Sarah Mirsadeghi

Uncertainty Analysis and Quantification in Olympus Synthetic Reservoir Model

Master's thesis in Petroleum Engineering
Supervisor: Milan Stanko
Co-supervisor: Semyon Fedorov
June 2023

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Geoscience and Petroleum

NTNU

ABSTRACT

This study offers a Bottom Hole Pressure (BHP) uncertainty analysis performed on the Olympus synthetic reservoir model. Different BHP scenarios were taken into consideration by using various reservoir realizations. The reservoir's Net Present Value (NPV) was estimated using economic considerations and discount rate for each realization. By altering the number of Monte Carlo samples, convergence analysis was done to evaluate the precision of the NPV estimates. To depict the distribution of NPV values, probability density function (PDF) and cumulative distribution function (CDF) graphs were created. The research offers insightful information about how BHP uncertainties affect the reservoir's economic performance. The findings support reservoir management and investment choices, allowing for better-informed field development decisions.

Denne studien tilbyr en analyse av usikkerheten knyttet til bunnhullstrykk (BHP) utført på det kunstige Olympus-reservoarmodellen. Forskjellige BHP-scenarier ble tatt i betraktning ved bruk av ulike reservoarrealiseringer. Reservoarets nettonåverdi (NPV) ble estimert ved å ta økonomiske hensyn og diskonteringsrente for hver realisering. Ved å endre antall Monte Carlo-eksempler, ble konvergensanalyse utført for å evaluere presisjonen til NPV-estimatene. For å illustrere fordelingen av NPV-verdier ble det opprettet sannsynlighetstetthetsfunksjon (PDF) og kumulativ fordelingsfunksjon (CDF) -grafer. Forskningen gir innsikt i hvordan BHP-usikkerheter påvirker reservoarets økonomiske ytelse. Funnene støtter reservoarstyring og investeringsvalg, og muliggjør bedre informerte beslutninger om feltutvikling.

PREFACE

This report is a comprehensive documentation of my research project on uncertainty analysis and quantification. It presents the findings, methodologies, and interpretations derived from extensive research conducted in this field.

I want to express my sincere gratitude to Professor Milan Stanko for his invaluable guidance and support throughout the project. His expertise and insights have played a crucial role in shaping the direction and outcomes of this research.

The report is organized into six chapters. Chapter 1 introduces the research topic and outlines the objectives of the study. Chapters 2 describe the research methodology, including data collection and analysis techniques employed.Chapter 3 presents the findings and interpretations of the collected data, followed by a discussion.Chapter 4 provides conclusion of the study and the further work can be done related to this project.Additionally, Chapter 6 includes an appendix that supplements the main content.

I also extend my most profound appreciation to my co-supervisor, Semyon Fedorov, for his valuable input and guidance throughout the research process. Furthermore, I am grateful to my beloved husband for his unwavering support and encouragement during this endeavor.His belief in me and his constructive feedback has been invaluable in shaping this report.

I hope this report contributes to the existing uncertainty analysis and quantification knowledge, providing an understanding of the subject matter. It may be a valuable resource for researchers and practitioners in this field.

CONTENTS

Abstract i

Preface ii

Contents iv

List of Figures iv

List of Tables vi

Abbreviations viii

1 Introduction 1
1.1 Motivation . . . 1
1.2 Background and Definitions . . . 3
1.2.1 Parameter Uncertainty Quantification . . . 4
1.2.2 The Value of Information . . . 4
1.2.3 Value of Information and Sampling Bias . . . 4
1.2.4 Variability and Uncertainty . . . 5
1.2.5 Error and Uncertainty . . . 5
1.2.6 Uncertainty and Risk . . . 6
1.2.7 Risk and Reward . . . 6
1.2.8 Decision Analysis Under Uncertainty or Risk . . . 6
1.3 Previous Studies . . . 6
1.4 Objectives . . . 8
1.5 Working Tools . . . 9
1.5.1 Python Programming Language . . . 9
1.5.2 Eclipse Reservoir Simulator . . . 9

2 Methodology 11
2.1 Olympus Synthetic Reservoir Model . . . 11
2.1.1 Model Dimensions . . . 11
2.1.2 Facies and Property Modeling . . . 11
2.1.3 Oil-Water Contact and Model Initialization . . . 12
2.1.4 Model Realizations . . . 12
2.2 Determining Key Drivers of Production Profile Variability . . . 13
2.3 Monte Carlo . . . 14

2.4 Distribution functions 15
2.4.1 Probability density function (PDF) 15
2.4.2 Cumulative distribution function (CDF) 16
2.5 Economic Evaluation 18
2.6 Mean Absolute Percentage Error (MAPE) 19
2.7 Case Design and Procedure 20

3 Results and Discussion 23
3.1 PDF and CDF Analysis for Each Realization 23
3.2 Uncertainty Quantification 26
3.3 Convergence analysis 27

4 Conclusions and Further Work 31
4.1 Further Work 31

References 33

5 Appendices 37

A - Python code 38
5.1 Appendix A - 1: Sample Setup and Simulation run 38
5.2 Appendix A - 2: NPV Calculation and Result visualization 55

B - Tables 73
5.3 Appendix B : Default input values 73

C - Figures 74
5.4 Appendix C : PDF and CDF curves 74

D - GUI 81
5.5 Appendix D - 1: GUI of the first script 81
5.6 Appendix D - 2: GUI of the second script 82

Appendices: 82

References 83

LIST OF FIGURES

1.2.1 Illustration of the VOI using NTG ratios for reservoir delineation. (A) Only three data points were initially available. (B) Four additional data points have helped delineate the channel complex (shown between the two dotted lines)[11]. 5

2.4.1 Left: continuous PDF, right: discrete CDF[29]. 16
2.4.2 Left: continuous CDF/PDF, right: discrete CDF/PDF[29]. 17

3.1.1 OLYMPUS 49 PDF and CDF 23
3.1.2 OLYMPUS 40 PDF and CDF 24
3.1.3 OLYMPUS 45 PDF and CDF 25
3.2.1 Olympus six realiations PDF 26
3.2.2 Olympus six realiations CDF 27
3.3.1 OLYMPUS 45 - 25 Samples PDF and CDF 28
3.3.2 OLYMPUS 45 - 75 Samples PDF and CDF 28
3.3.3 OLYMPUS 45 - 125 Samples PDF and CDF 28
3.3.4 OLYMPUS 45 - 200 Samples PDF and CDF 29
3.3.5 Percentage error among different set of samples 29
3.3.6 Mean absolute percentage error 30

5.4.1 OLYMPUS 8 PDF and CDF 74
5.4.2 OLYMPUS 14 PDF and CDF 74
5.4.3 OLYMPUS 22 PDF and CDF 75
5.4.4 OLYMPUS 8 - 25 Samples PDF and CDF 75
5.4.5 OLYMPUS 8 - 75 Samples PDF and CDF 75
5.4.6 OLYMPUS 8 - 125 Samples PDF and CDF 76
5.4.7 OLYMPUS 8 - 200 Samples PDF and CDF 76
5.4.8 OLYMPUS 14 - 25 Samples PDF and CDF 76
5.4.9 OLYMPUS 14 - 75 Samples PDF and CDF 77
5.4.10 OLYMPUS 14 - 125 Samples PDF and CDF 77
5.4.11 OLYMPUS 14 - 200 Samples PDF and CDF 77
5.4.12 OLYMPUS 40 - 25 Samples PDF and CDF 78
5.4.13 OLYMPUS 40 - 75 Samples PDF and CDF 78
5.4.14 OLYMPUS 40 - 125 Samples PDF and CDF 78
5.4.15 OLYMPUS 40 - 200 Samples PDF and CDF 79
5.4.16 OLYMPUS 45 - 25 Samples PDF and CDF 79
5.4.17 OLYMPUS 45 - 75 Samples PDF and CDF 79

5.4.18 OLYMPUS 45 - 125 Samples PDF and CDF 80
5.4.19 OLYMPUS 45 - 200 Samples PDF and CDF 80
5.5.1 Graphical user interface of the first Python script 81
5.6.1 Graphical user interface of the second Python script 82

LIST OF TABLES

2.1.1 Summary of facies properties[26] 12
3.2.1 NPV values for different Olympus models 27
5.3.1 Default input values 73

ABBREVIATIONS

List of all abbreviations in alphabetic order:

- BHP Bottom Hole Pressure
- CAPEX Capital Expenditures
- CDF Cumulative Distribution Function
- CF Cash Flow
- GUI Graphical User Interface
- MAPE Mean Absolute Percentage Error
- NPV Net Present Value
- NTG Net To Gross
- OPEX Operational Expenses
- OWC Oil Water Contact
- PDF Probability Density Function
- VOI Value Of Information

CHAPTER ONE

INTRODUCTION

1.1 Motivation

Risk analysis is a crucial component that can be applied to various phases of the development process of a petroleum field. Each phase of the reservoir's lifecycle presents distinct decisions and uncertainties, necessitating the adoption of specific methodologies and tools tailored to the corresponding phase [1].

During the exploration phase, well-defined risk methodologies are typically employed to evaluate the potential of discovering hydrocarbon reserves. These methodologies involve analysing seismic data, geological studies, and other exploration techniques to assess the reservoir's presence, size, and quality. The focus is identifying prospects most likely to contain economically viable reserves. Risk assessments during this phase help inform decisions regarding drilling exploration wells, determining the feasibility of production, and estimating the initial reserves [2].

As the project transitions from the appraisal to the development phase, the level of uncertainty may decrease compared to the exploration phase. However, the significance of risk associated with the recovery factor, which relates to the percentage of hydrocarbons that can be extracted from the reservoir, may increase significantly. In the development phase, critical decisions need to be made regarding the production strategy, facilities design, and infrastructure planning. The complexity of the decision-making process arises from factors such as substantial irreversible investments, a large number of uncertainties, a strong dependence on the results associated with the production strategy definition, and the necessity of accurately predicting reservoir behavior[3].

Incorporating additional information on uncertain attributes and allowing for flexibility during the development phase is crucial to mitigate risks effectively; This can involve gathering more data through well testing, reservoir modelling, and dynamic simulation studies. By integrating this additional information, decision-makers can enhance their understanding of reservoir behaviour, optimize production strategies, and improve the overall project economics[4]. However, it is essential to note that acquiring additional information in offshore petroleum fields can be challenging and expensive due to the high costs associated with offshore operations and limited flexibility. As a result, probabilistic risk analysis becomes an essential tool, enabling decision-makers to assess and quantify the uncertainties associated with each possible scenario and make informed decisions based on the probabilities involved[5].

During the development phase of a petroleum field, various uncertainties arise that can significantly impact production and revenue. One such critical uncertainty is bottom hole pressure (BHP). BHP is the pressure at the bottom of the wellbore and plays a vital role in determining the flow rate and ultimate recovery of hydrocarbons. However, accurately predicting and managing BHP is challenging due to the complex interplay of reservoir characteristics, fluid properties, and wellbore dynamics. Uncertainties in BHP can result from variations in reservoir permeability, fluid behaviour, and the effectiveness of reservoir management techniques. The uncertainty in BHP can have significant consequences on production and revenue. For example, if the predicted BHP is higher than the actual BHP, reservoir deliverability can be overestimated, potentially causing production constraints and lower than expected output. On the other hand, if the predicted BHP is lower than the actual BHP, it may result in an underestimation of production potential and missed opportunities for maximizing hydrocarbon recovery. Therefore, accounting for BHP uncertainties and developing strategies to mitigate their impact on production and revenue is crucial[6].

In addition to BHP uncertainty, other uncertainties can affect the development phase. For example, geological uncertainties, such as variations in reservoir properties and fluid distribution, pose significant challenges in accurately characterizing the reservoir. These uncertainties can lead to variations in production performance, reservoir behaviour, and, ultimately, the project's economic viability. Therefore, quantifying and incorporating these geological uncertainties into reservoir modelling and production forecasting is essential to make informed decisions about well placement, facility design, and production strategies[7].

Moreover, operational uncertainties, such as drilling and completion uncertainties, equipment failures, and production disruptions, can further impact production and revenue. Unforeseen issues during drilling or completion operations can lead to delays, cost overruns, and suboptimal well performance. Equipment failures or production disruptions can result in downtime and reduced production rates, directly affecting revenue generation. Proper risk assessment and contingency planning are crucial to mitigate these operational uncertainties and ensure smooth operations throughout the development phase[8].

Companies can optimize production and maximize revenue by addressing and managing uncertainties, including BHP, geological, and operational uncertainties during the development phase. Integrated reservoir modelling, dynamic simulation studies, and advanced data analytics techniques can help quantify and mitigate these uncertainties. Additionally, incorporating sensitivity analyses and scenario planning can provide insights into the potential range of outcomes, enabling decision-makers to make more robust and informed decisions regarding production strategies, investment allocation, and risk mitigation measures[9].

Decision-making under uncertainty implies that at least one course of action has multiple potential outcomes. The utility of decision analysis methods lies not in eliminating risk entirely but in providing tools to evaluate, quantify, and understand the risks associated with different courses of action. Decision analysis techniques such as decision trees, Monte Carlo simulations, and sensitivity analyses can be employed to assess the impact of uncertainties on crucial project parameters and identify risk-mitigation strategies[1].

In conclusion, risk analysis and management play crucial roles throughout the lifecycle of a petroleum field. From exploration to development, understanding, and quantifying uncertainties, incorporating additional information, and utilizing decision analysis techniques are essential for effective risk mitigation and informed decision-making.

1.2 Background and Definitions

Risk and risk assessments have a long history. The Athenians demonstrated their ability to evaluate risk before making judgments more than 2400 years ago[10]. However, the scientific study of risk assessment and management is still very new, having only existed for thirty to forty years. The first scientific publications, papers, and conferences that addressed core concepts and guidelines for assessing and managing risk date back to this period effectively[11]. Uncertainty is a crucial notion in terms of risk conceptualization and risk assessments. Since the beginning of risk assessment in the 1970s and 1980s until now, there has been extensive discussion in the literature about how to comprehend and handle uncertainty.

The subject, however, is quite essential. A modern viewpoint on the difficulties, complications, and potential techniques for characterizing and communicating uncertainty in risk assessment is given in [12]. Probabilistic analysis is the most commonly used method for dealing with risk analysis uncertainties, both aleatory (representing variation) and epistemic (owing to a lack of information).

Due to data limitations and inference problems, there are uncertainties in reservoir characterization that affect the outcomes. Probability distributions are used to characterize the imprecision or incompleteness of measurements or observations, referred to as data uncertainty. Errors, variability, or restrictions in the data collection process are the causes of this uncertainty.
Inference uncertainty, conversely, denotes a lack of comprehensive understanding or certainty in drawing inferences from the existing evidence. It results from poor models, incomplete data, or the system's inherent variability. For effective analysis and well-informed decision-making, it is necessary to consider both data and inference uncertainties.

To address these uncertainties extensively, it is common practice to create probability distributions by developing several scenario models and generating multiple realizations of each contributing characteristic. With the help of this method, it is feasible to examine uncertainties more thoroughly and have a better knowledge of the range of potential results for reservoir characterization.

The definitions in this section come from Y. Zee Ma and Paul R. La Pointe's book "Uncertainty Analysis and Reservoir Modeling: Developing and Managing Assets in an Uncertain World". These definitions are specifically contained in Chapter 1 of the book[11].

1.2.1 Parameter Uncertainty Quantification

A probability distribution often represents uncertainty in reservoir parameters, like uncertainties in measurements. Many have examined how to quantify the level of uncertainty surrounding these parameters throughout the early stages of field development using a variety of data sources and geologic scenarios [13]. In experimental design, parameter uncertainty is often classified into two or three categories, low, medium, and high, rather than employing a probability distribution.

Probability distributions of uncertain quantities in reservoir characterization are often empirical, while some general mathematical laws may cause them to be nearly normal or lognormal. For example, as a result of the central limit theorem, adding numerous random variables typically results in a normal distribution[14], whereas multiplying numerous variables typically results in a lognormal distribution[15]. However, the distributions of the input variables and their relationships will significantly impact the resulting probability distribution, particularly when data are scarce.

1.2.2 The Value of Information

In light of the fact that uncertainty can be viewed as an issue of underdetermination, more data would inevitably lessen the uncertainty. This is the 'value of information'. In the oil and gas business, decision analysis is mainly used to discuss the VOI[16][17]. Therefore, reducing uncertainty in reservoir characterization and management is crucial for VOI.

1.2.3 Value of Information and Sampling Bias

Theoretically, as additional information becomes accessible, our understanding of the reservoir should advance. Nevertheless, sampling biases can make the VOI more challenging. Consider the situation in Figure 1.2.1 . Based on the first three wells, the average NTG was 23%. The average NTG from all seven wells after the four extra wells were drilled was reportedly 57%. However, there were five data points for the demarcated channel complex, which makes up around 60% of the area. Only two data points were available for the overbank region, which makes up two-fifths (40%) of the total area. As a result, there is a sample bias that needs to be taken into account.

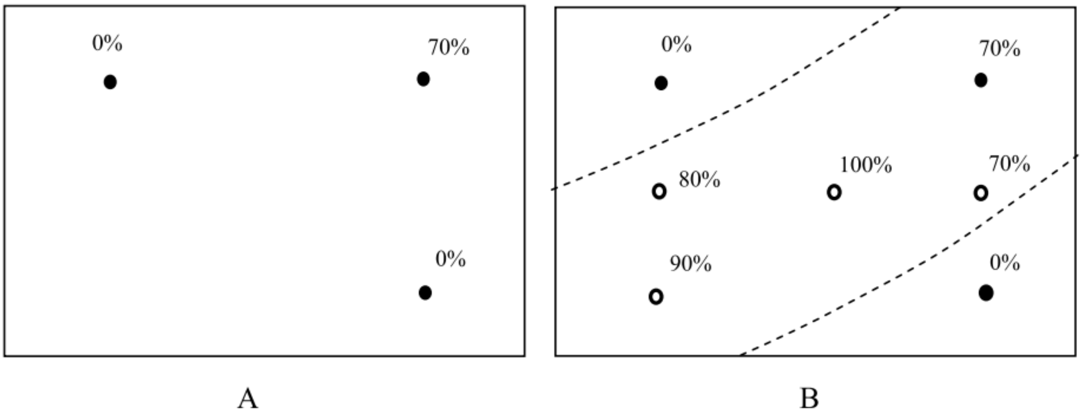


Figure 1.2.1: Illustration of the VOI using NTG ratios for reservoir delineation. (A) Only three data points were initially available. (B) Four additional data points have helped delineate the channel complex (shown between the two dotted lines)[11].

1.2.4 Variability and Uncertainty

Understanding the "variability" of geologic processes, petrophysical characteristics, and other reservoir variables is necessary for reservoir characterization. Variability is an attribute that may be measured and refers to how much these events fluctuate over time. Due to heterogeneities at several scales, including structural, stratigraphic, depositional facies, petrophysical characteristics, and fluid distributions, the subsurface exhibits great variability.

In contrast, "uncertainty" results from a lack of understanding about a certain variable. Uncertainty can result from mistakes in data or the indeterminacy and indefiniteness of a variable. Uncertainty, in contrast to variability, is not reliant on the occurrence of changes; it can still exist in the presence of a constant parameter if knowledge is incomplete. Despite this, variability and uncertainty frequently go hand in hand because high variability tends to add more unknowns, which increases uncertainty.

1.2.5 Error and Uncertainty

Keeping uncertainty and error separate is crucial. While an error is the difference between a single result and a number's real value, an uncertain quantity takes the form of a range due to the unknown elements. Error and uncertainty do, however, have a fundamental link. The likelihood of errors in the reservoir characterization result and business decision increases with larger uncertainty in the input data. In contrast, inaccurate geology, geophysical, petrophysical, or engineering data will result in more significant uncertainty in reservoir characterization and modeling.

Simply said, inadequate or uncertain input data can result in inaccuracies and uncertainties in the reservoir's final analysis. This can have an impact on the ability to make appropriate business decisions about the reservoir, such as resource estimation, production scheduling, or investment plans. Thus, it is essential to reduce uncertainty and assure the data's trustworthiness in reservoir characterization to enhance the validity of the findings and the decision-making procedures.

1.2.6 Uncertainty and Risk

The probability of unfavorable outcomes is frequently implied by the word "risk" in common speech. For instance, there is a chance that a well will turn out to be dry. Risk and uncertainty are related to some extent in this regard. In a more theoretical context, "risk" is defined as the likelihood of an unfavorable occurrence and its impact or consequences. Risk, therefore, consists of two components: uncertainty (how likely something is to occur) and consequence (what would happen if it did occur). It's vital to remember that risk can occasionally also refer to an uncertain outcome with no clear consequences.

Risk directly influences decision-making due to its consequence component. For example, the potential loss arising from a faulty prediction is included in risk analysis but not in uncertainty analysis alone. Indeed, it is frequently claimed in decision-making that the potential repercussions of being incorrect are more important than the probability of being wrong. For example, the Port-Royal writers' 1662 claim that fear of damage should be proportional to the likelihood of an event supports the idea of uncertainty analysis, but it ignores the consequence component of risk. Modern risk analysis, on the other hand, created on the basis of utility theory, says that only the irrational make decisions based simply on the probability of an outcome without considering its consequences.

1.2.7 Risk and Reward

Discussing risk and reward is important because risk may be lowered to zero if one does not care about the prospective benefits. Although it is commonly recognized that the oil exploration and production industry carries significant risks, there are also possible advantages. If not, nobody would take the chance to discover oil.

1.2.8 Decision Analysis Under Uncertainty or Risk

Making decisions requires reducing uncertainty to a manageable degree. A strategy for reducing uncertainty is to take the value of information (VOI) into account. In addition, enhancing reservoir characterization technologies, approaches, and procedures can also aid in lowering reservoir management uncertainty.

1.3 Previous Studies

Several pieces of research have significantly advanced the study and measurement of uncertainty in the oil and gas sector. In order to improve the precision and dependability of uncertainty assessment and to support well-informed decision-making processes, this research has used a variety of methodologies and strategies.

In one study, an innovative method for quantifying uncertainty in decision-making was presented[1]. The researchers created sophisticated mathematical methods and algorithms to analyze uncertainties more accurately. These techniques included sensitivity assessments, Monte Carlo simulations, and Bayesian inference. These methods help decision-makers comprehend the uncertainties surrounding the decision variables better and help them to make more informed decisions.

Researchers studied strategies for uncertainty quantification unique to porous media flows[3]. To account for uncertainty about fluid flow behavior in porous media, they investigated several modeling techniques, such as numerical simulation techniques. Statistical techniques, including Latin hypercube sampling and polynomial chaos expansion, were used to quantify uncertainties and evaluate their influence on flow behavior. These techniques helped decision-makers improve their production plans by giving them valuable insights into the unpredictability of flow parameters.

Model validation and uncertainty quantification were dealt with concurrently using multiobjective optimization approaches[4]. This research considered variables like data fitting, prediction accuracy, and model complexity while integrating several optimization objectives into the analysis. In addition, the researchers used strategies like evolutionary algorithms, genetic algorithms, and surrogate modeling to examine the trade-offs between various aims. By considering different objectives, decision-makers could improve model validation and acquire a more thorough grasp of the uncertainties related to the model inputs and outputs.

Comparative studies have been done in the field of petroleum reservoir engineering to assess various approaches to uncertainty quantification[18]. The probabilistic collocation and experimental-design methodologies were compared to measure their effectiveness in capturing reservoir uncertainty. While experimental-design methods used strategies like Latin hypercube sampling and orthogonal arrays, probabilistic-collocation methods used polynomial chaos expansions and spectral algorithms. These studies helped decision-makers choose the best strategy for assessing reservoir uncertainty by illuminating the advantages and disadvantages of each method.

Stochastic sampling techniques were compared to assess how good they were at estimating uncertainty[19]. Many algorithms were examined for performance and computational efficiency, including Markov Chain Monte Carlo, Latin hypercube sampling, and sequential Monte Carlo approaches. Decision-makers could compare various methods and make educated decisions based on the computational needs, accuracy, and applicability for particular uncertainty quantification activities.

In order to improve history matching and uncertainty quantification in petroleum reservoir modeling, population-based techniques were presented[20]. These algorithms, which include evolutionary methods, particle swarm optimization, and genetic algorithms, were used to calibrate reservoir models and quantify uncertainty related to model parameters. As a result, decision-makers could enhance the matching of historical production data and generate more accurate estimates of reservoir behavior uncertainty by utilizing these population-based methods.

Researchers considered both technical and market concerns when studying the staged development of a marginal oil field[21]. They used probabilistic modeling approaches, such as Monte Carlo simulations and scenario analysis, to evaluate the risks associated with reservoir features, oil prices, and project economics. These techniques allowed decision-makers to analyze various development scenarios under various uncertainties and estimate the risks of staged development.

In conclusion, a wide range of techniques were used in these investigations, including Bayesian inference, Monte Carlo simulations, sensitivity analyses, numerical simulations, evolutionary algorithms, surrogate modeling, probabilistic modeling, and population-based algorithms. As a result, decision-makers may efficiently measure and assess uncertainties in the oil and gas business using these methodologies, resulting in better management strategies and more informed decision-making processes.

1.4 Objectives

This project aims to create a computational framework for uncertainty analysis in reservoir modeling utilizing Monte Carlo sampling and considering BHP uncertainties in addition to reservoir characteristics uncertainty. The following are the precise objectives:

1. Implement a Python code that uses Monte Carlo sampling to provide a significant number of samples for each reservoir realization while accounting for BHP uncertainties using the normal probability distributions.

2. Create a parallel execution approach to use the Eclipse reservoir simulator to execute several reservoir realizations concurrently, utilizing parallel computing resources to lower the computational time needed for simulation runs.

3. Extract production profiles and calculate the Net Present Value (NPV) for each sampling realization, considering BHP uncertainties and assessing their impact on reservoir performance and economic viability.

4. Determine the point of convergence for the results by increasing the number of samples until the NPV calculations show negligible changes, ensuring an accurate representation of the uncertainty in the reservoir model considering BHP uncertainties.

5. Create visualizations such as plots and charts to effectively communicate and interpret the uncertainty analysis results, providing insights into the range of feasible production profiles and the corresponding economic uncertainty considering BHP uncertainties.

1.5 Working Tools

In this section, the primary assets used in this research to build the computational framework for reservoir modeling uncertainty analysis were reviewed. Two primary tools that had a significant contribution were Python Programming Language and Eclipse Reservoir Simulator.

1.5.1 Python Programming Language

Python was utilized as the programming language to put the computational foundation into place. As a result, it is the best option for scientific computing and data analysis activities due to its adaptability, extensive ecosystem of libraries, and simplicity of usage.

The Monte Carlo sampling technique for uncertainty analysis using Python was successfully applied. Various libraries, including NumPy and Pandas, were used to create random samples based on normal distributions, do statistical calculations, and handle numerical operations with ease. The seaborn package was also used to produce meaningful visualizations of the results.

Several functionalities, such as parallel execution and data processing, were easily integrated into the framework due to the flexibility of Python, which made it easier to design a modular and adaptable codebase. In addition, Python's extensive documentation and active community provided invaluable resources and support throughout the project's development[22].

1.5.2 Eclipse Reservoir Simulator

The Eclipse reservoir simulator is an industry-standard program for simulating fluid flow and predicting reservoir behavior. Due to its robust capabilities and algorithms, it is a trusted instrument in the oil and gas sector for reservoir engineering jobs.

The Eclipse reservoir simulator's sophisticated modeling capabilities and practical simulation algorithms were taken advantage of by integrating our computational framework with it. In addition, several reservoir realizations were run concurrently thanks to the parallel execution method, which considerably reduced the calculation time needed for the simulations.

The smooth integration of Python with the Eclipse reservoir simulator allowed data sharing, allowing us to extract simulation outputs and further analyze and display the results using Python's data manipulation and charting packages[23].

Overall, the combination of Python and the Eclipse reservoir simulator demonstrated a powerful and effective working environment for developing and implementing the uncertainty analysis framework.

CHAPTER

TWO

METHODOLOGY

2.1 Olympus Synthetic Reservoir Model

The Olympus synthetic reservoir model serves as a representative simulation of a newly discovered oil field in the North Sea. Developed collaboratively by researchers from TNO (the Netherlands Organization for Applied Scientific Research), TU Delft, and industry partners ENI, Equinor (previously Statoil), and PETROBRAS, this model was specifically designed for benchmark studies and field development optimization activities[24].

2.1.1 Model Dimensions

The Olympus reservoir model encompasses a field with a border fault on one side and measures 9 km by 3 km in size. To capture the reservoir's complexities, the model incorporates 16 distinct strata representing the 50-meter-thick reservoir. In addition to the boundary fault, six minor faults are included within the reservoir. The model consists of two zones: the top zone features fluvial channel sands intermixed with floodplain shales, while the bottom zone comprises alternating layers of coarse, medium, and fine sands, resembling a clinoformal stratigraphic sequence. The impermeable shale layer separates the two zones.

The grid cells in the Olympus reservoir model are approximately 50 m Œ 50 m Œ 3 m in size. All the geological and petrophysical parameters are modeled at this grid scale without upscaling. The model consists of approximately 341,728 total grid cells, of which 192,750 are active. The presence of a single-layer shale barrier accounts for the inactive grid cells. Moreover, the model incorporates five non-sealing faults, enabling unrestricted fluid flow throughout the reservoir[25].

2.1.2 Facies and Property Modeling

Multiple facies types are represented in the Olympus reservoir model, with each zone containing four different facies. Table 2.1.1 summarizes the various facies types and their corresponding geological properties, including porosity, permeability, and Net-To-Gross (NTG). Conventional geostatistical approaches were employed to derive the geological characteristics of each facies type. A porosity-permeability relationship was not established at this field development stage due to limited information. The permeability values in the X and Y directions are identical, while the permeability in the Z direction is 10 percent of the X-direction permeability[25].

Table 2.1.1: Summary of facies properties[26]

Facies Type	Zones Present	Porosity Ranges	Permeability Ranges (mD)	Net-To-Gross
Channel Sand	Top	0.2-0.35	40-1000	0.8-1
Shale	Top and Barrier	0.03	1	0
Coarse Sand	Bottom	0.1-0.2	75-150	0.7-0.9
Sand	Bottom	0.1-0.2	75-150	0.75-0.95
Fine Sand	Bottom	0.05-0.1	10-50	0.9-1

2.1.3 Oil-Water Contact and Model Initialization

The depth of the Oil-Water Contact (OWC) in the Olympus reservoir model was determined to be 2090 m, along with an in-situ hydrostatic pressure of 206 Bar. This information was obtained from available exploration well logs. Due to the distinct relative permeability curves associated with each facies, each realization of the reservoir model exhibits a unique initial water saturation distribution[25].

2.1.4 Model Realizations

In this study six realizations of Olympus have been used to account for reservoir uncertainty. Among these realizations, one is the worst-case scenario, representing unfavorable reservoir properties and challenging conditions. Another realization is chosen as the best-case scenario, representing ideal reservoir characteristics and optimal performance. Also, four realizations fall between these extreme cases, representing intermediate reservoir behaviors.

The worst-case(Olympus 49) realization allows for an in-depth analysis of the potential challenges and limitations in reservoir performance. It serves as a critical reference point for understanding the impact of unfavorable reservoir properties on production outcomes under BHP uncertainty.

Conversely, the best-case realization(Olympus 40) serves as a benchmark for optimal reservoir performance. It provides valuable insights into the factors that contribute to successful reservoir development and allows for identifying best practices and potential opportunities for further optimization.

The four intermediate realizations(Olympus 8, 14, 22, 45) comprehensively understand the reservoir's behavior under varying conditions. As a result, they offer insights into possible production outcomes and the associated uncertainties.

By studying these realizations, it becomes possible to assess the sensitivity of reservoir performance to changes in porosity, permeability, net-to-gross characteristics, and initial water saturation under BHP uncertainty.

2.2 Determining Key Drivers of Production Profile Variability

The significance of understanding flow patterns and accurately predicting BHP cannot be overstated in upstream oil and gas production. Reliable knowledge about flow behavior is essential for upstream professionals to design and implement effective production schemes. A crucial aspect of this is the ability to accurately estimate the pressure drop from the reservoir bottom to the surface through production wells, which relies on the proper prediction and representation of BHP[27].

Despite numerous efforts to develop mechanistic approaches and conventional models or correlations, accurately predicting and describing BHP with high accuracy and low uncertainty remains challenging. Many existing models fail to capture the complexity of BHP behavior, resulting in limited predictive capability. This failure is particularly problematic considering BHP's substantial impact on flow pattern distribution through production wells.

In recognition of this challenge, it is imperative to consider the uncertainty associated with BHP and the uncertainty related to reservoir characteristics such as porosity, permeability, and net-to-gross ratio. A more comprehensive understanding of the overall system behavior can be achieved by incorporating BHP uncertainty into reservoir simulations. This incorporation allows for a more realistic assessment of the uncertainties that may affect production performance, enabling better decision-making and the development of more robust production strategies.

Considering BHP uncertainty alongside reservoir characteristic uncertainty offers a more holistic approach to uncertainty analysis in the upstream sector. It acknowledges the interplay between reservoir properties and flow behavior, recognizing that accurate BHP prediction is vital for optimizing production performance. Integrating BHP uncertainty into the analysis makes the overall uncertainty assessment more robust, providing a more accurate representation of the potential range of production outcomes and identifying appropriate mitigation strategies[25].

Addressing the challenges associated with BHP prediction and uncertainty analysis is crucial for improving production planning and optimizing reservoir performance. By accounting for BHP uncertainty in addition to reservoir characteristic uncertainty, upstream professionals can enhance their understanding of flow behavior, improve production scheme design, and make informed decisions that maximize the economic potential of oil and gas reservoirs.

2.3 Monte Carlo

The Monte Carlo method is a computational methodology used to handle problems combining uncertainty and probability. Numerous disciplines use it extensively, including engineering, economics, physics, and statistics. The technique comes from Monaco's renowned Monte Carlo Casino, well-known for its chance and randomness-based games.

The Monte Carlo approach is used in the context of uncertainty quantification to evaluate the propagation of uncertainty through a model or system. It entails producing numerous random samples or situations based on probability distributions linked to uncertain model parameters. Realizations or iterations are standard terms used to describe these samples.

The following steps make up the Monte Carlo process[28]:

1. Define Probability Distributions:
   Determine the model's uncertain parameters' probability distributions based on the information at hand or the knowledge of a professional. The normal (Gaussian), uniform, exponential, and other distributions are frequently utilized.

2. Create Random Samples:
   Random samples are created from the specified probability distributions for each uncertain parameter. The desired accuracy and problem complexity determine how many samples are needed.

3. Model evaluation:
   The model or system is assessed for each sampled parameter value set. This entails conducting simulations, resolving equations, or completing calculations to gain desired model outputs or reactions.

4. Statistical Analysis:
   After gathering all of the model outputs from the previous stage, statistical approaches are used to examine the data. Descriptive statistics like mean, standard deviation, and percentiles are computed to describe the distribution of the outputs.

5. Uncertainty Propagation:
   The statistical analysis sheds light on how uncertainty in the input parameters affects the outcome by spreading across the model. It aids in comprehending the likelihood of various outcomes and their range.

The Monte Carlo approach excels at handling complicated systems with numerous uncertain parameters. It captures the entire range of potential values and their corresponding probability by taking samples from the parameter distributions, enabling a thorough investigation of uncertainty.

The Monte Carlo approach is adaptable and can be used with a wide range of models, including physical experiments, computer simulations, and mathematical models. It can, however, be computationally taxing, particularly for models with several parameters or intricate interactions. Methods including variance reduction, importance sampling, and parallel computing are frequently used to increase effectiveness in these situations.

Overall, the Monte Carlo method is an effective tool for quantifying uncertainty, enabling decision-makers to assess risks, make informed decisions, and gain insights into the behavior of complex systems under uncertain conditions.

2.4 Distribution functions

Probability distributions are used to model random events for which the outcome is uncertain. They represent how probabilities are distributed across the possible values of a random variable. Probability distributions have various properties, such as expected value and variance, which can be calculated. Continuous random variables are denoted as X or T, while discrete random variables are denoted as K [29].

2.4.1 Probability density function (PDF)

A probability density function (PDF) is used in probability theory to express the relative likelihood that a continuous random variable takes on a specific value or falls within a particular range. The PDF, denoted as $f(t)$, defines the probability of the random variable falling within a range of values rather than taking on a specific value. The probability is determined by integrating the PDF over that range, which lies beneath the density function but above the horizontal axis and between the lowest and highest values of the range. The area under the entire curve is equal to 1, and the probability density function is nonnegative everywhere, i.e., $f(t)$ [30].

$$\int_{-\infty}^{\infty} f(t)dt = 1, \quad \sum_{k} f(k) = 1(2.1)$$

The probability that an event will occur between limits a and b is given by:

$$P(a \leq T \leq b) = \int_{a}^{b} f(t)dt = F(b) - F(a)(2.2)$$

$$P(a \leq K \leq b) = \sum_{i=a}^{b} f(k) = F(b) - F(a-1)(2.3)$$

Where $F(t)$ and $F(k)$ are the cumulative distribution functions (CDF) of the continuous and discrete random variables, respectively.

The probability of a discrete PDF at an instant value k_i can be calculated by minimizing the limits to $[k_{i-1}, k_i]$:

$$P(K = k_i) = P(k_i < K \leq k_i) = f(k)(2.4)$$

For a continuous PDF, where limits are minimized to $[t, t+\Delta t]$, the probability is calculated as:

$$P(T = t) = \lim_{\Delta t \to 0} P(t < T \leq t + \Delta t) = \lim_{\Delta t \to 0} f(t) \cdot \Delta t (2.5)$$

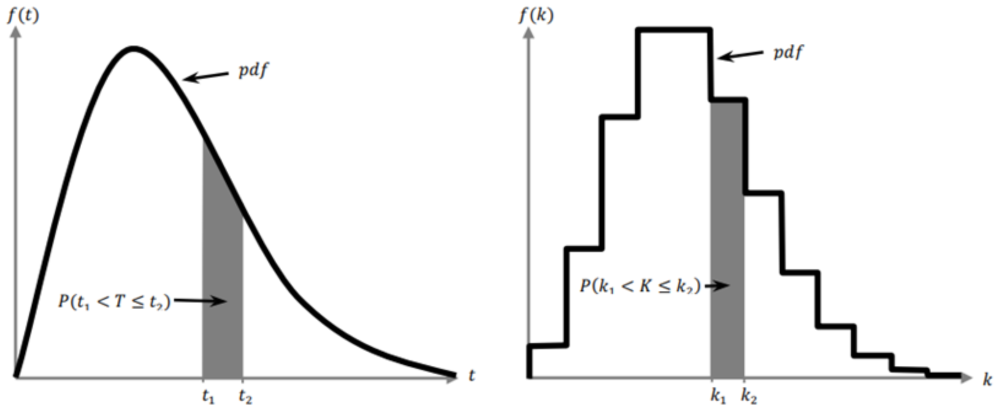


Figure 2.4.1: Left: continuous PDF, right: discrete CDF[29].

2.4.2 Cumulative distribution function (CDF)

The cumulative distribution function (CDF), denoted as $F(t)$ or $F(k)$, represents the probability that a random event will occur before or at a certain value of the random variable. The CDF is obtained by integrating the PDF:

$$F(t) = P(T \leq t) = \int_{-\infty}^{t} f(x) dx (2.6)$$

$$F(k) = P(K \leq k) = \sum_{i=a}^{b} f(k_i) \quad \text{for } k_i \leq k (2.7)$$

The limits of the CDF for $-\infty < t < \infty$ and $0 \leq k \leq \infty$ are:

$$\lim_{t \to \infty} F(t) = 0, \quad F(-1) = 0 (2.8)$$

$$\lim_{t \to \infty} F(t) = 1, \quad \lim_{k \to \infty} F(K) = 1 (2.9)$$

The CDF can also be used to calculate the probability of an event occurring between two limits:

$$P(a \leq T \leq b) = \int_{a}^{b} f(t) dt = F(b) - F(a) (2.10)$$

$$P(a \leq K \leq b) = \sum_{i=a}^{b} f(k) = F(b) - F(a-1) (2.11)$$

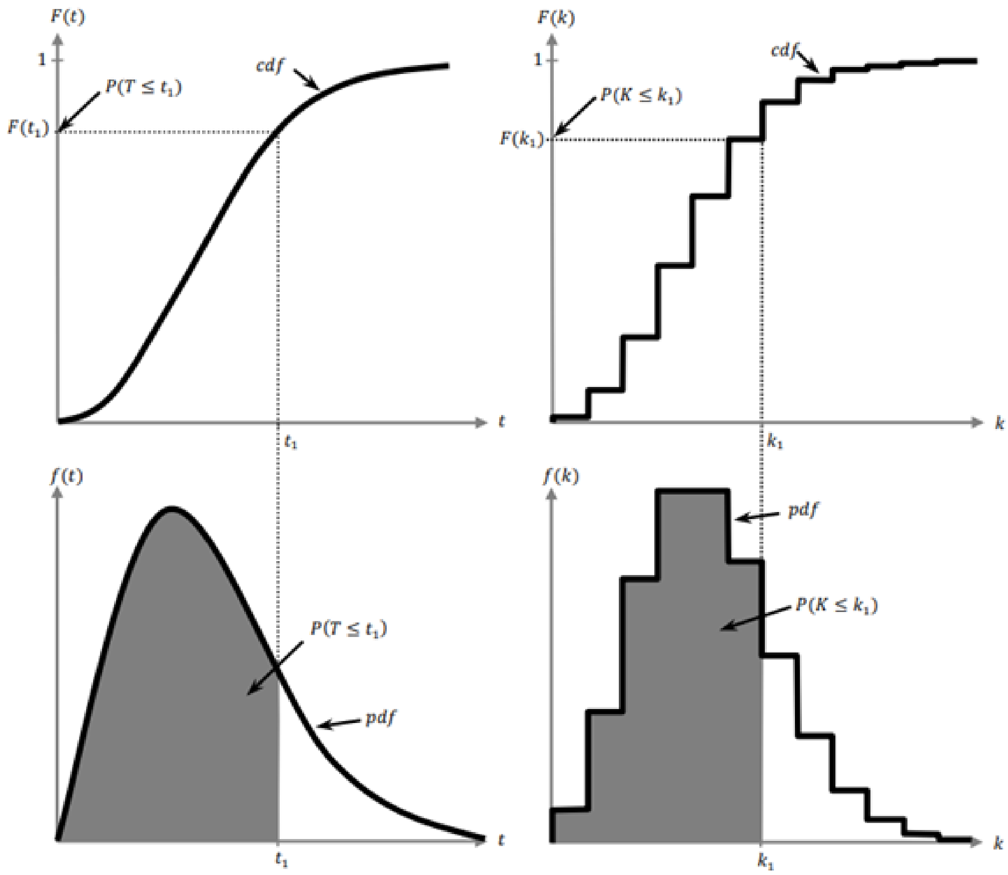


Figure 2.4.2: Left: continuous CDF/PDF, right: discrete CDF/PDF[29].

2.5 Economic Evaluation

The economic evaluation of the project relies on the Net Present Value (NPV) as a key metric for assessing the financial performance across different case realizations. The NPV is an essential indicator of revenue generation and expense management throughout the development project, aiding decision-making in uncertainty.

The NPV represents the sum of discounted future cash flows, accounting for both positive and negative financial outcomes, brought back to their present value [31]. It is calculated using the formula:

$$\mathrm{NPV} = \sum \left(\frac{\mathrm{CF}_t}{(1+r)^t} \right) \quad (2.12)$$

Where:

$$\begin{aligned} \mathrm{NPV} &\text{ denotes the Net Present Value,} \\ \mathrm{CF}_t &\text{ represents the cash flow in each time period,} \\ r &\text{ is the discount rate,} \\ t &\text{ is the time period.} \end{aligned}$$

The cash flow (CF_t) is determined by subtracting the operating expenses (OPEX) and capital expenditures (CAPEX) from the generated revenue. The revenue comprises the combined annual revenue from oil and gas production.

$$\mathrm{CF}_t = (\text{Oil Revenue} + \text{Gas Revenue}) - \text{CAPEX} - \text{OPEX} \quad (2.13)$$

The components of the formula are as follows:

$$\begin{aligned} \text{Oil Revenue} &: \text{Annual oil production multiplied by the oil price,} \\ \text{Gas Revenue} &: \text{Annual gas production multiplied by the gas price,} \\ \text{CAPEX} &: \text{Total cost of drilling, piping, and manifold expenses,} \\ \text{OPEX} &: \text{Operational costs after production commences,} \\ &\quad \text{including fixed operational and water disposal expenses.} \end{aligned}$$

To perform the NPV calculation, the following inputs are needed from the user:

1. The number of years before production refers to the years required for the project before the production phase starts. It represents the period during which initial investments and preparations are made.

2. The number of production years indicates the duration of the production phase, during which oil and gas are extracted, and revenue is generated.

3. Drilling costs for production wells: The program can access the model schedule file to retrieve the number of wells and calculate the total drilling costs for these wells.

4. Piping cost: This represents the cost of piping infrastructure required for transporting the extracted oil and gas. Additionally, the user needs to provide the duration in years for which this cost is applicable.

5. Manifold cost: The cost of the manifold, which is an essential component of the production infrastructure, should be provided in millions of dollars ($M). The user also needs to specify the number of manifolds required.

6. OPEX (fixed): Operational costs after production commences, including fixed operational expenses, such as maintenance and personnel costs.

7. Oil price: The price of oil per standard cubic meter ($/Sm3).

8. Gas price: The price of gas per standard cubic meter ($/Sm3).

9. Water cost: The cost of water disposal per standard cubic meter ($/Sm3).

10. Interest rate: The discount rate used in the NPV calculation, expressed as a percentage. It represents the opportunity cost of investing in the project and is used to discount future cash flows to their present value.

Default values for the inputs can be found in the Appendix B.

The resulting NPV values were used to derive probability and cumulative distribution functions for the set of realizations, providing valuable insights into potential financial outcomes across various scenarios.
it should be noted that the NPV calculation stops when the cash flow becomes negative after starting the production.

For a detailed breakdown of the NPV calculations, please refer to Appendix A - 2.

2.6 Mean Absolute Percentage Error (MAPE)

In order to estimate the accuracy of the results and assess their convergence, the mean absolute percentage error (MAPE) method has employed. MAPE is a widely used statistic for analyzing the accuracy of forecasts or estimations by calculating the average percentage variation between expected and actual values. It offers a relative measure of error, making it easier to compare data of various scales and magnitudes.

The MAPE is calculated using the following formula[32]:

$$\text{MAPE} = \frac{1}{n} \sum \left(\left| \frac{\text{Actual} - \text{Forecast}}{\text{Actual}} \right| \right) \times 100 (2.14)$$

where:

- MAPE is the mean absolute percentage error.
- n is the number of data points or observations.

- $\sum$ denotes the summation symbol.
- Actual represents the actual or observed values.
- Forecast represents the predicted or estimated values.

The method determines the absolute percentage difference between each data point's actual and predicted values, adds up these differences, and then divides the total number of data points by the sum to determine the average. A percentage is used to represent the outcome.

A smaller average percentage difference between the anticipated and actual values is a sign of higher accuracy, which is indicated by a lower MAPE. A larger MAPE, on the other hand, denotes greater variance and lower forecast or estimate accuracy.

We can quantify the convergence of the NPV estimations by comparing the average percentage differences between the NPV value from a smaller sample size and the NPV value from a larger sample size using the MAPE.

2.7 Case Design and Procedure

The first Python script is used to analyze the impact of Bottom Hole Pressure (BHP) uncertainty on oil production profiles. This script's primary goal is to alter the schedule file by introducing variations in BHP values to analyze their impact. Six different reservoir model realizations are considered to account for the uncertainty in the reservoir characteristics and ensure a thorough assessment.

Each realization has two folders located in the machine's hard drive. The reservoir's critical properties are located in the first folder, and a data file containing a reference to these attributes is located in the second folder. The script creates various scenarios for each realization by modifying the schedule file inside the first folder, specifically by changing the BHP values.
these modifications are done by creating several samples of BHP values for the eleven production wells using the Monte Carlo sampling technique during each simulation run. The BHP values cover the range of uncertainties related to the wells' behavior and are between 140 and 200 psi.

In Addition, the first script makes use of multiprocessing techniques to speed up simulations and reduce computational time. Multiprocessing techniques allow several simulations to run simultaneously, significantly lowering the overall calculation time. As a result, in varied BHP conditions, the oil production profiles may be studied more effectively

For the organization and convenience of analysis, the script creates distinct folders with the name of each realization and a particular sample number. This folder structure makes it easy to save and retrieve simulation data, encouraging further study and result comparison.

In the final step, the script extracts production data from RSM files, which are the output files generated by the Eclipse reservoir simulator and the production profiles for oil, gas, and water are extracted and stored in an Excel sheet within the folder of each sample. These production profiles and the user-defined input data serve as the basis for further calculations.

The second script determines each sample's Net Present Value (NPV) using the user's input and the retrieved production data. The NPV values are then kept in each sample's folder, giving a thorough overview of the financial viability of various BHP scenarios. In addition, a summary file that compiles the production profiles for all samples and realizations is created in the main folder.

The main folder contains individual and aggregated NPV values for all samples and the recovery factor. This organized data makes it simple to compare and analyze the financial results of various BHP scenarios.

Based on the saved data, the script also provides the option to produce Probability Density Function (PDF) and Cumulative Distribution Function (CDF) charts. These maps offer insightful information on the distribution of production profiles and aid in determining the degree of uncertainty related to various BHP scenarios. The created plots are saved as files in the "myplots" directory in the main folder

To summarize, these Python scripts produced for this analysis update the schedule file and make simulation runs more effortless, but they also include data extraction, NPV calculation, and thorough result storage and presentation. By smoothly integrating these capabilities, the script enables users to examine the economic viability of various BHP scenarios, investigate recovery factor variations, and derive essential insights from output profiles.

In addition, these scripts also offers a convenient GUI, allowing one to change modeling parameters and obtain updated results.(Please refer to Appendix D to see a preview of the GUI)

CHAPTER

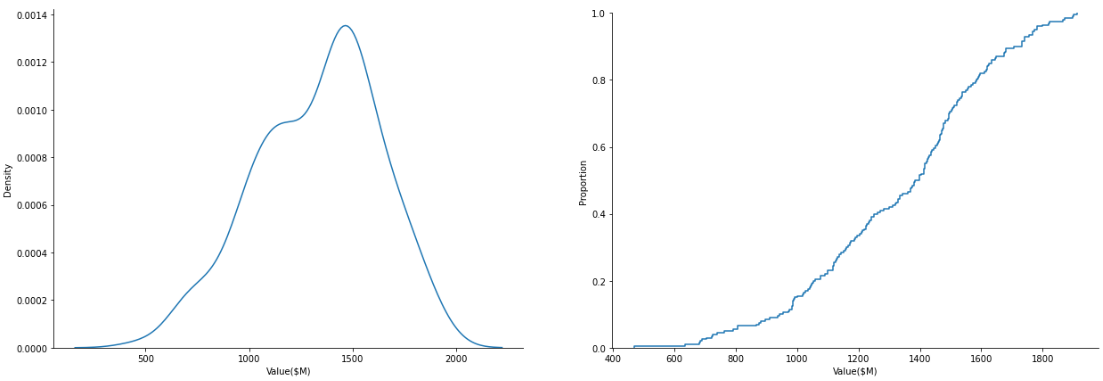
THREE

RESULTS AND DISCUSSION

3.1 PDF and CDF Analysis for Each Realization

The PDF plot sheds light on the probability of various NPV outcomes. The CDF plot demonstrates the cumulative likelihood of various NPV outcomes, while the distribution of NPV values reveals how probable each value is. It displays the probability of obtaining a specific NPV value or less.

OLYMPUS 49 (The best case)



PDF CDF

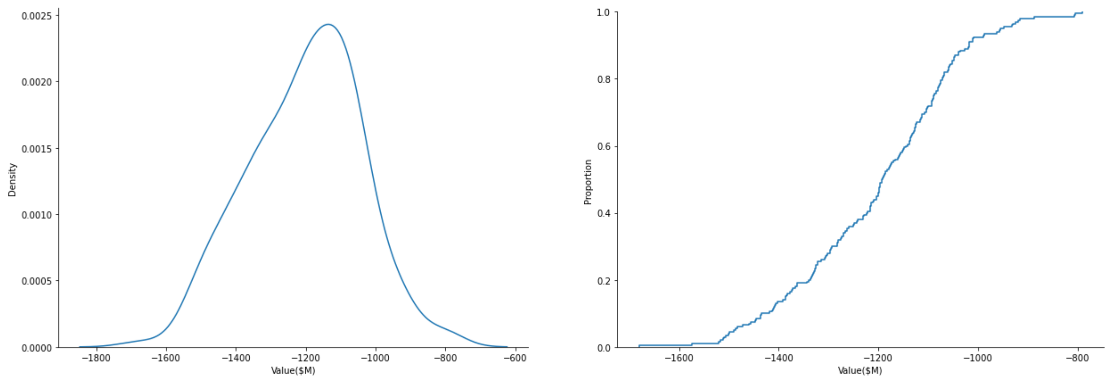
Figure 3.1.1: OLYMPUS 49 PDF and CDF

The left-skewed PDF curve distribution in Figure 3.1.1 shows that the probabilities are heavily weighted in favor of larger NPV values in Olympus 49. While the curve's tail extends to the left, its peak is displaced to the right. Peak displaced to the right implies a higher likelihood of reaching NPV values above the mean or median and a substantially lower likelihood of achieving lower NPV values.

Regarding the Olympus 49's CDF curve ,Figure 3.1.1, its integral-like shape shows that the probability initially builds gradually before increasing more quickly as the NPV values rise. Lower NPV values initially have a lower probability of being attained. However, as NPV values rise, the probability increases more quickly, suggesting a more significant probability of achieving NPV values towards the upper end of the range, in this example, closer to 1800 $M.

The CDF curve of Olympus 49 contains inflection point, which indicate a region where the rate of change in probability changes. This point reflect crucial BHP values or ranges significantly impacting NPV results. A key decision point or risk linked with BHP uncertainty, such as profitability thresholds, regulatory compliance, or operational limitations, can be identified by locating and comprehending these inflection points.

OLYMPUS 40(The worst case)



PDF CDF

Figure 3.1.2: OLYMPUS 40 PDF and CDF

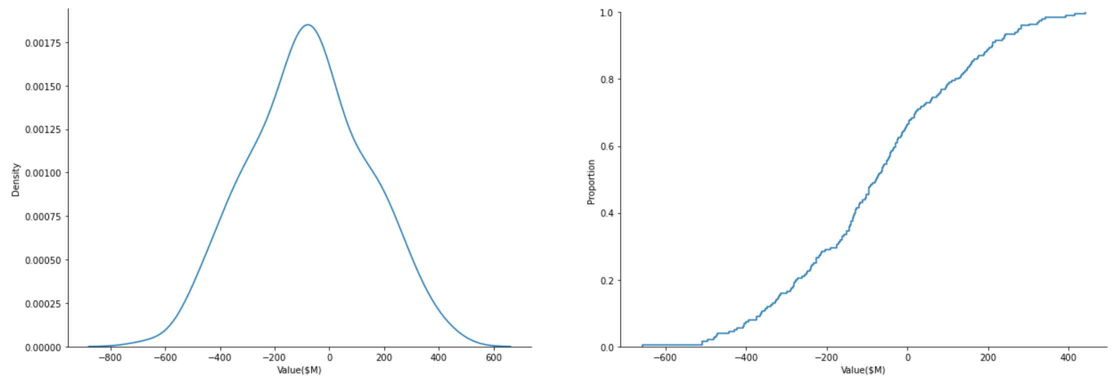
Figure 3.1.2's CDF curve shows various traits that provide the likelihood of obtaining greater NPV values in Olympus 40. The curve initially shows a positive trend, indicating a gradual rise in the probability of exceeding particular NPV targets as the values rise.

Notably, an almost plateau phase occurs on the CDF curve of Olympus 40. The plateau reflects a saturation point when there is little chance of future NPV value increases. The possibility of attaining even higher NPV values after this point is either constant or does not considerably rise.

The existence of the plateau phase in the CDF curve can be interpreted in several ways:

- Resource restrictions: The plateau could indicate restrictions on some resources, including production capacity or available funding. It implies limitations prohibiting NPV from increasing or expanding beyond a certain point.
- Factors that could cause risk: The plateau phase could indicate the existence of essential risks or uncertainties above and beyond a specific NPV value. In order to achieve larger NPV values, one must accept additional risks or uncertainties with a lower probability or impact.

OLYMPUS 45 (A middle case)



PDF CDF

Figure 3.1.3: OLYMPUS 45 PDF and CDF

The NPV values are distributed symmetrically and bell-shaped according to the Olympus 45's PDF plot in Figure 3.1.3. In contrast to the earlier examples, this distribution shows that probabilities are uniformly distributed around the mean or central NPV value. The NPV value most likely to be found is close to the distribution's center, as seen by the curve's peak aligning with the mean.

The NPV estimates have moderate fluctuation or uncertainty, given the PDF plot's symmetry and bell-shaped form. In addition, The curve's short tails suggest a lesser likelihood of extreme NPV values. This short tail implies that the NPV distribution centers more on the central or most likely NPV value. The narrower range of possible outcomes and the short tails suggest less variability or uncertainty in the NPV estimations.

Like the first instance, the CDF curve of Olympus 45 in Figure 3.1.3 displays an integral-shaped pattern representing the cumulative likelihood of attaining NPV values up to a specific threshold. The curve steadily rises on the y-axis as NPV values rise along the x-axis, representing the rising likelihood of achieving NPV values within that range. The CDF curve has an inflection point, just like in the first instance.

A critical range of NPV values that substantially impact the probability distribution is highlighted by this inflection point, which denotes a location where the rate of change in probability changes. The slope of the curve declines beyond the inflection point, indicating a decreased rate of probability growth. Nonetheless, the trend is still positive, showing that the likelihood of reaching greater NPV values is increasing, albeit slower.

Three additional realizations, in addition to the ones already covered, support mentioned justifications. Please refer to the appendix C for a visual reference.

3.2 Uncertainty Quantification

The P10/P50/P90 strategy is a reliable methodology used in this study to assess uncertainty in the analysis accurately. This approach uses the Monte Carlo simulation technique, which permits the creation of numerous alternative scenarios. The "P" in P10, P50, and P90 here stands for percentile.

A minimum of 90% probability that the quantities retrieved from the project will meet or surpass the low estimate is guaranteed by the P90 value. In light of the lower end of the estimate, this suggests a relatively conservative approach. P50, on the other hand, denotes a probability of at least 50% that the quantities match or exceed the best estimate. It acts as a trustworthy intermediate mean and forecasted value, representing a fair estimate within the range of possibilities.

The P10 number also includes a minimum 10% likelihood that the amounts will match or surpass the high estimation in the oil and gas sector. This represents a higher-end estimate that accounts for the possibility of better results[33].

The cumulative probability function is used to calculate these values. This function offers a thorough assessment of the probability distribution while considering numerous uncertainties and variables that affect the project's success. This method allows for more detailed knowledge of the uncertainty surrounding the project's results.

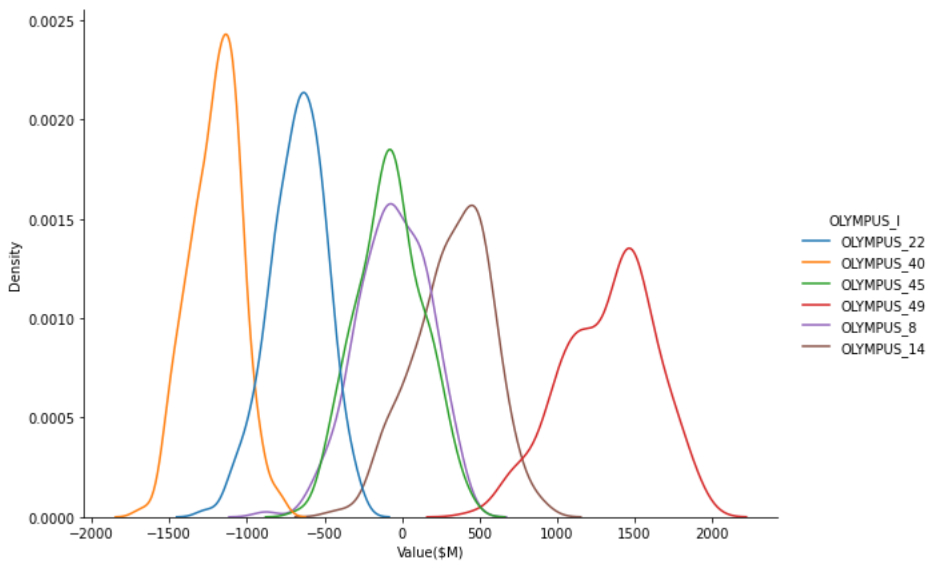


Figure 3.2.1: Olympus six realiations PDF

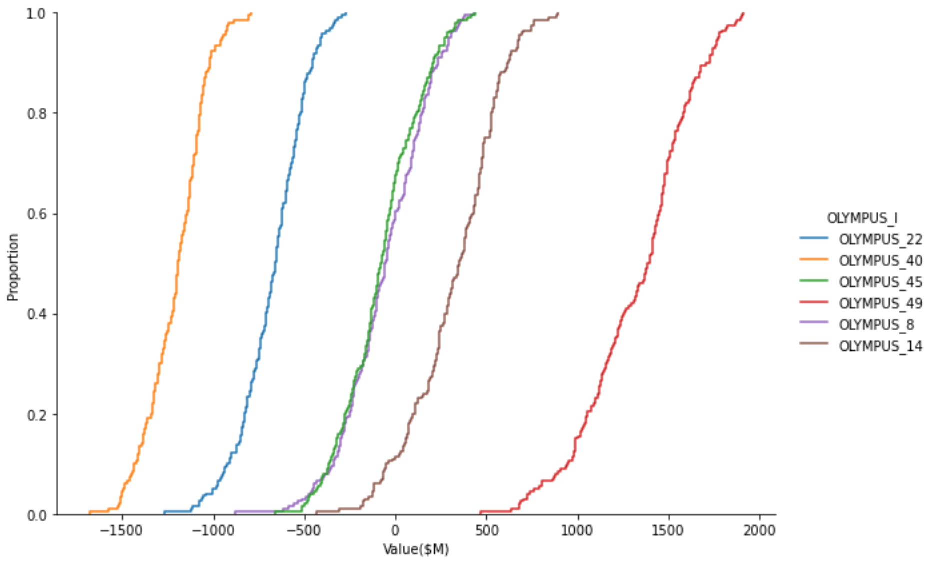


Figure 3.2.2: Olympus six realiations CDF

	Olympus 40			Olympus 22			Olympus 45		
	P90	P50	P10	P90	P50	P10	P90	P50	P10
NPV($M)	-1427	-1187.5	-1010.4	-916.6	-645.8	-458.3	-364.5	-83.3	208.3
	Olympus 8			Olympus 14			Olympus 49		
NPV($M)	P90	P50	P10	P90	P50	P10	P90	P50	P10
	-333.3	-52	229.1	-41.6	354.1	625	937.5	1385.4	1687.5

Table 3.2.1: NPV values for different Olympus models

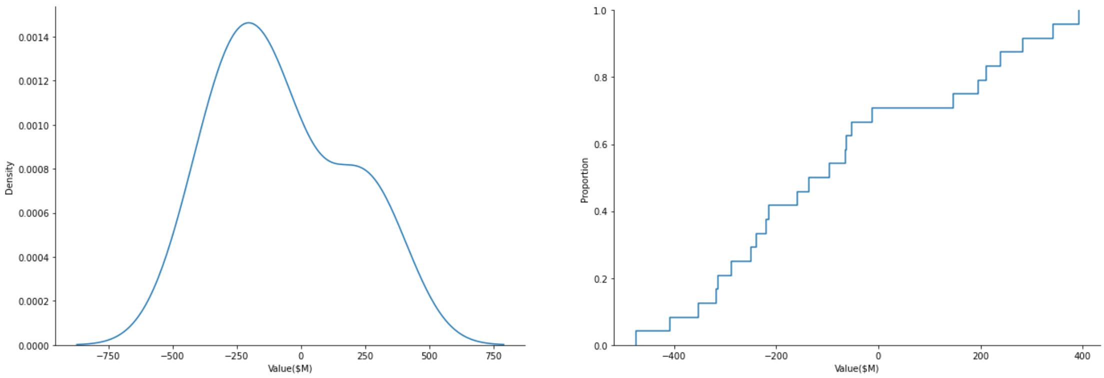
Both of these figures provide useful information about the possible production rates and the economic value of the field.

Consequently, the decision maker must devise a field development concept that effectively harnesses the upside potential of the field (in the case of Olympus 49) while safeguarding against potential downside risks (Olympus 40).

In summary, the analysis indicates that Olympus has a marginal economic outlook. Among the various realizations, only realization 49 and the optimistic scenarios for realization 14 are projected to yield a positive net present value (NPV).

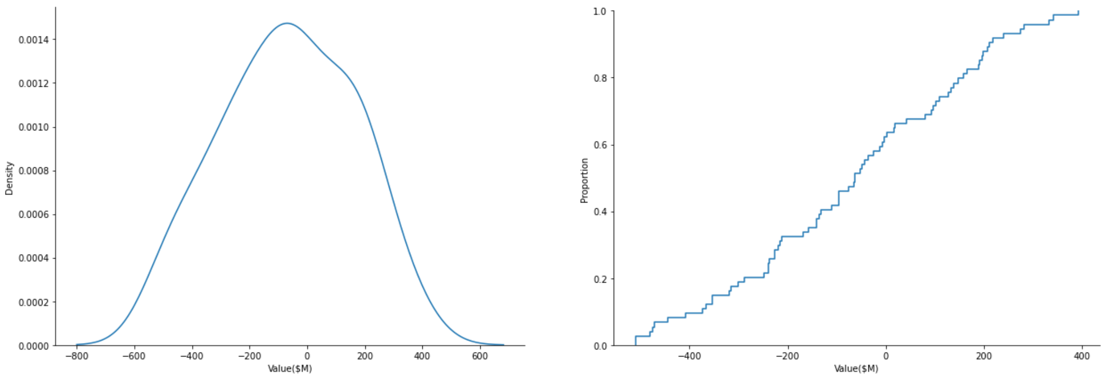
3.3 Convergence analysis

A convergence study was performed to examine the convergence behavior of NPV estimations using four different sample sizes (25, 75, 125, and 200 Monte Carlo samples). The graphs below show how the sample size and the calculated NPV values are related.



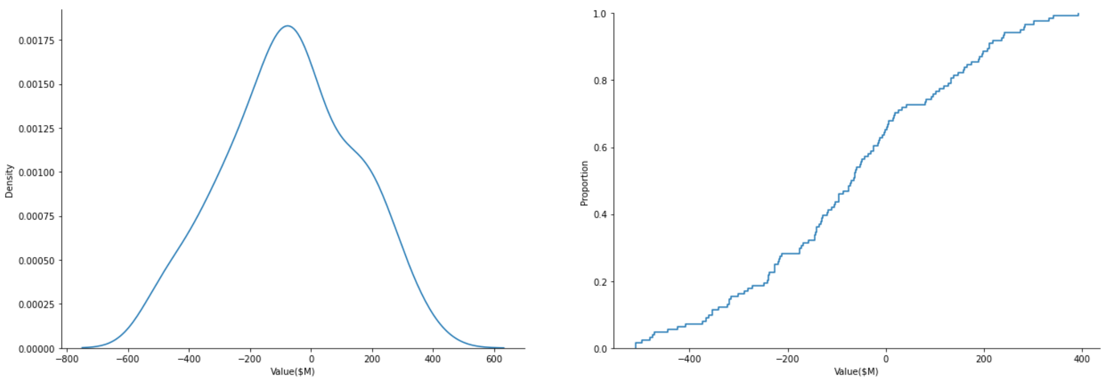
PDF CDF

Figure 3.3.1: OLYMPUS 45 - 25 Samples PDF and CDF



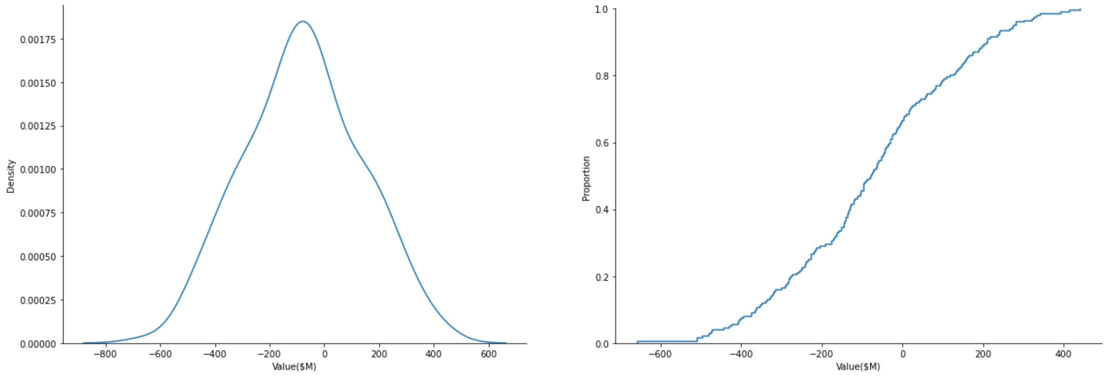
PDF CDF

Figure 3.3.2: OLYMPUS 45 - 75 Samples PDF and CDF



PDF CDF

Figure 3.3.3: OLYMPUS 45 - 125 Samples PDF and CDF



PDF CDF

Figure 3.3.4: OLYMPUS 45 - 200 Samples PDF and CDF

In addition, a comparative analysis was conducted to assess the convergence using a fixed set of 10 NPV values for each case. The mean absolute percentage error (MAPE) was calculated between these fixed NPV values obtained from varying sample sizes of 25, 75, 125, and 200 Monte Carlo samples in OLYMPUS 45.

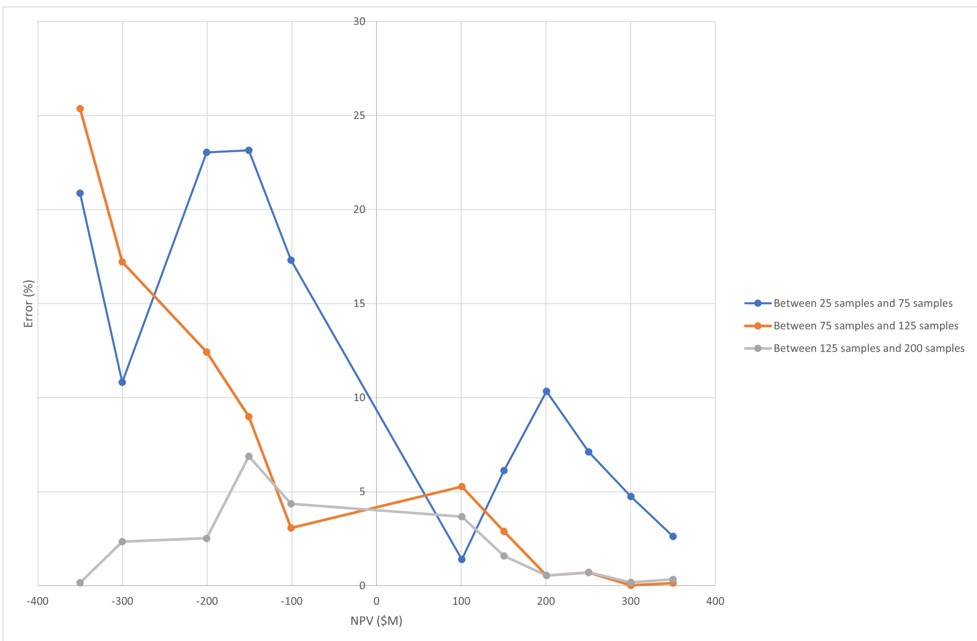


Figure 3.3.5: Percentage error among different set of samples

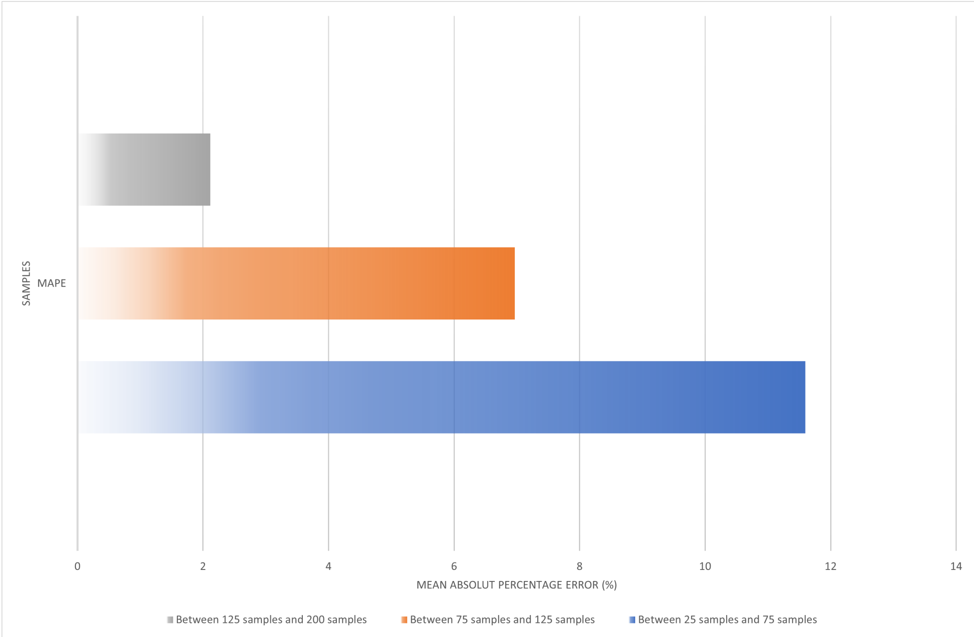


Figure 3.3.6: Mean absolute percentage error

Figures 3.3.5 and 3.3.6 show an apparent convergence between 25 and 200 samples. With smaller sample sizes, the NPV estimations' initial degree of variability is higher. However, as the sample size grows, the NPV values stabilize and exhibit less variability.
.Figure 3.3.5 has visual evidence of the convergence and diminishing error between observed and actual values as the sample size grows is provided by the observed values from the smaller sample set and the actual values from the larger sample set connected by a line.

According to Figure 3.3.6, The MAPE values also steadily decline with sample size, showing a decline in the average percentage difference and an improvement in the precision of the NPV predictions. The declining MAPE values imply a convergence to NPV estimates that are more precise.

Please refer to the Appendix C to see further convergence examples.

CHAPTER

FOUR

CONCLUSIONS AND FURTHER WORK

Conclusion

In conclusion, this work analyzed Bottom Hole Pressure (BHP) uncertainty in the Olympus synthetic reservoir model. The analysis determined the reservoir's Net Present Value (NPV) based on economic considerations and discount rates by considering several BHP scenarios and reservoir realizations. The convergence analysis revealed the precision of NPV estimates, which was conducted by altering the number of Monte Carlo samples. To further illustrate the distribution of NPV values, probability density function (PDF) and cumulative distribution function (CDF) graphs were used.

The study provided insight into how BHP's uncertainty affected the reservoir's economic performance. These revelations have important effects on reservoir management and investment choices. Stakeholders might choose better field development plans if they are aware of the connection between BHP uncertainty and financial results. The study's findings offer helpful insights to enhance reservoir management procedures and encourage the best investment choices in the oil and gas sector.

4.1 Further Work

Increase the number of simulation samples: Running simulations with more samples can increase the outcomes' precision and bring the Mean Absolute Percentage Error (MAPE) closer to zero. As a result, the analysis will be more precise overall, and more accurate estimations of the Net Present Value (NPV) will be provided.

Examine several production and injection well pricing scenarios: To evaluate their effect on the reservoir's economic performance, consider different pricing for production and injection wells. A more thorough understanding of the sensitivity of NPV to pricing changes can be attained by evaluating various price scenarios, allowing for improved risk management and decision-making.

Include abandonment prices in the NPV calculation: Include probable expenditures linked with abandonment activities in the NPV calculation. A more accurate evaluation of the project's total financial viability can be obtained by including abandonment costs in the economic analysis, which guarantees that long-term liabilities are properly considered.

Examine CDF curve inflection points and plateau phases: Analyze the cumulative distribution function (CDF) curves' observed inflection points and plateau phases in great detail. Investigate the fundamental causes and contributing elements of these patterns. The behavior of the reservoir can be better understood by knowing what causes inflection points and plateaus. This knowledge can also help make decisions about production plans and risk reduction techniques.

REFERENCES

[1] RB B Bratvold, SH H Begg, and Svitlana Rasheva. “A new approach to uncertainty quantification for decision making”. In: SPE Hydrocarbon Economics and Evaluation Symposium. SPE. 2010, SPE–130157.

[2] Gian Luigi Chierici and Gian Luigi Chierici. “Forecasting Well and Reservoir Performance Through the Use of Decline Curves and Identified Models”. In: Principles of Petroleum Reservoir Engineering (1995), pp. 231–253.

[3] Mike Christie, Vasily Demyanov, and Demet Erbas. “Uncertainty quantification for porous media flows”. In: Journal of Computational Physics 217.1 (2006), pp. 143–158.

[4] Ralf Schulze-Riegert et al. “Multiobjective optimization with application to model validation and uncertainty quantification”. In: SPE Middle East oil and gas show and conference. OnePetro. 2007.

[5] Jan-Erik Vinnem, Willy Røed, et al. “Offshore Risk Assessment Vol. 1”. In: Principles, Modelling and Applications of QRA Studies (2014).

[6] Ronald E Terry, J Brandon Rogers, and Benjamin Cole Craft. Applied petroleum reservoir engineering. Pearson Education, 2015.

[7] Reza Yousefzadeh et al. “Uncertainty Management in Reservoir Engineering”. In: Introduction to Geological Uncertainty Management in Reservoir Characterization and Optimization: Robust Optimization and History Matching. Springer, 2023, pp. 1–14.

[8] Peter R Rose et al. Risk analysis and management of petroleum exploration ventures. Vol. 12. American Association of Petroleum Geologists Tulsa, OK, 2001.

[9] John Fanchi. Integrated reservoir asset management: principles and best practices. Gulf Professional Publishing, 2010.

[10] James V Wertsch. “BASIL BERNSTEIN, Pedagogy, symbolic control and identity: Theory, research, critique. London (UK) & Bristol (PA): Taylor & Francis, 1996. Pp. xiv, 216. Hbč 40.00, pbč 14.95.” In: Language in Society 27.2 (1998), pp. 257–259.

[11] Y Zee Ma. “Uncertainty analysis in reservoir characterization and management: How much should we know about what we don’t know?” In: (2011).

[12] Roger Flage et al. "Concerns, challenges, and directions of development for the issue of representing uncertainty in risk assessment". In: Risk analysis 34.7 (2014), pp. 1196–1207.

[13] Guillaume Caumon and André G Journel. "A framework to assess global uncertainty: development and case studies". In: (2004).

[14] Athanasios Papoulis. "Ambiguity function in Fourier optics". In: JOSA 64.6 (1974), pp. 779–788.

[15] Lester G Telser. "The Lognormal Distribution, J. Aitchison and JAC Brown, Cambridge, England: Cambridge University Press, 1957, Pp. xviii, 176." In: American Journal of Agricultural Economics 41.1 (1959), pp. 161–162.

[16] Peter Cunningham and Steve Begg. "Using the value of information to determine optimal well order in a sequential drilling program". In: AAPG bulletin 92.10 (2008), pp. 1393–1402.

[17] Reidar B Bratvold, J Eric Bickel, and Hans Petter Lohne. "Value of information in the oil and gas industry: past, present, and future". In: SPE Reservoir Evaluation & Engineering 12.04 (2009), pp. 630–638.

[18] Heng Li, Pallav Sarma, and Dongxiao Zhang. "A comparative study of the probabilistic-collocation and experimental-design methods for petroleum-reservoir uncertainty quantification". In: SPE Journal 16.02 (2011), pp. 429–439.

[19] Linah Mohamed, Mike Christie, and Vasily Demyanov. "Comparison of stochastic sampling algorithms for uncertainty quantification". In: SPE journal 15.01 (2010), pp. 31–38.

[20] Yasin Hajizadeh. "Population-based algorithms for improved history matching and uncertainty quantification of petroleum reservoirs". PhD thesis. Heriot-Watt University, 2011.

[21] Semyon Fedorov, Verena Hagspiel, and Thomas Lerdahl. "Real options approach for a staged field development with optional wells". In: Journal of Petroleum Science and Engineering 205 (2021), p. 108837.

[22] Guido Van Rossum and Fred L Drake. Introduction to python 3: python documentation manual part 1. CreateSpace, 2009.

[23] Freddy H Escobar et al. "Pressure and pressure derivative analysis for linear homogeneous reservoirs without using type-curve matching". In: Nigeria Annual International Conference and Exhibition. OnePetro. 2004.

[24] RM Fonseca et al. "Overview of the olympus field development optimization challenge". In: ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery. Vol. 2018. 1. EAGE Publications BV. 2018, pp. 1–10.

[25] RM Fonseca, CR Geel, and O Leeuwenburgh. "Description of OLYMPUS reservoir model for optimization challenge". In: Integrated Systems Approach to Petroleum Production. Netherlands (2017).

[26] Susana MG Santos, Ana Teresa FS Gaspar, and Denis J Schiozer. "Risk management in petroleum development projects: Technical and economic indicators to define a robust production strategy". In: Journal of Petroleum Science and Engineering 151 (2017), pp. 116–127.

[27] Charles V Millikan and V Sidwell. “Bottom-hole pressures in oil wells”. In: Transactions of the AIME 92.01 (1931), pp. 194–205.

[28] George Fishman. Monte Carlo: concepts, algorithms, and applications. Springer Science & Business Media, 2013.

[29] Jun S Liu and Jun S Liu. Monte Carlo strategies in scientific computing. Vol. 75. Springer, 2001.

[30] Ronald W Shonkwiler and Franklin Mendivil. Explorations in monte carlo methods. Springer Science & Business Media, 2009.

[31] Werner Wetekamp. “Net Present Value (NPV) as a tool supporting effective project management”. In: Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems. Vol. 2. IEEE. 2011, pp. 898–900.

[32] Arnaud De Myttenaere et al. “Mean absolute percentage error for regression models”. In: Neurocomputing 192 (2016), pp. 38–48.

[33] JR Etherington and JE Ritter. “The 2007 SPE/WPC/AAPG/SPEE Petroleum Resources Management System (PRMS)”. In: Journal of Canadian Petroleum Technology 47.08 (2008).

CHAPTER

FIVE

APPENDICES

A - PYTHON CODE

All code and latex-files used in this document are included in the Github repository linked below. Further explanations are given in the readme-file.

5.1 Appendix A - 1: Sample Setup and Simulation run

Here is the Python code used for the calculations:

```python


import shutil
import random
import os
import numpy as np
from multiprocessing import Pool
import subprocess
from time import perf_counter
import pandas as pd
from tkinter import ttk
import tkinter as tk



def run1(address):
    '''Function to run a datafile in Eclipse'''
    subprocess.run(['eclrun', 'eclipse', address])




class MyGUI:
    def __init__(self):
        self.root = tk.Tk()
        self.frame1 = self.create_frame1()
        self.frame2 = self.create_frame2()
        self.frame3 = self.create_frame3()
        self.status_label = tk.Label(self.root, text='')
        #self.frame4 = self.create_frame4()
```

```
        #self.frame5 = self.create_frame5()
        #self.add_logo ()
        self.root.geometry('500x700')
        self.root.title('PPG')  # Set the window title
        #self.root.iconbitmap('C:/Users/sarah/OneDrive/Desktop/Thesis/NTNU_logo_400x400.ico')  # Set the window icon

        self.excel_file = None
        self.sheet_name = None
        self.sheet_data = None



    def create_frame1(self):
        frame1 = tk.Frame(self.root)

        #Frame title
        self.title1_label = tk.Label(text='Modifying realizations', font = ('Calibri', 12, 'bold'),fg='#149998')
        self.title1_label.pack(anchor=tk.CENTER, padx=5, pady=5,)


        # Label and entry for realizations
        self.irange_label = tk.Label(self.root, text='Enter a comma-separated list of realizations:')
        self.irange_label.pack()
        self.irange_entry = tk.Entry(self.root)
        self.irange_entry.pack()

        # Label and entry for BHP samples
        self.jrange_label = tk.Label(self.root, text='Enter the number of BHP samples:')
        self.jrange_label.pack()
        self.jrange_entry = tk.Entry(self.root)
        self.jrange_entry.pack()

        # Label and entry for the Lowest limit for BHP
        self.LL_label = tk.Label(self.root, text='Enter the lowest limit for BHP:')
        self.LL_label.pack()
        self.LL_entry = tk.Entry(self.root)
        self.LL_entry.pack()

        # Label and entry for the highest limit for BHP
        self.HL_label = tk.Label(self.root, text='Enter the
```

```
 highest limit for BHP:')
        self.HL_label.pack()
        self.HL_entry = tk.Entry(self.root)
        self.HL_entry.pack()

        #Creat the realizations
        self.submit_button = tk.Button(self.root, text='Create realizations', command=self.run_function1,font=('Calibri', 12, 'bold'))
        self.submit_button.pack(padx=10, pady=10)

        return frame1



    def create_frame2(self):
        frame2 = tk.Frame(self.root)
        #Frame title
        self.title1_label = tk.Label(text='Running Process', font = ('Calibri', 12, 'bold'),fg='#149998')
        self.title1_label.pack( padx=5, pady=5,)
        frame2.pack()

        # Label and entry for maximum number of processes
        self.max_processes_label = tk.Label(frame2, text='Maximum number of processes:')
        self.max_processes_label.pack(pady=10)
        self.max_processes_entry = tk.Entry(frame2) # fixed variable name
        self.max_processes_entry.pack(pady=10) # fixed variable name

        #button to run simulations in parallel
        self.submit_button = tk.Button(frame2, text='Run', command=self.run_function2,font=('Calibri', 12, 'bold'))
        self.submit_button.pack( padx=5, pady=5)

        return frame2

    def create_frame3(self):
        frame3 = tk.Frame(self.root)
        #Frame title
        self.title_label = tk.Label(text='Gathering & Reading Output', font=('Calibri', 12, 'bold'),fg='#149998')
        self.title_label.pack(padx=5, pady=5)
        frame3.pack()
```

```

        #button to change RSM files to text files
        self.rename_button = tk.Button(frame3, text='Change RSM files format', command=self.rename_file, font=('Calibri', 12, 'bold'))
        self.rename_button.pack(pady=10)

        #button to save production profiles
        self.submit_button = tk.Button(frame3, text='Save production profiles', command=self.run_function3,font=('Calibri', 12, 'bold'))
        self.submit_button.pack(pady=10)

        return frame3



    #Function to change RSM extension to text
    def rename_file(self):
        # Get the input values from the entry fields
        input_str = self.irange_entry.get()
        input_list = input_str.split(',')
        input_list = [int(num.strip()) for num in input_list]
        jrange = int(self.jrange_entry.get())

        # Iterate over the input values
        for i in input_list :
            for j in range(1, jrange+1):
                my_file = f'E:/OLYMPUS_{i}/OLYMPUS_{i}_{j}/OLYMPUS_{i}/OLYMPUS_{i}.RSM'
                # Check if the file exists
                if not os.path.exists(my_file):
                    # RSM file doesn't exist, so pass and go to the next one
                    continue
                # Rename the file by changing the extension to '.txt'
                base = os.path.splitext(my_file)[0]
                os.rename(my_file, base +'_RSM'+ '.txt')




    # Function to copy files from  the source address and making new sets of samples and realiaztions based on user input
    def run_function1(self):
```

```
        # Get the input values from the entry fields
        input_str = self.irange_entry.get()
        input_list = input_str.split(',')
        input_list = [int(num.strip()) for num in input_list]
        jrange = int(self.jrange_entry.get())
        LL = self.LL_entry.get()
        HL = self.HL_entry.get()


        # Iterate over the input values
        for i in input_list:
            for j in range(1, jrange+1):
                # Define the directory paths
                directory_path = f'C:/Test2/OLYMPUS_{i}/OLYMPUS_{i}_{j}'
                os.makedirs(directory_path, exist_ok=True)

                # Copy source directories to destination directories
                src = f'C:/Users/sarah/OneDrive/Desktop/Test 3/OLYMPUS_{i}/OLYMPUS_{i}'
                dest = f'C:/Test2/OLYMPUS_{i}/OLYMPUS_{i}_{j}/OLYMPUS_{i}'
                shutil.copytree(src, dest)

                src = r'C:/Users/sarah/OneDrive/Desktop/Test 3/OLYMPUS'
                dest = f'C:/Test2/OLYMPUS_{i}/OLYMPUS_{i}_{j}/OLYMPUS'
                shutil.copytree(src, dest)




        Row = 11
        Column = 3
        matrix = np.zeros([Row,Column])
        #Write production years and BHP limitations in a matrix
        for i in range (0, Row):
          matrix[i][0]=i+1
          matrix[i][1]=LL
          matrix[i][2]=HL


        # Function to open the schadule file and modify BHP pressure for production wells based on user
```

```
        input
    def replace_line(file_name, line_num, text):
        # Read all lines from the file
        lines = open(file_name, 'r').readlines()
        # Replace the line at the specified line number with the given text
        lines[line_num] = text
        # Open the file in write mode
        out = open(file_name, 'w')
        # Write the modified lines back to the file
        out.writelines(lines)
        # Close the file
        out.close()

    # Iterate over the input values
    for i in input_list:
      for j in range (1,jrange+1):

        for k in range (0, 11) :
         # Open the file for reading
         with open(f'C:/Test2/OLYMPUS_{i}/OLYMPUS_{i}_{j}/OLYMPUS/OLYMPUS_SCH.INC', 'r') as my_file:

            #read all lines in a list
            keyword = 'WCONPROD'
            lines = my_file.readlines()
            # Iterate over the lines in the file
            for line in lines:
              # Check if the keyword is present in the line
              if line.find(keyword) != -1:
                kw_line=lines.index(line)
                well_line=int(matrix[k][0]+kw_line)
                with open(f'C:/Test2/OLYMPUS_{i}/OLYMPUS_{i}_{j}/OLYMPUS/OLYMPUS_SCH.INC') as f:
                  particular_line = f.readlines()[well_line]
                  #Convert string to array
                  x = particular_line.split()
                  # Generate a new random BHP value within the specified limits
                  bhp=int(random.uniform(matrix[k][1], matrix[k][2]))
                  #New BHP value as string
                  x[4]= '%s '%bhp
                  #convert array to string
                  x1=' '.join(x)
```

```
                        x =f'  {x1}\n'
                        # Call the replace_line function to replace the line in the file
                        replace_line( f'C:/Test2/OLYMPUS_{i}/OLYMPUS_{i}_{j}/OLYMPUS/OLYMPUS_SCH.INC', well_line, x)

    # Progress window shows  the loading bar for simulations parallel run
    def open_progress_window(self):
        self.progress_window = tk.Toplevel(self.root)
        self.progress_window.title('Progress')
        self.progress_window.geometry('500x80')

        self.status_label = tk.Label(self.progress_window, text='Launching Eclipse in Parallel')
        self.status_label.pack()

        self.status_label2 = tk.Label(self.progress_window, text='Wait for the bar to be filled, then close the progress window and continue.')
        self.status_label2.pack()

        self.progress_bar = ttk.Progressbar(self.progress_window, length=200, mode='determinate')
        self.progress_bar.pack()

        return self.progress_window

    # Function to update the loading bar
    def update_progress(self, value):
        self.progress_bar['value'] = value
        self.progress_window.update()

    # Function for parallel run in Eclipse
    def run_function2(self):
        # Get the maximum number of processes from the entry field
        MAX_PROCESSES = int(self.max_processes_entry.get())

        # Get the initial time
        t0 = perf_counter()

        # Open the progress window and bring it to front
        self.progress_window = self.open_progress_window()
        self.progress_window.lift()

        # Get the input values from the entry fields
```

```
        input_str = self.irange_entry.get()
        input_list = input_str.split(',')
        input_list = [int(num.strip()) for num in input_list]
        jrange = int(self.jrange_entry.get())

       # Generate the parallel run input based on the input values
        parallel_run_input = [
            os.path.join(f'C:/Test2/OLYMPUS_{i}/OLYMPUS_{i}_{j}/OLYMPUS_{i}/OLYMPUS_{i}.DATA')
            for i in input_list
            for j in range(1, jrange + 1)
       ]

        # Set the initial value and maximum value of the progress bar
        self.progress_bar['value'] = 0
        self.progress_bar['maximum'] = len(parallel_run_input)

        # Update the progress window before starting the calculations
        self.progress_window.update()
        # Start the parallel execution using a pool of processes
        with Pool(processes=min(len(parallel_run_input), MAX_PROCESSES)) as pool:
            results = []
            for i, _ in enumerate(pool.imap_unordered(run1, parallel_run_input)):
                results.append(_)
                self.update_progress(i + 1)
        # Update the status label to indicate the completion time
        self.status_label.config(text=f'Finished in {perf_counter() - t0:.2f} seconds')


        # Update the progress window after the calculation
        self.progress_window.update()



    def run_function3(self):
            # Get input values from the entry fields
            input_str = self.irange_entry.get()
            input_list = input_str.split(',')
            input_list = [int(num.strip()) for num in input_list]
```

```
        jrange = int(self.jrange_entry.get())

        for i in input_list :
            for j in range(1, jrange+1):
                # Define the path to the RSM file
                rsm_file = f'E:/OLYMPUS_{i}/OLYMPUS_{i}_{j}/OLYMPUS_{i}/OLYMPUS_{i}_RSM.txt'

                if not os.path.exists(rsm_file):
                    # RSM file doesn't exist, so pass and go to the next one
                 continue



                # Initialize production profile arrays
                production_profile = np.zeros(20)

                with open(rsm_file, 'r') as f:
                    lines = f.readlines()

                    for d in range(0, 20):
                        # Extract cumulative oil production data
                        keyword = 'FOPT'
                        kw_line = next((index for (index, line) in enumerate(lines) if keyword in line), None)
                        print(f'kw_line: {kw_line}')
                        wanted_line = 2 + int(0) + kw_line
                        particular_line = lines[wanted_line]
                        x = particular_line.split()
                        search_string = '*10**3'

                        if any(search_string in element for element in x):
                            wanted_line1 = 7 + d + kw_line
                            particular_line = lines[wanted_line1]
                            x1 = particular_line.split()
                            production_profile[d] = float(x1[8]) * 10**3
                            #
                        else:
                           wanted_line1 = 6 + d +
```

```
                                kw_line
                            particular_line = lines[wanted_line1]
                            x1 = particular_line.split()
                            production_profile[d] = float(x1[8])



            # Create a DataFrame to store the production profile data
            df_oil = pd.DataFrame(data={'Year': range(1, 21), 'cumulative oil production(Sm3)': production_profile})

            production_profile1 = np.zeros(20)
            with open(rsm_file, 'r') as f:

                lines = f.readlines()
                for d in range(0, 20):
                    # Extract cumulative water production data
                    keyword1 = 'FWPT'
                    kw_line1 = next((index for (index, line) in enumerate(lines) if keyword1 in line), None)
                    #kw_line1 = next((index for (index, line) in enumerate(lines) if keyword1 in line), None)
                    print(f'kw_line1: {kw_line1}')
                    wanted_line1 = 2 + int(0) + kw_line1
                    particular_line1 = lines[wanted_line1]
                    x1 = particular_line1.split()
                    search_string = '*10**3'

                    if any(search_string in element for element in x1):
                        wanted_line2 = 7 + d + kw_line1
                        particular_line2 = lines[wanted_line2]
                        x2 = particular_line2.split()
```

```
                        #print (x)
                        production_profile1[d] = float(x2[6]) * 10**3
                    else:
                        wanted_line2 = 6 + d + kw_line1
                        particular_line2 = lines[wanted_line2]
                        x2 = particular_line2.split()
                        #print (x)
                        production_profile1[d] = float(x2[6])


        # Create a DataFrame to hold the production profile data

        df_water = pd.DataFrame(data={'Year': range(1, 21), 'cumulative water production(Sm3)': production_profile1})


        production_profile2 = np.zeros(20)
        with open(rsm_file, 'r') as f:
            lines = f.readlines()
            for d in range(0, 20):
                # Extract cumulative gas production data
                keyword1 = 'FGPT'
                kw_line1 = next((index for (index, line) in enumerate(lines) if keyword1 in line), None)
                #kw_line1 = next((index for (index, line) in enumerate(lines) if keyword1 in line), None)
                print(f'kw_line1: {kw_line1}')
                wanted_line1 = 2 + int(0) + kw_line1
                particular_line1 = lines[wanted_line1]
                x1 = particular_line1.split()
                search_string = '*10**3'

                if any(search_string in element
```

```
                        for element in x1):
                            wanted_line2 = 7 + d + kw_line1
                            particular_line2 = lines[wanted_line2]
                            x2 = particular_line2.split()
                            #print (x)
                            production_profile1[d] = float(x2[6]) * 10**3
                    else:
                            wanted_line2 = 6 + d + kw_line1
                            particular_line2 = lines[wanted_line2]
                            x2 = particular_line2.split()
                            #print (x)
                            production_profile1[d] = float(x2[6])

                # Create a DataFrame to hold the production profile data

                df_gas = pd.DataFrame(data={'Year': range(1, 21), 'cumulative gas production(Sm3)': production_profile2})


                Proprof = f'E:/OLYMPUS_{i}/OLYMPUS_{i}_{j}/OLYMPUS_{i}/production_profiles_{i}_{j}.xlsx'

            # Save production profiles to an Excel file
                with pd.ExcelWriter (f'E:/OLYMPUS_{i}/OLYMPUS_{i}_{j}/OLYMPUS_{i}/production_profiles_{i}_{j}.xlsx') as writer:
                    df_water.to_excel(writer, sheet_name='Water', index=False)
                    df_oil.to_excel(writer, sheet_name='Oil', index=False)
                    df_gas.to_excel(writer, sheet_name='Gas', index=False)

                # Read the Excel file and perform
```

```
calculations
        df = pd.read_excel(Proprof, sheet_name='Oil')

        # Calculate the yearly oil production
        yearly_oil_production = df['cumulative oil production(Sm3)'] - df['cumulative oil production(Sm3)'].shift(fill_value=0)

        # Add the new column to the DataFrame
        df['yearly oil production(Sm3)'] = yearly_oil_production


        # Write the DataFrame with the new column to the same Excel file and sheet
        with pd.ExcelWriter(Proprof, engine='openpyxl', mode='a', if_sheet_exists='replace') as writer:
            df.to_excel(writer, sheet_name='Oil', index=False)

        # Repeat the same steps for water and gas
        df = pd.read_excel(Proprof, sheet_name='Water')


        yearly_water_production = df['cumulative water production(Sm3)'] - df['cumulative water production(Sm3)'].shift(fill_value=0)


        df['yearly water production(Sm3)'] = yearly_water_production



        with pd.ExcelWriter(Proprof, engine='openpyxl', mode='a', if_sheet_exists='replace') as writer:
            df.to_excel(writer, sheet_name='Water', index=False)

        df = pd.read_excel(Proprof, sheet_name='
```

```
Gas')


                yearly_gas_production = df['cumulative gas production(Sm3)'] - df['cumulative gas production(Sm3)'].shift(fill_value=0)


                df['yearly gas production(Sm3)'] = yearly_gas_production
                # Write the DataFrame to an Excel file and save it


                with pd.ExcelWriter(Proprof, engine='openpyxl', mode='a', if_sheet_exists='replace') as writer:
                    df.to_excel(writer, sheet_name='Gas', index=False)


        results = {}


        for l in input_list:
            for s in range(1, jrange+1):
                proprof2 =f'E:/OLYMPUS_{l}/OLYMPUS_{l}_{s}/OLYMPUS_{l}/production_profiles_{l}_{s}.xlsx'

                if not os.path.exists(proprof2):
                    # RSM file doesn't exist, so pass and go to the next one
                    continue



                # Read the Excel file and extract the 'oil' sheet
                df = pd.read_excel(proprof2, sheet_name='Oil')

                # Calculate the sum of the values in the 'oil production' column
                total_oil = df['yearly oil production(Sm3)'].sum()

```

```
                # Store the result in the dictionary
                name = 'OLYMPUS'+f'_{l}_{s}'
                results[name] = 100*(total_oil/49000000)

                # Create a new DataFrame from the results dictionary
                new_df = pd.DataFrame.from_dict(results, orient='index', columns=['Recovery factor'])

                # Add a 'name' column
                new_df['name'] = new_df.index.str.replace('OLYMPUS_', 'OLYMPUS_', regex=True).str.replace('.xlsx', '', regex=False)


                # Write the new DataFrame to a new Excel file
                output_file_name = 'E:/OLYMPUS_Recovery.xlsx'
                new_df.to_excel(output_file_name, index=False, header=True)

                # Read the data from the Excel file
                input_file_name = 'E:/OLYMPUS_Recovery.xlsx'
                df = pd.read_excel(input_file_name)

                # Group the dataframe by the first part of the name (i.e. OLYMPUS_{i})
                groups = df.groupby(df['name'].str.split('_', expand=True)[1])

                # Write each group to a separate sheet in a new Excel file
                output_file_name = 'E:/Recovery_separated.xlsx'
                with pd.ExcelWriter(output_file_name) as writer:
                    for name, group in groups:
                        sheet_name = f'OLYMPUS_{name}'
                        group.to_excel(writer, sheet_name=sheet_name, index=False)
```

```

        # Create an empty list to store the dataframes for oil, water and gas production
        oil_df_list = []
        water_df_list = []
        gas_df_list = []

        for i in input_list:
            for j in range (1, jrange+1):
                proprof3 =f'E:/OLYMPUS_{i}/OLYMPUS_{i}_{j}/OLYMPUS_{i}/production_profiles_{i}_{j}.xlsx'

                if not os.path.exists(proprof3):
                    # RSM file doesn't exist, so pass and go to the next one
                  continue


                # Read the oil, water and gas production data from the Excel file
                oil_df = pd.read_excel(proprof3, sheet_name='Oil')
                oil_column_name = f'Oil_production_{i}_{j}'
                oil_df = oil_df.iloc[:, 2].rename(oil_column_name)
                oil_df_list.append(oil_df)

                water_df = pd.read_excel(proprof3, sheet_name='Water')
                water_column_name = f'Water_production_{i}_{j}'
                water_df = water_df.iloc[:, 2].rename(water_column_name)
                water_df_list.append(water_df)

                gas_df = pd.read_excel(proprof3, sheet_name='Gas')
                gas_column_name = f'Gas_production_{i}_{j}'
                gas_df = gas_df.iloc[:, 2].rename(gas_column_name)
                gas_df_list.append(gas_df)


            # Concatenate all oil, water and gas dataframes into a single dataframe
```

```
                oil_df_final = pd.concat(oil_df_list, axis=1)

                water_df_final = pd.concat(water_df_list, axis=1)

                gas_df_final = pd.concat(gas_df_list, axis=1)

                # Write the oil, water and gas dataframes to a new Excel file
                output_file = f'E:/production_summary_OLYMPUS{i}.xlsx'


                with pd.ExcelWriter(output_file) as writer:
                    oil_df_final.index += 1
                    water_df_final.index += 1
                    gas_df_final.index += 1

                    # Write to Excel file
                    oil_df_final.to_excel(writer, sheet_name='Oil_production', index=True)
                    water_df_final.to_excel(writer, sheet_name='Water_production', index=True)
                    gas_df_final.to_excel(writer, sheet_name='Gas_production', index=True)







    def run(self):

        # The `run` method is a part of a class and is used to start the main event loop of the GUI application.
        # It runs indefinitely until the GUI window is closed by the user.
        self.root.mainloop()


if __name__ == '__main__':
```

```
    gui = MyGUI()
    gui.run()
    # This is the entry point of the script when it is run as a standalone program.
    # It creates an instance of the ‘MyGUI‘ class and calls its ‘run‘ method to start the GUI application.
    # The ‘if __name__ == '__main__':‘ condition ensures that this block of code is only executed when the script is run directly,
    # and not when it is imported as a module.
```

5.2 Appendix A - 2: NPV Calculation and Result visualization

```

import os
import numpy as np
import pandas as pd
import tkinter as tk
import seaborn as sns
import openpyxl
import matplotlib.pyplot as plt
import re
from tkinter import filedialog


def NPV_summary(self):
    input_str = self.irange_entry.get()
    input_list = input_str.split(",")
    input_list = [int(num.strip()) for num in input_list]
    jrange = int(self.jrange_entry.get())
    # Create a new workbook to store the results
    result_workbook = openpyxl.Workbook()
    result_worksheet = result_workbook.active

    # Set the column header for the result worksheet
    result_worksheet['A1'] = 'Name'
    result_worksheet['B1'] = 'Value($M))'

    # Iterate through the input_list and process each Excel file
    for D in input_list:
        for M in range(1, jrange+1):

            dir_path =f"E:/OLYMPUS_{D}/OLYMPUS_{D}_{M}/OLYMPUS_{D}"
```

```

            if not os.path.exists(dir_path):
                # RSM file doesn't exist, so pass and go to the next one
                continue

            # Construct the file path and load the Excel file
            npv_file_path = f"{dir_path}/NPV_Calc_{D}_{M}.xlsx"
            if not os.path.exists(npv_file_path):
                # RSM file doesn't exist, so pass and go to the next one
                continue
            npv_workbook = openpyxl.load_workbook(npv_file_path)

            # collect the final NPV value
            # Get the active worksheet from the NPV workbook
            npv_worksheet = npv_workbook.active

            # Variable to store the last non-zero cell in the worksheet
            last_non_zero_cell = None

            # Iterate over rows in reverse order, starting from the last row
            for i in range(npv_worksheet.max_row, 0, -1):
                # Get the cell at the last column of the current row
                cell = npv_worksheet.cell(row=i, column=npv_worksheet.max_column)
                # Get the value of the cell
                value = cell.value

                # Check if the value is non-zero
                if value != 0:
                    # Store the reference to the last non-zero cell
                    last_non_zero_cell = cell
                    # Exit the loop as we have found the last non-zero value
                    break
            # Check if a non-zero cell was found
            if last_non_zero_cell is not None:
                    # Get the value from the last non-zero cell
```

```
                    value = last_non_zero_cell.value


            # Write the result to the result worksheet
            name = f"OLYMPUS_{D}_{M}"
            row = (name, value)
            result_worksheet.append(row)

    # Save the result workbook to a file
    result_workbook.save('E:/OLYMPUS_NPV.xlsx')

    # Write the DataFrame to the same Excel file
    input_file_path = 'E:/OLYMPUS_NPV.xlsx'
    df = pd.read_excel(input_file_path)
    df.to_excel('E:/OLYMPUS_NPV.xlsx', index=False, header=True)

    # Group the DataFrame by the second component of the 'name' column
    groups = df.groupby(df['Name'].str.split('_', expand=True)[1])

    # Write each group to a separate sheet in a new Excel file
    output_file_name = 'E://NPV_separated.xlsx'
    with pd.ExcelWriter(output_file_name) as writer:
        for name, group in groups:

            # Create a sheet name for the current group
            sheet_name = f'OLYMPUS_{name}'

            # Write the current group to a new sheet in the Excel file
            group.to_excel(writer, sheet_name=sheet_name, index=False)



class MyGUI:
    def __init__(self):
        self.root = tk.Tk()
        self.frame1 = self.create_frame1()
        self.frame4 = self.create_frame4()
        self.frame5 = self.create_frame5()
        #self.add_logo ()
        self.root.geometry("900x700")
        self.root.title("PPG")  # Set the window title
```

```
        #self.root.iconbitmap("C:/Users/sarah/OneDrive/Desktop/Thesis/NTNU_logo_400x400.ico")  # Set the window icon

        self.excel_file = None
        self.sheet_name = None
        self.sheet_data = None


    def create_frame1(self):
        frame1 = tk.Frame(self.root)
        frame1.pack()

        # First column
        col1 = tk.Frame(frame1)
        col1.pack(side=tk.LEFT, padx=10, pady=5)

        self.title1_label = tk.Label(col1, text="General Setup", font = ('Calibri', 12, 'bold'),fg='#149998')
        self.title1_label.pack(anchor=tk.CENTER, padx=5, pady=5,)

        # Label and entry for irange
        self.irange_label = tk.Label(col1, text="Enter a comma-separated list of realizations:")
        self.irange_label.pack(anchor=tk.W, padx=5, pady=5,)
        self.irange_entry = tk.Entry(col1)
        self.irange_entry.insert(tk.END, "40,50")
        self.irange_entry.pack(anchor=tk.W, padx=5, pady=5,)

        # Label and entry for jrange
        self.jrange_label = tk.Label(col1, text='Enter the number of BHP samples:')
        self.jrange_label.pack(anchor=tk.W, padx=10, pady=5, )
        self.jrange_entry = tk.Entry(col1)
        self.jrange_entry.insert(tk.END, "1")
        self.jrange_entry.pack(anchor=tk.W, padx=10, pady=5, )

        # Label and entry for pre production years
        self.well_label = tk.Label(col1, text='Enter the number years before starting the production:')
        self.well_label.pack(anchor=tk.W, padx=10, pady=5,)
        self.well_entry = tk.Entry(col1)
```

```
        self.well_entry.insert(tk.END, "5")
        self.well_entry.pack(anchor=tk.W, padx=10, pady=5, )

        # Label and entry for production years
        self.pro_label = tk.Label(col1, text='Enter the number of production years:')
        self.pro_label.pack(anchor=tk.W, padx=10, pady=5, )
        self.pro_entry = tk.Entry(col1)
        self.pro_entry.insert(tk.END, "25")
        self.pro_entry.pack(anchor=tk.W, padx=10, pady=5,)


        # Second column
        col2 = tk.Frame(frame1)
        col2.pack(side=tk.LEFT, padx=10, pady=5)

        self.title2_label = tk.Label(col2, text="NPV values", font = ('Calibri', 12, 'bold'),fg='#149998')
        self.title2_label.pack( anchor=tk.W, padx=10, pady=5,)

        # Label and entry for drilling cost
        self.drillcost_label = tk.Label(col2, text='Drilling cost for production wells($M)')
        self.drillcost_label.pack(anchor=tk.W, padx=10, pady=5)
        self.drillcost_entry = tk.Entry(col2)
        self.drillcost_entry.insert(tk.END, '100')
        self.drillcost_entry.pack(anchor=tk.W, padx=10, pady=5)

        # Label and entry for piping cost
        self.pipcost_label = tk.Label(col2, text='Piping cost($M)')
        self.pipcost_label.pack(anchor=tk.W, padx=10, pady=5)
        self.pipcost_entry = tk.Entry(col2)
        self.pipcost_entry.insert(tk.END, "500")
        self.pipcost_entry.pack(anchor=tk.W, padx=10, pady=5)

        # Label and entry for piping years
        self.pipt_label = tk.Label(col2, text='Piping years')
        self.pipt_label.pack(anchor=tk.W, padx=10, pady=5)
        self.pipt_entry = tk.Entry(col2)
        self.pipt_entry.insert(tk.END, "3")
```

```
        self.pipt_entry.pack(anchor=tk.W, padx=10, pady=5)

        # Label and entry for manifold cost
        self.mfcost_label = tk.Label(col2, text='Manifold cost($M)')
        self.mfcost_label.pack(anchor=tk.W, padx=10, pady=5)
        self.mfcost_entry = tk.Entry(col2)
        self.mfcost_entry.insert(tk.END, "200")
        self.mfcost_entry.pack(anchor=tk.W, padx=10, pady=5)

        # Label and entry for number of manifolds
        self.mfnum_label = tk.Label(col2, text='Number of manifolds')
        self.mfnum_label.pack(anchor=tk.W, padx=10, pady=5)
        self.mfnum_entry = tk.Entry(col2)
        self.mfnum_entry.insert(tk.END, "3")
        self.mfnum_entry.pack(anchor=tk.W, padx=10, pady=5)

        # Label and entry for fixed OPEX cost
        self.OPEX_label = tk.Label(col2, text='Enter the OPEX')
        self.OPEX_label.pack(anchor=tk.W, padx=10, pady=5)
        self.OPEX_entry = tk.Entry(col2)
        self.OPEX_entry.insert(tk.END, "100")
        self.OPEX_entry.pack(anchor=tk.W, padx=10, pady=5)

        col3 = tk.Frame(frame1)
        col3.pack(side=tk.LEFT, padx=10, pady=5)

        # Label and entry for oil price
        self.oil_label = tk.Label(col3, text='Oil price($/Sm3)')
        self.oil_label.pack(anchor=tk.W, padx=10, pady=5)
        self.oil_entry = tk.Entry(col3)
        self.oil_entry.insert(tk.END, "760")
        self.oil_entry.pack(anchor=tk.W, padx=10, pady=5)

        # Label and entry for gas price
        self.gas_label = tk.Label(col3, text='gas price($/Sm3)')
        self.gas_label.pack(anchor=tk.W, padx=10, pady=5)
        self.gas_entry = tk.Entry(col3)
        self.gas_entry.insert(tk.END, "76")
        self.gas_entry.pack(anchor=tk.W, padx=10, pady=5)

        # Label and entry for water expenses
```

```
        self.wcost_label = tk.Label(col3, text='Water cost ($/Sm3)')
        self.wcost_label.pack(anchor=tk.W, padx=10, pady=5)
        self.wcost_entry = tk.Entry(col3)
        self.wcost_entry.insert(tk.END, "50")
        self.wcost_entry.pack(anchor=tk.W, padx=10, pady=5)

        # Label and entry for interest rate
        self.interest_label = tk.Label(col3, text='interest rate')
        self.interest_label.pack(anchor=tk.W, padx=10, pady=5)
        self.interest_entry = tk.Entry(col3)
        self.interest_entry.insert(tk.END, "5")
        self.interest_entry.pack(anchor=tk.W, padx=10, pady=5)


        return frame1



    # Frame for NPV button
    def create_frame4(self):
        frame4 = tk.Frame(self.root)
        frame4.pack()
        self.NPV_button = tk.Button(frame4, text="NPV ", command=self.run_function4,font=('Calibri', 12, 'bold'))
        self.NPV_button.pack(pady=10)
        return frame4


    # Frame for result visualization
    def create_frame5(self):
        frame5 = tk.Frame(self.root)
        self.title1_label = tk.Label(text="Result Visualization", font = ('Calibri', 12, 'bold'), fg='#149998')
        self.title1_label.pack( padx=5, pady=5,)
        frame5.pack()

        self.selected_file = tk.StringVar()
        self.selected_file_label = tk.Label(frame5, textvariable=self.selected_file)
        self.selected_file_label.grid(row=0, column=2 ,padx=10, pady=10)

```

```
        self.selected_sheet = tk.StringVar()
        self.selected_sheet_label = tk.Label(frame5, textvariable=self.selected_sheet)
        self.selected_sheet_label.grid(row=1, column=2, padx=10, pady=10,)

        # create widgets
        self.file_label = tk.Label(frame5, text="Excel file:")
        self.file_button = tk.Button(frame5, text="Select file", command=self.select_file)
        self.sheet_label = tk.Label(frame5, text="Sheet:")
        self.sheet_var = tk.StringVar()
        self.sheet_dropdown = tk.OptionMenu(frame5, self.sheet_var, [])
        self.sheet_dropdown.configure(state="disabled")
        self.seperated_NPV_button = tk.Button(frame5, text="Seperated NPV Plot", command=self.NPV_sep)
        self.seperated_RF_button = tk.Button(frame5, text="Seperated RF Plot", command=self.RF_sep)
        self.total_NPV_button = tk.Button(frame5, text="Total NPV Plot", command=self.NPV_tot)
        self.total_RF_button = tk.Button(frame5, text="Total RF Plot", command=self.RF_tot)

        # layout widgets
        self.file_label.grid(row=0, column=0, padx=10, pady=10)
        self.file_button.grid(row=0, column=1, padx=10, pady=10)
        self.sheet_label.grid(row=1, column=0, padx=10, pady=10)
        self.sheet_dropdown.grid(row=1, column=1, padx=5, pady=5)
        self.seperated_NPV_button.grid(row=20, column=1, padx=20, pady=20)
        self.seperated_RF_button.grid(row=20, column=2, padx=20, pady=20)
        self.total_NPV_button.grid(row=20, column=3, padx=20, pady=20)
        self.total_RF_button.grid(row=20, column=4, padx=20, pady=20)


        return frame5

    def select_file(self):
```

```
        self.excel_file = filedialog.askopenfilename(filetypes=[("Excel files", "*.xlsx")])
        if self.excel_file:
            self.update_sheet_dropdown()
            self.sheet_dropdown.configure(state="normal")

            # Add the following lines to update the selected file label
            self.selected_file.set("Selected file: " + self.excel_file)

    def update_sheet_dropdown(self):
        workbook = openpyxl.load_workbook(filename=self.excel_file)
        sheets = workbook.sheetnames
        self.sheet_dropdown["menu"].delete(0, "end")
        for sheet in sheets:
            self.sheet_dropdown["menu"].add_command(label=sheet, command=lambda s=sheet: self.select_sheet(s))

    def select_sheet(self, sheet_name):
        self.sheet_name = sheet_name
        self.selected_sheet.set("Selected sheet: " + sheet_name)

    def NPV_sep(self):
        if self.sheet_name and self.excel_file:
            workbook = openpyxl.load_workbook(filename=self.excel_file)
            sheet = workbook[self.sheet_name]
            data = []
            for row in sheet.iter_rows(min_row=2, values_only=True):
                data.append(row)
                # Create the first plot (KDE plot)
            df = pd.DataFrame(data, columns=["name", "Value($M)"])

            # Create the first plot (KDE plot)
            kde_plot = sns.displot(data=df, x="Value($M)", kind="kde", height=6, aspect=1.4, warn_singular=False)


            # Create the second plot (ECDF plot)
            ecdf_plot = sns.displot(data=df, x='Value($M)', kind="ecdf", height=6, aspect=1.4)
```

```
            # Create the directory to save the plots in, if it doesn't already exist
            if not os.path.exists("my_plots"):
                os.makedirs("my_plots")

            # Save the plots in the 'my_plots' directory
            kde_plot.savefig(f'E:/my_plots/{self.sheet_name}_NPV_sep_kde_plot.png')
            ecdf_plot.savefig(f"E:/my_plots/{self.sheet_name}_NPV_sep_ecdf_plot.png")

    def RF_sep(self):
        if self.sheet_name and self.excel_file:
            workbook = openpyxl.load_workbook(filename=self.excel_file)
            sheet = workbook[self.sheet_name]
            data = []
            for row in sheet.iter_rows(min_row=2, values_only=True):
                data.append(row)
                # Create the first plot (KDE plot)
            df = pd.DataFrame(data, columns=["Recovery factor", "name"])

            # Create the first plot (KDE plot)
            kde_plot = sns.displot(data=df, x="Recovery factor", kind="kde", height=6, aspect=1.4, common_norm=False)


            # Create the second plot (ECDF plot)
            ecdf_plot = sns.displot(data=df, x="Recovery factor", kind="ecdf", height=6, aspect=1.4,common_norm=False)

            # Create the directory to save the plots in, if it doesn't already exist
            if not os.path.exists("my_plots"):
                os.makedirs("my_plots")

            # Save the plots in the 'my_plots' directory
            kde_plot.savefig(f'E:/my_plots/{self.sheet_name}RF_sep_kde_plot.png')
            ecdf_plot.savefig(f"E:/my_plots/{self.sheet_name}RF_sep_ecdf_plot.png")


```

```
    def NPV_tot(self):
        if self.sheet_name and self.excel_file:
            workbook = openpyxl.load_workbook(filename=self.excel_file)
            sheet = workbook[self.sheet_name]
            data = []
            for row in sheet.iter_rows(min_row=2, values_only=True):
                data.append(row)

            # Create filtered dataframe
            filtered_df = pd.DataFrame(data, columns=['Name', 'Value($M)'])
            filtered_df['OLYMPUS_I'] = filtered_df['Name'].str.extract(r'(OLYMPUS_\d+)')

           # Create the KDE plot
            kde_plot = sns.displot(data=filtered_df, x="Value($M)", hue="OLYMPUS_I", kind="kde", height=6, aspect=1.4,common_norm=False )
            kde_plot.savefig('E:/my_plots/NPV_OLYMPUS_sets_pdf.png')
            cdf_plot = sns.displot(data=filtered_df, x="Value($M)", hue="OLYMPUS_I", kind="ecdf", height=6, aspect=1.4,common_norm=False)
            cdf_plot.savefig('E:/my_plots/NPV_OLYMPUS_sets_cdf.png')



    def RF_tot(self):
            if self.sheet_name and self.excel_file:
                workbook = openpyxl.load_workbook(filename=self.excel_file)
                sheet = workbook[self.sheet_name]
                data = []
                for row in sheet.iter_rows(min_row=2, values_only=True):
                    data.append(row)
                    # Create the first plot (KDE plot)
                df = pd.DataFrame(data, columns=["Recovery factor", "name"])
                df['OLYMPUS_I'] = df['name'].str.extract(r'(OLYMPUS_\d+)')

               # Create the KDE plot
                kde_plot = sns.displot(data=df, x="Recovery factor", hue="OLYMPUS_I", kind="kde",
```

```
                        height=6, aspect=1.4)
                    kde_plot.savefig('E:/my_plots/RF_OLYMPUS_sets_pdf.png')

                    cdf_plot = sns.displot(data=df, x="Recovery factor", hue="OLYMPUS_I", kind="ecdf", height=6, aspect=1.4)
                    cdf_plot.savefig('E:/my_plots/RF_OLYMPUS_sets_cdf.png')
                    #g = sns.FacetGrid(df, col='OLYMPUS_set', col_wrap=3, height=4)

                    # Plot the PDF and ECDF for each OLYMPUS set in the FacetGrid
                    #g.map(sns.histplot, 'Recovery factor', kde=False, stat='density', alpha=0.5, color='b',common_norm=False)
                    #g.map(sns.ecdfplot, 'Recovery factor', alpha=0.5, color='r',common_norm=False)

                    # Set titles for each plot
                    #for ax in g.axes.flat:
                     #   ax.set_title(ax.get_title().replace("OLYMPUS_set = ", "OLYMPUS Set "))

                     # Save the plot
                    #plt.savefig('E:/my_plots/RF_OLYMPUS_sets_pdf_ecdf.png')
                    #plt.show()




        def browse_directory(self):
            directory_path = filedialog.askdirectory() +'/'
            print("Selected directory:", directory_path)
            self.directory_path_var.set(directory_path)





        def run_function4(self):


            input_str = self.irange_entry.get()
            input_list = input_str.split(",")
```

```
        input_list = [int(num.strip()) for num in input_list]
        jrange = int(self.jrange_entry.get())
        for l in input_list:
            for s in range(1, jrange+1):
                dir_path =f'E:/OLYMPUS_{l}/OLYMPUS_{l}_{s}/OLYMPUS/OLYMPUS_SCH.INC'

                if not os.path.exists(dir_path):
                    # RSM file doesn't exist, so pass and go to the next one
                    continue
                with open(dir_path, 'r') as f:
                    text = f.read()
                    # matches "Prod_" followed by one or more digits
                    pattern = r"PROD-(\d+)"

                # find all matches of the pattern in the text( to extract number of production wells)
                matches = re.findall(pattern, text)

                if matches:
                    last_match = matches[-1]  # select the last match
                    PROD_wells = int(last_match)  # convert the string to an integer

                else:
                    print("No production well  found")

                with open(dir_path, 'r') as f:
                    text = f.read()
                    pattern = r"INJ-(\d+)"  # matches "Prod_" followed by one or more digits

                    # find all matches of the pattern in the text
                    matches = re.findall(pattern, text)

                if matches:
                    # select the last match
                    last_match = matches[-1]
                    # convert the string to an integer
                    INJ_wells = int(last_match)

```

```

                file_name = f"E:/OLYMPUS_{l}/OLYMPUS_{l}_{s}/OLYMPUS_{l}/production_profiles_{l}_{s}.xlsx"
                if not os.path.exists(file_name):
                    # RSM file doesn't exist, so pass and go to the next one
                    print('file doesnt exist')
                    continue


                # set the columns to extract
                #sheet1_columns = "yearly oil production(Sm3)"
                #sheet2_columns = "yearly water production(Sm3)"
                #sheet3_columns = "yearly gas production(Sm3)"

                # Read the Excel file into a dictionary of DataFrames
                df_oil = pd.read_excel(file_name, sheet_name=["Oil"])
                df_gas = pd.read_excel( file_name, sheet_name=['Gas'])
                df_water = pd.read_excel( file_name, sheet_name=[ "Water"])


                end_idx = int(self.pro_entry.get())


                df_oil_dict = pd.read_excel(file_name, sheet_name=["Oil"])
                df_oil = df_oil_dict["Oil"]
                end_idx = int(self.pro_entry.get())
                oil_data = df_oil.iloc[:end_idx, 2].tolist()


                df_gas_dict = pd.read_excel(file_name, sheet_name=["Gas"])
                df_gas = df_gas_dict["Gas"]
                end_idx = int(self.pro_entry.get())
                gas_data = df_gas.iloc[:end_idx, 2].tolist()


```

```
                df_water_dict = pd.read_excel( file_name, sheet_name=["Water"])
                df_water = df_water_dict["Water"]
                end_idx = int(self.pro_entry.get())
                water_data = df_water.iloc[:end_idx, 2].tolist()


                prod_years = len(oil_data)



                Row = prod_years + int( self.well_entry.get())
                Column = int(16)


                # Initialize the NPV calculation matrix
                matrix = np.zeros([Row,Column])
                N_wells = INJ_wells + PROD_wells

                #Number of production wells and injection wells, respectively
                temp=N_wells
                Drilling_cost = int(self.drillcost_entry.get())
                Piping_cost = int(self.pipcost_entry.get())
                piping_years = int(self.pipt_entry.get())
                N_manifolds = int(self.mfnum_entry.get())
                Manifold_cost = int(self.mfcost_entry.get())
                oil_price = int(self.oil_entry.get())
                gas_price = int(self.gas_entry.get())
                water_cost = int(self.wcost_entry.get())
                OPEX = int(self.OPEX_entry.get())
                r = float(self.interest_entry.get())/100


                # Add column names
                column_names = ['Year','Number of wells', 'DRILLEX($M)','PipingEX($M)', 'ManifoldEx($M)', 'OPEX($M)', 'OilProd(SM3)', 'WaterProd(SM3)','GasProd(SM3)', 'OilRev($M)','GasRev($M)', 'WaterEx($M)', 'CAPEX($M)','Cashflow($M)','Discounted CF($M)','Cumulative CF($M)']
                matrix_df = pd.DataFrame(matrix, columns=column_names)
```

```

        for i in range (1,Row+1): #Write years in the matrix
            matrix[i-1][0]=i
            for j in range (1,5):
                if temp != 0:
                    matrix[i-1][1]=j
                    temp-=1

            matrix[i-1][2] = Drilling_cost*matrix[i-1][1]  # calculate DRILLEX
            for k in range (piping_years): #Piping ependiture in 2 years
                matrix[k][3] = Piping_cost

            matrix[0][4] = N_manifolds * Manifold_cost

            sum = matrix[i-1][2] + matrix[i-1][3] + matrix[i-1][4] #Calculate CAPEX
            matrix[i-1][12] = sum
            sum = 0



          # oil_data_flat = list(itertools.chain.from_iterable(oil_data))
            for L in range (int(self.well_entry.get())-1 ,Row):
              # print(oil_data)

                matrix[L][6] = oil_data[L-int(self.well_entry.get())] #Import production profile , production starts at year 5
               #print(oil_data)
                matrix[L][7] = water_data[L-int(self.well_entry.get())]
                matrix[L][8] = gas_data[L-int(self.well_entry.get())]
                matrix[L][5] = OPEX


            matrix[i-1][9] = matrix[i-1][6] * oil_price /(1000000)
            matrix[i-1][10] = matrix[i-1][8]* gas_price / 1000000
```

```
            matrix[i-1][11] = matrix[i-1][7]*water_cost / 1000000
            matrix[i-1][13] = matrix[i-1][9] +matrix[i-1][10]  - matrix[i-1][11] - matrix[i-1][5] - matrix[i-1][12] #Calculate the cash flow
            matrix[i-1][14] = matrix[i-1][13] / (1+r)**i #Calculate discounted cash flow

            matrix[0][15]= matrix[0][14] #Calculate cumulative discounted cash flow
            negative_cash_flow = False  # Flag variable to track negative cash flow


            if negative_cash_flow:
                # Exclude the row with negative cash flow from calculations
                matrix[i + 1][15] = 0  # Set the cumulated cash flow of the row to 0 or handle it as desired
            for i in range(Row - 1):
                if negative_cash_flow:
                    matrix[i + 1][15] = 0  # Replace all numbers with zero
                    continue

                matrix[i + 1][15] = matrix[i][15] + matrix[i + 1][14]
                prepro = int(self.well_entry.get())

                if i >= prepro and matrix[i + 1][13] < 0:
                    print("Negative cash flow encountered. Stopping the process.")
                    negative_cash_flow = True





        matrix_df.to_excel(f'E:/OLYMPUS_{l}/OLYMPUS_{l}_{s}/OLYMPUS_{l}/NPV_Calc_{l}_{s}.xlsx')
        NPV_summary(self)
```

```



    def run(self):
            self.frame1.pack()
            #self.frame2.pack()
            #self.frame3.pack()
            self.frame4.pack()
            self.frame5.pack()
            #self.frame.pack()

            # Pack the logo Label widget to make it visible
            #self.logo_label.pack()
            self.root.mainloop()

if __name__ == '__main__':
    gui = MyGUI()
    gui.run()
```

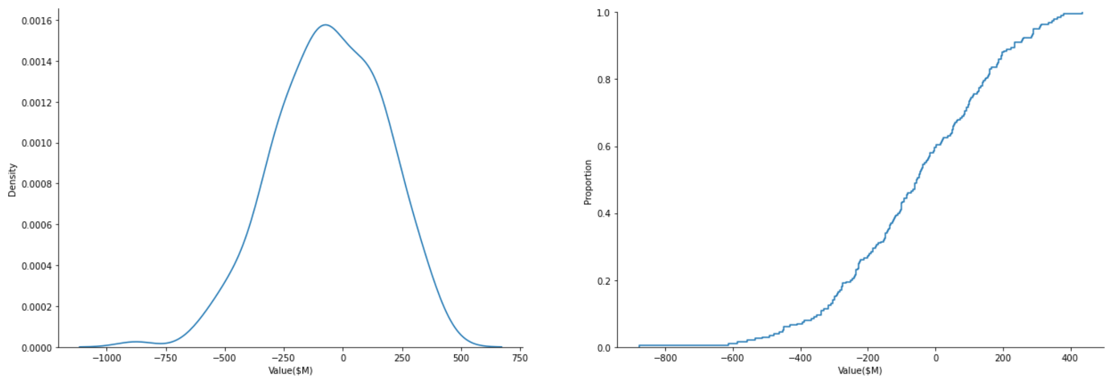
B - TABLES

5.3 Appendix B : Default input values

Input	Value
Number of years before production	5
Number of production years	25
Drilling costs for production wells	100
Piping cost	500
Piping years	3
Manifold cost ($M)	200
Number of manifolds	3
OPEX (fixed)	100
Oil price ($/sm3)	760
Gas price ($/sm3)	76
Water cost ($/sm3)	50
Interest rate (%)	5

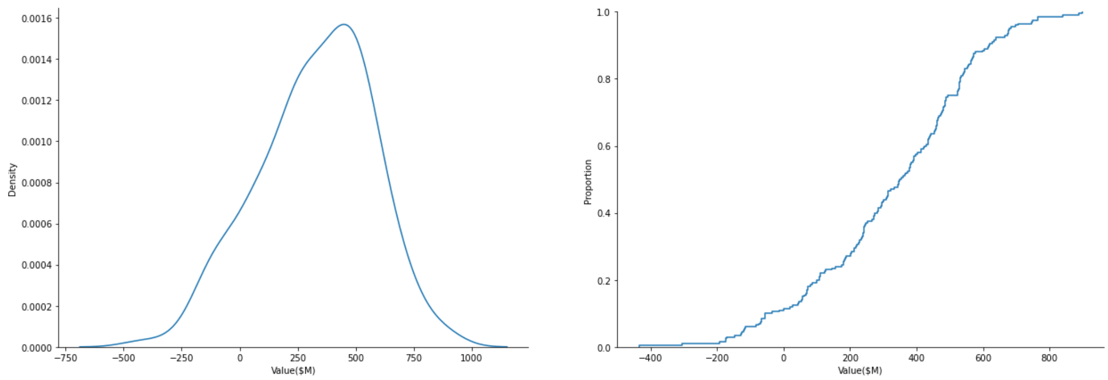
Table 5.3.1: Default input values

5.4 Appendix C : PDF and CDF curves



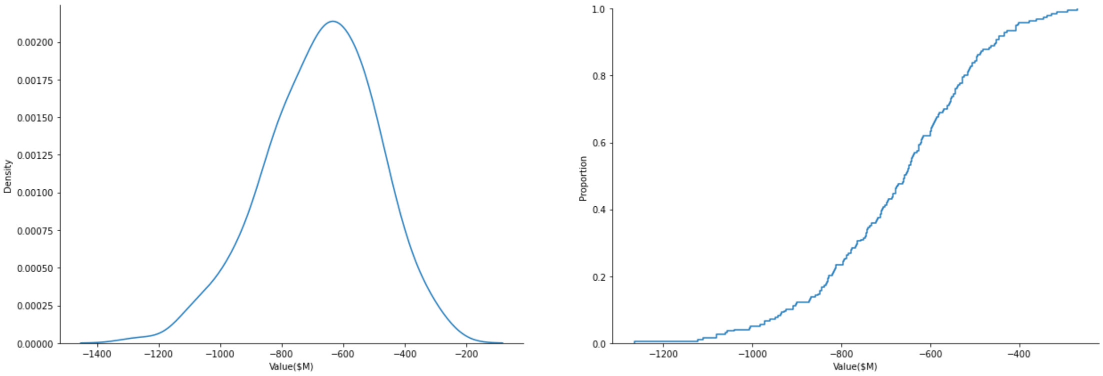
PDF CDF

Figure 5.4.1: OLYMPUS 8 PDF and CDF



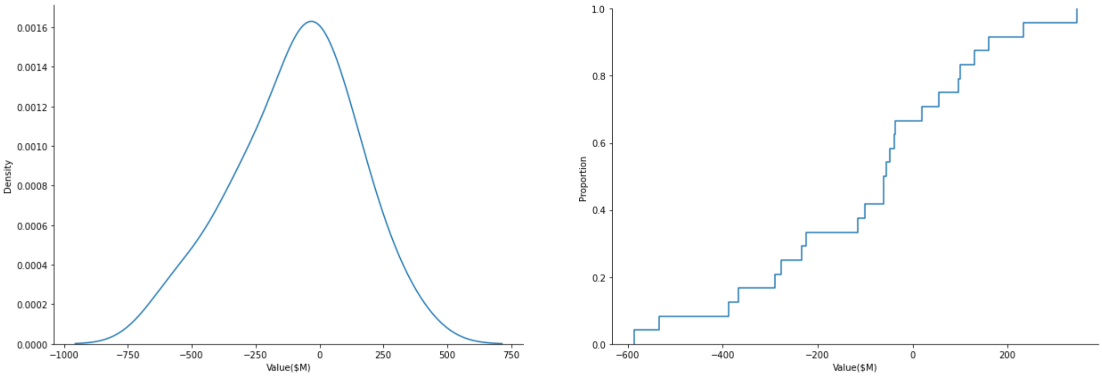
PDF CDF

Figure 5.4.2: OLYMPUS 14 PDF and CDF



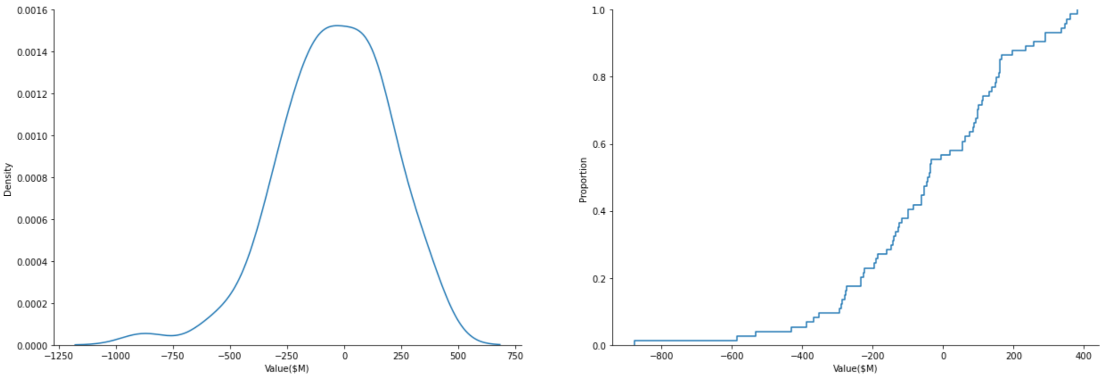
PDF CDF

Figure 5.4.3: OLYMPUS 22 PDF and CDF



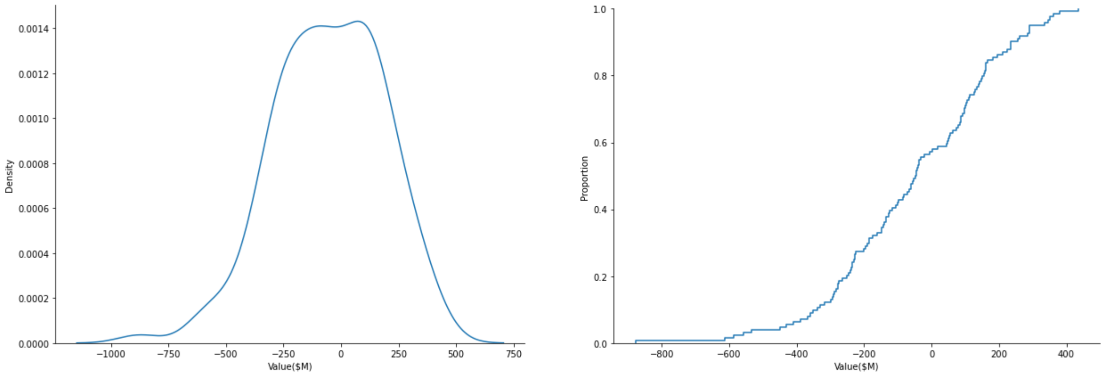
PDF CDF

Figure 5.4.4: OLYMPUS 8 - 25 Samples PDF and CDF



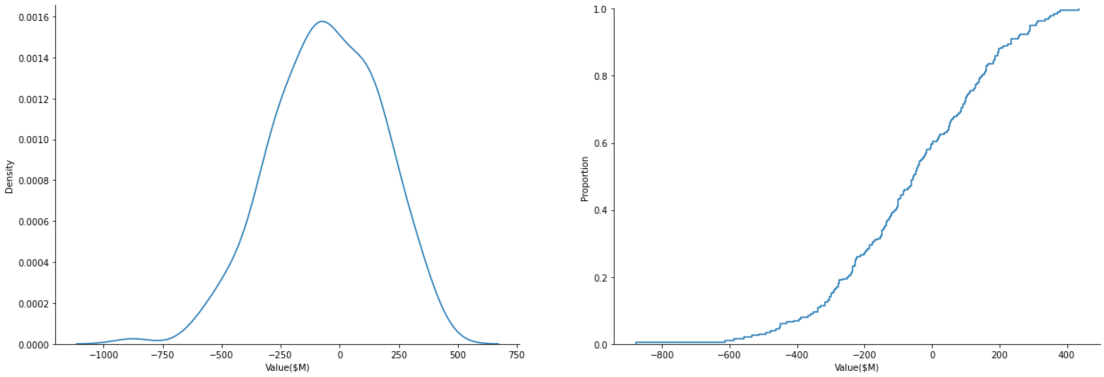
PDF CDF

Figure 5.4.5: OLYMPUS 8 - 75 Samples PDF and CDF



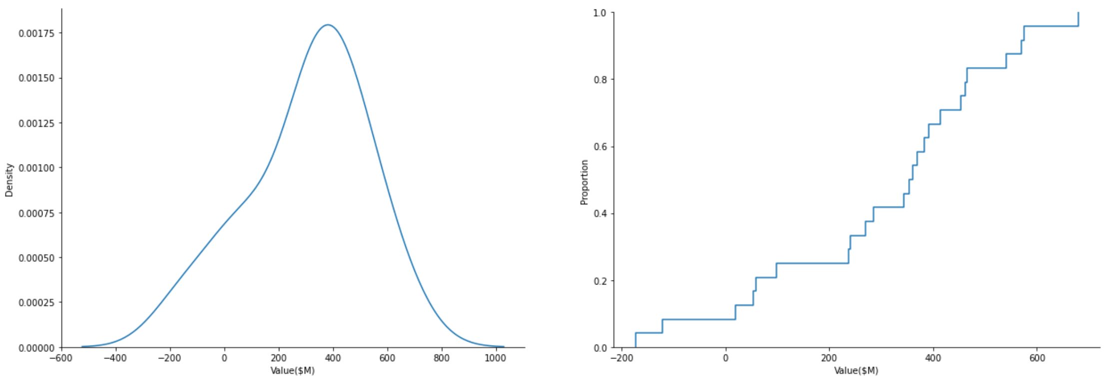
PDF CDF

Figure 5.4.6: OLYMPUS 8 - 125 Samples PDF and CDF



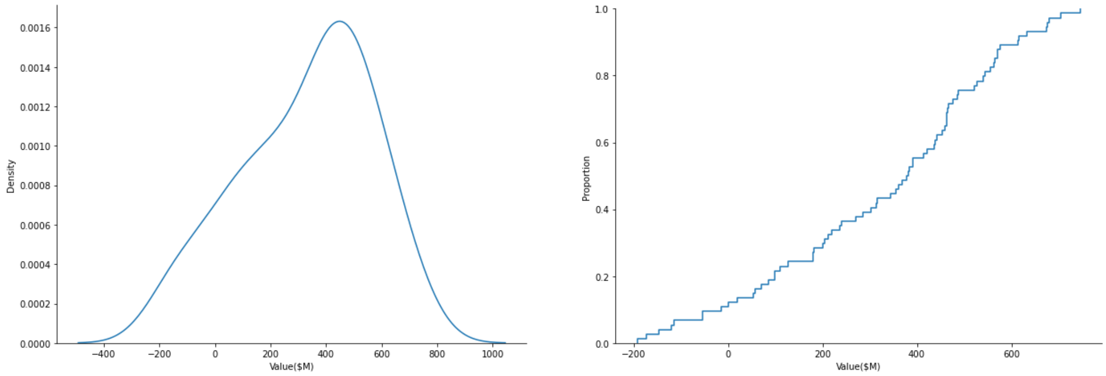
PDF CDF

Figure 5.4.7: OLYMPUS 8 - 200 Samples PDF and CDF



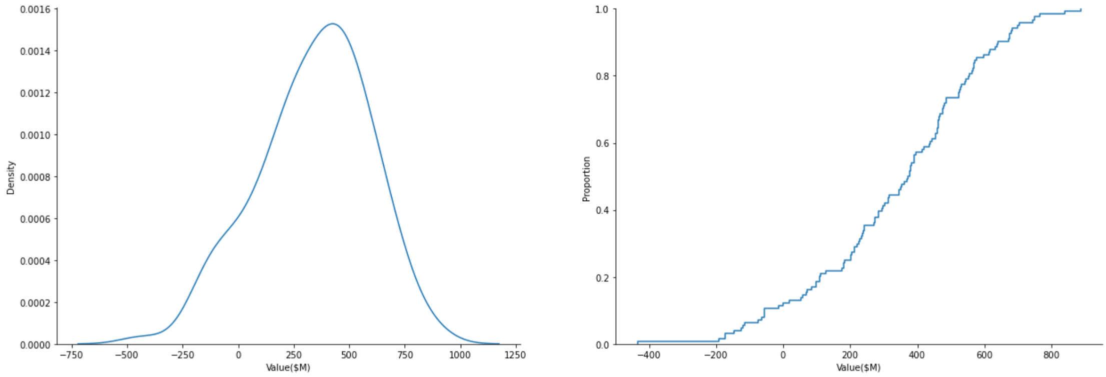
PDF CDF

Figure 5.4.8: OLYMPUS 14 - 25 Samples PDF and CDF



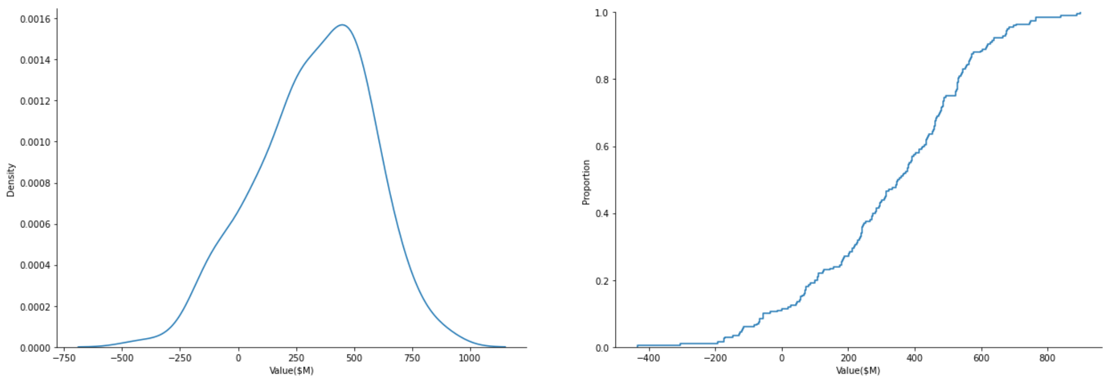
PDF CDF

Figure 5.4.9: OLYMPUS 14 - 75 Samples PDF and CDF



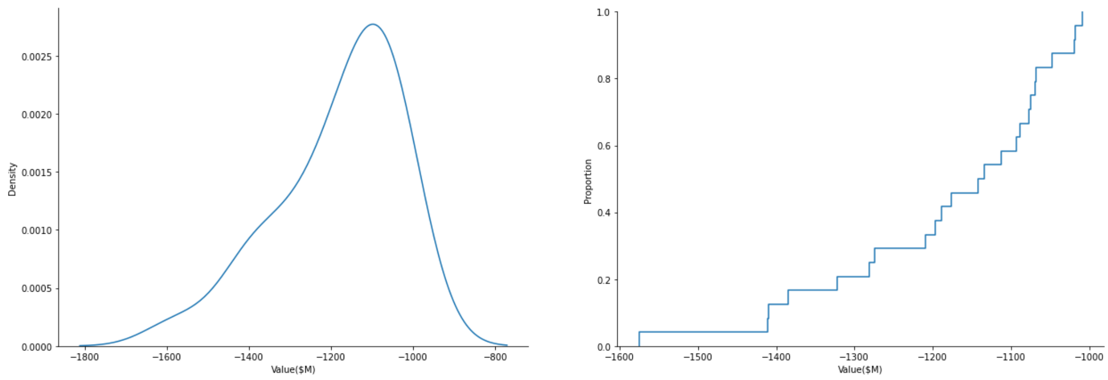
PDF CDF

Figure 5.4.10: OLYMPUS 14 - 125 Samples PDF and CDF



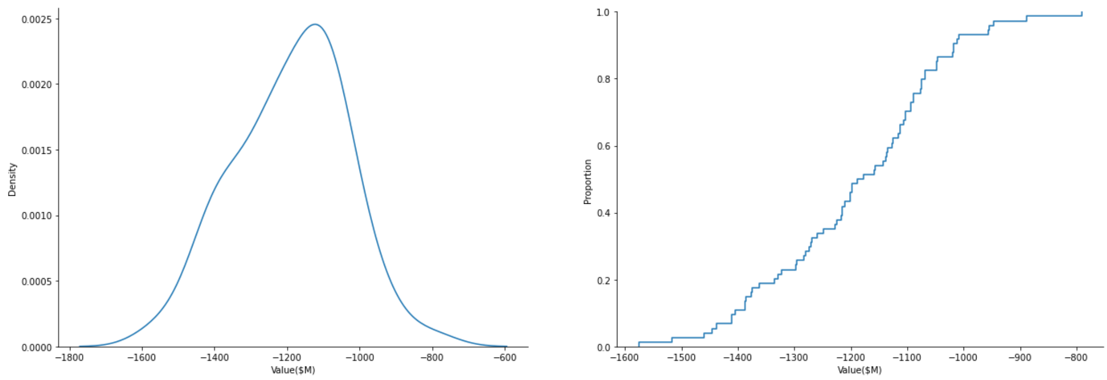
PDF CDF

Figure 5.4.11: OLYMPUS 14 - 200 Samples PDF and CDF



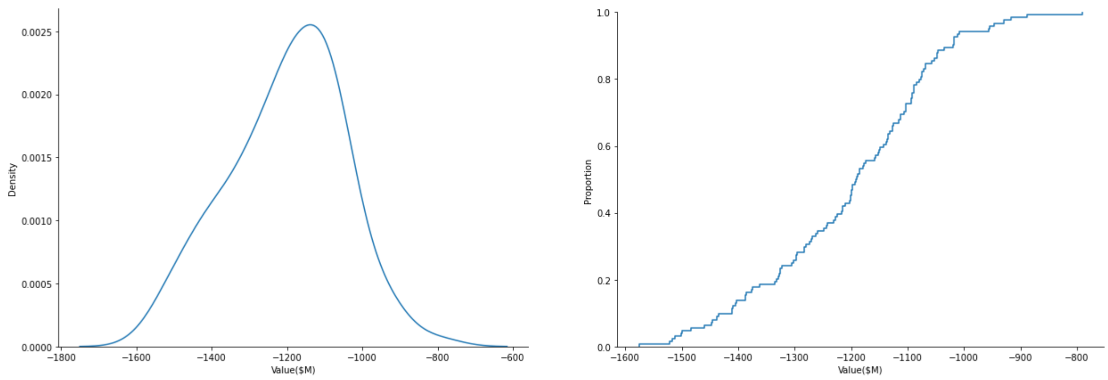
PDF CDF

Figure 5.4.12: OLYMPUS 40 - 25 Samples PDF and CDF



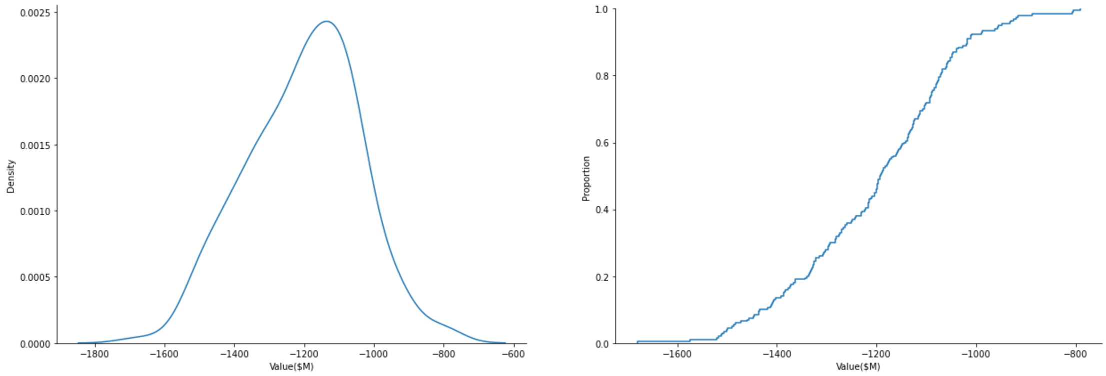
PDF CDF

Figure 5.4.13: OLYMPUS 40 - 75 Samples PDF and CDF



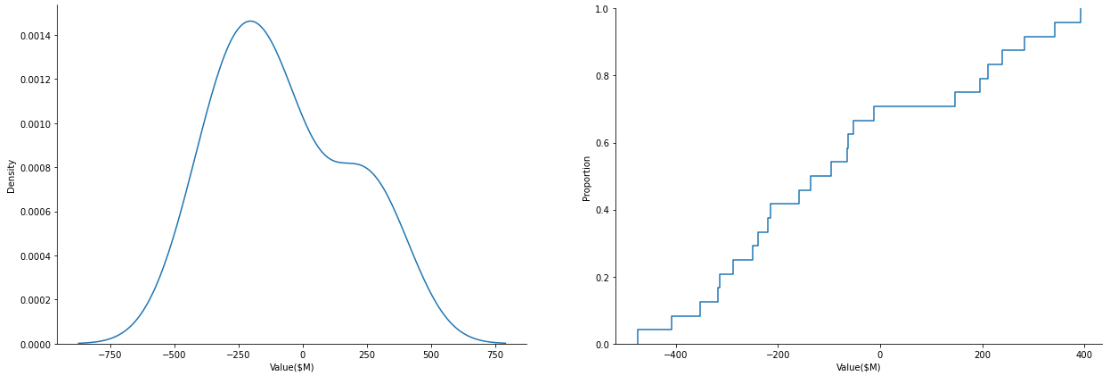
PDF CDF

Figure 5.4.14: OLYMPUS 40 - 125 Samples PDF and CDF



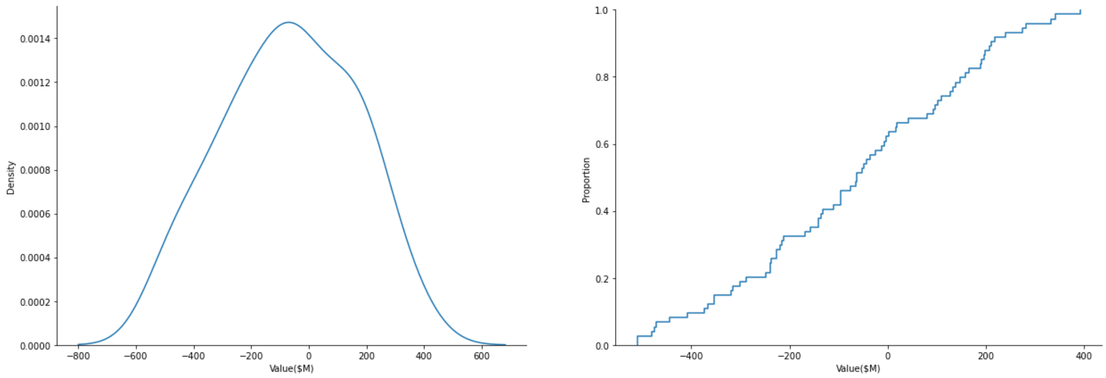
PDF CDF

Figure 5.4.15: OLYMPUS 40 - 200 Samples PDF and CDF



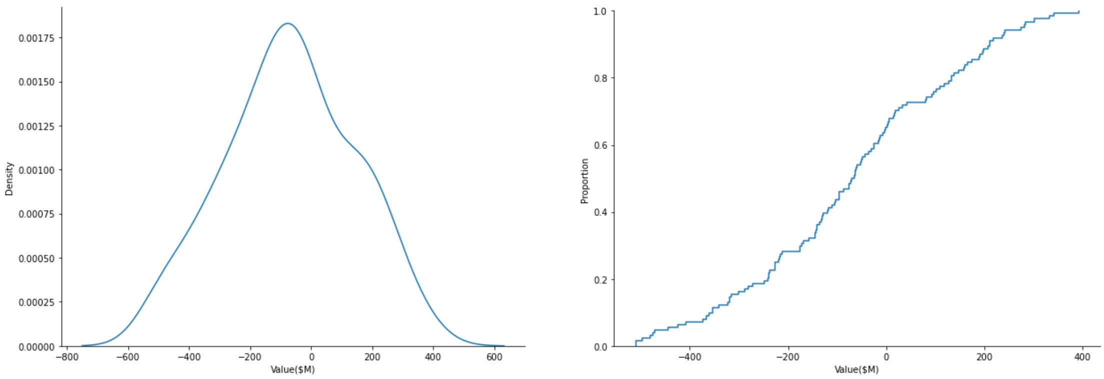
PDF CDF

Figure 5.4.16: OLYMPUS 45 - 25 Samples PDF and CDF



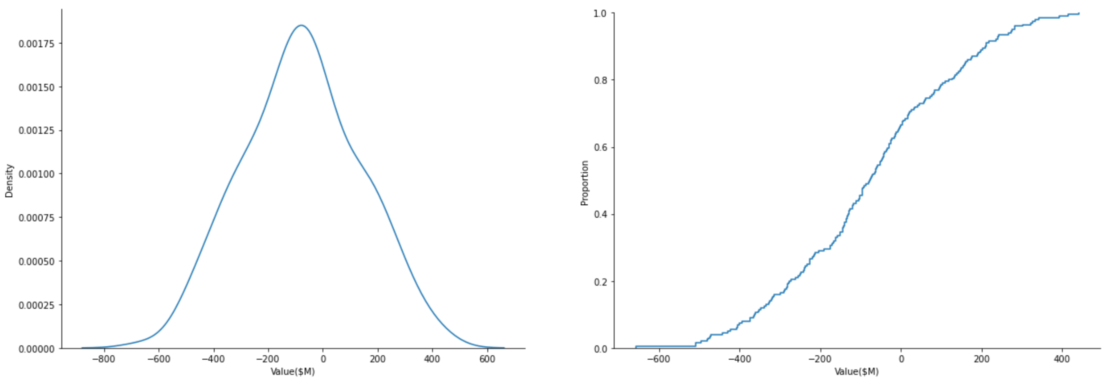
PDF CDF

Figure 5.4.17: OLYMPUS 45 - 75 Samples PDF and CDF



PDF CDF

Figure 5.4.18: OLYMPUS 45 - 125 Samples PDF and CDF



PDF CDF

Figure 5.4.19: OLYMPUS 45 - 200 Samples PDF and CDF

D - GUI

5.5 Appendix D - 1: GUI of the first script

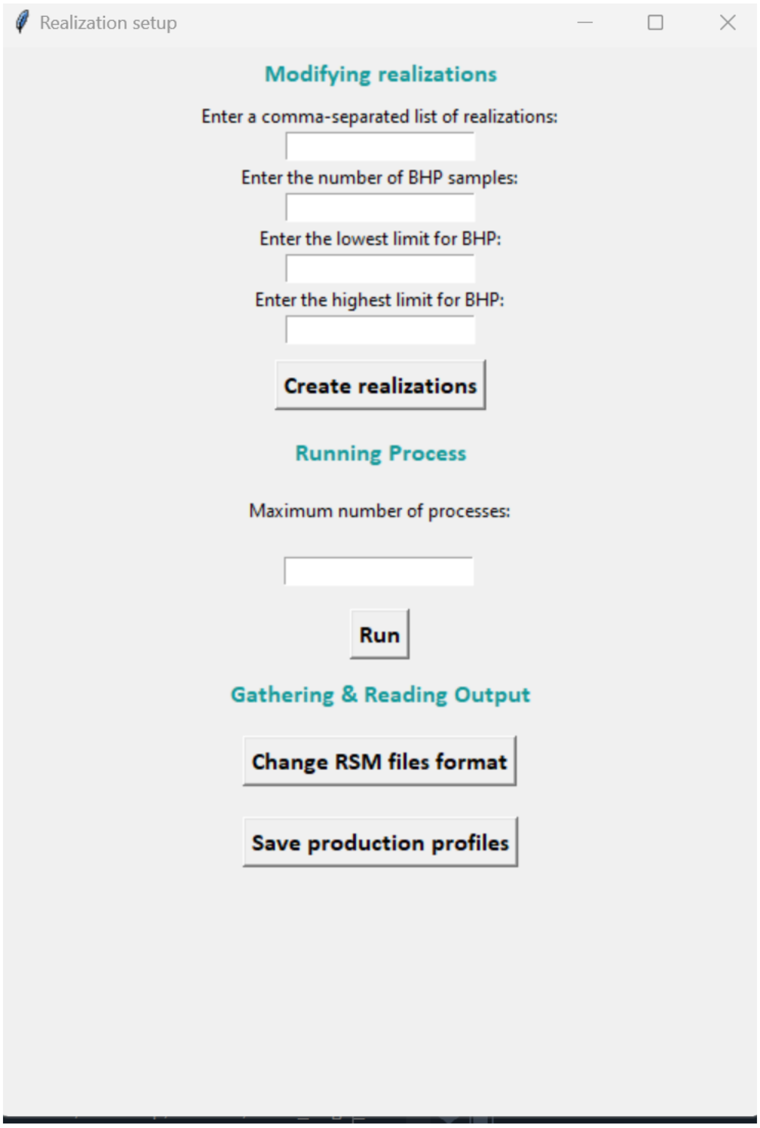


Figure 5.5.1: Graphical user interface of the first Python script

5.6 Appendix D - 2: GUI of the second script

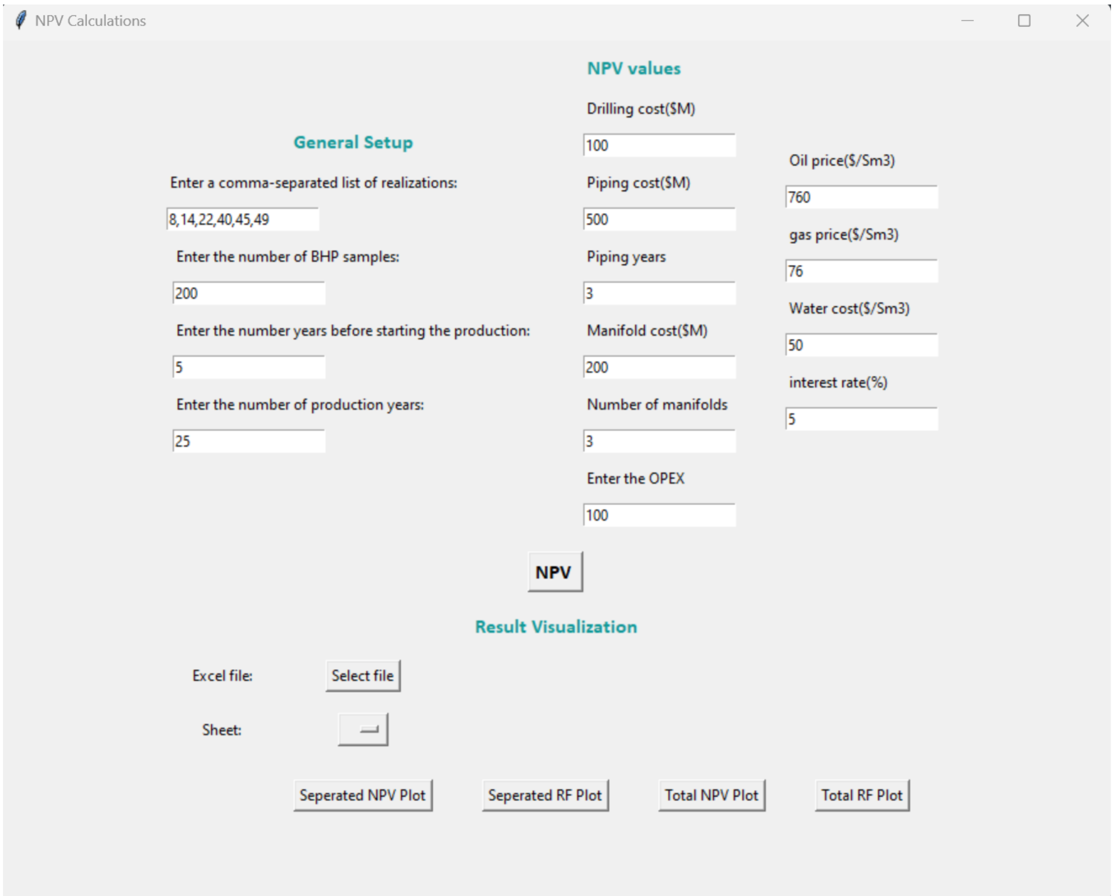


Figure 5.6.1: Graphical user interface of the second Python script

REFERENCES

[1] RB B Bratvold, SH H Begg, and Svitlana Rasheva. “A new approach to uncertainty quantification for decision making”. In: SPE Hydrocarbon Economics and Evaluation Symposium. SPE. 2010, SPE–130157.

[2] Gian Luigi Chierici and Gian Luigi Chierici. “Forecasting Well and Reservoir Performance Through the Use of Decline Curves and Identified Models”. In: Principles of Petroleum Reservoir Engineering (1995), pp. 231–253.

[3] Mike Christie, Vasily Demyanov, and Demet Erbas. “Uncertainty quantification for porous media flows”. In: Journal of Computational Physics 217.1 (2006), pp. 143–158.

[4] Ralf Schulze-Riegert et al. “Multiobjective optimization with application to model validation and uncertainty quantification”. In: SPE Middle East oil and gas show and conference. OnePetro. 2007.

[5] Jan-Erik Vinnem, Willy Røed, et al. “Offshore Risk Assessment Vol. 1”. In: Principles, Modelling and Applications of QRA Studies (2014).

[6] Ronald E Terry, J Brandon Rogers, and Benjamin Cole Craft. Applied petroleum reservoir engineering. Pearson Education, 2015.

[7] Reza Yousefzadeh et al. “Uncertainty Management in Reservoir Engineering”. In: Introduction to Geological Uncertainty Management in Reservoir Characterization and Optimization: Robust Optimization and History Matching. Springer, 2023, pp. 1–14.

[8] Peter R Rose et al. Risk analysis and management of petroleum exploration ventures. Vol. 12. American Association of Petroleum Geologists Tulsa, OK, 2001.

[9] John Fanchi. Integrated reservoir asset management: principles and best practices. Gulf Professional Publishing, 2010.

[10] James V Wertsch. “BASIL BERNSTEIN, Pedagogy, symbolic control and identity: Theory, research, critique. London (UK) & Bristol (PA): Taylor & Francis, 1996. Pp. xiv, 216. Hbč 40.00, pbč 14.95.” In: Language in Society 27.2 (1998), pp. 257–259.

[11] Y Zee Ma. “Uncertainty analysis in reservoir characterization and management: How much should we know about what we don’t know?” In: (2011).

[12] Roger Flage et al. “Concerns, challenges, and directions of development for the issue of representing uncertainty in risk assessment”. In: Risk analysis 34.7 (2014), pp. 1196–1207.

[13] Guillaume Caumon and André G Journel. “A framework to assess global uncertainty: development and case studies”. In: (2004).

[14] Athanasios Papoulis. “Ambiguity function in Fourier optics”. In: JOSA 64.6 (1974), pp. 779–788.

[15] Lester G Telser. “The Lognormal Distribution, J. Aitchison and JAC Brown, Cambridge, England: Cambridge University Press, 1957, Pp. xviii, 176.” In: American Journal of Agricultural Economics 41.1 (1959), pp. 161–162.

[16] Peter Cunningham and Steve Begg. “Using the value of information to determine optimal well order in a sequential drilling program”. In: AAPG bulletin 92.10 (2008), pp. 1393–1402.

[17] Reidar B Bratvold, J Eric Bickel, and Hans Petter Lohne. “Value of information in the oil and gas industry: past, present, and future”. In: SPE Reservoir Evaluation & Engineering 12.04 (2009), pp. 630–638.

[18] Heng Li, Pallav Sarma, and Dongxiao Zhang. “A comparative study of the probabilistic-collocation and experimental-design methods for petroleum-reservoir uncertainty quantification”. In: SPE Journal 16.02 (2011), pp. 429–439.

[19] Linah Mohamed, Mike Christie, and Vasily Demyanov. “Comparison of stochastic sampling algorithms for uncertainty quantification”. In: SPE journal 15.01 (2010), pp. 31–38.

[20] Yasin Hajizadeh. “Population-based algorithms for improved history matching and uncertainty quantification of petroleum reservoirs”. PhD thesis. Heriot-Watt University, 2011.

[21] Semyon Fedorov, Verena Hagspiel, and Thomas Lerdahl. “Real options approach for a staged field development with optional wells”. In: Journal of Petroleum Science and Engineering 205 (2021), p. 108837.

[22] Guido Van Rossum and Fred L Drake. Introduction to python 3: python documentation manual part 1. CreateSpace, 2009.

[23] Freddy H Escobar et al. “Pressure and pressure derivative analysis for linear homogeneous reservoirs without using type-curve matching”. In: Nigeria Annual International Conference and Exhibition. OnePetro. 2004.

[24] RM Fonseca et al. “Overview of the olympus field development optimization challenge”. In: ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery. Vol. 2018. 1. EAGE Publications BV. 2018, pp. 1–10.

[25] RM Fonseca, CR Geel, and O Leeuwenburgh. “Description of OLYMPUS reservoir model for optimization challenge”. In: Integrated Systems Approach to Petroleum Production. Netherlands (2017).

[26] Susana MG Santos, Ana Teresa FS Gaspar, and Denis J Schiozer. “Risk management in petroleum development projects: Technical and economic indicators to define a robust production strategy”. In: Journal of Petroleum Science and Engineering 151 (2017), pp. 116–127.

[27] Charles V Millikan and V Sidwell. "Bottom-hole pressures in oil wells". In: Transactions of the AIME 92.01 (1931), pp. 194–205.

[28] George Fishman. Monte Carlo: concepts, algorithms, and applications. Springer Science & Business Media, 2013.

[29] Jun S Liu and Jun S Liu. Monte Carlo strategies in scientific computing. Vol. 75. Springer, 2001.

[30] Ronald W Shonkwiler and Franklin Mendivil. Explorations in monte carlo methods. Springer Science & Business Media, 2009.

[31] Werner Wetekamp. "Net Present Value (NPV) as a tool supporting effective project management". In: Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems. Vol. 2. IEEE. 2011, pp. 898–900.

[32] Arnaud De Myttenaere et al. "Mean absolute percentage error for regression models". In: Neurocomputing 192 (2016), pp. 38–48.

[33] JR Etherington and JE Ritter. "The 2007 SPE/WPC/AAPG/SPEE Petroleum Resources Management System (PRMS)". In: Journal of Canadian Petroleum Technology 47.08 (2008).