$

$$\frac{\partial E}{\partial t_i^O} = 2\Bigg( \sum_{j=1}^{N} \frac{t_j^O((t_j^O - t_i^O) - \frac{t_i^O}{\tau}(t_j^O + t_i^O)}{(t_j^O + t_i^O)^3} e^{-\frac{t_j^O + t_i^O}{\tau}} - \sum_{j=1}^{N} \frac{t_j^D((t_j^D - t_i^O) - \frac{t_i^O}{\tau}(t_j^D + t_i^O)}{(t_j^D + t_i^O)^3} e^{-\frac{t_j^D + t_i^O}{\tau}} \Bigg) \tag{51}$$

$$\frac{\partial E}{\partial \beta_{i,l}} = \sum_{k \in F} \frac{\partial E}{\partial t_k^O} \frac{D t_k^O}{\partial \beta_{i,l}} \tag{52}$$

$$\frac{\partial E}{\partial \mu_k} = \sum_{k \in F} \frac{\partial E}{\partial t_k^O} \frac{D t_k^O}{\partial \mu_k} \tag{53}$$

The equations (9) and (10) are the vanilla gradient descent update rules. The terms $\alpha_\beta$ and $\alpha_\mu$ are the respective learning rates while the summation represents the gradient.

$$\beta_i = \beta_i - \alpha_\beta \sum_{k \in F_l} \frac{\partial E}{\partial \beta_{i,l}} \tag{9}$$

$$\mu = \mu - \alpha_\mu \sum_{k \in F_l} \frac{\partial E}{\partial \mu_k} \tag{10}$$

In practice, vanilla gradient descent can be too slow. One way to speed this up is by first updating a momentum term $p$ with respect to the variable $E$ (54). The momentum then replaces the gradient in equations (9) and (10). When $\alpha_p = 0$ this reduces to vanilla gradient descent.

$$p_{i+1} = \alpha_p p_i + \nabla E \tag{54}$$

## Second order gradient calculation

The analysis for $2^{nd}$ order kernels is essentially identical, excluding simple differences in algebra. The core changes are in the setup (55) and in denominator of (56) which matches the

partial derivatives. The numerator for the partials are the same except for $\beta$. The chain rule and update equations are the same. Further generalizations continue to follow this pattern.

$$\tilde{\Theta} = \int\int K(\tau;\beta_{i,l})x(t_l^O-\tau_1)x(t_l^O-\tau_2)d\tau_1 d\tau_2 + \sum \eta(t_l^O - t_k^O;\mu_k) \tag{55}$$

$$\Delta t_l^O = \frac{\sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O-t_k^O}\Delta t_k^O - \sum \frac{\partial \eta}{\partial \mu}\Big|_{t_l^O-t_k^O}\Delta\mu_k - \int\int(\sum B_i(\tau_1,\tau_2)\Delta\beta_{i,l})x(t_l^O-\tau_1)x(t_l^O-\tau_2)d\tau_1 d\tau_2}{\int\int K(\tau_1,\tau_2;\beta_{i,l})x(t_l^O-\tau_1)\frac{\partial x}{\partial t}\Big|_{t_l^O-\tau_2}d\tau_1 d\tau_2 + \int\int K(\tau_1,\tau_2;\beta_{i,l})\frac{\partial x}{\partial t}\Big|_{t_l^O-\tau_1}x(t_l^O-\tau_2)d\tau_1 d\tau_2 + \sum \frac{\partial \eta}{\partial t}\Big|_{t_l^O-t_k^O}} \tag{56}$$

# References


[1] Rokem, A. *et al.* Recordings from grasshopper (locusta migratoria) auditory receptor cells. (2009). URL `http://crcns.org/data-sets/ia/ia-1`.

[2] Hodgkin, A. L. & Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology* **117**, 500–544 (1952).

[3] Boyd, S. & Chua, L. Fading memory and the problem of approximating nonlinear operators with volterra series. *IEEE Transactions on circuits and systems* **32**, 1150–1161 (1985).

[4] Wiener, N. Nonlinear problems in random theory. *Nonlinear Problems in Random Theory, by Norbert Wiener, pp. 142. ISBN 0-262-73012-X. Cambridge, Massachusetts, USA: The MIT Press, August 1966.(Paper)* 142 (1966).

[5] Bialek, W., Rieke, F., Van Steveninck, R. D. R. & Warland, D. Reading a neural code. *Science* **252**, 1854–1857 (1991).

[6] Rieke, F., Warland, D., de Ruyter van Steveninck, R. & Bialek, W. Spikes: Exploring the neural code mit press. *Cambridge MA* (1997).

[7] De Boer, E. & Kuyper, P. Triggered correlation. *IEEE Transactions on Biomedical Engineering* 169–179 (1968).

[8] Marmarelis, P. Z. & Naka, K.-I. White-noise analysis of a neuron chain: an application of the wiener theory. *Science* **175**, 1276–1278 (1972).

[9] Chichilnisky, E. A simple white noise analysis of neuronal light responses. *Network: Computation in Neural Systems* **12**, 199–213 (2001).

[10] Schwartz, O., Pillow, J. W., Rust, N. C. & Simoncelli, E. P. Spike-triggered neural characterization. *Journal of vision* **6**, 13–13 (2006).

[11] Bussgang, J. J. Crosscorrelation functions of amplitude-distorted gaussian signals (1952).

[12] Gerstner, W., Van Hemmen, J. L. & Cowan, J. D. What matters in neuronal locking? *Neural computation* **8**, 1653–1676 (1996).

[13] Rathbun, D., Ghorbani, N., Shabani, H., Zrenner, E. & Hosseinzadeh, Z. Spike-triggered average electrical stimuli as input filters for bionic visiona perspective. *Journal of neural engineering* **15**, 063002 (2018).

[14] Banerjee, A. Learning precise spike train–to–spike train transformations in multilayer feedforward neuronal networks. *Neural computation* **28**, 826–848 (2016).

[15] Michie, D. memo functions and machine learning. *Nature* **218**, 19–22 (1968).

[16] Qian, N. On the momentum term in gradient descent learning algorithms. *Neural networks* **12**, 145–151 (1999).

[17] Allen-Zhu, Z. Katyusha: The first direct acceleration of stochastic gradient methods. *The Journal of Machine Learning Research* **18**, 8194–8244 (2017).

[18] Rokem, A. *et al.* Spike-timing precision underlies the coding efficiency of auditory receptor neurons. *Journal of Neurophysiology* **95**, 2541–2552 (2006).

[19] Eyherabide, H. G., Rokem, A., Herz, A. V. & Samengo, I. Bursts generate a non-reducible spike-pattern code. *Frontiers in Neuroscience* **3**, 2 (2009).

[20] Eyherabide, H. G., Rokem, A., Herz, A. V. & Samengo, I. Burst firing is a neural code in an insect auditory system. *Frontiers in Computational Neuroscience* **2**, 3 (2008).

[21] Nemenman, I., Lewen, G. D., Bialek, W. & Van Steveninck, R. R. D. R. Neural coding of natural stimuli: information at sub-millisecond resolution. *PLoS computational biology* **4** (2008).

[22] d. Boor, C. *A Practical Guide to Splines* (Springer Verlag, New York, 1978).

[23] Farin, G. E. & Farin, G. *Curves and surfaces for CAGD: a practical guide* (Morgan Kaufmann, 2002).

[24] Milovanović, G. V. & Udovičić, Z. Calculation of coefficients of a cardinal b-spline. *Applied mathematics letters* **23**, 1346–1350 (2010).

[25] Marschner, S. & Shirley, P. *Fundamentals of computer graphics* (CRC Press, 2015).

[26] Meise, R. & Vogt, D. *Introduction to functional analysis* (Clarendon press, 1997).

[27] Schechter, E. *Handbook of Analysis and its Foundations* (Academic Press, 1996).

[28] Hoffman, K. & Kunze, R. Linear algebra. 1971. *Englewood Cliffs, New Jersey* .

[29] Rudin, W. *et al.* *Principles of mathematical analysis*, vol. 3 (McGraw-hill New York, 1964).