Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

D I P L O M A R B E I T

Simulation von Schaltkreisen im menschlichen Rückenmark

Ausgeführt am Institut für
Analysis und Scientific Computing
der Technischen Universität Wien

unter der Anleitung von
Ao.Univ.Prof. Dipl.-Ing. Dr.sc.med Dr.techn. Dr.rer.nat Frank Rattay

durch

Astrid Bös
Matrikelnummer: 0325705
Leystrasse 115-117/5/2
1200 Wien

September 2010

Kurzfassung

In meiner Diplomarbeit habe ich mich vor allem auf die Stimulation von Rhythm Generator Neuronen konzentriert. Diese Neuronen sind wichtig da unter anderem durch sie der Bewegungsapparat gesteuert wird. Die menschlichen Nervenfasern sind natürlich sehr komplex aufgebaut und bestehen aus vielen einzelnen Nervenzellen. Ich habe diese sehr komplexe Vernetzung von Nervenzellen auf ein sehr einfaches Modell reduziert, in dem nur mehr 10 einzelne Nervenzellen direkt verbunden sind.

Zunächst habe ich mein Modell in NEURON implementiert und einen lang anhaltenden gleichbleibenden Impuls appliziert. Dieser konstante Impuls entspricht der Steuerung des Bewegungsapparates. Bei diesen Simulationen habe ich die wichtige Bedeutung des persistenten Natrium-Kanals erkannt. Im nächsten Schritt habe ich den lang anhaltenden Impuls durch kurze Impulse, von einer Frequenz zwischen 25Hz und 50Hz, ersetzt. Diese kurzen Impulse bewirken in der Realität unwillkürliche, schrittähnliche Bewegungen in den Beinen von Propanden mit kompletter Querschnittsläsion.

Abstract

In my thesis I have focused mainly on the stimulation of rhythm generator neurons. These neurons are important, because the locomotor system is more or less controlled by them. The human nerve fibers are very complex and consist of many individual nerve cells. I have reduced the very complex network of neurons to a simple model in which only 10 individual nerve cells are connected directly.

First, I implemented my model in NEURON and applied a prolonged steady pulse. This constant impulse corresponds to the control of the locomotor system. In these simulations, I detected the important significance of the persistent sodium channel. In the following step I replaced the long lasting pulse by short pulses of a frequency between 25Hz and 50Hz. These short pulses produce automatic stepping-like movements in the legs of humans with complete spinal cord injury.

Danksagung

Ich möchte mich bei Professor Frank Rattay dafür bedanken, dass er mir dieses interessante Thema vermittelt und mich im weiteren Verlauf meiner Arbeit unterstützt hat. Des Weiteren möchte ich mich auch bei Simon Danner für seine Unterstützung bedanken.

Ein besonderer Dank gilt meiner Familie und meinen Freunden, die mir in den Jahren meines Studiums immer zu Seite gestanden haben.

Inhaltsverzeichnis

1 Einleitung **6**

2 Biologische Grundlagen **8**

2.1 Nervenzelle 8

2.1.1 Aufbau der Nervenzelle 8

2.1.2 Zellmembran und Membranpotential 10

2.1.3 Weiterleitung des Aktionspotentials 11

2.2 Hodgkin-Huxley Gleichungen 14

3 NEURON **17**

3.1 Einführung mittels eines einfachen Beispiels 18

3.2 Benutzeroberfläche von NEURON 22

3.3 NMODL 26

3.3.1 NMODL-Beispiel: lokaler Shunt 27

3.3.2 NMODL-Beispiel: besonderer Kaliumkanal in HH 29

4 Stimulation mittels eines langen Impulses **36**

4.1 Erklärung des Modells 36

4.2 Persistenter Natrium-Kanal 41

4.3 Ergebnisse der Simulation 42

5 Stimulation mittels kurzen Implusen **50**

5.1 Erklärung des Modells 50

5.2 Ergebnisse der Simulationen 52

5.2.1 Gleiche Impulsintensität 52

5.2.2 Unterschiedliche Impulsintensität 56
5.2.3 Unterschiedliche Frequenz 59
5.2.4 Zusätzlicher konstanter langer Impuls 62

6 Zusammenfassung und Vorschau **65**

A Source-Codes **67**

Literaturverzeichnis **85**

Webverzeichnis **88**

Kapitel 1

Einleitung

Die elektrische Stimulation des unteren (lumbosakralen) Rückenmarks von Probanden mit kompletter Querschnittsläsion mittels rückenmarksnahen Implantaten kann Aktivitäten in der gelähmten Beinmuskulatur generieren. [Hof09, S.2]

Viele Forschungsarbeiten haben sich schon mit diesem Thema auseinandergesetzt. Der Umfang hierbei ist beinahe grenzenlos, da einerseits in der Technik neue Ideen entwickelt und andererseits in der medizinschen Forschung immer wieder neue Entdeckungen gemacht werden.

Mit meiner Diplomarbeit kann ich natürlich nur einen sehr kleinen Teil zu den Forschungen hinzufügen. Ein wichtiger Bestandteil in der Weiterleitung von Reizen in Nervenzellen, die den Bewegungsapparat steuern, ist der persistente Natrium-Strom. Ziel meiner Diplomarbeit ist es die bereits vorhanden Ergebnisse mittels Simulationen in dem Programm NEURON zu belegen und dabei die Wichtigkeit dieses Stromes zu unterschreichen.

Im Kapitel 2 werden die biologischen Grundlagen vorgestellt. Hierbei wird vor allem auf die Nervenzelle und die Hodgkin-Huxley Gleichungen eingegangen. Diese dienen zur Beschreibung der Weiterleitung eines Aktionspotentials in einer Nervenzelle. Kapitel 3 befasst sich mit dem Programm NEURON. Die Simulationen in dieser Diplomarbeit werden in NEURON durchgeführt und daher wird zuerst das Programm erklärt. In den Kapitel 4 und 5 wird

das Modell beschrieben und die Ergebnisse der Simulationen angeführt. Diese werden nochmal kompakt in Kapitel 6 zusammengefasst. Die Source-Codes des Modells sind in Anhang A zu finden.

Kapitel 2

Biologische Grundlagen

Um das Modell aus Kapitel 4 besser zu verstehen werden einige biologische Grundlagen benötigt, die ich in diesem Kapitel vorgestellt möchte.

2.1 Nervenzelle

Im folgenden Teil meiner Diplomarbeit werde ich die wichtigsten Bestandteile einer Nervenzelle (Neuron) erklären.

2.1.1 Aufbau der Nervenzelle

Die drei wichtigsten Bestandteile einer Nervenzelle sind der Zellkörper (Soma), die Dendriten und das Axon, wobei die beiden letzteren Fortsätze sind, die an den Zellkörper anschließen. Normalerweise gibt es nur ein Axon in einer Nervenzelle, wobei dieses bis zu 1m lang sein kann. Es verbindet verschiedene Teile des Nervensystems. Die Dendriten hingegen sind wesentlich kürzer und dicker als ein Axon. Es gibt normalerweise mehrere Dendriten pro Nervenzelle, wobei diese wiederum sehr verzweigt sind.

Die Aufgabe der Dendriten ist eine Erregung von anderen Nervenzellen zu übernehmen und an den Zellkörper weiterzuleiten, wobei hingegen das Axon die Erregung vom Zellkörper weiterleitet und an benachbarte Nervenzellen über Synapsen abgibt. Die Dendriten folgen bei der Weiterleitung dem Prinzip

„je mehr desto mehr“, womit gemeint ist, dass die ankommenden Erregungen an dem Ort, wo sie zusammenkommen aufsummiert, werden.

Bei den Erregungen, die die Dendriten aufnehmen und weiterleiten, kann man weiters zwischen erregenden und hemmenden Signalen unterscheiden. Das Axon gibt die eingehenden Signale weiter und folgt dabei dem Alles-oder-Nichts-Gesetz. Dies besagt, dass ein Signal (hier auch genannt Aktionspotential) weitergeleitet wird, wenn die aufsummierten eingehenden Signale einen gewissen Schwellenwert überschreiten. Diese Entscheidung erfolgt am Beginn des Axons, dem sogenannten Axonhügel, der das Axon mit dem Zellkörper verbindet.

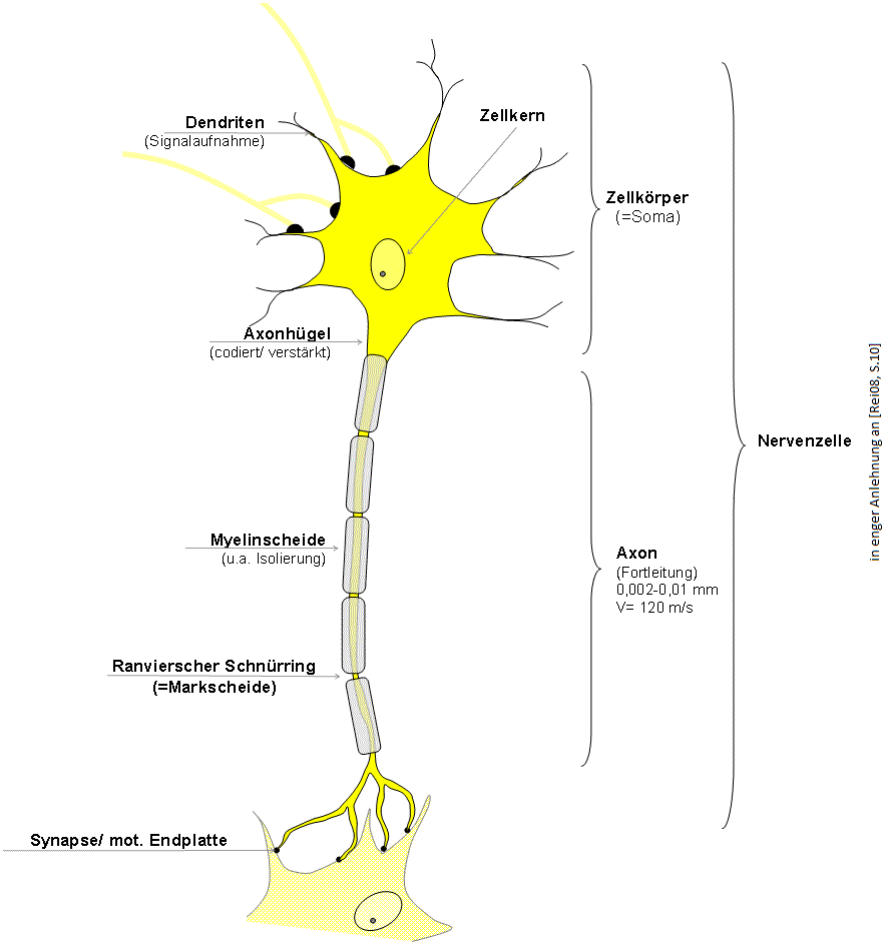

Abbildung 2.1: Aufbau der Nervenzelle

Damit das Axon isoliert ist, befindet sich darum eine Myelinhülle. Diese wird durch die Ranvier-Schnürringe unterteilt, deren einzelne Teile nennt man

Myelinscheiden. Durch diese Myelinscheiden kommt es zu einer höheren Fortleitungsgeschwindigkeit des Aktionspotentials innerhalb des Axons. Dies wird bewirkt, da die Myelinhülle die Membrankapazität im Axon verringert.

Das Axon verbindet die Nervenzellen mittels den Synapsen miteinander. An diesen Synapsen kommt es also zur Impulsübertragung von einer Nervenzelle zur nächsten.

2.1.2 Zellmembran und Membranpotential

Die Zellmembran trennt das Innere einer Zelle von ihrer Umgebung, wobei das Innere eine Salzlösung enthält und somit viele K^+-Ionen vorhanden sind, jedoch nur wenige Na^+- und Cl^--Ionen. Außerhalb einer Zelle ist dies hingegen genau umgekehrt, es gibt viele Na^+- und Cl^--Ionen aber nur wenige K^+-Ionen.

In einer Zellmembran gibt es zum Austausch mit der Umgebung Ionenkanäle (zum Beispiel K^+- und Na^+-Kanäle). Diese Kanäle sind meist selektiv permeabel, dies bedeudet, dass zum Beispiel der K^+-Kanal nur für K^+-Ionen durchlässig ist, andere Ionen jedoch (auch mit gleicher Ladung) nicht durchgelassen werden.

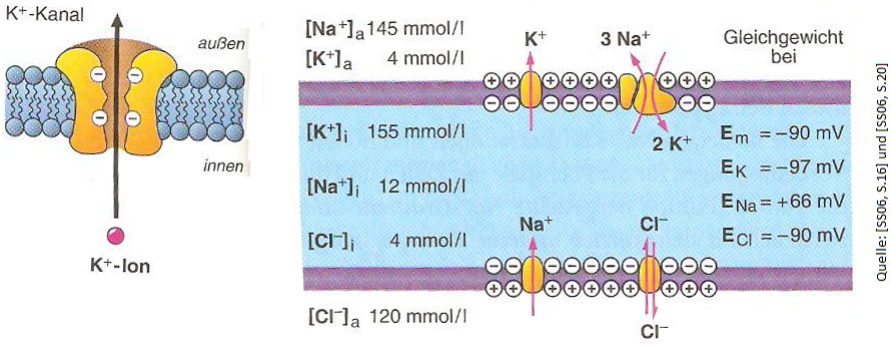

Abbildung 2.2: Links: Skizze eines K-Kanals in einer Zellmembran; **rechts**: Skizze einer Zellmembran mit verschiedenen Kanälen und den dazugehörigen Gleichgewichtspotentialen

Da das Innere und das Äußere einer Zelle normalerweise unterschiedlich geladen sind kommt es zu einem bestimmten Membranpotential. Wenn man dieses Membranpotential genauer betrachtet merkt man, dass es sich hierbei um

die Differenz zwischen den unterschiedlichen Potentialen handelt. Das Membranruhepotential (welches besteht wenn sich die Ionen in einer Ruhephase befinden) liegt zum Beispiel bei Muskelzellen bei -90mV. Dieses Ruhepotential steht natürlich in engem Zusammenhang mit den Gleichgewichtspotentialen der verschiedenen Ionen – siehe Abbilidung 2.2. Da das K^+-Gleichgewichtspotential ein wenig positiver ist als das Membranruhepotential, strömen K^+-Ionen durch den K^+-Kanal aus.

Die Gleichgewichtspotentiale kann man durch die Nernst-Gleichung berechnen:

$$E_{\text{ion}} = \frac{RT}{zF} \ln \frac{[\text{Ion}]_{\text{außen}}}{[\text{Ion}]_{\text{innen}}}$$

Hierbei entspricht R der Gaskonstanten, T der absoluten Temperatur, z der Ladungszahl des Ions, F der Faradaykonstanten und [Ion] der entsprechenden Ionenkonzentration.

Bei den Ionenkanälen muss man des Weiteren darauf achten, dass sie nicht immer geöffnet sind. Man spricht von zwei Zuständen: offen und geschlossen. Die unterschiedlichen Ionenkanäle haben unterschiedliche Verhaltensmuster bezüglich ihres Zustandes. Mittels des englischen Begriffs „Gating“ werden folgende Fragen beschrieben: Warum öffnet sich ein Ionenkanal? Wie lange bleibt dieser offen und ähnliche Fragen. Die meisten Ionenkanäle werden durch das Membranpotential gesteuert. So sind zum Beispiel Na^+-Kanäle, während ein Membranruhepotential vorliegt, nicht durchlässig und werden erst durch eine Depolarisation geöffnet. Andere Ionenkanäle werden unter anderem durch physikalische Einflüsse (wie zum Beispiel Druck oder Vibration) aktiviert.

2.1.3 Weiterleitung des Aktionspotentials

Jede Zuckung in unserem Körper beginnt mit einem Aktionspotential in einer Muskelzelle. Zur Untersuchung von Aktionspotentialen und deren Weiterleitung verwendet man oftmals die Riesenfasern von Tintenfischen. Hierbei werden je eine Messelektrode und eine Reizelektrode in die Faser eingeführt und man bringt zu den jeweiligen Elektroden außerhalb der Faser noch Be-

zugselektroden an, welche mit einem Reizgenerator verbunden sind. Mittels eines Oszillographen kann man im weiteren Versuchsverlauf die Reizspannung ablesen.

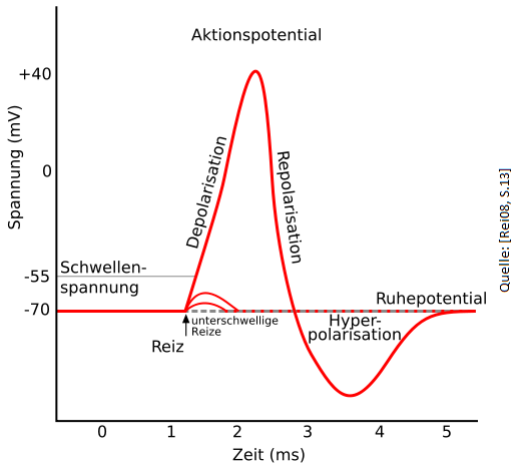

Abbildung 2.3: Skizzierung der verschiedenen Phasen eines Aktionspotentials

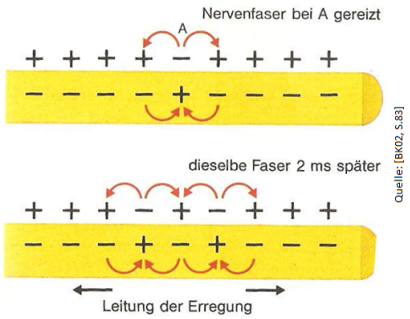

Abbildung 2.4: Schematische Darstellung von Weiterleitung des AP

Jede Zelle hat ein gewisses Ruhepotential (in Nervenzellen beträgt dies normalerweise Werte zwischen -90mV und -70mV), welches sich durch ein Aktionspotential innerhalb kürzester Zeit ändern kann. Das veränderte Potential kann einen Wert bis zu $+30$mV annehmen. Dieser Sprung dauert gewöhnlicherweise weniger als 1ms, die sogenannte Depolarisationsphase. Unter normalen Umständen dauert die Rückkehr zum Ruhepotential, die sogenannte Repolarisation, ebenso weniger als 1ms – siehe Abbildung 2.3.

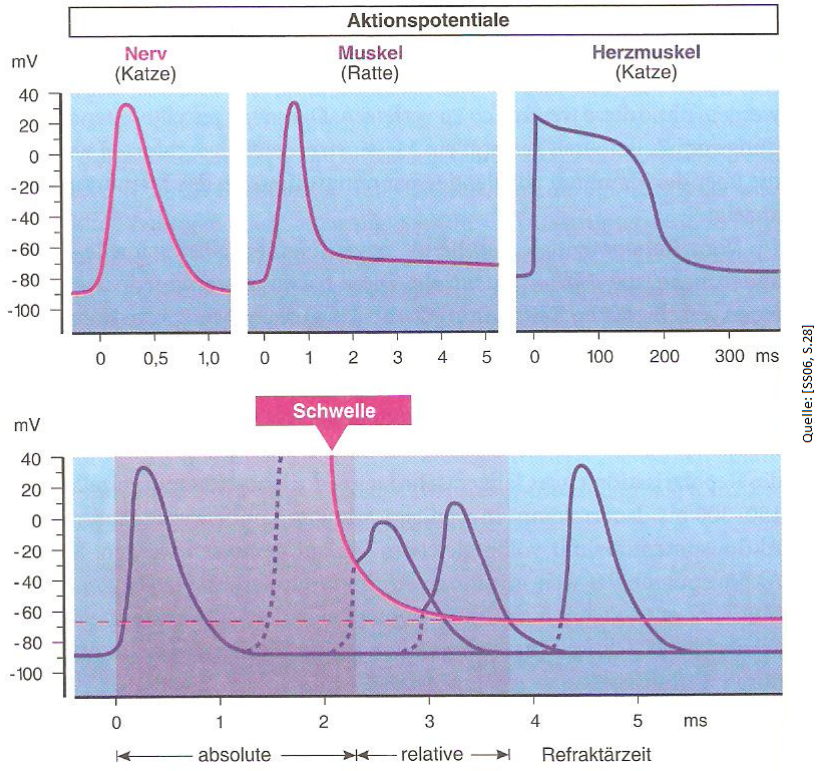

Abbildung 2.5: Oben: Aktionspotentiale bei einer Katze und Ratte – hierbei kann man erkennen, dass der Verlauf des APs von der Nervenart abhängt; **unten**: Skizzierung der Refraktärzeit

Die Weiterleitung eines Aktionspotentials kann man folgendermaßen erklären: Wenn durch Reizung einer bestimmten Stelle (A) ein Aktionspotential entsteht, grenzen an dieser Stelle positive und negative Ladungen ohne trennende Membran aneinander. Da sich gegensätzliche Ladungen anziehen, entstehen Ionenströme (Ausgleichsströmchen). Diese erniedrigen das Membranpotential der benachbarten Stellen. Ist die Nachbarstelle unter den Schwellenwert depolarisiert, entsteht auch hier ein Aktionspotential – siehe Abbildung 2.4. [BK02, S.82]

Während ein Ruhepotential vorherrscht befinden sich alle Membranströme im Gleichgewicht. Durch eine Veränderung des Potentials um bereits 20mV wird ein Aktionspotential ausgelöst und es beginnt ein depolarisierender Strom (dieser wird auch Reiz genannt). Der Wert der überschritten werden muss damit ein Aktionspotential ensteht heißt Schwelle.

Wird kurz nach einem Aktionspotential die Membran depolarisiert, so kann die normale Schwelle weit überschritten werden ohne dass ein Aktionspotential ausgelöst wird. Für die nächsten 1-2ms ist kein Aktionspotential auslösbar, das heißt die Zelle ist absolut refraktär. Danach kehrt die Schwelle von einem hohen Wert zum Ausgangspotential zurück. In der relativen Refraktärphase können nur im Vergleich zum Normalwert überschwellige Depolarisationen Aktionspotentiale auslösen, deren Amplituden vermindert sind – siehe Abbildung 2.5. [SS06, S.27]

2.2 Hodgkin-Huxley Gleichungen

1952 entwickelten A.L. Hodgkin und A.F. Huxley ein Modell zur Beschreibung der Weiterleitung eines Aktionspotentials in einem Neuron. Die beiden Wissenschaftler führten ursprünglich Versuche an einem Riesenaxon eines Tintenfisches durch und konnten dadurch die Entstehung von Aktionspotentialen beschreiben.

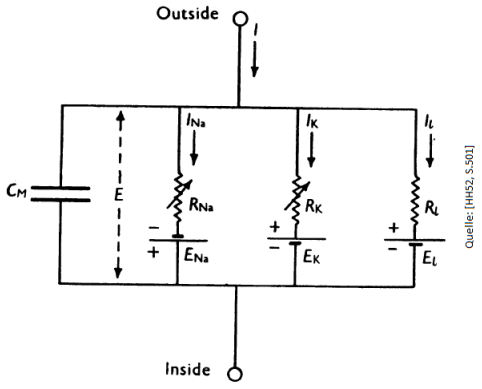

Abbildung 2.6: Schaltkreis einer Membran mit $R_{\mathrm{Na}} = 1/g_{\mathrm{Na}}$, $R_{\mathrm{K}} = 1/g_{\mathrm{K}}$ und $R_{\mathrm{L}} = 1/g_{\mathrm{L}}$

Man kann den Gesamtmembranstrom als Summe von Kalium-, Natrium- und Leckstrom und den externen Strom auffassen. Folglich kann das Membranpotential durch folgende Differentialgleichung beschrieben werden:

$$C\frac{dV}{dt} = I_{\text{ext}} - I_{\text{K}} - I_{\text{Na}} - I_{\text{L}}$$

Die einzelnen Ströme sind die Differenz des Membranpotentials mit dem jeweiligen Gleichgewichtspotential. Diese Differenz wird multipliziert mit der dazugehörigen Leitfähigkeit:

$$I_{\text{K}} = g_{\text{K}}(V - V_{\text{K}})$$

$$I_{\text{Na}} = g_{\text{K}}(V - V_{\text{Na}})$$

$$I_{\text{L}} = g_{\text{K}}(V - V_{\text{L}})$$

Da die Leitfähigkeiten bei Kalium- und Natriumströmen zeitabhängig sind, sind diese für die Entstehung eines Aktionspotentials verantwortlich. Durch das Einführen von sogenannten Gating-Variablen (n, m und h) können der Kalium- und Natriumstrom folgendermaßen beschrieben werden:

$$I_{\text{K}} = \overline{g}_{\text{K}} n^4 (V - V_{\text{K}})$$

$$I_{\text{Na}} = \overline{g}_{\text{Na}} m^3 h (V - V_{\text{Na}})$$

Wobei für n, m und h folgende Differentialgleichungen gelten:

$$\frac{dn}{dt} = \alpha_{\text{n}}(V)(1-n) - \beta_{\text{n}}(V)n$$
$$\frac{dm}{dt} = \alpha_{\text{m}}(V)(1-m) - \beta_{\text{m}}(V)m$$
$$\frac{dh}{dt} = \alpha_{\text{h}}(V)(1-h) - \beta_{\text{h}}(V)h$$

Die α- und β-Funktionen sind folgendermaßen definiert:

$$\alpha_{\mathrm{n}} = 0{,}01\frac{V+10}{\exp(0,1V+1)-1} \qquad \beta_{\mathrm{n}} = 0{,}125\ \exp(\frac{V}{80})$$

$$\alpha_{\mathrm{m}} = 0{,}1\frac{V+25}{\exp(0{,}1V+2{,}5)-1} \qquad \beta_{\mathrm{m}} = 4\ \exp(\frac{V}{18})$$

$$\alpha_{\mathrm{h}} = 0{,}07\ \exp(\frac{V}{20}) \qquad \beta_{\mathrm{h}} = \frac{1}{\exp(0{,}1V+3)-1}$$

Die zuletzt genannten Funktionen sind mit Hilfe des Verhaltens des Riesenaxons eines Tintenfischen empirisch bestimmt worden.

Kapitel 3

NEURON

In diesem Kapitel werde ich das Programm NEURON genauer vorstellen. Dies ist ein Programm mit dem vor allem Neuronen und ganze Neuronennetzwerke simuliert werden können. Diese Simulationsumgebung wurde an der Yale-Universität von Nicholas T. Carnevale und Michael L. Hines entwickelt und ist über das Internet frei zugänglich.

Bevor ein reales Problem in einem Programm implementiert werden kann, muss man sich davon ein genaues Modellkonzept machen. Dabei wird versucht, sich auf das Wichtige zu konzentrieren und das Unwichtige wegzulassen. Es kann kein reales Problem genauestens modelliert werden, da dies jeden Computer überfordern würde, also muss man sich auf das Wesentliche beschränken und somit das Modell möglichst einfach gestalten.

Im Programm NEURON kann mit bereits vorhandenen Nervenmodellen gearbeitet werden und weiters kann man diese nach belieben verändern. Des Weiteren ist es möglich neue Konzepte mittels NMODL einzubinden. Neben den vorgefertigten Beispielen für Nervenmodelle, die bereits im Programm vorhanden sind, findet man im Internet viele weitere Nervenmodelle.

3.1 Einführung mittels eines einfachen Beispiels

Ich werde versuchen anhand eines einfachen Modells das Programm vorzustellen. Hierzu werde ich folgende Darstellung eines Neurons verwenden.

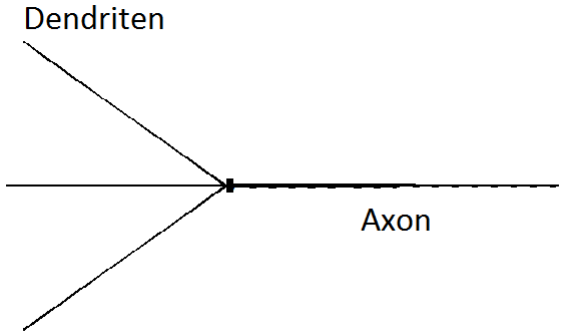

Abbildung 3.1: Skizze des einfachen Modells: Denriten – Soma – Axon

Hierbei handelt es sich um ein Neuron, das aus einem Zellkörper, einem Axon auf der rechten Seite und drei Dendriten auf der linken Seite besteht. Der Zellkörper ist kugelförmig und hat einen Durchmesser von 18,8μm. Die Länge des Axons beträgt 815,3μm und der Durchmesser bei der Anschlussstelle zum Zellkörper 5,6μm und am Ende 3μm. Die Dendriten haben eine Länge von 549,1μm und einen Durchmesser von 2μm. Das Neuron wird durch eine Elektrode stimuliert, die in die Mitte des Axons eingeführt wird.

Nun beginnen wir mit dem Erstellen unseres Modells. Zuerst müssen wir mittels der `create`-Funktion die drei Teile unseres Neurons (soma entspricht dem Zellkörper, dend entspricht den Dentriten) definieren. Danach verbinden wir diese mittels der `connect`-Funktion. Jedem Bestandteil (Segment) des Neurons ist ein Positionsparameter zugeteilt, der zwischen 0 und 1 variiert. Mittels der `connect`-Funktion kann genau bestimmt werden, an welcher Position die einzelnen Teile verbunden werden, üblicherweise werden sie dies an den Stellen 0 und 1. Um nicht jeden einzelnen Dendriten mit dem Zellkörper zu verbinden, habe ich eine `for`-Schleife benutzt. Das Programm weist den drei Dendriten die Positionen 0, 1 und 2 zu, daher läuft die Schleife von 0 bis 2.

```
create soma, dend[3], axon
for i = 0, 2 {connect dend[i](1), soma(0)}
connect soma(1), axon(0)
```

Als nächstes werden den einzelnen Segmenten die jeweiligen anatomischen und biophysikalischen Eigenschaften zugewiesen. Hierzu gibt es mehrere wichtige Variablen, die ich in den folgenden Tabellen zusammenfasse. Die Variablen kann man in Sektionsvariablen und Bereichsvariablen unterscheiden. Die Bereichsvariablen sind abhängig von ihrer Position im Segment, man muss ihnen also einen Positionsparameter zuweisen. Die Sektionsvariablen hingegen sind unabhängig von ihrer Position und gelten somit für das ganze Segment.

Name	Bedeutung	Einheit
`L`	Sektionslänge	[μm]
`Ra`	cytoplasmischer Widerstand	[Ωcm]
`nseg`	Anzahl der Unterteilungen	[1] - natürliche Zahl

Tabelle 3.1: Sektionsvariable

Name	Bedeutung	Einheit
`diam`	Durchmesser	[μm]
`cm`	Membrankapazität	[μF/cm^2]
`v`	Membranpotential	[mV]
`ina`	Natriumstrom	[mA/cm^2]

Tabelle 3.2: Bereichsvariable

Ich werde die Zuweisung der Eigenschaften an Hand des Axons vorstellen.

```
axon {nseg = 50
     diam(0:1) = 5.6:3
     L = 815.3
     Ra = 123
     insert hh
     }
```

Mittels der `insert`-Funktion kann festgelegt werden, welche Kanäle für das Membranpotential verwendet werden sollen. Hierfür sind bereits zwei Möglichkeiten in NEURON vorgegeben – `pas` und `hh`. Bei den ersten Kanälen sind die passiven Ionenkanäle gemeint und bei `hh` wird auf die Hodgkin-Huxley Kanäle zurückgegriffen. Man kann mit `insert` nicht nur die bereits in NEURON implementierten Modelle verwenden, sondern auch Modelle die man selbst in NMODL proprammiert hat. Hierbei ist an Stelle von `hh` oder `pas` der Namen der Datei einzugegeben.

Zum Schluss wird noch eine Elektrode die das Neuron stimuliert gebraucht. Dies lässt sich in NEURON folgendermaßen implementieren:

```
objectvar stim
axon stim = new IClamp(0.5)
stim.del = 2 // Verzögerung [ms]
stim.dur = 5 // Dauer der Stimulation [ms]
stim.amp = 40 // Amplitude [nA]
```

Der Elektrode wird in die Mitte des Axon eingeführt. Dies kann auch variiert werden indem man der Funktion `IClamp` einen Parameter zwischen 0 und 1 übergibt und somit die Position in dem Segment verändert. Eine weitere Möglichkeit für eine Stimulation ist `AlphaSynapse`. Bei dieser Stimulation kann man ebenfalls die Verzögerung bestimmen, sowie weitere Eigenschaften der Leitfähigkeit.

Um ein Programm in NEURON zu verfassen, schreibt man dieses am besten in einem Texteditor und speichert es unter `name.hoc` ab. Mittels Doppelklick

auf diese Datei wird das Programm unter NEURON geöffnet und man kann mit der Simulation beginnen. Im Folgenden füge ich die vorangegangenen Schritte zu einem Programm zusammen:

```
 loadfile(nrngui.hoc") // hiermit wird NEURON geöffnet
ndend = 3 // Anzahl der Dendriten
create soma, dend[ndend], axon
access soma // Festlegung der Standard-Sektion
//Festlegung der Eigenschaften:
soma {nseg = 1
     diam = 18.8
     L = 18.8
     Ra = 123
     insert hh
     }
axon {nseg = 50
     diam(0:1) = 5.6:3
     L = 815.3
     Ra = 123
     insert hh
     }
for i = 0, ndend-1 dend[i] {nseg = 5
                            diam = 2
                            L = 549.1
                            Ra = 123
                            insert pas
                            }
//Verbindung der einzelnen Segmente:
for i = 0, ndend - 1 {connect dend[i](1), soma(0)}
connect soma(1), axon(0)
```

```
//Setzen der Stimulation mittels einer Prozedur:
objectvar stim
proc stimver() {
     axon {stim = new IClamp(0.5)
          stim.del = 2
          stim.dur = 5
          stim.amp = $1 }
     }
//Ausgabe der Membranpotentiale der verschiedenen Segmente:
print "V soma = ", soma.v, mV"
print "V dend[0] = ", dend[0].v, mV"
print "V dend[1] = ", dend[1].v, mV"
print "V dend[2] = ", dend[2].v, mV"
print "V axon = ", axon.v, mV"
```

Durch einen Doppelklick auf die `hoc`-Datei öffnet sich die NEURON Oberfläche. Wenn man nun `stimver()` mit einem Parameter, welcher der gewünschten Amplitude entspricht, und danach den Befehl `run()` eingibt, berechnet NEURON die verschiedenen Membranpotentiale. Im NEURON Main Menu kann man nun unter `Graph` zum Beispiel den Graphen aussuchen, der das Membranpotential in Veränderung der Zeit zeigt (`voltage axis`).

Nach dem in diesem Unterkapitel die Progammiersprache im Mittelpunkt stand, werde ich im folgenden Unterkapitel die Benutzeroberfläche von NEURON mittels eines kleinen Beispiels näherbringen.

3.2 Benutzeroberfläche von NEURON

Zum Starten von NEURON klickt man auf die Datei `nrngui`, dadurch öffnen sich die Fenster `nrniv` und NEURON Main Menu. In `nrniv` Fenster kann man die Befehle die ich im vorangegangen Beispiel vorgestellt habe eingeben. Der Befehl zur endgültigen Durchführung lautet `run()`.

Im NEURON Main Menu kann man verschiedene `Tools` auswählen und hierbei ist `RunControl` eines der Interssanteren. Mittels der verschiedenen Buttons im `RunControl`-Fenster kann man verschiedene Bedingungen der Berechnung verändern. Hierbei sind die folgende Buttons erwähnenswert: `Init (mV)` (gibt den Startwert für das Membranpotential an), `Init & Run` (startet die Simulation), `Tstop (ms)` (gibt das Ende der Simulation an), `Points plotted/ms` (die hier eingetragen Zahl legt fest wieviele Punkte pro Millisekunde gezeichnet werden).

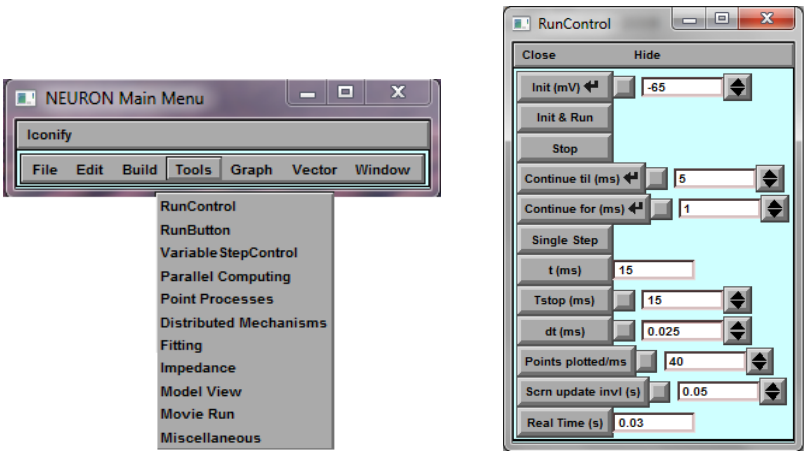

Abbildung 3.2: Rechts: `MainMenu`-Fenster; **links**: `RunControl`-Fenster

Wenn man darauf verzichten möchte zuerst ein Programm zu schreiben und darin die Eigenschaften für das gewünschte Neuron festzulegen, kann man mittels der NEURON Benutzeroberfläche ein Neuron graphisch erstellen. Dazu wählt man im Main Menu unter `Build`, `Cell Builder`. Hier kann man unter anderen den Button `Topology` wählen. Auf der erscheinenden graphischen Oberfläche wird nun das Neuron erstellt. Mittels `Make Section` zeichnet man die gewünschten Sektionen und bei `Basename` und unter `Change Name` werden die Sektionen dem Modell entsprechend festgelegt – siehe Abbildung 3.3. Durch Klicks auf die Buttons `Geometry` und `Biophysics` können die Eigenschaften der verschiedenen Sektionen definiert werden. Hierzu wählt man zuerst für jede Sektion die Variablen, die man veränderen will und klickt dann auf `Specify Strategy` um danach die Variablen festzulegen.

Um dem so entstandenen Neuron eine Elektrode einzuführen, klickt man

unter `Tools` auf `Point Processes` und hier auf `Managers` und `Point Manager`. Durch das Klicken auf den Button `SelectPointProcess` kann man aus einer Vielfalt an Stimulationen wählen – siehe Abbildung 3.4. Nach der Festlegung der gewünschten Stimulation, kann die Stimulation mittels `RunControl` gestartet werden.

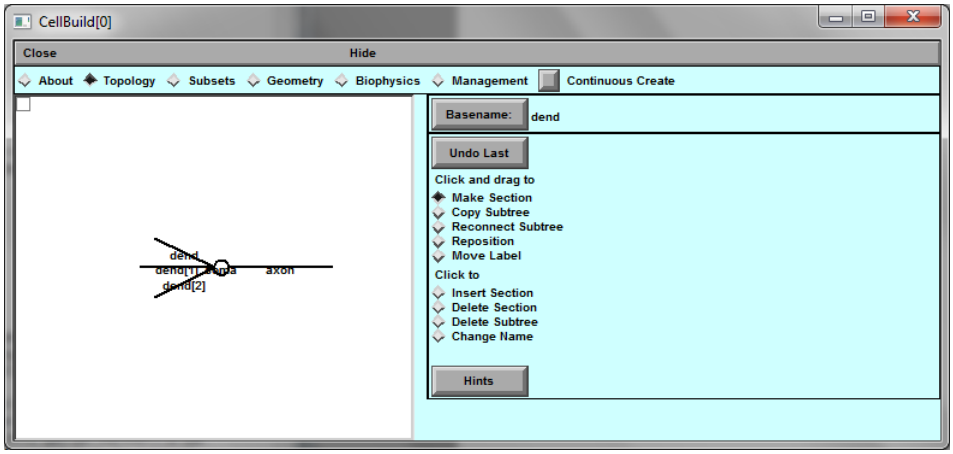

Abbildung 3.3: `CellBuild`-Fenster

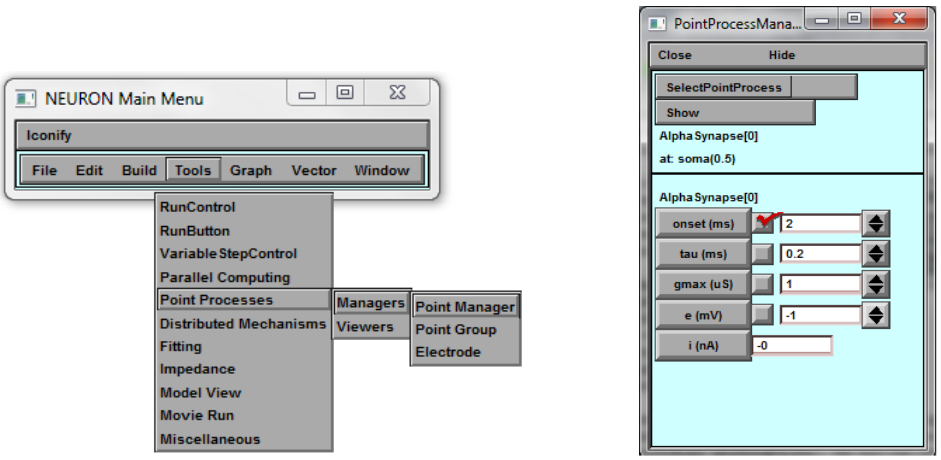

Abbildung 3.4: Links: `MainMenu`-Fenster; **rechts**: `PointProcessManager`-Fenster

Das Ergebnis der Stimulation sieht man am besten mit einer Grafik. Im Main Menu kann man unter `Graph` unter verschiedenen Achsen auswählen. Hierbei sind die `Voltage axis` und `Shape plot` am interssantesten. Beim `Shape plot` öffnet sich zuerst ein Fenster mit der Darstellung des Neurons.

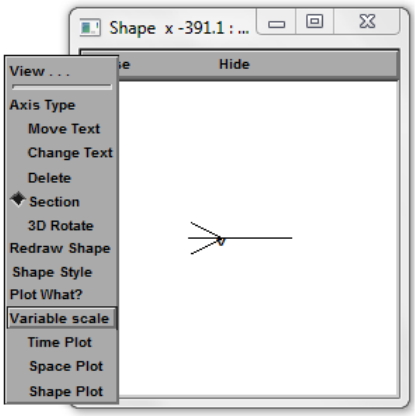

Abbildung 3.5: `ShapePlot`-Fenster

Hierbei wird zwischen dem `Time Plot`, `Space Plot` und `Shape Plot` entschieden. Wenn man `Space Plot` wählt, muss man mit gedrückt gehaltener linken Maustaste den gewünschte Bereich makieren. Danach öffnet sich ein neues Fenster mit einem Koordinatensystem, bei dem die x-Achse der Entfernung von einem zum anderen Ende des markierten Bereiches entspricht – siehe Abbildung 3.6. Der Graph verändert sich nur in der Zeit und zeigt am Ende den Graphen zum Zeitpunkt Tstop.

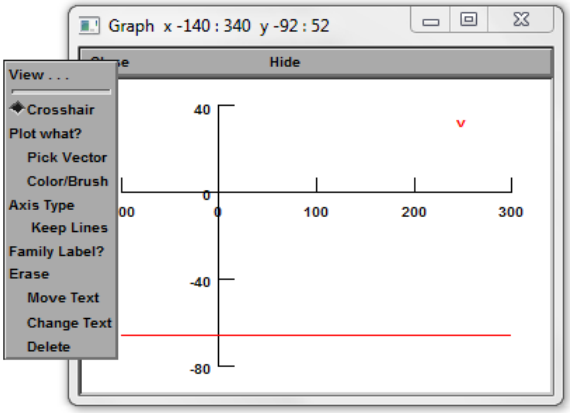

Abbildung 3.6: `SpacePlot`-Fenster

Wenn man die abgebildete Variable ändern möchte, so drückt man auf die rechte Maustaste und wählt danach `Plot What?`. Durch einen rechten Maus-Klick kann man nicht nur die Variable verändern, sondern auch den Achsen-Typ neu wählen. Mittels `keep lines` bleiben die verschiedenen Graphen zu

den verschieden Zeitpunkten bestehen.

Im nächsten Unterkapitel werde ich nun wieder zum Programmieren zurückkehren und anhand zweier einfacher Beispiele die wichtigsten Bestandteile eines NMODL Programmes erklären.

3.3 NMODL

Mittels NMODL kann man neuronale Funktionen programmieren und diese dann in NEURON einfügen. Dazu schreibt man zuerst das Programm, welches einen biophysikalischen Mechanismus durch eine nichtlineare algebraische Gleichung oder eine Diffentialgleichung definiert, in einem Texteditor und speichert dieses unter `name.mod` ab. Danach generiert man diesen einfachen Code in einen C-Code, um ihn dann in NEURON verwenden zu können.

Die Idee hinter NMODL basiert auf dem von Professoren der Duke Universität entwickelten Programm MODL (MOdel Description Language). Ein Programm in NMODL ist ebenso wie in MODL in mehrere Blöcke unterteilt, von denen ein paar die Variablen deklarieren (`PARAMETER`, `STATE` und `ASSIGNED`) und andere die Gleichungen definieren (unter anderem `INITIAL`, `BREAKPOINT`, `DERIVATIVE`, `FUNCTION` und `PROCEDURE`).

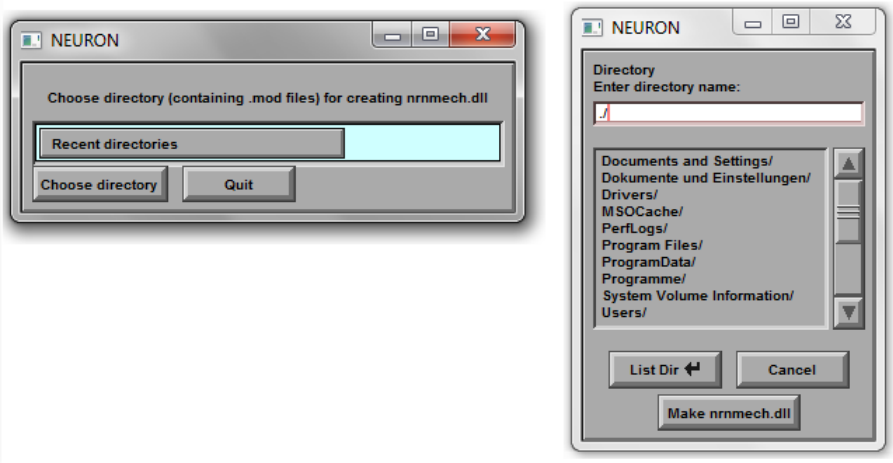

Abbildung 3.7: `mknrdll`-Fenster

Nachdem man den NMODL-Code in einem Texteditor geschrieben und unter `.mod` gespeichert hat, muss man die Datei noch kompilieren. Dies macht

man in WINDOWS folgendermaßen: Man klickt im Ordner NEURON auf die Datei `mknrndll`, dadurch öffnet sich ein Fenster bei diesem kann man wiederum `Choose directory` auswählen – siehe Abbildung 3.7. Danach sucht man den Ordner, in der die `mod`-Datei gespeichert ist und klickt auf den Button `Make nrnmech.dll`. Jetzt wird die Datei kompiliert und eine neue Datei mit dem gleichen Namen nur der Endung `.o` erscheint im Ordner. Beim Kompilieren wird auf etwaige Fehler im NMODL-Code hingewiesen.

3.3.1 NMODL-Beispiel: lokaler Shunt

Anhand dieses einfachen Beispieles werde ich die logischen Blöcke von NMODL erläutern. Bei einem Shunt handelt es sich um eine Art von Synapse, die auf einen kleinen Bereich beschränkt ist. [CH06, S.214 ff]

```
NEURON {
  POINT_PROCESS Shunt
  NONSPECIFIC_CURRENT i
  RANGE i, e, r
}
PARAMETER {
  r = 1 (gigaohm) <1e-9, 1e9>
  e = 0 (millivolt)
}
ASSIGNED {
  i (nanoamp)
  v (millivolt)
}
BREAKPOINT { i = (0.001)*(v-e)/r }
```

`NEURON`-Block

Mittels des `NEURON`-Blocks wird die Verbindung zwischen der `hoc`-Datei und der `mod`-Datei hergestellt. Der Befehl `POINT_PROCESS` gibt an, dass es sich um

eine Stimulation handelt. `Shunt` ist der Namen des Point Process und die `mod`-Datei kann folglich durch `stim = new Shunt(0.5)` eingebunden werden. Durch `NONSPECIFIC_CURRENT i` wird klargestellt, dass der Strom nicht bei der Dichte- und Ladungsberechnung von Ionenkonzentrationen berücksichtigt wird. Dies hat den Vorteil, dass die Rechenzeit verkürzt wird. Da `i`, `e` und `r` als `RANGE`-Variablen deklariert wurden, bedeutet dies, dass sie von der Position im Segment abhängig sind und somit verschiedene Werte innerhalb einer Sektion (zum Beispiel Axon oder Dendrit) annehmen können.

`PARAMETER`-Block

Variablen, die im `PARAMETER`-Block definiert werden, bleiben während der Simulation konstant und dürfen innerhalb der `mod`-Datei nicht neu berechnet werden. In unserem Beispiel wurden den konstanten Variablen `r` und `e` die Werte 1GΩ und 0mV zugewiesen. Diese Konstanten können mittels der Benutzeroberfläche von NEURON verändert werden. Die beiden Werte `1e-9` und `1e9` geben den minimalen und maximalen Wert der Konstanten an.

`ASSIGNED`-Block

Im `ASSIGNED`-Block werden zwei Arten von Variablen festgelegt, einerseits jene die in der `mod`-Datei berechnet werden, andererseits die, die Werte an NEURON übergeben. Diese Variablen können in NEURON mittels `range` oder `global` sichtbar gemacht werden. Es gibt jedoch auch Variablen die immer sichtbar sind (`v`, `celsius`, `t`, `dt`, `diam` und `area`). Variablen, die in diesem Block deklariert werden, sind `range`-Variablen, also abhängig von der Position im Segment.

`BREAKPOINT`-Block

In diesem Block steht die Gleichung zur Berechnung des Shunt.

Wie bereits erwähnt, wird dieser neue Point Process in ein NEURON Programm mittels `stim = new Shunt(0.5)` eingefügt. Man kann zum Beispiel

durch den Befehl `stim.r = 0.2` die Konstante `r` verändern oder man macht dies auf der Benutzeroberfläche. Dabei muss man im `PointProcessManager` zuerst den neuen Point Process auswählen und kann diesen danach nach Belieben verändern – siehe Abbildung 3.8.

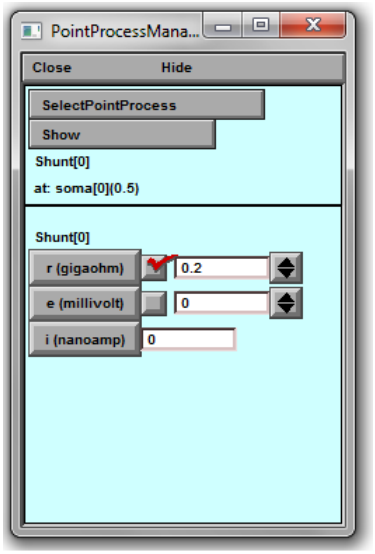

Abbildung 3.8: `Shunt`-Fenster

3.3.2 NMODL-Beispiel: besonderer Kaliumkanal in HH

Bei diesem Beispiel wird bei der Implementierung der Hodgkin-Huxley Gleichungen berücksichtigt, dass es eine post-hyperpolarisierte Reaktion auf ein Stimualtionspotential gibt. Diese Reaktion wird durch einen bestimmten Kaliumkanal, den T-Typ Kaliumkanal, gesteuert. Im Folgenden wird dieser Kaliumkanal in eine `mod`-Datei implementiert.

Die Grundlage zu diesem Beispiel habe ich [WEB2] entnommen.

```
NEURON {
  SUFFIX CaT
  USEION ca READ eca WRITE ica
  RANGE gmax
}
```

```
UNITS {
  (mV) = (millivolt)
  (mA) = (milliamp)
}
PARAMETER {
  gmax = 0.002 (mho/cm2)
}
ASSIGNED {
  v (mV)
  eca (mV)
  ica (mA/cm2)
  ralpha (/ms)
  rbeta (/ms)
  salpha (/ms)
  sbeta (/ms)
  dalpha (/ms)
  dbeta (/ms)
}
STATE {
  r s d
}
BREAKPOINT {
  SOLVE states METHOD cnexp
  ica = gmax*r*r*r*s*(v-eca)
}
INITIAL {
  settables(v)
  r = ralpha/(ralpha+rbeta)
  s = (salpha*(dbeta+dalpha) - (salpha*dbeta))/
      ((salpha+sbeta)*(dalpha+dbeta) - (salpha*dbeta))
```

```
  d = (dbeta*(salpha+sbeta) - (salpha*dbeta))/
      ((salpha+sbeta)*(dalpha+dbeta) - (salpha*dbeta))
}
DERIVATIVE states {
  settables(v)
  r' = ((ralpha*(1-r)) - (rbeta*r))
  d' = ((dbeta*(1-s-d)) - (dalpha*d))
  s' = ((salpha*(1-s-d)) - (sbeta*s))
}
UNITSOFF
 PROCEDURE settables(v (mV)) {
  LOCAL bd
  TABLE ralpha, rbeta, salpha, sbeta, dalpha, dbeta
  FROM -100 TO 100 WITH 200
  ralpha = 1.0/(1.7+exp(-(v+28.2)/13.5))
  rbeta = exp(-(v+63.0)/7.8)/(exp(-(v+28.8)/13.1)+1.7)
  salpha = exp(-(v+160.3)/17.8)
  sbeta = (sqrt(0.25+exp((v+83.5)/6.3))-0.5) *
       (exp(-(v+160.3)/17.8))
  bd = sqrt(0.25+exp((v+83.5)/6.3))
  dalpha = (1.0+exp((v+37.4)/30.0))/(240.0*(0.5+bd))
  dbeta = (bd-0.5)*dalpha
}
UNITSON
```

`NEURON`-Block

Im Gegensatz zum vorherigen Beispiel definiert dieser Code keinen Point Process sondern eine besondere Art der Stimulationsweiterleitung und wird dementsprechend in NEURON mit dem Befehl `insert CaT` aufgerufen. Mittels `USEION` wird definiert welche Ionen dieser implementierte Mechanismus beeinflußt.

NEURON kennt drei vorgegebene Ionen `na`, `k`, `ca` und da unser Beispiel auf den Kaliumkanal eingeht, beeinflußt unser Mechanismus nur Kaliumionen. `READ` bestimmt die Variablen die für die Berechnung des Ionen-Kanalstroms benötigt werden (`eca` entspricht dem Equilibrium Potential) und `WRITE` gibt an, welche Variablen im Folgenden berechnet werden (`ica` ist der zu berechnende Kaliumstrom).

`UNITS`-Block

In diesem Block werden Abkürzungen für die vorgegeben Einheiten deklariert. In einem NMODL-Code muss immer jeder deklarierten Variablen eine Einheit zugewiesen werden. Es gibt jedoch auch Variablen bei denen die Einheiten durch NEURON bereits vorgegeben sind: `v [millivolt]`, `t [millisecond]`, `celsius [degrees centigrade]`, `diam` [μm], `area` [μm^2]. Den Größen die durch den Befehl `USEION` erzeugt werden, müssen ebenfalls Einheiten zugewiesen werden.

`PARAMETER`- und `ASSIGNED`-Block

In unserem Beispiel gibt es nur einen Parameter, der gleichzeitig eine `RANGE`-Variable ist (also abhängig von der Position) und diese entspricht der maximalen Kanalleitfähigkeit. Wie bereits erwähnt müssen auch jenen Variablen, die durch `USEION` erzeugt werden, Einheiten zugewiesen werden und dies passiert im `ASSIGNED`-Block.

`STATE`-Block

Im `STATE`-Block werden die Zustandsvariablen definiert, also jene Variablen, die durch Ableitungen im Differentialsystem abhängig sind. Natürlich sind diese Variablen auch von der Position im Segment abhänigig, da die Kanalöffnungswahrscheinlichkeit unterschiedlich sein kann.

Gleichungen definierende Blöcke

Im Folgenden fasse ich die für die Differentialgleichungen wichtigen Blöcke zusammen (`BREAKPOINT`-, `INITIAL`-, `DERIVATE`- und `PROCEDURE`-Block). In diese Blöcken werden folgende Differentialgleichungen berechnet:

$$\dot{r} = \alpha_{\mathrm{r}}(1 - r) - \beta_{\mathrm{r}} r$$

$$\dot{s} = \alpha_{\mathrm{s}}(1 - s - d) - \beta_{\mathrm{s}} s$$

$$\dot{d} = \beta_{\mathrm{d}}(1 - s - d) - \alpha_{\mathrm{d}} d$$

Im `BREAKPOINT`-Block wird zuerst angegeben mit welcher Methode die Differentialgleichungen gelöst werden (`SOLVE states METHOD cnexp`). Die hier angewandte Methode eignet sich vor allem für Differentialgleichungen der Form $\dot{y} = f(V, y)$, wobei f eine lineare Funktion ist und keine anderen Zustandsgleichungen beeinflusst. Die Gleichung in diesem Block entspricht folgender Gleichung zur Berechnung des Kaliumstroms:

$$I_{\mathrm{T}} = g_{\mathrm{T(max)}} r^3 s (V - E_{\mathrm{Ca}})$$

Mittels des `INITIAL`-Blocks werden die Anfangswerte der Differentialgleichungen festgesetzt – normalerweise verwendet man hierfür die Werte aus dem Gleichgewicht. In unserem Fall wird zuerst die Funktion `settables(v)` aufgerufen, diese Funktion wird im `PROCEDURE`-Block definiert, um mit den Alpha- und Beta-Werten die initialen Werte von `r`, `s` und `d` zu berechnen. Auf die Initialisierung der Zustandsvariablen darf nicht vergessen werden, da sie wichtig ist zur Berechnung der Differentialgleichungen.

Einer der wesentlichen Blöcke ist natürlich der `DERIVATE`-Block. In diesem werden die oben angegebenen Differentialgleichungen festgelegt, die dann durch die im `BREAKPOINT`-Block definierte Methode gelöst werden. Bei jeder Berechnung der Differentialgleichungen durch NEURON müssen die Alpha- und Beta-Werte auf den neuesten Stand gebracht werden und daher benötigt man vor den Differentialgleichungen den Befehl `settables`.

Zu guter Letzt gibt es noch den `PROCEDURE`-Block. In diesem wird die Funktion `settables(v)` definiert, da die Alpha- und Beta-Variablen für jede neue Berechnung der Differentialgleichung wieder berechnet werden müssen. Durch die Definition der Funktion erleichtert dies die Berechnung der Variablen. In unserem Beispiel benötigt man hierfür den akutellen Strom v. Die Berechnungszeit kann durch den Befehl `TABLE` verkürzt werden, da eine Tabelle für die zu berechnenden Variablen angelegt wird und somit die Werte nicht jedes mal neu berechnet sondern aus der Tabelle abgelesen werden. Mittels `TABLE` legt man zuerst fest um welche Variablen es sich handelt und gibt dann durch `FROM` und `TO` den niedrigsten und höchsten Wert von `v` und `WITH` die Schritte dazwischen an. Die danach folgenden Gleichungen berechnen die Alpha- und Beta-Variablen.

Da in einem NMODL-Code sehr darauf geachtet werden muss, dass die Einheiten übereinstimmen, kann man die strenge Einheitenübereinstimmungsprüfung bei der Berechnung der einzelnen Alpha- und Beta-Variablen mit den Befehlen `UNITSOFF` und `UNITSON` aus- und wieder einschalten.

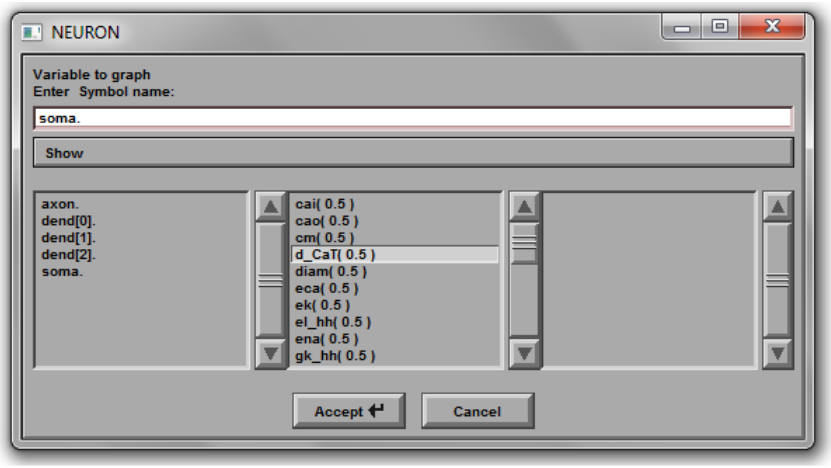

Abbildung 3.9: `CaT`-Fenster

Wie bereits weiter oben erwähnt bindet man diesen bestimmten Kaliumkanal durch `insert CaT` in eine NEURON-Datei ein. Man kann dann auch die hier neu definierten Variablen graphisch darstellen lassen – siehe Abbildung 3.9.

Im folgenden Kapitel werde ich die hier vorgestellten Programme NEURON und NMODL anwenden um meine Berechnung darzustellen.

Kapitel 4

Stimulation mittels eines langen Impulses

4.1 Erklärung des Modells

Als Grundlage meiner Diplomarbeit dient unter anderem die wissenschaftliche Veröffentlichung *Modelling spinal circuitry involved in locomotor pattern generation: insights from deletions during fictive locomotion* von Ilya A. Rybak, Natalia A. Shebtsova, Myriam Lafreniere-Roula und David A. McCrea. Das Rückenmark eines Säugetieres enthält einen zentralen Bewegungsmustergenerator, der eine alternierende rhytmische Aktivität im Beuge- und Streckmuskel ohne einen rhytmischen Input oder propriozeptive Resonanz erzeugen kann. [RSLR06, S.617]

Es wird ein mathematisches Modell aufgestellt dessen Differentialgleichungen den Hodgkin-Huxley Gleichungen ähneln. In dem Modell treten folgende Ionen-Ströme auf: schneller Natrium- (I_{Na} mit maximaler Leitfähigkeit g_{Na}), verzögerter-gleichgerichteter Kalium- (I_{K} mit maximaler Leitfähigkeit g_{K}), Kalzium-N- (I_{CaN} mit maximaler Leitfähigkeit g_{CaN}), Kalzium-L- (I_{CaL} mit maximaler Leitfähigkeit g_{CaL}), Kalzium-abhängiger Kalium- ($I_{\mathrm{K,Ca}}$ mit maximaler Leitfähigkeit $g_{\mathrm{K,Ca}}$) sowie persistenter (langsam deaktivierender) Natrium- (I_{NaP} mit maximaler Leitfähigkeit g_{NaP}) und der Leckstrom (I_{L} mit konstanter Leitfähigkeit g_{L}) . Die Gleichungen für die Ionen-Ströme lau-

ten folgendermaßen:

$$I_{\mathrm{Na}} = \bar{g}_{\mathrm{Na}} m_{\mathrm{Na}}^3 h_{\mathrm{Na}} (V - E_{\mathrm{Na}})$$

$$I_{\mathrm{NaP}} = \bar{g}_{\mathrm{NaP}} m_{\mathrm{NaP}} h_{\mathrm{NaP}} (V - E_{\mathrm{Na}})$$

$$I_{\mathrm{K}} = \bar{g}_{\mathrm{K}} m_{\mathrm{K}}^4 (V - E_{\mathrm{K}})$$

$$I_{\mathrm{CaN}} = \bar{g}_{\mathrm{CaN}} m_{\mathrm{CaN}}^2 h_{\mathrm{CaN}} (V - E_{\mathrm{Ca}})$$

$$I_{\mathrm{CaL}} = \bar{g}_{\mathrm{CaL}} m_{\mathrm{CaL}} (V - E_{\mathrm{Ca}})$$

$$I_{\mathrm{K,Ca}} = \bar{g}_{\mathrm{K,Ca}} m_{\mathrm{K,Ca}} (V - E_{\mathrm{K}})$$

$$I_{\mathrm{L}} = g_{\mathrm{L}} (V - E_{\mathrm{L}})$$

V steht hierbei für das Membran-Potential des Neurons; E_{Na}, E_{K}, E_{Ca} und E_{L} sind die Umkehrpotentiale für den Natrium- , Kalium- , Kalzium- und Leckstrom. Die Variablen m und h beschreiben die Aktivierung und Deaktivierung der jeweiligen Ionen-Kanäle und werden wie folgt definiert:

$$\tau_{\mathrm{m_i}} \frac{d}{dt} m_{\mathrm{i}} = m_{\infty \mathrm{i}}(V) - m_{\mathrm{i}},$$

$$\tau_{\mathrm{h_i}} \frac{d}{dt} h_{\mathrm{i}} = h_{\infty \mathrm{i}}(V) - h_{\mathrm{i}}$$

wobei i für die verschiedenen Ionen-Kanäle steht. $m_{\infty \mathrm{i}}$ und $h_{\infty \mathrm{i}}$ definieren die stabilen Zustände der Aktivierung und Inaktivierung und $\tau_{\mathrm{m_i}}$ sowie $\tau_{\mathrm{h_i}}$ die dazugehörigen Zeitkonstanten – siehe Tabellen 4.2 und 4.1.

In dem mathematischen Modell wird auch noch ein besonderes Augenmerk auf die Ca^{2+}-Konzentration gelegt. Die Bewegungen dieser können mittels einer Differentialgleichung modelliert werden:

$$\frac{d}{dt}\mathrm{Ca} = f(-\alpha I_{\mathrm{Ca}} - k_{\mathrm{Ca}}\mathrm{Ca})$$

Bei dieser Gleichung definiert f den Prozentsatz der freien zu den ganzen

Ionen-Kanäle	$\tau_{\mathrm{m}}(V)$ $\tau_{\mathrm{h}}(V)$
Na^+	$\tau_{\mathrm{mNa}} = 0$
	$\tau_{\mathrm{hNa}} = 30/(e^{((V+50)/15)} + e^{(-(V+50)/16)})$
Na_P^+	$\tau_{\mathrm{mNaP}} = 0$
	$\tau_{\mathrm{hNaP}} = 1200/\cosh((V+59)/16)$
K^+	$\tau_{\mathrm{mK}} = 7/(e^{((V+40)/40)} + e^{(-(V+40)/50)})$
	$h_{\mathrm{K}} = 1$
Ca_N^{2+}	$\tau_{\mathrm{mCaN}} = 4$
	$\tau_{\mathrm{hCaN}} = 20$
Ca_L^{2+}	$\tau_{\mathrm{mCaL}} = 40$
	$h_{\mathrm{CaL}} = 1$

Tabelle 4.1: Parameter der Hodgkin-Huxley Gleichungen – τ_m und τ_h

Ionen-Kanäle	$m_\infty(V)$ $h_\infty(V)$
Na^+	$m_{\infty\mathrm{Na}} = (1 + e^{(-(V+35)/7{,}8)})^{-1}$
	$h_{\infty\mathrm{Na}} = (1 + e^{((V+55)/7)})^{-1}$
Na_P^+	$m_{\infty\mathrm{NaP}} = (1 + e^{(-(V+47{,}1)/3{,}1)})^{-1}$
	$h_{\infty\mathrm{NaP}} = (1 + e^{((V+59)/8)})^{-1}$
K^+	$m_{\infty\mathrm{K}} = (1 + e^{(-(V+28)/15)})^{-1}$
	$h_{\mathrm{K}} = 1$
Ca_N^{2+}	$m_{\infty\mathrm{CaN}} = (1 + e^{(-(V+30)/5)})^{-1}$
	$h_{\infty\mathrm{CaN}} = (1 + e^{((V+45)/5)})^{-1}$
Ca_L^{2+}	$m_{\infty\ \mathrm{CaL}} = (1 + e^{(-(V+40)/7)})^{-1}$
	$h_{\mathrm{CaL}} = 1$

Tabelle 4.2: Parameter der Hodgkin-Huxley Gleichungen – m_∞ und h_∞

	RG-Neuron	Motoneuron
$\bar{g}_{\mathrm{Na}}$	$30\mathrm{mS/cm}^2$	$120\mathrm{mS/cm}^2$
$\bar{g}_{\mathrm{NaP}}$	$0{,}25\mathrm{mS/cm}^2$	
$\bar{g}_{\mathrm{K}}$	$1\mathrm{mS/cm}^2$	$100\mathrm{mS/cm}^2$
$\bar{g}_{\mathrm{CaN}}$		$14\mathrm{mS/cm}^2$
$\bar{g}_{\mathrm{K,Ca}}$		$5\mathrm{mS/cm}^2$
g_{L}	$0{,}1\mathrm{mS/cm}^2$	$0{,}51\mathrm{mS/cm}^2$
E_{L}	$-64 \pm 0{,}64\mathrm{mV}$	$-65 \pm 6{,}5\mathrm{mV}$

Tabelle 4.3: Parameter der Ionen-Kanäle in den dazugehörigen Neuronen

Ca^{2+}-Ionen, α konvertiert den Ca^{2+}-Strom (I_{Ca}) in eine Ca^{2+}-Konzentration und k_{Ca} gibt die Ca^{2+} Abbau-Rate an.

Alle Gleichungen, Definitionen und Parameter sind [RSLR06] entnommen.

In der Veröffentlichung von Ilya A. Rybak wird davon ausgegangen, dass der zentrale Bewegungsgenerator auf zwei Ebenen basiert. Einerseits gibt es den Generator, der den Rhythmus erzeugt (Rhythm Generator), welcher besagt, dass der Beug- und der Streckmuskel (Extensor und Flexor) unterschiedlich beansprucht werden. Andererseits gibt es die Ebene, die den Rhytmus erhält und an die Motoneuronen weitergibt (Pattern Formation) - siehe Abbildung 4.1.

Für meine Berechnungen reicht eine Ebene, die den Rhythmus vorgibt. Daher vereinfache ich das gerade vorgestellte Modell und lasse die zweite Ebene beiseite. Somit gibt es im vereinfachten Modell nur mehr die Ebene des Rhythm Generator und danach folgen die Motoneuronen. Diese sind mittels erregender (exzitatorischer) und hemmender (inhibitorischer) Synapsen verbunden. Es gibt nicht nur die beiden Zellen des Rhythm Generator und die der Motoneuronen, sondern auch einfache Interneuronen. Diese sind mit den Zellen des Rhythm Generators und den Motoneuronen durch hemmende Synapsen verbunden – siehe Abbildung 4.2.

Durch die speziellen Differentialgleichungen der Zellen im Rhythm Gene-

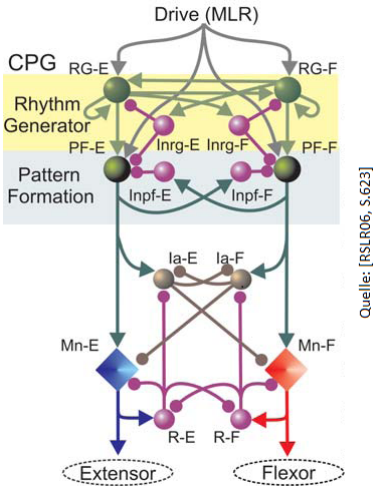

Abbildung 4.1: Schematische Darstellung des Modells von Ilya A. Rybak. Die Interneuronen werden durch Kugeln dargestellt. Erregende Synapsen sind mittels Pfeile gekennzeichnet, hemmende Synapsen mittels kleiner Kreise. Die Rauten repräsentieren die Motoneuronen.

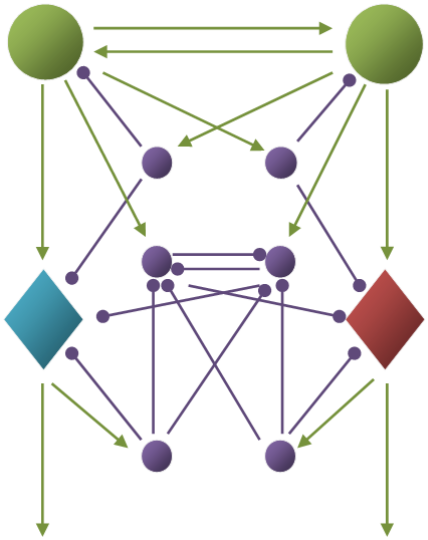

Abbildung 4.2: Schematische Darstellung des vereinfachten Modells. Die Kugeln repräsentieren die Interneuronen. Die Pfeile kennzeichnen erregende Synapsen, die kleinen Kreise hingegen hemmende Synapsen. Die Motoneuronen werden durch Rauten dargestellt.

rator und durch die hemmende Wirkung der Interneuronen kommt es zu dem für diese Zellen typischen alternierenden Muster. Das alternierende Muster entsteht vor allem durch den persistenten Natrium-Kanal.

4.2 Persistenter Natrium-Kanal

Der wesentliche Bestandteil der Rhythm Generator Zellen ist der persistente Natrium-Kanal, welcher ausschlaggebend für das spezifische Verhalten dieser Zellen ist. Das rhythmische Verhalten entsteht dadurch, dass der zusätzliche Natrium-Kanal langsam inaktivierend ist. In den Abbildungen 4.3 bis 4.5 ist das spezifische Verhalten dieser Zellen zu sehen.

Im Folgenden vergleiche ich die RG-Zelle mit einer Zelle, die mittels der Hodgkin-Huxley-Gleichungen beschrieben werden (HH-Zellen). Die HH-Zellen haben nach einer Aktivierung durch ein Aktionspotential eine Depolarisierungsphase, die es in den RG-Zellen nicht gibt. Die RG-Zellen bleient bei einem längeren gleichbleibenden Impuls auf einer gewissen Höhe und erzeugen in kürzeren Abständen Aktionspotentiale, wobei hingegegen die HH-Zellen nach dem Aktionspotential depolarisiert werden und danach erst wieder das nächste Aktionspotential erzeugen.

Weiters will ich noch die beiden Natrium-Kanäle vergleichen. Der persistente Natrium-Kanal erreicht 50ms nach dem Beginn des Impulses ein konstantes Niveau und schwingt dann in einem gewissen Rahmen. Der schnelle Natrium-Kanal hingegen kehrt immer wieder zu seinem Ausgangsniveau zurück. Außerdem braucht der persistente Natrium-Kanal länger um nach der Aktivierung durch einen Impuls wieder auf sein Ausgangsniveau zurückzukehren.

Der persistente Natrium-Kanal erzeugt also die Höhe von der aus die RG-Zelle die Aktionspotentiale während eines anhaltenden Impulses generiert. Verbindet man nun zwei RG-Zellen und fügt hemmende Zellen dazwischen ein, so entsteht das von diesen Zellen gewünschte rhytmische Verhalten.

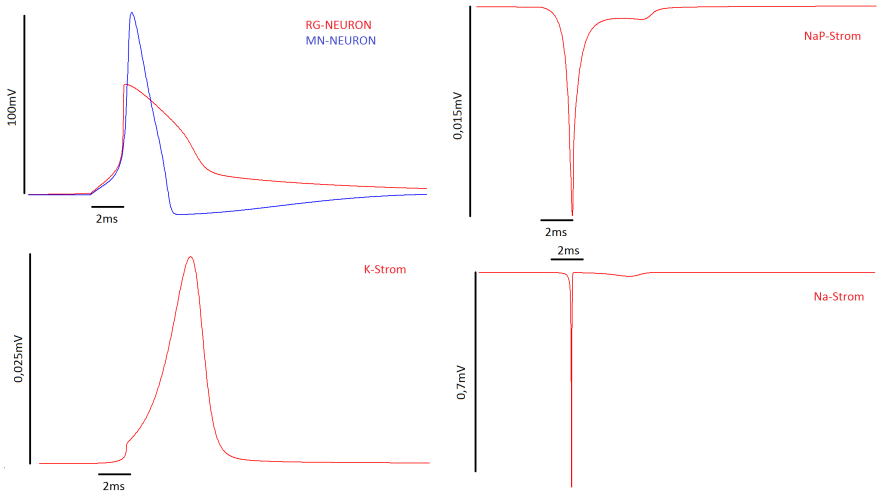

Abbildung 4.3: Links oben: RG-Zelle (rot) und HH-Zelle (blau) bei einem Impuls von 2ms; **rechts oben**: persistenter Natrium-Strom; **links unten**: Kalium-Strom; **rechts unten**: Natrium-Strom

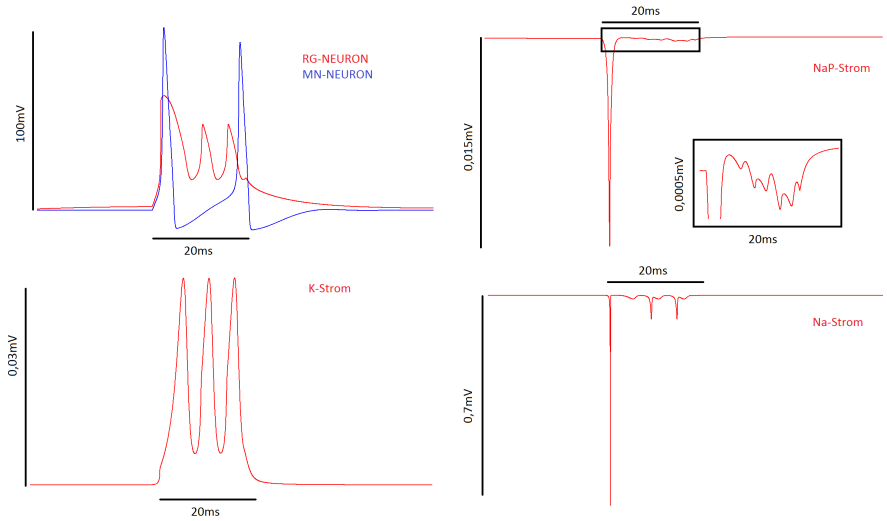

Abbildung 4.4: Links oben: RG-Zelle (rot) und HH-Zelle (blau) bei einem Impuls von 20ms; **rechts oben**: persistenter Natrium-Strom; **links unten**: Kalium-Strom; **rechts unten**: Natrium-Strom

4.3 Ergebnisse der Simulation

Das hier verwendete vereinfachte Modell (Abbildung 4.2) simuliere ich in dem Programm NEURON (Version 7.1). Dazu werden zuerst die verschiedenen Ner-

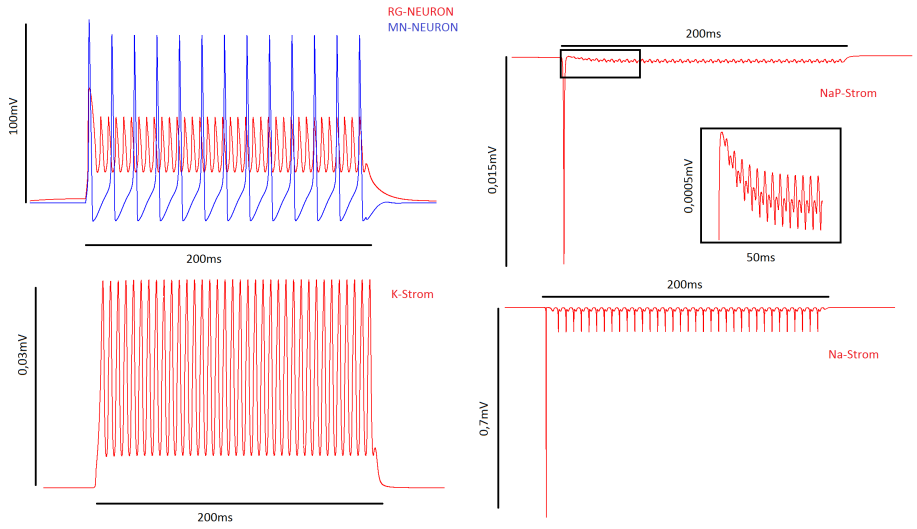

Abbildung 4.5: Links oben: RG-Zelle (rot) und HH-Zelle (blau) bei einem Impuls von 200ms; **rechts oben**: persistenter Natrium-Strom; **links unten**: Kalium-Strom; **rechts unten**: Natrium-Strom

venzellen programmiert und danach mit `ExpSyn` verbunden. Die Verbindung erfolgt direkt von Soma zu Soma. Des Weiteren füge ich einen fünf Sekunden langen gleichmäßigen Stimulus ein (`IClamp`).

Der gleichmäßige Impuls wird gleichzeitig in die beiden RG-Neuronen impliziert. Durch den persistenten Natrium-Strom entsteht das alternierende Muster in den RG-Neuronen und folglich auch in den Motoneuronen. Während der Phase, in der ein Neuron aktiv ist, feuert dieses in kurzen Abständen Impulse an folgende Zellen. Die Motoneuronen übernehmen also das rhythmische Verhalten der RG-Neuronen.

Die Stärke des Stimulus beeinflusst die Feuerungsdauer der Extensor- oder Flexor-Zelle. Bei einem Impuls von 0,012nA liegt die Dauer ungefähr bei 200ms – siehe Abbildungen 4.6 und 4.7. Erhöht man diesen um 0,002nA, so beträgt die Dauer ungefähr 500ms – siehe Abbildungen 4.8 und 4.9. Wenn man einen Impuls der Stärke 0,018nA in die RG-Zellen impliziert, so dauert die Feuerung bereits 3800ms – siehe Abbildungen 4.12 und 4.13. Man kann also erkennen, dass man die Feuerungsdauer erhöhen kann in dem man den Impuls verstärkt.

Im weiteren Verlauf betrachte ich den Abstand zwischen den Aktionspo-

Impuls	0,01nA	0,011nA	0,012nA	0,013nA	0,014nA
Dauer	100ms	150ms	200ms	350ms	500ms
Abstand	40ms	35ms	30ms	25ms	22ms

Impuls	0,015nA	0,016nA	0,017nA	0,018nA	0,019nA
Dauer	800ms	1200ms	1900ms	3800ms	über 5000ms
Abstand	19ms	18ms	17ms	15ms	13ms

Tabelle 4.4: Auflistung der Feuerungsdauer einer Zelle sowie des Abstandes zwischen den Aktionspotentialen in Abhängigkeit der Stärke des Impulses

tentialen, die eine RG-Zelle an die folgenden Zellen weitergibt. Der Abstand verringert sich mit steigender Stärke des Impulses. Bei einem Impuls von 0,012nA beträgt der Abstand ungefähr 30ms, bei 0,014nA ungefähr 22ms und bei 0,018nA nur mehr ungefähr 15ms – siehe Abbildung 4.14 und Tabelle 4.4.

Im nächsten Kapitel wird das gerade beschriebene Modell weiterhin verwendet, jedoch wird der Impuls verändert.

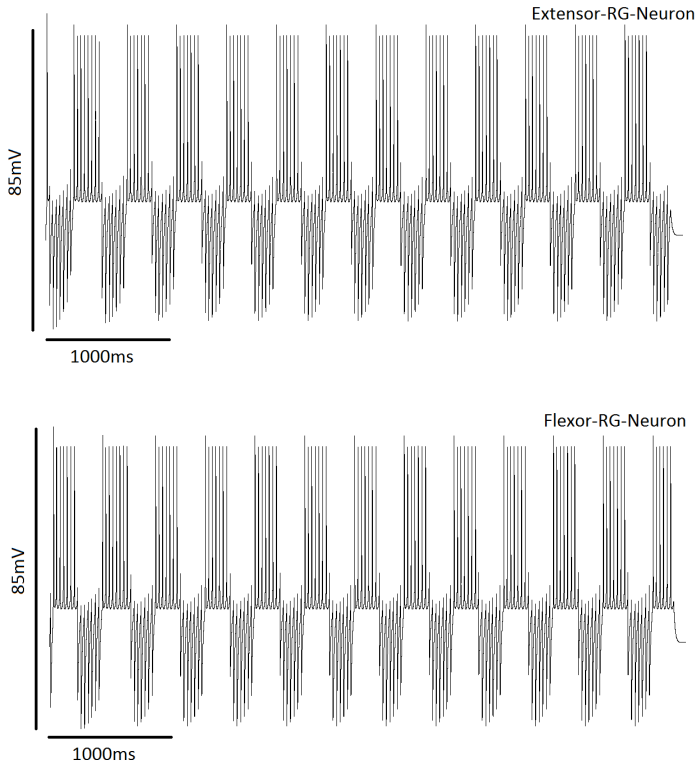

Abbildung 4.6: Oben: RG-Extensor-Zelle bei einem Stimulus von 0,012nA, **unten**: RG-Flexor-Zelle; Feuerungsdauer: 200ms

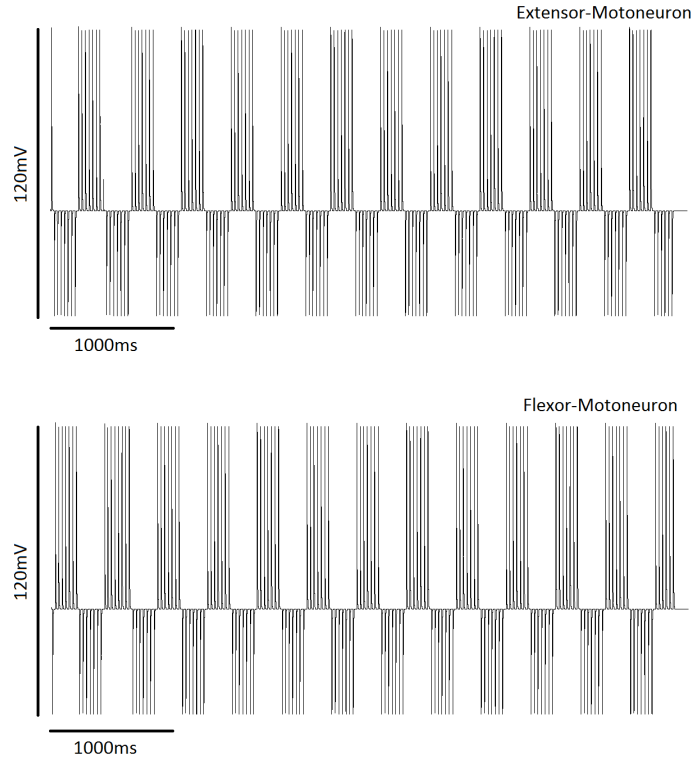

Abbildung 4.7: Oben: Moto-Extensor-Zelle, **unten**: Moto-Flexor-Zelle

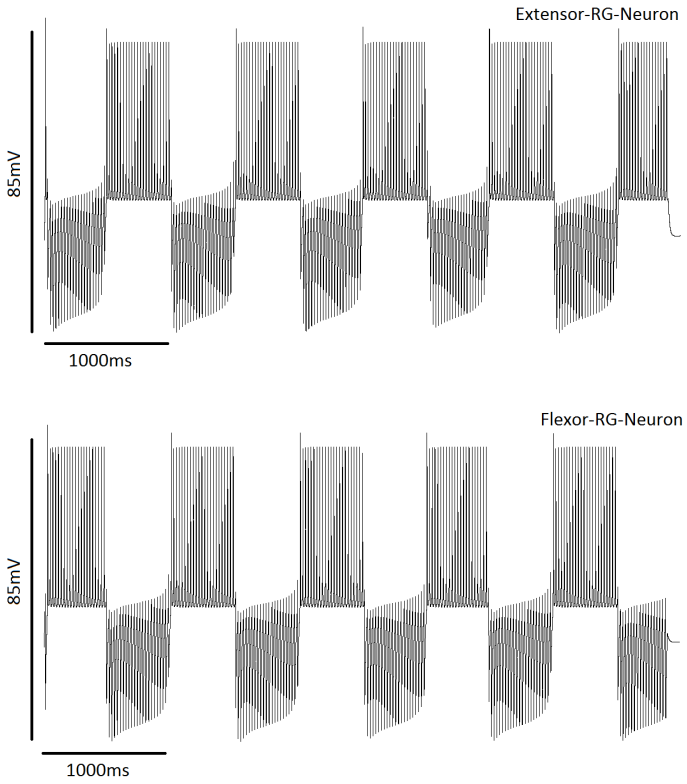

Abbildung 4.8: Oben: RG-Extensor-Zelle bei einem Stimulus von 0,014nA, **unten**: RG-Flexor-Zelle; Feuerungsdauer: 500ms

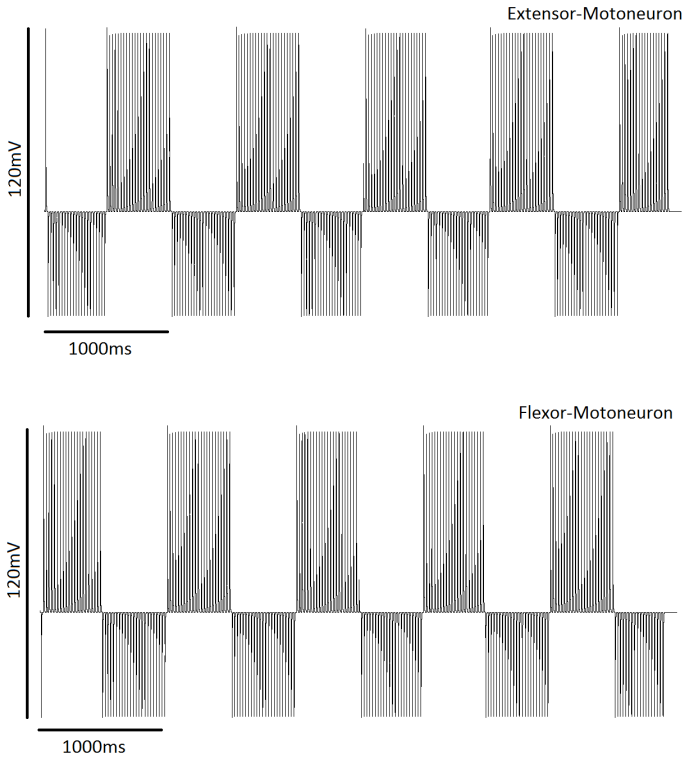

Abbildung 4.9: Oben: Moto-Extensor-Zelle, **unten**: Moto-Flexor-Zelle

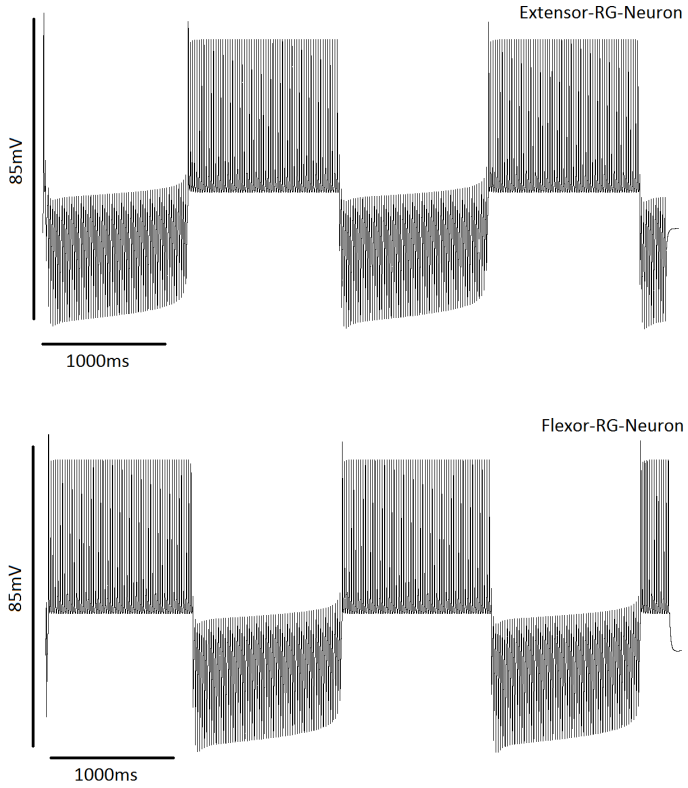

Abbildung 4.10: Oben: RG-Extensor-Zelle bei einem Stimulus von 0,016nA, **unten**: RG-Flexor-Zelle; Feuerungsdauer: 1200ms

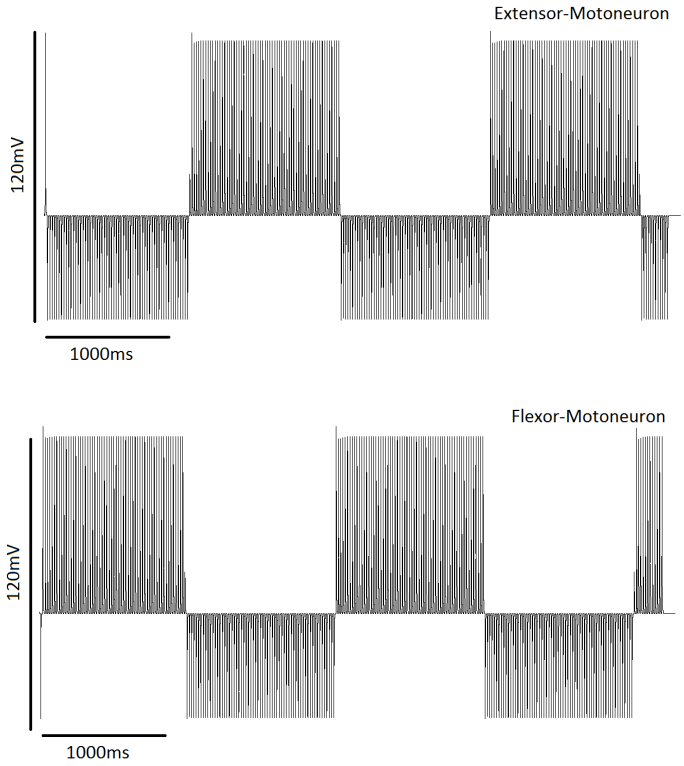

Abbildung 4.11: Oben: Moto-Extensor-Zelle, **unten**: Moto-Flexor-Zelle

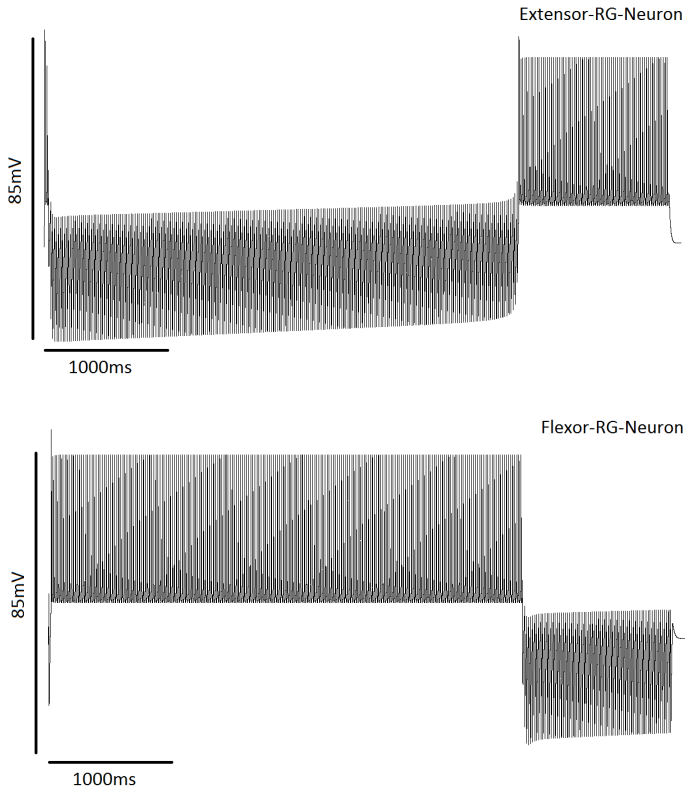

Abbildung 4.12: Oben: RG-Extensor-Zelle bei einem Stimulus von 0,018nA, **unten**: RG-Flexor-Zelle; Feuerungsdauer: 3800ms

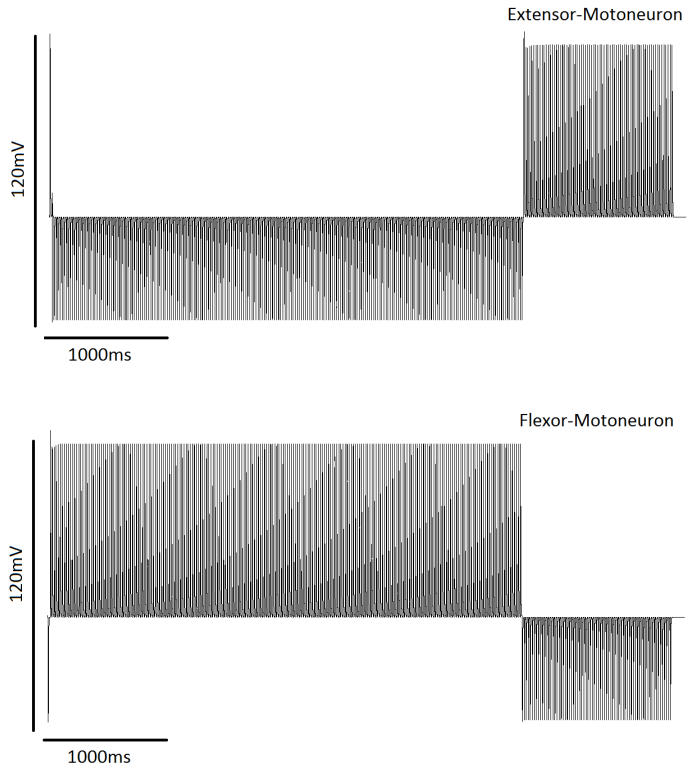

Abbildung 4.13: Oben: Moto-Extensor-Zelle, **unten**: Moto-Flexor-Zelle

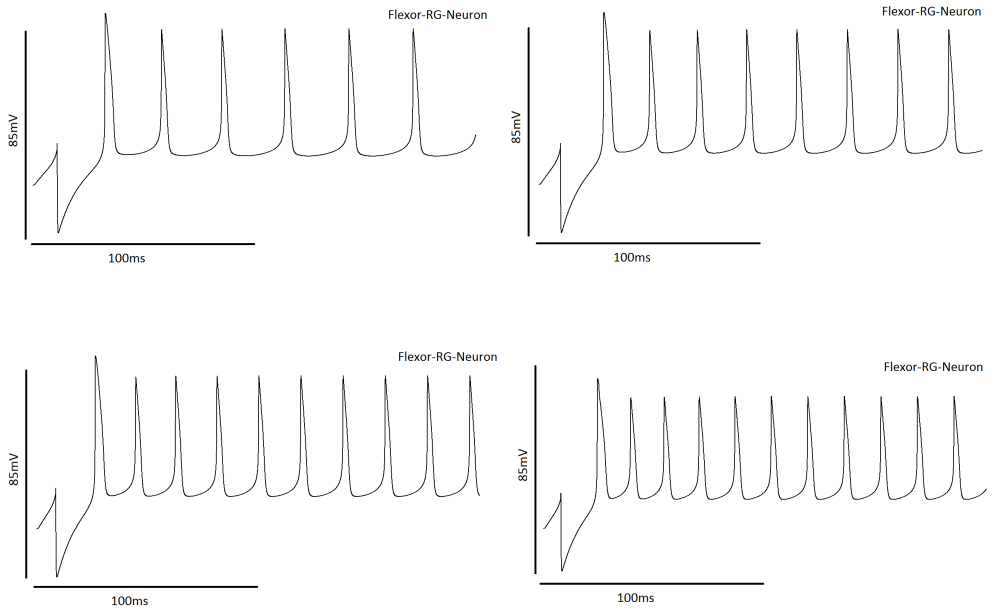

Abbildung 4.14: RG-Flexor-Zellen bei einem Stimulus von 0,012nA (links oben), 0,014nA (rechts oben), 0,016nA (links unten) bzw. 0,018nA (rechts unten)

Kapitel 5

Stimulation mittels kurzen Implusen

5.1 Erklärung des Modells

Im folgenden Teil meiner Diplomarbeit wird das Modell des letzten Kapitels wieder verwendet, nur wird dieses Mal nicht ein fünf Sekunden dauernder gleichmäßiger Impuls die Zellen antreiben sonderen mehrere kurze Impulse. Als Grundlage meiner Überlegungen dient mir die wissenschaftliche Veröffentlichung von K. Minassian, I. Persy, F. Rattay et al. *Human lumbar cord circuitries can be activated by extrinsic tonic input to generate locomotor-like activity* sowie die Dissertation von Ursula Hofstötter *Model of spinal cord reflex circuits in humans: Stimulation frequency-dependence of segmental activities and their interactions.*

In den beiden Arbeiten wird die Stimulation des Bewegungsapparates von querschnittsgelähmten Propanden untersucht. Dazu wird den Propanden eine Elektrode im Rückenmark platziert bei denen kurze Impulse abgegeben werden. Oberflächlich platzierte Elektromyographen (EMG) zeichnen die Aktivitäten der Muskeln (Quadriceps, Hamstrings, Tibialis anterior und Tirceps surae) in den Beinen auf.

Die elektrische Stimulation des unteren (lumbosakralen) Rückenmarks von Probanden mit kompletter Querschnittsläsion mittels rückenmarksnaher Im-

plantaten kann Aktivitäten in der gelähmten Beinmuskulatur generieren. Niedrige Stimulationsfrequenzen (2,1Hz) führen zu Muskelzuckungen, die als die einfachsten spinalen Reflexe identifiziert werden konnten und nur die Aktivität monosynaptischer Bahnen involvieren. Im Gegensatz dazu können höhere Stimulationsfrequenzen (25-50Hz) unwillkürliche, automatisierte schreitähnliche Bewegungen in den gelähmten Beinen der liegenden Propanden bewirken. [Hof09, S.2]

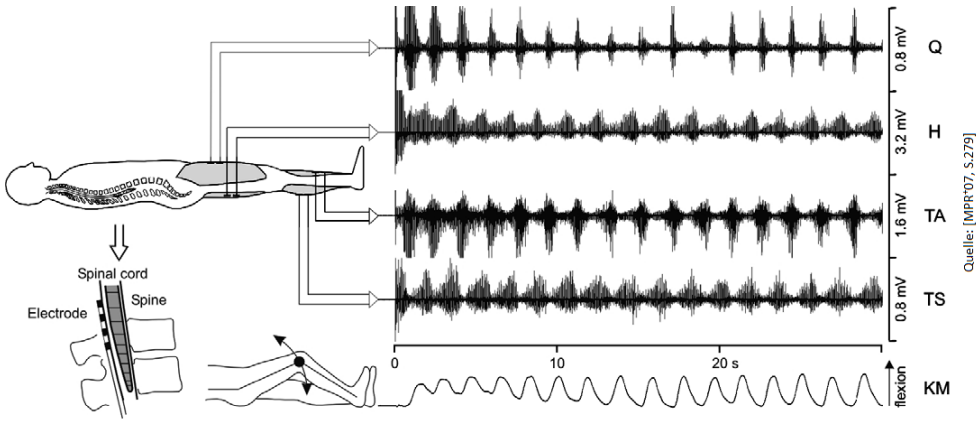

Abbildung 5.1: Skizzierung der Studie (links) und Beispiel für eine rhythmische schrittähnliche Bewegung der unteren Extremitäten (rechts). Oberflächliche Elektroden zur Messung der Spannung werden über dem Quadriceps (Q), Hamstrings (H), Tibialis anterior (TA) und Triceps surae (TS) platziert. Der Graph rechts unten zeigt die Kniebewegungen während einer Epiduralstimulation (10V, 25Hz).

5.2 Ergebnisse der Simulationen

In meiner Arbeit konzentriere ich mich vor allem auf die Stimulationsfrequenzen zwischen 25Hz und 50Hz. Zuerst werden einige Simulationen durchgeführt bei denen die Impulse in beiden RG-Zellen die gleiche Intensität haben. Danach halte ich die Impulse in der Flexor-RG-Zelle bei den Simulationen konstant auf einem gewissen Wert und die Impulse in der Extensor-RG-Zelle variieren. Im nächsten Schritt werden dann die Intensitäten in beiden RG-Zellen auf einen gewissen Wert festgelegt und die Frequenz der Impulse in einer der beiden Zellen wird in kleinen Schritten verändert. Zum Schluß führe ich zu den schon vorhandenen kurzen Impulsen einen langen konstanten Stimulus ein. In den folgenden Simulationen haben alle kurzen Impulse eine Dauer von 1ms.

In den folgenden Abbildungen sind nur mehr die RG-Neuronen abgebildet, da sich die Motoneuronen synchron zu den RG-Neuronen verhalten. Die Abbildungen zeigen immer zwei Neuronen, wobei der rote Graph den Extensor-RG-Neuron darstellt und der blaue den Flexor-RG-Neuron.

5.2.1 Gleiche Impulsintensität

In diesem Unterkapitel werden einersteits beide RG-Neuronen mit der gleichen Impulsintensität stimuliert und die Frequenz der Impulse varriert und andererseits bleibt die Frequenz gleich und die Intensität ändert sich. Diese Änderungen erfolgen in beiden RG-Zellen simultan. Da die Extensor- und Flexor-Zellen symmetrisch aufgebaut sind, stimuliere ich, um zu einem Ergebnis zu kommen, in allen Simulationen die Extensor-RG-Zellen um eine Periode früher. Nehmen wir 50Hz her, da beginne ich mit der Stimulation in der Extensor-RG-Zelle bei 1ms und in der Flexor-RG-Zelle erst bei 21ms. Würde man bei beiden Zellen gleichzeitig mit der Stimulation beginnen, würden sich beide Zellen synchron verhalten und es käme nicht zu dem gewünschten rhythmischen Verhalten.

Wenn die Impulsstärke gleichbleibend ist und man die Frequenz erhöht, kann man erkennen, dass die Länge der Phasen, in der jeweils eine der Extensor- bzw. Flexor-Zelle aktiv ist, sich vergrößert – siehe Abbildungen 5.2 bis 5.5 und

Tabelle 5.1.

Bei einer Frequenz von 25Hz und einer Impulsintensität von 0,16nA alternieren die Zellen bei jedem Impuls – siehe Abbildung 5.2. Bei gleicher Impulsintensität aber einer Frequenz von 40Hz beträgt die Feuerungsdauer bereits 875ms und bei 50Hz schon über 5000ms – siehe Abbildung 5.5. Erfolgen die Impulse also in immer kürzeren Abständen so verlängert sich die Phase in der die Extensor- bzw. Flexor-Zellen aktiv sind.

Im weiteren Verlauf der Arbeit verändere ich die Impulsintensität bei gleichbleibender Frequenz. Dabei können wir erkennen, dass wenn man die Intensität in einem gewissen Rahmen erhöht, sich die Feuerungsdauer verringert. Bei einer Frequenz von 50Hz beläuft sich die Dauer bei einer Intensität von 0,16nA auf über 5000ms, hingegen beträgt sie bei 0,17nA 2200ms – siehe Abbildungen 5.5 und 5.6. Dies kann man auch bei 33,3Hz erkennen, hierbei dauert die Feuerung bei 0,16nA 330ms und bei 0,165nA 180ms – siehe Abbildungen 5.3 und 5.7.

Frequenz	25Hz	28,8Hz	33,3Hz	40Hz	50Hz
Dauer	40ms	105ms	330ms	875ms	über 5000ms

Tabelle 5.1: Feuerungsdauer einer RG-Zelle in Abhängigkeit von der Frequenz, wobei die Impulsintensität gleichbleibend 0,16nA beträgt

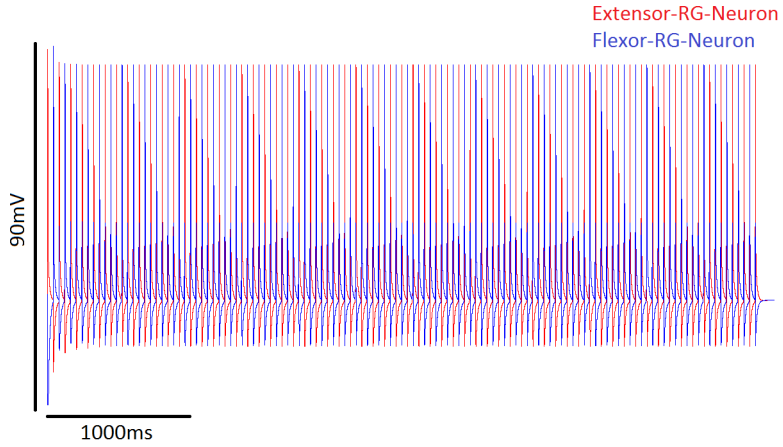

Abbildung 5.2: Frequenz: 25Hz, Impulsstärke: 0,16nA

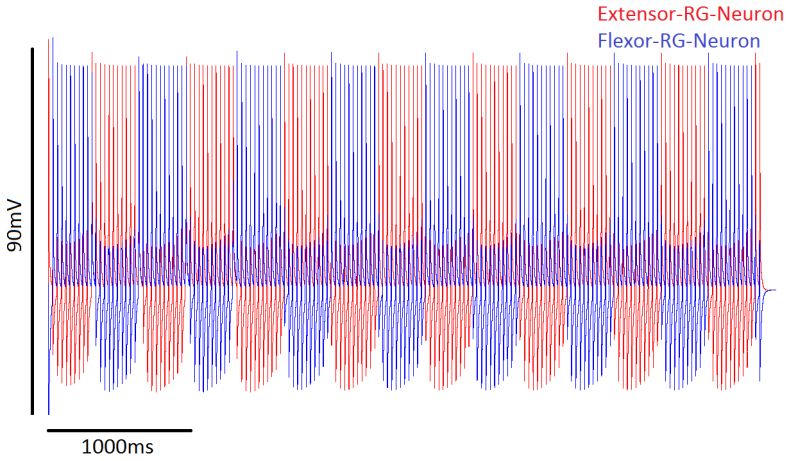

Abbildung 5.3: Frequenz: 33,3Hz, Impulsstärke: 0,16nA

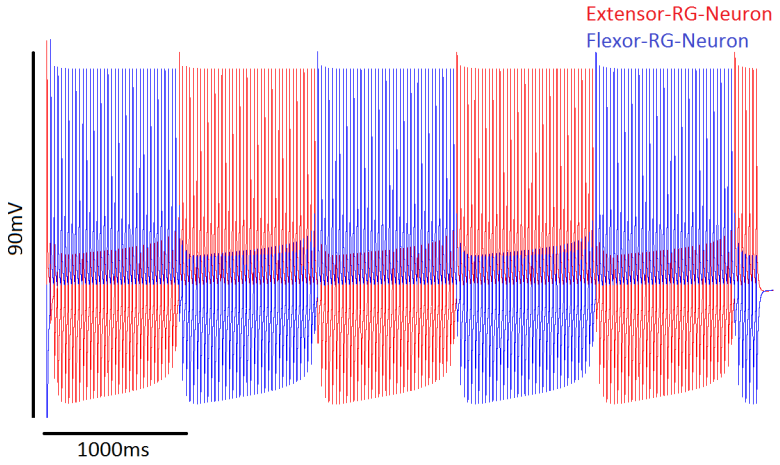

Abbildung 5.4: Frequenz: 40Hz, Impulsstärke: 0,16nA

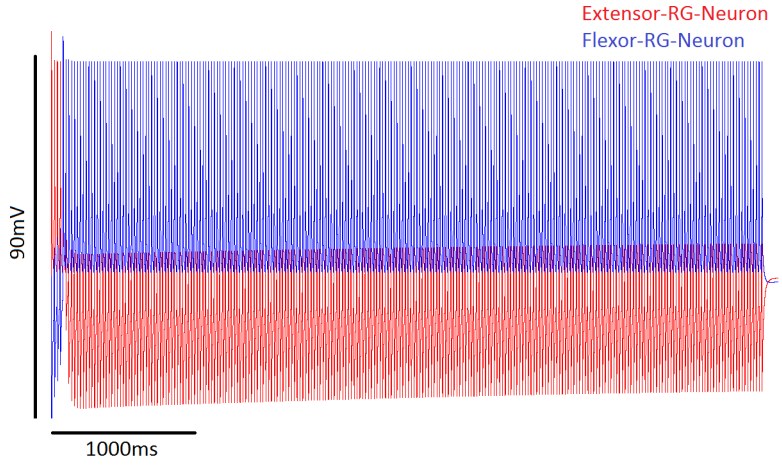

Abbildung 5.5: Frequenz: 50Hz, Impulsstärke: 0,16nA

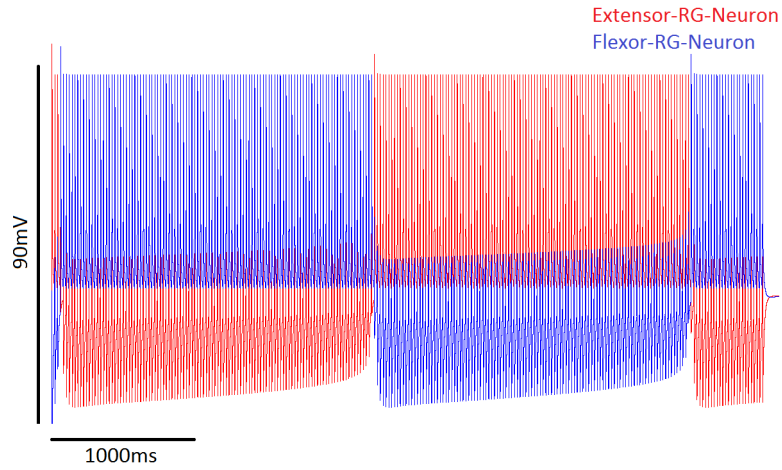

Abbildung 5.6: Frequenz: 50Hz, Impulsstärke: 0,17nA

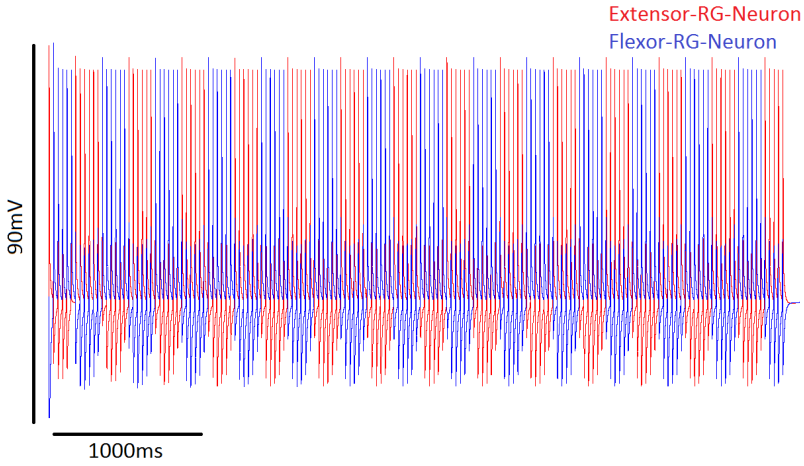

Abbildung 5.7: Frequenz: 33,3Hz, Impulsstärke: 0,165nA

5.2.2 Unterschiedliche Impulsintensität

Des Weiteren untersuche ich wie sich die Extensor- und Flexor-Zellen verhalten, wenn man die Intensität der Impulse in einer der beiden Zellen verändert. Da die Extensor- und die Flexor-Zellen im menschlichen Körper nicht vollkommen symmetrisch aufgebaut sind, kann es sein, dass die Impulsweiterleitung in einer der beiden Zellgruppen schneller erfolgt als bei der anderen. Da in meinem Modell die beiden Zellen vollkommen ident aufgebaut sind, kann ich diese schnellere Impulsweiterleitung durch die Veränderung der Intensität der Impulse in nur einer Zelle simulieren.

Hierzu wird bei einer Impulsfrequenz von 50Hz die Stärke in der Flexor-RG-Zelle konstant auf 0,17nA festgelegt und die Impulsstärke in der Extensor-RG-Zelle zwischen 0,161nA und 0,17nA variiert.

Wenn die Impulse in der Extensor-RG-Zelle eine Stärke von 0,161nA haben, wird diese gänzlich von der Flexor-RG-Zelle gehemmt und somit ist nur die Flexor-RG-Zelle aktiv. Vergrößert man den Impuls um 0,001nA, kann man bereits erkennen, dass auch die Extensor-RG-Zellgruppe angeregt wird. Jedoch ist die Phase in der die Flexor-RG-Zelle aktiv ist wesentlich größer als die in der die Extensor-RG-Zelle aktiviert ist. Je kleiner die Differenz zwischen den Impulsintensitäten ist, desto mehr nähern sich die Phasenlängen der beiden RG-Zellen an - siehe Abbildungen 5.8 bis 5.12 und Tabelle 5.2.

	0,161nA	0,162nA	0,163nA	0,164nA	0,165nA
Extensor	-	2380ms	2340ms	2260ms	2240ms
Flexor	über 10000ms	5760ms	4700ms	4050ms	3560ms

	0,166nA	0,167nA	0,168nA	0,169nA	0,17nA
Extensor	2220ms	2220ms	2200ms	2200ms	2200ms
Flexor	3200ms	2880ms	2620ms	2380ms	2200ms

Tabelle 5.2: Feuerungsdauer der Extensor- bzw.- Flexor-Zellen in Abhängigkeit von den Impulsintensitäten in der Extensor-Zelle bei gleichbleibender Intensität von 0,17nA in der Flexor-Zelle und bei konstanter Frequenz von 50Hz.

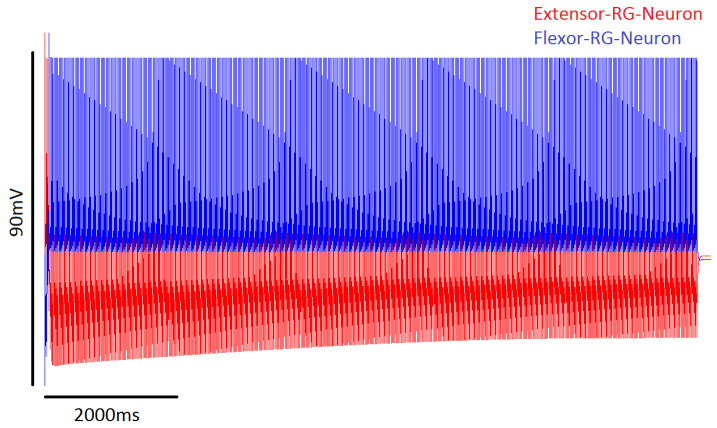

Abbildung 5.8: Frequenz: 50Hz, Impulsstärke: Extensor: 0,161nA, Flexor: 0,17nA

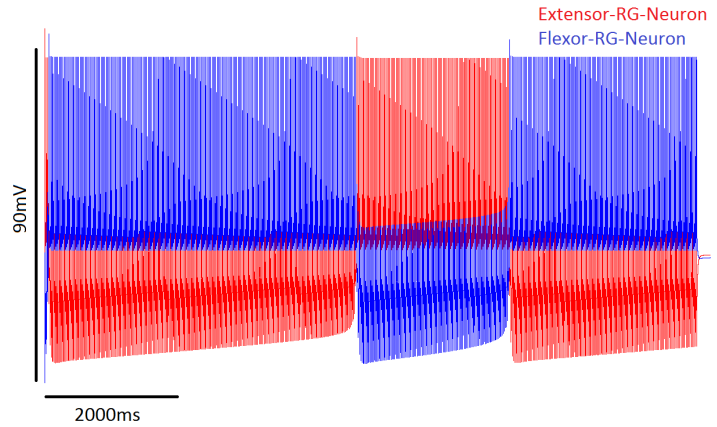

Abbildung 5.9: Frequenz: 50Hz, Impulsstärke: Extensor: 0,163nA, Flexor: 0,17nA

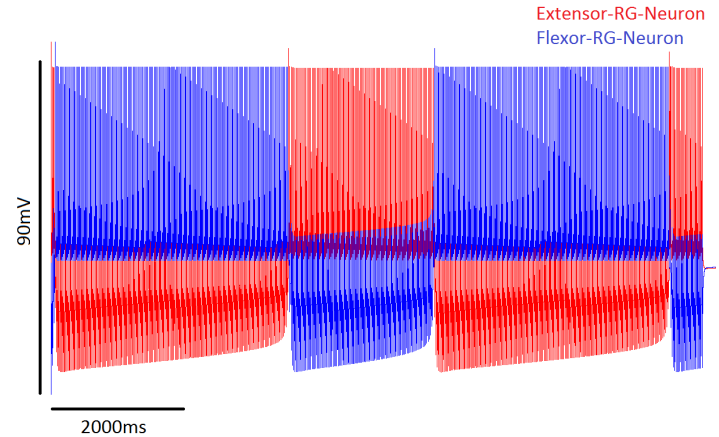

Abbildung 5.10: Frequenz: 50Hz, Impulsstärke: Extensor: 0,165nA, Flexorr: 0,17nA

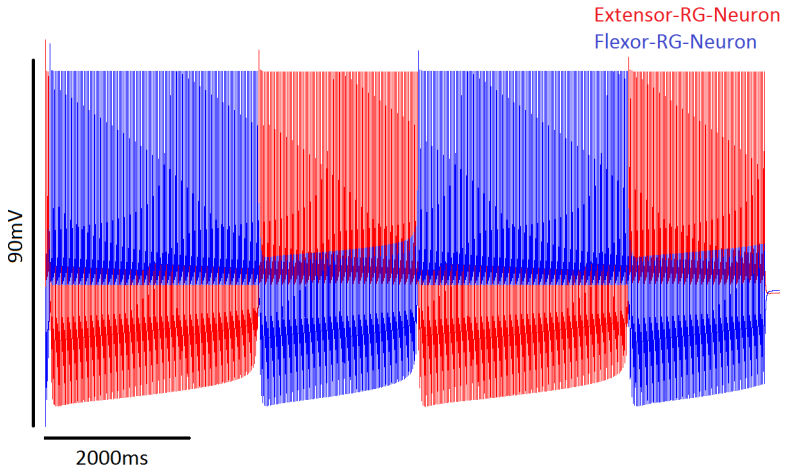

Abbildung 5.11: Frequenz: 50Hz, Impulsstärke: Extensorr: 0,167nA, Flexor: 0,17nA

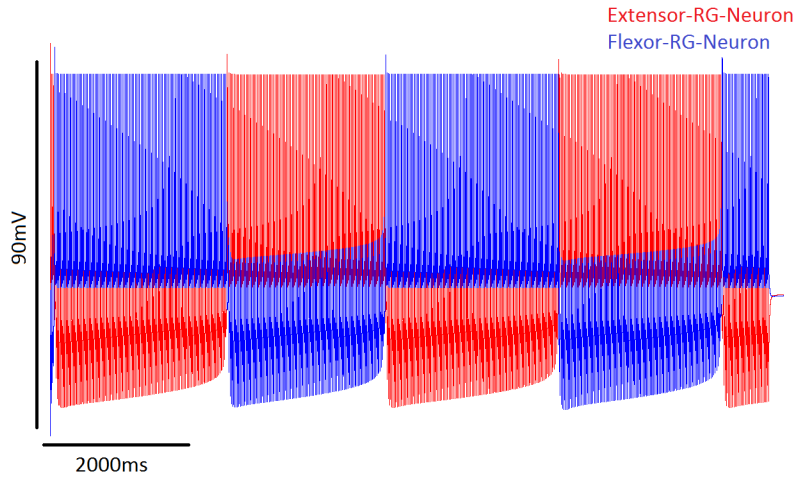

Abbildung 5.12: Frequenz: 50Hz, Impulsstärke: Extensor: 0,169nA, Flexor: 0,17nA

5.2.3 Unterschiedliche Frequenz

Man kann nicht nur die Impulsintensitäten verändern, sondern auch die Frequenzen. Durch unterschiedliche Frequenzen kann die unterschiedliche Impulsweiterleitung in den Extensor- und Flexor-Zellen ebenso simuliert werden wie durch unterschiedliche Impulsintensitäten. Man lässt also die Stärke der Impulse gleich und variiert die Frequenzen in einem sehr kleinen Rahmen.

In den Abbildungen 5.13 bis 5.18 sind folgende Werte gleichbleibend: die Stärke der Impulse: 0,16nA und die Frequenz in der Extensor-RG-Zelle: 50Hz. Die Frequenz in der Flexor-RG-Zelle varriert zwischen 48,26Hz und 49,87Hz. Dies entspricht einem Abstand zwischen den Impulsen von 20,3ms bis 20,05ms.

Eigentlich ist ein ähnliches Ergebnis bei dieser Simulation wie bei der Simulation mit unterschiedlicher Impulsintensität zu erwarten. Die Phasenlängen der jeweiligen Feuerung nähern sich zwar an, je kleiner die Differenz wird, jedoch verläuft das nicht so gleichmäßig wie bei den Impulsintensitäten – siehe Tabelle 5.3. Man kann erkennen, dass bei einer Differenz von 0,74Hz die Feuerungsdauer der Extensor-RG-Zelle (9500ms) wesentlich länger ist als die der Flexor-RG-Zelle (3920ms).

	49,26Hz	49,38Hz	49,5Hz	49,63Hz	49,75Hz	49,87Hz
Extensor	9500ms	8300ms	8240ms	8300ms	8420ms	8180ms
Flexor	3920ms	4660ms	3900ms	4140ms	7680ms	7180ms
Differenz	5580ms	3640ms	4340ms	4160ms	740ms	1000ms

Tabelle 5.3: Feuerungsdauer in der Extensor- bzw. Flexor-Zelle bei gleichbleibender Frequenz von 50Hz in der Extensor-RG-Zelle und bei konstanter Impulsstärke von 0,16nA in beiden Zellen sowie die Differenz zwischen den Dauern

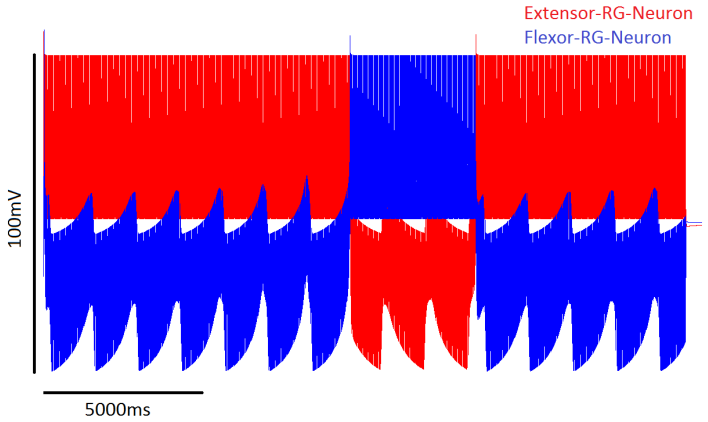

Abbildung 5.13: Frequenz: Extensor: 50Hz, Flexor: 49,26Hz, Impulsstärke: 0,16nA

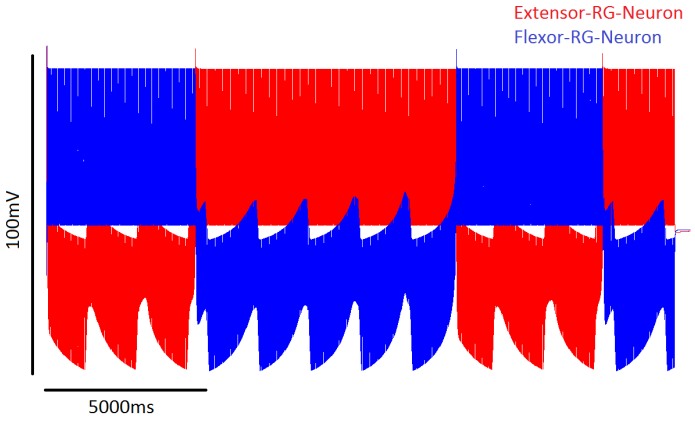

Abbildung 5.14: Frequenz: Extensor: 50Hz, Flexor: 49,38Hz, Impulsstärke: 0,16nA

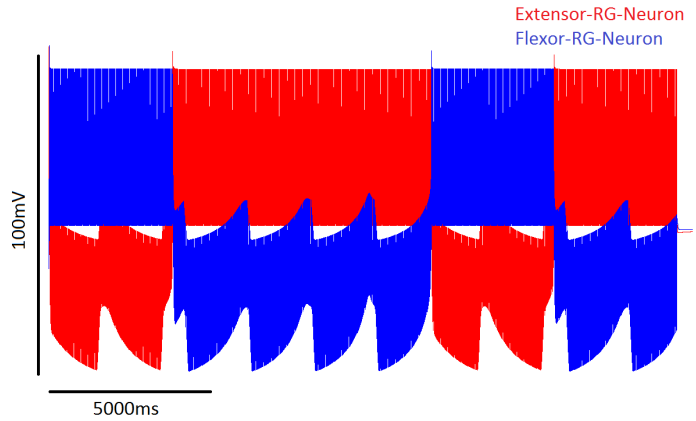

Abbildung 5.15: Frequenz: Extensor: 50Hz, Flexor: 49,5Hz, Impulsstärke: 0,16nA

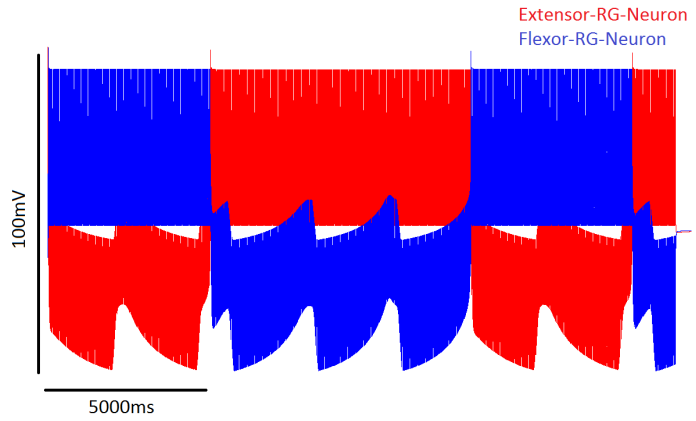

Abbildung 5.16: Frequenz: Extensor: 50Hz, Flexor: 49,63Hz, Impulsstärke: 0,16nA

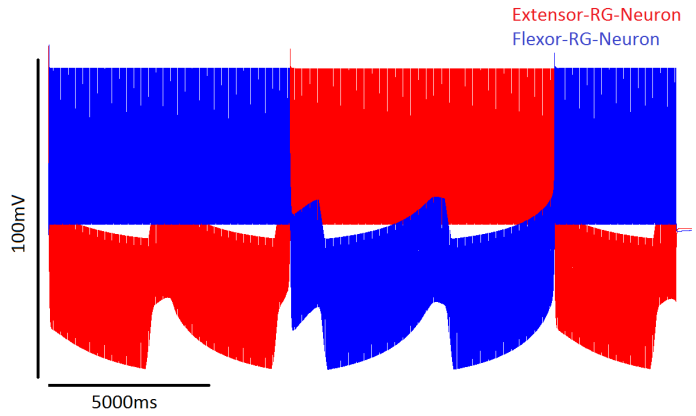

Abbildung 5.17: Frequenz: Extensor: 50Hz, Flexor: 49,75Hz, Impulsstärke: 0,16nA

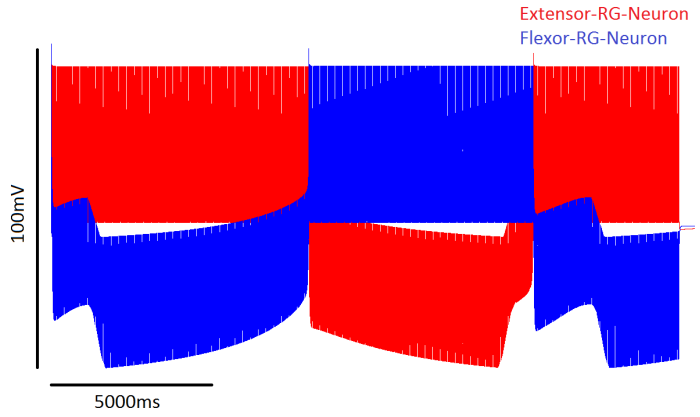

Abbildung 5.18: Frequenz: Extensor: 50Hz, Flexor: 49,87Hz, Impulsstärke: 0,16nA

5.2.4 Zusätzlicher konstanter langer Impuls

In diesem Unterkapitel konzentriere ich mich auf die Impulsstärke von 0,15nA in beiden Zellen bei einer Frequenz von 50Hz. Bei dieser Impulsstärke beginnen die Zellen sich abwechselnd zu aktivieren jedoch sind die beiden bereits nach kurzer Zeit gleichzeitig aktiv. Dies ist jedoch ein Phänomen, das in der Realität so gut wie nie vorkommt, da es im Allgemeinen eine Energieverschwendung ist. Um dieses Problem zu lösen, habe ich einen gleichbleibenden 5000ms langen Stimulus appliziert. Durch diesen zusätzlichen Impuls, der einer geringen Anhebung der Ruhespannung entspricht, beginnen die RG-Zellen zu alternieren. Dieser langanhaltende Impuls ist so gering, dass er alleine die Zellen nicht aktivieren kann.

Bei einer Frequenz von 50Hz und einer Impulsintensität von 0,15nA reicht bereits ein 5000ms andauernder Stimulus von $2 \cdot 10^{-4}$nA um die RG-Zellen abwechselnd zu aktivieren. Jedoch variieren hierbei die Längen der Perioden der Feuerung innerhalb der simulierten 5000ms. Erhöht man den langen Stimulus auf $2{,}4 \cdot 10^{-4}$nA sind die Feuerungsdauern bereits gleich lang. Wenn man den zusätzlichen Simulus noch ein wenig erhöht (um $4 \cdot 10^{-5}$nA), wird die Extensor-RG-Zelle von der Flexor-RG-Zelle gänzlich gehemmt. Die Ergebnisse dieser Simulation kann man in Abbildungen 5.19 bis 5.24 sehen.

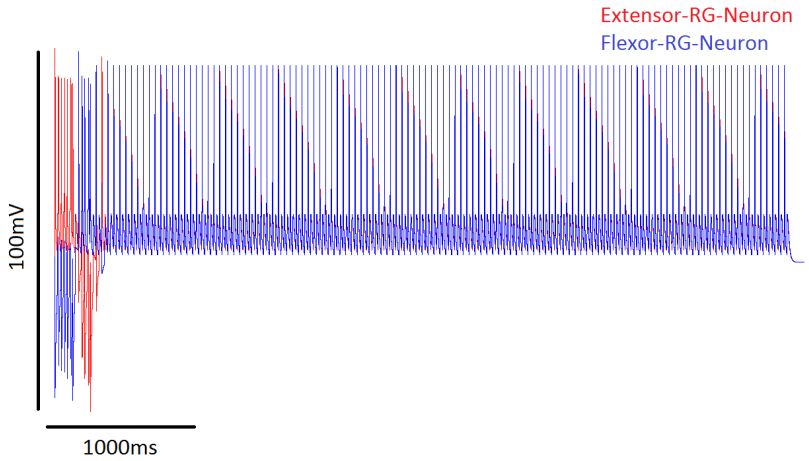

Abbildung 5.19: Frequenz: 50Hz, Impulsstärke: 0,15nA

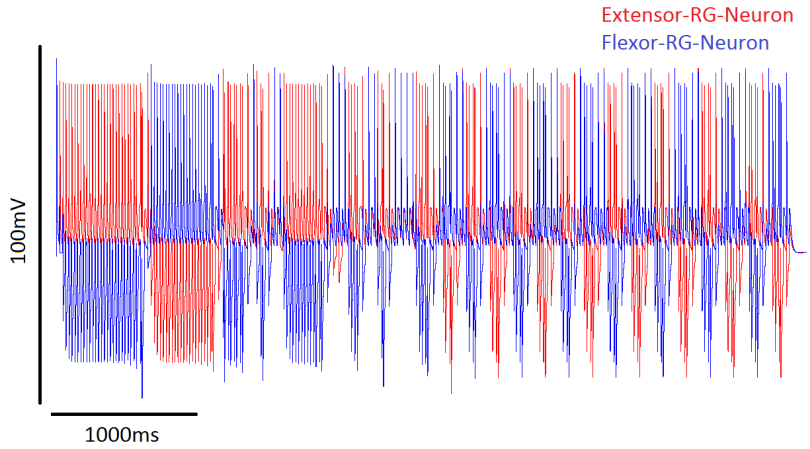

Abbildung 5.20: Frequenz: 50Hz, Impulsstärke: kurze Impulse: 0,15nA, langer Stimulus: $2 \cdot 10^{-4}$nA

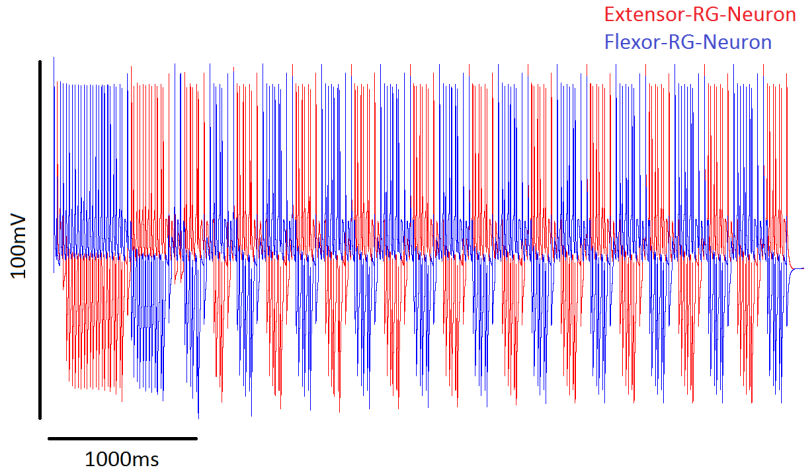

Abbildung 5.21: Frequenz: 50Hz, Impulsstärke: kurze Impulse: 0,15nA, langer Stimulus: $2{,}2 \cdot 10^{-4}$nA

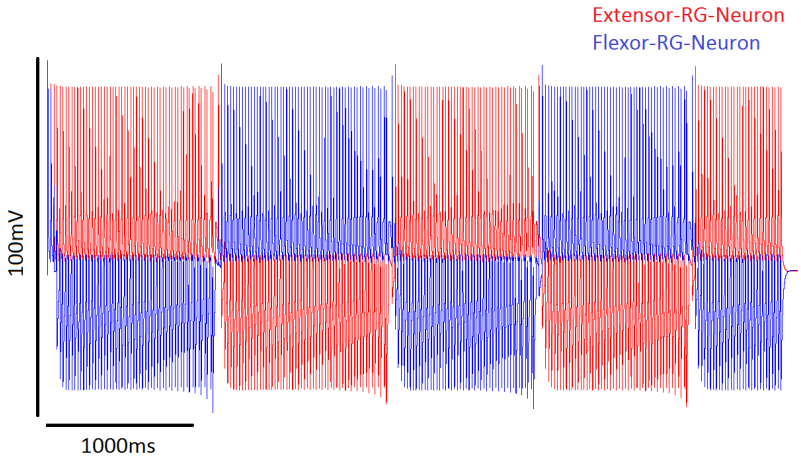

Abbildung 5.22: Frequenz: 50Hz, Impulsstärke: kurze Impulse: 0,15nA, langer Stimulus: $2{,}4 \cdot 10^{-4}$nA

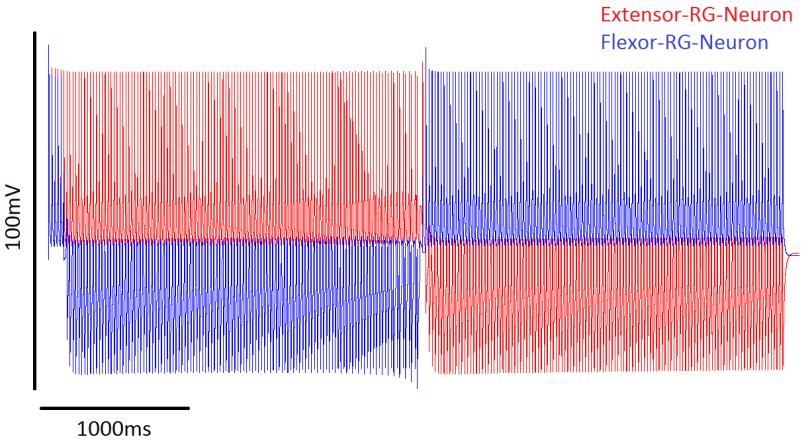

Abbildung 5.23: Frequenz: 50Hz, Impulsstärke: kurze Impulse: 0,15nA, langer Stimulus: $2{,}6 \cdot 10^{-4}$nA

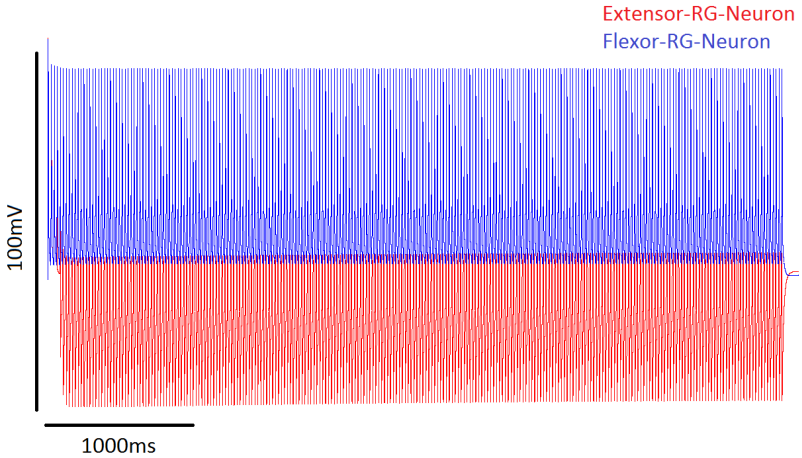

Abbildung 5.24: Frequenz: 50Hz, Impulsstärke: kurze Impulse: 0,15nA, langer Stimulus: $2{,}8 \cdot 10^{-4}$nA

Kapitel 6

Zusammenfassung und Vorschau

In meiner Diplomarbeit habe ich mich vor allem auf die Stimulation von Rhythm Generator Neuronen konzentriert. Diese Neuronen sind wichtig da unter anderem durch sie der Bewegungsapparat gesteuert wird. Die menschlichen Nervenfasern sind natürlich sehr komplex aufgebaut und bestehen aus vielen einzelnen Nervenzellen. Ich habe diese sehr komplexe Vernetzung von Nervenzellen auf ein sehr einfaches Modell reduziert, in dem nur mehr 10 einzelne Nervenzellen direkt verbunden sind.

Zu Beginn habe ich mich mit dem Programm NEURON vertraut gemacht und habe dabei erkannt, dass dieses Programm sehr hilfreich bei der Implementierung von Nervenzellen ist, jedoch hat es so seine Tücken. Im dritten Kapitel meiner Arbeit habe ich versucht das Programm NEURON möglichst gut zu beschreiben, damit der Einstieg zur Benutzung des Programms leichter fällt.

Zunächst habe ich mein Modell in NEURON implementiert und einen lang anhaltenden gleichbleibenden Impuls appliziert. Dieser konstante Impuls entspricht der Steuerung des Bewegungsapparates. Bei diesen Simulationen habe ich die wichtige Bedeutung des persistenten Natrium-Kanals erkannt. Dieser Kanal ist ausschlaggebend für das rhytmische Verhalten der RG-Neuronen. Des Weiteren habe ich belegt, dass bei steigender Impulsstärke die Dauer in der die Extensor- bzw. Flexor-Zellen aktiv sind ebenfalls zunimmt.

Im nächsten Schritt habe ich den lang anhaltenden Impuls durch kurze

Impulse, von einer Frequenz zwischen 25Hz und 50Hz, ersetzt. Diese kurzen Impulse bewirken in der Realität unwillkürliche, schrittähnliche Bewegungen in den Beinen von Propanden mit kompletter Querschnittsläsion. In meiner Arbeit kann man sehen, dass auch durch die kurzen Impulse ein ähnliches Muster in den RG-Neuronen erzeugt werden kann wie durch einen lang anhaltenden Impuls.

In den Simulationen erkennt man, dass ein Zusammenhang zwischen der Frequenz der Impulse und der Feuerungsdauer der RG-Neuronen besteht. Erhöht man die Frequenz so verlängert sich die Feuerungsdauer. Um der Realität ein klein wenig näher zu kommen werden die Extensor- und die Flexor-Zelle nicht synchron stimuliert. Hierzu verändert man die Impulsintensität oder die Frequenz in einer der beiden Zellen. Dadurch kann man die unterschiedlichen Impulsweiterleitung in der Extensor- bzw. Flexor-Zellgruppe simulieren. Je größer die Differenz zwischen den Impulsintensitäten bzw. den Frequenzen ist, desto wesentlicher ist der Unterschied zwischen den Feuerungsdauern der Extensor- bzw. Flexor-Zellen.

Abschließend möchte ich noch ein paar mögliche Ansatzpunkte für folgende Arbeiten anführen:

Einerseits könnte man das von mir verwendete einfache Modell an das Modell von [RSLR06] anpassen und dabei ein besonderes Augenmerk auf die Motoneuronen legen. Andererseite wäre es interessant, die Zellen, die in meinem Modell Repräsentanten für ganze Zellgruppen sind, durch Zellgruppen zu ersetzen und zu untersuchen wie die Zellgruppen auf die Stimulation reagieren.

Anhang A

Source-Codes

Hier führe ich die beiden `mod`-Dateien an, die die RG-Neuronen (`rgsoma.mod`) und die Motoneuronen (`mnsoma.mod`) definieren.

`rgsoma.mod`-Datei

```
1   //Definition der Einheiten:
2   UNITS {
3     (mV) = (millivolt)
4     (mA) = (milliamp)
5     (S) = (siemens)
6   }
7
8   NEURON {
9     SUFFIX rgsoma
10  //Festlegung der verwendeten Kanaele:
11    USEION Na READ eNa WRITE iNa VALENCE 1
12    USEION K READ eK WRITE iK VALENCE 1
13    USEION NaP WRITE iNaP VALENCE 1
14    NONSPECIFIC_CURRENT iLeak
15    RANGE gKbar, gK
16    RANGE gNabar, gNa
```

```
17    RANGE gNaPbar, gNaP
18    RANGE gLeakbar, eLeak
19  }
20
21  //Festlegung der Parameter mit den dazugehoerigen Einheiten:
22  PARAMETER {
23    gNabar = 0.030 (S/cm2)
24    gNaPbar = 0.00025 (S/cm2)
25    gKbar = 0.036 (S/cm2)
26    gLeakbar = 0.0001 (S/cm2)
27    eLeak = -64 (mV)
28  }
29
30  //Festlegung der Zustandsvariablen:
31  STATE {
32    mNa  hNa  mNaP  hNaP  mK
33  }
34
35  //Zuweisung der Einheiten an die Variablen:
36  ASSIGNED {
37    v (mV)
38    eNa (mV)
39    iNa (mA/cm2)
40    iNaP (mA/cm2)
41    eK (mV)
42    iK (mA/cm2)
43    gNa (S/cm2)
44    gNaP (S/cm2)
45    gK (S/cm2)
46    mNainf
47    mNatau (ms)
```

```
48      hNainf
49      hNatau (ms)
50      mNaPinf
51      mNaPtau (ms)
52      hNaPinf
53      hNaPtau (ms)
54      mKinf
55      mKtau (ms)
56      iLeak (mA/cm2)
57    }
58
59    BREAKPOINT {
60    //Methode zur Loesung der Differentialgleichungen
61      SOLVE states2 METHOD cnexp
62
63    //Berechnung der verschiedenen Stroeme
64      gNa = gNabar*mNa*mNa*mNa*hNa
65      iNa = gNa*(v-eNa)
66      gNaP = gNaPbar*mNaP*hNaP
67      iNaP = gNaP*(v-eNa)
68      gK = gKbar*mK*mK*mK*mK
69      iK = gK*(v-eK)
70      iLeak = gLeakbar*(v-eLeak)
71    }
72
73    //Festlegung der Anfangsbedingungen
74    INITIAL {
75      rates3(v)
76
77      mNa = mNainf
78      hNa = hNainf
```

```
79      mNaP = mNaPinf
80      hNaP = hNaPinf
81      mK = mKinf
82      eNa = 55
83      eK = -80
84    }
85
86    //Definition der Differentialgleichungen
87    DERIVATIVE states2 {
88      rates3(v)
89
90      mNa' = (mNainf-mNa)/mNatau
91      hNa' = (hNainf-hNa)/hNatau
92      mNaP' = (mNaPinf-mNaP)/mNaPtau
93      hNaP' = (hNaPinf-hNaP)/hNaPtau
94      mK' = (mKinf-mK)/mKtau
95    }
96
97    UNITSOFF
98
99    //Berechnung der Variablen in den Differentialgleichungen -
100   //hierbei ist es wichitg darauf zu achten, dass man die Einheiten-
101   //uebereinstimmungspruefung mit UNITSOFF ausschaltet:
102   PROCEDURE rates3(v(mV)) {
103     TABLE mNainf, mNatau, hNainf, hNatau, mNaPinf, mNaPtau, hNaPinf,
104      hNaPtau, mKtau, mKinf FROM -100 TO 100 WITH 32000
105
106     mNainf = NaPvtrapB(-(v+35),7.8)
107     mNatau = 1e-6
108     hNainf = NaPvtrapB((v+55),7)
109     hNatau = 30/(NaPeFun((v+50),15)+NaPeFun(-(v+50),16))
```

```
110     mNaPinf = NaPvtrapB(-(v+47.1),3.1)
111     mNaPtau = 1e-2
112     hNaPinf = NaPvtrapB((v+59),8)
113     hNaPtau = 1200/(NaPeFun((v+59),16)+NaPeFun(-(v+59),16))
114     mKinf = NaPvtrapB(-(v+28),15)
115     mKtau = 7/(NaPeFun((v+40),40)+NaPeFun(-(v+40),50))
116   }
117
118   FUNCTION NaPvtrapB(x,y) {
119     LOCAL a
120     a = x/y
121     if (fabs(a)<1e-4) {
122         NaPvtrapB = 1/(2+a)
123     }else{
124         NaPvtrapB = 1/(exp(a)+1)
125     }
126   }
127
128   FUNCTION NaPeFun(x,y) {
129     LOCAL a
130     if (fabs(a)<1e-4) {
131         NaPeFun = 1+a
132     }else{
133         NaPeFun = exp(a)
134     }
135   }
136
137   UNITSON
```

mnsoma.mod-Datei

Diese Datei ist genauso aufgebaut, wie die vorangegangene.

```
1   UNITS {
2     (mV) = (millivolt)
3     (mA) = (milliamp)
4     (S) = (siemens)
5   }
6
7   NEURON {
8     SUFFIX mnsoma43
9     USEION Ca READ eCa WRITE iCa VALENCE 1
10    USEION Na READ eNa WRITE iNa VALENCE 1
11    USEION K READ eK WRITE iK VALENCE 1
12    USEION KCa WRITE iKCa VALENCE 1
13    NONSPECIFIC_CURRENT iLeak
14    RANGE gKbar, gK, gKCabar, gKCa
15    RANGE gNabar, gNa
16    RANGE gCaNbar, gCaN
17    RANGE gLeakbar, eLeak
18  }
19
20  PARAMETER {
21    alpha = 0.9 (mol/C/mm)
22    kCa = 2 (/ms)
23    gKCabar = 0.00003 (S/cm2)
24    gNabar = 0.01 (S/cm2)
25    gCaNbar = 0.0001 (S/cm2)
26    gKbar = 0.0003 (S/cm2)
27    gLeakbar = 0.00051 (S/cm2)
28    eLeak = -65 (mV)
```

```
29  }
30
31  STATE {
32    Ca  mNa  hNa  mCaN  hCaN  mK
33  }
34
35  ASSIGNED {
36    v (mV)
37    eCa (mV)
38    iCa (mA/cm2)
39    eNa (mV)
40    iNa (mA/cm2)
41    eK (mV)
42    iK (mA/cm2)
43    iKCa (mA/cm2)
44    gKCa (S/cm2)
45    gK (S/cm2)
46    gNa (S/cm2)
47    gCaN (S/cm2)
48    oldt (ms)
49    oldv (mV)
50    mNainf
51    mNatau (ms)
52    hNainf
53    hNatau (ms)
54    mCaNinf
55    mCaNtau (ms)
56    hCaNinf
57    hCaNtau (ms)
58    mKinf
59    mKtau (ms)
```

```
60    iLeak (mA/cm2)
61  }
62
63  BREAKPOINT {
64    SOLVE states2 METHOD cnexp
65
66    gNa = gNabar*mNa*mNa*mNa*hNa
67    iNa = gNa*(v-eNa)
68    gCaN = gCaNbar*mCaN*mCaN*hCaN
69    iCa = gCaN*(v-eCa)
70    gK = gKbar*mK*mK*mK*mK
71    iK = gK*(v-eK)
72    gKCa = gKCabar*Ca/(Ca+0.2)
73    iKCa = gKCabar*(v-eK)
74    iLeak = gLeakbar*(v-eLeak)
75  }
76
77  INITIAL {
78    rates3(v)
79
80    Ca = -alpha*iCa/kCa
81    mNa = mNainf
82    hNa = hNainf
83    mCaN = mCaNinf
84    hCaN = hCaNinf
85    mK = mKinf
86    eNa = 55
87    eCa = 80
88    eK = -80
89  }
90
```

```
91  DERIVATIVE states2 {
92    rates3(v)
93
94    Ca' = 0.01(-alpha*iCa-kCa*Ca)
95    mNa' = (mNainf-mNa)/mNatau
96    hNa' = (hNainf-hNa)/hNatau
97    mCaN' = (mCaNinf-mCaN)/mCaNtau
98    hCaN' = (hCaNinf-hCaN)/hCaNtau
99    mK' = (mKinf-mK)/mKtau
100 }
101
102 UNITSOFF
103
104 PROCEDURE rates3(v(mV)) {
105   TABLE mNainf, mNatau, hNainf, hNatau, mCaNinf, mCaNtau,
106    hCaNinf, hCaNtau, mKinf, mKtau FROM -100 TO 100 WITH 32000
107
108   mNainf = NaPvtrapB(-(v+35),7.8)
109   mNatau = 1e-2
110   hNainf = NaPvtrapB((v+55),7)
111   hNatau = 30/(NaPeFun((v+50),15)+NaPeFun(-(v+50),16))
112   mCaNinf = NaPvtrapB(-(v+30),5)
113   mCaNtau = 4
114   hCaNinf = NaPvtrapB((v+45),5)
115   hCaNtau = 40
116   mKinf = NaPvtrapB(-(v+28),15)
117   mKtau = 7/(NaPeFun((v+40),40)+NaPeFun(-(v+40),50))
118 }
119
120 FUNCTION NaPvtrapB(x,y) {
121   LOCAL a
```

```
122     a = x/y
123     if (fabs(a)<1e-4) {
124         NaPvtrapB = 1/(2+a)
125     }else{
126         NaPvtrapB = 1/(exp(a)+1)
127     }
128   }
129
130   FUNCTION NaPeFun(x,y) {
131     LOCAL a
132     a=x/y
133     if (fabs(a)<1e-4) {
134         NaPeFun = 1+a
135     }else{
136         NaPeFun = exp(a)
137     }
138   }
139
140   UNITSON
```

Zellen.hoc-Datei

Die nachfolgenden Zeilen sind die hoc-Datei, in der zuerst die verschiedenen Neuronen des von mir verwendeten Modells definiert und verbunden werden. Anschließend platziere ich jeweils einen langen Impuls in die RG-Neuronen sowie die kurzen Impulse mit einer Frequenz von 50Hz.

```
1   load_file("nrngui.hoc")
2
3   //Definition der RG-Neuronen
4   begintemplate RGCell
5   public soma, nclist
```

```
6   create soma
7   objectvar nclist
8   proc init() {
9   create soma
10  //Erstellung einer Liste, diese ist wichtig um die Zellen
11  //zu verbinden
12  nclist = new List()
13  soma {
14        nseg = 1
15        diam = 20
16        L = 20
17        Ra=35.4
18        cm = 1
19        insert rgsoma {eNa=55 eK=-80}
20        gLeakbar_rgsoma = 0.0001
21        eLeak_rgsoma = -64
22        gNabar_rgsoma = 0.03
23        gKbar_rgsoma = 0.036
24        gNaPbar_rgsoma = 0.00025
25      }
26  }
27  endtemplate RGCell
28
29  objectvar preRGcells[2]
30  for i = 0, 1 {
31  preRGcells[i] = new RGCell()
32  }
33
34  //Definition der Interneuronen
35  begintemplate ICell
36  public soma, nclist
```

```
37  create soma
38  objectvar nclist
39  proc init() {
40  create soma
41  nclist = new List()
42  soma {
43        nseg = 1
44        diam = 20
45        L = 20
46        Ra = 35.4
47        cm = 1
48        insert hh
49        gLeakbar_hh = 0.00051
50        eLeak_hh = 57.5
51        gNabar_hh = 0.12
52        gKbar_hh = 0.1
53       }
54  }
55  endtemplate ICell
56
57  objectvar preIcells[2]
58  for i = 0,1 {
59  preIcells[i] = new ICell()
60  }
61
62  //Definition der Motoneuronen
63  begintemplate MNCell
64  public soma, nclist
65  create soma
66  objectvar nclist
67  proc init() {
```

```
68  create soma
69  nclist = new List()
70  soma {
71        nseg = 1
72        diam = 20
73        L = 20
74        Ra = 35.4
75        cm = 1
76        insert mnsoma {eNa=55 eK=-80 eCa=80}
77       }
78  }
79  endtemplate MNCell
80
81  objectvar preMNcells[2]
82  for i = 0,1 {
83  preMNcells[i] = new MNCell()
84  }
85
86  //Definition der IA-Neuronen
87  begintemplate IACell
88  public soma, nclist
89  create soma
90  objectvar nclist
91  proc init() {
92  create soma
93  nclist = new List()
94  soma {
95        nseg = 1
96        diam = 20
97        L = 20
98        Ra = 35.4
```

```
99          cm = 1
100         insert hh
101         gLeakbar_hh = 0.00051
102         eLeak_hh = -64
103         gNabar_hh = 0.12
104         gKbar_hh = 0.1
105        }
106   }
107   endtemplate IACell
108
109   objectvar preIAcells[4]
110   for i = 0,3 {
111   preIAcells[i] = new IACell()
112   }
113
114   objectvar syn[24]
115
116   //Verbindung der Zellen mittels einer Synapse
117
118   preRGcells[1].soma syn[0] = new ExpSyn(1)
119   preRGcells[0].soma preRGcells[1].nclist.append
120     (new NetCon(&v(1), syn[0], -20, 0, 0.01))
121   //Die Zahlen -20, 0, 0.01 geben die Schwelle, die Verzoegerung
122   //und die Gewichtung an.
123
124   preRGcells[0].soma syn[1] = new ExpSyn(1)
125   preRGcells[1].soma preRGcells[0].nclist.append
126     (new NetCon(&v(1), syn[1], -20, 0, 0.01))
127
128   preRGcells[0].soma syn[2] = new ExpSyn(1)
129   preRGcells[0].soma preRGcells[0].nclist.append
```

```
130     (new NetCon(&v(1), syn[2], -20, 0, 0.01))
131
132   preRGcells[1].soma syn[3] = new ExpSyn(1)
133   preRGcells[1].soma preRGcells[1].nclist.append
134     (new NetCon(&v(1), syn[3], -20, 0, 0.01))
135
136   preIcells[1].soma syn[4] = new ExpSyn(1)
137   preRGcells[0].soma preIcells[1].nclist.append
138     (new NetCon(&v(1), syn[4], -20, 0, 0.5))
139
140   preIcells[0].soma syn[5] = new ExpSyn(1)
141   preRGcells[1].soma preIcells[0].nclist.append
142     (new NetCon(&v(1), syn[5], -20, 0, 0.5))
143
144   preRGcells[0].soma syn[6] = new ExpSyn(1)
145   preIcells[0].soma preRGcells[0].nclist.append
146     (new NetCon(&v(1), syn[6], -20, 0, -0.06275))
147
148   preRGcells[1].soma syn[7] = new ExpSyn(1)
149   preIcells[1].soma preRGcells[1].nclist.append
150     (new NetCon(&v(1), syn[7], -20, 0, -0.06275))
151
152   preMNcells[0].soma syn[8] = new ExpSyn(1)
153   preRGcells[0].soma preMNcells[0].nclist.append
154     (new NetCon(&v(1), syn[8], -20, 0, 0.1))
155
156   preMNcells[1].soma syn[9] = new ExpSyn(1)
157   preRGcells[1].soma preMNcells[1].nclist.append
158     (new NetCon(&v(1), syn[9], -20, 0, 0.1))
159
160   preIAcells[0].soma syn[10] = new ExpSyn(1)
```

```
161  preRGcells[0].soma preIAcells[0].nclist.append
162    (new NetCon(&v(1), syn[10], -20, 0, 0.4))
163
164  preIAcells[1].soma syn[11] = new ExpSyn(1)
165  preRGcells[1].soma preIAcells[1].nclist.append
166    (new NetCon(&v(1), syn[11], -20, 0, 0.4))
167
168  preMNcells[1].soma syn[12] = new ExpSyn(1)
169  preIAcells[0].soma preMNcells[1].nclist.append
170    (new NetCon(&v(1), syn[12], -20, 0, -0.05))
171
172  preMNcells[0].soma syn[13] = new ExpSyn(1)
173  preIAcells[1].soma preMNcells[0].nclist.append
174    (new NetCon(&v(1), syn[13], -20, 0, -0.05))
175
176  preIAcells[2].soma syn[14] = new ExpSyn(1)
177  preMNcells[0].soma preIAcells[2].nclist.append
178    (new NetCon(&v(1), syn[14], -20, 0, 0.25))
179
180  preIAcells[3].soma syn[15] = new ExpSyn(1)
181  preMNcells[1].soma preIAcells[3].nclist.append
182    (new NetCon(&v(1), syn[15], -20, 0, 0.25))
183
184  preMNcells[0].soma syn[16] = new ExpSyn(1)
185  preIAcells[2].soma preMNcells[0].nclist.append
186    (new NetCon(&v(1), syn[17], -20, 0, -0.01))
187
188  preIAcells[0].soma syn[17] = new ExpSyn(1)
189  preIAcells[2].soma preIAcells[0].nclist.append
190    (new NetCon(&v(1), syn[18], -20, 0, -0.01))
191
```

```
192  preMNcells[1].soma syn[18] = new ExpSyn(1)
193  preIAcells[2].soma preMNcells[1].nclist.append
194    (new NetCon(&v(1), syn[19], -20, 0, -0.02))
195
196  preMNcells[1].soma syn[19] = new ExpSyn(1)
197  preIAcells[3].soma preMNcells[1].nclist.append
198    (new NetCon(&v(1), syn[19], -20, 0, -0.02))
199
200  preIAcells[1].soma syn[20] = new ExpSyn(1)
201  preIAcells[3].soma preIAcells[1].nclist.append
202    (new NetCon(&v(1), syn[20], -20, 0, -0.01))
203
204  preMNcells[0].soma syn[21] = new ExpSyn(1)
205  preIAcells[3].soma preMNcells[0].nclist.append
206    (new NetCon(&v(1), syn[21], -20, 0, -0.02))
207
208  preMNcells[0].soma syn[22] = new ExpSyn(1)
209  preIcells[0].soma preMNcells[0].nclist.append
210    (new NetCon(&v(1), syn[22], -20, 0, -0.01))
211
212  preMNcells[1].soma syn[23] = new ExpSyn(1)
213  preIcells[1].soma preMNcells[1].nclist.append
214    (new NetCon(&v(1), syn[23], -20, 0, -0.01))
215
216  objectvar stim[501]
217
218  //Festlegung der beiden langen Imulse in den RG-Neuronen
219
220  preRGcells[0].soma {
221  stim[0] = new IClamp(0.5)
222  stim[0].dur = 5000
```

```
223   stim[0].amp = 0.005
224   }
225
226   preRGcells[1].soma {
227   stim[1] = new IClamp(0.5)
228   stim[1].dur = 5000
229   stim[1].amp = 0.005
230   stim[1].del = 1
231   }
232
233   //Erstellung einer Prozedur um die Impulsstaerke der kurzen
234   //Impulse zu varrieren
235
236   proc ver() {
237   for i= 2, 251 preRGcells[0].soma {
238   stim[i] = new IClamp(0.5)
239   stim[i].dur = 1
240   stim[i].del = i*20-39
241   stim[i].amp = $1
242   }
243   for i = 252, 500 preRGcells[1].soma {
244   stim[i] = new IClamp(0.5)
245   stim[i].dur = 1
246   stim[i].del = i*20-5019
247   stim[i].amp = $1
248   }
249   }
250
251   tstop = 5500
```

Literaturverzeichnis

[BK02] Horst Bayrhuber and Ulrich Kull. *Lindner Biologie, Lehrbuch für die Oberstufe, Teil 2*. Verlag Gustav Swoboda & Bruder, 2002.

[CH06] Ted Carnevale and Michael Hines. *The NEURON Book*. Cambridge University press, 2006.

[DGP98] Milan R. Dimitrijevic, Yuri Gerasimenko, and Michaela M. Pinter. Evidence for a spinal central pattern generator in humans. *Annals of the New York Academy of Science*, 860:360–376, 1998.

[Gue09] Pierre A. Guertin. The mammalian central pattern generator for locomotion. *Brain Research Reviews*, 62:45–56, 2009.

[HH52] Alan L. Hodgkin and Andrew F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117:500 – 544, 1952.

[HHK52] Alan L. Hodgkin, Andrew F. Huxley, and B. Katz. Measurement of current-voltage relations in the membrane of the giant axon of *Loligo*. *The Journal of Physiology*, 116:424 – 448, 1952.

[Hof09] Ursula Hofstötter. *Model of spinal cord reflex circuits in humans: Stimulation frequency-dependence of segmental activities and their interactions*. PhD thesis, TU Wien, 2009.

[Hol08] Mario Holzer. Extrazelluläre Stimulation von Nervenzellen: eine Anwendung mit NEURON. Master's thesis, TU Wien, 2008.

[JMR+04] B. Jilge, K. Minassian, F. Rattay, M. M. Pinter, F. Gerstenbrand, H. Binder, and M.R. Dimitrijevic. Initiating extension of the lower limbs in subjects with complete spinal cord injury by epidural lumbar cord stimulation. *Experimantal Brain Research*, 154:308–326, 2004.

[MJR+04] K. Minassian, B. Jilge, F. Rattay, M.M. Pinter, H. Binder, F. Gerstenbrand, and M.R. Dimitrijevic. Stepping-like movements in humans with complete spinal cord injury induced by epidural stimulation of the lumbar cord: electromyographic study of compound muscle action potentials. *Spinal Cord*, 42:401–416, 2004.

[MPR+07] K. Minassian, I. Persy, F. Rattay, M.M. Pinter, H. Kern, and M.R. Dimitrijevic. Human lumbar cord circuitries can be activated by extrinsic tonic input to generate locomotor-like activity. *Human Movement Science*, 26:275–295, 2007.

[MR07] David A. McCrea and Ilya A. Rybak. Modeling the mammalian locomotor cpg: insights from mistakes and perturbations. *Progress in Brain Research*, 165:235–253, 2007.

[MR08] David A. McCrea and Ilya A. Rybak. Organization of mammalian locomotor rhythm and pattern generation. *Brain Research Research*, 57:134–146, 2008.

[Rat99] Frank Rattay. The basic mechanism for the electrical stimulation of the nervous system. *Neuroscience*, 89(2):335–346, 1999.

[Rei08] Heiko Simon Reitner. Der Einsatz von Computersimulationssoftware für Nervenzellenstimulation zu Lehrzwecken. Master's thesis, TU Wien, 2008.

[RSLR06] Ilya A. Rybak, Natalia A. Shevtsova, and Myriam Lafreniere-Roula. Modelling spinal circuitry involved in locomotor pattern

generation: insights from deletions during fictive locomotion. *The Journal of Physiology*, 577.2:617–639, 2006.

[SS06] Robert F. Schmidt and Hans-Georg Schaible. *Neuro- und Sinnesphysiologie*. Springer Medizin Verlag, 2006.

Webverzeichnis

[WEB1] NEURON an der Yale University, Stand: März 2010
http://www.neuron.yale.edu/neuron/

[WEB2] Tutorial zu NEURON, Stand: März 2010
http://www.anc.ed.ac.uk/school/neuron/tutorial/tutD.html

[WEB3] Datenbank von Nervenmodellen, Stand: April 2010
http://senselab.med.yale.edu/neurondb/default.asp

[WEB4] Nervenmodell für Rückenmarkzellen, Stand: April 2010
http://senselab.med.yale.edu/modeldb/ShowModel.asp?model=3805&
file=\dodge73\motor.hoc