May 2023

Honour School of Engineering Science
Part C Project

Applying Bayesian Optimisation as a Controller for Peripheral Stimulation

James Mason

St Hugh's College

University of Oxford

Supervised by: Professor Tim Denison

Acknowledgements

I would like to thank my supervisor, Prof. Tim Denison for his support throughout the project. I would also like to thank Prof. Hayriye Cagnan and Dr. John Fleming for their regular valuable advice and assistance. Finally, thanks to Beatriz Silveira de Arruda for sharing her work on her device with me.

1 Abstract

Background: Selecting input parameters to give optimal outputs for time-varying functions is a challenging controller design problem without total knowledge of the underlying functions (time-varying bandit optimisation problem). This problem arises when designing controllers in electrical stimulation devices for the treatment of neurological diseases. Tremor is a disabling symptom for patients with many neurological conditions. The MRC Brain Network Dynamics Unit has developed an electrical peripheral neuromodulation device to treat neurological tremor using electrical nerve stimulation. Such devices require optimisation of the underlying (objective) function. In typical open-loop control devices this can be selected manually using inefficient methods such as grid searches through the parameter space or iterative tuning of parameters. However, the objective function and the optimal stimulation parameter set are likely to change as the result of time-varying external factors such as time of day, emotional state, activity levels, or use of medications. Changing stimulation parameters in response to the changes to objective function could give better results for the therapy. I have built and tested a controller for this device that runs on a limited memory microcontroller and selects optimal input parameters for a time-varying system.

Methods: The controller adapts to variations of the unknown function between input parameters and measured symptom change (change in tremor severity). The controller uses a modified version of the Time Varying Gaussian Process Upper Confidence Bound (TV-GP-UCB) method. This is a form of Bayesian optimisation which allows for the controller to rapidly update a model for the unknown function and select stimulation parameters that are predicted to reduce tremor. The main modification that was made to the TV-GP-UCB method was to include dynamic exploration/exploitation as (1) exploration, when the model is correct, does not maximise the potential of the model by exploring the parameter space when not needed, and (2) exploitation, when the model is incorrect, does not provide optimal stimulation. To achieve this, the controller makes the exploration bias a function of the information gained by the previous sample and the entropy of the previous iteration's model.

Other key design choices, such as the number of stimulations that feed the model, were chosen to ensure the controller was computationally efficient enough to be run on a microcontroller at an appropriate speed. The controller was tested against both step changes (e.g., sudden discrete changes to patient/device environment) and gradual changes of different timescales (e.g., disease progression, circadian rhythms, medications) in the objective function.

Results: Use of the modified TV-GP-UCB method improved results compared to fixed open-loop strategies. It showed better mean results for the re-acquisition of optimal parameters after a step change. It also showed better mean results for continuous changes over those timescales that correspond to circadian rhythms (24 hours) and medication effects (approximately three hours). The controller was outperformed by fixed open-loop strategies over much longer timescales which might be associated with disease progress. The controller was able to acquire new parameters by an average of 3.5 iterations after the step change to the objective function, faster than possible with a hard-resetting Bayesian optimisation

method (an alternative solution to the time varying bandit problem). On-device speed tests of the design, realised in C++ on an Teensy 4.0 microcontroller, gave an iteration decision run time of 2.28mS, showing an efficient design capable of running in between stimulations by the device.

Conclusions: A controller, using a new Bayesian optimisation method based on TV-GP-UCB with dynamic exploration/exploitation bias, tracks the optimal inputs for a time-varying unknown function subject to additive, zero-mean, Gaussian noise. Run time per iteration is extremely fast (2.28ms), allowing for on-device control. In simulated tests, it outperformed manually tuned fixed parameter methods for step changes and continuous drift changes to the objective function that may be relevant to real-world variations resulting from circadian rhythms and the use of medications. The changes to the objective function over longer timeframes such as encountered in disease progression are so gradual that the advantage of having a responsive controller is outweighed by the effect of having better, manually selected, parameters.

Contents

2 Introduction

Electrical stimulation therapies are used to treat many different neurological diseases[1]. These therapies have many stimulation parameters such as stimulation frequency and pulse duration. Between the stimulation parameters and the effect on the symptoms/disease exists a mystery objective function. The stimulation parameter set that optimises this function can be selected manually with slow methods such as grid searches through the parameter space or iterative tuning of parameters. This is often what is implemented for many electrical stimulation devices that use open-loop control.

For many neurological diseases such as tremor, the nature of symptoms change with time-varying external factors such as emotional state, activity level, time of day, and device [2, 3, 4]. In addition to this, it has been shown that the effects of electrical stimulation are state-dependent [5, 6]. Because of this the objective function and the optimal stimulation parameter set are likely to change as the result of the time-varying external factors. Changing stimulation parameters in response to the changes to objective function such as those caused by circadian rhythms could give better results for the therapy [7]. The aim of this project is to develop a controller that chooses optimal stimulation parameters in response to changes in the objective function.

For this, I designed a controller using a modified form of Bayesian optimisation. My method is a modification of the Time-Varying, Gaussian Process, Upper Confidence Bound TV-GP-UCB method[8]. I made the UCB variable (β) a function of the information gain and model entropy to give the controller a dynamic exploration/exploitation bias. This allows the controller to adapt to changing objective functions and exploit the model when the objective function is stationary. This is important to the medical device application. The exploitation of an incorrect model gives non-optimal results. Exploration, when the model was correct, is unnecessary and will not provide the best possible effect to the patient. Ideally, the controller will only opt to explore the parameter space when the model is incorrect and provides a sub-optimal effect.

This controller was designed in conjunction with the development of an electrical peripheral neuromodulation device by the MRC Brain Network Dynamics Unit. The device delivers non-invasive stimulation to the median nerve in the wrist [9]. It is hoped that by delivering stimulation phase-locked to the tremor oscillations, tremor severity can be reduced better than existing peripheral stimulation treatments.

A key design point, as a result of the intended use of this device, is that the controller can be used in real-time on the device. The embedded device has computational limits, such as limited memory, which introduce design constraints to the controller. This drove design choices throughout development. The controller is implemented on-device using C++, focusing on performance, reliability and computational efficiency.

The performance of the controller was tested in a virtual simulated test against perfect, manual, initial parameter selection. I compared the ability to respond to continuous changes in the objective function across different time scales of objective function drift. I also tested the ability to reacquire the optimal

stimulation parameters in the event of a sudden change in the objective function. The controller was also run in real life on the device microcontroller (Teensy 4.0). This validated the speed and computational efficiency of the design.

This report will first provide context to the problem, summarising tremor and its current treatments. Then a brief overview of the design problem for the controller, and a brief summary of existing methods for the stimulation parameter selection. The design process will then be explained followed by the testing methods and results.

3 Literature Review

The controller is intended to give optimal stimulation parameters for a time-varying function between stimulation parameters and effects on a symptom. Specifically this controller has been designed with an electrical peripheral stimulation device for tremor. The following section aims to give a background to tremor, the research behind the project and relevant control/optimisation methods.

3.1 Tremor

Tremor is defined as "Rhythmical, involuntary oscillatory movement of a body part" [10]. Tremor can be classified into many different types: Rest, postural, and kinetic tremors describe the conditions under which the tremor is apparent. Depending on these, and other conditions such as neurological damage the tremor will be classified as a type of tremor. Common types of pathological tremor include: Essential tremor (ET), Parkinson's Disease-Tremor (PDT), Drug-Induced Tremor (DIT).

Tremor is an oscillatory movement. It, therefore, makes sense to look at how tremor can be represented as an oscillatory system. The limb with muscles, tendons and ligaments can be treated as a second-order system with a spring, damper and mass. This in itself will have a resonant frequency.

The central nervous system has 'central oscillators' which are neuronal networks that fire in rhythmic patterns without sensory inputs [11]. These central oscillators drive the limb in an oscillatory fashion which can now be thought of as a driven second-order system. When the system is driven at or close to its resonant frequency it will result in large amplitude oscillations.

PDT and ET are thought to be caused by central oscillators [12, 13]. The neuronal networks involved include the motor cortex and importantly the thalamus [11]. This is important as this is what the device targets through afferent (towards the brain) signals via the median nerve [9]. Because of this, this project will focus on PD-T and ET.

3.2 Parkinson's Disease - Tremor

Tremor is a very common symptom of Parkinson's Disease (PD), In a study of 100 cases of confirmed PD, 69% experienced tremor at the onset of the disease with more developing tremor during disease progression [14]. PD itself has a prevalence of about 0.3% of the entire population and 1% of the population older

than 60 years [15]. Tremor presents in multiple forms, the most common being a combination of rest and kinetic tremor [10].

An important feature of the disease is that tremor and motor effects vary with time and factors such as blood pressure and respiratory rate [2]. Situational effects such as mental stress and concentration level can affect tremor scores [3]. These effects cannot be predicted reliably as they can be caused by unpredictable environmental factors such as a random conversation. Predictable changes can be observed with diurnal variations in the motor activity being observed suggesting a circadian rhythm contribution [16]. These variations motivate the need for a controller in the device.

3.3 Essential tremor

ET is the most common form of tremor estimated to affect between 0.4-3.9% of the global population [17]. It is defined as an "isolated tremor syndrome characterized by bilateral upper-limb action tremor, duration of at least 3 years, with or without tremor in other locations (e.g., head, voice, or lower limbs), absence of other neurologic signs, such as dystonia, ataxia, or Parkinsonism." [12]

Similar to PDT, tremor severity varies with time. Factors that can influence tremor severity include emotional state, fatigue and environmental temperature [4]. These are unpredictable, again, suggesting an adaptive treatment could be useful.

3.4 Current Treatments For Tremor

Current treatments for tremor include drugs, deep brain stimulation(DBS), and peripheral stimulation. In this section, I will provide a brief overview of the common examples for each and the current limitations with them. The summary of this is: Drugs have limited effects, especially in the latter stages of disease progression, DBS is only available to a minority of patients, and peripheral stimulation technology has room for improvements such as closed-loop control.

3.4.1 Drugs

Pharmacological treatment is the most suitable treatment for most patients with PD or ET. Propranolol and Primidone are effective pharmacological treatment options. Unfortunately, some patients lose this efficacy with time [4]. An alternative treatment is alcohol which is very effective for most patients but comes with obvious addiction problems, as well as side effects associated with drunkenness.

For PD, the drugs used typically target the lack of dopamine in the brain. These typically increase dopamine levels or mimic dopamine [18]. Because of the mixed effectiveness and undesirable side effects, there exists a portion of both patient populations for whom these drugs are not considered to be worthwhile. This drives work on alternative treatments such as electrical stimulation therapies.

3.4.2 Deep Brain Stimulation (DBS)

DBS is electrical stimulation applied within the brain through electrodes surgically implanted in the target brain region. Research has shown DBS has been used effectively for tremor reduction in ET and PD. Specifically using high frequency DBS (120 to 180 Hz) in ventral intermediate nucleus of the thalamus can provide a significant reduction in tremor severity for both PDT and ET [19].

DBS technology is quite established but is yet to be widely implemented - about 160,000 DBS implants are used worldwide [20]. Current implementations have an effectiveness of 80% [21]. DBS works well in treating tremor but it's widespread use has been limited by both its adverse effects and accessibility issues.

Open-loop implementations of DBS may be sub-optimal due to the side effects of high-frequency DBS (including discomfort and impairment of motor function) outweigh the benefit at periods of the day when tremor suppression is not needed such as sleep and periods of reduced baseline tremor. Because of this lots of work has gone into improving the effectiveness of DBS treatment (e.g., phase locked stimulation work[22]) and using closed-loop and open-loop strategies to limit when DBS is applied [23, 24].

The major reason DBS isn't widely used is its accessibility. It's highly invasive and since many PD patients are potentially vulnerable to haemorrhage and infection, implanting the electrodes is not always an acceptably risky option when other routes such as pharmacological treatment are available[25].

3.4.3 Peripheral Stimulation

Whilst other non-invasive stimulation techniques such as transcranial magnetic stimulation [26] have been researched, many non-invasive techniques focus on peripheral stimulation. A fundamental concept to this is that by delivering stimulation to the affected limb, you are guaranteeing the connection to the tremor - as opposed to stimulating at another nerve or location. Types of stimulation include non-electrical and electrical stimulation. Non-electrical stimulation through vibration has been shown to reduce action tremor [27].

Electrical stimulation at the wrist can be divided into functional electrical stimulation (FES) which acts to trigger or suppress motor action to suppress the tremor. [28, 29] Problems with this include fatigue and pain/damage associated with the muscle activation caused by the high currents and voltages needed [30]. Alternatively, sensory neurons can be stimulated. By applying stimulation below the motor threshold, sensory nerves are selectively activated. This sends afferent signals back to the central nervous system. This has been shown to modulate physical tremor [31] which suggests that it is somehow affecting the mechanism that causes tremor at a root level. The device this project focuses on falls into this category.

3.4.4 Phase locked Peripheral Stimulation

The group have been developing a wrist-mounted peripheral stimulator. This aims to disrupt central oscillators with affluent signals to the thalamus caused by stimulation to the median nerve in the wrist.

This is based on work with DBS involving stimulation of the thalamus and phase-locked stimulation of the thalamus [22].

Phase-locked stimulation aims to stimulate the thalamus at the right time in the oscillatory cycle to suppress the oscillations, breaking the coupling of different central oscillators in the brain, and suppressing tremor. This effect is similar to a timed push on a playground swing. Timed right the swing's oscillation amplitude will be driven larger. Timed incorrectly the driving force counteracts the swing's movement and suppresses the oscillation. This suppression effect is what phase locked stimulation hopes to exploit.

By applying stimulation to the thalamus (using DBS) phase-locked to wrist tremor, the group managed to modulate tremor severity in ET patients. showing the potential to both increase and decrease the severity of the tremor given different phases [22]. Transferring this to peripheral stimulation of ET patients was not found to give significant effects.

3.4.5 Time varying tremor and treatment

PDT and ET both evolve with time and other external factors. These were discussed in sections 3.2&3.3 with reasons such as emotional state, fatigue and environmental temperature affecting tremor severity. Other potential factors effecting the objective function between stimulation parameters and change in tremor severity could include medication, device positioning, variation in mechanical effects of how the arm is held and dimensions of the arm. In addition to all of this, it is suggested that controllers which can account for biological rhythms such as circadian rhythms can optimise patient outcomes for diseases such as PD [7].

To make a device that can capitalize on maximum tremor suppression at a specific stimulation phase, it will have to be able to account for all these effects, many of which are non-predictable transient effects. This is explored further in section 3.8, exploring how best to deal with the time-varying optimisation problem

3.5 Technology Stack

The device is wrist mounted to position the electrodes above the median nerve. It targets the thalamus and subthalamic nucleus with phase-locked stimulation delivered with afferent signals through the median nerve [9]. The device is pictured with a diagram of its technology stack in figures 1 & 2

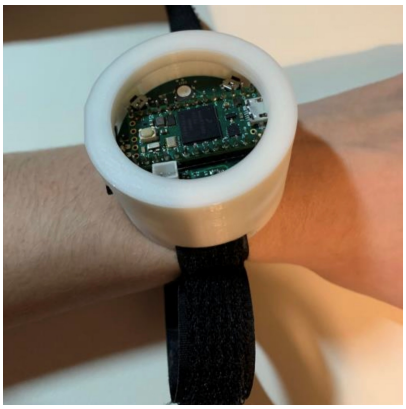


Figure 1: Photo of the prototype device, showing wrist mounting, case, and internal electronics.

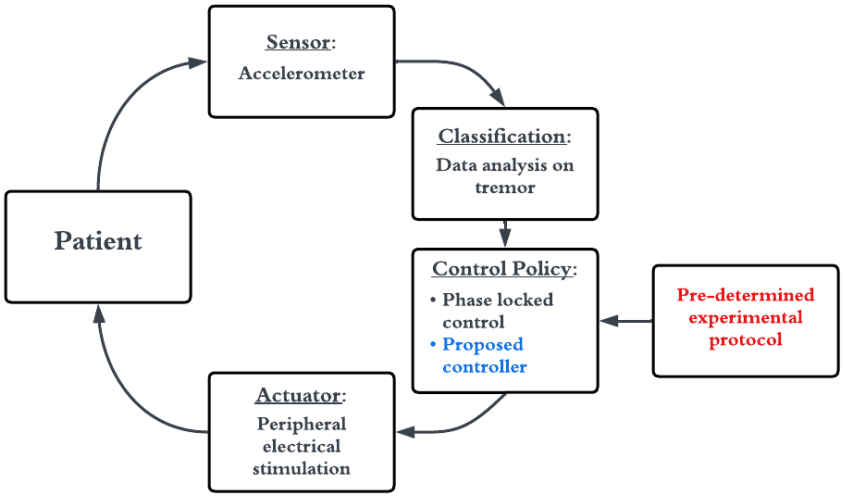


Figure 2: The technology stack for the prototype device. In red is the current pre-determined control sequence which will be replaced with the integrated on-device controller from this project (in blue).

It uses an Adafruit triaxial accelerometer and a real-time phase tracker similar to that of McNamara et al. [32] to measure the phase of the tremor oscillation. The accuracy of this allows the device to select between 6 different phase options: [0, 60, 120, 180, 240, 300] degrees. It can deliver stimulation at frequencies that are multiples of 52Hz [52, 54, 156]. Pulse width is more flexible but increments of 50μs have been decided as sensible for the device. Practically there is an upper limit to avoid causing patient discomfort. Anecdotally this may lie close to 400μs. Stimulation is at 100V with a current of between 2 to 10 mA.

Change in tremor severity is calculated by comparing a one-second average of tremor amplitude from just before the stimulation is applied, and a one-second average of tremor amplitude at the end of the stimulation. This gives outputs of 0 = no change, -1 = full elimination of tremor, +1 = double tremor severity.

For any controller designed for the device, it will need to minimise an objective function, a mapping between stimulation parameters and the change in tremor severity. The three stimulation parameters that are suitable to vary are phase, frequency, and pulse width and are what will be considered in section 6.

In initial testing, the device has been implemented to cycle stimulation for 10 seconds on and 10 seconds off. This gives a target running time for the controller of fewer than 10 seconds so it can be run each stimulation cycle. This will allow for the fastest dynamic behaviour. The microcontroller used in the device, the Teensy 4.0, has 1024K of memory. This puts an additional constraint on the control algorithm as it must be efficient enough to be run in less than 10 seconds on the limited memory device.

3.6 Existing Controller Technology

The design problem is to design a controller that picks a set of stimulation parameters to minimise tremor. First, we consider a simplified (time-invariant) situation and explore what parameter selection methods exist. The design problem can be reduced to selecting the best set of inputs (the stimulation parameters) for a black box function (body/device) for minimising an output (tremor severity) as shown in Figure 3.

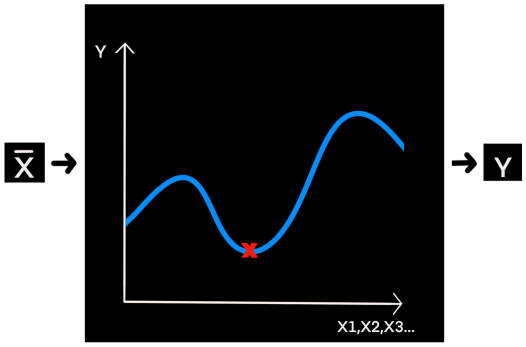


Figure 3: Simple black box function representation of design problem. A mystery objective function(blue) between inputs x, and output y exists. The aim is to find the input set that gives the minimum output(red)

From now we will deal with the abstracted black box minimisation problem. A set of inputs will be represented by $\boldsymbol{x}$, the black box/objective function $f(\boldsymbol{x})$, and our sampled tremor severity metric/output is $y = f(\boldsymbol{x}) + z$ where z is 0-mean, normally-distributed, additive noise: $z \sim \mathcal{N}(0, \sigma_z^2)$. Our aim is to find $\boldsymbol{x'}$ that gives a minimum for the output. For this, a few strategies exist:

Option A = Search Methods

These methods use an initial search over the parameter space where the parameter set that gives the minimum output is reused for the future.

One of these methods is *grid search* where the input parameter space is sampled at regular intervals. The main problem with this is that to get a good resolution of the parameter space and hence find a good minimum, a high sampling density is needed. When many input parameters are to be optimised, the number of samples needed increases. In our application, the cost of one sample is high so 1000+ samples (10 sample locations for 3 parameters) is infeasible.

Another search method is random sampling. The parameter space is sampled randomly using a uniform distribution. This leaves it likely that areas of the parameter space are left unexplored and minimums are missed. It is often employed with a stopping criterion, the parameter space is sampled randomly until a suitable output is found then the search is stopped. This can require many samples to obtain a suitable minimum.

Option B = Algorithmic Heuristic Methods

These methods search the parameter space in a smarter way. The search methods will preferentially explore regions of the parameter space if an initial search implies they will be more fruitful. This

is similar to how manual parameter tuning (like what is used by many clinical devices currently) is performed. Examples of these methods include the Genetic Algorithm. This is where promising samples in the parameter space will give rise to future samples around that region in the parameter space [33]. These methods don't guarantee convergence and don't work with the time-variant optimisation problem presented later. This is because previously promising samples of the parameter space will potentially no longer be relevant or correct when the objective function has changed.

Option C = Surrogate models

In surrogate models, initial samples feed a model which is used to predict outputs over the parameter space. This is then used to choose the next sample.

This is the premise of **Bayesian Optimisation**. The core concept of Bayesian Optimisation is to minimise or maximise an unknown function in as few samples as possible. The method works by fitting a **Gaussian Process** (GP) model over the parameter space using data already collected. An acquisition function is evaluated over the model to choose the next sample parameters such that the minimum/maximum is found effectively.

A Gaussian process is a Gaussian distribution for a single output over a parameter space. The parameter space can have any number of dimensions > 0. This means for any set of parameters within the space, a Gaussian distribution with mean μ and standard deviation σ exists for the output. A Gaussian process is a function described by a mean and a standard deviation that vary over $\boldsymbol{x}$, a vector of input parameters within the defined parameter space: $\mu(\boldsymbol{x}), \sigma(\boldsymbol{x})$. The maths used to fit a GP to the data (Gaussian Process Regression) will not be covered in this write-up as it is covered well in the literature. A good intuitive tutorial is by Jie Wang [34].

Algorithm 1 Basic Bayesian Optimisation

Initialise with N samples
repeat
  Fit Gaussian Process Model to data-set of samples
  Apply acquisition function to model, select next parameter set to sample at
  Sample at selected parameter set and update data-set
until Termination condition is met e.g., M iterations performed, minimum change in model minimum/maximum

Bayesian Optimisation is a good option when the costs of exploring the parameter space are expensive(sub-optimal therapy) and the underlying function is unknown. It converges quickly using few samples, perfect for the high sample cost. This reduces the amount of tuning time needed at the beginning of the device's operational life as well as the number of sub-optimal stimulations delivered.

The two key parts to Bayesian optimisation are the kernel used for the Gaussian process (this affects the quality of the model fit) and the acquisition function (this affects how the model converges and what samples are taken). The kernel choice will be readdressed later in section 5.2 when the control method is fitted to the specific application of our device as it very much depends on the nature of the objective

function and noise. The acquisition function is discussed in detail now as we consider how a controller can implement Bayesian optimisation.

3.7 Bayesian Optimisation Acquisition Functions

The acquisition function chooses where in the parameter space to sample next, making this decision based on the GP model. Common acquisition functions include: Expected Improvement (EI), Probability of Improvement, Entropy Search, and Confidence bound (UCB & LCB) [35]. I will now run through some of them. Most acquisition functions involve maximising a utility function over the parameter space. The utility function represents how the model is expected to change, and the maximum value should give the parameter set for the best improvement of the model minimum/maximum.

3.7.1 Probability of Improvement

Define a utility function over the model as 1/0 for better/no-better than the previous maximum. Evaluate the expected value for the utility function over the parameter space, as predicted by the model. The parameter set with the largest expected utility function corresponds to the parameter set with the highest probability of improvement and is selected for the next sample.

$$\bar{x}' = argmax\{u(\bar{x})\}, u(\bar{x}) = \begin{cases} 1 & \mu_n(\bar{x}) > f'_{n-1} \\ 0 & \mu_n(\bar{x}) < f'_{n-1} \end{cases} \tag{1}$$

3.7.2 Expected Improvement

This is the same as the Probability of Improvement acquisition function, except that the utility function is replaced to be $|\mu_n(x) - f'_{n-1}|/0$ for better/no-better. This includes the magnitude of the improvement within the utility function. Therefore, the acquisition function chooses the largest expected improvement.

$$u(\bar{x}) = \begin{cases} |\mu_n(x) - f'_{n-1}| & \mu_n(\bar{x}) > f'_{n-1} \\ 0 & \mu_n(\bar{x}) < f'_{n-1} \end{cases} \tag{2}$$

3.7.3 Entropy search

The aim is to minimise the uncertainty of the parameter set for the current model minimum. The utility function is given as the difference in expected entropy for the parameter set.

$$u(\bar{x}') = H[\bar{x}|D] - H[\bar{x}|D, x', f(x) \tag{3}$$

3.7.4 Upper Confidence bound

Upper Confidence Bound (UCB) is used for maximisation/ Lower Confidence Bound (LCB) is used for finding a minimum but otherwise, they act on the same principle. For this project, LCB fits but the

literature mostly focuses on UCB so I will discuss the method in terms of maximisation and UCB.

This acquisition function selects a parameter set that maximises the mean of the model + a confidence interval using the model's standard of deviation. The parameter denoted β controls the size of the confidence interval with $\sigma(x)$ controlling the shape over the parameter space. A small β prioritises exploiting the model minimum/maximum. A larger β prioritises exploring the parameter space. This method forms the basis for my later design. β **is an important variable and will be referenced for the rest of the document.**

$$Maximise : a_{ucb}(x;\beta) = \mu(x) + \beta\sigma(x) \tag{4}$$

$$Minimise : a_{lcb}(x;\beta) = \mu(x) - \beta\sigma(x) \tag{5}$$

As shown, the difference with LCB is just using minimisation and subtracting the confidence bound instead of addition.

An abstract understanding of this method is explaining this as maximising a utility function of the weighted sum of the expected outcome for the parameter set, and the expected information gain of the parameter set (the Entropy). This is more in line with the previous methods evaluating expected values over the parameter space. This understanding is useful for the method I use later in section 4.1.

From our GP:

$$P(f(x)) = \frac{1}{\sigma(x)\sqrt{2\pi}} e^{-\frac{(f(x)-\mu(x))^2}{2\sigma^2}} \tag{6}$$

Explicitly $\mu(x)$ is the expected outcome for the parameter set x:

$$E[f;x] = \int_{-\infty}^{\infty} f(x)P(f(x))\,df(x) \tag{7}$$

$$[E[f;x] = \int_{-\infty}^{\infty} f(x)\frac{1}{\sigma(x)\sqrt{2\pi}} e^{-\frac{(f(x)-\mu(x))^2}{2\sigma^2}}\,df(x) \tag{8}$$

$$E[f;x] = \mu(x) \tag{9}$$

$\beta\sigma$ is related to the expected information of the parameter set x:

$$h[x] = E[I(x)] = \int_{-\infty}^{\infty} -P(f(x))log(P(f(x)))\,df(x) \tag{10}$$

$$h[x] = \int_{-\infty}^{\infty} -\frac{1}{\sigma(x)\sqrt{2\pi}} e^{-\frac{(f(x)-\mu(x))^2}{2\sigma^2}} log(\frac{1}{\sigma(x)\sqrt{2\pi}} e^{-\frac{(f(x)-\mu(x))^2}{2\sigma^2}})\,df(x) \tag{11}$$

$$h[x] = log(\sigma(x)\sqrt{2\pi e}) \longrightarrow \beta\sigma + \alpha \tag{12}$$

This holds for any chosen base of the logarithm. Different bases will give different units (e.g., base 2 =

bits which are used later in the project). $h[x] < 0$ is mathematically valid, as this is differential entropy, but this cannot be used in a utility function as it would flip the maximisation of the utility function. We, therefore, approximate the logarithm with a linear function of $\sigma(x)$. α and β depend on chosen base for the logarithm (which is arbitrary) and also what domain of $\sigma(x)$ we want our approximation of $H[x]$ to be most accurate for. As both the linear and the original logarithm are monotonic functions, maximising $\beta\sigma(x)$ will also maximise $h[x] = log(\sigma(x)\sqrt{2\pi e})$. α is left out of the utility function as it is constant over the parameter space. Returning to the acquisition function:$a_{ucb}(x;\beta) = \mu(x) - \beta\sigma(x)$, the variable β now affects: *a)* The accuracy of the estimate of the entropy $h[x]$ and *b)* The bias between exploration (sampling a region in the parameter space of high entropy) and exploitation (sampling a region in the parameter space with an attractive prediction).

β can be made a function of time/iteration such that the acquisition function guarantees no regret[36]. This is done by increasing β in a sub-linear fashion so the $\beta\sigma(x)$ term goes to zero as the iteration number goes to infinity. This means the acquisition decision settles on pure exploitation of the correct model, averaging the regret to 0 as time increases to infinity.

3.8 The Real, Time-Varying Problem

The black box representation before was a simplification. Our design problem needs to allow for the objective function to change, in a non-predictable fashion, over time.

ET and PD-T change in severity and nature over time, as discussed in greater detail in section 3.4.5, ideally this can be accounted for with the controller. The electrode-skin interface for the device is also subject to non-predictable transient effects such as pressure, location and humidity/sweat. This is incorporated into the black box model with a second vector of known and unknown, time-varying inputs.

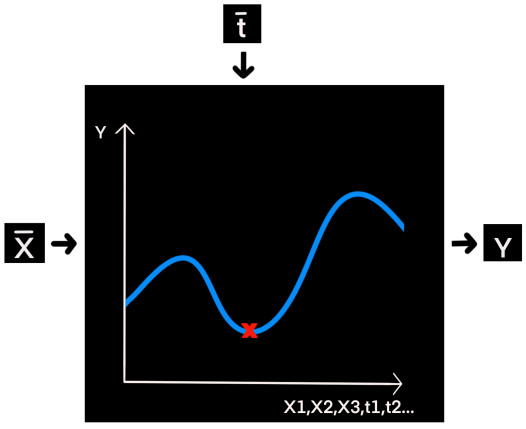


Figure 4: Time varying black box function representation of design problem. Similar to before, the design problem is to find the parameter set of X that minimises the output Y. This time, however, the function (blue) and hence the minimum (red) also is affected by time-varying inputs t. These cannot be controlled so the optimisation problem is still over just X.

The controller needs to account for this. At any one point in time, the previous time invariant black-box optimisation problem still exists, but now we cannot guarantee previous samples or minimums still apply.

3.8.1 Including time as a model parameter

One option to account for this is to attempt to model changes over time. This has been previously worked on in a 4YP project in 2021-2022 [37] with work on including circadian rhythms. This was done by including time of day as a model parameter. I have decided this type of prediction is infeasible to include in the controller for the following reasons:

- To include the time scale that circadian rhythms act over requires a large matrix inversion when fitting a Gaussian process model. This would make on-device evaluations of the model impossible and move the control algorithm off-device, likely to cloud computing. This is associated with larger energy costs [38], reliability issues caused by dependency on internet access, and greater complexity and financial running costs for commercial use.
- It lengthens the iterations period to daily, preventing fast convergence on an optimal parameter set.
- It is unable to deal with short-time scale and unpredictable changes to the objective function.

3.8.2 Contextual Bayesian Optimisation

Contextual Bayesian optimisation allows for outside environmental factors to be considered into Bayesian optimisation [39]. This allows the model to treat different situations separately. This, however, requires the context to be fed to the model. This would be impossible for the situation as factors such as emotional state would be impractical to measure an input into the device.

3.8.3 Time-Varying Gaussian Process Bandit Optimisation Problem

If we accept that predicting the changes to the system and its objective function is impossible as at least some of the changes are unpredictable we can define the problem as a Time-Varying Gaussian Process Bandit Optimisation Problem [8, 40, 41]. At the moment there are a few strategies for dealing with this, all involving modifying the GP-UCB method. As discussed earlier in section 3.7.4, in the UCB method the β variable gives the balance of exploration and exploitation. In time-invariant situations, β is a function of iteration - β increases in a sub-linear fashion to guarantee convergence and 0 regret. This over-samples the parameter space. 0 regret is not achievable with the unpredictable time-varying problem as the model cannot predict the unpredictable changes to the objective function.

The two groups of modifications to the GP-UCB method are:

TV-GP-UCB/SW-GP-UCB

Previous samples are forgotten so that old samples, and hence those that are less relevant, influence the model less than newer samples. There are two versions of this, the TV-GP-UCB method involves forgetting the samples in a smooth fashion [8]. SW-GP-UCB uses a 'sliding window' so that only the previous samples within that window influence the model [40]. Both of these methods allow the model to adapt continuously to changes in the objective function. My method, developed in the design section is based loosely on this method.

R-GP-UCB/ET-GP-UCB

For R-GP-UCB (Resetting) the Gaussian process model is reset periodically so the Bayesian optimisation algorithm re-explores the parameter space [8]. ET-GP-UCB (Event Triggered) is a modified version of this where the algorithm can detect an event and trigger a reset[41]. I did not think these methods were suitable for this project as I expect there to be smooth changes to the objective function from influences like time of day. I later compare the speed of how the algorithm deals with a discrete objective function change with ET-GP-UCB.

These methods still follow the same rules for β which give oversampling when the model is correct.

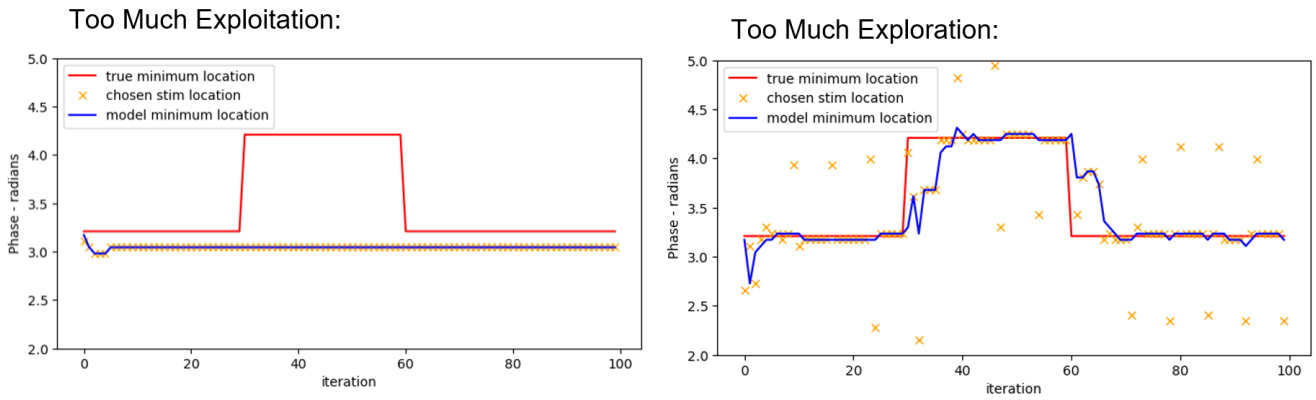


Figure 5: SW-GP-UCB implemented on a sinusoidal objective function over the dummy periodic variable of phase. The objective function's minimum location has a step change (red). The GP model mean's minimum (blue), chosen sample location (orange) Left: Explore/Exploit Bias (β) is set too low so the model never explores the parameter space and instead picks the incorrectly predicted minimum, Right: Explore/Exploit Bias (β) is set too high so the model always explores the parameter space instead of using the correctly predicted minimum.

As shown in the figure, the choice of β will affect how well the method is following changes in the objective function. In time-variant systems, it needs to shift to favour exploration when the model has diverged from the objective function and shift to favour exploitation when the model has converged on the objective function. With a fixed β or a β with a predefined evolution, the correct β cannot be applied to match the situation. Therefore a compromise of slightly over-sampling has to be used. With our application, sampling could cause an increase in tremor severity, so exploration, when the model is correct, has a high penalty. Not adapting to a change in objective function could also cause an increase in tremor severity so not exploring could give a high penalty. I have, therefore, designed a new method in the design section, that makes β a dynamic variable. This combined with the forgetting of previous samples gives dynamic exploration/exploitation

3.9 Design Specification

From the literature review, a set of requirements for the design can be made. These are what the design needs to achieve in order to be better than existing technologies.

The device is needed because of the ineffectiveness of pharmacological treatments and the inaccessibility of DBS. In order for this device to work better than existing technology, it needs to deliver the optimal set of stimulation parameters - including the phase for phase-locked stimulation, a key aspect of the device. The other parameters include frequency and pulse width. In order for the optimal parameters to be continually selected the controller must be able to adapt as they change.

The controller must be on device and fast enough to run between stimulation cycles (less than 10 seconds using less than 1024K of memory).

Initial design directions taken from the literature review are to use a modified version of the SW-GP-UCB method to continually track the minimum of the changing objective function. The key modification that will be made is enabling dynamic changes to the explore/exploit bias. This should stop unnecessary exploration when the model is correct and allow for the model to update when the model is wrong.

4 Design

The design of the controller is based on the SW-GP-UCB method. In this section, I design an algorithm that continuously changes β, dynamically changing the explore/exploit bias. A reminder of the UCB acquisition function:

$$Maximise : a_{ucb}(x;\beta) = \mu(x) + \beta\sigma(x) \tag{13}$$

The explore/exploit balance must be made dynamic because in periods when the model is correct, exploration is wasteful. In periods when the objective function has changed/is changing the model is wrong so exploration is needed and exploitation is ineffective. This is shown in Figure 5.

4.1 Estimating the Accuracy of the Model

We wanted to make the exploration/exploitation dynamic with how well the model fits the objective function. Assuming the system takes noisy readings, the uncertainty $\sigma(x)$ will never go to zero, instead approaching the SD of the additive noise applied to the sample. This is because $\sigma(x)$ encapsulates both the uncertainty of the model mean and the model's predicted standard of deviation over the parameter space. We do want to reduce exploration to nearly zero if the model is correct though. So β itself has to go to zero. To do this we will make β a function of how correct our model is.

We need to estimate how correct our model is. For this, I considered a few options:

- Compare the sample to the expected output (is the model mean correct)
- Compare the information gain to the expected information gain (Entropy) (is the model uncertainty correct)
- Compare the sample to the expected output + confidence bound (is the combination of model mean and model uncertainty correct)

These each work as follows:

4.1.1 Model mean accuracy

The most obvious metric of this is the difference between the sample taken and the previous model mean at the chosen parameters.

$$|f'_{n-1}(x^*_{n-1}) - \mu(x^*_{n-1})| \tag{14}$$

The main problem with this is that for noisy observations of $f(x)$ this will become:

$$|f_{n-1}(x^*_{n-1}) - \mu(x^*_{n-1})| + \mathcal{N}(0, \sigma_z^2) \tag{15}$$

Using this to choose a β value will not solve the problem as β will not go to zero as the model becomes correct.

4.1.2 Model uncertainty accuracy

We already have a measure of the uncertainty of the model: $\sigma(x)$. The more correct $\sigma(x)$ is, the more correct the next decision about where to sample will be based on previous samples.

It is less obvious what to use to measure the correctness of $\sigma(x)$ than for $\mu(x)$. We need a measure of how unexpected the result was - this leads to using information gained about the model from the result. A measure of predicted information gain is also needed for which we can use entropy. What we want is for our metric to be larger when the information gained was more than the entropy, and to be smaller when the information gained was less than the entropy. This will allow the exploration/exploration to be scaled accordingly.

As the input parameter set has already been selected, we reduce the probability distribution from a multivariate Gaussian (which governs the Gaussian process) to a univariate Gaussian. This has a continuous model prediction $f'_{n-1}(x^*_{n-1})$ with a probability density function of $\frac{1}{\sigma(x)\sqrt{2\pi}} e^{-\frac{(f(x)-\mu(x))^2}{2\sigma^2}}$. It is easy to calculate differential entropy, as done in section 3.7.4 to get $h[x] = log(\sigma(x)\sqrt{2\pi e})$. The probability of the sample outcome $y(x^*_{n-1})$ is defined by a probability density function. It is, therefore, not immediately possible to get a measure of information gained from this sample without comparing it with the entropy of the model built including the sample. Calculating this is not ideal for the speed and efficiency of the program (something which is key with the limited power application of our wrist-mounted device.) Instead the probability density function is discretised into probabilities of obtaining a sample between two values. This allows the calculation the 'probability of the sample observed' as: $P(y(x^*_{n-1}) - \frac{\Delta}{2} < f'_{n-1}(x^*_{n-1}) < y(x^*_{n-1}) + \frac{\Delta}{2})$. Where Δ is the discretisation of the distribution. This evaluates as:

$$P(y(x^*_{n-1}) - \frac{\Delta}{2} < f'_{n-1}(x^*_{n-1}) < y(x^*_{n-1}) + \frac{\Delta}{2}) = \int_{y(x^*_{n-1})-\frac{\Delta}{2}}^{y(x^*_{n-1})+\frac{\Delta}{2}} \frac{1}{\sigma(x^*_{n-1})\sqrt{2\pi}} e^{-\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2}} df'(x^*_{n-1}) \tag{16}$$

for small Δ:

$$\approx \Delta \frac{1}{\sigma(x^*_{n-1})\sqrt{2\pi}} e^{-\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2}} \tag{17}$$

The self-information of this sample can then be evaluated as the negative logarithm (any base) of the probability:

$$I[y(x^*_{n-1})] \approx -log(\frac{\Delta}{\sigma(x^*_{n-1})\sqrt{2\pi}} e^{-\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2}}) \tag{18}$$

Δ is chosen as dividing the range of $\pm 3\sigma(x^*_{n-1})$, which is 99.7% of the distribution, into 2^n sections. This division requires n bits which becomes useful when we apply this discretisation to the differential entropy of the distribution.

$$\Delta = \frac{6\sigma(x^*_{n-1})}{2^n} \tag{19}$$

The information now becomes:

$$I[y(x^*_{n-1})] \approx \log(\frac{6}{2^n\sqrt{2\pi}}) + \log(e)\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2}) \tag{20}$$

From here I use **base two for the logarithms, measuring information in bits.** This is convenient with the 2^n divisions.

$$I[y(x^*_{n-1})] \approx n - 1.26 + 0.721\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2}) \tag{21}$$

With this discretisation applied to the probability of the sample, it also needs to be applied to the entropy of the model. The n-bit quantisation of the continuous random variable to a discrete random variable changes the entropy by:

$$H([X]_{2^{-n}}) \approx h(X) + n \tag{22}$$

[42] This gives:

$$H[f'_{n-1}(x^*_{n-1})] \approx \log_2(\sigma(x^*_{n-1})\sqrt{2\pi e}) + n \tag{23}$$

Two options I have considered for how to use these measures are:

1. The difference between information gained from the sample and information gain expected before the sample(entropy).

$$(I[y(x^*_{n-1})] - H[x^*_{n-1}]) \tag{24}$$

   Written out this becomes:

$$n - 1.26 + 0.721\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2}) - (\log_2(\sigma(x^*_{n-1})\sqrt{(2\pi e)}) + n) \tag{25}$$

$$-1.26 + 0.721\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2}) - \log_2(\sigma(x^*_{n-1})\sqrt{(2\pi e)}) \tag{26}$$

It is worth noting how this evaluates Given:$H > I$ then $I - H < 0$
$H < I$ then $I - H > 0$

2. The information gained from the sample as a proportion of information expected before the sample(entropy).

$$\frac{I[f'_{n-1}(x^*_{n-1}); x^*_{n-1}]}{H[x^*_{n-1}]} \tag{27}$$

This evaluates as:

$$\frac{n - 1.26 + 0.721\frac{(f'(x^*_{n-1}) - \mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2})}{\log_2(\sigma(x^*_{n-1})\sqrt{(2\pi e)}) + n} \tag{28}$$

This should always be greater than 0. But it doesn't always evaluate as this due to how the method has discretised the probability density function (pdf). Differential entropy can be negative as the distribution's pdf does not have to be less than zero at any one point (it just has to integrate to one over its domain). When converting from differential entropy to entropy was used $H([X]_{2^{-n}}) \approx h(X) + n$. As $h(x) = \sigma(x^*_{n-1})\sqrt{2\pi e}$, for small $\sigma(x)$ it can be negative and large. This means the approximation for entropy will be negative when $\sigma(x^*_{n-1})\sqrt{2\pi e} > n$. One way to prevent this from happening would be to increase n to infinity. This would however increase the expected and sample's information to infinity as for a sample to fall within an infinitely small range would be infinitely unlikely. Instead, I used a practical solution of limiting σ so that it cannot take a value less than $\tilde{}\frac{2^n}{\sqrt{2\pi e}}$ just for when the algorithm evaluates this metric. This also helps to prevent overflow errors as the method divides by $\sigma(x)^2$.

4.1.3 Model mean and uncertainty accuracy

The other explanation for confidence bounds is that $\mu(x) \pm \beta\sigma(x)$ is the confidence interval of β standard deviations about the mean. This allows you to calculate the probability of seeing your sample within this interval. If the sample taken lies in the interval, it suggests the interval was too big. If the sample taken lies outside the interval, it suggests the interval was too small. Ideally, the sample will fall at the edge of the interval - In this case, the bound used for the decision reflects the sample taken so the decision was well made. Establishing a sensible metric for this:

The error of the model prediction:

$$|f(x) - \mu(x))| \tag{29}$$

1/2 the Confidence bound length:

$$\beta\sigma(x) \tag{30}$$

Comparing the two:

$$\frac{|f(x) - \mu(x)|}{\beta\sigma(x)} \tag{31}$$

4.2 Beta as a Dynamic Variable

My proposed strategy is to make β a function of one of these four accuracy metrics. This will allow the value for β to increase/decrease dynamically as the relation between the model and the objective function changes (as the model converges or the objective function diverges).

The methods I have tried:

4.2.1 Make β a linear function of the difference between the predicted output and sample.

$$\beta_n = K|f'_{n-1}(x^*_{n-1}) - \mu(x^*_{n-1})| \tag{32}$$

The hyper-parameter K is introduced to scale the sensitivity.

4.2.2 Make β a function of the difference between the Information gained and the Information gained expected.

$$\beta_n = \beta_{n-1}K^{I-H} \tag{33}$$

The hyper-parameter K is introduced to scale the sensitivity. K is raised to the power of the difference as this maps I-H: $-ve \to 0 \to +ve$ to $0 \to 1 \to> 1$. This becomes:

$$\beta_n = \beta_{n-1}K^{-1.26+0.721\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2})-\log_2(\sigma(x^*_{n-1})\sqrt{(2\pi e)})} \tag{34}$$

This was changed to a more tuneable version:

$$\beta_n = \beta_{n-1}K^{A\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2})-B\log_2(\sigma(x^*_{n-1}))} + C \tag{35}$$

4.2.3 Make β a function of the fraction Information gained over the Information gain expected.

$$\beta_n = K\frac{I}{H} \tag{36}$$

The hyper-parameter K is introduced to scale the sensitivity. This evaluated out is:

$$\beta_n = K\frac{n - 1.26 + 0.721\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2})}{\log_2(\sigma(x^*_{n-1})\sqrt{(2\pi e)}) + n} \tag{37}$$

This function is best visualised as a function of $(f'(x^*_{n-1}) - \mu(x^*_{n-1}))$ and a function of $\sigma(x^*_{n-1})^2$

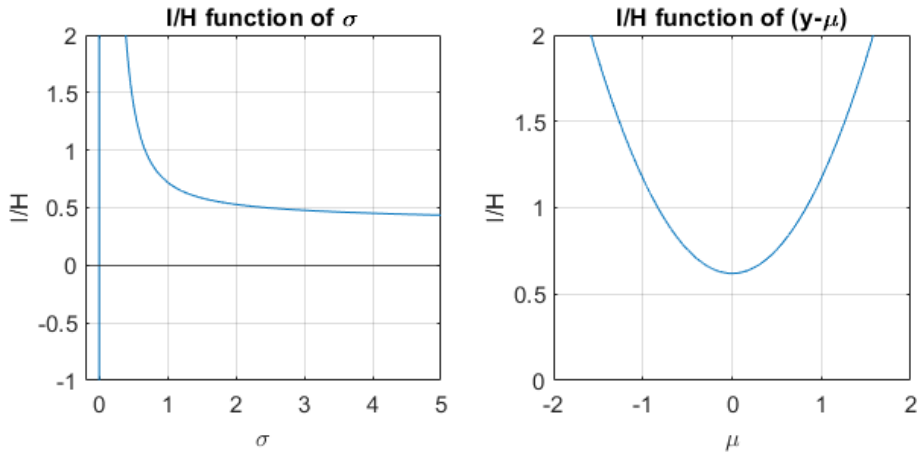


Figure 6: $\frac{I}{H}$, Information gain by the sample over the entropy of the model. This shows how varying σ and varying μ affect the value for $\frac{I}{H}$.

Things to note are:

- for a given $\sigma(x^*)$, there will be a minimum over $(f'(x^*_{n-1}) - \mu(x^*_{n-1}))$. This is because of the $n - 1.26$.
- As $\sigma(x^*)$ goes to zero, the function first gets very large, then decreases to a negative asymptote at $\sigma(x^*) = 0$. This will be a problem for the acquisition function as β must be greater than zero.

As discussed in section 4.1.2 a practical method for preventing this function from going negative is to limit $\sigma(x^*)$ to greater than some small value. Instead of this, however, by looking at the nature of the function we can simplify the equation by removing the logarithm from the denominator:

$$\beta_n = K(n - 1.26) + K\frac{(f'(x^*_{n-1}) - \mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2})) \tag{38}$$

This has eliminated the negative asymptote, which would have been cut off anyway, without dramatically changing the shape of the function elsewhere. We do still need to limit sigma to greater than some small value to prevent large results that give overflow problems.

The final modification is to remove $n - 1.26$ as this just shifts the minimum over $(f'(x^*_{n-1}) - \mu(x^*_{n-1}))$ to greater than 0. This isn't really needed for the function intended and will just give another calculation for the microprocessor to perform, very slightly slowing down performance. This leaves a practical version shown in Equation 39 and shown in Figure 7.

$$\beta_n = K\frac{(f'(x^*_{n-1}) - \mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2} \tag{39}$$

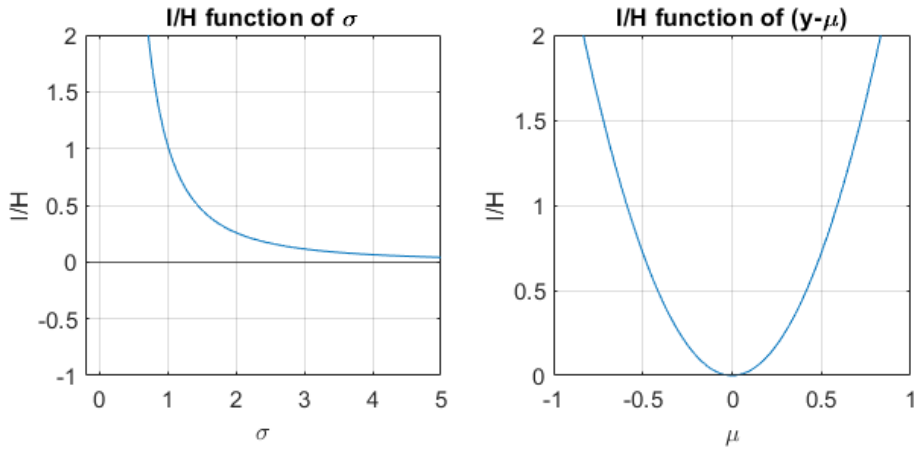


Figure 7: $\frac{I}{H}$, Information gain by the sample over the entropy of the model, simplified. This shows how varying σ and varying μ effect the value for $\frac{I}{H}$. Here the negative asymptote at $\sigma = 0$ has been removed and for $y = \mu$, $\frac{I}{H} = 0$. These simplifications give a more practical controller of: $\beta_n = K\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2}$

4.2.4 Make β a function of the difference between the sample/mean error and the acquisition function's confidence bound.

$$\beta_n = K\frac{|f'_{n-1}(x^*_{n-1}) - \mu(x^*_{n-1})|}{\sigma(x^*_{n-1})} \tag{40}$$

The hyper-parameter K is introduced to scale the sensitivity.

4.3 Choosing the Metric Used to Control Beta

Now we look at making a decision on which method to use. This requires consideration of the intended use so we no longer consider the totally abstracted problem. At this stage we have 4 different options for how to control β, each with 1-2 hyper-parameters. This is in combination with multiple hyper-parameters already part of the Gaussian process regression.

We want to establish which one will perform the best but it is incredibly inefficient to evaluate how well each one performs over a large range of combinations of hyper-parameters for a large range of possible situations with objective functions. Instead of doing this, I first chose a dummy objective function with three input parameters. The parameters were chosen to somewhat represent real stimulation parameters with one being phase (so is periodic with 2 π) and two representing non-periodic parameters for example frequency or voltage where it moves from 0 effects to full over some threshold-like value. The function over phase changes significantly at the 25th iteration, representing a very discrete change in the system that could be caused by something like the patient picking up an object. I then varied the hyperparameters for each method, recording the sample location in phase. To compare the methods, I am looking for best convergence and minimal sampling away from the model when the model is correct. A good method will have a wide range of hyperparamters that give good results as this will make the method easy to tune in an implemented setting.

I first tested a fixed β UCB method for a reference. After exploring values as shown in figure 8 the best result was using $\beta = 0.02$ as shown in figure 9.

$$a_{lcb}(x;\beta) = \mu(x) - \beta\sigma(x) \qquad (41)$$

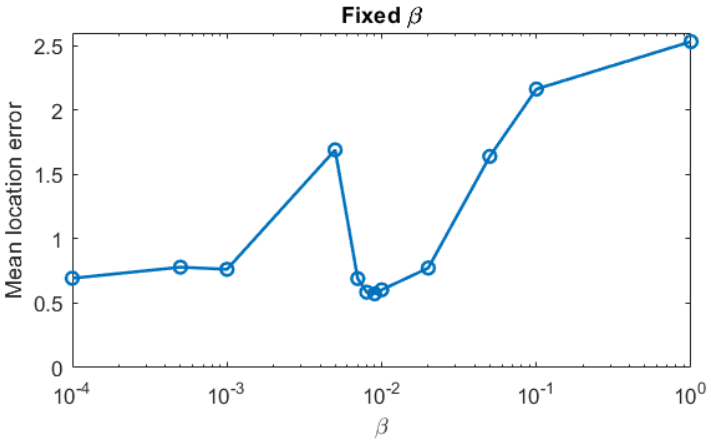


Figure 8: Results for different fixed values for β. The best result was at K = 0.02 giving a mean error in location (phase): 0.57341868

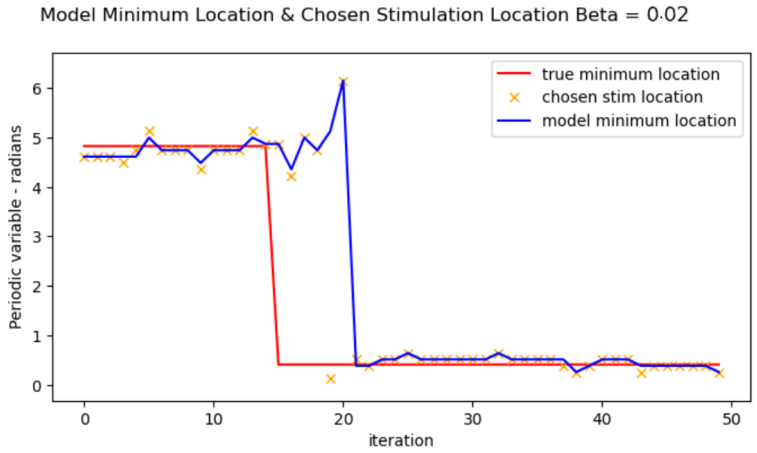


Figure 9: Tracking a sinusoidal objective function over the dummy periodic variable of phase. The objective function's minimum location has a step change (red). The GP model mean's minimum (blue), chosen sample location (orange). Here a fixed value of 0.02 for β was used.

For β values that approach 0, the model has a fixed performance with error of ≈ 0.69. This represents using just $\mu(x^*_{n-1})$ as the acquisition function. 0 exploration still finds the correct model as we have limited the number of remembered stimulations like done in SW-UCB-GP [40]. This means after relatively few stimulations at the new objective function the model will be fully based on samples at the new objective function. As β increases, first, the error gets worse, this is because the model is just exploring a non-productively about the model minimum. Then quickly the error gets better. This is when the exploration is a good balance of sampling around the parameter space, but it still exploits the model when it has a small σ. The error then gets significantly worse as β gets larger. This represents needless and too much sampling (no exploiting the model).

Because of this, we can set targets for our methods: For a **minimally acceptable method**, it should give **better than 0.69 mean location error**. A **good method** will give **better than 0.57 mean location error**. Ideally, it should be **easier to tune** - with a wide range of hyper-parameter values giving good, small errors.

Using the hyper-parameters set, I then implemented the 4 different dynamic β approaches, tuning their

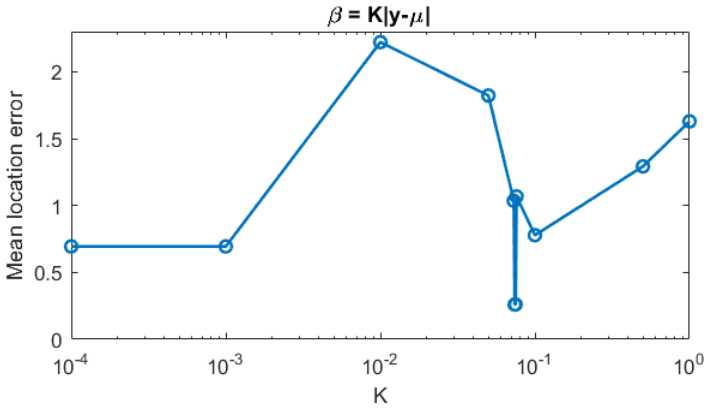
hyper-parameters to get an idea of which worked the best:

4.3.1 Make β a linear function of the difference between the predicted output and sample.

After exploring values for K as shown in figure 10 the best result was using $K = 0.075$ as shown in figure 11.

$$\beta_n = K|f'_{n-1}(x^*_{n-1}) - \mu(x^*_{n-1})| \quad (42)$$

Figure 10: Results for different values for K. This shows a very good result for one value of K but a give significant fall off for any small change to K. This will make tuning the value in a different setting incredibly difficult. The best result was a mean error in location (phase): 0.2600815 at K = 0.075



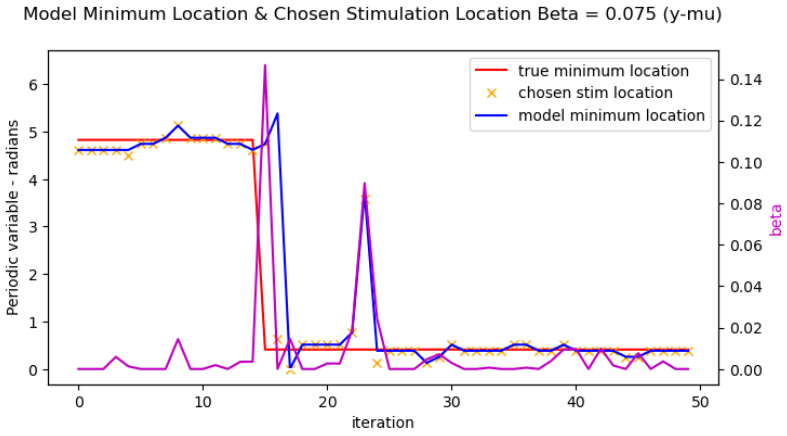


Figure 11: Tracking a sinusoidal objective function over the dummy periodic variable of phase. The objective function's minimum location has a step change (red). The GP model mean's minimum (blue), chosen sample location (orange). Here dynamic control for β was used. β (purple) was set to a proportion of the difference between the sample output and the predicted output.

There is strong performance potential with this method but only for a very narrow region of K (0.075-0.074). For nearby values of K = (0.076, 0.073) this method produces an error worse than 0.69. Because of this, even though this method can give much better results than fixed β, it would be too hard to tune correctly in a device setting.

4.3.2 Make β a function of the difference between the Information gained and the Information gained expected.

$$\beta_n = \beta_{n-1} K^{A\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{2\sigma(x^*_{n-1})^2})-B\log_2(\sigma(x^*_{n-1})} + C \quad (43)$$

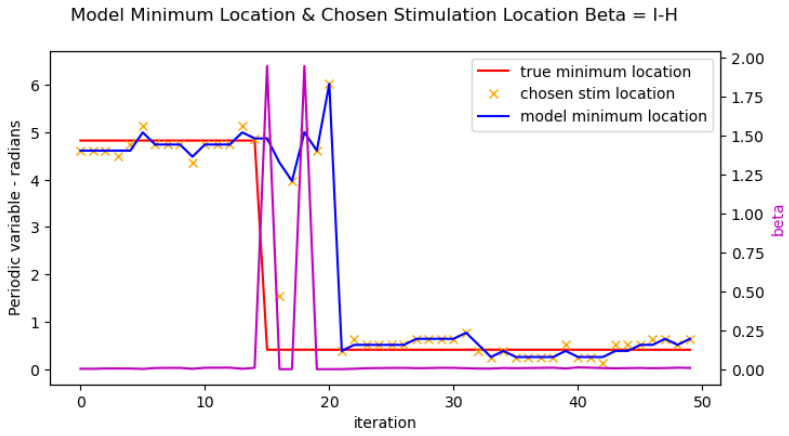


Figure 12: Tracking a sinusoidal objective function over the dummy periodic variable of phase. The objective function's minimum location has a step change (red). The GP model mean's minimum (blue), chosen sample location (orange). Here dynamic control for β was used. β (purple) was set to the difference between the information gained from the sample and the model entropy ($\beta_n = I - H$). This did not give promising results in this case the mean error in location (phase): 0.61597 for the variables: (K = 1.1, A = 2, B = 2, C = -50)

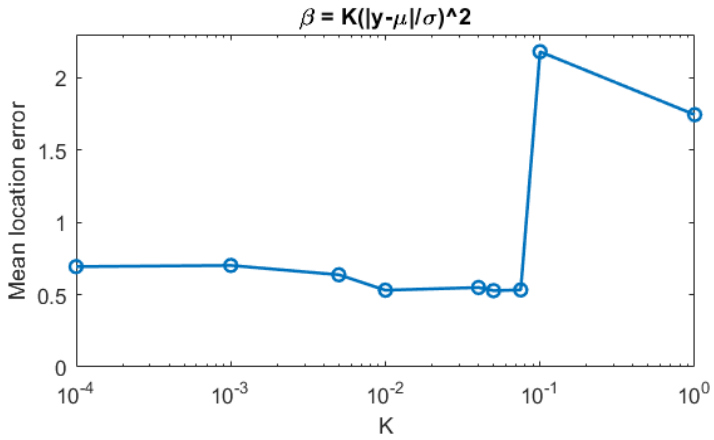
Due to the large number of hyper-parameters, this introduces a new optimisation problem. This method would introduce too much complexity when applied to a real-world case. I decided this method was not worth further development.

4.3.3 Make β a function of the fraction Information gained over the Information gain expected.

After exploring values as shown in figure 13 the best result was using $K = 0.01$ as shown in figure 14.

$$\beta_n = K\frac{(f'(x^*_{n-1}) - \mu(x^*_{n-1}))^2}{\sigma(x^*_{n-1})^2} \quad (44)$$

Figure 13: Results for different values for K. This shows a good results for a wide range of K values. This will make tuning the value easy. The best result was a mean error in location (phase): 0.5311362 at K = 0.05



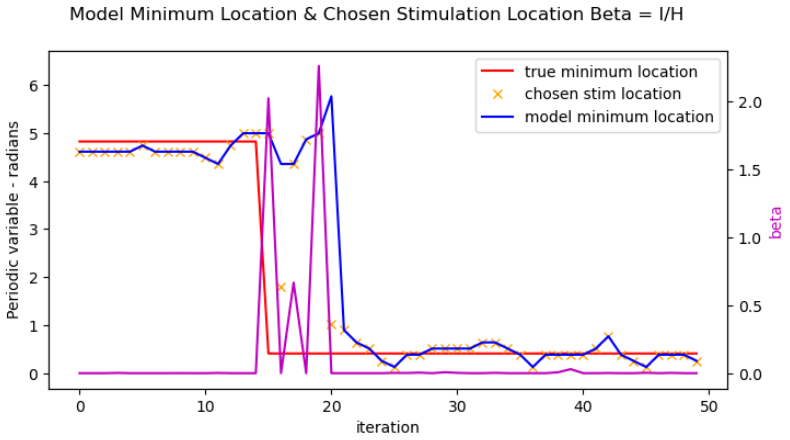


Figure 14: Tracking a sinusoidal objective function over the dummy periodic variable of phase. The objective function's minimum location has a step change (red). The GP model mean's minimum (blue), chosen sample location (orange). Here dynamic control for β was used. β (purple) was set to the simplified fraction of the information gained from the sample divided by the model entropy ($\beta_n = \frac{I}{H}$).

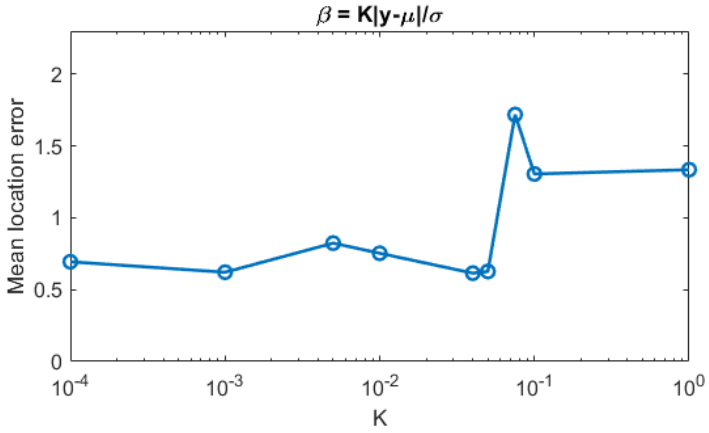
This method has good performance (< 0.57) over a good range of K and better than minimal performance over a larger range of K. The trend for this controller for β is general improvement as K is increased (making it more sensitive to changing toward exploration) until K is too large and the method suddenly fails causing a large discontinuity in error. This is because at this point the K value outweighs the fraction $\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{\sigma(x^*_{n-1})^2}$ when the model is correct and the controller should be exploiting. This means the controller would always choose to explore and so it never picks the correct parameters.

4.3.4 Make β a function of the difference between the sample/mean error and the acquisition function's confidence bound.

After exploring values as shown in figure 15 the best result was using $K = 0.04$ as shown in figure 16.

$$\beta_n = K\frac{|f'_{n-1}(x^*_{n-1}) - \mu(x^*_{n-1})|}{\sigma(x^*_{n-1})} \tag{45}$$

Figure 15: Results for different values for K. The best result was a mean error in location (phase): 0.6148815 at K = 0.04



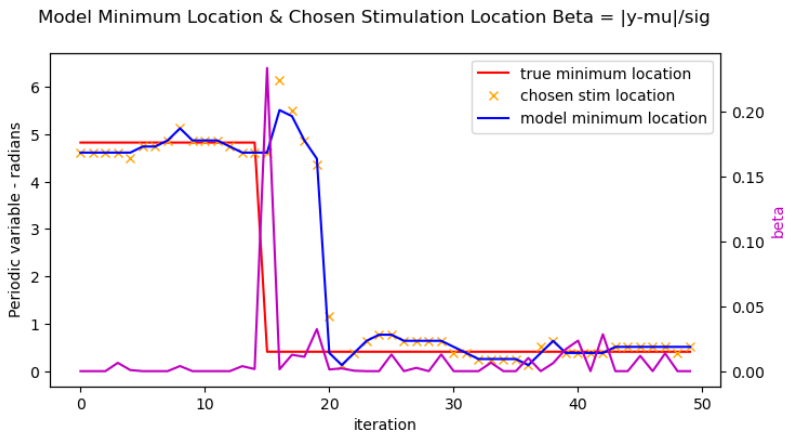


Figure 16: Tracking a sinusoidal objective function over the dummy periodic variable of phase. The objective function's minimum location has a step change (red). The GP model mean's minimum (blue), chosen sample location (orange). Here dynamic control for β was used. β (purple) was set to the difference between the sample result and the predicted result, normalised by the model's standard deviation.

Performance isn't great with better than minimum performance over an okay range but generally is quite unstable.

4.3.5 Summary and different method

From the methods that worked there is an overarching theme:

$K|f'_{n-1}(x^*_{n-1}) - \mu(x^*_{n-1})|$ gives unpredictable but good performance. Normalising this using the standard of deviation, $K\frac{|f'_{n-1}(x^*_{n-1})-\mu(x^*_{n-1})|}{\sigma(x^*_{n-1})}$, improves the stability but reduces the best case result. By squaring this normalised version, $K\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{\sigma(x^*_{n-1})^2}$, the stability is further improved as well as the best case result. This leaves the question - what happens as we further increase the exponent? I tried squaring again. This gave a larger range of K that gives good results:

$$\beta_n = K\frac{(f'(x^*_{n-1}) - \mu(x^*_{n-1}))^4}{\sigma(x^*_{n-1})^4} \quad (46)$$

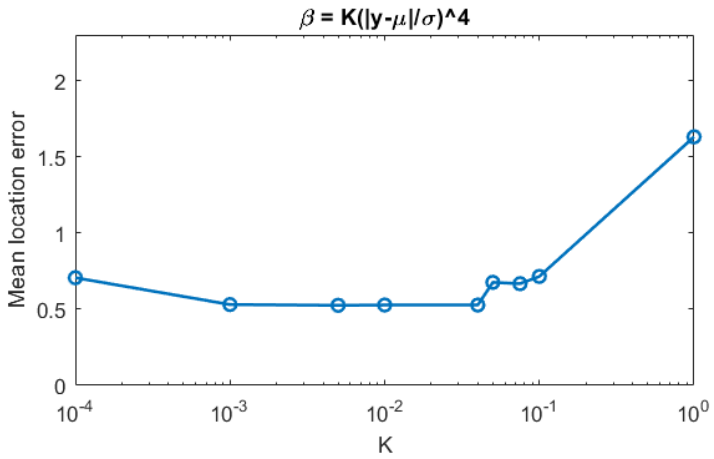


Figure 17: Results for different values for K. This shows good results for the widest range of K values. This will make tuning the value easy. The best result was a mean error in location (phase) of 0.52755254 at K = 0.01

This made the minimum over K shallower. This is a good change as it makes results less sensitive to the choice for K. This was very much an empirical observation. I believe the reason why this is an improvement is because it further discretises explore and exploit. By squaring the $\frac{I}{H}$ metric $\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^2}{\sigma(x^*_{n-1})^2}$, top-heavy ($> 1$) fractions become much larger and bottom heavy (< 1) fractions become much smaller. This is shown in Figure 18. This more extreme separation of values, based on

the fraction $\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))}{\sigma(x^*_{n-1})}$, allows only for pure exploitation (small outputs) or pure exploration (large outputs).

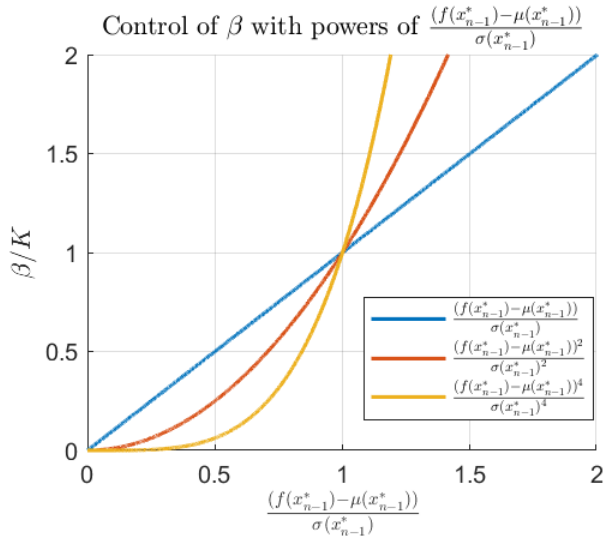


Figure 18: Increasing the power that $\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))}{\sigma(x^*_{n-1})}$ is raised to, pushes the result away from 1 allowing for discrete exploitation exploration with either very small or very large values for β.

If increasing the exponent is improving the stability, why not raise it to an extremely high power? As things are being squared it opens up the possibility of large numbers being made huge and causing overflow issues. Based on initial tests the improvements by raising to further higher powers diminish. Because of this, I will use $K\frac{(f'(x^*_{n-1})-\mu(x^*_{n-1}))^4}{\sigma(x^*_{n-1})^4}$ and not a higher power. I have also carefully limited the range of $\sigma(x^*_{n-1})\&(f'(x^*_{n-1})-\mu(x^*_{n-1}))$ that are fed into the function to prevent too large numbers being generated, preventing overflow issues.

4.4 Applying Control to Beta

The previous sections allow us to gain an understanding of how correct our chosen value for β and then to simply control β. A logical next step is to attempt to better control β. Linear control can be applied to updating β. The previous implementation of varying β is essentially a version of proportional control acting upon our feedback metric: $\frac{(f(x^*_{n-1})-\mu(x^*_{n-1}))^4}{\sigma^4}$.

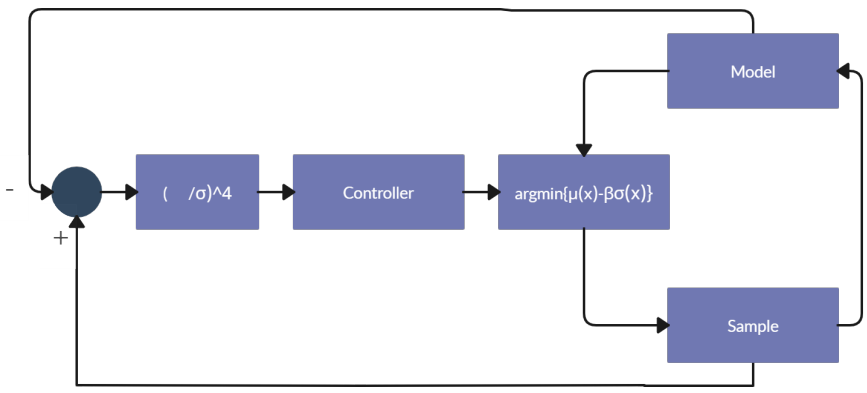


Figure 19: Control diagram showing how control can be applied to β. It shows the loop where the model is updated by the previous sample then the next sample is chosen using the UCB method which is fed a value of β by the controller. The controller receives information from the sample and the previous model.

4.4.1 PID

An good first step to further this method is to try to use PID control. By retaining previous values we can implement integral and derivative control.

$$d_n = f(x_n^*) - \mu(x_n^*) \tag{47}$$

$$\beta_n = max\{0,\ K_P \frac{d_{n-1}^4 - c}{\sigma^4} + K_I \frac{(d_{n-1}^4 - c) + \sum_{j=2}^{n}(d_{n-j}^4 - c)}{\sigma^4} + K_D \frac{d_{n-1}^4 - d_{n-2}^4}{\sigma^4})\} \tag{48}$$

C becomes another tuning variable as the choice of logarithmic base dictates its value. This does have a big affect on the integrator as if not carefully chosen will pin the integral limits. Integral windup is prevented by limiting the integral sum to sensible (design-specific) limits.

Initial tests with this did not produce good results. The conclusions I made were that the differential term isn't useful.I couldn't get the controller to work unless the differential term was 0. This is likely because it made the algorithm very noise responsive. This could potentially be dealt with by low pass filtering over multiple iteration cycles. The integral term seemed to help with noise but made the algorithm very slow to pick up a change to the objective function.

4.4.2 Exponential decay

The integral term has a low pass effect. This is good as it helped with noise rejection. It is bad because it made the controller slow to respond to changes in objective function. A slightly different strategy is to retain the influence of previous readings of difference in information but weight them, such that their influence on the current value of β decays.

$$\beta_n = K_P \sum_{j=1}^{n} \frac{K_{exp}^j (f(x_{n-j}^*) - \mu(x_{n-j}^*))^4}{\sigma(x_{n-j}^*)^4} \tag{49}$$

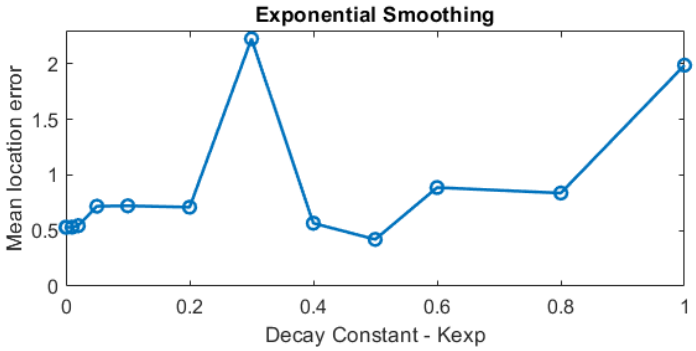


Figure 20: Results for different values for K_exp (decay constant) using $K_P = 0.001$ There is an optimal value around $K\_exp = 0.5$ that balances the benefits of noise rejection with over smoothing and reducing responsiveness.

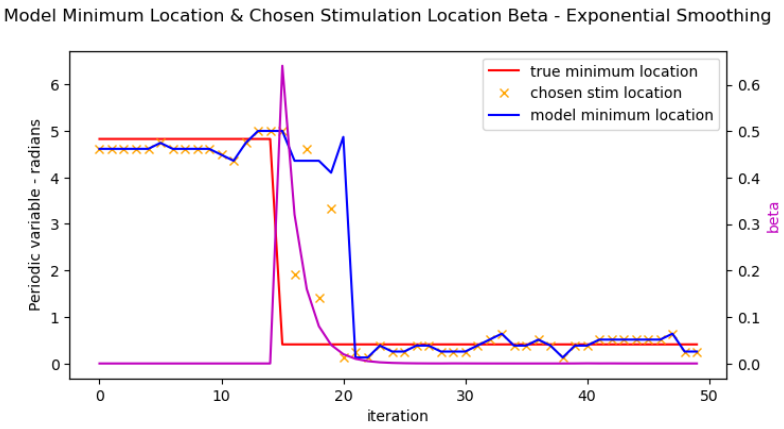


Figure 21: Tracking a sinusoidal objective function over the dummy periodic variable of phase. The objective function's minimum location has a step change (red). The GP model mean's minimum (blue), chosen sample location (orange). Here dynamic control for β was modified with exponential smoothing. This gives noise rejection and a smoother re-acquisition at sudden changes to the objective function.

The exponential smoothing keeps the exploration lasting longer and the correct K_{exp} will improve the result. Because of this, I included this into the algorithm. Initially, the error gets worse then better then worse again as K_{exp} is increased. The ideal amount is to have the decaying β pass swap to an exploit bias once the model means is predicting the correct minimum location. This isn't necessarily predictable so this hyperparameter will have to be tuned on a more realistic function as done in section 6.

5 Implementation

This section covers the practical elements of the design such as speed/memory concerns.

5.1 Remembered Stimulations

A key part of the algorithm is the fact that only N previous samples are used to feed the model. These samples are important to manage. In a time-invariant system, old results would still be relevant so the only reason to get rid of them is the computational cost of using more samples to evaluate the model. This effect still applies to a time-variant system, this is discussed further in the section 5.6.1. The summary of this is: for N samples that feed the model, the order of growth with respect to N is either $\mathcal{O}(N^2)$ for small N or $\mathcal{O}(\approx N^2 \log(N)$ for large N.

More importantly, in a time-variant system, N affects how dynamic the system can be. If the model uses 100 samples and then the objective function suddenly changes, then 100 samples will have to be replaced for the model to be using 100% relevant data. If the model only uses 10 samples, then it will be able to replace its old irrelevant data faster, making it more dynamic.

More samples will give a better representation of the parameter space. This will give a better model allowing for a more accurate decision to be made. If the objective function has a lot of detail this becomes more important. I am not expecting lots of detail for the objective function in this application when looking at the relations between phase and tremor severity[22]. More samples will also provide better noise rejection. If the model can afford to have multiple samples in one location it will get more knowledge

of the distribution of samples at that point. This will make the standard deviation, at that point in the parameter space, more representative of the sampling noise instead of the uncertainty in the model.

Balancing N between these factors very much depends on the real-world effects of noise and the rate of change of the objective function. I cannot predict these accurately, instead, I tested increasing levels of noise in response to a step change.

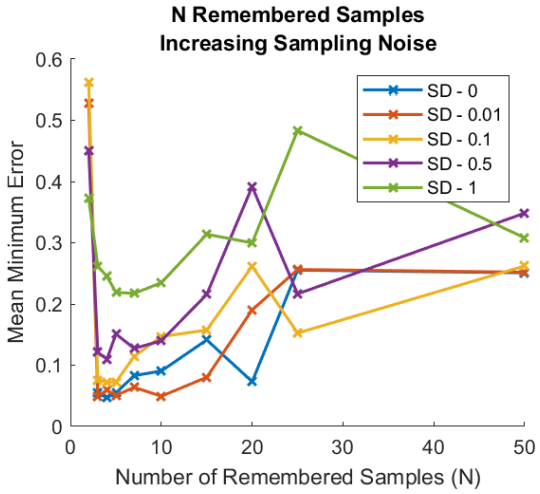


Figure 22: Mean error in output compared to the true minimum for different values for N (remembered samples) and for different levels of 0-mean addictive sampling noise.

Different noise levels had minimums at different N-remembered samples. The trend is that the minimum lay between N=3 and N=10. For most levels of noise, this was a shallow minimum with many values within that range of N giving similar results. Based on the results here I chose to use 7 remembered stimulations as this lay within the shallow minimum of all the noise levels. This also favours a higher sample noise. I decided this was the safest approach since the true noise levels are unknown. This is also suitably small for fast results based on the predicted time cost from section 5.6.1, order of growth should be 49M where M is the combined domain resolution of the model parameter space.

5.1.1 Choosing which samples to keep

Since the algorithm only retains a limited number of samples, some samples have to be removed in order for new ones to be included. Initially, I was going to just replace the oldest sample. The problem with this is as the model moves towards exploitation, the distribution of samples over the parameter space will bunch up over the model's minimum. This will leave large uncertainties elsewhere in the parameter space. To prevent this I have implemented a replacement decision that incorporates how close the new sample's parameters are to other remembered samples as well as the age of the samples.

Two options for this were:

- Each sample is given a metric evaluated as the weighted sum of the inverse of the distance to the new sample, and the age. Then pick the sample with the largest value. (it is important to normalise each parameter using the mean for each one, this prevents one parameter from blowing the decision out of proportion. A key feature/issue with this is that eventually all samples will be replaced as

the age outweighs the distance (which is limited by the parameter space which is confined to safe and feasible stimulation parameters)

- With respect to the new sample, the nearest m samples are considered, and then the eldest is replaced. This will not eventually replace all of the samples.

I ended up taking the first one as, by eventually replacing all the samples, stale regions in the parameter space will become sparsely populated. This means when stimulation location shifts towards exploration, the regions not recently explored will be prioritised.

5.2 Kernel Choice

The kernel is actually the covariance function for the Gaussian process. It is used in the Gaussian process model to evaluate the similarity of two samples. This determines the fit of the model to the samples. There are many choices for kernels to use [43]. The kernel can be chosen automatically using the method in Duvenaud et al[44]. However, if you have knowledge of your objective function, a choice can be made.

In the case of this project, we know the phase (which will definitely be one of the parameters) is by definition periodic over 2π. This means the periodic kernel will be useful. This is defined as:

$$k_p(x, x') = \sigma^2 \exp -\frac{2 \sin \frac{\pi|x-x'|}{p}}{l^2}$$

p defines the period - so in the case of phase will be 2π as phase repeats with this period. Choosing 2π as p does not prevent smaller period harmonics from appearing in the model but since 2π has to be the largest period this is chosen for p. l is the length scale and which determines how much detail the model will try to incorporate. σ should try to match the average distance the samples lie away from the mean. (captures the sampling noise) [45]

For our other variables (for example voltage, frequency, and pulse duration) we are not expecting linear, or periodic behaviours which makes the Squared Exponential Kernel the most appropriate choice.

$$k_{SE}(x, x') = \sigma^2 \exp -\frac{(x - x')^2}{2l^2}$$

l and σ have the same function as before.

Since we have multiple different parameters, choosing how the kernels are combined is important. Both adding or multiplying the kernels from each dimension is a valid route. I chose multiplication as I expect the effects of the different parameters to combine in this manner. My justification is that if you expect a negative or positive effect from the function over phase, the effects of pulse width or frequency would always accentuate or diminish the effect when using multiplication. When using addition, instead an increased effect for pulse width or frequency would become a decreased effect when the polarity of the phase effect swaps.

5.3 Safety Considerations

Stimulation parameters such as voltage and frequency have the potential to cause pain injury, or no effect if chosen incorrectly. It is an important design consideration to ensure that the controller does not produce unwanted stimulation parameters. IEC 60601 provides safety requirements for physiologic closed-loop controllers[46]. Key considerations from this are dealing with intra-patient and inter-patient variances, actuation limits and fallback modes if problems occur.

The variance between patients should be largely dealt with by how the controller builds a model from the patient-specific samples. Variances in the suitable stimulation parameters due to pain/discomfort will be dealt with by initially setting hard limits on the stimulation parameters in the clinic when the patient first interacts with the device. These limits could be changed manually at another time if needed.

Actuation limits are achieved by how the controller is designed. The algorithm such that it shouldn't produce any stimulation using parameters outside of the clinician's set limits. This is because the Gaussian process model, which the optimisation code is running off, is only evaluated over the defined parameter space (defined by the clinician-set limits) and the acquisition function can only select a candidate stimulation from within the model.

A fallback mode will be included when the controller is implemented into the device. This will ensure if the controller encounters a problem it will default to open-loop control at pre-defined characteristics. This will include a measure to prevent the controller from being stuck in an iteration. If the code does not complete an iteration within a certain amount of time then it automatically triggers a reset. If the code stops iterating repeatably working then the device will swap to the open-loop fallback mode.

In addition to this, if there is a safety or discomfort issue with the device, it will have an accessible button that immediately turns off the stimulation. The device could also be removed from the wrist.

5.4 Microcontroller Choice

Currently, the wrist-mounted device uses a Teensy 4.0 microprocessor. This is being used due to its good speed for its size. It is what I have implemented my code because this is what the device is using and will ensure that it works on the correct microcontroller and makes speed tests relevant.

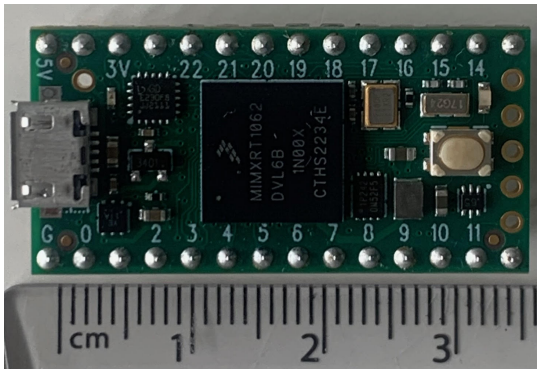


Figure 23: Teensy 4.0 board with scale.

5.5 C++ and Code

5.5.1 Libraries Used

The Bayesian optimisation method used relies heavily on linear algebra. This is not a feature available in basic C++ syntax. Instead, the library Eigen [47] is used in the code - specifically, Eigen version 3.4.0. It provides fast and reliable linear algebra functions and structures. When vectors and matrices are mentioned in this section it refers to the VectorXd and MatrixXd classes.

5.5.2 Code structure

The structure of the program is in 3 parts, a .cpp and a .hpp file form a library-like set of functions. These are called from a top level .cpp file which follows the same structure as the pseudo algorithm below. This design makes for easy troubleshooting and development as separate functions can be treated as separable modules. Clarity is important for the top level as this will be integrated into a separately developed program for the device.

Algorithm 2 Top Level Controller Algorithm

Define the stimulation parameter space with domain and domain resolution.
Take initial N randomly distributed samples across parameter space to fill remembered stimulations.
repeat
  Fit Gaussian Process Model to remembered stimulations.
  Apply acquisition function, selecting the next set of parameters
  Sample with selected parameters
  Evaluate how good the prediction/decision was and choose the next β
  Choose replacement in remembered stimulations (in memory)
  Save the new stimulation into the device storage.
  Update remembered stimulations (memory)
  (Replace chosen stimulation with the new one
  Increase the age of all samples)
until

This relies on a few structures that store groups of information:

- Memory structure - This is outlined in table 1. Practically, this is stored as a C++ structure: a vector of ages, a matrix of stimulation parameters, and a vector of tremor securities. Row corresponds to which sample and the columns in the matrix correspond to different stimulation parameters. By using the three different sub-structures, each can be independently accessed and used when building the model and choosing which stimulation to replace.

Table 1: Memory structure for stimulation samples

Age	Stim parameter 1 (x)	Stim parameter 2 (x)	Stim parameter 3 (x)	.... (x)	Tremor severity output (y)
1	2.1	0.5	10.0		-0.4
5	4	0.2	14.3		0.1
3	3.3	0.5	14.3		0.5
...	...	...	...		...

- Model Structure - This stores the most current Gaussian process model and consists of:
  - domain - A matrix evenly dividing the defined domain of the stimulation parameter space. Columns correspond to parameters, each row consists of a location in the parameter space. The matrix will have M columns: $M = m_1 m_2 m_3 ...$ Where m_n corresponds to the resolution given to the n^{th} stimulation parameter. In section 5.6 it is shown that M has a big effect on the speed of the algorithm.
  - mu - a vector of the Gaussian process model's mean over the parameter locations defined in the domain matrix. Length M.
  - sd - a vector of the Gaussian process model's standard of deviation over the parameter locations defined in the domain matrix. Also length M.
- Target Stimulation Structure - This stores the intended or most recent sample's stimulation parameters (stored as a vector) and what the previous model predicted for them (the model mean and standard of deviation at the chosen stimulation parameters stored as two floats). This is referenced when performing the stimulation and also to retrospectively check how accurate the model was when selecting β for the next acquisition function.

5.6 Optimisation

As mentioned earlier 3.9 a key design criterion is optimising speed and memory usage. Speed can be broken down into the order of growth of time for different parts of the code and the speed of individual 1-line functions being performed. Memory allocation is then discussed.

5.6.1 Order of growth

The algorithm has many different loops within the code. Each time a loop is iterated it comes with a time cost associated with the functions being run inside it. The time cost of a loop that runs a times can be represented without knowing the individual time cost of what basic functions are performed inside of it. This is done by attributing an order of growth to it: $\mathcal{O}(a)$. This says that the time taken to perform the loop grows linearly with a. If loops are nested: say a loop is run a times within a loop that runs a times this grows with: $\mathcal{O}(a^2)$. This principle can be applied to the algorithm to establish which single line functions affect the speed the most (are executed the most times) and which variables have the greatest order of growth (so to optimise speed, the order of growth should be kept small).

The variables that affect loop size are: M is defined as the combined domain resolution. n is the number of remembered stimulations.

Algorithm 3 Order of growth

Initialisation

1

- Define domain and domain resolution of the stimulation parameters
- Take initial N randomly distributed samples across parameter space to fill remembered stimulations.

repeat

- Fit Gaussian Process Model to remembered stimulations.
  1
  - Calculate covariance matrix $\leftarrow \mathcal{O}(n^2)$
  - Invert matrix $\leftarrow \mathcal{O}(n^2 \log n)$
  - m
    - Calculate kernel values $\leftarrow \mathcal{O}(n+1)$
    - $\mu(i) = \bar{n}\overline{n \times n}\bar{n} \leftarrow \mathcal{O}(n^2+n)$
    - $\sigma(i) = 1 - \bar{n}\overline{n \times n}\bar{n} \leftarrow \mathcal{O}(n^2+n)$
- Apply acquisition function, selecting the next set of parameters $\leftarrow \mathcal{O}(m)$
- Sample with selected parameters
- Evaluate how good the prediction/decision was and choose the next Beta
- Choose replacement in remembered stimulations (in memory) $\leftarrow \mathcal{O}(n)$
- Save the new stimulation into the device storage.
- Update remembered stimulations (memory)
- (Replace chosen stimulation with the new one
- Increase the age of all samples) $\leftarrow \mathcal{O}(n)$

The largest orders of growth are: calculating $\mu(x)\&\sigma(x) \rightarrow \mathcal{O}(Mnn)$, inverting the Covariance matrix $\mathcal{O}(nnlog(n))$. $M > n$ since n is kept relatively small to allow for a fast response to a changing objective function and M is kept relatively large to give a good resolution over the parameter space. This means $Mnn > nnn > nnlog(n)$. So evaluating $\mu(x)\&\sigma(x)$ becomes the slowest evaluation. Before evaluating the order of growth explicitly, inverting the matrix as part of the Gaussian process regression was expected to be the largest time cost with an order of growth somewhere around $\mathcal{O}(nnn)$ to $\mathcal{O}(nnlog(n))$. Because of this M, the combined domain resolution, is a major influence on the speed of the algorithm and a trade-off of speed against the resolution of stimulation parameter choice must be considered. n is also important, as expected. For a small n: good speed performance and response to a changing objective function. For a big n: good noise rejection. The order of growth $\mathcal{O}(Mnn)$ is represented in the real-time performance of the algorithm when implemented and tested on the Teensy 4.0 board as seen in Figure 24.

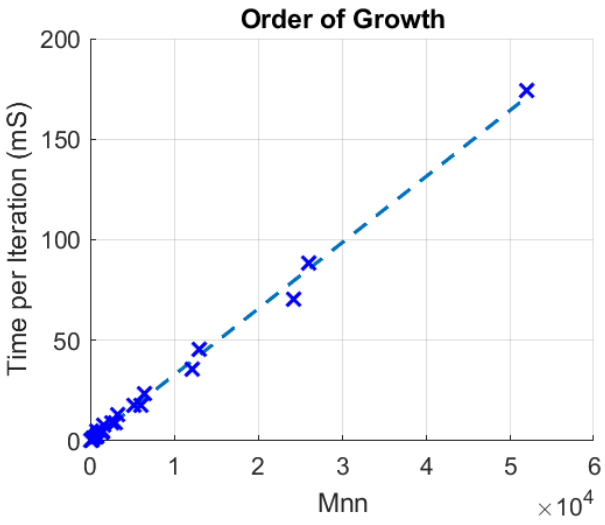


Figure 24: Time cost of one top-level iteration with increasing Mnn (averaged from 100 iterations). This was obtained without significant time cost for obtaining a stimulation result as it was calculated with a dummy function. This shows a linear increase in run-time per iteration with Mnn, experimentally proving $\mathcal{O}(Mnn)$.

6 Simulation Testing

So far the algorithm was optimised using abstracted functions. Before this algorithm goes near a real-world test it needs to be validated in a virtual simulation that approximates what it would see when implemented. I will also use this to establish the speed performance of the device. In section 6.1, I build an objective function using this data and other research. I will then test how the algorithm deals with time-changing objective functions in sections6.3,7.

6.1 Building an Objective Function from Real Data

A dynamic objective function needs to be designed between input parameters and output. For our device, 3 inputs make sense to optimise over Phase, Frequency, and Pulse Width 3.5. For each of these, I looked for research that characterises the objective function for the parameter. This research does not directly translate onto the intended device, so the testing objective functions built here cannot be expected to match what will be observed in real-life tests. Instead, they should validate that the controller can deal with the type of situations that may be observed.

6.1.1 Objective Function for Phase

Initial work from Cagnan et al.[22] on phase-locked thalamus stimulation gave the results shown in figure 25. This is for DBS not peripheral stimulation so using this only transfers onto peripheral stimulation loosely. The metric "change in tremor severity" used in this paper is the same as that used by the device [9]. This is: $\frac{T\_stim - T\_nostim}{T\_nostim}$ Where T is tremor severity. This means 0 change in tremor severity is no change in tremor severity, -1 is the total elimination of tremor, and $+1$ is double tremor severity.

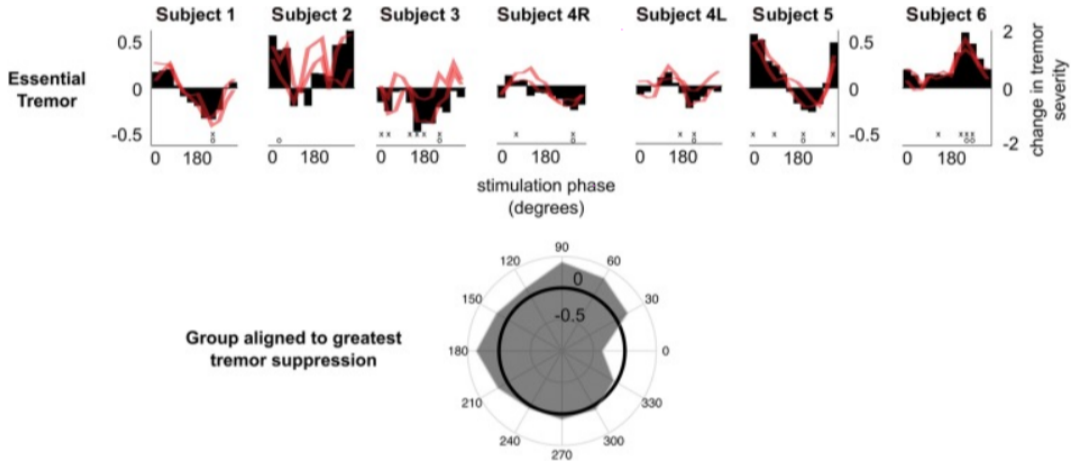


Figure 25: Cagnan, Hayriye et al. "Stimulating at the right time: phase-specific deep brain stimulation." Brain : a journal of neurology vol. 140,1 (2017) The results for ET severity modulation by phase locked stimulation to the thalamus. This is useful for the waveform shape of tremor modulation over phase.

Things to capture in the objective function:

- A positive and negative effect on tremor severity.
- Periodic with $2\pi Radians$.
- Finer detail than a $2\pi Radians$ periodic sinusoid.
- Magnitude of effect can change.
- Waveform not consistent in shape.

The variable objective function I created for testing. The parameters a,b, and m are varied to give different functions. Figure 26 shows some randomly generated examples.

$$y = m(a\sin x + b + (1-a)\sin 2x + b) \quad (50)$$

$$0 < m < 1, 0 < a < 1, 0 < b < 6.28$$

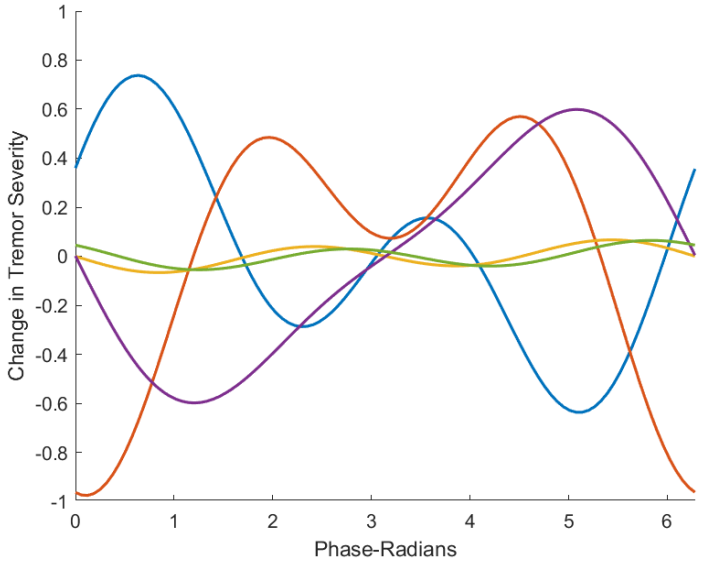


Figure 26: Randomly generated objective functions over phase using the model objective function. These are similar to those found in phase-locked deep brain stimulation [22]. The output is "change in tremor severity" used in Cagnan et al. and used by the device [9].

This function is periodic over 2π. It can have false minimums which provide a good test for the algorithm. Depending on the parameters it uses any phase can give a positive or negative result. It can also give a very shallow function that represents no significant result which is important as this is what was observed in early use of phase-locked peripheral stimulation for ET [9]. It can also give a flat distribution (no significant effects). This is important as this is what happened in an early look into the effects of phase-locked peripheral stimulation for ET [9].

6.1.2 Objective Function for Frequency

Kim et al. investigated the effects of stimulation frequency on Tremor [48]. This was a study for ET using close-loop and open-loop, phase-locked peripheral stimulation over the radial nerve. The results are shown in Figure 27. The metric "normalized power" used in this paper is different to that used by the device [9]. It used: $\frac{T\_stim}{T\_nostim}$ where T is tremor severity. This means 0 normalized power is the total elimination of tremor, 1 is no change in tremor severity and > 1 is an increase in tremor severity. This is similar to the metric "change in tremor severity" used before but with a shift of $+1$.

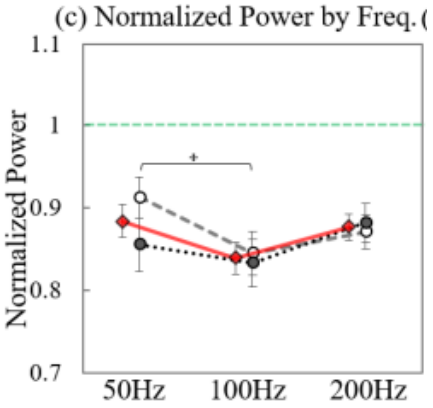


Figure 27: Source: A wearable system for attenuating ET based on peripheral nerve stimulation— Jeonghee Kim and Thomas Wichmann and Omer T. Inan and Stephen P. Deweerth. This shows that there is an optimal simulation frequency somewhere within the range of 50Hz-200Hz and that either side of this the effect will be reduced.

The most effective frequency was 100Hz with the effect diminishing as frequency was increased and decreased. The metric used was also not as rigorous as what is used by the device this controller is intended for and the scale of effect was not huge, but the trend over frequency is still useful for the purpose of a test function.

Things to capture in the objective function:

- Always has a reduction in tremor.
- One minimum
- Low detail (although few data points)

The objective function must also be combined with the objective function for the phase. In the method used the phase lock was not varied and gave only a reduction in tremor [48]. Therefore, the mutual relationship cannot be estimated. Instead, I assumed that the stimulation frequency will modulate the magnitude of the phase lock effect. This is because of how pulse width and frequency combine in transcutaneous electrical nerve stimulation (TENS) to induce local nerve conduction [49]. For this

modulation in amplitude of effect to be incorporated, the objective function over frequency becomes a function for stimulation effect modulation. +1 is the maximum potential from the phase lock effect, 0 is no stimulation effect. This is multiplied with the objective function for phase to give a combined objective function for change in tremor severity. Figure 28 shows some randomly generated examples of this function.

$$1 + a + b(x - s)^2 \quad (51)$$

$$-0.2 < a < 0$$

$$0.000005 < b < 0.00005$$

$$50 < s < 200$$

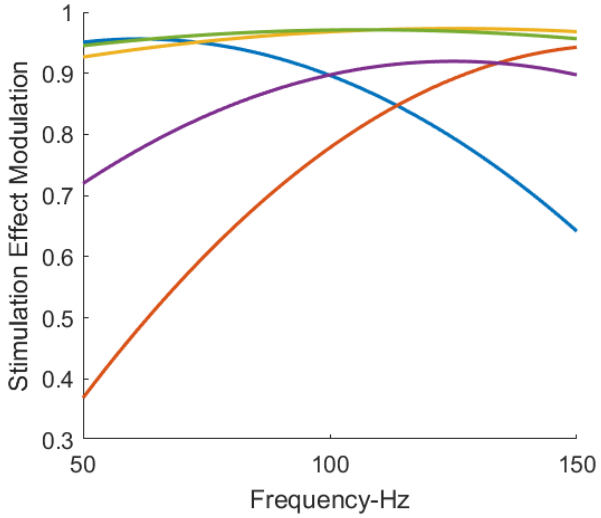


Figure 28: Randomly generated objective functions over frequency. There exists a frequency which gives maximum stimulation effect modulation within the range 50 - 200 Hz. This objective function is multiplied with the objective function for the phase to get a change in tremor severity.

6.1.3 Objective function for Pulse Width

I couldn't find research to give the relationship between pulse width and tremor severity that was related enough to the device to create an objective function. Instead, I used research on the effects of pulse width on nerve conduction for TENS [49]. This suggests that pulse duration and frequency both cause increased nerve conduction as they increase and that the effects somewhat combine. Because of this, I expect that the effect of pulse width will have a similar effect to frequency. In a summary of research on peripheral stimulation for pathological tremor, the range of pulse width used when testing the effects of afferent signals on tremor was: $150 - 400\mu s$ [50]. I have made my objective function for pulse width take the same form as the one for frequency but over the range $150 - 400\mu s$. As it will be multiplied with the objective function for phase in the same way as frequency, it also will be a function for stimulation effect modulation. Figure 29 shows some randomly generated examples of this function.

$$1 + a + b(x - s)^2 \quad (52)$$

$$-0.2 < a < 0$$

$$0.000001 < b < 0.00001$$

$$150 < s < 400$$

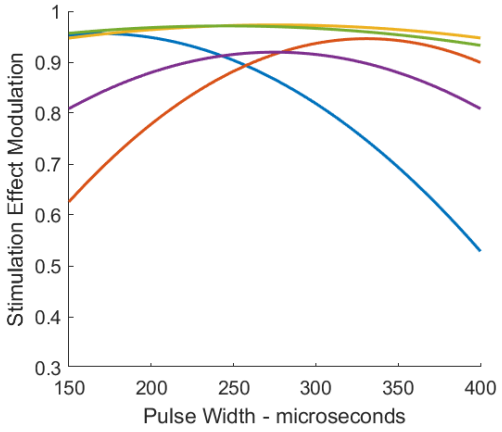


Figure 29: Randomly generated objective functions over pulse width. There exists a pulse width which gives maximum stimulation effect modulation within the range 150-400μs. This objective function is multiplied with the objective function for phase to get change in tremor severity.

These three functions were combined to give the total objective function:

$$\begin{aligned} y_{tremor} = m_{phase}(a_{phase}\sin x + b_{phase} + (1 - a_{phase})\sin 2x + b_{phase})(1+ \\ a_{freq} + b_{freq}(x - s_{freq})^2)(1 + a_{pw} + b_{pw}(x - s_{pw})^2) \end{aligned} \quad (53)$$

An important consideration is that the research used for the objective function was only for ET and not for the specific device. It cannot be assumed that this will carry over to the device or to PD patients. This, therefore, cannot prove that the algorithm will have this effect on real patients (ET & PD). Instead, this should be taken as a test against the type of objective function that would be expected. From the patient tests of the device that have been/are being conducted (late 2022-early 2023) another objective function should be constructed to check the algorithm against to give hopefully transferable performance metrics.

6.2 Constraints From the Device

Some additional constraints to the algorithm must be included. The device cannot provide a continuous set of parameter options. Instead, the algorithm must be given the appropriate set of stimulation parameters. This is done with a range and a resolution for each parameter:

- The stimulation takes one of 6 phase settings - this gives a domain resolution in the phase dimension of 6 over the range $0 - 2\pi$.
- It can stimulate at different 3 frequencies (multiples of 52) giving a domain resolution of 3 and range of 52-156 for frequency.
- Pulse-width takes spacing of $50\mu s$. For the range of $150 - 400\mu s$ (Based on what is used in research up until now [50]), This gives a domain resolution of 6.

This fixes the domain resolution to: $M = 6 * 3 * 6 = 108$. This has impacts on speed as shown in section

5.6.1.

6.3 Test Design

There are two situations I wanted to test the algorithm for: A) A drift in objective function representing slow changes on the system like time of day. B) A sudden change to an objective function representing fast changes such as limb and device positioning.

6.3.1 Drift Test

The aim of this test is to measure performance for continuous changes to the objective function. This is done with a random step each iteration with 0 mean normal distribution. The standard deviation gives the drift rate. This drift rate is varied to give effects of different timescales. Ten seeds (1-10) were reserved for hyperparameter testing. Ten seeds (11-20) were to be used for testing (with 20+ also available if needed) For each test seed, the drift rate was increased using: SD of step size = [0.000001,0.00001,0.0001,0.001,0.00316,0.01,0.0316,0.1] to cover effects of timescales between years and minutes. The test was over 50 iterations. This affects the objective function as shown in Figure 33. Additive noise for each sample taken from the objective function was 0 mean, SD = 0.1 normally distributed.

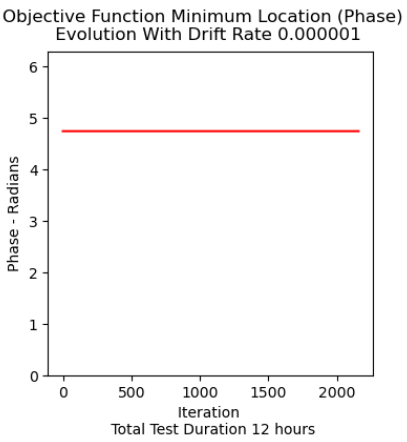


Figure 30: Drift rate significant over 1 year time scale

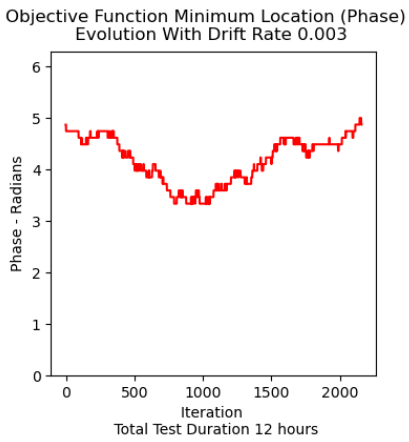


Figure 31: Drift rate significant over 24 hour time scale

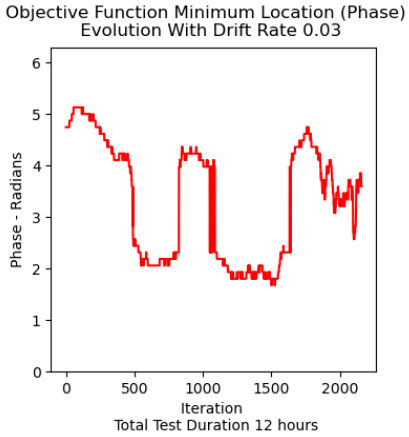


Figure 32: Drift rate significant over 4 hours approximately the time scale for medication or meals

Figure 33: Objective function changes as drift rate is increased.

The benchmark the algorithm was tested against was manual parameter selection. This was done by manually picking the best input parameters for the initial objective function and keeping them constant for the 50 iterations. I decided to measure the output, not the input location, as a success metric. This is because when the objective function has multiple minimums with similar outputs, the location could be very wrong but output could still be quite correct.

6.3.2 Step Test

The ability to re-acquire the minimum after a sudden change to the objective function is very important in dealing with sudden changes to the system. To test this, first a random objective function is generated using a uniform distribution over the given objective function parameter ranges. This is kept constant for 25 iterations. A new random objective function is then generated and kept constant for the last 25 iterations. This gives an opportunity for the algorithm to settle twice. Again, ten seeds (1-10) were reserved for hyper-parameter testing, and ten seeds (11-20) were to be used for testing (with 20+ also available if needed). The algrothm will be compared against the results for fixed paramter. It's settling time will also be compared to ET-GP-UCB which would take N (7 in this case) iterations to reset the GP model.

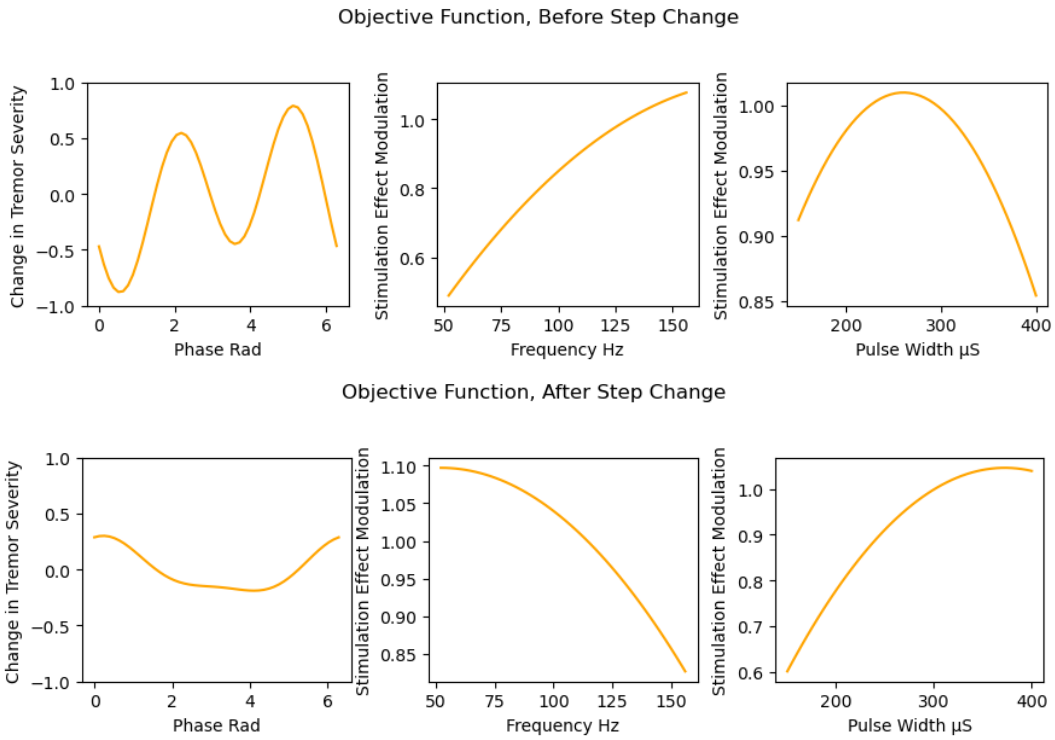


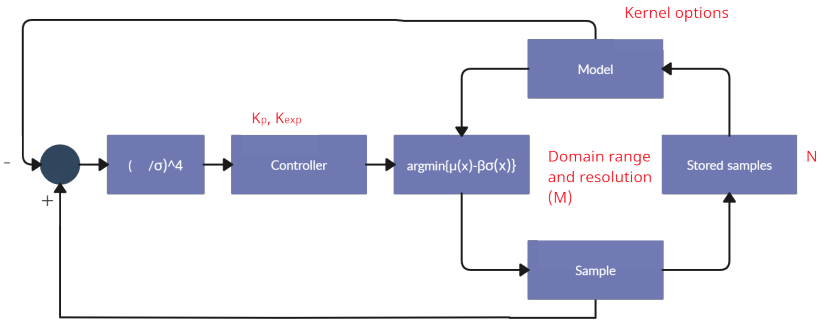
Figure 34: Examples of the objective function before and after the step.

6.4 Tuning Hyperparamters

Before testing, I used seeds 1-5 with random drift, SD = [0.05,0.2] to iteratively tune the hyperparameters that were not fixed in the Design and Implementation sections. With new ranges for the input parameters, different kernel parameters are needed. Kernel choice didn't need to change. The control parameters also needed tuning. I kept using 7 remembered stimulations and the same decay constant as I believe these are linked and re-choosing them was not necessary.

First I worked through the kernel parameters and ended up with: l = [0.6, 25, 75] I chose these as sensible sub-divisions of their respective parameter spaces. I chose the kernel variables σ = [3,3,3] as this worked well. After this was adequate I then varied the Kp constant from 4.4.2, the best value was 0.009.

Figure 35: Looking back over the design model parameters need to be selected for the specific application. The key parameter values in this are: The controller constants: K_P, K_{exp}, The domain ranges and resolutions that define the parameter space, The number of previous samples that feed the model: N, and the hyper-parameters that are part of the kernel.



This parameter optimisation was not done in great detail and I'm sure with greater time this optimisation could be done better, but would be pointless as it would be repeating what was done in the design stage and, as the testing objective functions are unlikely to perfectly match the device, further work is not meaningful. This should give a lower bound on performance.

7 Results

7.0.1 Drift Test

The results for the drift test are shown in Figure 36. The controller algorithm outperformed the manual parameter selection over drift rates faster than 0.001. This approximately corresponds to signifigant effects that occur faster than in 24 hours. For slower drift rates, the controller is outperformed by the fixed parameter method.

The worst case results from the 10 test seeds are better for the controller than the fixed parameter method. For drift rates slower than a 24 hour time scale, the worst case for both methods is approximately 0 effect. The worst case for the fixed parameters for rates faster than this are worse than 0 effect. This is where the fixed parameters are now causing an increase in tremor severity. The algorithm's worst case stays below 0 effect. This significant as this means the controller is less likely to cause troublesome effects as a result to a change to the system.

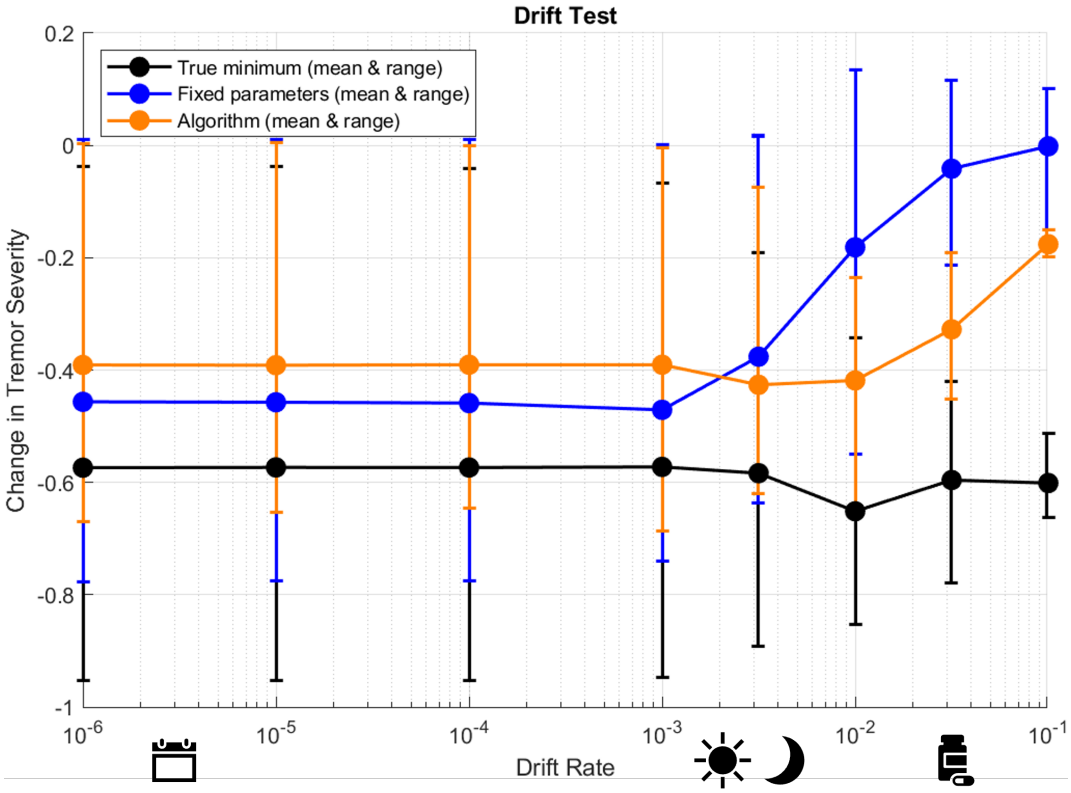


Figure 36: The results for increasing objective function drift rate. The mean(dots) and range (error bars) of the 10 seeds are plotted for the average effect over 100 iterations. In black is the true minimum for the objective functions. In blue is the reference method of stimulation parameters that have been perfectly manually selected at iteration 0 and then held constant despite any changes to the objective function. This represents clinician-set stimulation parameters. In orange is the algorithm designed in this project.

7.0.2 Step test

The results of the step test are shown in figure 37.

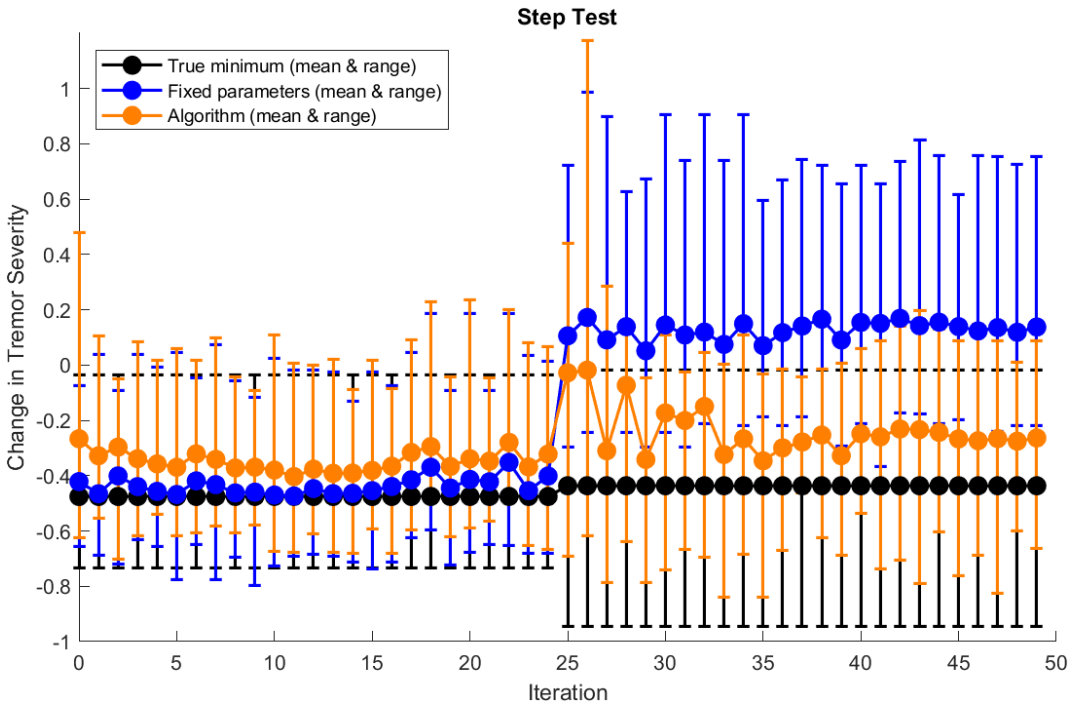


Figure 37: The results for a step change in the objective function. The mean(dots) and range (error bars) of the 10 seeds are plotted, In black is the true average minimum for the objective functions. In blue is the reference method of stimulation parameters that have been perfectly manually selected at iteration 0 and then held constant despite any changes to the objective function. This represents clinician-set stimulation parameters. In orange is the algorithm designed in this project.

The fixed parameters outperform the algorithm before the step change, although not by a large amount. After the step change, the algorithm loses the minimum but quickly re-acquires it, although not quite with the same accuracy as before. This is largely due to a few false minimums the algorithm has locked onto. This is particularly likely if the new false minimum lies in a close location to the previous true minimum. The fixed parameter results after the step change are particularly bad as they give a mean greater than 0. The worst-case results for the fixed parameters after the step change to the objective function are also far worse than those for the algorithm, giving significant worsening in tremor severity.

Looking at the individual test results (fig 38) we can establish the settling time:

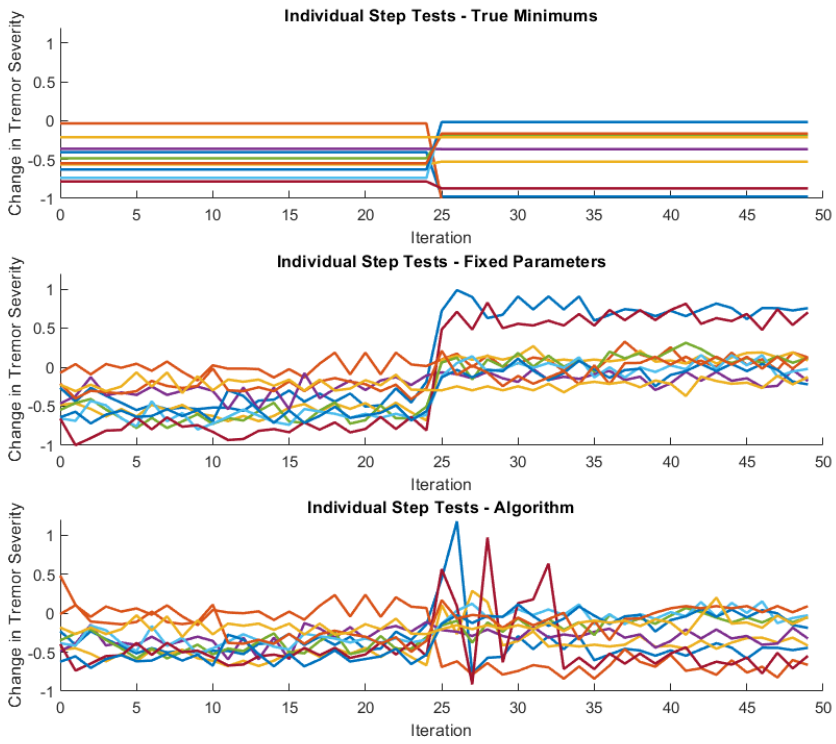


Figure 38: The results for a step change in the objective function. This includes the actual results for the algorithm for all 10 seeds. From this, the settling time can be estimated for the re-acquisition of the new best parameters.

Settling time was measured as iterations from after the 25th iteration until it stopped changing by more than 0.2 for at least the following 5 iterations. The mean over the 10 tests was **3.5 iterations**. The reason in the previous graph 37 it looks longer is largely due to Seed 17 where it takes much longer, although does still settle by 9 iterations. This average is significantly less than 7 which is what ET-GP-UCB would take to reset the GP model fully.

7.1 Speed Test

A significant part of the design of this controller is that it must be implemented on a real-world device. It must be able to select the next stimulation parameters between stimulations for the most dynamic control possible. I tested how successful this aspect of the device was with speed tests on the Teensy 4.0 board as this is the intended board used on the device.

I tested the iteration speed by running 10 tests of 100 iterations on two different Teensy 4.0 boards and took the average run-time for the iterations. This came out to be: 2.28mS per iteration.

8 Discussion

The results from the drift test show that the algorithm performs worse than fixed parameters when the objective function remains roughly stationary after the initial parameter selection. This is not hugely surprising as for the fixed parameters, the parameter selection was done manually to be as perfect as possible.

Since the test was the equivalent of 12 hours worth of iterations, the swap in performance shows that the algorithm does track changes and this improves the result. If the test could have been run for longer an earlier decrease in relative performance from the fixed parameters may have been observed. Unfortunately, the computer cost of running the high iteration tests was too much to perform within the scope of the project. For reference, 24 hours would require 4320 iterations and a year 1578000 iterations. With the data processing included, each run of 12 hours (2360 iterations) took about 1 minute to get results. Each point in figure 36 takes 10 runs so each data point it takes roughly 10 minutes to obtain. Longer tests would have required too much time to run.

When considering only the slow (month/year-long) objective function changes, a device with manual parameter selection which is retrained periodically could perform better than the algorithm. However it would be impractical to manually (in clinic) reset the fixed parameters at very regular intervals, hence the performance of the algorithm on shorter time scales (24 hour or less) is important. If the real-world short-term changes to the objective function are insignificant then the algorithm would be better replaced with periodic manual resetting (e.g., regular clinician tuning), or automatic resetting (e.g., Bayesian optimisation is run for an hour every month).

In the drift test, the fixed parameters outperform the fixed parameters up to a drift rate between 0.001&0.003. This is approximately the drift rate that corresponds to diurnal variations. This is good as if the algorithm was designed with these changes in mind.

Drugs are dosed throughout the day so the time scale of variation due to pharmacological treatment will be of the order of hours. This time scale is where the algorithm outperforms the fixed parameter method the most. For the algorithm, this is on the edge of when the variation is so fast that it can not exploit having adjusted to fit. For any faster drift (continuous drift at a speed of full 2π cycle every 10 minutes $\sim$ 0.1) the change in tremor severity is reduced significantly from the apparent steady-state presented by the very slow drift scales. This was expected as the algorithm was designed to swap to explore when a change is detected and then swap back to exploitation when the model is correct. If the objective function never settles then the algorithm can never exploit the model so performance will approach random sampling of the parameter space.

The results for the step test match the results from the drift test. In the initial period of stationary for the objective function, the fixed parameter method outperforms the algorithm. The sharp change to the objective function initially throws both the algorithm and the fixed parameter method by similar amounts. This matches results from the very high levels of drift from the previous test, both methods perform badly for large rapid changes to the objective function. The algorithm does, however, re-converge onto an optimal set of parameters, exploiting the new period of stability. This is what the algorithm was designed for. Compared to the fixed parameter method which remains poorly performing after the change to the objective function, it performs very well.

The re-convergence is fast - an average of 3.5 iterations. This, with a 20-second iteration cycle, corresponds to 1 minute 10 seconds. Importantly, this is faster than if Bayesian optimisation was just re-run from

scratch after the change is detected such as in ET-GP-UCB(this would be 7 iterations as the algorithm uses 7 samples to feed the model).

As shown by the step test, when the objective function is stationary the algorithm will settle onto a new, close-to-optimal parameter set. This algorithm is, therefore, very good for periods of slow change/no change with brief times of change. Translating into real-life situations, step-changes to the objective function may occur. There will be periods of low environmental change (e.g., patient sitting still) and periods of large environmental change (e.g., standing up and walking out of the house). The algorithm may not achieve good tremor suppression in the period of change but will quickly settle to good tremor suppression once the objective function becomes approximately stationary again.

Arguably the results from the step test are better than the continual drift test as the algorithm outperforms the fixed parameter method by a greater margin. This is because the algorithm was designed to deal with periods of change and stationarity separately. The design solution was to discretise exploration and exploitation. The trigger for exploitation cannot be too low as then it would be triggered by noisy readings. This leads to small changes (as caused by drift) being unable to trigger the algorithm to re-explore. Only when the drift has caused a significant change to the objective function, the algorithm re-explores to correct the model. When the noise and rates of change to the objective function are known more accurately what it takes to trigger exploration can be optimised.

8.0.1 Speed Test Discussion

The average iteration run time, when implemented on the Teensy 4.0 microcontroller was 2.28mS. This is very fast in comparison to the downtime of the device between stimulation bursts (10 seconds off in between stimulation sessions of 10 seconds). This means the code can comfortably run each cycle and update the stimulation parameters in real-time. This fast speed is a product of the low number of remembered stimulations that feed the model (7) and the low domain resolution used by the device (6x3x6). This means the computational cost of building the model is actually very low.

Potentially this is too fast. As there is a lot more dead time, this could potentially be used for a more complex control algorithm. An example is if the domain resolution for the stimulation parameters (M) could be increased, from every 60 degrees to every 3 degrees. This could allow the algorithm to stimulate closer to the true optimal stimulation parameters. As discussed in section 5.6.1, this would increase the run-time of the algorithm, but this would be acceptable because of the amount of spare time.

The speed that this controller runs at opens up the potential for the controller to be run on a faster scale. PDT occurs at approximately 4-5Hz [10], corresponding to a period of roughly 200ms. Each iteration is $\sim 1\%$ of this which means the decision for the stimulation parameters could be made between each oscillation. This would change the objective function and likely increase the noise on each sample so the results from section 7 would not hold, but this could be explored further.

9 Conclusion

To summarise, in this project I have made a controller, using a new Bayesian optimisation method based on TV-GP-UCB with dynamic exploration/exploitation bias, that tracks the optimal inputs for a time-varying unknown function subject to additive, zero-mean, Gaussian noise. This was achieved by making the exploration bias a function of the information gained by the previous sample and the entropy of the previous Gaussian process model. The controller was designed for computational efficiency, which enables the controller to be run on the intended microcontroller with a run-time suitable for real-time control.

In simulated tests, it outperformed manually tuned fixed parameter methods for step changes and continuous drift changes to the objective function that may be relevant to real-world variations resulting from circadian rhythms and the use of medications. The changes to the objective function over longer timeframes such as encountered in disease progression are so gradual that the advantage of having a responsive controller is outweighed by the effect of having better, manually selected, parameters. The controller was also able to acquire new parameters by an average of 3.5 iterations after the step change to the objective function, faster than possible with a hard-resetting Bayesian optimisation method such as ET-GP-UCB.

9.1 Further Work

I'd like to see this controller implemented into the device as part of some real-world trials. The first step to this is packaging the controller such that it can be called as a library by the main device program. This is mostly a matter of removing the testing parts of the program. Before this, it would be great to test the controller on longer timescale drifts. As discussed in section 8, this was impossible due to computing requirements/available time. These longer time-scale experiments could be run faster with more efficient computing.

I would also like to further explore the new time-varying Bayesian optimisation technique as I believe this is novel and could be useful for other applications. One further idea I have had for this is using a comparison of the distribution of samples taken against the model. Instead of just obtaining one output for the sample multiple can be taken. These should give a distribution of outputs. This can be compared better to the model-predicted distribution of outputs, than a single sample. It should give the natural variance in the output and allow the model to account for it.

As discussed in section 7.1, the algorithm is fast enough to run between phase-locked stimulations. Implementing this would require changes to how the device runs as it currently calculates its success metric, and change in tremor severity, over 20-second intervals. It would be interesting to explore other success metrics that could incorporate this fast control.

The algorithm would work just as well for other devices which have a time-varying function between multiple inputs and one output. The controller could be implemented in other devices such as for DBS and also for other diseases.

References

[1] Siwei Zhao, Abijeet Singh Mehta, and Min Zhao. "Biomedical applications of electrical stimulation". In: *Cellular and Molecular Life Sciences* 77 (14 July 2020), pp. 2681–2699. ISSN: 14209071. DOI: 10.1007/s00018-019-03446-1.

[2] Anne Beuter and Konstantinon Vasilakos. "Tremor: Is Parkinson's disease a dynamical disease?" In: *Chaos* 5 (1 1995), pp. 35–42. ISSN: 10541500. DOI: 10.1063/1.166082.

[3] Jan Raethjen et al. "Provocation of Parkinsonian tremor". In: *Movement Disorders* 23 (7 May 2008), pp. 1019–1023. ISSN: 08853185. DOI: 10.1002/mds.22014.

[4] Theresa A Zesiewicz, Elmyra Encarnacion, and Robert A Hauser. "Management of Essential Tremor". In: *Current Neurology and Neuorscience Reports* 2 (2002), pp. 324–330. ISSN: 1528-4042.

[5] Simon Hanslmayr and Frederic Roux. "Human Memory: Brain-State-Dependent Effects of Stimulation". In: *Current Biology* 27.10 (2017), R385–R387. ISSN: 0960-9822. DOI: https://doi.org/10.1016/j.cub.2017.03.079. URL: https://www.sciencedirect.com/science/article/pii/S0960982217304098.

[6] Laura J. Arendsen et al. "Peripheral Electrical Stimulation Modulates Cortical Beta-Band Activity". In: *Frontiers in Neuroscience* 15 (2021). ISSN: 1662-453X. DOI: 10.3389/fnins.2021.632234.

[7] John E Fleming et al. "iScience Embedding digital chronotherapy into bioelectronic medicines". In: *ISCIENCE* 25 (2022), p. 104028. DOI: 10.1016/j.isci. URL: https://doi.org/10.1016/j.isci..

[8] Ilija Bogunovic, Jonathan Scarlett, and Volkan Cevher. *Time-Varying Gaussian Process Bandit Optimization.*

[9] Carolina Reis et al. "Essential tremor amplitude modulation by median nerve stimulation". In: *Scientific Reports* 11 (1 Dec. 2021). ISSN: 20452322. DOI: 10.1038/s41598-021-96660-6.

[10] G. Deuschl et al. "Consensus statement of the Movement Disorder Society on tremor". In: vol. 13. Lippincott Williams and Wilkins, 1998, pp. 2–23. DOI: 10.1002/mds.870131303.

[11] Rodger J. Elble. "Central mechanisms of tremor". In: *Journal of Clinical Neurophysiology* 13 (1996), 133–144. DOI: 10.1097/00004691-199603000-00004.

[12] Kailash P. Bhatia et al. "Consensus Statement on the classification of tremors. from the task force on tremor of the International Parkinson and Movement Disorder Society". In: *Movement Disorders* 33 (1 Jan. 2018), pp. 75–87. ISSN: 15318257. DOI: 10.1002/mds.27121.

[13] V. Homberg et al. "Differential effects of changes in mechanical limb properties on physiological and pathological tremor". In: *Journal of Neurology, Neurosurgery and Psychiatry* 50 (5 1987), pp. 568–579. ISSN: 00223050. DOI: 10.1136/jnnp.50.5.568.

[14] Andrew J Hughes et al. *A Clinicopathologic Study of 100 Cases of Parkinson's Disease Mini-Mental State examination6 and Diagnostic and Sta¬ tistical Manual of Mental Disorders, Third Edition criteria. Dementia was defined as a Mini-Mental State examination score of less than 20 in cases satisfying Diagnostic and Statistical Manual of Mental.* URL: https://jamanetwork.com/.

[15] Lonneke M L de Lau and Monique M B Breteler. *Review Introduction.* 2006. URL: http://neurology.thelancet.comVol.

[16] J. J. Van Hilten et al. "Diurnal effects of motor activity and fatigue in Parkinson's disease". In: *Journal of Neurology, Neurosurgery and Psychiatry* 56 (8 1993), pp. 874–877. ISSN: 00223050. DOI: 10.1136/jnnp.56.8.874.

[17] Elan D. Louis, Ruth Ottman, and W. Allen Hauser. "How common is the most common adult movement disorder? Estimates of the prevalence of essential tremor throughout the world". In: *Movement Disorders* 13 (1 1998), pp. 5–10. ISSN: 08853185. DOI: 10.1002/mds.870130105.

[18] Fatemeh N. Emamzadeh and Andrei Surguchov. *Parkinson's disease: Biomarkers, treatment, and risk factors.* Aug. 2018. DOI: 10.3389/fnins.2018.00612.

[19] William Koller et al. "High-frequency unilateral thalamic stimulation in the treatment of essential and Parkinsonian tremor". In: *Annals of Neurology* 42 (3 Sept. 1997), pp. 292–299. ISSN: 03645134. DOI: 10.1002/ana.410420304.

[20] Andres M. Lozano et al. *Deep brain stimulation: current challenges and future directions.* Mar. 2019. DOI: 10.1038/s41582-018-0128-2.

[21] Kai Zhang et al. "Long-term results of thalamic deep brain stimulation for essential tremor". In: *Journal of Neurosurgery* 112.6 (2010), 1271–1276. DOI: 10.3171/2009.10.jns09371.

[22] Hayriye Cagnan et al. "Stimulating at the right time: phase-specific deep brain stimulation". In: (). DOI: 10.1093/aww308.

[23] Shenghong He et al. "Closed-Loop Deep Brain Stimulation for Essential Tremor Based on Thalamic Local Field Potentials". In: *Movement Disorders* 36 (4 Apr. 2021), pp. 863–873. ISSN: 15318257. DOI: 10.1002/mds.28513.

[24] Sabato Santaniello et al. "Closed-Loop Control of Deep Brain Stimulation: A Simulation Study". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 19.1 (2011), pp. 15–24. DOI: 10.1109/TNSRE.2010.2081377.

[25] Michael S. Okun. "Deep-Brain Stimulation for Parkinson's Disease". In: *New England Journal of Medicine* 367 (16 Oct. 2012), pp. 1529–1538. ISSN: 0028-4793. DOI: 10.1056/NEJMct1208070. URL: http://www.nejm.org/doi/abs/10.1056/NEJMct1208070.

[26] Jessica Frey et al. *Transcranial Magnetic Stimulation in Tremor Syndromes: Pathophysiologic Insights and Therapeutic Role.* Aug. 2021. DOI: 10.3389/fneur.2021.700026.

[27] Silvijus Abramavičius et al. "Local vibrational therapy for essential tremor reduction: A clinical study". In: *Medicina (Lithuania)* 56 (10 Oct. 2020), pp. 1–9. ISSN: 16489144. DOI: 10.3390/medicina56100552.

[28] Arthur Prochazka, Josef Elek, and Manouchehr Javidan. "Attenuation of pathological tremors by functional electrical stimulation I: Method". In: *Annals of Biomedical Engineering* 20.2 (Mar. 1992), pp. 205–224. DOI: 10.1007/bf02368521. URL: https://doi.org/10.1007/bf02368521.

[29] Manouchehr Javidan, Josef Elek, and Arthur Prochazka. "Attenuation of pathological tremors by functional electrical stimulation II: Clinical evaluation". In: *Annals of Biomedical Engineering* 20.2 (Mar. 1992), pp. 225–236. DOI: 10.1007/bf02368522. URL: https://doi.org/10.1007/bf02368522.

[30] O. K. Sujith. *Functional electrical stimulation in neurological disorders*. May 2008. DOI: 10.1111/j.1468-1331.2008.02127.x.

[31] Aparna Wagle Shukla. *Rationale and Evidence for Peripheral Nerve Stimulation for Treating Essential Tremor*. 2022. DOI: 10.5334/tohm.685.

[32] Colin G Mcnamara, Max Rothwell, and Andrew Sharott. "Phase-dependent closed-loop modulation of neural oscillations in vivo". In: (). DOI: 10.1101/2020.05.21.102335. URL: https://doi.org/10.1101/2020.05.21.102335.

[33] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. "A review on genetic algorithm: past, present, and future". In: *Multimedia Tools and Applications* 80 (5 Feb. 2021), pp. 8091–8126. ISSN: 15737721. DOI: 10.1007/s11042-020-10139-6.

[34] Jie Wang. "An Intuitive Tutorial to Gaussian Processes Regression". In: (Sept. 2020). URL: http://arxiv.org/abs/2009.10862.

[35] Peter I. Frazier. "A Tutorial on Bayesian Optimization". In: (July 2018). URL: http://arxiv.org/abs/1807.02811.

[36] Niranjan Srinivas et al. "Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting". In: *IEEE Transactions on Information Theory - TIT* 58 (May 2012), pp. 3250–3265. DOI: 10.1109/TIT.2011.2182033.

[37] Denison-Smith. *[Optimizing Phase-Locked Peripheral Stimulation for Tremor Suppression, Honour School of Engineering Science Part C Project]*. University of Oxford, 2022.

[38] Steven Gonzalez Monserrate. *MIT Case Studies in Social and Ethical Responsibilities of Computing • Winter 2022 The Cloud Is Material: On the Environmental Impacts of Computation and Data Storage License: Creative Commons Attribution-NonCommercial 4.0 International License (CC-BY-NC 4.0) MIT Case Studies in Social and Ethical Responsibilities of Computing • Winter 2022 The Cloud Is Material: On the Environmental Impacts of Computation and Data Storage 2 Learning Objectives*. 2022.

[39] Andreas Krause Cheng and Soon Ong. *Contextual Gaussian Process Bandit Optimization*.

[40] Xingyu Zhou and Ness Shroff. "No-regret algorithms for time-varying bayesian optimization". In: Institute of Electrical and Electronics Engineers Inc., Mar. 2021. ISBN: 9781665412681. DOI: 10.1109/CISS50987.2021.9400292.

[41] Paul Brunzema et al. "Event-Triggered Time-Varying Bayesian Optimization". In: (Aug. 2022). URL: http://arxiv.org/abs/2208.10790.

[42] Thomas M Cover and Joy A Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing) (Hardcover)*.

[43] Pembroke College David Kristjanson Duvenaud. *Automatic Model Construction with Gaussian Processes Declaration*. 2014.

[44] David Duvenaud et al. "Structure Discovery in Nonparametric Regression through Compositional Kernel Search". In: (Feb. 2013). URL: http://arxiv.org/abs/1302.4922.

[45] D. J. C. MacKay. "Introduction to Gaussian Processes". In: *Neural Networks and Machine Learning*. Ed. by C. M. Bishop. NATO ASI Series. Kluwer, 1998, pp. 133–166.

[46] *Medical electrical equipment. Part 1-10, General requirements for basic safety and essential performance. Collateral Standard: Requirements for the development of physiologic closed-loop controllers [electronic resource]*. eng. London, 2021.

[47] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

[48] Jeonghee Kim et al. "A wearable system for attenuating essential tremor based on peripheral nerve stimulation". In: *IEEE Journal of Translational Engineering in Health and Medicine* 8 (2020). ISSN: 21682372. DOI: 10.1109/JTEHM.2020.2985058.

[49] Deirdre M. Walsh et al. "Transcutaneous electrical nerve stimulation". In: *American Journal of Physical Medicine amp; Rehabilitation* 74.3 (1995), 199–206. DOI: 10.1097/00002060-199505000-00004.

[50] Alejandro Pascual-Valdunciel et al. "Peripheral electrical stimulation to reduce pathological tremor: a review". In: *Journal of NeuroEngineering and Rehabilitation* 18 (1 Dec. 2021). ISSN: 17430003. DOI: 10.1186/s12984-021-00811-9.