Reproducing network dynamics of a neocortical microcircuit with point-neuron models

Master Thesis

Author(s): Bucher, Stefan F.

Publication date: 2013

Permanent link: https://doi.org/10.3929/ethz-a-010039180

Rights / license: In Copyright - Non-Commercial Use Permitted
REPRODUCING NETWORK DYNAMICS OF A NEOCORTICAL MICROCIRCUIT WITH POINT-NEURON MODELS

STEFAN F. BUCHER

MASTER’S THESIS, MARCH TO AUGUST 2013
MSC IN COMPUTATIONAL SCIENCE AND ENGINEERING
DEPARTMENT OF MATHEMATICS, ETH ZURICH

SUPERVISORS
Prof. Dr. Henry Markram
Dr. Marc-Oliver Gewaltig
Prof. Dr. Felix Schürmann
Blue Brain Project, EPFL

EXPERT
Prof. Dr. Ruedi Stoop
Institute for Neuroinformatics,
ETH and University of Zurich

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Blue Brain Project
École Polytechnique Fédérale de Lausanne
© Stefan F. Bucher, August 2013: Reproducing Network Dynamics of a Neocortical Microcircuit with Point-Neuron Models, MSc Thesis, ETH Zurich.

Correspondence should be addressed to: web@stefan-bucher.ch

The title page shows a visualization of the detailed model (left, © Blue Brain Project) and of a point-neuron column (right, Erő 2013a).
ABSTRACT

Recent attempts to create large-scale computer models of the brain rely on morphologically detailed Hodgkin-Huxley type multi-compartment models (Markram 2006). Due to the large complexity of these models, simulations require supercomputers such as an IBM Blue Gene. Applications where runtimes close to real time are desirable require simpler models however. This thesis focuses on networks of point-neurons as a simplified model to replicate the behaviour of a detailed simulation of a neocortical microcircuit.

A unifying formulation for integrate-and-fire models is introduced that improves the comparability of their dynamics and biophysical interpretation. A new numerical solution of the adaptive exponential integrate-and-fire model was developed to reduce the runtime of the simulation while improving accuracy.

An automated approach is introduced to set up a point-neuron simulation congruent to a morphologically detailed simulation and to align it by tuning its neuron model parameters. This is achieved by a novel fitting approach that determines single neuron parameters based on the global network model performance. While convergence of the optimization routine is difficult to achieve in such a parameter space, results show that the approach is capable of simplifying a complex model.

Keywords: Point-Neuron Network, Adaptive Exponential Integrate-and-Fire (AdEx) Model, Model Fitting, Network Dynamics, Neocortical Microcircuit.
## CONTENTS

1 INTRODUCTION ........................................ 1
   1.1 Mathematical Modelling of Nerve Cells ...... 2
   1.2 Simulation of a Neuronal Network ........ 5

2 DYNAMICS OF PHENOMENOLOGICAL NEURON MODELS .... 9
   2.1 Review of Integrate-and-Fire Models ........ 9
      2.1.1 (Leaky) Integrate-and-Fire Model .... 9
      2.1.2 Resonate-and-Fire Model ............ 10
      2.1.3 Quadratic Integrate-and-Fire Model ... 10
      2.1.4 Exponential Integrate-and-Fire Model .. 10
      2.1.5 Izhikevich Model .................... 11
      2.1.6 Adaptive Exponential Integrate-and-Fire Model 11
   2.2 Unifying Model Formulation ............... 12
   2.3 Analysis of Systems Dynamics ............... 15
      2.3.1 Voltage Traces ....................... 16
      2.3.2 Phase Portraits ..................... 18
      2.3.3 Equilibria and Bifurcations ........ 18
      2.3.4 Linearization and Stability Analysis ... 21
      2.3.5 Linear Subthreshold Regime .......... 22

3 NUMERICAL INTEGRATION OF ADEX MODEL ODES .... 25
   3.1 Problem Formulation .......................... 25
   3.2 Runge-Kutta Single Step Method ............. 29
   3.3 Performance and Accuracy Tests ............. 31
   3.4 Semianalytical Approach .................... 32

4 REPLICA TING A MICROCIRCUIT MODEL WITH POINT-NEURONS 35
   4.1 Reduction to Point-Neuron Simulation ....... 35
      4.1.1 Reducing the Number of Populations ... 36
      4.1.2 Synaptic Connectivity ................ 38
      4.1.3 Replicating the Experiment Protocol ... 42
      4.1.4 Initial Conditions .................... 42
   4.2 Calibrating Model Parameters ............... 43
      4.2.1 Network-Level Optimization of Point-Neuron Parameters ....... 43
      4.2.2 Spike Statistics as a Measure for Network Dynamics ........... 45
      4.2.3 Optimisation Algorithm ............... 45
   4.3 Computational Considerations ............... 46
   4.4 The Simulation to Be Replicated ............ 47

5 RESULTS AND DISCUSSION .......................... 49

6 CONCLUSION ......................................... 55

BIBLIOGRAPHY ........................................ 56

A FITTED PARAMETERS ................................. 63
B  UPDATE FORMULAE FOR THE SEMIANALYTICAL SOLUTION OF ADEX 67
C  NEST-IMPLEMENTATION OF NUMERICAL SOLUTION OF ADEX 69
   C.1  Test Script ........................................... 69
   C.2  Update Function from aeif_cond_alpha_RK5.cpp ... 70
D  OPONC CODE 79
   D.1  main.py ............................................ 79
   D.2  simulation.py ...................................... 91
   D.3  bbsim.py ........................................... 111
E  OPONC DOCUMENTATION 115
LIST OF FIGURES

Figure 1.1 Visualization of a Neuron .................... 2
Figure 1.2 Schematic Illustration of a Neuron .......... 3
Figure 1.3 Schematic Illustration of Multi-Compartment Model .................................. 4
Figure 1.4 Visualization of a Network of Neurons ...... 5
Figure 1.5 e-Types ........................................ 6
Figure 2.1 Experimental Voltage Trace .................. 16
Figure 2.2 Right-Hand Side for \( w = 0 \) \((V)=\text{mV})\) ........ 17
Figure 2.3 Voltage Traces of the Models ................ 17
Figure 2.4 Phase Portraits ................................ 19
Figure 2.5 Derivative Landscapes ......................... 20
Figure 2.6 Bifurcation Diagrams ......................... 21
Figure 2.7 Approximations at Equilibria ................. 23
Figure 2.8 Convergence towards Subthreshold Regime . 24
Figure 3.1 Sample voltage trace of a neuron receiving spikes 26
Figure 3.2 \( \alpha \)-Function as Synaptic Conductance Kernel .. 27
Figure 3.3 Accuracy of New Numerical Implementation . 32
Figure 4.1 Morphologically Detailed and Simplified Point-Neuron Column ...................... 36
Figure 4.2 Size of different populations .................. 38
Figure 4.3 Synaptic Conductance Matrix ................ 39
Figure 4.4 Change in Network Behaviour Due to Weight Scaling ................................... 41
Figure 5.1 Mean Firing Rates per Population .......... 50
Figure 5.2 Layer 5 Pyramidal Cell ....................... 51
Figure 5.3 Pyramidal Cell within the Column ............ 52
Figure 5.4 Zoomed Raster Plot of Simplified Simulation . 53
Figure 5.5 Raster Plots of Full and Simplified Simulation .... 54
Figure A.1 Fitted Threshold Potential Values ............ 65
Figure A.2 Fitted Leak Conductance Values .............. 66

LIST OF TABLES

Table 2.1 Neurocomputational Features ................ 12
Table 2.2 Parameter Correspondence of Point-Neuron Models ........................................ 14
Table 2.3 Corresponding Parameter Sets .................. 15
Table 3.1 Butcher Schemes of Runge-Kutta Methods .... 29
Table 3.2 Performance Comparison of New Numerical Implementation ............................. 33
Table 4.1 Type Mapping .................................... 37
AdEx  Adaptive Exponential Integrate-and-Fire Model
AMPA  Alpha-amino-3-hydroxy-5-methyl-4-isoxazolepropionic Acid
ATP   Adenosine Triphosphate
BG/Q  Blue Gene/Q (IBM supercomputer)
CPU   Central Processing Unit
dNAC  Delayed Firing, Non-accommodating
dSTUT Delayed Firing, Stuttering
e-type Electrical Type
GABA  Gamma-Aminobutyric Acid
GSL   GNU Scientific Library
HDF5  Hierarchical Data Format 5
LIF   Leaky Integrate-and-Fire
m-type Morphology Type
me-type combination of an m- and an e-type
Na+/K+-ATPase Sodium-Potassium Adenosine Triphosphatase
NEST  NEural Simulation Tool
NMDA  N-Methyl-D-Aspartate
ODE   Ordinary Differential Equation
OPoNC Optimizing Point-Neuron Column, software framework developed to find simplified version corresponding to a detailed model
P14-P16 Postnatal State (week 14-16)
PSP   Postsynaptic Potential
RAM   Random Access Memory
RK    Runge-Kutta Method (Single Step Numerical ODE Integrator)
I would like to express my gratitude to Prof. Dr. Henry Markram and Prof. Dr. Felix Schürmann for welcoming me in the Blue Brain Project, and my great appreciation for the advice and assistance given by Dr. Marc-Oliver Gewaltig, who was an extremely committed supervisor responsible for making this project such an invaluable learning experience.

My special thanks are extended to Prof. Dr. Ruedi Stoop, the expert of my university who enabled me to carry out this project at another university.

I wish to acknowledge the assistance provided by Werner van Geit, Giuseppe Chindemi, Dr. Eilif Muller, Dr. Srikanth Ramaswamy, and Dr. Athanassia Chalimourda that was always very valuable and greatly appreciated.

Finally, I would like to thank my fellow students Csaba Erő and Alex Antrobus for the very fruitful collaboration and helpful discussions.
INTRODUCTION

One of the major scientific challenges of the 21st century is to understand how brains work and in particular how they allow phenomena like intelligence, memory, learning or what we call “mind”. Much has been described of the neural anatomy and physiology throughout the last decades and it seems that nerve cells communicating by the exchange of electrical signals form the basis for the functioning of the brain. And yet, neuroscience is still lacking a comprehensive theory of how information is represented and processed in networks of neurons. While much is known about brain areas, as well as of neurons, the brain’s complexity seems prohibitive to link abstract, higher level abilities to a neural correlate. Recent progress in information technology moves it to the realm of the feasible to conduct computer simulations with mathematical models, giving hope that the process of constructing models and testing them against experimental evidence will eventually lead to a better understanding.

A major effort in this direction is undertaken at the École Polytechnique Fédérale de Lausanne by the Blue Brain Project (and its EU FET (Future and Emerging Technologies) Flagship Initiative “Human Brain Project”) with the goal of creating a computer model of the human brain (Markram 2006). An important milestone towards this ambitious goal has been achieved with the creation of a model of a rat’s neocortical column, a part of the neocortex containing in the order of 10'000 neurons and believed to form the basic functional unit of cerebral cortex, a view that is contested however (Horton and Adams 2005). One might thus prefer to speak of a local neocortical microcircuit.

While the simulation is very accurate, its computational complexity is prohibitively high for the use in applications that require simulation close to real time, as is the case in neurorobotics, for instance: One goal of the recently approved FET Flagship Human Brain Project is to couple brain models to robots in order to study learning behaviour in a realistic setting. It is therefore of interest to find a simplified version of the model that may be simulated considerably faster while maintaining the most important properties of the detailed model. One approach - the use of phenomenological neuron models in a network simulation - is subject of this Master’s Thesis submitted for the MSc of ETH Zurich in Computational Science and Engineering.

In the following, we introduce the foundations for the three main parts of the thesis: Chapter 2 investigates the mathematical properties of the phenomenological neuron models used in this study. In chapter
3, the numerical solution of these neuron models is discussed. Chapter 4 describes the methodology of properly setting up such a simulation and how to align it with a detailed simulation by calibrating its model parameters. The results of this parameter fitting are presented and discussed in chapter 5.

### 1.1 Mathematical Modelling of Nerve Cells

The human brain consists of approximately $10^{11}$ neurons that are connected by about $10^{15}$ synapses and communicate information by brief electrical pulses. Nerve cells exhibit an electrical potential across their cell membrane due to different ion concentrations in the intracellular and extracellular medium (surplus of K$^+$ in the cell and of Na$^+$ and Cl$^-$ outside, resulting in a negative polarization) that are maintained under consumption of ATP by Na$^+$/K$^+$-ATPase.

*Electrophysiology of Neurons*

If the membrane potential exceeds a critical value, voltage-gated ion channels start a feedback loop which leads to a rapid exponential rise of the membrane potential, followed by an equally fast return of the membrane potential to its resting value. These electrical pulses (so-called action potentials) propagate from the cell soma to the axon, at whose end they are transmitted to other neurons: At the synapse to a recipient neuron, an action-potential will cause the release of neurotransmitters from vesicles into the synaptic cleft. At the postsynaptic neuron, these transmitters open ion channels (via receptors), thereby

![Figure 1.1: Visualization of a neuron. © Blue Brain Project.](image)
leading to a change in postsynaptic potential. The postsynaptic potential of all incoming spikes is integrated in the postsynaptic neuron and may or may not cause it to emit a spike itself.

The electrophysiological processes can be quantitatively modelled since Hodgkin and Huxley (1952) by differential equations describing the course of the membrane voltage and the ion channel kinetics. This corresponds to describing the dynamics over the cell membrane as an electrical circuit, where the membrane (with capacitance $C$) with the sodium-potassium pump maintaining the resting potential is represented by a voltage source, and different ion currents (and a leak current, respectively) by parallel branches consisting of a capacitor (corresponding to the equilibrium or reversal potential $E$ for the respective ion) and a resistor (corresponding to its conductance $g$).

If the gate variables $n, m, h$ - with time constants $\tau$ and stable states $n_\infty(V), m_\infty(V), h_\infty(V)$ - are introduced to model ion channel kinetics, this gives rise to the following four dimensional, classical formulation of the Hodgkin-Huxley model describing the time course of the membrane voltage $V$ under an external current $I$. $\bar{g}$ are maximal conductances.

\[ CV = I - \bar{g}_K n^4(V - E_K) - \bar{g}_Na m^3 h(V - E_Na) - \bar{g}_L(V - E_L)(1.1) \]

\[ \frac{\dot{n}}{n} = (n_\infty(V) - n)/\tau_n(V) \]  
\[ \frac{\dot{m}}{m} = (m_\infty(V) - m)/\tau_m(V) \]  
\[ \frac{\dot{h}}{h} = (h_\infty(V) - h)/\tau_h(V) \]
Biophysical models of this kind (many extensions and simplifications can be introduced) generally give an accurate description of the electrophysiology of a neuron. This spatial resolution can be accounted for with multicompartment models that divide a neuron into several cylindric compartments (Gerstner and Kistler 2012, chapter 2.6). This approach is used in the Blue Brain Project to model morphologically detailed neurons. The soma is a single compartment with Hodgkin-Huxley dynamics, whereas the compartments of the dendrites are passive. The flow of current between the compartments is modelled by cable theory (giving adequate boundary conditions). Axons are not explicitly represented in the model.

When one is to simulate not only a single neuron, but a network of neurons, the synaptic communication between neurons has to be taken into account in addition to the neuron model itself. This is done here with the Tsodyks-Markram model that is based on the assumption that synapses have resources (neurotransmitters available for release) at their disposal and keeps track of their fraction that is in the recovered ($R$), effective ($E$) or inactive ($I$) state (Markram and Tsodyks 1997; Fuhrmann et al. 2002): A presynaptic spike at time $t_{sp}$ activates (i.e. releases) the fraction $U_{SE}$ (utilization of synaptic efficacy) of resources available (i.e. in the recovered state) - they become effective. The effective resources gradually become inactive (with time constant $\tau_{in}$), and inactive resources are recovered (with time constant $\tau_{rec}$). The postsynaptic response $I_{postsyn}(t) \propto E(t)$ is proportional to the fraction of resources in the effective state.
The Blue Brain Project aims at reconstructing the neocortical column in a detailed model, based on these multi-compartment models of morphologically realistic neurons. It assumes that the neocortical column, or neocortical microcircuit, is an area of the neocortical cortex spanning all its six layers, i.e. 2 to 4 millimeters thick (Kandel et al. 1991, 324). The cerebral cortex plays an important role for cognitive functions and contains in the order of 4 million neurons for mice, 15 million for rats, and 11.5 billion for humans (Roth and Dicke 2005).

The construction of the cortical column in silico in the Blue Brain Project is based on systematically retrieved experimental data (from...
rats) like neuron morphologies, electrophysiological behaviour, and synapse kinetics. The different morphologies are classified into 55 so-called m-types (e.g. L6_BPC - bipolar cell in layer 6, cf. table 4.1). Similarly, cells are group into 11 so-called e-types (e.g. cAC - continuous firing with accommodation, cf. figure 1.5) according to their electrophysiological properties. By combining m- and e-types, one obtains a variety of 207 different neuron types (me-types). Each of the 31'000 cells is typically modelled with around 400 compartments.

![Electrophysiological Types of Interneurons](image)

Figure 1.5: Electrophysiological Types of Interneurons (and a Layer 5 Pyramidal Cell) used in Detailed Simulation (Druckmann et al. 2012). c - continuous firing, b - burst firing (at beginning of stimulus), d - delayed firing; AC - accommodating, NAC - non-accommodating, STUT - stuttering, IR - irregular firing.

The simulation is implemented in an adapted version of NEURON (Carnevale 2006). Simulating 3000 ms requires a runtime of about two hours on 256 sixteen-core nodes (Power A2, 1.6 GHz) of the IBM Blue Gene/Q supercomputer “Lemanicus” at EPFL. The simulation is, thus, about 1’000 times slower than real time.

The goal of the “Simplification Group” within the Blue Brain Project is thus to simplify this detailed simulation in a way that makes simulations close to real time feasible, while still matching the behaviour of the detailed simulation as closely as possible. Such a simplified model of the neural network of a neocortical column could then be used in applications like robotics to study the learning capabilities of a neocortical column. With this goal in mind, an important part of

1 http://bluegene.epfl.ch
the thesis is aimed at replicating the behaviour of the neocortical column model with a simplified simulation. While different approaches are worthy of consideration to achieve this (mean-field models, for instance), this project focused on the use of a network of point-neurons.

For the use in a simulation of a microcircuit of around 31,000 neurons, we will rely on phenomenological two-dimensional models of the integrate-and-fire type as discussed in chapter 2. These models are also called point-neuron models, as they abstract from the morphological and biophysical details and attempt to model the input-output relation of a neuron. Even though they are less accurate, phenomenological neuron models are preferred here to biophysically realistic models for their computational simplicity. I implemented a simulation of a network of point-neurons (Brette et al. 2007; Brunel 2000) in NEST (Gewaltig and Diesmann 2007), a simulation tool for spiking neural network models. The kernel of this software updates the state of the neurons according to the model equations and handles the spike communication between them. The latter is done for neurons emitting spikes by propagating the spike events to all postsynaptic neurons (where they are stored in a ring buffer accounting for the delay of the synaptic connection).

There are two major challenges in constructing a simplified simulation that replicates the behaviour of the morphologically detailed Blue Brain model: First, the precise design of the model and experimental protocols have to be extracted from the different configuration files and translated to a corresponding NEST specification. Second, a set of parameters has to be determined for the point-neuron model to best approximate the rich behaviour of high-dimensional multi-compartment models. Details are described in chapter 4. We based the fitting of neuron model parameters on a novel approach optimizing at the network rather than the single neuron level. The advantage of this approach is that it allows to automatically extract parameters from a detailed simulation; in contrast to many other approaches, it does not rely on a manually specified input protocol adapted to a certain regime. But before addressing these questions on how the neuron model of choice shall be tuned, we will first investigate the dynamics and numerics of the models used, in order to gain a deeper understanding of their characteristics.
In this chapter, the dynamics of two wide-spread formal spiking neuron models of the integrate-and-fire type shall be investigated, in the spirit of Izhikevich (2010). In contrast to Hodgkin-Huxley type models, phenomenological models do not attempt to describe the internal electrophysiological processes of a nerve cell, they rather content themselves with giving a description of the input-output relation of a “black box” neuron, i.e. to describe the membrane voltage given the incoming current. As they are often not applied to multicompartment models, they are sometimes called point-neuron models, where point-neuron refer to the fact that the morphology of a neuron is neglected and collapsed to a merely logical representation without any spatial extension.

2.1 Review of Integrate-and-Fire Models

An important class of formal spiking neuron models are so-called integrate-and-fire models. Their name refers to the fact that they compute the voltage by integrating the incoming current and emit a spike when the voltage exceeds a predefined threshold.

2.1.1 (Leaky) Integrate-and-Fire Model

In the simplest form, a spike is thus just an artificially defined logical event that consists of resetting the membrane voltage to a given value. This model goes back to Louis Lapicque (Lapicque 1907; Tuckwell 1988). The model consists of a linear ordinary differential equation (ODE) describing the membrane potential, and of a threshold condition for the generation of spikes. The membrane equation can be written as

\[ CV' = I - g_L(V - E_L) \] (2.1)

\( V \) is the membrane potential, \( I \) an external current, and \( C \) the membrane capacitance. The leak current accounts for the decay of a potential across the cell membrane due to leakage and is described by the leak conductance \( g_L \) and a leak reversal potential \( E_L \). A spike is emitted if \( V \geq V_{th} \), afterwards the voltage is reset to \( V = V_{reset} \).
2.1.2 Resonate-and-Fire Model

One extension of the leaky integrate-and-fire model is the resonate-and-fire model (Young 1937; Izhikevich 2001; Brunel et al. 2003; Richardson 2003) that can exhibit resonance and introduces an additional variable \( W \), corresponding to a low-threshold persistent \( K^+ \) current (Izhikevich 2010, 269). The model is defined as:

\[
\begin{align*}
\dot{V} &= I - g_L(V - V_{eq}) - W \\
\dot{W} &= (V - V_{eq})/k - W
\end{align*}
\]

\( V_{eq} \) is the equilibrium potential and \( k \) a constant. The threshold condition is specified as follows: If \( V \geq V_{th} \), then \( V = V_{reset} \) and \( W = W_{reset} \). With the addition of \( W \), it becomes possible to capture many dynamical properties observed in different neuron types.

2.1.3 Quadratic Integrate-and-Fire Model

Both, the integrate-and-fire, as well as the resonate-and-fire model, are linear and only model subthreshold dynamics - there is no action potential, the spike is added artificially. The quadratic integrate-and-fire model (Ermentrout and Kopell 1986; Kopell and Ermentrout 1986) includes a non-linear term to model the action potential. Then, there is just a “threshold” \( V_{peak} \) for the detection of a spike (but not the generation), where \( V \) is reset to \( V_{res} \). This is, thus, the simplest model of a spiking neuron with an explicit action potential. It is defined as:

\[
V = I + V^2
\]

We can write (\( k \) being a constant)

\[
\dot{V} = I - I_{\infty}(V)
\]

\[
I_{\infty}(V) = g_L(V - V_{rest}) - k(V - V_{th})^2
\]

2.1.4 Exponential Integrate-and-Fire Model

A more realistic action potential can be obtained if depolarizing the nonlinearity is exponential, leading to the following model (Fourcaud-Trocmé et al. 2003), where \( k \) and \( p \) are constants:

\[
\dot{V} = I - I_{\infty}(V)
\]

\[
I_{\infty}(V) = g_L(V - V_{rest}) - ke^{pV}
\]
2.1.5 Izhikevich Model

The so-called Izhikevich model (Izhikevich 2003) has a quadratic non-linearity, similar as seen before, but also an additional variable $w$ called recovery current (accounting for different slow currents).

\[ C \dot{V} = k(V - V_{\text{rest}})(V - V_{\text{th}}) - w + I \quad (2.9) \]
\[ \dot{w} = a_{Izh}(b_{Izh}(V - V_{\text{rest}}) - w) \quad (2.10) \]

Again, $V$ is the membrane voltage, $I$ the external current and $C$ the membrane capacitance. There is a parameter $V_{\text{th}}$ called instantaneous threshold potential and a resting potential $V_{\text{rest}}$. If $V \geq V_{\text{peak}}$, $V$ is reset to $c$ and $w$ incremented by $d$ ($w+ = d$). Again, there is no threshold for the spike generation, only a technical one ($V_{\text{peak}}$) for spike detection (a spike is theoretically defined as $V \to \infty$). The recovery time constant $a$, the parameter $b$ ($w$ amplifying if $b < 0$, otherwise resonating), the reset voltage $c$, and the net outwards currents activated during a spike ($d$), as well as $k$, are model parameters.

The model can also be written as:

\[ \dot{v} = I + v^2 - w \quad (2.11) \]
\[ \tau_w \frac{dw(t)}{dt} = a(V(t) - E_L) - w(t) \quad (2.12) \]

This illustrates (Izhikevich 2010, 273) that the model has only 4 independent, dimensionless parameters: $a$ and $b$ for the subthreshold dynamics, and $c$ and $d$ for after-spike transient behaviour. For $b = 0$, the model can be seen as quadratic integrate-and-fire model with adaptation. Without $v^2$, it is linear and equivalent to the resonate-and-fire model.

2.1.6 Adaptive Exponential Integrate-and-Fire Model

The Adaptive Exponential Integrate-and-Fire (AdEx) Model (Brette and Gerstner 2005, see also Clopath et al. 2007, Naud et al. 2008, Jolivet et al. 2004, and Jolivet et al. 2006) is similar to the Izhikevich model, but the nonlinearity is exponential here:

\[ C \frac{dV(t)}{dt} = -g_L(V(t) - E_L) + g_L \Delta_T \exp\left(\frac{V(t) - V_{\text{th}}}{\Delta_T}\right) - w \quad (2.13) \]
\[ \tau_w \frac{dw(t)}{dt} = a(V(t) - E_L) - w(t) \quad (2.14) \]

Again, there is an adaptation variable (that can generally be seen as modelling calcium-activated potassium channels). After a spike, $V$ is set to $V_{\text{reset}}$ and $w$ incremented by $b$. In addition to $C$ and $V_{\text{th}}$,
parameters comprise an adaptation time constant $\tau_w$, a sub-threshold adaptation parameter $a$, a slope factor $\Delta_T$, as well as $g_L$ and $E_L$ describing a leak current as above.

Table 2.1 lists the classes of neurocomputational features that can be observed in neurons (Izhikevich 2004, esp. figure 2). While most of the models listed above are only able to reproduce a subset of these features, the AdEx and Izhikevich model show all of them. They are thus the preferred point-neuron models for our purpose. For a detailed comparison, we refer to Izhikevich (2004).

### 2.2 Unifying Model Formulation

The Izhikevich model and the AdEx model are similar, but it is difficult to compare them directly, because the formulations presented above are not perfectly consistent, and model parameters do not necessarily mean the same. In the following, we therefore introduce a unifying formulation that contains both the Izhikevich and the AdEx model and thereby clarifies similarities and differences.

The formulation of our unifying model follows the AdEx model, as this seems to be closer to the leaky integrate-and-fire and conductance-based models and may thus be more interpretable for some. Both models can be expressed in the following form:

<table>
<thead>
<tr>
<th>Neurocomp. Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tonic Spiking</td>
</tr>
<tr>
<td>Phasic Spiking</td>
</tr>
<tr>
<td>Tonic Bursting</td>
</tr>
<tr>
<td>Phasic Bursting</td>
</tr>
<tr>
<td>Mixed Mode</td>
</tr>
<tr>
<td>Spike Frequency Adaptation</td>
</tr>
<tr>
<td>Class 1 Excitability</td>
</tr>
<tr>
<td>Class 2 Excitability</td>
</tr>
<tr>
<td>Spike Latency</td>
</tr>
<tr>
<td>Damped Subthreshold Oscillations</td>
</tr>
<tr>
<td>Resonator</td>
</tr>
<tr>
<td>Integrator</td>
</tr>
<tr>
<td>Rebound Spike</td>
</tr>
<tr>
<td>Rebound Burst</td>
</tr>
<tr>
<td>Threshold Variability</td>
</tr>
<tr>
<td>Bistability (“Switching Off”)</td>
</tr>
<tr>
<td>Depolarizing After-Potential</td>
</tr>
<tr>
<td>Self-Sustained Oscillations</td>
</tr>
<tr>
<td>Mixed Chatter/Class 1 Excitability</td>
</tr>
<tr>
<td>Purely Oscillatory Mode</td>
</tr>
<tr>
<td>Acommodation</td>
</tr>
<tr>
<td>Inhibition-Induced Spiking</td>
</tr>
<tr>
<td>Inhibition-Induces Bursting</td>
</tr>
</tbody>
</table>
\[
\begin{align*}
C_m \frac{dV(t)}{dt} &= F(V(t)) - g_L \cdot (V(t) - V_{\text{rest}}) - w(t) + I \quad (2.15) \\
\tau_w \frac{dw(t)}{dt} &= a \cdot (V(t) - V_{\text{rest}}) - w(t) \quad (2.16)
\end{align*}
\]

If \( V \) attains \( V_{\text{peak}} \), the two variables are reset according to \( V(t) = V_{\text{reset}} \) and \( w(t) = b \). Again, \( C_m \) is the membrane capacitance and \( \tau_w \) an adaptation time constant. The leak current is described by the leak conductance \( g_L \) and the reversal potential that is now called \( V_{\text{rest}} \) (resting potential). Observe that the membrane time constant \( \tau_m \) is contained in \( C_m \), as \( \tau_m = 1/g_L \cdot C_m = R \cdot C_m \) (time constant of an exponential process, as in Electrical Engineering with a first order RC circuit). As in the AdEx model, \( a \) is a model parameter describing subthreshold adaptation. \( F(V(t)) \) finally stands for the depolarizing nonlinearity describing the action potential that differs between the models.

The AdEx model can be expressed in this formulation straightforward, by using \( V_{\text{rest}} = E_L \) and

\[
F_{\text{AdEx}}(V) = g_L \Delta T \exp \left( \frac{(V - V_{\text{rest}}) - (V_{\text{th}} - V_{\text{rest}})}{\Delta T} \right) = g_L \Delta T \exp \left( \frac{V - V_{\text{th}}}{\Delta T} \right). \quad (2.17)
\]

In order to express the Izhikevich model in the same formalism, we have to rewrite the non-linear term:

\[
k(V - V_{\text{rest}})(V - V_{\text{th}}) = k(V^2 - (V_{\text{rest}} + V_{\text{th}}) \cdot V + V_{\text{rest}} \cdot V_{\text{th}})
\]

\[
= kV^2 - k(V_{\text{rest}} + V_{\text{th}}) \cdot V - V_{\text{rest}} \\
= kV^2 - kV^2_{\text{rest}} - k(V_{\text{rest}} + V_{\text{th}}) \cdot (V - V_{\text{rest}}) \quad (2.19)
\]

The nonlinearity is now split into a linear term of the form of a leak current and

\[
F_{\text{Izh}}(V) := kV^2 - kV^2_{\text{rest}}.
\]

Table 2.2 summarizes how different neuron models can be expressed in our formalism.

A common notation for the Izhikevich model is

\[
\begin{align*}
\frac{dV}{dt} &= 0.04V^2 - 5V + 140 - w + I \quad (2.20) \\
\frac{dw}{dt} &= a_{\text{Izh}}(b_{\text{Izh}}(V - V_{\text{rest}}) - w) \quad (2.21)
\end{align*}
\]

leaving us with far less parameters at first sight (only \( a, b, c, d \)). However, note that while the coefficients for the quadratic and linear term, as well as the constant term, have already been fixed in this
Table 2.2: Correspondence table indicating for the spiking neuron models introduced above the corresponding value for each of the following parameters of the unified model formulation: threshold potential $V_{\text{th}}$, resting potential $V_{\text{rest}}$, membrane capacitance $C_m$, leak conductance $g_L$, adaptation time constant $\tau_w$, subthreshold adaptation coefficient $a$, reset potential $V_{\text{reset}}$, after-spike adaptation update $b$, and nonlinearity $F(V)$.
In order to get to a better understanding of the role of different model parameters, we discuss some important aspects of the dynamics of our two models of choice (Izhikevich and AdEx model, as they exhibit all neurocomputational features). We do this relying on the unified model formulation introduced above that allows for a clear distinction of the linear subthreshold and the nonlinear spike dynamics. For a more detailed discussion, we refer to (Touboul 2008; Touboul and Brette 2008; Toubol 2007; Izhikevich 2010, 5.2.4, 8.1).

The following plots were all generated with the parameters presented in table 2.3 (unless varying within the plot); these correspond to those reported in (Izhikevich 2010, 274) and thus reproduce figure 8.6c (ibid., 275). The operating parameters of Izhikevich correspond to physically interpretable parameters in the unified model formulation, they are actually given by \( \frac{k}{C_m} \cdot \frac{1}{V_{rest} + V_{th}} \), i.e. there are three parameters more: \( V_{rest}, V_{th} \), and \( \frac{k}{C_m} \cdot \frac{1}{V_{rest} + V_{th}} \) - at the cost of a less obvious interpretation as a Hodgkin-Huxley type model. Units are fulfilled in this notation when assuming \( C_m = 1 \) and considering the fact that coefficients have units, as well.

### Table 2.3: Parameters for the Izhikevich Model (Izhikevich 2010, 274) with their Corresponding Value in the Unified Model Formulation, as well as AdEx Parameters (in Unified Model Formulation) Used for Comparable Plots.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Izhikevich</th>
<th>AdEx</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{th} )</td>
<td>= -40mV</td>
<td>= -40mV</td>
</tr>
<tr>
<td>( V_{rest} )</td>
<td>= -60mV</td>
<td>= -60mV</td>
</tr>
<tr>
<td>( C )</td>
<td>= 100pF</td>
<td>= 100pF</td>
</tr>
<tr>
<td>( k )</td>
<td>= 0.7</td>
<td>= 0.7</td>
</tr>
<tr>
<td>( g_{Izh} )</td>
<td>= -70nS</td>
<td>= 5nS</td>
</tr>
<tr>
<td>( \alpha_{Izh} )</td>
<td>= 0.03</td>
<td>= 0.03</td>
</tr>
<tr>
<td>( \tau_w )</td>
<td>= 33.3ms</td>
<td>= 33.3ms</td>
</tr>
<tr>
<td>( b_{Izh} )</td>
<td>= -2</td>
<td>= -2</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>= -2nS</td>
<td>= -2nS</td>
</tr>
<tr>
<td>( c_{Izh} )</td>
<td>= -50</td>
<td>= -50</td>
</tr>
<tr>
<td>( V_{reset} )</td>
<td>= -50mV</td>
<td>= -50mV</td>
</tr>
<tr>
<td>( V_{peak} )</td>
<td>= 35mV</td>
<td>= 35mV</td>
</tr>
<tr>
<td>( d_{Izh} )</td>
<td>= 100</td>
<td>= 100</td>
</tr>
<tr>
<td>( b )</td>
<td>= 100pA</td>
<td>= 100pA</td>
</tr>
<tr>
<td>( I_e )</td>
<td>= 0</td>
<td>= 0</td>
</tr>
<tr>
<td>( \Delta_T )</td>
<td>= 2mV</td>
<td></td>
</tr>
</tbody>
</table>
tion, up to $g_L$ that is negative. The linear term in the Izhikevich model will thus be positive and does therefore not model the leak current. The actual leak current is taken in account by the corresponding term in the adaptation equation, by choosing $b_{1zh}$ accordingly.

2.3.1 Voltage Traces

First, we have a look at the input-output relation of a neuron: Figure 2.1 shows the voltage trace from a patch-clamp experiment, figure 2.3 the stimulus response for the two models.

The spike initiation is more abrupt for the AdEx model and thus closer to experimental data. But differences can also be seen in the subthreshold regime, where the AdEx model is reacting faster at first, but spike initiation happens later compared to the Izhikevich model. The reason is to find in the right-hand side of the model equation depicted in figure 2.2.

It also becomes apparent in figure 2.2 that the effective threshold decisive for spike generation (the right zero crossing) differs between the two models (even if the two parameters named threshold are the same). We conclude that not all parameters in the model have a direct biophysical interpretation, even if the name suggests so.
Figure 2.2: Right-Hand Side for $w = 0$ ([V]=mV).

Figure 2.3: Voltage traces of Izhikevich (red) and AdEx (blue) model for 50 ms step current injections of amplitude 75, 110, and 145 pA (encoded in line types). The horizontal, fine, dotted lines are the effective thresholds (where a spike is generated if exceeded).
2.3.2 Phase Portraits

As both models are two-dimensional, phase portraits (figure 2.4) are highly informative and provide an intuitive understanding of the dynamics of the model equations by visualizing (with streamlines, among others) the vector field of their derivatives (i.e., the right-hand side) that governs the evolution of the state variables.

The nullclines of the system are given by

\[ \dot{V} = 0 \iff w(V) = F(V) - g_L \cdot (V - V_{\text{rest}}) + I \quad (2.22) \]

\[ \dot{w} = 0 \iff w(V) = a \cdot (V - V_{\text{rest}}) \quad (2.23) \]

While the w-nullcline is the same for both models, the shape of the V-nullcline depends on the non-linearity (in the case of the AdEx model, it is a parabola). Above the vertex of the V-nullcline, the effective spike threshold runs along its right branch (the ridge of a potential well that is the separatrix between two attractor basins). The different shape of the V-nullclines thus explain the difference in the effective spike threshold as observed in figure 2.3.

2.3.3 Equilibria and Bifurcations

Equilibria \((V_*, w_*)\) occur at the intersections of the nullclines, i.e., where

\[ F(V_*) - g_L \cdot (V_* - V_{\text{rest}}) + I = a \cdot (V_* - V_{\text{rest}}) \quad (2.24) \]

In the Izhikevich model, this amounts to

\[ \frac{g_L}{V_{\text{rest}} + V_{\text{th}}} \cdot V_*^2 - (a + g_L) \cdot V_* + (a + g_L) \cdot V_{\text{rest}} - \frac{g_L}{V_{\text{rest}} + V_{\text{th}}} \cdot V_{\text{rest}}^2 + I = 0. \quad (2.25) \]

Observe that \(a\) and \(g_L\) in fact share the same role in the model (cf. discussion regarding the leak current above) and indeed always show up together; \(a + g_L\) can therefore be seen as some kind of “effective conductance”. The equilibria are found analytically as

\[ \begin{pmatrix} V_* \\ w_* \end{pmatrix} = \begin{pmatrix} V_* \\ a \cdot (V_* - V_{\text{rest}}) \end{pmatrix} \quad (2.26) \]

with

\[ V_* = \frac{a + g_L}{2 \cdot g_L/(V_{\text{rest}} + V_{\text{th}})} \pm \sqrt{\left( \frac{a + g_L}{2 \cdot g_L/(V_{\text{rest}} + V_{\text{th}})} \right)^2 - \frac{1}{g_L/(V_{\text{rest}} + V_{\text{th}})} \left( (a + g_L) \cdot V_{\text{rest}} + I \right) - V_{\text{rest}}^2}. \quad (2.27) \]
Figure 2.4: Phase Portrait of the Two Models. Voltage $V[\text{mV}]$ on the abscissa, adaptation $w[\text{pA}]$ on the ordinate. The blue arrows indicate the stream lines of the vector field of the derivatives (right hand side of model ODEs). Green lines are nullclines for $V$ (i.e. to be crossed only vertically), blue lines are $w$-nullclines (i.e. to be crossed only horizontally). Intersection of the $V$ and $w$-nullcline are equilibria, for these parameters (table 2.3), the system is close to a bifurcation, but still able emit a spike (trajectory in phase space in red - after-spike reset as dashed arrow). An external current of $I_e = 55\text{pA}$ was applied (in this figure only).
Figure 2.5: Three-dimensional phase portraits illustrating the time derivative of $V$ and $w$. (A) and (B) show the $V$-derivative for the Izhikevich and the AdEx model in blue, the nullcline is visible as the intersection with the zero level in brown. (C) and (D) show $V'$ in green and $w'$ as a (linear) blue plane. Equilibria are visible as intersections of the nullclines.
Figure 2.6: (Numerical) Bifurcation diagrams for the Izhikevich (A) and AdEx (B) model. $w = a \cdot (V - V_{\text{rest}})$ (corresponding to a section along the w-nullcline in the phase plot).

If the vertex of the $V$-nullcline crosses the $w$-nullcline, a saddle node bifurcation occurs as a transition from zero to two equilibria (or vice versa). The bifurcation occurs when the term under the root is zero, because then there is exactly one equilibrium.

For the AdEx model,

$$g_L \Delta T \exp\left(\frac{V_s - V_{\text{th}}}{\Delta T}\right) - (a + g_L) \cdot (V_s - V_{\text{rest}}) + I = 0 \quad (2.28)$$

cannot easily be solved analytically, the root $V_s$ is rather found numerically (figure 2.6) with Newton’s method (the function is well-behaved).

2.3.4 Linearization and Stability Analysis

Linearizing the model equations around the equilibria reveals their stability. The first-order Taylor approximations around any $V_0$ are given by

$$(J_{V_0}^1 F_{\text{Izh}})(V) = \frac{g_L}{V_{\text{rest}} + V_{\text{th}}} (V_0^2 - V_{\text{rest}}^2) + 2 \frac{g_L}{V_{\text{rest}} + V_{\text{th}}} V_0 \cdot (V - V_0)$$

$$(J_{V_0}^1 F_{\text{AdEx}})(V) = g_L \Delta T \exp\left(\frac{V_0 - V_{\text{th}}}{\Delta T}\right) + g_L \exp\left(\frac{V_0 - V_{\text{th}}}{\Delta T}\right) \cdot (V - V_0) \quad (2.29)$$

With the Jacobi matrix, we can write the Izhikevich system linearized around $V_0$ as

$$\begin{pmatrix} \dot{V} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} \frac{g_L}{C_m} \left( \frac{2V}{V_{\text{rest}} + V_{\text{th}}} - 1 \right) & -\frac{1}{C_m} \\ \frac{1}{\tau_w} & -\frac{1}{\tau_w} \end{pmatrix} \begin{pmatrix} V \\ w \end{pmatrix} \left[ \begin{pmatrix} -\frac{V^2}{V_{\text{rest}} + V_{\text{th}}} + V_{\text{rest}} + \frac{1}{C_m} \\ -\frac{a}{\tau_w} V_{\text{rest}} \end{pmatrix} \right]. \quad (2.30)$$
For the AdEx model, we find
\[
\begin{pmatrix}
\dot{V} \\
\dot{w}
\end{pmatrix} = \begin{pmatrix}
\frac{g_L}{C_m} (\exp\left(\frac{V_0-V_{th}}{\Delta_T}\right) - 1) & -\frac{1}{\tau_w} \\
\frac{a}{\tau_w} & -\frac{1}{\tau_w}
\end{pmatrix}
\begin{pmatrix}
V \\
w
\end{pmatrix}
+ \begin{pmatrix}
\frac{g_L}{C_m} (\Delta_T - V_0) \exp\left(\frac{V_0-V_{th}}{\Delta_T}\right) + \frac{g_L}{C_m} V_{rest} + \frac{1}{\tau_w} \\
-\frac{a}{\tau_w} V_{rest}
\end{pmatrix}.
\tag{2.31}
\]

When slightly deflecting the system from its equilibrium, it will be driven back to the lower equilibrium, but diverge from the upper equilibrium (in the V-direction). As perturbations in the w-direction are always driven back to the w-nullcline, we conclude that the lower equilibrium is a stable node and the upper one a saddle.

### 2.3.5 Linear Subthreshold Regime

For \(V_{th} \to \infty\), we find the subthreshold regime that is equivalent to a Leaky Integrate-and-Fire (LIF) model:
\[
\lim_{V_{th} \to \infty} \frac{1}{C_m} \left(\frac{g_L}{V_{rest} + V_{th}} V^2 - \frac{g_L}{V_{rest} + V_{th}} V_{rest}^2 - g_L \cdot (V - V_{rest}) - w + I\right)
= \frac{1}{C_m} (-g_L \cdot (V - V_{rest}) - w + I) \tag{2.32}
\]
\[
\lim_{V_{th} \to \infty} \frac{1}{C_m} \left(\frac{g_L \Delta_T \exp\left(\frac{V - V_{th}}{\Delta_T}\right) - g_L \cdot (V - V_{rest}) - w + I}{\Delta_T}\right)
= \frac{1}{C_m} (-g_L \cdot (V - V_{rest}) - w + I) \tag{2.33}
\]

It follows that the parameters \(\Delta_T\) and \(V_{th}\) are irrelevant for the subthreshold regime if far enough from \(V_{th}\). The two models would thus be equivalent in this regime. This is also apparent in the linearized system, where the Jacobians both converge to the LIF model:
\[
\lim_{V_{th} \to \infty} \left(\frac{g_L}{C_m} (\exp\left(\frac{V_0-V_{th}}{\Delta_T}\right) - 1) & -\frac{1}{\tau_w} \\
\frac{a}{\tau_w} & -\frac{1}{\tau_w}
\right)
= \left(\frac{-g_L}{C_m} & -\frac{1}{\tau_w} \\
\frac{a}{\tau_w} & -\frac{1}{\tau_w}\right) \tag{2.34}
\]
\[
\lim_{V_{th} \to \infty} \left(\frac{g_L}{C_m} (\frac{2V_0}{V_{rest} + V_{th}}) - 1) & -\frac{1}{\tau_w} \\
\frac{a}{\tau_w} & -\frac{1}{\tau_w}
\right)
= \left(\frac{-g_L}{C_m} & -\frac{1}{\tau_w} \\
\frac{a}{\tau_w} & -\frac{1}{\tau_w}\right) \tag{2.35}
\]

However, as one is not far enough from \(V_{th}\) in practice, the models behave differently in the subthreshold regime, due to the different speed of convergence (figure 2.8). We chose the AdEx model for our network simulations because it shows the distinctive behaviour of subthreshold regime and action potential generation that is observed in real neurons. The non-linearity of the Izhikevich model however has a global scope (figure 2.7).
Figure 2.7: $V'(V)$ for (A) Izhikevich and (B) AdEx Model. Dashed lines are linear approximations at the equilibria, the dotted line a quadratic approximation. $I_e = 0$, $V_{\text{rest}} = -60\text{mV}$, and $V_{\text{th}} = -40\text{mV}$ as before. The sign of $F(V)$ determines the direction of evolution in the phase space, the roots accordingly corresponding to the equilibria (as $w = a \cdot (V - V_{\text{rest}})$, a section along the $w$-nullcline is shown). The lower (stable) equilibrium is at $V_{\text{rest}}$, the upper one (unstable) however slightly deflected from $V_{\text{th}}$. 
Figure 2.8: (A) Derivative as a function of $V_{th}$ (for $V = -59\text{mV}$), showing convergence towards the leaky integrate-and-fire model as a subthreshold regime (for $w = a \cdot (V - V_{rest})$). AdEx (blue) converges exponentially fast to $\frac{1}{C_m} (-g_L \cdot (V - V_{rest}) - w + I)$, while Izhikevich (red) converges only polynomially after having crossed the singularity in $-V_{rest}$. Relative deviation from the subthreshold regime (B) as a function of $V_{th}$ (for $V = -59\text{mV}$) and (C) as a function of $V$, approaching the threshold $V_{th} = -40\text{mV}$. 
The coupled differential equations defining the dynamics of the Adaptive Exponential Integrate-and-Fire model introduced in chapter 2 cannot be solved analytically due to the nonlinearity. Every use of the model thus relies on numerical integration of the model ODEs, and so does NEST - the software used for the simulation of a network of point-neurons. As with any numerical method, accuracy and performance are critical to achieve reliable simulation results - this is of particular importance here as a point-neuron for which the equations are to be solved will be part of a network of several 10,000 neurons, with two implications to be considered: First, the computational costs will scale with the number of neurons in the network, and second, errors will propagate (and possibly amplify) throughout the network.

Due to these considerations, an accurate and efficient solver is central for the simulation of a network of spiking neurons. Thus, an important part of this thesis consisted of investigating the numerical integration of the model equations with the aim of a gain in performance. As one fitness function evaluation in the model fitting procedure corresponds to a NEST-simulation with about 30,000 neurons, this is also crucial for the convergence of the optimization in chapter 4. Another important motivation was to dissolve the dependency of NEST on an external solver.

Up to now, a 4th order Runge-Kutta-Fehlberg solver from the GNU Scientific Library (GSL, Galassi 2009) has been used in NEST to solve the ODEs of the AdEx model. Relying on an external solver has the disadvantage of a potential overhead due to specific design constraints and external function calls and furthermore creates dependencies on third party software that might be problematic from a long-term perspective. An attempt at reimplementing the numerical integration routines was therefore appropriate.

### 3.1 Problem Formulation

The simulation of a network of point-neurons is discretized in time, i.e. the states of all neurons are synchronously updated at every simulation step of size dt (0.1 ms by default). At every step, spikes emitted by a neuron are propagated to its postsynaptic neurons and stored in a ring buffer to ensure they are only considered after the respective synaptic delay time. When actually arriving after the delay, incoming spikes give rise to a postsynaptic-potential (with a certain rise and de-
lay time), either modelled as a change in synaptic conductance or as the postsynaptic current directly, depending on the implementation.

Within these simulation time steps, the state of every neuron is updated independently by integrating the model ODEs over the time step $dt$, given the current state as an initial condition. This numerical integration in turn is generally carried out by a single-step method proceeding by the numerical integration step size $h \leq dt$, thus conducting the numerical integration on a refined version of the simulation time grid where spikes are communicated. In addition, higher-order Runge-Kutta methods internally rely on intermediate steps.

The AdEx model is represented in NEST as

\[ C \frac{dV(t)}{dt} = g_L \Delta T \cdot \exp \left( \frac{V(t) - V_T}{\Delta T} \right) - g_L \cdot (V(t) - E_L) - w(t) + I_e - g_{exc} \cdot (V(t) - E_{exc}) - g_{inh} \cdot (V(t) - E_{inh}) \] (3.1)

\[ \tau_w \frac{dw(t)}{dt} = a \cdot (V(t) - E_L) - w(t), \] (3.2)

where $g_{exc} \cdot (V(t) - E_{exc})$ is the postsynaptic current into the cell after the excitation of an excitatory synapse, with the synaptic conductance $g_{exc}$ and the excitatory reversal potential $E_{exc}$. As the case of an inhibitory synapse is perfectly analogous, we will focus on excitatory synapses in the following reasoning for reasons of clarity.

The postsynaptic currents are governed by the kinetics of neurotransmitter receptor channels (e.g. GABA, AMPA, NMDA). These synaptic dynamics can be implemented by either modelling the postsynaptic currents directly (analytical calculation), or by modelling the synaptic conductances that lead to the currents (implemented with numerical solution of ODEs). For both cases, either an exponential function or a so-called $\alpha$-function of the form

\[ \alpha(t) = \frac{\alpha_{\text{max}} \cdot e}{\tau_s} \cdot t \cdot \exp \left( \frac{-t}{\tau_s} \right) \] (3.3)

with a time constant $\tau_s$ can be used ($e$ being Euler’s number and $\alpha_{\text{max}}$ the peak of the kernel). If the conductance is modelled as an $\alpha$-function, the postsynaptic current is given by

\[ I(t) = g(t) \cdot (V - E_{\text{syn}}) = \alpha(t) \cdot (V - E_{\text{syn}}), \quad \text{syn} \in \{\text{exc, inh}\} \] (3.4)

Alternatively, the postsynaptic current can be modelled to be an $\alpha$-function itself:
Figure 3.2: Modelling synaptic conductance with an \( \alpha \)-kernel (shown for the case of an excitatory synapse). The maximal conductance \( \bar{g}_{\text{max}} \) (that is the synaptic weight) is attained after the synaptic time constant \( \tau_{\text{syn,exc}} \). The incoming current corresponding to the postsynaptic potential (PSP) is given by \( I_{\text{syn,exc}}(t) = g_{\text{exc}}(t) \cdot (V(t) - E_{\text{exc}}) \) with the synaptic reversal potential \( E_{\text{exc}} \).

\[
I(t) = \alpha(t) \quad (3.5)
\]

The same reasoning applies for the exponential functions. Modelling conductances is more accurate, but \( I(t) \) and \( g(t) \) can often be assumed to be of a similar form if \( V - E_{\text{syn}} \) is close to constant (which may be assumed for excitatory synapses, as \( V \ll E_{\text{exc}} \)). For most neuron models, NEST implements both versions; for the AdEx model, only the version modelling the conductances is present.

Modelling the synaptic conductances as exponential kernels amounts to the differential equations

\[
\frac{d}{dt}g_{\text{exc}} = \frac{-g_{\text{exc}}}{\tau_{\text{syn,exc}}} \quad (3.6)
\]

\[
\frac{d}{dt}g_{\text{inh}} = \frac{-g_{\text{inh}}}{\tau_{\text{syn,inh}}} \quad (3.7)
\]

that correspond to a postsynaptic current

\[
I_{\text{syn,exc}}(t) = \left( \frac{\bar{g}_{\text{exc}} \cdot \exp(-\frac{t}{\tau_{\text{syn,exc}}}) + g_{0}}{\bar{g}_{\text{exc}}} \right) \cdot (V(t) - E_{\text{exc}}). \quad (3.8)
\]

In the following, we choose \( g_{0} = 0 \) (such that \( \lim_{t \to \infty} g = 0 \)) and \( \bar{g}_{\text{exc}} = 1 \) (as NEST sets the synaptic weights as maximum conductance \( \bar{g}_{\text{exc}} \), thus normalizing the equation).
Another possibility is to model the conductances as $\alpha$-functions, giving rise to a postsynaptic current of the form

$$I_{\text{syn,exc}}(t) = \left( \frac{g_{\text{exc}} \cdot e^{-t/\tau_{\text{syn,exc}}}}{\tau_{\text{syn,exc}}} \cdot \exp\left(-\frac{t}{\tau_{\text{syn,exc}}} \right) \right) \cdot (V(t) - E_{\text{exc}}). \quad (3.9)$$

We choose $g_{\text{exc}} = 1$, for the same reason as before. The $\alpha$-function $g_{\text{exc}}(t)$ can be shown to be a solution of the differential equation (Diesmann and Gewaltig 1994, 89)

$$\frac{d^2}{dt^2} g_{\text{exc}} + \frac{2}{\tau_{\text{syn,exc}}} \frac{d}{dt} g_{\text{exc}} + \frac{1}{\tau_{\text{syn,exc}}^2} g_{\text{exc}} = 0 \quad (3.10)$$

with initial conditions

$$g_{\text{exc}}(t)|_{t=0} = g_0 = 0 \quad (3.11)$$
$$\frac{d}{dt} g_{\text{exc}}(t)|_{t=0} = 0 \quad (3.12)$$
$$\frac{d^2}{dt^2} g_{\text{exc}}(t)|_{t=0} = -\frac{e}{\tau_{\text{syn,exc}}} \quad (3.13)$$

This 2nd order differential equation describes a critically damped harmonic oscillator and can be rewritten as a system of two linear differential equations of first order:

$$\begin{align*}
\frac{d}{dt} \left( \frac{d}{dt} g_{\text{exc}} + \frac{1}{\tau_{\text{syn,exc}}} g_{\text{exc}} \right) + \frac{1}{\tau_{\text{syn,exc}}} \left( \frac{d}{dt} g_{\text{exc}} + \frac{1}{\tau_{\text{syn,exc}}} g_{\text{exc}} \right) &= 0 \\
\Leftrightarrow \begin{cases} 
\frac{d}{dt} dg_{\text{exc}} + \frac{1}{\tau_{\text{syn,exc}}} dg_{\text{exc}} = 0 \\
\frac{d}{dt} g_{\text{exc}} = \frac{d}{dt} g_{\text{exc}} + \frac{1}{\tau_{\text{syn,exc}}} g_{\text{exc}} \\
\frac{d}{dt} dg_{\text{exc}} = -\frac{dg_{\text{exc}}}{\tau_{\text{syn,exc}}} \\
\frac{d}{dt} g_{\text{exc}} = dg_{\text{exc}} - \frac{g_{\text{exc}}}{\tau_{\text{syn,exc}}} 
\end{cases} \quad (3.14)
\end{align*}$$

Equation (3.14) and its analogon for inhibitory synapses (3.15) are used in NEST to model $\alpha$-kernels, whereas equations (3.6) and (3.7) model exponential kernels.

$$\begin{align*}
\begin{cases} 
\frac{d}{dt} dg_{\text{inh}} &= -\frac{dg_{\text{inh}}}{\tau_{\text{syn,inh}}} \\
\frac{d}{dt} g_{\text{inh}} &= dg_{\text{inh}} - \frac{g_{\text{inh}}}{\tau_{\text{syn,inh}}} 
\end{cases} \quad (3.15)
\end{align*}$$

(3.1) $\cap$ (3.2) together with (3.6) $\cap$ (3.7) (exponential postsynaptic conductance) or with (3.14) $\cap$ (3.15) ($\alpha$-function conductance) define an Initial-Value Problem that cannot be solved analytically due to the non-linear term in equation (3.1) (for a semianalytical approach that is presented later reasons, see section §3.4).
3.2 RUNGE-KUTTA SINGLE STEP METHOD

For the numerical integration within a simulation step we choose a Runge-Kutta method. Recall that any \( s \)-stage explicit Runge-Kutta Single Step Method to solve the ODE-IVP

\[
\dot{y} = f(t,y) \quad (3.16)
\]

\[
y(t_0) = y_0 \quad (3.17)
\]

can be written in a general form as

\[
k_i = hf(t_n + c_i h, y_n + \sum_{j=1}^{i-1} a_{ij} k_j) \quad (3.18)
\]

\[
y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i \quad (3.19)
\]

The method is specified by a set of coefficients that are often presented in a Butcher scheme (table 3.1).

A first object-oriented implementation allowed the use of different integration methods with the strategy pattern (Gamma 1995). As this design showed to be no faster than the use of the GSL (due to
indirections and lookups in the virtual function table), we decided to implement the method inline. After having tested different implementations (Runge-Kutta methods of different order with or without adaptive step size), we opted for the method that performed best with respect to accuracy and runtime: the Dormand-Prince method (Dormand and Prince 1980) as described in (Press et al. 2007) and presented in algorithm 3.1A, a combined 4th/5th order Runge-Kutta method with adaptive stepsize control governing the stepsize depending on the error estimate \( \text{err} \). The error estimate \( \text{err} = \frac{|y_n^{(5)} - y_n^{(4)}|}{\text{MAXERR}} \) is calculated as the difference between the 4th order and 5th order solutions and the nominator guaranteed to be smaller than \( \text{MAXERR} \). A step is only accepted if \( \text{err} < 1 \). The next integration step is given by \( h = h \cdot 0.98 \cdot \left( \frac{1}{\text{err}} \right)^{1/5} \), but cannot become smaller than \( \text{HMIN} \). The central parts of the implementation are presented in algorithm 3.1A, they are part of the function that updates the state of a neuron (i.e. its voltage, adaptation variable, and synaptic conductances).

Listing 3.1: Updating Neuron by Integration over Simulation Step

```c
while(t < B_.step_) { // while not reached end of sim. step
do{
if(B_.step_ - t < h) // stop at end of simulation step
h = B_.step_ - t;
t_return = t+h; //update t by numerical integration step
aiif_condo_alpha_RK_dynamics(S_.y_,S_.k1); //k1=f(told,y)
for(int i=0; i < S_.STATE_VEC_SIZE; ++i) // k2
S_.yin[i] = S_.y_[i] + h * S_.k1[i]/5.0;
aiif_condo_alpha_RK_dynamics(S_.yin,S_.k2);
for(int i=0; i < S_.STATE_VEC_SIZE; ++i) // k3
S_.yin[i] = S_.y_[i] + h * ( 3.0/40.0*S_.k1[i]
 + 9.0/40.0*S_.k2[i] );
aiif_condo_alpha_RK_dynamics(S_.yin,S_.k3);
for(int i=0; i < S_.STATE_VEC_SIZE; ++i) // k4
S_.yin[i] = S_.y_[i] + h * ( 44.0/45.0*S_.k1[i]
 - 56.0/15.0*S_.k2[i] + 32.0/9.0*S_.k3[i] );
aiif_condo_alpha_RK_dynamics(S_.yin,S_.k4);
for(int i=0; i < S_.STATE_VEC_SIZE; ++i) // k5
S_.yin[i] = S_.y_[i] + h * ( 19372.0/6561.0*S_.k1[i]
 - 25360.0/2187.0*S_.k2[i] + 64448.0/6561.0*S_.k3[i]
 - 212.0/729.0*S_.k4[i] );
aiif_condo_alpha_RK_dynamics(S_.yin,S_.k5);
for(int i=0; i < S_.STATE_VEC_SIZE; ++i) // k6
S_.yin[i] = S_.y_[i] + h * ( 9017.0/3168.0*S_.k1[i]
 - 355.0/33.0*S_.k2[i] + 46732.0/5247.0*S_.k3[i]
 + 49.0/176.0*S_.k4[i] - 5103.0/18656.0*S_.k5[i] );
aiif_condo_alpha_RK_dynamics(S_.yin,S_.k6);
for(int i=0; i < S_.STATE_VEC_SIZE; ++i) // 5th order
S_.ynew[i] = S_.y_[i] + h * ( 35.0/384.0*S_.k1[i]
 + 500.0/1113.0*S_.k3[i] + 125.0/192.0*S_.k4[i]
 - 2187.0/6784.0*S_.k5[i] + 11.0/84.0*S_.k6[i] );
}
```

30 NUMERICAL INTEGRATION OF ADEX MODEL ODES
aeif_cond_alpha_RK_dynamics(S_.yin,S_.k7);

for(int i=0; i < S_.STATE_VEC_SIZE; ++i) // 4th order
S_.yref[i] = S_.y_[i] + h * ( 5179.0/57600.0*S_.k1[i]
+ 7571.0/16695.0*S_.k3[i] + 393.0/640.0*S_.k4[i]
- 92097.0/339200.0*S_.k5[i] + 187.0/2100.0*S_.k6[i]
+ 1.0/40.0*S_.k7[i] );

err = std::fabs(S_.ynew[0] - S_.yref[0]) / MAXERR + 1.0e-200;
bool done = (h<=HMIN); // Exit loop if h was <=HMIN already
h *= 0.98*std::pow(1.0/err,1.0/5.0); // next integr. step
size
h = std::max(h,HMIN);
}
while( (err > 1.0) and (not done)); //reject step if err > 1
for(unsigned int i=0; i<S_.STATE_VEC_SIZE; ++i)
S_.y_[i] = S_.ynew[i]; // pass updated values
t = t_return;
}

The fact that the numerical integration steps of varying size \( h \) by which the solver proceeds are bounded by the simulation step of size \( dt \), makes the implementation unexpectedly tricky. Not only is it a constraint to the adaptive step size control limiting its leeway, but the fixed time grid leads to a “binning” of the spike detection and may also have delicate effects on the errors: The smallest deviation in the integration within a bin (simulation time step) can make an important difference in the spike detection: whether a spike falls in one or the next bin (simulation time step). As this does not only shift the spike time by \( dt \), but also affects the after-spike reset, this can lead to an error accumulation. For this reason, the method has to be implemented and tuned with all the more care. In particular, the performance is sensitive with respect to the choice of MAXERR, as this governs the step size control.

3.3 PERFORMANCE AND ACCURACY TESTS

The new numerics implementation of the AdEx model was tested by conducting a 3s NEST simulation of an AdEx neuron that received a 700pA step current of duration 2s, leading to a total of 17 spikes. A simulation time step of \( dt = 0.1 \text{ms} \) was used for both the previous GSL implementation and the new implementation based on the Dormand-Prince Runge-Kutta method. The minimum stepsize HMIN was 0.001 ms and the error tolerance MAXERR \( 10^{-10} \text{ mV} \) in the new implementation (the default values), whereas GSL relied on a maximum error tolerance of \( 10^{-6} \text{ mV} \) (can only be compared with each other \textit{addito salis grano} however). The previous implementation with a 100 times higher resolution (i.e. simulation time step \( dt=0.001\text{ms} \)) was used as a reference to calculate the following error measures: deviation in spike times and \( L^2 \)-error of the voltage traces between the implementations to be benchmarked on the one hand,
and the reference on the other hand. The correct integration of conductances was verified by connecting the test neuron to a second one, where arriving spikes were visible as postsynaptic potentials.

The performance gain of the new implementation was 12% (Table 3.2) while improving accuracy, as well (Figure 3.3). The new numerical implementation was therefore contributed as a new model to the NEST repository and will be included in the next release.

3.4 SEMIANALYTICAL APPROACH

Let us come back to the statement of above that we cannot solve the problem analytically. In fact, this is only true for equation (3.1), as the other equations (3.2) and (3.6), (3.7) or (3.14), (3.15) of the system are linear and analytically solvable. As seen in (3.14), the analytical solution is known by construction, and can also be used when modelling conductances (as is the case in NEST already for modelling postsynaptic currents).
It is thus possible to use a semianalytical approach to solve the ODE system describing the neural and synaptic dynamics in the AdEx model: To update the voltage at every time step, we integrate equation equation (3.1) numerically, relying on a time discretization with potentially adaptive stepsize (governed by a Runge-Kutta method). To update the other variables, we use the analytical solutions of the remaining equations. The corresponding formulae are presented in appendix B. However, the implementation showed to be extremely sensitive with respect even to the slightest deviation in w. Using this semianalytical approach led to such small deviations, probably due to the difference in the updating of the coupled equations (in numerics, updates are asynchronic, delayed). As the expected gain is only moderate (because the analytical solution has to be updated at every Runge-Kutta-substep, an operation that is not cheaper than computing the derivative), the semianalytical approach was therefore not competitive.

<table>
<thead>
<tr>
<th></th>
<th>Runtime [s] (average over 50 runs)</th>
<th>Max. Deviation in Spike Time [ms]</th>
<th>L2-Error (per s) of Voltage Trace [mV]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous Implementation (GSL)</td>
<td>0.043</td>
<td>0.33</td>
<td>7.452 · 10^{-5}</td>
</tr>
<tr>
<td>New Implementation</td>
<td>0.038</td>
<td>0.15</td>
<td>5.68 · 10^{-5}</td>
</tr>
<tr>
<td>Ratio</td>
<td>0.88</td>
<td>0.45</td>
<td>0.76</td>
</tr>
<tr>
<td>Difference</td>
<td>−12%</td>
<td>−55%</td>
<td>−24%</td>
</tr>
</tbody>
</table>

Table 3.2: Overview of performance and accuracy tests. Results of the algorithms for simulation time step dt=0.1 ms were compared to the previous implementation with dt=0.001 ms as a reference. Results were generated from NEST version 2.2.1 (compiled with g++ 4.6.3) with Python 2.7.3 (IPython 0.12.1) on a 8x Intel Core i7 CPU 960 at 3.20 GHz computer with 5.8 GiB RAM under Ubuntu 12.04 (64 bit). The GSL version used is 1.15.
In this chapter, I introduce an automated approach to build a simplified version of a morphologically detailed model of the Blue Brain column. To this end, I developed and implemented the “Optimizing Point-Neuron Column” (OPoNC) framework that creates a point-neuron simulation replicating the behaviour of a given detailed simulation.

The simplified simulation is based on a network of point-neuron models which are reduced to the minimum, while still being able to reliably predict (Brette and Gerstner 2005) spikes. Neurons can be described by different point-neuron models (see 2). In the following, we will focus on the Adaptive Exponential Integrate-and-Fire (AdEx) model, but the developed methodology can also be used for other point-neuron models with minor changes. The simulation of the point-neuron column is implemented in NEST (NEural Simulation Tool, Gewaltig and Diesmann 2007; Gewaltig et al. 2012) and could possibly be extended to PyNN (Davison 2008) at a later point (for portability towards other simulators such as NEURON (Carnevale 2006) and Brian (Goodman 2008)).

The behaviour and characteristics of the network simulation depend on the choice of parameters of the point-neuron models. The problem of constructing a simulation that matches the Blue Brain simulation is thus two-fold: 1. Setting up a NEST-simulation of a network of point-neurons that is - as far as possible - congruent to the detailed simulation with respect to the static structure of the column, as well as the environment specified by the experiment protocol. 2. Tuning the neuron model parameters in order to align the two simulations such that their dynamical behaviour matches as closely as possible. The following sections describe how these tasks are accomplished.

4.1 REDUCTION TO POINT-NEURON SIMULATION

In the first step we set up a point-neuron network simulation that is congruent to a detailed simulation, by considering only the positions and synaptic connectivity of neurons, and neglecting the neuronal morphology. As a consequence, synapses are collapsed onto the soma. The single neurons are modelled in NEST as AdEx point-neurons with $\alpha$-functions as postsynaptic conductance kernels (using the new numerics implementation “aeif_cond_alpha_RK5” of 3). To reduce the computational complexity of fitting one parameter set
per neuron, we restrict our approach to fitting one parameter set per morphological-electrophysiological type (me-type). Thus instead of searching for 31,000 parameter sets, we only need to find 207 parameter sets. These values are retrieved from an HDF5 file by the Python-module “neurob” (Erő 2013b) and then saved to another HDF5 file containing the resulting reduced circuit loaded by OPoNC.

4.1.1 Reducing the Number of Populations

Because fitting a parameter set for all of the 207 me-types present in the model would still lead to a too large parameter space, the number of populations distinguished in the simplified model has to be reduced further. This is done by introducing a mapping that lumps several similar me-types into one reduced type.

The suggested mapping (table 4.1) clusters me-types into four groups (while layer information is maintained): pyramidal cells, and interneurons targeting the dendrites of the postsynaptic neuron proximally or distally (to the soma), or targeting its Axon Initial Segment, respectively. Thus, the mapping relies exclusively on the morphological type and neglects the electrophysiological type. This division is justified by its attempt to maintain the most important characteristics of the me-types (Srikanth Ramaswamy, pers. comm.). As there are only dendrite targeting interneurons in layer 1, and layers 2 and 3 are combined (as
<table>
<thead>
<tr>
<th>m-Type</th>
<th>Full Name</th>
<th>Reduced Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>Pyramidal Cell</td>
<td>Pyramidal Cell</td>
</tr>
<tr>
<td>SP</td>
<td>Star Pyramidal Cell</td>
<td></td>
</tr>
<tr>
<td>SS</td>
<td>Spiny Stellate Cell</td>
<td></td>
</tr>
<tr>
<td>BPC</td>
<td>Bipolar Pyramidal Cell</td>
<td></td>
</tr>
<tr>
<td>IPC</td>
<td>Inverted Pyramidal Cell</td>
<td></td>
</tr>
<tr>
<td>TPC</td>
<td>Tufted Pyramidal Cell</td>
<td></td>
</tr>
<tr>
<td>STPC</td>
<td>Slender-Tufted Pyramidal Cell</td>
<td></td>
</tr>
<tr>
<td>UTPC</td>
<td>Untufted Pyramidal Cell</td>
<td></td>
</tr>
<tr>
<td>TTPC1</td>
<td>Thick-Tufted PC (Late Apical Tuft)</td>
<td>Prox. Dendrite Targ. Interneuron</td>
</tr>
<tr>
<td>TTPC2</td>
<td>Thick-Tufted PC (Early Apical Tuft)</td>
<td></td>
</tr>
<tr>
<td>LBC</td>
<td>Large Basket Cell</td>
<td></td>
</tr>
<tr>
<td>NBC</td>
<td>Nest Basket Cell</td>
<td></td>
</tr>
<tr>
<td>SBC</td>
<td>Small Basket Cell</td>
<td></td>
</tr>
<tr>
<td>ChC</td>
<td>Chandelier Cell</td>
<td>Axon Initial Segment Targeting Interneuron</td>
</tr>
<tr>
<td>MC</td>
<td>Martinotti Cell</td>
<td></td>
</tr>
<tr>
<td>BTC</td>
<td>Bitufted Cell</td>
<td></td>
</tr>
<tr>
<td>DBC</td>
<td>Double Bouquet Cell</td>
<td></td>
</tr>
<tr>
<td>BP</td>
<td>Bipolar Cell</td>
<td>Distal Dendrite Targeting Interneuron</td>
</tr>
<tr>
<td>NGC</td>
<td>Neurogliaform Cell</td>
<td></td>
</tr>
<tr>
<td>SLAC</td>
<td>Sparse Local Arborizing Cell</td>
<td></td>
</tr>
<tr>
<td>NGC-DA</td>
<td>Neurogliaform Cell - Dense Arborizing</td>
<td></td>
</tr>
<tr>
<td>NGC-SA</td>
<td>Neurogliaform Cell - Sparse Arborizing</td>
<td></td>
</tr>
<tr>
<td>DLAC</td>
<td>Dense Local Arborizing Cell</td>
<td></td>
</tr>
<tr>
<td>HAC</td>
<td>Horizontal Arborizing Cell</td>
<td></td>
</tr>
<tr>
<td>DAC</td>
<td>Descending Axon Collateral Cell</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Type mapping to reduce the number of populations. Pyramidal neurons are excitatory, interneurons inhibitory.
is the case in the full model), this mapping results in a total number of 17 populations, with the frequency distribution of figure 4.2.

4.1.2 **Synaptic Connectivity**

In principle, synaptic connectivity is also imported from the full simulation, but there are several delicate points that have to be kept in mind: The detailed model uses a stochastic model of synaptic transmission while the corresponding model in NEST is deterministic, describing the average behaviour. Also, there are only one type of excitatory and inhibitory synapses available in NEST, therefore only AMPA and GABA A receptors can be modelled; slower NMDA receptors are neglected. As synapses are collapsed to the soma, attenuation (and delay) of post-synaptic potentials (PSP) due to their propagation along the dendritic arbor is also not modelled. The delay time of the synapse itself however is stored and can be retrieved from the simulation.

The same holds for the synaptic weights, i.e. the maximal conductance of a synapse (amplitude of the corresponding kernel, cf. figure 3.2). These weights can however not be retrieved straight-forward from the Blue Brain database, as they are rescaled in the configuration file of the simulation. The reason for this rescaling is that conductances are measured at the soma in experiments; synaptic conduc-
Data will be published in the forthcoming paper on the Blue Brain Neocortical Column. Details are available upon request.

Figure 4.3: Peak synaptic conductances (synaptic weights) of different synaptic pathways. From: Srikanth Ramaswamy, unpublished data.
tances are thus scaled in the simulation to correct for dendritic filtering, such that the conductance at the soma matches the experiment (Srikant Ramaswamy, pers. comm.). We solved this issue by parsing the configuration file and rescaling the weights accordingly (runtime of 6 hours). The effect of the weight scaling is visible in figure 4.4.

While the synaptic weights, the delay time, the synapse dynamics time constants $\tau_{\text{recovery}}$ and $\tau_{\text{facilitating}}$ as well as the synaptic efficacy $U_{\text{SE}}$ are set by the NEST function DataConnect(), it remains to determine the synaptic time constants. NEST only allows one or two synaptic time constants per neuron, rather than per synapse as in the full model, so we average its value over all incoming synapses. But as was the case for the synaptic weights, the synaptic time constants from the full model must also be scaled. The postsynaptic potentials are modelled in the detailed simulation by adding two exponentials (with a maximum of $\bar{g}_{\text{max}}$) of the form

$$\frac{(\tau_d/\tau_r)^{(\tau_d)/(\tau_d-\tau_r)} \cdot \bar{g}_{\text{max}}}{(\tau_d/\tau_r)-1} \left( \exp \left( -\frac{t}{\tau_d} \right) - \exp \left( -\frac{t}{\tau_r} \right) \right) \quad (4.1)$$

to the conductance for every incoming spike. The synaptic decay time constant $\tau_d$ is drawn from a distribution and can be retrieved from the full simulation for every synapse; rise time constants $\tau_r$ are 0.2 ms for all AMPA and GABA A synapses. The conductance kernels used in NEST however only have one synaptic time constant, in the case of an exponential kernel $\bar{g}_{\text{max}} \cdot \exp \left( -\frac{t}{\tau_{\text{syn}}} \right)$ it is the decay time, whereas it is the rise time for the $\alpha$-function $\frac{\bar{g}_{\text{max}} \cdot e^{\tau_{\text{syn}} \cdot t} \cdot \exp \left( -\frac{t}{\tau_{\text{syn}}} \right)}{\tau_{\text{syn}}}$ (cf. figure 3.2) that relies on one exponential term only. It is however not appropriate to use $\tau_d$ for exponential kernels or - even worse - the constant $\tau_r$ for $\alpha$-kernels, as this does not necessarily lead to a comparable effect in the postsynaptic neuron. The synaptic time constant to be used is best determined by maintaining the integral over the conductance kernels (thereby maintaining the current flowing into the cell):

$$\int_{0}^{\infty} \frac{(\tau_d/\tau_r)^{(\tau_d)/(\tau_d-\tau_r)} \cdot \bar{g}_{\text{max}}}{(\tau_d/\tau_r)-1} \cdot \left( e^{-\frac{t}{\tau_d}} - e^{-\frac{t}{\tau_r}} \right) dt = \bar{g}_{\text{max}} \cdot \tau_{\text{syn}} \quad (4.2)$$

For exponential kernels it follows from

$$\int_{0}^{\infty} \frac{\bar{g}_{\text{max}}}{\tau_{\text{syn}}} \cdot \exp \left( -\frac{t}{\tau_{\text{syn}}} \right) dt = \bar{g}_{\text{max}} \cdot \tau_{\text{syn}} \quad (4.3)$$

that $\tau_{\text{syn}} = \frac{(\tau_d/\tau_r)^{(\tau_d)/(\tau_d-\tau_r)} \cdot \bar{g}_{\text{max}}}{(\tau_d/\tau_r)-1} \cdot (\tau_d - \tau_r)$, and for $\alpha$-kernels,

$$\int_{0}^{\infty} \frac{\bar{g}_{\text{max}} \cdot e^{-\frac{t}{\tau_{\text{syn}}}} \cdot \exp \left( -\frac{t}{\tau_{\text{syn}}} \right)}{\tau_{\text{syn}}} \cdot \frac{t}{\tau_{\text{syn}}} dt = \frac{\bar{g}_{\text{max}} \cdot e^{\tau_{\text{syn}} \cdot t_{\text{syn}}^2}}{\tau_{\text{syn}} \cdot \tau_{\text{syn}}^2} \quad (4.4)$$
Figure 4.4: Comparison of network activity without the weight scaling (A) and with the weight scaling (B), with a constant input current of 200 pA. With Csaba Erő, unpublished data.
leads to $\tau_{\text{syn}} = \frac{1}{e^{\frac{(\tau_d/\tau_r)^{1/\tau_d}}{(\tau_d/\tau_r)-1}}}(\tau_d - \tau_r)$.

4.1.3 Replicating the Experiment Protocol

In order to align the simplified with the full simulation, their inputs have to be identical. Given the diversity of stimuli applied, this is a non-trivial task. The stimulus protocol specified in the configuration file, such as injection of hyperpolarizing step currents and Gaussian noise of a certain duration, are replicated using the NEST step_current_generator. The noise is also treated as a constant current because the variances used in the simulations are close to zero. The amplitudes of these current injections are specified in units of the threshold potential of a cell, therefore all threshold potentials have to be retrieved by parsing one file per neuron morphology prior to replicating the current injections. The spontaneous activation of synapses however that is present in the detailed model was not implemented in the simplified model (due to the design of NEST, this is a non-trivial task).

The focus of the simplified simulation is limited to one column (whereas the detailed simulation generally consists of several columns). Therefore, spikes that the column of interest received from neighbouring columns have to be emulated. This amounts to retrieving all spikes emitted by a neuron that is presynaptic to one in the column of interest, and defining a filter that considers those only that are not themselves in this column (as these shall not be emulated, but arise in the simulation proper). The resulting spikes are then replayed into the simulation by a NEST spike_generator for every neuron. This feature is implemented and fully functional (at the cost of a considerable rise in runtime due to the large amounts of data to be processed), but was not used for this simulation as there were no spikes propagating from outside into the central column (neighbouring columns not stimulated accordingly).

4.1.4 Initial Conditions

The state variables of the neuron model are initialized at $V = V_{\text{rest}}$ and $w = 0$ when starting the simulation. During the fitting procedure, several runs of the simulation are run with the same NEST network. At the beginning of every simulation, the network is reset to its initial state. The only values that cannot be reset in the current version of NEST (a feature that will be available in a future version) are the states of the synapse dynamics. As spikes are not emitted from the very beginning of a simulation, this slight inaccuracy should have minimal effects only.
Once the point-neuron simulation is properly set up, we can aim at aligning the two simulations by tuning the neuron model parameters such that the behavioural properties of the two simulations match as closely as possible. This is achieved by fitting the neuron model parameters. Each of the populations introduced in section 4.1.1 shall be represented by a distinct set of model parameters to account for their diversity and eventually gain insight into their function and properties.

While parameter values are critical to the behaviour of the model (cf. chapter 2), many different sets of parameters are used in the literature. Usually, the model fitting is performed for one neuron to accurately reproduce the firing and subthreshold behaviour of another neuron, by minimizing the error of the predicted output (e.g. $L^2$-error of voltage trace, or deviation in spike times) for a certain input protocol (e.g. protocol of ramp and step current injections or spike trains with spike times drawn from a Poisson or Ornstein-Uhlenbeck process). Often, different stages of fitting are performed to estimate the different parameters (Brette and Gerstner 2005). There is a diversity of well approved procedures (Druckmann et al. 2008; Bahl et al. 2012; Pool and Mato 2011; Mensi et al. 2012). Most of them have two disadvantages: First, the fitting procedure often depends on the model. We are in search of a generic approach however that is independent of the model (non-parametric fitting). Second, they often rely on specifically designed inputs. The fact that neurons within a microcircuit are not subject to a well defined input makes it difficult to generalize the approach, in view of the diversity of detailed neurons present in the model.

### 4.2.1 Network-Level Optimization of Point-Neuron Parameters

We therefore adopted a novel approach: As we are interested in the dynamics of the column rather than the exact response or spike times of a single neuron, the parameters are fitted within the network and with respect to statistical measures of the network dynamics. This is justified by the argument that there is always a considerable amount of noise and stochasticity in the system that smears out exact spike times. This method ensures on the one hand that neurons are fitted for a regime of input that comes close to what they receive in vivo. On the other hand, it might help capturing important features of the network dynamics that would otherwise not be represented in the model. As the problem is being relaxed, these advantages are gained at the cost of an immensely increasing parameter space - with the risk of an under-constrained problem. Indeed, convergence is difficult to achieve, as we will see in chapter 5.
Table 4.2: AdEx parameters (above) and further NEST simulation parameters (below) as reported for a pyramidal neuron in Brette and Gerstner 2005, and as fitted previously (Anirudh Vij, unpublished data) for excitatory and inhibitory neurons.
The parameter set that is to be fitted for the Adaptive Exponential Integrate-and-Fire Model was introduced in chapter 2 (the dynamic state variables being the membrane voltage $V_m$, the adaptation current $w$, and the synaptic conductances $g_{ex}$ and $g_{in}$). Table 4.2 gives an overview about the parameters reported in the original paper for a pyramidal neuron (Brette and Gerstner 2005) and those that were used previously in point-neuron column simulations at the Blue Brain Project, resulting from a fitting by Anirudh Vij (unpublished data).

### 4.2.2 Spike Statistics as a Measure for Network Dynamics

The (multi-valued) objective function to be minimized in the parameter fitting is the relative error between a statistical measure calculated for a detailed and a simplified simulation. We aim at a hierarchical fitting that proceeds from coarse to fine by considering different spike statistics. Mean firing rates of populations serve as the first approximation. This will only capture some aspects of the network dynamics - in order to describe them more accurately, other spike statistics could then be taken into account successively, like distributions of firing rates, time-dependent firing rates, inter-spike intervals, pairwise correlations, and peristimulus time histograms (Perkel et al. 1967).

### 4.2.3 Optimisation Algorithm

The approach introduced above amounts to a multiobjective optimisation problem. Apart from the large parameter space (8 parameters for 17 populations, i.e. 136 dimensions), little is known about the shape of the fitness landscape; it may however be expected to be very rugged. An evolutionary or genetic algorithm thus seems most appropriate. The software relies on a simple evolutionary algorithm implemented with “DEAP” (Fortin et al. 2012). The genome of the genetic algorithm

<table>
<thead>
<tr>
<th>SYNAPTIC PARAMETERS</th>
<th>Brette and Gerstner 2005</th>
<th>DETAILED MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Excitatory Reversal Potential $E_{exc}$</td>
<td>0 mV</td>
<td>0 mV</td>
</tr>
<tr>
<td>Inhibitory Reversal Potential $E_{inh}$</td>
<td>-85 mV</td>
<td>-80 mV</td>
</tr>
<tr>
<td>Excitatory Synaptic Time Constant $\tau_{syn,exc}$</td>
<td>0.2 ms</td>
<td>~1.8 ms on average</td>
</tr>
<tr>
<td>Inhibitory Synaptic Time Constant $\tau_{syn,inh}$</td>
<td>2 ms</td>
<td>~8 ms on average</td>
</tr>
</tbody>
</table>

Table 4.3: Synaptic Parameters.
is a list of all parameters and is subject to Gaussian mutations and uniform cross-overs during heredity. It is initialized with the current parameters perturbed by a percentage of choice. Implementations of other optimisation algorithms like the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) (Hansen 2011) or algorithms of the “inspyred” framework are provided, but have not been tested in detail.

The major difficulty is the limited number of evaluations that can be made: One fitness function evaluation takes about 200 seconds (for 700 ms simulations). We can therefore expect to run about 450 evaluations a day, a number that will not be sufficient to optimise many dependent parameters simultaneously. For every neuron population, the optimization therefore relied on 5 genomes with 4 generations only. Under these conditions, the genetic algorithm is not expected to display its full capacity - it rather amounts to a mere randomized sampling in the parameter space. Investing more computing power into the simplification process would thus be worthwhile (as it is reducing the runtime of simulations in turn) and is expected to provide room for considerable performance improvements.

4.3 Computational Considerations

The tasks described in the previous sections are carried out by the Python module “OPoNC” (Optimizing Point-Neuron Column) that creates a simplified version of any simulation. A documentation is available in appendix E, the code in appendix D.

As the NEST-simulations themselves already implement shared memory parallelization - with a load balancing close to perfect - and (optionally) distributed memory parallelization, further parallelization of the simulations would have been useless. The time-consuming functions setting up the simulation and replicating the experiment protocol however have been parallelized to a large extent with the Python “multiprocessing” module.

The implementation used NEST version 2.2.1 and the new numerics implementation “aeif_cond_alpha_RK5” (cf. chapter 3) compiled (without MPI) with gcc 4.4.7. The simulations were run with Python 2.6.6 (IPython 0.10) on one node of the Blue Brain Project Analysis and Visualization Facility that consists of 12 nodes with 2x Intel Xeon CPU X5690 (6 cores at 3.47GHz) and 6x 4GiB RAM running under Red Hat Enterprise Linux Server release 6.4. The NEST parallelization allowed a perfect load balancing on the 12 processor cores. On average, 16.5 GiB of memory were used.

Performing a single simulation of the simplified cortical column model for 700 ms (corresponding to one objective function evaluation) required a runtime of around 200 seconds. While this is on average 285 slower than real time, it is by a factor of roughly 5 faster.
than the full simulation (2 hours for 3500 ms of seven columns). However, the simulation can be run on 12 cores only, whereas its detailed counterpart is run on 4096 cores of the Blue Gene supercomputer - a considerable speed-up towards a real time simulation.

4.4 THE SIMULATION TO BE REPLICATED

The simulation to be replicated\(^1\) emulates an *in vitro* experiment *in silico* to observe the dynamics of a neocortical microcircuit that consists of 31,000 neurons connected by 32,056,430 synapses. It is constructed based on statistical information from electrophysiological experiments with 300 µm slices of the hind limb non-barrel somatosensory cortex S1 of P14-P16 rats. It comprises seven (hexagonal) columns that are arranged in one line and that were stimulated according to the protocol described in (Silberberg et al. 2004). Excitatory and inhibitory cells were brought to their threshold potential by current injections, dNAC and dSTUT cells were brought to 85% of their threshold and injected a hyperpolarizing current after a delay of 200 ms.

---

\(^1\)Path to simulation used:

For internal use only. Removed for online version.
In the following, the performance of a simplified model constructed as a proof of concept with the methodology described in chapter 4 is reported. To reduce the dimensionality of the optimisation problem, we restricted the parameter search to the parameters with the most direct effect on the firing rates: the threshold potential $V_{\text{th}}$ and the leak conductance $g_L$. All remaining parameters were kept at their default values (cf. table 5.1). The default values result from a fitting by Vij (2012) and have been previously used in point-neuron simulations of a neocortical column (Erő 2013a).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Fixed Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Membrane Capacitance $C_m$</td>
<td>73.05 pF</td>
</tr>
<tr>
<td>Resting Potential $V_{\text{rest}}$</td>
<td>-70.0 mV</td>
</tr>
<tr>
<td>Slope Factor $\Delta T$</td>
<td>2.0 mV</td>
</tr>
<tr>
<td>Adaptation Time Constant $\tau_w$</td>
<td>55.27 ms</td>
</tr>
<tr>
<td>Subthreshold Adaptation $a$</td>
<td>-5.165 pA</td>
</tr>
<tr>
<td>Spike-Triggered Adaptation $b$</td>
<td>111.8 pA</td>
</tr>
</tbody>
</table>

Table 5.1: Fixed parameter values for the optimization. $V_{\text{peak}} = 0$, the remaining technical parameters were chosen as in table 4.2 and the synaptic as in figure 4.3.

To further reduce the search space, we used an iterative optimisation in which the different populations are fitted one after the other, rather than simultaneously. This approach assumes that the populations are independent of each other. Even though this is not true, this approach was able to approximate the mean firing rates of the detailed simulation far better than the previously used parameter set (table 5.2 and figure 5.1).
Figure 5.1: Mean firing rates per population for the previous parameter set (Anirudh Vij, unpublished data) (A), the parameters fitted population-wise (B), and the selected parameter set (C).
Figure 5.1A shows the mean firing rates of the point-neuron simulation with the default parameters compared to those of the morphologically detailed simulation. Notice that even though only two parameter sets were used, rates differ largely due to the network structure. Most populations hardly match the detailed simulation, the simulation does not replicate the network dynamics of the detailed simulation. While firing rates in the layers 1 to 3 are generally underestimated, the contrary is the case for layers 4 to 6, where the firing rates are much higher than the target value. The performance of the simulation with the fitted parameters shown in figure 5.1B is far better: Here, most populations show a similar firing rate as in the full simulation. The rates in layers 1 to 3 are still underestimated, but in layers 4 to 6 most rates have decreased to a level close to the target value. Deviations are observed in L5_dist and L6_dist, but even more pronounced in L4_prox (that is underestimated) and pyramidal neurons in layers 4 and 5 (as well as 6, in a more moderate form) that are extremely underestimated. The parameters which result from the fitting are presented in table A.1 and figures A.1 and A.2 in the appendix. While \( g_L \) is similar to the previously used values, some threshold potentials become unphysically high in order to reduce the firing rate to the level observed in the morphologically detailed simulation. This may partly be a consequence of the fact that only a small fraction of the AdEx model parameters is fitted.

As some population rates matched less good than with the default parameters, we conducted another simulation (figure 5.1C) where the fitted parameter sets of L23_prox, L4_prox, L4_pyr, and L6_AIS were reset to their default values \( V_{th,exc} = -52 \text{mV}, V_{th,inh} = -57 \text{mV}, g_{L,exc} = 8.6 \text{nS}, \) and \( g_{L,inh} = 7.35 \text{nS} \). In addition, the population of layer 5 pyramidal neurons that performed very poorly was fitted again, this time based on the results and with more evaluations. The result was almost unchanged for the threshold potential, the leak conductance increased from 7.77 nS to 9.36 nS in this refit. Almost all performances further improved with this adjusted set of manually selected parameters (namely L23_prox and L6_AIS). The rates of pyramidal cells in layers 4 and 5 however remained far too high (9.8 Hz instead of 0.2 Hz for L5).
Table 5.2: Mean firing rates (in Hz) of the simplified simulation (for the previous parameter set, the fitted and the adjusted parameters) compared to the detailed simulation.

<table>
<thead>
<tr>
<th>Population</th>
<th>Detailed</th>
<th>Previous (±%)</th>
<th>Fitted (±%)</th>
<th>Adjusted (±%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1探</td>
<td>0.995</td>
<td>0.028 (-97.2%)</td>
<td>0.168 (-83.1%)</td>
<td>0.168 (-83.1%)</td>
</tr>
<tr>
<td>L23 AIS</td>
<td>2.745</td>
<td>0.771 (-71.9%)</td>
<td>0.320 (-88.3%)</td>
<td>0.105 (-96.2%)</td>
</tr>
<tr>
<td>L23 dist</td>
<td>0.339</td>
<td>0.250 (-26.3%)</td>
<td>0.335 (-1.2%)</td>
<td>0.207 (-38.0%)</td>
</tr>
<tr>
<td>L23探</td>
<td>2.367</td>
<td>1.630 (-31.1%)</td>
<td>0.442 (-81.3%)</td>
<td>1.767 (-25.3%)</td>
</tr>
<tr>
<td>L2探</td>
<td>0.108</td>
<td>0.402 (+272.2%)</td>
<td>0.534 (+394.4%)</td>
<td>0.191 (+76.9%)</td>
</tr>
<tr>
<td>L4 AIS</td>
<td>8.312</td>
<td>9.409 (+13.2%)</td>
<td>12.530 (50.7%)</td>
<td>10.152 (+22.1%)</td>
</tr>
<tr>
<td>L4 dist</td>
<td>0.933</td>
<td>2.985 (+220.0%)</td>
<td>0.514 (-44.9%)</td>
<td>0.284 (-69.6%)</td>
</tr>
<tr>
<td>L4探</td>
<td>6.455</td>
<td>10.034 (+55.4%)</td>
<td>0.651 (-90.0%)</td>
<td>10.433 (+61.6%)</td>
</tr>
<tr>
<td>L4探</td>
<td>1.385</td>
<td>5.590 (+300.7%)</td>
<td>8.252 (+495.8%)</td>
<td>8.039 (+480.4%)</td>
</tr>
<tr>
<td>L5 AIS</td>
<td>9.184</td>
<td>19.937 (+117.1%)</td>
<td>8.365 (-8.9%)</td>
<td>7.857 (-14.4%)</td>
</tr>
<tr>
<td>L5 dist</td>
<td>0.399</td>
<td>11.058 (+1746.1%)</td>
<td>3.180 (+430.9%)</td>
<td>1.854 (+209.5%)</td>
</tr>
<tr>
<td>L5探</td>
<td>8.281</td>
<td>21.517 (+159.8%)</td>
<td>8.836 (+6.7%)</td>
<td>7.116 (-14.1%)</td>
</tr>
<tr>
<td>L5探</td>
<td>0.191</td>
<td>27.288 (+14176.4%)</td>
<td>14.452 (+17466.3%)</td>
<td>9.843 (+5053.4%)</td>
</tr>
<tr>
<td>L6 AIS</td>
<td>14.176</td>
<td>18.949 (+33.7%)</td>
<td>9.974 (-29.6%)</td>
<td>12.051 (-15.0%)</td>
</tr>
<tr>
<td>L6 dist</td>
<td>1.533</td>
<td>11.809 (+670.3%)</td>
<td>3.518 (+129.5%)</td>
<td>2.126 (+38.7%)</td>
</tr>
<tr>
<td>L6探</td>
<td>12.632</td>
<td>18.320 (+45.0%)</td>
<td>10.944 (-13.4%)</td>
<td>8.830 (-30.1%)</td>
</tr>
<tr>
<td>L6探</td>
<td>2.361</td>
<td>17.589 (+645.0%)</td>
<td>5.561 (+135.5%)</td>
<td>3.747 (+58.7%)</td>
</tr>
</tbody>
</table>

Figure 5.3: Dendrite of pyramidal cell (red) reaching up in the column. © Blue Brain Project.
The bad fit for pyramidal neurons in layers 4 and 5 can possibly be explained by their anatomy: Their dendrites reach up far to the top layers of the column (figures 5.2 and 5.3), dendritic attenuation is therefore strong. As the point-neuron simulation does currently not account for attenuation by the dendritic tree, the postsynaptic potentials arriving at the soma are strongly overestimated. The unphysically high threshold potentials (around -30 mV) encountered for L5_pyr and L6_pyr (and L4_prox, as well) (figure A.1) could therefore be explained as a compensation for the too high postsynaptic potentials. Accounting for dendritic attenuation will therefore be an important next step towards improving the methodology developed here. Some work in this direction is undertaken by Wybo et al. (2013).

Our fitting procedure relied only on the mean firing rate of the respective populations. We can, thus, not expect that the temporal dynamics on short time-scales are also well described. This can be seen in figures 5.4 and 5.5, which show the actual spike activity of the detailed and simplified simulation, respectively. The point-neuron simulation shows a marked oscillating behaviour. Slight oscillations can also be observed in the detailed simulation, namely in layers 4 and 6, however not in this pronounced form. In the simplified simulation, oscillations are triggered by a strong activity of pyramidal cells in layers 4 and 5 (figure 5.4) and seem undamped. The overestimation of activity of L4_pyr and L5_pyr may thus provide an explanation for this behaviour. It is conceivable that the network dynamics of the simulations can be aligned more by accounting for attenuation and by proceeding to other statistical measures like firing rate distributions, inter-spike intervals, and cross-correlations.
Figure 5.5: Raster plots of (A) the full Simulation (Blue Brain Project, unpublished data) and (B) the point-neuron simulation with the adjusted parameters (where pyr-green, prox-red, dist-blue, and AIS-black).
In this thesis, I aimed at developing an automated approach to create simplified versions of morphologically detailed simulations. To this end, I first compared the dynamical properties of two of the most popular point-neuron models and introduced a unifying formulation that clarifies differences in the model components and allows switching between the models by translating the parameters. Next, I improved performance and accuracy of the Adaptive Exponential Integrate-and-Fire Model in NEST by implementing a 5th order Runge-Kutta method with adaptive stepsize control. This model was then used in a network simulation to replicate the dynamics of a computationally expensive Blue Brain simulation. Setting up such a simulation requires much care in emulating the experiment protocol as closely as possible. Model parameters of 17 populations were fitted within the network to match the mean firing rate observed in the detailed model. Results proved the feasibility of the method by approximating the firing rates. Neglecting the morphology of neurons lead to an overestimation of the firing rates of pyramidal cells due to the lack of dendritic attenuation. By correcting this effect and applying more sophisticated optimisation techniques, the method could construct point-neuron simulations that closely replicate the dynamical behaviour of morphologically detailed simulations.
BIBLIOGRAPHY


57


Csaba Erő. neurob, 2013b. (Cited on page 36.)


URL http://jp.physoc.org/content/556/1/19. PMID: 14978208. (Cited on page 47.)


Willem AM Wybo, Benjamin Torben-Nielsen, and Marc-Oliver Gewaltig. Synaptic synergies and their role in integrating distinct synaptic pathways. volume 14, page P140, 2013. (Cited on page 53.)

### Table A.1: Fitted values of the threshold potential and the leak conductance for the 17 populations.

<table>
<thead>
<tr>
<th>POP.</th>
<th>$V_{th}$ [mV]</th>
<th>$g_L$ [nS]</th>
<th>POP.</th>
<th>$V_{th}$ [mV]</th>
<th>$g_L$ [nS]</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1_prox</td>
<td>-61.7</td>
<td>8.31</td>
<td>L5_AIS</td>
<td>-41.9</td>
<td>7.66</td>
</tr>
<tr>
<td>L23_AIS</td>
<td>-42.3</td>
<td>5.01</td>
<td>L5_dist</td>
<td>-45.8</td>
<td>10.15</td>
</tr>
<tr>
<td>L23_dist</td>
<td>-56.8</td>
<td>8.87</td>
<td>L5_prox</td>
<td>-39.5</td>
<td>7.44</td>
</tr>
<tr>
<td>L23_prox</td>
<td>-47.2</td>
<td>9.27</td>
<td>L5_pyr</td>
<td>-28.0</td>
<td>7.77</td>
</tr>
<tr>
<td>L23_pyr</td>
<td>-53.3</td>
<td>12.13</td>
<td>L6_AIS</td>
<td>-48.9</td>
<td>9.57</td>
</tr>
<tr>
<td>L4_AIS</td>
<td>-57.6</td>
<td>10.36</td>
<td>L6_dist</td>
<td>-48.2</td>
<td>10.93</td>
</tr>
<tr>
<td>L4_dist</td>
<td>-44.8</td>
<td>7.36</td>
<td>L6_prox</td>
<td>-51.0</td>
<td>8.82</td>
</tr>
<tr>
<td>L4_prox</td>
<td>-31.0</td>
<td>8.87</td>
<td>L6_pyr</td>
<td>-32.7</td>
<td>10.0</td>
</tr>
<tr>
<td>L4_pyr</td>
<td>-52.7</td>
<td>12.58</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The previous values $V_{th,exc} = -52$mV, $V_{th,inh} = -57$mV, $g_{l,exc} = 8.6$nS, and $g_{l,inh} = 7.35$nS were the basis for the initialization and used in the manually selected parameter set for L23_prox, L4_prox, L4_pyr, L6_AIS. The refitting of L5_pyr performed for the selected parameter set resulted in $V_{th} = -28.2$mV and $g_L = 9.36$nS.
Figure A.1: Fitted threshold potential values. Previous values were $V_{\text{th,exc}} = -52\text{mV}$ and $V_{\text{th,inh}} = -57\text{mV}$. 
Figure A.2: Fitted leak conductance values. Previous values were $g_{L,\text{exc}} = 8.6\text{nS}$ and $g_{L,\text{inh}} = 7.35\text{nS}$. 
This section presents update formulae for the analytically solvable formulae of the AdEx model, as mentioned in chapter 3.4.

Equation 3.2 can be solved analytically for the initial conditions $w(t)|_{t=0} = w_0$ and $\frac{d}{dt}w(t)|_{t \to \infty} = \frac{a}{\tau_w} \cdot (V(t) - E_L)$. We choose $w_0 = 0$ as the initial conditions will anyways be given by the reset mechanism.

$$w(t) = (w_0 - \frac{a}{\tau_w} \cdot (V(t) - E_L)) \cdot \exp\left(-\frac{t}{\tau_w}\right) + \frac{a}{\tau_w} \cdot (V(t) - E_L) \quad (B.1)$$

From this, one can derive an update formula:

$$w(t + h) = \frac{a}{\tau_w} \cdot (V(t + h) - E_L) \cdot e^{-\frac{h}{\tau_w}} + \frac{a}{\tau_w} \cdot (V(t + h) - E_L) \cdot e^{-\frac{t}{\tau_w}} + \frac{a}{\tau_w} \cdot (V(t + h) - E_L) \cdot e^{-\frac{h}{\tau_w}}$$

$$= w(t) \cdot e^{-\frac{h}{\tau_w}} + \frac{a}{\tau_w} \cdot (V(t + h) - E_L) \cdot e^{-\frac{h}{\tau_w}}$$

$$= w(t) \cdot e^{-\frac{h}{\tau_w}} + \frac{a}{\tau_w} \cdot (V(t + h) - E_L) \cdot e^{-\frac{h}{\tau_w}}$$

Note the similarity of equation B.2 with the approximation $w(t + h) = w(t) \cdot \exp\left(-\frac{h}{\tau_w}\right) + \frac{a}{\tau_w} \cdot (V(t) - E_L) \cdot \left(1 - \exp\left(-\frac{h}{\tau_w}\right)\right)$ (Bernard et al. 1994, 272).

The differential equations modelling the synaptic dynamics can also be solved analytically. An exponential current is to be updated as follows:

$$I_{syn,exc}(t + h) = I_{syn,exc}(t) \cdot \exp\left(-\frac{h}{\tau_{syn,exc}}\right) \quad (B.3)$$
Updating the current corresponding to an exponential conductance kernel amounts to

\[
I_{\text{syn,exc}}(t + h) = I_{\text{syn,exc}}(t) \cdot \exp\left(-\frac{h}{\tau_{\text{syn,exc}}}\right) \cdot [V(t + h) - E_{\text{exc}}]
\]

For \(\alpha\)-functions, we use the analytical solution of the two 1st-order ODEs rather than the solution of the 2nd-order ODE (the kernel itself) directly. When the current is modelled, this amounts to

\[
dI_{\text{syn,exc}}(t + h) = dI_{\text{syn,exc}}(t) \cdot \exp\left(-\frac{h}{\tau_{\text{syn,exc}}}\right)
\]

\[
I_{\text{syn,exc}}(t + h) = I_{\text{syn,exc}}(t) \cdot \exp\left(-\frac{h}{\tau_{\text{syn,exc}}}\right) + dI_{\text{syn,exc}}(t) \cdot h \cdot \exp\left(-\frac{h}{\tau_{\text{syn,exc}}}\right)
\]

The update for the case of an \(\alpha\)-conductance is

\[
dg_{\text{exc}}(t + h) = dg_{\text{exc}}(t) \cdot \exp\left(-\frac{h}{\tau_{\text{syn,exc}}}\right)
\]

\[
g_{\text{exc}}(t + h) = g_{\text{exc}}(t) \cdot \exp\left(-\frac{h}{\tau_{\text{syn,exc}}}\right) + dg_{\text{exc}}(t) \cdot h \cdot \exp\left(-\frac{h}{\tau_{\text{syn,exc}}}\right)
\]

\[
I_{\text{syn,exc}}(t + h) = g_{\text{exc}}(t + h) \cdot (V(t + h) - E_L)
\]

A similar approach has been used in (Rotter and Diesmann 1999).