Fakultät Angewandte Informatik
(Faculty Applied Computer Science)
Bachelor Künstliche Intelligenz

Bayesian search threader for split foils in Transferline to CERN SPS Fixed Target Experiments

Bayesscher Suche für split foils in der Transferlinie zu den CERN SPS Fixed Target Experimenten

Bachelorarbeit zur Erlangung des akademischen Grades:
Bachelor of Science an der Technischen Hochschule Deggendorf

Reviewer: Prof. Dr. Glauner Patrick
Advisor: Dr. Kain Verena

vorgelegt von
Name, Vorname: von Hohenbühel, Maximilian
Matrikelnummer: 00817439
Ort, Datum: Geneva, 01/10/2023

CERN-THESIS-2023-301
27/11/2023

Contents

1 Introduction **2**

2 Theoretical background **3**

2.1 Controlling particle trajectories in a transfer line 3

2.2 Symmetry as figure-of-merit of trajectory correction goodness . . 5

2.3 Bayesian Optimisation 5

2.3.1 What is Bayesian Optimisation? 5

2.3.2 Bayesian Optimisation Basics 6

2.3.2.1 Components of Bayesian Optimisation 6

2.3.2.2 Pseudo code for the implementation of the Bayesian Optimisation 10

2.3.3 What are the weaknesses? 10

2.3.4 Why is Bayesian Optimisation suitable for finding the optimal magnet configuration? 11

3 Steering DC beams in the TT20 transfer lines at CERN with Optimisation Algorithms **11**

3.1 The challenge 11

3.1.1 Mad-X 12

3.2 BO for steering DC beams 13

3.2.1 Objective function 13

3.2.2 Measurements 17

3.2.3 Results 18

3.2.4 Further steps and open questions 24

3.3 Conclusion 25

Appendix **26**

A Measurement tables 26

References **30**

1 Introduction

The large CERN accelerator complex comprises high energy physics experiments at the Large Hadron Collider and various so-called Fixed Target experiments. In the case of colliders two counter-rotating particle beams are accelerated and brought into collision at interaction points. For Fixed Target experiments the particle beam is extracted from the accelerator and transferred to a target station where it impacts the target to generate showers of secondary particles. These are either studied directly or sorted into various particle types before transporting them to the actual experiment. The focus in this thesis will be on the transport of the beam from the Super Proton Synchrotron (SPS) to the Fixed Target experiments in the CERN North Area. Figure 2 shows an overview of the about 7 km circumference SPS with its many transfer lines. There are three primary beam targets, T2, T4 and T6, at the end of the transfer line TT20 towards the North Area. The SPS uses a special extraction technique to extract the proton or heavy ion beams as second-long particle spills. This technique is called *slow extraction* [Fraser Et Al. (2017)] and produces quasi-DC beams in the transfer line at low instantaneous intensity. The TT20 transfer line is long, almost 1 km, and design dipole magnet settings along the transport need to be adjusted to correct for accumulating errors. Beam position monitors (BPMs) along the line are used to steer the beam to the center of the vacuum chambers and on target. The algorithms to correct the trajectory given BPM measurements and a set of dipole corrector magnets are well established. They are based on inverting the so-called *response matrix*, that relates a corrector strength change to a position change at a given BPM.

With slow extracted DC beams the usual BPM technology, that relies on bunched beams, does not work. TT20 is therefore not equipped with standard BPMs but rather so-called *split foils* that relate secondary emission intensity measurements, when beam passes through the foils, to position measurements. Still their measurements do not give reliable position measurements as will be discussed in more detail later. Without reliable position measurement, the usual deterministic trajectory correction algorithms also do not work reliably. In fact the worse the trajectory excursions through the line, the worse the algorithm performs. The goal of this thesis is to implement Bayesian Optimisation to correct the trajectory in TT24 (the transfer line that forks off from TT20 to target T4) based on the feedback from split foil measurements. Results with a simulated test environment will be presented.

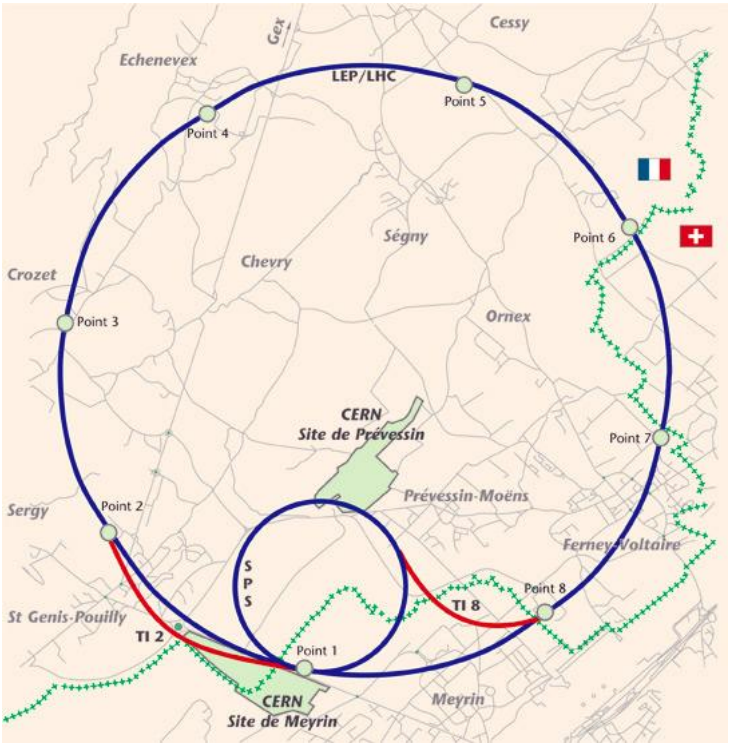

Figure 1: This diagram shows the LHC and the SPS pre-accelerator (in blue) and the transfer lines that will connect them (in red). Spanning the France-Swiss border (shown by green crosses), the 27-km LHC tunnel will receive a beam that has been pre-accelerated to 450 GeV in the smaller SPS storage ring. The transfer lines will remove each beam from the SPS and inject them into the LHC where they will be accelerated to the full energy of 7 TeV. CERN, Laurent Guiraud (2011)

2 Theoretical background

2.1 Controlling particle trajectories in a transfer line

Accelerator transfer lines are used to link accelerators into a chain, as is used for the LHC injector chain which consists of almost all accelerators at CERN to pre-accelerator LHC beams, or to transport the beam from an accelerator to an experimental facility, i.e. fixed target. While transfer lines do not have accelerating units, they consist of electro-magnets for trajectory control and quadrupole magnets for beam size control as you will also find in the actual accelerators. In addition they are equipped with various beam instrumentation to assess properties of the beam such as intensity, trajectory position, beam size, beam loss etc.

Corrector dipole magnets are used to adjust the trajectory of the charged particles in the vertical and horizontal plane wrt the design trajectory by providing bending angles $\Delta\theta$ of typically mrad (the bending angle corresponds to the integrated

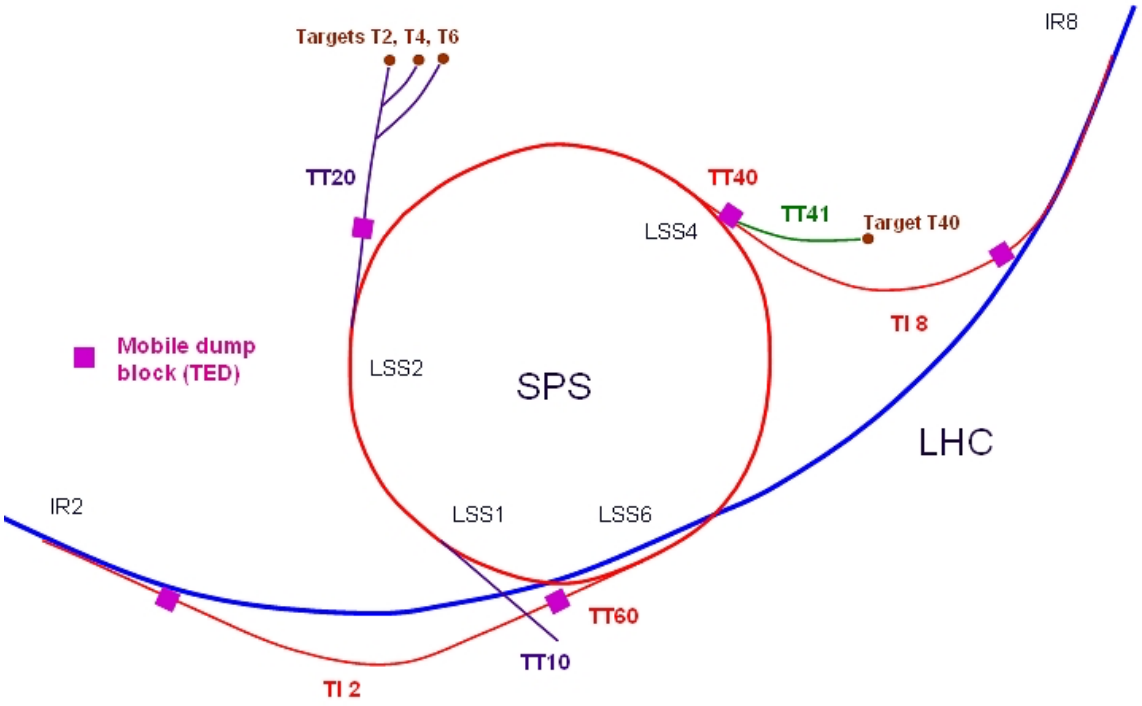

Figure 2: Schematic of the SPS and all adjacent transfer tunnels. The SPS (in red) has three outgoing transfer tunnels TT20, TT40, TT60. The Targets T2, T4, T6 at the end of the TT20 are the focus of this thesis. CERN, Henric Wilkens (2014)

dipole field divided by the ratio of momentum and charge p/q of the particle). The transfer or transport matrix between the corrector and the BPM linearly models the transport of phase-space coordinates from location 0 to location 1 along the line.

$$\begin{pmatrix} x^{(1)} \\ x'^{(1)} \end{pmatrix} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{pmatrix} \begin{pmatrix} x^{(0)} \\ x'^{(0)} \end{pmatrix} \tag{1}$$

A corrector dipole magnet with deflection angle $\Delta\theta$ at the location 0 would change the position measured at BPM by Δx. With equation (1) and initial conditions ($x^{(0)}$, $x'^{(0)}$) = (0, $\Delta\theta$), the following response is obtained:

$$\Delta x = \mathbf{R}_{12}\Delta\theta \tag{2}$$

This idea can be extended to an arbitrary number of BPMs and correctors with the result of an $M \times N$ matrix for M BPMs and N trajectory correctors desribing the response. The response can be derived in several manners, either using simulation codes or measuring it directly with beam. The assumption of the response to be linear also does not hold under all circumstances and then universal function

approximators can be a useful tool for modelling Baird (2007).

2.2 Symmetry as figure-of-merit of trajectory correction goodness

As mentioned earlier, the slow extracted beam sent to the North Area targets is de-bunched and has no RF structure, which makes position measurements with standard SPS BPM electronics impossible. The slow extracted beam position is therefore measured with Secondary Emission Monitors (SEMs) that are made of 1 mm thick Al or 20 μm thick Ti foils. These have a conversion efficiency of 4% electrons per proton. The SEMs used for position monitoring are called BSPs consisting of two foils to measure the position, by comparing Left/Right or Top/Bottom foil signals. BSPs have many short comings as position measurement system. For example the conversion from intensity difference to beam position needs an assumption on beam size and shape, which is not a-priori known. Also, if all the beam ends up on one foil only, it is impossible to convert to a position. The figure-of-merit is the so-called *symmetry S* which is 100% in case the beam is perfectly centered, i.e. same amount of intensity on both foils. It is calculated via the intensity signals from both foils per given monitor as:

$$S = \sqrt{1 - \frac{|I_1 - I_2|}{I_1 + I_2}} \tag{3}$$

where I_1 and I_2 are the intensities measured at the two foils. Obviously symmetry cannot be used as input to the above mentioned inverted response matrices for trajectory correction.

2.3 Bayesian Optimisation

2.3.1 What is Bayesian Optimisation?

Bayesian Optimisation (BO) is a powerful optimization technique for finding the optimal solution to a complex problem, such as tuning hyperparameters of a machine learning model or optimizing physical experiments. The problems or objective functions are treated as blackbox functions. A blackbox function is an input-output mapping where the inner workings are not known, i.e. derivatives of the function wrt input parameters are not available. Often these functions are highly

complex with high-dimensional input parameter space and non-convex behaviour with several local minima. In contrast, popular optimisation techniques such as newton's method, that are very sample-efficient, require the knowledge of the first and second derivative of the objective function and are therefore often suitable.

2.3.2 Bayesian Optimisation Basics

The main component of Baysian Optimzation is the generation of a probabilistic surrogate model (SM) of the objective as function of the input parameters. The algorithm is boostrapped by building an intial surrogate model from random sampling in the input parameter space. The next data point is chosen by optimizing a so-called *acquisition function (AF)* that takes the model uncertainties into account instead of the mean of the model directly. Gaussian processes are usually used to model the objective function due to their many beneficial characteristics. The model is updated with the new data by using Bayesian statistics.

2.3.2.1 Components of Bayesian Optimisation

- Initial datapoints
  To get an initial idea of the objective function, the Bayesian Optimisation algorithm has to first create an idea of the relationship between the input and output arguments. This is usually done by randomly generating a fix number of input parameters, obtaining the objective function for those, and then fitting a regression model with a Gaussian Prozess.

  It is important to remember that even though the actual Bayesian Optimisation is not started yet at this stage, the required number of initial random points to obtain a useful model already counts towards the sample-efficiency of the algorithm. For each of the sampled parameters the objective function has to be obtained from the environment. The sensitivity to number of random data points was investigated in this study for our particular problem.

- Surrogate model Gramacy (2020)
  The surrogate model is the core of the Bayesian Optimisation. In order to find a first approximation of the objective function, a probability distribution has to be generated as best possible approximation. A Gaussian ProcessGörtler, Kehlbeck, and Deussen (2019) is used for that purpose. A Gaussian Process uses a covariance matrix to produce a probability distribution over functions. This distribution is of the multivariate normal distribution, also called the

multivariate Gaussian distribution. The probability density function is given by:

$$f(\mathbf{x};\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)\right)$$

Here, $\mathbf{x}$ represents the vector of random variables, n is the dimension of $\mathbf{x}$, μ is the mean vector, Σ is the covariance matrix, $|\Sigma|$ denotes the determinant of the covariance matrix, and exp is the exponential function.

The covariance matrix, also known as the kernel matrix, plays a crucial role in modelling the relationships between the datapoints. It is usually depicted as Σ or K. It encodes the covariance or correlation between a pair of datapoints $\Sigma(x,x')$. It quantifies the similarity between the outputs of the Gaussian process at those two input locations. Different choices of covariance functions lead to different Gaussian process models with distinct properties. Common covariance functions include the Radial Basis Function (RBF) or Gaussian kernel, linear kernel, periodic kernel, and more.

The RBF kernel is used for this task. This kernel's shape is like a bell curve. Datapoints that are close to eachother will have high similarity scores close to 1. Datapoints that are further away go closer to 0. Mathematically, the RBF is defined as:

$$k(x,x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$$

Where:

- $k(x,x')$ represents the similarity (or covariance) between two points x and x'
- $\|x-x'\|$ is the Euclidean distance between x and x'
- σ^2 is a hyperparameter known as the bandwidth or length scale parameter. It controls the width of the Gaussian bell curve. Smaller σ^2 values result in narrower curves, which means that data points must be very close to each other to have a significant similarity, while larger σ^2 values result in wider curves, allowing for a broader similarity between data points

- Acquisition function

  The acquisition function helps to decide where to evaluaet the objective func-

tion next. This decision is made by balancing exploration and exploitation.
Exploitation prioritises the value with the highest immediate gain based on existing knowledge. It is a risk-averse strategy that tries to maximise short-term results. It is useful when confidence in the data exists.
Exploration involves making choices that are aimed at gathering more information or learning about the environment. It prioritizes uncertainty reduction and expanding knowledge rather than immediately maximizing gains. It is a risk-taking strategy that involves trying out different options, even if they may not seem optimal based on existing information. Exploration is useful when you have limited knowledge or when the environment is dynamic and may change over time. It allows you to discover potentially better options that you might have missed if you only focused on exploitation. It essentially provides a way to choose the most promising points for evaluation while considering both the expected improvement and the uncertainty in those points. For this optimization, the following acquisition functions are compared against eachother:
Upper confidence bound (UCB)

$$UCB(x;\lambda) = \mu(x) + \lambda\sigma(x)$$

The upper confidence bound is a popular method that gives good control over choosing exploration or exploitation over the other. The key element is the variable λ. This variable is a hyperparameter that allows the control over the significance of the uncertainty $\sigma(x)$. The uncertainty, which is represented by using the standard deviation, which is calculated by taking the square root from the predictive variance.

$$\sigma(x) = \sqrt{\sigma^2(x)}$$

The predictive variance is returned by the gaussian process together with the mean μ.
The mean μ is used for exploitation. The mean encourages the algorithm to exploit regions of the search space where the surrogate model predicts high mean values for the objective function. High mean values suggest that these regions are likely to contain good solutions, so exploiting them is a way to focus on areas that are currently perceived as promising.
The standard deviation $\sigma(x)$ encourages the algorithm to explore regions of

the search space where the surrogate model is uncertain about the objective function's behavior. High uncertainty implies that the algorithm is less confident about the true function values in that region, so exploring such areas may lead to the discovery of better solutions. Expected Improvement (EI) can be broken down into multiple computations. All of them rely on the two values that the gaussian process provides:

- $f(x)$, provides the predicted mean (expected value) of the objective function at point x, based on the surrogate model
- $\sigma(x)$, provides the predicted standard deviation (uncertainty) from the surrogate model at point x

From there we can calculate something called *Improvement* as follows

$$I(x) = max(0, f(x) - f_{\text{best}})$$

This calculates how much better the predicted value of the objective function is at point x compared to the best known value. f_{best} is saved and kept track during the algorithm to always know the best known value. This function is then used for the final formula like this:

$$EI(x) = \int_{-\infty}^{\infty} I(x) p(f(x)|\text{data}) df(x)$$

- data, represents all known points up until this point
- $p(f(x)|\text{data})$, calculates the probability distribution of the objective function at point x given data

To compute this integral, various numerical techniques can be used, such as Monte Carlo sampling, numerical integration, or other approximation methods, depending on the complexity of the surrogate model and the objective function.

- Objective function

  The objective function is the core component of the optimization problem. This mathematical function has to be optimised and in our case this means to find the global minimum. The objective function has a blackbox nature. This means that the actual computations of the function are not important and can be ignored. The Bayesian Optimisation doesn't rely on assumptions. Another reason for this is the computational cost to evaluate the objective

function once. In this case, it would mean to send a real beam once through the tunnel, which doesn't only require time, but also energy and the costs that come with the CERN infrastructure. So, the function is only evaluated when needed.

2.3.2.2 Pseudo code for the implementation of the Bayesian Optimisation

1. Initialise N_p number of random datapoints
2. Generate a surrogate model based on the random datapoints with the gaussian process
3. Find the best next x with the acquisition function
4. Run the objective function with the x value from the previous step
5. Add the result of the previous step to the known datapoints
6. Repeat steps 2-5 until the maximum number of iterations or the budget of the objective function has been reached

2.3.3 What are the weaknesses?

- Initial Sample Dependency: The performance of Bayesian Optimisation can be influenced by the quality and quantity of initial data points used to build the surrogate model. Poorly chosen initial samples can lead to suboptimal results.
- Convergence to Local Optima: Like other optimization methods, Bayesian Optimisation is not guaranteed to find the global optimum of a function. It may get stuck in local optima if the surrogate model and acquisition function fail to explore regions with potentially better solutions.
- Limited Exploration in High Dimensions: In high-dimensional spaces, Bayesian Optimisation can struggle to explore effectively due to the curse of dimensionality. It may require an impractical number of evaluations to find a good solution.
- Sensitivity to Hyperparameters: Bayesian Optimisation itself has hyperparameters that need to be tuned, such as the choice of surrogate model (e.g., Gaussian Process or Random Forest) and the acquisition function (e.g., Expected Improvement or Upper Confidence Bound). The performance of Bayesian Optimisation can be sensitive to the choice of these hyperparameters.

- High Computational Cost: Bayesian Optimisation can be computationally expensive, especially when the black-box function is expensive to evaluate. The iterative nature of Bayesian Optimisation, which involves fitting surrogate models and optimizing acquisition functions, can make it slow for problems with a large number of parameters or evaluations.

2.3.4 Why is Bayesian Optimisation suitable for finding the optimal magnet configuration?

The objective of finding the optimal magnet configuration can scale up depending on the transfer tunnel or the accelerator complex. This is not necessarily a black-box function, since we know the internal workings, but nontheless the function is very expensive to compute and that only increases with the number of correctors. For these reasons, an universal algorithm is needed, that can adapt to different dimensionality and for different scenarios and accuracies. Another reason lies in the bayesian nature of the algorithm, making predictions depending on the prior results, which is crucial when it comes to adjusting the magnet configuration step by step until the optimal value is reached.

3 Steering DC beams in the TT20 transfer lines at CERN with Optimisation Algorithms

3.1 The challenge

The challenge during this thesis was to find the global minimum to a blackbox function in as few iterations as possible. The number of iterations has to be small enough to be able to make the adjustments in real time while reaching a set threshold for the beam symmetry. The data during this research was provided through simulations. The simulations are accurate representations of the beam optics inside the transfer tunnel from a CERN-internal tool called Mad-X (Methodical Accelerator design) CERN (2022). Mad-X requires knowledge of the actual transfer tunnel and all the magnets. This data is given through a configuration file. From there, all data that is used for the optimisation algorithm is calculated in real-time.

An important note for this research is that on the virtual PC (VPC) given by CERN, the latency between cycles is depending on the CPU of the VPC. In a real scenario it is important to note that the real-time capabilities of this algorithm depends on the machines it is run on. So far, CERN only has access to a CPU driven computers, which could be replaced with a GPU based approach. This would require enough evidence to assure that a GPU would make the entire process more efficient and can shorten the time it needs to calibrate the accelerator.

3.1.1 Mad-X

The methodical accelerator design tool, short MAD, is a CERN-internal program for particle physics design and simulation. It provides a scripting language, which allows the detailed description of the accelerator or transfer line.

Mad-X simulates an accurate depiction of the accelerator/transfer tunnel through a custom scripting language. CERN maintains a repository with all the important configuration files for the existing accelerators and transfer lines. These files contain a digital copy of all the important parameters that are necessary to make the simulations as real as possible. With all these values Mad-X can initialise a live environment that is a real-time accelerator. Through this, the beams can be simulated to travel through the tunnel, measure the symmetries on the way and adjust the values based on our objective function before they start their second loop. In addition, Mad-X allows to restrict the number of correctors and BSPs that are being monitored. For example it is possible to initially only run the optimisation on a few magnets and split foils instead of taking the entire tunnel. This drastically reduces the time it needs to simulate all values. Afterwards, it is possible to omit these restrictions to compare how the results differ.

3.2 BO for steering DC beams

For the implementation, the Bayesian Optimization was not written from scratch but with help from a framework called BoTorch Balandat et al. (2020). This framework is written upon PyTorch Paszke et al. (2019), which is a python library that focusses on accelerating machine learning techniques. It also simplifies research by providing pre-implemented algorithms. These algorithms are written in a way that let them be easily manipulated with hyperparameters and custom extensions to make it best suitable for research and testing purposes. This implementation takes the already implemented Bayesian Optimization from BoTorch and adds all the details that have been discussed in the previous sections.

3.2.1 Objective function

The goal of the objective function is to provide a value that shows how well the beam trajectory is throughout the entire tunnel. The symmetries can be calculated as shown in 3. This calculation will return a list of symmetries. Its length depends on how many BSPs are used. In reality you would always look at every single BSP, but the simulation through MAD-X allows to focus on only a few to make it computationally simpler. From there, the goal is to find a way to compute a single value that indicates how good or bad the list of symmetries is.

The function can only be visualised up to 4 dimensions, where the 3 axis represent the symmetries and the color represents the result of the objective function. This means that at the beginning only 3 correctors and their BSPs were selected to visualise the 3 dimensional landscape of the function.

For this task, 4 different objective functions were chosen to be evaluated against each other. Every objective function is by default made for maximising, since the focus here is on minimising, all results of the objective functions are multiplied by -1. Furthermore, X represents all known symmetries and n is the number of known symmetries up to this point. For the charts, all three axis were spread evenly with values from -1 to 1. The points are far enough apart to make the 3 dimensional landscape as visible as possible.

$$MAE = \frac{\Sigma_{i=1}^{n} |X_i|}{n}$$

Mean absolute error (MAE) is usually chosen as an error or loss function when you have prediction values and truth values. In this case, it is just calculating the

mean of the absolute values of the symmetries. As shown in figure 3, the scale of the output ranges from 0 to -1. The chosen threshold of success is at -0.85.

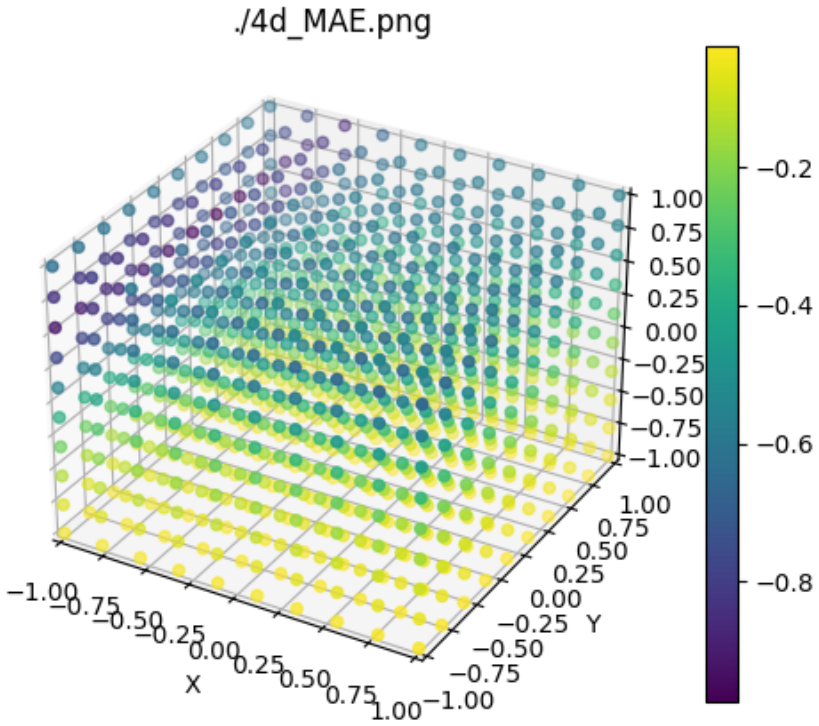

Figure 3: Visualisation of the objective function using MAE.

$$RMS = \sqrt{\frac{\Sigma_{i=1}^{n} X_i^2}{n}}$$

The root mean square (RMS) is also often used for error and loss functions. Due to squaring all values before the mean is taken, higher values will have more weight in regard to lower ones. In figure 4 it is shown how the objective function behaves with 3 input parameters. The threshold for success is also set to -0.85.

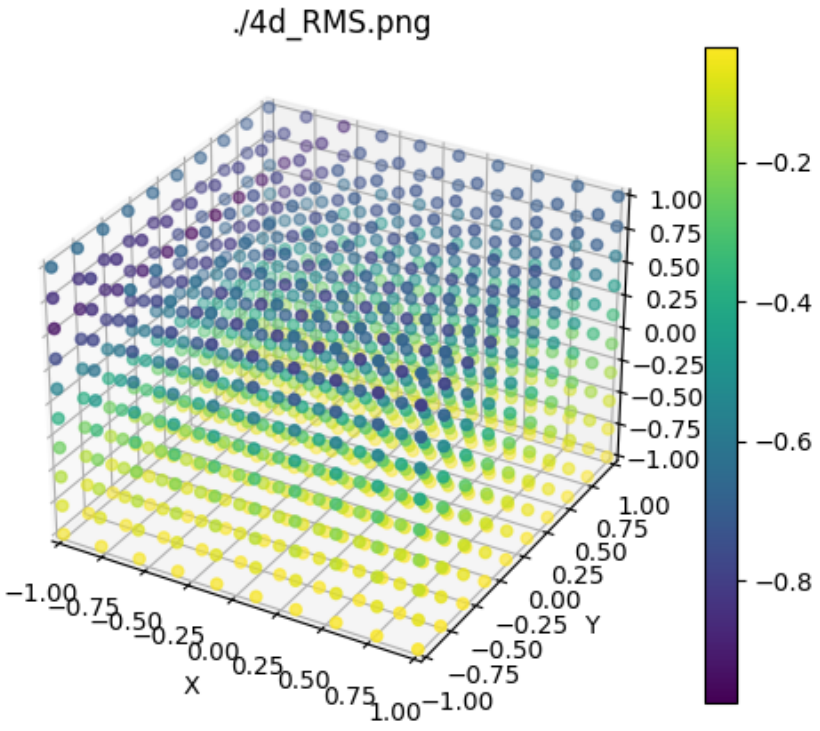

Figure 4: Visualisation of the objective function using RMS.

$$RMC = \sqrt{\frac{\Sigma_{i=1}^{n} X_i^3}{n}}$$

The root mean cube (RMC), also known as the Cubic Mean, is used less often than the root mean square, because of its computational complexity and its usefulness in only few occasions. In this case, comparing both functions can be done through our charts figure 4 and 5. If you follow the dark blue dots you can see that the region of the RMC is narrower than the one from RMS, which means that the objective function should have an easier time to find the one big global minima. Since the output range is here as well between 0 and -1, the threshold is set to -0.85.

$$AbsSum = \Sigma_{i=1}^{n} |X_i|$$

Taking the sum of the absolute values was an experiment to see how well such a simple approach would work. The difference that is immediately recognisable is that the output doesn't range from 0 to -1, but from 0 to $-\text{len}(S)$. S stands for the set of known symmetries up to this point. This means that the threshold of success

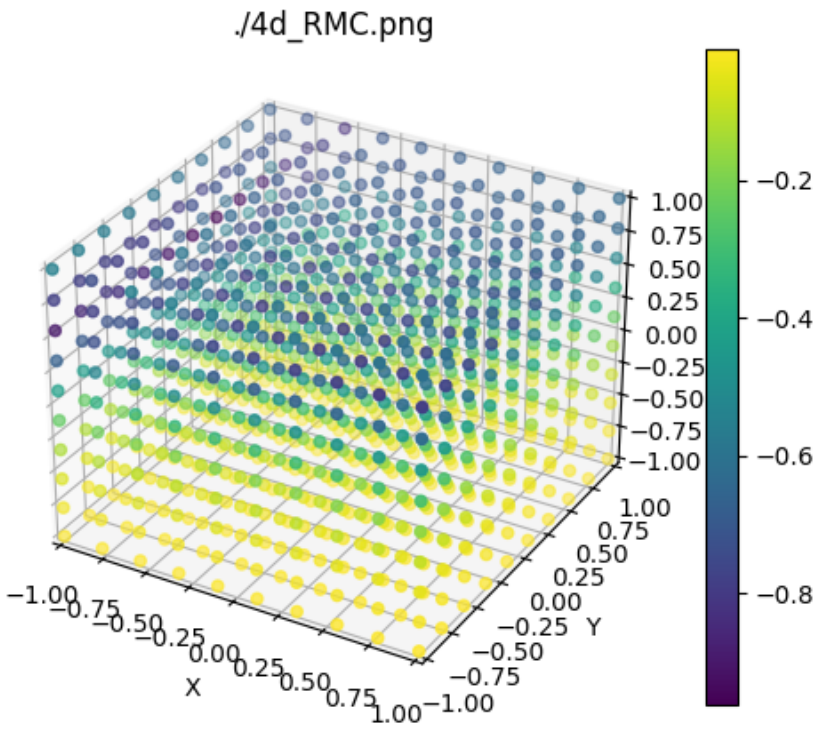

Figure 5: Visualisation of the objective function using RMC.

has to be set differently. In figure 6 it can be seen that the darkblue values start at -2.5, so the threshold was set to -2.7 to guarantee a precise setting. Another interesting point is that the 3 dimensional landscape in figure 6 looks very close to the one from MAE in figure 3.

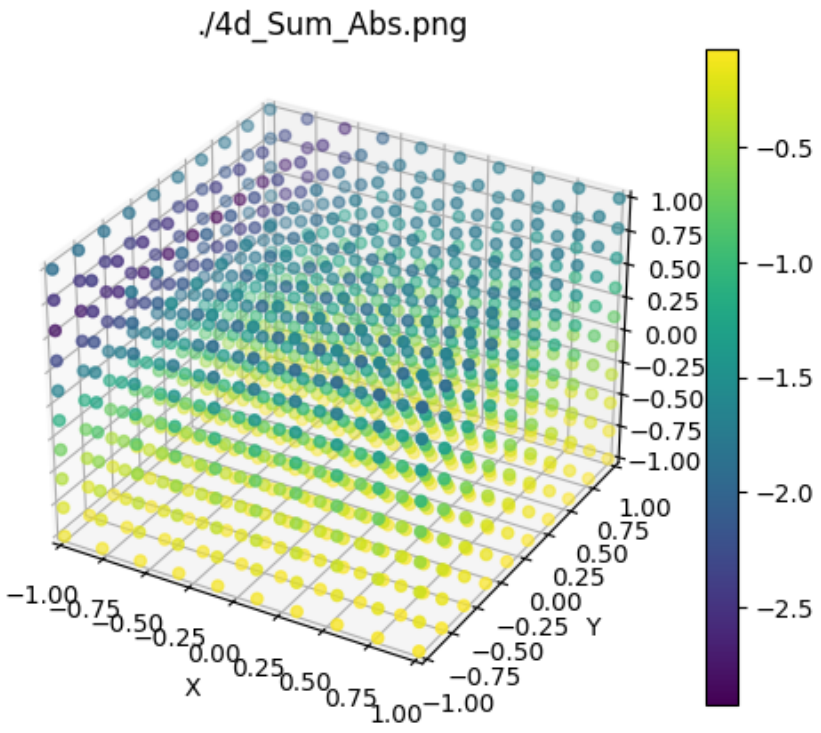

Figure 6: Visualisation of the objective function using the sum of the absolute values.

3.2.2 Measurements

All measurements were done by only looking at 3 correctors, which means always one BSP and a following corrector magnet to adjust the course. The performance should be scalable up to the entire tunnel, but because of computation limits, all tests were done with only 3.

The initialisation pairs represent the initial values of the correctors. These values can range from -1 to 1. For these measurements, all initialisation pairs were chosen at random. This removes the bias of choosing the values, which can lead to favouring values that will lead to more favourable results. In addition, it shows the weakness that the initial values make a huge difference in the number of iterations needed.

The Bayesian Optimisation was tested with UCB and EI as acquisition function and four different objective functions. Each combination was run 5 times. For each run, the amount of necessary iterations was saved and returned. After all runs, the mean was taken to get a useful result as shown in the tables below. For

the measurements, the maximum amount of iterations is set to 50 and the number of initial datapoints is 15. If the number of iterations did not reach 50, then the Bayesian Optimisation found an optimal solution that is equal to or better than the set threshold. The threshold was set differently based on the objective function, since the objective functions can have different ranges of their output. The *RMS*, *RMC*, and *MAE* all return -1 in the most optimal case. For these the threshold was set to -0.85. For the Absolute sum, the threshold is dependent on the amount of symmetries that are looked at. The best value is the number of symmetries. In this test, 3 symmetries are used and the threshold was set to -2.7. The minus comes from the fact that the goal is to minimise the objective function and not maximise it, therefore all results of the objective functions are multiplied by -1. Everything below 50 means that it reached the desired threshold of a proper optimised result. The actual iterations used for optimising, can be counted as every iteration past 15.

The beta value of the UCB algorithm is set to 0.01, which means that it focuses on exploitation.

From the measurements across the 30 different initialisation values, the following ranking can be justified by counting the best mean average for each of the 30 runs:

1. Absolute Sum (7 UCB, 8 EI)
2. MAE (8 UCB, 3 EI)
3. RMS (3 UCB, 1 EI)
4. RMC (0 UCB, 0 EI)

This shows that in the 30 random initialization points, the Absolute Sum has the lowest number of iterations quite equally for UCB and EI. The close second being MAE, can be explained by looking at the figure 3 and 6. Both show similar landscapes with a very narrow area where the top 10% is reached. The Absolute Sum has a slightly better landscape, since the optimal points area is even smaller. This kind of landscape makes acquisition functions that prioritise exploitation better. It is clear from the landscape that exploration leads to more and more flat and uninteresting parts of the objective function.

3.2.3 Results

In this part of the research, the focus is on the upper confidence bound (UCB) acquisition function. Compared to Expected Improvement (EI), UCB allows for a

more fine grained control over the exploration against exploitation problem through the variable beta. The random points are a very important metric to look at, because they indicate how many iterations are at least going to be computed. This allows to have a better estimate how expensive this operation would be if done on the real tunnel/accelerator. The beta value on the other hand has no direct correlation with the real usage. But it is very important to realise the efficiency of the objective function. For example, it was said in the previous section that exploitation is preferred over exploration in this case, since all objective landscapes show narrow global minima.

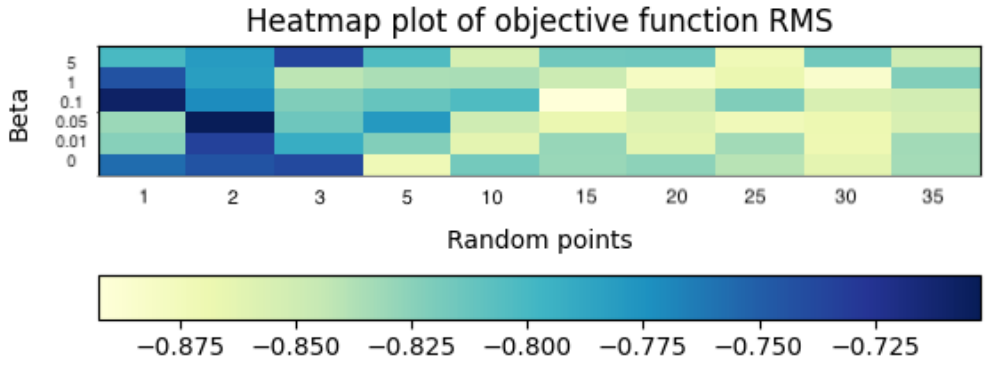

Figure 7: Heatmap to show the RMS objective function with various beta and random datapoints

The heatmaps in figures 7, 8, 9 show the average outcome for various combinations of beta and random point values. The objective function RMC was purposefully left out, since it performed badly in the first long measurement. To work with statistically valid results, every combination was run 5 times with different initial values. The final result shown on the heatmap is the mean of all 5 runs. The higher the negative number from the objective function, the better. Besides that, it is important to see which combinations went above the given thresholds, defined in 3.2.1. In comparison with MAE and Absolute Sum, the RMS objective function doesn't stay stable when the random points increase. Surprisingly, the Absolute sum has the best heatmap, which can be seen that the lowest objective function values are already showing at 5 random points.

Compared to figure 7, the heatmap of the MAE objective function in figure 8 shows a much brighter average. This means not much if the legend is consulted and it is seen that the threshold is reached at -0.85, which makes only the brightest spots a good candidate.

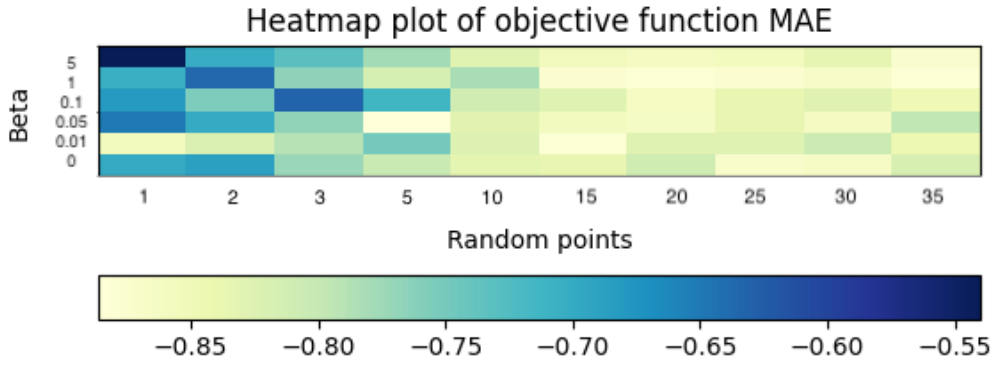

Figure 8: Heatmap to show the MAE objective function with various beta and random datapoints

It is again very surprising that such a simple objective function as taking the sum of the absolute values, gives such a good result. The heatmap 9 shows that a lot of configurations after the 5 random points are within reach of the threshold, which is -2.7. Even more interesting is that the value of beta, which controls the uncertainty and the exploration aspect of the acquisition function, is weighted only very lightly and has only very limited effect on the outcome.

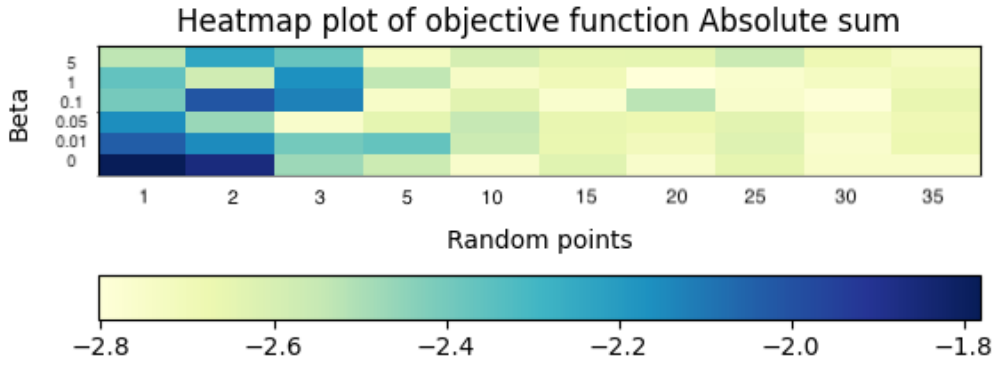

Figure 9: Heatmap to show the Absolute Sum objective function with various beta and random datapoints

The histograms in figures 10, 11, 12 show a frequency distribution over 200 runs of each objective function. The hyperparameters (random points, beta) were set to the best values taken from the heatmaps 7, 8, 9.

The outcome of the frequency distribution from the RMS objective function can be directly seen as well at the first measurements in 3.2.2. It was already clear that RMS is not the best function to be used in this scenario, which explains the huge spike at bucket number 65 (the maximum number of iterations). Furthermore, you can still see a dominance in the 15-25 range, which would make it a very good distribution if it would be possible to get rid of the final spike. The frequency distribution from MAE at figure 11 shows why it is a good match. It has a very high representation at the 15-20 range. With the random points set to 15, that means that the optimization alone took less than 5 iterations. Even here, the spike at 65 can be seen to be quite substantial. This can be related back to the random initialisation, which can make a crucial difference in the search for the global optima.

A very good example is given by figure 12. This is the first objective function where the main spike isn't at the 65 mark, but around the 15-20 mark. It then

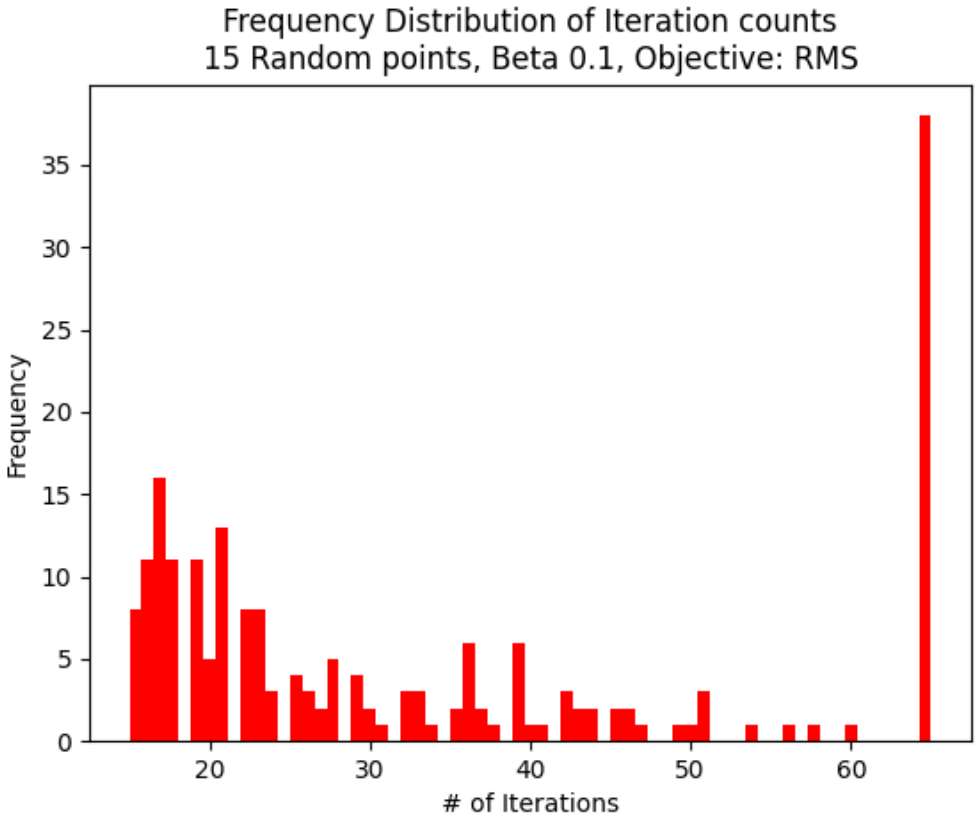

Figure 10: Frequency distribution of Iteration Count with RMS

shrinks a bit, but it is still very respectable until point 32. This makes it the best objective function with these chosen hyperparameters.

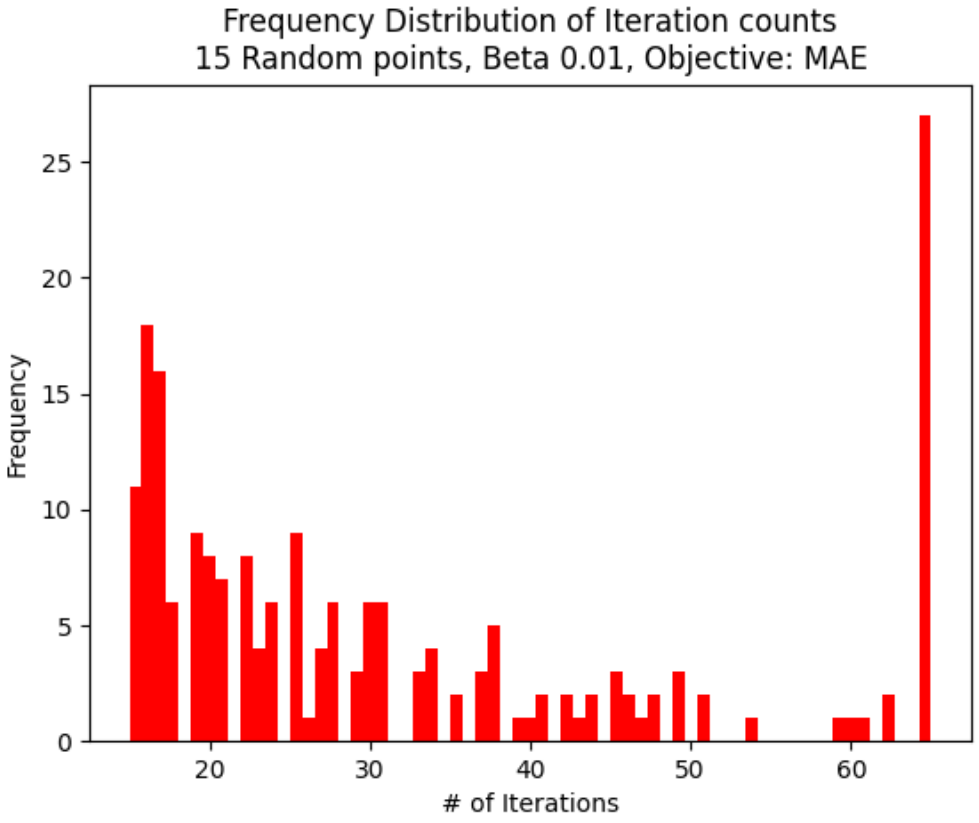

Figure 11: Frequency distribution of Iteration Count with MAE

3.2.4 Further steps and open questions

One of the biggest open question is how the simulations manifest in the real tunnel/accelerator. From the results, it is clear that this automated configuring of the magnets will be an improvement over the tedious manual labour that is being done today. Before trying it on the real tunnel, a thorough final check on the full tunnel simulation will be necessary. Afterwards, the best hyperparameters were found with enough data to have some fallback values. In addition, to just setting the correct hyperparameters, there is the option of using a GPU for accelerated computing. This can be an interesting next question for a different research topic to figure out if a GPU provides any necessary benefits. It is also to be seen if it is even necessary, in case the CPU solution already works well enough for the operators of the accelerator complex. Furthermore, if the tests on the real beam deem successful, this research could be a first step in implementing a generalised and automated beam steering, which could take a lot of work from accelerator operators and technicians. It can also help in guaranteeing more accurate results, by allowing to run this calibration every time no experiments are being observed.

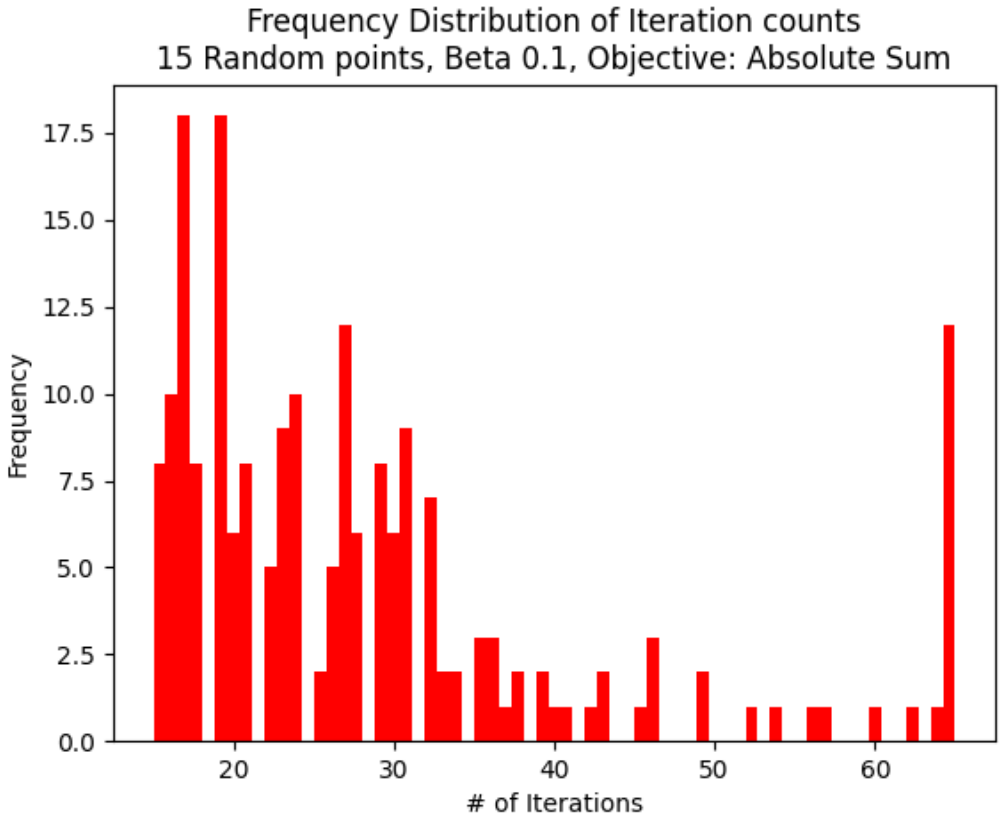

Figure 12: Frequency distribution of Iteration Count with Absolute Sum

3.3 Conclusion

In conclusion, this thesis demonstrates the successful application of Bayesian Optimization in configuring corrector magnets to improve beam trajectory accuracy in particle accelerators specifically the transfer tunnel 20. This investigation has highlighted the strengths of using Mean Absolute Error (MAE) and the Absolute Sum as objective functions. From those both the optimal number of iterations was given by taking the upper confidence bound (UCB) acquisition function with a beta value of 0.01 for MAE and 0.1 for the Absolute sum. Since the best starting number of random points was found to be 15, the optimal number of iterations is between 15 and 20. Those values are reached most reliably with Absolute Sum and UCB combined. Notably, there is still untapped potential for optimization, and further research is warranted to determine which objective function performs best under full tunnel conditions.

Appendix

A Measurement tables

Initialisation	RMS ucb	RMS ei
-0.889821 -0.45911028 -0.97197473	22.8	28.4
-0.08648741 0.35128407 -0.59522686	42.6	29.6
-0.1697427 , -0.29823901, 0.13900941	26.8	24.2
-0.32824758, -0.1626315 , -0.0576121	30	19.8
0.86922725, 0.39667686, -0.18956784	37.6	28.4
-0.16841741, 0.08013397, 0.18117423	32.4	32
0.42761643, -0.38877302, 0.97708437	28.8	26.4
-0.7232354 , 0.21295243, 0.08764695	36.8	35
0.18727662, -0.59378722, -0.5497707	32	24.6
-0.57401718, -0.39141282, 0.20204945	31.6	30.8
0.32814274, 0.85092472, -0.07302344	33.6	34.4
-0.38150082, -0.68143146, -0.60265766	28.2	28.2
0.96845872, -0.80165834, 0.72170899	38.6	37.2
-0.83331525, 0.20334717, 0.33646319	38	36.8
0.37827371, 0.95393953, 0.67516138	19	31.4
0.92841881, -0.86599673, 0.26651932	28.8	44.2
-0.13603431, 0.04622159, -0.05236855	31.4	31.2
0.26753038, 0.74071163, 0.73690539	40.4	42.8
-0.84497461, -0.08071735, -0.27230307	25.8	35.2
0.51817697, 0.21757885, -0.4773089	29.6	31.6
-0.71706358, 0.87371478, 0.59304287	25.6	28
0.83885419, -0.90713608, -0.9514095	40.4	41.4
0.2989861 , -0.00851078, -0.82124938	32.6	28.4
0.91670533, 0.3951833 , 0.67572874	46	29.4
-0.59725766, -0.28990502, -0.44151466	37.2	31.2
-0.76467599, 0.48124502, 0.7794903	45.2	38.2
-0.28281368, -0.65184719, -0.01108447	32.6	25.6
0.11451572, 0.02891114, 0.57412882	27	22.6
-0.74374021, 0.56128327, -0.93293891	41	36.8
0.90126865, 0.43904643, 0.33599889	30.6	47.2

Table A.1: The mean of the results from the RMS objective function

Initialisation	MAE ucb	MAE ei
-0.889821 -0.45911028 -0.97197473	26.8	26.2
-0.08648741 0.35128407 -0.59522686	30.6	30.4
-0.1697427 , -0.29823901, 0.13900941	29.4	22.4
-0.32824758, -0.1626315 , -0.0576121	20.8	23.6
0.86922725, 0.39667686, -0.18956784	27	30.6
-0.16841741, 0.08013397, 0.18117423	24.2	20.6
0.42761643, -0.38877302, 0.97708437	20.2	26
-0.7232354 , 0.21295243, 0.08764695	24.6	32.4
0.18727662, -0.59378722, -0.5497707	24.2	24.4
-0.57401718, -0.39141282, 0.20204945	35.8	40.6
0.32814274, 0.85092472, -0.07302344	25.8	23.4
-0.38150082, -0.68143146, -0.60265766	18.6	21
0.96845872, -0.80165834, 0.72170899	38	31.6
-0.83331525, 0.20334717, 0.33646319	19	35
0.37827371, 0.95393953, 0.67516138	28.8	25.4
0.92841881, -0.86599673, 0.26651932	24.2	32
-0.13603431, 0.04622159, -0.05236855	22	17.4
0.26753038, 0.74071163, 0.73690539	22.4	31.6
-0.84497461, -0.08071735, -0.27230307	36	23.2
0.51817697, 0.21757885, -0.4773089	29.6	37.8
-0.71706358, 0.87371478, 0.59304287	36.4	31.6
0.83885419, -0.90713608, -0.9514095	44	41.6
0.2989861 , -0.00851078, -0.82124938	18.6	31
0.91670533, 0.3951833 , 0.67572874	22.6	31.6
-0.59725766, -0.28990502, -0.44151466	34.4	32
-0.76467599, 0.48124502, 0.7794903	35.8	32.8
-0.28281368, -0.65184719, -0.01108447	18.4	24.8
0.11451572, 0.02891114, 0.57412882	26.2	23.4
-0.74374021, 0.56128327, -0.93293891	33.4	36.4
0.90126865, 0.43904643, 0.33599889	28	36.8

Table A.2: The mean of the results from the MAE objective function

Initialisation	RMC ucb	RMC ei
-0.889821 -0.45911028 -0.97197473	30	39.4
-0.08648741 0.35128407 -0.59522686	42	45
-0.1697427 , -0.29823901, 0.13900941	36	34.4
-0.32824758, -0.1626315 , -0.0576121	39.4	33
0.86922725, 0.39667686, -0.18956784	46.4	45.4
-0.16841741, 0.08013397, 0.18117423	34.4	26.6
0.42761643, -0.38877302, 0.97708437	30.8	36.4
-0.7232354 , 0.21295243, 0.08764695	50	43.4
0.18727662, -0.59378722, -0.5497707	41.2	33.2
-0.57401718, -0.39141282, 0.20204945	39.8	45.4
0.32814274, 0.85092472, -0.07302344	33	24.6
-0.38150082, -0.68143146, -0.60265766	33.8	39.6
0.96845872, -0.80165834, 0.72170899	29.2	31.6
-0.83331525, 0.20334717, 0.33646319	41.4	36.8
0.37827371, 0.95393953, 0.67516138	31.2	33.4
0.92841881, -0.86599673, 0.26651932	40	37.8
-0.13603431, 0.04622159, -0.05236855	29.4	40.4
0.26753038, 0.74071163, 0.73690539	32.4	41.6
-0.84497461, -0.08071735, -0.27230307	42	35.2
0.51817697, 0.21757885, -0.4773089	34.2	30.4
-0.71706358, 0.87371478, 0.59304287	45.2	40
0.83885419, -0.90713608, -0.9514095	43	43.8
0.2989861 , -0.00851078, -0.82124938	37.6	42
0.91670533, 0.3951833 , 0.67572874	34	35.2
-0.59725766, -0.28990502, -0.44151466	41.4	40
-0.76467599, 0.48124502, 0.7794903	50	50
-0.28281368, -0.65184719, -0.01108447	30	30.8
0.11451572, 0.02891114, 0.57412882	25.8	40.2
-0.74374021, 0.56128327, -0.93293891	50	49.6
0.90126865, 0.43904643, 0.33599889	46	39.6

Table A.3: The mean of the results from the RMC objective function

Initialisation	Sum Abs ucb	Sum Abs ei
-0.889821 -0.45911028 -0.97197473	25.2	21.6
-0.08648741 0.35128407 -0.59522686	26.6	29
-0.1697427 , -0.29823901, 0.13900941	26	25.2
-0.32824758, -0.1626315 , -0.0576121	31.8	23.8
0.86922725, 0.39667686, -0.18956784	26.6	35.4
-0.16841741, 0.08013397, 0.18117423	20	20
0.42761643, -0.38877302, 0.97708437	26.2	30.6
-0.7232354 , 0.21295243, 0.08764695	33	30.4
0.18727662, -0.59378722, -0.5497707	26.6	23.3
-0.57401718, -0.39141282, 0.20204945	25.4	22.6
0.32814274, 0.85092472, -0.07302344	19.2	25
-0.38150082, -0.68143146, -0.60265766	23	28
0.96845872, -0.80165834, 0.72170899	33.2	22.8
-0.83331525, 0.20334717, 0.33646319	22.8	21.4
0.37827371, 0.95393953, 0.67516138	29.6	39
0.92841881, -0.86599673, 0.26651932	30.8	22.2
-0.13603431, 0.04622159, -0.05236855	21.6	23.4
0.26753038, 0.74071163, 0.73690539	20.2	20.8
-0.84497461, -0.08071735, -0.27230307	23.8	24.8
0.51817697, 0.21757885, -0.4773089	23	31.2
-0.71706358, 0.87371478, 0.59304287	29.8	35
0.83885419, -0.90713608, -0.9514095	40.8	42.8
0.2989861 , -0.00851078, -0.82124938	24	22.6
0.91670533, 0.3951833 , 0.67572874	32	31
-0.59725766, -0.28990502, -0.44151466	29.6	26.8
-0.76467599, 0.48124502, 0.7794903	40.6	27.4
-0.28281368, -0.65184719, -0.01108447	20.4	26.2
0.11451572, 0.02891114, 0.57412882	21.4	31.4
-0.74374021, 0.56128327, -0.93293891	39.8	40.4
0.90126865, 0.43904643, 0.33599889	31.6	28.6

Table A.4: The mean of the results from the absolute sum objective function

References

Baird, S. (2007). *Accelerators for pedestrians; rev. version* (Tech. Rep.). Geneva: CERN. Retrieved from https://cds.cern.ch/record/1017689

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., & Bakshy, E. (2020). BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in neural information processing systems 33*. Retrieved from http://arxiv.org/abs/1910.06403

CERN. (2022). *Mad-x official website.* Retrieved from https://madx.web.cern.ch/madx/ ([Online; accessed September 28, 2023])

CERN, Henric Wilkens. (2014). *Sps complex delivers first beams.* Retrieved from https://ep-news.web.cern.ch/content/sps-complex-delivers-first-beams ([Online; accessed September 28, 2023])

CERN, Laurent Guiraud. (2011). *Plan of sps to lhc transfer tunnels.* Retrieved from https://www.flickr.com/photos/arselectronica/5679905067/in/photostream/ ([Online; accessed September 28, 2023])

Fraser Et Al., M. A. (2017). Slow extraction at the sps: Extraction efficiency and loss reduction studies. *CERN Proceedings*, Vol 2 (2017): Proceedings of the Injector MD Days 2017. Retrieved from https://e-publishing.cern.ch/index.php/CP/article/view/653 doi: 10.23727/CERN-PROCEEDINGS-2017-002.87

Gramacy, R. B. (2020). *Surrogates: Gaussian process modeling, design and optimization for the applied sciences.* Boca Raton, Florida: Chapman Hall/CRC. (http://bobby.gramacy.com/surrogates/)

Görtler, J., Kehlbeck, R., & Deussen, O. (2019). A visual exploration of gaussian processes. *Distill*. (https://distill.pub/2019/visual-exploration-gaussian-processes) doi: 10.23915/distill.00017

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf