

Tor Arne Tjøland
Mads Sundseth

Design and Implementation of a Simulation System for Vertical and Directional Drilling in an Autonomous Miniature Drilling Rig

Master's thesis in Cybernetics and Robotics

Supervisor: Lars Struen Imsland

June 2020

Tor Arne Tjøland
Mads Sundseth

Design and Implementation of a Simulation System for Vertical and Directional Drilling in an Autonomous Miniature Drilling Rig

Master's thesis in Cybernetics and Robotics
Supervisor: Lars Struen Imsland
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics



Abstract

The vast impact of digitalization and automation of processes in multiple industries has become apparent over the past years. While research and development has evolved multiple manual processes to become autonomous, the case for autonomous oil drilling is not the same. Therefore, the Society of Petroleum Engineers (SPE) established the Drilling System Automation Technical Section (DSATS) in order to accelerate the implementations of autonomous drilling.

Initiated in 2014, the Drillbotics competition, with SPE and DSATS in the front, annually challenges students from around the world to build a physical autonomous miniature drilling rig to conquer given challenges. This year, the competition was scheduled to be held 22.-26. June, but got canceled due to the ongoing pandemic. This master's thesis has therefore undergone a change of problem statement, such that the new main goal became to create a simulation system that realistically represents the true drilling environments of the physical competition.

The idea of the simulation system is for it to be used by the next year's Drillbotics team for early control system testing. As such, the program has been implemented in MATLAB and Simulink with a compartmentalized approach, giving interchangeability of blocks for easy testing. The main blocks consist of the state dynamics, state machine, reference generator, controller, and state estimation, and may all be easily edited and tested.

To find the optimal control method for different drilling environments, there has been implemented multiple types of controllers consisting of a proportional–integral–derivative controller (PID), a model predictive controller (MPC), and a nonlinear model predictive controller (NMPC). For accurate state representation, the state dynamics have been designed and implemented close to the true dynamics of the system, containing noise and bias in both the process and simulated sensors. To combat this noise and bias, an extended Kalman filter is used. As well, it is important to ensure a safe drilling environment. As such, a state machine with 9 potential states has been implemented. Lastly, there are multiple viable paths for the drill to follow. A reference generator with multiple optional path algorithms has therefore also been implemented.

The results show that there are multiple use cases for each of the controller methods, as well as the path generation methods. They should be used based on the drilling environment, such as the given coordinate points, mechanical constraints of the physical system, and other parameters that the competition score is based on. It has been found that the simulation system is an excellent tool to test different hypotheses and theories, as it makes visualization of system response fast and easy.

Furthermore, the simulation system could be expanded upon with regards to state dynamics to better represent the true physical dynamics. As well as this, there should also be done more research on using an NMPC for path generation, as this should be better at finding the optimal path based on the mechanical constraints and score-optimization in an eventual competition.

Sammendrag

Digitalisering og automatisering av prosesser har i flere bransjer hatt stor påvirkning, særlig over de siste årene. Selv om R&D har utviklet en rekke før manuelle prosesser til å nå bli autonome, gjelder dette fortsatt ikke for oljedrilling. Dette har ført til at the Society of Petroleum Engineers (SPE) opprettet the Drilling System Automation Technical Section (DSATS) for å fremskynde implementeringen av autonom drilling.

Med SPE og DSATS i fronten ble den årlige Drillbotics-konkurransen opprettet, hvor studenter over hele verden blir utfordret til å bygge en autonom miniatyrversjon av en borerigg som skal utføre gitte utfordringer. I år skulle konkurransen opprinnelig avholdes fra 22. til 26. juni, men på grunn av den pågående pandemien, ble den kansellert. Denne masteroppgaven fikk derfor endret sin problemstilling, hvor det nye målet ble å konstruere et simuleringssystem som realistisk klarer å fremstille de fysiske utfordringene boreriggen ville møtt under konkurransen.

Ideen bak simuleringssystemet er at det skal kunne brukes av kommende års Drillbotics-lag for tidlig testing av teorier og kontrollsystemer. For å oppnå enkel testing er programmet implementert i MATLAB og Simulink med en seksjonert tilnærming, slik at deler av programmet lett kan endres. Hoveddelene består av tilstandsdynamikk, tilstandsmaskin, tilstandsestimering, referansegenerator og kontrollere, som alle enkelt kan bli endret og testet.

For å finne den optimale kontrollmetoden for ulike boremiljøer, har det blitt implementert en rekke typer kontrollere bestående av proporsjonal–integral–derivat-kontroller (PID), modell-prediktiv-kontroller (MPC) og ulineær-modell-prediktiv-kontroller (NMPC). For å oppnå realistisk simulering, har tilstandsdynamikken blitt utformet og implementert så nært som mulig den fysiske dynamikken, som også inneholder støy og bias i både prosessen og de simulerte sensorene. For å motvirke støy og bias benyttes et utvidet Kalman-filter. Det er også viktig å sikre et trygt boremiljø, så i tillegg har det blitt implementert en tilstandsmaskin med 9 potensielle tilstander. Til slutt har en referansegenerator med en rekke mulige algoritmer blitt implementert, slik at ulike referansemetoder for kontrollerne kan testes.

Resultatene viser at det er flere fordeler og ulemper med de forskjellige kontroll- og referansemetodene. Valg av disse avhenger av boremiljøet, slik som gitte koordinater, mekaniske begrensninger av det fysiske systemet og andre parametre som bestemmer utfallet av konkurransen. Det ble funnet at simuleringssystemet er et utmerket verktøy for å teste ulike hypoteser og teorier, ettersom det gjør visualisering av systemrespons raskt og enkelt.

Videre kan simuleringssystemet utvides med hensyn til tilstandsdynamikken for å representere den fysiske dynamikken bedre. Det bør også utforskes mer på bruken av NMPC som referansegenerator, da dette burde kunne finne optimal bane for drilling basert på mekaniske begrensninger og poeng-optimalisering i en eventuell konkurranse.

Preface

This master's thesis marks the end of a 5-year journey on the Cybernetics and Robotics program at NTNU. The years have presented us with an adventure through multiple disciplines, ending with the ultimate challenge of autonomous drilling in the oil industry. Although the path of the master's thesis has deviated from the original plan, it has still been a fun and rewarding ride, as we believe it offers great help to the next year's Drillbotics team.

Firstly, we would like to express our gratitude to the Norwegian University of Science and Technology (NTNU) for the great opportunity to work together as a multidisciplinary team consisting of students from both the Department of Engineering Cybernetics and the Department of Geoscience and Petroleum. It has shown us the importance of knowledge sharing, as well as team coordination for efficient results.

Secondly, we would like to thank our main supervisor, Lars S. Imsland, for his great advice and understanding during the uncertain times of the COVID-19 pandemic. The task of finding a new problem statement, as well as transitioning to it, could not be better. We would also like to thank the supervisors prior to the shutdown of NTNU; Alexey Pavlov, Tor Berge S. Gjersvik, Sigbjørn Sangesland, and Sigve Hovda for giving great feedback during the initial design of the physical drilling rig. As well as the supervisors, we would like to thank the provided engineers Noralf Vedvik and Steffen W. Moen for assisting with the practical problems that we have encountered.

Lastly, we would like to extend our gratitude to our pre-COVID-colleagues Vilde Romslo Meyer, Arman Hiwa, and Jonas Mannsverk from the Department of Geoscience and Petroleum for sharing their domain knowledge, as well as being great teammates in general. Also, a big thanks is due to our families and friends for their never-ending support, which has been a significant source of motivation during these unique times.

Contents

Nomenclature	x
List of Figures	xi
List of Tables	xvii
1 Introduction	1
1.1 Motivation	1
1.2 Original Problem Description	1
1.3 Problem Description After COVID-19	2
1.4 Thesis Outline	3
2 Control Theory	5
2.1 Coordinate Frames	5
2.1.1 Translation and Rotational Matrices	5
2.1.2 Angular Velocity	7
2.2 PID Controller	8
2.2.1 The General Dynamics of a PID Controller	8
2.2.2 Parameters in PID Control	8
2.2.3 Tuning of Controller Using Pole Placement	9
2.3 Model Predictive Control	10
2.3.1 The General Principle of Model Predictive Controller	10
2.3.2 Cost- and Reward Function in an MPC	11
2.3.3 Weighing of the Q and P Matrices	12
2.3.4 Prediction and Control Horizon	12
2.4 State Estimation with Kalman Filter	13
2.4.1 Basic Principles of the Kalman Filter	13
2.4.2 The Discrete Kalman Filter Algorithm	14
2.4.3 Defining the Q and R Matrices	15
2.4.4 Extended Kalman Filter (EKF)	15
2.5 State Calculation with Euler's Method	17
2.5.1 The General Principle of Euler's Method	17
2.5.2 Stability of Euler's Method	17
2.6 Cubic Spline Interpolation for Path Generation	18
2.7 Inertial Measurement Unit	19
2.7.1 Calculating Yaw Angle Using a Magnetometer	20
2.7.2 Calculating Roll and Pitch Using Accelerometer	20

3	Drilling Theory	21
3.1	Directional Drilling	21
3.1.1	Why Directional Drilling?	21
3.1.2	Well Path	21
3.2	Bottom Hole Assembly (BHA)	23
3.2.1	Directional Steering	23
3.2.2	Positive Displacement Motor (PDM)	24
3.3	Drill String Mechanics	26
3.3.1	Buckling	27
3.3.2	Burst	29
3.3.3	Twist-off	29
3.3.4	Pipe Bending	30
3.3.5	Fatigue	30
3.4	Calculations	31
3.4.1	Torque and RPM Calculations	31
3.4.2	Twist-off	31
3.4.3	Pipe Bending	32
3.4.4	Bit Tilt	34
4	Rig Specifications	35
4.1	NTNU Miniature Drilling Rig	35
4.2	Hoisting System	37
4.2.1	Hoisting Motor	38
4.2.2	Load Cell	39
4.3	Rotary System	41
4.3.1	Top Drive Motor	41
4.3.2	Drill Pipe	42
4.3.3	Power Section	42
4.3.4	Bent Housing	43
4.4	Sensor Card and Communication	44
5	Initial Work for the Drillbotics Competition	48
5.1	Implementation of New Sensor Cards	48
5.1.1	Soldering of Sensor Card	48
5.1.2	Soldering Connector to the J-link Adapter	49
5.1.3	Soldering USB-connection to Sensor Card	51
5.1.4	Programming the Sensor Card	52
5.2	Simulation System - Simple Simulation for Verification	52
5.2.1	Reference Coordinate Generation	53
5.2.2	Control System	54

5.2.3	State Dynamics	54
5.2.4	Simulation Results	55
6	System Description and Control Design	60
6.1	System Description	60
6.1.1	System Inputs	60
6.1.2	System Measurements	61
6.1.3	Control Objective	61
6.1.4	Coordinate Frames	61
6.2	States and State Dynamics	65
6.2.1	Orientation of Drill Bit	65
6.2.2	x -, y - and z -position of Drill bit	67
6.2.3	Weight on Bit (WOB) Dynamics	68
6.2.4	Rate of Penetration (ROP) Dynamics	68
6.2.5	Complete State Space Model	69
6.3	State Machine	70
6.3.1	Vertical Drilling	70
6.3.2	Directional Drilling	71
6.4	Reference Generation	73
6.4.1	Cubic Spline Interpolation	74
6.4.2	Circular Trajectory	74
6.4.3	NMPC for Optimal Trajectory Generation	76
6.4.4	Azimuth Calculation for Angle Reference	77
6.5	Controller Scheme	78
6.5.1	PID for Simple Operations	79
6.5.2	PID for Directional Drilling	79
6.5.3	MPC and NMPC for Directional Drilling	81
6.6	State Estimation	82
6.6.1	State Transition Using the Extended Kalman Filter(EKF)	82
7	Implementation of Simulation System	85
7.1	System Overview	85
7.2	Reference Generator	87
7.2.1	Reference Path	87
7.2.2	Angle Reference	88
7.3	Controller Scheme	89
7.3.1	PID Controllers for Simple Operations	91
7.3.2	MPC for Directional Control	92
7.3.3	NMPC for Directional Control	94
7.3.4	PID for Directional Control	95

7.4	Plant	96
7.4.1	Orientation and Position Calculation	96
7.4.2	Load Cell for WOB Measurement	97
7.4.3	Implementation of Inertial Measurement Unit in Simulink	97
7.5	Kalman Filter for State Estimation	98
7.5.1	Inputs to Kalman Filter	98
7.5.2	State Transition Function	99
7.5.3	Example of Kalman Estimated State Feedback Versus Actual States	99
7.6	State Machine and ROP Logic	99
7.6.1	States and Transitions	100
7.6.2	ROP Logic	103
7.7	Data Plotting and Drilling Visualization	103
7.7.1	General Path Plotting	103
7.7.2	Plotting of Relevant Data	104
7.7.3	Live Drilling Visualization	105
8	Results and Discussion	108
8.1	Path Generation Methods	108
8.1.1	Cubic Spline Interpolation	108
8.1.2	Circular Interpolation	109
8.1.3	NMPC for Path Generation	111
8.2	Controller Options without Noise and Bias	112
8.2.1	Controller Results for Simple Operations	112
8.2.2	MPC for Directional Control	116
8.2.3	NMPC for Directional Control	118
8.2.4	PID for Directional Control	122
8.3	Introducing Noise and Bias	124
8.3.1	Sensor Noise and Bias	124
8.3.2	Process Noise	126
8.3.3	Complete Noise and Bias for Realistic Simulation	127
8.4	Realistic Simulations with Noise and Bias	128
8.4.1	Scoring Method	128
8.4.2	Simulation with Expected Drilling Environment	129
8.4.3	ROP Failure during Drilling	138
8.5	Kalman Filtering	139
8.5.1	Observability	139
8.5.2	Kalman Performance	141
8.6	Note about Choice of Coordinate Frames	143
8.6.1	Inertial Frame	143
8.6.2	Drill Bit Frame: z -axis Out of Drill Bit bit	144

8.6.3	Drill Bit Frame: x -axis Out of Drill Bit	145
8.6.4	Control Frame as Solution	147
8.7	Discussion and Future Work	147
8.7.1	State Dynamics	147
8.7.2	Reference Generation	147
8.7.3	Controller Options	147
8.7.4	State Estimation	148
9	Conclusion and Recommendations for Upcoming Teams	149
	Bibliography	150
	Appendix	154
A	Matlab Code	154
A.1	Initializing Script	154
A.2	Kalman Model	157
A.3	NMPC Code	158
A.4	Cubic Spline Reference Trajectory	163
A.5	Circular Reference Trajectory	163
A.6	Initialize Hoisting Motor	166
A.7	Initialize Controllers	167
B	Matlab Code in Simulink Blocks	168
B.1	Angle Calculation from Accelerometer	168
B.2	Calculation of Angle Derivatives	169
B.3	Calculation of Reference Angle	170
B.4	Finding the Reference Coordinates	171
B.5	Rotation Matrix \mathbf{R}_{bc}^{Ic}	172
B.6	Calculate Coordinates to be Used for Angle Reference Calculation	173
C	Simulink Program	174
C.1	Simulink Overview	174
C.2	Reference Generator	174
C.3	ROP Logic Implementation	175
C.4	Plant Implementation	176
C.5	Implementation of Controllers	179

Nomenclature

API American Petroleum Institute.

BHA Bottom Hole Assembly.

BUR Build-Up Rate.

CL Course Length.

CO Control output.

DD Directional Drilling.

DH Downhole.

DLS Dogleg Severity.

DP Drill pipe.

DSATS Drilling System Automation Technical Section.

DTM Downhole Turbine Motor.

EKF Extended Kalman Filter.

EMM Electrical Miniature Motor.

ERD Extended Reach Drilling.

HM Hoisting motor.

HP Horsepower.

HSE Health, Safety and Environment.

IGP Department of Geoscience and Petroleum.

IMU Inertial Measurement Unit.

ITK Department of Engineering Cybernetics.

KOP Kick-Off Point.

MIMO Multiple-Input, Multiple-Output.

MPC Model Predictive Control.

NMPC Nonlinear Model Predictive Controller.

NTNU Norwegian University of Science and Technology.

OD Outer Diameter.

PCB Printed Circuit Board.

PDC Polychrystalline-Diamond Compact Bits.

PDM Positive Displacement Motor.

POI Point of Interest.

PPE Personal Protective Equipment.

PV Process variable.

RC Radius of Curvature.

RKB Rotary Kelly Bushing.

ROP Rate of Penetration.

RPM Revolutions per Minute.

RSS Rotary Steerable System.

SP Set point.

SPE Society of Petroleum Engineers.

TD Top Drive.

TVD True Vertical Depth.

WOB Weight on Bit.

List of Figures

2.1	A UAV relative to an inertial frame [5].	5
2.2	A typical PID controller scheme [8].	8
2.3	A typical response from a PID controller showing the different parameters [9].	9
2.4	Block diagram better explaining the principle of an Model Predictive Control (MPC).	11
2.5	Visualization of how the prediction and control horizon affects the response of the system [11].	13
2.6	The working principle of the Kalman filter is described with the different probability distributions [12].	14
2.7	The output after adding Gaussian white noise to a nonlinear function is no longer Gaussian. .	16
2.8	The resulting output from feeding a gaussian with a linearized nonlinear function [14]. . . .	17
2.9	Stability region of Euler’s method [15].	18
3.1	Relevant parameters when developing a well path [17].	22
3.2	Dogleg Severity parameters based on BHA configurations [17].	24
3.3	PDM Components [17].	24
3.4	Cross section of power section at different depths [17].	25
3.5	Number of lobes affect on Torque and RPM [17].	25
3.6	Bent sub with fixed angle.	26
3.7	Adjustable bend in bent sub [19].	26
3.8	Illustrates how the drill string might act as a result of overload [21].	27
3.9	Calculated twist-off torques for different trajectories and WOB.	32
3.10	Axial stresses for different values of RC , P and WOB	33
3.11	Axial stress from bending for different values of RC , compared with material yield strength.	33
4.1	The miniature rig and its components from last year.	35
4.2	Rig dimensions in operational position.	36
4.3	Rig dimensions in transport position.	37
4.4	The Hoisting motor, ball screw and load cell locations [29].	38
4.5	Hoisting system with different components labeled.	38
4.6	Load cell.	40
4.7	Load cell: TC4-AMP transducer by APE Transducer [33].	41
4.8	The rotary system of the drilling rig with its different components [29].	41
4.9	The Schneider Electric servo motor and servo drive [29].	42
4.10	3D-printed plastic version of stator and rotor.	43
4.11	Bent housing.	43
4.12	The downhole sensor sub on the top of the Bottom Hole Assembly (BHA).	44
4.13	Concept design of BHA with Positive Displacement Motor (PDM) as power section.	44
4.14	The wired downhole communication scheme [29].	45
4.15	The measureable dimensions of the Omnetics Nano connectors [36].	46
4.16	The ICM-20948 IMU used in the sensor card [37].	47

5.1	The first two trial attempts of soldering new sensor cards, and the final working third attempt.	49
5.2	The 20 pin connection scheme used for the J-link adapter.	50
5.3	The wires connected to the sensor card.	51
5.4	The different wires found inside a USB cable.	52
5.5	Overview of simple simulation system.	53
5.6	An overview of the implementation of extracting the reference angle ψ_r .	54
5.7	PI controller	54
5.8	State dynamics for the first version of the simulation system.	55
5.9	Simulation result with no saturation and no lookahead.	55
5.10	Simulation result with lookahead and no saturation.	56
5.11	Simulation result with saturation and no lookahead.	57
5.12	Problem with reference coordinate point generation.	58
5.13	Example of how lookahead distance may mitigate directional vector problem.	58
5.14	Using constant lookahead distance does not mitigate problem.	59
6.1	The relation between Inertial Measurement Unit (IMU) and the drill bit. The center of the coordinate frames are located at the center of the IMU and the center of the tip of the drill bit.	65
6.2	Added Weight on Bit (WOB) contribution for θ build.	67
6.3	If one assumes constant rotary speed of downhole power, and constant WOB, one can assume constant ROP [24].	69
6.4	State machine for vertical drilling.	70
6.6	An overview of the different states when the system is in the directional drilling mode.	71
6.5	Three of the possible states when in vertical drilling mode.	72
6.7	An example state when orient drill state transitions to new a state.	73
6.8	A reference trajectory created with cubic spline interpolation.	74
6.9	A reference trajectory created with circle interpolation.	75
6.10	Relevant parameters when developing a well path [17].	76
6.11	A reference trajectory created using Nonlinear Model Predictive Controller (NMPC).	77
6.12	Drill bit orientation and reference vector in the y-z-plane seen from above.	78
7.1	An overview of the simulation system in Simulink.	85
7.2	Selector is used for signal extraction from multidimensional input signals.	87
7.3	The main goal of the reference generator is to generate real-time coordinate point references as well as real time angle reference.	87
7.4	Creating optimal trajectory by running the NMPC algorithm once with prediction and control horizon large enough to cover the entire drilling operation.	88
7.5	The angle reference is found by using current position and closest reference point for a relative angle between current drill bit orientation and needed drill bit orientation.	89
7.6	The controller scheme consists of 4 PID controllers for the simple operations, and a directional controller that can be customized to use MPC, NMPC og PID.	90
7.7	There are 3 directional controllers to choose from: PID, NMPC, MPC.	91

7.8	The PID controller for initial orientation controls only the top drive position, and sets Hoisting motor (HM) input to 0.	91
7.9	The used MPC block from the model predictive control toolbox by MATLAB.	92
7.10	Editing default conditions for the MPC block.	93
7.11	The constraints and weights can be edited inside the MPC block.	94
7.12	Function for creating the NMPC object where nx is the number of states, ny is the number of outputs, nu is the number of manipulated variables, p and c is the prediction and control horizon and T_s is the sampling time.	95
7.13	An overview of the plant shows the load cell, IMU and state dynamics.	96
7.14	The state dynamics include white noise for realistic drilling environment.	96
7.15	The load cell uses a spring mechanic to measure WOB.	97
7.16	Implementation of the IMU in Simulink.	98
7.17	The inputs to the Extended Kalman Filter.	98
7.18	An example simulation of Kalman estimation versus actual states.	99
7.19	The implementation of the state machine has two main phases with safety sequences.	100
7.20	The safety sequence of pilot hole drilling is triggered with too low ROP measurement, which hoistens the system up and resets.	101
7.21	The safety sequence of directional drilling is triggered with too low ROP measurement, which hoistens the system up, orients the drill bit and tags rock.	102
7.22	The ROP output is based on the <code>stuck</code> and <code>start_rop</code> variables from the state machine.	103
7.23	Plotting of actual path, estimated path, reference path and reference points.	104
7.24	Example plots of relevant information needed to determine a safe and reasonable drilling environment.	105
7.25	Drilling visualization with drill bit coordinate system. The blue axis represents the directional vector straight out of the drill bit.	106
7.26	Drilling visualization with the BHA and drill bit orientation.	106
8.1	Cubic spline interpolation gives a path with changing Dogleg Severity (DLS), and thus needed bent angle of the drill bit.	108
8.2	Cubic Spline Interpolation creates steeper curve from p_2 to p_3 compared to p_1 to p_2	109
8.3	Circular interpolation gives a reference trajectory and drilled path with the same shape.	110
8.4	An error between p_2 and the drilled path of approximately $z_{diff} = 3.5\text{cm}$	110
8.5	Scaling the calculated DLS given from the circle gives perfect following of reference trajectory.	111
8.6	Excessive DLS build gives close to 180° turns of the BHA to compensate.	111
8.7	The NMPC path does not go through all points, but is generated with the physical constraints in mind.	112
8.8	The state transition plot shows that the simple operations controllers are tuned reasonably with regards to time.	113
8.9	The hoist up and tag rock controllers controls the hoisting motor position and velocity to feasible values.	113

8.10	The initial orientation controller orients the drill bit to the initial orientation reference at a reasonable pace.	114
8.11	The WOB response holds constant except of two oscillatory phases with initially starting vertical drilling and orientation before directional drilling.	115
8.12	The TD and HM inputs are reasonable with regards to physical restrictions.	116
8.13	By only controlling based on x , y and z error, the path is not followed as well as with the short term ϕ error.	117
8.14	The circular path is not easy to follow due to uneven bend in the path in the x - y -plane.	118
8.15	The cubic spline interpolation method has a more consistent bend throughout the path which renders an easier path to follow.	118
8.16	The path is held very accurately on top of the reference path.	119
8.17	The NMPC handles a normal path well without angle error weight w_ϕ	120
8.18	The NMPC is able to hold a difficult path well with $w_\phi = 1$	121
8.19	When using the difficult path and keeping all other weights the same, but $w_\phi = 0$, the NMPC starts to struggle.	121
8.20	By increasing the directional error weights w_x , w_y and w_z , the difficult path is held well without angle weight w_ϕ	122
8.21	The PID is able to control the path nicely by only regarding the angle reference.	123
8.22	The directional PID controller gives good results, but with no regards to the input constraints.	124
8.23	Using both noise and bias in the sensor creates a deviation from the path. However, the noise is filtered out as the path is smooth.	125
8.24	Increased noise makes no effect on the result, while increased bias makes the path deviate far from the reference path.	125
8.25	Process noise has a clear impact on the smoothness of the path.	126
8.26	Increased process noise shows the dramatic effect on the actual state of the system.	127
8.27	The combined process noise and measurement noise and bias gives a realistic system and satisfactory result.	128
8.28	An example resulting plot to show how score is measured.	129
8.29	The PID controller solely controls direction based on angle. Given the small error in the start due to noise, a wrong angle reference is used, which turns the drill bit away from the path.	131
8.30	The generated angle reference takes an unnecessary initial turn because of the noise in angle measurement.	131
8.31	The resulting WOB keeps itself around the nominal reference value when using PID for directional control.	132
8.32	The PID controller produces TD input that has small short term spikes, but changes from $[-0.06,0.06]$ during the directional drilling phase.	132
8.33	Using the NMPC path generation method when using PID as directional controller does not work well in this case, as the drill bit is taken underneath the reference path.	133

8.34	The resulting path when using the MPC is not that different from when using the PID directional controller.	134
8.35	When using the MPC for WOB control, the WOB is held close to the nominal value of $30N$	134
8.36	The TD input RPM compared to when using PID as directional controller has more spikes, but hovers around $[0, 0.05]$ for most of this phase.	135
8.37	The resulting path when using the NMPC is not that different from when using the PID or MPC directional controller.	136
8.38	When using the NMPC for WOB control, the WOB is at the setpoint of $30N$	136
8.39	The TD RPM input of the NMPC is more stable than when using the MPC or PID.	137
8.40	Input rate change can be penalized harder with the NMPC to reduce actuator rate change.	137
8.41	The bias helps the drill bit in getting closer to the reference coordinates.	138
8.42	The state machine does its job as the hoisting motor position and WOB values look reasonable with regards to ROP failure.	139
8.43	Error covariance with regards to position and orientation estimation.	140
8.44	Estimation error for position estimation using an Extended Kalman Filter (EKF) without biases in the orientation measurements. The estimation error keeps on growing as the simulation goes on.	141
8.45	Estimation error of angles, using an EKF without bias in the measurements. The estimation error is created by the filtering of the process noise by the Kalman filter.	141
8.46	Measured orientation angles versus estimated orientation angles obtained by using an EKF	142
8.47	Estimated position vs. actual position given in the inertial frame.	142
8.48	Choosing small values of Q renders a slow filter. In the figure it can be seen that the EKF lags behind the actual state.	143
8.49	The defined inertial frame with the stone being the grey block.	144
8.50	The defined drill bit frame with the stone being the grey block, and z -axis out of the drill bit.	144
8.51	Encountered a bug when the body frame was defined with the z^b -axis pointing along down the BHA towards the drill bit.	145
8.52	The defined drill bit frame with the stone being the grey block, and x -axis out of the drill bit.	146
8.53	The roll motion around the x -axis mitigates the previous problem of BHA rotation.	146
B.1	The Matlab-function block used to calculate θ and ψ using measurements from an accelerometer.	168
B.2	The Matlab-function block used to calculate the angle derivatives $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$	169
B.3	The Matlab Fuction Block Used to Calculate the Reference Angle Based on The Position Error.	170
B.4	Finding the Reference Coordinates Used By the Directional Controller.	171
B.5	Matlab function block used to rotate the ROP from $\{b^c\}$ to $\{I^c\}$	172
B.6	The function block is used to calculate the reference used to calculate the angle reference.	173
C.7	Overview Over the Entire Simulink Program.	174
C.8	The Reference Generation Block Used in Figure C.7.	174
C.9	The Figure Shows the Top Drive Angle Reference block used in figure C.8.	175

C.10	The rate of penetration is modeled as a binary value. If it is on or of is determined by the state output from the state machine and the jam drilling button seen in C.7.	175
C.11	The plant subsystem in figure C.7 contains state dynamics of the drill, and models for the HM and IMU.	176
C.12	The implementation of the IMU subsystem in the plant, seen in figure C.11	176
C.13	The implementation of the LoadCell/HoistingMotor subsystem in the plant, seen in figure C.11.	177
C.14	The implementation of the load cell subsystem in the HM, seen in figure C.13.	177
C.15	The implementation of the state dynamics subsystem in the plant, seen in figure C.11.	178
C.16	Simple model for pipe bend when the WOB exceeds its set point to generate a bigger pitch angle. The subsystem is found in the state dynamics shown in figure C.15.	178
C.17	The controller subsystem shown in figure C.7 contains this controller scheme. The controllers are separated to easily show which controller is active at each drilling state.	179
C.18	The vertical drilling subsystem in the controller scheme shown in figure C.17 contains a PI controller for the Top Drive (TD) RPM and a PID controller for WOB.	179
C.19	Different controller options for the directional drilling state.	180
C.20	MPC controls orientation and WOB during the directinal drilling state.	180
C.21	NMPC controls orientation and WOB during the directinal drilling state.	181
C.22	Implementation of orientation controller, used in orientation and directional drilling with the PID option. The switching value for clearing the integrator will be different depending on the drilling state.	181
C.23	Implementation of WOB controller, used in tag rock, vertical drilling, and Directinal drilling when the PID option is chosen. The switching value for clearing the integrator will be different depending on the drilling state.	182
C.24	Implementation of PID for hoisting motor position. Used in the Hoist Up state.	182

List of Tables

2.1	PID pole-placement algorithm [10]	10
3.1	Effective Length Factor, K, which depends on the end conditions of the column [22].	28
3.2	Showing test data from drilling. A constant top drive velocity is set with increasing WOB.	31
3.3	Twist-off torque ranges for different bending cases and WOB.	32
4.1	Change in dimensions for the wire connectos [36].	45
6.1	Rig actuators and their respective variables, units and modes.	60
6.2	Measured variables from the simulation system	61

1 Introduction

1.1 Motivation

Given the past years' decline in oil prices, there has been a need to extensively increase the productivity of the oil industry as a whole. The oil and gas industry is not alone, as there are several other industries working towards the same goal. One area of focus is to gradually move manual processes to become automated where this is feasible. Such a solution gives multiple economic advantages such as higher production rates, increased productivity, better quality, but also lower risk of injuries due to human error.

As petroleum resources are becoming less accessible due to lower prices and more strict environmental restrictions, there is a demand for new and innovative solutions. This includes different aspects such as digitization and automatization of processes where this is feasible. As an effort to work towards this goal, NTNU started BRU21, which is a research and innovation program in digital and automation solutions for the oil and gas industry [1].

As a big factor of automation innovation involves automation of drilling itself, the Society of Petroleum Engineers (SPE) established the Drilling System Automation Technical Section (DSATS). The purpose of this section is to accelerate the development and implementation of systems automation in the well drilling industry [2]. As a collaboration with multiple organizations with SPE and DSATS in the front, the Drillbotics competition was initiated in 2014 and has continued annually since [3]. The purpose of this competition is to create new solutions for the aforementioned problems of drilling automation.

The specific problem in the Drillbotics competition is to design and build a small drilling rig that uses sensors and control algorithms to autonomously drill a rock sample provided by SPE [3]. In previous years, NTNU finished 2nd (2017) and 1st (2018). In 2019, the rig was not received at the competition in time due to problems with customs. Despite this, the design report of the 2019 team was recognized to be the best design report from the participating universities.

1.2 Original Problem Description

As the competition was first held in 2014, the drilling goal has changed from what it initially was. The first competition guidelines revolved around focusing on vertical drilling, while in 2019, the problem was changed to directional drilling. The 2020 competition goal has been set to create a rig that can autonomously drill a directional well through three given coordinate points using downhole measurements in a feedback control system. This involves the use of multiple sensors and a sophisticated control system that utilizes the measurements to safely and reliably drill a directional well that satisfies the requirements. The competition was scheduled to be held 22.-26 of June 2020 but has been canceled due to COVID-19.

Problem statement for the 2020 Drillbotics competition

Design a rig and related equipment to autonomously drill a well, using downhole sensors, that are able to hit multiple directional targets, as quickly as possible while maintaining borehole quality and integrity of the drilling rig and drill string.

Competition objectives

Given the problem statement, the more specific competition objectives can be summarized as follows [4]

- Hit one or more targets, given X/Y coordinates and vertical depth(s).
- Drill 4" in vertical direction before kicking-off. Targets given will not exceed 30° inclination from vertical, 15° azimuth change or 10" displacement.

The mechanical design has been the main objective in the competition thus far, while the committee encourages the teams to shift the focus over to the autonomous aspect this year. A closed-loop control system is therefore a requirement and can be summarized as follows

- Drilling/survey mode switching should be automated. This entails a built-in survey interval and drill string movement.
- Calculations concerning steering, such as slide face and tool face direction, must be performed autonomously.
- Directional surveying should be made entirely autonomous.
- Rig floor display must show dogleg severity required in order to hit the target(s). The distance and direction respectively must be calculated autonomously.

Though the committee emphasized making the system autonomous this year, the mechanical design is still significant in order to maintain borehole quality, well integrity, and avoid drilling dysfunctions. Mechanical requirements include

- 1.5" bit diameter.
- Stainless steel or aluminum drill pipe with diameter 3/8" and wall thickness 0.049".

1.3 Problem Description After COVID-19

The original plan of this master's thesis was to build and implement the needed sensors and control system in order to succeed in the Drillbotics 2020 competition. Because of the COVID-19 pandemic that has been taking place, it was no longer feasible to physically work on the rig, which in turn caused the cancellation of the competition.

The scope of this master's thesis as it was defined before the COVID-19 pandemic was to successfully solve the Drillbotics 2020 problem statement. For the students from Cybernetics and Robotics, this meant to

improve the sensor card measurements, modeling of the system, and optimization of the drilling process through control system improvements. Due to the unexpected events of COVID-19, the goal has deviated to help the next years' team with their success in the competition. A working simulator in MATLAB with accurate physical representation and drag and drop features for easy testing has therefore become the main goal of this master's thesis, as this potentially can accelerate the next years' team with their solution. The problem of this master's thesis can therefore be summarized by a simulation system in MATLAB where the following is considered:

- Implement accurate state-space models to represent the physical drilling dynamics of the Drillbotics rig. This includes accurate representations of actuators, sensor card measurements, and other mechanical representations.
- Implement multiple path generation algorithms based on the given coordinate points. These paths can then be used as a reference for the control algorithms.
- Implement multiple control algorithms to follow different reference values.
- Implement a state machine for the different drilling phases. This includes the vertical and directional drilling phases, as well as handling unexpected events.
- Implement accurate state estimation for realistic measurement feedback with noise as expected from sensors such as an Inertial Measurement Unit (IMU).
- Compartmentalized simulation system with interchangeable parts for easy drag and drop edit and testing.

1.4 Thesis Outline

The thesis consists of work done with a change of problem statement in early March. As the first goal was to compete in the Drillbotics 2020 competition, and the second goal became to create a simulation system for this competition, a lot of the same theory is applied with some changes. The sections in this thesis will therefore be presented here, as well as which sections are taken from the previously written design report from phase 1 [4]. Some sections have lesser relevance to the current problem statement of creating a simulation system, but will still be included for a better picture of the physical system considered by the previous problem statement as these are closely related. The sections of the thesis are therefore presented below:

- **2 - Control Theory**

Consists of the used control theory when designing the simulation system. Most of this material is from the project thesis with some exceptions summarized by the following points:

- 2.1.2 - Angular Velocity
- 2.2.3 - Tuning of Controller Using Pole Placement

- 2.3 - Model Predictive Control
- 2.4.3 - Defining the \mathbf{Q} and \mathbf{R} matrices
- 2.4.4 - Extended Kalman filter

- **3 - Drilling Theory**

This section contains petroleum theory from the project thesis [4] that was deemed relevant to better understand the context of the simulation system. Some of the theory and calculations are important to render a safe drilling environment, but have currently not been taken account for in the implementation. Therefore the calculations based on the theory presented in this section should be accounted for in later iterations of the system and have therefore been included here.

- **4 - Rig Specifications**

Contains a detailed explanation of the physical rig, and is gotten from the project thesis [4]. The main goal of this section is to give a thorough understanding of the foundation of which the simulation system is built upon.

- **5 - Initial Work for the Drillbotics Competition**

Consists of the tangible work done before the COVID-19 shutdown, i.e initial workings on the physical sensor card, as well as a simple simulation system for verification of state dynamics and controller scheme.

- **6 - System Description and Control Design**

Consists of all design and calculations that the implementation of the simulation system is based on. Section [6.3 - State Machine] is designed as before, and is therefore gotten from the project thesis [4], while the rest is new.

- **7 - Implementation of Simulation System**

Consists of the implementation of the simulation system based on the planned design. Implementation is done in MATLAB and Simulink.

- **8 - Results and Discussion**

Includes the results of the implemented simulation system, as well as discussions around the results.

As seen, sections 2-4 are mainly from the project thesis with a few exceptions, while sections 5-8 are written after the change of problem statement, and are new without the exception of Section 6.3.

2 Control Theory

To be able to create a simulation system to accurately represent an autonomous drilling rig, control theory has to be utilized. In this section, relevant theory and how these may be utilized will be presented.

2.1 Coordinate Frames

When working with points and vectors given in different frames, it is needed to translate them into the same frame. An example of such a situation is given in Figure 2.1, where there is an inertial frame, and a UAV with measurements given in its coordinate frame. The dynamics of controlling a drilling rig has many similarities with this situation, as there are downhole measurements that need to be used to control the drill bit in a path defined in the inertial frame.

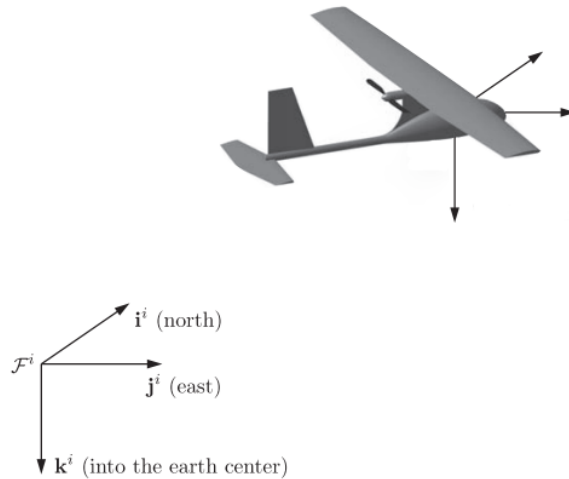


Figure 2.1: A UAV relative to an inertial frame [5].

2.1.1 Translation and Rotational Matrices

In order to transform vectors given in one coordinate frame to another, the team will use both translation and rotation matrices. Translation matrices are used to linearly move vectors from one frame to another, while rotation matrices are used to get the correct direction of the vector in the new coordinate frame. The team will base all transformation from the desired inertial frame to the drill bit using linear translation matrices, and the three rotational matrices given in equations (2.1), (2.2) and (2.3).

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad (2.2)$$

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \quad (2.3)$$

These are the essential rotation matrices in the often used ROLL-PITCH-YAW system. To get from the inertial frame to the body frame, a translation transformation is done, followed by a YAW-PITCH-ROLL rotation. Yaw is given by ψ , pitch by θ and roll by ϕ . Rotations as these, where the rotation is described as a rotation around axis by axis is called simple rotation [6]. An important observation is that these matrices are a more general class of orthonormal rotation matrices, which gives the properties given by equations (2.4), (2.5) and (2.6),

$$(\mathbf{R}_a^b)^{-1} = (\mathbf{R}_a^b)^T = \mathbf{R}_b^a \quad (2.4)$$

$$\mathbf{R}_b^c \mathbf{R}_a^b = \mathbf{R}_a^c \quad (2.5)$$

$$\det(\mathbf{R}_a^b) = 1, \quad (2.6)$$

where the subscript variable of the matrices represent the original coordinate frame, and the superscript variable is the destination coordinate frame. By using these properties, it is possible to transform vectors from inertial to body frame, as well as from body to inertial frame [6]. The full rotation around each of the axis to get from coordinate frame a to b is given by equation (2.7),

$$\mathbf{R}_b^a = \mathbf{R}_z(\psi) \mathbf{R}_y(\theta) \mathbf{R}_x(\phi), \quad (2.7)$$

where the coordinate frame a is first rotated ϕ around the x -axis and then θ and ψ around the already rotated coordinate frames. A coordinate point \mathbf{p}^b given in frame $\{b\}$, can be translated to the point \mathbf{p}^a in $\{a\}$ as seen in equation (2.8).

$$\mathbf{p}^a = \mathbf{R}_b^a \mathbf{p}^b \quad (2.8)$$

When the angle $\theta = \pm \frac{\pi}{2}$ the system ends in a Gimbal-lock. In this case the Euler angle representation will not work. Equation (2.10) shows that it is impossible to separate the roll and yaw rotations. This will also result in a singularity when calculating the angular velocities. The same can be seen when using equation (2.16) as $\frac{1}{\cos \frac{\pi}{2}}$ is undefined.

$$\mathbf{R}(\phi, \frac{\pi}{2}, \psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \quad (2.9)$$

$$= \begin{bmatrix} 0 & \sin(\phi - \psi) & \cos(\phi - \psi) \\ 0 & \cos(\phi - \psi) & -\sin(\phi - \psi) \\ -1 & 0 & 0 \end{bmatrix} \quad (2.10)$$

2.1.2 Angular Velocity

In the many cases where the orientation measurements are gotten from sensors in multiple frames that are not the inertial frame, it is not straight forward to represent the integrals of these. In the case of drilling, the angular velocities are all given in different frames as shown by the following angular velocity vectors, where frame a for example can be the inertial frame, and frame d is the frame of the Bottom Hole Assembly (BHA).

$$\boldsymbol{\omega}_{ab}^a = \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}, \boldsymbol{\omega}_{bc}^b = \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix}, \boldsymbol{\omega}_{cd}^c = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \quad (2.11)$$

By using the simple rotation matrices as defined above, the angular velocity in frame d can be defined by

$$\boldsymbol{\omega}_{ad}^d = \mathbf{R}_{x,-\phi} \mathbf{R}_{y,-\theta} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + \mathbf{R}_{x,-\phi} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \quad (2.12)$$

$$= \begin{bmatrix} -\sin \theta \dot{\psi} + \dot{\phi} \\ \sin \phi \cos \theta \dot{\psi} + \cos \phi \dot{\theta} \\ \cos \phi \cos \theta \dot{\psi} - \sin \phi \dot{\theta} \end{bmatrix}. \quad (2.13)$$

By defining the vector

$$\boldsymbol{\phi} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \quad (2.14)$$

the relation between the angular velocity in frame d and $\dot{\boldsymbol{\phi}}$ is given by

$$\boldsymbol{\omega}_{ad}^d = \mathbf{E}_d(\boldsymbol{\phi}) \dot{\boldsymbol{\phi}} = \begin{bmatrix} 1 & 0 & -\sin \theta \\ 0 & \cos \phi & \sin \phi \cos \theta \\ 0 & -\sin \phi & \cos \phi \cos \theta \end{bmatrix} \dot{\boldsymbol{\phi}}. \quad (2.15)$$

By inverting the $\mathbf{E}_d(\boldsymbol{\phi})$ matrix, one can now define an equation for $\dot{\boldsymbol{\phi}}$ as shown in equation (2.16).

$$\dot{\boldsymbol{\phi}} = \mathbf{E}_d(\boldsymbol{\phi})^{-1} \boldsymbol{\omega}_{ad}^d = \frac{1}{\cos \theta} \begin{bmatrix} \cos \theta & \sin \phi \sin \theta & \cos \phi \sin \theta \\ 0 & \cos \phi \cos \theta & -\sin \phi \cos \theta \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \boldsymbol{\omega}_{ad}^d \quad (2.16)$$

By using this equation, it is now possible to integrate the angular velocity found in frame d to calculate the current orientation of this frame.

2.2 PID Controller

When controlling the drill bit to follow a reference path, it is possible to use a proportional-integral-derivative controller, or PID in short. A PID controller is shown in Figure 2.2.

2.2.1 The General Dynamics of a PID Controller

A reference signal of the desired states is calculated and then compared with the estimated states of the drill bit. Once the error is calculated, the PID controller multiplies the proportional, integral and derivative of the error by parameters K_p , K_d and K_i respectively, which produces an input for the actuators [7]. The calculated input is therefore expressed by equation (2.17).

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}. \quad (2.17)$$

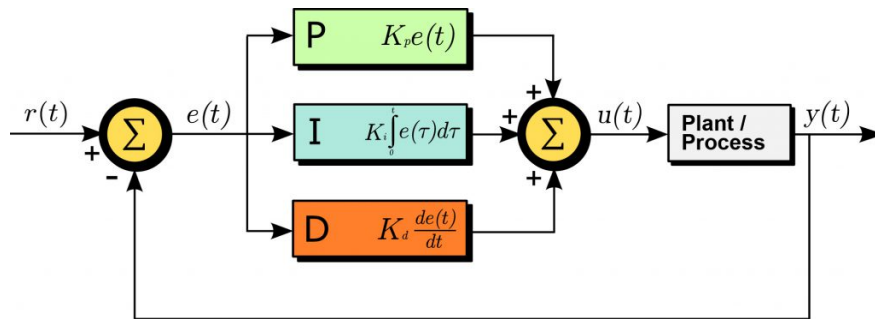


Figure 2.2: A typical PID controller scheme [8].

2.2.2 Parameters in PID Control

To better understand the different parts of the PID controller and how these affect the response of the states in the system, the different parts will be explained below together with Figure 2.3 [9].

- **P - Proportional**

The parameter K_p is used to change the impact of the proportional part of the PID controller. By changing K_p , one can determine how fast the states of the system should reach a certain reference. The rise time is the time it takes for the state to reach a certain percentage of the reference, and is what will be controlled by changing K_p .

- **I - Integral**

The parameter K_i is used to increase or decrease the integral part of the PID controller, which is used to remove the steady-state error. Usually, the proportional part of a PID controller is not enough for

the state to reach the desired reference as the error decreases. The integral part is therefore used to make sure the deviation gets canceled.

- **D - Derivative**

The parameter K_d is used to control the relevance of the derivative part of the PID controller. This part is often also referenced to as the dampening effect, as it dampens out the response. The percent of overshoot and settling time can thus be controlled by this part.

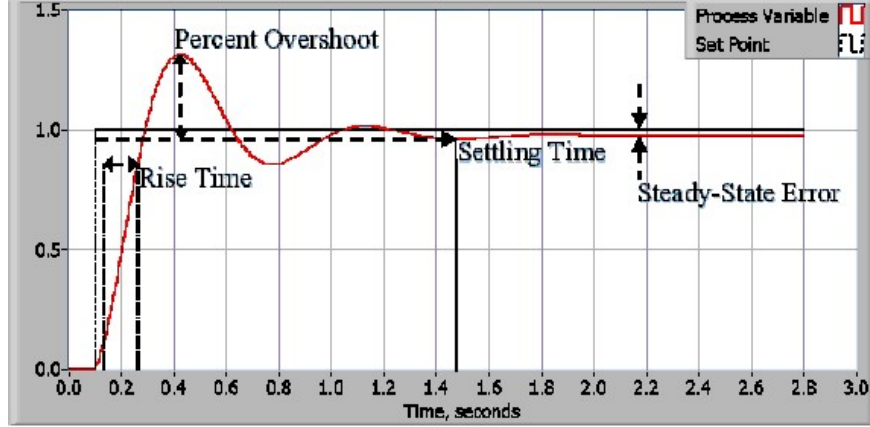


Figure 2.3: A typical response from a PID controller showing the different parameters [9].

2.2.3 Tuning of Controller Using Pole Placement

For simplicity, consider a mass-spring-damper-system controlled by a PD controller.

$$m\ddot{x} = -d\dot{x} - kx + F_{external} \quad (2.18)$$

By substituting the external forces in the mass-spring-damper system with the proportional and derivative terms from the PID-equation (2.17), the following equation is obtained

$$m\ddot{x} = -d\dot{x} - kx + K_p e + K_d \dot{e}, \quad (2.19)$$

where e can be written as $x_r - x$. With a constant reference, the equation (2.19) can be written as

$$\ddot{x} = -\frac{d + K_d}{m} \dot{x} - \frac{k + K_p}{m} x + \frac{K_p}{m} x_r. \quad (2.20)$$

From this, it is possible to obtain the transfer function

$$\frac{x}{x_r} = \frac{\frac{kp}{m}}{s^2 + \frac{d+K_d}{m}s + \frac{K_p+k}{m}} \quad (2.21)$$

Now consider the second order system written on the form

$$\frac{K}{s^2 + 2\zeta\omega_n s + \omega_n^2}. \quad (2.22)$$

By comparing the two equations, the controller gains K_p and K_d can be computed by specifying ζ and ω_n . Since ζ is expressed by both the K_d and ω_n values, the K_b has to be calculated first based on the desired bandwidth of the controller, ω_b

$$\omega_n = \frac{1}{\sqrt{1 - 2\zeta^2 + \sqrt{4\zeta^4 - 4\zeta^2 + 2}}}\omega_b \quad (2.23)$$

In many cases a critically damped system is wanted, which in this case gives

$$\omega_n = 1.56\omega_b. \quad (2.24)$$

The following steps can be followed to tune the controller [10].

Table 2.1: PID pole-placement algorithm [10]

1.	Specify bandwidth $\omega_b > 0$ and the relative damping ratio $\zeta > 0$
2.	Compute the natural frequency: ω_n
3.	Compute the P gain: $K_p = m\omega_n^2 - k$
4.	Compute the D gain: $K_d = 2\zeta\omega_n m - d$
5.	The integral effect can be added with I gain: $K_i = \frac{\omega_n}{10} K_p$

2.3 Model Predictive Control

Instead of using simple independent PID-controllers for the multiple control objectives, it might be a better solution to use Model Predictive Control (MPC). One big advantage of using such a controller scheme, is that it can be hard to tune multiple PID-controllers with an assumption of independence. Since the directional controller in the PID case will be affected by tuning the WOB controller, tuning can be hard.

2.3.1 The General Principle of Model Predictive Controller

Implementing an MPC requires a reasonably accurate model and measurements to be able to predict the future outputs of the system. If this is the case, a Multiple-Input, Multiple-Output (MIMO)-system can be controlled with regards to constraints both on the input and the output. The inputs are calculated based on the predicted outputs. Using an MPC while having an inaccurate model can on the contrary make matters worse. The objectives of an MPC can therefore be summarized with the following points:

1. Prevent violation of input and output constraints
2. Drive some output variables to their optimal set points while maintaining other outputs within specified ranges.

3. Prevent excessive change of input variables.
4. Control as many process variables as possible when a sensor or actuator is not available.

A block diagram of how an MPC works can be seen in Figure 2.4. As seen, the MPC takes the measured output from the plant, as well as the reference values for the different states at a time step t . At this point, the MPC tests control actions by putting them into a linear model describing the plant. The model then outputs predicted states, which is then fed back to the optimizer that tries new control actions. Based on the optimization, the MPC generates manipulated variables as control inputs for the system, and the process repeats itself.

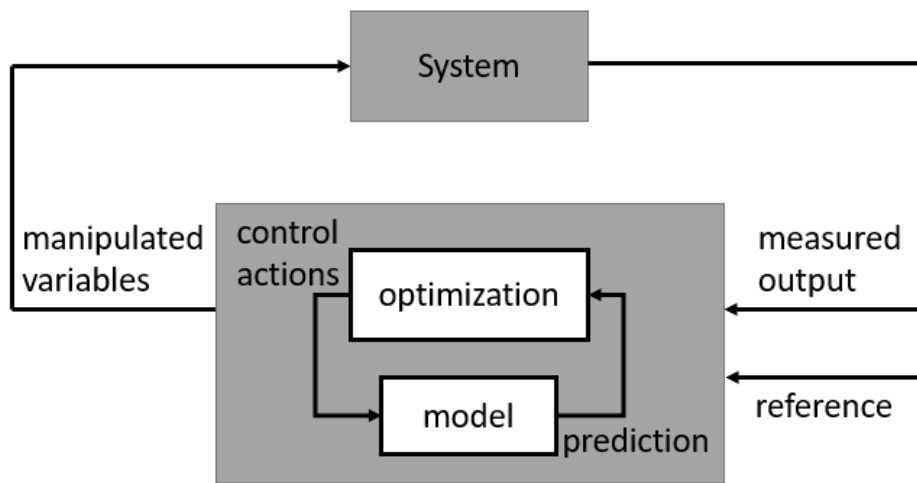


Figure 2.4: Block diagram better explaining the principle of an MPC.

2.3.2 Cost- and Reward Function in an MPC

The MPC uses a model of the system to make a prediction about future plant output behavior. To make sure the predicted outputs follow the desired reference, the MPC uses an optimizer. The MPC uses the plants model to predict the future P steps, referred to as the prediction horizon. Simulations are done to find the best possible path closest to the reference. The simulations are done in a systematic order and the best path is chosen by maximizing a reward function or minimizing a cost function. An example of such a function is shown in equation (2.25)

$$J = \sum_{i=1}^p \mathbf{e}_{k+i}^T \mathbf{Q} \mathbf{e}_{k+i} + \sum_{i=0}^{p-1} \Delta \mathbf{u}_{k+i}^T \mathbf{P} \Delta \mathbf{u}_{k+i}, \quad (2.25)$$

where \mathbf{e} is the error vector, $\Delta \mathbf{u}$ is the input increments and \mathbf{Q} and \mathbf{P} are the weights on how much the cost function will take the states and inputs into consideration. By minimizing this particular cost function, the states of the system will tend to the reference and at the same time do this with small increments on input.

This is desirable if the goal is to for example control the heading of a commercial airplane to keep the ride as comfortable as possible. By minimizing the cost function J with subject to some constraints, the MPC makes sure that boundaries with regards to the states and inputs are not exceeded.

2.3.3 Weighing of the Q and P Matrices

The \mathbf{Q} and \mathbf{P} matrices used in the cost function defined in equation (2.25) describes how much penalization one should give to an error in state and penalization of using higher inputs. \mathbf{Q} is a positive-definite matrix, while \mathbf{P} is a positive semi-definite matrix. Usually, they are both diagonal matrices with positive diagonal elements, with the value of the diagonal elements representing how much to penalize the individual states or input and input changes. An example where one has three states given by the state vector $\mathbf{x} = [p, q, r]^T$ and possible inputs $\mathbf{u} = [u_1, u_2]$ with cost matrices

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{P} = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}, \quad (2.26)$$

an error in state q will be penalized with a factor of 2 higher than states p and r , while a change in the input u_2 will cost a factor of 5 more than an input change in u_1 .

2.3.4 Prediction and Control Horizon

The MPC predicts the states and the optimal sequence of inputs over the prediction horizon. At the current time step, the control algorithm only applies the first input from the calculated sequence and disregards the rest. After applying the input, the system is taken to its new state at a new time step. This state can deviate from the earlier prediction due to unknown disturbances. The same procedure as described above is done for the next time step, with a shifted prediction horizon. MPC is called an online control algorithm because of the calculations done at each time step in real-time.

The prediction horizon, as mentioned above, decides how far in the future the MPC predicts the states. Choosing a large prediction horizon can result in a slow system due to limitations in computational power. A large prediction horizon P means that calculations has to be done from the current step t until $t = P$. If a control input is needed within a time t_d , the computation time of each step can not exceed this. Limiting the prediction horizon is therefore necessary, and should be evaluated based on the t_d constraint.

At the same time as the prediction horizon can not be too large, it should not be too small either, as it then will not be able to cover for the most significant dynamics of the system. An example is if an MPC is used for controlling the speed of a car, and the prediction horizon P only predicts $2m$ ahead of the car, a traffic light will not be taken account for before it is too late.

Choosing an excessively large control horizon can result in a very aggressive system. This is because the MPC then finds the most optimal inputs to get the current state to the end state as fast as possible. An example is if the end state of a car is 100m ahead of the current position state, using a large control horizon calculates the most optimal control inputs to get the car to this end state as fast as possible. Using a too low control horizon can also render the system too slow, as it now calculates the optimal inputs for the states that are close to the current state instead of accounting for the states further ahead. After the control horizon, the MPC holds a constant control input for the rest of the prediction. A visualization of how the prediction and control horizon affects the system can be seen in Figure 2.5.

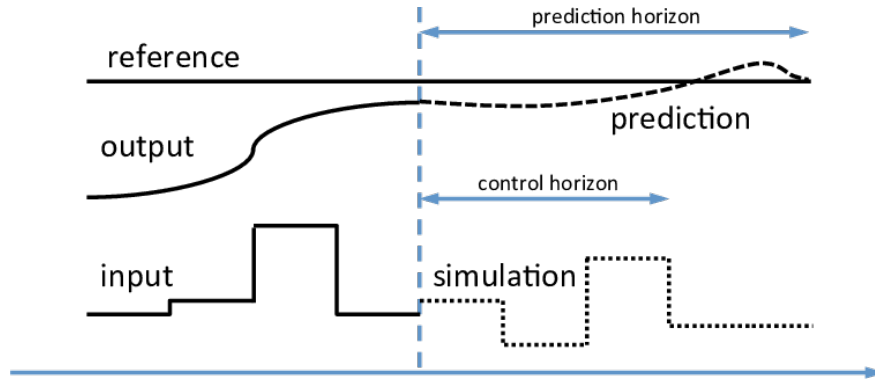


Figure 2.5: Visualization of how the prediction and control horizon affects the response of the system [11].

2.4 State Estimation with Kalman Filter

A Kalman filter is an optimal state estimator that estimates states of a linear or a nonlinear system based on measurements and a physical model of the system. It is used in combination with sensors since it can estimate unmeasured states as well as remove noise from the measurements. In the case of a temporary loss of measurements, the Kalman filter can also work as a state predictor, often also called dead reckoning [10]. In the case of using a Kalman filter in autonomous drilling, it is well suited to both estimate the position of the drill bit based on other measurements, as well as removing measurement noise and noise from the high amount of vibrations.

2.4.1 Basic Principles of the Kalman Filter

The working principles of a Kalman filter can be illustrated by an example of vehicle position estimation. The initial estimate is given in Figure 2.6, and is denoted as $\hat{\mathbf{x}}_{k-1}$. The car can be anywhere inside this curve, but it is expected that the car will be at the mean of the curve with the highest probability. To estimate the next position of the car, a prediction is done to get a prior estimate $\hat{\mathbf{x}}_k^-$. In Figure 2.6, it can be seen that the prior estimate is not very accurate. The error covariance matrix \mathbf{P}_k^- translates to uncertainty in predicting new states solely based on the state-space model. By combining the uncertain prior estimate with the system measurements y_k , a better estimate can be found. This is called the posterior estimate $\hat{\mathbf{x}}_k$, and is found by first calculating the Kalman gain \mathbf{K}_k , which then is used to minimize the error covariance for the posterior

estimate $\hat{\mathbf{x}}_k$.

When driving a car, one would be able to estimate the position of the car by integrating the angular velocity and size of the wheels. Once the car drives in holes, or if it is slippery, the wheels might slip or spin, which would be interpreted as a positional movement by the model $\hat{\mathbf{x}}_k^-$. By now combining this estimation with the measured position of the car with for example a GPS, which also has measurement noise, one would be able to better estimate the real position of the car, $\hat{\mathbf{x}}_k$.

If there are no disturbances in the measurements, and there is only one state that can be measured perfectly, $C = 1$, which gives a Kalman gain of $\mathbf{K}_k = 1$. In this case, the posterior estimate will be equal to the measurement $\hat{\mathbf{x}}_k = \mathbf{y}_k$. On the other hand, if one has a perfect model of the system, the prediction is perfect, which gives a prior error covariance of $\mathbf{P}^- = 0$, which gives a Kalman gain of $\mathbf{K}_k = 0$. This will, in turn, produce a posterior estimate which is only dependent on the prediction $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^-$. This illustrates how the Kalman filter is able to combine the certainties of the predictions and measurements to output the optimal estimate for the system.

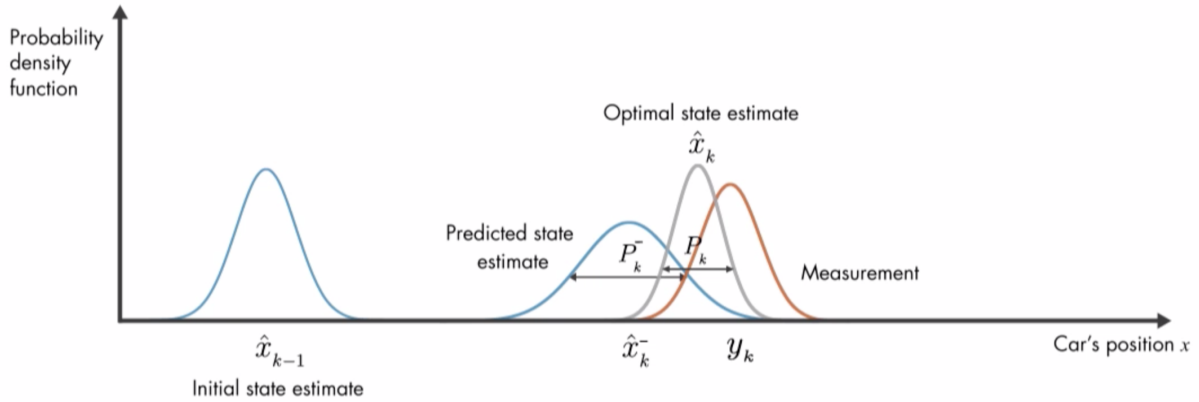


Figure 2.6: The working principle of the Kalman filter is described with the different probability distributions [12].

2.4.2 The Discrete Kalman Filter Algorithm

If the general state dynamics are defined as the following

$$\mathbf{x}[k+1] = \mathbf{A}_d \mathbf{x}[k] + \mathbf{B}_d \mathbf{u} + \mathbf{w}[k] \quad (2.27)$$

$$\mathbf{y} = \mathbf{C} \mathbf{x} + \mathbf{v}[k], \quad (2.28)$$

the discrete Kalman filter algorithm initializes as follows

$$\hat{\mathbf{x}}^-(0) = \mathbf{x}_0 \quad (2.29)$$

$$\mathbf{P}^-(0) = E[(\mathbf{x}(0) - \hat{\mathbf{x}}(0))(\mathbf{x}(0) - \hat{\mathbf{x}}(0))^T] \quad (2.30)$$

$$\mathbf{Q} = E [\mathbf{w}\mathbf{w}^T] , \mathbf{R} = E [\mathbf{v}\mathbf{v}^T] . \quad (2.31)$$

The state estimation algorithm then proceeds by doing the following steps at each time step:

$$\mathbf{K}[k] = \mathbf{P}^-[k]\mathbf{C}^T (\mathbf{C}\mathbf{P}^-[k]\mathbf{C}^T + \mathbf{R})^{-1} \quad (2.32)$$

$$\hat{\mathbf{x}}[k] = \hat{\mathbf{x}}^-[k] + \mathbf{K}[k] (\mathbf{y}[k] - \mathbf{C}\hat{\mathbf{x}}^-[k]) \quad (2.33)$$

$$\mathbf{P}[k] = (\mathbf{I} - \mathbf{K}[k]\mathbf{C}) \mathbf{P}^-[k] (\mathbf{I} - \mathbf{K}[k]\mathbf{C})^T + \mathbf{K}[k]\mathbf{R}\mathbf{K}[k]^T \quad (2.34)$$

$$\hat{\mathbf{x}}^-[k+1] = \mathbf{A}_d\hat{\mathbf{x}}[k] + \mathbf{B}_d u[k] \quad (2.35)$$

$$\mathbf{P}^-[k+1] = \mathbf{P}[k]\mathbf{A}_d\mathbf{P}^T[k] + \mathbf{Q}. \quad (2.36)$$

At each iteration, the Kalman filter algorithm uses the prior estimate $\hat{\mathbf{x}}^-[k]$ and the prior error covariance matrix $\mathbf{P}^-[k]$ to calculate the new Kalman gain $\mathbf{K}[k]$. The new Kalman gain is then used to estimate the new states and the new error covariances, $\hat{\mathbf{x}}[k]$ and $\mathbf{P}[k]$. The final two steps are to calculate the next prior estimate of the state and the prior error covariance, $\hat{\mathbf{x}}^-[k+1]$ and $\mathbf{P}^-[k+1]$.

2.4.3 Defining the Q and R Matrices

The covariance matrix of the measurement noise is defined by the \mathbf{R} matrix. The covariance of the measurement noise is often given by the manufacturer of the given measurement unit. It is also possible to tune the values in the \mathbf{R} matrix and analyze the results to determine good values for \mathbf{R} . The covariance for process noise, \mathbf{Q} , is however not that easily found. It can be used to tune how much the Kalman filter should rely on the state-space model and the measurements. With high values in \mathbf{Q} , the uncertainty of the prediction is increased, extending the range of where the true state vector lies. This means that with low \mathbf{Q} values, there is more computation needed which renders a slower system, while the measurement noise gets better filtering. Both the process noise and measurement noise is assumed to be zero-mean Gaussian [13][10].

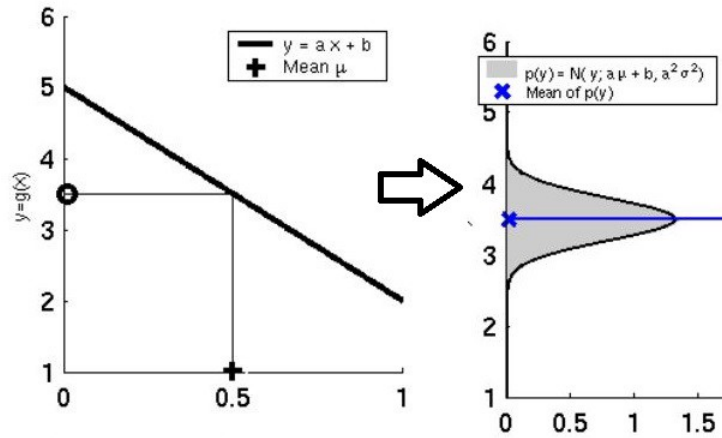
2.4.4 Extended Kalman Filter (EKF)

An important feature of the Kalman filter, is that it only works with linear systems. The reason why, is that when Gaussian white noise is added, the output will still be Gaussian when the system is linear. An example of a linear system can be given on the form

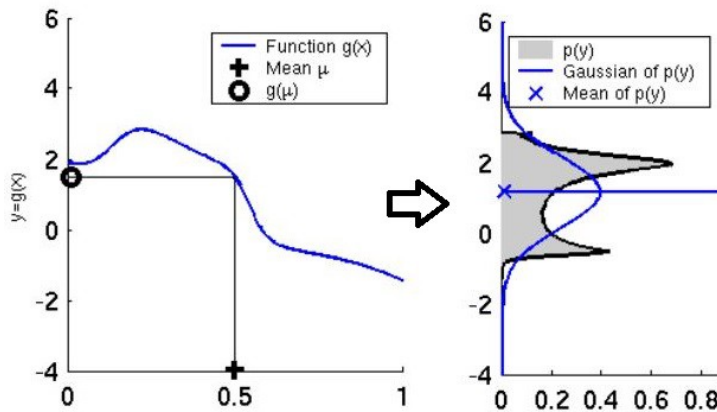
$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t) + \mathbf{w}(t) \quad (2.37)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{v}(t). \quad (2.38)$$

When Gaussian white noise is added to a nonlinear system, the output is no longer Gaussian, which is a critical assumption for the Kalman filter to work. The result of feeding Gaussian white noise with a linear and nonlinear function can be seen in Figure 2.7.



(a) Gaussian white noise added to a linear function [14]



(b) Gaussian white noise added to a nonlinear function [14]

Figure 2.7: The output after adding Gaussian white noise to a nonlinear function is no longer Gaussian.

As seen from the figure, the output from feeding Gaussian white noise with a nonlinear system, no longer outputs a Gaussian. An example of such a nonlinear system can be given on the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{w}(t) \quad (2.39)$$

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t)) + \mathbf{v}(t). \quad (2.40)$$

Since a Kalman filter only works on linear systems, an extended Kalman filter is often used when nonlinearity is in the picture. The extended Kalman filter will linearize the nonlinear function around a state, for example, $\sin x$ at $x = 0$ by using Taylor Series. An example of a resulting output from feeding a Gaussian together with a linearized function is given in Figure 2.8.

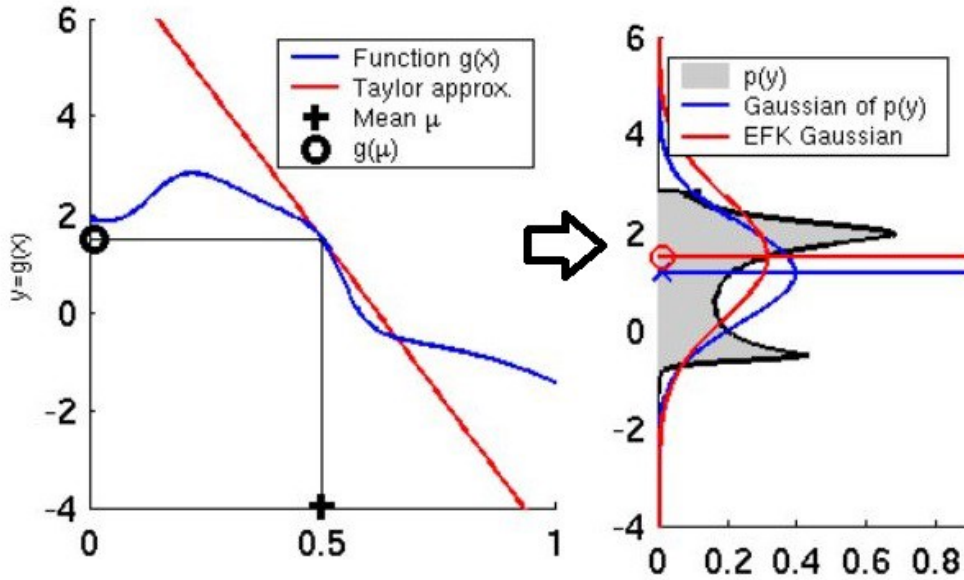


Figure 2.8: The resulting output from feeding a gaussian with a linearized nonlinear function [14].

2.5 State Calculation with Euler's Method

When calculating state variables based on their derivatives, Euler's method is often used. In the case of drilling, it is a good way to estimate the current orientation and position of the BHA by using Euler's method and the derivative of these states.

2.5.1 The General Principle of Euler's Method

The state variables evolve according to equation (2.41),

$$y_{n+1} = y_n + hf(y_n, t_n), \quad (2.41)$$

where y_{n+1} is the new calculated state, y_n is the previous state, h is a defined step length, and $f(y_n, t_n)$ is the time-derivative of the function $y(t)$. By using Euler's method, the team will be able to integrate the time derivative states measured by the bottom hole sensors, which in turn gives the positional states.

2.5.2 Stability of Euler's Method

An important consideration to make when using Euler's method is the step length. It is important to choose this variable so that the system is stable. The stability function for Euler's method is given by

$$R(h\lambda) = 1 + h\lambda, \quad (2.42)$$

where λ is the eigenvalue of the system. Stability of Euler's method is then achieved as long as

$$|R(h\lambda)| \leq 1, \quad (2.43)$$

which gives stability when the following criteria is met

$$h \leq -\frac{2}{\lambda}. \quad (2.44)$$

The stability region is therefore given by a circle of radius $r = 1$, with center in $Re = -1$, and $Im = 0$, as seen in Figure 2.9. This theory is based on [6].

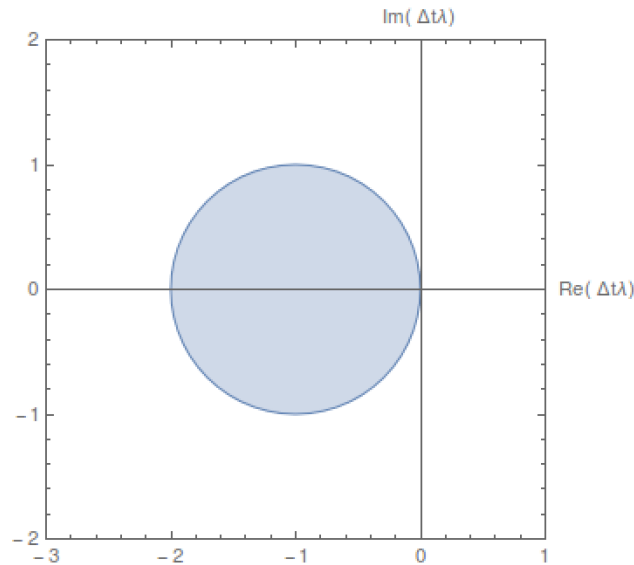


Figure 2.9: Stability region of Euler's method [15].

2.6 Cubic Spline Interpolation for Path Generation

Cubic spline interpolation is a method used to create a smooth function between some given coordinate points. By using cubic spline interpolation, it is possible to avoid oscillations that may occur when trying high polynomial methods. This is also called Runge's phenomenon. The method is based on solving a set of equations based on the number of points given. Given $N + 1$ points, the algorithm uses N equations given by the general form shown in equation (2.45). The equations created are valid for two following points, i.e. the first equation C_1 is valid for p_0 and p_2 , and C_2 is valid for p_1 and p_2 [16].

$$C_i(x) = a_i + b_i x + c_i x^2 + d_i x^3 \quad (2.45)$$

Given three points $[x_0, y_0]$, $[x_1, y_1]$ and $[x_2, y_2]$, the method starts by creating two equations based on equation (2.45). This gives the following two equations

$$C_1(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3 \quad (2.46)$$

$$C_2(x) = a_2 + b_2x + c_2x^2 + d_2x^3. \quad (2.47)$$

The first equation is valid for the left two points, and the second is valid for the right two points. The equation will join each other in x_1 , which means that the equations are equal to each other in that point. This also holds for their derivatives and double derivatives. For a natural cubic spline, the endpoints of the second derivative is set to be zero. The coefficients can be found by solving the following system of equations

$$C_1(x_0) = y_0 \quad (2.48)$$

$$C_1(x_1) = y_1 \quad (2.49)$$

$$C_2(x_1) = y_1 \quad (2.50)$$

$$C_2(x_2) = y_2 \quad (2.51)$$

$$C_1'(x_1) = C_2'(x_1) \quad (2.52)$$

$$C_1''(x_1) = C_2''(x_1) \quad (2.53)$$

$$C_1''(x_0) = 0 \quad (2.54)$$

$$C_2''(x_3) = 0. \quad (2.55)$$

After solving for the coefficients, the functions between the points will be given by equation (2.56).

$$S(x) = \begin{cases} C_1(x) & \text{for } x_0 \leq x \leq x_1 \\ C_2(x) & \text{for } x_1 \leq x \leq x_2 \end{cases} \quad (2.56)$$

2.7 Inertial Measurement Unit

An Inertial Measurement Unit (IMU) consists of three different measurement units. Angular rate is measured by a gyroscope, acceleration by an accelerometer, and heading by using a magnetometer. The measurements obtained can be used to calculate position, velocity and orientation. These calculations might be inaccurate due to the fact that computations are done by integrating measurements containing bias, which can cause drift. If the IMU is mounted at the center of the body frame, $\{b\}$, the measurements can be expressed as [10]

$$\mathbf{a}_{imu}^b = \mathbf{R}_n^b(\Theta)(\dot{\mathbf{v}}_{nb}^b - \mathbf{g}^n) + \mathbf{b}_{acc}^b + \mathbf{w}_{acc}^b \quad (2.57)$$

$$\boldsymbol{\omega}_{imu}^b = \boldsymbol{\omega}_{nb}^b + \mathbf{b}_{gyro}^b + \mathbf{w}_{gyro}^b \quad (2.58)$$

$$\mathbf{m}_{imu}^b = \mathbf{R}_n^b(\Theta)\mathbf{m}^n + \mathbf{b}_{mag}^b + \mathbf{w}_{mag}^b, \quad (2.59)$$

where $\Theta = [\phi, \theta, \psi]$ is a vector of Euler angles and $\mathbf{R}_n^b(\Theta)$ is the rotation matrix between the inertial frame and body frame. The accelerometer and gyro bias are denoted as b_{acc}^b and b_{gyro}^b , while b_{mag}^b is the local magnetic disturbance, and w_{acc}^b , w_{gyro}^b and w_{mag}^b are Gaussian measurement noises. This IMU measurement model is only valid for low-speed operations.

2.7.1 Calculating Yaw Angle Using a Magnetometer

When ϕ and θ are non-zero, the angle ψ_m can be calculated by using equations (2.60) and (2.61). First, the magnetometer measurements gets converted to the horizontal plane by using equation (2.60).

$$\begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} = \mathbf{R}_{y,\theta} \mathbf{R}_{x,\psi} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} \quad (2.60)$$

The sign of h_x and h_y must be taken into account when computing the heading ψ . This can be done by using equation (2.61).

$$\psi_m \begin{cases} 180^\circ - \frac{180^\circ}{\pi} \arctan \frac{h_y}{h_x} & \text{if } h_x < 0 \\ -\frac{180^\circ}{\pi} \arctan \frac{h_y}{h_x} & \text{if } h_x > 0, h_y < 0 \\ 360^\circ - \frac{180^\circ}{\pi} \arctan \frac{h_y}{h_x} & \text{if } h_x > 0, h_y > 0 \\ 90^\circ & \text{if } h_x = 0, h_y < 0 \\ 270^\circ & \text{if } h_x = 0, h_y > 0 \end{cases} \quad (2.61)$$

2.7.2 Calculating Roll and Pitch Using Accelerometer

To calculate the pitch angle θ and roll angle ϕ , it must be assumed that $\dot{\mathbf{v}}_n^b = 0$. This assumption can not be made in every system. For aircrafts, this assumption can not be made, since they perform high acceleration movements lasting over time. For ships and directional drilling, this assumption works quite well since the processes are slow. There is also an assumption that the biases \mathbf{b}_{acc}^b is removed by calibration, and the measurement noise \mathbf{w}_{acc}^b is removed by filtering the signal. Equation (2.57) can be simplified to

$$\mathbf{a}_{imu}^b \approx -\mathbf{R}_n^b(\Theta) \mathbf{g}^n. \quad (2.62)$$

Using equation (2.62), it is possible to express roll and pitch as [10]

$$\phi \approx \arctan \left(\frac{a_y}{a_z} \right) \quad (2.63)$$

$$\theta \approx -\arctan \left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}} \right). \quad (2.64)$$

3 Drilling Theory

There are a lot of theories behind directional drilling in the oil industry. Therefore, to be able to simulate such a system, it is important to get into the details of how directional drilling works, and the mechanics behind it. The relevant parts for this has been presented in the previous Drillbotics design report, and will be included here with some modifications.

3.1 Directional Drilling

Directional Drilling (DD) was first introduced as an objective in last year's competition. This year, in addition to inclination, it goes a step further by implementing change in azimuth. In this section, main applications of DD as well as theory regarding the well path will be covered.

3.1.1 Why Directional Drilling?

DD is applied if the reservoir is hard to access or in cases of limitations at the surface. Some applications of DD include [17]:

- Drilling a secondary well from an existing wellbore, usually referred to as sidetracking.
- Drilling multiple wells from the same offshore platform.
- Avoiding challenging formations or geological structures, such as salt domes.
- Geo-Steering.
- Increasing the drainage area, as the reservoir lateral length is usually significantly greater than the vertical length.

3.1.2 Well Path

Competing teams will be given a set of X/Y coordinates and vertical depths during the competition, where each team will be scored based on hitting accuracy. The control system must, therefore, be able to understand and handle azimuth and inclination change. Briefly defining these two [17]:

- Azimuth is the compass direction of a directional survey, more specifically the angle between the well path and North axis measured clockwise from North in the plane view.
- Inclination is irrespective of the compass direction and is the deviation from the vertical at a certain point. More specifically it is the angle between a vertical line and a tangent to the well path.

A known fixed point is the reference for measurements and well location during drilling. The coordinates $\{X, Y, Z\}$ of the reference point are set to $\{0, 0, 0\}$, which is equivalent to $\{0^\circ \text{ North}, 0^\circ \text{ East}, 0 \text{ m TVD}\}$. All well paths are developed from the reference point to the desired target(s), beginning with drilling vertically until reaching a formation that can withstand the extra strain applied from a deviated section. Kick-Off

Point (KOP) is the measured depth where the drill string starts building angle if a change in compass direction is desired. This is referred to as turn rate. A 3D curved well path is created, however this curve cannot separately be composed into azimuth or inclination due to the turning and building. To avoid getting inaccurate coordinates it should be presented as dogleg angle (ϕ) or Dogleg Severity (DLS). $DLS[^\circ/m]$ is predicted using the following equation:

$$DLS = \frac{\phi}{CL} \quad (3.1)$$

Where $\phi[^\circ]$ is the dogleg angle and $CL[m]$ is the course length. Figure 3.1a illustrates where dogleg angle (ϕ) is found and Figure 3.1b gives the parameters for calculating the Course Length (CL).

$$CL = \frac{RC\pi(I_2 - I_1)}{180} \quad (3.2)$$

$RC[m]$ is the radius of curvature and can be calculated for inclination and azimuth angle separately.

$$RC_I = \frac{(180)(30)}{\pi B}, \quad (3.3)$$

$$RC_A = \frac{(180)(30)}{\pi T}$$

Here B is build-up rate and T is turn rate. Figure 3.1 illustrates important parameters when developing a well path, including inclination, azimuth, and dogleg angle (ϕ).

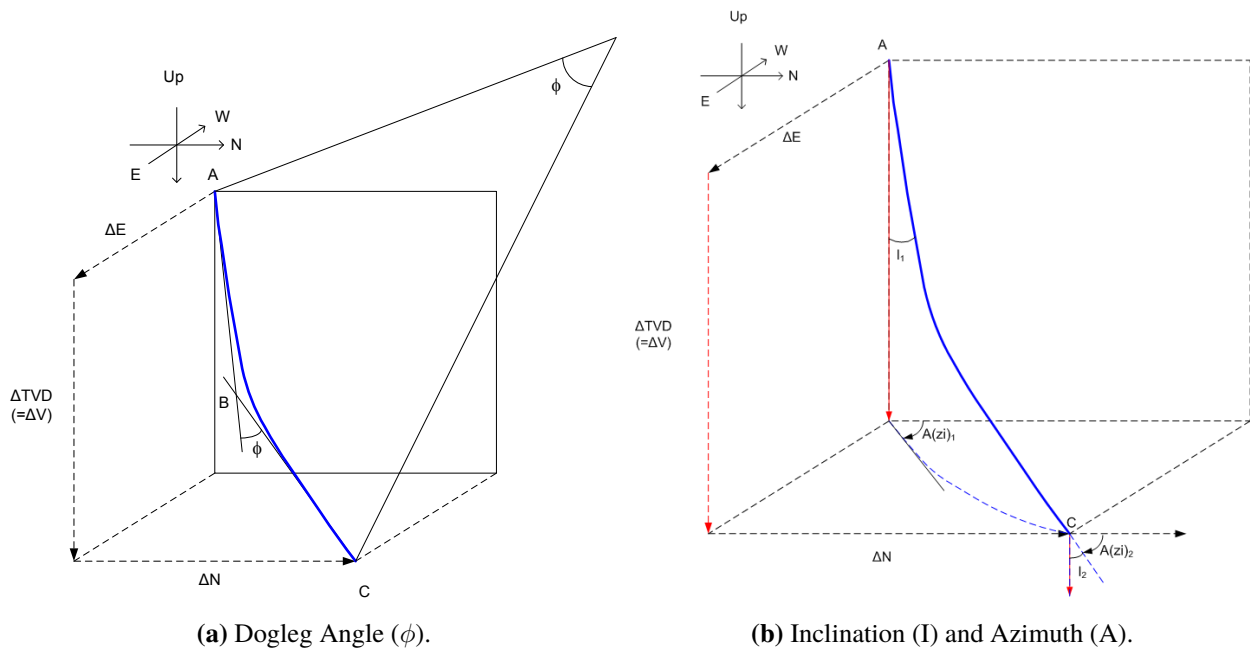


Figure 3.1: Relevant parameters when developing a well path [17].

3.2 Bottom Hole Assembly (BHA)

The Bottom Hole Assembly (BHA) design is one of the most decisive factors whether the solution proposed is mechanically sustainable. The basis for further discussion and final proposed design in regards to the BHA is presented in this section.

3.2.1 Directional Steering

The trajectory of the wellbore is affected by the BHA, therefore the intention when designing the BHA is to obtain directional control to foremost match the trajectory planned. To acquire direction, the most commonly used BHAs include [17]:

- Traditional assemblies
- Steerable motor assemblies
- Rotary Steerable System (RSS)

Traditional assemblies are usually a straight motor in combination with a bent sub. This solution carries out some restrictions though, as the bit depends on a mud motor to cut the formation because the drill string itself cannot rotate, ultimately limiting its ability to create curvature. For these reasons, the traditional assemblies are only applicable in cases with larger hole sizes [18].

Directional control evolved significantly with steerable motor assemblies, which consist of a mud motor together with a bent sub or bent housing. Compared to the traditional assemblies, this solution is much more sustainable and versatile emphasized with its ability to kick off and build angle, providing accurate directional control and its ability to drill tangent sections. This technology is often utilized with vigorous parameters to increase drilling performance in challenging drilling environments.

3.2.1.1 Dogleg Severity Based on BHA Configuration

Equation (3.1) is one way to predict DLS, there is however an alternative based on the BHA configuration [17]:

$$DLS = \frac{2\theta}{L_1 + L_2} \quad (3.4)$$

Where θ [°] is the bit tilt, L_1 [m] is the distance from the motor stabilizer to the bend and L_2 [m] is the distance from the bend to the bit; all parameters are illustrated in Figure 3.2.

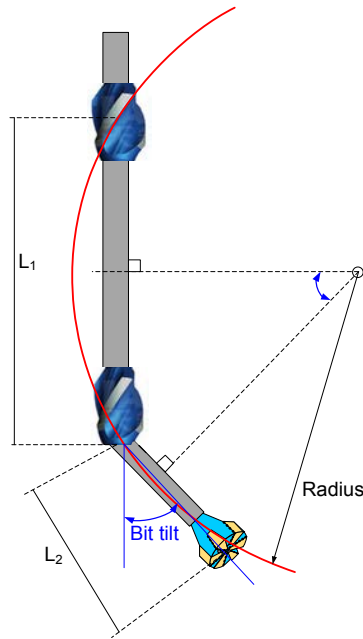


Figure 3.2: Dolog Severity parameters based on BHA configurations [17].

3.2.2 Positive Displacement Motor (PDM)

As mentioned, a steerable motor assembly is made up of a mud motor and a bent housing or bent sub. There are two types of downhole mud motors, respectively Positive Displacement Motor (PDM) and Downhole Turbine Motor (DTM), of them PDM is by far most common. PDM was first introduced in the late 1950s, and has since improved directional drilling applications greatly. A steerable drilling system is required to manage both inclination and azimuth change, which is the main objective of this year's competition. Components making up a standard PDM are respectively a power section, an adjustable bend, and a bearing section. These components are illustrated in Figure 3.3, and their function and applicability will be discussed below.

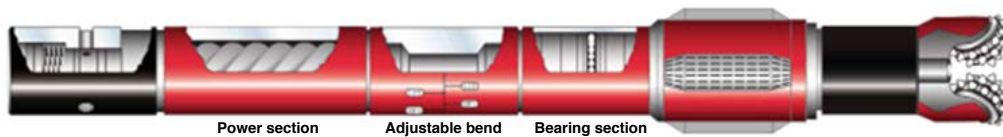


Figure 3.3: PDM Components [17].

Power Section

The power section is based on the Moineau principle and generates mechanical energy from hydraulic energy making the driveshaft rotate. The bit is connected to the driveshaft, therefore the bit Revolutions per Minute (RPM) will depend on motor speed and drill string rotation. The main components, rotor and stator, have similar profiles, except for the rotor having one less lobe as shown in Figure 3.5. Moineau principle states that a helical rotor will rotate eccentrically if the stator has more lobes than the rotor, illustrated in Figure 3.4.

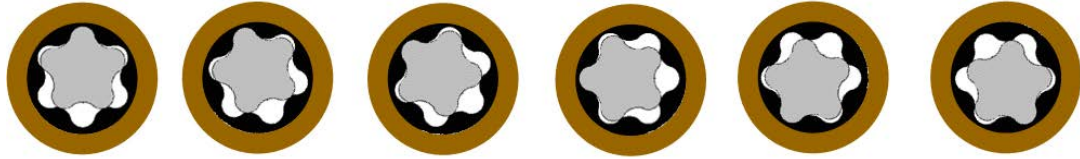


Figure 3.4: Cross section of power section at different depths [17].

The drilling fluid circulating through the motor shall have high velocity, therefore the rotor should be manufactured using corrosion-resistant stainless-steel to reduce abrasion and friction. The material of the stator should preferably be of steel in combination with an elastomer or rubber lining. A total number of lobes affects torque and RPM, with the general idea being that torque is increasing with the number of lobes present, while RPM decreases as shown in Figure 3.5.

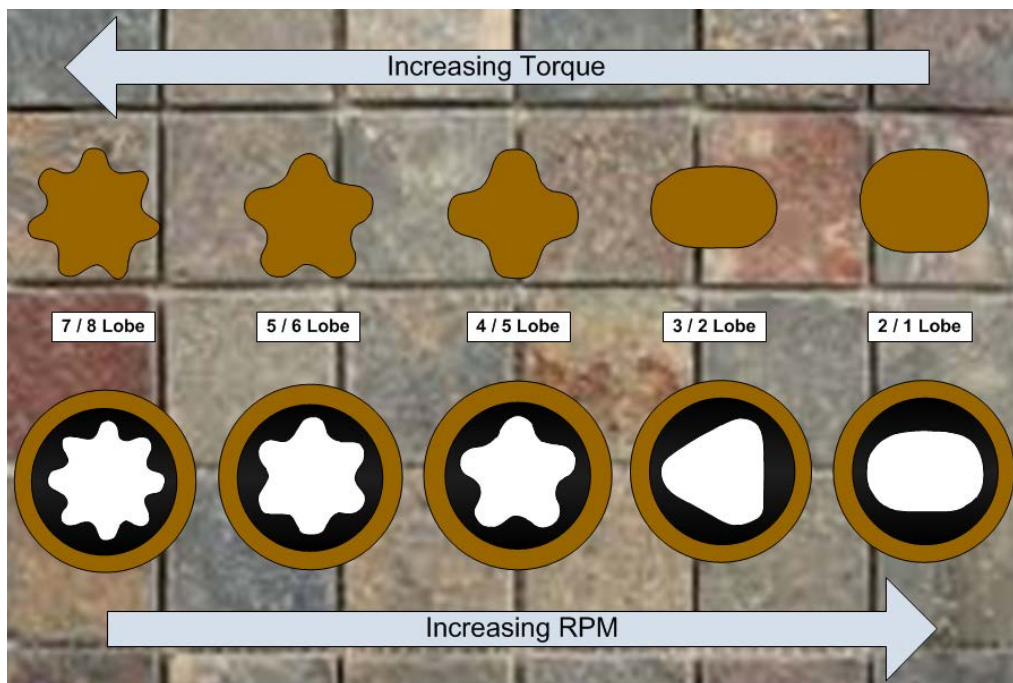


Figure 3.5: Number of lobes affect on Torque and RPM [17].

To best utilize the PDM, an estimate of its performance is vital. The pressure lost through the PDM will affect the hydraulic system, which again will change the RPM and torque it can deliver. In the event of the required torque to rotate ends up being too high, the PDM will begin to stall, and possibly lose its functionality. This is due to no pressure increase in the cavities, resulting in no PDM rotation. Knowing the motor producing capacity is therefore of the utmost importance, as the control system has to keep the PDM within its continuous operating range while still optimizing the drilling parameters to obtain the highest possible Rate of Penetration (ROP).

Bent Housing

Bit tilt is accomplished using a bent housing, which respectively is divided into two possible solutions; fixed

angle or adjustable angle. A fixed angle bent sub, illustrated in Figure 3.6, is made of steel in a fixed bit tilt.

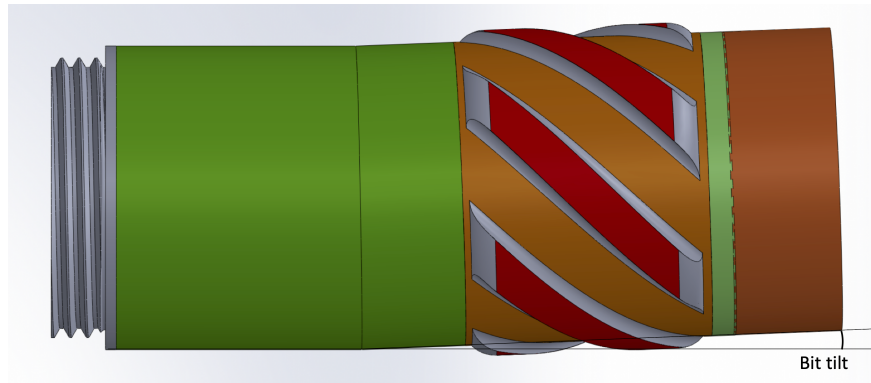


Figure 3.6: Bent sub with fixed angle.

Surface adjustable bent subs consist of a double pin, lock housing, adjusting sleeve and offset housing. Figure 3.7 shows the angle adjusting sequence, which is performed as follows:

1. Unscrew the lock housing and disengage the sleeve from its gear teeth.
2. Adjust to the preferred angle.
3. Tighten the lock housing to detain the wanted angle.

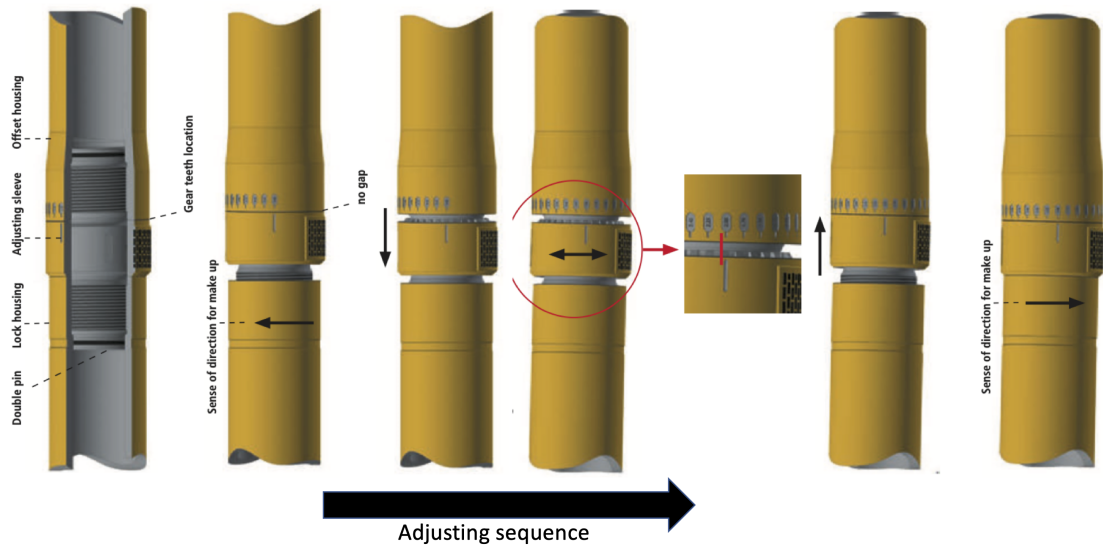


Figure 3.7: Adjustable bend in bent sub [19].

3.3 Drill String Mechanics

Being aware of mechanical limits while drilling is essential, as exceeding them might result in drill string failure. This section covers probable challenges encountered during a drilling operation, and formulas associated to calculate mechanical limits.

3.3.1 Buckling

Avoiding drill string failure is essential. Drill string buckling appears in two stages, sinusoidal and helical [20]. The first stage occurs with increasing compressive loads and is known as sinusoidal buckling, which in short means the drill string will resemble a sinus wave, in other words, a two-dimensional waveform. In practice, this means the drill string will wind back and forth against the wellbore. If further compressive load is applied, the second stage will be entered which is referred to as helical buckling. The drill string will then go up and down the side of the wellbore in a helix shape. Maintaining the same Weight on Bit (WOB) requires more axial load as the increased contact area between the drill string and wellbore increases drag.

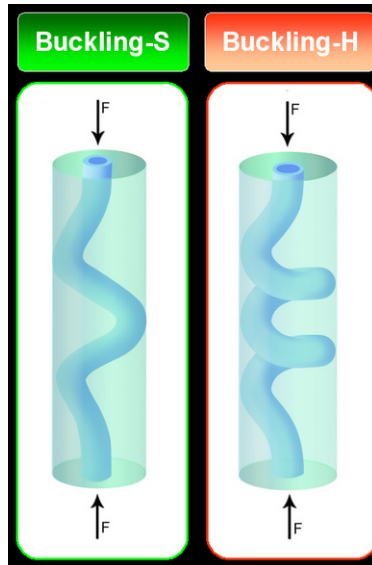


Figure 3.8: Illustrates how the drill string might act as a result of overload [21].

As described, in the event of excessive axial compressive force exceeding its critical value, the drill pipe will buckle. Critical buckling limit is predicted using Euler's column formula for long columns [22]

$$\sigma_{cr} = \frac{F_{cr}}{A} = \frac{\pi^2 E}{\left(K \frac{L}{r_g}\right)^2}, \quad (3.5)$$

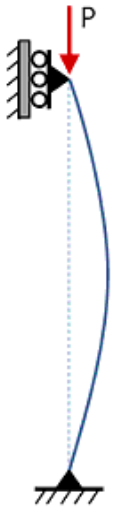



where $\sigma_{cr}[Pa]$ is the critical load, $E[Pa]$ is the modulus of elasticity, the unsupported pipe length is given by $L[m]$, and radius of gyration is $r_g[m]$. The effective length factor is given by K and is determined by the end condition of the column; the different scenarios are presented in Table 3.1.

The radius of gyration is a function of the second moment of area for a pipe, $I[m^4]$, and the cross-sectional area of the pipe, $A[m^2]$.

$$r_g = \sqrt{\frac{I}{A}} \quad (3.6)$$

$$I = \frac{\pi}{64} (OD^4 - ID^4) \quad (3.7)$$

Table 3.1: Effective Length Factor, K, which depends on the end conditions of the column [22].

End condition	Pinned-pinned	Fixed-fixed	Fixed-pinned	Fixed-free
Illustrations				
Theoretical K	1	0.5	$1/\sqrt{2}$	2
Recommended K	1	0.9	0.9	2.1

Whether a column is susceptible to buckling is indicated by its slenderness ratio. "Long" columns have a high slenderness ratio and are therefore more susceptible to buckling. The slenderness ratio, referred to as R_s , is observed in equation (3.8) as L/r_g .

$$R_s = \frac{L}{r_g} \quad (3.8)$$

As mentioned, the Euler formula (equation (3.5)) is used for long columns, while Johnson formula is used for intermediate columns [22]. Johnson's formula is presented in equation (3.9), where $\sigma_{ys}[Pa]$ is the material yield strength.

$$\sigma_{cr} = \sigma_{ys} - \left(\frac{\sigma_{ys}KL}{2\pi r} \right)^2 \left(\frac{1}{E} \right), \quad \frac{L}{r_g} \leq \left(\frac{L}{r_g} \right)_{cr} \quad (3.9)$$

The critical slenderness ratio decides whether Euler's or Johnson's formula is used to predict the buckling limit for a given column, with certain end conditions. Equation (3.10) is used to predict the critical slenderness ratio.

$$\left(\frac{L}{r_g}\right)_{cr} = \sqrt{\frac{2\pi^2 E}{K^2 \sigma_{ys}}} \quad (3.10)$$

Maximum allowable WOB to avoid buckling can easily be estimated from the critical compression, predicted from the Euler formula (equation (3.5)) or the Johnson formula (equation (3.9)) depending on the critical slenderness ratio

$$F_{max\ WOB} = \sigma_{cr} A. \quad (3.11)$$

3.3.2 Burst

Pipe burst occurs as a result of the internal pressure exceeding what the pipe can withstand. Barlow's equation has, for a long time, been used as a standard equation for calculating burst in the industry. However, the equation has shown to be over-conservative for thick-walled pipes [23]. The American Petroleum Institute (API) burst-pressure equation (3.12) is based on Barlow's equation, but takes into account the uncertainties concerning wall thickness by adding a reduction factor of 0.875 [24]

$$P_{burst} = 2 \frac{0.875 \sigma_{ys} t}{OD\ SF}, \quad (3.12)$$

where $t[m]$ is the material thickness and SF a safety factor, here with $SF = 3$ when drilling and $SF = 2$ when tripping [21].

3.3.3 Twist-off

Twist-off is a result of induced shear stress, caused by high torque, exceeding shear strength of the pipe. This will then be a limiting factor for torque allowance on the Drill pipe (DP). Using the thin wall assumption, constant $\tau(r)$, and the mean radius, the maximum allowable torque is defined by [21]:

$$T_{max} = \tau_{max} \frac{\pi}{16} (OD^2 - ID^2)(OD + ID) \quad (3.13)$$

The Von-Mises criterion is then used to find τ_{max} , assuming $\sigma_{23} = \sigma_{31} = 0$ and $\sigma_{12} = \tau_{max}$:

$$\tau_{max} = \sqrt{\frac{2\sigma_{ys}^2 - [(\sigma_z - \sigma_\theta)^2 + (\sigma_\theta - \sigma_r)^2 + (\sigma_r - \sigma_z)^2]}{6}} \quad (3.14)$$

The radial and angular stresses are only caused by internal pressure and can be found by using equations (3.15) and (3.16) [25]. Total axial stress on the drill string is a sum of axial stress from pressure, bending and WOB, which is calculated using equations (3.17), (3.18) and (3.19).

$$\sigma_r = \frac{\left(\frac{ID}{OD}\right)^2 - \left(\frac{ID}{2r}\right)^2}{1 - \left(\frac{ID}{OD}\right)^2} p \quad (3.15)$$

$$\sigma_{\theta} = \frac{(\frac{ID}{OD})^2 + (\frac{ID}{2r})^2}{1 - (\frac{ID}{OD})^2} p \quad (3.16)$$

$$\sigma_z^p = \frac{(\frac{ID}{OD})^2}{1 - (\frac{ID}{OD})^2} p \quad (3.17)$$

Where $r[m]$ is the distance from center of pipe to Point of Interest (POI)

$$\sigma_z^{WOB} = \frac{WOB}{A_{cs}}. \quad (3.18)$$

3.3.4 Pipe Bending

Bending stress is the axial stress induced by DP bending. Bending stress for a beam is given by equation (3.19). Assuming the cross-sections perpendicular to the neutral axis of the beam remains constant [25]. A pipe becomes temporarily thinner on the stretch side when bent, and thicker where it is compressed [26]

$$\sigma_z^b = \frac{E}{RC} r. \quad (3.19)$$

3.3.5 Fatigue

The most common and costly consequential failure when drilling is drill string fatigue. Fatigue is a dynamic phenomenon resulting from stresses applied repeatedly initiating microcracks in the drill pipe which with time can propagate into macrocracks. In combination with corrosion, the cyclic stress shortens the expected lifetime of a drill string significantly.

Fatigue, in principle, only occurs if the drill string rotates while it is axially curved, this curved section respectively experiences one stress cycle per revolution [20]. The cyclic stress is affiliated with bending stress generated from the curvature, as the stress amplitude is directly proportional to the degree of curvature. Bending stress, which is calculated using equation (3.19), represents the cyclic stress and relationship illustrated in this equation shows that the bending stress increases with increasing Outer Diameter (OD). In wells with high DLS, a smaller size drill string might be desirable to minimize fatigue damage.

The fatigue limit of a material is expressed by its correlating S-N curve. The bending stress should not exceed the endurance stress limit given by the fatigue (S-N) curve, as the drill string is not limited to number of rotations. In cases where the material reaches its plastic limit due to high enough appliance of stress, fewer stress cycles are needed to break the material.

3.4 Calculations

When creating a reasonable simulation of a physical system, it is important to know the constraints of the different parts that the system consists of. In this section, the relevant calculations done in phase 1 will be presented.

3.4.1 Torque and RPM Calculations

As this year's development will further progress on last year's rig, the team will use data acquired during last year's testing. Shown in Table 3.2, the results from a test drilling is presented, with a required ROP of $0.434\text{cm}/\text{min}$ achieved with 147N Weight on Bit (WOB) and 68RPM .

Table 3.2: Showing test data from drilling. A constant top drive velocity is set with increasing WOB.

Run	Time [min]	RPM	WOB [N]	Torque diff. [Nm]	Avg. ROP [cm/min]	Sufficient ROP
1	0,35	68	98	0,22	0,23	No
2	0,36	68	108	0,21	0,29	No
3	0,30	68	118	0,20	0,26	No
4	0,33	68	127	0,21	0,37	No
5	0,64	68	137	0,21	0,31	No
6	0,64	68	147	0,21	0,67	Yes

As it is important to have a Bottom Hole Assembly (BHA) that can handle the requirements of around 0.21Nm torque and a drilling rate of $0.434\text{cm}/\text{min}$, a reasonable minimum requirement of 0.7Nm and 70RPM when taking into account the bit increase from last year's testing.

3.4.2 Twist-off

Twist-off calculations estimates the limit for applied torque on the Drill pipe (DP) while drilling. Twist-off torque is calculated using equations (3.13) and (3.14) with σ_r (equation (3.15)), σ_θ (equation (3.16)) and σ_z as input variables. The axial stress, σ_z , is the sum of axial stresses from internal pressure (equation (3.17)), pipe bending (equation (3.19)) and WOB (equation (3.18)).

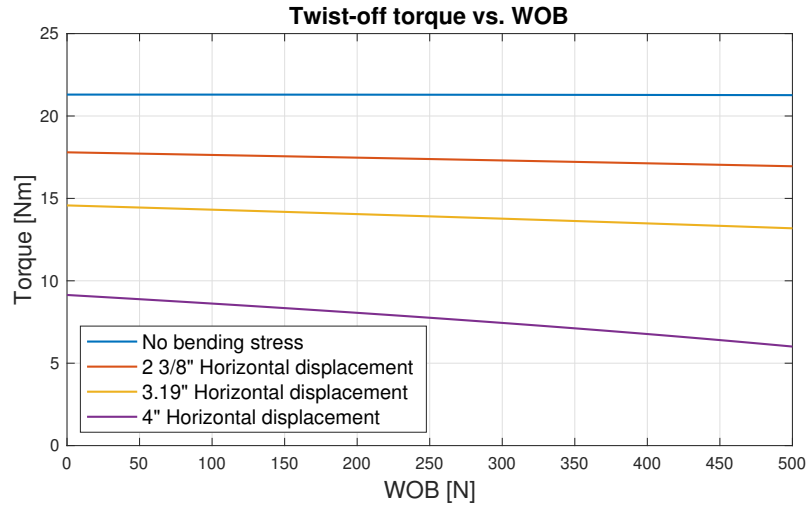


Figure 3.9: Calculated twist-off torques for different trajectories and WOB.

Figure 3.9 shows the result from the twist-off calculations from four different bending cases and four different WOBs, all stresses are calculated outside of the pipe wall. The internal pressure of the pipe had a negligible effect on the torque and therefore it is calculated with $P = 100 \text{ bar}$ in Figure 3.9. The twist-off torque limits for different pressures are presented in Table 3.3, which shows that the difference in pressure is negligible for twist-off torque.

Table 3.3: Twist-off torque ranges for different bending cases and WOB.

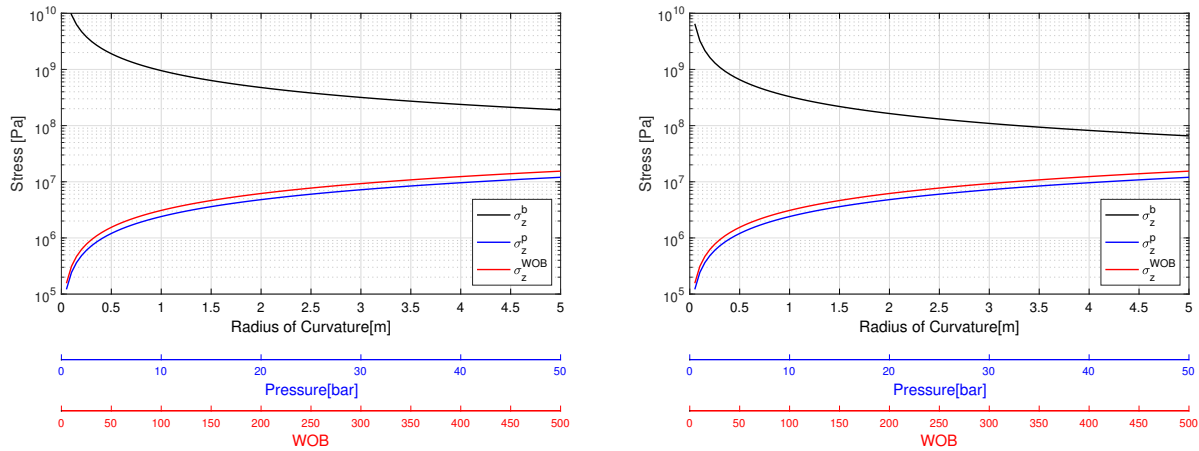
Pressure	Horizontal Displacement			
	0"	2 3/8"	3.19"	4"
10 bar	21.3 - 21.4 Nm	17.0 - 17.9 Nm	13.3 - 14.7 Nm	6.2 - 9.3 Nm
100 bar	21.3 - 21.3 Nm	17.0 - 17.8 Nm	13.2 - 14.6 Nm	6.0 - 9.1 Nm
Twist-off limit	21.3 Nm	17.0 Nm	13.2 Nm	6.0 Nm

Last year's team performed test drilling of the 4" vertical hole, where they observed a maximum torque of 3.34 Nm [27]. These observations are well within the limits estimated.

3.4.3 Pipe Bending

As this year's objective is to hit multiple targets at varying True Vertical Depths (TVDs), it is critical that the pipe is operating within its elastic limits. The pipe will stay within its elastic zone as long as the total axial stress does not exceed the material yield strength.

Equation (3.17), (3.18) and (3.19) is used to calculate the total axial stress, with the result plotted in Figure 3.10 for both aluminium and stainless steel material.

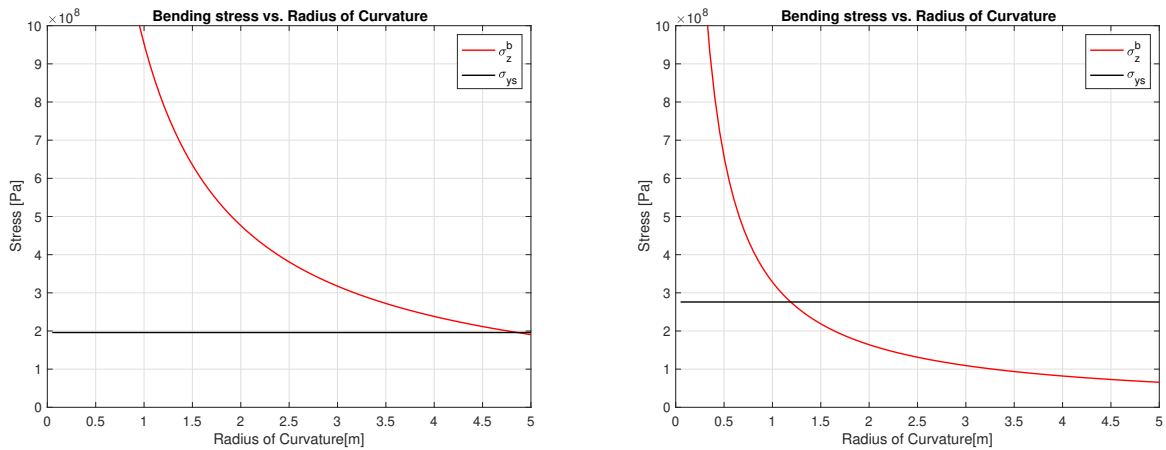


(a) Stainless steel grade 316.

(b) Aluminium 6061-T6.

Figure 3.10: Axial stresses for different values of RC , P and WOB .

Figure 3.10 shows that for both materials, the axial stress from pressure and applied WOB is negligible compared to axial stress from bending. For the pipe to stay within its elastic zone the minimum Radius of Curvature (RC) is predicted based on the axial stress from bending compared to the material yield strength.



(a) Stainless steel grade 316.

(b) Aluminium 6061-T6.

Figure 3.11: Axial stress from bending for different values of RC , compared with material yield strength.

Figure 3.11a illustrates that if stainless steel is chosen as the DP material, the minimum RC is $4.86m$ which corresponds to a horizontal displacement of $1.05''$. Figure 3.11b presents the estimated axial stress from bending for aluminium DP. The minimum RC for aluminium to still stay within its elastic zone is $1.19m$, which corresponds to a horizontal displacement of $4.48''$.

Based on the results shown in Figure 3.11, a stainless steel DP will not be able to obtain sufficient horizontal displacement without exceeding its elastic limit. As a precaution, safety limits for the minimum RC will be applied. For further calculations, the minimum RC used is 1.32, which is equivalent to a horizontal displacement of 4". Based on this, the team has decided to use aluminum as the DP material and this will be considered for further calculations.

3.4.4 Bit Tilt

To achieve directional drilling, it is necessary for the bit to have a tilted angle. This angle will be created by the bent housing and therefore needed to be determined before manufacturing it.

With the use of equation (3.1), where $\phi = 22.6^\circ$, and $CL = 52.1\text{cm}$ acquired from Section 3.4.3, the Dogleg Severity (DLS) is equal to $43.4^\circ/m$. With the knowledge of required DLS, the bit tilt ϕ can be calculated with the use of equation (3.4).

With the use of estimated lengths in the BHA, the section length above the bent sub will be $L_1 = 15\text{cm}$ and the section length below bent sub $L_2 = 8.5\text{cm}$. This results in a maximum required bit tilt angle of:

$$\phi = \frac{DLS(L_1 + L_2)}{2} = 5.1^\circ$$

4 Rig Specifications

When designing and implementing a simulation system for a physical miniature drilling rig, it is important to represent the physical dynamics in a correct way. The physical rig is therefore presented in this section, where large parts are taken from [4] with a few modifications.

4.1 NTNU Miniature Drilling Rig

Figure 4.1 shows last year's rig with its main mechanical components. Given that this year's competition has some minor changes regarding the physical parts of the rig, several alternatives the team has discussed will also be presented, in addition to also explain the reason behind some of the design choices made. The rig's mechanical components can be divided into four different "systems": the Hoisting system, Rotary system, Hydraulic system and Drilling system. Each system with its components will be thoroughly explained in this chapter.

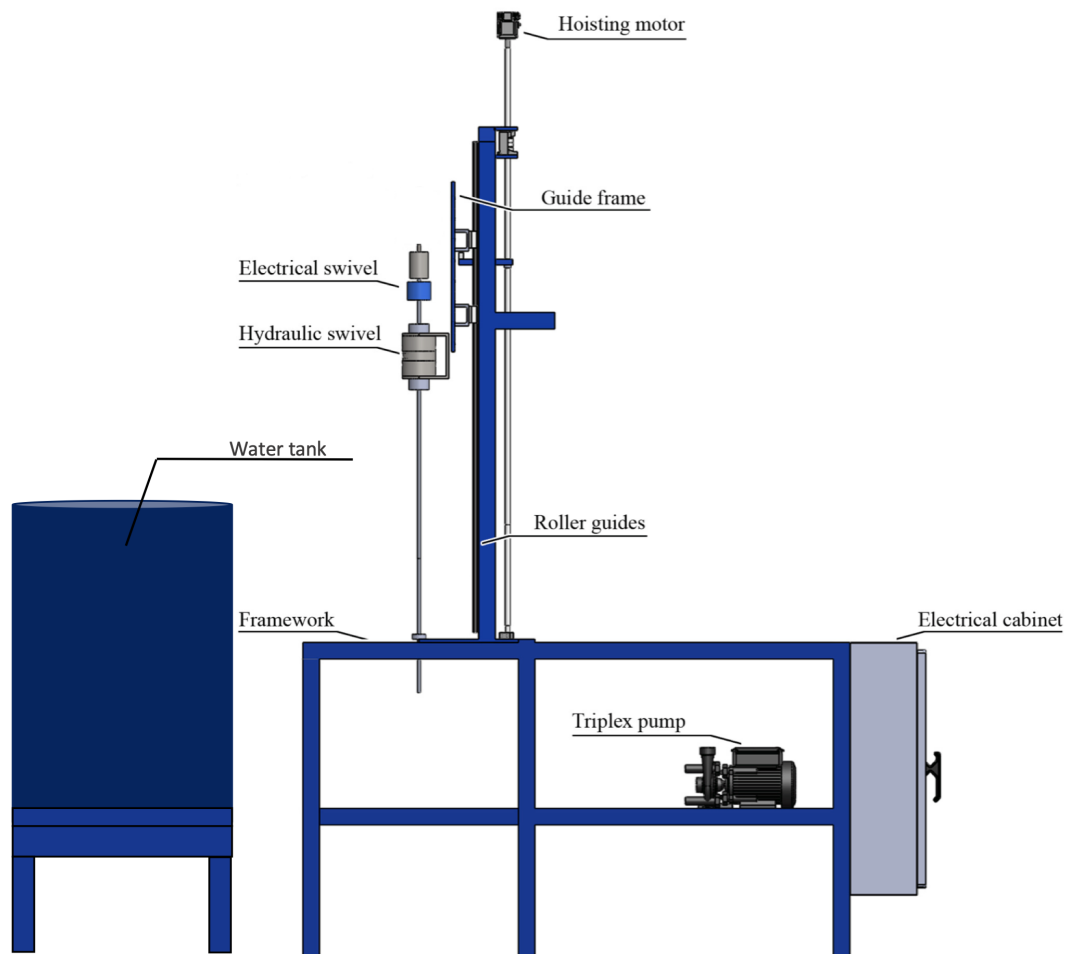


Figure 4.1: The miniature rig and its components from last year.

The rig framework is made of 50×50 mm hollow steel beams which make a total vertical height of 2849

mm, length of 1980 mm and 765 mm wide. As the protection glass is lifted up during rig up/down, the rig's maximum vertical height is 3995 mm. The rig is designed in such a way that it can accommodate several components needed to drill the given well path. The rig's dimensions are also shown in Figure 4.2.

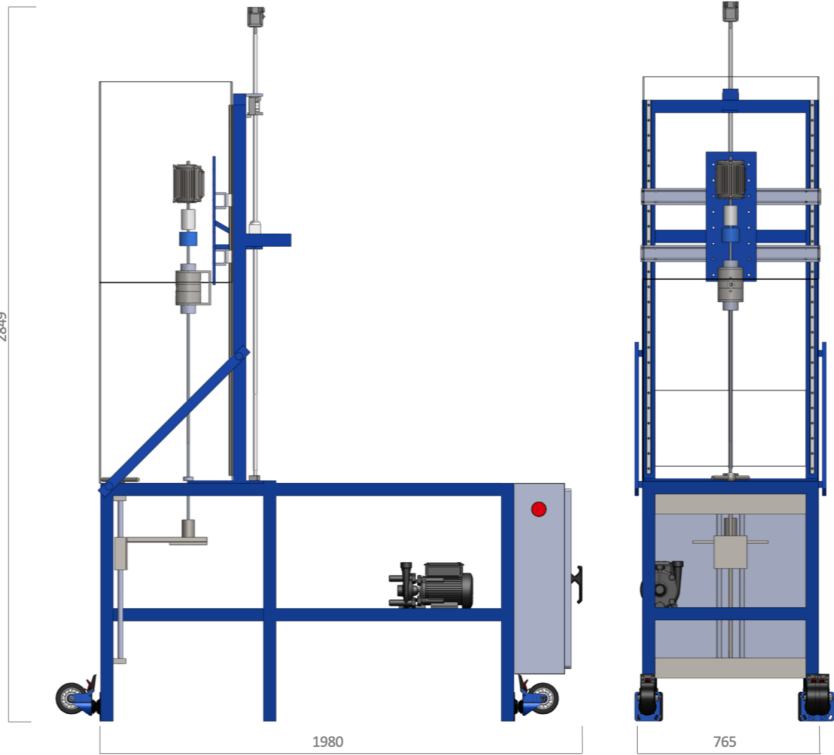


Figure 4.2: Rig dimensions in operational position.

To ease transportation of the rig, there is a possibility to lie down the derrick. This gives a total vertical height of 1685 mm and a length of 2540 mm. The rig's dimensions add up a chargeable weight of 546 kg [28]. The rig in transport position is shown in Figure 4.3.

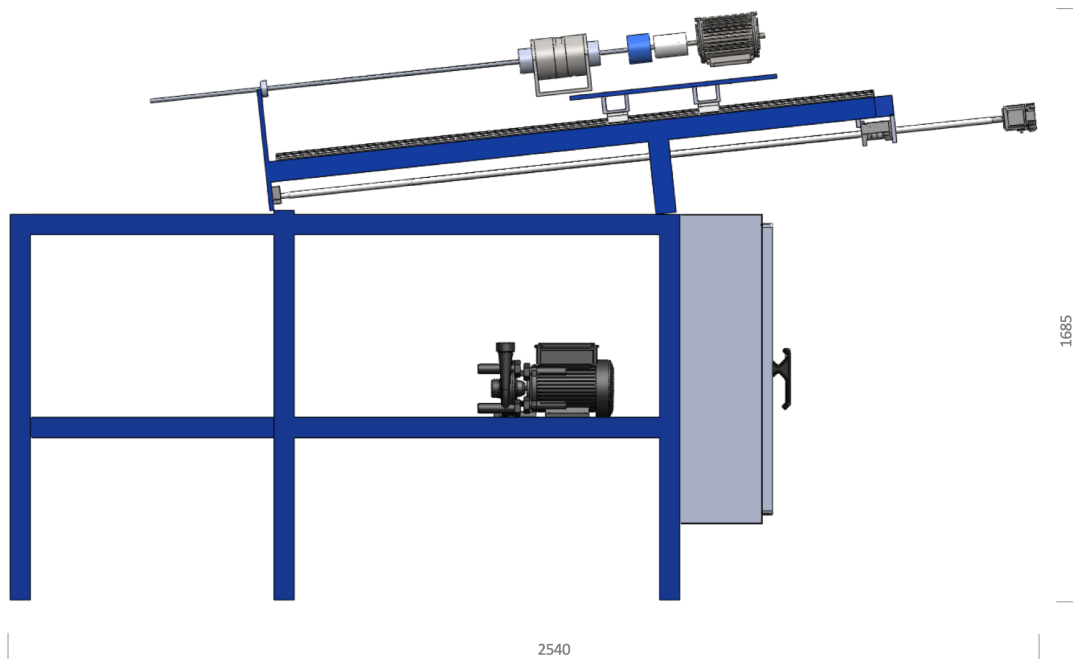


Figure 4.3: Rig dimensions in transport position.

4.2 Hoisting System

The main function behind the hoisting system is to hoist the rotary part of the rig up or down. This will in turn provide Weight on Bit (WOB) and Rate of Penetration (ROP) to the drill bit. An AC motor is connected to the ball screw that hoists the rotary system up or down. At the bottom part of the ball screw, a load cell is connected. This is used to measure the WOB which travels through the rotary system, and into the ball screw. An overview of the system can be seen in Figure 4.4. The hoisting system has two important parts; the hoisting motor and the load cell. These will each be presented in the two following sections.

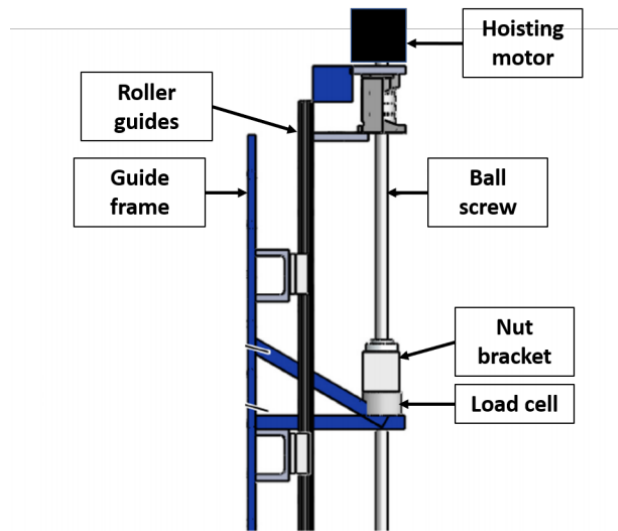


Figure 4.4: The Hoisting motor, ball screw and load cell locations [29].

All the mechanical components of the hoisting system can be seen in Figure 4.5.

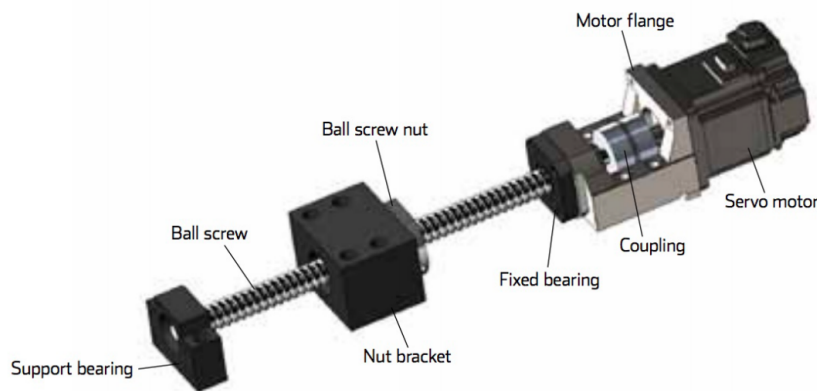


Figure 4.5: Hoisting system with different components labeled.

4.2.1 Hoisting Motor

In full-scale drilling, WOB is achieved by adding heavy weighted drill pipes and collars in the drill string. This is challenging in the miniature drilling rig format, as the pipes do not have enough weight to add the required WOB necessary. So what the previous year's teams have done, is to come up with an elegant solution where a hoisting system is used. Through the use of a hoisting motor, an upward and downward motion of the drill string is achieved, which will provide the WOB necessary. As the automated part of the

drilling requires continuous monitoring and changes to the WOB, it is important that the hoisting system is accurate. This is achieved with the use of a ball screw. The motor provides rotational motion to the ball screw, which further converts this motion into the vertical movement of a guide frame where the rotating system is mounted. The ball screw is chosen as the solution due to its precision, accurate data gathering function and efficiency.

The hoisting motor used for this year's competition is fully capable of achieving the end goal of this year, and will therefore be used for this year's competition as well. The motor model is "Lenze GST03-2M VBR 063C42", and is used to hoist the whole rotary system up or down. The maximum output power is 0.75 kW, and the motor works at a frequency of 120 Hz, with a rotary speed of 3400 RPM. The maximum torque of the motor is 45 Nm, and it has a power factor of 69%, with a maximum efficiency of 79%. The gear ratio between the connection of the hoisting motor RPM and the ball screw RPM is roughly 1:9.

The hoisting motor has an encoder, which makes it possible to measure position, RPM and torque with high precision. This makes it possible to find the well depth of how far the drill bit has drilled. It is important to note that even if the encoder shows an increasing count, this does not necessarily mean that the well depth is increasing. This is due to a possible bend in the drill string. To better estimate the length, the hoisting motor also has a load cell which will be presented in the next section. A more detailed description of the presented numbers can be found in [30][31][32].

4.2.2 Load Cell

The load cell is located in the middle of the ball screw. As mentioned, the hoisting system's purpose is to apply WOB, which is measured with the use of the load cell. This part is connected to a ball screw nut bracket and is shaped like a hollow cylindrical cell. The load cell is welded onto the guide frame, this is to get a high enough precision on the WOB readings. Figure 4.6 shows the placement of the load cell on the rig with a ball screw going through it.

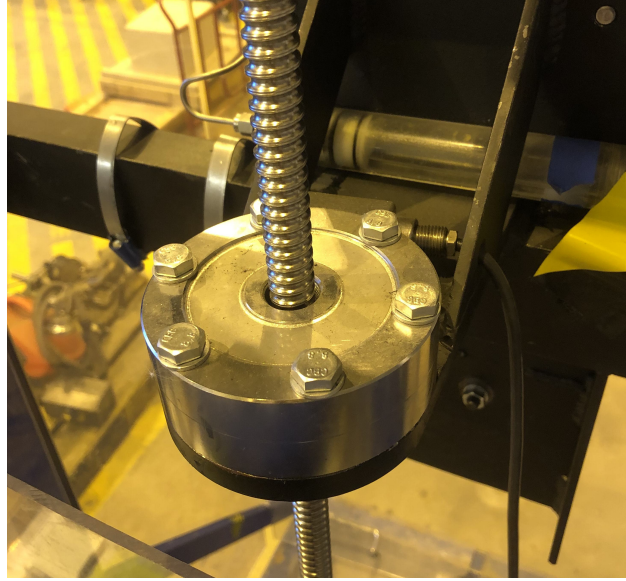


Figure 4.6: Load cell.

The load cell that is used is the same as the load cell used in the competition of year 2018 and 2019. The model is TC4-AMP transducer by APE Transducer [33], and is presented in Figure 4.7. As mentioned, the load cell is mounted between the rig frame and the ball screw shown in Figure 4.4. When there is a normal force acting on the drill bit, i.e after the drill bit tags the rock, there will be a normal force acting through the rotary system and through the ball screw to the load cell. The output of the measured force can both be represented in voltage or current. In this case, the travel distance for the signal to be read is quite small, which is why the voltage will be used. The output voltage is between $[-10V, 10V]$, which translates to a maximum and minimum force measurements of $[-2500N, 2500N]$. This gets translated to WOB [kg] by equation (4.1)

$$WOB = V \frac{F_2 - F_1}{g(V_2 - V_1)} - m_{offset}. \quad (4.1)$$

V is the measured voltage, F_2 and F_1 are the maximum and minimum forces ($2500N$ and $-2500N$ respectively), V_2 and V_1 are the maximum and minimum voltages possible ($10V$ and $-10V$ respectively), and m_{offset} is a constant to cancel the weight of the rotary system. Together with the measurement of the WOB and the measurement from the hoisting motor, the position of the drill bit can be estimated with the use of a Kalman filter and the orientational measurements gotten from the IMU.

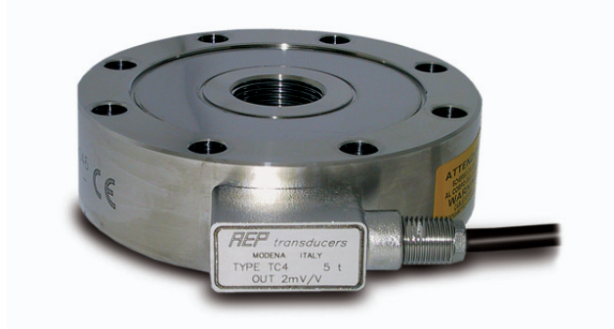


Figure 4.7: Load cell: TC4-AMP transducer by APE Transducer [33].

4.3 Rotary System

The main function behind the rotary system is to provide torque and RPM to the drill string and the directional tool. The rotary system can be seen in Figure 4.8.

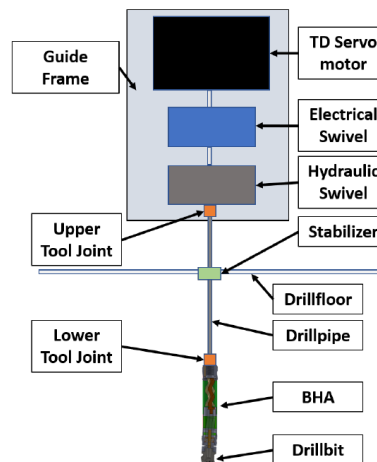


Figure 4.8: The rotary system of the drilling rig with its different components [29].

4.3.1 Top Drive Motor

The top drive motor is used to directly give RPM and torque to the drill pipe for initial drilling of the vertical hole, while it will be used for positional orientation of the directional tool when doing position control of the drill bit. This will be used to control the drill bit to follow the reference path. The servo motor used is Schneider Electric BCH2MM1523CA6C [34], which can provide an RPM to the drill pipe of 2902 RPM, which is more than enough for this drilling operation. The top drive motor can give a stall torque of 7.24 Nm, and peak torque of 19.7 Nm. The nominal speed of the top drive is 2000 RPM, and with the calculations given in Section 3.4.1, which states a needed RPM of 70, this should be well inside the Top Drive (TD) limits.

Since this year's competition guidelines set more focus on the directional part of the drilling, accurate measurements of the top drive position and drill bit orientation are even more important this year. Therefore,

the team is using Schneider Electric LXM28AU15M3X [35] as a servo drive. The accuracy of the internal position controller of the top drive servo drive is stated to be 0.1%, which translates to 0.36° . This should be sufficient enough to control the drill bit to follow the reference path. The servo motor and servo drive used in the top drive is shown in Figure 4.9.

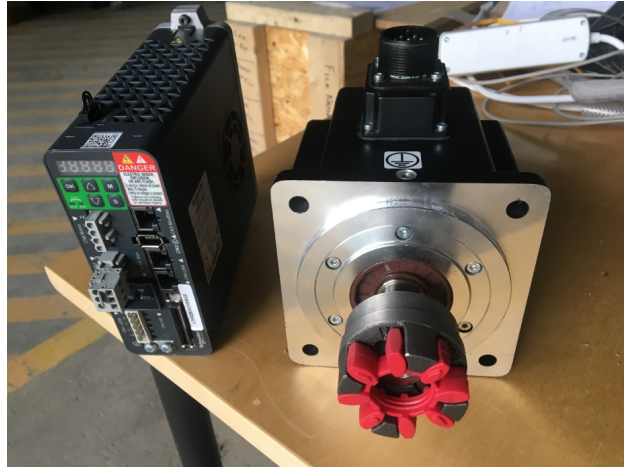


Figure 4.9: The Schneider Electric servo motor and servo drive [29].

4.3.2 Drill Pipe

This year, the guidelines have allowed the use of stainless steel as Drill pipe (DP) material, with the same dimensions as last year. The selection of stainless steel is a solution the team highly has taken into consideration. With the use of stainless steel, the wear and tear on the DPs will have a much higher limitation and will allow for more bending and stress. As the calculation and discussion in Section 3.4.3 states, the use of stainless steel DP will not be within its elastic zone with the required Radius of Curvature (RC). As the stainless steel possibility is ruled out, the choice of DP material this year will be aluminum of the alloy 6061-T6.

4.3.3 Power Section

To be able to use the TD as a control input to the directional drilling, the need for a power section in the Bottom Hole Assembly (BHA) is needed. A Positive Displacement Motor (PDM) is responsible for providing the necessary torque and Revolutions per Minute (RPM) during slide drilling. The PDM power section is closest alternative to real size solutions. The PDM will consist of a rotor and stator. Through the design of the PDM, the hydraulic power from the drilling fluid will be converted into mechanical power to supply torque and RPM to spin the drill bit independently from the rest of the drill strings rotation.



Figure 4.10: 3D-printed plastic version of stator and rotor.

4.3.4 Bent Housing

Below the power section, the bent housing is located. As there are no specified requirements to determine the required dogleg angle, there is a need to use an adjustable bent housing. The adjustable bend will require a max dogleg angle of 43.4° , as calculated in Section 3.4.4.

Inside of the bent housing the transmission section will be placed, which will transmit the torque and RPM from the power section and towards the bit. The planned transmission type to use is a flexible shaft, this is chosen instead of universal joints to limit the needed length and less moving parts than universal joints.

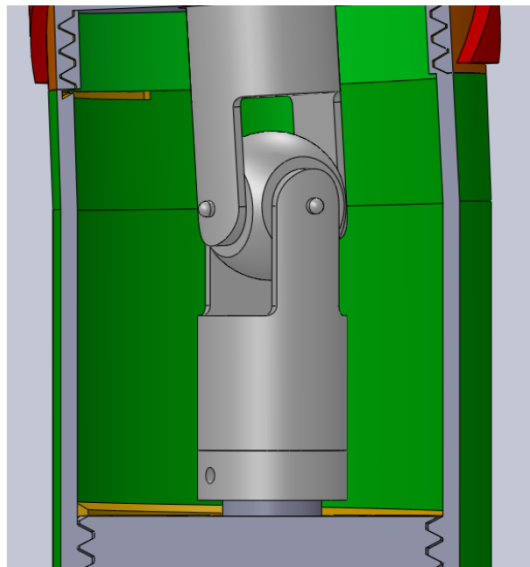


Figure 4.11: Bent housing.

4.4 Sensor Card and Communication

Sensor Sub

This year, the team has decided to continue with the PDM, which means that there is no electrical power needed beneath the sensor card. In Figure 4.12, the sensor sub is presented, which is where the sensor card will be placed. The sensor sub is placed on the top of the BHA, which is presented in Figure 4.13.

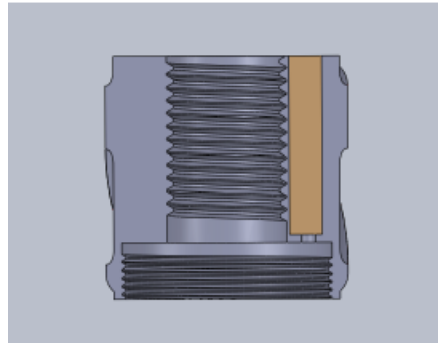


Figure 4.12: The downhole sensor sub on the top of the BHA.

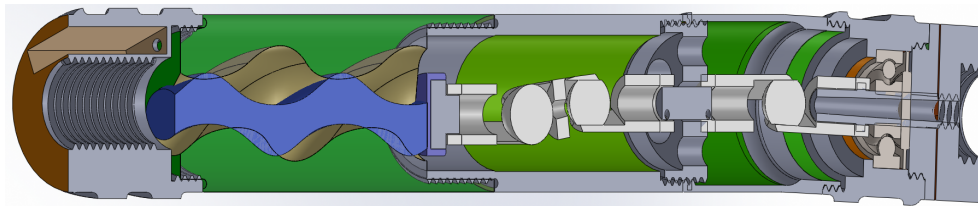


Figure 4.13: Concept design of BHA with PDM as power section.

The current design of the sensor sub takes in account the wires that are needed for downhole communication. If the communication method is changed to wireless, the design of the sensor sub will go through significant changes. First, there will no longer be a need for wires into the sensor sub, which removes the leakage problem. On the other hand, the card will need power from a battery, and the amount of space needed is therefore increased.

Wired Communication

The current communication solution is shown in Figure 4.14.

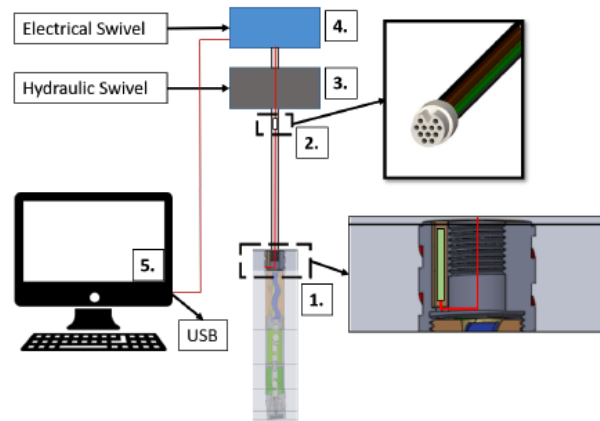


Figure 4.14: The wired downhole communication scheme [29].

As can be seen in the figure, the wires are pulled through the electrical swivel that transforms the signals such that they can be used in the rotary system. The wires are then pulled into the bottom entrance of the sensor sub. With the change from an electrical downhole motor to the PDM, the number of wires downhole needed this year is only 4 compared to last year's 10. If wireless communication works well after testing with regards to the quality of the data passing through the different material in the stone, the wires will be removed completely, and also the electrical swivel.

In the case of using wires, the team will reduce the last year's size of the connectors, as there is no longer a need for 10 downhole wires. This year, the team will use a 6 pin circular plastic shell connector compared to last year's 11 pin connector. The new sizes compared to the two last years are presented in Table 4.1. To better understand the specifications presented in the table, Figure 4.15 shows the different dimensions of the connectors.

Table 4.1: Change in dimensions for the wire connectors [36].

Year	Dimension A	Dimension B	Number of pins
2018	5,6	1,0	6
2019	3,89	0,64	11
2020	3,10	0,64	6

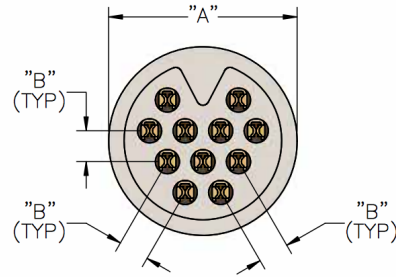


Figure 4.15: The measurable dimensions of the Omnetics Nano connectors [36].

The wires that go into the sensor sub is the most vulnerable part of this communication scheme, as the problem with leakage is hard to prevent when there is used high pressured water for the PDM that passes the sensor sub. However, the team will use epoxy to seal the sub from the water.

Sensor Card

The sensor card is the same as in the previous year, which has an Inertial Measurement Unit (IMU) that collects data for further use in the control system. The IMU is of type ICM-20948 [37] 9-axis motion tracking from TDK InvenSense, and is shown in Figure 4.16, and has a gyroscope, accelerometer and magnetometer which can be summarized as follows:

1. Gyroscope

- Digital x -, y - and z -axis angular rate sensors
- Configurable output range of ± 250 , ± 500 , ± 1000 or ± 2000 degrees per second (dps)
- User-selectable low pass filters

2. Accelerometer

- Digital-output x -, y -, and z -axis accelerometer
- Configurable output range of $\pm 2g$, $\pm 4g$, $\pm 8g$ or $\pm 16g$
- Integrated 16-bit analog-to-digital converter (ADC)
- User-selectable low pass filters
- Wake-on-motion interrupt for low power operation of applications processor

3. Magnetometer

- 3-axis Hall-effect magnetic sensors
- Output data resolution of 16-bits
- Maximum output range of $\pm 4900\mu T$



Figure 4.16: The ICM-20948 IMU used in the sensor card [37].

With the measurements from the IMU, the team will be able to control the drill path based on the orientation of the IMU.

5 Initial Work for the Drillbotics Competition

Before the COVID-19 pandemic, the goal of this report was to present a working solution for a real-time, autonomous drilling system to successfully achieve the problem statement of the Drillbotics competition 2020 presented in Section 1.2. As all Norwegian universities were closed 13th of March 2020 due to COVID-19, the scope changed dramatically due to inaccess to physical equipment. However, there has been done work on the physical solution before this pandemic, which will be presented in this section.

5.1 Implementation of New Sensor Cards

To have a reliable control system for autonomous drilling, reliable downhole data is needed. To achieve this, a custom-made sensor card was to be placed inside the Bottom Hole Assembly (BHA) for real-time positional data of the drill bit. Since the schematics of the sensor card already had been made by the previous year, it was only needed to solder the needed components onto the already designed Printed Circuit Board (PCB). All the needed components were present, but the PCB was of an older version, specifically version 1 of the final design as presented in [29].

5.1.1 Soldering of Sensor Card

To be able to solder the components, a microscope is utilized as the components are very tiny. After several attempts and some trial and error, a sensor card was finally soldered correctly with all pins connected. In Figure 5.1, one can see the two first trial attempts were a pin was destroyed, and a component was soldered incorrectly with ground. The third attempt represents a correctly soldered card together with wired connections to the J-link adapter.

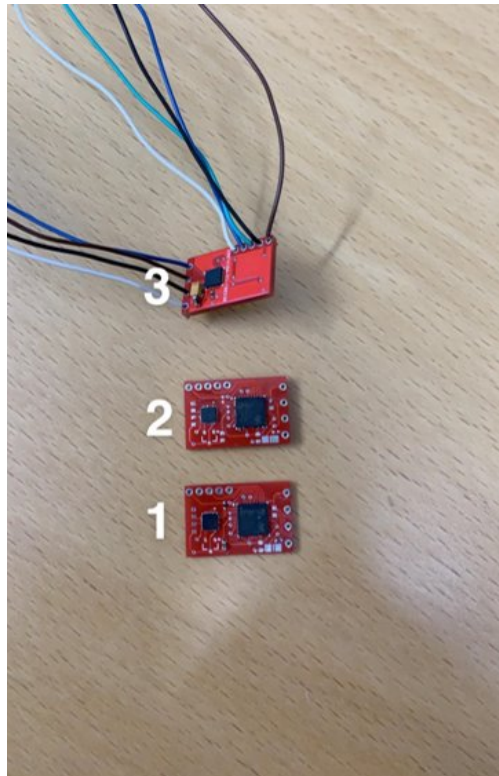


Figure 5.1: The first two trial attempts of soldering new sensor cards, and the final working third attempt.

Because of the COVID-19 situation, it was impossible to order version 2 of the final design, as it was no longer possible to get it sent from China. The solution was therefore to use version 1 but to solder a connection between the ADO pin on the ICM-20948 to ground. If this was not done, there would be a 25% chance to guess the right address of the ICM-20948 chip.

5.1.2 Soldering Connector to the J-link Adapter

To be able to flash the code onto the sensor card, a J-link adapter was used. To create this connection, one must solder a 20-pin-connector with the correct wires according the schema shown in Figure 5.2 [38]. For easier soldering, it is recommended to short circuit all the ground pins together. Also, "NC" means "not connected", so this should not be connected to anything.

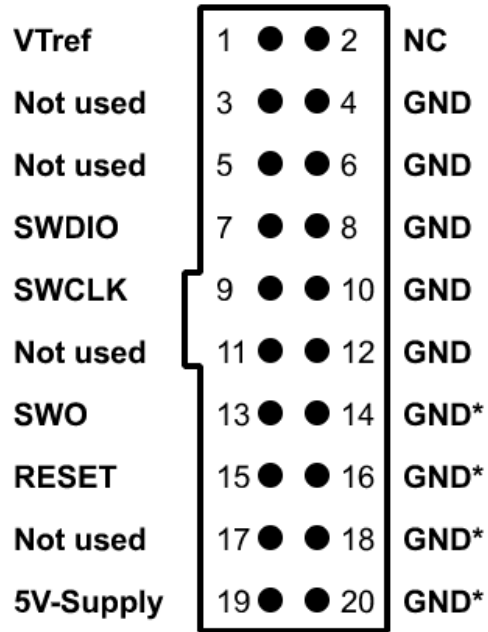


Figure 5.2: The 20 pin connection scheme used for the J-link adapter.

Once a 20-pin connector has been soldered up with wires according to Figure 5.2, the wires must be soldered correctly onto the sensor card. The 5 connections points are shown to the right in Figure 5.3, and are as follows in descending order

1. Vcc
2. GND
3. SWO
4. SWDIO
5. SWCLK

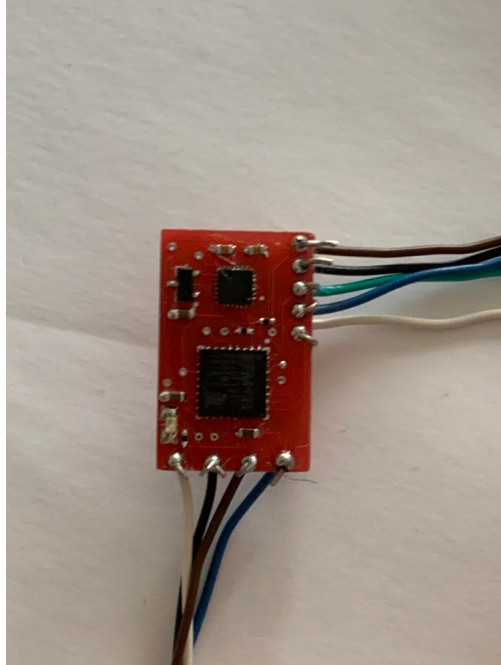


Figure 5.3: The wires connected to the sensor card.

Note that the USB-connection wires are also soldered on, and are shown as the 4 bottom connections.

5.1.3 Soldering USB-connection to Sensor Card

To power the sensor card, it must be connected with a USB cable to a computer. It is therefore necessary to solder a USB-cable that connects to the correct pins on the sensor card. This is done by cutting up an existing USB-cable and locating the 4 wires as shown in Figure 5.4. The meaning of the different wires are as follows

1. Red: +5V
2. White: Data-
3. Green: Data+
4. Black: Ground

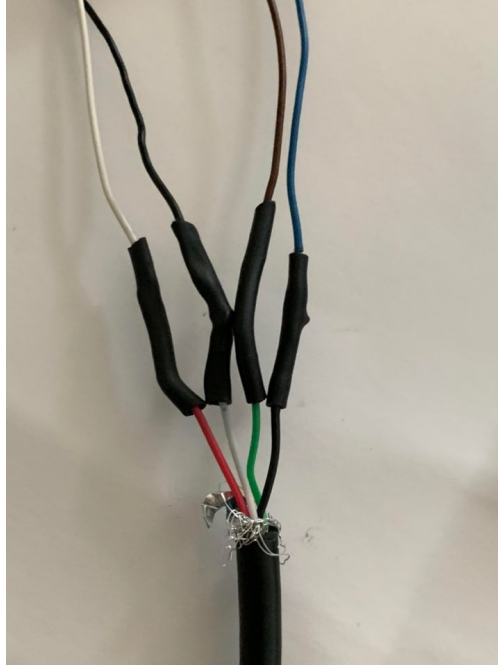


Figure 5.4: The different wires found inside a USB cable.

These wires are connected with longer wires. For better durability, heat shrink tubing is used. These wires are then connected to the sensor card shown in the previous Figure 5.3 at the bottom, where each pin can be described in the following order

1. +5V
2. Data-
3. Data+
4. Ground

5.1.4 Programming the Sensor Card

As all the connection wires have been soldered, the sensor card can now be connected to Simplicity Studio. Both cables are plugged into the computer, and Simplicity Studio will then recognize the J-link adapter. It is then possible to flash the code onto the sensor card. This was not worked further upon due to the mentioned pandemic.

5.2 Simulation System - Simple Simulation for Verification

In order to simulate the drilling environment and its response to different control methods, MATLAB is used as it offers simple implementation and a good overview of the system. The objective of this simulation system was to get an overview to see if the calculations and the design of the control algorithm would work as early in the project as possible.

For this version of the simulation system, only the most simple dynamics were considered. The system consists of four main compartments, and are highlighted with colors in the overview found in Figure 5.5. The blue box generates reference coordinates of which the green PI controller uses to generate input. The yellow box consists of simple state dynamics where the inputs are the Rate of Penetration (ROP) and the controller input from the Top Drive (TD) actuator. This yellow block also outputs the orientation and position calculations. The state machine is in the orange box, and will not be covered in this version.

State estimation was also not implemented in the version, such that it is assumed that all the states are measured perfectly. The main focus here is to create a backbone of which easy change of compartments is possible. The main objective of this version is therefore to check that the plotted responses are reasonable to show that all matrix rotations and coordinate frames are correct.

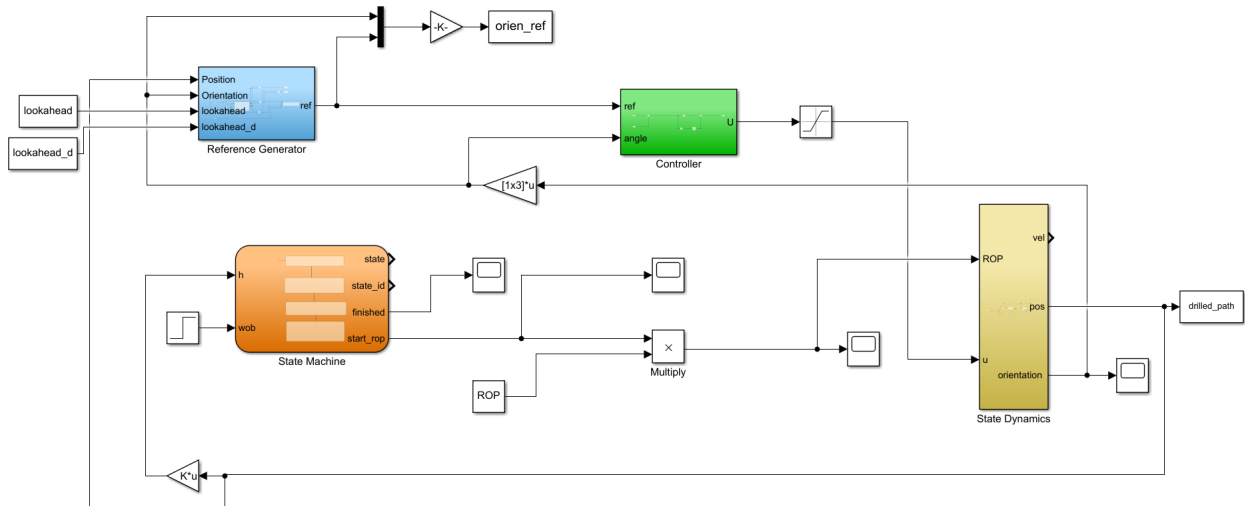


Figure 5.5: Overview of simple simulation system.

5.2.1 Reference Coordinate Generation

As the Drillbotics committee will provide coordinate points of which the borehole has to go through, it is important to generate a path that satisfies this condition. In this version, the cubic spline method is implemented and used as described in Section 2.6. Cubic spline interpolation was chosen in this version of the simulation, based on work done in the Drillbotics design report.

The path generated by the interpolation is then fed into the blue box which decides which of the coordinate points in the path that shall be used at every simulation step. In this iteration, the point is chosen by finding the coordinate point that has the closest z^I value to the drill bit z position. It is also possible to choose a lookahead distance, which chooses the coordinate point that is d steps ahead of the drill bit's z^I position. The reference coordinate point generation subsystem can be seen in Figure 5.6, with two MATLAB functions seen by the green and blue box.

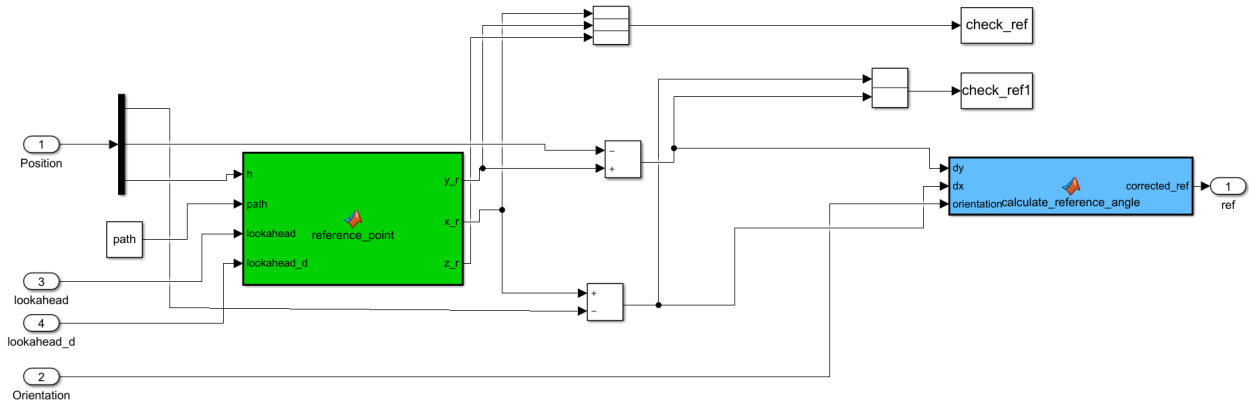


Figure 5.6: An overview of the implementation of extracting the reference angle ψ_r .

The green box includes the MATLAB function to generate the reference coordinate point as described previously. This coordinate point is then used to find the difference of the x and y coordinates between the reference point and drill bit position.

By comparing the reference coordinate points and the position of the drill bit, a directional vector is created. The directional vector is then used together with the known orientation of the drill bit to create a reference orientation. The used calculation is described in Section 6.4.4. The code can be found in the Appendix in Section B.3.

5.2.2 Control System

For this version, a simple PI-controller is used and is implemented in the green box shown in Figure 5.5. Since the drilling operation is a slow process, a PI controller is sufficient to avoid any steady-state errors and overshoots. The controller takes the generated reference orientation as well as the orientation of the drill bit, which it then uses to generate an input as shown in Figure 5.7.

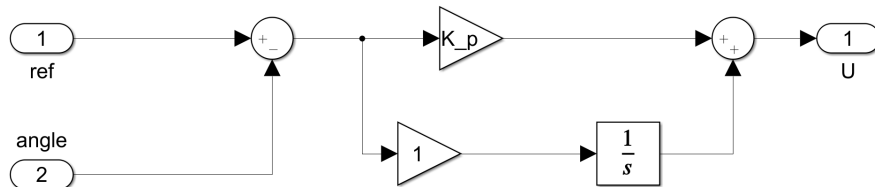


Figure 5.7: PI controller

5.2.3 State Dynamics

The input from the controller is fed into the yellow state dynamics box, together with a constant ROP. The new orientation and position of the drill bit is then calculated based on matrix rotations and theory presented in Section 2.1. The Simulink system for this can be seen in Figure 5.8. The new states are then fed back into the system for the next iteration.

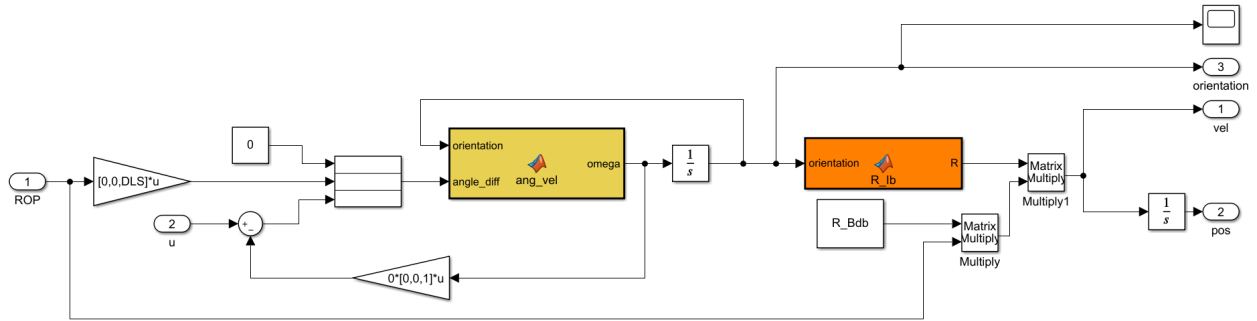


Figure 5.8: State dynamics for the first version of the simulation system.

5.2.4 Simulation Results

As the system has different parameters such as lookahead and saturation limits, the results are presented as these parameters get changed. The objective of the simulation was to see if there were any flaws in the control algorithm used by the rig, as well as checking that the state dynamics are correct.

No Saturation and no Lookahead

With no saturation and no lookahead, the drill bit is allowed to rotate with no regards to the stone being in the way, or restrictions in the top drive actuator. It is also important to note that ROP is set to a constant value, as there is not implemented any state dynamics for this state in this version. The resulting simulation can be seen in Figure 5.9.

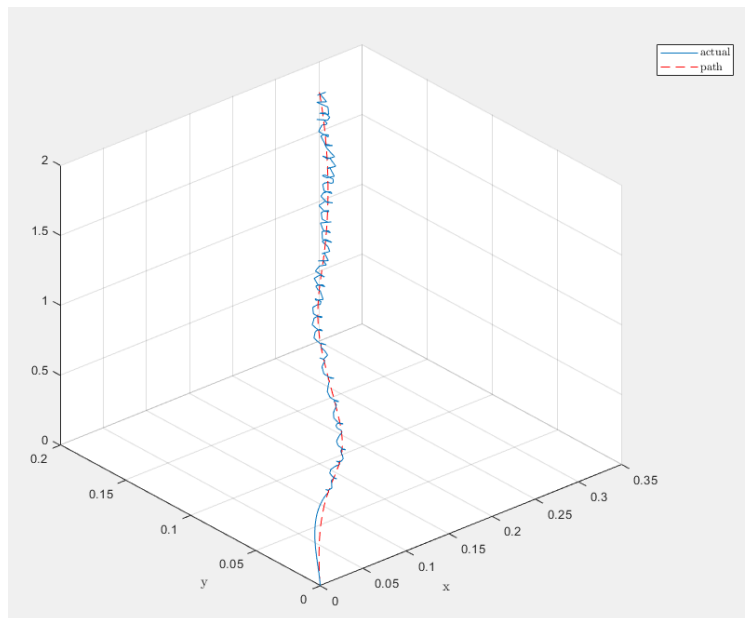


Figure 5.9: Simulation result with no saturation and no lookahead.

As seen in the figure, it can be confirmed that the correct mapping of coordinate points from the drill bit frame to the inertial frame is correct. However, the drill bit is drilling around the reference path, and making turns

that are completely infeasible with real drilling. This result occurs because of the lack of state dynamics for drill bit rotation. An important observation that can be extracted from this simulation, is why the controller decides to spin around the reference trajectory. The spinning starts in the cases where the Dogleg Severity (DLS) is too big with regards to the path, i.e the inclination angle of the drill bit builds faster than the trajectory path. The result of this is that the drill bit builds up a lead in the x^I and y^I coordinates compared to the drilled depth, z^I , and thus, the controller wants to turn back.

No Saturation with Lookahead

By using lookahead, it can be seen in Figure 5.10 that the drill tries to follow points that are further ahead in the reference path. This can be useful in situations where you do not want the drill bit to strictly follow the path at the same height as itself, but rather look ahead. This can be seen as the drilled path is beneath the reference path, as it tries to follow coordinate points that have a greater z values than itself.

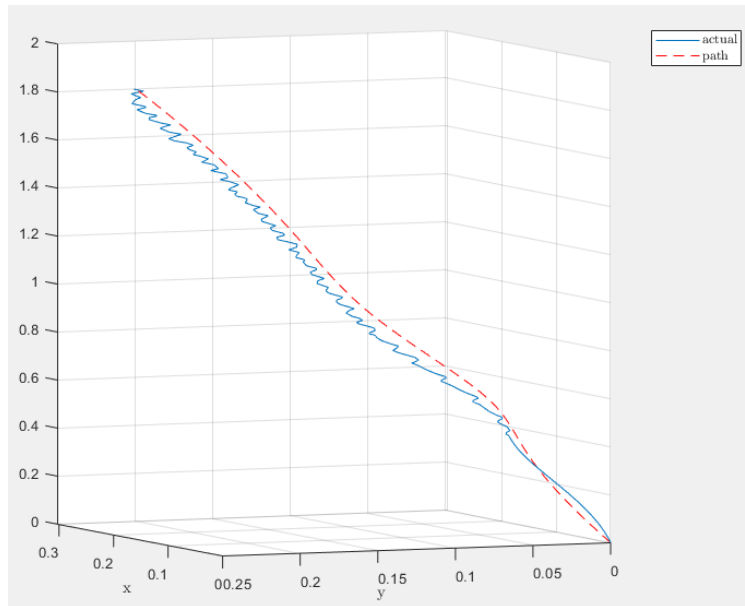


Figure 5.10: Simulation result with lookahead and no saturation.

With Saturation and no Lookahead

A fatal flaw with the reference point generation algorithm is first found when saturation is used. It is also apparent with no saturation, and is shown by the circles the drill bit is doing around the reference path. In Figure 5.11, one can see that instead of following the reference path properly, the path deviates. The reason for this is that once the drill bit passes the reference path in the $x - y$ -plane, the algorithm chooses the longer path back to the reference path, which makes it do a 360° turn.

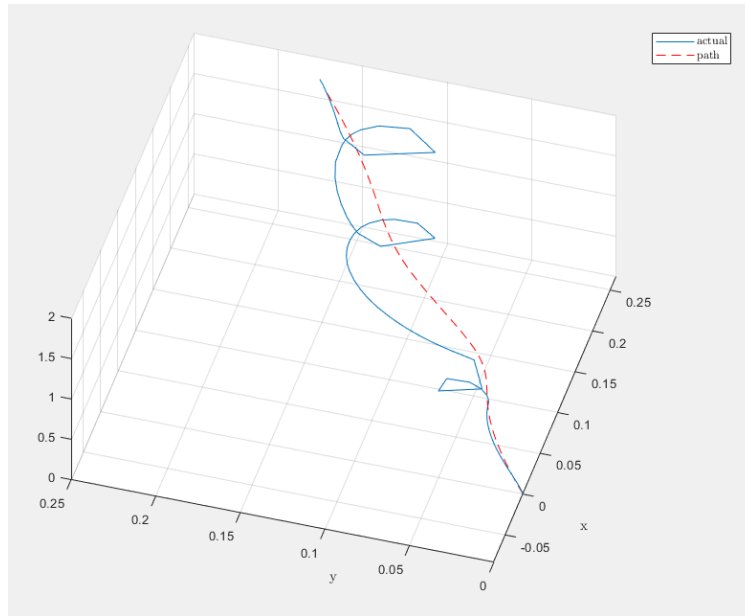


Figure 5.11: Simulation result with saturation and no lookahead.

By further investigation of the first intersection between the actual and reference path, it is easy to visualize why this becomes a problem. In Figure 5.12, the right coordinate point shows the intersection point, while the left coordinate point shows the x,y -coordinates of a point on the reference path that has greater z value than the actual position. With no lookahead, the generated point will even be further behind the drill bit position. The generated directional vector is therefore pointing in the opposite direction of the future path, which in turn gives the shortest orientation reference in the opposite direction.

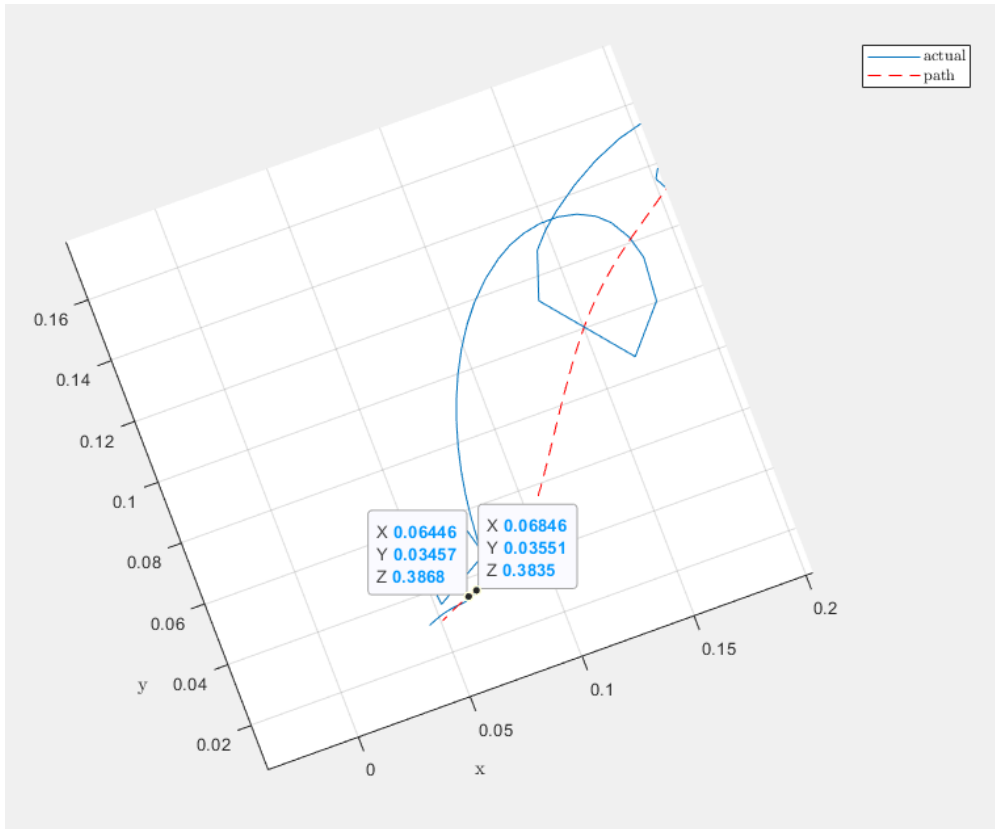


Figure 5.12: Problem with reference coordinate point generation.

With Saturation and Lookahead

To fix the problem of using coordinate points behind the drill bit position, there can be defined a lookahead distance of how many steps in front of the closest z value that should be used. An example of how this may mitigate the problem of using coordinate points behind the drill bit position can be seen in Figure 5.13, where the left point is the position of the drill bit in the intersection, and the right point is the reference point that should be used as the reference instead of a point behind the drill bit position.

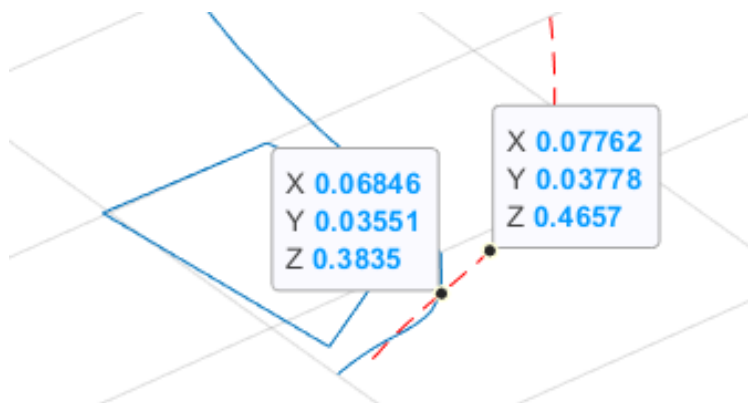


Figure 5.13: Example of how lookahead distance may mitigate directional vector problem.

As there is used a total of $t = 501$ time steps and total height of $z_t = 1.8$, one can calculate the needed lookahead steps z_s as follows

$$\frac{z_t}{t} \cdot z_s = z_{diff} \quad (5.1)$$

$$z_s = \frac{z_{diff} \cdot t}{z_t} \quad (5.2)$$

$$z_s = \frac{(0.4675 - 0.3835) \cdot 501}{1.8} \approx 23 \text{ steps} \quad (5.3)$$

In Figure 5.14, one can see what happens when all parameters are the same, except there is included lookahead distance of $z_s = 23$. As can be seen, the intersection point is now at around $z_i = 0.4614$. By adding the lookahead distance, one gets that the reference point that is used is $z_r = 0.5453$, which again is behind the position of the drill bit. A constant lookahead distance is therefore not sufficient.

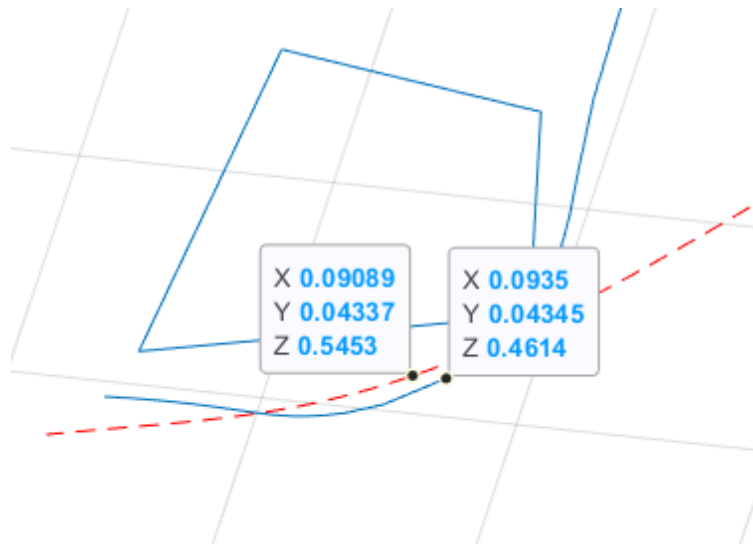


Figure 5.14: Using constant lookahead distance does not mitigate problem.

6 System Description and Control Design

In order to create a simulation system that accurately represents the true dynamics of autonomous drilling in the Drillbotics competition, the system must be described and designed as close to the real drilling environment. In this section, the design of the simulation system will be presented.

6.1 System Description

The real drilling environment consists of many variables with complex relations which makes it important to define the multiple variables of the simulation system and their meaning.

6.1.1 System Inputs

When autonomously controlling a drilling rig, the possible inputs are a key factor of the design considerations. In this simulation system, the two possible inputs are the hoisting motor and top drive, while in the real system one would also have to control downhole power output with for example a pump motor for a Positive Displacement Motor (PDM) or electricity to an Electrical Miniature Motor (EMM). This functionality is included in the simulation system by an assumption of sufficiently high Weight on Bit (WOB) as described in Section 6.2.4. The actuators can therefore be defined as the following:

- **Hoisting motor**

Controls WOB by hoisting the rig either up or down when the drill bit is in contact with the stone. If not, the motor controls the z -position of the drill bit when the z^I -axis is given in the inertial frame.

- **Top drive servo motor**

Controls both the drill string rotation position and RPM. It can only control one of the prior mentioned states at one time. Initially, the actuator generates Rate of Penetration (ROP) by controlling RPM when drilling the pilot hole, and changes to position control for directional drilling.

Since the Hoisting motor (HM) always uses RPM as its input reference and the Top Drive (TD) can use both RPM and position reference, the different input variables are defined as in table 6.1.

Table 6.1: Rig actuators and their respective variables, units and modes.

Motor	Named Variable	Unit	Mode
Hoisting motor	ω_{hm}	RPM	All configurations
TD motor	ω_{td}	RPM	TD-RPM mode
TD motor	ϕ_{td}	Angle	TD-Position mode

6.1.2 System Measurements

To be able to generate input for the actuators, an estimation of the relevant states is needed. To get a good estimation, measurements are used together with the generated inputs in a Kalman filter. It is therefore important to define the measurements from the system, which are shown in Table 6.2.

Table 6.2: Measured variables from the simulation system

Measurement Description	Named Variable	Unit
Hoisting motor RPM	ω_{hm}	RPM
Hoisting motor position	p_{hm}	Meter
TD motor RPM	ω_{td}	RPM
TD motor position	ϕ_{td}	Angle
TD motor torque	T_{td}	Nm
WOB	P_{wob}	N
Downhole orientation	$\phi = [\phi, \theta, \psi]$	rad

6.1.3 Control Objective

Now that the system inputs and measurements are defined, it is needed to define the control objective before defining the relevant states of the system. Comparing the current year's competition objective with the previous year's objective, the reference path for the drill bit may no longer be calculated prior to the competition. Instead, the path should be created during the competition with the given three x -, y - and z -points. These points may create a path that varies both in azimuth and inclination, which gives a more complex position control system, especially with regards to strain on the drill pipe. The controller's goal is to get as close to the given points as possible using downhole measurements.

6.1.4 Coordinate Frames

The system consists of three coordinate frames; the inertial frame, body frame, and drill bit frame. The inertial frame $\{I\}$ is located with its origin at the point where the drill bit tags the rock. The x^I -axis of $\{I\}$ is pointing towards north while the z^I -axis is pointing down.

The body frame is centered at the Inertial Measurement Unit (IMU), and is representing the position and the orientation of the Bottom Hole Assembly (BHA). The z^b -axis is parallel with the BHA pointing downwards. The x^b -axis is pointing straight into the BHA. Initially, when drilling starts, the coordinate frame $\{b\}$ has

the same position and orientation as the inertial frame $\{I\}$.

The drill bit is placed with a constant linear offset to where the IMU is placed. This frame is located at the center of the drill bit and has a z^{db} -axis that points in the same direction as the drill bit, and x^{db} -axis that points in the same direction as x^b . As explained in Section 8.6.2, a bug was encountered when using the frames explained above. To fix this problem, two new coordinate frames were introduced and are collectively called the control-frame. This frame consists of the inertial frame and the body frame rotated -90° around their y -axis'. This is obtained by multiplying coordinates with the rotation matrix

$$\mathbf{R}_y\left(-\frac{\pi}{2}\right) = \begin{bmatrix} \cos -\frac{\pi}{2} & 0 & \sin -\frac{\pi}{2} \\ 0 & 1 & 0 \\ -\sin -\frac{\pi}{2} & 0 & \cos -\frac{\pi}{2} \end{bmatrix}. \quad (6.1)$$

The reason for introducing two new coordinate frames, and not only change the body frame, is to avoid the Gimbal-lock. This happens when the pitch-angle equals 90° , as described in Section 2.1.

Derivation of Position

When calculating the position of the drill bit, it is important to note that all of the position change is contained in the ROP. Since ROP also travels through the x -axis of the drill bit coordinate frame, it is possible to transform this vector into its inertial frame to find position equations for x , y and z . This gives

$$\dot{\mathbf{p}}^{I_c} = \mathbf{R}_{b_c}^{I_c} \mathbf{R}_{db}^{b_c} \dot{\mathbf{p}}^{db}, \quad (6.2)$$

where $\mathbf{R}_{db}^{b_c}$ is the rotation matrix from the drill bit to the control body frame $\{b_c\}$, and $\mathbf{R}_{b_c}^{I_c}$ is the rotation matrix from $\{b_c\}$ to $\{I_c\}$. As shown in Section 6.1.4, the rotation of the ROP-vector from the drill bit frame to the control body frame is given by

$$\mathbf{R}_{db}^{b_c} = \begin{bmatrix} \cos \alpha_{db} & 0 & \sin \alpha_{db} \\ \sin \alpha_{db} \sin \theta_{db} & \cos \theta_{db} & -\cos \alpha_{db} \sin \theta_{db} \\ -\sin \alpha_{db} \cos \theta_{db} & \sin \theta_{db} & \cos \alpha_{db} \cos \theta_{db} \end{bmatrix}. \quad (6.3)$$

The rotations from the control body frame to the control inertial frame are given by the ROLL-PITCH-YAW rotation matrices as introduced in Section 2.1. When multiplying these matrices together it can be shown that

$$\mathbf{R}_{b_c}^{I_c} = \begin{bmatrix} \cos \psi \cos \theta & -\sin \phi \cos \psi + \cos \phi \sin \theta \sin \psi & \sin \phi \sin \psi + \cos \phi \sin \theta \cos \psi \\ \sin \phi \cos \theta & \cos \phi \cos \psi + \sin \phi \sin \theta \sin \psi & -\cos \phi \sin \psi + \sin \phi \sin \theta \cos \psi \\ -\sin \theta & \cos \theta \sin \psi & \cos \theta \cos \psi \end{bmatrix}, \quad (6.4)$$

where ϕ is roll, θ is pitch, and ψ is yaw. Calculating $\mathbf{R}_{b_c}^{I_c} \mathbf{R}_{db}^{b_c}$ gives a very big matrix. An important note here is that the ROP only moves in the x^{db} -axis of the drill bit coordinate frame. This means that

$$\dot{\mathbf{P}}^{db} = \begin{bmatrix} \dot{x}^{db} \\ 0 \\ 0 \end{bmatrix}, \quad (6.5)$$

which gives that the two last columns of the calculated matrix $\mathbf{R}_{db}^{I_c} = \mathbf{R}_{b_c}^{I_c} \mathbf{R}_{db}^{b_c}$ does not matter, as they get multiplied by zero. The ROP orientation in inertial frame is given by equation (6.6).

$$ROP^{I_c} = \dot{x}^{db} \begin{bmatrix} \cos \alpha_{db} \cos \psi \cos \theta - \sin \alpha_{db} (\sin \phi \sin \psi) + \cos \phi \cos \psi \sin \theta \\ \sin \alpha_{db} (\cos \psi \sin \phi - \cos \phi \sin \psi \sin \theta) + \cos \alpha_{db} \cos \theta \sin \psi \\ \cos \alpha_{db} \sin \theta - \cos \phi \sin \alpha_{db} \cos \theta \end{bmatrix} \quad (6.6)$$

The angles ϕ , θ , and ψ are estimated using an Extended Kalman Filter (EKF) with measurements from an IMU.

Drill Bit Location

Since the BHA is a rigid body during the entire drilling operation, it is easy to transform the position of the IMU to the position of the drill bit. The bent sub has a fixed angle α_{db} with respect to the vertical center line. When the bent sub is installed, it will also point in a random direction in the zy_{b_c} -plane, depending on how much torque is used. This angle which the bent sub is pointing with respect to the body frame is denoted θ_{db} . The position of the bit with respect to the body frame $\{b_c\}$ is denoted as $\mathbf{p}_{b_c,db}^{b_c}$. This position is given by the matrix

$$\mathbf{T}_{b_c,db}^{b_c} = \mathbf{T}_1^{b_c} \mathbf{T}_2^1 \mathbf{T}_3^2 \mathbf{T}_4^3 \mathbf{T}_{db}^4 \quad (6.7)$$

$$\begin{aligned}
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -s_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & s_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta_{db} & -\sin \theta_{db} & 0 \\ 0 & \sin \theta_{db} & \cos \theta_{db} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&\begin{bmatrix} \cos \alpha_{db} & 0 & \sin \alpha_{db} & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \alpha_{db} & 0 & \cos \alpha_{db} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & s_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.8}
\end{aligned}$$

$$\begin{aligned}
&= \begin{bmatrix} \cos \alpha_{db} & 0 & \sin \alpha_{db} & s_2 + s_3 \cos \alpha_{db} \\ \sin \alpha_{db} \sin \theta_{db} & \cos \theta_{db} & -\cos \alpha_{db} \sin \theta_{db} & s_3 \sin \alpha_{db} \sin \theta_{db} \\ -\sin \alpha_{db} \cos \theta_{db} & \sin \theta_{db} & \cos \alpha_{db} \cos \theta_{db} & -s_1 - s_3 \sin \alpha_{db} \cos \theta_{db} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.9}
\end{aligned}$$

where s_1 is the distance from the IMU to the center line along the z^{bc} -axis, and s_2 is the distance from the IMU to the bent sub along the translated x^{bc} -axis. s_3 is the distance from the bent sub to the drill bit along the translated and rotated x^{bc} -axis. The position and orientation relations between the IMU and the drill bit can be seen in Figure 6.1.

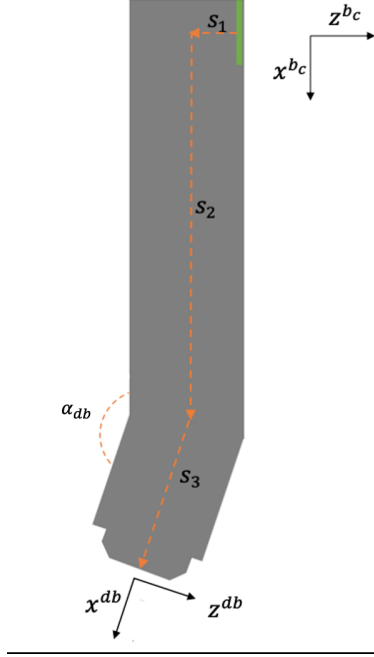


Figure 6.1: The relation between IMU and the drill bit. The center of the coordinate frames are located at the center of the IMU and the center of the tip of the drill bit.

When transforming the ROP from the drill bit frame to the body control frame, one only has to consider the rotation and not the translation since the ROP is a velocity vector. To do this, it is enough to use the upper left 3×3 -matrix of the matrix shown in equation (6.9). For the purpose of simplifying the simulation, it is assumed that the angle $\theta_{db} = 0$. This results in a simple rotation matrix shown by (6.10)

$$\mathbf{R}_{db}^{bc} = \begin{bmatrix} \cos \alpha_{db} & 0 & \sin \alpha_{db} \\ 0 & 1 & 0 \\ -\sin \alpha_{db} & 0 & \cos \alpha_{db} \end{bmatrix}. \quad (6.10)$$

6.2 States and State Dynamics

With no real measurements available, the state dynamics of the system become even more important. As downhole orientation is arguably the most important measurement, one first has to define how the measurement is generated based on the available actuators of the system. Once downhole measurements are available, it is possible to calculate an estimated positional x , y , and z state for the drill bit.

6.2.1 Orientation of Drill Bit

Since the rate of change of the different orientation angles are known, it is possible to estimate the orientation of the drill bit by using the theory presented in Section 2.1.2. Since Euler angles are used, it is not possible to integrate them directly. Firstly, the known drill bit orientation is given relatively to the control frame seen

from the drill bit as shown by

$$\omega_{I_c, b_c}^{b_c} = \begin{bmatrix} u \\ DLS \cdot ROP \\ 0 \end{bmatrix}, \quad (6.11)$$

where u is the input of the top drive, such that it drives a rotation around the x -axis, while $DLS \cdot ROP$ provides rotation around the y -axis. In order to integrate these angles correctly to get

$$\phi = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \quad (6.12)$$

the relation between ϕ and $\omega_{I_c, b_c}^{b_c}$ must be defined. From equation 6.318 in [6], the relation is shown to be

$$\dot{\phi} = \mathbf{E}_d(\phi) \omega_{I_c, b_c}^{b_c} \quad (6.13)$$

where $\mathbf{E}_d(\phi)$ is defined to be

$$\mathbf{E}_d(\phi) = \frac{1}{\cos \theta} \begin{bmatrix} \cos \theta & \sin \phi \sin \theta & \cos \phi \sin \theta \\ 0 & \cos \phi \cos \theta & -\sin \phi \cos \theta \\ 0 & \sin \phi & \cos \phi \end{bmatrix}. \quad (6.14)$$

With these relations, the angles can now easily be found by integrating $\dot{\phi}$. An additional factor for θ is that WOB can provide extra build angle around the y -axis. In absence of a proper model for this dynamic, another solution is to add a value that increases linearly when WOB is greater than the nominal value. A visualization of this can be seen in Figure 6.2.

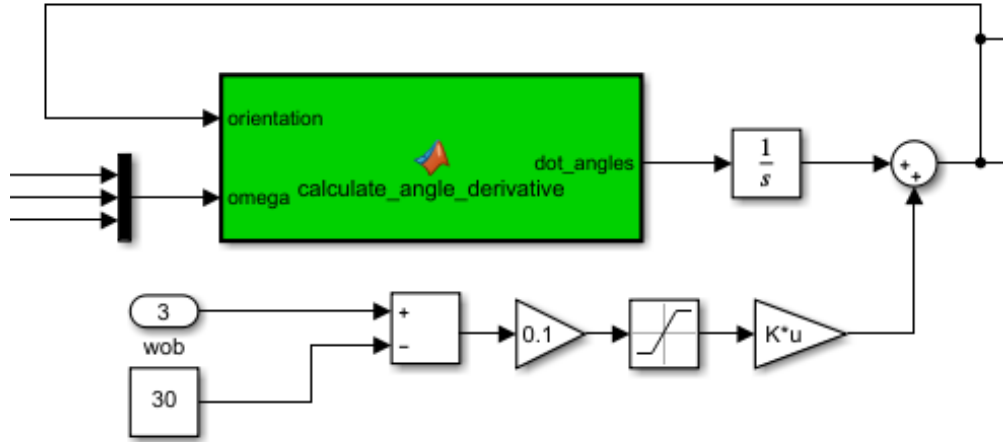


Figure 6.2: Added WOB contribution for θ build.

There can also be added saturation as seen in Figure 6.2. This is done to prevent negative extra build, and also constraining the maximum extra build if the WOB exceeds its limits.

6.2.2 x -, y - and z -position of Drill bit

With the orientation of the BHA, it is now possible to calculate the position of the drill bit based on the orientation and the ROP. Since ROP only works in the direction of one axis relative to the drill bit frame, it can for example be defined as

$$\mathbf{ROP} = \begin{bmatrix} ROP \\ 0 \\ 0 \end{bmatrix} \quad (6.15)$$

when the x -axis of the drill bit is defined to point out of the drill bit. In order to bring the movement generated by the \mathbf{ROP} to the inertial frame, two rotations are needed. First, it must be rotated around the y -axis by α_{db} since this is the bent angle of the drill bit relative to the BHA. After this, the result must be rotated back to the inertial frame by using the calculated orientations from Section 6.2.1. The resulting state dynamics for the positions x -, y - and z - can therefore be found by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \mathbf{R}_{bc}^{Ic}(\phi) \mathbf{R}_y(\alpha_{db}) \mathbf{ROP}, \quad (6.16)$$

where $\mathbf{R}_{b_c}^{I_c}(\phi) = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi)$ as defined in Section 2.1. The position of the drill bit can then be found by integrating the result.

6.2.3 Weight on Bit (WOB) Dynamics

In the absence of a load cell that would be used with the physical rig, one has to model the state given the drilled distance and the position of the hoisting motor. If the hoisting motor position increases with the drilled distance being the same, it means that the WOB should increase. In order to do this, one can take the difference between these two variables, and multiply it by a constant with saturation such that the value of WOB will not go under zero. The load cell can therefore be simulated by equation

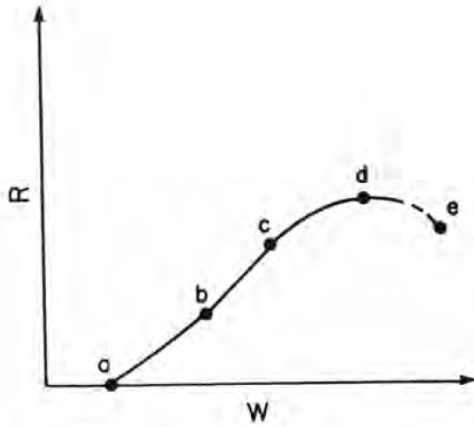
$$P_{wob} = K_{wc}(p_{hm} - d_l), \quad (6.17)$$

where K_{wc} is a constant to make the measured WOB realistic based on the difference between the hoisting motor position p_{hm} and drilled length d_l . This is modeled as a spring, using Hooks law, where the difference between hoisting motor position and drilled length corresponds to the compression of the spring and K_{wc} is the equivalent to the spring constant. The position of the hoisting motor p_{hm} is found by converting the rotational hoisting motor RPM ω_{hm} to the linear velocity by multiplying with the lead and divide it by the circumference of the ball screw. Afterward, this gets integrated, which gives the linear position of the hoisting motor. For the drilled length d_l , the ROP is simply integrated.

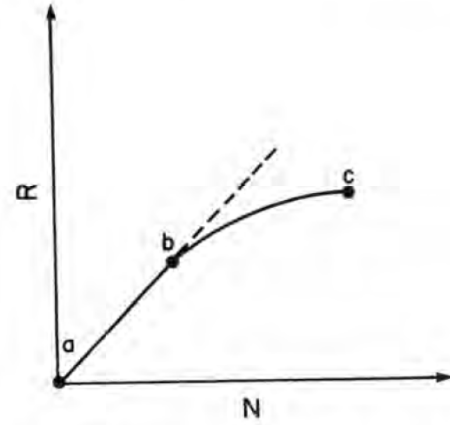
6.2.4 ROP Dynamics

When creating a simulation, accurate models are important. Looking at findings made by the previous years' Drillbotics teams, it was found that by experimentation, the ROP does not change significantly when WOB changes. From Figure 6.3 from [24], one can see the relation between WOB and ROP, and downhole rotary speed and ROP, respectively.

Based on these relations, the dynamics of the ROP have been designed to be quite simple, while also depending on the state machine. It has been designed such that the value of ROP can be in one of two states based on which state the state machine is in. This is because when in some states, one can assume sufficient WOB values, and therefore also ROP values. Therefore, the ROP has a defined positive value when the state machine is in vertical or directional drilling state, and 0 in the other states. When the system is in a drill state, it can be assumed that there is sufficient WOB, and is hovering around the nominal WOB value. The ROP is measured based on the HM Revolutions per Minute (RPM) and the measured WOB. This lies on the assumption that there is sufficient rotational downhole speed when WOB is at the nominal value.



(a) The relation between WOB and ROP.



(b) The relation between rotary downhole speed and ROP.

Figure 6.3: If one assumes constant rotary speed of downhole power, and constant WOB, one can assume constant ROP [24].

6.2.5 Complete State Space Model

With the shown approach of deriving the state dynamics for the different states in the simulation system, it is now possible to define the whole state space system where the following states are considered

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \\ P_{wob} \end{bmatrix} \quad (6.18)$$

where x , y and z are the coordinate points of the drill bit, and ϕ , θ and ψ are the orientation angles of the BHA. Using the equations defined in Section 6.2.1 and Section 6.2.2, the equations for all the states can be written in the following form

$$\dot{\mathbf{x}} = f(\mathbf{x}, u) \quad (6.19)$$

where $f(\mathbf{x}, u)$ consists of the nonlinear functions given in equations (6.20) - (6.26).

$$\dot{x} = -ROP_x [\sin \alpha_{db} (\sin \phi \sin \psi + \cos \phi \cos \psi \sin \theta) - \cos \alpha_{db} \cos \psi \cos \theta] \quad (6.20)$$

$$\dot{y} = ROP_x [\sin \alpha_{db} (\cos \psi \sin \phi - \cos \phi \sin \psi \sin \theta) + \cos \alpha_{db} \cos \theta \sin \psi] \quad (6.21)$$

$$\dot{z} = -ROP_x [\cos \alpha_{db} \sin \theta + \cos \phi \sin \alpha_{db} \cos \theta] \quad (6.22)$$

$$\dot{\phi} = u + \frac{1}{\cos \theta} DLS \cdot ROP \sin \phi \sin \theta \cos \theta \quad (6.23)$$

$$\dot{\theta} = DLS \cdot ROP \cos \phi \quad (6.24)$$

$$\dot{\psi} = \frac{1}{\cos \theta} DLS \cdot ROP \sin \phi \quad (6.25)$$

$$\dot{P}_{wob} = \left(\frac{l}{60d\pi} u - ROP \right) K_{wc} \quad (6.26)$$

where ROP follows the dynamics described Section 6.2.4.

6.3 State Machine

The drilling rig has many states that it can be in. Based on the states, there are several operations that must be done, and the rig must not move to a new state before the different requirements are met. Also, with unexpected events, the rig must be brought to a safe state from which it may continue.

The state machine is divided into two parts, where the first part concerns the initialization of the rig and drilling of the pilot hole, and the second part concerns the directional drilling part of the operation.

6.3.1 Vertical Drilling

As the competition guidelines state that the initial well must be a pilot hole of at least 4", the state machine will have its own separate part dedicated to this demand. The different normal states that the rig may be in, are shown in Figure 6.4.

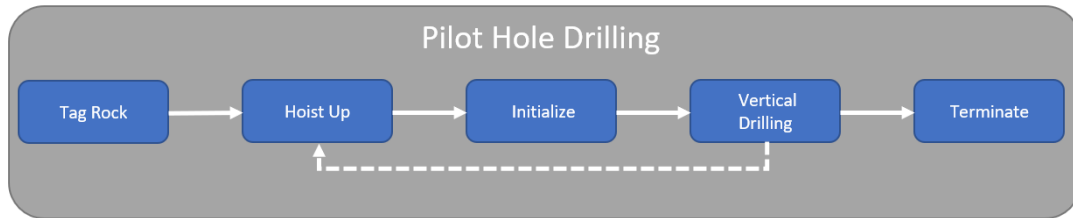


Figure 6.4: State machine for vertical drilling.

As seen, there are five normal states the rig can be in during vertical drilling. The active actuators, measurements and transition requirements in the different states can be summarized as follows:

1. Tag Rock

The drill bit has not yet touched the rock, and therefore will not require any WOB. The hoisting motor pushes the rotary system down towards the rock while there is no other rotation. A figure of the state of the rig can be seen in Figure 6.5a. Once there is registered a WOB greater than a set threshold t_{tag} , i.e. $P_{wob} > t_{tag}$, the state transitions to Hoist Up.

2. Hoist Up

Hoist up is the next state after tag mode, but will also be used whenever vertical drilling needs to be reset due to factors as exceedingly high torque, pressure or WOB. When in hoist up mode, the main purpose is to hoist the rotary system up a height h , as shown in Figure 6.5b. When this is done, the state transitions to Initialize.

3. Initialize

Initialize mode will always come after Hoist up mode. Hoist up mode hoisted the rotary system up a height h , so that it is possible to reach the desired RPM on the drill bit since there is no contact with the rock. The TD and pump motor for the PDM starts, and when sufficient RPM is met, the rig transitions to the vertical drilling state.

4. Vertical Drilling

When in vertical drilling mode, one knows that the previous state was initialize, and that there is sufficient RPM on the TD and PDM. The hoisting motor will start to push the rotary system down, and drilling starts. The WOB, pump pressure, and torque are measured continuously, and if they exceed defined thresholds, the state is immediately transitioned to hoist up. The z^I -coordinate of the drill bit is also monitored, and when $z \geq d_r$, where d_r is the desired depth of the pilot hole, vertical drilling stops, and the state transitions to the terminate state. This transitional state is shown in Figure 6.5c.

5. Terminate

The termination state of the vertical hole drilling part is only reached when $z \geq d_r$. When this happens, the rotary system is hoisted up a height h_{vt} , before it transitions to the first state of directional drilling.

6.3.2 Directional Drilling

After the 4" pilot hole, the rig should drill a well path that follows a pre-calculated trajectory to hit multiple points given by their x , y , z -coordinates in the inertial frame. This is the directional drilling part of the autonomous rig, and its states can be seen in Figure 6.6.

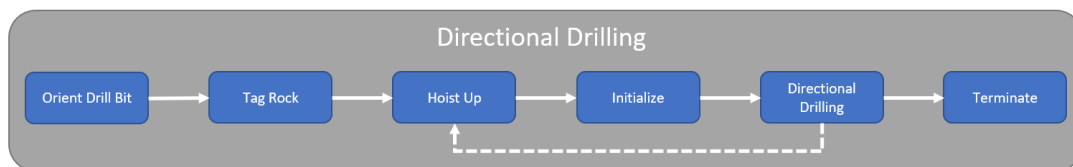


Figure 6.6: An overview of the different states when the system is in the directional drilling mode.

As seen, there are 6 states in the directional part compared to 5 in the vertical part. The extra state is the orient drill bit, as in this case it is needed to orient the drill bit as well before drilling. The active actuators, measurements and transition requirements in the different states can be summarized as follows:

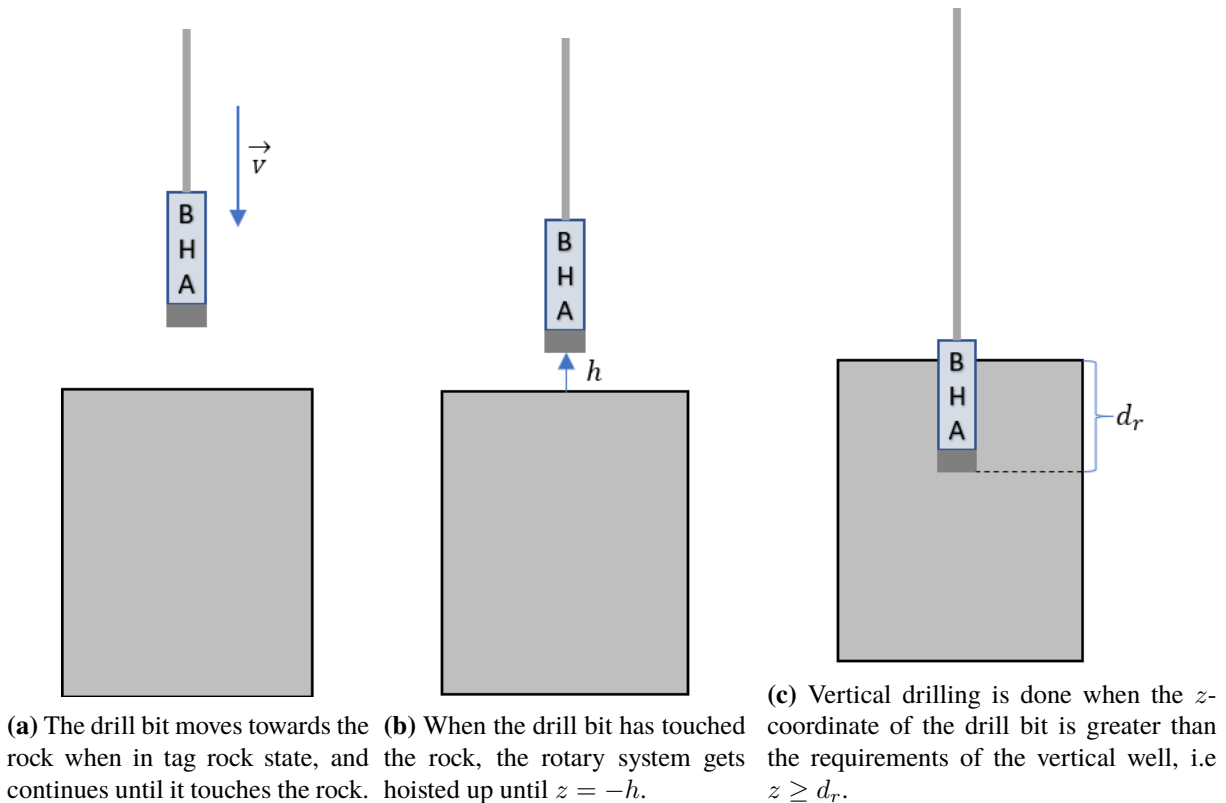


Figure 6.5: Three of the possible states when in vertical drilling mode.

1. Orient Drill Bit

After terminating the vertical drilling part, the rotary system is hoisted up a height h_{vt} over the bottom of the 4" hole drilled by the vertical drilling part. The purpose is to let the control system orient the drill bit in the correct direction without touching the rock. The closest point in the reference path is calculated based on the z^I -coordinate of the drill bit, and a directional vector is calculated, which gives a TD reference position. This is better described in Section 6.4.4. The TD is steered into this position, and then transitions to the Tag Rock state. An example of such a state can be seen in Figure 6.7.

2. Tag Rock

When the drill bit is oriented in the right direction, it will enter Tag Rock mode, which is the same as in the vertical drilling case. The rotary system is hoisted down until $P_{wob} > t_{tag}$, and then transitions to the Hoist Up state.

3. Hoist Up

After the drill bit tags the rock, the state transitions to hoist up, which does the same as in the vertical drilling case. The rotary system is hoisted up a height h , and then transitions to the Initialize state.

4. Initialize

When the drill bit has been hoisted up a height h , it is ready to gain RPM on the drill bit. The difference between the initialize state in directional versus vertical drilling mode, is that with directional drilling,

only the pump motor is used to gain RPM on the PDM. That is, the TD is not used with directional drilling. When the desired RPM is reached, the state transitions to the directional drilling state.

5. Directional Drilling

When in the Directional Drilling state, the RPM of the PDM is sufficient, since the previous state always is initialize. The rotary system is also hoisted up a height h from the hoist up state, so when the state transitions to directional drilling, it starts to hoist the rotary system down while monitoring the different states. The system then uses a directional controller described in Section 6.5 to control the position of the drill bit and WOB. The directional drilling state can transition to one of two states based on the state of the system. If the torque goes over a defined threshold, or the RPM falls under a certain threshold, it will transition to hoist up state, just as with the vertical drilling case. If the position of the drill bit has reached the last goal point, it will transition to the terminate state.

6. Terminate

When terminate state is reached, the well path is done. The hoisting system will then hoist the system up while maintaining a low RPM on the PDM to prevent it from sticking to the curved borehole.

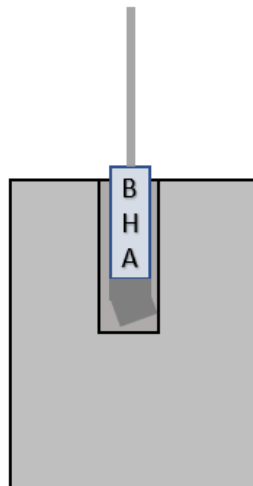


Figure 6.7: An example state when orient drill state transitions to new a state.

6.4 Reference Generation

When creating a controller for the simulation system, it is important to consider the possible references for the different states in the system. One obvious reference scheme is to generate a reference path for the drill to follow. By using a reference path, the drilling operation might deviate from the real control objective, which is to hit the coordinate points as close as possible, but by using a reference path, there are defined control objectives at every time step, which is to be as close to the path as possible. There has been designed three methods of path generation that goes through the given coordinate points, and one method that uses a Nonlinear Model Predictive Controller (NMPC) and cost function to determine the best possible path given

the physical constraints.

The paths are created in the inertial frame, where the x^I -axis is pointing north, the y^I -axis is pointing east and the z^I -axis is pointing down. In Section 6.1.4, two new coordinate frames were introduced. The system requires the paths to be given in the inertial control frame, which is done by one simple rotation

$$\mathbf{p}^{Ic} = \mathbf{R}_y \left(-\frac{\pi}{2} \right)^T \mathbf{p}^I. \quad (6.27)$$

6.4.1 Cubic Spline Interpolation

One way of creating a reference path for the drilling operation is to use cubic spline interpolation. This way, it is possible to get a smooth curve that passes through the given coordinate points. This is done by interpolating each x – y – and z –coordinates. The output from the function is the reference path in all x –, y – and z –directions. An example of a trajectory given by this method is shown in Figure 6.8.

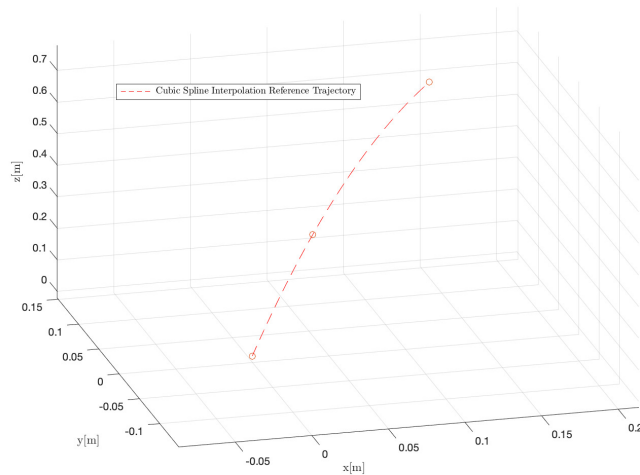


Figure 6.8: A reference trajectory created with cubic spline interpolation.

6.4.2 Circular Trajectory

A problem when using cubing spline interpolation for path generation is that the required angle build rate while drilling may change to follow the path correctly. In reality, this is hard to achieve as the only ways are to either manually change the drill bit angle, or apply extra pressure with WOB for pipe bend. A more feasible path is to have a circle passing through the three given coordinate points. This way, one can assure that the drill should be able to follow the path without the need for a bent sub angle change. An obvious drawback of this approach is if the points are aligned such that the approximated circular path deviates a lot from the reference points.

An example of such a trajectory can be seen in Figure 6.9. As seen, the reference path is cut out from the generated circle trajectory, such that it starts at the first reference point, and ends at the last reference point.

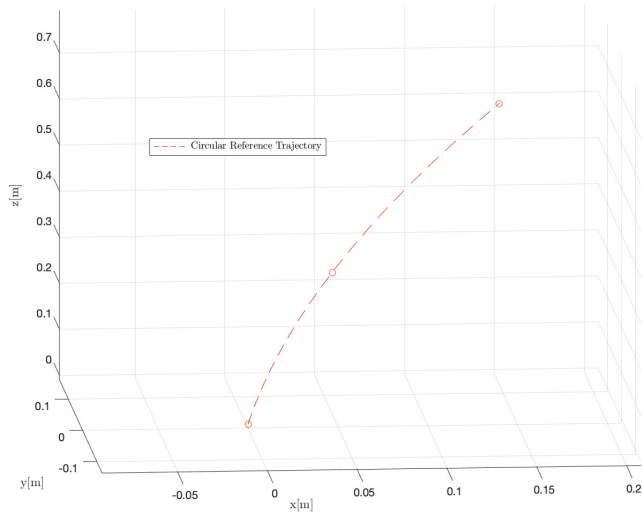


Figure 6.9: A reference trajectory created with circle interpolation.

Finding Optimal Dogleg Severity (DLS) With Circular Path

When it is not possible to interpolate a circle that closely follows the coordinate points, cubic spline interpolation must be used. In this case, the needed DLS might change during the drilling operation, in which a simulation must be done with multiple DLS values to find the best one. In the case where an interpolated circle closely follows the points, a general DLS that can be used for the entirety of the drilling operation may be used. By calculating the DLS from the reference circle path, one can calculate the needed angle of the bent sub before drilling operation starts. The equations used for this are already presented in section 3 in equation (3.1) and (3.4), but will be repeated here for convenience:

$$DLS = \frac{\phi}{CL} \quad (6.28)$$

$$DLS = \frac{2 \alpha_{db}}{L_1 + L_2} \quad (6.29)$$

where CL is the curve length of the path, ϕ is the angle of the path, L_1 and L_2 are the lengths of the BHA and bent sub, respectively, and α_{db} is the bent sub angle. These can better be visualized in Figure 6.10.

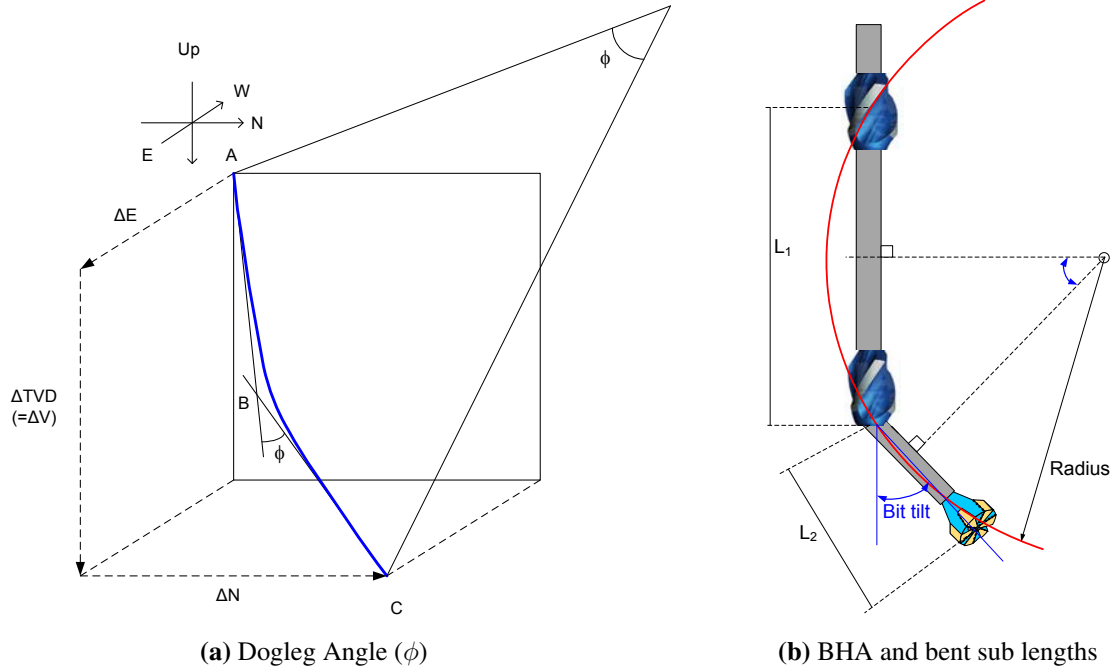


Figure 6.10: Relevant parameters when developing a well path [17].

As all the variables in these two equations are known apart from the bent sub angle α_{db} , the equations can be rearranged to give the needed calculated α_{db} by the following equation

$$\alpha_{db} = \frac{1}{2} \frac{\phi}{CL} (L_1 + L_2) \quad (6.30)$$

6.4.3 NMPC for Optimal Trajectory Generation

When generating the reference trajectory as explained in Section 6.4.1 and Section 6.4.2, the path might not be possible for the system to follow due to mechanical constraints. The trajectories created only considers a set of coordinates. Following an infeasible reference trajectory can make matters worse by taking the system to a state where the system can not follow the rest of the path in a good way. The coordinates to be followed are not known before the day of the competition, which means that the system might not be able to hit all the points due to the mechanical constraints.

A solution to this problem is to create the path using the NMPC. The NMPC computes the optimal p next states, where p is the prediction horizon, using the c next optimal inputs where c is the control horizon. By designing a cost function based on hitting the coordinates, and making the prediction and control horizon large enough to cover the entire drilling operation, an optimal reference trajectory can be computed.

The reference trajectory created by the NMPC will not necessarily hit all the given coordinates but get as close as possible taking the mechanical constraints into account. An example of a reference trajectory created by the NMPC is shown in Figure 6.11. Also, the advantage of creating the reference trajectory by using the

NMPC is that one can extract the optimal cost. By doing this, the correct BHA configuration can be chosen.

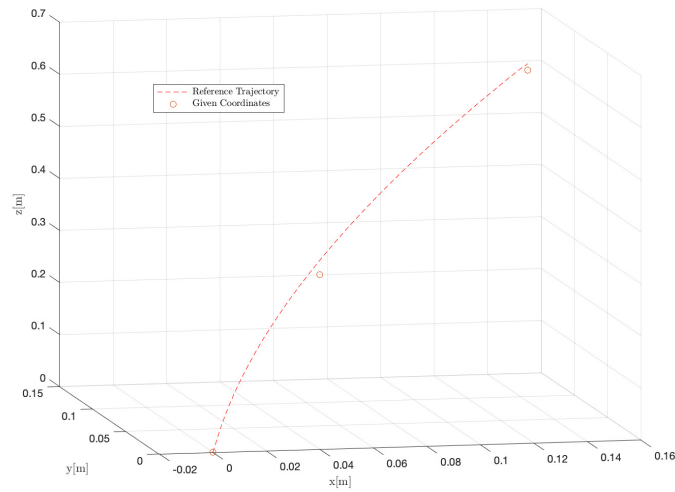


Figure 6.11: A reference trajectory created using NMPC.

Since the coordinates are given at the competition day, different BHA are produced in advance with different configurations with regards to the angle of the bent sub. By using the cost, as well as visual inspection of the simulation plots, the BHA with the best configuration can be chosen. By using an Model Predictive Control (MPC) or NMPC, it would be possible to drill just based on the given points, but the prediction horizon would have to be large. This can result in a computationally expensive and slow controller. By creating the reference trajectory, the prediction horizon can be kept smaller, resulting in a faster controller.

6.4.4 Azimuth Calculation for Angle Reference

Since the orientation of the IMU is known based on the measurements, it is possible to check whether the drill bit frame is oriented correctly with regards to the reference path. An example is given in Figure 6.12, where the drill bit and reference point is given in the y - z -plane in the inertial control frame. First, the system calculates an estimated x^{I_c} -position of the drill bit using the measurements. Then this x^{I_c} -position gets mapped to the reference path to find the corresponding y^{I_c} - z^{I_c} -coordinates of the closest point in the path. By using the estimated y^{I_c} - z^{I_c} -coordinates of the drill bit and the y^{I_c} - z^{I_c} -coordinates of the reference point, it is possible to calculate a directional vector to use as a reference for drill bit orientation as shown in Figure 6.12.

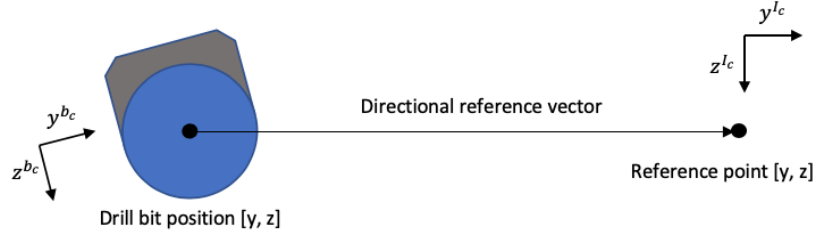


Figure 6.12: Drill bit orientation and reference vector in the y - z -plane seen from above.

One of the objectives for the directional controller is to minimize the cross-track error which is described in equation (6.33). This is done by rotating the x^{bc} -axis. In other words, to get the drill bit to point in the desired direction, the reference angle ψ_r gets calculated. This angle is the only reference used when using a PID controller in the directional drilling phase. This may also be used as a reference when using the MPC and NMPC. When the inclination angle is relatively small, the drilling direction in the yz^{lc} -plane, is the same as the orientation of the drill bit, ψ .

$$e_z = z_r - z \quad (6.31)$$

$$e_y = y_r - y \quad (6.32)$$

$$e_{ct} = \sqrt{e_z^2 + e_y^2} \quad (6.33)$$

After the directional reference vector is calculated, the reference angle can be calculated as shown by

$$\phi_r = \begin{cases} \phi & \text{if } e_y = 0 \text{ \& } e_z = 0 \\ -\text{atan2}(-e_y, -e_z) & \text{otherwise} \end{cases} \quad (6.34)$$

To make the drill bit converge to the reference path as fast as possible, as well as minimizing stress on the drill pipe, the reference angle ψ_r is calculated according to equation (6.35) and equation (6.36). An example is if $\psi = \frac{\pi}{6}$, the reference angle should be $\psi_r = -\frac{\pi}{6}$ not $\psi_r = \frac{11\pi}{6}$.

$$n = \arg \min_{n \in \mathbb{Z}} \{|\phi - (\phi_r + 2\pi n)|\} \quad (6.35)$$

$$\phi_r = \phi_r + 2\pi n \quad (6.36)$$

6.5 Controller Scheme

As the state machine has been designed, there are several control phases that need to be addressed. For this, the design has been divided into two groups, where the first one considers the PID controllers for simple control, for example under a hoist-up operation, as well as using PID for directional drilling. The second group considers the complex control during the directional drilling phase, where multiple variables are considered with the use of an MPC or NMPC.

6.5.1 PID for Simple Operations

When the state machine is located in certain states, the operation is quite simple. These states are Tag Rock, Hoist Up, Vertical Drilling, and Orient Drill Bit as they are defined in Section 6.3. All these states use a simple PID controller with varying control parameters. When the state machines transition to one of these states, the parameters of the PID controller changes to suitable values for that operation. Also, the reference for the controllers at these drilling operations changes based on which state the state machine is in.

Tag Rock

When in tag rock mode, the operation is to lower the rotary system until an excessive WOB is measured to ensure that the drill bit has touched the rock. In this case, the TD RPM is held at 0, while the HM RPM is controlled by a WOB controller.

Hoist Up

The operation of hoist up mode is quite similar to that of tag rock, but in the opposite way. The use of this controller is to hoist the rotary system up a defined height such that the drill bit no longer touches the rock. Therefore, the TD RPM in this case is also set to 0, while the HM RPM reference is held at the negative equivalent of the HM RPM reference in tag rock mode.

Vertical Drilling

When drilling the pilot hole of the well path, both the TD and HM actuators need to be used to gain ROP. Therefore, in this case, both are set to hold constant RPM setpoints that render stable operation conditions, i.e stable WOB while maintaining an adequate speed of operation.

Orient Drill Bit

When transitioning to the directional part of the drilling operation, the drill bit is first oriented in the azimuth direction of the planned trajectory. This is done by using a PID controller with a positional TD reference generated by the method described in Section 6.4.4. The angle is found by using the described method on the two first points of the reference trajectory. The state transitions when $|\phi - o_r| < o_e$, where ϕ is the orientation, o_r is the orientation reference, and o_e is the threshold of which one accepts the orientation to be close enough to the orientation reference.

6.5.2 PID for Directional Drilling

Since both the MPC and NMPC can control multiple variables at the same time, they do not need an independent WOB controller. In the case of using a PID controller for directional drilling, one also needs a controller for WOB. Therefore, the design is based on having one PID controller for angle reference, which is used for the directional part of the drilling, and also one PID controller for WOB reference.

PID for Orientation Control

First, one has to find the control parameters for the orientation controller. Since θ and ψ are small, the following equation can be written

$$\dot{\phi} = \frac{2\pi}{60} u. \quad (6.37)$$

Then, by choosing u to be

$$u = \frac{60}{2\pi} k_p (\phi - \phi_r) + k_i \int (\phi - \phi_r) dt - k_d \dot{\phi}, \quad (6.38)$$

the following dynamics are obtained

$$\dot{\phi} = k_p (\phi - \phi_r) + k_i \int (\phi - \phi_r) dt - k_d \dot{\phi}. \quad (6.39)$$

Taking the Laplace transform, it is possible to obtain the transfer function as such

$$\phi s = k_p (\phi - \phi_r) + k_i (\phi \frac{1}{s} - \phi_r \frac{1}{s}) - k_d \phi s \quad (6.40)$$

$$\frac{\psi}{\psi_r} = \frac{\frac{k_p}{1+k_d} s + \frac{k_i}{1+k_d}}{s^2 + \frac{k_p}{1+k_d} s + \frac{k_i}{1+k_d}}. \quad (6.41)$$

As explained in Section 2.2.3, the equation (6.41) can be compared to the second order system shown in equation (2.22). This leaves

$$\zeta = \frac{k_p}{2(1+k_d)\omega_n}, \omega_n = \sqrt{\frac{k_i}{1+k_d}}. \quad (6.42)$$

Choosing $k_p = 10$, $k_i = 5$ and $k_d = 4$ leaves the closed loop system with $\zeta = 1$ and $\omega_n = 1$. The system is critically damped with the controller bandwidth $\omega_b = 0.64$.

WOB Controller

For the WOB controller, the dynamics are described in equation (6.17). The rate of change can therefore be written as

$$\dot{P}_{wob} = \left(\frac{L}{60D\pi} u - ROP \right) k_{wc} \quad (6.43)$$

$$G = \frac{L}{D\pi} k_{wc} \quad (6.44)$$

$$\dot{P}_{wob} = Gu - ROP k_{wc}, \quad (6.45)$$

where L is the lead and D is the diameter of the ballscrew. By assuming that the ROP can be measured and choosing u to be

$$u = \frac{1}{G} \left(K_p \tilde{P}_{wob} + K_i \int \tilde{P}_{wob} dt - K_d \dot{\tilde{P}}_{wob} + ROP k_{wc} \right), \quad (6.46)$$

one gets the equation

$$\dot{P}_{wob} = K_p \tilde{P}_{wob_r} + K_i \int \tilde{P}_{wob} dt - K_d \dot{P}_{wob}, \quad (6.47)$$

where $\tilde{P}_{wob} = P_{wob_r} - P_{wob}$. The transfer function of the closed loop system can then be written as

$$\frac{P_{wob}}{P_{wob_r}} = \frac{\frac{K_p s + K_i}{1 + K_d}}{s^2 + \frac{kp}{1 + kd} s + \frac{ki}{1 + kd}}. \quad (6.48)$$

By choosing $K_p = 3$, $K_i = 1$ and $K_d = 2$, the system has a the damping factor of $\zeta = 1$ and a natural frequency $\omega_n = 0.33$.

6.5.3 MPC and NMPC for Directional Drilling

As the system transitions to the directional drilling part of the operation, the control becomes more complex as there are multiple dependent variables that need to be controlled. For this, either an MPC or NMPC can be used, where it both controls the HM and TD input dynamically to optimize a cost function based on the state errors and given constraints.

Directional Control

As described in Section 6.4, there are multiple ways to generate reference trajectories for the directional drilling operation. The chosen method generates a reference path which can be used to calculate a cost based on the error of four variables of the system: x -, y - and z - position of the drill bit, as well as the orientation angle ϕ generated by the method described in Section 6.4.4. This means that at each time step a reference value x_r , y_r , and z_r are extracted from the generated path, while there also is calculated a ϕ_r based on the current ϕ and the angle between the current position and reference path.

With regards to following the path, the TD is the most important actuator as it controls the azimuth of the drill bit. However, the HM may also be used for extra DLS build if this is needed, which in turn affects the WOB of the system.

WOB Control

While controlling the path of the drilling operation, it also is important to consider the WOB for stable and realistic drilling operations. The reference WOB is held at a constant value for the whole drilling operation, but may exceed this value if it gives better cost results.

State Weights and Costs

Both the MPC and NMPC takes customizable weights for each of the states that are used to calculate the cost. For directional path following, the positional components of the drill bit are penalized the further away from the positional references gotten from the reference path. Also, the error from the nominal WOB reference is penalized by its corresponding weight.

For the input, there is also included a cost. This is usually done by enforcing a cost based on the rate of which the input changes. An example is that one often does not want the TD to go from one angle to the next in a rapid movement. Therefore, one has rate input weights both on the TD input, and the HM input.

Constraints

Physical limitations in the actuators make it infeasible to do infinite inputs with the actuators. These constraints are also included by setting constraints for both the values that they can take, as well as rate limitations based on how fast the actuators should be able to act.

Prediction and Control Horizon

As described in Section 2.3, the prediction and control horizon are important parameters to configure for an accurate and stable system. To be able to account for large changes in the reference path, one needs a prediction horizon that can cover these dynamics. Given the constraints of the competition coordinate points, the reference path is usually smooth and predictable enough for a prediction horizon that is not that large.

6.6 State Estimation

In the current system, there are limited ways to measure the state of the drill. Measurements used to define the position of the drill, are measurements giving information about the orientation of the drill bit as well as the velocity. The orientation is given by an IMU, and the velocity is given by the hoisting motor RPM. Since there is no way of measuring the position, it has to be estimated.

Estimating the position is done by using an EKF. In addition to state estimation for the position, the EKF is used as a filter for the measurements. The drilling process consists of a lot of vibrations and noisy measurements, which means that raw data is not very useful for the controllers, and using the raw measurements and integrating the velocity vector would give a very poor estimate of the position.

6.6.1 State Transition Using the Extended Kalman Filter(EKF)

A part of the EKF is to predict the future state which corresponds to the prior estimate. Finding the discrete quantities are done by applying forward Euler integration to the system described Section 6.2.5. This gives the prediction model [10]

$$\bar{\mathbf{x}}(k+1) = \hat{\mathbf{x}}(k) + h\mathbf{f}(\hat{\mathbf{x}}(k), \mathbf{u}(k)). \quad (6.49)$$

Since there is no way of measuring the position, the posterior estimate of the position will equal the prior estimate, in other words, there are no corrections to the prediction. This means that if there are any biases in the angular measurements, the estimated position of the drill bit will drift from the real position.

The error covariance matrices \mathbf{Q} and \mathbf{R} are of the dimensions 6×6 and 3×3 . The states included in the Kalman filter are position and orientation, where the measurements are the orientation measurements. The ROP is not represented as a state in the EKF, but instead as an input to the state model. This is done because

the dynamics of the ROP is simplified, as explained in Section 6.2.4.

To be able to estimate all the states, the system needs to be observable. To find observability for a nonlinear system is not as easy as it is for a linear system. An indication that the nonlinear system might be locally observable is if the linearized system is observable by finding the observability matrix and confirming that the matrix has full rank. The system might be observable even if the linearized system is not, and vice versa. The **A**-matrix of the system linearized around 0 and the **C** matrix can be written as

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & -0.0008 & 0 \\ 0 & 0 & 0 & 0.0008 & 0 & 0.0100 \\ 0 & 0 & 0 & 0 & -0.0100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0067 & 0 & 0 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (6.50)$$

which yields the observability matrix

$$\mathcal{O} = \begin{bmatrix} 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.000060 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 \\ 0 & 0 & 0 & 0.0007 & 0.0007 & 0 \\ 0 & 0 & 0 & -0.0007 & 0 & 0 \\ 0 & 0 & 0 & 0.0067 & 0.0001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (6.51)$$

The observability matrix indicates that the system is not observable. However, this will be further discussed in Section 8.5.1.

7 Implementation of Simulation System

As the desired functionality and design of the simulation system has been presented in section 6, a clear vision for the implementation has been done. The design has been used to implement a simulation system in MATLAB and Simulink, which will be presented in this section. Since the system uses both MATLAB and Simulink, there is written code in MATLAB which runs together with the blocks given in the Simulink file. It is also important to note that the system has been implemented with interchangeability and modularity in mind, such that testing of new solutions is a simple task.

7.1 System Overview

Firstly, an overview of the whole simulation system will be presented as it consists of several blocks that interact with each other. An overview of the system can be seen in Figure 7.1, where the main Simulink blocks are presented. All of the code can be found in the appendix.

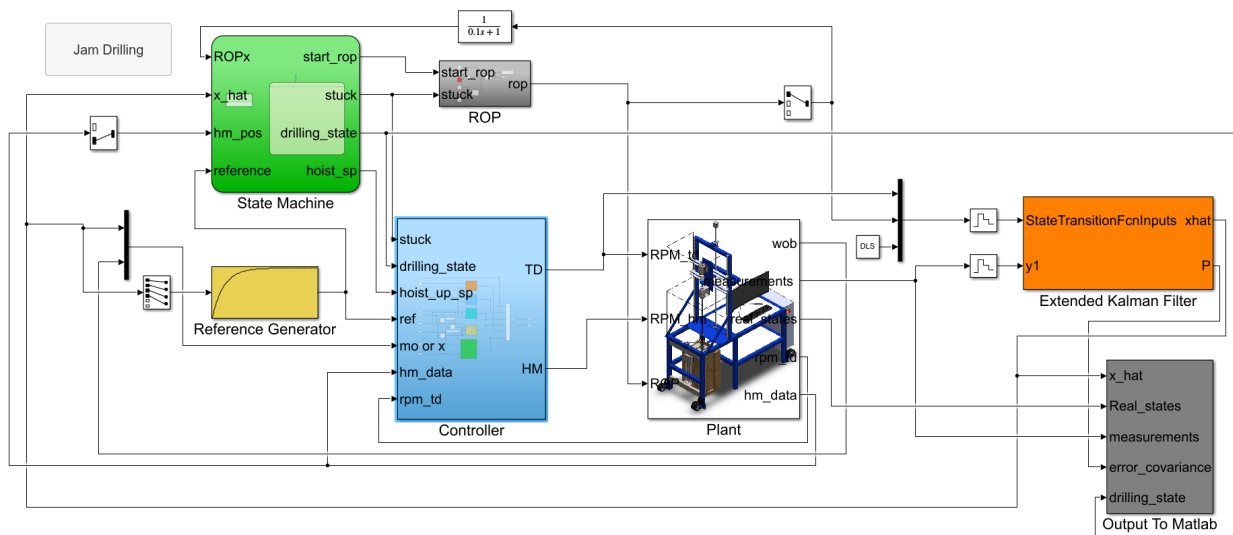


Figure 7.1: An overview of the simulation system in Simulink.

As can be seen, the system is compartmentalized into 7 separate main blocks and 1 button. The functionalities of the different blocks can be summarized as follows

- **State Machine**

Implements the state machine that ensures a safe drilling environment. Uses measurements to determine which state the system is in, and changes the used controllers based on this. If Rate of Penetration (ROP) falls under a given threshold, the state machine will initiate a recovery sequence.

- **Button: Jam Drilling**

Button used under simulation to simulate a loss of downhole power which in turn gives low ROP and high Weight on Bit (WOB). This transitions the state machine to the recovery sequence as the ROP falls under a given threshold.

- **Reference Generator**

Responsible for generating reference values for the controller. This can be divided into several parts, where it both extracts the correct x_r -, y_r - and z_r -coordinates from the reference path given the current time step, as well as it creates a reference angle reference ϕ_r based on the design principles presented in Section 6.4.4. This block only generates references when in directional drilling mode since the references are constant otherwise.

- **Controller**

Contains all the controllers that are used at different phases of the drilling operation. There are 4 PID controllers for the simple operations as described in Section 6.5.1, and 1 directional controller that contains 3 alternatives for directional drilling as presented in Section 6.5.3. The system switches between the different controllers by using the `drilling_state` given by the state machine. The reference values for the simple operation controllers are contained in this block, while the reference values for the directional controllers are generated by the reference generator.

- **ROP**

Either outputs 0 or the defined ROP value based on the state machine. This is justified by the system knowing which state it is in, which makes it possible to assume sufficient WOB, and therefore ROP as discussed in Section 6.2. If the Jam Drilling button is pressed, the ROP is set to 0 for a brief moment to trigger the recovery sequence.

- **Plant**

Includes all the state dynamics of the system as described in Section 6.2, as well as the simulated sensors of the system. This block is therefore responsible for updating the states of the system based on the input that is generated by the controller, as well as giving measurements of these states through simulated sensors.

- **Extended Kalman Filter**

Takes the generated input from the controller as well as the measured orientation of the Bottom Hole Assembly (BHA) to estimate the orientational and positional states of the system.

- **Output To MATLAB**

Converts the outputs from the different blocks such that it can be interpreted properly in MATLAB for data analysis and plotting.

The connections between the blocks sometimes utilize the `Selector`-blocks, which extracts the relevant signals from a multidimensional input signal. An example of the usage is shown in Figure 7.2, where the positional x -, y -, z -coordinates and roll angle ϕ are extracted for reference calculation.

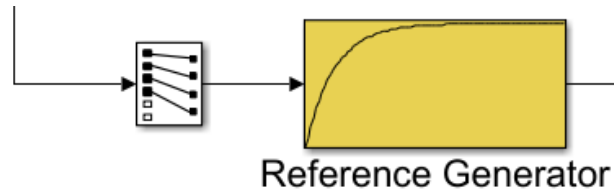


Figure 7.2: Selector is used for signal extraction from multidimensional input signals.

7.2 Reference Generator

As the directional drilling part of the operation carries the most complexity, it requires a reference generator that can calculate new real-time reference values for the positional coordinates of the drill bit, as well as a relative reference angle for the Top Drive (TD) such that the drill bit builds the well path towards the reference path. This is accomplished by the block diagram shown in Figure 7.3.

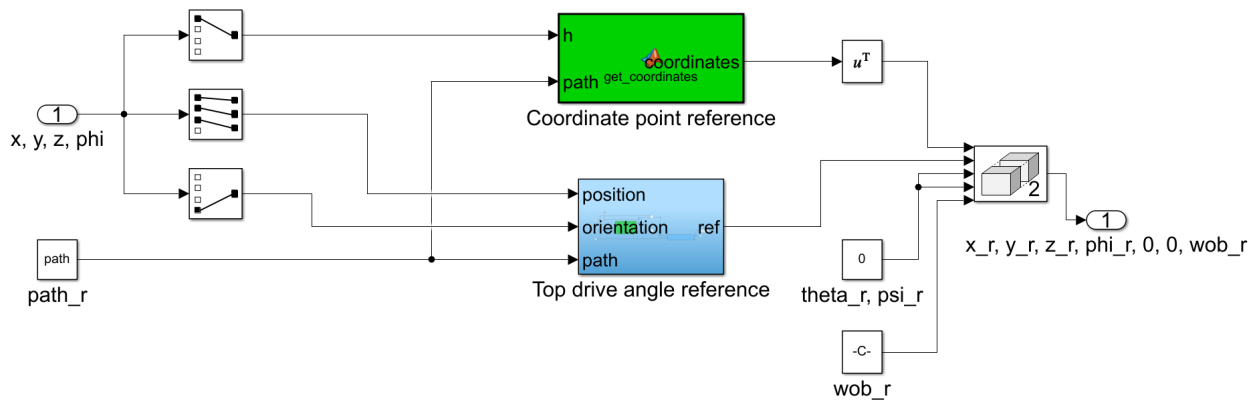


Figure 7.3: The main goal of the reference generator is to generate real-time coordinate point references as well as real time angle reference.

The green block extracts the closest coordinate point from the reference trajectory based on the position of the drill bit. It uses the current height of the drill bit, which with the current coordinate frame system corresponds to the x -coordinate of the drill bit.

As there are no control actions for the pitch and yaw motion of the BHA, the reference angles for these are set to 0. The WOB reference value is also set to a constant value based on the optimal conditions for drilling.

7.2.1 Reference Path

Both the reference coordinate point and reference angle are generated based on the reference path. As described earlier in Section 6.4, there has been designed 3 methods for trajectory generation. These methods have been implemented as functions in MATLAB, and are chosen at initialization of the simulation program by the following code

```
1 trajectory_type = 0; %NMPC=2, Cubic spline=1, Circular trajectory=0.
```


Cubic Spline Interpolation

If cubic spline interpolation method is used, the path gets generated by using the function `path = cubic_spline(p1,p2,p3,t)`, which takes the given coordinate points as well as the timestamps of those points as input. The function then interpolates between these points, and outputs a smooth curve that intersects all 3 points.

Circular Interpolation

To generate a circle that interpolates the three given coordinate points, two functions are used, and are shown below

```
1 function [center,rad,v1n,v2nb] = calculate_circle_parameters(p1,p2,p3)
2 function [points, radian] = calculate_circle_points(center,normal,
    radius)
```

The first function `calculate_circle_parameters(p1,p2,p3)` is from [39] and takes the three coordinate points p_1 , p_2 and p_3 , and outputs the center and radius of the circle, as well as two basis vectors for the circle. The second function `calculate_circle_points(center,normal,radius)` takes the found center and radius, as well as the normal vector of this circle. This is found by creating a normal vector `norm = cross(v1n, v2nb)`; from the two basis vectors. The first output `[points]` from the second function are the points for the circle in three dimensions. The second output `[radian]` is the radian of the curve that is cut out from the circle and used for the reference path.

NMPC

The Nonlinear Model Predictive Controller (NMPC) is implemented as explained in Section 7.3.3. If one picks a sufficiently large prediction horizon and control horizon as explained in Section 6.4.3, and run the function in Figure 7.12 with the input argument `create_path = 1`, the code in Figure 7.4 will run.

```
1     [~,~,info] = nlmprmove(nlobj,x0,u0,ref);
2     path = info.Xopt(:,1:3)';
3     cost = info.Cost;
```

Figure 7.4: Creating optimal trajectory by running the NMPC algorithm once with prediction and control horizon large enough to cover the entire drilling operation.

The `nlmprmove` functions takes the NMPC object, initial values and the reference, and runs one iteration. The cost function used in this case is a function that returns cost only based on how close the optimal trajectory is to the given coordinates. As stated in Section 6.4.3, the cost determines which BHA configuration to use.

7.2.2 Angle Reference

A good short term objective for the drill bit is to orient itself towards the reference path. This is implemented as the blue block shown in Figure 7.3, and includes the blocks shown in Figure 7.5.

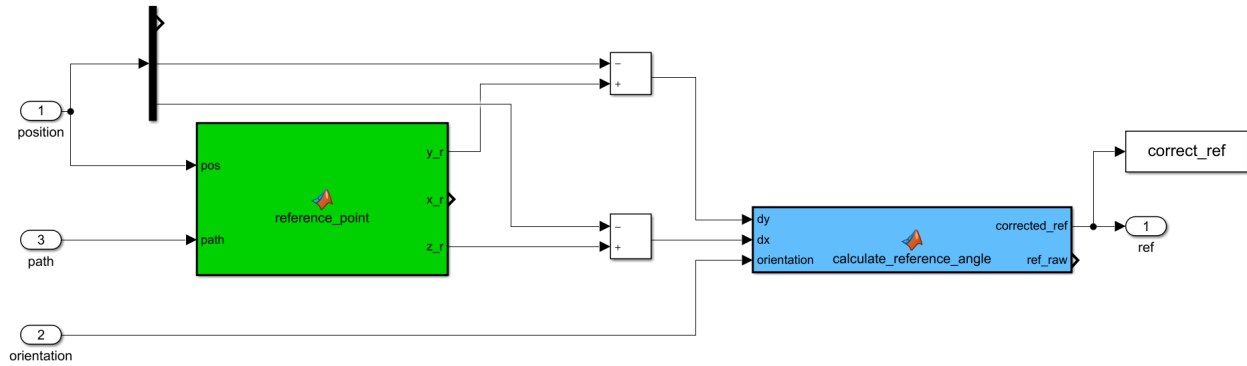


Figure 7.5: The angle reference is found by using current position and closest reference point for a relative angle between current drill bit orientation and needed drill bit orientation.

The algorithm is based on the design presented in Section 6.4.4, and extracts the closest reference point to the current position of the drill bit. It then uses this information to create a directional vector between the current position and reference position, which can be used to find the needed orientation to steer the drill bit towards the path. The needed orientation is found to be in the interval $[0, 2\pi]$, while the current orientation may be outside of these bounds. Therefore a check is done by the following code to ensure that the closest reference is used:

```

1 N = -10:10;
2 ref_raw = - atan2(-dy, -dx);
3 shifted_ref = ref_raw + 2*pi*N;
4 [d, ix] = min(abs(shifted_ref - orientation));
5 corrected_ref = shifted_ref(ix);

```

7.3 Controller Scheme

As described in Section 6.5, the control operation is mainly divided into two parts, where the first one can be considered simple operations, and the second is the complex directional control operation of multiple variables. The controller can be seen in Figure 7.6.

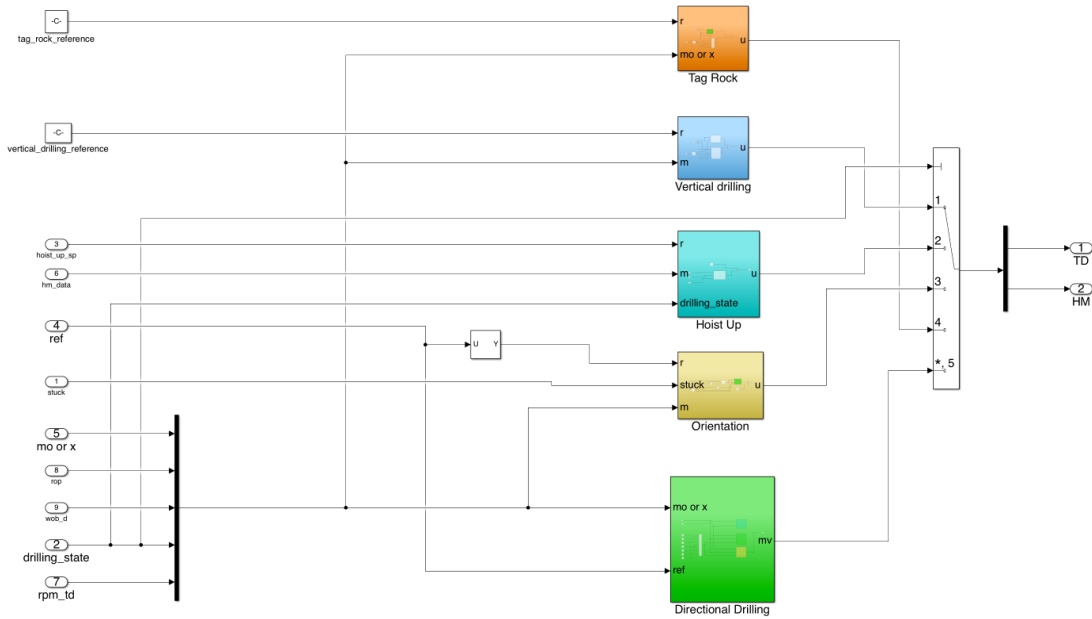


Figure 7.6: The controller scheme consists of 4 PID controllers for the simple operations, and a directional controller that can be customized to use MPC, NMPC or PID.

As seen, the controller consists of 5 blocks, where the first 4 are simple PID controllers for the earlier mentioned simple operations. The last block consists of the 3 possible controlling methods for directional drilling: MPC, NMPC, and PID. The drilling state output from the state machine is used to determine which of the controllers should be used at each iteration by a multiport switch, which then outputs the generated input for both the TD and Hoisting motor (HM).

For the directional controllers, there are as mentioned 3 options as seen in Figure 7.7. Both the MPC and NMPC use weights to determine the cost of errors in the different states as seen in the left of the figure. It is important to note that the PID controller only uses the dynamic angle reference, while the MPC and NMPC use both the dynamic angle reference, as well as the positional coordinates of the reference path.

It is possible to use the same PI/PID controllers in the different stages of the drilling operation. However, the controllers are implemented individually, such that it is easy to get an overview of which part of the control system is running at each stage.

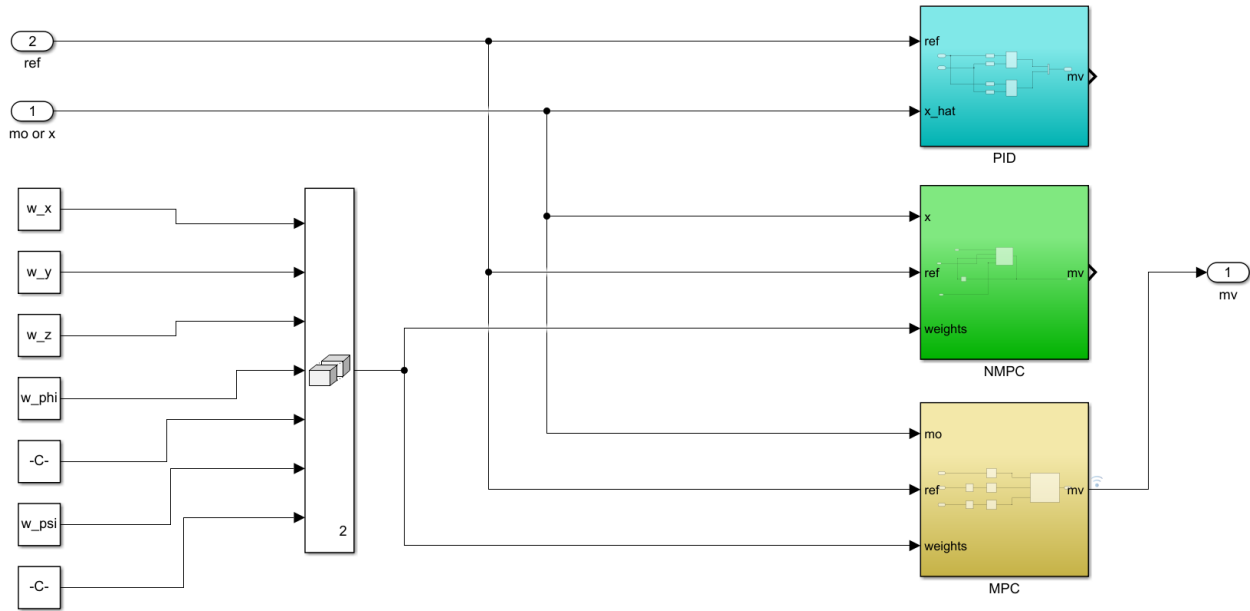


Figure 7.7: There are 3 directional controllers to choose from: PID, NMPC, MPC.

7.3.1 PID Controllers for Simple Operations

For the operations that are intermittently needed under circumstances where drilling has to be reset, simple PID controllers are used. An example of a PID controller used is the initial orientation PID, where the TD reference orientation is set to the calculated initial orientation before directional drilling starts. In this case, the HM input is set to 0. The controller can be seen in Figure 7.8, where the green block is the PID for TD position, and HM input is set to 0.

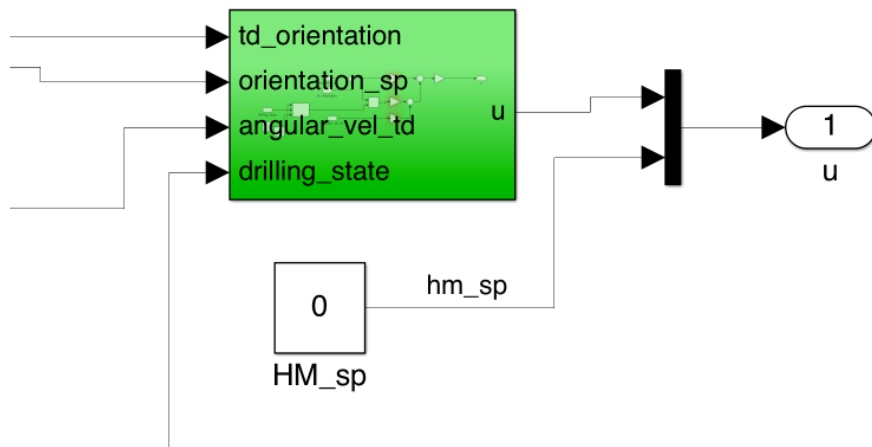


Figure 7.8: The PID controller for initial orientation controls only the top drive position, and sets HM input to 0.

The parameters of the PID, i.e K_p , K_i , and K_d are easily editable in MATLAB, as well as the constant references used.

7.3.2 MPC for Directional Control

In order to design a simulation program using a Model Predictive Control (MPC) to control the system, an MPC-block is used with documentation found in [40], and seen in Figure 7.9. The MPC controller block can take three inputs, which are the measured output signal (mo), reference signal (ref), and error cost weights. The block then outputs an optimal control input that minimizes the state error costs by solving a quadratic problem using the KWIK solver [41].

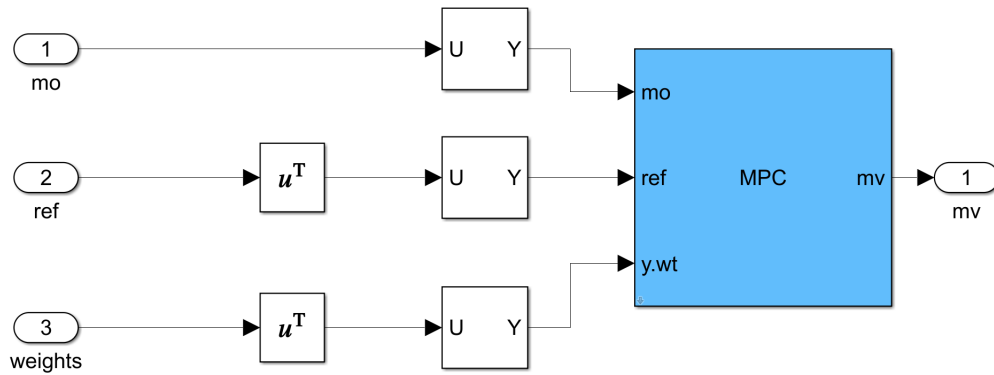


Figure 7.9: The used MPC block from the model predictive control toolbox by MATLAB.

The sample time, number of manipulated variables (mv), measured outputs (mo), and prediction horizon must be defined. This is done in the default conditions tab of the block as seen in Figure 7.10.

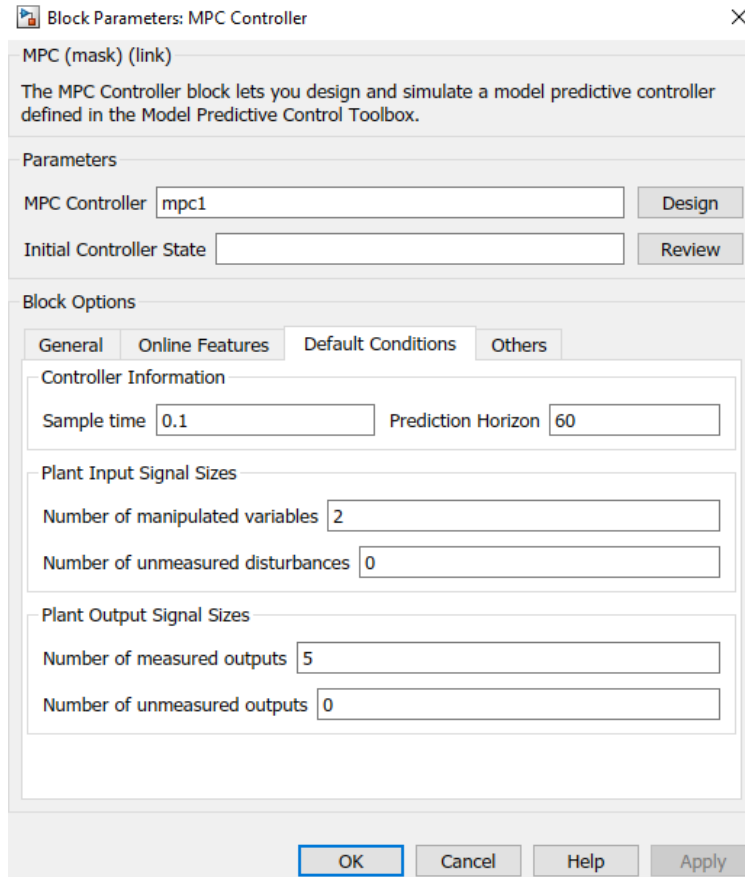


Figure 7.10: Editing default conditions for the MPC block.

When designing an MPC, it is important to define the different constraints and penalty weights for the states as the controller optimizes based on the cost defined by these parameters. The constraints of the manipulated variables reflect the constrictions of the input the plant can handle. These parameters can easily be edited inside the MPC block as shown in Figure 7.11.

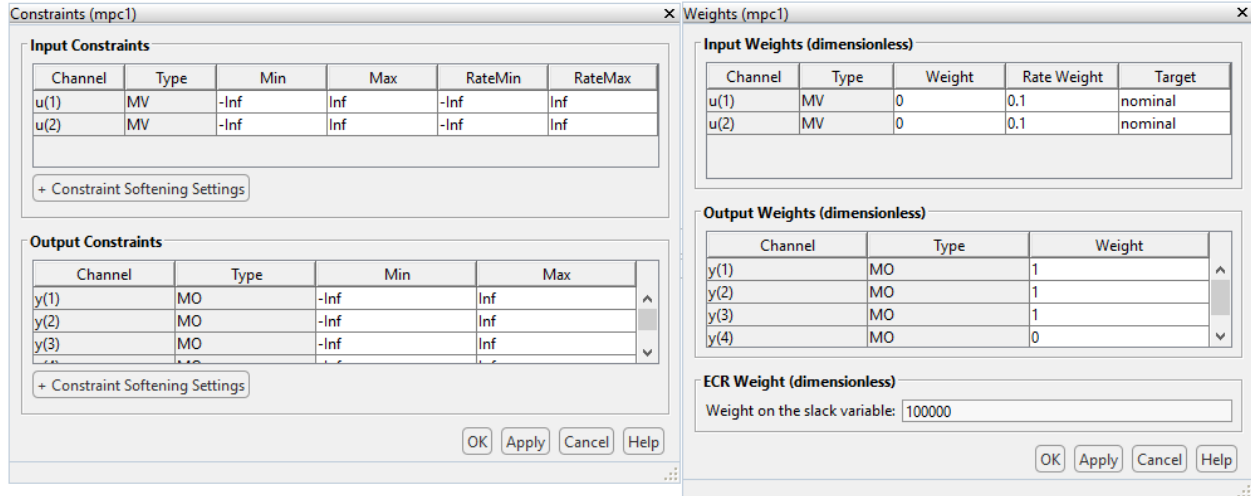


Figure 7.11: The constraints and weights can be edited inside the MPC block.

7.3.3 NMPC for Directional Control

The design of the NMPC is done by using the MPC-toolbox in MATLAB. The toolbox includes an NMPC-object containing variables such as prediction and control horizon, number of inputs, number of outputs, and member functions such as cost function, state function, and state Jacobian function. When implementing an MPC in Simulink, the linearization is done by the toolbox, and it uses a default cost function. When creating an NMPC-object, these functions need to be implemented. The different functions and parameters are specified for the NMPC as shown in Figure 7.12.

```

1 function [NMPC,path,cost] = PlanningControlDirectionalDrilling(nx,ny,nu
    ,p,c,Ts,create_path,initial_orientation,pts)
2 %Takes arguments nx,ny,nu,p,c,Ts,create_path
3 %And returns an nmpc object
4 %if create path is set to 1, the optimal reference trajectory and path
    will
5 %be returned
6
7     nlobj = nlmpc(nx,ny,nu); %creating the nonlinear mpc object
8     nlobj.Model.StateFcn = "DrillingStateFunction";
9     nlobj.Jacobian.StateFcn = @DirectionalDrillingStateJacobian;
10    nlobj.Ts = Ts;
11    nlobj.PredictionHorizon = p;
12    nlobj.ControlHorizon = c;
13    nlobj.MV(1).Max <= 0.05;
14    nlobj.MV(1).Min >= -0.05;
15    nlobj.Weights.ManipulatedVariablesRate = [0.1 0.1];
16    x0 = [0.2;0;0;initial_orientation;0;0;0];
17    u0 = zeros(nu,1);
18    nlobj.Optimization.CustomCostFcn = "customCostFunction";
19    validateFcns(nlobj,x0,u0);
20    NMPC = nlobj;

```

Figure 7.12: Function for creating the NMPC object where nx is the number of states, ny is the number of outputs, nu is the number of manipulated variables, p and c is the prediction and control horizon and T_s is the sampling time.

As seen, there are many variables to be defined, where nx is the number of states, ny is the number of output and nu is the number of manipulated variables, and p and c is the prediction and control horizon, respectively. The custom functions defined by `DrillingStateFunction`, `DirectionalDrillingStateJacobian` and `customCostFunction` can be found in Appendix, in Section A.3.

7.3.4 PID for Directional Control

This controller consists of two PID controllers. The first PID controller controls the TD as described in Section 7.3.1 for the initial orientation. This is one of the cases where the same controller could be used in different drilling states as explained in Section 7.3. This controller takes in a reference angle ϕ_r , and computes the error based on the Kalman filtered angle ψ . The second PID controller controls WOB by taking measurements from the weight cell and comparing it to a constant reference, which then produces an RPM reference to the hoisting motor. This controller uses the same orientation controller as orient drill bit

and the same WOB controller as when the pilot hole is drilled. The implementation of these controllers can be found in the appendix in Figure C.22 and Figure C.23.

7.4 Plant

The plant block of the system contains the logic for calculating the next orientation and position of the system based on the previous states and inputs. As well as this, the block implements the load cell for WOB measurement, and the Inertial Measurement Unit (IMU) for orientation measurement. An overview of the plant can be seen in Figure 7.13.

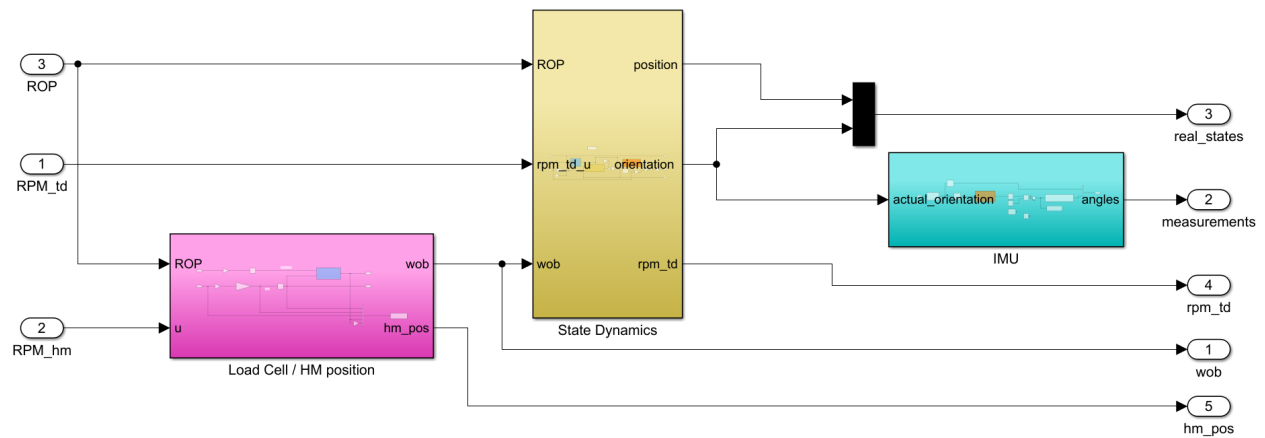


Figure 7.13: An overview of the plant shows the load cell, IMU and state dynamics.

As seen, in the absence of a real IMU, the orientation state is found by the state dynamics as described in Section 6.2.1, which is then fed into the IMU which creates realistic orientation measurements. Also, the HM position is found in the pink block by translating the RPM_hm to linear velocity, and then integrating it.

7.4.1 Orientation and Position Calculation

Arguably the most important part of the simulation system is an accurate representation of state dynamics such that the simulation represents the real-world dynamics. In the yellow block shown in Figure 7.13, these state dynamics are implemented, and shown in Figure 7.14.

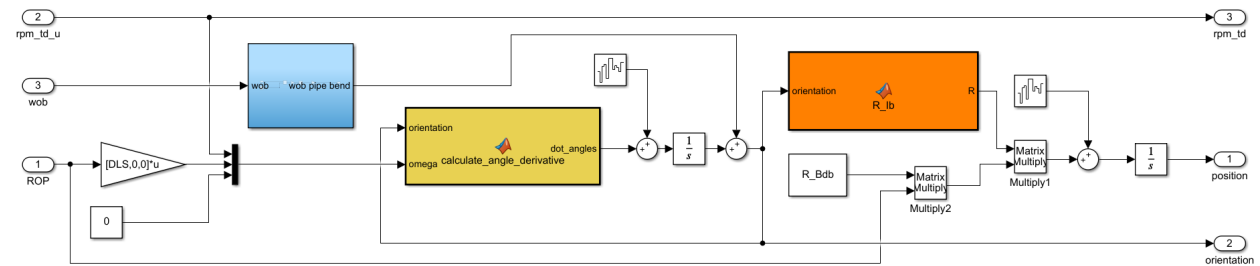


Figure 7.14: The state dynamics include white noise for realistic drilling environment.

The yellow block is implemented according to the design presented in 6.2.1, and uses the ω as defined in equation (6.11) to calculate the orientation of the BHA with the use of equation (6.13) and (6.14). The resulting output is then added together with process noise before integration. It is worth mentioning that there is a possibility to somewhat control extra build in the pitch angle by providing extra WOB. This dynamic is included by the blue block, which gives extra pitch build when WOB exceeds its nominal value.

Now that the orientation has been calculated, the method for position calculation presented in Section 6.2.2 can be utilized. First, the simple rotation matrices are used together with the found orientation to calculate the $\mathbf{R}_{b_c}^{I_c}(\phi)$ matrix. This gives the rotation from the inertial frame to the frame of the BHA, which then is multiplied with the $\mathbf{R}_y(\alpha_{bha})$ matrix to account for the constant bent sub angle before the drill bit. With these matrices, the derivative of the position can be found by using equation (6.16). Together with process noise, this gets integrated such that the x -, y - and z -coordinates of the drill bit are known.

7.4.2 Load Cell for WOB Measurement

Without a load cell sensor to measure the WOB, it must be simulated instead. This implementation is located inside the pink block in Figure 7.13, and is shown by Figure 7.15.

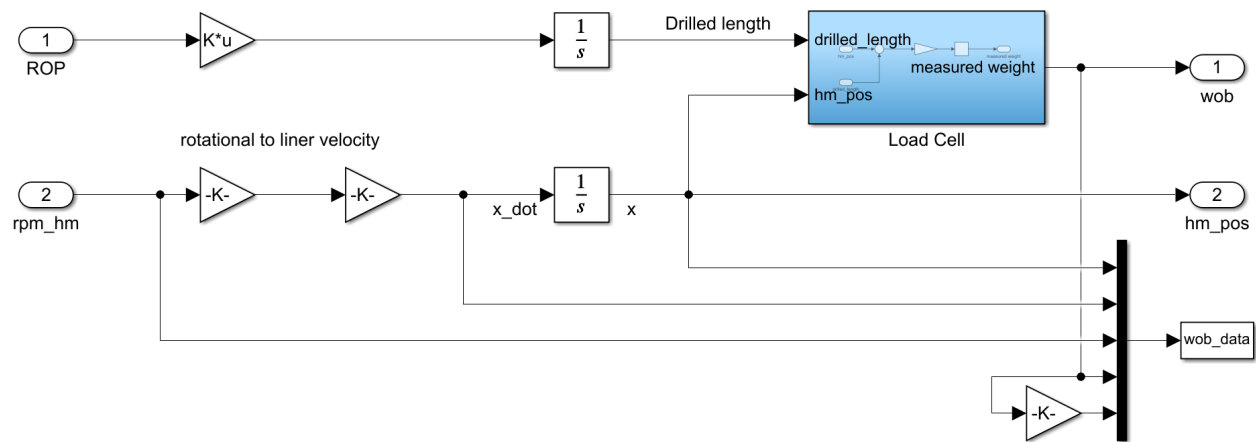


Figure 7.15: The load cell uses a spring mechanic to measure WOB.

In order to simulate the generated WOB that a load cell would measure, a spring mechanic is implemented in the blue block as defined in Section 6.2.3. Firstly, it takes the hoisting motor position found by integrating the rpm_hm , and subtracts the drilled length given by integrating the ROP. In theory, this difference should correlate with the actual WOB, and is therefore multiplied with a gain K_{wc} and saturated such that the WOB is constrained between $[0, \infty]$.

7.4.3 Implementation of Inertial Measurement Unit in Simulink

The implementation of the measurement unit can be seen in Figure 7.16. To find θ and ψ , the acceleration g is decomposed from inertial frame to find its components in the body frame in order to model the output of

an accelerometer. Noise and bias can be added to the output before computing the resulting angles. In the case of physical drilling, the angle ϕ is obtained by using a magnetometer. In this simulation system, this is simplified by feeding the actual orientation together with adding measurement noise and bias.

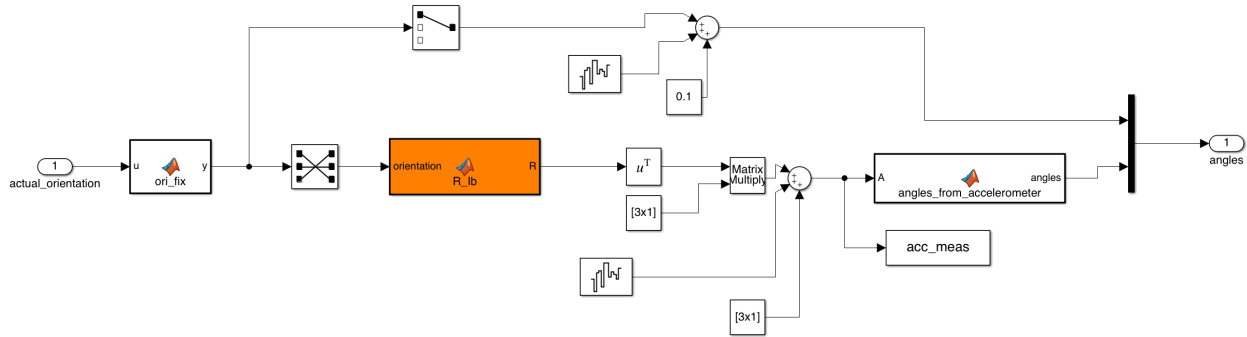


Figure 7.16: Implementation of the IMU in Simulink.

7.5 Kalman Filter for State Estimation

In the real-life environment, the states will not be known directly. Instead, there will be measurements with noise, which has to be simulated in this case. Therefore, after the states are calculated by the plant, noise is added to simulate a real working environment. In order to better estimate the states based on these simulated measurements, an extended Kalman filter is added. The Kalman filter is implemented using a Extended Kalman Filter (EKF) block in Simulink, from the Control System Toolbox.

7.5.1 Inputs to Kalman Filter

The inputs to the Kalman filter consist of the manipulated variables created by the controller, as well as the measured output calculated by the IMU-model. Since the block is a discrete EKF, the inputs to the EKF must be discretized before being fed to the block. This is done by using a zero-order hold block [13].

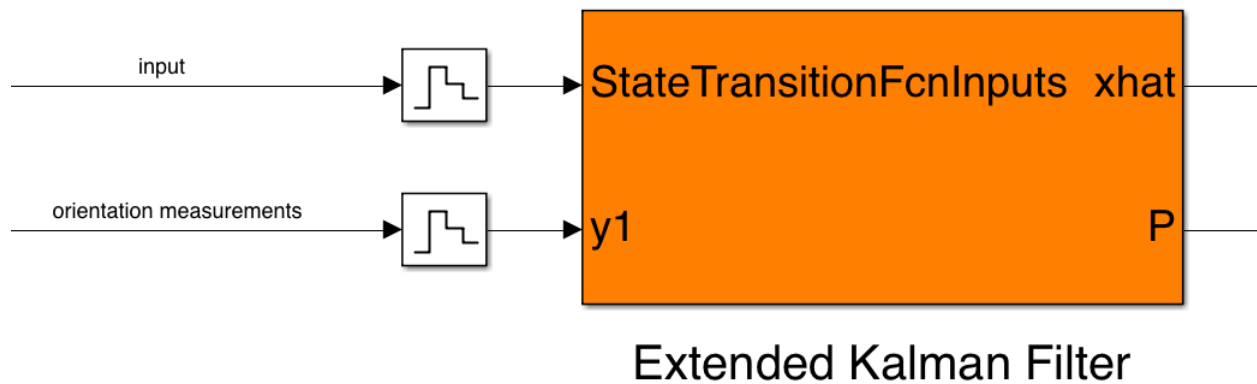


Figure 7.17: The inputs to the Extended Kalman Filter.

7.5.2 State Transition Function

With these inputs, the extended Kalman filter utilizes the implemented state transition function to estimate the next states. The function first finds the next orientation states by using Euler's method as done earlier and explained in Section 2.5. In short, it uses the previously estimated state, as well as the angle derivative at the current measurement. After having found the estimated orientation, Euler's method is again used together with the rotation matrices defined in Section 6.2.2, and the ROP vector to find the next estimated x -, y - and z -position of the drill bit. The EKF-block in Simulink uses the same algorithm as explained in Section 2.4. The code for the transition functions can be found in the Appendix in Section A.2.

7.5.3 Example of Kalman Estimated State Feedback Versus Actual States

By using the estimated states from the Kalman filter in the feedback loop, it is reasonable to expect a deviation in the estimated path to the actual path. An example simulation is shown in Figure 7.18, which shows that the results from calculating the x -, y - and z -positions of the drill bit by using noisy orientation measurements create large deviations from actual position states.

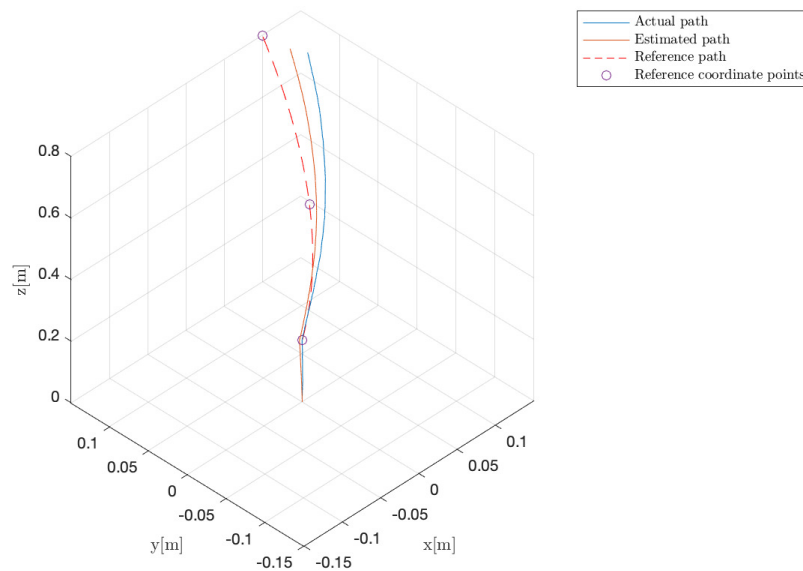


Figure 7.18: An example simulation of Kalman estimation versus actual states.

7.6 State Machine and ROP Logic

The state machine is implemented according to the design presented in Section 6.3, and has two main operations; vertical drilling and directional drilling, which can be seen in Figure 7.19. During these phases, ROP may fall under a certain threshold that renders drilling unsafe, which brings the state machine to a recovery sequence as described.

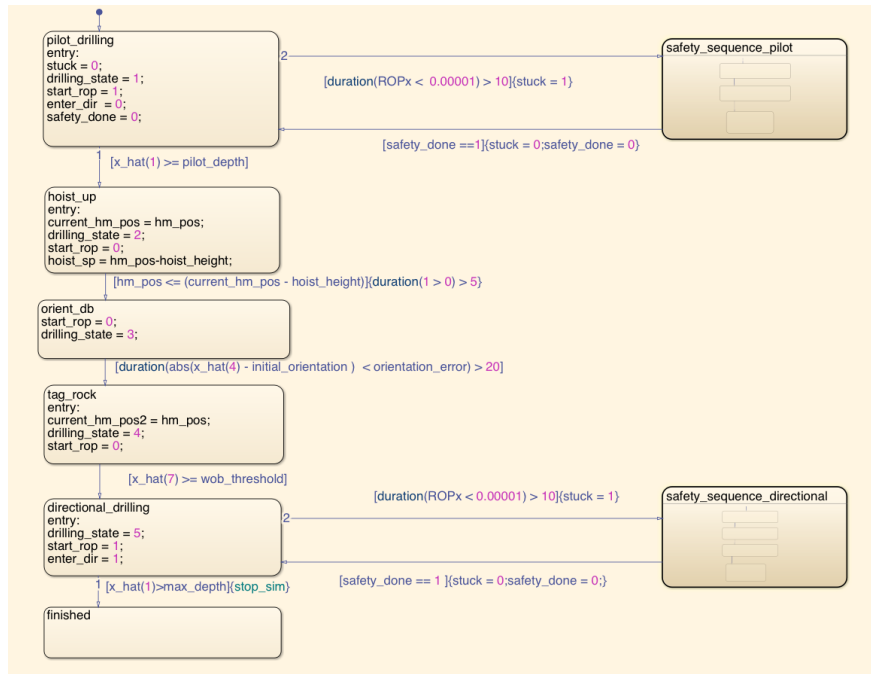


Figure 7.19: The implementation of the state machine has two main phases with safety sequences.

The most essential part of the state machine is the `drilling_state` variable, which decides the controller that should be used as described in Section 7.3. Also, the `stuck` variable propagates throughout the system if the recovery sequence is initiated, which in turn holds ROP at 0 through the ROP block until the recovery sequence is finished. It is also made sure that drilling only happens when the drill bit is touching the stone block, which is given by the `start_rop` variable.

7.6.1 States and Transitions

Pilot Drilling

The state machine initializes the simulation in the state of pilot drilling, where WOB is assumed to be sufficient. This means that the ROP block outputs the defined ROP value into the rest of the system. Whenever ROP is found to be under a certain threshold, the state will transition to the pilot safety sequence, which can be seen in Figure 7.20.

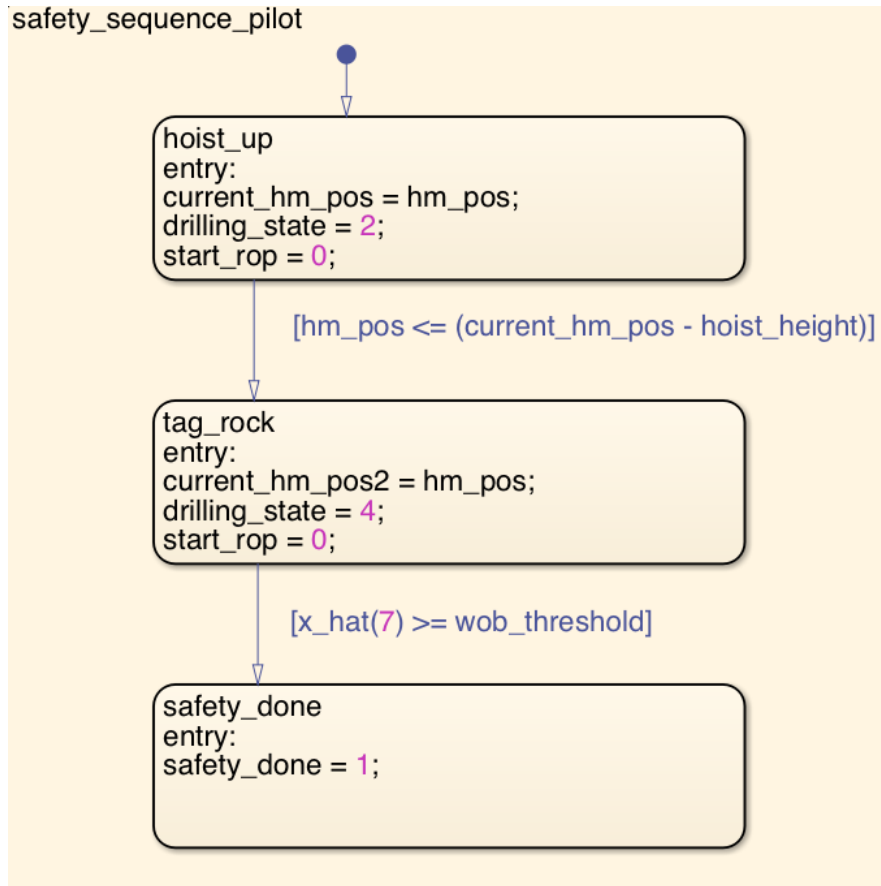


Figure 7.20: The safety sequence of pilot hole drilling is triggered with too low ROP measurement, which hoistens the system up and resets.

The state transitions to hoist up when the drill bit is located beneath the pilot hole length, as seen by $[x_hat(1) \geq pilot_depth]$.

Hoist Up

When in hoist up mode, the hoist up PI controller is used, which holds TD input at 0, and HM RPM input negative to hoist the system upwards. The state transitions when the hoisting motor has hoisted the rotary system up a given length `hoist_height`.

Orient Drill Bit

When the system is finished with hoisting the rotary system up, the initial orientation PID controller is utilized with a TD positional reference and 0 HM RPM input. The state transitions when the orientation of the drill bit is found to be close enough to the reference as seen by $[abs(x_hat(4) - reference(4)) < orientation_error]$.

Tag Rock

When the drill bit is at the correct orientation, the system uses the tag rock PID controller to control the

WOB to a given set point. The Tag Rock controller is implemented the same way as the WOB controller for directional and pilot drilling. It sets the TD input at 0, while the controller is hoisting down the BHA until the WOB hits the `wob_threshold`.

Directional Drilling

This state is reached once pilot drilling and the initialization of directional drilling is finished. The drill bit can be assumed to have the correct orientation, as well as it is touching the rock. ROP is started by `start_rop = 1`, and the directional MPC, NMPC or PID controller is used. Whenever the measured ROP falls under a certain threshold, the system transitions to the directional drilling safety sequence, which is shown in Figure 7.21.

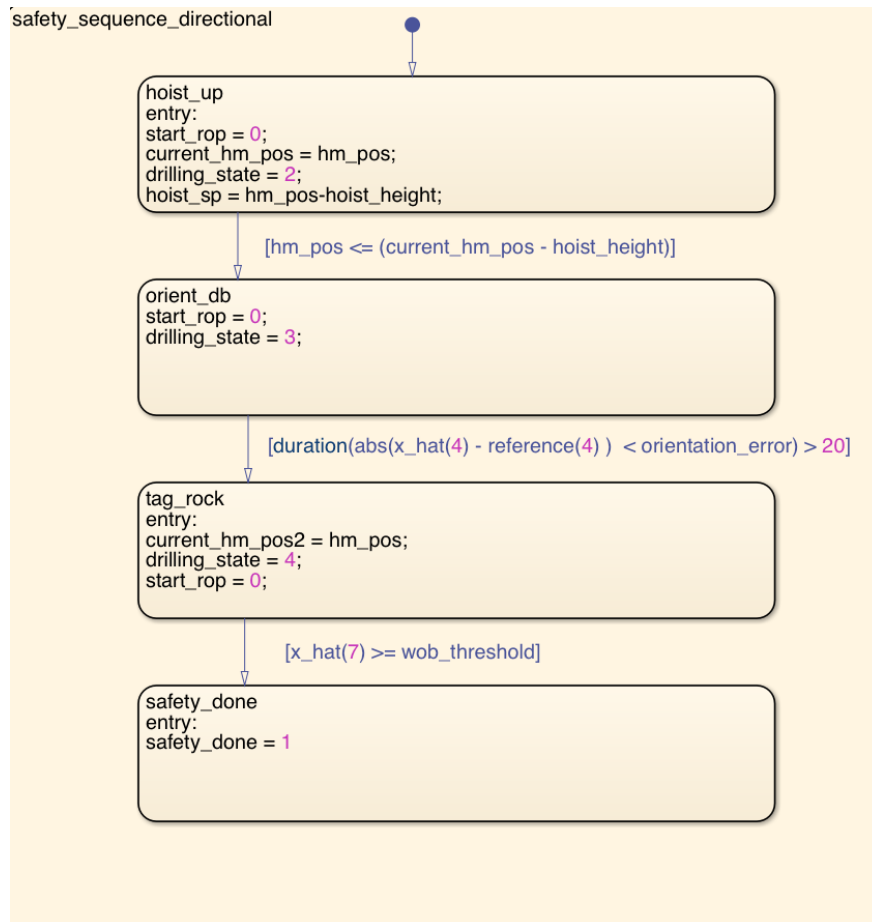


Figure 7.21: The safety sequence of directional drilling is triggered with too low ROP measurement, which hoistens the system up, orients the drill bit and tags rock.

The safety sequence for directional drilling is quite similar to that of vertical drilling, excepts it makes sure the drill bit has the correct orientation before tagging the rock again.

Directional drilling terminates when the x -position of the drill bit is found to be greater than the depth of the last reference coordinate point as seen by `[x_hat(1)>max_depth]{stop_sim}`, which transitions the

state machine to finished.

7.6.2 ROP Logic

As described in Section 6.2.4, the state machine controls the ROP state based on the assumptions that are made when at certain states, such as sufficient WOB. The implementation of this can be seen in Figure 7.22.

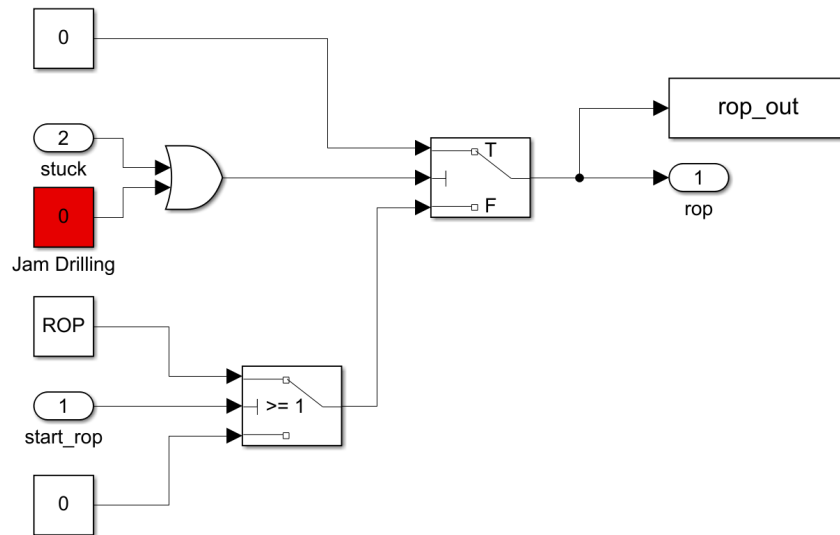


Figure 7.22: The ROP output is based on the `stuck` and `start_rop` variables from the state machine.

As seen in the figure, the ROP output usually uses the `start_rop` state machine variable to decide if the ROP should be a defined positive constant, or 0. However, if the Jam Drilling button in Figure 7.1 is pushed, the red constant turns to 1 for a brief moment, making the ROP 0, which propagates back to the state machine. When the state machine notices that the ROP is falling under a certain threshold, it goes into the safety sequence and changes the `stuck` variable to 1. This keeps the ROP at 0 until the safety sequence is done and `stuck = 0`.

7.7 Data Plotting and Drilling Visualization

To be able to interpret the results correctly from the simulation, it is important to have good visualization schemes of the values that are generated through simulation. Since the system is quite complex with movement in multiple coordinate systems, it is not always easy to visualize what is happening without proper plotting and figures. In this section, these methods will be presented.

7.7.1 General Path Plotting

It is not always needed to visualize the orientation of the drill bit and BHA, but only the resulting path of the simulation. In the latter case, after the reference path generation is done by cubic spline interpolation, circular interpolation or by NMPC, the simulation runs with an independent Kalman filter. An example of a

simulated path can be seen in Figure 7.23, where noise and bias have been removed to find the actual path of the drill bit.

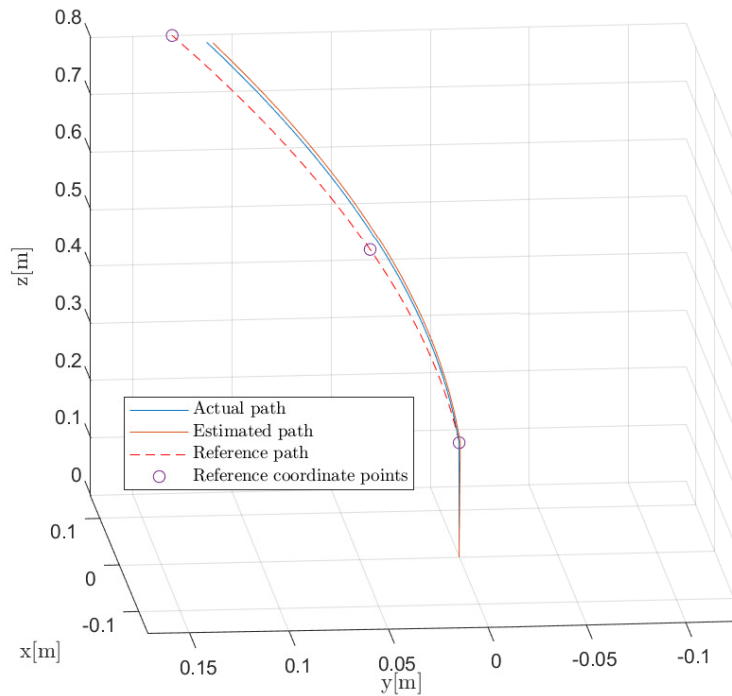
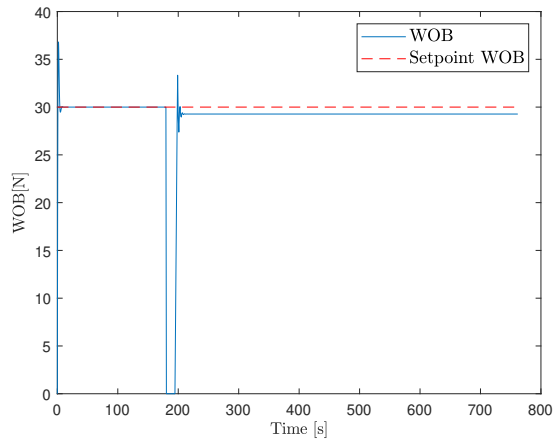


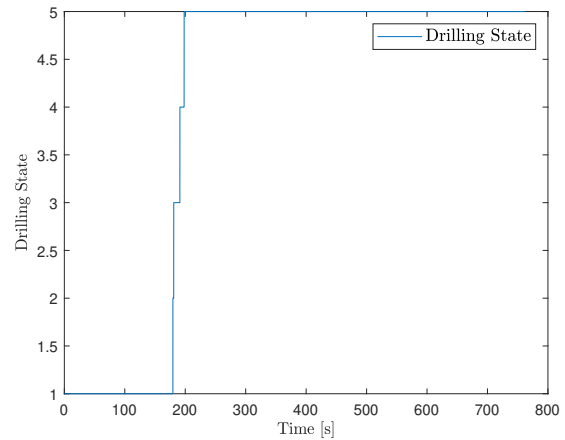
Figure 7.23: Plotting of actual path, estimated path, reference path and reference points.

7.7.2 Plotting of Relevant Data

Another important part of the drilling simulation is to ensure that the operation is safe. Therefore, the variables of interest are all plotted such that even though the drilling path seems correctly drilled, it is made sure that it is done under reasonable circumstances. An example of such a case is that the drilling path is fine, but WOB exceeds its thresholds multiple times. Another example is that the path might seem fine, but the drill might use an unreasonable amount of time in a certain state, such as during initial orientation. Example of these two plots can be seen in Figure 7.24.



(a) Plot of what the WOB dynamics may look like.



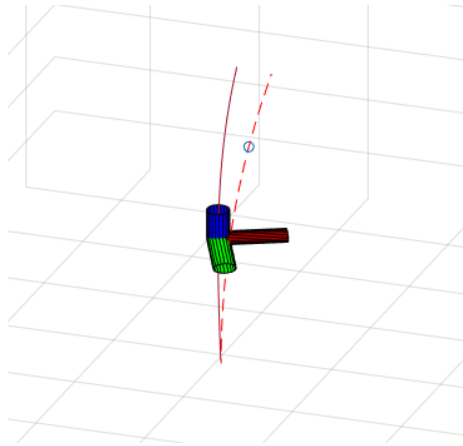
(b) Plot of the time the simulation uses at each state.

Figure 7.24: Example plots of relevant information needed to determine a safe and reasonable drilling environment.

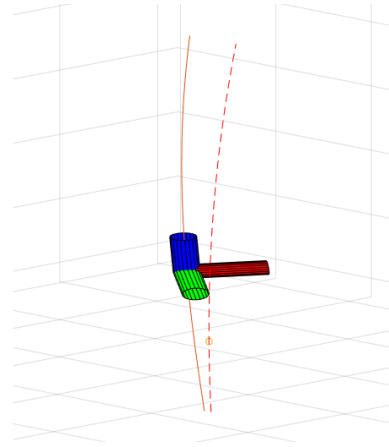
These two examples show two important parameters of the drilling simulation. However, there are many more, and they will all be presented when discussing the results in section 8.

7.7.3 Live Drilling Visualization

When running the drilling simulation, it is not always easy to understand the dynamics of the BHA and drill bit. There has therefore been added two visualization options to be able to easier see what is happening under simulation, which makes debugging easier. The first option is to see the resulting coordinate system of the drill bit at all times. If for example the drill bit is defined to drill in the direction that the z -axis is pointing, it is easy to see if the drill bit is following the reference path. An example of this visualization scheme can be seen in Figure 7.25.



(a) Green z -axis following the reference trajectory.



(b) Green z -axis delayed reaction following the reference trajectory.

Figure 7.25: Drilling visualization with drill bit coordinate system. The blue axis represents the directional vector straight out of the drill bit.

The second option is quite similar to the first option, except it shows the BHA and only the direction of which the drill bit is oriented in. This example is shown in Figure 7.26.

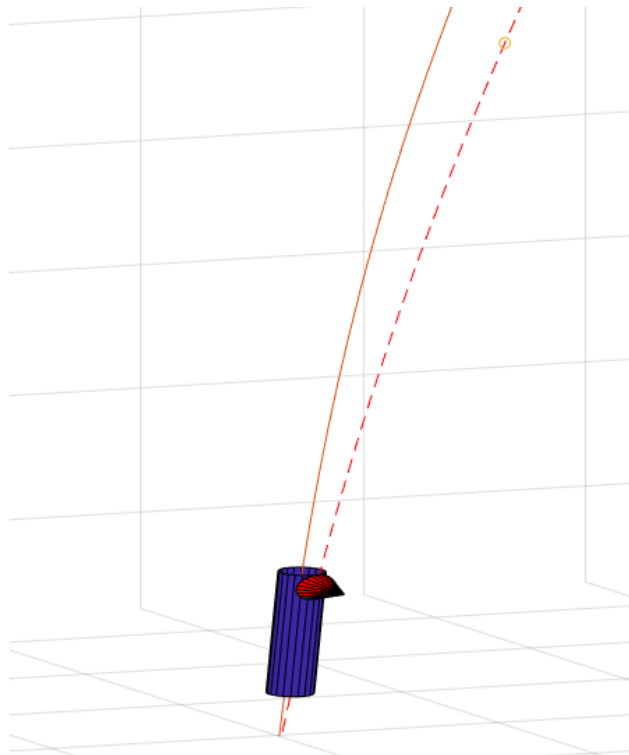


Figure 7.26: Drilling visualization with the BHA and drill bit orientation.

This visualization method gives a more clear picture when looking at the orientation and position of the BHA, as it is possible to see the dynamics of the BHA as well. Both of these visualization options can either

run automatically where the drill bit and BHA follows the reference path over a defined amount of seconds. If there is required thorough analysis around a few points in the path, the simulation can run one time step for every click on the enter-button.

8 Results and Discussion

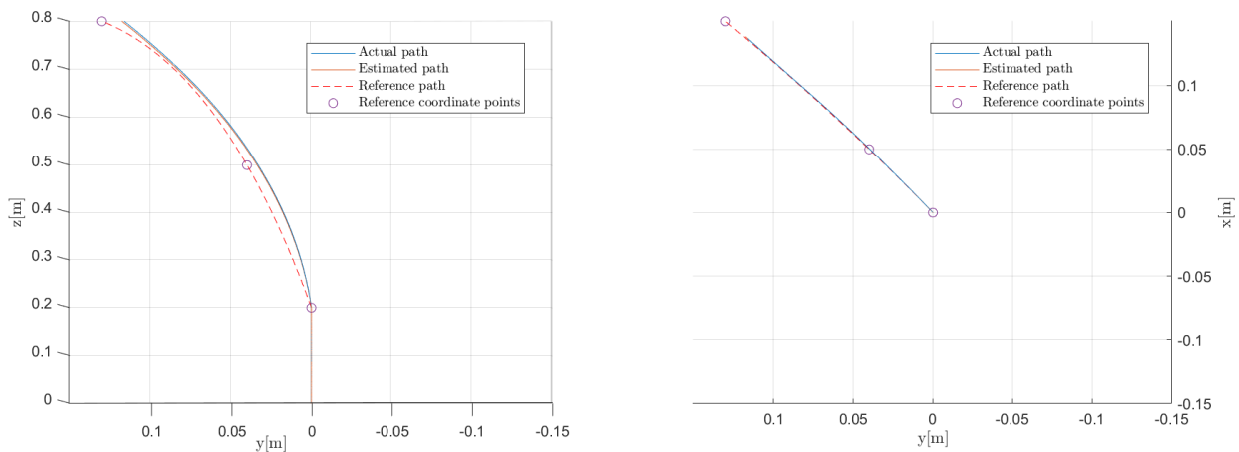
As the simulation system has been designed and implemented, it is important to test it under different circumstances and options. There have been several methods implemented for both trajectory planning and controller options, which all may yield different results based on the drilling environment. In this section, the different results will be presented, as well as discussions around the findings made.

8.1 Path Generation Methods

As there are multiple path generation plans, it is important to examine them for which cases they are most useful, and which minimizes the error from hitting the given coordinate points. In this section, the results from using the different algorithms will be presented, with the assumption of no noise and bias for clarity. All methods will use the reference coordinate points $p_1 = [0.0 \ 0.0 \ 0.2]$, $p_2 = [0.05 \ 0.04 \ 0.5]$ and $p_3 = [0.15 \ 0.13 \ 0.8]$ as a foundation for testing.

8.1.1 Cubic Spline Interpolation

When using cubic spline interpolation, one has to manually change the chosen bent angle of the bent sub to provide Dogleg Severity (DLS). This is one of the drawbacks of this approach, as the trajectory planning does not take into account a constant bent angle. A plot of the system using this approach and the Model Predictive Control (MPC) for directional drilling can be seen in Figure 8.1. The left figure shows the result seen from the y - z -plane, and the right shows the result seen from the x - y -plane.



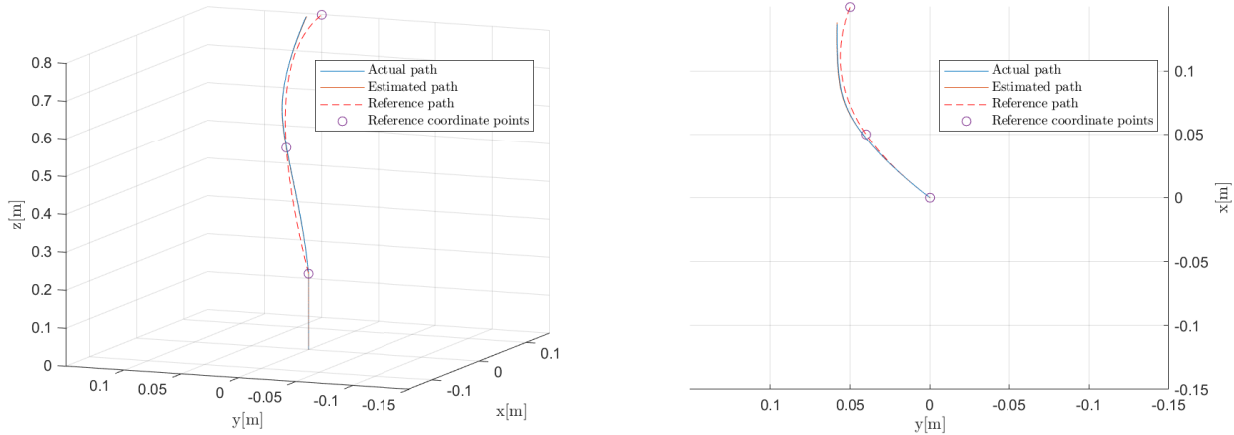
(a) The result using cubic spline interpolation seen from the y - z -plane with points that create no azimuth change. (b) The result using cubic spline interpolation seen from the x - y -plane with points that create no azimuth change.

Figure 8.1: Cubic spline interpolation gives a path with changing DLS, and thus needed bent angle of the drill bit.

As seen from Figure 8.1a, the paths build rate changes from p_1 to p_2 versus p_2 to p_3 , while a constant bent angle of the drill bit gives a more circular path. However, a possibility is to calculate the needed bent angles

at certain times during drilling for manual change of bent sub. As seen in Figure 8.1b, the drill bit manages to stay directly on top of the reference trajectory. It is important to note that there is no noise or bias, which gives close to perfect state knowledge and therefore control input.

An important note of coordinate points used is that there is almost no azimuth change, hence the straight line in Figure 8.1b. By changing the last coordinate point such that one gets an azimuth change to see how the MPC handles this trajectory when it comes to the Top Drive (TD) actuator, one gets a plot seen in Figure 8.2.



(a) The result using cubic spline interpolation with azimuth change. (b) The result using cubic spline interpolation seen from the x -, y -plane with azimuth change.

Figure 8.2: Cubic Spline Interpolation creates steeper curve from p_2 to p_3 compared to p_1 to p_2 .

From the figure it is seen that the path is quite closely held, but with deviations once the azimuth change becomes large.

It is also important to note that at competition day, one may only have a few options when choosing the bent angle of the bent sub, as it may not be possible to change the bend during competition. If this is the case, then simulating with the different possibilities will be a good approach. On the other hand, if it is possible to change it, one can simulate with multiple bends to find the ones that follow the path most accurately.

8.1.2 Circular Interpolation

One clear advantage of using circular interpolation is the constant angular build rate one gets from it as discussed in Section 6.4.2, while a drawback is that the coordinate points must be lined up such that it is possible to estimate a circle going through given points. Using the reference points defined in the introduction of this section, one gets the results shown in Figure 8.3.

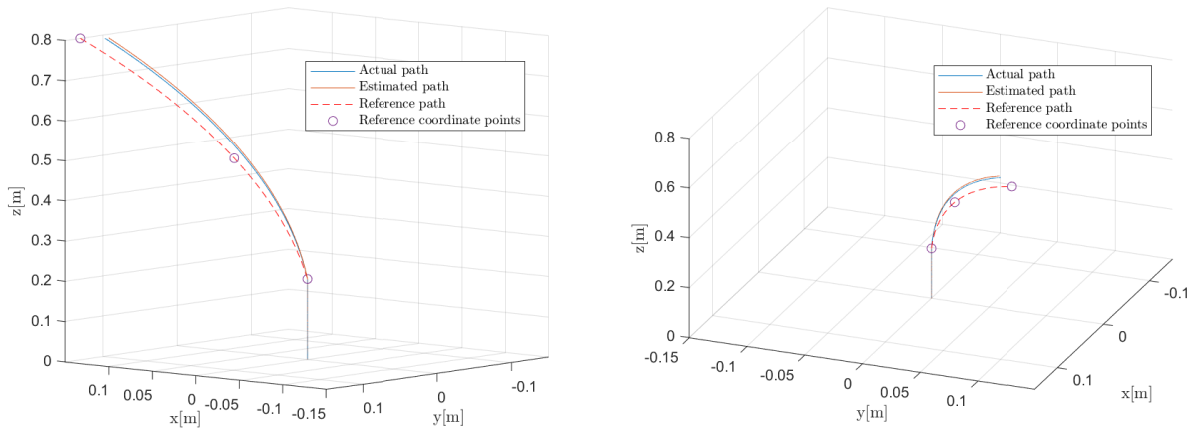


Figure 8.3: Circular interpolation gives a reference trajectory and drilled path with the same shape.

As seen from the figure, the reference path and drilled path share the same constant DLS, which makes it possible to follow the path using a constant bent angle sub. As described in Section 6.4.2, the constant bent angle is calculated by using the parameters of the circular trajectory, but as seen in Figure 8.4, the drilled path is not exactly on point.

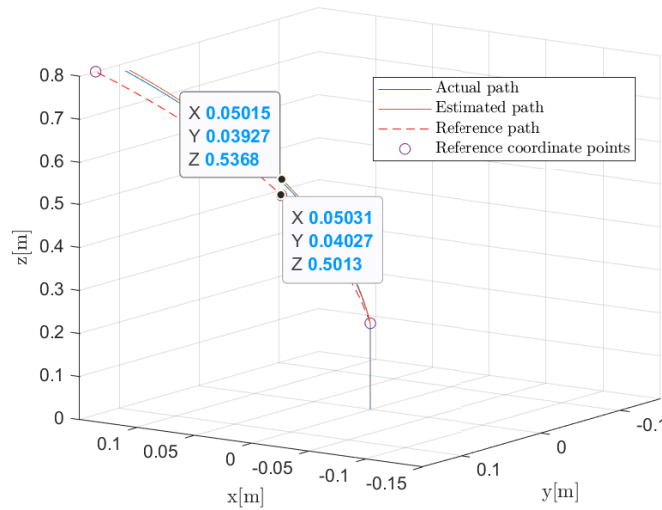


Figure 8.4: An error between p_2 and the drilled path of approximately $z_{diff} = 3.5\text{cm}$.

The reason for this error is due to the entry angle of the Bottom Hole Assembly (BHA) compared to the reference trajectory. However, this problem can be mitigated by scaling the DLS and therefore also the bent sub angle by a constant factor. In the case of simulation with the previous coordinate points, a scale factor of $k_s = 1.2$ gives the plot shown in Figure 8.5.

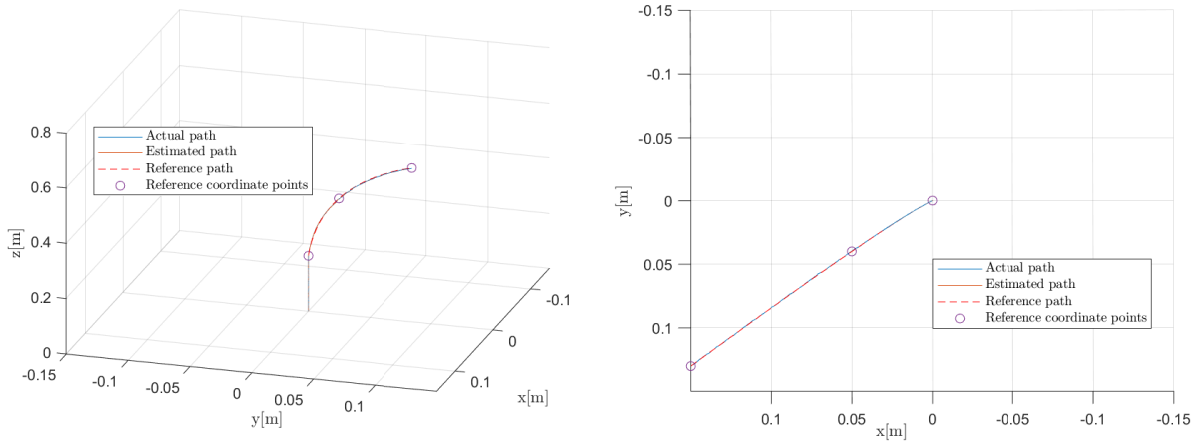


Figure 8.5: Scaling the calculated DLS given from the circle gives perfect following of reference trajectory.

As seen, it is possible to tweak the calculated DLS by gaining it with a constant factor k_s . This in turn proportionally increases the bent sub angle such that perfect following of the reference path is possible. However, this must not be overdone as an excessive DLS will make the drill bit go under the reference path, which in turn can only be corrected by an infeasible turn of 180° of the drill bit as seen in Figure 8.6 by the oscillating BHA orientation from 400s to 800s.

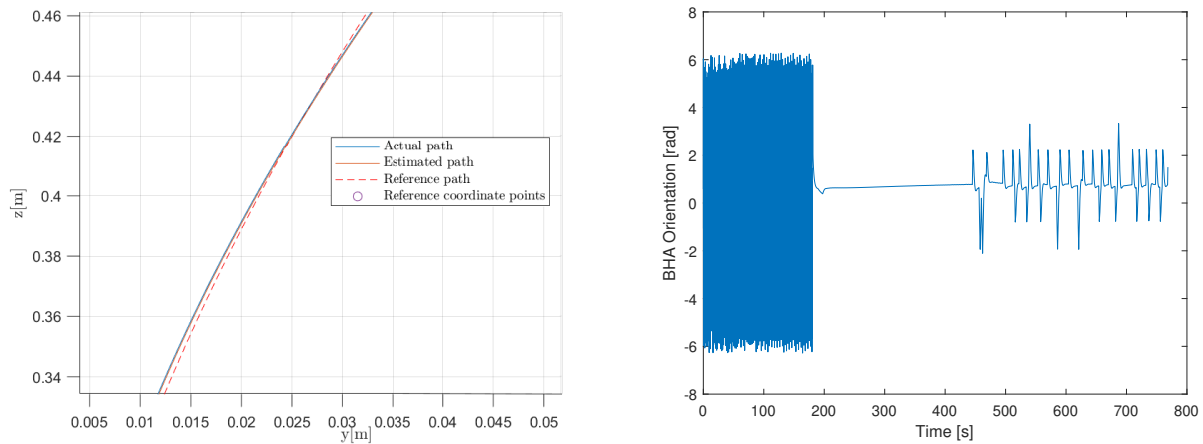


Figure 8.6: Excessive DLS build gives close to 180° turns of the BHA to compensate.

8.1.3 NMPC for Path Generation

As discussed in Section 6.4.3, a possibly better way of creating the reference path, is to generate it based on the constraints of the system such as the used bent sub, and then minimizing the cost calculated by adding the closest distance to the given coordinate points up. An example of a path using this method is shown in Figure 8.7.

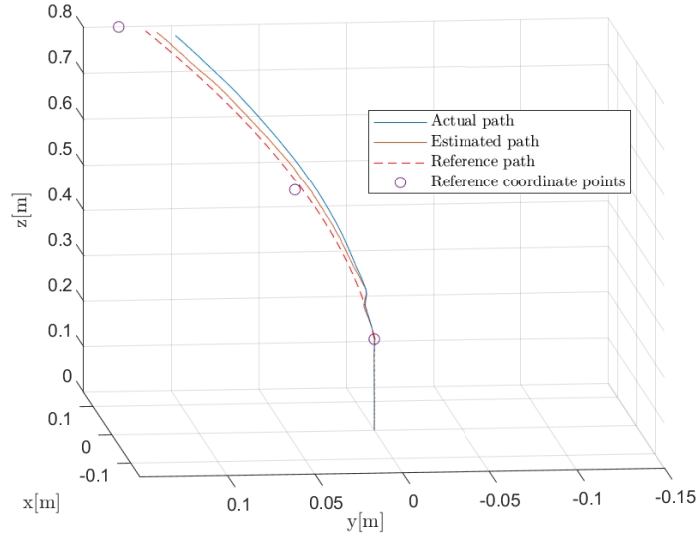


Figure 8.7: The NMPC path does not go through all points, but is generated with the physical constraints in mind.

8.2 Controller Options without Noise and Bias

Since there are four controllers for the simple operations, and one directional controller that can either be an MPC, Nonlinear Model Predictive Controller (NMPC) or PID, there are multiple parameters that need to be tuned by checking the response of the system. The results of all the controlling phases and overall response of the system will be presented in this section.

8.2.1 Controller Results for Simple Operations

The simple operations consist of all system control prior to the directional drilling phase as mentioned in Section 6.5.1. Given that the system changes from one controller to the next given a new state in the state machine, one must make sure that the overall drilling environment stays consistent even in the transitions of controllers. As well as this, it is important to note the amount of time a controller uses to finish its job is important, as one does not want to spend several minutes in for example initial orientation of drill bit.

Using the tuning parameters k_p , k_i and k_d calculated and presented in Section 6.5.1, one can see that the simulation system uses a reasonable amount of time at each state as shown in Figure 8.8. It is important to note that the results shown in this section uses the MPC as the directional controller, so all data shown after approximately 200s is generated by this. Also, the circular interpolation method is used for path generation.

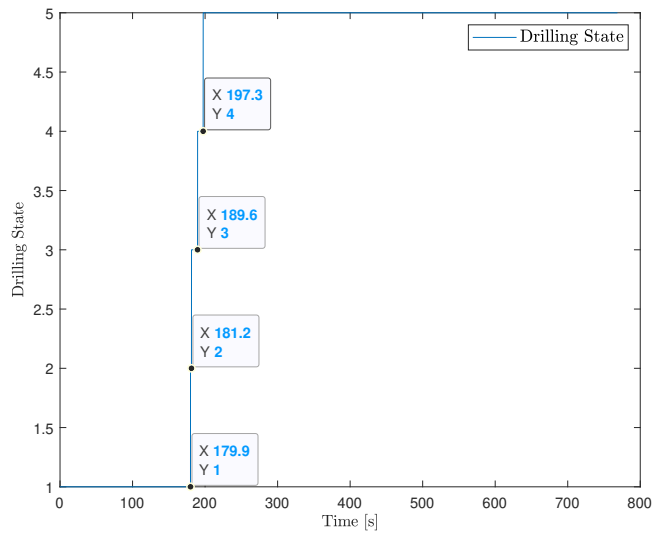


Figure 8.8: The state transition plot shows that the simple operations controllers are tuned reasonably with regards to time.

As the time perspectives of the controllers are good, one must check the values of relevant parameters to check that they do not exceed to infeasible values. In Figure 8.9, one can see the Hoisting motor (HM) position and velocity.

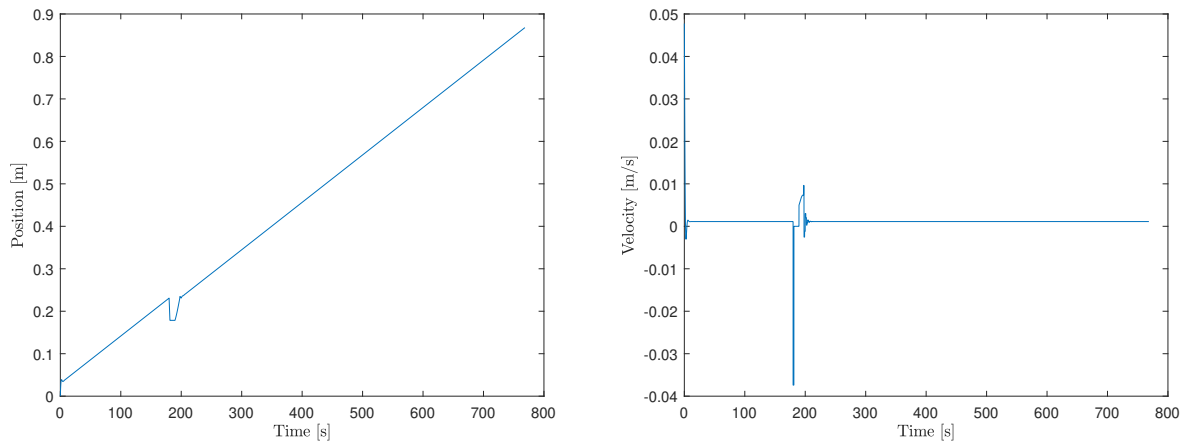


Figure 8.9: The hoist up and tag rock controllers controls the hoisting motor position and velocity to feasible values.

As seen, the HM stays at the same spot while the initial orientation controller takes over to orient the drill bit. After this, it transitions to the tag rock controller, which promptly starts the HM which increases its position.

It is seen from both Figure 8.8 and Figure 8.9 that the system uses about 8.5s with the initial orientation

controller. Looking at the measured orientation in this time horizon, one can see a reasonable response with little overshoot as seen in Figure 8.10.

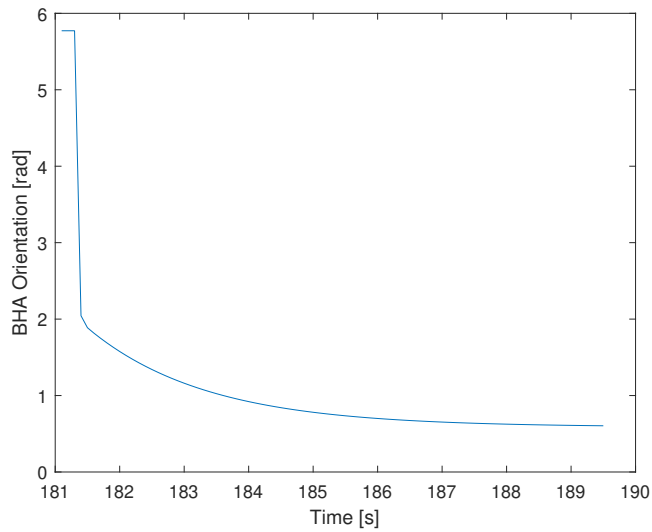


Figure 8.10: The initial orientation controller orients the drill bit to the initial orientation reference at a reasonable pace.

During these operations, probably the most important parameter is the Weight on Bit (WOB) values as this corresponds to the stress of the system as it is the most probable way of drill pipe buckling or snapping. In Figure 8.11, the WOB response during the entire drilling operation is shown. As seen, when comparing this plot to Figure 8.8, the transitions are handled quite well. One can see two overshoots when initial vertical drilling starts, as well as when the drill bit initially touches the rock while in the tag rock state.

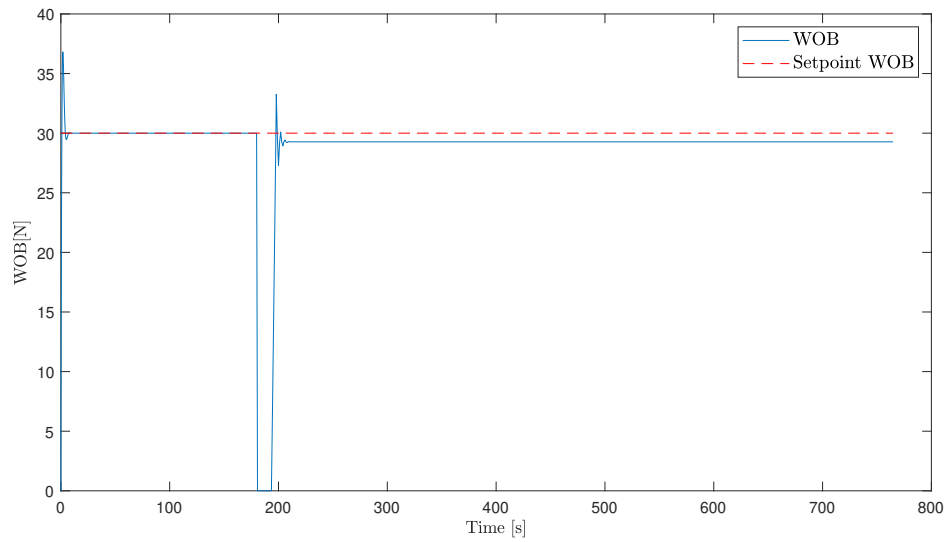


Figure 8.11: The WOB response holds constant except of two oscillatory phases with initially starting vertical drilling and orientation before directional drilling.

As the responses of the state dynamics are reasonable, it is important to see that the inputs generated are reasonable and that they fit the physical constraints of the actuators. In Figure 8.12, one can see this response. As seen, the TD holds a constant value under vertical and directional drilling, but spikes during initial orientation. If this input is not feasible, it is possible to tune the orientation controller to less aggressive values, rendering a slower but more stable system.

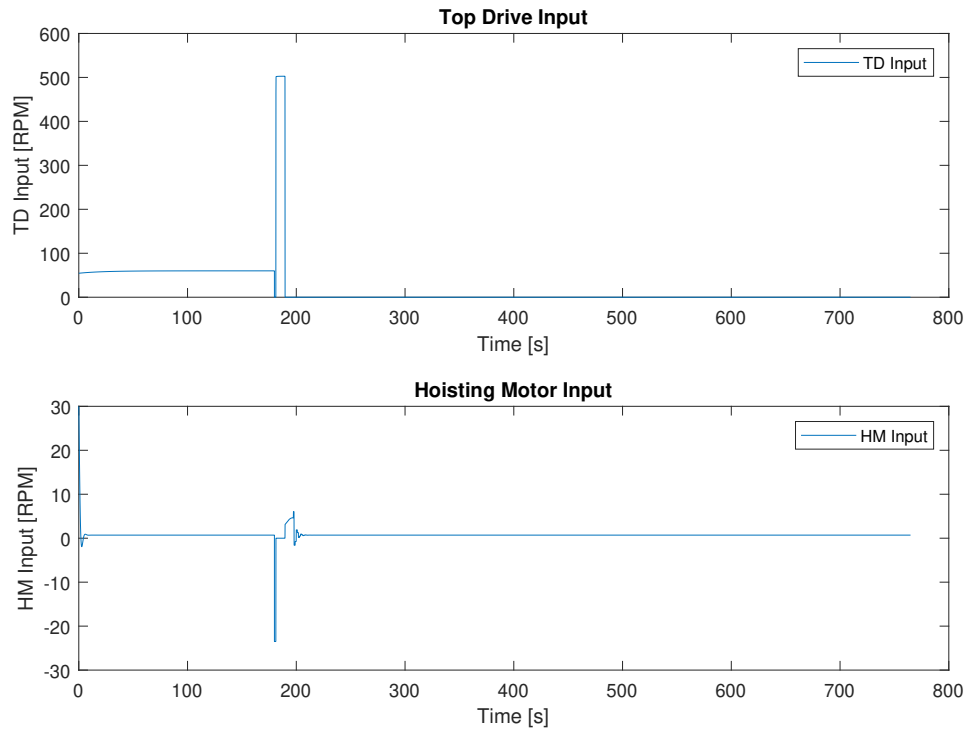


Figure 8.12: The TD and HM inputs are reasonable with regards to physical restrictions.

8.2.2 MPC for Directional Control

As seen from the previous plot, the MPC handles directional control quite well under simple conditions and with no bias and noise. As the MPC mainly has five parameters it considers when calculating the cost, it is interesting to examine what happens when these change. Until now, all presented plots have used MPC state error weights as shown below

$$w_x = 1 \tag{8.1}$$

$$w_y = 0.5 \tag{8.2}$$

$$w_z = 0.5 \tag{8.3}$$

$$w_\phi = 1 \tag{8.4}$$

$$w_\theta = 0 \tag{8.5}$$

$$w_\psi = 0 \tag{8.6}$$

$$w_{wob} = 5 \tag{8.7}$$

Theoretically, the MPC should be able to control without having a weight on w_ϕ , as it can try multiple inputs to better minimize the cost only based on x , y and z . By setting $w_\phi = 0$, one gets the plot seen in Figure 8.13.

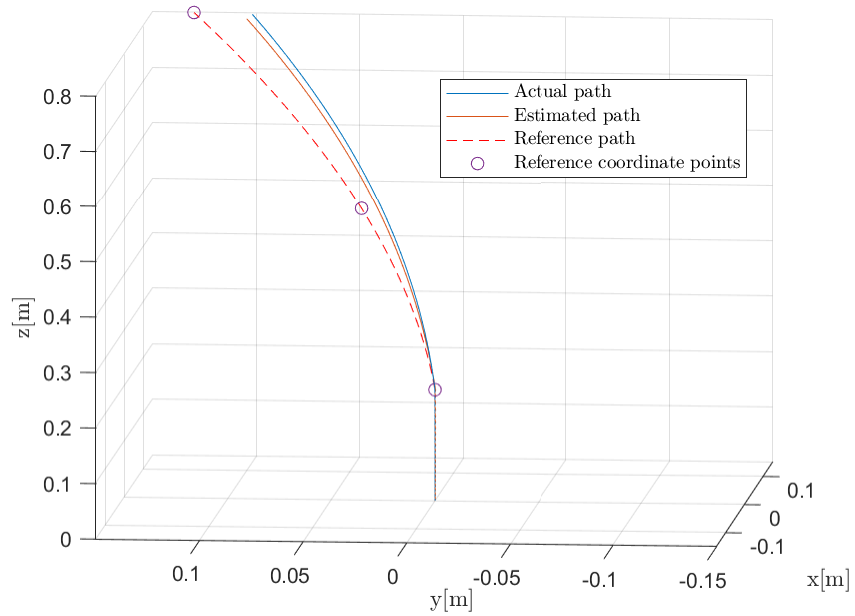


Figure 8.13: By only controlling based on x , y and z error, the path is not followed as well as with the short term ϕ error.

As seen from the figure, the MPC is not able to hold the reference path as well as with a weight on the ϕ error.

As the used reference point creates little to no azimuth change, it is interesting to see how the MPC handles this, as well as the different trajectory planning options. First, the reference points are changed to

$$p_1 = \begin{bmatrix} 0 & 0 & 0.2 \end{bmatrix} \quad (8.8)$$

$$p_2 = \begin{bmatrix} 0.05 & 0.04 & 0.5 \end{bmatrix} \quad (8.9)$$

$$p_3 = \begin{bmatrix} 0.1 & 0.15 & 0.8 \end{bmatrix}, \quad (8.10)$$

with the circular trajectory used. With $w_\phi = 1$ used again, one can see in Figure 8.14 that the start of the trajectory is not very easy to handle as it has quite a bend in the x - y -plane between p_1 and p_2 . However, it is able to hold itself on top of the reference quite well, but given the needed movement to hold itself on the path before p_2 , it is not that easy.

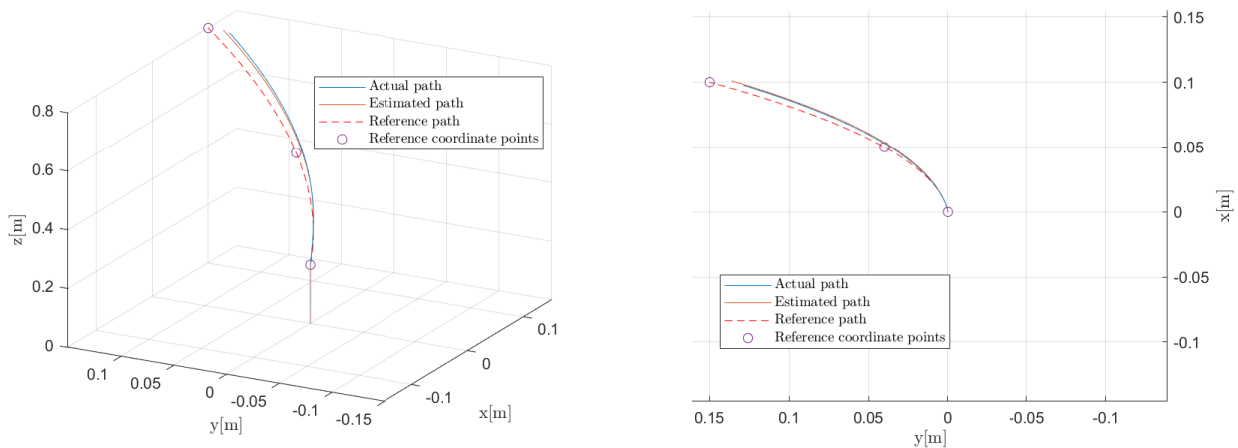


Figure 8.14: The circular path is not easy to follow due to uneven bend in the path in the x - y -plane.

When changing the reference trajectory to cubic spline interpolation, one can see that the drill is able to hold the reference trajectory a lot better, as seen in Figure 8.15. This is because of the bend between p_1 and p_2 is smaller here, but equivalently higher between p_2 and p_3 , which gives a more consistent movement.

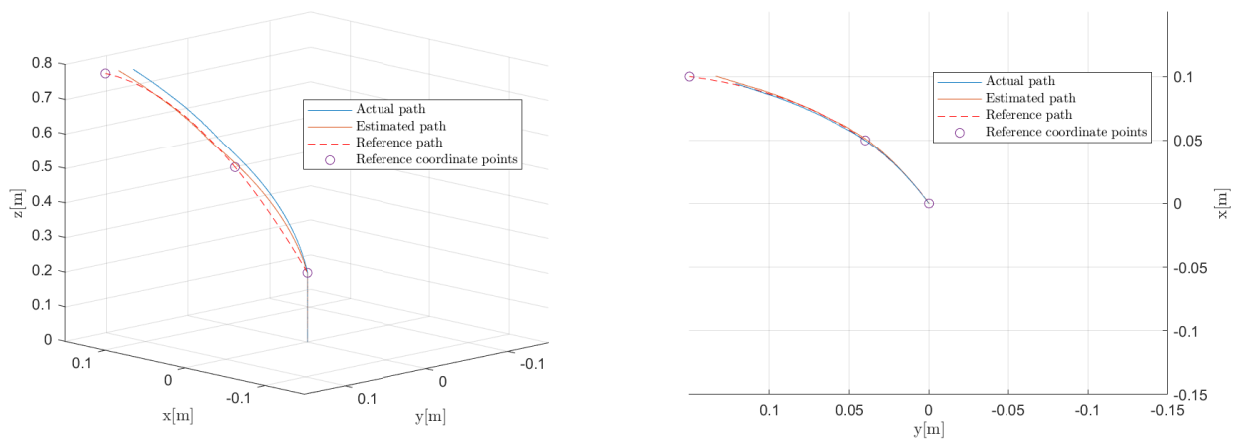


Figure 8.15: The cubic spline interpolation method has a more consistent bend throughout the path which renders an easier path to follow.

8.2.3 NMPC for Directional Control

When using the NMPC, it better considers the nonlinear dynamics of the plant, which in turn may yield better results with the trade-off of longer computation times. To check how well the NMPC controls the path to follow the reference under normal conditions, the following realistic coordinate points are used

$$p_1 = \begin{bmatrix} 0 & 0 & 0.2 \end{bmatrix} \quad (8.11)$$

$$p_2 = \begin{bmatrix} 0.05 & 0.04 & 0.5 \end{bmatrix} \quad (8.12)$$

$$p_3 = \begin{bmatrix} 0.15 & 0.13 & 0.8 \end{bmatrix} \quad (8.13)$$

$$(8.14)$$

together with the circular path generation algorithm. As seen in Figure 8.16, by using the same weights w_x , w_y , w_z , w_ϕ , w_θ and w_ψ , the results look quite similar.

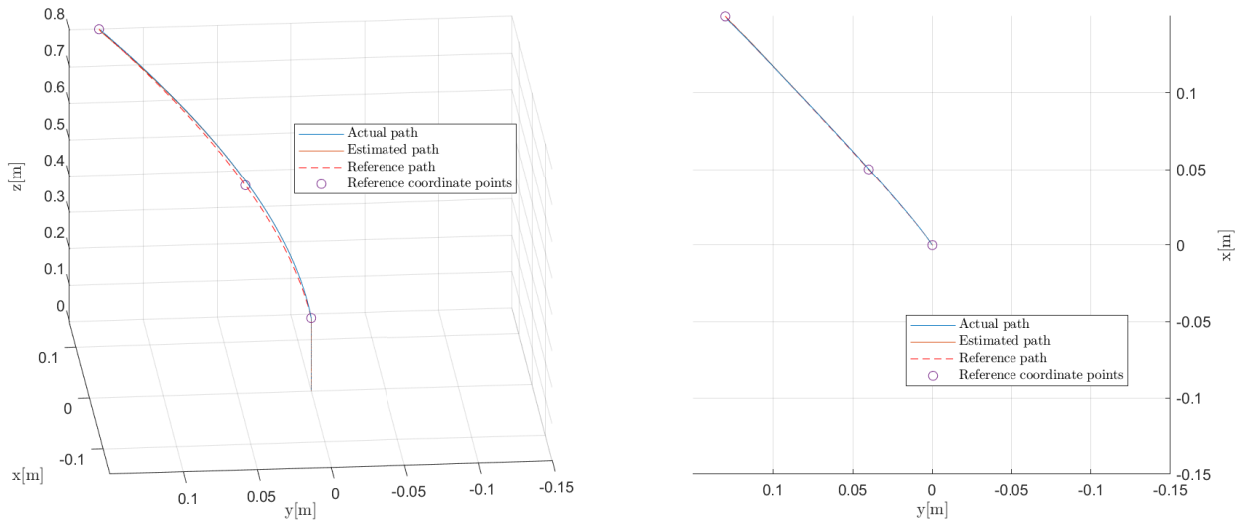


Figure 8.16: The path is held very accurately on top of the reference path.

As seen, the path is held very well with the physical constraints in mind. Here the DLS could be changed such that the actual path follows the reference path perfectly. As also seen from the x - y -plane, the path stays on top of the reference path.

From the results of the MPC simulations, it has been shown that the control heavily relies on an error weight on the angle reference w_ϕ . In some cases, this might not be a feasible approach, as the orientation measurements of the BHA might be unreliable. Therefore, it is interesting to see how the NMPC handles the case of no angle error weight, which can be seen in Figure 8.17. In the case of the current solution, the position is estimated based on the measurements from the BHA. A tuning like this would be a good option if the position could be measured.

While the MPC gets linearized at zero, and as the simulation is running, the prediction gets worse since the states move further away from the linearization point, the NMPC is constantly linearizing the model using the Jacobian function, giving better predictions as seen in the figure.

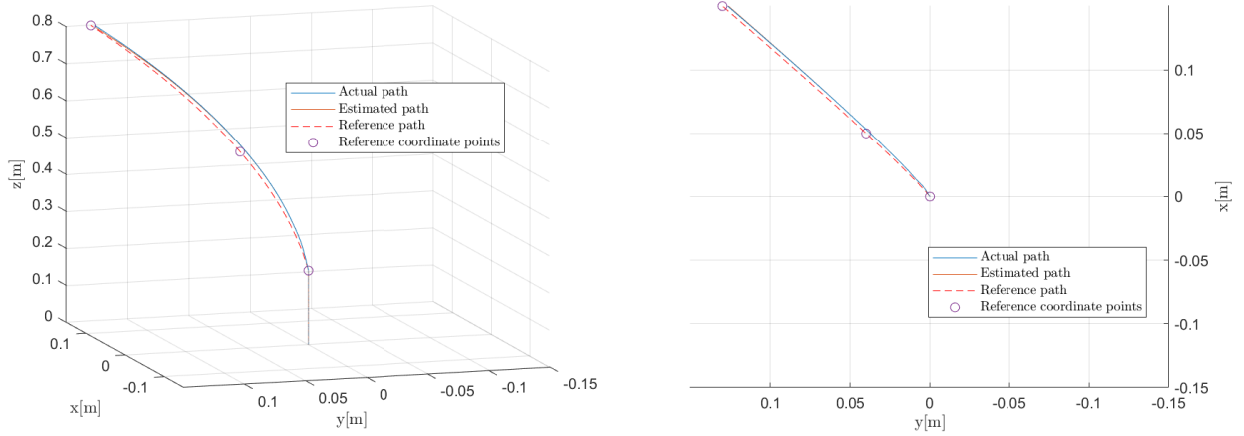


Figure 8.17: The NMPC handles a normal path well without angle error weight w_ϕ .

As seen from the figure, it does not work quite as well without w_ϕ , but it is still a lot better than the MPC without this weight. From the first plot in Figure 8.17, one can see that the actual path is located a little over the reference path, while on the second plot, one can see that the actual path has deviated slightly in the x - y -plane.

As with the MPC, it is interesting to see how the NMPC handles coordinate points that create a needed azimuth change. For this, the same points used previously will be tested again, but with the NMPC as the controller. The reference points are therefore changed to the following

$$p_1 = \begin{bmatrix} 0 & 0 & 0.2 \end{bmatrix} \quad (8.15)$$

$$p_2 = \begin{bmatrix} 0.05 & 0.04 & 0.5 \end{bmatrix} \quad (8.16)$$

$$p_3 = \begin{bmatrix} 0.1 & 0.15 & 0.8 \end{bmatrix}, \quad (8.17)$$

with $w_\phi = 1$ and using the circular trajectory. The result in Figure 8.18, shows that the NMPC is a lot better at holding the reference path than the MPC when the trajectory becomes harder to manage.

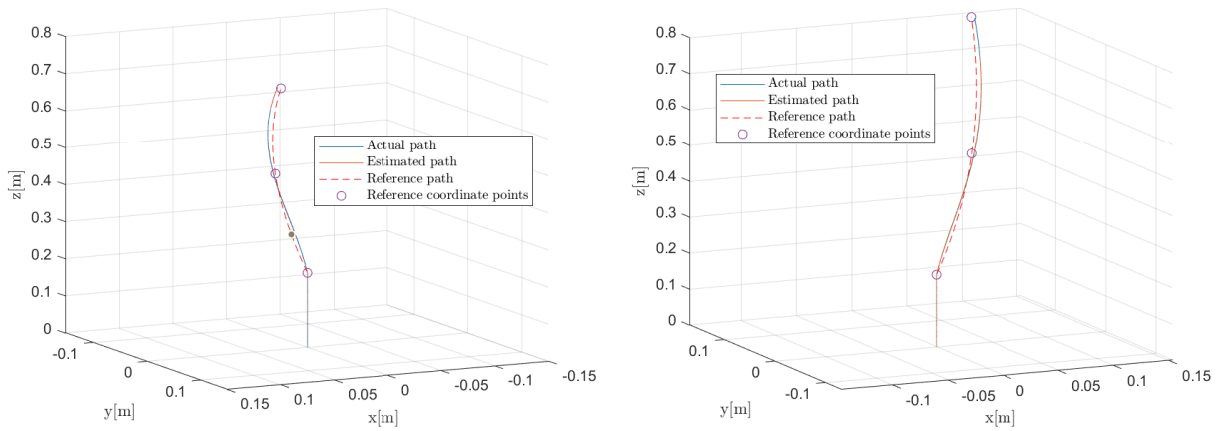


Figure 8.18: The NMPC is able to hold a difficult path well with $w_\phi = 1$.

Now again, one can see the vast difference w_ϕ makes, as Figure 8.17 shows the same simulation with $w_\phi = 0$.

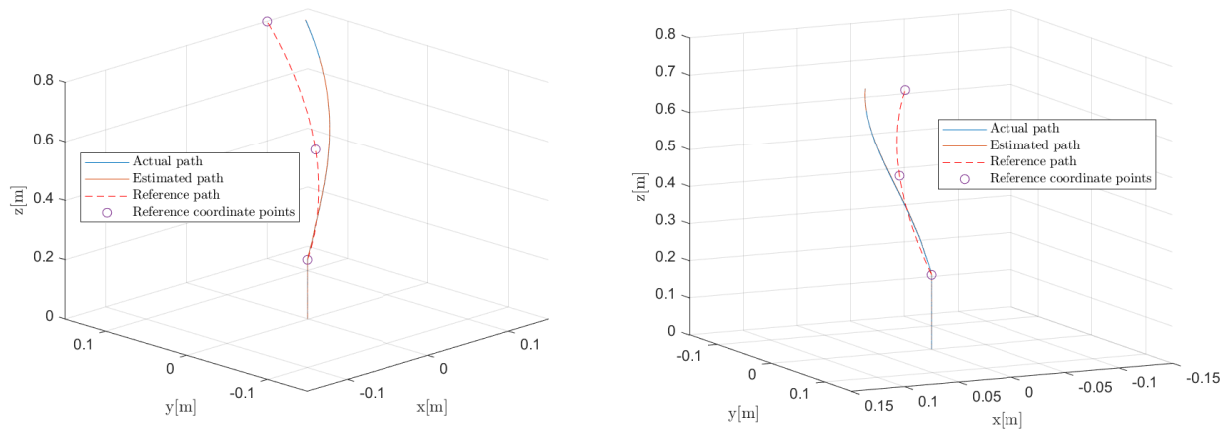


Figure 8.19: When using the difficult path and keeping all other weights the same, but $w_\phi = 0$, the NMPC starts to struggle.

It is important to consider the fact that the rest of the weights are still kept the same, which means that the cost of WOB outweighs the costs of keeping the path on its reference. Therefore, by still keeping $w_\phi = 0$, but increasing the path weights such that $w_x = 1$, $w_y = 1$ and $w_z = 1$, the path holds well, as seen in Figure 8.20.

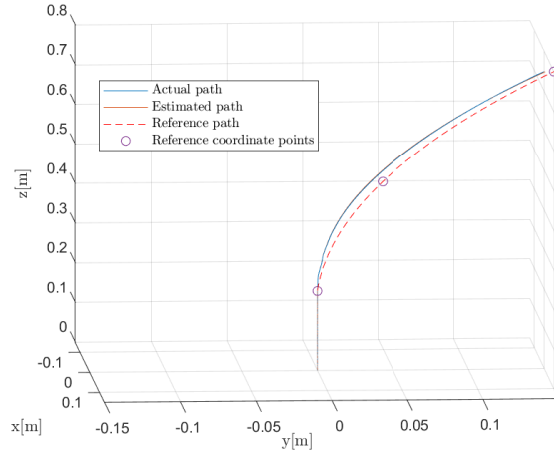
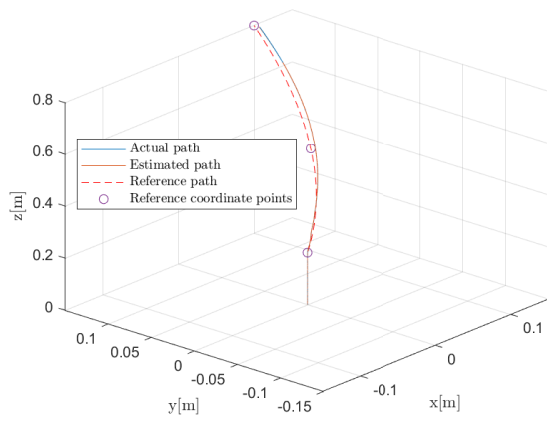


Figure 8.20: By increasing the directional error weights w_x , w_y and w_z , the difficult path is held well without angle weight w_ϕ .

8.2.4 PID for Directional Control

The PID directional controller works in the same way as the initial orientation controller as described in Section 6.5.2. It is very simple, as it only controls based on the angle reference calculated by checking the drill bit position and reference path. Using the simple circular path given by the reference points

$$p_1 = \begin{bmatrix} 0 & 0 & 0.2 \end{bmatrix} \quad (8.18)$$

$$p_2 = \begin{bmatrix} 0.05 & 0.04 & 0.5 \end{bmatrix} \quad (8.19)$$

$$p_3 = \begin{bmatrix} 0.15 & 0.13 & 0.8 \end{bmatrix}, \quad (8.20)$$

one gets the result shown in Figure 8.21. It is important to note that there is used no noise or bias, which makes it easy to control. As these are introduced, the result is expected to be different.

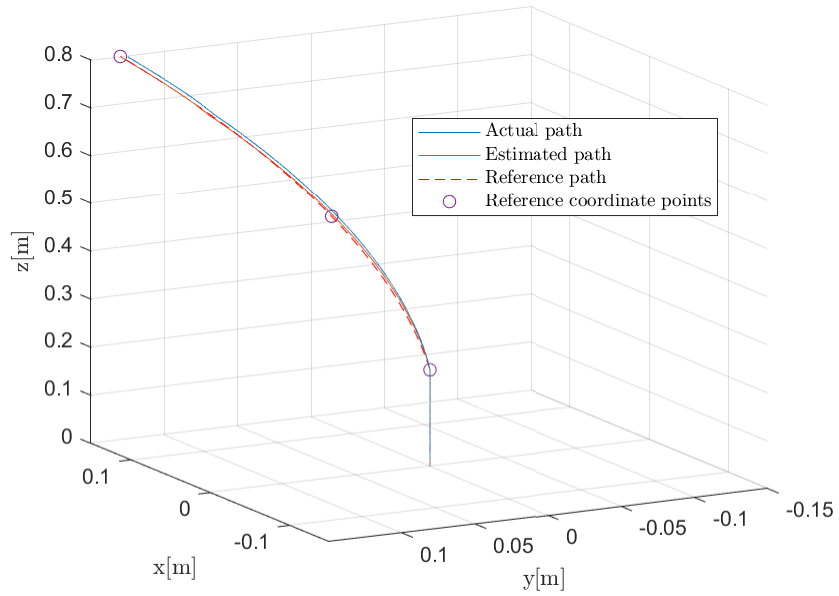


Figure 8.21: The PID is able to control the path nicely by only regarding the angle reference.

When changing the path to the more difficult one given by the points

$$p_1 = \begin{bmatrix} 0 & 0 & 0.2 \end{bmatrix} \quad (8.21)$$

$$p_2 = \begin{bmatrix} 0.05 & 0.04 & 0.5 \end{bmatrix} \quad (8.22)$$

$$p_3 = \begin{bmatrix} 0.1 & 0.15 & 0.8 \end{bmatrix}, \quad (8.23)$$

the result is still good, as shown in Figure 8.22. Important to note here also, is that even though the result seems good, the constraints of the physical rig are not taken into account here, as they are with the MPC and NMPC with their inherent nature of penalizing input rate.

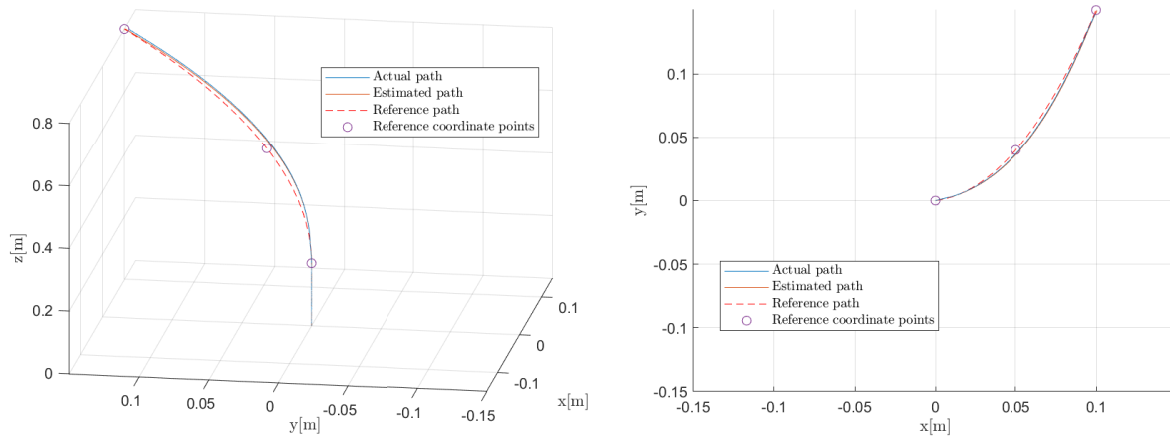


Figure 8.22: The directional PID controller gives good results, but with no regards to the input constraints.

8.3 Introducing Noise and Bias

Until now, all results shown are without noise and bias, such that an assumption of close to perfect state knowledge has been assumed. In the real drilling environment, this is not the case. As presented in Section 7.4, there has been added both process noise and measurement noise and bias. In this section, the effect of these dynamics will be presented.

8.3.1 Sensor Noise and Bias

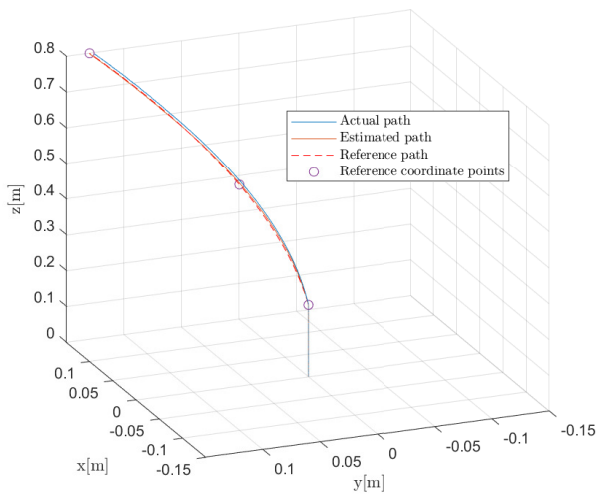
As presented in Section 7.4.3, there has been added noise and bias to the known orientation to simulate an Inertial Measurement Unit (IMU). There is added noise and bias for the magnetometer for the ϕ measurement, as well as noise and bias for the accelerometer to measure θ and ψ . These are given by b_{mag} , v_{mag} , b_{acc} and v_{acc} . Using the MPC and cubic spline interpolation, one can compare the result with and without noise and bias, as shown in Figure 8.23, with the following noise and bias parameters:

$$b_{\text{mag}} = 0.1 \quad (8.24)$$

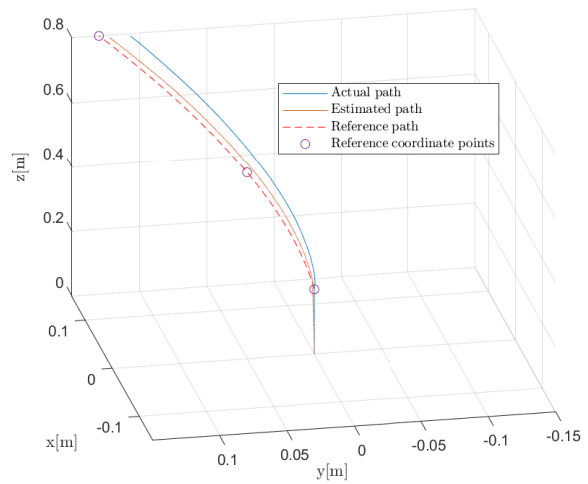
$$v_{\text{mag}} = 0.03 \quad (8.25)$$

$$b_{\text{acc}} = 0.1 \quad (8.26)$$

$$v_{\text{acc}} = 0.03 \quad (8.27)$$



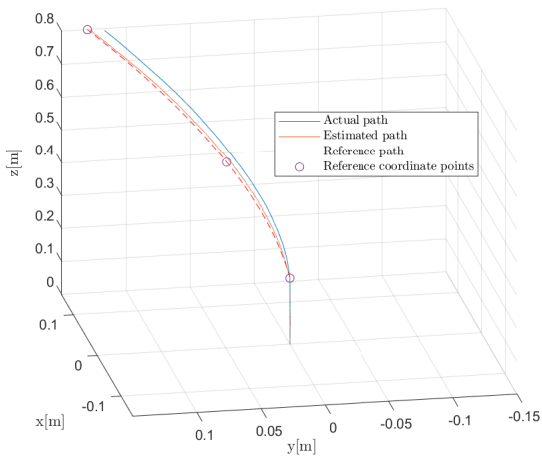
(a) Using no bias and no noise.



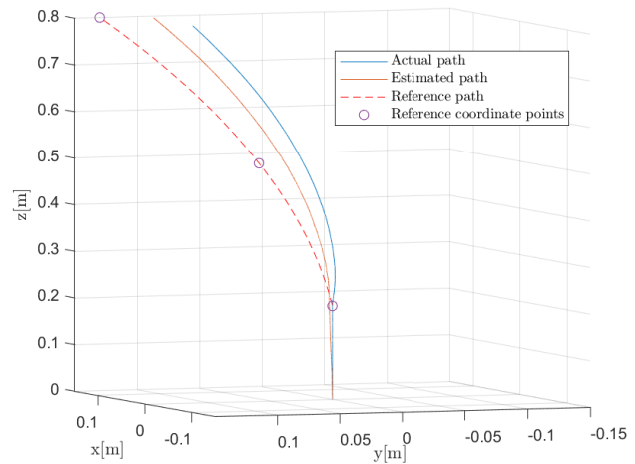
(b) $b_{\text{mag}}=0.1, b_{\text{acc}} = 0.1, v_{\text{mag}} = 0.03, v_{\text{acc}} = 0.03$.

Figure 8.23: Using both noise and bias in the sensor creates a deviation from the path. However, the noise is filtered out as the path is smooth.

As seen, without noise and bias, the drilled path is directly on top of the reference path, while in the case of bias and noise, it is not. However, one can see that the noise is filtered out quite well by the Kalman filter as the path is smooth. By increasing the noise in one case, and bias in the other case, one can see how these two parameters affect the system separately, as seen in Figure 8.24.



(a) Increased noise: $b_{\text{mag}} = 0.1, b_{\text{acc}} = 0.1, v_{\text{mag}} = 0.1, v_{\text{acc}} = 0.1$.



(b) Increased bias: $b_{\text{mag}} = 0.3, b_{\text{acc}} = 0.3, v_{\text{mag}} = 0.03, v_{\text{acc}} = 0.03$.

Figure 8.24: Increased noise makes no effect on the result, while increased bias makes the path deviate far from the reference path.

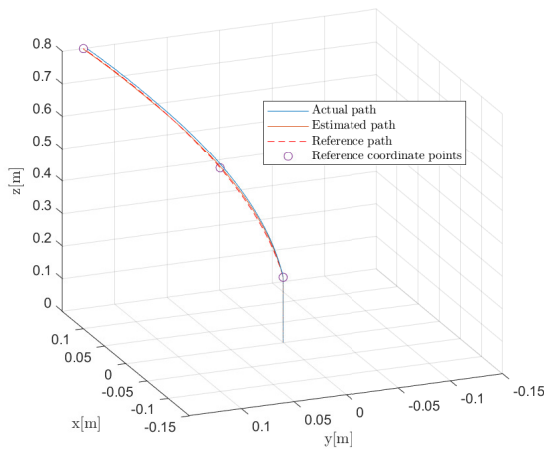
8.3.2 Process Noise

As only measurement noise and bias have been considered, it is interesting to see how the system is affected by process noise. For this, sensor noise and bias are turned off, while process noise is turned on. As described in Section 7.4.1, there is added noise after the angle derivative calculation, as well as after the positional derivative calculation. These noise powers are denoted w_ω and w_{pos} , respectively. By setting these parameters to

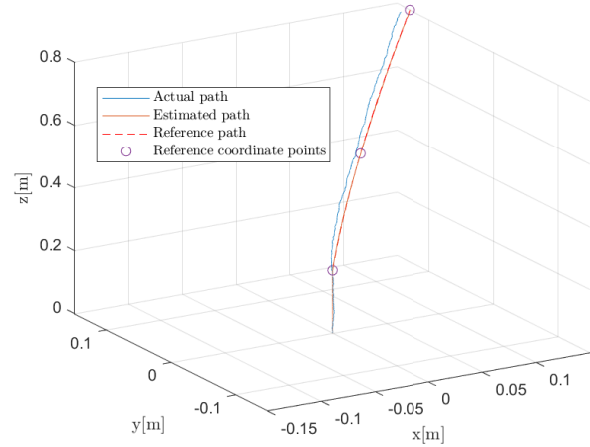
$$w_\omega = 0.00000001 \quad (8.28)$$

$$w_{\text{pos}} = 0.00000001 \quad (8.29)$$

and comparing it with the result from no noise and no bias, one gets the plots shown in Figure 8.25. As seen, it clearly has an impact on the smoothness of the path. However, the path is still moving in the direction of the reference trajectory, and keeps itself quite accurate.



(a) Using no bias and no noise.



(b) Using process noise $w_\omega = 0.00000001$ and $w_{\text{pos}} = 0.00000001$.

Figure 8.25: Process noise has a clear impact on the smoothness of the path.

By increasing the noise by a factor of 10, i.e

$$w_\omega = 0.0000001 \quad (8.30)$$

$$w_{\text{pos}} = 0.0000001 \quad (8.31)$$

one can clearly see the effect on the system, as seen in Figure 8.26. However, this is expected, as the noise is added onto the derivative of each state, which means they get integrated up at every iteration. Another key finding is that the estimated state still follows the reference path perfectly, which shows that it is able to filter out the white process noise.

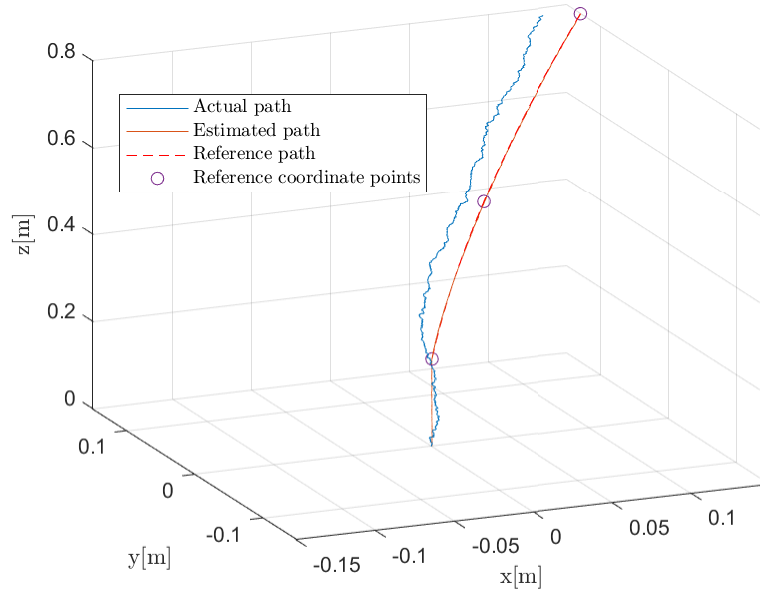


Figure 8.26: Increased process noise shows the dramatic effect on the actual state of the system.

8.3.3 Complete Noise and Bias for Realistic Simulation

Now as the process noise and sensor noise and bias have been shown individually, one can enable them simultaneously to create a simulation system that mirrors reality more accurately. Using the following parameters

$$b_{\text{mag}} = 0.05 \quad (8.32)$$

$$v_{\text{mag}} = 0.01 \quad (8.33)$$

$$b_{\text{acc}} = 0.05 \quad (8.34)$$

$$v_{\text{acc}} = 0.01 \quad (8.35)$$

$$w_{\omega} = 0.00000001 \quad (8.36)$$

$$w_{\text{pos}} = 0.00000001 \quad (8.37)$$

one gets the result shown in Figure 8.27. As seen, there are small deviations from the path, but the overall result is satisfactory. It is hard for the controllers to counteract the process noise, as well as the Kalman filter to correct for the measurement bias.

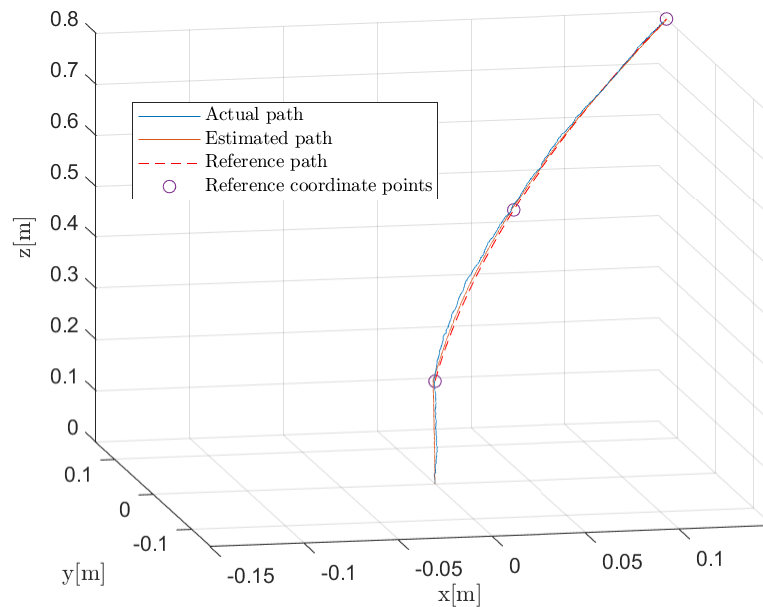


Figure 8.27: The combined process noise and measurement noise and bias gives a realistic system and satisfactory result.

8.4 Realistic Simulations with Noise and Bias

As the result of path generation options and controller options have been presented, as well as the impact of noise and bias, the resulting combination of these will be presented. For this, a simple scoring method is introduced for a better comparison of options. Also, important parameters such as WOB and inputs will be considered.

8.4.1 Scoring Method

Since the physical competition revolves around getting the best score, where this is based on many parameters of which being close to the given coordinate points is the most important, a scoring system that accurately reflects this is used. It is very simple, and checks the distance between the reference points and the closest points on the actual and estimated path. An example of such a score can be given by the resulting plot seen in Figure 8.28, where the reference points have been highlighted.

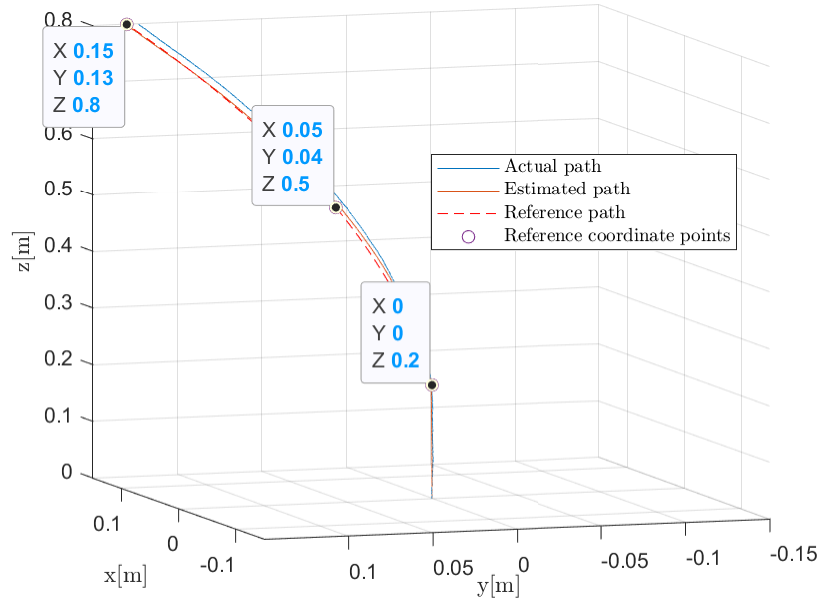


Figure 8.28: An example resulting plot to show how score is measured.

From this example, there is calculated an actual and estimated score of

$$s_a = e_{p1a} + e_{p2a} + e_{p3a} = 0.0122 \quad (8.38)$$

$$s_e = e_{p1e} + e_{p2e} + e_{p3e} = 0.0047, \quad (8.39)$$

where s_a and s_e represents the score between the actual and reference path, and estimated and reference path, respectively. e_{pna} represents the error between reference point p_n and the closest point of the actual path, while e_{pne} does the same with the estimated path. This means that a particular simulation is considered better the lower score it gets.

8.4.2 Simulation with Expected Drilling Environment

To properly test the system in a realistic simulation, all parameters of this run will be based on the competition requirements and constraints. This means that both the azimuth change in the path, as well as the DLS will be based on these calculations. Both process noise and sensor noise and bias will be turned on, such that they mirror the physical environments. Based on this, both the controller options and path generation methods will be compared based on their score as defined in the previous section, and other measurements such as WOB, HM input and TD input.

The chosen noise and bias are based on the previous section, and are therefore set to be

$$b_{\text{mag}} = 0.05 \quad (8.40)$$

$$v_{\text{mag}} = 0.01 \quad (8.41)$$

$$b_{\text{acc}} = 0.05 \quad (8.42)$$

$$v_{\text{acc}} = 0.01 \quad (8.43)$$

$$w_{\omega} = 0.00000001 \quad (8.44)$$

$$w_{\text{pos}} = 0.00000001. \quad (8.45)$$

In the competition guidelines described in Section 1.2, the maximum horizontal and vertical displacements are stated. By doing the calculation as done in Section 3.4.4, one can find the maximum needed DLS and therefore bit tilt angle needed with the configuration of $L_1 = 15\text{cm}$ and $L_2 = 8.5\text{cm}$ as shown in Figure 3.2. The maximum DLS and bit tilt angle have been calculated to be $\text{DLS}_{\text{max}} = 43.4^\circ/m$ and $\phi_{\text{max}} = 5.1^\circ$, respectively.

Also, the maximum azimuth change is stated in the problem description to be 15° , and a maximum of 30° inclination or $10''$ displacement. As such, a set of reference points that simulates realistic competition objectives, as well as being reasonably hard, has been found to be

$$p_1 = \begin{bmatrix} 0 & 0 & 0.2 \end{bmatrix} \quad (8.46)$$

$$p_2 = \begin{bmatrix} 0.07 & 0 & 0.5 \end{bmatrix} \quad (8.47)$$

$$p_3 = \begin{bmatrix} 0.2 & 0.05 & 0.8 \end{bmatrix}, \quad (8.48)$$

which gives an azimuth change of 14° and an inclination of 19° , with a displacement of $8.07''$. Depending on the path generation method, the needed bit tilt may vary, but not exceed 5.1° . Also, these reference points are not well suited for circular interpolation, so only cubic spline interpolation and NMPC for path generation will be used.

Simulation with PID

From the previous run with the PID controller for directional drilling, it was shown that it behaves nicely. Now, when noise and bias are included, one can see that the result is quite different, as seen in Figure 8.29. As seen, the PID controller is not able to counteract the bias and noise that well, since it only uses the current angle and reference angle to calculate the input.

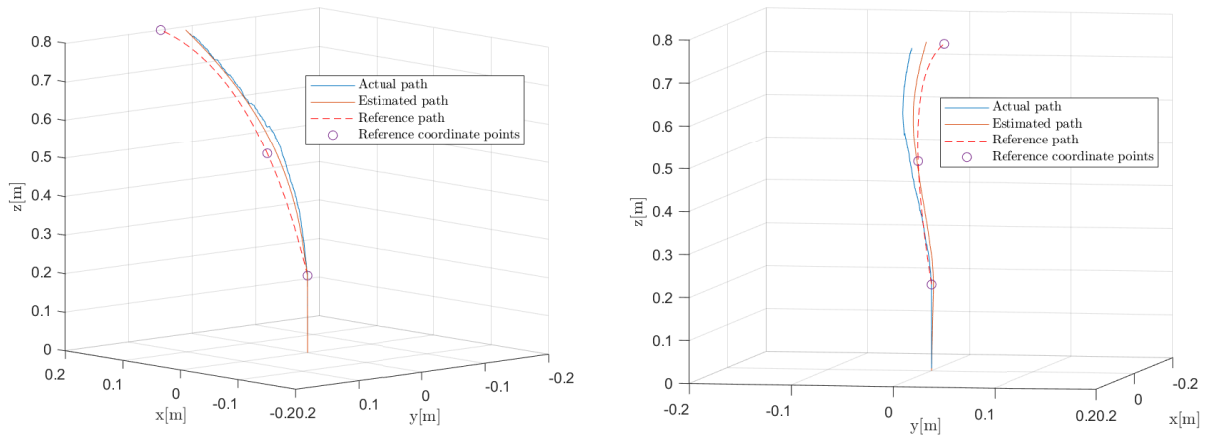


Figure 8.29: The PID controller solely controls direction based on angle. Given the small error in the start due to noise, a wrong angle reference is used, which turns the drill bit away from the path.

One can also see from the plot in Figure 8.30 that the orientation noise affects the reference generator, as it tells the orientation PID controller to do an unnecessary large input, which brings the state further away from the path. Because of this, as well as the fact that the simulation stops when the estimated position of the drill bit exceeds the height of the path, the drill bit is not able to reach the reference path before it drills itself to the height of the reference path, which stops the simulation.

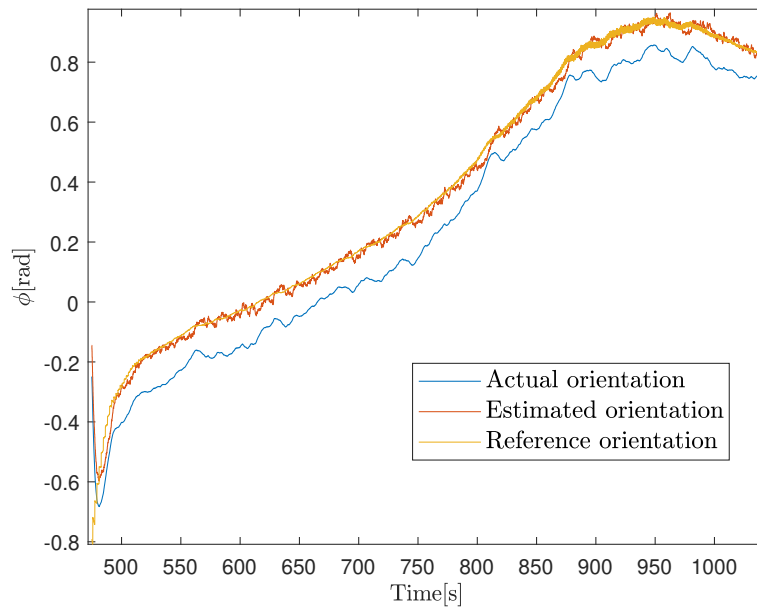


Figure 8.30: The generated angle reference takes an unnecessary initial turn because of the noise in angle measurement.

An important parameter to look at whilst drilling is the WOB values. Keep in mind that when using PID for directional drilling, there is a separate WOB controller used. As seen in Figure 8.31, it keeps itself around

the nominal WOB of 30N even with the substantial noise added.

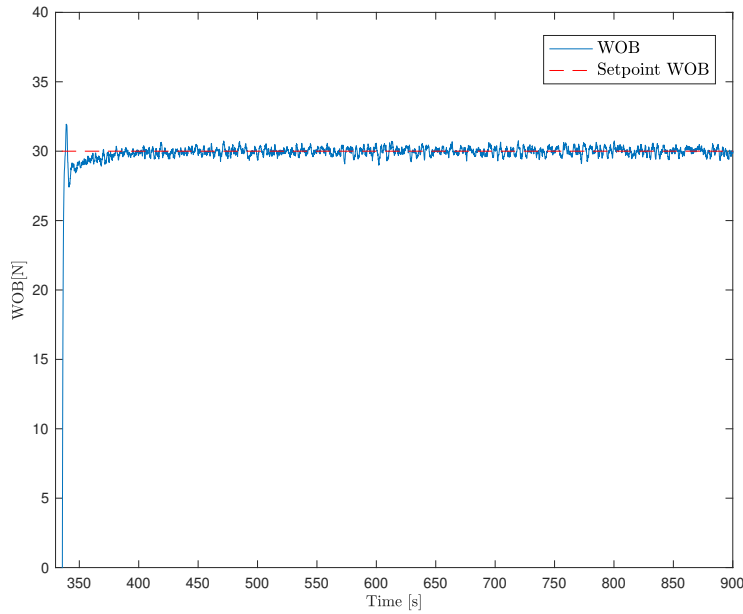


Figure 8.31: The resulting WOB keeps itself around the nominal reference value when using PID for directional control.

To see if the simulation is realistic with regards to the TD and HM actuators of the system, one can look at the generated input, as seen in Figure 8.32. As seen, the TD input may not be feasible as the rate of change could lead to wear and tear of the actuator. The HM input is however feasible, as it can be seen that it starts off high when in the tag rock state, and after that, hovers at around 0.5 RPM when in directional drilling mode.

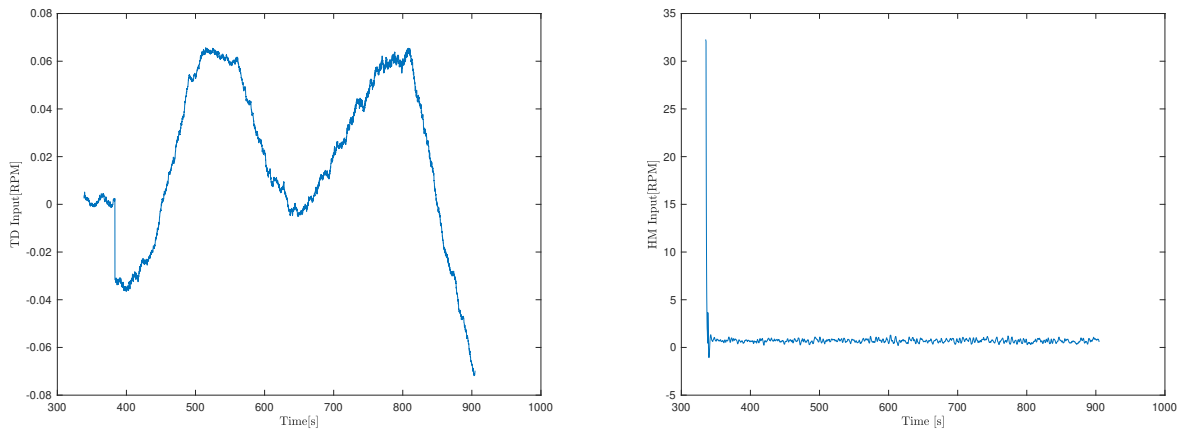


Figure 8.32: The PID controller produces TD input that has small short term spikes, but changes from [-0.06,0.06] during the directional drilling phase.

The shown simulation was done with cubic spline interpolation as path generation method, and the presented plots look quite alike when using the NMPC method instead. The score, however, presents quite a difference, as given by

$$\text{PID-Score}_{\text{cs-real}} = 0.0587 \quad (8.49)$$

$$\text{PID-Score}_{\text{cs-estimated}} = 0.0512 \quad (8.50)$$

$$\text{PID-Score}_{\text{nmpc-real}} = 0.0947 \quad (8.51)$$

$$\text{PID-Score}_{\text{nmpc-estimated}} = 0.107. \quad (8.52)$$

The reason for the bad score when using NMPC path trajectory can be clearly seen in Figure 8.33.

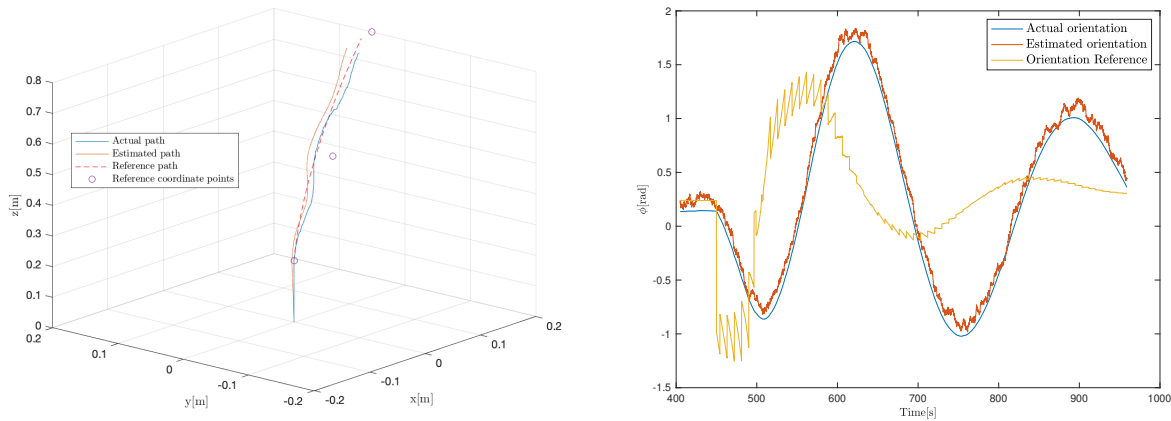


Figure 8.33: Using the NMPC path generation method when using PID as directional controller does not work well in this case, as the drill bit is taken underneath the reference path.

As seen, the problem occurs when the drill bit goes underneath the reference path, as the PID tries to steer it back, and therefore wobbles back and forth around this path. When using an MPC or NMPC as the controller, the result is expected to be different, as there are multiple control parameters used in these cases, and not just the angle reference.

Simulation with MPC

Switching the controller to the MPC, the biggest difference is that it controls with regards to the positional values x , y , and z , as well as the angle ϕ and WOB, while the PID directional controller only controls with regards to ϕ and WOB. As seen from Figure 8.34, the resulting path is not held that well, arguably worse than the PID controller.

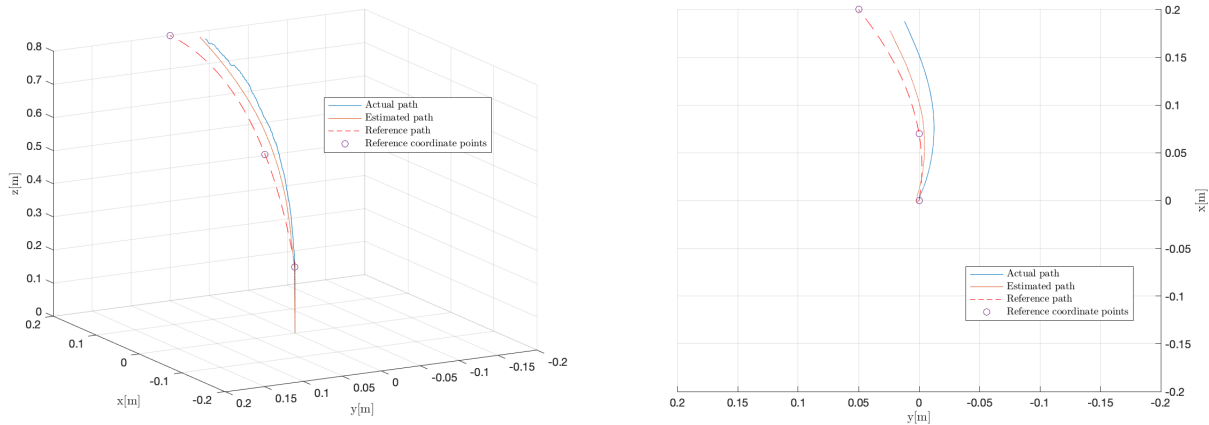


Figure 8.34: The resulting path when using the MPC is not that different from when using the PID directional controller.

Also, the WOB value is not held exactly on the nominal value, but hovers a little underneath it as seen in Figure 8.35. This is because the MPC regards multiple variables, such that if more weight is put on the WOB error, then less attention is put on the directional part of the controller.

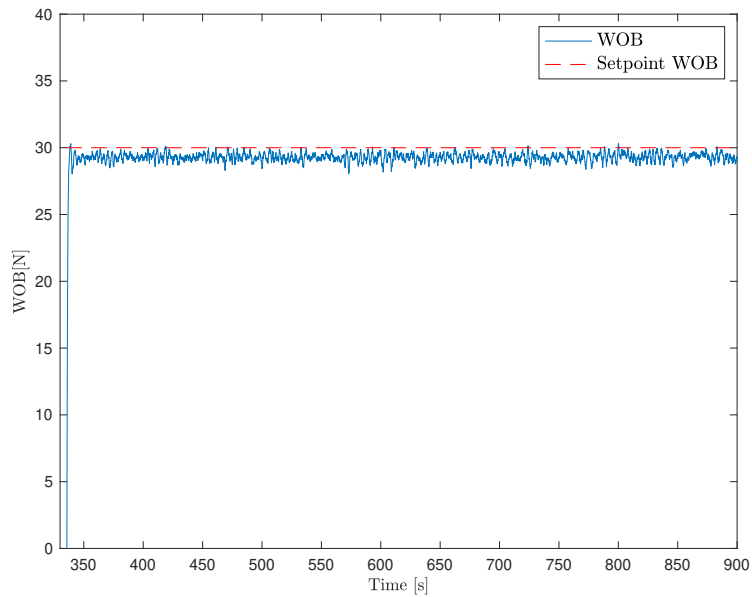


Figure 8.35: When using the MPC for WOB control, the WOB is held close to the nominal value of 30 N.

The difference between using the PID for directional control and the MPC is however seen when plotting the TD and HM input, as seen in Figure 8.36. The MPC is kept around $[0, 0.05]$ for the longest part of this phase. It is important to remember that these inputs are given in RPM, such that large spikes are not that dangerous.

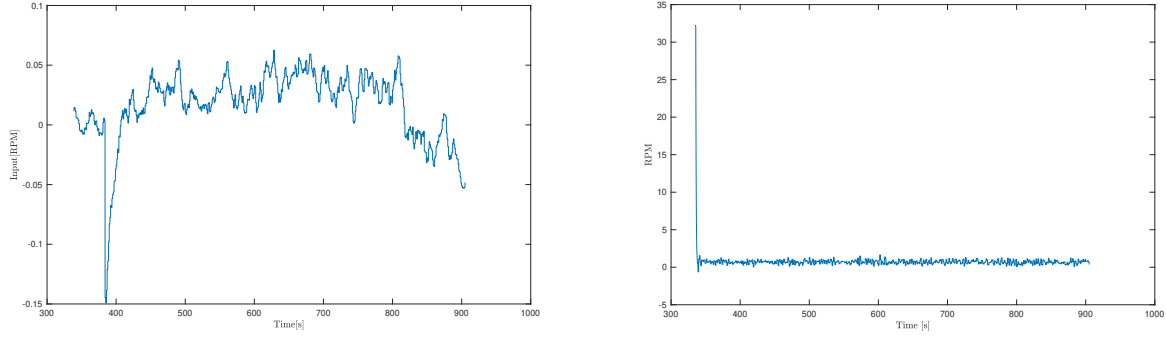


Figure 8.36: The TD input RPM compared to when using PID as directional controller has more spikes, but hovers around $[0, 0.05]$ for most of this phase.

The resulting score using the MPC and cubic spline interpolation and NMPC for trajectory generation can be seen by

$$\text{MPC-Score}_{\text{cs-real}} = 0.0593 \quad (8.53)$$

$$\text{MPC-Score}_{\text{cs-estimated}} = 0.0508 \quad (8.54)$$

$$\text{MPC-Score}_{\text{nmpc-real}} = 0.0471 \quad (8.55)$$

$$\text{MPC-Score}_{\text{nmpc-estimated}} = 0.0519. \quad (8.56)$$

As the score shows, the better result is given when the NMPC is used for path generation. The score may also be improved by setting new weight parameters. This can be done by testing and simulating with new parameters, and checking the results.

Equations (8.57) and (8.58) shows that the distance from the target grows towards the end. This is due to the fact that the path is impossible to follow because of mechanical limitations. This is also a result of the MPC being linearized at the initial conditions, and as the drilling process is running, the drilling states are getting further away from the linearization point. As explained in Section 8.5.1, the estimate of the position will not converge towards the actual position. This will in turn lead to worse control performance. This also explains the reason why the gap between the estimated distance to the target and real distance to the target is growing as the drilling process is running.

$$D_r = \begin{bmatrix} 0.0001 & 0.0166 & 0.0427 \end{bmatrix} \quad (8.57)$$

$$D_e = \begin{bmatrix} 0.0025 & 0.01430.0340 \end{bmatrix} \quad (8.58)$$

Simulation with NMPC

When using the NMPC, which on one hand is a better fit for the system, but on the other hand takes a long time to simulate, one would expect better results. From the plots seen in Figure 8.37, it can be seen that the resulting path is not that different.

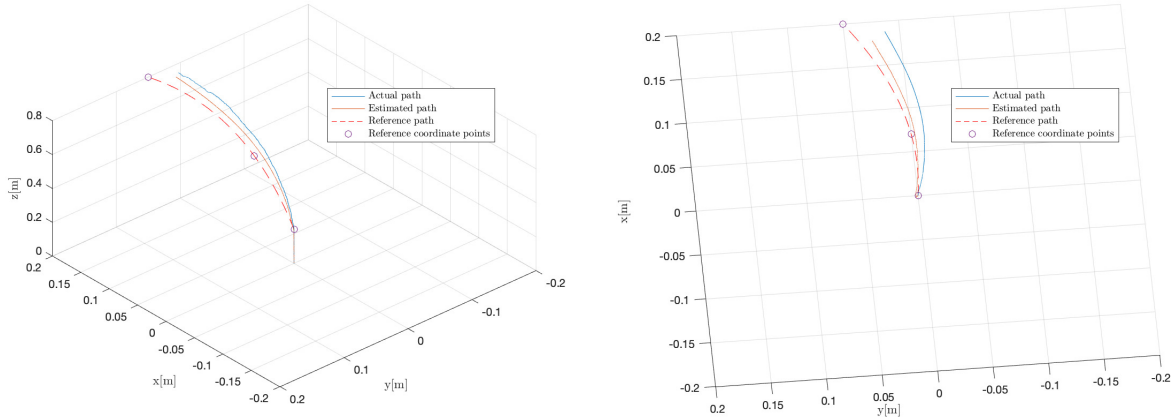


Figure 8.37: The resulting path when using the NMPC is not that different from when using the PID or MPC directional controller.

When looking at the WOB measurement, one can see that the NMPC differs from MPC, but has the same properties as the results when using the PID controller. It is seen that the value hovers around the WOB value of $30N$.

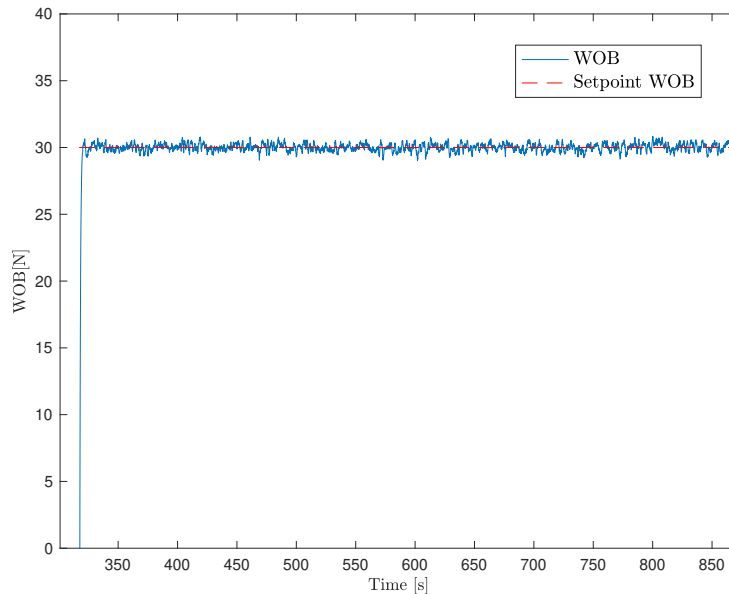


Figure 8.38: When using the NMPC for WOB control, the WOB is at the setpoint of $30N$.

By looking at the input charts, one can see that this is where the NMPC differentiates itself from the MPC

and PID. The TD input is constantly held in the interval of $[-0.2, 0.2]$ RPM, which leads to a more stable performance.

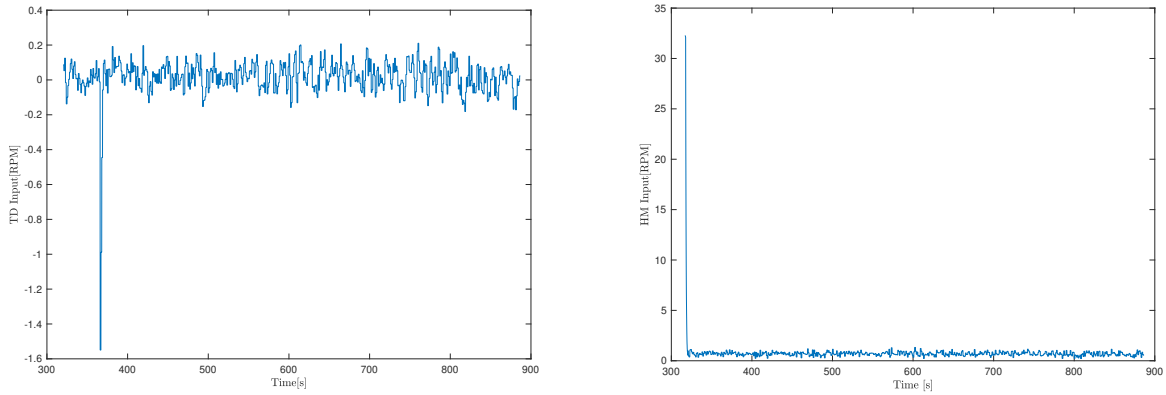


Figure 8.39: The TD RPM input of the NMPC is more stable than when using the MPC or PID.

To reduce the spikes in the input, one can enforce a higher penalization rate on the input rate change, as seen from the result shown in Figure 8.40. This can in turn reduce the quality of path following, such that it must be changed with regards to this.

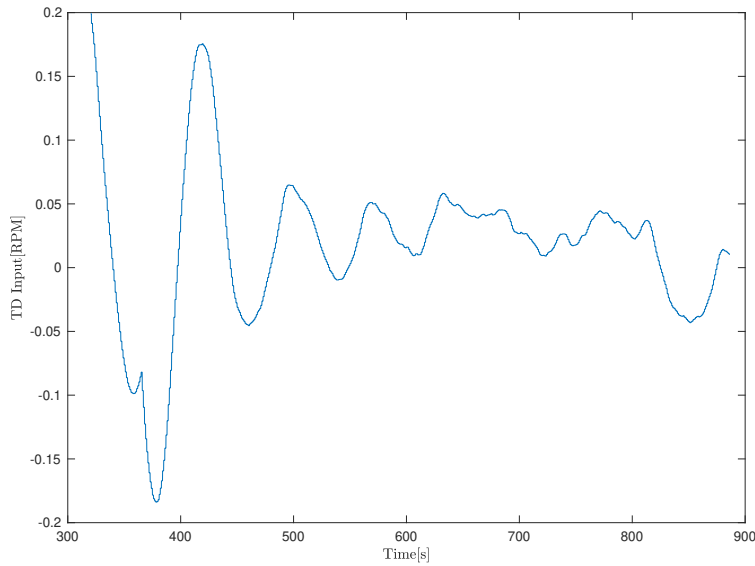


Figure 8.40: Input rate change can be penalized harder with the NMPC to reduce actuator rate change.

The score of the NMPC is as expected, with being lower than when using the MPC and PID controllers for directional control, as given by

$$\text{NMPC-Score}_{\text{cs-real}} = 0.0557 \quad (8.59)$$

$$\text{NMPC-Score}_{\text{cs-estimated}} = 0.0492 \quad (8.60)$$

$$\text{NMPC-Score}_{\text{nmpc-real}} = 0.0411 \quad (8.61)$$

$$\text{NMPC-Score}_{\text{nmpc-estimated}} = 0.0524. \quad (8.62)$$

Even though it seems like the NMPC path generation method beats cubic spline interpolation in this case, a closer look at the drilled path in Figure 8.41 reveals that it is merely based on luck because of the included bias. In this case, the bias helps the drill bit in getting closer to the reference coordinates.

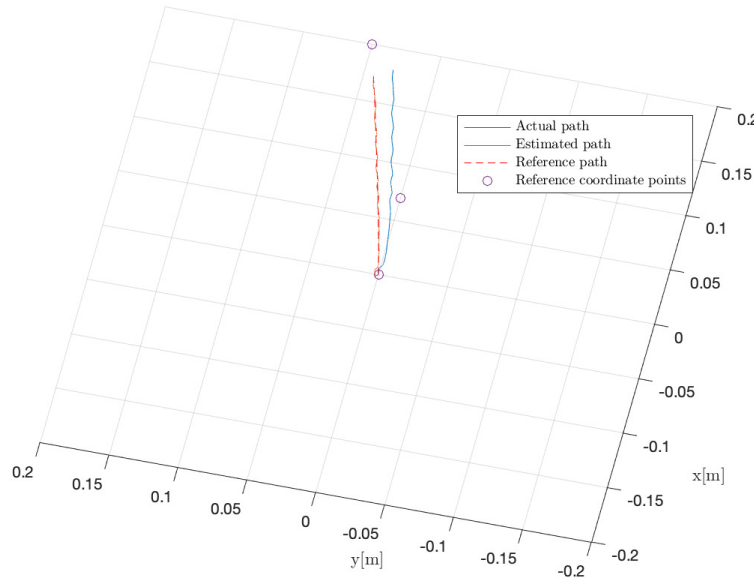


Figure 8.41: The bias helps the drill bit in getting closer to the reference coordinates.

These results are quite alike even when the rate change penalization is set higher to prevent input spikes. The real and estimated error between the actual path and the reference points is shown by the following equations

$$D_r = \begin{bmatrix} 0.0001 & 0.0161 & 0.0414 \end{bmatrix} \quad (8.63)$$

$$D_e = \begin{bmatrix} 0.0025 & 0.0141 & 0.0326 \end{bmatrix}. \quad (8.64)$$

8.4.3 ROP Failure during Drilling

When doing these simulations, it is important to include cases where realistic failures might happen. One of these cases is with loss of downhole power output, such as the PDM getting stuck. For this, there has been implemented functionality to make $\text{ROP} = 0$ to simulate the loss of downhole power output. In this case, the state machine will notice, and transition itself to the safety sequence. Using the MPC as controller, cubic spline interpolation for reference generation and the same reference points as used earlier in this section, one

gets the hoisting motor position and WOB data with the `drilling_state` as shown in Figure 8.42.

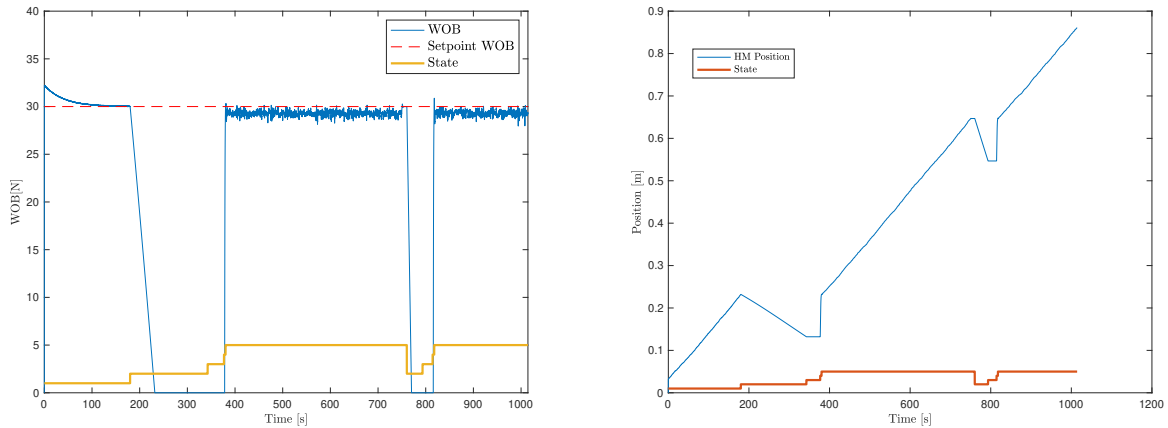


Figure 8.42: The state machine does its job as the hoisting motor position and WOB values look reasonable with regards to ROP failure.

As seen, the state machine handles the unexpected event of a Rate of Penetration (ROP) failure well. It does the same operation as it does initially before starting the directional drilling phase, as seen by the changing `drilling_state` shown in the plots. It is also worth noting that the simulation takes a longer time now, as the drill needs to spend time going through the safety sequence.

8.5 Kalman Filtering

8.5.1 Observability

Since the system is nonlinear, it is hard to determine the observability of the system. As mentioned in Section 6.6, one way to get a pinpoint is to linearize the system and check the linearized system's observability properties. As seen, the observability matrix given in equation (6.51) did not have full rank. The rank of \mathcal{O} is equal to 3, when full rank requires the rank of 6. This does not, on the other hand, imply that the nonlinear system is not observable.

Another way to determine the observability of the nonlinear system is to check that the error covariance matrix \mathbf{P} is positive definite at all times and upper bounded, that is

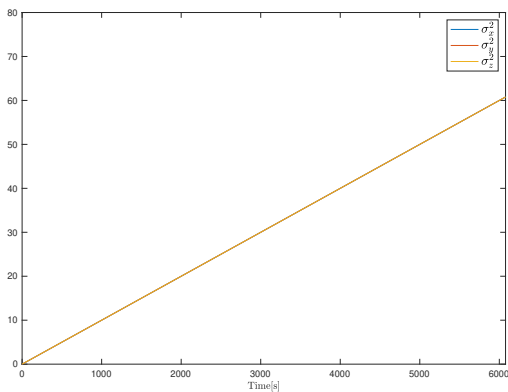
$$p_{min}\mathbf{I} \leq \mathbf{P}(t) \leq p_{max}\mathbf{I} \quad (8.65)$$

where p_{min} and p_{max} are strictly positive constants. If this is the case, it is guaranteed that the estimate converges exponentially to the actual states, Globally Exponentially Stable (GES) [10]. With the initial error

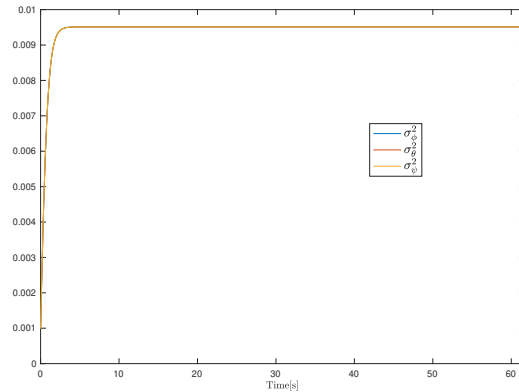
covariance matrix

$$\mathbf{P}_0 = \begin{bmatrix} 0.003 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.003 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.003 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.003 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.003 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.003 \end{bmatrix}, \quad (8.66)$$

the matrix \mathbf{P} stays positive definite during the entire drilling operation. Based on equation (8.65) and looking at Figure 8.43, the expectations are that the estimates of the orientation would converge to the actual states. This is not true for the position estimation however, as seen in Figure 8.43a. The error covariances keep on growing, and the covariances are not upper bounded.



(a) Error covariance from the position estimation.



(b) Error covariance from the orientation estimation.

Figure 8.43: Error covariance with regards to position and orientation estimation.

By comparing the position estimation with the true position, it can be seen in Figure 8.44 that the estimation error keeps on growing as the simulation is running. This is happening despite having no bias in the measurements. When looking at the orientation estimation, one can see that the error is bounded, as seen in Figure 8.43b.

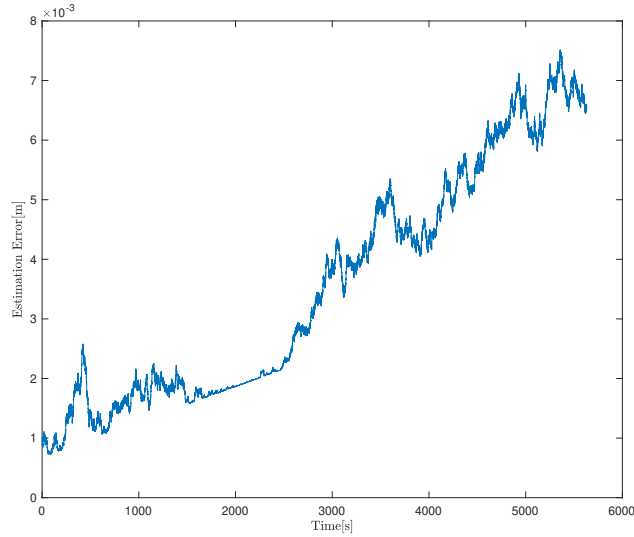


Figure 8.44: Estimation error for position estimation using an Extended Kalman Filter (EKF) without biases in the orientation measurements. The estimation error keeps on growing as the simulation goes on.

The system does not seem to be observable when looking at the growing error covariance and estimation error for position. On the other hand, the estimation error is small. Figure 8.44 shows that the largest estimation error is $0.0074m$, which is a reasonable estimate.

When looking at the orientation error in Figure 8.45, one can see that it is not increasing, and is bounded within an interval. It only fluctuates between the maximum and minimum of this interval because of noise.

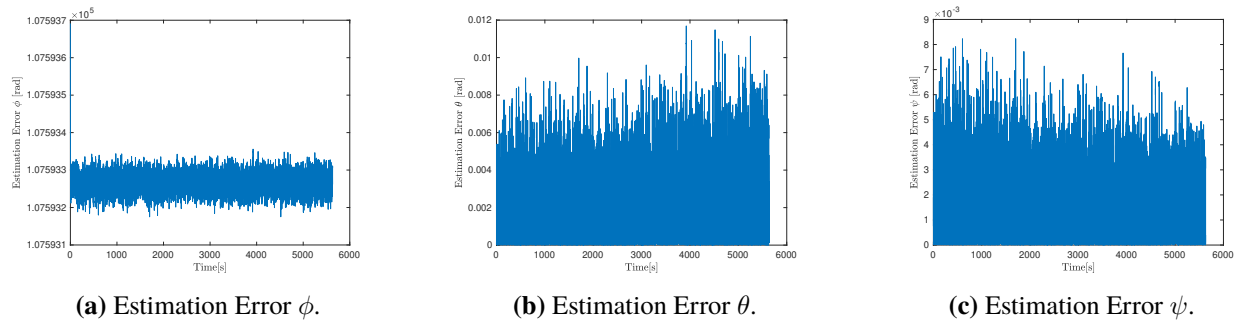
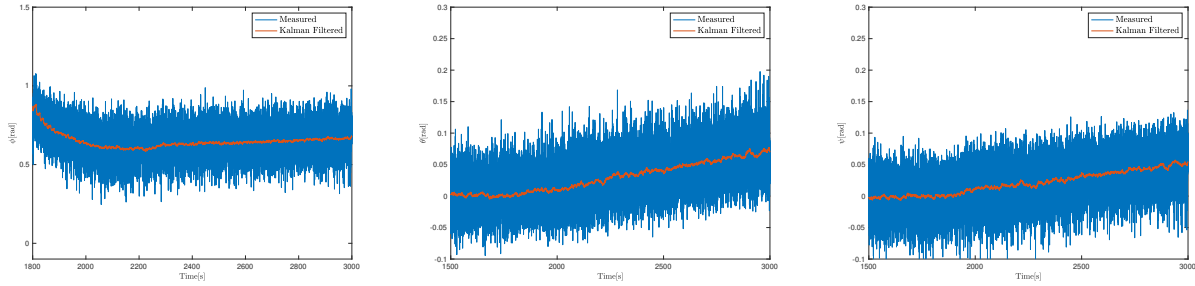


Figure 8.45: Estimation error of angles, using an EKF without bias in the measurements. The estimation error is created by the filtering of the process noise by the Kalman filter.

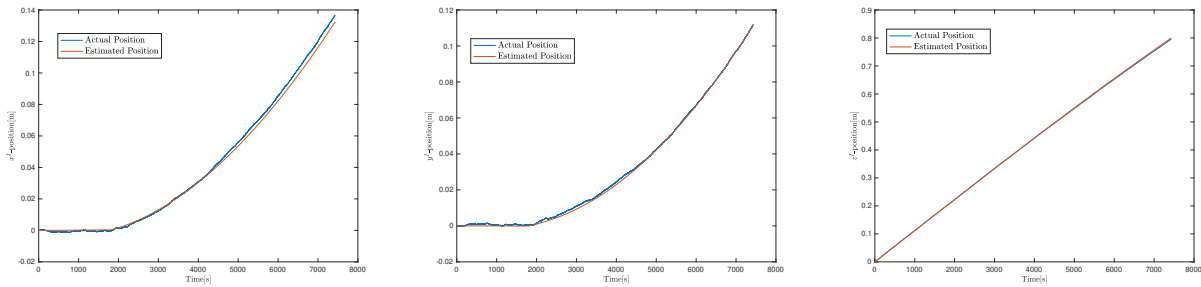
8.5.2 Kalman Performance

As seen in Figure 8.46, the EKF can very effectively filter out measurement noise and process noise. The estimation of position is also very accurate as seen in Figure 8.47. As explained above it can be seen that the estimation is less accurate towards the end of the simulation. The estimation error grows despite no bias in the measurement.



(a) Measured angle ϕ vs. filtered angle. (b) Measured angle θ vs. filtered angle. (c) Measured angle ψ vs. filtered angle.

Figure 8.46: Measured orientation angles versus estimated orientation angles obtained by using an EKF



(a) Estimated angle x vs. actual x . (b) Estimated angle y vs. actual y . (c) Estimated angle z vs. actual z .

Figure 8.47: Estimated position vs. actual position given in the inertial frame.

As seen in Figure 8.46, the EKF handles noise well. This is because the values of the \mathbf{Q} is chosen relatively low compared to the matrix \mathbf{R} . This renders a slower filter, but the noise filtering gets better. This is wanted behavior since the drilling process is slow, and the rate of change in azimuth is low.

When entering the orientation state before the directional drilling begins, the drill bit needs to be oriented to the initial orientation reference. This is done by the initial orientation controller, which has a faster response time than the controller used during the directional drilling state. This causes an overshoot of the actual state as seen in Figure 8.48. Before starting the directional drilling phase, it is necessary to let the EKF converge to make sure that the actual state equals the initial orientation. Starting with the actual angle deviating too much from the initial orientation can leave the system in a state where it can not recover, and therefore not be able to follow the path.

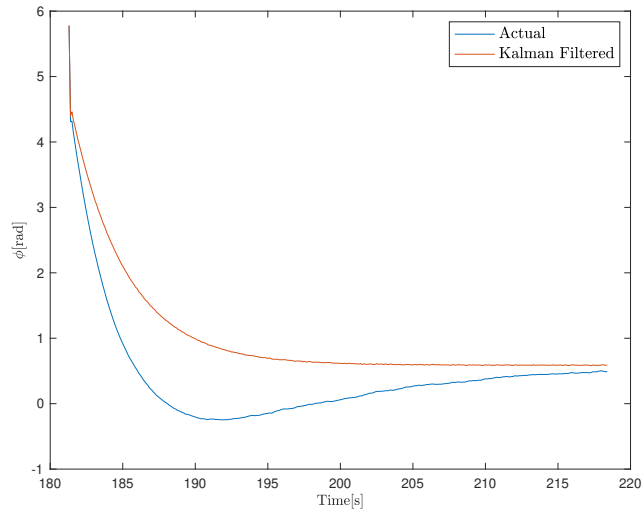


Figure 8.48: Choosing small values of Q renders a slow filter. In the figure it can be seen that the EKF lags behind the actual state.

8.6 Note about Choice of Coordinate Frames

As there are several coordinate frames that have to be implemented in order to create a realistic simulation system, it is important that they are defined properly. The initial definition of the drill-bit frame gave unexpected bugs, which is the reason for using the x -axis out of the drill bit as the main design given in Section 6.1.4. In this section, the discovery of the bug and its solution is presented.

8.6.1 Inertial Frame

The inertial frame has been defined with a positive z -axis down into the stone. Using the right-hand rule, the x - and y -axis have been defined according to Figure 8.49 where the grey block is the stone.

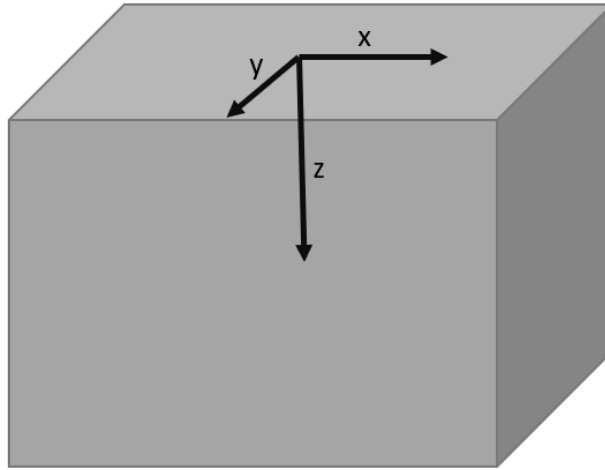


Figure 8.49: The defined inertial frame with the stone being the grey block.

8.6.2 Drill Bit Frame: z -axis Out of Drill Bit bit

With the defined inertial frame, the drill bit must be defined relative to this. The most intuitive approach is to define the drill bit coordinate frame the same way as the inertial frame, with the z -axis pointing out of the drill bit. This means that the ROP only provides movement along the z -axis of the drill bit coordinate system. An example of how these coordinate frames may look relative to each other is shown in Figure 8.50.

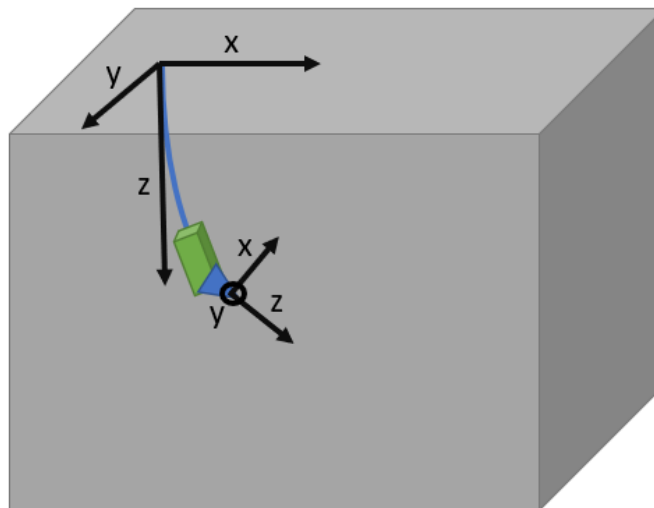
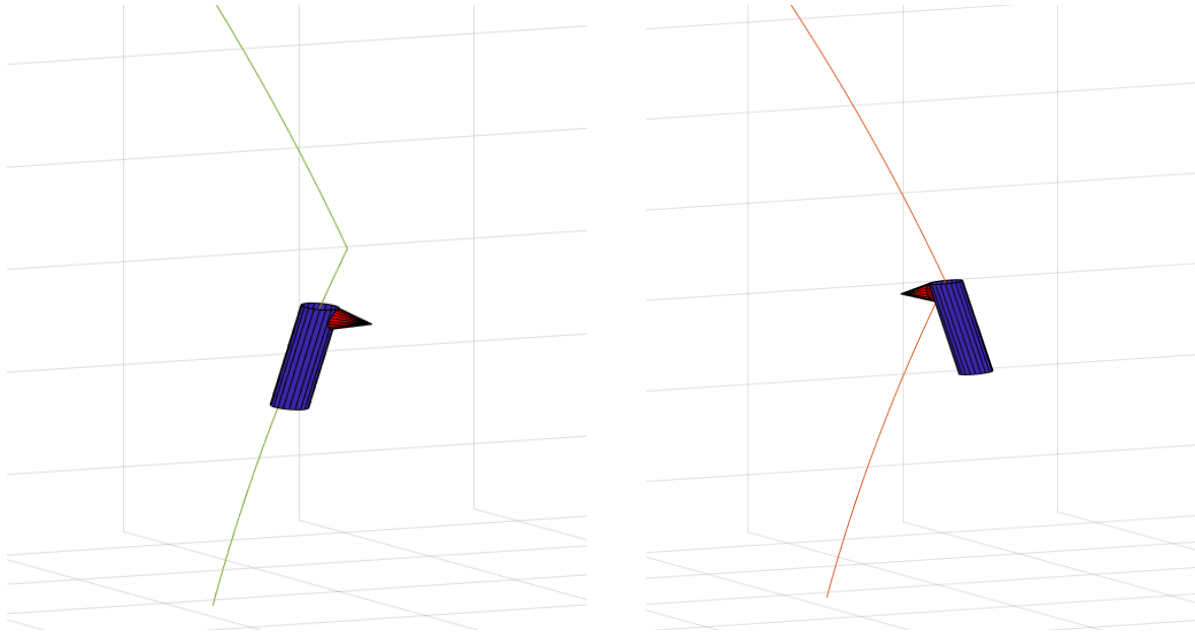


Figure 8.50: The defined drill bit frame with the stone being the grey block, and z -axis out of the drill bit.

By using this coordinate frame system, the top drive input, which rotates the BHA and drill bit, creates a

roll motion around the z -axis, while the inclination pitch angle creates a pitch rotation around the y -axis. After numerous simulations, it was discovered a few weird dynamics regarding the realistic constraints of the BHA and drill bit. With visualization of the BHA, it became apparent that creating a roll motion around the z -axis gave problems. An example is shown in Figure 8.51, where the top drive first is kept constant to create an inclination angle, and then turned 180° .



(a) Orientation of drill bit right before $\frac{\pi}{2}$ rotation about z^b . (b) Orientation of drill bit right after $\frac{\pi}{2}$ rotation about z^b .

Figure 8.51: Encountered a bug when the body frame was defined with the z^b -axis pointing along down the BHA towards the drill bit.

Realistically, this should only rotate the drill bit, while the BHA has the same position on the drill path. As seen, not only the drill bit is rotated around the path, but also the BHA. The reason for this is because there is a roll rotation around the z -axis instead of the x -axis as used in [6] from inertial to drill bit frame. By defining the body frames this way, the pitch angle does not get rotated because of the multiplication order of the rotation matrices. As such, the pitch angle will increase linearly.

8.6.3 Drill Bit Frame: x -axis Out of Drill Bit

To investigate the previous problem of using the z -axis of the drill bit frame out of the drill bit, it has been tried to use the same coordinate frames used in [6], which uses the roll, pitch and yaw angles around the x -, y - and z -axis', respectively. This means that the axis pointing out of the drill bit now will be the x -axis, as shown in Figure 8.52.

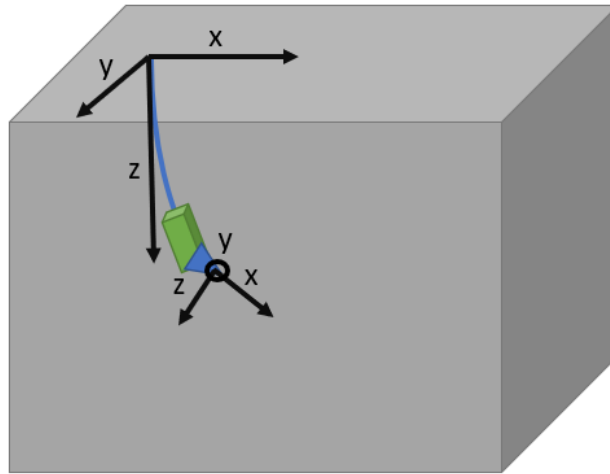
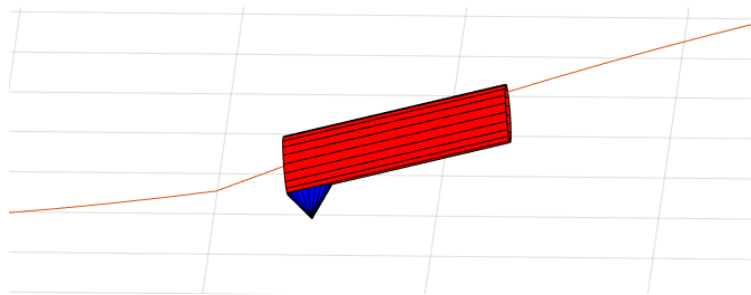
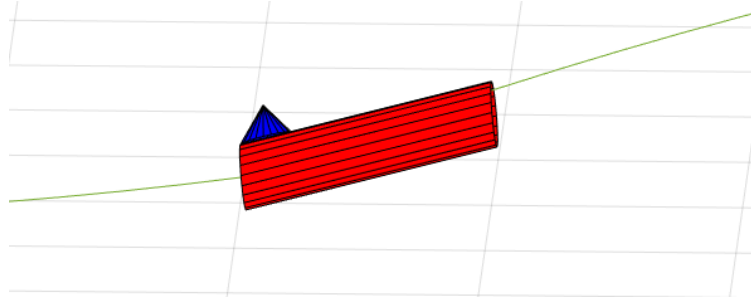


Figure 8.52: The defined drill bit frame with the stone being the grey block, and x -axis out of the drill bit.

When comparing this coordinate system to the previous one with the z -axis out of the drill bit, it can be seen that the difference is just a simple rotation of -90° around the y -axis, such that it is easy implemented by rotating the reference points with $R_y(-90^\circ)$, and then rotating the result back. The problem with the previous drill bit coordinate system is not present when using this coordinate system scheme, as seen in the example simulation in Figure 8.53.



(a) The drill bit orientation is facing down and the BHA is on the path.



(b) The drill bit rotates 180° around the x -axis and the BHA has the same position.

Figure 8.53: The roll motion around the x -axis mitigates the previous problem of BHA rotation.

8.6.4 Control Frame as Solution

By defining the x -axis out of the drill bit, an additional frame is needed between the inertial and drill bit frame. This is defined to be called the control frame, and only rotates the reference points and output values -90° around the y -axis. As explained, this is done to avoid the Gimbal-lock. Another solution to this problem would be to use quaternions for angle representation. By using quaternions, the inertial control frame is not needed, but instead defining the initial orientation as $\frac{\pi}{2}$.

8.7 Discussion and Future Work

As there are multiple dynamics of the physical system, they have not all been implemented and taken in consideration with regards to design. In this section, the discussion around these flaws, as well as future work will be presented.

8.7.1 State Dynamics

As the main goal of this simulation system has been to verify that certain theories and controller schemes would work, the essential parts of the physical rig's state dynamics have been designed and implemented. However, there are still several dynamics that can be included for more accurate state representation. These include drill pipe mechanics and ROP dynamics that include drill bit and stone characteristics.

8.7.2 Reference Generation

In this thesis, cubic spline interpolation and circular interpolation has been used to generate reference paths. As these create paths that only considers going through the given reference coordinates, they do not actually optimize for competition score while taking regards to the physical constraints. As such, further work on the simulation system should put more emphasis on methods like using an NMPC for path generation such that optimization for competition score is done when generating the controller's references.

8.7.3 Controller Options

As the results that have been presented only show a few examples of simulation with some reference points and controller parameters, there is still a lot to explore with regards to testing under different conditions. Also, an important takeaway was the immense effect dynamic angle reference had on using MPC and NMPC for directional control. However, when using the PID controller for directional control, one got infeasible simulations under certain conditions. To prevent this, it has been theorized that static angle references may be used, and should be tested. This is due to the fact the PID controller will be less aggressive, as it will not try to do the infeasible turns of 180° due to the dynamic angles that gets calculated between the current noisy position estimate and reference path.

8.7.4 State Estimation

Due to the fact that noisy orientation measurement is used for position calculation, the estimated position does not converge to the actual position. As such, it should be further investigated if any positional measurements can be gathered through other sensors. This way, a better estimation of the position can be calculated, which can give a better overall performance.

9 Conclusion and Recommendations for Upcoming Teams

The work in this master's thesis consists of the design, implementation and testing of a simulation system for an autonomous miniature scale drilling rig with the main purpose of helping the future NTNU teams in the annual Drillbotics competition. The simulation system can be summarized by the following points:

- **State Dynamics:** As the foundation of the simulation system, the state dynamics of the physical rig was defined through the use of control theory and drilling theory, and then implemented in MATLAB and Simulink. The state dynamics are not complete, as the ROP and drill string dynamics should be designed and implemented.
- **Controllers:** A Model Predictive Controller (MPC), Nonlinear Model Predictive Controller (NMPC) and Proportional-Integral-Derivative Controller (PID) have been designed and implemented for control the system. The MPC and NMPC are implemented with the MPC toolbox in MATLAB, and the PID weights have been tuned by pole placement methods. The tuned parameters seem to give good results in the simulations, but should be further tested on the physical system.
- **Reference Generation:** Reference generating algorithms using cubic spline interpolation, circle interpolation, Nonlinear Model Predictive Controller (NMPC) and angle references have been used for directional reference. The methods work well in different cases based on the mechanical constraints and given coordinate points. By running the simulation based on these constraints, one should be able to determine which method to use in each case.
- **State Estimation:** For realistic simulation, noise and bias has been added, as well as an Extended Kalman Filter (EKF) for state estimation. Because of process noise and measurement noise and bias, the positional estimate error grows unbounded. As such, sensors for positional measurements in the physical system should be considered, as well as testing the result of such an implementation in the simulation system.

The importance of using a simulation system to test theories and calculations before doing anything on a physical system has made itself apparent. There have been multiple bugs with the initial design with regards to many aspects of the system that were assumed to be correct. Both the coordinate frames representation and state dynamics had shown themselves to be wrong, but through multiple simulations and analysis, they were found and fixed. It is clear that during the next Drillbotics competition, there should be done heavy testing of all calculations and assumptions through a simulation system to check that everything behaves as expected. The hope is that the simulation program presented in this paper will be of great use for the next team, helping them with creating a successful physical autonomous drilling rig that ultimately wins them the competition!

Bibliography

Literature

- [1] NTNU, ed. *BRU21*. Mar. 24, 2020. URL: <https://www.ntnu.edu/bru21>.
- [2] DSATS, ed. *DSATS*. Mar. 24, 2020. URL: <https://connect.spe.org/dsats/home>.
- [3] Drillbotics, ed. *Drillbotics*. Mar. 24, 2020. URL: <https://drillbotics.com/about-drillbotics/>.
- [4] DrillboticsTeam2020. "Design Report NTNU - Drillbotics™ 2020 Phase I". Thesis. Department of Petroleum Engineering and Department of Engineering Cybernetics, 2019.
- [5] R.W. Beard and T.W. McLain. *Small Unmanned Aircraft. Theory and Practice*. Princeton University Press, 2012.
- [6] O Egeland and J.T. Gravdahl. *Modeling and Simulation for Automatic Control*. Marine Cybernetics, 2003.
- [7] J.G. Balchen, T. Andresen, and B.A Foss. *Reguleringsteknikk*. Department of Engineering Cybernetics, 2016.
- [10] T.I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, 2011.
- [11] Farbod Akhavan Niaki, ed. *Prediction and control horizon*. May 9, 2020. URL: https://www.researchgate.net/figure/Model-Predictive-Control-The-MPC-controller-uses-a-model-of-the-process-to-predict_fig1_317057942.
- [13] MathWorks, ed. *Understanding Kalman Filters*. Nov. 12, 2019. URL: <https://www.mathworks.com/videos/series/understanding-kalman-filters.html>.
- [14] Harveen Singh Chadha, ed. *Extended Kalman Filter: Why do we need an Extended Version?* May 11, 2020. URL: <https://towardsdatascience.com/extended-kalman-filter-43e52b16757d>.
- [16] A.M. Kvarving. *Natural cubic splines*. Ed. by Department of Mathematical Sciences Norwegian University of Science and Technology. URL: <https://www.math.ntnu.no/emner/TMA4215/2008h/cubicsplines.pdf>.
- [17] B. Brechan et al. *Drilling, Completion, Intervention and P&A - design and operations*. Department of Geoscience and Petroleum, 2017.
- [18] PetroWiki, ed. *Directional Deviation Tools*. Nov. 12, 2019. URL: https://petrowiki.org/Directional_deviation_tools.
- [20] J. J. Azar and G. Robello Samuel. *Drilling Engineering*. PennWell Books, 2007.
- [21] B. Brechan, S. Hovda, and P. Skalle. *Introduction to Drilling Engineering*. Department of Geoscience and Petroleum, Jan. 30, 2017.
- [22] MechaniCalc, ed. *Column Buckling*. Nov. 1, 2019. URL: <https://mechanicalc.com/reference/column-buckling>.

- [23] A. J. Adams et al. *The Barlow Equation for Tubular Burst: A Muddled History*. Society of Petroleum Engineers, Mar. 6, 2018. DOI: <https://doi.org/10.2118/189681-MS>.
- [24] A. T. Bourgoyne Jr et al. *Applied Drilling Engineering*. 2nd ed. Society of Petroleum Engineers, 1991.
- [25] Fridtjov Irgens. *Fasthetslære*. 7th ed. Tapir Akademisk Forlag, 2006.
- [26] A. Austin and J.H. Swannell. “Stresses in a pipe bend of oval cross-section and varying wall thickness loaded by internal pressure”. In: *International Journal of Pressure Vessels and Piping* 7.3 (1979), pp. 167–182. ISSN: 0308-0161. DOI: [https://doi.org/10.1016/0308-0161\(79\)90016-4](https://doi.org/10.1016/0308-0161(79)90016-4). URL: <http://www.sciencedirect.com/science/article/pii/0308016179900164>.
- [27] H. E. Helle, M. U. Azam, and J. M. Montoza. “Design and Implementation of an Autonomous Miniature Drilling Rig for Directional Drilling”. Thesis. Department of Geoscience and Petroleum, 2019.
- [28] Society of Petroleum Engineers (SPE) and Drilling Systems Automation Technical Section (DSATS), eds. *Drillbotics™ Guidelines*. Sept. 20, 2019. URL: <https://drillbotics.com/download/guidelines/2020-Drillbotics-Guidelines.pdf>.
- [29] M. B. Nåmdal. “Design and Implementation of an Autonomous Miniature Drilling Rig for Directional Drilling”. Thesis. Department of Engineering Cybernetics, 2019.
- [30] Lenze. *Lenze AC Motor*. URL: <https://www.lenze.com/en-no/products/motors/ac-motors-inverter-operation/ie3-m550-p-ac-motors/>.
- [31] Lenze. *Lenze Motor and Gearbox*. URL: https://www.lenze.com/fileadmin/lenze/documents/en/catalogue/CAT_GST_GFL_MF_15593808_en_GB.pdf.
- [32] Lenze. *Lenze Frequency Inverter*. URL: <https://www.lenze.com/en-no/products/inverters/control-cabinet-installation/8400-topline-frequency-inverters/>.
- [33] AEP Transducers. *TC4-AMP AEP Transducer*. URL: <http://www.aeptransducers.com/force-transducers/104-tc4-amp.html>.
- [34] Schneider Electric. *Top Drive Servo Motor BCH2MM1523CA6C*. URL: <https://docs-emea.rs-online.com/webdocs/14d7/0900766b814d71da.pdf>.
- [35] Schneider Electric. *Top Drive Servo Drive LXM28AU15M3X*. URL: <https://www.se.com/ww/en/product/LXM28AU15M3X/motion-servo-drive---lexium-28---single-and-three-phase-200...230-v---1.5-kw/>.
- [36] Omnetics, ed. *Nano 360 Plastic connectors*. Dec. 9, 2019. URL: <https://www.omnetics.com/products/micro-and-nano-circulars/nano-360-circulars-plastic#1803-tab2>.
- [37] InvenSense, ed. *ICM-20948 IMU from TDK InvenSense*. Dec. 9, 2019. URL: <https://www.invensense.com/wp-content/uploads/2016/06/DS-000189-ICM-20948-v1.3.pdf>.
- [38] Segger, ed. *20 pin connector*. Apr. 25, 2020. URL: <https://www.segger.com/products/debug-probes/j-link/technology/interface-description/>.

- [39] Johannes Korsawe, ed. *circlefit3d - fit circle to three points in 3d space*. May 2, 2020. URL: <https://se.mathworks.com/matlabcentral/fileexchange/34792-circlefit3d-fit-circle-to-three-points-in-3d-space>.
- [40] MATLAB, ed. *MPC box documentation*. Apr. 29, 2020. URL: <https://se.mathworks.com/help/mpc/ref/mpccontroller.html>.
- [41] L.T.Biegler C.Schmid, ed. *Quadratic programming methods for reduced hessian SQP*. Apr. 29, 2020. URL: <https://www.sciencedirect.com/science/article/abs/pii/S0098135494E00014?via%3DIihub>.
- [42] B. Brechan et al. *Drilling, Completion, Intervention and P&A - design and operations*. Department of Geosciense and Petroleum, 2017.
- [43] S. J. Quin and Badgwell T. A. *A Survey of Industrial Model Predictive Control Technology*. Control Eng. Practice, 2003.
- [44] D. E. Seborg et al. *Process Dynamics and Control*. John Wiley & Sons, Inc., 2011.
- [45] PetroWiki, ed. *Directional Drilling*. Nov. 12, 2019. URL: https://petrowiki.org/PEH:Directional_Drilling.
- [46] Jacques F. Smuts. *Ziegler-Nichols versus Cohen-Coon for PID tuning*. URL: <https://blog.opticontrols.com/archives/383>.
- [47] United States Department of Labor. *Safety Hazards Associated with Oil and Gas Extraction Activities*. URL: <https://www.osha.gov/SLTC/oilgaswelldrilling/safetyhazards.html>.
- [48] Senring, ed. *Electrical Swivel SNG012-12 by Senring*. Dec. 9, 2019. URL: <https://www.senring.com/high-speed-slip-ring/g012-12.html>.
- [49] Silicon Labs, ed. *EFM32 Gecko microcontroller*. Dec. 9, 2019. URL: <https://www.silabs.com/documents/public/data-sheets/efm32g-datasheet.pdf>.
- [50] Wikipedia, ed. *I2C communication*. Dec. 9, 2019. URL: <https://en.wikipedia.org/wiki/I%C2%B2C>.
- [51] Circuit Basics, ed. *UART Communication*. Dec. 9, 2019. URL: <http://www.circuitbasics.com/basics-uart-communication/>.
- [52] RS Pro, ed. *24V NC solenoid valve by RS Pro*. Dec. 9, 2019. URL: <https://uk.rs-online.com/web/p/solenoid-valves/1440801/>.
- [53] Omron, ed. *P2RF.05.S solid state relay by Omron*. Dec. 9, 2019. URL: https://no.rs-online.com/web/p/products/7948230/?grossPrice=Y&cm_mmc=NO-PPC-DS3A--google--3_NO_NO_Relay+Sockets_Omron_Exact--Omron+-+Relay+Sockets+-+7948230--p2rf05s&matchtype=e&kwd=306523915131&gclid=CjwKCAjw0ZfoBRB4EiwASUMdYWnB0LVI-fC-hJ0hafG4ENo2YJhZ_CqipyiAqMOV-FimtZQGwMWxKxoCC5EQAvD_BwE&gclidsrc=aw.ds.
- [54] Cat Pumps, ed. *5CP6120 pump by Cat Pumps*. Dec. 9, 2019. URL: http://www.catpumps.com/products/pdfs/5CP6120_B.pdf.

- [55] Aplisens A.S., ed. *PCE-28 pressure gauge by Aplisens*. Dec. 9, 2019. URL: <https://www.aplisens.com/pc-28.html>.
- [56] Analog Dialogue Ian Beavers, ed. *Gyroscope drift*. Dec. 9, 2019. URL: <https://www.analog.com/en/analog-dialogue/raqs/raq-issue-139.html#>.
- [57] Schlumberger, ed. *PowerPak - Steerable Motor Handbook*. Dec. 4, 2019. URL: <https://www.slb.com/-/media/files/drilling/brochure/powerpak-handbook-br.ashx>.
- [59] G. R. Samuel and S. Miska. *Analytical Study of the Performance of Positive Displacement Motor (PDM): Modelling for Incompressible Fluid*. Society of Petroleum Engineers, 1997. DOI: <https://doi.org/10.2118/39026-MS>.
- [60] H. R. Motahhari, G. Hareland, J. A. James, et al. “Improved drilling efficiency technique using integrated PDM and PDC bit parameters”. In: *Journal of Canadian Petroleum Technology* 49.10 (2010), pp. 45–52. DOI: <https://doi.org/10.2118/141651-PA>.
- [61] K. T. Lowe. *Selection and Integration of Positive Displacement Motors into Directional Drilling Systems*. University of Tennessee Honors Thesis Project, 2004.

Figures

- [8] Wikipedia, ed. *PID Controller figure*. Dec. 6, 2019. URL: https://en.wikipedia.org/wiki/PID_controller#/media/File:PID_en.svg.
- [9] National Instruments, ed. *PID Response*. Dec. 6, 2019. URL: <https://www.ni.com/en-no/innovations/white-papers/06/pid-theory-explained.html>.
- [12] MathWorks, ed. *Understanding Kalman Filters*. Nov. 19, 2019. URL: <https://www.mathworks.com/videos/series/understanding-kalman-filters.html>.
- [15] S3 Data Science, ed. *Euler Stability*. Dec. 6, 2019. URL: <http://www.s3datascience.com/z-s3-data/educating-the-user/a-brief-history-of-calculation>.
- [19] Micon Downhole Tools, ed. *Positive Displacement Motors (PDM)*. Dec. 3, 2019. URL: https://micon-drilling.de/Download/Catalog_PDM_EN.pdf.
- [58] Control Engineering, ed. *Cohen Coon Process Gain*. Dec. 6, 2019. URL: <https://www.controleng.com/articles/tuning-pid-control-loops-for-fast-response/>.

Appendix

A Matlab Code

A.1 Initializing Script

```
1 clear all;
2 close all;
3 %% Initialize Hoisting motor
4 run('init_hoisting_motor')
5
6 %% Initialize controller gains
7 run('controller_gains');
8
9 %% Initialize MPC weights
10 run('mpc_weights')
11
12 %% Initialize controller set points
13 run('controller_set_points');
14
15 %% Create drilling reference
16 trajectory_type = 2; % NMPC = 2 Cubic spline = 1. Circular trajectory =
    0.
17 run('create_reference');
18
19
20 %% Drilling info
21 run('drilling_parameters')
22
23 %% initial conditions
24 x_0 = [0;0;0];
25 w_0 = [initial_orientation;0;0];
26
27 %% Kalman filter
28 run('kalman_filter');
29
30 %% Load MPC
31 load('mpc1')
32
33 %% Run simulation
```

```

34 nx = 7; ny = 7; nu = 2;Ts = 1; p = 10; c = 5;
35 [nlobj,~,~] = PlanningControlDirectionalDrilling(nx,ny,nu,p,c,Ts,0,
    initial_orientation,pts);
36 %sim('Drilling');
37 %run('plot_drilled');
38 % run('plot_wob_data');

```

```

1 %Calculate bent sub
2 l1_bha = 0.15; %Length form
    stabilizer to bend
3 l2_bha = 0.085; %Length form bend
    to tip of drill bit
4
5 if trajectory_type == 0
6     DLS = (radians / curve_length);
7     alpha_bha = (1/2)*DLS*(l1_bha + l2_bha); %Calculate bent sub
    based on circle reference
8 else
9     alpha_bha = 5.1*pi/180; %Define custom
    bent sub
10    DLS = 2*alpha_bha/(l1_bha + l2_bha);
11    %DLS = 0.7;
12 end
13
14 R_Bdb = [ %Calculate rotation
    matrix from BHA to bent sub
15    cos(alpha_bha) 0 sin(alpha_bha);
16    0 1 0;
17    -sin(alpha_bha) 0 cos(alpha_bha)
18    ];
19
20 %State machine
21 max_depth = p_z(end);
22 hoist_height = 0.1;
23 orientation_error = 0.05;
24 wob_threshold = 30;
25
26 %Rate of penetration
27 ROP = [

```

```
28     0.0067/60 * 10;  
29     %0.01;  
30     0;  
31     0  
32     ];
```

A.2 Kalman Model

```
1
2 function x = kalman_orientation_transition(x,u)
3     dt = 0.1;
4     phi_d = 2*pi/60*u(1);
5     theta_d = u(3)*u(2);
6     alpha_bha = 0.0890;
7     x(4:6) = x(4:6) + calculate_angle_derivative(x(4:6),[phi_d;theta_d
8         ;0])*dt;
9     while x(4) > 2*pi
10         x(4) = x(4) - 4*pi;
11     end
12     while x(4) < -2*pi
13         x(4) = x(4) + 4*pi;
14     end
15     Rdbb = [
16         cos(alpha_bha) 0 sin(alpha_bha);
17         0 1 0;
18         -sin(alpha_bha) 0 cos(alpha_bha)
19     ];
20     x(1:3) = x(1:3) + R_bi(x(4:6))*Rdbb*[u(2);0;0] * dt;
21 end
```

```
1 function y = myMeasurementFcn(x)
2 y = [0 0 0;0 0 0;0 0 0;1 0 0; 0 1 0;0 0 1]'*x;
3 end
```

A.3 NMPC Code

```
1 function dxdt = DrillingStateFunction(x,u)
2
3 dxdt = zeros(7,1);
4 ROP = [0.0067/60 * 10; 0; 0];
5 %ROP = [0.01;0;0];
6 DLS = 0.6658;
7 alpha_bha = 0.0782;
8 lead = 9e-2;
9 k_wc = 1000;
10 omega = [2*pi/60*u(1);DLS * ROP(1); 0];
11 diameter_hm = 30 * 10e-3;
12
13 Rdbb = [
14     cos(alpha_bha) 0 sin(alpha_bha);
15     0 1 0;
16     -sin(alpha_bha) 0 cos(alpha_bha)
17 ];
18 Ea = [
19     cos(x(5)) sin(x(4))*sin(x(5)) cos(x(4))*sin(x(5));
20     0 cos(x(4))*cos(x(5)) -sin(x(4))*cos(x(5));
21     0 sin(x(4)) cos(x(4))
22 ];
23 dxdt(4:6) = 1/cos(x(5)) * Ea*omega;
24
25 Rx = [
26     1 0 0;
27     0 cos(x(4)) -sin(x(4));
28     0 sin(x(4)) cos(x(4))
29 ];
30 Ry = [
31     cos(x(5)) 0 sin(x(5));
32     0 1 0;
33     -sin(x(5)) 0 cos(x(5))
34 ];
35
36 Rz = [
37     cos(x(6)) -sin(x(6)) 0;
38     sin(x(6)) cos(x(6)) 0;
```

```
39     0 0 1
40     ];
41 dxdt(1:3) = Rz*Ry*Rx*Rdbb*ROP;
42 dxdt(7) = (u(2)*1/60*lead/(diameter_hm*pi) - ROP(1)) * k_wc;
43 end
```



```

1 function [A,B] = DirectionalDrillingStateJacobian(x,u)
2 %DirectionalDrillingStateJacobian
3 A = zeros(7,7);
4 B = zeros(7,2);
5 adb = 0.089;
6 phi = x(4);
7 theta = x(5);
8 psi = x(6);
9 ROPx = 0.0067/60 * 10;
10 %ROPx = 0.01;
11 DLS = 0.7575;
12 lead = 9e-2;
13 k_wc = 1000;
14 diameter_hm = 30 * 10e-3;
15
16 A(1,4) = -ROPx*sin(adb)*(cos(phi)*sin(psi) - cos(psi)*sin(phi)*sin(
    theta));
17 A(1,5) = -ROPx*(cos(adb)*cos(psi)*sin(theta) + cos(phi)*cos(psi)*sin(
    adb)*cos(theta));
18 A(1,6) = -ROPx*(sin(adb)*(cos(psi)*sin(phi) - cos(phi)*sin(psi)*sin(
    theta)) + cos(adb)*cos(theta)*sin(psi));
19
20 A(2,4) = ROPx*sin(adb)*(cos(phi)*cos(psi) + sin(phi)*sin(psi)*sin(theta
    ));
21 A(2,5) = -ROPx*(cos(adb)*sin(psi)*sin(theta) + cos(phi)*sin(adb)*cos(
    theta)*sin(psi));
22 A(2,6) = -ROPx*(sin(adb)*(sin(phi)*sin(psi) + cos(phi)*cos(psi)*sin(
    theta)) - cos(adb)*cos(psi)*cos(theta));
23
24 A(3,4) = ROPx*sin(adb)*cos(theta)*sin(phi);
25 A(3,5) = -ROPx*(cos(adb)*cos(theta) - cos(phi)*sin(adb)*sin(theta));
26
27 A(4,4) = (DLS*ROPx*cos(phi)*sin(theta))/cos(theta);
28 A(4,5) = DLS*ROPx*sin(phi) + (DLS*ROPx*sin(phi)*sin(theta)^2)/cos(theta
    )^2;
29
30 A(5,4) = -DLS*ROPx*sin(phi);
31
32 A(6,4) = (DLS*ROPx*cos(phi))/cos(theta);

```

```
33 A(6,5) = (DLS*ROPx*sin(phi)*sin(theta))/cos(theta)^2;  
34  
35 B(4,1) = 2*pi/60;  
36 B(7,2) = lead/(diameter_hm*pi)*1/60 * k_wc;  
37 end
```

```

1 function [NMPC,path,cost] = PlanningControlDirectionalDrilling(nx,ny,nu
, p,c,Ts,create_path,initial_orientation,pts)
2     nlobj = nlmpc(nx,ny,nu); %creating the nonlinear mpc object
3     nlobj.Model.StateFcn = "DrillingStateFunction";
4     nlobj.Jacobian.StateFcn = @DirectionalDrillingStateJacobian;
5     nlobj.Ts = Ts;
6     nlobj.PredictionHorizon = p;
7     nlobj.ControlHorizon = c;
8     nlobj.Weights.ManipulatedVariablesRate = [1 1];
9     nlobj.OV(7).Max = 32;
10    nlobj.OV(7).Min = 28;
11    nlobj.OV(4).Max = initial_orientation + pi/12;
12    nlobj.OV(7).Min = initial_orientation - pi/12;
13    nlobj.MV(1).Max = 60;
14    nlobj.MV(1).Min = -60;
15    x0 = [0.2;0;0;initial_orientation;0;0;0];
16    u0 = zeros(nu,1);
17    validateFcns(nlobj,x0,u0);
18    NMPC = nlobj;
19
20    if create_path == 1
21        nlobj.Weights.ManipulatedVariablesRate = [0.1 0.1];
22        nlobj.Optimization.CustomCostFcn = "customCostFunction";
23        nlobj.Optimization.ReplaceStandardCost = true;
24        ref = zeros(3,7);
25        ref(1:3,1:3) = (pts)';
26        [~,~,info] = nlmpcmove(nlobj,x0,u0,ref);
27        path = info.Xopt(:,1:3)';
28        cost = info.Cost;
29    else
30        path = -1;
31        cost = -1;
32    end
33
34 end

```

A.4 Cubic Spline Reference Trajectory

```
1 function path = cubic_spline(p1,p2,p3,t)
2     % Apply interpolation for each x,y and z
3     tt = linspace(t(1),t(end),501);
4     xx = interp1(t,p1,tt,'spline');
5     yy = interp1(t,p2,tt,'spline');
6     zz = interp1(t,p3,tt,'spline');
7     path = [xx;yy;zz];
8 end
```

A.5 Circular Reference Trajectory

```
1 function [center,rad,v1n,v2nb] = calculate_circle_parameters(p1,p2,p3)
2 % Author: Johannes Korsawe
3 % E-mail: johannes.korsawe@volkswagen.de
4 % Release: 1.0
5 % Release date: 26/01/2012
6 % Default values
7 center = [];rad = 0;v1n=[];v2nb=[];
8 % check inputs
9 % check number of inputs
10 if nargin~=3,
11     fprintf('??? Error using ==> cirlefit3d\nThree input matrices are
12         needed.\n');rad = -1;return;
13 end
14 % check size of inputs
15 if size(p1,2)~=3 || size(p2,2)~=3 || size(p3,2)~=3,
16     fprintf('??? Error using ==> cirlefit3d\nAll input matrices must
17         have three columns.\n');rad = -2;return;
18 end
19 n = size(p1,1);
20 if size(p2,1)~=n || size(p3,1)~=n,
21     fprintf('??? Error using ==> cirlefit3d\nAll input matrices must
22         have the same number or rows.\n');rad = -3;return;
23 end
24 % more checks are to follow inside calculation
25 % Start calculation
26 % v1, v2 describe the vectors from p1 to p2 and p3, resp.
```

```

24 v1 = p2 - p1;v2 = p3 - p1;
25 % l1, l2 describe the lengths of those vectors
26 l1 = sqrt((v1(:,1).*v1(:,1)+v1(:,2).*v1(:,2)+v1(:,3).*v1(:,3)));
27 l2 = sqrt((v2(:,1).*v2(:,1)+v2(:,2).*v2(:,2)+v2(:,3).*v2(:,3)));
28 if find(l1==0) | find(l2==0), %#ok<OR2>
29     fprintf('??? Error using ==> cirlefit3d\nCorresponding input points
           must not be identical.\n');rad = -4;return;
30 end
31 % v1n, v2n describe the normalized vectors v1 and v2
32 v1n = v1;for i=1:3, v1n(:,i) = v1n(:,i)./l1;end
33 v2n = v2;for i=1:3, v2n(:,i) = v2n(:,i)./l2;end
34 % nv describes the normal vector on the plane of the circle
35 nv = [v1n(:,2).*v2n(:,3) - v1n(:,3).*v2n(:,2) , v1n(:,3).*v2n(:,1) -
        v1n(:,1).*v2n(:,3) , v1n(:,1).*v2n(:,2) - v1n(:,2).*v2n(:,1)];
36 if find(sum(abs(nv),2)<1e-5),
37     fprintf('??? Warning using ==> cirlefit3d\nSome corresponding input
           points are nearly collinear.\n');
38 end
39 % v2nb: orthogonalization of v2n against v1n
40 dotp = v2n(:,1).*v1n(:,1) + v2n(:,2).*v1n(:,2) + v2n(:,3).*v1n(:,3);
41 v2nb = v2n;for i=1:3,v2nb(:,i) = v2nb(:,i) - dotp.*v1n(:,i);end
42 % normalize v2nb
43 l2nb = sqrt((v2nb(:,1).*v2nb(:,1)+v2nb(:,2).*v2nb(:,2)+v2nb(:,3).*v2nb
        (:,3)));
44 for i=1:3, v2nb(:,i) = v2nb(:,i)./l2nb;end
45 % remark: the circle plane will now be discretized as follows
46 %
47 % origin: p1                normal vector on plane: nv
48 % first coordinate vector: v1n  second coordinate vector: v2nb
49 % calculate 2d coordinates of points in each plane
50 % p1_2d = zeros(n,2); % set per construction
51 % p2_2d = zeros(n,2);p2_2d(:,1) = l1; % set per construction
52 p3_2d = zeros(n,2); % has to be calculated
53 for i = 1:3,
54     p3_2d(:,1) = p3_2d(:,1) + v2(:,i).*v1n(:,i);
55     p3_2d(:,2) = p3_2d(:,2) + v2(:,i).*v2nb(:,i);
56 end
57 % calculate the fitting circle

```

```

58 % due to the special construction of the 2d system this boils down to
    solving
59 % q1 = [0,0], q2 = [a,0], q3 = [b,c] (points on 2d circle)
60 % crossing perpendicular bisectors, s and t running indices:
61 % solve [a/2,s] = [b/2 + c*t, c/2 - b*t]
62 % solution t = (a-b)/(2*c)
63 a = l1;b = p3_2d(:,1);c = p3_2d(:,2);
64 t = 0.5*(a-b)./c;
65 scale1 = b/2 + c.*t;scale2 = c/2 - b.*t;
66 % centers
67 center = zeros(n,3);
68 for i=1:3,
69     center(:,i) = p1(:,i) + scale1.*v1n(:,i) + scale2.*v2nb(:,i);
70 end
71 % radii
72 rad = sqrt((center(:,1)-p1(:,1)).^2+(center(:,2)-p1(:,2)).^2+(center
    (:,3)-p1(:,3)).^2);

```

```

1 function [points, radians] = calculate_circle_points(center,normal,
    radius,p_x,p_y,p_z)
2     step_length = 0.001;
3     dx = p_x(2) - p_x(1);
4     dy = p_y(2) - p_y(1);
5     if (atan2(dy, dx) > 0)
6         theta=2*pi:-step_length:0;
7     elseif (atan2(dy, dx) < 0)
8         theta=0:step_length:2*pi;
9     else
10        theta = 0;
11    end
12    v=null(normal);
13    points= repmat(center',1,size(theta,2))+radius*(v(:,1)*cos(theta)+v
        (:,2)*sin(theta));
14    start_point = find(points(3,:) >= p_z(1));
15    end_point = find(points(3,:) > p_z(end));
16    points = points(:,start_point(1):end_point(1));
17    radians = size(points,2) * step_length;
18 end

```

A.6 Initialize Hoisting Motor

```
1 lead_hm = 3*3e-2;
2 diameter_hm = 30 * 10e-3;
3 mu_hm = 0;
4 travel_relation_hm = pi*diameter_hm/(lead_hm);
5 alpha = atan(1/travel_relation_hm);
6 beta = atan(mu_hm);
7 ma_hm = atan(alpha)/atan(alpha/beta) * travel_relation_hm; %mechanical
   advantage
8 kw_hm = 45;
9 hp_td = 45 /0.746;
10
11 % weight cell
12 K_wc = 1000;%5000; %stiffness
```

A.7 Initialize Controllers

```
1 %% Top drive controller gains
2 kp_tdrpm = 10;
3 ki_tdrpm = 0.5;
4
5 %% Topdrive orientation controller gain
6 w0 = 0.2;
7 zeta = 1;
8 kp_orien = 0.5;
9 kd_orien = kp_orien/(2*w0*zeta)-1;
10 ki_orien = w0^2*(1+kd_orien);
11 %% Hoisting motor controller gains
12 zeta = 1.5;
13 w0 = 0.1;
14 kp_wob = 0.5;
15 kd_wob = kp_orien/(2*w0*zeta)-1;
16 ki_wob = w0^2*(1+kd_orien);
17
18
19 %% Directional drilling controller gains
20 w0_d = 0.1; zeta = 1;
21 kp_td_d = 0.1;
22 kd_td_d = kp_td_d/(2*w0_d*zeta)-1;
23 ki_td_d = w0_d^2*(1+kd_td_d);
24
25 w_0 = 10;
26 zeta = 1;
27 kp_wob_d = 10;
28 kd_wob_d = kp_wob_d/(2*w0*zeta)-1;
29 ki_td_d = w0_d^2*(1+kd_td_d);

1 %% Weight on the different outputs to the mpc and the nmpc
2 %% These weights are set in the simulink program
3 w_x = 0.1;
4 w_y = 2;
5 w_z = 2;
6 w_phi = 1;
7 w_theta = 0;
8 w_psi = 0;
```



```
9 w_wob = 10;
```

```
1 tag_rock_td_sp = 0;  
2 tag_rock_hm_sp = 30;  
3 hoist_up_td_sp = 0;  
4 hoist_up_hm_sp = -30;  
5 vertical_drilling_td_sp = 60;  
6 vertical_drilling_hm_sp = 30;  
7 initial_orientation_hm_sp = 0;
```

B Matlab Code in Simulink Blocks

B.1 Angle Calculation from Accelerometer

```
1 function angles = angles_from_accelerometer(A)  
2 ax = A(1);  
3 ay = A(2);  
4 az = A(3);  
5 psi = atan(ay/az);  
6 theta = -atan(ax/(sqrt(ay^2 + az^2)));  
7 angles = [theta;psi];  
8 end
```



Figure B.1: The Matlab-function block used to calculate θ and ψ using measurements from an accelerometer.

B.2 Calculation of Angle Derivatives

```
1 function dot_angles = calculate_angle_derivative(orientation,omega)
2     E_a = [
3         cos(orientation(2)), sin(orientation(1))*sin(orientation(2)),
4         cos(orientation(1))*sin(orientation(2));
5         0, cos(orientation(1))*cos(orientation(2)), -sin(orientation(1)
6         )*cos(orientation(2));
7         0, sin(orientation(1)), cos(orientation(1));
8     ];
9     dot_angles = 1/cos(orientation(2)) * E_a * omega;
10 end
```



Figure B.2: The Matlab-function block used to calculate the angle derivatives $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$.

B.3 Calculation of Reference Angle

```
1 function [corrected_ref, ref_raw]= calculate_reference_angle(dy,dz,
   orientation)
2     if dz == 0 && dy == 0
3         corrected_ref = orientation;
4         ref_raw = 0;
5     else
6         N = -4:4;
7         ref_raw = -atan2(-dy,-dz);
8         shifted_ref = ref_raw + 2*pi*N;
9         [d,ix] = min(abs(shifted_ref - orientation));
10        corrected_ref = shifted_ref(ix);
11    end
12 end
```

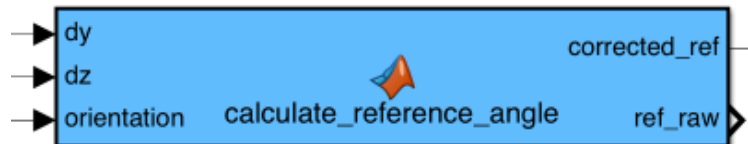


Figure B.3: The Matlab Fuction Block Used to Calculate the Reference Angle Based on The Position Error.

B.4 Finding the Reference Coordinates

```
1 function coordinates = get_coordinates(h,path)
2   [val,ix] = min(abs(path(1,:)-h));
3   if ix + 5 <= size(path,2)
4     coordinates = path(:,ix+5);
5   else
6     coordinates = path(:,end);
7   end
8 end
```

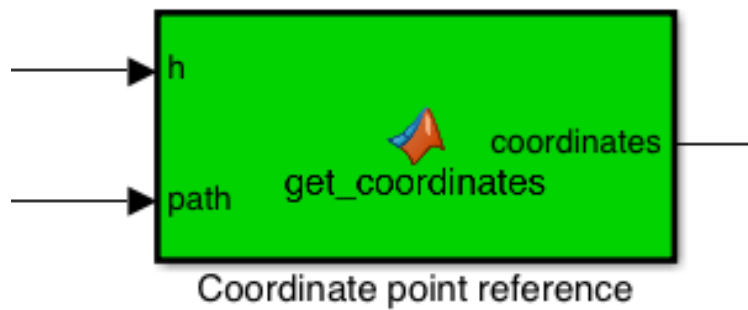


Figure B.4: Finding the Reference Coordinates Used By the Directional Controller.

B.5 Rotation Matrix $R_{bc}^{I^c}$

```
1 function R = R_Ib(orientation)
2     R_z = [
3         cos(orientation(3)) -sin(orientation(3)) 0;
4         sin(orientation(3))  cos(orientation(3)) 0;
5         0 0 1
6     ];
7
8
9     R_y = [
10        cos(orientation(2)) 0 sin(orientation(2));
11        0 1 0;
12        -sin(orientation(2)) 0 cos(orientation(2))
13    ];
14
15
16    R_x = [
17        1 0 0;
18        0 cos(orientation(1)) -sin(orientation(1));
19        0 sin(orientation(1))  cos(orientation(1))
20    ];
21
22    R = R_z * R_y * R_x;
23 end
```



Figure B.5: Matlab function block used to rotate the ROP from $\{b^c\}$ to $\{I^c\}$.

B.6 Calculate Coordinates to be Used for Angle Reference Calculation

```
1 function [y_r,z_r] = reference_point(pos, path)
2   [d, ix ] = min(abs(path(1,:) - pos(1)));
3   dz = abs(pos(3) - path(3,ix));
4   dy = abs(pos(2) - path(2,ix));
5   if dz < 0.001 && dy < 0.001
6       ix = ix + 5;
7   end
8   if pos(1) < 0.25
9       ix = ix + 100
10  end
11  if ix > size(path,2)
12      ix = size(path,2);
13  end
14  y_r = path(2,ix);
15  x_r = path(1,ix);
16  z_r = path(3,ix);
17 end
```

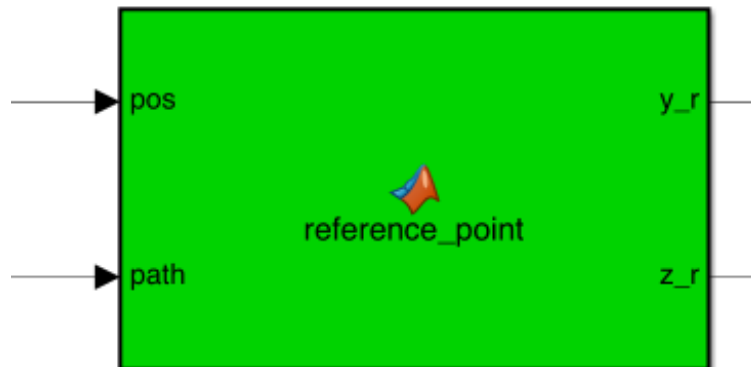


Figure B.6: The function block is used to calculate the reference used to calculate the angle reference.

C Simulink Program

C.1 Simulink Overview

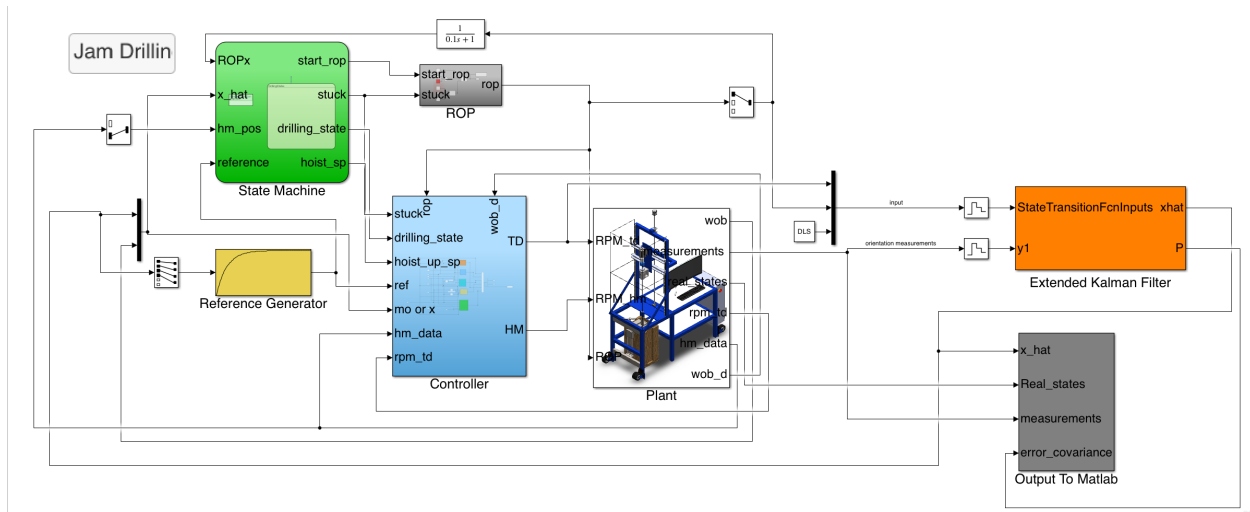


Figure C.7: Overview Over the Entire Simulink Program.

C.2 Reference Generator

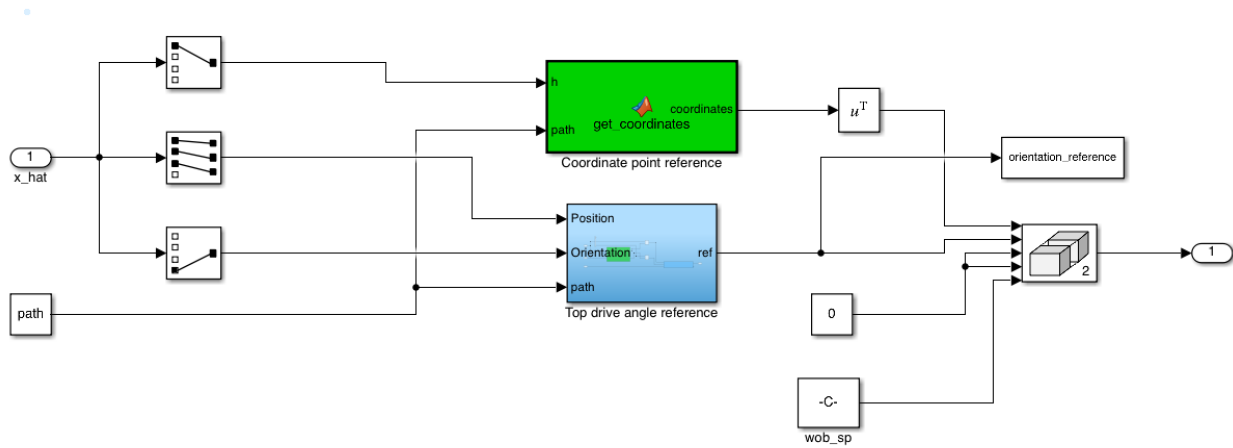


Figure C.8: The Reference Generation Block Used in Figure C.7.

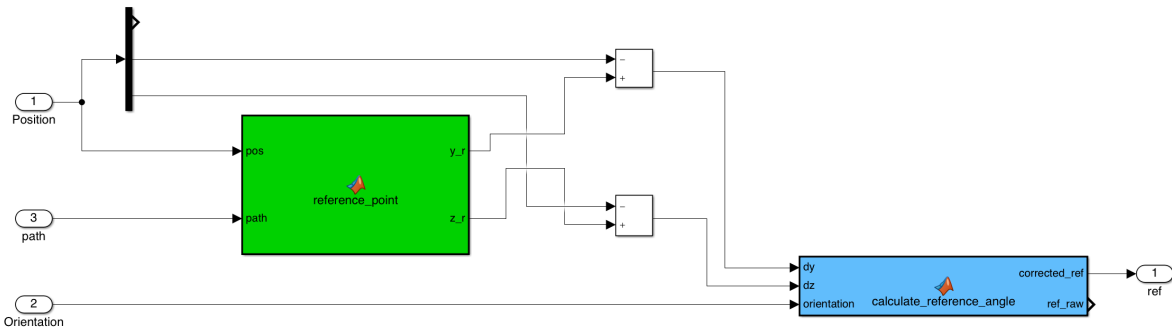


Figure C.9: The Figure Shows the Top Drive Angle Reference block used in figure C.8.

C.3 ROP Logic Implementation

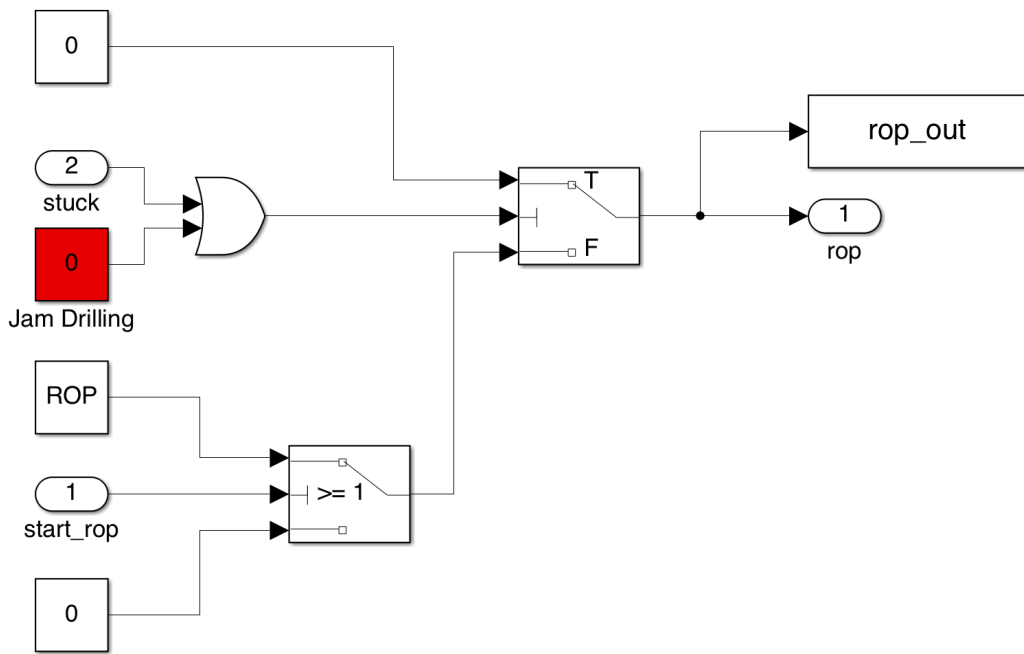


Figure C.10: The rate of penetration is modeled as a binary value. If it is on or of is determined by the state output from the state machine and the jam drilling button seen in C.7.

C.4 Plant Implementation

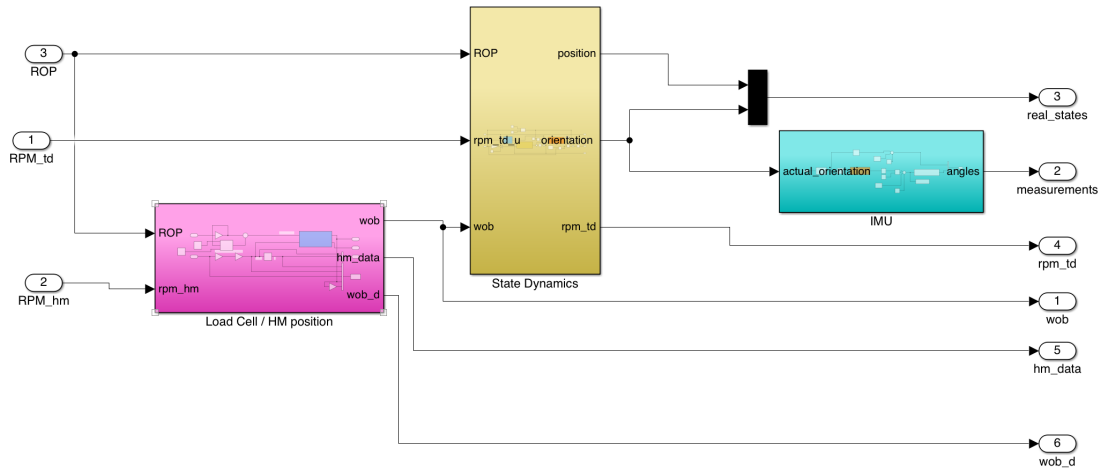


Figure C.11: The plant subsystem in figure C.7 contains state dynamics of the drill, and models for the Hoisting motor (HM) and Inertial Measurement Unit (IMU).

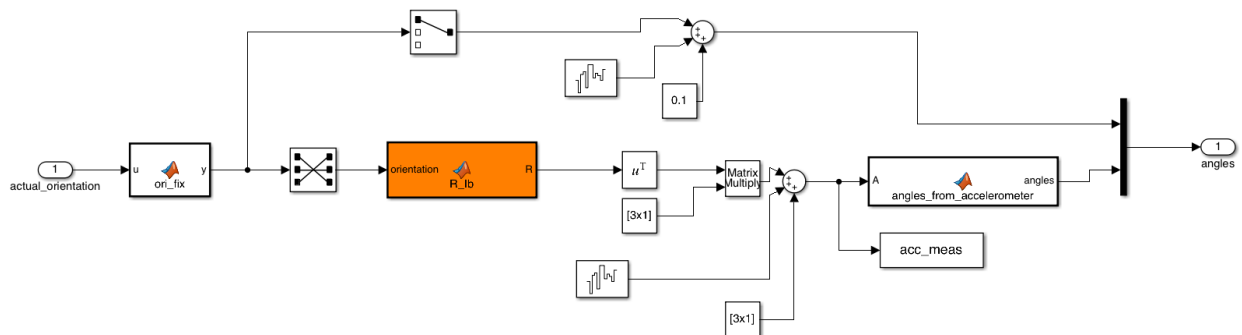


Figure C.12: The implementation of the IMU subsystem in the plant, seen in figure C.11

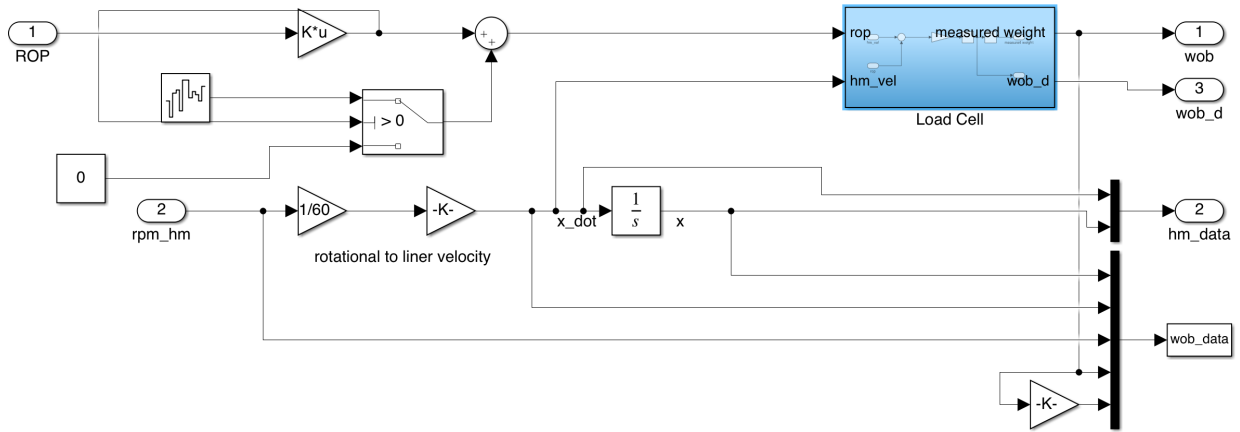


Figure C.13: The implementation of the LoadCell/HoistingMotor subsystem in the plant, seen in figure C.11.

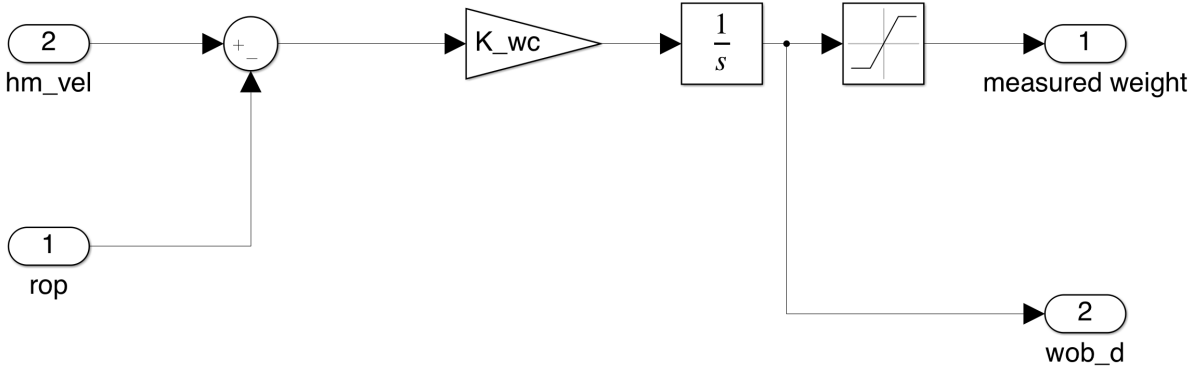


Figure C.14: The implementation of the load cell subsystem in the HM, seen in figure C.13.

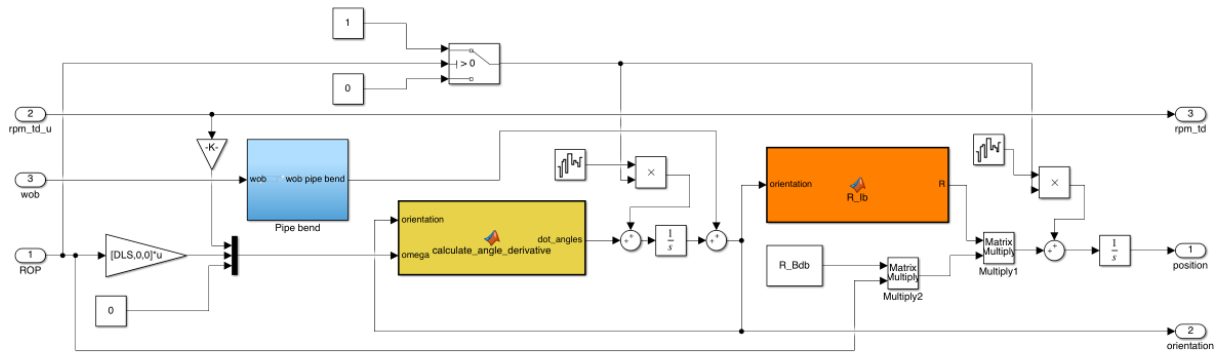


Figure C.15: The implementation of the state dynamics subsystem in the plant, seen in figure C.11.

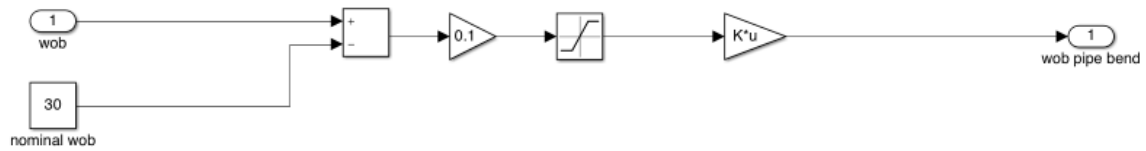


Figure C.16: Simple model for pipe bend when the Weight on Bit (WOB) exceeds its set point to generate a bigger pitch angle. The subsystem is found in the state dynamics shown in figure C.15.

C.5 Implementation of Controllers

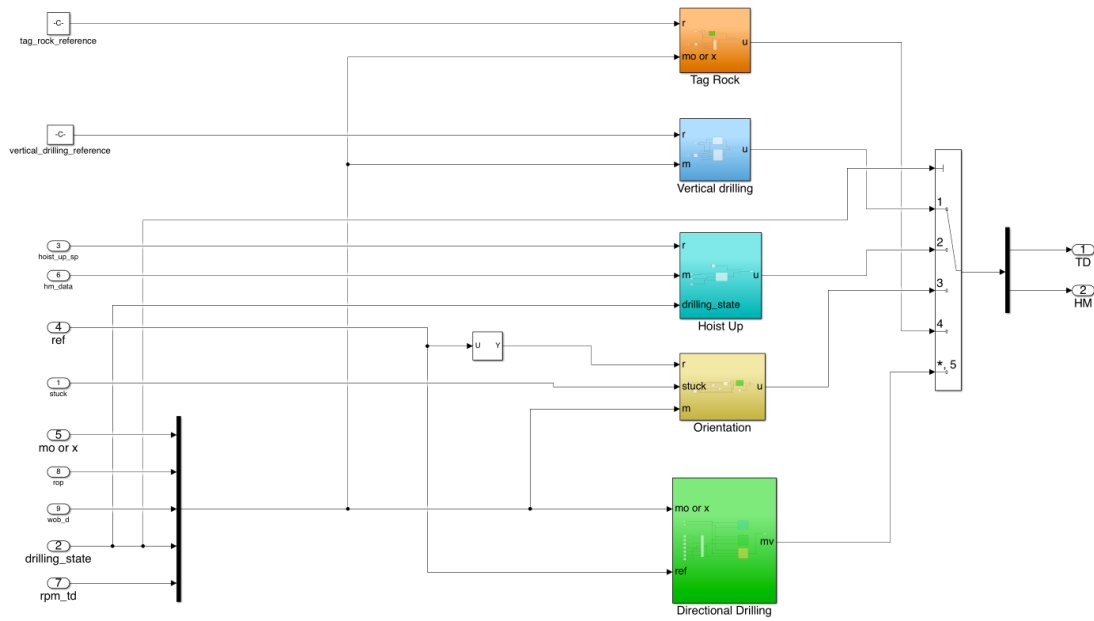


Figure C.17: The controller subsystem shown in figure C.7 contains this controller scheme. The controllers are separated to easily show which controller is active at each drilling state.

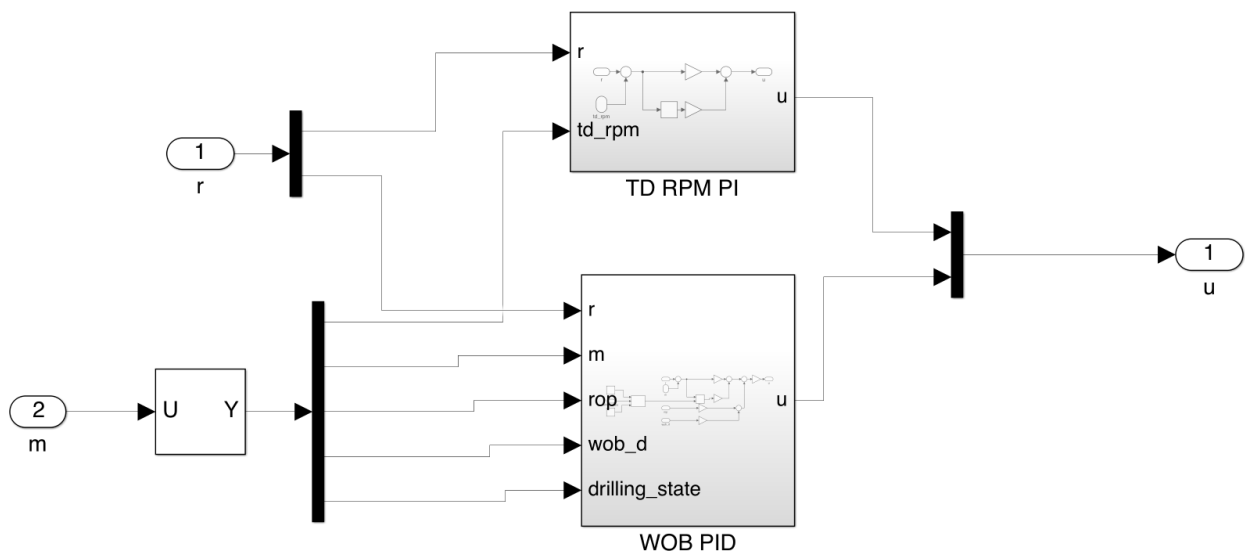


Figure C.18: The vertical drilling subsystem in the controller scheme shown in figure C.17 contains a PI controller for the Top Drive (TD) RPM and a PID controller for WOB.

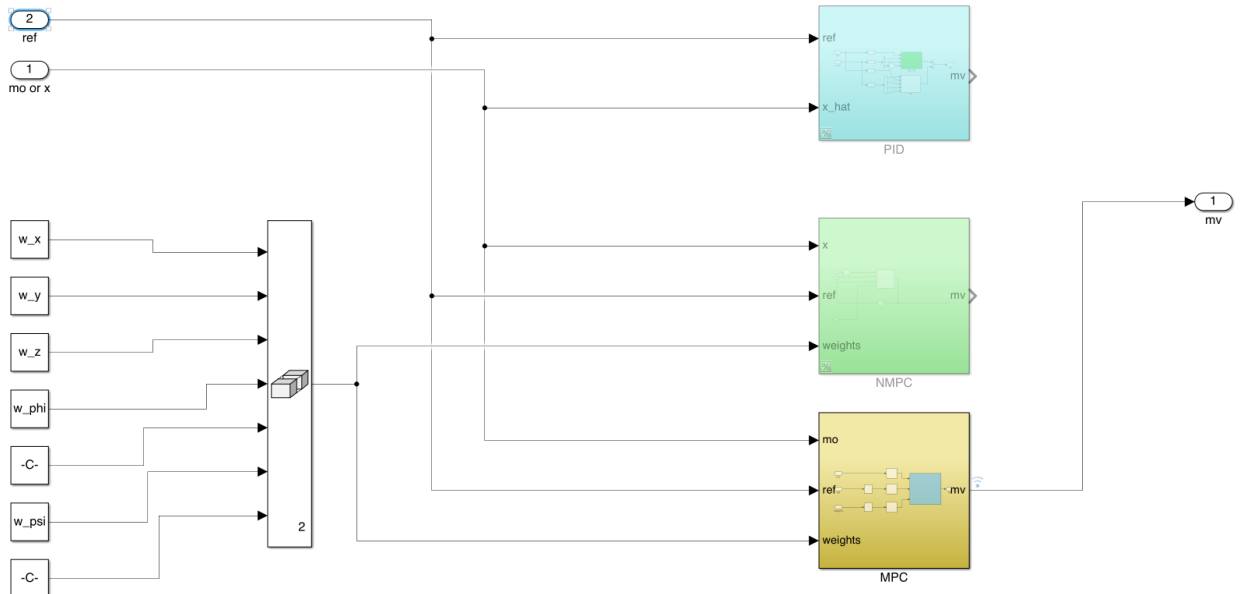


Figure C.19: Different controller options for the directional drilling state.

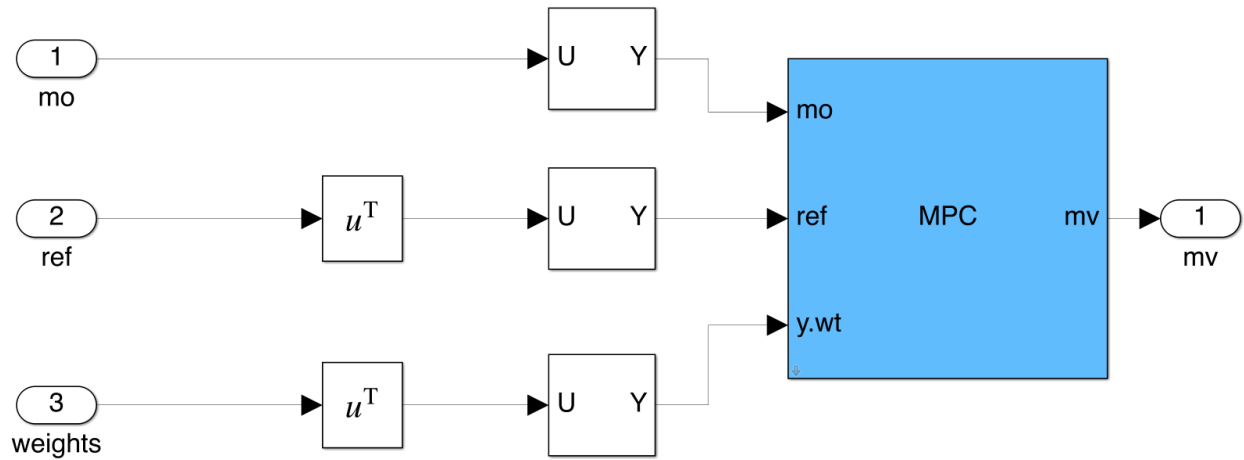


Figure C.20: Model Predictive Control (MPC) controls orientation and WOB during the directional drilling state.

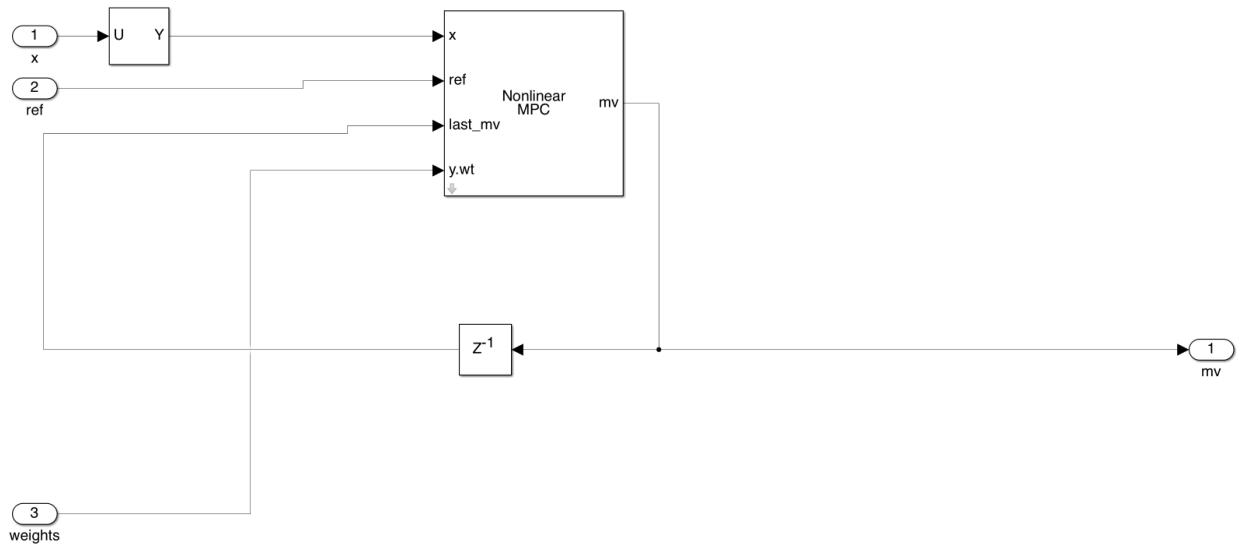


Figure C.21: Nonlinear Model Predictive Controller (NMPC) controls orientation and WOB during the directional drilling state.

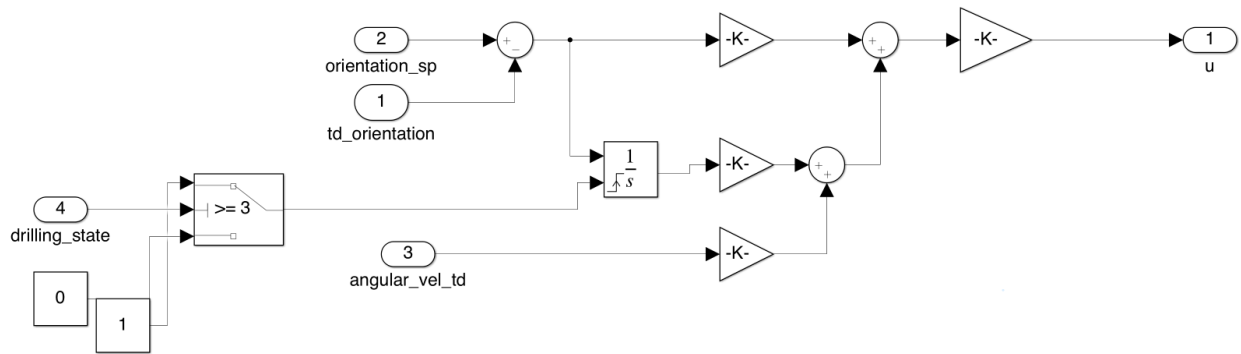


Figure C.22: Implementation of orientation controller, used in orientation and directional drilling with the PID option. The switching value for clearing the integrator will be different depending on the drilling state.

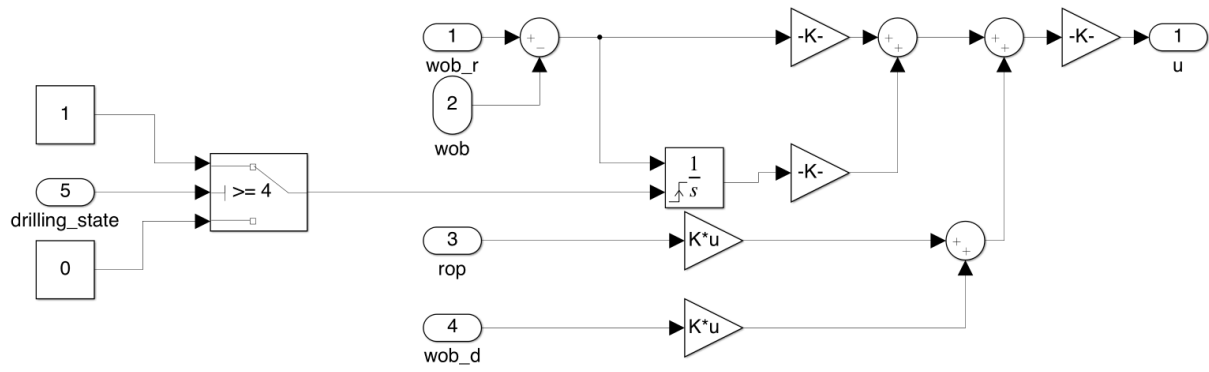


Figure C.23: Implementation of WOB controller, used in tag rock, vertical drilling, and Directinal drilling when the PID option is chosen. The switching value for clearing the integrator will be different depending on the drilling state.

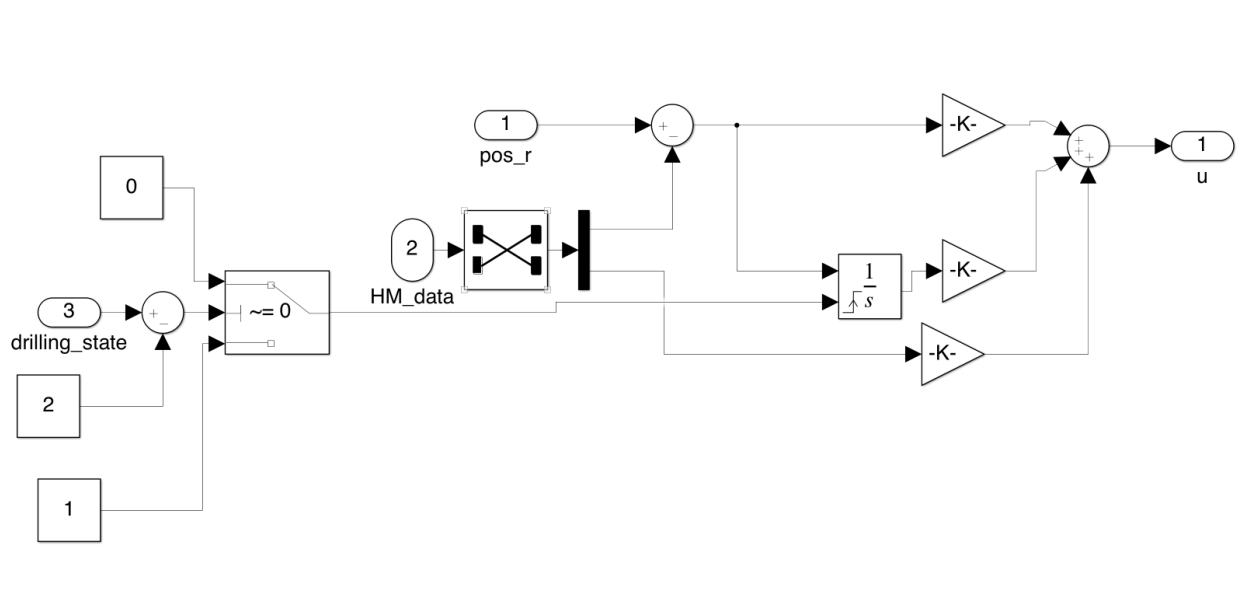


Figure C.24: Implementation of PID for hoisting motor position. Used in the Hoist Up state.

