

# Neuroinformatics — lab exercises manual

authors:

Eduard Bakstein (TA), Daniel Novák (Lecturer)  
<http://nit.felk.cvut.cz>, Prague 2015

April 23, 2015

## Abstract

This is supportive text for labs of Neuroinformatics course at Czech Technical University in Prague, Faculty of Electrical Engineering. Exercises for each task are provided in Matlab language.

We further recommend a book [3], which is nicely readable. Good summary is provided by [1]. Particular topics are covered by [2], [4] Some advance material is covered by [2], [4].

## Contents

<b>0</b>	<b>Mathematical apparatus</b>	<b>2</b>
0.1	Numerical solution of differential equations . . . . .	2
<b>1</b>	<b>Neuron models</b>	<b>4</b>
1.1	Model of membrane and synapse: simpler case . . . . .	4
1.2	Membrane and synapse modelling 2: Hodgkin-Huxley . . . . .	6
1.3	Spike train modelling . . . . .	8
<b>2</b>	<b>Real data analysis, modelling of neuronal populations</b>	<b>9</b>
2.1	Spiketrain modelling II . . . . .	9
2.1.1	Artificial and real signal - comparison . . . . .	10
2.2	Real data processing: Spike sorting . . . . .	11
2.3	Analysis of real data: Eye movements . . . . .	12
<b>3</b>	<b>Information coding and transfer in the brain</b>	<b>13</b>
3.1	Hebbian learning . . . . .	13
	<b>Bibliography</b>	<b>14</b>

## 0 Mathematical apparatus

We will introduce numerical apparatus, which will be applied during activity modelling of neurons.

### 0.1 Numerical solution of differential equations

As in other sectors dealing with dynamic systems, we will use differential equations throughout this course. Due to the fact that their exact analytical solutions can often be difficult or even impossible, we introduce in this exercise the approximate numerical solutions.

#### Euler's method

Let's consider first order

$$\frac{\delta x}{\delta t} = f(x, t), \quad (0.1)$$

Euler's first order metoda consist of discretization  $\frac{\delta x}{\delta t} = \frac{\Delta x}{\Delta t}$ . Lets' take:

$$\begin{aligned} \Delta t &= t_2 - t_1 \\ \Delta x &= x(t + \Delta t) - x(t), \end{aligned}$$

then we can express eq (0.1) as

$$\frac{\Delta x}{\Delta t} = f(x(t), t)$$

finally

$$x(t + \Delta t) = x(t) + \Delta t f(x(t), t)$$

We can approximate solution by taking into account the slope of the line at that point. However, if the slope is dependent on t, this will be a very rough approximation. In this case, we can refine the solutions by other parameters of the Taylor series according to the formula:

$$x(t + \Delta t) = x(t) + \Delta t \frac{\delta x}{\delta t} + \frac{1}{2} (\Delta t)^2 \frac{\delta^2 x}{\delta t^2} + O, \quad (0.2)$$

where O represents all members of the higher orders. In the second order therefore we model in addition the slope curvature. This approximation should be closer to analytical solutions.

#### Runge-Kutta metoda

In the methods of higher orders we can additionally refine the solution by estimation not in the point  $x$  and  $x + \Delta x$ , but in the middle of this period - the so-called „midpoint method“. The resulting value should be more representative in a given interval and lead to a more accurate estimate. Moreover, if we do not have analytical expression of parameters of the Taylor expansion of higher orders, we can again estimate parameters of high orders using numerical methods.

The actual method of Runge-Kutta numerical integration is the 4th order, which combines estimation in the middle of the interval with a numerical estimate of higher order. It can therefore approximate solution of arbitrary functions. This leads to a higher computational cost, but - as we shall see - leads to very accurate estimates of the solution. Runge-Kutta method is implemented in Matlab function `ode45()`.

**Exercise 0.1** Our task is numerical approximation of the following differential equation

$$\frac{dx}{dt} = t - x + 1, \quad (0.3)$$

initial conditions:  $x(0) = 1$  using Euler's method (1 and 2 order) and Runge-Kutta methods.

The analytical solution of this equation has the form  $x = t + e^{-t}$ <sup>1</sup>. Solve the equation in the interval  $(0, 5)$ , choose  $\Delta t = 0.02$ .

---

<sup>1</sup>You can check e.g. at <http://www.wolframalpha.com/>, question: „solve differential equation x '= ... “

**Task 0.1** (2 b) Plot the solution  $x(t) = f(t)$  for  $t = 0, \dots, 2s$  using Euler's method of first and second order <sup>2</sup> ( $x_{Euler}$ ) and plot into graph along with the analytical solution.

**Task 0.2** (1 b) Solve the task using Runge-Kutta  $x_{Runge}$  and add the solution to the same graph.

*Hints:* function `ode45` has as first parameter a callback to the function, which describes the equation to be solved. In this case the callback can be easily defined using so called *function handle* and *anonymous function*<sup>3</sup> as `ode_func = @(t,x,flag) 1-x+t;`

**Task 0.3** (1.5 b) Plot the dependence of relative error of each numerical methods against analytical solution. For example (for Euler's method)  $(x_{Euler} - x_{exact})/x_{Euler} = f(\Delta t)$

**Task 0.4** (1.5 b) Plot absolute error of each numerical method taking into consideration size of integration step  $\Delta t \in (0.001, 1)s$  in time  $t = 1s$ .

---

<sup>2</sup>when solving the second order Euler method you can proceed in two ways. 1) Derivating analytic function  $f(x, t)$ .  
2) estimate value of  $f'(x, t)$  using the slope  $f(x, t)$  between points  $t$  and  $t + \Delta t$

<sup>3</sup>see for example. [http://www.mathworks.com/help/matlab/matlab\\_prog/creating-a-function-handle.html](http://www.mathworks.com/help/matlab/matlab_prog/creating-a-function-handle.html)

# 1 Neuron models

In this part we will describe properties of action potential and behaviour of neurons.

## 1.1 Model of membrane and synapse: simpler case

Taking into account that nerve impulses propagate through changes in electrical potential on the cell membrane, it seems natural to create the basic model as an equivalent electrical circuit. The simplest model, that we introduce in this exercise, is working only with one type of channel - leakage chloride channels (always opened) - and takes the form of an RC circuit, the resting battery, replacing the resting potential of the membrane (Nernst potential). As we shall see, the response to the input excitation is far away from the real action potential but serve us in understanding the basic concept.

**Exercise 1.1 (RC model)** Model membrane behavior using RC model - see figure 1. Input membrane current  $I_{stim}$  is rectangular signal  $10pA$  lasting  $20ms$ . It is necessary to convert the stimulating current  $I_{Cl}$  and  $I_C$  to the electrode surface  $A^4$ . The electrode is stimulating and at the same time recording the membrane current (expressed in  $cm^2$ ). Hence  $I'_{stim} = I_{stim}/A \approx 10^{-11} \cdot 10^6 \approx 10^{-5}$ . The membrane parameters are the following:

- capacity:  $C_m = 1 \mu F/cm^2$ ,
- conductance:  $g_{Cl} = 0.3 ms/cm^2$ ,
- time constant :  $\tau = C_m/g_{Cl}$ ,
- membrane surface :  $A \approx 1 \cdot 10^{-6} cm^2$
- Nernst potential of Cl:  $V_{Cl} = -68 mV$ ,
- initial conditions:  $V(0) = -68 mV$ ,  $I_{Cl}(0) = 0 \mu A/cm^2$ ,  $I_C(0) = 0 \mu A/cm^2$ .

**Task 1.1** (2 b) Plot dependence  $V(t)(= \phi_{in} - \phi_{out})$ ,  $I_{Cl}(t)$ ,  $I_C(t)$  for these time interval  $(0, 40)ms$ ,  $\Delta t = 0.01ms$

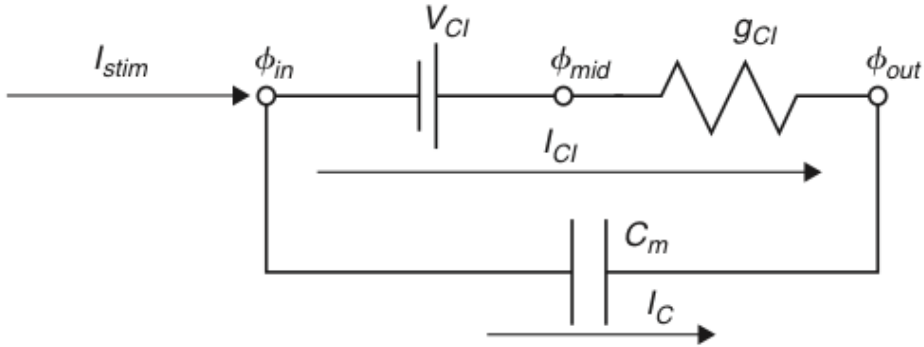


Figure 1: Membrane model with Cl leakage channel

Hence

$$I_C(t) = C_m \frac{dV}{dt}(t) \tag{1.1}$$

$$I_C(t) = \frac{I_{stim}(t)}{A} - I_{Cl}(t) \quad \rightarrow I_{stim} = A \cdot I_C(t) + A \cdot I_{Cl}(t) \tag{1.2}$$

$$I_{Cl}(t) = g_{Cl}(V(t) - V_{Cl}) \tag{1.3}$$

$$\tau = \frac{C_m}{g_{Cl}} \tag{1.4}$$

<sup>4</sup>This is current density when we should use the following convention  $g$ . However, we will stick to the terminology used during lectures  $I$ .

Combining the above equation we get

$$C_m \frac{dV}{dt} = \frac{I_{stim}(t)}{A} - g_{Cl}(V(t) - V_{Cl}) \quad (1.5)$$

$$\tau \frac{dV}{dt} = V_{Cl} - V(t) + \frac{I_{stim}(t)}{Ag_{Cl}} \quad (1.6)$$

$$(1.7)$$

Euler's method (forward)

$$\tau \frac{V(j) - V(j-1)}{dt} = V_{Cl} - V(j-1) + \frac{I_{stim}(j-1)}{Ag_{Cl}} \quad (1.8)$$

$$V(j) = V(j-1) + \frac{dt}{\tau} [V_{Cl} - V(j-1) + \frac{I_{stim}(j-1)}{Ag_{Cl}}] \quad (1.9)$$

Euler's method (backward)

$$\tau \frac{V(j) - V(j-1)}{dt} = V_{Cl} - V(j) + \frac{I_{stim}(j)}{Ag_{Cl}} \quad (1.10)$$

$$V(j)(1 + \frac{dt}{\tau}) = V(j-1) + dt(\frac{V_{Cl}}{\tau} + \frac{I_{stim}(j)g_{Cl}}{Ag_{Cl}C_m}) \quad (1.11)$$

$$V(j) = \frac{V(j-1) + dt(\frac{V_{Cl}}{\tau} + \frac{I_{stim}(j)}{AC_m})}{(1 + \frac{dt}{\tau})} \quad (1.12)$$

**Exercise 1.2 (EPSP model)** The task is analysis of a model depicted in figure 2. This model incorporates so called „excitatory postsynaptic potential“ (EPSP) simulating behavior of membrane dendrite of the postsynaptic's neuron after receiving excitation. Synapse is modelled by chanchagable conductance  $g_{syn}$ . In other words: the model contains additionally channels, which react to release to the neurotransmitter (literally they are opening, hence their conductance is decreasing - see result of eq (1.13)) with time constant  $\tau_{syn} = 1 \text{ mS}$ . The parameters are the same like in the previous case. Stimulation current  $I_{stim} = 0$ ,  $V_{syn} = 10 \text{ mV}$ . In time  $t = 1 \text{ ms}$  the neurotransmitter is released, hence  $g_{syn}(1 + \delta) = 1$ . Initial conditions  $V(1) = 0$ ,  $I_{syn} = 0$ ,  $g_{syn}(1) = 0$ ,  $g_L = 1$ .

$$\tau_{syn} \frac{dg_{syn}(t)}{dt} = -g_{syn}(t) + \delta(t - t_{pre} - t_{delay}) \quad (1.13)$$

**Task 1.2** (0 b) Plot dependence  $V(t)$ ,  $I_{Cl}(t)$ ,  $I_C(t)$ ,  $I_{syn}(t)$ . Explain trends and compare them with the previous task 1.1.

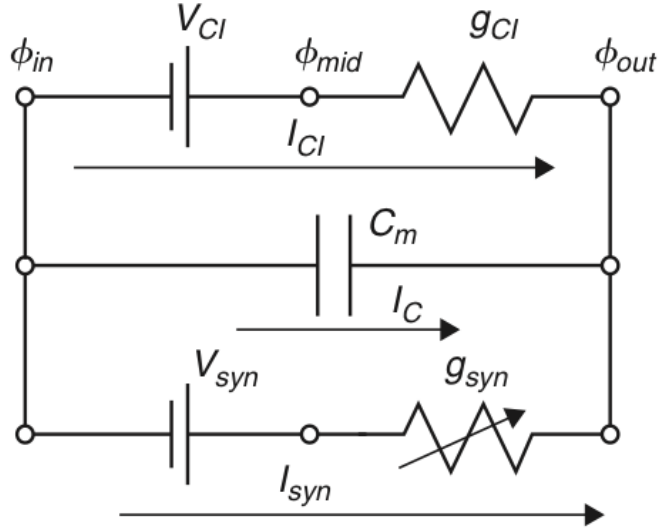


Figure 2: EPSP: Synapse model with leakage Cl channel

## 1.2 Membrane and synapse modelling 2: Hodgkin-Huxley

**Exercise 1.3 (Hodgkin-Huxley)** The task is analysis of Hodgkin-Huxley(HH) model depicted in figure 3. Compared to the previous models, HH incorporates variables which are dependent both on time model and voltage, hence  $Na^+$  a  $K^+$  channels are controlled by voltage. The model is based on the following set of equations

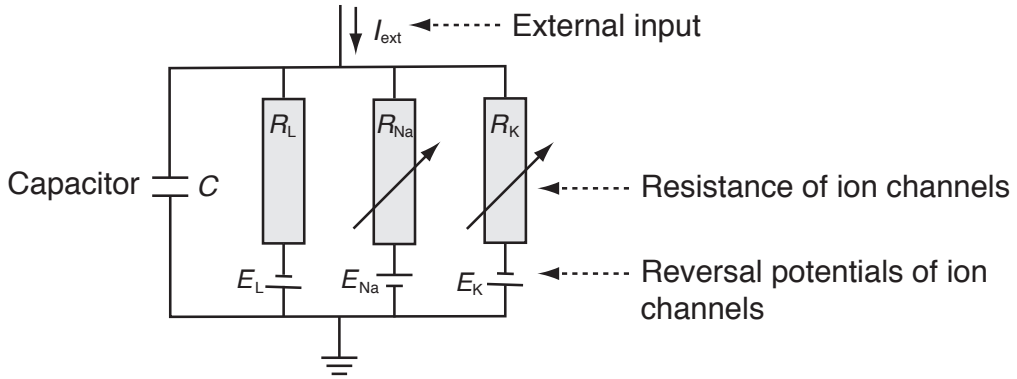


Figure 3: Hodgkin-Huxley model. Each branch represents resting potential and resistivity of a particular channel:  $E_L$ ,  $R_L$  - channels always opened („Leakage channels“),  $E_{Na}$ ,  $R_{Na}$  - sodium channels controlled by voltage,  $E_K$ ,  $R_K$  - potassium channels controlled by voltage.

Current in ion channels can be described by Ohm's law:

$$I_{ion} = \hat{g}_{ion}(V - E_{ion}) \quad \left( = (V - E_{ion}) / \hat{R}_{ion} \right), \quad (1.14)$$

where  $\hat{g}_{ion}$  is the maximum conductance of ion channel

Next, we introduce auxiliary variables dependent on voltage and time  $n(V, t)$ ,  $m(V, t)$ ,  $h(V, t)$ , resulting in the following conductances:

$$\hat{g}_K(V, t) = g_K n^4 \quad (1.15)$$

$$g_{\hat{N}a}(V, t) = g_{Na} m^3 h \quad (1.16)$$

Combining eq.1.16-1.14 we get

$$C \frac{dV}{dt} = -g_K n^4 (V - E_K) - g_{Na} m^3 h (V - E_{Na}) - g_L (V - E_L) + I_{ext}(t) \quad (1.17)$$

Next, we define time constants

$$\tau_n(V) \frac{dn}{dt} = -[n - n_0(V)] \quad (1.18)$$

$$\tau_m(V) \frac{dm}{dt} = -[m - m_0(V)] \quad (1.19)$$

$$\tau_h(V) \frac{dh}{dt} = -[h - h_0(V)] \quad (1.20)$$

For each variable we get

$$\frac{dx}{dt} = -\frac{1}{\tau_x(V)} [x - x_0(V)], \quad x \in \{n, m, h\} \quad (1.21)$$

and after substituting Euler's numerical integration we get

$$x(t + \Delta t) = \left(1 - \frac{\Delta t}{\tau_x}\right) x(t) + \frac{\Delta t}{\tau_x} x_0. \quad (1.22)$$

Initial conditions hold:

$$x(0) = \frac{\alpha}{\alpha + \beta}, \quad \tau_x = \alpha\beta, \quad x \in \{n, m, h\} \quad (1.23)$$

$$\alpha_n = \frac{10 - V}{100 \left(e^{\frac{10-V}{10}} - 1\right)}, \quad \beta_n = 0.125e^{-\frac{V}{80}} \quad (1.24)$$

$$\alpha_m = \frac{25 - V}{10 \left(e^{\frac{25-V}{10}} - 1\right)}, \quad \beta_m = 4e^{-\frac{V}{18}} \quad (1.25)$$

$$\alpha_h = 0.07e^{\frac{V}{20}}, \quad \beta_h = \frac{1}{e^{\frac{30-V}{10}} + 1} \quad (1.26)$$

**Task 1.3** (0 b) Download model implementation file `cv3_HH.m` from course website. Visualize time output of the model (membrane voltage  $V$ ), which is excited by the constant current  $I_{ext} = 60 \mu A/cm^2$  (default settings). Implementation works from time -30s to get steady system. Plot output for time 0 – 100ms.

**Task 1.4** (1 b) Expand visualization by adding time trend of conductance  $g$  for each channel type (describe different time series using `legend`)

**Task 1.5** (2 b) Visualize dependence of firing frequency on external constant current  $I_{ext}$  - so called *activation or transfer function*. Set the input current  $I_{ext}$  in the range 0 – 15  $\mu A/cm^2$ . You should compute firing frequency after reaching the steady state (hint: use `fce diff`, `find alt`, `findpeaks`). Which shape does the activation function resemble? For which values of the current  $I_{ext}$  will the shape of the activation function change significantly (perform a discussion)?

**Task 1.6** (2 b) Adding noise to the input current will change behaviour of the model. Visualize membrane voltage  $V$  and activation function as a response to input current with added noise. Use  $I_{ext} = 30 \mu A/cm^2$  and white noise (`randn`) with the standard deviation 60 and zero mean, for plotting activation function use  $I_{ext} = \langle 0, 15 \rangle \mu A/cm^2$ , noise with standard deviation 30. How did the resulting output change? Try out for different noise levels.

**Task 1.7 (bonus)** (1 b) Plot ISI histogram for the input current  $I_{ext} = 30 \mu A/cm^2$  and noise with standard deviation 60. Compare to the case with zero noise and discuss the results.

### 1.3 Spike train modelling

We showed in the previous tasks, how does noise on the input affect temporal and frequency characteristics of a neuron. Noise signal on neuron input is much more realistic than constant current - mainly due to the fact that neuron input is formed by a summary signal from many synapses of pre-synaptic neurons, connected to its dendrites. In this section, we will simulate such signal and use it as input to a neuron model.

You have learned in the lectures, that interspike intervals (ISI) of typical cortical neurons exhibit log-normal distribution. Before we start modelling spiketrains, let us verify this on a model.

**Exercise 1.4 (LIF neuron model and Poisson spiketrain)** The complex models from previous exercises, such as Hodgkin-Huxley, are able to model the action potential shape based on physiological modelling of ion channels' behavior. If we want to model more complex neural networks, even simpler models (such as Wilson model) are overly complex for such application. When modelling neural networks, the accurate shape of action potential is not as important (we will see it can be simply modelled using  $\alpha$  functions) as the spike timing. For these purposes, the *Leaky integrate and fire* neuron is often used. The basics of this model includes simple integrator, which is reset to resting potential when a preset threshold is exceeded.

**Task 1.8** (1 b) Download LIF model from the course website - `cv4_lif.m`. Show model output for constant input current (default setting). Plot output voltage, along with spike times (= reset times) over time. Compute firing rate and display an ISI histogram.

**Task 1.9** (2 b) Modify previous task by applying input noise with mean 12 and standard deviation 100, instead of constant current (function `randn`). Simulate for times in the range  $0 - 10^5$  and integration step  $dt = 0.01$ . Set bin number to 60 when computing histograms. Convert histograms to probability density function estimates ( $\sum pdf = 1$ ).

**Task 1.10** (3 b) Fit lognormal distribution to the data from the previous task. The lognormal distribution is given by:

$$pdf^{lognormal}(x; \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\log(x)-\mu)^2}{2\sigma^2}} \quad (1.27)$$

One possible solution would be to fit the analytical pdf to ISI histogram from the previous task. This, however, would be completely wrong, as one point in the histogram does not mean one, but many data points (think about this for a while!).

Thus, we will use the function `lognfit`, which estimates lognormal distribution parameters using maximum likelihood estimates. Based on these estimates, calculate the lognormal pdf and display along with the histogram. (You will probably have to adjust the bin number to get a good representation)



## 2 Real data analysis, modelling of neuronal populations

In this block, we will play with analyzing of real  $\mu EEG$  (*micro-EEG*) data, recorded from brains of Parkinson's disease patients. We will compare the data to our simulations from the previous block and we will analyze correlation with other biosignal - the electrooculogram (EOG).

Further in this section we will simulate more complicated network of artificial neurons and (so called *spiking network*) and analyze its behavior under varying conditions.

### 2.1 Spiketrain modelling II

**Exercise 2.1 (Lognorm spiketrain)** We showed in the last exercise how does LIF neuron model respond to a noisy input. In this task, we will show that real signal, produced as a combination of presynaptic potentials, really resembles gaussian noise in many ways.

You know from the lectures, that the *Poisson spiketrain*, - a spiketrain where the number of events per unit time follows poisson distribution - has exponential distribution of interspike intervals (ISIs). Further, the physiological constraints prevent neurons from firing earlier than after the *refractory period* - a fact we have to incorporate in our implementation.

In this example, we will generate a set of spiket trains using exponential distribution, taking refractory period into account, and mix them to create an artificial *muEEG* signal.

**Task 2.1** (1 b) Generate 100 000 ISI intervals from the exponential distribution with mean  $E[X] = 50ms$  (hint: use `expnrnd` function). Display ISI histogram.

Remove all ISI values lower than the refractory period of 1ms. Display ISI histogram and compare to the one before refractory period filtering. How many spikes were removed? Compute mean firing rate and coefficient of variation before and after filtering and compare.

Convert the resulting ISI intervals to a spiketrain (logical vector with ones at spike positions hint: `cumsum`). Use time step of  $dt = 0.1ms$ . Check for errors by plotting the result

**Task 2.2** (1 b) To convert binary spiketrain to time course, we will use convolution with  $\alpha$ -function, which will play the role of spike template.

Generate time course of the  $\alpha$ -function according to:

$$I(t) = c \cdot t \cdot e^{-\frac{t}{t_{peak}}} \quad (2.1)$$

$$c = \frac{g_{peak}}{t_{peak}} e^{-1} \quad (2.2)$$

where  $g_{peak} = 0.5$ ,  $t_{peak} = 1ms$ . Compute the function values for times in the interval  $t \in \langle 0, 30 \rangle$ , the time step should be again  $dt = 0.1ms$ . Plot the output. After assessing the result, you may reduce the time range if appropriate.

Convolve the alpha function with Poisson spiketrain from the previous task (`conv`). Check the result again by plotting.

**Task 2.3** (1 b) Divide the time course from previous task into 1000 signals of the same length (`reshape`). In this task, we will simulate mixing of presynaptic potentials (represented by the simulated neurons), at dendrites. The resulting summary signal will be given by a linear combination of the individual spiket rains. Coefficients will be given by weights  $w$ , which will be modelled as random from exponential distribution with  $\lambda = 1$ .

In order to have better control of the amplitude of the resulting signal, we will normalize the weights so that their sum equals a constant  $k$ . Set the value  $k = 100$ .

Combine the 1000 spiket rains into one signal, using a linear combination with weights  $w$ . You can represent the linear combination as matrix multiplication (elegant + much faster in Matlab).

Compare the resulting signal to gaussian noise (time and distribution). What are the key differences?

**Task 2.4** (1 b) Apply the signal from previous task to the input of LIF neuron from Exercise and observe the output. What changes do you observe when you change the normalizing constant  $k = 150$ . What does the ISI histogram look like, now? Investigate and comment behavior of the LIF neuron.

### 2.1.1 Artificial and real signal - comparison

We have now generated artificial signal as a combination of Poisson spiketrains, convoluted with alpha function. In these tasks, we will compare the simulated signal to real  $\mu EEG$ , recorded from basal ganglia of a Parkinson's disease patient during deep brain surgery. We will show, that this simple method of data generation is not far from reality.

**Task 2.5** (1 b) Use artificial signal generated in the previous task. It is a time course with sampling rate  $f_s = 10kHz$ . Download  $\mu EEG$  data from the course website. The .mat file contains variable `realmEEG` with 1s recording of activity around Thalamus ( $f_s = 24kHz$ ). Display both signals in parallel plots (with correct time coordinates). Investigate both signals.

**Task 2.6** (1 b) From visual comparison of time courses of real and simulated data, we get a rough idea of how similar these two signals are. Your next task is to try out some quantitative comparisons. Choose and implement appropriate methods of signal comparison. You can use e.g.: frequency spectrum, amplitude distribution etc.

## 2.2 Real data processing: Spike sorting

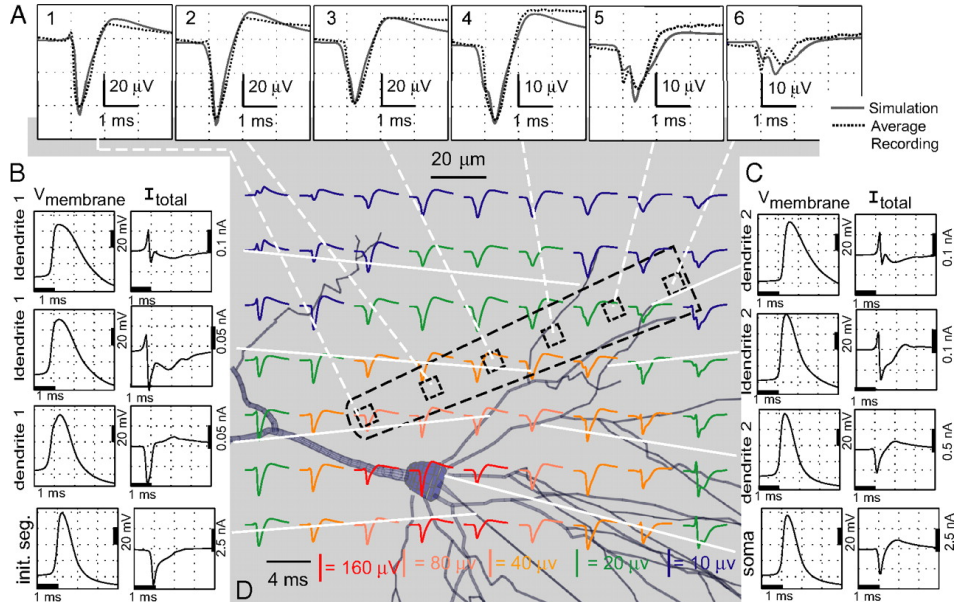
**Exercise 2.2 (Spike sorting)** In this exercise we will focus on spike detection and spike sorting (i.e.) in real  $\mu$ EEG data.

### Spikes and background noise

The real  $\mu$  EEG data we will use in this exercise were recorded using microelectrodes placed in the brain of patients suffering from Parkinson's disease, specifically in the area of the thalamus. As we know, the recorded signal consists of activity of a large number of neurons in the vicinity of the electrodes - this is source of a background noise<sup>5</sup> - from which we can get an approximate idea of the activity of the population in the area. However, we are usually interested in neurons in the vicinity of the electrode - here we can recognize individual action potentials of individual neurons and thus accurately analyze their activity. The recorded AP's (*spikes*) from near neurons have higher amplitude and we are usually able to distinguish them from the background - this procedure is called *spike detection*. Then, to assign individual spikes to different neurons we need a method called *spike sorting*, which usually uses unsupervised learning techniques based on the spike shape.

### Spike sorting

Suppose we have a  $\mu$ EEG signal with detected spikes. Our goal is to classify spikes to individual neurons. Due to the spatial arrangement of different neurons relative to the electrode (plus possibly their individual characteristics) the shapes of recorded spikes originating in different neurons differ. This can be used to sort spikes on the basis of their shape. Given that the correct solution is unknown and we can not know (the number of neurons in the vicinity of the electrode can not be detected by the available imaging techniques, as well as correct assignment of individual spikes to the neurons), we need to use unsupervised learning methods, namely clustering. Regarding the choice of features describing the shape of individual spikes, we have a great range of options, including amplitude, duration, as well as various transformations - such as wavelet, fourier and many more.



Dependence of AP's shape on location to electrodes. <sup>6</sup>

In this task we will use for spike detection a method designed by R.Q. Quiroga that is available from WaveClus toolbox<sup>7</sup>. The method detects each AP based on amplitude thresholding. The threshold is determined by the number of standard deviation. Furthermore, signal is filtered by band-pass filter in range 300 – 3000 Hz. You can download a script `cv7_amp_detect.m`.

<sup>5</sup>Frequently the noise carries the majority of the signal energy

<sup>7</sup>source code is available here [http://www.vis.caltech.edu/~rodri/Wave\\_clus/Wave\\_clus\\_home.htm](http://www.vis.caltech.edu/~rodri/Wave_clus/Wave_clus_home.htm). Those who are interested can find more information here, specially in „Introduction“)

**Task 2.7** (1 b) Download `cv7_spikesorting_data` and load it into the MATLAB. Sampling frequency is again  $fs = 24kHz$ . Plot the data and roughly estimate the spikes position. How long is record (in seconds)?

Download a script for spike detection `cv7_amp_detect.m`. Launch the script using microelectrode data and plot the time series along with spikes detected with the threshold used for detection (output variable is `index` and threshold is `thr`). .

**Task 2.8** (1 b) Variable `spikes` returns spikes detected in rows sorted by max value. Plot a few first spikes and estimate the number of neurons. Design at least one feature that will be used for spikesorting. Set manually threshold and sort the spikes according your feature. Plot the average spike for each group.

**Task 2.9** (4 b) Now just imagine the task when we need to sort the signal without manually setting of the threshold. Suggest and calculate at least one more feature. Plot the spikes in feature space and color them based on your sorting from the previous task. Was your threshold calculated well?

Cluster your features by k-means algorithm (function `kmeans` from Statistics Toolbox). Estimate the number of clusters and show results in feature space. Next, show all signals and average signal for each cluster.

Analyse clustering results. Do you consider your features as sufficient? Is there any possible improvement?

## 2.3 Analysis of real data: Eye movements

**Exercise 2.3 (Eye movements)** In this exercise, we will show how to analyze reaction of single neuron to a stimulus. The data comes from experiments, where we evaluate connection between eye movements and neuronal activity in human Basal ganglia. If the activity of given neuron correlates with eye movements, we should see a change in its firing pattern when eye movements are present. The data come from ten repetitions of the following experiment: The subject is looking at a PC screen, which is black at the beginning. At given time, a white cross appears and the task of the subject to focus at it. Electrooculogram (EOG) is recorded throughout the experiment using electrodes placed near subject's eyes, as well as micro EEG activity in the area of basal ganglia. Once the experiment is over, the data goes through spike detection and spike sorting and neurons, correlating with EOG are identified.

**Task 2.10** (2 b) Download exercise data from the course website. Data contain recording of activity from single neuron in ten repetitions of the same experiments + corresponding EOG. The variable `firingTimes` contains spike times of given neuron, relative to the time of stimulus (cross appears). Variables `eogTimes` and `eogVals` specify corresponding EOG signal for each repetition.

*Note:* plotting of EOG and corresponding spike times can be done as follows:

```
plot(eogTimes{1},eogVals{1});
hold on;
stem(firingTimes{1},ones(size(firingTimes{1})))
```

Plot firing of the neuron in time, aligned with the stimulus (time 0). What can we say about the activity? How many times did the neuron fire during all repetitions before and after the stimulus?

**Task 2.11** (4 b) Use an appropriate way to aggregate activity over all repetitions. (e.g. number of spikes in time windows, instantaneous firing rate, etc.). How does the activity change right after the stimulus? Is there some trend even before the stimulus? How long does it take for the activity to drop and how long to return to original level? Be creative!

### 3 Information coding and transfer in the brain

In this block, we will model some of the processes taking part in transferring and storing information in the brain. We will go through the basics of cognitive modelling and we will try out the so-called *Self-organizing maps*.

#### 3.1 Hebbian learning

All simulations that we did until now were using fixed synaptic weights - such approach ignores the *synaptic plasticity*, i.e. the adaptation of neuron's input weights. One model for weight adaptation is based on correlation of pre-synaptic and post-synaptic firing and is based on the theory by Donald Hebb.

**Exercise 3.1 (Hebbian learning)** This exercise will be based on the codes implemented for 2.1 („Lognorm spiketrain“), which will be adapted to include synaptic weight adaptation based on Hebb's learning. Simply put, the theory states that synapses, that fired shortly before the post-synaptic spike was emitted, are strengthened, while those firing after the post-synaptic spike are weakened.

The weight adaptation will be based on the following equation:

$$w_{n+1} = w_n + \delta w - f_{decay}, \quad (3.1)$$

where  $w_n$  are actual synaptic weights,  $f_{decay}$  is forgetting constant and  $\delta w$  is weight change computed according to:

$$\delta w = \alpha_{learn} \cdot pm \cdot e^{-pm \frac{\Delta t}{\tau}} \quad (3.2)$$

$$\Delta t = t_{post} - t_{pre} \quad (3.3)$$

$$pm = \text{sign}(\Delta t) \quad (3.4)$$

**Task 3.1** (3 b) Implement function for weight adaptation based on 3.2. Let the parameters be  $\tau = 40$ ,  $\alpha = 1$ , display the function in the range  $(-100, 100)ms$  (do not forget axis labels). Comment the result.

**Task 3.2** (7 b) In this task, we will use code for LIF neuron, fed with a large number of pre-synaptic spiketrains from exercise 2.1. Implement code for Hebbian weight adaptation. All synaptic weights will be adapted every time the LIF neuron fires. Use constants from the previous task, set forgetting constant to  $f_{decay} = 0.01$  and set initial sum of weights to  $k = 600$ .

Display time course of the summary pre-synaptic potential in time, including post-synaptic spikes, generated by the LIF neuron. Display the changes of weights over time (you can use e.g. `imagesc` function). How do the weights change over time? What is the change during the first 500ms? And during the following time? What effect does the value of forgetting constant have? Experiment with parameter settings and comment the results.

## Bibliography

- [1] David Fitzpatrick William C. Hall Anthony-Samuel LaMantia Leonard E. White Dale Purves, George J. Augustine. *Neuroscience*. Sinauer Associates, Inc., 5th. edition edition, 2011.
- [2] Michael L. Hines Nicholas T. Carnevale. *The Neuron Book*. Cambridge University Press, 2006.
- [3] Thomas Trappenberg. *Fundamentals of Computational Neuroscience*. Oxford University Press, USA, June 2010.
- [4] Werner M. Kistler Wulfram Gerstner. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.