LUND UNIVERSITY

EITN65

Measurement and Modeling of the Central Nervous System Function

Project 1: Receptive fields

Background

Our skin harbors a variety of somatic sensory receptors, which are specialized on providing information on, e.g., touch, vibration, pressure, stretching of the skin, temperature and pain to the central nervous system. More specifically, these receptors are somatosensory neurons of the peripheral nervous system, and they provide information to neurons in the somatic sensory cortex.

By making a single-unit electrophysiological recording from a neuron in the somatic sensory cortex it is possible to define its receptive field – the region in sensory space (e.g., the body surface) within which a specific stimulus (e.g., touch) elicits the greatest action potential response in that neuron, see Figure 1 below. In this project, you are going to study and simulate receptive fields, using previous MATLAB scripts as a starting point. You can then for example study the concept of two-point discrimination.

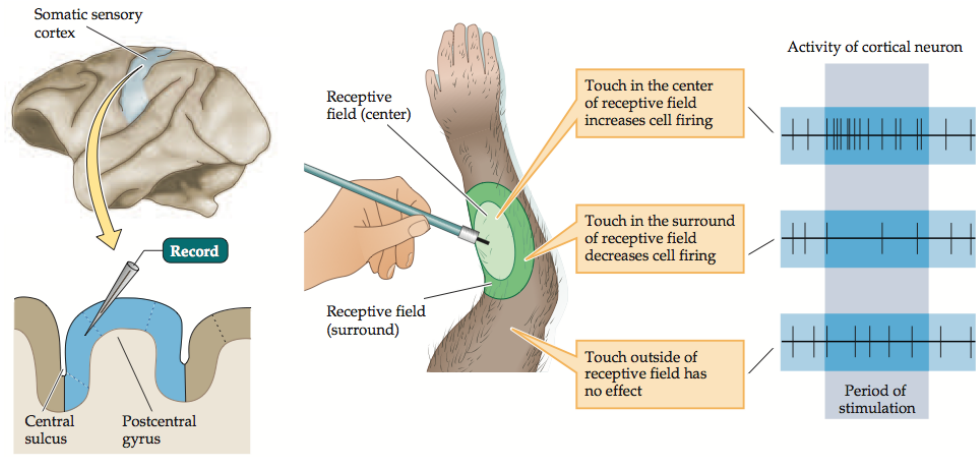

Figure 1 Single-unit electrophysiological recording from a neuron in the somatic sensory cortex, showing the firing pattern in response to a specific peripheral stimulus. (Left) Typical experimental set-up. (Right) Defining neuronal receptive fields. ("Neuroscience" 3rd edition, edited by D. Purves et al.)

Contact: Ulrike Richter, ulrike.richter@med.lu.se

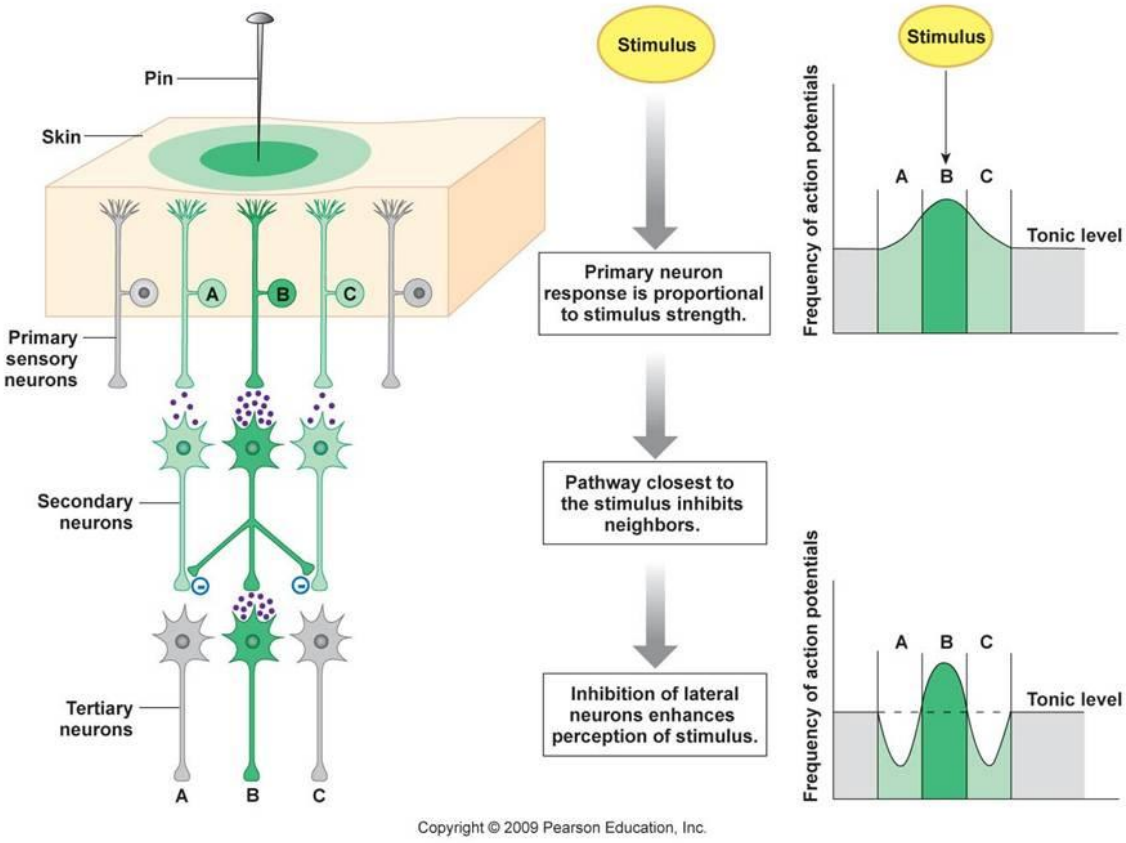

Figure 2 The effect of lateral inhibition The *left-hand side* shows the somewhat simplified three-neuron delay via which mechanosensory information reaches the brain. Note that in reality, a secondary neuron cannot both excite and inhibit tertiary neurons: the inhibition would take place via inhibitory interneurons. The *right-hand side* shows the spatial spread of neuronal activation at the level of the primary sensory neurons (top) and the level of the tertiary neurons (bottom).

Literature

Parts of book chapters: "Neuroscience" (ed. Purves et al.): Chapter 1 "Studying the Nervous Systems of Humans and Other Animals" and Chapter 8 "The Somatic Sensory System"
Review article with more in-depth information: E.A. Lumpkin & M.J. Caterina, "Mechanisms of sensory transduction in the skin", Nature 445: 858–865, 2007.
Inspirational article: L.L. Bologna et al., "A closed-loop neurobotic system for fine touch sensing", J. Neural Eng. 10: 1–16, 2013.

Hand-in assignment 5

As a preparation for the project work, you are to do the following tasks/reach the following milestones and hand in your results **no later than Thursday 28/2/2019 at 12:00**. In contrast to the other hand-in assignments, in which you handed in a report individually, this assignment is to be written and handed in group-wise by the project-groups. Both/all members of the groups need to hand in the report.
Keep your answers brief and don't forget that the purpose of this assignment is to help you get started working on the project. After solving the assignment, you should have all the most essential tools ready to do the project-work.

Project outline
Take help from reviewing the literature to get a better understanding of the biology behind this project, and also to get ideas of what could be interesting questions to pursue. Then, write a short (maximum 250 words) outline of your project. What are your aims with the project and how do you intend to go about to reach them? Why do you think this is an interesting and relevant project?

Modelling assumptions
For the sake of simplicity, we are going to make the following assumptions in our model:

- Let us assume that our primary neurons are slowly adapting mechanoreceptors, and that we want to perform our simulation for the time after the firing rate adaptation has been completed.
- Let us furthermore assume 1:1 excitatory connections from the primary to the secondary neurons, just as shown in Fig. 2, and that an action potential fired in primary neuron A reliably will result in an action potential in secondary neuron A. Thus, we basically assume identical firing patterns in the primary and the secondary layer, which therefore allows us to skip modelling the primary neurons explicitly. Instead, for a certain choice of applied pressure on the skin, we can directly go to simulating spike trains for the secondary neurons.
- Let us also assume that we have a small patch of skin, in which mechanoreceptors are embedded in a very regular fashion, just like a matrix. For a 5 x 5 matrix, we can number these neurons from 1 to 25 (see table below). We can use these number to refer to the primary neurons (i.e., the mechanoreceptors) at a certain spot of the skin, and at the same also to the secondary neurons (due to the previous assumption).

1	6	11	16	21
2	7	12	17	22
3	8	13	18	23
4	9	14	19	24
5	10	15	20	25

Programming milestone 1: Simulate the activity (i.e., the spike train) of a secondary neuron in the presence of input from the periphery

Take the script intfire.m from hand-in exercise 2 as a starting point. Rename it to simulateSecondaryNeuron.m and make the following changes:

1. In Section 1, change `tstop` such that the simulation runs for 1000 ms.
2. In Section 2, where the input current is defined, choose `flag = 1` for simulating a constant input current. You can run the script to confirm that this will give you a completely regular firing.
3. At the end of Section 3, add two rows to compute how many action potentials were fired during the course of the simulation, as well as the firing rate (spikes/s). Hint: You can count the number of spike times stored in the vector `spike_train` using the function `sum`. For calculating the firing rate, you need to divide that number by the length of your simulation, which is given by `tstop`. Note: take care of converting `tstop` into seconds, otherwise you will obtain the firing rate in spikes/ms!
   ```
   % Number of action potentials fired
   N_spikes = ...;
   % Firing rate (spikes/s)
   f_secondary = ...;
   ```
4. Instead of a script, let's make this code into a function that returns the spike train and the firing rate. Also, as the setting of `I_const` determines the firing rate and thus, in a way, the stimulus intensity that is coded for, let's make `I_const` an input to this function. Add at the very top of the code
   ```
   function [spike_train,f_secondary] = ...
                              simulateSecondaryNeuron(I_const)
   ```
   Important! i) Make sure to comment or remove the line for cleaning up the workspace, as this would otherwise clear your input, i.e., in this case `I_const`. ii) In Section 2, make sure to comment or remove the line which is setting `I_const`, as it would otherwise overwrite the input.

You can now call this function from the command window or from another script, and thereby simulate the activity of a secondary neuron in the presence of a stimulus whose intensity is coded for by `I_const`. **Document your results for this milestone by showing one example plot each for a weaker and a stronger stimulus and reporting the corresponding values for `I_const` and the firing rates.**

As we want to call this function many times later on, it will be impractical that the plots are produced every single time. In the settings on top, you can insert a flag for plotting which you set to either 0 or 1 (e.g. `plotFlag = 0;`), and you can put the entire plot code on the bottom inside an if-sentence:

```
if plotFlag
   % Plot code
   ...
end
```

Programming milestone 2: Simulate the spike trains in 5 x 5 secondary neurons for a certain applied stimulus to the skin patch

We now want to write a new function, whose output are the spike trains and firing rates for 5 x 5 secondary neurons, given that the corresponding patch of skin is stimulated in a certain way.

1. Open a new file and name it simulateSecondLayer.m.
2. Make the first row in this file the following:
   ```
   function [spike_trains,f_secondary] = ...
                              simulateSecondLayer(Stim)
   ```
   `Stim` will be a 25 x 1 vector, through which you can determine the input current `I_const` to each of the 25 secondary neurons individually. E.g., if you want to simulate a weak pressure to the center of your skin patch only, you would define `Stim` as
   ```
   Stim = zeros(25,1);
   Stim(13) = ...; % Insert your value for a weak stimulus
   ```
3. For simulating the activity in each of your secondary neurons, you will need to program a for-loop:
   ```
   f_secondary = zeros(25,1); % For ensuring a column vector
   for i = 1:25
      [spike_trains(:,i),f_secondary(i)] = ...
                           simulateSecondaryNeuron(Stim(i));
   end
   ```
4. As for our other functions, let us conclude this function with a plotting section, which easily lets us confirm that our function works as intended. For this purpose, we are going to convert our vectors into the 5 x 5 matrixes they actually represent by using the function `reshape`:
   ```
   StimMatrix = reshape(StimMatrix,5,5);
   fMatrix = reshape(f_secondary,5,5);
   ```
5. Add code for opening two new figures, in which you plot `StimMatrix` and `fMatrix`, respectively, by using the function `imagesc`. As a second input to `imagesc`, set the color limits. The lower limit can be 0, and the upper limit should be set with the strong stimulus in mind. This is important for making sure that the same color always represents the same value, such that you can easily compare plots for different stimuli. Use `colorbar` in order to show what the colors represent. As you have a square matrix, it is also useful to have the axis in square size (`axis square`). You can also add a title with the function `title` (it could e.g. be useful to describe what you see and in which unit).
   ```
   figure;
   imagesc(yourMatrix, ...
      [yourLowerColorLimit yourUpperColorLimit]);
   colorbar
   title('Your title');
   axis square
   ```
   Besides the values that the colors are coding for (the input current `I_const` for each secondary neuron and the resulting firing rate of each secondary neuron, respectively), these two figures should basically look the same, as your secondary layer is truthfully coding your stimulus.

You can now call this function from the command window or from another script. Run the function for different setups for the input `Stim` – you can vary which spot(s) on the 5 x 5 matrix on the skin patch you are stimulating, and how strongly you are stimulating. **Document your results for this milestone by describing your stimulus and showing the resulting plots for two examples of your choice.**

Programming milestone 3: Simulate tonic firing activity in a tertiary neuron

Even when there is no input from the periphery, tertiary neurons should fire randomly at a low firing rate, i.e., they should be tonically active. This is very crucial, because otherwise the neurons could only increase their firing (i.e. be excited), but not decrease it (i.e. be inhibited). Without going into further details we are going to assume that this firing is produced by a continuously present random, balanced current across the membrane.

Take the script alpha_neuron.m and rename it to simulateTertiaryNeuron.m. Make the following changes:

1. In Section 3, set the input firing rate to f = 0, such that there is for now no synaptic input.
2. In Section 2, where the input current is defined, copy and paste the settings for a random input current from the script intfire.m from hand-in assignment 2. These are the following lines which you can find in Section 2 of intfire.m for flag = 3:
```
% Peak-to-peak variation of input current [nA]
(default: 50 nA)
A_pp_I = 50;
% Ratio between positive and negative current injection
(default: 1)
Pos_to_neg_ratio = 1;
% Random input current [nA]
I = A_pp_I*( rand(Nt,1) - 1/(Pos_to_neg_ratio+1));
```
Such a random input current will make your neuron fire randomly.
3. At the end of Section 5, add two rows to compute how many action potentials `N_spikes` were fired during the course of the simulation, as well as the firing rate `f_tertiary` (spikes/s). Do this similarly as you did in milestone 1.
4. Instead of a script, let's make this code into a function that takes `A_pp_I` as an input and returns the firing rate of the tertiary neuron. Add at the very top of the code
```
function f_tertiary = simulateTertiaryNeuron(A_pp_I)
```
Important! i) Make sure to comment or remove the line for cleaning up the workspace, as this would otherwise clear your input, i.e., in this case `A_pp_I`. ii) In Section 2, remove again the line defining `A_pp_I`, as it would otherwise overwrite the input. iii) On top of the script, comment the line `rand('state',0)`, such that you get a different random input current everytime you run the script.

You can now call this function from the command window or from another script, and thereby simulate the tonic firing activity of a tertiary neuron. Run the function a number of times and check the produced plots. Adjust the peak-to-peak variation of the input current, `A_pp_I` if you find it necessary (but note that there will be a later milestone entirely dedicated to parameter tuning). **Document your results for this milestone by showing an example plot and reporting your setting for `A_pp_I` as well as the resulting firing rate.**

Programming milestone 4: Simulate the activity of a tertiary neuron in the presence of input from the periphery

We now want to augment our function simulateTertiaryNeuron.m such that it can receive input from the secondary neurons via synapses. For this purpose, we are going to add two inputs to the function: the first is the matrix `spike_trains` which we obtain from the function simulateSecondLayer.m. The second is going to be `E_syn`, which is a 25 x 1 vector defining the synaptic connections from the secondary neurons to the tertiary neuron that we want to simulate. Note that this vector essentially defines the receptive field of the tertiary neuron! For non-existing synaptic connections, we will put the entry in `E_syn` to "not a number", or `NaN`. In our case, for example a tertiary neuron receiving excitatory input from secondary neuron 13 only (i.e. the smallest possible receptive field with no lateral inhibition) would have the setup

```
E_syn = nan(25,1);
E_syn(13) = 0;
```

In order to also add lateral inhibition you should add the following line:

```
E_syn([7 8 9 12 14 17 18 19]) = -70;
```

1. Change the first line in your function to
   ```
   function f_tertiary = ...
         simulateTertiaryNeuron(A_pp_I,spike_trains,E_syn)
   ```
   In Section 1b, comment or remove the line defining `E_syn`, so that you do not overwrite your input. Furthermore, comment or remove the entire Section 3 that was used to generate the input spike train, as we are going to use our input `spike_trains` now instead.
2. Change Section 4 to a for-loop instead, such that you compute the time course of the synaptic conductance for each of the 25 potential synapses separately:
   ```
   for i = 1:25
       g_syn(:,i) = conv(alpha_func,spike_trains(:,i));
   end
   g_syn = g_syn(1:Nt,:);
   ```
3. In Section 5, we finally need to calculate the total synaptic current by adding up all individual synaptic currents at time point `i`. This we will do with a for-loop over all 25 potential synapses. For a non-existing synapse, for which the equilibrium potential will be `NaN`, we will skip that iteration of the for-loop:
   ```
   for j = 1:25
      if isnan(E_syn(j))
          continue
      end
      I_syn(i) = I_syn(i) + g_syn(i,j)*(V(i)-E_syn(j));
   end
   ```
4. As a last step, comment the entire plot section, so that the calling the function does not result in any errors now that you've made large changes to its functionality.

Programming milestone 5: Parameter tuning

At this point you are going to have to do some parameter tuning, which is inevitable when building models. The question is whether your weak stimulus is actually too weak in order to change the tonic firing of the tertiary neuron at all (both for an excitatory and an inhibitory synapse), and whether the difference between your weak and strong stimulus is big enough to make the tertiary neuron fire differently much. The parameters which need to be tuned are the peak-to-peak variation of the input current that makes the tertiary neuron tonically active, `A_pp_I`, the values for `I_const` representing a weaker and a stronger stimulus, and if needed also settings concerning the inhibitory synapses. Write a script `parameterTuning.m` as follows:

1. Define the receptive field for e.g. tertiary neuron 13 as a small receptive field with lateral inhibition:
```
E_syn = nan(25,1);
E_syn(13) = 0;
E_syn([7 8 9 12 14 17 18 19]) = -70;
```
2. Define the values you want to use for a weaker and a stronger stimulus, and the peak-to-peak variation of the current producing the tonic firing:
```
I_const_weak = …;
I_const_strong = …;
A_pp_I = …;
```
3. In the first scenario, simulate the firing in your tertiary neuron a number of times (e.g. 20) upon weakly touching outside of its receptive field. This will only result in the tonic firing. Save each of your results in the first column of what will be matrix, `f`.
```
% Define the stimulation
Stim = zeros(25,1);
Stim(1) = I_const_weak;
% Simulate the secondary layer
spike_trains = simulateSecondLayer(Stim);
% Simulate the firing in your tertiary neuron a number
of times
for i = 1:20
    f(i,1) = ...
     simulateTertiaryNeuron(A_pp_I,spike_trains,E_syn);
end
```
4. Similarly, run the following scenarios and save them in columns 2 to 5 of `f`:
   - Weak touch in the center
   - Strong touch in the center
   - Weak touch in the surround
   - Strong touch in the surround
5. Plot your results with the `boxplot` command.
```
figure; boxplot(f);
```

Are you satisfied with the outcome? If not, tune your parameters (best is one at a time) and rerun the script until you are happy. **Document your results by reporting the final values for `A_pp_I`, `I_const` for the weak and the strong stimulus and the equilibrium potential for the inhibitory synapses, as well by including your final figure.** You could also easily illustrate the entire receptive field as described further down.

Further advice

Having solved the tasks of the hand-in assignment, you should have all the building blocks needed to implement to look at receptive fields and e.g. two-point discrimination, and investigate some specific questions of your own. Below are some further hints to keep you going.

Simulate the receptive field of a tertiary neuron (corresponding to Fig. 1)
Let's take for example the very simple setup of simulating the receptive field for tertiary neuron #13, for the setting of the smallest possible receptive field with no lateral inhibition:

```
E_syn = nan(25,1);
E_syn(13) = 0;
```

Now, run a for-loop for doing your 25 stimulations and recording the firing of your tertiary neuron in each case:

```
f_tertiary = zeros(25,1); % For ensuring a column vector
for i = 1:25
   % Make the setting to stimulate at spot i
   Stim = zeros(25,1);
   Stim(i) = ...;%Insert your value for I_const
   % Obtain the spike trains for your secondary neurons
   spike_trains = simulateSecondLayer(Stim);
   % Obtain the firing rate of your tertiary neuron
   f_tertiary(i) = simulateTertiaryNeuron(spike_trains,E_syn);
end
% Convert vector into the 5 x 5 matrix
f_tertiary = reshape(f_tertiary,5,5);
% Plot with imagesc (see Step 5 in Milestone 2)
...
```

Simulate the spike trains in 5 x 5 tertiary neurons for a certain applied stimulus to the skin (corresponding to Fig. 2 and similar to Milestone 2)
This code you would use for investigating two-point discrimination. Let's take for example the very simple setup of that all tertiary neurons have the smallest possible receptive field with no lateral inhibition.

```
% Decide on which spot(s) you want to touch
Stim = zeros(25,1);
Stim(...) = ...;%Insert your value for I_const
Stim(...) = ...;%Insert your value for I_const
% Simulate the activity in your secondary layer
spike_trains = simulateSecondLayer(Stim);
```

Now, run a for-loop simulating the activity in each of your 25 tertiary neurons:

```
f_tertiaryLayer = zeros(25,1); % For ensuring a column vector
for i = 1:25
   % Make the setting e.g. for the smallest possible
     receptive field with no lateral inhibition
   E_syn = nan(25,1);
   E_syn(i) = 0
   % Obtain the firing rate of your tertiary neuron
   f_tertiary(i) = simulateTertiaryNeuron(spike_trains,E_syn);
end
% Convert vector into the 5 x 5 matrix
f_tertiary = reshape(f_tertiary,5,5);
% Plot with imagesc (see Step 5 in Milestone 2)
...
```