




University  
of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

# MASTER'S THESIS

Study programme/specialization:	Spring semester, 2023
MSc. Petroleum Engineering	Open
Author:	
Gideon Ofosu-Budu	..... Gideon Ofosu-Budu
Programme coordinator: Øystein Arild	
Supervisors: UiS - Prof. Dan Sui Carnain AS - Abdul-Razzaq Aqrawi	
Title of master's thesis:  Automated Drill Plan Using Reinforcement Machine Learning	
Credits: 30	
Keywords: Wellbore Design, Well Planning, Drilling, Machine Learning, Q-learning.	Number of pages: 53 + supplemental material/other: 10  Stavanger, 30th August 2023

# Automated Drill Plan Using Reinforcement Machine Learning

Master Thesis Project for the degree of  
MSc in Petroleum Engineering

Stavanger, August 2023

University of Stavanger  
Faculty of Science and Technology  
Department of Energy and Petroleum Engineering



# Abstract

Well planning is considered to be one of the most demanding aspect of drilling. Well planning is carried out to formulate a program for drilling a well in the safest and most cost effective way possible. Designing the trajectory of a well, which is part of the well planning process, has always been a mathematical-based approach that requires a lot of experience from the well planner. Geology of different regions usually differ from each other, and the experience gained from one region might not be applicable in the other. This makes the design process quite difficult and time consuming. Therefore, due to time constraints, there are always a lot of uncertainties in the well design.

In this thesis we purpose a novel use of reinforcement machine learning to automatically design the well trajectory given any type of geological constraints. The use of machine learning has surged in recent times and it has been applied to other well planning modules like bit selection and optimization. A successful implementation of a machine learning model to design a well trajectory could cut down on many uncertainties and reduce the time needed in the design process.

The results of the work demonstrate that reinforcement learning is a viable and very capable approach to automating the design of a well path without having to do complex calculations, such as in traditional design methods. Further experimentation and implementation of reinforced-based learning using 3D and 4D environments, as well introducing further realistic drilling constraints, could potentially lead to a more accurate and desirable design of well path.

# Acknowledgments

First and foremost, I express my profound gratitude to God for providing me with the strength and guidance to complete this thesis.

I would also like to thank my supervisors, Professor Dan Sui and Abdul-Razzaq Aqrabi for their advice and support throughout the entire time I was writing the thesis. I am very grateful.

Special thanks to Maalidefaa Moses Tantuoyir and Jarle Haukås for offering their time and advice anytime I needed them. This project would not have been possible without you, and I appreciate everything.

And, to my family back home, thank you for all your prayers, morale support and motivation through the ups and downs. I want to say that I highly appreciate you and love you.

# List of Abbreviations

<b>2D</b>	2 Dimensional
<b>3D</b>	3 Dimensional
<b>ACO</b>	Ant Colony
<b>AHC</b>	Agglomerative Hierarchical Clustering
<b>AI</b>	Artificial Intelligence
<b>ANN</b>	Artificial Neural Network
<b>AutoML</b>	Automated Machine Learning
<b>DBSCAN</b>	Density-Based Spatial Clustering of Applications with Noise
<b>E&amp;P</b>	Exploration and Production
<b>EM</b>	Expectation-Maximization algorithm
<b>GA</b>	Genetic Algorithm
<b>KNN</b>	K Nearest Neighbors
<b>KOP</b>	Kick-Off Point
<b>MD</b>	Measured Depth
<b>MDP</b>	Markov Decision Process
<b>ML</b>	Machine Learning
<b>MOGA</b>	Multi-Objective Genetic Algorithm
<b>MRST</b>	MATLAB Reservoir Simulation Toolbox
<b>NCS</b>	Norwegian Continental Shelf
<b>NN</b>	Neural Network
<b>PSO</b>	Particle Swarm Optimization
<b>Q-learning</b>	Quality-learning
<b>SDK</b>	Software Development Kit
<b>SVM</b>	Support Vector Machine
<b>TVD</b>	True Vertical Depth
<b>UTM</b>	Universal Transverse of Mercator

# Table of Contents

<b>Abstract</b>	<b>ii</b>
<b>Acknowledgments</b>	<b>iii</b>
<b>List of Abbreviations</b>	<b>iv</b>
<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Problem Definition . . . . .	2
1.3 Outline . . . . .	3
<b>2 Literature Review</b>	<b>4</b>
2.1 Well Trajectory Design Background . . . . .	4
2.1.1 Basic Parameters of Well Path . . . . .	6
2.1.2 Types of Well Profiles . . . . .	7
2.1.3 Survey Calculation Techniques . . . . .	8
2.1.4 Traditional Methods of Designing Well Trajectory . . . . .	9
2.1.5 Computer Applications . . . . .	10
2.1.6 Well Trajectory Design using Search Methods . . . . .	12
2.2 Machine learning Background . . . . .	13
2.2.1 Supervised Learning . . . . .	15
2.2.2 Unsupervised Learning . . . . .	16

2.2.3	Semi-supervised Learning . . . . .	16
2.2.4	Reinforcement Learning . . . . .	17
2.2.5	Automated Machine Learning (AutoML) . . . . .	19
2.3	Related Work . . . . .	20
2.3.1	Proposed Solution in 2D environment . . . . .	20
2.3.2	Proposed Solutions in 3D environment . . . . .	21
<b>3</b>	<b>Platforms, Dataset &amp; Model</b>	<b>23</b>
3.1	Platforms Used . . . . .	23
3.1.1	OSDU Data Platform . . . . .	23
3.1.2	DELFI Digital Platform . . . . .	24
3.1.3	Dataiku Data Science Studio (Dataiku DSS) . . . . .	24
3.1.4	OpenZGY . . . . .	24
3.2	Dataset . . . . .	24
3.2.1	Overview of the Volve field dataset . . . . .	25
3.3	Model . . . . .	27
3.3.1	Technicalities to be Considered . . . . .	27
3.3.2	Reward Function . . . . .	29
<b>4</b>	<b>Implementation</b>	<b>31</b>
4.1	Introduction . . . . .	31
4.1.1	Start and Target Points . . . . .	31
4.1.2	Obstacles . . . . .	32
4.2	2D Approach Using Q-Learning . . . . .	33
4.2.1	Initial Proposed Model . . . . .	33
4.2.2	Flow of Implemented Model . . . . .	34
<b>5</b>	<b>Results and Discussion</b>	<b>39</b>
5.1	2D representation of VOLVE field . . . . .	39
5.2	RL 2D Grid Environment . . . . .	40
5.3	Sensitivity Analysis . . . . .	42

5.3.1	Effect of grid size location, start and end points in finding the shortest path . . . . .	42
5.3.2	The effect of reward or cost factor in finding the shortest path . . .	44
5.3.3	Effect of hyper-parameters in finding the shortest path . . . . .	45
<b>6</b>	<b>Conclusions and Future Work</b>	<b>46</b>
6.1	Conclusions . . . . .	46
6.2	Future Work . . . . .	47
6.2.1	Training data . . . . .	47
6.2.2	Adding more granularity to the obstacles . . . . .	47
6.2.3	Considering the technicalities of drilling . . . . .	48
6.2.4	Using 4D seismic . . . . .	48
	<b>References</b>	<b>53</b>
	<b>Appendices</b>	<b>54</b>
	<b>Appendix A GitHub Link to Python Code</b>	<b>56</b>
A.1	Adding Configuration . . . . .	56
A.2	Initializing Environment and Training . . . . .	56
A.3	Terminal state . . . . .	56
A.4	Search for endLoc . . . . .	56
A.5	Shortest path to endLoc . . . . .	56
A.6	Training of the reinforcement algorithm . . . . .	57
	<b>Appendix B Plots</b>	<b>58</b>
B.1	Shortest path training failures . . . . .	58



# List of Figures

1.1	<i>An illustration of the concept of this research</i>	1
2.1	<i>Well Trajectories designed to avoid constraints [6]</i>	5
2.2	<i>Figure showing multiple well trajectories designed to enter a single reservoir</i>	8
2.3	<i>Type 1 (Build and Hold or J-type) Well Profile designed in Petrel</i>	11
2.4	<i>Type 2 (Build, Hold and Drop or S-type) Well Profile designed in Petrel</i>	12
2.5	<i>Relationship between Artificial Intelligence and Machine Learning</i>	14
2.6	<i>An overview of the Supervised Learning process</i>	15
2.7	<i>Block diagram of Semi-supervised Learning</i>	16
2.8	<i>The flow to update Q-tables in Q-Learning</i>	19
3.1	<i>Location of Volve field in the North Sea</i>	25
4.1	<i>Illustration of the effects of drilling through risky zone [68]</i>	32
4.2	<i>Flow of the main 2D program</i>	35
4.3	<i>Flowchart of a single training episode of the 2D model</i>	36
5.1	<i>Display of a Section of Seismic cube from Volve Field</i>	39
5.2	<i>2D grid representation of a RL environment with start and end location</i>	40
5.3	<i>Shortest well path in a 2D grid environment</i>	41
5.4	<i>Upscaling well path</i>	42
5.5	<i>Short location distance. End location is in-front of an obstacle</i>	43
5.6	<i>Short location distance between start and need location. End location is not behind an obstacle</i>	43
5.7	<i>Increased location distance in 2D grid</i>	44
5.8	<i>Failed training episode</i>	45

B.1 <i>Plot 1</i> . . . . .	58
B.2 <i>Plot 2</i> . . . . .	59
B.3 <i>Plot 3</i> . . . . .	59
B.4 <i>Plot 4</i> . . . . .	60
B.5 <i>Plot 4</i> . . . . .	60
B.6 <i>Plot 5</i> . . . . .	61
B.7 <i>Plot 6</i> . . . . .	61
B.8 <i>Plot 7</i> . . . . .	62
B.9 <i>Plot 8</i> . . . . .	62
B.10 <i>Plot 9</i> . . . . .	63
B.11 <i>Plot 10</i> . . . . .	63

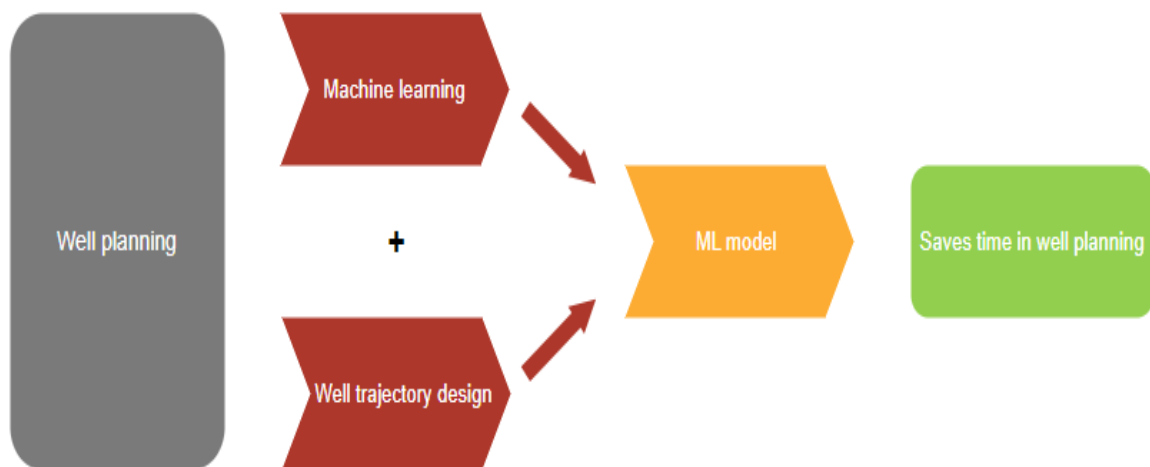
# List of Tables

3.1	Volve Dataset files . . . . .	26
3.2	Sample of a Well Trajectory Spreadsheet for F-1 A well from Volve dataset taken from Petrel . . . . .	30

# Chapter 1

## Introduction

Artificial Intelligence(AI), and its application in the energy industry, has seen a drastic increase and a great deal of success in the past decade. For example, operations within oil and gas recovery, such as from drilling to production to transportation, have been automated these days and employ AI in one form or the other [1][2]. Well trajectory design during well planning is a potential area to optimize by applying machine learning techniques, but is yet to be explored in much detail. This thesis investigates the feasibility and applicability of reinforcement learning (RL) as a potential use case in reducing well planning time and effort in the energy industry.



**Figure 1.1:** An illustration of the concept of this research

As shown in *figure 1.1* above, the concept of this thesis is to combine machine learning, specifically reinforcement learning, with already existing ideas and techniques in well planning to create a machine learning model. The resulting ML model is expected to save time in well

planning by operating automatically and finding the shortest drill path, while maintaining a high level of quality.

## 1.1 Motivation

Drilling operations are performed to create a channel to connect a point on the earth's surface to a target depth of the geological formation containing "*sweet spots*" of hydrocarbons [3]. The objective is to create a path that allows fluid transfer, that is, either reservoir fluids to the surface or fluids from the surface into the well. Having the best possible well path for drilling, is very vital to the overall future integrity of the well and great effort must be put into this stage of well planning.

Accidents during drilling do happen, ranging from minor to catastrophic ones. Designing the trajectory through a shallow gas pocket in the formation and drilling through it, for instance, can lead to a kick or blowout in extreme cases. Also, designing very tight angles of deviation can lead to casings or tubing getting stuck and buckled in the drilled hole. This can result in a lot of extra costs and requires a lot of time to be remedied, which is highly undesired. Essentially, these accidents are to be avoided and proper care needs to be taken in the path planning phase; the genesis of everything.

## 1.2 Problem Definition

A review of well design literature published over the past few decades reveals that the design of well trajectories heavily relies on the designer's expertise and judgment. And, the outcomes are not always the best due to the absence of very strict mathematical models and pertinent optimization theories. The mathematical models that exist often ignore the geological features of the formation and other key variables of drilling. This issue is also the same with computer programs made for trajectory design. Possible questions that arise include;

- Can the drill plan be designed to consider all the constraints in each formation?
- Can the design include drilling technicalities such dogleg severity, inclination, azimuth?

- Is it possible for the resulting method to be automated, such that engineers can spend more time focusing on other critical tasks?
- Will the resulting method be more efficient and more cost effective compared to existing solutions?

The aim of this study is to find a solution that is not only equally or more accurate than traditional methods, but also requires less time, which can free up an engineer to concentrate on other important tasks. To achieve this, a 2D reinforcement learning model is attempted in this work to study its applicability and viability in the automation of the well path process in drill planning.

## 1.3 Outline

This section gives a general overview of the thesis structure. A short description of what is to be expected in the subsequent chapters is presented.

*Chapter 1* introduces the problem that this thesis looks to solve and the main motivation for doing this project.

*Chapter 2* gives a review of the literature. Some traditional approaches as well as computer-based solutions are discussed. A general overview of machine learning is also examined.

*Chapter 3* has three sections. The first section describes the various software platforms used in this thesis. The second section discusses the dataset used. The last section go into detail about how the different components of the model compare to real-world scenarios.

*Chapter 4* discusses the methodology in detail. It describes step-by-step all the processes involved in implementing the model. Both the failures and the successes will be reviewed in this chapter. Finally the main 2D model implementation is described.

*Chapter 5* presents the results and analysis from the experiments presented in *chapter 4*.

*Chapter 6* concludes the thesis, examines the lessons learned, and makes suggestions for future work.

# Chapter 2

## Literature Review

Development of subsurface drilling processes and programs is critical and cost intensive in upstream operations [4]. A poorly designed well trajectory could result in major accidents such as stuck pipes and wellbore collapse, so the optimization of the well path is essential in reducing drilling risk. Proper well spacing in a multi-well region also adds to finding the ideal development strategy for the field [5]. Well planning consists of many modules which include, well path design, mud design, casing/tubing design, bit selection etc.

Well path design is very crucial, and it involves designing the best possible trajectory that connects the surface to a predefined target in the reservoir. The trajectory must be designed such that it avoids all constraints that will make the drilling operation a challenge. Having constraints such as hard formations, salt formations, faults, shallow gas pockets etc. in the well path introduces drilling risk and ultimately, the drilling operation turns costly and an increase in non-productive days. As a result, any opportunity to maintain a reasonable length of the wellbore, while staying in the allowable curvatures and avoiding constraints, usually speeds up the drilling process and lowers overall drilling costs [6]. Examples of trajectories designed to avoid some constraints are shown in *figure 2.1*.

### 2.1 Well Trajectory Design Background

Up until the late 1920s, only vertical wells were primarily planned by drillers [7]. Since then, directional wells have been successfully designed and drilled in various oilfields. A vertical

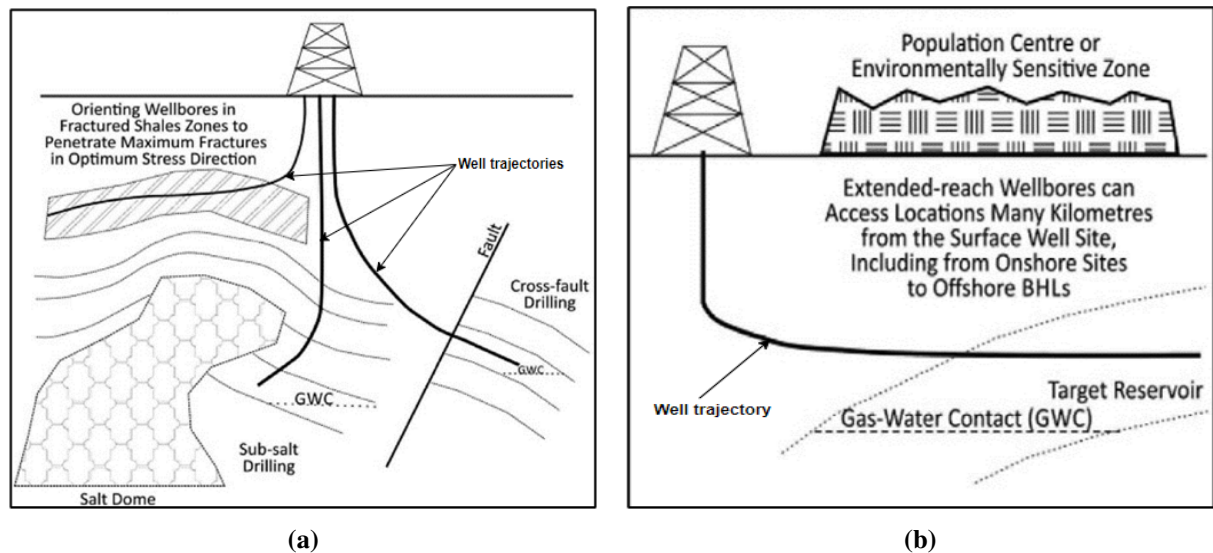


Figure 2.1: Well Trajectories designed to avoid constraints [6]

wellbore is a “*straight*” hole or a well with only a small angle of deviation. Drilling high-angle-deviation wells has been made possible because of advances in technology. Notably, the Rotary Steerable System (RSS) [8], offered by several companies. The application of directional drilling is expanded into three major domains as follows:

- **Environmental constraints:** These include natural land obstructions such as mountains, lakes swamps, rivers, farmlands, or critical buildings.
- **Underground geological constraints:** Directional drilling can be a means to access oil deposits for formations with geological obstructions such as faults, unconsolidated formations that could collapse easily, salt domes etc by avoiding these obstructions. Directional wells like sidetracking wells, multilateral wells, extended-reached wells have considerably widened the potential of accessing hydrocarbons from previously inaccessible depths.
- **Treatment for downhole drilling accidents:** Directional wells can also be drilled when there is an accident downhole. For example, if there is a stuck pipe or a “*fish*”, a directional well can be drilled from a section above.

To design the trajectory the surface co-ordinate, target co-ordinate, TVD, KOP and build-up rate are required. For designing good trajectories, these properties and their correlations are crucial.



### 2.1.1 Basic Parameters of Well Path

Well path refers to the actual shape of the borehole axis when the drilling process is finished. There are some significant parameters of the well path that determine its resulting shape after drilling. Basic concept of well path parameters and the relationship between each parameter is very essential for the trajectory design, path measurements and calculations as well as well path control. Some of these basic parameters are discussed below.

- ◆ **Azimuth:** The azimuth of a wellbore at any point is defined as the projection of the borehole direction onto a horizontal plane. It is measured clockwise from the true north. The unit for azimuth is degrees ( $^{\circ}$ ).
- ◆ **Dip:** Dip is the magnitude of the inclination of a plane from horizontal.
- ◆ **Dogleg severity:** A measure of the curvature of a borehole. It is usually measured in degrees per 30m or 100ft.
- ◆ **Dogleg:** A dogleg is a particularly bent place in a wellbore where the trajectory changes rapidly.
- ◆ **Displacement:** The horizontal distance between the starting point at the surface and the target co-ordinate in the subsurface.
- ◆ **Build-up rate:** Defined as the rate at which angle is built up from the kick-off point. Unit of measurement is ( $^{\circ}/m$ ). High change of angle per length results in severe doglegs in the trajectory. This creates sharp bends and makes drilling difficult.
- ◆ **Displacement:** The horizontal distance between the starting point at the surface and the target co-ordinate in the subsurface.
- ◆ **Inclination:** Angle formed between the tangent of borehole axis and the gravity line. The inclination shows the severity of the wellbore deflection. The inclination has the unit of degree ( $^{\circ}$ ).
- ◆ **Kick-off point (KOP):** The point at which the well starts deviating from the vertical. It is always vital to kick-off in a soft formation.

- ◆ **Measured depth (MD):** Distance from top of well to the target along the well path.
- ◆ **True-vertical depth (TVD):** Vertical distance between the start point at the surface and the target point.

### 2.1.2 Types of Well Profiles

The well path may follow a few different routes. For a vertical well, the profile is just a straight line with little to no deviation from the original starting point. For directional wells, there are three possible types of well profile. They are:

#### **Type 1: Build and Hold or J-type**

This is the most common and simplest design for a directional well. The well is vertical until the KOP, where the well deviates at a predetermined angle. When the desired inclination is reached, it is maintained over the tangential section to intersect the target. This type of profile is often applied when a large horizontal displacement is required at relatively shallow target depths.

#### **Type 2: Build, Hold and Drop or S-type**

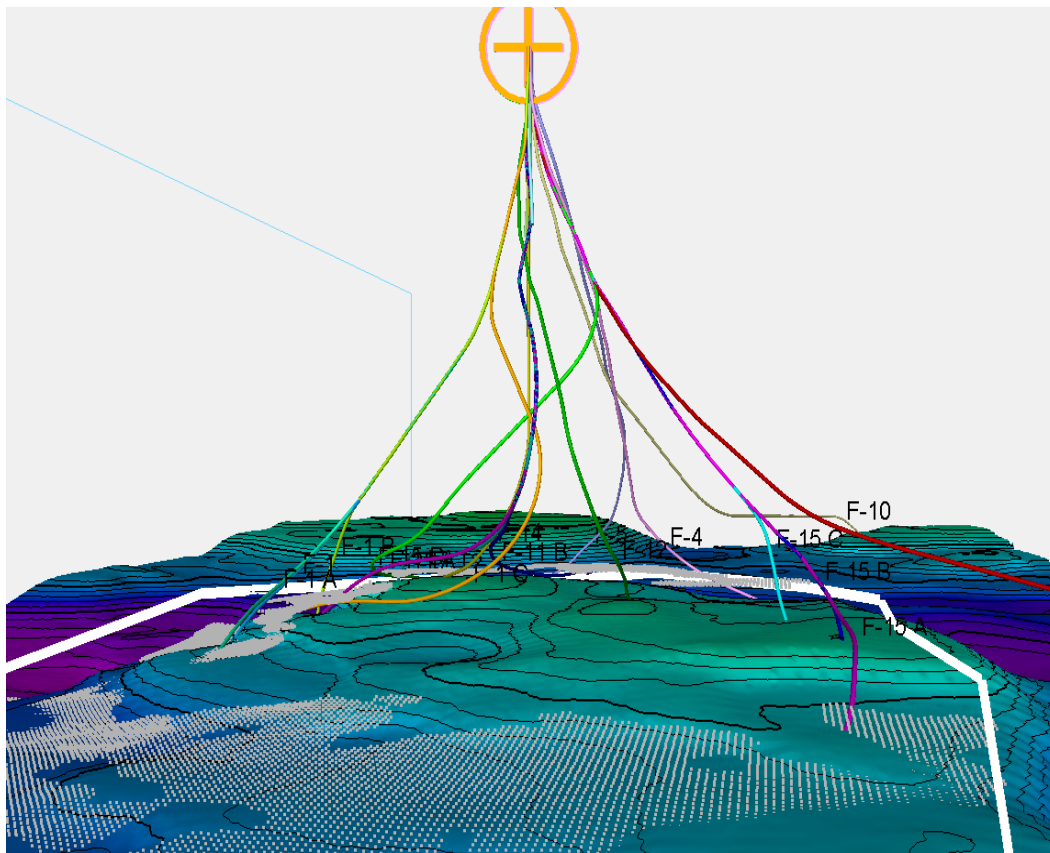
This profile type has a similar shape to Type 1 down to the lower part of the build-up. Here the inclination is reduced, and the tangent section is dropped-off. In some cases, the drop-off section becomes vertical as it reaches the target. This type of profile is used for deep targets with small horizontal displacement.

#### **Type 3: Deep Kick-off and Build**

The profile has a long vertical with a deep KOP and then inclination built quickly to the target. They are similar to *Type 1* above except the KOP is at a deeper depth. Well under this category are mostly used as appraisal wells to assess newly discovered reservoirs. The profile is used when there are obstructions such as salt dome or in sidetracking.

## Horizontal Wells

Wells with long horizontal endings in the reservoir have been designed and drilled in the industry today. Horizontal wells are high-angle wells in which the inclination is generally greater than  $85^\circ$ . A long section of the wellbore is placed within the reservoir. This allows for improved production from certain reservoirs that would otherwise be uneconomic. It requires more than one build-up section to achieve the inclination. Drilling horizontal wells has lots of challenges and so drilling costs are extremely high.



**Figure 2.2:** Figure showing multiple well trajectories designed to enter a single reservoir

### 2.1.3 Survey Calculation Techniques

At every point in time during drilling of a wellbore, engineers always want to have an idea of how accurate the wellbore being drilled is, as compared to the trajectory designed prior to drilling. Several calculation methods have been developed to estimate the wellbore path as the wellbore is being drilled [7] [9]. Some of the widely used methods are discussed below.

**The radius-of-curvature method** has the shape of a spherical arc and uses sets of angles measured at each end of the course length to generate a space curve representing the wellbore path. This approach is perhaps the most accurate of them all but is difficult to do manually.

**The angle-averaging method** calculates the wellbore tangentially using the simple averages of the angles at the top and bottom over the course length. This method is simple and accurate.

**The tangential method** uses the inclination and direction angles measured at the lower end of the course length and the path is assumed to be tangent to these angles throughout the section length.

#### 2.1.4 Traditional Methods of Designing Well Trajectory

Designing the well path, as mentioned earlier, has always been a vital part of the drilling process. Before the introduction of technology into the industry, most designs were done mathematically by hand calculations. These methods give satisfactory results that are still used in the oil and gas industry today. For instance, *Sampaio [10]* discusses using Bezier curves in producing well trajectories with continuous and smooth change of curvature. The technique can be used to find the trajectories for any of the four common end conditions in terms of azimuth and inclination, namely: free-end, set-end, set-inclination/free azimuth, and free-inclination/set-azimuth. *Eren and Suicmez [11]* also proposed a novel method of calculation, "Improved Tangential Method." Findings of the proposed method are compared with other existing methods using two different actual trajectories of an oilfield in addition to the three well geometries (J-type, S-type, and Horizontal well profile). Results show that it calculates directional trajectories to produce correct results regardless of the type of well trajectory.

In another study by *Sampaio [12]*, a mathematical method using the cubic function to design complex trajectories for 3D wells is presented. There is a degree of freedom in this design which allows parameters to be adjusted to fit the design requirements and to perfect the trajectory. The resulting trajectories are smooth continuous functions according to the study, and several numerical examples in the paper illustrate the various end conditions.

According to *Liu et al. [13]*, adopting the 3D Dubin's curve, which has been widely used

in autopilot for path planning but yet to be employed in drilling industry, has the potential to effectively optimize the path design. The 3D trajectory design is based on the 3D Dubin's curve and is particularly useful in subsea wells where multiple wells are drilled from one drilling site.

Some of these traditional well path design methods strongly depend on the designer's experience and judgment and results are not necessarily the best. Traditional methods used in well planning have been in existence for decades though, and have been extremely useful in that time. In recent times, they are found to be very time consuming and lacking in some respects considering the advancement and use of technology in the oil and gas industry.

### 2.1.5 Computer Applications

In most cases, several different trajectories may have to be investigated to find the best trajectory among them that avoids all constraints that can cause drilling problems. Each choice of kick-off points or build up rate means a different trajectory to be calculated. Such repetitive calculations are best done by computers. The use of computer software, makes planning easier, especially when dealing with substantial amounts of data. Several operating companies have their own computer programs used for well planning and trajectory design. Example of such computer programs include SLB's Petrel™ E&P Software, Oliasoft WellDesign, WellAssist by Oliasoft AS and FutureOn, WellArchitect and Integrated Well Designer by Dynamic Graphics, Inc., COMPASS by Halliburton etc.

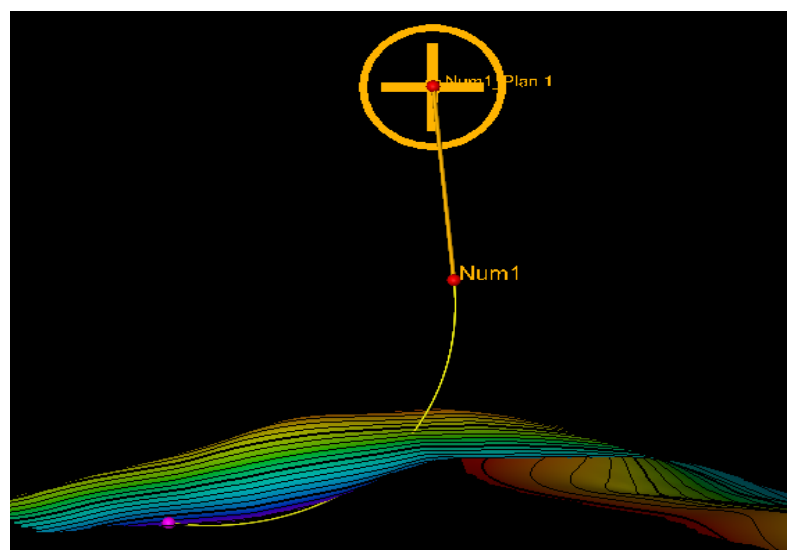
- Oliasoft WellDesign: It has a trajectory design feature that allows planning of different versions of a trajectory, with full revision tracking and bulk management of wells [14].
- WellAssist: A software as a service (SaaS) tool which is an integration of Oliasoft's Well Design with FutureOn's FieldTwin. This tool helps in conceptualizing well placement and field layouts before well development [15].
- WellArchitect and Integrated Well Designer: Both allow visualization of geologic and reservoir models, as well as 2D surfaces into the display while planning, including evaluating hazard-potential such as intersection with already planned paths and faults [16].

- COMPASS: Designed for both oil companies and directional contractors, COMPASS is a directional path planning software for well planning, survey data management, as well as anti-collision analysis [17].

### Petrel™ E&P Software Platform

Petrel E&P Software Platform by SLB is a well-known oil and gas industry platform made to bring the different disciplines of an oil and gas industry together with best-in-class science in an unparalleled productivity environment. This makes it possible for experts from different fields to work together and share ideas from exploration to production [18]. The Petrel platform can be used both on-premises and in the Delfi cognitive E&P environment.

Petrel platform has a trajectory planning module that efficiently delivers a drillable trajectory with constraints and sensitivity analysis. It allows effective design of the best trajectory, run of anti-collision analysis, analysis of drilling targets, and leverages 3D visualization tools. Teams may work more productively together to plan wells, evaluate and build assets because of the unified visualization, interpretation, and modeling workspace provided by this. In *figures 2.3 and 2.4* are samples of well trajectories designed with Petrel software.



**Figure 2.3:** Type 1 (Build and Hold or J-type) Well Profile designed in Petrel

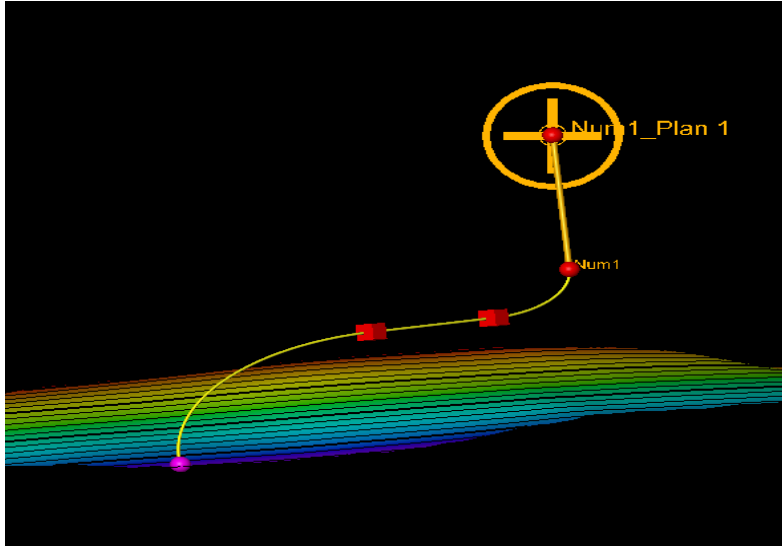


Figure 2.4: Type 2 (Build, Hold and Drop or S-type) Well Profile designed in Petrel

### 2.1.6 Well Trajectory Design using Search Methods

Several search methods or algorithms have also been applied in the attempt to optimize the designing process of well paths. Search methods help in solving various problems in AI [19]. Examples of application of search methods in algorithms include the different variations of the genetic algorithm, models based on fuzzy sets, swarm intelligence, etc. Using search algorithms proves to be an improvement and a better option.

In this study for example, PSO, which is a metaheuristic optimization method that makes few assumptions about a problem being optimized, is proposed by *Atashnezhad et al.* [6] to solve the trajectory design problem. Particles in swarms move around in n-dimensional space of the workable solutions until finding the best global position in every iteration. When an improved position is discovered, it replaces the earlier one until the best position is found. The result of the study proved that a tuned PSO algorithm proved to be a powerful and rapid optimization method for solving the non-linear deviated wellbore trajectory problem.

Genetic algorithm is researched as a possible optimization method for a 3D design [20]. The paper introduces a software package that uses a genetic algorithm to find the optimum drilling depth of directional and horizontal wells in 3D. A special penalty function, mutation, crossover probabilities, and stopping criterion were used to obtain the global minimum drilling depth. This minimum was achieved at the minimum values for kickoff point, inclination angle, build-up and drop-off rates, which were set as constraints. The minimum values of these

parameters reduce the dogleg severity, which in turn reduces the drilling operation problems. The GA's performance was superior compared to conventional well design methods and a computer program called WELLDES in the same case study, and the overall length of the output trajectory designed was much reduced.

*Mansouri et al. [21]* also developed a multi-objective genetic algorithm (MOGA) to design a 3D well path in an attempt to solve the well trajectory design optimization problem. The radius-of-curvature survey method discussed in *section 2.1.3* is employed in this design. The idea of the multi-objective genetic algorithm methodology, in this case, is to develop and apply it to two objective functions, namely wellbore length and torque, subject to multiple constraints. By minimizing both wellbore length and torque it is likely that a wellbore designed to reach a specific target can be drilled more quickly and cheaply than other potential trajectories. Upon the findings of this research, it was concluded that although the two (2) objective functions have no explicit relationship, the MOGA has rapid convergence and is high performing compared to previously discussed single objective function studies.

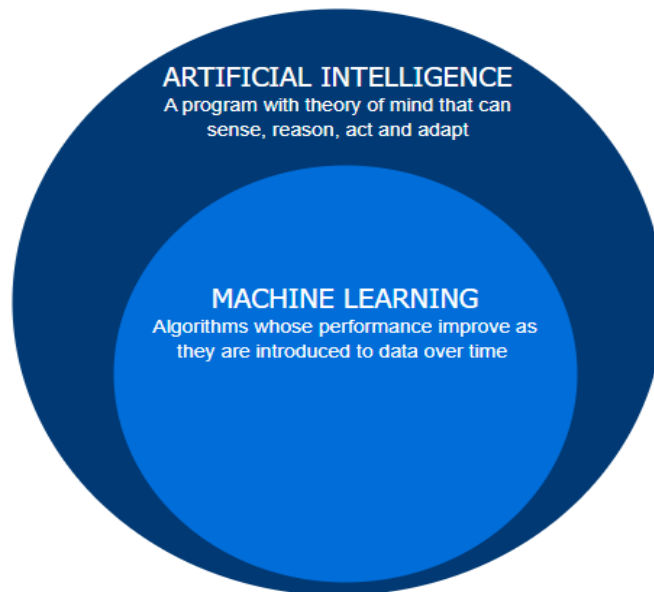
In another comparative research [22], the performances of different algorithms are evaluated. Genetic, ant colony, artificial bee colony and harmony search algorithms are evaluated to seek the best performance among them, and the primary aim of the study is to derive less time-consuming algorithms that can be deployed to solve a range of complex well-path design challenges. Each of the metaheuristic algorithms applied is tuned for the wellbore path optimization. It was observed that all the algorithms evaluated except ACO are very fast solving these complex cases. The minimum total measured depth located by all studied algorithms is similar (except for ACO), but the computation times and number of iterations taken by each one is different.

## 2.2 Machine learning Background

In the past few decades, the concept of using ML models to develop AI to solve challenging issues in the various industrial sectors and in academia has skyrocketed, and for a good reason as well. As time has gone, it has become clear that the "*fourth industrial revolution*" is the digital transformation, which is characterized by the convergence of technologies like robotics,



artificial intelligence, and autonomous vehicles that blur the lines between the physical, digital, and biological realms [23].



**Figure 2.5:** *Relationship between Artificial Intelligence and Machine Learning*

Machine learning, which is a subset of AI, is defined as a computing paradigm in which the ability to solve the presented problem is developed by referring to similar prior cases called training examples. It reasons the problem by “learning” from the training examples. There are several tasks to which we can apply ML methods to. These include:

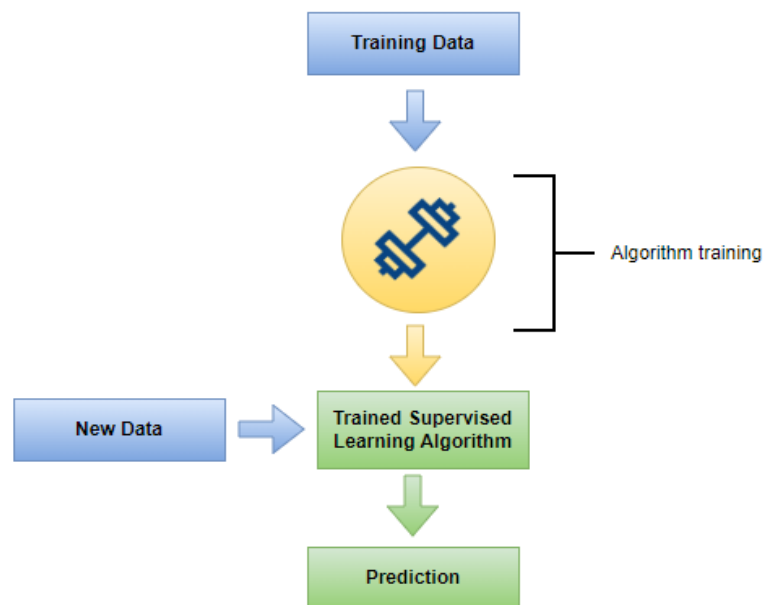
- Classification; a process of defining items into one or more among the predefined categories.
- Regression: a process of estimating a continuous value or values based on the current input factors.
- Clustering: a process of segmenting an entire group into subgroups based on similarities among items.
- The hybrid task: a mixture of any two of the above-mentioned tasks.

*Jo [24]* explores the four types of ML methods listed as the supervised learning, the unsupervised learning, the semi-supervised learning, and the reinforcement learning.

### 2.2.1 Supervised Learning

Supervised learning is the type of learning where all the training examples are labelled. The primary aim is to create an estimator that can predict an object's label using a set of features [25]. In this learning paradigm, the parameters undergo iteration and are perfected for minimizing the error difference between the target output and the computed output. Supervised learning is usually applied to the data classifications and the regressions tasks.

Examples of supervised learning algorithms include the different variants of the KNN algorithm, Bayesian Learning, SVM, Decision Tree algorithms like Simple Linear Regression and Random Forest, etc. A number of these algorithms have been applied in solving world issues. For example, in population genetics [26], and in the health sector for research in autism spectrum disorder [27].



**Figure 2.6:** An overview of the Supervised Learning process

There are pros and cons of supervised learning algorithms. Supervised learning algorithms are helpful in making predictions based on prior experience. On the other hand, they struggle to perform complex tasks and they require a huge amount of datasets to train, validate and make predictions and classifications.

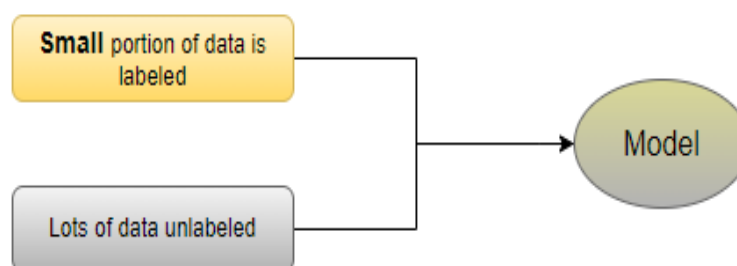
## 2.2.2 Unsupervised Learning

Unsupervised learning is assumed to have training examples that are unlabeled. Unlabeled training examples are “*learned*” depending on their similarities and therefore it is important to define the similarity metric among them. A typical task to which unsupervised learning algorithms are applied is clustering. K-Means Clustering, Mean-shift, AHC, DBSCAN and EM algorithm, and The Kohonen Networks, etc., are all unsupervised machine learning algorithms. Some of these unsupervised learning algorithms have been applied in the field of medicine [28], in the research area for text mining [29], in solar panel deployment optimization [30].

Unsupervised learning algorithms have some advantages which include the use for more complex tasks compared to supervised machine learning and data can be smaller in size as compared to the supervised learning. On the other hand, due to the lack of a corresponding output, unsupervised learning is generally more challenging to implement and explain than as compared to supervised learning.

## 2.2.3 Semi-supervised Learning

The concept in semi-supervised learning is to use both labeled and unlabeled training examples for the training of the algorithm. Numerous studies have shown how the amount of labeled data necessary to obtain adequate supervised learning performances can be greatly reduced by the information learned from unlabeled data [31] [32]. Semi-supervised learning is described using a block diagram in *figure 2.7* below.



**Figure 2.7:** Block diagram of Semi-supervised Learning

Research works like *Aissaoui et al. [33]* proposes a combination of supervised algorithms and unsupervised methods in predicting learners' learning styles. Semi-supervised learning methods are also applied by *Wu et al. [34]* to develop a model that finds a specified vehicle from many vehicles.

An advantage of this type of Machine Learning is that it helps solve the drawbacks of Supervised and unsupervised Learning algorithms. A possible disadvantage is that iterations results may not be stable and less accurate.

## 2.2.4 Reinforcement Learning

Reinforcement learning works on feedback-based process, in which an agent automatically explores its surroundings by acting, learning from experiences, and improving its performance. It is based on the reward and penalty system where parameters are updated for getting and maximizing the reward while avoiding and minimizing the penalty. In this learning type, the input is from the external environment and the output is generated as the action. Q-learning is the most popular algorithm under this category of machine learning. Recently, reinforcement learning has received considerable attention, with many successful applications in various fields such as game theory, operations research, information theory, simulation-based optimization, control theory, and statistics. Applied in health care research *[35] [36] [37]*, in robotics *[38] [39] [40]*, as well as in autonomous vehicles *[41]*.

Using reinforcement learning has an edge over other forms of machine learning because it can be used to solve very complex real-world problems. The downside of it though, is that it usually requires long computational times for successful training. Also, reinforcement is very powerful such that too much training can make the algorithms over militarized with input data, which affects prediction.

### Q-learning

Q-learning is a reinforcement learning technique used for learning the optimal policy in a Markov Decision Process. Q-learning is probably the most applied representative reinforcement learning and is an off-policy algorithm that looks for the optimal action to take in a given

state. It is deemed off-policy because a policy is not necessary. Q-learning function learns from actions outside the given policy by making actions random.

First, reinforcement learning uses the Markov Decision Process and the value function to construct the Bellman equation, then Q-learning is applied to solve the Bellman equation problem [42]. The Q-value is also called action value, which is defined as the expected long-term return with a discount when taking a given action.

- **State:** The state is a set,  $S$  of agent observable states.
- **Action:** An action is a set of possible actions,  $A$  in a state,  $S$ . Usually, the actions that an agent can do are the same in all states.
- **Reward:** The reward is the information given to the agent in the environment so that the agent can learn.
- **Policy:** When an agent arrives at a certain state, it determines the action using policy. Reinforcement learning approach learns better policies than the current one to obtain an optimal policy.
- **Value function:** For the agent to calculate the reward that he will receive in the future it has to consider which action he will perform. The criterion that determines which policy is a better policy is the value function.

Q-learning algorithms can be classified into single-agent and multi-agent algorithms. Single-agent algorithms include basic q-learning, deep q-learning, Hierarchical q-learning. Examples of multi-agent algorithms are modular q-learning, and q-learning, swarm-based q-learning, etc.

The Q-learning algorithm has been used in a variety of fields due to its success, including industrial processes, network processes, game theory, robotics, operation research, control theory, and image recognition [42]. Variants of Q-learning have been applied to games in helping find optimal moves for the games themselves thereby outperforming humans [43]. In robotics, q-learning is introduced together with other theories in generating continuous actions for robots [44].

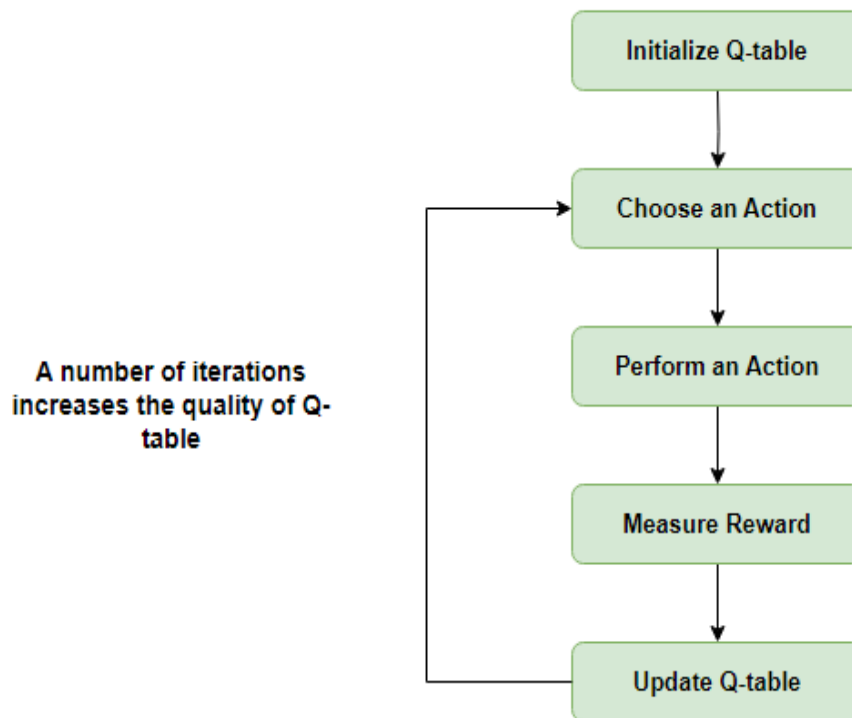


Figure 2.8: The flow to update Q-tables in Q-Learning

### 2.2.5 Automated Machine Learning (AutoML)

As pointed out earlier, ML continues to grow at an astonishing rate and recent advancement has made it possible to integrate ML in different sectors. One key challenge in fully embracing ML in the various sectors though is the need for coding (or programming) experience. This barrier can be overcome through the availability of platforms that provide automated and coding-free ML services [45] [46] [47].

The development of automated ML (autoML) platforms has shown benefits to the users who have little to no knowledge of manual ML approach. Such platforms include Dataiku DSS, Google Vertex AI, BigML, DataRobot, Exploratory, RapidMiner etc. Some advantages of using autoML include:

1. Compared to manual ML, which involves lots of codes, execution time is drastically reduced and still maintains high performance.
2. The option of tuning hyperparameters makes it possible to achieve even higher performance.

3. Since there is little to no programming experience required to use, people from diverse backgrounds can still use ML and achieve reliable results.

## 2.3 Related Work

Numerous ML research has been conducted in various fields in the past two decades, because of the increased availability of data and the quick development of AI technology. The Oil and Gas industry is no different, and in recent times, automation of the various processes in the industry has been the number one goal of most companies [23]. Using advanced technology in the design process would not only reduce the time spent and cost involved in the well planning process, it would also produce highly articulated well trajectories adapted to reservoir properties compared to the traditional methods [48].

Research on developing ML models, specifically RL models, to help design the well path is very few and little success has been achieved so far. Works using search methods discussed earlier in *section 2.1.6* differs from RL, in the sense that search algorithms explore search spaces, while RL models learn a policy that maximizes the expected reward by interacting with an environment. RL algorithms can be used to solve more difficult problems compared to using traditional search algorithms.

In this section, works that are similar to what is done in this thesis are discussed. Previously proposed solutions by other researchers are either in 2D or 3D environments.

### 2.3.1 Proposed Solution in 2D environment

*Ehsan Chowdhury* [49], in their thesis and a later published article [50], investigated the possibility of optimizing the well path using RL. A 2D program in Python using deep Q-learning algorithm trained, where the agent takes series of actions based on Q-tables until the set goal is reached. To imitate a real-world scenario, other well paths representing obstacles are introduced into the environment. Obstacles are categorized into hard obstacles and soft obstacles. Hard obstacles are no-go zones in the environment while soft obstacles are defined as avoid, if possible, but if not, the algorithm can go through. The goal was to avoid collision with any

of the nearby wells. The shortest path is found from the Q-Table. The program ends if any of the defined terminal states in the study is reached. The results in the study are visualized in a grid-based window.

Although the algorithm can efficiently find the trajectory with the least length and avoid collision, it still does not solve the issue in question. The path created in the 2D grid environment is not smooth and realistic enough as the algorithm does not consider the various technicalities in well path design like dogleg and dogleg severity.

### 2.3.2 Proposed Solutions in 3D environment

A more realistic environment, in the form of 3D environment, has also been used in the search for an optimal way of designing the trajectory of the well with reinforcement learning. *Ehsan Chowdhury* [49] explored this option as well. The implementation is built in Unity, a game engine. ML scripts were used to set up the environment that allows building objects, determine the functions of the objects, and give an output. The agent gets information about the various objects from the scripts, runs training episodes and finds the optimal path to the specified point. Obstacles used are vertical and horizontal cylindrical obstacles. The limitation with this work is that it lacks in performance and also does not consider curvature limits.

A different 3D attempt has previously been made by a different author, *Haque* [51]. They suggested using the MRST Matlab program to plot a 3D grid, which then served as an environment for the algorithm to learn. The goal of the work was to create a path that connects the surface and the target by avoiding other nearby wells. The 3D environment contains 3D cubes generated using the north, east and TVD values. Each cube has six different directions, and the agent can decide on the best option. The optimal path is found by the agent based on the total cost/reward. The result from the research proved to have some interesting and valuable qualities. However, the computation time of the MATLAB program is too long and the whole goal of this study was to reduce the time needed to design a well trajectory.

In a research work reported by *Yu et al.* [52], deep reinforcement learning was employed to train the agent in a simulated environment to develop an automated drilling agent. Although not necessarily related to the work done in this thesis, the approach has some possibilities that could



be explored for research in trajectory design. They designed a motor-based drilling system that interacts with the environment (i.e., formations, wellbore geometry, equipment) given a predefined well trajectory to drill an actual well.

The agent perceives the various states such as inclination, measured depth, true vertical depth, and the planned trajectory to decide how best to move. The reward for the agent in this RL training is based on how well the agent can follow the planned trajectory. With a series of training, the agent gained the ability to steer through random exploration in the simulation environment.

# Chapter 3

## Platforms, Dataset & Model

To aid in the implementation of the proposed work in this thesis, some software platforms such as DELFI by SLB, Dataiku DSS and OpenZgy were used. This section gives a general overview of the of these platforms and explains why they were chosen for this work. *Sections 2 and 3* of this *chapter* introduces the dataset used and the model, respectively. This is an essential part of the thesis, and it would further help understand all the elements involved in implementing the resulting model.

### 3.1 Platforms Used

#### 3.1.1 OSDU Data Platform

The Open Subsurface Data Universe (OSDU) is a data platform that was developed to help solve the data challenge in exploration and production in the migration to the cloud. This platform is a cross-industry collaboration that addresses the problem of storing, organizing, migrating, and accessing subsurface data on the cloud [53]. OSDU platform is at the heart of the DELFI digital cloud platform, which means that platforms that exist on DELFI such as Dataiku DSS communicate directly with OSDU and fetch their required engines to perform their tasks from there.

### 3.1.2 DELFI Digital Platform

The DELFI digital platform is a secure, cloud-based space built on key premises that make it unique. The platform makes applications and workflows accessible to all users and enables team members to build common workspaces for data, models, and interpretations. The DELFI Data Science section is a package of AI and analytics solutions for energy workflows that enables geoscientists and engineers to build, manage, and deploy AI solutions to challenges. SLB partners with Dataiku to provide a cloud-based environment to make AI and analytics work.

### 3.1.3 Dataiku Data Science Studio (Dataiku DSS)

Dataiku DSS is a software platform that systemizes the use of data and AI, bringing people together to deliver desired results. It allows for easy collaboration, where programmers and non-programmers can simultaneously work on the same data projects in a shared space. Some of the key capabilities of Dataiku platform is that it allows Data preparation, Visualization, AutoML, Dataops, MLOps, Security, Collaboration, Architecture, etc. [54]. For this thesis, much more focus is placed on AutoML ability of Dataiku, as an approach with less coding is preferred. In comparison with other AutoML platforms and manual ML approach, Dataiku has produced equal if not better performance [55] [56].

### 3.1.4 OpenZGY

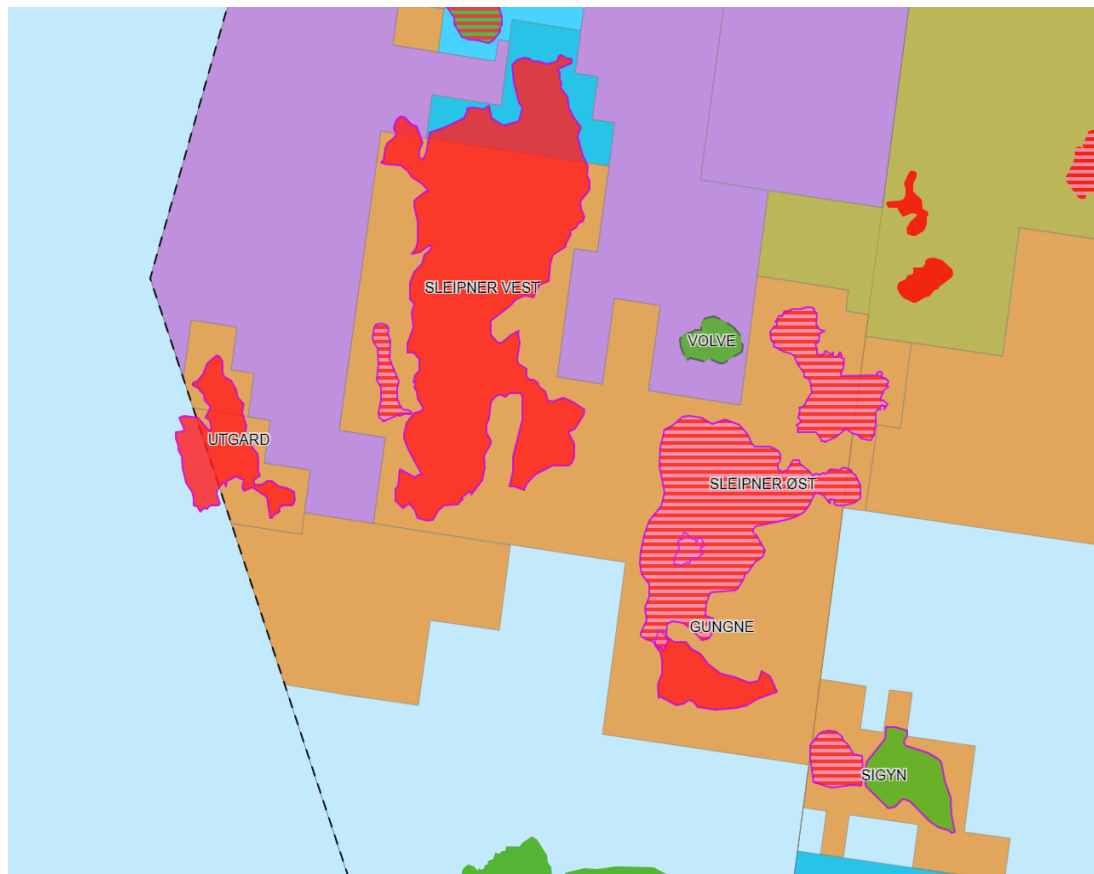
Seismic data is usually stored in a ZGY or SEG-Y format, especially if it is created in Petrel™ E&P Software Platform. OpenZGY is an open-source solution for how to ingest, store, compress, read and write ZGY seismic data [57]. It has a dedicated repository in Gitlab [58] and with it, interested people are able to read and write ZGY files. The objective of using OpenZGY in this thesis is to read from actual seismic with actual faulting and converting into csv format.

## 3.2 Dataset

Data is essential in research work as it provides means to assess findings on real information and not synthetic ones. Data availability has long been an issue in the Oil and Gas industry and has

been fully dependent on laboratory scale work and commercial partners [59] [60]. Equinor has taken steps to remedy this since 2018 by making available to the public the Volve field dataset.

### 3.2.1 Overview of the Volve field dataset



**Figure 3.1:** Location of Volve field in the North Sea

#### Volve Field

Volve is a field in the central part of the North Sea in the Norwegian Continental Shelf (NCS), which was produced from 2008 to 2016 and shutdown in 2018. Volve produced mainly oil from sandstone of Middle Jurassic age in the Hugin Formation at a reservoir depth of 2700-3100m. A total of twenty-five (25) wellbores were drilled: five (5) exploration, nine (9) production, eight (8) observation and three (3) injection wells.

## Volve Dataset

In 2018, Equinor decided to disclose a comprehensive and complete dataset, totaling approximately 40,000 files of various kinds, of all the subsurface and production data on the Volve field after it was shutdown. The idea behind the release was to provide data to the general public that is highly useful, contributing to further learning and experience transfer in the industry and in academia [61].

There are a total of fourteen (14) compressed archives available for download, and their size references are provided in *table 3.1* below. Information stored in these files include geological and geophysical interpretations, static and dynamic reservoir models, production data, well logs, well technical data, and real-time drilling data. Several research works have analyzed the distinct aspects of the dataset [62] [59]. *Viggen et al.* [60] for instance, investigates the well log data from the Volve Data Village dataset and explains how it could be processed and used for maximum result.

Description	Compressed size
Geophysical Interpretations	99 MB
GeoScience OW Archive	54.6 GB
Production Data	2 MB
Reports	162 MB
Reservoir Model (Eclipse)	390 MB
Reservoir Model (RMS)	2.1 GB
Seismic ST0202	1.2 TB
Seismic ST0202 vs ST10010 4D	330.4 GB
Seismic ST10010	2.6 TB
Seismic VSP	95 MB
Well Logs	6.9 GB
Well Logs (Per Well)	7 GB
Well Technical Data	212 MB
WITSML Real-Time Drilling Data	5 GB

**Table 3.1:** Volve Dataset files

Ever since it was made available, the dataset has been used by various researchers including students in their research works. Research involving Machine Learning methods have benefited the most from the dataset unveiling. *Tunkiel et al. [63]* used the dataset in researching into improving rate of penetration by applying Machine Learning methods. It has been used in other studies such as improving the quality of oil production estimation and estimating reservoir characteristics for appropriate wells placement [64], in production prediction and ultimate recovery [65].

### 3.3 Model

This section discusses some technicalities that are to be considered in the implementation process. It defines the most basic components of a well path that would have great influence in the success of the model.

#### 3.3.1 Technicalities to be Considered

Many factors influence the trajectory of the borehole and some of such factors can be very difficult to estimate. The experience gained from drilling previous wells in the same area is therefore very useful and should be incorporated at the planning stage of a new well.

In planning a well, it is more convenient if the curved surface of the Earth is projected onto a flat surface in a system such as the Universal Transverse of Mercator (UTM) [66]. Coordinates of various points such as the target must be converted and referenced back to a defined reference point/origin after they are taken from the UTM. All calculations made are then simplified by adopting the Northing and Easting coordinates relative to the reference point.

When planning the geometry of a well, some technicalities must be considered. For instance, the coordinates of desired start position (where the wellhead will be positioned) and the target point should be known relative to a reference point. The displacement of the target point from the start point can be calculated using the coordinates by:

$$H_t = [(N_t - N_a)^2 + (E_t - E_a)^2]^{\frac{1}{2}} \quad (3.1)$$

Where:

$H_t$  = Horizontal Displacement

$N_t$  = Northing of target

$E_t$  = Easting of target.

$N_a$  = Northing of start point

$E_a$  = Easting of start point

Proposed direction or target bearing  $\beta$  can be found by calculating the angle formed between the well trajectory and the horizontal plan:

$$\beta = \tan^{-1} \left( \frac{E_t - E_a}{N_t - N_a} \right) \quad (3.2)$$

Vertical sections are a vital part of the well path. As part of the drilling technicalities, a well should have a vertical section from the start point to a distance (typically ranges from 100m to 1000m). At this point, if a directional well is desired to be drilled, kick-off begins.

In theory, an average borehole curvature can be calculated. The borehole curvature represents the curvature of the borehole axis. This borehole curvature has to be within reasonable drillable limits to avoid complications. Both inclination and azimuth vary as the drilling progresses, and the overall angle change of the borehole per interval length indicates the average borehole curvature, which can be expressed as follow:

$$\gamma = \sqrt{\delta\alpha^2 + \delta\phi^2 \sin^2 \alpha_c} \quad (3.3)$$

$$\alpha_c = \frac{\alpha_1 + \alpha_2}{2} \quad (3.4)$$

Where:

$\gamma$  = Overall angle change within a certain survey interval, ( $^\circ$ )

$\alpha_c$  = Average inclination in a certain survey interval, ( $^\circ$ )

$\alpha_1, \alpha_2$  = Inclination of the upper and lower survey points, respectively, ( $^\circ$ )

$\phi$  = Azimuth, ( $^\circ$ )

It is also important to note that although the target is defined as a point with north and east

coordinates, it is very difficult to drill to a defined point in real drilling scenarios. Therefore, acceptable boundary limits are set, and a zone is rather targeted. Entering the reservoir through anywhere within the set zone is determined to be a success [67].

### 3.3.2 Reward Function

In a real-world scenario during drilling, hitting a constraint is a bad thing and very much undesired. Drilling through different types of constraints lead to different levels of consequences. For example, the cost of drilling through an already existing well causing collision between the wells can have potentially catastrophic consequences, particularly if the offset well impacted is a producing well. Such type of constraint should be avoided at all costs.

The model intended to be implemented follows similar ideology. There are rewards (or costs) for every action the algorithm takes. The size of the reward depends on the possible consequence. In the example stated above, the reward for hitting that type of constraint would be the least possible reward. Assigning a small reward is considered as a punishment to the algorithm for making the decision to go through such a risky zone. Other less risky zones are also assigned low value rewards but not as low as the rewards for hitting a hard constraint. For example, if the reward for a hard constraint is set as  $y$ , a soft constraint could be set two times bigger, that is  $2y$ . A reward ten (10) times bigger than that for hard constraints could be set for an ultra-soft obstacle since this type of constraint is not as risky and the path can go through it if there are no other options. The algorithm learns with every training episode to stay away from the obstacles.

The ultimate goal in drilling is to drill successfully into the targeted zone with as little problems as possible. In order to motivate the algorithm to reach the algorithm to reach that goal, the reward is set as high as possible. Every individual point should also carry some reward so as to differentiate them from the cells with obstacles.



x/m	y/m	z/m	MD/m	incl/deg	Azi GN/deg	Azi TN/deg	DX	DY	DX TN	DY TN	z tvd	TWT	DLS
435247.31	6479164.651	-2512.36	145.9	0	0	359	0	0.00	0	0	0		0
435246.801	6479183.15	-2549.04	150.15	0	0.94	0	0	0.00	0	0	0		0
435245.078	6479146.922	-2477.25	160.45	0	0.94	0	0	0.00	0	0	0		0
435243.144	6479201.413	-2584.99	170.04	0	0.94	0	0	0.00	0	0	0		0
435240.137	6479128.578	-2441.27	179.79	0	0.94	0	0	0.00	0	0	0		0
435235.976	6479219.619	-2620.2	189.74	0	0.94	0	0	0.00	0	0	0		0
435235.293	6479116.688	-2418.33	198.5	0	0.94	0	0	0.00	0	0	0		0
435226.978	6479101.812	-2388.8	210.7	0.28	302.21	301.26	0	0.00	0	0	0.705		0.705
435224.857	6479237.111	-2654.62	237.60	1.55	303.2	302	-0.36	0.23	-0.36	0.22	1.41		1.415
435218.823	6479089.669	-2363.65	251.1	2.03	304.4	303.52	-0.73	0.47	-0.74	0.46	1.06		1.06
435210.088	6479254.024	-2688.67	264.9	2.22	305.77	304	-1.16	0.77	-1.1	0.75	0.4		0.430
435206.888	6479073.031	-2328.48	279.2	2.08	305.8	304.9	-1.61	1.1	-1.6	1.08	0.29		0.29
435192.925	6479054.308	-2288.39	292.6	1.63	309.81	308.8	-1.98	1.37	-2.01	1.34	1.043		1.043
435192.621	6479270.467	-2721.03	302.8	1.34	321.6	320.6	-2.16	1.5	-2.19	1.5	1.25		1.25

**Table 3.2:** Sample of a Well Trajectory Spreadsheet for F-1 A well from Volve dataset taken from Petrel

# Chapter 4

## Implementation

This chapter explains in depth the proposed solution for using reinforcement learning to design a well trajectory. It explains all the experimental procedures that led to the final system implemented. The idea implemented in this thesis follows a similar approach to the ones previously discussed in *section 2.3*. It employs basic q-learning as the reinforcement learning algorithm as it is more effective than other reinforcement algorithms like SARSA.

Obstacle avoidance is very key in this work since it is modelled to be as realistic as possible. Heavily faulted zones, especially, are to be avoided since the risk of drilling complications is extremely high. In this study, the focus is more on designing a well path that avoids obstacles like heavy faults in formations and avoid clashing with other well paths in the same formation.

### 4.1 Introduction

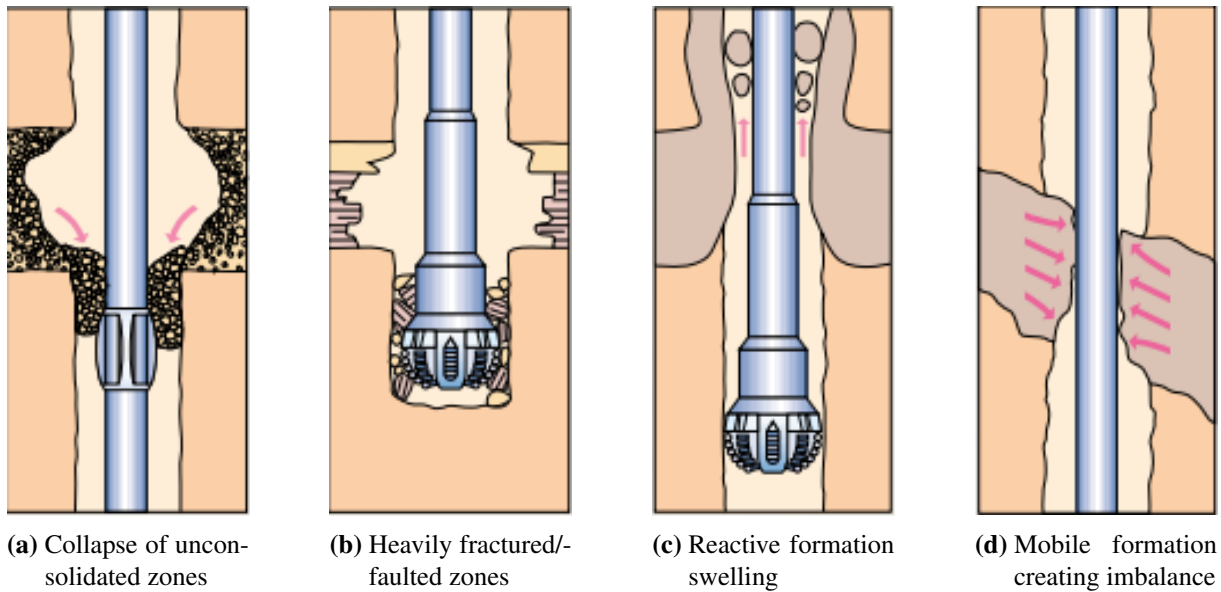
This section explains in detail how the basic aspects of the model are set up. Basic aspects of developing the model such as the start and end points of the search process and obstacles are paramount in the trajectory design task.

#### 4.1.1 Start and Target Points

The implemented model has two (2) points, just like in a real drilling scenario: a start and a target point. The start points stand for a point on the surface where drilling starts (for example wellhead on land and subsea wells), and the target is the point in the reservoir where the well

path ends. In a fully automated well plan development, a probabilistic approach could be used to determine the wellhead location. A semi-automated approach, whereby the input of a drilling engineer to determine the ideal start and end location of the well, could also be used.

### 4.1.2 Obstacles



**Figure 4.1:** Illustration of the effects of drilling through risky zone [68]

Obstacles are drilling risks/hazards which the well path should avoid. The impact of each drilling hazard is ranged according to the resultant costs involved in drilling through these obstacles. Therefore, in this 2D grid case, the drilling risks or obstacles are defined into 4. They are ultra-soft, semi-soft, soft, and hard obstacles.

Very high risky zones are defined as hard obstacles and the well trajectory should not go through them. Such zones include well paths of nearby wells and heavily fractured/faulted zones. Soft obstacles are much less risky zones compared to hard obstacles. In a drilling case, soft obstacles include unconsolidated sand formations and highly mobile zones (salt formations). These zones can be drilled through if proper caution, such as slowing down tripping speed, is taken by the drillers. Semi-soft obstacles include reactive shale formations. Reactive shale absorbs water and swells leaving the borehole with smaller diameter. A way of preventing such zones from causing drilling problems is to use an inhibited mud system like potassium/polymer fluids. Ideally such a zone should be avoided but if the trajectory must go through it

to avoid going through a hard or semi-soft obstacle then it is preferred. The last obstacle which is the ultra-soft obstacle has the least restriction. In an area with multiple wells, anti-collision of the well paths is a priority and they must be kept apart as much as possible. In some cases, new wells can be drilled just a little closer to others within a set limit. Ultra-soft obstacles fall under this limit.

This 2D RL model is designed to avoid the high risks zones completely while searching for the shortest well path. The ultra-soft and semi-soft zones can be drilled through but at a cost (negative reward value). Examples of such high-risk zones causing drilling problems are illustrated above in *figure 4.1*.

## 4.2 2D Approach Using Q-Learning

The initial approach is designed in a 2D environment. The environment is designed to be as realistic as possible, although with some differences compared to an actual well design. For example, the position where drilling of the conductor begins (position of wellhead) is predetermined and fed to the model. The target in the reservoir is also specified and fed to the model.

### 4.2.1 Initial Proposed Model

Different approaches are assessed to find the best system. The Initial approach in this work is similar to what *Ehsan Chowdhury* did in their 2D program discussed in *section 2.3.1*. The objective was to expand on the previous work done by introducing more obstacles to increase the complexity and make training results more accurate. The 2D program is written in a grid-world environment and at first try a 30x30 window size by height and width respectively is used. Only two obstacles are added because of the small size of the window. A feasible way to improve on this result and make the model more robust is to increase the size of the working environment, which makes room for more obstacles to be added.

To emulate an actual formation, the environment created is populated with lots of obstacles. Seismic data in the ZGY format is first exported from the open source Volve field dataset. Open-ZGY library is then used to read the zgy file to extract the different faults in the seismic data.

The faults are represented as spikes in the amplitude values and the coordinates of these faults are imported into the training environment. This results in a substantial number of obstacles positioned randomly in the training environment and gives it a more realistic feel. All this was done on Dataiku DSS because the jupyter notebook feature on Dataiku has all the necessary libraries built into the kernel.

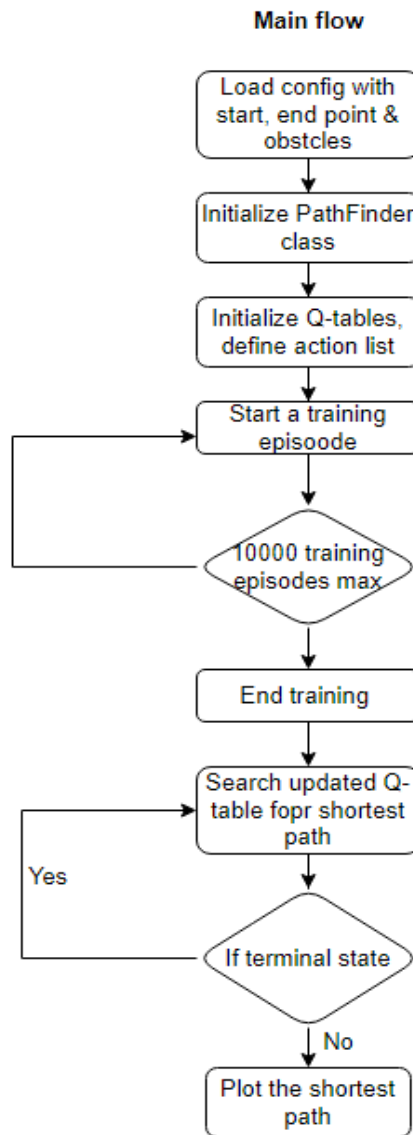
The grid size is then increased to a 300x300 as the next approach, thereby allowing to populate the environment with the fault coordinates imported from the Volve seismic data discussed in *section 3.2.1*. This would create a more practical environment for training and improving the model. This approach is computationally expensive and time consuming, since the grid space for model training is larger.

## 4.2.2 Flow of Implemented Model

This section presents the details of the model. It follows the flowcharts diagrams in *figure 4.2* and *figure 4.3*. The main 2D program flowchart summarizes the steps involved; from creating the environment and populating it, running training, and getting the shortest path. The other flowchart zooms in to the various steps a single training episode goes through. It starts from finding a random start position, then it takes several actions while receiving rewards to update the q-table. The random start position is only applicable during the training phase.

### Adding configuration

The configuration of the 2D program begins with setting some initial conditions. The environment window is made to a scale of one (1) with both height and weight set to 32x32. The start and end points are defined as positions in the 2D grid using the grid cell indexes and are represented as startLoc and endLoc, respectively. Just like in a real-world scenario, obstacles are added to the configuration and their positions are specified in the grid. There are four (4) distinct types of obstacles in this design: ultrasoftObstacles, semisoftObstacles, softObstacles and hardObstacles as explained in *section 4.1.2*. Each obstacle carries its own weight which helps in the model training. Also, a reward of -1 is set for every cell in the grid. *Adding configuration.py* in *appendix A* has the details.



**Figure 4.2:** Flow of the main 2D program

Ideally the resulting path should avoid all the four types of obstacles, but just like in an actual formation, some obstacles can be ignored since the risk of going through them is not much. Hard obstacles are to be avoided at all costs, but the path can go through soft obstacles if the cost of avoiding the obstacle is higher than the cost of passing through it. Rewards are set for the individual obstacles as well as the the find the endLoc successfully.

A class is created and named “*PathFinder*”. The class is responsible for finding the optimal path in the given environment. The *PathFinder* has under it some functions that define *environment initialization, terminal states, the next action, the next location etc.*

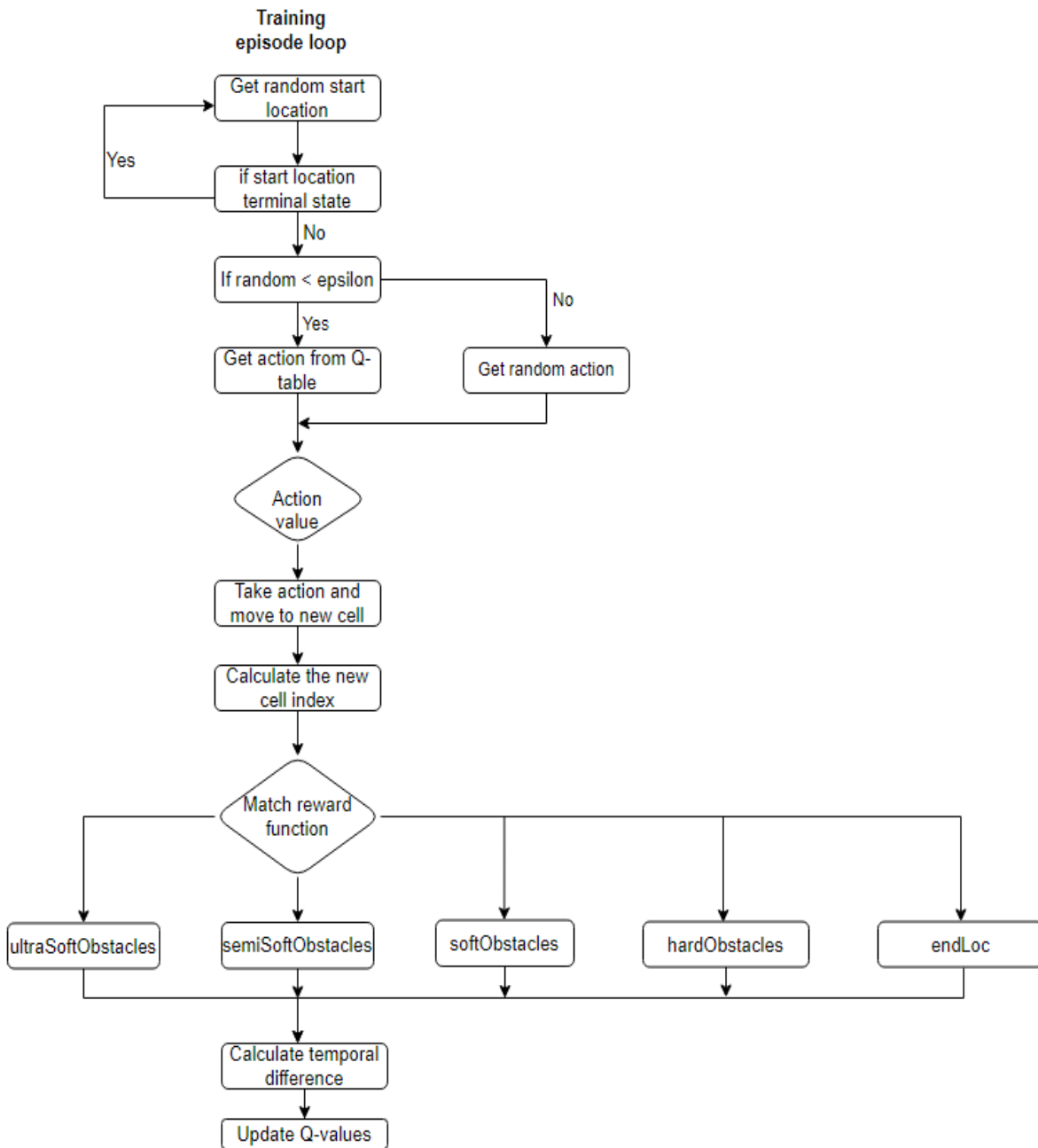


Figure 4.3: Flowchart of a single training episode of the 2D model

## Initializing Environment

After adding the initial configuration, the next step is to initialize the environment using the configuration. This involves setting up the possible actions to be made at every instance (that is, whether to move up, down, right, or left in the grid), creating a table of q-values, which determines the best possible action for every movement. Since the startLoc can never be in a hard obstacle, a line of code is added to check that, and it raises an *AssertionError*.

### Terminal State

There are some conditions set in the algorithm that terminates/ends the path search whenever those conditions are met. These conditions include running into a hard obstacle and reaching the target point (endLoc).

### Search for endLoc

The algorithm is tasked with moving from the startLoc to the endLoc by making a series of decisions on the direction to move. It chooses the next actions based on the *get\_next\_action* function listed in the *appendix A*. The next action is taken into a cell to the right, left, up or down of its current position based on the one with the highest reward. An element of randomness is added by introducing an epsilon. Epsilon has a value between 0 and 1 and it determines how random the actions taken must be and it helps the algorithm to explore more.

Once the best possible action is taken, the new location is retrieved. The new location is represented as a position in the 2D grid with the cell indexes as an array of [rows, column].

### Training of the algorithm

In the training of the algorithm, the start location is not defined. The algorithm is made to choose a start location at random in every training episode. In this case, the algorithm learns more about the environment as it is able to explore more space due to the different start locations. *Search for endLoc.py* defined in the *appendix A*, a function called *get\_start\_location* shows how the algorithm is made to get a random start location every time. The resulting cell with row and column indexes selected should be valid and not a terminal state to be accepted as a start location.

During the training episodes, some hyperparameters are required to help fine-tune the algorithm and make it more accurate. These hyperparameters include the epsilon, discussed in *section 4.2.2*, learning rate and discount factor. Learning rate determines to what extent new information overrides the old information. The value is between 0 and 1, where 0 means the algorithm does not override old information and learns nothing. 1 on the other hand means new information replaces the old information. Discount factor determines if future reward is



to be considered or not. A discount factor of 1 means the algorithm strives for the best overall future result instead of the short-term reward after an action is taken. The hyperparameters are implemented as shown in *Training the algorithm.py* in *appendix A*.

An extra code is added to the training to plot out some of the training episodes that got that failed as a result of the algorithm getting "*stuck*", causing termination of the episode.

### **Shortest path to endLoc**

After several training episodes (10000 episodes in this model), the algorithm is made to get the shortest path from the start point to the end point among all the successful training episodes. *Shortest path to endLoc.py* code in the *appendix A* gives the detailed code.

The workflow of the implementation could be summarized as following. First, the algorithm chooses a random point in the environment filled with obstacles as start point for training. It then takes a series of best possible actions at every point while considering the overall reward while exploring the environment. Each new location is made known until the algorithm reaches the defined target point. A shortest path connecting the startLoc and endLoc is plotted out. The cycle is repeated for 10000 episodes, in all episodes considering epsilon, the learning rate and discount factors. After training, the shortest path connecting the start point to the target point is plotted and visualized using the OpenCV library in Python and the q-table is updated according to the Bellman equation.

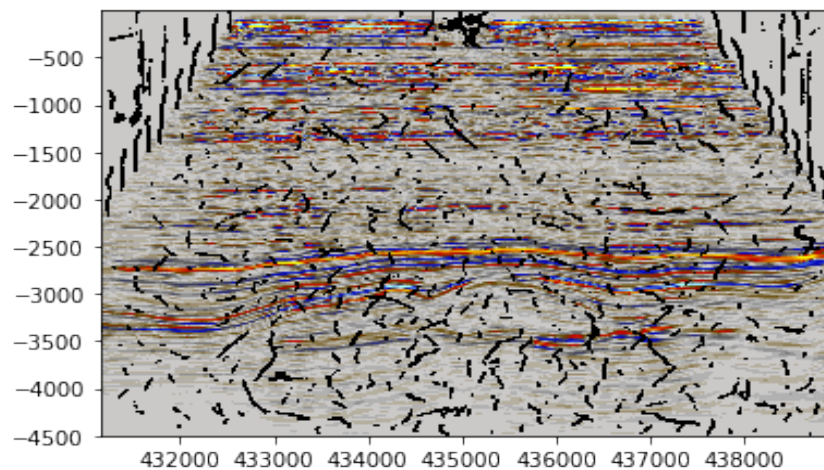
# Chapter 5

## Results and Discussion

This chapter presents the results achieved from 2D reinforcement learning described and implemented in the previous chapters.

### 5.1 2D representation of VOLVE field

To create a 2D environment for our RL algorithm, analysis of a real reservoir and well was done to show a visual representation of the aim of this study. In this work, a 2D representation of a section of Volve field was used. Seismic inline X and Y values, inline amplitude and faults are visualized in a 640 x 1125 diagram, to represent a close to ideal case for the study. The figure below shows a section of the Volve field plotted using `openzgy.api`, `numpy` and SLB's Python Software Development Kit (SDK) for Dataiku DSS.



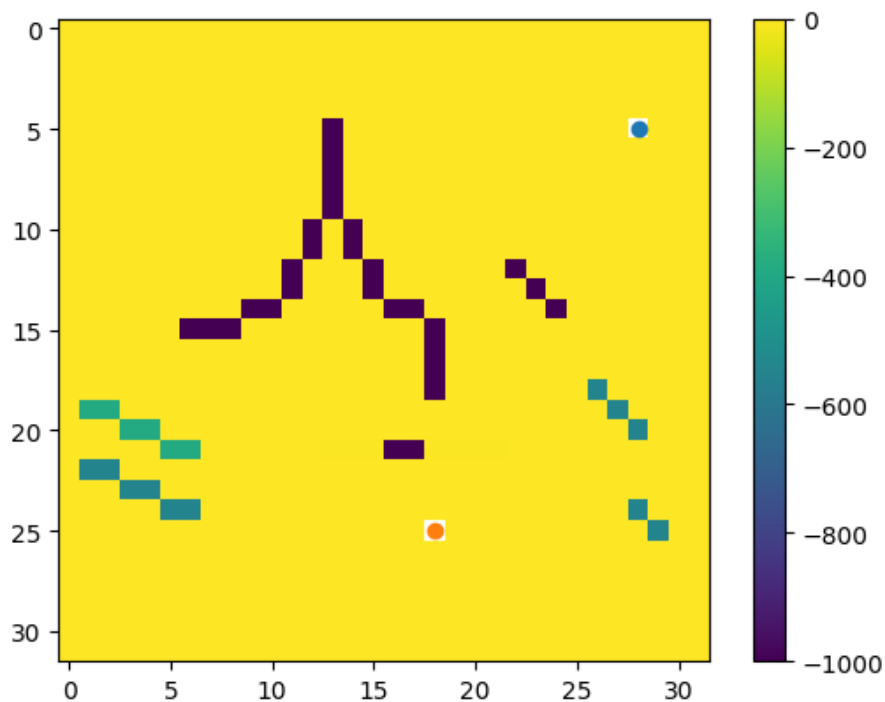
**Figure 5.1:** *Display of a Section of Seismic cube from Volve Field*

The black color shows the faults that are projected on the inline, while the multi colored areas shows the different seismic amplitudes, the peaks and the troughs of the seismic/sound signals, in this section of the reservoir.

## 5.2 RL 2D Grid Environment

Initial attempts to use the data and section of the Volve field to create a 2D environment for training our RL algorithm proved to be computationally intensive due to the wide range of points in the datasets and the computational intensity of the resources and applications available.

To mitigate this, a 2D grid was created as a simplified recreation of the Volve dataset for this case study and RL environment. A 300x300 grid required days of training with grid obstacles. This led to a further down-scaling of the grid to a 32x32 grid to achieve faster results and visualisation. *Figure 5.2* shows the visual representation of the training environment.

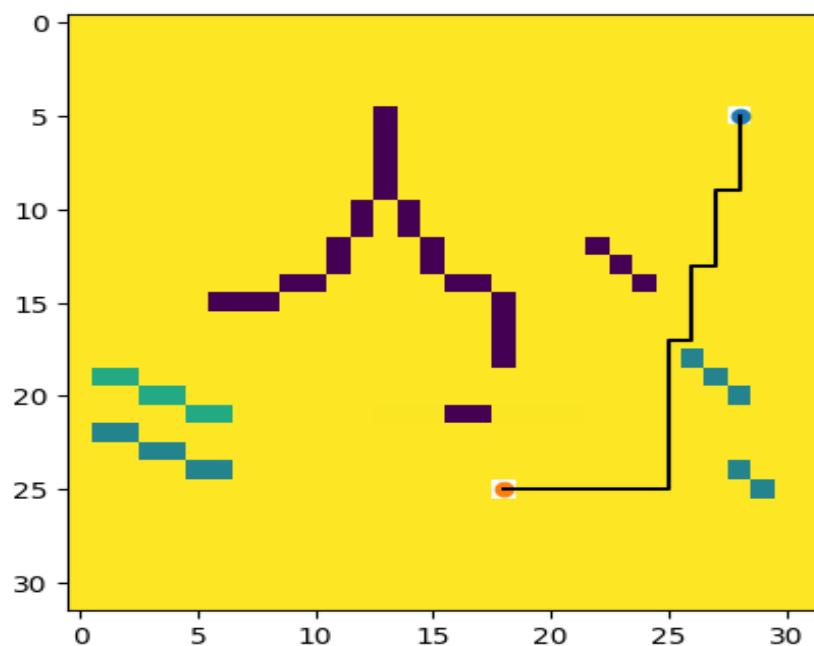


**Figure 5.2:** 2D grid representation of a RL environment with start and end location

In the diagram above, the a color code of -1000 (dark purple) represents a hard obstacle as well as a no-gone zone during drilling or training of the algorithm. The lighter greens, or less negative colour coded areas, represent soft, semi-soft and ultra soft areas in which the a well

path can pass through at a cost. The blue dot is the start point (well head) while the orange dot is the end point (reservoir target area).

*Figure 5.3* shows a diagram of an ideal shortest path in the chosen RL 2D training environment. In this example, this is trained for 10000 episodes. The randomness effect in this algorithm tends to bend the well path as much as possible to represent an ideal well path. Since this is a 2D grid environment and drilling parameters such as 3D coordinates, azimuth and angle of inclination are not considered, the path tends to be straight lines in both horizontal and vertical directions.



**Figure 5.3:** Shortest well path in a 2D grid environment

Although this can be considered to be too rough of an estimate, by upscaling using a post-processing spline interpolation method, this issue can be mitigated as shown in *figure 5.4*.

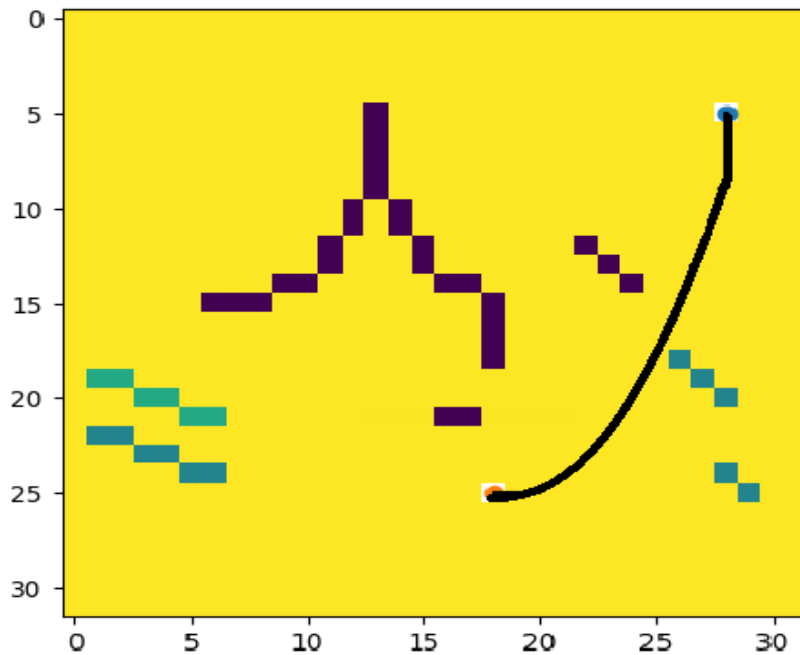


Figure 5.4: Upscaling well path

## 5.3 Sensitivity Analysis

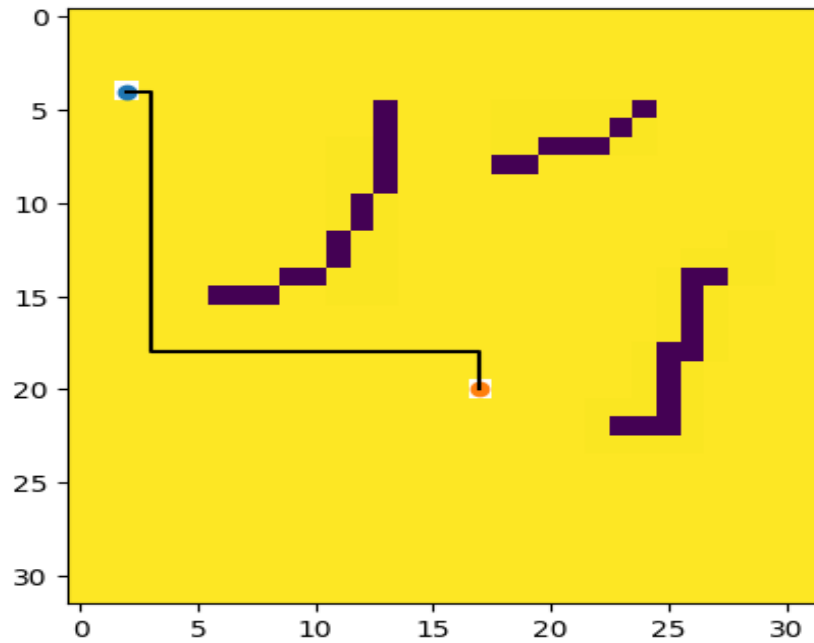
This section addresses various effects that affect the algorithm in determining the shortest path in our 2D training environment. These effects include: grid size and location of start and end points, reward or cost factor and hyper-parameters.

### 5.3.1 Effect of grid size location, start and end points in finding the shortest path

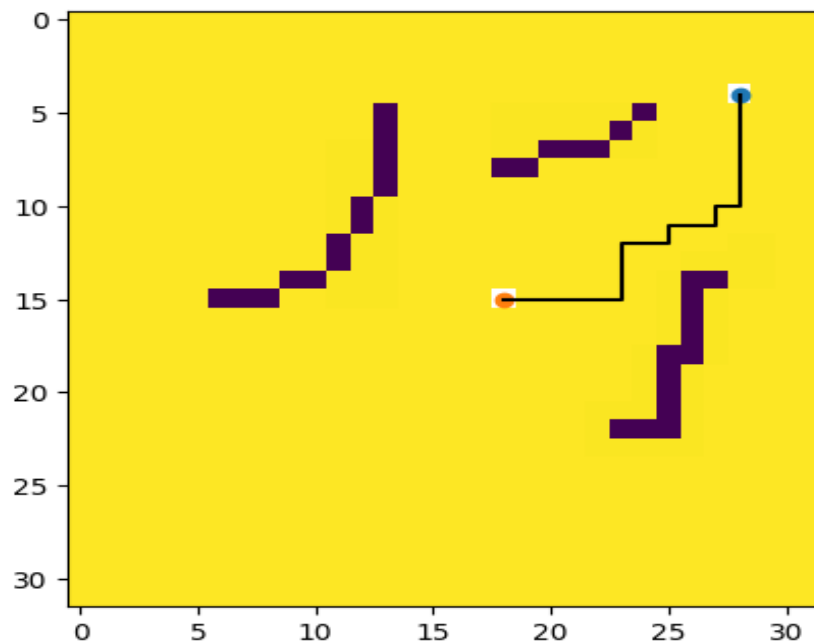
To analyze how the grid size, and start and end location influence the algorithm in finding the shortest path, a constant values for hyper-parameters such as reward, number of obstacles, training episodes, epsilon, discount factor and learning rate are maintained. The average number of steps and time in finding a shortest path while changing the start and end location is measured. It was observed that, the closer the points are to each other, the smaller the number and steps and amount of time in finding the shortest path.

In *figure 5.5*, the average training steps was 47 while the training time in finding the shortest path is 0.0005 seconds.

In *figure 5.6*, the average training steps was 55 while the training time in finding the shortest



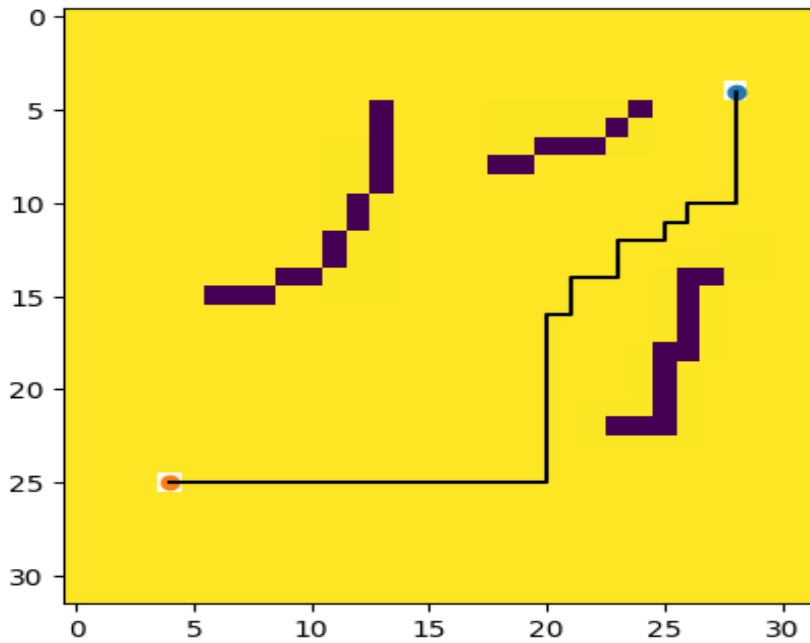
**Figure 5.5:** Short location distance. End location is in-front of an obstacle



**Figure 5.6:** Short location distance between start and need location. End location is not behind an obstacle

path is 0.00035 seconds.

The further the start and end locations are way from each other, the larger the number of steps and search time. In *figure 5.7*, the number of steps increased to an average of 76 steps while the average training time is 0.0045. The start and end locations are row 2, column 30 and row 30 column 1 respectively.



**Figure 5.7:** Increased location distance in 2D grid

A large grid size demands more computational power and more obstacles for training. One 300x300 2D grid environment took 384 hours to train for just 100 episodes and a limit of 100 steps. This and other grid sizes do not lead to significant results and the cost of training only increases. So the grid size was reduced for further experimentation and analysis.

### 5.3.2 The effect of reward or cost factor in finding the shortest path

The RL algorithm rewards each move during search for the shortest path. A higher cost is assigned to hard obstacles as they are seen as no-gone zones or areas for drilling. A cost of -100000 is used. Also, if a training episode locates a hard obstacle, it is configured to stop immediate training and search and the process is repeated again. The shortest path can pass through all kinds of soft areas but a cost that range from -550 to -50.

This set up has resulted in multiple failed training and the need for higher episodes/epochs for training each environment. The bigger the grid, the higher the number of training episodes needed. Larger grids also demand 100s of obstacles to help in training and finding the shortest path. The results is an increase in computation power and time per training. An example of a failed training episode out of the 10000 episodes, whiles training for the successful shortest path shown in *figure 5.3* is displayed in *figure 5.8*. It can be noted that the start position (represented

by a black dot in training episodes) is at a random position. The training fails after hitting a terminal state, a hard obstacle in this case.

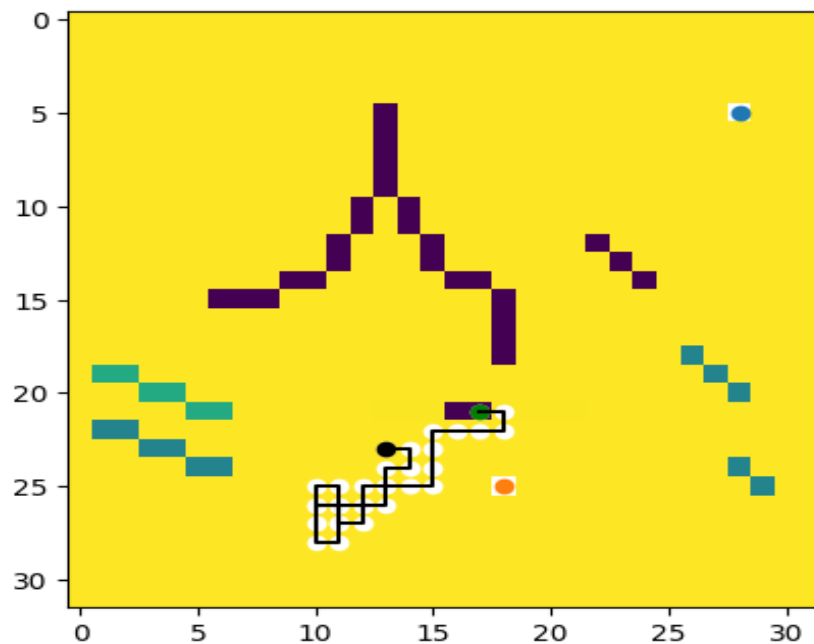


Figure 5.8: Failed training episode

Appendix B shows more examples of training paths per epoch in finding the shortest path.

### 5.3.3 Effect of hyper-parameters in finding the shortest path

Hyper parameters discussed in chapter 4 has an effect in training and achieving a smooth well path. The lower the learning rate with constant values of discount factor and epsilon, the less smooth the shortest drilling path. The best learning rate that was used was 0.85.

The lower the discount factor and epsilon values, the more random the self learning process and the height the number of training episodes it takes for the model to achieve a smooth well path. In this work, the optimal discount factor is 0.99 while that of epsilon is 0.01.

All these values are obtained while training for 10000 epochs and a maximum steps to stop training at 100. It is expected that increasing the number of episodes would increase accuracy and make the model robust as the algorithm would explore more.



# Chapter 6

## Conclusions and Future Work

In this chapter, the work is summarized and a conclusion is drawn from everything that was done. A suggestion for potential future works is also presented.

### 6.1 Conclusions

The main goal of this thesis is to explore designing a well trajectory using reinforcement learning. In this work, it can be established that reinforcement learning is a good approach and is more than capable in helping in well trajectory design. This is due to its self-learning nature, making it possible to find multiple creative solutions that might otherwise be missed when using traditional methods. However, this requires a lot of computational power and further optimization, such as more realistic and varied constraints. 3D or 4D models should also be considered, but will need to evaluate well inclination, azimuth, coordinates and location of realistic drilling obstacles in more detail, to name a few.

The results mentioned and discussed in the previous *chapter 5* have been submitted and approved as a conference paper [69]. The conference is the AAPG ICE 2023 Madrid [70] under the "*Digitalization in Geosciences*" theme in the "*Machine Learning and G&G Workflows*" sub-theme. An oral presentation will be given surrounding the work done in this study. The review process showed a lot of interest in this topic, with multiple senior advisors applauding the results achieved.

The 2D model discussed in this thesis, although proved to be successful, has a few limita-

tions enforced on it. This was due to the time constraints implicit. There are a few improvements that are worth investigating moving forward. One such improvement is the use of a dataset in a more direct manner. Using such a dataset for training should give a closer and more realistic well path. This will, however, require much more training time and processing power.

## **6.2 Future Work**

The question can be asked whether an appropriate result was reached for this project according to goals and expectations that were set. Since the thesis work had a limited time to be turned in, there was not enough time to explore all the ideas to improve on the work. In this section more suggestions are made on what could be implemented for this project to further develop and make it a finished product.

### **6.2.1 Training data**

In the attempt to introduce faults from the seismic data during the 2D model implementation for an expanded grid (3000 x 3000), the algorithm ran for weeks and never gave a result. This is believed to be due to the size of the training dataset used in that instance. Finding a way to optimize for this size of datasets, without losing critical information, could be extremely useful for future work.

### **6.2.2 Adding more granularity to the obstacles**

In actual formations, there is much more granularity in the obstacles. One such example are the salt domes. Not only should one consider the body of the salt dome, but also the flanks and the deformed stratigraphic layers due to the salt body protrusion. All these factors would have differing levels of constraints, which will effect the overall cost effectiveness of the purposed solution by the algorithm. This should be considered and factored into future models.

### **6.2.3 Considering the technicalities of drilling**

There are some very well-known drilling technicalities that are not really considered in the thesis due to time limitation. For instance, it is well known that any deviated or horizontal well must start as with a vertical path for some distance before deviation kicks in at the KOP. Also, having multiple dogleg angles that are  $90^\circ$  would be difficult to drill and is unacceptable [71]. Introducing these technicalities would result in a more robust model that would give a more realistic representation of the desired well path.

### **6.2.4 Using 4D seismic**

4D seismic refers to a technique used in the field of geophysics and petroleum exploration to monitor and visualize changes in subsurface over time instead of as a static snapshot like in a 2D or 3D seismic. Incorporating this into the future model would make well planning better as future subsurface changes could be predicted based on the 4D seismic. A well could be planned over a period to avoid future subsurface changes affecting the planned well.

# References

- [1] Ahsan Waqar, Idris Othman, Nasir Shafiq, and Muhammad Shoaib Mansoor. Applications of ai in oil and gas projects towards sustainable development: a systematic literature review. *Artificial Intelligence Review*, pages 1–28, 2023.
- [2] Parth Solanki, Dhruv Baldaniya, Dhruvikkumar Jogani, Bhavesh Chaudhary, Manan Shah, and Ameya Kshirsagar. Artificial intelligence: New age of transformation in petroleum upstream. *Petroleum Research*, 7(1): 106–114, 2022.
- [3] Saurabh Tewari, Umakant Dhar Dwivedi, and Susham Biswas. Intelligent drilling of oil and gas wells using response surface methodology and artificial bee colony. *Sustainability*, 13(4):1664, 2021.
- [4] Maciej Z. Lukawski, Brian J. Anderson, Chad Augustine, Louis E. Capuano, Koenraad F. Beckers, Bill Livesay, and Jefferson W. Tester. Cost analysis of oil, gas, and geothermal well drilling. *Journal of Petroleum Science and Engineering*, 118:1–14, 2014.
- [5] Trent Jacobs. “Simple” Well Spacing Calculations are Inaccurate and Costly. *Journal of Petroleum Technology*, 71(11):33–35, 11 2019.
- [6] Amin Atashnezhad, David A Wood, Ali Fereidounpour, and Rasoul Khosravanian. Designing and optimizing deviated wellbore trajectories using novel particle swarm algorithms. *Journal of Natural Gas Science and Engineering*, 21:1184–1204, 2014.
- [7] Zhichuan Guan, Tinggen Chen, and Hualin Liao. *Theory and technology of drilling engineering*. Springer, 2021.
- [8] Anton Epikhin, Vitaly Zhironkin, and Michal Cehlar. Prospects for the use of technology of rotary steerable systems for the directional drilling. In *E3S Web of Conferences*, volume 174, page 01022. EDP Sciences, 2020.
- [9] Adams Neals J. and Charrier Tommie. *Drilling Engineering: A complete well planning approach*. Tulsa, Okla: PennWell Pub. Co, 1985.
- [10] Jr. Sampaio, Jorge H. B. Designing Three-Dimensional Directional Well Trajectories Using Bézier Curves. *Journal of Energy Resources Technology*, 139(3), 10 2016.
- [11] Tuna Eren and Vural Sander Suicmez. Directional drilling positioning calculations. *Journal of Natural Gas Science and Engineering*, 73:103081, 2020.
- [12] Jr. Sampaio, Jorge H. B. Planning 3D Well Trajectories Using Cubic Functions. *Journal of Energy Resources Technology*, 128(4):257–267, 04 2006.
- [13] Haoge Liu, Tor Berge Gjersvik, and Audun Faanes. Subsea field layout optimization (part i)—directional well trajectory planning based on 3d dubins curve. *Journal of Petroleum Science and Engineering*, 208:109450, 2022.
- [14] OLIASOFT. Well trajectory feature of oliasoft welldesign. Online; accessed 24-Aug-2023. URL <https://www.oliasoft.com/oliasoft-welldesign/trajectory-design/>.
- [15] FutureOn. Wellassist powered by oliasoft. Online; accessed 24-Aug-2023. URL <https://www.futureon.com/integrations/wellassist/>.

- [16] INC Dynamic Graphics. Advanced directional well planning/survey management. Online; accessed 24-Aug-2023. URL <https://www.dgi.com/well-planning-and-survey-management/>.
- [17] Halliburton. Compass software. Online; accessed 24-Aug-2023. URL <https://www.halliburton.com/en/software/decisionspace-365-enterprise/decisionspace-365-well-construction/well-construction-suite/compass-software>.
- [18] Petrel exploration & production software platform. Online; accessed 27-June-2023. URL <https://www.software.slb.com/products/petrel>.
- [19] Richard E Korf. Artificial intelligence search algorithms, 1999.
- [20] EM El-M Shokir, MK Emera, SM Eid, and AW Wally. A new optimization model for 3d well design. *Oil & gas science and technology*, 59(3):255–266, 2004.
- [21] Vahid Mansouri, Rassoul Khosravanian, David A Wood, and Bernt S Aadnoy. 3-d well path design using a multi objective genetic algorithm. *Journal of natural gas science and engineering*, 27:219–235, 2015.
- [22] Rassoul Khosravanian, Vahid Mansouri, David A Wood, and Masood Reza Alipour. A comparative study of several metaheuristic algorithms for optimizing complex 3-d well-path designs. *Journal of Petroleum Exploration and Production Technology*, 8:1487–1503, 2018.
- [23] Anirbid Sircar, Kriti Yadav, Kamakshi Rayavarapu, Namrata Bist, and Hemangi Oza. Application of machine learning and artificial intelligence in oil and gas industry. *Petroleum Research*, 6(4):379–391, 2021.
- [24] Taeho Jo. Machine learning foundations. *Supervised, Unsupervised, and Advanced Learning*. Cham: Springer International Publishing, 2021.
- [25] Vladimir Nasteski. An overview of the supervised machine learning methods. *Horizons. b*, 4:51–62, 2017.
- [26] Daniel R Schrider and Andrew D Kern. Supervised machine learning for population genetics: A new paradigm. *Trends in Genetics*, 34(4):301–312, 2018.
- [27] Kayleigh K Hyde, Marlena N Novack, Nicholas LaHaye, Chelsea Parlett-Pelleriti, Raymond Anden, Dennis R Dixon, and Erik Linstead. Applications of supervised machine learning in autism spectrum disorder research: a review. *Review Journal of Autism and Developmental Disorders*, 6:128–146, 2019.
- [28] Gabriel Mauricio Martínez Toro, Dewar Willmer Rico Bautista, Efrén Romero Riaño, and Paola Andrea Romero Riaño. Unsupervised learning: application to epilepsy. *Revista Colombiana de Computación*, 20(2):20–27, 2019.
- [29] Nina Janasik, Timo Honkela, and Henrik Bruun. Text mining in qualitative research: Application of an unsupervised learning method. *Organizational Research Methods*, 12(3):436–460, 2009.
- [30] Shoaib Kamal, PS Ramaprabha, Avinash Kumar, Bikash Chandra Saha, M Lakshminarayana, S Sanal Kumar, Anitha Gopalan, and Kuma Gowwomsa Erko. Optimization of solar panel deployment using machine learning. *International Journal of Photoenergy*, 2022, 2022.
- [31] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39:103–134, 2000.
- [32] Yan Leng, Chengli Sun, Xinyan Xu, Qi Yuan, Shuning Xing, Honglin Wan, Jingjing Wang, and Dengwang Li. Employing unlabeled data to improve the classification performance of svm, and its application in audio event classification. *Knowledge-based systems*, 98:117–129, 2016.
- [33] Ouafae El Aissaoui, Yasser El Alami El Madani, Lahcen OUGHDIR, and Youssouf EL ALLIOUI. Combining supervised and unsupervised machine learning algorithms to predict the learners’ learning styles. *Procedia Computer Science*, 148:87–96, 2019. THE SECOND INTERNATIONAL CONFERENCE ON INTELLIGENT COMPUTING IN DATA SCIENCES, ICDS2018.

- [34] Fangyu Wu, Shiyang Yan, Jeremy S. Smith, and Bailing Zhang. Vehicle re-identification in still images: Application of semi-supervised learning and re-ranking. *Signal Processing: Image Communication*, 76: 261–271, 2019.
- [35] Justin Weltz, Alex Volfovsky, and Eric B Laber. Reinforcement learning methods in public health. *Clinical therapeutics*, 44(1):139–154, 2022.
- [36] Chan-Yun Yang, Chamani Shiranthika, Chung-Yih Wang, Kuo-Wei Chen, and Sagara Sumathipala. Reinforcement learning strategies in cancer chemotherapy treatment: A review. *Computer Methods and Programs in Biomedicine*, page 107280, 2022.
- [37] Siddharth Nath, Edward Korot, Dun Jack Fu, Gongyu Zhang, Kapil Mishra, Aaron Y Lee, and Pearse A Keane. Reinforcement learning in ophthalmology: potential applications and challenges to implementation. *The Lancet Digital Health*, 2022.
- [38] Xudong Zhu, Fan Zhang, and Hui Li. Swarm deep reinforcement learning for robotic manipulation. *Procedia Computer Science*, 198:472–479, 2022.
- [39] Chengxi Li, Pai Zheng, Yue Yin, Yat Ming Pang, and Shengzeng Huo. An ar-assisted deep reinforcement learning-based approach towards mutual-cognitive safe human-robot interaction. *Robotics and Computer-Integrated Manufacturing*, 80:102471, 2023.
- [40] Íñigo Elguea-Aguinaco, Antonio Serrano-Muñoz, Dimitrios Chrysostomou, Ibai Inziarte-Hidalgo, Simon Bøgh, and Nestor Arana-Arexolaleiba. A review on reinforcement learning for contact-rich robotic manipulation tasks. *Robotics and Computer-Integrated Manufacturing*, 81:102517, 2023.
- [41] S Alagumuthukrishnan, S Deepajothi, Rajasekar Vani, and S Velliangiri. Reliable and efficient lane changing behaviour for connected autonomous vehicle through deep reinforcement learning. *Procedia Computer Science*, 218:1112–1121, 2023.
- [42] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, and Jong Wook Kim. Q-learning algorithms: A comprehensive classification and applications. *IEEE access*, 7:133653–133667, 2019.
- [43] Kaveh Madani and Milad Hooshyar. A game theory–reinforcement learning (gt–rl) method to develop optimal operation policies for multi-operator reservoir systems. *Journal of Hydrology*, 519:732–742, 2014.
- [44] José Del R Millán, Daniele Posenato, and Eric Dedieu. Continuous-action q-learning. *Machine Learning*, 49:247–265, 2002.
- [45] Esteban Real, Chen Liang, David So, and Quoc Le. Automl-zero: Evolving machine learning algorithms from scratch. In *International Conference on Machine Learning*, pages 8007–8019. PMLR, 2020.
- [46] Yuval Heffetz, Roman Vainshtein, Gilad Katz, and Lior Rokach. Deepline: Automl tool for pipelines generation using deep reinforcement learning and hierarchical actions filtering. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2103–2113, 2020.
- [47] Zhihe Zhao, Kai Wang, Neiwen Ling, and Guoliang Xing. Edgtml: An automl framework for real-time deep learning on the edge. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, IoTDI '21, page 133–144, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383547.
- [48] Brage S Kristoffersen, Mathias C Bellout, Thiago L Silva, and Carl F Berg. An automatic well planner for complex well trajectories. *Mathematical Geosciences*, 53(8):1881–1905, 2021.
- [49] Samiul Ehsan Chowdhury. Investigation and study on reinforcement learning for optimizing well path, 2021. URL <https://hdl.handle.net/11250/2788790>.
- [50] Samiul Ehsan Chowdhury, Jie Cao, Tomasz Wiktorski, and Dan Sui. Well path design using q-learning algorithms and bezier curves with obstacles avoidance. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 85956, page V010T11A007. American Society of Mechanical Engineers, 2022.

- [51] Md Fazlul Haque. Path design and optimization with obstacle avoidance via reinforcement learning, 2022. URL <https://hdl.handle.net/11250/3024569>.
- [52] Yingwei Yu, Wei Chen, Qihua Liu, Minh Chau, Velizar Vesselinov, and Richard Meehan. Training an automated directional drilling agent with deep reinforcement learning in a simulated environment. In *SPE/IADC International Drilling Conference and Exhibition*. OnePetro, 2021.
- [53] SLB. Osdu data platform. Online; accessed 05-Aug-2023. URL <https://www.software.slb.com/data/guide-to-osdu-data-platform>.
- [54] Dataiku. Dataiku key capabilities. Online; accessed 10-May-2023. URL <https://www.dataiku.com/product/key-capabilities/>.
- [55] Xiao Xiao, Tung X Trinh, and Tae-Hyun Yoon. Improved performance of nanotoxicity prediction models using automated machine learning.
- [56] MZ Naser. Machine learning for all! benchmarking automated, explainable, and coding-free platforms on civil and environmental engineering problems. *Journal of Infrastructure Intelligence and Resilience*, 2(1): 100028, 2023.
- [57] Victor Aarre and Jimmy Kilinger. Solving the challenge of seismic data management—from compression to open-sourcing. Online; accessed 10-May-2023. URL <https://www.software.slb.com/blog/solving-the-challenge-of-seismic-data-management#>.
- [58] Gitlab link to openzgy. Online; accessed 10-May-2023. URL <https://community.opengroup.org/osdu/platform/domain-data-mgmt-services/seismic/open-zgy>.
- [59] Andrzej T Tunkiel, Tomasz Wiktorski, and Dan Sui. Drilling dataset exploration, processing and interpretation using volve field data. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 84430, page V011T11A076. American Society of Mechanical Engineers, 2020.
- [60] Erlend Magnus Viggen, Erlend Hårstad, and Jørgen Kvalsvik. Getting started with acoustic well log data using the dlisio python library on the volve data village dataset. In *Proceedings of the 43rd Scandinavian Symposium on Physical Acoustics*. Norsk Fysisk Selskap, 2020.
- [61] Disclosing all volve data. Online; accessed 03-July-2023. URL <https://www.equinor.com/news/archive/14jun2018-disclosing-volve-data>.
- [62] Attila Nagy. Availability and quality of drilling data in the volve dataset. Master’s thesis, University of Stavanger, Norway, 2019.
- [63] Andrzej T Tunkiel, Dan Sui, and Tomasz Wiktorski. Reference dataset for rate of penetration benchmarking. *Journal of Petroleum Science and Engineering*, 196:108069, 2021.
- [64] Nikolay O Nikitin, Ilia Revin, Alexander Hvatov, Pavel Vychuzhanin, and Anna V Kalyuzhnaya. Hybrid and automated machine learning approaches for oil fields development: The case study of volve field, north sea. *Computers & Geosciences*, 161:105061, 2022.
- [65] Christine Ikram Noshi, Marco Risk Eissa, and Ramez Maher Abdalla. An intelligent data driven approach for production prediction. In *Offshore Technology Conference*, page D041S048R007. OTC, 2019.
- [66] Tom Inglis. *Directional drilling*, volume 2. Springer Science & Business Media, 2013.
- [67] Eric Cayeux, Jean-Michel Genevois, Stephan Crepin, and Sylvain Thibeau. Well planning quality improved using cooperation between drilling and geosciences. In *SPE Annual Technical Conference and Exhibition*. OnePetro, 2001.
- [68] Walt Aldred, Dick Plumb, Ian Bradford, John Cook, Vidhya Gholkar, Liam Cousins, Reginald Minton, John Fuller, Shuja Goraya, and Dean Tucker. Managing drilling risk. *Oilfield review*, 11(2):2–19, 1999.
- [69] Abdul-Razzaq Aqrabi, Moses Tantuoqir Maalidefaa, Jarle Haukås, Gideon Ofosu-Budu, and Dan Sui. Reinforced q-learning for automation in well trajectory design and drill planning. *AAPG ICE*, 2023.

- [70] ICE. Aapg international conference exhibition. AAPG ICE. URL <https://madrid2023.iceevent.org/>.
- [71] Mohamed Halafawi and Lazar Avram. Wellbore trajectory optimization for horizontal wells: the plan versus the reality. *J Oil Gas Petrochem Sci*, 2(1):49–54, 2019.
- [72] Gideon Ofosu-Budu. Thesis code, 2023. Online; accessed 30-Aug-2023. URL <https://github.com/GideonOB23/well-path-planning/tree/main>.



# Appendices



# Appendix A

## GitHub Link to Python Code

The whole python code for the model can be found on [\[72\]](#)

### A.1 Adding Configuration

`Adding configuration.py` is where configuration is added

### A.2 Initializing Environment and Training

`Initializing the environment.py` initializes environment

### A.3 Terminal state

`Terminal state.py` set terminal conditions

### A.4 Search for endLoc

`Search for endLoc.py` has next action function, next location functions

### A.5 Shortest path to endLoc

`Shortest path to endLoc.py` searches for shortest path

## A.6 Training of the reinforcement algorithm

**Training of the algorithm.py** is where training begins

# Appendix B

## Plots

### B.1 Shortest path training failures

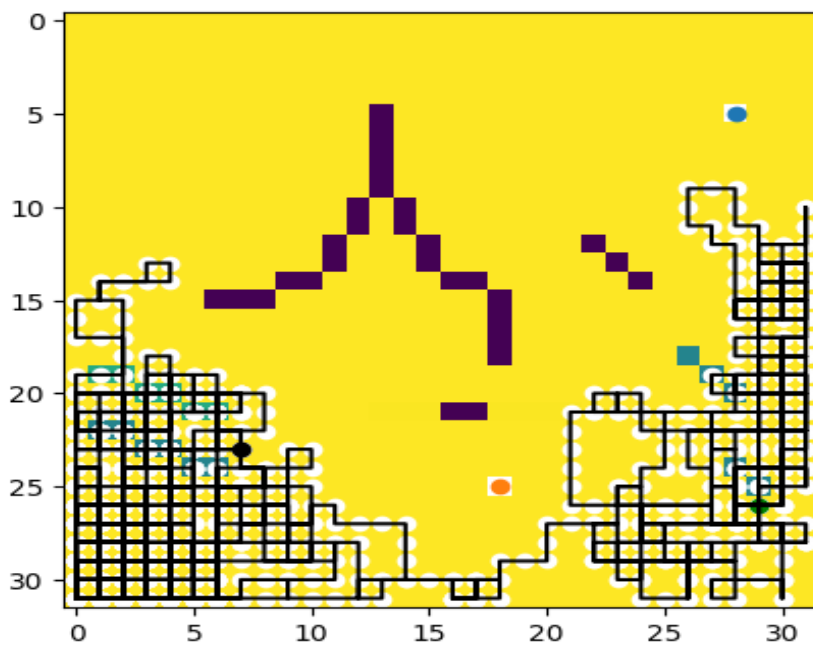


Figure B.1: *Plot 1*

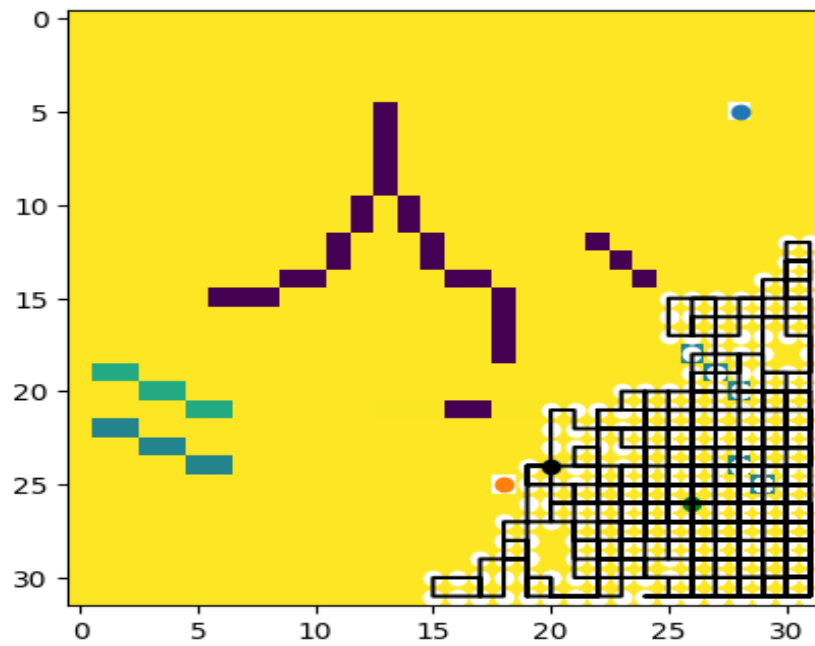


Figure B.2: Plot 2

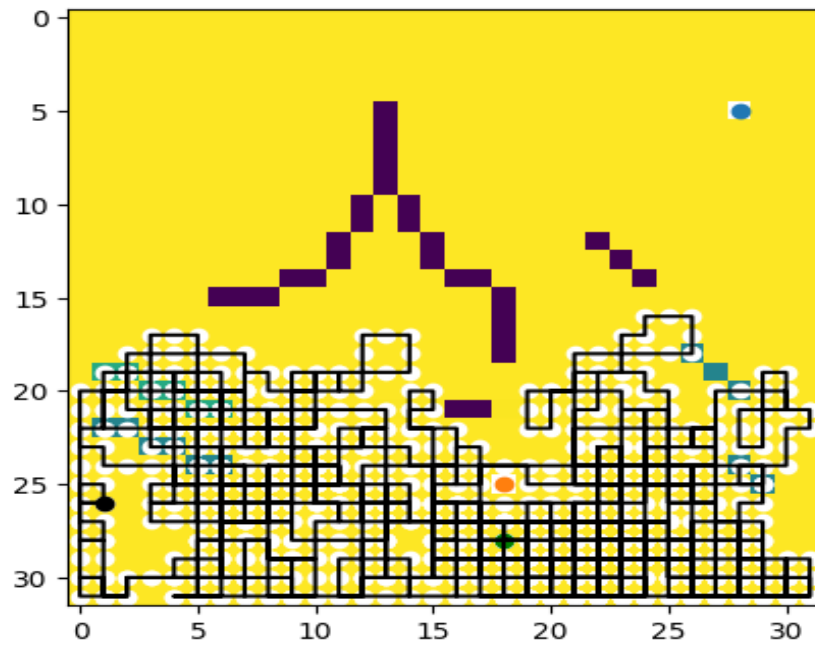


Figure B.3: Plot 3

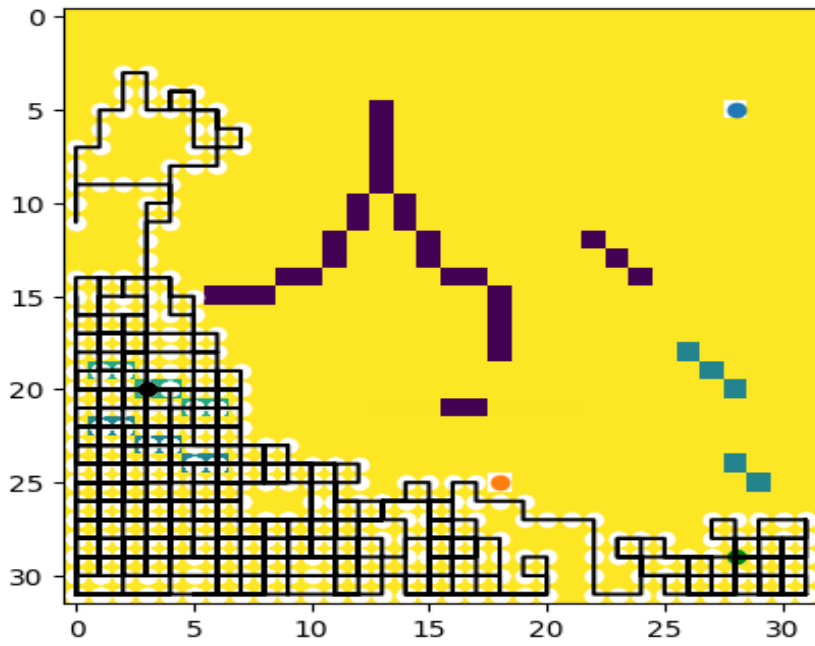


Figure B.4: Plot 4

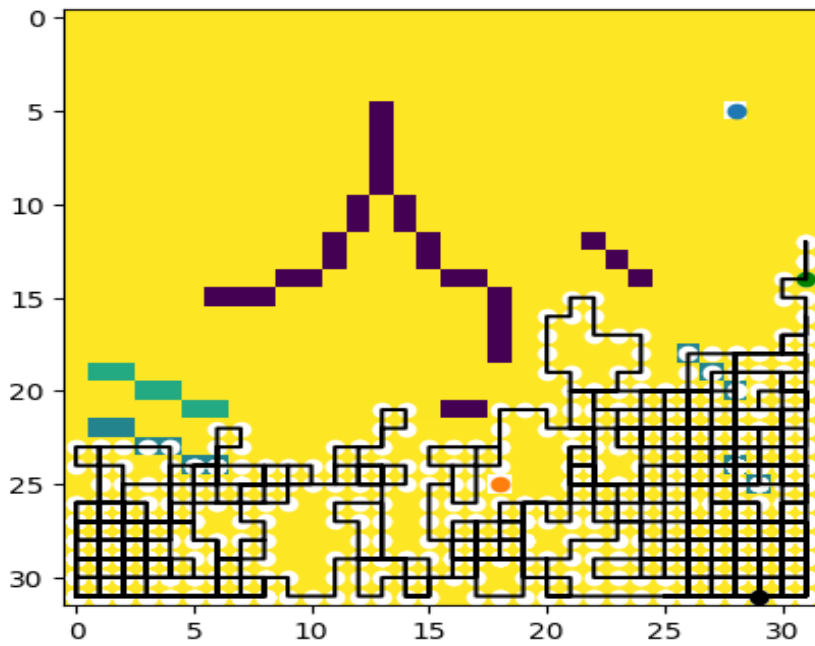


Figure B.5: Plot 4

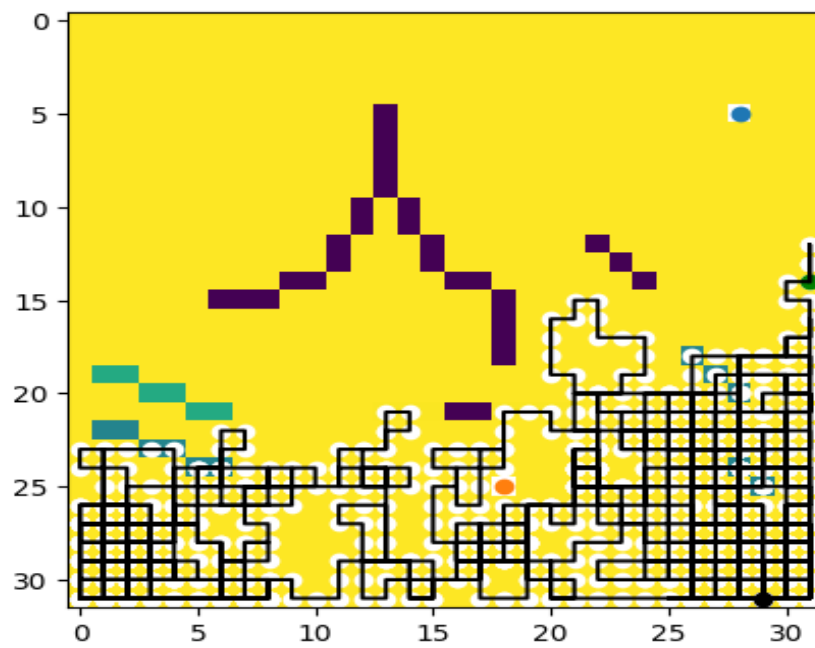


Figure B.6: Plot 5

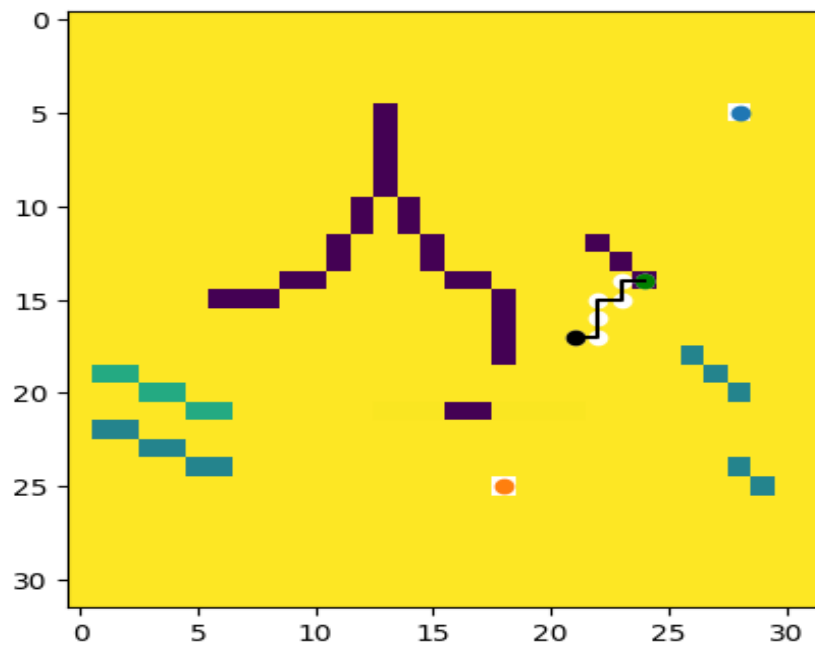


Figure B.7: Plot 6



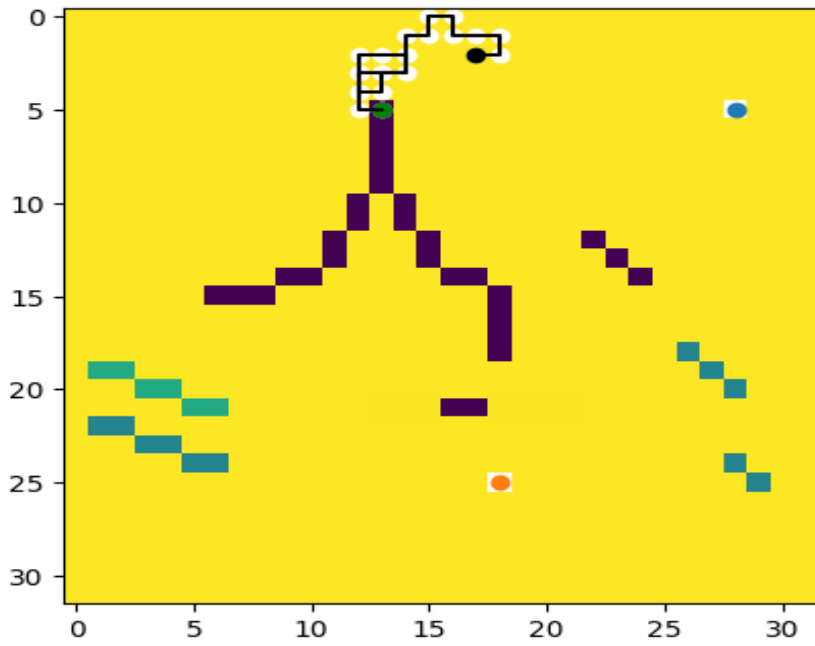


Figure B.8: Plot 7

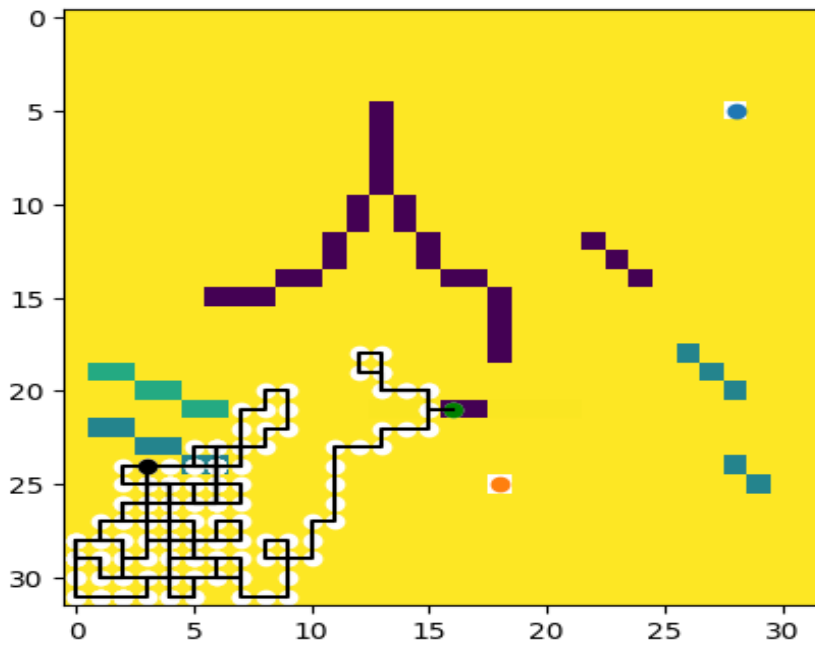


Figure B.9: Plot 8

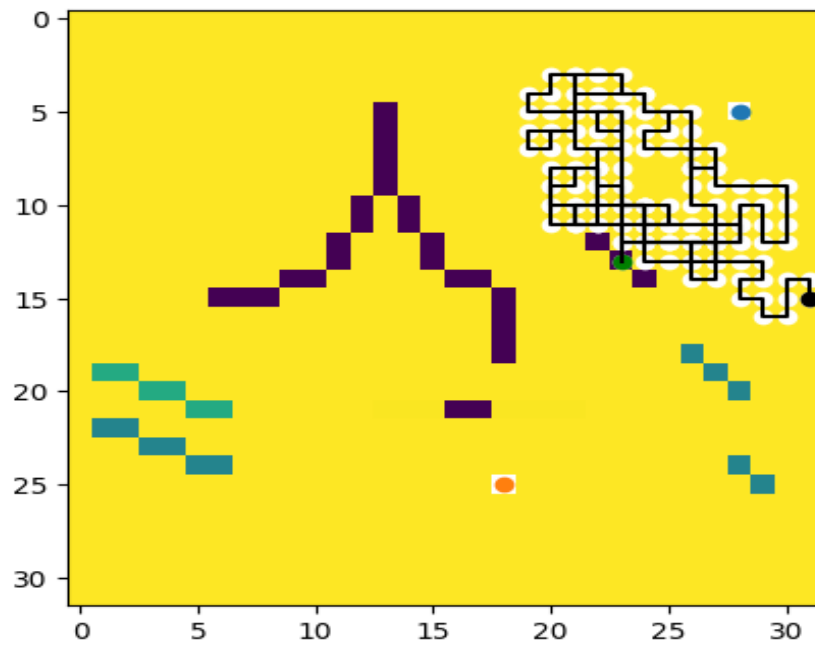


Figure B.10: Plot 9

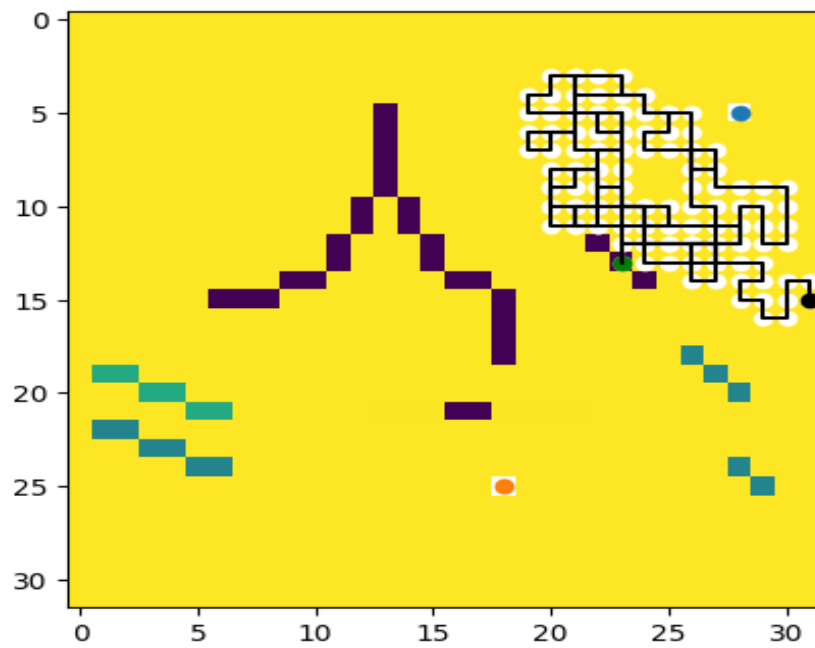


Figure B.11: Plot 10