FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program: Computational Engineering	The spring semester 2023 Open
Author: Seyedehmina Mirmohammadhosseini	
Supervisor at UiS: Professor Reidar B. Bratvold Co-supervisor: Dr. Aojie Hong Co-supervisor: Peyman Kor- PhD fellow	
Thesis Title: Probabilistic Machine Learning for Production Optimization Under Uncertainty	
Credits: 30 ECTS	
Keyword: Bayesian Optimization, Machine learning, Production optimization under uncertainty	Pages: + appendix: 54 Stavanger, June 2023

Abstract

This thesis investigates the application of Bayesian Optimization (BO) in the optimization of the Expected Net Present Value (ENPV) for oil field development. The objective is to maximize the ENPV while reducing the computation time required for the optimization process. The challenges associated with this optimization problem, such as the absence of an analytical form of the objective function, the presence of multiple local optima, and the computational demands of evaluating the ENPV, are addressed using BO.

I begin by providing a comprehensive overview of BO and its suitability for addressing these challenges. I demonstrate its effectiveness through a 1D toy problem, where the objective function is treated as a black box and iteratively approximated using a Gaussian Process. This serves as a foundation for applying BO to a realistic 3D reservoir simulation model, where I optimize water injection rates to maximize the ENPV.

To further improve the optimization process, I investigate the impact of initial value selection methods on computation time and ENPV convergence. I compare two approaches: random selection using a uniform distribution and Latin Hypercube Sampling (LHS) to ensure scattered initial injection rates for wells. The results highlight the significance of initial value selection, with LHS demonstrating reduced computation time while maintaining comparable ENPV outcomes.

Moreover, I propose a modified method to decrease the optimization time by employing an eliminating criterion. I evaluate the feasibility of using the first realization NPV as a decision point to determine whether further NPV calculations are required for other realizations. I demonstrate the effectiveness of this approach in reducing computation time, particularly in cases with low uncertainty.

Overall, this thesis contributes to the understanding of how initial value selection and modified BO methods can improve the efficiency and effectiveness of ENPV optimization in oil field development. The results provide insights for decision-making processes and can guide practitioners in maximizing the value of oil field investments while reducing computation time.

Acknowledgement

I would like to express my deepest gratitude to my advisor Professor Reidar B. Bratvold for his guidance, support, and invaluable expertise throughout the course of this research. His insightful feedback, encouragement, and dedication have been instrumental in shaping this thesis.

I would also like to extend my appreciation to my co-advisor Peyman Kor for his valuable input and contributions to my academic journey. His knowledge and guidance have been immensely beneficial in broadening my understanding of the subject matter.

I would like to thank my colleagues and friends for their encouragement, stimulating discussions, and moral support. Their presence has made this research experience more enjoyable and fulfilling.

Lastly, I would like to express my heartfelt gratitude to my family for their unwavering love, encouragement, and belief in my abilities. Their continuous support and understanding have been my source of strength throughout this journey.

This research would not have been possible without the collective efforts, guidance, and support of all those mentioned above. I am sincerely thankful to each and every one of them.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. Introduction

Optimization refers to the process of maximizing the utilization or effectiveness of a given situation or resource. Essentially, it involves modifying an existing procedure with the aim of increasing the frequency of favorable results and reducing the occurrence of unfavorable outcomes. This inclination towards optimization is inherent in human nature, as individuals consistently endeavor to enhance their personal well-being and the environment in which they reside. On a collective level, societies struggle to allocate limited resources seeking to improve the welfare of their members. Optimization has played a pivotal role in driving societal advancements, dating back to the domestication of crops through the practice of selective breeding over 12,000 years ago. This ongoing endeavor towards optimization remains a significant undertaking in contemporary society (Garnett, Roman 2023). The process of optimization is often lengthy and demanding, requiring considerable investment of time and resources for the evaluation of various alternatives. Therefore, it becomes essential to explore methodologies that efficiently allocate time and resources in order to streamline the optimization process.

Decision making can be defined as “choosing the best alternative that best fits a set of goals” (Bratvold and Begg 2010). Decision making and optimization are intricately interconnected. The process of decision making involves assessing available options and choosing the most favorable course of action based on certain criteria or objectives. Optimization, on the other hand, entails finding the best possible solution or maximizing the desired outcome within a given set of constraints. In this way, decision making relies on optimization techniques to identify the optimal choice among available alternatives, while optimization relies on effective decision making to determine the most advantageous approach.

Bayesian Optimization is an approach that uses Bayes Theorem to direct the search in order to find the minimum or maximum of an objective function. Bayesian optimization has found a niche in optimizing objectives that:

- are costly to compute, precluding exhaustive evaluation,
- lack a useful expression, causing them to function as “black boxes,”
- cannot be evaluated exactly, but only through some indirect or noisy mechanism, and/or

- offer no efficient mechanism for estimating their gradient (Garnett, Roman 2023).

Bayesian Optimization is best suited for optimization over continuous domains of less than 20 dimensions and tolerates stochastic noise in function evaluations. It builds a surrogate for the objective and quantifies the uncertainty in that surrogate using a Bayesian machine learning technique, Gaussian process regression, and then uses an acquisition function defined from this surrogate to decide where to sample (Frazier 2018). Bayesian Optimization can be instrumental in various domains where the efficient search for an optimal solution is crucial. Consider a data scientist crafting a complex machine learning model – say a deep neural network – from training data. To ensure success, the scientist must carefully tune the model's hyperparameters, including the network architecture and details of the training procedure, which have massive influence on performance. Unfortunately, effective settings can only be identified via trial-and-error: by training several networks with different settings and evaluating their performance on a validation dataset (Garnett, Roman 2023) which is expensive in terms of time and resource. The search for the best hyperparameters is of course an exercise in optimization (Garnett, Roman 2023). As another example, hydrocarbon production optimization is a decision-making process in which a set of control (or decision) variables is tuned to maximize a predefined objective function (i.e., net present value (NPV)) (Kor, Hong, and Bratvold, n.d.). The exploration and production industry is about exploring, appraising and producing oil and gas. Each phase involves numerous decision points that require significant commitments of time and resources from companies. These commitments can vary in scale, ranging from minor engagements, such as a few days of work or the expenditure of a modest budget, to substantial endeavors involving thousands of personnel working on billion-dollar investments spanning several years (Bratvold and Begg 2010). Since the flow models are typically time dependent and nonlinear, the solution of the resulting nonconvex optimization problem may be computationally demanding (de Brito and Durlofsky 2020). Bayesian optimization has proven successful in settings spanning science, engineering, and beyond, including of course hyperparameter tuning (Turner et al. 2021).

1.1. Bayesian Optimization

Bayesian optimization (BO) refers to a sequential design approach utilized for globally optimizing black-box functions (Boender and Mockus 1991) that does not assume any functional forms. It is a class of machine-learning-based optimization methods focused on solving the problem:

$$\max f(x) \quad x \in A \qquad \textit{Equation 1}$$

where the feasible set and objective function typically have the following properties:

- The input x is in $\mathbb{R}^d$ for a value of d that is not too large. Typically, $d \leq 20$ in most successful applications of BO.
- The objective function f is continuous. This will typically be required to model f using Gaussian process regression.
- f is "expensive to evaluate" in the sense that the number of evaluations that may be performed is limited, typically to a few hundred. This limitation typically arises because each evaluation takes a substantial amount of time (typically hours) but may also occur because each evaluation bears a monetary cost (i.e.., from purchasing cloud computing power, or buying laboratory materials), or an opportunity cost (i.e.., if evaluating f requires asking a human subject question who will tolerate only a limited number).
- f lacks known special structure like concavity or linearity that would make it easy to optimize using techniques that leverage such structure to improve efficiency. We summarize this by saying f is a "Black box."
- When we evaluate f, we observe only $f(x)$ and no first- or second-order derivatives. This prevents the application of first- and second-order methods like gradient descent, Newton's method, or QuasiNewton methods. We refer to problems with this property as "derivative-free".
- Through most of this thesis, I will assume $f(x)$ is observed without noise.
- The focus is on finding a global rather than local optimum.

The ability to optimize expensive black-box derivative-free functions makes BO extremely versatile (Frazier 2018). It is important to note that the emphasis on maximizing rather than

minimizing a function is purely arbitrary. The two problems can be easily transformed into each other by simply negating the objective function (Garnett, Roman 2023).

input: initial dataset $\mathcal{D}$ ▸ can be empty
repeat
  $x \leftarrow \text{POLICY}(\mathcal{D})$ ▸ select the next observation location
  $y \leftarrow \text{OBSERVE}(x)$ ▸ observe at the chosen location
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x, y)\}$ ▸ update dataset
until termination condition reached ▸ e.g., budget exhausted
return $\mathcal{D}$

Figure 1- Sequential optimization algorithm (Garnett, Roman 2023)

1.2. How Bayesian Optimization Works?

Bayesian Optimization (BO) employs a surrogate model, known as a probability model or surrogate of the objective function, to approximate the true shape of the function. In an ideal scenario with unlimited resources, every point of the objective function would be computed to gain knowledge of its exact shape. However, BO utilizes a finite number of initial objective function evaluations to infer and construct the probabilistic model. This surrogate model serves as an approximation for the objective function.

The subsequent step involves selecting another sample from the objective function. To accomplish this, various acquisition functions are utilized within BO. These acquisition functions aid in determining the most promising location to sample the objective function for the next iteration. The aforementioned steps, including surrogate model construction, sample selection using acquisition functions, and subsequent objective function evaluations, are iteratively repeated until the maximum number of iterations is reached.

To make long story short, BO consists of below two main steps:

1. Choose some initial values.

2. Using acquisition function, choose one new control vector x^{new} and evaluate $f(x^{new})$, update the probabilistic model and iterate until stopping rule.

Place a Gaussian process prior on f
Observe f at n_0 points according to an initial space-filling experimental design. Set $n = n_0$.
while $n \leq N$ **do**
  Update the posterior probability distribution on f using all available data
  Let x_n be a maximizer of the acquisition function over x, where the acquisition function is computed using the current posterior distribution.
  Observe $y_n = f(x_n)$.
  Increment n
end while
Return a solution: either the point evaluated with the largest $f(x)$, or the point with the largest posterior mean.

Figure 2- Bayesian optimization pseudo-code

1.2.1. Acquisition Function

Acquisition functions play a crucial role in proposing sampling points within the search space. These functions aim to strike a balance between exploitation and exploration. Exploitation involves selecting sampling points where the surrogate model predicts a high objective value, while exploration involves sampling at locations where there is high uncertainty in the predictions. Both exploitation and exploration correspond to high values of the acquisition function. The objective is to maximize the acquisition function in order to identify the next sampling point (Krasser 2018).

As previously mentioned, there exists a wide range of available acquisition functions to choose from. In this thesis, the Expected Improvement (EI) acquisition function, which is widely recognized as one of the most commonly used acquisition functions, was employed. This choice was made due to its ability to be evaluated analytically under the Gaussian Process (GP) model.

1.2.2. Gaussian Processes

The majority of the research papers use the Gaussian Process model as the surrogate model for its simplicity and ease of optimization (Wei 2020).

A Gaussian Process (GP) can be considered as an extension of the Gaussian distributions into functional spaces. The key assumption in GP is that the function values at a collection of $M > 0$ inputs, $f_X = [\, f(x_1), \dots, f(x_M)]$, jointly follow a Multivariate Normal distribution (MVN), with a mean $\mathbb{E}[f(x_i)] = m(x_i)$ and covariance $Cov\left[f(x_i), f\left(x_j\right)\right] = k(x_i\,, x_j)$.

The function $m(x_i)$ represents the mean which encodes our prior knowledge of the value if $f(x_i)$. Although we use the term "mean function" $m(.)$, we eventually end up with a vector $\mu = [m(x_1), \dots, m(x_M)]$. The Covariance matrix is an N x M matrix where each element i, j is defined by the covariance function $k(x_i\,, x_j)$. Covariance offers valuable insight into the relationship between two variables. It helps to understand how these variables are related and provides information about the direction and strength of their association. More precisely, covariance is a statistical measure that quantifies the degree to which two random variables in a dataset vary together. It provides information about the extent to which changes in one variable correspond to changes in another variable. A positive covariance indicates that the variables tend to move in the same direction, while a negative covariance suggests they move in opposite directions. The magnitude of the covariance value indicates the strength of the relationship between the variables.

$$k\left(x_i\,, x_j\right) = \begin{pmatrix} k(x_1\,, x_2) & \cdots & k(x_1\,, x_M) \\ \vdots & \ddots & \vdots \\ k(x_N\,, x_1) & \cdots & k(x_N\,, x_M) \end{pmatrix} \qquad \textit{Equation 2}$$

The Gaussian Process (GP) is defined as a probability distribution over a finite number of M points, which is completely specified by its mean vector and covariance matrix. There are many choices for the covariance function. In this thesis, the RBF (Radial Basis Function) covariance function was utilized. The RBF covariance function, also known as the squared exponential or Gaussian covariance function, is a commonly used kernel in Gaussian Process models. It is characterized by its smoothness and flexibility. The RBF covariance function calculates the similarity or correlation between two points in the input space based on their Euclidean distance. It assigns higher values for points that are closer together and lower values for points that are farther apart.

1.3. What is Net Present Value?

Net Present Value (NPV) is a financial metric that measures the difference between the present value of expected cash inflows and the present value of expected cash outflows over a specific time period. NPV is commonly employed in capital budgeting and investment analysis to assess the profitability and viability of a projected investment or project. By discounting future cash flows to their present value, NPV takes into account the time value of money and provides a quantitative measure of the net financial impact of an investment after considering its initial cost and expected returns.

NPV is derived through computations that determine the present value of future cash flows by applying an appropriate discount rate. In essence, it quantifies the current value of an anticipated series of payments. As a general principle, projects with a positive NPV are considered financially viable and worthwhile, whereas those with a negative NPV are not.

For an oil company, the objective is to enhance shareholder or stakeholder value by maximizing the net present value (NPV) of its activities. This entails identifying and pursuing projects or investments that yield positive NPV, as these are expected to contribute positively to the company's financial performance and overall value. By evaluating potential projects based on their NPV, oil companies can make informed decisions regarding resource allocation and prioritize initiatives that offer the greatest financial returns.

1.4. What is Expected Net Present Value?

Expected Net Present Value (ENPV) is a technique that addresses uncertainty by incorporating multiple scenarios and assigning probabilities to calculate the most probable NPV. This approach involves determining the net present value under different potential scenarios and weighting them based on their likelihood. By considering the inherent uncertainty associated with projecting future scenarios, Expected NPV provides a more reliable estimate compared to traditional NPV.

ENPV takes into account the range of possible outcomes and their associated probabilities, offering a more comprehensive assessment of the potential value of an investment or project. This technique allows decision-makers to incorporate risk considerations into their evaluation process and make informed decisions based on a more realistic and probabilistic estimation of NPV. By factoring in uncertainty, ENPV provides a more robust framework for analyzing investments and mitigating potential risks.

1.5. Thesis Goals

1. Investigating the impact of initial values on the convergence of Bayesian Optimization (BO).
2. Optimizing the maximization of Expected Net Present Value (ENPV) for an oil field using BO techniques.
3. Exploring how the selection of initial values affects computation time and ENPV outcomes.

These goals serve as the driving force behind the research and provide a clear framework for the subsequent sections and analysis conducted in the thesis.

1.6. Thesis Structure

Section 2 of this thesis provides a concise background to contextualize the study. It offers relevant information and prior research related to the topic at hand, establishing the foundation for the subsequent sections.

In Section 3, the problem statement and the main purpose of the thesis are clearly stated. The implementation of Bayesian Optimization (BO) is introduced, initially demonstrated on a 1D toy problem to illustrate its functionality. Subsequently, BO is applied to a field-scale problem consisting of 10 realizations. The methods employed are explained in detail, and various scenarios are tested and analyzed. The achieved results are compared and discussed, highlighting the strengths and limitations of the implemented BO approach.

Section 4, the conclusion, addresses the goals outlined in Section 1.5. It summarizes the key findings, discusses the implications of the results, and provides insights into the effectiveness and applicability of BO in solving the considered problem. The conclusion serves to reinforce the significance of the study and offers suggestions for future research directions.

All Python codes utilized in the thesis are included in the Appendices as well as my Github repository[1], allowing readers to refer to the implemented algorithms and reproduce the experiments conducted throughout the research.

[1] https://github.com/MinaHosseini66/Bayesian-Optimization

2. Literature Review

The literature on well-control optimization can be divided into two categories: gradient-based and gradient-free.

Van Essen et al. (2009) used an adjoint-based method to estimate gradients of an objective function of control variables and optimize hydrocarbon production under geological uncertainty. The adjoint-based method requires access to the source code of the reservoir simulator, which is seldom available for commercial simulators, and deriving exact gradients from the simulator is computationally demanding (Kor, Hong, and Bratvold, n.d.).

Chen et al. (2009) introduced the ensemble-based optimization method (EnOpt), in which the covariance between objective values and the values of a control variable approximates a gradient.

Fonseca et al. (2015) proposed a modified EnOpt formulation, and Fonseca et al. (2017) demonstrated that the modified EnOpt formulation gives more accurate gradient approximations than the original EnOpt formulation proposed by Chen et al. (2009).

Harding et al. (1998) compared different gradient-free methods and showed that the genetic algorithm (GA) outperforms the simulated annealing algorithm (SA), the sequential quadratic programming (SQP), and a hybrid of SA and SQP.

Oliviera et al. (2019) proposed an upper confidence bound (UCB) algorithm for BO problems where both the outcome of a query and the true query location are uncertain. The algorithm employs a Gaussian process model that takes probability distributions as inputs.

Peyman et al. (2023) introduced and discussed a new optimization workflow by providing computational efficiency, i.e., achieving a near-global optimum of the predefined objective function with minimal forward model (flow-simulation) evaluations. He finally compared the workflow with two other gradient-free algorithms, Particle Swarm Optimization (PSO) and Genetic Algorithm (GA).

3. Problem Statement

As mentioned in the portion dedicated to Bayesian Optimization within the Introduction section, production optimization is defined as a search for an optimum value (maximum or minimum) of a well-behaved objective function f by tuning control variable x. The mathematical expression of the optimization is $f: X \rightarrow \mathbb{R}$ where X denotes the feasible domain of x and has a dimension of $D(X \subseteq \mathbb{R}^D)$:

$$\max f(x) \quad x \in A$$

In the context of oil and gas decision making, assuming our knowledge of the world to be "deterministic" is generally a modeling choice- not a feature of reality. In rare circumstances, it may be a reasonable approximation (Bratvold and Begg 2010).

The objective of this thesis is to optimize the Expected Net Present Value (ENPV) while minimizing the computational effort required. The focus is on developing efficient approaches that reduce the number of calculations needed to obtain the ENPV, thereby saving time and computational resources. By streamlining the optimization process, the thesis aims to provide practical methods that enable decision-makers to make informed choices in a timely manner, without compromising the accuracy and reliability of the ENPV estimation. In this thesis, only the geological uncertainties are considered. Initially, the analysis begins with a deterministic case where the geological uncertainties are not taken into account. First, The optimization process is performed based on a single geological model, assuming a known and fixed subsurface condition, then geological uncertainties and their impact on the optimization results are included. By considering these uncertainties, the research aims to enhance the accuracy and robustness of the optimization process and provide a more realistic assessment of the expected net present value (ENPV) of the oil field or reservoir.

In deterministic case, the objective function (NPV) looks like below:

$$f(x,G) = \sum_{k=1}^{K} \left([\sum_{j=1}^{N_p} p_o q_{o,j,k}(x,G) - \sum_{j=1}^{N_p} p_{wq} q_{wp,j,k}(x,G) - \sum_{i=1}^{N_{wi}} p_{wi} q_{wi,i,k}(x,G)] \frac{\Delta t_k}{(1+b)^{\frac{t_k}{\tau}}} \right)$$

Equation 3

Where G denotes a specific geological model, K is the number of timesteps, N_p is the number of production wells subject to optimization, N_{wi} is the number of water injection wells subject to optimization, k is the timestep index, j is the production well index, i is the water injection well index, p_o is the selling price of the produced oil in USD/m^3 , p_{wp} is the cost of produced water in USD/m^3 , p_{wi} is the cost of injected water in USD/m^3 , q_o is oil production rate in m^3/day , q_{wp} is the water production rate in m^3/day , q_{wi} is the water-injection rate in m^3/day, Δt_k is the time interval in years of time step k, b is the discount rate which is dimensionless, t_k is the cumulative time in years for discounting at timestep k, and τ is the reference time for discounting ($\tau = 1$ year if b is expressed as a fraction per year and cash flows are discounted yearly), and x is the control value defined as $x = [x_1 + x_2 + \cdots + x_D]^T$ where x_d is the dth control variable and D is the number of control variables which is the dimension of optimization.

Definition	Symbol	Value	Unit
Oil price	p_o	300	USD/m^3
Water production cost	p_{wp}	47.5	USD/m^3
Water injection cost	p_{wi}	12.5	USD/m^3
Discount rate	b	8% (yearly)	-
Fraction per year	τ	1	year
Study period	K	10	years
Number of realizations	N_e	10	-

Table 1- parameters for NPV calculations

The objective function defined in (3) is used for a deterministic case since it does not capture geological uncertainties. Equation (4) is the expected form of equation (3), and it considers uncertainties in geological aspect:

$$\mathbb{E}_G[f(x,G)] = \int_\Omega f(x,G)p(G)d(G) \qquad \textit{Equation 4}$$

Where $\mathbb{E}_G$ denotes the expectation over G, $p(G)$ is the possibility density function (PDF) of the uncertain geological variables in G, and Ω is the feasible domain of G. A common practice is that the geological uncertainties are represented by an ensemble of N_e equiprobable geological realizations, G for $r = 1,2,\dots,N_e$ sampled from the PDF $p(G)$. There for, the approximation of $\mathbb{E}_G[f(x,G)]$ is:

$$\mathbb{E}_G[f(x,G)] \approx \overline{f}\,(x) = \frac{\sum_{r=1}^{N_e} f(x,G_r)}{N_e} \qquad \textit{Equation 5}$$

The main objective of this thesis is to optimize the function $\overline{f}\,(x)$ while concurrently minimizing the time required for the optimization process. To achieve this, the research introduces and implements an eliminating criterion, which serves as a filtering mechanism to expedite the optimization procedure.

The optimization of the aforementioned function poses several challenges for several reasons:

- Absence of an analytic form due to the use of a black box simulator. Without an explicit mathematical expression for the objective function, traditional gradient-based optimization methods cannot be directly applied. Gradient-based methods rely on calculating derivatives of the objective function, which is not feasible in this case.
- Finding the global maximum of the function poses a challenge due to the presence of numerous local optima. Local optima are points in the search space where the objective function reaches a relatively high value but may not be the global maximum. The presence of multiple local optima makes it difficult to guarantee that the global maximum will be discovered, especially when using optimization algorithms that can get trapped in local optima.
- The evaluation of the objective function requires significant computational resources and time. This can pose challenges in terms of the efficiency and speed of the optimization process.

The challenges associated with the function optimization, such as the absence of an analytic form, the presence of multiple local optima, and the computational demand, provide strong motivation for exploring alternative methods that can effectively overcome these obstacles. Bayesian

Optimization has emerged as a promising approach that has demonstrated its effectiveness in addressing such challenges, making it a suitable candidate for consideration in this thesis.

3.1. Implementation of BO on 1D Toy Problem

In this section, the focus is on providing a clear and detailed demonstration of how Bayesian Optimization (BO) functions in practice. To facilitate this, a simple 1D problem is selected as an illustrative example. While the problem is limited to one dimension, it should be noted that the workflow can be expanded and applied to higher-dimensional problems[2]. The true function subject to optimize looks like below:

$$Maximize_x \quad f(x) = 1 - \frac{1}{2}\left(\frac{sin(12x)}{1+x} + 2\,cos(7x)\,x^5 + 0.7\right) \qquad bound: 0 < x < 1 \qquad \textit{Equation 6}$$

The global maxima of above function can be calculated. The $f(x)$ is maximized when $x = 0.39$. It is also shown in figure 3 below. This function has a local maximum around $x = 0.82$. The intention of using this function is to show that BO can avoid local maximum and reach the global maximum.

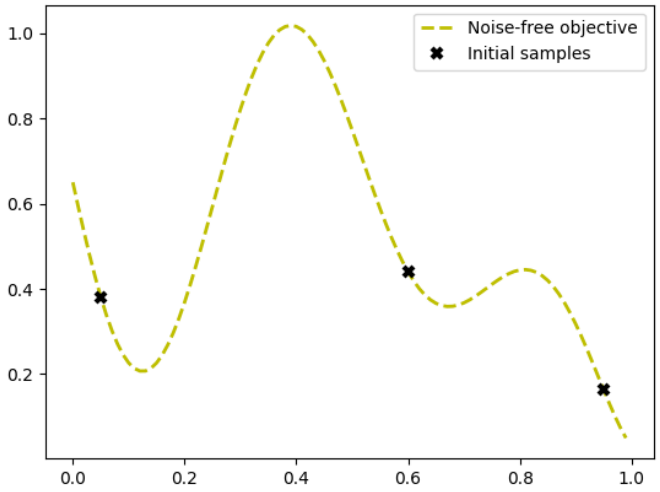


Figure 3-The true visualization of the equation (6)

[2] The codes related to this part can be found in Appendix 1

Although there exists an analytical expression for the optimization objective function f, it is treated as a black box. This means that the internal workings of the function are unknown, and we rely solely on its input-output behavior. The goal is to iteratively approximate the objective function using a Gaussian Process within the framework of Bayesian Optimization.

During the optimization process, the objective function f will be evaluated at different points within a specified bound. The bound defines the range of values over which the optimization will be performed. By exploring this bounded space and observing the function values at various points, the Gaussian Process surrogate model is constructed and updated iteratively.

We have a set of initial values available. A set of $[x, f(x)]$ which is shown by D:

$$D = [x, f(x)] = ([0.05,0.5,0.95], [0.38,0.77,0.16]) \qquad \textit{Equation 7}$$

Bayesian Optimization is run for 10 iterations. In each iteration, a row with two plots is generated. A plot of the objective function and the surrogate function which is the GP posterior predictive mean with 95% confidence interval from the mean. The other plot is acquisition function which helps with the selection of the next proposed sampling x. Next sampling point is where the acquisition function is maximized, and it is shown with dashed vertical line.

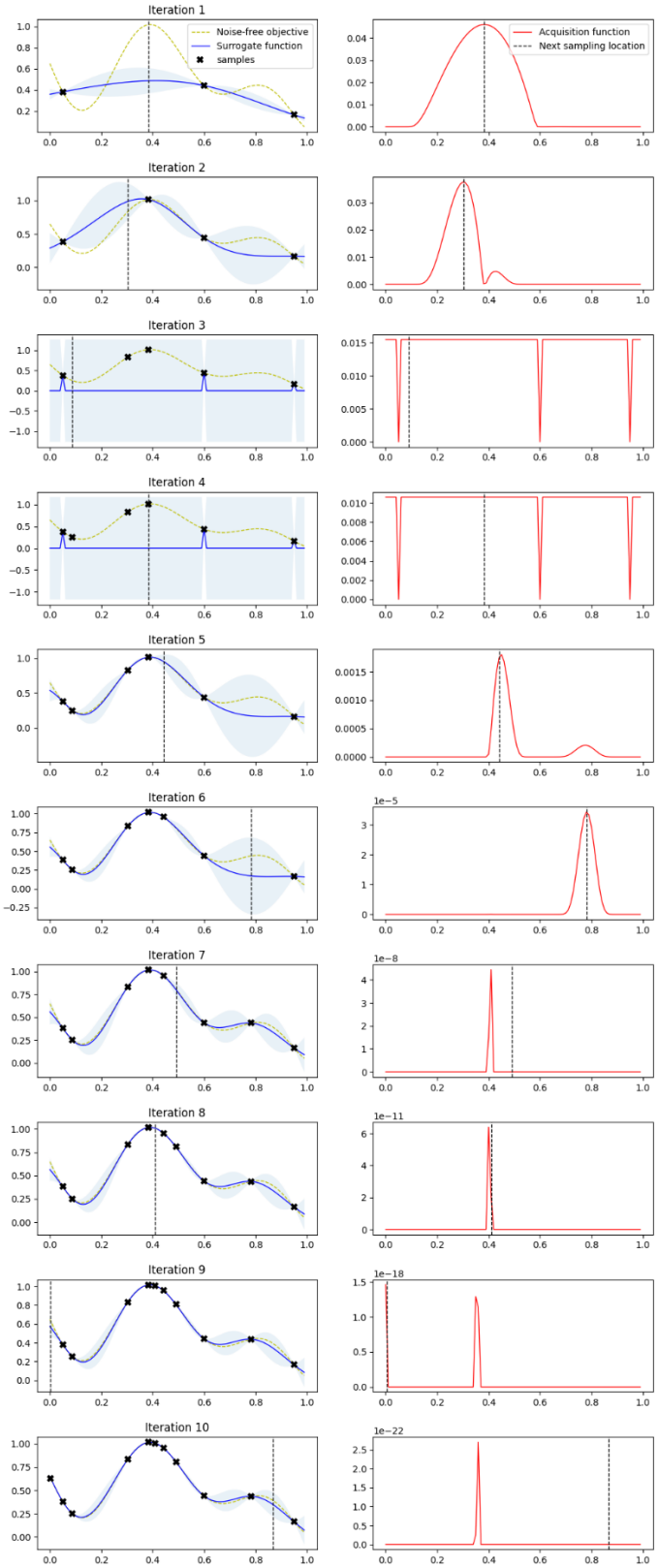


Figure 4- Bayesian Optimization flow for equation (6)

Finally, BO has suggested the best point which maximizes the objective function to be $x = 0.39$.

We showed that function f will be maximized at point $x = 0.39$. The accuracy of BO in determining this optimal solution is 100%.

It is worth noting that exploration helps BO not to be trapped in the local optimum point. Where the uncertainty is high, BO tried to reduce the uncertainty by exploring new points, where uncertainty is low and optimum result has produced, BO tries to exploit and search for more optimum results.

In the convergence plot below, we can see how many iterations are needed to achieve the optimum value. We can see that we are able to catch the global maximum with fewer iterations which is shown in Figure 5.

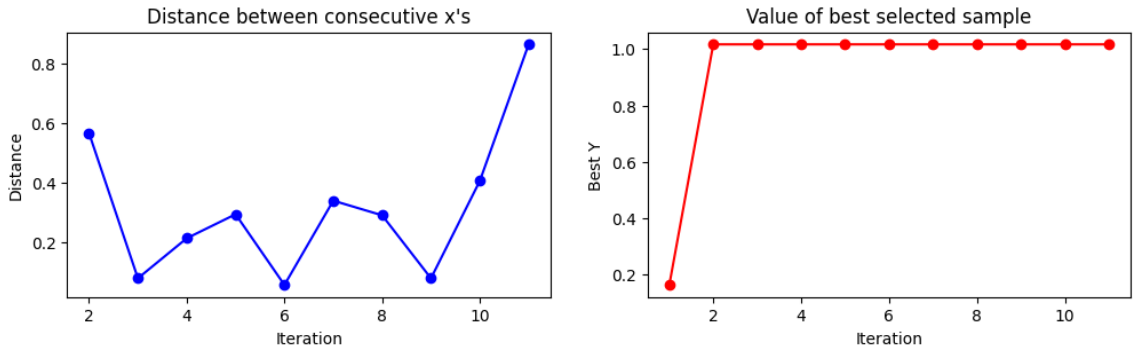


Figure 5- Convergence of BO for 1D toy problem

In this example, only 3 initial values were considered, it can be shown that if we increase the number of scattered initial values, the BO will converge faster. This will be further illustrated in the field scale analysis, where we will explore the impact of a larger set of initial values on the convergence of BO.

3.2. Implementation of BO on Field Scale Problem

In this section, Bayesian Optimization (BO) is applied to a 3D reservoir simulation model with the objective of maximizing the Expected Net Present Value (ENPV) of a synthetic, water flooded oil

field. The focus is on controlling the water injection rates in order to optimize the field's performance while minimizing the simulation time. In (Kor, Hong, and Bratvold, n.d.) Peyman has calculated ENPV in his paper (Base case) and in this thesis (Modified method) I tried to reduce the amount of time spent in finding the global maximum (reducing calculation time). Base case serves as the benchmark or reference point for evaluating the performance of the modified method proposed in this thesis. The objective of this thesis is to optimize the calculation time required to find the global maximum of the ENPV.

The synthetic field, known as the “Egg Model” has a geology of channelized depositional system. Ten realizations are used, we assume that all 10 realizations are equal-probable geological realizations. The only uncertainty considered in this model is the cell permeabilities. The model has 8 water injectors and four producers which is depicted in figure 6.

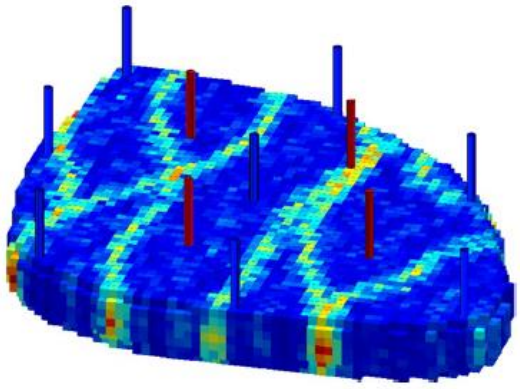


Figure 6- model of the reservoir

The objective is to maximize the ENPV by controlling the water injection rates for each injector over a production period of 10 years. There are 8 control variables for 8 wells in the control vector.

In the calculation of ENPV, Peyman took into account the NPV of all realizations corresponding to the injection rates selected by BO. This means that for each set of injection rates suggested by the optimization process, the NPV was computed for each individual realization. By calculating the NPV of the first realization, it is possible to gain initial insights into the potential value of the selected injection rates. If the NPV of the first realization is relatively low, it may indicate that the chosen injection rates are not optimal and may not yield desirable financial outcomes. This early

assessment can be used to make informed decisions and potentially explore alternative injection rate configurations.

In this thesis, I propose a modified method to expedite the calculation of NPVs for the injection rates by incorporating a criterion based on the NPV of the first realization. This criterion allows for the identification of injection rate configurations that are unlikely to yield high NPVs, thereby reducing the need to compute NPVs for the remaining nine realizations.

Two criteria that can be presented for this purpose are:

1. Maximum ENPV criterion: The NPV calculated from the first realization should not be lower than the maximum ENPV obtained from the initial values. If the NPV of the first realization falls below this threshold, it indicates that the corresponding injection rates are unlikely to result in a favorable financial outcome, and further computation of NPVs for the remaining realizations can be skipped.

2. Average ENPV criterion: The NPV calculated from the first realization should not be lower than the average ENPV obtained from the initial values. If the NPV of the first realization is below this average value, it suggests that the selected injection rates may not yield competitive financial results compared to the overall set of initial values. In this case, the computation of NPVs for the remaining realizations can be bypassed.

By implementing these criteria, I aim to significantly reduce the computational time required to assess the NPVs of the injection rate configurations. This approach takes advantage of the observation that the NPV of the first realization provides valuable information about the likelihood of obtaining a high NPV outcome for the entire set of realizations. By skipping the computation of NPVs for unpromising injection rate configurations, the overall time spent on the analysis is substantially reduced.

The results of my testing indicate that this modified method effectively reduces the computational time while still capturing injection rate configurations that yield competitive ENPV. This not only enhances the efficiency of the optimization process but also allows for quicker decision-making and resource allocation in oil and gas projects.

I implemented Bayesian Optimization (BO) with two different scenarios for the injection rates in an oil field:

Same injection rate for all 8 wells: In this scenario, I considered a uniform injection rate for all wells in the oil field. This approach simplifies the optimization process as it involves finding the optimal single injection rate that maximizes the ENPV across all wells. Uniform distribution is used to select initial value injection rate(s) samples.

Different injection rates for each well: In this scenario, I allowed for variation in the injection rates across the 8 wells. This introduces additional complexity to the optimization process, as it requires finding the optimal combination of injection rates for each individual well that collectively maximizes the ENPV. Two different approaches were utilized for selecting the initial injection rates and results compared.

3.2.1. Scenario 1 - Same injection rate for all wells[3]

This particular scenario represents a simplified iteration of the overarching scenario, wherein distinct injection rates are assigned to individual wells. Through this simplified version, the objective is to illustrate the extent of process time reduction achieved by employing the modified method, as well as to investigate the influence of the number of initial values on the early convergence of Bayesian Optimization (BO).

3.2.1.1. Same initial values for both methods (Base case and Modified method)

Assumptions: Two initial values, 10 BO iterations with seed[4]

Figure 7 presents the outcome of the analysis. It demonstrates a significant 20% reduction in computation time achieved through the modified method. Furthermore, the optimal injection rate is observed to be approximately 56 m3.

The concept of convergence cannot be addressed in this context as the use of a seed ensured that both methods achieved equal convergence. Figure 8 illustrates the convergence of both methods, indicating that they both reached convergence after the 6th iteration.

[3] Codes for this part can be found in Appendix 2

[4] With seed, the selected injection rate in each iteration is same for both methods. I used seed in order to be able to compare the calculation time for both methods.

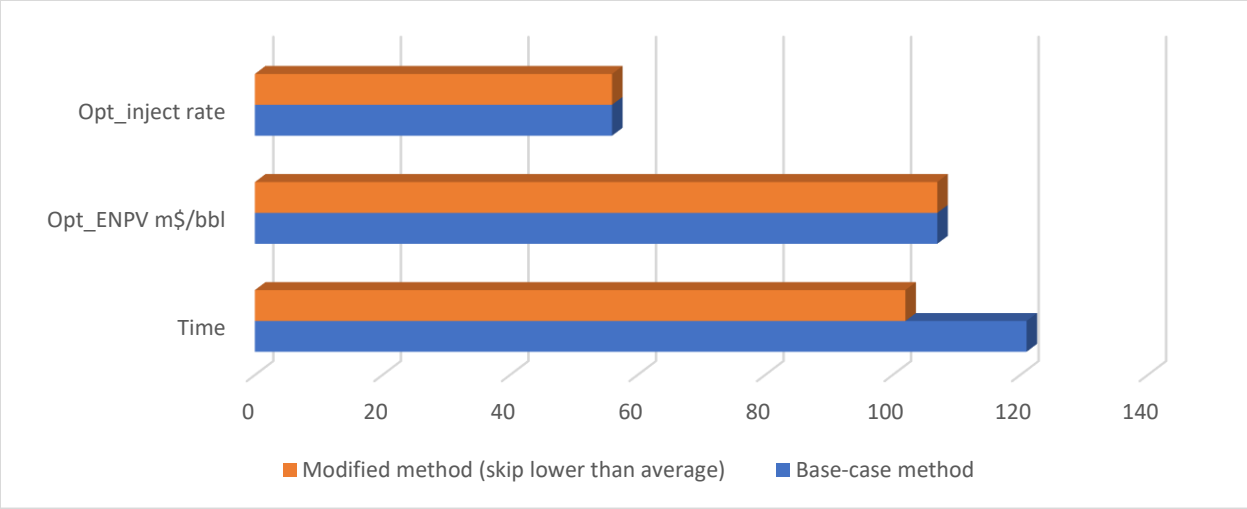


Figure 7- Comparison of the result of two methods with 2 initial values and 10 BO iteration with seed

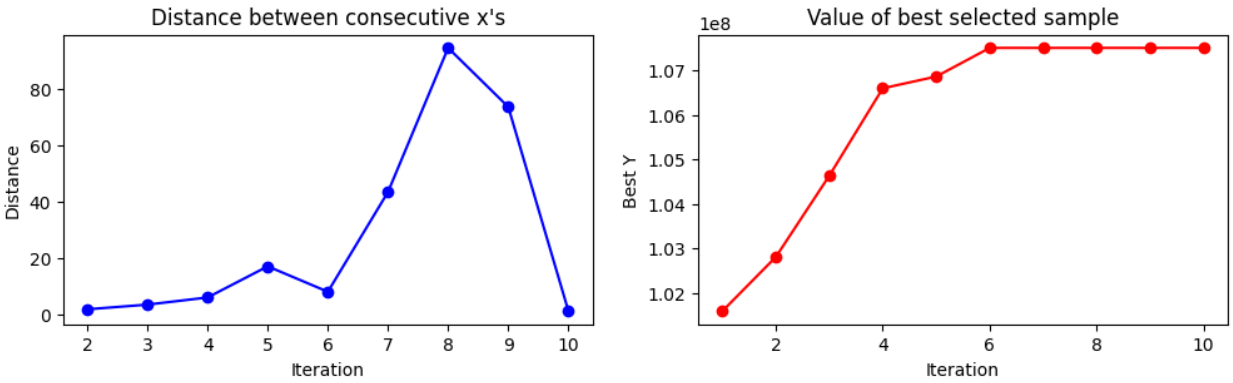


Figure 8- 2 initial values and 10 iterations, with seed - convergence plot

3.2.1.2. Modified methods: compare to max and compare to average.

Assumptions:

- Number of initial injection rates: 2 and 10
- Number of BO iterations: 10 without seed

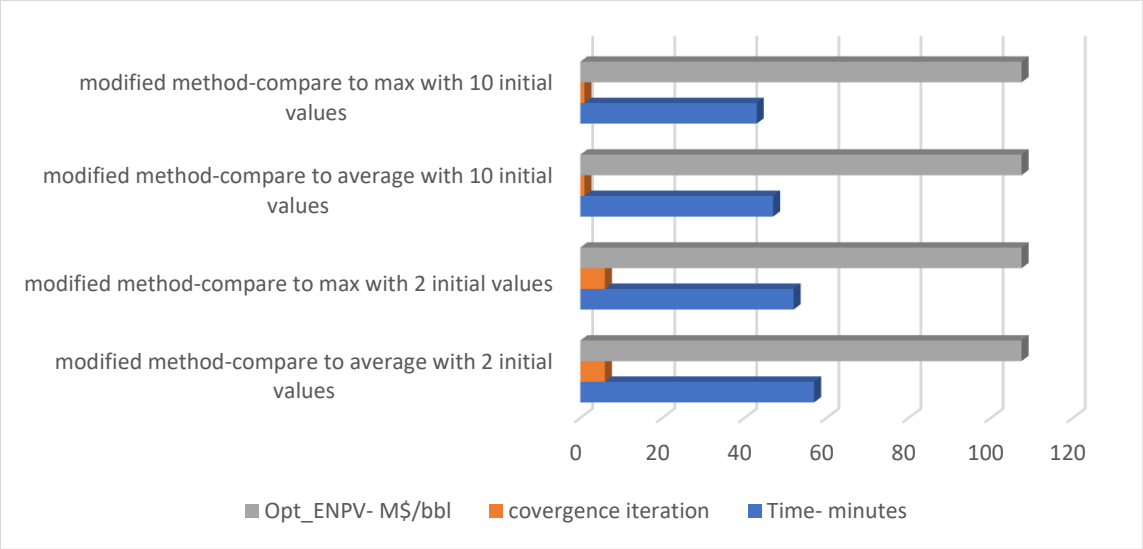


Figure 9- Compare the effect of increasing the number of initial values on time and convergence for 10 BO iterations.

Figure 9 provides evidence that increasing the number of initial values yields a positive effect, as it leads to earlier convergence of the Bayesian Optimization (BO) method. This observation is further supported by figures 10 and 11, which illustrate similar outcomes. Alternatively, employing the maximum ENPV of the initial values as a decision criterion to determine whether further NPV calculations are necessary after evaluating the NPV of the first realization can result in even greater reduction in computation time. This approach leverages the knowledge gained from the initial values to make informed decisions and optimize the calculation process.

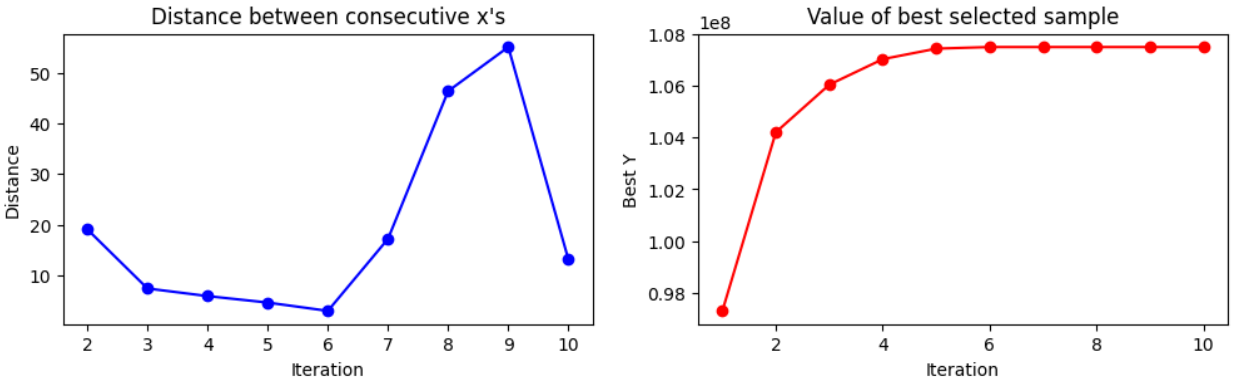


Figure 10- 2 initial values and 10 iterations, modified method convergence plot.

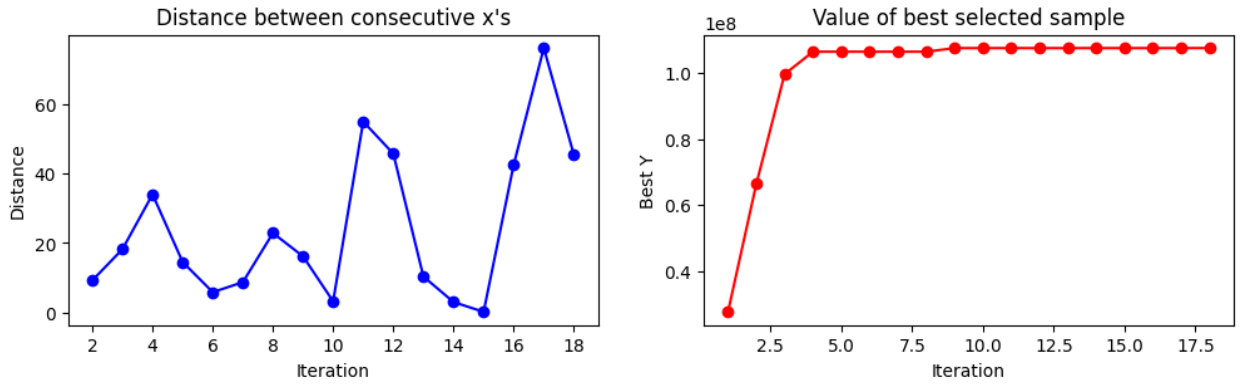


Figure 11- 10 initial values and 10 iterations, modified method convergence plot.

As mentioned earlier, the optimal injection rate when assuming the same injection rate for all wells is around 56 which corresponds to maximum ENPV. The figure 12 is shown below:

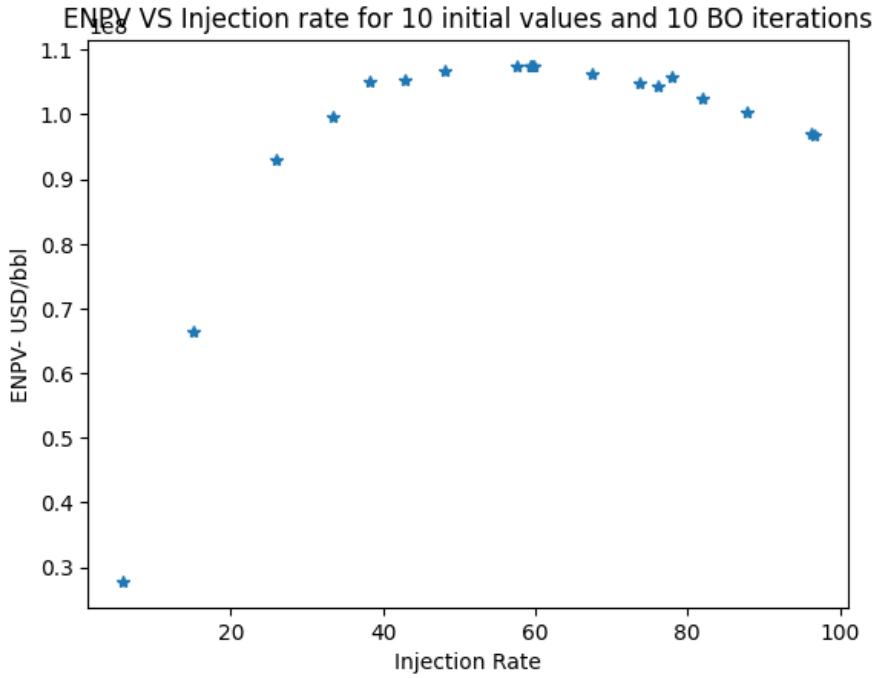


Figure 12- ENPV vs injection rate for 10 initial values and 10 iterations

Figure 13 depicts a comparison between the ENPV calculated from the initial values and the ENPV calculated using BO. It is evident that with the utilization of 10 initial values, BO achieved

convergence in the very first iteration, aligning with the findings presented in the convergence plot illustrated in figure 11.

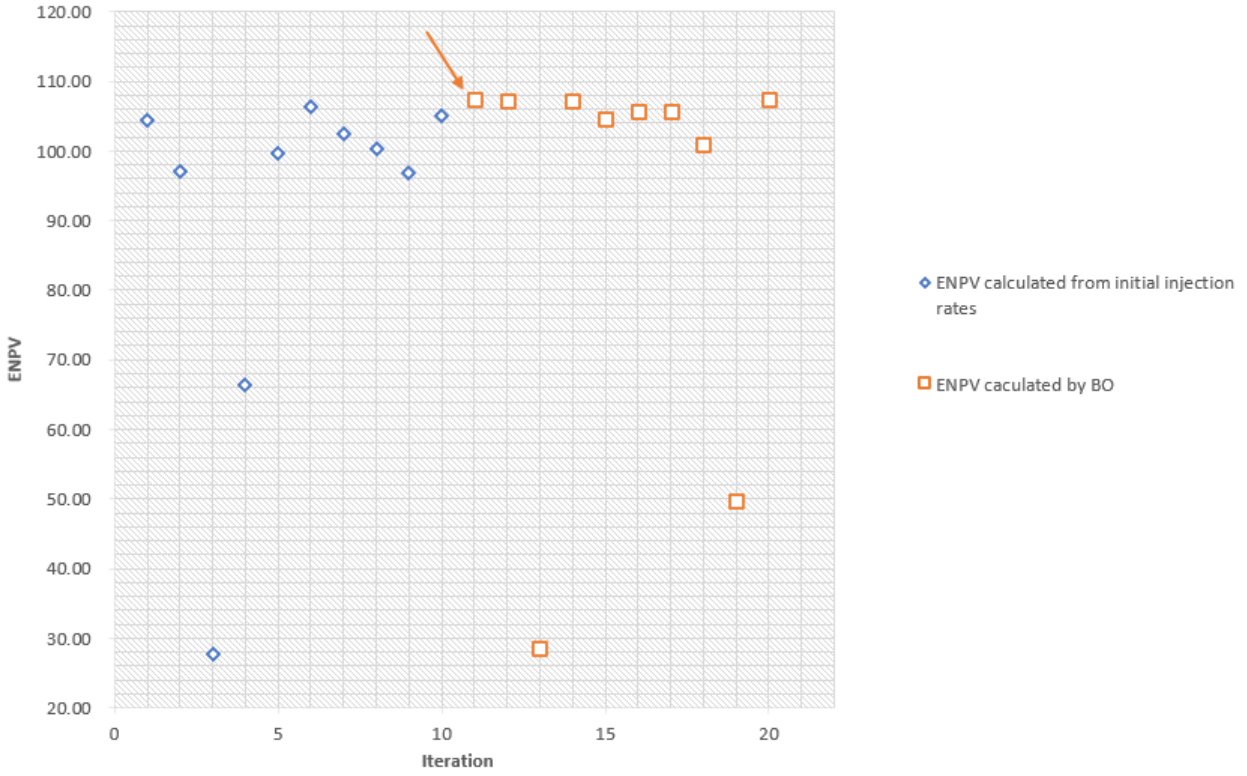


Figure 13- ENPV calculations for initial values and BO with modified method (compare to average)

3.2.2. Scenario 2 - Different injection rates for each well[5]

In this case, the input is an array with 8 elements. In the initial scenario, the application of the modified method demonstrated a notable reduction in calculation time compared to traditional approaches. Furthermore, it was observed that increasing the number of initial values led to earlier convergence in the optimization process. In this specific scenario, the objective is to investigate the impact of selecting different sets of initial value samples on various aspects such as computation time, ENPV, and convergence. The analysis solely focuses on the implementation of the modified method, which has proven to be effective in reducing

[5] The codes related to this part can be found in Appendix 3

computation time. To conduct this investigation, a total of 30 sets of initial injection rates are carefully selected as samples. These samples serve as the starting points for the optimization process. By utilizing the modified method, the optimization algorithm iteratively refines and adjusts the injection rates over a set number of iterations, which in this case is set to 20. The primary goal of this analysis is to evaluate how the selection of different initial value samples influences the computational efficiency, measured in terms of computation time, as well as the resulting ENPV and the convergence behavior of the optimization process. By comparing and contrasting the outcomes obtained from the various sets of initial values, researchers can gain insights into the impact of initial value selection on the overall optimization performance.

Two different initial values selection methods are utilized:

1) The selection of initial value samples is carried out using a uniform distribution. A total of 30 random samples are generated, and this process is repeated 8 times to account for the presence of 8 wells in the oil field.
2) The Latin Hypercube Sampling (LHS) method is employed for the selection of initial value samples. The LHS method ensures a stratified and well-distributed sampling approach within the parameter space[6]. To apply the LHS method, a total of 8 random samples are selected, and this process is repeated 30 times. Each repetition generates a set of initial value samples, resulting in a diverse collection of 30 sets of initial values.

By employing the LHS method, each time the initial injection rates for the different wells are scattered in a stratified manner across the parameter space. This ensures that the samples obtained are more evenly distributed and representative of the possible range of injection rates for each well.

In contrast, using a uniform distribution does not guarantee the same level of scattering for the initial injection rates. While using a uniform distribution one can still generate random samples, they may not exhibit the same level of stratification and scattering as with LHS. This can lead to potential biases in the distribution of initial injection rates for the wells, potentially missing out on certain regions of the parameter space.

[6] The codes for this part can be found in Appendix 4

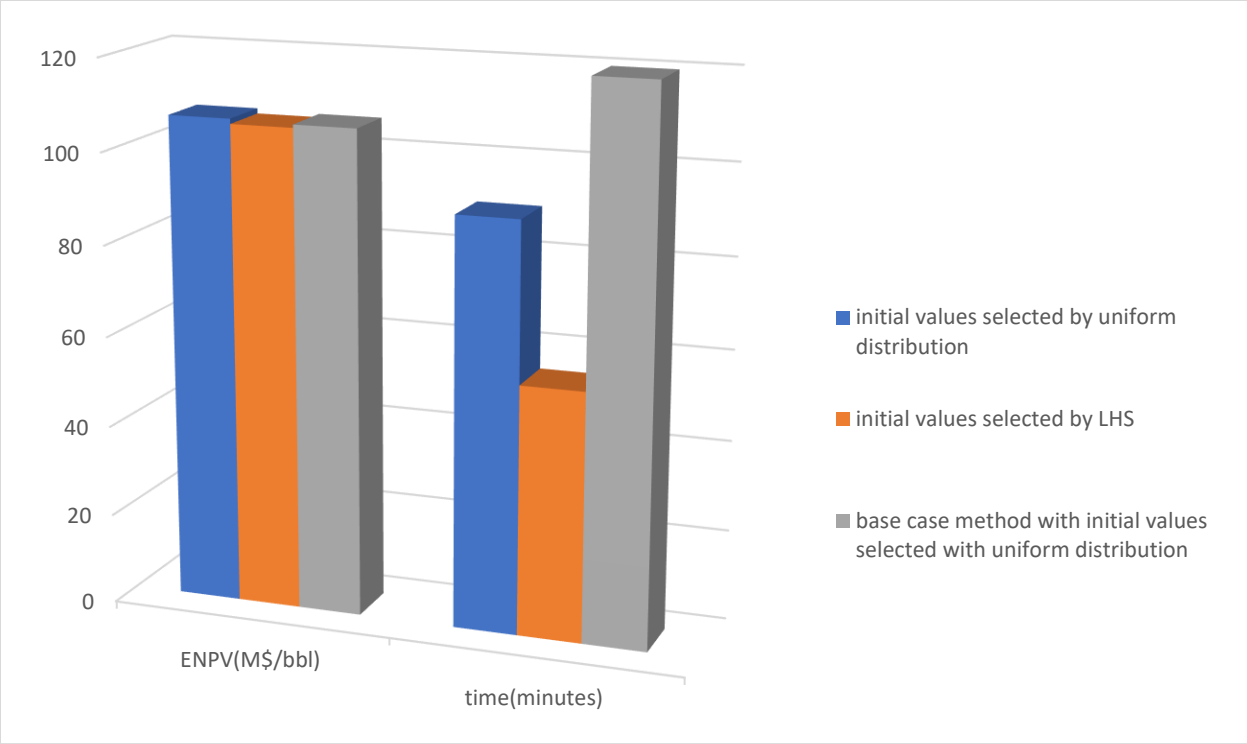


Figure 14- Comparison of ENPV and computation time for different initial value selection approaches

The observation from Figure 14 indicates that when the initial injection rate samples are selected using the Latin Hypercube Sampling (LHS) method, a notable reduction in computation time is achieved compared to the alternative initial value selection method. Additionally, both initial value selection methods lead to similar Expected Net Present Value (ENPV) outcomes. Overall, the results from Figure 14 demonstrate the advantage of using the LHS method in terms of computational efficiency, while highlighting the robustness of the optimization process in delivering similar ENPV outcomes regardless of the initial value selection method

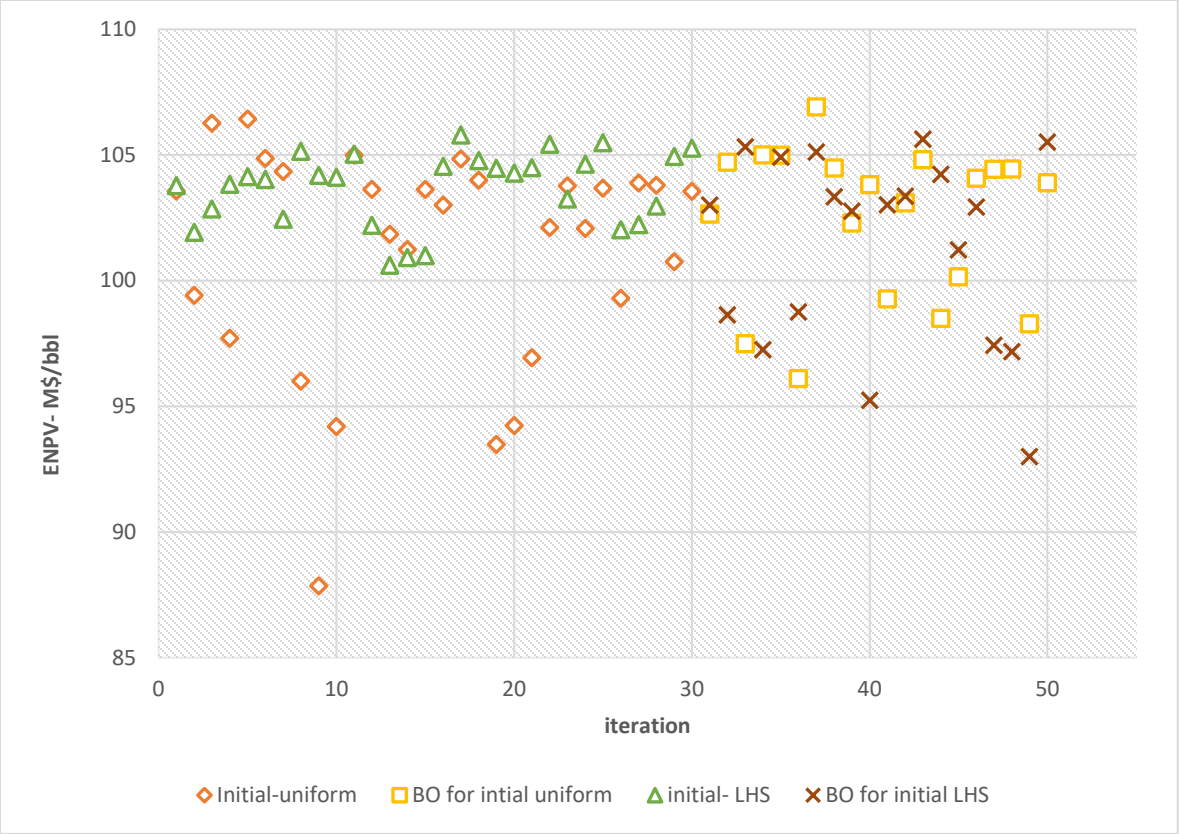
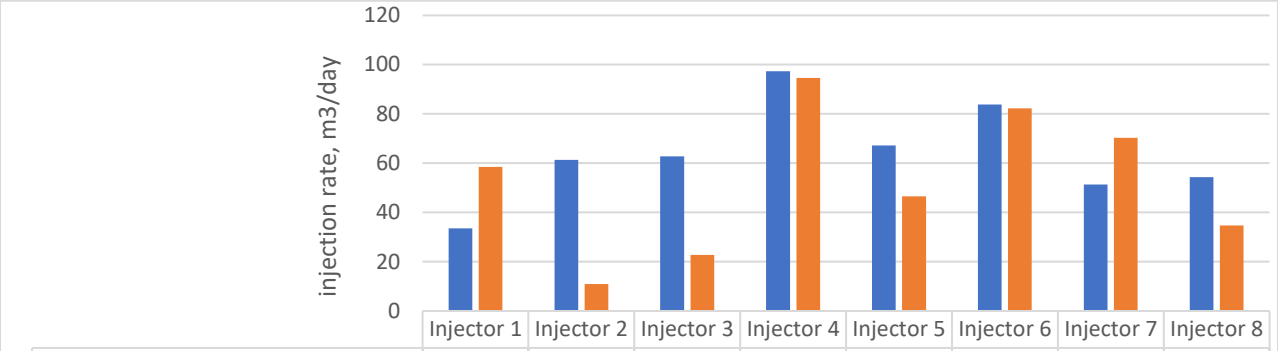


Figure 150- ENPVs calculated for initial injection rates and BO.

Figure 15 provides a graphical representation of the ENPVs calculated for the initial injection rates selected by two different selection methods well as the ENPVs calculated at each iteration of the Bayesian Optimization (BO) process.

When the initial injection rates are selected using a uniform distribution, the Bayesian Optimization (BO) process was able to achieve a higher Expected Net Present Value (ENPV) in the 37th iteration compared to the initial injection rates. This suggests that the exploration of the parameter space led to the discovery of more favorable injection rate combinations, resulting in increased profitability. On the other hand, when the initial injection rates are selected using the Latin Hypercube Sampling (LHS) method, the BO process did not surpass the ENPV calculated from the initial injection rates. Despite this, it is worth noting that the ENPV values obtained using both sampling methods were very close to each other. Additionally, they were also close to the ENPV calculated when the same injection rate was considered for all wells. This indicates that the

optimization process was able to identify injection rate combinations that yielded comparable profitability outcomes, regardless of the initial value selection method.



	Injector 1	Injector 2	Injector 3	Injector 4	Injector 5	Injector 6	Injector 7	Injector 8
optimum injection rate for each well (m3/day)	33.5	61.41	62.75	97.33	67.22	83.78	51.4	54.36
optimum injection rate for each well (m3/day)	58.43	10.93	22.81	94.62	46.56	82.28	70.31	34.68

Figure 16- Optimal injection rate for each well obtained from two different methods.

Figure 16 displays the optimal injection rates for each well obtained when two different methods were employed for the initial injection rate selection. It provides a visual representation of the injection rate combinations that yielded the highest Expected Net Present Value (ENPV) for each well under each method.

The results obtained from this study indicate that the initial injection rate selection method has a notable impact on the workflow of Bayesian Optimization (BO). Specifically, it has been observed that the choice of initial value selection method can lead to a reduction in calculation time.

Comparing the modified method with the base case, it has been demonstrated that the modified method is more time-efficient. This time efficiency becomes particularly significant in scenarios that require a larger number of BO iterations. By implementing the modified method, the overall computational burden can be reduced, resulting in improved efficiency in the optimization process. The findings suggest that careful consideration of the initial injection rate selection method can lead to notable improvements in the performance of BO. The modified method, in particular, offers advantages in terms of time efficiency and computational resource utilization.

Selecting the first realization as the basis for deciding whether to calculate the NPV for the remaining realizations may not be an optimal approach, particularly when dealing with significant

uncertainties. In situations where the uncertainty is high, the first realization may be an outlier and not representative of the overall NPV distribution.

Based on the specific characteristics of my thesis case, where the uncertainty is not high and the realizations follow an almost uniform distribution, choosing the first realization as a decision point for NPV calculations can be considered reasonable. In situations where the uncertainty is low and the realizations exhibit a uniform distribution, the first realization can serve as a reasonable representation of the overall NPV trend. Since the distribution is relatively consistent across the realizations, the first realization can provide a reliable indication of whether subsequent NPV calculations are likely to meet the specified criterion.

4. Conclusion and Future Work

In conclusion, this thesis focused on the application of Bayesian Optimization (BO) for the optimization of Expected Net Present Value (ENPV) in a real field-scale problem with 10 geological model realizations. The main goals outlined in the first section of the thesis were addressed and several key findings were obtained.

Firstly, it was demonstrated that increasing the number of initial values can lead to earlier convergence of the BO algorithm. This allows for a reduction in the number of iterations required to find the maximum ENPV. Notably, this observation holds true for problems with one-dimensional inputs, while higher-dimensional problems may necessitate a larger number of iterations for convergence.

Additionally, the thesis highlighted the importance of implementing an eliminating criterion to determine whether to calculate NPV for all realizations or not. The criterion, such as the average of NPVs calculated from the initial injection rates, proved to be valuable in reducing calculation time. However, it was noted that the selection of the decision point for the criterion may vary depending on the geological realizations and the level of uncertainty.

Furthermore, it was observed that the method used for initial value selection had an impact on the results and computation time of BO. Specifically, Latin Hypercube Sampling (LHS) offers the advantage of generating well-spread points within the control spaces. This characteristic enables the initial model, typically a Gaussian Process, to gather more informative insights about the explored space. Consequently, LHS enhances the effectiveness of the initial model in capturing the characteristics and patterns present in the optimization problem.

Overall, this thesis successfully applied BO to the ENPV optimization problem in a field-scale analysis. The findings emphasized the importance of careful consideration in selecting initial values, implementing an eliminating criterion, and understanding the influence of uncertainty on decision points. These insights contribute to the broader understanding and application of BO in the context of oil and gas decision making.

Future work in this area can focus on several aspects to further enhance the application of Bayesian Optimization in optimizing Expected Net Present Value (ENPV) for oil and gas decision making, here are some potential directions for future research:

1. To calculate the NPV for each realization using the initial value and then employ fit distribution methods to identify the distribution that best represents the NPV variations. By analyzing the results, one can determine the distribution that closely matches the observed NPV distribution. Utilizing a maximum likelihood approach, candidate realizations can be selected based on this analysis, and NPV calculations can be performed exclusively for those chosen realizations. These NPV values can then be compared against a specified criterion to determine whether further NPV calculations are necessary or if BO should select a new injection rate. This proposed approach effectively accounts for the inherent uncertainty in the system by considering the full distribution of NPV values, thus offering a more comprehensive evaluation than solely relying on the results of the first realization.
2. To Explore the integration of machine learning techniques, such as neural networks or deep learning, with Bayesian Optimization. This integration can improve the surrogate model's accuracy and enable more efficient exploration of the objective function landscape.
3. Incorporating Value of Information analysis into the ENPV optimization process can provide a systematic framework for decision-makers to assess the value of acquiring additional data and guide their resource allocation strategies. Future work can focus on developing methodologies, algorithms, and case studies to effectively integrate VOI analysis with Bayesian Optimization for enhanced oil and gas decision-making.

References

Boender, C. G. E., and Jonas Mockus. 1991. “Bayesian Approach to Global Optimization--Theory and Applications.” *Mathematics of Computation* 56 (194): 878. https://doi.org/10.2307/2008419.

Bratvold, Reidar B., and Steve H. Begg. 2010. *Making Good Decisions*. Society of Petroleum EngineersRichardson, Texas, USA. https://doi.org/10.2118/9781555632588.

Brito, Daniel U. de, and Louis J. Durlofsky. 2020. “Well Control Optimization Using a Two-Step Surrogate Treatment.” *Journal of Petroleum Science and Engineering* 187 (April): 106565. https://doi.org/10.1016/j.petrol.2019.106565.

Frazier, Peter I. 2018. “A Tutorial on Bayesian Optimization.” arXiv. http://arxiv.org/abs/1807.02811.

Garnett, Roman. 2023. *Bayesian Optimization*. Cambridge University Press.

Kor, Peyman, Aojie Hong, and Reidar B. Bratvold. n.d. “Reservoir Production Management with Bayesian Optimization: Achieving Robust Results in a Fraction of the Time.” *Under Review*.

Krasser, Martin. 2018. “Bayesian Optimization.” 2018. http://krasserm.github.io/2018/03/21/bayesian-optimization/.

Turner, Ryan, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. 2021. “Bayesian Optimization Is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020.” arXiv. http://arxiv.org/abs/2104.10201.

Wei, Wang. 2020. “Bayesian Optimization Concept Explained in Layman Terms.” 2020. https://towardsdatascience.com/bayesian-optimization-concept-explained-in-layman-terms-1d2bcdeaf12f.

Appendix 1

In this notebook, I implemented the 1D toy problem using BO. I used sklearn package for BO implementation

```
: import numpy as np

bounds = np.array([[0, 1.0]])

def f(X):
    return  1 - 0.5 * (np.sin(12*X) / (1+X) + 2 * np.cos(7*X) * X**5

X_init = np.array([[0.05],
                   [0.6],
                   [0.95]])
Y_init = f(X_init)
```

```
: import matplotlib.pyplot as plt

# Dense grid of points within bounds
X = np.arange(bounds[:, 0], bounds[:, 1], 0.01).reshape(-1, 1)

# Noise-free objective function values at X
Y = f(X)

# Plot optimization objective with noise level
plt.plot(X, Y, 'y--', lw=2, label='Noise-free objective')
plt.plot(X_init, Y_init, 'kx', mew=3, label='Initial samples')
plt.legend()
```

```
: from scipy.stats import norm

def expected_improvement(X, X_sample, Y_sample, gpr, xi=0.01):
    '''
    Computes the EI at points X based on existing samples X_sample
    and Y_sample using a Gaussian process surrogate model.

    Args:
        X: Points at which EI shall be computed (m x d).
        X_sample: Sample locations (n x d).
        Y_sample: Sample values (n x 1).
        gpr: A GaussianProcessRegressor fitted to samples.
        xi: Exploitation-exploration trade-off parameter.

    Returns:
        Expected improvements at points X.
    '''
    mu, sigma = gpr.predict(X, return_std=True)
    mu_sample = gpr.predict(X_sample)

    sigma = sigma.reshape(-1, 1)

    mu_sample_opt = np.max(mu_sample)

    with np.errstate(divide='warn'):
        imp = mu - mu_sample_opt - xi
        Z = imp / sigma
        ei = imp * norm.cdf(Z) + sigma * norm.pdf(Z)
        ei[sigma == 0.0] = 0.0

    return ei
```

```
from scipy.optimize import minimize

def propose_location(acquisition, X_sample, Y_sample, gpr, bounds, n_
    '''
    Proposes the next sampling point by optimizing the acquisition fu

    Args:
        acquisition: Acquisition function.
        X_sample: Sample locations (n x d).
        Y_sample: Sample values (n x 1).
        gpr: A GaussianProcessRegressor fitted to samples.

    Returns:
        Location of the acquisition function maximum.
    '''
    dim = X_sample.shape[1]
    min_val = 1
    min_x = None

    def min_obj(X):
        # Minimization objective is the negative acquisition functio
        return -acquisition(X.reshape(-1, dim), X_sample, Y_sample, g

    # Find the best optimum by starting from n_restart different ran
    for x0 in np.random.uniform(bounds[:, 0], bounds[:, 1], size=(n_
        res = minimize(min_obj, x0=x0, bounds=bounds, method='L-BFGS-
        if res.fun < min_val:
            min_val = res.fun[0]
            min_x = res.x

    return min_x.reshape(-1, 1)
```

```
import matplotlib.pyplot as plt

def plot_approximation(gpr, X, Y, X_sample, Y_sample, X_next=None, sh
    mu, std = gpr.predict(X, return_std=True)
    plt.fill_between(X.ravel(),
                     mu.ravel() + 1.96 * std,
                     mu.ravel() - 1.96 * std,
                     alpha=0.1)
    plt.plot(X, Y, 'y--', lw=1, label='Noise-free objective')
    plt.plot(X, mu, 'b-', lw=1, label='Surrogate function')
    plt.plot(X_sample, Y_sample, 'kx', mew=3, label='samples')
    if X_next:
        plt.axvline(x=X_next, ls='--', c='k', lw=1)
    if show_legend:
        plt.legend()

def plot_acquisition(X, Y, X_next, show_legend=False):
    plt.plot(X, Y, 'r-', lw=1, label='Acquisition function')
    plt.axvline(x=X_next, ls='--', c='k', lw=1, label='Next sampling
    if show_legend:
        plt.legend()

def plot_convergence(X_sample, Y_sample, n_init=2):
    plt.figure(figsize=(12, 3))

    x = X_sample[n_init:].ravel()
    y = Y_sample[n_init:].ravel()
    r = range(1, len(x)+1)

    x_neighbor_dist = [np.abs(a-b) for a, b in zip(x, x[1:])]
    y_max_watermark = np.maximum.accumulate(y)

    plt.subplot(1, 2, 1)
    plt.plot(r[1:], x_neighbor_dist, 'bo-')
    plt.xlabel('Iteration')
    plt.ylabel('Distance')
    plt.title('Distance between consecutive x\'s')

    plt.subplot(1, 2, 2)
    plt.plot(r, y_max_watermark, 'ro-')
    plt.xlabel('Iteration')
    plt.ylabel('Best Y')
    plt.title('Value of best selected sample')
```

```
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import ConstantKernel, Matern

m52 = ConstantKernel(1.0) * Matern(length_scale=1.0, nu=2.5)
gpr = GaussianProcessRegressor(kernel=m52)

# Initialize samples
X_sample = X_init
Y_sample = Y_init

# Number of iterations
n_iter = 10

plt.figure(figsize=(12, n_iter * 3))
plt.subplots_adjust(hspace=0.4)

for i in range(n_iter):
    # Update Gaussian process with existing samples
    gpr.fit(X_sample, Y_sample)

    # Obtain next sampling point from the acquisition function (expe
    X_next = propose_location(expected_improvement, X_sample, Y_sampl

    # Obtain next noisy sample from the objective function
    Y_next = f(X_next)

    # Plot samples, surrogate function, noise-free objective and nex
    plt.subplot(n_iter, 2, 2 * i + 1)
    plot_approximation(gpr, X, Y, X_sample, Y_sample, X_next, show_le
    plt.title(f'Iteration {i+1}')

    plt.subplot(n_iter, 2, 2 * i + 2)
    plot_acquisition(X, expected_improvement(X, X_sample, Y_sample, g

    # Add sample to previous samples
    X_sample = np.vstack((X_sample, X_next))
    Y_sample = np.vstack((Y_sample, Y_next))
```

```
plot_convergence(X_sample, Y_sample)
```

Appendix 2

Below is the code for calculationg ENPV with a modified method, using uniform distribution to select the initial injection rates and seed to keep the BO gusses same to be able to compare the caculation time between base case and modified method. all injection rates are considered to be the same.

```
# import necessary packages

from ecl.summary import EclSum
import sys
import os
import numpy as np
from array import *
import pandas as pd
import GPy
import GPyOpt
from GPyOpt.methods import BayesianOptimization
import matplotlib.pyplot as plt
from numpy.random import seed
seed(12345)
```

```
# parameters that we need to calculate ENPV

OP =  300   #Oil Price $ per m3
WPP = 47.5    #Water Production Price $ per m3
WIP = 12.5  #WaterInjection Price $ per m3
Study_period = np.arange(10) #Years
DiscountRate = 0.08
NPV = []
Inj_rate_Opt = []
NPV_Opt = []
```

```
# Modify Injection Rates

def modify_InjRate(Init_InjRate):

    Schedule_NEW_sample=pd.read_csv('SCHEDULE_NEW_sample.INC')

    for i in np.arange(0,8):
        #print("InjRate=" , Init_InjRate)
        Schedule_NEW_sample['WCONINJE'][0] = Schedule_NEW_sample['WC(
        Schedule_NEW_sample['WCONINJE'][1] = Schedule_NEW_sample['WC(
        Schedule_NEW_sample['WCONINJE'][2] = Schedule_NEW_sample['WC(
        Schedule_NEW_sample['WCONINJE'][3] = Schedule_NEW_sample['WC(
        Schedule_NEW_sample['WCONINJE'][4] = Schedule_NEW_sample['WC(
        Schedule_NEW_sample['WCONINJE'][5] = Schedule_NEW_sample['WC(
        Schedule_NEW_sample['WCONINJE'][6] = Schedule_NEW_sample['WC(
        Schedule_NEW_sample['WCONINJE'][7] = Schedule_NEW_sample['WC(

    Schedule_NEW_sample.to_csv('SCHEDULE_NEW.INC',index=False)
```

```
# NPV calculation function

def NPV_Calc():
    ''' In this function, we need to read the data provided by Eclips
    then, Eclips provides data for 10 years with intervals of 30 days
    since the discount rate is provided as yearly basis, I chose year
    discounted yearly and summed up all the discounted cash flows to
    This is done for every realizations'''

    summary = EclSum("ENSEMBLE_1.UNSMRY")
    Time = summary.numpy_vector("TIME")
    Years = summary.numpy_vector("YEARS")
    FOE = summary.numpy_vector("FOE")
    FOPR = summary.numpy_vector("FOPR")
    FOPT = summary.numpy_vector("FOPT")
    FPR = summary.numpy_vector("FPR")
    FWCT = summary.numpy_vector("FWCT")
    FWIR = summary.numpy_vector("FWIR")
    FWPR = summary.numpy_vector("FWPR")
    FWPT = summary.numpy_vector("FWPT")

    df = pd.DataFrame(list(zip(Time, Years, FOE, FOPR, FOPT,FPR, FWCT
    columns=["Time", "Years", "FOE","FOPR", "FOPT","FPR","FWCT", "FWI
    df2=df[['Time','FOPT','FWPT', 'FWIR']]
    df2=df2.iloc[11::12]
    df2.reset_index(drop=True, inplace=True)
    NetCashFlows = OP * np.array(df2['FOPT']) - WPP * np.array(df2['F
    DiscountFactors_end = 1/(1+DiscountRate)**Study_period
    DiscountedNetCashFlows = NetCashFlows*DiscountFactors_end
    NPV_end = sum(DiscountedNetCashFlows)/6.29
    NPV.append(NPV_end)
    #print("NPV_end for realization: " ,  NPV_end)
    return NPV
```

```
: # NPV function which will be passes to BO

def NPV_BO(X_init):
    '''In this function, we will calculate the ENPV, first we need to
    change the injection rates according to what BO suggests, run the
    1st realization, if the NPV calculated in the 1st realization is
    from initial injection rates, other 9 realizations with be run ar
    BO will provide another injection rate'''
    for i in X_init:
        for k in i:
            for u in range(1,11):
                os.chdir('/home/mina/Documents/Data/ENPV-Calc/' + str
                modify_InjRate(k)
                os.system(command='flow Ensemble_1.DATA > /dev/null')
                NPV_Calc()
                if u ==1 and NPV < ENPV_max:
                    break
            #print('NPV: ' , NPV)
            ENPV=round(sum(NPV)/len(NPV),4)
            print('ENPV of ' + str(k) , ': ', ENPV)
        NPV.clear()
        #print("NPV_now: " , NPV)
    return ENPV
```

```
# 10 initial values selected with uniform distribution
X_init = np.array([[76.0329],[96.3241],[5.7969],[15.1122],[33.3769],[
Y_init = np.array([[1.04339392e+08],
                   [9.69726948e+07],
                   [2.78021960e+07],
                   [6.64303849e+07],
                   [9.96255671e+07],
                   [1.06391346e+08],
                   [1.02514223e+08],
                   [1.00394251e+08],
                   [9.68430430e+07],
                   [1.04960784e+08],])
```

```
# I used Bayesian optimization package here I initialize the BO

#X_init = np.around(np.random.uniform(low=5, high=100, size=(10)), 4)
ENPV_max=np.max(Y_init)
bounds = np.array([[5, 100]])
k = GPy.kern.RBF(1)
bds = [{'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()}]

optimizer = BayesianOptimization(f=NPV_BO,
                                 domain=bds,
                                 model_type='GP',
                                 kernel=k,
                                 acquisition_type ='EI',
                                 X=X_init,
                                 Y = -Y_init,
                                 maximize=True)
```

```
# I run the BO for 10 iterations

optimizer.run_optimization(max_iter=10)
```

```
ENPV of 57.55270173036845 :  107483486.7496
ENPV of 77.90350705373396 :  105710578.8942
ENPV of 59.759011446595125 :  107379034.3401
ENPV of 59.535516841385146 :  107393012.5878
ENPV of 59.41871135952031 :  107401072.1495
ENPV of 59.315618863736724 :  107407165.3897
ENPV of 42.88839434097978 :  105264167.4191
ENPV of 25.972934200058024 :  93019668.6723
ENPV of 48.195449147980185 :  106759922.4245
ENPV of 38.30370466158602 :  105179153.8253
```

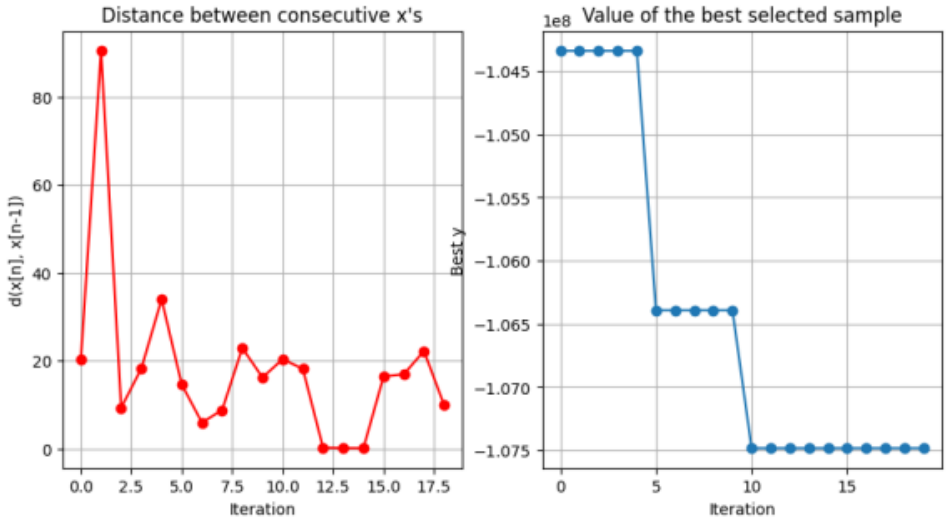
```
optimizer.fx_opt
```

```
-107483486.7496
```

```
optimizer.x_opt
```

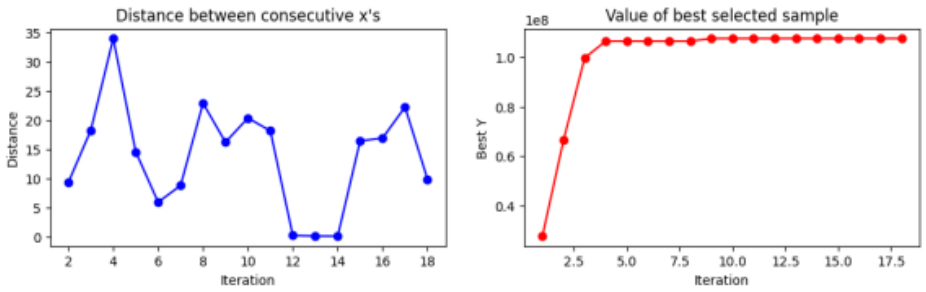
```
array([57.55270173])
```

```
optimizer.plot_convergence()
```

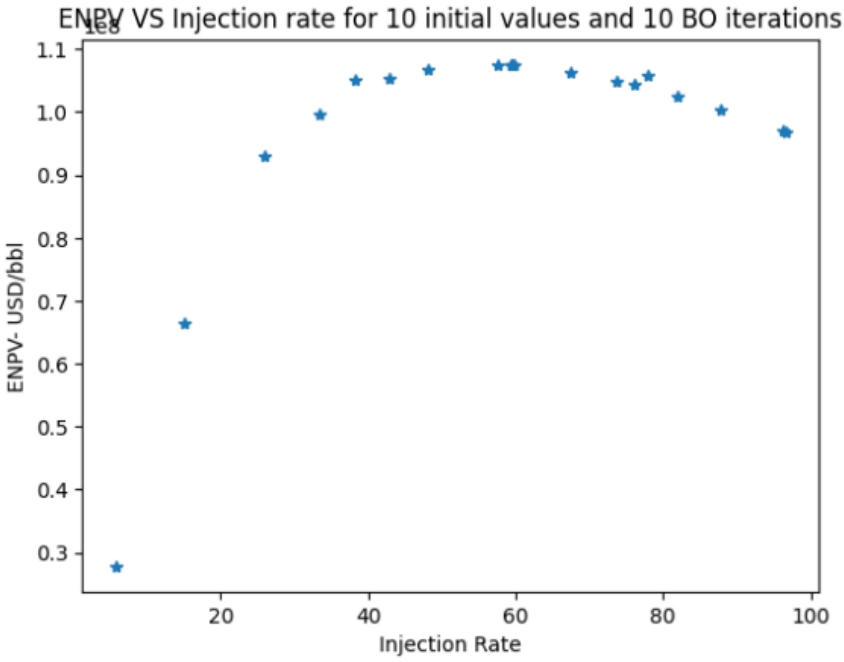



```
def plot_convergence(X_sample, Y_sample, n_init=2):
    plt.figure(figsize=(12, 3))

    x = X_sample[n_init:].ravel()
    y = Y_sample[n_init:].ravel()
    r = range(1, len(x)+1)

    x_neighbor_dist = [np.abs(a-b) for a, b in zip(x, x[1:])]
    y_max_watermark = np.maximum.accumulate(y)

    plt.subplot(1, 2, 1)
    plt.plot(r[1:], x_neighbor_dist, 'bo-')
    plt.xlabel('Iteration')
    plt.ylabel('Distance')
    plt.title('Distance between consecutive x\'s')

    plt.subplot(1, 2, 2)
    plt.plot(r, y_max_watermark, 'ro-')
    plt.xlabel('Iteration')
    plt.ylabel('Best Y')
    plt.title('Value of best selected sample')
```

:
```
plot_convergence(optimizer.X,-optimizer.Y)
```



```
x= optimizer.X
y=-optimizer.Y
x=x.flatten()
y=y.flatten()
plt.plot(x,y,'*')
plt.xlabel("Injection Rate")
plt.ylabel("ENPV- USD/bbl")
plt.title("ENPV VS Injection rate for 10 initial values and 10 BO ite
plt.show()
```



Appendix 3

Below is the code for calculationg ENPV with a modified method, using uniform distribution to select the initial injection rates. we have 8 arrays for 8 wells and they have different injection rates. we have 30 initial injection rates.

```
# import necessary packages

from ecl.summary import EclSum
import sys
import os
import numpy as np
from array import *
import pandas as pd
import GPy
import GPyOpt
from GPyOpt.methods import BayesianOptimization
import matplotlib.pyplot as plt
```

```
# parameters that we need to calculate ENPV

OP =  300   #Oil Price $ per m3
WPP = 47.5    #Water Production Price $ per m3
WIP = 12.5  #WaterInjection Price $ per m3
Study_period = np.arange(10) #Years
DiscountRate = 0.08
NPV = []
Inj_rate_Opt = []
NPV_Opt = []
```

```
#initial random injection rates selection using uniform distribution

Inj_Data = {'U1' : np.random.uniform(low=5, high=100, size=(30)),
            'U2' : np.random.uniform(low=5, high=100, size=(30)),
            'U3' : np.random.uniform(low=5, high=100, size=(30)),
            'U4' : np.random.uniform(low=5, high=100, size=(30)),
            'U5' : np.random.uniform(low=5, high=100, size=(30)),
            'U6' : np.random.uniform(low=5, high=100, size=(30)),
            'U7' : np.random.uniform(low=5, high=100, size=(30)),
            'U8' : np.random.uniform(low=5, high=100, size=(30))}

InjRate_NPV_df=pd.DataFrame(Inj_Data)
InjRate = InjRate_NPV_df.T
display(InjRate_NPV_df)

Inject_Different_Rates=InjRate_NPV_df.to_numpy()
Inject_Different_Rates=np.around(Inject_Different_Rates,2)
```

```
: # Modify Injection Rates with different Injection Rates

def modify_Different_InjRate(Init_InjRate):

    Schedule_NEW_sample=pd.read_csv('SCHEDULE_NEW_sample.INC')
    #print("InjRate=" , Init_InjRate)

    Schedule_NEW_sample['WCONINJE'][0] = Schedule_NEW_sample['WCONIN
    Schedule_NEW_sample['WCONINJE'][1] = Schedule_NEW_sample['WCONIN
    Schedule_NEW_sample['WCONINJE'][2] = Schedule_NEW_sample['WCONIN
    Schedule_NEW_sample['WCONINJE'][3] = Schedule_NEW_sample['WCONIN
    Schedule_NEW_sample['WCONINJE'][4] = Schedule_NEW_sample['WCONIN
    Schedule_NEW_sample['WCONINJE'][5] = Schedule_NEW_sample['WCONIN
    Schedule_NEW_sample['WCONINJE'][6] = Schedule_NEW_sample['WCONIN
    Schedule_NEW_sample['WCONINJE'][7] = Schedule_NEW_sample['WCONIN

    Schedule_NEW_sample.to_csv('SCHEDULE_NEW.INC',index=False)
```

```
# NPV calculation function

def NPV_Calc():
    ''' In this function, we need to read the data provided by Eclips
    then, Eclips provides data for 10 years with intervals of 30 days
    since the discount rate is provided as yearly basis, I chose year
    discounted yearly and summed up all the discounted cash flows to
    This is done for every realizations'''

    summary = EclSum("ENSEMBLE_1.UNSMRY")
    Time = summary.numpy_vector("TIME")
    Years = summary.numpy_vector("YEARS")
    FOE = summary.numpy_vector("FOE")
    FOPR = summary.numpy_vector("FOPR")
    FOPT = summary.numpy_vector("FOPT")
    FPR = summary.numpy_vector("FPR")
    FWCT = summary.numpy_vector("FWCT")
    FWIR = summary.numpy_vector("FWIR")
    FWPR = summary.numpy_vector("FWPR")
    FWPT = summary.numpy_vector("FWPT")

    df = pd.DataFrame(list(zip(Time, Years, FOE, FOPR, FOPT,FPR, FWCT
    columns=["Time", "Years", "FOE","FOPR", "FOPT","FPR","FWCT", "FWI
    df2=df[['Time','FOPT','FWPT', 'FWIR']]
    df2=df2.iloc[11::12]
    df2.reset_index(drop=True, inplace=True)
    NetCashFlows = OP * np.array(df2['FOPT']) - WPP * np.array(df2['F
    DiscountFactors_end = 1/(1+DiscountRate)**Study_period
    DiscountedNetCashFlows = NetCashFlows*DiscountFactors_end
    NPV_end = sum(DiscountedNetCashFlows)/6.29
    NPV.append(NPV_end)
    #print("NPV_end for realization: " ,  NPV_end)
    return NPV
```

```
# NPV function which will be passed to BO

def NPV_BO(X_init):
    '''In this function, we will calculate the ENPV, first we need to
    change the injection rates acording to what BO suggests, run the
    1st realization, if the NPV calculated in the 1st realization is
    from initial injection rates, other 9 realizations with be run an
    BO will provide another injection rate'''
    for i in X_init:
        for u in range(1,11):
            os.chdir('/home/mina/Documents/Data/ENPV-Calc/' + str(u))
            modify_Different_InjRate(i)
            os.system(command='flow Ensemble_1.DATA > /dev/null')
            NPV_Calc()
            if u ==1 and NPV < ENPV_avg:
                break
        #print('NPV: ' , NPV)
        ENPV=round(sum(NPV)/len(NPV),4)
        print('ENPV of ' + str(i) , ': ', ENPV)
    NPV.clear()
    #print("NPV_now: " , NPV)
    return ENPV
```

```
#initial injection rates and the corresponding ENPVs are read to be p
X_init = Inject_Different_Rates
Y_init = pd.read_csv('/home/mina/Documents/Data/ENPV-Calc/Y_initial_u
Y_init = Y_init.to_numpy()
```

```
# I used Bayesian optimization package here I initialize the BO

ENPV_avg=np.average(Y_init)
bounds = np.array([[5, 100]])
k = GPy.kern.RBF(1)
bds = [{'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()},
       {'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()},
       {'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()},
       {'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()},
       {'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()},
       {'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()},
       {'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()},
       {'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()}]

optimizer = BayesianOptimization(f=NPV_BO,
                                 domain=bds,
                                 model_type='GP',
                                 kernel=k,
                                 acquisition_type ='EI',
                                 X=X_init,
                                 Y = -Y_init,
                                 maximize=True)
```

```
# I run the BO for 20 iterations
optimizer.run_optimization(max_iter=20)
```

```
#In below code, I saved the initial random injection rates so I can
np.savetxt('X_initial_uniform2.csv', optimizer.X, delimiter=',')
```

```
#here, I saved the ENPV calculated from initial values, so I can use
np.savetxt('Y_initial_uniform2.csv', -optimizer.Y , delimiter=',')
```

```
# best injection rate
optimizer.x_opt
```

```
# best ENPV
optimizer.fx_opt
```

```
# convergence plot from BO package which I did not use
optimizer.plot_convergence()
```

```
# convergence plot function that I used to show the BO convergence
def plot_convergence(Y_sample, n_init=2):
    plt.figure(figsize=(12, 3))

    y = Y_sample[n_init:].ravel()
    r = range(1, len(y)+1)

    y_max_watermark = np.maximum.accumulate(y)

    plt.subplot(1, 2, 2)
    plt.plot(r, y_max_watermark, 'ro-')
    plt.xlabel('Iteration')
    plt.ylabel('Best Y')
    plt.title('Value of best selected sample')
```

```
plot_convergence(-optimizer.Y)
```

Appendix 4

Below is the code for calculationg ENPV with a modified method, using LHS to select the initial injection rates. we have 8 arrays for 8 wells and they have different injection rates. we have 30 initial injection rates.

```
from ecl.summary import EclSum
import sys
import os
import numpy as np
from array import *
import pandas as pd
import GPy
import GPyOpt
from GPyOpt.methods import BayesianOptimization
import matplotlib.pyplot as plt
from smt.sampling_methods import LHS
```

```
Init_Inject_Rate = np.empty((30,8))
```

```
# initial injection rates with LHS method
for i in np.arange(0,30):
    limits = np.array([[5, 100]])
    sampling = LHS(xlimits=limits)

    num = 8
    Init_Inject_Rate [i,:] = sampling(num).flatten()

print(Init_Inject_Rate)
print(Init_Inject_Rate.shape)
```

```
OP =  300   #Oil Price $ per m3
WPP = 47.5    #Water Production Price $ per m3
WIP = 12.5  #WaterInjection Price $ per m3
Study_period = np.arange(10) #Years
DiscountRate = 0.08
NPV = []
Inj_rate_Opt = []
NPV_Opt = []
```

```
# Modify Injection Rates test with different Injection Rates

def modify_Different_InjRate(Init_InjRate):

    Schedule_NEW_sample=pd.read_csv('SCHEDULE_NEW_sample.INC')
    #print("InjRate=" , Init_InjRate)

    Schedule_NEW_sample['WCONINJE'][0] = Schedule_NEW_sample['WCONIN
    Schedule_NEW_sample['WCONINJE'][1] = Schedule_NEW_sample['WCONIN
    Schedule_NEW_sample['WCONINJE'][2] = Schedule_NEW_sample['WCONIN
    Schedule_NEW_sample['WCONINJE'][3] = Schedule_NEW_sample['WCONIN
    Schedule_NEW_sample['WCONINJE'][4] = Schedule_NEW_sample['WCONIN
    Schedule_NEW_sample['WCONINJE'][5] = Schedule_NEW_sample['WCONIN
    Schedule_NEW_sample['WCONINJE'][6] = Schedule_NEW_sample['WCONIN
    Schedule_NEW_sample['WCONINJE'][7] = Schedule_NEW_sample['WCONIN

    Schedule_NEW_sample.to_csv('SCHEDULE_NEW.INC',index=False)
```

```
# NPV calculation function

def NPV_Calc():

    summary = EclSum("ENSEMBLE_1.UNSMRY")
    Time = summary.numpy_vector("TIME")
    Years = summary.numpy_vector("YEARS")
    FOE = summary.numpy_vector("FOE")
    FOPR = summary.numpy_vector("FOPR")
    FOPT = summary.numpy_vector("FOPT")
    FPR = summary.numpy_vector("FPR")
    FWCT = summary.numpy_vector("FWCT")
    FWIR = summary.numpy_vector("FWIR")
    FWPR = summary.numpy_vector("FWPR")
    FWPT = summary.numpy_vector("FWPT")

    df = pd.DataFrame(list(zip(Time, Years, FOE, FOPR, FOPT,FPR, FWC
    columns=["Time", "Years", "FOE","FOPR", "FOPT","FPR","FWCT", "FWI
    df2=df[['Time','FOPT','FWPT', 'FWIR']]
    df2=df2.iloc[11::12]
    df2.reset_index(drop=True, inplace=True)
    NetCashFlows = OP * np.array(df2['FOPT']) - WPP * np.array(df2['F
    DiscountFactors_end = 1/(1+DiscountRate)**Study_period
    DiscountedNetCashFlows = NetCashFlows*DiscountFactors_end
    NPV_end = sum(DiscountedNetCashFlows)/6.29
    NPV.append(NPV_end)
    #print("NPV_end for realization: " ,  NPV_end)
    return NPV
```

```
# NPV function which will be passes to BO

def NPV_BO(X_init):
    for i in X_init:
        for u in range(1,11):
            os.chdir('/home/mina/Documents/Data/ENPV-Calc/' + str(u)
            modify_Different_InjRate(i)
            os.system(command='flow Ensemble_1.DATA > /dev/null')
            NPV_Calc()
            if u ==1 and NPV < ENPV_avg:
                break
        #print('NPV: ' , NPV)
        ENPV=round(sum(NPV)/len(NPV),4)
        print('ENPV of ' + str(i) , ': ', ENPV)
    NPV.clear()
    #print("NPV_now: " , NPV)
    return ENPV
```

```
X_init = Init_Inject_Rate
Y_init = pd.read_csv('/home/mina/Documents/Data/ENPV-Calc/Y_initial_L
Y_init = Y_init.to_numpy()
```

```


ENPV_avg=np.average(Y_init)
bounds = np.array([[5, 100]])
k = GPy.kern.RBF(1)
bds = [{'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()},
       {'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()},
       {'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()},
       {'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()},
       {'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()},
       {'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()},
       {'name': 'X', 'type': 'continuous', 'domain': bounds.ravel()}]

optimizer = BayesianOptimization(f=NPV_BO,
                                 domain=bds,
                                 model_type='GP',
                                 kernel=k,
                                 acquisition_type ='EI',
                                 X=X_init,
                                 Y=-Y_init,
                                 maximize=True)
```

```
np.savetxt('Y_initial_LHS .csv', -optimizer.Y , delimiter=',')
```

```
np.savetxt('X_initial_LHS .csv', optimizer.X , delimiter=',')
```

```
optimizer.run_optimization(max_iter=20)
```

```
optimizer.x_opt
```

```
optimizer.fx_opt
```

:
```
optimizer.plot_convergence()
```

:
```
def plot_convergence(Y_sample, n_init=2):
    plt.figure(figsize=(12, 3))

    y = Y_sample[n_init:].ravel()
    r = range(1, len(y)+1)

    y_max_watermark = np.maximum.accumulate(y)


    plt.subplot(1, 2, 2)
    plt.plot(r, y_max_watermark, 'ro-')
    plt.xlabel('Iteration')
    plt.ylabel('Best Y')
    plt.title('Value of best selected sample')
```

:
```
plot_convergence(-optimizer.Y)
```