Precision Neuromodulation Therapies Using Artificial Intelligence

By

Parisa Sarikhani
Doctor of Philosophy

Computer Science and Informatics

Babak Mahmoudi, Ph.D.
Advisor

Mayuresh Kothare, Ph.D.
Committee Member

Rishi Kamaleswaran, Ph.D.
Committee Member

Johnathan McKay, Ph.D.
Committee Member

Accepted:

Kimberly Jacob Arriola
Dean of the James T. Laney School of Graduate Studies

Date

Precision Neuromodulation Therapies Using Artificial Intelligence

By

Parisa Sarikhani
B.Sc., Shiraz University, Iran, 2014
M.Sc., Shiraz University, Iran, 2017
M.Sc., Emory University, GA, 2022

Advisor: Babak Mahmoudi, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2023

Abstract

Precision Neuromodulation Therapies Using Artificial Intelligence
By Parisa Sarikhani

Implantable neuromodulation devices, such as deep brain stimulation (DBS) and vagus nerve stimulation (VNS) have revolutionized neuroscience research and clinical care thanks to their ability to directly intervene in pathological circuits. These implantable devices provide a powerful paradigm for treating neurological disorders, restoring and enhancing neural functions, and understanding the causal links between neural and behavioral processes. Yet, despite their growing adoption in clinical care, challenges persist, impeding their seamless integration into standard of care. Recent advancements in next-generation implantable devices offer considerable customization in stimulation parameters, paving the way for delivering precision neuromodulation therapies. Moreover, given the variations in electrode placement, local anatomy, and the diversity in symptom type and severity, it is imperative to design adaptive, patient-specific treatments. Proper programming of implantable devices is a critical step for optimizing patients' therapeutic outcomes and avoiding inducing adverse side effects. Despite the efforts in developing standard clinical guidelines for programming neuromodulation devices, these approaches are very time-consuming and may lead to sub-optimal therapy for patients. Additionally, these approaches do not take into account the complex and dynamic nature of the nervous system, which can change over time and require ongoing adjustment of stimulation parameters. Therefore, there is a growing need to develop automated intelligent closed-loop neuromodulation systems (iCLON) to facilitate the programming of implantable devices.

Despite these challenges, the potential benefits of these systems for treating neurological disorders make this a promising area of research and development. Designing automated closed-loop neuromodulation systems is a complex task and requires multifaceted considerations. This research is an effort toward facilitating the design and development of automated iCLON systems by developing a translational design paradigm. This dissertation contributed to the development of multiple simulation environments, pivotal in the design of novel and effective iCLON systems. The simulation platforms offer a safe and controlled environment for rigorous testing and refinement before clinical implementation. This research also introduces multiple control tasks, replicating the actual experimental and clinical applications and ensuring reproducibility and easier translation from simulation to clinical practice. Moreover, a control policy is at the core of iCLON systems which automatically learns and adjusts the stimulation parameters. In this research, I developed data-driven control strategies using optimization and reinforcement learning techniques that are able to learn and optimize neuromodulation control strategies autonomously, via closed-loop interaction with the nervous system. Notably, I developed and clinically evaluated a fully automated DBS programming framework for treatment of tremor in patients with Parkinson's disease and essential tremor that was shown to be efficient and safe while providing outcomes comparable to that achieved by expert clinicians. Finally, this research presents a collaborative effort towards developing an end-to-end translational platform for the design and implementation of iCLON systems. Utilizing an algorithm-hardware co-design approach, the platform facilitates the exploration of brain-implantable devices capable of autonomously learning and adapting control policies. This platform aims to enable research and development of brain-implantable iCLON systems for a wide community of neuroscientists, clinicians, and engineers.

Precision Neuromodulation Therapies Using Artificial Intelligence

By

Parisa Sarikhani
B.Sc., Shiraz University, Iran, 2014
M.Sc., Shiraz University, Iran, 2017
M.Sc., Emory University, GA, 2022

Advisor: Babak Mahmoudi, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2023

Acknowledgments

I would like to express my gratitude to the incredible network of people that have supported me throughout my journey towards completing this PhD thesis.

I extend my deepest gratitude to Dr. Babak Mahmoudi, my research advisor, whose invaluable mentorship and support have significantly shaped my academic journey. His remarkable expert guidance, constructive feedback, patience, and faith in my abilities have been instrumental in shaping the direction of my research and in academic growth. Being a part of his research group has been an invaluable opportunity for which I am profoundly grateful. My gratitude also extends to Dr. Svjetlana Miocinovic, whose endless support, expert perspectives, and constructive feedback have greatly enriched my research and my professional development. I am equally indebted to Dr. Mayuresh Kothare, whose invaluable expert insights and feedback have been instrumental in my research. Working with his group has been a pleasure, offering me numerous learning opportunities and enriching my academic experience. I express profound gratitude to Dr. Joseph R. Manns and Dr. Hadi Esmaeilzadeh for their expert perspectives. The immense knowledge I have gained from our many enlightening discussions and working with their groups has been invaluable. I am very thankful to my committee members Dr. Rishi Kamaleswaran, and Dr. Lucas J. MacKay for their guidance and constructive feedback on my thesis.

I am lucky to have collaborated with talented colleagues and friends in Neuroinformatics and Intelligent Systems (NISys) lab Dr. Pradeeban Kathiravelu, Mahmoud Zeydabadinezhad, Yusen Zhu, and Elizabeth Nemeti. I am grateful to my friends Nasim and Yashar, who have been pillars of support throughout my journey. Their presence and encouragement during all the ups and downs have been a source of comfort. I am grateful to my lifelong friends Aida Afkhamizadeh and Mandana Jahanbozorgi for their friendship and motivation.

Reflecting on my journey, I realize how good fortune in many small moments has significantly affected my path to success. To my parents, whose sacrifices and endless encouragement to pursue my dreams have shaped the person I am today, I am forever indebted. Your belief in me has been a constant source of motivation. To my sister, Afrooz, whose companionship have been priceless, thank you for always being supportive. And to my best friend and husband, Mohammadreza, your continuous support, patience, and love has been a constant source of strength and motivation. This achievement is as much yours as is mine and I am incredibly grateful for having such an incredible partner by my side.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Neuromodulation, as defined by the International Neuromodulation Society (INS), is a field of science, medicine, and bioengineering that encompasses implantable and non-implantable technologies, electrical or chemical, for the purpose of improving the quality of life and functioning of humans [1]. Neuromodulation is a rapidly expanding field of medicine that involves a wide range of specialties and affects hundreds of thousands of patients worldwide who suffer from various disorders. There have been significant advancements in the scientific understanding of neuromodulation, its mechanisms, clinical applications, and technological development [1]. Neuromodulation is being used to manage a wide range of conditions including disorders of cardiac pacing [2], epilepsy [3], movement disorders [4], chronic pain [5], psychiatric and neurobehavioral disorders [6], and many more conditions, especially in medication refractory patients.

The use of neuromodulation devices has become a standard and widely accepted treatment for many neurological disorders. However, to achieve the therapeutic benefit, stimulation often requires time-consuming programming by an expert [7]. Most of the current clinical approaches are based on trial-and-error evaluation of therapeutic response, which is very time-consuming and may lead to sub-optimal efficacy of the stimulation treatment [8], [9]. Hence, there is a growing need to create intelligent closed-loop neuromodulation (iCLON) strategies that autonomously learns the optimal stimulation settings and adapts to the complex underlying dynamics of the nervous systems. The development of iCLON systems simplifies the initial programming of the implanted pulse generator (IPG) [10], facilitating the programming of neuromodulation devices, while also enhancing long-term therapy for patients by automatically adjusting therapy to continuously optimize patient outcomes [11].

Designing iCLON systems is a complex and multidisciplinary task that requires expertise from various backgrounds including neuroscience, engineering, signal processing, etc. Moreover, the complexities of the interactions between the nervous system and the neuromodulation systems, with large parameter spaces, pose challenges for designing and effective clinical deployment of neuromodulation technologies [12]. Despite the challenges, the potential benefits of these systems for treating neurological and psychiatric disorders make this a promising area of research and development. The focus of this research is to contribute to the advancement of automated iCLON systems by developing a translational design paradigm that facilitate the design, implementation, and clinical translation of these potentially life saving treatments.

1.2 Challenges and requirements of designing intelligent closed-loop neuromodulation Systems

One critical aspect of designing closed-loop neuromodulation systems is ensuring modularity. The intricate nature of the nervous system motivates partitioning the design of iCLON systems into multiple components [11]. By using a modular design, the system becomes flexible and scalable, allowing for straightforward modifications and customization, which is especially important for closed-loop neuromodulation systems that need to adapt to individual patient responses and changing treatment needs [9]. In addition, the modular design facilitates collaboration among experts with different expertise, allowing for the reproducibility of previous works, and ensuring that the system can integrate new technologies and advancements in the field [11].

Another challenge is the limitations of the current standard of care in the programming of neuromodulation devices. Typically, standard clinical approaches in the programming of neuromodulation devices involve a trial-and-error evaluation of therapeutic response (clinical benefit and unwanted side effects) at numerous stimulation settings [7], [13]. This is often performed over several sessions, which can be inconvenient and costly for patients and clinicians and can be a challenge for patients who live far away from specialty care [14]. Developing automated iCLON systems eliminates the need to frequently access to specialized centers and unlocks access to a wider range of patients. Such frameworks could be beneficial for remote programming for patients with limited access to the clinic. In addition, the standard clinical approaches are often limited to a small subset of stimulation parameters, which may not allow for optimal customization to individual patient needs [15]. Current clinical approaches may not take into account the complex and dynamic nature of the nervous system, which can change over time and require ongoing adjustment of stimulation parameters. As a result, there is a growing interest in the development of automated iCLON systems, which can provide real-time feedback and automatic adjustment of stimulation parameters based on individual patient needs and changing physiological conditions.

Moreover, automation of iCLON systems is an essential feature that plays a significant role in various aspects of the system. First, it can help to reduce the burden on clinicians by automating routine tasks such as parameter adjustment and data logging allowing clinicians to focus on other tasks [14]. In addition, the evaluation of the behavioral or neurophysiological response of patients to modifications in therapy can be challenging given the subjective nature of visual observation. Automated discovery of the target objective functions reduces the burden and helps to objectively validate the subjects' response to therapy [8, 11]. Another aspect of automation is automating the treatment plan (control policy) which automatically adjusts stimulation parameters based on patients' state and reduces the delay between stimulation updates compared to human intervention [11].

Another challenge in developing effective iCLON systems is to take the complex and dynamic nature of the nervous system into account. Adaptability enables designing adaptable systems in a patient-specific way based on each patient's unique needs and responses [16, 11]. Moreover, it enables the adjustment of stimulation parameters in real time based on individual patient needs and changing physiological conditions. This can help to optimize treatment outcomes and improve patient satisfaction. Furthermore, adaptability can also enhance patient safety by minimizing the risk of overstimulation or unintended side effects.

Most of the current standard approaches of treatment planning (control policy) are based on trial-and-error open-loop testing and evaluation of the subject's response to therapy. However, this method is very time-consuming, requires extensive training and specialized skills, and tests a limited subset of the parameter space leading to sub-optimal responses to therapy. Previous studies have made efforts to automate treatment planning using classical control strategies [17, 18, 19, 20, 21] which have several limitations. Although classical control approaches like proportional integral (PI) controllers or model predictive control (MPC) has shown to be effective in many studies, they have some disadvantages which make them impractical for many real-world physiological applications. Some disadvantages of PI controllers are including limited controllability which makes them less effective in transient responses, tuning complexity, high sensitivity to model parameters, and sub-optimal performance in non-linear systems [22], [23]. While MPC has been shown to address some of the challenges of classical control algorithms, it still requires tuning of various parameters, such as the prediction horizon and control weights, to achieve optimal performance which can be challenging. In addition, MPC requires having access to an accurate model of the system, and errors or inaccuracies in the model can affect the controller's performance [24]. Recent advances in artificial intelligence (AI) enable the design of automated iCLON systems, that are able to learn and optimize neuromodulation control strategies autonomously, via closed-loop interaction with the nervous system with minimal assumptions and requirements and may be used to address the algorithmic challenges.

Finally, although recent advances in AI enable the design of iCLON systems that are able to learn and optimize neuromodulation control strategies autonomously, there are many challenges in designing iCLON systems and translating them in clinical settings including software implementation, hardware integration, experimental validation, and clinical deployment in implantable devices. These complexities may make designing iCLON systems out of reach for the broader biomedical research community and may render designing systems that are not translatable into clinical settings. The computational complexity of these classes of algorithms cannot be met with general-purpose embedded systems and there is a need for specialized, yet programmable, hardware.

1.3 Significance and contributions

Despite the challenges faced in developing and implementing the iCLON systems, the potential benefits of utilizing them to treat neurological and psychiatric disorders make this field of research and development highly promising. These systems hold the potential to provide more targeted and effective treatment options, reduce the burden of medication side effects, and improve the overall quality of life for patients. The focus of this thesis is to develop a translational design paradigm that considers the aforementioned requirements into account and facilitates quick design, prototyping, and clinical translation of iCLON systems.

This thesis contributed to the development of multiple simulation environments, pivotal in the design of novel and effective iCLON systems. The intricate and complex nature of nervous systems, coupled with practical limitations and safety concerns in interacting with them, presents significant challenges in designing successful iCLON systems. To address this challenge and accelerate the development of new and more effective iCLON systems, I developed and integrated multiple simulation environments to accelerate the development of new and more effective iCLON systems. The simulation environments can provide a safe and controlled environment for testing and prototyping such systems, and allow researchers to refine system parameters before clinical implementation. Furthermore, simulation environments enable the reproducibility of results, facilitating the comparison of different systems and methodologies. They also allow for the rapid prototyping of iCLON systems, enabling researchers to design, test, and optimize new systems quickly.

In addition to the simulation environments which are surrogates of the actual nervous system, this thesis also introduces multiple control tasks replicating the actual experimental and clinical applications for better reproducibility of the simulations and easier translation into clinical practice. These tasks include designing a minimization task to minimize tremor severity while avoiding adverse side effects for Parkinson's disease (PD) and essential tremor (ET) patients, a set-point tracking task for regulating the heart rate (HR) and mean arterial pressure (MAP) for the treatment of various cardiovascular diseases that include heart failure, arrhythmia, and hypertension, and a synchrony suppression task designed to suppress synchrony in a mean-field model of the neural population which is known to be an underlying cause of many adverse symptoms in PD patients.

Moreover, a control policy is at the core of iCLON systems which automatically learns and adjusts the stimulation parameters in order to achieve the goals of a desired neuromodulation control or optimization task. Most of the current standard approaches of treatment planning (control policy) are based on trial-and-error open-loop testing and evaluation of the subject's response to therapy. There have been previous efforts to automate treatment planning using classical control strategies which have multiple limitations as discussed in the previous section. Recent advances in AI enable the design of automated iCLON systems, that are able to learn and optimize neuromodulation control strategies autonomously, via closed-loop interaction with the nervous system. One major contribution of this research is developing data-driven control strategies using optimization and reinforcement learning techniques that are able to learn and optimize neuromodulation control strategies autonomously, via closed-loop interaction with the nervous system. The developed iCLON systems using novel data-driven control strategies has been designed and evaluated in the context of multiple neuromodulation applications including an automated DBS programming framework for tremor, an automated closed-loop system for regulation cardiovascular system with selective VNS, and an automated iCLON system for synchrony suppression with adaptive DBS.

Additionally, this research presents a collaborative effort towards developing an end-to-end translational platform, called Neuroweaver, for the design and implementation of iCLON systems. Utilizing an algorithm-hardware co-design approach, the platform facilitates the exploration of brain-implantable devices capable of autonomously learning and adapting control policies. AI and RL can lay a pathway since they have proven to be effective in dynamic environments. However, the computational complexity of these classes of algorithms cannot be met with general-purpose embedded systems and there is a need for specialized, yet programmable, hardware.

As such, a fixed architecture is incapable of accommodating design space explorations, matching specific algorithmic needs, and/or operational constraints. Thus, this thesis presents a template architecture that is a highly parametric design, capable of being scaled down or scaled up before fabrication to match the requirements and enable algorithm-hardware design exploration. After explorations and analyses, the framework can generate a concrete design that can be deployed on Field Programmable Gate Arrays (FPGAs) for prototyping/use in-vivo experimentation. This design is also ready to be fabricated as a stand-alone programmable custom chip that can be implanted. This platform aims to enable research and development of brain-implantable iCLON systems for a wide community of neuroscientists, clinicians, and engineers. Some lines of work in the development of Neuroweaver (as described in sections 5.4, 5.5 and the results associated with them) are implemented by our collaborator, but are provided here to describe the comprehensive view of the research platform. The dissertation contributed to the design of the platform and the development of novel data-driven and interactive RL-based control strategies. In addition to the success of the various RL-based iCLON control strategies in performing the synchrony suppression task, these approaches are being used to explore the algorithm-hardware co-design approach which enables the the design and development of novel brain-implantable neuromodulation devices.

In summary, this research contributed to the advancement of automated interactive iCLON systems from design and simulation to implementation. Novel interactive data-driven control strategies that allow the iCLON systems learn and adapt through interaction with the nervous system were designed. Data-driven and mechanistic models of the nervous systems under external stimuli were utilized for developing in-silico simulation environments for prototyping and evaluating the performance of these data-driven approaches. Additionally, software development efforts enabled clinical implementation of my data-driven and patient-specific DBS programming approach for further clinical evaluation of the system. Finally, the algorithm-hardware co-design approach enabled the research and development of brain-implantable iCLON devices using the proposed interactive AI-based strategies.

1.4 Thesis outline

This thesis comprises six chapters, besides the introduction and the conclusion, all of the other chapters have been published or are under review in key journals and conferences in the field (see section 1.5).

Chapter 2 presents the background of this thesis. This chapter provides an overview of closed-loop neuromodulation systems, with a focus on control policy algorithm development. This chapter provides insight on the recent trends in developing closed-loop neuromodulation systems and emphasizes on the need for developing research platforms that enable the research and development of intelligent closed-loop neuromodulation systems.

Chapter 3 presents the design, implementation, and clinical validation of an automated DBS programming framework with safety constraints for tremor suppression in patients with Parkinson's disease and essential tremor. (Safe) Bayesian optimization which is a sample-efficient global optimization method was used as the core of this DBS programming framework to adaptively learn each patient's response to DBS and suggest the next best settings to be evaluated. This study demonstrated that fully automated DBS programming framework for treatment of tremor is efficient and safe while providing outcomes comparable to that achieved by expert clinicians.

Chapter 4 provides a novel interactive AI framework using RL which provides an automated data-driven approach for closed-loop regulation of cardiovascular system with selective VNS. The proposed VNS control policies was used to regulate HR and MAP in computational models of rat cardiovascular system with minimal assumptions and without the need for prior knowledge about the underlying physiological dynamics of the system. The results from this study demonstrated the capabilities of the closed-loop RL-based approaches to learn optimal VNS control policies and to adapt to variations in the target set points and the underlying dynamics of the cardiovascular system. The findings of this study highlighted the trade-off between sample-efficiency and generalizability, providing insights for proper algorithm selection. In addition, transfer learning was utilized to improve the sample efficiency of Deep RL algorithms allowing the development of more efficient and personalized closed-loop VNS systems.

Chapter 5 presents an open-source end-to-end platform, called Neuroweaver, for design and development of translatable embedded AI algorithms that enable intelligent closed-loop neuromodulation devices autonomously to learn and adapt control policies from interacting with the nervous system. Neuroweaver is an AI algorithm-hardware co-design platform that provides an integrated environment for modular designing and prototyping intelligent closed-loop neuromodulation control systems and deploying AI pipelines in hardware through a Python-embedded cross domain interface. Neuroweaver aims to build an end-to-end translational platform from design to implementation of iCLON systems to enable research and development of iCLON systems for a broader community of neuroscientists, clinicians, engineers, and developers.

1.5 List of publications

Journal Papers

- **P. Sarikhani**, H.-L. Hsu, M. Zeydabadinezhad, Y. Yao, M. Kothare, and B. Mahmoudi, "Reinforcement Learning for Closed-loop Regulation of Cardiovascular System with Selective Vagus Nerve Stimulation," *submitted to Journal of Neural Engineering, 2023.*

- **P. Sarikhani**, B. Ferleger, K. Mitchell, J. Ostrem, J. Herron, B. Mahmoudi, S. Miocinovic, "Automated Deep Brain Stimulation Programming with Safety Constraints for Tremor Suppression in Patients with Parkinson's Disease and Essential Tremor," *Journal of Neural Engineering, 2022.* (link)

- P. Kathiravelu, M. Arnold, J. Fleischer, Y. Yao, S. Awasthi, A.K. Goel, A. Branen, **P. Sarikhani**, G. Kumar, M.V. Kothare, and B. Mahmoudi, "CONTROL-CORE: A Framework for Simulation and Design of Closed-Loop Peripheral Neuromodulation Control Systems," *IEEE Access, 2022.* (link)

- J.K. Kim, B.H. Ahn, S. Kinzer, S. Ghodrati, R. Mahapatra, B. Yatham, S.T. Wang, D. Kim, **P. Sarikhani**, B. Mahmoudi, and D. Mahajan, J. Park, H. Esmaeilzadeh "Yin-Yang: Programming Abstractions for Cross-Domain Multi-Acceleration," *IEEE Micro, 2022.* (link)

- P. Kathiravelu, **P. Sarikhani**, P. Gu, B. Mahmoudi, "Software-Defined Workflows for Distributed Interoperable Closed-Loop Neuromodulation Control Systems," *IEEE Access, 2021.* (link)

Conference Papers and Presentations

- **P. Sarikhani**, H. Xu, S.-T. Wang, S. Kinzer, Y. Zhu, J. Krasney, J. R. Manns, H. Esmaeilzadeh, B. Mahmoudi, "Neuroweaver: a translational platform for embedding artificial intelligent in closed-loop neuromodulation systems," *Neuroscience, 2023.*

- Y. Zhu, S. Hossein, **P. Sarikhani**, P. Gu, S. Betters, S. Liu, R. Tweedy, P. Kathiravelu, T. Pan, M. Treadway, B. Mahmoudi, "Nexus: an interoperable and distributed platform for optimal closed-loop experimental design," *Neuroscience, 2023.*

- **P. Sarikhani**, H.-L. Hsu, and B. Mahmoudi, "Automated Tuning of Closed-loop Neuromodulation Control Systems using Bayesian Optimization," *44th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2022.* (link)

- **P. Sarikhani**, H.-L. Hsu, J. K. Kim, S. Kinzer, E. Mascarenhas, H. Esmaeilzadeh B. Mahmoudi, "Neuroweaver: Towards a Platform for Designing Translatable Intelligent Closed-loop Neuromodulation Systems," *Bridging the Gap: From Machine Learning Research to Clinical Practice, NeurIPS, 2021.*

- **P. Sarikhani**, H.-L. Hsu, M. Zeydabadinezhad, Y. Yao, M. Kothare, B. Mahmoudi, "Sparc: Adaptive Closed-loop Control of Vagal Nerve Stimulation for Regulating Cardiovascular Function using Deep Reinforcement Learning: a Computational Study," *Neuroscience, 50th Annual Meeting, 2021.*

- **P. Sarikhani**, B. Ferleger, J. Herron, B. Mahmoudi, S. Miocinovic, "Automated Deep Brain Stimulation Programming with Safety Constraints for Tremor Suppression," *4th International Brain Stimulation Conference, 2021.* (link)

- **P. Sarikhani**, H.-L. Hsu, O. Kara, J. K. Kim, H. Esmaeilzadeh B. Mahmoudi, "Neuroweaver: a Platform for Designing Intelligent Closed-loop Neuromodulation Systems," *4th International Brain Stimulation Conference, 2021.* (link)

- **P. Sarikhani**, S. Miocinovic, B. Mahmoudi, "Towards Automated Patient-Specific Optimization of Deep Brain Stimulation for Movement Disorders," *41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2019.* (link)

Patent

- S. Miocinovic, B. Mahmoudi, **P. Sarikhani**, "Systems and Methods for Automated Deep Brain Stimulation Programming," U.S. Patent Application No. 17/315,129, filed on May 7, 2021. (link)

Chapter 2

A review on closed-loop neuromodulation systems, with a focus on control policy algorithms

2.1 Introduction

Neuromodulation stands at the intersection of science, medicine, and biomedical engineering which encompasses implantable and non-implantable technologies for the purpose of improving the quality of life and functioning of humans [1]. This is a rapidly expanding field of medicine that involves a wide range of specialties and affects hundreds of thousands of patients worldwide who suffer from various neurological conditions including disorders of cardiac pacing [2], epilepsy[3], movement disorders [4], chronic pain[5], psychiatric and neurobehavioral disorders[6], and many more. The recent remarkable advancements in neurotechnology, underscore its potential for significant societal impact, necessitating tailored efforts to address its evolving scientific, technical, and social needs.

Despite the widespread acceptance of neuromodulation devices, achieving therapeutic efficacy heavily relies on time-consuming trial-and-error stimulation adjustments by experts [7]. The open-loop trial-and-error process enables constant stimulation regardless of patients' symptoms and neurophysiological state. This arduous process poses a significant challenge in their effective clinical deployment, limiting the efficacy and accessibility of the therapy. The current approaches often lead to sub-optimal stimulation settings [8, 9] due to the variations in individual recovery trajectories, subjective symptom evaluation methods, and lack of a systematic approach for stimulation adjustment with large parameter spaces. In addition, constant open-loop stimulation may lead to over-stimulation and decrease in the device's battery lifespan [25, 26].

An alternative form of stimulation, known as closed-loop or adaptive stimulation, has emerged to address the challenges of open-loop stimulation. Closed-loop neuromodulation provides the benefit of continuously monitoring patients' neurophysiological and/or behavioral activities. It allows for the on-demand adjustment of stimulation parameters in response to alterations in the patients' pathological states and symptoms. Although conventional open-loop neuromodulation is a standard treatment, many studies have demonstrated that closed-loop approaches are more effective and efficient. Previous studies has reported improved therapeutic efficacy of closed-loop DBS compared to standard open-loop approaches [25, 27]. A review study [28] stated that adjusting stimulation parameters in closed-loop DBS could reduce adverse side effects induced by DBS in Parkinson's disease. Additionally, recent studies reported a reduction in the stimulation time by integrating closed-loop neuromodulation strategies, leading to less energy consumption of the neuromodulation devices [25, 29]. The increased efficiency and reduced energy consumption leads to a higher battery lifespan and fewer replacement surgeries.

Despite the numerous advantages of closed-loop neuromodulation strategies, there are no commercially available closed-loop neuromodulation device that allows for designing flexible and adaptive stimulation strategies. Since the FDA approval of commercial DBS devices in 1997 for tremor treatment [30], they have been authorized for treating other neurological disorders like Parkinson's Disease [31], dystonia [32], and obsessive-compulsive disorder [33]. The NeuroPace's RNS system is a commercial closed-loop DBS system which has been approved by FDA for treating drug-resistant epilepsy [34]. The RNS system works by activating stimulation once a seizure-related predefined pattern is detected in patients' ECoG signal to disrupt the detected patterns in brain signals and avoid seizures. Furthermore, Medtronic's Activa PC + S is an implantable sensing-stimulating DBS device designed for investigational purposes. The Activa PC + S does not use the recorded LFP as a feedback signal to close the loop, however, the user is able to compile a program to close the loop with limited flexibility [26].

Designing and clinical implementation of closed-loop neuromodulation systems is a challenging and multidisciplinary task that requires expertise from various backgrounds including neuroscience, engineering, software, and hardware development. The limited understanding of the dynamics of the nervous system with different time scales in response to stimuli further complicates their effective clinical deployment. Despite the challenges, the potential benefits of these systems for treating neurological disorders make this a promising area of research and development. Developing closed-loop neuromodulation systems holds the potential to revolutionize the standard of care in treatment-resistant neurological disorders. Hence, there is a growing need to develop intelligent closed-loop neuromodulation (iCLON) systems that are able to learn and optimize the neuromodulation control strategies autonomously, via closed-loop interaction with the nervous system.

An iCLON system is composed of multiple essential component including the signal acquisition, biomarker detection, optimization or control algorithm, stimulation policy, and the hardware components to enable stimulation delivery (Figure 2.1b) while software platforms enables seamless communication of these components [35, 36]. Although all of these components are crucial in designing a successful iCLON system, the optimization and control algorithm is at the core of iCLON systems which automatically learns and adjusts the stimulation parameters. In this narrative review, I introduced the common closed-loop control strategies that has been used in the literature. For the purpose of this review, the applications of deep brain stimulation (DBS) and vagus nerve stimulation (VNS) are included. Based on the findings of several recent studies of closed-loop DBS and VNS, I highlighted the recent trends in developing novel iCLON systems and the need for developing research platforms to facilitate designing novel and more effective approaches.

2.2 Introduction to closed-loop neuromodulation control

Closed-loop control systems are designed to achieve and maintain a specified outcome by autonomously adjusting their input parameters based on real-time feedback [37]. This self-regulating framework operates on the principle of a dynamic feedback loop where the system's output is continuously monitored and fed back into the system to adjust the input, ensuring the output remains within the desired range. The input is the variable that the system can manipulate directly, while the output is the response that needs to be controlled, which could be either a single variable or a combination of variables depending on the complexity of the system. In essence, closed-loop control systems are capable of adapting to disturbances and maintaining their function without external intervention as opposed to open-loop systems which operate without feedback and are thus unable to self-correct in the face of perturbations (Figure 2.2).

In neuroscience, closed-loop neuromodulation systems are aimed at regulating the nervous system. Closed-loop neuromodulation systems, such as DBS and VNS, are aimed to adjust therapeutic interventions in real-time, responding to neural signals that signify changes in the patient's condition. Here, the input takes the form of electrical or optogenetic stimuli delivered to the target neural system, and the output is the neurophysiological or behavioral response that represent the neural activity. These iCLON systems not only deliver therapeutic stimulation but also must be equipped with either internal sensors that monitor electrophysiological biomarkers or external sensors to monitor behavioral biomarkers indicative of the current state of the nervous system, thereby forming a feedback loop (Figure 2.1b).

A general design of iCLON systems consists of hardware modules to enable interactions with the nervous system and algorithmic modules to enable intelligent automation of the interactions to modulate and maintain the desired neural activity (Figure 2.1b). The hardware modules consist of a sensing component called signal acquisition in Figure 2.1b to measure the neurophysiological and/or behavioral responses of the nervous system to therapy and another component called stimulation delivery in Figure 2.1b to apply stimulation or proper interventions when needed to optimize therapeutic outcomes. The algorithmic component of the system needs to be modular enough for easier modifications while replicating the clinical workflow for better translatability of the system into clinical practice. In clinical practice, there is typically a separation between patient assessment and treatment planning. Translating the clinical practice into an automated iCLON system motivates the partitioning of the algorithmic component into a biomarker or objective identification module and a control policy module. The biomarker or objective identification estimates the subjects' state or symptoms in response to therapy and replicates the clinical assessment, while the control policy module replicates the treatment plan. A high-level overview of the modular design of iCLON systems is depicted in Figure 2.1.

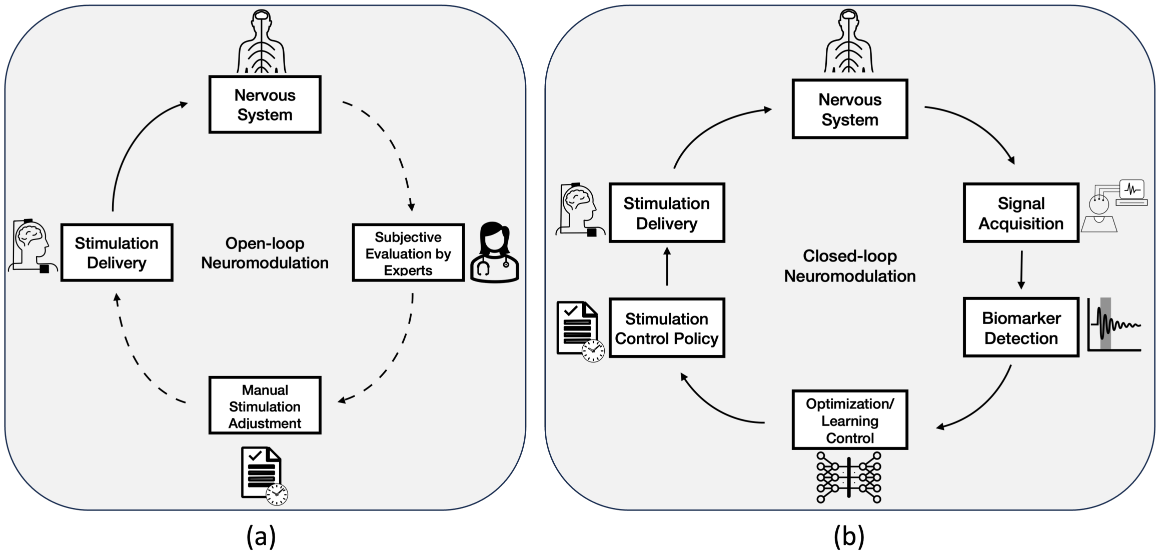

Nervous System
Subjective Evaluation by Experts
Manual Stimulation Adjustment
Stimulation Delivery
Open-loop Neuromodulation
(a)
Nervous System
Signal Acquisition
Biomarker Detection
Optimization/ Learning Control
Stimulation Control Policy
Stimulation Delivery
Closed-loop Neuromodulation
(b)

Figure 2.1: Comparison of (a) an open-loop versus (b) a closed-loop Neuromodulation programming system.

2.3 Control strategies

2.3.1 Open-loop control

Open-loop also known as feed-forward control strategies operate on the principle of generating commands for a system under control (plant) with the expectation of achieving a specific output without the incorporation of feedback to modulate a command [38]. As illustrated in Figure 2.2, this control scheme relies on having access to a perfectly defined model of the system's behavior, assuming that the system will respond predictably to the commands given. However, without the ability to measure the actual output or the error of the system, the open-loop controller lacks the ability to adjust its commands in response to real-world variables such as noise or measurement discrepancies. This assumption of a flawless system model is a significant limitation of open-loop control, as it does not account for the unpredictable and often complex variations that can occur in practical scenarios, making it less adaptive and reliable when precision and adaptability are crucial.

2.3.2 Closed-loop control

Closed-loop control strategies, also known as feedback control, introduce a critical component into the system: sensors. These sensors are intrinsic to the system's ability to self-regulate, as they continuously monitor the system's output in response to the commands issued by the controller. If there is any deviation from the expected output the sensors provide an error signal that quantifies the discrepancy. This error signal is then utilized by the feedback controller to adjust subsequent commands, fine-tuning the system's behavior. This dynamic process of monitoring and adjustment is mathematically sophisticated, with various algorithms available to optimize the controller's response [37, 39]. Through this continual loop of action, measurement, and correction, closed-loop control systems achieve greater accuracy and stability, adapting in real-time to ensure the desired outcome is maintained. A block diagram of a closed-loop control strategy is depicted in Figure 2.2.

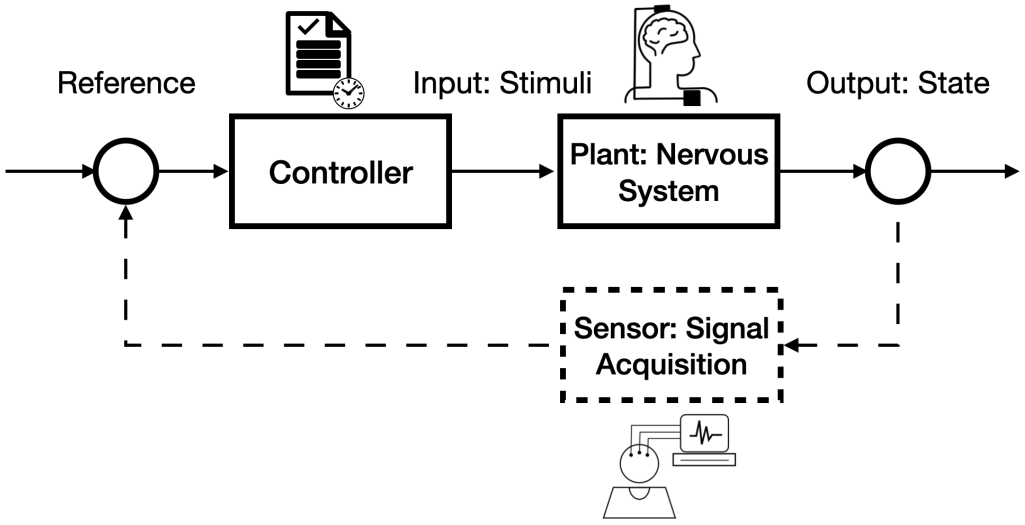


Figure 2.2: Open-loop and closed-loop control strategies. The open-loop control strategy is demonstrated with solid lines.

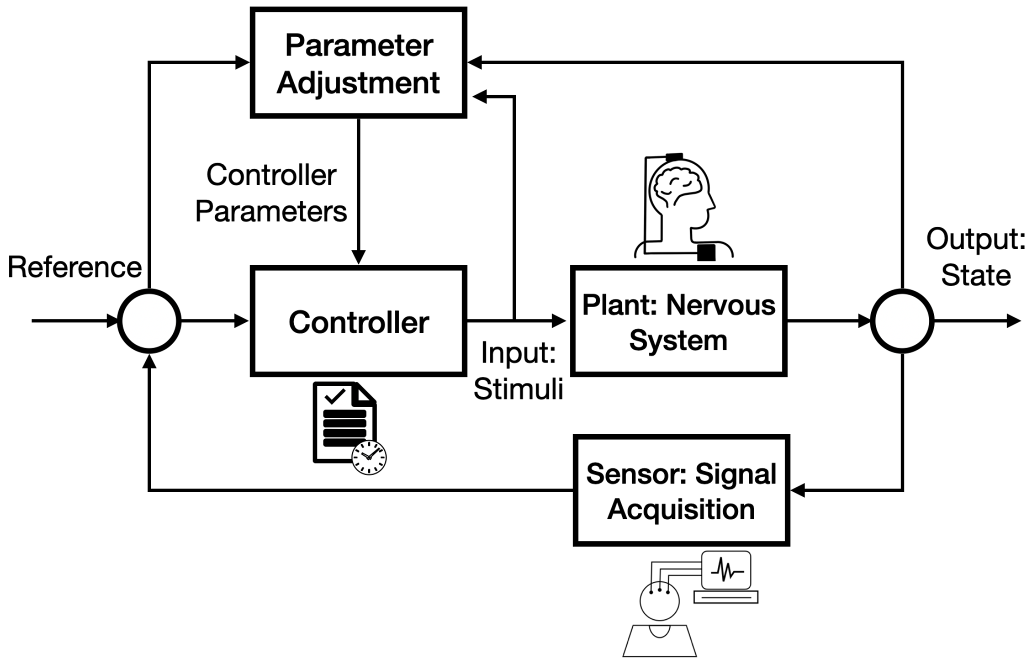


Figure 2.3: Overview of an adaptive control strategy.

2.3.3 Adaptive control

Adaptive control [40] strategies integrate with both open-loop and closed-loop systems, using sensor data to tailor the controller's actions to real-time environmental changes or shifts within the underlying dynamics of the system. This approach allows for a flexible control strategy that compensates for the lack of complete system knowledge and is especially advantageous when dealing with non-stationary systems. Although not always optimal, adaptive controllers excel by tuning parameters to maintain stability and convergence with evolving system's characteristics (Figure 2.3).

2.3.4 Model-based control

Model-based control strategies hinge on the utilization of an accurate mathematical representation of the system to predict its behavior and formulate control commands accordingly [41]. This comprehensive model encapsulates the dynamics of the system, including its responses to various inputs and disturbances. By incorporating this knowledge, the control algorithm can anticipate the system's reactions to different control actions and adjust the inputs to achieve the desired outcome. The strength of model-based approaches lies in their predictive power, which enables foresight and planning in the control process. However, this also implies that inaccuracies in the model predictions can lead to sub-optimal control performance. When the model precisely reflects the system's actual dynamics, model-based control can achieve high levels of efficiency and accuracy, making it ideal for systems where the underlying processes are well-understood and can be reliably captured.

2.3.5 Model-free control

Model-free control strategies represent a paradigm shift in control systems, designed to operate effectively without a comprehensive mathematical model of the system they regulate. These strategies rely on real-time data and adaptive algorithms to make appropriate control decisions, learning directly from interactions with the system. This approach is particularly useful in complex or highly uncertain environments where developing an accurate system model is challenging or impractical. By employing techniques such as reinforcement learning [42], neural networks [43], or fuzzy logic [44], model-free controllers can deduce the optimal control actions through continuous trial and error and pattern recognition, thus improving their performance over time. This methodology not only bypasses the need for exact model of the underlying dynamics of the system but also enhances flexibility, enabling control systems to adapt to the changes.

2.4 Classical control algorithms in iCLON systems

Control policies can be implemented by one or multiple control strategies as described in section 2.3. In this section, the common control algorithms used to design closed-loop DBS or VNS systems are provided.

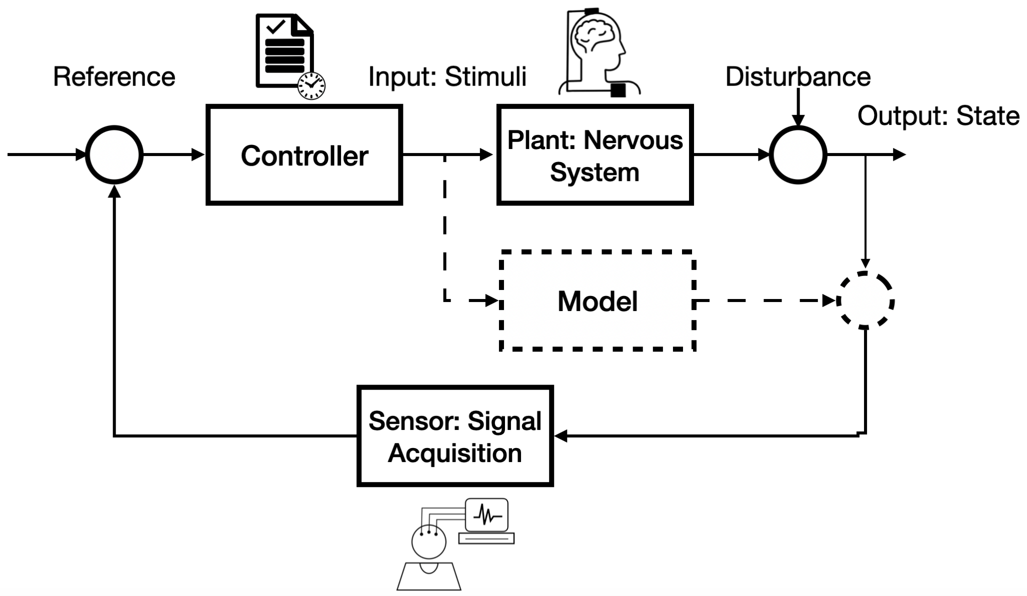


Figure 2.4: Model-based vs. model-free control strategies. The model-free approach is demonstrated with solid lines.

2.4.1 On-Off and threshold-based controller

The on-off control is activated when a measured variable or a target biomarker is passed a predefined threshold. This simple control scheme has been in many closed-loop neuromodulation studies and is capable of preventing brain from over-stimulation. Authors in [29] employed an on-off control scheme which was was utilized to modulate electrical stimulation with the objectives of suppressing locomotion and inducing freezing when hippocampal theta oscillations, measured from local field potential (LFP) signals, exceeded a specified threshold. Numerous studies designed an on-off closed-loop control system based on beta band frequency for synchrony suppression in PD [45, 46, 47]. Authors in [25, 48, 49] designed a threshold-based on-off controller based on beta frequency amplitude using subject-specific threshold values. Other studies have designed phase-responsive closed-loop DBS for tremor suppression [50, 51].

2.4.2 Proportional-integral-derivative control

The Proportional Integral-Derivative (PID) controller [52] is a very common approach in designing closed-loop systems which has been widely explored in the context of closed-loop neuromodulation systems. The PID controller operates through three key components: the proportional element, which adjusts the control action in direct proportion to the error signal; the integral element, which targets the reduction of steady-state errors by implementing low-frequency compensation through an integrator; and the derivative element, which enhances transient response by offering high-frequency compensation via a differentiator. This three-tiered approach equips the PID controller with a comprehensive mechanism to offer a simple yet efficient solution to many applications [52].

Authors in [53] presented a PID controller for synchrony suppression of neural activity. Proportional (P) and proportional-integral (PI) closed-loop controllers for amplitude and frequency modulation were investigated in [54] for regulating network beta-band activity whilst accounting for clinical considerations. A P controller based on beta power has been used in [55] to reduce motor symptoms of PD in parkinsonian rats. An adaptive P controller is presented in [56] in which the gain of a feedback controller is continuously adjusted to sustain suppression of pathological beta-band oscillatory activity at a desired level. A multivariable control architecture based on PID control is presented in [57] to selectively target suppression of either tremor or subthalamic nucleus beta band oscillations. In addition, model-based algorithms that operate based on PID feedback has been studied in [58, 59]. I introduced a data-driven approach to automated tuning of PI controller to modulate the dominant frequency of neural activity in a computational model [60]. I used Bayesian optimization for the automated parameter tuning of a PI neuromodulation controller.

2.4.3 Delayed-feedback controller

The idea of the delayed-feedback control (DFC) algorithm is to apply a feedback signal that is proportional to the difference between the current state and a delayed state of the system [61]. This delay element is crucial as it enables the control mechanism to effectively adjust for deviations from desired behavior. The DFC algorithm has been used fro the suppression of pathological oscillations in a neural mass model [62]. To restore the desynchronized dynamics in networks of oscillatory neurons, multiple model-based closed-loop stimulation methods have been developed including single-site linear [63, 64], multi-site linear [65, 66] DFC algorithm. In addition, non-linear DFC has been studied in other studies to restore desynchronized dynamics of neural activity [67, 68, 69, 70], where the LFP signal was used as a feedback to adjust the stimulation amplitude using linear or nonlinear DFC.

2.4.4 Fuzzy logic controller

A Fuzzy Logic Controllers (FLCs) are adaptive and versatile control system introduced in [71, 72] that has garnered popularity due to its their adaptability and high performance. Unlike traditional control systems, FLCs utilize linguistic or qualitative information for its control algorithm,allowing them to manage multivariable and inconsistent processes, where the exact measurement of input and its effect is difficult. FLCs rely on fuzzy sets and fuzzy reasoning and their design involves creating “if ... then ...” rules based on historical data and expert knowledge. This flexible framework enables FLCs to make effective decisions in complex real-world scenarios where traditional binary logic does not work.

FLCs have been utilized in the literature to design closed-loop DBS for seizure suppression [73]. This study has utilized adaptive fuzzy terminal sliding mode control (AFTSMC) for eliminating the oscillatory spiking behavior in childhood absence epilepsy in a computational model consisting of cortical, thalamic relay, and reticular nuclei neurons. Authors in [74] used a robust multi-input multi-output AFTSMC approach to control of membrane potential of thalamic neuron populations through continuous adaptive DBS current applied to the thalamus. A previous study [75] presented a multi-disease closed-loop DBS device that can detect the disease using a classifier and adaptively deliver electrical stimulation pulses based on the disease state. Both the classifier and controller are designed using the fuzzy algorithm [75]. A combined control strategy using intelligent single input interval type-2 fuzzy logic (iSIT2-FL) and non-integer sliding mode control (SMC) was presented in [76] to control Parkinson's tremor and reduce the value of stimulation intensity efficiently.

2.4.5 Model predictive control

Model Predictive Control (MPC) is a control methodology characterized by its optimization-driven approach [41]. In MPC, the control strategy aims to minimize a cost function for a constrained dynamical system over a finite, receding horizon. At each discrete time step, the MPC controller acquires or estimates the current state of the system and employs this information to compute a sequence of control actions that optimally minimize the cost over the prediction horizon. This computation involves solving a constrained optimization problem utilizing an internal model of the plant and the current system state as inputs. However, unlike traditional control methods, MPC only implements the first control action from the calculated sequence while disregarding the subsequent ones. This process then iterates at each time step, allowing MPC to adaptively and optimally control dynamic systems in real-time.

MPC has been widely explored in designing iCLON systems. MPC has been shown to be effective in minimizing patient symptoms and device power consumption based on identifying patient-specific models of symptom response to DBS [77]. These patient-specific models was used to formulate a model predictive control strategy for closed-loop DBS.

A recent study developed a closed-loop brain-computer interface system of predictive neuromodulation for treating major depressive disorder (MDD) [78]. This study used a biophysically plausible ventral anterior cingulate cortex (vACC)-dorsolateral prefrontal cortex (dlPFC) neural mass model of MDD to simulate nonlinear and multiband neural dynamics in response to DBS [78] and integrated a MPC strategy that takes the estimated MDD brain state as the feedback signal and optimally adjusts DBS parameters. Nonlinear MPC (NMPC) is an extension of linear MPC which was used in [79] to design a closed-loop stimulation strategy in a delayed neural fields model of parkinsonian STN-GPe network. Authors in [80] used a NMPC strategy for for adaptive adjustments of deep brain stimulation parameters in basal ganglia-thalamic network. Other studies investigated the use of NMPC in the context of closed-loop VNS for regulating heart rate and blood pressure [81, 82].

2.5 Recent trends in developing novel closed-loop neuromodulation systems

The majority of the literature employed classical control theory approaches as described in the previous section. Although the classical control approaches has proven to be effective in many studies, they have multiple limitations that can hinder their efficacy in many complex and dynamic systems including the nervous system. These limitations include difficulties in handling nonlinearities, uncertainties and noise of measurement, and time-varying dynamics, as well as the need for accurate mathematical models of the system, which are often challenging or impossible to obtain for the complex neural dynamics. Data-driven optimization and control approaches, on the other hand, are suitable for situations where classical control methods struggle by learning directly from data, allowing for adaptive control strategies that do not rely on precise models of the underlying dynamics of the target nervous system under neuromodulation interventions.

Some data-driven optimization strategies like Bayesian optimization [83] eliminate the need of having access to underlying equations of the system to develop automated closed-loop neuromodulation systems. Bayesian optimization is a non-parametric global optimization approach that is suitable for optimizing black-box objective functions that are unknown or expensive to evaluate. Bayesian optimization does not require prior knowledge regarding patients' response to neuromodulation therapy and tries to find the optimal setting in a purely data-driven manner and through interactions with the nervous system. Bayesian optimization has been successfully adopted for developing closed-loop neuromodulation applications in the context of optimizing the experimental design with closed-loop real-time functional magnetic resonance imaging (fMRI) [84], optimizing electrical stimulation for seizure control [85], searching through a large transcranial alternating current stimulation (tACS) parameter space based on relative judgment [86], and for predicting optimal DBS parameters using fMRI data for PD patients [87]. The authors in [88] introduced a semi-automated approach to optimize DBS parameters in PD patients for minimizing rigidity and provided preliminary evidence on the utility of using Bayesian optimization in determining optimal DBS parameters. In addition, Bayesian preference learning was used in [89] for identifying personalized optimal stimulation patterns based on the participant's expressed preference for stimulation settings. The authors in [90] introduced a Bayesian adaptive dual control in a computational model of Parkinson's disease to reduce the beta power. In chapter 3, I designed, implemented and clinically evaluated an automated deep brain stimulation programming for tremor suppression in patients with Parkinson's disease and essential tremor [60]. These data-driven optimization strategies have been successfully implemented in various applications of closed-loop neuromodulation systems. However, they share a common underlying assumption regarding the objective function's stationary, implying that its behavior remains relatively constant within the region of interest. This assumption is essential for the acquisition function to effectively estimate regions of high uncertainty and is not applicable to dynamic systems like the cardiovascular system.

Alternatively, machine learning and deep learning techniques can capture complex system behaviors, adapt to changing conditions, and handle nonlinearity and uncertainty effectively. Integration of ML approaches has been widely explored in the literature to detect biomarkers of the disease state. However, they have been mainly used in conjunction with a simple classical control strategies to design iCLON systems. A recent study [91] used a logistic regression used a logistic-regression model to detect freezing of gait and used the model predictions as a control signal. This study combined this ML-based biomarker detection approach combined with a simple decision-table controller [91]. A similar combined approach is used in [92] to design a binary classifiers for extracting patient-specific features from cortical signals and determining when volitional, tremor-evoking movement is occurring to alter stimulation voltage in real time. The control strategy is a simple condition-based controller [92]. A neural network-based design of an on-off adaptive control for Deep Brain Stimulation in movement disorders is presented in [93]. To eliminate the need of having an accurate model of the neural dynamics, a recent study [81] employed a data-driven modeling approach using long short-term memory (LSTM) neural networks. Authors in [81]demonstrated the utility of an LSTM model by generating synthetic data from the computational cardiac model introduced in [94] and developed an MPC controller to regulate heart rate (HR) and mean arterial pressure (MAP). However, this approach still requires access to a substantial amount of experimental data to sufficiently cover the stimulation parameters space annd accurately model the effect of VNS parameters on HR and MAP, which is not practical due to the limitations in experimental data collection.

RL, in particular, provides a framework for learning optimal control policies through interactions with the nervous system, making it well-suited for cases where classical control approaches are not efficient or lack predefined control laws. These data-driven approaches have the potential to enhance control strategies, making them more versatile and robust across a wide range of neuromodulation applications. Despite the recent success of RL in many applications, its utility in the context of closed-loop neuromodulation has not been extensively explored. A previous study presented a simulation environment, called the ContinuousFlappyBird, to resemble the dynamic environment in PD patients for early development of adaptive DBS systems using RL approaches [95]. Authors in [96] presented a closed-loop DBS technique for rehabilitation in PD patients in which a RL algorithm is used to adaptively tune the parameters of a PID controller. In chapter 4, I presented a novel interactive AI framework using RL which provides an automated data-driven approach to design closed-loop VNS control policies with minimal assumptions and without the need for prior knowledge about the underlying physiological dynamics of the cardiovascular system.

2.6 The need for developing research platforms to enable research and development of implantable iCLON systems

Designing intelligent closed-loop neuromodulation (iCLON) control strategies involves creating a modular design capable of seamlessly integrating with in-vivo experimental setups and computational approaches from multiple domains. Most of the current implantable neuromodulation devices offer limited computational capabilities and does not support designing adaptive stimulation policies to account for the underlying complex and varying dynamics of the nervous system. recent advances in AI may enable designing intelligent neuromodulation systems, that are able to learn and optimize neuromodulation control strategies autonomously, via closed-loop interaction with the nervous system. However, the computational complexity of these class of algorithms cannot be met with general-purpose embedded systems and there is a need for specialized, yet programmable, hardware. In chapter 5, I introduced an open-source platform, dubbed Neuroweaver, for end-to-end designing, prototyping and deploying iCLON algorithms without the complexities of translating AI algorithms to implementation.

Chapter 3

Automated deep brain stimulation programming with safety constraints for tremor suppression in patients with Parkinson's disease and essential tremor[1]

3.1 Introduction

Deep brain stimulation (DBS) surgery has become a standard treatment for neurological disorders such as Parkinson's disease (PD) and essential tremor (ET), to ameliorate tremor when medications are insufficient. DBS significantly improves both symptoms and quality of life, however to achieve therapeutic benefit, stimulation often requires time consuming programming by an expert [7]. DBS devices enable considerable customization of stimulation parameters including contact configuration (cath-

[1] © IOP Publishing. Reproduced with permission. All rights reserved.

ode and anode selections), current amplitude, pulse width and frequency allowing for customization of stimulation to account for variations in electrode placement, differences in local anatomy, symptom type, and severity [97]. Typically, a programming session involves a trial-and-error evaluation of therapeutic response (clinical benefit and unwanted side effects) at numerous stimulation settings. This is often performed over several sessions, which can be a challenge for patients who live far away from specialty care. In addition, evaluation of clinical response can be challenging given the subjective nature of visual observation to determine if tremor and other motor symptoms are responding to stimulation. Therefore, designing objective markers of therapeutic response to DBS is needed. Recently DBS device innovations (8-contact electrodes, current fractionation, widened pulse width range and anodic stimulation) have significantly expanded the parameter space making programming even more complex [12]. These limitations suggest a need for an automated and patient-specific DBS programming framework that facilitates DBS programming without requiring an expert clinician.

We previously presented an automated DBS programming framework using an exhaustive grid search-based sampling strategy that mimics heuristic clinical DBS programming [14]. Although this automated framework was effective in programming DBS devices, sampling similar settings for all patients using a grid-search sampling method is a suboptimal approach since each patient responds to DBS differently [98]. Moreover, the number of required samples to converge to an optimal DBS setting was high, and we hypothesize that more advanced sampling and optimization techniques could improve the process. Two recent studies have assessed the efficacy of a closed-loop optimization algorithm for DBS programming using external motion sensor-based motor assessments in patients with PD [99, 100]. The details of the proprietary algorithm have not been published and the system required presence of a clinician to manually change the DBS settings based on algorithm recommendations and if side effects occur. In addition, the algorithm-based DBS suggestions have only been tested with monopolar stimulation settings which may be suboptimal for some patients.

In this study, we combined the knowledge from clinical decision-making strategies with Bayesian optimization, to develop an automated real-time DBS programming framework that enables sample-efficient and patient-specific DBS programming to simultaneously ameliorate tremor and avoid side effects. Bayesian optimization has been successfully adopted for developing closed-loop neuromodulation applications in the context of optimizing the experimental design with closed-loop real-time functional magnetic resonance imaging (fMRI) [84], optimizing electrical stimulation for seizure control [85], searching through a large transcranial alternating current stimulation (tACS) parameter space based on relative judgment [86], and for predicting optimal DBS parameters using fMRI data for PD patients [87]. The authors in [88] introduced a semi-automated approach to optimize DBS parameters in PD patients for minimizing rigidity and provided preliminary evidence on the utility of using Bayesian optimization in determining optimal DBS parameters. In addition, Bayesian preference learning was used in [89] for identifying personalized optimal stimulation patterns based on the participant's expressed preference for stimulation settings. The authors in [90] introduced a Bayesian adaptive dual control in a computational model of Parkinson's disease to reduce the beta power.

In addition, several recent studies explored DBS programming in closed-loop paradigm using tremor measurements. The authors in [101] investigated the use of isostable amplitude using computational models of ET patients to optimize DBS. Another study modelled the dynamics of patient tremor and their phase response curve to investigate the effect of phase-locked DBS in tremor suppression and proposed a closed-loop phase tracking stimulation regimens [21]. Several studies explored the utility of surface electromyography (EMG) and acceleration in tremor prediction and the design of a simple on-off DBS controller in closed-loop [102, 103, 104, 105, 106].

The authors in [107] used electrocorticography for sensing movement intention alongside with worn accelerometers and EMG sensors to deliver responsive closed-loop stimulation to treat tremor in a closed-loop fashion.

To the best of our knowledge Bayesian optimization with safety constraints has not been tested in the context of clinical DBS programming for tremor suppression. We hypothesized that implementation of a Bayesian optimization [83] algorithm for DBS programming for tremor suppression would have high efficiency (fewer samples than the grid search) and that safe programming (avoidance of uncomfortable side effects) can be achieved in an automated system using safe Bayesian optimization algorithm. We further provided clinical assessment of the closed-loop DBS programming framework in a cohort of 15 PD and ET patients [108, 109].

3.2 Patient selection criteria and clinical experiment procedure

Patients with ET or tremor-dominant PD were recruited from a large academic movement disorders clinic. Patients had been implanted with Medtronic Activa neurostimulator systems for at least six months and had DBS settings optimized during standard clinical programming visits prior to study enrollment. The Emory University Institutional Review Board (IRB) approved the study and all patients signed written informed consent.

At the beginning of each experimental session, DBS was turned off, and clinical setting stimulation effects washed out for 10 minutes. Patients were asked to hold their tremor or PD medications for at least 12 hours prior to testing to avoid medication fluctuations during optimization. DBS optimization was performed in one lead for each patient, contralateral to the arm with more severe tremor. The non-tested lead was kept off during optimization unless rest tremor was too bothersome to sit comfortably. DBS implantable pulse generator (IPG) was reprogrammed to create 4 groups (4 contact configurations with amplitude control) that the optimization algorithm could explore (at the beginning of the experiment, the four groups are set as monopolar contact configuration, where the IPG case is set as an anode and each of the four contacts are set as a cathode, and if 'advanced' stimulation was needed based on algorithm's decision scheme, we set the four groups as bipolar or multipolar contact configurations). Stimulation pulse width and frequency were not changed during optimization and were the same as the patient's clinical setting. For phase I experiments, clinician determined maximum allowable amplitude for each stimulation group that could be safely sampled by the automated optimization algorithm (to prevent the automated system from inducing severe side effects). In phase II experiments, the maximum allowable amplitude was set to 5V for all groups, and the safe Bayesian optimization algorithm was utilized to avoid inducing severe side effects (additional safety feature allowed rapid stimulation shut off if necessary).

During optimization, two standard clinical motor tasks were performed at each stimulation setting to assess tremor, depending on each patient's tremor profile (rest, arms extended, arms flexed, or finger-nose motion). A commercial smartwatch (LG-W100) worn on patient's wrist was used to determine a tremor score using a previously validated classifier [14] and this score was used as input into the optimization algorithm (Figure 3.1). A clinician blinded to the stimulation setting also scored the tremor during optimization using Fahn-Tolosa-Marín rating scale (FTM) [110] to further assess previously validated tremor classifier [4]. At each stimulation setting, the patient reported stimulation-induced side effects (typically tingling or muscle contractions in the face, arm or leg) and rated them on a scale from 0-3 (none, transient or mild, moderate, severe). At the end of the optimization session, DBS IPG was set to the best 'automated setting' and a clinical tremor exam and objective watch tremor measurement were performed after a 5-minute wash-in (examiner and patient

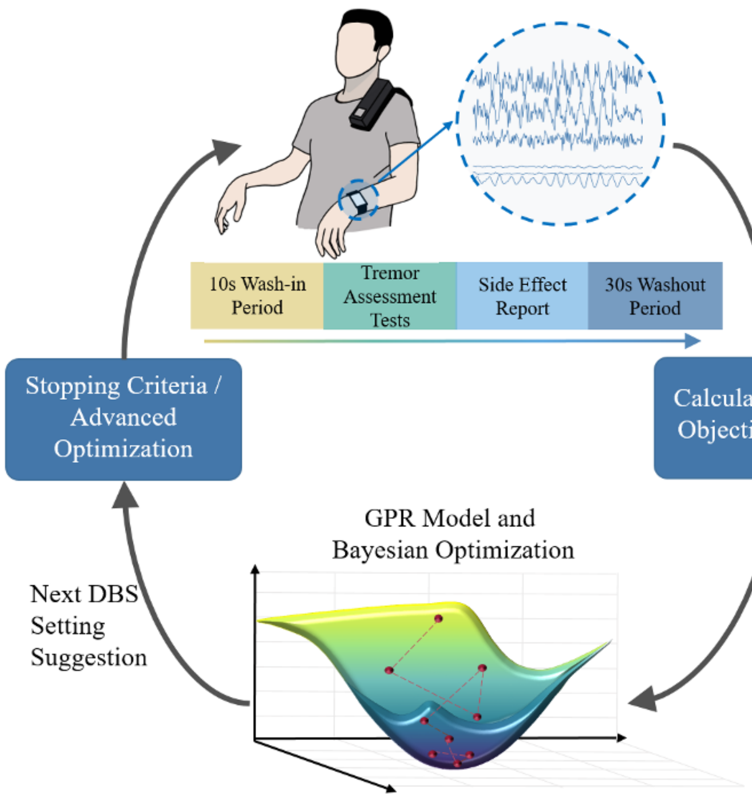

10s Wash-in Period
Tremor Assessment Tests
Side Effect Report
30s Washout Period
IMU Data, Side Effect Reports
Calculate the target Objective function
GPR Model and Bayesian Optimization
Next DBS Setting Suggestion
Stopping Criteria / Advanced Optimization

Figure 3.1: Overview of the automated DBS optimization framework for tremor programming. After performing the initial baseline tremor evaluation tests without stimulation, at each iteration, the software automatically sets the next DBS setting to be tested followed by 10 seconds wash-in period, followed by tremor evaluation tests each for 10 seconds. The recorded IMU data and side-effect reports are used to update the surrogate GPR model and optimizer suggests the next best sample to be tested. Before evaluating the next suggested DBS setting, the stopping criteria module determines whether the optimum has been found or advanced stimulation is needed.

were aware of the stimulation condition). Clinical tremor exam for both PD and ET patients was a subset of FTM scale and included rest, postural, and action arm and leg tremor contralateral to DBS lead, handwriting (if dominant hand tested) and spiral and line drawings. DBS IPG was then set to patient's 'clinical setting' and after another 5-minute wash-in period, tremor was reassessed by exam and watch (for the first two patients, tremor assessment at clinical setting was done at the beginning of the visit, but protocol was changed for subsequent subjects to facilitate direct comparison between automated and clinical settings).

3.3 Automated DBS programming framework: software design

We performed these automated DBS programming experiments using a custom software application developed for a Windows PC (Figure 3.2). This application collected patient-reported side effect data and inertial measurement unit (IMU) data from the smartwatch. Side effects were entered through a user interface and constituted a total of 13 common acute side effects of DBS therapy and included reports of magnitude ("0: none,", "1: mild", "2: moderate", "3: severe"), type ("paraesthesia", "muscle spasm", "speech", "vision", "dizziness", "dyskinesia"), and body location ("head", "arm", "leg", "torso"). IMU data, constituting 3 dimensions each of accelerometer and gyroscope data for a total of 6 channels, was streamed to the study PC via Bluetooth at 100Hz for processing and feature extraction. All data was logged on receipt in CSV format, as were DBS stimulation parameters.

The software application made use of a previously developed C# application programming interface (API) [111, 112, 113] for interfacing with the Nexus-D, a Medtronic research communication bridge that allows an application to update IPG stimulation parameters. IMU data collection and interfacing with the DBS device through the Nexus-D was conducted through this application. Time-series IMU data review and side effect inputs were conducted using an application written in Python to take advantage of superior data processing and capacity to deploy advanced machine learning and optimization techniques. Lab streaming layer, a publicly available library for cross-platform port handling and communication of time-series data in research applications, formed the interface between these applications.

Using this combination of side effect data and tremor severity estimate, a quantitative therapeutic value was derived for each DBS setting using Python. The details of this derivation are described in more detail in the following sections. Updated

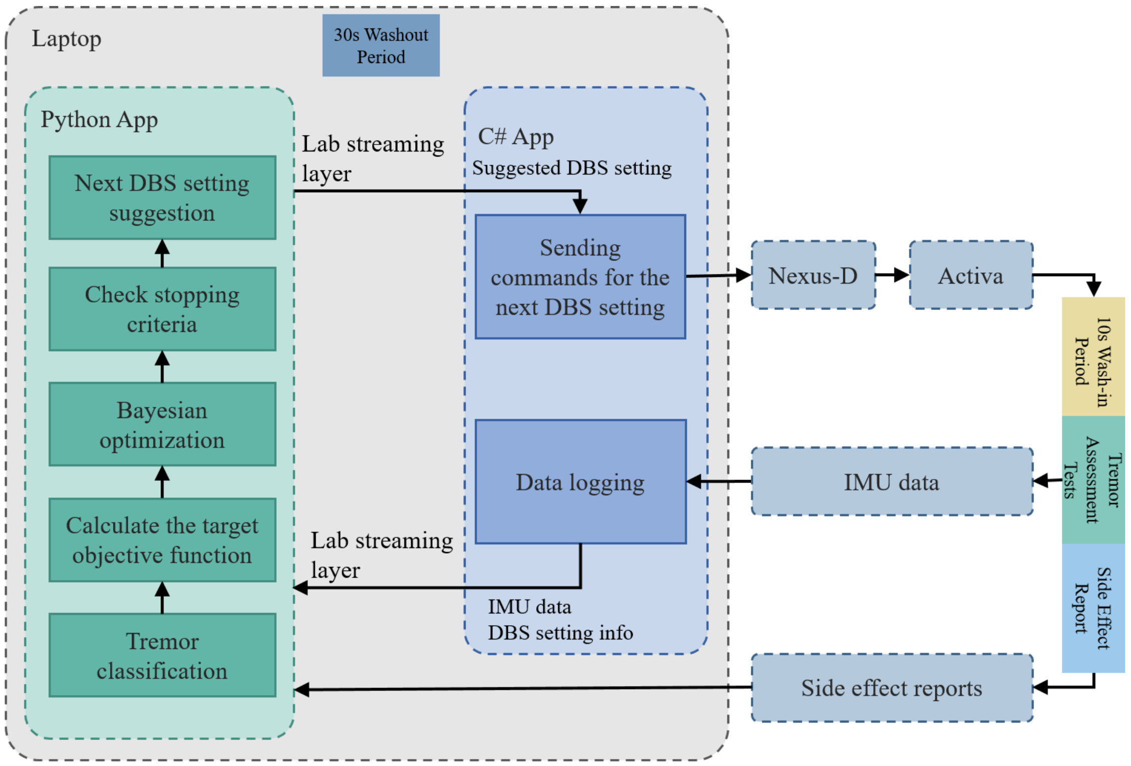


Figure 3.2: A detailed schematic demonstrating the software design of the automated DBS programming system. The software application receives IMU data over a Bluetooth connection from the smartwatch, as well as side effects reported by the patient through a graphical user interface and send the information to the Python section of the application. The calculation of the objective measure (surrogate function) and choice of the next DBS setting (acquisition function) are handled within the Python section. The C# software application receives the stimulation settings from the Python application and sends stimulation commands to the Nexus-D, which communicates with patient's implanted Activa IPG.

settings were forwarded to the C# application. Both the therapeutic value of the preceding DBS settings and the recommended next settings were recorded at this point. Following a brief final review and safety check, these settings were then communicated via USB connection to the Nexus-D. The C# application was also capable of manual override in case of emergency.

3.4 GPR modelling of the effect of DBS settings using a quantified objective measure

Gaussian process regression [114] is a nonparametric, Bayesian regression approach, which is well-suited for small datasets and provides the measurement uncertainty for the predictions. In this study, the patient-specific GPR models used the Matern kernel function [83] and were trained using the cumulatively collected samples $D = \{(x_i, y_i) | i = 1, \ldots, n\}$ from each patient, where x_i were stimulation parameters, i.e., stimulation amplitude and stimulation contact configuration and y_i represented the corresponding combined objective measure as defined in equation (4). A GP is a nonparametric model that is fully characterized by its mean and covariance function as following

$$f(x) \sim GP(m(x), K(x, x^{'})) \tag{3.1}$$

where we can define the mean function as $m(x) = E[f(x)]$ and the covariance function as $K(x, x^{'}) = E[(f(x) - m(x))(f(x^{'}) - m(x^{'}))]$. Here, we used Matern kernel function as in

$$K_{MATERN3}(x, x^{'}) = \sigma_f^2 exp(-\sqrt{3}r(1 + \sqrt{3}r)) + \sigma_n^2 I, \tag{3.2}$$

where $r^2 = (x - x^{'})^T \Lambda (x - x^{'})$ and Λ is the diagonal matrix of squared length scales. The output variance σ_f^2, the length-scales, and the noise variance σ_n^2 are hyperparameters of the covariance function. By incorporating the knowledge from the training data (prior distribution in equation 3.1), we can make predictions at any new test point (x_*, f_*), where $f_* = f(x_*)$. The predictive conditional distribution of f_* given the training data and test input is calculated as in

$$f_* | X, y, X_* \sim N(\bar{f}_*, cov(f_*)), \tag{3.3}$$

where $\bar{f}_* = E[f_*|X, y, X_*] = K(X_*, X)[K(X,X) + \sigma_n^2 I]^{-1}y$, $cov(f_*) = K(X_*, X_*) - K(X_*, X)[K(X,X) + \sigma_n^2 I]^{-1}K(X, X_*)$, and σ_n^2 denotes the noise variance.

This GPR modelling technique is used as a surrogate model for Bayesian optimization described in the following section. We employed the GPflow library [115] for implementing GPR models. We defined a combined quantitative measure of clinical efficacy consisting of a quantitative objective tremor score, measured by the smartwatch IMU and patient-reported side-effects, with the goal of maximizing tremor improvement and minimizing side-effects. The quantification measure of DBS setting value, J_{DBS_i}, is calculated based on the results of the tremor assessment tests while the patient's IPG was active in a particular DBS setting DBS_i as in

$$J_{DBS_i} = J_{tremor_i} + J_{SE} \tag{3.4}$$

Each DBS setting DBS_i is evaluated based on the tremor score improvement, J_{tremor_i}, which is a baseline subtracted tremor severity score as in equation 3.5, and J_{SE} which is the patient-reported side effect severity scores defined as in equation 3.6.

A predictive model of clinical tremor assessment from IMU data was trained and validated in a previous study [14], where features from accelerometer and gyroscope data were used to train an ordinal multinomial logistic regression classifier based on the neurologist's provided tremor ratings [14]. To directly compare the performance of our Bayesian automated DBS optimization framework with the state-of-the-art automated DBS programming framework introduced in [14], which uses a grid-based search approach, we used the same classifier [14] in order to avoid introducing new parameters to the system. The output of this classifier was used to calculate the tremor score improvement, J_{tremor_i}, as in equation 3.5.

The term $(tremor_{DBS_i})$ in equation 3.5 is the average watch tremor severity score (predicted from the classifier) over the selected tasks while the patient's IPG was active with the ith DBS setting DBS_i. The term $tremor_0$ is the average watch tremor severity scores (predicted from the classifier) over all selected tasks with inactive IPG that reflects patients' baseline tremor score obtained at the beginning of optimization session.

$$J_{tremor_i} = tremor_{DBS_i} - tremor_0 \tag{3.5}$$

The magnitude of the overall baseline subtracted tremor score (J_{tremor_i}) shows the level of change in the average tremor score comparing to the baseline and a negative sign reflects tremor improvement compared to the baseline. The lower the J_{tremor_i}, the more clinical benefit the DBS_i setting provides.

In addition, each DBS setting DBS_i is penalized by the patient-reported side effect severity scores. We defined the term J_{SE} in equation 3.4 as follows based on patients' reports.

$$J_{SE} = \begin{cases} 0 & \text{if no SE} \\ 1 & \text{if mild SE} \\ 4 & \text{if moderate SE} \\ inf(5 \text{ in practice}) & \text{if severe SE} \end{cases} \tag{3.6}$$

Watch tremor severity scores are on a scale of 0 to 4, so the J_{tremor_i} term could get any value in the $[-4, 4]$ interval depending on the baseline score and the tremor score in the DBS_i setting. Specifically, $J_{tremor_i} = 0$ means no tremor improvement, $J_{tremor_i} < 0$ reflects a tremor score improvements compared to baseline, and $J_{tremor_i} > 0$ corresponds to cases where the watch tremor score is worse than the watch tremor score at baseline. If the patient experiences some level of side effect, we penalize J_{tremor_i} by adding a positive value to it. If the side effect is mild, we penalize it by 1, meaning that a DBS setting with a score of 1 for tremor improvement with mild

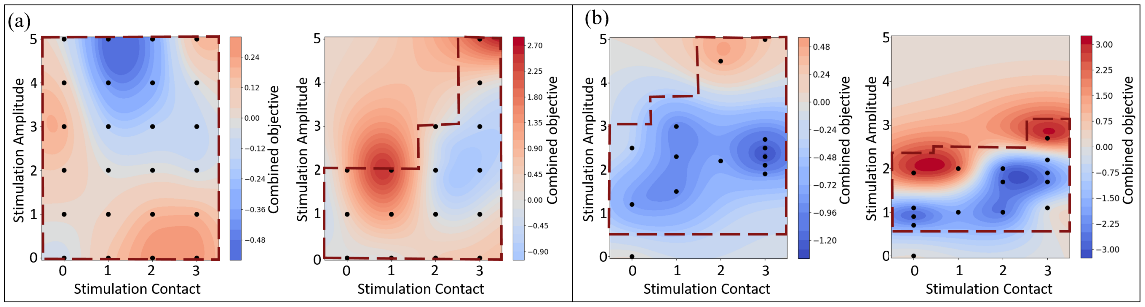


Figure 3.3: GPR model mean surface of the combined objective measure (including baseline-subtracted watch tremor score and side effect score) varies across patients. (a) Mean surfaces for two patients with grid-search sampling strategy from a prior study [4]. The sampling resolution is 1V amplitude increments. (b) Mean surfaces for two patients from the current study with sampling using Bayesian optimization that evaluates more samples in areas with greater chance of tremor improvement and with a finer resolution (0.2V amplitude increments). The surfaces are color-coded with the value of the combined objective measure where blue shows negative objective values reflecting tremor improvement compared to baseline either without or with mild side effect and red shows positive values reflecting that DBS settings are not effective or side effects are pronounced. The black circles represent sampled DBS settings during the automated DBS optimization. The red dashed lines show the clinician-defined safe exploration boundaries of the parameter space.

side effect will have a total score 0 which is similar to a setting with no improvement and no side effects. If the side effect is moderate, we penalized it by 4 because any amount of tremor improvement with moderate SE will be considered as untenable for clinical use (resulting in a non-negative J_{DBS_i} score). If the SE is severe, we penalize it even more to prevent the optimizer from testing that area again.

The quantified objective measure in equation 3.4 was calculated for each DBS setting tested on the patients and the cumulatively collected samples were used to train a GPR model that models patients' response to DBS. The mean surfaces of GPR models of the combined objective measure defined in equation 3.4 capture both the effect of baseline-subtracted watch tremor score and the side effect severity scores simultaneously and justifies the use of the combined objective measure for Bayesian optimization to ameliorate tremor while avoiding side effects (Figure 3.3). Furthermore, the mean surface of the GPR model varies across patients (Figure 3.3). Specifically, the general shape of the surface, the location of the optima, and the maximum tolerable boundaries vary between individuals reflecting their unique response to stimulation profile. These variations are partly due to disease type and tremor severity, DBS lead position (which varies even for patients with the same target nucleus), and individual anatomy. This subject variability necessitates designing a patient-specific DBS optimization framework with an adaptive sampling strategy, while still remaining sample-efficient. Due to variability in patients' responses, a grid search-based approach with 1V amplitude increments for each contact configuration that was utilized in a previous similar study [14] was hypothesized to be inefficient (Figure 3.3.a). The Bayesian optimization which we have utilized in this study evaluates more samples in areas with greater chance of tremor improvement and searches for the optimal point with a finer resolution of 0.2V increments in amplitude for each contact configuration (Figure 3.3.b).

3.5 DBS programming algorithms

3.5.1 Bayesian optimization

We formulated the automated DBS programming as a global optimization problem over the stimulation parameter space, D, as in:

$$\min_{x \in D} f(x), \tag{3.7}$$

where $f(x)$ was the objective measure that represented the desired clinical outcome, and D is the two-dimensional space of the DBS parameters including stimulation amplitudes and contact configurations. One of the main challenges in solving this problem was that the functional relationship between the DBS parameter space and clinical outcome was not known. Bayesian optimization is a non-parametric global optimization approach that is suitable for optimizing black-box objective functions that are unknown or expensive to evaluate.

The objective function $f(x)$,which represents a mapping between the DBS parameters and the clinical outcome, is unknown and does not have a closed form. Although $f(x)$ is unknown at every $x \in D$, we can observe its measurements at sampled DBS settings (the objective measure as described in equation (3.4) is a measurement of $f(x)$ at the suggested DBS settings DBS_i by the optimizer). Bayesian optimization proceeds by maintaining a probabilistic belief over $f(x)$ by building a GPR surrogate model as described in section 3.4 using the cumulatively collected data from each patient. The GPR model prescribe a prior belief over the possible objective functions given the cumulatively collected data. In a previous study we characterized the functional relationship between the clinical outcome and DBS parameters using the GPR modeling approach [98].

The Bayesian optimization algorithm is based on a sequential decision-making process to search for the optimal stimulation parameters in two steps. First, it builds a surrogate probabilistic model of the latent objective function $f(x)$ based on the available data at each iteration and sequentially retrain the model as more data is observed. Second, it proposes the next DBS setting to be evaluated by optimizing a surrogate-dependent acquisition function. The acquisition function assesses the utility and the informativeness of the candidate points for the next evaluation of the objective measure $(f(x))$ by leveraging the uncertainty in the posterior to guide exploration [83].

During the burn-in phase of Bayesian optimization, the objective function was evaluated at predefined stimulation settings in a randomized order (at 40% and 80% of maximum amplitude for each contact configuration). Then, a GPR prior based on these initial evaluations of the burn-in phase was employed. Thereafter, a new stimulation setting was sequentially selected by optimizing the acquisition function to be evaluated at next iteration. At each iteration, we augmented the dataset, updated the surrogate GPR prior, and optimized the updated surrogate-dependent acquisition function to suggest the next samples to be tested in the patient until convergence. Our stopping criteria is explained in section 2.6. For the first round of experiments, with amplitude limits determined by a clinician, we used the expected improvement (EI) acquisition function [116] which automatically balanced exploration versus exploitation. The EI acquisition function calculates the expectation of improvement over the current best observation with respect to the predictive distribution of the surrogate model and is defined as in:

$$EI(x) = E[max(f_{min} - m(x), 0)] = (f_{min} - m(x))\Phi(z) + \sigma(x)\phi(z) \tag{3.8}$$

,where $\phi(.)$ and $\Phi(.)$ are the standard normal density and distribution functions, respectively. In equation (3.8), $z = \frac{(f_{min}-m(x))}{\sigma(x})$, $m(x)$ is the predictive mean and $\sigma(x)$ is the predictive standard deviation of a point $x \in D$ and f_{min} is the optimum observed value. We implemented the EI acquisition function using the GPflowOpt library [117].

The global optimization problem defined in equation (3.7) is straightforward to solve using Bayesian optimization algorithm if the parameter space is fully defined. To show the feasibility of Bayesian optimization as the core of the automated DBS optimization framework during the first phase of the experiments, the maximum amplitude for each contact configuration is defined at the beginning of the experiment by the expert neurologist and will stay fixed during the experiment as shown in red dashed boundaries in Figure 3.4. The minimum exploration boundary of stimulation amplitude in the parameter space is set to $0.5V$ (that is the dashed horizontal red line at $0.5V$ in Figure 3.4); meaning that the optimizer will not explore the effect of DBS settings with amplitudes smaller than $0.5V$ .With more samples being collected at each iteration, the underlying GPR model gets updated. At each iteration, the updated GPR model is used to build the acquisition function and get the next DBS

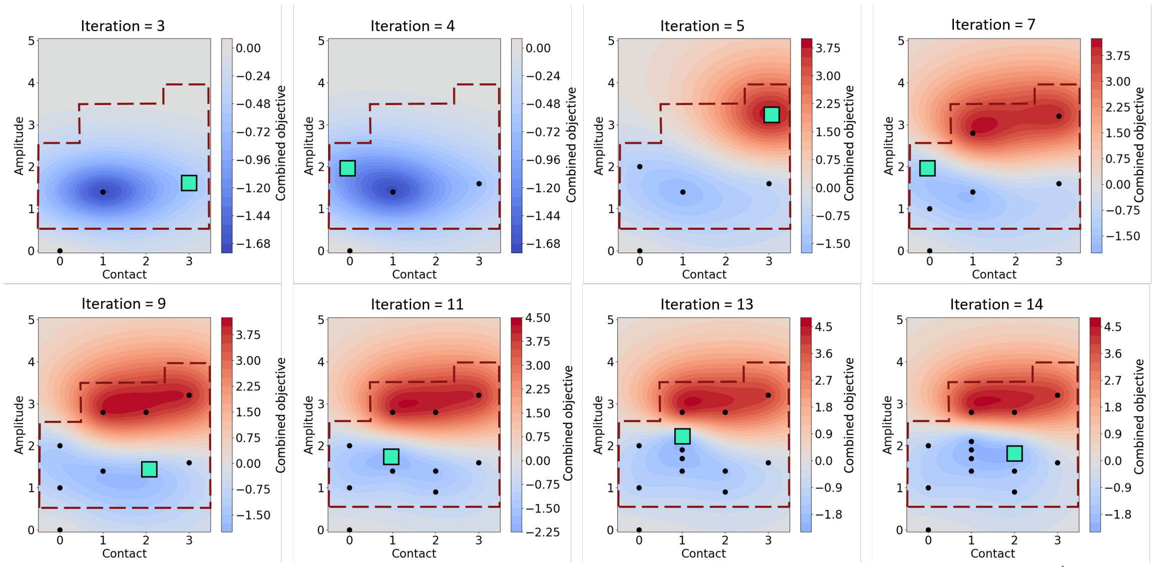


Figure 3.4: Example of patient-specific adaptive sampling of Bayesian optimization (patient 02). Each panel shows the mean surface of the GPR model that updates after each iteration. The value of the combined objective measure is color-coded. The dashed dark red lines demonstrate the clinician-defined maximum tolerable exploration boundaries. The black circles show the previously collected samples and the green square show the sample being tested at each iteration. The black circle outside the red dashed lines at (0,0) demonstrates the baseline, where the patient's IPG was inactive. The sample suggestions are automated by the DBS optimization framework. Samples are more densely distributed around the more promising regions of the parameter space (more tremor improvement with fewer side effects). This adaptive behavior of the DBS optimization framework makes it patient-specific; that is the samples are adaptively suggested based on the patient's response at previous iterations.

setting suggestions to be evaluated at the next iteration. DBS settings are adaptively sampled during the DBS optimization session based on patient's response in a patient-specific manner (Figure 3.4).

Since the objective function $f(x)$ is unknown, our Bayesian optimization algorithm does not assume convexity. As a result, if multiple optima are found, the setting with the lowest amplitude was selected as the optimum automated setting to ensure lower power is used when possible.

3.5.2 Safe Bayesian optimization

The DBS optimization problem is safety-critical, as there is a safety constraint for each DBS setting. The safety constraints are defined by the side effects that patients may experience for each DBS setting. In the second phase of the experiments, instead of using the clinician-defined safe exploration boundaries, we modified the problem statement as a global optimization problem with safety constraint as in:

$$\min_{x \in D} f(x);\ \textit{such that}\ g_i(x) < 3\ \textit{for}\ i = 1, 2, 3, 4, \tag{3.9}$$

where $g_i(x)$ is the magnitude of patient-reported side effect for each DBS setting x and i is the contact configuration number. Since the safety boundaries for each contact configuration are different, a separate GP model was trained for each contact configuration based on the patient-reported side-effects. To solve the above constraint optimization problem, we used safe Bayesian optimization [118, 119] which is an extension of regular Bayesian optimization. We used the GPflow [115] and GPflowOpt [117] libraries for our implementation of the algorithm.

Safe Bayesian optimization combines a GP model of the safety constraints with discretization of the parameter space to define a set of DBS parameters S_n, called the safe set, with a high probability to satisfy the safety constraints [118]. The safe set was defined by the upper bound of the safety GP models ($g_i(x)$) and contained the points where the GP upper bound was smaller than the safety threshold. Our parameter space for the DBS programming was discrete, with four contact configurations and stimulation amplitudes with $0.2V$ increments. The safe Bayesian optimization algorithm defined two sets of parameters within the safe set called potential minimizers and expanders. The set $M_n \subseteq S_n$ contains potential minimizers that is the parameters that could potentially obtain the minimum within the current safe set. The minimizers set (M_n) was defined by the mean and confidence interval of the GPR model of the objective measure, $f(x)$, and contained the subset of safe parameters in which the lower confidence bound of the GPR model was lower than the upper confidence bound at the best measurement at each iteration. The expander set $G_n \subseteq S_n$ is considered to be the points that if tested, their measurements would lead to values in the lower confidence bound and hence potentially expand the safe set [119]. In this paper, we directly used the confidence bounds of the GPR models to define the aforementioned sets, which is mainly used in practice with limited prior knowledge of the underlying objective and safety models (considers the Lipschitz constant to be infinity). This modified version has been shown to be more aggressive in expanding the safe set [119]. Hence, to further ensure safety, we considered the maximum allowable amplitude expansions based on the maximum severity of the patient-reported side effects for each contact configuration. The safe set would only expand by the minimum of the safe set suggested by the safe Bayesian optimization algorithm and the maximum allowable constant defined as follows. In general, the lesser the reported side effect, the more the parameter space could expand. In other words, the expanded space gets closer to the safety boundaries as the patient begins to experience mild side effects by increasing the stimulation amplitude. Hence, the expansion should be done with more caution. Another consideration in selection of the constant was that patients were more likely to experience more severe side effects during the monopolar stimulation settings than advanced stimulation settings. Therefore, the constant during the monopolar stimulation was selected to be smaller. If the patient experienced no side effects for a particular contact configuration, then the constant was set to $1V$ for monopolar stimulation and $1.5V$ for the advanced stimulation settings, respectively; these amplitudes represent the maximum allowed jump in the stimulation amplitude for each contact configuration. If the maximum side effect severity level was mild, then the constant was set to $0.5V$ for monopolar stimulation and $0V$ for advanced stimulation settings (zero because we aimed for no side effects at optimal stimulation setting during advanced stimulation; during monopolar stimulation mild side effects were tolerated during optimization as results were informative for selection of advanced stimulation configurations). If the patient experienced moderate or severe side effects either during the monopolar or advanced stimulation, the constant was $0V$ and the expansion of the stimulation amplitude was stopped for the corresponding contact configuration.

Safe Bayesian optimization starts with evaluating a set of initial parameters that is known to be safe, which defines the initial safe parameter space. Here, we started with testing 1V for each of the four contact configurations as the initial safe set of parameters (S_n). The initial evaluations were used to train GP models on the safety constraints which were used to safely expand the parameter space at each iteration. After the safe expansion of the parameter space, the next DBS settings to be tested were suggested by optimizing the surrogate-dependent acquisition function at each iteration. As mentioned above, we used GPR modeling technique to model the constraint/safety functions. A common assumption in training GPR models is that a GP prior is zero-mean. However, this assumption did not apply to the DBS programming problem since the side effect severity report was a monotonically increasing function of stimulation amplitude. Hence, we fitted a second-degree polynomial function of the collected samples and used that as the prior mean of the safety GP models. In addition, in phase II of the experiments, we changed the acquisition function to min-value entropy search [120] as it was more sample efficient and had a more exploratory behavior that is required given the nature of Safe Bayesian optimization algorithm with gradual expansion of the parameter space.

A visual representation of the sampling behavior and safe exploration boundary expansion of the automated framework is presented for some sample iterations in Figure 3.5 for patient 14. Each figure shows the mean surface of the GPR model that gets updated as we collect more data at each iteration. Note that the safe boundaries shown in dashed red line are updated after evaluating the suggested DBS setting at each iteration.

In both phases of the experiments, we considered some modifications of the reg-

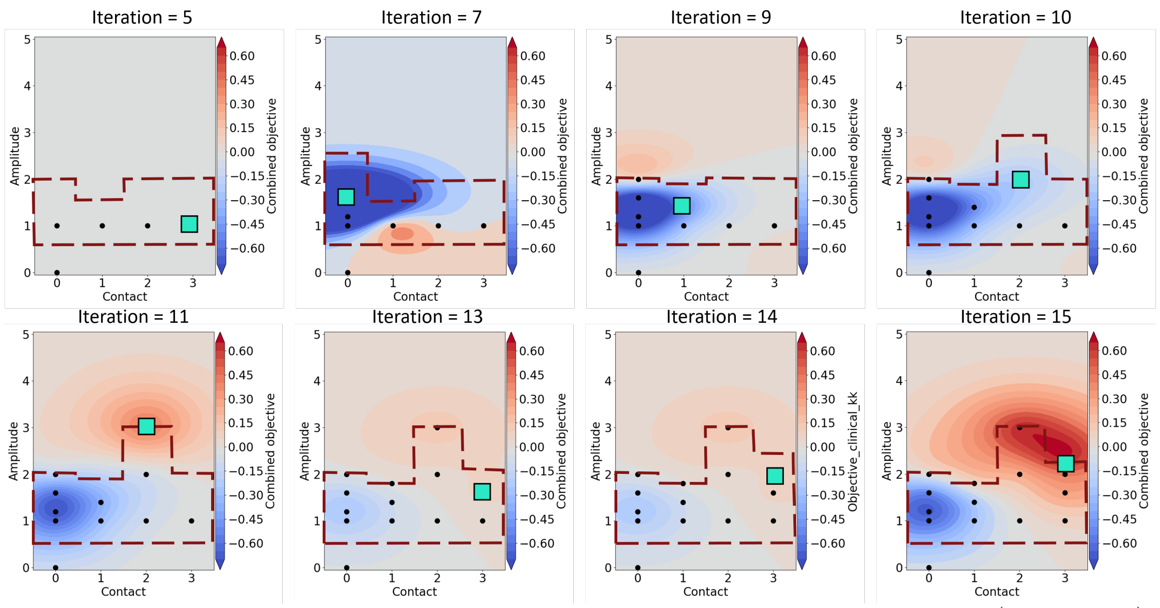


Figure 3.5: Safe Bayesian optimization during phase II of the experiments (patient 14). Selected iterations during monopolar stimulation. The mean surface of the GPR model and safe stimulation exploration boundaries (dashed lines) update as more data are collected at each iteration. The value of the combined objective measure is color-coded. The black circles represent collected samples and the green square is the current sample being tested at each iteration. The black circles outside the red dashed lines at (0,0) demonstrates the baseline, where the patient's IPG was inactive.

ular (safe) Bayesian optimization algorithm to account for the requirements of the automated DBS programming in practice. First, we considered a discrete parameter space including four contact configurations and stimulation amplitude with $0.2V$. The optimization of the acquisition function is performed by evaluating the acquisition function at every setting in the parameter space at each iteration and the next best sample is selected using the selection of the best strategy. One challenge in designing Bayesian optimization in discrete parameter spaces is the suggestion of repeated samples. In order to avoid suggesting repeated samples, we used a rank and select strategy; that is if the best sample suggested by optimizing the acquisition function is already sampled, the next best sample will be suggested.

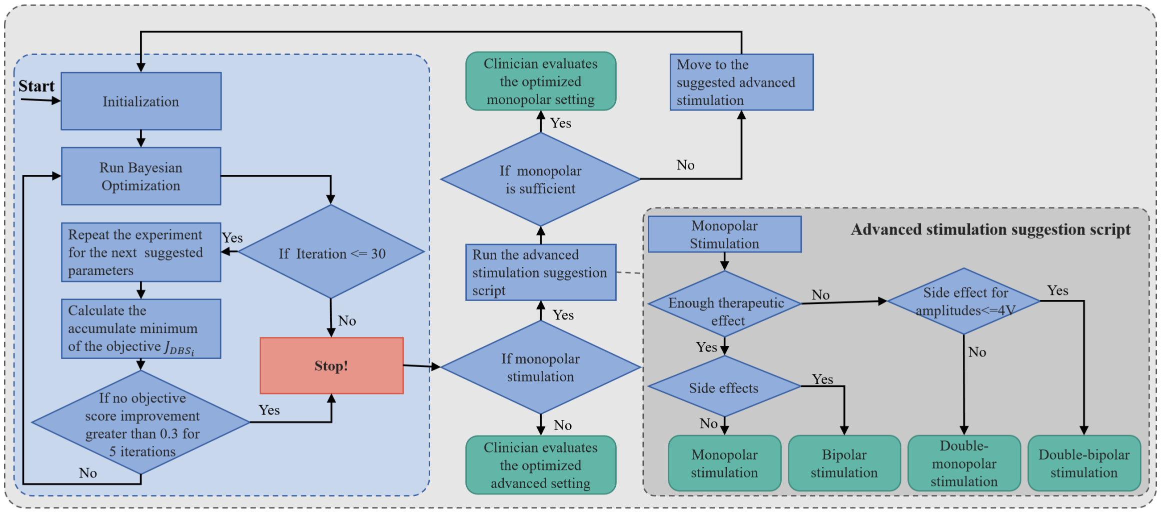

Start
Initialization
Run Bayesian Optimization
If Iteration <= 30
Yes
No
Repeat the experiment for the next suggested parameters
Calculate the accumulate minimum of the objective J_{DBS_i}
If no objective score improvement greater than 0.3 for 5 iterations
Yes
No
Stop!
If monopolar stimulation
Yes
No
Clinician evaluates the optimized advanced setting
Run the advanced stimulation suggestion script
If monopolar is sufficient
Yes
No
Clinician evaluates the optimized monopolar setting
Move to the suggested advanced stimulation
Advanced stimulation suggestion script
Monopolar Stimulation
Enough therapeutic effect
No
Yes
Side effect for amplitudes<=4V
Yes
No
Side effects
Yes
No
Monopolar stimulation
Bipolar stimulation
Double-monopolar stimulation
Double-bipolar stimulation

Figure 3.6: High-level schematic of the decision-making process of the automated DBS optimization framework. The darker gray area is the schematic of the advanced optimization suggestion algorithm modelled after the clinical decision-making process.

3.6 Stopping criteria and advanced optimization

Our automated programming framework combined prior knowledge from standard clinical DBS programming approaches with Bayesian optimization. Clinical DBS programming is guided by clinical decision-making strategies to maximize stimulation benefit and minimize side effects [121]. During clinical DBS programming sessions, the clinician often performs a monopolar screening, where each of the four electrode contacts are set as cathode and IPG case as anode. Clinicians evaluate the tremor suppression benefits and side effects by incrementally increasing the amplitude for each of the four monopolar contact configurations. If a monopolar configuration leads to a satisfactory tremor suppression without side effects, this setting is chosen for chronic stimulation. If not, contact configuration may be changed to one of the advanced optimization settings (bipolar, double-monopolar, and double-bipolar) based on patients' responses to monopolar stimulation (Figure 3.6).

Similar to the standard clinical approaches, the proposed automated DBS programming framework started with the monopolar screening with four different contact configurations. The monopolar optimization was terminated either once the stopping criteria was satisfied or the predefined budget of maximum 30 iteration was exhausted (to avoid patient fatigue). The stopping criterion was defined as no objective score improvement of 0.3 or greater for five consecutive iterations. The threshold of 0.3 was defined by our expert movement disorder neurologist that reflected a clinically meaningful score improvement.

After completing the monopolar stimulation trials, the advanced stimulation module used the data that was collected during the monopolar stimulation to determine if advanced stimulation was necessary, and which contact configurations should be utilized (Figure 3.6). If there was at least one DBS setting with sufficient therapeutic effect and without any side effect (acceptable monopolar setting), then the monopolar setting with the lowest average tremor score was selected as the optimized setting. Otherwise, if there was sufficient therapeutic effect with the presence of side effects, then bipolar stimulation was suggested. The sufficient therapeutic effect was dependent on the baseline score. If the average baseline tremor score was less than 1, then sufficient therapeutic effect was defined as 50% improvement in watch tremor score. Otherwise, having a watch tremor score of less than 1 for all of the tremor assessment task was considered as having sufficient therapeutic effect. In cases where there was no setting with sufficient therapeutic effect and no side effect for amplitudes less than $4V$, then double-monopolar stimulation was suggested. If there was no setting with sufficient therapeutic effect and side effects at less than $4V$ were present, double-bipolar stimulation was suggested. If no acceptable monopolar setting was identified, the advanced stimulation suggestion script (darker gray area in Figure 3.6) provided four advanced contact configurations based on clinical heuristics. Since the number of possible advanced stimulation settings is high and due to the limitations of the Medtronic research communication bridge, we could only test four contact configurations at each round of automated DBS programming. We used the advanced stimulation suggestion script to use the clinical heuristics and narrow down the number of advanced settings to 4 best settings to be tested (as depicted in the blue box in Figure 3.6). Testing new settings continued until either the stopping criteria was satisfied, or the maximum number of iterations was reached.

3.7 Results

We recruited 15 patients (9 with tremor-dominant PD and 6 with ET) with the average age of 709 years (range $57-85$) to undergo the automated DBS optimization. All ET patients and one PD patient had leads implanted in ventral intermediate nucleus of thalamus (VIM), while the remaining PD patients had leads implanted in subthalamic nucleus (STN). The average time since DBS lead implantation was 5219 months (range $30-105$ months). We evaluated the performance of the automated DBS optimization framework in two phases. In the first phase, the software used the clinician-defined maximum tolerable amplitudes for each contact configuration to define the safe boundaries of the parameter space. The main goal of the first phase of the experiments was to evaluate the performance of the automated framework in finding the optimized settings using Bayesian optimization algorithm. Data from 10 patients (5 PD and 5 ET) were acquired for the first phase (table 3.1). We further expanded the work and used safe Bayesian optimization to gradually expand the parameter space and automatically discover a safe and tolerable parameter thus avoiding severe side effects as reported by the patient. Seven patients (5 PD and 2 ET; 2 from phase 1) underwent the automated DBS optimization in the second phase of experiments (table 3.2).

3.7.1 Quantifying tremor response to stimulation

The automated DBS optimization framework automatically quantifies and calculates the target objective measure that includes tremor scores and side effects (Figure 3.1).

Table 3.1: Clinical characteristics and automated programming experiment outcome during the phase I of the Experiments using the clinician-defined maximum safe boundaries of the parameter space.

Patient ID	Age, Sex	Dx	DBS Target	Time since DBS surgery (months)	Time since selection of clinical setting (months)	Tremor Score* at Baseline	Clinical DBS Setting	Tremor Score* at Clinical Setting	Automated DBS Setting	Tremor Score* at Automated Setting	Number of Tested DBS Settings	Patient Prefer-ence
1	63, M	PD	R STN	34	28	7	C+3-, 3.5V, 90µs, 130Hz	1	C+3-, 2.3V, 90µs, 130Hz	1	Monopolar = 15, Advanced = NA	Same
2	71, F	PD	L STN	42	12	17	2-3+, 4.6V, 90µs, 160Hz	8	C+1-, 1.9V, 90µs, 160Hz	1	Monopolar = 17, Advanced = NA	Same
3	61, M	PD	R STN	63	17	12	C+2-, 3.5V, 60µs, 165Hz	9	C+1-, 0.7V, 60µs, 165Hz	7	Monopolar = 15, Advanced = NA	Automated
4	82, F	ET	L VIM	54	1	24	1-3+, 2.4V, 90µs, 130Hz	3	C+1-, 1.1V, 90µs, 130Hz	7	Monopolar = 15, Advanced = NA	Clinical
5	79, M	ET	R VIM	54	1	22	9-11+, 2.5V, 60µs, 190Hz	11	C+10-, 2.3V, 60µs, 190Hz	6	Monopolar = 14, Advanced = NA	Automated
6	76, M	ET	L VIM	37	11	9	1-3+, 2.5V, 90µs, 170Hz	3	1-2+, 3.5V, 90µs, 170Hz	3	Monopolar = 15, Advanced = 14	Same
7	77, M	ET	L VIM	43	1	18	1-2-0+, 4.6V, 60µs, 170Hz	7	C+1-, 1.1V, 60µs, 170Hz	10	Monopolar = 15, Advanced = NA	Clinical
8	76, F	ET	L VIM	63	1	19	2-3+, 3.6V, 90µs, 160Hz	4	2-1+, 4V, 90µs, 160Hz	4	Monopolar = 15, Advanced = 15	Same
9	69, M	PD	R STN	70	5	12	0-1-2+, 4V, 60µs, 140Hz	5	1-0+, 3.7V, 60µs, 140Hz	7	Monopolar = 15, Advanced = 15	Clinical
10	57, M	PD	R STN	42	1	24	0-3+, 4.7V, 90µs, 130Hz	5	2-1+, 4.7V, 90µs, 130Hz	5	Monopolar = 15, Advanced = 15	Same

* Clinician administered FTM tremor scale subset including rest, postural (extended and flexed), and action arm tremor contralateral to optimized DBS lead, handwriting (if dominant hand tested), and drawings (max score 28-32).

Table 3.2: Clinical characteristics and automated programming experiment outcome phase II with automated discovery of the safe parameter space using safe Bayesian optimization algorithm.

Patient ID	Age, Sex	Dx	DBS Target	Time since DBS surgery (months)	Time since selection of clinical setting (months)	Tremor Score* at Baseline	Clinical DBS Setting	Tremor Score* at Clinical Setting	Automated DBS Setting	Tremor Score* at Automated Setting	Number of Tested DBS Settings	Patient Prefer-ence
11	67, F	ET	R VIM	108	2	16	0+1-3+, 5.2V, 90µs, 150Hz	4	3-2+, 4.8V, 90µs, 150Hz	3	Monopolar = 13, Advanced = 13	Automated
10	57, M	PD	R STN	45	3	28	0-3+, 4.7V, 90µs, 130Hz	7	2-3+, 4.8V, 90µs, 130Hz	6	Monopolar = 17, Advanced = 13	Same
12	60, M	PD	L STN	49	21	23	0-1-3+, 4.9V, 90µs, 180Hz	6	C+0-1-, 2.2V, 60µs, 180Hz	7	Monopolar = 28, Advanced = 19	Same
5	79, M	ET	L VIM	58	4	26	2-3+, 3.7V, 120µs, 190Hz	9	2-1+, 3.8V, 120µs, 190Hz	7	Monopolar = 13, Advanced = 13	Clinical
13	61, M	PD	R STN	49	6	26	1-2+, 4.4mA, 90µs, 130Hz	2	3-1+, 4V, 90µs, 130Hz	0	Monopolar = 21, Advanced = 15	Automated
14	85, M	PD	L VIM	33	2	32	0-2+, 2.8V, 90µs, 130Hz	11	0-2+, 3.6V, 90µs, 130Hz	7	Monopolar = 15, Advanced = 14	Automated
15	68, M	PD	L STN	63	3	15	C+3-, 2.6V, 60µs, 150Hz	4	C+2-, 1.6V, 60µs, 150Hz	3	Monopolar = 17, Advanced = NA	Same

* Clinician administered FTM tremor scale subset including rest, postural (extended and flexed), and action arm tremor contralateral to optimized DBS lead, handwriting (if dominant hand tested), and drawings (max score 28-32).

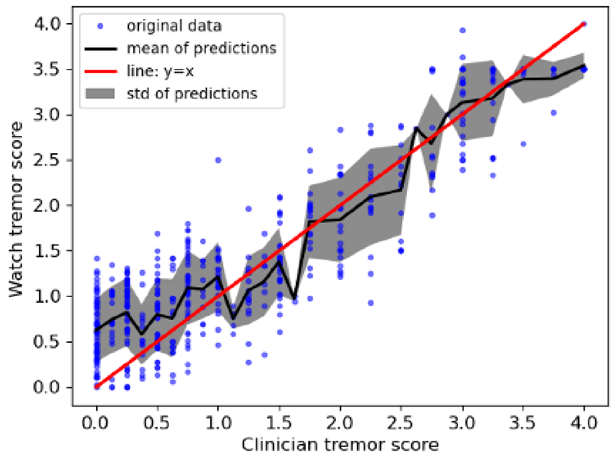


Figure 3.7: Additional validation of tremor score classifier [14]. Blue dots represent the average watch tremor scores plotted against the average clinician tremor score for selected tremor assessment tasks (rest, arms extended, arms flexed, finger-to-nose motion). Each dot represents one DBS setting that was tested during the experiments. The black solid line and the grey shaded area show the mean and standard deviation of the watch tremor scores. The red solid line is the line $y = x$ and the r-squared value of the fit to the $y = x$ line is 0.69.

We confirmed that the tremor classifier [14] performed well in this cohort of patients and its estimated tremor scores matched well with clinician scores ($r^2 = 0.69$; Figure 3.7). The clinician tremor score includes only tremor assessment tasks during the automated programming sessions and was used to further validate watch tremor classifier. The clinician administered FTM tremor scale is a more comprehensive examination consisting of a subset of FTM scale items used to evaluate tremor severity before and after automated programming session. Only the automated classifier tremor scores were used as the input into the optimization algorithm.

3.7.2 Comparison of the clinical settings and the automated settings

There was a statistically significant improvement in tremor scores from baseline (no stimulation) to the best automated setting, using both the objective watch scores and blinded clinician scores during both phases of the experiment (Figure 3.8) (the clinician tremor score is the score for selected tremor assessment tasks during the experiment where the clinician was blinded to the DBS settings). The patients also underwent a comprehensive tremor assessment exam at baseline (no stimulation), best automated setting, and their chronic clinical settings (tables 3.1 & 3.2 and Figure 3.9). We demonstrate that best automated setting and clinical setting significantly reduce the tremor to the same extent (in other words, residual tremor at automated setting was comparable to tremor at clinical setting) (Figure 3.9).

In phase I experiments with the clinician-defined safe and tolerable exploration boundaries, two patients preferred the automated setting, five had no preference, and three preferred their clinical settings. In phase II experiments with automated discovery of the safe exploration boundaries, three patients preferred the automated setting, three had no preference, and one preferred the clinical setting.

3.7.3 Speed of convergence of the automated DBS programming system

We hypothesized that our Bayesian DBS programming framework would improve sample efficiency compared to the grid search-based method in terms of the number of stimulation settings had to be tested to arrive at the optimal solution (not in terms of the required time). The grid-search approach closely resembled clinical monopolar mapping (testing all contacts and amplitudes, $0-5V$, in $1V$ increments). We could not compare the current algorithm directly against clinical monopolar mapping in terms of time since these patients had already been clinically optimized. In order to provide a fair comparison between the two approaches, we only considered the number of required samples during the monopolar programming. Grid search algorithm tested in a prior study (in a different cohort of patients) required 25.2±4.8 samples on average [14], while Bayesian automated programming in this study used 15.1±0.7 (phase I), and 17.7±4.9 samples (phase II).

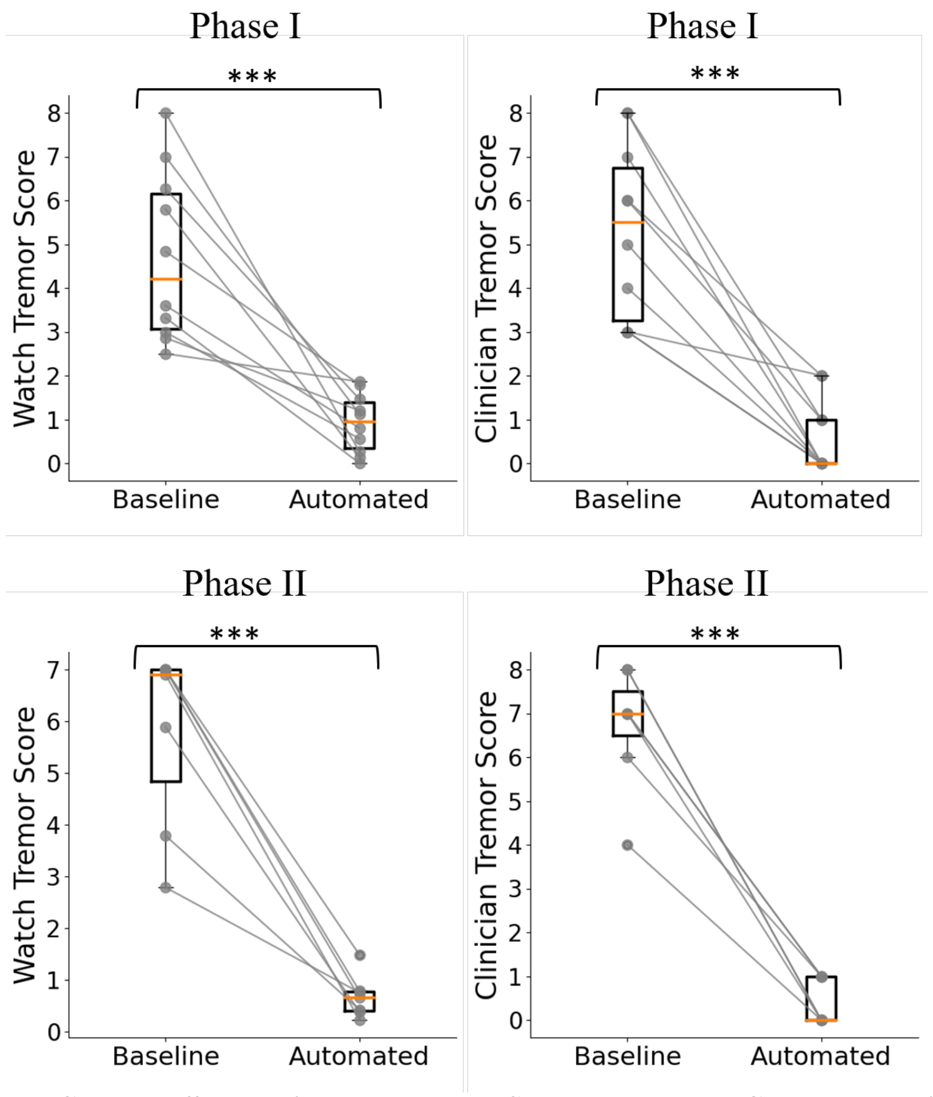


Figure 3.8: Clinical efficacy of automated DBS programming. Comparison of the patients' tremor severity scores at baseline stimulation off condition and the optimal automated setting measured by the watch (left column) and the optimal automated setting scored by a blinded clinician (right column). Top row refers to phase I (clinician-defined safe amplitudes), and bottom row to phase II (safe Bayesian optimization algorithm) experiments. The asterisk shows the conditions with statistically significant difference ($*p < 0.05, **p < 0.01, ***p < 0.001$). The tremor score is the sum of two tremor assessment tasks utilized during the automated DBS optimization session (max 8).

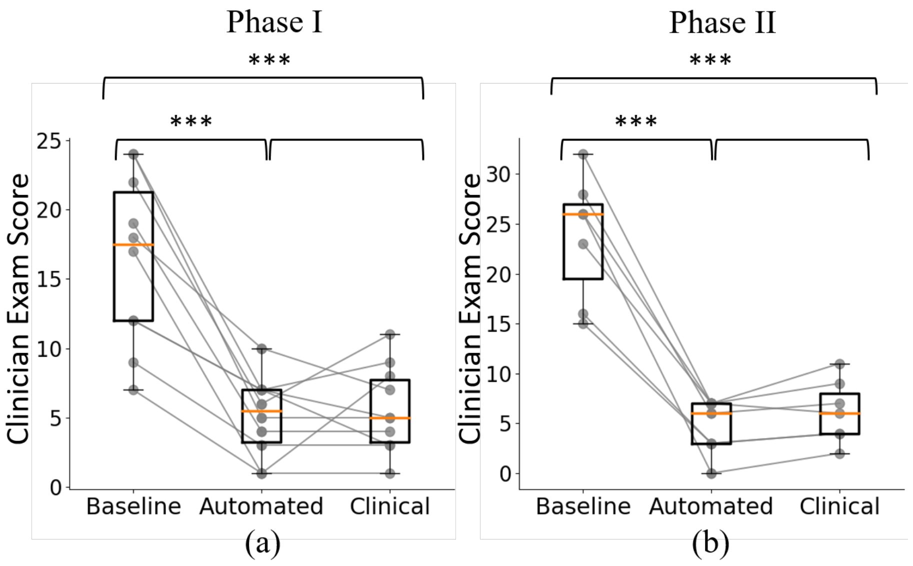


Figure 3.9: Clinical efficacy of automated DBS programming compared to clinical setting. Comparison of the patients' tremor severity scores at baseline (no stimulation), the best automated setting, and previously established best clinical setting during phase I (a) and phase II (b) based on the clinician scores during the comprehensive clinical exam. The comprehensive exam included the following items from FTM tremor scale: rest, arms extended, arms flexed, and finger-to-nose motion arm tremor contralateral to DBS lead, handwriting (if dominant hand tested), two spiral drawings, and line drawing. Both the patient and clinician were aware of the stimulation condition. The asterisk shows the conditions with statistically significant difference ($*p < 0.05, **p < 0.01, ***p < 0.001$).

3.8 Discussion

In this pilot study, we describe and evaluate an automated and patient-specific DBS programming framework for tremor treatment in 15 patients with PD or ET. A fully automated system with the Nexus-D communication bridge was developed that automatically activates the patients' IPG with the optimizer recommended DBS settings. We showed that DBS programming framework using Bayesian optimization was able to find DBS settings that were comparable in efficacy to clinical settings (previously determined by expert clinician programmers). Bayesian optimization was more efficient than previously tested grid-search method. We also describe how to use safe Bayesian optimization to automatically find safe stimulation boundaries. Finally, by incorporating the information from monopolar stimulation and clinical heuristics, we were able to add advanced DBS contact configurations (bipolar, double monopolar) that some patients require for optimal therapy into the automated DBS programming workflow and perform further optimization using four advanced DBS contact configurations. These developments may reduce the need for an expert clinician programmer to be present at the DBS programming session to perform DBS device control, symptom and side effect assessment, DBS programming decision making, and defining the safe and tolerable amplitudes for each contact configuration.

A physician can manually explore any number of settings; they are limited by the time available for a clinical visit and the patient's ability to actively participate. The purpose of the algorithm was to test only the settings most likely to yield the optimal solution. Because of the type of DBS device that the patients were implanted, the algorithm was limited to amplitude changes in 4 contact configurations during each optimization, so we tested 4 configurations during monopolar stimulation, and additional 4 during advanced stimulation if monopolar did not yield the optimal setting. Future DBS devices may provide more flexible interfaces for automated stimulation adjustments allowing wider parameter exploration.

Bayesian optimization has unique properties that make it a suitable choice to be employed at the core of an automated DBS optimization framework. Bayesian optimization is a sample-efficient and global optimization algorithm that is suitable for cases where the objective function is unknown or expensive to evaluate (the patients' response to DBS settings are unknown prior to testing and evaluating patients' responses to DBS settings is expensive from optimization standpoint as prolonged sessions are fatiguing which may compromise the accuracy with which tremor is assessed during testing). We confirmed our hypothesis that Bayesian optimization was more sample-efficient than the state-of-the-art grid-search sampling strategy introduced in [14] which closely resembled clinical monopolar mapping and directly compared the results in terms of the number of required samples to be tested to arrive at the optimal solution. Other recent studies investigated the utility of developing an objective measure for the automated selection of DBS parameters [122] and introduced a computer-guided DBS programming framework that is designed based on the clinical DBS programming strategies for the monopolar survey [123] using a grid-search approach resembling the standard clinical approaches. Their sampling strategy was a grid-search approach with $0.5V$ and 0.3V amplitude increments, respectively, leading to an even larger number of required samples to be tested compared to the grid-search approach with 1V increment introduced in [14]. Two recent clinical papers compared their proprietary algorithms with the standard of care (SoC) DBS programming in terms of the number of steps (stimulation settings) required to be tested to arrive at an optimal solution [99, 100]. The SoC was designed to be similar to the grid-search based approach. However, we could not conduct a direct and fair comparison with approaches used in [99, 100] since their parameter space was different (8 monopolar contact configurations and stimulation current (mA)). Due to very different workflows in SoC and the proposed closed-loop algorithm, the authors did not perform significance testing between the two programming modalities for time consumption.

Here, I could not compare the current algorithm directly against clinical monopolar mapping in terms of time since these patients had already been clinically optimized. Another recent work [124] used Bayesian optimization to develop a semi-automated approach for optimizing DBS parameters and provided preliminary data that shows the efficacy of Bayesian optimization in DBS programming. Here, we presented and evaluated the utility of Bayesian optimization in a fully automated DBS programming framework for tremor in a cohort of 15 PD and ET patients.

I further showed that employing safe Bayesian optimization algorithm enables unsupervised determination of safe stimulation parameters. Safe Bayesian optimization is less sample efficient in nature than the regular Bayesian optimization as it gradually expands the parameter space. To improve the sample efficiency, I used ideas from [119] to balance the tradeoff between exploring, expanding, and optimizing in addition to using a more efficient acquisition function (min-value entropy search). I showed that although safe Bayesian optimization in phase II experiments required more samples to converge than the regular Bayesian optimization in phase I, it is still more sample-efficient than the grid-search approach.

Incorporating clinical heuristic into the optimization pipeline allowed us to efficiently explore advanced contact configurations (bipolar, double monopolar). The number of possible contact configurations beyond simple monopolar is very large and I could only test a relatively small number given that patients fatigue after prolonged and repetitive testing. As a result, I used information obtained from monopolar stimulation to determine which contact configurations should be tested during advanced stimulation using clinical guidelines programmed into the programming platform rather than allowing the optimizer to make this decision. Integrating other algorithms that are more efficient for high-dimensional parameter spaces or other methods such as image-guided programming [124] to more efficiently reduce the dimensionality of the parameter space before performing the DBS optimization is a promising approach for future applications.

Our DBS programming framework relied on automated tremor detection using a wrist-worn sensor which has challenges [125], but we demonstrate that some of the issues can be overcome through GPR modeling. Tremor assessment tasks need to be synchronized with IMU recordings during the assigned tasks so patients need to be instructed to start and stop at appropriate times. Furthermore, artifact and voluntary movements unrelated to assigned tasks are often included in raw IMU data which affects the prediction of watch tremor scores, while an expert clinician programmer could detect those unrelated movements and ignore them while making judgment about the tremor severity scores. In this study, we monitored the patients and gave instructions to minimize the unrelated movements (including repeating a task if performed incorrectly), however variability in task performance (e.g. speed of movement) particularly during kinetic tremor assessment led to score predictions that were at times inconsistent with clinical scoring. Another challenge with automated tremor detection is that tremor severity can change depending on patient's internal state; for example, there can be less tremor when relaxed, and more tremor when nervous or talking, regardless of DBS settings. For example, tremor intensity during the optimization session varied significantly in patients 03 and 09 regardless of applied stimulation. A clinician can easily incorporate this information into clinical decision-making however an external sensor is agnostic to patient's internal state. To compensate for imperfections with tremor scoring, we employed the GPR model as the surrogate model of Bayesian optimization which is robust to noise of observations. Our results confirmed that the automated DBS programming method could identify effective DBS settings even in the presence of the measurement and prediction (i.e. classifier) noise. GPR model takes the uncertainty of observations into account and the model can be trained in a patient-specific manner which makes it suitable for the DBS optimization application.

The GPR modeling could be integrated in clinical decision-making process as a visualization technique that provides insight into the patient's response to DBS even without utilizing the fully automated platform. This would be particularly useful when addressing symptoms other than tremor which are even harder to quantify using sensors (e.g. bradykinesia or rigidity in PD). This visualization technique could provide insight into the spatial information (location of the active contacts) for clinicians that may not be straightforward to capture using the traditional clinical programming approaches. There have been attempts to improve visualization of DBS programming outcomes for clinicians especially with more complex segmented electrodes (e.g. [126]). We propose that GPR models could be used to not only track clinical responses but also provide suggestions for the clinicians for further DBS parameter exploration.

The success of the optimization algorithm will also depend on the choice of tremor assessment tests which are performed at each stimulation setting. In this study, we used two out of four available tremor tests (rest, postural extended, postural flexed, and kinetic) that were integrated into the automated programming software system, based on patient's clinical presentation. Although the limited set of tremor assessment tests was sufficient to evaluate DBS response in majority of patients, some patients may require other types of tests including spiral and line drawing or handwriting tests to effectively find an optimal DBS setting. For example, Patients 04 and 07 had worse tremor control on automated setting than on clinical setting, likely because they had more tremor on handwriting and spiral drawing tasks, which were not tested during automated programming session.

The automated system relied on patient reports of side effect severity as part of the combined objective measure for Bayesian optimization. Although this approach has been effective in avoiding parameters that lead to side effects in our study, this is a potential limitation since some patients find it difficult to give a score to the side effects that they experience. Developing automated side effect detection techniques could be possible for certain types of side effects (e.g. muscle contractions measured by electromyography or stimulation outside DBS target volume estimated by computational DBS activation models [124]), and could further streamline implementation of automated programming.

The objective measure defined in this study is based on aggregation of two terms including the baseline subtracted tremor score and patient-reported side effect severity score. This works since the two aggregated terms are in a comparable range. If another out of range term needs to be added to the objective measure, methods like adding a multiplier should be used to map the new term to the same range. Moreover, the work in this study can be extended to multi-objective optimization to incorporate more advanced objective measures of the clinical outcomes.

Finally, the proposed automated DBS programming framework could be beneficial for remote DBS programming for patients with limited access to the clinic. For example, a smartwatch could be mailed to the patient prior to a remote programming session or patient's own phone could provide accelerometer signal to quantify tremor. The optimization algorithm could be implemented as a standalone system providing guidance to the remote programmer, or even incorporated into the remote programming software.

3.9 Conclusion

This study developed and tested automated and patient-specific closed-loop DBS programming framework based on Bayesian optimization. This approach was more efficient than grid search method employed in clinical practice, and it yielded comparable clinical outcomes for tremor reduction as traditional clinical programming. Using such system would eliminate the need for an expert clinician programmer to be present at the DBS programming sessions. This would be particularly valuable for patients without easy access to DBS center such as those living in remote geographical locations or patients receiving care via telemedicine. Automated DBS programming methodologies will be of increasing importance as next generation DBS systems expand the number of possible parameters for delivering precise, optimized therapy to patients.

Chapter 4

Reinforcement learning for closed-loop regulation of cardiovascular system with selective vagus nerve stimulation

4.1 Introduction

Cardiovascular diseases (CVDs) pose a significant health hazard and financial strain [127]. The primary cause of death attributed to CVDs in the US is coronary heart disease, followed by stroke, high blood pressure, heart failure, diseases of the arteries, and other CVDs, as per the annual statistical report by the American Heart Association [128]. The insufficient effectiveness of existing pharmaceutical therapies in treating cardiovascular diseases has motivated research into alternative therapeutic options. Vagus nerve stimulation (VNS) has been identified as a potential treatment for various cardiac conditions, including heart failure, hypertension, atrial fibrillation, and stroke [129, 130].

VNS refers to delivering electrical stimulation to the vagus nerve through a pulse generator and is characterized by various stimulation parameters that include current amplitude (mA), pulse width (mµ), and pulse frequency (Hz). One major challenge in delivering effective VNS therapy is determining optimal VNS parameters that can produce the desired physiological response. Currently, the optimal VNS parameters are determined through an open-loop trial-and-error approach as was used in three recent clinical papers that studied the effectiveness of VNS in the treatment of heart failure [131, 132, 133]. Authors in [129, 134] provided a review of available evidence regarding the effectiveness of VNS for preventing heart failure and emphasized the lack of systematic approaches for optimizing the VNS parameters. Hence, there is a need to develop systematic approaches aimed at optimizing VNS parameters to maximize the therapeutic effects.

Closed-loop VNS strategies offer the advantage of systematically tuning the stimulation parameters. Previous studies used classical control theory approaches to investigate the utility of developing closed-loop VNS. Authors in [18] utilized a proportional-integral (PI) controller [135] for real-time regulation of instantaneous heart rate through closed-loop VNS. There have been other investigations exploring the use of proportional-integral controller designs for heart-rate regulations [19, 136, 20]. In addition, the utilization of state-space transition models was examined by [137] for the development of closed-loop VNS systems. While these classical control approaches have demonstrated their effectiveness in recent studies, they have inherent drawbacks that make them impractical for real-world physiological applications. For instance, PI controllers have limited controllability making them less effective in transient responses. Tuning complexity, high sensitivity to model parameters, and sub-optimal performance in non-linear systems are among other disadvantages of PI controllers [22, 23]. While model predictive control (MPC) has been shown to address some of the challenges of classical control algorithms, it still requires extensive tuning of parameters such as the prediction horizon and control weights to achieve optimal performance [138]. Moreover, MPC requires having access to an accurate model of the system, where inaccuracies in the model predictions can affect the controller's performance [138]. In practice, it can be challenging to obtain an accurate model of a complex system, and uncertainties can lead to sub-optimal control. There is a need to design novel closed-loop VNS techniques to address these limitations.

Developing and prototyping novel closed-loop VNS strategies requires utilizing in-silico simulation environments for evaluating the performance of these closed-loop systems before integrating in in-vivo experimental setups. Computational models of cardiac system under the effect of VNS play a crucial role in developing simulation environment to effectively design closed-loop VNS control strategies. However, the lack of the necessary variables to account for the physiological effect of VNS in most of the existing computational models makes it challenging to adopt these models for applications of closed-loop VNS. Previous research predominantly concentrated on the optimization of VNS parameters for a sole physiological signal (i.e. heart rate (HR)) and for only one VNS stimulation location [136, 20, 19]. A recent study conducted an in silico study to develop a rat cardiac model subjected to VNS in multiple VNS locations and implemented a MPC framework for regulating multiple physiological signals, i.e., HR and mean arterial pressure (MAP) [94].

While developing computational models of the cardiac system under VNS guides the design of novel closed-loop VNS strategies, the selection of correct underlying dynamics, parameter tuning, and difficulty of evaluation of such models caused by the variability of fiber recruitment in the vagus nerve makes it a challenging task. Furthermore, the computational cost of simulating full-scale in silico physiological models for real-time closed-loop control systems adds to the challenges. To address the challenges of developing a computational cardiac model, a recent study [81] employed a data-driven modeling approach using long short-term memory (LSTM) neural networks.

Authors in [81]demonstrated the utility of an LSTM model by generating synthetic data from the computational cardiac model introduced in [94] and developed an MPC controller to regulate HR and MAP. However, this approach still requires access to a substantial amount of experimental data across sufficiently covering the stimulation parameters space to accurately model the effect of VNS parameters on HR and MAP, which is not practical due to the limitations in experimental data collection.

Some data-driven optimization strategies like Bayesian optimization [83] eliminate the need of having access to underlying equations of the system to develop automated closed-loop neuromodulation systems. Authors in [84] adopted Bayesian optimization in the context of optimal experimental design with closed-loop real-time functional magnetic resonance imaging (fMRI). Bayesian optimization has been successfully utilized for seizure control [85], optimizing the DBS parameters using fMRI data [87] and for minimizing rigidity [88] for PD patients. Authors in [98] and [60] used Bayesian optimization and safe Bayesian optimization to develop an automated DBS programming framework with safety constraints for tremor suppression in PD and essential tremor patients. Bayesian adaptive dual control was introduced to reduce the beta power in a computational model of PD [90]. These data-driven optimization strategies have been successfully implemented in various applications of closed-loop neuromodulation systems. However, they share a common underlying assumption regarding the objective function's stationarity, implying that its behavior remains relatively constant within the region of interest. This assumption is essential for the acquisition function to effectively estimate regions of high uncertainty and is not applicable to dynamic systems like the cardiovascular system.

In order to overcome the aforementioned challenges while still achieving effective closed-loop VNS control, this paper presented a novel interactive AI framework using RL which provides an automated data-driven approach to design closed-loop VNS control policies with minimal assumptions and without the need for prior knowledge about the underlying physiological dynamics of the cardiovascular system. In addition, our approach enables continuous learning, where the system can learn from its experiences and continuously improve its performance which makes it suitable for developing long-term adaptive and patient-specific therapies.

By integrating the recent advancements in developing biophysics based models of the rat cardiovascular system under multi-location VNS [42], multiple simulation environments were developed with a standard application programming interface (API) to design, prototype, and evaluate the performance of the proposed data-driven closed-loop neuromodulation framework. The standard API is called Gymnasium (previously known as OpenAI Gym) [139] and is developed in Python. The original implementation of the biophysical models was in MATLAB, which proved prohibitively computationally expensive. To reduce the computational cost, a data-driven surrogate of biophysics based computational models of the rat cardiovascular system using temporal convolutional neural networks (TCN) [140] in Python. Since we are modeling time-series data, integrating TCNs are useful as they can capture temporal dependencies effectively. In addition, TCNs introduce several advantages over the canonical Recurrent Neural Networks (RNN), such as longer memory retention and ability to exploit parallelism which makes it more computationally efficient and suitable for this applications. This approach aims to reduce computational complexity and provide a unified programming environment in Python.

In this research, I tested the hypothesis that the proposed closed-loop VNS programming framework effectively learns the neuromodulation control task. Multiple simulation environments of healthy and hypertensive rat cardiovascular system in rest and exercise states were developed and designed a set point tracking task to regulate HR and MAP. Two approaches of experimental design were introduced to perform the set point tracking of HR and MAP in cardiovascular system. First, a general policy was designed to regulate the cardiovascular system (HR and MAP)

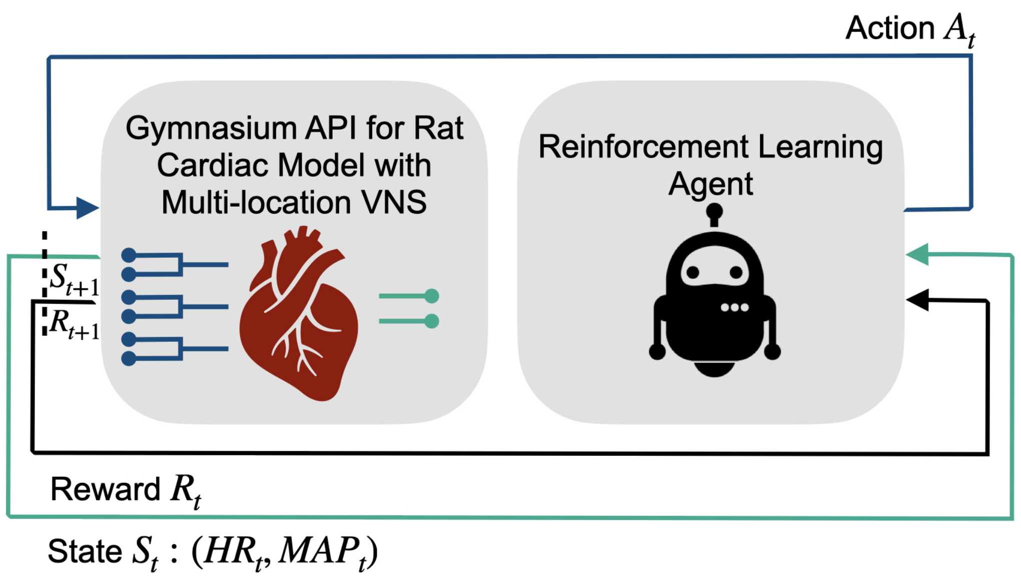

Action A_t
Gymnasium API for Rat Cardiac Model with Multi-location VNS
Reinforcement Learning Agent
S_{t+1}
R_{t+1}
Reward R_t
State S_t : (HR_t, MAP_t)

Figure 4.1: Overview of the architecture of the simulation environments for developing closed-loop VNS system demonstrating the interactions of the RL agent with the rat cardiac model using the standard API. The left block represents the reduced-order surrogates of the physiological cardiac models wrapped with the standard Gymnasium API, where the inputs of the model (color-coded as dark blue) are stimulation frequency and stimulation amplitude across three different locations at time t (A_t). The outputs of the model (color-coded as green) are the response of HR and MAP to the VNS parameters. The model estimates the response of the cardiac system (HR_{t+1}, MAP_{t+1}) to the action A_t taken at time step t given the current state of the system (HR_t, MAP_t). The right block represents the reinforcement learning agent, which takes action A_t according to its policy at time step t, and observes the next state S_{t+1}, and Reward R_{t+1}.

using deep RL algorithms, i.e., soft actor-critic (SAC) [82] and proximal policy optimization (PPO) [141]. Additionally, since sample efficiency is critical in the design of closed-loop neuromodulation systems, I designed a sample-efficient adaptive policy using a model-based RL algorithm known as probabilistic inference for learning and control (PILCO) [142], which represents a sample-efficient approach to policy search. Furthermore, I examined the adaptability of the proposed frameworks to variations in both the target set-point and the underlying dynamics of the environment. Finally, transfer learning [143] was employed to improve sample efficiency for deep RL algorithms.

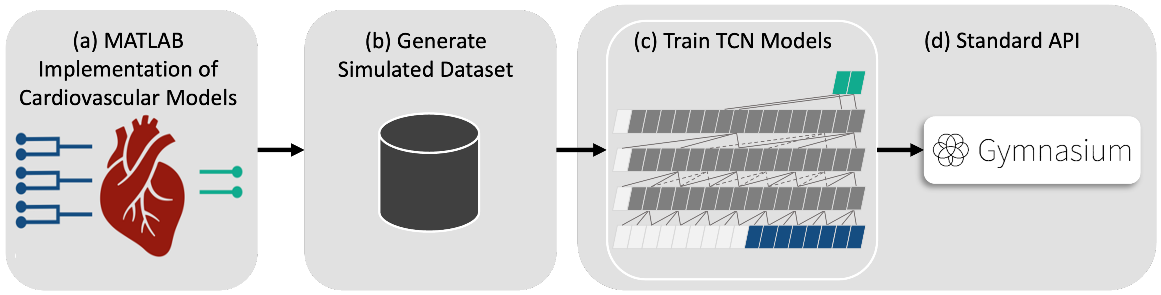

(a) MATLAB Implementation of Cardiovascular Models
(b) Generate Simulated Dataset
(c) Train TCN Models
(d) Standard API
Gymnasium

Figure 4.2: The pipline used for developing the simulation environments using the Gymnasium API to test and prototype RL algorithms for regulating the cardiovascular system. (a) Used the physiological models of rat cardiac system under multi- location VNS implemented in MATLAB, (b) generated a simulated data set of the response of the cardiac system by varying randomly selecetd VNS parameters, (c) trained the reduced-order TCN model to model the response of HR and MAP to VNS parameters, and (d) used the Gymnasium standard API wrapper over the trained TCN models for easier compatibility with RL algorithms.

4.2 Simulation environments

Multiple simulation environments were developed for evaluating the performance of RL algorithms in developing closed-loop VNS systems. The high-level overview of the proposed closed-loop VNS system is depicted in Figure 4.1. The pipeline of developing the simulation environments with the standard Gymnasium API is presented in Figure 4.2. Detailed description of each component is provided in the following subsections.

4.2.1 Standard API for rat cardiac model

The simulation environments of the rat cardiac model under multi-location VNS were developed using a standard API called Gymnasium for testing and prototyping RL algorithm for regulating the cardiovascular system. Gymnasium (previously known as OpenAI Gym [139]) is a standard environment for developing and testing learning agents, especially reinforcement learning agents. Gymnasium API is adopted for implementing the in-silico rat cardiac model to provide a standard interface for the users and provide the flexibility of designing their own control task with their learning agents of choice (Figure 4.2d). The details of the in-silico physiological rat cardiac model are described in section 4.2.2.

4.2.2 In-silico rat cardiac model

In this study, a previously published physiological model of the integrated cardiovascular system and baroreflex regulation under multi-location vagal nerve stimulation was integrated [82]. The model was composed of three different parts including the cardiovascular system, the baroreflex system, and the VNS device. The cardiovascular model uses a lumped-parameter approach to predict the blood circulation in five elastic chambers representing the left heart, the arteries and veins in the upper and lower body. The right heart and the pulmonary circulation are modelled by capacitance, which is added to the venous capacitance in the upper body. The baroreflex system functions to regulate the arterial pressure through the baroreceptor, afferent pathway, efferent pathway and the effectors in cardiovascular system. Each of the compartments in the baroreflex system was modelled using a firing-rate- based approach. The VNS device model predicts the response of firing rate of different fibers to VNS parameters [144]. Three fiber types are engaged during VNS, representing the baroreceptive fibers, the vagal fibers, and the sympathetic fibers. Each type of fibers distributes nonhomogeneously in different locations. Activation of each fibre type in each stimulation location due to stimulation amplitude is modelled by an activation function, while the change of fibre activities due to stimulation frequency is represented by a conduction map. The overall physiological model of the rat cardiac system models the effect of VNS parameters (stimulation amplitude and frequency) in three different locations (leading to six total VNS parameters) on two physiological variables (HR (BPM) and MAP (mmHg)). The short-term effect of VNS parameters on the output of HR and MAP was calculated through the interactions between the cardiac systems and the neural regulation system. More details on the physiological model is provided in [82].

A modification of the described physiological model is used in this study to simulate the physiological model in four different conditions adding the effect of hypertension and physical exercise (healthy and hypertensive rat cardiac models in rest and exercise states) by changing the related internal states and parameters [42]. I refer to the four models as healthy cardiac environment (HC Env), healthy cardiac environment with the effect of exercise (HCE Env), hypertension cardiac environment (HTC Env), and hypertension cardiac environment with the effect of exercise (HTCE Env). Hypertension is related to increased arterial stiffness, vascular remodelling, increased sympathetic activities and decreased vagal activities. An offset in sympathetic and vagal activity coupled with modifications on the gain of each effector are used to represent the hypertensive condition. An acute exercise triggers multiple physiological responses, including redistribution of blood flow and modification of sympathetic and vagal activities by central command. An additional offset caused by exercise on sympathetic and vagal pathway, as well as the separation of the peripheral resistance into resting muscle resistance and active muscle resistance are used to represent the exercise condition. The new steady state during exercise consists of a higher arterial pressure, heart rate, stoke volume and cardiac output compared with rest state in both healthy and hypertensive condition.

4.2.3 Reduced order model of the physiological rat cardiac model using temporal convolutional neural networks

The physiological model of the rat cardiac system described in the previous section originally was implemented in MATLAB using the dde23 solver [145] which is computationally expensive. In addition, since the standard API for RL algorithms (Gymnasium) as well as most of the common RL algorithm libraries are developed in Python, data-driven reduced-order models of the physiological models were developed using temporal convolutional neural networks (TCN)[140] to not only reduce the computational complexity, but also to provide a unified programming environment in Python.

The four physiological models were used to generate a dataset by varying the VNS parameters (stimulation amplitude and frequency) across the three VNS locations (leading to a six-dimensional parameter space) and measuring the effect of VNS parameters on HR and MAP (Figure 4.2.a, 4.2.b). The range of VNS parameters in generating the synthetic data was $0 - 50Hz$ for stimulation frequency and $0 - 1mA$ for stimulation amplitude. The response of HR and MAP to randomly selected VNS stimulation parameters was collected in 2000 runs. Each run consisted of 30 cardiac cycles to generate the dataset. The data was divided into training and test sets with an $80\% - 20\%$ split. The training set was then used to train a reduced-order model using TCN (Figure 4.2.c). A TCN model with an input layer of width 8 and output layer of width 2, and three hidden layers of width 16 with the dilation factors of 1, 2, 4 was used. The standard API of Gymnasium (the maintained fork of OpenAI's Gym library) was used to communicate between RL algorithms and the environments. The pipeline of developing the simulation prototyping of RL agents for regulating the cardiovascular function is depicted in Figure 4.2.

4.3 Experimental design

4.3.1 Regulating cardiovascular system using RL through designing a set point tracking Task

Designing neuromodulation control systems depend on multiple factors including the computational and sample efficiency within the constraints of the problem, having access to related data sets or the equations of the underlying dynamics of the system, etc. Here, the underlying dynamics of the environment are assumed to be unknown and I hypothesized that utilizing RL eliminates this requirement while effectively learning to perform a set point tracking task for regulating HR and MAP values.

The subsequent subsections outline the simulation design employed to optimize the VNS parameters to regulate HR and MAP values during a set point tracking task. Specifically, two experimental design approaches were considered to elaborate the advantages and disadvantages of each method and provide guidance on future experimental design selection. Both experimental design approaches were used to perform the same set-point tracking task, where the set-point was a two-dimensional vector of desired HR and MAP values, and the agents were trained to apply proper stimulation parameters according to their policies with the goal of reaching the desired set points. The RL algorithm's reward function was defined to perform the set point tracking task as described in section 4.4.4.

4.3.2 Designing a general policy using deep RL algorithms

The first experimental design approach was to design a general policy to perform the set-point tracking task, where the agent learned the general policy during the training mode and the trained policy was used in the inference mode to perform the set-point tracking task. The overview of designing a general policy is depicted in Figure 4.3. The term general is used to denote that the trained policy was designed to perform the set-point tracking task for all of the potential set-points within the possible range of HR and MAP values (Table 4.1) rather than learning to follow a single set-point at a time. Here, PPO and SAC algorithms as described in sections 4.4.1 and 4.4.2 were used to train a general policy. In both algorithms, the policy architecture was a fully connected feed-forward neural network also known as a multi-layer perceptron (MLP) [146]. The input of the policy network is extended to account for learning multiple set points at the same time (general policy) by feeding the desired set points HR_{target} and MAP_{target} as additional inputs to the policy network besides the current state of the environments HR_t and MAP_t as depicted in Figure 4.3. During training, a randomly selected set point was assigned within the possible range of HR and MAP values (Table 1) for each of the four models in each episode with an episode length

Table 4.1: Table 1. Sampling range of HR and MAP values ([minimum, maximum]) for the four cardiac environments.

	HC Env	HCE Env	HTC Env	HTCE Env
HR (BPM)	[234.4, 414.7]	[309.6, 578.6]	[245.8, 428.9]	[290.36, 578.1]
MAP (mmHg)	[71.9, 173.6]	[117.4, 158.1]	[86.6, 194.8]	[117.3, 178.3]

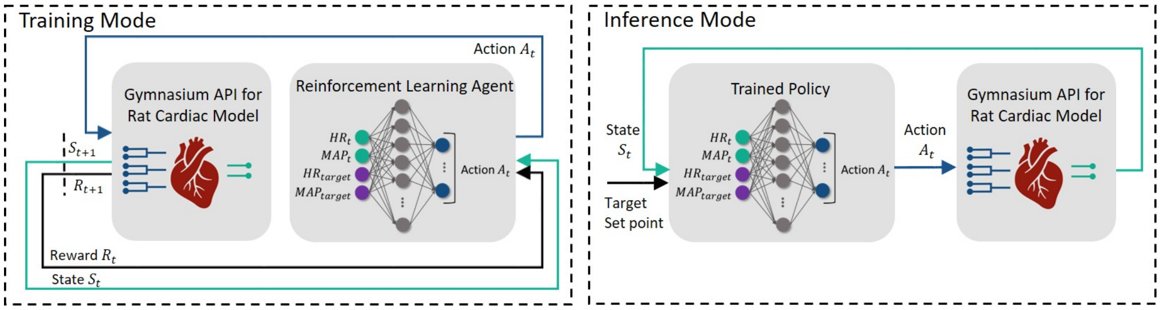


Figure 4.3: The overview of designing a general policy. The left panel represent the structure of the simulation environment with the standard Gymnasium API during the training mode. The right panel depicts the simulation environment in the inference mode. The policy network of the RL agents was designed as a simple MLP model, where HR_t and MAP_t are the current states of the environment. The input of the policy network was extended by adding $HR_{target}$and MAP_{target} (target set-points) to design a general policy. The environment is the reduced-order surrogate of the physiological cardiac models wrapped with the standard Gymnasium API, where the input of the model (color-coded as dark blue) are stimulation frequency and stimulation amplitude across three different locations at time $t(A_t)$. The output of the model (color-coded as green) are the response of HR and MAP to the VNS parameters.

of 500. After training the RL agents, the policy network was used in the inference mode to perform the control task (right panel in Figure 4.3).

4.3.3 Designing an adaptive policy using PILCO

The second experimental approach aimed to dynamically learn an adaptive policy through interactive engagement with the environment (see Figure 4.4). In this approach, PILCO was utilized, as described in section 4.4.3, to train the adaptive policy on-the-fly. PILCO operates by executing actions based on its policy for N iterations, gathering state transitions and reward values from the environment, augmenting its dataset, updating the underlying Gaussian Process (GP) [147] model of the state transition, and adjusting its policy parameters using the augmented data and repeats the same process. The key distinction in designing an adaptive policy, as opposed to

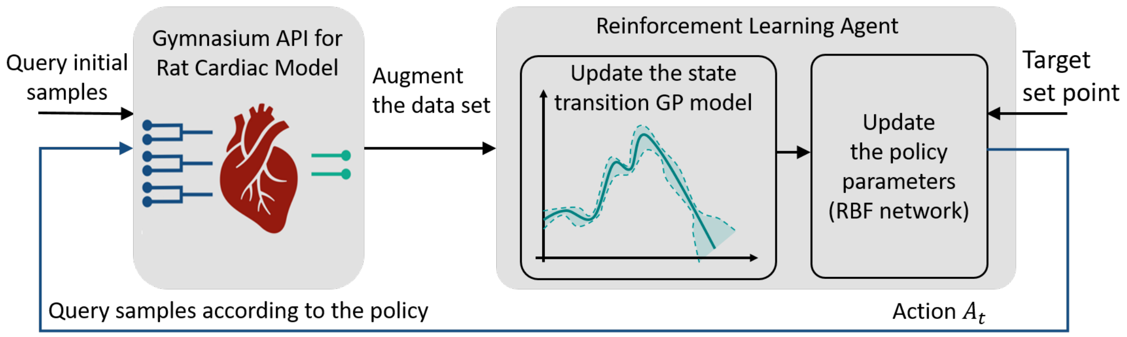

Query initial samples
Gymnasium API for Rat Cardiac Model
Augment the data set
Reinforcement Learning Agent
Update the state transition GP model
Update the policy parameters (RBF network)
Target set point
Query samples according to the policy
Action A_t

Figure 4.4: The workflow of adaptive policy using PILCO, illustrating the iterative process where actions are executed according to the recent policy (or randomly selected from the parameter space for the initial query) for N iterations. PILCO collects state transitions and reward values from the environment in response to the actions, augments its dataset, updates the Gaussian Process (GP) model of the state transition, and adjusts the policy parameters based on the augmented data. This process is repeated to improve the adaptive policy over time. The environment is the reduced-order surrogate of the physiological cardiac models wrapped with the standard Gymnasium API, where the input of the model (color-coded as dark blue) are stimulation frequency and stimulation amplitude across three different locations at time $t(A_t)$. The output of the model (color-coded as green) are the response of HR and MAP to the VNS parameters.

a general policy, lies in its ability to learn and track a specific setpoint during interactions with the environment. In contrast, a general policy in the inference mode can be used without further training to track multiple setpoints.

4.4 Reinforcement learning agents

A standard RL task can be formulated as a Markov Decision Process (MDP) defined by a tuple (S, A, R, T, P), where S and A are state and action spaces, R is a reward function ($R_t = f(s_t, a_t, s_{t+1})$), T is the set of terminal conditions, and P is the state transition probability. The goal of reinforcement learning is to find the optimal policy π^* by maximizing the cumulative reward, typically with the discount factor γ. The overview of the standard RL framework is depicted in Figure 4.1. The closed-loop flow of the interactions for the environment with the RL agents is as follows. At each time step, the agent interacts with the environment to learn the policy purely from interactions and without requiring prior knowledge about the underlying dynamics of the environment. The agent observes the current state at time t and then takes an action with respect to the policy. Next, the environment returns the next state and reward at time step $t + 1$ (S_{t+1}, R_{t+1}). This information is used to improve the policy. In this study, two deep RL algorithms (PPO and SAC) were designed to train a general policy and PILCO was employed to train an adaptive policy. The details of these algorithms are provided in the following sections.

4.4.1 Proximal policy optimization algorithm

Policy gradient (PG) algorithms are a type of RL algorithms that rely upon optimizing parametrized policies with respect to the expected long-term return using gradient descent. Unlike vanilla PG [148] that keep new and old policies close in the parameter space, trust region policy optimization (TRPO) [149] algorithm updates policies by taking the largest step possible to enhance the performance while satisfying a constraint expressed in terms of KL-Divergence on how close the new and old policies are allowed to be. Proximal policy optimization (PPO) combines the advantages of vanilla PG and TRPO to ensure stability and scalability by employing a surrogate objective function to update the policy parameters.

In this study, PPO-Clip was used, a variant of PPO that utilizes specialized clipping in the objective function to prevent significant deviations of the new policy from the old policy. As a result, PPO offers a simpler implementation, while empirically performs at least as well as TRPO. PPO is applied in an actor-critic framework. The actor maps the state to action and the critic gives an expectation of the agent's reward with its corresponding state. The policy is updated via a stochastic gradient ascent optimizer to ensure the exploration while the agent will gradually tend to exploit what it has learned over the course of training. A MLP model with input layer of width 4 $(HR_t, MAP_t, HR_{target}, MAP_{target})$, one hidden layer of width 64, and output layer of width $6(A_t)$ was used to represent the policy network. Stable Baselines library [150] for implementing PPO algorithm was used in this study.

4.4.2 Soft actor-critic algorithm

SAC [151, 143] is a RL algorithm widely employed in continuous action spaces for various control tasks. Soft actor-critic (SAC) is from the family of off-policy RL algorithms that optimizes a stochastic policy. As the name suggests, SAC is also an actor-critic algorithm. A central feature of SAC is entropy regularization to encourage effective exploration during learning. The policy is trained to maximize a trade-off between expected return and entropy, a measure of randomness in the policy, which has a close connection to the exploration-exploitation trade-off. Increasing entropy results in more exploration, which can accelerate learning. It can also prevent the policy from prematurely converging to a bad local optimum, resulting in stable training. Moreover, SAC leverages neural networks to represent both the policy and the value functions, enabling it to handle high-dimensional observation spaces effectively. A MLP model with input layer of width 4 $(HR_t, MAP_t, HR_{target}, MAP_{target})$, one hidden layer of width 64, and output layer of width $6(A_t)$ was used to represent the policy network. The same architecture of the policy network for both PPO and SAC algorithms was used. The Stable Baselines library [150] was integrated for implementing SAC algorithm.

4.4.3 Probabilistic inference for learning and control

PILCO is a model-based data-efficient approach to policy search [145], which offers improved sample-efficiency compared to model-free RL algorithms. Model-based RL algorithms rely on accurate models of the underlying dynamics of the system, which can result in reduced performance in the presence of model bias. Model bias is particularly an issue in cases where there is limited prior knowledge available. PILCO mitigates the need for prior access to the underlying dynamics of the environment by learning the model from observed data. Furthermore, PILCO utilizes GP, a non-parametric probabilistic model [147], to effectively address the issue of model bias by accounting for model uncertainty. The main advantage of PILCO is that it remarkably improves the sample efficiency in continuous state-action spaces which sets the pathway for the integration of PILCO in the implementation and deployment of closed-loop VNS systems in clinical settings and experimental setups.

Consider the following dynamics system

$$x_t = f(x_{t-1}, u_{t-1}), \tag{4.1}$$

where f is the unknown state transition function with continuous state, x, and action, u domains. The goal of PILCO is to find a deterministic policy that maximizes the expected return or minimizes the expected cost, $c(x_t)$ of following the policy π for T time steps as in:

$$J_\pi(\theta) = \Sigma_{t=0}^{T} E_{x_t}[c(x_t)], x_0 \sim N(\mu_0, \Sigma_0). \tag{4.2}$$

PILCO assumes that π is a function parametrized by Θ and that the cost function $c(x)$ encodes some information about a target state x_{target}. I used the squared exponential cost function as in equation 4.4. The GP model uncertainty is used for planning and policy evaluation steps. PILCO evaluates the policy using the deterministic approximate inference method which enables policy improvement through analytic policy gradients. Analytic policy gradient is more efficient than estimating policy gradients with sampling and enables using standard non-convex optimization methods like LBFGS to find the optimal policy parameters. Here, the learned state-feedback controller is the nonlinear radial basis function network as follows:

$$\pi(x, \theta) = \Sigma_{i=1}^{n} \omega_i \phi_i(x), \phi_i(x) = \exp(-\frac{1}{2} - \mu_i)^T \Lambda^{-1}(x - \mu_i)), \tag{4.3}$$

where the parameters of the RBF network controller were optimized using LBFGS optimization. The implementation in this GitHub repository was integrated for implementing PILCO.

4.4.4 Reward Function

The exponential reward function was used in training all RL algorithms, where x_t was the two-dimensional state of the environment at time $t(HR_t, MAP_t)$ and x_{target} was the two-dimensional target set-point of HR and MAP values $(HR_{target}, MAP_{target})$.

$$c(x_t) = 1 - exp(\frac{||x_t - x_{target}||^2}{\sigma_c^2}) \tag{4.4}$$

4.5 Results

4.5.1 Performance of TCN model

The four different rat cardiac models (i.e. the healthy and hypertensive models in rest and exercise states) were used to generate synthetic data for training a computationally more efficient TCN model as a surrogate of the mechanistic model implemented in MATLAB. I evaluated the performance of the TCN models in predicting HR and MAP values as a function of stimulation parameters in three different locations. The normalized mean squared error (NMSE) of the predictions from the TCN models was calculated and reported in Table 4.2. In addition, the computational efficiencies (how many times the predictions from the TCN model in the inference mode were faster than the same predictions from the biophysics model implemented in MATLAB) of the four reduced order TCN models are reported in Table 4.2. Additionally, a sample comparison of the HR and MAP values using the MATLAB implementation versus the predictions from the reduced order TCN model is depicted in Figure 4.5.

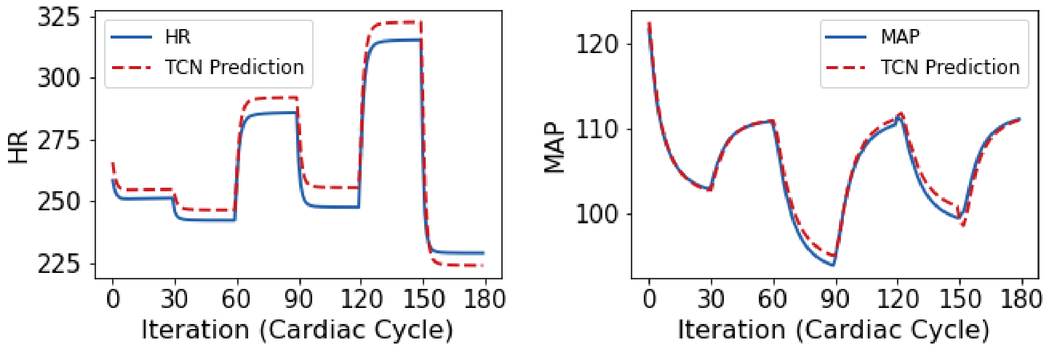


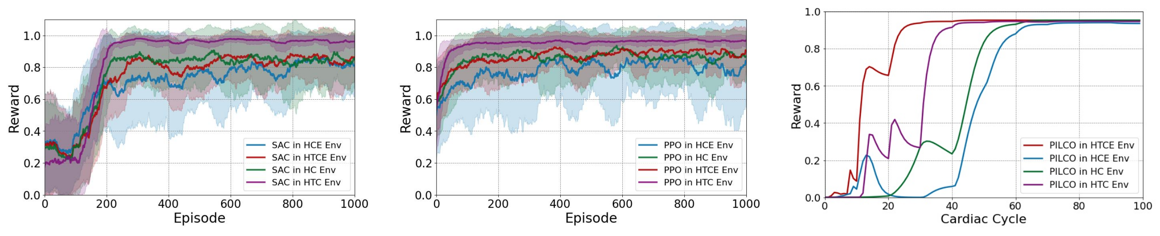
Figure 4.5: Comparison of the HR and MAP values predicted from the original biophysical model implement in MATLAB versus the predictions from the reduced-order TCN model. The blue solid lines represent the HR and MAP values generated from the HC biophysics model implemented in MATLAB. The red dashed line represents the corresponding valued generated with reduced-order TCN model.

Table 4.2: Performance of TCN models; normalized mean squared error (NMSE) of TCN models, and their computational efficiency compared to the MATLAB implementations.

	HC Env	HCE Env	HTC Env	HTCE Env
NMSE (on test set)	0.001142	0.001686	0.002336	0.002029
Computational efficiency (x times faster than Matlab)	11.65	8.77	10.90	8.74

4.5.2 Training performance of RL agents

Multiple experiments were conducted to evaluate and compare the performance of RL algorithms described in section 4.4 in performing the set-point tracking task for regulating HR and MAP values (Figure 4.6). The normalized reward values of model-free deep RL algorithms (PPO and SAC) were demonstrated in Figure 4.6a, 4.6b. The total reward values per episode were normalized by the episode length of 500 and passed through an moving average function with window length of 50 to provide a better representation of the agents' performances over time in four different environments (HC, HCE, HTC, and HTCE). As demonstrated in Figure 4.6, PPO converges faster compared to SAC. The reward values of PILCO during the set-point tracking task for 100 iterations were depicted in Figure 4.6c for the four rat cardiac models. As shown in Figure 4.6, the convergence speed of PILCO is much faster than Deep RL algorithms making it more suitable for experiments with a limited budget in terms of the number of interactions with the nervous system of interest.



(a) SAC (b) PPO (c) PILCO

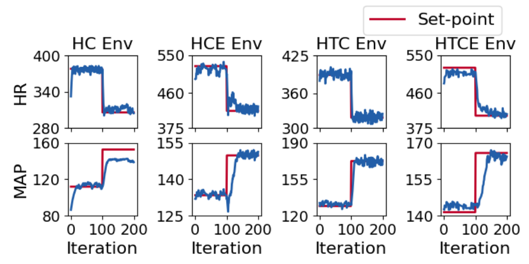
Figure 4.6: Reward values of RL agents during the set-point tracking task in four cardiac environments; (a) Normalized training reward values per episode for SAC during the training mode, (b) Normalized training reward values per episode for PPO during the training mode, (c) Reward values of PILCO during the experiment. The normalized reward for deep RL algorithms (a, b) represent the mean ± standard deviation of the reward calculated through a moving average with window length of 50 to provide a better representation of the agents' performances over time.

4.5.3 Performance of Deep RL agents in set-point tracking task in four cardiac models

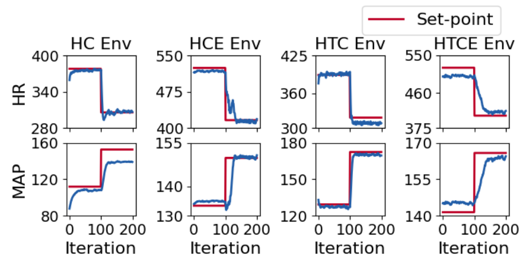
The performance of Deep RL algorithms in inference mode for set-point tracking task across four cardiac environment using PPO and SAC algorithms is depicted in Figures 4.7a and Figure 4.7b, respectively. Their associated actions (stimulation parameters) taken during the inference mode were demonstrated in Figure 4.8a, 4.8b. The set-points in Figure 4.7 are set to be 40% and 80% of the maximum range of HR and MAP for each of the four environments. As shown in the figures, the trained general policy in PPO and SAC algorithms learned to track the target set points.

4.5.4 Performance of PILCO in set-point tracking task in four cardiac models

The performance of set-point tracking for the PILCO algorithm is depicted in Figure 4.9a.The associated actions (i.e. stimulation parameters) taken across the three VNS

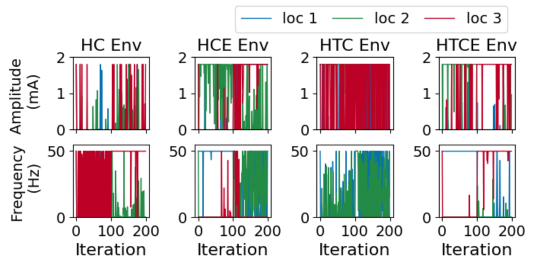


(a) PPO

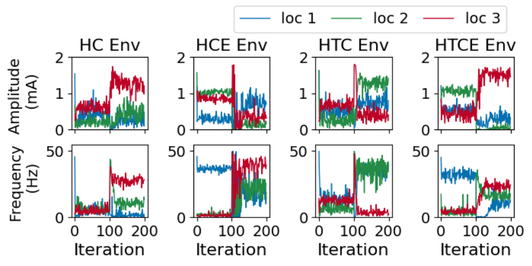


(b) SAC

Figure 4.7: The performance of Deep RL algorithms in inference mode for set-point tracking task across four cardiac environments using (a) PPO and (b) SAC algorithms for 200 iterations. The red solid lines represent the desired set points and the blue lines represent the states of the four cardiac models (HR and MAP). The target set points were changed after 100 iterations, where iterations are equal to the cardiac cycle.



(a) PPO



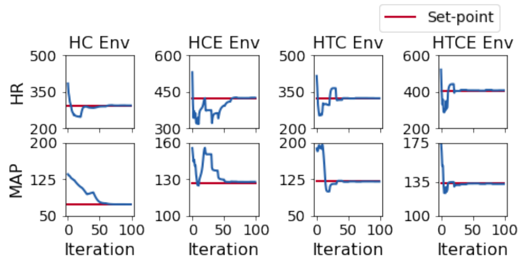
(b) SAC

Figure 4.8: The stimulation parameters used during the inference mode for the set-point tracking task across the four cardiac environments (a) using PPO and (b) SAC algorithms for 200 iterations. The stimulation parameters were amplitude and frequency across three VNS locations. The target set points were changed after 100 iterations, where iterations are equal to the cardiac cycle.

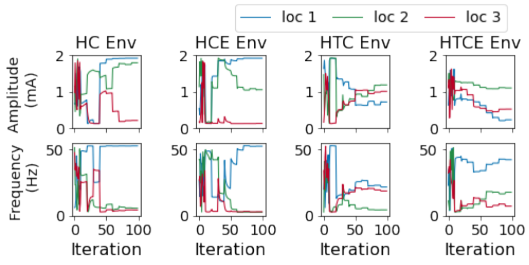
stimulation locations are demonstrated in Figure 4.9b. As shown in these figures, PILCO started by taking N=10 random samples and gradually learned a GP model of the underlying dynamics as well as a RBF network policy to track the target set-point. As shown in Figure 4.6c and Figure 4.9a PILCO learns to track the target set point in less than 100 iterations.

4.5.5 Adaptability of PILCO to variations in target set point

Unlike model-free deep RL algorithms (i.e. PPO and SAC) that were designed to train a general policy that has the ability to track a wide range of set points, PILCO is designed to track a single set point at a time. We designed an experiment to change



(a)



(b)

Figure 4.9: The performance of PILCO in set-point tracking task across four cardiac environments using PILCO (a) and its corresponding stimulation parameters (amplitude and frequency) across three stimulation locations (b). In the left figure (a), the red lines represent the desired set points and the blue lines represent the states of the four cardiac models (HR and MAP).

the target set-point after the first 100 iterations to test the performance of PILCO in learning a new randomly selected set-point in the HC environment. The reward values for PILCO in adapting to a new target set point for the HC environment was provided in Figure 4.10a. As shown in Figure 4.10a-c, PILCO learned to track the new set point in around 40 iterations.

4.5.6 Adaptability of PILCO to variations in the underlying dynamics of the environment

Another experiment was designed to validate the ability of PILCO in performing the set-point tracking task when the underlying dynamics of the system changes over time. After 100 iterations, we changed the environment from HC to HTC environment. The reward values for PILCO in adapting to a target set point in a new cardiac environment (HTC) after 100 iterations was provided in Figure 4.10d. As shown in Figure 4.10d-e, PILCO learned to track the new set point in the new environment in around 20 iterations.

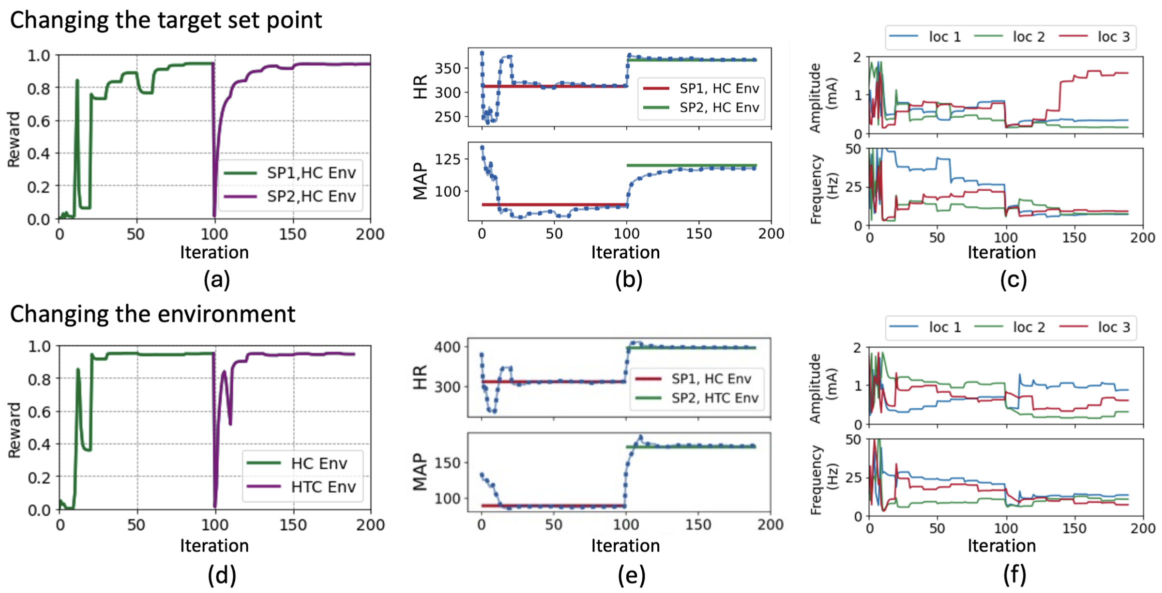
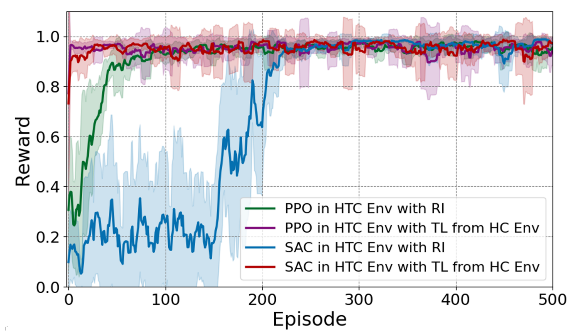
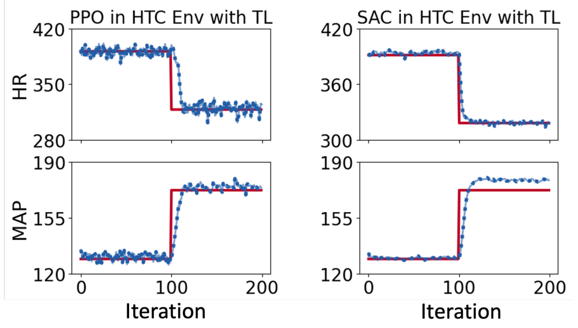


Figure 4.10: Adaptability of PILCO during the set point tracking task to variations in the target set point (a-c) and to variations in the underlying dynamics of the environment (d-f). (a, d) Reward value; (b,e) the state trajectory, and (c,f) stimulation parameters for 200 iterations, where the changes where applied after 100 iterations.

4.5.7 Adaptability of deep RL agents to variations in the underlying dynamics of the environment using transfer learning

An experiment was conducted to assess if PPO and SAC can adapt to changes in the underlying dynamics of the environment. To achieve this, transfer learning (TL) was employed, and the pre-trained policy in the HC model was used as the initial policy instead of starting with a random policy. TL was adopted to fine-tune the model to perform the set-point tracking task in the HTC environment. As depicted in Figure 4.11a, TL considerably improves the speed of convergence for both RL agents (PPO and SAC) and quickly adapts to the new dynamics of the environment. SAC and PPO converge in less than 10 episodes with TL as opposed to more than 200 and 100 episodes without TL, respectively. The result of performing the set-point tracking task with the fine-tuned policy using TL from the HC environment to HTC environment is depicted in Figure 4.11b.





(a) (b)

Figure 4.11: Adaptability of PPO and SAC algorithms to the variations in the underlying dynamics of the environment using transfer learning; (a) comparison of the reward values of PPO and SAC with random initialization (RI) and with transfer learning (TL), (b) performance of PPO and SAC in set point tracking task with the trained policy using TL approach.

4.6 Discussion

In this study, I described and evaluated an interactive AI framework using RL for automated data-driven design of closed-loop VNS control systems. I implemented this framework to regulate HR and MAP in computational models of rat cardiovascular system under multi-locations VNS. I provided multiple simulation environments using biophysics-based computational models of rat cardiovascular system in four different conditions including healthy and hypertensive rat in rest and exercise states (HC, HTC, HCE, and HTCE models). The simulation environments were built through the standard Gymnasium API (previously known as OpenAI gym) to facilitate testing and prototyping of the interactive RL-based closed-loop neuromodulation systems. Furthermore, the utility of the framework in designing adaptive closed-loop neuromodulation systems through a set-point tracking task was demonstrated. In addition, two experimental design approaches (i.e., general policy and adaptive policy) feasible for the integration of RL algorithms were introduced which could be utilized based on the limitations and requirements of the application of interest. I compared the performance of the framework using multiple model-free and model-based RL algorithms in terms of sample efficiency and quality of performing the set-point tracking task as well as their ability to adapt to the new target set pints and to the variations in the underlying dynamics of the environment.

Multiple simulation environments were provided for testing and prototyping RL agents and designed a set-point tracking task to modulate the desired HR and MAP values. TCN was used to build a reduced order model of the original MATLAB implementation in Python with the standard OpenAI Gym environment and our results confirmed the improved computational time (Table 4.2) while providing a coherent programming environment in Python.

A control policy is at the core of an automated closed-loop neuromodulation system which automatically adjusts the stimulation parameters to achieve the goals of a desired neuromodulation task. Current approaches to adjusting VNS parameters are based on open-loop trial-and-error method and a systematic approach of tuning VNS parameters is needed [98, 99, 84]. Recent studies investigated the utility of MPC in regulating HR and MAP values through VNS[81, 83], however, they required having access to an accurate model of the underlying dynamics which is not practical for many applications. Our results support the hypothesis that our interactive AI framework can generate effective VNS control policies in a data-driven fashion with minimal assumptions and without the requirement of having access to the exact underlying dynamics of the nervous system. Our framework enables continuous learning from the experience which makes it suitable for developing long-term patient-specific therapies. Scalability to continuous state and action spaces, actively exploring to improve the performance and the ability to learn in real-time from data are among the other advantages of the RL-based control strategies.

The automated and adaptive VNS programming framework was evaluated using multiple model-free and model-based RL algorithms. The comparison of PPO, SAC, and PILCO algorithms provides insights into the performance of different classes of RL approaches and guides the algorithm selection for the design of closed-loop VNS neuromodulation system. Model-free deep RL algorithms have shown to be less sample efficient than PILCO at the expense of learning a general policy (Figure 4.6). PPO, known for its stability and sample efficiency, demonstrated promising results in maintaining control over the set-point tracking task across different cardiac environments and over the range of potential target set points. SAC, with its emphasis on exploring the action space, exhibited competitive performance with a smoother behaviour in the action space (Figure 4.8). On the other hand, PILCO, an uncertainty-driven and model-based algorithm, showed robustness in handling the inherent variability in the neuromodulation system (underlying dynamics of the environment or the target set point) while requiring a considerably smaller number of samples to learn the set point tracking task.

In addition, two experimental design approaches were considered. First, I integrated Deep RL algorithms to train a general policy capable of tracking a range of set points in the inference mode. Additionally, PILCO was used to train an adaptive policy on the fly, which is capable of tracking one predefined set of set points while being able to adapt to the variations in the target set point over time. These findings highlight the trade-offs between sample efficiency, generalizability, and adaptability in the context of closed-loop VNS neuromodulation, offering researchers and practitioners valuable insights for selecting the most suitable algorithm for their specific application requirements. Further research could explore hybrid approaches or algorithm modifications to enhance the performance of these algorithms in closed-loop neuromodulation systems.

While PILCO demonstrated promising performance in quickly learning the set point tracking task as well as in adaptability to the target set point and the variations in the underlying dynamics of the environment, one of the limitations of PILCO is the use of GP approach and its limited generalizability to higher dimensions. While GP modelling and PILCO have proven to be very sample-efficient effective in lowdimensional control problems, their performance tends to degrade as the dimensionality increases. Therefore, future directions should explore alternative approaches that can enhance sample efficiency and improve generalizability to higher dimensions.

While Deep RL algorithms are less sample efficient than PILCO, their ability to learn a general policy is valuable in the context of developing a generalized therapy with the capability to adapt to the unique needs of individual patients. Therefore, it is crucial to come up with potential approaches to improve their sample efficiency which facilitates their integration in clinical and experimental setups. To address this, I integrated TL as a potential approach to improve sample efficiency of deep RL algorithms. By incorporating TL, we can leverage pre-existing knowledge from related tasks or domains to initialize and guide the learning process of the deep RL algorithms. In this study, I trained a general policy using the healthy rat cardiac model in rest state (HC Env) and incorporated this prior knowledge by using TL to improve sample efficiency for the hypertensive cardiac model in rest state (HTC Env). Our results demonstrated a considerable improvement in sample efficiency of Deep RL algorithms using TL (Figure 11a). TL offers a promising avenue paving the way for the development of more efficient and personalized closed-loop VNS systems. Future research can further explore novel TL techniques tailored specifically for closed-loop VNS systems to enhance their performance and usability in real-world scenarios.

4.7 Conclusion

In this study, I developed and evaluated an interactive AI framework using RL for automated data-driven design of closed-loop VNS control systems. Multiple simulation environments were created to model different cardiac conditions, and RL algorithms, including PPO, SAC, and PILCO, were employed to autonomously learn the set-point tracking tasks. Our results confirmed that the proposed interactive closed-loop VNS control framework offer a data-driven alternative to classical control methodologies, allowing for continuous learning and the development of precision neuromodulation therapies that autonomously learns and adapts to the underlying dynamics of the cardiovascular system. The integration of transfer learning (TL) was found to improve the sample efficiency of deep RL algorithms, offering the potential for the development of more efficient and personalized closed-loop VNS systems.

Chapter 5

Neuroweaver: a translational platform for embedding artificial intelligence in closed-loop neuromodulation systems

5.1 Introduction

Designing intelligent closed-loop neuromodulation (iCLON) control strategies involves creating a modular design capable of seamlessly integrating with in-vivo experimental setups and computational approaches from multiple domains. This integration is crucial to effectively account for the various components of the system, as illustrated in Figure 5.1. To achieve this, the selection of algorithms may include approaches from multiple domains including digital signal processing (DSP), control, AI and RL domains. Each of these domains offers unique methodologies and techniques essential for a comprehensive research and development platform that enables design and implementation of iCLON systems. The majority of the literature is focused on designing iCLON systems using classical control and signal processing approaches. However, recent advances in AI may enable designing intelligent neuromodulation systems, that are able to learn and optimize neuromodulation control strategies autonomously, via closed-loop interaction with the nervous system. RL is a data-driven approach to design such iCLON control strategies with minimal assumptions and the need for prior knowledge about the underlying physiological dynamics. These properties allow applying data-driven optimization and RL-based control strategies to real-world applications including iCLON systems. However, there are many challenges in designing AI-enabled iCLON systems and translating them in clinical settings including algorithmic design, software implementation, hardware integration, experimental validation, and clinical deployment in implantable devices. These complexities may make designing iCLON systems out of reach for broader biomedical research community and may render designing systems that are not translatable into clinical settings.

Understanding the behavior and optimally designing novel and effective control algorithms requires interactive simulation environments where the brain is in closed-loop with various candidate neuromodulation control and optimization algorithms. Due to the ethical, clinical, and experimental limitations of physical interaction with the nervous system, a promising approach is to employ computational models of neural systems that enables designing, prototyping, and evaluating control algorithms before testing in in-vivo experimental setups. Leveraging mechanistic models as a surrogate of an in-vivo brain is a promising path that enables designing and prototyping various closed-loop neuromodulation strategies. The computational models of brain dynamics can be used to create benchmarks for designing state of the art closed-loop neuromodulation algorithms in different neuromodulation control tasks. In this study, I have designed multiple closed-loop neuromodulation system using advanced data-driven optimization and control strategies and compared them with respect to challenges of integrating iCLON systems in clinical practice including sam-

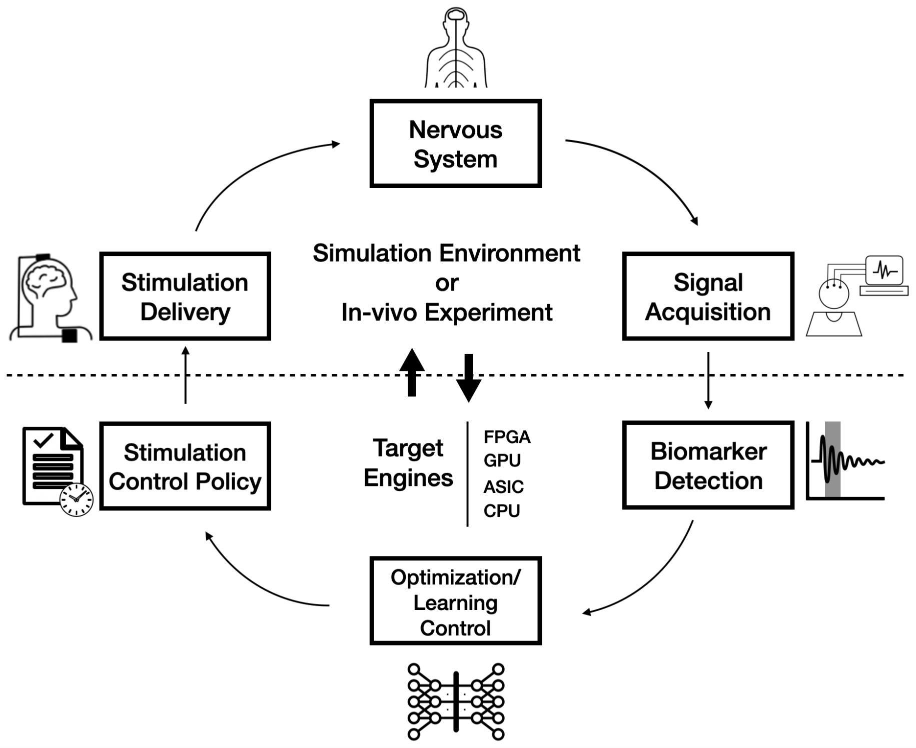

Nervous System
Simulation Environment or In-vivo Experiment
Stimulation Delivery
Signal Acquisition
Stimulation Control Policy
Target Engines
FPGA
GPU
ASIC
CPU
Biomarker Detection
Optimization/ Learning Control

Figure 5.1: Overview of the modular design of an iCLON system supported by Neuroweaver that enables research and development of implantable iCLON systems. Neuroweaver enables the modular design of iCLON systems using simulation environments and computational models of the nervous system for the design and prototyping of iCLON systems. The modular design also allows for seamless integration with the in-vivo experimental setup for evaluating the performance of the designed algorithms. In addition, Neuroweaver enables cross-domain acceleration that not only provides the flexibility of adapting to different algorithmic domains but also improving the performance and efficiency of hardware implementation in designing specialized implantable iCLON systems.

ple efficiency and quality of the learned control policies. These features are designed to emphasize the capabilities of the Neuroweaver platform for designing and prototyping iCLON systems in-silico before integrating in in-vivo experimental setups.

Software implementation of iCLON algorithms requires programming expertise to translate an algorithm to semantically equivalent code for hardware implementation, while also carefully considering synchronization between an interactive prototyping environment and the algorithm. Further complicating implementation challenges are the timing, physical, and energy constraints imposed by real-time interaction with the nervous systems. Clinical implementation of the AI algorithms for iCLON systems may depend on design considerations such as power consumption, form factor, and even operational temperature which cannot be achieved by general-purpose processors and therefore require highly specialized hardware. Furthermore, prototyping and clinical testing of iCLON algorithms require translating high-level programs to operational code on specialized hardware. Just designing a hardware module typically takes years for design experts, let alone the end users without relevant technical expertise. The hardware modules and the associated programming stack need to enable freedom in developing and adopting ever evolving and novel algorithms. This requirement is at odds with the conventional specialized hardware system design practices. These challenges make translating AI algorithms for iCLON systems currently rather infeasible.

To address these challenges, this research introduces an open-source platform, dubbed Neuroweaver [152], for end-to-end designing, prototyping and deploying iCLON algorithms without the complexities of translating AI algorithms to implementation. Although there are various general purpose abstractions for accelerators such as OpenCL [153], CUDA [154], and Weld [155], these frameworks do not incorporate the algorithmic domain knowledge. The Neuroweaver platform specifically enables the end-users to prototype iCLON systems in simulation environments through a

Python-embedded Domain-Specific Language (DSL) framework which compiles algorithms to one or more software frameworks or target architectures, and introduce open-source, flexible accelerators capable of efficiently executing algorithm kernels. The framework is capable of taking the high-level algorithms and efficiently scheduling different compute kernels of the algorithms to different targets while also ensuring valid communication mechanisms for transferring data between frameworks.

In this research, to make progress toward addressing the translational challenges of embedding AI algorithms in implantable iCLON devices, a cross-domain framework enabling multi-acceleration is introduced in section 5.4. Thereafter, I introduce the simulation environment using a computation model of the brain which can be used in closed-loop with different interactive AI-enabled control policies to perform a synchrony suppression task. The computational model of neural population under electrical stimulation is described in section 5.6. This thesis also contributed to the development of a library of RL-based closed-loop control strategies from different classes as described in section 5.7. These algorithms has been used to design, prototype, and evaluate the performance of iCLON systems in a synchrony suppression tasks. This research represents a collaborative effort. Some key lines of work (as described in sections 5.4, 5.5 and the results associated with them in sections 5.8.2, 5.8.3, and 5.8.4) are provided to describe the comprehensive view of the research direction and has been the contribution of our collaborators.

5.2 Challenges and considerations

1. **The need for a familiar programming interface:** The need for a familiar programming interface Even for seasoned software engineers, writing programs for accelerators is non-trivial, requiring careful consideration for low-level hardware attributes to ensure things like memory allocation, operation schedul-

ing, and data communication are appropriately defined. Therefore, to achieve widespread adoption by a broader community, an intuitive, concise, and high-level interface for creating programs is required which minimizes the burden of specifying how the program is executed to the compiler. In addition, the programming interface should use syntax closely resembling the algorithmic notations used in AI, DSP, and Control to further reduce the learning curve for the users who are designing these algorithms instead of involving them in the process of hardware design or compilation. Recall that Neuroweaver flexibly generates the hardware and also compilers the code to that moving parameterized target. Neuroweaver defines such an programming interface through a python-embedded, cross-domain interface (CDI) which captures the shared mathematical notation in each of the target algorithm domains through it's syntax, and relieves the programmer of needing to specify low-level details of how the program should execute. By embedding the programming interface in a widely used language for intelligent and scientific computing (i.e., Python) rather than defining a new, standalone language, the hurdles of installing and setting up additional software are avoided, removing another possible barrier to entry for neuroscientists.

2. **The need to enable research and exploration:** In this work, we take a stance that is different than just offering a chip that is ready to be implanted in brain. We as a community are not there yet. As such, we firmly advocate and offer a framework that enables researchers to explore ideas so that the research community get to a point that can build intelligent brain-implantable devices for neuromodulation.

3. **The need for cross-domain accelerations:** To enable the use of the Machine Intelligence algorithms for biological neuromodulation, there is need for a

computation core that can run not only algorithms from the domain of AI but also Digital Signal Processing (DSP) and Control. That is, the computational core either needs to be general-purpose to accommodate the various domains of algorithm (AI, DSP, Control) or go well beyond the state-of-the-art that only offers specialized accelerator cores that are Domain-Specific Architectures (DSA). Clearly, general-purpose cores cannot accommodate the needs of brain-implantable devices, which require low energy consumption to preserve battery life. Alternatively, DSAs are capable of meeting such requirements, but are unable to adapt to algorithms outside of their target domain. Therefore, the remaining solution is to create a new design point between the flexible, but energy inefficient general purpose processors and the rigid but energy efficient DSAs. Such a solution requires a computation core capable of adapting to AI, DSP, and Control algorithms, while also achieving orders of magnitude improvements in performance and efficiency, called cross-domain acceleration.

4. **The need for algorithm-hardware design space explorations:** The objective of Neuroweaver is not to provide a single architecture for intelligent neuromodulation, rather, to provide a complete framework to explore algorithm-hardware co-design for neuroscientists and clinicians to explore research in building such brain implantable devices. As such, a fixed architecture is incapable of accommodating design space explorations, matching of specific algorithmic needs, and/or operational constraints. Thus, we propose a template architecture that is a highly parametric design, capable of being scaled down or scaled up before fabrication to match the requirements and enable algorithm-hardware design exploration. After explorations and analyses, the Neuroweaver framework can generate a concrete design that can be deployed on Field Programmable Gate Arrays (FPGAs) for prototyping/use in-vivo experimentation. This design is also ready to be fabricated as a stand-alone programmable custom chip

that can be implanted.

5. **The need for compilation to moving targets:** As is the case with all computation cores, a compiler is required to translate an abstraction to executable code. However, by introducing a flexible, parametric architecture, additional responsibility is placed on the compiler to not only perform translation, but also identify optimal architecture parameters depending on the input program. Traditional compilers have achieved a degree of adaptability to different targets by mapping fine-grained intermediate representations to Instruction Set Architectures (ISAs) of general purpose processors. In the case of a parametric architecture, an alternative solution is required because the compiler cannot rely on pre-defined mappings, as different combinations of architectural parameters create different possible mappings. Therefore, the compiler implements the design space exploration by ingesting architecture parameters in addition to the input program, and identifying the parameters for generating an optimal program. Depending on the architecture parameters, the compiler must also adapt the translation of abstraction to executable to abide by constraints imposed by the parameters, such as on-chip storage availability, bandwidth, and varying degrees of parallelism. Compilation for Neuroweaver will have completed when both a set of architecture parameters has been identified, and a valid executable has been generated.

5.3 Neuroweaver in a glance

Designing, prototyping, and experimenting with neuromodulation control systems requires implementing closed-loop analytic pipelines using interoperable modules (Figure 5.1). These systems can be modeled as several interacting modules (Figure 5.1) in computational environments that are often not limited to a single domain as shown in

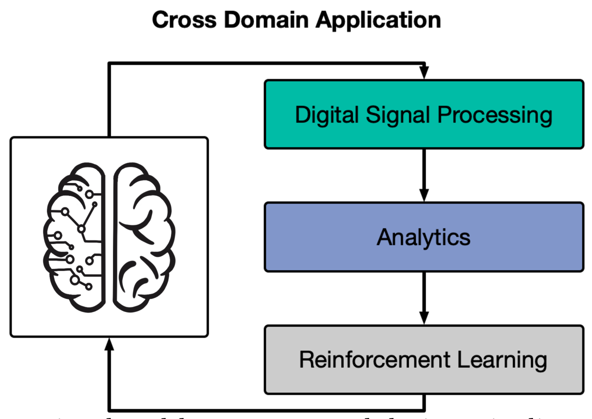


Figure 5.2: A cross-domain closed-loop neuromodulation pipeline. A modular design of iCLON systems include multiple interoperable modules that often include several analytic steps from multiple algorithmic domains including DSP, analytics, RL.

Figure 5.2. Closed-loop neuromodulation pipelines include several analytic steps including biomarker detection and control policy that employ algorithms from multiple domains including DSP, analytics, Deep learning, ML, and RL approaches. However, most of these approaches are computationally expensive, and rely on compute kernels spanning multiple domains of algorithm as well as multiple compute stacks, making their integration into the design of iCLON systems an arduous task and implementation of these pipelines in resource-constrained computational infrastructures such as implantable devices infeasible. To bridge the transnational gap and enable hardware implementation of iCLON systems a framework capable of acceleration of a cross-domain application on different accelerators, called cross-domain multi-acceleration, is required.

To enable programmers to readily develop cross-domain applications using multiple accelerators on FPGA, we devised Neuroweaver, a full-stack framework comprised of a front-end which with a programming interface in Python for cross-domain algorithmic specification and a back-end with the capabilities of domain-specific accelerators. By delineating between front-end algorithm and the possible back-end targets for the hardware implementation of that algorithm, cross-domain end-to-end closed-loop neuromodulation applications can be compiled to multiple heterogeneous accelerators.

To facilitate research and development in implantable iCLON systems, Neuroweaver offers a programming interface within the most popular [156] programming language, Python. The high-level users of the platform use this programming interface in Python to design their novel iCLON systems. The modular simulation environment empowers users to design, prototype, and rigorously evaluate their candidate closed-loop neuromodulation algorithms within a simulated environment. It allows for rigorous testing and refinement of algorithms in a controlled virtual environment, significantly reducing the risks and costs associated with in-vivo experimentation. The users have the flexibility of either using the computational models as described in section 5.6 or simply integrate their own data-driven or mechanistic models related to their application of interest.

The pipeline design in simulation facilitates the transition from the simulation to real-world in-vivo experimentation. By prototyping and fine-tuning algorithms within the simulation environment, researchers gain insights and confidence in their approaches before proceeding to in-vivo experiments. This not only minimizes potential risks but also accelerates the pace of research and development. In addition, the platform offers flexible multi-acceleration features for easier hardware design and implementation which enables and informs the effective design of implantable chips (Figure 5.3).

5.4 Neuroweaver platform

5.4.1 Cross-domain programming interface in python

To enable research and development in implantable iCLON systems, Neuroweaver offers a programming interface within the most popular [156] programming language, Python. Python was selected to minimize the barrier to entry for neuroscience re-

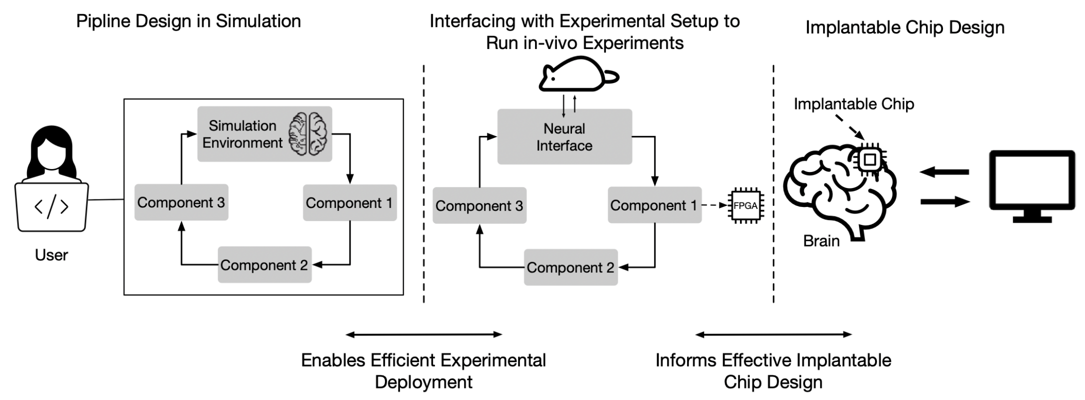

Pipline Design in Simulation
Interfacing with Experimental Setup to Run in-vivo Experiments
Implantable Chip Design
User
Simulation Environment
Component 3
Component 1
Component 2
Neural Interface
FPGA
Implantable Chip
Brain
Enables Efficient Experimental Deployment
Informs Effective Implantable Chip Design

Figure 5.3: Step-by-step conceptual design strategy enabled by the Neuroweaver platform that enables research and development of iCLON systems. The modular simulation environment allows the user to design, prototype, and evaluate their candidate closed-loop neuromodulation algorithms in simulation. This simulation step allows for efficient translation into in-vivo experimentation and flexible hardware implementation which eventually informs the design of brain-implantable chips.

searchers and practitioners instead of forcing them to learn a new language. With the advent of domain-specific accelerators and architectures; there has been significant research on domain-specific languages (DSL) [157, 158, 159, 160, 161, 162], some of which have been also embedded in Python [163, 164]. However, iCLON crosses the boundary of multiple domains and needs a novel interface that can incorporate multiple domains in a Cross-Domain Interface (CDI) for seamless programming and ease of use.

A iCLON application crosses multiple algorithm domains including DSP, Analytics, and AI in each iteration to accomplish it's goal, demonstrating the need for a programming interface flexible enough to express each domain. To implement such an application, the end user would therefore require intimate knowledge of possibly one or more DSLs for each of the different algorithm domains, as well as how to compose the DSLs as a single program with data communication across three different devices. Instead, CDI allows users to write their application as a single program, thus, eliminating the overhead of stitching together different programs and specifying communication across multiple devices. Keeping the properties of target domains in mind, CDI is designed to reduce the time to code a mathematical expression into a formula-based textual format, enabled by the reuse of Python language constructs for modularity and customized Python type annotations. Moreover for code organization and reduction in implementation time, CDI allows the use of Python function decorators to capture user-defined functions as reusable components which perform operations on flows of data.

These Components encapsulate a task comprised of either other component(s) or mathematical expressions which use syntax similar to the targeted algorithm domains to facilitate familiarity for experienced programmers. For modularity and reusability, component(s) have distinct boundaries and arguments which are distinguished by type modifiers defined as Python type annotations, consisting of input, output, state, and param; each of which is associated with how the component will use the argument. By using type annotations in component arguments to explicitly identify data semantics, CDI binds operations to data being operated on, allowing Neuroweaver to determine data reuse and dataflow properties of programs.

CDI uniquely targets multiple domains, each of which is eventually compiled and executed with one of multiple different possible compute cores. As such, CDI offers a light-weight mechanism to specify the target device for component instantiations. The device can also be left unspecified, which allows the compiler to select the compilation target based on the available resources, as well as device annotations which might be included in sub-components in the program. This relieves the programmer from having to annotate all components within their program. More details on the programming abstractions for cross-domain multi-acceleration is provided in [165].

5.4.2 Multi-target cross-domain compilation

Providing a high-level programming interface is necessary for enabling usability amongst neuroscientists, but it also requires a compiler capable of scheduling and generating code in the absence of low-level details included in other programming languages. To facilitate algorithm-hardware design space exploration, the compiler must also have the capability to adapt it's workflow to different compilation targets with distinct characteristics. The Neuroweaver compiler achieves these requirements by including architecture parameters associated with the compilation target as inputs, in addition to the high-level program. By using architecture parameters as inputs, compilation can be repeatedly invoked for different compilation targets with a given program until the optimal set of parameters is found, based on performance estimates. Finally, once the architecture parameters are selected, the Neuroweaver compiler is capable of generating code for the selected target.

CDI programs are defined with minimal specification for how the program should be executed, to allow usability for neuroscientists. However, CDI preserves the dataflow required for execution in it's programs by creating a hierarchical dataflow graph. By using a hierarchical approach, we preserve different levels of operation granularity, which is required for adapting to different compilation targets with support for different operations. By combining the graph with input architecture parameters, the compiler traverses the graph nodes and identifies operations supported by the architecture parameters and maps them to the equivalent code templates used for scheduling.

5.5 An example implementation with CNF program using the CDI in Python

Neuroweaver provides a simple and lightweight programming interface, called Component and Flow (CNF) programming model, which allows application programmers to specify various components of their program to be targeted for acceleration. Neuroweaver is also equipped with a runtime system which creates a component and flow graph, schedules the components and handles the data transfers between components.

These characteristics are demonstrated in the following example, which implements a very simple example consisting of two components connected together with a defined flow and will be used to delve further into CDI.

Components: A programmer creating a new component for their application extends the `Component` baseclass. The programmer optionally defines one or more of the following four methods with the `@property` decorator: The `input_names`, the `output_names`, the `state_names`, and the `property_names`. Each of them return a list of strings which are the respective queue names. The programmer also extends the baseclass with the following two methods: The `initialize` , and the `execute` methods.

In the following toy example first a simple component called Brain is created (lines 4-15). It has one output queue, called `brain_signal`. The `initialize` method can be used for code to set up the initial state of the component and perform other one time tasks The `execute` method contains most of the functionality of the code. The `execute` function runs every time an input is available on any of the input queues.

The following example then creates another component called `SignalSink` (lines 17-27). This component will receive a `brain_signal` input. The `Component` class provides the following methods which initialize the queues for each of the interfaces with a shape.

The `Component` class provides the following methods which initialize the queues for each of the interfaces with a shape: `set_input`, `set_output`,and `set_state`. The shape is provided as a tuple, e.g. $(100, 1)$ for a 1-D array with 100 elements. Every element pushed to the queue needs to be this particular shape. The following code snippet initilizes `Brain` and `SignalSink` components. The output queue `brain_signal` of `brain` instance is set to aceept shape $(1, 1)$. The input queue `brain_signal` of sink instance is set to accept shape $(1, 1)$ as well (lines 36-38).

Graph: The graph class allows the programmer to declare a component and flow program. The `add_component` method is used to add components to the graph. In the following example, we add `brain` as a component to a graph called `graph`. The `add_flow` method is used to connect two components with a flow. We specify source and destination component instance along with their corresponding queue names. In the following example, we add a flow to a graph called `graph` by specifying the source component brain with its output queue name `brain_singal` and the destination component with its input queue name `brain_signal` (lines 40-42).

The following code adds `Brain` and `SignalSink` components to a graph called `deep_brain_stimulation`. Their queues are initialized. A flow is created between the two components (lines 30-45).

Runtime: The runtime is initialized with an instance of the component and flow graph object. The `initialize` method of the graph class is used to initialize all the components in the graph. The `execute` method begins the execution. The runtime will schedule ready components and do the data transfers according to the flows specified (lines 47-51).

With the all the pieces from the previous sections, we now have a runnable CNF program with two components.

```python
1  from runtime.runtime import Runtime
2  from qfdfg.graph import Graph
3  
4  class Brain(Component):
5      @property
6      def output_names(self) -> List[str]:
7          return ["brain_signal"]
8  
9      def initialize(self):
10         print("brain-v0")
11 
12     def execute(self, brain_signal):
        signal = np.array([10])
        print(f"{self._name}: signal {signal}")
        brain_signal.push(signal, (1,1))

class SignalSink(Component):
    @property
    def input_names(self) -> List[str]:
        return ["brain_signal"]

    def initialize(self):
        print(f"initialize does nothing")

    def execute(self, brain_signal):
        signal = brain_signal.pop()
        print(f"{self._name}: signal {signal}")

# Create a component and flow (CNF) graph. This is named
   DeepBrainStim
deep_brain_stimulation = Graph("DeepBrainStim")

# Initialize the Brain and SignalSink components
brain = Brain()
sink = SignalSink()

# Bind the output and input names to Queues
brain.set_output("brain_signal", (1,1))
sink.set_input("brain_signal", (1,1))

# Add components to the graph
deep_brain_stimulation.add_component(brain)
deep_brain_stimulation.add_component(sink)

# Create a flow between the brain component and the sink component
```

```python
deep_brain_stimulation.add_flow("brain_signal", brain, "brain_signal
    ", sink)

runtime = Runtime(deep_brain_stimulation)
# Run the initialize functions of all components
runtime.initialize()
# Begin execution
runtime.execute()
```

Listing 5.1: Example of a program using the cross-domain programming interface in Python.

5.6 Simulation environments and control tasks for designing iCLON systems

Multiple simulation environments has been integrated to enable designing iCLON systems in simulation. These simulation environments enabled the design and prototyping of iCLON systems in-silico before integrating with in-vivo experimental setups. Details of the simulation environments using computational models of the brain are provided in the following subsections.

5.6.1 Interactive AI-enabled closed-loop synchrony suppression in Bonhoeffer–van der Pol model

Pathological synchronous network activities in the brain is hypothesised as a potential source of many neurological disorders like Parkinson's disease [166]. The collective synchronous activity of neural ensembles can lead to symptoms such as tremor. DBS modulates the desired functionality of the neural systems through locally delivering stimulation to the targeted brain regions. Here, we consider to use a population of N regularly oscillating neurons, i.e. Bonhoeffer–van der Pol also known as

FitzHugh–Nagumo oscillators, globally coupled via the mean field X which is implemented as an OpenAI gym environment [167]. I employed this computational model as a simulation framework to design closed-loop neuromodulation systems using different classes of RL algorithms during a synchrony suppression task. The regularly oscillating neurons in Bonhoeffer–van der Pol model follow the equations below:

$$\left\{ \begin{array}{l} \dot{x} = x_k - \frac{x_k^3}{3} - y_k + I_k + \epsilon X + A \\ \dot{y} = 0.1(x_k - 0.8y_k + 0.7), \end{array} \right\} \tag{5.1}$$

where $X = \frac{1}{N}\sum_{k=1}^{N} x_k$ is the mean field, A is the action stimuli applied to each individual neuron $k = 1, \ldots, N$ of the total N neurons. Actions, are considered to be ideal δ-shaped pulses with the amplitude $-A_{max} \leq A_t \leq A_{max}$ which gets updated at each time step $t_n = n\Delta$ and Δ is the sampling rate of the environment.

The state of the environment at time step t is the value of the mean field model, i.e. $X(t)$. We used the exponential reward function as in:

$$R(t) = exp^{(-(X(t) - <X_{state}>)^2 - \beta \|A_t\|)}. \tag{5.2}$$

In equation 5.2, $< X_{state} >_t = \frac{1}{M}\sum_{l=1}^{M} X(t-l+1)$ consists of M most recent values of the mean filed and it is considered to account for the oscillatory activity of the neural populations. The total energy supplied to an ensemble of neurons is a measure that we aim to minimize in practical DBS settings and the second term in equation 5.2 is added in favor of minimizing the total stimulation energy.

This simulation environment, which is a mean field model of neural population activity with electrical stimulation, was used to design and develop an interactive AI-enabled iCLON systems for sunchrony suppression. Multiple interactive iCLON strategies was designed as illustrated in Figure 5.4. The performance of five different RL-based control strategies (see section 5.7) in performing the synchrony suppression task was evaluated in terms of the quality of learning the task and sample efficiency which are two key factors in designing iCLON algorithms.

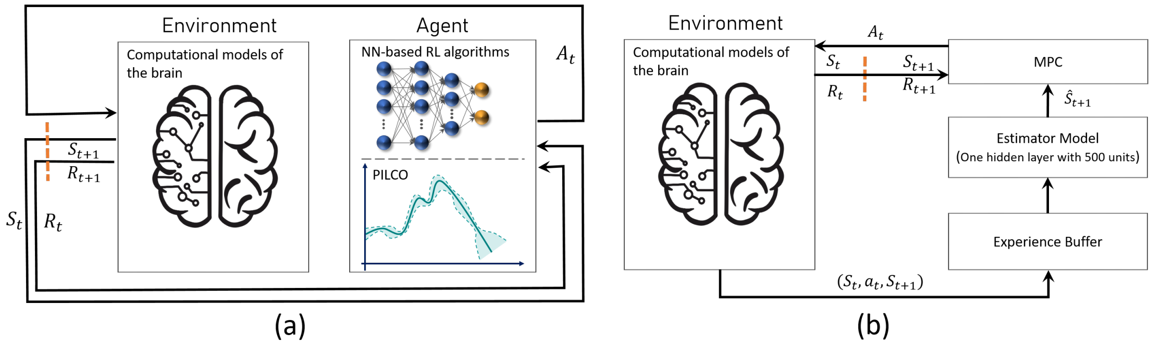


Figure 5.4: The modular architecture of the Neuroweaver simulation environment including the computational model of the neural population under electrical stimulation in closed-loop with RL-based control strategies to learn a synchrony suppression control task; (a) neural network-based model-free algorithms and PILCO, (b) Model-based RL with MPC.

5.7 RL algorithms integrated in the design of iCLON systems

A standard RL task can be formulated as a Markov Decision Process (MDP) [168] defined by a tuple (S, A, r, T, P), where S and A are state and action spaces, $r = R(\tau)$ is a reward function, T is the set of terminal conditions, and P is the state transition probability. The general goal of reinforcement learning algorithms are to find the optimal policy π maximizing the discounted cumulative expected reward ,$J(\pi)$, as follows.

$$\pi^\star = argmax_\pi J(\pi), where J(\pi) = E_{\tau\sim\pi}[R(\tau)] \tag{5.3}$$

We provide an algorithm library for designing data-driven intelligent control strategies which consists of candidates from multiple classes of RL algorithms. One of the most important factors in dividing RL algorithms into different categories, is whether the RL agent has access to or can learn an underlying model of the target environment, leading to two broad categories of model-based and model-free algorithms. In model-based RL, having a model allows the agent to plan ahead by forecasting outcomes of various choices of action, enabling it to derive a learned policy from these projections. This results in substantial sample efficiency in model-based RL compared to model-free RL algorithms. However, having access to a precise model of the target environment is often not practical. In the absence of a ground-truth model, the agent has to learn the transition model from experience, which can lead to biases causing sub-optimal and poor performance in the actual environment. Model-free algorithms, on the other hand, despite missing out on sample efficiency benefits, are simpler to implement and tune, making them more popular and extensively explored compared to model-based approaches.

Model-free RL algorithms are mainly categorized as policy optimization and Q-learning methods. The policy optimization approaches optimizes policy parameters θ either with respect to the actual performance objective $J(\pi_\theta)$ or its local approximations in an on-policy way. However, Q-learning methods are off-policy strategies that learn an approximator of the optimal action-value function. Policy optimization methods tend to be more stable and reliable at the cost of being less sample efficient compared to Q-learning approaches since the off-policy agents utilize the replay buffer containing the old experiences in contrast to the on-policy agents. Many RL algorithms has been developed which are able to carefully trade-off between the strengths and weaknesses of either side.

The selection of a control policy in designing intelligent closed-loop neuromodulation systems heavily relies on the dynamics of the underlying nervous system, existence of prior knowledge about the dynamics of the environment and the effect of stimulation, dimensionality of the action and state spaces, and more. In addition, complexities of the interactions between the neuromodulation systems and the nervous system as well as ethical constraints avoids us from testing any novel closed-loop strategies in in-vivo experimental setups. Thus, I provided a library of RL algorithms including model-free, model-based, on-policy, and off-policy RL algorithms for testing in closed-loop using the the computational model described in section 5.6 for in-silico prototyping of control strategies. We show the utility and the extensibility of this in-silico simulation environment in providing insight on the behavior of RL algorithms in the context of a neuromodulation tasks in terms of speed of convergence and the quality of learning the optimal control policies which are two important performance metrics in employing RL algorithms in clinical practice. The high-level explanations of the RL algorithms are provided in the following sections. The high-level overview of the modular simulation environment including the computational model in closed-loop with RL agents is depicted in Figure 5.4.

I deployed three model-free RL algorithms in closed loop with the simulation environment, including proximal policy optimization (PPO) [169], soft actor-critic (SAC) [170], and Deep Deterministic Policy Gradient (DDPG) [171] to evaluate the feasibility of utilizing model-free RL approaches in intelligent closed-loop neuromodulation control systems. The advantage of model-free RL algorithms over more complex methods is that they do not rely on constructing a sufficiently accurate environment model and hence, their performance are not affected by model bias. These three algorithms can be divided into two main categories: on-policy (e.g., PPO) and off-policy (e.g., SAC and DDPG).

5.7.1 Proximal policy optimization

PPO is a policy gradient (PG) method that has shown high quality of performance in many applications. PG algorithms are a type of RL algorithms that rely upon optimizing parametrized policies with respect to the expected long-term return using gradient descent. Unlike vanilla PG [148] that keep new and old policies close in the parameter space, trust region policy optimization (TRPO) [149] algorithm updates policies by taking the largest step possible to enhance the performance while satisfying a constraint expressed in terms of KL-Divergence on how close the new and old policies are allowed to be. PPO combines the advantages of vanilla PG and TRPO to ensure stability and scalability by employing a surrogate objective function to update the policy parameters.

In this study, we employed PPO-Clip, a variant of PPO that utilizes specialized clipping in the objective function to prevent significant deviations of the new policy from the old policy. As a result, PPO offers a simpler implementation, while empirically performs at least as well as TRPO. PPO is applied in an actor-critic framework. The actor maps the state to action and the critic gives an expectation of the agent's reward with its corresponding state. The policy is updated via a stochastic gradient ascent optimizer to ensure the exploration while the agent will gradually tend to exploit what it has learned over the course of training. Here, we used stable baseline library [150] for implementing PPO algorithm.

5.7.2 Soft actor-critic network

SAC [151, 143] is an actor-critic RL algorithm widely employed in continuous action spaces for various control tasks. SAC is from the family of off-policy RL algorithms that optimizes a stochastic policy. A central feature of SAC is entropy regularization to encourage effective exploration during learning. The policy is trained to maximize a trade-off between expected return and entropy, a measure of randomness in the policy, which has a close connection to the exploration-exploitation trade-off. Increasing entropy results in more exploration, which can accelerate learning. It can also prevent the policy from prematurely converging to a bad local optimum, resulting in stable training. Moreover, SAC leverages neural networks to represent both the policy and the value functions, enabling it to handle high-dimensional observation spaces effectively. We employed Stable Baselines library [150] for implementing SAC algorithm.

5.7.3 Deep deterministic policy gradient

DDPG is a deep variant of the deterministic PG algorithm, which can also be viewed as an actor-critic algorithm, using a Q-function estimator to enable off-policy learning, and maximizing this Q-function by an actor. Since DDPG is a deterministic policy, we add adaptive noises to the parameters of the neural network to encourage exploration [172]. We employed Stable Baselines library [150] for implementing DDPG algorithm.

5.7.4 Model-based reinforcement learning with model predictive control

To improve sample efficiency which is a critical factor in iCLON systems, we investigate model-based RL algorithms. The first method is a combination of a model-based RL algorithm with model predictive control (MPC). Fig. 5.4(b) shows the high-level overview of this method that consists of two components: learning the underlying dynamics of the environment, and using a MPC controller to plan and execute actions. To approximate the state transition model of the simulation environment, we initially collect random trajectories and add the history of collected samples to the experience buffer. The estimated dynamical model $\hat{f}$ is formulated as $\hat{s}_{t+1} = s_t + \hat{f}_\theta(s_t, a_t)$, where s_t and a_t are the state and action at step t respectively, following the setting in this work [173]. We used as a neural network to model the dynamics, where the parameter vector θ represents the weights of the neural network, aiming to minimize the mean squared error $\xi = 1/D \sum_{(s_t,a_t,s_{t+1})\in D} ||\delta - \hat{\delta}||^2$ between the observed difference of two consecutive time steps, i.e. $\delta = s_{t+1} - s_t$, and the model predictions, i.e. $\hat{\delta} = \hat{f}_\theta(s_t, a_t)$. After initialization, MPC selects the next actions to be evaluated with the goal is to minimize the cost function to achieve the synchrony suppression task. The cost function is in the same format as the reward function in equation 5.2 with the negative sign and the different value of β.

5.7.5 Probabilistic inference for learning control

Probabilistic inference for learning control (PILCO) [145] is model-based data-efficient approach to policy search without considering any prior domain knowledge about the underlying dynamic. Model-based RL approaches often assume that the learned dynamics model is sufficiently accurate which will lead to low performance in the presence of model bias. Model bias is particularly an issue in cases where there is limited prior knowledge or limited data available. PILCO employ Gaussian process (GP), a non-parametric probabilistic model [114], that takes the model uncertainty into account to address the model bias issue. The main advantage of PILCO is that it remarkably improves the sample efficiency in continuous state-action spaces which sets the pathway of integration of PILCO in closed-loop clinical settings and experimental setups.

Consider the following dynamical system $x_t = f(x_{t-1)}, u_{t-1})$, Where f is the unknown state transition function with continuous state, x, and action, u, domains. The goal of PILCO is to find a deterministic policy that maximizes the expected return or minimizes the expected cost, $c(x_t)$ of following the policy π over the time horizon T as in $J_\pi(\theta) = \sum_{t=0}^{T} E_{x_t}[c(x_t)], x_0 \sim N(\mu_0, \Sigma_0)$. PILCO assumes that π is a function parametrized by θ and that the cost function $c(x)$ encodes some information about a target state x_{target}. We used the squared exponential cost function.

5.8 Results

5.8.1 Synchrony suppression using reinforcement learning algorithms

We have created a suite of simulation environments using a biophysical models of brain stimulation to design and test intelligent closed-loop neuromodulation systems. We employed the mechanistic model of the population of N regularly oscillating neurons, i.e. Bonhoeffer–van der Pol model implemented in the format of OpenAI gym which is a standard environment for designing intelligent agents. Using this setup, we demonstrated in-silico experiments to design iCLON systems using the state-of-the-art RL algorithms for suppressing pathological synchrony. We evaluated the performance of five different RL algorithms in the synchrony suppression task in terms of the quality of learning the task and sample efficiency which are two key factors in designing iCLON algorithms. We used the stable baseline library for implementation of the model-free RL algorithms, i.e. PPO, SAC, and DDPG. In our implementations, we used two hidden layers MLPs with 64 neurons for all three model-free RL algorithms. Fig. 5.5(a) shows the reward function during the training phase of the three model-free algorithms. As shown in Fig. 5.5(a), SAC and DDPG which are off-policy algorithms converge faster compared to PPO which is an on-policy method. However, PPO achieves a higher reward and a better final performance in learning the synchrony suppression task at the expense of more interaction with the neural environment. The quality of performing the task after convergence is shown in Fig. 5.6(a)-(c), where the first 7500 samples are showing the oscillatory behavior of the neural populations without taking any action, i.e. applying any stimulation pulses. The middle 7500 samples is showing the neural states by taking actions with the learned RL policies, and the last 7500 samples show that the population of neurons start synchronizing again if we stop the intervention.

We further expanded the library of algorithms by adding two model-based approaches to improve sample-efficiency. The first approach is model-based RL with MPC. In our implementations we used a single-layer MLP with 500 neurons for modeling the underlying neural dynamics. Fig. 5.5(b) show the reward value of the MPC approach. Although model-free methods require at least $1e6$ samples to converge, the model-based RL with MPC has shown improvement in terms of sample efficiency and converges at around $2.5e4$ steps. However, there is still room for improvement in terms of its final performance in the synchrony suppression task as depicted in Fig. 5.6(d) and having $2.5e4$ interactions with the nervous system might still be impractical for in-vivo experiments.

The next model-based RL algorithm that we tested is PILCO. As shown in Fig. 5.5(c), after a random initialization phase of length 300 steps, the RL agent quickly converges and learns the synchrony suppression task as shown in Fig. 5.6(e). Although the best final performance in terms of synchrony suppression and minimizing the power of actions is achieved by PPO, but that is at the cost of having at least $3e6$ interactions with the environment that is not practical for being integrated in iCLON systems in clinical practice. On the other hand, PILCO shows a noticeable improvement in sample efficiency at a cost of consuming slightly higher action power and slightly higher level of synchrony which is much more well-suited for clinical practice. In general, our evaluations support the hypothesis that RL algorithms are capable of handling the decision-making process in closed-loop neuromodulation control systems. In addition, we showed the utility of simulation environment in designing and prototyping iCLON systems in silico before testing in in-vivo experiments.

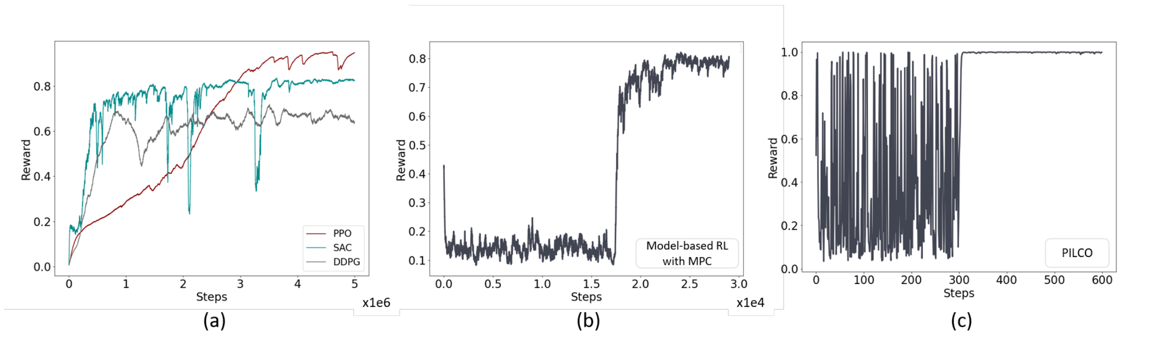


Figure 5.5: Learning performance (reward values) of different RL-based iCLON systems using (a) Deep RL algorithms, (b) model-based RL with MPC, and (c) PILCO.

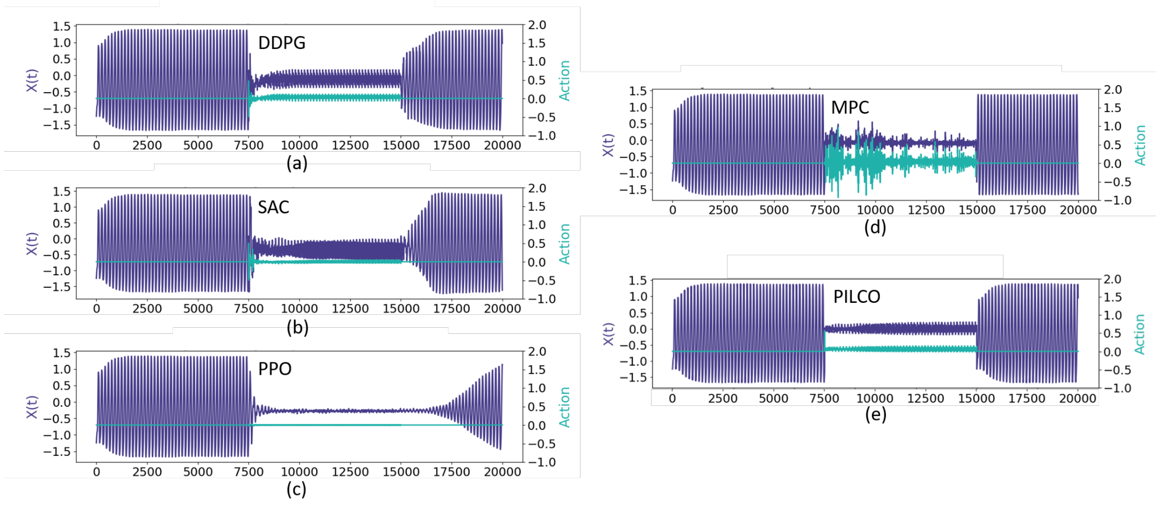


Figure 5.6: Performance of different RL algorithms in the synchrony suppression task during the inference mode.

5.8.2 CNF implementation of iCLON systems using deep RL algorithms

The three iCLON system designed to perform the synchrony suppression task using deep RL algorithms have been implemented using CNF program. A modular design has been considered to separate the training from inference mode of the RL agents as depicted in Figure 5.7. The main reason behind separating the training and inference mode is that the RL policies in inference mode are being implemented on FPGA (see Figure 5.7, and section 5.8.3). In addition, the runtime latency measurements of these implementations are added to table 5.1.

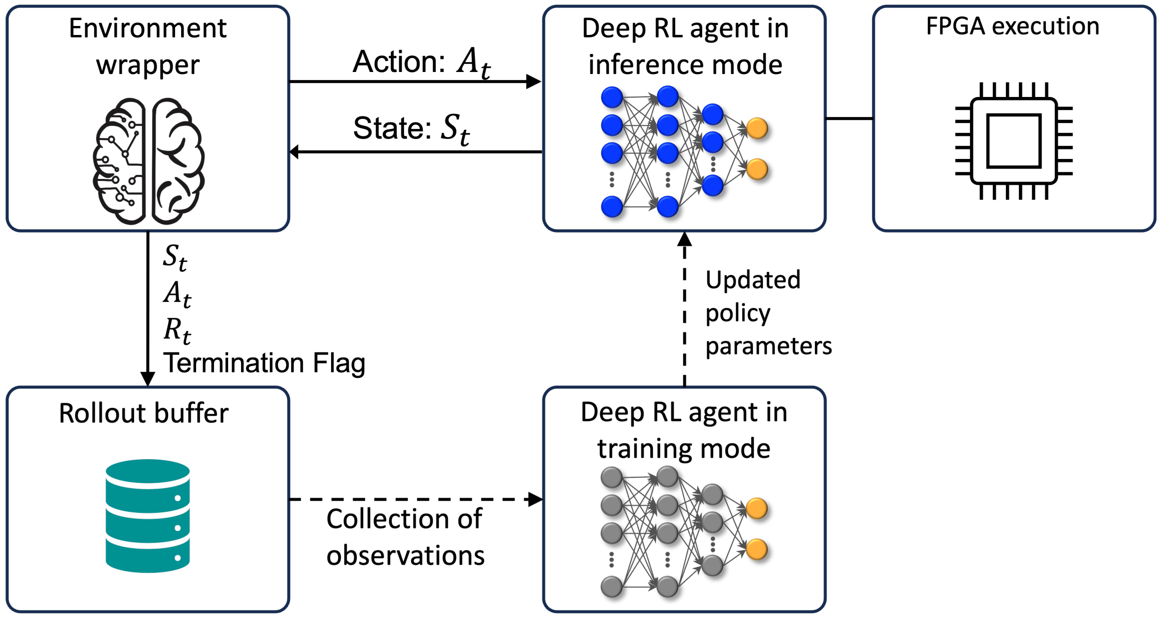

Environment wrapper
Action: A_t
State: S_t
Deep RL agent in inference mode
FPGA execution
S_t
A_t
R_t
Termination Flag
Updated policy parameters
Rollout buffer
Collection of observations
Deep RL agent in training mode

Figure 5.7: Modular implementation of RL-based iCLON systems using Deep RL algorithms. This modular design separates the training from inference modes. The RL policy in inference mode is targeted to be implemented on FPGA.

Table 5.1: Runtime latency measurement in ms for CNF implementation of the three iCLON systems for synchrony suppression using deep RL algorithms.

	Gym Environment	RL Inference	Rollout Collection	End-to-End latency
PPO	2.67	3.43	9.45	28.4
SAC	2.58	1.18	1.34	5.93
DDPG	2.61	1.24	1.28	6.17

5.8.3 FPGA execution of deep RL agents in inference mode

The Deep RL policies during the inference mode was implemented in SystemVerilog. We used Xilinx Vitis tools to synthesize and implement the design on a Xilinx U280 FPGA. The FPGA prototype sustains an operating frequency of 100 Mhz. We compare our hardware implementation with Mr.Wolf, a publicly available, state-of-the-art fully programmable low-power SoC. Mr Wolf consists of a two-stage RISC-V low-power processor for system control and a compute cluster of 8 RISC-V cores as an accelerator for compute-intensive workloads. It is further augmented by the XPulpNN ISA extension that enables SIMD arithmetic operations on INT8 data type. Mr Wolf SoC consumes 153mW at 350MHz with an active Compute Cluster.

Quantization is a fundamental approach to enabling performance, energy/memory efficient Neural network inference on low-power architectures. Although there is extensive research on the effect of quantization on accuracy for DNN, there is far less research on how quantization affects the accuracy of RL networks. Recent works [174] demonstrated that post-training quantization has minimal effect on network accuracy for a wide range of RL networks. Based on the open-sourced evaluation framework from QuaRL [174], we quantized the benchmark networks using the PyTorch quantization tool and evaluated its effect on network accuracy. Our evaluation confirmed the conclusion of previous work. Based on this finding, we use these quantized networks for compilation and performance evaluation on our FPGA prototype. We use NEMO, a quantization tool developed by the same research group that developed Mr. Wolf to o execute the quantized network on Mr. Wolf.

We evaluate the FPGA implementation through iso-power inference latency comparison with the baseline. The inference latency is measured in cycle count. To measure the latency on the FPGA prototype, we first compile the benchmark networks' ONNX format into executable binaries using the Neuroweaver compiler. The compiled binaries and the network parameters are then stored in main memory buffers shared with FPGA implementation using CDI and PyOpenCL. We launched the inference task through CDI and a performance counter module integrated with the cycle of execution during inference. Once the inference is done, the performance counter passes the cycle count statistics back to the host through the CDI.

We obtained the performance number of Mr. Wolf using GVSoC, an open-source software simulator developed by the same research group that developed Mr. Wolf. GVSoC targets the full-platform simulation of the Mr. Wolf SoC and has less than 10% simulation error for performance analysis compared to an actual physical implementation. To obtain the best possible performance on Mr.Wolf with GVSoC, we employ the Neural Network deployment framework native to Mr.Wolf which consists of NEMO, a network quantization tool, DORY, a neural network deployment tool that generates memory-optimized code for neural network workloads, and XPulpNN, a RISC-V ISA extension that enables low-bitwidth arithmetic SIMD vector instruction for Neural Network workload. We use the same set of benchmark network ONNX format as inputs to the NEMO quantization tool and execute the binaries generated by DORY on GVSoC. We keep Mr. Wolf's hardware configuration the same as reported in the paper for iso-power comparison. We also simulated standalone network kernels such as Matmul and Relu provided by XPulpNN to cross-validate the end-to-end performance number obtained using the NEMO/DORY framework.

The experimental results of this analysis is provided in table 5.2.

Table 5.2: Comparison of layer-by-layer and end-tot-end execution of Deep RL algorithms in inference mode on FPGA compared to XPulpNN in terms of speed up.

Workload	Sum of layer-by-layer results on FPGA	Projected end-to-end results on FPGA
PPO	2.95x	3.54x
SAC	4.45x	4.61x
DDPG	4.75x	4.87x

5.8.4 In-vivo experiments

To demonstrate the capabilities of the Neuroweaver platform in integration with in-vivo experimental setups, we ran real-time in-vivo experiments using Neuropixels multi-channel probes to control an LED light based on the theta-band power of the LFP signals recorded from the hippocampus in rats (Figure 5.8. The Neuropixels probes collected LFP signals with a sampling frequency of $30KHz$, bit depth of 16 and bit volt of 0.195μV. The recoreded LFP signals was filtered to remove low frequency LFP data and high frequency spiking data, and the LFP signal was subsequently down sampled into $2.5KHz$. Data from Neuropixel probes was streamed into Open Ephys software on the recording PC, which recorded all data and streamed selected LFP channels to our platform.

A workflow in Neuroweaver platform was created for the real-time closed-loop system. The workflow consisted of a pacemaker component that creates dummy input to all data collecting components to drive their execution. Data collecting components included 2x Open Ephys interface components that received data from each probe sent by separate ZeroMQ plugin in Open Ephys, and a camera component that communicated with FILR Flea 3 camera with Spinnaker SDK provided by FILR.

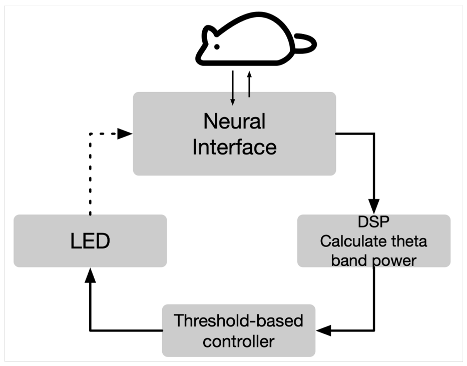
The Open Ephys interface components stored the collected data in a buffer and pushed a data packet of 4,096 samples downstream every 256 samples (both numbers are adjustable parameters). They also recorded and logged the timestamp when the Open Ephys ZeroMQ plugin encoded the last packet of data, marking when data became available. The camera interface component received every frame taken by the camera and its timestamp. The frames were stored into a MJPG video file encoded with OpenCV on disk, and also displayed a real-time monitor window on screen. The timestamps of each frame were also logged.

Data processing for this experiment consists of a simple FFT component that picked the data of one channel (data from all received channels are sent to the processing component), performed FFT, and calculated the power within the theta frequency band ($5 - 9Hz$). This power was logged, pushed into a separate component that displayed a real-time plot on screen, and sent to the stimulation algorithm downstream. Here the stimulation algorithm was simple threshold-based control.

The stimulus presentation in this experiment was an on-screen display of an indicator light turning green or red. The latency overhead by Neuroweaver in in-vivo experiments was measured to be $38.26 \pm 7.69ms$ during a 20 minutes recording session.

5.9 Discussion

We presented an open-source platform, dubbed Neuroweaver, for end-to-end designing, prototyping and deploying translatable intelligent closed-loop neuromodulation



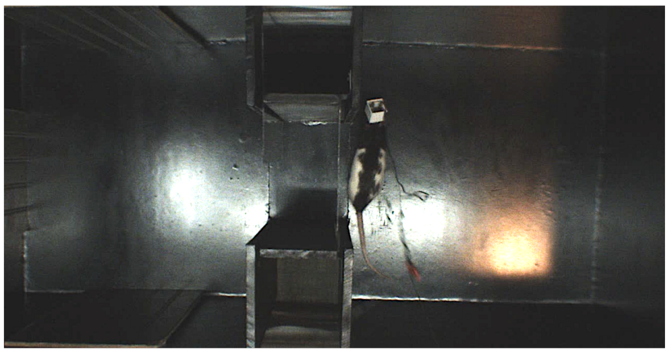


Figure 5.8: In-vivo experiments. (a) Schematic of the experimental design where the LFP signals are being recorded from hippocampus in rats, DSP techniques was used to preprocess the recorded LFP signals and calculate the theta-band power. A simple threshold-based controller based on the theta-band power is used to turn an LED on/off. The LED is used as a surrogate of stimuli to close the loop.

systems. The main purpose of Neuroweaver is to bridge the translational gap between designing and deploying clinically useful iCLON systems from multiple perspectives. First, it provides a simulation environment using computational model(s) of the neural systems for designing, testing, and prototyping closed-loop neuromodulation systems before deploying in clinical or in in-vivo experimental settings. Second, it provides libraries of algorithms from multiple domains including RL, signal processing and machine learning to enable modular design of closed-loop pipelines. Finally, Neuroweaver enables cross-domain multi-acceleration to enable developing implantable iCLON systems.

In this study, I demonstrated the utility of Neuroweaver in experimental design of iCLON systems using a computational model of the brain under electrical stimulation. The Performance of different RL-based iCLON strategies (including model-based, model-free, on-policy, and off-policy approaches) have been evaluated in a synchrony suppression control task. The utility of employing the RL algorithms in iCLON systems considering two metrics including speed of convergence as in Fig. 5.5 and the quality of learning task as shown in Fig. 5.6. This model represented an example of physiological models that can be used for designing iCLON systems. The Neuroweaver platform is capable of incorporating different computational models as surrogates of the target physiological system. These models may include mechanistic biophysical models or data-driven models that are built from the experimental data. Although we showed the utility of Neuroweaver in experimental design using the simulation framework, adding compatibility to multiple well-developed libraries of computational models is a promising future direction.

Neuroweaver provides a framework to explore algorithm-hardware co-design for a broader community including neuroscientists, clinicians, and engineers to explore research in building such brain implantable devices. FPGA acceleration has been implemented for multiple RL-based iCLON systems as described in section 5.8.3. The flexible cross-domain multi-target acceleration capabilities of Neuroweaver enables the users to explore the design of novel iCLON systems. Although, hardware acceleration on FPGA has been demonstrated for multiple algorithms, these capabilities need to be expanded for a wider range of algorithms to provide more flexibility to non-expert users.

In this study a prototype closed-loop experiment was implemented using CDI, compiled, and executed the simulation using Neuroweaver workflow to interface with an in-vivo experimental setup. The results showed that Neuroweaver workflow does not add a significant computing overhead to the execution of the closed-loop pipelines. This is an important factor especially for the real-time implementation of iCLON algorithms.

5.10 Conclusion

This research presented Neuroweaver, an open-source translational platform for embedding AI in iCLONs system. The platform is aimed to bridge the translational gap between research studies and clinical practice. Neuroweaver not only provides a simulation environment for modular designing and prototyping closed-loop neuromodulation pipelines, but also minimizes the complexities of translating the algorithm to implementation through a Python-embedded CDI which compiles algorithms to one or more software frameworks or target architectures. The utility of Neuroweaver in designing and prototyping iCLON systems was demonstrated by designing multiple RL-based iCLON strategies. Moreover, the results showed that implementation and execution of the closed-loop simulations using Neuroweaver workflow does not add execution overhead which is critical for real-time implementation of iCLON interfaces. Integrating hardware acceleration on FPGA showed an improved execution time compared to other platforms. This hardware acceleration is aimed to be performed with minimal complexity of translating the algorithms to implementation for the end-users to enable exploration of embedded implantable iCLON systems for a broader community.

Chapter 6

Conclusion and future direction

Despite the challenges faced in developing and implementing the iCLON systems, the potential benefits of utilizing them to treat neurological disorders make this field of research and development highly promising. These systems hold the potential to provide more targeted, flexible, and effective treatment options, reduce the burden of medication side effects, and improve the overall quality of life for patients. This thesis significantly contributed to the development of intelligent precision neuromodulation therapies that has the potential to revolutionize the standard of care.

6.1 Contributions to the field

This research significantly advances the field of closed-loop neuromodulation and contributed to the development of iCLON systems that autonomously learns and adapts to the intricate and varying nature of the target nervous system. The key contributions of this thesis are described below.

This thesis contributed to the development of multiple simulation environments. These environments, essential for designing iCLON systems, provide a safe and controlled setting for testing and prototyping these systems in silico before integrating in experimental setups. They enable rapid design iterations, facilitating the transition from concept to clinical application. This approach addresses the challenges of interacting with the complex and sensitive nervous system, ensuring safety and efficacy.

Introduction of control tasks to facilitate the development and clinical translation of iCLON systems is another contribution of this research. The thesis introduces specific control tasks that replicate real-world clinical scenarios. These tasks include designing systems for minimizing tremor severity in PD and ET patients, regulating HR and MAP in cardiovascular system with selective VNS, and suppressing neural synchrony in PD.

Furthermore, this research contributed to the advancement in control policies for designing novel and more effective iCLON systems. Moving beyond trial-and-error and traditional control approaches, this research leverages AI and RL to develop data-driven control strategies. These strategies enable iCLON systems to autonomously learn and optimize neuromodulation controls, offering a more effective, adaptive, and patient-specific approach to treatment.

Implementation of novel data-driven control strategies in the context of multiple neuromodulation applications is another key contribution of this research. I successfully applied these innovative control strategies in various neuromodulation applications, including automated DBS programming framework with safety constraints for tremor suppression, cardiovascular system regulation with selective VNS, and synchrony suppression with DBS in PD. This demonstrates the generalizability and effectiveness of the developed systems across different neuromodulation applications.

Another contribution is the development of an end-to-end translational research platform for the design and implementation of iCLON systems which has been a collaborative effort. This platform incorporates an algorithm-hardware co-design approach, facilitating the development of brain-implantable devices that can learn and adapt control policies autonomously. It addresses the computational challenges of AI and reinforcement learning, offering a scalable and customizable architecture suitable for a broad range of research and clinical applications.

In general, this thesis lays the groundwork for a new era in treating treatment-resistant neurological disorders, offering more targeted, effective, and personalized treatment options. The integration of advanced simulation environments, practical control tasks, AI-driven control policies, and an end-to-end translational research and development platform illustrates the potential of iCLON systems to revolutionize the patient care and open new frontiers in neuromodulation technologies.

6.2 Future work

The development of automated intelligent closed-loop neuromodulation systems is an exciting and rapidly evolving field, with numerous future directions for research and development.

Interpretability is becoming an increasingly important aspect of AI and machine learning systems, especially in the medical domain, where decisions made by these systems can have a significant impact on patient health outcomes. In the context of closed-loop neuromodulation systems, interpretability is important in ensuring that clinicians and patients can understand how the system is making decisions and have confidence in the system's ability to optimize treatment outcomes. Moreover, developing intelligent closed-loop neuromodulation systems that are more interpretable will also improve the regulatory approval process, as it will be easier to assess the system's safety and efficacy. Integration of interpretable machine learning and reinforcement learning models into the design of iCLON system is a promising future direction.

Neuromodulation therapies modulates the desired neural activity by delivering electrical or magnetic stimuli to the targeted regions. Due to the intricate and dynamic nature of the nervous system, the underlying causes of the disease may induce multiple symptoms that we are interested to treat simultaneously. In addition, applying the intervention to targeted regions might induce adverse side effects. Defining the algorithmic design of iCLON systems as a multi-task learning (MTL) problem enables the system to simultaneously perform multiple related tasks. Multi-task learning is a machine learning technique that involves training a single model to perform multiple related tasks simultaneously which helps with improved accuracy, better generalization, and increased efficiency. By sharing information between tasks, MTL can learn representations that are more robust and transferable across tasks, leading to improved accuracy and generalization. Additionally, by training a single model to perform multiple tasks, MTL can be more computationally and sample efficient than training separate models for each task, particularly when the tasks share similar features or require similar computations.

Although developing simulation environments help with testing and prototyping iCLON systems and algorithm selection in silico before testing in clinical experimental setups, there is still a translational gap in simulating potential challenges in real experimental setups. Future developments in this area help with minimizing the translational gaps by simulating real-world scenarios including simulating the noise of observation and action spaces, simulating the effect of artifacts, variations across different subjects, and disease subtypes. Incorporating data-driven dynamical modeling approaches may also better reproduce the real-world experimental setups and helps with reducing the translational gaps.

Bibliography

[1] Elliot S Krames, P Hunter Peckham, Ali Rezai, and Farag Aboelsaad. What is neuromodulation? In *Neuromodulation*, pages 3–8. Elsevier, 2009.

[2] Yuemei Hou, Qina Zhou, and Sunny S Po. Neuromodulation for cardiac arrhythmia. *Heart Rhythm*, 13(2):584–592, 2016.

[3] Philippe Ryvlin, Sylvain Rheims, Lawrence J Hirsch, Arseny Sokolov, and Lara Jehi. Neuromodulation in epilepsy: state-of-the-art approved therapies. *The Lancet Neurology*, 20(12):1038–1047, 2021.

[4] Patric Blomstedt and Marwan I Hariz. Deep brain stimulation for movement disorders before dbs for movement disorders. *Parkinsonism & Related Disorders*, 16(7):429–433, 2010.

[5] Helena Knotkova, Clement Hamani, Eellan Sivanesan, María Francisca Elgueta Le Beuffe, Jee Youn Moon, Steven P Cohen, and Marc A Huntoon. Neuromodulation for chronic pain. *The Lancet*, 397(10289):2111–2124, 2021.

[6] Yasin Temel, Sarah A Hescham, Ali Jahanshahi, Marcus LF Janssen, Sonny KH Tan, Jacobus J van Overbeeke, Linda Ackermans, Mayke Oosterloo, Annelien Duits, Albert FG Leentjens, et al. Neuromodulation in psychiatric disorders. *International review of neurobiology*, 107:283–314, 2012.

[7] Karen Hunka, Oksana Suchowersky, Susan Wood, Lorelei Derwent, and

Zelma HT Kiss. Nursing time to program and assess deep brain stimulators in movement disorder patients. *Journal of Neuroscience Nursing*, 37(4):204–210, 2005.

[8] Meng-Chen Lo and Alik S Widge. Closed-loop neuromodulation systems: next-generation treatments for psychiatric illness. *International review of psychiatry*, 29(2):191–204, 2017.

[9] Iñaki Iturrate, Michael Pereira, and José del R Millán. Closed-loop electrical neurostimulation: challenges and opportunities. *Current Opinion in Biomedical Engineering*, 8:28–37, 2018.

[10] Matthew D Johnson, Hubert H Lim, Theoden I Netoff, Allison T Connolly, Nessa Johnson, Abhrajeet Roy, Abbey Holt, Kelvin O Lim, James R Carey, Jerrold L Vitek, et al. Neuromodulation for brain disorders: challenges and opportunities. *IEEE Transactions on Biomedical Engineering*, 60(3):610–624, 2013.

[11] Pedram Afshar, Ankit Khambhati, Scott Stanslaski, David Carlson, Randy Jensen, Dave Linde, Siddharth Dani, Maciej Lazarewicz, Peng Cong, Jon Giftakis, et al. A translational platform for prototyping closed-loop neuromodulation systems. *Frontiers in neural circuits*, 6:117, 2013.

[12] Derrick Soh, Timo R Ten Brinke, Andres M Lozano, and Alfonso Fasano. Therapeutic window of deep brain stimulation using cathodic monopolar, bipolar, semi-bipolar, and anodic stimulation. *Neuromodulation: Technology at the Neural Interface*, 22(4):451–455, 2019.

[13] Breanna Sheldon, Michael D Staudt, Lucian Williams, Tessa A Harland, and Julie G Pilitsis. Spinal cord stimulation programming: a crash course. *Neurosurgical review*, 44:709–720, 2021.

[14] Andrew Haddock, Kyle T Mitchell, Andrew Miller, Jill L Ostrem, Howard J Chizeck, and Svjetlana Miocinovic. Automated deep brain stimulation programming for tremor. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(8):1618–1625, 2018.

[15] Cameron C McIntyre, Ashutosh Chaturvedi, Reuben R Shamir, and Scott F Lempka. Engineering the next generation of clinical deep brain stimulation technology. *Brain stimulation*, 8(1):21–26, 2015.

[16] Nicole C Swann, Coralie De Hemptinne, Margaret C Thompson, Svjetlana Miocinovic, Andrew M Miller, Jill L Ostrem, Howard J Chizeck, Philip A Starr, et al. Adaptive deep brain stimulation for parkinson's disease using motor cortex sensing. *Journal of neural engineering*, 15(4):046006, 2018.

[17] Y. Yao and M. V. Kothare. Model predictive control of selective vagal nerve stimulation for regulating cardiovascular system. *American Control Conference (ACC) (IEEE)*, pages 563–568, 2020.

[18] Hector M Romero Ugalde, David Ojeda, Virginie Le Rolle, David Andreu, David Guiraud, Jean-L Bonnet, Christine Henry, Nicole Karam, Albert Hagege, Philippe Mabo, et al. Model-based design and experimental validation of control modules for neuromodulation devices. *IEEE Transactions on Biomedical Engineering*, 63(7):1551–1558, 2015.

[19] Youhua Zhang, Kent A Mowrey, Shaowei Zhuang, Don W Wallick, Zoran B Popovic, and Todor N Mazgalev. Optimal ventricular rate slowing during atrial fibrillation by feedback av nodal-selective vagal stimulation. *American Journal of Physiology-Heart and Circulatory Physiology*, 282(3):H1102–H1110, 2002.

[20] Elliot Greenwald, Ernest So, Qihong Wang, Mohsen Mollazadeh, Christoph Maier, Ralph Etienne-Cummings, Gert Cauwenberghs, and Nitish Thakor. A

bidirectional neural interface ic with chopper stabilized bioadc array and charge balanced stimulator. *IEEE transactions on biomedical circuits and systems*, 10 (5):990–1002, 2016.

[21] Benoit Duchet, Gihan Weerasinghe, Hayriye Cagnan, Peter Brown, Christian Bick, and Rafal Bogacz. Phase-dependence of response curves to deep brain stimulation and their relationship: from essential tremor patient data to a wilson–cowan model. *The Journal of Mathematical Neuroscience*, 10:1–39, 2020.

[22] M Maheedhar and T Deepa. A behavioral study of different controllers and algorithms in real-time applications. *IETE Journal of Research*, pages 1–25, 2022.

[23] Stephen Bassi Joseph, Emmanuel Gbenga Dada, Afeez Abidemi, David Opeoluwa Oyewola, and Ban Mohammed Khammas. Metaheuristic algorithms for pid controller parameters tuning: Review, approaches and open problems. *Heliyon*, 2022.

[24] KS Holkar and Laxman M Waghmare. An overview of model predictive control. *International Journal of control and automation*, 3(4):47–63, 2010.

[25] Simon Little, Alex Pogosyan, Spencer Neal, Baltazar Zavala, Ludvic Zrinzo, Marwan Hariz, Thomas Foltynie, Patricia Limousin, Keyoumars Ashkan, James FitzGerald, et al. Adaptive deep brain stimulation in advanced parkinson disease. *Annals of neurology*, 74(3):449–457, 2013.

[26] Mahboubeh Parastarfeizabadi and Abbas Z Kouzani. Advances in closed-loop deep brain stimulation devices. *Journal of neuroengineering and rehabilitation*, 14(1):1–20, 2017.

[27] Boris Rosin, Maya Slovik, Rea Mitelman, Michal Rivlin-Etzion, Suzanne N Haber, Zvi Israel, Eilon Vaadia, and Hagai Bergman. Closed-loop deep brain

stimulation is superior in ameliorating parkinsonism. *Neuron*, 72(2):370–384, 2011.

[28] Clement Hamani, Erich Richter, Jason M Schwalb, and Andres M Lozano. Bilateral subthalamic nucleus stimulation for parkinson's disease: a systematic review of the clinical literature. *Neurosurgery*, 62:SHC–863, 2008.

[29] Hemmings Wu, Hartwin Ghekiere, Dorien Beeckmans, Tim Tambuyzer, Kris van Kuyck, Jean-Marie Aerts, and Bart Nuttin. Conceptualization and validation of an open-source closed-loop deep brain stimulation system in rat. *Scientific Reports*, 5(1):9921, 2015.

[30] FDA. Medtronic activa® tremor control system p960009. `https://www.accessdata.fda.gov/cdrh_docs/pdf/p960009.pdf`, 1997.

[31] FDA. Medtronic activa® parkinson's control system p960009/s7. `https://www.accessdata.fda.gov/cdrh_docs/pdf/p960009s007b.pdf`, 2002.

[32] FDA. Medtronic activa® dystonia therapy - h020007. `https://www.accessdata.fda.gov/cdrh_docs/pdf2/H020007A.pdf`, 2003.

[33] FDA. Fda approves humanitarian device exemption for deep brain stimulator for severe obsessive-compulsive disorder. `https://wayback.archive-it.org/7993/20170113195459/http://www.fda.gov/NewsEvents/Newsroom/PressAnnouncements/2009/ucm149529.htm`, 2009.

[34] FDA. Rns® system - p100026. `https://www.accessdata.fda.gov/cdrh_docs/pdf10/P100026A.pdf`, 2013.

[35] Pradeeban Kathiravelu, Parisa Sarikhani, Ping Gu, and Babak Mahmoudi. Software-defined workflows for distributed interoperable closed-loop neuromodulation control systems. *IEEE Access*, 9:131733–131745, 2021.

[36] Pradeeban Kathiravelu, Mark Arnold, Jake Fleischer, Yuyu Yao, Shubham Awasthi, Aviral Kumar Goel, Andrew Branen, Parisa Sarikhani, Gautam Kumar, Mayuresh V Kothare, et al. Control-core: a framework for simulation and design of closed-loop peripheral neuromodulation control systems. *IEEE Access*, 10:36268–36285, 2022.

[37] Gene F Franklin, J David Powell, Abbas Emami-Naeini, and J David Powell. *Feedback control of dynamic systems*, volume 4. Prentice hall Upper Saddle River, 2002.

[38] James C Houk. Control strategies in physiological systems. *The FASEB journal*, 2(2):97–107, 1988.

[39] Stavros Zanos. Closed-loop neuromodulation in physiological and translational research. *Cold Spring Harbor perspectives in medicine*, 9(11), 2019.

[40] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[41] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[42] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[44] Timothy J Ross. *Fuzzy logic with engineering applications*. John Wiley & Sons, 2009.

[45] Konstantinos P Michmizos, Polytimi Frangou, Pantelis Stathis, Damianos Sakas, and Konstantina S Nikita. Beta-band frequency peaks inside the subtha-

lamic nucleus as a biomarker for motor improvement after deep brain stimulation in parkinson's disease. *IEEE Journal of Biomedical and Health Informatics*, 19(1):174–180, 2014.

[46] Emma J Quinn, Zack Blumenfeld, Anca Velisar, Mandy Miller Koop, Lauren A Shreve, Megan H Trager, Bruce C Hill, Camilla Kilbane, Jaimie M Henderson, and Helen Brontë-Stewart. Beta oscillations in freely moving parkinson's subjects are attenuated during deep brain stimulation. *Movement Disorders*, 30 (13):1750–1758, 2015.

[47] Clare M Davidson, Annraoi M de Paor, Hayriye Cagnan, and Madeleine M Lowery. Analysis of oscillatory neural activity in series network models of parkinson's disease during deep brain stimulation. *IEEE Transactions on Biomedical Engineering*, 63(1):86–96, 2015.

[48] Simon Little, Alek Pogosyan, Spencer Neal, Ludvic Zrinzo, Marwan Hariz, Thomas Foltynie, Patricia Limousin, and Peter Brown. Controlling parkinson's disease with adaptive deep brain stimulation. *JoVE (Journal of Visualized Experiments)*, (89):e51403, 2014.

[49] Simon Little, Martijn Beudel, Ludvic Zrinzo, Thomas Foltynie, Patricia Limousin, Marwan Hariz, Spencer Neal, Binith Cheeran, Hayriye Cagnan, James Gratwicke, et al. Bilateral adaptive deep brain stimulation is effective in parkinson's disease. *Journal of Neurology, Neurosurgery & Psychiatry*, 87(7): 717–721, 2016.

[50] Anders Christian Meidahl, Gerd Tinkhauser, Damian Marc Herz, Hayriye Cagnan, Jean Debarros, and Peter Brown. Adaptive deep brain stimulation for movement disorders: the long road to clinical therapy. *Movement disorders*, 32(6):810–819, 2017.

[51] Hayriye Cagnan, David Pedrosa, Simon Little, Alek Pogosyan, Binith Cheeran, Tipu Aziz, Alexander Green, James Fitzgerald, Thomas Foltynie, Patricia Limousin, et al. Stimulating at the right time: phase-specific deep brain stimulation. *Brain*, 140(1):132–145, 2017.

[52] Kiam Heong Ang, Gregory Chong, and Yun Li. Pid control system analysis, design, and technology. *IEEE transactions on control systems technology*, 13 (4):559–576, 2005.

[53] K Pyragas, OV Popovych, and PA Tass. Controlling synchrony in oscillatory networks with a separate stimulation-registration setup. *Europhysics Letters*, 80(4):40002, 2007.

[54] John E Fleming, Eleanor Dunn, and Madeleine M Lowery. Simulation of closed-loop deep brain stimulation control schemes for suppression of pathological beta oscillations in parkinson's disease. *Frontiers in neuroscience*, 14:166, 2020.

[55] Judith Evers, Jakub Orłowski, Hanne Jahns, and Madeleine M Lowery. On-off and proportional closed-loop adaptive deep brain stimulation reduces motor symptoms in freely moving hemiparkinsonian rats. *Neuromodulation: Technology at the Neural Interface*, 2023.

[56] John E Fleming, Jakub Orłowski, Madeleine M Lowery, and Antoine Chaillet. Self-tuning deep brain stimulation controller for suppression of beta oscillations: analytical derivation and numerical validation. *Frontiers in neuroscience*, 14: 639, 2020.

[57] John E Fleming, Sageanne Senneff, and Madeleine M Lowery. Multivariable closed-loop control of deep brain stimulation for parkinson's disease. *Journal of Neural Engineering*, 20(5):056029, 2023.

[58] Eleanor M Dunn and Madeleine M Lowery. Simulation of pid control schemes for closed-loop deep brain stimulation. In *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 1182–1185. IEEE, 2013.

[59] P Gorzelic, SJ Schiff, and Alok Sinha. Model-based rational feedback controller design for closed-loop deep brain stimulation of parkinson's disease. *Journal of neural engineering*, 10(2):026016, 2013.

[60] Parisa Sarikhani, Hao-Lun Hsu, and Babak Mahmoudi. Automated tuning of closed-loop neuromodulation control systems using bayesian optimization. In *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 1734–1737. IEEE, 2022.

[61] Kestutis Pyragas. Continuous control of chaos by self-controlling feedback. *Physics letters A*, 170(6):421–428, 1992.

[62] Chen Liu, Changsong Zhou, Jiang Wang, Chris Fietkiewicz, and Kenneth A Loparo. Delayed feedback-based suppression of pathological oscillations in a neural mass model. *IEEE Transactions on Cybernetics*, 51(10):5046–5056, 2019.

[63] Michael G Rosenblum and Arkady S Pikovsky. Controlling synchronization in an ensemble of globally coupled oscillators. *Physical Review Letters*, 92(11): 114102, 2004.

[64] Michael Rosenblum and Arkady Pikovsky. Delayed feedback control of collective synchrony: An approach to suppression of pathological brain rhythms. *Physical review E*, 70(4):041904, 2004.

[65] Christian Hauptmann, O Popovych, and Peter A Tass. Effectively desynchronizing deep brain stimulation based on a coordinated delayed feedback stimulation via several sites: a computational study. *Biological cybernetics*, 93(6):463–470, 2005.

[66] Christian Hauptmann, O Popovych, and Peter A Tass. Delayed feedback control of synchronization in locally coupled neuronal networks. *Neurocomputing*, 65: 759–767, 2005.

[67] Oleksandr V Popovych, Christian Hauptmann, and Peter A Tass. Effective desynchronization by nonlinear delayed feedback. *Physical review letters*, 94 (16):164102, 2005.

[68] Oleksandr V Popovych, Christian Hauptmann, and Peter A Tass. Control of neuronal synchrony by nonlinear delayed feedback. *Biological cybernetics*, 95 (1):69–85, 2006.

[69] István Z Kiss, Craig G Rusin, Hiroshi Kori, and John L Hudson. Engineering complex dynamical structures: Sequential patterns and desynchronization. *Science*, 316(5833):1886–1889, 2007.

[70] Oleksandr V Popovych and Peter A Tass. Synchronization control of interacting oscillatory ensembles by mixed nonlinear delayed feedback. *Physical Review E*, 82(2):026204, 2010.

[71] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[72] Chuen-Chien Lee. Fuzzy logic in control systems: fuzzy logic controller. i. *IEEE Transactions on systems, man, and cybernetics*, 20(2):404–418, 1990.

[73] Ehsan Rouhani, Ehsan Jafari, and Amir Akhavan. Suppression of seizure in childhood absence epilepsy using robust control of deep brain stimulation: a simulation study. *Scientific Reports*, 13(1):461, 2023.

[74] Ehsan Rouhani and Yaser Fathi. Robust multi-input multi-output adaptive fuzzy terminal sliding mode control of deep brain stimulation in parkinson's disease: A simulation study. *Scientific Reports*, 11(1):21169, 2021.

[75] Mahboubeh Parastarfeizabadi, Roy V Sillitoe, and Abbas Z Kouzani. Multi-disease deep brain stimulation. *IEEE Access*, 8:216933–216947, 2020.

[76] Meysam Gheisarnejad, Behnam Faraji, Zahra Esfahani, and Mohammad-Hassan Khooban. A close loop multi-area brain stimulation control for parkinson's patients rehabilitation. *IEEE Sensors Journal*, 20(4):2205–2213, 2019.

[77] Andrew Haddock, Anca Velisar, Jeffrey Herron, Helen Bronte-Stewart, and Howard J Chizeck. Model predictive control of deep brain stimulation for parkinsonian tremor. In *2017 8th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 358–362. IEEE, 2017.

[78] Hao Fang and Yuxiao Yang. Predictive neuromodulation of cingulo-frontal neural dynamics in major depressive disorder using a brain-computer interface system: A simulation study. *Frontiers in Computational Neuroscience*, 17: 1119685, 2023.

[79] Georgios Is Detorakis, Antoine Chaillet, Stéphane Palfi, and Suhan Senova. Closed-loop stimulation of a delayed neural fields model of parkinsonian stn-gpe network: a theoretical and computational study. *Frontiers in neuroscience*, 9:237, 2015.

[80] Fei Su, Jiang Wang, Shuangxia Niu, Huiyan Li, Bin Deng, Chen Liu, and Xile Wei. Nonlinear predictive control for adaptive adjustments of deep brain stimulation parameters in basal ganglia–thalamic network. *Neural Networks*, 98:283–295, 2018.

[81] Andrew Branen, Yuyu Yao, Mayuresh V Kothare, Babak Mahmoudi, and Gautam Kumar. Data driven control of vagus nerve stimulation for the cardiovascular system: An in silico computational study. *Frontiers in Physiology*, 13: 798157, 2022.

[82] Yuyu Yao and Mayuresh V Kothare. Nonlinear closed-loop predictive control of heart rate and blood pressure using vagus nerve stimulation: An in silico study. *IEEE Transactions on Biomedical Engineering*, 2023.

[83] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[84] Romy Lorenz, Ricardo Pio Monti, Inês R Violante, Christoforos Anagnostopoulos, Aldo A Faisal, Giovanni Montana, and Robert Leech. The automatic neuroscientist: A framework for optimizing experimental design with closed-loop real-time fmri. *NeuroImage*, 129:320–334, 2016.

[85] Bethany J Stieve, Thomas J Richner, Chris Krook-Magnuson, Theoden I Netoff, and Esther Krook-Magnuson. Optimization of closed-loop electrical stimulation enables robust cerebellar-directed seizure control. *Brain*, 146(1):91–108, 2023.

[86] Romy Lorenz, Laura E Simmons, Ricardo P Monti, Joy L Arthur, Severin Limal, Ilkka Laakso, Robert Leech, and Ines R Violante. Efficiently searching through large tacs parameter spaces using closed-loop bayesian optimization. *Brain stimulation*, 12(6):1484–1489, 2019.

[87] Alexandre Boutet, Radhika Madhavan, Gavin JB Elias, Suresh E Joel, Robert Gramer, Manish Ranjan, Vijayashankar Paramanandam, David Xu, Jurgen Germann, Aaron Loh, et al. Predicting optimal deep brain stimulation parameters for parkinson's disease using functional mri and machine learning. *Nature communications*, 12(1):3043, 2021.

[88] Kenneth H Louie, Matthew N Petrucci, Logan L Grado, Chiahao Lu, Paul J Tuite, Andrew G Lamperski, Colum D MacKinnon, Scott E Cooper, and Theoden I Netoff. Semi-automated approaches to optimize deep brain stimulation

parameters in parkinson's disease. *Journal of NeuroEngineering and Rehabilitation*, 18(1):83, 2021.

[89] Zixi Zhao, Aliya Ahmadi, Caleb Hoover, Logan Grado, Nicholas Peterson, Xinran Wang, David Freeman, Thomas Murray, Andrew Lamperski, David Darrow, et al. Optimization of spinal cord stimulation using bayesian preference learning and its validation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:1987–1997, 2021.

[90] Logan L Grado, Matthew D Johnson, and Theoden I Netoff. Bayesian adaptive dual control of deep brain stimulation in a computational model of parkinson's disease. *PLoS computational biology*, 14(12):e1006606, 2018.

[91] Johanna J O'Day, Yasmine M Kehnemouyi, Matthew N Petrucci, Ross W Anderson, Jeffrey A Herron, and Helen M Bronte-Stewart. Demonstration of kinematic-based closed-loop deep brain stimulation for mitigating freezing of gait in people with parkinson's disease. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 3612–3616. IEEE, 2020.

[92] Brady Houston, Margaret Thompson, Andrew Ko, and Howard Chizeck. A machine-learning approach to volitional control of a closed-loop deep brain stimulation system. *Journal of neural engineering*, 16(1):016004, 2019.

[93] Pitamber Shukla, Ishita Basu, Daniel Graupe, Daniela Tuninetti, and Konstantin V Slavin. A neural network-based design of an on-off adaptive control for deep brain stimulation in movement disorders. In *2012 annual international conference of the IEEE engineering in medicine and biology society*, pages 4140–4143. IEEE, 2012.

[94] Yuyu Yao and Mayuresh V Kothare. Model predictive control of selective va-

gal nerve stimulation for regulating cardiovascular system. In *2020 American Control Conference (ACC)*, pages 563–568. IEEE, 2020.

[95] Sebastián Castaño-Candamil, Mara Vaihinger, and Michael Tangermann. A simulated environment for early development stages of reinforcement learning algorithms for closed-loop deep brain stimulation. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2900–2904. IEEE, 2019.

[96] Behnam Faraji, Korosh Rouhollahi, Akram Nezhadi, and Zahra Jamalpoor. A novel closed-loop deep brain stimulation technique for parkinson's patients rehabilitation utilizing machine learning. *IEEE Sensors Journal*, 23(3):2914–2921, 2022.

[97] Marina Picillo, Andres M Lozano, Nancy Kou, Renato Puppi Munhoz, and Alfonso Fasano. Programming deep brain stimulation for tremor and dystonia: the toronto western hospital algorithms. *Brain stimulation*, 9(3):438–452, 2016.

[98] Parisa Sarikhani, Svjetlana Miocinovic, and Babak Mahmoudi. Towards automated patient-specific optimization of deep brain stimulation for movement disorders. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6159–6162. IEEE, 2019.

[99] Gregor R Wenzel, Jan Roediger, Christof Brücke, Ana Luísa de A Marcelino, Eileen Gülke, Monika Pötter-Nerger, Heleen Scholtes, Kenny Wynants, León M Juárez Paz, and Andrea A Kühn. Clover-dbs: Algorithm-guided deep brain stimulation-programming based on external sensor feedback evaluated in a prospective, randomized, crossover, double-blind, two-center study. *Journal of Parkinson's Disease*, 11(4):1887–1899, 2021.

[100] Fuyuko Sasaki, Genko Oyama, Satoko Sekimoto, Maierdanjiang Nuermaimaiti,

Hirokazu Iwamuro, Yasushi Shimo, Atsushi Umemura, and Nobutaka Hattori. Closed-loop programming using external responses for deep brain stimulation in parkinson's disease. *Parkinsonism & Related Disorders*, 84:47–51, 2021.

[101] Benoit Duchet, Gihan Weerasinghe, Christian Bick, and Rafal Bogacz. Optimizing deep brain stimulation based on isostable amplitude in essential tremor patient models. *Journal of neural engineering*, 18(4):046023, 2021.

[102] Ishita Basu, Daniel Graupe, Daniela Tuninetti, Pitamber Shukla, Konstantin V Slavin, Leo Verhagen Metman, and Daniel M Corcos. Pathological tremor prediction using surface electromyogram and acceleration: potential use in 'on–off'demand driven deep brain stimulator design. *Journal of neural engineering*, 10(3):036019, 2013.

[103] Daniel Graupe, Ishita Basu, Daniela Tuninetti, Prasad Vannemreddy, and Konstantin V Slavin. Adaptively controlling deep brain stimulation in essential tremor patient via surface electromyography. *Neurological research*, 32(9):899–904, 2010.

[104] P Shukla, I Basu, D Graupe, D Tuninetti, KV Slavin, L Verhagen Metman, and DM Corcos. A decision tree classifier for postural and movement conditions in essential tremor patients. In *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 117–120. IEEE, 2013.

[105] Nivedita Khobragade, Daniel Graupe, and Daniela Tuninetti. Towards fully automated closed-loop deep brain stimulation in parkinson's disease patients: A lamstar-based tremor predictor. In *2015 37th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, pages 2616–2619. IEEE, 2015.

[106] Takamitsu Yamamoto, Yoichi Katayama, Junichi Ushiba, Hiroko Yoshino,

Toshiki Obuchi, Kazutaka Kobayashi, Hideki Oshima, and Chikashi Fukaya. On-demand control system for deep brain stimulation for treatment of intention tremor. *Neuromodulation: Technology at the Neural Interface*, 16(3):230–235, 2013.

[107] Jeffrey A Herron, Margaret C Thompson, Timothy Brown, Howard J Chizeck, Jeffrey G Ojemann, and Andrew L Ko. Chronic electrocorticography for sensing movement intention and closed-loop deep brain stimulation with wearable sensors in an essential tremor patient. *Journal of neurosurgery*, 127(3):580–587, 2016.

[108] Parisa Sarikhani, Benjamin Ferleger, Kyle Mitchell, Jill Ostrem, Jeffrey Herron, Babak Mahmoudi, and Svjetlana Miocinovic. Automated deep brain stimulation programming with safety constraints for tremor suppression in patients with parkinson's disease and essential tremor. *Journal of neural engineering*, 19(4):046042, 2022.

[109] Parisa Sarikhani, Benjamin Ferleger, Jeffrey Herron, Babak Mahmoudi, and Svjetlana Miocinovic. Automated deep brain stimulation programing with safety constraints for tremor suppression. *Brain Stimulation: Basic, Translational, and Clinical Research in Neuromodulation*, 14(6):1699–1700, 2021.

[110] Stanley Fahn, Eduardo Tolosa, Concepcíon Marín, et al. Clinical rating scale for tremor. *Parkinson's disease and movement disorders*, 2:271–280, 1993.

[111] Jeffrey Herron and Howard Jay Chizeck. Prototype closed-loop deep brain stimulation systems inspired by norbert wiener. In *2014 IEEE Conference on Norbert Wiener in the 21st Century (21CW)*, pages 1–6. IEEE, 2014.

[112] Jeffrey Herron, Tim Denison, and Howard Jay Chizeck. Closed-loop dbs with

movement intention. In *2015 7th international IEEE/EMBS conference on neural engineering (NER)*, pages 844–847. IEEE, 2015.

[113] Jeffrey Andrew Herron. *Closed-loop deep brain stimulation: bidirectional neuroprosthetics for tremor and BCI*. PhD thesis, 2016.

[114] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[115] Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. Gpflow: A gaussian process library using tensorflow. *Journal of Machine Learning Research*, 18(40):1–6, 2017. URL `http://jmlr.org/papers/v18/16-537.html`.

[116] Nicolas Knudde, Joachim van der Herten, Tom Dhaene, and Ivo Couckuyt. Gpflowopt: A bayesian optimization library using tensorflow. *arXiv preprint arXiv:1711.03845*, 2017.

[117] Rikky RPR Duivenvoorden, Felix Berkenkamp, Nicolas Carion, Andreas Krause, and Angela P Schoellig. Constrained bayesian optimization with particle swarms for safe adaptive controller tuning. *IFAC-PapersOnLine*, 50(1): 11800–11807, 2017.

[118] Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. Safe exploration for optimization with gaussian processes. In *International conference on machine learning*, pages 997–1005. PMLR, 2015.

[119] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning*, pages 3627–3635. PMLR, 2017.

[120] Jens Volkmann, Elena Moro, and Rajesh Pahwa. Basic algorithms for the programming of deep brain stimulation in parkinson's disease. *Movement disorders: official journal of the Movement Disorder Society*, 21(S14):S284–S289, 2006.

[121] Christopher L Pulliam, Dustin A Heldman, Tseganesh H Orcutt, Thomas O Mera, Joseph P Giuffrida, and Jerrold L Vitek. Motion sensor strategies for automated optimization of deep brain stimulation in parkinson's disease. *Parkinsonism & related disorders*, 21(4):378–382, 2015.

[122] Dustin A Heldman, Christopher L Pulliam, Enrique Urrea Mendoza, Maureen Gartner, Joseph P Giuffrida, Erwin B Montgomery Jr, Alberto J Espay, and Fredy J Revilla. Computer-guided deep brain stimulation programming for parkinson's disease. *Neuromodulation: Technology at the Neural Interface*, 19 (2):127–132, 2016.

[123] Angela M Noecker, Anneke M Frankemolle-Gilbert, Bryan Howell, Mikkel V Petersen, Sinem Balta Beylergil, Aasef G Shaikh, and Cameron C McIntyre. Stimvision v2: Examples and applications in subthalamic deep brain stimulation for parkinson's disease. *Neuromodulation: Technology at the Neural Interface*, 24(2):248–258, 2021.

[124] Hyoseon Jeon, Woongwoo Lee, Hyeyoung Park, Hong Ji Lee, Sang Kyong Kim, Han Byul Kim, Beomseok Jeon, and Kwang Suk Park. Automatic classification of tremor severity in parkinson's disease using a wearable device. *Sensors*, 17 (9):2067, 2017.

[125] P Rebelo, AL Green, TZ Aziz, A Kent, D Schafer, L Venkatesan, and B Cheeran. Thalamic directional deep brain stimulation for tremor: spend less, get more. *Brain stimulation*, 11(3):600–606, 2018.

[126] Philipp Mahlknecht, Harith Akram, Dejan Georgiev, Elina Tripoliti, Joseph

Candelario, Andre Zacharia, Ludvic Zrinzo, Jonathan Hyam, Marwan Hariz, Thomas Foltynie, et al. Pyramidal tract activation due to subthalamic deep brain stimulation in parkinson's disease. *Movement Disorders*, 32(8):1174–1182, 2017.

[127] Una Buckley, Kalyanam Shivkumar, and Jeffrey L Ardell. Autonomic regulation therapy in heart failure. *Current heart failure reports*, 12:284–293, 2015.

[128] Dariush Mozaffarian. Heart disease and stroke statistics—2016 update. *Circulation*, 133:1, 2016.

[129] Matteo Maria Ottaviani, Fabio Vallone, Silvestro Micera, and Fabio A Recchia. Closed-loop vagus nerve stimulation for the treatment of cardiovascular diseases: state of the art and future directions. *Frontiers in Cardiovascular Medicine*, 9: 866957, 2022.

[130] Michael J Capilupi, Samantha M Kerath, and Lance B Becker. Vagus nerve stimulation and the cardiovascular system. *Cold Spring Harbor perspectives in medicine*, 10(2), 2020.

[131] Rajendra K Premchand, Kamal Sharma, Sanjay Mittal, Rufino Monteiro, Satyajit Dixit, Imad Libbus, Lorenzo A DiCarlo, Jeffrey L Ardell, Thomas S Rector, Badri Amurthur, et al. Autonomic regulation therapy via left or right cervical vagus nerve stimulation in patients with chronic heart failure: results of the anthem-hf trial. *Journal of cardiac failure*, 20(11):808–816, 2014.

[132] Faiez Zannad, Gaetano M De Ferrari, Anton E Tuinenburg, David Wright, Josep Brugada, Christian Butter, Helmut Klein, Craig Stolen, Scott Meyer, Kenneth M Stein, et al. Chronic vagal stimulation for the treatment of low ejection fraction heart failure: results of the neural cardiac therapy for heart

failure (nectar-hf) randomized controlled trial. *European heart journal*, 36(7): 425–433, 2015.

[133] Michael R Gold, Dirk J Van Veldhuisen, Paul J Hauptman, Martin Borggrefe, Spencer H Kubo, Randy A Lieberman, Goran Milasinovic, Brett J Berman, Sanja Djordjevic, Suresh Neelagaru, et al. Vagus nerve stimulation for the treatment of heart failure: the inovate-hf trial. *Journal of the American College of Cardiology*, 68(2):149–158, 2016.

[134] Zain UA Asad and Stavros Stavrakis. Vagus nerve stimulation for the treatment of heart failure. *Bioelectronics in Medicine*, 2(1):43–54, 2019.

[135] James Crowe, GR Chen, R Ferdous, DR Greenwood, MJ Grimble, HP Huang, JC Jeng, Michael A Johnson, MR Katebi, S Kwong, et al. *PID control: new identification and design methods*. Springer, 2005.

[136] Marco Tosato, Ken Yoshida, Egon Toft, Vitas Nekrasas, and Johannes J Struijk. Closed-loop control of the heart rate by electrical stimulation of the vagus nerve. *Medical and Biological Engineering and Computing*, 44:161–169, 2006.

[137] Hector M Romero-Ugalde, Virginie Le Rolle, Jean-Luc Bonnet, Christine Henry, Alain Bel, Philippe Mabo, Guy Carrault, and Alfredo I Hernández. A novel controller based on state-transition models for closed-loop vagus nerve stimulation: Application to heart rate regulation. *PloS one*, 12(10):e0186068, 2017.

[138] Eduardo F Camacho, Carlos Bordons, Eduardo F Camacho, and Carlos Bordons. *Model predictive controllers*. Springer, 2007.

[139] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[140] Nicolas Heess, Dhruva Tb, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[141] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[142] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[143] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[144] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[145] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[146] I Goodfellow, Y Bengio, and A Courville. *Deep learning.* 2016. ISBN 9780262035613. doi: 10.4258/hir.2016.22.4.351. URL `https://books.google.com/books?hl=en&lr=&id=omivDQAAQBAJ&oi=fnd&pg=PR5&dq=Goodfellow+I,+Bengio+Y,+Courville+A.+Deep+learning.+MIT+press%3B+2016+Nov+10.&ots=MNU5imtzRX&sig=OyZE7RO07rhYN6M-8B6pKsE7hcc`.

[147] Carl Edward Rasmussen. Gaussian processes in machine learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3176:63–71, 2004. ISSN 16113349. doi: 10.1007/978-3-540-28650-9_4/COVER. URL `https://link.springer.com/chapter/10.1007/978-3-540-28650-9_4`.

[148] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

[149] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. *Proceedings of the International Conference on Machine Learning*, pages 1889–1897.

[150] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines, 2018. URL `https://github.com/hill-a/stable-baselines`.

[151] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *35th International Conference on Machine Learning, ICML 2018*, 5: 2976–2989, 1 2018. URL `https://arxiv.org/abs/1801.01290v2`.

[152] Parisa Sarikhani, Hao-Lun Hsu, Ozgur Kara, Joon Kyung Kim, Hadi Esmaeilzadeh, and Babak Mahmoudi. Neuroweaver: a platform for designing intelligent closed-loop neuromodulation systems. *Brain Stimulation: Basic, Translational, and Clinical Research in Neuromodulation*, 14(6):1661, 2021.

[153] Aaftab Munshi, Benedict Gaster, Timothy G Mattson, and Dan Ginsburg. *OpenCL programming guide*. Pearson Education, 2011.

[154] CUDA Nvidia. Compute unified device architecture programming guide. 2007.

[155] Shoumik Palkar, James J Thomas, Anil Shanbhag, Deepak Narayanan, Holger Pirk, Malte Schwarzkopf, Saman Amarasinghe, Matei Zaharia, and Stanford InfoLab. Weld: A common runtime for high performance data analytics. In *Conference on Innovative Data Systems Research (CIDR)*, volume 19, 2017.

[156] Pypl popularity of programming language index. URL `https://pypl.github.io/PYPL.html`.

[157] E. Del Sozzo, R. Baghdadi, S. Amarasinghe, and M. D. Santambrogio. A unified backend for targeting fpgas from dsls. In *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 1–8, July 2018. doi: 10.1109/ASAP.2018.8445108.

[158] David Koeplinger, Matthew Feldman, Raghu Prabhakar, Yaqi Zhang, Stefan Hadjis, Ruben Fiszel, Tian Zhao, Luigi Nardi, Ardavan Pedram, Christos Kozyrakis, and Kunle Olukotun. Spatial: A language and compiler for application accelerators. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2018, pages 296–311, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5698-5. doi: 10.1145/3192366.3192379. URL `http://doi.acm.org/10.1145/3192366.3192379`.

[159] Marco Frigerio, Jonas Buchli, and Darwin G. Caldwell. A domain specific language for kinematic models and fast implementations of robot dynamics algorithms. In *International Workshop on Domain-Specific Languages and Models for Robotic Systems*, 2015.

[160] Mirko Bordignon, Kasper Stoy, and Ulrik Pagh Schultz. Generalized programming of modular robots through kinematic configurations. In *International Conference on Intelligent Robots and Systems*, 2011.

[161] Jonathan Ragan-Kelley, Andrew Adams, Dillon Sharlet, Connelly Barnes, Sylvain Paris, Marc Levoy, Saman Amarasinghe, and Frédo Durand. Halide: Decoupling algorithms from schedules for high-performance image processing. *Commun. ACM*, 61(1):106–115, December 2017. ISSN 0001-0782. doi: 10.1145/3150211. URL `http://doi.acm.org/10.1145/3150211`.

[162] Arvind K. Sujeeth, HyoukJoong Lee, Kevin J. Brown, Hassan Chafi, Michael Wu, Anand R. Atreya, Kunle Olukotun, Tiark Rompf, and Martin Odersky. Optiml: An implicitly parallel domain-specific language for machine learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 609–616, USA, 2011. Omnipress. ISBN 978-1-4503-0619-5. URL `http://dl.acm.org/citation.cfm?id=3104482.3104559`.

[163] A. Paszke et al. PyTorch: An imperative style, high-performance deep learning library. *NeurIPS*, 2019.

[164] M. Abadi et al. TensorFlow: A system for large-scale machine learning. *OSDI*, 2016.

[165] Joon Kyung Kim, Byung Hoon Ahn, Sean Kinzer, Soroush Ghodrati, Rohan Mahapatra, Brahmendra Yatham, Shu-Ting Wang, Dohee Kim, Parisa Sarikhani, Babak Mahmoudi, et al. Yin-yang: Programming abstractions for cross-domain multi-acceleration. *IEEE micro*, 42(5):89–98, 2022.

[166] Constance Hammond, Hagai Bergman, and Peter Brown. Pathological synchro-

nization in parkinson's disease: networks, models and treatments. *Trends in neurosciences*, 30(7):357–364, 2007.

[167] Dmitrii Krylov, Remi Tachet, Romain Laroche, Michael Rosenblum, and Dmitry V Dylov. Reinforcement learning framework for deep brain stimulation study. *arXiv preprint arXiv:2002.10948*, 2020.

[168] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[169] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[170] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on machine learning (ICML-18)*, pages 1861–1870. Citeseer, 2018.

[171] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Yuval Tassa Tom Erez, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[172] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

[173] Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *arXiv preprint arXiv:1708.02596*, 2017.

[174] Srivatsan Krishnan, Sharad Chitlangia, Maximilian Lam, Zishen Wan, Aleksandra Faust, and Vijay Janapa Reddi. Quantized reinforcement learning (quarl). *arXiv preprint arXiv:1910.01055*, 2019.