Modeling of Extracellular Potentials Measured by Microelectrode Arrays

Øystein Sørensen

Norwegian University of Life Sciences
Department of Mathematical Sciences and Technology
Master Thesis 30 credits 2010

"We are at the very beginning of time for the human race. It is not unreasonable that we grapple with problems. But there are tens of thousands of years in the future. Our responsibility is to do what we can, learn what we can, improve the solutions, and pass them on."

–Richard P. Feynman

Preface

This thesis is the fulfillment of my Master's degree at the Norwegian University of Life Sciences.

On the occasion, I would like to thank my supervisors, Professor Gaute T. Einevoll and Associate Professor Bjørn F. Nielsen, for inspiring weekly meetings and firm guidance in scientific thinking. Special thanks go to Gaute for sharing his vast knowledge of physics and neuroscience as well as including me in the computational neuroscience group; it has given great motivation to pursue an academic career.

My gratitude also goes to Dr. Johan Hake, for comprehensively answering all my questions about FEniCS and Linux, and to Dr. Klas H. Pettersen, for very instructive discussions about neural cable theory and useful comments on the manuscript.

Cheers go to my student colleagues for their exquisite sense of humor, serving to keep the spirit up during late hours in the office. I also want to thank my parents for nature and nurture, the latter of which they have been great providers since 1985.

Ås, 10th December, 2010

Øystein Sørensen
oystein_sorensen@hotmail.com

Abstract

Microelectrode arrays (MEAs) offer a promising way to study single-cell and network activity in cortical columns, with high spatial and temporal resolution. This thesis investigates models relating the measured local field potential (LFP) to the underlying neural activity. For the two-monopole approximation, a *point-source formula*, derived on the assumption of an infinite and homogeneous medium, was compared to the results of *finite element simulations*. The latter incorporated the saline solution, surrounding chamber, and electrode plate used in real MEA experiments. The impact of the electrical *conductivity profile* of tissue was also studied, for a two-monopole and a ball-and-stick neuron. The experimental set-up was seen to significantly affect the LFP measured by the MEA. Inhomogeneous and anisotropic tissue conductivity also influenced the LFP, but to a smaller extent. The results indicate that finite element methods improve modeling of MEA measurements, and should be preferred to the simple point-source formula.

Sammendrag

Mikroelektrodematriser muliggjør måling av både enkeltcelle- og nettverksaktivitet i biter av hjernebarken, med høy oppløsning i tid og rom. Formålet med denne masteroppgaven er å undersøke modeller som relaterer det elektriske potensialet målt på matrisen, til den underliggende nevrale aktiviteten. En punktkildeformel for et homogent og uendelig medium, ble sammenlignet med elementmetodesimuleringer som omfattet det virkelige fysiske oppsettet i slike eksperimenter. I tillegg ble betydningen av anisotropi og inhomogenitet i vevets elektriske ledningsevne undersøkt, både for en to-monopol og for et ball-og-pinne-nevron. De utførte simuleringene tyder på at det eksperimentelle oppsettet har en anselig påvirkning på potensialene som måles. Inhomogenitet og anisotropi var også av betydning, dog i mindre grad. Det konkluderes at elementmetoden er fordelaktig for modellering av elektriske potensialer i mikroelektrodematrisemålinger, og bør foretrekkes framfor den enkle punktkildeformelen.

Contents

Chapter 1

Introduction

Microelectrode array (MEA) measurements are a promising way of recording neural activity. The method has traditionally been used to study *cultured cells*, but succesful applications with *brain slices* have recently been reported [10, 20]. Considered here is *single-cell stimulation*, in which a current-supplying electrode penetrates the soma of a neuron in the slice. Currents traversing the cell membrane are generated as a response to the stimulus, changing the *extracellular field potential* of surrounding tissue. The MEA readily measures electric potentials on the base of the slice, with high spatial and temporal resolution.

In order to interprete the recordings, an important question is: *What do the measured potentials tell us about the underlying neural activity?* This poses an inverse problem; that of finding the transmembrane currents most likely to have set up the potentials measured. Comparison with MEA potentials obtained *in silico* by forward modeling can be useful to obtain such insight. This requires a good neuron model for calculating transmembrane currents, from which the extracellular potentials are predicted by means of a *volume conduction model* [60].

A simple analytic expression for the extracellular potentials arising from membrane currents can be used, when the tissue is assumed to be an *infinite medium* of *isotropic* and *homogeneous* electrical conductivity. The reference potential is set to zero infinitely far away from the sources, similar to the approach used to calculate the potential field set up by charge distributions in electrostatics. The current source is divided into N discrete *point sources*, giving the formula

$$\phi(r) = \frac{1}{4\pi\sigma} \sum_{n=1}^{N} \frac{I_n}{r_n}, \tag{1.1}$$

where $\phi(r)$ is the electric potential at a point r, r_n is the distance from source n to the field point r, I_n is the current entering the volume conductor at source n, and σ is the electrical conductivity of the medium. A similar line-source formula can also be derived [41, 44].

In real MEA measurements, the *infinite medium assumption* is not immediately justified. A brain slice, of extent on the order of cubic millimeters, is im-

mersed in artificial cerebrospinal fluid (ACSF) [9, 10]. This *saline solution* is again surrounded by the electrically insulating walls of an experimental chamber, and air[1] [19]. Hence, the conductivity is practically zero a few millimeters from the current sources, contrary to the assumption of an infinite and homogeneous medium. Mathematically, the air interface is modeled by a homogeneous Neumann boundary condition. In addition, a reference electrode is typically present, giving a Dirichlet condition to take into account [10, 19, 20]. Furthermore, saline has higher conductivity than tissue, and the conductivity of the slice may be depending both on position and direction [22, 44, 51].

Analytic expressions incorporating boundary conditions, saline, and conductivity profile, are, if they exist, very complicated. Numerical computation of the extracellular potential may thus be a good alternative. With the *finite element method* (FEM), the whole experimental set-up can be modeled with a high level of detail. FEM implementations are relatively laborious, so it is of interest to probe how much of an improvement they offer compared to using analytic formulae based on simplifying assumptions.

This study investigates the impact of the experimental set-up and tissue conductivity profile on MEA potentials. The goal is to gain insight into the level of detail needed for accurate modeling of MEA measurements. The *two-monopole* approximation and the *ball-and-stick* neuron are used to generate transmembrane currents [35, 43, 44]. A finite element formulation of the boundary value problem for volume conduction has been implemented using FEniCS [2]. An additional goal of the work has been to investigate how FEniCS can be employed to simulate extracellular potentials, and to serve as a starting point for more sophisticated use of the tool.

The next chapter surveys necessary biophysical background, and the typical experimental set-up used in MEA mesurements is described. Chapter 3 gives an introduction to the use of finite element methods for solving boundary value problems, and chapter 4 describes how the models derived were implemented in FEniCS. Simulation results are presented in chapter 5 and discussed further in chapter 6.

[1] Cf. figure 2.10 on page 21

Chapter 2

Background

2.1 Bioelectricity

2.1.1 Neural Electrical Activity

This section is partly based on the book by Nelson [39].

The human brain consists of about 10^{12} cells, that can be divided into *neurons* and *glia*. The glial cells are more numerous than neurons, of which there are about 10^{11}. However, the information processing done by the brain is presumably performed mainly by neurons. Figure 2.1 shows a sketch of a cortical pyramidal neuron [11, 15, 29].

Hierarchical response to stimuli is a hallmark of the nervous system. *Neocortex*, often simply called cortex, can be thought of as the highest level, in which specific representations are made [11, 29]. It consists of highly specialized domains. For example, area V1 represents visual orientation, and the columns of the rat barrel cortex respond specifically to stimulation of a particular whisker [17]. Figure 2.2 shows an illustration of a column from the rat cortex. Note that this figure only sketches the cell types present. In reality, nervous tissue is densely packed, with an extracellular volume fraction of less than 20 % [26, 33].

The *soma* is the main body of the neuron and contains the same organelles as are found in other mammalian cells. Functions like transcription, translation, protein synthesis, and energy-releasing conversion of ATP to ADP, take place in the soma [11]. What mainly distinguishes neurons from other cells is their *axons* and *dendrites*. The numerous dendrites receive input from other neurons, and the axons send output on to yet others. This information processing takes place in the form of electric currents, so much of the function of neurons can be understood by their electrical properties.

The *intracellular* and *extracellular* fluids differ significantly in their concentrations of different ion species. Table 2.1 gives some illustrative values. The large molecules inside the cell, like DNA, are acidic, carrying a net negative charge. Their concentration is equivalent to that of 125 mM electrons. The lipid bilayer cell membrane is practically impermeable, but embedded in it are specialized channels

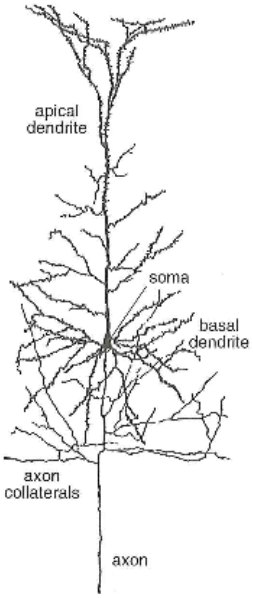

Figure 2.1: ***Sketch of a pyramidal neuron.*** *The soma is the main body of the cell, from which axons and dendrites protrude. For pyramidal neurons, the numerous dendrites both make up a basal bush close to the soma and extend apically. Output is sent through the axon and its collaterals. Taken from Dayan and Abbott [15].*

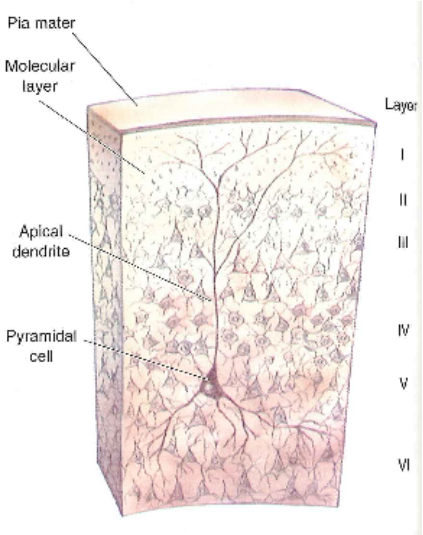

Figure 2.2: ***Illustration of a cortical column.*** *Columns are divided into 6 characteristic layers, numbered from I to VI. A pyramidal cell (figure 2.1) is situated in layer V, with dendrites extending through the whole column. Taken from Bear et al. [11].*

Ion	c_e	c_i	$\mathrm{V}^\mathrm{Nernst}$
K^+	5	100	-80 mV
Na^+	150	15	62 mV
Ca^{2+}	2	2e-4	123 mV
Cl^-	150	13	-65 mV
Macromolecules (-)	0	125	-

Table 2.1: ***Typical ionic concentration values.*** *c_e and c_i are the extracellular and intracellular concentrations, respectively, and their values are given in millimolar (mM). The interior of the cells contain acidic macromolecules which make up an equivalent of 125 mM electrons. The rightmost column shows the Nernst potential of the ions (equation 2.4). Data from Bear et al. [11] and Nelson [39].*

through which ions can cross. Macromolecules, on the other hand, are not able to escape. Some channels show a strong selectivity to particular ion species, and their permeability may depend on environmental variables like ionic concentrations or the electric potential across it. Overall, the net effect of ion channels is to give the membrane some electrical conductivity, g_n, and permeability, P_n, to ion species n.

In electrostatic equilibrium the bulk interior and exterior have to be electrically neutral; due to mutual repulsion, charges of the same type minimize their potential energy by being as far away from each other as possible. But the membrane stops them, and a layer of net charge establishes on its intracellular side, attracting charges of opposite type from the extracellular medium. Neither these are able to cross the lipid bilayer. Thus, an electric potential gradient is established between the the two sides, and the cell membrane acts as a parallel-plate capacitor.

The concentration differences are maintained by ion pumps embedded in the membrane. An example is the *sodium-potassium pump*. It uses ATP to transport 3 Na^+ ions out of the cell per 2 K^+ ions pushed in. Hence, both a concentration gradient is set up and a net outward current of one electron per cycle is maintained. The *calcium pump* transports Ca^{2+} ions out of the cell, thereby being responsible for the very low intracellular calcium concentration. These pumps oppose passive current flow and keep the concentrations relatively constant. Thus, in the rest of the section we consider the values in table 2.1 as given and neglect the particular currents set up by ion pumps. A justification for this approach is given in section 12.1.2 of reference [39].

The connections between neurons are called *synapses* and may be characterized as either *chemical* or *electrical*. In mammals, the chemical synapses by far outnumber their electrical counterparts [11]. Stereotypically, input from *presynaptic* cells is received at the synapses of the dendrites and soma, whereas output to *postsynaptic* cells is transmitted through the axons [15].

Electrical synapses are gap junctions, channels through which ions may directly travel from one neuron to another. An action potential in a presynaptic cell then causes a rapid increase in the postsynaptic potential of the cells to which it is connected via electrical synapses. If a neuron receives sufficient input to raise

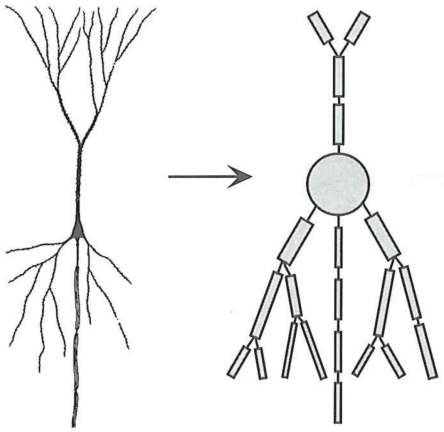

Figure 2.3: ***Compartmental modeling.*** *The figure illustrates how a pyramidal neuron may be divided into compartments, each assumed to have a constant potential gradient across its membrane. Taken from Dayan and Abbott [15].*

its membrane potential above threshold, an action potential will be fired. Hence, electrical synapses connecting groups of neurons mediate fast synchronization of firing [11].

Chemical synapses work in a more indirect way. An action potential arriving at a terminal causes release of neurotransmitters into the synaptic cleft. The neurotransmitters then diffuse over to the postsynaptic side, where they influence the behavior of active channels. If the synapse is *excitatory*, the net effect of the neurotransmitter release is to open channels, causing a positive current into the cell. This contributes to the depolarization of the postsynaptic neuron. *Inhibitory* synapses work in the opposite way, i.e., by closing channels, causing negative current flow into the postsynaptic cell hyperpolarizing the neuron [11].

2.1.2 One-Compartment Model

This section is partly based on the book by Nelson [39].

Neurons have a complex morphology, and input and output are constantly transmitted at numerous parts of their membrane through synaptic connections. Hence, the transmembrane potential is varying throughout the cell body. If the neuron is divided into many small parts, we can assume each such *compartment* to be at approximately constant potential, as illustrated by figure 2.3. This section will present the electrical properties of a single compartment, and section 2.1.3 will explain how they are connected. Eventually we will let their length go to zero, in order to obtain the *cable equation*.

Consider a membrane only permeable to potassium. With the values of table 2.1, Fick's first law predicts an outward diffusive flux,

$$j_{\mathrm{diff,K}} = -P_{\mathrm{K}}(c_{\mathrm{e,K}} - c_{\mathrm{i,K}}). \tag{2.1}$$

Due to the membrane potential,

$$\Delta V = V_{\mathrm{i}} - V_{\mathrm{e}}, \tag{2.2}$$

there will also be an electrical flux, with positive direction towards the extracellular medium,

$$j_{\mathrm{el,K}} = g_{\mathrm{K}} \Delta V, \tag{2.3}$$

where the conductance g_{K} in principle may be any function. Therefore, in the case of potassium, an *entropic force* is pushing ions out of the cell, driving the flux in equation (2.1), whereas an *electric force* is pulling current in, when $V_{\mathrm{i}} < V_{\mathrm{e}}$ in equation (2.2).

An equilibrium is reached when the net flux is zero. This happens at the *Nernst potential*, in general given by

$$V_n^{\mathrm{Nernst}} = -\frac{k_{\mathrm{B}} T}{ze} \ln(c_{\mathrm{e},n}/c_{\mathrm{i},n}), \tag{2.4}$$

where z is the valency of ion species n, k_{B} is the Boltzmann constant, T is the temperature, and $c_{\mathrm{e},n}$ and $c_{\mathrm{i},n}$ are the extracellular and intracellular concentration of n, respectively. Accordingly, the net current is

$$I_n = (\Delta V - V_n^{\mathrm{Nernst}}) g_n S,$$

where S is the membrane surface area. When the actual membrane potential is higher than the Nernst potential, there will be a net outward current, and when it is lower, there will be a net inward current.

Because the membrane is permeable to several ion species, whose Nernst potentials differ, currents are in general always present. A situation of interest is the *steady state*, at which the net charge transport across the membrane is zero. This happens when

$$I = \sum_n (\Delta V - V_n^{\mathrm{Nernst}}) g_n S = 0,$$

and the sum goes over all relevant ions. A little algebraic manipulation shows that the steady state is obtained when ΔV equals the *resting potential*,

$$V^0 = \frac{\sum_n V_n^{\mathrm{Nernst}} g_n}{\sum_n g_n}. \tag{2.5}$$

The numerator is a sum of individual Nernst potentials weighted by their respective conductivities. Hence, V^0 tends to be closer to the equilibrium potential of ions which more easily traverse the membrane. At rest,

$$g_{\mathrm{K}} \approx 25 g_{\mathrm{Na}} \approx 2 g_{\mathrm{Cl}},$$

seemingly in accordance with the actual value around -65 mV, close to the Nernst potential of K^+ and Cl^-, but far from that of Na^+.

Changes in ionic concentrations or membrane potential may cause abrupt alterations of particular conductivity values. This nonlinear behavior of gating variables is what generates action potentials and can be described by Hodgkin-Huxley-like models [15]. Izhikevich [28] gives a good introduction to the nonlinear dynamics of spiking neurons. Since the main focus of this thesis is on subthreshold phenomena, we will not go into the details of the action potential. Membrane conductivities are rather assumed to be constant. This approach can be justified as a first order approximation close to the resting potential.

Using ideas from basic circuit theory, the membrane can be pictured as containing conductors and capacitors coupled in parallel. The total conductance is then a linear sum over the individual conductivity values [57],

$$g_{\mathrm{tot}} = \sum_n g_n.$$

Hence, the conductive current crossing the membrane area is

$$I = g_{\mathrm{tot}} S(\Delta V - V^0).$$

This equation clearly shows that if the membrane potential exceeds the resting potential net current will flow outward, which by convention is the positive direction. If $\Delta V < V^0$, current will go into the cell.

The amount of charge on either side of the membrane is

$$q = c_{\mathrm{m}} S \Delta V,$$

where c_{m} is the capacitance per unit surface area. The capacitive current is the temporal derivative of the net charge,

$$i_{\mathrm{C}} = \frac{\partial q}{\partial t} = c_{\mathrm{m}} S \frac{\partial}{\partial t} \left(\Delta V \right).$$

The corresponding circuit diagram is shown in figure 2.4, where

$$R_{\mathrm{m}} = 1/(g_{tot} S)$$

is the resistance, and

$$C_{\mathrm{m}} = c_{\mathrm{m}} S$$

is the capacitance of the membrane area considered [15].

2.1.3 Cable Equation

This section is partly based on chapter 6 in the book by Dayan and Abbott [15].

Excitatory input causes a localized membrane potential increase close to the synapse. Hence, the equivalent circuit model assuming a uniform intracellular potential cannot be used for the whole neuron. As mentioned at the beginning of the last section, compartmental modeling provides a fix. For example, assuming a

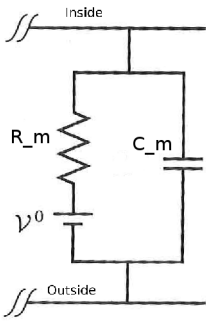

Figure 2.4: ***Equivalent circuit of a cell membrane.*** *A patch of cell membrane can be modeled by a parallel connection of a resistor and a capacitor. The difference between the membrane potential and the Nernst potential (V^0) drives the resistive (conductive) current. Modified from Nelson [39].*

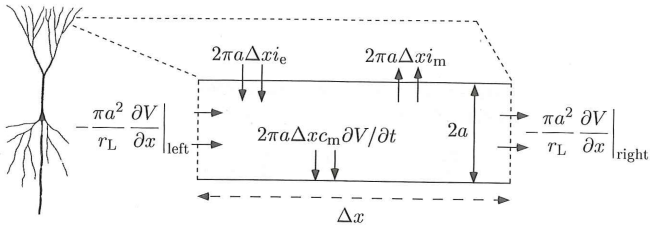

Figure 2.5: ***Magnified view of a compartment.*** *A small dendritic section can be modeled as a cylindrical cable at constant potential. The equivalent circuit is illustrated. The membrane current density, i_m, and the resistance per unit length, r_L, correspond to j_m and r_i in section 2.1.3, respectively. i_e denotes electrode current applied across the dendritic membrane, and will not be considered here. Taken from Dayan and Abbott [15].*

uniform potential inside soma, this can be one compartment. The dendrites can be split in their longitudinal direction into thin parts at a uniform potential. Figure 2.5 illustrates one compartment on a distal dendrite.

As explained, the axons and dendrites are partitioned into cable segments. For a cylindrical segment of surface area $S = 2\pi a \Delta x$, where a is its radius and Δx the length, the capacitive current across its membrane is given by

$$i_{\mathrm{C}} = 2\pi a \Delta x c_{\mathrm{m}} \frac{\partial V}{\partial t},$$

where V is the potential difference between the inside and outside (ΔV of equation (2.2)). The Δ has been dropped for notational convenience.

By Ohm's law, the volume current inside the cell is

$$\mathbf{j}_{\mathrm{i}} = -\frac{1}{r_{\mathrm{i}}} \nabla V,$$

where r_{i} is the intracellular resistance times unit length. Its longitudinal component is

$$j_{\mathrm{i}}(x) = -\frac{1}{r_{\mathrm{i}}} \frac{\partial V}{\partial x}, \tag{2.6}$$

where the x-direction is taken to be along a compartment. Assuming the potential to be radially uniform, we obtain the total current by multiplying $j_{\mathrm{i}}(x)$ with the cross sectional area πa^2. Then, the current entering the segment in the end closer to the soma is

$$i_{\mathrm{i}}(x) = -\frac{\pi a^2}{r_{\mathrm{i}}} \frac{\partial V}{\partial x}\Big|_x,$$

and the current leaving through the end of the segment is

$$i_{\mathrm{i}}(x + \Delta x) = -\frac{\pi a^2}{r_{\mathrm{i}}} \frac{\partial V}{\partial x}\Big|_{x+\Delta x}.$$

Since the net current into a compartment, by Kirchhoff's current law, has to be zero, the conductive transmembrane current (positive outwards) is given by

$$i_{\mathrm{g}} = i_{\mathrm{i}}(x) - i_{\mathrm{i}}(x + \Delta x) - i_{\mathrm{C}}.$$

Written in terms of the current density crossing the membrane, j_{g}, it becomes

$$i_{\mathrm{g}} = 2\pi a \Delta x j_{\mathrm{g}}.$$

Kirchhoff's current law can now be stated as

$$2\pi a \Delta x c_{\mathrm{m}} \frac{\partial V}{\partial t} = -\left(\frac{\pi a^2}{r_{\mathrm{i}}} \frac{\partial V}{\partial x}\right)\Big|_x + \left(\frac{\pi a^2}{r_{\mathrm{i}}} \frac{\partial V}{\partial x}\right)\Big|_{x+\Delta x} - 2\pi a \Delta x j_{\mathrm{g}}.$$

Letting the length of each compartment tend to zero[1] while assuming the segments to have constant radii between junctions, the *linear cable equation* becomes

$$\tau_{\mathrm{m}} \frac{\partial V}{\partial t} = \lambda^2 \frac{\partial^2 V}{\partial x^2} - V, \tag{2.7}$$

[1] For a finite stick length, this is equivalent to letting the number of compartments tend to infinity.

where

$$j_{\mathrm{g}} = \frac{V}{r_{\mathrm{m}}}$$

has been used. The membrane time constant, τ_{m}, is given by

$$\tau_{\mathrm{m}} = r_{\mathrm{m}} c_{\mathrm{m}},$$

and

$$\lambda = \sqrt{\frac{a r_{\mathrm{m}}}{2 r_{\mathrm{i}}}}$$

is the electrotonic length. Equation (2.7) describes a leaky cable. For each infinitesimal segment, some current will pass out though the membrane while the remainder goes on to the next segment. By extending the general principles considered here, active conductances as well as synapses along the cable can be incorporated [34, 40].

The total membrane current is the sum of the conductive and capacitive contribution,

$$j_{\mathrm{m}} = c_{\mathrm{m}} \frac{\partial V}{\partial t} + \frac{V}{r_{\mathrm{m}}},$$

which by reorganization of terms in the cable equation can be shown to equal

$$j_{\mathrm{m}} = \frac{\lambda^2}{r_{\mathrm{m}}} \frac{\partial^2 V}{\partial x^2}.$$

This is the current entering the extracellular space, generating the electric potentials measured on the MEA.

For any compartment, $V = V_{\mathrm{i}} - V_{\mathrm{e}}$ is the difference between the potential inside and just outside the membrane. However, the fluctuations of the extracellular potential typically are small compared to the intracellular variations. Hence, when calculating the transmembrane currents, V is just set to the difference between the intracellular potential and ground, i.e., the extracellular potential is assumed constant [35, 44]. Since the membrane currents are subsequently used to compute extracellular potentials, this introduces an inconsistency. By explicit modeling of both the intracellular and extracellular space, this simplification would be avoided [21]. However, for the subthreshold phenomena considered here it seems well justified [27].

2.1.4 Volume Conduction

Up to now, membrane currents have been discussed. This section goes on to describe how they set up an extracellular electric potential and is partly based on the book by Nunez and Srinivasan [41].

A volume V of charge density ρ confines a total charge

$$Q = \iiint_V \rho \mathrm{d}V.$$

By charge conservation, the negative rate of change of Q equals the net current passing out through ∂V, the surface enclosing the volume. The latter is given by the surface integral of the current density[2], $\mathbf{J}$, over ∂V, so

$$-\frac{\partial Q}{\partial t} = \iint_{\partial V} (\mathbf{J} \cdot \mathbf{n}) \, \mathrm{d}A,$$

where $\mathbf{n}$ is an outward unit normal. Combining these two equations gives

$$-\frac{\partial}{\partial t} \iiint_{V} \rho \mathrm{d}V = \iint_{\partial V} (\mathbf{J} \cdot \mathbf{n}) \, \mathrm{d}A.$$

The temporal derivative on the left-hand side can be put inside the integral, assuming the volume does not change with time. The right-hand side can be rewritten using the divergence theorem. This yields

$$-\iiint_{V} \frac{\partial \rho}{\partial t} \mathrm{d}V = \iiint_{V} (\nabla \cdot \mathbf{J}) \, \mathrm{d}V.$$

Since the integrals on both sides are over V, also the arguments have to equal. This gives the continuity equation [23],

$$\nabla \cdot \mathbf{J} = -\frac{\partial \rho}{\partial t}. \tag{2.8}$$

Assuming an *Ohmic* medium[3], the volume current can be expressed as

$$\mathbf{J} = \sigma \mathbf{E}, \tag{2.9}$$

where $\mathbf{E}$ is the electric field and σ the electrical conductivity of the medium. In biological tissue, the time-derivative of the magnetic field induced by the currents is negligible, so Faraday's law of induction,

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t},$$

can be simplified to

$$\nabla \times \mathbf{E} = 0.$$

Using the mathematical identity

$$\nabla \times (\nabla f) = 0$$

for any scalar f, it is clear that the electric field can be expressed as the gradient of ϕ, a *scalar potential field*,

$$\mathbf{E} = -\nabla \phi. \tag{2.10}$$

The minus sign is by convention [23, 44].

[2] **Boldface** is here used to represent vector quantities.

[3] Explained in section 1.2.1 of reference [44].

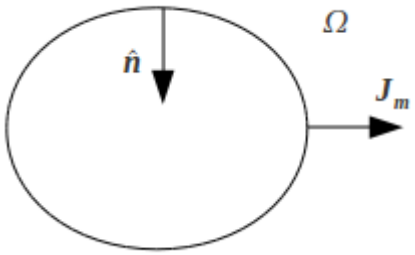

Figure 2.6: ***Neuron membrane modeled as inner boundary.*** *The outward unit vector points into the hole, which is not part of the computational domain, Ω. J_{m} is the current density on the boundary and represents current crossing the cell membrane.*

It follows from insertion of equation (2.10) into (2.9) that the volume current can be expressed as

$$\mathbf{J} = -\sigma \nabla \phi.$$

Plugging this into the continuity equation (2.8) we get Poisson's equation,

$$\nabla \cdot (\sigma \nabla \phi) = \frac{\partial \rho}{\partial t}. \tag{2.11}$$

The conductivity σ may in general be a 3×3 matrix. In the case of an isotropic medium, it is a scalar.

In experiments it is possible to stimulate one or a few neurons. Upon repeated trials, the background noise of all other cells is averaged out, thereby making it possible to model the electrical behavior of a few neurons in the tissue [15]. In the following, only one stimulated neuron will be considered. All others are part of the passive volume conductor and their presence only expressed indirectly through the tissue conductivity. Instead of specifying a source function for the stimulated neuron, its membrane can represented by an inner boundary [21]. Transmembrane currents enter the volume conductor through Neumann boundary conditions. With no current sources in the extracellular space, Poisson's equation reduces to Laplace's equation,

$$\nabla \cdot (\sigma \nabla \phi) = 0. \tag{2.12}$$

Now consider an inner hole in a three-dimensional domain, as illustrated by figure 2.6. On the membrane, the scalar radial outward current is

$$-\mathbf{J}_{\mathrm{m}} \cdot \mathbf{n},$$

where $\mathbf{J}_{\mathrm{m}}$ is the current density across the surface and $\mathbf{n}$ is the outward unit normal of the extracellular space. The extracellular current is given by

$$\mathbf{J}_{\mathrm{e}} = -\sigma \nabla \phi.$$

Because of continuity the volume current perpendicular to the membrane, very close to its surface, has to equal the current density crossing the membrane [23], i.e.,

$$-\mathbf{J}_{\mathrm{e}} \cdot \mathbf{n} = -\mathbf{J}_{\mathrm{m}} \cdot \mathbf{n}.$$

This gives

$$-\mathbf{J}_{\mathrm{e}} \cdot \mathbf{n} = -\left(-\sigma \nabla \phi \cdot \mathbf{n}\right) = \sigma \frac{\partial \phi}{\partial n}.$$

In the case of a long, thin cable, figure 2.6 can be thought to show a cross section through which the intracellular potential is constant. For a single compartment, e.g., the soma, the potential is constant by definition. Hence, the current density will at any time have the same value throughout the surface and be directed normal to the membrane, i.e.,

$$-\mathbf{J}_{\mathrm{m}} \cdot \mathbf{n} = J_{\mathrm{m}},$$

where J_{m} is the scalar membrane current density, positive outwards. The boundary condition for the extracellular potential on the membrane of the active neuron hence becomes

$$\sigma \frac{\partial \phi}{\partial n} = J_{\mathrm{m}}. \tag{2.13}$$

An approach similar to the one described in the last paragraphs, i.e., modeling the membrane of the active neuron as an inner boundary, was also used by references [21, 27, 51].

2.1.5 Extracellular Conductivity

The electrolytic extracellular medium makes up less than 20 % of cortical tissue. However, since its conductivity is much higher, volume currents tend to flow extracellularly rather than through the hardly penetrable cell membranes. Hence, a piece of passive tissue can be imagined as packed with almost impenetrable neurons immersed in a conductive fluid [11, 22, 26].

Despite this, the tissue conductivity, σ, is often assumed to be a constant scalar [35, 43, 44]. The assumption is that microscale-fluctuations average out, giving macroscale homogeneity. A justification of this *homogenization approximation* can be found in appendix A of G. Holt's PhD thesis [26]. In that work, an array of axons, i.e., nonconductive membranes, were explicitly modeled. The extracellular potentials were compared with those obtained by assuming the whole domain to be extracellular, containing volume current sources. The results showed that the two approaches were practically equivalent. In this thesis, a current source in an extracellular domain has been modeled, equivalent to the last approach done by Holt [26].

Goto et al. [22] measured the conductivity profile of rat barrel cortex. Statistically siginficant anisotropy was measured in layers II/III and V, although with the conductivity parallel to the column not larger than 1.5 times the value in the

perpendicular direction. Inhomogeneities and anisotropies were found throughout the column but not pronounced enough to reject the null hypothesis. The modeling study by Holt [26] reported an anisotropy ratio of about 5-6.

With a proper choice of coordinate system, anisotropy may in general be modeled by representing the bulk conductivity as a second order diagonal tensor,

$$\sigma = \begin{bmatrix} \sigma_{xx} & 0 & 0 \\ 0 & \sigma_{yy} & 0 \\ 0 & 0 & \sigma_{zz} \end{bmatrix},$$

where σ_{ii} is the conductivity along the i direction in the coordinate system chosen [56]. In the case of modeling a cortical column, cylindrical symmetry is assumed, and the principal directions are perpendicular to the column (y and z) and parallel to it (x). In accordance with Goto et al. [22], it can then be written

$$\sigma = \begin{bmatrix} \sigma_{\parallel} & 0 & 0 \\ 0 & \sigma_{\perp} & 0 \\ 0 & 0 & \sigma_{\perp} \end{bmatrix}. \tag{2.14}$$

The condictivities parallel and perpendicular to the column are represented by $\sigma_{\parallel}$ and $\sigma_{\perp}$, respectively. It should also be pointed out that in the case of anisotropy, $\sigma\nabla\phi$ becomes a matrix-vector product, so the electric field will still be a three-dimensional vector.

Possible frequency-dependence of extracellular conductivity, as suggested by, e.g., Bedard et al. [12], will not be modeled in this work.

2.2 Neuron Models

Realistic calculations of transmembrane currents in morphologically reconstructed neurons require a simulation environment like NEURON [4]. However, when the passive electrical properties of cells are studied, more simplified models can be utilized.

In slice experiments, the MEA is typically situated a few hundred microns below the active neuron. Hence, the top priority of both the cell model and the volume conduction model should be that they sufficiently reproduce potentials on the electrodes this distance away. Good simplified models may allow analytical solutions, which are important in order to investigate, e.g., parameter dependences and power laws [43]. In this work, where the main scope is to study the impact of boundary conditions and conductivity profile, the details of the neuron model are presumably not critical for the results obtained.

Two quite simple models will be used. The following sections present these, along with the assumptions on which they rest. To obtain a coherent explanation, the derivation of particular expressions used in this work will also be shown. The dipole approximation, typically valid for distances more than 1 mm away from the

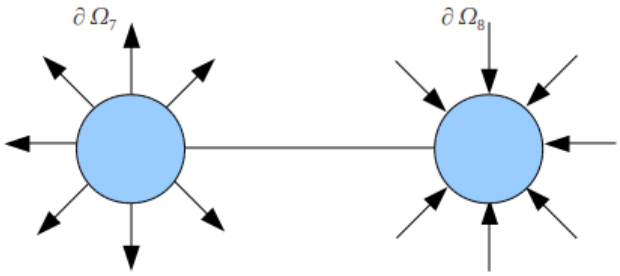

Figure 2.7: ***Two-monopole model.*** *The arrows show the direction of the current; the left monopole is a source, and the right is a sink.*

neuron, has not been used, since the far-field limit is never reached in MEA slice measurements[4].

2.2.1 Two-Monopole Approximation

Excitatory synaptic stimulation results in an inward current. Hence, net positive charge leaves the extracellular space, for example by Na^+ entering the cell or Cl^- leaving it. According to Kirchhoff's current law, the net current into the cell must be zero at any time. This means that a return current of equal size as the synaptic input current must instantaneously leave the neuron. For synaptic stimulation at the dendrites, the return current typically has its weighted mean position close to the soma, especially frequency components of the input current below about 10 Hz [35,44]. This lets us model the neuron by two monopoles, a *sink* situated at the site of synaptic stimulation and a *source* at the soma [35].

Laplace's equation (2.12) determines the potential in the passive extracellular medium, and the two monopoles are modeled as spherical inner boundaries. Ω denotes the whole extracellular space, while $\partial\Omega_7$ and $\partial\Omega_8$ are the boundaries surrounding each of the two monopoles[5]. Assume further that $\partial\Omega_8$ represents a point on a dendrite receiving excitatory synaptic stimulus. Then, by Kirchhoff's current law, the boundary conditions on the two monopoles become

$$\sigma\frac{\partial\phi}{\partial n} = J_{\mathrm{m}}(t) \quad \text{on } \partial\Omega_7$$

and

$$\sigma\frac{\partial\phi}{\partial n} = -J_{\mathrm{m}}(t) \quad \text{on } \partial\Omega_8.$$

Figure 2.7 illustrates the model.

Lindén et al. [35] set the source and sink current to give the same dipole moment as found by compartmental modeling. Here, the two-monpole will be used for:

[4] See e.g., figure 6 in reference [35].

[5] The reason for this numbering will become clear in section 4.3.

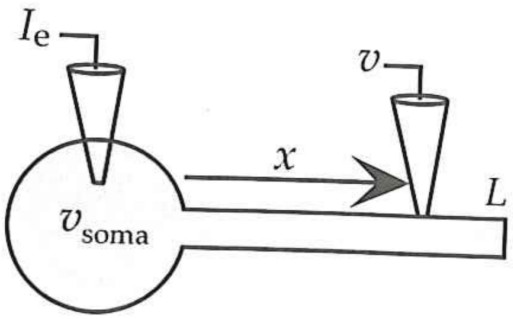

Figure 2.8: ***Ball-and-stick model.*** *Schematic illustration of a ball-and-stick neuron with current stimulation, I_e, in the soma. L, the cable length, is denoted by l in section 2.2.2. Taken from Dayan and Abbott [15].*

- Testing the numerical accuracy.
- Comparison of the FEM solution, explicitly modeling the experimental set-up, to the analytical solution, which assumes an infinite and homogeneous medium.
- Studying the impact of anisotropic and inhomogeneous tissue conductivity.

Hence, the monopole currents will be somewhat arbitrarily chosen because their amplitude and power law behavior are not essential for the problems to be investigated.

2.2.2 Ball-and-Stick Neuron

The ball-and-stick model is based on work by Wilfrid Rall from the late 1950's through the 60's [48]. It models the soma as one spherical compartment and maps the entire dendritic tree into an *equivalent cylinder*. The somatic transmembrane currents are straightforwardly calculated from the soma potential by modeling the membrane as containing a resistor and a capacitor in parallel. The stick is a cylinder, and both its membrane potential and the resulting currents are given by the cable equation (2.7) with appropriate boundary conditions. Figure 2.8 shows an illustration of the model.

We will here consider a ball-and-stick model with a sinusoidal current stimulus,

$$I_{\mathrm{e}} = I_0 \cos(2\pi f t) = I_0 \cos(\omega t), \tag{2.15}$$

in the soma. The injected current faces a parallel connection of the ball and the stick, whose complex[6] admittances are $\mathbf{Y}_{\mathrm{ball}}$ and $\mathbf{Y}_{\mathrm{stick}}$, respectively [42]. From

[6] **Boldface** here symbolizes complex variables. In the rest of this thesis, the exact usage will be apparent from the setting.

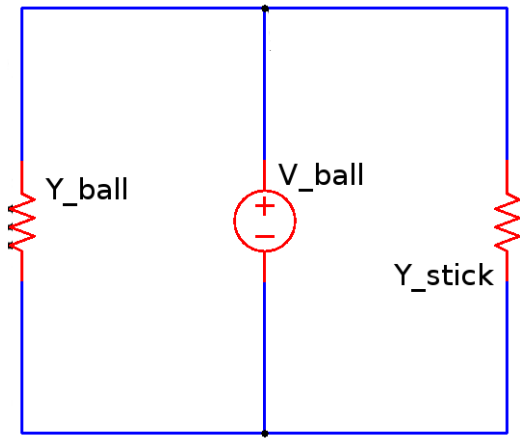

Figure 2.9: ***Equivalent circuit for the ball-and-stick model receiving somatic current stimulus.*** *The sinusoidal stimulation current yields a sinusoidal soma potential, $V_{\text{ball}}(t)$, which is modeled as an alternating voltage source. The stimulation current leaves the neuron either through the soma membrane or the stick membrane, making it a parallel connection with the complex admittances $\mathbf{Y}_{\text{ball}}$ and $\mathbf{Y}_{\text{stick}}$.*

electric circuit theory, the equivalent admittance is just the sum of these. By the complex form of Ohm's law, the soma (ball) potential becomes [54]

$$\mathbf{V}_{\text{ball}} = \frac{\mathbf{I}_e}{\mathbf{Y}_{\text{ball}} + \mathbf{Y}_{\text{stick}}}.$$

The somatic return current is given by

$$\mathbf{I}_{\text{ball}} = \mathbf{Y}_{\text{ball}}\mathbf{V}_{\text{ball}},$$

or equivalently,

$$\mathbf{I}_{\text{ball}} = \frac{\mathbf{Y}_{\text{ball}}}{\mathbf{Y}_{\text{ball}} + \mathbf{Y}_{\text{stick}}}\mathbf{I}_{\text{e}}.$$

The electrode current can be split into a complex amplitude $\hat{\mathbf{I}}_0$, also containing the phase, and a complex exponential, i.e.,

$$\mathbf{I}_{\text{e}} = \hat{\mathbf{I}}_0 e^{j\omega t}. \tag{2.16}$$

The current density out of the ball membrane hence becomes

$$\mathbf{J}_{\text{ball}} = \frac{1}{S}\frac{\mathbf{Y}_{\text{ball}}}{\mathbf{Y}_{\text{ball}} + \mathbf{Y}_{\text{stick}}}\hat{\mathbf{I}}_0 e^{j\omega t}, \tag{2.17}$$

where S is its surface area. In the implementations, S will be found numerically from the mesh being used.

Equation (2.16) lets us write the soma potential as

$$\mathbf{V}_{\text{ball}} = \frac{\hat{\mathbf{I}}_0 e^{j\omega t}}{\mathbf{Y}_{\text{ball}} + \mathbf{Y}_{\text{stick}}}.$$

By introducing the amplitude

$$\hat{\mathbf{V}}_0 = \frac{\hat{\mathbf{I}}_0}{\mathbf{Y}_{\text{ball}} + \mathbf{Y}_{\text{stick}}},$$

we can write

$$\mathbf{V}_{\text{ball}} = \hat{\mathbf{V}}_0 e^{j\omega t}. \tag{2.18}$$

We assume that the potential at the point of connection between the ball and the stick is $\mathbf{V}_{\text{ball}}$. Figure 2.9 shows the equivalent circuit. Note that the membrane currents of the stick have not been explicitly modeled. $\mathbf{Y}_{\text{stick}}$ represents the admittance to longitudinal current inside the stick at the point where it meets the ball.

Pettersen and Einevoll [43] assumed that the distal end of the stick was insulating. By using the expression (2.6) for longitudinal current density in a cable,

$$j_{\text{i}}(x,t) = -\frac{1}{r_{\text{i}}}\frac{\partial V(x,t)}{\partial x},$$

insulation is imposed by the boundary condition

$$j_{\text{i}}(l,t) = -\frac{1}{r_{\text{i}}}\frac{\partial V(l,t)}{\partial x} = 0,$$

where l is the stick length, denoted L in figure 2.8. From the equivalent circuit model, the soma end of the stick is at the potential given by equation (2.18). This gives the Dirichlet boundary condition

$$\mathbf{V}(0,t) = \hat{\mathbf{V}}_0 e^{j\omega t}. \tag{2.19}$$

Since the BC consists of a single frequency, the potential will also be sinusoidal due to the linearity of the cable equation. A solution of the form

$$\mathbf{V} = \hat{\mathbf{V}} e^{j\omega t}$$

can be assumed, and inserted into the cable equation. This was done by Pettersen and Einevoll [43], who found the expression

$$\mathbf{i}_{\text{stick}}(x,t) = \mathbf{H}(x)\hat{\mathbf{V}}_0 e^{j\omega t}$$

for the current per unit length along the stick, where $\mathbf{H}$ is the stick transfer function. For a stick of diameter d, the current per unit area crossing the membrane becomes

$$\mathbf{J}_{\text{stick}}(x,t) = \frac{1}{\pi d}\mathbf{H}(x)\hat{\mathbf{V}}_0 e^{j\omega t}. \tag{2.20}$$

The real part of this expression gives the physical transmembrane current.

The soma admittance is [34]

$$\mathbf{Y}_{\text{ball}} = \frac{4\pi d^2 \mathbf{s}^2}{R_{\text{m}}}.$$

Stick admittance,

$$\mathbf{Y}_{\text{stick}} = \frac{\pi d^{3/2}\mathbf{s}}{2\sqrt{R_{\text{i}}R_{\text{m}}}}\left[\frac{1}{1+\exp(2\mathbf{s}l/\lambda)} - \frac{1}{1+\exp(-2\mathbf{s}l/\lambda)}\right],$$

and stick transfer function,

$$\mathbf{H}(x) = \frac{\pi \mathbf{s}^2 d}{R_{\text{m}}}\left[\frac{\exp(\mathbf{s}x/\lambda)}{1+\exp(2\mathbf{s}x/\lambda)} + \frac{\exp(-\mathbf{s}x/\lambda)}{1+\exp(-2\mathbf{s}x/\lambda)}\right],$$

were derived by [43]. The frequency-dependence arises through the term

$$\mathbf{s} = \sqrt{1+j\omega\tau_{\text{m}}}.$$

Letting $\partial\Omega_7$ and $\partial\Omega_8$ denote the ball and the stick membrane, respectively, the boundary conditions on their surfaces become

$$\sigma\frac{\partial\phi}{\partial n} = J_{\text{ball}}(t) \quad \text{on } \partial\Omega_7$$

and

$$\sigma\frac{\partial\phi}{\partial n} = J_{\text{stick}}(x,t) \quad \text{on } \partial\Omega_8,$$

where x is the distance from the point of connection between the two.

2.3 Microelectrode Arrays

2.3.1 Neural Activity Measurements

Recordings of neural electrical activity may be performed in several ways. A first distinction is between extracellular and intracellular measurements. In the latter, the cell membrane is penetrated with an electrode. From the measured potentials it is straightforward to read out the firing of action potentials in the cell being considered and also to study subthreshold fluctuations. Intracellular recordings, however, are hard to perform in vivo. Also, they usually have to be done in the soma due to the difficulty of inserting an electrode into the very thin axons and dendrites. By placing an electrode just outside the soma, spikes can be counted without breaking the membrane and in a way that is technically easier [11, 44].

Extracellular recordings are performed at a variety of spatial scales, from single neuron measurements to the electroencephalogram (EEG) which measures electric potentials on top of the skull arising from activity in millions or billions of neurons. The EEG hence pictures the overall electrical activity in the brain, though on a coarse scale [41]. The recorded extracellular potentials can be filtered, giving the *multi-unit activity* (MUA) which consists of frequency components above about 500 Hz and the *local field potential* (LFP) containing frequencies less than about 500 Hz [44]. The MUA gives a picture of the action potential firing in the population being recorded.

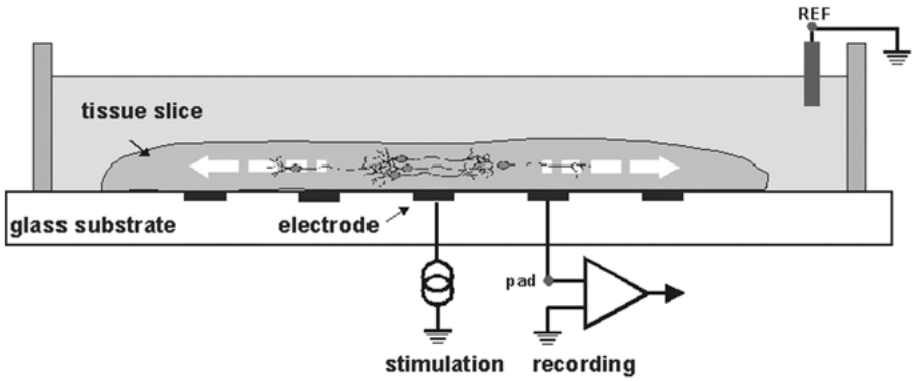

Figure 2.10: ***Experimental set-up of MEA measurements.*** *A tissue slice immersed in saline is placed on top of a glass substrate containing a dense array of electrodes. In the experiments considered in this thesis, the stimulation electrodes on the array have not been used. The reference electrode is embedded in the surrounding saline solution. Taken from Fejtl et al. [19].*

Measured extracellular potentials may have contributions from many neurons in the vicinity of the electrode. In order to follow the firing activity of a single cell, the electrode needs to be placed very close to its soma. For devices with multiple electrodes, the temporal shape of a spike from a particular cell varies between recording sites. This can be used to identify neurons contributing to the measured potential [13, 17].

Microelectrode chips have been able to record extracellular potentials from brain slices at hundreds of electrodes. Frey et al. [20] reported simultaneous recordings from 126 sites using a 7.5×6.1 mm chip. This large number provides a means of detecting the network acitivity of the neurons in the sample [10, 20]. The MUA part of the signal can be used to study firing rates of the different neurons. The LFP, on the other hand, is harder to interpret directly [16]. It is not as strongly attenuated with distance as MUA, so the LFP at a recording site contains signals from neurons in a relatively large volume [44]. Synaptic activity in the dendrites seems to be the main source of the LFP [17].

2.3.2 Experimental Set-up

MEA measurements are performed using either cell cultures are acute slices. *Cultured neurons* are typically taken out from fetuses. Under the right conditions their development continues and the growth of axons and dendrites may be followed. An example of a microelectrode array specially produced for this purpose is the *neurocage* [18]. With this, a single neuron is trapped on top of an electrode, so its development can be studied both optically and via electric potentials [14, 45, 46].

Recent MEA recordings from *acute slices* of rat cerebellum have revealed a very clear sink/source pattern [20]. In recordings from several slices, a challenge

is to position the different samples similarly on the MEA, so that a comparison can be made. Bakker et al. [10] reported doing this succesfully, and the spread of action potentials in the slice, with current sinks and sources, was measured.

An introduction to microelectrode arrays written by developers at Multi Channel Systems [3] is given by Fejtl et al. [19]. Figure 2.10 sketches the set-up modeled in this thesis, i.e., a tissue slice embedded in saline (ACSF) etched on top of an array of electrodes. The slice thickness is typically somewhere between 300 and 400 μm [10, 20].

2.3.3 Stimulation Electrodes

The stimulation considered in this thesis is performed by injecting an electrode into the soma of a particular neuron in the tissue. The resulting extracellular potential arising from the induced transmembrane currents are measured on the MEA [10, 21]. One or two of the electrodes on the array may also be used for stimulation, giving a current into the bulk tissue. From the measured potentials it may be hard to distinguish between the response of the neurons and the stimulation itself. However, this type of stimulation can be used to measure tissue impedance [22].

2.3.4 Recording Electrodes

By proper adjustment of the measurement apparatus, the recording electrodes on the MEA can be modeled as electrical insulators [21]. This means that the electrode impedance is high enough to let practically no current pass through it. When this simplification is used, there is no need to distinguish the electrodes from the rest of the array in the FEM model [51]. Instead, the potentials at the electrode positions can be extracted from the data after simulation.

In this work, ideal electrodes have been assumed, and the whole microelectrode array has been model as an electrical insulator. The Neumann boundary condition at its surface thus is

$$\sigma \frac{\partial \phi}{\partial n} = 0 \quad \text{for } x \in \partial\Omega_6,$$

where $\partial\Omega_6$ is the bottom boundary, cf. section 4.3.

Chapter 3

Finite Element Method

The theory in this chapter is largely based on the book by Langtangen [30], the FEniCS tutorial by Langtangen [32], and the book by Strang [55].

Finite element discretization has been used in this work. It provides flexible simulation of partial differential equations (PDEs). Compared to the finite difference method, the FEM is particularly advantageous when it comes to handling domains of complicated geometry [30, 55], exemplified here by various boundary conditions (cell membranes, air, reference electrode, MEA), media with different electrical properties (tissue, saline), as well as the geometry of the neuron model employed.

The following chapter presents, somewhat heuristically, the mathematical basis of the FEM. The integral notation differs from that of chapter 2. In particular,

$$\int_\Omega f \mathrm{d}x$$

denotes the integral of a function f over the whole domain Ω, independent of Ω's dimension. E.g., if Ω is the unit cube, then

$$\int_\Omega f \mathrm{d}x = \int_{x=0}^{x=1} \int_{y=0}^{y=1} \int_{z=0}^{z=1} f(x,y,z) \mathrm{d}z \mathrm{d}y \mathrm{d}x.$$

Also,

$$\int_{\partial\Omega} f \mathrm{d}s$$

denotes the surface integral of f over the boundary $\partial\Omega$ of Ω. In the case of the unit cube, $\mathrm{d}s$ hence is an infinitesimal area element.

3.1 Weak Formulation of Boundary Value Problems

Consider the *elliptic* [47] boundary value problem

$$-\nabla \cdot (\sigma \nabla u) = f \qquad x \in \Omega \tag{3.1}$$

$$u = g \qquad x \in \partial\Omega_{\mathrm{D}} \tag{3.2}$$

$$\sigma \frac{\partial u}{\partial n} = h \qquad x \in \partial\Omega_{\mathrm{N}}, \tag{3.3}$$

in which Ω is the domain, u is the primary unknown, and $\partial\Omega_{\mathrm{D}}$ and $\partial\Omega_{\mathrm{N}}$ are the boundary parts with Dirichlet and Neumann boundary conditions, respectively. f, g, and h are scalar functions, possibly time-dependent, and $x \in \mathbb{R}^d$, where d is the dimension of the space considered (typically, $d = 1$, 2, or 3). The coefficient σ can in general be a $d \times d$ matrix (tensor of order 2). With the appropriate choice of functions and boundaries, this boundary value problem could describe volume conduction around a neuron, with transmembrane current density specified by $h(x, t)$.

The solution to (3.1)-(3.3) must have finite second derivatives because the source function f on the right-hand side of (3.1) has to be finite; an infinitely large source makes no sense physically. This means that both u and $\sigma\nabla u$ must be continuous in Ω, since discontinuous functions have infinite derivatives at their points of discontinuity. When σ contains step discontinuities, e.g., between the saline and tissue, $\sigma\nabla u$ is still a continuous function representing the volume current across the interface [23]. To simplify the discussion, σ is now assumed to be continuous, making ∇u continuous whenever $\sigma\nabla u$ is. For discontinuous σ, the same results would be obtained by replacing ∇u with $\sigma\nabla u$. Mathematically, u is an element of the Hilbert space $H^2(\Omega)$, containing all functions whose second derivatives are square integrable on Ω. That is,

$$\int_\Omega \|\nabla^2 u\|^2 \mathrm{d}\Omega < \infty \quad \forall u \in H^2(\Omega).$$

Now, both sides of equation (3.1) are multiplied with a scalar valued *test function*, v, and the product is integrated over Ω, giving

$$-\int_\Omega \nabla \cdot (\sigma \nabla u) v \mathrm{d}x = \int_\Omega f v \mathrm{d}x. \tag{3.4}$$

The left-hand side is integrated by parts,

$$-\int_\Omega \nabla \cdot (\sigma \nabla u) v \mathrm{d}x = \int_\Omega \sigma \nabla u \cdot \nabla v \mathrm{d}x - \int_{\partial\Omega} v\sigma \frac{\partial u}{\partial n} \mathrm{d}s. \tag{3.5}$$

Insertion of (3.5) into (3.4) yields

$$\int_\Omega \sigma \nabla u \cdot \nabla v \mathrm{d}x = \int_\Omega f v \mathrm{d}x + \int_{\partial\Omega} v\sigma \frac{\partial u}{\partial n} \mathrm{d}s.$$

We require

$$v = 0 \quad \text{on } \partial\Omega_{\mathrm{D}},$$

i.e., equation (3.4) is satisfied on $\partial\Omega_{\mathrm{D}}$ for any u. Essential (Dirichlet) boundary conditions can thus be imposed directly on the solution u. This implies

$$\int_{\partial\Omega} v\sigma\frac{\partial u}{\partial n}\mathrm{d}s = \int_{\partial\Omega_{\mathrm{N}}} v\sigma\frac{\partial u}{\partial n}\mathrm{d}s,$$

which gives

$$\int_{\Omega} \sigma\nabla u\cdot\nabla v\mathrm{d}x = \int_{\Omega} fv\mathrm{d}x + \int_{\partial\Omega_{\mathrm{N}}} v\sigma\frac{\partial u}{\partial n}\mathrm{d}s.$$

Inserting the Neumann boundary condition (3.3) yields

$$\int_{\Omega} \sigma\nabla u\cdot\nabla v\mathrm{d}x = \int_{\Omega} fv\mathrm{d}x + \int_{\partial\Omega_{\mathrm{N}}} vh\mathrm{d}s. \tag{3.6}$$

Integration by parts removed the second derivatives of the original boundary value problem. To ensure finiteness, u and v must be continuous, but their first derivatives may have step discontinuities. Hence, the test function and the solution must be elements of $H^1(\Omega)$. Since $H^2(\Omega)\subset H^1(\Omega)$, more functions satisfy the integrated form than the original formulation involving second derivatives. Because of this weaker condition on u, equation (3.6) is called a *weak formulation* of the boundary value problem.

We now introduce the notation

$$a(u,v) = \int_{\Omega} \sigma\nabla u\cdot\nabla v\mathrm{d}x \tag{3.7}$$

and

$$L(v) = \int_{\Omega} fv\mathrm{d}x + \int_{\partial\Omega_{\mathrm{N}}} vh\mathrm{d}s, \tag{3.8}$$

where $a(u,v)$ and $L(v)$ are called the *bilinear form* and the *linear form*, respectively. In general, the bilinear form is the term involving both u and v, whereas the linear form involves v only.

The weak formulation of the boundary value problem (3.1)-(3.3) is: Find $u\in V$ such that

$$a(u,v) = L(v) \quad \forall v\in\hat{V}, \tag{3.9}$$

where

$$V = \quad \{v\in H^1(\Omega) : v = g \text{ on } \partial\Omega_{\mathrm{D}}\} \tag{3.10}$$

$$\hat{V} = \quad \{v\in H^1(\Omega) : v = 0 \text{ on } \partial\Omega_{\mathrm{D}}\}. \tag{3.11}$$

This is referred to as a *variational problem*, and u is called the *trial function*. Dirichlet boundary conditions enter in the definition of the *trial space*, V, whereas Neumann boundary conditions are expressed in the linear form, $L(v)$. Under certain conditions, which are always met here, the Lax-Milgram theorem ensures existence and uniqueness of the solution u.

3.2 Discretization

The trial space (equation (3.10)) and the test space (equation (3.11)) from the weak formulation in the last section were infinite dimensional. In order to calculate an approximation, u_h, to the true solution, u, the variational problem must be required to hold for only a finite number of test functions. It is standard notation to let a subscript h denote a discretization.

Let $\phi_1, \ldots, \phi_n$ be linearly independent test functions. They form the basis of an n-dimensional test space,

$$\hat{V}_h = \{v \in \text{span}(\phi_1, \ldots, \phi_n) : v = 0 \text{ on } \partial\Omega_\text{D}\}.$$

With *Galerkin's method*, u_h is sought as a linear combination of the test functions,

$$u \approx u_h = \sum_{i=1}^{n} U_i \phi_i, \tag{3.12}$$

where $U_1, \ldots, U_n$ are scalar coefficients. Hence, the numerical approximation is a projection of the true solution onto $\hat{V}_h$. Dirichlet boundary conditions are kept out of the discussion and will be handled later.

The bilinear and linear form possess the property of linearity, i.e.,

$$a(u_1 + u_2, v_1 + v_2) = a(u_1, v_1) + a(u_2, v_1) + a(u_1, v_2) + a(u_2, v_2)$$

and

$$L(v_1 + v_2) = L(v_1) + L(v_2).$$

This can be seen from their definitions, equations (3.7) and (3.8), since the integral of a sum equals the sum of the integrals over each term. Exploiting linearity we can write the bilinear form of the discretized problem as

$$a(u_h, \phi_i) = a(\sum_{j=1}^{n} U_j \phi_j, \phi_i) = \sum_{j=1}^{n} a(\phi_j, \phi_i) U_j, \quad i = 1, \ldots, n.$$

Symmetry of the bilinear form,

$$a(u, v) = a(v, u), \tag{3.13}$$

is always satisfied in the cases considered here, giving

$$a(u_h, \phi_i) = \sum_{j=1}^{n} a(\phi_i, \phi_j) U_j, \quad i = 1, \ldots, n.$$

The linear form is just

$$L(\phi_i), \quad i = 1, \ldots, n.$$

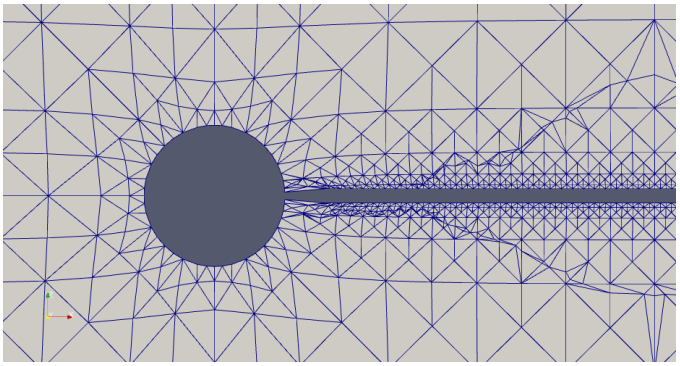

Figure 3.1: ***Mesh example.*** *The figure shows a cross section of the three-dimensional mesh representing a ball-and-stick neuron. Each triangle is part of a tetrahedron, making up an element* Ω_i.

The discretized version of the variational problem now is: Find $U_1, \ldots, U_n$ such that

$$\sum_{j=1}^{n} a(\phi_i, \phi_j) U_j = L(\phi_i), \quad i = 1, \ldots, n.$$

where $\phi_i \in \hat{V}_h$. This amounts to requiring that the numerical solution exactly satisfies the continuous weak problem (equation (3.9)) for n independent test functions.

If A is a matrix with entries

$$A_{ij} = a(\phi_i, \phi_j),$$

the coefficient vector is defined as

$$U = (U_1, \ldots, U_n)^{\mathrm{T}},$$

and the right-hand side is

$$b = (L(\phi_1), \ldots, L(\phi_n))^{\mathrm{T}},$$

we get

$$AU = b. \tag{3.14}$$

The solution vector, U, gives the weights in the series expansion for the numerical approximation, u_h. The Dirichlet boundary conditions are directly imposed on the linear system, to be demonstrated in the next section.

3.3 Piecewise Polynomials

In the discretization procedure the domain is divided into nonoverlapping *elements*. Figure 3.1 illustrates this by a close-up of a mesh used to represent the space surrounding a ball-and-stick neuron. A cross section of a three-dimensional mesh is

shown. Each triangle is part of a tetrahedron making up an element. In three dimensions the altogether m elements consist of points, lines, and planes, in FEniCS terminology called *vertices*, *edges*, and *faces*, respectively. Two-dimensional elements are made up of vertices and edges only, and in one dimension they consist of vertices [8, 55].

One test function is assigned to each vertex, and we require test function number i to equal unity on vertex i and zero on all other vertices. This is formulated mathematically as

$$\phi_i(x_j) = \delta_{ij}, \tag{3.15}$$

where δ_{ij} is the Kronecker delta and x_j the position of vertex j. The approximate solution at vertex i thus is

$$u_h(x_i) = \sum_{j=1}^{n} U_j \phi_j(x_i) = \sum_{j=1}^{n} U_j \delta_{ij} = U_i. \tag{3.16}$$

Hence, the value at each vertex is just the value of the corresponding vector element in U, obtained from solution of the linear system (3.14). The approximated value between the vertices will be a series expansion over the test functions whose value at the point considered is nonzero.

3.3.1 One Dimension

The basic ideas are easier to explain in one dimension and readily extend to a d-dimensional domain. Consider the following boundary value problem on $\Omega = [0, 1]$,

$$\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = 1 \tag{3.17}$$

$$u(0) = 1 \tag{3.18}$$

$$u(1) = 0. \tag{3.19}$$

Here, $\partial\Omega_{\mathrm{D}} = \{0, 1\}$ and $\partial\Omega_{\mathrm{N}} = \emptyset$. The analytical solution is

$$u(x) = \frac{x^2}{2} - \frac{3x}{2} + 1.$$

Multiplying both sides with a test function and integrating over Ω, we get

$$\int_0^1 u'' v \mathrm{d}x = \int_0^1 v \mathrm{d}x.$$

Next, we require $v = 0$ on $\partial\Omega_{\mathrm{D}}$, i.e., $v(0) = v(1) = 0$. Integration by parts of the left-hand side thus gives

$$\int_0^1 u'' v \mathrm{d}x = u'v\Big|_0^1 - \int_0^1 u'v' \mathrm{d}x = -\int_0^1 u'v' \mathrm{d}x,$$

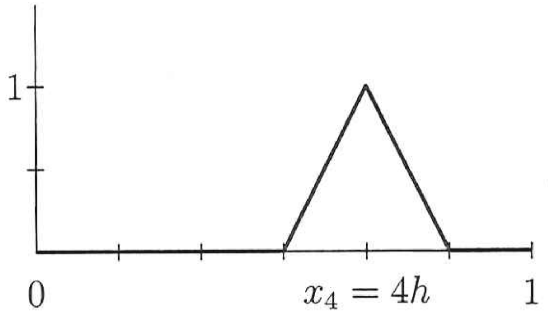

Figure 3.2: ***Piecewise linear test function on a discretized unit interval.*** *The interval* $[0, 1]$ *is partitioned into 6 elements and 7 edges. Also shown is the piecewise linear test function* ϕ_4*. Taken from Strang [55].*

so the weak formulation of the boundary value problem becomes

$$-\int_0^1 u'v'\mathrm{d}x = \int_0^1 v\mathrm{d}x. \tag{3.20}$$

Now, the unit interval is uniformly partitioned into $n+2$ nodes with coordinates $x_i = ih$, where $i = 0, \ldots, n+1$ and

$$h = \frac{1}{n+1}.$$

Each subinterval, $[x_i, x_{i+1}]$, makes up an element. In figure 3.2 there are 7 nodes, so the number of elements is $m = 6$.

This leads to the definition of the test functions. *Piecewise linear* functions are most common, and were used for all computations in this thesis. In one dimension they are defined by

$$\phi_i = \begin{cases} (x - x_{i-1})/h & \text{if } x \in [x_{i-1}, x_i] \\ (x_{i+1} - x)/h & \text{if } x \in [x_i, x_{i+1}] \\ 0 & \text{else} \end{cases} \tag{3.21}$$

Figure 3.2 shows the graph of ϕ_4. Note that for nonuniform partitioning the h in the denominator will become a variable. The functions considered are $\phi_1, \ldots, \phi_n$, since x_0 and x_{n+1} are subject to essential boundary conditions. Their derivatives are

$$\phi_i' = \begin{cases} 1/h & \text{if } x \in [x_{i-1}, x_i] \\ -1/h & \text{if } x \in [x_i, x_{i+1}] \\ 0 & \text{else} \end{cases} \tag{3.22}$$

This implies that

$$\int_\Omega \|\nabla\phi_i\|^2\mathrm{d}x = \int_0^1 \left(\phi'\right)^2 \mathrm{d}x < \infty, \quad i = 1, \ldots, n,$$

so

$$\phi_i \in H^1(\Omega) \quad \text{for } i = 1, \ldots, n,$$

and the piecewise linear test functions obey the requirements stated in section 3.1.

Equation (3.20) gives the continuous weak form. With Galerkin's method, the discretized version is

$$-\sum_{j=1}^{n} \left(\int_0^1 \phi_i' \phi_j' \mathrm{d}x \right) U_j = \int_0^1 \phi_i \mathrm{d}x, \quad i = 1, \ldots, n. \tag{3.23}$$

The integrals in the sum on the left-hand side follow from equation (3.22):

$$\begin{aligned}
\int_0^1 \phi_i' \phi_j' \mathrm{d}x = \int_0^1 \phi_i' 0 \mathrm{d}x = &\quad 0 && \text{if } j \notin \{i-1, i, i+1\} \\
\int_0^1 \phi_i' \phi_j' \mathrm{d}x = \int_{x_{i-1}}^{x_i} \left(\frac{1}{h}\right)\left(\frac{-1}{h}\right) \mathrm{d}x = &\quad \tfrac{-1}{h} && \text{if } j = i-1 \\
\int_0^1 \phi_i' \phi_j' \mathrm{d}x = \int_{x_{i-1}}^{x_i} \frac{1}{h}\frac{1}{h} \mathrm{d}x + \int_{x_i}^{x_{i+1}} \frac{-1}{h}\frac{-1}{h} \mathrm{d}x = &\quad \tfrac{2}{h} && \text{if } j = i \\
\int_0^1 \phi_i' \phi_j' \mathrm{d}x = \int_{x_i}^{x_{i+1}} \left(\frac{-1}{h}\right)\left(\frac{1}{h}\right) \mathrm{d}x = &\quad \tfrac{-1}{h} && \text{if } j = i+1
\end{aligned}$$

Here,

$$h = x_i - x_{i-1} = x_{i+1} - x_i$$

has been used. Defining the coefficients of the *stiffness matrix*, A, as

$$A_{i,j} = \int_0^1 \phi_i' \phi_j' \mathrm{d}x,$$

we see that $A_{i,i-1} = -1/h$, $A_{i,i} = 2/h$, and $A_{i,i+1} = -1/h$, while all other entries are zero.

The right-hand side can also be evaluated analytically in this case:

$$\begin{aligned}
\int_0^1 \phi_i \mathrm{d}x =& \int_{x_{i-1}}^{x_i} \frac{(x - x_{i-1})}{h} \mathrm{d}x + \int_{x_i}^{x_{i+1}} \frac{(x_{i+1} - x)}{h} \mathrm{d}x \\
=& \frac{1}{h} \int_{x_{i-1}}^{x_i} x \mathrm{d}x - \frac{x_{i-1}}{h} \int_{x_{i-1}}^{x_i} \mathrm{d}x + \frac{x_{i+1}}{h} \int_{x_i}^{x_{i+1}} \mathrm{d}x - \frac{1}{h} \int_{x_i}^{x_{i+1}} x \mathrm{d}x \\
=& \frac{1}{2h} x^2 \Big|_{x_{i-1}}^{x_i} - x_{i-1} + (x_{i-1} + 2h) - \frac{1}{2h} x^2 \Big|_{x_i}^{x_{i+1}} \\
=& \frac{1}{2h} \left(x_i^2 - x_{i-1}^2 - x_{i+1}^2 + x_i^2 \right) + 2h \\
=& \frac{1}{2h} \left(x_i^2 - (x_i - h)^2 - (x_i + h)^2 + x_i^2 \right) + 2h \\
=& \frac{-2h^2}{2h} + 2h \\
=& h
\end{aligned}$$

An alternative to the calculations above is to observe that the value of the integral equals the area under the curve of the function, cf. figure 3.2. In general, though, it has to be computed using some numerical integration scheme.

The linear system (3.23) can now be written as

$$\frac{-1}{h}\left(-U_{i-1}+2U_i-U_{i+1}\right)=h, \quad i=1,\ldots,n$$

This is exactly the same tridiagonal system as would be obtained by a second order accurate finite difference discretization of the problem. Dividing both sides by h makes this even more clear. The advantage of the finite element method is that it easily extends to problems of higher dimensions and irregular domains. With the division shown in figure 3.2, i.e., 7 equally spaced edges, the linear system is

$$\frac{-1}{h}\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} U_0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ U_6 \end{pmatrix}=\begin{pmatrix} -1/h \\ h \\ h \\ h \\ h \\ h \\ 0 \end{pmatrix} \tag{3.24}$$

The first and the last equation, $U_0 = 1$ and $U_6 = 0$, directly impose the essential boundary conditions, $u(0) = 1$ and $u(1) = 0$. In practice a few more manipulations would be done in order to create a symmetric coefficient matrix. This enables fast algorithms for solving the linear system.

Given the vector U in equation (3.24), the finite element solution of the boundary value problem (3.17)-(3.19) using the test functions of equation (3.21) is

$$u_h(x)=\sum_{j=0}^{6}\phi_j(x)U_j.$$

Figure 3.3 shows a plot of u_h. Note that because piecewise linear test functions are used, u_h consists of straight lines between the nodes, as opposed to the exact solution which is a second order polynomial [58].

3.3.2 Multiple Dimensions

The piecewise linear test functions described in the last section generalize straightforwardly to multiple dimensions. Figure 3.4 shows a sketch of a tetrahedron, which is the type of element used in this work. The typically irregular shape of the tetrahedra depends on the local geometry and fineness of the mesh. A transformation needs to be done between the physical coordinates, (x, y, z), and a regular reference tetrahedron with local coordinates (α, β, γ). FEniCS performs this automatically.

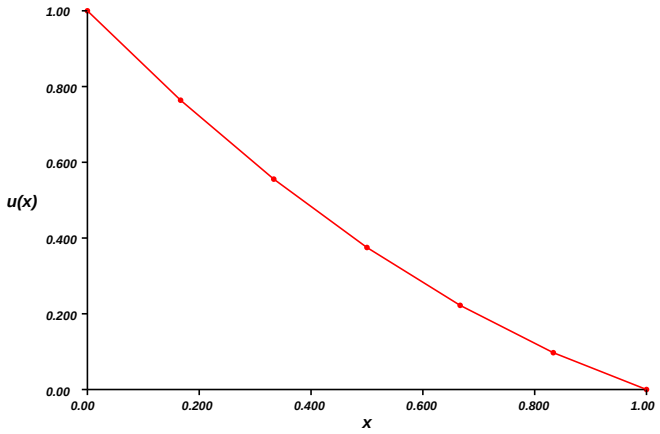

Figure 3.3: ***FEM solution in one dimension.*** *The curve shows the discretized solution of the boundary value problem (3.17)-(3.19) with the partitioning of figure 3.2. FEniCS was used to perform the computations and generate the plot.*

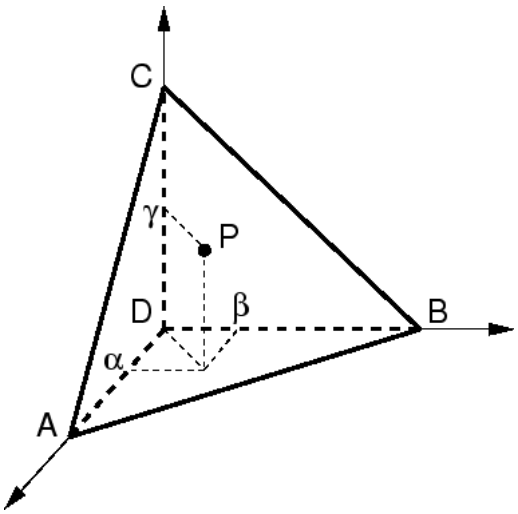

Figure 3.4: ***Tetrahedron used in a three-dimensional finite element mesh.*** *The figure shows a regular reference tetrahedron with local coordinates α, β, and γ. Taken from Scholz et al. [50].*

In figure 3.4, this means that the local coordinates of the points are A: $(1, 0, 0)$, B: $(0, 1, 0)$, C: $(0, 0, 1)$, and D: $(0, 0, 0)$. Still using piecewise linear polynomials, the test functions for a particular cell defined in local coordinates become

$$\begin{aligned}
\phi_A(\alpha, \beta, \gamma) &= \alpha \\
\phi_B(\alpha, \beta, \gamma) &= \beta \\
\phi_C(\alpha, \beta, \gamma) &= \gamma \\
\phi_D(\alpha, \beta, \gamma) &= 1 - \alpha - \beta - \gamma
\end{aligned}$$

Just as in the one-dimensional case, the discretized solution is a series expansion over these functions on the mesh. Local coordinates could equally well have been used in the simple one-dimensional case of the last section, but on a uniformly partitioned interval the integrals can easily be evaluated directly in the global coordinate system.

Chapter 4

FEniCS Implementations

FEniCS is a collection of software tools specialized for solving differential equations [2]. For the partial differential equations considered here, the important component is DOLFIN, which can be accessed through C++ or Python code [36]. In this work, Python [6] has been the language of choice. The following sections will emphasize important steps necessary for the simulation of MEA measurements, and present the computer implementations of the simulations performed. Complete source code can be found on the attached CD, cf. appendix B.

This chapter is partly based on the texts by Langtangen [30–32] and the demos and documentation available on the FEniCS website [2].

4.1 Weak Form

The DOLFIN package contains the components necessary for simulating PDEs with FEniCS. The statement

```
from dolfin import *
```

makes them accessible from a Python script.

The next step is to specify the domain in which the simulations will be performed. DOLFIN comes with meshes of some basic shapes in 1D, 2D, and 3D. For example, a three-dimensional unit cube with 10+1 vertices in each direction is generated by the command

```
mesh = UnitCube(10,10,10)
```

The `mesh` variable now holds a reference to the DOLFIN `UnitCube` object. Note that discretization of the domain Ω is done automatically, with parameters specified as attributes upon declaration. This cube can later be transformed to a more complicated region.

The mesh generated is a *simplex mesh*. A simplex is a line segment in 1D, a triangle in 2D, and a tetrahedron in 3D [47, 59]. In other words, the elements making up the mesh have 2, 3, or 4, vertices in 1D, 2D, or 3D, respectively.

Trial functions, test functions, and test space will be named `u`, `v`, and `V`, respectively. The necessary declarations are done with the lines:

```
V = FunctionSpace(mesh, 'CG', 1)
u = TrialFunction(V)
v = TestFunction(V)
```

The first attribute to `FunctionSpace` specifies the mesh on which the functions are defined. `CG` stands for *Continuous Galerkin*, which are simply piecewise polynomials, and the last attribute tells that they are first-order, i.e., linear. The next two lines define the trial function and test functions as instances of the respective DOLFIN classes. Since Galerkin's method is used, `u` and `v` are elements of the same function space.

Other functions in the problem need to be specified as instances of the `Expression` class. For example, an electrode current $I = I_0 \cos(\omega t)$ is defined by[1]

```
I_0 = 250 # value of current amplitude
omega = 1 # value of angular frequency

# Declare the current I
# I_0 and omega are handed over to I in a dictionary
I = Expression('xI_0*cos(xomega*t)', {'xI_0' : I_0, '
   xomega' : omega})
# Update the time
I.t = t
```

The first two lines declare the amplitude and angular frequency. They can be handed over to the `Expression` instance `I` in a *dictionary* on declaration. When parameters need to be updated, e.g., in a time loop, the assignment shown in the last line is more convenient.

Weak forms are implemented in FEniCS using the *Unified Form Language* (UFL), which provides a syntax close to mathematical formulation [8]. Consider the general bilinear and linear form arising from the Poisson problem of section 3.1,

$$a(u, v) = \int_\Omega \sigma \nabla u \cdot \nabla v \mathrm{d}x$$

and

$$L(v) = \int_\Omega f v \mathrm{d}x + \int_{\partial\Omega_\mathrm{N}} v h \mathrm{d}s.$$

Translated into UFL, they become

```
a = sigma*inner(grad(u),grad(v))*dx
L = f*v*dx + sigma*v*h*ds(1)
```

Multiplication by `dx` implies an integral over the whole domain. Multiplication by `ds` means that an integral over a region of one topological dimension less

[1] The somewhat peculiar variable names starting with an x are used to distinguish variables inside the expression from those holding parameter values in the Python script.

than the original domain is evaluated, a surface integral in this case. However, the surface integral should only be calculated on the part of the boundary with Neumann conditions. Here it is assumed that $\partial\Omega_{\mathrm{N}}$ is marked with the value `1`, which is given as an attribute in the statement `ds(1)`. This ensures that the integral is evaluated over the right surface. An instance of the `MeshFunction` class, named `boundaries`, holds the information about the boundary markers, cf. section 4.3.

The terms in the forms need to be properly defined. E.g., if σ is a constant, it can be declared by

```
sigma = Constant(3.0) # Example value 3.0 S/m
```

or as an `Expression` instance.

The example problem in section 3.1 also was subject to a Dirichlet boundary condition. Assuming that `boundaries` has the value 2 on $\partial\Omega_{\mathrm{D}}$, an instance of `DirichletBC` is declared with the line

```
bc = DirichletBC(V, g, boundaries, 2)
```

Here, `g` is assumed to be an instance of the `Expression` or `Constant` class.

Now, the problem is solved by:

```
problem = VariationalProblem(a, L, bc,
    exterior_facet_domains=boundaries)
phi = problem.solve()
```

The first line declares a `VariationalProblem` instance. The bilinear form and the linear form are given as the first two attributes. The Dirichlet boundary condition, represented by `bc`, also needs to be specified. The last attribute states that the information about boundary markers is contained in the `boundaries` instance. The second line of the snippet solves the discrete variational problem, returning its solution to the variable `phi` as an instance of the DOLFIN `Function` class. It can be plotted using the built-in tool Viper, exported to file formats suitable for data analysis and visualization, and it can be used to compute functionals, e.g., the volume current, $-\sigma\nabla\phi$.

4.2 Mesh Generation

DOLFIN's native mesh generator, which was used in this work, is sufficient for relatively simple geometries. Here, the generation of a mesh for simulation of a ball-and-stick neuron in a MEA experiment will be explained in detail. The same principles apply for a two-monopole, but in that case the steps are somewhat simpler. For larger and more complex problems the FEniCS package contains TriTetMesh [36], a Python wrapper for the preprocessors Triangle (2D) [52] and Tetgen (3D) [53]. Meshes generated with other packages can also be converted to a format compatible with FEniCS. The article by Means et al. [37] demonstrates mesh generation reproducing the complex geometry of a single cell, based on electron tomography imaging.

The coordinate axes in FEniCS are numbered from 0, so (x, y, z) translates to `(x[0],x[1],x[2])`. The neuron is centered at $y = 0$, with its stick and the x-axis aligned. Its height above the MEA is varied. In this example, the ball radius has been set to 10 μm and the stick radius and length to 1 μm and 1 mm, respectively. Other values were also used in the simulations, cf. section 5.5 [43]. The length unit in all scripts is the millimeter. Thus, the relevant code for generating an outer box, i.e., the domain Ω, is

```
box_x_dim=4; box_y_dim=4; box_z_dim=1
mesh = Box(-box_x_dim/2.,-box_y_dim/2.,-box_z_dim/2.,
    box_x_dim/2.,box_y_dim/2.,box_z_dim/2.,nx,ny,nz)
```

nx, ny, and nz are the number of cells in each direction.

The tissue is defined in the code as a subclass of the DOLFIN `SubDomain` class

```
tissue_x_dim=2; tissue_y_dim=2; tissue_z_dim=0.3
class Tissue(SubDomain):
    def inside(self, x, on_boundary):
        return -tissue_x_dim/2.<= x[0] <= tissue_x_dim/2.
           and -tissue_y_dim/2. <= x[1] <= tissue_y_dim/2.
            and -box_z_dim/2.<= x[2] <= -box_z_dim/2. +
           tissue_z_dim
```

The `inside` method returns `True` if a point `x` is situated within the tissue boundaries and `False` otherwise.

The potential is not expected to vary vigorously in the saline, a homogeneous medium far from the current source. Hence, the piecewise polynomials need not be very close in order to approximate the true solution well, and a coarse mesh is used for computational efficiency. Inside the tissue, and close to the neuron in particular, both potentials and currents are expected to vary significantly within a small spatial range. A finer mesh is therefore needed.

Even without concern for numerical accuracy, the mesh has to be fine around the neuron. In order to make holes representing the ball and the stick, the vertices must be separated by distances significantly smaller than their radii, 10 μm and 1 μm, at the respective positions. However, if a partitioning of, say, 0.2 μm was applied throughout the whole domain, the number of vertices would be on the order of

$$\frac{(4\text{ mm})}{(0.2\times 10^{-3}\text{ mm})}\times\frac{(4\text{ mm})}{(0.2\times 10^{-3}\text{ mm})}\times\frac{(0.3\text{ mm})}{(0.2\times 10^{-3}\text{ mm})}=6\times 10^{11},$$

resulting in a

$$(6\times 10^{11})\times(6\times 10^{11})$$

linear system to be solved! Such a computational bottleneck is avoided by *local mesh refinement*, to be explained later.

The geometry of the holes also has to be provided. This is done by defining two subclasses of the DOLFIN `SubDomain` class, here named `Ball` and `Stick`.

These are used to mark the parts of the mesh containing the ball and the stick, just as a subclass for the tissue was defined earlier in this section. `Ball` and `Stick` will later be removed, creating an inner boundary through which the transmembrane currents will enter.

The ball-and-stick cell, centered at $(x, y, z) = (0, 0, z_0)$, has length

$$2r_{\text{ball}} + l,$$

where r_{ball} is the ball radius and l the stick length. Since the stick is oriented in the x-direction, the neuron will be confined within

$$\left(-\frac{2r_{\text{ball}} + l}{2} \leq x \leq \frac{2r_{\text{ball}} + l}{2}\right)$$

along the x-axis. With the soma on the negative x-axis, the center of the ball will be situated at $(x, y, z) = (-l/2, 0, z_0)$. This means that the radial distance from the center point of the ball to a point $(x, y, z) \in \Omega$ is

$$r = \sqrt{(x + l/2)^2 + y^2 + (z - z_0)^2}.$$

The point is inside the ball if $r < r_{\text{ball}}$, in case of which it should be removed from the mesh in order to create a hole.

The code defining the subclass representing the ball becomes[2]

```
class Ball(SubDomain):
   def inside(self, x, on_boundary):
       # Distance from center of ball to x
       r = sqrt((x[0]-ball_x_coor)**2 + (x[1]-ball_y_coor)
           **2 + (x[2]-ball_z_coor)**2)
       return r<1.5*radius # slightly larger than ball
   def snap(self, x):
       r = sqrt((x[0]-ball_x_coor)**2 + (x[1]-ball_y_coor)
           **2 + (x[2]-ball_z_coor)**2)
       if r < 1.5*radius:
           x[0] = ball_x_coor + (radius/r)*(x[0]-
              ball_x_coor)
           x[1] = ball_y_coor + (radius/r)*(x[1]-
              ball_y_coor)
           x[2] = ball_z_coor + (radius/r)*(x[2]-
              ball_z_coor)
```

The `inside` method returns `True` if a point in the mesh is within $3r_{\text{ball}}/2$ from the center of the ball, to make sure that the boundary points are all included. Instead using the statement

```
return r <= radius
```

[2]Note that in the scripts, x is a point, whereas x in mathematical formulation denotes position along the first coordinate axis.

would make the code very sensitive to roundoff error, possibly leading to points lying inside the ball, close to the boundary, not being properly marked as parts of it. As will be explained later, the `snap` method sets the radius of the ball back to r_{ball} again.

The stick is confined within

$$\left(-\frac{l}{2} + r_{\text{ball}} \leq x \leq \frac{l}{2} + r_{\text{ball}}\right)$$

along the x-axis, and its symmetry line lies at $y = 0$ and $z = z_0$. The radial distance from its axis to an arbitrary point is

$$r = \sqrt{y^2 + (z - z_0)^2}.$$

Accordingly, the subclass `Stick` is defined with the code

```
class Stick(SubDomain):
    def inside(self, x, on_boundary):
        # Radial distance from stick to x
         r = sqrt((x[1]-stick_y)**2 + (x[2]-stick_z)**2)
        # Return True for slightly larger region
         return r<1.5*stick_radius and stick_x_left-.5*
            radius <= x[0] <= stick_x_right+.5*radius
    def snap(self, x):
         r = sqrt((x[1]-stick_y)**2 + (x[2]-stick_z)**2)
         if r<1.5*stick_radius and stick_x_left <= x[0] <=
            stick_x_right+.5*radius:
             x[1] = stick_y+(stick_radius/r)*(x[1]-stick_y)
             x[2] = stick_z+(stick_radius/r)*(x[2]-stick_z)
```

As for the ball, slightly larger radius and length are actually used and subsequently corrected by the `snap` method.

Having defined the appropriate subclasses, the next step is mesh refinement around the ball and the stick. The method used here was to start out with a rather coarse mesh for the whole domain, with a uniform resolution of 0.1 mm. An outer loop iterating over the predefined number of refinements contained the subclass `Refinement`, which marked the cells to be refined. `Refinement` was updated in each iteration, marking a successively smaller volume for refinement. The most important parts of the code are shown here.

```
for i in range(num_refinements):
   class Refinement(SubDomain):
       def inside(self, x, on_boundary):
          ... # boolean tests, returns True if x is in the
              region to be refined

   ref_region = MeshFunction('uint', mesh, mesh.topology()
      .dim())
   ref_region.set_all(0)
   # Mark region to be refined
```

```
    Refinement().mark(ref_region,1)

    # Boolean mesh function
    markers = MeshFunction('bool', mesh, mesh.topology().
        dim())
    markers.set_all(False)
    # Transfer ref_region information to markers
    for cell in cells(mesh):
        if ref_region[cell.index()] == 1:
            markers[cell.index()] = True

    # Return new mesh to variable
    mesh = refine(mesh, markers)
```

The mid section declares a `MeshFunction` instance, `ref_region`, whose value is 1 in the cells to be refined and 0 elsewhere. This is used to define the boolean mesh function, `markers`. The code has to go the way through the `ref_region` function before declaring `markers` because the `set_all` and `mark` methods do not work for a boolean mesh function instance at the time of this writing. In the last line of the snippet, the refined mesh is used to declare a new `mesh` variable.

When proper refinement is done, the ball and stick can be extracted from the mesh. Using the subclasses defined above, this is easily done by

```
subdomains = MeshFunction('uint', mesh, mesh.topology().
    dim())
subdomains.set_all(0)
Ball().mark(subdomains,1)
Stick().mark(subdomains,2)
mesh = SubMesh(mesh, subdomains, 0)
```

`SubMesh` returns the parts of the mesh not marked as ball or stick, i.e., it only keeps the cells in which `subdomains` has the value 0.

Submesh extraction removes the cells contained in the ball and stick but does not define a smooth inner boundary. In order to obtain this, the edges close to the boundary have to be reorganized. This is performed by the `snap` methods of `Ball` and `Stick`, automatically called with

```
mesh.snap_boundary(Ball())
mesh.snap_boundary(Stick())
```

For an illustration of the algorithm, assume in general that the ball is situated at (x_0, y_0, z_0). The `snap` method of `Ball` first calculates the distance to its center,

$$r = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2},$$

for each mesh point. When $r < 3r_{\text{ball}}/2$ is satisfied, the following *snapping* is performed:

$$x \leftarrow x_0 + \frac{r_{\text{ball}}}{r}(x - x_0),$$

$$y \leftarrow y_0 + \frac{r_{\text{ball}}}{r}(y - y_0),$$

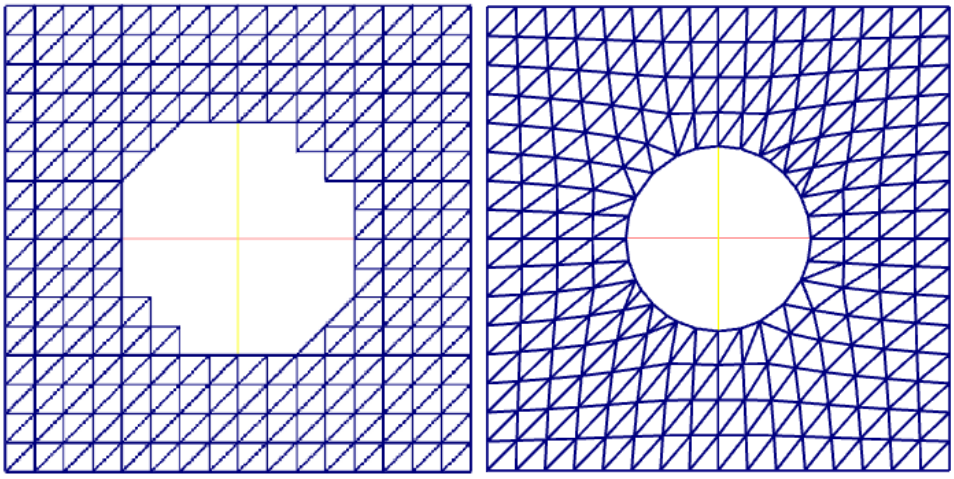

Figure 4.1: ***Snapping smooths the boundary.*** *In the mesh to the left, a submesh is extracted, creating a hole in the domain. To the right, a snap method has been called, moving the edges in order to give a smooth boundary.*

and

$$z \leftarrow z_0 + \frac{r_{\text{ball}}}{r}(z - z_0).$$

The same procedure applies to the two radial axes of the stick. Figure 4.1 shows snapping of a circular hole.

4.3 Boundary Conditions

Electric potentials are determined up to some constant and only have a physical interpretation when measured between two points in space. As shown in figure 2.10 on page 21, in MEA experiments, a reference electrode is normally immersed in the saline. The recordings express the difference between the potentials on the array electrodes and that of the reference electrode. Bakker et al. [10] report on having such an electrode immersed in the saline during measurements. Frey et al. [20] did a finite element simulation of the extracellular potentials arising from transmembrane currents calculated in NEURON [4]. A cylindrical outer geometry was used and the MEA assumed to be an electrical insulator, while all the other walls were grounded. The same outer boundary conditions were used in a modeling study by Joucla and Yvert [51].

The concept of boundary markers provides a flexible way of handling multiple boundary conditions. This is done by declaring a `MeshFunction` instance over cell facets, whose topological dimension is one lower than the cells. In 3D, the facets are cell surfaces, e.g., the ABC-plane of the tetrahedron in figure 3.4 on page 32. As far as boundary conditions are concerned, markers on exterior facets are of interest.

Boundary	Marker
Side with $\mathbf{n} = \hat{\mathbf{i}}$	1
Side with $\mathbf{n} = \hat{\mathbf{j}}$	2
Side with $\mathbf{n} = \hat{\mathbf{k}}$	3
Side with $\mathbf{n} = -\hat{\mathbf{i}}$	4
Side with $\mathbf{n} = -\hat{\mathbf{j}}$	5
Side with $\mathbf{n} = -\hat{\mathbf{k}}$	6
Ball surface	7
Stick surface	8
Source monopole	7
Sink monopole	8
Reference electrode	9

Table 4.1: ***Summary of boundary markers.*** *The table shows the boundary markers used in the implementations. The sides of the outer box are labeled from 1-6, the membranes with 7 and 8, and the reference electrode with a 9.* ***n*** *denotes an outward unit normal.*

Table 4.1 summarizes the markers used. The mathematical notation is kept consistent with the markers defined in the scripts, so, e.g., $\partial\Omega_5$ is the outer plane whose normal points in the negative y-direction. Figure 3.10 (b) on page 311 in the book by Langtangen [30] sketches a box whose sides are labeled in the same way as here. Since the whole outer boundary, including the MEA, is assumed to be insulating, boundaries 1-6 could as well have been given a single marker. In any case, the boundary conditions on these sides do not show up in the variational forms, so the particular marking has no impact on the computational efficiency after the mesh has been generated.

Since the set-up is surrounded by air and glass, practically insulating materials [57], the following homogeneous Neumann condition has been used for the surrounding walls in all the simulations reported in this thesis:

$$\sigma \frac{\partial \phi}{\partial n} = 0, \quad x \in (\partial\Omega_1 \cup \ldots \cup \partial\Omega_6)\,.$$

I.e., the net current across the interface is zero. The reference electrode has been modeled as a square plane situated in an upper corner of the fluid,

$$\begin{aligned}\partial\Omega_9 =& \{(x, y, z) : (1.9 \text{ mm} \leq x \leq 2.0 \text{ mm}) \\ & \cap\, (1.9 \text{ mm} \leq 2.0 \text{ mm}) \cap (z = 0.5 \text{ mm})\}.\end{aligned}$$

An alternative would be to model it explicitly in three dimensions, but since the electrode in any case is far away, its exact placement and geometry is not expected to play an important role. The relevant boundary condition becomes

$$\phi = 0, \quad (x, y, z) \in \partial\Omega_9.$$

A particular boundary is represented by a subclass of `SubDomain`. E.g., boundary 1 is declared with the code

```
class Boundary1(SubDomain):
   def inside(self, x, on_boundary):
       return on_boundary and abs(x[0]-box_x_dim/2.)<=
          DOLFIN_EPS
```

The `return` statement first tests whether a point actually is on the boundary and then whether its x-value equals 2.0 mm to machine precision. Similar subclasses are defined for all the other boundaries considered. The ball and stick boundaries are of particular interest. Their declarations are

```
class BallBoundary(SubDomain):
   def inside(self, x, on_boundary):
       r = sqrt((x[0]-ball_x_coor)**2 + (x[1]-ball_y_coor)
          **2 + (x[2]-ball_z_coor)**2)
       return on_boundary and r < 1.1*radius

class StickBoundary(SubDomain):
   def inside(self, x, on_boundary):
       r = sqrt((x[1]-stick_y)**2 + (x[2]-stick_z)**2)
       return on_boundary and r<1.5*stick_radius and
          stick_x_left <= x[0] <= stick_x_right+radius
```

Because the ball and the stick are connected to each other, the boundary markers will be somewhat inaccurate close to their junction. Again, the boolean tests use slightly larger radii. If the ball boundary is marked first, points on the stick boundary closer than or at $11r_{\text{ball}}/10$ from the center of the ball will be marked as parts of the ball boundary. Next, these values are overwritten by the method marking the stick boundary. Since most parts of the stick are far from any other boundary, $3/2$ times the stick radius can safely be used in the test as well as an extra term on the right-hand side of the last inequality in the `return` statement. The `on_boundary` method will in any case return `False` for interior points. However, `stick_x_left` has been used as the lower limit along the x-axis in order to accurately mark the junction between the ball and stick.

Finally, a `MeshFunction` instance is declared and set to appropriate values on the different boundary parts:

```
boundaries = MeshFunction('uint', mesh, mesh.topology().
   dim()-1)
Boundary1().mark(boundaries,1)
 ... # boundaries 2-6
ReferenceElectrode().mark(boundaries,9)
mesh.snap_boundary(Ball())
BallBoundary().mark(boundaries,7)
mesh.snap_boundary(Stick())
StickBoundary().mark(boundaries,8)
```

Note that snapping is performed just before the boundary is marked. This turned out to be necessary for correct marking. The Dirichlet condition on the reference electrode is now declared by

```
bc = DirichletBC(V, Constant(0), boundaries, 9)
```

The homogenous Neumann conditions for the insulating outer boundary do not enter the variational form and will therefore not be of concern. The transmembrane currents for the ball and stick give rise to the linear form

$$L(v) = \int_{\partial\Omega_7} J_{\text{ball}} v ds + \int_{\partial\Omega_8} J_{\text{stick}} v ds.$$

Translated into unifified form language it becomes

```
L = J_ball*v*ds(7) + J_stick*v*ds(8)
```

4.4 Conductivity Profile

In this work, only the saline and tissue have been explicitly modeled. The whole apparatus has been given the dimensions

$$(4\text{ mm}) \times (4\text{ mm}) \times (1\text{ mm}).$$

The slice dimensions were assumed to be

$$(2\text{ mm}) \times (2\text{ mm}) \times (0.3\text{ mm}).$$

A Cartesian coordinate system is placed in the center of the domain, meaning that the tissue can be represented by the set

$$\begin{aligned}\Omega_{\text{tissue}} =& \{(x,y,z) : (-1\text{ mm} \leq x \leq 1\text{ mm}) \\ & \cap (-1\text{ mm} \leq y \leq 1\text{ mm}) \cap (-0.5\text{ mm} \leq z \leq -0.2\text{ mm})\}\end{aligned}$$

and the whole domain by

$$\begin{aligned}\Omega =& \{(x,y,z) : (-2\text{ mm} \leq x \leq 2\text{ mm}) \\ & \cap (-2\text{ mm} \leq y \leq 2\text{ mm}) \cap (-0.5\text{ mm} \leq z \leq 0.5\text{ mm})\}.\end{aligned}$$

Hence, saline is represented by

$$\Omega_{\text{saline}} = \Omega \setminus \Omega_{\text{tissue}}.$$

4.4.1 Homogeneous and Isotropic Tissue

The media are distinguished by their conductivities. When σ is a constant scalar in each domain, it is defined by

$$\sigma = \begin{cases} \sigma_{\text{saline}} & \text{if } x \in \Omega_{\text{saline}} \\ \sigma_{\text{tissue}} & \text{if } x \in \Omega_{\text{tissue}} \end{cases}$$

Assuming a `Tissue()` subclass is properly defined, the following lines now mark the saline with a 0 and the tissue with a 1:

Layer	$\sigma_{\parallel}$ (S/m)	$\sigma_{\perp}$ (S/m)	Extent (mm)
II/III	0.319	0.231	$0.6 < x < 1.0$
IV	0.325	0.24	$0.2 < x < 0.6$
V	0.353	0.228	$-0.2 < x < 0.2$
IV	0.294	0.268	$-1.0 < x < -0.2$

Table 4.2: ***Anisotropic and inhomogeneous conductivity profile.*** *The table summarizes the values used in the conductivity tensors implemented. $\sigma_{\parallel}$ and $\sigma_{\perp}$ are the conductivity values parallel and perpendicular to the respective columns. The rightmost column shows the assumed spatial extent of the layers. Based on data from Goto et al. [22].*

```
subdomains = MeshFunction('uint', mesh, mesh.topology().
    dim())
subdomains.set_all(0)
Tissue().mark(subdomains,1)
```

Using the ideas presented in section 6.2 of the FEniCS tutorial [32], an `Expression` instance named `sigma` representing the piecewise constant conductivity is defined by

```
V0 = FunctionSpace(mesh, 'DG', 0)
sigma_saline = 3.0 # S/m
sigma_tissue = 0.3 # S/m
sigma = Function(V0)
sigma_values = [sigma_saline, sigma_tissue]
help = numpy.asarray(subdomains.values(), dtype = numpy.
    int32)
sigma.vector()[:] = numpy.choose(help, sigma_values)
```

Note that a space of constant functions, polynomials of degree zero, is used for the piecewise constant `sigma`. The bilinear form is now declared with

```
a = sigma*inner(grad(u),grad(v))*dx
```

4.4.2 Inhomogeneous and Anisotropic Tissue

Inhomogeneous and anisotropic tissue conductivity may quite easily be implemented in FEniCS. Goto et al. [22] measured the conductivity in the barrel cortex of Wistar rats. The conductivity tensor was estimated in each of the layers II/III, IV, V, and VI, and the mean values found will be used in the present implementation ([22], table V). To compare the potentials calculated assuming such a conductivity profile to those with $\sigma = 0.3$ S/m throughout, the slices need to be of the same size. Thus, the extent of the layers measured by Goto et al. were adjusted somewhat and is presented in the rightmost column of table 4.2.

Layers II/III, IV, and V have been given the height 0.4 mm, while layer VI is twice as tall. Since the soma is placed on the negative part of the x-axis in the ball-and-stick model, layer VI is confined longitudinally by

$$-1 \text{ mm} < x < -0.2 \text{ mm},$$

and layers II/III are within

$$0.6 \text{ mm} < x < 1 \text{ mm}.$$

Accordingly, the soma is placed somewhere in layer VI, with dendrites (stick) extending to layer IV. The modeling of anisotropy and inhomogeneity described here is done for testing its possible impact on the resulting MEA potentials and to illustrate how a conductivity tensor can be implemented in FEniCS. Hence, the accurate extent of the layers and the particular conductivity values are not critical.

In mesh generation with inhomogeneous conductivity, the `Tissue` needs to be replaced by one subclass for each layer. E.g., for layer V:

```
class LayerV(SubDomain):
    def inside(self, x, on_boundary):
        return -tissue_x_dim/2.+layer_VI_length<= x[0] <=
           -tissue_x_dim/2.+layer_VI_length+layer_V_length
            and -tissue_y_dim/2. <= x[1] <= tissue_y_dim
           /2. and -box_z_dim/2.<= x[2] <= -box_z_dim/2. +
            tissue_z_dim
```

Marking is done with

```
LayerVI().mark(subdomains,1)
LayerV().mark(subdomains,2)
LayerIV().mark(subdomains,3)
LayerII_III().mark(subdomains,4)
```

As the code snippet shows, the layers are marked from 1 to 4 in the positive direction along the x-axis. Saline gets a 0.

The conductivity function in Ω can be defined by

$$\sigma(x) = \begin{cases} \sigma_{\text{saline}} & \text{if } x \in \Omega_0 \\ \sigma_{\text{VI}} & \text{if } x \in \Omega_1 \\ \sigma_{\text{V}} & \text{if } x \in \Omega_2 \\ \sigma_{\text{IV}} & \text{if } x \in \Omega_3 \\ \sigma_{\text{II/III}} & \text{if } x \in \Omega_4 \end{cases} \tag{4.1}$$

where Ω_i denotes the subdomain marked with i. In this case, σ_{saline} is a scalar, whereas the other four are second order tensors. Having partitioned Ω into subdomains, the bilinear form[3],

$$a(u, v) = \int_{\Omega} (\sigma(x)\nabla u) \cdot \nabla v \mathrm{d}x,$$

can be written as a sum of the integrals over each subdomain,

$$a(u, v) = \sum_{i=0}^{4} \int_{\Omega_i} (\sigma(x)\nabla u) \cdot \nabla v \mathrm{d}x. \tag{4.2}$$

[3]The parentheses are included to specify that the matrix-vector product must be evaluated *before* the inner product.

The conductivity tensors were programed as matrices. The following code shows the declaration of σ_V.

```
# Define the matrix elements
s11 = Expression('s', {'s': sigma_V_parallel})
s12 = Constant(0.0); s13 = Constant(0.0); s21 = Constant
    (0.0)
s22 = Expression('s', {'s':sigma_V_perpendicular})
s23 = Constant(0.0); s31 = Constant(0.0); s32 = Constant
    (0.0)
s33 = Expression('s', {'s':sigma_V_perpendicular})

# Declare the matrix
layer_V_conductivity = as_matrix(((s11,s12,s13),(s21,s22,
    s23),(s31,s32,s33)))
```

The bilinear form is declared by writing out the whole sum of equation (4.2):

```
a = saline_conductivity*inner(grad(u),grad(v))*dx(0) +
    inner(layer_VI_conductivity*grad(u),grad(v))*dx(1) +
    inner(layer_V_conductivity*grad(u),grad(v))*dx(2) +
    inner(layer_IV_conductivity*grad(u),grad(v))*dx(3) +
    inner(layer_II_III_conductivity*grad(u),grad(v))*dx(4)
```

Just as surface integrals over parts of the boundary were performed by handing a marker to `ds`, the argument given to `dx` defines the subdomain over which to calculate the integral. The variational problem is now solved with

```
phi = VariationalProblem(a, L, bc, cell_domains=subdomains
    , exterior_facet_domains=boundaries).solve()
```

Information about the subdomain markers used in the weak form is passed to the `cell_domains` attribute.

4.5 Visualization

FEniCS comes with the plotting tool Viper. It is based on the Visualization Toolkit (VTK) [7]. A solution, u, is plotted by the statement

```
plot(u)
interactive()
```

Viper is intended for quick-and-easy visualization while working with the code. More sophisticated software is needed for data analysis or to create figures for use in papers and presentations.

The grids of finite element computations are often highly irregular. For a 3D tetrahedral mesh, whose vertices are spread in a nonuniform way, an array of data values for all points does not contain sufficient information for plotting. The connectivity pattern between vertices must also be provided. In contrast, the grids used in finite difference methods are regular, so the connectivity follows straightforwardly from the location of nodes along the axes of the coordinate system used.

Description	Symbol	Value
Membrane resistance	r_{m}	$3.0 \times 10^4\ \Omega\mathrm{cm}^2$
Dendrite resistance	r_{i}	$150\ \Omega\mathrm{cm}$
Membrane capacitance	c_{m}	$1\ \mu\mathrm{F}/\mathrm{cm}^2$
Stick length	l	1 mm
Stick diameter	d	2 μm (3 μm)
Ball radius	r_{ball}	10 μm (42.5 μm)
Stimulus amplitude	I_0	250 pA
Membrane time constant	$\tau_{\mathrm{m}} = r_{\mathrm{m}} c_{\mathrm{m}}$	3 μs
Electrotonic length	$\lambda = \sqrt{d r_{\mathrm{m}}/4 r_{\mathrm{i}}}$	10^{-3} m
Height above the MEA	z_0	100 μm (200 μm)

Table 4.3: ***Ball-and-stick parameters.*** *The table summarizes the parameter values used in the ball-and-stick model. The values in parentheses are used in the adjusted model of section 5.5.*

VTK-based plotting tools are very good for irregular grids. One such, ParaView [5], is freely available and has been used for the generation of figures in this thesis. In the case of a timeloop, the following code is needed to export a finite element field, u, for use with ParaView:

```
file = File('u.pvd')
while t < T:
    ... # code
    file << u
```

In each iteration, a file `u#.vtu` is created[4], containing the data at the current time in VTK format. The file `u.pvd` is a list keeping track of the data files. The ParaView tutorial [38] gives a good introduction to the software. ParaView also has support for automatic plot generation with Python scripts, which is convenient when creating a large number of figures, e.g., for different parameter values.

4.6 Membrane Currents

Table 4.3 shows the parameters used in the implementations described in this section.

The weak form of Laplace's equation for volume conduction around a ball-and-stick neuron is given by

$$\int_\Omega \sigma \nabla u \cdot \nabla v \mathrm{d}x = \int_{\partial\Omega_7} J_{\mathrm{ball}} v \mathrm{d}s + \int_{\partial\Omega_8} J_{\mathrm{stick}} v \mathrm{d}s.$$

When the soma receives an electrode current, $I_{\mathrm{e}}(t)$, the resulting membrane currents are $J_{\mathrm{ball}}(t)$ and $J_{\mathrm{stick}}(x,t)$. As far as stimulation of a single frequency is

[4] # is replaced by a sequence of numbers starting with 000000 at the first time step.

concerned, the currents are given by the real parts of equation (2.17) on page 18 and (2.20) on page 19. Finding the real parts analytically leads to very complicated expressions, so the approach used here is to rather evaluate the complex expressions numerically and then use the real part. The code defining the `Expression` instances representing the membrane currents now become more involved than for the two-monopole model. Two ways of defining them were tested, both of which will be described in the following.

The first approach consists of creating a subclass of `Expression`, overloading its `eval` method. The essential parts of the code are described by:

```
class MembraneCurrent(Expression):
    def eval(self,value,x):
        J_m = ... # Expression for stick current
        value[0] = J_m.real   # return real part

class SomaCurrent(Expression):
    def eval(self, value, x):
        Y = ... # Expression for admittance
        J_s = Y*... # Expression for soma current
        value[0] = J_s.real # return real part

J_s = SomaCurrent()
J_m = MembraneCurrent()
```

In addition to these lines, the necessary parameters need to be supplied.

Defining complex expressions in this way produces accurate results, but each time they are evaluated a callback from C++ to Python is involved, leading to slow code [36]. On the other hand, when the `Expression` instances are defined with a C++ string, the formula is *precompiled* [25]. This is sketched in the following snippet:

```
# String with C++ code for soma current
soma_code = """
class SomeExpr : public Expression
{
public:
    SomeExpr() : Expression(),... {}
    // variable declarations
    void eval(Array<double>& values, const Data& data)
        const
    {
    ... // code defining current
        values[0] = J_soma.real();
    }
};
"""
# String with C++ code for stick current
stick_code = """
class SomeExpr : public Expression
```

```
{
public:
  SomeExpr() : Expression(), ...  {}
  // variable declarations
  void eval(Array<double>& values, const Data& data) const
  {
    ... // code defining current
    values[0] = J_m.real();
  }
};
"""
J_ball = Expression(soma_code)
J_stick = Expression(stick_code)
```

A test for only one time step with the ball-and-stick neuron on a mesh of 41342 vertices took 126 seconds for subclassed expressions and 26 seconds with precompiled expressions. Running a time loop on the same mesh, the difference was even more pronounced. Hence, precompiled expression have been used for the simulations presented in this thesis.

4.7 Simple Kirchhoff-Fix

The analytic expressions for the membrane currents resulting from a current injection in the soma obey Kirchhoff's current law. I.e., the following equation holds at any time:

$$I_e(t) = I_{\text{ball}}(t) + \int_0^l i_{\text{stick}}(x,t)\mathrm{d}x.$$

$I_{\text{ball}}(t)$ is the total current leaving the soma, and $i_{\text{stick}}(x,t)$ is the outward membrane current per unit length of the stick, whose soma end is situated at $x = 0$. In the implementations, the *current density* is used. For the soma, this is simply

$$J_{\text{ball}}(t) = \frac{I_{\text{ball}}}{S},$$

where S is its surface area. Since the soma and stick are connected, the actual value of S computed by FEniCS depends sensitively on the boundary marking close to their junction. By using this computed S in the definition of the soma current density, numerical evaluation of

$$\int_{\partial\Omega_7} J_{\text{ball}}\mathrm{d}s,$$

in any case reproduces I_{ball} to machine precision.

The errors are larger for the stick because its membrane current depends on longitudinal position. The expression for the stick current is declared in the code

as a function of distance from the soma end. But this is also where there are significant inaccuracies. The current density is defined as

$$\frac{i_{\text{stick}}(x,t)}{\pi d},$$

but close to the ball the points at the distance x marked as being on the membrane do not exactly correspond to a circumference πd. Also, the very thin distal end of the stick is not marked separately from the curved surface, so the Neumann condition for the stick current is also applied there. This means that the membrane current at its end, $i_{\text{stick}}(l,t)$, will be smeared over a larger surface area than intended. Since the end area of the long and thin stick is very small compared to the rest of its surface, this error is not expected to play a significant role.

In any case, some error will arise, i.e., numerical evaluation of

$$\int_{\partial\Omega_8} J_{\text{stick}} \mathrm{d}s$$

does not reproduce

$$\int_0^l i_{\text{stick}}(x,t) \mathrm{d}x$$

to machine precision (table 4.4). A simple fix was created, to be explained in the following. The goal is to make the membrane satisfy Kirchhoff's current law, expressed in terms of current densities as,

$$I_e = \int_{\partial\Omega_7} J_{\text{ball}} \mathrm{d}s + \int_{\partial\Omega_8} J_{\text{stick}} \mathrm{d}s. \tag{4.3}$$

For sinusoidal stimulus, the rightmost term can be written as[5]

$$\int_{\partial\Omega_8} J_{\text{stick}} \mathrm{d}s = \mathrm{Re}\left(\int_{\partial\Omega_8} \frac{\mathbf{H}}{\pi d} \frac{\hat{\mathbf{I}}_0 e^{j\omega t}}{\mathbf{Y}_{\text{ball}} + \mathbf{Y}_{\text{stick}}} \mathrm{d}s \right).$$

The phase-dependent current amplitude, $\hat{\mathbf{I}}_0$, is just a real number and can be pulled out of the integral. This gives

$$\int_{\partial\Omega_8} J_{\text{stick}} \mathrm{d}s = I_0^{\text{stick}} \cdot \mathrm{Re}\left(\int_{\partial\Omega_8} \frac{\mathbf{H}}{\pi d} \frac{e^{j\omega t}}{\mathbf{Y}_{\text{ball}} + \mathbf{Y}_{\text{stick}}} \mathrm{d}s \right),$$

where the amplitude has been named I_0^{stick}, to specify that it is used in the stick current formula. Equation (4.3) can now be written as

$$I_0^{\text{stick}} = \frac{I_{\text{e}} - \int_{\partial\Omega_7} J_{\text{ball}} \mathrm{d}s}{\mathrm{Re}\left(\int_{\partial\Omega_8} \frac{\mathbf{H}}{\pi d} \frac{e^{j\omega t}}{\mathbf{Y}_{\text{ball}} + \mathbf{Y}_{\text{stick}}} \mathrm{d}s \right)}. \tag{4.4}$$

[5] Cf. equation (2.20) on page 19.

I_0^{stick} is a constant multiplier to the stick current and will be adjusted to make equation (4.3) hold. The amplitude used to calculated the somatic membrane current will be fixed at I_0. By changing, e.g., the membrane resistance instead, its shape could also have been altered[6].

Multiplying both the numerator and denominator of equation (4.4) by I_0, we get

$$I_0^{\text{stick}} = \frac{I_0 \left(I_{\text{e}} - \int_{\partial\Omega_7} J_{\text{ball}}(t) \mathrm{d}s \right)}{\text{Re} \left(\int_{\partial\Omega_8} \frac{\mathbf{H}}{\pi d} \frac{\hat{\mathbf{I}}_0 e^{j\omega t}}{\mathbf{Y}_{\text{ball}} + \mathbf{Y}_{\text{stick}}} \mathrm{d}s \right)},$$

which is the same as

$$I_0^{\text{stick}} = \frac{I_0 \left(I_{\text{e}} - \int_{\partial\Omega_7} J_{\text{ball}}(t) \mathrm{d}s \right)}{\int_{\partial\Omega_8} J_{\text{stick}}(t) \mathrm{d}s}.$$

The denominator is the stick current originally computed, i.e., the one causing trouble with Kirchhoff's current law. Now, re-calculating the stick current using I_0^{stick}, the net current entering the neuron will be zero, to machine precision, at any time.

The essential parts of the code are reproduced here.

```
# Adjust stick current to obey Kirchhoff's current law
while t < T:
    # Update time
    J_soma.t = t
    J_stick.t = t

    # Soma current and preliminary stick current
    soma_current = assemble(J_soma*ds(7), mesh=mesh,
        exterior_facet_domains=boundaries)
    stick_current = assemble(J_stick*ds(8), mesh=mesh,
        exterior_facet_domains=boundaries)

    # Calculate electrode current
    I_e = I_0*cos(omega*t) # [mA], electrode current

    # Calculate correction current
    I_0_corr = (I_e-soma_current)*I_0/stick_current

    # Update the value used in J_stick
    J_stick.I_0 = I_0_corr

    ... # Code for solving variational problem

    # Update time
    t += dt
```

[6] See, e.g., fig. 6 in reference [34].

t/T	ΔI_{before}	ΔI_{after}	I_0^{stick}
0	8.22×10^{-9}	6.12×10^{-22}	2.59×10^{-7}
1/8	5.77×10^{-9}	1.92×10^{-22}	2.59×10^{-7}
2/8	-6.71×10^{-11}	9.28×10^{-24}	1.55×10^{-7}
3/8	-5.86×10^{-9}	3.06×10^{-21}	2.59×10^{-7}
4/8	-8.22×10^{-9}	-6.12×10^{-22}	2.59×10^{-7}
5/8	-5.77×10^{-9}	1.94×10^{-21}	2.59×10^{-7}
6/8	6.71×10^{-11}	7.16×10^{-24}	1.55×10^{-7}
7/8	5.86×10^{-9}	2.98×10^{-21}	2.59×10^{-7}

Table 4.4: ***Current correction for 1 Hz stimulation.*** *The table shows the effect of the Kirchhoff-fix-algorithm. 8 time steps of the period, $T = 1/f = 1$ s, are shown. ΔI_{before} and ΔI_{after} are the differences between the electrode current and the return current before and after the current is adjusted, respectively (equation (4.5)). I_0^{stick} is the adjusted current amplitude, and the original value was $I_0 = 250 \times 10^{-9}$ mA. All units are in mA.*

```
    # Reset electrode current before next time step
    J_stick.I_0 = I_0
```

Table 4.4 shows corrections for a ball-and-stick neuron receiving somatic electrode current, $I_{\text{e}}(t) = (250\ \text{pA})\cos(2\pi(1\ \text{Hz})t)$. The period of I_{e} is divided into 8 equally spaced time steps. The deviation from Kirchhoff's current law is calculated as

$$\Delta I = I_{\text{e}} - \left(\int_{\partial\Omega_7} J_{\text{ball}}(t)\mathrm{d}s + \int_{\partial\Omega_8} J_{\text{stick}}(t)\mathrm{d}s \right). \tag{4.5}$$

For nonzero phases of the stimulus, a 3.5 % correction of I_0 was the result. In the zero phases ($t = 2T/8$ and $t = 6T/8$), the original error is more pronounced. The adjusted amplitude for the stick current expression, $I_0^{\text{stick}} = 155$ pA, represents a 38 % change of the original value. The third column of table 4.4 shows that, in any case, zero net current into the membrane was achieved to very high precision after the adjustment. This consistent pattern in the values of I_0^{stick} was not found for all frequencies. With $f = 100$ Hz, for example, I_0^{stick} for nonzero I_e varied between 257 and 266 pA and for zero electrode currents it was found to be 152 pA.

The large error in the zero phase of the electrode current shows that the simulation results at these times should be interpreted carefully. On a more powerful computer than the laptop used in this work, a finer mesh around the ball-and-stick could have the potential to improve the accuracy.

Chapter 5

Results

5.1 Numerical Accuracy

For a homogeneous, isotropic, and infinite medium containing N point current sources, the electric potential at some point r is given by equation (1.1),

$$\phi(r) = \frac{1}{4\pi\sigma} \sum_{n=1}^{N} \frac{I_n}{r_n}.$$

In order to test the numerical accuracy, an infinite medium containing two current monopoles was simulated and the computed potentials compared to the exact solution given by the above equation. Since a computer cannot handle an infinite domain, infinity was mimicked by imposing $\phi(r)$ as the outer boundary condition. The deviation between the numerical and the exact solution in the inner domain then indicates the size of the discretization error. In the FEM implementation, the point current sources have been converted to inner boundaries, through which the currents enter. According to Gauss' law, this has no physical effect on the potential outside of the boundaries, as long as the current leaves the monopoles in a radially symmetric way.

The two monopoles were placed at

$$(x_1, y_1, z_1) = (-0.5 \text{ mm}, 0.0 \text{ mm}, 0.0 \text{ mm})$$

and

$$(x_2, y_2, z_2) = (0.5 \text{ mm}, 0.0 \text{ mm}, 0.0 \text{ mm}).$$

The tissue was assumed to be a box of size

$$(8 \text{ mm}) \times (8 \text{ mm}) \times (8 \text{ mm}),$$

which is 256 times the slice volume used in the other simulations. The monopole at $x = -0.5$ mm was set as a current source and the one at $x = 0.5$ mm a current sink[1], so the net current entering the domain was zero.

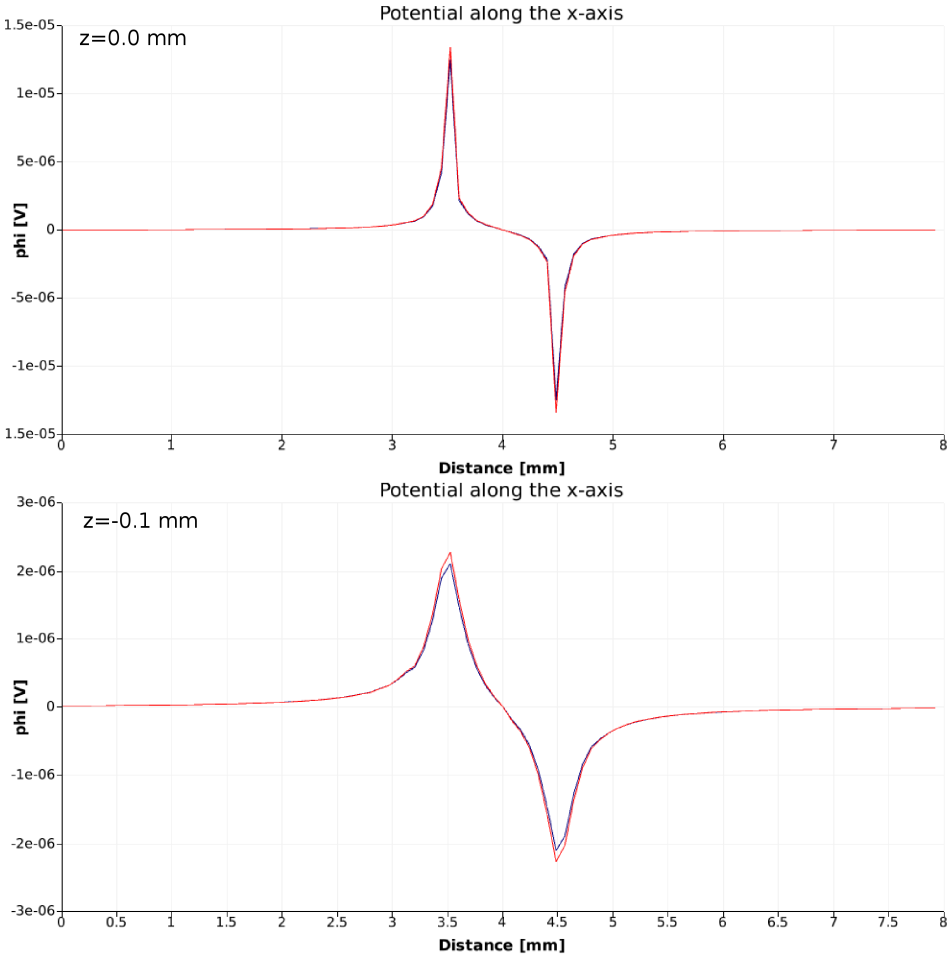

Figure 5.1: ***Verification of numerical accuracy.*** *Line plots parallel to the two-monopole axis are shown. Blue line: FEM solution. Red line: infinite medium solution. In the upper figure, the potential along a line crossing through the monopoles is shown, i.e., along the x-axis at* $y = 0$ *and* $z = 0$*. In the lower figure, the potential is shown along a line* 0.1 *mm below the two-monopole, i.e., along the x-axis at* $y = 0$ *and* $z = -0.1$ *mm. The curves in the upper figure are nearly identical. Also in the lower figure they are very close, except just below the monopoles, i.e., around* 3.5 *mm and* 4.5 *mm along the horizontal axis. This suggests that the numerical accuracy is good. See section 5.1 for details.*

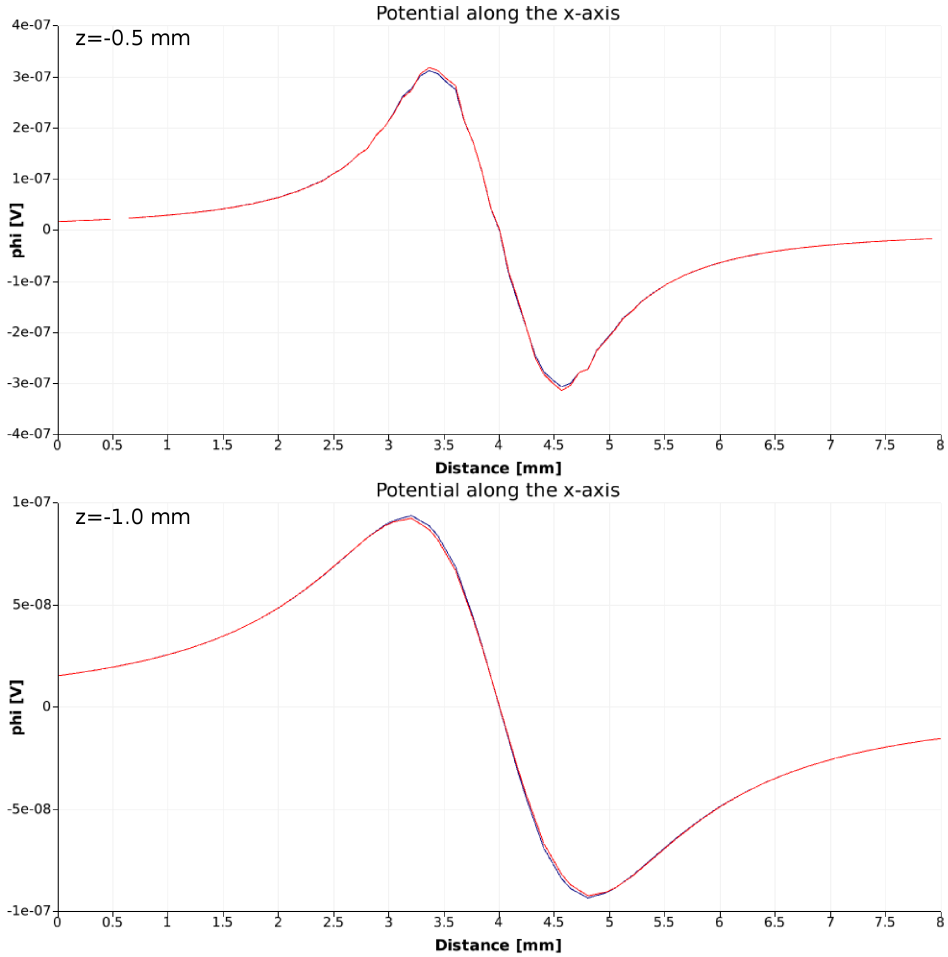

Figure 5.2: ***Verification of numerical accuracy.*** *Line plots parallel to the two-monopole axis are shown. Blue line: FEM solution. Red line: infinite medium solution. In the upper figure, the potential along a line* 0.5 *mm below the monopoles is shown, i.e., along the x-axis at* $y = 0$ *and* $z = -0.5$ *mm. In the lower figure, the potential is shown along a line* 1 *mm below the monopoles, i.e., along the x-axis at* $y = 0$ *and* $z = -1$ *mm. Both curves are nearly identical, confirming that the numerical solution is correctly implemented. See section 5.1 for details.*

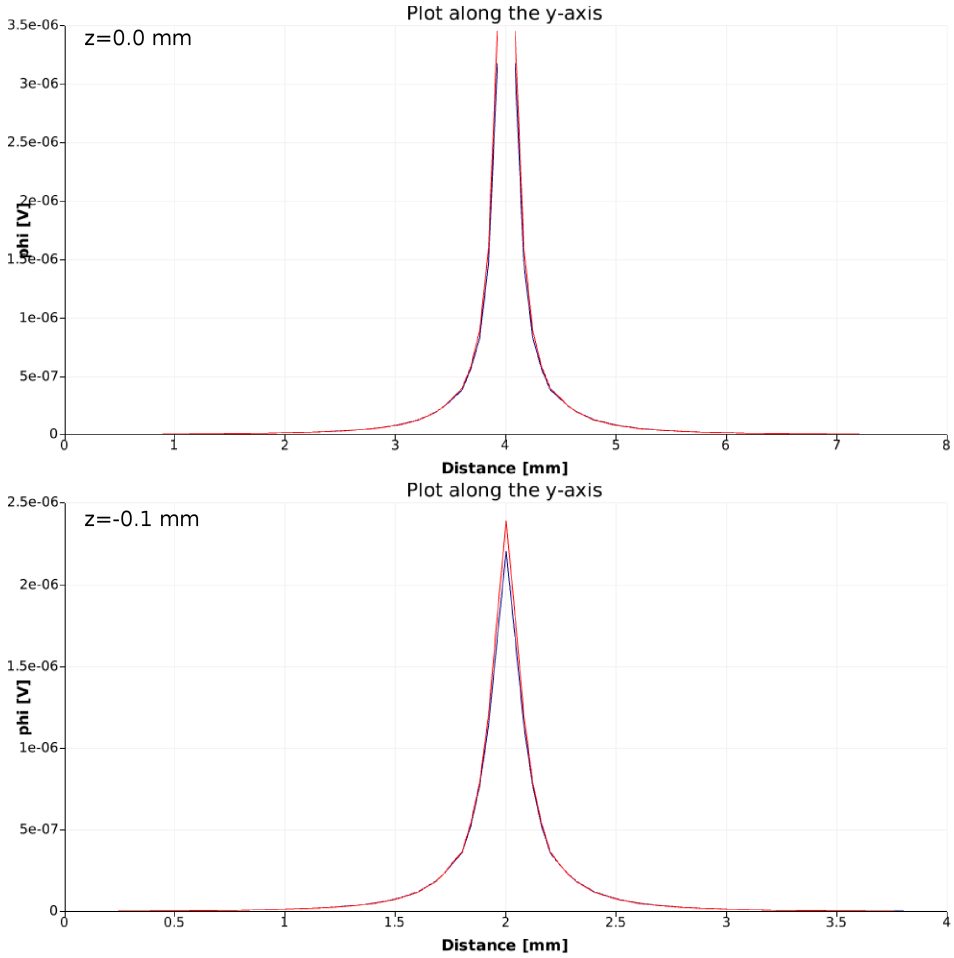

Figure 5.3: ***Verification of numerical accuracy.*** *Line plots perpendicular to the two-monopole axis are shown. Blue line: FEM solution. Red line: infinite medium solution. In the upper figure, the potential is plotted along a line crossing through the current source, i.e., along the y-axis at $x = -0.5$ mm and $z = 0$ mm. In the lower figure, the potential is shown along a line 0.1 mm below the source, i.e., along the y-axis at $x = -0.5$ mm and $z = -0.1$ mm. The lower curve shows that the analytical solution predicts a higher potential just below the monopole (at 2 mm on the axis), but otherwise the agreement is excellent. The discrepancy very close to the current source is also present in the lower image of figure 5.1. See section 5.1 for details.*

Using the labels from section 4.3, the boundary value problem to be compared with the analytical solution is

$$\begin{aligned}
\nabla \cdot (\sigma \nabla \phi) = &\quad 0 &&\quad x \in \Omega \\
\phi = &\quad \tfrac{1}{4\pi\sigma}\left(\tfrac{I}{r_1} - \tfrac{I}{r_2}\right) &&\quad x \in \partial\Omega_1 \cup \ldots \cup \partial\Omega_6 \\
\sigma \frac{\partial \phi}{\partial n} = &\quad \tfrac{I}{S_7} &&\quad x \in \partial\Omega_7 \\
\sigma \frac{\partial \phi}{\partial n} = &\quad \tfrac{-I}{S_8} &&\quad x \in \partial\Omega_8
\end{aligned}$$

The current was set to $I = 1$ nA. The monopoles in the mesh were of radius

$$r_{\mathrm{m}} = 10\ \mu\mathrm{m},$$

and S_i ($i = 7, 8$) represents their surface area.

In this comparison, it is important that the point current I of the analytical solution exactly equals the current density times surface area, JS, used in the FEM discretization. Otherwise, the sources get different strengths in the two methods, and a comparison makes no sense. Since the vertices on the surface of the ball are connected by straight lines, the numerically computed area, S_i, in general differs from $4\pi r_{\mathrm{m}}^2$. Hence, the surface area of each ball was found by numerically evaluating

$$S_i = \int_{\partial\Omega_i} \mathrm{d}s \quad \text{for } i = 7, 8.$$

An example illustrates this procedure for the current source:

```
# Calculate surface area
area7 = assemble(Constant(1)*ds(7), mesh=mesh,
    exterior_facet_domains=boundaries)
# Find current density across membrane
J_membrane7 = Expression('xI/(area)', {'xI':I, 'area':
    area7})
```

Similar code applies for the sink. In the mesh used, numerical experiments showed that the areas differed from $4\pi r_{\mathrm{m}}^2$ by no more than 1 %.

The analytical solution was projected onto the mesh by using the same space of linear functions as for the variational problem. This means that it is exact at all vertices but approximated by straight lines between. To check for errors arising because of this, the solution was also projected onto the mesh using a function space of second order piecewise polynomials. The resulting plots were practically indistinguishable, so piecewise linear functions were chosen in the implentations, for computational efficiency.

In figures 5.1 and 5.2, the potential is plotted along lines parallel with the axis connecting the two monopoles. It is clear that the FEM solution reproduces the

[1] Cf. figure 2.7 on page 16

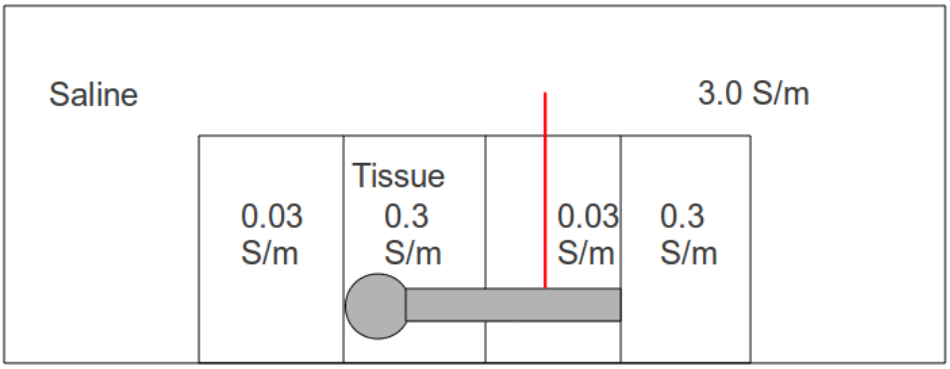

Figure 5.4: ***Ball-and-stick neuron in an inhomogeneous tissue***. *The tissue is split into 4 equally sized chunks, and each is assigned a conductivity value. The surrounding saline solution has the conductivity 3.0 S/m. The position of the ball-and-stick neuron is also illustrated. This model is used for the simulations in section 5.2. The red line sketches the axis along which the plots of figure 5.9 were taken for conductivity values varying between* 0.03 *S/m and* 0.3 *S/m.*

analytical solution in a good way. Figure 5.3 shows the potential along a line perpendicular to the two-monopole axis. The upper image plots the potential along the line crossing through the source, and here the agreement is very good except for some boundary effects just on the surface of the ball. The second row shows the corresponding line plot at $z = -0.1$ mm, which is 0.09 mm below the surface of the ball. The analytical solution is seen to give a higher value just below the monopole, but they quickly become essentially equal.

These figures, together with surface plots not shown here, confirm that the simulations were very precise. Since similar implementations are used for the rest of the problems considered, the high numerical accuracy is presumed to apply to these cases as well.

5.2 Large Inhomogeneity and Anisotropy

To obtain insight into how anisotropic and inhomogeneous conductivity affects the extracellular potentials, exaggerated and probably unrealistic values were used. First, the tissue was split into 4 equally sized pieces, each having isotropic conductivities either 0.3 S/m or 0.03 S/m. By comparing the extracellular potentials found with this conductivity profile to those calculated for a homogeneous tissue with $\sigma = 0.3$ S/m, the impact of inhomogeneity can be highlighted. Figure 5.4 shows a sketch of the conductivity profile used in the simulations. Next, the impact of anisotropy was investigated by simulating tissue with conductivity $\sigma_{\parallel} = 0.3$ S/m parallel to the column and $\sigma_{\perp} = 0.03$ S/m perpendicular to the column. This is illustrated by figure 5.5.

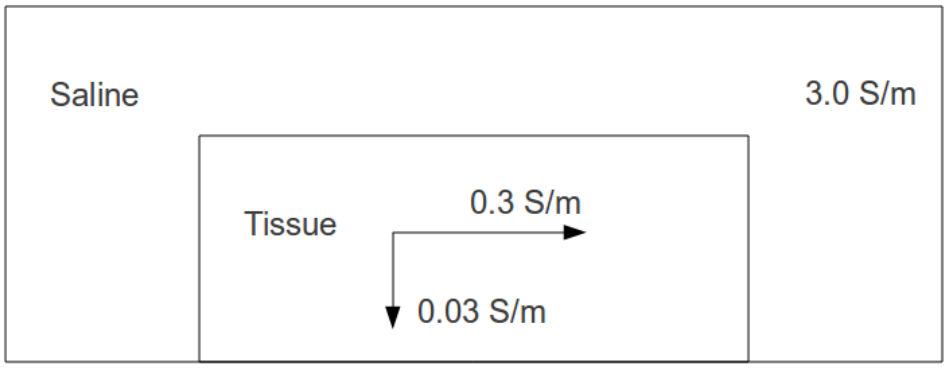

Figure 5.5: ***Anisotropic tissue***. *Illustration of the conductivity profile used to test the impact of anisotropic conductivity. The direction along the axons has a conductivity of* 0.3 *S/m and the perpendicular direction* 0.03 *S/m. The surrounding saline solution has the conductivity 3.0 S/m. The position of the ball-and-stick (not shown) is the same as in figure 5.4.*

All cases were run with a ball-and-stick model receiving somatic electrode current,

$$I_{\mathrm{e}} = (250\ \mathrm{pA})\sin(2\pi(10\ \mathrm{Hz})t).$$

The following plots illustrate the simulation results:

- Figure 5.6: Homogeneous and isotropic tissue
- Figure 5.7: Inhomogeneous and isotropic tissue
- Figure 5.8: Homogeneous and anisotropic tissue

The cuts shown in the figures are parallel to the ball-and-stick neuron and as wide as the tissue. Their height is 0.5 mm, i.e., the lower 3/5 of each plot in the figures shows the potential in the tissue, while the upper 2/5 is saline.

Comparison of figures 5.6 and 5.7 reveals that large inhomogeneities in the conductivity create substantial deviations between the calculated extracellular potentials. This is readily explained by Ohm's law, $\mathbf{J} = -\sigma\nabla\phi$. For a given current leaving the membrane, the potential gradient is inversely proportional to conductivity. The effect of anisotropy is shown by figure 5.8. In the case of higher conductivity along the stick, the potential is seen to have a steeper decay perpendicular to it, explained by just the same Ohmic argument. A qualitative feature to notice is that the potential is more confined along the stick in the anisotropic case.

Parameter dependence was also investigated by plotting the extracellular potential perpendicularly upward from the stick, illustrated by the red line in figure 5.4. The conductivity in the regions labeled with 0.03 S/m in figure 5.4 was gradually increased, with all other values kept constant. The curves in figure 5.9 clearly show that the rate of change of the extracellular potential away from the stick is larger for low conductivity values and smaller for high conductivity values.

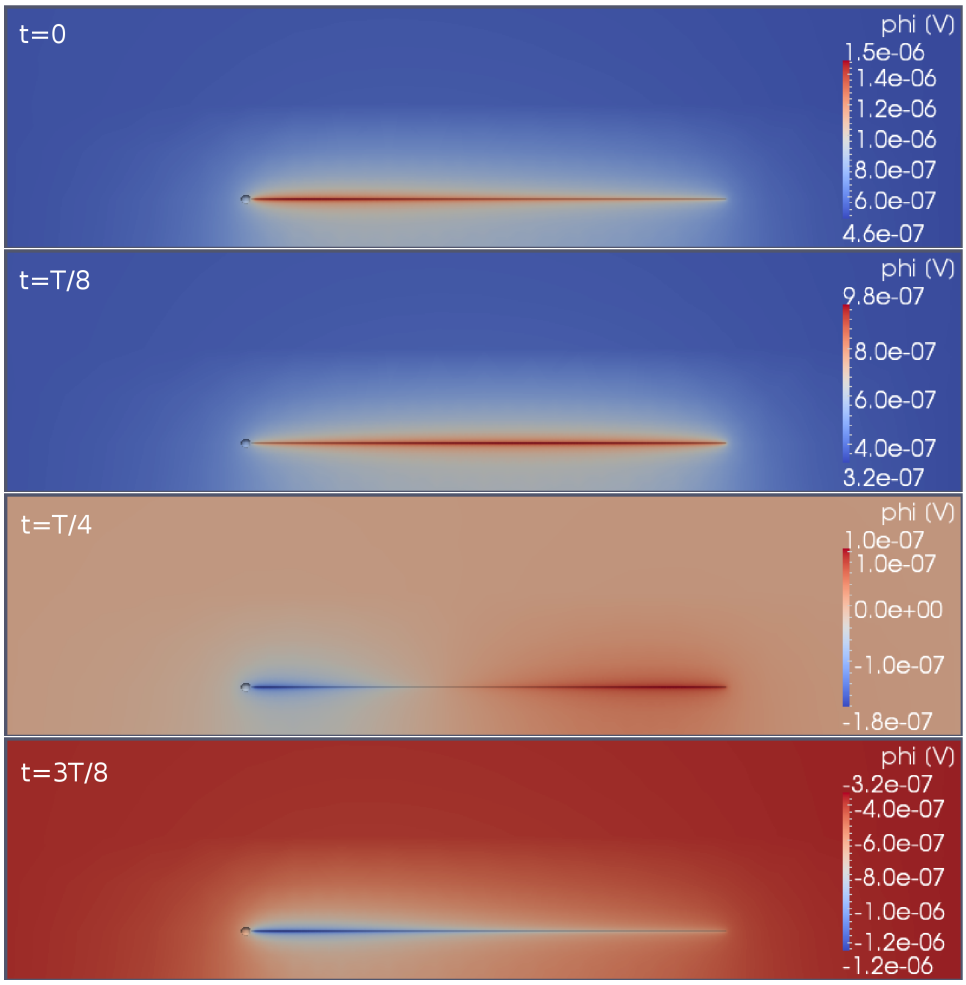

Figure 5.6: ***Ball-and-stick neuron in homogeneous and isotropic tissue.*** *The lower 3/5 of each image shows a ball-and-stick neuron in a tissue with conductivity 0.3 S/m, and the upper 2/5 is saline with conductivity 3.0 S/m. An electrode current,* $I_e = (250\ pA)\cos(2\pi ft)$*, is injected in the soma, producing transmembrane return currents. The four rows show* $t = 0$*,* $t = T/8$*,* $t = T/4$*, and* $t = 3T/8$ *for* $f = 10$ *Hz, i.e.,* $T = 1/f = 0.1$ *s.*

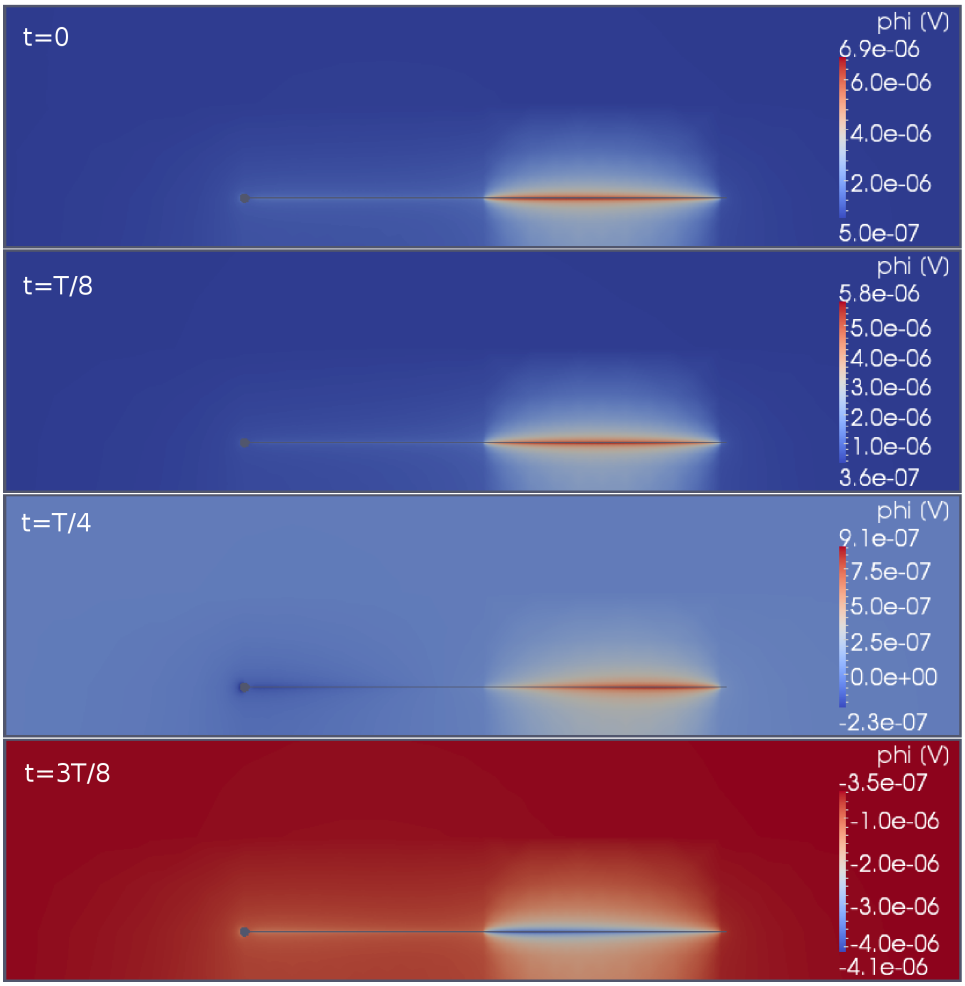

Figure 5.7: ***Ball-and-stick neuron in inhomogeneous and isotropic tissue.*** *The lower 3/5 of each image shows a ball-and-stick neuron in a tissue with conductivity stepwise varying between* 0.3 *S/m and* 0.03 *S/m (figure 5.4). The upper 2/5 is saline with conductivity 3.0 S/m. An electrode current,* $I_e = 250\,pA\cos(2\pi ft)$*, is injected in the soma, producing transmembrane return currents. The four rows show* $t = 0$*,* $t = T/8$*,* $t = T/4$*, and* $t = 3T/8$ *for* $f = 10\,Hz$*, i.e.,* $T = 1/f = 0.1$ *s. The impact of the inhomogeneity is clearly visible. In the part along the stick with* $\sigma = 0.03$ *S/m, the extracellular potential is varying much more than in the part with* $\sigma = 0.3$ *S/m. See section 5.2 for a discussion.*

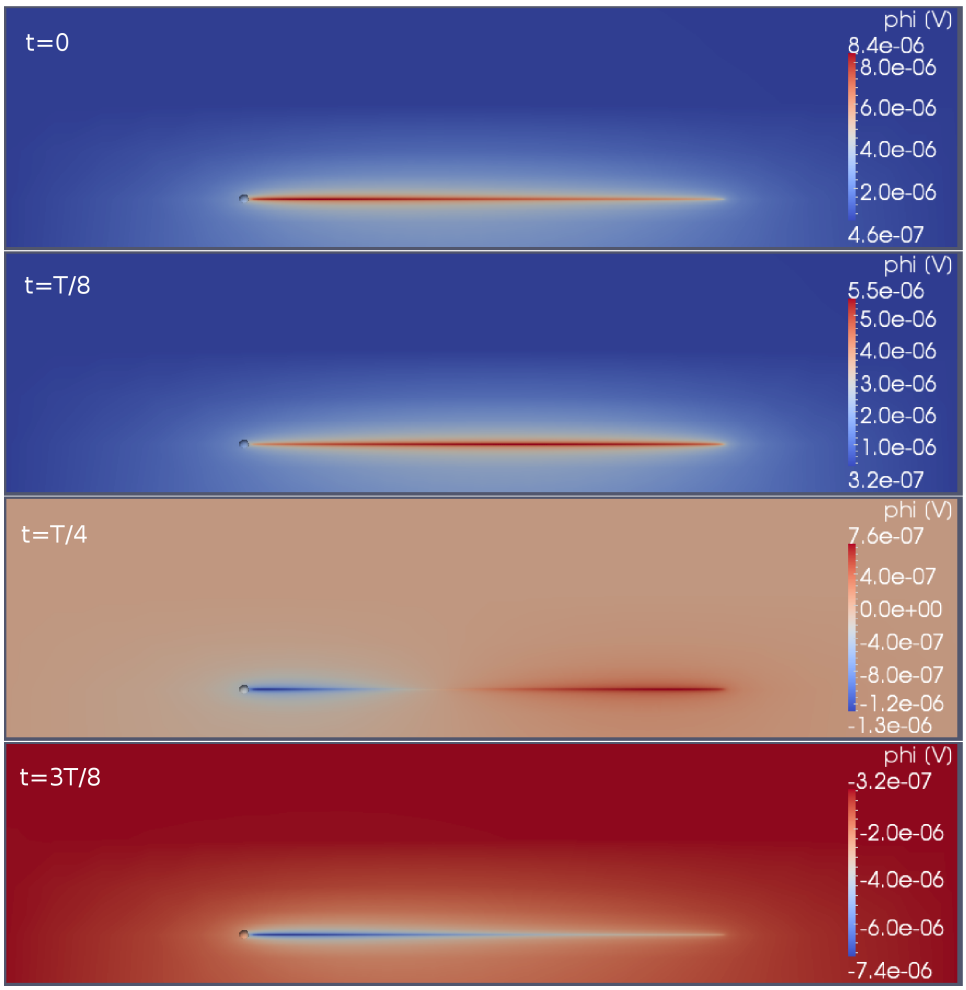

Figure 5.8: ***Ball-and-stick neuron in homogeneous and anisotropic tissue.*** *The lower 3/5 of each image shows a ball-and-stick neuron in a tissue with conductivity* 0.3 *S/m in the horizontal direction and* 0.03 *S/m in the directions perpendicular to the stick, as shown in figure 5.5. The upper 2/5 is saline with conductivity 3.0 S/m. An electrode current,* $I_\mathrm{e} = 250$ *pA* $\cos(2\pi f t)$*, is injected in the soma, producing transmembrane return currents. The four rows show* $t = 0$*,* $t = T/8$*,* $t = T/4$*, and* $t = 3T/8$ *for* $f = 10$ *Hz, i.e.,* $T = 1/f = 0.1$ *s. The impact of the conductivity profile can be seen by a comparison to figure 5.6. With anisotropic tissue, whose perpendicular conductivity is lower, the potential gradient away from the neuron is steeper. The maximum magnitude of the potential with respect to the reference electrode is also about 5 orders of magnitude larger than for homogeneous and isotropic tissue. See section 5.2 for a discussion.*

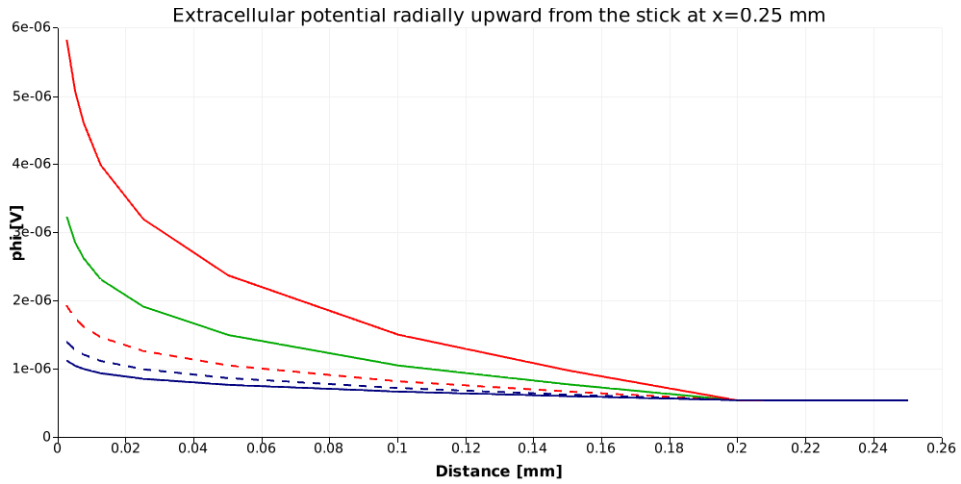

Figure 5.9: ***Extracellular potential for various conductivity values.*** *The curves show line plots of the extracellular potential perpendicularly upward from the stick, along the red line sketched in figure 5.4. The conductivity in the parts labeled by* 0.03 *S/m in figure 5.4 was gradually increased, in order to investigate the dependence of the potential on the conductivity, while the conductivity of the other areas was fixed. Red solid line:* 0.03 *S/m. Green solid line:* 0.06 *S/m. Red dashed line:* 0.12 *S/m. Blue dashed line:* 0.2 *S/m. Blue solid line:* 0.3 *S/m. The last case corresponds to a homogeneous tissue. It is clear from the curves that the rate of change of the extracellular potential decreases with increasing conductivity, as expected from Ohm's law.*

5.3 Two-Monopole: Effect of Boundary Conditions and Conductivity Profile

The impact of the experimental set-up and tissue conductivity profile on the MEA potentials was investigated, using a two-monopole model to represent the neuron. The solution for an infinite medium of constant conductivity, σ, will in this section be referred to as ϕ_∞. In the case of balanced monopolar sources, it becomes

$$\phi_\infty(r) = \frac{1}{4\pi\sigma_{\text{tissue}}}\left(\frac{I}{r_1} - \frac{I}{r_2}\right), \tag{5.1}$$

where r_1 and r_2 are the distance from the field point r to the source and the sink, respectively. Similar expressions, involving sums over point sources or line sources, have been used to evaluate the extracellular potential from transmembrane currents in many previous studies [20, 27, 35, 43].

The weak form for the extracellular potential around a two-monopole model is

$$\int_\Omega \sigma\nabla u\cdot\nabla v\mathrm{d}x = \int_{\partial\Omega_7} J_{\mathrm{m}} v\mathrm{d}s - \int_{\Omega_8} J_{\mathrm{m}} v\mathrm{d}s.$$

The monopoles were modeled as spheres of diameter 10 μm, and the straight line between the source and the sink was aligned with the x-axis and centered on the y-axis.

Since the model is frequency-independent for a given monopole separation, direct current was assumed, i.e., J_{m} is a constant. The current value used was $I = 1$ nA.

The parameter-dependence of the MEA potentials was investigated in a stepwise manner. First, in order to study the impact of a finite medium, i.e., the insulating boundary conditions and reference electrode, $\phi_\infty(r)$ was compared to the FEM model. The latter incorporated the finite medium but neglected saline by assuming the whole interior domain to consist of tissue, characterized by σ_{tissue}. The resulting potentials on a $(2\text{ mm}) \times (2\text{ mm})$ square centered on the glass substrate are shown in figure 5.10. This corresponds to the area covered by the tissue in the later simulation where saline is taken into account. Comparison of the finite medium potentials (upper plot) to the infinite medium potentials (middle row) shows that the boundary conditions result in more than a doubling of the potentials on the MEA. The absolute values around the current sink (blue area in upper plot) are lower than those surrounding the source. This may be caused by the reference electrode being closer to the sink.

Next, the impact of saline surrounding the tissue was investigated by comparing the finite medium solution described in the last paragraph to the solution for a slice with conductivity 0.3 S/m surrounded by saline of conductivity 3.0 S/m. Figure 5.11 shows the resulting plots. Interestingly, the presence of saline reduces the potentials back towards ϕ_∞, as seen by comparing the top image of figure 5.11 to the middle image of figure 5.10. There are two opposite effects: The insulating boundary conditions draw the potential on the MEA up, whereas the presence of saline decreases it again. Since the conductivity of saline is 10 times that of tissue, and volume current is conserved, the potential is almost constant in saline. Hence, it serves like a grounded medium surrounding the slice, thereby causing a steeper potential reduction from the neuron out to the interface.

Finally, the impact of moderate inhomogeneities and anisotropies, with the values given in table 4.2 on page 46, was investigated. The potentials calculated were compared to those described in the last paragraph, which assumed a constant tissue conductivity 0.3 S/m in all directions. Saline ($\sigma_{\mathrm{saline}} = 3.0$ S/m) was taken into account in both models. A comparison hence indicates how the exact conductivity profile influences the potentials recorded on the MEA. The top image of figure 5.12 shows the plot for an inhomogeneous and anisotropic tissue, while the middle image is identical to the upper row of figure 5.11, i.e., $\sigma_{\mathrm{tissue}} = 0.3$ S/m. It is evident that the moderate differences between the conductivity profiles do not have a large impact on the potentials. The bottom plot in figure 5.12 shows the difference between the solutions. It confirms that the effect of anisotropy is to make the potentials more aligned with the two-monopole axis, a feature which is hard to see from the top plot.

To better display the shape of the potentials, line plots were taken along the glass substrate (MEA). Figure 5.13 shows the potential along a line parallel to and lying directly underneath the two-monopole axis. The insulating boundaries are again seen to cause a great increase of both the crest and trough of the numerically

simulated potential (red curve), compared to ϕ_∞ (blue curve). As indicated in figure 5.12, the exact conductivity profile of the tissue has a very moderate impact (green and purple curves). When the saline is modeled, the slice is situated between 1 mm and 3 mm along the axis. This is seen to produce a nearly constant potential underneath the saline, as shown by the green and purple curve being nearly flat in this region. In the absence of saline, the potential is smoothly changing along the whole axis.

Figure 5.14 shows a similar plot along a line on the glass substrate, parallel to the two-monopole axis but shifted aside by 0.4 mm. Here, the inhomogeneities and anisotropies found by Goto et al. [22] are seen to have no impact on the potential: The green and the purple curve overlap. Interesting to notice is the finite medium solution (red curve), whose crest has a much large magnitude than its trough. This is likely due to the presence of the reference electrode, which is closer to the sink than to the source. Figures 5.15 and 5.16 show line plots perpendicular to the two-monopole axis, still on the glass substrate and crossing underneath the current source and the current sink, respectively. The impression given by the other plots in this section is confirmed.

5.4 Ball-and-Stick: Effect of Stimulation and Conductivity Profile

This section describes investigations into how the conductivity profile of the tissue affects the resulting potentials on the MEA. A ball-and-stick neuron with somatic current stimulus was modeled. Subthreshold phenomena are still considered, so a passive membrane has been assumed. The model has an intrinsic frequency filtering through the complex root,

$$\mathbf{s} = \sqrt{1 + j\omega\tau_{\mathrm{m}}},$$

arising in the expressions for the membrane currents of the ball and the stick. It is of interest to see how this effect shows up in the MEA potentials. By comparison with experimental data from somatically stimulated cells, the correctness of the model can be tested. The ball-and-stick parameters of table 4.3 on page 49 have been used, and the electrode current was

$$I_{\mathrm{e}}(t) = (250\ \mathrm{pA})\cos(2\pi f t).$$

Figures 5.17-5.20 show cross sections of the tissue and saline, with a ball-and-stick neuron situated close to the bottom. The soma (ball) is on the left end of the stick. In these plots, the tissue is assumed to have a constant conductivity, $\sigma_{\mathrm{tissue}} = 0.3$ S/m. As usual, the conductivity of saline is set to $\sigma_{\mathrm{saline}} = 3.0$ S/m.

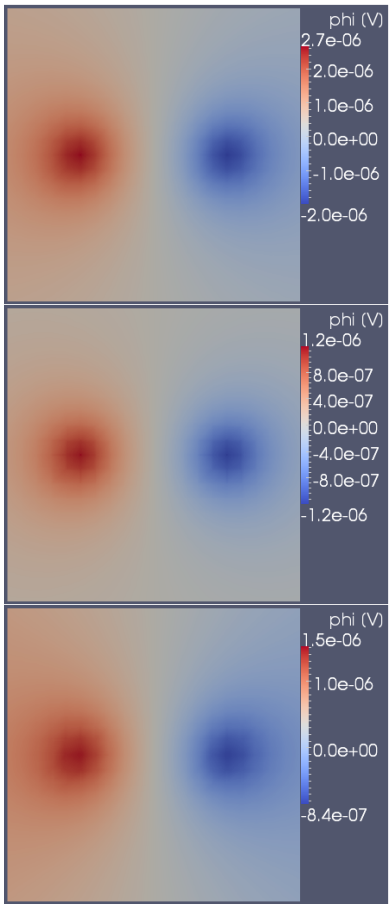

Figure 5.10: ***Impact of boundary conditions.*** *Potentials are shown on a* $(2\ mm) \times (2\ mm)$ *square at the center bottom of the domain, i.e., where the MEA recordings from the tissue are done. The figure in the upper row shows the FEM-computed MEA potentials assuming the whole medium to have conductivity* $\sigma = 0.3$ *S/m. In the second row, the potentials are calculated analytically assuming an infinite medium also with* $\sigma = 0.3$ *S/m. The figure in the bottom row shows the difference, i.e., the FEM potentials minus the analytically calculated potentials. Comparing the top and middle images reveals that the potentials obtained by the different assumptions differ by more than a factor two. This suggests that the boundary conditions of the experimental set-up, i.e., the surrounding glass, air, and MEA plate, have a significant impact on the extracellular potentials. See section 5.3 for a discussion.*

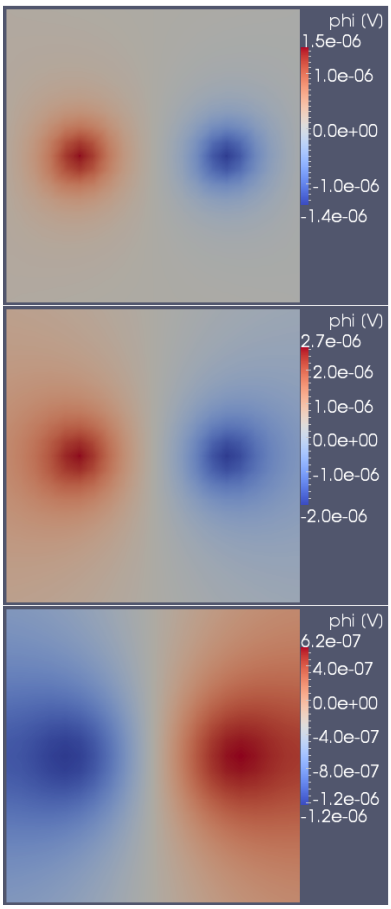

Figure 5.11: ***Impact of saline.*** *Potentials are shown on a* $(2\,mm) \times (2\,mm)$ *square at the center bottom of the domain, i.e., where the MEA recordings from the tissue are done. The figure in the upper row shows the FEM-computed MEA potentials assuming a slice of extent* $(2\,mm) \times (2\,mm) \times (0.3\,mm)$ *is placed upon the MEA and surrounded by saline. The tissue is assumed to have conductivity* $\sigma = 0.3$ *S/m, while the saline has* $\sigma = 3.0$ *S/m. The image in the second row is identical to the upper row of figure 5.10. It shows the potentials calculated with the FEM, assuming the whole domain to consist of tissue of conductivity* $\sigma = 0.3$ *S/m. The bottom image shows the difference between the two. It is clear from the plots that the presence of saline around the tissue significantly reduces the potentials on the MEA. See section 5.3 for a discussion.*

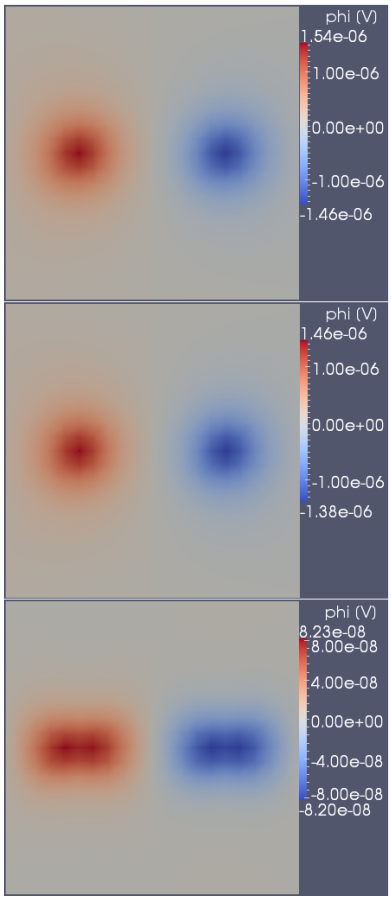

Figure 5.12: ***Impact of anisotropic and inhomogeneous conductivity profile.*** *Potentials are shown on a* $(2\ mm) \times (2\ mm)$ *square at the center bottom of the domain, i.e., where the MEA recordings from the tissue are done. The image in the upper row shows the potentials calculated by assuming an inhomogenous and anisotropic tissue with conductivity tensors given by the values in table 4.2 on page 46. The middle row is identical to the top image of figure 5.11. It shows the potentials calculated assuming a homogeneous tissue with conductivity* $\sigma = 0.3$ *S/m. For both simulations, the surrounding saline was assumed to have a constant conductivity* $\sigma = 3.0$ *S/m. The bottom image shows the difference between the two. It is clear that any errors arising from modest degrees of anisotropy and inhomogeneity in the tissue are small. See section 5.3 for a discussion.*

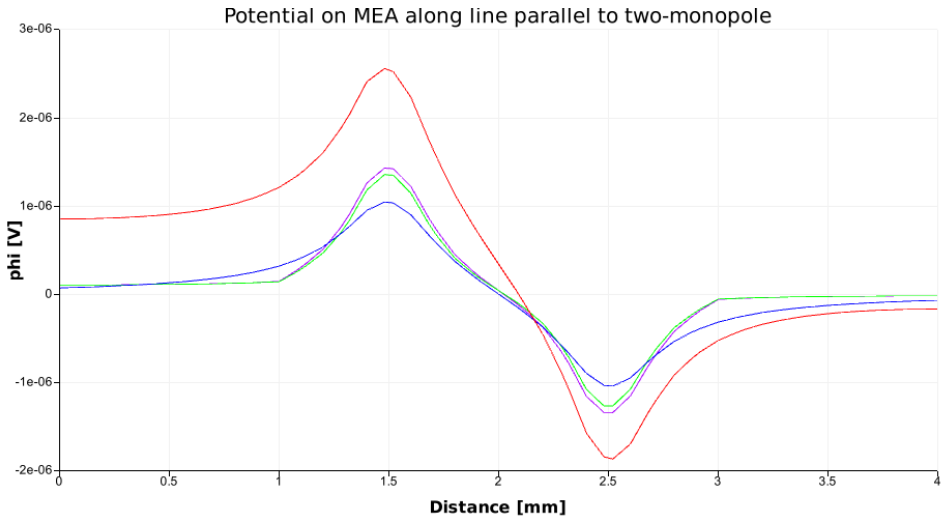

Figure 5.13: ***Impact of boundaries and conductivity.*** *Line plots along the bottom plate, parallel to and right below the two-monopole axis. The potential along the whole glass substrate is shown (figure 2.10), and the tissue lies between* 1 *mm and* 3 *mm on the horizontal axis. Blue line: potential, assuming an infinite medium with conductivity* 0.3 *S/m, i.e., equation 5.1 on page 65. Red line: potential computed, assuming a finite medium consisting only of tissue with conductivity* 0.3 *S/m. Green line: potential computed, assuming a tissue slice of conductivity* 0.3 *S/m is embedded in saline of conductivity* 3.0 *S/m, i.e., similar to the real experimental set-up. Purple line: potential computed, assuming a tissue slice with anisotropic and inhomogeneous conductivity values given by table 4.2 on page 46 surrounded by saline of conductivity* 3.0 *S/m. The curves reveal that the insulating boundary conditions greatly increase the potential. Taking the presence of saline into account, then reduces it again. Small inhomogeneities and anisotropies do not seem to have a strong impact. See section 5.3 for a discussion.*

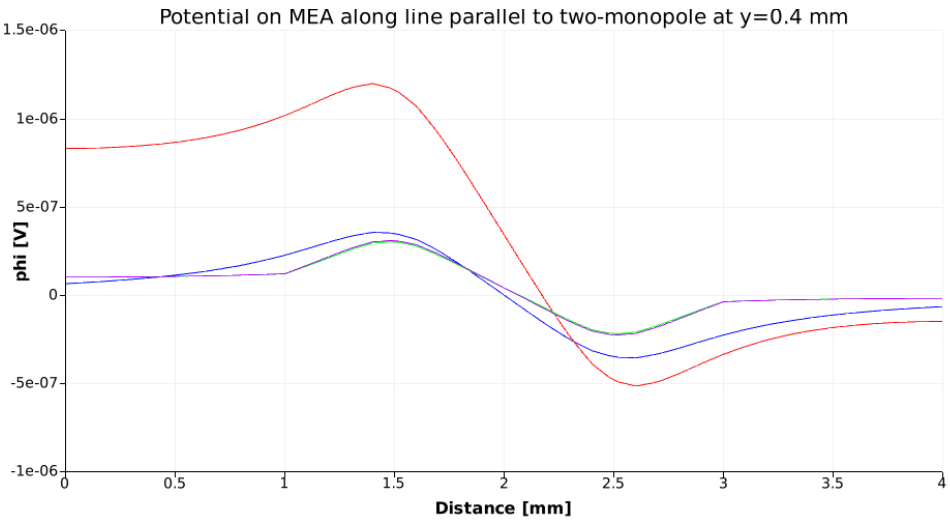

Figure 5.14: ***Impact of boundaries and conductivity.*** *Line plots along the bottom plate, parallel to the two-monopole axis but* 0.4 *mm aside from it. The potential along the whole glass substrate is shown (figure 2.10), and the tissue lies between* 1 *mm and* 3 *mm on the horizontal axis. Blue line: potential, assuming an infinite medium with conductivity* 0.3 *S/m, i.e., equation 5.1 on page 65. Red line: potential computed, assuming a finite medium consisting only of tissue with conductivity* 0.3 *S/m. Green line: potential computed, assuming a tissue slice of conductivity* 0.3 *S/m is embedded in saline of conductivity* 3.0 *S/m, i.e., similar to the real experimental set-up. Purple line: potential computed, assuming a tissue slice with anisotropic and inhomogeneous conductivity values given by table 4.2 on page 46 surrounded by saline of conductivity* 3.0 *S/m. The curves reveal that the finiteness of the medium and the insulating boundary conditions greatly increase the potential, whereas taking the presence of saline into account then reduces it again. Small inhomogeneities and anisotropies do not seem to have a strong impact on the potentials. Interestingly, in this figure, for the potential with a finite medium consisting only of tissue (red curve), the crest under the source-monopole has a magnitude more than twice as big as the trough under the sink. This is supposedly due to the sink being closer to the reference electrode. See section 5.3 for a discussion.*

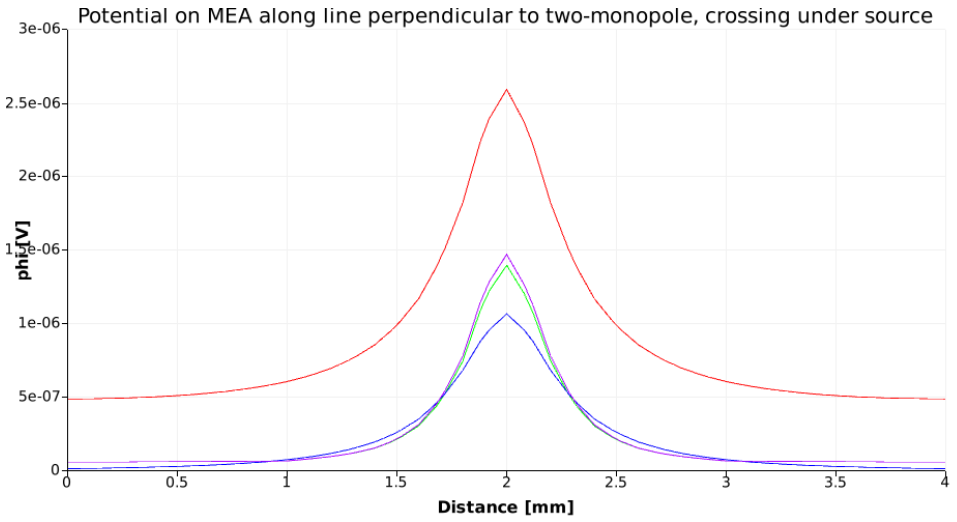

Figure 5.15: ***Impact of boundaries and conductivity.*** *Line plots along the bottom plate, perpendicular to the two-monopole axis and crossing underneath the current source. The potential along the whole glass substrate is shown (figure 2.10), and the tissue lies between* 1 *mm and* 3 *mm. Blue line: potential, assuming an infinite medium with conductivity* 0.3 *S/m, i.e., equation 5.1 on page 65. Red line: potential computed, assuming a finite medium consisting only of tissue with conductivity* 0.3 *S/m. Green line: potential computed, assuming a tissue slice of conductivity* 0.3 *S/m is embedded in saline of conductivity* 3.0 *S/m, i.e., similar to the real experimental set-up. Purple line: potential computed, assuming a tissue slice with anisotropic and inhomogeneous conductivity values given by table 4.2 on page 46 surrounded by saline of conductivity* 3.0 *S/m. Again, the insulating boundary conditions greatly increase the potential (red line) compared to the infinite medium solution (blue line). Taking the presence of saline into account reduces the potentials again. See section 5.3 for a discussion.*

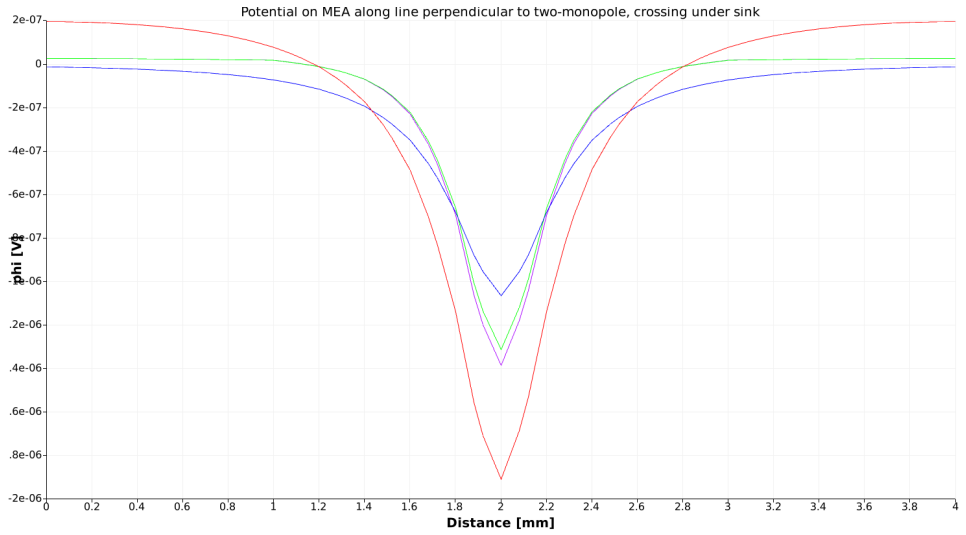

Figure 5.16: ***Impact of boundaries and conductivity.*** *Line plots along the bottom plate, perpendicular to the two-monopole axis and crossing underneath the current sink. The potential along the whole glass substrate is shown (figure 2.10), and the tissue lies between* 1 *mm and* 3 *mm. Blue line: potential, assuming an infinite medium with conductivity* 0.3 *S/m, i.e., equation 5.1 on page 65. Red line: potential computed, assuming a finite medium consisting only of tissue with conductivity* 0.3 *S/m. Green line: potential computed, assuming a tissue slice of conductivity* 0.3 *S/m is embedded in saline of conductivity* 3.0 *S/m, i.e., similar to the real experimental set-up. Purple line: potential computed, assuming a tissue slice with anisotropic and inhomogeneous conductivity values given by table 4.2 on page 46 surrounded by saline of conductivity* 3.0 *S/m. Again, the insulating boundary conditions greatly increase the potential (red line) compared to the infinite medium solution (blue line). Taking the presence of saline into account reduces the potentials again. See section 5.3 for a discussion.*

In figure 5.17 the stimulus frequency is 1 Hz. Here, the membrane currents resulting from the stimulus, manifested through the extracellular potential gradient alongside of the stick, are seen to leave close to uniformly. The exceptions are the zero phase moments of the electrode current, i.e., at $t = T/4$ and $t = 3T/4$. In these cases, the membrane currents must sum to zero. At $t = T/4$, this is seen by membrane current entering the ball and the proximal end of the stick, while current is leaving the distal end of the stick. At $t = 3T/4$, the currents have the opposite direction.

The extracellular potentials arising from stimulus current of frequency 10 Hz are shown in figure 5.18. No clear difference from the 1 Hz case displays. However, at $t = 3T/8$ and $t = 7T/8$ a slight low-pass filtering effect is seen in comparison with figure 5.17. Here, the rate of change of the potential along the distal end of the stick is smaller. Since the volume current is $\mathbf{J} = -\sigma\nabla\phi$, a smaller potential gradient means less current in the tissue. Since this current has to come from the membrane, a lower rate of change of the potential close to the stick indicates that its membrane currents are reduced. Figure 5.19 shows the potentials for 100 Hz stimulus, revealing a distribution of membrane currents clearly different from those of the two lower frequencies considered. The currents leave the cell much closer to the soma at all time steps. At the zero phase of the stimulus, still $t = T/4$ and $t = 3T/4$, the bulk of the stick currents crosses the membrane closer to the soma than for 1 Hz and 10 Hz.

Since there is no net current at the zero phase, the *monopole moment* of the ball-and-stick model is zero. The first nonzero multipole moment generated by the membrane currents is a *dipole* [23]. When the currents crossing the stick are drawn towards soma for increasing frequency, the dipole length decreases. The trend continues in figure 5.20, which shows the same plots for $f = 1$ kHz. Here, the membrane currents are centered close to the soma. The zero phase currents are also seen to get even closer.

These figure verify that the ball-and-stick model implemented here reproduces the low-pass filtering properties shown by, e.g., references [35, 43]. This serves to confirm that the FEniCS programs work correctly, while also being an instructive way to see how the MEA potentials are generated. Special care should be taken when interpreting the potentials at the zero phase of the electrode current. Although it behaves as expected in figures 5.17-5.20, large corrections of the stick current were needed to obey Kirchhoff's current law[2]. More detailed simulations with a finer mesh are needed before making any quantitative predictions of the dipole length.

The low-pass filtering effect is also seen in figures 5.21-5.24, which show the simulated potential on the MEA for stimulation frequencies $f = 1$ Hz, $f = 10$ Hz, $f = 100$ Hz, and $f = 1$ kHz. Both for 1 Hz and 10 Hz, the potential is spread out along the length of the stick. For the two higher frequencies it is more confined to the region of the MEA underneath the soma. The color scale of each plot is adjusted

[2]Cf. section 4.7 and table 4.4 on page 54.

to be between the minimum and maximum value in the view shown. Hence, it is apparent that the maximum potential in the 1 Hz case is 8.9×10^{-7} V while reaching $5/3$ times that value for 1 kHz, i.e., 1.5×10^{-6} V. A physical explanation for this is simply that at any time, independent of frequency, a net current of the same amount as supplied by the electrode has to leave the cell. For a patch of membrane close to the soma, the amount of current across it in general increases with frequency, thereby creating a larger potential difference.

Figures 5.25-5.28 provide the same view as just discussed but with inhomogeneous and anisotropic tissue conductivity. The conductivity profile, taken from the recent work by Goto et al. [22], is shown in table 4.2 on page 46. For each frequency, the overall shapes of the potentials are very similar to those calculated for isotropic and homogeneous tissue conductivity 0.3 S/m. Similar to the potentials for a two-monopole (figure 5.12), the minima and maxima[3] are seen to slightly increase with the inhomogeneous and anisotropic conductivity. However, the deviation is less than 10 % in all cases. Again, it does not seem like the exact conductivity profile is critical for the MEA potentials, as long as the values are close to the one assumed for the homogeneous and isotropic tissue, cf. the extreme cases of section 5.2.

By comparing the potentials along a line on the MEA for the two conductivity profiles considered, it is easier to get an impression of their impact. Figures 5.29-5.32 show plots on the glass substrate, saline part not included, along an axis straight below the ball-and-stick neuron. The wiggly appearance of some curves is just a numerical artifact.

In the low frequency case, current leaves the stick quite uniformly along its whole length. Away from the ends, the *volume current* then has a direction perpendicular to the stick axis; the directional derivative of the extracellular potential is close to zero along the stick and maximum radially outwards. The conductivity in the latter direction is between 0.228 S/m and 0.268 S/m, as opposed to 0.3 S/m in the homogeneous and isotropic tissue. Accordingly, the potential at a distance 0.1 mm from the neuron is expected to be higher, as confirmed by the plots.

With respect to the horizontal axes of the graphs, the ball is situated at $x = 0.49$ mm, and the stick ends at $x = 1.51$ mm. As figure 5.29 clearly shows, the potential for the inhomogeneous and anisotropic tissue (red line) is pronouncedly higher between around 0.6 mm and 1.4 mm. The exception is in the zero phase of I_{e}, when the two curves follow each other closely. A very similar picture is given by figure 5.30 for $f = 10$ Hz. For 100 Hz stimulation, shown in figure 5.31, the two solutions are closer to each other in the second and the last row. With $f = 1$ kHz (figure 5.32) they are very close except for a small part where the return current is concentrated.

[3] When discussing "magnitudes" of potentials, what is actually meant is of course the absolute value of the potential difference between the point considered and the reference electrode.

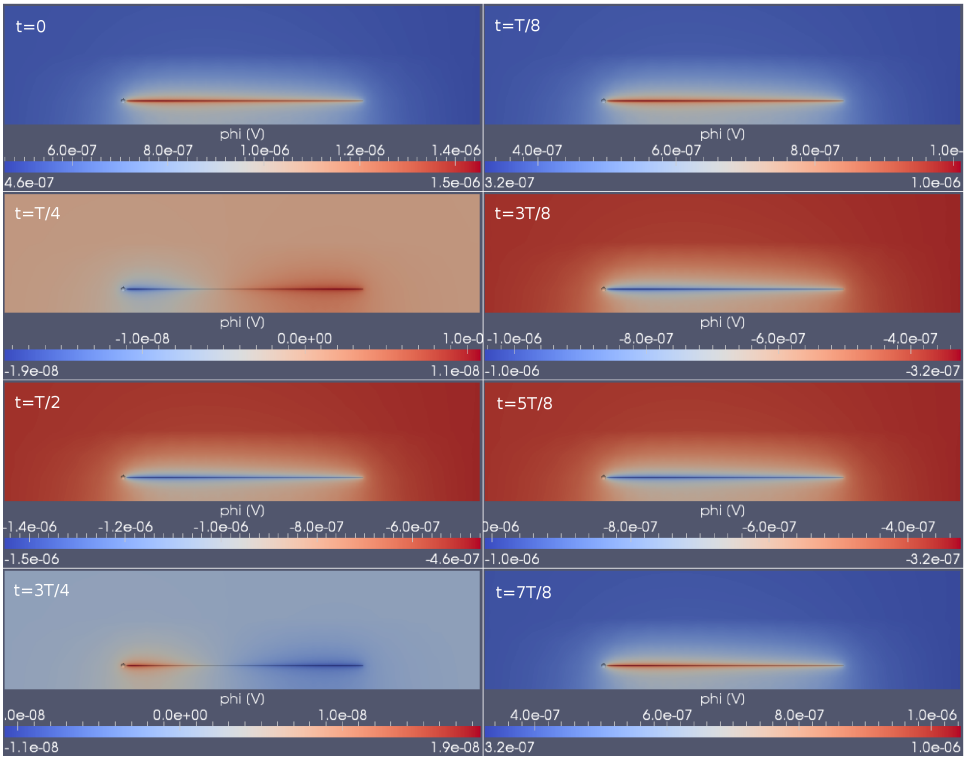

Figure 5.17: ***Ball-and-stick neuron with 1 Hz electrode current in the soma.*** *In the cross section shown, the upper 2/5 is saline, while the lower part is tissue. The figures show $t = 0, \ldots, 7T/8$, for a somatic electrode current, $I_e = (250\ pA)\cos(2\pi f t)$, where $T = 1/f = 1$ s. The extracellular potential reveals that the current leaves the stick quite uniformly at this frequency. See section 5.4 for a discussion.*

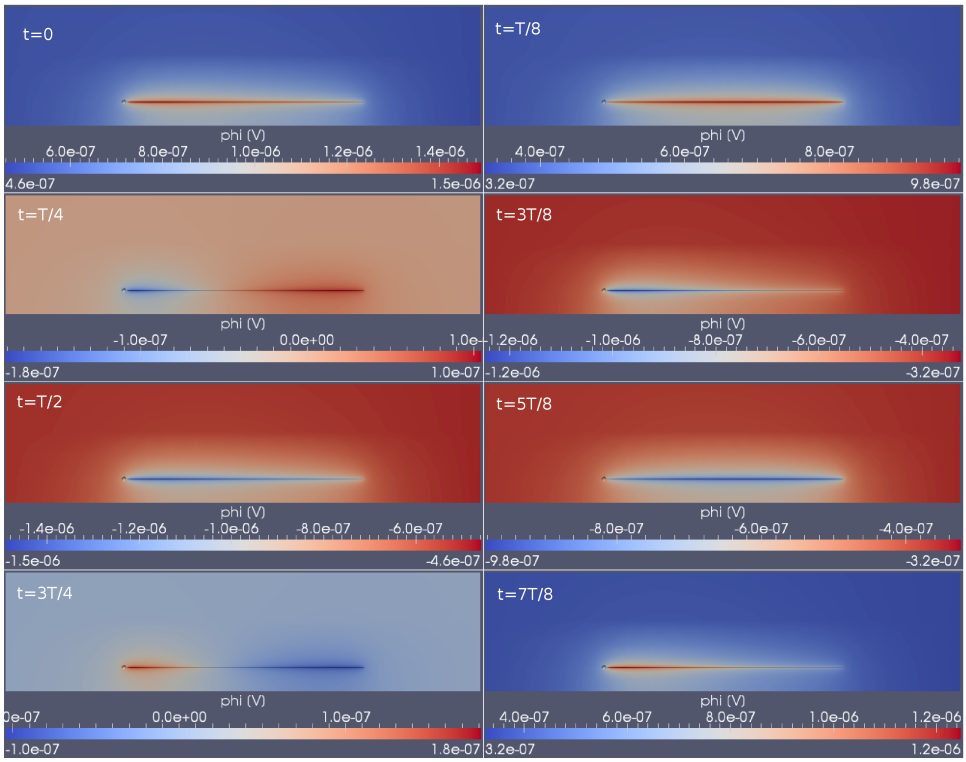

Figure 5.18: ***Ball-and-stick neuron with 10 Hz electrode current in the soma.*** *In the cross section shown, the upper 2/5 is saline, while the lower part is tissue. The figures show $t = 0, \ldots, 7T/8$, for a somatic electrode current, $I_e = (250\ pA)\cos(2\pi f t)$, where $T = 1/f = 0.1$ s. It is hard to notice any consistent low-pass filtering compared to figure 5.17, although at $t = 3T/8$ and $t = 7T/8$, the potential seems to decays faster along the stick in the 10 Hz case. See section 5.4 for a discussion.*

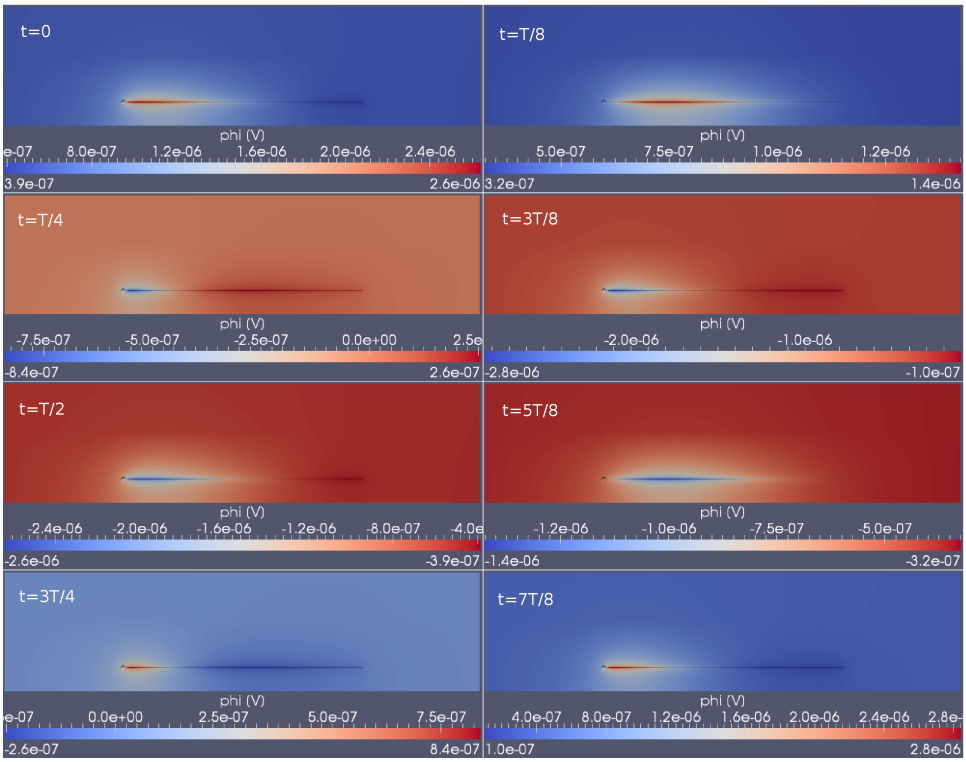

Figure 5.19: ***Ball-and-stick neuron with 100 Hz electrode current in the soma.*** *In the cross section shown, the upper 2/5 is saline, while the lower part is tissue. The figures show $t = 0, \ldots, 7T/8$, for a somatic electrode current, $I_e = (250\,pA)\cos(2\pi f t)$, where $T = 1/f = 10$ ms. Comparison to the cases with 1 Hz and 10 Hz stimulus frequency (figures 5.17 and 5.18, respectively), clearly shows a low-pass filtering effect. The rate of change of the potential, i.e., the volume current, is much larger close to the soma than on the distal parts of the stick. See section 5.4 for a discussion.*

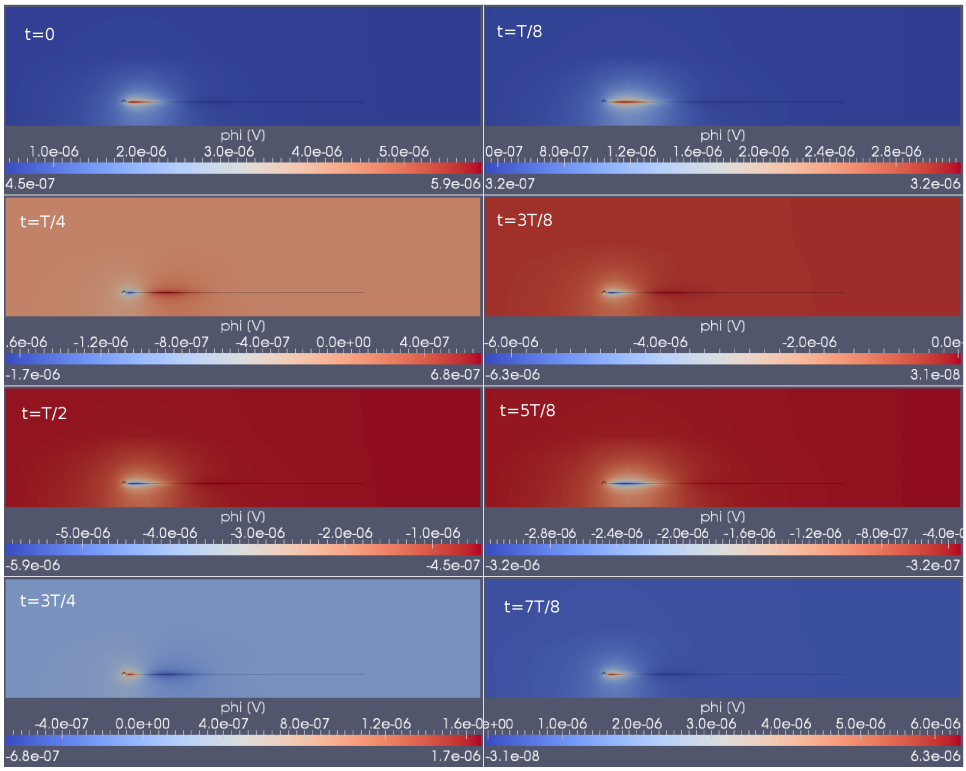

Figure 5.20: ***Ball-and-stick neuron with 1 kHz electrode current in the soma.*** *In the cross section shown, the upper 2/5 is saline, while the lower part is tissue. The figures show $t = 0, \ldots, 7T/8$, for a somatic electrode current, $I_e = (250\,pA)\cos(2\pi f t)$, where $T = 1/f = 1$ ms. A pronounced low-pass filtering effect is seen. The membrane currents are large at the soma and on the proximal parts of the stick, whereas the potential is close to constant along its proximal parts, indicating very weak membrane currents. See section 5.4 for a discussion.*

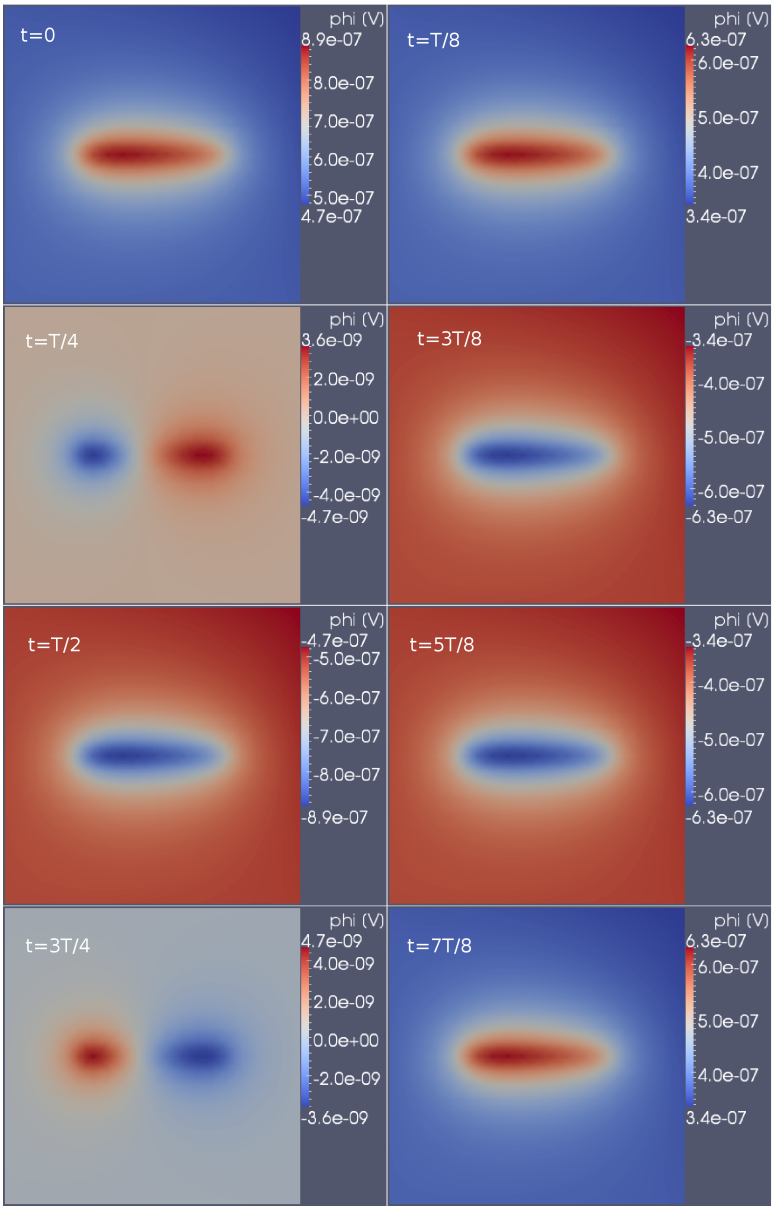

Figure 5.21: ***MEA potentials for ball-and-stick neuron with 1 Hz stimulus in homogeneous and isotropic tissue.*** *The potential on the glass substrate underneath the tissue slice is shown, i.e., a* $(2\ mm) \times (2\ mm)$ *square. The figures show* $t = 0, \ldots, 7T/8$*, for a somatic electrode current,* $I_e = (250\ pA)\cos(2\pi f t)$*, where* $T = 1/f = 1\ s$*. The potentials are spread out along the length of the stick, as in figure 5.17. See section 5.4 for a discussion.*

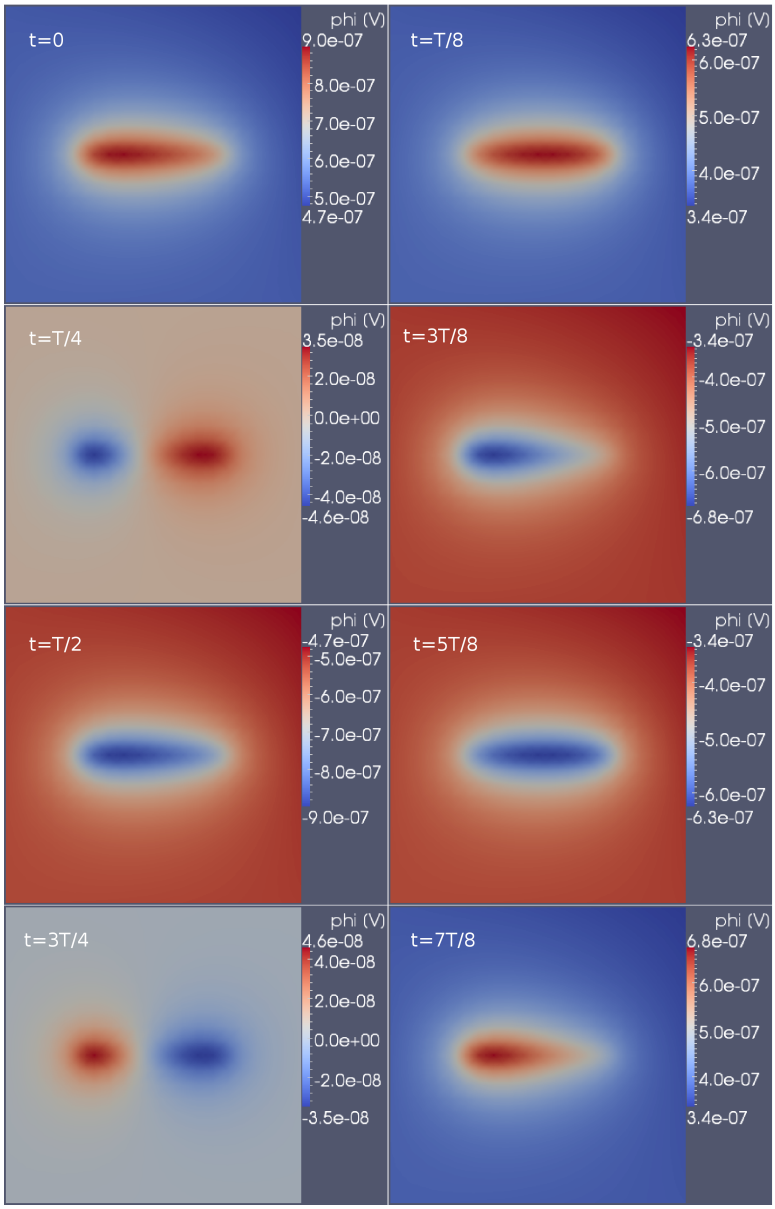

Figure 5.22: ***MEA potentials for ball-and-stick neuron with 10 Hz stimulus in homogeneous and isotropic tissue.*** *The potential on the glass substrate underneath the tissue slice is shown, i.e., a* $(2\ mm) \times (2\ mm)$ *square. The figures show* $t = 0, \ldots, 7T/8$*, for a somatic electrode current,* $I_e = (250\ pA)\cos(2\pi ft)$*, where* $T = 1/f = 0.1\ s$*. The potentials are very similar to those for* $f = 1\ Hz$*, but at* $t = 3T/8$ *and* $t = 7T/8$*, the potentials are more concentrated below the soma in this case when compared to the corresponding plots in figure 5.21. This is a manifestation of the same effect seen in the ball-and-stick view of figures 5.17 and 5.18. See section 5.4 for a discussion.*

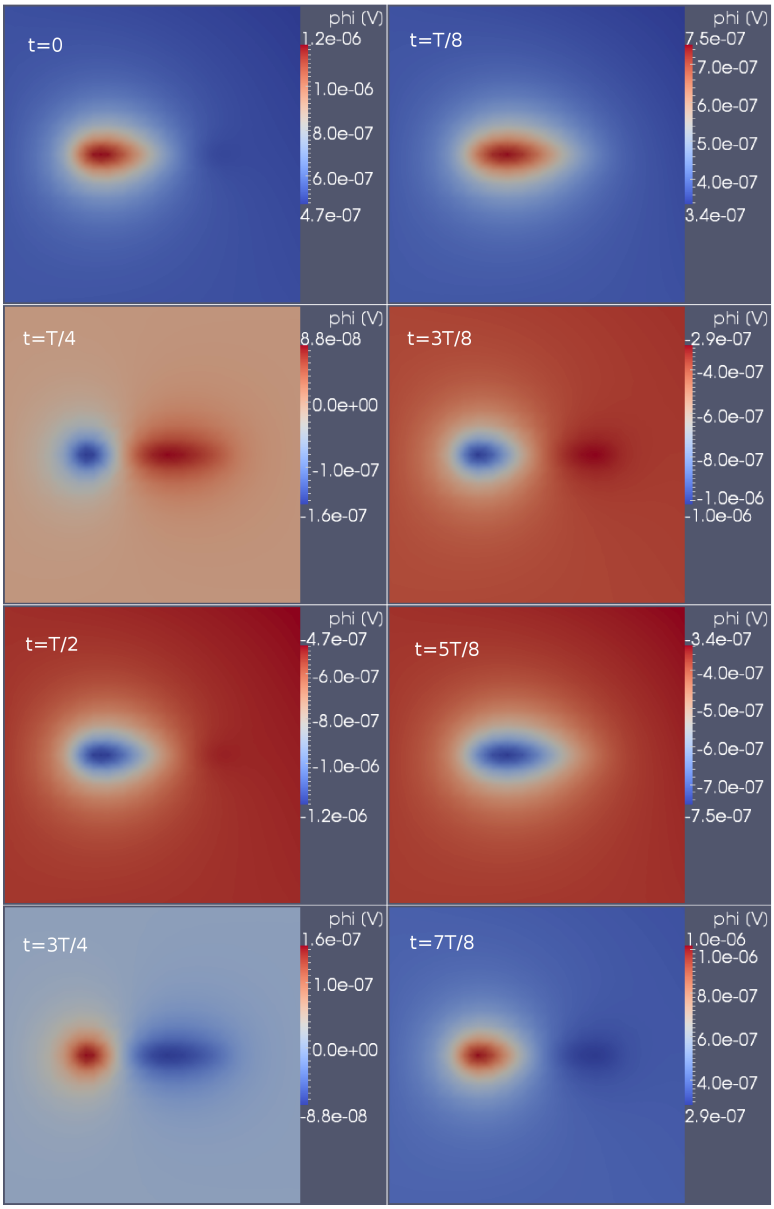

Figure 5.23: ***MEA potentials for ball-and-stick neuron with 100 Hz stimulus in homogeneous and isotropic tissue.*** *The potential on the glass substrate underneath the tissue slice is shown, i.e., a* $(2\ mm) \times (2\ mm)$ *square. The figures show* $t = 0, \ldots, 7T/8$*, for a somatic electrode current,* $I_e = (250\ pA)\cos(2\pi f t)$*, where* $T = 1/f = 10\ ms$*. The potentials are much more concentrated here than for the two lower frequencies considered. A decrease in the dipole length for the zero phase of the stimulus current (*$t = T/4$ *and* $t = 3T/4$*) can also be seen by comparison with the corresponding plots of figures 5.21 and 5.22, but the effect is quite modest. See section 5.4 for a discussion.*

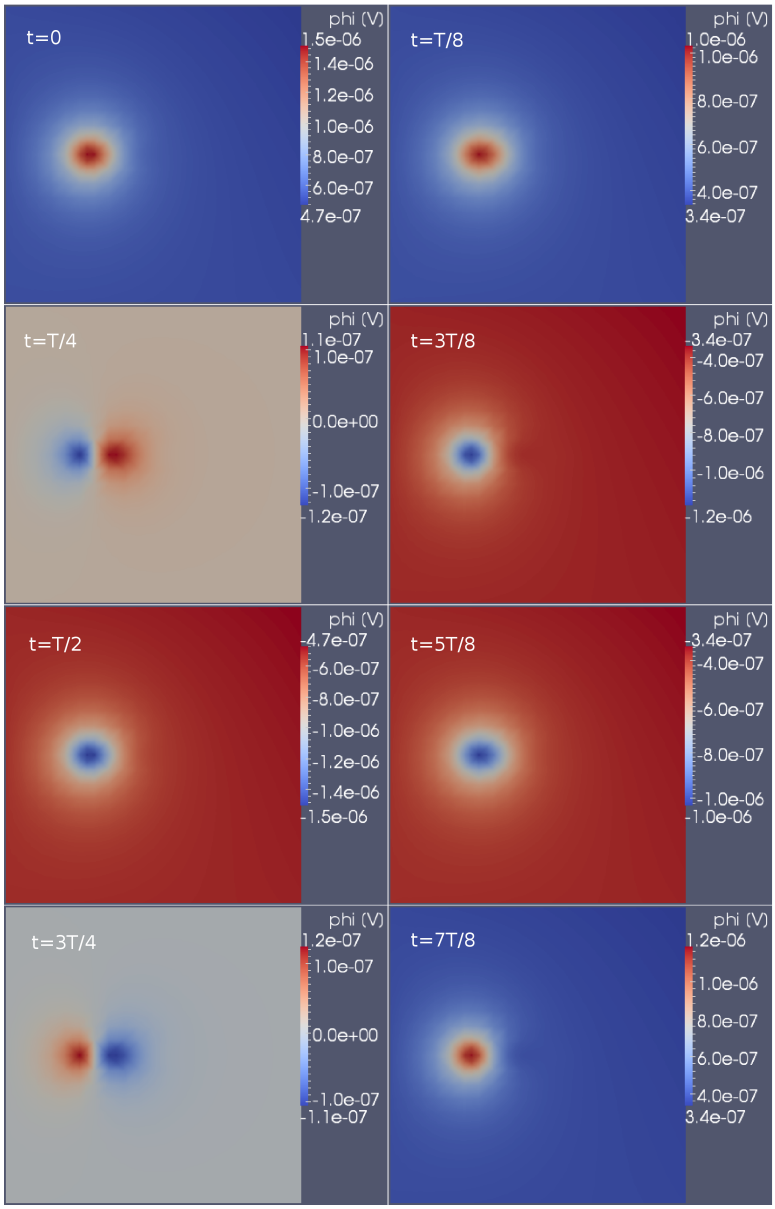

Figure 5.24: ***MEA potentials for ball-and-stick neuron with 1 kHz stimulus in homogeneous and isotropic tissue.*** *The potential on the glass substrate underneath the tissue slice is shown, i.e., a* $(2\ mm) \times (2\ mm)$ *square. The figures show* $t = 0, \ldots, 7T/8$*, for a somatic electrode current,* $I_e = (250\ pA)\cos(2\pi ft)$*, where* $T = 1/f = 1$ *ms. The potentials are very concentrated, as would be expected from figure 5.20, which shows that the bulk membrane current leaves near the soma. The dipole potentials in the zero phase, at* $t = T/4$ *and* $t = 3T/4$*, are also very close compared to the three lower frequencies considered. See section 5.4 for a discussion.*

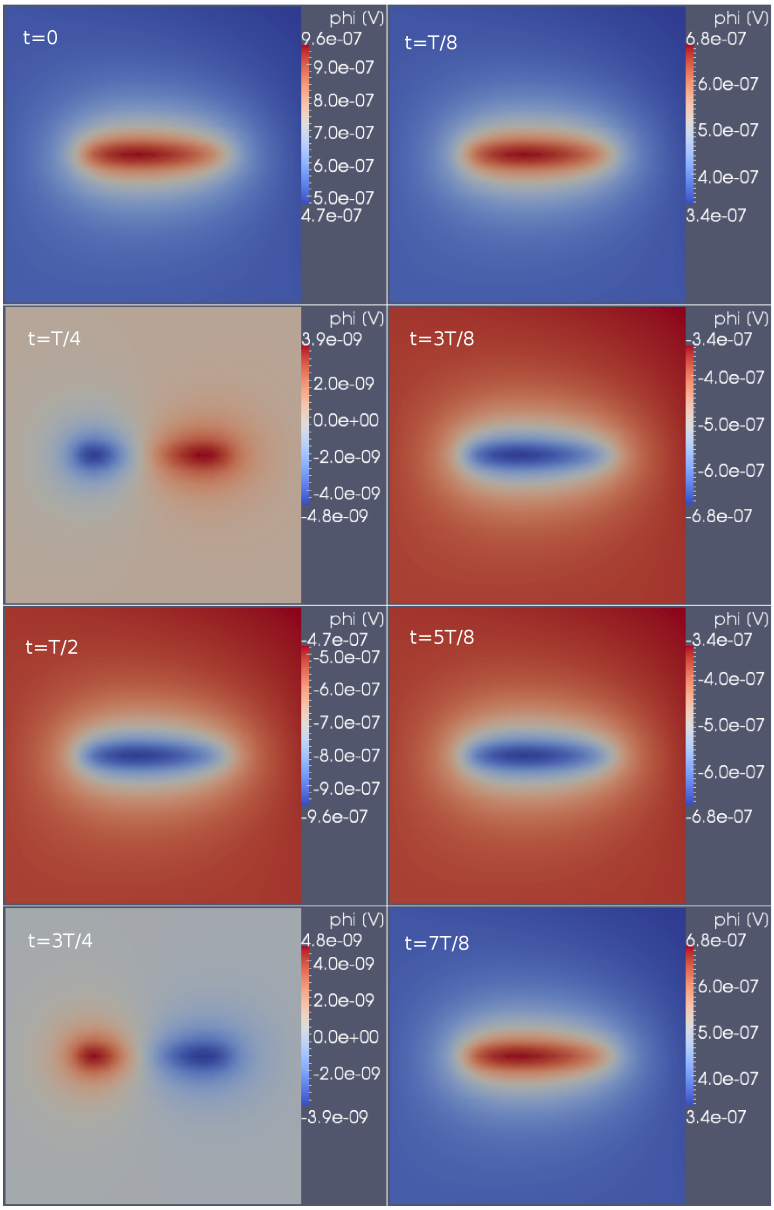

Figure 5.25: ***MEA potentials for ball-and-stick neuron with 1 Hz stimulus in inhomogeneous and anisotropic tissue.*** *The potential on the glass substrate underneath the tissue slice is shown, i.e., a* $(2\ mm) \times (2\ mm)$ *square. The figures show* $t = 0, \ldots, 7T/8$*, for a somatic electrode current,* $I_e = (250\ pA)\cos(2\pi f t)$*, where* $T = 1/f = 1\ s$*. See section 5.4 for a discussion.*

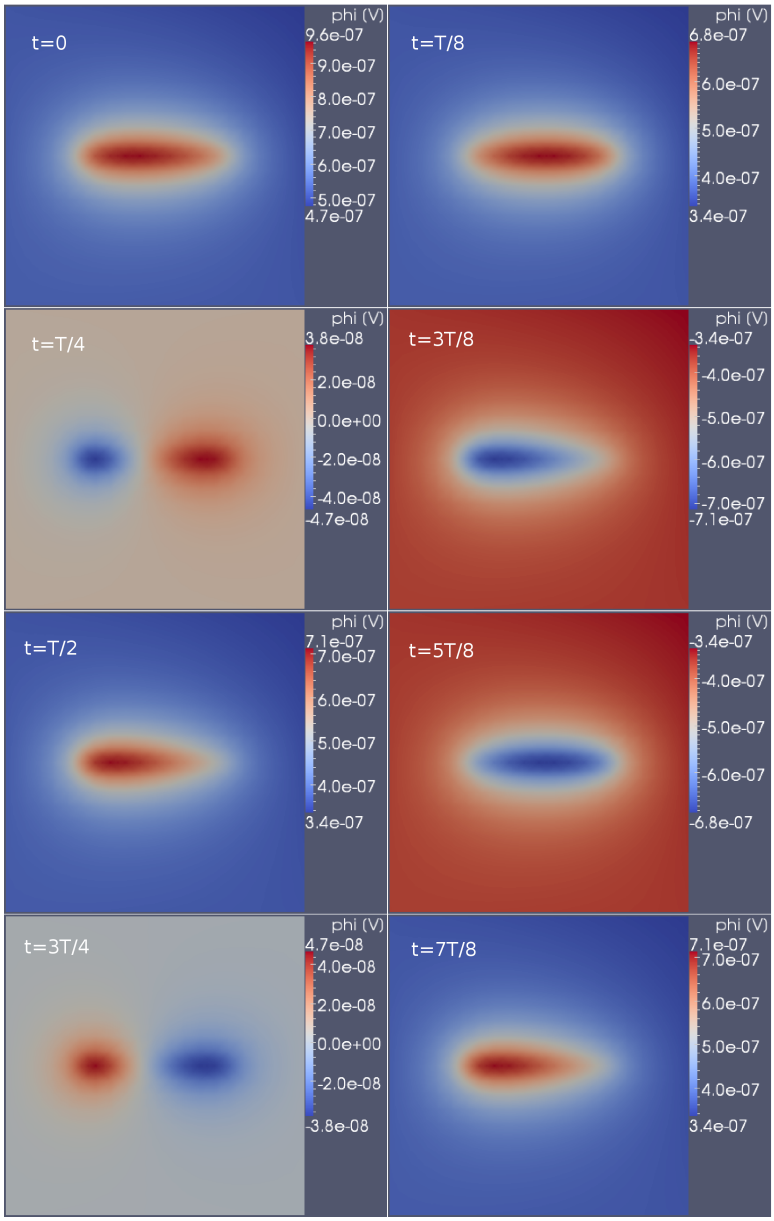

Figure 5.26: ***MEA potentials for ball-and-stick neuron with 10 Hz stimulus in inhomogeneous and anisotropic tissue.*** *The potential on the glass substrate underneath the tissue slice is shown, i.e., a* $(2\ mm) \times (2\ mm)$ *square. The figures show* $t = 0, \ldots, 7T/8$*, for a somatic electrode current,* $I_e = (250\ pA)\cos(2\pi ft)$*, where* $T = 1/f = 0.1\ s$*. See section 5.4 for a discussion.*

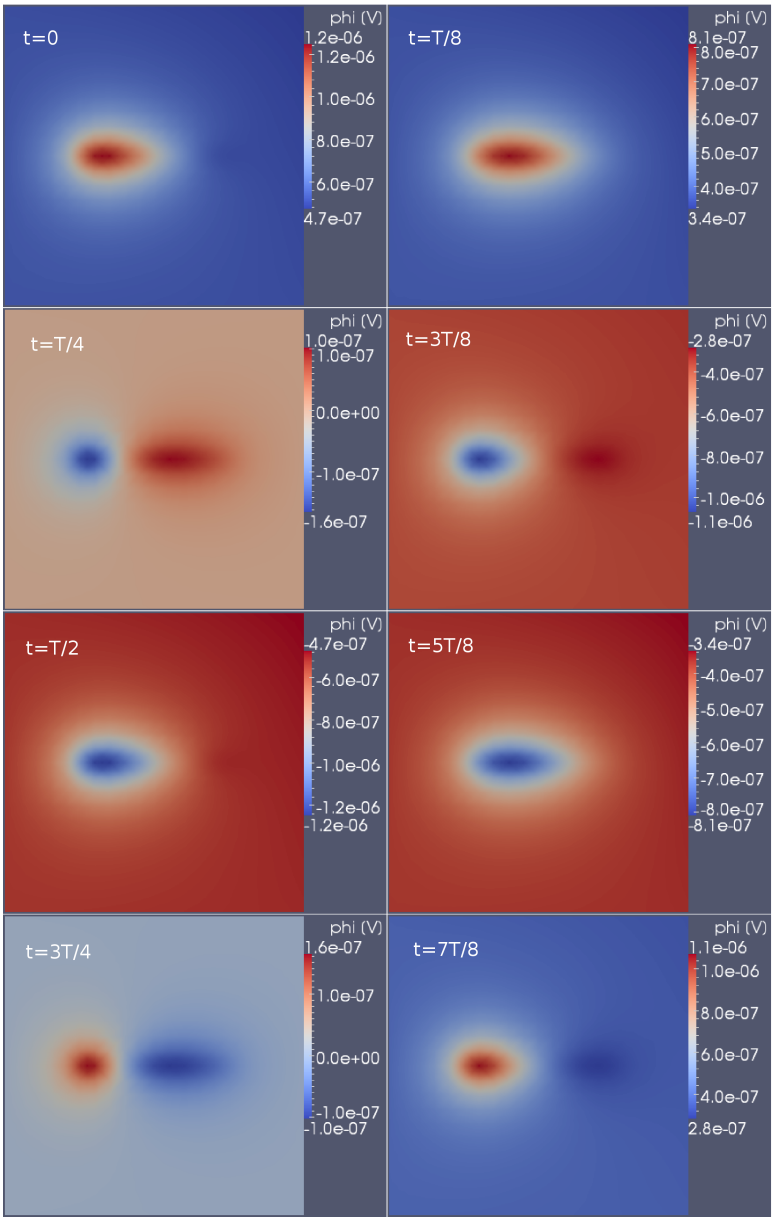

Figure 5.27: ***MEA potentials for ball-and-stick neuron with 100 Hz stimulus in inhomogeneous and anisotropic tissue.*** *The potential on the glass substrate underneath the tissue slice is shown, i.e., a* $(2\ mm) \times (2\ mm)$ *square. The figures show* $t = 0, \ldots, 7T/8$*, for a somatic electrode current,* $I_e = (250\ pA)\cos(2\pi f t)$*, where* $T = 1/f = 10\ ms$*. See section 5.4 for a discussion.*

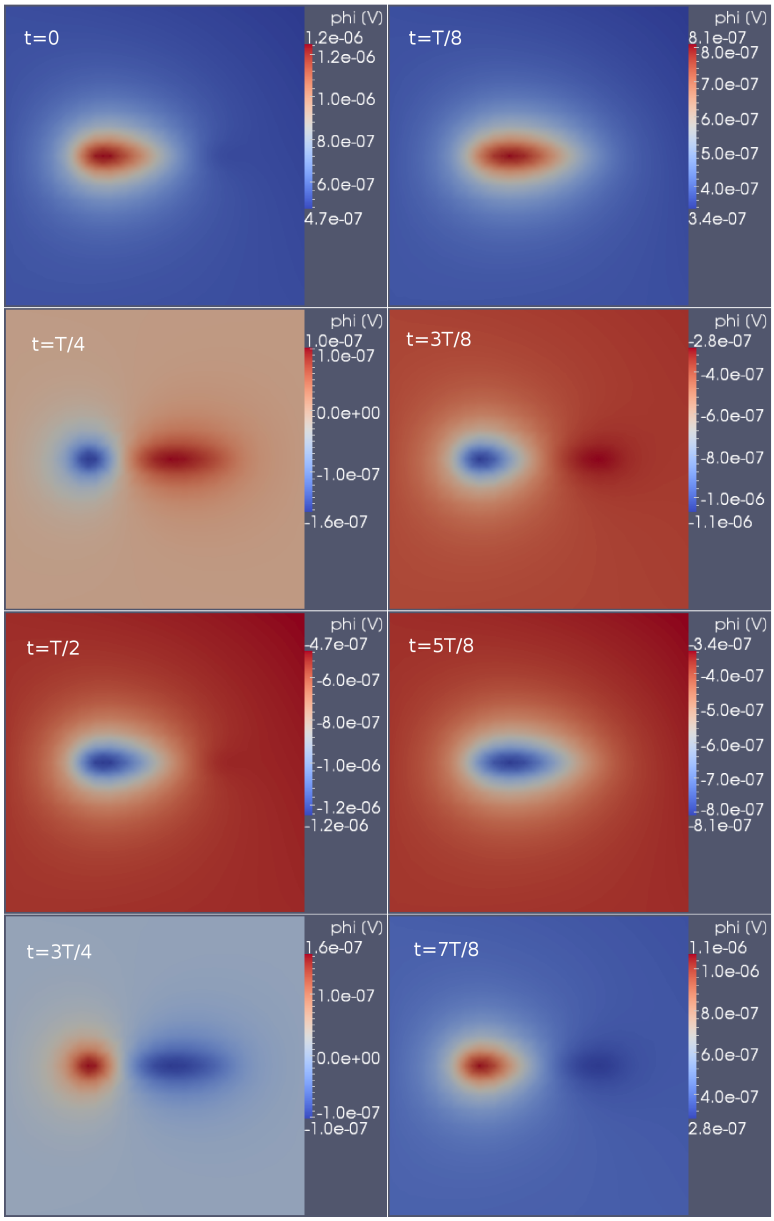

Figure 5.28: ***MEA potentials for ball-and-stick neuron with 1 kHz stimulus in inhomogeneous and anisotropic tissue.*** *The potential on the glass substrate underneath the tissue slice is shown, i.e., a* $(2\ mm) \times (2\ mm)$ *square. The figures show* $t = 0, \ldots, 7T/8$*, for a somatic electrode current,* $I_e = (250\ pA)\cos(2\pi ft)$*, where* $T = 1/f = 1\ ms$*. See section 5.4 for a discussion.*

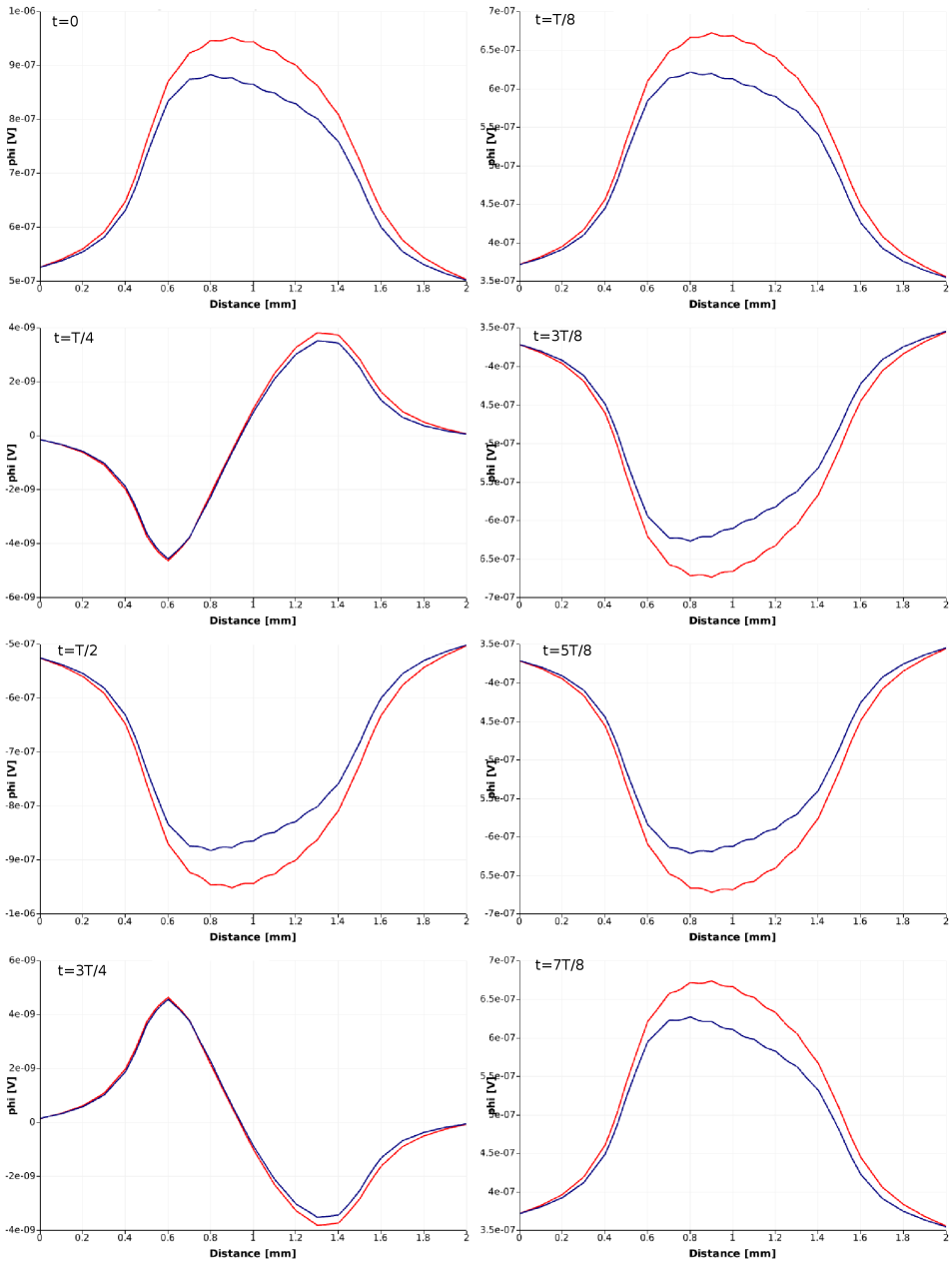

Figure 5.29: ***Impact of conductivity for ball-and-stick neuron with 1 Hz stimulus.*** *Line plots on the MEA along an axis parallel to the stick. The figures show $t = 0, \ldots, 7T/8$, for a somatic electrode current, $I_e = (250\,pA)\cos(2\pi ft)$, where $T = 1/f = 1$ s. Blue line: MEA potential for homogeneous and isotropic tissue with conductivity $\sigma_{\text{tissue}} = 0.3$ S/m. Red line: MEA potential for inhomogeneous and anisotropic tissue with conductivity values given in table 4.2 on page 46. Except for the zero phase of the stimulus, the potentials are seen to have a similiar shape, with an upward shift in the the case of anisotropic and inhomogeneous conductivity. See section 5.4 for a discussion.*

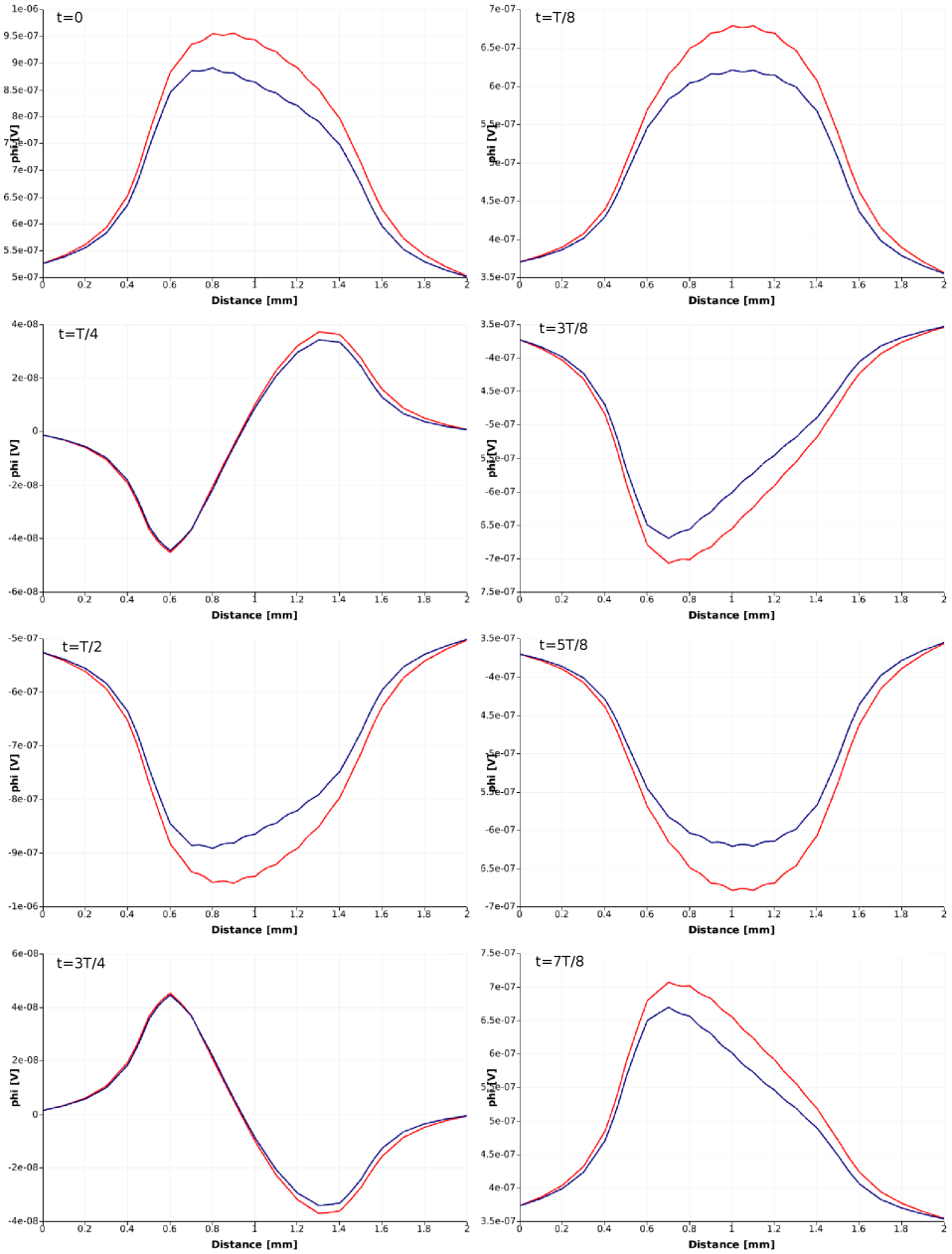

Figure 5.30: ***Impact of conductivity for ball-and-stick neuron with 10 Hz stimulus.*** *Line plots on the MEA along an axis parallel to the stick. The figures show $t = 0, \ldots, 7T/8$, for a somatic electrode current, $I_e = (250\,pA)\cos(2\pi ft)$, where $T = 1/f = 0.1$ s. Blue line: MEA potential for homogeneous and isotropic tissue with conductivity $\sigma_{\text{tissue}} = 0.3$ S/m. Red line: MEA potential for inhomogeneous and anisotropic tissue with conductivity values given in table 4.2 on page 46. A very similar picture to that for 1 Hz stimulation appears. An exception is the steeper decay of the potential at $t = 3T/8$ and $t = 7T/8$. See section 5.4 for a discussion.*

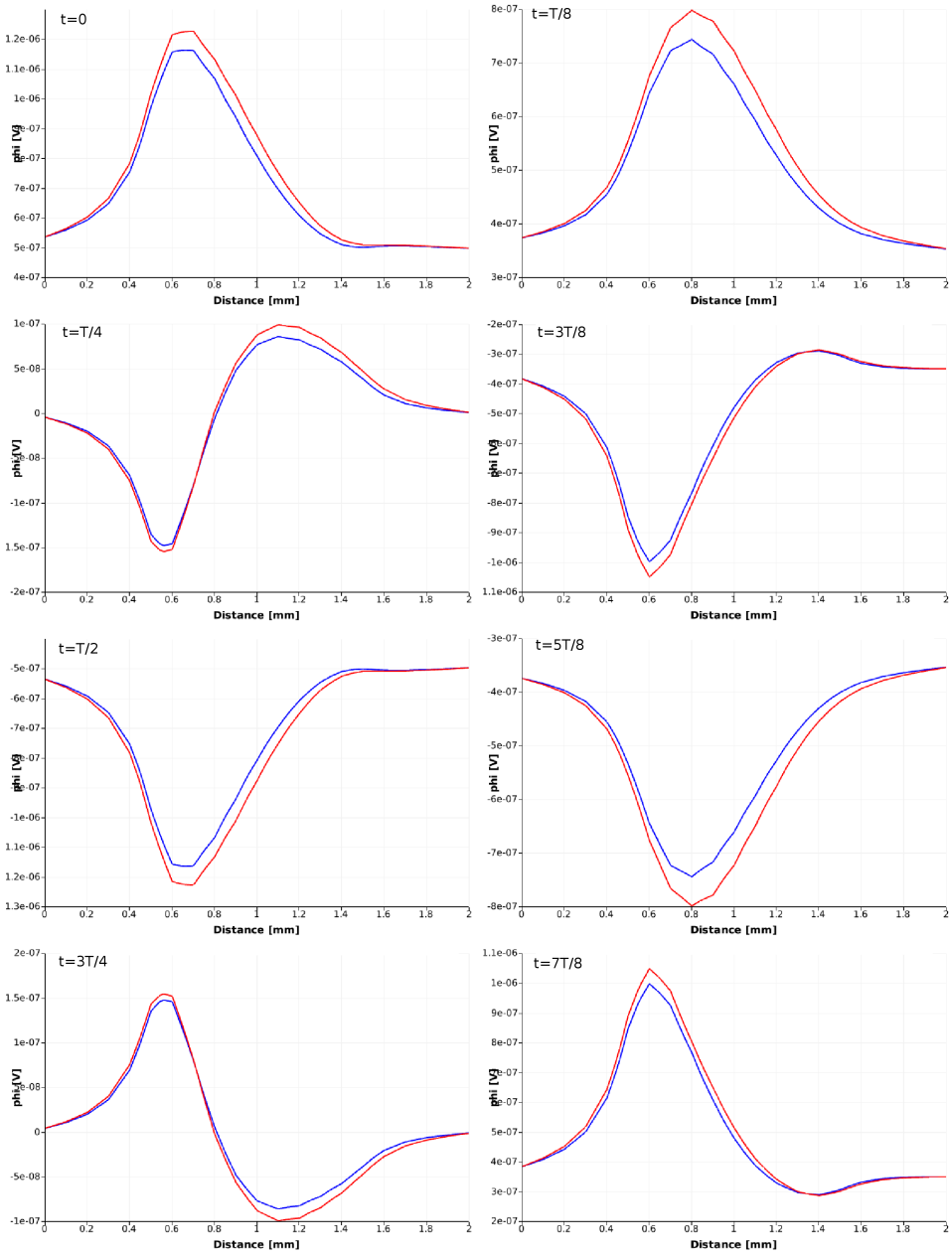

Figure 5.31: ***Impact of conductivity for ball-and-stick neuron with 100 Hz stimulus.*** *Line plots on the MEA along an axis parallel to the stick. The figures show* $t = 0, \ldots, 7T/8$*, for a somatic electrode current,* $I_{\mathrm{e}} = (250\ pA)\cos(2\pi ft)$*, where* $T = 1/f = 10$ *ms. Blue line: MEA potential for homogeneous and isotropic tissue with conductivity* $\sigma_{\mathrm{tissue}} = 0.3$ *S/m. Red line: MEA potential for inhomogeneous and anisotropic tissue with conductivity values given in table 4.2 on page 46. Since the potential peaks become narrower with higher frequency, the curves follow each other closely in a larger region compared to* 1 *Hz and* 10 *Hz stimulation (figures 5.29 and 5.30). See section 5.4 for a discussion.*

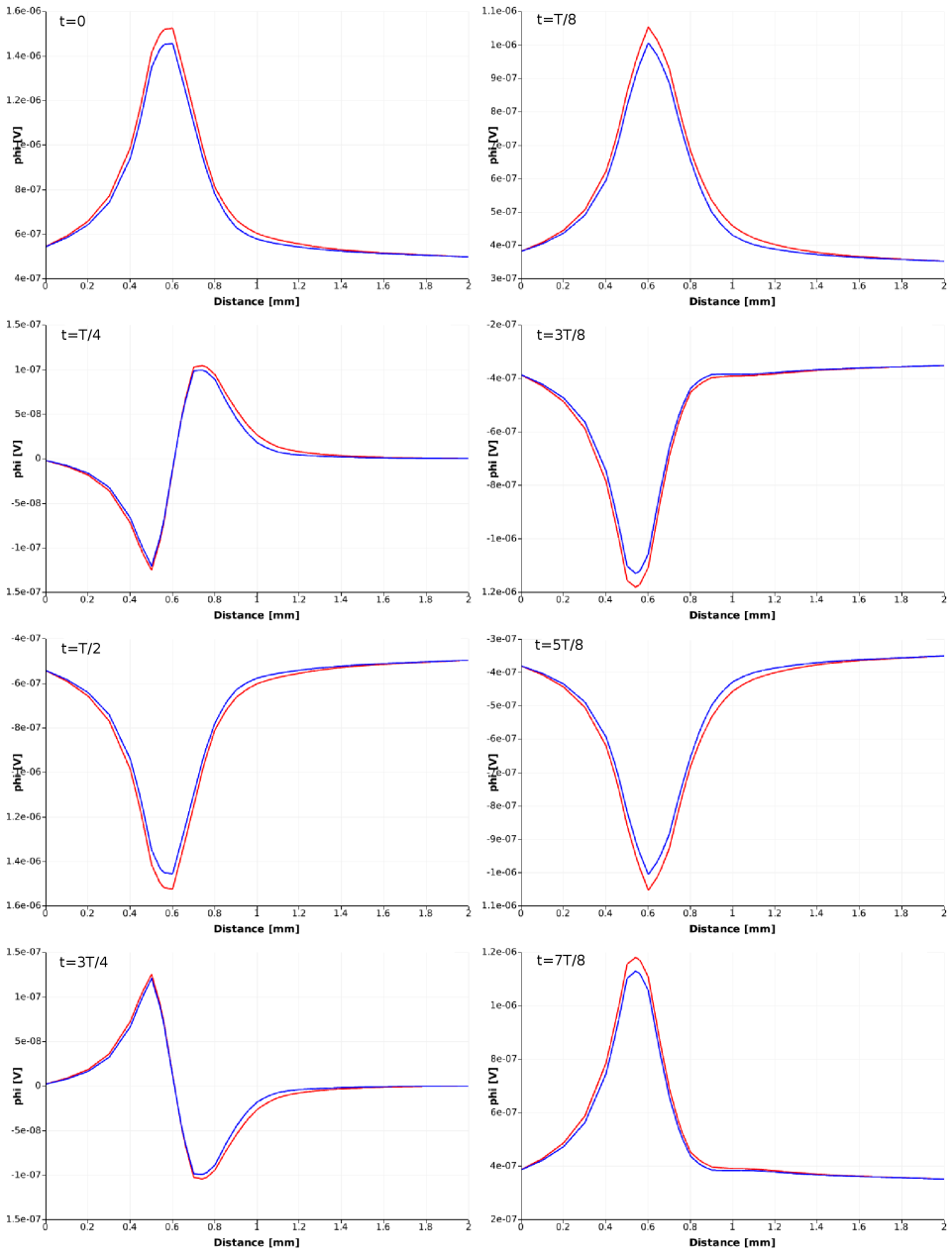

Figure 5.32: ***Impact of conductivity for ball-and-stick neuron with 1 kHz stimulus.*** *Line plots on the MEA along an axis parallel to the stick. The figures show $t = 0, \ldots, 7T/8$, for a somatic electrode current, $I_e = (250\,pA)\cos(2\pi f t)$, where $T = 1/f = 1$ ms. Blue line: MEA potential for homogeneous and isotropic tissue with conductivity $\sigma_{\text{tissue}} = 0.3$ S/m. Red line: MEA potential for inhomogeneous and anisotropic tissue with conductivity values given in table 4.2 on page 46. The conductivity profile influences the height of the potential peaks, whereas along most of the MEA the two curves follow each other closely. See section 5.4 for a discussion.*

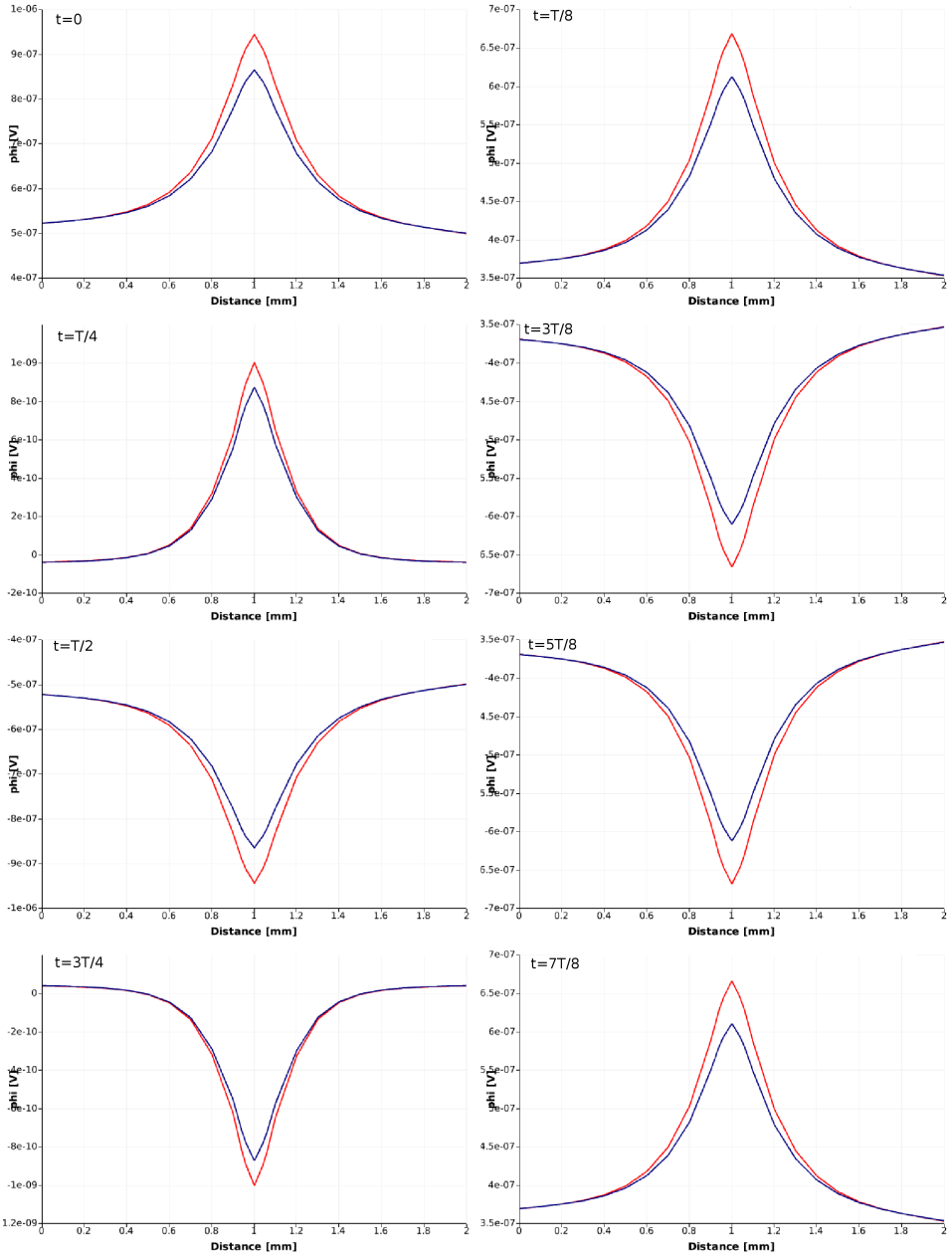

Figure 5.33: ***Impact of conductivity for ball-and-stick neuron with 1 Hz stimulus.*** *Line plots on the MEA along an axis perpendicular to the stick, crossing underneath its center. The figures show* $t = 0, \ldots, 7T/8$*, for a somatic electrode current,* $I_{\mathrm{e}} = (250\ pA)\cos(2\pi f t)$*, where* $T = 1/f = 1$ *s. Blue line: MEA potential for homogeneous and isotropic tissue with conductivity* $\sigma_{\mathrm{tissue}} = 0.3$ *S/m. Red line: MEA potential for inhomogeneous and anisotropic tissue with conductivity values given in table 4.2 on page 46. See section 5.4 for a discussion.*

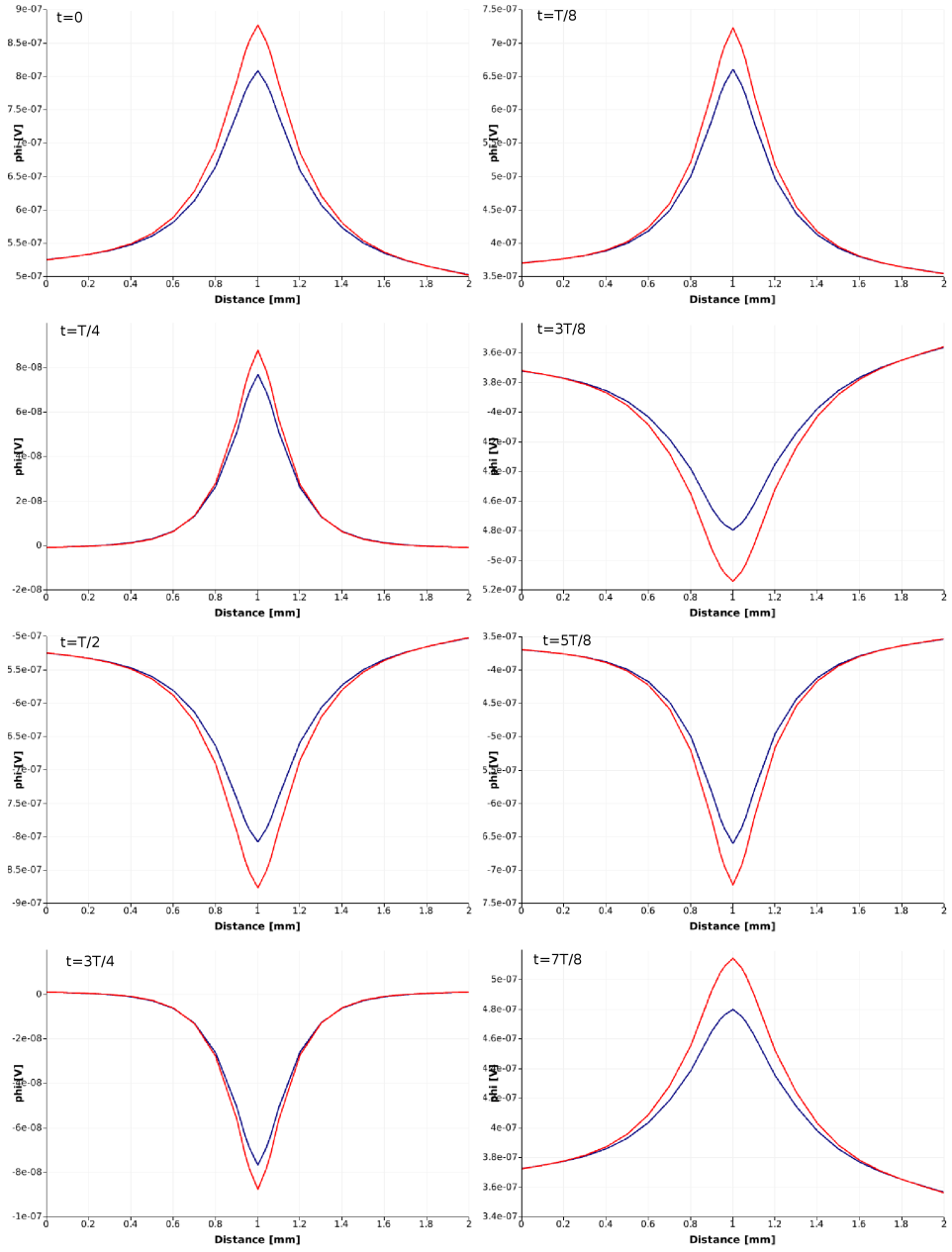

Figure 5.34: ***Impact of conductivity for ball-and-stick neuron with 100 Hz stimulus.*** *Line plots on the MEA along an axis perpendicular to the stick, crossing underneath its center. The figures show $t = 0, \ldots, 7T/8$, for a somatic electrode current, $I_e = (250\,pA)\cos(2\pi f t)$, where $T = 1/f = 10$ ms. Blue line: MEA potential for homogeneous and isotropic tissue with conductivity $\sigma_{\text{tissue}} = 0.3$ S/m. Red line: MEA potential for inhomogeneous and anisotropic tissue with conductivity values given in table 4.2 on page 46. See section 5.4 for a discussion.*

For the two lowest frequencies, it could be seen that the blue and the red line were quite different between 0.6 and 1.4 mm in all nonzero phases. In figure 5.31 they converge somewhat earlier than 1.4 mm, but no clear picture emerges. With $f = 1$ kHz, on the other hand, the difference between the two calculated potentials is clearly seen between 0.5 and 0.6 mm, i.e., from the soma and 0.1 mm down the stick. Outside of that region, they follow each other closely. This effect is caused by the returns currents leaving close to the soma due to low-pass filtering.

The ratio between the peak of the curves at $t = 0$ is 1.09 for 1 Hz, 1.07 for 10 Hz, 1.06 for 100 Hz, and 1.05 for 1 kHz, i.e., slightly decreasing with frequency. This can be interpreted as follows: For low frequencies, the current leaves quite uniformly throughout the stick length. Thus, the directional derivative of the potential along the stick axis is smaller than with higher f. Hence, more of the volume current flows radially outward from the membrane. As a result, the net perpendicular current between the ball-and-stick neuron and the MEA becomes larger for lower frequencies. By Ohm's law, for a given current, the potential difference in a direction of lower conductivity becomes larger. Thus, the potential gradient radially outward from the stick is higher in the anisotropic case because $\sigma_\perp < 0.3$ S/m. For higher frequencies, the membrane current will show a steeper decay along the length of the stick, so the directional derivative of the potential parallel to the stick has a larger magnitude. Hence, more volume current will turn in the stick direction after leaving the membrane, and less will be forced against the lower perpendicular conductivity.

Figures 5.33 and 5.34 show similar plots on the MEA along an axis perpendicular to the stick, passing underneath its mid point. The frequencies are $f = 1$ Hz and $f = 100$ Hz, respectively. In these two cases, the deviation between the red and the blue line seems to be very similar. Frequencies 10 Hz and 1 kHz were computed but not included here, since they showed practically the same as the two figures presented. The curve shapes do not change with frequency in these plots, since when a line perpendicular to its axis is traversed, the stick acts like a point source. Also, the small increase of magnitude for the inhomogeneous and anisotropic conductivity profile is seen here as well.

5.5 Parameter-Fitted Ball-and-Stick Model

In the simulations presented thus far, the goal has been to investigate what impact boundary conditions and conductivity profile have on MEA potentials. Now, an example application will be considered, illustrating how FEM modeling can be utilized in combination with real experiments.

When subthreshold phenomena are studied, the electrode current injected in the soma must be small enough to avoid firing of action potentials. An amplitude as low as 250 pA ensures this [9]. Further, neurons closer to the array than 200 μm are hard to stimulate, and due to electrode sensitivity the frequency typically has to be between between 5 Hz and 100 Hz [9]. Finally, when the soma of a particular cell is localized, its geometry can also be obtained and reconstructed in a simulation environment like NEURON.

Espen Hagen [24] performed a parameter-fitting of the ball-and-stick model, to reproduce the same potentials on the MEA as would a layer V pyramidal cell. This was done by adjusting the radii of the ball and the stick, and the potentials were calculated with the infinite medium solution (equation (5.1)). The values found were

$$r_{\mathrm{ball}} = 42.5 \ \mu\mathrm{m}$$

and

$$r_{\mathrm{stick}} = 1.5 \ \mu\mathrm{m}.$$

All other parameters were as shown in table 4.3 on page 49. Since the exact conductivity profile of the tissue for a given experiment is a priori unknown, it was assumed homogeneous and isotropic, with $\sigma_{\mathrm{tissue}} = 0.3$ S/m. Also note that the cell in this case was placed 200 μm above the MEA, as opposed to 100 μm in the previous cases.

The MEA potentials for the fitted ball-and-stick model are shown in figures 5.35 and 5.36, for the first and second half cycle of the stimulus current, respectively. Three frequencies, 5 Hz, 40 Hz, and 100 Hz, are compared. For the nonzero phases, the spatial spread of the potential is clearly seen to decrease with frequency. In the zero phase, $t = T/4$ and $t = 3T/4$, no clear picture emerges.

Figures 5.37 and 5.38 show line plots of the potential along axes parallel to and directly underneath the stick, for 5 different frequencies. These graphs illustrate how the shape of the measured potential changes with frequency. For the nonzero phases of the stimulus, the potential becomes steeper and narrower with increasing frequency. Comparing, e.g., the solid blue line (5 Hz) and the dashed red line (100 Hz) makes this very clear. Another very interesting result occurs when the stimulation current is zero. Here, the lowest frequency is seen to produce a very variations in the potential compared to the higher frequencies. This is because the *phase shift* depends on frequency. With near-constant current, the shift will be small, whereas for higher frequencies, a significant amount of current will cross the membrane also at the zero phase. As also noted in the last section, these potentials

should be interpreted with care, since at the zero phases, the stick current had to be adjusted quite significantly for the neuron to obey Kirchhoff's current law.

In future experiments, the ball-and-stick model can be fitted to match the particular cell that was stimulated to compute the resulting LFP. This paves the way for experimental testing of the volume conduction model and the ball-and-stick model. In principle, one could also measure the conductivity profile of the slices used in the experiment, and implement it into the FEM model.

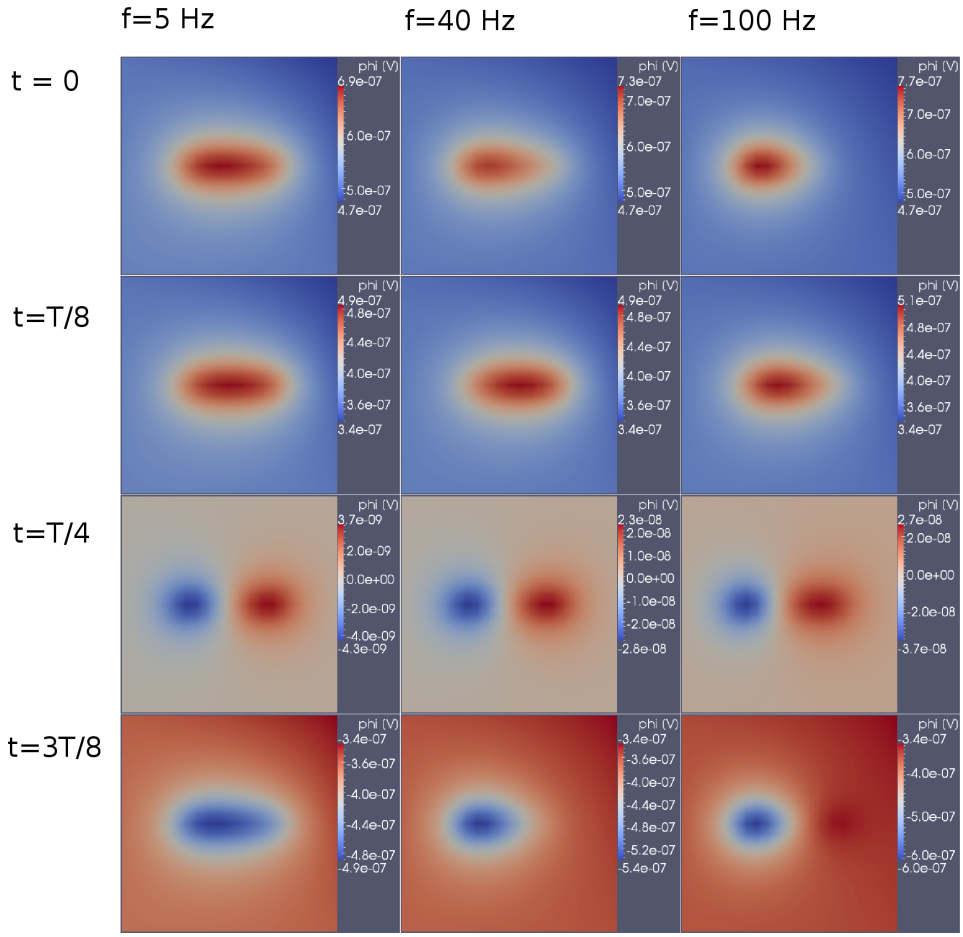

Figure 5.35: ***MEA potentials for ball-and-stick model fitted to layer V pyramidal cell.*** *The first half cycle of the electrode current is shown for the three frequencies considered. The time in terms of the period, $T = 1/f$, is shown for each plot. For the nonzero phases of the stimulus, the region with shifted potential is clearly seen to shrink with increasing frequency.*

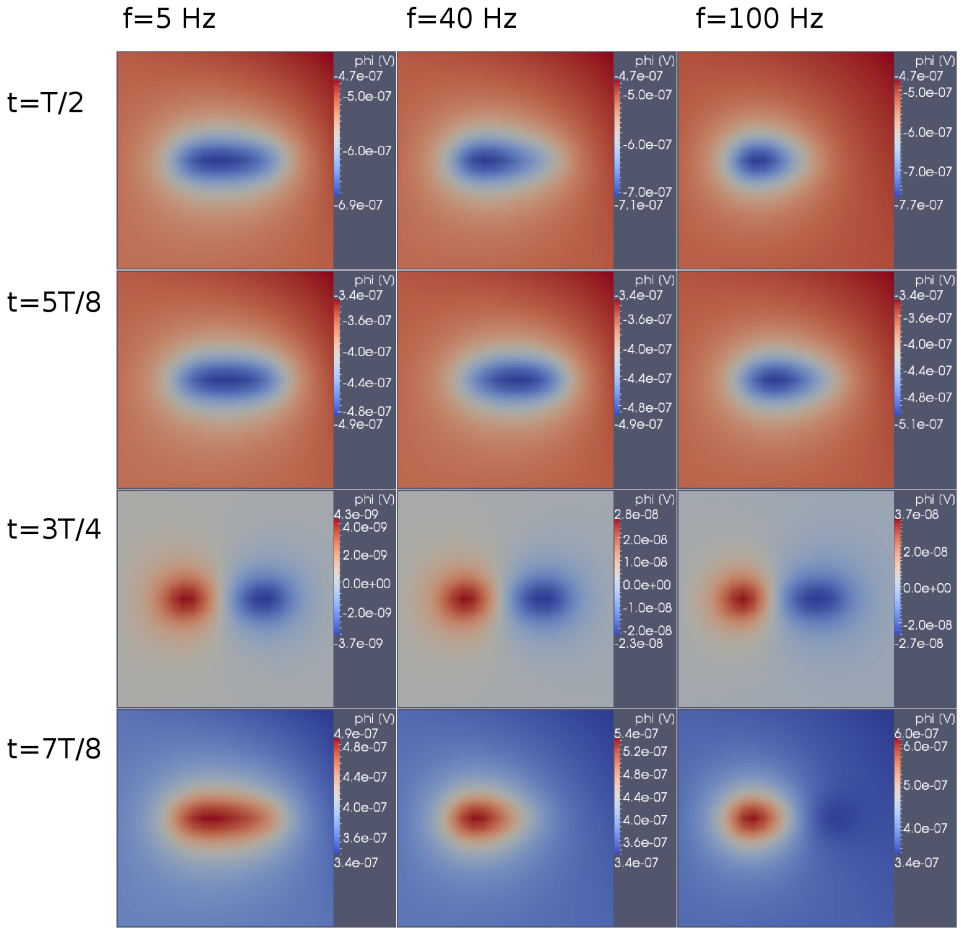

Figure 5.36: ***MEA potentials for ball-and-stick model fitted to layer V pyramidal cell.*** *The second half cycle of the electrode current is shown for the three frequencies considered. The time in terms of the period, $T = 1/f$, is shown for each plot. For the nonzero phases of the stimulus, the region with shifted potential is clearly seen to shrink with increasing frequency.*

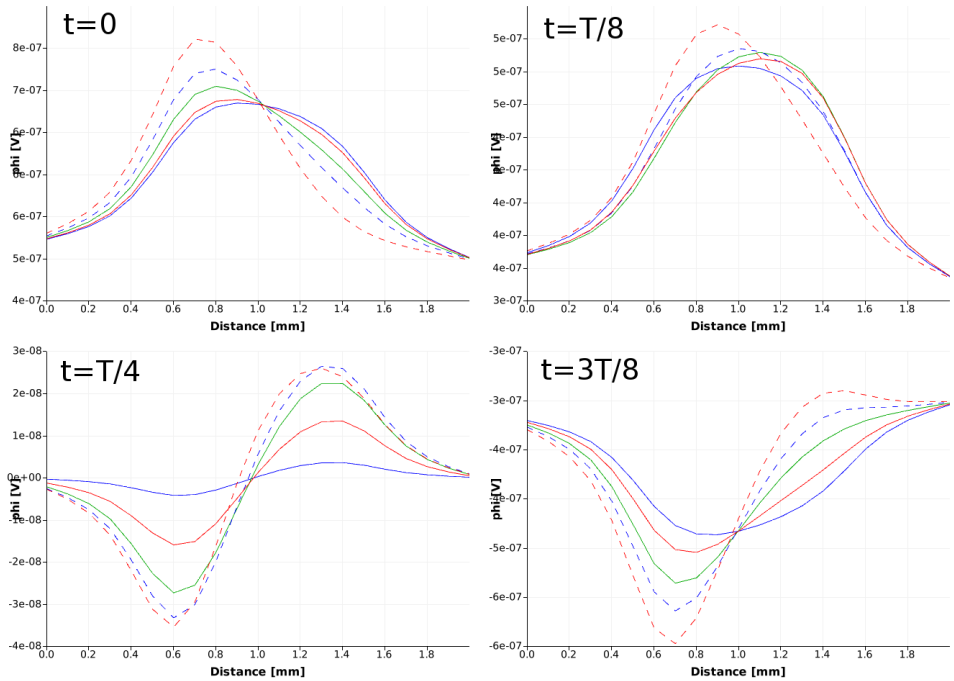

Figure 5.37: ***Line plot of MEA potential parallel to stick axis.*** *The first half cycle of the electrode current for the frequencies considered is shown. The time in terms of the period, $T = 1/f$, is shown for each plot. Blue line:* 5 *Hz, red line:* 20 *Hz, green line:* 40 *Hz, dashed blue line:* 60 *Hz, and dashed red line:* 100 *Hz.*

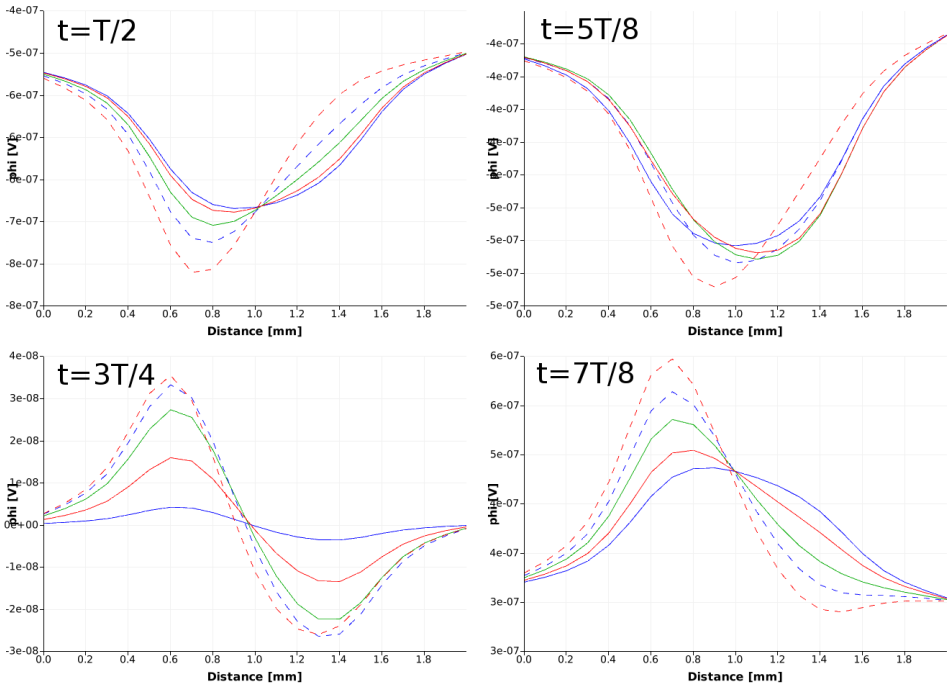

Figure 5.38: ***Line plot of MEA potential parallel to stick axis.*** *The second half cycle of the electrode current for the frequencies considered is shown. The time in terms of the period, $T = 1/f$, is shown for each plot. Blue line:* 5 *Hz, red line:* 20 *Hz, green line:* 40 *Hz, dashed blue line:* 60 *Hz, and dashed red line:* 100 *Hz.*

Chapter 6

Discussion

A two-monopole model is the simplest representation of a neuron receiving synaptic input, which generates extracellular potentials [35, 44]. The formula for the extracellular potential of two monopoles in an infinite medium, was seen to give other results than a more detailed numerical model. The latter took the insulating boundary conditions of MEA measurements into account (figure 5.10). Explicitly introducing the surrounding saline into the FEM implementation, gave potentials much closer to those predicted by the point source formula. However, the peaks still differed by about 20 % (middle plot of figure 5.10 and upper plot of 5.11). For the two-monopole approximation, biologically realistic variations of extracellular conductivity did not produce large changes in the calculated MEA potentials, compared to tissue with constant conductivity 0.3 S/m (figure 5.12).

The effects of large, and probably unrealistic, variations of the extracellular conductivity were investigated with a ball-and-stick model. Reduced conductivity resulted in a larger rate of change of the potential away from the neuron, and hence a rise of the MEA potentials (figures 5.6 and 5.7). This effect, predicted by Ohm's law, is illustrated by the line plots of figure 5.9.

The intrinsic low-pass filtering of the ball-and-stick model expressed itself in the potentials calculated on the microelectrode array (figures 5.21-5.24). Since the neuron receives a current stimulus, the net current crossing its membrane equals what the electrode supplies. The neuron will thus have a nonzero monopole moment, as opposed to the *in vivo* situation, in which the membrane currents always cancel. In any case, a low-pass filtering effect can be observed by the narrowing of the MEA potential peak, with increasing frequency. The maximum potential, with respect to the reference electrode, also increased with frequency.

As for two monopoles, the exact nature of the tissue conductivity was not seen to be a critical variable for the the MEA potentials of a ball-and-stick model. With the biologically plausible conductivity profile obtained by Goto et al. [22], a change of less than 8 % was observed, when compared to an identical cell situated in tissue of constant conductivity 0.3 S/m. Since there may be significant a variation of conductivity from one slice to another [22], and because the ball-and-stick model is

a coarse simplification of a real neuron, the possible error introduced by assuming constant tissue conductivity seems relatively small. The impact of anisotropy and inhomogeneity also slightly decreased with frequency (figures 5.29-5.34).

The ball-and-stick model, with radii adjusted to reproduce the LFP of a pyramidal neuron receiving the same electrode current, can be compared to real experiments with such a cell. Although exact match is not expected, qualitative similarities in the *shape* of the MEA potentials would suggest that the ball-and-stick and volume conduction model used here are catching some of the biologically important features.

A conclusion to draw from the work, is that the simple analytical solution, for an infinite and homogeneous medium containing point sources, does not reproduce the MEA potentials of more detailed numerical simulations. In future studies, reconstructed cells, with transmembrane currents calculated in a simulation environment like NEURON [4], can hopefully be imported into FEniCS. This would allow the extracellular potentials to be computed with a finite element model, taking the actual experimental set-up into account. The accuracy of the point source formula was only investigated for a two-monopole model. For the future, it would be interesting to see how the point- and line-source formulae [26, 43] for a ball-and-stick neuron, as well as more complex cells, compare to FEM computations.

As long as the exact nature of the conductivity profile of cortical tissue is uncertain, it seems like a good compromise to assume some constant value, e.g., 0.3 S/m. Small fluctuations from this, either spatial variations or directional dependence, do not produce large differences in the MEA potentials. Considering stimulation frequencies separately, a possible *frequency-dependent conductivity* can also be modeled, as pointed out by Pettersen et al. [44]. Due to linearity, the extracellular potential is oscillating with the same frequency as the stimulus current, although phase shifts in general occur. Hence, by assigning a particular conductivity tensor for each frequency, MEA potentials could be calculated within the framework of the FEM simulations presented here.

An additional goal of this work was to evaluate the suitability of FEniCS, both for simulation of MEA measurements and for computation of extracellular potentials in general. As should be clear, FEniCS requires a basic understanding of FEM, mesh generation, and Python scripting or C++ programing. More advanced use, e.g., meshing reconstructed neurons, requires additional packages, like TriTetMesh [36]. A timely question is whether the graphical environment of commercial packages, like COMSOL [1], may provide more user-friendliness to the non-expert. COMSOL was used by, e.g., references [20, 21] for modeling of MEA potentials. The Python interface of FEniCS, however, was found by the author to be very easy to adapt to. One particular advantage of scripting is the ease with wich more complicated models can be implemented. The potential user is therefore recommended to try the source code accompanying this thesis, before deciding on whether or not to use FEniCS.

Appendix A

Some Notes on Units

The expressions for the ball-and-stick membrane currents contain a variety of parameters with different dimensions and units, so careful calculations are needed to get the desired output. The primary unknown in the simulations was the extracellular potential, and units of *volts* were preferred. Table A.1 shows the parameters used, together with their units.

Now, consider the weak form for the extracellular potential generated by a ball-and-stick model with somatic current stimulus,

$$\int_{\Omega} \sigma \nabla u \cdot \nabla v \mathrm{d}x = \int_{\partial\Omega_7} J_{\mathrm{ball}} v \mathrm{d}s + \int_{\partial\Omega_8} J_{\mathrm{stick}} v \mathrm{d}s. \tag{A.1}$$

The soma current is the real part of

$$\mathbf{J}_{\mathrm{ball}}(t) = \frac{1}{S} \frac{\mathbf{Y}_{\mathrm{ball}}}{\mathbf{Y}_{\mathrm{ball}} + \mathbf{Y}_{\mathrm{stick}}} \hat{\mathbf{I}}_0 e^{j\omega t}, \tag{A.2}$$

and the stick current is the real part of

$$\mathbf{J}_{\mathrm{stick}}(x,t) = \frac{1}{\pi d} \frac{\mathbf{H}(x)}{\mathbf{Y}_{\mathrm{ball}} + \mathbf{Y}_{\mathrm{stick}}} \hat{\mathbf{I}}_0 e^{j\omega t}. \tag{A.3}$$

The complex variable $\mathbf{s}$ is given by

$$\mathbf{s} = \sqrt{1 + j\omega\tau_{\mathrm{m}}},$$

where $[\omega\tau_{\mathrm{m}}] = [10^{-3}]$ from table A.1. Thus, in order to get the units right, the time constant is multiplied by 10^{-3} upon definition. The following snippet illustrates:

```
# Membrane time constant:
tau = R_m*C_m*1e-3 # [ms]
```

The soma admittance,

$$\mathbf{Y}_{\mathrm{ball}} = \frac{4\pi d^2 \mathbf{s}^2}{r_{\mathrm{m}}},$$

Symbol	Unit	Description
l	[mm]	Stick length
d	[mm]	Stick diameter
r_m	[Ω cm^2]	Membrane resistivity
r_i	[Ω cm]	Intracellular resistance
c_m	[μF /cm^2]	Membrane capacitance
σ	[S/m]	Volume conductivity
f	[kHz]	Stimulus frequency
I_0	[mA]	Stimulus amplitude
t	[ms]	Time
S	[mm^2]	Membrane surface area
$\omega = 2\pi f$	$[(ms)^{-1}]$	Angular frequency
$\tau_\mathrm{m} = r_\mathrm{m} c_\mathrm{m}$	[μs]	Membrane time constant
$\lambda = \sqrt{d r_\mathrm{m}/4 r_\mathrm{i}}$	[$10^{-5/2}$ m]	Electrotonic length

Table A.1: ***Parameter names and units.*** *Parameters used for the membrane currents of the ball-and-stick model.*

has units

$$\left[\frac{\mathrm{mm}^2}{\Omega\mathrm{cm}^2}\right] = \left[10^{-2}\ \mathrm{S}\right].$$

For the stick admittance,

$$\mathbf{Y}_\mathrm{stick} = \frac{\pi d^{3/2}\mathbf{s}}{2\sqrt{r_\mathrm{i} r_\mathrm{m}}}\left[\frac{1}{1+\exp(2\mathbf{s}l/\lambda)} - \frac{1}{1+\exp(-2\mathbf{s}l/\lambda)}\right],$$

the factor in the exponential has units

$$\left[\frac{\mathbf{s}l}{\lambda}\right] = \frac{\mathrm{mm}}{10^{-5/2}\ \mathrm{m}} = \sqrt{10}.$$

Thus, λ is be multiplied by $\sqrt{10}$ when defined, i.e.,

```
# Electrotonic length
lbda = sqrt(stick_diameter*R_m/4/R_i)*sqrt(10) # [mm]
```

After this correction, the stick admittance has units

$$[\mathbf{Y}_\mathrm{stick}] = \left[\frac{\mathrm{mm}^{3/2}}{\sqrt{\Omega^2\ \mathrm{cm}^3}}\right] = \left[10^{-3/2}\ \mathrm{S}\right].$$

The stick transfer function,

$$\mathbf{H}(x) = \frac{\pi \mathbf{s}^2 d}{R_\mathrm{m}}\left[\frac{\exp(\mathbf{s}x/\lambda)}{1+\exp(2\mathbf{s}x/\lambda)} + \frac{\exp(-\mathbf{s}x/\lambda)}{1+\exp(-2\mathbf{s}x/\lambda)}\right],$$

has units

$$[\mathbf{H}] = \left[\frac{\mathrm{mm}}{\Omega\ \mathrm{cm}^2}\right] = [10\ \mathrm{S/m}].$$

In the denominator of equations (A.2) and (A.3),

$$\mathbf{Y}_{\text{ball}} + \mathbf{Y}_{\text{stick}},$$

$\mathbf{Y}_{\text{ball}}$ has to be multiplied with 10^{-2} and $\mathbf{Y}_{\text{stick}}$ with $10^{-3/2}$, in order to get units of *siemens* in both terms. It follows that

$$[\mathbf{J}_{\text{ball}}] = \left[\frac{\text{mA}}{\text{mm}^2}\right].$$

The following lines, C++ code defining the membrane current of the stick, illustrates:

```
std::complex<double> Y_soma = (4.0*pi*pow(d,2)/R_m)*
    s_squared*pow(10,-2);
std::complex<double> Y_stick = (pi*pow(d,1.5)/2.0*pow(R_i*
    R_m,-.5))*s*std::tanh(s*l/lbda)*pow(10,-1.5);
```

For the stick current, the units become

$$[\mathbf{J}_{\text{stick}}] = \left[\frac{(10\ \text{S/m})\ \text{mA}}{\text{S mm}^2}\right] = \left[10^{-2}\frac{\text{mA}}{\text{mm}^2}\right].$$

Hence, the numerical value for $\mathbf{J}_{\text{stick}}$ in the code must be multiplied by 10^{-2} to give units [mA/mm^2]. This is illustrated by the snippet below.

```
std::complex<double> J_m = s_squared/R_m*(std::exp(s*
    distance/lbda)/(1.0+std::exp((2.0*l/lbda)*s))+std::exp
    (-s*distance/lbda)/(1.0+std::exp(-(2.0*l/lbda)*s)))*(
    V_soma*pow(10,-2));
```

With these multiplying factors, all membrane currents get units [mA/mm^2]. Now consider the weak form for the ball-and-stick model, equation (A.1). The length unit used in the mesh is [mm]. Thus,

$$\begin{aligned}[\mathrm{d}x] &= \left[\text{mm}^3\right]\\ [\mathrm{d}s] &= \left[\text{mm}^2\right]\\ [\nabla] &= \left[\text{mm}^{-1}\right]\end{aligned}$$

Denoting the units of the extracellular potential by X, we get[1]

$$\left[\int_\Omega \sigma\nabla u\cdot\nabla v\mathrm{d}x\right] = \left[\frac{\text{(S/m) X mm}^3}{\text{mm}^2}\right]$$

and

$$\left[\int_{\partial\Omega_7} J_{\text{ball}} v\mathrm{d}s + \int_{\partial\Omega_8} J_{\text{stick}} v\mathrm{d}s\right] = \left[\frac{\text{mA}}{\text{mm}^2}\text{mm}^2\right].$$

Equating the right-hand sides of these expressions and solving in terms of X gives

$$[\text{X}] = [V].$$

Accordingly, the extracellular potential has units of *volts* (V).

[1] The test function v is neglected. It appears in every term, so its units cancel.

Appendix B

Source Code

Figure B.1 demonstrates the general organization of the code. For each of the simulations performed, a file, `create_mesh.py`, has been used to generate the mesh, subdomain markers, and boundary markers. These are saved to `mesh.xml`, `subdomains.xml`[1], and `boundaries.xml`. The `*.xml` files are subsequently used by `simulation.py`, which runs the simulations. Its results are again exported to `*.pvd` and `*.vtu` files for post-processing [49], e.g., with ParaView [5]. Figure B.1 illustrates this.

The attached CD contains the complete source code. It is organized in folders as shown in table B.1. Each contains the files `create_mesh.py` and `simulation.py`, for the particular implementation. On a Linux computer with FEniCS properly installed, mesh generation is now just a matter of typing

```
python create_mesh.py
```

in the shell. The simulation runs with the command

```
python simulation.py
```

The code for a ball-and-stick neuron in an anisotropic and inhomogeneous tissue (`BS_Anis_Inhom` in table B.1) is included here. The code for a two-monopole approximation, testing the impact of saline (`TM_Impact_of_Saline` in table B.1), is also included.

[1] The test of the numerical accuracy (section 5.1) did not use different subdomains, and hence no subdomain file was generated.

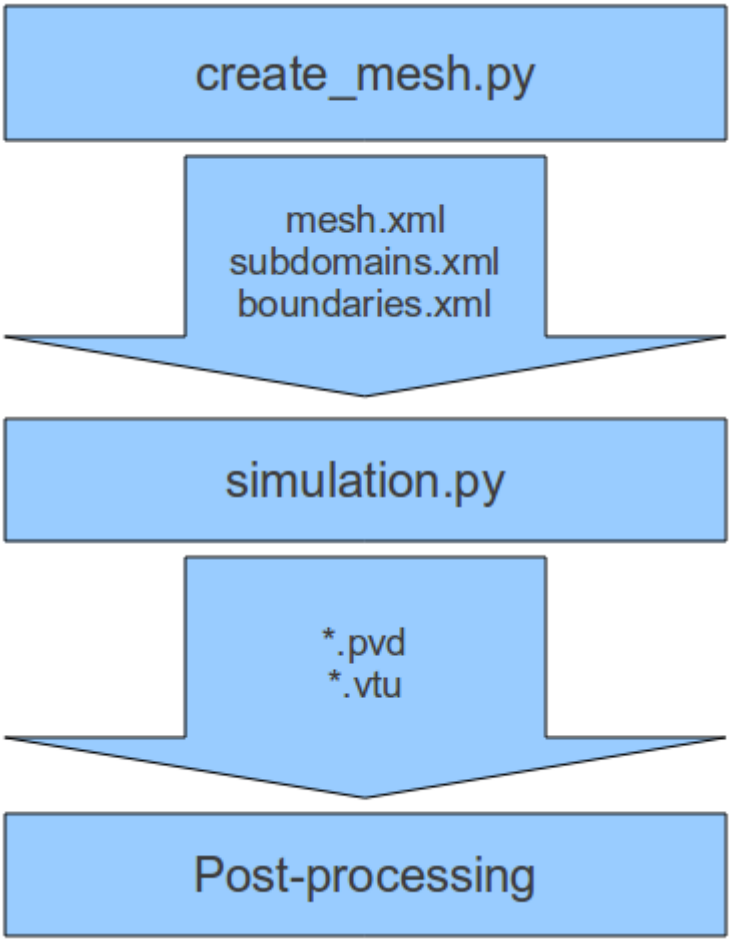

Figure B.1: ***Organization of the code.*** *A script,* `create_mesh.py`*, generates the mesh, together with subdomain and boundary markers. These are saved on *.xml files, subsequently taken as input to* `simulation.py`*, which runs the simulations. Final results are output in files suitable for visualization in ParaView [5].*

Folder Name	Content
`TM_Numerical_Accuracy`	Verification of numerical accuracy (section 5.1)
`BS_Large_Inhomogeneity`	Large inhomogeneities (section 5.2)
`BS_Large_Anisotropy`	Large anisotropy (section 5.2)
`TM_Impact_of_BSs`	Comparison, finite vs. infinite medium (section 5.3)
`TM_Impact_of_Saline`	Comparison, taking only boundary conditions into account vs. also modeling saline (section 5.3)
`TM_Impact_of_Conductivity`	Comparison, homogeneous and isotropic tissue vs. inhomogeneous and anisotropic tissue (section 5.3)
`BS_Isotropic_Homogeneous`	Ball-and-stick in an isotropic and homogeneous tissue (section 5.4)
`BS_Anis_Inhom`	Ball-and-stick in an anisotropic and inhomogeneous tissue (section 5.4)
`BS_Fitted`	Parameter-fitted ball-and-stick model (section 5.5)

Table B.1: ***Organization of source code.*** *Folders in the source code containing scripts for the different simulations performed. In the folder names, TM stands for two-monopole and BS for ball-and-stick.*

Ball-and-Stick Neuron

Mesh Generation

The file `create_mesh.py` from `BS_Anis_Inhom` is shown below.

```
 #!/usr/bin/env python
"""
create_mesh.py
BS_Anis_Inhom
This script generates the mesh to be used in the ball and stick
    simulation with an inhomogeneous and anisotropic tissue
    conductivity.
"""

from dolfin import *

# Outer box dimensions (millimeters are used as units for all
    length variables in the script!)
box_x_dim = 4; box_y_dim = 4; box_z_dim = 1

# Initial number of cells in each spatial direction (0.1 mm
    resolution)
nx = 40; ny = 40; nz = 10

# Tissue slice dimensions
tissue_x_dim = 2; tissue_y_dim = 2; tissue_z_dim = 0.3

# Ball and stick dimension
radius = 0.01 # radius of ball
stick_radius = 0.001 # radius of stick
stick_diameter = stick_radius*2
stick_length = 1.
ball_z_height = 0.1 # the height of the center of the ball (as
    well as the stick) above the electrode array

# Coordinates:
# Center of the ball:
ball_x_coor = -stick_length/2.
ball_y_coor = 0.
ball_z_coor = -box_z_dim/2. + ball_z_height
# Point representing the ball origin:
ball_coor = Point(ball_x_coor, ball_y_coor, ball_z_coor)
# Stick coordinates:
stick_x_left = -stick_length/2.+radius # the left end, attached to
     the ball
stick_x_right = stick_length/2.+radius # the right end
stick_y = 0 # the center in the y-direction
stick_z = -box_z_dim/2. + ball_z_height # the center in the z-
    direction

# Extent of the layers with different conductivities
layer_VI_length = .8
layer_V_length = .4
```

```
layer_IV_length = .4
layer_II_III_length = .4

# One class for each cortical layer
class LayerVI(SubDomain):
    def inside(self, x, on_boundary):
        return -tissue_x_dim/2.<= x[0] <= -tissue_x_dim/2.+
            layer_VI_length and -tissue_y_dim/2. <= x[1] <=
            tissue_y_dim/2. and -box_z_dim/2.<= x[2] <= -box_z_dim
            /2. + tissue_z_dim

class LayerV(SubDomain):
    def inside(self, x, on_boundary):
        return -tissue_x_dim/2.+layer_VI_length<= x[0] <= -
            tissue_x_dim/2.+layer_VI_length+layer_V_length*1.1 and
             -tissue_y_dim/2. <= x[1] <= tissue_y_dim/2. and -
            box_z_dim/2.<= x[2] <= -box_z_dim/2. + tissue_z_dim

class LayerIV(SubDomain):
    def inside(self, x, on_boundary):
        return -tissue_x_dim/2.+layer_VI_length + layer_V_length<=
             x[0] <= -tissue_x_dim/2.+layer_VI_length+
            layer_V_length+layer_IV_length and -tissue_y_dim/2. <=
             x[1] <= tissue_y_dim/2. and -box_z_dim/2.<= x[2] <= -
            box_z_dim/2. + tissue_z_dim

class LayerII_III(SubDomain):
    def inside(self, x, on_boundary):
        return -tissue_x_dim/2.+layer_VI_length+layer_V_length+
            layer_IV_length <= x[0] <= tissue_x_dim/2. and -
            tissue_y_dim/2. <= x[1] <= tissue_y_dim/2. and -
            box_z_dim/2.<= x[2] <= -box_z_dim/2. + tissue_z_dim

# Class representing the ball
class Ball(SubDomain):
    def inside(self, x, on_boundary):
        r = sqrt((x[0]-ball_x_coor)**2 + (x[1]-ball_y_coor)**2 + (
            x[2]-ball_z_coor)**2)
        return r<1.5*radius
    def snap(self, x):
        r = sqrt((x[0]-ball_x_coor)**2 + (x[1]-ball_y_coor)**2 + (
            x[2]-ball_z_coor)**2)
        if r < 1.5*radius:
            x[0] = ball_x_coor + (radius/r)*(x[0]-ball_x_coor)
            x[1] = ball_y_coor + (radius/r)*(x[1]-ball_y_coor)
            x[2] = ball_z_coor + (radius/r)*(x[2]-ball_z_coor)

# Class representing the stick
class Stick(SubDomain):
    def inside(self, x, on_boundary):
        r = sqrt((x[1]-stick_y)**2 + (x[2]-stick_z)**2)
        return r<1.5*stick_radius and stick_x_left-.5*radius <= x
            [0] <= stick_x_right+.5*radius
    def snap(self, x):
```

```
        r = sqrt((x[1]-stick_y)**2 + (x[2]-stick_z)**2)
        if r<1.5*stick_radius and stick_x_left <= x[0] <=
            stick_x_right+.5*radius:
            x[1] = stick_y + (stick_radius/r)*(x[1]-stick_y)
            x[2] = stick_z + (stick_radius/r)*(x[2]-stick_z)

# Classes representing the outer boundaries (numbered from 1-6)
class Boundary1(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and abs(x[0]-box_x_dim/2.)<=DOLFIN_EPS
class Boundary2(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and abs(x[1]-box_y_dim/2.)<=DOLFIN_EPS
class Boundary3(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and abs(x[2]-box_z_dim/2.)<=DOLFIN_EPS
class Boundary4(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and abs(x[0]+box_x_dim/2.)<=DOLFIN_EPS
class Boundary5(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and abs(x[1]+box_y_dim/2.)<=DOLFIN_EPS
class Boundary6(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and abs(x[2]+box_z_dim/2.)<=DOLFIN_EPS

# Class representing the boundary of the ball
class BallBoundary(SubDomain):
    def inside(self, x, on_boundary):
        r = sqrt((x[0]-ball_x_coor)**2 + (x[1]-ball_y_coor)**2 + (
            x[2]-ball_z_coor)**2)
        return on_boundary and r < 1.1*radius

# Class representing the boundary of the stick
class StickBoundary(SubDomain):
    def inside(self, x, on_boundary):
        r = sqrt((x[1]-stick_y)**2 + (x[2]-stick_z)**2)
        return on_boundary and r<1.5*stick_radius and stick_x_left
             <= x[0] <= stick_x_right+radius

# Class representing the part of the boundary to be grounded (
    reference electrode)
class GroundBoundary(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and box_x_dim/2.-0.1 <= x[0] <=
            box_x_dim/2. and box_y_dim/2.-0.1<=x[1]<=box_y_dim/2.
            and abs(x[2]-box_z_dim/2.)<=.01

# Generation of the original mesh:
mesh = Box(-box_x_dim/2.,-box_y_dim/2.,-box_z_dim/2.,box_x_dim/2.,
    box_y_dim/2.,box_z_dim/2.,nx,ny,nz)

# Print mesh information to the shell
print mesh
```

```
# Multipliers used in the successive refinements:
# Because the stick is narrower than the ball, the region of the
    mesh to be occupied by the stick has to be refined more times
    than the ball before the stick can be extracted. Hence the
    number of stick multipliers is larger. All the ones at the end
     are the to avoid reaching the end of the ball_multipliers
    vector before the refinement is finished. ball_multipliers has
     to be at least as large as stick_multipliers
ball_multipliers = [15,15,10,10,8,8,7,6,5,5,4,3,2,2,
    1.5,1.5,1.4,1.3,1,1, 1,1,1,1,1,1,1,1, 1,1,1,1]
stick_multipliers = [200,100,90,75,50,50,40,35,30,25,20,16,12,8,7,
     6,4,3,3,2.5,2]
num_refinements = len(stick_multipliers)

# Loop over the mesh refinements
for i in range(num_refinements):

# Class representing the part of the mesh to be refined. It varies
     according to the multipliers defined above.
    class Refinement(SubDomain):
        def inside(self, x, on_boundary):
            r = sqrt((x[0]-ball_x_coor)**2 + (x[1]-ball_y_coor)**2
                 + (x[2]-ball_z_coor)**2)
            r_stick = sqrt((x[1]-stick_y)**2 + (x[2] - stick_z)
                **2)

# If the value of ball_multipliers for the particular i is equal
    to 1, only the stick region should be refined. Otherwise both
    the stick and the ball region should be refined.
            if not ball_multipliers[i] == 1:
                return (r < ball_multipliers[i]*radius) or (
                    r_stick < stick_multipliers[i]*stick_radius
                    and stick_x_left <= x[0] <= stick_x_right)
            else:
                return ( r_stick < stick_multipliers[i]*
                    stick_radius and stick_x_left <= x[0] <=
                    stick_x_right)

# Mesh function ref_region equals 1 on the part of the mesh to be
    refined an 0 otherwise. A boolean mesh function can not be
    constructed directly, and thus we have to go the way through
    ref_region:
    ref_region = MeshFunction('uint', mesh, mesh.topology().dim())
    ref_region.set_all(0) # set all markers to 0
    Refinement().mark(ref_region,1) # mark region to be refined
        with a 1

# Define a boolean mesh function 'markers', which is true in cells
     to be refined and false otherwise.
    markers = MeshFunction('bool', mesh, mesh.topology().dim())
    markers.set_all(False)
```

```
# Run through the whole mesh. If ref_region[cell]=1, the cell
    should be refined (i.e. markers[cell]=True), otherwise it
    should not (i.e. markers[cell]=False).
    for cell in cells(mesh):
        if ref_region[cell.index()] == 1:
            markers[cell.index()] = True

# Define a new mesh; the refined version of the last mesh
    mesh = refine(mesh, markers)

# Print mesh information to the shell
    print mesh

# Extract submesh excluding ball and stick

# Define the mesh function 'subdomains', which is a 'helper'
    function for extracting the ball and stick
subdomains = MeshFunction('uint', mesh, mesh.topology().dim())
# Set 'subdomains' to 0 all over the mesh
subdomains.set_all(0)
# Define 'ball' as an instance of the Ball() subclass and mark it
    with the value 1 in the 'subdomains' function
ball = Ball()
ball.mark(subdomains, 1)
# Define 'stick' as an instance of the Stick() subclass and mark
    it with the value 2 in the 'subdomains' function
stick = Stick()
stick.mark(subdomains,2)

# Extract the parts of the mesh with the value 1 or 2 in the '
    subdomains' function and return it to the new 'mesh' instance
mesh = SubMesh(mesh, subdomains, 0)

# Because we now have a new mesh, the mesh functions have to be
    defined again!
subdomains = MeshFunction('uint', mesh, mesh.topology().dim())
subdomains.set_all(0)
# Mark the layers
LayerVI().mark(subdomains,1)
LayerV().mark(subdomains,2)
LayerIV().mark(subdomains,3)
LayerII_III().mark(subdomains,4)

# 'boundaries' is a mesh function to be used for marking the
    boundaries with the correct indicators
# Note that the boundaries are 2-dimensional surfaces, hence the
    term 'mesh.topology().dim()-1'
# The values are initialized to zero over the whole mesh
boundaries = MeshFunction('uint', mesh, mesh.topology().dim()-1)
boundaries.set_all(0)

# Mark the boundaries by giving the 'boundaries' meshfunction the
    right values on the boundary
```

```
# Note that the snapping of the ball and stick boundaries happens
    just before these boundaries are marked. If putting the snap
    statements on the top, the markers will not be defined propely
Boundary1().mark(boundaries,1)
Boundary2().mark(boundaries,2)
Boundary3().mark(boundaries,3)
Boundary4().mark(boundaries,4)
Boundary5().mark(boundaries,5)
Boundary6().mark(boundaries,6)
GroundBoundary().mark(boundaries,9)
mesh.snap_boundary(ball)
BallBoundary().mark(boundaries,7)
mesh.snap_boundary(stick)
StickBoundary().mark(boundaries,8)

# Save the mesh and the markers to file. They will be used by the
    script 'simulation.py'
file = File('mesh.xml')
file << mesh
file = File('subdomains.xml')
file << subdomains
file = File('boundaries.xml')
file << boundaries
```

Simulation

The file `simulation.py` from `BS_Anis_Inhom` is shown below.

```
 #!/usr/bin/env python
"""
simulation.py
BS_Anis_Inhom
This Python script runs the simulations for a ball-and-stick
    neuron in a tissue with anisotropic and inhomogeneous
    conductivity
"""

from dolfin import *
import numpy

# Use the mesh, subdomains, and boundary markers generated by '
    create_mesh.py'
# After running the script 'create_mesh.py', these files should
    reside in the folder
mesh = Mesh('mesh.xml')
subdomains = MeshFunction('uint', mesh, 'subdomains.xml')
boundaries = MeshFunction('uint', mesh, 'boundaries.xml')

# Outer box dimensions (millimeters are used as units for all
    length variables in the script!)
box_x_dim = 4; box_y_dim = 4; box_z_dim = 1
```

```
# Initial number of cells in each spatial direction (0.1 mm
    resolution)
nx = 40; ny = 40; nz = 10

# Tissue slice dimensions
tissue_x_dim = 2; tissue_y_dim = 2; tissue_z_dim = 0.3

# Ball and stick dimension
radius = 0.01 # radius of ball
stick_radius = 0.001 # radius of stick
stick_diameter = stick_radius*2
stick_length = 1.
ball_z_height = 0.1 # the height of the center of the ball (as
    well as the stick) above the electrode array

# Coordinates:
# Center of the ball:
ball_x_coor = -stick_length/2.
ball_y_coor = 0.
ball_z_coor = -box_z_dim/2. + ball_z_height
# Point representing the ball origin:
ball_coor = Point(ball_x_coor, ball_y_coor, ball_z_coor)
# Stick coordinates:
stick_x_left = -stick_length/2.+radius # the left end, attached to
     the ball
stick_x_right = stick_length/2.+radius # the right end
stick_y = 0 # the center in the y-direction
stick_z = -box_z_dim/2. + ball_z_height # the center in the z-
    direction

# Electrophysiological parameters
R_m = 3e4 # [Ohm cm^2], membrane resistance
R_i = 150 # [Ohm cm], intracellular resistance
C_m = 1 # [microFarad / cm^2], membrane capacitance
f = 1.0 # [kHz], frequency of soma potential
I_0 = 250e-9

sigma_saline = 3. # [S/m], saline conductivity

# Membrane time constant:
tau = R_m*C_m*1e-3 # [ms]
# Electrotonic length
lbda = sqrt(stick_diameter*R_m/4/R_i)*sqrt(10) # [mm]
# Angular frequency of stimulus
omega = 2*pi*f

# Conductivity values in the different layers [S/m]
# Values are taken from Goto et al.
sigma_II_III_parallel = .319
sigma_II_III_perpendicular = .231
sigma_IV_parallel = .325
sigma_IV_perpendicular = .24
sigma_V_parallel = .353
sigma_V_perpendicular = .228
```

```
sigma_VI_parallel = .294
sigma_VI_perpendicular = .268

# Define a Hilbert space of piecewise linear basis functions
V = FunctionSpace(mesh, 'CG', 1)

# Explicitly calculate soma and stick area numerically
# Soma surface area
area_soma = assemble(Constant(1)*ds(7), mesh=mesh,
    exterior_facet_domains=boundaries)
# Stick surface area
area_stick = assemble(Constant(1)*ds(8), mesh=mesh,
    exterior_facet_domains=boundaries)

# C++ string for soma current
soma_code = """
class SomeExpr : public Expression
{
public:
    SomeExpr() : Expression(), omega(1), tau(30.), d(.002), R_i
        (150),radius(.01), R_m(3e4), l(1.), lbda(1.), t(0.),
        area_soma(.00125), I_0(250e-9) {}
    //variable declarations
    double omega, tau, d, R_i, R_m, l, lbda, t, area_soma, I_0,
        radius;

    void eval(Array<double>& values, const Data& data) const
    {
        std::complex<double> s_squared(1,omega*tau);
        std::complex<double> s = std::sqrt(s_squared);
        std::complex<double> Y_soma = (4.0*pi*pow(d,2)/R_m)*
            s_squared*pow(10,-2);
        std::complex<double> Y_stick = (pi*pow(d,1.5)/2.0*pow(R_i*
            R_m,-.5))*s*std::tanh(s*l/lbda)*pow(10,-1.5);
        std::complex<double> euler(cos(omega*t),sin(omega*t));
        std::complex<double> dividend = (Y_soma+Y_stick)*area_soma
            ;
        std::complex<double> J_soma = I_0*euler*Y_soma/dividend;
        values[0] = J_soma.real();
    }
};
"""

# C++ string for stick current
stick_code = """
class SomeExpr : public Expression
{
public:

  SomeExpr() : Expression(), omega(1.), tau(1), d(.002), R_m(3e4),
       R_i(150), lbda(1), l(1.), radius(.01), t(0), x_left(-.49),
      I_0(250e-9)  {}
```

```
  double omega, tau, d, R_m, R_i, lbda, l, Vs, radius, t, x_left,
      I_0;

  void eval(Array<double>& values, const Data& data) const
  {
        std::complex<double> s_squared(1,omega*tau);
        std::complex<double> s = std::sqrt(s_squared);
        double distance = data.x[0]-x_left;
        std::complex<double> euler(cos(omega*t),sin(omega*t));
        std::complex<double> Y_soma = (4.0*pi*pow(d,2)/R_m)*
            s_squared*pow(10,-2);
        std::complex<double> Y_stick = (pi*pow(d,1.5)/2.0*pow(R_i*
            R_m,-.5))*s*std::tanh(s*l/lbda)*pow(10,-1.5);
        std::complex<double> V_soma = I_0*euler/(Y_soma+Y_stick);
        std::complex<double> J_m = s_squared/R_m*(std::exp(s*
            distance/lbda)/(1.0+std::exp((2.0*l/lbda)*s))+std::exp
            (-s*distance/lbda)/(1.0+std::exp(-(2.0*l/lbda)*s)))*(
            V_soma*pow(10,-2));
        values[0] = J_m.real();
  }
};
"""

# Define soma current expression
J_soma = Expression(soma_code)
# Hand over parameters
J_soma.omega = omega
J_soma.tau = tau
J_soma.d = stick_diameter
J_soma.R_i = R_i
J_soma.R_m = R_m
J_soma.l = stick_length
J_soma.lbda = lbda
J_soma.t = 0
J_soma.area_soma = area_soma
J_soma.I_0 = I_0
J_soma.radius = radius

# Define stick current expression
J_stick = Expression(stick_code)
# Hand over parameters
J_stick.omega = omega
J_stick.tau = tau
J_stick.d = stick_diameter
J_stick.R_m = R_m
J_stick.R_i = R_i
J_stick.lbda = lbda
J_stick.l = stick_length
J_stick.radius = radius
J_stick.t = 0
J_stick.x_left = stick_x_left
J_stick.I_0 = I_0

# Define the constant scalar conductivity of saline
```

```
saline_conductivity = Expression('s', {'s':sigma_saline})

# Expression for conductivity in layers II/III
# Define the matrix elements first
s11 = Expression('s', {'s': sigma_II_III_parallel})
s12 = Constant(0.0)
s13 = Constant(0.0)
s21 = Constant(0.0)
s22 = Expression('s', {'s':sigma_II_III_perpendicular})
s23 = Constant(0.0)
s31 = Constant(0.0)
s32 = Constant(0.0)
s33 = Expression('s', {'s':sigma_II_III_perpendicular})
#Then define the conductivity tensor for the layer
layer_II_III_conductivity = as_matrix(((s11,s12,s13),(s21,s22,s23)
    ,(s31,s32,s33)))

# Expression for conductivity in layer IV
# Define the matrix elements first
s11 = Expression('s', {'s': sigma_IV_parallel})
s12 = Constant(0.0)
s13 = Constant(0.0)
s21 = Constant(0.0)
s22 = Expression('s', {'s':sigma_IV_perpendicular})
s23 = Constant(0.0)
s31 = Constant(0.0)
s32 = Constant(0.0)
s33 = Expression('s', {'s':sigma_IV_perpendicular})
#Then define the conductivity tensor for the layer
layer_IV_conductivity = as_matrix(((s11,s12,s13),(s21,s22,s23),(
    s31,s32,s33)))

# Expression for conductivity in layer V
# Define the matrix elements first
s11 = Expression('s', {'s': sigma_V_parallel})
s12 = Constant(0.0)
s13 = Constant(0.0)
s21 = Constant(0.0)
s22 = Expression('s', {'s':sigma_V_perpendicular})
s23 = Constant(0.0)
s31 = Constant(0.0)
s32 = Constant(0.0)
s33 = Expression('s', {'s':sigma_V_perpendicular})
#Then define the conductivity tensor for the layer
layer_V_conductivity = as_matrix(((s11,s12,s13),(s21,s22,s23),(s31
    ,s32,s33)))

# Expression for conductivity in layer VI
# Define the matrix elements first
s11 = Expression('s', {'s': sigma_VI_parallel})
s12 = Constant(0.0)
s13 = Constant(0.0)
s21 = Constant(0.0)
s22 = Expression('s', {'s':sigma_VI_perpendicular})
```

```
s23 = Constant(0.0)
s31 = Constant(0.0)
s32 = Constant(0.0)
s33 = Expression('s', {'s':sigma_VI_perpendicular})
#Then define the conductivity tensor for the layer
layer_VI_conductivity = as_matrix(((s11,s12,s13),(s21,s22,s23),(
    s31,s32,s33)))

# Time loop parameters, defined in terms of stimulation frequency
T = 1/f
dt = T/8.
t = 0

# Test and trial function
u = TrialFunction(V)
v = TestFunction(V)

# Defining the reference electrode
bc = DirichletBC(V, Constant(0), boundaries, 9)

# Write current information to file
ofile = open('current.dat', 'w') # open file for writing
ofile.write('f = %s kHz\n' % f)

file = File('phi.pvd')
while t<T:
    # Update time
    J_soma.t = t
    J_stick.t = t

    # Necessary adjustments for currents to obey Kirchhoff's
        current law
    # Write current (not electron current but contemporary) time
        to file
    ofile.write('t = %s ms\n' % t)
    # Calculate soma current [mA]
    soma_current = assemble(J_soma*ds(7), mesh=mesh,
        exterior_facet_domains=boundaries)
    # Write soma current to file
    ofile.write('Soma current = %s mA\n' % soma_current)
    # Calculate stick current before correction
    stick_current = assemble(J_stick*ds(8), mesh=mesh,
        exterior_facet_domains=boundaries)
    # Write stick current (before correction) to file
    ofile.write('Stick current = %s mA\n' % stick_current)
    # Calculate the electrode current at this instance
    I_e = I_0*cos(omega*t) # [mA], electrode current
    # Write the electrode current to file
    ofile.write('Electrode current = %s mA\n' % I_e)
    # Calculate the deviation from Kirchhoff's current law
    delta = I_e - stick_current - soma_current
    # Write the deviation to file
    ofile.write('I_e - I_soma - I_stick = %s mA\n' % delta)
```

```
    # Correction current
    # Calculate the new current amplitude to use for the stick
    I_0_corr = (I_e-soma_current)*I_0/stick_current
    # Write it to file
    ofile.write('Correction current = %s mA\n' % I_0_corr)

    # Update the stick current
    J_stick.I_0 = I_0_corr
    # Calculte the total stick current [mA]
    stick_current = assemble(J_stick*ds(8), mesh=mesh,
        exterior_facet_domains=boundaries)
    # Write it to file
    ofile.write('Stick current = %s mA\n' % stick_current)

    # Calculate the difference and write it to file
    delta = I_e - stick_current - soma_current
    # Write the new error, hopefully zero, to file
    ofile.write('I_e - I_soma - I_stick = %s mA\n' % delta)

    # Variational problem
    # Bilinear form
    a = saline_conductivity*inner(grad(u),grad(v))*dx(0) + inner(
        layer_VI_conductivity*grad(u),grad(v))*dx(1) + inner(
        layer_V_conductivity*grad(u),grad(v))*dx(2) + inner(
        layer_IV_conductivity*grad(u),grad(v))*dx(3) + inner(
        layer_II_III_conductivity*grad(u),grad(v))*dx(4)
    # Linear form
    L = Constant(0)*v*dx + J_soma*v*ds(7) + J_stick*v*ds(8)

    # Assemble the stiffness matrix
    A = assemble(a, exterior_facet_domains=boundaries,
        cell_domains=subdomains)
    # Assemble the RHS vector
    b = assemble(L, exterior_facet_domains=boundaries,
        cell_domains=subdomains)
    # Apply Dirichlet boundary condition
    bc.apply(A,b)
    # Declare a function to hold the solution
    phi = Function(V)
    # Solve the linear system
    solve(A, phi.vector(), b)

    # Export the current solution
    file << phi
    # Update the time
    t += dt

    # Reset stick current. THIS IS IMPORTANT!!
    J_stick.I_0 = I_0

# Close output file
ofile.close()
```

Two-Monopole Approximation

Mesh Generation

The file `create_mesh.py` from `TM_Impact_of_Saline` is shown below.

```
 #!/usr/bin/env python
from dolfin import *

# Outer box dimensions (millimeters are used as units for all
    length variables in the script!)
box_x_dim = 4; box_y_dim = 4; box_z_dim = 1

# Initial number of cells in each spatial direction (0.1 mm
    resolution)
nx = 40; ny = 40; nz = 10

# Tissue slice dimensions
tissue_x_dim = 2; tissue_y_dim = 2; tissue_z_dim = 0.3

# Ball dimension
radius = 0.01 # [mm]
separation = 1. # [mm]

# Ball height above array
height = .2 # [mm]

# Ball coordinates
left_ball_x = -separation/2.
left_ball_y = 0
left_ball_z = -box_z_dim/2. + height
left_ball_coor = Point(left_ball_x, left_ball_y, left_ball_z)

right_ball_x = separation/2.
right_ball_y = 0
right_ball_z = -box_z_dim/2. + height
right_ball_coor = Point(right_ball_x, right_ball_y, right_ball_z)

# Class representing the tissue
class Tissue(SubDomain):
    def inside(self, x, on_boundary):
        return -tissue_x_dim/2.<= x[0] <= tissue_x_dim/2. and -
            tissue_y_dim/2. <= x[1] <= tissue_y_dim/2. and -
            box_z_dim/2.<= x[2] <= -box_z_dim/2. + tissue_z_dim

# Class representing the left ball
class LeftBall(SubDomain):
    def inside(self, x, on_boundary):
        r = sqrt((x[0]-left_ball_x)**2+(x[1]-left_ball_y)**2+(x
            [2]-left_ball_z)**2)
        return r<1.5*radius
    def snap(self, x):
        r = sqrt((x[0]-left_ball_x)**2+(x[1]-left_ball_y)**2+(x
            [2]-left_ball_z)**2)
```

```
        if r<1.5*radius:
            x[0] = left_ball_x + (radius/r)*(x[0]-left_ball_x)
            x[1] = left_ball_y + (radius/r)*(x[1]-left_ball_y)
            x[2] = left_ball_z + (radius/r)*(x[2]-left_ball_z)

# Class representing the right ball
class RightBall(SubDomain):
    def inside(self, x, on_boundary):
        r = sqrt((x[0]-right_ball_x)**2+(x[1]-right_ball_y)**2+(x
            [2]-right_ball_z)**2)
        return r<1.5*radius
    def snap(self, x):
        r = sqrt((x[0]-right_ball_x)**2+(x[1]-right_ball_y)**2+(x
            [2]-right_ball_z)**2)
        if r<1.5*radius:
            x[0] = right_ball_x + (radius/r)*(x[0]-right_ball_x)
            x[1] = right_ball_y + (radius/r)*(x[1]-right_ball_y)
            x[2] = right_ball_z + (radius/r)*(x[2]-right_ball_z)

# Classes representing the outer boundaries (numbered from 1-6)
class Boundary1(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and abs(x[0]-box_x_dim/2.)<=DOLFIN_EPS
class Boundary2(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and abs(x[1]-box_y_dim/2.)<=DOLFIN_EPS
class Boundary3(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and abs(x[2]-box_z_dim/2.)<=DOLFIN_EPS
class Boundary4(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and abs(x[0]+box_x_dim/2.)<=DOLFIN_EPS
class Boundary5(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and abs(x[1]+box_y_dim/2.)<=DOLFIN_EPS
class Boundary6(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and abs(x[2]+box_z_dim/2.)<=DOLFIN_EPS

# Class representing the boundary of the left ball
class LeftBallBoundary(SubDomain):
    def inside(self, x, on_boundary):
        r = sqrt((x[0]-left_ball_x)**2+(x[1]-left_ball_y)**2+(x
            [2]-left_ball_z)**2)
        return on_boundary and r<1.5*radius

# Class representing the boundary of the left ball
class RightBallBoundary(SubDomain):
    def inside(self, x, on_boundary):
        r = sqrt((x[0]-right_ball_x)**2+(x[1]-right_ball_y)**2+(x
            [2]-right_ball_z)**2)
        return on_boundary and r<1.5*radius

# Class representing the part of the boundary to be grounded
```

```
class GroundBoundary(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and box_x_dim/2.-0.1 <= x[0] <=
            box_x_dim/2. and box_y_dim/2.-0.1<=x[1]<=box_y_dim/2.
            and abs(x[2]-box_z_dim/2.)<=.01

# Generation of the original mesh
mesh = Box(-box_x_dim/2.,-box_y_dim/2.,-box_z_dim/2.,box_x_dim/2.,
    box_y_dim/2.,box_z_dim/2.,nx,ny,nz)

# Print mesh information to the shell
print mesh

multiplier = [10,8,7,6,5,4, 4,3,2.5,2.2, 2,1.8,1.7,
    1.6,1.5,1.4,1.3,1.3]
num_refinements = len(multiplier)
for i in range(num_refinements):
        # Mark cells for refinement
    markers = MeshFunction("bool", mesh, mesh.topology().dim())
    markers.set_all(False)
    for cell in cells(mesh):
        if cell.midpoint().distance(left_ball_coor) < multiplier[i
            ]*radius or cell.midpoint().distance(right_ball_coor)
            < multiplier[i]*radius:
            markers[cell.index()] = True

    mesh = refine(mesh, markers)
    print mesh

# Extract the submesh, excluding the two holes
subdomains = MeshFunction('uint', mesh, mesh.topology().dim())
subdomains.set_all(0)
LeftBall().mark(subdomains,1)
RightBall().mark(subdomains,2)
mesh = SubMesh(mesh, subdomains, 0)

# New mesh --> new MeshFunction() instance must be defined
subdomains = MeshFunction('uint', mesh, mesh.topology().dim())
subdomains.set_all(0)
Tissue().mark(subdomains,1)

# Function for boundary marking
boundaries = MeshFunction('uint', mesh, mesh.topology().dim()-1)
boundaries.set_all(0)

# Mark the boundaries
Boundary1().mark(boundaries,1)
Boundary2().mark(boundaries,2)
Boundary3().mark(boundaries,3)
Boundary4().mark(boundaries,4)
Boundary5().mark(boundaries,5)
Boundary6().mark(boundaries,6)
GroundBoundary().mark(boundaries,9)
mesh.snap_boundary(LeftBall())
```

```
LeftBallBoundary().mark(boundaries,7)
mesh.snap_boundary(RightBall())
RightBallBoundary().mark(boundaries,8)

# Save the mesh and the markers to file. They will be used by the
    script 'simulation.py'
file = File('mesh.xml')
file << mesh
file = File('subdomains.xml')
file << subdomains
file = File('boundaries.xml')
file << boundaries
```

Simulation

The file `create_mesh.py` from `TM_Impact_of_Saline` is shown below.

```
 #!/usr/bin/env python
"""
simulation.py
TM_Impact_of_Saline
This Python script runs the simulations for a two-monopole
    approximation, testing the impact of saline
"""
from dolfin import *
import numpy

# Outer box dimensions (millimeters are used as units for all
    length variables in the script!)
box_x_dim = 4; box_y_dim = 4; box_z_dim = 1

# Initial number of cells in each spatial direction (0.1 mm
    resolution)
nx = 40; ny = 40; nz = 10

# Tissue slice dimensions
tissue_x_dim = 2; tissue_y_dim = 2; tissue_z_dim = 0.3

# Ball dimension
radius = 0.01 # [mm]
separation = 1. # [mm]

# Ball height above array
height = .2 # [mm]

# Ball coordinates
left_ball_x = -separation/2.
left_ball_y = 0
left_ball_z = -box_z_dim/2. + height
left_ball_coor = Point(left_ball_x, left_ball_y, left_ball_z)

right_ball_x = separation/2.
right_ball_y = 0
```

```
right_ball_z = -box_z_dim/2. + height
right_ball_coor = Point(right_ball_x, right_ball_y, right_ball_z)

# Conductivity
sigma_saline = 3. # S/m
sigma_tissue = .3 # S/m

# Current amplitude
I_0 = 1e-6 # [mA]

# Use the mesh, subdomains, and boundary markers generated by '
    create_mesh.py'. These files should reside in the folder
mesh = Mesh('mesh.xml')
boundaries = MeshFunction('uint', mesh, 'boundaries.xml')
subdomains = MeshFunction('uint', mesh, 'subdomains.xml')

# Function space for the trial and test functions
V = FunctionSpace(mesh, 'CG', 1)
# Function space for the step conductivity
V0 = FunctionSpace(mesh, 'DG', 0)

# Calculate the area of the two monopoles, and write them to the
    shell
area7 = assemble(Constant(1)*ds(7), mesh=mesh,
    exterior_facet_domains=boundaries)
area8 = assemble(Constant(1)*ds(8), mesh=mesh,
    exterior_facet_domains=boundaries)
print 'Area7 =', area7
print 'Area8 =', area8

# Define the conductivity function (uses numpy)
sigma = Function(V0)
sigma_values = [sigma_saline, sigma_tissue]
help = numpy.asarray(subdomains.values(), dtype= numpy.int32)
sigma.vector()[:] = numpy.choose(help, sigma_values)

# Write conductivity function to file (for verification)
file = File('sigma.pvd')
file << sigma

# Reference electrode
bc = DirichletBC(V, Constant(0), boundaries, 9)

# Define trial and test function
u = TrialFunction(V)
v = TestFunction(V)

# Membrane current
# Source
J7 = Expression('I_0A/area7A', {'I_0A':I_0, 'area7A':area7})
# Sink
J8 = Expression('-I_0A/area8A', {'I_0A':I_0, 'area8A':area8})

"""
```

```
First, the saline is taken into account, and the resulting
    potential field is saved to 'phi_fem_saline.pvd'
"""
# Taking saline into account
a = sigma*inner(grad(u),grad(v))*dx
L = Constant(0)*v*dx+ J7*v*ds(7)+J8*v*ds(8)

# Assemble stiffness matrix and RHS
A = assemble(a, exterior_facet_domains=boundaries)
b = assemble(L, exterior_facet_domains=boundaries)
# Apply essential boundary condition, i.e., the reference
    electrode
bc.apply(A,b)
# Define a function to hold the solution
phi_fem_saline = Function(V)
# Solve the problem and give it to phi_fem_saline
solve(A, phi_fem_saline.vector(), b)

# Export solution to file
file = File('phi_fem_saline.pvd')
file << phi_fem_saline

"""
Next, for comparison, we neglect saline, and save the potential
    field to 'phi_fem_bc.pvd'
It is named so because it DOES take the boundary conditions into
    account
"""

# Not taking saline into account
# Define variational problem
a = Expression('sigma', {'sigma':sigma_tissue})*inner(grad(u),grad
    (v))*dx
L = Constant(0)*v*dx+ J7*v*ds(7)+J8*v*ds(8)

# Assemble stiffness matrix and RHS
A = assemble(a, exterior_facet_domains=boundaries)
b = assemble(L, exterior_facet_domains=boundaries)
# Apply essential boundary condition
bc.apply(A,b)
# Define a function to hold the solution
phi_fem_bc = Function(V)
# Solve the problem and give it to 'phi_fem_bc.pvd'
solve(A, phi_fem_bc.vector(), b)

# Export the solution to file
file = File('phi_fem_bc.pvd')
file << phi_fem_bc

# Find the difference between the FEM solution and the infinite
    medium solution and send it to file
diff = Function(V)
diff.vector()[:] = phi_fem_saline.vector()[:]-phi_fem_bc.vector()
    [:]
```

```
file = File('diff.pvd')
file << diff
```

Bibliography

[1] COMSOL Multiphysics. http://www.comsol.com.

[2] FEniCS project. http://www.fenics.org/.

[3] Multi Channel Systems. http://www.multichannelsystems.com/.

[4] Neuron. http://www.neuron.yale.edu/neuron/.

[5] Paraview 3.8.0 64-bit. http://www.paraview.org/.

[6] Python Programming Language. http://www.python.org/.

[7] Visualization Toolkit. http://www.vtk.org/.

[8] Martin S. Alnæs and Anders Logg. UFL specification and user manual 0.3. http://www.fenicsproject.org/wiki/Documentation, 2010.

[9] Rembrandt Bakker, Ingo Bojak, and Dirk Schubert. Personal communication, 2010.

[10] Rembrandt Bakker, Dirk Schubert, Koen Levels, Gleb Bezgin, Ingo Bojak, and Rolf Kötter. Classification of cortical microcircuits based on micro-electrode-array data from slices of rat barrel cortex. *Neural Networks*, 22(8):1159–1168, October 2009.

[11] Mark F. Bear, Barry W. Connors, and Michael A. Paradiso. *Neuroscience*. Lippincott Williams & Wilkins, 2001.

[12] Claude Bédard, Helmut Kröger, and Alain Destexhe. Modeling extracellular field potentials and the frequency-filtering properties of extracellular space. *Biophysical Journal*, 86(3):1829–1842, 2004.

[13] Gyorgy Buzsaki. Large-scale recording of neuronal ensembles. *Nature Neuroscience*, 7(446 - 451), April 2004.

[14] Enric Claverol-Tinture and Jerome Pine. Extracellular potentials in low-density dissociated neuronal cultures. *Journal of Neuroscience Methods*, 117(1):13–21, 2002.

[15] Peter Dayan and L.F. Abbott. *Theoretical Neuroscience*. Computational Neuroscience series. MIT Press, 2001.

[16] Gaute T. Einevoll. Modeling of extracellular potentials recorded with multicontact electrodes. In *Proceedings of 7th Int. Meeting on Substrate-Integrated Microelectrodes*, 2010.

[17] Gaute T. Einevoll, Klas H. Pettersen, Anna Devor, Istvan Ulbert, Eric Halgren, and Anders M. Dale. Laminar population analysis: Estimating firing rates and evoked synaptic activity from multielectrode recordings in rat barrel cortex. *J Neurophysiol*, 97(3):2174–2190, 2007.

[18] Jonathan Erickson, Angela Tooker, Y.-C. Tai, and Jerome Pine. Caged neuron mea: A system for long-term investigation of cultured neural network connectivity. *Journal of Neuroscience Methods*, 175(1):1–16, 2008.

[19] Michael Fejtl, Alfred Stett, Wilfried Nisch, Karl-Heinz Boven, and Andreas Möller. On Micro-Electrode Array Revival: Its Development, Sophistication of Recording, and Stimulation. In Makoto Taketani and Michel Baudry, editors, *Advances in Network Electrophysiology Using Multi-Electrode Arrays*, pages 24–37. Springer US, 2006.

[20] U. Frey, U. Egert, F. Heer, S. Hafizovic, and A. Hierlemann. Microelectronic system for high-resolution mapping of extracellular electric fields applied to brain slices. *Biosensors and Bioelectronics*, 24(7):2191–2198, 2009.

[21] A. Gliere, C. Moulin, D. Barbier, S. Joucla, B. Yvert, P. Mailley, R. Guillemaud, et al. A new 3-D finite-element model based on thin-film approximation for microelectrode array recording of extracellular action potential. *IEEE Transactions on Biomedical Enginering*, 55(2):683–692, February 2008.

[22] Takakuni Goti, Rieko Hatanaka, Takeshi Ogawa, Akira Sumiyoshi, Jorge Riera, and Ryuta Kawashima. An evaluation of the conductivity profile in the somatosensory barrel cortex of wistar rats. *Journal of Neurophysiology*, 2010.

[23] David J. Griffiths. *Introduction to Electrodynamics*. Prentice Hall, 3rd edition, 1999.

[24] Espen Hagen. Personal communication, 2010.

[25] Johan Hake. Personal communication, 2010.

[26] Gary R. Holt. *A Critical Reexamination of Some Assumptions and Implications of Cable Theory in Neurobiology*. PhD thesis, California Institute of Technology, 1998.

[27] Gary R. Holt and Christof Koch. Electrical interactions via the extracellular potential near cell bodies. *Journal of Computational Neuroscience*, 6:169–184, 1999. 10.1023/A:1008832702585.

[28] Eugene Izhikevich. *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting*. Computational Neuroscience series. The MIT Press, 2007.

[29] Christof Koch. *The Quest For Consciousness*. Roberts & Company, 2004.

[30] Hans Petter Langtangen. *Computational Partial Differential Equations*. Texts in Computational Science and Engineering. Springer, 2003.

[31] Hans Petter Langtangen. *Python Scripting for Computational Science*. Texts in Computational Science and Engineering. Springer, 3rd edition, 2007.

[32] Hans Petter Langtangen. A FEniCS tutorial. `http://www.fenicsproject.org/doc/`, 2010.

[33] A. Lehmenkühler, E. Syková, J. Svoboda, K. Zilles, and C. Nicholson. Extracellular space parameters in the rat neocortex and subcortical white matter during postnatal development determined by diffusion analysis. *Neuroscience*, 55(2):339–351, 1993.

[34] Henrik Lindén. *Modeling and analysis of extracellular field potentials in the brain*. PhD thesis, Norwegian University of Life Sciences, 2010.

[35] Henrik Lindén, Klas H. Pettersen, and Gaute T. Einevoll. Intrinsic dentritic filtering gives low-pass power spectra of local field potentials. *Journal of Computational Neuroscience*, May 2010.

[36] Anders Logg, Garth N. Wells, et al. DOLFIN. `http://www.fenicsproject.org/dolfin/`.

[37] Shawn Means, Alexander J. Smith, Jason Shepherd, John Shadid, John Fowler, Richard J.H. Wojcikiewicz, Tomas Mazel, Gregory D. Smith, and Bridget S. Wilson. Reaction diffusion modeling of calcium dynamics with realistic er geometry. *Biophysical Journal*, 91(2):537–557, 2006.

[38] Kenneth Moreland. *The ParaView Tutorial*, 3.8 edition, 2010.

[39] Philip Nelson. *Biological Physics*. Freeman, 2008.

[40] Ernst Niebur. Neuronal cable theory. *Scholarpedia*, 3(5):2674, 2008.

[41] Paul L. Nunez and Ramesh Srinivasan. *Electric fields of the brain*. Oxford University Press, 2006.

[42] Klas H. Pettersen. Personal communication, 2010.

[43] Klas H. Pettersen and Gaute T. Einevoll. Amplitude variability and extracellular low-pass filtering of neuronal spikes. *Biophysical Journal*, 2008.

[44] Klas H. Pettersen, Henrik Lindén, Anders M. Dale, and Gaute T. Einevoll. Extracellular Spikes and Current-Source density. In R. Brette and A. Destexhe, editors, *Handbook of Neural Activity Measurements*. Cambridge University Press, 2010.

[45] J. Pine. Neurochip. *Scholarpedia*, 3(10):7766, 2008.

[46] Jerome Pine. A History of MEA Development. In Makoto Taketani and Michel Baudry, editors, *Advances in Network Electrophysiology Using Multi-Electrode Arrays*, pages 3–23. Springer US, 2006.

[47] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C++*. Cambridge University Press, 3rd edition, 2007.

[48] W. Rall. Rall model. *Scholarpedia*, 4(4):1369, 2009.

[49] Steve Roensch. Finite element analysis: Post-processing. `http://www.finiteelement.com/feawhite4.html`.

[50] Werner Scholz, Josef Fidler, Thomas Schrefl, Dieter Suess, Rok Dittrich, Hermann Forster, and Vassilios Tsiantos. Scalable parallel micromagnetic solvers for magnetic nanostructures. *Comp. Mat. Sci*, 28:366–383, 2003.

[51] Sébastien and Yvert Blaise Joucla. Improved focalization of electrical microstimulation using microelectrode arrays: A modeling study. *PLoS ONE*, 4(3):e4828, 03 2009.

[52] Jonathan Shewchuk. Triangle: A two-dimensional quality mesh generator and delaunay triangulator. `http://www.cs.cmu.edu/~quake/triangle.html`.

[53] Hang Si. Tetgen: A quality tetrahedral mesh generator and a 3D Delaunay triangulator. `http://tetgen.berlios.de/`.

[54] Ralph J. Smith. *Electronics - Circuits & Devices*. John Wiley & Sons, 3rd edition, 1987.

[55] Gilbert Strang. *Linear Algebra and its Applications*. Thomson Brooks/Cole, 4th edition, 2006.

[56] Joakim Sundnes, Glenn Terje Lines, Xing Cai, Bjørn Frederik Nielsen, Kent-Andre Mardal, and Aslak Tveito. *Computing the Electrical Activity in the Heart*. Monographs in Computational Science and Engineering. Springer, 1st edition, 2006.

[57] Paul A. Tipler and Gene Mosca. *Physics for Scientist and Engineers*, volume 2. W. H. Freeman and Company, 6th edition, 2008.

[58] Eric W. Weisstein. Polynomial - From MathWorld - A Wolfram Web Resource. http://mathworld.wolfram.com/Polynomial.html.

[59] Eric W. Weisstein. Simplex - From MathWorld - A Wolfram Web Resource. http://mathworld.wolfram.com/Simplex.html.

[60] C. Wolters and J. C de Munck. Volume conduction. *Scholarpedia*, 2(3):1738, 2007.