Final Project Report:
Statistical models of visual neurons

Anna Sotnikova
asotniko@math.umd.edu

Project Advisor: Prof. Daniel A. Butts
dab@umd.edu

Department of Biology

Abstract

Studying visual neurons may explain how a human brain performs its sophisticated and efficient image analysis. The main question is what mathematical model can represent the processes occurring in the brain. Nowadays there have been proposed a variety of models. The measure of accuracy of such models is the difference between the predicted output (neuron's firing rate) and the real experimental data. During this project five models were implemented, which focused on estimating model parameters by moment analysis and stochastic optimization.

May, 13 2017

Content

1 Introduction

Visual system of a neuron is a fundamental area of current research. Scientists are interested in finding answers to the following questions: how a human brain processes a visual information and which mathematical model can describe processes occurring in a human brain during image processing. Answers to these questions will help to predict a brain's responses on certain stimuli, which will allow us to improve image recognition techniques. This way, we will be able to reconstruct an image, which is projected into an eye, by knowing only a few observed spikes.

Here is a brief introduction to the visual system of a neuron illustrated in the Fig. 1. A sensory stimulus is cast into a human eye. Stimulus attributes have to match a sensor, which is light for a visual system. Stimulus has x,y-coordinates, which change in time. The result of stimulus action is a neural activity. The neural activity is represented by spikes. A spike is a change in the response of a neuron. The activity level is reflected by the number of spikes at a certain moment of time - spike frequency.

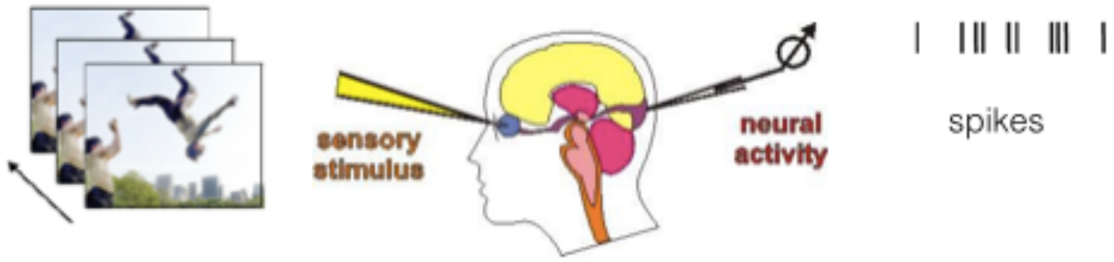


Fig. 1 Visual system of a neuron.
picture source: http://www.pc.rhul.ac.uk/staff/j.zanker/ps1061/l2/ps1061_2.htm

The main difficulty is that a human brain has approximately 86 billions of neurons, and all these neurons process information non-linearly. The main goal is to reproduce the real response of neurons. In other words, knowing the input (a stimulus), what is the output (a firing rate)? The prediction of the output can be viewed as a probability of a neuron to have a spike at a certain moment of time. This prediction is called a firing rate. Obviously, a function, which describes a relation between the stimulus and the response, cannot be a simple linear function. We therefore must spilt this function into a set of other functions with optimally distributed properties (for example, linear filter and non-linear estimator) and this choice of functions will be our model. Simultaneously, our model should not be too complicated so that we could avoid too complicated parameters' fitting procedures and overfitting.

2 Project Objective

The main goal of this project was to study statistical models of visual neurons and implement them for synthetic data and for real experimental data. These models estimate the firing rates of a neuron.

For the first part of the project, we worked with a basic linear model, which is widely used in the computational neuroscience due to its effectives and simplicity. This model is called Linear-Nonlinear-Poisson model [9].
In the model, there are two major steps: estimation of a linear filter that is applied to the stimulus signal and estimation of a non-linear function that converts the filtered stimulus into the firing rate. The linear filter was estimated by moment-based statistical models - Spike Triggered Average, first order moment-based model, and Spike Triggered Covariance, second order moment-based model.

★ Linear-Nonlinear-Poisson model (LNP)

- Spike Triggered Average (STA)

- Spike Triggered Covariance (STC)

For the second part of the project, we worked with maximum likelihood estimators. These models are based on parameters optimization, such that a maximum log-likelihood is achieved.

★ Maximum Likelihood Estimators

- Generalized Linear Model (GLM) [10]

- Generalized Quadratic Model (GQM) [2]

- Nonlinear Input Model (NIM) [2]

Models were applied for three data sets.

- Synthetic data set - Retinal Ganglion Cells data (RGC data) [2]
- Real data set - Lateral Geniculate Nucleus data (LGN data) [3]
- Artificial data set

This manuscript is organized as follows. In section 3 I describe the data and define main parameters and variables. In section 4 I present the studies of the moment-based estimators using mostly LGN data set. In section 5, I introduce maximum likelihood estimator models. I start with the GLM model, and use both synthetic data and LGN data to illustrate all aspects of optimization in these types of models. I then move on to describe more sophisticated non-linear models, GQM and NIM. Section 6 contains project conclusion, sections 7, 8, 9 provide formal information on the project: hardware and software used int this project and project timelines, and a list of deliverables.

3 Data sets description

We begin with a general description of variables and data sets we used. In this section the manipulations with the main data elements are described. In Section 3.1, I present the synthetic data set and briefly cover which models from Section 2 were applied to this data. In Section 3.2, I present the real data set and comment about target goals for this data set as well as for the synthetic data set. The table below defines the notations that are used throughout the text.

$$S - \text{list of all stimulus values sorted by time}$$
$$\mathbf{s}(t) - \text{vector of stimulus values preceding time t}$$
$$s_j(t) - \text{j-th component of the stimulus vector } \mathbf{s}(t)$$

$$n - \text{list of all number of spikes values sorted by time}$$
$$n(t) - \text{number of spikes that occurred at a time t}$$
$$\mathbf{r}_{obs}(t) - \text{vector of number of spikes values preceding time t}$$
$$r_{obs_j}(t) - \text{j-th component of the number of spikes vector } \mathbf{r}_{obs}(t)$$

$$\mathbf{k},\ \mathbf{h},\ \mathbf{k}_i - \text{vectors of parameters defining the linear/history/non-linear filters}$$

Every data set contains the values of the stimulus and the number of fired spikes at different times. Visual stimulus is cast into an eye, every point of an image has a stimulus value and also changes in time. However, we pick up only one point and work with the values of stimulus at this particular point changing with time. So, stimulus data is a list S of scalar values ordered by time. These values are either provided by an external data set (LGN, RGC), or generated by me for validation. In all three cases the components of S are generated as a Gaussian white noise.

Next, we use the values of the list S to define the vector of stimulus values **s**(t) preceding time t. This vector contains values of the stimulus in a time interval between (t-τ, t], where the length of the interval is essentially defined by the length of the temporal filters. The interval τ has a simple biological interpretation. When τ is too long, the values of the stimulus that far away from the moment of time t cannot have any effect on the neuron's response at time t. The length of the stimulus vector **s**(t) is given by the interval τ and by the stimulus discretization time step. Note that the scalar component sj(t) of the stimulus vector **s**(t) represents the value of the stimulus not at time t, but at an earlier time. For example, if the time discretization step is dt, then sj(t) = S(t-(j-1)*dt). Every stimulus vector **s**(t) is placed in correspondence with the number of spikes n(t) at time t that it triggered.

The spikes vector **s**(t) is formally defined as

(1) $$\mathrm{s}(t) = (S(t-\tau), ..., S(t))$$

By analogy, can define spikes vector as

(2) $$\mathbf{r}_{obs}(t) = (n(t-\tau), \ldots, n(t))$$

We use the notation $r_{obs}(t)$ for the spikes vector to be consistent with some of the literature. In principle, this makes sense, because the number of spikes in a time interval dt and the observed firing rate are simply proportionally related by the value of dt. Equations (1) and (2) are going to be used in all calculations. We also note that the time interval τ can be chosen differently for **s**(t) and for $\mathbf{r}_{obs}$(t), and in both cases it is defined by the length of the temporal filters.

In Figure 2 we can see an example of the first 10 seconds of RGC stimulus vector **s** and the corresponding number of spike times **n**.

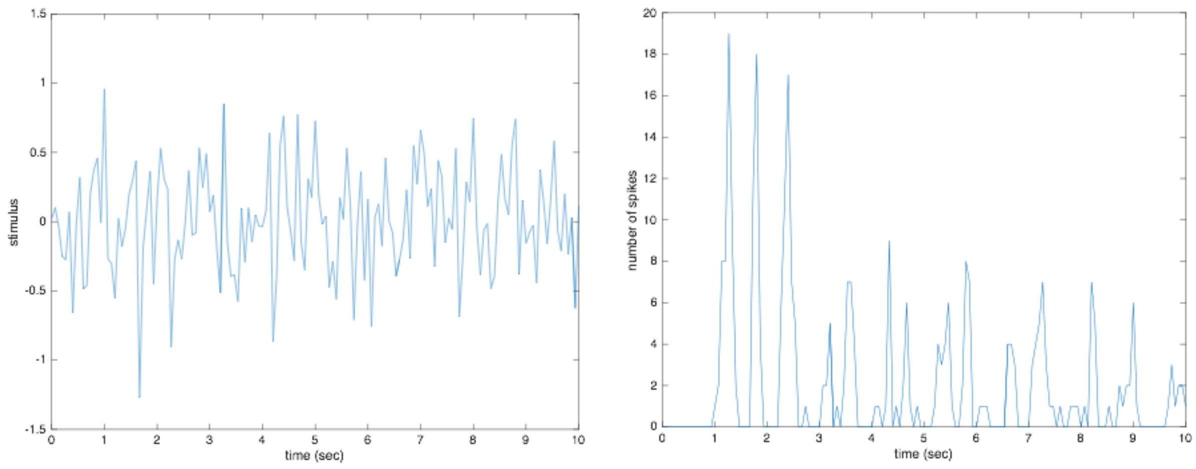


Fig. 2 First 10 seconds of the stimulus and spikes values for the synthetic data (RGC).

3.1 Synthetic data set (RGC data)

Synthetic data set of retinal ganglion cells is referred as RGC data for the rest of the report paper. This set was used in Butts 2013 [2]. RGC data set has the following variables :

1. Stimulus vector **n**
2. Spikes vector **s**

3.2 Real data set (LGN data)

Real data set presents recordings from lateral geniculate nucleus bodies of 3 cats, and is referred as LGN data for the rest of the report paper. This data set was used in Butts 2011 [3] and has the following variables :

1. Stimulus for an experimental duration of 120 seconds and the corresponding spikes vector.
2. Another stimulus of 10 seconds length, which was repeatedly cast into cats' eyes 64 times, and the corresponding spikes vector. (Further I am referring to this stimulus as to the "repeated stimulus.")

3.3 Artificial data set for algorithm validations

Here I generate stimulus as a white noise, calculate the firing rate according to a model under test, and generate spikes using Poisson distribution. Such an artificial data is very useful to check that the algorithm finds correct model parameters.

4 The Linear-Nonlinear-Poisson model

In this section I present the description of Linear-Nonlinear-poisson model. This is a basic widely used model for estimating a neuron's firing rate. The goal of the model is to represent those features of a stimulus that have higher influence on a neuron's response. An important step in LNP model implementation is a filter estimation, which can be done by using one of two moment-based statistical models: either STA or STC model. Procedures of filters' estimations are provided in Sections 4.1 and 4.2 respectively.

The linear models project the stimulus onto a filter, then map this projection nonlinearly into a firing rate. The firing rate at a certain moment of time gives us the probability of the neuron spiking at this moment of time [7],

$$r(t) = F(\mathbf{k} \cdot \mathbf{s}(t)) \tag{3}$$

where F is non-linear function, **k** is a linear filter, and **s**(t) is a stimulus vector preceding a spike at moment t.

The algorithm, which I used for finding the firing rate in the formula (3), may be detailed as follows:

1. Obtain a matrix of stimuli from the given stimulus by the formula (1).
2. Estimate the linear filter **k** by one of the moment-based statistical models.
3. Project all stimuli onto a filter and get the generator signal vector. The value of the generator signal at a certain moment t is given by the formula:

(4) $$g(t) = \mathbf{k} \cdot \mathbf{s}(t)$$

4. Estimate non-linear function F by histograming the spikes and the values of g
5. Apply formula (3) for finding the firing rate.

In order to estimate the linear filter **k**, moment-based statistical models (STA and STC) were used. Their details are provided in the Section **4.1**.

4.1 Moment-based statistical models for filter estimation

The general idea is that the stimulus is a group of points in stimulus space; the stimuli that elicited spikes are a separate group of points in this same stimulus space, and we want to describe the ways in which these two groups differ. The first approach, the Spike-Triggered Average (STA),will look for a difference in the means of these groups, or the difference in the first moment. The second approach expands upon the STA by looking at the difference in the second moment, and is hence called the Spike-Triggered Covariance (STC).

4.1.1 Spike Triggered Average Model

In the Spike Triggered Average(STA) model the output is the linear filter, which does not depend on time. This is the empirical first-order moment-based statistical model, and it provides an estimate of the first linear term in a polynomial series (Wiener/Volterra series) expansion of the response function . The most general physical interpretation of STA is a receptive field of a neuron, which defines the preferred stimulus for the neuron [1]. In other words, the receptive field is the location in space where the presence of visual stimulus can produce a spike. STA estimates the linear stage, which corresponds to one filter, and has the length of a chosen time window [1].

The STA is represented by the formula

(5) $$STA = \frac{1}{N} \sum_{t} n(t)\mathbf{s}(t)$$

where N is the total number of spikes per experiment, n(t) is the number of spikes at time t, and **s**(t) is the stimulus preceding the spike at a time t. Even though **s**(t) are created as random noise, the STA can be non-zero, because the average is weighted by the number of spikes. This is to be contrasted with a conventional mean value

(6) $$\bar{\mathbf{s}} = \frac{1}{N}\sum_t \mathbf{s}(t)$$

This average stimulus is approaching zero at large M because **s** is generated as a white noise.

An important remark about STA model is that a filter found by this model is an optimal filter for the described LNP model in the case the stimuli are Gaussian [5]. This statement can be proven as follows:

If we assume for Gaussian stimuli that a firing rate is represented by the formula:

(7) $$r(t) = \mathbf{k} \cdot \mathbf{s}(t)$$

where the vector **k** is a filter, **s**(t) is a stimulus vector preceding a spike.

Then the mean squared error, which is the log-likelihood of Gaussian distribution, equals to the sum of squared differences between real observations and estimated firing rate for every moment t :

(8) $$MSE = \sum_t (n(t) - r(t))^2$$

In order to find an optimal filter k for our model (7), we need to take derivative of MSE with respect to all the components k_i of the vector k and make them equal to 0:

(9) $$dMSE/dk_i = 2\sum_t (n(t) - r(t))s_i(t)$$

Here we introduced the i-th component of the s(t), given by s_i(t). If we rewrite r(t) as in the formula (7) and get rid of constant 2, we get:

(10) $$\sum_t n(t)s_i(t) = \sum_j k_j \sum_t s_j(t)t_i(t)$$

where sum of stimuli products on the right hand side is white noise. Therefore the right hand side is simply proportional to ki. Consequently, rewriting formula (10) we get:

(11) $$\mathbf{k} = \sum_t \mathbf{s}(t)n(t)$$

If we compare formula (11) and (5), we can notice that they are the same. However, this STA very often does not fit fully the neural feature space because neural responses are mostly non-linear [5]. In addition, for some nonlinearities there might be that the mean of the raw stimuli and mean of the spike-triggered stimuli do not differ, then the STA will be zero. Therefore, the linear filter cannot be estimated, and model does not provide a good characterization. In this case, STC model might be used.

4.1.2 Spike Triggered Covariance Model

The STA model analyzes changes in the spike-triggered stimulus's mean for estimating linear part of LNP model. However, it corresponds only to a single direction of a stimulus. The Spike Triggered Covariance(STC) is used when we need to predict a probability of a spike along more than one direction. STC is second order moment-based statistical model. This model gives us a variance-covariance matrix, and our goal is to find such directions in the stimulus space in which the variances of the spike-triggering stimuli differ from raw stimuli ensemble [6]. The stand-alone eigenvalues of this matrix reveal us possible filters as the corresponding eigenvectors. Notice that having more than two filters makes the problem of fitting the model complicated, because a very large number of stimulus-spikes data points is required for accurate extraction of the multi-dimensional histograms. For a given data set, we are able to work at most with two filters.

STC matrix is represented by the formula:

(12) $$STC = STC_{triggered} - STC_{untriggered}$$

where

(13) $$STC_{trig} = \frac{1}{N}\sum_t n(t)(\mathbf{s}(t) - STA)^T(\mathbf{s}(t) - STA)$$

and

(14) $$STC_{untrig} = \frac{1}{N}\sum_t \mathbf{s}(t)^T\mathbf{s}(t)$$

STC matrix after subtracting the untriggered part gives us two possible filters. Consequently, the formula (3) was adjusted for the firing rate of the two dimensional case:

$$r(t) = F(\mathbf{k_1} \cdot \mathbf{s}(t), \mathbf{k_2} \cdot \mathbf{s}(t))$$
(15)

where instead of one filter as it was in formula (3), we have two filters, which can be extracted from STC matrix defined in formula (12).

Geometrical idea of STC is that we are looking for such directions along which the variance of spike-triggered stimulus differs from the raw stimulus. STC model determines excitatory or suppressive properties of neurons' responses. Excitatory property is defined by the increase in variance, and suppressive property is defined by the decrease in variance [5].

STC gives a quadratic model for neural responses and as well as STA cannot fit data behaving nonlinearly completely. That is why it is often used as starting point for estimation of another model.

4.2 Results of analysis of the RGC data set

For the synthetic data set we only estimated filters by using STA and STC models. The main purpose of using this data set was to validate the code and algorithms for these two models. The detailed explanation for the synthetic data set is given in the Sections 3 and 3.1.

4.2.1 STA filter for the synthetic data set

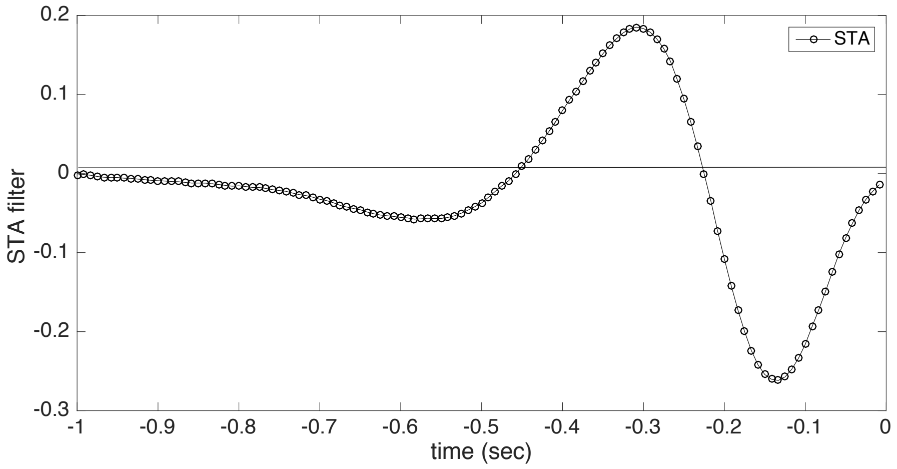


Fig. 3 STA filter extracted for a stimulus preceding a spike of the length of 120 time steps, which corresponds to 1 second.

The first step for the project was to write a code for STA model, obtain the filter using formula (5) and compare the filter with results from Butts 2013 [2]. The result is show in Fig. 3 and matches well the filter reported in the paper.

4.2.2 STC filter for the RGC data set

After calculating the STC matrix using Equations (11-13), the next step was to find its eigenvalues. Since most eigenvalues have approximately the same values around zeros, only the outliers will give the desired filters (Fig. 4). The respective two eigenvectors are the possible choice for filters **k**. Only one filter was shown in the paper of 2013 [2], which is used for the validation of the results. The respective filter is presented on Fig. 4.2.2 and was visually compared with paper's result.

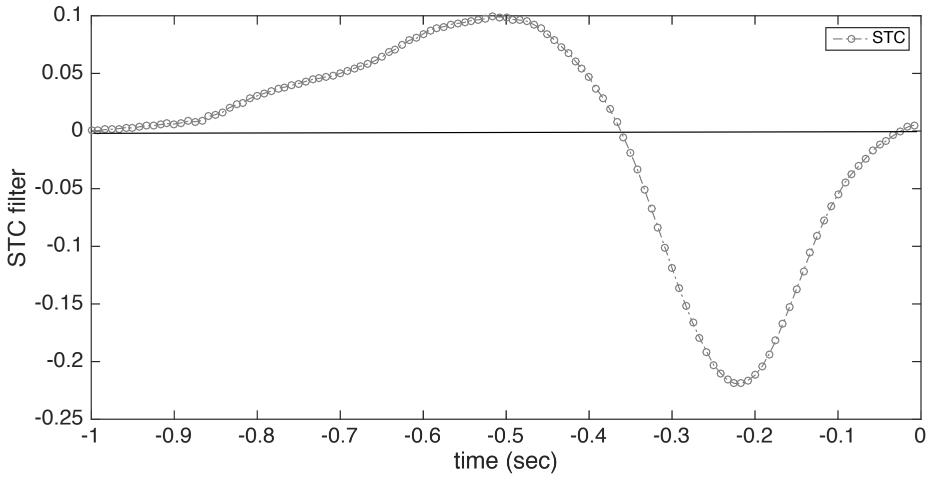


Fig. 4 An STC filter (this is **k** in the report's notations), for a time window length of 120 time steps, which corresponds to 1 second of data.

4.3 Results of analysis of the LGN data set

The LGN data set was used in order to get the firing rates using both models for filters estimation. The obtained firing rates were compared with the real averaged firing rates and R-squared tests were performed. This allowed us to check how close the predictions were to the real measurements. This type of test is called cross-validation.

Cross validation procedure consists of first extracting the model for predicting the neuron's firing rate using the stimulus and spike times data for an experimental duration of 120 seconds. In the second step, we apply the extracted model to an independent piece of the stimulus part of it, to predict the firing rate and compare it directly to the observed rate, which is obtained by averaging the spike count over the trials. The R - squared test was used in order to get a quantitative comparison of how close the predicted spike rate are to the measured one. This tests the performance of the used model [6],[9]. It is crucial to use different stimuli for the model extraction (getting filters' and non-linearities estimations) and for the performance testing [10]. If we use the same data for the model extraction and for the performance estimation, then we get a good estimation for the particular data set, but such model will work poorly for another data set.For a perfect match, the R-squared test yield a value of unity by definition (see formula (16)). The deviations from a unity indicate a discrepancy.

The formula for R-squared is the following:

(16) $$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i y_i^2}$$

where y_i corresponds to measured data, and $\hat{y}_i$ to estimated data.

The good R-squared rate is when the ratio in (16) approaching zero. In other words, the closer R-squared value to 1, the better estimated spike rate matches to the measured spike rate.

For the real data set we found firing rates using two models for filters estimations. The estimated results were cross-validated by comparing them with the measured values. As additional validation of the results, R-squared values were obtained and compared them with expectations for both models' performances.

4.3.1 Firing rate obtained using STA filter for the LGN data set

We used the following scheme in order obtain all mentioned above checks.

- Take stimulus of 120 seconds duration and the corresponding spike times.
  1. Estimate a single linear filter **k** using STA model, formula (5).
  2. Estimate non-linear function F using histogram method.
- Take the repeated stimulus, where the 10 seconds stimulus was repeated 64 times, and the corresponding spike times.
  3. Apply **k** and F from 1st and 2nd steps to the stimulus and obtain the firing rate by the formula (3).
  4. Calculate the average spike rate by averaging the repeated spike times.
  5. Compare the prediction with actual measurements, calculate R-squared value.

4.3.1.1 Step 1 : estimate STA filter

Here the time window length P equals to 15 time steps. The STA filter was obtained by the formula (5) from the stimulus and the respective spike times for the experiment duration of 120 seconds and represented on Fig. 5.

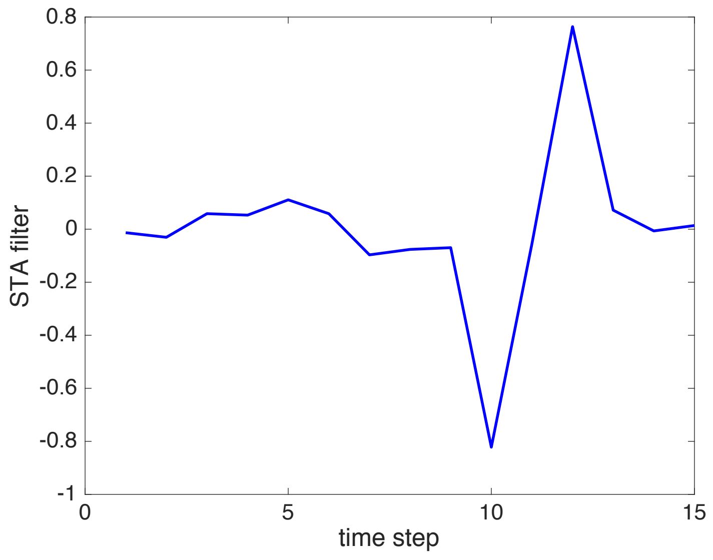


Fig. 5 STA filter, which is **k** in notations for LNP model, extracted for 120 seconds stimulus from the LGN data set. The filter length is 15 time steps, which corresponds to 0.1251 seconds.

The found STA filter was used at step three of the scheme described in Section 4.3.1.

4.3.1.2 Step 2 : estimate non-linearity

In order to estimate nonlinearity we used the histogram method [1], which can be described as follows:

1. Find time-dependent vector of generator signal values by formula (4).
2. Choose number of bins and make a histogram of generator signal values, Fig. 6
3. Calculate the average number of spikes per bin.
    a. Take values of times corresponding to the generator signal values, which belong this particular bin.
    b. Find out how many spikes were at these moments of time by checking the vector of spikes (**n**).
    c. Average the number of spikes corresponding to the bin
    d. Repeat steps a-c for every bin.
4. To do this, we looked up the values of times that corresponded to the stimulus that fits into a particular bin, and, by looking at the spikes data calculate the average number of spikes corresponding to every bin. This, by definition, gives us the non-linearity function F evaluated for discrete values of the argument, given by the bins in Fig. 6. This is the essence of the so-called "histogram method".

This procedure may be also described by the formula (17) and gives us by the definition the non-linearity function F evaluated for discrete values of the argument (Fig. 8.)

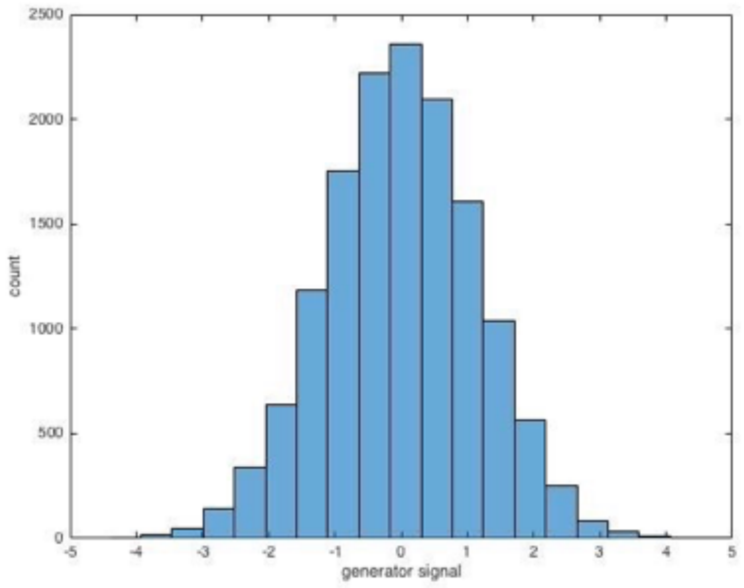


Fig. 6 The histogram of the generator signal g(t) (see Eq. 4) values.

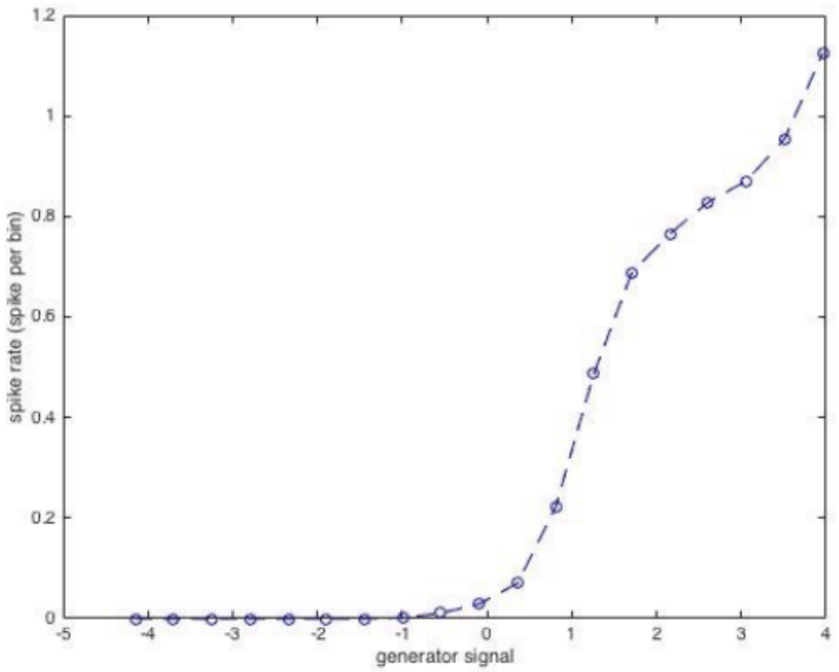


Fig. 7 Reconstructed non-linearity F, which maps calculated generator signal values onto the observed spike rate. Note that according to the LNP model r(t) = F(g(t)).

4.3.1.3 Steps 3-5: cross-validation

The length of the time window for the repeated stimulus is the same as for stimulus used in Sections 4.3.1.1, i.e. 15 time steps. Here we applied the filter found at the first step to the repeated stimulus; this gives us the generator signal values for the repeated stimulus. Then we apply non-linearity F found at the second step to the generator signal. In order to apply the non-linearity F to the new generator signal, the stimulus was projected at a time t onto the linear filter to obtain the argument of F at a time t. Then non-linearity F was applied to this argument and the spike rate was obtained. This procedure estimated the spike rate for the repeated stimulus. Also in order to compare the found spike rate we need to average the number of spikes vector (**n**) for the repeated stimulus (averaging across 64 repetitions). An important remark, since there is a finite number of trials, the more trials give better estimation. As a result, we can see that the estimated spike rate matches to the measured very well (Fig. 8) since the R-squared test defined by formula (16) is 0.7696, which is considered to be a good value because R-squared value equal 1 means the absolute match. The approached result goes along with expectations for LNP model using STA filter.

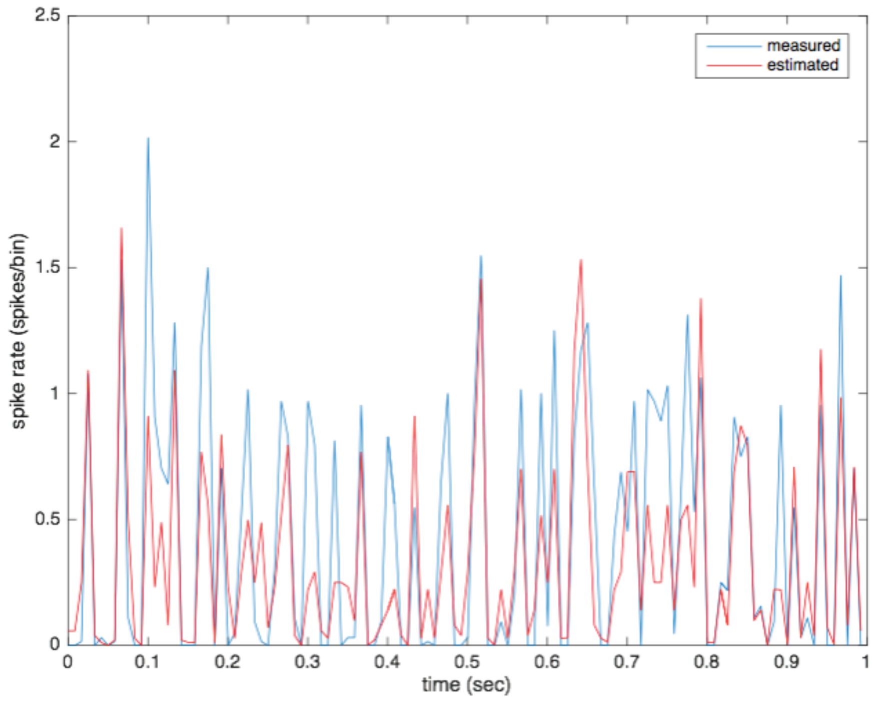


Fig. 8 Cross-validation for spike rates. Red line is the estimated spike rate, blue line is the measured spike rate.

4.3.2 Firing rate obtained using STC filters for the LGN data set

Since STC model presented by formula (12) gives us two possible filters, this makes our model two dimensional and we need to use the formula (15) for the firing rate.

As in 1D case with STA filter, we used the following scheme:

- Take stimulus of 120 seconds' duration and the corresponding spike times.
  1. Estimate two filters **k** using STC model, formula (12).
  2. Estimate non-linear function F using the histogram method.
- Take the repeated stimulus, where the 10 seconds stimulus was repeated 64 times, and the corresponding spike times.
  3. Apply **k**'s and F from 1st and 2nd steps to the stimulus and obtain the firing rate by formula (15).
  4. Calculate the average spike rate by averaging the repeated spike times.
  5. Compare the prediction with actual measurements, calculate R-squared value.

4.3.2.1 Step 1 : estimate STC filters

Here we defined the time window length P equals to 15 time steps. The STC matrix was obtained by the formula (12), and its eigenvalues are on Fig. 9
We can observe two outliers at the lower left corner. The respective eigenvectors are the desired filters.

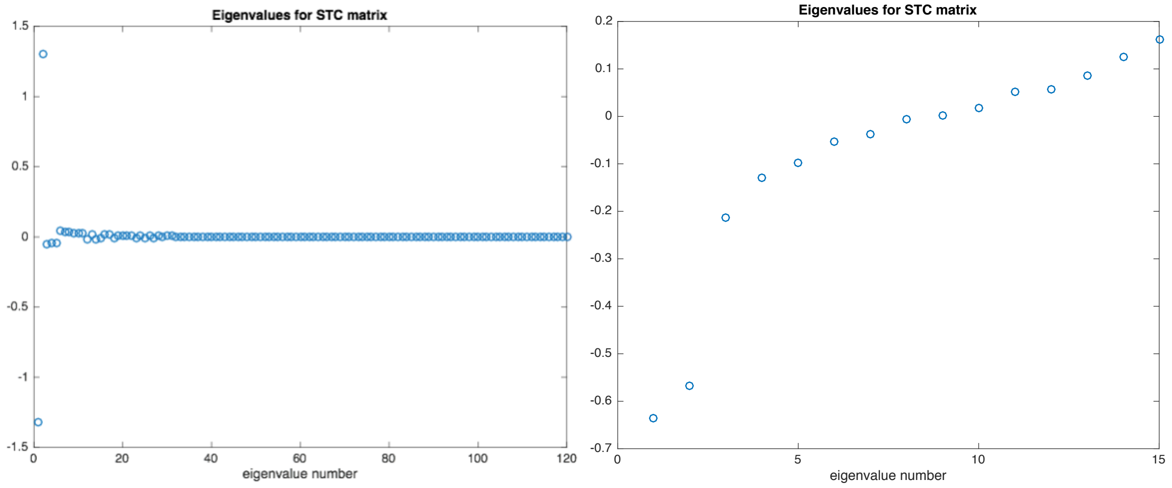


Fig. 9 STC matrix eigenvalues for RGC (left) and LGN (right) datasets. Note two special eigenvalues

4.3.2.2 Step 2 : estimate non-linearity F

Ideologically, the procedure is the same as in the Section 4.3.1.2, but now we have two dimensional problem. Consequently, we find two generator signal values by the formula (4). Then use 2D histogram method for non-linearity estimation.

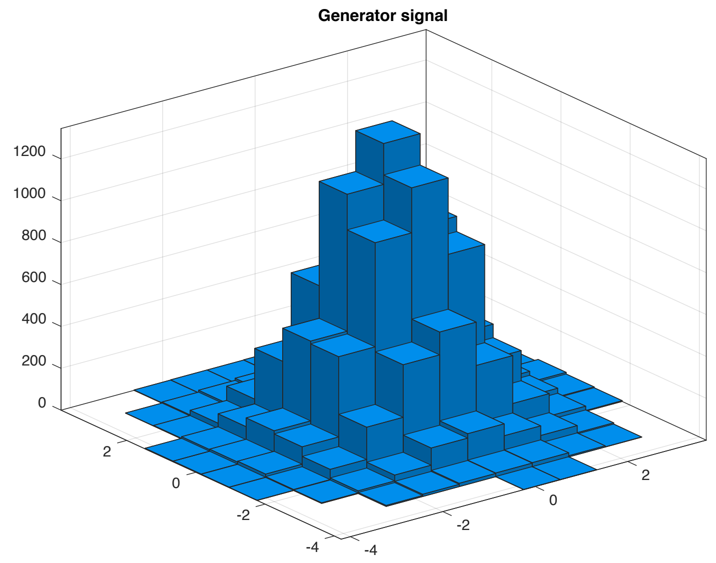


Fig. 10 2D generator signal for the two STC filters (See formula (15)).

Here we can see the impact of lack of data. If in 1D STA case we used 20 bins, here we have 100 bin (10 per dimension), but the stimulus length is still the same 120 seconds. Thus, STC model performs better on larger vectors.
For non-linearity estimation we also followed the procedure explained in Section 4.3.1.2 i.e. we calculated the average number of spikes per bin. This gives 2D non-linear function F that maps generator signals onto a spike rate, which is shown on the Fig. 11 (main 2D picture) and the Fig. 12 (1D perspective)

From Fig. 11 and 12 we can notice which region of the stimulus ensemble in the stimulus subspace is more likely(or vise versa: less likely) to elicit spikes. The drawback is that the amount of data required for N-dimensional space (even for 2D case) grows exponentially with N [5].

4.3.2.3 Steps 3-5 : cross-validation part

Here we applied the filters found at the first step in the Section 4.3.2.1 to the repeated stimulus; this gives us the generator signals for the repeated stimulus. Then we apply non-linearity F found at the second step in the Section 4.3.2.2 to these generator signal values and get spike rate for the repeated stimulus. Also in order to compare the spike rate we need to average the number of spikes vector for the repeated stimulus (averaging across 64 repetitions). As a result, we can see that estimated spike rate matches the measured rate less well than for the 1D case with STA filter (Fig. 8). The R-squared test formula (16) is 0.5374. The result goes along with expectations for LNP model with STC filter. The R-squared test should be lower for 2D case simply because we have less data per bin to fit the non-linear function reliably.

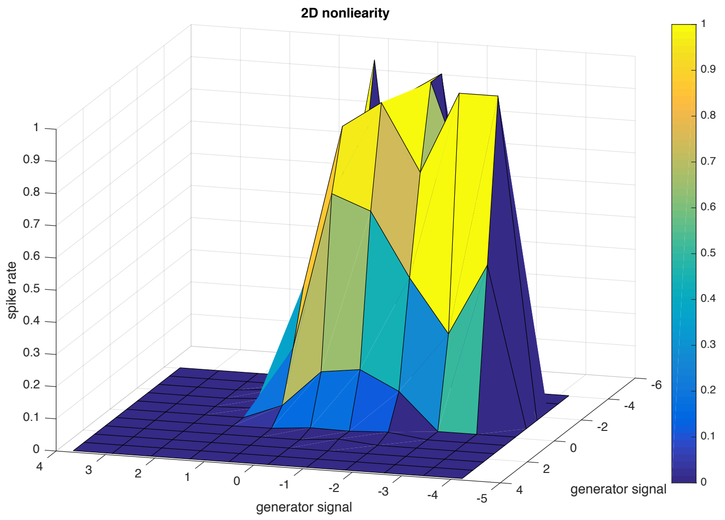


Fig. 11 2D non-linearity F(*,*).

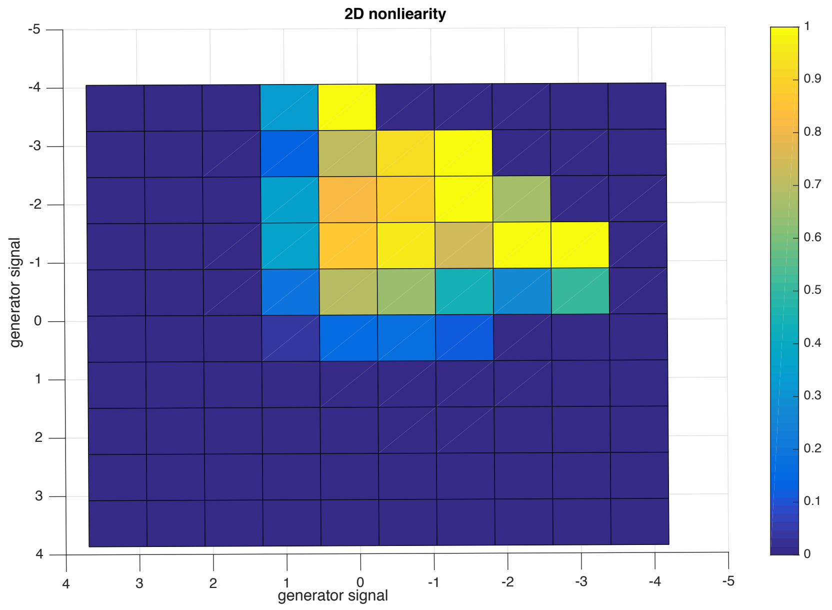


Fig. 12 Density plot of 2D non-linearity F(*,*).

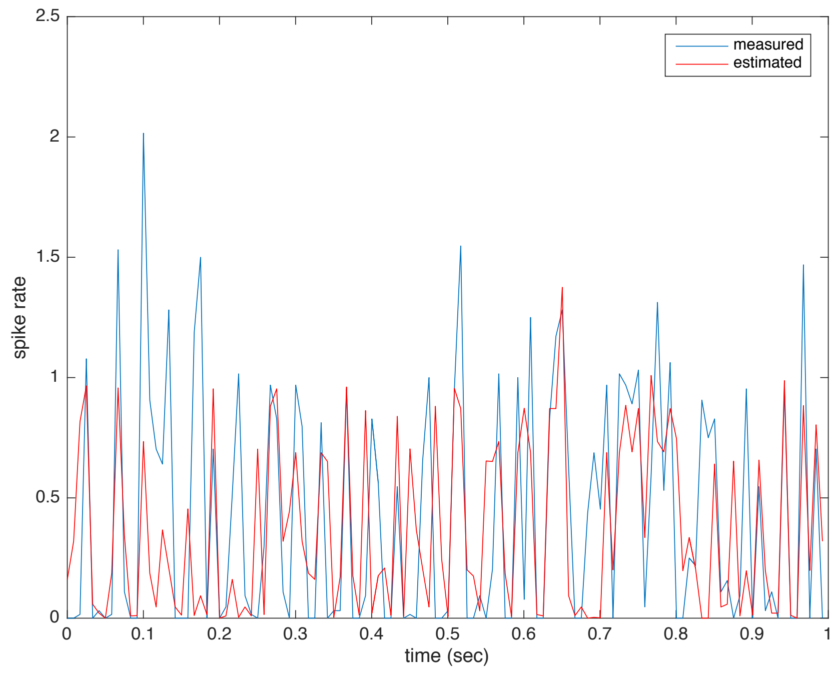


Fig. 13 Cross-validation for spike rates. Red line is the estimated spike rate, blue line is the measured spike rate.

5 Maximum Likelihood estimators

Here we take a different approach. For a given model of the firing rate, which depends on a set of parameters, we ask the question: which parameters are most probable given the observed firing spikes as a function of time? For the same firing rate, a neuron can output a fluctuating number of spikes. In other words, we cannot measure firing rate accurately. Instead, we make an assumption, that spikes are generated by the neuron according to a Poisson distribution and search for the most likely parameters of the model given the observed spikes. Such an approach is widely known as a maximum likelihood estimation, and in some sense replaces conventional list square fitting procedure for a more deterministic data set. Instead of describing the maximum likelihood estimation abstractly, we first introduce the Generalized Linear Model (GLM) and illustrate maximum likelihood estimation using this model. Later on, we generalize this approach to other, more sophisticated models.

5.1 Generalized Linear Model (GLM)

In this section we present the Generalized Linear Model, and use it to illustrate all the details of maximum likelihood estimation methods. We will first present the model and the idea behind Log-likelihood optimization. Then we demonstrate how our GLM algorithm can find optimal model parameters using artificially generated data. We also prove that GLM without a history term is equivalent to STA both by analytical calculation and by comparing the simulations of the two models using the LGN data. We introduce the concept of regularization of the optimal parameter search using a priori information on the expected parameters. Finally, we demonstrate the recovered absolute refractory period of a neuron from the LGN data and show how it can predict the outcome of a real neuron.

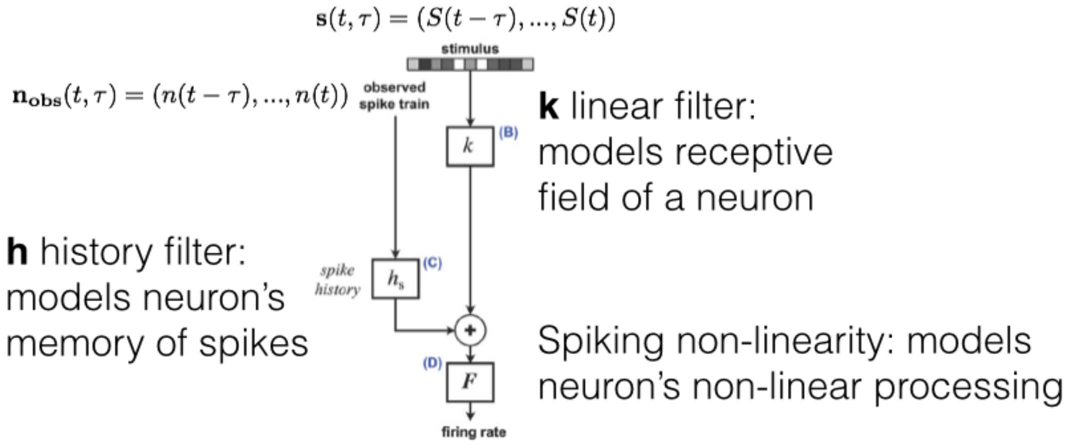

$\mathbf{s}(t,\tau) = (S(t-\tau), ..., S(t))$
stimulus
$\mathbf{n_{obs}}(t,\tau) = (n(t-\tau), ..., n(t))$
observed spike train
k linear filter: models receptive field of a neuron
(B)
k
h history filter: models neuron's memory of spikes
spike history
h_s
(C)
+
(D)
F
Spiking non-linearity: models neuron's non-linear processing
firing rate

Fig. 14 Schematic of GLM model

The Generalized Linear model involves two filters (the linear filter **k** and the history filter **h**) and a spiking non-linearity function F. The components of the two filters are the parameters of the model which need to be found from the optimization procedure. The first filter is responsible for defining the receptive field of a neuron. It preprocesses the stimulus signal prior to converting its information into a decision to fire or not to fire. The second filter takes into account the history of neuron's previous firings, which influence its reaction on the current stimulus. The spiking non-linearity is a rectifying function, whose exact shape is not important, and which models the non-linear processing of the information by a neuron.

Mathematically, the model is defined with the following equations below.

(17)
$$r(t) = F(\mathbf{k} \cdot \mathbf{s}(t) + \mathbf{h} \cdot \mathbf{r}_{obs}(t) + b)$$

with

(18)
$$\mathbf{s}(t) = (S(t-\tau), \ldots, S(t))$$

and

(19)
$$\mathbf{r}_{obs}(t) = (n(t-\tau), \ldots, n(t))$$

5.1.2 GLM: LogLikelihood and gradient formulas

Since a neuron generates spikes probabilistically, how do we compare theory to experiment? For this purpose maximization of LogLikelihood (LL) is used. LL fits the probability and compares two statistical processes. For this project, the negated function, -LL, was minimized using a gradient descent method.

A neuron produces spikes following Poisson distribution. Consequently, probability of having n(t) spikes for a given firing rate r(t) equals

(20) $$P(N|\Theta) = \prod_t \frac{(r(t))^{n(t)}}{n(t)!} exp(-r(t))$$

where r(t) is a firing rate at moment t, n(t) is a number of spikes at this moment.
To simplify equation 9, instead of thinking about the rate parameter r(t) as given in Hertz (spikes per second), we can consider it to be the rate of spikes per bin.

(21) $$N = \{n(t)\}, \Theta = \{r(t)\} = \{\mathbf{k}, \mathbf{h}, b\}$$

In order to fit r(t) to n(t), we need to maximize the probability to have that many spikes at a certain moment of time. Then LL can be written as

(22) $$LL(\Theta) = log(P(N|\Theta)) = \sum_t n(t) log(r(t)) - \sum_t r(t)$$

The formula for the firing rate is the model choice and given by the formula (17). It has been shown by Paninski in [8] that with two reasonable restrictions on the nonlinear function F, the log-likelihood function is guaranteed to have no non-local maxima, which avoids computational issues associated with gradient ascent techniques. The main requirement on F is that it is a convex function and log(F) is a concave function. Most commonly used F that satisfies the two criteria are F(x) = Exp(x) and F(x) = log(1+ Exp(x)). Here, for simplicity we will use F(x) = Exp(x).

Consequently, we can write LL as

(23) $$LL(\Theta) = \sum_t n(t)(\mathbf{k} \cdot \mathbf{s}(t) + \mathbf{h} \cdot \mathbf{r}_{obs}(t) + b) - \sum_t exp(\mathbf{k} \cdot \mathbf{s}(t) + \mathbf{h} \cdot \mathbf{r}_{obs}(t) + b)$$

The maximum value of this function, known as the maximum likelihood, will correspond to the optimal parameters **k** and **h** that are most likely to produce the spike train given by s(t) and n(t).

To proceed with the optimum search with the gradient method we compute analytically the gradients, which are given by

(24) $$\frac{dLL}{dk_i} = \sum_t n(t)s_i(t) - \sum_t exp(\mathbf{k}\cdot\mathbf{s}(t) + \mathbf{h}\cdot\mathbf{r}_{obs}(t) + b) * s_i(t)$$

(25) $$\frac{dLL}{dh_i} = \sum_t n(t)r_{obs_i}(t) - \sum_t exp(\mathbf{k}\cdot\mathbf{s}(t) + \mathbf{h}\cdot\mathbf{r}_{obs}(t) + b) * r_{obs_i}(t)$$

(26) $$\frac{dLL}{db} = \sum_t n(t) - \sum_t exp(\mathbf{k}\cdot\mathbf{s}(t) + \mathbf{h}\cdot\mathbf{r}_{obs}(t) + b)$$

5.1.3 Validation of GLM algorithm using artificial data

Here we have generated artificial spiking data using a white noise stimulus, several test filters **k** and **h**, and the expression for the GLM's firing rate. We first take a simple exponentially decaying filter k, which simply implies that the stimulus values immediately before the spike are more influential than the older stimulus values. In this test the history term was set to zero. It is clear that our GLM algorithm recovered the filter perfectly

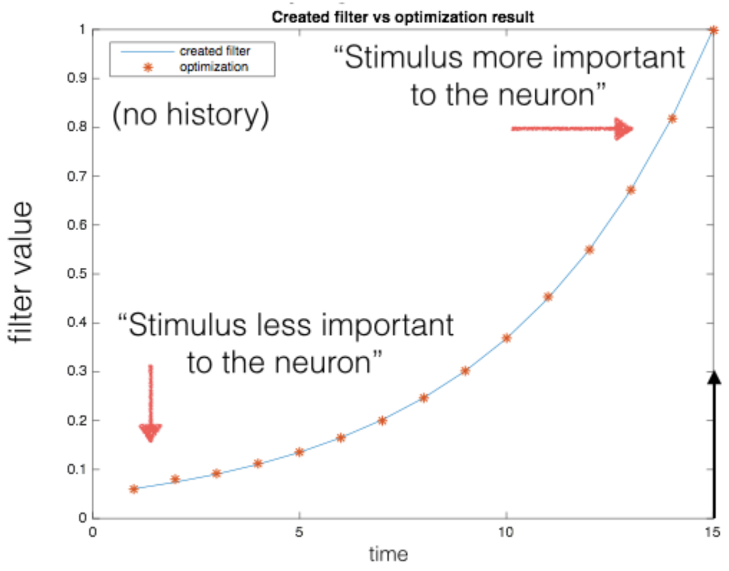


Fig. 15 Recovered decaying k-filter vs model filter with artificial GLM data

Next, we use a more sophisticated linear filter which oscillates while decaying in time. Such a filter is sensitive to a more subtle patterns in the stimulus signal. For instance, it will nullify the effect of a constant stimulus, but will amplify the one which oscillates with a commensurate period, which could potentially model a biological process. Such an oscillatory filter is also perfectly recovered by our algorithm.

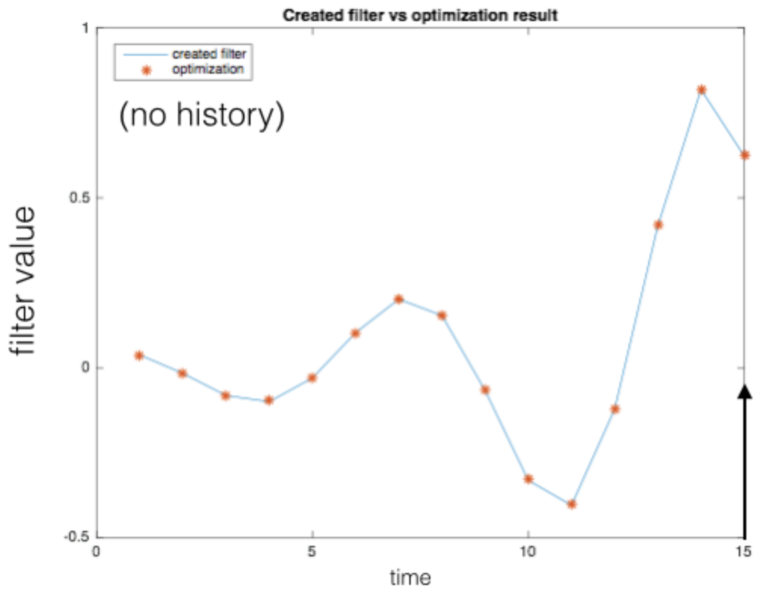


Fig. 16 Recovered oscillatory k-filter vs model filter with artificial GLM data

As a next test, we introduce both k-filter and h-filters in the form of decaying exponents. It should be noted that an h-filter only makes sense if all of its values are chosen negative. This way history term will suppress firing for certain history of previous firing. A positive h-filter will set up a positive feedback and will result in exponentially growing number of spikes. As one can see, our algorithm perfectly recovers the two filters as well.

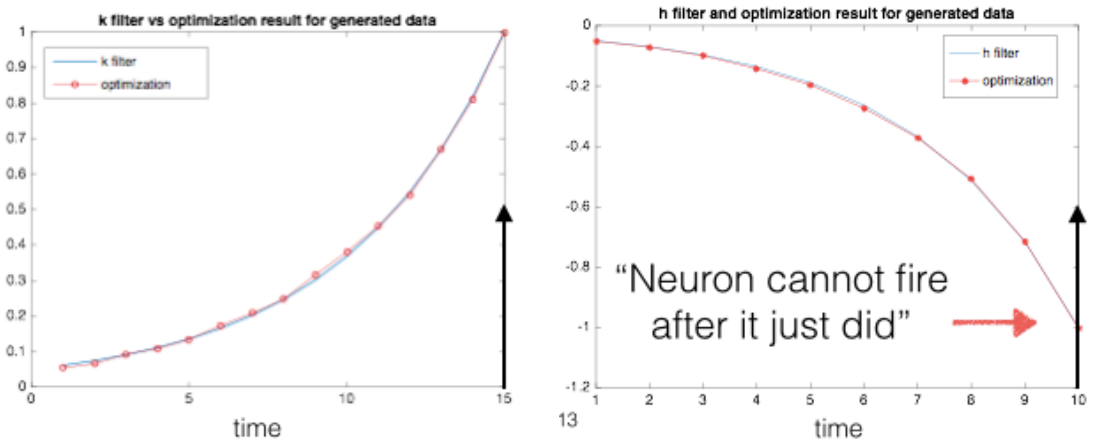
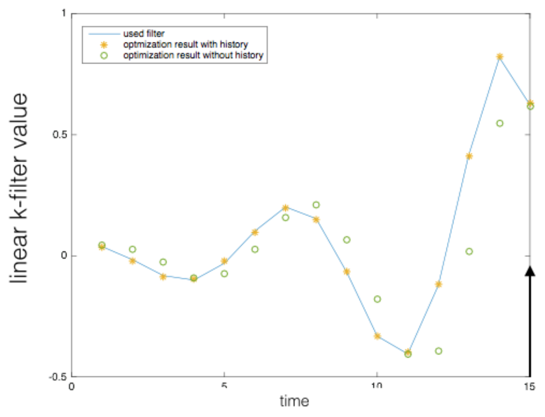


Fig. 17 Recovered linear k-filter and history h-filters from an artificial GLM data

As a final algorithm test we input a decaying history filter h and an oscillatory filter k to generate artificial data and run the GLM algorithm with and without a history term. It turns out that if the history was simulated in but not used in recovery, the recovered k-filter substantially differs from the original one. This finding illustrates the importance of history term in GLM. When the history term is properly taken into account, the recovered k-filter perfectly matches the modeled one.



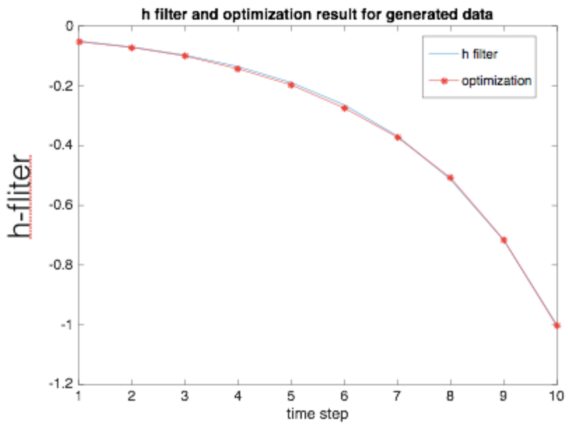


Fig. 18 Recovered linear k-filter and history h-filters from an artificial GLM data. Green circles correspond to not taking into account the history term and thus don't match with the true k-filter.

5.1.4 Equivalence of GLM without a filter and STA models

Here we take the synthetic RGC data and recover a linear filter k and plot it as a function of the calculated STA. The two match perfectly. This match is a manifestation of a well known fact that for a gaussian process, minimization of the least squares of the difference between the observed rate and the model rate yields STA. Here we have a Poisson process, but at a large spike number, the Poisson distribution becomes indistinguishable from a gaussian distribution, and hence the calculations of Section 4 fully apply to this case.

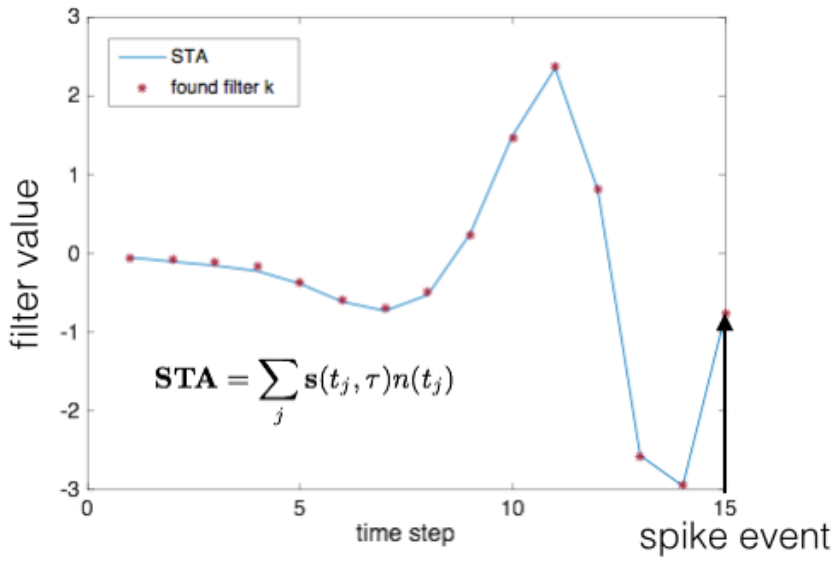


Fig. 19 RGC data: comparison of the calculated STA (see section 4) and recovered linear filter from the GLM model without history.

5.1.5 Regularization of LogLikelihood optimization

In case of the LGN data set, the history filter can be detected only at high resolution (small time interval). This means that the stimulus stays constant for several time intervals, while the spiking behavior still changes due to the stochastic nature of the spiking process. The increase of the resolution requires some changes in the LogLikelihood function. Namely, we need to add a regularization term, which penalizes LL if there are unphysical high frequency variations in the other filter **k** . The linear filter should be smooth from biological considerations, the regularization term helps to get rid of noise in the filter, which makes the problem ill-posed and might lead to the false local maximum. The regularization term is only sensitive to the large fluctuations in the filter **k** values.

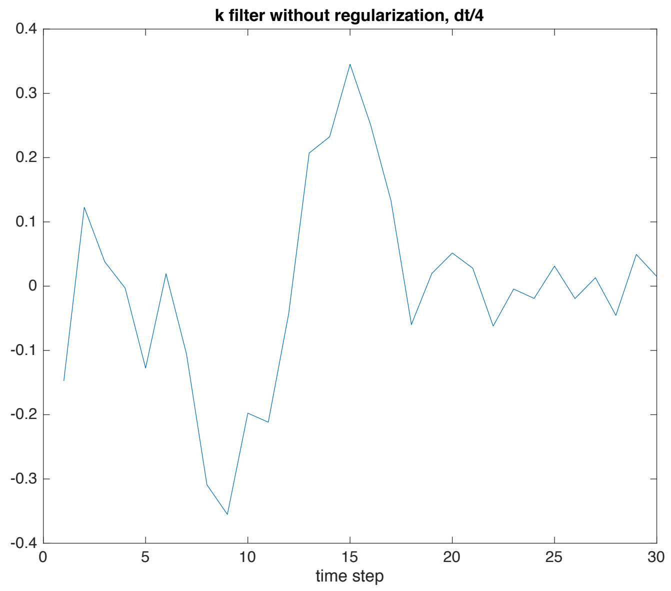


Fig. 20 Noisy linear filter recovered by a GLM model at high resolution for the LGN data

Adding the regularization term changes LL to the Regularized LL (RLL):

$$RLL(\Theta) = \sum_t n(t)(\mathbf{k} \cdot \mathbf{s}(t) + \mathbf{h} \cdot \mathbf{r}_{obs}(t) + b) - \sum_t exp(\mathbf{k} \cdot \mathbf{s}(t) + \mathbf{h} \cdot \mathbf{r}_{obs}(t) + b) - \lambda \sum_i (k_i - k_{i-1})^2 \tag{27}$$

with lambda parameter, which should be optimized as well. The procedure for finding the optimal lambda is described later in this section. The regularization term for an optimal lambda would be sufficiently small not to alter then optimal filters, and it grows quickly for a noisy filter, such as that in the figure above.

As with LL case, in order to maximize RLL we are using gradient ascent method. The gradient for RLL is given by the following formulas:

(28) $$\frac{dRLL}{db} = \sum_t n(t) - \sum_t exp(\mathbf{k}\cdot\mathbf{s}(t) + \mathbf{h}\cdot\mathbf{r}_{obs}(t) + b)$$

(29) $$\frac{dRLL}{dh_i} = \sum_t n(t) r_{obs_i}(t) - \sum_t exp(\mathbf{k}\cdot\mathbf{s}(t) + \mathbf{h}\cdot\mathbf{r}_{obs}(t) + b) * r_{obs_i}(t)$$

(30) $$\frac{dRLL}{dk_i} = \sum_t n(t) s_i(t) - \sum_t exp(\mathbf{k}\cdot\mathbf{s}(t) + \mathbf{h}\cdot\mathbf{r}_{obs}(t) + b) * s_i(t) - 2*\lambda(k_i - k_{i-1}) + 2*\lambda(k_{i+1} - k_i)$$

in (29-30) i is from 2 to P-1

(31) $$\frac{dRLL}{dk_1} = \sum_t n(t) s_1(t) - \sum_t exp(\mathbf{k}\cdot\mathbf{s}(t) + \mathbf{h}\cdot\mathbf{r}_{obs}(t) + b) * s_1(t) + 2*\lambda(k_2 - k_1)$$

(32) $$\frac{dRLL}{dk_P} = \sum_t n(t) s_P(t) - \sum_t exp(\mathbf{k}\cdot\mathbf{s}(t) + \mathbf{h}\cdot\mathbf{r}_{obs}(t) + b) * s_P(t) - 2*\lambda(k_P - k_{P-1})$$

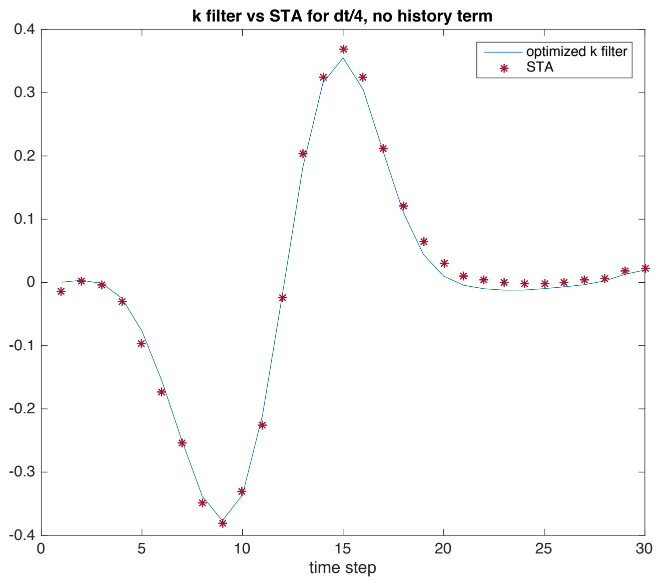


Fig. 21 Same filter at high resolution as in the previous figure with regularization.

The procedure for the choice of lambda is the following. The data is divided into two equal parts (folds). We first use the first part to obtain the filters using the regularization with some value of lambda, and then plug the obtained optimal filters into the second part of data to find the value of the log-likelihood. We then swap the two parts, and repeat the procedure. The two found log likelihood values are averaged and plotted as a function of lambda. The results are plotted in a figure below

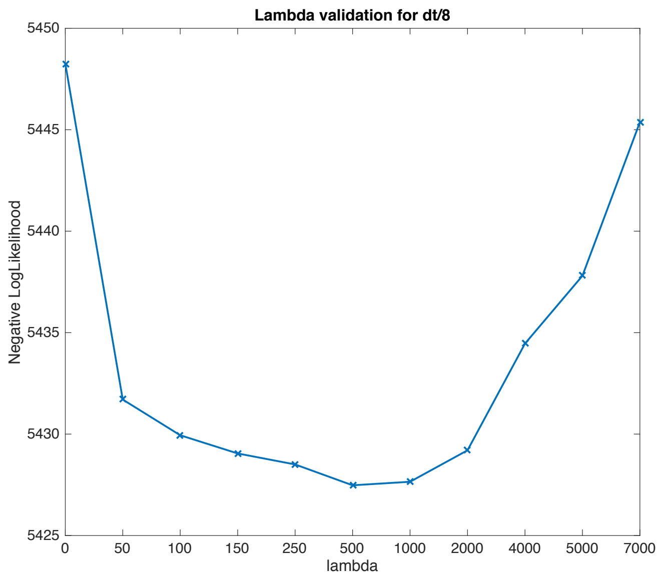


Fig. 22 Cross-validated values of the regularization parameter for LGN data

It is clear that there is wide range of lambdas where log-likelihood has a flat minimum, which indicates that the role of lambda reduces to simply throwing away noisy filters.

5.1.6 Optimal filters for the LGN (experimental) data set

Here we first go to resolution “dt/4” which means that we a stimulus that is kept constant at 4 consecutive time steps (but the spiking output will of course vary due to stochastic nature of the firing process). The LGN data was taken such that the spike times were recorded with a much higher resolution than the variation of the stimulus. This is why we can easily change the resolution in looking for an optimal filter. Higher resolution is only needed if it is expected that a filter shows sharp variations in time. This is not expected from the linear k-filter. However, a history filter is expected to vary sharply. The time scale of these variations is associated with the time period during which takes a neuron to “recharge” after firing.

First, we estimate **k** and **h** filters at a resolution dt/4. As we might see on the Fig. 23 k filter is found, but for h filter the resolution is too low and the algorithm cannot catch it. It was found out that the resolution at which we can identify the filter is dt/16. From Fig. 23 we notice that k filter is found, but h filter still requires increase in resolution.

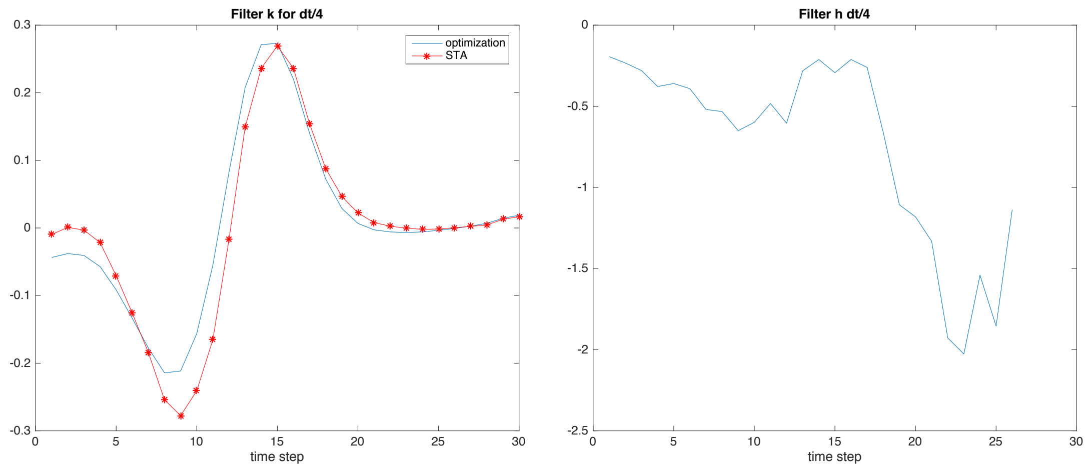


Fig. 23 Filters k and h found from optimization with the LGN data. Note that with the history filter taken into account, STA calculated from the data does not match anymore with the optimal linear filter.

Increasing resolution further allows us to learn more details of the history filter. Note that regularization is only applied to the k-filter, and its smooth shape, implied already from the STA, justifies the search for a smooth filter. By contrast, in order to catch possible sharp features history filter should not be regularized. We increased the resolution by 4 more times, to the “dt/16” (i.e. stimulus was repeated 16 times first and then changed to another value), which made the model sensitive to the dynamics at a time scale 16 times faster than in conventional simulation. Going to this resolution confirmed the previously recovered shape of the history filter at dt/4 but also shows another sharp feature just preceding the spike. This feature would significantly suppress the response of a neuron to any stimulus right before the the neuron already fired.

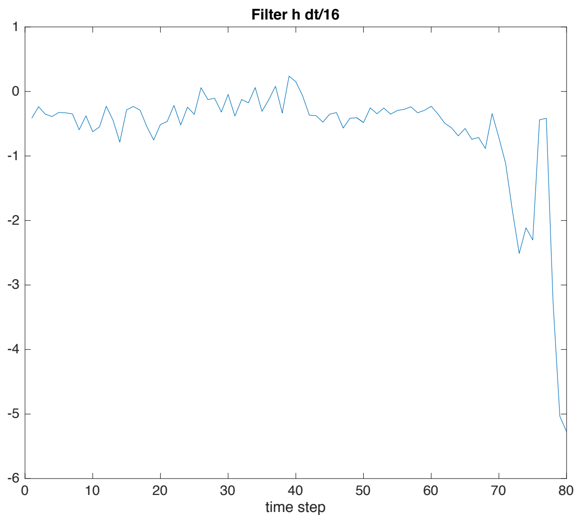


Fig. 24 Absolute refractory period of a neuron found in the optimal history filter at a high time resolution

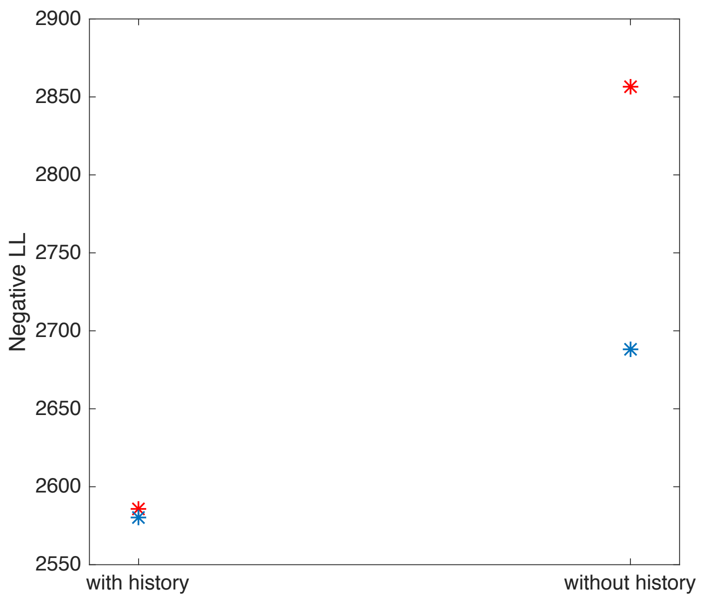


Fig. 25 Comparison of Log-likelihood for LGN data and GLM model with and without a history term. Including history clearly reduces negative LL, which means it's a better model.

In conclusion, the algorithm to optimize log-likelihood for the GLM model was validated using artificial data generated by me with a couple of test filters. Next, a regularization procedure was described and cross-validated using the LGN data and absolute refractory period of a neuron was identified in the history filter. It was pointed out that as expected, without history term, the optimal linear filter of the GLM matches STA, which is not the case if the history is taken into account. The extracted LGN filters allowed us to predict a firing rate with a better LL values than accounting for history.

5.2 GQM

The Generalized Quadratic Model (GQM) is defined by the firing rate equation below. Here there is no history like in GLM. Instead now quadratic filters are introduced. Quadratic filtering corresponds to the same linear filtering of the stimulus, but the resulting scalar is squared, so it can only be positive. Quadratically-filtered stimulus can be taken with either positive or negative sign ($w_i = 1$ or $w_i = -1$). Thus, a non-zero positive quadratic filter always enhances the firing rate, while non-zero negative quadratic filter always suppresses the firing rate. This is the main difference with the linear filter.

(33) $$r(t) = F(\mathbf{k}_L \cdot \mathbf{s}(t) + \sum_i \omega_i (\mathbf{k}_i \cdot \mathbf{s}(t))^2)$$

One can motivate the GQM by noticing that the expression for the firing rate is the simplest way to introduce non-linear pre-processing to the stimulus prior to using it for spiking decision. A more biological interpretation is the following. We consider a stimulus space defined by a set of all possible linearly-independent filters. One of such directions is given by the linear filter kL. In the absence of quadratic term kL matches with the STA. However, it is possible that a neuron is sensitive to several other specific directions, which would result in suppressed or enhanced fluctuations of the spikes, revealed by the simple STC analysis (see Section 4). One can interpret such directions as follows. Stimulus projected onto some directions results in an enhanced firing rate, so the filters defining that direction are called “excitatory”. Other directions can result in a suppressed firing rate, so the filters associated with those directions are called “inhibitory”. The expression for the GQM’s firing rate simply takes into account the possible sensitivity of a neuron to such excitatory and inhibitory directions in the stimulus space. To give a specific mathematical example (which has nothing to do with a real neuron), imagine that a sinusoidally varying stimulus with a certain frequency results in an enhanced spiking rate but the variation at a twice the frequency results in a suppressed spiking rate. In this case one can simply take the corresponding Fourier components as the excitatory and inhibitory filter, respectively. In practice these two filters will have a more complex behavior of course.

Here we compute the both the LL function and its gradients analytically as well, using F as an exponential function. The LL is given by

$$LL(\Theta) = \sum_t n(t)(\mathbf{k}_L \cdot \mathbf{s}(t) + \sum_i \omega_i (\mathbf{k}_i \cdot \mathbf{s}(t))^2) - \sum_t exp(\mathbf{k}_L \cdot \mathbf{s}(t) + \sum_i \omega_i (\mathbf{k}_i \cdot \mathbf{s}(t))^2)$$

The gradients are given by

$$\frac{dLL}{dk_{ij}} = \sum_t 2n(t)\omega_i(\mathbf{k}_i \cdot \mathbf{s}(t))s_j(t) - \sum_t 2exp(\mathbf{k}_L \cdot \mathbf{s}(t) + \sum_i \omega_i (\mathbf{k}_i \cdot \mathbf{s}(t))^2)\omega_i(\mathbf{k}_i \cdot \mathbf{s}(t))s_j(t)$$

5.2.1 Validation of GQM algorithm using artificial data

Validation of the GQM follows the same procedure as the GLM. We first generate a white noise stimulus and spikes according to the GQM’s firing rate expression, using some test filters. We then run the LL-optimization algorithm in order to recover the filters and compare them with the test filters.

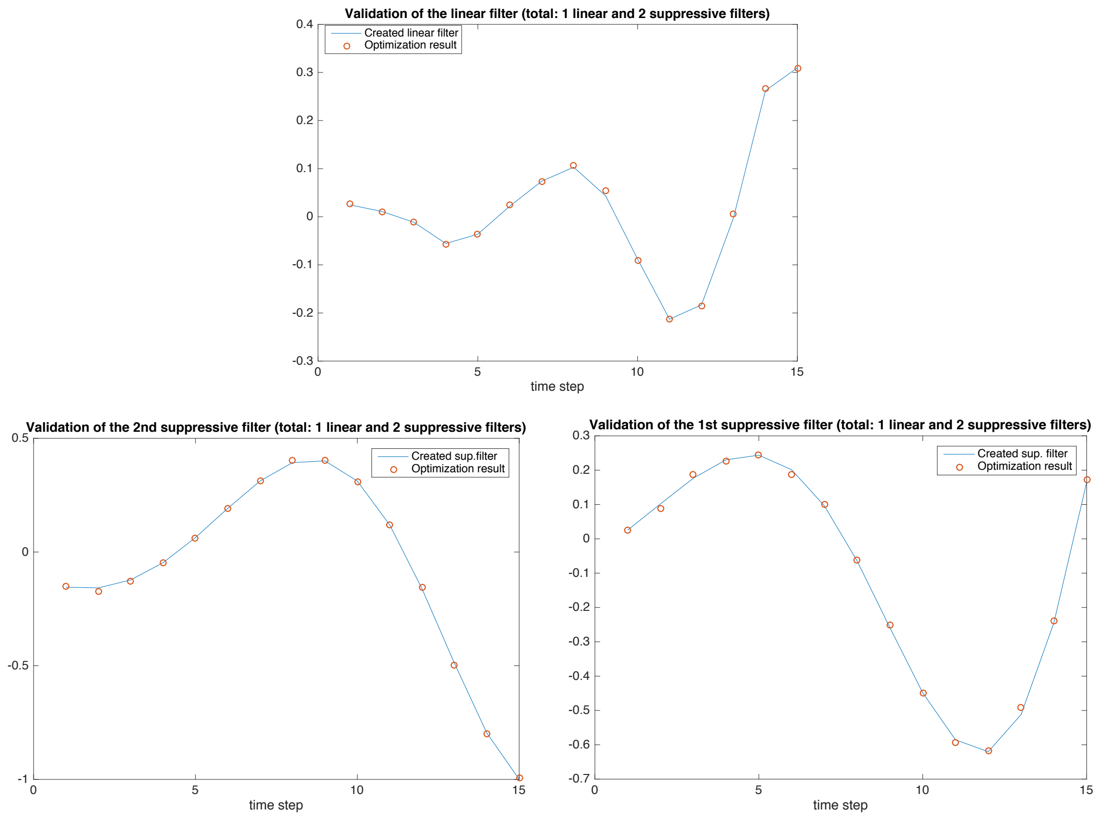


Fig. 26 Comparison of the GQM test filter used in simulating artificial GQM spiking data with the recovered filters from the LL optimization. Two positive quadratic filters were simulated and two positive quadratic filters were assumed in the optimization procedure. In this case the agreement between the test filters and optimal filters is perfect.

First, we test algorithm on two positive quadratic filters and search for two positive quadratic filters. We also create a linear filter similar to that used for GLM tests. When we run the optimization algorithm assuming that there are only two positive filters, the algorithm finds them well.

A more tricky situation is when the LL optimization algorithm does not know in advance how many filters are to search for. For instance, we run a test where we used the same artificial data with two positive filters but ask the algorithm to search for three positive filters. The data shows that the algorithm can be trapped in a trivial local minimum, where it finds two identical positive filters plus an extra one, which is clearly equivalent to finding only two distinct filters. We also note, that searching for both positive and negative filters is tricky, as then tend to cancel each other, and that creates a possibility for multiple false minima. In practice GQM is used to identify either strong excitatory or inhibitory behavior.

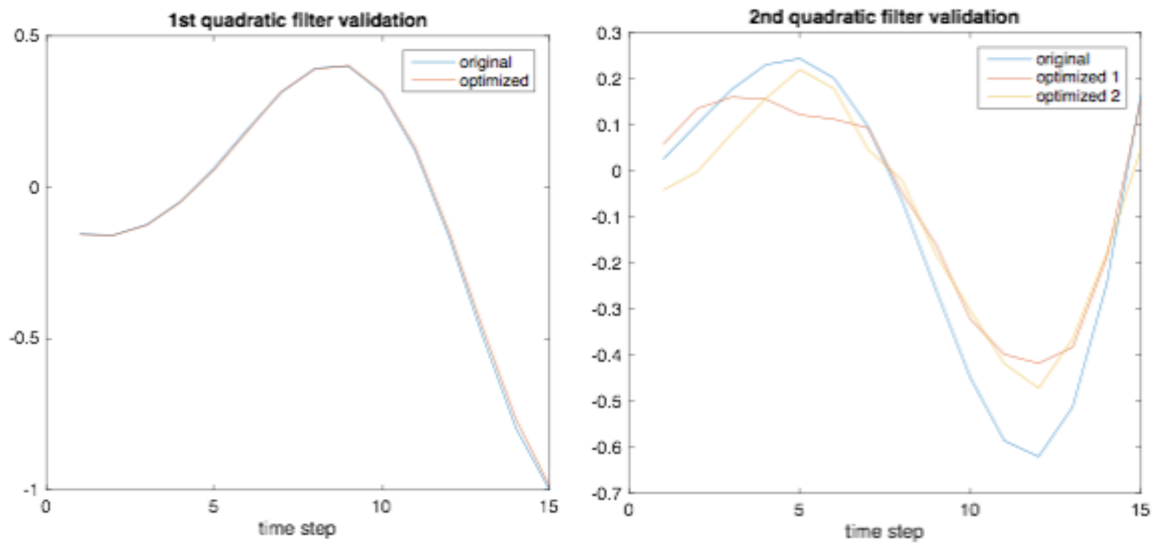


Fig. 27 Comparison of the GQM test filter used in simulating artificial GQM spiking data with the recovered filters from the LL optimization. Two positive quadratic filters were simulated and three positive quadratic filters were assumed in the optimization procedure. One of the filters was identified correctly. The two other filters were identified as nearly matching, and their shape, up to a scaling factor, matches with the test filter. This implies that the algorithm identified that there are only two distinct quadratic filters.

Neither LGN nor RGC data contains features relevant for the GQM, so we will not perform detailed studies of the GQM model applied to these two data sets. We do compare the results of GQM to NIM with an RGC data set later, in the NIM section of the report.

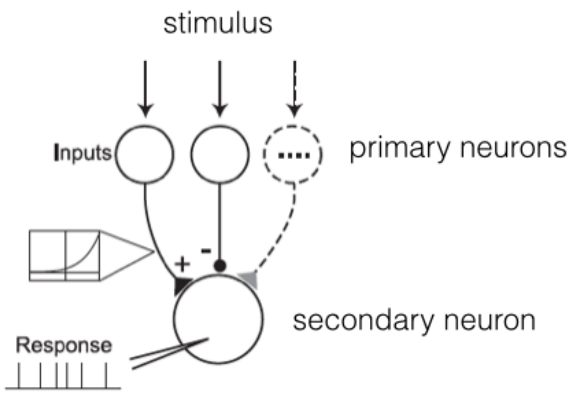
5.3 Non-linear input model (NIM)

The non-linear input model is probably the most interesting of all the models studied in this study. It is based on a fundamentally different principle of non-linear processing of information by neurons which takes into account interaction between neurons. The expression for the NIM's firing rate is given by

(34)
$$r(t) = F(\sum_i \omega_i f(\mathbf{k}_i \cdot \mathbf{s}(t)))$$

where now there are two rectifying non-linear functions, F and f, which will be described below. The model is defined by the components of the filters k and an optional constant b. Just like in GQM, the filters can be negative or positive.

NIM is motivated by the following schematic. Every neuron in reality receives a signal from multiple other neurons. Since the output of each neuron is a non-linear function of its input, like in the GLM model, it is natural to assume that each neuron on average makes a decision to fire based on a combined non-linear input from other neurons. The simplest way to model this would be to say that the argument of the spiking non-linearity F is a sum of a stimulus that was filtered such that it looks like the output of another GLM-type neuron.



For this model we will consider a specific non-linearity in the form of F(x) = f(x) = Log(1+Exp(x)). This function is equivalent to Exp(x) at large negative x and is a linear function at large positive x, which clearly creates a rectifying behavior. The LL-function is given by

$$LL(\Theta) = \sum_t n(t) log(log(1+exp(\sum_i \omega_i log(1+exp(\mathbf{k}_i \cdot \mathbf{s}(t)))+b))) - \sum_t log(1+ exp(\sum_i \omega_i log(1+exp(\mathbf{k}_i \cdot \mathbf{s}(t))) + b))$$

and the gradients are given by

$$\frac{dLL}{dk_{ij}} = \sum_t \left(\frac{n(t)}{r(t)} - 1\right)\left(\frac{1}{1+exp(A)} exp(A)\omega_i \frac{1}{1+exp(\mathbf{k}_i \cdot \mathbf{s}(t))} exp(\mathbf{k}_i \cdot \mathbf{s}(t)) s_j(t)\right)$$

5.3.1 Validation of NIM algorithm using artificial data

Proceeding by analogy with GLM validation, we generate a while noise stimulus signal and create two test filters, and generate spikes according to the NIM firing rate expression. Then we run the NIM model to recover the two filters and use the non-linearity f(x) = F(x) = Log(1+exp(x)).

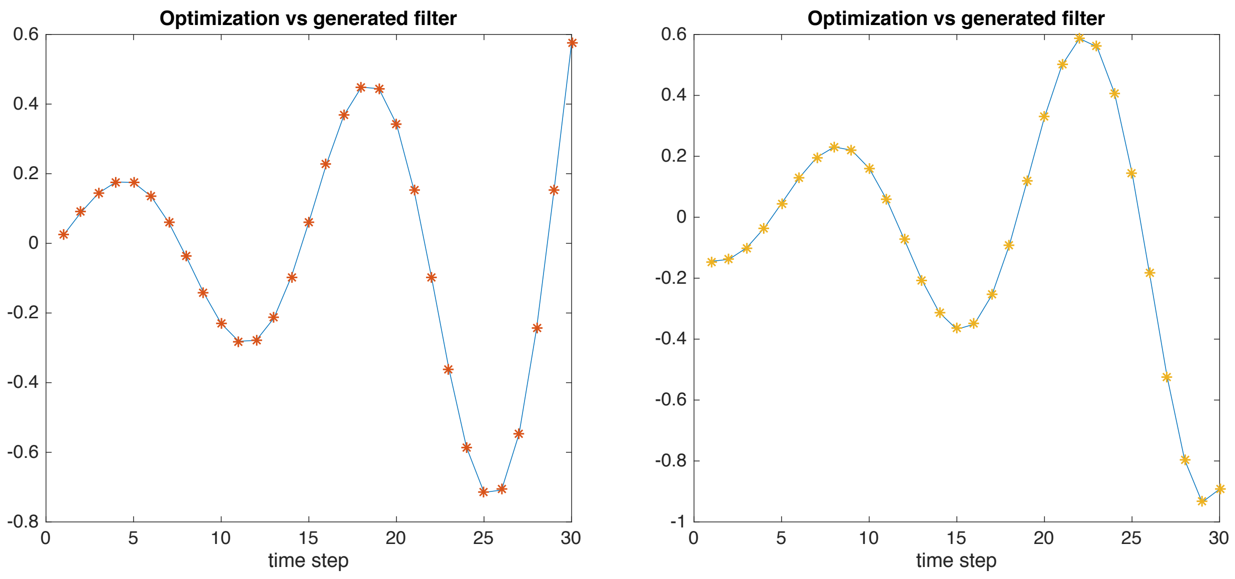


Fig. 28 Two recovered NIM filters plotted vs test filters used to generate artificial data. The agreement is perfect here.

It is useful to check that the specific shape of the non-linear function is not too important for NIM. For instance, we take the same data and recover filters with a modified NIM model where f(x) = 0 if x<0 and f(x) = x, is x>0. The gradients are adjusted accordingly piecewise. We see that the filters agree with the test filters and the agreement is reasonable, although not perfect.

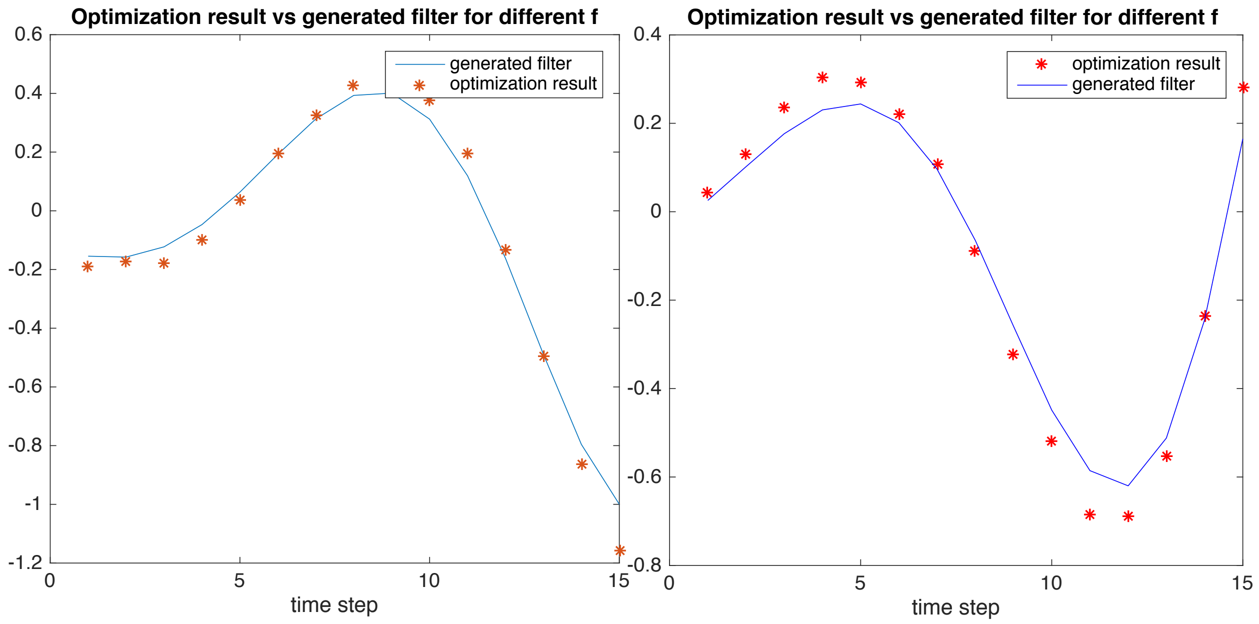


Fig. 29 Two recovered NIM filters plotted vs test filters used to generate artificial data. The agreement is good but imperfect because we modified the upstream non-linearity function compared to the one used to generate synthetic data. This plot shows that the details of the non-linearity is not important as long as it is a rectifying function

5.3.1 Reconstructing NIM model parameters from RGC data

RGC data was generated by NIM model using one positive filters and one negative filter. The algorithm recovered both filters with a very good agreement. The plot below shows the “true” filters (provided to me with the RGC data) together with the optimal filters found by my code

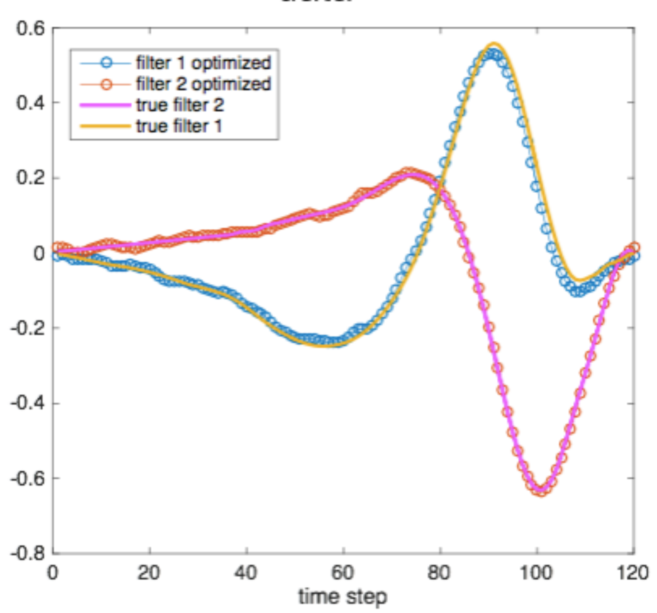


Fig. 30 Identification of two NIM filters from the RGC data

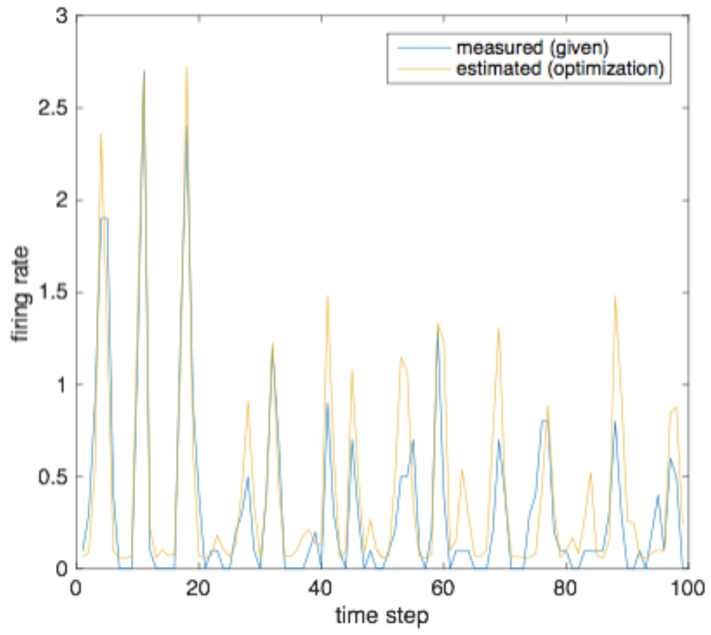


Fig. 31 Comparison of the RGC firing rate with that predicted by the NIM model using the recovered optimal filters. The agreement between the blue and yellow noisy traces demonstrates that NIM indeed is capable of predicting the RGC data.

To further test the predicting power of my NIM algorithm, I have generated the spikes according to the recovered filters and compared them to the RGC spiking data. The agreement is very good, and it can be characterized by the R2 value of 0.679

5.4 Comparison of GLM/GQM/NIM using RGC & GLM data

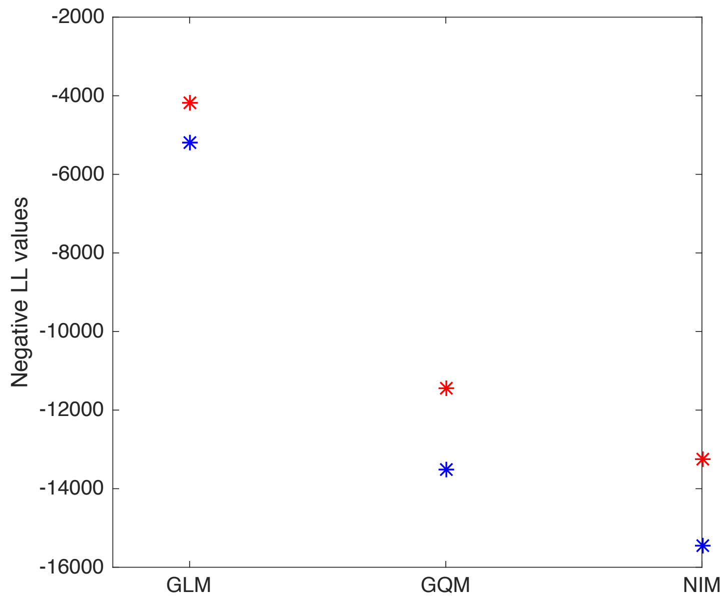


Fig. 33 High (red) and low (blue) bounds on negative Log-likelihood calculated using optimal parameters of GLM, GQM, and NIM models using the RGC data. It is clear that non-linear processing of the stimulus helps to find a better model than a purely linear GLM, but NIM still outperforms GQM presumably due to a more clever scheme of non-linear preprocessing.

Here we have show lower and upper bounds of cross-validated log-likelihood using the optimal filters found for the three models: GLM, GQM, and NIM using the RGC data set. Log-likelihood was calculated in a cross-validated manner: the data was split into two equal pieces. The first piece was used to extract optimal parameters, the second piece was used to calculate the LL. Then the two pieces were switched roles and LL was calculated again. We plot the higher and lower bounds on LL for the three models. It is clear that the two models with non-linear filtering perform better than the linear generalized linear model, and NIM appears to outperform GQM, although fluctuations in the determined values of LL for the two overlap. Applying the same procedure to the LGN data set yields qualitatively similar results. NIM performs better than both GLM and GQM, presumably because it builds on a more natural idea that real neurons receive non-linear input already preprocessed by their partner neurons.

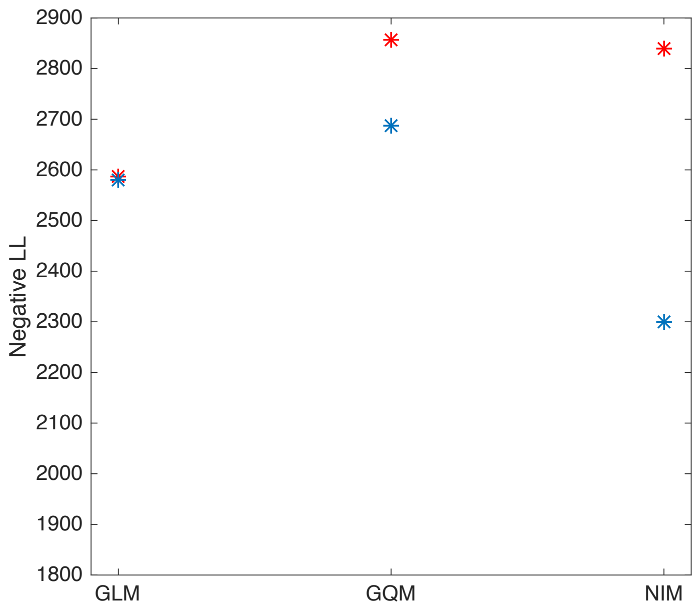


Fig. 34 Same plot as in Fig. 33 but for GLM data set. Here the trend is similar: NIM outperforms the other two models.

6 Conclusion

In conclusion, I have implemented the Linear-Nonlinear-Poisson model with filters estimated by the two moment-based statistical models, the Spike Triggered Average and the Spike Triggered Covariance models. I validated both STA and STC models using the synthetic data set by obtaining the results similar to the previously published research that used the same data set [2]. Next, I reconstructed the LNP model from a real LGN data set and performed cross-validation by comparing the predicted spike rate to the measured one using a different data set from the same neuron. The STA-based model matched the data with a good precision (R-squared approximately 0.77), while the STC-based model resulted in a worse agreement (R-squared approximately 0.54). I suspect the reason for this is that 2D STC-based model requires a much larger data set in order to recover the spiking non-linearity compared to the 1D STA-based model. A longer data acquisition would probably improve the performance of the STC-based model. In the second semester, I have focused on maximum likelihood estimation modes: GLM, GQM, and NIM. I used a synthetic data set, generated by myself, in order to validate optimization algorithms. I then tested all three models on LGN and RGC data sets. For GLM model, I have successfully identified the history behavior of the LGN neuron, where it cannot fire right after it just fired. Inclusion of the history term increased the log-likelihood, indicating the importance of taking into account neuron's refractory period in modeling of its stochastic response. As for the non-linear filter models, I found that NIM model in general works much better than GQM and GLM, on both LGN and

RGC data sets. This is expected because the NIM model takes into account the fact that in a network of neurons, each neuron receives a signal from another one, and this signal is already non-linearly preprocessed. It would be interesting to extent the methods used in this work for large neuron networks which is needed to process interesting information types.

7 Implementation

Hardware

- MacBook Air, 1.4 GHz Intel Core i5, 4 GB 1600, MHz DDR3

Software

- Matlab_R2015b

8 Updated project schedule (12/08/16)

October - November

- Implement Spike Triggered Average (STA) and Spike Triggered Covariance models (STC)
- Test models on synthetic data set and validate models on LGN data set

December - Mid February

- Implement Generalized Linear Model (GLM)
- Test model on synthetic data set and validate model on LGN data set

Mid February - Mid April

- Implement Generalized Quadratic Model (GQM) and Nonlinear Input Model (NIM)
- Test models on synthetic data set and validate models on LGN data set

Mid April - May

- Collect results and prepare final report

Original project schedule (10/01/16)

October - mid November

- Implement STA and STC models
- Test models on synthetic data set and validate models on LGN data set

<u>November - December</u>

- Implement GLM
- Test model on synthetic data set and validate model on LGN data set

<u>January - March</u>

- Implement GQM and NIM
- Test models on synthetic data set and validate models on LGN data set

<u>April - May</u>

- Collect results and prepare final report

Despite some changes to the schedule in the updated version, I am still staying within the time frames for the project, which I set up originally at the meeting with the course instructors in October/2016. I achieved this by decreasing the time for results collection and report preparation from 1.5 month to 1 month. However, I do not think that it should affect the quality of the results collection since most part of it will occur in parallel with testings of my models.

9 Deliverables

At the end of the semester I will provide

- Matlab code for all 5 models (LNP model with STA filter, LNP model with STC filter, GLM,GQM,NIM)
- Reports and presentations
    - Final paper and presentation

10 References

[1] Chichilnisky EJ (2001) A simple white noise analysis of neuronal light responses. Network 12:199 –213.
[2] McFarland JM, Cui Y, Butts DA (2013) Inferring nonlinear neuronal computation based on physiologically plausible inputs. PLoS Computational Biology 9(7): e1003142.

[3] Butts DA, Weng C, Jin JZ, Alonso JM, Paninski L (2011) Temporal precision in the visual pathway through the interplay of excitation and stimulus-driven suppression. J. Neurosci. 31: 11313-27.
[5] Simoncelli EP, Pillow J, Paninski L, Schwartz O (2004) Characterization of neural responses with stochastic stimuli. In: The cognitive neurosciences (Gazzaniga M, ed), pp 327–338. Cambridge, MA: MIT.
[6] Aljadeff J, Lansdell BJ, Fairhall AL, and Kleinfeld D (2016) Analysis of neuronal spike trains, deconstructed. neuron, Volume 91, Issue 2, Pages 221–259
[7] Paninski, L., Pillow, J., and Lewi, J. (2007). Statistical models for neural encoding, decoding, and optimal stimulus design. Prog Brain Res.;165:493-507.
[8] Paninski, L. (2004). Maximum Likelihood estimation of cascade point-process neural encoding models. Network: Computational Neural Systems , 15, 243-262.
[9] Schwartz, O. et al. (2006). Spike-triggered neural characterization. Journal of Vision , 6, 484-507.
[10] Shlens, J. (2014) Notes on Generalized Linear Models of Neurons. Google Research documentation.