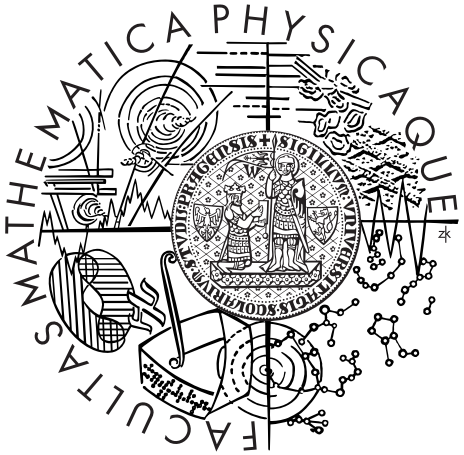
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Markéta Popelová

Software tool for modelling coding and processing of information in auditory cortex of mice

Department of Software and Computer Science Education

Supervisor of the master thesis: Mgr. Cyril Brom, Ph.D.
Study programme: Computer Science
Specialization: Theoretical Computer Science

Prague 2013

I would like to thank my supervisor Cyril Brom for advises and comments. Huge thanks also belong to other three people. First, to Ondřej Novák for numerous and important consultations, encouragement and help in becoming at least a little more familiarized with the auditory neuroscience and neuroscientific modelling. Second, to Ondřej Zelenka for biological explanations and recommendations for literature. And finally to Jakub Tomek for his enthusiasm, support, and possibility of numerous and inspiring discussions.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date ..... signature of the author

Název práce: Software tool for modelling coding and processing of information in auditory cortex of mice

Autor: Markéta Popelová

Katedra: Kabinet software a výuky informatiky

Vedoucí diplomové práce: Mgr. Cyril Brom, Ph.D., Kabinet software a výuky informatiky

Abstrakt: Porozumění zpracovávání a kódování informací ve sluchové kůře (AC) je stále nedostatečné. Z několika různých důvodů by bylo užitečné mít výpočetní model AC, například z důvodu vysvětlení, či ujasnění procesu kódování informací v AC. Prvním cílem této práce bylo vytvořit softwarový nástroj (simulátor SUSNOIMAC), zaměřený na modelování AC. Druhým cílem bylo navrhnout výpočetní model AC s následujícími vlastnostmi: Izhikevichův model neuronu, dlouhodobá plasticita ve formě Spike-timing-dependent plasticity (STDP), šestivrstvá architektura, parametrizované typy neuronů, hustota neuronů a pravděpodobnost vzniku synapsí. Navržený model byl testován v desítkách experimentů, s různými sadami parametrů a v různých velikostech (až 100 000 neuronů s takřka 21 milióny synapsí). Experimenty byly analyzovány a jejich výsledky srovnány s pozorováním skutečné AC. V práci popisujeme a analyzujeme několik zajímavých pozorování o aktivitě modelované sítě a vzniku tonotopického uspořádání AC.

Klíčová slova: sluchová kůra, Izhikevichův model neuronu, spiking neurony, STDP

Title: Software tool for modelling coding and processing of information in auditory cortex of mice

Author: Markéta Popelová

Department: Department of Software and Computer Science Education

Supervisor: Mgr. Cyril Brom, Ph.D., Department of Software and Computer Science Education

Abstract: The coding and processing of information in the auditory cortex (AC) is still poorly understood. Having a computational model of the AC could be beneficial for many reasons and could help to explain or clarify some questions about the coding processing of information in the AC. The first goal of this thesis was to develop a software tool (simulator SUSNOIMAC), specialized to modelling the AC. The second goal of the thesis was to design a computational model of the AC, with the following features: Izhikevich model of neuron, long-term plasticity in the form of the Spike-timing-dependent plasticity (STDP), six layers, and parametrized neuron types, density of neurons and probability of synapses. The designed model was tested in dozens of experiments, with different parametrization and network sizes (up to 100 thousand of neurons and almost 21 million of synapses). The experiments were analyzed and compared to the observations from the real AC. Several interesting observations of network activity and emergence of tonotopy were made, described and analyzed.

Keywords: auditory cortex, Izhikevich model of neuron, spiking neurons, STDP

Table of contents

General Introduction

The ultimate target of ascending auditory pathways is the *auditory cortex* (AC), a region of the brain cortex (see Figure 1). The research in auditory neuroscience has made great progress in recent years. A lot is known about the electrophysiological and morphological characteristics of single neurons in the auditory cortex (Metherate and Aramakis, 1999; Watson, 2012; Oswald and Reyes, 2008; Wu et al., 2011). We also have an idea of certain high-level functions of the individual auditory areas. For example Broca's area is traditionally considered a language area and Heschl's gyrus is associated with music perception (Meddis, 2010, chap. 5). It is even known that the perception of language and music are closely related and that they share the same neural resources (Koelsch and Siebel, 2005; Ross et al., 2007; Meddis, 2010, chap. 5). However the exact mechanisms of music, speech, and other complex sounds processing, is unclear.

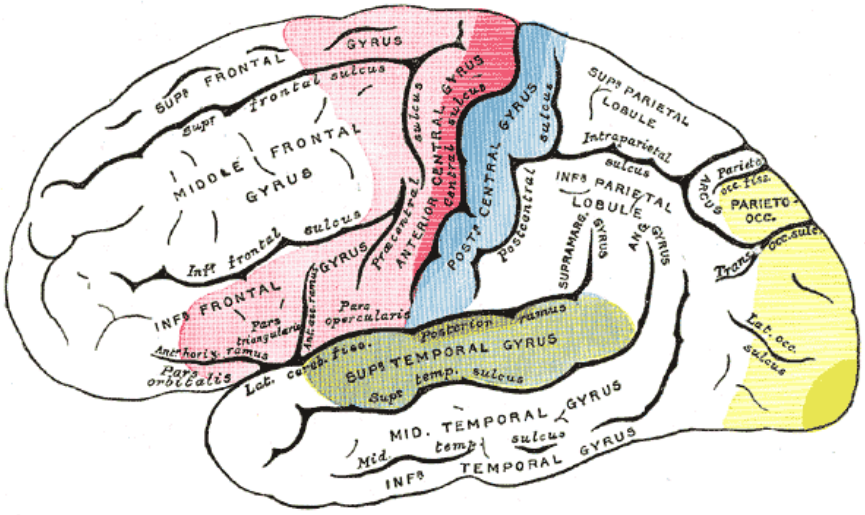


Figure 1: *The human brain and areas of localization on lateral surface of hemisphere:* ***auditory area****,* ***motor area****,* ***visual area****, and area of* ***general sensation****. From (wikipedia.org, 2007a).*

Research of processing and representation of complex sounds in the AC is full of contradictory results. Whereas, according to some studies, it appears that AC performs relatively simple operations (e.g. representations of physical aspects of the stimuli, such as sound frequency or amplitude (Brodal, 2010)), other studies show large degree of complexity in the neuronal responses (Näätänen et al., 2001), for review see (Nelken, 2004). Sound categorization (Nelken et al., 2003; Ohl et al., 2001; Russ et al., 2007; Bathellier et al., 2012) and sound representation in terms of auditory objects (Nelken, 2004) are generally considered to be some of the main functions of the AC. However, mechanisms behind these functions are still a subject of debate and research.

Many experimental studies have researched the neuronal activity of single neurons. Single-units recordings have revealed a lot about the properties and behaviour of single neurons, such as receptive fields (Brugge and Reale, 1985; Sally and Kelly, 1988; Phillips and Kelly, 1989; Bathellier et al., 2012). However, single neurons carry only pieces of information and their strength is in putting the infor-

mation together. Therefore, the focus has gradually moved on researching activity of neuronal populations (Gaese, 2001; Harris et al., 2011; Rothschild et al., 2010; Bathellier et al., 2012). This is possible thanks to development of new technology, such as silicon microelectrode arrays, tetrodes, spike-sorting techniques, or two-photon *in vivo* microscopy.

In general, one of the current open problems is processing and coding information in large-scale neuronal networks. The question is to what extent and exactly how the short-term and long-term synaptic plasticity, precise spike timing, or synaptic delays contribute to it. It appears that all these factors are important in the processing of stimuli and sound representation.

Computational models may be a useful tool for verification of possible mechanisms of information coding and processing. Having a model that to a certain extent corresponds to the observed reality allows us to test which mechanisms work, what features are responsible for the observed behaviour and how would the system behave without them. Such *in silico* experiments can have various applications.

First, they can help to decompose and understand the researched system. At the same time, these experiments can provide additional ideas to explain particular phenomena. They can also propose potentially promising experiments *in vivo* for testing hypotheses confirmed by the computational model. This is advantageous especially in those cases where *in vivo* experiments are difficult, costly, or hard to repeat (e.g., due to time constraints, or impossibility of many replications), but ideas what to test are plentiful and only some of them will lead to useful results. Experiments *in silico*, running of which is typically fully automated, can identify promising candidates. Finally, the traditional and probably the most useful application of the models is modelling neural diseases or brain disorders. It allows researching which mechanisms are responsible for which functions and which interventions lead to the observed pathophysiological state. This knowledge can be very useful in the design of methods for treating these disorders (e.g., tinnitus, some types of hearing loss, epilepsy, etc.).

Computational models of the auditory system are by no means new. On the contrary, many models of lower stages of the auditory system (e.g., auditory periphery, or cochlear nucleus) have been published and a part of them has led to practical results. Majority of them help us in understanding the system, some of them help with speech intelligibility (Assmann and Summerfield, 2004; Zilany and Bruce, 2007), other help in development of hearing-aid signal processing algorithms (Edwards, 2004, 2007; Levitt, 2004), or prostheses (Biondi and Schmid, 1972; Biondi, 1978). Auditory models are used at least to some extent in all current designs of cochlear implant systems (Meddis, 2010, chap. 9). A detailed description of the design of implantable auditory prostheses is presented in (Wilson, 2004). Other computational models were developed to research tinnitus and its relation to sensorineural hearing loss (Schaette and Kempter, 2006).

Modelling the auditory cortex is much less explored area. Few mathematical and more abstract models have been developed, but large-scale AC models with spiking neurons are almost untouched topic (Meddis, 2010, chap. 5). The only few related works are described in the Chapter 6. Modelling of the AC is rather challenging. It is still difficult to obtain parameters for neurons: their spatial locations, features and connections, input signals from thalamus and other

cortical areas and data for model verification. Nevertheless, imaging techniques are developing rapidly, allowing to record neuronal activity more precisely (in time or spatial aspect), larger areas and areas deeper in the brain. Thanks to that the models can be more accurate and plausible. We suppose that developing such a model of the AC may lead to better comprehension of signal coding and processing in the AC.

The main goals of this thesis were:

1. To develop a software tool (a simulator), in which models of the AC could be created, run and analysed.

   (a) The main requirements were: Izhikevich neuron model (as currently one of the most successful spiking neuron models in terms of plausibility and computational cost (Izhikevich, 2003; Izhikevich et al., 2004a)), synaptic delays (as it turned out to be a very important factor of information encoding) and long-term synaptic plasticity in the form of spike-timing dependent plasticity (STDP) (Song et al., 2000).

   (b) The main expected use of the simulator should be development of models of the AC with the basic properties of the real cortex of mammals: six layers, several types of neurons with different parameters of the Izhikevich neuron model, different quantity and distribution in layers, as well as different distribution of synapses from other neurons.

2. To design and create one simple model of the AC and design and run several experiments on this model (using the developed simulator) in order to test and describe basic features of the model.

The model was consulted with Mgr. Ondřej Novák and MUDr. Ondřej Zelenka, neuroscientists from Institute of Experimental Medicine AS CR, the Department of Auditory Neuroscience. In the rest of the thesis, by personal communication with neuroscientists, we mean them.

The text of the thesis is divided into four main parts:

1. Introductory:

   - Chapter 1 provides a brief introduction to neurobiology and neuroscientific modelling

2. Simulator-related:

   - Chapter 2 reviews other software tools for neuroscientific modelling
   - Chapter 3 describes the simulator developed in this thesis
   - Chapter 4 describes validation tests and performance of the simulator

3. Model-related:

   - Chapter 5 provides an introduction to the auditory system and especially the AC

- Chapter 6 presents a motivation for a model of the AC, reviews existing AC models and their limitations, and states the main features of the model designed in this thesis
- Chapter 7 describes the model of the AC designed in this thesis
- Chapter 8 contains evaluation of the model: performed experiments, their results and discussion of these results
- Chapter 9 summarizes the main features of the model, compares it to other existing models, and suggests possible future works related to the model

4. Concluding:

    - Chapter 10 concludes the whole thesis and its achievements

We should note that although the main use of the simulator should be models of the AC, there was an effort to develop the simulator universal enough to allow creating general models, not only AC models. Therefore the chapters related to the simulator are separated from and even precede the chapters related to the model.

1. Introduction to Neurobiology and Computational Neuroscience

Computational neuroscience has become a very popular field over the past decades. Many books have been written about this topic. A suitable introduction can be found for example in (Dayan et al., 2001; Gerstner and Kistler, 2002; Trappenberg, 2010; Schutter, 2010). It would be futile to try to repeat such an introduction in this text. Instead, this chapter provides a brief summary of the basics of neurobiology and it puts the used computational methods into a broader context of the computational neuroscience.

There is a wide array of approaches to computational modelling. Some of them focus on microscopically precise features, while others use number of simplifications and high degree of abstraction, which allows computing large-scale systems over long period of time. It is impossible to have a model which is both perfectly plausible[1] and computationally cheap. If we want to model a large system, we have to decide which parts of the system will be simplified. Interestingly, although the number of neurons in mammal's brain reaches values in the order of billions (10^9) (Herculano-Houzel, 2009), even models with thousands of neurons are considered large-scale (e.g., a barrel cortical model by (Phoka et al., 2012). In recent years, ambitious projects such as the Blue Brain Project (Markram, 2006), the Spaun project (Eliasmith et al., 2012), the Cognitive Computation Project (Ananthanarayanan et al., 2009), and others (Izhikevich and Edelman, 2008) have appeared with the goal of building biologically accurate model of the entire brain. Although their results are respectable, we still cannot say that modelling human brain is a solved problem[2]. Moreover, a more detailed model is not necessarily superior, because a simpler model can be easier to analyse (Schutter, 2010; Dayan et al., 2001).

The rest of the chapter briefly describes fundamental overview of neurobiology (based on (Brodal, 2010; Bear et al., 2007)), basic approaches to neuroscientific modelling and somewhat more detailed description of the spiking neuron models, the class of models, which we use in this thesis.

1.1 Brief overview of neurobiology

Before we start with the overview, we would like to mention that goal of this section is to provide just a basic grasp of the principle and established knowledge from neurobiology. Naturally, none of these facts are so simple in reality and moreover a great part of reality remains unclear. However, the aim of this section is not to describe the most recent discoveries, but only basic knowledge, even though exceptions may exist, in some cases.

The core of the information transmission in the brain is mediated by neurons, excitable cells which are able to generate electrical signals in response to chemical and electrical inputs, and transmit the signals to other cells (see Figure 1.1). These

[1] Already the idea of having a perfectly plausible model is, in most cases, impossible and we must be aware of the fact, that a model is always just an abstraction of the reality.

[2] It is a rather philosophical question if this will be ever possible.

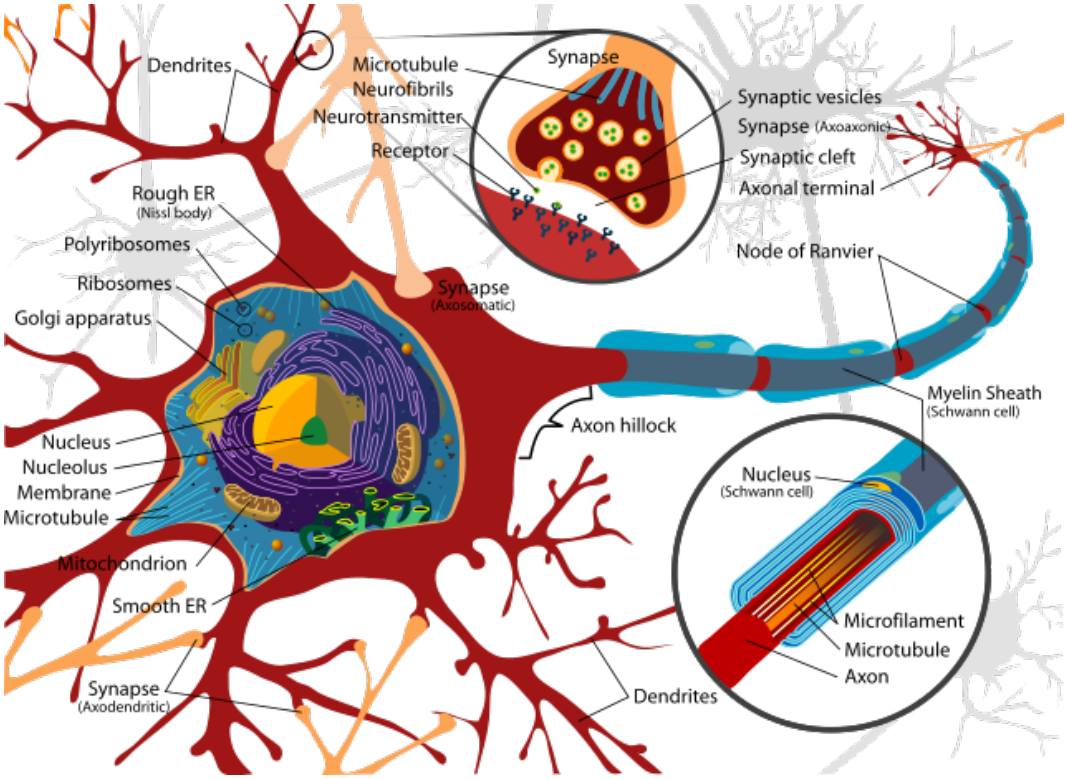


Figure 1.1: *A diagram of a neuron cell. From (wikipedia.org, 2007b).*

signals are called action potentials (or nerve impulses or, more simply, spikes). The action potential is a brief change of the membrane potential, caused by opening of channels that allow Na^+ ions to enter the neuron, followed by an outward flow of K^+ ions. A spike arises in the axon hillock, the initial segment of the axon (a long protrusion of the neuronal cell), where the density of voltage-gated Na^+ channels is higher than in the rest of the neuron membrane. Subsequently, the action potential is propagated along the axon, to the terminal branches of the axon, called boutons. The bouton lies close to the surface membrane of another neuron cell (its dendrite, soma, or even axon) and the site of this contact is called a synapse (see Figure 1.2).

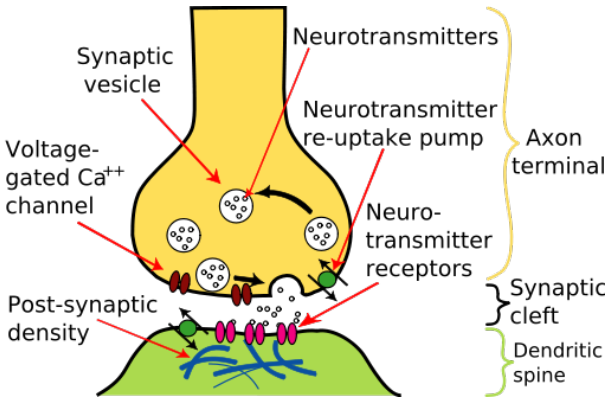


Figure 1.2: *A diagram of a synapse. From (wikipedia.org, 2006).*

The majority of synapses between neurons are chemical synapses. The other type, electrical synapses, called gap junctions occur infrequently among neurons

(Brodal, 2010) and therefore we do not discuss them in this text. The signal transfer at a chemical synapse starts when an action potential reaches the bouton of the presynaptic neuron. It depolarizes the bouton, which leads to opening Ca^{2+} channels, enabling Ca^{2+} ions to enter the bouton. The increased concentration of intracellular Ca^{2+} causes a release of neurotransmitter from synaptic vesicles to the synaptic cleft, a narrow space between presynaptic and postsynaptic neuron. The released neurotransmitter briefly binds to receptors of the postsynaptic membrane, which causes a change of the postsynaptic membrane potential, called postsynaptic potential (PSP). If the synaptic potential depolarizes the postsynaptic cell (usually by opening cation channels allowing Na^+ to enter and K^+ to leave the cell), it is called an excitatory postsynaptic potential (EPSP) and it increases the probability that the postsynaptic neuron will fire an action potential. If the synaptic potential hyperpolarizes the postsynaptic cell (usually by opening channels allowing Cl^- to enter or K^+ to leave the cell), it is called an inhibitory postsynaptic potential (IPSP) and it decreases the probability that the postsynaptic neuron will fire an action potential. The changes in potential are summated from many synapses and if the membrane is depolarized to a threshold value, the action potential in the postsynaptic cell is evoked. When depolarization reaches its maximum value (usually around +30 mV), the repolarisation follows (and K^+ ions are driven outside the cell). The whole sequence of depolarization and repolarization lasts 1 to 2 ms (Brodal, 2010). Usually, the threshold is approximately 10 mV more positive than the resting potential (typically around -60 mV). The size of one EPSP is usually less than 1 mV (Brodal, 2010). Therefore a summation of several EPSP is needed to reach a threshold for eliciting an action potential. Otherwise, the membrane potential returns to the resting value.

If the released neurotransmitter produces an EPSC, we use the terms excitatory synapse and excitatory neurotransmitter (e.g., glutamate, acetylcholin, aspartate). Analogically, IPSC is caused by an inhibitory neurotransmitter (e.g., GABA or glycine) at an inhibitory synapse. Since the majority of neurons release either excitatory or inhibitory neurotransmitters, it is common (especially between computational neuroscientists) to simplify the terminology and refer to the (presynaptic) neurons with excitatory effect as excitatory neurons and to the cells with inhibitory effect as inhibitory neurons. However, the distinction between excitatory and inhibitory neurotransmitters is not perfect. For example, glutamate can produce an EPSC when bound to group I mGluRs (metabotropic glutamate receptors), whereas binding to group II mGluRs produces an IPSC (Brodal, 2010). This dual effect of glutamate has an essential impact on retinal signal processing, where ON bipolar cells are hyperpolarized by glutamate, whereas OFF bipolar cells are depolarized (Gerber, 2003).

The size of an EPSC and an IPSC is influenced by the synaptic efficacy. The synaptic efficacy may change over time, making the synapses plastic. Synaptic plasticity is regarded as a crucial factor in learning and memory (Brodal, 2010; Martin et al., 2000). We distinguish between short-term plasticity, which lasts from less than a second to several minutes, and long-term plasticity, which can last from hours to weeks and even longer.

Short-term plasticity refers to short-term changes of synaptic efficacy in reaction to a stimulus repeated in brief intervals. While some synapses are enhanced

by repeated stimuli, others are depressed. Forms of synaptic enhancement, such as facilitation, augmentation, and post-tetanic potentiation, are usually attributed to effects of residual elevation of Ca^{2+} ions in presynaptic bouton and subsequentl increased transmitter release. Short-term synaptic depression is usually attributed to insufficient renewal of the releasable synaptic pool, or feedback activation of presynaptic receptors, or postsynaptic processes such as receptor desensitization. More detailed principles and mechanisms of short-term plasticity are discussed in (Zucker and Regehr, 2002).

Long-term plasticity refers to changes in synaptic efficacy lasting for hours and more. It is generally supposed that strong activity and synchronization of specific inputs strengthens the synaptic connections, whereas low activity or desynchronized inputs weakens the synaptic connections. The latter may be interpreted as noise or unimportant information. It is assumed (Brodal, 2010) that the long-term potentiation (LTP) is induced by a presynaptic action potential repeatedly preceding a postsynaptic action potential. In this situation, the presynaptic spike contributes to the postsynaptic spike; hence the synapse should be enhanced. On the contrary, when the presynaptic spike arrives after postsynaptic spike, the information arrives late and the synapse is depressed by the long-term depression (LTD). Even though these phenomena are opposite, they are both induced by an increase in intracellular Ca^{2+}. Important role in long-term plasticity is performed by NMDA (N-methyl-D-aspartate) receptors thanks to higher permeability to Ca^{2+} of NMDA-gated ion channels. More about LTP, STP and memory in general can be found in (Sweatt, 2010).

To sum up this section so far, the information in brain is transmitted among neurons via action potentials, conducted from a presynaptic neuron along its axon, through a synapse to dendrite, axon or soma of the postsynaptic neuron, which may lead to an action potential being fired by the postsynaptic neuron. The value of PSP evoked in postsynaptic cell is influenced by the synapse type and state (such as synapse efficacy, or state of neurotransmitter vesicles). The action potential is shaped by the neuron type and state (e.g., the preceding membrane potential value). Although the duration, amplitude and shape of the action potential may vary, these features generally do not influence the information transmitted to other neurons (Brodal, 2010; Bear et al., 2007; Dayan et al., 2001). Therefore, it is called as all-or-none phenomenon. The information is encoded in frequency and pattern of consecutive action potentials. Therefore, from the computer science point of view, the information transmitted by one neuron can be characterized as a spike train, a simple list of the times, when spikes occurred.

Further detailed information from the field of neurobiology can be found for example in (Brodal, 2010; Bear et al., 2007).

1.2 Brief overview of neuroscientific modelling

In the field of computational neuroscience various types and kinds of models can be found. To facilitate the orientation in the wide array of models, we divide them into three groups, according the degree of abstraction and precision.

1.2.1 Level of Detail in models

1.2.1.1 Low-level models

The first group of models consists of low-level models which model individual molecules, their movement and reactions. Mathematical background of these models is based on the chemical equations, into which the modelled molecules enter. Extensive knowledge of neurotransmitters, receptors, ions, channels, enzymes, and other chemical structures, which play a role in the nervous system, is utilized here. Authors of the models consecutively try to compose the chemical computations that occur at synapses, ion channels, axons, dendrites and neuron somas. Further reading can be found in (Bower and Bolouri, 2004; Fall, 2005; Schutter, 2010, chap. 3–4).

1.2.1.2 Middle-level models

In the group of middle-level models, the basic units are neurons and synapses and the transmission of action potential is modelled. Each neuron reacts to the PSPs from presynaptic neurons in a specific manner. This manner typically means changing the value of the neuron's membrane potential in the reaction to input potential (or input current). The input potential typically consists of the sum of PSPs (or Post Synaptic Currents (PSCs)) from presynaptic neurons. This behaviour is called the neuron model and it defines not only the progression of the membrane potential, but it also defines when the neuron fires a spike.[3] The neuron model can be really simple, such as a test, whether input potential exceeds a fixed threshold, or much more complicated based on computation of one or more differential equations (as it is in the well-known Hodgkin-Huxley model (Hodgkin and Huxley, 1952), which will be described later). One of these differential equations typically captures the membrane potential course and the others may capture for instance the fast sodium current I_{Na} and delayed potassium rectifier I_K mediated by Na^+ and K^+ ions, respectively. There are many other features in which these models vary, such as single-compartment models versus multiple-compartment model. The most important of these features are described later.

The second elementary units in these intermediate models are synapses. Their behaviour (model) may be again very simple, such as a fixed weight (which corresponds to a neurobiological term of synaptic efficacy). Or it can be intermediately complicated, such as a weight influenced by the short-term and/or long-term plasticity. Finally, it can be more complicated, such as modelling synaptic conductance, its changes in time in reaction to synapse type, passing current and other factors (e.g., plasticity).

Since these middle-level models are widely used in the neuroscientific community, their description can be found in almost every book dealing with computational neuroscience, such as (Dayan et al., 2001; Gerstner and Kistler, 2002; Trappenberg, 2010; Schutter, 2010; Izhikevich, 2007). Except really simple models, these models are often called "spiking neuron models". The exception includes simple and rather artificial models, such as perceptron or back-propagation

[3]This may be sometimes confusing that the term model sometimes mean behaviour of the whole network, other times behaviour of a single neuron, or even synapse.

neural network (more about them, e.g., in: (Reed and Marks, 1999)). Their basic units are also neurons and connections between neurons. However these units resemble the real neurons only remotely, there is not even the membrane potential modelled and instead of coding information in the frequency and pattern of spikes, it is usually coded in the magnitude of the transmitted signal. On the other hand, the purpose of the artificial neural networks is not plausibility, but the success in machine learning tasks (for more about machine learning, see e.g., Alpaydin, 2004).

1.2.1.3 High-level models

The group of high-level models contains models with some kind of high-level abstraction. They often work with statistical or mean features and basic units are typically neuron populations. This group includes firing rate models and population models, where instead of single neurons, the neuron populations are modelled by the mean activity. "*It is clear that rate models cannot incorporate all aspects of networks of spiking neurons. However, many of the principles behind information processing in the brain can be illuminated on the level of population models, and many of the features of population models have been confirmed with spiking neurons.*" (Trappenberg, 2010, pp. 74). In addition to that, for some features, a more abstract model is more suitable. More information about these high-level models can be found for instance in (Wilson and Cowan, 1972; Brunel and Wang, 2001; Trappenberg, 2010, chap. 3).

1.2.1.4 Model group decision

In this thesis, the middle-level approach was chosen, as the golden mean. Compared to the high-level approach, it helps to capture more precisely features of the network and it allows to research transmission of the information at the level of single neurons and synapses. In the case of the model of the auditory cortex, it may be useful e.g., for observing tonotopic arrangement (see the Chapter 5), where receptive fields of single neurons are important. On the other hand, the low-level approach is too detailed and does not allow computing large scale networks (in the order of at least 10^4 of neurons).

The next sections contain a description of basic techniques and models of the spiking neuron networks. That means models of neurons, models of synapses, simulation techniques and other simulation aspects, such as inputs, noise, or spontaneous activity.

1.2.2 Neuron models

1.2.2.1 The Hodgkin-Huxley model

The model of Alan Hodgkin and Andrew Huxley (Hodgkin and Huxley, 1952) was based on results of voltage-clamp experiments of ion channels of the squid giant axon. They characterized the kinetics of the fast sodium current and the delayed potassium rectifier. The resulting mathematical model was used also as a verification that the proposed kinetics leads to the plausible action potential generation.

The basic form of the model consists of four non-linear ordinary differential equations:

$$\begin{aligned} C\frac{dV}{dt} &= -\overline{g}_L(V - E_L) - \overline{g}_{Na}m^3h(V - E_{Na}) - \overline{g}_K n^4(V - E_K) \\ \frac{dn}{dt} &= \alpha_n(V)(1 - n) - \beta_n(V)n \\ \frac{dm}{dt} &= \alpha_m(V)(1 - m) - \beta_m(V)m \\ \frac{dh}{dt} &= \alpha_h(V)(1 - h) - \beta_h(V)h \end{aligned} \tag{1.1}$$

where V is the membrane potential, C is the membrane capacitance, $\overline{g}_L$ is the maximal value for membrane conductance for the leak current, $\overline{g}_{Na}$ for the Na^+ current and $\overline{g}_K$ for the K^+ current, E_L, E_{Na}, E_K are their respective reversal potentials given by the Nernst equation. Variable n describes activation of the potassium channels, m describes activation of the sodium channels, and h describes deactivation of the sodium channels. Functions $\alpha_n(V)$, $\alpha_m(V)$, $\alpha_h(V)$, $\beta_n(V)$, $\beta_m(V)$, and $\beta_h(V)$ describe the transition rates between open and closed states of the channels. They were fitted experimentally and their description can be found for instance in (Hodgkin and Huxley, 1952; Trappenberg, 2010; Izhikevich, 2007; Schutter, 2010, chap. 5).

The model reproduces the behaviour of the recorded currents well and its parameters can be relatively easily obtained from the experimental data. Therefore, this model is still one of the most important models in computational neuroscience. On the other hand, due to the computational cost is its use in large-scale models very limited. In the rest of the text, the model will be referenced as HH model.

1.2.2.2 The Leaky Integrate and Fire model

As opposed to the HH model, which aims for maximum plausibility, but leads to high computational demands, the Leaky Integrate-and-Fire neuron model (Lapicque, 1907; Knight, 1972; Tuckwell, 1988) is computationally cheap, although much simpler. The model is sometimes referred as I&F, or LIF model (we will use the latter abbreviation). The origins of the LIF model may be traced to a model with a simple capacitor circuit, introduced by Louis Lapique already in 1907 as the result of observations of frog nerve stimulation (Brunel and van Rossum, 2007). Presently used form of the model can be described by one linear differential equation:

$$C\frac{dV}{dt} = -g_L(V - E_L) + I_{syn}(t) \tag{1.2}$$

$$\text{if} \quad (V \geq V_t) \quad \text{then} \quad V \leftarrow V_r \tag{1.3}$$

where V is the membrane potential, C is neuron capacitance, g_L is the leak conductance, E_L is the leak (or resting) potential, and $I_{syn}(T)$ are the synaptic inputs. When the voltage reaches a threshold V_t, the neuron is said to emit a spike and the voltage is reset to a reset potential value V_r, after an absolute refractory period τ_{rp}.

The model has the following features:

- all-or-none spike: all spikes have an identical size and duration (because the shape is not modelled),
- well-defined threshold: with the value of V_t,
- reset value of membrane potential after a spike is emitted: with the value of V_r,
- refractory period: with the value of τ_{rp},
- distinction between excitation and inhibition: excitatory inputs ($I_{syn}(t) > 0$) facilitate the firing, while inhibitory inputs ($I_{syn}(t) < 0$) do the opposite,
- class 1 excitability (Hodgkin, 1948; Izhikevich, 2007, pp. 218): the neuron can continuously encode the strength of an input into the frequency of spiking.

It is probably the simplest model described by a membrane potential variable with these features. Actually, it should not even be called a spiking model, since it lacks any spike generation mechanism, where spike means a brief regenerative depolarization of membrane potential (Izhikevich, 2007). Despite its simplicity and limited plausibility, it is still widely used in large-scale simulations.

1.2.2.3 The Izhikevich neuron model

Since HH-type models are too computationally demanding and the LIF model is too simple, many scientists tried to combine the qualities of both types: keep some nontrivial features of the dynamics of the HH model, as well as keep the model computationally efficient. This effort led to birth of many models better or worse combining the opposite demands. One of the most successful results is the model of spiking neuron by Izhikevich. The model was published first in 2003 (Izhikevich, 2003) and used e.g., in (Izhikevich, 2004; Izhikevich et al., 2004b; Izhikevich, 2006). In this text, we call this version the "Original form". The second version is from 2007 (Izhikevich, 2007) and in this text, we call the version as the "Generalized form". This version was used e.g., in (Izhikevich and Edelman, 2008).

1.2.2.3.1 Original Form

In 2003, Izhikevich introduced a new model of spiking neuron in the following form:

$$\begin{aligned} \frac{dv}{dt} &= 0.04v^2 + 5v + 140 - u + I \\ \frac{du}{dt} &= a(bv - u) \end{aligned} \tag{1.4}$$

with the auxiliary after-spike resetting:

$$\text{if} \quad (v \geq 30mV) \quad \text{then} \quad v \leftarrow c, \quad u \leftarrow u + d \tag{1.5}$$

where v is the membrane potential, u represents a membrane recovery current, which accounts for the activation of K^+ ionic currents and inactivation of Na^+ ionic currents and it provides negative feedback to v, I is the sum of synaptic inputs, and a, b, c, and d are parameters of the model. When v reaches its apex (+30 mV), the membrane voltage and the recovery variable are reset according to the 1.5. The value +30 mV is not a threshold, but the peak of the spike. The threshold value of the model neuron is between −70 mV and −50 mV, and it is dynamic. The equation 1.4 was fitted so that values of v have mV scale and values of the time has ms scale.

The Izhikevich model can reproduce various types of neuronal dynamics, such as *regular spiking* (RS), *intrinsically bursting* (IB), and *chattering* (CH), the main excitatory cortical neuron classes (Connors and Gutnick, 1990; Gray and McCormick, 1996), or *fast spiking* (FS), and *low-threshold spiking* (LTS)), the main inhibitory cortical neuron classes (Gibson et al., 1999). Possible parameters for these types are listed in the Table 1.1.

	RS	FS	LTS
v_p	30	30	30
a	0.02	0.1	0.02
b	0.2	0.2	0.25
c	-65	-65	-65
d	8	2	2

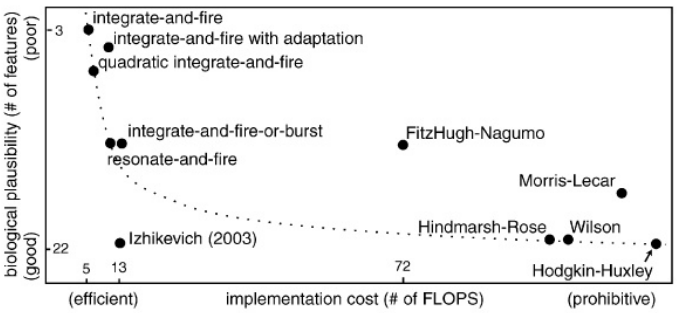
Table 1.1: *Possible values of parameters of the (Izhikevich, 2003) neuron model for the main cortical neuron classes, based on (Izhikevich, 2003).*

This model successfully combines the opposite demands: the biological plausibility and computational efficiency. To compare the main neuron models according to these demands, Izhikevich (Izhikevich et al., 2004a) defined two metrics for the models comparison. The first metric was defined by three factors:

- twenty of the most prominent features of biological spiking neurons,
- ability to exhibit autonomous chaotic activity,
- decision, whether the model has biophysically meaningful and measureable parameters.

The second metric measured approximate number of floating point operations (FLOPS), such as addition, or multiplication, needed to simulate the model during a 1 ms time span. The results (see Figure 1.3 and for more details see (Izhikevich et al., 2004a) showed that at least 15 features of the first metric have only the following models: HH (Hodgkin and Huxley, 1952), Wilson (Wilson, 1999), Hindmarsh-Rose (Rose and Hindmarsh, 1989) and Izhikevich (2003). However, the first three models needed 120 FLOPS (HH needed even 1200 FLOPS), while Izhikevich (2003) needed only 13 FLOPS. On the contrary, the most efficient model was the LIF model; however it fulfilled only 3 features of the first metric and it was not able to exhibit many other fundamental properties. According to this study, if maximal biological plausibility (and dependence on measurable physiological parameters such as maximal conductances) is needed, the HH model is

the best. On the other hand if large-scale network and computational efficiency and also reasonable plausibility are needed, then the Izhikevich (2003) model is the best.[4]



Models	biophysically meaningful	tonic spiking	phasic spiking	tonic bursting	phasic bursting	mixed mode	spike frequency adaptation	class 1 excitable	class 2 excitable	spike latency	subthreshold oscillations	resonator	integrator	rebound spike	rebound burst	threshold variability	bistability	DAP	accommodation	inhibition-induced spiking	inhibition-induced bursting	chaos	# of FLOPS
integrate-and-fire	-	+	-	-	-	-	-	+	-	-	-	-	+	-	-	-	-	-	-	-	-	-	5
integrate-and-fire with adapt.	-	+	-	-	-	-	+	+	-	-	-	-	+	-	-	-	-	+	-	-	-	-	10
integrate-and-fire-or-burst	-	+	+		+	-	+	+	-	-	-	-	+	+	+	-	+	+	-	-	-		13
resonate-and-fire	-	+	+	-	-	-	-	+	+	-	+	+	+	+	-	-	+	+	+	-	-	+	10
quadratic integrate-and-fire	-	+	-	-	-	-	-	+	-	+	-	-	+	-	-	+	+	-	-	-	-	-	7
Izhikevich (2003)	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	13
FitzHugh-Nagumo	-	+	+	-		-	-	+	-	+	+	+	-	+	-	+	+	-	+	+	-	-	72
Hindmarsh-Rose	-	+	+	+			+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	120
Morris-Lecar	+	+	+	-		-	-	+	+	+	+	+	+	+		+	+	-	+	+	-	-	600
Wilson	-	+	+	+			+	+	+	+	+	+	+	+	+		+	+					180
Hodgkin-Huxley	+	+	+	+			+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	1200

Figure 1.3: *Results of the comparison of 11 neuron models, conducted by Izhikevich et al. (2004a). From (Izhikevich et al., 2004a), ©2004 IEEE, used with permission.*

1.2.2.3.2 Generalized Form

In 2007, Izhikevich published a generalized form of his model:

$$\begin{aligned} C\frac{dv}{dt} &= k(v - vr)(v - vt) - u + I_{syn}(t) \\ \frac{du}{dt} &= a(b(v - vr) - u) \end{aligned} \tag{1.6}$$

with the auxiliary after-spike resetting:

[4]At least of the models included in this study using the defined metrics.

$$\text{if} \quad (v \geq v_p) \quad \text{then} \quad v \leftarrow c, \quad u \leftarrow u + d \tag{1.7}$$

where v is the membrane potential, u represents a membrane recovery current, C is the membrane capacitance, v_r is the resting membrane potential, v_t is the instantaneous threshold potential, v_p is a spike cutoff value (the peak), $I_{syn}(t)$ is the input current, and a, b, c, and d are parameters of the model.

The derivation of the parameter values for various neuron types is described in (Izhikevich, 2007; Izhikevich and Edelman, 2008). Here, we provide again only the example values of main cortical neuron classes, in the Table 1.2.

	RS	FS	LTS
C	100	20	100
k	3	1	1
v_r	-60	-55	-56
v_t	-50	-40	-42
v_p	50	25	40
a	0.01	0.15	0.03
b	5	8	8
c	-60	-55	-50
d	400	200	20

Table 1.2: *Possible values of parameters of the (Izhikevich, 2007) neuron model for the main cortical neuron classes, based on (Izhikevich and Edelman, 2008).*

There are also many other spiking neuron models, with other advantages, not reviewed in this text. They can be found in (Dayan et al., 2001; Schutter, 2010; Trappenberg, 2010). Since the other models are not needed in the rest of the thesis, we do not mention them in this brief introduction.

1.2.3 Synapse Model

Synaptic connections have typically two main characteristics: *(axonal) conduction delay* and *synaptic efficacy*, frequently called *synaptic weight*. The conduction delay represents the time needed for signal transmission along the axon (and possibly dendrite of the postsynaptic neuron) and sometimes is simplified as unitary. Sometimes, also *synaptic response delays* are modelled (see e.g., (Trappenberg, 2010)), however they are much shorter and therefore often neglected. In the rest of the text, by delay we will always mean the (axonal) conduction delay.

There are many approaches to modelling a synaptic transmission: the process when an action potential from a presynaptic neuron arrives to a synapse and leads to a postsynaptic potential (PSP) on a postsynaptic neuron. Since the synaptic connections in the central nervous system are highly diverse (Schutter, 2010, chap. 6) and continually change their properties in several ways, effort to plausibly model synaptic transmission is a challenging task. In addition to that, synaptic transmission is a stochastic process (it is considered as an important source of noise in the nervous system (Schutter, 2010, chap. 6)). Therefore, all

computational models of synaptic transmission require some degree of simplifications and abstractions.

In the simplest model of synaptic transmission the input $I_{syn}(t)$ of neuron N_{post} in tick t may be computed as

$$I_{syn}(t) = \sum_{i=1,\ldots,k} w_{s_i} \tag{1.8}$$

where $s_1, \ldots, s_k$ are synapses (to neuron N_{post}), to which an action potential arrived at tick t, and w_{s_i} is the weight of synapse s_i. This means that, for all $i \in 1, \ldots, k$, presynaptic neuron of the synapse s_i fired at time $t - d_{s_i}$, where d_{s_i} is the conduction delay of the synapse s_i.

Both the synaptic weights and conduction delays may be fixed during the entire simulation (but different between synapses), or changeable (i.e., plastic). However, the delays and weight are the only two features, which make the synapses different, in this model. The model does not include any specific receptors, or synaptic types. It also does not internally include any source of noise and it does not correspond to real synapses in many respects (e.g., changeable behaviour caused by limited amount of neurotransmitter and sometimes depleted synaptic vesicles). However (also because neuro-computational modelling requires many simplifications), it is used in various models (e.g., Izhikevich, 2006; Phoka et al., 2012).

To achieve a better plausibility, the development of the synaptic conductances could be modelled. Subsequently, the input $I_{syn}(t)$ of neuron N_{post} at tick t is not a sum of synaptic weights, but PSCs (postsynaptic currents):

$$I_{syn}(t) = \sum_{i=1,\ldots,k} PSC_i \tag{1.9}$$

where PSC_i can be for instance computed as:

$$\begin{aligned} PSC_i &= w_{s_i} \cdot g_{s_i}(V_{N_{post}} - E_{s_i}) \\ \frac{dg_{s_i}}{dt} &= -\frac{g_{s_i}}{\tau_{s_i}} \end{aligned} \tag{1.10}$$

where w_{s_i} is again the weight of the synapse s_i, g_{s_i} is the instantaneous synaptic conductance, E_{s_i} is the reversal potential, V is the membrane potential of the postsynaptic neuron N_{post}, and τ_{s_i} is the time constant for the decay of the synaptic conductance after the neurotransmitter release. This approach enables to distinguish between different types of synaptic kinetics. For instance, E_{s_i} is typically 0 mV for excitatory synapses and around +80 mV for inhibitory (e.g., +70 mV for $GABA_A$ receptors and +90 mV for $GABA_B$ receptors). Another difference could be done by different settings of τ_{s_i} (e.g., small around 5–6 ms for $AMPA$ and $GABA_A$ receptors and bigger around 150 ms for $NMDA$ and $GABA_B$ receptors). This kind of model was used in (Muresan and Savin, 2007; Izhikevich et al., 2004a; Izhikevich and Edelman, 2008).

Many other and more realistic models of synaptic kinetics can be found in (Dayan et al., 2001; Schutter, 2010; Trappenberg, 2010).

1.2.3.1 Short-term synaptic plasticity

One of the simplest models of the short-term synaptic plasticity defines for each synapse a scalar factor x, by which the synapse weight is scaled, which leads to short-term depression and facilitation. The scalar factor x can be modelled by one-dimensional equation:

$$\begin{aligned} \frac{dx_{s_i}}{dt} &= \frac{(1 - x_{s_i})}{\tau'_{s_i}} \\ x_{s_i} &\leftarrow p_{s_i} x_{s_i} \quad \text{when presynaptic neuron fires} \end{aligned} \tag{1.11}$$

where τ'_{s_i} is a time constant parameter (typically around 100–200 ms) and p_{s_i} is a dimension-less parameter (typically between 0.5–1.5). Each presynaptic spike resets the value of x_{s_i} to the new value $p_{s_i} x_{s_i}$. Subsequently, x_{s_i} tends to recover the equilibrium of value 1, with the time constant τ'_{s_i}. Values of $p_{s_i} < 1$ lead to the short-term synaptic depression, whereas values of $p_{s_i} > 1$ result in the short-term synaptic facilitation. The values of τ'_{s_i} and p_{s_i} may be used to distinguish among different synapse types. This kind of short-term synaptic plasticity was used in (Izhikevich and Edelman, 2008).

For detailed reviews of short-term synaptic plasticity models see (Morrison et al., 2008; Schutter, 2010).

1.2.3.2 Long-term synaptic plasticity

A popular form of the long-term synaptic plasticity is the Spike-Timing-Dependent Plasticity (STDP) (Song et al., 2000). The main idea is that repeated arrivals of presynaptic spike shortly before the postsynaptic spike lead to long-term potentiation (LTP) of the synapse, whereas a repeated spike arrival after the postsynaptic spike leads to long-term depression (LTD). This basic idea may be implemented in many ways. We only describe the way used in the rest of the thesis. See (Sjöström and Gerstner, 2010; Schutter, 2010; Song et al., 2000) for other possibilities.

The original form of STDP describes the amount of synaptic modification F (a value, which is added to the synaptic weight) in dependence on the time Δt between presynaptic spike arrival and postsynaptic spike. This value is computed as:

$$F(\Delta t) = \begin{cases} A_+ \cdot e^{\Delta t / \tau_+} & \text{if } \Delta t < 0 \quad \{ \text{ LTP part } \} \\ -A_- \cdot e^{-\Delta t / \tau_-} & \text{if } \Delta t > 0 \quad \{ \text{ LTD part } \} \end{cases}$$

where τ_+ and τ_- are time constant parameters (typically around $20ms$) and parameters A_+ and A_+ determine the maximum amounts of synaptic modification, which occur when Δt is close to zero.

1.2.4 Multi-compartmental models

In this thesis, we deal only with single-compartmental models – models that describe the membrane potential of a neuron by a single variable (Dayan et al., 2001). The *multi-compartmental models* include the conductance properties and physical shape of the neuron, especially of its dendrites. More about multi-compartmental models can be found in (Dayan et al., 2001; Schutter, 2010;

Trappenberg, 2010) and an example of multi-compartmental model based on Izhikevich neuron model can be found in (Izhikevich and Edelman, 2008).

1.2.5 Simulation techniques

The techniques of neural network simulation may be divided into two groups: *clock-driven* (also called *synchronous*) and *event-driven* (also called *asynchronous*). In the clock-driven algorithm, all neurons are updated at every tick of a clock, whereas in event driven algorithms neurons are updated only when they receive or emit a spike. Well-arranged review of these techniques and usage in various simulators can be found in (Brette et al., 2007). In the following text, we describe the main features of both techniques.

1.2.5.1 Clock-driven approach

The main advantage of clock-driven approach is that it is flexible and easy to implement for any model. A typical implementation has the structure described in Pseudocode 1.1.

```
1. For each tick t:

   (a) For each neuron, test the spike condition and possibly
       send a spike {spikes detection}

   (b) For each neuron, process spikes that arrived in this
       tick {spikes processing}

   (c) Update state variables of all neurons (and possibly
       synapses) {state updates}
```

Pseudocode 1.1: *Typical structure of a clock-driven simulation.*

The spike condition (1a) has typically a form of $(v > v_t)$, where v is a neuron's membrane potential and v_t is a constant threshold value (a parameter specific for the neuron's type).

Sending (1a) and processing spikes (1b) may be done via a simple calendar in the form of a *circular array* (Morrison et al., 2005). This is an array of `MAX_DELAY` bins, where `MAX_DELAY` is a maximal conduction delay. Each bin contains a list of "*synaptic events*" (identifiers of synapses). When neuron N_{pre} emits a spike at tick t, we insert a synaptic event for all the synapse connections leading from neuron N_{pre} into the calendar. The position in the circular array for a synapse with delay d is calculated as $(p + d)$ mod `MAX_DELAY`, where p is the present position (t mod `MAX_DELAY`). Subsequent spike processing (1b) is as simple as to go through the bin corresponding to the current tick and for each synapse s in this bin, process the spike, which arrived in this tick through synapse s, e.g., add the weight of the synapse s to the input of the postsynaptic neuron. When all synaptic events from the bin are processed, the bin is cleared.

The last step of the algorithm (1c) consists of update of all state variables, i.e., changing state variable $X(t)$ to $X(t + dt)$, where dt is the duration of one simulation tick. The difference can be computed using Euler or Runge-Kutta (Press et al., 2007) integration methods.

Using a circular array for spikes processing is both easy to implement and effective solution. Important fact is that inserting a synaptic event into this data structure has a constant time complexity (e.g., adding an element to the start of the linked list) and removing all synaptic events from one bin has a complexity $\mathcal{O}(n)$, where n is the number of the elements in the bin. Therefore, from the point of view of asymptotic time complexity, this implementation of the clock-driven algorithm is the best one, if nontrivial conduction delays are needed.

The main disadvantage of the clock-driven approach is the possible loss of precision. First, the spike timings are aligned to a grid (ticks of the clock). Second, the spike conditions are checked only at the ticks of the clock, implying that some spikes might be missed. Both drawbacks can be moderated by a shorter clock tick, which, on the other hand, decelerates the computation. The exact form of the clock-driven simulation may be reached using a priority queue (e.g., "*calendar queue*" by (Brown, 1988)) and storing the exact times of spikes and synaptic events. This approach is a sort of hybrid between clock-driven and event-driven approach.

1.2.5.2 Event-driven approach

The event-driven approach does not use any discrete time ticks and updates the variable states only when spikes are emitted or received. This brings two main advantages. First, the spike timings are computed exactly. Second, it may lead to higher computational speed, because only active changes (when a spike is emitted or received) are computed, instead of many small updates for every neuron in every tick as it is in the clock-driven approach. The second advantage takes place especially in the case of a small number of synaptic events (e.g., in sparse networks with low mean firing rate).

This approach requires two main features from the model. First, we must be able to calculate the neuron state at any given time (i.e., we need the explicit solution of the differential equations). Second, if the synaptic interactions are not instantaneous, we need the function that maps the current state of the neuron to the timing of the next spike (possibly $+\infty$ if there is none). These demands make the asynchronous approach very inflexible. Generally, only simple models (such as LIF) can be used in combination with event-driven approach. In addition to that, the implementation may be much more complex than in the previous approach. Here, we need to store events and their times in some kind of priority queue, such as binary heap, Fibonacci heap (Cormen et al., 2001), calendar queue (Brown, 1988), etc. The stored events are not only synaptic events as in the former approach, but also spike events. Moreover, each spike or synaptic event can influence the other planned events. They must be recalculated and possibly moved inside the priority queue. Overall, the queue management is not only difficult to implement, but may be also time consuming, which can reduce the computational advantage of this approach.

1.2.6 Inputs, heterogeneity, noise and spontaneous activity

Besides the dynamics of neurons and synaptic transmissions, precise structure of connections between neurons and other already mentioned network features, there are many other important features of a network model. We shall mention at least four of them: inputs, network heterogeneity, noise and spontaneous activity.

According to the purpose of the network model, various kinds of external inputs are used. The external inputs can represent inputs from parts of the central nervous system, which are not modelled, or a current injected by an electrode. All these inputs are usually simply added to the internal inputs from synapses (variable I_{syn}).

Biological diversity leads to almost never deterministic and homogenous systems. Since models of neural networks are typically based on uniform templates of neurons and synapses, it is important to introduce heterogeneity in them (Schutter, 2010, chap. 13). It can help to prevent deadlocks (due to disturbed symmetry) and it extends the processing capacity (Eliasmith and Anderson, 2003). This heterogeneity may be introduced by stochastic connectivity, variations in neuronal model parameters or randomization of synaptic weights. Alternatively, the heterogeneity can be reached by an external noise. Noise can be part of the inputs, or can be implemented using stochastic spike threshold instead of fixed one (Jolivet et al., 2006).

External noise added to inputs can also help the emergence of the spontaneous activity, which is a phenomenon observed in real neocortical networks. It is defined as ongoing dynamics that is not triggered by external events (Arieli et al., 1995; Muresan and Savin, 2007). We discuss the possible solutions of introducing spontaneous activity into a model in the Section 3.5.5. The main reason why it is not reviewed already in this chapter is that this topic is not so well researched and we did not consider it as the part of the fundamentals of the computational neuroscience. However, it is very interesting and important factor.

1.3 Concluding remarks

The chapter is concluded by the summary of the main elements of the process of creating a computational model of neuronal network.

1. Definition of model purpose

   (a) The model should be created with a specific intent, such as biological hypothesis, verification of a conceptual model, test of the "what if" questions, which are difficult or impossible to test in the real network, etc.

2. Choice of a model and a simulator

   (a) This step is highly dependent on the previous step. The authors of the model should decide what contrasting features of the model are the most important: plausibility (on the level of molecules, or neurons with properly modelled ion channels and synapses with properly

modelled synaptic conductance, neurotransmitters and receptors, or point neurons, or neuron populations, etc.), precision (e.g., short time ticks in the clock-driven simulation), or computational efficiency, which allows to model large-scale networks and compute long experiments.

3. Setting model parameters

   (a) The authors should consider the following features:

      i. Neuron types and for each of them the neuron model parameters settings

         A. The typical neuron types (classes) are well described and many models suggest how to set their parameters for specific neuron types.

      ii. Synapse types and, for each of them, the synapse model parameters settings

         A. The basic synapse types are also well described and recommendations to their parameter settings exist.

      iii. Neuron numbers and arrangement in the network according to their type

         A. Numbers of neurons of specific neuron types may be acquired from published literature for many parts of the nervous system and many species. However, acquirement of the arrangement of many neurons in large parts of the nervous system (and especially brain) is still difficult. In addition to that, less numerous neuron types (e.g., some inhibitory types) are often difficult to observe (Wu et al., 2011). Then a stochastic and approximate approach is needed.

      iv. Network connectome, i.e., complete description of the connections between neurons (including the synapse parameters)

         A. Obtaining neuron connectome is difficult. The in vivo techniques are generally constrained to some areas (e.g., superficial layers of brain), or limited in number of neurons and variety of neuron classes which may be observed. The data obtained *in vitro* are morphologically incomplete, owing to the slicing of the brain, and need additional heuristic reconstructions (Schutter, 2010, chap. 9).

      v. Network inputs and noise

      vi. Other sources of the spontaneous activity

      vii. Initial settings of all parameters + rules of their dynamics (they are typically determined by the neuron and synapse models)

      viii. Network outputs, i.e., what variables are monitored for the (typically retrograde) analysis

         A. This should be selected according to observed real data, with which we can the model results compare

4. Model validation

(a) This can include the validation of the static features (such as verification, that the resulting structure and scaling of the network is reasonable) and dynamic features. The latter is much more complex and it may be very difficult to propose a suitable set of validation experiments and metrics and especially perform a proper analysis of the results and explain the reasons of the negative results. Some advices to model validation may be found in (Schutter, 2010, chap. 13).

5. Proposition of experiments

   (a) After the model validation, other experiments may be designed and run. This part is dependent of the purpose of the model. The experiments run on the model (and therefore called *in silico* experiments) may have a structure similar to the experiments *in vivo* or *in vitro*.

6. Analysis of experiments

   (a) This part has similar pitfalls as the analysis of the model validation. Ideal analysis of experiments contains not only the results, but also suggestions of the reasons for the results. This may be very difficult in large networks with complex dynamics, where many opposite factors are present, such as excitation and inhibition, short-term facilitation and short-term depression, long-term potentiation and long-term depression and many others.

The whole process is demanding but also challenging in many aspects. It requires solid background and knowledge from mathematics, biology, neuroscience, and computer science. It is a quickly developing field which requires knowledge of both old (older than 5 years) and new (up to 5 years) findings and publications. The whole modelling process is time consuming both for the modellers, and for hardware. As (Schutter, 2010) said: "*A common misconception is that a modeling project can be achieved quickly, which is almost never the case. Therefore the experiment-modeling cycle is best implemented as a team effort with close interaction of all partners.*" (Schutter, 2010, pp. xi).

2. Simulator: Requirements and Related Works

This chapter defines the requirements on the simulator, reviews related works and explains reasons for developing a new simulator software.

2.1 Requirements

Before listing the overall requirements on the simulator, we will briefly describe the choice of neuron model. We had the following requirements on the neuron model:

1. Reasonable plausibility (however, single-compartmental model is sufficient).
2. Computational efficiency, which will allow simulating a network with thousands of neurons in experiments lasting minutes to hours of model time.
3. Possibility of different behaviour according to the neuron type, such as RS, FS, or LTS neuron types.

Due to the reasons described in the Section 1.2.2.3, we chose to use the Izhikevich neuron model, because it fulfils all the requirements and to our knowledge, there is not any other model which would better satisfy these requirements. In the concrete, we chose the generalized form, because it is newer (also the Izhikevich has since then used the generalized form), and it allows for more flexible settings.

In the rest of the section we will list the requirements on the simulation software. We divided them into two groups: requirements based on needs of the AC model and requirements of other simulator features and control. The reasons for the first group of requirements are in detail explained in the model part of the thesis, in the Section 6.3. The requirements based on the model needs are:

(R1) The generalized form of the Izhikevich neuron model.

(R2) Synaptic connections with specified synaptic conduction delays and changeable weights, formed by the long-term plasticity, e.g., STDP.

(R3) Sufficient computational precision.

(R4) Sufficient flexibility in network structure definition: e.g., different probabilities of synaptic connections between specific neuron types in specific dependence on distance ((Izhikevich and Edelman, 2008, for instance as it is in).

(R5) Sufficient flexibility in inputs definition: e.g., each tick different neurons will be stimulated with a value specified for each neuron.

(R6) A mechanism that would lead to spontaneous activity.

The requirements, which do not concern the model, but rather features and control of the simulator, are:

(R7) Sufficient tools for simulation analysis, i.e., at least:

1. Analysis of spike trains (spike times raster plot, development of mean activity of neurons of the same neuron type, etc.).
2. Analysis of synaptic weights (development of mean weights over time).
3. Analysis of oscillations and synchronicity (e.g., to measure presence of waves).
4. Visualisation of the network activity, preferably in 3D view (spikes, membrane potential of neurons, etc.).

(R8) Practical (user-friendly) settings of the model and experiment definition with a possibility of automatic batch processing of several experiments.

(R9) Effective implementation of the whole simulation, preferably with a possibility of parallel computing.

(R10) In the case that an existing framework does not fulfil all the previous requirements as built-in features, then the framework must be easily extensible and allow implementing these features. This means that the framework must be well documented, with transparent and well-arranged code.

(R11) The framework should run on Windows.

In the rest of this chapter, we will refer to these requirements as R1–R11.

2.2 Related Works

Initially, we searched for an existing tool, which would fulfil all of our requirements. During the last twenty years, a large number of neuronal simulators were developed. We provide here just the basic description. For more details, see either the relevant publications, or the following reviews: (Schutter, 2010; Brette et al., 2007).

The best known and most traditional group contains large simulation environments with general use, but with primary focus on the HH neuron types: NEURON (Hines and Carnevale, 1997, 2001), GENESIS (Bower et al., 1998; Bower and Beeman, 2007), SNNAP (Ziv et al., 1994; Baxter and Byrne, 2007), and neuroConstruct (Gleeson et al., 2007), which also allows generating models with runnable in NEURON or GENESIS.

Another group contains tools oriented more on single-compartment and point neuron models, with specific focus on computational efficiency through distributed computing: NEST (Morrison et al., 2007), CSIM (Natschläger et al., 2003) and its parallel version PCSIM (Pecevski et al., 2009), and NCS (Brette et al., 2007).

Tool specialized for efficient simulations of large-scale multi-compartmental models based on HH formalism, SPLIT (Hammarlund and Ekeberg, 1998), is rather a pure, generic neural simulation kernel, without support for analysis of results, or graphical interface.

Simulation framework Mvaspike (Rochel et al., 2003) is exceptional for its event-driven approach (advantages and disadvantages of event-based modelling are reviewed in (Brette et al., 2007)).

There are also smaller simulators specializing in efficient simulations of networks with Izhikevich neurons: GPU-SNN (Nageswaran et al., 2009), NeMo (Fidjeland and Shanahan, 2010), and Neocortex (Mureşan and Ignat, 2004).

A large range of tools with more general use exist. For instance XPAUT (Ermentrout, 2004) is a general numerical tool for simulating, animating, and analysing dynamical systems. Rather than pure simulator, it is a tool for understanding the dynamical systems. Another tools focus on related domains of applicability: systems biology (Catacomb (Cannon et al., 2003), MOOSE), biochemical and generally molecular reactions (E-Cell (Tomita et al., 1999), MesoRD (Elf and Ehrenberg, 2004), STEPS, StochSim (Le Novere and Shimizu, 2001)), movements and reactions of molecules within and between cells (MCell (Stiles and Bartol, 2001)), biochemical signalling networks (Kinetkit (Bhalla, 2002)). A review of these tools can be found in (Schutter, 2010).

Finally, a relatively new simulator, Brian (Goodman and Brette, 2008, 2009; Brette et al., 2007), is written in Python, focused on minimising user's learning and development time rather than simulation time.

2.3 Discussion

The first group of simulators (NEURON, GENESIS, SNNAP, and neuroConstruct) focuses on HH neuron types, which was not our aim. Many features (R1, R4–R7) would need additional programing and extending the original framework, which would be difficult and time consuming; which means that these simulators do not ideally fulfil R8 and R10.

The second group of simulators contains the most promising ones, except NCS, which does not provide any Izhikevich neuron model. Both NEST and PCSIM (the parallel version of CSIM) provide many features, computational efficiency and make an impression of well organised projects. However, neither of them is being developed for Windows, which makes them very difficult to run under Windows with no guarantee of smooth running (PCSIM "*was not tested under Windows*" (Pecevski, 2008) and "*it is difficult to compile NEST natively under Windows*" (Initiative, 2013)). In addition to that, neither of them contains built-in generalized form of the Izhikevich neuron model. To summarize our observations, they have problems with requirements R1, R11 (and possibly also R4–R6 and R10).

The SPLIT simulator would need many extensions (e.g., due to R7) and most importantly it does not fulfil requirements R2 and R11.

The Mvaspike simulator does not fulfil R9 and R11.

The group of smaller simulators focusing on Izhikevich neuron model are close to our aims. However, although they achieve high speed of simulation, for example by using Graphics Processing Units (GPUs), none of them supports the generalized form of Izhikevich neuron model and are not easily suitable for extensions. The Neocortex Neural Simulator does not seem to have a sufficient support and freely available codes and therefore does not fulfil R8 and R10. GNU-SNN and NeMo may lead to insufficient precision due to single-precision floating point ari-

thmetic (Nageswaran et al., 2009; Fidjeland and Shanahan, 2010) and therefore may not fulfil R3. In addition to that due to specialized low-level enhancements, the extensibility of these frameworks may be rather demanding, or even impossible, and therefore they fail to meet R10.

The penultimate group simulation frameworks does not focus on spiking neural networks and thus does not fulfil, among others, R1 and R2.

The last simulator, Brian, is focused on user-friendly control via Python scripting language. The whole project is well arranged and contains many tutorials. In addition to that, after the general research of existing simulators we asked Dr. Eugene Izhikevich, whether he knows any existing tool supporting his neuron model that would fit our usage, and he recommended us Brian. However, since the simulator is written in Python and does not focus on performance so much, we considered necessary to test the performance in the use with typical features of our network. We decided to test it on network from (Izhikevich, 2006) with Izhikevich neuron model, STDP and conduction delays. (The tested network was exactly the network described in (Izhikevich, 2006).) Besides an implementation in Brian, we used the original implementation in Matlab and C++ provided by Izhikevich (2006) and implemented the same in Java.

The results are summarized in the Table 2.1. The fastest implementations were in C++ and Java. Brian was ten times slower and Matlab even seventy times slower. According to personal communication with the authors of Brian, we verified that the lack of speed was not caused by incorrect usage of the simulator, or by mistakes in formulation of the model. The difference between running an experiment for one day versus ten days (or even more than two months) is so considerable, that we decided to use neither Brian, nor Matlab. The difference between C++ and Java was relatively negligible.

Brian	Matlab	C++	Java
90s	510s	7.3s	6.9s

Table 2.1: *The results of the performance of four implementations of the network with Izhikevich neuron model, synaptic delays, and STDP. Duration was averaged over several measurements (the differences between tests were negligible).*

2.4 Outcome

In conclusion, none of the researched simulators fulfils all the requirements. Some of the simulators may be possibly used after large extensions. However, extending a software requires good knowledge of the software and all used languages (also the special scripting languages developed by the software's authors). Misunderstanding the code may lead to errors, which are difficult to trace and debug.

However, our aim was relatively narrow (we did not need a general and all-round framework with a wide range of neuron and synapse models) and therefore we considered developing a new simulator instead of extending an existing one to be the best solution.

For the simulator language, we chose Java for the following reasons: code transparency, speed of development, performance (based on the test described in

the previous section), excellent IDE support (e.g., Netbeans IDE (Oracle, 2013)) and personal preferences.

We developed the whole simulator in Java, except the analytical part. For the analysis of results, we used combination of Java and Matlab; the reasons are listed in the Section 3.6.1.2.

3. Simulator: Methods

This chapter describes the software part of the thesis: the simulator SUSNOIMAC (Simulator Using Spiking Neurons Originally Intended for Modelling Auditory Cortex). Although the main domain of the simulator should be models of the AC (one possible model is described in the Chapter 7), we designed the simulator independently on the auditory features. Thereby, this chapter does not include anything auditory specific with two exceptions. First, there are occasional examples of auditory motivation for some simulator functionality. Second, the Section 3.6 contains the description of the analytical module which may be used for the analysis of both general features and auditory specific features.

The description of the simulator starts with the high-level aspects, such as design and architecture, proceeds to low-level aspects, such as individual modules and used algorithms. The whole chapter follows a uniform structure: when more solutions of a particular sub-problem are possible, we first describe the possible solutions, then discuss their advantages and disadvantages, and finally state the chosen solution. Although it makes the text slightly less flowing, we chose it due to lucidity of the explanations of the chosen solutions.

3.1 Language of model definition

3.1.1 Language of model definition: possible choices

The high-level design of the simulator architecture is fundamentally influenced by the relation of the model and the computational part of the simulator. Before describing the possible relations, we should clarify the meaning of the term "model". In the general meaning (in neuroscientific computational modelling), the model should define:

- the structure of the modelled network: the neurons, their locations and possibly the neuron types (or populations), and the connectivity (synapses) between neurons
- the dynamics of neurons and synapses: the used neuronal models and their parameters for individual neuron and synapse types
- the inputs: the external stimuli, which do not emerge from the network dynamics (and eventually also the network outputs)

The dynamics can be obviously only such that is supported by the simulator, in contrast to the network structure and inputs, which can be typically much more independent of the simulator. For our models, we did not have demands on diversity of dynamics: just one model of neurons (the generalized Izhikevich neuron model (Izhikevich, 2007; Izhikevich and Edelman, 2008)) and simple model of synapses (Izhikevich, 2006), where each synapse has its delay and weight, which is adjusted by the STDP long-term plasticity. Since these properties are the same for all of our models, we will use the meaning of the model in the rest of the thesis as:

- the network structure (and parameters related to it)
- parameters of the simulation (this includes also parameters of STDP)
- the inputs (and parameters related to it)

It is generally considered that a model description (or definition) and the computational core of the simulator should be separated. However, the approaches to how and where to define the model, are different. Here is the list of the most typical solutions:

1. XML-based declarative standard, such as NetworkML, which is part of the NeuroML project (Goddard et al., 2001; Gleeson et al., 2008).
2. Imperative scripting language, such as PyNN (Davison et al., 2008) written in Python.
3. Custom plain-text format, such as CSV (comma-separated values) used for numerical parameters organised into tables.
4. Custom scripting language – for example NEURON uses Hoc and NMODL (Carnevale and Hines, 2006), NEST uses SLI (Gewaltig and Diesmann, 2007), and GENESIS uses different language SLI (Bower et al., 1998).
5. Model definition written in the language of the simulator.

The advantages and disadvantages are summarized in the Table 3.1. We can divide the solutions according to their approach: declarative and imperative (or programmatic). The declarative approach is typically simpler and leads to well defined behaviour. On the other hand, it has limited flexibility for defining complex networks (such as complexly defined connectivity, which works with probability – an example is our AC model in the Chapter 7. We knew that our model's definition needs to be more complex than this declarative approach enables.

The opposite approach is imperative, using a scripting or compiled language. This solution is certainly more flexible and complex. A disadvantage common for all solutions of this approach is the need to learn the language. Therefore it is more practical to use an existing and used language by users. The scripting languages (such as Python) seem to be suitable for this task. Their code is compact and usually pretty comprehensible. On the other hand, if the whole simulator is written in a scripting language, the speed performance of the simulator may be limited. If the simulator is written in another language, the whole framework is less uniform and needs connecting the both languages (however, this is common practise in bigger SW frameworks).

3.1.2 Language of model definition: outcome

For our simulator, we decided to use combination of the fifth and the third solution. The models are therefore also defined in Java. However, their numerical parameters are loaded from plain-text files: properties and CSV files. This solution is simple, easy for development and testing, uniform, easy to extend and well sufficient for our aims. However, implementation of some commonly used standard (be it declarative or imperative approach) could be a meaningful future work.

No.	Name	approach	example	Pros	cons
1	XML-based standard	declarative	NeuroML	simplicity, well-defined behaviour, compatibility between simulators	limited flexibility and expressive power
2	existing scripting language	imperative	PyNN	flexibility, compatibility between simulators	need to learn the language
3	custom plain-text format	declarative	CSV	simplicity	limited flexibility and expressive power
4	custom scripting language	imperative	NMODL, SLI	flexibility	need to learn the language, incompatibility between simulators
5	language of the simulator	imperative	Java	high flexibility, uniformity, easy to write and understand for programmers	need to learn the language, incompatibility between simulators

Table 3.1: *A comparison of different approaches to the model definition.*

3.2 Architecture design

3.2.1 Architecture design: possible choices

The main parts of the simulator may be divided into Simulation Core (the computational part) and three model parts: Network, Inputs and Analysis; see Figure 3.1. There are three main possibilities, how to implement the relation between Simulation Core and model parts in Java:

1. Using hard-coded classes with the model definition, where only possibility of using another model with the simulator is to change either these hard-coded classes, or the simulator classes.

2. Using one or more interfaces, which define the compulsory methods of the model.

3. Using abstract classes, which define the compulsory methods and data structures of the model.

3.2.2 Architecture design: discussion of possible choices

The first solution is unequivocally the least structured, flexible and usable. The second and third solutions are relatively similar to each other. Using interface

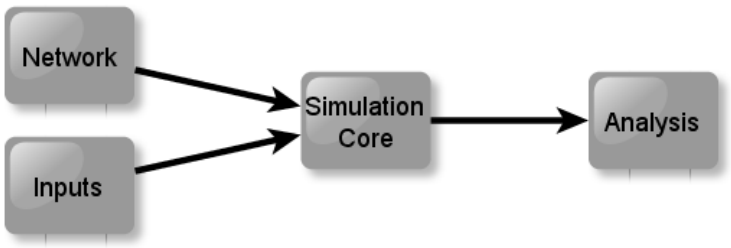


Figure 3.1: *The high-level structure of the architecture design of the simulator.*

instead of abstract classes is useful in those cases when a class would need to have more parent classes (i.e., be derived from more ancestors), in that case it can easily just implement more interfaces. On the other hand, abstract classes can contain fields that are not static and final, and they can contain implemented methods. Since we do not need extending multiple classes, but we can utilize the inherited data structures, we chose the third solution.

3.2.3 Architecture design: outcome: Hierarchical-modular architecture

The architecture of the SUSNOIMAC tool consists of four modules: the simulation core, the network module, the input module, and the analysis module. The simulation core provides the main simulating work, implementation of neuronal model and synapse dynamics, etc. It uses the network defined in the network module and inputs defined in the input module. During the simulation, specific values can be monitored and stored for the backward analysis, which is provided by the analysis module.

We can compare the architecture to the structure of the onion (see Figure 3.2): the simulation core of the simulator corresponds to the core of the onion. The surrounding layer of the core contains three abstract modules: network, inputs, and analysis module. In these abstract classes, it is defined, which data structures and methods must the concrete modules provide. The concrete modules (which extend the abstract ones) are in the peripheral layer. Each of them can use other own classes, methods and data structures.

This architecture provides an excellent flexibility in defining the model (such as using specific random generators, or complex conditions and algorithms in network generating). At the same time, the code of the model is strictly separated from the simulator code.

In the rest of this section, the individual modules are described. The section is concluded by a list of used external libraries.

3.3 Network module

This module contains the entire definition of the neural network structure. The basic units are neurons and synapses between neurons. The compulsory parts of the network module are: list of neurons, data structure for the network structure defined by synaptic connections between neurons and the parameters of these

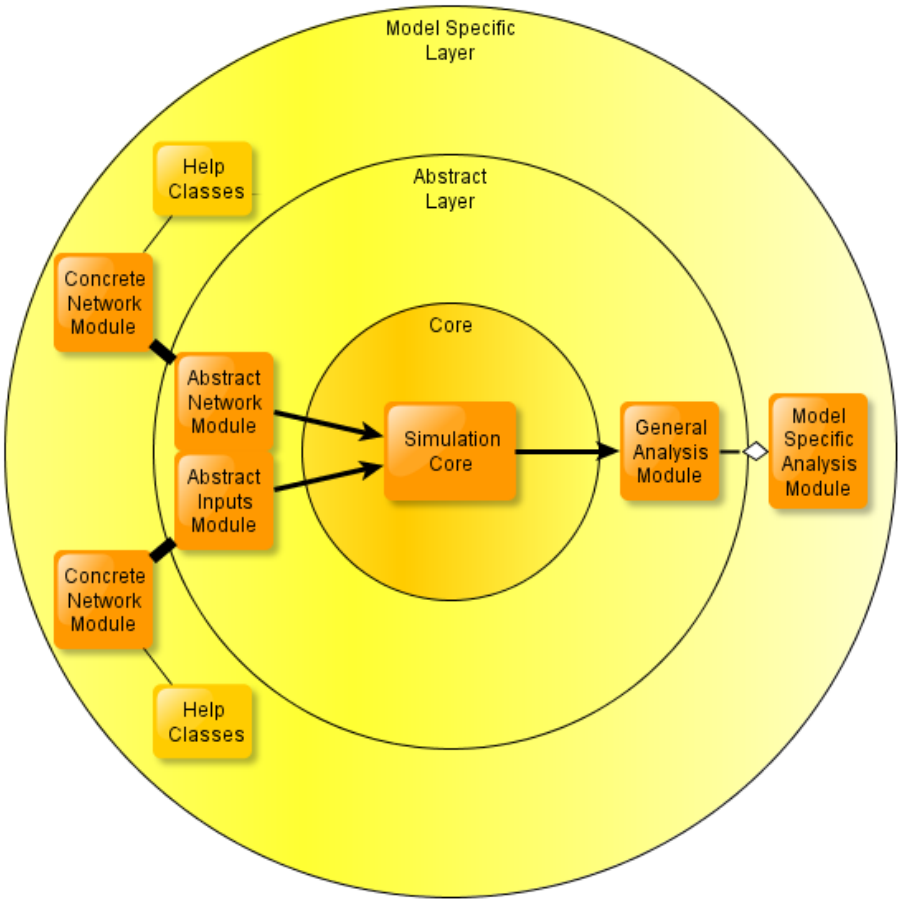


Figure 3.2: *The resulting high-level architecture of the SUSNOIMAC simulator. The Simulation Core and Abstract Layer are common for all models. In the Abstract Layer the abstract classes of the Network and Input Module are defined. In the Model Specific Layer, their concrete versions are implemented.*

connections (we call them simply "synapses"), data structure for the network inputs, and finally several general attributes of the network. Since we will refer to these data structures later, it is necessary to describe them somewhat more in detail.

3.3.1 Neurons and synapses

Each neuron must have the non-dynamic attributes, which are listed in the Table 3.2.

Each neuron has the dynamic variables, which are listed in the Table 3.3.

Although the neurons may be internally arranged in groups, populations, brain centres, layers, columns and other structures, the abstract network module captures only two types of these arrangements: layers and neuron types. Neurons with the same neuronal type may represent one neuron group or population (the terminology is not consistent in the literature). Neither layers, nor neuronal types influence the course of simulation; they are just used in the analyses. It is often useful to see results from neurons of one layer or of a single neuronal type.

Two neurons may be connected by one or more synapses. Each synapse has the non-dynamic attributes, which are listed in the Table 3.4.

name	type	description
`neuronNumber`	integer	unique number
`neuronTypeNumber`	integer	identifier of the neuron type
`location`	Location	3D coordinates in the network
`layer`	integer	number of the layer, where the neuron is located
`excitatory`	boolean	neuron must be either excitatory, or inhibitory
`Cap, k, vr, vt, vp, a, b, c, d`	double	parameters of the Izhikevich neuron model

Table 3.2: *The non-dynamic attributes of each neuron. The neurons must be numbered from 0 to* `N_NEURONS`*, where* `N_NEURONS` *is the number of all neurons and is a general attribute of the network module.*

name	type	description
`v`	double	variable of the Izhikevich neuron model
`u`	double	variable of the Izhikevich neuron model
`LTD`	double	variable of the STDP
`LTP`	array of double	variable of the STDP

Table 3.3: *The dynamic variables of each neuron. These variables are initiated in the network module; however their following values are controlled by the simulation core.*

name	type	description
`fromNeuronNumber`	integer	number of the presynaptic neuron
`toNeuronNumber`	integer	number of the postsynaptic neuron
`delay`	integer	duration of the synapse delay in ms

Table 3.4: *The non-dynamic attributes of each synapse. The delay must be a positive number from the interval* $\langle 0,$ `MAX_DELAY`$\rangle$*, where* `MAX_DELAY` *value is a general attribute of the network module.*

Each synapse has the dynamic variables, which are listed in the Table 3.5.

name	type	description
`weight`	double	the weight of the synapse
`weightDerivate`	double	derivation of the weight synapse

Table 3.5: *The dynamic variables of each synapse. These variables are initiated in the network module; however their following values are controlled by the simulation core.*

3.3.2 Data structure for the network structure

The data structure of the network may significantly influence the simulation performance. Therefore we will describe the main possibilities.

3.3.2.1 Data structure for the network structure: possible choices

The network can be viewed as the oriented multigraph from the field of graph theory (e.g., in Matousek and Nesetril, 1998): neurons correspond to vertices and synapses correspond to edges between vertices. Since we allow more synapses between one pair of neurons, we call the structure a multigraph, instead of a graph. If two neurons are connected by a synapse, we call them neighbours.

Effective data structures for storing graphs and multigraphs are well researched. When a fast access to the neurons' neighbours is important, the structures such as adjacency matrix, are unsuitable, because there the access costs $\mathcal{O}(N)$, where N is number of all neurons, instead of $\mathcal{O}(M_X)$, where M is number of neighbours of the neuron X. On the contrary, data structures, such as Adjacency list, or Incidence list, are more suitable.

3.3.2.2 Data structure for the network structure: outcome

Since in simulation a fast access to the neurons' neighbours is important (during spike propagation and STDP), we decided to define the network as a list of neighbours for each neuron. This means that the network module defines successors of each neuron as a linked list.

3.3.3 Input neurons

Different neurons may be influenced by different input stimuli. This can mean the stimuli from different brain centres, which are not modelled. It can also mean noise (of arbitrary meaning). In the auditory cortex, the main inputs are inputs from thalamus, arranged into bands.

The number and types of these stimuli are defined in the input module. However, the inputs are tightly related to neurons, because some neurons may have an external input and others not. Therefore there must be a connection between input module and network module. This connection is provided by the data structure of the network module, which for each input type (here called band [1]) contains a linked list of neurons that are stimulated by inputs of this input type. Thanks to this mechanism, the input module does not need to work with neurons directly, but only with the input types.

3.3.4 General network attributes

The network must have the non-dynamic attributes, which are listed in the Table 3.6. Except the last two attributes, the values could be also derived from the other data structures. The `MAX_DELAY` is used in the simulation core (e.g., in the calendar and STDP implementation). The `MAX_SYN_WEIGHT` is also used in the simulation core to cut the synapse weights, which would overflow this boundary.

[1] Although the name is aurally inspired, its usage is totally general and allows defining lists for input types of arbitrary meaning.

name	type	description
N_NEURONS	integer	number of all neurons
N_SYNAPSES	integer	number of all synapses
N_BANDS	integer	number of all bands (input types)
N_LAYERS	integer	number of all layers
N_NEURON_TYPES	integer	number of all neuron types
MAX_DELAY	integer	maximal synaptic delay
MAX_SYN_WEIGHT	double	maximal synaptic weight

Table 3.6: *The general non-dynamic attributes of the network. There are several other minor data structures (such as list of neuron types and their features, or neuron layers and their features), which are not so important, and they are described in the programmer documentation.*

3.4 Input module

Typical network needs some (at least initial) input to stimulate the activity. This input may lead from sense receptors or other parts of the nervous system. At the same time, the network may be stimulated by some type of internal input in the form of organised or random noise to achieve a spontaneous activity of the network in the absence of external stimuli. This topic is discussed in the Section 3.5.5.

We decided to use a simple solution, which allows all these requirements. In the network module, lists of neurons of specific input classes are defined. These classes are internally called "bands". Subsequently, for each step of the simulation, one input is defined by the input module. The input must have an `INPUT_NUMBER` attribute. The meaning of this number is entirely left on the model's author. The `INPUT_NUMBER` does not influence the simulation itself, but it is an essential information during the analysis (e.g., for receptive fields reconstruction).

There are two possible ways how to deliver the inputs to the simulation. First, input module could create an array indexed by time ticks filled with inputs (one input for each time tick) and deliver the array to the simulation core at the start of the simulation. Second, the input module could implement a method, which would return the input relevant for the given time tick.

The first approach has a main disadvantage in unnecessary memory demands. This is true especially in the case of long experiment (possibly with simple and repeated inputs). For this reason the second approach was chosen.

3.5 The simulation core

The simulation core is the basic part of the simulator. It uses a network created by the network module, simulation parameters loaded from the settings files (see the Section 3.5.6) and runs the simulation with the inputs generated by the input module. During the simulation, specific variables may be monitored (again according to the settings loaded from the settings files).

In this section, the main parts of the simulation core will be described. This means the choice of the simulation technique, pseudocode of the main simulation

algorithm, several specific features (alternative reality and spontaneous activity), computational optimizations, and description of the settings files.

3.5.1 Choice of the simulation technique

Since the event-driven simulation technique is not overly suited for the Izhikevich neuron model (Brette et al., 2007), we decided to use the clock-driven approach (1.2.5). In addition to that, we later measured ratio of the time spent on the step with state updates versus the time spent on event steps (spike detection and processing) and the state update parts took much less time (e.g., 10 times less) than the events steps. Therefore the event-driven technique would not bring much (or any) gain in computational time.

As the data structure for event storage, we first chose a circular array of linked lists (see Figure 3.3), described already in the Section 1.2.5. We tried also other data structures (such as Fibonacci heap), but they were less effective. Afterwards, we implemented an improved version of the circular array, which helped both with time and memory consumption. It is described in the section with improvements 3.5.4.

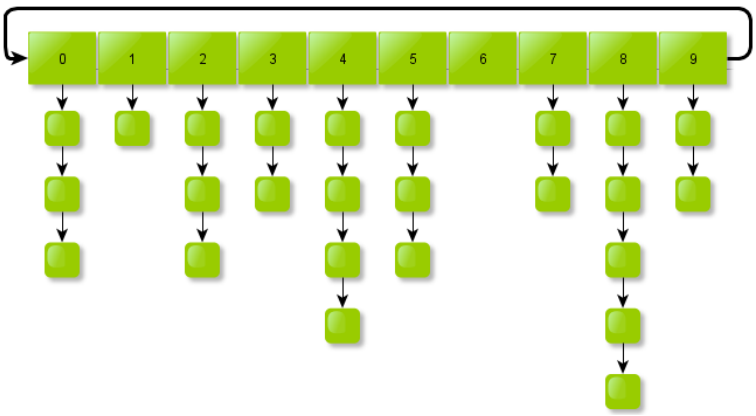


Figure 3.3: *The data structure of the calendar (for storing synaptic events) in a form of a circular array of array lists.*

3.5.2 Main simulation algorithm

The simulation core implements both the original (Izhikevich, 2003) and the generalized (Izhikevich, 2007) forms of Izhikevich neuron models and STDP in the form described in the Section 1.2.2.3. The code is based on code and algorithm structure from (Izhikevich, 2006) and (Izhikevich and Edelman, 2008). Therefore we do not analyse the reasons for structure and particular steps of the algorithm. Regardless, we explain the main steps (but details and reasons can be found in (Izhikevich, 2006) and (Izhikevich and Edelman, 2008)).

To ensure a numerical stability, each neuron is simulated with a time step of 0.5 ms using the first-order Euler method (see the Pseudocodes 3.5 and 3.6). The time step of the clock is 1 ms. This setting is often used in combination with the Izhikevich neuron model (Izhikevich, 2003; Izhikevich et al., 2004a; Izhikevich, 2006; Muresan and Savin, 2007; Nageswaran et al., 2009). Inhibitory synapses (synapses from inhibitory neurons) are not plastic, whereas excitatory synapses are evolved according to the STDP rule. The synaptic delays are fixed and set by

the network module and their values are limited to the interval ⟨1, MAX_DELAY⟩. As it is used in (Izhikevich, 2006), the synaptic weights are not changed directly. Instead, their derivatives are changed and weights are updated once a second according to the rule:

$$\begin{aligned} \texttt{weight} &+= \texttt{WEIGHT\_INCREASE} + \texttt{weightDerivate} \\ \texttt{weightDerivate} &*= \texttt{WEIGHT\_DER\_MULT} \end{aligned}$$

where WEIGHT_INCREASE describes activity independent increase of synaptic weight needed to potentiate synapses to silent neurons (Turrigiano et al., 1998; Desai et al., 2002). The synaptic weights are restricted in the permitted interval ⟨0,MAX_EXC_W⟩ for excitatory weights and ⟨-MAX_INH_W, 0⟩ for inhibitory weights. All WEIGHT_INCREASE, WEIGHT_DER_MULT, MAX_EXC_W, and MAX_INH_W are general parameters of the simulation (for the overall description of used parameters, see 3.5.6).

The main algorithm is described by the Pseudocode 3.1. The main steps of the algorithm are described independently: Detection of Spikes: 3.2, Propagation of Spikes: 3.3, Update of States: 3.4, and End of Second: 3.7. The part Generation of Inputs is by default relatively simple, since it contains only setting of current inputs, which are generated by the input module.

For a better comprehension, we will explain the STDP part of the pseudocode:

1. In the step 1e of the Detection of Spikes 3.2, an excitatory synapse is potentiated by the value of LTP of the postsynaptic neuron at the time before `synapse.delay`. This means that if, at that time, or shortly before that time, a spike in presynaptic neuron was emitted, then this spike possibly contributed to the postsynaptic spike and the synapse will be potentiated up to the maximal value LTP. The longer from the presynaptic spike it was, the less will be the synapse potentiated.

2. In the step 1(b)i of the Propagation of Spikes 3.3, excitatory synapse is depressed by the current LTD value of the postsynaptic neuron. If the postsynaptic neuron fired a short time ago, the LTD value is high. This means that the presynaptic spike arrived shortly after the postsynaptic spike, i.e. "late", and the synapse is markedly depressed. The longer the time from the postsynaptic spike it is, the less will be the synapse depressed.

```
Main simulation algorithm:

  1. Initialization

  2. for each second s

     (a) for each tick t in s

          i. Generation of Inputs
         ii. Detection of Spikes
        iii. Propagation of Spikes
         iv. Update of States

     (b) End of Second
```

Pseudocode 3.1: *This pseudocode describes the main steps of the simulation core process. The steps written in* Small Caps *mean larger parts of the code and are described independently in Pseudocodes* Detection of Spikes*: 3.2,* Propagation of Spikes*: 3.3,* States Update*: 3.4, and* End of Second*: 3.7, and are also referred later from the text.*

```
Detection of Spikes:

  1. for each neuron:
     if (neuron.v >= neuron.vt)      {spike detected}

     (a) neuron.v = neuron.c
     (b) neuron.u += neuron.d
     (c) neuron.LTD = PARAM_LTD
     (d) neuron.LTP[t+MAX_DELAY] = PARAM_LTP
     (e) for each excitatory synapse to neuron
          i. fromTime = t-synapse.delay-1
         ii. synapse.weightDerivate +=
             synapse.from.LTP[fromTime+MAX_DELAY]
     (f) for each synapse from neuron
          i. insert into calendar event synapse at time
             t+synapse.delay
```

Pseudocode 3.2: Detection of Spikes *part of the main algorithm. The notation "`neuron.v`" means attribute `v` of the variable `neuron`. If a spike is detected (membrane potential reached its apex), the membrane potential is reset to parameter `neuron.c`, membrane recovery is incremented by parameter `neuron.d`, and STDP variables are reset to their extreme values. In the step 1e, excitatory `synapse` is potentiated by the value of `LTP` of the postsynaptic neuron at the time before `synapse.delay`. Finally, in the step 1f, all synapses from the `neuron` are added to the calendar to be processed at time after `synapse.delay` ticks.*

```
Propagation of Spikes:

  1. for each event synapse at time t

     (a) synapse.to.input += synapse.weight
     (b) if (synapse.from is excitatory)
          i. synapse.weightDerivate -= synapse.to.LTD
```

Pseudocode 3.3: Propagation of Spikes *part of the main algorithm. All synaptic events planned to this tick (`t`) are removed from the calendar and processed. The synaptic event `synapse` means that at tick `t` a spike from neruon `synapse.from` to neuron `synapse.to` arrived. Therefore the `weight` of this synapse is added to the `input` of the postsynaptic neuron. If the synapse is excitatory (thus changeable by STDP), in the step 1(b)i, the `synapse` is depressed by the current `LTD` value of the postsynaptic neuron.*

```
UPDATE OF STATES:

1. for each neuron

   (a) Update neuron state according the Izhikevich model

   (b) neuron.LTP[t+MAX_DELAY+1] = STDP_MULT *
       neuron.LTP[t+MAX_DELAY]

   (c) neuron.LTD *= STDP_MULT
```

Pseudocode 3.4: UPDATE OF STATES *part of the main algorithm. The neuron state update in the case of original form of Izhikevich neuron model (Izhikevich, 2003) is described by the Pseudocode 3.5, whereas in the case of generalized form (Izhikevich, 2007) is described by the Pseudocode 3.6.*

```
Update of neuron state in the original form:

1. neuron.v += 0.5*((0.04*neuron.v+5)*neuron.v+140-
   neuron.u+neuron.input)

2. neuron.v += 0.5*((0.04*neuron.v+5)*neuron.v+140-
   neuron.u+neuron.input)

3. neuron.u += neuron.a*(neuron.b*neuron.v-neuron.u)
```

Pseudocode 3.5: *Update of a neuron state according to the original form of Izhikevich neuron model (Izhikevich, 2003). The solution of a membrane potential part of the equation is computed in two steps, which leads to a higher stability and accuracy.*

```
Update of neuron state in the generalized form:

1. neuron.v += 0.5*((neuron.k*(neuron.v-neuron.vr)*
   (neuron.v-neuron.vt)-neuron.u+neuron.input)/neuron.Cap)

2. neuron.v += 0.5*((neuron.k*(neuron.v-neuron.vr)*
   (neuron.v-neuron.vt)-neuron.u+neuron.input)/neuron.Cap)

3. neuron.u += neuron.a*(neuron.b*(neuron.v-neuron.vr)-
   neuron.u)
```

Pseudocode 3.6: *Update of a neuron state according to the generalized form of Izhikevich neuron model (Izhikevich, 2007). The solution of a membrane potential part of the equation is computed in two steps, which leads to a higher stability and accuracy.*

```
End of Second:

  1. for each neuron and for j from 0 to MAX_DELAY+1

     (a) neuron.LTP[j] = LTP[TICKS+j]

  2. for each synapse

     (a) synapse.weight += WEIGHT_INCEREASE+snapse.weightDerivate

     (b) synapse.weightDerivate *= WEIGHT_DER_MULT

     (c) synapse.weight = min(MAX_WEIGHT, max(0,
         synapse.weight))
```

Pseudocode 3.7: End of Second *part of the main algorithm. All weights are updated and* `LTP` *is shifted, to be prepared for the next second.*

3.5.3 Alternative reality

3.5.3.1 Alternative reality: motivation and description

One of the advantages of models is the fact, that model also allows experiments which are impossible to run in the reality (in vivo or in vitro). Example of this situation could be running certain *in silico* experiments in an alternative reality. To explain this idea, it will beneficial to describe the motivation for such alternative reality first.

Measurement of some characteristics in the auditory cortex requires specific and possibly long experiments. Receptive fields could represent such an example. Receptive field may be defined as follows: "*The receptive field is a portion of sensory space that can elicit neuronal responses when stimulated. The sensory space can be defined in a single dimension (e.g. carbon chain length of an odorant), two dimensions (e.g. skin surface) or multiple dimensions (e.g. space, time and tuning properties of a visual receptive field). The neuronal response can be defined as firing rate (i.e. number of action potentials generated by a neuron) or include also subthreshold activity (i.e. depolarizations and hyperpolarizations in membrane potential that do not generate action potentials).*" (Alonso and Chen, 2009)

In the case of neurons in the auditory cortex, the space of inputs is often characterized by input amplitude (volume) and frequency, the main characteristics of the sound. To measure the receptive field of a neuron, a battery of pure tones of different frequencies and amplitudes is typically selected and played to the observed individual repeatedly.

Receptive field is a type of feature that may evolve during both shorter and longer periods. It can be influenced by the sounds heard during the initial phase of life. Therefore, if we want to measure the development of a receptive field over time, when the individual listens to a very specific type of sounds (e.g., total silence, some type of noise, pure tones, etc.), the experiment that measures the receptive fields may significantly influence the measured feature.

However, in experiments *in silico*, such kind of experiment could be possible. We would only need to have the possibility to suspend the "long experiment" (in which the observed individual listens to the specific type of sounds) and run a "short experiment" (in which the receptive fields are measured) in the alternative reality and then again continue with the long experiment. This sequence of interlaying in the normal and the alternative reality may be done repeatedly, see Figure 3.4.

Such a possibility of the alternative reality could be useful in any kind of model (not only auditory), where one needs to observe development of some complex feature during a longer experiment. Before description of possible implementations of such alternative reality, we should first describe the requirements for this function:

1. Have a possibility to run an experiment B inside an experiment A (at time t), which will not be influenced by the experiment B. This means that experiment A would come out identically with and without interruption by the experiment B.

2. For a simplification, the experiment B could be run without changes of long-term plasticity. This simplification is quite reasonable, because the experiment B is typically short.

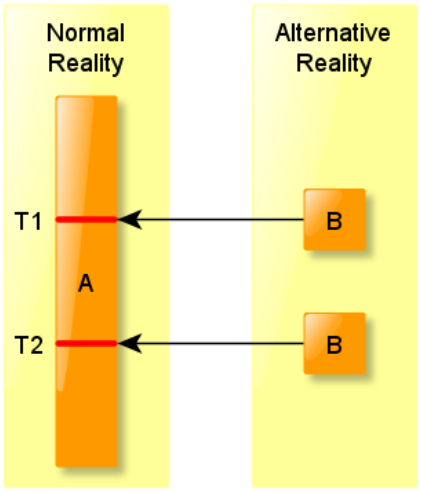


Figure 3.4: *A diagram of the alternative reality. Here, the experiment A should be interrupted at times T1 and T2 and in both these times, the experiment B should be run. However, the experiment B should not influence the experiment A, which we call running in an alternative reality.*

3.5.3.2 Alternative reality: possible solutions

There are three main possible solutions how to implement the function of the alternative reality:

1. The first solution stops an experiment A at time t, copies all relevant data structures to a local copies, runs the experiment B on these copies and

after the end of the experiment B continues with the experiment A on the original data structures. The relevant data structures are calendar and changing parts of the network (attributes `u, v, LTD` and `LTP` of neurons and attributes `weight` and `weightDerivate` of synapses).

2. The second solution also stops an experiment A at time t, but continues with the experiment B on the same data structures with frozen weights (and turned off STDP), and then again continues with the experiment A with again turned on STDP and unfrozen weights, still on the same data structures.

3. The last solution instead of stopping the experiment A only saves the changing data structures into a file. After the end of the experiment A, the changing values are loaded and experiment B is run.

3.5.3.3 Alternative reality: discussion of possible solutions

1. The first solution is accurate to the intent that the experiment B does not change the experiment A. On the other hand, it has quite large both memory and time demands, since it copies large data structures. It is important to realise that the changing values of the network and the calendar comprise the majority of the memory demands of the simulator. These demands are not caused only by the elements inside these structures, but their quantity, in the case of large-scale networks.

2. The second solution does not bring any extra memory demands. However, the result is only approximate and does not exactly correspond to the marked out requirements.

3. The third solution is accurate, simple and the only extra memory demands concern the external memory (on a hard-disk), which is typically not a limited resource. It has also other positive advantages. If the data structures are saved from more times of the experiment A and in all these positions the experiment B should be run, these experiments may be run parallelly, which can accelerate the whole process. In addition to that, the saved data structures may be used also as a log for other retro-analysis. The only drawback of this solution may be the deceleration of the experiment A by saving the data structures to HDD (hard disk drive).

3.5.3.4 Alternative reality: outcome

Since the first solution has an essential drawback in memory consumption and the third solution is not accurate, we chose the third solution. We tested the time needed to save the data structures of a typical network (up to 10^5 neurons). The time was in the order of few seconds (1 s for a network with 10^4 of neurons, 14s for a network with 10^5 of neurons). If only few times in the simulation the alternative reality is needed, this deceleration is not a problem.[2]

[2]Furthermore, the time consumption could be largely decreased using a SSD (solid state drive) hard disk.

3.5.4 Computational improvements

Although the Izhikevich neuron model and the chosen implementation of STDP are computationally advantageous, the speed of the simulation for large-scale network is still an important issue. After the whole simulator was implemented in the basic form, we designed and implemented several improvements of both time and memory aspect. We list them in the following subsections in the order from the smaller improvements to the bigger ones.

3.5.4.1 Limiting frequent use of methods and constructors

Java code style often contains techniques, which are nice, transparent and safe, such as encapsulation, separating everything into object, etc. However, use and especially overuse of these techniques may have negative impact on the performance. Frequent calls of constructors and methods should be used carefully and tested in a profiler. After a conscionable profiling, we decided to substitute some private attributes encapsulated in getters and setters with public attributes. At the same time, we made an effort to avoid creating huge amount of new objects. On the other hand, we tried to keep the whole code well-arranged and transparent.

3.5.4.2 Effective data structures

During the initial stage when high-level algorithms and usage of data structures were designed, we followed the classical approach to basic data structures (linked list, array, hash table, etc.) choice. Java offers built-in collections, such as ArrayList, LinkedList, or HashMap, which are easy to use. However, it is known that own implementations based on basic arrays, may be more effective if one is willing to give up some functionality. To test the efficiency for our usage, we compared their performance in operations used in the algorithm. The main result of this improvement is a usage of a dynamic array (see Figure 3.5) instead of the linked list in the circular array of the calendar. This improvement helped not only in the terms of time consumption, but also a memory efficiency.

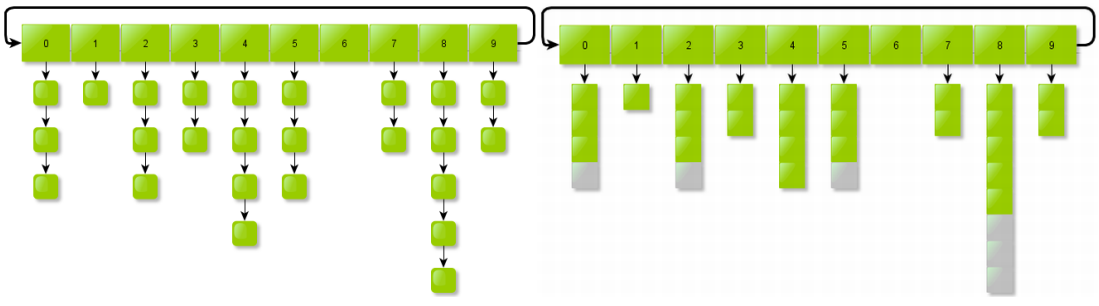


Figure 3.5: *The two tested data structures of the calendar. Left: the circular array of linked lists. Right: the circular array of dynamic arrays. The* ***green*** *array indexed are the used ones, whereas the* ***gray*** *are the unused ones. Every time, when the dynamic array is too small, it is two times enlarged.*

3.5.4.3 Parallel processing: description

The last and most significant group of improvements is based on parallel processing. It is obvious that the best tasks for parallelization are such tasks that can be divided into parts solvable separately in different threads without need of work with shared data. In the case of risk that more threads would change the same data via a non-atomic operation, then additional mechanisms must be used to prevent inconsistency. The possible mechanisms in Java are for instance locks and synchronization (for more information about concurrency in Java see e.g., Peierls et al., 2005).

3.5.4.4 Parallel processing: preliminary notes and trivial solutions

In our case, there is not any trivial division into parts. If the network consisted of separated parts of neurons with limited "communication" between these parts, the trivial division would be to compute simulation of these parts in separate threads (one thread for each network part). By the "communication" we mean transmission of the action potential. However, the network does not have to contain any separated parts in general, so this trivial solution is out of the question.

Another trivial solution of a certain sort of parallelism is to run more separate experiments in parallel. This solution does not need any ingenious algorithms, just the possibility to run more experiments in one Java process, or even in separate processes. It could be done always, when memory resources are sufficient for all these experiments.

To analyse the possibilities of non-trivial solution, we need to recapitulate the basic structure of the simulation algorithm and realize, which section could be run in parallel threads. There are five main sections (steps) of the algorithm:

1. Generation of Inputs
2. Detection of Spikes
3. Propagation of Spikes
4. Update of States
5. End of Second

The last section is run only once a second and consumes much less time than the other sections. The Generation of Inputs in the basic algorithm is also simple and is provided by the input module. For these reasons, we will discuss only the sections Detection of Spikes, Propagation of Spikes and Update of States. We will start from the Update of States section, which is easiest to parallelize, and then continue with sections Detection of Spikes and Propagation of Spikes. The used solution for these two sections will be described together in the end of the chapter due to their common work with the calendar. In all the following sections, let T be the number of available threads.

3.5.4.5 Parallel processing: Update of States section

The Update of States (see Pseudocode 3.4) section can be computed in parallel quite easily. Since the natural solution of making Update of States parallel is easy and without drawbacks, we do not describe any other solutions and move on directly to the used solution.

The neurons are divided into T groups and each thread updates states of neurons from one group. Useful implementation of this idea is to have the group of neurons defined by boundaries of the half-bounded interval ⟨start, end). This means that this group contains neurons with numbers start, start+1, ..., end−1. Then the thread needs to know only the value of start and end variables and have an access to the network. This implementation is thread-safe (threads work with another parts of the network) and easy to implement.

3.5.4.6 Parallel processing: Detection of Spikes section

Let us recall the section Detection of Spikes (see Pseudocode 3.2). There are three types of changes done during this section:

1. Changes of attributes of the firing neuron.
2. Changes of attributes of the synapse going to the firing neuron.
3. Adding an event to the calendar.

The natural approach to division work among threads would be again to let each thread take care of separate group of neurons and for each of them test the firing condition and possibly do the rest of changes. The two types of changes would be thread-safe, because they are related only to the neurons from one group. However, the action of adding an event to the calendar may present a risk, since it is not an atomic operation.

Possible solutions. We designed three possible solutions of the problem with thread-unsafe operation of adding an event to the calendar:

1. In the first solution, the operation of adding an event into the calendar is synchronized. (There is a simple way how to do it in Java: add a keyword synchronized to the relevant method.)
2. In the second solution, each thread adds all events into its own local calendar and at the end of the tick, all calendars are merged together.
3. In the third solution, during the whole simulations, T calendars are maintained. Each thread adds event only to "its" calendar and these data are never merged together.

Discussion of the possible solutions. The proposed solutions have the following advantages and disadvantages:

1. The first solution is easy to implement. On the other hand, the synchronization may be computationally expensive and may lead to more serial computing than parallel computing, because adding to the calendar is one of the most frequent operations in the Detection of Spikes section.

2. The second solution would need some effective implementation of merging the calendars. It is a question whether merging calendars instead the first solution would be more effective. The probably best way how to find out the answer would be to implement both solutions and measure them in the profiler.

3. The third solution is both easy to implement and thread-safe. Because it does not need any synchronization, it is effective. The only possible drawback of this solution is the fact that events in the calendars may be distributed unevenly. This means that one calendar could contain only few events, whereas other calendars would contain plenty of them.

3.5.4.7 Parallel processing: Propagation of Spikes section

The section Propagation of Spikes (see Pseudocode 3.3) is the shortest one. Unlike the two previous sections, this is iterated over the events from the calendar (instead of neurons). Therefore the natural distribution among threads is to divide these events into groups and let each thread go through one group of events. There are two matters that must be solved:

1. How to divide the events into the groups.

2. How to make the changes of neuron attribute (`input`) and synapse attribute (`weightDerivate`) thread-safe.

Possible solutions. There are two main possible approaches how to solve these matters:

1. In the first approach, the events are distributed into groups in such way, that events-synapses with the same postsynaptic neurons will be in the same group. Then, each thread will change attributes of different neurons and synapses, and the whole code will be thread-safe.

2. In the second approach, the events are distributed into groups in any suitable way and both operations are made atomic (e.g., using a lock).

These approaches do not specify the exact method, how the events are distributed into groups. It depends on the structure of the calendar. The basic structure described is a circular array of linked list. However, also other structures may be used – for instance the structure of T separate calendars, suggested in the Section 3.5.4.6, where possible solution of Detection of Spikes parallelism were described.

There are at least four possible solutions, how to divide the events into groups:

1. The first solution works with the basic data structure for the calendar (the circular array of linked lists). Here, the linked list is iterated over and divided into T (nearly) evenly large groups. Since in this solution, we go through the whole list, it is possible to divide the synaptic events according the postsynaptic neuron or any other rules.

2. The second solution works with slightly adjusted basic calendar. When the calendar is created (events are added), there are kept T pointers to different parts of the list. Subsequently, when we need to divide one list into T groups, we just cut the sub-lists defined by the pointers. There is a problem that during creating calendar we do not know how many events will be added and hence how distant the pointers should be. It would not be probably so difficult to design an effective algorithm or heuristic for this solution. However, because other solutions have better qualities, it is not necessary to describe this one in greater details.

3. The third solution works with the calendar in the form of circular array of arrays. These array may be in the form of the ArrayList, a built-in collection in Java, or the array of fixed maximum length (if we know the maximum number of events for one tick), or the dynamic array. Dividing an array into T parts is easy and can be done in the same way as it was used in Update of States and Detection of Spikes, where we divided an array of neurons.

4. The fourth and last solution works with the set of T calendars in cooperation with the last solution of the section Detection of Spikes. The usage is again very simple: each thread works directly with events from one "its" calendar (e.g., the calendar with the same serial number as the thread's serial number).

Discussion of the possible solutions. The proposed solutions have the following advantages and disadvantages:

1. The first solution is relatively simple and flexible, because it allows dividing the events into the groups according any rule. Therefore it can be used in combination with the first approach how to make the Propagation of Spikes thread-safe. This means that the synaptic events can be divided into groups with disjoint postsynaptic neurons. However, this solution is relatively ineffective, since it requires going through the whole list of events before the parallel section starts. This nearly two times more work (each event will be processed twice instead of once) where nearly half of this work would not be parallelized. Therefore it is even a question whether this solution would bring any speedup.

2. The second solution could be faster than the first one, but the final performance would depend of the implementation of pointers maintenance. However, it does not allow any added flexibility (as the first solution) and is more complicated than all the other solutions.

3. The third solution is both fast and easy to implement.

4. The fourth solution is also fast and easy to implement.

3.5.4.8 Parallel processing: outcome

For both the Detection of Spikes and Propagation of Spikes we chose the last solution – the solution which uses T separate calendars. The resulting flow of processing is depicted in the Figure 3.6.

First, neurons are separated into T groups and all groups are processed concurrently, each in a separate "Detection Thread". Each thread uses its own calendar, to which it adds the synapses from the spiking neuron. It is good to realise, that synapses leading from one neuron will be added to one calendar.

Subsequently, after all "Detection Threads" are finished, the computation is again divided into T threads: the "Propagation Threads". Each thread processes neurons from one calendar. Thanks to the fact, that one presynaptic neuron can occur only in synaptic events of one calendar, there is no need to wrap the step

```
synapse.weightDerivate -= synapse.to.LTD
```

with a lock, because there is no risk that two different threads would change the same synapse. However, the step with update of the neuron's input must be wrapped with a lock.

When all "Propagation Threads" finish, the neurons are divided into T groups and each group is updated by one "Update Threads".

The extent of the performance upgrade as well as comparison of performance demands for each section are described and discussed in the Chapter with results 4.

3.5.5 Spontaneous activity

It is a well known fact that neocortical networks in the brain produce an activity in the absence of any sensory stimuli (Arieli et al., 1995). One of the mechanisms underlying this spontaneous activity are the miniature Postsynaptic Currents (mPSCs)[3] called "minis" (Timofeev et al., 2000), a spontaneous release of neurotransmitter, independently on the spikes. In some models, the spontaneous activity is modelled just as a nonspecific and typically stochastic background input (e.g., in Hansel and Mato, 2001). However, several models model exactly the process of mPSCs (Muresan and Savin, 2007; Izhikevich and Edelman, 2008; Phoka et al., 2012, e.g.,). Since this approach corresponds more accurately to the reality, we chose it for introducing a spontaneous activity in our model of the AC, and therefore to implement it in the SUSNOIMAC simulator.

First, we will describe the main features of the basic mPSCs mechanism used in computational models. Each synapse has a probability P that an mPSC will occur on this synapse at this time with amplitude A. The probability can be expressed also in the form of frequency F with the relation P=F/fs, where fs is number of ticks in one second. Subsequently F in expressed in Hz and A in for instance in pA. In general, the parameters F and A may be fixed and global for the whole network and the whole simulation (which is probably a simplification), or they can be specified according the synapse type and eventually they can also

[3] Or miniature Postsynaptic Potentials (mPSPs).

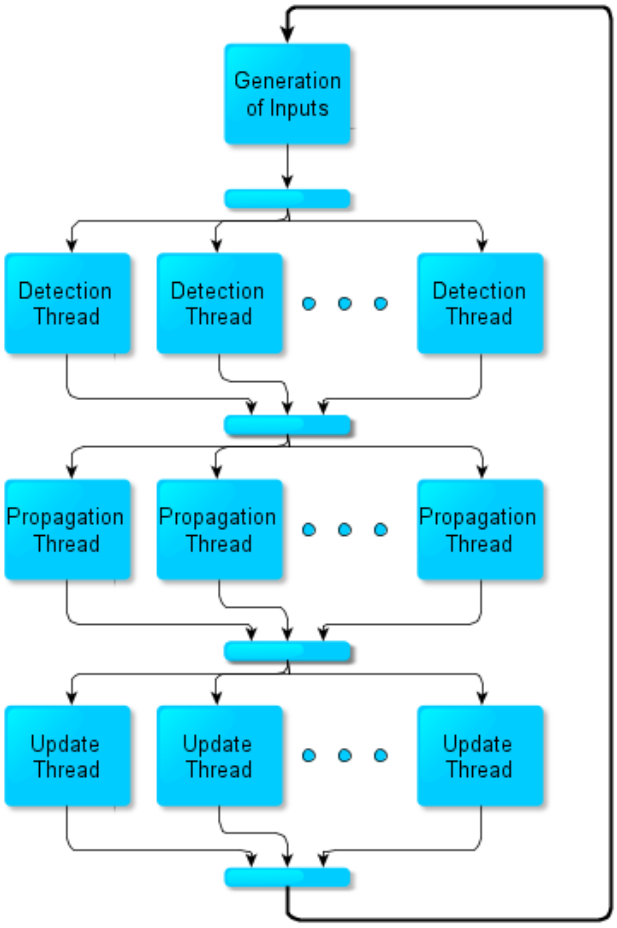


Figure 3.6: *A diagram of a parallel computing of the main simulation algorithm.*

vary somehow evolve the simulation. However, in this simulator just the version of fixed and global parameters was implemented. The main reason is that we would not use any more general version, because we did not have data to which the values of these parameters (dependent on synapse type and possibly time) would be fitted. The extension for more general version is a possible future work.

Nevertheless, at first, we will describe and discuss possible solutions of mPSCs implementation.

3.5.5.1 Implementation of spontaneous activity: possible solutions

In this section we use the phrase "throw a die against a number X". By this phrase, we mean generate a number from $\langle 0, 1\rangle$ from uniform distribution and test, whether the generated number is lower or equal than X. If yes, we say that throw was successful. Otherwise the throw was unsuccessful.

We considered three possible implementations of spontaneous activity in the form of mPSCs:

1. The first and most direct implementation goes through all synapses and for each throws a die against P and in the case of success, it emits a mPSC with amplitude A at this synapse. This means that the value A is added to the

input of postsynaptic neuron of the synapse, if the synapse is excitatory, and value −A if the synapse is excitatory.

2. The second implementation generates M synapses, on which a mPSC will occur in this tick. M is defined as P·S, where S is the number of synapses in the network. We should note, that besides the synapses numbers generation it is also necessary to go through these generated synapses and on each emit a mPSC with the given amplitude A.

3. The third implementation goes through the neurons and for each neuron generates two numbers from the binomial distribution: N1 and N2. The first number, N1, is from Bi(nE, P), where Bi means binomial distribution and nE means number of excitatory synapses that lead to this neuron. The second number, N2, is from Bi(nI, P), where Bi again means binomial distribution and nI means number of inhibitory synapses that lead to this neuron. Subsequently to the input attribute of this neuron, the value N1·A + N2·(−A) is added.

3.5.5.2 Implementation of spontaneous activity: discussion of possible solutions

First, we should mention that these three implementations do not do the same. The number of synapses with mPSC in the second implementation is always P·S, whereas in the first and third implementation the value P·S is only a mean value of the number of synapses with mPSC. Since we do not have any preferences between these behaviours, we do not assess them as positive or negative. However, it is good to be aware of this difference.

There are other two main factors which could be considered in the comparison of the suggested implementations. The first factor is the performance and possibility of parallelization and the second is a possibility of extension for more general version of parameters P (resp. F) and A: parameters specific for the synapse type or changing over time. We will call this extension as synapse-specific parameters P and A, and changing parameters P and A.

1. The first implementation could be trivially combined with both synapse-specific and changing parameters P and A. On the other hand, this implementation is very time consuming. It iterates over all synapses in every tick. It could be parallelized but, it would significantly decelerate the whole simulation nevertheless.

2. The second implementation could also be combined with the extended versions of parameters P and A. The performance is better than in the previous case, because we do not go through all synapses, but only through P*S synapses. It could be also parallelized.

3. The third implementation is the most effective one, because the number of neurons is typically (significantly) smaller than both S and P·S. The final performance depends also on the performance of generating the numbers from the binomial distribution. (For instance the performance of trivial implementation of generating a number from Bi(N,P), where for each of N

numbers we throw a dice against P, would be as slow as the first solution.) Besides the natural efficiency of this solution, it could be also easily parallelized. We should also mention that this third implementation needs to know the number nE and nI for each neuron. However, these numbers can be stored in two simple arrays indexed by neurons. To end the evaluation of this solution, it is not well suited for the extended versions of parameters P and A. The implementation could be extended for the synapse-specific parameters (e.g., by storing the numbers of the synapses of each synapse type leading to the neuron). However, the extension for changing parameters would be either complicated, or inefficient.

3.5.5.3 Implementation of spontaneous activity: outcome

With regards to the main goal of the simulator – the model of Auditory Cortex, the factor of performance was much more important than the factor of introducing the extended versions of parameters P and A. Therefore we chose the third implementation in the parallel version. The resulting flow of the whole simulation is depicted in the Figure 3.7. First, the basic inputs are generated by the input module. Then the mPSCs are generated and added to the inputs. This phase is performed in concurrency in parallel threads called "Input Threads". The rest of the flow is same as in the previous version.

3.5.5.4 Implementation of spontaneous activity: choice of pseudorandom generator

When using a pseudorandom number generator (PRNG) it is very important to use a good one, which generate numbers with high degree of randomness, and which is fast enough. In the past certain research results turned out unreliable, because they were produced by low-quality PRNGs, such as the RANDU generator from IBM mainframes (Schutter, 2010, chap. 2). Also, the built-in PRNGs of the C and C++ languages are often badly implemented and should be avoided (Press et al., 2007).

Therefore we made an effort to find a PRNG which would be both of high-quality and fast enough. After research of possible PRNGs, we chose the Mersenne Twister method (Matsumoto and Nishimura, 1998) as one of the best PRNGs suitable for most applications except cryptography and recommended also for the computational neuroscience (Schutter, 2010, chap. 2). We used the implementation of this method provided by the Colt Parallel Java library described in the Section 3.7.

3.5.6 Setting simulation parameters

Instead of using constants fixed in the code, we decided to set the simulation parameters via a settings file. Actually, it is a rather obviously better approach, but we will mention the main three advantages of this approach explicitly. First, the user does not have to change code to run an experiment. Second, the saved settings for the experiment could be used also during analysis of the results, or as a log. Third, this approach can be easily used for batch processing of more experiments in a row.

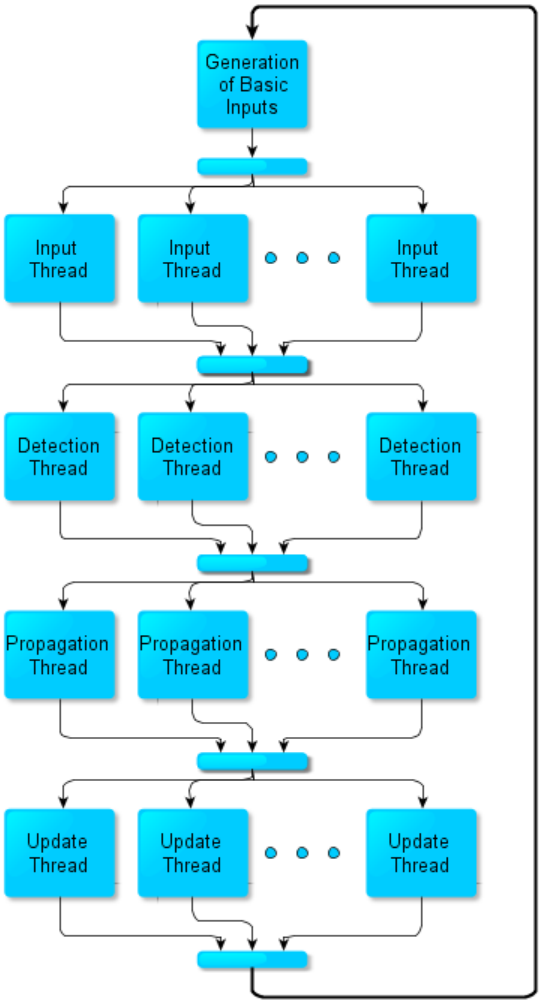


Figure 3.7: *A diagram of the resulting parallel computing of the main simulation algorithm with parallelization of* Generation of Inputs*, where also mPSCs are generated.*

We decided to use a transparent format of Java.propeties, see 3.8 for an example. The main simulation parameters are listed in the Tables 3.7, 3.8, 3.9, and 3.10. Other minor parameters (such as those related to logs, loading and saving) are described in the User Documentation 1.

Name	Unit	Description	Used values
`TICKS_SIM`	ms	Duration of the simulation	up to 7200000
`LOG_START`	s	Frequency of time frames of logs	various values
`LOG_DURATION`	s	Duration of time frames of logs	various values
`CHV`	s	Frequency of alternative reality	various values

Table 3.7: *The simulation parameters related to* ***time****.*

The simulation parameters are set in the file `simulationX.properties`, where `X` means the number of the experiment (starting with value 1). The exact names used in this file are slightly different than those in the listed tables (some of them are shortened here), but their names can be easily found both in the example files and User Documentation 1.

```
#-----------TIME-----------
#Duration of the simulation
TICKS_SIMULATION = 3000
#Frequency of time frames of logs. Each
<k*LOG_START_IN_S,k*LOG_START_IN_S+LOG_DURATION_IN_S) secons the logs (spike trains,
weights, and potentially potentials).
LOG_START_IN_S = 1800
#Duration of time frames of logs. Each
<k*LOG_START_IN_S,k*LOG_START_IN_S+LOG_DURATION_IN_S) seconds the logs (spike trains,
weights, and potentially potentials).
LOG_DURATION_IN_S = 5
#Frequency of alternative reality. If SAVE_CHANGING_VALUES is true, then each
CHANING_VALUES_INTERVAL_IN_SECONDS seconds, the changing variables of neurons and synapses
are saved.
CHANING_VALUES_INTERVAL_IN_SECONDS = 600
#-----------GENERAL-----------
#True for the original Izhikevich mode, false for the generalized one. The original form
of the Izhikevich neuron model uses only 4 parameters a, b, c, and d. The generalized form
uses all 9 parameters.
OLD_EQ = false
#true ==> True for continuing with simulation after initialization.
SIMULATE = true
#True for continuing with visualization (in Java) after simulation. The visualization
shows development of the network from the time frame <M_MIN_T,M_MAX_T).
VISUALIZE = true
#The tick, when the monitorind of values for retrospective visualization (in Java) starts
M_MIN_T = 0
#The tick, when the monitoring of values for retrospective visualization (in Java) ends
M_MAX_T = 5000
```

Figure 3.8: *Example of a part of the properties file, which is used to the setting the parameters of the simulation.*

Name	Unit	Description	Used values
OLD_EQ	boolean	True for the original Izhikevich mode, false for the generalized one	both
SIMULATE	boolean	True for continuing with simulation after initialization	both
VISUALIZE	boolean	True for continuing with visualization (in Java) after simulation	both
M_MIN_T	ms	The tick, when the monitoring of values for retrospective visualization (in Java) starts	various
M_MAX_T	ms	The tick, when the monitoring of values for retrospective visualization (in Java) ends	various

Table 3.8: *The **general** simulation parameters.*

Name	Unit	Description	Used values
M_FREQUENCY	Hz	The frequency of mPSC on each synapse (called F in the text)	15–60
M_AMPLITUDE	pA	The amplitue of mPSC on each synapse (called A in the text)	13–27
STDP	boolean	True for STDP applied during the simulation	both
PARAM_LTD	-	Reset value of LTD after spike (the A_- part of STDP)	1.2, 2
PARAM_LTP	-	Reset value of LTP after spike (the A_+ part of STDP)	1
W_INCREASE	-	Activity independent increase of synaptic weight (this value is added to each excitatory synapse once a second)	0.1
W_DER_MULT	-	Each excitatory weight derivative is multiplied by this value once a second	0.9
STDP_MULT	-	Values of LTP and LTD are multiplied by this value each tick – e.g., value 95 can be used for the rate $e^{(-t/20)} = 0.951229424$	0.95

Table 3.9: *The simulation parameters related to parameters of the used* ***synapse dynamics****.*

Name	Unit	Description	Used values
THREAD_INP	boolean	True for parallelization of the phase Generation of inputs	both
THREAD_DET	boolean	True for parallelization of the phase Detection of spikes	both
THREAD_PROP	boolean	True for parallelization of the phase Propagation of spikes	both
THREAD_UPD	boolean	True for parallelization of the phase Update of states	both

Table 3.10: *The simulation parameters related to* ***parallelization****.*

3.6 Analysis module

3.6.1 Possible choices of implementation of the analysis module

The main part of the analysis of the simulation results is related to analysis of spike trains (list of spikes: who fired and when), or development of other values (e.g., membrane potential, or weights). There are (at least) three possible solutions, how to implement such analysis:

1. The first solution analyses and visualises the result during the simulation.
2. The second solution monitores the relevant values and saves them into inner data structures and after end of the simulation, the analysis and visualization starts.
3. The third solution also monitores the relevant values, but saves them into log files on HDD and after end of the simulation, the monitored values are loaded, analysed and visualised.

3.6.1.1 Discussion of possible choices of implementation of the analysis module

1. The first solution is advantageous for two reasons. First, such concurrent visualization could be beneficial for debugging. However, the classical debugging such as watching values of singular variables is in the case of large scale networks very difficult. Other approaches to debugging are in this case more suitable (such as colouring the neurons, or observing average values), and these approaches typically do not need the concurrency of simulation and visualization. Second, the concurrent visualization could be beneficial in combination with user interventions: e.g., changing the inputs, or parameters. On the other hand, this solution of analysis module has several drawbacks. Sometimes, the simulation is faster than analysis, and analysis (and its visualization) would retard the simulation. Other times, the simulation is slower than analysis, and observation of such analysis would be slow and inconvenient.
2. The second solution is ideal for small or medium-size networks and rather short experiments. Simulation is not retarded (e.g., by analysis, or saving to log files) and analysis can be run automatically immediately after simulation. This makes the debugging cycle fast and comfortable. However, the large networks and long experiments may lead to depleting the possible memory. Then, the third solution must be used.
3. The third solution is actually a necessity in the case of large networks and long experiments. Moreover, it brings several advantages. The analysis may be done separately (possibly also in parallel) and whenever after the simulation. The log files may be used also for other uses. For instance they can be loaded by other programs, e.g., specialized for the spike trains analysis.

3.6.1.2 Outcome of possible choices of implementation of the analysis module

In this thesis, we chose the combination of second and third solution: for smaller networks and shorter experiments a possibility of visualization of the coarse of the simulation in Java, and for all experiments possibility of a subsequent more complex analysis in Matlab. This combines all important features of the analysis module: fast and comfortable debugging for smaller experiments and sufficiently rich analysis for all (also big) experiments. For the subsequent analysis (loaded from log files) we chose tha Matlab language for the following reasons:

1. Matlab is specialized for processing the numerical data: it is fast in this processing (without necessity of additional optimalizations) and contains many built-in functions, useful for the analysis (e.g., statistical functions, FFT, visualization of graphs). In contrast, own implementation of these functions in Java, or connecting third-site libraries, would be much more time-consuming.

2. Matlab is frequently used in the community of neuroscientists. Therefore, implementing the analysis module in Matlab is advantageous also for those users, who would like to change or extend the analysis, but are not familiar with Java.

First, we list the analysis in Java, and second, the analysis in Matlab.

3.6.1.3 Functions of the analysis module: Java part

It is possible to specify an interval $\langle$ `M_MIN_T`, `M_MAX_T` $\rangle$, when the following neuron the dynamic variables (i.e, `v`, `u`, `LTD`) and `inputs` of neurons are monitored. After the simulation, if switch parameter `VISUALIZE` is on, the visualization window is opened. It shows two views:

1. The 3D view on the network, which is possible to zoom in and out and rotate. In this view either the whole network, or one layer can be displayed (see Figure 3.10).

2. The 2D view on a single layer (see Figure 3.9).

In addition to that the visualization window contains the time line, which allows to display (in both 2D and 3D view) development of the network. The neurons are coloured according to a value of the chosen dynamic variable.

Finally, the development of all dynamic variables of a singular may be displayed in the given interval (see Figure 3.11).

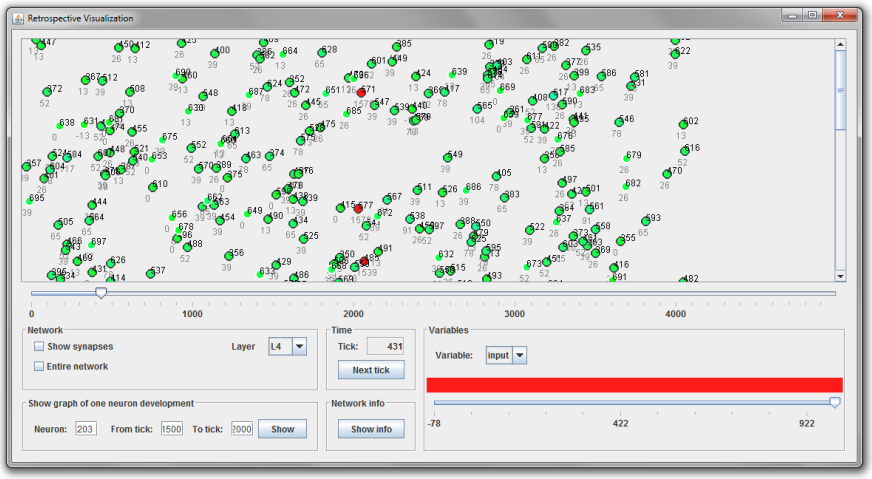


Figure 3.9: *An example of 2D view of an input layer and neurons coloured according to the input value. Here, three neurons in one band are stimulated.*

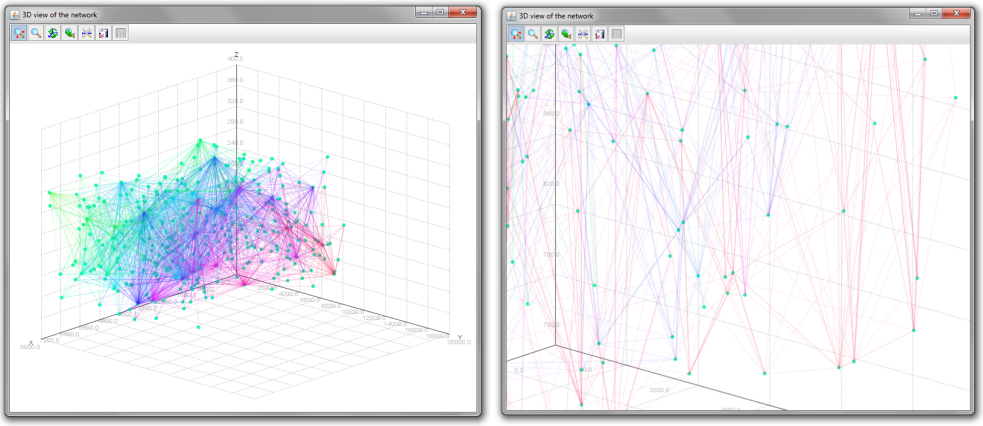


Figure 3.10: *The 3D view of the network (here only a single layer). It is possible to zoom, move and rotate the view.*

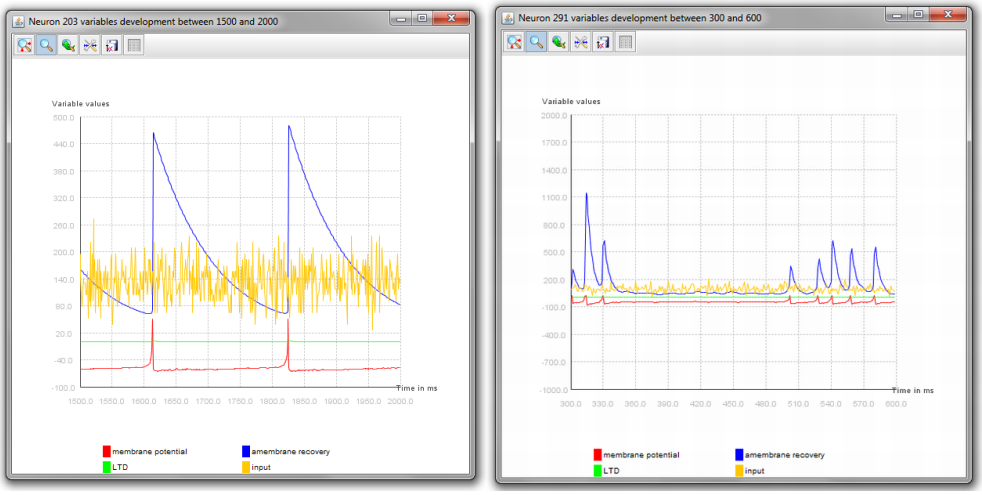


Figure 3.11: *The development of variables of a certain neuron (left excitatory, right inhibitory). It is possible to depict such plot for any neuron in the network and any given time interval of the simulation.*

3.6.1.4 Functions of the analysis module: Matlab part

The analysis in Matlab is divided into three scripts: analysis of the main features, development of the weight distribution, and analysis of the features related to tonotopy (this is a part specific for the auditory model).

Script A The script A computes and displays:

1. The spike time raster plot, coloured according to the neuron types: Figure 3.12.
2. The development of the mean firing rates, according to the neuron types: Figure 3.13.
3. The box-plot of the mean firing rates, according to the neuron types: Figure 3.14.
4. The development of mean excitatory weights (only excitatory, because inhibitory are static): Figure 3.15.
5. The 3D view on the network, neurons are coloured according to the neuron types: Figure 3.16.
6. The global oscillations (waves) from spike trains of the whole network: Figure 3.17.
7. The local oscillations (waves) from smaller areas of the network (these areas can be specified in the parametrization of the script): Figure 3.18.

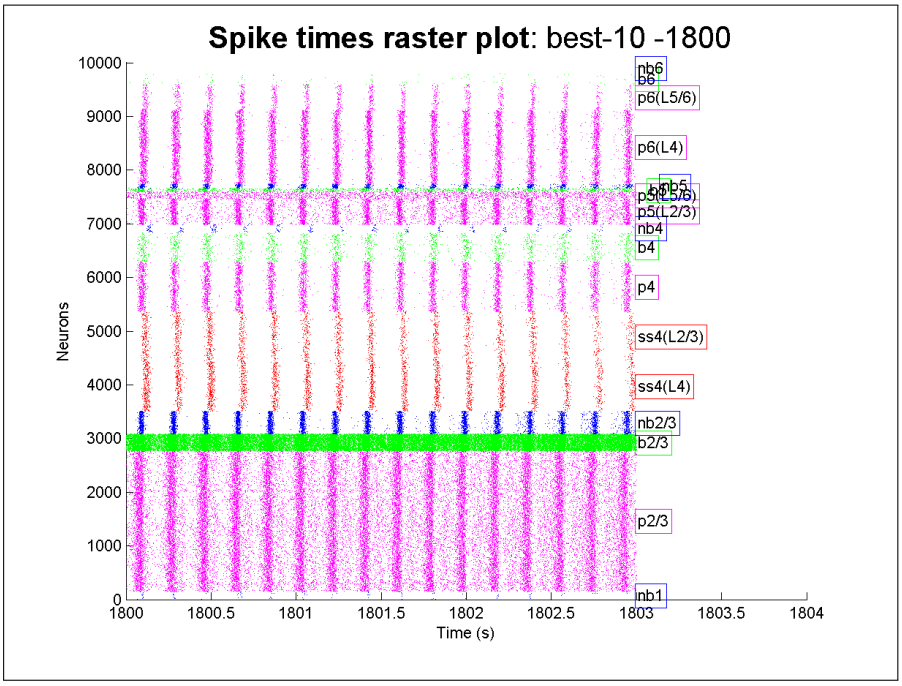


Figure 3.12: *The spike time raster plot draw spike trains in the raster plot, where the axis x corresponds to time and the axis y to the neuron numbers. The neurons are sorted by their location in the network: neurons from L1 at the bottom and neurons from L6 are at the top. The different neuron types are distinguished by the colour:* ***pyramidal*** *(p),* ***spiny stellate*** *(ss),* ***basket*** *(b), and* ***non-basket*** *(nb).*

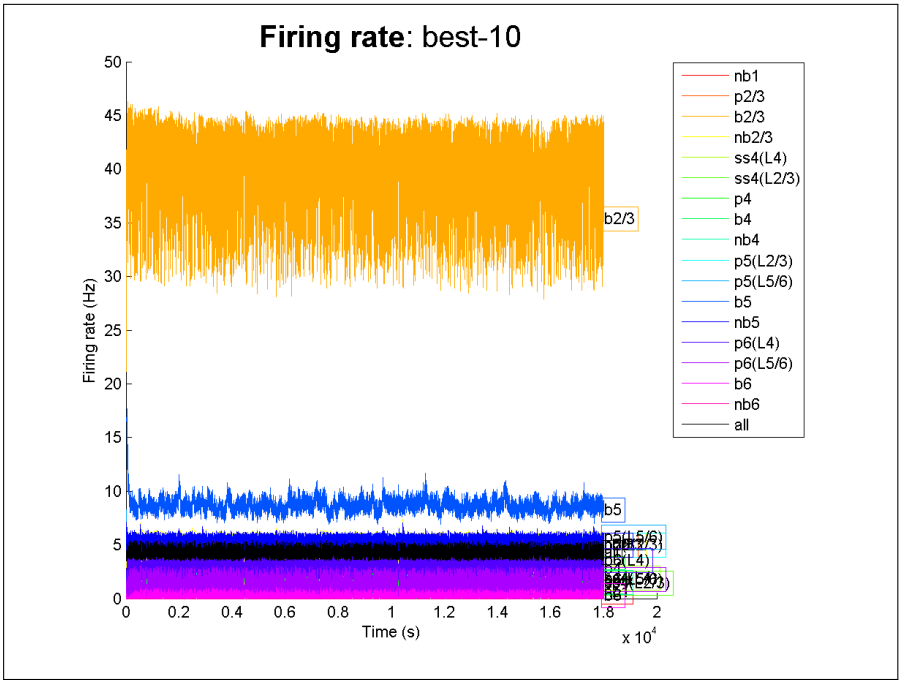


Figure 3.13: *Firing rate plot depicts mean firing rate of each neuron type over time. The axis x corresponds to time (in s) and the axis y corresponds to firing rate (in Hz). The different neuron types are distinguished by the colour (see the legend). Black colour represents the mean firing rate of the entire network. Here, the most active neuron types are* ***b2/3*** *(basket neurons from L2/3) and* ***b5*** *(basket neurons from L5).*

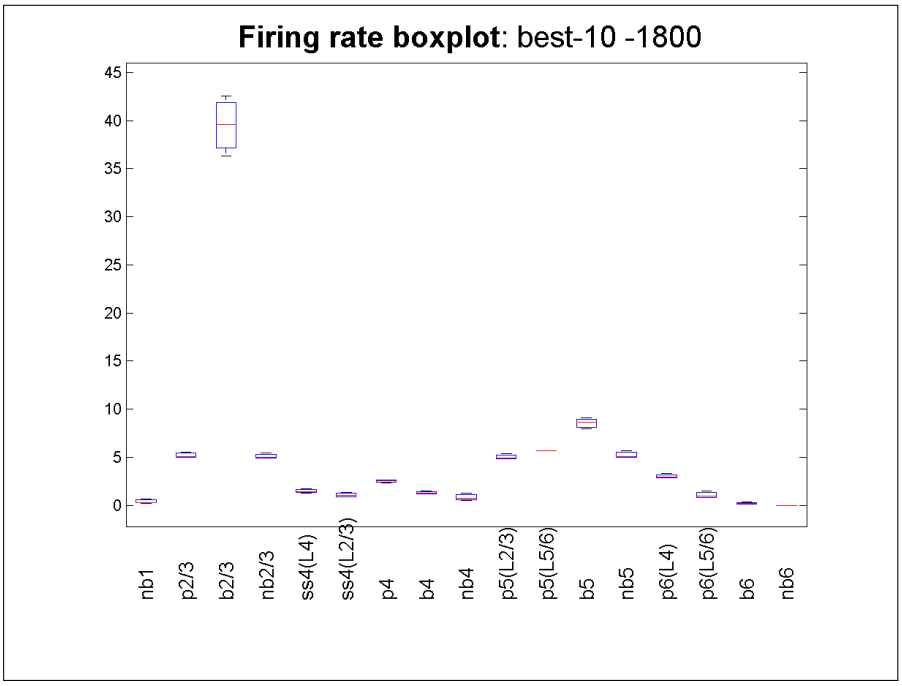


Figure 3.14: *This boxplot is an additional statistic to the mean firing rate of neuron types. The axis x corresponds to the neuron types and the axis y corresponds to firing rate (Hz).*

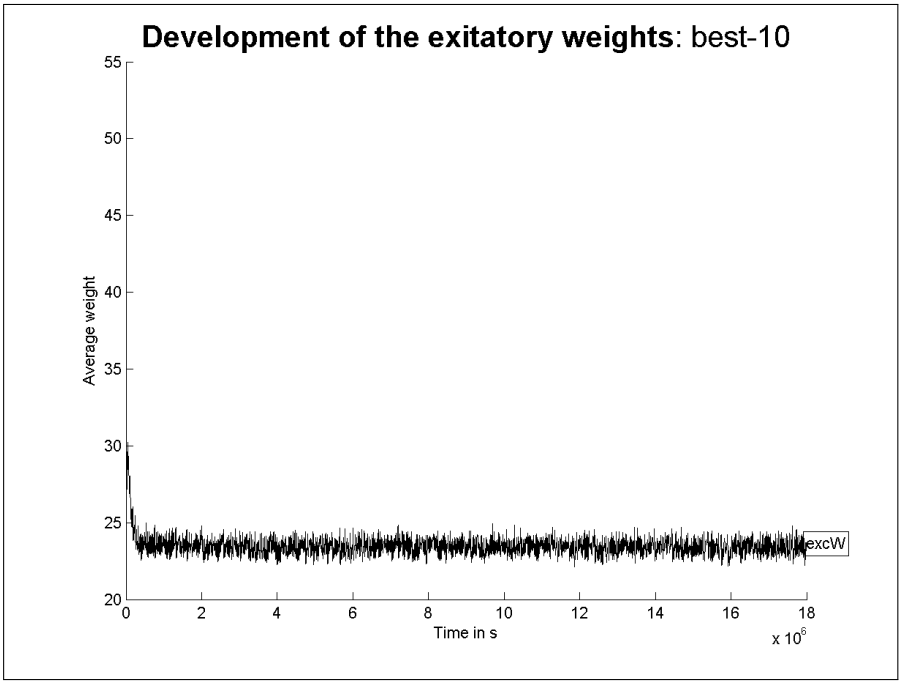


Figure 3.15: *This plot depicts the development of the mean values of the excitatory weights over time. The axis x corresponds to time and the axis y corresponds to the weight value. The inhibitory weights do not change over time and therefore they are not included.*

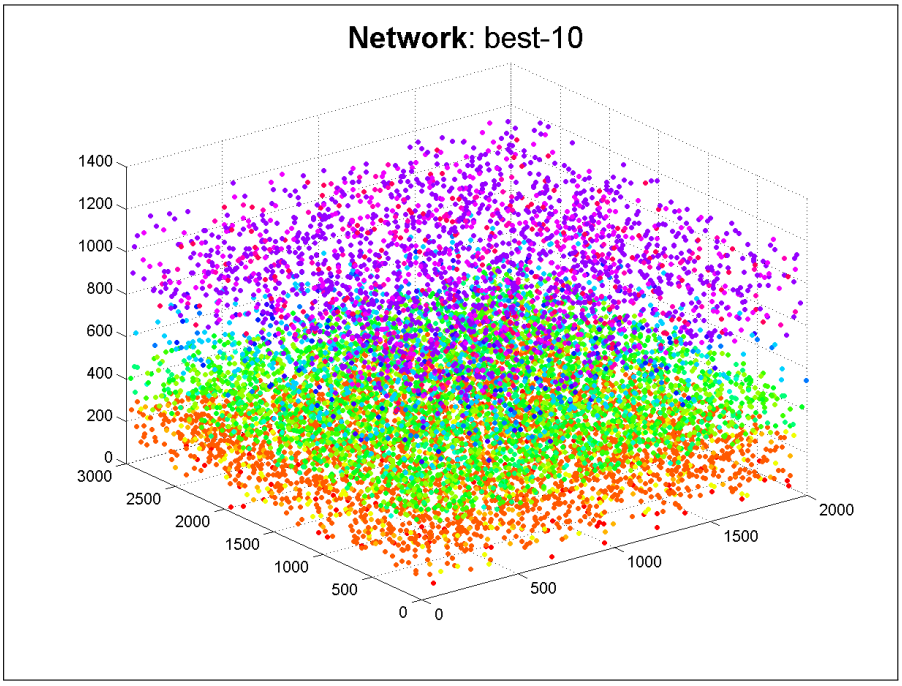


Figure 3.16: *This plot depicts the arrangement of the neurons in the network. The neuron types are distinguished by the same colours as in 3.13. Here, (atypically) from bottom to top:* ***L1*** *(very sparse),* ***L2/3****,* ***L4****,* ***L5****, and* ***L6****.*

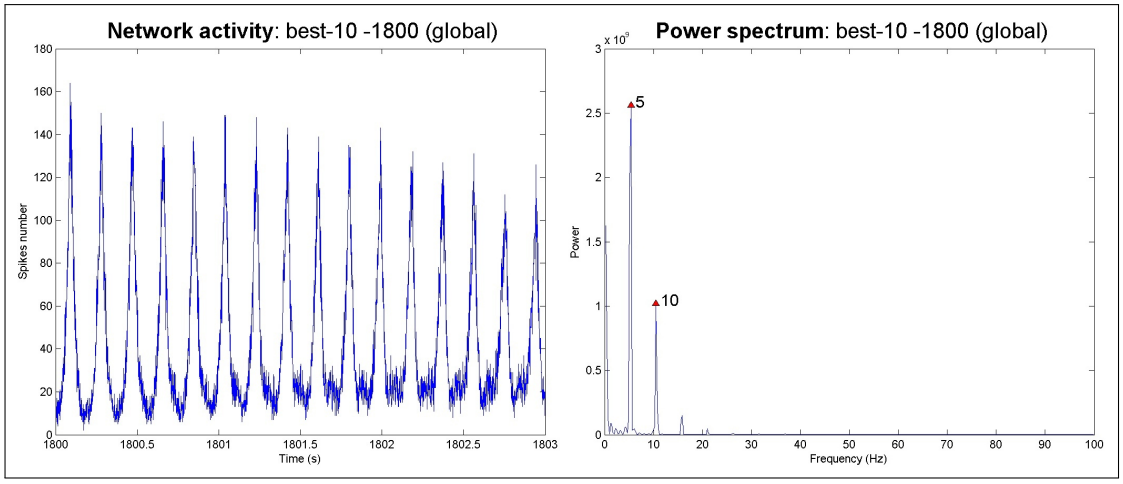


Figure 3.17: *The analysis of the oscillation spectrum of the entire network. Left: the plot of number of spikes in each tick over time. Right: the oscillation spectrum. The axis x corresponds to the wave frequency (in Hz) and the axis y corresponds to the oscillation power. Here, waves with frequencies 5 (theta) and 10 (alpha) are present.*

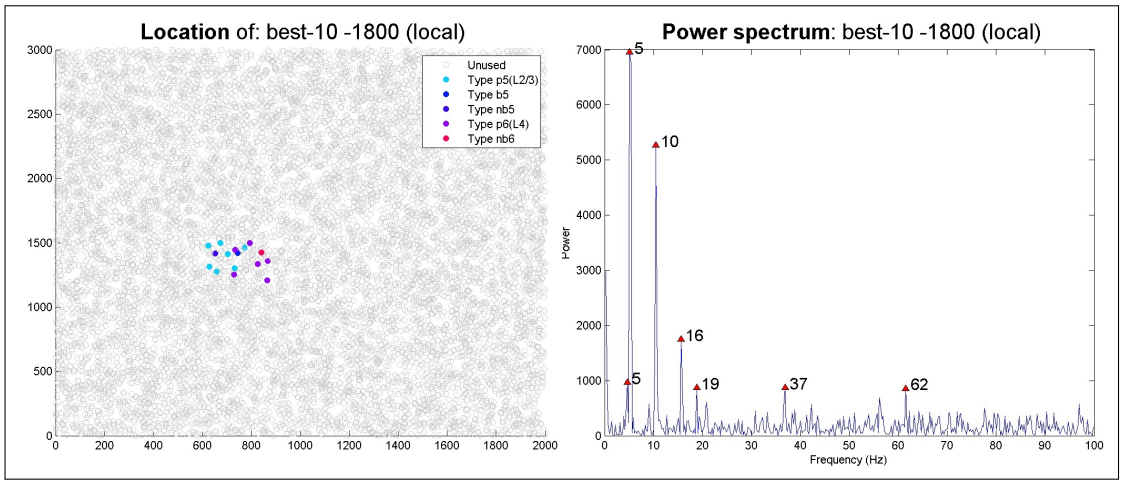


Figure 3.18: *The analysis of the oscillation spectrum of a local area. Left: the location of the measured area (here, from layers **L5** and **L6**). Right: the oscillation spectrum. The axis x corresponds to the wave frequency (in Hz) and the axis y corresponds to the oscillation power. Here, waves with frequencies 5 (theta), 10 (alpha), 16 and 19 (beta), 37 and 62 (gamma) are present.*

The algorithm chosen for computation of oscillations is based on the algorithm recommended in (Trappenberg, 2010, pp. 138). First, for each tick of a given interval, number of spikes is computed. Second, the Fast Fourier Transformation (FFT) of this vector is computed. Third, the absolute square of each (complex) Fourier component is computed and plotted against the frequencies. The values of wave frequencies used in the analysis 8 are listed in the Table 3.11.

Script B The script B displays the weight distribution: Figure 3.19.

Frequencies	Wave name
0.5–4 Hz	delta
5–7 Hz	theta
8–12 Hz	alpha
18–30 Hz	beta
30–50 Hz	gamma

Table 3.11: *The used names of oscillation waves.*

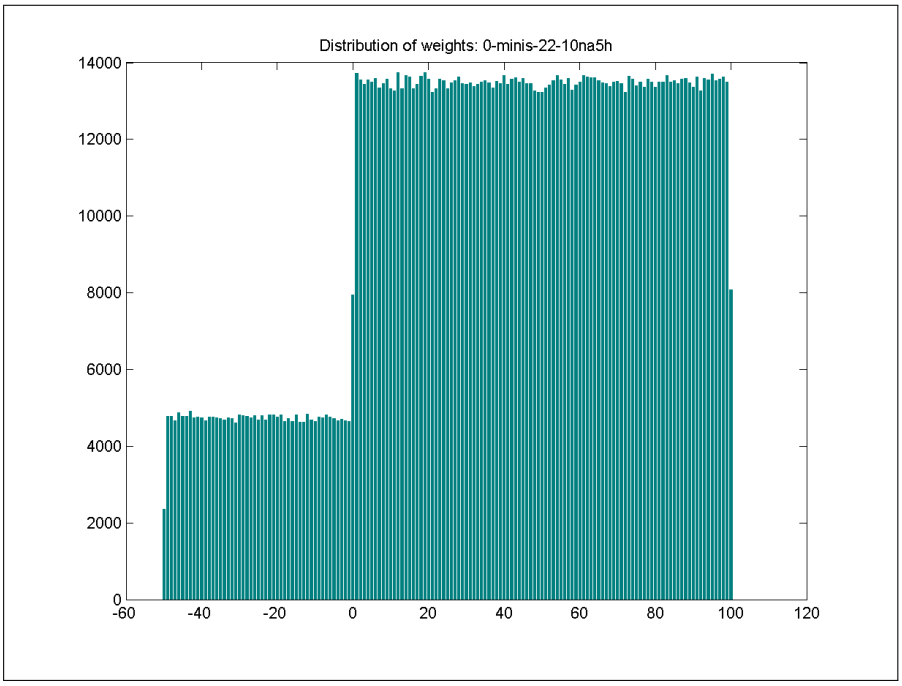


Figure 3.19: *The distribution of synaptic weights in the network. The axis x corresponds to the weight value (positive are excitatory and negative are inhibitory) and the axis y corresponds to the number of weights with that value.*

Script C The script C computes and displays the auditory-related features. They are described in the Section 7.5.3.

3.7 Other Software

The SUSNOIMAC tool uses the following Java libraries:

1. Parallel Colt (Wendykier and Nagy, 2010; Wendykier, 2013) for generating pseudorandom numbers with the Mersenne-Twister method (Matsumoto and Nishimura, 1998).

2. JMathPlot (Richet, 2013) for drawing 3D plots.

4. Simulator: Results

This chapter briefly describes the validation tests performed on the simulator and the performance of the simulator.

4.1 Validation tests

All parts of the simulator was tested and validated continuously during the software development. Namely, the following tests were performed:

1. Saving and loading interrupted simulation: test verified that the simulation is saved and loaded correctly and that this interruption does not change the experiment.[1]

2. Saving and loading the parallel reality: test verified the same features as in 1.

3. Parallel computing: test verified that parallel versions do not change the experiment.

4. Visualization: we manually verified that visualization part displays the right values.

5. Simulation of single neurons: we compared the behaviour of all neuron types used in the AC model is the same as when using the Matlab code provided by Izhikevich (2003).

6. Simulation of a basic network: we verified that a network published in (Izhikevich, 2006) simulated in SUSNOIMAC gives the same results as when using the C++ code provided by Izhikevich (2006) (except small differences caused by the pseudorandom generator).

4.2 Performance

The measurements of performance were performed on two machines:

1. "[PC]" processor: Intel(R) Core(TM) i7 CPU 930 @ 2.80 GHz, RAM: 6 GB

2. "[server]" processor: Intel(R) Xeon(R) CPU X5660 @ 2.80 GHz, RAM: 64 GB

We will refer to these machines as "[PC]" and "[server]". All measurements in this chapter are based on the values averaged over several experiments (5–10). However, the differences were negligible (lower than 1/100 of the measured value).[2]

[1] We must note that stochastic parts of the simulation are dependent on the state of the pseudorandom number generator.

[2] In addition to that, the aim of this section is not to perform an exhausting analysis of performance of different approaches to simulations, but to provide an idea about what improvements were important.

The speed of simulation is highly dependent on the used network (number of neurons and density of synapses), inputs (both external inputs and generated mPSCs). During the simulator development, we gradually implemented several computational improvements (described in the Section 3.5.4). To provide an idea of state before these improvements, the Table 4.1 contains duration (in real s) of simulation, which lasted 1s of model time, measured on the "[PC]". All other features (such as synapse density) are described in the Chapter 7 with the model description[3]. As we can see in the table, the simulation of larger networks was so slow that it could be hardly used (simulation of one hour (model time) would last almost 2 days).

1,000	10,000	50,000	100,000
6 s	147 s	2,730 s	N/A

Table 4.1: *Performance before improvements. The duration of execution of 1s (model time) simulation according to the network size (with the same density of synapses). The results from the network with 100,000 neurons are not shown due to high memory demands. All experiments measured on "[PC]".*

The most important improvements were parallelization and buffered array in the calendar. The improved array helped not only in the terms of time, but also memory consumption. Therefore, also the simulation of larger networks (such as with 100,000) was possible.

The Table 4.2 shows results from the final simulator version with all improvements. The measured experiments are exactly the experiments described in the Chapter 8. Here, we mention just the basic features: the mean firing rate ranged between 3–4Hz, the mean number of synapses to neuron was approximately 200 in average. The execution time of 1s is averaged over the whole experiment (all experiments lasted at least 1000s of model time).

1,000	10,000	50,000	100,000
3.15 s	4.39 s	19.83 s	33.73 s

Table 4.2: *Performance after improvements. The duration of execution of 1 s (model time) simulation according to the network size (with the same density of synapses). The two smaller networks were measured on "[PC]" (12 threads were used) and the two larger on "[server]" (32 threads were used).*

The scalability of simulator performance according to the number of neurons (with the same density of synapses) is shown also in Figure 4.1. We can see that the time of execution is approximately linear, which is a good result. Important factor for the linear dependency is that the firing rate was in all network sizes very similar.

Besides, we compared the execution time for each part of the simulation cycle. All these parts have been described in the Section 3.5.2. The results are listed in the Table 4.3 and Figures 4.1 and 4.2.

[3]With the exception that here the original form of the Izhikevich neuron model was used.

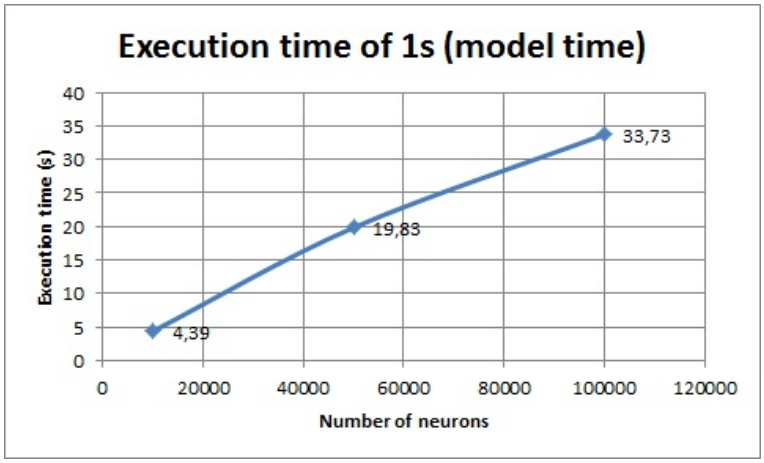


Figure 4.1: *The execution time of simulation of 1 s (model time) on networks with different number of neurons.*

NEURONS	INP	DET	PROP	UPD	END	LOGS	ALL
1,000	0.75 s	0.85 s	0.81 s	0.72 s	0.02 s	0.00 s	3.15 s
10,000	1.51 s	0.98 s	0.95 s	0.83 s	0.11 s	0.00 s	4.38 s
50,000	5.06 s	3.36 s	5.73 s	3.26 s	2.42 s	0.00 s	19.83 s
100,000	7.08 s	6.43 s	9.95 s	5.22 s	5.04 s	0.01 s	33.73 s

Table 4.3: *Performance after improvements according to the parts of the simulation cycle. The duration of each part (columns) in s (real time) of execution of 1 s (model time) simulation according to the network size (rows). The two smaller networks were measured on "[PC]" (12 threads were used) and the two larger on "[server]" (32 threads were used).*

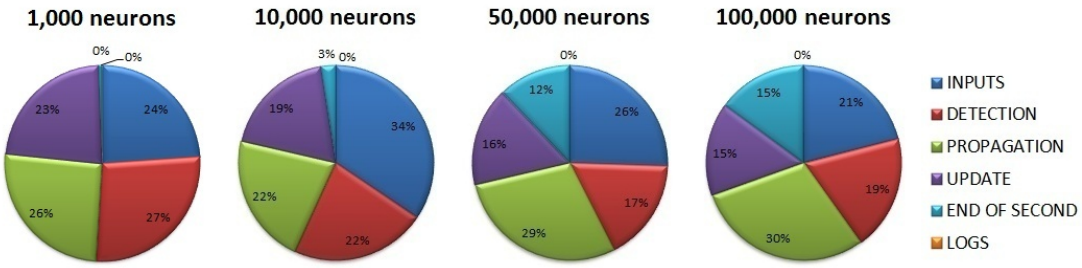


Figure 4.2: *The structure of the execution time of 1 s (model time) according to the parts to the simulation cycle and network size. The values are based on the Table 4.3.*

4.3 Outcome

The SUSNOIMAC simulator is able to simulate a network with 100 thousand of neurons and 21 million of synapses in reasonable time (34 times slower than real-time). A network with 10 thousand of neurons (and 1.6 million of synapses) is simulated only 4.4 ties slower than real-time. One of the main ways how to accelerate it in the future would be usage of Computer Unified Device Architecture (CUDA) Graphics Processing Units (GPUs), as it is used e.g. in (Nageswaran et al., 2009) .

5. Model of the Auditory Cortex: Biology Primer

The auditory cortex (AC) is a part of the auditory system, the sensory system which provides the sense of hearing (audition). The stimulus triggering auditory receptors is sound, a mechanical wave with certain frequency and amplitude. The frequency (with units called Hz) of the sound wave determines the sound pitch and amplitude determines the intensity (with units called dB). This chapter describes the basics of the auditory system (again many simplifications were made to capture just the basic facts). Unless stated otherwise, it is based on (Bear et al., 2007; Brodal, 2010; Purves et al., 2001; Watson, 2012).

The auditory system starts in ear (see Figure 5.1). The sound waves move the *tympanic membrane* in the ear. It moves the *ossicles* (series of tiny bones in the middle ear). They move the *membrane at the oval window* (a membrane covering a hole in the bone of the skull called oval window), which moves the fluid in the *cochlea*. The cochlea contains the apparatus to transform the physical motion of the oval window membrane into a neuronal response. Subsequently the neuronal signal is transmitted by the *auditory nerve fiber* to be processed by a series of nuclei in the brain stem (e.g., the *dorsal cochlear nucleus*, *ventral cochlear nucleus* and *superior olive*, where information from both ears is composed), and the *inferior colliculus* in midbrain, where all ascending auditory path ways converge. Output of the inferior colliculus is transmitted to *medial geniculate nucleus* (MGN) in thalamus. Axons leaving the MGN project to the AC.

The auditory cortex (see Figures 5.2 and 5.3) consists of several areas. The number of these areas ranges from five in mice (Watson, 2012) to over 30 in some studies of humans (Malmierca and Hackett, 2010; Stiebler et al., 1997). These areas may be divided into primary and secondary (belt) areas. An important primary area, common for many mammals, is the *primary auditory cortex* (A1). It is located in the superior temporal gyrus in the temporal lobe (in human) and receives direct inputs from the MGN. Belt areas of the AC receive more diffused input from dorsal part of the MGN.

The A1 consists of six layers. Layer L4 is the main recipient of the inputs from thalamus. Layers L2 and L3 (often marked together as L2/3) are largely innervated by L4 projections from L4 and they send outputs to other cortical regions for intercortical processing, while L5 neurons receive relayed information from L2/3 and project to various subcortical nuclei to modulate subcortical responses (Wu et al., 2011; Mitani and Shimokouchi, 1985; Winer and Prieto, 2001; Winer, 2005; Barbour and Callaway, 2008). Layer 6 sends cortical feedback specifically to the thalamus (Ojima, 1994; Prieto and Winer, 1999; Winer, 2005). Layer L1 is relatively sparse and seems to receive also a direct thalamic input (Mitani and Shimokouchi, 1985; Huang and Winer, 2000; Zhu and Zhu, 2004; Theyel et al., 2010), although the functional significance of this input remains unclear. Also L6 is weakly innervated by thalamus (Wu et al., 2011; Cruikshank et al., 2002).

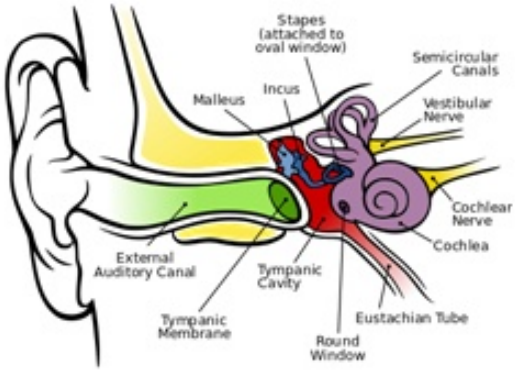


Figure 5.1: *A diagram of the human ear. From (Brockmann, 2009).*

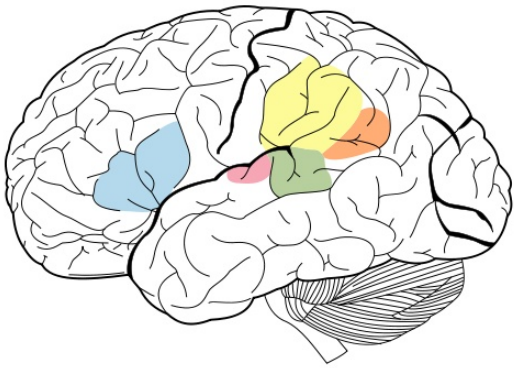


Figure 5.2: *A diagram of the human brain showing the* ***Primary Auditory Cortex*** *and surface gyri:* ***Angular Gyrus****,* ***Supramarginal Gyrus****,* ***Broca's Area****, and* ***Wernicke's Area****. From (wikipedia.org, 2007c).*

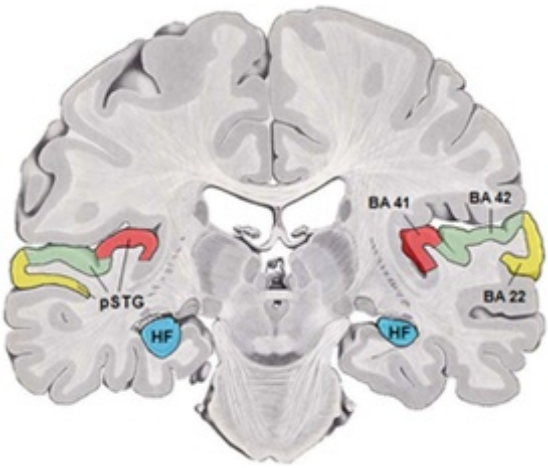


Figure 5.3: *A coronal section of a human brain:* ***BA41*** *and* ***BA42*** *belong to the Primary Auditory Cortex,* ***BA22*** *is Brodmann ara 22, and* ***HF*** *is hippocampal formation. From (Talbot, 2011).*

One of the main features of A1 is the tonotopic organization, analogous to retinotopic organization in the visual system. It means that many neurons in A1 are relatively sharply tuned for a particular sound frequency (called *characteristic frequency*, or *best frequency*[1]) and these characteristic frequencies are organized from low frequencies (in the human brain represented rostrally and laterally) to high frequencies (represented caudally and medially). This columnar organisation is sometimes referred to as isofrequency bands, running mediolaterally across A1. The tonotopic organization does not occur first in the AC, but originates from cochlea and is more or less maintained throughout the entire auditory pathway.

In addition to the frequency tuning, some neurons in A1 are tuned to a particular intensity. However, other neurons are barely tuned at all. Although reactions to variety of sound inputs (such as clicks, bursts of noise, frequency modulated (FM) sounds and amplitude modulated (AM) sounds, vocalizations, or even more complex sounds) are intensively researched, the processing of complex sound stimuli is still one of the currently faced challenges.

[1]The characteristic frequency (CF) is typically defined as the frequency, at which a least intensity is needed to stimulate it. The best frequency (BF) is typically the frequency, which leads to the highest activity of the neuron. However, sometimes also other definitions are used (we will also use an adjusted one later).

6. Model of the Auditory Cortex: Motivation and Related Works

This chapter is divided into three sections. In the first section we outline, what led us to create a model of the AC. The second section lists other published models of the AC, summarizes their major achievements and limitations and explains a need of the model designed in this thesis. The third section lays out the main features (requirements) of the model and explains the reasons for them.

Although, in this chapter, we write about models of the AC in general (e.g., some reviewed models concern human A1, one even bird analogous to the AC), the rest of the thesis concerns mainly the AC of mice.

6.1 Motivation

There are many unanswered questions about the AC. The fundamental question is how the auditory perception is coded, represented, and how the AC works at all. However, this question is very complex and complicated and even many of its sub-questions are still poorly understood. For instance, how is the spontaneous activity influenced by stimuli? How do different stimuli affect the features of the AC, such as receptive fields of neurons, their best frequencies or characteristic frequencies? How are the sounds coded and represented in the AC? Is there any specific and easy-to-describe representation for auditory objects[1]? Are auditory objects that are behaviourally important, represented differently? What role do the single neurons versus neuron populations play in these tasks? Other questions concern auditory disorders (Polster and Rose, 1998, e.g., cortical deafness, or auditory agnosia: see), or disease symptoms (e.g. tinnitus, which may be defined as "*conscious experience of a sound that originates in the head or neck, and without voluntary origin obvious to person*" (McCombe et al., 2001; McFadden, 1982).): where and why do they origin and by which neuronal features are they influenced by? These questions are intensively researched by *in vivo* and *in vitro* experiments. However, an additional method in the form of *in silico* experiments on a computational model could be very beneficial. It could be rewarding at least in the following applications:

1. Experiments, which are long or their conduction in real conditions is costly.
2. Experiments, which are impossible to run in real conditions (e.g., due to inaccurate or insufficient imaging techniques, or ethic).
3. To verify a hypothesis (which cannot be verified in real conditions) or a conceptual model. For instance: Is our concept of mechanisms of a certain function/phenomena right?

[1] "*An auditory object might be defined as an acoustic experience that produces a two-dimensional image with frequency and time dimensions.*" (Griffiths and Warren, 2004) As a practical example, it can be a voice of a particular individual, or sound of a musical instrument.

Besides these general questions, another motivation for a computational model of the AC was a hypothesis on emergence of tonotopy. Author of this hypothesis is Jakub Tomek. The following subsection describes his hypothesis.

6.1.1 Hypothesis on emergence of tonotopy

One of the well-known features of the AC is the tonotopic organization based on tuning to sound frequency. The tonotopic maps can be observed using large-scale imaging techniques and multi-unit recordings (Bizley et al., 2005; Nelken et al., 2004; Stiebler et al., 1997). However, with improving possibilities of fine-scale imaging and recording techniques (such as *in vivo* two-photon calcium imaging), surprising lack of tonotopic homogeneity on the fine scale was revealed (Bandyopadhyay et al., 2010; Rothschild et al., 2010). Since then, the local heterogeneity attracted attention of many auditory neuroscientists, but the mechanisms of its development are still poorly understood.

We propose a hypothesis, that the coarse tonotopy, but local heterogeneity in all layers of the AC may be an emergent result of tonotopic input from the thalamus and long-term plasticity, which could play a key role in this process.

A mouse brain develops most rapidly during the first month after birth (Oswald and Reyes, 2008) (this period is typically called *critical period*). Several studies researched the impact of sound listened during this period on the resulting tonotopy. When noise was played to mice throughout the critical period, it led to a severe impairment of tonotopy in the AC (Chang and Merzenich, 2003; Zhang et al., 2002). On the contrary, in the normal conditions, the tonotopy emerged, but only coarsed. Other experiments with pure tones (monotone stimuli) were conducted and indicated that their number and frequency may also have impact on the resulting receptive fields and state of tonotopy (Zhang et al., 2001).

We propose that the development of the AC can be viewed as follows: The starting point of the system is the tonotopic input to L4, with other layers rather chaotic (or "randomly") initialized. Then, when a stimulus is played to the animal, the relevant part of inputs starts "conquering" zones in the formerly chaotic part of brain via mechanisms of brain plasticity (e.g., STDP). By "conquering", we mean enlargement of the zone responding to a given stimulus.

This view is in good agreement with the observation above: noise should lead to development of very weak tonotopy and exposition to only a few pure tones should lead to a rather weak tonotopy as well. The critical period in animals is probably being optimized towards "natural" sounds, i.e., when the animal hears ordinary sounds, the brain develops in a useful way. However, when only a several tones are being heard by the animal, the critical learning period is too short for the zones belonging to the several tones to "conquer" the whole possible area of the AC, therefore keeping a lot of the original chaotic organization.

6.2 Related Works

This section reviews the existing models of the AC: exactly the AC, or part of the AC (typically A1), or on the contrary a larger model with more auditory stages and AC being one of them. Models of lower stages of the auditory pathways

without AC are not included, but they can be found in existing well-arranged and detailed reviews (Meddis, 2010; Greenberg, 2001).

Although the AC hides many secrets, which could be revealed using computational models, the modelling of the AC is relatively unexplored area. There are several models published, but their number is much lower than e.g., in visual cortex, or lower stages of the auditory pathways. Moreover, to our knowledge, no synoptic review of the AC computational models with spiking neurons exists. The only review of AC models, which we are aware of, was written by Eggermont (Eggermont, 2010) and concerns only more abstract or conceptual models (high-level using the terminology from the Section 1.2.1). Therefore, the review, which we provide in this section, is in some way the first of its kind.

We divided the computational AC models into three groups: the spiking neuron models, the firing rate models, and other models, typically more abstract. The first group is closest to our aim; therefore we will describe them slightly more in depth. On the contrary, the third group is the farer one and we will describe only two representatives from this group even though this group is larger and contains for instance a specific class of computational models focusing on auditory scene analysis (Brown and Cooke, 1994; Fodróczi and Radványi, 2006, e.g.,), which we do not describe in this text.

We also do not describe the conceptual non-computational models, such as models of long-term memory traces in the A1 (Weinberger et al., 1990; Suga and Ma, 2003, e.g.,), models of hierarchic representation of auditory objects in the AC (Sharpee et al., 2011, e.g.,), or general models of representation, transformation, and coding of sound in the A1 (Eggermont, 2001, e.g.,).

The aim of the following text is not to provide an exhausting description of the computational AC models, performed experiments and their results. This information is (typically) provided in the referred publications. Instead, goal of this text is to summarize the basic features:

1. Motivation for the model
2. A basic description of the model: populations of neurons, number of modelled neurons, main characteristics of the connectome, and the used models (dynamics) of neurons, synapses and possibly channels
3. Other information about simulation, such as method of model computing, or time step (we include these information, only if they are provided by the referred publications)

In addition to that, in the case where it is possible (and where it would not require long explanations), we try to summarize also:

1. Main measured features and performed experiments
2. Some main results, which are clear enough to be summarized in few sentences

6.2.1 The spiking neuron models

6.2.1.1 Model by Pinho, Mazza and Roque (1999–2006)

This model was gradually developed and published in (de Pinho and Roque-da Silva, 1999; de Pinho et al., 2000, 2001, 2002, 2006), all versions were generally similar with minor differences or different experiments conducted. The version from 2006 is the most extended one and we will therefore base the following description on this version. The model contains four structures: cochlea, MGVBv, MGVBm, and A1. Cochlea is modelled as 47-receptor linear layer. Each MGBv and MGBm contains 564 excitatory relay cells and 186 inhibitory cells arranged in an array. The A1 area contains 3384 excitatory pyramidal cells, arranged in an array of 188 $\times$ 12 nodes, and 1128 inhibitory basket cells, arranges in an array of 94 $\times$12 nodes. The connectome is based on specified probabilities, such as that "*each pyramidal cell is connected to itself and all pyramidal cells in its first ring of neighbours and has a probability of 50% of being connected to pyramidal cells in its second ring of neighbours*" (de Pinho et al., 2006). The description of the connectome is detailed enough for reproductions of the model. On the contrary, the neuron models are described only approximately by basic features: number of compartments (1–6), types of ionic channels and receptors (AMPA, NMDA, and GABA). The only details of the used neuron models can be found in the first version (de Pinho and Roque-da Silva, 1999), but the reproducibility from this description would be limited.

Most experiments done with the model are related to tonotopic organisation of cortical neurons and influence of NMDA receptors on this organisation. The description of the experiments and their results do not contain many details. Each experiment lasted in order of few seconds of the model time.

6.2.1.2 Model by Larson, Billimoria, Perrone, and Sen (2008, 2010)

The model was published in (Larson et al., 2008, 2010). The motivation for this model was to design and implement a neural circuit capable of songbird recognition. In (Larson et al., 2008), authors proposed 3 possible solutions (a coincidence detection circuit, a rate detection circuit, and a vR circuit with 3 LIF neurons) and in (Larson et al., 2010) an extended model with chain of 20 LIF neurons. The basic idea is the computation of dissimilarities between spike trains using the van Rossum spike distance metric (Rossum, 2001). The synaptic connections between neurons have specified conduction delays and the synapse weights are modified by STDP. The spike trains recorded from field L (an analogous to the auditory cortex in mammals) of zebra finch were used as model inputs. The model was tested on 100 repetitions of 20 zebra finch songs truncated to 820 ms and achieved relatively high performance (>98% accuracy) in the given task.

6.2.1.3 Model by Zhou et al. (2012)

The motivations for the model (Zhou et al., 2012) were questions about spike latency tuning: how is it generated, what role does in it play the background spontaneous activity, or what is the relation between cortical and thalamic latency tuning and. The model consisted of four layers, each containing 800 cells: thalamic neurons, inhibitory interneurons, excitatory interneuron, and pyramidal

neurons. The connections were deterministic. All the cortical cells were modelled as LIF neurons. The input consisted of background activity modelled as a train of Poisson events. Several features concerning spike latency tuning were measured (comparison between neuronal populations, impact of the background activity, etc.). The main results from both the model and *in vivo* observations were related to influence of inhibition and excitation on spike latency tuning at optimal and off-optimal frequencies.

6.2.1.4 Model by Chrostowski et al (2011)

The motivation for the model (Chrostowski et al., 2011) was to explore the conditions under which travelling waves occur in the AC and their relation to hearing loss. The model consisted of the thalamus layer with 201 neurons, and the A1 layer with 201 pyramidal neurons 67 inhibitory interneurons. Neurons in all the three populations were uniformly distributed along the latero-medial tonotopic axis. The connectome was deterministic and initial synaptic weight was defined as a Gaussian function of distance. All cortical neurons were modelled as LIF neurons. A novel mechanism of homeostatic plasticity of synaptic weights was designed and used, but no other types of plasticity (e.g., Hebbian-like such as STDP) were used. Inputs from thalamus were generated as Poisson process. Besides, the effects of peripheral hearing loss were modelled by reduction of thalamic inputs at impaired frequencies. Four degrees of hearing loss were researched. Model equations were solved numerically using Runge-Kutta method and fixed time-step of 0.1 ms.

During simulations, several features were measured, such as distribution of the synaptic weights, mean firing rate, cross-correlation of spike trains, or occurrence of travelling waves in dependence of degree of hearing loss. One of the main results were qualitatively same changes after hearing loss as in real (except of differences in several features, such as shifts in characteristic frequency) and observation that spontaneous activity prevents the development of long-range travelling waves of excitation.

6.2.2 The firing rate models

6.2.2.1 Model by Harris et al. (2011)

The motivation for the model (Harris et al., 2011) was to search for reasons in complex dependence of sensory responses on dynamic and activity states at the time of stimulus presentation. The main goal of the model was to show, that sensory responses are shaped by the same dynamics that generate spontaneous activity prior to stimulus presentation. The model was based on two Fitzhugh-Nagumo equations (FitzHugh, 1955) with two variables v (average population firing rate) and w (related to reducing network excitability) and 5 parameters. The ability of the model to predict the structure of activity after sensory stimulus was tested and compared to MUA (multi-unit-activity) and LFP recordings.

6.2.2.2 Model by Schiff, Reyes, de La Rocha, and Marchetti (2008, 2011)

The motivation for the model (Schiff and Reyes, 2012) was to examine how feedforward inhibition and synaptic depression affect cortical responses to time-varying inputs that mimic sinusoidal amplitude-modulated tones. As a predecessor if this model can be identified a firing rate model designed four year before this one (de la Rocha et al., 2008). The model consisted of three populations: one layer of 50 thalamic neurons, population of excitatory neurons (E) and population of inhibitory neurons (I), both with firing rate dynamics (de la Rocha et al., 2008; Wilson and Cowan, 1972). Both the connectome and inputs are clearly described with all necessary details. The thalamic neurons were evenly distributed along a one-dimensional tonotopic axis and had only a transfer function to the two other populations (input to I was to times stronger than to E). In addition to that, I inhibited E. Synaptic weights were influenced by a short-term depression. The parameters of the model were adjusted to preserve differences in RS and FS cells observed experimentally (e.g., their f-I curves, strength of thalamic input, and degree of thalamic short-term depression). Differential equations were solved numerically with the forward Euler method with a fixed time step 0.1 ms. All experiments lasted up to 1000 ms of the model time.

6.2.2.3 Model by Loebel, Nelken and Tsodyks (2007)

The motivation for the model (Loebel et al., 2007) was to explore the hypothesis that temporally locked spiking activity of A1 neurons in response to auditory stimuli reflects the near coincident activity of ensembles of neurons with similar best frequency. They supposed that this temporal coherence emerges from intrinsic properties of the intracortical circuitry (such as short-term depression at recurrent excitatory connections). Besides, the model was designed to explore the influence of sharp transient synchronization of network activity (called Population Spike). The model consisted of 15 cortical iso-frequency columns; each simulated by a fully connected recurrent network of 100 excitatory and 100 inhibitory neurons with firing rate model dynamics (Wilson and Cowan, 1972). In addition to connections within a column, neurons from nearby columns (distance up to 2 columns) were connected. Synaptic connections were influenced by the short-term plasticity. All necessary details about dynamics, connectivity, and inputs (sensory inputs were introduced only to excitatory neurons, but all neurons received a background input leading to spontaneous activity) are described sufficiently for reproductions. Many characteristic were measured, such as frequency tuning curves, features of excitatory and inhibitory responses (e.g., temporal ordering), two-tone interactions, or response to frequency modulated sweeps.

6.2.2.4 Model by Bart, Bao, and Holcman (2005)

Motivation for the model (Bart et al., 2005) was to determine how neural connections may influence EEG dynamics and to elucidate how epileptic regime in the AC can emerge. The model consisted of 100 firing-rate neurons (similar to (Wilson and Cowan, 1972), but different). The synaptic connections were generated according to the distribution described in (Bao et al., 2003) and their weight

was directly proportional to the length of the connection (distance of the presynaptic and postsynaptic neurons). The synapses were influenced by the short-term depression. To achieve a spontaneous activity in the network, an additive random Brownian noise was modelled and sent to neuron inputs. The differential equations were solved numerically using the Runge-Kutta-Fehlberg method with adaptively controlled step size to keep the error low (step size was always kept below $2 \cdot 10^{-5}$s). Several characteristics were measured, such as stability of the system (a degree of epileptic behaviour), mean voltage, depression rate and synchronization index in dependence on the synaptic weights (weak, normal, and strong). One of the main results was achievement of two regimes of the spontaneous activity (normal and epileptic) with dependence on the synaptic weights.

6.2.3 Other models

6.2.3.1 Model by Otazu and Leibold (2011)

Motivation for the model (Otazu and Leibold, 2011) was to provide a possible explanation of sound identification in complex scenes. The goal of the model were not realistic anatomy and morphology features of neural network, but an computer algorithm, which gets abstract inputs corresponding to the real auditory inputs and recognizes auditory objects in them. Authors proposed an abstract dictionary-based algorithm called Corrected Projections Algorithm (CPA). It uses a dictionary of known sounds. The main idea is the minimization of the difference between the sensory representation of the incoming sound and an internal estimate to identify sources present in the auditory scene. In addition to measured success of the model in the object recognition task, they analysed simple features of neurons (such as number of spikes during spontaneous activity versus in response to a repetitive sound stimulus) in the rat AC. These recordings showed several similarities to the proposed model suggesting that the thalamocortical circuit may implement an iterative version of the proposed CPA algorithm.

6.2.3.2 Model by Mesgarani, Fritz, and Shamma (2009)

Motivation for the model (Mesgarani et al., 2009) was to catch several features of receptive field properties of auditory cortex neurons (such as their rapid adaptation during discrimination and detection tasks, especially where specific sounds are related to reward or punishment). Authors proposed a mathematical model of cortical receptive fields modelled as filters that change their spectro-temporal tuning properties in order to extract the discriminatory features relevant to the ongoing behavioural task. The model was compared to the *in vivo* observations of plasticity of the receptive fields of the neurons in the ferrets AC during different types of spectral and temporal discrimination tasks (such as detecting a tone or multiple tones against a contrasting background of spectro-temporally modulated noise, or discriminating between tones of different frequencies or rates). The model predicted generally the same results as the physiological data: the plasticity effect for cells tuned to frequencies near the target tone is bigger when the reference and target are more distinct.

6.2.4 Concluding remarks to the related works

To conclude the review of the AC models, the most accurate (i.e., with many measured characteristics and their comparison to the real data) and precisely described computational models of the AC, described in this review, were the firing rate models. On the contrary, not many spiking neuron models of the AC exist and those reviewed in this text are very limited (see also the Table 6.1) in the following features[2]:

1. Size (up to 4600 neurons)
2. Detailed connectome based on the real data
3. Diversity of neuron types (only one excitatory and one inhibitory, e.g., pyramidal and basket cells)
4. Model of neuron: except of the model by Pinho, the model type was a simple leaky integrate-and-fire model (LIF, I&F), which has very limited plausibility – citing Izhikevich "*... the model [I&F] cannot exhibit even the most fundamental properties of cortical spiking neurons, and for this reason it should be avoided by all means. The only advantage of the I&F model is that it is linear, and hence amenable to mathematical analysis. If no attempts to derive analytical results are made, then there is no excuse for using this model in simulations.*" (Izhikevich et al., 2004a)
5. Synaptic plasticity (only short-term, if any)
6. Validation and comparison to the real observations
7. Duration of simulations (up to few seconds)

Main author	Size	Neuron types	Neuron model	Synaptic plasticity	Duration
Pinho et al.	4512	p, b	HH-type	-	up to few s
Larson et al.	20	E	LIF	STDP	up to few s
Zhou et al.	2400	E, I, p	LIF	-	up to few s
Chrostowski et al.	268	p, I	LIF	homeostatic	up to 30s

Table 6.1: *Comparison of the existing spiking models of the AC. Size: number of neurons in the AC part of the model. Neuron types: neuron types in the AC part of the model (p=pyramidal, b=basket, E=general excitatory, I=general inhibitory). Neuron model: used model for the neurons in the AC parth of the model. Synaptic plasticity. Duration: duration of performed experiments.*

On the other hand, models, which fulfil all these features, but do not focus on the AC, exist: for instance the large-scale (with 10^6 neurons and almost $5 \cdot 10^8$ synapses) model of mammalian thalamocortical system by Izhikevich and Edelman (Izhikevich and Edelman, 2008). This model exhibits several features of the real mammalian cortex, such as emergence of waves and rhythms. However, the

[2]Not each of them is limited in all listed features, but each in at least several of them.

model was observed only during the spontaneous activity with no other specific sensory inputs.

Therefore, we decided to fill up these two gaps and design and implement a model of the AC, based on the thalamocortical model proposed by Izhikevich and Edelman (Izhikevich and Edelman, 2008), but adjusted to the features of the AC (mainly mouse AC, because of the access to the data from AC of mice) and sound inputs from the auditory thalamus.

6.3 Model features

Finally, we will conclude this introductory chapter about our model with the features, which we considered to be important for the model. All subsequent works (simulator choice and development, and model design and implementation) were subordinated to these features.

The chosen features of the model are identical as the first group of simulator requirements, described in the Chapter 2, and the simulator software was developed to meet all these conditions. For convenience, we will list them again and explain the reasons for including them:

1. The generalized form of the Izhikevich neuron model.
2. Synaptic connections with specified synaptic conduction delays and changeable weights, formed by the long-term plasticity, e.g. STDP.
3. Sufficient computational precision.
4. Sufficient flexibility in network structure definition: e.g., different probabilities of synaptic connections between specific neuron types in specific dependence on distance (for instance as it is in (Izhikevich and Edelman, 2008)).
5. Sufficient flexibility in definition of the network inputs: e.g., each tick another neurons will be stimulated with value specified for each neuron.
6. A mechanism, which would lead to spontaneous activity, e.g., the mPSCs.

Each requirement/feature was selected for carefully considered reasons:

1. The Izhikevich neuron model was chosen for the (to our knowledge) optimal ratio between plausibility and computational efficiency and to its ability to reproduce the behaviour of the main neuronal types in the auditory cortex: regular spiking (RS) neurons, fast spiking (FS) neurons, and low-threshold spiking (LTS) neurons.
2. The synaptic connection delays play a crucial rule in coding information: they lead to significantly increased information capacity and specific stable firing patterns, which would be impossible without delays (Izhikevich, 2006). The synaptic weights are fundamental for development and memory of the network. The long-term plasticity is vital in research long-term changes, such as those in the hypothesis on emergence of tonotopy. STDP is a widely used form of long-term plasticity.

3. Precision is important for the reliability of the results. However, a higher precision usually leads to a higher computational cost, so a compromise must be done.

4. The flexibility in network definition is needed to reflect the diversity of the real network.

5. The flexibility in definition of the network inputs is necessary to stimulate the network with inputs corresponding to the inputs from thalamus.

6. Spontaneous activity supposedly plays a critical role in higher cognitive processes, such as decision making or working memory (Wang, 2003) and in synaptic maturation during development (Gonzalez-Islas and Wenner, 2006).

We believe that having a computational model of the AC using the state-of-the art computational techniques could help in searching for answers to many of the unexplained phenomena in the processing the sound stimuli. Naturally, creating a state-of-the art model of the AC requires not only good modelling techniques and software, but also a comprehensive knowledge of the AC, types of experiments (which are run in real conditions and could be run also on the model to compare the results), typical results of these experiments, huge amount of accurate data, and many other things. Therefore, this thesis suggests rather a basic version of such a model. However, it may be extended: the approximate parameters (e.g., based on non-auditory parts of cortex) may be replaced by the more accurate data, when these are researched, and the developed simulation software and tools for results analysis can be used without changes.

7. Model of the Auditory Cortex: Methods

This section describes the created model. It follows the guidelines suggested in (Nordlie et al., 2009), which, as we hope, will help to make the description well comprehensible and reproducible.

The model is based on two models by Izhikevich (Izhikevich and Edelman, 2008; Izhikevich, 2006) and adjusted to the AC features (inputs, spontaneous activity and some other intrinsic parameters). Although we call it model of AC, it represents mainly the part of the primary auditory cortex (and only from one half of the brain). The model has several groups of parameters: general parameters, parameters related to neuron types, connections, or layers. We will describe them successively at that part of the text, where they are first needed. However, their synoptic list with all descriptions is placed in the end of the chapter, as well as the used values.

The model was implemented in the SUSNOIMAC simulator, which is described in the Chapter 3. Let us mention that all used parameters may be easily changed in settings files and even the numbers of neuron types and layers may be both increased (then the relevant parameters of the added neuron types and layers must be specified), or decreased (then the relevant parameters must be removed). This makes the implementation easily extensible for possible fitting to more accurate data.

7.1 Model composition

The model uses 6 layers, but layers 2 and 3 are treated as one layer (that is a common approach, because the distinction between these layers in real cortex is not clear (DeFelipe et al., 2002)). Parameters of the layers are listed in the Table 7.1. The layer names of used layers are: L1, L2/3, L4, L5, and L6. Only the layer L4 is an input layer.

Label	Type	Description
`layer`	string	name of the layer
`thickness`	integer	thickness of the layer (in μm)
`input`	boolean	true, if it contains input neurons

Table 7.1: *The parameters of each layer in the model.*

The neurons are distinguished according to the populations, called neuron types. Parameters of the neuron types are listed in the Table 7.2. All neurons of the same neuron type lie in the same `layer` of the cortex and have the same parameters of the neuron model – the generalized form of the Izhikevich neuron model (Izhikevich, 2007; Izhikevich and Edelman, 2008). The number of neurons of this type is specified in percentage with the name `cells`, i.e. the resulting number of neurons of this type is `cells`*`N_NEURONS`, where `N_NEURONS` is the total number of neurons in the network and it is a general parameter of the model.

Label	Type	Description
`name`	String	name of the neuron type
`layer`	Integer	layer, in which the neurons of this type lie
`cells`	Integer	percentage of neurons of this type
`type`	String	additional information
`excitatory`	Boolean	true for excitatory, false for inhibitory
`Cap, k, vr, vt, vp, a, b, c, d`	Double	parameters of the Izhikevich neuron model

Table 7.2: *The parameters of each neuron type in the model.*

In addition to these parameters, each neuron type has a specified area reached by its axon (and its terminals). This area is specified as an axonal radius (in μm) for each layer of the network (see Figure 7.1). We will refer to these values as table **X0**, where **X0**[T][L] means value of axonal radius of neuron type T in layer L. These values are used during generation of synaptic connections: a synapse can be created only between such neurons, where soma of postsynaptic neuron is reached by axonal area of presynaptic neuron. This is a simplification based on (Izhikevich and Edelman, 2008).

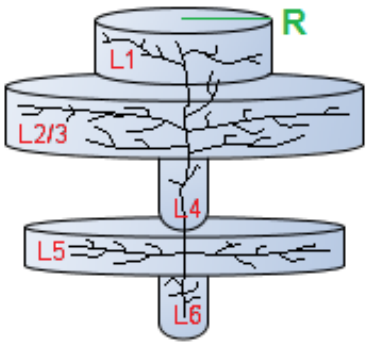


Figure 7.1: *An example of an axonal area. It is in each layer specified by the value of* ***R****: radius of the axonal area in this layer.*

The concrete parameters of used 17 neuron types are listed in the Section 7.8. The parameter `type` (additional information) is used to distinguish between 4 cell types:

1. pyramidal neurons (p): they exhibit regular spiking (RS), or sometimes chattering (CH), or intrinsically bursting (IB) firing patterns (Connors and Gutnick, 1990; Contreras, 2004)

2. spiny stellate neurons (ss): they exhibit RS firing patterns (Contreras, 2004)

3. basket interneurons (b): they exhibit fast spiking (FS) firing patterns (Connors and Gutnick, 1990; Contreras, 2004)

4. non-basket interneurons (nb), which morphologically include double-bouquet cells, neurogliaform cells, and Martinotti cells (Binzegger et al., 2004) and can exhibit low threshold spiking (LTS) firing pattern (Beierlein et al., 2003), latent spiking (LS) and other firing patterns (Kawaguchi, 1995; Kawaguchi and Kubota, 1997; Markram et al., 2004)

The `names` of neuron types are based on nomenclature as in (Izhikevich and Edelman, 2008; Beierlein et al., 2003), where the first part is the cell type (b, nb, p, ss), the second part is the layer and optional third part in brackets is the layer, to which the neuron projects (only in ss and p neurons), for example $p5_{(L2/3)}$ means pyramidal neurons in L5 that project (mainly) to L2/3.

The algorithm for initializing neurons, their locations and other parameters is described in the Pseudocode 7.1. The locations are generated randomly in the relevant layer, but in such a way that neurons do not overlap. Here, the general parameter `RADIUS` is used (all neuron's use the same parameter radius). By neuron's location we mean the location of neuron's soma. Other parameters are used according to the neuron type.

```
1. i=0;

2. For each neuron type T:

   (a) N_T ← T.cells * N_NEURONS / 100;

   (b) For j from 0 to N_T:

        i. Create a neuron with the following parameters:
           A. number i
           B. random location in the layer T.layer, which does
              not overlap any previous neuron
           C. T.excitatory
           D. T.layer
           E. membrane potential T.vr and membrane recovery 0
           F. parameters T.Cap, T.cr, T.vt, T.vp, T.a, T.b,
              T.c and T.b
       ii. If (T.layer is input layer) then
           A. add the created neuron to a band
              neuron.location.y / BAND_WIDTH
      iii. i++;
```

Pseudocode 7.1: *The algorithm of the generation of neurons in the model.*

7.2 Coordinate systems and topology

All neurons are numbered from interval ⟨0, N_NEURONS), each neuron having a unique number. According to the coordinate systems listed in (Nordlie et al., 2009) we use the *anatomical coordinates*, because population (neuron type) determines the layer within the AC and each neuron has a location that refers to position in the AC.

7.3 Connectivity

The connectivity mechanism is based on the mechanism described in (Izhikevich and Edelman, 2008). However, the connectivity is not explained there in details sufficient for reimplementation (this opinion has been voiced also by Nordlie (Nordlie et al., 2009)). Therefore we proposed (in cooperation with neuroscientists) additional mechanisms and algorithms to make a clearly defined connectivity.

Since the connectivity mechanisms are not trivial and we want to describe all topics related to them with enough details (sufficient for reimplementation and with enough explanations), the section is divided into several subsections. First, the required data are described. The following 4 subsections address the problems, which were necessary to solve: which neurons should be connected, how to set the synaptic conductance delays and how to initialize the synaptic weights. The Subsection 7.3.5 describes the proposed connectivity algorithm in the basic form and extended by one improvement. The Subsection 7.3.6 concern the possible pitfalls.

7.3.1 Connectivity data

The connectivity mechanism uses two tables. First table (table **X1**) defines for each neuron type T the recommended number of synapses which lead to each neuron of this type. We will call this number **X1**[T].

Second table (table **X2**) has three dimensions: postsynaptic neuron type T_post, presynaptic neuron type T_pre, and layer L of the synapse location. The value **X2**[T_post][T_pre][L] means that each neuron of type T_post should receive **X2**[T_post][T_pre][L] % of its synapses from neurons of type T_pre in layer L. The range of the values in this table is $\langle 0, 100 \rangle$. The resulting number of synapses from T_pre neurons to a neuron of T_post type in layer L is:

X1[T_post] * **X2**[T_post][T_pre][L] / 100

7.3.2 Selecting presynaptic candidates

7.3.2.1 Selecting presynaptic candidates – description

In the algorithm of generating connectivity, it was necessary to solve the following situation. The tables **X1** and **X2** determine the number of synapses to neuron N_post (of type T_post) from neurons of type T_pre in the layer L. Let S be this number. As presynaptic neurons may be chosen only those, who are near enough – for simplicity only those, of which axonal radius in layer L reaches the soma of the neuron N_post. The nearer neurons should have higher probability

of connection than the distant ones (Izhikevich and Edelman, 2008; Levy and Reyes, 2012). For simplicity we can consider that this probability will linearly decay from the centre of the circle defined by the axonal radius to a zero value at the edge (as it is used in (Izhikevich and Edelman, 2008)). The distance d can be considered only two dimensional (in our coordinate system named as x and y) – not in the dimension of height (which goes through all layers). This is reasonable for the fact that if axon goes through a layer (its radius is in this layer nonzero), then the distance from the axis of the axon should be considered. As a metric, the Euclidean distance (Black, 2004) can be used. For simplicity, in this model, axons grow directly vertically. The situation is depicted in the Figure 7.2, the relevant distance has a label d in the figure.

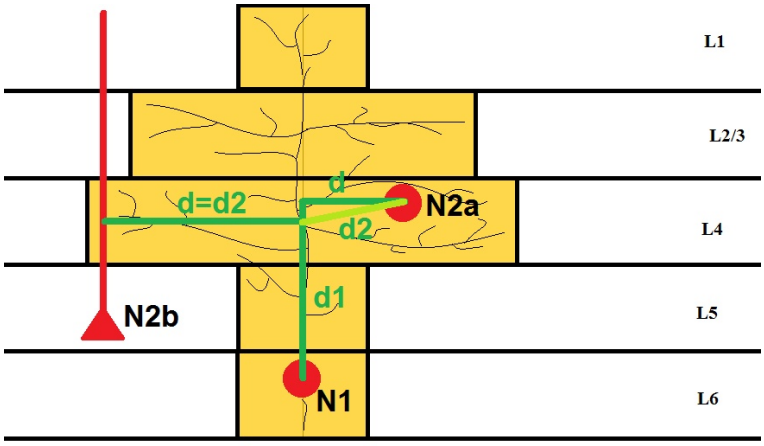


Figure 7.2: *A schema of the possible connections. The neuron N1 is a presynaptic neuron and N2a and N2b are postsynaptic neurons. (N2b is a pyramidal neuron.) Distance d1 and d2 will be used during initialization of the conductance delays. Distance d is used in computation of probability of the connection.*

This means that we have a set K, a set of candidates, from which may lead a connection to neuron N_post, and each candidate N_pre has a clearly probability of this connection. We denote this probability by N_pre.P. Subsequently we want to choose S candidates (presynaptic neurons) from the set K and create a synaptic connection from these chosen candidates to neuron N_pre.

7.3.2.2 Selecting presynaptic candidates – solution

Hence, the situation is simplified to abstract mathematical problem how to select S elements from set of $|K|$ candidates, where each has a given probability of selection. One of the typical computer-science solutions of this problem is the *Roulette Wheel Selection* (used in the Genetic Algorithms, see e.g. (Goldberg, 1989)). We create roulette with $|K|$ pockets, where pocket N_pre has a size N_pre.P. Subsequently we throw a ball in the roulette and spin the wheel and the element (neuron) associated with the winning pocket is selected. The wheel is spun S times. Therefore we say that we throw (in sum) S balls in the roulette, which determines S winning candidates. The Roulette Wheel Selection is depicted in the Figure 7.3.

It is obvious that bigger pockets have bigger probability to be selected. It is good to consider that we do not mind selecting one candidate more times (this

means that more balls fell to the same pocket). In such a case, more synapses between one pair of neurons are created (we call it a multi-connection), which corresponds to the reality (see the Section 7.3.6). Besides, the algorithm could be easily adjusted to avoid multi-connections (e.g., when a pocket is selected, we remove the pocket from the roulette).

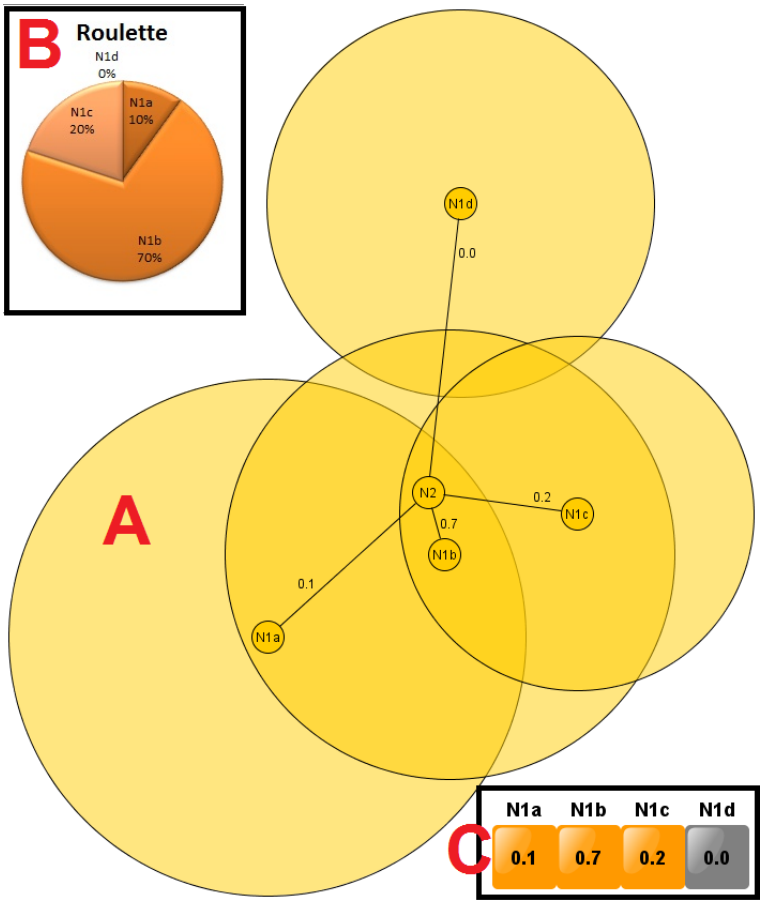


Figure 7.3: *A schema of the Roulette Wheel Selection of presynaptic neurons. Part A of the figure: The from above view on the network. N2 is a postsynaptic neuron (in the center) and N1a, N1b, N1c, N1d form a set K of possible presynaptic neurons. The probabilities of these connections are respectively: N1a: 0.1, N1b: 0.7, N1c: 0.2, N1d: 0.0 (the axonal area of N1d does not reach N2). Part B of the figure: a schema of the relevant roulette wheel with pockets corresponding to the presynaptic neurons (and sizes of the pockets corresponding to the probabilities of the connections from these neurons to N2). Part C of the figure: the resulting roulette represented as an array.*

This algorithm can be easily and effectively implemented. The sizes of pockets are stored in an array (see part C of the Figure 7.3). Then a throw of a ball into the roulette means generating one (pseudo)random number from uniform distribution within an interval $\langle 0, \mathrm{SUM})$, where SUM means the sum of all pocket sizes. Subsequently, the relevant pocket (which corresponds to the generated number) is looked up and represents the index number of the first selected candidate. This procedure is repeated S times.

Other solutions may be also possible (e.g., based on Monte-Carlo generating), but the proposed solution is both correct and effective enough, so we do not

consider necessary to present other possible solution. In addition to that, the performance efficiency is not crucial, since this algorithm is used just once during the network initialization.

7.3.3 Setting synaptic conductance delays

The synaptic conductance delay should be dependent on the length of the connection and speed of transmission. For simplicity, we will consider the same transmission speed for all connections (even though the real speed depends on many factors, such as myelination, type of connection, distinction between transmission speed on axon and dendrite, etc.). The length of the connection consists of two parts: the vertical part along axon (called d1) and horizontal part to the axon bouton (d2); see Figure 7.2. Also this scheme is very simplified. However, for a more accurate (and complicated) scheme the concrete real data would be necessary.

Let N_pre be the presynaptic neuron, N_post be the postsynaptic neuron and L be the layer, where the synapse lies. Distance d1 can be defined as Manhattan distance (also called taxicab distance, city block distance, or rectilinear distance, see (Black, 2006)) between location of N_pre and location L, which is defined as an axis of the axon of N_pre in the middle of layer L. Distance d2 can be defined as Manhattan distance between L and location of N_post.

Besides this deterministic component of the delay, we decided to add a stochastic component in the form of small (pseudo)random perturbation generated from the uniform distribution within an interval $\langle 0,$ `DELAY_PERTURBATION`$)$, where `DELAY_PERTURBATION` is a general parameter of the model. The resulting value of delay is bounded from above by the value `MAX_DELAY`, which is also a general parameter of the model.

7.3.4 Setting synaptic weights

Excitatory synapses (the synapses from excitatory neurons) have non-negative weights and inhibitory synapses (the synapses from inhibitory neurons) have non-positive weights. Although in some models, the weight values are dependent on connection length (e.g., Bart et al., 2005), according to results obtained from recordings of the mouse primary auditory cortex (Levy and Reyes, 2012), the synaptic weight does not seem to be correlated with distance between somata. We therefore do not account this factor in the initial values of synaptic weight. The resulting initial weight values were chosen as (pseudo)randomly generated from uniform distribution within an interval $\langle 0,$ `PARAM_W_EXC`$)$ for excitatory synapses and (-`PARAM_W_INH`$, 0\rangle$ for inhibitory synapses, where `PARAM_W_EXC` and `PARAM_W_INH` are general parameters of the model.

7.3.5 Connectivity algorithm

The resulting algorithm of generating synaptic connections is described in the Pseudocode 7.1. It is based on mechanisms explained in the previous subsections. The algorithm contains just the main ideas. In terms of efficiency, it is worth skipping those passages where one of these values is zero:

- `AXONAL_RADIUS` (it means that the axonal area of presynaptic neuron does not reach the layer L)
- S, i.e. number of balls (it means that no such connection should be created)
- size of the set of candidates (it means that any presynaptic neuron does not fulfil the condition of the connection)

```
1. For each postsynaptic neuron type T_post, layer L, and
   presynaptic neuron type T_pre:

   (a) S ← X1[T_post] * X2[T_post][T_pre][L] / 100

   (b) AXONAL_RADIUS ← X0[T_pre][L]

   (c) For each neuron N_post of type T_post:

        i. Create a roulette of possible presynaptic
           candidates: each presynaptic neuron N_pre of
           type T_pre (other than N_post) will have a pocket
           of size max(0, AXONAL_RADIUS -d), where d is an
           Euclidean distance from N_pre to N_post in x and y
           coordinates
       ii. SUBPROGRAM(roulette, N_post)
```

Pseudocode 7.2: *The algorithm of generating the connectome. The Roulette Wheel Selection is used to select the presynaptic neurons, which will be connected with* `N_post`*. Probability of these connections is based on the distance d (see Figure 7.2. When a roulette is created, the presynaptic neurons are selected via a process described in the SUBPROGRAM: 7.3.*

It is important to realise that the set of candidates can be small or even empty. If the set is empty, it means that no presynaptic neuron of the right type reaches the postsynaptic neuron. Then no such connections will be created. This may be considered as reasonable result. However, if the set is non-empty but small (e.g., smaller than S, the number of balls thrown into the roulette), it may lead to enforced "mega-multi-connections", such as 100 connections between one pair of neurons. The resulting network may behave in a very inappropriate way, such as groups of neurons firing every tick for a long period of time (this is based on our observations). On the other hand, neither a total prevention of multi-connection is the best solution (see the Section 7.3.6 for details).

Therefore we proposed a mechanism, which prevents creation of mega-multi-connections, but does not avoid all multi-connections. This mechanism simply limits the number of balls S in a single roulette pocket to maximal value of $|K|$, the number of candidates (nonzero pockets in the roulette). Therefore, there is always a possibility of no multi-connections, but in practise a reasonable number of multi-connections is created.

```
SUBPROGRAM(roulette, N_post):

  1. S times:

     (a) Throw a ball into roulette and call N_pre the winning
         pocket.

     (b) delay ← max( MAX_DELAY, 1+round(d1+d2/VELOCITY) +
         rand(DELAY_PERTURBATION) ), where:

          i. d1 ← distance between N_pre.location.z and z_L,
             which is z coordinates of the centre of the layer L

         ii. d2 ← Manhattan distance between locations
             (N_pre.location.x, N_pre.location.y, z_L) and N_post

     (c) If (N_pre is excitatory)

          i. weight ← PARAM_W_EXC*randomDouble {from ⟨0,1)}

     (d) Else

          i. weight ← PARAM_W_INH*randomDouble {from ⟨0,1)}

     (e) Create a synapse from N_pre to N_post with parameters
         delay, weight and zero weightDerivate.
```

Pseudocode 7.3: *The algorithm of selecting S presynaptic neurons (which will be connected with N_post), using a roulette created in the Pseudocode 7.2. When a presynaptic neuron is selected, the synaptic delay and weight are initialized. The distances d1 and d2 are depicted in Figure 7.2.*

7.3.6 Possible pitfalls and other features

All mechanisms and the resulting algorithm were based on discussion with neuroscientists and considered as reasonable simplification. However, many steps of the algorithm (detailed principle of connections generating, setting delays and weights) could be researched furthermore and in more details and compared to the real data (e.g., in future work).

According to the guidelines proposed in (Nordlie et al., 2009), we should discuss the three remaining issues (in the guidelines it means questions 5–6 in paragraph about connectivity, page 13; all other questions have been already answered in the previous subsections): self-connections, multi-connections, and boundary effects.

7.3.6.1 Self-connections: "If the same neuron can be selected as pre- and post-synaptic neuron, is this connection allowed?"

The proposed model does not allow connection from one neuron to the same neuron. This restriction was decided after consultation with neuroscientists to prevent the negative and inappropriate effects on dynamics of the whole network. Recurrence, which is an important feature in both real and artificial neural networks, is present in the form of cycles longer than one edge, i.e. loop (using the terminology from theory of graphs, see Matousek and Nesetril, 1998). However, extending to model to allow self-connections would be trivial.

7.3.6.2 Multi-connections: "If a pair of pre- and post-synaptic neurons can be chosen more than once, is this connection allowed?"

The proposed model allows such multi-connections between one pair of neurons, but prevents occurrence of "mega-multi-connections" (i.e. high number of connections between one pair of neurons). This was based on consultation with neuroscientists, because multi-connections occur in reality, but mega-multi-connections do not (personal communication with neuroscientists). However, measuring the number of connections between pairs of neurons is methodically difficult. Therefore it is difficult to decide, which distribution of multi-connections is plausible.

7.3.6.3 Boundary effects: "How are boundary effects in topological connections handled?"

The proposed model does not treat the connections on borders differently than those inside. To analyse possible resulting pitfalls, sometimes called boundary effects (but we did not find any established exact definition), we should consider, how could the network behave differently on the network boundaries. First, it is good to realize, that the number of synaptic connections leading from a neuron does not influence the behaviour of the neuron in this model. On the contrary, the number of synaptic connections leading to a neuron is important. Two situations with a neuron N_post situated on a boundary may occur:

1. First, the surroundings of the neuron N_post is sufficiently "dense", i.e., contains enough candidates of all needed presynaptic neuron types (at least

S for each type). Then the value of S will not be limited and the neuron N_post will receive the same number of synapses as any other neuron of the same neuron type inside the network. However, the probability of multi-connections to N_post may be higher than to a neuron of the same type inside the network. It may lead to higher [1] stimulation of N_post – in the case, when presynaptic neuron with multi-connection to N_post fires.

2. Second, the surroundings of the neuron N_post is rather "sparse" and does not contain enough candidates of a certain type. Then the number of synaptic connections to N_post from the particular neuron type will be lower. If the presynaptic neuron type is excitatory, then N_post may be less stimulated (and more silent for that reason). Otherwise, if this type is inhibitory, then N_post may be more stimulated (and fire more often).

To research, whether one of these situations is significantly represented, we propose two metrics:

1. The first metric computes a graph of multi-connections in dependency on distance from the border. This mean that for each neuron N_post a point $[x, y]$ is plotted on, where x is the distance of this neuron from the network border (e.g., minimum in all coordinates) and y is number of multi-connections leading to this neuron (e.g., for each presynaptic neuron N_pre $\max(0, z)$, where z is a number of synapses from N_pre to N_post). Subsequently the metric measures how descending the dependency is (e.g., by fitting it to a certain function, or simply by subjective observation) In the case that this dependency is clearly descending, i.e., the closer to border, the more multi-connections are present, then we should consider possible negative effects leading from this fact.

2. The second metric computes a similar graph, but with that difference that y is a number of absent connections. Again, if this dependency is clearly descending, i.e., the closer to border, the more connections are absent, then we should consider possible negative effects leading from this fact.

If one or both of these metrics indicates possible negative effects, it would be good to consider how to prevent these negative effects. We propose four possible solutions and describe their advantages and disadvantages:

1. In the first solution, the synaptic connections are additionally generated (instead of missing connections or superfluous multi-connections) from near (or distant) neurons, from which any connection does not lead yet. The prerequisite for this solution is sufficient number of candidates of all needed presynaptic types (otherwise distant presynaptic neurons must be used). To our knowledge, it does not reflect the reality. However it would have good results according both proposed metrics and it could prevent the origin of the described negative effects.

[1] Or lower, it depends on whether presynaptic neurons with multi-connections to N_post are rather excitatory or inhibitory.

2. In the second solution, the network is mapped on a ring (Vida et al., 2006; Tsodyks and Sejnowski, 1995), torus (Percha et al., 2005; Mehring et al., 2003), or sphere (Izhikevich et al., 2004a). Since this solution removes boundaries, any boundary effects can not happen. On the other hand, unnaturally connecting neurons which are actually very distant is absolutely implausible and could significantly influence the results of such models, where this distance is somehow important. This is the case of the AC, where tonotopy is arranged along one axis of the network. In this solution, neurons, which prefer low input frequencies, would stimulate neurons, which prefer high input frequencies, and vice versa. For this reason, we assume that this solution is not suitable for the case of the AC.

3. In the third solution, a boundary of void neurons, which are simulated like all other neurons but whose output is not considered in the analysis, is attached to the network (Worgotter and Koch, 1991). This solution can be easily implemented only by analysing the inner neurons. On the other hand, it is a rather expensive method, because it spends significant amount of time for simulating neurons, which are not considered in the analysis and therefore the resulting number of (analysed) neurons is smaller.

4. In the fourth solution, the inputs from other parts of the nervous system are modelled. We consider this solution as the best one in terms of plausibility, because it represents the real situation. However, in terms of feasibility, it is probably the most demanding solution, because it requires the knowledge of inputs from other parts of the nervous system. Should they be modelled as some kind of noise? Should they be defined according to some recorded data? What should be their exact definition? For a computer scientist, it may seem that recording such inputs (e.g., in the case of the AC) is easy and apparent. However, in reality, it is already very difficult to define the border of the AC (or primary auditory cortex). Recording the data on the border and recognizing which firings is caused by a stimulus outside the AC is even more difficult. In the case of modelling the inputs in the form of some kind of random noise, one must be very careful not to introduce by this noise any other negative effects, possibly worse than the original boundary effects.

To conclude the topic of boundary effects and solutions how to prevent these effects, we assume that the first two solutions are not sufficiently plausible for the AC model. The third solution could be used, but would decrease the number of analysed neurons (or it would increase the computational demands). The last solution would be probably the best one, if one had the required data. However, for a proper analysis of this problem, it would be very rewarding to know the connectivity on the boundaries of the real network. It is possible that the connectivity in the real network is also different on the boundaries (either it is sparser, or more multi-connections are present) and therefore the possible boundary effect in the model are a feature instead of "a bug" (incorrect behaviour).

7.4 Neurons, synapses, and channels

All neurons are single-compartmental and they are modelled by the generalized form of the Izhikevich neuron model (Izhikevich, 2007), described already in the Chapter 1. The parameters of the equations are determined by the neuron type (see the Section 7.1 for definition of the used neuron types and the Section 7.8 for the used parameters). The synaptic conduction delays are static (they do not change over time). The synaptic weights evolve according to the STDP, described in the Chapter 1. All implementation details can be found in the Chapter 3 with simulator description. We do not use any specific model of ion channels.

7.5 Model input, output, and free parameters

We use two kinds of inputs: first, the external inputs from the thalamus, and second, the internal inputs in the form of mPSCs. The latter are not actually inputs, but an internal mechanism leading to spontaneous activity. Inputs from other parts of the brain are not modelled (including them could be a possible future work).

The rest of this section includes description of inputs from Thalamus 7.5.1, mechanisms of spontaneous activity 7.5.2, and measured features of model output 7.5.3. The free parameters are described overall in the end of the chapter 7.8.

7.5.1 Inputs from thalamus

The main source of inputs to the real A1 from the lower stages of the auditory pathway are inputs from the auditory thalamus (MGV and MGD) and their major recipient is the layer L4 (Watson, 2012; Wu et al., 2011; Cruikshank et al., 2002; Kimura et al., 2003; Romanski and LeDoux, 1993; Winer and Lee, 2007; Barbour and Callaway, 2008). These inputs are topographically organized to form a gradient of frequency representation (Watson, 2012; Schreiner et al., 2000; Winer, 2005), reflecting the tonotopic organization within the auditory thalamus.

Our model reflects all these facts. The layer L4 is organized into tonotopically organized bands stimulated by external inputs (representing inputs from thalamus). The proposed model does not include other types of inputs, such as specific inputs to intensity tuned auditory neurons (observed e.g., in Greenwood and Maruyama, 1965), neurons selective for direction of frequency modulated (FM) sound sweeps (Suga, 1965; Mendelson and Cynader, 1985; Zhang et al., 2003). (These features, such as sound intensity, FM, or amplitude modulation (AM) are mainly decoded already in the lower stages of the auditory pathways.) We did not include these inputs (other than "tonotopic inputs") for two reasons. First, the model should be first observed during less complicated tasks. After tuning these basic properties, more complicated properties (such as other types of inputs) can be integrated. Second, the main experiments, which we wanted to run, did not include FM or AM and based rather on pure tones and their combinations.

In the following text, the coding of inputs used in our model is described. Although it is based on knowledge about real inputs (and all these mechanisms were

consulted with neuroscientists), the whole coding represents many simplification compared to reality.

The main physical features of sound are frequency and amplitude. Frequency corresponds to the bands (their total number is `N_BANDS`, a general parameter of the model). Pure tone means stimulating neurons in one band. The correspondence between frequency of the pure tone and the band number is linear and in the rest of the text we use just the numbers of the bands.

The amplitude of the input tone is coded as the probability of the input stimulus. The maximal amplitude (e.g., 100 dB) corresponds to probability 1, i.e., the certain input stimulus. On the other hand the minimal amplitude (e.g., 0 dB) corresponds to probability 0. The inner values are interpolated linearly. Therefore for instance a pure tone with amplitude 70 dB and frequency corresponding to band with number 25 means that in each tick (when this input is present) each neuron from the band 25 is stimulated with probability 0.7 by the input of a given value (the general parameter `INPUT_VALUE`).

In the rest of the thesis, we will use only values of bands and probabilities of the inputs, not their original units Hz and dB.

The Figure 7.4 captures several types of possible inputs: silence, noise, pure tone, more pure tones (a chord), and composed tone with "normal" distribution of intensities.

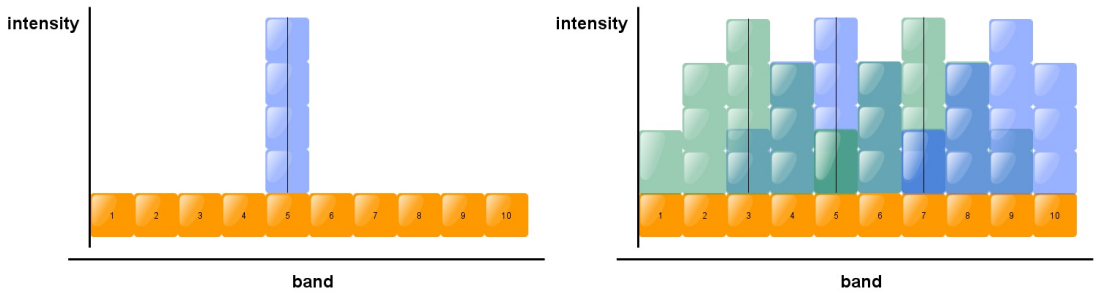


Figure 7.4: *A diagram of the representation of tonotopic inputs from the auditory thalamus. The axis x corresponds to the bands (i.e., frequencies). The axis y corresponds to the sound intensity. Left: a representation of a pure tone (only one band is stimulated). Right: representation of alterning blue and gree "normal" tones (each of them stimulates 5 bands, but the central one with the highest intensity, and the border ones with lower intensity).*

7.5.2 Spontaneous activity

We conscientiously researched possible mechanisms leading to spontaneous activity, used models of networks with spiking neurons. The results of this effort have been already described in the Chapter 3. The chosen mechanism, introducing of mPSCs (miniature Postsynaptic Currents) was used also in (Izhikevich and Edelman, 2008; Phoka et al., 2012; Muresan and Savin, 2007; Timofeev et al., 2000). The detailed implementation has been already described in the Section 3.5.5. We use the variant with one fixed pair of parameters F (`M_FREQUENCY` and A `M_AMPLITUDE`, common for the whole network, because we did not have any more detail and specific data. The parameter F means the frequency of mPSC

on each synapse, with the unit of Hz. The parameter A means the amplitude of one mPSC with the unit of pA.

We search for the values of F and A observed in real auditory cortex. The values of F were observed around 3.3 Hz and values of A around 12.7 pA (Kotak et al., 2005). Since our model scales the number of synapses per neuron by scaling factor of 20 (20 times less), we consider as reasonable to use a frequency multiplied by this scaling factor.

We assume that spontaneous activity is the ground of all experiments and therefore we decided to try more values of the parameters F and A to observe its influence on the network behaviour and to choose the reasonable combination for the rest experiments. We refer to this process as the first group of experiments called "Searching Parameters" and it is described in all details in the Chapter 8 with model results.

7.5.3 Model outputs

During experiment analysis we use the following measurements:

1. mean activity of the whole network and mean activity of the neuron types and development of this activity over the experiment
2. spike-time raster plot from some parts of the experiment (specified in the experiment); neuron types are clearly separated
3. global and local waves
4. average of excitatory weights and its development over the experiment (inhibitory weights do not develop)
5. features related to tonotopy and their development:
   (a) receptive fields of single neurons
   (b) best and characteristic frequencies of single neurons and their visualization in space (overall and distinguished according to the neuron types)
   (c) degree of local heterogeneity (overall and distinguished according to the neuron types or bands)

Implementation of the non-auditory measurements has been already described in the Section 3.6. Description of the auditory measurements follows in the next subsection.

7.5.3.1 Auditory-related measurements

We measured several features related to the organisation of tonotopy. All these features are based on the receptive fields (RF) of single neurons. The measured RF is a matrix of 13 bands $\times$ 5 intensities and contains number of spikes fired in reaction to this input. The higher number of spikes, the warmer colour in the illustration of the RF is used, see Figure 7.5.

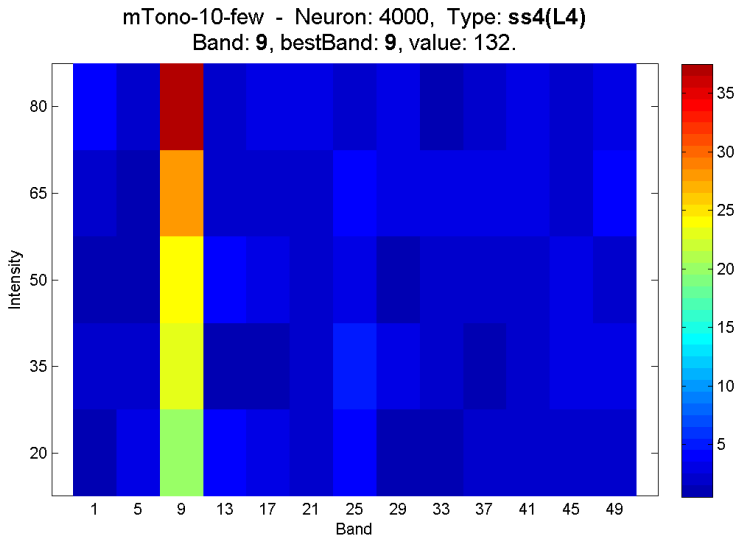


Figure 7.5: *An example of receptive field (RF) of a clearly tuned neuron from layer L4. The axis x corresponds to the bands and the axis (13 equidistant bands were tested) y corresponds to the intensities. The colour corresponds to the number of spikes fired in reaction to the input of the particular frequency and intensity (the colour scale is on the right).*

There are more possibilities, how to determine a "popular input" of a neuron, or "popular band", or "popular intensity". In the case that a neuron is tuned to one particular frequency and fires only in reaction to this frequency, then the definition of its popular input is simple. In the case of such tuning of one particular frequency, the frequency tuning curve is called "I", because of the shape of the RF resembling this letter (see Figure 7.5). However, not all neurons are tuned to a particular type of input. Therefore other definitions of "popular input" must be specified for them – or they must be specified as neurons without any popular input.

We use in this thesis two definitions of the "popular input". For simplicity, we concern only popularity of frequency, which is a common approach (besides, the intensity component did not seem so important from the results).

Best Frequency (BF). BF of a neuron is in here defined as the frequency with the highest number anywhere in the RF matrix, if this number is higher than a given threshold T_{BF}. Thanks to this threshold, generally quiet neurons can be eliminated from the subsequent measurements. However, neurons which are clearly not tuned to any frequency are not eliminated and may derange the subsequent measurements.

Characteristic Frequency (CF). CF of a neuron is in this thesis[2] defined as follows. First, for each frequency (band), a sum of spike numbers in RF over all intensities is computed. This gives us a vector of positive integers. We will call this vector SRF (sum of RF). The CF is defined as the frequency with the maximum value in the vector SRF. However, in some cases, the neuron is classified as not

[2]Other definitions may be possible.

tuned, i.e. that it does not prefer one frequency more than other frequencies.

There are more heuristics how to detect the case of tuned neuron. For example the maximum value must be somehow higher than all other values, e.g., higher than $k \cdot m$, where k is a parameter and m is an average value. However, this metric has problems with V shape of tuning curve, where RF has higher numbers in the area resembling the letter "V" and thus SRF has a shape of a hill. Such neuron is clearly tuned to the peak of the hill. Therefore it would be beneficial to recognize, whether SRF has a dominant hill. It can be recognized by the following approach. Let M be the maximal value in the SRF and T be a parameter. All positions of the SRF vector with values over $M - T$ are marked. If these positions form a connected space, we define the neuron as tuned to the peak (or mean position of all positions with the maximal SRF value), see Figure 7.6. On the contrary, if the positions form two or more separated sequences, we eliminate this neuron from the subsequent measurements, see Figure 7.7.

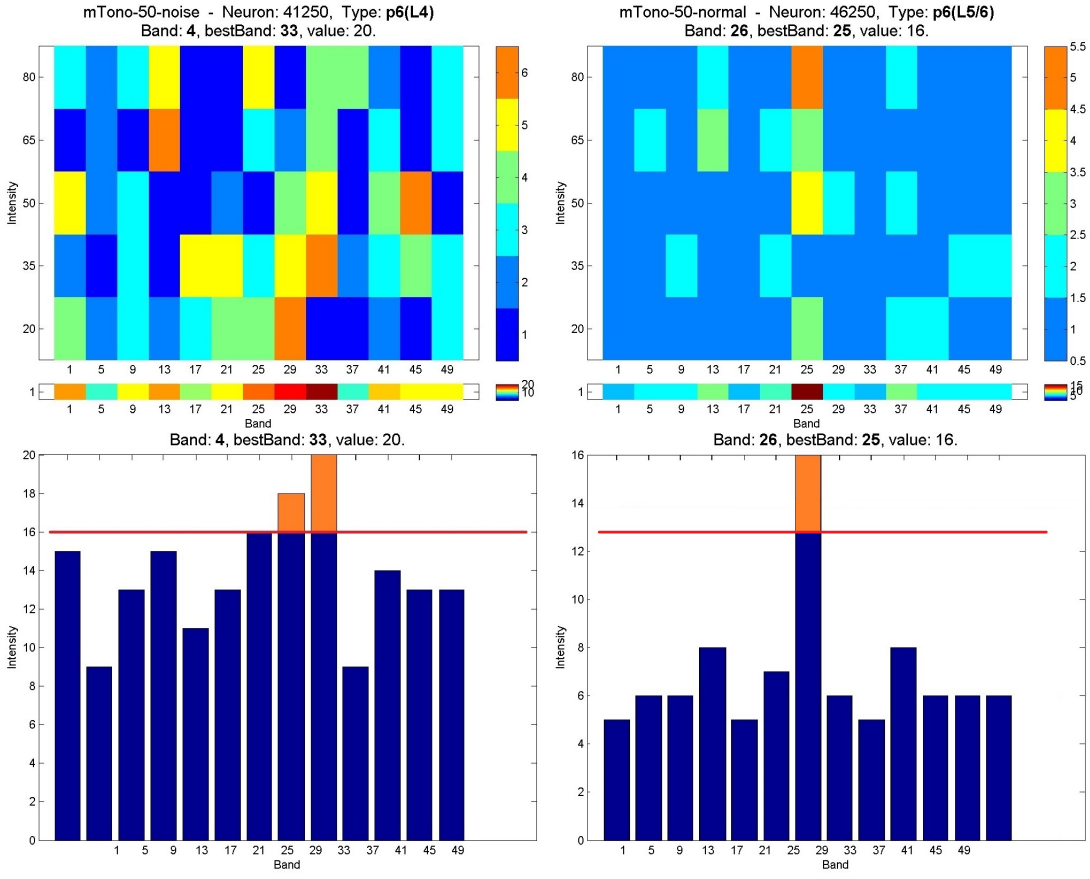


Figure 7.6: *A RF of two neurons, which were defined as tuned. Both sides (left and right) of the picture have the same parts: RF (top), SRF represented by a colour (in the middle), and SRF represented by a bar graph. Left: two bands exceed the threshold $M - T$, but they are neighbouring, therefore they form a connected space. Right: only one band exceed the threshold $T - M$ and therefore it trivially forms a connected space, to be defined as a tuned neuron.*

Tonotopic maps. When each neuron has a single popular frequency (or is eliminated from the measurements), it is possible to draw each neuron in the

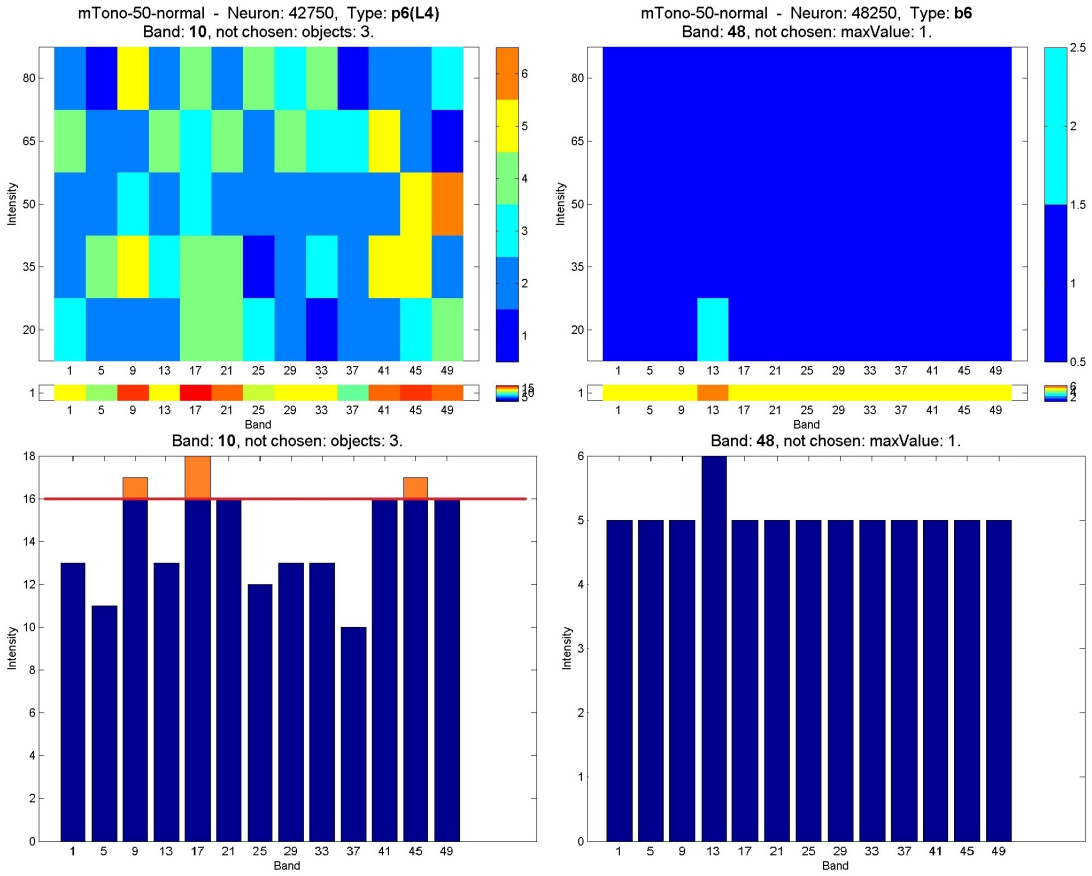


Figure 7.7: *A RF of two neurons, which were not defined as tuned. Left: more bands exceed the threshold level, but they do not form a connected space. This RF is also an example, where the band with maximal value in the SRF matrix (band 17) is different from the band with maximal value in the RF matrix (band 49). According to the used BF definition, the BF value would be 49. Right: the maximal value in the RF is too low and therefore the neuron was not defined as tuned.*

whole network (or its part) by a different colour dependent on the value of the popular frequency, independently whether it is BF, CF, or any other definition of the popular frequency. We analysed these tonotopic map from both BF and CF. An example of such map is given in Figure 7.8.

Degree of local heterogeneity. To compare the degree of local heterogeneity, we defined a simple measure as a root mean square of distances of band of the neuron from its popular band (i.e., popular frequency):

$$TM1 = \sqrt{\frac{\sum_{i=1}^{n}(B_i - P_i)}{n}}$$

where n is number of the neurons in this group (e.g., neuron type) included in the measurement, and each of these neurons has P_i the popular frequency (CF or BF) and B_i the number of the band of the column, in which the neuron lies.

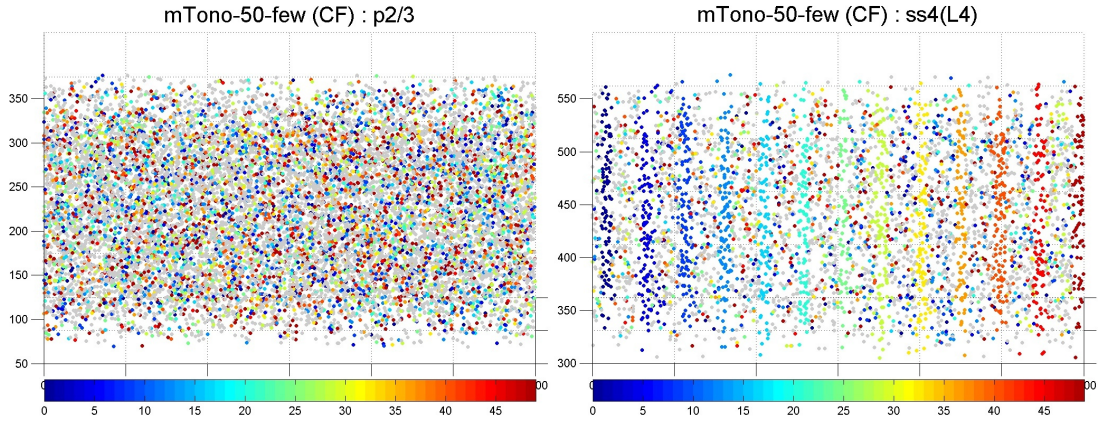


Figure 7.8: *Two examples of tonotopic maps. All neurons one neuron type was drawn in their locations (depth from the surface) and coloured by the colour corresponding to the "popular band" (here CF). The legend at the bottom shows colours of the bands. Left: neurons of type p2/3 are depicted. The degree of local heterogeneity is apparently high. Right: neurons of type ss4 are depicted. Since this layer is the input layer, it is not so surprising that the degree of tonotopy is apparently high.*

7.6 Model validation

Model validation can have several meanings. First (as it is meant in Nordlie et al., 2009), it can mean to provide information and partial results that will allow others to test and validate re-implementations of the model. There, results can be, for example, membrane potential traces of different neuron types used in the model – if the models are not well known. Second, it can mean a comparison the real network and explanation that the model behaves right (but also unexpected behaviour may be right, see (Schutter, 2010, pp. x)).

According to the first meaning, the used models of neurons and synapses are well known and their behaviour may be found e.g., in (Izhikevich, 2003, 2006).The second meaning is (in the basic form) provided in the next Chapter 8.

7.7 Model implementation

We used the SUSNOIMAC simulator tool for the simulation and additional Matlab scripts for the results analyses. All implementation details have been described in the Chapter 3.

7.8 Model parameters

In this section we list all used parameters, their description and used values. The parameters of the model are divided into 5 groups: the general model parameters, parameters of neuronal types, layers, and connectivity. In addition to the, each experiment has own setting of input parameters and simulation parameters, which have been described in the section 3.5.6.

7.8.1 General model parameters

The general model parameters are listed in the Table 7.3.

Name	Unit	Description	Used values
WIDTH	μm	Width of the network.	2000
LENGTH	μm	Length of the network.	3000
N_NEURONS	-	Total number of neurons in the network.	1000, 10000, 50000, 100000
RADIUS	μm	Radius of a typical neuron (it is used to prevent overlapping).	10
VELOCITY	μm/ms	Speed of signal transmission along synaptic connection.	100
N_BANDS	-	Number of input bands in the input layers.	50
MAX_DELAY	ms	Maximal synaptic conductance delay.	20
MAX_WEIGHT	-	Maximal weight of excitatory synapses.	100
DELAY_PERTURBATION	ms	Each initial synaptic delay is perturbed by a random number from interval $\langle 0,$ DELAY_PERTURBATION$)$.	5
PARAM_W_EXC	-	Initial weight of an excitatory synapse is random number from interval $\langle 0,$PARAM_W_EXC$)$.	100
PARAM_W_INH	-	Initial weight of an inhibitory synapse is random number from interval $($PARAM_W_INH$,0\rangle$.	-50

Table 7.3: *The* ***general*** *parameters of the model.*

The general model parameters are set in the file `modelX.properties`, where `X` means the number of the experiment (starting with value 1). The exact names used in this file are slightly different than those in the Table 7.3, but their names can be easily found in the example files. We chose use in the text of the thesis different names to have a standardized form of their format and we did not change the format used in properties file, because it would be necessary to change all files of already finished experiments.

7.8.2 Layer parameters

The used layers and their parameters are listed in the Table 7.4. These values are set in the `table3.csv` file in the CSV format with semicolon as delimiter.

Layer	Thickness	Input
L1	69	False
L2/3	235	False
L4	208	True
L5	248	False
L6	451	False

Table 7.4: *Parameters of the model related to* ***layers****.*

7.8.3 Neuron types parameters

The used neuron types and their parameters are listed in the Table 7.5. The used axonal radii of neuron types in all layers (table called **X0** in the text) are listed in the Table 7.6. All these values are set in the `table1.csv` file in the CSV format with semicolon as delimiter.

Label	T	L	Cells	C	k	vr	vt	vp	a	b	c	d	exc.
nb1	nb	L1	1.512	20	0.3	-66	-40	30	0.17	5	-45	100	false
p2/3	p	L2/3	26.21	100	3	-60	-50	50	0.01	5	-60	400	true
b2/3	b	L2/3	3.125	20	1	-55	-40	25	0.15	8	-55	200	false
nb2/3	nb	L2/3	4.234	100	1	-56	-42	40	0.03	8	-50	20	false
$ss4_{(L4)}$	ss	L4	9.274	100	3	-60	-50	50	0.01	5	-60	400	true
$ss4_{(L2/3)}$	ss	L4	9.274	100	3	-60	-50	50	0.01	5	-60	400	true
p4	p	L4	9.274	100	3	-60	-50	50	0.01	5	-60	400	true
b4	b	L4	5.444	20	1	-55	-40	25	0.15	8	-55	200	false
nb4	nb	L4	1.512	100	1	-56	-42	40	0.03	8	-50	20	false
$p5_{(L2/3)}$	p	L5	4.839	100	3	-60	-50	50	0.01	5	-60	400	true
$p5_{(L5/6)}$	p	L5	1.31	100	3	-60	-50	50	0.01	5	-60	400	true
b5	b	L5	0.605	20	1	-55	-40	25	0.15	8	-55	200	false
nb5	nb	L5	0.806	100	1	-56	-42	40	0.03	8	-50	20	false
$p6_{(L4)}$	p	L6	13.71	100	3	-60	-50	50	0.01	5	-60	400	true
$p6_{(L5/6)}$	p	L6	4.839	100	3	-60	-50	50	0.01	5	-60	400	true
b6	b	L6	2.016	20	1	-55	-40	25	0.15	8	-55	200	false
nb6	nb	L6	2.016	100	1	-56	-42	40	0.03	8	-50	20	false

Table 7.5: *Parameters of the model related to* ***neuron types****.*

7.8.4 Connectivity parameters

The used parameters for the connectivity are listed in the Table 7.9 (in text this table is called as **X1** and **X2**). All these values are set in the `table2.csv` file in the CSV format with semicolon as delimiter.

7.8.5 Input parameters

The input parameters are listed in the Table 7.7. The exact used values are listed in the descriptions of experiments, in the Chapter 8. These parameters are set in the file `inputsX.properties`, where `X` means the number of the experiment (starting with value 1).

Label	L1	L2/3	L4	L5	L6
nb1	200	200	200	200	200
p2/3	550	1120	150	1000	150
b2/3	0	150	150	150	0
nb2/3	200	200	200	200	200
$ss4_{(L4)}$	0	300	1120	400	150
$ss4_{(L2/3)}$	150	400	500	150	150
p4	150	1120	150	550	150
b4	0	0	500	0	0
nb4	200	200	200	200	200
$p5_{(L2/3)}$	150	400	300	500	250
$p5_{(L5/6)}$	0	0	150	500	1000
b5	0	0	0	500	0
nb5	200	200	200	200	200
$p6_{(L4)}$	0	0	150	500	1000
$p6_{(L5/6)}$	0	150	1000	150	150
b6	0	0	0	0	500
nb6	200	200	200	200	200

Table 7.6: *Parameters of the model related to* ***axonal areas*** *(axonal radii in layers).*

Name	Unit	Description	Used values
`INPUT_TYPE`	-	Type of the inputs (0 $\sim$ nothing, 1 $\sim$ minis, 2 $\sim$ noise, 3 $\sim$ few pure tones, 4 $\sim$ more pure tones, 5 $\sim$ more normal sounds, 6 $\sim$ tonotopy experiment)	0, 1, 2, 3, 4, 5, 6
`INPUT_VALUE`	pA	Value of the input current.	1500
`INPUT_DURATION`	ms	Duration of one input.	500
`BANDS`	-	List of used bands separated by semicolons.	different values
`INTENSITIES`	-	List of used intensities expressed as probabilities in the range $\langle 0, 1 \rangle$ separated by semicolons.	different values

Table 7.7: *Input parameters of an experiment.*

Label	Layer	Syn	nb1	p2/3	b2/3	nb2/3	ss4(L4)	ss4(L2/3)	p4	b4	nb4	p5(L2/3)	p5(L5/6)	b5	nb5	p6(L4)	p6(L5/6)	b6	nb6
nb1	L1	445	10.1	6.3	0.6	1.1	x	x	0.1	x	x	0.1	x	x	x	x	x	x	x
p2/3	L2/3	290	x	59.9	9.1	4.4	0.6	6.9	7.7	x	0.8	7.4	x	x	x	2.3	x	x	0.8
p2/3	L1	65	10.2	6.3	0.1	1.1	x	x	0.1	x	x	0.1	x	x	x	x	x	x	x
b2/3	L2/3	193	1.3	51.6	10.6	3.4	0.5	5.8	6.6	x	0.8	6.3	x	x	x	2.1	x	x	0.7
nb2/3	L2/3	165	1.7	48.6	11.4	3.3	0.5	5.5	6.2	x	0.8	5.9	x	x	x	1.8	x	x	0.6
ss4(L4)	L4	290	x	2.7	0.2	0.6	11.9	3.7	4.1	7.1	2	0.8	0.1	x	x	32.7	x	x	5.8
ss4(L2/3)	L4	249	x	5.6	0.4	0.8	11.3	3.8	4.3	7.2	2.1	1.1	0.1	x	x	31.1	x	x	5.5
p4	L4	252	x	4.3	0.2	0.6	11.5	3.6	4.2	7.2	2.1	1.2	0.1	x	x	31.4	0.1	x	5.9
p4	L2/3	43	x	63.1	5.1	4.1	0.6	7.2	8.1	x	0.6	7.8	x	x	x	2.5	x	x	0.8
p4	L1	40	10.2	6.3	0.1	1.1	x	x	0.1	x	x	0.1	x	x	x	x	x	x	x
b4	L4	162	x	5.8	0.5	0.8	11	3.8	4.2	8.4	2.4	1.1	x	x	x	30.3	x	x	5.4
nb4	L4	184	x	2.7	0.2	0.6	11.7	3.6	4	8.2	2.3	0.8	0.1	x	x	32.2	x	x	5.7
p5(L2/3)	L5	216	x	45.9	1.8	0.3	3.3	2	7.5	x	0.9	11.7	1	0.8	1.1	2.3	2.1	x	11.5
p5(L2/3)	L4	14	x	2.8	0.1	0.7	12.2	3.8	4.2	5.2	1.5	0.8	0.1	x	x	33.7	x	x	5.9
p5(L2/3)	L2/3	21	x	63.1	5.1	4.1	0.6	7.2	8.1	x	0.6	7.8	x	x	x	2.5	x	x	0.8
p5(L2/3)	L1	9	10.2	6.3	0.1	1.1	x	x	0.1	x	x	0.1	x	x	x	x	x	x	x
p5(L5/6)	L5	255	x	44.3	1.7	0.2	3.2	2	7.3	x	0.8	11.3	1.2	0.8	1.1	2.3	2.5	0.3	11.3
p5(L5/6)	L4	47	x	2.8	0.1	0.7	12.2	3.8	4.2	5.2	1.5	0.8	0.1	x	x	33.7	x	x	5.9
p5(L5/6)	L2/3	68	x	63.1	5.1	4.1	0.6	7.2	8.1	x	0.6	7.8	x	x	x	2.5	x	x	0.8
p5(L5/6)	L1	283	10.2	6.3	0.1	1.1	x	x	0.1	x	x	0.1	x	x	x	x	x	x	x
b5	L5	149	x	45.5	2.3	0.2	3.3	2	7.5	x	1.1	11.6	1	0.9	1.3	2.3	2	x	11.4
nb5	L5	149	x	45.5	2.3	0.2	3.3	2	7.5	x	1.1	11.6	1	0.9	1.3	2.3	2	x	11.4
p6(L4)	L6	163	x	2.5	0.1	0.1	0.7	0.9	1.3	x	0.1	0.1	4.9	x	0.3	1.2	13.2	7.7	7.7
p6(L4)	L5	53	x	46.8	0.8	0.3	3.4	2.1	7.7	x	0.6	11.9	1	0.6	0.8	2.3	2.1	x	11.7
p6(L4)	L4	96	x	2.8	0.1	0.7	12.2	3.8	4.2	5.2	1.5	0.8	0.1	x	x	33.7	x	x	5.9
p6(L4)	L2/3	6	x	63.1	5.1	4.1	0.6	7.2	8.1	x	0.6	7.8	x	x	x	2.5	x	x	0.8
p6(L5/6)	L6	279	x	2.5	0.1	0.1	0.7	0.9	1.3	x	0.1	0.1	4.9	x	0.3	1.2	13.2	7.8	7.8
p6(L5/6)	L5	13	x	46.8	0.8	0.3	3.4	2.1	7.7	x	0.6	11.9	1	0.6	0.8	2.3	2.1	x	11.7
p6(L5/6)	L4	12	x	2.8	0.1	0.7	12.2	3.8	4.2	5.2	1.5	0.8	0.1	x	x	33.7	x	x	5.9
p6(L5/6)	L2/3	14	x	63.1	5.1	4.1	0.6	7.2	8.1	x	0.6	7.8	x	x	x	2.5	x	x	0.8
p6(L5/6)	L1	3	10.2	6.3	0.1	1.1	x	x	0.1	x	x	0.1	x	x	x	x	x	x	x
b6	L6	161	x	2.5	0.1	0.1	0.7	0.9	1.3	x	0.1	0.1	4.9	x	0.4	1.2	13.2	7.7	7.7
nb6	L6	161	x	2.5	0.1	0.1	0.7	0.9	1.3	x	0.1	0.1	4.9	x	0.4	1.2	13.2	7.7	7.7

Figure 7.9: *Parameters of the model defining the connectome.*

7.9 Tabular description

Finally, we provide an overall summarized description of the network in tables, as it is recommended in guidelines by Nordlie (Nordlie et al., 2009).

A	Model Summary
Populations	Seventeen: nb1, p2/3, b2/3, nb2/3, $ss4_{(L4)}$, $ss4_{(L2/3)}$, p4, b4, nb4, $p5_{(L2/3)}$, $p5_{(L5/6)}$, b5, nb5, $p6_{(L4)}$, $p6_{(L5/6)}$, b6, nb6
Topology	3D coordinates within a network
Connectivity	Generated according probabilities of connections between all pairs of neuron types
Neuron model	Generalized Izhikevich neuron model (Izhikevich, 2007)
Channels models	-
Synapse model	Fixed synaptic conduction delays and changeable synaptic weights
Plasticity	STDP
Input	Tonotopic inputs from thalamus
Measurements	Spike trains (and from them spike-time raster plot and global and local waves); development of: mean activity (within a network and individual neuron types), weights and their distribution, tonotopic features (receptive fields, best frequencies and measure of tonotopy)

Table 7.8: *The summarized tabular description of the designed model.*

8. Model of the Auditory Cortex: Results

This chapter describes experiments run on the model, their results and discussion. The experiments were divided into three groups:

1. **Parameter Space Search**. The goal of these experiments was to try more values of certain parameters of the model, to observe and describe their influence on the network behaviour, and choose a reasonable combination of parameters for the other experiments.

2. **Features of the Chosen Parameters**. The goal of these experiments was to observe the chosen combination of parameters in more detail: in longer simulation and using different sizes of the network.

3. **Tonotopy Experiments**. The goal of these experiments was to observe the model development during different types of inputs and research the resulting influence on tonotopic organization of the network. These experiments were designed to test the hypothesis described in the Section 6.1.1.

The first two groups of experiments concern just the spontaneous activity (no external input), whereas the third group concerns also different types of sound stimuli (such as noise, or pure tones).

8.1 Parameter Space Search

This first group of experiments was designed to search at least a part of the parameter space. We had two main reasons for this: first, to observe and describe influence of different settings of these parameters on the network behaviour, and second to choose a reasonable combination for the subsequent experiments.

8.1.1 Description of the experiments

Since the space of all parameters is very large, it is not possible to test all the possible values of all parameters and it would be even more impossible to analyse the results. Therefore we had to choose just a limited parameter space and other parameters set fixed. We considered two types of parameters to be important to have tested: parameters of spontaneous activity, i.e., mPSC, and parameters of the long-term plasticity, i.e., STDP, because both these mechanisms have fundamental influence on the network activity. Other parameters were set fixed. The values of `WEIGHT_INCREASE`, `WEIGHT_DER_MULT`, `PARAM_LTP`, and `STDP_DER_MULT` were adopted from (Izhikevich, 2006) and the parameters `PARAM_W_EXC` and `PARAM_W_INH` were adjusted to the generalized form of the Izhikevich neuron model. The researched parameters are listed in the Table 8.1.

`PARAM_LTD` has only two tested values, because we wanted to research the influence of the ratio between `PARAM_LTP` and `PARAM_LTD`, which was in (Izhikevich, 2006) used as 1.0:1.2 and in (Izhikevich2008) used as 1:2.

no.	N	T	F	A	LTD	INPUTS	LOG_S	LOG_D
1	10,000	2,000 s	15	13	1.2	minis	1,995 s	5 s
2	10,000	2,000 s	15	13	2.0	minis	1,995 s	5 s
3	10,000	2,000 s	15	20	1.2	minis	1,995 s	5 s
4	10,000	2,000 s	15	20	2.0	minis	1,995 s	5 s
5	10,000	2,000 s	15	27	1.2	minis	1,995 s	5 s
6	10,000	2,000 s	15	27	2.0	minis	1,995 s	5 s
7	10,000	2,000 s	30	13	1.2	minis	1,995 s	5 s
8	10,000	2,000 s	30	13	2.0	minis	1,995 s	5 s
9	10,000	2,000 s	30	20	1.2	minis	1,995 s	5 s
10	10,000	2,000 s	30	20	2.0	minis	1,995 s	5 s
11	10,000	2,000 s	30	27	1.2	minis	1,995 s	5 s
12	10,000	2,000 s	30	27	2.0	minis	1,995 s	5 s
13	10,000	2,000 s	45	13	1.2	minis	1,995 s	5 s
14	10,000	2,000 s	45	13	2.0	minis	1,995 s	5 s
15	10,000	2,000 s	45	20	1.2	minis	1,995 s	5 s
16	10,000	2,000 s	45	20	2.0	minis	1,995 s	5 s
17	10,000	2,000 s	45	27	1.2	minis	1,995 s	5 s
18	10,000	2,000 s	45	27	2.0	minis	1,995 s	5 s
19	10,000	2,000 s	60	13	1.2	minis	1,995 s	5 s
20	10,000	2,000 s	60	13	2.0	minis	1,995 s	5 s
21	10,000	2,000 s	60	20	1.2	minis	1,995 s	5 s
22	10,000	2,000 s	60	20	2.0	minis	1,995 s	5 s
23	10,000	2,000 s	60	27	1.2	minis	1,995 s	5 s
24	10,000	2,000 s	60	27	2.0	minis	1,995 s	5 s

Table 8.1: *The settings of 14 experiments from the first group of experiments:* ***Parameter Space Search****. The meaning of the columns is following: N is number of neurons, T is duration of the experiment (in s of model time), F and A are values of the parameters* `M_FREQUENCY` *and* `M_AMPLITUDE` *of mPSCs, LTD is value of the parameter* `PARAM_LTD`*, INPUTS is type of inputs (here only spontaneous activity), LOG_S and LOG_D are values of parameters* `LOG_START` *and* `LOG_DURATION`*, which here means that we saved logs and analysed exactly the last 5 s of each experiment.*

Before choosing the tested values for F (parameter `M_FREQUENCY`) and A (parameter `M_AMPLITUDE`), we did a preliminary research. Their values in the real AC fluctuate around F=3.3 ± 0.2 Hz and A=12.7 ±1.4 pA (based on values about mEPSC from the AC (Kotak et al., 2005)). Initially, we tested the same values in the model. However, these values were not sufficient for any activity. It is relatively logical, because in our model each neuron has 20 times less incoming synapses than in the real AC. Therefore we considered reasonable to multiply the real value of parameter F by the scaling factor (5–20).

To sum it up, we tested $4 \times 3 \times 2 = 24$ variants, each of them on the network of 10,000 neurons, with no external stimulation, only the spontaneous activity (minis). The relevant settings of all experiments of **Parameter Space Search** group are listed in the Table 8.1. Parameters of the model, which are not listed in the table, are same as in all other experiments and can be found in the Chapter

7. We run each experiment for 2,000 s of model time and analysed spike trains from the last 5 s of the simulation.

All the experiments described in this chapter were run once (because of their number and duration). However, many preliminary experiments preceded the final experiments (with very similar results), and the total number of performed experiments is relatively high (50), so the repetitive running should not change the results significantly. Regardless, it could be meaningful future work.

8.1.2 Results of the experiments

Experiments with values 15 Hz of the parameter F did not lead to any activity (in the case of A=13 pA), or almost any activity (A=20 pA), or very low activity (A=27 pA). Therefore, they are included neither in the following description of results, nor in the analysis of the results.

Because the parameter space has three dimensions and we analyse many features, the following text is divided into two main subsections. First, the overall description will be provided and the main results will be listed in a tabular format. Second, the results of the individual features will be described, grouped the parameters.

8.1.2.1 Overall description of the results

The synoptic results are listed in the Table 8.1. Values of the mean firing rate (mean activity) range from 0.5 to 10 Hz. The most active neuron types are b2/3 (up to 80 Hz) and b5 (up to 70 Hz) and the second most active neuron types are p2/3 and p5. On the contrary, the neuron types nb1 and nb6 fire only exceptionally.

Global waves are present only in some experiments and then only with low frequencies (up to 11 Hz). Local waves are markedly dependent on parameter values and location in the network. Generally frequencies of all types are present, however the highest amplitude have again only the waves with low frequencies.

The development of the mean of the excitatory weights is dependent of the values of F and A. The very low values (F up to 30 Hz) led to sudden drops. However, in the majority of the experiments (with higher values of F or A), the weights became stable (after 20–2,000 s) and in some cases slightly fluctuated, in other cases remained stable without fluctuations. The resulting values of the mean of the excitatory weights ranged from 20 to 60.

The duration of the experiment ranged between 150 to 190 min of the real time.

	F	A	LTD	FR	GLOBAL	LOCAL	WEIGHTS	FLUCT	MW	T
7	30	13	1.2	1.8	delta, theta	delta, theta, gamma	descent	tiny	22	157
8	30	13	2.0	0.5	-	delta, gamma	increase → stable	no	57	159
9	30	20	1.2	2.5	delta	delta, alfa, beta, gamma	stable	small	25	163
10	30	20	2.0	1.5	-	delta, alfa, beta, gamma	increase → stable	no	45	164
11	30	27	1.2	4.3	-	**theta, gamma**	stable	small	25	168
12	30	27	2.0	2.6	-	delta, alfa, gamma	increase → stable	no	33	169
13	45	13	1.2	2.0	**delta, alfa**	**gamma**	stable	small	23	172
14	45	13	2.0	1.3	-	-	increase → stable	no	48	171
15	45	20	1.2	4.8	-	theta	stable	small	23	174
16	45	20	2.0	2.9	-	delta, gamma	increase → stable	no	30	171
17	45	27	1.2	7.0	**alfa**	**alfa, beta**	stable	tiny	21	182
18	45	27	2.0	5.2	-	-	stable	no	11	176
19	60	13	1.2	4.8	**theta, alfa**	**gamma**	stable	small	24	181
20	60	13	2.0	2.0	-	alfa	increase → stable	no	38	178
21	60	20	1.2	6.5	-	alfa, beta	stable	tiny	31	186
22	60	20	2.0	4.9	-	-	stable	no	14	175
23	60	27	1.2	9.0	alfa	theta, alfa	stable	tiny	20	189
24	60	27	2.0	7.2	-	gamma	stable	no	4	186

Figure 8.1: *The results of the first group of experiments:* ***Parameter Space Search****. Rows correspond to the experiments (experiments with F=15 Hz were not included for too low activity). The first four columns describes main setting of the experiments. The other columns summarize the main measured features. FR: mean firing rate of all neurons. GLOBAL: global oscillations (rarely, medium often, and* ***often****). LOCAL: local oscillations. WEIGHTS: main characteristic of the development of the mean excitatory weights. FLUCT: presence of fluctuations in the mean excitatory weights. MW: mean value of the excitatory weight in the end of the experiment. T: duration of execution of the experiment in s (real time).*

8.1.2.2 Description of the results grouped by features and parameters

The network activity. As can be seen from Figures 8.2, 8.3, 8.4, and 8.5, each of the tested LTD values led to markedly different behaviour.

- The lower (LTD=1.2) value led to much clearer synchronization (see Figure 8.2 versus Figure 8.3). On the other hand, the higher (LTD=2) value led to much narrower amplitude of fluctuations in activity: in all neuron types, but most significantly in basket neurons in layers L2/3 and L5.
- In addition to that, the mean activity of these basket neurons was higher with the lower LTD value.
- Moreover, as can be seen from Figure 8.2, the lower LTD value led to noticeable activity even in the less active neuron types (e.g., ss4, p4, nb4), whereas in the case of the higher LTD value, these neuron types fired more rarely and in solitude.
- The last significant distinction between LTD values is the course of the activity of b2/3 and b5. While in the case of lower LTD (see 8.4), it has

first a rather downward trend before it gets relatively stable (but with high amplitude of fluctuations), in the case of higher LTD (see 8.3), the trend is upward before it gets stable (here the amplitude of fluctuations is much lower).

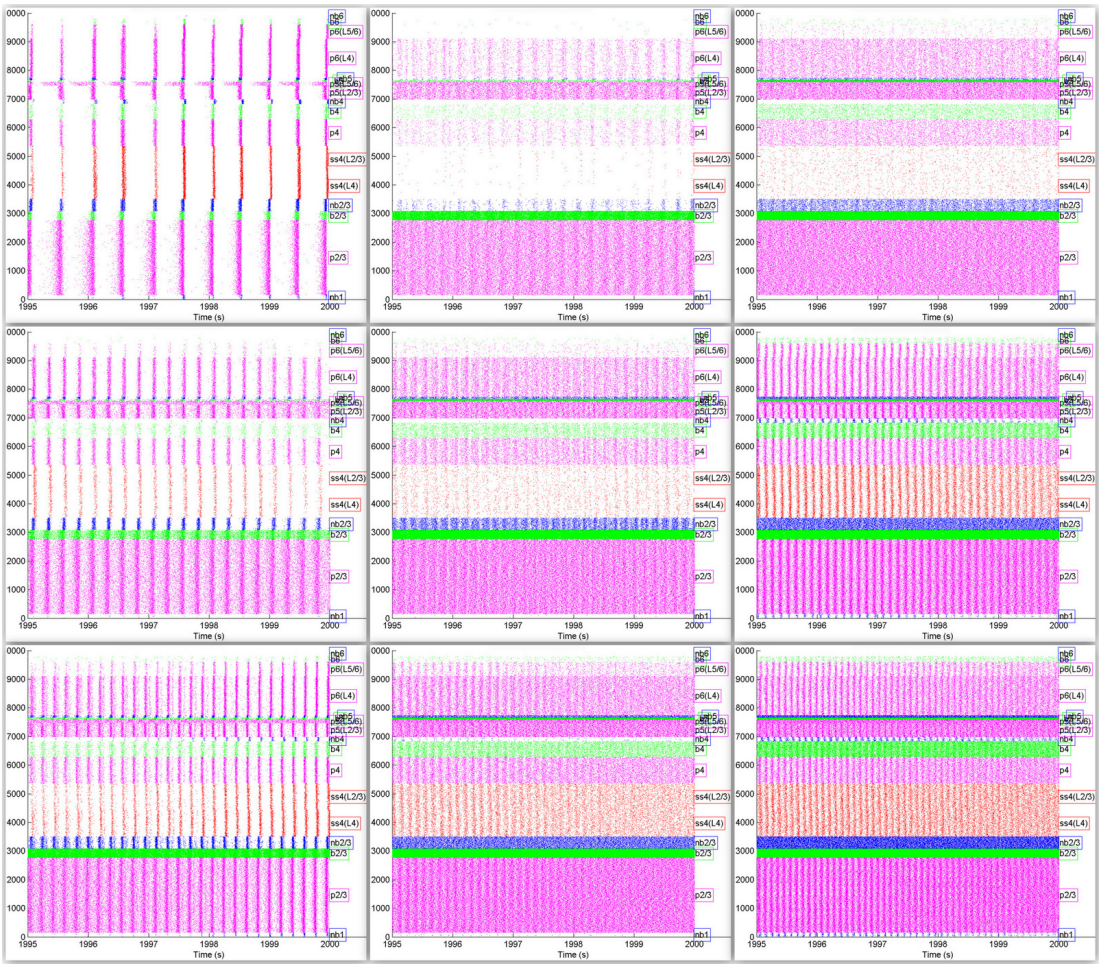


Figure 8.2: *STRP of experiments with LTD=1.2. The first row corresponds to F=30 Hz, the second row to F=45 Hz, the third row to F=60 Hz. The first column corresponds to A=13 pA, the second column to A=20 pA, the third column to A=27 pA.*

The influence of F and A is very similar and can be relatively clearly described.

- The overall (and also mean) activity of the network is approximately directly proportional to the value of F and A.
- The higher values of F and A lead to smaller fluctuations of all neuron types, see Figures 8.4, and 8.5.
- With growing F and A, the values of b5 activity are closer to the values of b2/3 activity, see Figures 8.4, and 8.5.

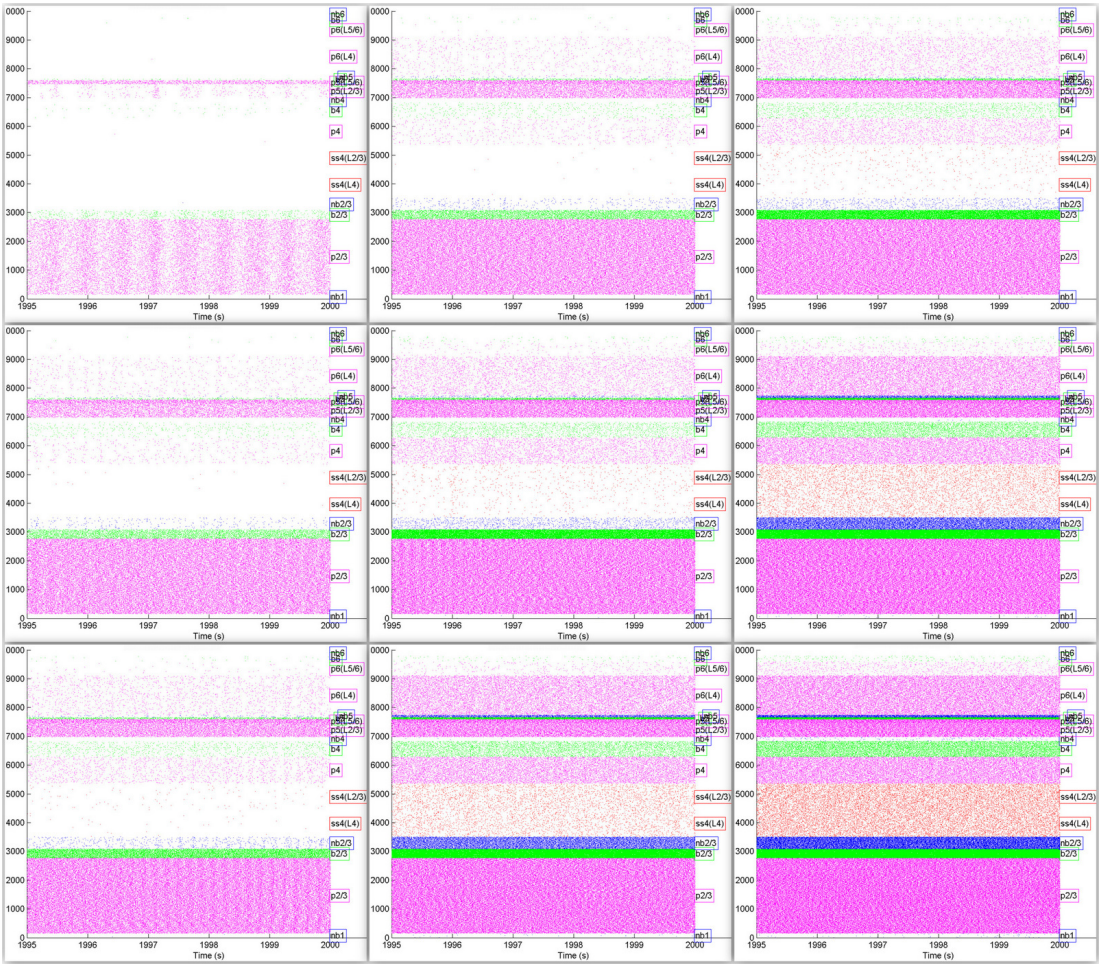


Figure 8.3: *STRP of experiments with LTD=2.0. The first row corresponds to F=30 Hz, the second row to F=45 Hz, the third row to F=60 Hz. The first column corresponds to A=13 pA, the second column to A=20 pA, the third column to A=27 pA.*

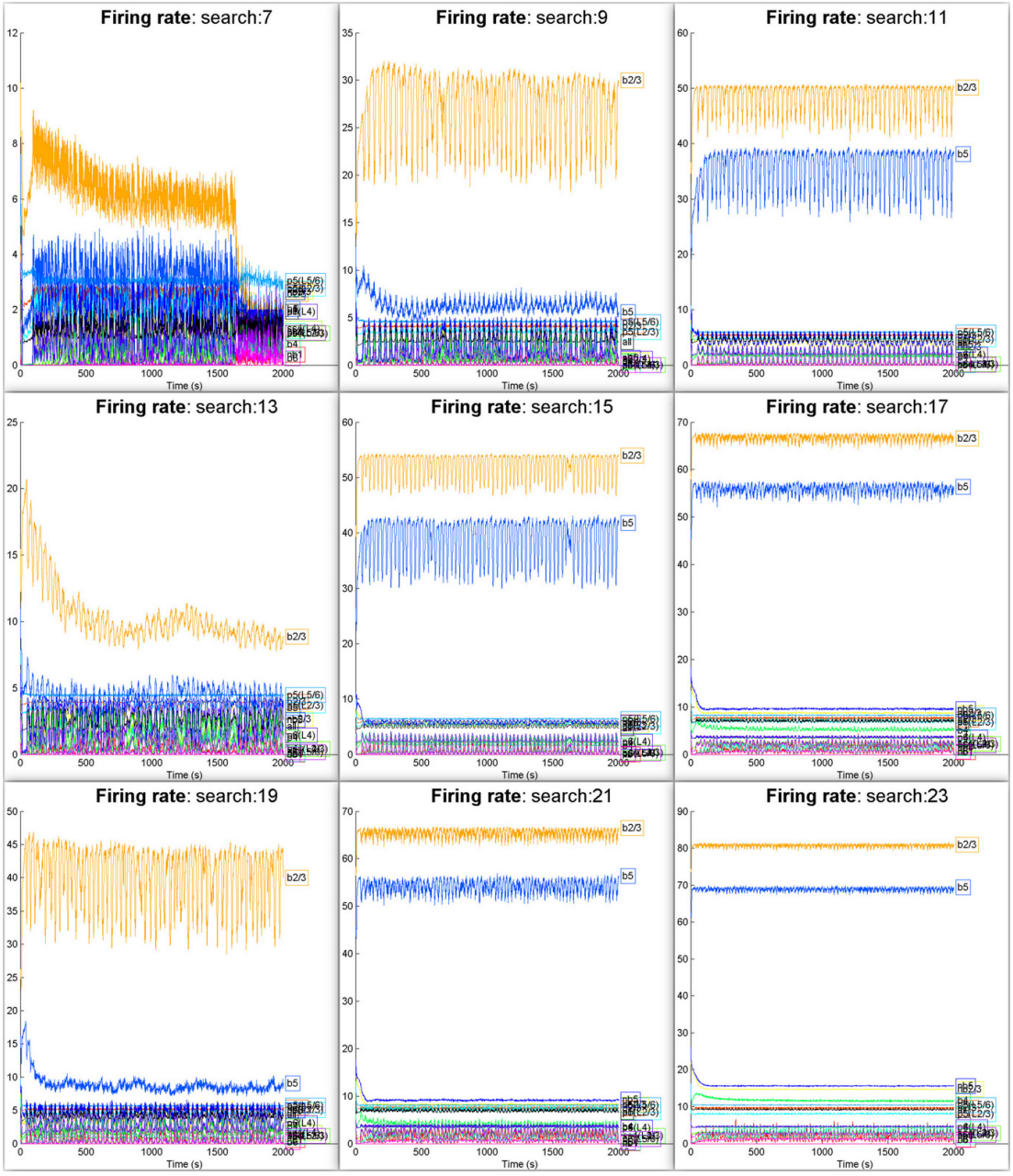


Figure 8.4: *Firing rate of experiments with LTD=1.2. The first row corresponds to F=30 Hz, the second row to F=45 Hz, the third row to F=60 Hz. The first column corresponds to A=13 pA, the second column to A=20 pA, the third column to A=27 pA.*

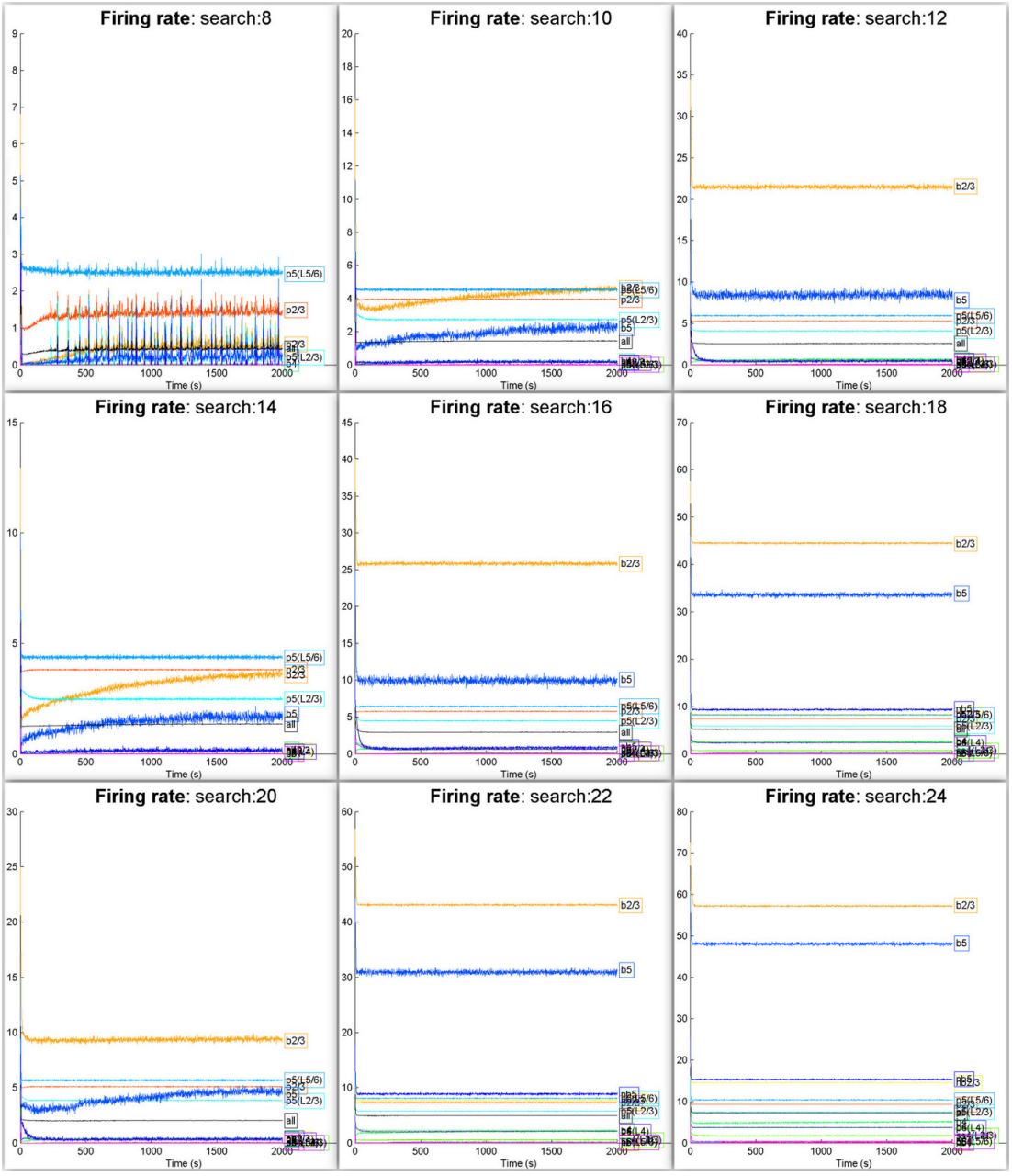


Figure 8.5: *Firing rate of experiments with LTD=2.0. The first row corresponds to F=30 Hz, the second row to F=45 Hz, the third row to F=60 Hz. The first column corresponds to A=13 pA, the second column to A=20 pA, the third column to A=27 pA.*

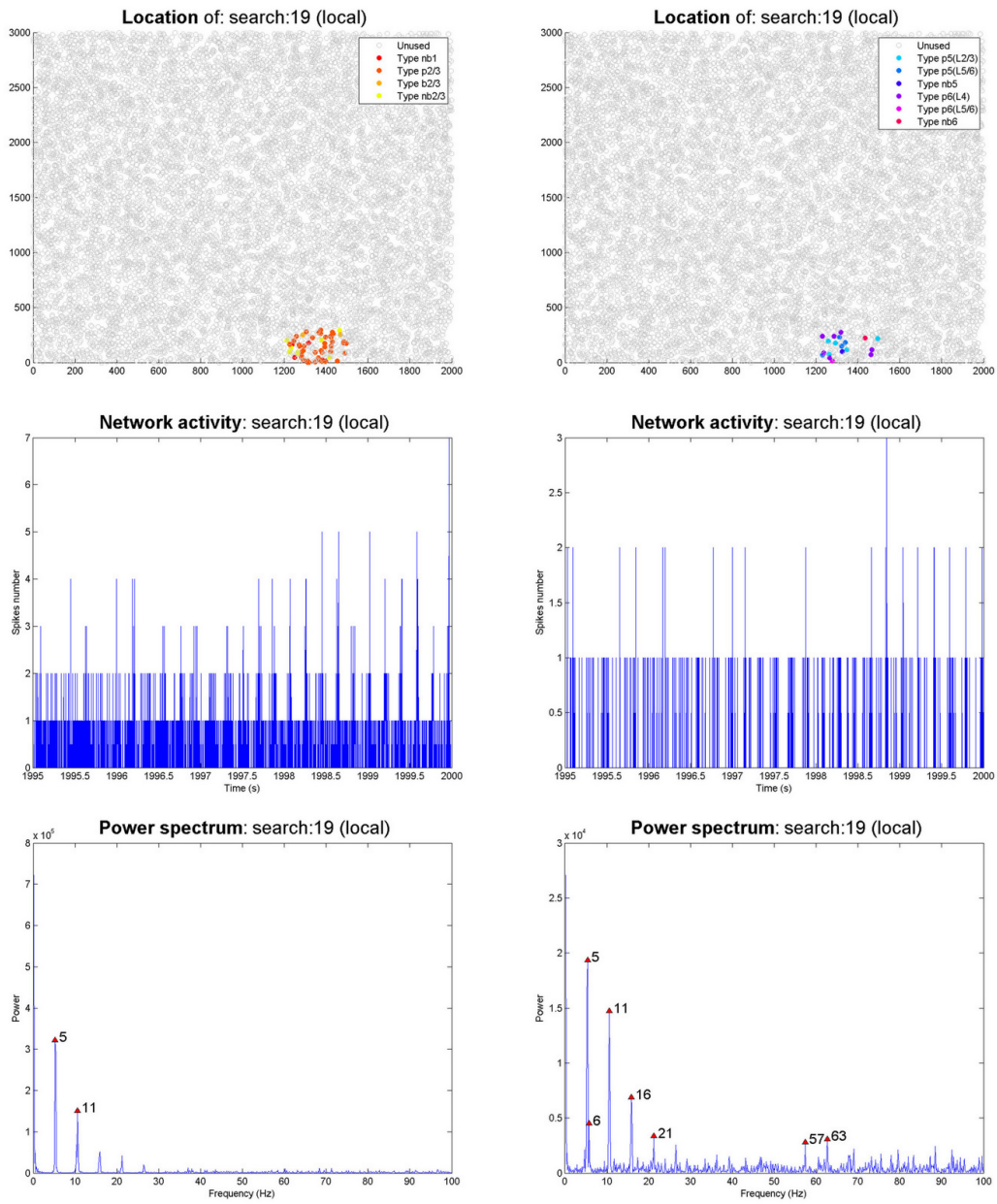


Figure 8.6: *First group of experiments, examples of local waves (from experiment no. 19). The first row depicts locations of the measured local areas. The second row depicts number of spikes over time, in this local area. The third row depicts the present oscillations. Left: an area from layers L1–L3, where only low-frequency waves were present. Right: an area from layers L4–L6, where a broader spectrum of waves was present.*

The oscillation spectrum. Each of the tested LTD values led to significantly different behaviour.

- The higher value of LTD led to totally suppressed emergence of global waves. On the contrary, in the case of the lower LTD, the synchronized activity can be observed even in spike times raster plot Figure 8.2 and in some experiments, the global waves were confirmed also by the power spectrum analysis.

- Likewise the local waves occurred in the case of the higher LTD only exceptionally and with small power, whereas in the case of the lower LTD, local waves occurred much more frequently and in richer spectrum. Interestingly, the spectrum was dependent on the position within the network. For example on Figure 8.6, the location in the range of layers L1–L3 shows only alpha and theta waves, whereas the location in the similar position, but in the range of layers L4–L6 during the same time shows also beta and gamma waves.

The influence of values F and A on oscillations is not so clear. In the case of lowest value of A (13 pA), the global waves were present in every experiment. However in the case of highest value of A (27 pA), the global waves were also sometimes present.

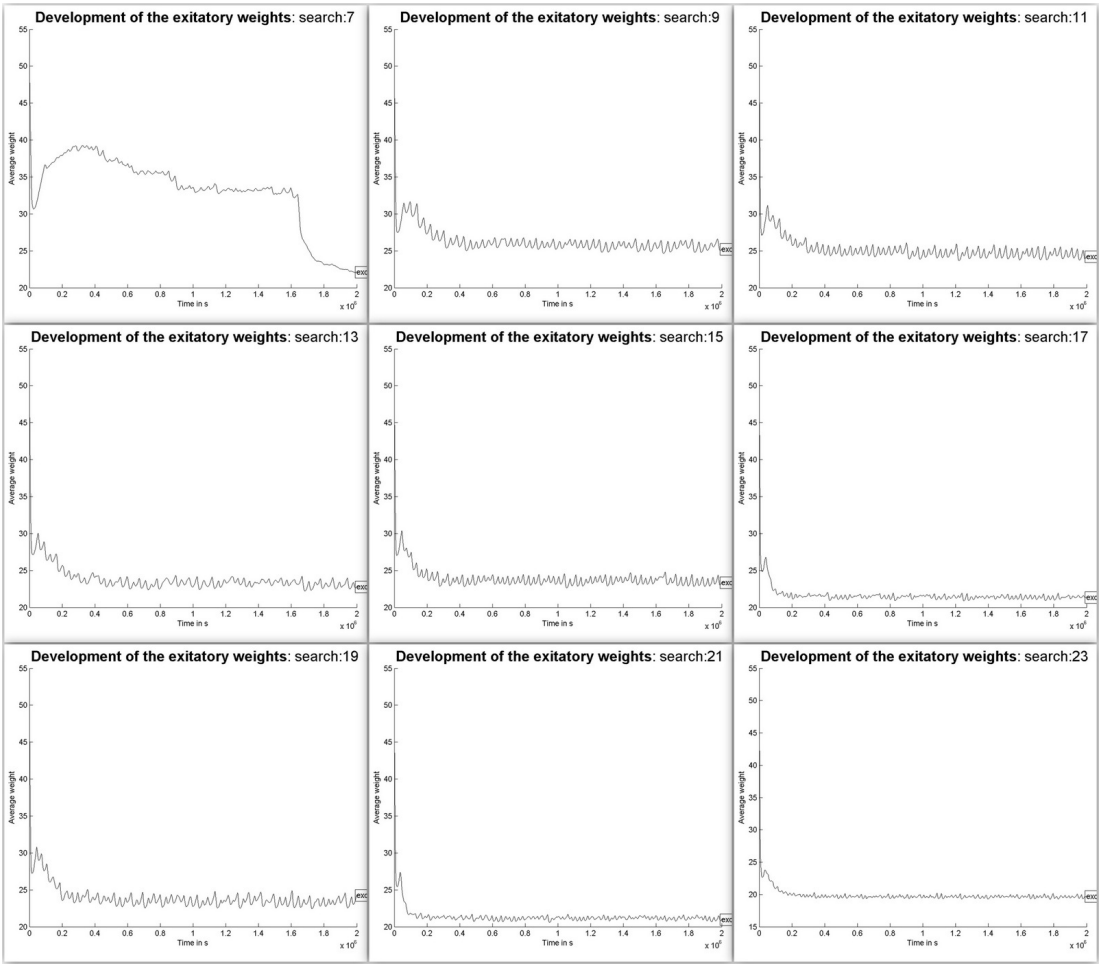


Figure 8.7: *Development of the excitatory weights in experiments with LTD=1.2. The first row corresponds to F=30 Hz, the second row to F=45 Hz, the third row to F=60 Hz. The first column corresponds to A=13 pA, the second column to A=20 pA, the third column to A=27 pA.*

The development of the excitatory weights. The development of excitatory weights is again significantly influenced by the LTD value.

- Whereas the lower LTD led to early (up to 200 s) stabilization and then small fluctuations around 23, the higher LTD led to gradual growth with later (around 1,000 s) stabilization on a value in the range between 30 and 60, (almost) without fluctuations.

There is not any clear trend in influence of F and A on the development of excitatory weights.

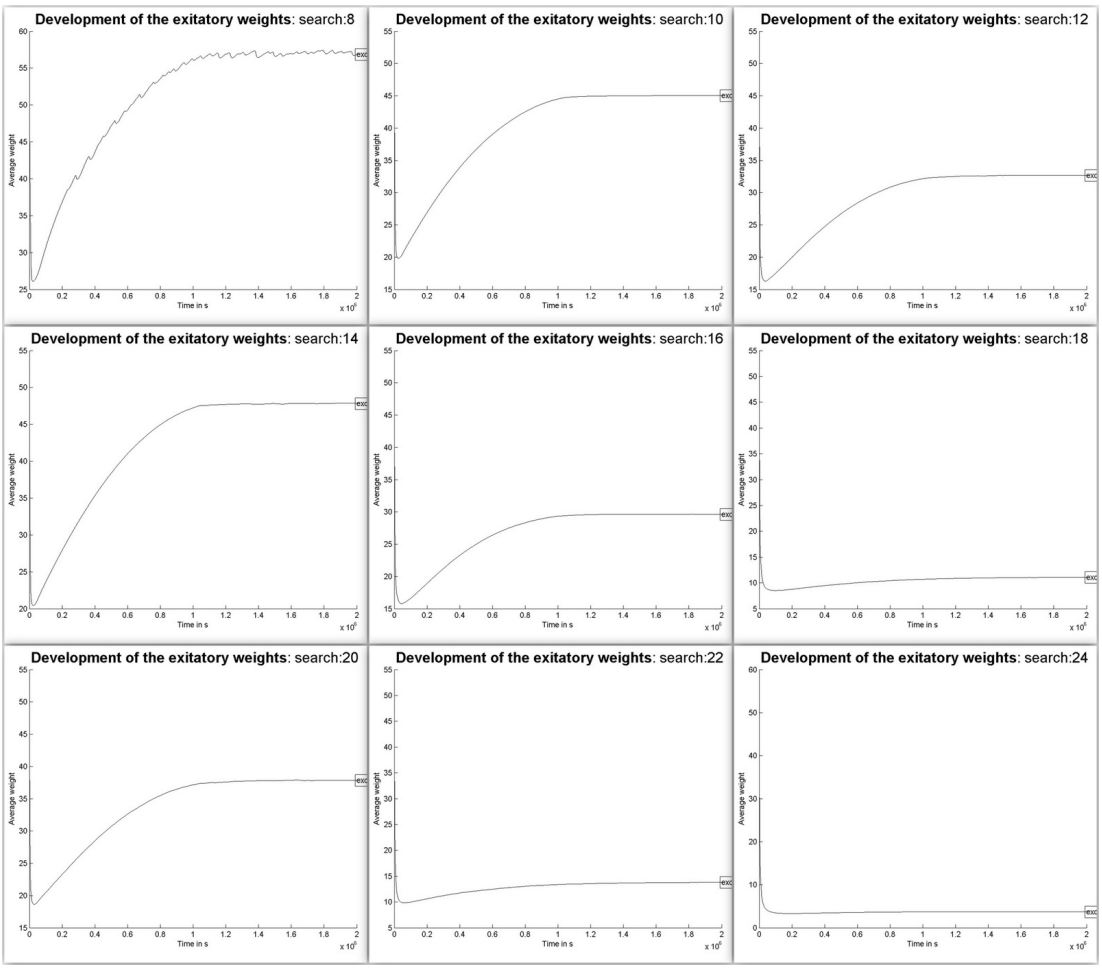


Figure 8.8: *Development of the excitatory weights in experiments with LTD=2.0. The first row corresponds to F=30 Hz, the second row to F=45 Hz, the third row to F=60 Hz. The first column corresponds to A=13 pA, the second column to A=20 pA, the third column to A=27 pA.*

8.1.3 Analysis and discussion of the results

First, we suggest some explanations of the described results. Second, we compare the results to the real observed data.

8.1.3.1 Explanations

The influence of the LTD value is significant and rich (in number of features which are influenced by this value). However, we can observe much logical coherence in the results. With the higher value, the network activity and weights are more stable and they fluctuate less. On the other hand the fluctuations in the case of lower LTD are correlated with the synchronous activity with the same frequency as the frequency of the fluctuations. It is a question, what is the exact mechanism of the origin of these fluctuations. It could be interesting to research it in detail, because level of neuronal synchrony seems to be an important factor of the memory and higher level processes, but also several diseases, such as Parkinson's disease (Nini et al., 1995).

The influence of F and A values is much simpler. It is logical that the instead of exact values of F and A, their product is the important factor (see Figures 8.2 and 8.3). As expected, the higher values of F·A lead to a higher network activity.

8.1.3.2 Comparison to the real data

We briefly comment the results in comparison with the real data. We divided this comparison according to the classes of features: the network activity, the oscillations (waves), and the weights.

The network activity. Comparing the absolute values of the firing rate to the real data is problematic for several reasons. Mainly, the results from reality are highly diverse and contradictory. The absolute value of the firing rate depends on the method of the experiment, for instance the anaesthesia reduces activity level (Young and Brownell, 1976; Evans and Nelson, 1973; Kuwada et al., 1987; Gaese, 2001; Wang et al., 2005) and different anaesthetics reduce the activity in different level. Other important factors are the area of recorded neurons, the particular neurons types, their number and distribution (e.g., higher percentage of FS neurons will lead to higher activity), the age of the individual (Oswald and Reyes, 2008) and many others.

However, the values of the mean firing rate between 0.5–20 Hz are generally corresponding to the reality (e.g., Sakata and Harris, 2012).

Instead of comparing the absolute values, it is reasonable to compare the qualitative features or proportion between firing rates of different neuron types. Surprisingly, there are not many studies describing the firing rate of different neuron types in different layers in the AC. In the study (Sakata and Harris, 2012), the putative fast spiking interneurons were significantly more active than the putative pyramidal neurons (recorded from layers L2/3 and L5). That is in correspondence with results observed in our model, where b2/3 and b5 (the only FS interneurons) are the clearly most active neuron types. On the other hand, the putative FS interneurons from layer L5 were in the study more active than those in layer L2/3, whereas in the model, it is almost always contrariwise. However,

it could be caused by the fact, that many parameters of the model (e.g., the distribution of the synaptic connections between particular neuron types) are not based on data from the AC, but also from other cortical areas (and other animals), see (Izhikevich and Edelman, 2008).

The network oscillations. The observed network oscillations are similar to those observed in the model by (Izhikevich and Edelman, 2008). The low frequencies (mainly delta and alpha) arose in the entire network, whereas the higher frequencies (especially gamma) cancelled each other when averaged over a bigger area, therefore being present only in local areas (see local waves in the Table 8.1). This is consistent with the experimental observation that gamma rhythms are weaker in EEG recordings than in LFPs and intracranial EEGs (Nunez and Srinivasan, 2006).

However, the comparison of the results of oscillations in our model and oscillations obtained from methods such as LFP, EEG, or ECoG, can be only approximate. Although these recording methods somehow register the activity of neurons from a certain area (sometimes called as population activity), they depend on many sources, and therefore do not exactly correspond just to the mean membrane potential, or even number of spikes. These sources include synaptic activity, intrinsic currents and resonances, gap junctions and neuron-glia interactions, or ephaptic effects (Buzsáki et al., 2012).

The development of the excitatory weights. It is generally positive that all tested parameters led to stable weights. Small fluctuations, occasional and gradual changes are generally consistent with results obtained from the AC of young (up to 1 month) mice (Oswald and Reyes, 2008).

8.1.4 Outcome

For the next experiments we chose the combination of parameters from the experiment number 19, i.e. LTD=1.2, F=60 Hz, A=13 pA, for the following reasons:

1. The values F=60 Hz and A=13 pA best correspond to the values observed in the real AC (after the relevant scaling, see the Section 3.5.5).

2. Development of the firing rate of all neuron types is relatively stable and the values are relatively reasonable. The higher values of F and A led to rather too high activity of the basket interneurons, whereas lower values of F and A led to a rather less stable activity of the basket interneurons.

3. In this combination of parameters, the theta and alpha global waves occurred and a rich range of local waves occurred. For this feature, the lower value of LTD was necessary.

4. The development of excitatory weights levelled off and remained relatively stable only with small fluctuations around the value of 24.

8.2 Features of the Chosen Parameters

The second group of experiments was designed to observe and describe the features of the parameters chosen in the previous section more in detail: in longer time scale and different size scale of the network.

8.2.1 Description of the experiments

We decided to test three sizes of the network: first: 10,000, second: 50,000, and third: 100,000 neurons. The approximate number of neurons in the modelled area in the real AC may be estimated between 500,000–1,000,000 of neurons, based on densities in the cerebral cortex (but not directly AC) published in (DeFelipe et al., 2002). The duration of the simulation was designed for 5 h, 2 h, and 30 min, respectively. In addition to analyse the spike trains only from the end of the simulation, we recorded and analysed the spike trains from several (10, 4, 10, respectively) equidistant segments of the simulation. All these values are listed in the Table 8.2. Parameters of the model, which are not listed in the table, are same as in all other experiments and can be found in the Section 7.8.

no.	N	T	F	A	LTD	INPUTS	LOG_S	LOG_D
1	10,000	5 h	60	13	1.2	minis	30 min	3 s
2	50,000	2 h	60	13	1.2	minis	30 min	3 s
3	100,000	30 min	60	13	1.2	minis	3 min	3 s

Table 8.2: *The settings of 3 experiments from the second group of experiments:* ***Features of the Chosen Parameters****. The meaning of the columns is following: N is number of neurons, T is duration of the experiment (in s of model time), F and A are values of the parameters* `M_FREQUENCY` *and* `M_AMPLITUDE` *of mPSCs, LTD is value of the parameter* `PARAM_LTD`*, INPUTS is type of inputs (here only spontaneous activity), LOG_S and LOG_D are values of parameters* `LOG_START` *and* `LOG_DURATION`*, which here means that we analysed each 30 min (3 min in the case of the largest network) a time frame with duration of 3 s.*

8.2.2 Results of the experiments

The description of the results is divided into three parts: the network activity, the oscillation spectrum, and the development of the excitatory weights. Each part describes how the related features developed over time and in comparison among the network sizes.

The network activity. The overall activity over the simulation is generally similar in all three network sizes; see Figure 8.9. Minor differences are observable in the shape of curves of the mean firing rate of particular neuron types. For instance, the amplitude of the fluctuations of b2/3 declines with the growing network size. In all sizes, the activity is relatively stable. However, in the case of the largest network, the longer simulation duration would be necessary, to confirm this observation (e.g., the mean firing rate of b5 did not stabilize until half of the simulation). The range of the mean firing rate values is also generally

very similar in all three network sizes. The most active neuron type is b2/3 with the mean firing rate between 30–40. The second most active neuron type is b5 with the mean firing rate around 10 in the case of 10,000 and 50,000 neurons, and around 15 in the case of 100,000 neurons. The values of other neuron types are in all cases lower than b2/3 and b5. For details, see Figure 8.9.

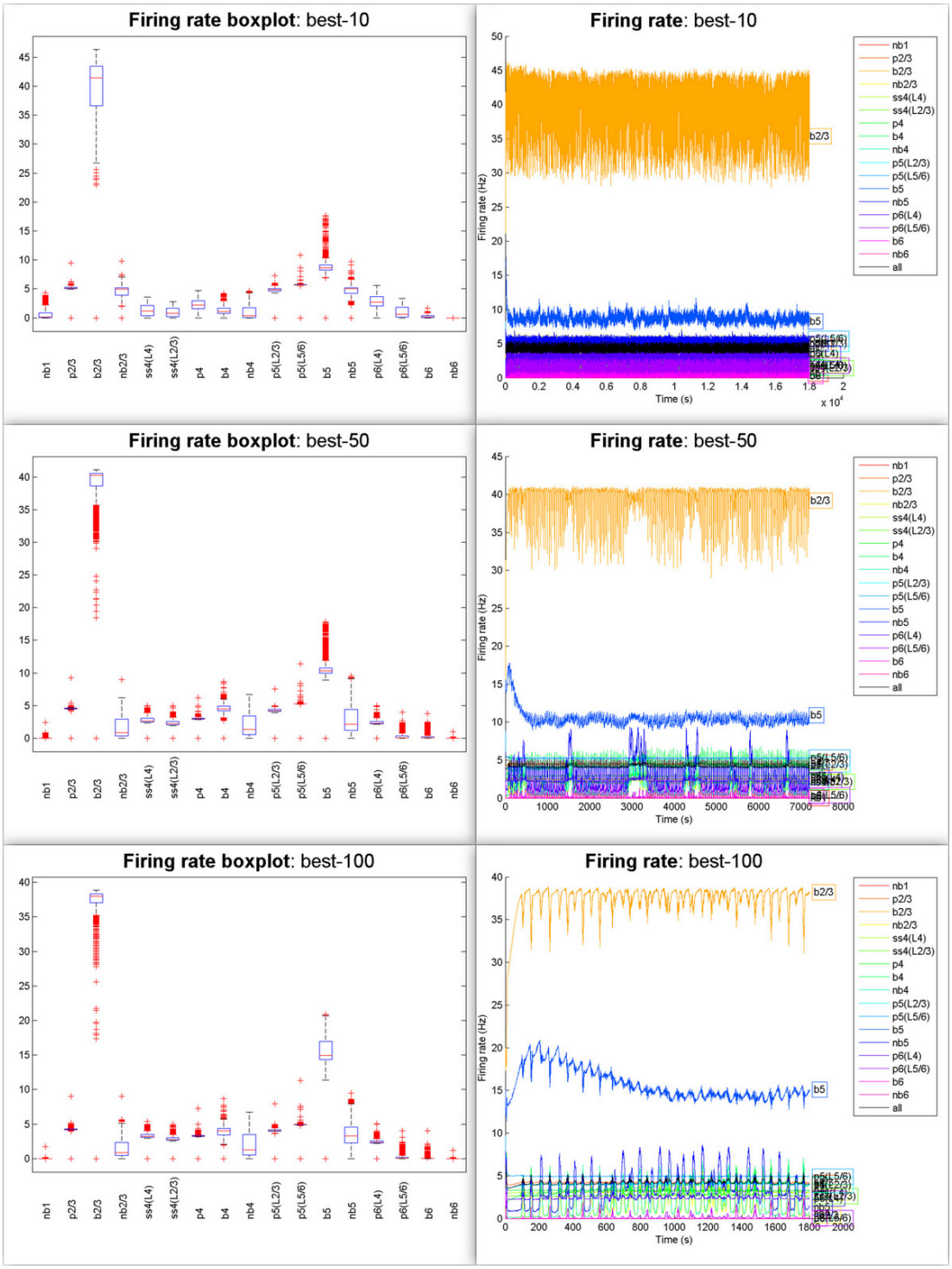


Figure 8.9: *A comparison of the mean network activity between three network sizes: 10,000 neurons (first row), 50,000 neurons (second row), and 100,000 neurons (third row).*

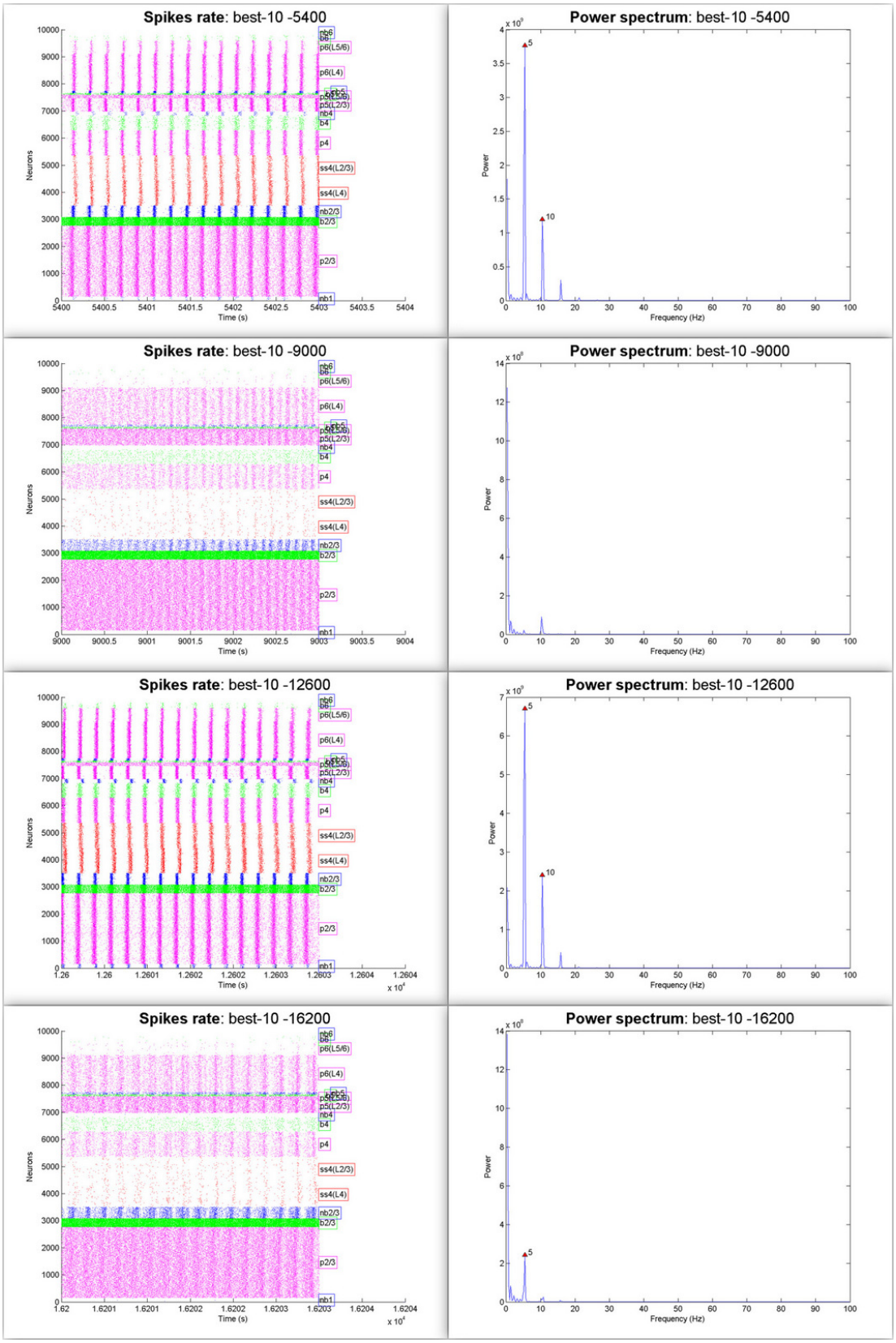


Figure 8.10: *The development of the network activity over time: network with 10,000 neurons. Left column depicts STRP, right column depicts global waves. Note the alternation of synchronized activity (first and third row) and desynchronized activity (second and fourth row).*

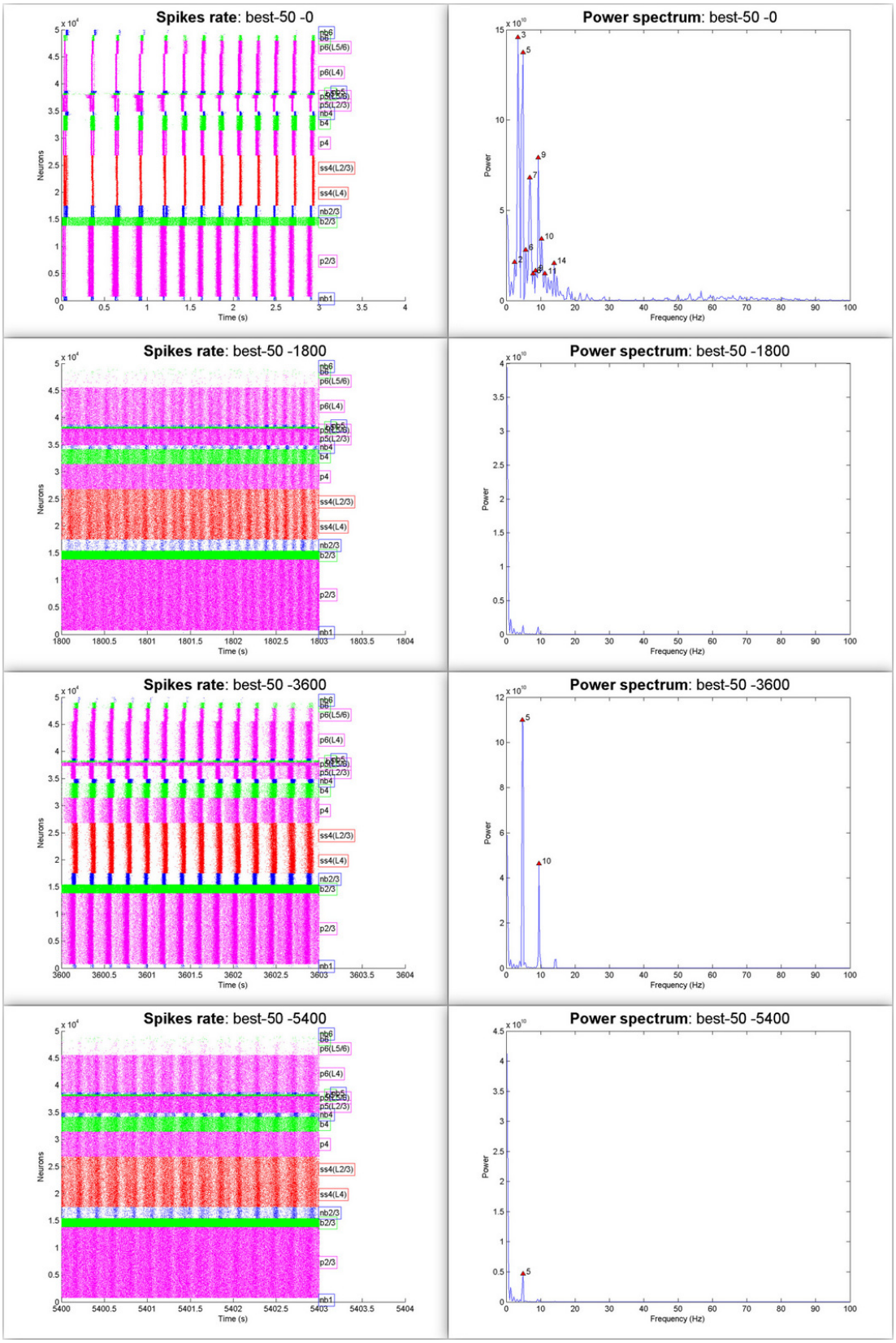


Figure 8.11: *The development of the network activity over time: network with 50,000 neurons. Left column depicts STRP, right column depicts global waves. Note the alternation of synchronized activity (first and third row) and desynchronized activity (second and fourth row).*

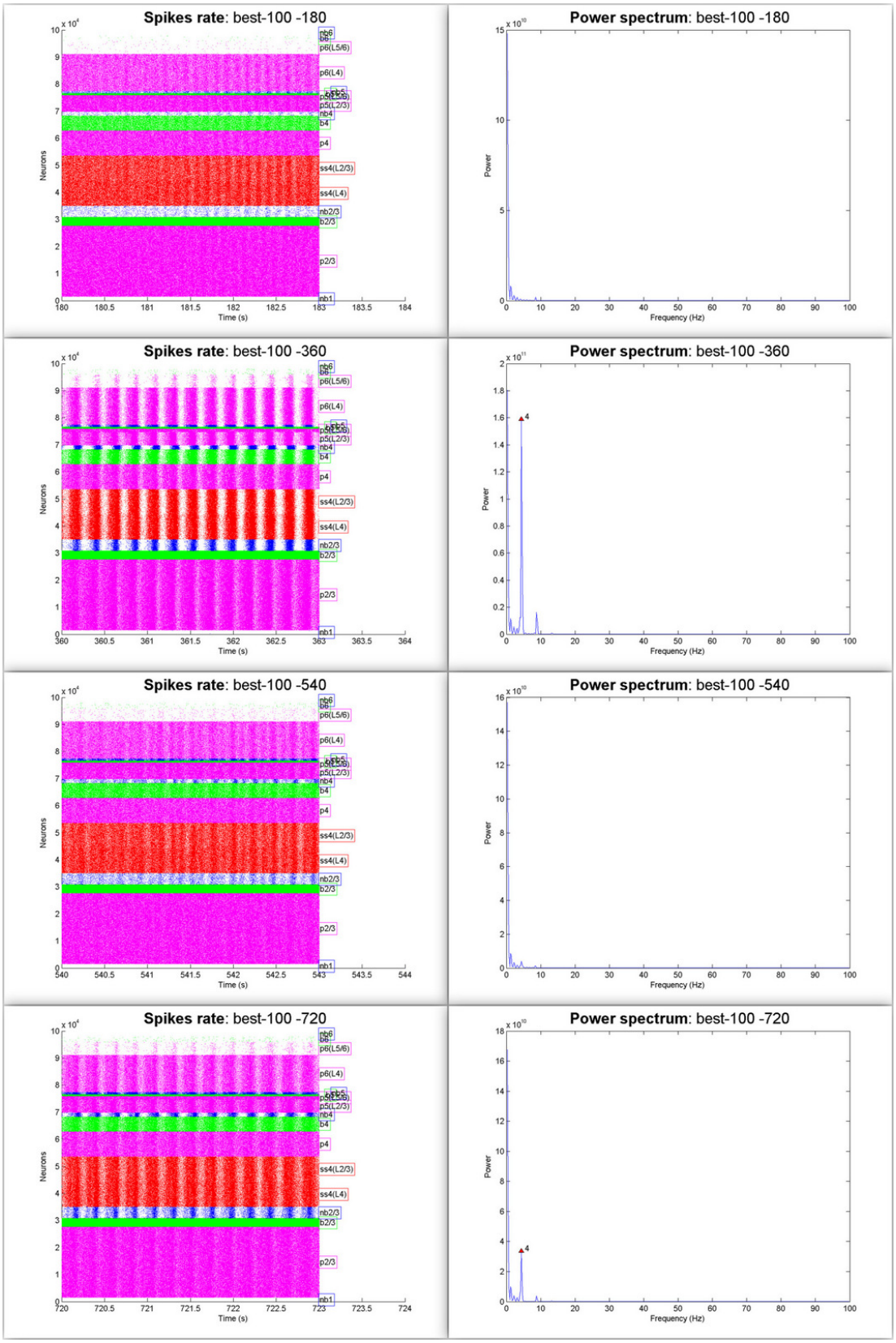


Figure 8.12: *The development of the network activity over time: network with 100,000 neurons. Left column depicts STRP, right column depicts global waves. Note the alternation of synchronized activity (first and third row) and desynchronized activity (second and fourth row).*

The network oscillations. The recordings of spike trains from more time frames over the simulation showed remarkable result in the network synchrony. As can be seen from the Figures 8.10, 8.11, and 8.12 two types of synchrony were present: a low synchrony with no global waves, and rather high synchrony with noticeable global waves (with frequencies 5 and 10, in the case of both smaller networks, and 4 or 9 in the case of the largest network).

To describe the exact structure of these two states (synchronized and desynchronized), more experiments would be necessary. It could be really interesting to observe and describe the development of alternations between these states, as well as an approximate duration. From the existing results, it seems that synchronized states are more often, or last longer. (This observation is not visible from the figures in the text, but from figures over the whole experiments.)

The development of the excitatory weights. In all the three cases, the weights were rather stable over the entire simulation, with values fluctuating between 20 and 40. The fluctuations were smallest in the smallest network and largest in the largest network; see Figure 8.13.

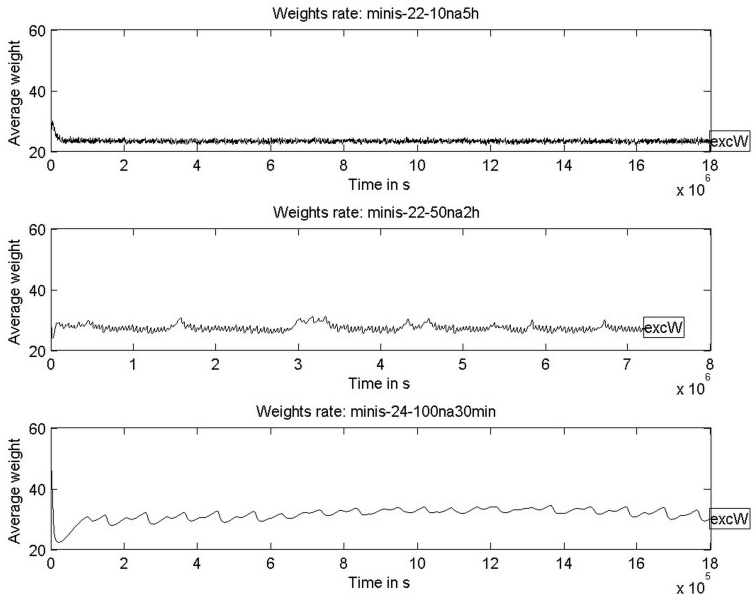


Figure 8.13: *A comparison of the development of the mean excitatory weights between the network sizes: 10,000 (first row), 50,000 (second row), and 100,000 (third row).*

8.2.3 Analysis and discussion of the results

The results display two positive qualities of the model: the observed characteristics are generally stable over time and very similar in networks with different size. In addition to that, a striking feature of two different states of synchrony can be observed (in all network sizes). Similar alternations of synchronized and desynchronized states were observed in the AC *in vivo* in both anesthetized (Sakata and Harris, 2012; Clement et al., 2008; Sakata and Harris, 2009), and unanesthetized (Sakata and Harris, 2012) animals.

8.3 Tonotopy Experiments

The last group of experiments researched the designed model under stimulation from the thalamus in addition to the spontaneous activity (i.e., the mPSCs were present in all experiments). The experiments were designed to test the network behaviour during (and after) stimulation by different types of inputs: noise, few pure tone, more pure tones and more complex tones (called normally distributed). In addition to the features measured in the previous two groups of experiments, in this last group we measured also features related to the tonotopic organization: receptive fields of single neurons and organization of their best frequencies. This group of experiments aims to basically test the hypothesis described in the Section 6.1.1, where the basic idea is that STDP could be a sufficient mechanism for the following observations: noise leads to very low topotopy, few pure tones to low tonotopy, and more complex tones to the typically observed tonotopic organization: tonotopy present in coarse, but not in fine, spatial scales (Bandyopadhyay et al., 2010).

8.3.1 Description of the experiments

Because the results of the previous group of experiments verified that different network sizes exhibited the qualitatively (but generally also quantitatively) the same features, we decided to run the tonotopy experiments only on the two smaller networks (i.e., first with 10,000 neurons and second with 50,000 neurons). Both networks were initialized by the final state from the previous experiments (i.e., after 5 h and 2 h of spontaneous activity, respectively). We conducted 4 types of experiments for both networks. These types differed in the inputs: noise, few pure tones, more pure tones, and "normal tones". First, we will briefly describe these inputs.

1. Noise: Each tick, 5 neurons from the input (L4) layer were randomly chosen and each of them was stimulated by an input with intensity of 50 dB (i.e., probability 0.5).

2. Few pure tones: One input lasted for 500 ms. Every 500 ms a new input was randomly chosen. In total, 6 inputs were used: nothing, or one of the given bands (see Figure 8.14). Each of these 6 inputs had the same probability to be chosen. All these inputs had the same intensity value 50 dB.

3. More pure tones: This experiment was the same as the previous one, except that here more bands were used. Instead of 5 bands as in the case of few pure tones, here 25 bands were used, see Figure 8.14.

4. Normal tones: In this experiment, one input does not mean stimulating a single band with a fixed intensity, but more bands, which specifically distributed intensities: the central band of the input is stimulated by the highest intensity and the intensity declines to both sides from the central band, as it is illustrated in Figure 8.14. In this experiment, 25 bands were chosen and the intensity declined with the following values: 50 db (the central band), 40 dB, 20 dB, 10 dB.

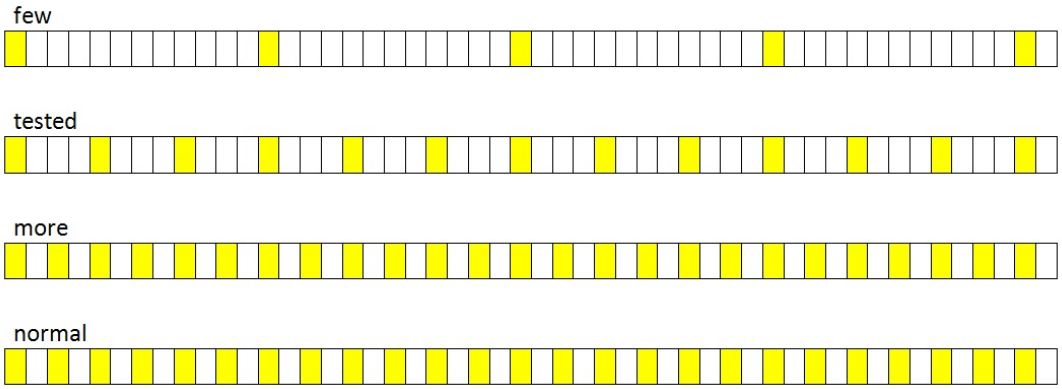


Figure 8.14: *A diagram of bands involved in different input types. First row: the input type "few pure tones" (5 bands). Second row: the tested input bands in the experiment, which measures RF and other tonotopy-related features (13 bands). Third row: the input type "more pure tones" (25 bands). Fourth row: the input type "normal tones" (25 bands, but with overlapping areas, see Figure 7.4).*

The smaller network was let to develop for 2 h, the bigger network for 30 min. The synoptic parameter settings are listed in the Table 8.3. Parameters of the model, which are not listed in the table, are same as in all other experiments and can be found in the Section 7.8.

no.	N	T	F	A	LTD	INPUTS	LOG_S	LOG_D
1	10,000	2 h	60	13	1.2	noise	30 min	5 s
2	10,000	2 h	60	13	1.2	few	30 min	5 s
3	10,000	2 h	60	13	1.2	more	30 min	5 s
4	10,000	2 h	60	13	1.2	normal	30 min	5 s
5	50.000	30 min	60	13	1.2	noise	10 min	5 s
6	50.000	30 min	60	13	1.2	few	10 min	5 s
7	50.000	30 min	60	13	1.2	more	10 min	5 s
8	50.000	30 min	60	13	1.2	normal	10 min	5 s

Table 8.3: *The settings of 8 experiments from the third group of experiments:* ***Tonotopy Experiments****. The meaning of the columns is following: N is number of neurons, T is duration of the experiment (in s of model time), F and A are values of the parameters* `M_FREQUENCY` *and* `M_AMPLITUDE` *of mPSCs, LTD is value of the parameter* `PARAM_LTD`*, INPUTS is type of inputs, LOG_S and LOG_D are values of parameters* `LOG_START` *and* `LOG_DURATION`*, which here means that we analysed each 30 min (10 min in the case of the larger network) a time frame with duration of 5 s.*

To analyse the tonotopic organization, another experiment on the network must be performed. A battery of particular inputs is designed and these inputs are step by step played to the network. We chose a battery of 65 inputs: a matrix of 13 bands and 5 intensities, illustrated in Figure 8.14. Each input lasted 100 ms and was followed by 400 ms of silence (but mPSCs were always present). This battery of inputs was randomly permutated and played to the network in five repetitions.

This experiment, which measures a tonotopic organization (we will call it mTono) was performed in an alternative reality each 10 min of the basic experiment (which will be called the lTono from learning). According to the chosen implementation of the alternative reality (see the Section 3.5.3), each 10 min each lTono experiment was saved and then on all these saved networks the mTono experiment was run.

8.3.2 Results of the experiments

First, the general features are described. Second, the features related to the tonotopy are described.

8.3.2.1 Description of the results of general features

The network activity. As can be seen from Figure 8.15, activity was in all experiments relatively stable over time. All inputs led to generally similar mean firing rates of all neuron populations, except of neuron types in L4, especially b4, which was most active when stimulated by "normal tones". All these observations hold both for smaller (10,000) and larger (50,000) network, but the figures are from the smaller one. Example of spike time raster plot are illustrated in Figure 8.16, all are from the same (last) part of the simulation.

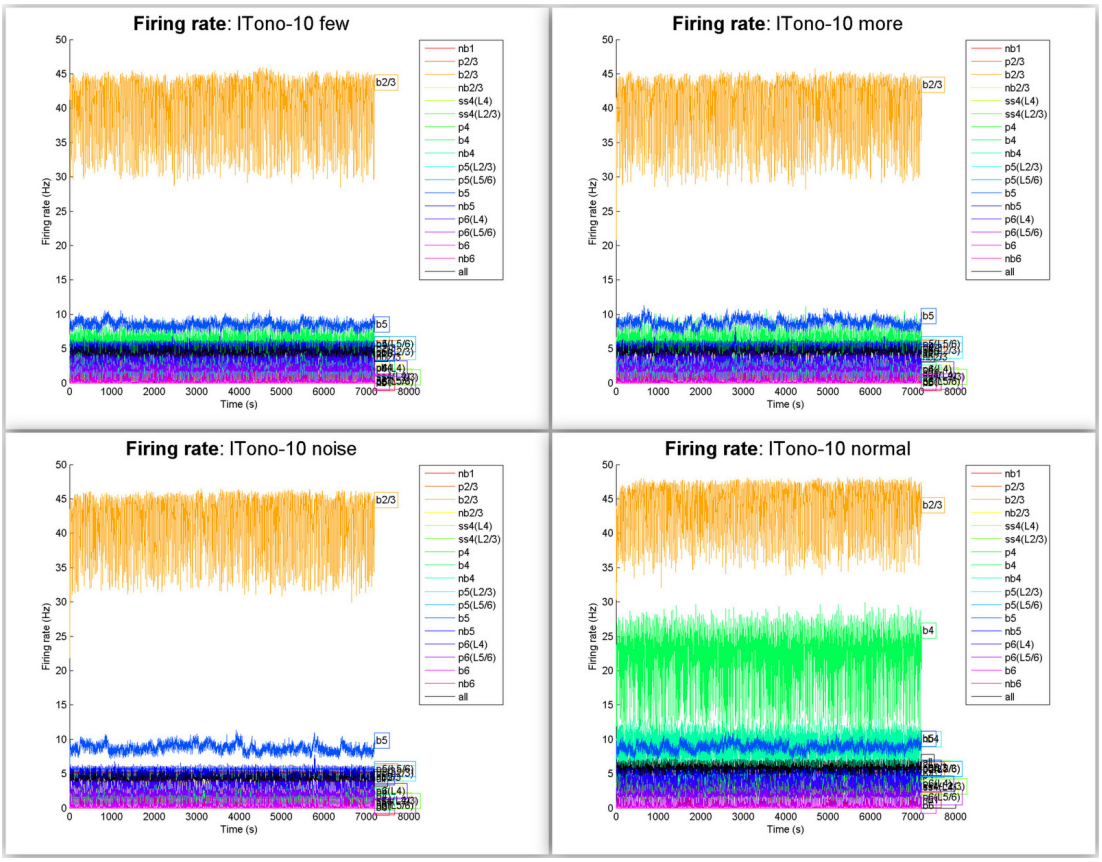


Figure 8.15: *The mean firing rate over time in comparison between input types. Left-top: few pure tones. Right-top: more pure tones. Left-bottom: noise. Right-bottom: Normal tones.*

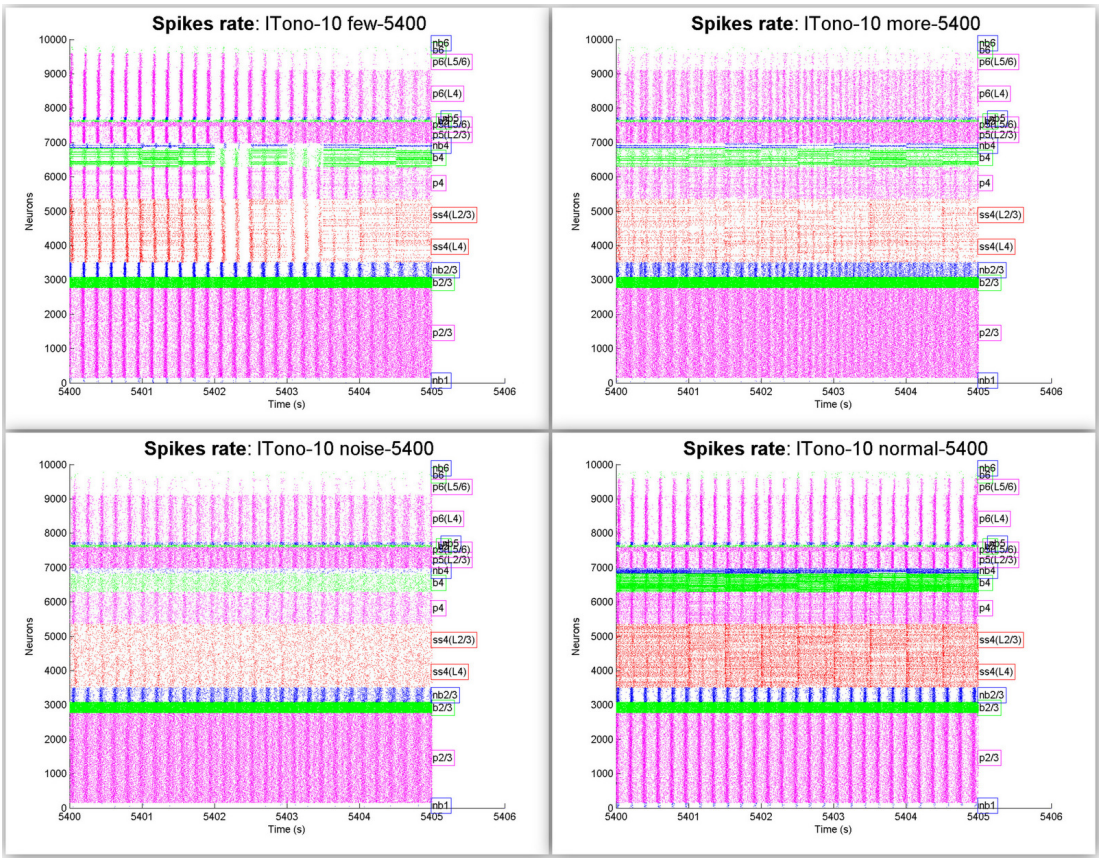


Figure 8.16: *STRP from the last time frame of the simulation in comparison between input types. Left-top: few pure tones. Right-top: more pure tones. Left-bottom: noise. Right-bottom: Normal tones.*

The network oscillations. Oscillations (both global and local) were generally the same in all input types, but differed between the network sizes (in the smaller network, local waves were more frequent and large). Again, the two states were noticeable: one, with very small (L4–6) or no (L1–3) oscillations, and second with markedly larger oscillations. These states were present in all experiments. Examples of local waves from these two states and deep and superficial layer are illustrated in Figure 8.17.

The development of the excitatory weights. Development of the excitatory weights was similar in both network sizes, and only slightly differed between input types. The most different was in the case of normal tones, where the weight fluctuations were lower than in other input types, noticeable in both network sizes.

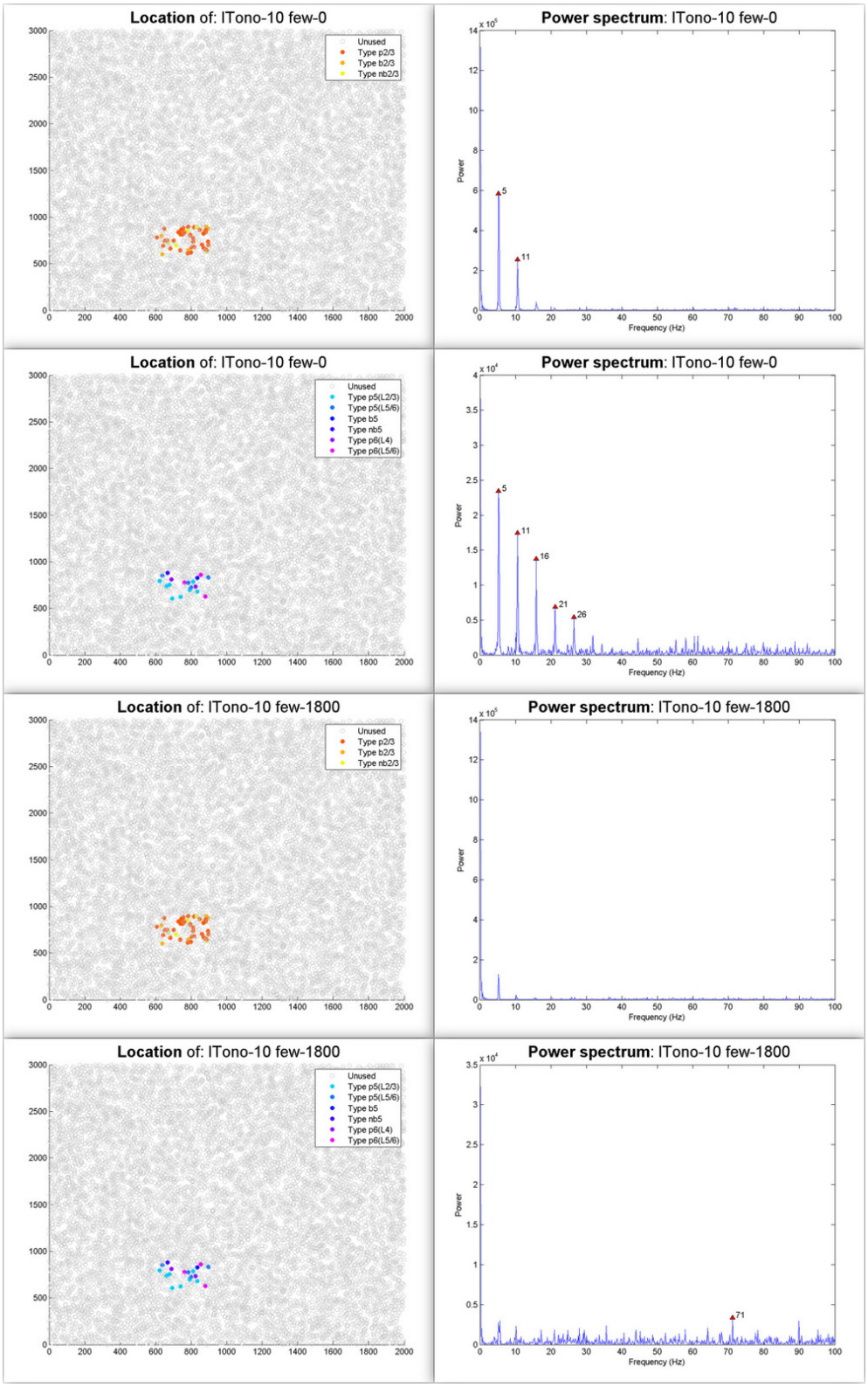


Figure 8.17: *The local oscillations. Two top rows: from the synchronized state of activity (first row from L1–L3, second row from L4–L6). Two bottom rows: from the desynchronized state of activity (third row again from L1–L3, fourth row from L4–L6).*

8.3.2.2 Description of the results of tonotopy-related features

We measured several features related to the organisation of tonotopy: receptive fields (RF), best frequency (BF), characteristic frequency (CF), tonotopic map, and measure of tonotopy (TM1). All these features have been described in the Section 7.5.3.

Receptive fields The receptive fields were very dependent on the neuron type. However, even within a neuron type, the receptive fields were typically highly different. To provide an idea of their diversity, see Figure 8.19. The most tuned neurons were present in the layer L4, which is not surprising, because this layer gets the direct tonotopic input from the auditory thalamus, in our model. The neurons, which lie directly in one of the tested bands and in L4, had mostly the "I" tuning curve. In other layers, such sharply tuned neurons were much less frequent, but not absent (see Figure 8.20). In contrary to the observations from the real AC, the "V" types of tuning curve were (almost) not present in the model. However, according to the data provided by Mgr. Ondřej Novák and Jakub Tomek from AC of mice, the observed RFs were generally relatively similar, see Figure8.21.

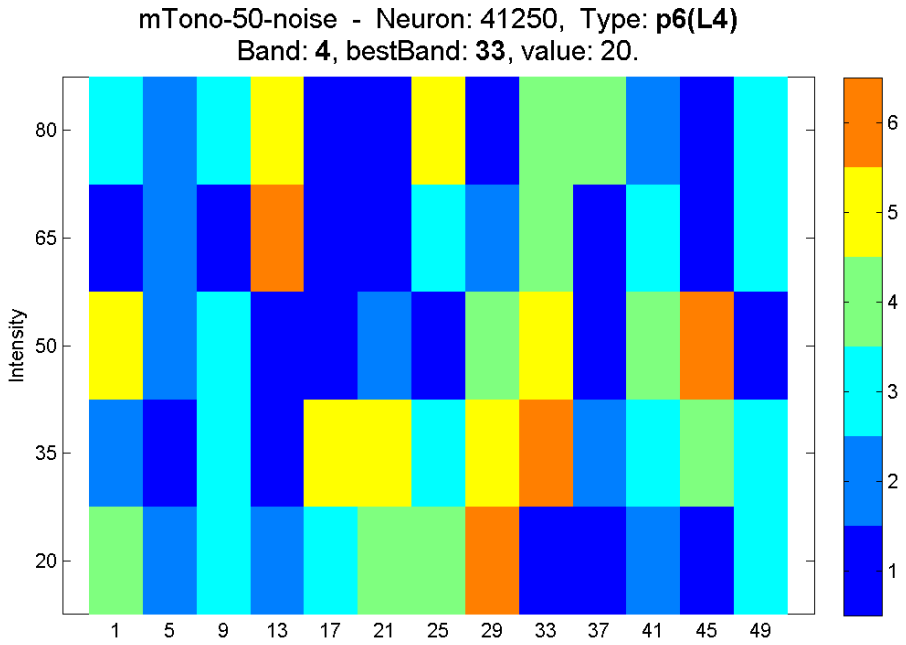


Figure 8.18: *A typical example of RF from the experiment with noise as inputs.*

The major difference between different input type experiments was noticeable in noise input type. The resulting RFs were much more diverse (see Figure 8.18) than in other input types.

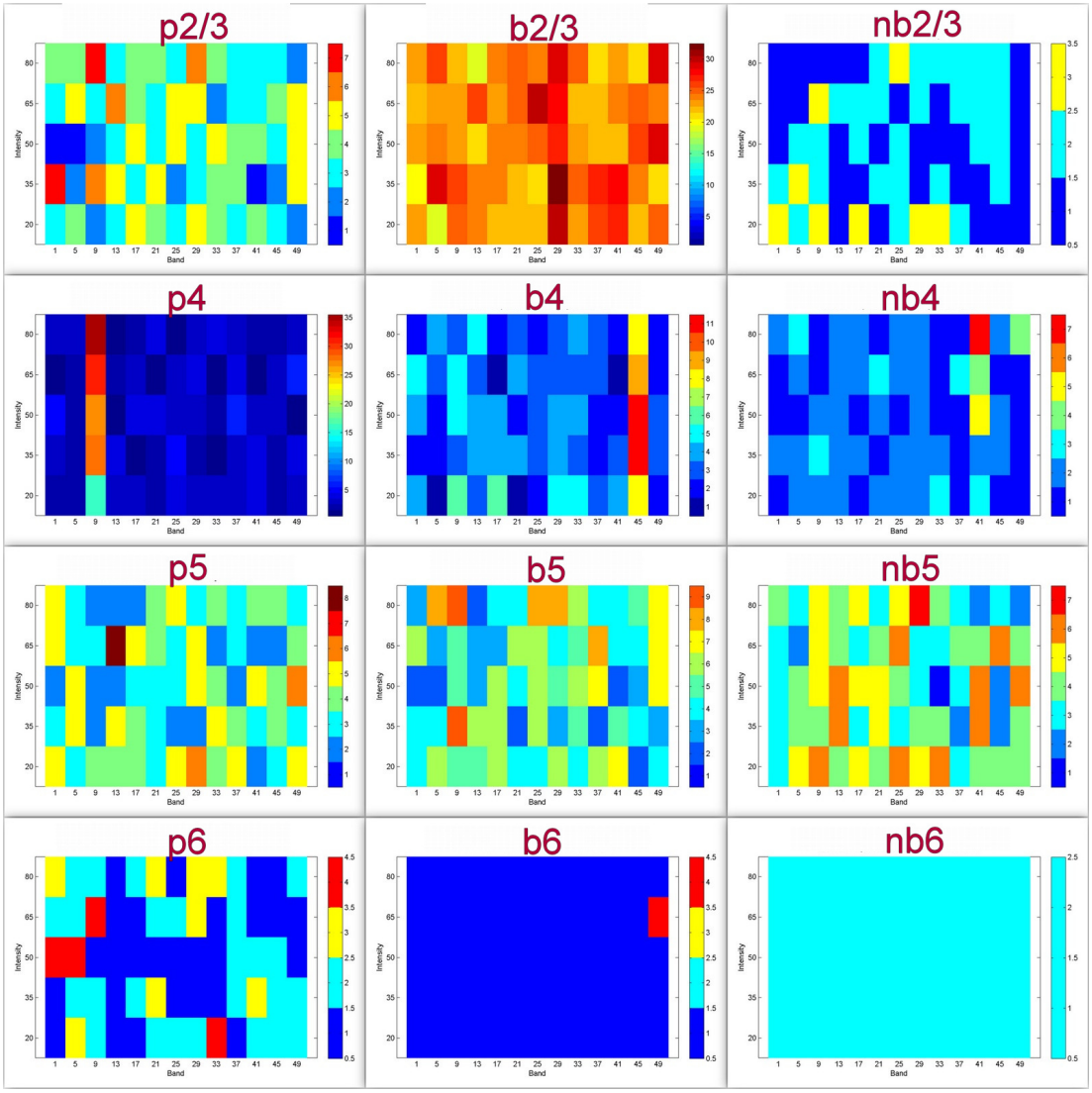


Figure 8.19: *Examples of receptive fields of different types of neurons. First row: layer L2/3, second row: layer L4, third row: layer L5, and fourth row: layer L6. All RFs were selected from the experiment with few pure tones.*

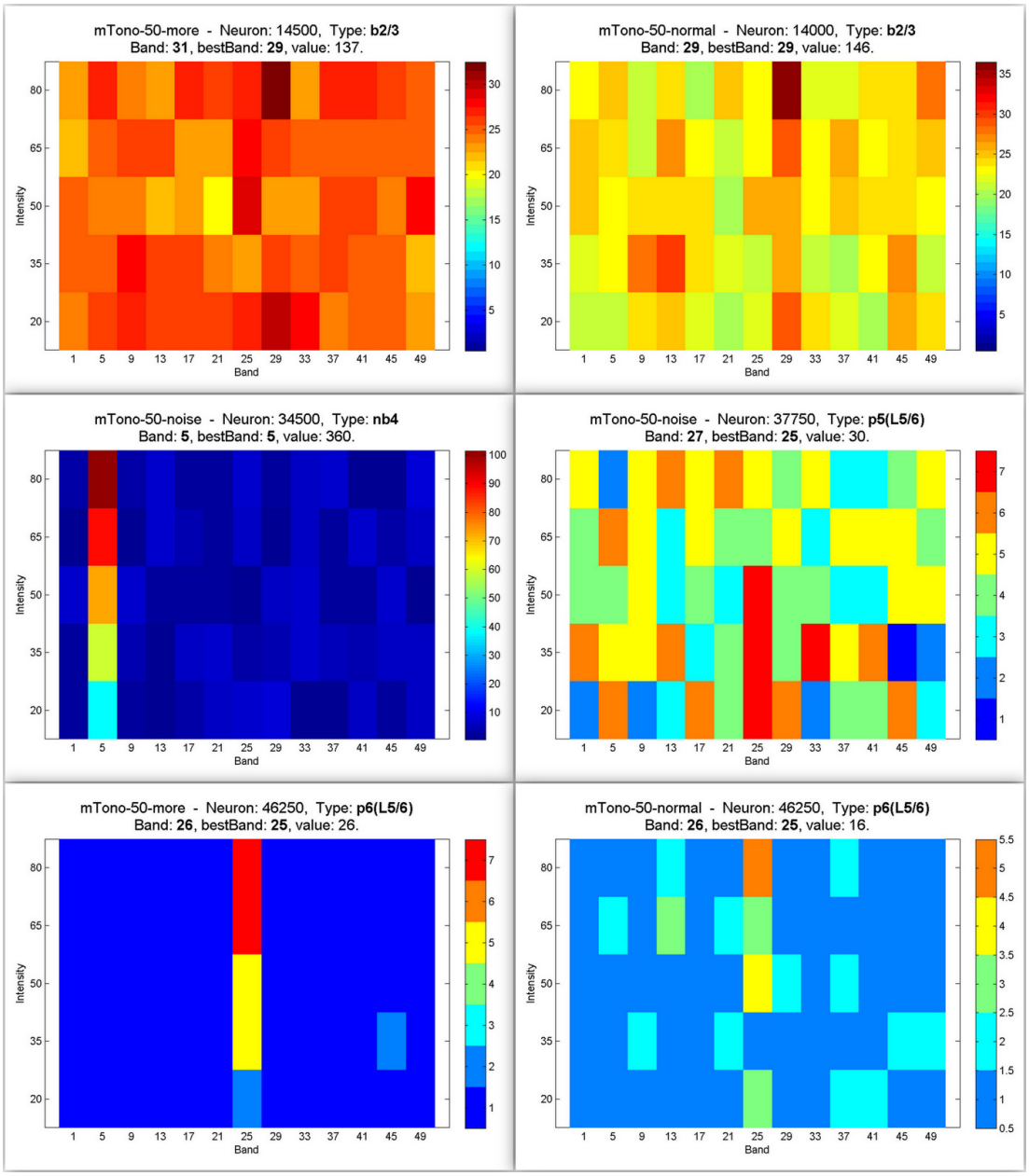


Figure 8.20: *Examples of tuned neurons (i.e., rather clear preference of one frequency) of different types of neurons. The apparent majority of tuned neurons was in the input layer.*

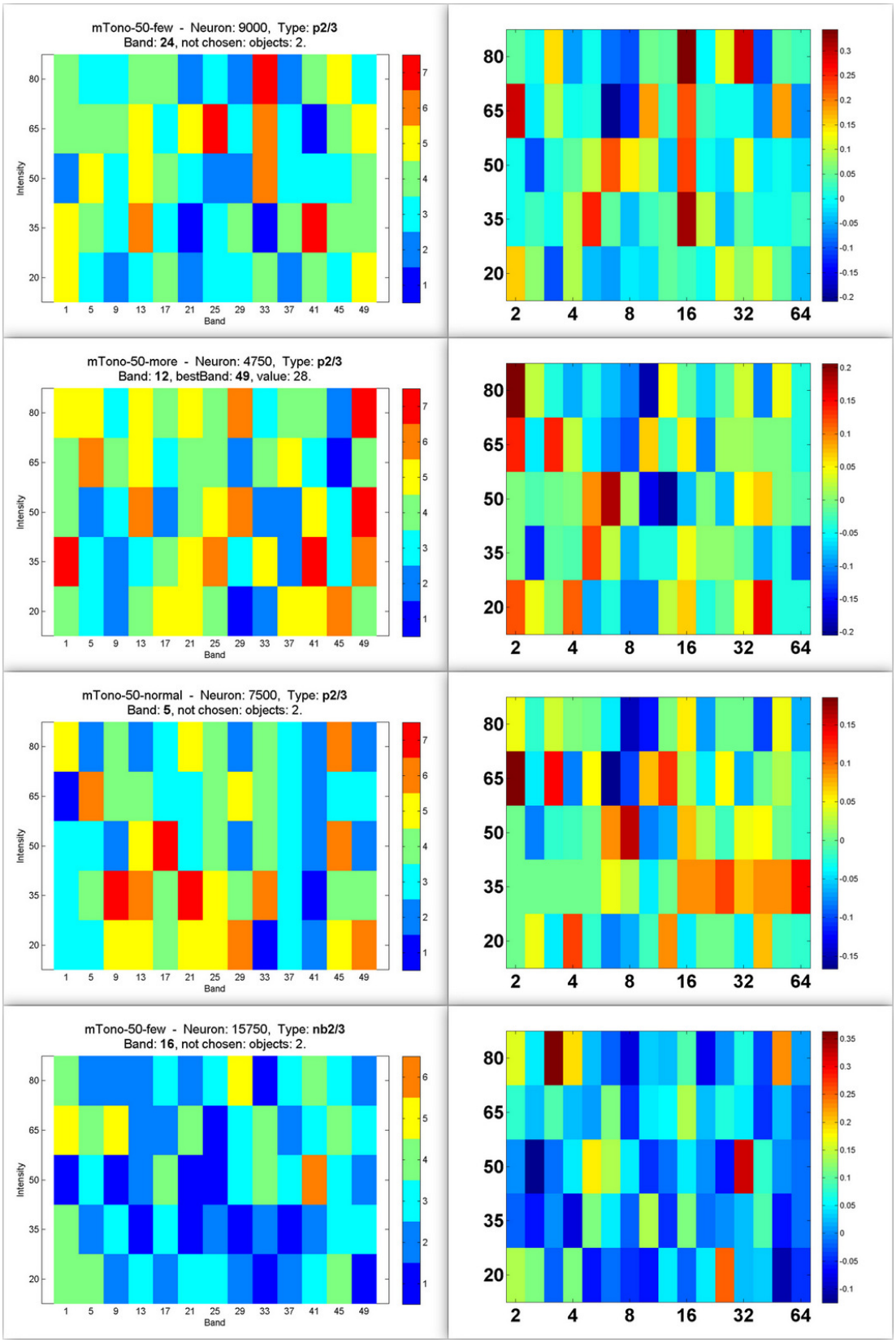


Figure 8.21: *A comparison of RFs from the model (left) and of RFs from in vivo experiments from the mouse AC (right). In each row a pair of rather similar RFs was manually selected – there is not any other known relation, e.g., location. However, all neurons were from layer L2/3 (both in silico and in vivo).*

Tonotopic maps The result obtained using BF and CF for the tonotopic map were not much different, but using CF led to easier recognition of the tonotopic organization, therefore in the rest of the results, we will describe the results from CF metric. It is relatively common, than neurons which are not tuned to any particular frequency are eliminated from the analysis (the same is used *in vivo* experiments, e.g., Rothschild et al., 2010).

The most clear characteristic observable from the tonotopic map is the difference between layers. While the input layer L4 contains clearly tuned bands, other layers are not so clearly tuned. The layer nb1 is very silent and with only minor exceptions were not included in the measurement. The layers L2/3 and L5 are not much tonotopic. On the contrary layer L6 (although it is not an input layer) was in some experiments noticeably tonotopically organised.

As can be seen in Figure 8.22 (data from the larger network), different tonotopic organization emerged in different input type experiments. The most tonotopic results can be seen in experiments with pure tones (both few and more). In the results with normal tones the tonotopy is also visible, but much less evidently. On the contrary, the experiment with noise resulted in very high degree of local heterogeneity. Interestingly, the resulting state in the case of noise inputs contained much higher percentage of neurons preferring high frequencies than low ones. We examined the development of the tonotopic organization in the case of noise inputs and observed that the development was highly unstable and changed from preferring low frequencies, over totally noisy distribution, to preferring high frequencies (see Figure 8.23, data from the smaller network). Other types of inputs led to much more stable development.

Interestingly, the results from the smaller network were not so clear (see Figure 8.23). The degree of local heterogeneity was higher than in the larger network. However, the noise input experiment is again distinct in having predominant preference for a common range of frequency (here lower).

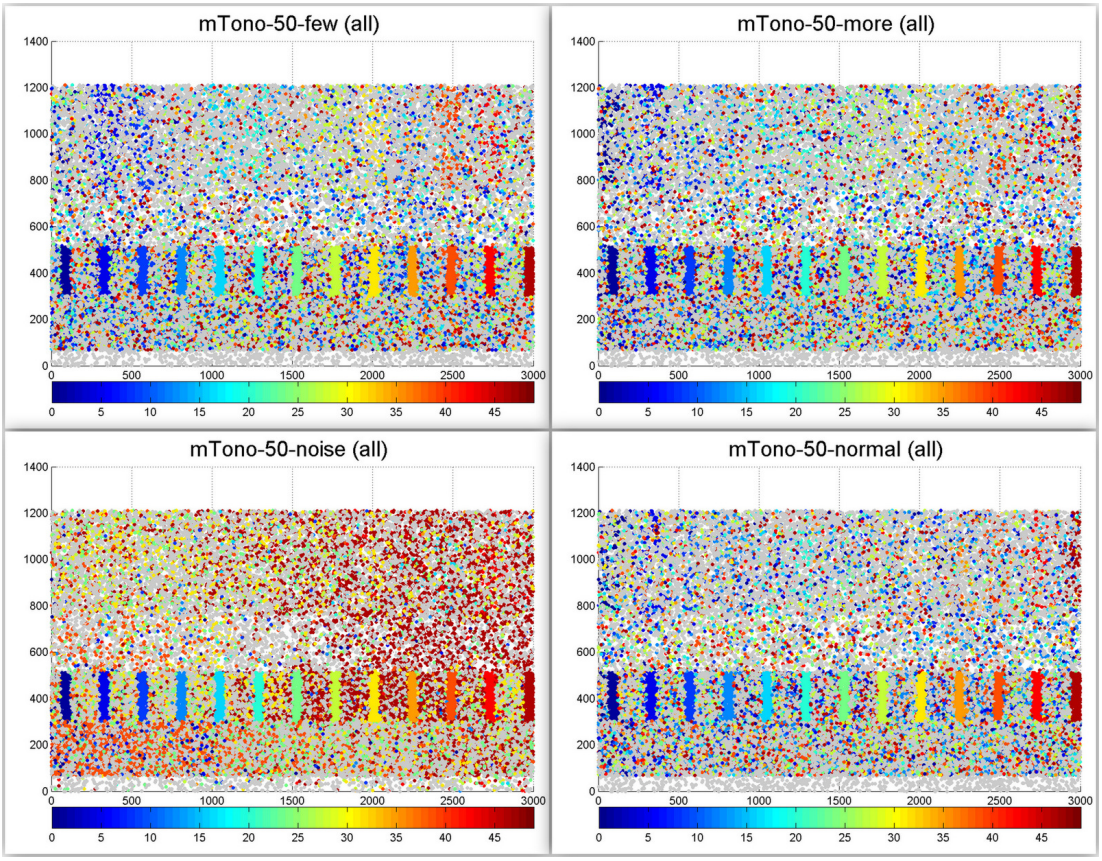


Figure 8.22: *A tonotopic map after the learning experiment. The network with 50,000 neurons. Left-top: few pure tones (note the tonotopy also in other than input layers). Right-top: mor pure tones (also rather tonotopic). Left-bottom: noise (disturbed tonotopy and apparent preference of extreme (here high) frequencies). Right-bottom: normal tones (very coarse tonotopy).*

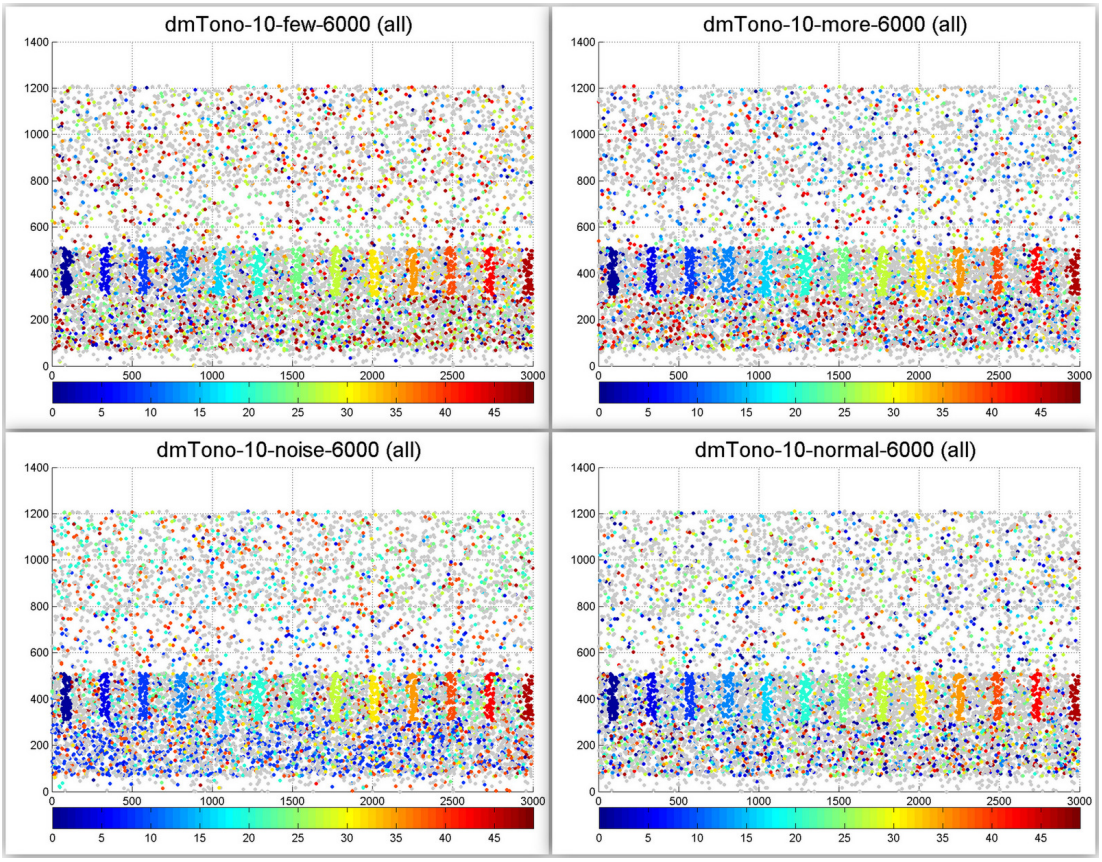


Figure 8.23: *A tonotopic map after the learning experiment. The network with 10,000 neurons. Left-top: few pure tones. Right-top: mor pure tones. Left-bottom: noise. Right-bottom: normal tones.*

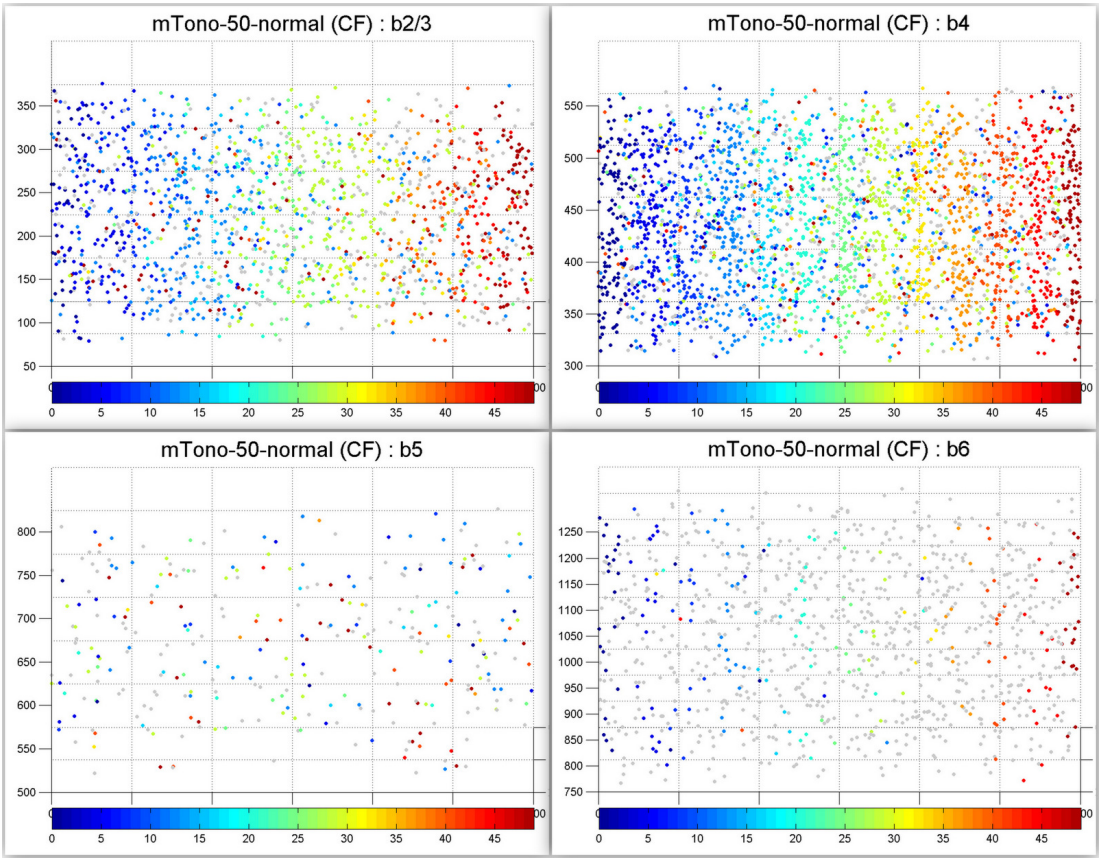


Figure 8.24: *An example of tonotopic maps within the basket neuron types. Results from the larger network, after the learning experiment. Note that the tonotopy is apparent even in non-input layers (also L2/3).*

Degree of local heterogeneity The resulting degree of local heterogeneity is displayed in Figure 8.25. It corresponds to the previous observations. The noise experiment is the most heterogeneous one, especially in layers L4–6. However the measure of tonotopy is relatively balanced between neuron types. The experiments with pure tones have clearly lowest degree of local heterogeneity in L6. On the other hand, experiment with normal tones led to high tonotopy of basket neurons in all layers (see also Figure 8.24). This observation was consistent between network sizes and during the experiment.

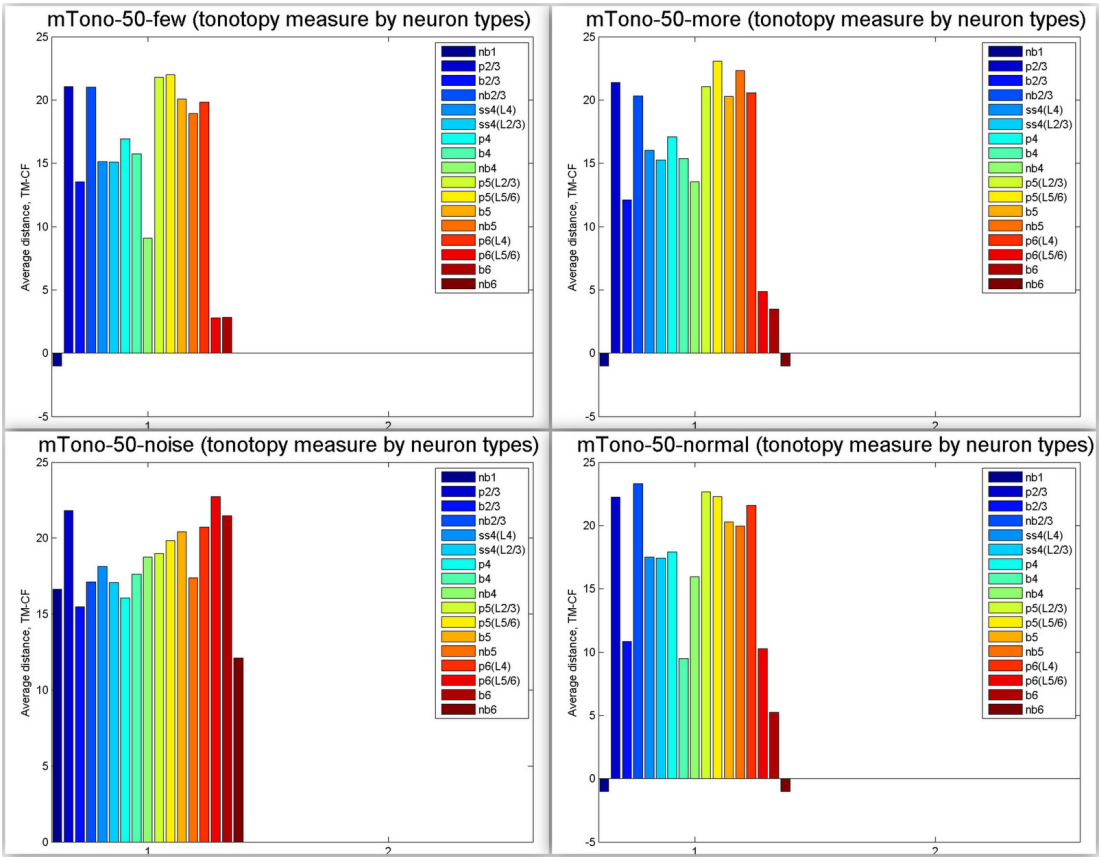


Figure 8.25: *Degree of local heterogeneity of the network with 50,000 neurons after the learning experiment. Left-top: few pure tones. Right-top: mor pure tones. Left-bottom: noise. Right-bottom: normal tones.*

8.3.3 Analysis and discussion of the results

We will summarize and discuss the five main results from the last group of experiments:

1. The frequency organization is more heterogeneous in L2/3 (the main output layer) than in L4 layer. This is consistent with *in vivo* observations from the mouse AC (Winkowski and Kanold, 2013).

2. The observed RFs are generally similar to RFs observed *in vivo* data from the mouse AC (see Figure 8.21).

3. The results confirmed the main points of the hypothesis on emergence of tonotopy: noise led to disrupted tonotopy, while pure tones led to reasonable tonotopy. However, the difference between few and more pure tones was not so distinct. The normal tones led to an intermediate state. But further experiments would be needed to compare these results to the *in vivo* observations.

4. The network size matters and small network may not be sufficient for emergence of clearer tonotopy. This point may be influenced by the fact that the smaller network was sparser and therefore neurons did not always receive the ideal number of synapses (recommended by the connectome table). It is possible that the density of connections was not sufficient for propagation of tonotopy.

5. The emergence of tonotopy is, on one hand, logical result of tonotopic inputs and higher probability of near connections than distant ones. However, on the other hand, when realising that in our model as well as in reality, the connections are not (at least not simply) deterministic and distant connections and relatively orderless connections are present, the emergence of tonotopy is also not obvious. (This is supported also by observations in our model, where noise or only spontaneous activity was not sufficient for emergence of tonotopy in other non-input layers.) We assume that for this balance of tonotopy and heterogeneity, the connectome of the model should not be totally deterministic (e.g., neurons arranged in a grid and nearby neurons connected), as it was often modelled in the existing AC models (de Pinho et al., 2006; Zhou et al., 2012; Chrostowski et al., 2011). In such deterministic connectome the tonotopic arrangement emerges much easier.

8.4 Conclusion

The designed model of the AC resulted in behaviour consistent, in several aspects, with *in vivo* observations. The main achievements are summarized in the following chapter. However, the performed experiments have several limitations. We list the main features, which could be done in future more in detail:

1. A better and more precise comparison to the real data. This phase should be performed in cooperation of a computer scientist with a neuroscientist(s), because a computer scientist is not typically so well versed in neuroscience

and does not have so broad knowledge of the published neuroscientific results. In addition to that, a more precise comparison to the real data would be beneficial not only for model validation, but especially for searching for new coherences. These could lead to design of new hypotheses and experiments (both *in vivo* and *in silico*). Among others, let us mention e.g., experiments with travelling waves, which are assumed to be related to various disorders (Charles and Brennan, 2009; Houweling et al., 2005).

In this thesis, the model was designed in cooperation with neuroscientists, but the analysis of the experiments must was done only by the author of the thesis.

2. A broader exploration the space of parameters: more parameters and more values.

3. Longer experiments. It is positive that all features measured during the spontaneous activity seemed stable. However, in such dynamic and stochastic systems as neural networks, changes may happen even after longer period of time. More importantly, the experiments with specific inputs lasted only up to 2 h of model time, which much less than the critical learning period of real brain (which lasts days to weeks).

4. Larger networks. It is positive that the main measured features were generally same in all tested network sizes. However in the tonotopic experiments, the results were different. On one hand, it indicates that networks with less than 10,000 of neurons may be too small to lead to plausible results. On the other hand, it also indicates that even 50,000 or 100,000 neurons (as tested in this thesis) may not be sufficient and at least verifying this fact would be beneficial.

5. More repetitions. The main measured features exhibited generally the same results in all performed experiments (even in many preliminary experiments, which are not described in the text). However, the tonotopic experiments should be done in more repetitions to prove their significance.

9. Model of the Auditory Cortex: Discussion

In this thesis, we presented a model of the AC with the following features:

1. Spiking neurons modelled by the Izhikevich neuron model, which is considered toe one of the most plausible models applicable in large-scale networks
2. A high diversity of neuronal types (17 types)
3. Synaptic connections with conduction delays and long term plasticity in the form of STDP
4. A detailed stochastic connectome based on the real data – however not exactly from the AC, but from the visual cortex of cat (Binzegger et al., 2004), adopted from (Izhikevich and Edelman, 2008)
5. Structure of network based on the real data from (DeFelipe et al., 2002)
6. Included model of spontaneous synaptic release (mPSCs) based on the real data from (Kotak et al., 2005)

We tested the model in three sizes: 10, 50, and 100 thousand of neurons. The largest network contained almost 21 million of synapses. We tested the model in dozens of experiments, where the longest lasted 5 h of model time. The results described in this thesis were based on 50 final experiments, which lasted together 133,000 s (over 1.5 day) of model time and 920,000 s (over 10 days) of real time. The automatic analyses of the experiment (in the Matlab part of SUSNOIMAC tool) lasted at least half of this time. In addition to that, we do not count here a large number of preliminary experiments, which were necessary to run before the final experiments.

To our knowledge, any other model of the AC of a comparable size and duration of simulation was not published yet.

The main results from the experiments are:

1. Stable development of all measured features (e.g., firing rate and weights)
2. Presence of the network oscillations: low frequencies arose in the entire network, whereas high frequencies (e.g, gamma) were present in local areas, which is consistent with real data (Izhikevich and Edelman, 2008; Nunez and Srinivasan, 2006).
3. Presence of two alternating states: one with higher synchronisation (observable from both the spike times raster plot and analysis of global and local waves) and one with lower synchronization. This observation seems similar to the observations from the real AC *in vivo* in both anesthetized (Sakata and Harris, 2012; Clement et al., 2008; Sakata and Harris, 2009), and unanesthetized (Sakata and Harris, 2012) animals. We consider this feature important also for two other reasons. First, this feature was not

built in the model, but emerged on its own. Second, we are not aware of other studies of models with Izhikevich neurons, which would describe this feature.

4. Clear emergence of tonotopy in the input layer L4 and much lower degree of tonotopy in the main output layer L2/3, which is in outline consistent with *in vivo* observations from the recent study (Winkowski and Kanold, 2013). We also studied resemblance of the model results and *in vivo* results in terms of the receptive fields of neurons from the layer L2/3. The RFs were in outline qualitatively comparable. However, a more detailed comparison could be contributing and potentially inspiring.

5. The results confirmed the main points of the hypothesis described in the Section 6.1.1: noise led to disrupted tonotopy (as *in vivo* (Chang and Merzenich, 2003; Zhang et al., 2002)), while pure tones led to reasonable tonotopy. Therefore it contributes the idea that the coarse tonotopy, but local heterogeneity may be an emergent result of tonotopic input from the thalamus and the long-term plasticity. However, additional experiments would be necessary to examine the influence of types of inputs on the degree of tonotopy (both *in vivo* and *in silico*). Also a more detailed comparison of degree of tonotopy would be necessary, to explore the hypothesis in its entire meaning.

6. The size of network had a great impact on the tonotopy-related results (in small network the degree of tonotopy in non-input layers was much lower). This may be considered a result useful for future modelling of the AC (e.g., when scientist looks for model of neuron and simulator, the knowledge of the bottom bound of network size to achieve a certain phenomenon may be beneficial).

We suggest the following future works with the model:

1. More detailed data about inner structure of the network (connectome, types of neurons, etc.), based on data exactly from the AC, ideally from one species.

2. More realistic and complex inputs from the auditory thalamus (e.g., inputs coding AM and FM sweeps)

3. Relation to other parts of the cortex/brain (e.g., inputs on the boudaries, potentially also feedback from L6 to thalamus)

4. Parameters of mPSCs dependent on the synapse or neuron types

5. Measurement of other features (e.g., Post Stimulus Time Histogram (PSTH) and classification of ON and OFF neurons, Rate Level Function (RLF), Inter-spike Intervals (ISI), travelling waves, other measurements related to tonotopy, noise correlations, oscillations measured as weighted average of membrane potentials, instead of spikes, polychronous groups, and others)

6. Other future work related to experiments, which have been described in the concluding section of the Model Results: Section 8.4, such as even larger network and longer simulation.

7. Other enhancements of the used models (multi-compartmental model of neuron, different synapse kinetics (AMPA, NMDA, GABA), short-term plasticity), dopaminergic rewards, etc.)

8. Population encoding of auditory stimuli (e.g., in the form of polychronous groups)

9. Extensions of the model to research origins of neural disorders (e.g., hearing loss, or epilepsy)

10. Conclusion

In this thesis, we presented a new simulator SUSNOIMAC (Simulator Using Spiking Neurons Originally Intended for Modelling Auditory Cortex) and a new model of the AC. The simulator SUSNOIMAC can be used for simulating networks with any parametrization of Izhikevich neurons, synaptic delays and STDP long-term plasticity. It contains many performance optimizations (as well as built-in paralellization) that make possible to compute large networks (we tested networks, which had up to 100 thousands of neurons and 21 millions of synapses). Besides, it allows any connectivity between neurons, and high flexibility in inputs, as well as built-in mechanism for spontaneous activity (mPSCs). The models can be implemented in Java, or the existing model can be used without need of programming. It also allows batch processing of more experiments. In addition to the simulation part, SUSNOIMAC contains also tools for analysis of the results: for smaller networks a retrospective visualization in Java and for all networks several scripts for analysis written in Matlab.

The main features and results of the model were described in the previous Chapter 9. The model consists of six layers (but layers L2 and L3 are treated as one), 17 neuron types, and detailed connectome.

The main future works related to the model were also described in the previous Chapter 9. Possible future works related to the simulator would be further optimization (e.g., using CUDA), extension for other modelling possibilities (e.g., short-term plasticity), or user graphical interface for creating new models.

The main achievement of this thesis could be summarized as follows:

1. Software: the SUSNOIMAC simulator (to our knowledge the first simulator specialized for modelling the AC).

2. Modelling impact: besides the model itself (its achievements were summarized in the previous chapter), we also performed a review of the existing models of the AC (to our knowledge the first review of the models of the AC with spiking neurons), and conducted several experiments with the model.

3. Biological: the results of the experiments lay new inspiration on several phenomenons (mainly the tonotopy) and may lead to new explanations of these phenomenons in the future.

Bibliography

ALONSO, Jose-Manuel, CHEN, Yao. Receptive field. *Scholarpedia*, **4**(1):5393, 2009.

ALPAYDIN, Ethem. *Introduction to machine learning*. The MIT Press, 2004.

ANANTHANARAYANAN, Rajagopal, ESSER, Steven K., SIMON, Horst D., MODHA, Dharmendra S. The cat is out of the bag: cortical simulations with 109 neurons, 1013 synapses. In *High Performance Computing Networking, Storage and Analysis, Proceedings of the Conference on*, pages 1–12. IEEE, 2009.

ARIELI, Amos, SHOHAM, D., HILDESHEIM, R., GRINVALD, A. Coherent spatiotemporal patterns of ongoing activity revealed by real-time optical imaging coupled with single-unit recording in the cat visual cortex. *Journal of Neurophysiology*, **73**(5):2072–2093, 1995.

ASSMANN, Peter, SUMMERFIELD, Quentin. *The perception of speech under adverse conditions*, pages 231–308. Springer, 2004.

BANDYOPADHYAY, Sharba, SHAMMA, Shihab A., KANOLD, Patrick O. Dichotomy of functional organization in the mouse auditory cortex. *Nature neuroscience*, **13**(3):361–368, 2010.

BAO, Shaowen, CHANG, Edward F., DAVIS, Jonathan D., GOBESKE, Kevin T., MERZENICH, Michael M. Progressive degradation and subsequent refinement of acoustic representations in the adult auditory cortex. *The Journal of neuroscience*, **23**(34):10765–10775, 2003.

BARBOUR, Dennis L., CALLAWAY, Edward M. Excitatory local connections of superficial neurons in rat auditory cortex. *The Journal of Neuroscience*, **28** (44):11174–11185, 2008.

BART, Evgeniy, BAO, Shaowen, HOLCMAN, David. Modeling the spontaneous activity of the auditory cortex. *Journal of computational neuroscience*, **19**(3): 357–378, 2005.

BATHELLIER, Brice, USHAKOVA, Lyubov, RUMPEL, Simon. Discrete neocortical dynamics predict behavioral categorization of sounds. *Neuron*, **76**(2): 435–449, 2012.

BAXTER, Douglas A., BYRNE, John H. *Simulator for neural networks and action potentials*, pages 127–154. Springer, 2007.

BEAR, Mark F., CONNORS, Barry W., PARADISO, Michael A. *Neuroscience*. Lippincott Williams & Wilkins, 2007.

BEIERLEIN, Michael, GIBSON, Jay R., CONNORS, Barry W. Two dynamically distinct inhibitory networks in layer 4 of the neocortex. *Journal of neurophysiology*, **90**(5):2987–3000, 2003.

BHALLA, Upinder S. Use of kinetikit and genesis for modeling signaling pathways. *Methods in enzymology*, **345**:3–23, 2002.

BINZEGGER, Tom, DOUGLAS, Rodney J., MARTIN, Kevan A. C. A quantitative map of the circuit of cat primary visual cortex. *The Journal of Neuroscience*, **24**(39):8441–8453, 2004.

BIONDI, Emanuele. Auditory processing of speech and its implications with respect to prosthetic rehabilitation. the bioengineering viewpoint. *International Journal of Audiology*, **17**(1):43–50, 1978.

BIONDI, Emanuele, SCHMID, Roberto. Mathematical models and prostheses for sense organs. *Theory and Applications of Variable Structure Systems. London: Academic*, pages 183–211, 1972.

BIZLEY, Jennifer K., NODAL, Fernando R., NELKEN, Israel, KING, Andrew J. Functional organization of ferret auditory cortex. *Cerebral Cortex*, **15**(10): 1637–1653, 2005.

BLACK, E. P. Euclidean distance, 2004. URL `http://www.nist.gov/dads/HTML/euclidndstnc.html`.

BLACK, E. P. Manhattan distance, 2006. URL `http://www.nist.gov/dads/HTML/manhattanDistance.html`.

BOWER, James M., BEEMAN, David. *Constructing realistic neural simulations with GENESIS*, pages 103–125. Springer, 2007.

BOWER, James M., BOLOURI, Hamid. *Computational modeling of genetic and biochemical networks*. The MIT Press, 2004.

BOWER, James M., BEEMAN, David, WYLDE, Allan M. *The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System*. Telos New York, NY, 1998.

BRETTE, Romain, RUDOLPH, Michelle, CARNEVALE, Ted, HINES, Michael, BEEMAN, David, BOWER, James M., DIESMANN, Markus, MORRISON, Abigail, GOODMAN, Philip H., HARRIS JR, Frederick C. Simulation of networks of spiking neurons: a review of tools and strategies. *Journal of computational neuroscience*, **23**(3):349–398, 2007.

BROCKMANN, L. (wikipedia.org) C. Anatomy of the human ear, 2009. URL `http://en.wikipedia.org/wiki/File:Anatomy_of_the_Human_Ear.svg`.

BRODAL, Per. *The central nervous system*. Oxford University Press, 2010.

BROWN, Guy J., COOKE, Martin. Computational auditory scene analysis. *Computer speech and language*, **8**(4):297–336, 1994.

BROWN, Randy. Calendar queues: a fast 0 (1) priority queue implementation for the simulation event set problem. *Communications of the ACM*, **31**(10): 1220–1227, 1988.

BRUGGE, John F., REALE, Richard A. Auditory cortex. *Cerebral cortex*, **4**: 229–271, 1985.

BRUNEL, Nicolas, ROSSUM, Mark C. W. Lapicque's 1907 paper: from frogs to integrate-and-fire. *Biological cybernetics*, **97**(5-6):337–339, 2007.

BRUNEL, Nicolas, WANG, Xiao-Jing. Effects of neuromodulation in a cortical network model of object working memory dominated by recurrent inhibition. *Journal of computational neuroscience*, **11**(1):63–85, 2001.

BUZSáKI, György, ANASTASSIOU, Costas A., KOCH, Christof. The origin of extracellular fields and currents—eeg, ecog, lfp and spikes. *Nature Reviews Neuroscience*, **13**(6):407–420, 2012.

CANNON, Robert C., HASSELMO, Michael E., KOENE, Randal A. From biophysics to behavior. *Neuroinformatics*, **1**(1):3–42, 2003.

CARNEVALE, Nicholas T., HINES, Michael L. *The NEURON book*. Cambridge University Press, 2006.

CHANG, Edward F., MERZENICH, Michael M. Environmental noise retards auditory cortical development. *Science*, **300**(5618):498–502, 2003.

CHARLES, Anthony, BRENNAN, K. C. Cortical spreading depression—new insights and persistent questions. *Cephalalgia*, **29**(10):1115–1124, 2009.

CHROSTOWSKI, Michael, YANG, Le, WILSON, Hugh R, BRUCE, Ian C, BECKER, Suzanna. Can homeostatic plasticity in deafferented primary auditory cortex lead to travelling waves of excitation? *Journal of computational neuroscience*, 30(2):279–299, 2011.

CLEMENT, Elizabeth A., RICHARD, Alby, THWAITES, Megan, AILON, Jonathan, PETERS, Steven, DICKSON, Clayton T. Cyclic and sleep-like spontaneous alternations of brain state under urethane anaesthesia. *PLoS One*, **3** (4):e2004, 2008.

CONNORS, Barry W., GUTNICK, Michael J. Intrinsic firing patterns of diverse neocortical neurons. *Trends in neurosciences*, **13**(3):99–104, 1990.

CONTRERAS, Diego. Electrophysiological classes of neocortical neurons. *Neural Networks*, **17**(5):633–646, 2004.

CORMEN, Thomas H., LEISERSON, Charles E., RIVEST, Ronald L., STEIN, Clifford. *Introduction to algorithms*. MIT press, 2001.

CRUIKSHANK, Scott J., ROSE, Heather J., METHERATE, Raju. Auditory thalamocortical synaptic transmission in vitro. *Journal of neurophysiology*, **87** (1):361–384, 2002.

DAVISON, Andrew P., BRüDERLE, Daniel, EPPLER, Jochen, KREMKOW, Jens, MULLER, Eilif, PECEVSKI, Dejan, PERRINET, Laurent, YGER, Pierre. Pynn: a common interface for neuronal network simulators. *Frontiers in neuroinformatics*, **2**, 2008.

DAYAN, Peter, ABBOTT, Laurence F., ABBOTT, L. *Theoretical neuroscience: Computational and mathematical modeling of neural systems*. MIT press Cambridge, MA, 2001.

ROCHA, Jaime, MARCHETTI, Cristina, SCHIFF, Max, REYES, Alex D. Linking the response properties of cells in auditory cortex with network architecture: cotuning versus lateral inhibition. *The Journal of Neuroscience*, **28** (37):9151–9163, 2008.

PINHO, Marilene, SILVA, Antônio C. A realistic computational model of formation and variability of tonotopic maps in the auditory cortex. *Neurocomputing*, **26**:355–359, 1999.

PINHO, Marilene, MAZZA, Marcelo, ROQUE, Antônio C. A biologically plausible computational model of classical conditioning induced reorganization of tonotopic maps in the auditory cortex. *Neurocomputing*, **32**:685–691, 2000.

PINHO, Marilene, MAZZA, Marcelo, ROQUE, Antonio C. A realistic model of tonotopic reorganization in the auditory cortex in response to cochlear lesions. *Neurocomputing*, **38**:1169–1174, 2001.

PINHO, Marilene, MAZZA, Marcelo, PIQUEIRA, José Roberto C., ROQUE, Antonio C. Shannon's entropy applied to the analysis of tonotopic reorganization in a computational model of classical conditioning. *Neurocomputing*, **44**: 359–364, 2002.

PINHO, Marilene, MAZZA, Marcelo, ROQUE, Antônio C. A computational model of the primary auditory cortex exhibiting plasticity in the frequency representation. *Neurocomputing*, **70**(1):3–8, 2006.

DEFELIPE, Javier, ALONSO-NANCLARES, Lidia, ARELLANO, Jon I. Microstructure of the neocortex: comparative aspects. *Journal of neurocytology*, **31**(3-5):299–316, 2002.

DESAI, Niraj S., CUDMORE, Robert H., NELSON, Sacha B., TURRIGIANO, Gina G. Critical periods for experience-dependent synaptic scaling in visual cortex. *Nature neuroscience*, **5**(8):783–789, 2002.

EDWARDS, Brent. *Hearing aids and hearing impairment*, pages 339–421. Springer, 2004.

EDWARDS, Brent. The future of hearing aid technology. *Trends in Amplification*, **11**(1):31–46, 2007.

EGGERMONT, J. J. *Computational models of the auditory system*, volume **35**, chapter The Auditory Cortex: The Final Frontier. Springer, 2010.

EGGERMONT, Jos J. Between sound and perception: reviewing the search for a neural code. *Hearing research*, **157**(1):1–42, 2001.

ELF, Johan, EHRENBERG, Måns. Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases. *Systems biology*, **1**(2): 230–236, 2004.

ELIASMITH, Chris, ANDERSON, C. Charles H. *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. The MIT Press, 2003.

ELIASMITH, Chris, STEWART, Terrence C., CHOO, Xuan, BEKOLAY, Trevor, DEWOLF, Travis, TANG, Charlie, RASMUSSEN, Daniel. A large-scale model of the functioning brain. *science*, **338**(6111):1202–1205, 2012.

ERMENTROUT, Bard. *Simulating, analyzing, and animating dynamical systems: a guide to XPPAUT for researchers and students*, volume **14**. Society for Industrial and Applied Mathematics, 2004.

EVANS, E. F., NELSON, P. G. The responses of single neurones in the cochlear nucleus of the cat as a function of their location and the anaesthetic state. *Experimental Brain Research*, **17**(4):402–427, 1973.

FALL, Christopher P. *Computational cell biology*, volume **20**. Springer Verlag, 2005.

FIDJELAND, Andreas K., SHANAHAN, Murray P. Accelerated simulation of spiking neural networks using gpus. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.

FITZHUGH, Richard. Mathematical models of threshold phenomena in the nerve membrane. *The bulletin of mathematical biophysics*, **17**(4):257–278, 1955.

FODRóCZI, Zoltán, RADVáNYI, András. Computational auditory scene analysis in cellular wave computing framework. *International journal of circuit theory and applications*, **34**(4):489–515, 2006.

GAESE, Bernhard H. Population coding in the auditory cortex. *Progress in brain research*, **130**:221–230, 2001.

GERBER, Urs. Metabotropic glutamate receptors in vertebrate retina. *Documenta ophthalmologica*, **106**(1):83–87, 2003.

GERSTNER, Wulfram, KISTLER, Werner M. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

GEWALTIG, Marc-Oliver, DIESMANN, Markus. Nest (neural simulation tool). *Scholarpedia*, **2**(4):1430, 2007.

GIBSON, Jay R., BEIERLEIN, Michael, CONNORS, Barry W. Two networks of electrically coupled inhibitory neurons in neocortex. *Nature*, **402**(6757):75–79, 1999.

GLEESON, Padraig, STEUBER, Volker, SILVER, R. Angus. neuroconstruct: a tool for modeling networks of neurons in 3d space. *Neuron*, **54**(2):219, 2007.

GLEESON, Padraig, CROOK, Sharon, BARNES, Simon, SILVER, Angus. Interoperable model components for biologically realistic single neuron and network models implemented in neuroml. In *Frontiers in Neuroinformatics. Conference Abstract: Neuroinformatics*, 2008.

GODDARD, Nigel H., HUCKA, Michael, HOWELL, Fred, CORNELIS, Hugo, SHANKAR, Kavita, BEEMAN, David. Towards neuroml: model description methods for collaborative modelling in neuroscience. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, **356**(1412): 1209–1228, 2001.

GOLDBERG, David E. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Professional, 1989.

GONZALEZ-ISLAS, Carlos, WENNER, Peter. Spontaneous network activity in the embryonic spinal cord regulates ampaergic and gabaergic synaptic strength. *Neuron*, 49(4):563–575, 2006.

GOODMAN, Dan, BRETTE, Romain. Brian: a simulator for spiking neural networks in python. *Frontiers in neuroinformatics*, **2**, 2008.

GOODMAN, Dan F. M., BRETTE, Romain. The brian simulator. *Frontiers in neuroscience*, **3**(2):192, 2009.

GRAY, Charles M., MCCORMICK, David A. Chattering cells: superficial pyramidal neurons contributing to the generation of synchronous oscillations in the visual cortex. *Science*, **274**(5284):109–113, 1996.

GREENBERG, Steven. *Computational models of auditory function*, volume **312**. Ios PressInc, 2001.

GREENWOOD, Donald D., MARUYAMA, Naoshige. Excitatory and inhibitory response areas of auditory neurons in the cochlear nucleus. *Journal of neurophysiology*, **28**(5):863–892, 1965.

GRIFFITHS, Timothy D., WARREN, Jason D. What is an auditory object? *Nature Reviews Neuroscience*, **5**(11):887–892, 2004.

HAMMARLUND, Per, EKEBERG, Örjan. Large neural network simulations on multiple hardware platforms. *Journal of computational neuroscience*, **5**(4): 443–459, 1998.

HANSEL, D, MATO, G. Existence and stability of persistent states in large neuronal networks. *Physical review letters*, 86(18):4175–4178, 2001.

HARRIS, Kenneth D., BARTHO, Peter, CHADDERTON, Paul, CURTO, Carina, ROCHA, Jaime, HOLLENDER, Liad, ITSKOV, Vladimir, LUCZAK, Artur, MARGUET, Stephan L., RENART, Alfonso. How do neurons work together? lessons from auditory cortex. *Hearing research*, **271**(1):37–53, 2011.

HERCULANO-HOUZEL, Suzana. The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in human neuroscience*, **3**, 2009.

HINES, Michael L., CARNEVALE, Nicholas T. The neuron simulation environment. *Neural computation*, **9**(6):1179–1209, 1997.

HINES, Michael L., CARNEVALE, Nicholas T. Neuron: a tool for neuroscientists. *The Neuroscientist*, **7**(2):123–135, 2001.

HODGKIN, Alan L. The local electric changes associated with repetitive action in a non-medullated axon. *The Journal of physiology*, **107**(2):165–181, 1948.

HODGKIN, Alan L., HUXLEY, Andrew F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, **117**(4):500, 1952.

HOUWELING, Arthur R., BAZHENOV, Maxim, TIMOFEEV, Igor, STERIADE, Mircea, SEJNOWSKI, Terrence J. Homeostatic synaptic plasticity can explain post-traumatic epileptogenesis in chronically isolated neocortex. *Cerebral Cortex*, **15**(6):834–845, 2005.

HUANG, Camillan L., WINER, Jeffery A. Auditory thalamocortical projections in the cat: laminar and areal patterns of input. *Journal of Comparative Neurology*, **427**(2):302–331, 2000.

INITIATIVE, NEST. Nest, 2013. URL `http://www.nest-initiative.org/index.php/Software:Installation`.

IZHIKEVICH, Eugene M. Simple model of spiking neurons. *Neural Networks, IEEE Transactions on*, **14**(6):1569–1572, 2003.

IZHIKEVICH, Eugene M. Which model to use for cortical spiking neurons? *Neural Networks, IEEE Transactions on*, **15**(5):1063–1070, 2004.

IZHIKEVICH, Eugene M. Polychronization: Computation with spikes. *Neural computation*, **18**(2):245–282, 2006.

IZHIKEVICH, Eugene M. Solving the distal reward problem through linkage of stdp and dopamine signaling. *Cerebral Cortex*, **17**(10):2443–2452, 2007.

IZHIKEVICH, Eugene M., EDELMAN, Gerald M. Large-scale model of mammalian thalamocortical systems. *Proceedings of the national academy of sciences*, **105**(9):3593–3598, 2008.

IZHIKEVICH, Eugene M., GALLY, Joseph A., EDELMAN, Gerald M. Spike-timing dynamics of neuronal groups. *Cerebral Cortex*, **14**(8):933–944, 2004a.

IZHIKEVICH, Eugene M, GALLY, Joseph A, EDELMAN, Gerald M. Spike-timing dynamics of neuronal groups. *Cerebral Cortex*, 14(8):933–944, 2004b.

JOLIVET, Renaud, RAUCH, Alexander, LüSCHER, Hans-Rudolf, GERSTNER, Wulfram. Predicting spike timing of neocortical pyramidal neurons by simple threshold models. *Journal of computational neuroscience*, **21**(1):35–49, 2006.

KAWAGUCHI, Yasuo. Physiological subgroups of nonpyramidal cells with specific morphological characteristics in layer ii/iii of rat frontal cortex. *The Journal of neuroscience*, **15**(4):2638–2655, 1995.

KAWAGUCHI, Yasuo, KUBOTA, Yoshiyuki. Gabaergic cell subtypes and their synaptic connections in rat frontal cortex. *Cerebral Cortex*, **7**(6):476–486, 1997.

KIMURA, Akihisa, DONISHI, Tomohiro, SAKODA, Takema, HAZAMA, Michio, TAMAI, Yasuhiko. Auditory thalamic nuclei projections to the temporal cortex in the rat. *Neuroscience*, **117**(4):1003–1016, 2003.

KNIGHT, Bruce W. Dynamics of encoding in a population of neurons. *The Journal of General Physiology*, **59**(6):734–766, 1972.

KOELSCH, Stefan, SIEBEL, Walter A. Towards a neural basis of music perception. *Trends in cognitive sciences*, **9**(12):578–584, 2005.

KOTAK, Vibhakar C., FUJISAWA, Sho, LEE, Fanyee Anja, KARTHIKEYAN, Omkar, AOKI, Chiye, SANES, Dan H. Hearing loss raises excitability in the auditory cortex. *The Journal of neuroscience*, **25**(15):3908–3918, 2005.

KUWADA, Shigeyuki, STANFORD, Terrence R., BATRA, Ranjan. Interaural phase-sensitive units in the inferior colliculus of the unanesthetized rabbit: effects of changing frequency. *Journal of neurophysiology*, **57**(5):1338–1360, 1987.

LAPICQUE, Louis. Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. *J. Physiol. Pathol. Gen*, **9**(1):620–635, 1907.

LARSON, Eric, BILLIMORIA, Cyrus P., SEN, Kamal. A biologically plausible computational model for auditory object recognition. *Journal of neurophysiology*, **101**(1):323–331, 2008.

LARSON, Eric, PERRONE, Ben P., SEN, Kamal, BILLIMORIA, Cyrus P. A robust and biologically plausible spike pattern recognition network. *The Journal of Neuroscience*, **30**(46):15566–15572, 2010.

LE NOVERE, Nicolas, SHIMIZU, Thomas Simon. Stochsim: modelling of stochastic biomolecular processes. *Bioinformatics*, **17**(6):575–576, 2001.

LEVITT, Harry. *Compression amplification*, pages 153–183. Springer, 2004.

LEVY, Robert B., REYES, Alex D. Spatial profile of excitatory and inhibitory synaptic connectivity in mouse primary auditory cortex. *The Journal of Neuroscience*, **32**(16):5609–5619, 2012.

LOEBEL, Alex, NELKEN, Israel, TSODYKS, Misha. Processing of sounds by population spikes in a model of primary auditory cortex. *Frontiers in neuroscience*, **1**(1):197, 2007.

MALMIERCA, Manuel S, HACKETT, Troy A. Structural organization of the ascending auditory pathway. *The Auditory Brain*, pages 9–41, 2010.

MARKRAM, Henry. The blue brain project. *Nature Reviews Neuroscience*, **7**(2):153–160, 2006.

MARKRAM, Henry, TOLEDO-RODRIGUEZ, Maria, WANG, Yun, GUPTA, Anirudh, SILBERBERG, Gilad, WU, Caizhi. Interneurons of the neocortical inhibitory system. *Nature Reviews Neuroscience*, **5**(10):793–807, 2004.

MARTIN, Stephen J., GRIMWOOD, Paul D., MORRIS, Richard G. M. Synaptic plasticity and memory: an evaluation of the hypothesis. *Annual review of neuroscience*, **23**(1):649–711, 2000.

MATOUSEK, Jiri, NESETRIL, Jaroslav. *Invitation to discrete mathematics*. Oxford University Press, 1998.

MATSUMOTO, Makoto, NISHIMURA, Takuji. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, **8**(1):3–30, 1998.

MCCOMBE, Andrew, BAGULEY, David, COLES, Ross, MCKENNA, Laurence, MCKINNEY, Catherina, WINDLE−TAYLOR, Paul. Guidelines for the grading of tinnitus severity: the results of a working group commissioned by the british association of otolaryngologists, head and neck surgeons, 1999. *Clinical Otolaryngology & Allied Sciences*, **26**(5):388–393, 2001.

MCFADDEN, Dennis. *Tinnitus: facts, theories, and treatments*. National Academies Press, 1982.

MEDDIS, Ray. *Computational models of the auditory system*, volume **35**. Springer, 2010.

MEHRING, Carsten, HEHL, Ulrich, KUBO, Masayoshi, DIESMANN, Markus, AERTSEN, Ad. Activity dynamics and propagation of synchronous spiking in locally connected random networks. *Biological cybernetics*, **88**(5):395–408, 2003.

MENDELSON, J. R., CYNADER, M. S. Sensitivity of cat primary auditory cortex (al) neurons to the direction and rate of frequency modulation. *Brain research*, **327**(1):331–335, 1985.

MESGARANI, Nima, FRITZ, Jonathan, SHAMMA, Shihab. A computational model of rapid task-related plasticity of auditory cortical receptive fields. *Journal of computational neuroscience*, **28**(1):19–27, 2009.

METHERATE, Raju, ARAMAKIS, V. Bess. Intrinsic electrophysiology of neurons in thalamorecipient layers of developing rat auditory cortex. *Developmental brain research*, **115**(2):131–144, 1999.

MITANI, Akira, SHIMOKOUCHI, Minoru. Neuronal connections in the primary auditory cortex: an electrophysiological study in the cat. *Journal of Comparative Neurology*, **235**(4):417–429, 1985.

MORRISON, Abigail, MEHRING, Carsten, GEISEL, Theo, AERTSEN, A. D., DIESMANN, Markus. Advancing the boundaries of high-connectivity network simulation with distributed computing. *Neural computation*, **17**(8):1776–1801, 2005.

MORRISON, Abigail, STRAUBE, Sirko, PLESSER, Hans Ekkehard, DIESMANN, Markus. Exact subthreshold integration with continuous spike times

in discrete-time neural network simulations. *Neural Computation*, **19**(1):47–79, 2007.

MORRISON, Abigail, DIESMANN, Markus, GERSTNER, Wulfram. Phenomenological models of synaptic plasticity based on spike timing. *Biological cybernetics*, **98**(6):459–478, 2008.

MUREşAN, Raul C., IGNAT, Iosif. The"neocortex"neural simulator. a modern design. In *International Conference on Intelligent Engineering Systems, Cluj-Napoca, Romania*, 2004.

MURESAN, Raul C., SAVIN, Cristina. Resonance or integration? self-sustained dynamics and excitability of neural microcircuits. *Journal of neurophysiology*, **97**(3):1911–1930, 2007.

NAGESWARAN, Jayram Moorkanikara, DUTT, Nikil, KRICHMAR, Jeffrey L., NICOLAU, Alex, VEIDENBAUM, Alexander V. A configurable simulation environment for the efficient simulation of large-scale spiking neural networks on graphics processors. *Neural Networks*, **22**(5-6):791–800, 2009.

NATSCHLäGER, Thomas, MARKRAM, Henry, MAASS, Wolfgang. *Computer models and analysis tools for neural microcircuits*, pages 123–138. Springer, 2003.

NELKEN, Israel. Processing of complex stimuli and natural scenes in the auditory cortex. *Current opinion in neurobiology*, **14**(4):474–480, 2004.

NELKEN, Israel, FISHBACH, Alon, LAS, Liora, ULANOVSKY, Nachum, FARKAS, Dina. Primary auditory cortex of cats: feature detection or something else? *Biological cybernetics*, **89**(5):397–406, 2003.

NELKEN, Israel, BIZLEY, Jennifer K., NODAL, Fernando R., AHMED, Bashir, SCHNUPP, Jan W. H., KING, Andrew J. Large−scale organization of ferret auditory cortex revealed using continuous acquisition of intrinsic optical signals. *Journal of neurophysiology*, **92**(4):2574–2588, 2004.

NINI, Asaph, FEINGOLD, Ariela, SLOVIN, Hamutal, BERGMAN, Hagai. Neurons in the globus pallidus do not show correlated activity in the normal monkey, but phase-locked oscillations appear in the mptp model of parkinsonism. *Journal of Neurophysiology*, **74**(4):1800–1805, 1995.

NORDLIE, Eilen, GEWALTIG, Marc-Oliver, PLESSER, Hans Ekkehard. Towards reproducible descriptions of neuronal network models. *PLoS Computational Biology*, 5(8):e1000456, 2009.

NääTäNEN, Risto, TERVANIEMI, Mari, SUSSMAN, Elyse, PAAVILAINEN, Petri, WINKLER, István. 'primitive intelligence'in the auditory cortex. *Trends in neurosciences*, **24**(5):283–288, 2001.

NUNEZ, Paul L., SRINIVASAN, Ramesh. *Electric fields of the brain: the neurophysics of EEG*. Oxford University Press, USA, 2006.

OHL, Frank W., SCHEICH, H., FREEMAN, W. J. Change in pattern of ongoing cortical activity with auditory category learning. *Nature*, **412**(6848):733–736, 2001.

OJIMA, Hisayuki. Terminal morphology and distribution of corticothalamic fibers originating from layers 5 and 6 of cat primary auditory cortex. *Cerebral Cortex*, **4**(6):646–663, 1994.

ORACLE, Corporation. Netbeans ide, 2013. URL `http://netbeans.org/`.

OSWALD, Anne-Marie M., REYES, Alex D. Maturation of intrinsic and synaptic properties of layer 2/3 pyramidal neurons in mouse auditory cortex. *Journal of neurophysiology*, **99**(6):2998–3008, 2008.

OTAZU, Gonzalo H., LEIBOLD, Christian. A corticothalamic circuit model for sound identification in complex scenes. *PloS one*, **6**(9):e24270, 2011.

PECEVSKI, Dejan. Parallel neural circuit simulator (pcsim) - tutorial, 2008. URL `www.lsm.tugraz.at/pcsim/tutorial/FIAS%20PCSIM%20tutorial%203.ppt`.

PECEVSKI, Dejan, NATSCHLäGER, Thomas, SCHUCH, Klaus. Pcsim: a parallel simulation environment for neural circuits fully integrated with python. *Frontiers in neuroinformatics*, **3**, 2009.

PEIERLS, Tim, GOETZ, Brian, BLOCH, Joshua, BOWBEER, Joseph, LEA, Doug, HOLMES, David. *Java concurrency in practice*. Addison-Wesley Professional, 2005.

PERCHA, Bethany, DZAKPASU, Rhonda, ŻOCHOWSKI, Michał, PARENT, Jack. Transition from local to global phase synchrony in small world neural network and its possible implications for epilepsy. *Physical Review E*, **72**(3): 031909, 2005.

PHILLIPS, Dennis P., KELLY, Jack B. Coding of tone-pulse amplitude by single neurons in auditory cortex of albino rats (rattus norvegicus). *Hearing research*, **37**(3):269–279, 1989.

PHOKA, Elena, WILDIE, Mark, SCHULTZ, Simon R., BARAHONA, Mauricio. Sensory experience modifies spontaneous state dynamics in a large-scale barrel cortical model. *Journal of computational neuroscience*, **33**(2):323–339, 2012.

POLSTER, Michael R., ROSE, Sally B. Disorders of auditory processing: evidence for modularity in audition. *Cortex*, **34**(1):47–65, 1998.

PRESS, William H., TEUKOLSKY, Saul A., VETTERLING, William T., FLANNERY, Brian P. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

PRIETO, Jorge J., WINER, Jeffery A. Layer vi in cat primary auditory cortex: Golgi study and sublaminar origins of projection neurons. *Journal of Comparative Neurology*, **404**(3):332–358, 1999.

PURVES, Dale, AUGUSTINE, George J., FITZPATRICK, David, KATZ, Lawrence C., LAMANTIA, Anthony-Samuel, MCNAMARA, James O., WILLIAMS, S. Mark. *The Auditory Cortex*. Sinauer Associates, 2001.

REED, Russel D., MARKS, Robert J. Neural smithing: supervised learning in feedforward artificial neural networks, 1999.

RICHET, Yann. jmathplot, 2013. URL `https://code.google.com/p/jmathplot/`.

ROCHEL, Olivier, MARTINEZ, Dominique et al. An event-driven framework for the simulation of networks of spiking neurons. In *Proc. 11th European symposium on artificial neural networks*, pages 295–300, 2003.

ROMANSKI, Lizabeth M., LEDOUX, Joseph E. Organization of rodent auditory cortex: anterograde transport of pha-l from mgv to temporal neocortex. *Cerebral Cortex*, **3**(6):499–514, 1993.

ROSE, R. M., HINDMARSH, J. L. The assembly of ionic currents in a thalamic neuron i. the three-dimensional model. *Proceedings of the Royal Society of London. B. Biological Sciences*, **237**(1288):267–288, 1989.

ROSS, Deborah, CHOI, Jonathan, PURVES, Dale. Musical intervals in speech. *Proceedings of the National Academy of Sciences*, **104**(23):9852–9857, 2007.

ROSSUM, M. C. W. van. A novel spike distance. *Neural Computation*, **13**(4): 751–763, 2001.

ROTHSCHILD, Gideon, NELKEN, Israel, MIZRAHI, Adi. Functional organization and population dynamics in the mouse primary auditory cortex. *Nature neuroscience*, **13**(3):353–360, 2010.

RUSS, Brian E., LEE, Yune-Sang, COHEN, Yale E. Neural and behavioral correlates of auditory categorization. *Hearing research*, **229**(1):204–212, 2007.

SAKATA, Shuzo, HARRIS, Kenneth D. Laminar structure of spontaneous and sensory-evoked population activity in auditory cortex. *Neuron*, **64**(3):404–418, 2009.

SAKATA, Shuzo, HARRIS, Kenneth D. Laminar-dependent effects of cortical state on auditory cortical spontaneous activity. *Frontiers in neural circuits*, **6**, 2012.

SALLY, Sharon L., KELLY, Jack B. Organization of auditory cortex in the albino rat: sound frequency. *Journal of Neurophysiology*, **59**(5):1627–1638, 1988.

SCHAETTE, Roland, KEMPTER, Richard. Development of tinnitus−related neuronal hyperactivity through homeostatic plasticity after hearing loss: a computational model. *European Journal of Neuroscience*, **23**(11):3124–3138, 2006.

SCHIFF, Max L., REYES, Alex D. Characterization of thalamocortical responses of regular-spiking and fast-spiking neurons of the mouse auditory cortex in vitro and in silico. *Journal of Neurophysiology*, **107**(5):1476–1488, 2012.

SCHREINER, Christoph E., READ, Heather L., SUTTER, Mitchell L. Modular organization of frequency integration in primary auditory cortex. *Annual review of neuroscience*, **23**(1):501–529, 2000.

SCHUTTER, Erik De. *Computational modeling methods for neuroscientists*. The MIT Press, 2010.

SHARPEE, Tatyana O., ATENCIO, Craig A., SCHREINER, Christoph E. Hierarchical representations in the auditory cortex. *Current opinion in neurobiology*, **21**(5):761–767, 2011.

SJöSTRöM, Jesper, GERSTNER, Wulfram. Spike-timing dependent plasticity. *Spike-timing dependent plasticity*, page 35, 2010.

SONG, Sen, MILLER, Kenneth D., ABBOTT, Larry F. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nat Neurosci*, **3** (9):919–26, 2000.

STIEBLER, I., NEULIST, R., FICHTEL, I., EHRET, G. The auditory cortex of the house mouse: left-right differences, tonotopic organization and quantitative analysis of frequency representation. *Journal of Comparative Physiology A*, **181**(6):559–571, 1997.

STILES, Joel R., BARTOL, Thomas M. Monte carlo methods for simulating realistic synaptic microphysiology using mcell, 2001.

SUGA, Nobuo. Functional properties of auditory neurones in the cortex of echolocating bats. *The Journal of physiology*, **181**(4):671, 1965.

SUGA, Nobuo, MA, Xiaofeng. Multiparametric corticofugal modulation and plasticity in the auditory system. *Nature Reviews Neuroscience*, **4**(10):783–794, 2003.

SWEATT, J. David. *Mechanisms of memory*. Academic Press, 2010.

TALBOT, K. Human temporal lobe areas, 2011. URL `http://en.wikipedia.org/wiki/File:Human_temporal_lobe_areas.png`.

THEYEL, Brian B., LEE, Charles C., SHERMAN, S. Murray. Specific and nonspecific thalamocortical connectivity in the auditory and somatosensory thalamocortical slices. *Neuroreport*, **21**(13):861, 2010.

TIMOFEEV, Igor, GRENIER, Francois, BAZHENOV, Maxim, SEJNOWSKI, Terrence J., STERIADE, Mircea. Origin of slow cortical oscillations in deafferented cortical slabs. *Cerebral Cortex*, **10**(12):1185–1199, 2000.

TOMITA, Masaru, HASHIMOTO, Kenta, TAKAHASHI, Koichi, SHIMIZU, Thomas Simon, MATSUZAKI, Yuri, MIYOSHI, Fumihiko, SAITO, Kanako, TANIDA, Sakura, YUGI, Katsuyuki, VENTER, J. Craig. E-cell: software environment for whole-cell simulation. *Bioinformatics*, **15**(1):72–84, 1999.

TRAPPENBERG, Thomas P. *Fundamentals of computational neuroscience*. Oxford University Press, 2010.

TSODYKS, Misha V., SEJNOWSKI, Terrence. Rapid state switching in balanced cortical network models. *Network: Computation in Neural Systems*, **6**(2):111–124, 1995.

TUCKWELL, Henry C. Introduction to theoretical neuroscience, 1988.

TURRIGIANO, Gina G., LESLIE, Kenneth R., DESAI, Niraj S., RUTHERFORD, Lana C., NELSON, Sacha B. Activity-dependent scaling of quantal amplitude in neocortical neurons. *Nature*, **391**(6670):892–896, 1998.

VIDA, Imre, BARTOS, Marlene, JONAS, Peter. Shunting inhibition improves robustness of gamma oscillations in hippocampal interneuron networks by homogenizing firing rates. *Neuron*, **49**(1):107–118, 2006.

WANG, Xiao-Jing. Persistent neural activity: experiments and theory. *Cerebral Cortex*, 13(11):1123–1123, 2003.

WANG, Xiaoqin, LU, Thomas, SNIDER, Ross K., LIANG, Li. Sustained firing in auditory cortex evoked by preferred stimuli. *Nature*, **435**(7040):341–346, 2005.

WATSON, Charles. *The mouse nervous system*. Academic Press, 2012.

WEINBERGER, N. M., ASHE, J. H., METHERATE, R., MCKENNA, T. M., DIAMOND, D. M., BAKIN, J. Retuning auditory cortex by learning: A preliminary model of receptive field plasticity. *Concepts Neurosci*, **1**(1):91–132, 1990.

WENDYKIER, Piotr. Parallel colt, 2013. URL `https://sites.google.com/site/piotrwendykier/software/parallelcolt`.

WENDYKIER, Piotr, NAGY, James G. Parallel colt: a high-performance java library for scientific computing and image processing. *ACM Transactions on Mathematical Software (TOMS)*, **37**(3):31, 2010.

WIKIPEDIA.ORG. Synapse illustration2 tweaked, 2006. URL `http://commons.wikimedia.org/wiki/File:Synapse_Illustration2_tweaked.svg`.

WIKIPEDIA.ORG. Gray756, 2007a. URL `http://commons.wikimedia.org/wiki/File:Gray756.png`.

WIKIPEDIA.ORG. Complete neuron cell diagram en, 2007b. URL `http://en.wikipedia.org/wiki/File:Complete_neuron_cell_diagram_en.svg`.

WIKIPEDIA.ORG. Brain surface gyri, 2007c. URL `http://en.wikipedia.org/wiki/File:Brain_Surface_Gyri.SVG`.

WILSON, Blake S. Engineering design of cochlear implants. *Springer Handbook of Auditory Research*, **20**:14–52, 2004.

WILSON, Hugh R. Simplified dynamics of human and mammalian neocortical neurons. *Journal of theoretical biology*, **200**(4):375–388, 1999.

WILSON, Hugh R., COWAN, Jack D. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical journal*, **12**(1):1–24, 1972.

WINER, Jeffery A. Decoding the auditory corticofugal systems. *Hearing research*, **212**(1):1–8, 2005.

WINER, Jeffery A., LEE, Charles C. The distributed auditory cortex. *Hearing research*, **229**(1-2):3, 2007.

WINER, Jeffery A., PRIETO, Jorge J. Layer v in cat primary auditory cortex (ai): cellular architecture and identification of projection neurons. *Journal of Comparative Neurology*, **434**(4):379–412, 2001.

WINKOWSKI, Daniel E., KANOLD, Patrick O. Laminar transformation of frequency organization in auditory cortex. *The Journal of Neuroscience*, **33** (4):1498–1508, 2013.

WORGOTTER, Florentin, KOCH, Christof. A detailed model of the primary visual pathway in the cat: comparison of afferent excitatory and intracortical inhibitory connection schemes for orientation selectivity. *The Journal of neuroscience*, **11**(7):1959–1979, 1991.

WU, Guangying K., TAO, Huizhong W., ZHANG, Li I. From elementary synaptic circuits to information processing in primary auditory cortex. *Neuroscience & Biobehavioral Reviews*, **35**(10):2094–2104, 2011.

YOUNG, Eric D., BROWNELL, William E. Responses to tones and noise of single cells in dorsal cochlear nucleus of unanesthetized cats. *Journal of neurophysiology*, **39**(2):282–300, 1976.

ZHANG, Li I, BAO, Shaowen, MERZENICH, Michael M. Persistent and specific influences of early acoustic environments on primary auditory cortex. *Nature neuroscience*, 4(11):1123–1130, 2001.

ZHANG, Li I., BAO, Shaowen, MERZENICH, Michael M. Disruption of primary auditory cortex by synchronous auditory inputs during a critical period. *Proceedings of the National Academy of Sciences*, **99**(4):2309–2314, 2002.

ZHANG, Li I., TAN, Andrew Y. Y., SCHREINER, Christoph E., MERZENICH, Michael M. Topography and synaptic shaping of direction selectivity in primary auditory cortex. *Nature*, **424**(6945):201–205, 2003.

ZHOU, Yi, MESIK, Lukas, SUN, Yujiao J., LIANG, Feixue, XIAO, Zhongju, TAO, Huizhong W., ZHANG, Li I. Generation of spike latency tuning by thalamocortical circuits in auditory cortex. *The Journal of Neuroscience*, **32** (29):9969–9980, 2012.

ZHU, Yinghua, ZHU, J. Julius. Rapid arrival and integration of ascending sensory information in layer 1 nonpyramidal neurons and tuft dendrites of layer 5 pyramidal neurons of the neocortex. *The Journal of neuroscience*, **24**(6): 1272–1279, 2004.

ZILANY, Muhammad S. A., BRUCE, Ian C. Predictions of speech intelligibility with a model of the normal and impaired auditory-periphery. In *Neural Engineering, 2007. CNE'07. 3rd International IEEE/EMBS Conference on Neural Engineering*, pages 481–485. IEEE, 2007.

ZIV, Israel, BAXTER, Douglas A., BYRNE, John H. Simulator for neural networks and action potentials: description and application. *Journal of neurophysiology*, **71**(1):294–308, 1994.

ZUCKER, Robert S., REGEHR, Wade G. Short-term synaptic plasticity. *Annual review of physiology*, **64**(1):355–405, 2002.

List of Tables

Attachments

The thesis contains two text attachments:

1. The User Documentation
2. The Technical Documentation