University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialization: MSc. Computational Engineering	Spring semester, 2022 Open
Author: Md Fazlul Haque	..... Md Fazlul Haque

Programme coordinator:
Aksel Hiorth

Supervisors:
Prof. Dan Sui
Jie Cao

Title of master's thesis:
Path design and optimization with obstacle avoidance via reinforcement learning

Credits: 30

Keywords: Machine Learning, Reinforcement Learning, Q-Learning, Drilling.	Number of pages: 68 Stavanger, 15th July 2022

Md Fazlul Haque

Path design and optimization with obstacle avoidance via reinforcement learning

Master Thesis Project for the degree of
MSc in Computational Engineering

Stavanger, July 2022

University of Stavanger
Faculty of Science and Technology
Department of Computational Engineering

Universitetet i Stavanger

Abstract

For the last couple of decades, finding an optimized drilling path has been one of the key concerns for drilling engineers. It takes a couple of months to plan a well for a large number of people. The motive of this thesis is to find the optimal drilling path based on coordinates. To trace the optimal path, this thesis will apply the reinforcement learning algorithm in Matlab.

Another approach for this thesis is to find the shortest path by avoiding collision in a three-dimensional grid view.

Acknowledgments

First, I would like to thank almighty God, for providing me with the strength needed to complete my studies satisfactorily.

To the University of Stavanger, in particular my supervisor Prof. Dan Sui for the weekly meetings and continuous guidance. Also, to Jie Cao for the productive discussions held during the elaboration of this study.

Finally, I am thankful to my family and friends for their unconditional support, without you all, this would not have been possible.

Md Fazlul Haque

July 15, 2022

University of Stavanger

List of Abbreviations

AFSA	Artificial Fish Swarm Algorithm
IDE	Integrated Development Environment
MRST	MATLAB Reservoir Simulation Toolbox
ML	Machine Learning
AI	Artificial Intelligence
RL	Reinforcement Learning
LSTM	Long Short Term Memory
StoSAG	Stochastic Simplex Approximate Gradien
GA	Genetic Optimization Algorithm
KOP	kick-off point
DLS	dog-leg severity
ROP	Rate of Penetration
RPM	Revolutions per minute
SWOB	Surface Weight on Bit
TOB	Torque on Bit
TVD	True Vertical Depth
MSE	Mechanical Specific Energy
MOP	Multiobjective Optimization Problem
API	Application Programming Interface
WOB	Weight on Bit

Table of Contents

List of Figures

Listings

Chapter 1

Introduction

1.1 Background, Motivation and Challenge

Directional drilling is one of the most popular drilling technologies. With the increasing application of directional drilling, solving problems fetched by a complex geological environment is of remarkableness. Optimization of drilling trajectory is important to decrease the risk of drilling accidents and increase the efficiency of the drilling process in advance.[1]

Most of the previous researchers focused on the length to optimize a drilling trajectory [2],[3] and [4]. Because reducing a trajectory length can directly reduce the drilling time, thereby decreasing the drilling cost. As drilling cost is one of the major concerns for drilling companies. However, the reduced trajectory length will lead to a more complex wellbore structure, which may result in drilling accidents.

Some of the researchers used different machine learning algorithms for drilling optimization. For instance, Chiranth and Ken [5] used two different machine learning algorithms named the random forests algorithm and the metaheuristic optimization algorithm. For applying these 2 models, researchers used the random forests algorithm to train a model on each formation using half the data for training and the trained model was evaluated for prediction accuracy on the data set.

The artificial fish swarm algorithm was used to optimize the goal function, which was the smallest well length (AFSA) introduced by Zhang et al. (2019).[6] The calculations were completed using the Matlab environment. Compared to previously published data, AFSA optimization offers the best numerical results and the shortest route while also providing great stability and reliability. The algorithm has a basic structure and fast convergence, resulting in a global optimum in a short amount of time. As a result, AFSA can be utilized to determine the best drilling path. Wendi et al. (2020) [7] have shown that because of the difference between an actual trajectory and a planned trajectory, it is defined as a multi-objective optimization problem (MOP) with parameter uncertainties.

Well planning is a tedious and time-consuming job for a drilling engineer. During the recent oil price slump, drilling companies have focused on cost reductions associated with drilling and well planning. In response, a concept of smart well planning complete with automation has developed. typical time spent for designing and planning a well in Norway is 2-3 months and predicts that it could be reduced by 80 percent if more automation and machine learning are involved in the process.[8]

1.2 Objectives and Scope

Drilling path optimization is not only meaning that to find the shortest path but also finding it without extending the drilling cost. Based on this statement the main goal of this thesis is to find the optimized path optimally in addition to avoiding collisions. After investigating previous and recent papers it is proven that machine learning is a proven method to complement existing optimization techniques and the reinforcement learning algorithm performs better to simulate path optimization than the other models. So, this thesis will use the technique from the reinforcement learning algorithm to give a user to interact with the model and change the coordinates together with start-end points to check the different behavior when drilling. Finally, will be made a model based on this thesis for industry-level where it includes proper documentation, and validation and considers the costing.

1.3 Methodology

The base of this study is coding and for such purpose Matlab [9] will be the application or IDE of choice as it is user-friendly, handles the selected programming language Matlab, and allows to set an appropriate environment to develop the study. Matlab has gained a lot of adepts during these last years, as it is easy to learn and apply since previous programming experience is not that required. Several packages were used to set the proper programming environment, the list of them is located in Appendix A (6.4).

To solve one of the most important objectives of this study, a Matlab [9] code-based toolbox is used which is called MRST (MATLAB Reservoir Simulation Toolbox) [10].To generate the environment, the MRST toolbox helps a lot and gives some of the built-in functionalities for 3-dimensional visualization. Based on this, the thesis used a dynamic environment where the user can change the start and the endpoints as well as the number of grids in each direction.

For constituting the environment, this thesis used some existing well data to generate the hard and soft constraints. Each hard constraint is surrounded by soft constraints in six directions. The hard obstacles or constraints are defined using a specific color which is dark blue for all the existing wells and the soft constraints are defined using light blue color.

The final generated path avoids these hard constraints and produce an optimized path.

Chapter 2

Background

This is an important chapter of this report, this chapter line up to provide a background to the work. It starts with short summaries of technical and theoretical information that is relevant to this thesis.

The following section describes the terms such as Machine Learning, Reinforcement Learning, Q-learning, Drilling Technique, and trajectory optimization are discussed, as well as more advanced terms such as Grid-world, 3D environment, and MRST toolbox.

It is also important to know about the currently available approaches as well as the open source and commercial software applications which will be discussed in the last section of this chapter.

2.1 Technical Background

As stated before, this section target to provide a little summary of the technical and theoretical knowledge as well as some related works that are required to understand the work that has been done in this thesis.

2.1.1 Machine Learning (ML)

"Machine learning" and "Reinforcement Learning" is fundamental to this thesis. "Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed" [11]. Expe-

rience can be some kind of input data (text, excel tables, images, sound clips, videos, etc). With Machine Learning a computer can find patterns in the data to make some kind of predictions that would normally be very complex for a human being [12].

2.1.2 Reinforcement Learning (RL)

Reinforcement learning (RL) is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning[13].

Reinforcement learning differs from supervised learning in not needing labeled input/output pairs to be presented, and in not needing sub-optimal actions to be explicitly corrected. Instead, the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge)[13].

2.1.3 Q Learning

Q-learning is an off-policy reinforcement learning algorithm that seeks to find the best action to take given the current state. It's considered off-policy because the q-learning function learns from actions that are outside the current policy, like taking random actions, and therefore a policy is not needed. More specifically, Q-learning seeks to learn a policy that maximizes the total reward[14].

The 'Q' in Q-learning stands for quality. Quality in this case represents how useful a given action is in gaining some future reward.

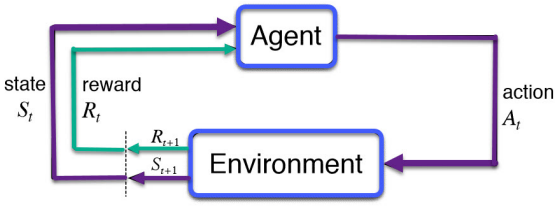


Figure 2.1: Basic components of Q-learning

The environment responds to the agent's actions by generating new observations, O(t), and

scalar reward signals, R. (t). The subsequent action is determined by history, which is defined as the sequence of observations, actions, and reward signals at time t.

States are the information used to determine what happens next. There are three main types of states- environment state (S_t^e), agent state (S_t^a), and the information/Markov state (S_t).

$$\sum_{t=0}^{t=\infty} \gamma^t r(x(t), a(t))$$

Figure 2.2: Cost/reward calculation function of Q-learning

The reward function r is sum-mated over t (the time steps) which means the objective function calculates all potential rewards that can be attained through the game. x represents the state at any given time step (denoted by t), and r represents the reward function for x and a.[15]

2.1.4 Grid World and 3D Environment

"Grid World" is the primary concept of this thesis and it carries the whole working environment. Grid World, a two-dimensional plane, is one of the easiest and simplest environments to test reinforcement learning algorithms. In this environment, agents can only move up, down, left, and right in the grid, and there are traps in some tiles. The agent starts at the fixed start position and when it arrives at the goal or trap, the episode ends.[16]

But 3D Environment is different and agents can move in six directions, up, down, left, right, and front, back. This is possible using the MRST toolbox in Matlab [10] which is used in this thesis. More about the MRST toolbox is discussed in the following section.

2.1.5 MRST toolbox in Matlab

The MATLAB Reservoir Simulation Toolbox (MRST) is a free open-source software for reservoir modelling and simulation, developed primarily by the Computational Geosciences group in the Department of Mathematics and Cybernetics at SINTEF Digital. The software has a large

international user base and also includes third-party modules developed by researchers from Heriot-Watt University, NTNU, University of Bergen, TNO, and TU Delft. [10]

2.2 Current Ideas

This section discusses existing approaches. This section has mainly three different parts.

1. Discuss some existing methods that are used to well path optimization

2. Some open source solutions and

3. Some existing commercial solutions.

There have been many attempts to incorporate machine learning with drilling engineering to reduce time and design costs. Reinforcement learning has been used as the algorithm can learn and perform based on previous experiences on its own. Some of the works related to this thesis are discussed below.

One of the current existing ideas is “Gradient based Well Trajectory Optimization”. A method to automate the process of optimizing the trajectory of production well was first proposed by Vlemmix et al.[17]. Motivated by the advancements in the adjoint-based well location optimization, the authors extended the method of Handels et al. [18] to determine an optimal well trajectory in a three-dimensional reservoir model. Since optimal well trajectory is crucial to avoid gas cusping and water coning, the authors verified the method on a thin oil rim reservoir with a relatively large gas cap and aquifer. The Vlemmix et al.[17] method is based on surrounding each trajectory point with ‘pseudo-sidetracks’. These fictional ‘pseudo sidetracks’ have very small perforations and thus production rates. The reason that this approach was chosen over placing vertical pseudo wells in each grid-block is that the effect of the sidetracks on the total well behavior including lift and well bore friction can be considered. The gradients of the objective function with respect to a dimensionless multiplication factor for perforations (interpreted as a ‘pseudo valve representation’ of an Inflow Control Valve) are used to find the improving directions for the trajectory points (Chowdhury [19]).

2.3 Open Source Solutions

This section introduces some of the open source web applications those are really helpful for drilling engineers to design and analysis on drilling data.

2.3.1 OpenLab App

OpenLab is developed and managed by the Drilling and Well Modeling group of NORCE Energy in collaboration with the University of Stavanger. They seek to offer a world-leading simulation environment within drilling and well technology for education, research, and innovation.[20]

A simulation in the Web Enabled Drilling Simulator is based on NORCE's computer models of well flow and drillstring mechanics. Although the computer models behind the simulator are among the world's most detailed models of their kind, they have created a web environment where you can easily run simulations and handle the results.

The simulator can be run interactively through a web browser, or also programmatically through our Matlab or Python packages..

2.3.2 Drilling Data Web Application

This is a web application where users can visualize different existing wells based on different parameters. Also, it is possible to compare and do an analysis of existing well data based on depth and time. The most important features are Well Configuration and Drilling Geology. The well configuration section shows the different well configurations as well as the whole section, well path design and drilling fluid, and drilling string in formations. On the other hand, the drilling geology section visualizes the geopressure and geothermal in a graph [21]. Note that this web application is still under construction.

2.4 Commercial Solutions

In contemplation of minimize the drilling challenges and decreased the costs and time associated with completion of wells. There are some others commercial solutions available in market

who are doing extra ordinary work in drilling industry. Some of this commercial software and its utilities will be presented in this section.

2.4.1 eDrilling Softwares

eDrilling AS is a well known world-leading supplier of AI, machine learning, and predictive analytical solutions to the oil and gas industry. The company has created a Life Cycle Drilling Simulation concept, diagnostics technology merged with a 3D visualization into a 'virtual well-bore' and advanced dynamic drilling models. All the models can interact with one another and be used in the whole drilling value chain from design and planning (wellPlanner) through scenario developments and training (wellSim) and then real-time operations/monitoring (wellA-head, wellBalance) and finally experience transfer and post-analysis.[22]

WellBalance

The wellBalance software improves any MPD control system to keep a better constant bottom hole pressure during MPD operations and perform planning with an offline model. It provides with real-time set points to the MPD control system, based on a dynamic real-time simulation calibrated against downhole measurements. The software is complemented with an offline sim-ulation tool, the wellBalance™ Offline, used to test and analyze operations and procedures.[22]

wellPlanner

wellPlanner Uses dynamic simulations to address the industry's need to improve safety margins and reduce and rid themselves of risk, as well as the need to drastically reduce well planning time.[22]

The core features are:

1. Dynamic models verified in wellAhead, wellPlanner and wellSim by operators
2. Unique combination of models and functionalities
3. All combined in one Integrated Drilling Simulator (IDS)named Intellectus
4. All flow related dynamic functionalities and responses combined in one IDS

5. Integrated part of Life Cycle Simulations with wellAhead and wellSim products

6. Seamless link to wellAhead, wellSim Interact and wellSim hiDRILL

Benefits of wellPlanner are increased drilling productivity, improved well planning accuracy,reduced drilling risks and uncertainty,improved drilling safety and quick well planning. [23]

WellSim

The wellSim™ is the software product family for engineering and training of all disciplines by improving insight and understanding of the dynamic well behavior; it has the potential to change the ways of planning and drilling several complex wells. [22]

The core features are:

1. Interactive ROP model

2. Coupled Flow and Torque/Drag model

3. 3D Visualization with risk/info messages, proximity wells, formations, etc.

4. Malfunctions, kicks and losses in scenarios

5. Model state handling enables transfer of simulation states between the eDrilling applications (“move state and data from drilling operation to onshore simulator to explore alternatives”)

6. Cross-application configuration enables transfer of virtual wells between the eDrilling applications

7. Simulation speed up to 10 times real time

This software utilized an advanced downhole simulator, Intellectus, including a dynamic ROP model and coupled Flow and T&D model. [24]

2.4.2 Schlumberger Softwares

Schlumberger has created a cloud-based environment called DELFI to compliment their large bank of data and different software solutions for the whole of the petroleum chain. It harnesses data, domain expertise, and scientific knowledge changing the way to perform in every part of the E&P value chain. The DELFI environment creates workflows and applications attainable to every single user. It provides members with access to build shared workspaces for models, interpretations, and data while respecting proprietary information boundaries.

The DELFI environment puts the full scope of available cognitive technologies to work from AI to analytics. Robust cognitive systems recognize each user to deliver a uniquely personalized experience. Intelligently searching, proactively learning, and automating tasks, enabling the user to predict, prioritize, and advise. The new data is automatically shared between jobs across the DELFI environment; each live project is dynamically optimized with the latest information.

Apart from the DELFI environment, the company also provides software for drilling design,such as the DrillPlan and the DrillBench software.[25]

DrillPlan

DrillPlan has the most exclusive fearures like automated engineering and design as well as design validation. The DrillPlan solution is a digital well construction planning solution that maximizes the results from shared teams by giving them access to all the data and science they need in a single, common system. The automation of repetitive tasks and validation workflows, enables better quality drilling programs to be produced quickly, and ensures entire plan is coherent.

DrillPlan solution includes circular workflows, plans are improved as new data is added—future programs learn from the experience of all wells planned before.

Designed for the cloud and accessible in the DELFI cognitive E&P environment, the DrillPlan solution provides easy access to all of your well construction projects.[26]

DrillBench

The DrillBench Dynamic Drilling Simulation Software is a Schlumberger software that provides the user with dynamic simulations for pressure control, blow out control, managed and underbalanced operations, and well control.

The company's recent enhancements on this software include the integration of well paths directly from the Petrel platform, being able to read pore and fracture pressure from the Techlog platform, the support for dual gradient drilling with improved modeling of managed pressure drilling, the strengthened dynamic S&S calculations, subsea pump, and the rig site Kick, which is a standardized kick sheet for operational rig site use, based on robust well control modeling and simulation. [27]

2.4.3 Halliburton Softwares

Halliburton is another famous company in the oil and gas industry, with a large bank of data and different software solutions for the whole of the petroleum chain. The company provides E&P professionals with a software-driven life-cycle named Halliburton Landmark Solutions. Landmark Solutions is a hybrid cloud environment with seamless connectivity that uses a digital twin technology to provide the user with a faster, more open, and collaborative open industry platform. The following two are the central systems in landmark solutions

- the WellPlan software and
- the DecisionSpace® 365.

WellPlan software

The WellPlan software is the latest evolution in Halliburton's well construction information solutions. It is integrated with EDT and EDM applications, providing a complete well engineering software tool kit capable of designing complex well string operations and navigate different challenges found in the well construction.[28]

DecisionSpace® 365

The DecisionSpace® 365 is a cloud-based subscription service for E&P applications found on OSDU that provides high throughput, low latency, and self-cleaning solutions to large quantities of data from various sources into the OSDU. Once the data is loaded, the DecisionSpace® 365 cloud provides modular, open, and plug-and-play solutions with an intelligent workflow to provide efficiency and insight.[29]

Chapter 3

Solution Approach

This chapter explores works that are related and gives an overall idea of what is done in this thesis to build a background and help understand the work done before this thesis was even possible. At the end of this chapter, it discusses the approach that has been used in this thesis.

3.1 Trajectory optimization

Mansouri et al. [30] suggest a trajectory optimization model based on the genetic optimization algorithm (GA) with an objective function that maximizes the minimum separation factor along the wellbore. The algorithm seeks the least distance between the planned well and the reference well in three dimensions. They applied the minimum separation factor constraint of 1.5. GA provides several random solutions, which are constrained in their study by KOP (kick-off point), DLS (dog-leg severity), inclination, and azimuth. The interval for calculations was selected to be 30 m. Busby et al. [31] point out that "manual optimization" of wellbore placement is a formidable task due to many possible solutions, uncertainties in handling, and complex wellbore trajectories. Therefore, they propose a methodology for well placement based on the experiences organized as machine learning regression models using simulated data. In their model, geological parameters, including uncertainties and trajectory parameters, were used as inputs. Lu et al. [32] develop a bi-objective optimization technique to increase production based on StoSAG (stochastic simplex approximate gradient). Two primary objectives of the defined method are 1) search for optimal wellbore trajectory (including anti-collision) and control set-

tings of injectors and producers to maximize the production and 2) minimize the risk related to achieving low net present value (NPV). The authors propose an iterative simultaneous procedure, where trajectory and control settings are updated for each iteration because it outperforms more common sequential optimization as claimed in that paper. Yeten et al. [33] used GA in conjunction with an artificial neural network, a hill-climber, and a near-well up-scaling technique to determine the optimum well location and trajectory for non-conventional (multilateral) wells. Due to its stochastic nature, GA requires multiple iterations.

The artificial fish swarm algorithm is used to optimize the goal function, which is the smallest well length (AFSA) [6]. The calculations were completed using the Matlab environment. Compared to previously published data, AFSA optimization offers the best numerical results and the shortest route while also providing great stability and reliability. The algorithm has a basic structure and fast convergence, resulting in a global optimum in a short amount of time. As a result, AFSA can be utilized to determine the best drilling path. An article has shown that because of the difference between an actual trajectory and a planned trajectory, it is defined as a multi-objective optimization problem (MOP) with parameter uncertainties [7].

3.2 Use of RL in Well Planning

For the past few years drilling engineers have been trying to reduce time and design costs using the help of machine learning. Some of them were using reinforcement learning to learn and perform based on previous experiences on their own. In the below section this report described one of them.

3.2.1 Location Optimization of a well

In this article, the researchers put the well location selection problem as a multi-stage sequential decision. Using reinforcement learning techniques to formulate sequential well location selection as a dynamic programming problem. Within the reinforcement learning framework, geostatistical simulation techniques have been applied to characterize the uncertainty in reservoir models and determine the updates to the beliefs regarding reservoir rock properties resulting

from alternative well location decisions and hypothetical observations at those locations. Because of the number of feasible sequences of well locations, the possible observations within each sequence, and the computational demands of simulating updated reservoir properties, traditional dynamic programming solution methods are not trackable. They show the application of reinforcement learning, a class of algorithms that combine "Monte Carlo" sampling methods with functional approximations of the objective function. These methods provide a computationally tractable approach to exploring sequential well location selection with explicit consideration of the information value of initial well locations.

Using this approach, they test the hypothesis that a well location selection strategy is more robust to uncertainty in the initial data than single-stage optimization approaches. To test this hypothesis, they apply a novel reinforcement learning framework to select well locations accounting for information value. They have developed several proxy geostatistical models to reduce the computational time and effort and explore the effects of using a proxy model on the suggested well placement strategy. To find the solution to the well location problem, they used Q-learning and discuss its limitations. They build the framework on a more straightforward two-dimensional well location problem, using tabular and artificial neural networks as functional approximators. In the 2D case, they explored the sensitivity of the policy developed using reinforcement learning to the hyperparameters that control the exploration of the state-action space and convergence to the optimal policy.

Moreover, the deep reinforcement learning algorithm is also used for the Stanford V and SPE comparative solutions project model 2 reservoirs, which are more prominent three-dimensional reservoir cases. The results of the more significant reservoir cases demonstrate the benefits of sequential well location selection relative to single-stage optimization approaches. [34]

3.3 Proposed Solution

There are several common methods such as steepest ascent algorithm; conjugate gradient method; LBFGS method; Levenberg−Marquardt algorithm; Gauss−Newton method;SPSA; EnOpt; EnKF; SID−PSM; NEWUOA; QIM−AG. have been applied to sort out the trajectory optimaization as well as the different machine learning algorithms has been pit into action as well, to present

the more clear and comprehensive picture of path−finding.[19]

This thesis has been used reinforcement learning (RL) for number of reasons:

- Most of the previous thesis used machine learning algorithms those thesis used some prior data for learning and training but RL does not use any sort of prior data.
- The RL model is efficient to solve the problems when there is a need to teach an AI agent to make decisions in a complex environment.
- Training does not involve human intervention but instead allowing exploring rules based on self-play only.

Last but not the least, Two of the main gaps of the method were restriction of the trajectory movement to only two directions and restriction to optimize the well length on Vlemmix et al. (2009).[17] Additionally, the adjoint-based optimization technique may get stuck in a locally optimal solution. The method employs a drill-ability/smoothing algorithm to ensure dogleg severity is below the predefined limit. This ensures that the final well trajectory is drill-able and realistic. This approach confirmed the scope for developing an algorithm to predict the well trajectory optimization and significantly improve the overall efficiency during FDP. Also, in Chiranth and Ken (2018) [5] paper, the optimization of the ROP model led to an increase in MSE and TOB. Because of this problem, it may lead to additional costs due to not-optimal use of bit energy, excessive vibrations, and drilling dysfunction. One more knowledge gap in Chowdhury (2021) [19] is that only used 2D in python and 3D visualization in Unity3D but did not mention the coordinates. Also, the path in 3D was not clear and directional.

Chapter 4

Implementation

This chapter manifests how each part of the system is brought about. The chapter begins with the experimental procedure and the resulting architectural design. The section 4.2 explains the 3D program in Q-learning in Matlab.

This report has described the key problem area in the previous chapter 3, which primarily sheds light on the view of well trajectory optimization through the implementation of ML techniques. The main goal of this report is to find an optimal path between two points by avoiding obstacles in a three-dimensional grid view.

This report is using a 3D program in Matlab to show the effectiveness of the algorithm in this case. Up to a point, the 2D program or environment reflects a partial picture of the path tracing of the well, which does not provide an intelligible or substantial solution in the real-life scenario.[19] That's why this thesis shows a 3D overview of a well and to build more concrete picture through the application of a 3D program. This paper is not only concerned to present a more efficient path tracing but also reflecting a more transparent and comparable picture of the well trajectory by avoiding collisions.

4.1 Experimental Procedure

There are several ways to detect the shortest paths, this thesis has used several tools to exhibit the results. Sequentially, this paper has used Travelling Salesmen Problem and Ant Colony Optimization(ACO) tools before determining the results through applying reinforcement learning.

4.1.1 Environmental Overview

The two most important factors lead the whole program or experimental environment. One is Obstacles that contain hard, soft, and normal constraints. The other one is desired well path where the essential factors are the start and the endpoints. Just as importantly, the 3D environment contains 3D cubes that are generated based on different location points and those points are defined based on north, east, and TVD.

4.1.2 Obstacles in 3D space

The implementation program in Matlab uses obstacles that are similar to real-world oil paths that have been drilled in that area. For this program, the primary constraint is a soft or a hard obstacle as well as some normal constraints.

Point to grid conversion

For the existing wells, it uses the east, north, and TVD values to generate the well structure. But for this research need 3D cubes to generate the 3-dimensional environment. To achieve that goal this thesis used a point conversion process based on the MRST toolbox shown in figure 4.1.

Hard obstacle or hard constraints

The hard obstacle is analogous to the existing oil path. In the real world, we cannot drill through an existing oil path while drilling a new one. Thus, when optimizing a trajectory, avoiding previous paths is critical. In the Matlab program, the algorithm determines the optimal path

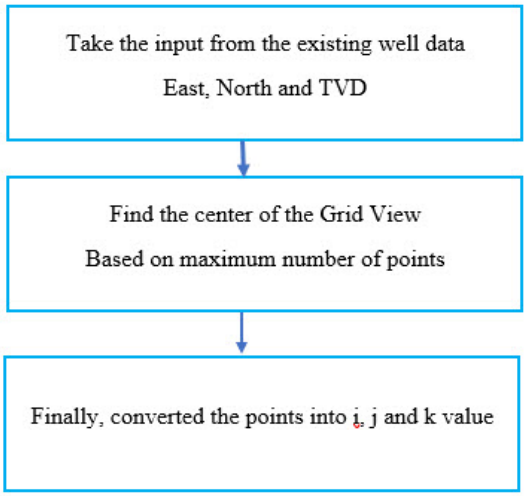


Figure 4.1: Points conversion flow

This function will always return three points as I J and K based on following conditions:

```
direction_xyz = pdims./dims;  % here pdims is the maximum number of the
points in each axis. Dims is the grids numbers in each side of the grid
environment.

centre_point = pdims./2; % Find the center

dem1 = north_val+centre_point(1); % I just change the point into center
of the grid world
dem2 = east_val+centre_point(2); % I just change the point into center of
the grid world

north_out = int32(dem1/direction_xyz(1));
east_out = int32(dem2/direction_xyz(2));
tvd_out = int32(tvd_val/direction_xyz(3));

conversion_out = [east_out,north_out,tvd_out]; % finally we have a grid
based on i,j and k.
```

Every time when we call this function it will give us "conversion_out" with 3 points in grid

Figure 4.2: Point conversion code

using cost estimation. As a result, the cost/weight of a hard obstacle is significantly greater (this thesis used -1000), preventing the new path from intersecting.

Soft Obstacle or soft constraints

The soft obstruction is analogous to the outer layer of the existing oil path. In ideal cases, drillers avoid drilling new paths adjacent to existing ones, but this can be done if there is no other option. Similarly, the cost/weight of soft obstacles is moderate (this thesis used -10),

allowing the algorithm to perform overall cost estimation and intersect with it only when the cost of avoiding is significantly greater than the cost of passing.

4.1.3 Start and End Points

The program contains two points: a start point and a target or end point. These points correspond to the positions of oil deposits on the surface and underground. Every point is generated based on east, north, and TVD value. In the Matlab program, the start point is located at the top or a specific position of the diagram and the target point is located at the bottom.The point conversion process described in 4.1.2 section.

4.1.4 Cost and Reward Calculation

Because RL determines the optimal path based on the total cost, cost estimation is a critical component of the algorithm.

Reinforcement Learning Program

Because the program is written in a grid-world environment, each grid has a cost of 10 and is called normal grid. Additionally, because the program is designed to avoid obstacles whenever possible, the cost of obstacle grids is higher than the cost of a standard grid. The cost of a hard obstacle is set to -1000 because the desired path should avoid it; additionally, a small cost of -10 has been specified for a soft obstacle. Additional obstacles can be added at a different cost.

For reaching the goal, a reward of 100000 is given.

4.1.5 Grid Index Structure

As this thesis stands for a grid view environment, the structure of the grid index is an essential factor. This section described the grid index structure. The figure 4.1 shows the point conversion process which returns only 3 points i,j, and k but this thesis used 2 more indexes. One is the index number or grid number and the other one is the reward value or the cost which shows in the 4.5 listing.

4.2 3D Program

This section discusses the three-dimensional implementation of the program. The primary objective of the 3D program was to find the optimal path while avoiding obstacles or hard constraints (existing oil paths) in a three-dimensional grid world.

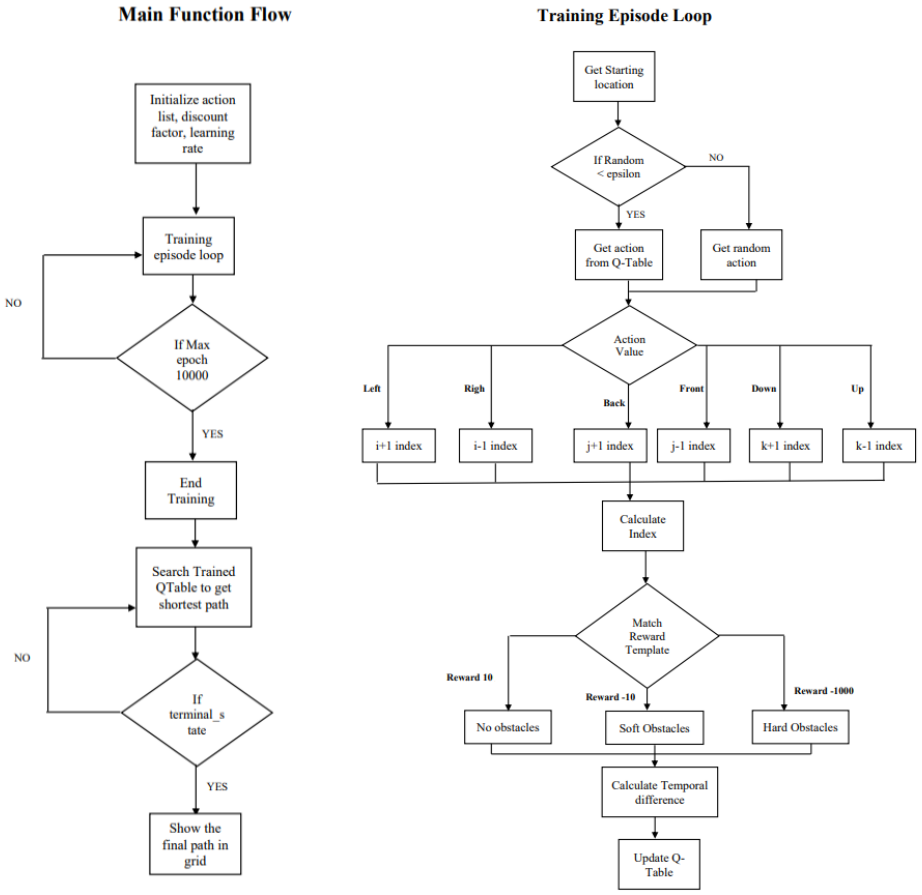


Figure 4.3: Flowchart of Q-learning program in 3D

This flow chart 4.3 figured the 3D program in Matlab. The flow chart is mainly divided into two parts of the program, where the main function flow refers to the initial conditions, and qTables are drawn. It also defines the parameters of the Greedy algorithm and passes to the training episode flow. Max epoch 10000 means maximum training episodes, after completion of the training, the algorithm finds the shortest path from the QTable, and if it reaches a terminal state, the program ends and shows the result.

In the training episode loop diagram, The algorithm gets starting and end locations for the path. If the probability factor is greater than the given random number in the training function, which starts the training procedure. Based on the condition, the movement kicks on (left, right,

up, down, front, and back). Based on the reward template, the algorithm updates the Q table and sends the information to the get shortest path function.

4.2.1 Program Configuration

The program configuration starts with the program's initial conditions, such as start and end-points for the drill path that are coming from the point conversion function. Then, obstacle coordinates those are coming from the existing well data mentioned in the code Appendix A.2 section. In the real world, a well path is composed of several layers and constraints. First and foremost, there is the existing drilled well path. This is named the hard layer or hard constraints. Consecutively, there is a protective outer layer which is denoted as a soft layer that surrounds the hard layer in six directions. Full code available in Appendix A.2.

4.2.2 Initializing the Environment and Training

There are some stop/terminal conditions mentioned below:

1. Start point can not be inside hard obstacles.
2. Crushed into hard obstacle
3. Reach End point

```
function [action_index1] = is_terminal_state(current_grid_index)
if current_grid_index(:,5) == -1000 || current_grid_index(:,5) ==
   100000
    %disp("Is terminal State true");
    action_index1 = true;
else
    action_index1 = false;
    %disp("Is terminal State false");
end
end
```

Listing 4.1: Termination condition for Q-learning program

Listing 4.2 shows how the agent enumerates many times starting from random position,trying to maximize its reward(reach start-point). It gets a random row and column index and continues choosing random row and column indexes until a non-terminal state is identified. Initially, the starting point is (-140,1320,200).

```
function [current_grid_index] = get_starting_location()
global Combine;
current_grid_index = point_conversion(-140,1320,200);
[~,id_9] = ismember(current_grid_index,Combine(:,1:3),'rows');
current_grid_index(:,4) = id_9;
disp(current_grid_index);
lidx_0 = Combine(id_9,end);
current_grid_index(:,5) = lidx_0;
disp(current_grid_index);
while is_terminal_state(current_grid_index)
    current_grid_index = Combine(randi(size(Combine,1)),:);
    disp("from while starting location");
    disp(current_grid_index);
end
end
```

Listing 4.2: Staring condition for Q-learning program

Then it chooses next action from given state with the highest probability.The epsilon is for adding some randomness to the agent.

```
function [outputArg1] = get_next_action(current_grid_index,epsilon)
global q_vals_for_6;
if rand < epsilon  && (current_grid_index(:,1))<7
    [tra,inda] = max(q_vals_for_6);
    outputArg1=inda;
else
    outputArg1 = randi(6);
   %disp(outputArg1);
```

```
end
end
```

Listing 4.3: Choose best action for Q-learning program

once the action is chosen,the agent returns the location and updates the action index after making that action.

```
function [new_grid_index] = get_next_location(current_grid_index,
    action_index)
global Combine;
new_grid_index = current_grid_index;
actions = ["up", "right", "down", "left","back","front"];
if actions(action_index) == "up" && current_grid_index(:,3) > 1
    new_grid_index = [current_grid_index(:,1),current_grid_index(:,2)
    ,current_grid_index(:,3)-1];
    [~,idx] = ismember(new_grid_index,Combine(:,1:3),'rows');
    new_grid_index(:,4) = unique(idx);
    lidx = Combine(idx,end);
    new_grid_index(:,5) = lidx;
elseif actions(action_index) == "down" && current_grid_index(:,3) <
    max(Combine(:,3))
    new_grid_index = [current_grid_index(:,1),current_grid_index(:,2)
    ,current_grid_index(:,3)+1];
    [~,idx] = ismember(new_grid_index,Combine(:,1:3),'rows');
    new_grid_index(:,4) = unique(idx);
    lidx = Combine(idx,end);
    new_grid_index(:,5) = lidx;
elseif actions(action_index) == "left" && current_grid_index(:,1) <
    max(Combine(:,3))
    new_grid_index = [current_grid_index(:,1)+1,current_grid_index
    (:,2),current_grid_index(:,3)];
    [~,idx] = ismember(new_grid_index,Combine(:,1:3),'rows');
```

```
20      new_grid_index(:,4) = unique(idx);
21      lidx = Combine(idx,end);
22      new_grid_index(:,5) = lidx;
23  elseif actions(action_index) == "right" && current_grid_index(:,1) >
        1
24      new_grid_index = [current_grid_index(:,1)-1,current_grid_index
        (:,2),current_grid_index(:,3)];
25      [~,idx] = ismember(new_grid_index,Combine(:,1:3),'rows');
26      new_grid_index(:,4) = unique(idx);
27      lidx = Combine(idx,end);
28      new_grid_index(:,5) = lidx;
29  elseif actions(action_index) == "back" && current_grid_index(:,2) <
        max(Combine(:,3))
30      new_grid_index = [current_grid_index(:,1),current_grid_index(:,2)
        +1,current_grid_index(:,3)];
31      [~,idx] = ismember(new_grid_index,Combine(:,1:3),'rows');
32      new_grid_index(:,4) = unique(idx);
33      lidx = Combine(idx,end);
34      new_grid_index(:,5) = lidx;
35  elseif actions(action_index) == "front" && current_grid_index(:,2) >
        1
36      new_grid_index = [current_grid_index(:,1),current_grid_index(:,2)
        -1,current_grid_index(:,3)];
37      [~,idx] = ismember(new_grid_index,Combine(:,1:3),'rows');
38      new_grid_index(:,4) = unique(idx);
39      lidx = Combine(idx,end);
40      new_grid_index(:,5) = lidx;
41  end
42  end
```

Listing 4.4: Choose next location for Q-learning program

The shortest path function in listing 4.5 takes start location as its parameters. The agent will immediately return if this is an invalid starting location. If the start location is valid then continue moving the path until reaches the end location. Moreover, while it is moving toward the goal, it updates the shortest path index. After the training episode is finished, it accepts the end location and returns the shortest path of both locations. The path location final array will be looked like 5.2.

```
function [shortest_path] = get_shortest_path(start_grid_index)
if is_terminal_state(start_grid_index)
    disp("The terminal state is starting index");
    return;
else
    disp("Starting index in not the terminal state");
    current_grid_index = start_grid_index;
    shortest_path =[];
    shortest_path = [shortest_path,current_grid_index];
    while ~is_terminal_state(current_grid_index)
        action_index = get_next_action(current_grid_index,0.9); % 0.4
    is the epsilon value
        current_grid_index = get_next_location(current_grid_index,
   action_index);
        shortest_path = [shortest_path,current_grid_index];
        %disp('From Shortest Path');
        disp(shortest_path);
    end % while end
end % if end
end %function end
```

Listing 4.5: Get shortest Path for Q-learning program

4.2.3 Generating 3D Environment

The core concept of this thesis is the 3D Environment and to generate this environment, this thesis used some techniques as well as some basic programming concepts which will be discussed in the following section. Firstly, it uses the MRST as a toolbox or third-party library.

```
mrstModule add ad-core ad-blackoil diagnostics wellpaths
```

Then for finding the center of the whole grid box it uses very basic programming logic.

```
pdims = [3300, 1400, 2000];
dims = [20, 20, 20];
XX = pdims./dims; % For middle point
Find_centre = pdims./2; %Used this variable to find the center
```

Finally, using the help of MRST it generates the 3D grid view which shown in below code and full workable code will be available on Appendix A.

```
G = cartGrid(dims, pdims);
G = computeGeometry(G);
pos_North = abs(North);   %Ref well data north value
a = [pos_North,East]; % combine north and east
aa = [a(:,1)+Find_centre(1),a(:,2)+Find_centre(2)];
wellpath1 = makeSingleWellpath([aa,TVD]); % Path generation
wellpath_fork = combineWellPaths({wellpath1}); % Combine into single
    well path
[cells_fork1, segInd_fork1, ~, ~, DT1] = findWellPathCells(G,
    wellpath_fork); % Determine the cells
```

Listing 4.6: Generating 3D Environment

4.2.4 Point Conversion Process

This section is the most important part of this thesis. As user only knows about the east value, north value, and the tvd values but to achieve the goal of this thesis need to covert these points into 3D grids. By using the following 4.7 code the point conversation has been done.

```
function [conversion_out] = point_conversion(east_val,north_val,
    tvd_val)
pdims = [3300, 1400, 2000]; %changed based on value
dims = [10, 10, 10]; %this can be changed based on grid
direction_xyz = pdims./dims;
centre_point = pdims./2;
disp(north_val+centre_point);
dem1 = north_val+centre_point(1);
dem2 = east_val+centre_point(2);
north_out = int32(dem1/direction_xyz(1));
east_out = int32(dem2/direction_xyz(2));
tvd_out = int32(tvd_val/direction_xyz(3));
conversion_out = [east_out,north_out,tvd_out];
%disp(conversion_out);
end
```

Listing 4.7: Point conversion in Matlab

4.2.5 3D Visualization

MRST Matlab toolbox allows to plot 3 dimensional grid using the plotGrid function. PlotGrid is an essential part of MRST's visualization routines. It simply draws a grid to a figure with a reversed z axis. [35].

```
plotGrid(G, final_plot_array(:,4), 'FaceColor','red','FaceAlpha', .9)
    ;
```

Listing 4.8: 3D final grid visualization

In this report, the figure 5.6 and figure 5.7 shows the out put results using the above code 4.8 and the final-plot-array from 5.2.

Chapter 5

Results

This chapter presents the results from the implementation done in Chapter 4. It starts by showing and explaining the results of Matlab program.

5.1 3D Environment using MRST

MRST contains a suite of visualization routines that make it easy to create visualizations of grids and results. The following figure 5.1 shows how to visualize grids, and subsets of grids and details the different routines included in MRST. [35]

5.2 Existing Wells in Grid View

Figure 5.2 and figure 5.3 shows the existing wells visualization. It is much important for this thesis because the existing wells are considered as hard and the surrounded grids are considered soft constraints. After generating the grid view the structure draws an existing well path and then converted the well path into grids. Those grids are equivalent to the specific points of the well path. Figure 5.2 shows the red curve inside the dark blue grids which is the actual existing well path.

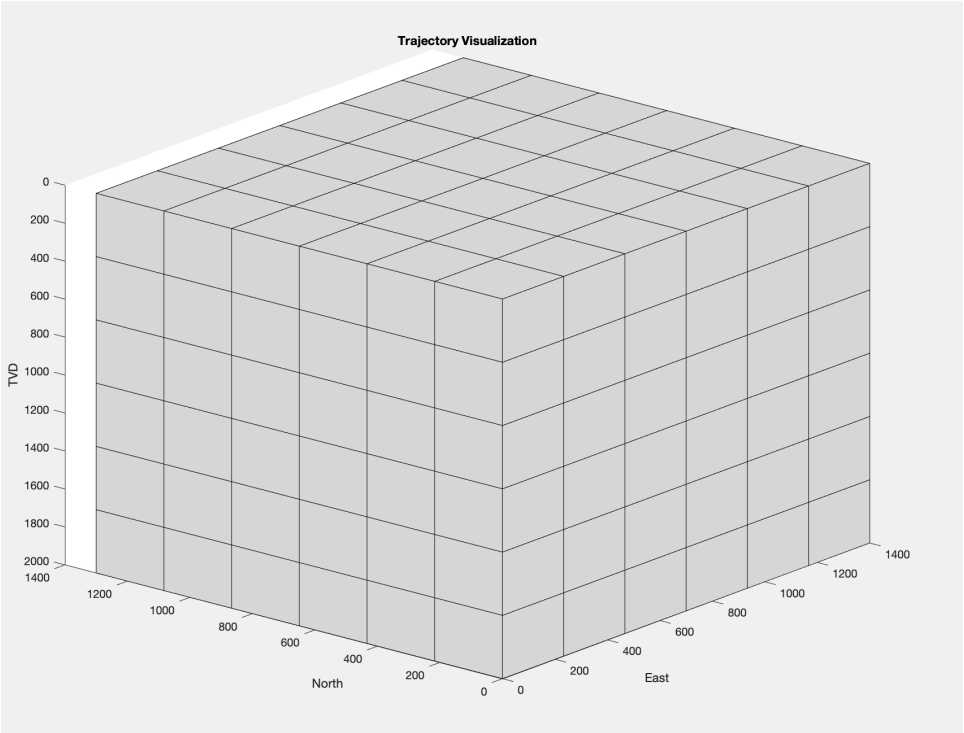


Figure 5.1: 3D visualisation of environment using MRST toolbox

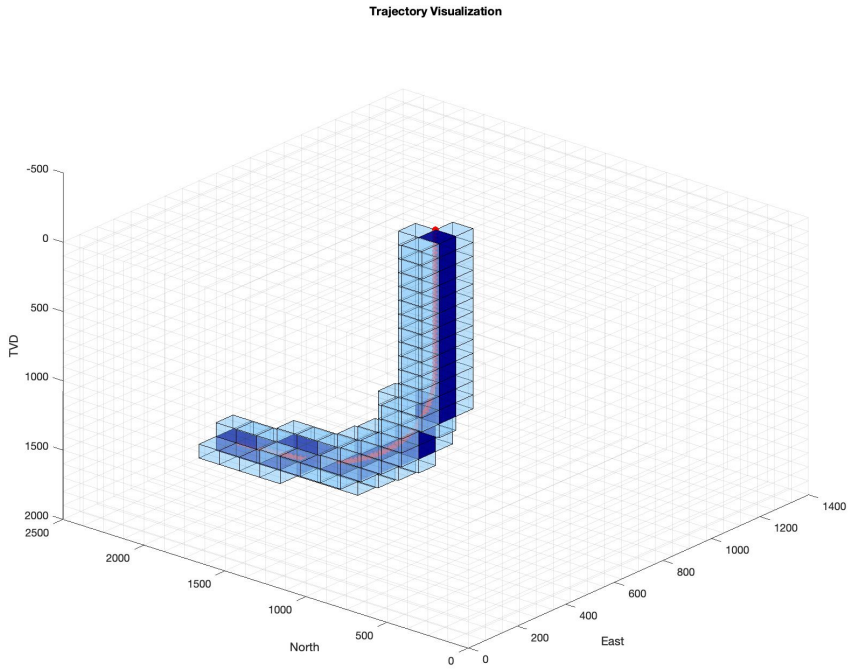


Figure 5.2: Single well 3D visualisation

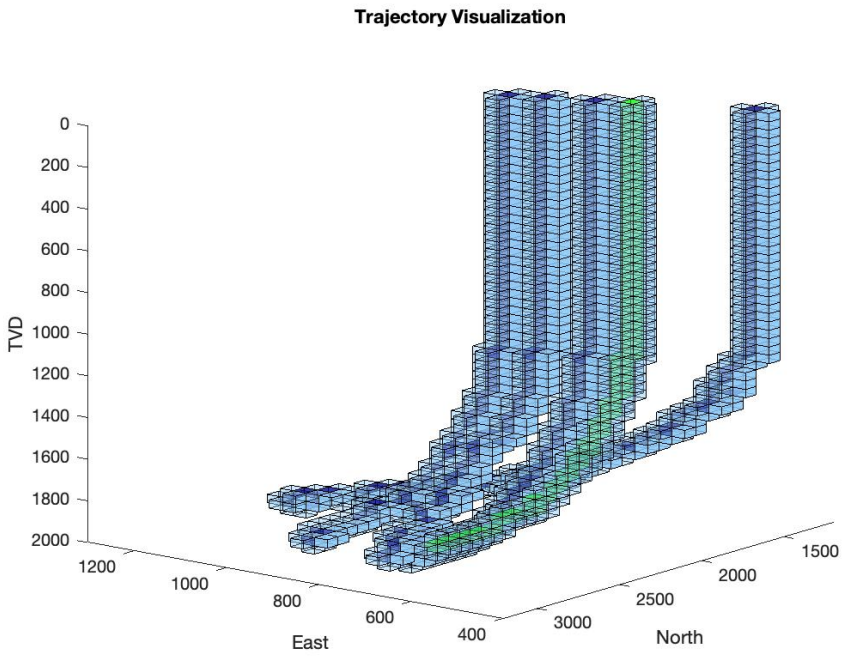


Figure 5.3: Multiple well 3D visualisation

5.3 Results from QLearning in Matlab

For the training purpose, this thesis used 8000 grids but the environment will work for any number of grids.The figure 5.4 and 5.5, shows the starting and the endpoint. These are grids in the 3D grid view but this grid was generated based on the North, East, and TVD value by using the point conversion process which is discussed in section 4.1. That means north was -140, east was 1320 and the TVD was 200 for the first point and (280,990,4000) for the second point respectively. As the system used [3300, 1400, 2000] points in each direction then the Matlab code found the center points based on grid numbers and finally visualize them into the graph.

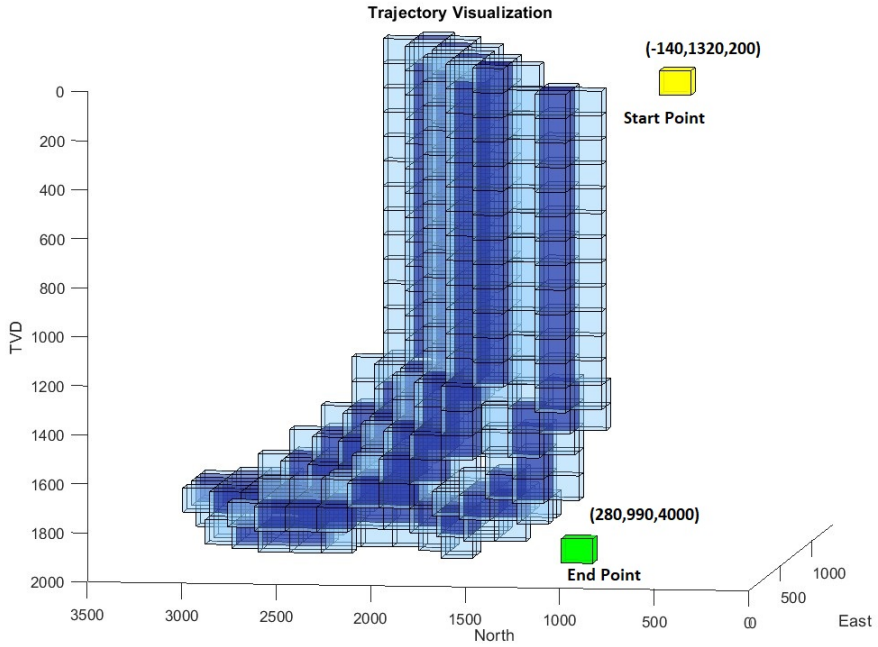
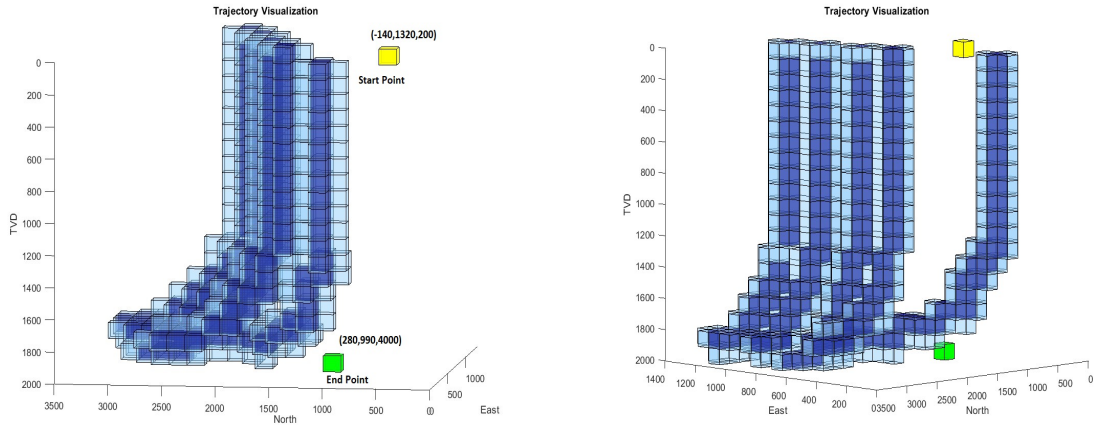


Figure 5.4: Start and end point zoom in view



(a) View another side **(b)** View One side

Figure 5.5: Start and end point

Figure 5.6 shows the optimized path from point A (start point shown in the yellow box) to point B (endpoint shown in the green box). The result is found after 10000 training episodes. It should be noted that the result is not unique. With the same condition, a different path can be found because all the paths/results have the same total cost. One more important note is that, as the system used the Q Learning greedy approach, it always tries to find a simpler and shortest path to reach its goal. That's why it gave a more flat and straight path as much as possible as well as a curve in some cases. This also indicates that, if the start point can be moved closer to the goal on the 3D surface, more drilling costs can be saved.

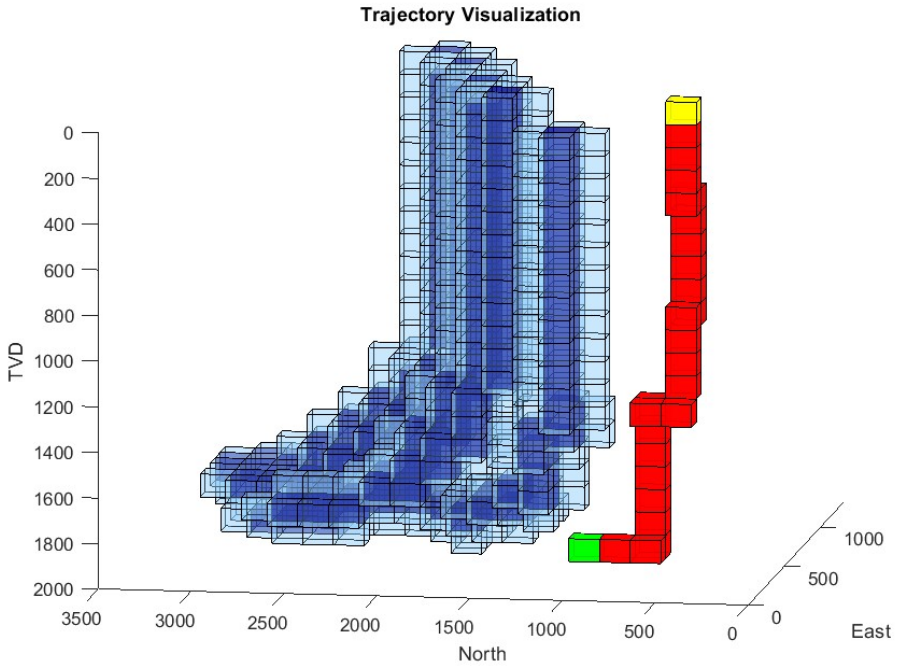


Figure 5.6: Path design using reinforcement learning

Figure 5.7 and 5.8 show two different points and their optimized path without any collisions. Here the endpoint is much closer to the soft as well as a hard obstacle.

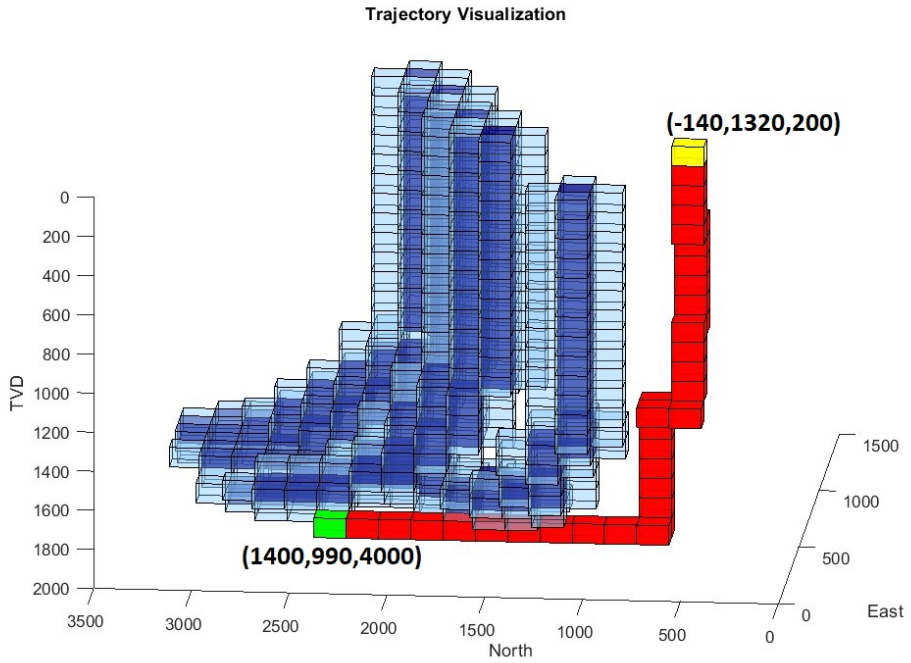


Figure 5.7: End point close to the obstacle

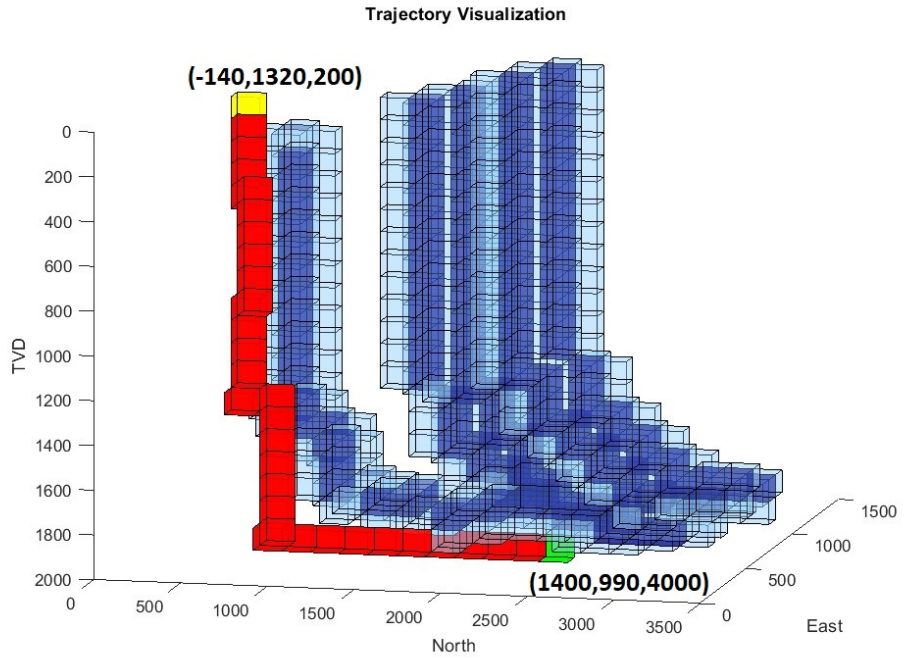


Figure 5.8: Avoiding collisions side view

Figure 5.9 shows a different angle view from the 3D visualization of the final result. The figure, clearly showed that the generated path is more clear in the three-dimensional grid world and represented the path using grids. As mentioned before that, all the 3D cubes have six directions and this result considered all the directions for finding the optimized path.

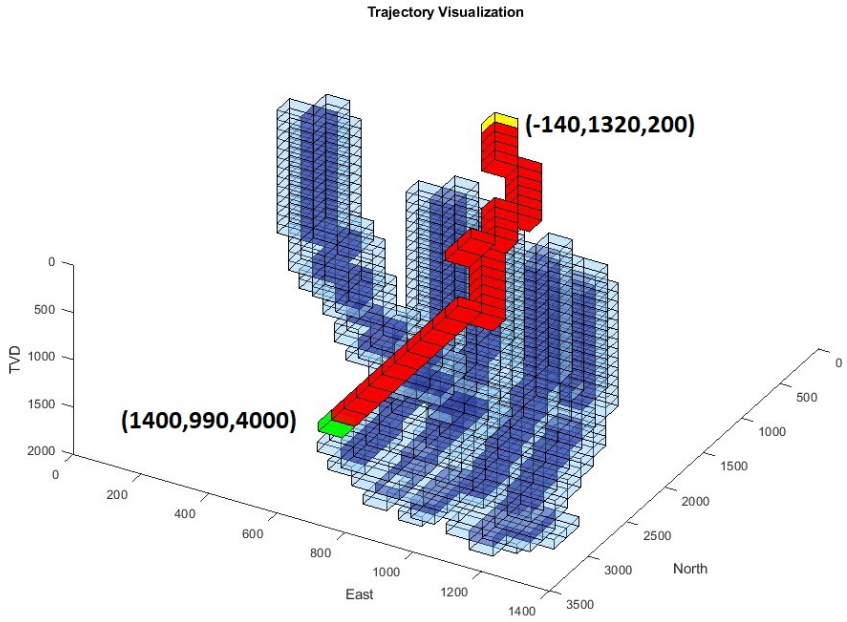


Figure 5.9: Avoiding collisions down side view

The listing 5.2 output array represents the path indexes and generates the path shown in figure 5.9 or 5.8 or 5.7. In this output array, the fourth column indicates the grid index or grid number. Also, the point-to-grid index conversion was done with the help of the MRST toolbox and used the following code:

```
s = 1:G.cells.num; %finding the cell number using mrst
[ijk{1:3}] = ind2sub(dims,G.cells.indexMap(:));
ijk = [ijk{:}];
indexw = ijk; % Storing i,j,k
index_up = indexw;
index_up(:,4) = s; % assigning the grid index or cell number in forth
    column
```

Listing 5.1: Grid index calculation using MRST

```
final_plot_array = [
     I   J    K    Index
     4   9    2    564
     4   9  3  964
     4   9  4  1364
     4   9  5  1764
     4   10   5  1784
     4   10   6  2184
     4   10   7  2584
     4   10   8  2984
     4   10   9  3384
     4   10   10   3784
     4   9  10   3764
     4   9  11   4164
     4   9  12   4564
     4   9  13   4964
     4   8    14   5344
     5   8  14   5345
```

```
19      5   9  14   5365
20      5   9  15   5765
21      5   9  16   6165
22      5   9  17   6565
23      5   9  18   6965
24      5   9  19   7365
25      5   9  20   7765
26      5   8  20   7745
27      6   8  20   7746
28      7   8  20   7747
29      8   8  20   7748
30      9   8  20   7749
31     10   8  20   7750
32     11   8  20   7751
33     12   8  20   7752
34     13   8  20   7753
35     14   8  20   7754
36 ];
```

Listing 5.2: Result points with index

Chapter 6

Discussion and Conclusion

6.1 Discussion on the results

Based on the results presented in Chapter 5, the Matlab program behaves as expected in terms of performance.

As only obstacles are given as a condition, the Matlab program finds a path by avoiding the obstacles. It should be noted that there are three dimensions in RL program and the result is as expected in the 3D grid view.

If more conditions are added, their behavior will change. If the condition of the mud layer is given, they may try to make a path close to obstacles or go through soft obstacles also.

6.2 Limitations

The Matlab program has some limitations in terms of time complexity. It takes more time to execute full code and train the whole model. This is one of the reasons for using 8000 grids for testing and training the agent. Also, did not consider the condition such as rock formation, mud layer, or steering condition, the algorithm provides a less realistic result for the instant curve structure of the well path. Furthermore, in some cases, the path goes into the soft obstacles that are near to the hard ones.

Another limitation was, that in the Matlab program, there were some visualization issues. For differentiating or positioning the existing wells, this program used variables to separate

these into the 3D environment.

6.3 Conclusion

By the way of conclusion, the objective of this thesis was to design a path and optimize with obstacle avoidance via reinforcement learning for well path optimization in a three-dimensional grid world. Although, there are similar works going on for well path optimization using different machine learning algorithms and enhancing existing optimization techniques. However, the result shows that reinforcement learning can be a good option for path optimization in the 3D environment using Matlab.

Last but not the least, it shows proper visualization in 3D than the previous 2D visualization in other thesis [19]. At the end of the conclusion, there will be some more opportunities to start work on using this 3D environment which is shown in this thesis section 4.2.

6.4 Future Work

This study can be used as a stepping stone to develop drilling optimization solutions, but before this, it is advisable to follow the next steps:

- Comprehensive testing, path design with different configurations.
- More research on MRST [10] toolbox to extent more knowledge.
- Evaluate using the Principal Component Analysis (PCA)[36], against traditional input selection based on petroleum engineering knowledge, and evaluate to determine the best methodology.

Some other drilling parameters should be considered in the future to make the outcomes more realistic.

References

[1] SPE. Directional Drilling. https://petrowiki.spe.org/Directional_drilling.

[2] Xiushan LIU. A true three-dimensional wellbore positioning method based on the earth ellipsoid. *Petroleum Exploration and Development*, 44:299–305, 04 2017. doi: 10.1016/S1876-3804(17)30034-4.

[3] 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. ISSN 1875-5100. doi: https://doi.org/10.1016/j.jngse.2014.05.029. URL https://www.sciencedirect.com/science/article/pii/S1875510014001553.

[4] David A. Wood. Hybrid bat flight optimization algorithm applied to complex wellbore trajectories highlights the relative contributions of metaheuristic components. *Journal of Natural Gas Science and Engineering*, 32:211–221, 2016. ISSN 1875-5100. doi: https://doi.org/10.1016/j.jngse.2016.04.024. URL https://www.sciencedirect.com/science/article/pii/S1875510016302475.

[5] Chiranth Hegde and Ken Gray. Evaluation of coupled machine learning models for drilling optimization. *Journal of Natural Gas Science and Engineering*, 56:397–407, 2018. ISSN 1875-5100. doi: https://doi.org/10.1016/j.jngse.2018.06.006. URL https://www.sciencedirect.com/science/article/pii/S1875510018302543.

[6] Zhang H. Gao D Sun, T. Application of the artificial fish swarm algorithm to well trajectory optimization. *Chem Technol Fuels Oils 55, 213–218*, 2019. doi: https://doi.org/10.1007/s10553-019-01023-7.

[7] Wendi Huang, Min Wu, Luefeng Chen, Jinhua She, Hiroshi Hashimoto, and Seiichi Kawata. Multiobjective drilling trajectory optimization considering parameter uncertainties. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–10, 2020. doi: 10.1109/TSMC.2020.3019428.

[8] Angelsen S. Sollie. O.K. Suyuthi A. Mistry R. Myrseth P. Tveiten Ellingsen, H.P. Study on machine learning in the norwegian petroleum industry. 2020. URL https://www.og21.no/en/strategy-and-analyses/og21-studies-and-analyses/previous-years-studies/.

[9] MATLAB. Get Started with MATLAB. https://se.mathworks.com/help/matlab/.

[10] MRST. MATLAB Reservoir Simulation Toolbox. https://www.sintef.no/projectweb/mrst/.

[11] Expert System. What is machine learning? a definition. URL https://expertsystem.com/machine-learning-definition.

[12] Tom Mitchel. *Machine Learning*. McGraw-Hill, illustrated edition, 1997. ISBN 0071154671, 9780071154673. URL http://www.cs.cmu.edu/~tom/mlbook.html.

[13] Wikipedia contributors. Reinforcement learning — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Reinforcement_learning&oldid=1029168114, 2021. [Online; accessed 13-July-2022].

[14] Andre Violante. Simple reinforcement learning: Q-learning. https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56, 2019.

[15] Suraj Bansal. Rl explained- reinforcing the intuition and math. https://medium.datadriveninvestor.com/rl-explained-reinforcing-the-intuition-and-math-fd1185369186, 2020. [Online; accessed 13-July-2022].

[16] Team-Saida. Gridworld. https://teamsaida.github.io/SAIDA_RL/GridWorld/.

[17] Joosten Gerard J.P. Brouwer Roald Vlemmix, Stijn and Jan-Dirk Jansen. Adjoint-based well trajectory optimization. 2009. doi: https://doi.org/10.2118/121891-MS.

[18] Zandvliet M.J. van Essen G.M. Brouwer D.R. Handels, M. and J.D. Jansen. Adjoint based well-placement optimization under production constraints. 2007. doi: https://doi.org/10.2118/105797-MS.

[19] UiS. Investigation and study on reinforcement learning for optimizing well path. https://uis.brage.unit.no/uis-xmlui/handle/11250/2788790.

[20] Open Lab App. OpenLab Drilling Web Application. https://openlab.app/.

[21] Drilling Data Web Application. Drilling Data Web Application. http://drillbotics.ddns.net/.

[22] eDrilling. Products. https://www.edrilling.no/#featured, 2021. [Online; accessed 11-July-2022].

[23] petromehras contribution. drilling-completion-software:wellplanner. https://www.petromehras.com/petroleum-software-directory/drilling-completion-software/wellplanner, 2019. [Online; accessed 11-July-2022].

[24] petromehras contribution. drilling-completion-software:wellsim. https://www.petromehras.com/petroleum-software-directory/drilling-completion-software/wellsim, 2019. [Online; accessed 11-July-2022].

[25] Schlumberger press release. Schlumberger announces delfi cognitive ep environment. https://www.slb.com/newsroom/press-release/2017/pr-2017-0913-delfi, 2017. [Online; accessed 11-July-2022].

[26] Schlumberger . Drillplan. https://www.software.slb.com/delfi/delfi-experience/drillplan, 2018. [Online; accessed 11-July-2022].

[27] Schlumberger . Drillbench. https://www.software.slb.com/products/drillbench#sectionFullWidthTable, 2015. [Online; accessed 11-July-2022].

[28] Halliburton . H012161 datasheet. https://www.landmark.solutions/Portals/0/LMSDocs/Datasheets/WellPlan_Software_DATASHEET-.pdf, 2016. [Online; accessed 11-July-2022].

[29] Halliburton . Decisionspace®365. https://www.landmark.solutions/Portals/0/LMSDocs/PDF/DecisionSpace365.pdf?ver=2021-04-13-194707-547, 2020. [Online; accessed 11-July-2022].

[30] Khosravanian R. Wood D.A. Mansouri, V. Optimizing the separation factor along a directional well trajectory to minimize collision risk. *J Petrol Explor Prod Techno*, 10: 2113–2125, 2020.

[31] Daniel Busby; Frédérik Pivot; Amine Tadjer. Use of data analytics to improve well placement optimization under uncertainty. *Abu Dhabi International Petroleum Exhibition Conference*, 2017. doi: https://doi.org/10.2118/188265-MS.

[32] Ranran Lu; Fahim Forouzanfar; A. C. Reynolds. Bi-objective optimization of well placement and controls using stosag. *SPE Reservoir Simulation Conference*, 2017. doi: https://doi.org/10.2118/182705-MS.

[33] Durlofsky Louis J. Yeten, Burak and Khalid Aziz. Optimization of nonconventional well type, location, and trajectory. *SPE J*, 2003. doi: https://doi.org/10.2118/86880-PA.

[34] Kshitij Dawar. Reinforcement learning for well location optimization.

[35] Visualizing in MRST. Visualizing in MRST. https://www.sintef.no/projectweb/mrst/documentation/tutorials/visualization-tutorial/.

[36] Hervé Abdi and Lynne J. Williams. Principal component analysis. *WIREs Computational Statistics*, 2(4):433–459, 2010. doi: 10.1002/wics.101. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.101.

Appendices

Appendix A

Matlab Code

A.1 Installed Packages and Softwares

Name	Version	Reference
MATLAB	R2022a	Download Matlab
MRST	MRST 2022a	Download MRST

A.2 Program Configuration Code

```
clear
clc
close all
mrstModule add ad-core ad-blackoil diagnostics wellpaths
data = 'data_refwell.xlsx'; %data_refwell
data1 = 'data_off2_north100.xlsx';
data3 = 'data_off3_east10.xlsx';
data4 = 'data_off4_east20.xlsx';
data5 = 'data_off5_angular.xlsx';
North = xlsread(data,'Wellpath','P2:P92');
East = xlsread(data,'Wellpath','Q2:Q92');
TVD = xlsread(data,'Wellpath','R2:R92');
```

```
North1 = xlsread(data1,'Wellpath','P2:P92');
East1 = xlsread(data1,'Wellpath','Q2:Q92');
TVD1 = xlsread(data1,'Wellpath','R2:R92');

North3 = xlsread(data3,'Wellpath','P2:P92');
East3 = xlsread(data3,'Wellpath','Q2:Q92');
TVD3 = xlsread(data3,'Wellpath','R2:R92');
North4 = xlsread(data4,'Wellpath','P2:P92');
East4 = xlsread(data4,'Wellpath','Q2:Q92');
TVD4 = xlsread(data4,'Wellpath','R2:R92');
North5 = xlsread(data5,'Wellpath','P2:P92');
East5 = xlsread(data5,'Wellpath','Q2:Q92');
TVD5 = xlsread(data5,'Wellpath','R2:R92');
pdims = [3300, 1400, 2000];
dims = [20, 20, 20];
XX = pdims./dims;
Find_centre = pdims./2; %Used this variable to find the center
G = cartGrid(dims, pdims);
G = computeGeometry(G);
% Here just using the absolate value of North
pos_North = abs(North);   %Ref well data north value
pos_North1 = abs(North1); %well1 data north value
pos_North3 = abs(North3); %well3 data north value
pos_North4 = abs(North4); %well4 data north value
pos_North5 = abs(North5); %well5 data north value

z = TVD;
zz = TVD1;
zzz = TVD3;

a = [pos_North,East];
```

```
44
45 aa = [a(:,1)+Find_centre(1),a(:,2)+Find_centre(2)];
46 Cent_North = pos_North+Find_centre(1);
47 Cent_East = East+Find_centre(2);
48 b = [pos_North1,East1];
49 bb = [b(:,1)+Find_centre(1)+XX(1),b(:,2)+Find_centre(2)+XX(1)];
50
51 c = [pos_North3,East3];
52 cc = [c(:,1)+Find_centre(1)+XX(1)*2,c(:,2)+Find_centre(2)+XX(1)*2];
53
54 c4 = [pos_North4,East4];
55 cc4 = [c4(:,1)+Find_centre(1)+XX(1)*2+XX(1),c4(:,2)+Find_centre(2)+XX
    (1)*2+XX(1)];
56
57 c5 = [pos_North5,East5];
58 cc5 = [c5(:,1)+Find_centre(1)-XX(1)*3,c5(:,2)+Find_centre(2)-XX(1)
    *3];
59
60 wellpath1 = makeSingleWellpath([aa,z]);
61 wellpath2 = makeSingleWellpath([bb, zz]);
62 wellpath3 = makeSingleWellpath([cc, zzz]);
63 wellpath4 = makeSingleWellpath([cc4, TVD4]);
64 wellpath5 = makeSingleWellpath([cc5, TVD5]);
65
66 % Combine into single well path
67 wellpath_fork = combineWellPaths({wellpath1}); %wellpath0 , ,
    wellpath3
68 wellpath_fork1 = combineWellPaths({wellpath2});
69 wellpath_fork2 = combineWellPaths({wellpath3});
70 wellpath_fork3 = combineWellPaths({wellpath4});
71 wellpath_fork4 = combineWellPaths({wellpath5});
```

```
72
73 % Determine the cells
74
75 [cells_fork1, segInd_fork1, ~, ~, DT1] = findWellPathCells(G,
       wellpath_fork);
76 [cells_fork2, segInd_fork2, ~, ~, DT2] = findWellPathCells(G,
       wellpath_fork1);
77 [cells_fork3, segInd_fork3, ~, ~, DT3] = findWellPathCells(G,
       wellpath_fork2);
78 [cells_fork4, segInd_fork4, ~, ~, DT4] = findWellPathCells(G,
       wellpath_fork3);
79 [cells_fork, segInd_fork, ~, ~, DT] = findWellPathCells(G,
       wellpath_fork4);
80
81 clf;
82 c = G.cells.centroids;
83 n = G.nodes.coords;
84 s = 1:G.cells.num;
85
86 [ijk{1:3}] = ind2sub(dims,G.cells.indexMap(:));
87 ijk = [ijk{:}];
88 indexw = ijk;
89 index_up = indexw;
90 index_up(:,4) = s;
91 dem1 = pos_North+Find_centre(1);
92 dem2 = East+Find_centre(2);
93
94 first = uint8(dem1/XX(1));
95 second = uint8(dem2/XX(2));
96 third = uint8(TVD/XX(3));
97
```

```
point_array = [first ,second ,third ];
point_array(~point_array) = 1;
point_array =unique(point_array ,'rows');
point_array(point_array > dims(1)) = dims(1);

[tf, index]=ismember(point_array ,indexw,'rows');
index = nonzeros(index');

%---------SECOND PATH or PATH B------
dem11 = pos_North1+Find_centre(1)+XX(1);
dem21 = East1+Find_centre(2)+XX(1);

first1 = uint8(dem11/XX(1));
second1 = uint8(dem21/XX(2));
third1 = uint8(TVD1/XX(3));

point_array1 = [first1 ,second1 ,third1 ];
point_array1(~point_array1) = 1;
point_array1 =unique(point_array1 ,'rows');
point_array1(point_array1 > dims(1)) = dims(1);
[tf01, index01]=ismember(point_array1 ,indexw,'rows');
index01 = nonzeros(index01');

%--------PATH B ENDED---------

%-----------------THIRD PATH or PATH C----------
dem12 = pos_North3+Find_centre(1)+XX(1)*2;
dem22 = East3+Find_centre(2)+XX(1)*2;

first2 = uint8(dem12/XX(1));
second2 = uint8(dem22/XX(2));
```

```
129 third2 = uint8(TVD3/XX(3));
130
131 point_array2 = [first2,second2,third2];
132 point_array2(~point_array2) = 1;
133 point_array2 =unique(point_array2,'rows');
134 point_array2(point_array2 > dims(1)) = dims(1);
135 [tf02, index02]=ismember(point_array2,indexw,'rows');
136 index02 = nonzeros(index02');
137
138 %----------------------PATH C ENDED-------------
139
140 %-----------------FOURTH PATH or PATH D--------
141 dem13 = pos_North4+Find_centre(1)+XX(1)*2+XX(1);
142 dem23 = East4+Find_centre(2)+XX(1)*2+XX(1);
143 first3 = uint8(dem13/XX(1));
144 second3 = uint8(dem23/XX(2));
145 third3 = uint8(TVD4/XX(3));
146 point_array3 = [first3,second3,third3];
147 point_array3(~point_array3) = 1;
148 point_array3 =unique(point_array3,'rows');
149 point_array3(point_array3 > dims(1)) = dims(1);
150 [tf03, index03]=ismember(point_array3,indexw,'rows');
151 index03 = nonzeros(index03');
152
153 %-----------------PATH D ENDED-------------------
154
155 %------------------FIFTH PATH or PATH E-----------
156 dem14 = pos_North5+Find_centre(1)-XX(1)*3;
157 dem24 = East5+Find_centre(2)-XX(1)*3;
158 first4 = uint8(dem14/XX(1));
159 second4 = uint8(dem24/XX(2));
```

```
160 third4 = uint8(TVD5/XX(3));
161 point_array4 = [first4 ,second4 ,third4 ];
162 point_array4(~point_array4) = 1;
163 point_array4 =unique(point_array4 ,'rows');
164 point_array4(point_array4 > dims(1)) = dims(1);
165 [tf04 , index04]=ismember(point_array4 ,indexw ,'rows');
166 index04 = nonzeros(index04 ');
167 %------------------PATH E ENDED--------------------
168
169 % -----------Left Side Calculation:------------------
170 %----------One---------------------
171 left_side_arr = [point_array(:,1)+1,point_array(:,2),point_array(:,3)
      ];
172 left_side_arr =unique(left_side_arr ,'rows');
173 left_side_arr(left_side_arr > dims(1)) = dims(1);
174 [tf2 , index2]=ismember(left_side_arr ,indexw ,'rows');
175 Acommon = intersect(index ,index2);
176 left_side_arr_point = setxor(index2 ,Acommon);
177 left_side_arr_point= nonzeros(left_side_arr_point ');
178
179 %-----------Two--------------------
180 left_side_arr2 = [point_array1(:,1)+1,point_array1(:,2),point_array1
      (:,3)];
181 left_side_arr2 =unique(left_side_arr2 ,'rows');
182 left_side_arr2(left_side_arr2 > dims(1)) = dims(1);
183 [tf2_2 , index2_2]=ismember(left_side_arr2 ,indexw ,'rows');
184 Acommon_2 = intersect(index ,index2_2);
185 left_side_arr_point_2 = setxor(index2_2 ,Acommon_2);
186 left_side_arr_point_2= nonzeros(left_side_arr_point_2 ');
187
188 %------------Three-----------------
```

```
189 left_side_arr3 = [point_array2(:,1)+1,point_array2(:,2),point_array2
    (:,3)];
190 left_side_arr3 =unique(left_side_arr3,'rows');
191 left_side_arr3(left_side_arr3 > dims(1)) = dims(1);
192 [tf2_3, index2_3]=ismember(left_side_arr3,indexw,'rows');
193 Acommon_3 = intersect(index,index2_3);
194 left_side_arr_point_3 = setxor(index2_3,Acommon_3);
195 left_side_arr_point_3= nonzeros(left_side_arr_point_3');
196
197 %------------Five-----------------
198 left_side_arr5 = [point_array3(:,1)+1,point_array3(:,2),point_array3
    (:,3)];
199 left_side_arr5 =unique(left_side_arr5,'rows');
200 left_side_arr5(left_side_arr5 > dims(1)) = dims(1);
201 [tf2_5, index2_5]=ismember(left_side_arr5,indexw,'rows');
202 Acommon_5 = intersect(index,index2_5);
203 left_side_arr_point_5 = setxor(index2_5,Acommon_5);
204 left_side_arr_point_5= nonzeros(left_side_arr_point_5');
205
206 %------------Angular-------------------
207 left_side_arr4 = [point_array4(:,1)+1,point_array4(:,2),point_array4
    (:,3)];
208 left_side_arr4 =unique(left_side_arr4,'rows');
209 left_side_arr4(left_side_arr4 > dims(1)) = dims(1);
210 [tf24, index24]=ismember(left_side_arr4,indexw,'rows');
211 Acommon_4 = intersect(index,index24);
212 left_side_arr_point_4 = setxor(index24,Acommon_4);
213 left_side_arr_point_4= nonzeros(left_side_arr_point_4');
214 %----------------Left Side Calculation End -----------------
215
216 %----------------Reference Well ONE------------
```

```
217 right_side_arr = [point_array(:,1),point_array(:,2)+1,point_array
    (:,3)];
218 [tf3, index3]=ismember(right_side_arr,indexw,'rows');
219
220 %----------------Reference Well Two--------------------
221 right_side_arr_2 = [point_array1(:,1),point_array1(:,2)+1,
    point_array1(:,3)];
222 [tf3_2, index3_2]=ismember(right_side_arr_2,indexw,'rows');
223
224 %----------------Reference Well Three--------------------
225 right_side_arr_3 = [point_array2(:,1),point_array2(:,2)+1,
    point_array2(:,3)];
226 [tf3_3, index3_3]=ismember(right_side_arr_3,indexw,'rows');
227
228 %----------------Reference Well Five--------------------
229 right_side_arr_5 = [point_array3(:,1),point_array3(:,2)+1,
    point_array3(:,3)];
230 [tf3_5, index3_5]=ismember(right_side_arr_5,indexw,'rows');
231
232 %----------------Reference Well Angular--------------------
233 right_side_arr_4 = [point_array4(:,1),point_array4(:,2)+1,
    point_array4(:,3)];
234 [tf3_4, index3_4]=ismember(right_side_arr_4,indexw,'rows');
235
236
237 % -------------Up Side Calculation Using K:-----------------------
238 up_side_arr = [point_array(:,1),point_array(:,2),point_array(:,3)-1];
239 [tf4, index4]=ismember(up_side_arr,indexw,'rows');
240 Acommon2 = intersect(index,index4);
241 up_side_arr_point = setxor(index4,Acommon2);
242 Acommon3 = intersect(left_side_arr_point,index4);
```

```
243 up_side_arr_point = setxor(up_side_arr_point,Acommon3);
244 up_side_arr_point= nonzeros(up_side_arr_point');
245 %--------------Two------------------
246 up_side_arr_2 = [point_array1(:,1),point_array1(:,2),point_array1
    (:,3)-1];
247 [tf4_2, index4_2]=ismember(up_side_arr_2,indexw,'rows');
248 Acommon2_2 = intersect(index,index4_2);
249 up_side_arr_point_2 = setxor(index4_2,Acommon2_2);
250 Acommon3_2 = intersect(left_side_arr_point_2,index4_2);
251 up_side_arr_point_2 = setxor(up_side_arr_point_2,Acommon3_2);
252 up_side_arr_point_2= nonzeros(up_side_arr_point_2');
253 %--------------Three------------------
254 up_side_arr_3 = [point_array2(:,1),point_array2(:,2),point_array2
    (:,3)-1];
255 [tf4_3, index4_3]=ismember(up_side_arr_3,indexw,'rows');
256 Acommon2_3 = intersect(index,index4_3);
257 up_side_arr_point_3 = setxor(index4_3,Acommon2_3);
258 Acommon3_3 = intersect(left_side_arr_point_3,index4_3);
259 up_side_arr_point_3 = setxor(up_side_arr_point_3,Acommon3_3);
260 up_side_arr_point_3= nonzeros(up_side_arr_point_3');
261 %--------------Five-------------------------------
262 up_side_arr_5 = [point_array3(:,1),point_array3(:,2),point_array3
    (:,3)-1];
263 [tf4_5, index4_5]=ismember(up_side_arr_5,indexw,'rows');
264 Acommon2_5 = intersect(index,index4_5);
265 up_side_arr_point_5 = setxor(index4_5,Acommon2_5);
266 Acommon3_5 = intersect(left_side_arr_point_5,index4_5);
267 up_side_arr_point_5 = setxor(up_side_arr_point_5,Acommon3_5);
268 up_side_arr_point_5= nonzeros(up_side_arr_point_5');
269
270 %--------------Angular-------------------------------
```

```
271 up_side_arr_4 = [point_array4(:,1),point_array4(:,2),point_array4
    (:,3)-1];
272 [tf4_4, index4_4]=ismember(up_side_arr_4,indexw,'rows');
273 Acommon2_4 = intersect(index,index4_4);
274 up_side_arr_point_4 = setxor(index4_4,Acommon2_4);
275 Acommon3_4 = intersect(left_side_arr_point_4,index4_4);
276 up_side_arr_point_4 = setxor(up_side_arr_point_4,Acommon3_4);
277 up_side_arr_point_4= nonzeros(up_side_arr_point_4');
278 %-----------------Up Side Calculation Using K End ------------------
279
280 % -------------Down Side Calculation Using K:-----------------------
281 down_side_arr = [point_array(:,1),point_array(:,2),point_array(:,3)
    +1];
282 down_side_arr(down_side_arr > dims(1)) = dims(1);
283 [tf5, index5]=ismember(down_side_arr,indexw,'rows');
284 Acommon4 = intersect(index,index5);
285 down_side_arr_point = setxor(index5,Acommon4);
286 down_side_arr_point= nonzeros(down_side_arr_point');
287 down_side_arr2 = [point_array(:,1),point_array(:,2)-1,point_array
    (:,3)];
288 [tf6, index6]=ismember(down_side_arr2,indexw,'rows');
289 Acommon5 = intersect(index,index6);
290 down_side_arr_point2 = setxor(index6,Acommon5);
291 down_side_arr_point2= nonzeros(down_side_arr_point2');
292
293 %---------------Two--------------------------
294 down_side_arr_2 = [point_array1(:,1),point_array1(:,2),point_array1
    (:,3)+1];
295 down_side_arr_2(down_side_arr_2 > dims(1)) = dims(1);
296 [tf5_2, index5_2]=ismember(down_side_arr_2,indexw,'rows');
297 Acommon4_2 = intersect(index,index5_2);
```

```
298 down_side_arr_point_2 = setxor(index5_2,Acommon4_2);
299 down_side_arr_point_2= nonzeros(down_side_arr_point_2');
300 down_side_arr2_2 = [point_array1(:,1),point_array1(:,2)-1,
    point_array1(:,3)];
301 [tf6_2, index6_2]=ismember(down_side_arr2_2,indexw,'rows');
302 Acommon5_2 = intersect(index,index6_2);
303 down_side_arr_point2_2 = setxor(index6_2,Acommon5_2);
304 down_side_arr_point2_2= nonzeros(down_side_arr_point2_2');
305
306 %----------------Three----------------------------
307 down_side_arr_3 = [point_array2(:,1),point_array2(:,2),point_array2
    (:,3)+1];
308 down_side_arr_3(down_side_arr_3 > dims(1)) = dims(1);
309 [tf5_3, index5_3]=ismember(down_side_arr_3,indexw,'rows');
310 Acommon4_3 = intersect(index,index5_3);
311 down_side_arr_point_3 = setxor(index5_3,Acommon4_3);
312 down_side_arr_point_3= nonzeros(down_side_arr_point_3');
313 down_side_arr2_3 = [point_array2(:,1),point_array2(:,2)-1,
    point_array2(:,3)];
314 [tf6_3, index6_3]=ismember(down_side_arr2_3,indexw,'rows');
315 Acommon5_3 = intersect(index,index6_3);
316 down_side_arr_point2_3 = setxor(index6_3,Acommon5_3);
317 down_side_arr_point2_3= nonzeros(down_side_arr_point2_3');
318
319 %----------------Five-------------------------
320 down_side_arr_5 = [point_array3(:,1),point_array3(:,2),point_array3
    (:,3)+1];
321 down_side_arr_5(down_side_arr_5 > dims(1)) = dims(1);
322 [tf5_5, index5_5]=ismember(down_side_arr_5,indexw,'rows');
323 Acommon4_5 = intersect(index,index5_5);
324 down_side_arr_point_5 = setxor(index5_5,Acommon4_5);
```

```
325 down_side_arr_point_5= nonzeros(down_side_arr_point_5');
326 down_side_arr2_5 = [point_array3(:,1),point_array3(:,2)-1,
    point_array3(:,3)];
327 [tf6_5, index6_5]=ismember(down_side_arr2_5,indexw,'rows');
328 Acommon5_5 = intersect(index,index6_5);
329 down_side_arr_point2_5 = setxor(index6_5,Acommon5_5);
330 down_side_arr_point2_5= nonzeros(down_side_arr_point2_5');
331
332 %-----------------Angular-------------------------
333 down_side_arr_4 = [point_array4(:,1),point_array4(:,2),point_array4
    (:,3)+1];
334 down_side_arr_4(down_side_arr_4 > dims(1)) = dims(1);
335 [tf5_4, index5_4]=ismember(down_side_arr_4,indexw,'rows');
336 Acommon4_4 = intersect(index,index5_4);
337 down_side_arr_point_4 = setxor(index5_4,Acommon4_4);
338 down_side_arr_point_4= nonzeros(down_side_arr_point_4');
339 down_side_arr2_4 = [point_array4(:,1),point_array4(:,2)-1,
    point_array4(:,3)];
340 [tf6_4, index6_4]=ismember(down_side_arr2_4,indexw,'rows');
341 Acommon5_4 = intersect(index,index6_4);
342 down_side_arr_point2_4 = setxor(index6_4,Acommon5_4);
343 down_side_arr_point2_4= nonzeros(down_side_arr_point2_4');
344 %-------Down Side Calculation Using K End ----------
345
346 % -------------Back Side Calculation:----------------------
347 %--------------One-------------------
348 back_side_arr = [point_array(:,1)-1,point_array(:,2),point_array(:,3)
    ];
349
350 [tf7, index7]=ismember(back_side_arr,indexw,'rows');
351
```

```
352 Acommon6 = intersect(index,index7);
353 back_side_arr_point = setxor(index7,Acommon6);
354 back_side_arr_point= nonzeros(back_side_arr_point');
355 %-------------Two-------------------
356 back_side_arr_2 = [point_array1(:,1)-1,point_array1(:,2),point_array1
    (:,3)];
357
358 [tf7_2, index7_2]=ismember(back_side_arr_2,indexw,'rows');
359
360 Acommon6_2 = intersect(index,index7_2);
361 back_side_arr_point_2 = setxor(index7_2,Acommon6_2);
362 back_side_arr_point_2= nonzeros(back_side_arr_point_2');
363 %-------------Three-------------------
364 back_side_arr_3 = [point_array2(:,1)-1,point_array2(:,2),point_array2
    (:,3)];
365
366 [tf7_3, index7_3]=ismember(back_side_arr_3,indexw,'rows');
367
368 Acommon6_3 = intersect(index,index7_3);
369 back_side_arr_point_3 = setxor(index7_3,Acommon6_3);
370 back_side_arr_point_3= nonzeros(back_side_arr_point_3');
371 %-------------Five-----------------
372 back_side_arr_5 = [point_array3(:,1)-1,point_array3(:,2),point_array3
    (:,3)];
373
374 [tf7_5, index7_5]=ismember(back_side_arr_5,indexw,'rows');
375
376 Acommon6_5 = intersect(index,index7_5);
377 back_side_arr_point_5 = setxor(index7_5,Acommon6_5);
378 back_side_arr_point_5= nonzeros(back_side_arr_point_5');
379 %-------------Angular-----------------
```

```
380 back_side_arr_4 = [point_array4(:,1)-1,point_array4(:,2),point_array4
    (:,3)];
381
382 [tf7_4, index7_4]=ismember(back_side_arr_4,indexw,'rows');
383
384 Acommon6_4 = intersect(index,index7_4);
385 back_side_arr_point_4 = setxor(index7_4,Acommon6_4);
386 back_side_arr_point_4= nonzeros(back_side_arr_point_4');
387
388 %-----------------Back Side Calculation End -----------------
389 soft1 = [left_side_arr;right_side_arr;down_side_arr;down_side_arr2;
    back_side_arr;up_side_arr];
390 soft2 = [left_side_arr2;right_side_arr_2;down_side_arr_2;
    down_side_arr2_2;back_side_arr_2];
391 soft3 = [left_side_arr3;right_side_arr_3;down_side_arr_3;
    down_side_arr2_3;back_side_arr_3];
392 soft4 = [left_side_arr4;right_side_arr_4;down_side_arr_4;
    down_side_arr2_4;back_side_arr_4];
393 soft5 = [left_side_arr5;right_side_arr_5;down_side_arr_5;
    down_side_arr2_5;back_side_arr_5];
394
395 start_point = point_conversion(-140,1320,200);
396 [tf_start, index_start]=ismember(start_point,indexw,'rows');
397
398 start_point(:,4) = index_start;
399 start_point(:,5) = 10;
400
401 end_point = [15,8,20]; %point_conversion(280,990,4000); %
402 [tf_end, index_end]=ismember(end_point,indexw,'rows');
403 end_point(:,4) = index_end;
404 end_point(:,5) = 100000;
```

```
405
406 %----------Reinforcement Learning Code-----------------
407 q_values = index_up;
408 q_values(:,5) = 10;
409 q_values = int32(q_values);
410 actions = ["up", "right", "down", "left"];
411 Hard_constraints = [point_array; point_array1;point_array2;
    point_array3;point_array4];
412 Hard_constraints_index = [index; index01;index02;index03;index04];
413 Soft_constraints = [soft1;soft2;soft3;soft4;soft5];
414 [soft_cons, Soft_constraints_index]=ismember(Soft_constraints,indexw,
    'rows');
415 Soft_constraints = int32(Soft_constraints);
416 %Soft_constraints_index = nonzeros(Soft_constraints_index);
417 Soft_constraints(:,4) = Soft_constraints_index;
418 idx = any(Soft_constraints==0,2);
419 zero_rows = find(idx);
420 Soft_constraints( zero_rows, : ) = [];
421 Soft_constraints = unique(Soft_constraints,'rows');
422 Hard_constraints = int32(Hard_constraints);
423 Hard_constraints(:,4) = Hard_constraints_index;
424
425 H_S = vertcat(Hard_constraints,Soft_constraints);
426 [tra,inda] = ismember(H_S,index_up,'rows');
427 [q_values,ps] = removerows(q_values,'ind',inda);
428
429 Hard_constraints(:,5) = -1000;
430 Hard_constraints = unique(Hard_constraints,'rows');
431 Hard_constraints= sortrows(Hard_constraints,4);
432
433 global Combine;
```

```
Soft_constraints(:,5) = -10;
Soft_constraints= sortrows(Soft_constraints,4);
Combine = vertcat(Hard_constraints,Soft_constraints,q_values);
Combine = unique(Combine,'rows');
Combine = sortrows(Combine,4);

[~,idCombine] = unique(Combine(:,4));
Combine = Combine(idCombine,:);

apstart = find(Combine(:,4)==start_point(:,4));
Combine(apstart,:) =  start_point;

apend = find(Combine(:,4)==end_point(:,4));
Combine(apend,:) =  end_point;

VarNames = {'i', 'j', 'k', 'Index','Rewards'};
T = table(Combine(:,1),Combine(:,2),Combine(:,3),Combine(:,4),Combine
    (:,5), 'VariableNames',VarNames);
tblB = sortrows(T,'Index');

equal_index = mod(1:G.cells.num,2) == 0;

color_index_trajectory = index; %[1 37 73 109 145 146 182 183 184 185
    186];%1 37
color_index_trajectory2 = index01;
color_index_trajectory3 = index02;
color_index_trajectory4 = index03;
color_index_trajectory5 = index04;

color_index_soft_right = index3;
color_index_soft_right_2 = index3_2;
```

```
463 color_index_soft_right_3 = index3_3;
464 color_index_soft_right_4 = index3_4;
465 color_index_soft_right_5 = index3_5;
466
467 color_index_soft_left = left_side_arr_point;
468 color_index_soft_left_2 = left_side_arr_point_2;
469 color_index_soft_left_3 = left_side_arr_point_3;
470 color_index_soft_left_4 = left_side_arr_point_4;
471 color_index_soft_left_5 = left_side_arr_point_5;
472
473 global old_index;
474 global Side_up_QVal;
475 global Side_down_QVal;
476 global Side_left_QVal;
477 global Side_right_QVal;
478 global Side_back_QVal;
479 global Side_front_QVal;
480 global q_vals_for_6;
481
482 epsilon = 0.8; %the percentage of time when we should take the best
      action (instead of a random action)
483 discount_factor = 0.9; %discount factor for future rewards
484 learning_rate = 0.4;
485
486  for episode = 1:5
487      start_index = get_starting_location();
488     %disp(start_index(:,3));
489
490      while ~is_terminal_state(start_index)
491
492             % Searching different Q Values
```

```
493             if start_index(:,3) > 1
494             Side_up = [start_index(:,1),start_index(:,2),start_index
    (:,3)-1];
495             [~,idx0] = ismember(Side_up,Combine(:,1:3),'rows');
496             Side_up(:,4) = idx0;
497             lidx0 = Combine(idx0,end);
498             Side_up(:,5) = lidx0;
499             Side_up_QVal = Side_up(:,5);
500             else
501             Side_up_QVal = 0;
502             end
503
504             if start_index(:,3) < max(Combine(:,3))
505             Side_down = [start_index(:,1),start_index(:,2),
    start_index(:,3)+1];
506             [~,idx] = ismember(Side_down,Combine(:,1:3),'rows');
507             Side_down(:,4) = idx;
508             lidx = Combine(idx,end);
509             Side_down(:,5) = lidx;
510             Side_down_QVal = Side_down(:,5);
511             else
512             Side_down_QVal = 0;
513             end
514
515             if start_index(:,1) < max(Combine(:,3))
516             Side_left = [start_index(:,1)+1,start_index(:,2),
    start_index(:,3)];
517             [~,idx] = ismember(Side_left,Combine(:,1:3),'rows');
518             Side_left(:,4) = idx;
519             lidx = Combine(idx,end);
520             Side_left(:,5) = lidx;
```

```
521             Side_left_QVal = Side_left(:,5);
522             else
523             Side_left_QVal = 0;
524             end
525
526             if start_index(:,1) > 1
527             Side_right = [start_index(:,1)-1,start_index(:,2),
    start_index(:,3)];
528             [~,idx] = ismember(Side_right,Combine(:,1:3),'rows');
529             Side_right(:,4) = unique(idx);
530             lidx = Combine(idx,end);
531             Side_right(:,5) = lidx;
532             Side_right_QVal = Side_right(:,5);
533             else
534             Side_right_QVal = 0;
535             end
536
537             if start_index(:,2) < max(Combine(:,3))
538             Side_back = [start_index(:,1),start_index(:,2)+1,
    start_index(:,3)];
539             [~,idx] = ismember(Side_back,Combine(:,1:3),'rows');
540             Side_back(:,4) = idx;
541             lidx = Combine(idx,end);
542             Side_back(:,5) = lidx;
543             Side_back_QVal = Side_back(:,5);
544             else
545             Side_back_QVal = 0;
546             end
547
548             if start_index(:,2) > 1
549             Side_front = [start_index(:,1),start_index(:,2)-1,
```

```
start_index(:,3)];
550             [~,idx] = ismember(Side_front,Combine(:,1:3),'rows');
551             Side_front(:,4) = unique(idx);
552             lidx = Combine(idx,end);
553             Side_front(:,5) = lidx;
554             Side_front_QVal = Side_front(:,5);
555             else
556             Side_front_QVal = 0;
557             end
558
559             q_vals_for_6 = [Side_up_QVal,Side_down_QVal,
Side_left_QVal,Side_right_QVal,Side_back_QVal,Side_front_QVal];
560             action_index = get_next_action(start_index,epsilon);
561             old_index = start_index; %store the old index here
562             start_index = get_next_location(start_index,action_index)
;
563             reward = start_index(:,5);
564             old_q_values = old_index(:,5);
565             temporal_difference = double(reward + (discount_factor *
(start_index(:,5)))-old_q_values);
566
567             new_q_values = old_q_values + (learning_rate *
temporal_difference);
568             %disp("new_q_values");
569             %disp(new_q_values);
570             old_index(:,5) = new_q_values;
571             %disp(old_index);
572             [~,idx09] = ismember(old_index(:,1:4),Combine(:,1:4),'
rows');
573             Combine(idx09,end)= new_q_values;
574   end
```

```
575 disp("Training complete");
576 end
577 plotGrid(G, color_index_trajectory, 'FaceColor',[0, 0, 0.545],'
    FaceAlpha', .5); %[0, 0, 0.545]
578 plotGrid(G, color_index_trajectory2, 'FaceColor',[0, 0, 0.545],'
    FaceAlpha', .5);
579 plotGrid(G, color_index_trajectory3, 'FaceColor',[0, 0, 0.545],'
    FaceAlpha', .5);
580 plotGrid(G, color_index_trajectory4, 'FaceColor',[0, 0, 0.545],'
    FaceAlpha', .5);
581 plotGrid(G, color_index_trajectory5, 'FaceColor',[0, 0, 0.545],'
    FaceAlpha', .5);
582 plotGrid(G, final_plot_array(:,4), 'FaceColor','red','FaceAlpha', .9)
    ;
583 plotGrid(G, index_start, 'FaceColor','yellow','FaceAlpha', .9);
584 plotGrid(G, index_end, 'FaceColor','green','FaceAlpha', .9);
585 plotGrid(G, Soft_constraints(:,4), 'FaceColor',[0.5843 0.8157
    0.9882],'FaceAlpha', .3);
586 set(gca,'Xdir','reverse');
587 %set(gca, 'layer', 'top');
588 view([40, 30])
589 xlabel('North');
590 ylabel('East');
591 zlabel('TVD');
592 title('Trajectory Visualization')
593 %--------------END ENVIRONMENT------------------------------------
594
```

Listing A.1: Program Configuration Code