Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
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/eng

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

OpenWorm: Design and Evaluation of Neural Circuits on the Virtual Worm, Caenorhabditis elegans

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

David Lung
Matrikelnummer 01026620

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr.rer.nat. Radu Grosu
Zweitbetreuung: Dott.mag. Ramin M. Hasani

Wien, 1. Dezember 2018

David Lung

Radu Grosu

OpenWorm: Design and Evaluation of Neural Circuits on the Virtual Worm, Caenorhabditis elegans

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

David Lung
Registration Number 01026620

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.rer.nat. Radu Grosu
Second advisor: Dott.mag. Ramin M. Hasani

Vienna, 1st December, 2018

David Lung

Radu Grosu

Erklärung zur Verfassung der Arbeit

David Lung
Preysinggasse 15/15, 1150 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Dezember 2018

David Lung

Acknowledgements

I am thankful to my supervisor Prof. Radu Grosu for giving me the opportunity to write this thesis at the Cyber-Physical Systems Group. Special thanks go to my co-advisor Dott. Mag. Ramin Hasani, for supporting me with valuable input and feedback. I also want to thank my family and friends for supporting me throughout the past years.

Kurzfassung

Der Fadenwurm C. elegans ist, mit dem vollständig abgebildeten Konnekotm und das sequenzierte Genom, das am besten erforschte Tier weltweit. Dennoch, wie genau die Eigenschaften einzelner Neuronen und die Wechselwirkungen mit verbundenen Zellen unterschiedliche Verhaltensweisen hervorrufen, wie zum Beispiel die Fortbewegung, ist noch nicht bekannt. Aufgrund der hohen Nummer an noch nicht bekannten Parametern des Nervensystems, können verschiedene Hypothesen, über die notwendigen Bedingungen für das Einleiten von Verhalten, aufgestellt werden.

In dieser Arbeit stellen wir ein neues, biologisch grundiertes, Modell eines neuronalen Netzwerkes vor, verantwortlich für die Generierung koordinierter Muskelaktivität, die dazu führt, dass sich der Wurm vorwärts bewegt. Das Modell umfasst, neben den Muskeln, einen Teil des Nervensystems des Wurms, welcher bekannt dafür ist für das Indizieren der Fortbewegung verantwortlich zu sein. Neuronen und Muskeln sind als Single-Compartment Zellen modelliert. Ihr Membranpotential wir mithilfe eines Hodgkin-Huxley Modell berechnet, welches beschreibt, wie sich das Membranpotential ändert, wenn sich Ionen durch die Membran hindurch bewegen.

Wir implementieren unser neues Modell innerhalb der OpenWorm Plattform, um das Nervensystem und den Körper des Wurms zu Simulieren und zeigen, dass unser Modell in der Lage ist Vorwärtsbewegung einzuleiten.

Um Wissenschafter zu Helfen ihre eigenen in silico Experimente mit OpenWorm zu definieren, erstellen wir ein Docker image welches eine Toolchain von OpenWorm Programmen und Skripten beinhaltet. Die generierten Daten über das Verhalten eines simulierten Wurms können mit Daten echter Würmer verglichen werden, um die Ergebnisse der Experimente zu Validieren.

Abstract

With a completely mapped connectome and a sequenced genome, the nematode *C. elegans* is the world's best understood animal. Nevertheless, how the dynamics of single cells and interactions with coupled cells give rise to high-level behavior, like locomotion, is poorly understood. It is the high number of not yet known parameters of the nervous system leading to various hypotheses about how a certain behavior gets induced.

In this work, we introduce a new, biologically grounded, neural circuit model, responsible for generating coordinated muscle activity leading to forward-crawling locomotion of *C. elegans*. The model incorporates a subset of neurons of the nervous system of *C. elegans*, known to be involved in forward locomotion, and body-wall muscles. Neurons and muscles are modeled as single-compartmental cells, with the soma as the only compartment. Their membrane potential dynamics are governed by a modified version of the Hodgkin-Huxley model, describing how the membrane potential changes over time based on ionic currents flowing across the membrane.

We implement our new model within the OpenWorm platform, to simulate the nervous system and the body of the worm, and show that our model is able to create forward crawling movement of a simulated worm inside a bounded environment.

To help scientists to design their own in silico experiments using OpenWorm, we create a Docker image containing a pipeline of tools to run the simulations. The generated worm behavior data can be used to validate the underlying model of the experiment, by statistically comparing the data with data extracted from videos of real worms.

Contents

Kurzfassung **ix**

Abstract **xi**

Contents **xiii**

1 Introduction **1**

2 State of the Art **3**
- 2.1 The Model Organism C. elegans 3
  - 2.1.1 Anatomy of C. elegans 4
  - 2.1.2 Behavior of C. elegans... 4
- 2.2 Neuron and Neural Network Simulation Frameworks 5
- 2.3 Models of C. elegans Locomotion 5

3 Methodology **9**
- 3.1 Neuronal Modeling 9
  - 3.1.1 Biological Background of Neurons 9
  - 3.1.2 Computational Neuron Model 11
- 3.2 Synaptic Modeling 14
  - 3.2.1 Biological Background of Synapses 14
  - 3.2.2 Computational Synaptic Models 16
- 3.3 Neural Circuits 18
  - 3.3.1 New Forward Crawling Neural Circuit - Description 18
- 3.4 C. elegans Simulation Platform 20
  - 3.4.1 OpenWorm 20
  - 3.4.2 c302 - OpenWorm's Nervous System Simulation Framework 21
  - 3.4.3 Sibernetic - OpenWorm's Body Simulation Framework 29
- 3.5 Virtualization Platform 30

4 Experiments and Results **33**
- 4.1 Modifying OpenWorm's Neuron Model 33
- 4.2 Optimization of a C. elegans' brain cell network 34
  - 4.2.1 Step 1: Optimization of the whole network 36

4.2.2 Step 2: Partial Optimization: AVB-DB, DB-DB 38
4.2.3 Step 3: Partial Optimization: DB1-Muscles, DB2-Muscles . . . 42
4.2.4 Step 4: Hand-tuning DB-Muscle conductance 43
4.2.5 Step 5: Hand-tuning DB-Muscle conductance 45
4.3 New Forward Crawling Neural Circuit - Implementation 47

5 Conclusion 57
5.1 Summary/Discussion 57
5.2 Future work 58

A Supplementary material 61
A.1 Network Configuration File - c302_FW.py 61

List of Figures 67

List of Tables 69

Bibliography 71

CHAPTER 1

Introduction

The nervous system of the adult hermaphrodite nematode *Caenorhabditis elegans* (*C. elegans*) comprises only 302 neurons. Although the nervous system of this worm is very small, compared to other organisms like humans with approximately 86 billion neurons [ACG$^+$09], it shows a surprisingly high variety of behaviors. The reason for that is the complexity of neurons and neuronal circuits within the nervous system of *C. elegans.* A single neuron can be part of multiple neuronal circuits, responsible for inducing different behaviors. For example, *C. elegans'* nervous system comprises a type of motor neurons which play a role in forward and backward locomotion [ZS15]. A single sensory neuron can respond differently, depending on the input stimulus, e.g., gentle or harsh touch [SKB$^+$03, COL$^+$18].

To study the behavior of a complex system, like the nervous system of an organism, computational models are being used. A proper computational model of *C. elegans'* nervous system and its simulated output can help to analyze a high-level behavior, e.g., locomotion, based on low-level interactions of neurons, or other components within the nervous system, and how modifications to neuronal or synaptic parameters can change the behavioral output. Without such a model, one has to work with real worms what is not always helpful to answer a specific question. To modify properties of neurons, or to remove or silence synapses, genetic modifications must be done and can be a very challenging task, or not possible at all. With a computational model, on the other hand, one can simply modify one of its parameters, rerun the simulation, and analyze the changed output. Unfortunately, there are still too many unknown parameters within the nervous system of *C. elegans* and scientists have constructed multiple models of the same behavior. Those models are all equally valid until they get proven false based on genetic experiments and electrophysiological measurements using real worms.

Computational model at different levels of details have been created, for addressing different sets of questions, including questions about the behavior of single neurons [KD17] and muscles [BC08a], subcircuits, comprising only a subset of neurons within the nervous system, responsible for generating a specific behavior, e.g. locomotion [RAL$^+$16, WRR96, App14, RSSK13], or the whole nervous system [KMK17, YVT$^+$17].

In this thesis, we construct a new single-compartment conductance-based neuron network model, incorporating a few simplifying, although biologically plausible assumptions, responsible for generating rhythmic muscle activity of *C. elegans* in a way that induces forward crawling locomotion. We implement this model within the OpenWorm platform and verify whether the output of our model is indeed able to induce forward crawling, by using OpenWorm's body simulation framework. To validate the accuracy of the movement of the simulated worm, we can create a Turing-like test [TUR50]. Two videos, one shows the recorded movement of a real worm, the other one shows the output of our simulation, must be compared by a person (figure 1.1). If the person cannot distinguish between the real worm and the simulation, our model passes the Turing test and can be certified valid, until newly gained insights, from experiments with real worms, prove the model to be wrong.

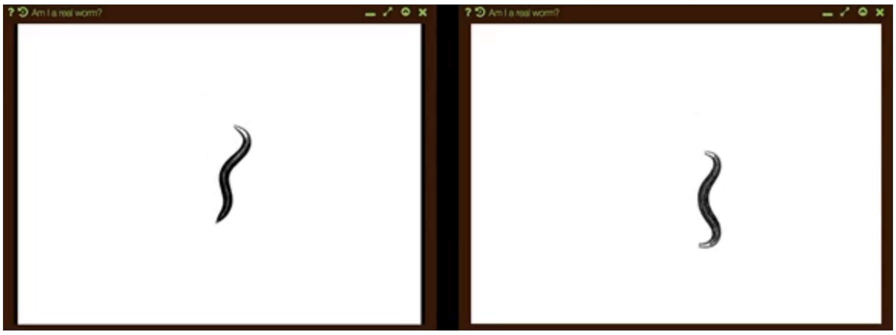


Figure 1.1: C. elegans Turing-like test - which worm is real? The figure is a snapshot of a video showing a real worm on one side, and a simulated worm using our forward crawling network model implemented within the OpenWorm platform, on the other side. The video was created by Matteo Cantarelli, a co-founder of the OpenWorm project.

The remainder of this work is structured as follows: In chapter 2, we give background information about the model organism *C. elegans*, describe state of the art tools for simulating dynamics of neurons and give an overview of existing models related to the locomotion of C. elegans. In chapter 3, we describe the computational models and tools we used to implement and to run ours in silico experiments. We also introduce a simplified neural circuit model responsible for inducing forward crawling of *C. elegans.* In chapter 4, we show and explain the results of our experiments. In the last chapter (5) we summarize and discuss our work, and describe possible future steps.

CHAPTER 2

State of the Art

In this chapter, we introduce the model organism *C. elegans*, state of the art simulation frameworks and existing models for generating locomotion in *C. elegans*. In section 2.1, we give biological background information about *C. elegans* and talk about the advantages of using this animal for all kinds of experiments. In section 2.2, we describe state of the art software for simulating neural networks. In section 2.3, we describe different models of *C. elegans* behavior with focus on locomotion.

2.1 The Model Organism C. elegans

Caenorhabditis elegans (*C. elegans*) is a tiny nematode with approximately 1 mm length living in soil all over the world [FF15]. It is transparent and non-parasitic what makes it easy to observe and manipulate the inner workings and safe to use in the laboratory. *C. elegans* can be cultured on petri dishes with agar and are usually fed with *E. coli* bacteria [AY12]. Due to the small size, it is possible to cultivate up to ten thousand individuals on a single petri dish what makes it also cheap for laboratories to work with this organism. *C. elegans* has a very short life cycle, depending on the temperature of the environment, of around three days from an embryo to a fertile adult [CWC15]. *C. elegans* was the first multicellular organism with a completely sequenced genome [eSC98] and the only one with a completely mapped connectome so far [WSTB86, VCP$^+$11]. There are two sexes of *C. elegans* nematodes; hermaphrodites and males. The self-fertilizing adult hermaphrodites have precisely 959 somatic cells [SSWT83]. The relatively simple nervous system of the adult hermaphrodite consists of only 302 neurons and around 7000 synapses. [WSTB86, VCP$^+$11] Male worms, with 1031 somatic cells, have some anatomic and behavioral differences, whereas the main difference is that they express mating behavior. [RBMP97]. In this thesis, we will only focus on the adult hermaphrodite *C. elegans* nematode.

Due to the simplicity of the nervous system and ..., many researchers use this worm for experiments on different topics like neuronal development, aging, learning, or memory.

C. elegans is also used for studies aiming in understanding neural circuits and behavior. How do neural circuits induce coordinated muscle activity and how can this activity be modulated from sensory input?

Although we know the genome and how many connections there are between a pair of neurons - we even know the exact location of the synapses - many important properties are still not known. For example, we do not know the polarity for most of the chemical synapses, whether they are excitatory or inhibitory, or the weight (the strength) of the synapses. Due to the unknowns, one can create different models for a single behavior of the worm, whereas, as long as we lack more detailed information about the parameters of the nervous system, each of them can be equally valid.

2.1.1 Anatomy of C. elegans

Like in other nematodes, *C. elegans'* body is unsegmented, cylindric and tapered at the ends. The body shape is maintained by internal hydrostatic pressure. The worm's body has two separated tubes. The outer tube consists of cuticle, hypodermis, muscles, and neurons. The pharynx, intestine, and gonad are located in the inner tube [RBMP97].

C. elegans has 95 body wall muscles divided into four longitudinal bundles, one quadrant with 23 muscle cells, the others with 24. Other muscles are located around the vulva and the pharynx [WSTB86, AH02]. In this thesis, only body wall muscles are of interest, because it is the activity of these muscles relevant for locomotory behavior.

2.1.2 Behavior of C. elegans...

Although *C. elegans'* nervous system is relatively simple, compared to other animals, it shows a surprisingly high variety of diverse behaviors. The worms can move to other locations, also known as locomotion. In soil, they crawl, while they swim in liquids with a low viscosity like water. They can move forward and backward, or change their direction. Experiments have shown that after tapping the petri dish, containing the worms, they start to move backward for a brief time before they move forward again. This behavior is known as tap withdrawal reflex [WR95]. With such an experiment, one can also show nonassociative learning or habituation; e.g., repeated taps result into responses with reduced amplitude and frequency [RBC90, WR97]. C. elegans can also sense chemicals; thus they may steer gradually towards a location of a higher concentration of chemicals, or away from it, which is known as chemotaxis [War73]. There is a similar behavioral output as a response for changes in temperature (thermotaxis [HR75]), oxygen concentration (aerotaxis, [GKL$^+$04, ZGP$^+$09]), or stiffness of the environment (durotaxis [PP16]). *C. elegans'* capacity to learn environmental properties is a form of associative learning.

2.2 Neuron and Neural Network Simulation Frameworks

For understanding the brain, the numerical simulation of single neurons and neural networks has become a critical part of research projects. Thus, many software tools have been developed to support research in the field of computational neuroscience.

NEURON [CH06] is a prevalent tool used in many neuroscience projects for simulating single neurons and networks of neurons. The focus of NEURON is on more detailed models, e.g., multi-compartmental neuron models; thus it is better suited for small networks of neurons. NEURON simulations can be run on one core machines or clusters, but code written for a single machine might require some modifications to enable cluster support.

NEST (NEural Simulation Tool) [GD07, PSV+17] is a simulator, optimized for large-scale neural networks using threads, or MPI-communication. It is one of the main simulators of the Human Brain Project and supports multiple neurons and synapse models limited to a small number of compartments.

Brian 2 [GB08, GB09, SGBB14] is a simulator, primarily, for networks of single-compartmental neuron models. Neuron models are defined by giving their differential equations as Python strings, unlike in other neural simulation tools where the user choose a predefined model. As another difference to other simulators, simulations using Brian can only run on a single machine.

SIM-CE is a MATLAB-Simulink implementation of single-compartmental, conductance-based neuron models and synapses within the nervous system of *C. elegans* [HBF+17]. It can be used to design and simulate small neural circuits composed of multiple neurons and synapses, but it has not been optimized for running large-scale networks, i.e., it can only utilize cores of a single machine, to run a simulation of the entire nervous system.

2.3 Models of C. elegans Locomotion

Locomotion, the task of an animal to move its body from one location to another one inside an environment, is an essential behavior of every animal. *C. elegans* move to find food or conspecifics, to flee from predators, or because they sense chemical concentrations indicating a toxic environment, which they need to avoid.

The form of motion (gait) of many animals depends on the environment. An animal can change its gait, to move more efficiently, if it gets into an environment with different properties. *C. elegans* crawl in an undulatory way, like snakes, in an environment with high viscosity (like agar gel). They crawl forward by propagating dorso-ventral bends from head to tail with an average frequency, wavelength, and amplitude of around 0.5 Hz, 0.8 mm, and 0.1 mm, respectively [SSK+12]. In an environment with low viscosity (like water), they swim and take "C"-shaped postures. The traveling waves during forward-swimming behavior propagate from head to tail with different properties. The average frequency, wavelength, and amplitude is around 2 Hz, 2.1 mm, and 0.26 mm, respectively

[SSK+12]. Other values for frequency, wavelength, and amplitude can be observed using physical surroundings with different viscosities [KCG+07, FYWX+10].

Forward crawling is often accompanied by foraging behavior and pharynx pumping (as well as egg laying and defecation). These behaviors are suppressed during swimming or backward locomotion [VGTY+11, VGDBPS12, MPS10].

One of the first attempts of modeling the worm's crawling behavior on agar was made by Niebur and Erdös [EN90, NE91, NE93]. In their model, a central pattern generator (CPG), located in the head, induced periodic dorsoventral bends of the head to initiate movement with a fixed frequency of undulation based on the output of the CPG. Due to local stretch receptors, muscles got excited based on the level of excitation (contraction) of anteriorly neighboring muscle cells. Therefore the rest of the body followed the bends of the head. They only modeled crawling locomotion because they thought that swimming is a separate gait and with their model, the worm would not be able to move in an environment with reduced viscosity.

In the work of Suzuki et al. [STO05] they used a simplified version of the nervous system that was able to respond to gentle touch. The body of the worm was modeled as a chain of rigid links. Neuronal oscillators controlled each joint angle. Locomotory behavior could be generated including forward and backward crawling, rest and turns (omega type turn and coil type turn). Although the model was able to reproduce locomotion, biological evidence was not taken into account.

Bryden and Cohen [Bry03, BC04, BC08b] introduced a model of the worm's nervous system without a CPG and a simplified model for the physical body of the worm. While their model could generate waves of muscle activity propagating from head to tail inducing forward locomotion, with the help of a proprioceptive mechanism, it failed to generate realistic speed and amplitude of undulation.

Rönkkö and Wong [RW08] modeled the body of worms and different environments (gel, liquid, and soil), including food and repellents, as a particle system. In their work, the worm's undulatory locomotion could be generated by only simulating physical forces of particles of the worm interacting with other particles of the environment. Although the result was impressive, their model missed a biologically grounded neural circuit.

Based on biological evidence for crawling and swimming being part of a single gait, Boyle created a model [Boy09] of a biologically realistic neural circuit within a more realistic body model than used in the work of Bryden and Cohen [BC08b]. Instead of a CPG, muscle activity got modulated, depending on the viscosity of the environment, due to a proprioceptive effect through stretch receptors. The model was able to generate crawling and swimming, as well as intermediate movements, based on the environment which modulated the speed and amplitude of undulation.

In this thesis, we introduce a new, biologically grounded although simplified, neural network model, responsible for generating staggered activation of body-wall muscles posterior to the neck. We assumed the existence of two CPGs, one CPG located in head inducing head movements, and another CPG inducing body bends posteriorly to the head, which gets propagated along the body from neck to tail due to a proprioceptive effect. We used OpenWorm's physical 3D model of the body of the worm which got fed with the output of our new neural network model to generate forward crawling movement based on physical forces of the worm interacting with its environment.

CHAPTER 3

Methodology

In this chapter, we will describe the components of neural circuits, a structure which is responsible for inducing behavior. It can also be seen as a subgraph of the nervous system of an animal. First, we will introduce a model of a single biological neuron, including the anatomy, different roles of neurons and the computational model we used to simulate the activity of neurons (section 3.1). Then we will describe how neurons are interconnected by different types of synapses enabling the communication of those neurons and the computational model simulating the synaptic current going into a target neuron (section 3.2). Building on that, we will talk about neural circuits composed of a population of neurons connected by synapses. We will also introduce a new model with a few simplifying assumptions, we designed, responsible for generating forward crawling movement of *C. elegans* (section 3.3). At the end of this chapter (section 3.4), we will describe some tools of the OpenWorm platform we used to simulate the neural circuit and the body of the worm. We will also describe how we created a toolchain based on these tools using a virtualization software, called 'Docker', to pack the tools and dependencies enabling users to run in silico experiments with minimal effort.

3.1 Neuronal Modeling

In this section, we give an overview of neurons, the primary components of the nervous system in almost all animals including *C. elegans.* We describe the anatomy of a biological neuron, different roles of neurons, and the computational neuron model we used for our experiments to simulate the membrane potential and the internal calcium concentration.

3.1.1 Biological Background of Neurons

Neurons, also known as nerve cells, are electrically excitable cells which can receive, integrate and process incoming signals, and transmit information to other cells via structures called synapses 3.2. Neurons, but also other cells like glia or muscle cells, are surrounded by a plasma membrane, impermeable to charged ions. Ions can only pass through ion channels. Ion channels are special membrane proteins forming a pore connecting the intracellular with the extracellular fluid allowing specific ions to flow into or out of the cell and are (unevenly) distributed across the cell membrane. Neurons are the primary components of the nervous system in animals. Nevertheless, a single neuron does not induce behavior, but they are interconnected to form neural circuits 3.3. The neural circuits receive specific information from the environment or internal state of the animal. After a neural circuit got activated, it responses with behavior, e.g., locomotion, or avoiding obstacles. While the nervous system consists not only of neural cells but also of glial cells, another cell type, we focus only on neurons, since the primary function of glial cells is to support neurons.

Anatomy of a Neuron

Neurons can be divided into three distinct parts: the soma, dendrites, and the axon.

- The **soma** (or cell body) contains the nucleus where the proteins, which are required from the neuron, are produced. Two types of processes, dendrites and the axon, extend from the soma, where dendrites are usually shorter than axons. In a specialized region of the soma, the axon hillock, action potentials are initiated and get propagated along the axon.
- In general, neurons receive signals at the **dendrites** which get propagated to the soma. Neurons can have multiple dendrites, each of them with up to several thousands of branches.
- Action potentials get propagated along the **axon**. At its end, the axon branches multiple times, to make synapses with other cells. Usually, the end of the axonal branch of the presynaptic neuron synapses onto a dendrite of the postsynaptic cell, but other locations are also possible.

Classes of Neurons

One way how to classify neurons is based on the function they fulfill. There are 3 groups of neurons: sensory neurons, interneurons, and motor neurons.

- **Sensory neurons** receive signals from sensory receptors, which are connected with the sensory neurons, and transmit information to other neurons in the nervous system. These receptors are located almost everywhere, e.g., in the mouth, the nose, in ears, in eyes, the skin, or internally.
- **Interneurons** receive information from sensory neurons or other interneurons and transmit information to other interneurons or motor neurons. In the *C. elegans* literature, interneurons connected to motor neurons are often called **command neuron**.

- **Motor neurons** receive information from other neurons and send signals to the connected muscles. When a muscle cell receives an excitatory/inhibitory signal from a neuron, it can lead to contraction/relaxation of the muscle.

3.1.2 Computational Neuron Model

Integrate-and-Fire Model

More then one century ago, in 1907, Lapicque modeled a neuron using an electric circuit of a capacitor and a resistor connected in parallel, representing the capacitance and the resistance of the membrane, respectively. Besides, Lapicque's model generates action potentials (a delta function spike) after the membrane capacitor got charged to a constant threshold value. Subsequently, the capacitor discharges and the membrane potential gets reset to a constant resting value [Abb99].

The change of the membrane potential in time can be described as a ordinary differential equation:

$$C_m \frac{dV_m}{dt} = I_{ext} \tag{3.1}$$

where V_m is the membrane potential, C_m is the membrane capacitance, and I_{ext} is the synaptic or externally injected current.

Leaky-Integrate-and-Fire Model

The leaky-integrate-and-fire (LIF) model extends the integrate-and-fire (3.1) model, by adding a term reflecting the diffusion of ions (ions passing through a leak channel) occurring when the cell is not in a resting state.

The LIF model can be described as a ordinary differential equation:

$$C_m \frac{dV_m}{dt} = -G_{Leak}(V_m - E_{Leak}) + I_{ext} \tag{3.2}$$

where V_m is the membrane potential, C_m is the membrane capacitance, and I_{ext} is the synaptic or externally injected current. G_{Leak} and E_{Leak} are the conductance and reversal potential of the leak channel, respectively.

Hodgkin-Huxley Model

We simulated the activity (membrane potential and internal calcium concentration) of a neuron and muscle cell with a conductance-based model, also known as the Hodgkin-Huxley (HH) model. In fact, it is a modified version of the HH model because, unlike the original version, *C. elegans* neurons and muscles do not undergo Na^+ (sodium) induced action potentials, because they lack voltage-gated Na^+-channels. Instead, *C. elegans* neurons show graded potentials where Ca^{2+} (calcium) acts as the major current for the depolarization of the cell [GHAL98, Bar98]. In *C. elegans*, only body-wall muscle cells have been shown to generate action potentials [LGC+11, GZ11]. The cells were modeled as single-compartmental neurons with the soma as the only compartment, since *C. elegans* neurons are very small, and many neurons have been shown to be isopotential [GHAL98].

The set or ordinary differential equations describing the conductance-based neuron model is the following:

$$C_m \frac{dV_m}{dt} = I_{K_f} + I_{K_s} + I_{Ca} + I_{Leak} + I_{ext} \tag{3.3}$$

where V_m is the membrane potential, C_m is the total membrane capacitance, and I_{ext} represents the synaptic or externally injected current. I_{K_f}, I_{K_s}, I_{Ca}, and I_{Leak} represent the currents passing through a fast and slow potassium, calcium and leakage channel, respectively.

$$C_m = C_{spec} * surfaceArea \tag{3.4}$$

$$surfaceArea = \begin{cases} 4 * (diam/2)^2 * \pi, & \text{if } length = 0. \\ 2 * (diam/2) * \pi * length, & \text{if } length > 0. \end{cases} \tag{3.5}$$

where C_{spec} is the specific capacitance per unit area, $surfaceArea$ is the surface area of the cell (the single compartment of neurons are modeled as spheres, as cylinders in the case of muscle cells). $diam$ is the diameter of the cell, $length$ is the length of muscles (not used for neurons).

$$I_{K_f} = p^4 q G_{K_f}(E_{K_f} - V_m) \tag{3.6}$$

$$I_{K_s} = n G_{K_s}(E_{K_s} - V_m) \tag{3.7}$$

$$I_{Ca^{2+}} = e^2 f(1 + (h-1)\alpha) G_{Ca^{2+}}(E_{Ca^{2+}} - V_m) \tag{3.8}$$

$$I_{Leak} = G_{Leak}(E_{Leak} - V_m) \tag{3.9}$$

G_{K_f}, G_{K_s}, $G_{Ca^{2+}}$, and G_{Leak} are the maximum conductance of the fast potassium, slow potassium, calcium, and leakage channel, respectively. E_{K_f}, E_{K_s}, $E_{Ca^{2+}}$, and E_{Leak} are the reversal potentials of the fast potassium, slow potassium, calcium, and the leakage channel, respectively. Potassium channels are voltage-gated, opening and closing depending on the membrane potential, while the leakage channel has a constant conductance. In addition to the voltage-dependent gates, the calcium channel is modeled with a calcium-dependent inactivation gate.

p, q, n, e, and f, are gating variables describing the probability of ion channel gates being in an open state and are as follows:

$$\frac{dx}{dt} = \frac{x_\infty - x}{\tau_x} \tag{3.10}$$

$$x_\infty = \frac{1}{1 + e^{-\frac{V_m - V_{mid_x}}{\kappa_x}}} \tag{3.11}$$

where x stands for p, q, n, e, and f.
p, n, and e are activation gates, q, and f are inactivation gates.

As described above, the calcium channel is not only voltage-gated but also calcium-gated. Thus, h (the calcium mediated inactivation gate) is modeled with a calcium concentration dependency as the following:

$$h = \frac{1}{1 + e^{\frac{[Ca^{2+}]_{half} - [Ca^{2+}]}{\kappa}}} \tag{3.12}$$

Izhikevich Model

Izhikevich's model [Izh03] is a simple spiking neuron model. It is a simplification of the HH model, able to reproduce spiking and bursting behavior of cortical neurons. A result of the simplification of the model is computational efficiency, comparable to the integrate-and-fire model. Unlike the HH model, the Izhikevich model is not biophysical meaningful. For that reason, we used the HH model to simulate the cells.

The Izhikevich model is a set of two ordinary differential equations:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I_{ext} \tag{3.13}$$

$$\frac{du}{dt} = a(bv - u) \tag{3.14}$$

with a after-spike resetting:

$$if \quad v \geq 30mV, then \begin{cases} v = c \\ u = u + d \end{cases} \tag{3.15}$$

where v represents the membrane potential, u a recovery variable related to the activation of K^+ currents and the deactivation of Na^+ currents. I_{ext} is the synaptic or externally injected current. By tuning the parameters a, b, c, and d, 20 firing patterns of a cortical neuron can be reproduced [Izh04].

Intracellular Calcium Concentration Model

In *C. elegans*, calcium is the major ionic current responsible for the depolarization of a cell. Calcium is also needed for synaptic transmission through chemical synapses, because it triggers the process inside the presynaptic end of the synapse leading to the release of neurotransmitter into the synaptic cleft. Especially for muscle cells, the internal calcium concentration plays a vital role because it is responsible for muscle contraction. When the internal calcium concentration reaches a threshold, a process inside the muscle gets activated leading to contraction. The amount of calcium ions inside a muscle determines how often this process gets activated, and thus also the strength of contraction. That is why calcium transients and internal calcium concentration is considered as an indicator for the level of activation of cells in *C. elegans*.

The intracellular calcium concentration pool was modeled with an ordinary differential equation as the following:

$$\frac{d[Ca^{2+}]}{dt} = \frac{I_{Ca^{2+}}}{surfaceArea}\rho - \frac{[Ca^{2+}] - [Ca^{2+}]_{rest}}{decayConstant} \tag{3.16}$$

where $I_{Ca^{2+}}$ is the current of calcium ions entering the cell, $surfaceArea$ is the surface area of the cell, ρ is a factor scaling the incoming calcium current, $[Ca^{2+}]$ is the internal calcium concentration of the cell, and $[Ca^{2+}]_{rest}$ is the internal calcium concentration the cell will reach at rest with a time course of $decayConstant$.

To simulate the membrane potential and internal calcium concentration of **muscle cells**, we used the same equations from above (equation 3.3 and 3.16) with a different set of parameters.

3.2 Synaptic Modeling

In this section, we will describe synapses, structures which enables cells to communicate with each other. We will describe the structure of a synapse, two different types of synapses, chemical and electrical, and how they work in general. We will also describe the computational model we used to simulate the communication of cells.

3.2.1 Biological Background of Synapses

Neurons can communicate with other cells. A signal of a neuron can change the global state of the neural network leading to a behavioral change of an organism. In the case of a neuron-to-neuron communication, the connection between the presynaptic neuron (source neuron) and the postsynaptic neuron (target neuron) is called synapse, while a neuron-to-muscle communication takes place at the so-called neuromuscular junction (NMJ). Most of the synapses (and NMJs) are chemical synapses where information is transmitted by chemical messengers called neurotransmitters. Electrical synapses (or gap junctions) on the other hand do not need these chemicals. The presynaptic and the postsynaptic cells are directly connected, to make it possible for molecules and ions to flow directly from one cell to another one. If not explicitly stated, we will use the term synapse for both, neuron-to-neuron and neuron-to-muscle connections, from now on.

Chemical Synapses

Chemical synapses transmit information from the presynaptic cell to the postsynaptic cell by releasing chemicals called neurotransmitters. The synapses are usually between the presynaptic axonal terminal and the postsynaptic dendrite or the soma. Inside the axonal terminal, there are many structures called synaptic vesicles containing the neurotransmitter which get released with a certain probability, when the membrane potential of the presynaptic cell reaches some threshold. Between the presynaptic cell and the postsynaptic cell, there is a small gap called synaptic cleft. At the postsynaptic side, receptor proteins are located where the released neurotransmitters bind to with a certain probability, leading to excitation or inhibition of the postsynaptic cell, depending on the neurotransmitter and the receptor type. Due to the neurotransmitter-receptor mechanism, also known as the key-lock model, chemical synapses are unidirectional. This means that a chemical synapse only transmits information in one direction.

A chemical synapse also has a specific strength referring to the probability of neurotransmitter getting released into the synaptic cleft and the number of the transmitter. The synaptic strength can change over time what is called synaptic plasticity. Synaptic plasticity is widely assumed to be an essential mechanism of the learning and memory process. For example, after touching the nose of a forward moving *C. elegans*, the worm starts to reverse [KH93]. If this touch gets repeated several times with a pause of a few seconds between the stimuli, the probability, and duration of the reverse movement decrease [CC14]. This behavior is called habituation and can be seen as a form of learning. This could be due to synaptic depression (decreased synaptic strength) of the synapse connecting the sensory neuron, which gets activated when the nose of the worm gets touched, and an interneuron. During this thesis, we did not model synaptic plasticity.

Synaptic transmission steps:

1. An action potential gets propagated along the presynaptic axon to the axon terminal.
2. Depolarization of the axon terminal leads to opening of voltage-gated calcium channels and thus to an influx of calcium (Ca^{2+}) ions.
3. Due to the high Ca^{2+} concentration, vesicles inside the axon terminal fuse with the membrane of the axon and release their available neurotransmitter into the synaptic cleft.
4. Neurotransmitter bind to receptors on the membrane of the postsynaptic cell leading, in general, to the opening of receptor channels followed by ions flowing into or out of the cell, resulting into excitation or inhibition of the postsynaptic cell.

Electrical Synapses

Besides chemical synapses, *C. elegans* uses electrical synapses, or gap junctions for direct cell-to-cell communication by forming a channel connecting the intracellular fluids of the cells. Such a channel is formed by innexin proteins. The genome of *C. elegans* encodes 25 different types of innexin genes [eSC98, SMN14, Hal17]. Based on the composition of innexins, different types of gap junctions, with up to 4 different innexin proteins for a single channel, are possible [Hal17]. In contrast to chemical synapses, electrical synapses transmit information much more rapidly. They are mostly bidirectional, allowing ions to flow in both directions. Depending on the structure of the gap junction (the combinations of the innexin proteins forming the gap junction channel), unidirectional connections are also possible and have been shown to exist in *C. elegans* [LCMW17]. Electrical synapses are less versatile than chemical synapses because the various combinations of neurotransmitter and receptors can induce all kinds of responses. Exciting/Inhibiting a presynaptic cell of an electrical synapse always excites/inhibits the postsynaptic cell, respectively. The reason for this behavior is the electrical gradient. Due to this gradient, ions - which are electrical charges - move from one cell to the connected cell through the gap junction, leading to a more balanced state of the membrane potentials of both cells. Therefore, bidirectional gap junction is often interpreted as a synchronization mechanism, synchronizing the activity of two cells. We modeled gap junctions as bidirectional connections, in our experiments.

3.2.2 Computational Synaptic Models

Chemical Synapse Models

The synaptic current of the chemical synapse is calculated as:

$$I_{chem1} = g_s s(E_s - V_{post}) \tag{3.17}$$

where g_s is the maximum conductivity of the synapse, V_{post} is the membrane potential of the postsynaptic cell, E_s is the reversal potential of the synapse, determining the polarity of the synapse, it can be either excitatory or inhibitory, and s is a synaptic activity variable, corresponding to the probability of a receptor channel of the postsynaptic cell being in an open state, thus s is also related to the release of neurotransmitter.

s is governed by:

$$\frac{ds}{dt} = \frac{s_\infty - s}{\tau_s} \tag{3.18}$$

with:

$$s_\infty = \frac{1}{1 + e^{\frac{V_{th} - V_{pre}}{\delta}}} \tag{3.19}$$

and

$$\tau_s = \frac{1 - s_\infty}{\kappa} \tag{3.20}$$

where V_{th} is the half-activation voltage of the synapse, V_{pre} is the membrane potential of the presynaptic cell, δ is the slope of the activation function, and κ is the dissociation rate (when the transmitter dissociates from the receptor, the channel switches to a closed state again)

We used a different model for the chemical synapses within the B-type motorneuron group, i.e. the synapses between DB1 and DB2 or VB1 and VB2 are modeled as:

$$I_{chem2} = g_s s(E_s - V_{post}) \tag{3.21}$$

where g_s is the maximum conductivity of a specific synapse, V_{post} is the membrane potential of the postsynaptic cell, E_s is the reversal potential of the synapse, and s is a synaptic activity variable governed by:

$$\frac{ds}{dt} = a_r \phi (1 - s) - a_d s \tag{3.22}$$

where a_r, and a_d is the synaptic activity's rise time and decay time, respectively, and ϕ is a sigmoidal activation function:

$$\phi = \frac{1}{1 + e^{\frac{V_{th} - V_{pre}}{\kappa}}} \tag{3.23}$$

where V_{pre} is the membrane potential of the presynaptic cell, V_{th} is the half-activation voltage, and κ is the slope of the sigmoid function.

Electrical Synapse (Gap Junctions) Model

Electrical synapses or gap junctions are bidirectional connections between two cells modeled as ohmic resistances:

$$I_{elec} = g_s (V_{pre} - V_m) \tag{3.24}$$

where g_s is the conductance of the synapse, V_{pre}, and V_m are the membrane potentials of the presynaptic and the postsynaptic cells, respectively.

For some of our experiments, where we wanted to create a propagating wave in a sequence of muscles (section 4.2), we used an uncommon type of gap junction with a sigmoid function modulating the conductance.

$$I_{delayedGJ} = g_s s (V_{pre} - V_m) \tag{3.25}$$

$$s = \frac{1}{1 + e^{\sigma(\mu - V_{pre})}} \tag{3.26}$$

where g_s is the conductance of the synapse, V_{pre} and V_m are the membrane potentials of the presynaptic and the postsynaptic cells, and s is a sigmoid function modulating the conductance with σ as the slope, and μ as the half-activation voltage.

3.3 Neural Circuits

In the sections above, we described what a neuron is and how two neurons are interconnected by synapses enabling a neuron to send information to the connected neuron. In the nervous system of *C. elegans* (but also in general), neurons never function in isolation; instead, populations of neurons interconnected by synapses form neural circuits responsible for inducing all kinds of behavior. For example, a neural circuit composed of sensory neurons, interneurons, and motor neurons can be responsible for sensing touch leading to acceleration or backward movement. Different neural circuits can be overlapping; thus a single neuron can be a member of multiple neural circuits and the activity of a single neuron can induce or modulate multiple behaviors.

In *C. elegans*, based on the wiring diagram, electrophysiological experiments, genetic manipulations, and cell ablation techniques, several circuits have been proposed to be responsible for generating a specific type of behavior upon activation. However, the circuits which have been described are just models and one could come to a more accurate version of a circuit after some experiments. As an example, it could be possible that the result of well-designed neuron ablation experiments shows that a specific neuron which was a member of the circuit, responsible for steering, is not relevant at all. Locomotion is another example of a model which was changed over time. Due to the many unknown parameters of the nervous system in *C. elegans*, multiple different models of locomotion were developed. The main distinction between these models is the existence and location of a CPG. While one model uses multiple phase-shifted CPGs generating phase-shifted activity of muscle cells, distributed along the body of the worm, another model does not use a CPG at all.

3.3.1 New Forward Crawling Neural Circuit - Description

We constructed a new model for the forward crawling locomotion comprising the following cells: left and right pair interneurons AVB (AVBL, AVBR), 18 excitatory B-type motor neurons including 7 dorsal (DB1-DB7) and 11 ventral (VB1-VB11), 19 inhibitory D-type motor neurons consist of 6 dorsal (DD1-DD6) and 13 ventral (VD1-VD13), as well as 95 body-wall muscle cells. We used a single-compartmental conductance-based HH like model to simulate the activity of the cells (3.1.2), and the wiring data of the adult hermaphrodite *C. elegans* generated by the WormWiring[1] project. The circuit is illustrated in figure 3.1.

To generate the alternating dorso-ventral activity of muscle cells, we used the following simplifying assumptions, while constructing the neural network:

---

[1] `wormwiring.org`

- **Synapses share the same weight** - The weights of the synaptic connections in the *C. elegans* connectome has not yet been defined [YVT+17]. Thus, a simplified version of the network will be used, by assuming that all the synaptic connections share the same weight in the network, even if multiple connections are present in the connectome.

- AVB neurons synapse onto DB and VB motor neuron groups mainly by gap-junctions. We assume that cholinergic B-type motor neurons excite their downstream cells with excitatory synapses while GABAergic D-type motor neurons inhibit their downstream cells with inhibitory synapses. DB motor neurons excite dorsal muscle cells and VD motor neurons inhibiting the opposing ventral muscles while inhibiting DD motor neurons to stop the inhibition of dorsal muscles. VB motor neurons excite ventral muscle cells and DD motor neurons inhibiting the opposing dorsal muscles while inhibiting VD motor neurons to stop the inhibition of ventral muscles.

- **AVB neurons are active during forward-crawling locomotion** - During forward locomotion, AVB neurons are active, observable with calcium imaging techniques [KKS+15]. They can induce or accelerate the forward movement. Synaptic inputs from upstream neurons (sensory neurons or other interneurons) were approximated by an input current into AVB neurons during the simulation to keep these neurons active.

- **Muscles located in the head are directly stimulated by periodic input pulses** - Upstream neurons of head muscle cells (the first seven muscle cells in each quadrant) are not included in this model. To generate alternating dorso-ventral bends of the head muscles, we directly stimulated these muscles with injected input pulses. The input pulses were adjusted to generate head bends of the same frequency as for the rest of the body.

- **First B-type motor neurons are directly stimulated by periodic input pulses** - A central pattern generator (CPG) mechanism was assumed to induce the activity of B-type motor neurons leading to dorso-ventral body bends posteriorly to the neck. We approximated the input from this hypothetical CPG mechanism by injecting input pulses to the first B-type motor neurons, DB1 and VB1.

- **A proprioceptive mechanism within B-type motor neurons** - Experiments have shown that the worm recognizes body bends, due to stretch receptors, which results into additional excitatory current going into B-type motor neurons posterior the bent region of the worm during forward-locomotion [WPH+12]. The result of such a chain of proprioceptive-coupled motor neurons is the propagation of the bends along the body. We approximated the proprioceptive effect by adding additional excitatory synapses between neighboring B-type motor neurons.

The result of the simulation of this model is described in section 4.3.

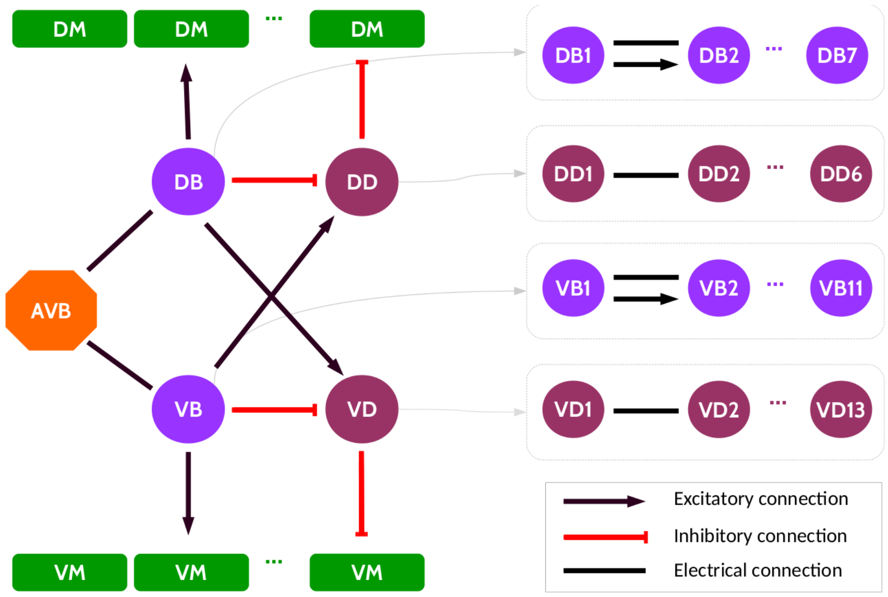


Figure 3.1: Symbolic representation of the neural circuit composed of AVB interneuron, B-type and D-type motor neurons, and muscle cells for generation of the forward crawling activity. Gap junction connections represented by black lines, excitatory chemical synapses by black arrows, inhibitory synapses by red connections.

3.4 C. elegans Simulation Platform

In this section, we describe the toolset we used to run ours in silico experiments. We give an overview of some of the core tools of the OpenWorm platform for simulating the nervous system and the body of *C. elegans*. At the end of this section, we explain the virtualization platform Docker we used to create a simulation pipeline. One of the advantages of using Docker is to enable users to run simulations more easily avoiding dependency hell.

3.4.1 OpenWorm

OpenWorm, launched in 2011, is an open-science project with the goal of building a detailed biophysical computational model of *C. elegans* and a powerful tool to understand the behavior of the worm, based on its biology. [SGV+14, GCC+15, SLP+18].

The approach of OpenWorm is to create different models for individual neurons, operating at multiple scales, based on insights collected from the research community, integrated into an unified computational platform.

One goal of neuroscientists is to determine how behavior of an organism arises from interactions with its environment. Without physical or chemical input from the external environment to the sensory system of the organism, an important link is missing [THC11].

That is why the OpenWorm community has agreed to incorporate biomechanics and interactions of the worm with its environment as critical components for understanding this organism. Therefore, a biomechanical model of the *C. elegans* body, Sibernetic (section 3.4.3), to simulate the interactions with a fluid or gel environment, and a modeling infrastructure for complex neuronal networks, c302 (section 3.4.2), has been developed.

The *C. elegans* research community has created multiple datasets about the nervous system, ranging from anatomical and structural data of the location of somatic cells and their interconnections [WSTB86, VCP$^+$11], to data sets of cholinergic and GABAergic neurons [PKSS$^+$15, GAH16], to the extrasynaptic connectome of neuropeptides [BBB$^+$16]. These heterogeneous datasets and other related data have been collected and merged into a graph database and a Python application programming interface has been developed by OpenWorm, to enable users to query this database to easily find information about the neuronal structure of *C. elegans* [SLP$^+$18].

OpenWorm is organized into a number of subprojects, all of them can be found on GitHub[2]. The simulation stack, i.e. the set of integrated tools to run an in silico experiment, of OpenWorm uses ion channel models and connectomes at the lowest level. At the next level there are models of neuromuscular coupling. At the top level, output from lower levels are fed into a 3D simulation of the physical body of the worm and the environment. The output of the body simulator can then be fed into a movement validation software system which compares the simulated worm with data of real worms [SLP$^+$18]. The simulation stack, including planned features, is illustrated in figure 3.2.

3.4.2 c302 - OpenWorm's Nervous System Simulation Framework

A crucial part of the OpenWorm project will be an accurate model of the worm's nervous system including all known cell types and connections.

It is still not clear, what level of biophysical detail is needed, to reproduce observed high-level behavior like locomotion. Therefore OpenWorm has developed a Python framework, c302[3], for generating models of the nervous system with different levels of anatomical and biophysical detail [GLG$^+$18] (figure 3.3).

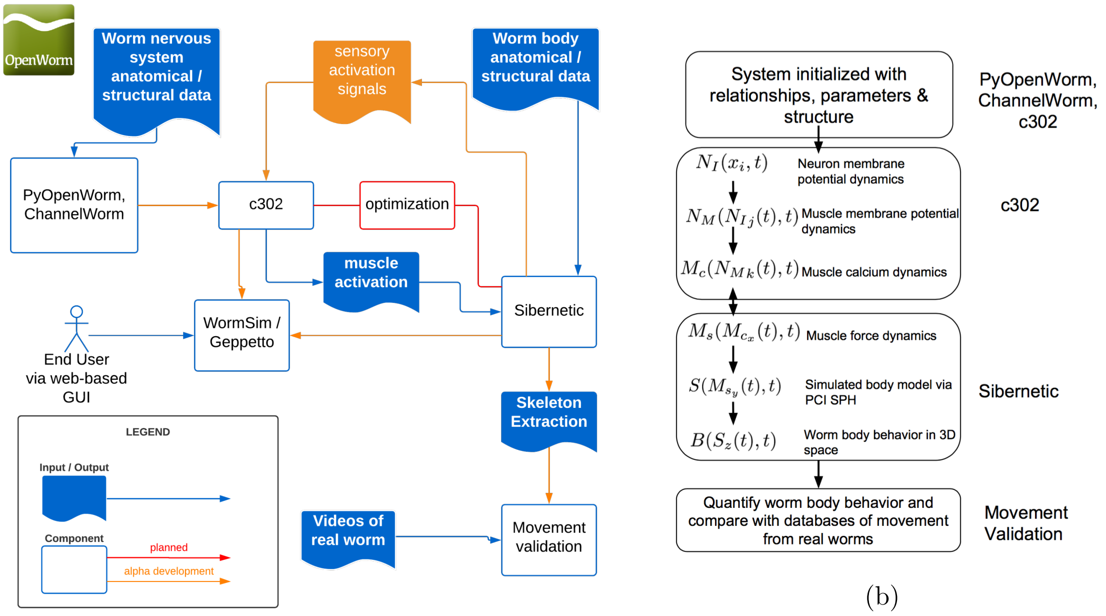
The varying levels of detail range from simple single-compartmental leaky integrate-and-fire, to multi-compartmental conductance-based cell models (figure 3.3, y-axis). Based on the network configuration file (usually created by the user), model instances generated with c302 can include single individual neurons or muscles, two cells connected by a single synapse, neural circuits, which are known or believed to generate a specific behavior, or the whole nervous system including all cells and connections of the connectome (figure 3.3, x-axis).

The creation of models with different level of complexity helps the user to address a specific set of questions. For example, a model which should be able to predict neuronal

---

[2] `https://github.com/openworm`

[3] `https://github.com/openworm/c302`


OpenWorm
Worn nervous system anatomical / structural data
sensory activation signals
Worm body anatomical / structural data
PyOpenWorm, ChannelWorm
c302
optimization
muscle activation
Sibernetic
WormSim / Geppetto
End User via web-based GUI
Skeleton Extraction
LEGEND
Input / Output
Component
planned
alpha development
Videos of real worm
Movement validation
System initialized with relationships, parameters & structure
PyOpenWorm, ChannelWorm, c302
$N_I(x_i,t)$ Neuron membrane potential dynamics
$N_M(N_{Ij}(t),t)$ Muscle membrane potential dynamics
$M_c(N_{Mk}(t),t)$ Muscle calcium dynamics
c302
$M_s(M_{c_x}(t),t)$ Muscle force dynamics
$S(M_{s_y}(t),t)$ Simulated body model via PCI SPH
$B(S_z(t),t)$ Worm body behavior in 3D space
Sibernetic
Quantify worm body behavior and compare with databases of movement from real worms
Movement Validation

(a)

(b)

Figure 3.2: Overview of the simulation stack of OpenWorm. (a) A component diagram of the current state of OpenWorm subprojects with connections between the different subprojects representing input and output. (b) A simplified overview of the dynamics simulated within the OpenWorm platform. Figure was taken from [SLP+18]

membrane potential dynamics could incorporate ion channels, whereas for predicting spikes over time, a simpler model can be sufficient and should be preferred, due to the reduced computational resources which are needed to run the simulation. For simulating a high-level behavior like locomotion, the model must incorporate a neural circuit, composed of neurons which are believed to be responsible for inducing locomotion, as well as muscle cells.

Model instances generated with c302, are saved as *NeuroML* files, a XML-based model description language for specifying the components, i.e. models of cells, ion channels, synapses, input stimuli, and 3D populations of synaptically connected cells, of a neural network as a hierarchical structure [GCC+10, CGC+14, GLG+18]. Models expressed in NeuroML format can be converted to formats of other preexisting tools and libraries, e.g. to the native format of NEURON, so the models can be visualized, simulated, and analyzed with these tools. After an instance of the network model got generated, it can be simulated using *pyNeuroML*. The output of the simulation are raw data of the membrane potentials and internal calcium concentrations over time of cells, and graphs visualizing these data (figure 3.4).

After installing c302 (with its dependencies), the following shell commands generate the NeuroML files of the model instance and run the simulation:

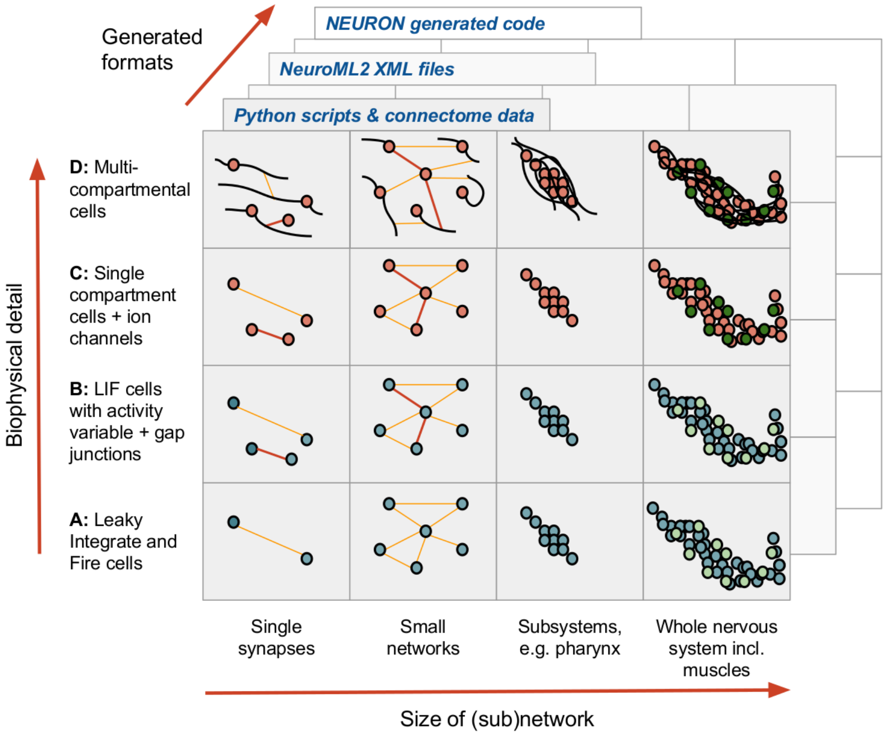

Generated formats
NEURON generated code
NeuroML2 XML files
Python scripts & connectome data
Biophysical detail
D: Multi-compartmental cells
C: Single compartment cells + ion channels
B: LIF cells with activity variable + gap junctions
A: Leaky Integrate and Fire cells
Single synapses
Small networks
Subsystems, e.g. pharynx
Whole nervous system incl. muscles
Size of (sub)network

Figure 3.3: Graphical representation of the anatomical and biophysical level of details which can be chosen for simulations of neural networks using c302. The increasing complexity of networks is shown on the x-axis, ranging from networks with a single synapse, to the whole nervous system including muscles. On the y-axis, different levels of biophysical neuronal models are shown, ranging from simple LIF, to multicompartmental conductance-based cell models. Chemical synapses are represented by orange, electrical synapses by red lines. LIF neurons/muscles are represented by blue/light green, conductance-based neurons/muscles by dark orange/dark green circles. Black lines represent dendrites and axons of multicompartmental cell models. Figure was taken from [GLG+18]

```
 # X is a placeholder for an arbitrarily chosen name of the network
configuration, e.g. ’FW’
 # Y is a placeholder representing the level of biophysical detail and
refers to the name of the parameters file, e.g. ’C2’ referring to
parameters_C2.py

 $ cd C302_PATH # Change current working directory to the location of c302
 $ python c302/c302_X.py Y # generates the model instance

 # To run the simulation with pyNeuroML
 $ pynml examples/LEMS_c302_Y_X.xml

 # To run the simulation with NEURON
```

```
$ cd examples
$ pynml LEMS_c302_Y_X.xml -neuron # Generate the Neuron files (Python/hoc
/mod)
$ nrnivmodl # Compile the mod files (used for cell/ion channel
definitions)
$ nrngui LEMS_c302_Y_X_nrn.py # Run the main Python file for the
simulation using NEURON
```

The commands above will run the simulation and show the results using the GUI of pyNeuroML or NEURON.

c302 provides a script, `runAndPlot.py`, for generating figures visualizing the results of the simulation, instead of displaying the results inside an GUI, what should take less time. To use that script for starting the simulation of a new network model instance, it has to be extended by a function call handling a new argument, which can be chosen arbitrarily, e.g. :

```
...

if __name__ == "__main__":
  if ARGUMENT in sys.argv:
    run_c302(...)

  elif OTHER_ARGUMENT in sys.argv:
    run_c302(...)

  ...

  elif "-fw" in sys.argv:
    run_c302("FW", "C2", "", 5000, 0.05, "jNeuroML", data_reader="
  UpdatedSpreadsheetDataReader", save=True, show_plot_already=True)

  elif "-fw_neuron" in sys.argv:
    run_c302("FW", "C2", "", 5000, 0.05, "jNeuroML_NEURON", data_reader="
  UpdatedSpreadsheetDataReader", save=True, show_plot_already=True)
```

The simulation, using `runAndPlot.py` with the new argument, can then be started with:

```
$ cd C302_PATH # Change current working directory to the location of c302
# Run the simulation using pyNeuroML, as defined inside runAndPlot.py
$ python c302/runAndPlot -fw

# Run the simulation using NEURON, as defined inside runAndPlot.py
$ python c302/runAndPlot -fw_neuron
```

A more detailed description of the installation and usage of c302 can be found online in the 'README' of the GitHub repository of c302[4].

---

[4] `https://github.com/openworm/c302`

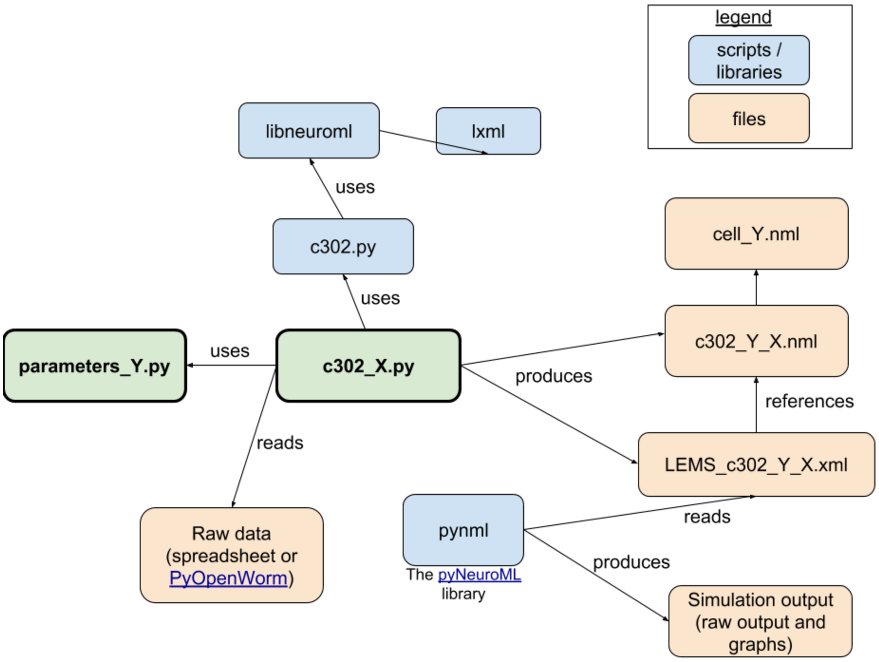


Figure 3.4: Overview of scripts and libraries used in the c302 framework and files created during the generation and simulation of a neural network model. Green boxes are files created or modified by the user, blue boxes are scripts/libraries used within c302, light brown/orange boxes are the files used as input, or created during the simulation of the network. X is an identifier of the neural network model and can be chosen arbitrarily, and Y refers to the level of biophysical detail of cells of the network, e.g. A for LIF model, or D for multicompartmental conductance-based models. Figure was taken from `https://github.com/openworm/c302/`

Modeling with c302

To create a new network configuration (c302_X.py) for our forward crawling neural circuit (c302_FW.py) model, we copied a preexisting one and modified its setup function accordingly. While it was possible to modify some of the parameters of the network, e.g. which neurons to include, if muscles should be included as well, or the number of connections of a synapse, we modified c302 in order to make the generation of networks more customizable.

We applied the following changes to c302, in order to make the generation of neural networks more configurable, using different parameters within the network configuration file. Below every bullet point, we provide a code snippet about the usage of the parameter inside the setup function of the network configuration file:

- *data_reader* - In our experiments we used the wiring data (connectome) of the adult hermaphrodite *C. elegans* generated by the WormWiring project. The format of that dataset did not match the format of the default dataset used within c302, e.g. the naming convention of cells differed from each other, thus we had to create a Python script to convert the data. We also had to add a new parameter to c302, enabling it to make use of our script reading the new connectome and using it as input for generating the network.

```
def setup(...):
  # Use UpdatedSpreadsheetDataReader.py for reading the 'WormWiring'-
  connectome
  data_reader = "UpdatedSpreadsheetDataReader"
  ...
  nml_doc = c302.generate(..., data_reader, ...)
```

- *muscles_to_include* - To enable the user to define only a subset of muscles to simulate instead of simulating all muscles, what makes a huge difference regarding computational resources needed for the simulation, we changed an existing parameter from taking a boolean value, defining if muscles should get simulated or not, to accept also a list as value, defining which muscles should get included during the simulation.

```
def setup(...):
  # To include all muscle cells
  muscles_to_include = True
  # To include only a subset of muscle cells, e.g.
  muscles_to_include = ["MDL10", "MDR10", "MVL10", "MVR10"]
  ...
  nml_doc = c302.generate(..., muscles_to_include, ...)
```

- *conn_number_override* - This parameter was already available but it had a little flaw. While it was possible to override the number of connections of synapses, e.g. with `conn_number_override={"AVAL-DA1": 1}` for changing the chemical synapse between the presynaptic cell 'AVAL' and the postsynaptic cell 'DA1' to have one connection only, an existing gap junction connecting AVAL and DA1 also got modified. We changed that, by introducing a new convention to define gap junctions differently. Now, the user can override the number of connections of chemical and electrical synapses separately.

```
def setup(...):
  # Override the number of connections of specific synapses
  conn_number_override = {"AVAL-DA1": 2, "AVAL-DA1_GJ": 1}
  ...
  nml_doc = c302.generate(..., conn_number_override, ...)
```

- *conns_to_include/conns_to_exclude* - Before we applied our changes to c302, the user defined a list of cells and all connections of the connectome, between cells from the list, got used for a simulation. We changed this behavior, by adding parameters in the form of lists, enabling the user to define a subset of connections which gets used during a simulation. We want to stress out that, in general, a connection will not get added to the network, unless the presynaptic and postsynaptic cells are part of the network as well, i.e. the non-empty include from below effects the generation of the network, only if VB2 and VB4 are part of the network.

```
def setup(...):
    # Include all but the (bidirectional) gap junction between the
   neurons 'VB2' and 'VB4':
    conns_to_include = []
    conns_to_exclude = ["VB2-VB4_GJ", "VB4-VB2_GJ"]
    # Include the (bidirectional) gap junction only:
    conns_to_include = ["VB2-VB4_GJ", "VB4-VB2_GJ"]
    conns_to_exclude = []
    ...
    nml_doc = c302.generate(..., conns_to_include, conns_to_exclude,
   ...)
```

- *conn_polarity_override* - The polarity of most of the synapses, whether they are excitatory or inhibitory, is not known in *C. elegans*. We added a parameter in the form of a dictionary, to override the default polarity of specific connections, e.g.

```
def setup(...):
    # Override the polarity of specific synapses
    conn_polarity_override = {"AVAL-DA1": "inh"}
    ...
    nml_doc = c302.generate(..., conn_polarity_override, ...)
```

- *param_overrides* - The default parameters, depending on the biophysical level of detail, are specified in the parameters file, e.g. parameters_C2.py. Inside that file, the user only could change the parameter of a collection of synapses, e.g. the conductance of all excitatory synapses. To override parameters of a specific synapse inside the file defining the neural network, we introduced the parameter `param_overrides` in the form of a dictionary. Now, it is possible to change the conductance of a specific excitatory synapse, e.g. the conductance of the excitatory chemical synapse from AVBR to VD3

```
def setup(...):
    # Override specific parameters of synapses
    param_overrides = {"AVBR_to_VD3_exc_syn_conductance", "0 nS"}
    ...
    nml_doc = c302.generate(..., param_overrides, ...)
```

- *mirrored_elec_conn_params* - Gap junctions are modeled as bidirectional connections. Inside the NeuroML files, generated by c302, there are two elements defining the bidirectional connection, one from cell A to cell B, and another one from cell B to cell A. In order to enable the user to make the same changes to both elements, instead of having to change the same parameter of both elements separately, we introduced the parameter *mirrored_elec_conn_params* (a dictionary inside the *param_overrides* dictionary).

```
def setup(...):
  # Override the conductance of gap junction elements 'AVBL-VB1' and '
  VB1-AVBL'
  param_overrides = {
    "mirrored_elec_conn_params": {
      "AVBL_to_VB1_GJ_elec_syn_gbase": "0.001 nS"
    }
  }
  ...
  nml_doc = c302.generate(..., param_overrides, ...)
```

- *Regular expressions* - We made it possible to override parameters using regular expressions, to modify a group of specific synapses with just one line, e.g. changing synapses from DB (DB1-DB7) motor neurons to DD (DD1-DD6) motor neurons, which are excitatory by default, to be inhibitory.

```
# Override the polarity of a group of synapses (DB-DD)
conn_polarity_override = {r"^DB\d+-DD\d+$": "inh"}

# Change the weights of all synapses to 1
conn_number_override = {r"^.+-.+$": 1}

# Override the conductance of a group of gap junction elements (AVB-DB
   and DB-AVB)
param_overrides = {
  "mirrored_elec_conn_params": {
    r"^AVB._to_DB\d+\_GJ$_elec_syn_gbase": "0.001 nS",
  }
}
```

- *Multiple inputs for one cell* - We made it possible to define multiple external inputs for one cell during the simulation.

```
def setup(parameter_set, ...):
  exec ('from parameters_%s import ParameterisedModel' % parameter_set
  , globals())
  params = ParameterisedModel()
  ...
  input_list = []

  input_list.append(("DB1", "190ms", "250ms", "3pA"))
  input_list.append(("DB1", "990ms", "250ms", "3pA"))
  ...
  nml_doc = c302.generate(..., param_overrides, ...)
  for (cell, start, dur, current) in input_list:
    c302.add_new_input(nml_doc, cell, start, dur, current, params)
```

We implemented our simplified neural circuit responsible for inducing forward crawling, described in section 3.3.1, within the OpenWorm platform. Hence, we created a Python file, c302_FW.py (listing A.1), to define the neural circuit and the simulation properties. In that file, the structure of the network is defined in line 35-37, the parameters of the synapses are modified in line 45-56 and 125-180, and the external currents, which get injected into different cells during the simulation, are specified in line 58-123.

3.4.3 Sibernetic - OpenWorm’s Body Simulation Framework

Sibernetic is a 3D computational biomechanical model of the *C. elegans* body and muscles converted into a 3D particle system and a simulator [PKL16, PKL18]. The simulation engine of Sibernetic implements the Predictive–Corrective Incompressible Smoothed Particle Hydrodynamics (PCISPH) algorithm [SP09] written in OpenCL and has a 3D visualization which builds on top of OpenGL. Sibernetic can simulate different predefined environments, e.g., filled with a gel or a fluid, for the simulation of crawling or swimming worms. The output of the simulation, e.g., a curvature plot, can then be compared to measurements of experiments using real worms. While writing this thesis, the feature for comparing simulated data with data of real worms was still under development.

After installing Sibernetic and its dependencies, a simulation of the body of the worm can be started as follows:

```
$ cd SIBERNETIC_PATH # Change current working directory to the location of Sibernetic
$ ./Release/Sibernetic -f worm # Run the body simulator with ’worm’ configuration
```

The following commands will run a Sibernetic simulation in combination with c302 providing the activity of muscle cells (generated by NEURON), as input:

```
$ cd SIBERNETIC_PATH # Change current working directory to the location of Sibernetic
$ export C302_HOME=PATH_TO_C302/c302
$ export PYTHONPATH=$PYTHONPATH:$C302_HOME:./src
$ python sibernetic_c302.py
```

Inside the `sibernetic_c302.py` script, there are several parameters defining the simulation, which can be changed by modifying the file directly or by executing the script with arguments overriding the default values of these parameters.

To following command will run a 5-second simulation of the body of the worm using the output of the forward crawling circuit as input:

```
$ cd SIBERNETIC_PATH # Change current working directory to the location of Sibernetic
$ python sibernetic_c302.py -duration 5000 -reference FW -c302params C2 -datareader UpdatedSpreadsheetDataReader2 -configuration worm_crawl_half_resolution
```

Executing `sibernetic_c302.py` will generate NEURON code for the c302 simulation (using pyNeuroML), run Sibernetic with the simulation of c302, using NEURON, running in the background, and save the resulting files inside a directory with a unique name (no Sibernetic GUI will be shown). The simulation directory will be created automatically with a name composed of the name of the parameter set, e.g. 'C2', the name of the network configuration, e.g. 'FW', and a timestamp of the start of the simulation. The simulation can then be rerun as follows (displayed using the Sibernetic GUI):

```
$ cd SIBERNETIC_PATH # Change current working directory to the location
of Sibernetic
# Displaying the result of a simulation with the Sibernetic GUI
$ ./Release/Sibernetic -l_from lpath=simulations/C2_FW_TIMESTAMP
```

A more detailed description of the installation and usage of Sibernetic can be found online in the 'README' of the GitHub repository of Sibernetic[5].

3.5 Virtualization Platform

Computational work becomes more and more part of scientific research projects while reproducing the outcome of scientific work has always considered being critical. Due to the rapidly changing and heterogeneous computer environments, reproducing some results can be a challenging task. Different platforms or operating systems, but also different versions of the same operating system running on the same platform can lead to the failure of running necessary software for reproducing the results. Different versions of software dependencies are often the source of such an issue. However, even within the very same environment, it can be hard to install the necessary software, e.g., due to the lack of documentation regarding the installing process [Boe15].

With Docker[6], a virtualization platform, it is possible to run software with all the dependencies on specific versions in an isolated environment, known as 'docker container'. The isolation of the environment can help to make software portable. Docker containers are comparable to virtual machines, but the containers are more lightweight and have a better performance [Boe15, RBA17].

We created a docker container[7], containing a pipeline of OpenWorm tools, which can be used to run OpenWorm simulations. The start of the container simulates the nervous system of *C. elegans*, feeds the output (the calcium concentration of the muscles) to the simulation of the worm's body and creates a movie of the behaving worm. In addition to the movie, different plots get generated, e.g., traces and heatmaps of membrane potential dynamics and intracellular calcium concentration, or a curvature plot. Datasets of raw data over time, generated during the simulation, are accessible and can also be analyzed with other tools.

For the automated setup of the Docker container, a 'Dockerfile' is needed containing all the necessary commands of the installation process of the OpenWorm tools, tools for creating a video of the output of the Sibernetic simulation, and their dependencies.

The installation process of the container can be summarized as follows:

---

[5] `https://github.com/openworm/sibernetic`

[6] `www.docker.com`

[7] `https://github.com/openworm/OpenWorm`

1. Install and update base operating system.
2. Install dependencies.
3. Install NEURON with Python support.
4. Clone and install OpenWorm tools:
    - 4.1. pyNeuroML
    - 4.2. PyOpenWorm
    - 4.3. c302
    - 4.4. Sibernetic
5. Set environment variables.
6. Copy main Python script (master_openworm.py), containing the commands for executing the simulation, into the container.

The commands inside master_openworm.py, for executing the simulation, are as follows:

1. Run Sibernetic simulation with parameters defined inside master_openworm.py - This will execute Sibernetic with c302 running in the background.
2. Move generated plots to a directory accessible from outside the docker container.
3. Rerun the latest Sibernetic simulation while displaying the graphical output of the behaving worm on a virtual screen.
4. Create a video of the simulation, by recording the output of the virtual screen.
5. Remove black frames at the beginning of the video.
6. Create a speed up version of the video.

A simulation of our forward crawling circuit, which is the default network model, can be started within the docker container as follows:

```
$ git clone https://github.com/openworm/openworm # Clone GitHub
repository of OpenWorm
$ cd openworm # Change current working directory to the location of
OpenWorm
$ ./build.sh # Create the docker container (create the isolated
environment)
$ ./run.sh # Run a 15 ms (default duration) simulation of our simplified
forward crawling network model using Sibernetic in combination with c302
running in the background.

# To change the duration, the argument ’-d’ can be used
$ ./run.sh -d 5000 # Run a 5 sec simulation
```

At the time of writing this thesis, only the duration of the simulation could be specified using a parameter. Other parameters, e.g., the name of the network configuration, or the parameter set, could only be changed directly inside the Python script 'master_openworm.py'

A more detailed description of the installation and usage of the docker container can be found online in the GitHub repository of OpenWorm[8].

---

[8] `https://github.com/openworm/openworm`

CHAPTER 4

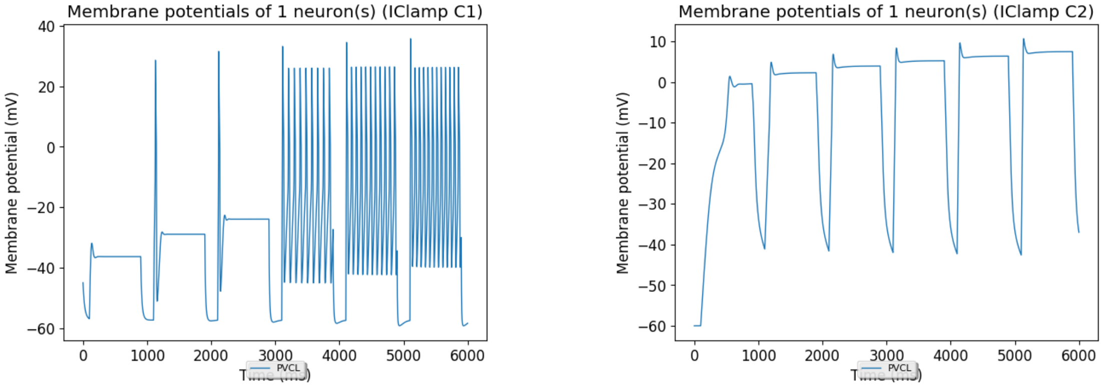
Experiments and Results

In this chapter we describe the in silico experiments we run, using the OpenWorm platform, and their results. In section 4.1, we modified the preexisting computational neuron model to change the dynamics of the membrane potential. The goal of this experiment was to get a neuron model generating graded membrane potentials instead of burst firing. In section 4.2, we describe experiments we run using an optimization algorithm. In this series of experiments, the goal was to get a wave of dorsal body-wall muscle activities propagating from head to tail, an essential part for forward-movement of *C. elegans*. At the beginning of these experiments, we let the optimizer optimize given a parameter space including all synapses. In subsequent experiments, we included only a subset of synapses to get a better outcome. In section 4.3, we describe the output of our new forward crawling neural circuit incorporating a few assumptions, implemented within the OpenWorm platform, and show the result of a moving worm using the OpenWorm's body simulator. We also show the output generated by the same circuit while sequentially omitting one of the assumptions.

4.1 Modifying OpenWorm's Neuron Model

At the beginning of our experiments, the single-compartmental neuron model, used within the OpenWorm platform, the membrane potential of neurons showed bursting behavior. One of our goals was to create a more biologically realistic model of *C. elegans* neurons. There is evidence that some neurons of *C. elegans* generate graded membrane potentials, rather than all-or-none action potentials, or burst firing. [LGF09]. Thus, as one of our first experiments, we started to modify parameters of the OpenWorm parameter set of the single-compartmental conductance-based neuron model (parameters_C1.py). We did this with a simple heuristic approach where we picked a single parameter and checked the output of the simulation after modifying the parameter a little bit. The resulting file (parameters_C2.py) contains a different set of parameters, mainly with modified conductance and reversal potential of ion channels. The membrane potential of the neuron PVCL, after receiving current pulses with increasing amplitude as input (current-clamp experiment), is shown in figure 4.1. In figure 4.1 (a), the membrane potential of PVCL, using the old parameters, starts to show heavy burst firing, with an input current pulse of 4 pA over 800 ms. Figure 4.1 (b) shows PVCL generating graded membrane potentials with increased amplitude, using the same input protocol and our new parameter set. Because all neurons share the same computational model, it did not matter which neuron we used for the simulation. The values of the parameters and the equation with the occurrence of the parameters are shown in table 4.1.

We want to note that while the output of the membrane potential using the new neuron model is probably more realistic, the parameters of that model are most likely unrealistic. To get accurate parameters, one needs to do electrophysiological experiments to measure the currents going throw all different kinds of ion channels of every neuron, because the combination of ion channels differs from neuron to neuron. Without these experiments, parameters can only be guessed, because it is possible to create the same output with different sets of parameters.



(a) (b)

Figure 4.1: Comparison of simulated current-clamp experiments with different parameters. 6 different current pulses, from 1 pA to 6 pA with a duration of 800 ms, got injected into the neuron. (a) Membrane potential traces of neurons generated with the parameter set 'C1'. (b) Membrane potential traces of neurons generated with our parameter set 'C2'

4.2 Optimization of a C. elegans' brain cell network

In the early experiments, the goal was to create a propagating wave in a sequence of dorsal body wall muscles in the *C. elegans*, what is needed for locomotion. In this experiments, automatic optimization was used to optimize parameters. However, too many parameters to optimize along creates the problem of being stuck in a local minimum. Therefore, further experiments used a targeted approach to select a subset of parameters to allow

Table 4.1: Neuron model parameters

Parameter	**Value**	**Used in equation X**
C_{spec}	5 uF_per_cm2	3.4
$diam$	5	3.5
$init_{V_m}$	-60 mV	
G_{K_f}	0.042711643917483308 mS_per_cm2	3.6
E_{K_f}	-70 mV	3.6
τ_p	2.25518 ms	3.10
V_{mid_p}	-8.05232 mV	3.11
κ_p	7.42636 mV	3.11
τ_q	149.963 ms	3.10
V_{mid_q}	-15.6456 mV	3.11
κ_q	-9.97468 mV	3.11
G_{K_s}	0.45833751019872582 mS_per_cm2	3.7
E_{K_s}	-60mV	3.7
τ_n	25.0007 ms	3.10
V_{mid_n}	19.8741 mV	3.11
κ_n	15.8512 mV	3.11
$G_{Ca^{2+}}$	1.812775772264702 mS_per_cm2	3.8
$E_{Ca^{2+}}$	10 mV	3.8
τ_e	0.100027 ms	3.10
V_{mid_e}	-3.3568 mV	3.11
κ_e	6.74821 mV	3.11
τ_f	150.88 ms	3.10
V_{mid_f}	25.1815 mV	3.11
κ_f	-5.03176 mV	3.11
α_h	0.282473	3.8
$[Ca^{2+}]_{half_h}$	6.41889e-8 mM	3.12
κ_h	-1.00056e-8 mM	3.12
G_{Leak}	0.002 mS_per_cm2	3.9
E_{Leak}	-60 mV	3.9

optimization to occur over. Some results from automated optimization are being fed into future experiments that used a heuristic approach where we picked a single parameter and modified it until we could observe a better result.

The cell network model, we used for the simulation, comprised the two command interneurons AVBL and AVBR, 7 DB motor neurons, and 95 dorsal body wall muscles (figure 4.2 and 4.3). Aside from 3 excitatory chemical synapses, the neurons are connected by forming gap junctions. One of the three chemical connections, from AVBR to DB4, has only one synapse; thus it is possible that it is of not much importance for the functioning of the network. The other two chemical connections, from AVBL to AVBR and vice versa, as well as the gap junction between the command neurons, most likely play a role in synchronizing their activity. Interestingly, there is a gap junction with only one synapse connecting DB3 and DB4. This could indicate, that the DB motor neurons can be divided into two subgroups. DB motor neurons activate dorsal body wall muscles exclusively via excitatory chemical synapses. As shown in figure 4.3, body wall muscles located in the head of *C. elegans* don't receive input coming from DB motor neurons. Instead, there is another subcircuit within the nervous system of *C. elegans* responsible for the activation of body wall muscles located in the head.

The rationale for the subsets of circuits that are being chosen is partially based on the goal, but also based on a decomposition process for simplicity of optimization.

The first goal was to go from simultaneous activation of muscles to get the first spike time to be staggered.

The first experiment (section 4.2.1) did not produce good results for first spike time because it hit a local minimum. The next experiment (section 4.2.2) used a subset of parameters but produced reasonable first spike times. The experiment on (section 4.2.3) attempted to optimize the spike frequency of the muscles, and was successful in spike frequency but then spike timing of subsequent spikes had jitter. In later experiments (section 4.2.4 and 4.2.5) we began hand tuning the spike timing of subsequent spikes.

4.2.1 Step 1: Optimization of the whole network

In our first optimization experiment, the goal was to get staggered activation of DB motor neurons (table 4.3), what should result in staggered activation of dorsal body wall muscles.

A more abstract representation of the neural network, used for the optimization experiments, is shown in figure 4.4.

We let the optimizer optimize along all synaptic parameters with the termination condition of 2000 evaluations. The parameters with valid ranges and a reference to the computational synapse models are listed in table 4.2.

During 1 second of real-time simulation, a current pulse with an amplitude of $15pA$ got injected into AVB command interneurons, with an offset of $50ms$ and duration of $850ms$.

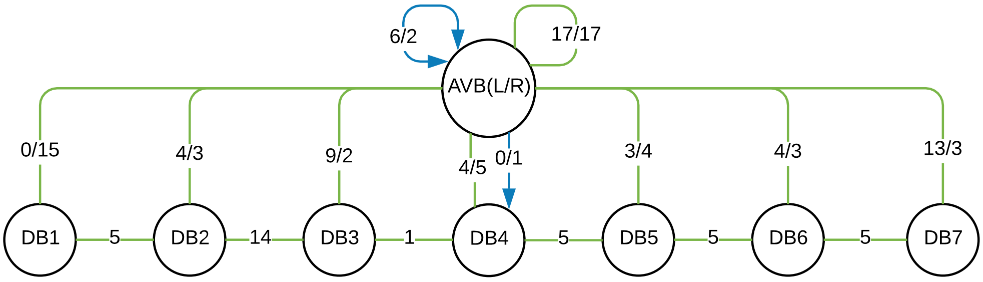


Figure 4.2: C. elegans' neural subcircuit comprising AVB command neuron pair and DB motor neurons. Green lines represent gap junctions, the blue arrow a chemical synapse. The labels, in the form of x/y, indicate the number of synapses connecting the neurons, e.g. '0/15' denotes that, within the nervous system of *C. elegans*, there is no gap junction from AVBL to DB1 but one with 15 synapses from AVBR to DB1.

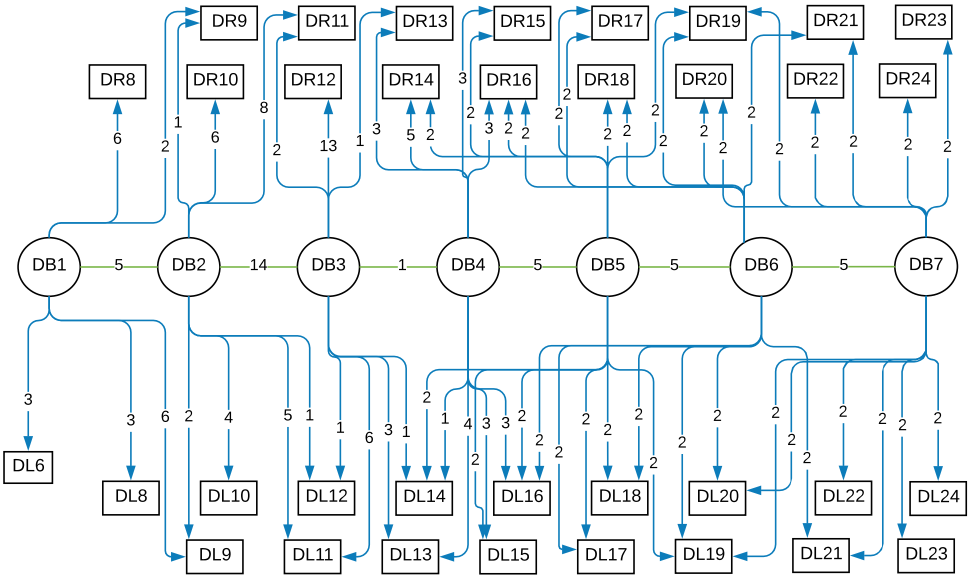


Figure 4.3: C. elegans' neural subcircuit comprising DB motor neurons (circles) and dorsal body wall muscles (rectangles). Green lines represent gap junctions, blue arrows chemical synapses. The labels indicate the number of synapses connecting the cells.

We did not choose to inject current for the whole simulation time, because we wanted to check whether the membrane potential decreases again after the end of the input current.

The resulting time of activation of the DB motor neurons, with a 139 ms - 147 ms range, are listed in table 4.4. The resulting cell dynamics after the optimization are shown in table 4.5. Column 1 and 3 show the membrane potentials of neurons and muscles in the form of a heatmap, column 2 and 4 show the same data as traces. Column 5 and seven show internal calcium concentration of neurons and muscles in the form of a heatmap, column 6 and 8 show the same data as traces. The results show that the outcome of the optimization was not good, probably because the optimization got stuck in a local minimum with the given number of maximal evaluations. It can be seen that DB motor neurons got activated almost at the same time, thus, also muscle cells, innervated by DB neurons, got activated at the same time.

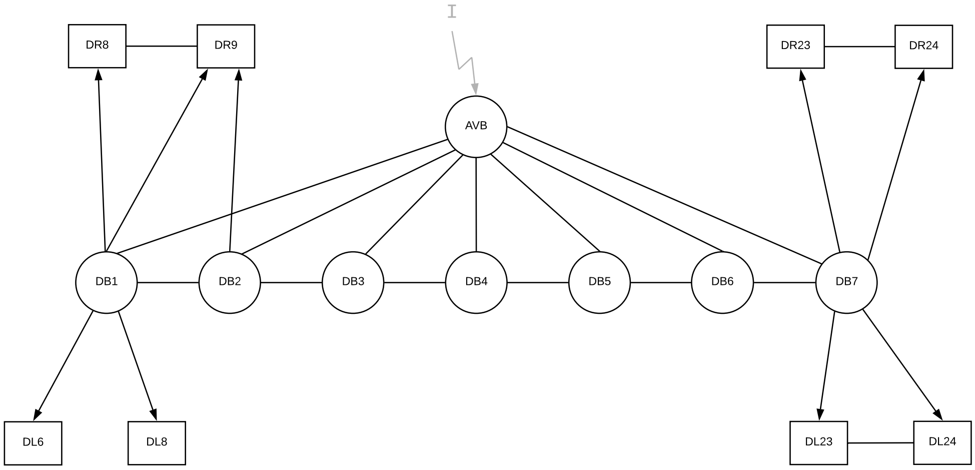


Figure 4.4: Neural network used for the optimization. Network comprises AVB command neuron pair, DB motor neurons, and dorsal muscle cells (not all are shown). Black arrows represent chemical synapses. Black lines represent gap junctions. Gap junctions from AVB to DB neurons use our 'delayed' gap junction model. During the optimization, AVB neurons receive an ongoing input pulse.

4.2.2 Step 2: Partial Optimization: AVB-DB, DB-DB

In the previous experiment, the parameter space includes parameters of connections from AVB to DB neurons and DB to dorsal muscles. That means the optimization is running for all parameters (conductances of synapses motor neurons to muscles, interneurons to motor neurons, delay functions). The previous experiment is stuck in a local minimum, defined by hitting the maximum number of evaluations. In that experiment, we are optimizing for the first spike time of the DB motor neurons, while introducing a delay to try to get a sequence of spike times.

In this experiment, AVB to DB delayed gap junction, and DB-DB gap junction connections were the only thing that was able to be optimized among, all other parameters were fixed (4.6). We also increased the maximum number of evaluations to 4000. Same target of optimization, first spike time of motor neurons, was made (table 4.7). This is shown on the diagram by the darker lines between AVB and the DB motor neurons, and the lines between DB motor neurons, gray lines are connections with fixed parameters (figure 4.5).

Table 4.2: Parameter ranges for the optimization (step 1)

Parameter	**Range**	**Equation**
	AVB-DB	
g_s	[0.00052, 0.06252] nS	3.25
σ	[0.1, 0.9] per_mV	3.26
μ	[-70, 20] mV	3.26
	DB-DB	
g_s	[0.00002, 0.04252] nS	3.24
	DB-Muscle	
g_s	[0.02, 2] nS	3.17
	Muscle-Muscle	
g_s	[0.00002, 0.00050] nS	3.24

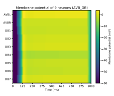
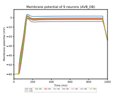
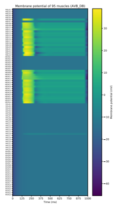
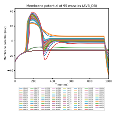
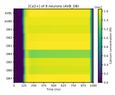
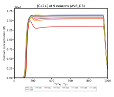
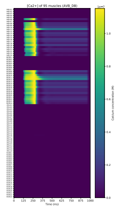
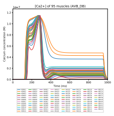
Table 4.3: Targets of the optimization (step 1)

First spike time	
DB1	115 ms
DB2	125 ms
DB3	135 ms
DB4	145 ms
DB5	155 ms
DB6	165 ms
DB7	175 ms

Table 4.4: Result of the optimization (step 1)

First spike time	
DB1	138.55 ms
DB2	143.35 ms
DB3	143.7 ms
DB4	147.05 ms
DB5	145.95 ms
DB6	146.8 ms
DB7	144.9 ms

Table 4.5: AVB-DB-DorsalMuscles optimization 1

Neurons (Voltage)	Neurons (Voltage)	Muscles (Voltage)	Muscles (Voltage)	Neurons ($[Ca^{2+}]$)	Neurons ($[Ca^{2+}]$)	Muscles ($[Ca^{2+}]$)	Muscles ($[Ca^{2+}]$)
							

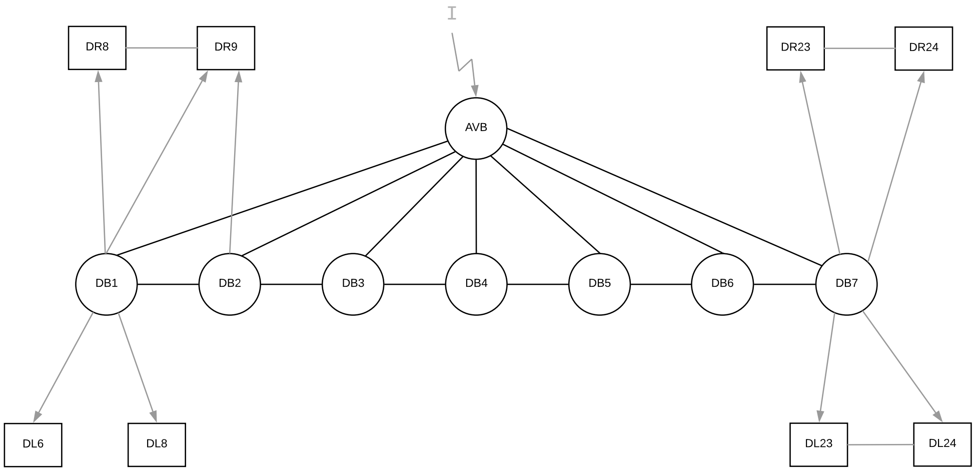


Figure 4.5: Neural network used for the optimization. The network comprises AVB command neuron pair, DB motor neurons, and dorsal muscle cells (not all are shown). Black arrows represent chemical synapses. Black lines represent gap junctions. Gap junctions from AVB to DB neurons use our 'delayed' gap junction model. During the optimization, AVB neurons receive an ongoing input pulse. Gray lines (chemical synapses from DB neurons to dorsal muscles) indicate connections (parameters thereof) which are set to a fixed value during the optimization.

The result was some success (116 ms - 170 ms range). However, the very last DB neuron showed a spike somewhat before the second to last DB neuron (table 4.8).

The resulting cell dynamics after the optimization are shown in table 4.9. It can be seen that, compared to the previous experiment, there is more delay between the activation of anterior and posterior muscle cells. Also, the muscle cells generate multiple spikes.

Table 4.6: Parameter ranges for the optimization (step 2)

Parameter	**Range**	**Equation**
	AVB-DB	
g_s	[0.00052, 0.06252] nS	3.25
σ	[0.1, 0.9] per_mV	3.26
μ	[-90, 40] mV	3.26
	DB-DB	
g_s	[0.00002, 0.04252] nS	3.24

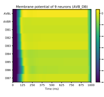
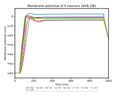
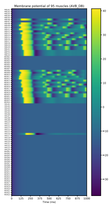
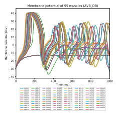
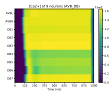
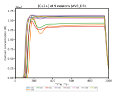
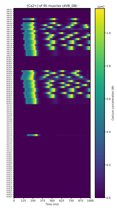
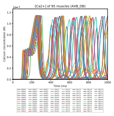
Table 4.7: Targets of the optimization (step 2)

First spike time	
DB1	115 ms
DB2	125 ms
DB3	135 ms
DB4	145 ms
DB5	155 ms
DB6	165 ms
DB7	175 ms

Table 4.8: Result of the optimization (step 2)

First spike time	
DB1	115.55 ms
DB2	130.6 ms
DB3	135.05 ms
DB4	153.25 ms
DB5	155.4 ms
DB6	169.95 ms
DB7	166.55 ms

Table 4.9: AVB-DB-DorsalMuscles optimization 2

Neurons (Voltage)	Neurons (Voltage)	Muscles (Voltage)	Muscles (Voltage)	Neurons ($[Ca^{2+}]$)	Neurons ($[Ca^{2+}]$)	Muscles ($[Ca^{2+}]$)	Muscles ($[Ca^{2+}]$)
							

4.2.3 Step 3: Partial Optimization: DB1-Muscles, DB2-Muscles

In the previous experiment, we narrowed the parameter space and optimized only the gap junction connections between AVB command interneurons to DB motor neurons, and between DB to DB neurons. In this experiment, we are taking the parameters that came out of the previous experiment for those connections, and we are fixing them in this experiment. We were optimizing over the conductance of DB to muscle excitatory chemical connections only. We have focused on DB1 and DB2 muscle connections only because to do it over all the DBs would still be too many parameters (table 4.10).

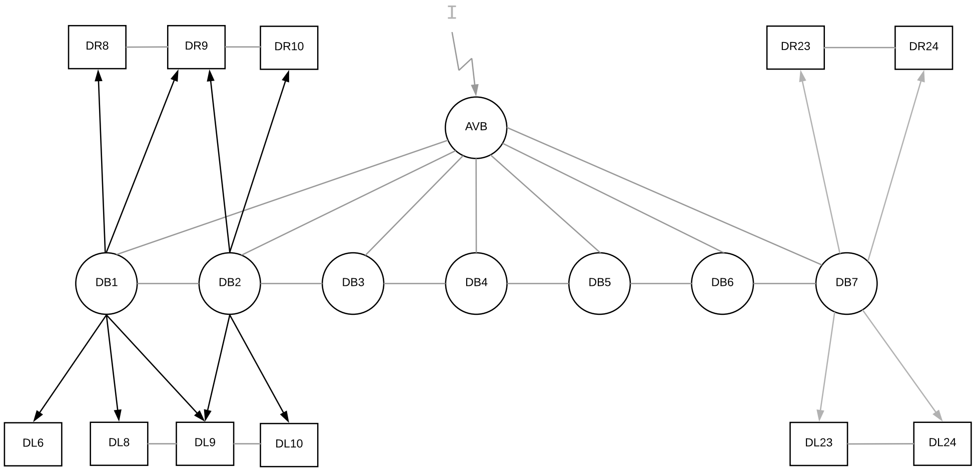


Figure 4.6: Neural network used for the optimization. Network comprises AVB command neuron pair, DB motor neurons, and dorsal muscle cells (not all are shown). Black arrows represent chemical synapses. Black lines represent gap junctions. Gap junctions from AVB to DB neurons use our 'delayed' gap junction model. During the optimization, AVB neurons receive an ongoing input pulse. Gray lines (gap junctions) indicate connections (parameters thereof) which are set to a fixed value during the optimization.

Table 4.10: Parameter ranges for the optimization (step 3)

Parameter	**Range**	**Equation**
	DB-Muscle	
g_s	[0.02, 2] nS	3.17

Table 4.11: Targets of the optimization (step 3)

Mean spike frequency	
MDL06	4.0 Hz
MDL08	4.0 Hz
MDL09	4.0 Hz
MDL10	4.0 Hz
MDR08	4.0 Hz
MDR09	4.0 Hz
MDR10	4.0 Hz

Table 4.12: Result of the optimization (step 3)

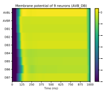
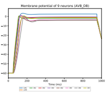
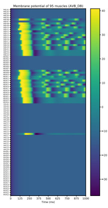
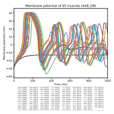
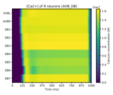
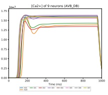
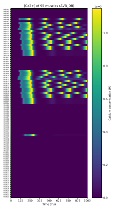
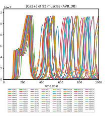
Mean spike frequency	
MDL06	4.077 Hz
MDL08	4.011 Hz
MDL09	3.962 Hz
MDL10	3.960 Hz
MDR08	4.101 Hz
MDR09	3.990 Hz
MDR10	4.032 Hz

Here we were targeting the spike frequency of the muscles to be 4 HZ (table 4.11), with a maximum number of evaluations, used as the stopping criterion, of 4000. The result was the muscles spiking within 3.9 - 4.1 HZ range (table 4.12). However, as it can be seen in the result, the first spike time looked good, but the subsequent spikes had a jitter in them (table 4.13).

4.2.4 Step 4: Hand-tuning DB-Muscle conductance

In these experiments, we were hand-tuning the conductances of excitatory synapses between DB motor neurons and muscle cells (only dorsal left). Because the results from

Table 4.13: AVB-DB-DorsalMuscles optimization 3

Neurons (Voltage)	Neurons (Voltage)	Muscles (Voltage)	Muscles (Voltage)	Neurons ($[Ca^{2+}]$)	Neurons ($[Ca^{2+}]$)	Muscles ($[Ca^{2+}]$)	Muscles ($[Ca^{2+}]$)
							

the optimizer were not satisfactory, doing some hand-tuning allowed for the development of intuitions of the effects of changing specific parameters. In this case, we are starting from only the parameters that provide a delayed spike time for the DB neurons from section 4.2.2, but not using the results of optimizing the spike frequencies from the previous experiment from section 4.2.3. The target for the hand tuning is an improved version of having multiple spikes staggered without the jitter that was found in the experiment in section 4.2.3. The strategy, to begin with, was to begin picking connections between motor neurons and muscles and changing conductances in a trial and error approach.

In case01-case05 we made some minor changes of the delayed GJs between AVB and DB neurons with the resulting parameters of a previous optimization, we also changed the conductance to muscles. Although we could observe a wave of action potentials propagating from neck to tail (case05), there was an issue with these parameters.

We were optimizing the wave of the first spikes, and after we increased the simulation duration, we noticed that there was only one complete wave (case06) because motor neuron activity still changed after the first spike had occurred. That meant there was jitter. Looking at the problem, the motor neurons do not have steady-state behavior until later in the simulation.

In subsequent experiments, we were trying to create a wave propagating from neck to tail, but this time we only had a look at the muscle output after motor neurons had reached steady state. This is done by changing the conductance between the motor neuron and muscle cell. If there is a muscle spiking too early or too late for the sequence, then we are shifting it by changing the weight of the synaptic conductance from DB to muscle. In order to take into account the jitter in later spikes, instead of targeting the first spike time, we are looking at the timing of the second, third or fourth spike and adjusting the conductance to get it to be at the right time in sequence.

In case 30 – they are well aligned in the DL muscles. The DRs are not optimized, so they act as control by using the default values. Increasing perfection was happening after that. By case 38 they are very well aligned, but the timings are not as linear as desired.

Some of the resulting cell dynamics after the hand-tuning experiments are shown in table 4.14. The entire table with all the experiments, as well as the NeuroML file of the network containing the parameters of ion channels and synapses, can be found online[1]

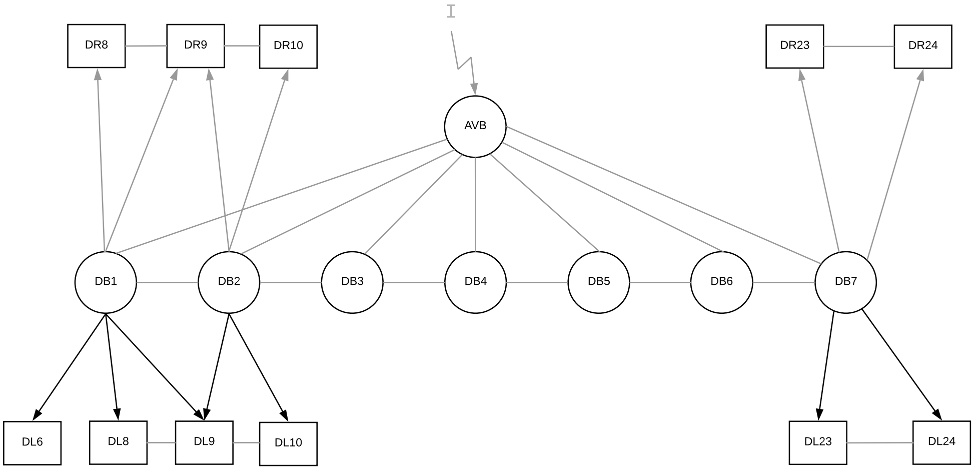


Figure 4.7: Neural network used for hand-tuning experiments. The network comprises AVB command neuron pair, DB motor neurons, and dorsal muscle cells (not all are shown). Black arrows represent chemical synapses. Black lines represent gap junctions. Gap junctions from AVB to DB neurons use our 'delayed' gap junction model. During the hand-tuning experiments, AVB neurons received an ongoing input pulse. Gray lines indicate connections (parameters thereof) which are set to a fixed value. Only the conductance of chemical synapses to dorsal left muscles was modified.

4.2.5 Step 5: Hand-tuning DB-Muscle conductance

In these experiments, we continued fine-tuning the conductance of DB-muscle excitatory synapses by hand (we only modified the conductance of connections to dorsal left muscles), while making use of the 'delayed' GJs between AVB and DB neurons. We also increased the simulation duration, even more, to check whether later spikes propagate as desired from neck to tail.

The result of case34 (table 4.15) is still not perfect yet - posterior muscles do not receive as much excitatory synaptic input as anterior muscles, resulting in different spike frequencies - but it could already be used as a basis of further experiments.

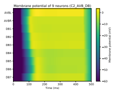
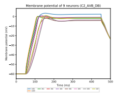
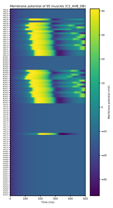
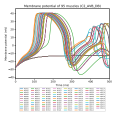
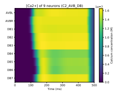
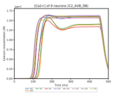
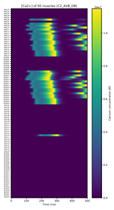
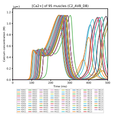
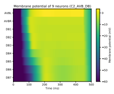
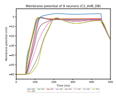
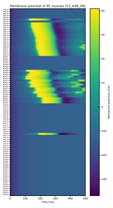
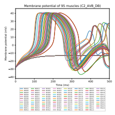
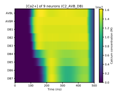
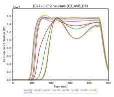
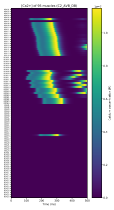
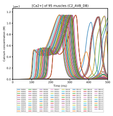
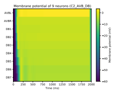
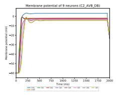
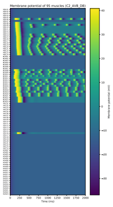
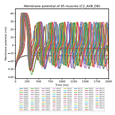
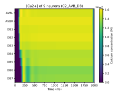
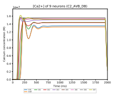
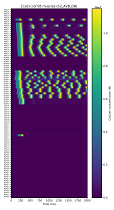
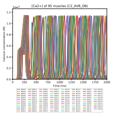
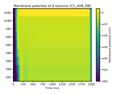
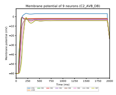
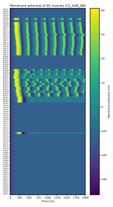
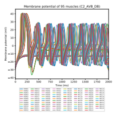
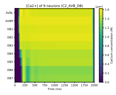
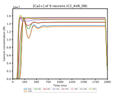
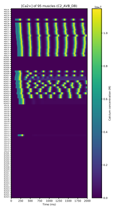
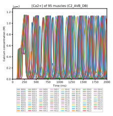
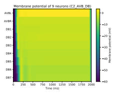
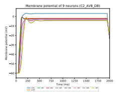
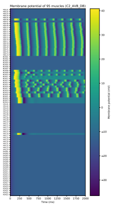
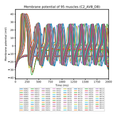
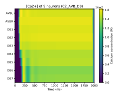
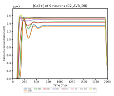
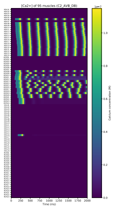
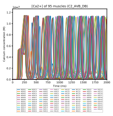
The entire table with all the experiments, as well as the NeuroML file of the network containing the parameters of ion channels and synapses, can be found online[2]

---

[1] `https://github.com/lungd/openworm-experiments/tree/master/2017-07-31`

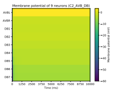
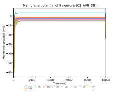
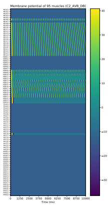
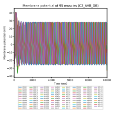
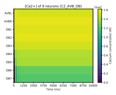
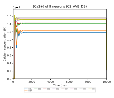
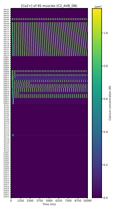
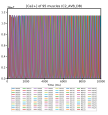
[2] `https://github.com/lungd/openworm-experiments/tree/master/2017-08-05`

Table 4.14: 7-31 results. The rows show the result of different experiments after the hand-tuning of some parameters.

Case	Neurons (Voltage)	Neurons (Voltage)	Muscles (Voltage)	Muscles (Voltage)	Neurons ($[Ca^{2+}]$)	Neurons ($[Ca^{2+}]$)	Muscles ($[Ca^{2+}]$)	Muscles ($[Ca^{2+}]$)
01								
05								
06								
30								
38								

Muscle cells do not receive only excitatory input from motor neurons; they also receive inhibitory input coming from other motor neurons (D-type motor neurons). One role of D-type motor neurons could be to balance out the excitatory input coming from DB neurons, to get the same frequency of muscle spikes. Thus, we decided to stop this series of experiments at this point.

Table 4.15: 8-05 results

Case	Neurons (Voltage)	Neurons (Voltage)	Muscles (Voltage)	Muscles (Voltage)	Neurons ($[Ca^{2+}]$)	Neurons ($[Ca^{2+}]$)	Muscles ($[Ca^{2+}]$)	Muscles ($[Ca^{2+}]$)
34								

4.3 New Forward Crawling Neural Circuit - Implementation

In this experiment, we implemented our simplified neural circuit inducing forward crawling (described in section 3.3.1) within the OpenWorm platform.

This circuit was designed, with a few assumptions, with the purpose of generating alternating dorso-ventral waves of muscle bends propagating from head to tail, leading to forward locomotion.

The output of the circuit, incorporating all our assumptions, is shown at the end of this section. Before that, we want to show the output of the circuit after weakening our assumptions by sequentially omitting one of them.

AVB command interneurons are known to be active during forward locomotion. They receive excitatory input coming from sensory neurons and other interneurons. Because we did not include those neurons, we injected current into AVB neurons directly, to keep AVB neurons active during the simulation of the forward crawling. The output of the circuit without input coming from AVB is shown in figure 4.8. Without active AVB neurons, the activity of B-type motor neurons posterior to DB1 and VB1 is reduced, due to the gap junctions between AVB and B-type neurons. Thus, also the activity of muscle cells is reduced.

Another assumption that we used while constructing the circuit was the proprioceptive effect approximated by additional excitatory chemical synapses (using the computational synapse model 3.21) between neighboring B-type motor neurons. Because it was not possible to add new connections, using c302, we had to add those connections to the connectome file containing all the synapses of *C. elegans*. We adjusted the parameters of the synapse model, by hand, to get a propagation wave of B-type neuron spikes, as shown in figure 4.13 (a). Due to the different number of B-type motor neurons in *C. elegans*, 7 DB and 11 VB motor neurons, the wave in VB neurons must propagate faster, what we took into account while adjusting the parameters of those synapses. We defined phase shifted inputs only for DB1 and VB1 neurons, i.e., we chose the starting time of the input going into VB1 so that it gets activated when DB1 is not active anymore and vice versa. The input into the first B-type motor neurons then gets propagated via our newly added chemical synapses to approximate the proprioceptive effect. The output of the circuit without periodic input into DB1 and VB1 is shown in figure 4.9. Besides directly stimulated cells, the network is entirely silent, because of weak gap junctions between AVB and B-type neurons and without activating B-type neurons the body-wall muscles do not receive any excitatory input. The output of the circuit without the additional chemical synapses between neighboring B-type motor neurons is shown in figure 4.10. DB1 and VB1 get activated, due to the directly injected input pulses, what can be seen in figure 4.10 (a). Due to the weak gap junctions connecting neighboring B-type motor neurons, the input from the first B-type neurons does not get propagated to the next neurons within the B-type neuron group. Thus, only body-wall muscles innervated by DB1 and VB1 get activated, as shown in figure 4.10 (b).

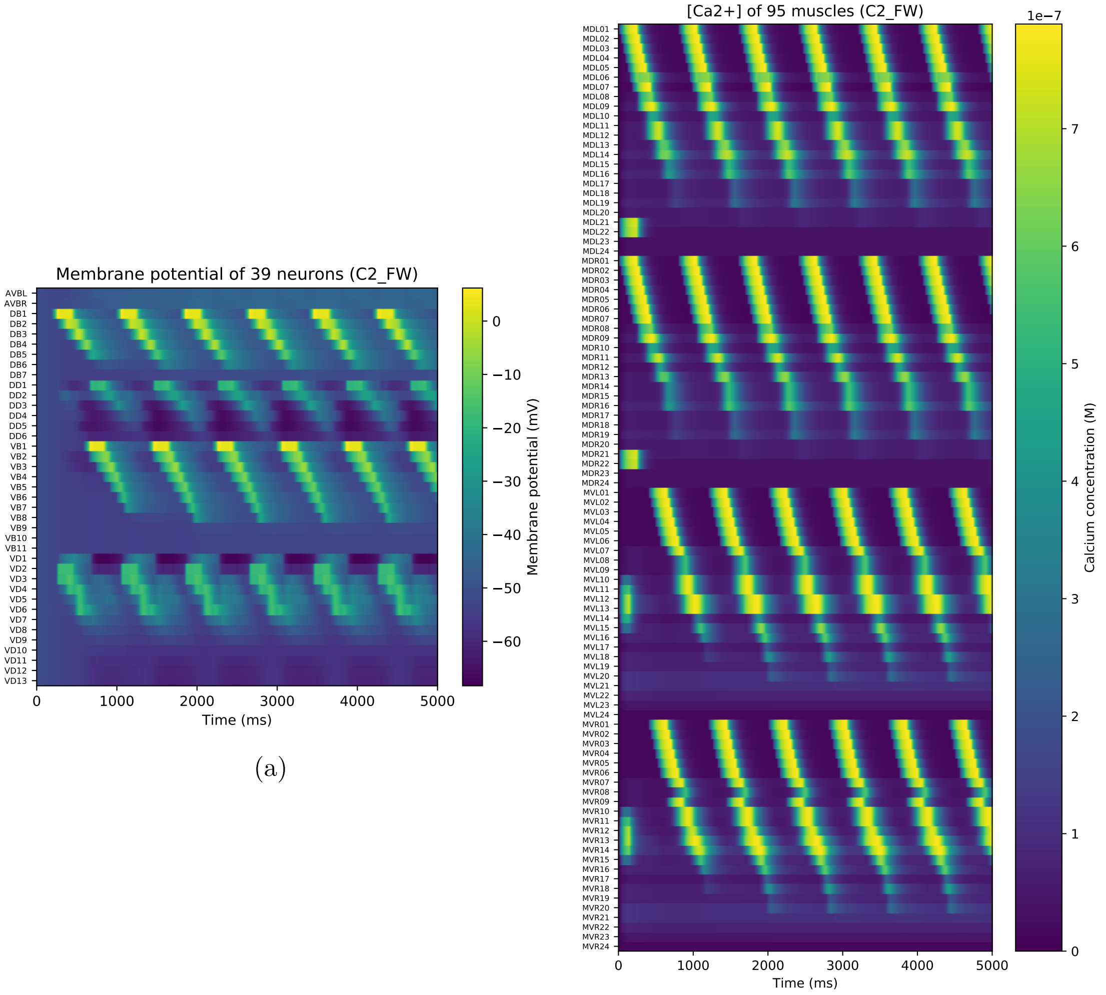
By default, all chemical synapses coming from B-type motor neurons are excitatory, synapses coming from D-type motor neurons are inhibitory. For example, DB motor neurons synapse onto VD and DD motor neurons. After a DB motor neuron gets activated, it excites VD and DD neurons at the same time through chemical synapses, resulting into inhibition of different ventral and dorsal muscle cells. Therefore, we decided to modify the polarity of chemical synapses from DB to DD and from VB to VD neurons to be inhibitory. The output of the circuit with inhibitory synapses from DB to DD and from VB to VD is shown in figure 4.11. As shown in figure 4.11 (b), ventral body-wall muscles receive almost no input, due to the reduced activity of VB motor neurons and the increased activity of VD motor neurons (figure 4.11 (a)). The activity of muscles on the dorsal side of the worm is also reduced although not as much as the activity of muscles on the ventral side. The reduced activity of dorsal muscles can be explained by the increased activity of DD motor neurons (figure 4.11 (a)).

Muscles located in the head (first seven muscles in each quadrant) don't receive input from B-type motor neurons; thus, we directly stimulated these muscles with periodic, phase shifted input pulses. We adjusted the input pulses, to generate head bends of the same frequency and aligned with the rest of the body. The output of muscle cells located in the head can be seen, for example in figure 4.9 (b). At the beginning of the simulation, some muscles left and right ventral body-wall muscles (10-15), as well as left and right dorsal muscles (21-22), also receive a single input pulse, because we did not want to start with a wholly straightened worm body.

We set the weight of all synapses to 1 because the actual weights are not known and by doing so we got a simplified circuit which was easier to analyze. The result of the circuit using the number of synapses as weight, taken from the connectome file we used, is shown in figure 4.12. We adjusted the output of our forward crawling circuit model while using equal weights. Thus, if we keep the original weights, as defined in the connectome, different neurons receive too much current via synapses, resulting in an undesired output.

The output of c302 of a 5 second of real-time simulation, incorporating all our assumptions, is shown in figure 4.13. Figure 4.13 (a) shows the simulated membrane potential of the neurons as a heatmap. Each line represents the membrane potential of a neuron over the simulated time. Yellow colored regions indicate high, dark blue low membrane potential. Figure 4.13 (b) shows the simulated internal calcium concentration inside all 95 muscle cells as a heatmap. Each line represents the internal calcium concentration of a body-wall muscle cell. Yellow colored regions indicate high internal calcium concentration. Dark blue indicates no calcium inside the muscle cell.

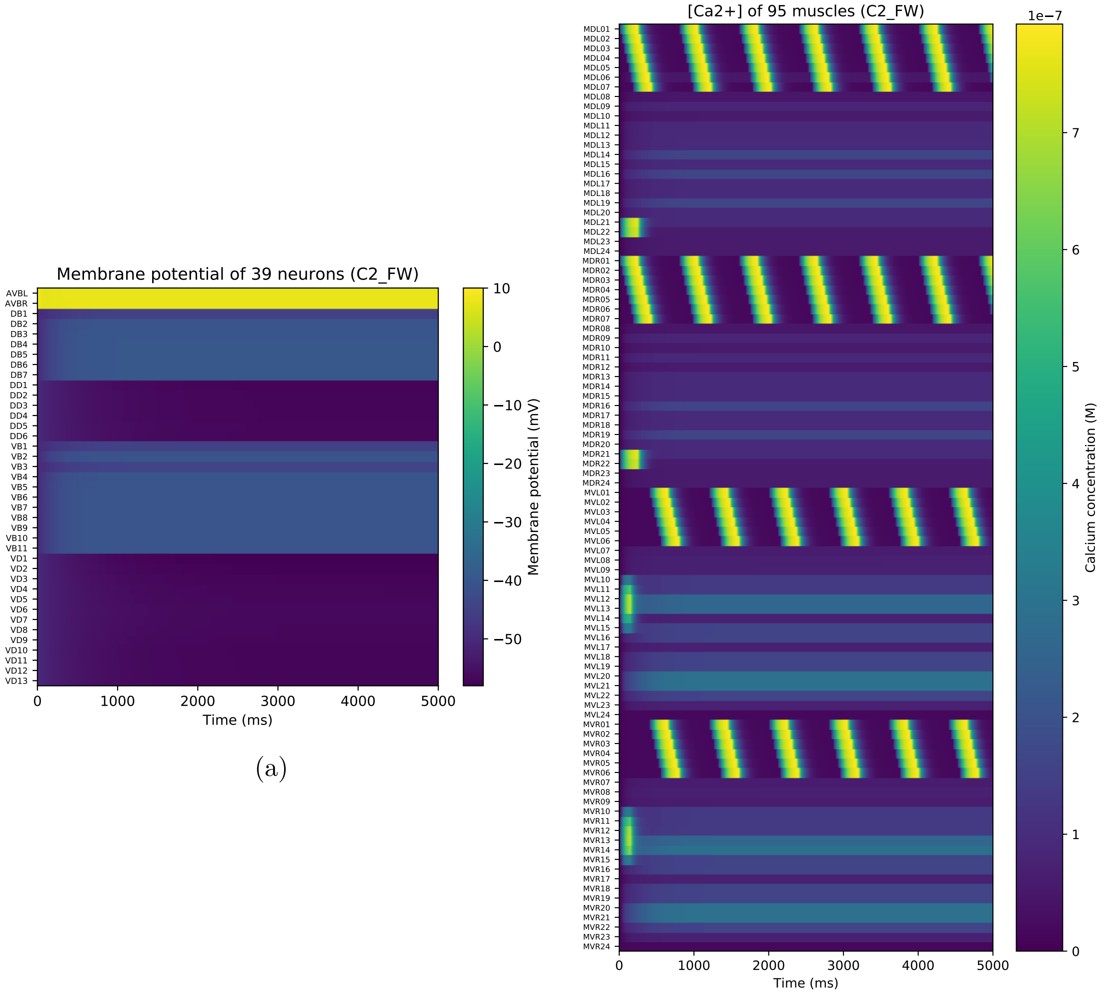
For the simulation of the body of the worm, we used Sibernetic in combination with c302. In this mode, the internal calcium concentration of muscle cells is governed by a neural network model generated with c302, and Sibernetic uses the calcium concentration of the muscles as input to determine how much a muscle should contract. The output of Sibernetic was a forward crawling worm inside a bounded environment, filled with a layer of agar. We took screenshots of Sibernetic's 3D visualization which are shown in figure 4.14.



(a)

(b)

Figure 4.8: Simulation of the worm's neural circuit and muscles, without input injected into AVB, during 5 seconds of real time simulation of the forward locomotion using c302. (a) Neuron membrane potential dynamics. (b) Internal calcium concentration of the 95 body-wall muscle cells.

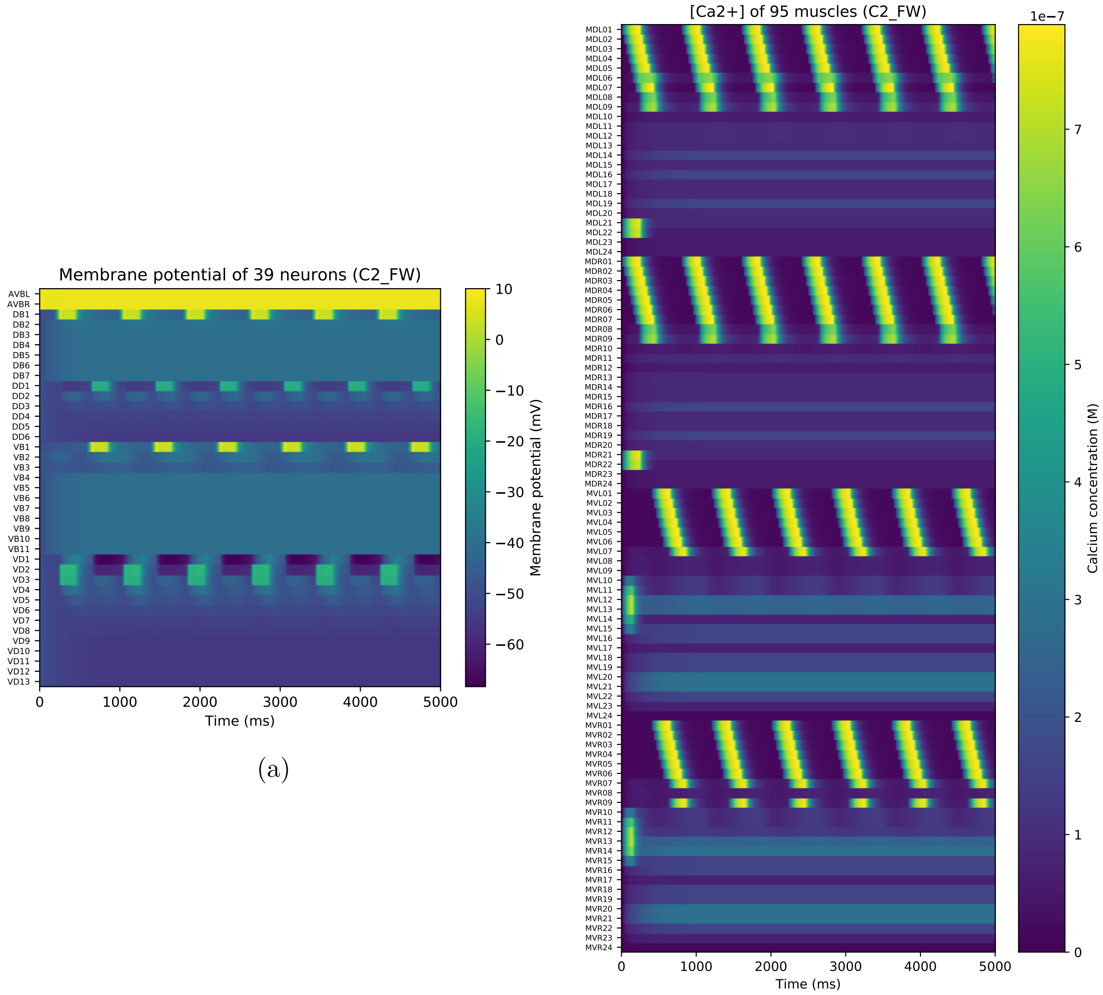

Membrane potential of 39 neurons (C2_FW)

[Ca2+] of 95 muscles (C2_FW)

(a)

(b)

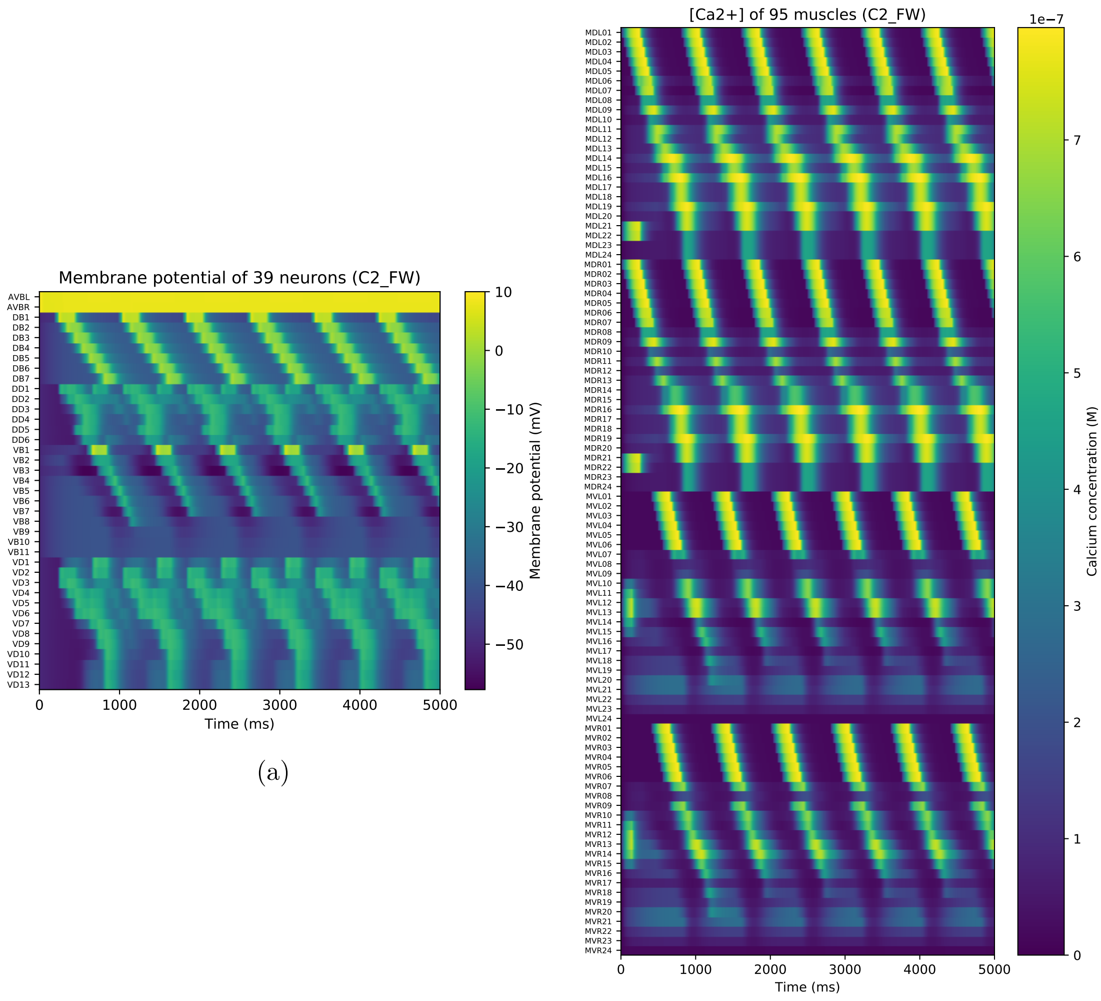
Figure 4.9: Simulation of the worm's neural circuit and muscles, without input injected into DB1 and VB1, during 5 seconds of real time simulation of the forward locomotion using c302. (a) Neuron membrane potential dynamics. (b) Internal calcium concentration of the 95 body-wall muscle cells.



(a)

(b)

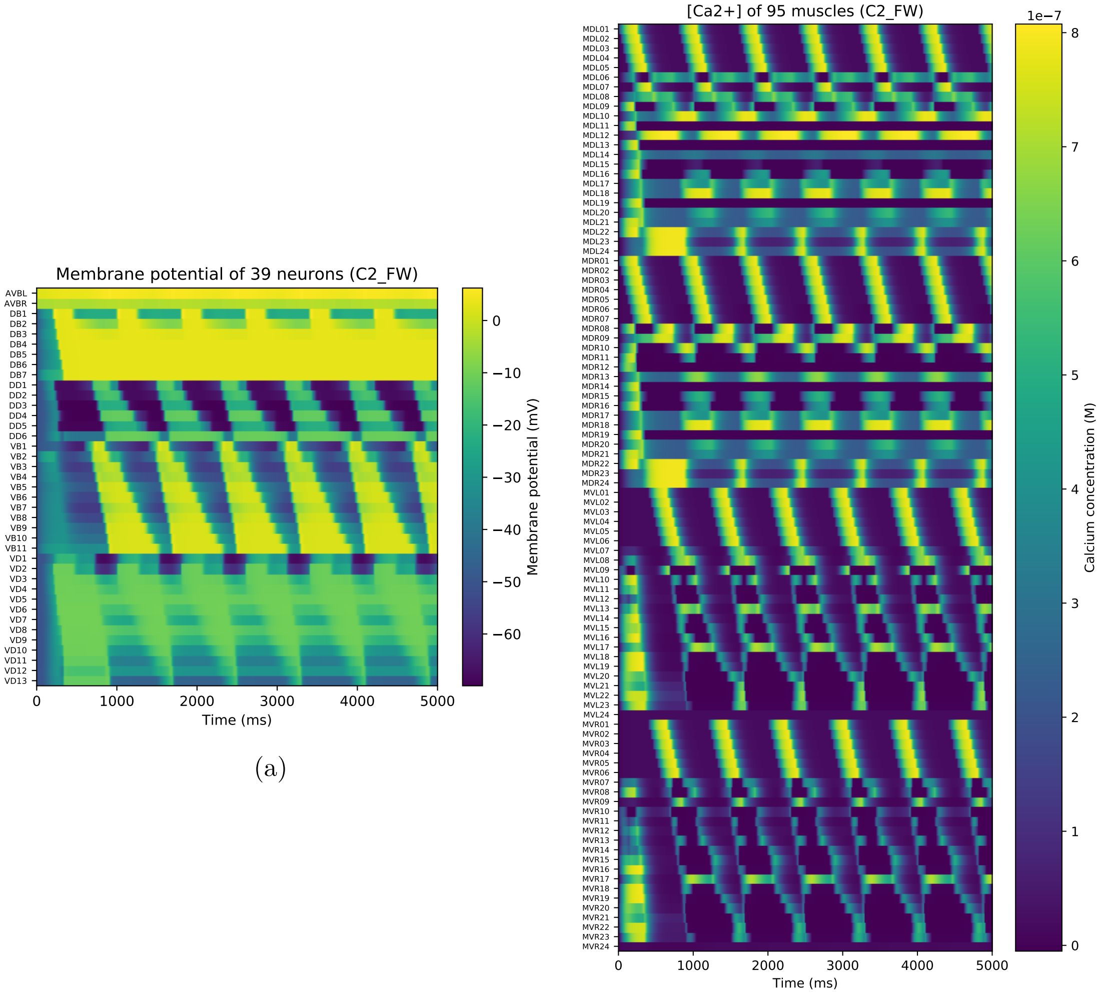
Figure 4.10: Simulation of the worm's neural circuit and muscles, without chemical synapses connecting neighboring B-type neurons, during 5 seconds of real time simulation of the forward locomotion using c302. (a) Neuron membrane potential dynamics. (b) Internal calcium concentration of the 95 body-wall muscle cells.



(a)

(b)

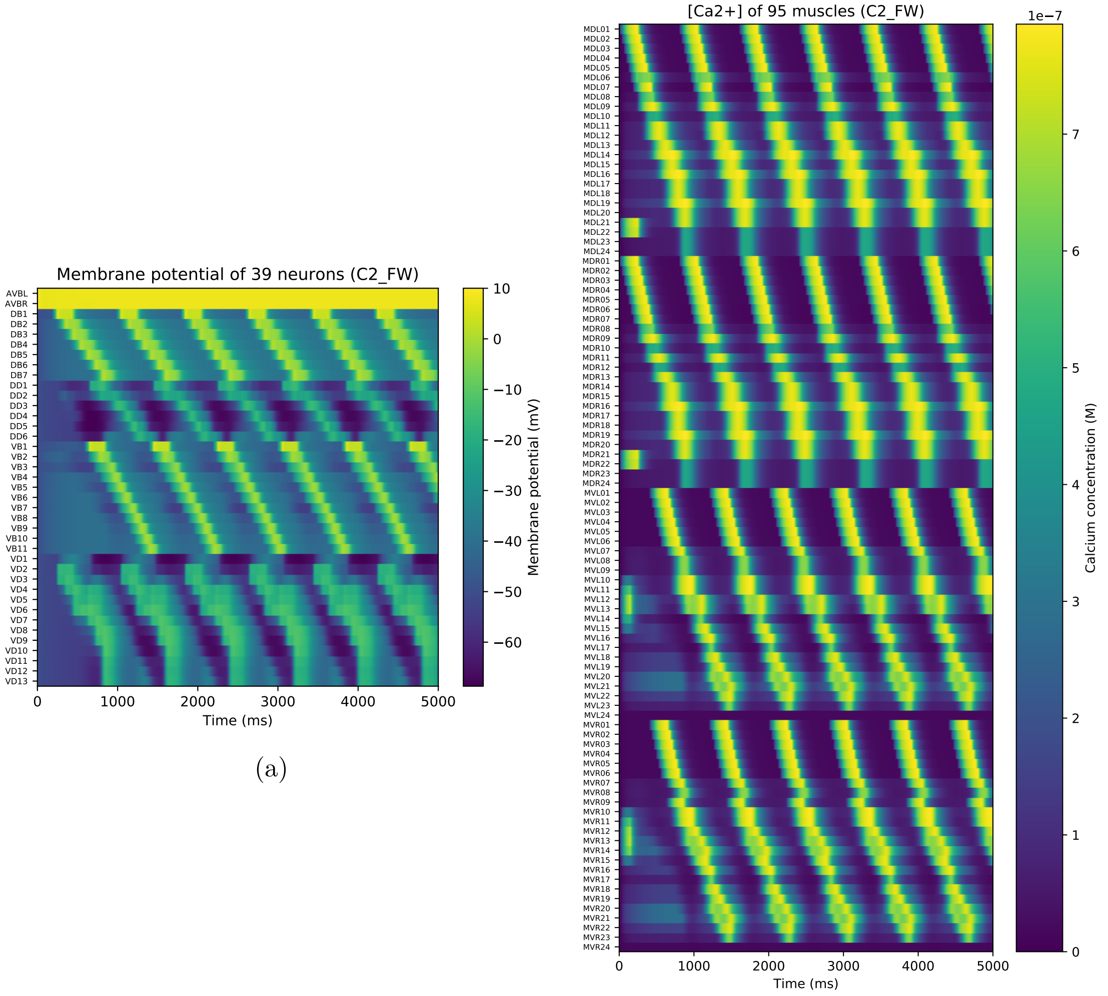
Figure 4.11: Simulation of the worm's neural circuit and muscles, with excitatory synapses from DB to DD and from VB to VD neurons, during 5 seconds of real time simulation of the forward locomotion using c302. (a) Neuron membrane potential dynamics. (b) Internal calcium concentration of the 95 body-wall muscle cells.



(a)

(b)

Figure 4.12: Simulation of the worm's neural circuit and muscles, with weighted synapses, during 5 seconds of real time simulation of the forward locomotion using c302. (a) Neuron membrane potential dynamics. (b) Internal calcium concentration of the 95 body-wall muscle cells.



(a)

(b)

Figure 4.13: Simulation of the worm's neural circuit and muscles during 5 seconds of real time simulation of the forward locomotion using c302. (a) Neuron membrane potential dynamics. (b) Internal calcium concentration of the 95 body-wall muscle cells.

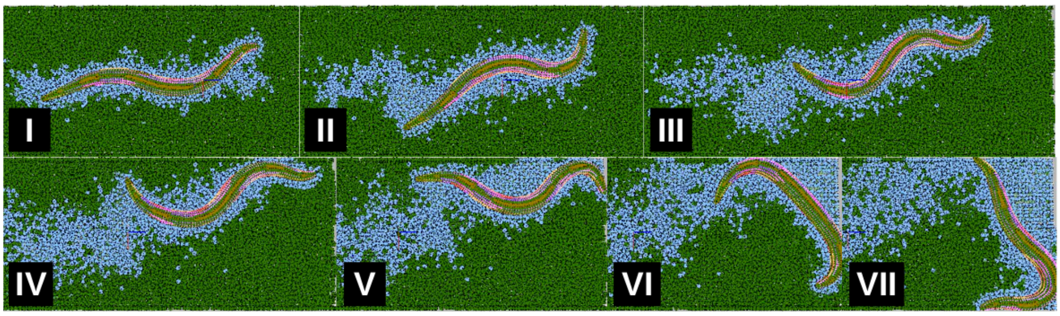


Figure 4.14: Simulation of the worm's crawling in Sibernetic worm simulator. Seven time-elapses in the 5-second forward crawling is represented from I to VII. The pink colored regions of the worm's body indicate contracted body wall muscles.

CHAPTER 5

Conclusion

5.1 Summary/Discussion

In this thesis, we improved the process of generating neural network models, using c302, one of the subprojects of OpenWorm, by adding different parameters, to make the definition of components and properties of the network more customizable.

We modified the parameters of the preexisting single-compartmental conductance-based neuron model, so it generates graded membrane potentials instead of burst firing. Although the modified neuron model was able to generate graded membrane potentials, the output of the simulated internal calcium concentration was not as desired because the internal calcium concentration decreases after increasing the input current, as shown in figure 5.1. The output of the simulation was generated, by injecting input pulses into a neuron with increasing amplitude, from 1 pA to 6 pA.

We run experiments while making use of a genetic algorithm, for automatic optimization of the activation time of motor neurons and the spike frequency of muscle cells. Due to the not-desirable result of the first experiment, we reduced the number of parameters, in subsequent experiments. This could be achieved, by sequentially optimizing parameters of different synapses.

The predominant part of this thesis is the design and implementation of a new single-compartmental conductance-based neuron circuit model responsible for generating phase-shifted activities of dorsal and ventral body-wall muscles, propagating from neck to tail, resulting into forward-crawling locomotion of *C. elegans*. We implemented our new model within the OpenWorm platform to induce forward crawling of the worm while making use of the output of our neural circuit model, instead of using a simple sine-wave generator, as before.

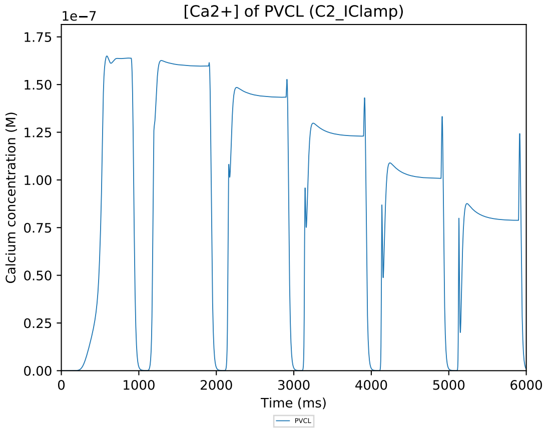


Figure 5.1: Internal calcium concentration of a single neuron, using our modified parameter set, after injecting 6 current pulses with increasing amplitude, from 1 pA to 6 pA, and a duration of 800 ms. It shows a decreased calcium concentration using an input current with increased amplitude.

To make it easier for users to design and run their own in silico experiments within the OpenWorm platform, we created a docker container integrating different OpenWorm projects, and their dependencies.

5.2 Future work

The neuron and muscle model we used has still some flaws. While it is able to generate graded membrane potentials, as desired, the amplitude of the internal calcium concentration reduces after increasing the amplitude of the input current. Thus, one of the next steps would be to change this behavior. This could be achieved, by modifying the parameters of the neuron model.

Muscle cells also generate graded membrane potentials, with our model. There is evidence that they rather generate all-or-none action potentials [LGC⁺11, GZ11]. Thus, we can implement a more realistic behaving muscle cell, based on the recordings of the muscles and ion channels.

Within the OpenWorm platform, all neurons share the same computational model. Based on electrophysiological experiments, we know that the activity of neurons within the nervous system of *C. elegans* show different dynamics. Thus, we can apply changes to c302, to enable the user to define different neuron models.

Gap junctions are modeled as bidirectional connections, within the OpenWorm platform. There is evidence that antidromic-rectifying gap junctions, i.e., current flow only in one direction, away from axon terminals to the soma, exist within the nervous system of *C. elegans*. Besides the rectifying aspect, it has been shown that they act as an amplifier of chemical synapses connecting the same pair of neurons as the gap junctions [LCMW17]. Although the recordings were made for neurons known to be part of a circuit responsible for inducing backward locomotion of the worm, rectifying gap junctions could also play an important role for proper functioning of the forward locomotion circuit. To analyze the importance of such gap junctions within our new forward crawling neuron circuit model, we can implement unidirectional gap junctions.

To generate a more biologically realistic model, for generating forward crawling of the worm, the next future steps are the relaxation of the assumptions that we incorporated during the design of our neural circuit model. AVB command neurons get activated with a directly injected pulse. In real worms, they receive input coming from multiple sensory neurons and other interneurons. Thus, we can include these neurons in our model. Instead of injecting current directly into muscle cells located in the head, we can include another circuit which is able to generate oscillating activity of those muscles. The periodic input into the first B-type motor neurons, DB1 and VB1, can be replaced by an oscillating network, or intrinsically oscillating motor neurons. The chemical synapses, for approximating the proprioceptive effect, can be removed, after the state of muscles, as calculated by Sibernetic, can be fed back to c302. To get forward crawling behavior without changing the synaptic weights, we can change the conductances of different synapses.

As another future work, we can implement a neuron circuit model responsible for generating backward crawling behavior of *C. elegans*. We can incorporate similar assumptions as used while we designed the forward crawling neuron circuit model. After we can generate forward and backward crawling behavior, we can combine the circuits and include input coming from sensory neurons, to simulate the tap-withdrawal reflex.

APPENDIX A

Supplementary material

A.1 Network Configuration File - c302_FW.py

```python
import sys
import os

sys.path.insert(0, os.path.abspath(’.’))

import c302

import neuroml.writers as writers

range_incl = lambda start, end:range(start, end + 1)


def setup(parameter_set,
          generate=False,
          duration=2000,
          dt=0.05,
          target_directory=’examples’,
          data_reader="UpdatedSpreadsheetDataReader2",
          param_overrides={},
          verbose=True,
          config_param_overrides={}):

    exec (’from parameters_%s import ParameterisedModel’ % parameter_set, globals())
    params = ParameterisedModel()

    params.set_bioparameter("unphysiological_offset_current", "0pA", "Testing TapWithdrawal", "0")
    params.set_bioparameter("unphysiological_offset_current_del", "0 ms", "Testing TapWithdrawal", "0")
    params.set_bioparameter("unphysiological_offset_current_dur", "2000 ms", "Testing TapWithdrawal", "0")

    VB_motors = ["VB%s" % c for c in range_incl(1, 11)]
    DB_motors = ["DB%s" % c for c in range_incl(1, 7)]
    DD_motors = ["DD%s" % c for c in range_incl(1, 6)]
    VD_motors = ["VD%s" % c for c in range_incl(1, 13)]

    cells = list(['AVBL', 'AVBR'] + DB_motors + VD_motors + VB_motors +
    DD_motors)

    muscles_to_include = True

    cells_to_stimulate = []

    cells_to_plot = list(cells)
    reference = "c302_%s_FW" % parameter_set


    conns_to_include = []
    conns_to_exclude = [
        'VB2-VB4_GJ',
        'VB4-VB2_GJ',
    ]
    conn_polarity_override = {
        r'^DB\d+-DD\d+$': 'inh',
        r'^VB\d+-VD\d+$': 'inh',
    }
    conn_number_override = {
        '^.+-.+$': 1,
    }

    input_list = []

    input_list.append(('MVR10', '0ms', '150ms', '1pA'))
    input_list.append(('MVR11', '0ms', '150ms', '2pA'))
    input_list.append(('MVR12', '0ms', '150ms', '3pA'))
    input_list.append(('MVR13', '0ms', '150ms', '3pA'))
    input_list.append(('MVR14', '0ms', '150ms', '2pA'))
    input_list.append(('MVR15', '0ms', '150ms', '1pA'))

    input_list.append(('MVL10', '0ms', '150ms', '1pA'))
    input_list.append(('MVL11', '0ms', '150ms', '2pA'))
    input_list.append(('MVL12', '0ms', '150ms', '3pA'))
    input_list.append(('MVL13', '0ms', '150ms', '3pA'))
    input_list.append(('MVL14', '0ms', '150ms', '2pA'))
    input_list.append(('MVL15', '0ms', '150ms', '1pA'))

    input_list.append(('MDL21', '0ms', '250ms', '3pA'))
    input_list.append(('MDL22', '0ms', '250ms', '3pA'))
    input_list.append(('MDR21', '0ms', '250ms', '3pA'))
    input_list.append(('MDR22', '0ms', '250ms', '3pA'))


    amp = '4pA'
    dur = '250ms'

    for stim_num in range(15):
        for muscle_num in range(7):
            mdlx = 'MDL0%s' % (muscle_num + 1)
            mdrx = 'MDR0%s' % (muscle_num + 1)
            mvlx = 'MVL0%s' % (muscle_num + 1)
            mvrx = 'MVR0%s' % (muscle_num + 1)

            if muscle_num >= 9:
                mdlx = 'MDL%s' % (muscle_num + 1)
                mdrx = 'MDR%s' % (muscle_num + 1)
                mvlx = 'MVL%s' % (muscle_num + 1)
                mvrx = 'MVR%s' % (muscle_num + 1)

            startd = '%sms' % (stim_num * 800 + muscle_num * 30)
            startv = '%sms' % ((stim_num * 800 + 400) + muscle_num * 30)

            input_list.append((mdlx, startd, dur, amp))
            input_list.append((mdrx, startd, dur, amp))
            if muscle_num != 6:
                input_list.append((mvlx, startv, dur, amp))
                input_list.append((mvrx, startv, dur, amp))

    d_v_delay = 400

    start = 190
    motor_dur = '250ms'

    input_list.append(('AVBL', '0ms', '%sms'%duration, '15pA'))
    input_list.append(('AVBR', '0ms', '%sms'%duration, '15pA'))
    input_list.append(('DB1', '%sms'%(start), motor_dur, '3pA'))
    input_list.append(('VB1', '%sms'%(start+d_v_delay), motor_dur, '3pA'))

    i = start + 2 * d_v_delay
    j = start + 3 * d_v_delay
    for pulse_num in range(1,15):
        input_list.append(('DB1', '%sms'%i, motor_dur, '3pA'))
        input_list.append(('VB1', '%sms'%j, motor_dur, '3pA'))
        i += d_v_delay * 2
        j += d_v_delay * 2

    config_param_overrides['input'] = input_list

    param_overrides = {
        'mirrored_elec_conn_params': {

            r'^AVB._to_DB\d+\_GJ$_elec_syn_gbase': '0.001 nS',
            r'^AVB._to_VB\d+\_GJ$_elec_syn_gbase': '0.001 nS',

            r'^DB\d+_to_DB\d+\_GJ$_elec_syn_gbase': '0.001 nS',

            r'^VB\d+_to_VB\d+\_GJ$_elec_syn_gbase': '0.001 nS',

            r'^DB\d+_to_VB\d+\_GJ$_elec_syn_gbase': '0 nS',
            r'^DB\d+_to_DD\d+\_GJ$_elec_syn_gbase': '0 nS',
            r'^VB\d+_to_VD\d+\_GJ$_elec_syn_gbase': '0 nS',

            'DD1_to_MVL08_elec_syn_gbase': '0 nS',
            'VD2_to_MDL09_elec_syn_gbase': '0 nS',
        },

        r'^VB\d+_to_VB\d+$_exc_syn_conductance': '18 nS',
        r'^VB\d+_to_VB\d+$_exc_syn_ar': '0.19 per_s',
        r'^VB\d+_to_VB\d+$_exc_syn_ad': '73 per_s',
        r'^VB\d+_to_VB\d+$_exc_syn_beta': '2.81 per_mV',
        r'^VB\d+_to_VB\d+$_exc_syn_vth': '-22 mV',
        r'^VB\d+_to_VB\d+$_exc_syn_erev': '10 mV',

        r'^DB\d+_to_DB\d+$_exc_syn_conductance': '20 nS',
        r'^DB\d+_to_DB\d+$_exc_syn_ar': '0.08 per_s',
        r'^DB\d+_to_DB\d+$_exc_syn_ad': '18 per_s',
        r'^DB\d+_to_DB\d+$_exc_syn_beta': '0.21 per_mV',
        r'^DB\d+_to_DB\d+$_exc_syn_vth': '-10 mV',
        r'^DB\d+_to_DB\d+$_exc_syn_erev': '10 mV',

        'initial_memb_pot': '-50 mV',

        'AVBR_to_DB4_exc_syn_conductance': '0 nS',

        'AVBL_to_VB2_exc_syn_conductance': '0 nS',

        'AVBR_to_VD3_exc_syn_conductance': '0 nS',

        'DD1_to_VB2_inh_syn_conductance': '0 nS',

        'neuron_to_muscle_exc_syn_conductance': '0.5 nS',
        r'^DB\d+_to_MDL\d+$_exc_syn_conductance': '0.4 nS',
        r'^DB\d+_to_MDR\d+$_exc_syn_conductance': '0.4 nS',
        r'^VB\d+_to_MVL\d+$_exc_syn_conductance': '0.6 nS',
        r'^VB\d+_to_MVR\d+$_exc_syn_conductance': '0.6 nS',
        'neuron_to_muscle_exc_syn_vth': '37 mV',
        'neuron_to_muscle_inh_syn_conductance': '0.6 nS',
        'neuron_to_neuron_inh_syn_conductance': '0.2 nS',

        'AVBR_to_MVL16_exc_syn_conductance': '0 nS',
        'ca_conc_decay_time_muscle': '60.8 ms',
        'ca_conc_rho_muscle': '0.002338919 mol_per_m_per_A_per_s',

    }


    nml_doc = None
    if generate:
        nml_doc = c302.generate(reference,
                                params,
                                cells=cells,
                                cells_to_plot=cells_to_plot,
                                cells_to_stimulate=cells_to_stimulate,
                                conns_to_include=conns_to_include,
                                conns_to_exclude=conns_to_exclude,
                                conn_polarity_override=conn_polarity_override
    ,
                                conn_number_override=conn_number_override,
                                muscles_to_include=muscles_to_include,
                                duration=duration,
                                dt=dt,
                                target_directory=target_directory,
                                data_reader=data_reader,
                                param_overrides=param_overrides,
                                verbose=verbose)

        for stim_input in input_list:
            cell, start, dur, current = stim_input
            c302.add_new_input(nml_doc, cell, start, dur, current, params)

        nml_file = target_directory + '/' + reference + '.net.nml'
        writers.NeuroMLWriter.write(nml_doc, nml_file)  # Write over network
    file written above...

        print("(Re)written network file to: " + nml_file)


    return cells, cells_to_stimulate, params, muscles_to_include, nml_doc


if __name__ == '__main__':
    parameter_set = sys.argv[1] if len(sys.argv) == 2 else 'C2'
    data_reader = sys.argv[2] if len(sys.argv) == 3 else '
    UpdatedSpreadsheetDataReader2'

    setup(parameter_set, generate=True, data_reader=data_reader)
```

Listing A.1: c302_FW.py

List of Figures

1.1 C. elegans Turing-test 2

3.1 New forward crawling neural circuit 20
3.2 Overview of the simulation stack of OpenWorm 22
3.3 Anatomical and biophysical level of details of network simulations using c302 23
3.4 Overview of the c302 framework 25

4.8 Result of the simulation of the forward crawling neural circuit without input injected into AVB using c302 50
4.9 Result of the simulation of the forward crawling neural circuit without input going into DB1 and VB1 using c302 51
4.10 Result of the simulation of the forward crawling neural circuit without our additional chemical synapses between B-type motor neurons using c302 . 52
4.11 Result of the simulation of the forward crawling neural circuit without B-type to D-type inhibition using c302 53
4.12 Result of the simulation of the forward crawling neural circuit with different weighted synapses using c302 54
4.13 Result of the simulation of the forward crawling neural circuit using c302 55
4.14 Result of the simulation of the worm's body, driven by the activity of the forward crawling neural circuit, using Sibernetic 56

List of Tables

4.1 Neuron model parameters 35
4.2 Parameter ranges for the optimization (step 1) 39
4.3 Targets of the optimization (step 1) 39
4.4 Result of the optimization (step 1) 39
4.5 AVB-DB-DorsalMuscles optimization 1 40
4.6 Parameter ranges for the optimization (step 2) 41
4.7 Targets of the optimization (step 2) 41
4.8 Result of the optimization (step 2) 41
4.9 AVB-DB-DorsalMuscles optimization 2 42
4.10 Parameter ranges for the optimization (step 3) 43
4.11 Targets of the optimization (step 3) 43
4.12 Result of the optimization (step 3) 43
4.13 AVB-DB-DorsalMuscles optimization 3 44
4.14 7-31 results. The rows show the result of different experiments after the hand-tuning of some parameters. 46
4.15 8-05 results 47

Bibliography

[Abb99] Larry F Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain research bulletin*, 50(5-6):303–304, 1999.

[ACG⁺09] Frederico AC Azevedo, Ludmila RB Carvalho, Lea T Grinberg, José Marcelo Farfel, Renata EL Ferretti, Renata EP Leite, Wilson Jacob Filho, Roberto Lent, and Suzana Herculano-Houzel. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541, 2009.

[AH02] ZF Altun and DH Hall. Wormatlas. *URL http://www.wormatlas.org*, 1384, 2002.

[App14] Peter A Appleby. The role of multiple chemotactic mechanisms in a model of chemotaxis in c. elegans: different mechanisms are specialised for different environments. *Journal of computational neuroscience*, 36(3):339–354, 2014.

[AY12] Leon Avery and Young-Jai You. C. elegans feeding. *WormBook: the online review of C. elegans biology*, pages 1–23, 2012.

[Bar98] Cornelia I Bargmann. Neurobiology of the caenorhabditis elegans genome. *Science*, 282(5396):2028–2033, 1998.

[BBB⁺16] Barry Bentley, Robyn Branicky, Christopher L Barnes, Yee Lian Chew, Eviatar Yemini, Edward T Bullmore, Petra E Vértes, and William R Schafer. The multilayer connectome of caenorhabditis elegans. *PLoS computational biology*, 12(12):e1005283, 2016.

[BC04] John Bryden and Netta Cohen. A simulation model of the locomotion controllers for the nematode caenorhabditis elegans. 2004.

[BC08a] Jordan H Boyle and Netta Cohen. Caenorhabditis elegans body wall muscles are simple actuators. *Biosystems*, 94(1-2):170–181, 2008.

[BC08b] John Bryden and Netta Cohen. Neural control of caenorhabditis elegans forward locomotion: the role of sensory feedback. *Biological Cybernetics*, 98(4):339–351, Apr 2008.

[Boe15] Carl Boettiger. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, 2015.

[Boy09] Jordan Hylke Boyle. *C. elegans locomotion: an integrated approach.* PhD thesis, University of Leeds, 2009.

[Bry03] John Bryden. A simulation model of the locomotion controllers for the nematode caenorhabditis elegans. 2003.

[CC14] Xiaoyin Chen and Martin Chalfie. Modulation of c. elegans touch sensitivity is integrated at multiple levels. *Journal of Neuroscience*, 34(19):6522–6536, 2014.

[CGC⁺14] Robert C Cannon, Padraig Gleeson, Sharon Crook, Gautham Ganapathy, Boris Marin, Eugenio Piasini, and R Angus Silver. Lems: a language for expressing complex biological models in concise and hierarchical form and its use in underpinning neuroml 2. *Frontiers in neuroinformatics*, 8:79, 2014.

[CH06] Nicholas T Carnevale and Michael L Hines. *The NEURON Book.* Cambridge University Press, 2006.

[COL⁺18] Yongmin Cho, David N Oakland, Sol Ah Lee, William R Schafer, and Hang Lu. On-chip functional neuroimaging with mechanical stimulation in caenorhabditis elegans larvae for studying development and neural circuits. *Lab on a Chip*, 18(4):601–609, 2018.

[CWC15] Ann K Corsi, Bruce Wightman, and Martin Chalfie. A transparent window into biology: a primer on caenorhabditis elegans. *Genetics*, 200(2):387–407, 2015.

[EN90] Paul Erdös and Ernst Niebur. The neural basis of the locomotion of nematodes. In *Statistical Mechanics of Neural Networks*, pages 253–267. Springer, 1990.

[eSC98] The C. elegans Sequencing Consortium. Genome sequence of the nematode c. elegans: a platform for investigating biology. *Science*, pages 2012–2018, 1998.

[FF15] Lise Frézal and M B Félix. C. elegans outside the petri dish. In *eLife*, 2015.

[FYWX+10] Christopher Fang-Yen, Matthieu Wyart, Julie Xie, Risa Kawai, Tom Kodger, Sway Chen, Quan Wen, and Aravinthan D. T. Samuel. Biomechanical analysis of gait adaptation in the nematode caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, 107(47):20323–20328, 2010.

[GAH16] Marie Gendrel, Emily G Atlas, and Oliver Hobert. A cellular and regulatory map of the gabaergic nervous system of c. elegans. *Elife*, 5:e17686, 2016.

[GB08] Dan FM Goodman and Romain Brette. Brian: a simulator for spiking neural networks in python. *Frontiers in neuroinformatics*, 2:5, 2008.

[GB09] Dan FM Goodman and Romain Brette. The brian simulator. *Frontiers in neuroscience*, 3:26, 2009.

[GCC+10] Padraig Gleeson, Sharon Crook, Robert C Cannon, Michael L Hines, Guy O Billings, Matteo Farinella, Thomas M Morse, Andrew P Davison, Subhasis Ray, Upinder S Bhalla, et al. Neuroml: a language for describing data driven models of neurons and networks with a high degree of biological detail. *PLoS computational biology*, 6(6):e1000815, 2010.

[GCC+15] Padraig Gleeson, Matteo Cantarelli, Michael Currie, Jim Hokanson, Giovanni Idili, Sergey Khayrulin, Andrey Palyanov, Balazs Szigeti, and Stephen Larson. The openworm project: currently available resources and future plans. *BMC neuroscience*, 16(1):P141, 2015.

[GD07] Marc-Oliver Gewaltig and Markus Diesmann. Nest (neural simulation tool). *Scholarpedia*, 2(4):1430, 2007.

[GHAL98] Miriam B. Goodman, David H. Hall, Leon Avery, and Shawn R. Lockery. Active currents regulate sensitivity and dynamic range in c. elegans neurons. *Neuron*, 20(4):763–772, Apr 1998. 9581767[pmid].

[GKL+04] Jesse M Gray, David S Karow, Hang Lu, Andy J Chang, Jennifer S Chang, Ronald E Ellis, Michael A Marletta, and Cornelia I Bargmann. Oxygen sensation and social feeding mediated by a c. elegans guanylate cyclase homologue. *Nature*, 430(6997):317, 2004.

[GLG+18] Padraig Gleeson, David Lung, Radu Grosu, Ramin Hasani, and Stephen D. Larson. c302: a multiscale framework for modelling the nervous system of caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 373(1758), 2018.

[GZ11] Shangbang Gao and Mei Zhen. Action potentials drive body wall muscle contractions in caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, page 201012346, 2011.

[Hal17] David H Hall. Gap junctions in c. elegans: Their roles in behavior and development. *Developmental neurobiology*, 77(5):587–596, 2017.

[HBF$^+$17] Ramin M Hasani, Victoria Beneder, Magdalena Fuchs, David Lung, and Radu Grosu. Sim-ce: An advanced simulink platform for studying the brain of caenorhabditis elegans. *arXiv preprint arXiv:1703.06270*, 2017.

[HR75] Edward M Hedgecock and Richard L Russell. Normal and mutant thermotaxis in the nematode caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, 72(10):4061–4065, 1975.

[Izh03] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[Izh04] Eugene M Izhikevich. Which model to use for cortical spiking neurons? *IEEE transactions on neural networks*, 15(5):1063–1070, 2004.

[KCG$^+$07] Jeremie Korta, Damon A. Clark, Christopher V. Gabel, L. Mahadevan, and Aravinthan D. T. Samuel. Mechanosensation and mechanical load modulate the locomotory gait of swimming c. elegans. *Journal of Experimental Biology*, 210(13):2383–2389, 2007.

[KD17] Masahiro Kuramochi and Motomichi Doi. A computational model based on multi-regional calcium imaging represents the spatio-temporal dynamics in a caenorhabditis elegans sensory neuron. *PloS one*, 12(1):e0168415, 2017.

[KH93] Joshua M Kaplan and H ROBERT HoRVITZ. A dual mechanosensory and chemosensory neuron in caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, 90(6):2227–2231, 1993.

[KKS$^+$15] Saul Kato, Harris?S Kaplan, Tina Schrödel, Susanne Skora, Theodore?H Lindsay, Eviatar Yemini, Shawn Lockery, and Manuel Zimmer. Global brain dynamics embed the motor command sequence of <em>caenorhabditis elegans</em>. *Cell*, 163(3):656–669, Oct 2015.

[KMK17] James M Kunert, Pedro D Maia, and J Nathan Kutz. Functionality and robustness of injured connectomic dynamics in c. elegans: linking behavioral deficits to neural circuit damage. *PLoS computational biology*, 13(1):e1005261, 2017.

[LCMW17] Ping Liu, Bojun Chen, Roger Mailler, and Zhao-Wen Wang. Antidromic-rectifying gap junctions amplify chemical transmission at functionally mixed electrical-chemical synapses. *Nature communications*, 8:14818, 2017.

[LGC$^+$11] Ping Liu, Qian Ge, Bojun Chen, Lawrence Salkoff, Michael I Kotlikoff, and Zhao-Wen Wang. Genetic dissection of ion currents underlying all-or-none action potentials in c. elegans body-wall muscle cells. *The Journal of physiology*, 589(1):101–117, 2011.

[LGF09] Shawn R Lockery, Miriam B Goodman, and Serge Faumont. First report of action potentials in a c. elegans neuron is premature. *Nature neuroscience*, 12(4):365, 2009.

[MPS10] Karen Mesce and Jonathan Pierce-Shimomura. Shared strategies for behavioral switching: Understanding how locomotor patterns are turned on and off. *Frontiers in Behavioral Neuroscience*, 4:49, 2010.

[NE91] Ernst Niebur and Paul Erdös. Theory of the locomotion of nematodes: dynamics of undulatory progression on a surface. *Biophysical journal*, 60(5):1132–1146, 1991.

[NE93] Ernst Niebur and Paul Erdös. Theory of the locomotion of nematodes: control of the somatic motor neurons by interneurons. *Mathematical biosciences*, 118(1):51–82, 1993.

[PKL16] Andrey Palyanov, Sergey Khayrulin, and Stephen D Larson. Application of smoothed particle hydrodynamics to modeling mechanisms of biological tissue. *Advances in Engineering Software*, 98:1–11, 2016.

[PKL18] Andrey Palyanov, Sergey Khayrulin, and Stephen D. Larson. Three-dimensional simulation of the caenorhabditis elegans body and muscle cells in liquid and gel environments for behavioural analysis. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 373(1758), 2018.

[PKSS^{+}15] Laura Pereira, Paschalis Kratsios, Esther Serrano-Saiz, Hila Sheftel, Avi E Mayo, David H Hall, John G White, Brigitte LeBoeuf, L Rene Garcia, Uri Alon, et al. A cellular and regulatory map of the cholinergic nervous system of c. elegans. *Elife*, 4:e12432, 2015.

[PP16] Lipika Parida and Venkat Padmanabhan. Durotaxis in nematode caenorhabditis elegans. *Biophysical Journal*, 111(3):666–674, 2016.

[PSV^{+}17] Alexander Peyser, Ankur Sinha, Stine Brekke Vennemo, Tammo Ippen, Jakob Jordan, Steffen Graber, Abigail Morrison, Guido Trensch, Tanguy Fardet, Håkon Mørk, Jan Hahne, Jannis Schuecker, Maximilian Schmidt, Susanne Kunkel, David Dahmen, Jochen Martin Eppler, Sandra Diaz, Dennis Terhorst, Rajalekshmi Deepu, Philipp Weidel, Itaru Kitayama, Sepehr Mahmoudian, David Kappel, Martin Schulze, Shailesh Appukuttan, Till Schumann, Hünkar Can Tunç, Jessica Mitchell, Michael Hoff, Eric Müller, Milena Menezes Carvalho, Barna Zajzon, and Hans Ekkehard Plesser. Nest 2.14.0, October 2017.

[RAL+16] William M Roberts, Steven B Augustine, Kristy J Lawton, Theodore H Lindsay, Tod R Thiele, Eduardo J Izquierdo, Serge Faumont, Rebecca A Lindsay, Matthew Cale Britton, Navin Pokala, et al. A stochastic neuronal model predicts random search behaviors at multiple spatial scales in c. elegans. *Elife*, 5:e12572, 2016.

[RBA17] Babak Bashari Rad, Harrison John Bhatti, and Mohammad Ahmadi. An introduction to docker and analysis of its performance. *International Journal of Computer Science and Network Security (IJCSNS)*, 17(3):228, 2017.

[RBC90] Catherine H Rankin, Christine DO Beck, and Catherine M Chiba. Caenorhabditis elegans: a new model system for the study of learning and memory. *Behavioural brain research*, 37(1):89–92, 1990.

[RBMP97] Donald L Riddle, Thomas Blumenthal, Barbara J Meyer, and James R Priess. C. elegans ii. 2nd edition. chapter 12. Cold Spring Harbor Laboratory Press, 1997.

[RSSK13] Franciszek Rakowski, Jagan Srinivasan, Paul W Sternberg, and Jan Karbowski. Synaptic polarity of the interneuron circuit controlling c. elegans locomotion. *Frontiers in computational neuroscience*, 7:128, 2013.

[RW08] Mauno Rönkkö and Garry Wong. Modeling the c. elegans nematode and its environment using a particle system. *Journal of theoretical biology*, 253(2):316–322, 2008.

[SGBB14] Marcel Stimberg, Dan FM Goodman, Victor Benichoux, and Romain Brette. Equation-oriented specification of neural models for simulations. *Frontiers in neuroinformatics*, 8:6, 2014.

[SGV+14] Balázs Szigeti, Padraig Gleeson, Michael Vella, Sergey Khayrulin, Andrey Palyanov, Jim Hokanson, Michael Currie, Matteo Cantarelli, Giovanni Idili, and Stephen Larson. Openworm: an open-science approach to modeling caenorhabditis elegans. *Frontiers in computational neuroscience*, 8:137, 2014.

[SKB+03] Hiroshi Suzuki, Rex Kerr, Laura Bianchi, Christian Frøkjær-Jensen, Dan Slone, Jian Xue, Beate Gerstbrein, Monica Driscoll, and William R Schafer. In vivo imaging of c. elegans mechanosensory neurons demonstrates a specific role for the mec-4 channel in the process of gentle touch sensation. *Neuron*, 39(6):1005–1017, 2003.

[SLP+18] Gopal P. Sarma, Chee Wai Lee, Tom Portegys, Vahid Ghayoomie, Travis Jacobs, Bradly Alicea, Matteo Cantarelli, Michael Currie, Richard C. Gerkin, Shane Gingell, Padraig Gleeson, Richard Gordon, Ramin M. Hasani, Giovanni Idili, Sergey Khayrulin, David Lung, Andrey Palyanov, Mark Watts, and Stephen D. Larson. Openworm: overview and recent advances in integrative biological simulation of caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 373(1758), 2018.

[SMN14] Karina Simonsen, Donald Moerman, and Christian C Naus. Gap junctions in c. elegans. *Frontiers in physiology*, 5:40, 2014.

[SP09] Barbara Solenthaler and Renato Pajarola. Predictive-corrective incompressible sph. In *ACM transactions on graphics (TOG)*, volume 28, page 40. ACM, 2009.

[SSK+12] Xiao N Shen, J Sznitman, P Krajacic, T Lamitina, and PE Arratia. Undulatory locomotion of caenorhabditis elegans on wet surfaces. *Biophysical journal*, 102(12):2772–2781, 2012.

[SSWT83] J.E. Sulston, E. Schierenberg, J.G. White, and J.N. Thomson. The embryonic cell lineage of the nematode caenorhabditis elegans. *Developmental Biology*, 100(1):64 – 119, 1983.

[STO05] Michiyo Suzuki, Toshio Tsuji, and Hisao Ohtake. A model of motor control of the nematode c. elegans with neuronal circuits. *Artificial Intelligence in Medicine*, 35(1-2):75–86, 2005.

[THC11] ED Tytell, P Holmes, and AH Cohen. Spikes alone do not behavior make: why neuroscience needs biomechanics. *Current opinion in neurobiology*, 21(5):816–822, 2011.

[TUR50] A. M. TURING. I.—computing machinery and intelligence. *Mind*, LIX(236):433–460, 1950.

[VCP+11] Lav R Varshney, Beth L Chen, Eric Paniagua, David H Hall, and Dmitri B Chklovskii. Structural properties of the Caenorhabditis elegans neuronal network. *PLoS computational biology*, 7(2):e1001066, 2011.

[VGDBPS12] Andrés G Vidal-Gadea, Scott Davis, Lindsay Becker, and Jonathan T Pierce-Shimomura. Coordination of behavioral hierarchies during environmental transitions in caenorhabditis elegans. In *Worm*, volume 1, pages 5–11. Taylor & Francis, 2012.

[VGTY+11] Andrés Vidal-Gadea, Stephen Topper, Layla Young, Ashley Crisp, Leah Kressin, Erin Elbel, Thomas Maples, Martin Brauner, Karen Erbguth, Abram Axelrod, Alexander Gottschalk, Dionicio Siegel, and Jonathan T. Pierce-Shimomura. Caenorhabditis elegans selects distinct crawling and swimming gaits via dopamine and serotonin. *Proceedings of the National Academy of Sciences*, 108(42):17504–17509, 2011.

[War73] Samuel Ward. Chemotaxis by the nematode caenorhabditis elegans: identification of attractants and analysis of the response by use of mutants. *Proceedings of the National Academy of Sciences*, 70(3):817–821, 1973.

[WPH+12] Quan Wen, Michelle Po, Elizabeth Hulme, Sway Chen, Xinyu Liu, Sen Wai Kwok, Marc Gershow, Andrew M. Leifer, Victoria Butler, Christopher Fang-Yen, Taizo Kawano, William R. Schafer, George Whitesides, Matthieu Wyart, Dmitri B. Chklovskii, Mei Zhen, and Aravinthan D. T. Samuel. Proprioceptive coupling within motor neurons drives c. elegans forward locomotion. *Neuron*, 76(4):750–761, Nov 2012. 23177960[pmid].

[WR95] Stephen R Wicks and Catharine H Rankin. Integration of mechanosensory stimuli in caenorhabditis elegans. *Journal of Neuroscience*, 15(3):2434–2444, 1995.

[WR97] Stephen R Wicks and Catharine H Rankin. Effects of tap withdrawal response habituation on other withdrawal behaviors: The localization of habituation in the nematode caenorhabditis elegans. *Behavioral neuroscience*, 111(2):342, 1997.

[WRR96] Stephen R Wicks, Chris J Roehrig, and Catharine H Rankin. A dynamic network simulation of the nematode tap withdrawal circuit: predictions concerning synaptic function using behavioral criteria. *Journal of Neuroscience*, 16(12):4017–4031, 1996.

[WSTB86] JG White, E Southgate, JN Thomson, and S Brenner. The structure of the nervous system of the nematode Caenorhabditis elegans: the mind of a worm. *Phil. Trans. R. Soc. Lond*, 314:1–340, 1986.

[YVT+17] Gang Yan, Petra E. Vértes, Emma K. Towlson, Yee Lian Chew, Denise S. Walker, William R. Schafer, and Albert-László Barabási. Network control principles predict neuron function in the caenorhabditis elegans connectome. *Nature*, 550(7677):519–523, Oct 2017. 29045391[pmid].

[ZGP+09] Manuel Zimmer, Jesse M Gray, Navin Pokala, Andy J Chang, David S Karow, Michael A Marletta, Martin L Hudson, David B Morton, Nikos Chronis, and Cornelia I Bargmann. Neurons detect increases and decreases in oxygen levels using distinct guanylate cyclases. *Neuron*, 61(6):865–879, 2009.

[ZS15] Mei Zhen and Aravinthan DT Samuel. C. elegans locomotion: small circuits, complex functions. *Current opinion in neurobiology*, 33:117–126, 2015.