Multi-Fidelity Model Predictive Control of Upstream Energy Production Processes

Ammon Nephi Eaton

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

John D. Hedengren, Chair
Morris D. Argyle
Randal W. Beard
Andrew R. Fry
Dean R. Wheeler

Department of Chemical Engineering

Brigham Young University

2017

ABSTRACT

Multi-Fidelity Model Predictive Control of Upstream Energy Production Processes

Ammon Nephi Eaton
Department of Chemical Engineering, BYU
Doctor of Philosophy

Increasing worldwide demand for petroleum motivates greater efficiency, safety, and environmental responsibility in upstream oil and gas processes. The objective of this research is to improve these areas with advanced control methods. This work develops the integration of optimal control methods including model predictive control, moving horizon estimation, high fidelity simulators, and switched control techniques applied to subsea riser slugging and managed pressure drilling. A subsea riser slugging model predictive controller eliminates persistent offset and decreases settling time by 5% compared to a traditional PID controller. A sensitivity analysis shows the effect of riser base pressure sensor location on controller response. A review of current crude oil pipeline wax deposition prevention, monitoring, and remediation techniques is given. Also, industrially relevant control model parameter estimation techniques are reviewed and heuristics are developed for gain and time constant estimates for single input/single output systems. The analysis indicates that overestimated controller gain and underestimated controller time constant leads to better controller performance under model parameter uncertainty. An online method for giving statistical significance to control model parameter estimates is presented. Additionally, basic and advanced switched model predictive control schemes are presented. Both algorithms use control models of varying fidelity: a high fidelity process model, a reduced order nonlinear model, and a linear empirical model. The basic switched structure introduces a method for bumpless switching between control models in a predetermined switching order. The advanced switched controller builds on the basic controller; however, instead of a predetermined switching sequence, the advanced algorithm uses the linear empirical controller when possible. When controller performance becomes unacceptable, the algorithm implements the low order model to control the process while the high fidelity model generates simulated data which is used to estimate the empirical model parameters. Once this online model identification process is complete, the controller reinstates the empirical model to control the process. This control framework allows the more accurate, yet computationally expensive, predictive capabilities of the high fidelity simulator to be incorporated into the locally accurate linear empirical model while still maintaining convergence guarantees.

Keywords: model predictive control, moving horizon estimation, advanced process control, switched control, high fidelity simulators, subsea riser slugging, pipeline wax depostion, parameter estimation, managed pressure drilling

ACKNOWLEDGMENTS

I would like to acknowledge the financial and technical assistance of the organizations that assisted in this work: SINTEF Petroleum Research, Huisman Well Technology, National Oilwell Varco (NOV), Astro Technology, the Research Council of Norway, the Utah Science Technology and Research initiative (USTAR), and Brigham Young University.

This work would not be possible without the incredible selflessness of Dr. John Hedengren and Dr. Morris Argyle. They have always put the interests of their students before their own, and I am forever indebted to their kindness and patience. I would also like to express my appreciation for the students who significantly assisted in the coding of this research, specifically, Logan Beal, Sam Thorpe, Casey Hubbell, and Kristy Moffat. I relied heavily on them to complete this work, and I learned significant things from each of them.

Finally, it must be made clear that nothing in this work would have been achieved without the constant encouragement and sacrifice of my cherished wife, Holly. She gave me hope when I was discouraged, healthy food when I was hungry, and was the motivation to faithfully finish this work. Her sacrifices to provide for and nurture our family for the last $9\frac{1}{2}$ years are beyond heroic. Without her constancy and unfailing support, this research would not have been accomplished.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Greek

α	Filter factor for additive bias
α_c	Confidence level, $1 - \alpha_c$ is the level used in nonlinear regression
α_i	ARX model output coefficients
α_o	Given initial condition for the Runge-Kutta method
α_L	Average fraction of liquid in the riser
α_{LT}	Liquid fraction of fluid immediately upstream of the riser valve
α^*_{LT}	α_{LT} without entrainment
β_a	Bulk modulus of the annulus
β_d	Bulk modulus of the drillstring
β_i	ARX model input coefficients
$\tilde{\delta}$	Kalman innovation
Δp	Change in model parameters
ΔP_v	Differential pressure across a valve
λ_{eq}	Lagrange multiplier of equality constraints
λ_{ineq}	Lagrange multiplier of inequality constraints
v_{G1}	Velocity of gas a the low point of the riser
θ	Process time delay
θ_m	Measurement time delay in slugging model
ϑ	Vector of estimated model parameters
ϑ^*	Vector of best estimates of model parameters
ρ_a	Density in the annulus
ρ_d	Density in the drillstring
ρ_{G1}	Density of gas in the section upstream of the riser
ρ_L	Density of the liquid in the slugging model
ρ_T	Average density of the total fluid flowing through the riser valve
σ_q	Standard deviation of state noise
σ_r	Standard deviation of measurement noise
τ_c	Desired CV time constant
τ_p	Process time constant
τ_I	Integral time constant for IDF
Φ	Objective function value

Latin

A	State transition matrix
AD	Automatic Differentiation
APC	Advanced Process Control
APOPT	Advanced Process OPTimizer
ARX	AutoRegressive eXogenous
$\hat{A}$	Cross-sectional area of the flowing gas at the riser base
b	Additive model bias
b_i	FIR model coefficients
B	Control matrix

BHA	Bottom Hole Assembly
c	Vector of constants
c_u	Cost vector to weight model inputs in the ℓ_1-norm objective function
c_y	Cost vector to weight changes in model inputs in the ℓ_1-norm objective function
c_L	Slack variables to penalize model value changes below the prior value
c_U	Slack variables to penalize model value changes above the prior value
$c_{\Delta p}$	Cost weight changes from the previous solution
$c_{\Delta u}$	Cost vector to weight changes in model inputs in the ℓ_1-norm objective function
C	Observation matrix
CAT	Computer Aided Tomography
CFD	Computational Fluid Dynamics
CV	Controlled Variable
d	Vector of unmeasured disturbance variables
d_b	Size of the dead-band in the ℓ_1-norm objective function
d_d	Vector of discontinuous unmeasured disturbance variables
$\hat{d}$	Vector of prior unmeasured disturbance variables
D	Riser diameter
$DAEs$	Differential and Algebraic Equations
$DRTO$	Dynamic Real Time Optimization
DTS	Distributed Temperature Sensing
DV	Disturbance Variable
D_d	Feedforward matrix
D_v	Jacobian of the model variables with respect to other model variables
D_V	Jacobian of the model equations with respect to model variables
e_L	Slack variables to penalize model values below the measurement dead-dand
e_U	Slack variables to penalize model values above the measurement dead-dand
E	Error between measurements and model predictions
EKF	Extended Kalman Filter
f	Vector of model equation residuals
f_a	Friction coefficient in the annulus
f_d	Friction coefficient in the drillstring
$f(l)$	Valve lift function
F	The F-statistic
FBG	Fiber Bragg Grating
FIR	Finite Impulse Response
$FOPDT$	First Order Plus Dead-Time
g	Inequality constraint functions
g_c	Gravitational constant
g_s	Specific gravity
h	Equality constraint functions
h_1	Actual liquid level upstream of the of the riser
h_{bit}	Well depth in low order drilling model
H	Hessian matrix
H_1	Critical liquid level at the low point of the riser

H_2	Height of the riser
i	Index variable
I	Integral term in PID and IDF
IDF	Implicit Dynamic Feedback
IMC	Internal Model Control
$IPOPT$	Interior Point OPTimizer
I_d	Identity matrix
J	Jacobian matrix
JCR	Joint Confidence Region
k	Sampling time index
k_{1-4}	Intermediate solutions for a forth order Runge-Kutta method
K	Kalman gain
KKT	Karush-Kuhn-Tucker conditions
K_1	Valve constant relating position to flow
K_2	Multiplicative factor that adjusts the magnitude of the gas flow through the riser base
K_c	Controller gain
K_p	Process gain
l	Linear valve lift term
L	Lagrange Function
LP	Linear Programming
m_{G1}	Gas mass upstream of the riser
m_{G2}	Gas mass in the riser
m_L	Liquid mass holdup in the riser
M	Effective density per unit length
MD	Measured Depth
MHE	Moving Horizon Estimation
$MILP$	Mixed Integer Linear Programming
$MINLP$	Mixed Integer NonLinear Programming
$MISO$	Multiple Input, Single Output
$MMPC$	Multiple Models Predictive Control
MPC	Model Predictive Control
MPD	Managed Pressure Drilling
MV	Manipulated Variable
M_a	M referring to the annulus
M_d	M referring to the drillstring
n	Tuning constant that adjusts the slope of the transition between full and no entrainment
N	Number of data points used for joint confidence region calculation
NLP	NonLinear Programming
$NMPC$	Nonlinear Model Predictive Control
ODE	Ordinary Differential Equation
p	Vector of model parameters
PDC	Polycrystalline Diamond Compact
PDE	Partial Differential Equation
PI	Proportional, Integral controller
PID	Proportional, Integral, Derivative controller

P_0	Pressure at the surface
P_1	Pressure in the section upstream of the riser
P_2	Pressure in the riser
P_{bit}	Pressure at the drill bit
P_c	Drilling choke valve pressure
P_k	Kalman error covariance matrix
P_p	Main mud pump pressure
P_{ref}	Reference pressure used in the slugging MPC control model
$\bar{P}$	Kalman predicted error covariance matrix
q	Parameter describing the transition between full and no entrainment in slugging model
q_{back}	Flow rate from the back pressure pump
q_{bit}	Flow rate through the drill bit
q_{choke}	Flow rate through choke valve
q_{pump}	Flow rate from the main mud pump
q_{res}	Gas flow rate from the reservoir
Q	Estimated process error covariance
QP	Quadratic Programming
Q_d	Weighting matrix on changes of the disturbance variables
Q_y	Inverse of the measurement error covariance
R	Estimated measurement error covariance
ROP	Rate of Penetration
ROV	Remotely Operated Vehicle
RPM	Rotations Per Minute of the drillstring
RTO	Real Time Optimization
s	Step size for Runge-Kutta methods
S	Kalman innovation covariance
$SISO$	Single Input, Single Output
$SMPC$	Switched Model Predictive Control
SP	Set Point
SP_{hi}	Upper limit of the dead-band region of the ℓ_1-norm objective function
SP_{lo}	Lower limit of the dead-band region of the ℓ_1-norm objective function
SQP	Sequential Quadratic Programming
SS	State Space
SV	State Variable
t	time
t_o	Given initial time for the Runge-Kutta method
T	Unspecified future time
TVD	True Vertical Depth
u	Vector of system inputs
U	Number of parameters in a model used for joint confidence region calculation
UKF	Unscented Kalman Filter
V_a	Volume of the annulus
V_d	Volume of the drillstring
w_{hi}	Weighting matrix for solutions above the ℓ_1-norm deadband
w_{lo}	Weighting matrix for solutions below the ℓ_1-norm deadband

w_m	Vector of weights on the model values outside a measurement dead-band
w_p	Vector of weights to penalize deviation from the prior solution
w_{out}	Total mass flow rate exiting the riser valve
w_{G1}	Mass flow rate of gas upstream of the riser
$w_{G,in}$	Mass flow rate of gas entering the riser
$w_{G,out}$	Mass flow rate of gas exiting the riser
$w_{L,in}$	Mass flow rate of liquid entering the riser
$w_{L,out}$	Mass flow rate of liquid exiting the riser
WAT	Wax Appearance Temperature
WDP	Wired Drill Pipe
W_{in}	Total mass flow of the riser system
x	Vector of system states
x_0	Vector of system initial states
$\bar{x}$	Kalman predicted states
$\dot{x}$	Derivative of the vector of system states
y	Vector of model predicted system outputs
y_d	Vector of discontinuous system outputs
$y_{t,hi}$	Upper limit of desired trajectory in the ℓ_1-norm objective function
$y_{t,lo}$	Lower limit of desired trajectory in the ℓ_1-norm objective function
$\hat{y}$	Vector of prior model values
z	Vector of measured system states
z_c	Valve percent open
z_{ref}	Reference valve position used in the slugging MPC control model

CHAPTER 1. INTRODUCTION

Petroleum currently fulfills 33% of total global energy demand, more than any other source of energy, and is likely to continue to do so for at least the next 30 years [1]. Increasing demand for oil brings with it increasing environmental and safety concerns. This is demonstrated by the fact that within the Gulf of Mexico alone there continue to be four uncontrolled oil or gas well situations each year [4]. Environmental and safety hazards have motivated the use of renewable energy sources. While the need for alternative sources of energy is clear, the growth and maturity of other energy sources has been slow and unable to meet energy demand, especially in transportation. Therefore, to meet the growing demand for energy, safety, and environmental responsibility, more efficient, robust, and reliable technological advances are needed in the petroleum industry. To assist in this effort, this dissertation presents several advanced control techniques that are applied to processes in the petroleum industry.

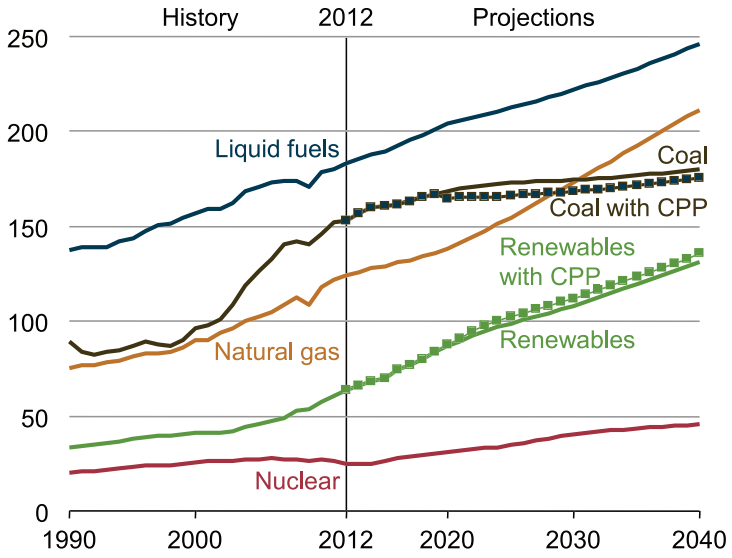


Figure 1.1: Total world energy consumption by energy source, 1990-2040 (quadrillion BTU). Note: Dotted lines for coal and renewables show projected efforts of the U.S. Clean Power Plan. Source: U.S. Energy Information Administration [1]

The petroleum industry is functionally divided into upstream and downstream divisions. The upstream division finds and extracts oil and gas from geologic formations, while the downstream division refines the crude oil and gas into usable products. The downstream sector has seen many technological advances in process control and optimization, but many processes in the upstream sector, such as oil well drilling, still lack any significant automation [5]. When automation is optimized, it improves safety and efficiency over manual control by responding faster to process disturbances and by operating closer to process constraints. To attain these benefits, optimal automation and control techniques require a sufficiently accurate model of the process. These models can be obtained from empirical data or from first-principles. First-principles-based models that very nearly approximate the actual process are known as high fidelity simulators. High fidelity simulators have exceptionally accurate predictions, which is highly desirable for predictive control applications. These models have rarely been implemented in real time control because of the large computational resources required to use them in the short time intervals necessary in feedback control. This research explores novel uses of high fidelity simulators for optimal control in upstream oil and gas processes.

1.1 Overview

Background information on advanced control and estimation techniques, and upstream oil and gas processes puts this research in context. Subsection 1.1.1 explains how and why the APMonitor modeling language modifies equations for simulation, estimation, and control. Subsection 1.1.2 describes subsea riser slugging and associated control challenges, as well as reviews pipeline wax deposition technologies. Subsection 1.1.3 introduces concepts in advanced predictive control, while subsection 1.1.4 discusses the benefits and draw backs of high fidelity simulators in control applications. Subsection 1.1.5 describes the oil well drilling process and the necessity of downhole pressure control. Finally, Subsection 1.1.6 covers the need for estimating control model parameters along with several industrially relevant estimation techniques. Additionally, the significance of the novel contributions made in this work are highlighted in each subsection. These subsections lay the foundation for the central theme of this research, which is improving controller reliability through Model Predictive Control (MPC) and multi-fidelity control models. The work includes theoretical and novel application contributions in model predictive and switched control

for upstream processes. Therefore, a review of state-of-the-art methods is presented along with the challenges related to controlling the production of hydrocarbons.

1.1.1 The APMonitor Modeling Language

Dr. John Hedengren began developing the APMonitor modeling language in 2003, and is still actively developing it [6]. Modeling languages are different than general use programming languages, such as Python or C++. A modeling language converts dynamic process control models into a form that allows an optimizer or solver to perform gradient-based optimization with the model equations. To convert the equations, APMonitor incorporates several recent developments in numerical methods and optimization techniques. These methods are explained in detail in this subsection because APMonitor is used extensively in this work.

Dynamic Process Control Models

Optimal, computer-aided, process control requires a mathematical description of the process. The collective equations are known as a control model, and can have various forms. Control models are grouped into two major divisions: empirical-based models and first-principles-based models. Empirical-based models are also known as "black box" models because they relate model inputs to model outputs without giving *a priori* insight into how the inputs and outputs are related. Several common empirical control model forms are in Table 1.1. In this dissertation, t is time, k is discrete time, i is an index variable, y is a time-varying output variable, x is a time-varying state variable, and u is a time varying input variable. τ_p is the process time constant, K_p is the process gain, and θ is the process time delay. The A, B, C, and D_d matrices are the state transition matrix, the control matrix, the observation matrix, and the feedforward matrix respectively. b_i are the FIR model coefficients, and α_i and β_i are the ARX model output and input coefficients.

Although nonlinear empirical models have been developed, empirical models often have a linear relationship between the model inputs and outputs. Linear control models used on nonlinear processes have a limited range of accuracy. This limitation has been addressed in several ways. Gain-scheduling [7] is a technique that switches among predetermined linear control models based on the current operating region. Methods based on Hammerstein-Weiner models [8] include a

Table 1.1: Common empirical control model forms

Name	Equation
First Order Plus Dead Time (FOPDT)	$\tau_p \frac{dy(t)}{dt} = -y(t) + K_p u(t-\theta)$ (1.2)
State Space (SS)	$\frac{dx(t)}{dt} = Ax(t) + Bu(t)$ (1.3a)
	$y(t) = Cx(t) + D_d u(t)$ (1.3b)
Finite Impulse Response (FIR)	$y(k) = \sum_{k=0}^{N} b_i u(k-i)$ (1.4)
AutoRegressive with eXogenous inputs (ARX)	$y(k) = \sum_{k=0}^{N} \alpha_i y(k-i) + \sum_{k=0}^{N} \beta_i u(k-i)$ (1.5)

term to the linear model that accounts for the nonlinear behavior of the system. It is also possible to simply use the linear model over the full operation range, but the controller may be limited in its ability to control the process. Empirical control models are common in controlling industrial processes [9]. These models are identified from dynamic process data that are used to fit parameters in the model. Generating data for empirical model parameter regression can be disruptive to operations and very costly. Many industrial processes, such as polymer grade transitions, are extremely nonlinear, to the extent that linear model approximations are insufficient to control the process [10]. In these situations, nonlinear first-principles models are valuable for control.

First-principles-based models come from fundamental energy, mass, and momentum balances and include equations that capture the underlying chemistry and physics of a process. These models explicitly describe the relationships among process variables, which are principally nonlinear in nature. Hence, first-principles-based models are characterized by dynamic nonlinear equa-

tions [11]. first-principles models are also inherently more complex than empirical models. Dynamic first-principles-based models generally consist of four elements:

1. Core differential equations that capture the major dynamics of the process, and may include mass, momentum, or energy balances

2. Ancillary algebraic equations that determine necessary variables within the core differential equations such as chemical kinetic rate constants or the cross-sectional area of a pipe.

3. Equation variables that change as process conditions change such as pressure, flow rates, or valve position.

4. Equation variables that do not change with changing process conditions such as fundamental physical constants or pipe diameter

Examples of first-principles models used in this work are in Chapters 2, 4, and 5. Few industrial processes use first-principles-based models when compared to empirical models. They take considerably more time to develop and to calibrate to changing process conditions. Also, the computational resources required to use them in optimization can be substantial. The advantage is that they are generally more accurate over a wider range of operating conditions, thus requiring significantly less tuning than empirical models.

If the control model contains any Partial Differential Equations (PDEs), they must be converted to Ordinary Differential Equations (ODEs) or Differential and Algebraic Equations (DAEs) to be used in APMonitor. Once a suitable control model is acquired, any differential equations must be solved at each instance of time.

Solving the Differential Equations

The model equations are the foundation of a controller. To optimize control, there are two major divisions in the methods used to numerically evaluate ODEs- simultaneous methods and sequential methods. Sequential methods solve the model equations from the given initial conditions by stepping sequentially forward in time. The model ODEs are solved at each time step by using Runge-Kutta [12] or other similar numerical integration techniques. A fourth-order Runge-Kutta

method evaluates the equations at the given initial conditions and three other intermediate steps to give the solution at the next time step. Equations 1.6a-1.6g show how this is done in general for a step size of s, and a known t_o and α_o.

$$\frac{dy(t)}{dt} = f(t,y), \quad y(t_o) = \alpha_o \tag{1.6a}$$

$$k_1 = sf(t_i, \alpha_i) \tag{1.6b}$$

$$k_2 = sf(t_i + \frac{s}{2}, \alpha_i + \frac{k_1}{2}) \tag{1.6c}$$

$$k_3 = sf(t_i + \frac{s}{2}, \alpha_i + \frac{k_2}{2}) \tag{1.6d}$$

$$k_4 = sf(t_i + s, \alpha_i + k_3) \tag{1.6e}$$

$$\alpha_{i+1} = \alpha_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{1.6f}$$

$$\alpha_{i+1} \approx y(t_{i+1}) \tag{1.6g}$$

This process is repeated at each time step, and the solution is sequentially propagated forward in time. Simplicity is one of the benefits of sequential methods; one of the limitations of sequential methods in control applications is the number of function calls, and subsequent computation time, necessary at each time step. Most of the required computation is spent on converging intermediate solutions that are not optimal. In contrast, simultaneous solution methods give a computationally efficient way of calculating the entire time horizon of interest. These methods converge the equations and minimize the objective function simultaneously.

Simultaneous methods can also be used to solve the core model equations. APMonitor uses a method developed by Carey and Finlayson in 1974 [13] called orthogonal collocation on finite elements. This method solves the ODEs by approximating the continuous solution at predeter-

mined points known as collocation points. These points are determined by the Lobatto quadrature [14], [15]. A quadrature is a formula for numerically approximating the integral of a function with a series of algebraic functions and weights at points where the solution exactly matches the continuous solution [16]. The Lobatto quadrature uses the roots of the Legendre polynomials to find the collocation points which are orthogonal to each other. The orthogonality is essential because it ensures the points are linearly independent, and the matrix used to find the weights in the Lobatto quadrature will not be singular. Once the weighting terms are determined, each time step in the control horizon is a boundary collocation point, and additional intermediate points can be chosen and computed. The time step is also known as a finite element [16]. Figure 1.2 gives a graphical representation of orthogonal collocation on finite elements in a control application.

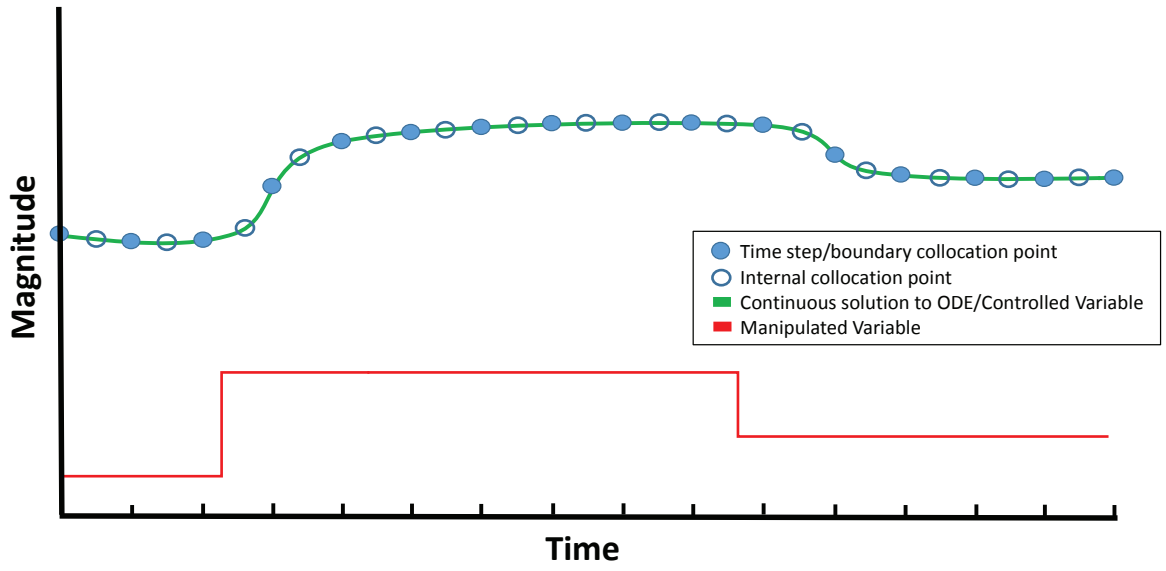


Figure 1.2: A graphical representation of orthogonal collocation on finite elements in a generic control application. This example uses a Legendre polynomial of degree two with one internal collocation point.

Simultaneous solution methods are beneficial for several reasons. These methods require less function calls per time step, which translates to less computational resources and faster solution times. Once the ODEs are converted to purely algebraic equations, additional algebraic constraint

Table 1.2: A summary of the discussed benefits and drawbacks of sequential and simultaneous solution methods in optimal control applications.

	Sequential	**Simultaneous**
Benefits	• Straightforward to implement	• Fast solution times
	• Solver always remains in feasible solution space	• Easy to implement equation and variable constraints
	• Easy to implement logical conditions in optimization routines	• Easy to incorporate into gradient-based optimization solvers
Drawbacks	• Requires more computation per time step	• Difficult to directly incorporate logical decisions
	• Difficult to implement variable constraints	• Possibility of not converging to a solution

equations and constraints on system and state variables can be directly integrated into the algorithm of a gradient-based solver for more efficient optimization in MPC applications.

As a flexible modeling language, APMonitor has options for both sequential and simultaneous solution methods. Each method has strengths and weaknesses, and each should be used with discretion. For example, when solving highly nonlinear model equations simultaneous methods may give poor algebraic approximations, and sequential methods may prove more reliable. Table 1.2 contrasts the benefits and drawbacks of sequential and simultaneous methods.

Solving the model differential equations is necessary for control, but to take advantage of current optimization techniques the first and second derivatives of the model equations must also be computed.

Differentiation Methods

APMonitor uses Automatic Differentiation (AD) to compute the first and second derivatives of the model equations, objective function, and constraint equations. These derivatives are used in the optimization algorithms as explained in the next subsection. AD applies the chain rule for calculating derivatives to computer code in a systematic way. There are two methods for using

AD to compute derivatives: forward and reverse. Equation 1.7 shows how the two methods are related.

$$(I_d - D_V)D_v = I_d = (I_d - D_V^T)D_v^T \tag{1.7}$$

Here, D_V is the Jacobian of the model equations with respect to the model variables, D_v is the Jacobian of each model variable with respect to the other model variables, I_d is the identity matrix, and the superscript T indicates the matrix transpose. The forward method solves the left hand side of Equation 1.7 for D_v, while the reverse method solves the right hand side for D_v^T. If the number of model equations is greater than the number of model variables (i.e. an optimization problem), then the reverse method is more computationally efficient. When there are more model variables than model equations (i.e. an estimation problem), then the forward method uses less computation [17]. Because the forward method uses less active memory than the reverse methods, APMonitor employs the forward method for both optimization and estimation problems. It does this in practice by using a technique called operator overloading. Operator overloading can only be done in a programming language that supports this feature such as Fortran 90 or C++. A new data structure is defined that includes a variable and its derivative. Then the basic computation operators are redefined to handle the variable and its derivative [17].

Once the derivatives of the system can be computed, and differential equations are converted to algebraic equations, APMonitor employs an active set or an interior point solver to find the optimal solution for simulation, estimation or control problems.

Constrained Gradient-Based Optimization and Nonlinear Solvers

APMonitor has the capability to solve several types of optimization problems. These problems can include equality or inequality constraints, and can be linear or nonlinear. The general formulation of these problems is Equation 1.8.

$$\begin{aligned} \text{minimize}: \quad & f(x) \\ subject\ to: \quad & g(x) \leq c, \\ & h(x) = c \end{aligned} \tag{1.8}$$

In this general formulation, f is the objective function, x is a vector of the model manipulated variables, g is the set of inequality constraint equations, c is a vector of constants, and h is the set of equality constraint equations. If the model equations are such that f is nonlinear, then the optimization problem is called a NonLinear Programming (NLP) problem . If certain variables in an NLP problem are required to be binary or integers, the problem type is known as a Mixed Integer NonLinear Programming (MINLP) problem . If the objective function is linear then it is termed a Linear Programming (LP) problem, with binary or integer variables it is called Mixed Integer Linear Programming (MILP) problem. If the objective function f is quadratic, then the problem is known as a Quadratic Programing (QP) problem. Because constrained NLP problems are the most challenging for gradient-based solvers, this type of problem is discussed in more detail. These types of problems are usually solved using either an established open source interior point solver called Interior Point OPTimizer or IPOPT [18], or an active set solver known as Advanced Process OPTimizer or APOPT [19]. Both IPOPT and APOPT use the method of Lagrange multipliers which employs the Lagrangian L as the objective function to be minimized. This method is an efficient way of incorporating the constraint equations into the optimization algorithm. The Lagrangian function is found in Equation 1.9.

$$L(x, \lambda_{eq}, \lambda_{ineq}) = f(x) + \lambda_{eq}(h(x) - c) + \lambda_{ineq}(g(x) - c) \tag{1.9}$$

Here, λ_{eq} and λ_{ineq} are weighting vectors, known as Lagrange multipliers, that are chosen to make the gradients of the equality and inequality constraints and the gradient of the objective function f sum to zero. This condition is part of the Karush-Kuhn-Tucker (KKT) conditions that define optimality. The KKT conditions are summarized in Table 1.3.

Table 1.3: A summary of the necessary and sufficient conditions for constrained optimality. These are known as the KKT conditions.

	Condition	Equation
Necessary	No search direction improves the objective function	$J = \nabla f(x) = \lambda_{eq} \nabla h(x) + \lambda_{ineq} \nabla g(x)$
	The solution is feasible	$h(x) - c = 0, \quad g(x) - c \leq 0$
	λ_{eq} is unconstrained and λ_{ineq} is nonnegative	$-\infty < \lambda_{eq} < \infty \ , \quad \lambda_{ineq} \geq 0$
Sufficient	The Hessian of the Lagrangian with respect to x is positive definite	$H = \nabla^2 L(x, \lambda_{eq}, \lambda_{ineq}) = \nabla^2 f(x) + \lambda_{eq} \nabla^2 (h(x) - c) + \lambda_{ineq} \nabla^2 (g(x) - c)$

Another KKT condition for constrained optima is positive definite Hessian and Jacobian matrices. The Jacobian matrix is the partial first derivatives of the model equations with respect to each variable in the equations and is shown in Equation 1.10.

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial \mathbf{f}}{\partial \mathbf{x_1}} & \cdots & \dfrac{\partial \mathbf{f}}{\partial \mathbf{x_n}} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{1.10}$$

The Hessian is the second derivative matrix of the Lagrangian as shown in Equation 1.11. When every eigenvalue of the Hessian is positive, it is call positive definite. If the second derivative with respect to each variable is positive, then the sufficient condition of optimality is satisfied.

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 L}{\partial x_1^2} & \frac{\partial^2 L}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 L}{\partial x_1 \partial x_n} \\ \frac{\partial^2 L}{\partial x_2 \partial x_1} & \frac{\partial^2 L}{\partial x_2^2} & \cdots & \frac{\partial^2 L}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 L}{\partial x_n \partial x_1} & \frac{\partial^2 L}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 L}{\partial x_n^2} \end{bmatrix} \tag{1.11}$$

Once IPOPT or APOPT verifies the KKT conditions are met, the search for optimality stops. While the optimal solution is the same for a given problem, IPOPT and APOPT arrive at

the solution in different ways. IPOPT uses an interior point method that utilizes a barrier term to force the search direction away from the constraints, and toward the interior of the solution space. At each iteration the barrier term is relaxed until the constraints are included in the search space. In contrast, APOPT uses an active set method that assumes the constraints given will be active in the optimal solution. The search direction follows the constraints until the KKT conditions are satisfied.

When solving constrained nonlinear, or nonconvex, optimization problems, there are several issues that the solver can encounter. For example, the search may lead to a local optimal point, instead of the actual global optimum. When using the simultaneous solution method, finding local minima can be over come by starting from an appropriate initial condition. Identifying the best initial conditions can be extremely challenging. However, several strategies exist that assist in finding the suitable beginning model values. Several of these strategies can be found in [20]. Figure 1.3 shows an example graphical representation of a constrained non-convex optimization problem and the pitfalls can occur in nonconvex optimization.

This work uses nonlinear solvers on nonconvex optimization problems. These methods are applied in MPC and Moving Horizon Estimation (MHE) to control upstream production processes. These processes are further described in the following sections to further put the value of this research in context.

1.1.2 Subsea Riser Slugging Control

Hydrocarbon production from offshore oil fields has many challenges. These challenges stem from drilling and producing oil from reservoirs that can be 20,000+ feet below the ocean floor in 1,500 to 12,000 feet of water. Once a geologic evaluation is completed, exploratory wells are drilled and prepared for producing oil or gas; this is known as completing a well. Well completion includes connecting the well to the reservoir, hydraulic fracturing, installing any necessary downhole equipment such as pumps and connecting the wellhead at the ocean floor to floating production facilities. Often these facilities are several miles from the wellhead, and a pipeline is placed on the sea floor as a connection. Pipelines within an oil field are known as flowlines. When the flowline approaches the floating facility it rises vertically to the floating vessel. This vertical section of the flowline is known as the production riser. Figure 1.4 shows two of the most com-

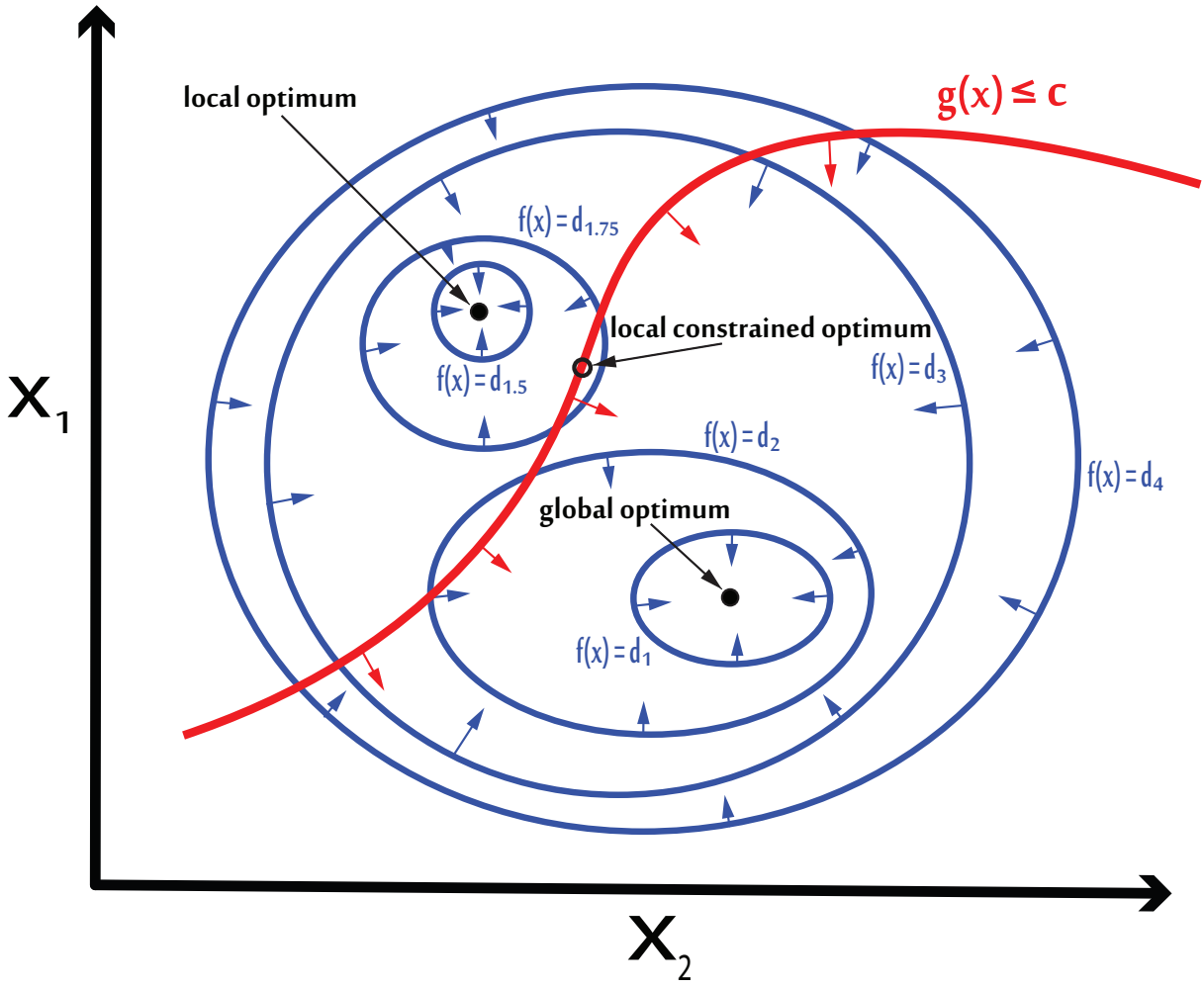


Figure 1.3: Example of a constrained non-convex optimization problem with two variables.

mon types of production riser configurations in offshore oilfields that are most likely to have flow instabilities. Flow instabilities can occur when a riser has a low point, caused by either the buoys or the sea floor terrain. A common flow instability is known as slugging. Slugging is a physical phenomenon that can occur in pipes with multiphase flow of liquid and vapor. The liquid gathers at the base of the riser and cuts off the gas from ascending in the riser.

As flow continues, liquid fills the riser volume and the gas builds pressure at the riser base. This continues until the gas pressure is sufficient to rapidly move the liquid in the riser into the topside receiving facilities. Figure 1.5 shows the stages of riser slugging.

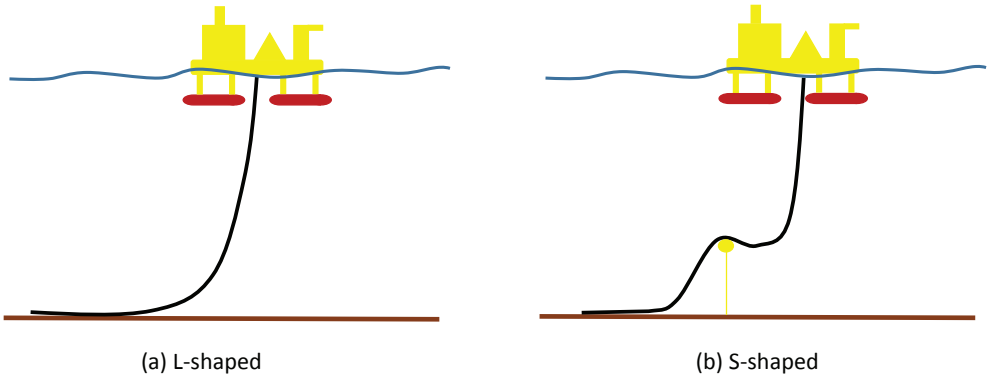


Figure 1.4: Common production riser types.

Severe slugging is large amplitudes of pressure and flow, and can occur in subsea oil well production risers. The undesired oscillations caused by severe slugging slow oil and gas production, and cause accelerated fatigue to production equipment. Many technologies have been developed to control the effects of slugging including design of separation equipment to better accommodate the slugs, a large topside holding tank to catch the slugs, and subsea phase separators that separate the liquid from the gas near the wellhead. These methods are often expensive or sub-optimal solutions [21]. Another way to mitigate the effects of severe slugging is through a choke valve at the top of the production riser. The valve can be used by a controller to dampen the oscillations caused by slugging. This inexpensive solution was first reported as successful in 1930 [22] and has since been studied extensively. Several controllers have been designed for slugging suppression including PI [23], cascaded PID [24], neural networks [25], and gain-scheduling Internal Model Control (IMC) [26] among others [27]. However, these control methods do not take advantage of predictive control techniques. One of the novel contributions of this work is the design and simulation of an advanced model predictive controller for severe subsea riser slugging mitigation with quantified benefits for direct pressure sensing at the base of the riser.

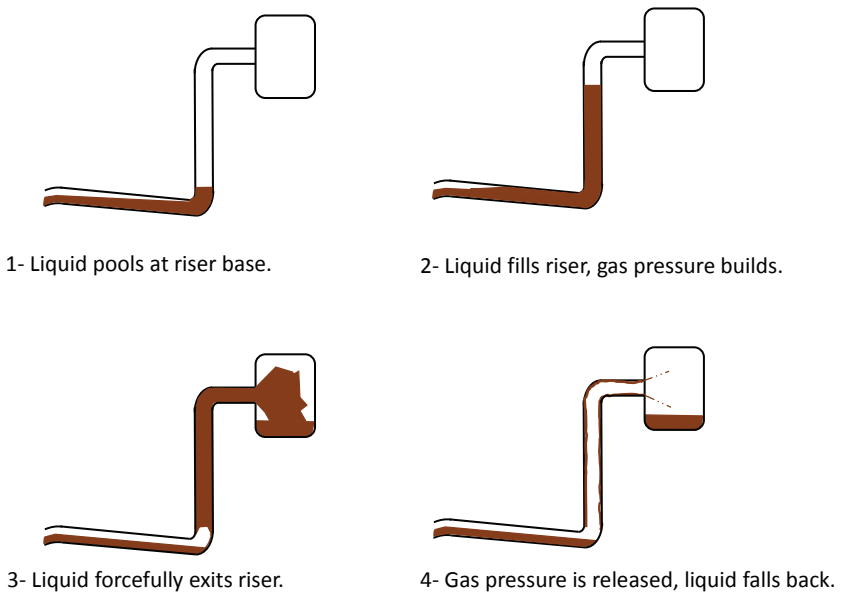


Figure 1.5: Stages of subsea riser slugging.

1.1.3 Model Predictive Control

Model predictive control (MPC) is an advanced dynamic control technique that uses optimization methods and a transient mathematical model to control a process. The algorithm minimizes the difference between the model predicted states and the process controlled variables (CVs) by adjusting the process manipulated variables (MVs) for a given set point (SP). This is done over a finite future time horizon to predict future process states and adjust for future constraint violations in the present time step (see Fig. 1.6). Only the first of the predicted changes in MVs is implemented. The latest process measurements are used to initialize subsequent optimization calculations, and the calculations are repeated at the next time step.

MPC depends on a sufficiently accurate model of the process. If an accurate model is available, then MPC offers several advantages over traditional control algorithms. The major benefits of MPC over traditional Proportional, Integral, Derivative (PID), or Proportional, Integral (PI) controllers, and the drawbacks of MPC, are shown in Table 1.4. The benefits of MPC are further enhanced when high fidelity simulators are used in the MPC algorithm.

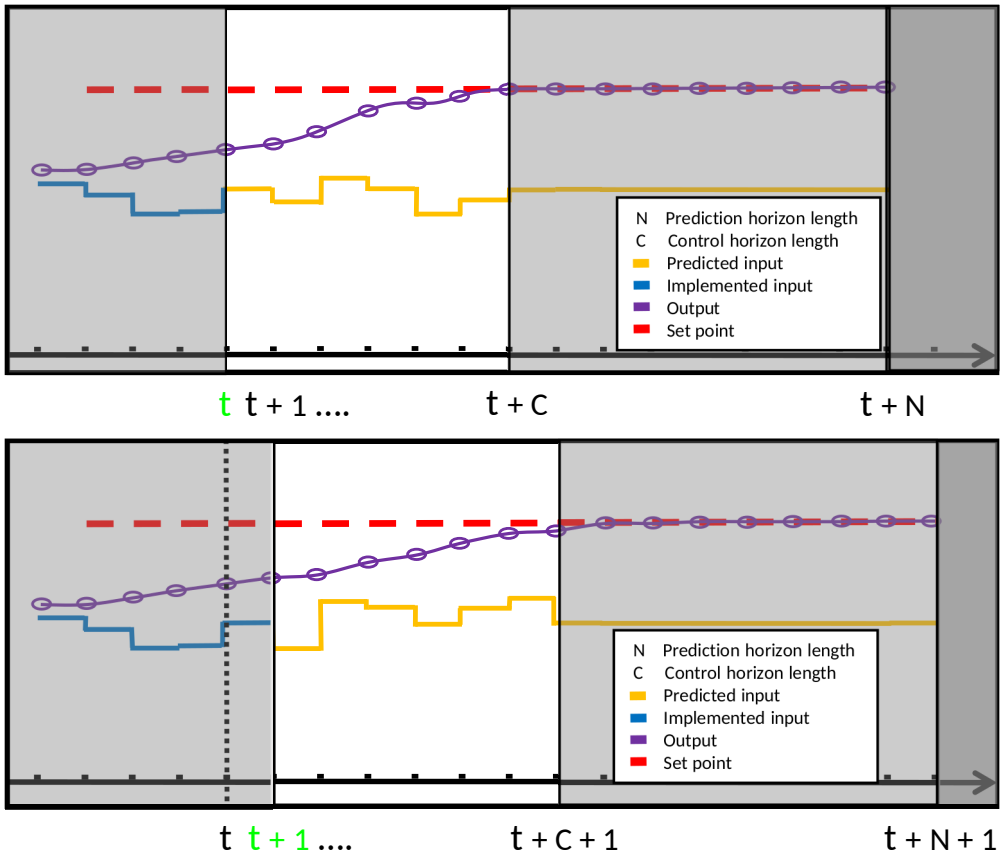


Figure 1.6: A graphical interpretation of the MPC algorithm. The optimization routine minimizes the error between the current plant outputs and the set point by changing the inputs at each time step over the prediction horizon *N*. Only the first control move is implemented, the horizon shifts, and the procedure is repeated incorporating the newest plant measurements. If highly accurate models are used, better predictions lead to better controller performance and more stable control.

1.1.4 High Fidelity Simulators in MPC

The value of the predictions from highly accurate first-principles models in real time feedback control is most apparent in MPC. Closed loop control systems can become unstable even with highly accurate models, and several robust control strategies have been developed to guarantee stability for linear MPC applications [28]–[30]. However, a control model must have a certain level of accuracy before any guarantees of controller stability and performance can be made. This

Table 1.4: General benefits and drawbacks of MPC.

Benefits	Drawbacks
• The controller takes advantage of the modeled dynamic and static interactions among MVs and CVs	• The algorithm requires more computational resources including an optimization routine at each iteration
• Process constraints are directly incorporated into the control law	• The controller is more complex with an inherent higher probability of error and maintenance issues
• Control and set point calculations can be coordinated and optimized	• There is a possibility that the algorithm will not converge to a solution at all or within the required time
• Adjusts for future constraint violations by predicting, and not just reacting to, modeled disturbances	

is particularly true in MPC because the optimization algorithm minimizes the error between the model predictions and the process measurements at each time step.

If the control model does not describe the plant behavior sufficiently, then the predictions and subsequent controller decisions can cause a loss of satisfactory control. For Nonlinear MPC (NMPC), it has been shown that if the optimizer converges to a solution, the feedback controller is guaranteed to be stable [31]. However, in order for the solver to converge within a specified tolerance of the true process, the control model must have a certain level of accuracy. It has been demonstrated that improved models provide better control than less rigorous models in optimized feedback control [32]. While performance improvements have been demonstrated, robust performance and stability guarantees in NMPC are the subject of current research [30], [33]. In addition to performance improvement, correct model predictions allow a controller to maintain control over a process, over the prediction horizon, even when there is no process feedback due to sensor failure, etc. Therefore, because predicted future states are used to control current conditions, MPC requires adequate model accuracy for any controller performance and stability guarantees. This motivates the use of high fidelity simulators in feedback control for applications in the upstream oil and gas industry.

One of the novel contributions of this work is a method to use the highly accurate, yet computationally intensive, predictions of high fidelity simulators in real-time feedback control. This novel control method is applied to managed pressure drilling with high fidelity simulators. To fully appreciate these contributions, it is necessary to understand the oil well drilling process.

1.1.5 Automated Managed Pressure Drilling

Before a well can be put into production, it must be drilled, cased, cemented, and sometimes hydraulically fractured with a number of complex steps that each have unique challenges. The focus of this section is on the drilling phase of well construction. Oil and gas wells are created by drilling for several hundred to several thousand feet, stopping to insert and cement casing pipe to the well bore, then repeating the process until the target depth in reached. As the well deepens, more drill pipe is connected to the drillstring. At the bottom of the drillstring, a Bottom Hole Assembly (BHA), consisting of measurement and steering equipment, is attached to the drill bit. The drill bit is cooled by the drilling fluid, or mud, which also moves the rock cuttings to the surface and maintains pressure in the well annulus (see Figure 1.7). In conventional drilling, the well annulus pressure must be greater than the geologic reservoir pressure to prevent hydrocarbons from entering the well during the drilling process. Excessively high mud pressure in the well can damage the rock formation, while excessively low pressure can allow hydrocarbons from the subsurface reservoir to reach the surface in an uncontrolled and dangerous manner. When mud is displaced by uncontrolled reservoir fluid flow, it is known as a blowout. The well bore pressure must be maintained within a range of pressures that balances the reservoir fluid pressure to prevent fractured formations and blowouts. To help achieve this pressure balance, a variation of traditional drilling, called Managed Pressure Drilling (MPD) was developed [34]. A simplified schematic of MPD is shown in Figure 1.7. A common implementation of MPD uses mud mass and volume balances from measurements to estimate and control the bit pressure and reject process disturbances. Another MPD strategy uses pressure measurements from the BHA to inform the driller of the need to adjust the main mud pump flow rate and choke valve opening to reach the desired pressure target in the well. Also, a back pressure pump is used to maintain well pressure during pipe connection procedures when the main mud pump is disconnected from the drillstring.

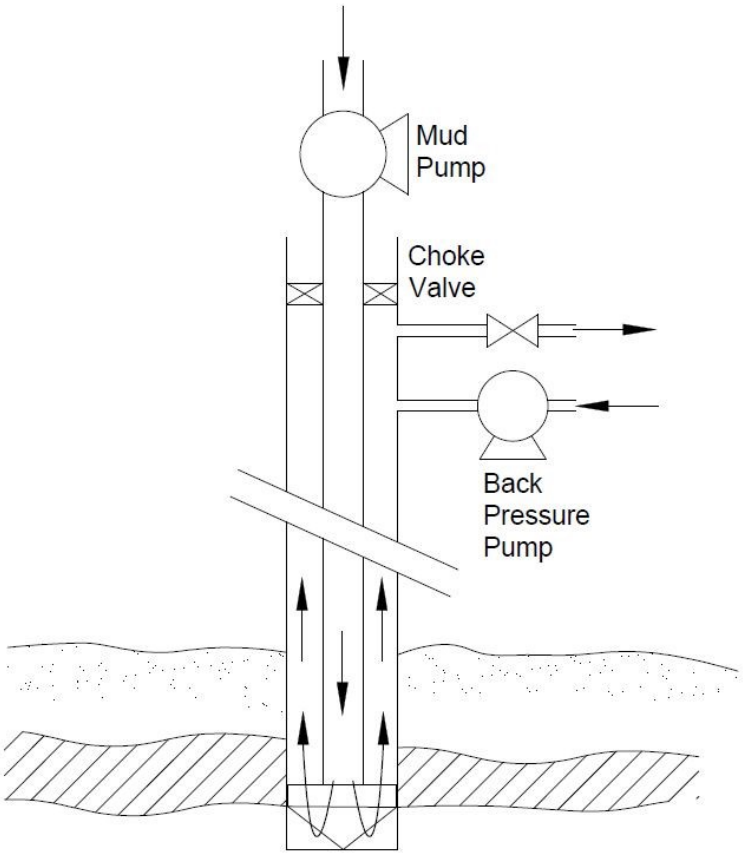


Figure 1.7: Simplified schematic of the MPD process.

Several abnormal events can disturb normal drilling operations. For example, irregular torsional vibrations in the drillstring can cause the bit to periodically stick in the well bore instead of cutting through the rock. This is known as stick/slip, and is detected by an erratic Rate of Penetration (ROP) through the rock and transient torque in the drilling rig topdrive which turns the drillstring [35]. Erratic ROP can affect bit pressure control by randomly changing the friction between the rotating drillpipe and the annulus fluid and subsequently the pressure in the annulus [36]. Additionally, a drilling disturbance known as pack-off can occur. Pack-off occurs when mud flow in the annulus is insufficient to carry the rock cuttings to the surface. The cuttings conglomerate in the mud which prevents drillstring rotation. The drillstring is unable to spin or move up or down in the well. When this occurs, pressure control ceases along with drilling operations until the issue is resolved. Another serious process disturbance occurs when drilling into an unexpected high pressure reservoir can offset the well/reservoir pressure balance and allow an unwanted influx of gas or oil into the well bore. This situation is known as unwanted gas

influx or kick. It is characterized by a sudden increase in pressure near the bit and increased flow rate in the annulus. Traditionally, a kick is addressed by stopping operations and adding bentonite powder to the mud to increase the fluid density and hydrostatic pressure of the mud column. This creates a new balance, at a higher pressure, between the annulus pressure and the reservoir pressure. However, the advances of MPD allow kicks to be rectified using increased mud flow rates and changing choke valve positions, without stopping operations.

Some of the novel contributions of this work include the use of a high fidelity simulator in the control loop to maintain bit pressure in the event of a simulated kick, and also when there is no bit pressure feedback due to no mud flow during a pipe connection procedure in drilling. The method utilizes a switching algorithm that selects the best control model based on availability and fidelity. The method is expanded to include a novel online model identification technique that incorporates high fidelity model predictions into the control loop. The method is used to control a kick simulation.

While using highly accurate models can result in greater control over a wider range of operating condition, it is critical that the model parameters are estimated correctly. Several estimation techniques exist to estimate model parameter values. These methods are reviewed and compared in this work.

1.1.6 Comparison of Model Parameter Estimation Methods

Several estimation techniques are used in the oil and gas refining, chemicals, exploration, and production sectors for process model parameter estimation. These techniques include a filtered bias update, Implicit Dynamic Feedback (IDF), Kalman Filtering (KF), and Moving Horizon Estimation (MHE) [37]. This work reviews the advantages and disadvantages of each of these methods. Because MHE is an estimation method used throughout this work, it is discussed in more detail.

MHE uses optimization to adjusts process model parameters to minimize the error between the model and process measurements over an estimation time horizon. Figure 1.8 gives a graphical representation of the MHE algorithm.The disadvantages of MHE, compared to other estimators, are the increased computational load required to solve the optimization problem and the difficulty in obtaining optimal tuning. This work discusses techniques to overcome both of these obstacles

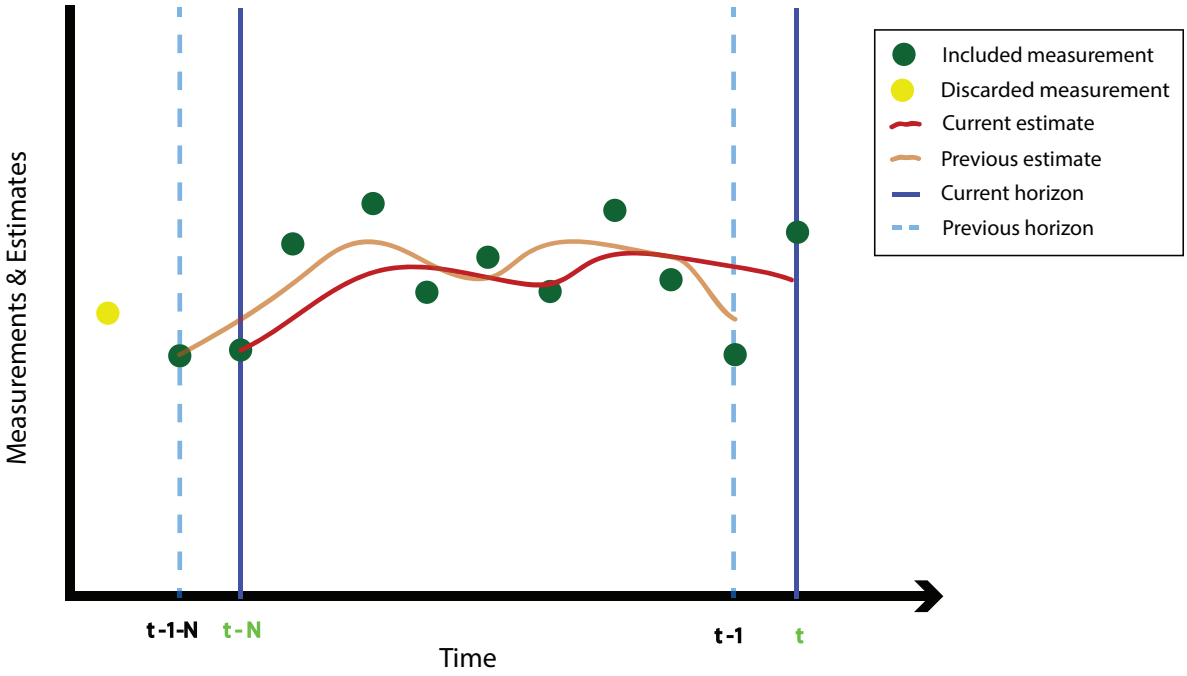


Figure 1.8: A diagram of a moving horizon estimation algorithm.

to enable fast and reliable solutions that are tuned to optimally utilize measured data in model predictive control applications.

The conventional standard in industrial process control has been that as long as the gain is within 50% of the true value, and the time constant is within 30% of the actual value, then the feedback will account for the differences and maintain satisfactory control over the process. One of the novel contributions of this work is to provide a generalized result that verifies this conventional standard and provides a more accurate estimate for the parameter bounds where a controller can still maintain acceptable control.

1.2 Summary of Novel Contributions

This work makes the following contributions to the body of chemical engineering process control knowledge:

- The design and simulation of a model predictive controller for severe subsea riser slugging mitigation using post-installed fiber optic sensors. The controller improves performance over a PID slugging controller.

- A method for quantifying the benefits of sensor location in post-installed sensors used in slugging feedback control.

- Guidelines for bounded control model parameter identification that allow the controller to maintain acceptable control over a process.

- An online process to give statistical significance to control model identification by using nonlinear confidence regions.

- A basic switching algorithm that uses multi-fidelity models in real-time feedback control. The controller allows for continued control, even with loss of feedback, by using the predictions of a high fidelity simulator.

- A technique for implementing bumpless switching among control models.

- An advanced switching algorithm that builds upon the basic switch for use of multi-fidelity models in real-time control. The predictions from a high fidelity simulator can be incorporated into the control law for more precise control.

- A method for online control model identification that does not interrupt the process. A high fidelity model is used to simulate data to identify a control model from the current process conditions.

1.3 Dissertation Outline

The remainder of this dissertation is divided into six chapters as follows:

Chapter 2 describes a basic model predictive control scheme for severe subsea riser slugging control. The predictive controller is compared to a traditional PID controller and shows improved performance. For this controller to have a practical feedback loop, there must be a pressure sensor near the base of the riser. Therefore, the technical challenge of placing a post-installed,

clamp style, fiber optic pressure sensor on the riser is also described in this chapter. The location of the sensor and the effect on controller performance are explored in simulation. The simulations use an established riser slugging model for controller validation. Corrosion of the riser over time can alter the accuracy of the sensor, and the effect of pipe corrosion on sensor accuracy is explored. A common problem that may affect the sensor performance is pipeline wax deposition. This chapter reviews the mechanisms of wax deposition and the technologies used to address this issue.

Chapter 3 reviews the various estimation methods used in the oil and gas downstream, upstream, and petrochemical industries. After establishing the need for updating process models on various timescales, the five most commonly used estimation techniques are explained. As part of the discussion, the advantages and disadvantages of each method are compared and contrasted. A simple automated drilling application is used as a case study to demonstrate the strengths and weaknesses of each algorithm. The case study includes measurement noise, outliers, drift, and an incorrect initial condition. The use of nonlinear statistics to validate and establish a 95% confidence interval for control model parameter estimates is explored. Additionally, generalized quantified guidelines are given for Single Input Single Output (SISO) predictive control model parameter estimation ranges.

Chapter 4 introduces a simple switched control scheme that selects control models of various fidelity to take advantage of the benefits of each model in MPC. The switch transitions among models in a smooth manner that avoids excessive MV movement. The controller is applied to managed pressure drilling. Three models are used in the control scheme: a high fidelity model, a reduced order first-principles model, and an empirical model. After explaining each model and the switch itself, the control advantages and disadvantages of each model are discussed. The effectiveness of the controller is demonstrated in common drilling scenarios, including a loss of feedback measurements.

Chapter 5 builds on the previous switched control concept, but introduces a more advanced switching algorithm. That allows the accuracy of a high fidelity simulator to be used in a control law in real time. The advanced algorithm incorporates a model identification routine that uses the high fidelity model to generate real time simulated data to estimate the empirical control model parameters. The model identification procedure does not interrupt the process, and is suitable for nonlinear processes where online model identification is needed due to model inaccuracies. After

reviewing other work on high fidelity models in control, the controller stability is discussed. The controller is applied to common drilling situations including a process disturbance scenario, and shows satisfactory control.

Chapter 6 discusses the conclusions, highlights, and the future research directions of this work. Future research directions include automated high fidelity model tuning methods, more rigorous controller stability proofs, and nonlinear parameter joint confidence regions for tuning procedure initialization.

Following the body of the text, several appendices contain the APMonitor, Python and MATLAB computer code developed for this work. Appendix A contains the APMonitor MPC initialization and controller code for subsea riser slugging control. Appendix B has the APMonitor code that was used to develop the SISO control model parameter estimate guidelines. Appendix C contains the novel basic switched control algorithm in APMonitor and MATLAB. It includes code for the empirical, low-order, and high fidelity controllers that make up the basic switched controller. Appendix D is similar in form to Appendix C, but contains the code for the advanced switched controller that was developed in this work.

It should be noted that the author of this work has made several contributions to collaborative research projects that are not included in this dissertation. These contributions include: Powell, Kody M., Eaton, Ammon N., Hedengren, John D., and Edgar, Thomas F., *A Continuous Formulation for Logical Decisions in Differential Algebraic Systems using Mathematical Programs with Complementarity Constraints*, Processes, vol. 4, num. 1, 2016, DOI: 10.3390/pr4010007. The contributions made to this paper were to identify and clarify the limitations of the novel formula for inserting logical conditions in DAE systems, analyze the case study results for a level comparison, and explain existing alternative methods for handling logical decisions in gradient-based optimization algorithms. This placed the novel complementarity formula in context for the wider optimization community.

Also, significant contributions were made to: Aghito, Manuel, Eaton, Ammon N., Bjørkevoll, Knut S., Nybø, Roar, and Hedengren, John D., *Automatic Model Calibration for Drilling Automation*, SPE Bergen One Day Seminar, SPE-185926-MS. The contributions include helping to improve the optimal convergence criteria for a Monte Carlo style model parameter estimation al-

gorithm, translating the algorithm from MATLAB to Python with platform benchmark testing, and implementing a parallel processing structure for the algorithm.

Other contributions were made to Asgharzadeh Shishavan, Reza., Pixton, David S., Eaton, Ammon N., Park, Junho, Perez, Hector D., Hedengren, John D., and Craig, Andrew, *Addressing UBO and MPD Challenges with Wired Drill Pipe Telemetry and Model Predictive Control*, SPE/IADC, Managed Pressure Drilling and Underbalanced Operations, 2014, DOI:10.2118/168953-MS. Also, a presentation at the 2015 AICHE National Meeting in Salt Lake City, entitled *Addressing Discontinuous Process Control Challenges with Multi-fidelity Model Predictive Control*, was given.

CHAPTER 2. SUBSEA RISER SLUGGING CONTROL AND ARTERIAL WAX REMEDIATION REVIEW

A major portion of this chapter is published as: Eaton, A.N., Safdarnejad, S.M., Hedengren, J.D., Moffat, K., Hubbell, C., Brower, D.V., and Brower, A.D., *Post-Installed Fiber Optic Pressure Sensors on Subsea Production Risers for Severe Slugging Control*, ASME-OMAE, 2015, Volume 5B: Pipeline and Riser Technology, DOI: http://dx.doi.org/10.1115/omae2015-42196.

2.1 Introduction

Two-phase flow in pipelines can lead to an unstable flow regime known as slugging. When slugging with large amplitudes of pressure and flow occurs in subsea oil well production risers it is termed severe slugging. The undesired oscillations caused by severe slugging can slow oil and gas production, and cause accelerated wear to production equipment. Many technologies have been developed to control the effects of slugging including changing the design of separation equipment to better accommodate the slugs, the addition of a large topside holding tank to catch the slugs, and subsea phase separators that separate the liquid from the gas near the wellhead. These methods are often expensive or sub-optimal solutions [21]. Another way to mitigate the effects of severe slugging is through a choke valve at the topside of the production riser. The valve can be used by a controller to dampen the oscillations caused by slugging. This inexpensive solution was first reported as successful in 1990 [23] and has since been studied extensively. Several controllers have been designed for slugging suppression including PI [23], cascaded PID [24], neural networks [25], and gain-scheduling Internal Model Control (IMC) [26]; yet, predictive control has not been attempted in the literature. The controllers that have been reported generally attempt to control the pressure at the base of the riser. Many of the prior studies assume that pressure is measured or estimated at the riser touchdown zone where the slugs are generated. However, most production risers do not have a pressure measurement at the riser base, and slugging models may not be able to

accurately estimate the necessary states. Without a pressure measurement in this area it is difficult to create an effective feedback control loop. However, recent advances in post-installed fiber optic clamp design now allow a pressure measurement near this location [38]. This chapter details the plausibility of using a non-penetrating, post-installed fiber optic pressure measurement at a production riser base for predictive control of riser slugging. Factors that may lead to poor sensor calibration are discussed, and include corrosion and arterial wax deposition. Current methods to monitor, prevent, and remediate wax pipeline deposition are reviewed. Advances in fiber optic pipeline wax deposition monitoring technologies are also reviewed due to the enormous potential to both monitor deposition within the pipeline and simultaneously measure pressure for control applications.

2.2 Slugging Model

The slugging process was modeled in this study using a simplified three state model that was developed by Storkaas [39]. While other higher order slugging models exist, the three-state model is simple and sufficiently accurate for control purposes. The model consists of an L-shaped riser as depicted in Figure 2.2. The major assumptions of the model are:

1. The liquid velocity in the section upstream of the riser is constant.
2. The gas volume in the upstream section is constant.
3. The liquid mass holdup in the riser section is described by one dynamic state (m_L).
4. The gas mass holdup in the riser is described by one dynamic state (m_{G2}) and is related to the dynamic state of the gas mass in the upstream section (m_{G1}) by a pressure-flow equation of the low-point of the riser.
5. The gas behaves ideally.
6. There is a static pressure balance between the upstream pressure (P_1) and the topside pressure (P_2).
7. The system is at a constant temperature. Refer to [39] for a complete description of the model assumptions.

The dynamic states in the model are expressed with Equation 2.1 as a liquid mass balance, Equation 2.2 as a gas mass balance upstream of the riser, and Equation 2.3 as a gas mass balance in the riser section.

$$\frac{dm_L}{dt} = w_{L,in} - w_{L,out} \tag{2.1}$$

$$\frac{dm_{G1}}{dt} = w_{G,in} - w_{G1} \tag{2.2}$$

$$\frac{dm_{G2}}{dt} = w_{G1} - w_{G,out} \tag{2.3}$$

Here, m_L is the mass of the liquid, m_{G1} is the mass of the gas in the section upstream of the riser, and m_{G2} is the mass of the gas in the riser. The variable w in its various forms is the mass flow rate with subscripts L for liquid and G for gas. The mass flow of gas upstream of the risers given by Equation 2.4.

$$w_{G1} = v_{G1}\rho_{G1}\hat{A} \tag{2.4}$$

Here $\hat{A}$ is the cross-sectional area of the flowing gas at the riser base, ρ_{G1} is the density of the gas in the upstream section of the system, and v_{G1} is the velocity of the gas at the low point of the riser. This velocity of the gas in the section upstream of the riser is described by Equation 2.5.

$$v_{G1} = K_2 \left(\frac{H_1 - h_1}{H_1} \right) \sqrt{\frac{P_1 - P_2 - \rho_L g_c \alpha_L H_2}{\rho_{G1}}} \tag{2.5}$$

In this case, K_2 is a multiplicative factor that adjusts the magnitude of the gas flow, H_1 is the critical liquid level at the low-point of the riser, h_1 is the actual liquid level in the upstream of the riser, P_1 is the pressure in the section upstream of the riser, P_2 is the pressure in the riser, ρ_L is the density of the liquid, g_c is the gravitational constant, α_L is the average fraction of liquid

in the riser, and H_2 is the height of the riser. The valve was modeled using a simplified equation, Equation 2.6.

$$w_{out} = K_1 z_c \sqrt{(P_2 - P_0)\rho_T} \tag{2.6}$$

Here w_{out} is the total mass flow rate exiting the valve, K_1 is a model tuning parameter, z_c is the valve percent opening, ρ_T is the average density of the fluid flowing through the valve, and $(P_2 - P_0)$ is the pressure drop across the valve. Additionally, the fluid distribution in the riser is modeled using Equation 2.7.

$$\alpha_{LT} = \alpha^*_{LT} + \frac{q^n}{1+q^n}(\alpha_L - \alpha^*_{LT}) \tag{2.7}$$

α_{LT} is the liquid fraction in the section immediately upstream of the control valve, α^*_{LT} is the liquid fraction without entrainment, q is a parameter that describes the transition between the full entrainment and no entrainment. n is a tuning constant that changes the slope of the transition. The equations presented here are the major equations used to define the model riser; for a complete description refer to [39]. One of the limitations of this model is that the mass flow rates entering the system ($w_{L,in}, w_{G,in}$) are constant. This attribute constrains the production to these values, and does not allow the controllers to maximize production. Figure 2.1 shows the open loop response of the riser base pressure, topside pressure, and mass flow rate out of the system as a function of valve percent open. When the valve is 10% open, the slugs are effectively dampened. The minimum valve position where slugging occurs is 13% open.

2.3 Controller

Two controllers were used in this study, a model predictive controller and a traditional PID controller.

2.3.1 MPC Controller

One of the advantages of MPC over traditional controllers is its ability to predict future modeled disturbances and respond to them before they affect the process. MPC uses a process

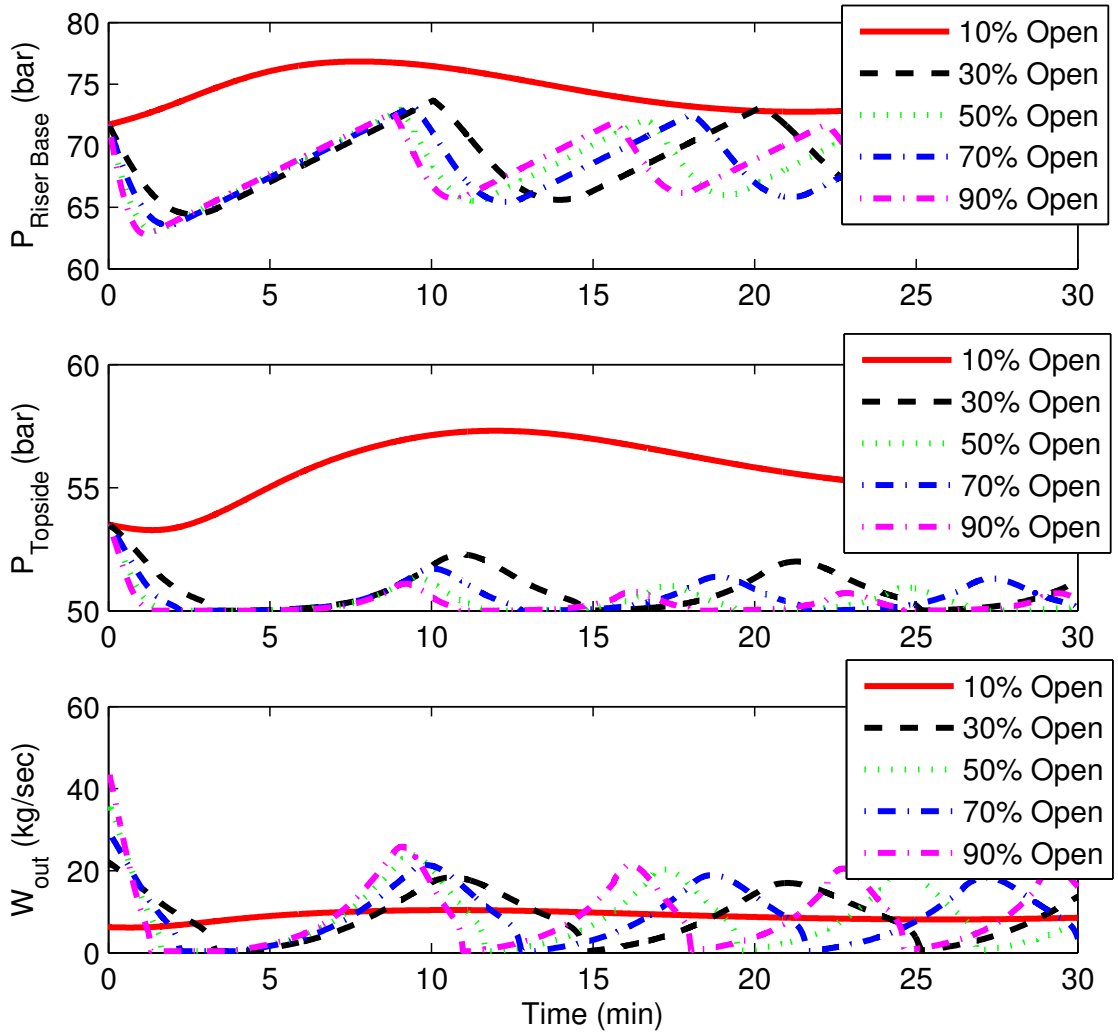


Figure 2.1: Open loop response of the riser base pressure, topside pressure, and mass flow rate out of the riser to valve percent open.

model to optimize the controllers output over a specified time horizon. The benefits of MPC come at the expense of increased computation time. The model used for optimization in this controller was a modified First Order Plus Dead-Time (FOPDT) model shown in Equation 2.8.

$$\tau_p \frac{dP_1}{dt} = -(P_1 - P_{ref}) + K_p(z_c(t-\theta) - z_{ref}(t-\theta)) \tag{2.8}$$

Here, τ_p is the process time constant, P_{ref} is a reference pressure, z_{ref} is a reference valve position, and κ_p is the process gain. The MPC controller for this project was created in the APMon-

itor modeling language. APMonitor uses collocation methods to convert the models Differential and Algebraic Equations (DAEs) into a nonlinear programming (NLP) optimization problem [40]. The NLP problem is then given to an active set solver, APOPT, to find the optimal controller output. The controller output is the valve position (z_c), and the inputs are the constant mass flow rates of liquid ($w_{L,in}$) and gas ($w_{G,in}$) into the pipeline. It also receives a pressure measurement from the fiber optic sensors at the base of the riser (P_1) and the topside (P_2). The MPC controller uses an ℓ_1-norm objective function in the optimization routine. This allows the controller to use a dead-band set point instead of just a single value as with the standard ℓ_2-norm objective function. This dead-band defines the range of acceptable values for the controlled variable, which in this case is the riser base pressure (P_1). This range of acceptable values gives the controller greater flexibility in arriving at an optimal solution. The ℓ_1-norm objective function has also demonstrated better rejection of measurement noise, outliers, and drift than a squared error objective function [6].

2.3.2 PID Controller

The PID controller used in this study was a modified version of the PID controller created by [39]. The modifications include the addition of anti-reset windup and deletion of rate limiting on the valve position. The derivative term was set equal to zero. After these modifications were made the controller was appropriately tuned and included in the study as a benchmark controller.

2.4 Simulation

The riser slugging is simulated in MATLAB and Simulink. The pipeline-riser system is simulated as a 0.12 meter (4.75 inch) diameter flowline with 4300 meters (2.67 miles) of line upstream of the riser. The riser is 300 meters (984 feet) deep and runs for 100 meters (328 feet) to the topside receiving facilities. The angle of incline at the base of the riser (θ) is 1.57 degrees (see Figure 2.2).

The gas and liquid mass flow rates entering the system are 0.36 kg/s ($w_{G,in}$) and 8.64 kg/s ($w_{L,in}$) respectively. The system temperature is assumed constant at 308 K. The molecular weight of the gas is 35 kg/kmol, and the liquid is pure oil with a density of 750 kg/m3. Finally, the pressure of the topside receiver is assumed constant at 50 bar. The pressure at the riser base is used

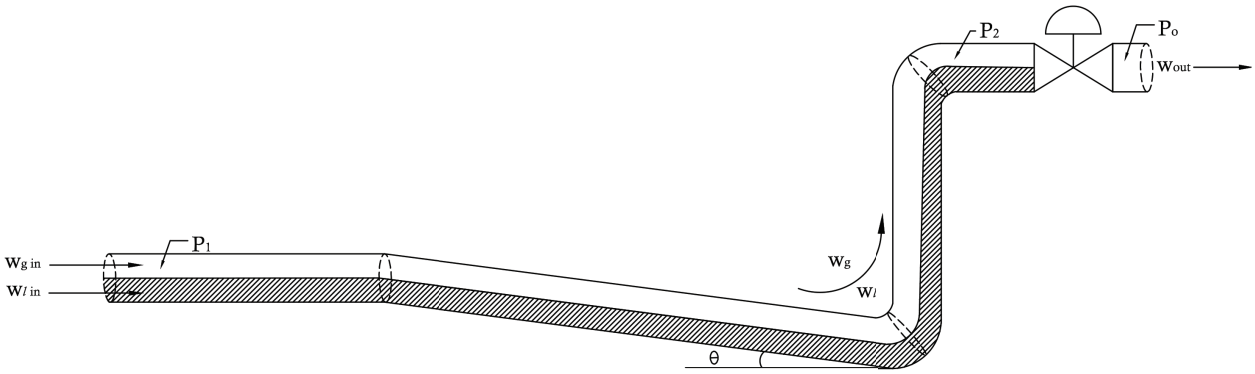


Figure 2.2: Illustration of the L-shaped riser simulated in this study.

as the controlled variable (CV) and the valve position is the manipulated variable (MV) for the simulation. When the pressure oscillations are dampened, the flow will also stabilize. The addition of a pressure measurement at the riser base completes the feedback control loop. The Simulink diagram of the process is found in Figure 2.3.

2.5 Simulation Results

In this simulation, the controllers were activated at 33 minutes. The set point is 70 bar until 50 minutes when it moves to 75 bar. At 67 minutes it moves again to 69 bar (see Figure 2.4). The controller output and the process response are shown in Figure 2.4.

Figure 2.4 demonstrates the superior performance of the MPC controller over the PID controller. While the rise times of the MPC and PID controllers are identical, the MPC controller achieves the set point quickly, while the PID controller has minor persistent offset.

2.5.1 Measurement Position and Time Delay

The effect of clamp position, and therefore pressure measurement delay, on riser slugging control was explored. If the pressure measurement location is at the riser base, then there will be no time delay in the measurement. However, if the position of the sensor clamp is moved vertically up the riser then the time that the controller has to adjust to the slugs will decrease. If a pressure measurement is only available on the topside then the measurement time delay will be

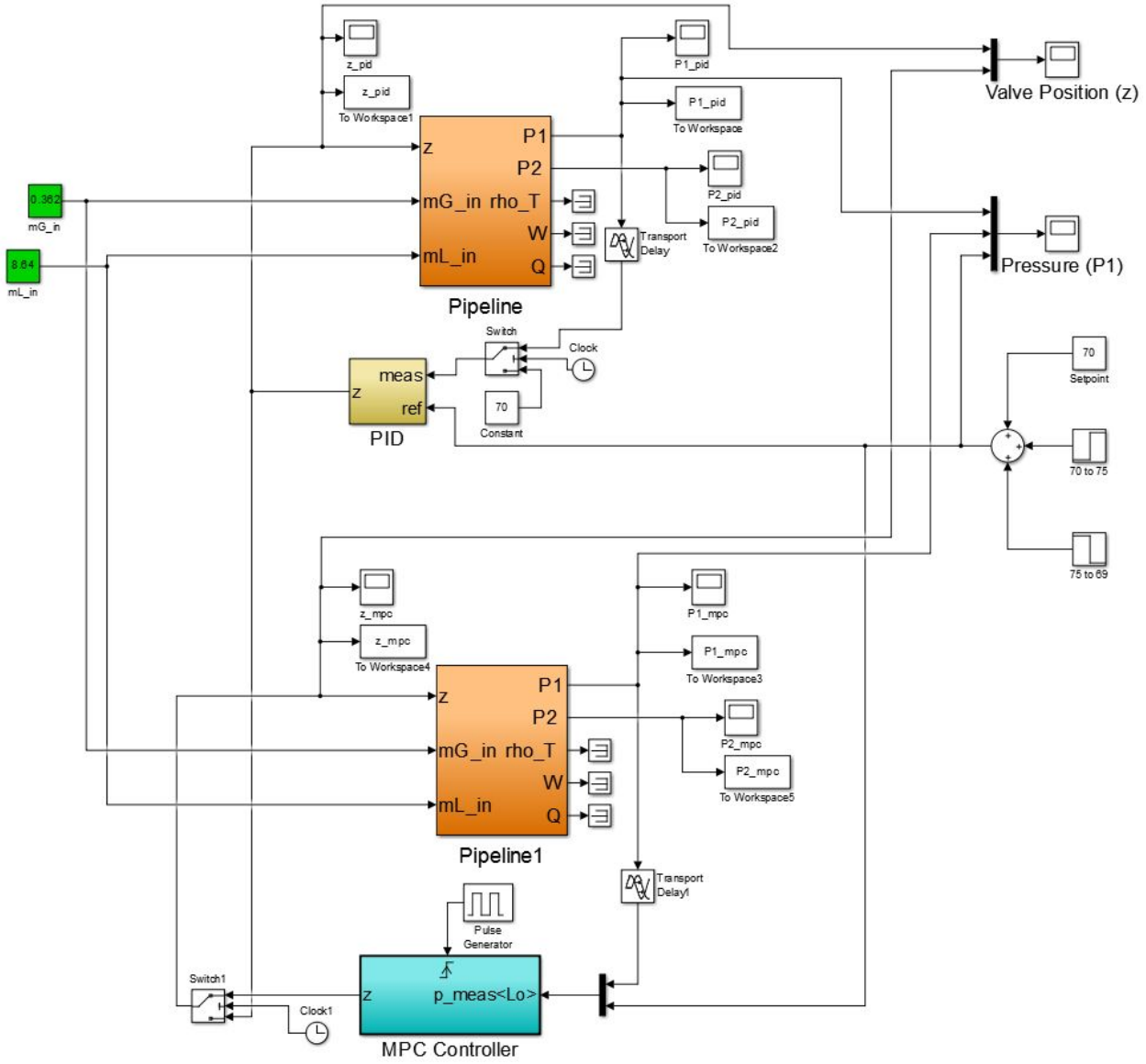


Figure 2.3: The Simulink diagram of the slugging controllers used in the simulation. The lower block is the MPC controller.

at a maximum and the controller will not have sufficient time to effectively control the slug. The theoretical time delay was calculated using Equation 2.9.

$$\theta_m = \frac{H_2 \rho_L D^2}{4 W_{in}} \tag{2.9}$$

In this equation, θ_m is the measurement time delay, D is the riser diameter, and W_{in} is the total mass flow of the system. All other variables are previously defined. The liquid density was

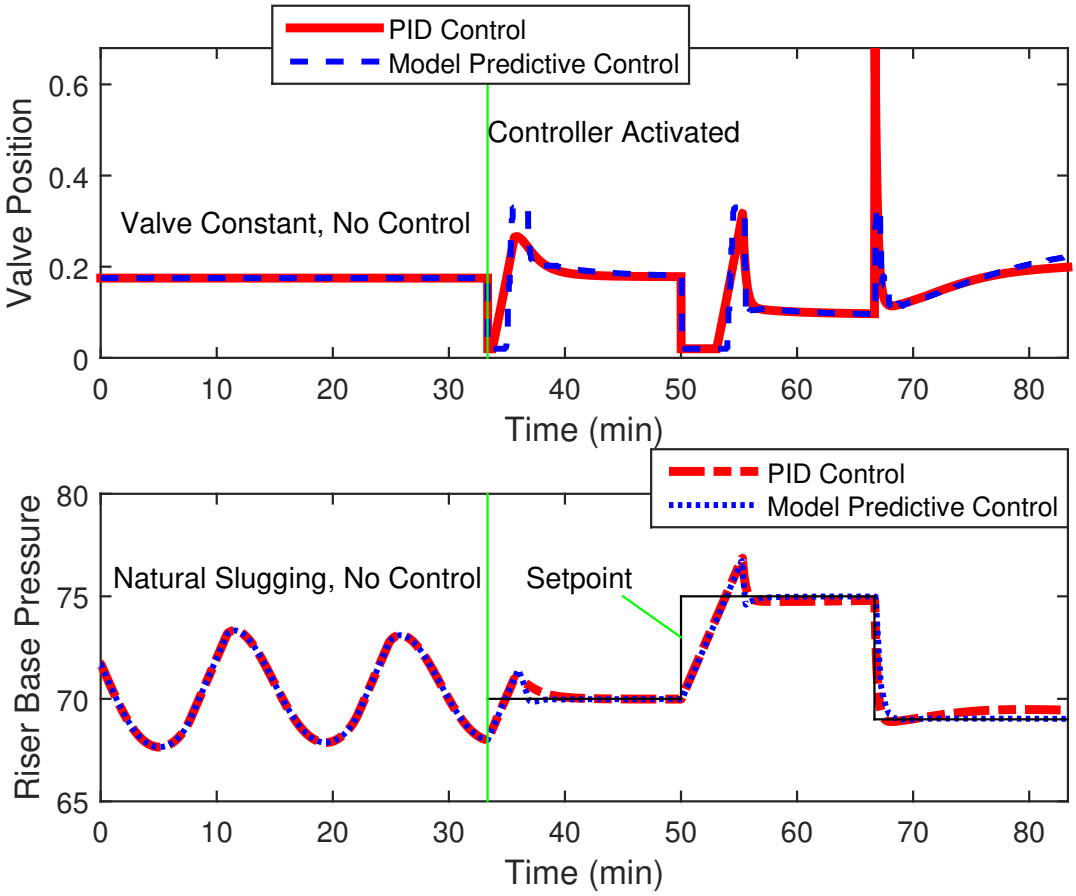


Figure 2.4: Results of the riser slugging simulation. The top graph is the valve position (MV) and the lower graph is the riser base pressure (CV). The PID controller is the solid line (red) while the MPC is the dotted line (blue). The controller was activated at 33 minutes and the set point was changed at 50 minutes and at 67 minutes. The lower plot shows the predictive controller preforming slightly better than the PID controller. This is indicated by the minor persistent off-set at from 55 to 67 minutes, and from 73 to 83 minutes. Also, the settling time of the MPC controller for the initial set point is 37 minutes compared to 39 minutes for the PID controller.

used in this calculation because it will result in the maximum possible time delay. The actual mixture density will be less and so will the delay. Using the liquid density constitutes the worst case scenario. Applying Equation 2.9 to the simulated case gives a measurement time delay of 105 seconds. This represents what the delay would be if the topside pressure measurement were the only measurement used in the control loop. Figure 2.5 shows the PID controller response to 105 seconds of time delay.

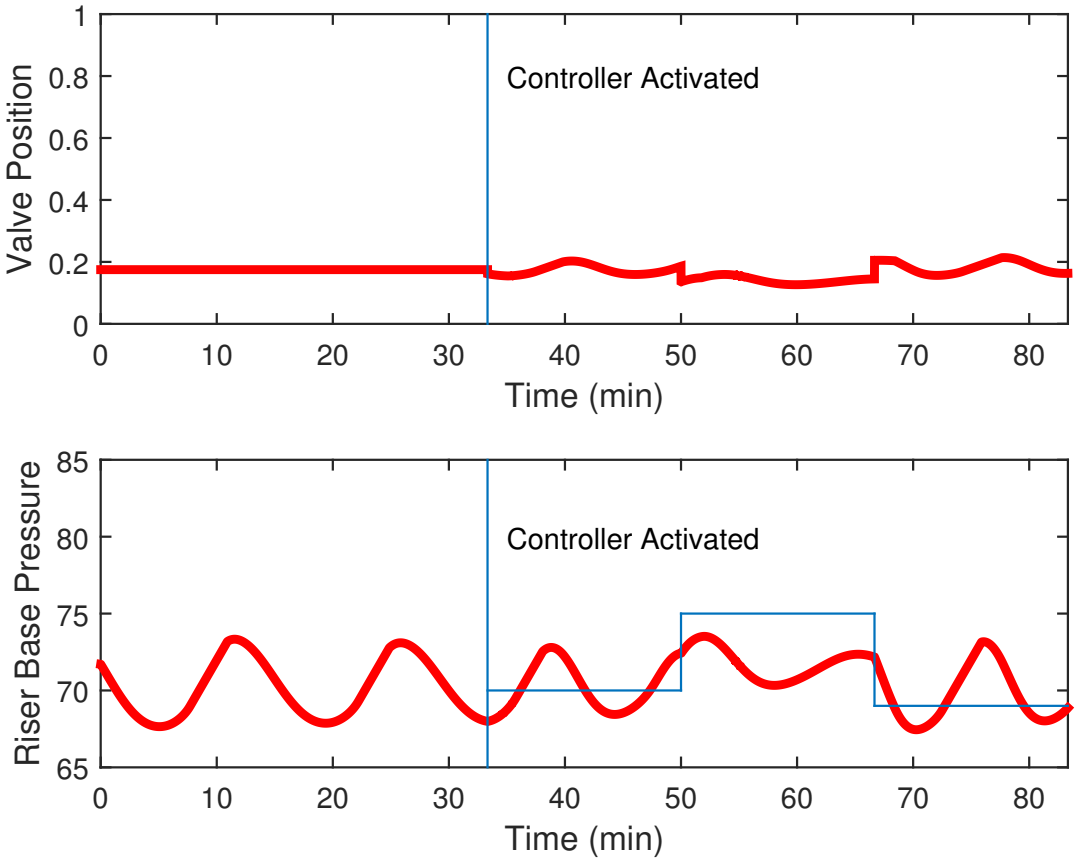


Figure 2.5: PID controller response with only a topside pressure measurement (105 second time delay). The top plot shows valve position, and the bottom plot shows riser base pressure. The controller is activated at 33 minutes. The set point changes from 70 bar to 75 bar at 50 minutes, then to 69 bar at 67 minutes.

This demonstrates the controller performance when only a topside pressure measurement is available in the control loop. Additionally, the time delay was changed to simulate the point at which the PID controller could no longer control the process. Figure 2.6 shows the PID controller response to varying measurement time delay. With 50 seconds of time delay, corresponding to 167 meters of riser, the controller is unable to dampen the oscillations. This is the maximum riser height that this controller can regulate using only topside pressure measurements.

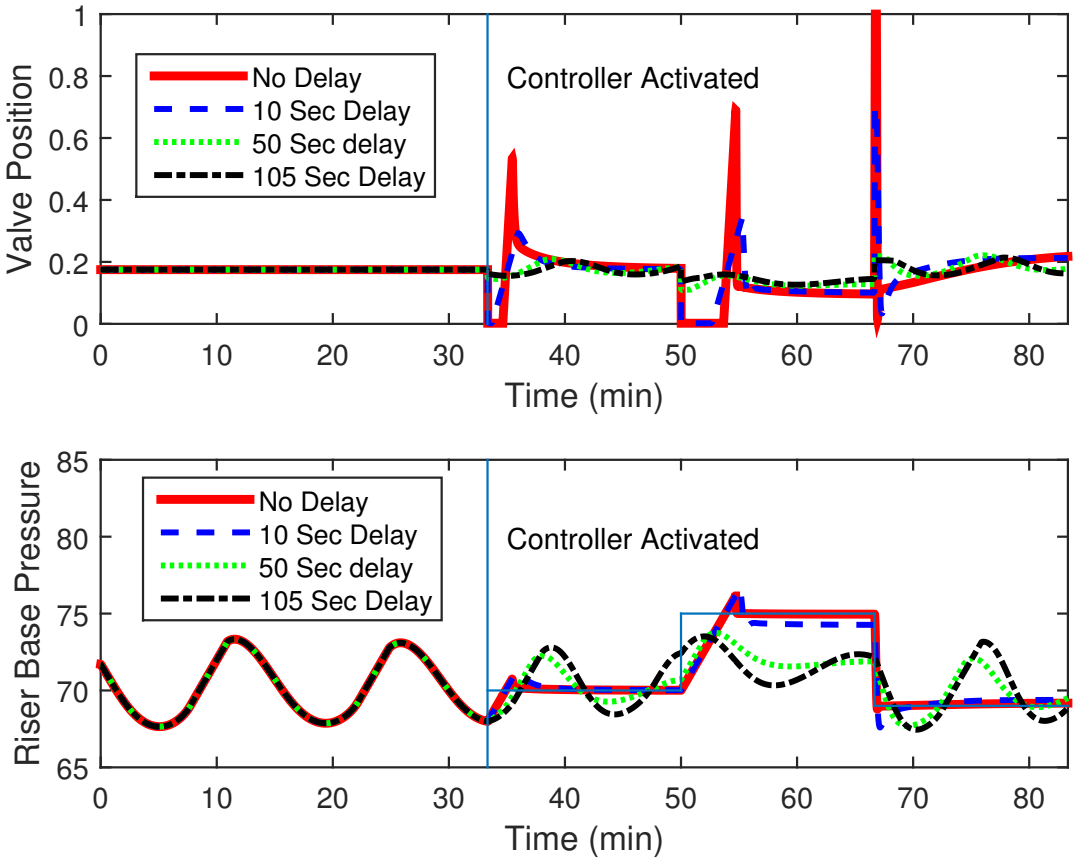


Figure 2.6: PID controller response with varying measurement time delay. The top plot shows valve position, and the bottom plot shows riser base pressure. The controller is activated at 33 minutes. The set point changes from 70 bar to 75 bar at 50 minutes, then to 69 bar at 67 minutes.

2.6 Post-Installed Fiber Optic Sensor Clamp

This work builds upon prior work on the design and deployment of fiber optic subsea sensing of temperature, pressure, vibration, strain, and flow assurance [38]. The post-installed and non-penetrating sensor can be installed by a diver or remotely operated vehicle (ROV), depending on the target depth. A pressure measurement at the riser base eliminates the need for estimators in the control scheme and reduces computation time. With advances in subsea fiber optic monitoring and post-installed clamp design, virtually any riser can be fitted with pressure measurements at the base of the riser. There are two types of clamps that can be used to secure the optical fiber Bragg grating (FBG) sensors to the pipe. The adhesive clamp and the friction clamp are shown in Figure 2.7. One of the major advantages of clamped FBG sensors over other fiber optic measurement

systems, like Distributed Temperature Sensing (DTS), is that the exact location of the sensor is known. The light interacts with gratings that have been etched into the glass core of the fiber, not macromolecular phasing within the fiber such as is the case with DTS. FBG sensors are manufactured by doping the optical fiber with Germania, then a laser is used to etch several gratings into the fiber core with spacing that is similar to the wave length of the light used in the application. Each grating on the fiber reflects a slightly different frequency allowing multiple FBGs to be multiplexed on one fiber [41]. FBGs act as an optical filter by allowing certain wavelengths to pass through the grating while others are reflected (see Figure 2.8). Both the reflected signal and the transmitted signal are interpreted by an optical interrogator allowing for measurement redundancy. The spacing, and therefore refractive index, of the gratings changes when the fiber experiences a change in temperature or spatial movement; from these readings, strain, pressure, and flow rate can be calculated [42]. For example, when the pressure in a pipe increases, the pipe wall expands minutely. This alters the grating spacing in the fiber, and the wavelength of light in the fiber is shifted. This shift has a linear relationship with strain, and temperature, and is used to determine the pressure [41]. When compared to traditional strain gages, FBGs were found to perform with similar sensitivity ($\sim 1.2pm/\mu\varepsilon$ [43]) [44]. Also, when compared to thermocouple measurements the FBGs performed with similar accuracy ($\pm 0.5°C$,[45]), but with a faster response time [42]. Because FBGs measure the expansion of pipe walls, they need to be calibrated when installed. If there is pipe wall thinning due to corrosion, or thickening due to material deposition, then the sensor can lose calibration.

2.7 Corrosion, Drifting, and Measurement Delay

For the controller to accurately regulate the choke valve and suppress slugging, it must be able to quickly interpret any change in pressure in the pipe. If there is a time delay between actual pressure change and the measurement by the FBG sensor, it could potentially cause the controller to become unstable. Therefore, the relationship between pressure change and pipe wall strain is explored. There are two principles that govern the change of strain. First, strain will change instantaneously on the inside of the pipe surface following fluctuation in pressure when the steel is modeled as linearly elastic [46]. This strain will then propagate through the thickness of the material at the longitudinal speed of sound. This was measured to be 16,600 feet per second in 1020

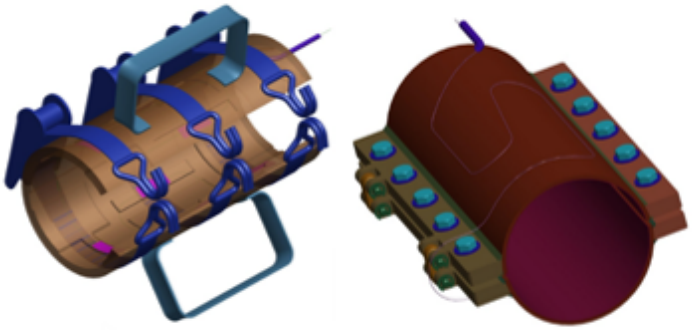


Figure 2.7: Adhesive clamp (left) and friction clamp (right) for installing a pressure sensor at the riser touchdown zone.

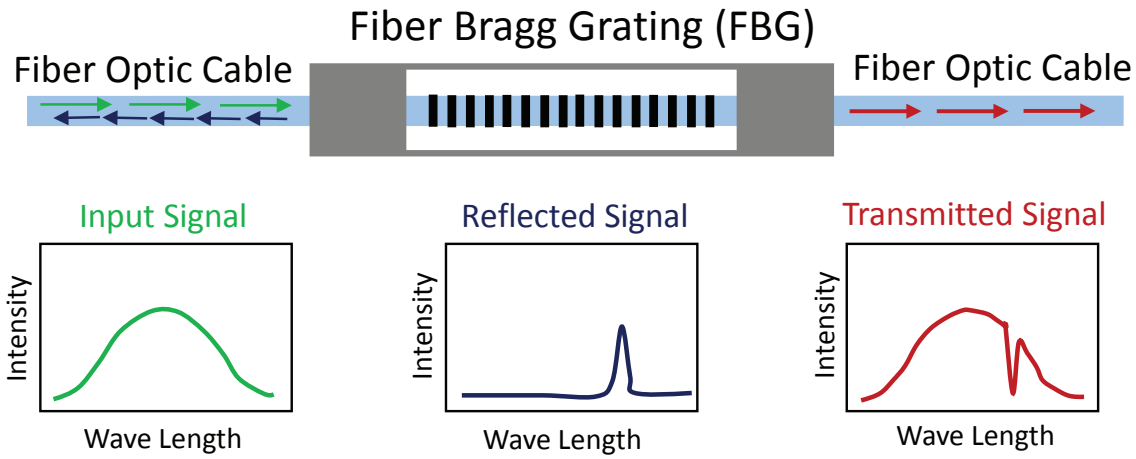


Figure 2.8: Diagram of FBG operational principles.

steel [47]. Assuming a 16 inch Schedule 80 pipe, the time required to detect a change in pressure is 4.23 microseconds. Compared to the average speeds of fluid flowing through the pipe, this amount of time is negligible. Therefore, the pressure sensor will return information to the controller fast enough to promptly adjust the choke valve opening. Re-calibration of the pressure sensor will become necessary once certain strain-inducing mechanisms become significant. Creep will not need to be considered since the pipe is operating at subsea temperatures [48]. However, corrosion on the inside of the pipe will thin the pipe wall, increasing strain and causing the calibration curve

to drift. A simulation where 0.01 inches of steel have corroded was analyzed and the results are shown in Figure 2.9.

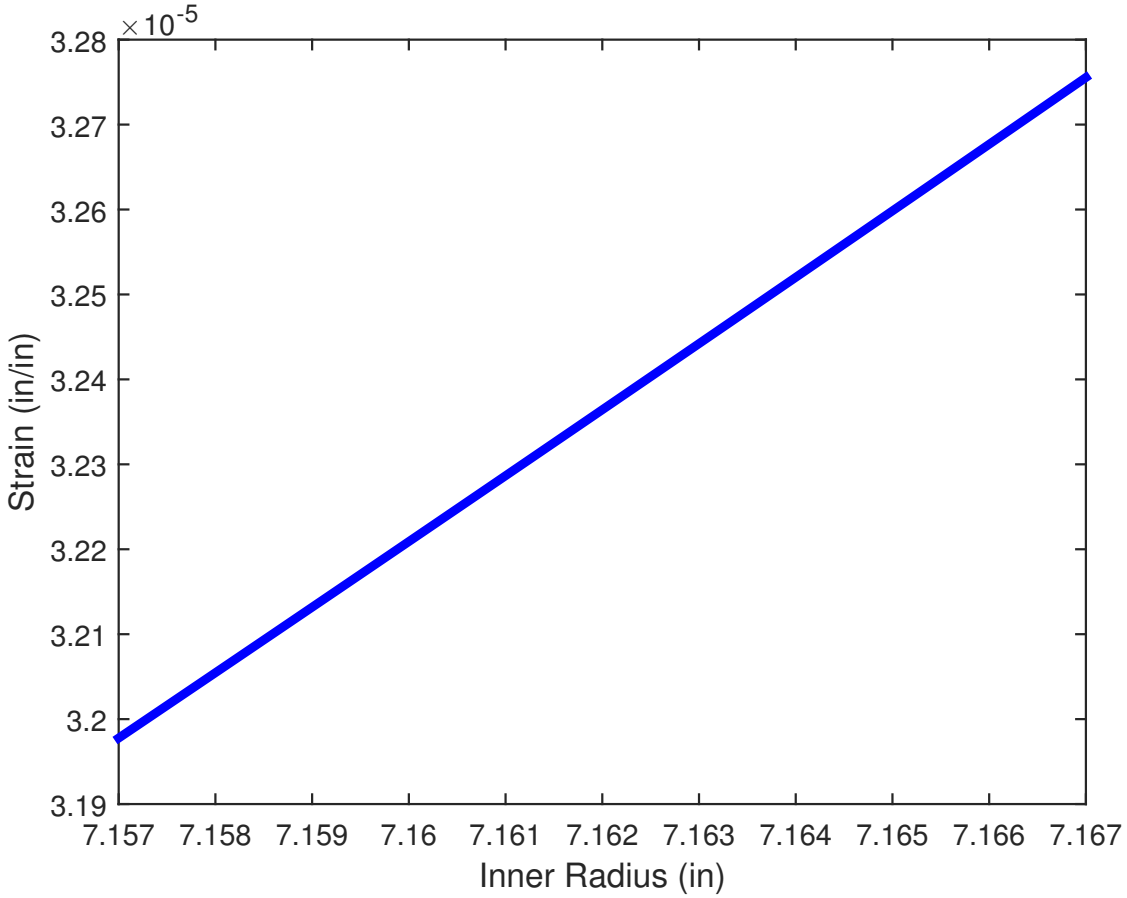


Figure 2.9: Strain vs. Pipe wall thickness in simulated corrosion of 0.01 inches of the inside of the pipe. Note: the relationship appears linear on this scale, but is actually nonlinear.

The method of calculating strain is based on contributions from both radial and tangential stresses [49]. Over this amount of corrosion, the strain rises by 2.6% as seen in Figure 2.9. Therefore, depending on the rate of corrosion within the pipe, the pressure sensor will need to be periodically re-calibrated in order to accurately measure the pressure.

2.8 Riser Arterial Wax Deposition Monitoring

Another phenomenon that can cause interference with the pressure sensor calibration is arterial wax deposition. Wax is a hydrocarbon molecule with chains longer than approximately 25 carbon atoms. It is naturally dissolved in crude oil, and is one of its major components. Wax crystals form on a pipeline wall when the crude oil temperature drops below the Wax Appearance Temperature (WAT) [50]. The precipitated wax can then lose momentum and deposit on the wall of the pipe [51]. Figure 2.10 is a diagram of the wax deposition process in crude oil pipelines such as subsea risers.

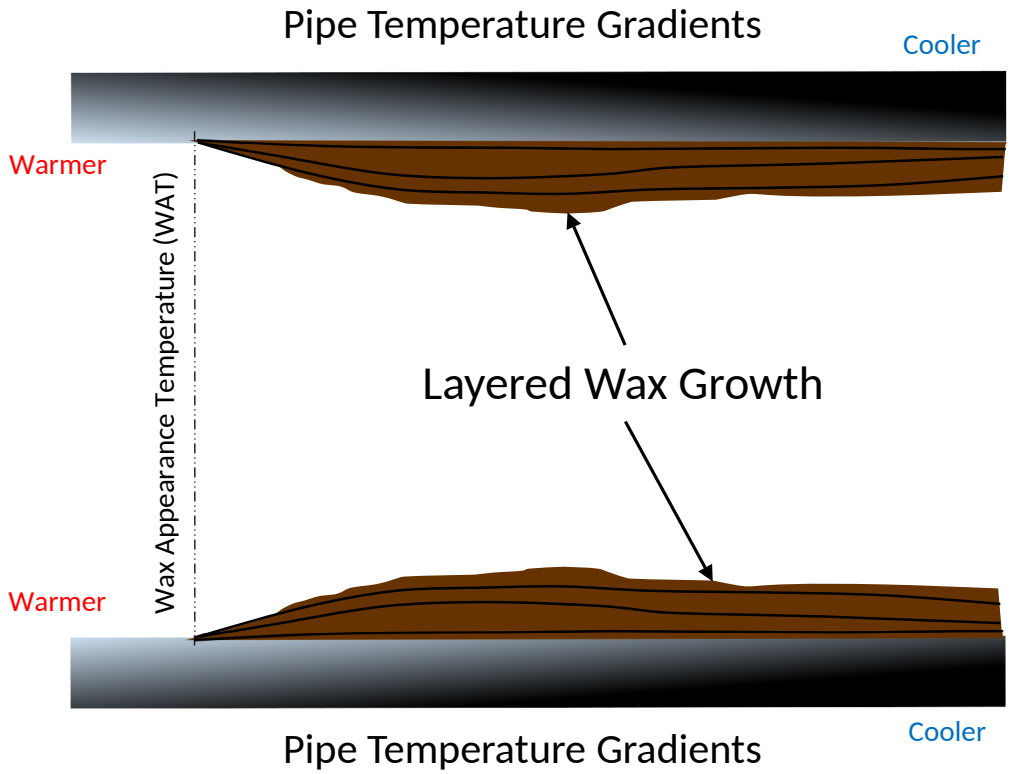


Figure 2.10: Wax deposition in a crude oil pipeline.

Wax deposition constrains the cross-sectional flow area of the pipe and leads to inaccurately calibrated fiber optic sensors and poor slugging control. It also increases drag in the pipe which increases pumping costs. If wax deposits are left untreated, they harden within the pipe and make cleaning the line nearly impossible [52]. Pipelines are typically cleaned through a process called pigging. A pig is a device put into a side flowline that is tied into the main production line. It

scrapes the pipe walls and pushes the wax buildup to the surface where it is removed from the line. To assist pigging, operators implement wax prevention and treatment programs. The programs generally include chemical additives such as pour point depressants and wax crystal modifiers [53]. Monitoring wax deposition, along with rigorous line cleaning programs will lead to better pressure sensing and rise slugging suppression.

Table 2.1 contains technologies and methods used to control arterial wax deposits. The table is divided into three major categories: *Used*, *Available*, and *Research*, and three control strategies: *Monitoring*, *Prevention*, and *Remediation*. The methods in the Used category are things that are commonly used by the petroleum industry. Those in the *Available* category are mature technologies that are on the market, but are not commonly used. The *Research* technologies have demonstrated viability, but are not currently used or even on the market. The list is not exhaustive, but does represent the most common methods and technologies for arterial wax deposition control.

Pipeline and flowline operators currently use few, if any, methods to actively monitor wax deposition; yet, researchers have developed various wax deposition monitoring techniques. This may be due to researchers attempting to address the lack of monitoring in industry, while operators feel the cost of active wax monitoring outweighs its benefits. Whatever the case, the monitoring techniques in Table 1 can be grouped into three major measurement categories, heat transfer ([65], [64], [66], [69]), electromagnetic wave, ([68], [62], [71], [41], [57], [54]), and compression wave ([70], [67], [60]). The heat transfer methods aim to determine wax deposit thickness by measuring the heat flux through the pipe wall. These techniques take advantage of the fact that wax acts as an insulator which causes the pipe to lose less heat to the surroundings as deposits thicken. Electromagnetic waves have also been used to detect wax deposition. These techniques use the same principles as medical imaging technologies such as Computer Aided Tomography (CAT) scanning [71], [54]. Compression wave methods include ultrasound imaging [70] and sending pressure pulses down the pipeline that reflect back when significant blockage has occurred [67]. While many methods of actively monitoring wax deposition in crude oil pipelines have been developed, none of them are commonly used by pipeline operators. Producers have favored wax deposition simulation and regular pigging over online monitoring.

2.9 Conclusion

The plausibility of using post-installed, non-penetrating fiber optic sensors for controlling severe riser slugging is considered. Recent advances in clamp design allow these pressure sensors to be post-installed on virtually any riser or pipeline. The effect of the measurement time delay is investigated as dictated by the pressure device location. For this simulated system, a traditional PID controller with topside-only pressure measurement performs poorly when the riser height exceeds 167 meters. In contrast, a PID controller with a pressure measurement at the touchdown zone of the riser can successfully control slugging. A MPC controller is compared to this PID controller and found to provide superior control of slugging. In addition to the predictive qualities of the MPC controller, it also utilized an ℓ_1-norm objective function which will allow for better noise, drift and outlier rejection in the field. Additionally, the corrosion effects on the sensor are simulated and as corrosion occurs the sensors will need to be re-calibrated. Technologies for preventing, monitoring, and remediating arterial wax deposition in risers and pipelines are reviewed. Potential exists for developing novel active deposition control systems based on active monitoring of pipeline deposition. The active monitoring capabilities and maturity of FBG sensors pose them to play a significant role in pipeline control applications in the future.

Table 2.1: Crude oil pipeline arterial wax deposition control technologies

	Monitoring	Prevention	Remediation
Used	• smart pigs	• chemical deposition inhibitors	• coiled tubing
		• chemical drag reducers	• cut out and replace pipe
		• cold finger (determines deposition rate)	• dispersants
		• deposition modeling and estimation	• electrical heating
		• external insulation	• hot oil injection
		• flow loop (determines deposition rate)	• hot water injection
		• operation at high flow rates	• scrapper pigs
		• pour point depressants	• solvents (xylene)
			• steam injection
Available	• computed tomography[54]	• various chemicals[55]	• chemical reaction heat generation[56]
	• radioisotope tracing[57]	• internal polymer lining[58]	• microbial treatments[58]
	• distributed temperature sensing[59]		
Research	• acoustic chemometrics[60]	• various chemicals[61]	• power ultrasonic methods[58]
	• capsule monitoring[62]	• magnetic fluid treatment[58]	• inductive heating[63]
	• electrical resistance[64]	• power ultrasonic treatment[58]	
	• heat pulse monitoring[65]		
	• heat transfer[66]		
	• optical fiber Bragg gratings[41]		
	• pressure pulses[67]		
	• radiography[68]		
	• thermal wave processing[69]		
	• ultrasound and strain gauges[70]		
	• x-ray diffraction[71]		

CHAPTER 3. REVIEW OF INDUSTRIAL ESTIMATION TECHNIQUES WITH MODEL PARAMETER ESTIMATION GUIDELINES

This chapter is published as: Hedengren, J.D. & Eaton, A.N., *Overview of Estimation Methods for Industrial Dynamic Systems*, Optimization and Engineering, Springer, Vol. 18 (1), 2017, pp. 155-178, DOI:10.1007/s11081-015-9295-9.

3.1 Introduction

Over the past 10 years many sectors of the oil and gas industry have seen a dramatic increase in the number and quality of available measurements. To capture the benefits of increased available measurements, the information must be distilled into relevant and actionable information [72], [73]. This chapter reviews the current industrial practice for estimation in the oil and gas refining, chemicals, exploration, and production sectors and provides guidance on model accuracy requirements for satisfactory control performance.

One opportunity is the increase in the available bandwidth to monitor the drilling process with along string and down-hole measurements to monitor pressure, vibration, temperature, and orientation. New technology has been deployed to drastically increase the data transmission rate to the BHA or along the drill string. Mud pulsing was previously the most common form of communication in which 3-45 bits per second could be transmitted from the BHA to the surface monitoring system via a series of pressure waves through the inner pipe. In addition to providing a communication pathway, the pumped mud removes tailings and cools the drill bit. As the depth of drilling increases, the attenuation of mud pulses increases and mud pulse data is frequently unavailable. Recently, wire-in-pipe technology has increased this rate by approximately 10,000 times (see Figure 3.1) [74], [75].

This increase in information allows two-way communication and presents opportunities for improved monitoring and control of directional, managed pressure, and under-balanced drilling [76]–

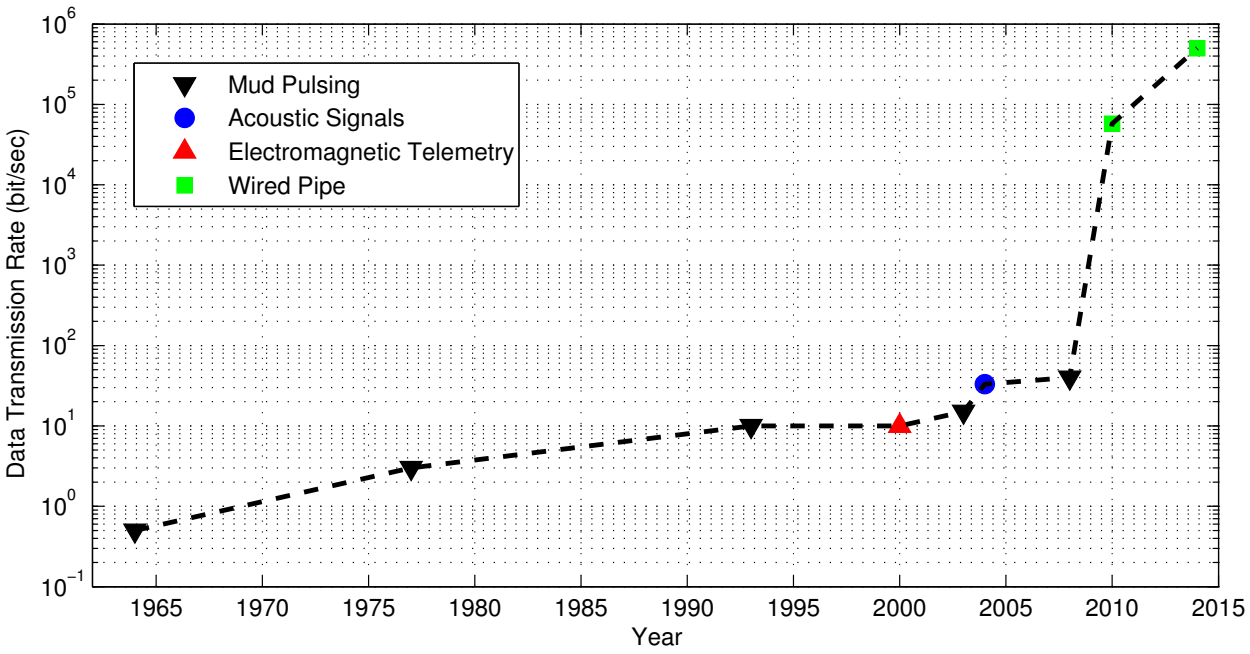


Figure 3.1: Best available data transmission rates in drill strings [2], [3]. The recent increase in throughput and bi-directional communication has created a new opportunity for better utilizing the information. Without interpretation, the increased data does not necessarily lead to increased understanding or value.

[78]. Similar improvements in measurement technologies are occurring in other parts of the oil and gas industry. This chapter reviews estimation techniques to garner the most useful data possible. These include a filtered bias update, Implicit Dynamic Feedback (IDF), Kalman filtering, and MHE [37].

MHE is an optimization approach that aligns process models with available measurements to determine a best estimate of the current state of the process and any potential disturbances. This presents the opportunity for earlier detection of disturbances such as gas influx into the borehole, process equipment faults, and improved state estimates for process automation. Explicit approaches commonly used in current practice, such as measured variable bias updating and Kalman filters, are compared to MHE approaches. The downside to MHE approaches is the increased computational load required to solve the problem and the difficulty to obtain optimal tuning. This chapter discusses techniques to overcome both of these obstacles to enable fast and reliable solutions that are tuned to optimally utilize measurement information in model predictive applications.

3.1.1 Time-Scales of Process Monitoring

Measurements of slow or fast processes pose unique challenges. The slow fouling of a heat exchanger [79] or the fast build-up of hydrates [80] are two examples of processes with different process time constants. With fouling or plugging as one of the top loss categories industry-wide, there are many opportunities for utilizing measurement technology to monitor the short or long term effects of these disturbances [81]. In particular, deep-sea pipeline monitoring poses a challenge due to the remote environment, intermittent weather incidents, and gradual fatigue factors [82]. There is a need for improved monitoring of existing and new projects to give insight into the conditions that lead to failure. Analytical models utilize the data to monitor and control the operational integrity for flow assurance and riser integrity [83].

Frequency of Optimization Updates

Before discussing techniques for measurements, it is informative to review the corresponding optimization applications. Optimization can occur after a model is synchronized to available process measurements or inputs. Process optimization is used in the oil and gas industry at various phases of the process lifecycle. As shown in Figure 3.2, optimization of process design occurs once at the beginning of the lifecycle [84]. This may include sizing of vessels, valves, etc. Optimization is also used to guide flow of products with supply chain optimization. This may occur on a weekly to monthly basis [85], [86]. Dynamic optimization is concerned with long time periods as well and covers processes such as defouling, turn-around operations, and production scheduling. On an hourly basis, Real Time Optimization (RTO) with large-scale steady state models [87] is used to determine new targets for plant-wide operations [88]. On the second to minute time-scales, the steady-state conditions from the RTO application are passed to MPC applications that dynamically drive the process to new target values [89], [90]. Recent work involves passing not only the new target values but also the economics from the RTO applications to the MPC applications as well [91], [92]. Ensemble methods increase the reliability of the control methods, much like redundant sensors or physical equipment increase the reliability of operations by making the system less sensitive to a single failure [93].

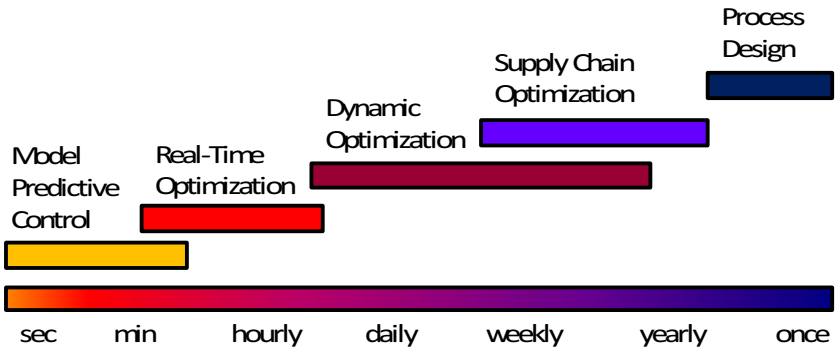


Figure 3.2: Time-scales of optimization technologies applied in oil and gas industry.

Frequency of Model and Measurement Alignment

Just as optimization is applied at varying time-scales, measurement reconciliation is performed at varying time-scales that are analogous to the optimization approaches (see Figure 3.3).

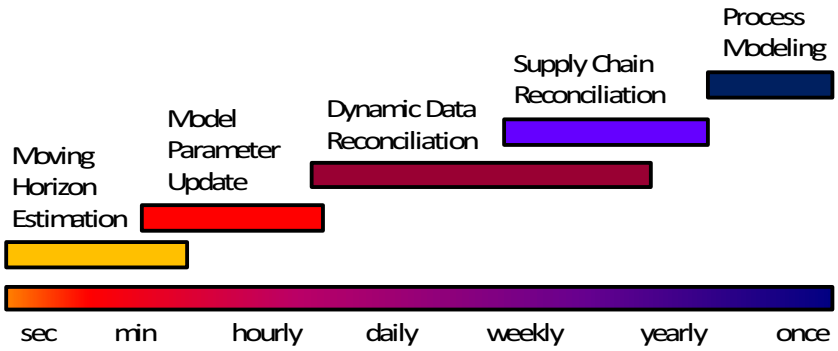


Figure 3.3: Time-scales of measurement reconciliation applied in the oil and gas industry.

A sufficiently accurate model is required to optimize the design or control a process. During the lifecycle of a facility, process modeling is typically conducted during the design and start-up of a new process. Data from other related processes are typically used to generate an initial process model which is then refined after the process unit comes online. Supply chain reconciliation seeks to align a model to the available inventories, capacities, and constraints [94]. Dynamic data reconciliation is a measurement estimation technique that uses large scale dynamic models, with a time horizon ranging from hours to years, to reconcile measurements with model predictions [95]–[99]. It is often used in conjunction with dynamic optimization to align a separate control model

with dynamic data [100]. For RTO applications, a precursor step is to adjust fouling factors, tray efficiencies, and other parameters with parameter estimation [88]. This parameter estimation may include single or multiple steady state snapshots or the process measurements. One restriction is that the process must be at steady state for the parameter estimation. Finally, MHE is a multi-variable approach for optimal measurement reconciliation in a dynamic model [101]. MHE applications are typically performed on a time scale faster than that of the process time constant of interest. It typically executes in the range of seconds to minutes and can be used to provide state and parameter updates to MPC applications.

3.1.2 Overview

This chapter is a review of strategies to incorporate measurements in optimization and monitoring applications. The mathematical models used in these applications have unmeasured or unmodeled disturbances that cause the model predictions to drift from actual values. This realignment of model and measurement can occur with a variety of techniques ranging from simplified to complex. When the application provides information in real-time, the results must be returned within a specified cycle time. Details on efficient implementation of the techniques are also presented in this chapter with two motivating applications.

The focus of this chapter is on measurement reconciliation for fast time processes in the range of seconds to minutes. New and established techniques are discussed that improve the information extraction from measurements to allow a fundamental understanding of a process.

3.2 Numerical Solution with Dynamic Models

The approach taken in this chapter uses a simultaneous solution method as opposed to a sequential method to solve the model equations and objective function [102], [103]. The general model form consists of nonlinear DAEs in open equation format as shown in Equation 3.1.

$$\begin{aligned} 0 &= f\left(\tfrac{\partial x}{\partial t}, x, u, d\right) \\ 0 &= g(y, x, u, d) \\ 0 &\leq h(x, u, d) \end{aligned} \tag{3.1}$$

The optimizer calculates future states in the horizon that are uniquely specified by the initial state x_0, a given sequence of inputs $u = (u_0,u_1,\ldots,u_{k-1})$, and a calculated set of disturbances $d = (d_0,d_1,\ldots,d_{k-1})$. In Figure 3.4, u and d are shown as discrete values over the horizon. Variables calculated from differential and algebraic equations are continuous over the time horizon.

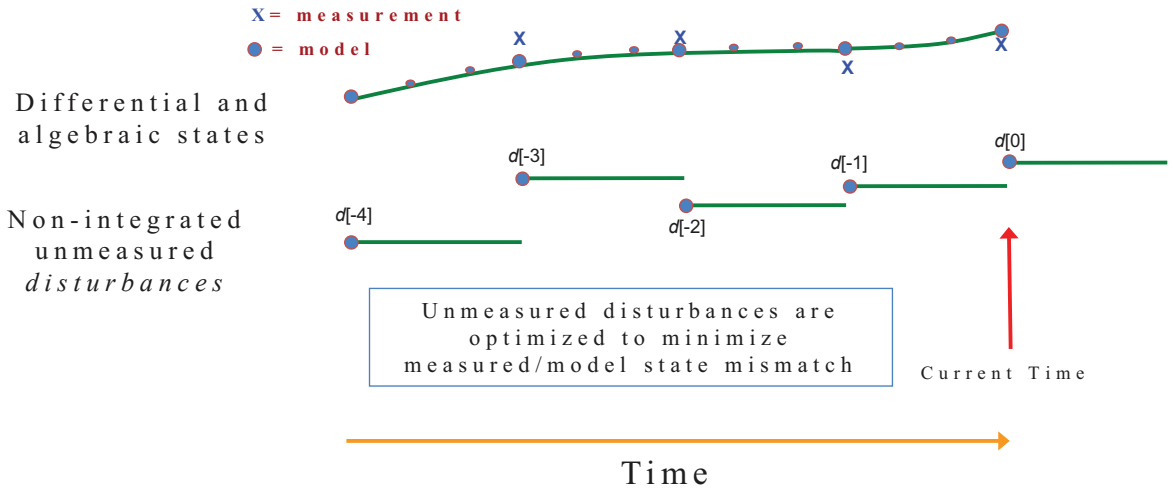


Figure 3.4: Dynamic equations are discretized over a time horizon and solved simultaneously.

The solution of the open equation system is accomplished by converting the differential terms to algebraic equations with orthogonal collocation on finite elements [104], also known as direct transcription [105]. Order reduction may assist in understanding the most important states that dominate the system dynamics [106], but, the full system can typically be solved directly.

The solution of the estimation problem is solved with an implicit solution technique such as large-scale NLP solvers [95], [107]. Other methods include the direct shooting approaches [108] or the explicit solution [109], [110] for simplified problems. The difference between competing implicit solution techniques is how the state equations are satisfied. Direct single or multiple shooting solves the state equations to a convergence tolerance for every iteration. Using orthogonal collocation on finite elements, the state equations are only satisfied at a converged solution. This generally leads to a more efficient solution, especially for large-scale problems with many decision variables [16].

3.3 Review of Current Strategies

Advanced Process Control (APC) has produced significant benefits in the oil and gas industry, especially in refining and chemicals and more limited in exploration and production [111], [112]. Simpler control applications such as PID controllers are often preferred in most single-input, single-output controllers. Measurement reconciliation also ranges from simple to complex [113]. Simple techniques include filtered bias updates or IDF. More complex strategies include Kalman filtering and MHE. Each of these techniques are discussed below.

3.3.1 Filtered Bias Update

A predominant approach for measurement feedback into many of the popular APC commercial packages continues to be a filtered bias update [111]. Filtered bias update is also commonly known as a low pass filter or an α filter. Adding an output constant or integrating disturbance introduces insignificant computational overhead and is easy to tune. In the case of a constant disturbance, an additive model bias b is updated at iteration k with a filter α as shown in Equation 3.2.

$$b_k = \alpha\,(z-y) + (1-\alpha)\,b_{k-1}, \quad 0 \leq \alpha \leq 1 \tag{3.2}$$

In this case, the difference between the measured state z and the predicted model y is used to update the offset of a controlled variable initial condition. With a weak filter with α near 1, almost all of the measurement value is accepted for updating the model predicted value. Strong filters that accept less of the measured value may cause the corresponding APC application to respond slowly to unmodeled disturbances. The value of α is typically chosen to balance noise rejection with speed of reaction. The strengths and drawbacks of the filtered bias update are summarized in Table 3.1.

In order for the bias to be updated, certain qualifications may also be set to detect bad measurements. These qualifications are commonly upper and lower validity limits as well as a rate of change validity limit. The validity limits are applied to either the raw measurement or the raw bias. If any of the validity limits are violated, the measurement is rejected and the bias value remains constant. Rate of change validity limits are frequently set too restrictively for process

Table 3.1: Filtered bias update trade-offs

Strengths of the Filtered Bias Update
Incorporated with many popular APC commercial packages
Single tuning parameter, α, that balances noise rejection with measurement tracking speed
Insignificant computational overhead
Drawbacks of the Filtered Bias Update
No capability to estimate parameters or unmeasured disturbances
No consideration of multivariable effects
Offset is present for integrating disturbances
Physical constraints may be violated

upset conditions such as shutdown or startup, necessitating the need for operator intervention or automatic application switching to manual control.

3.3.2 Implicit Dynamic Feedback

IDF estimates unmeasured disturbances related to the predictions of the measured state variables. IDF pairs a single measurement with a single unmeasured disturbance variable. The analogy to control is the Single Input, Single Output (SISO) controllers such as the ubiquitous PID controller. In the case of IDF, the unmeasured disturbance variable is adjusted to align the model with a process measurement. IDF consists of two equations that can be solved simultaneously with the control problem over a preceding horizon interval.

The IDF equations are similar to a PI controller. The IDF input is the difference between the measured state z and model state y. This is similar to a PI controller with a $SP = z$ and $CV = y$. The output is an unmeasured Disturbance Variable (DV) of the model, d, and is analogous to the MV of a PI controller. This DV is adjusted proportionally to the current and integrated measurement errors, as shown in Equation 3.3a. However, Equation 3.3a is not implemented in practice because of the integral term. To overcome this, the integral term 'I' is differentiated, and the IDF equations are solved as two separate expressions (see Equation 3.3b).

$$d = K_c\,(z-y) + \frac{K_c}{\tau_I}\int_{t=0}^{T}(z-y)\,dt \tag{3.3a}$$

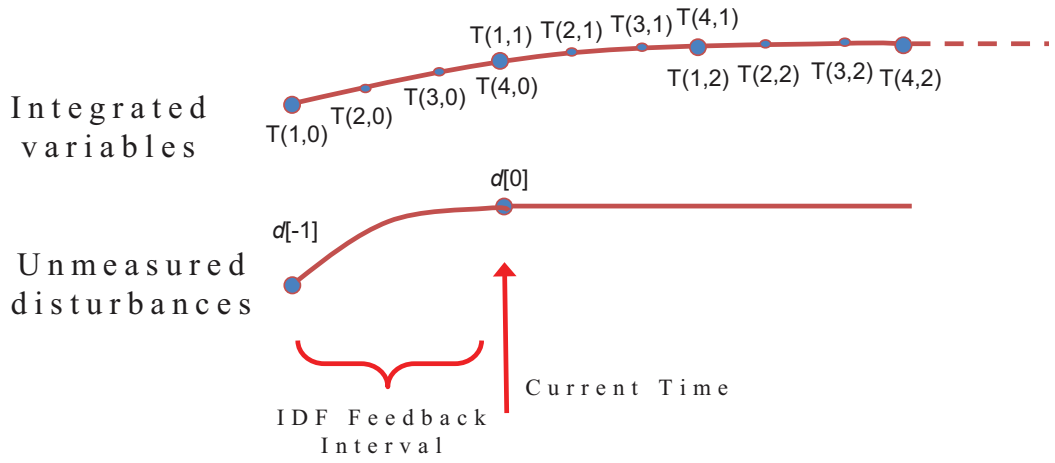


Figure 3.5: IDF horizon with simultaneous estimation and dynamic optimization.

Table 3.2: Implicit dynamic feedback trade-offs

Strengths of IDF
Only two differential equations are required to implement IDF
Similar tuning to a PID controller
Two intuitive parameters trade-off speed versus stability
Drawbacks of IDF
Restricted to one-to-one pairing of a measurement to an unmeasured disturbance
Potential wind-up of the integral term
One step estimation horizon makes it unsuitable for predictive applications (e.g. MPC)
Physical constraints cannot be enforced

$$d = K_c\,(z-y) + \frac{K_c}{\tau_I} I, \quad \frac{\partial I}{\partial t} = (z-y) \tag{3.3b}$$

The tuning parameters for IDF are K_c and τ_I, the same as a PI controller. Using a large value of τ_I and small K_c has the effect of heavily filtering the error term for feedback. In this case the algorithm will take longer to match the plant. Using these tuning parameters and knowing the quality and types of measurements enables trading off between *speed of tracking the process* versus *stability concerns*. The advantages and disadvantages of IDF are listed in Table 3.2.

IDF has been successfully used for many years to provide on-line estimation measurement biases, catalyst activities, kinetic parameter adjustment factors and heat transfer coefficients.

However, IDF is limited to a past horizon length of one, pairing of only one measurement to one disturbance, and the inability to handle constraints.

3.3.3 Kalman Filter

With a Kalman filter, sequential measurements are used to obtain the state of the system with a linear model. To obtain this model, Jacobian information from Equation 3.1 is rearranged into the discrete state space form (see Equation 3.4) where A, B, C are constant matrices, u is the input variable vector, x is the state vector, y is the vector of model outputs. In this case, the subscript k refers to the time step at which the model is computed.

$$x_{k+1} = Ax_k + Bu_k \tag{3.4a}$$

$$y_k = Cx_k \tag{3.4b}$$

The horizon of measurements is combined mathematically to generate the system's state at the current time with the Kalman filter as shown in Equation 3.5. The Kalman filter is divided into 4 subsets of equations. In Equation 3.5a the states $\bar{x}$ and covariance $\bar{P}$ are predicted in the absence of new measurement information. In the next step (see Equation 3.5b), the predictions are compared to the measured values. The innovation $\tilde{\delta}$ and innovation covariance S are the comparison of the model predictions to the measured reality. The innovation covariance S and covariance prediction $\bar{P}$ are then used to calculate the Kalman gain K in Equation 3.5c. As a final step, the new state and covariance estimates are computed in Equation 3.5d. The Kalman gain relates the fraction of the innovation $\tilde{\delta}$ and state prediction $\bar{x}$ that are used to construct the new state estimate x_k. Similarly, the Kalman gain relates the predicted covariance prediction to the new covariance prediction. Note that the covariance update is independent of the measurement values z_k and the time evolution is only a function of constant matrices.

$$\begin{aligned} \bar{x} &= Ax_{k-1} + Bu_k \\ \bar{P} &= AP_{k-1}A^T + Q \end{aligned} \tag{3.5a}$$

Table 3.3: Kalman filter trade-offs

Strengths of Kalman Filtering
Optimal estimator for linear systems without constraints
Solution approach is accomplished through matrix multiplications
Covariance estimate provides confidence interval for state estimate
Drawbacks of Kalman Filtering
Restricted to linearized model state updates
Physical constraints cannot be enforced
Gating methods needed to use infrequent or variable delay measurements

$$\begin{aligned} \tilde{\delta} &= z_k - C\bar{x} \\ S &= C\bar{P}C^T + R \end{aligned} \tag{3.5b}$$

$$K = \bar{P}C^T S^{-1} \tag{3.5c}$$

$$\begin{aligned} x_k &= \bar{x} + K\tilde{\delta} \\ P_k &= (I - KC)\,\bar{P} \end{aligned} \tag{3.5d}$$

The Kalman filter is optimal for unconstrained, linear systems subject to known normally distributed state and measurement noise [114]. The Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF) are an attempt to extend these techniques to nonlinear systems. A summary of the trade-offs related to the Kalman filter are listed in Table 3.3.

EKF is able to predict the nonlinear state evolution by re-linearizing the model at each time instant. Some effort has been made to incorporate constraints with EKF although the state augmentation strategy for parameter estimation is still a limitation [115]. Kalman based techniques suffer from a number of limitations. For nonlinear or constrained systems, optimization techniques such as MHE are better suited to providing an estimate of the true system state.

3.3.4 Squared-error MHE

MHE outperforms EKF in the presence of constraints [114]. Recent advances in computational capability and methods have improved the application of MHE to fast [116] and large-scale

Table 3.4: Trade-offs for MHE with a squared-error objective

Strengths of MHE (squared-error)
Least squares is an intuitive objective that is simple to implement
Model constraints can be added to model to improve the estimation accuracy
Optimal tuning has been established [123]
Drawbacks of MHE (squared-error)
Poor rejection of outliers or infrequent bad values common with real data
Difficult to obtain good estimates of P_0, Q, and R
Dense tuning matrices impractical for large-scale systems
Iterative optimization solution that may fail to converge in the required cycle time

industrial systems [117]. Just as APC has demonstrated significant benefits by considering multivariate relationships [118], MHE is better able to utilize measurements and deliver a more accurate description of the current state of the process and disturbances [119].

By using an optimization framework, the model and measurement values are aligned and present detailed information about the system dynamics. This optimization framework uses a receding horizon of process measurements. MHE attempts to optimally estimate the true state of the dynamic system, given a real-time stream of measurements and a model of the physical process. Offset free estimation and control is achieved by adding as many disturbance variables as the number of measurements [120]–[122]. The MHE objective function is posed as a squared-error minimization to reconcile the model with measured values. The trade-offs for MHE with a squared objective function are summarized in Table 3.4.

In a MHE form amenable to real-time solution, the unmeasured DVs, d, are adjusted to match the continuous model to discrete measured values [117].

$$\begin{aligned} \min_{d} \Phi &= \left\| \tfrac{z-y}{y} \right\|_{Q_y}^2 + \left\| d - \hat{d} \right\|_{Q_d}^2 \\ \text{s.t.} \quad 0 &= f(\dot{x}, x, u, d) \\ 0 &= g(z, x, u, d) \\ 0 &\leq h(x, u, d) \end{aligned} \tag{3.6}$$

In Equation 3.6, Φ is the objective function value, z is a vector of measurements at all nodes in the horizon $(z_0,\ldots,z_k)^T$, y is a vector of model values at the sampling times $(y_0,\ldots,y_k)^T$, Q_y is the inverse of the measurement error covariance, f is a vector of model equation residuals,

x represents the model states, u is the vector of model inputs, d is the vector of model parameters or unmeasured disturbances, $\hat{d}$ is the vector of prior unmeasured disturbances, Q_d is a matrix for the weight on changes of disturbance variables, g is an equality constraint function, and h is an inequality constraint function.

A graphical representation of the MHE squared-error reconciliation is shown in Figure 3.6. The objective for this measured value is a quadratic function with the minimum target between the previous model and measured values.

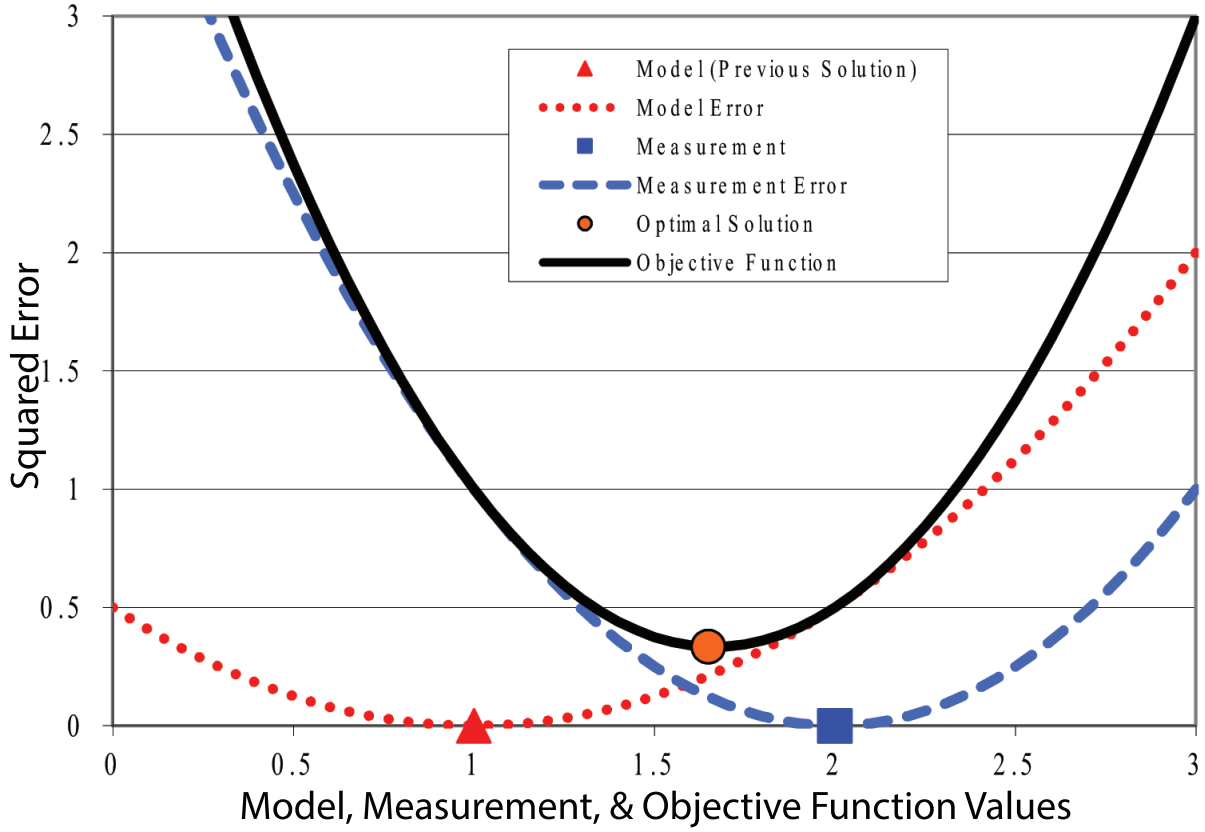


Figure 3.6: Graphical representation of the squared-error for a single measurement in the horizon.

The full estimation problem allows violation of the state constraints [119]. State equality constraints are relaxed and violations are penalized in the objective function. Without d the optimization problem found in Equation 3.6 does not allow state transition error because the state equations are exactly satisfied at a converged solution [124]. This can be overcome by creating a

discontinuous state y_d and disturbance d_d with an additional equation $y_d = x + d_d$ for each state subject to state noise. This allows discontinuities in the y_d states while preserving the continuity of the x states. However, allowing state noise is undesirable when employing first principles models. For material and energy balances, allowing state noise reduces the predictive potential of the model. Instead, only the decision variables are selected as x_0 and d_d instead of (x_0,...,x_k, P) as in the full MHE problem.

As the estimation horizon increases, the sensitivity of the solution at x_k to x_0 decreases. With a first-order approximation, the value of the final state x_k sensitivity decreases by $e^{-\frac{t}{\tau_p}}$ where τ_p is the approximate process time constant. For sufficiently long time horizons, it is then only d that has a significant effect on the current model state. Thus, it is generally unnecessary to estimate the initial states x_0 as degrees of freedom in the optimization problem.

3.3.5 ℓ_1-Norm MHE

A new form of MHE has been used in industry for a number of years that overcomes some of the limitations of the squared-error MHE approach [6]. The objective function in Equation 3.7 is implemented in a form that is amenable to numerical solutions of large-scale models. The use of an absolute value function is avoided by instead solving inequality constraints with slack variables. The slack variables and inequalities create an objective function that is smooth and continuously differentiable as a requirement for large-scale NLP solvers.

$$
\begin{aligned}
\min_d \Phi &= w_m^T (e_U + e_L) + w_p^T (c_U + c_L) \\
\text{s.t.} \quad 0 &= f(\dot{x}, x, u, d) \\
0 &= g(y, x, u, d) \\
0 &\leq h(x, u, d) \\
\\
e_U &\geq y - z \\
e_L &\geq z - y \\
c_U &\geq y - \hat{y} \\
c_L &\geq \hat{y} - y \\
e_U, e_L, c_U, c_L &\geq 0
\end{aligned}
\tag{3.7}
$$

Here, Φ is the objective function value, z is a vector of measurements at all nodes in the horizon $(z_0,\ldots,z_k)^T$, y is a vector of model values at the sampling times $(y_0,\ldots,y_k)^T$, $\hat{y}$ is a vector of previous model values at the sampling times $(\hat{y}_0,\ldots,\hat{y}_k)^T$, w_m is a vector of weights on the model values outside a measurement dead-band, w_p is a vector of weights to penalize deviation from the prior solution, f is a vector of model equation residuals, x represents the model states, u is the vector of model inputs, d is the vector of model parameters or unmeasured disturbances, g is an output function, h is an inequality constraint function, e_U and e_L are slack variables to penalize model values above and below the measurement dead-band, and c_U and c_L are slack variables to penalize model value changes above and below the previous values.

A graphical representation of the MHE ℓ_1-norm reconciliation is shown in Figure 3.7. Parameters are only adjusted if the measured value is more than the half of the dead-band away from the previous model value. Otherwise, the model is not adjusted because the measurement lies within the region of a flat objective function. In the case of Figure 3.7, the optimal solution lies at the edge of the measurement dead-band. This will always be the case for measurements that are more than half the dead-band distance from the prior model value.

The MHE ℓ_1-norm objective has a number of advantages and challenges compared with other methods such as the Kalman filter or the MHE squared-error. The next section details the trade-offs between the different techniques.

MHE ℓ_1-Norm Advantages

An important MHE ℓ_1-norm advantage is less sensitivity to data outliers, noise, and measurement drift [6]. This is important when dealing with industrial data where instruments drift or fail. Gross-error detection can eliminate a majority of bad data. With MHE ℓ_1-norm, any data that isn't filtered by gross-error detection has less impact on the parameter estimation and allows improved reliability of the solution. A squared-error objective is more sensitive and disproportionately weights values that are far from the model predictions.

An additional advantage of the MHE ℓ_1-norm is that only linear equations are added to the objective function. Without additional nonlinear expressions, the solution is generally easier for numerical solvers to find an optimal solution. In summary, the MHE ℓ_1-norm optimization problem with measurement noise dead-band has a number of advantages over the MHE squared-

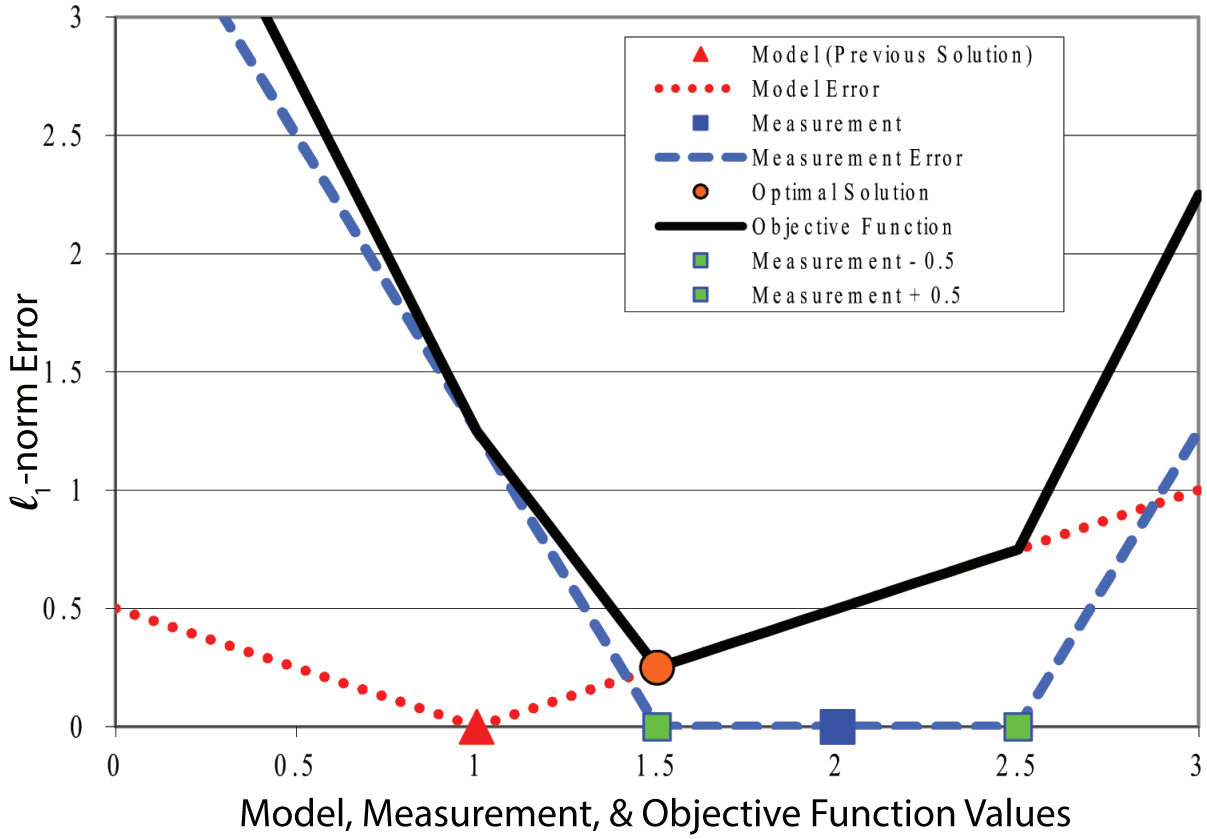


Figure 3.7: Graphical representation of the MHE ℓ_1-norm for a single measurement in the horizon.

error form of the objective function. The trade-offs for MHE with an ℓ_1-norm objective function are summarized in Table 3.5.

MHE ℓ_1-Norm Challenges

The challenges with the MHE ℓ_1-norm optimization problem include increased complexity and size. Although the MHE ℓ_1-norm uses only linear expressions in formulating an objective function, there are additional slack variables and inequality expressions, which increases the size of the optimization problem.

Many of the MHE ℓ_1-norm challenges are due to the increased complexity in the solution techniques. Commercial and academic software has been developed to meet this challenge. The software used to generate the results in this chapter is the APMonitor Modeling Language [6].

Table 3.5: Trade-offs for MHE with an ℓ_1-norm objective

Strengths of MHE (ℓ_1-norm)
Low sensitivity to occasional bad data
Linear objective function and sparse tuning techniques improve scaling to large-scale systems
Explicit measurement dead-band for improved noise rejection
Drawbacks of MHE (ℓ_1-norm)
Additional equality and inequality constraints and variables, leading to increased computation time
No optimal theory on best tuning parameters
Requires an iterative solver to reliably converge in a specified cycle time

Filtered bias updating, Kalman filtering, IDF, and MHE are implemented in this web-services platform through MATLAB or Python.

3.4 Managed Pressure Drilling Flow Estimation

As an example application, consider the problem of determining the flow of mud through the return annulus of a drilling pipe. In the return line, there is typically a flow paddle that rotates proportional to the flow rate. This flow paddle measurement is not very accurate so additional information such as pit tank level can be used to infer the return flow. Additionally, in MPD, a choke valve is adjusted to maintain well pressure as shown in Figure 3.8.

The flow, pressure, and level measurements have noise, creating random fluctuations around the true values. The flow through the choke valve can also be estimated from the valve position and differential pressure across the valve (see Equation 3.8).

$$\tau_p \frac{\partial q_{choke}}{\partial t} + q_{choke} = K_1 f(l) \sqrt{\left(\frac{\Delta P_v}{g_s}\right)} \tag{3.8}$$

For this example, the installed characteristic of the choke valve is assumed to be linear ($f(l) = l$) and the valve is fast acting ($\tau_p = 1\ sec$). Both the state and measurement noise are normally distributed with mean values of zero (see Figure 3.9). State noise has a standard deviation $\sigma_q = 0.1$ and measurement noise has a standard deviation $\sigma_r = 1.0$.

The Kalman filter updates the state estimates by operating in two phases: predict and update. In the prediction phase, the calculated flow is modified according to the equation that relates

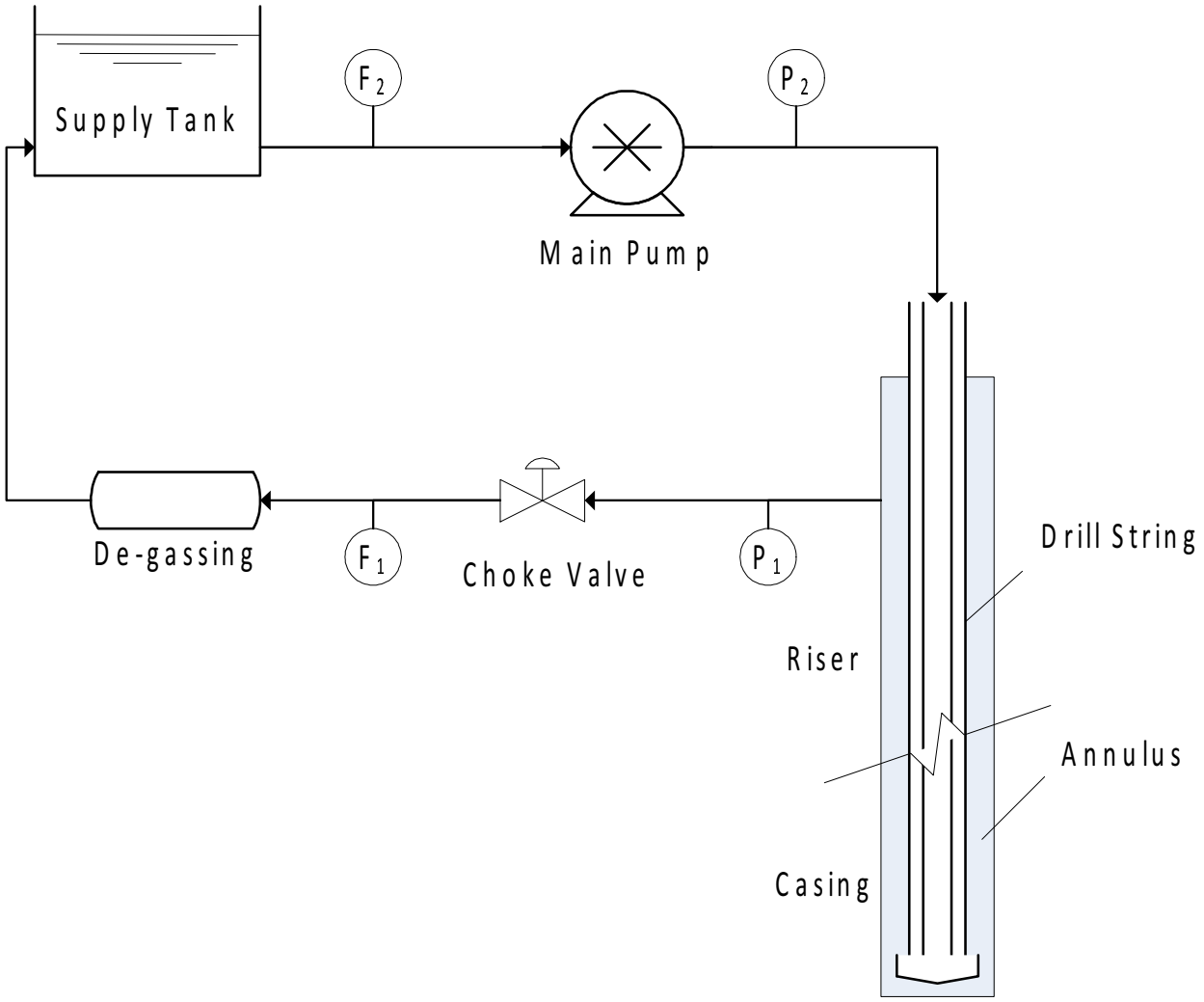


Figure 3.8: Schematic of Managed Pressure Drilling.

flow q_{choke} to the lift function $f(l)$ and the differential pressure, ΔP_v. For Kalman filters, the equation must first be linearized. With Extended Kalman filters, the nonlinear equations are re-linearized about the current state estimate. The other parameters, including τ_p, K_1, and g_s, are constants for a particular valve and fluid. For systems with multiple measurements, the covariance is used to tune the Kalman filter. In this case with one measurement, the variance is used instead. This information is essential for optimizing the update phase; yet state and measurement covariance information can be difficult to obtain. The results of the Kalman filter with the upper and lower 95% confidence intervals are shown in Figure 3.10.

In the update phase, a measurement of the flow is taken from the transmitter. Because of the noise, this measurement has some uncertainty. The calculated variance from the predict

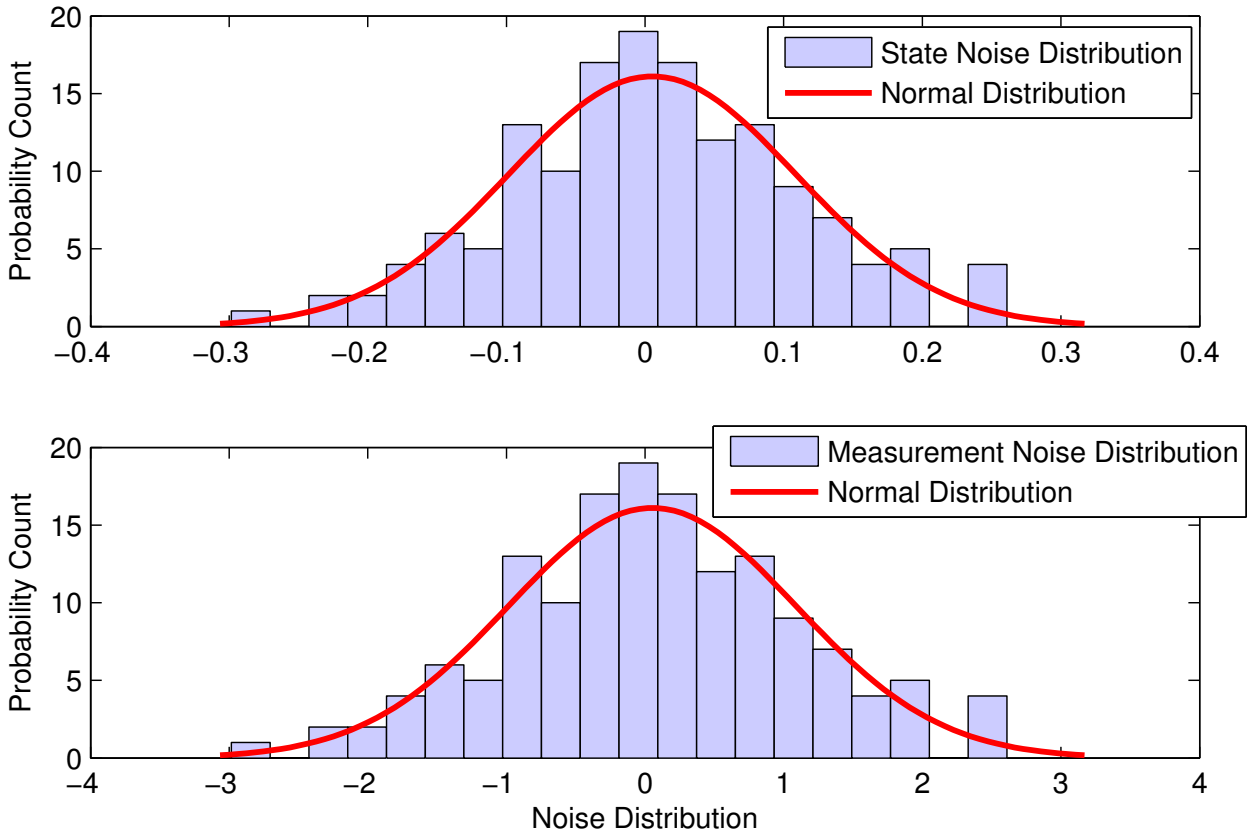


Figure 3.9: Noise distributions of state and measurement noise. These distributions are used to optimally tune the estimators.

phase determines how much the new measurement affects the updated prediction. If the model prediction drifts away from the real flow, the measurements from the flow transmitter should pull the flow estimate back towards the real flow but not disturb it to the point of introducing all of the noise from the measurement.

This model update could also employ other measurements such as mud pump speed, choke pressure, or supply tank level to infer the flow rate across the valve. For this simple example, only the valve position and flow measurements are used to predict the flow with a linear, first-order correlation. Each of the five techniques discussed in this chapter are compared over the same data set as shown in Figure 3.11.

The filtered bias update and IDF have been tuned to give equivalent responses. After an initialization period, they also align exactly with the Kalman filter results because the Kalman gain becomes constant after the estimate of the covariance matrix P_k also converges to a constant value.

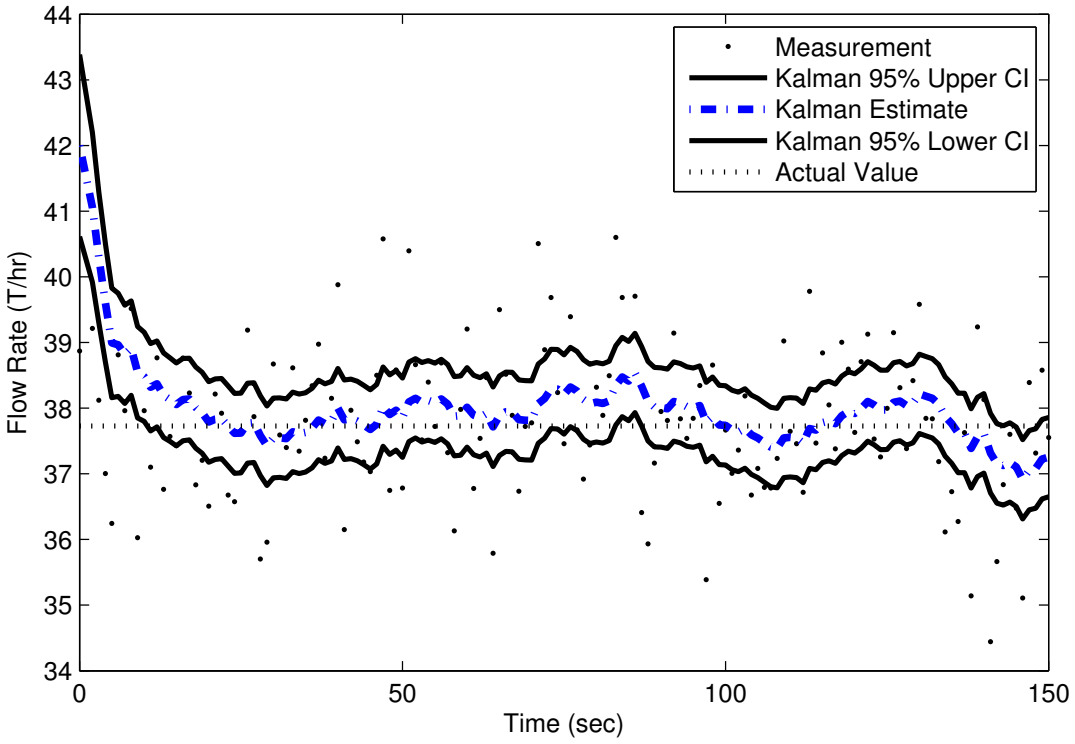


Figure 3.10: The Kalman filter uses two phases, predict and update, to obtain an estimate of the true flow. During the predict phase, the model calculates an updated flow due to the latest reported model inputs. During the update phase, part of the flow measurement is used to update the state, inversely proportional to the variance of the measurement error.

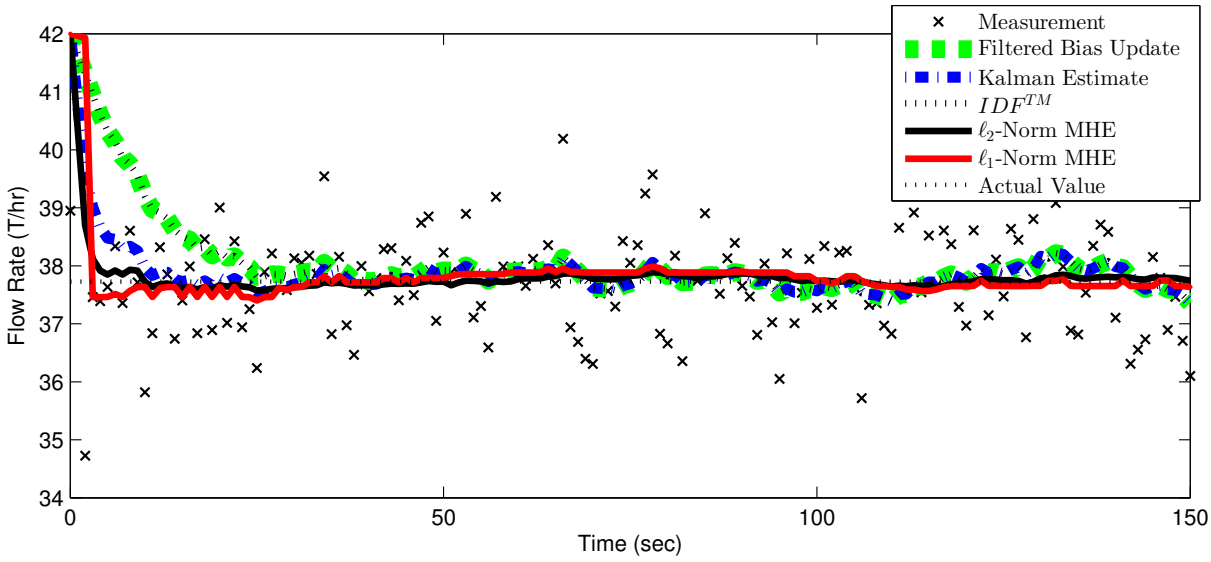


Figure 3.11: Actual, measured, and estimated flows for filtered bias update, IDF, the Kalman filter, squared-error MHE, and ℓ_1-norm MHE.

Table 3.6: Estimator configuration values

Estimation Method	Example Tuning	Equivalent Tuning for One Measurement
Filtered Bias Update	$\alpha = 0.0951$	Set α equal to the Kalman Gain K
IDF	$K_c = 0.0951e-10, \tau_I = 1e-10$	Set $\frac{K_c}{\tau_I}$ equal to the Kalman Gain as $K_c \rightarrow 0$
Kalman Filter	$P_{k=0} = 0.5, Q = 0.01, R = 1.0$	Set $P_{k=0} = P_{k=\infty}$ for equivalency to other methods during initialization
squared-error MHE	Horizon = 50, $Q_y = 100$, $Q_d = 10$	For linear systems with quadratic objective MHE reduces to Kalman Filter [101]
ℓ_1-Norm MHE	Horizon = 50	The ℓ_1-norm MHE does not have equivalent tuning correlations to the other methods

The first four methods including filtered bias update, IDF, the Kalman filter, and the squared-error MHE (with one horizon step) can be tuned to give equivalent results for this single measurement case. Table 3.6 shows the tuning values that make each of the estimators equivalent for this example case and in general.

In addition to signal loss, real data often contains bad data such as outliers, drift, and noise. Outliers do not typically fit a standard normal distribution but are instead drastic deviations from normal variation in the data. Outlier detection and removal is typically accomplished by setting rate of change limits, upper validity limits, and lower validity limits. This gross error detection eliminates many, but not all, of the data outliers. The effect of data outliers is shown in Figure 3.12 with the introduction of an outlier at cycle 50, drift starting at cycle 100, and increased noise starting at cycle 150.

The results with bad data with an outlier, drift, and noise clearly indicate that all state estimates, except the ℓ_1-norm MHE, are significantly affected by the bad data points. The insensitivity to bad data is a key advantage of the ℓ_1-norm MHE approach. This insensitivity is due to the longer time horizon used in MHE, and because the ℓ_1-norm does not square the error term. Squaring the error artificially amplifies bad data.

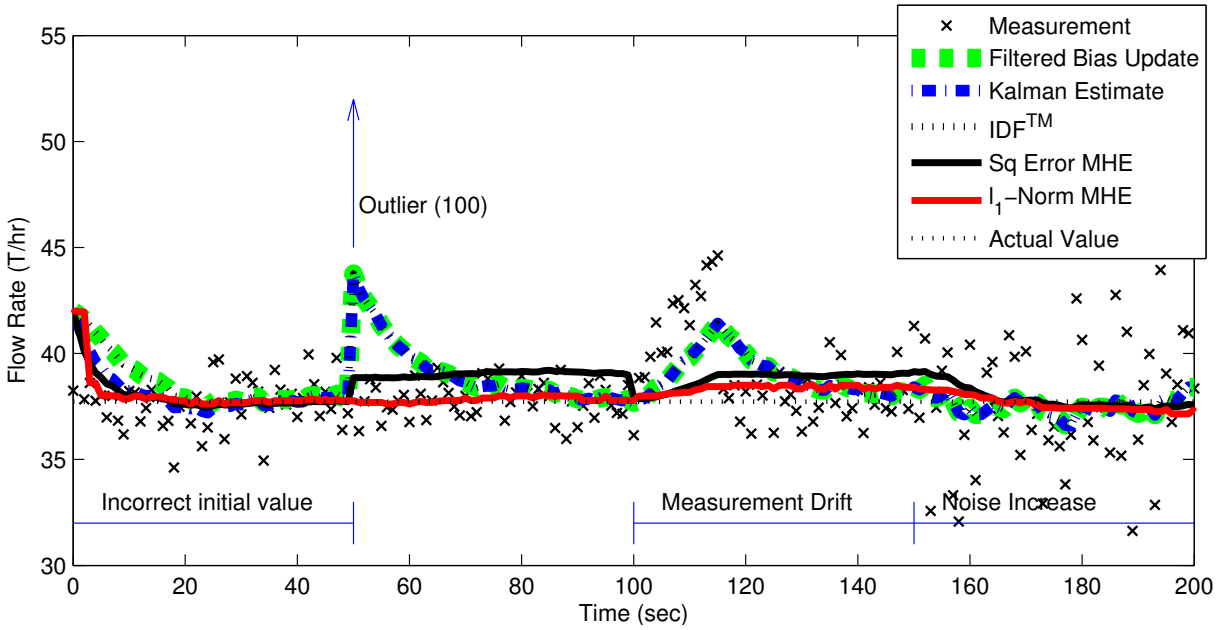


Figure 3.12: Outlier effect on the filtered bias update, IDF, the Kalman filter, squared-error MHE, and ℓ_1-norm MHE. The ℓ_1-norm MHE is least sensitive to brief periods of bad data.

3.5 Estimation for Control Relevant Models

One important function of estimation methods is to improve the predictive qualities of a model prior to forward prediction methods such as MPC. Process gains and time constants are often used to characterize the relationship between a MV and associated CV in control applications. An issue that occasionally arises in linear MPC applications is that the process conditions change and the model is no longer sufficiently predictive for satisfactory control performance. A simplified MPC application is developed in this section with the objective of establishing guidelines for quantifying controller performance affected by process/model mismatch. This is done by varying the combination of process gains and dynamics in the SISO control model, and evaluating the MPC objective function.

In this MPC application, a first order linear process $(\tau_p dx/dt = -x + K_p\, u)$ with a gain (K_p) of 1.0 and process time constant (τ_p) of 1.0 sec is controlled. The control horizon is set to 4.0 sec with a time discretization of 0.5 sec. The MPC controller minimizes the deviation of the CV from a target value of 5.0, starting from an initial condition of 0.0. The model gain and time constant are changed to incorrect values and the MPC performance is simulated over 20 control cycles. With a 0.5 sec cycle time, there is a total of 10 sec simulated control time. The absolute

value of the deviation from the target set point (5.0) is recorded for each combination of K_p and τ_p values with a range of mismatch of 0.2 to 5.0 for each. Figures 3.13 and 3.14 show the control performance over the range of mismatched models applied in the MPC controller.

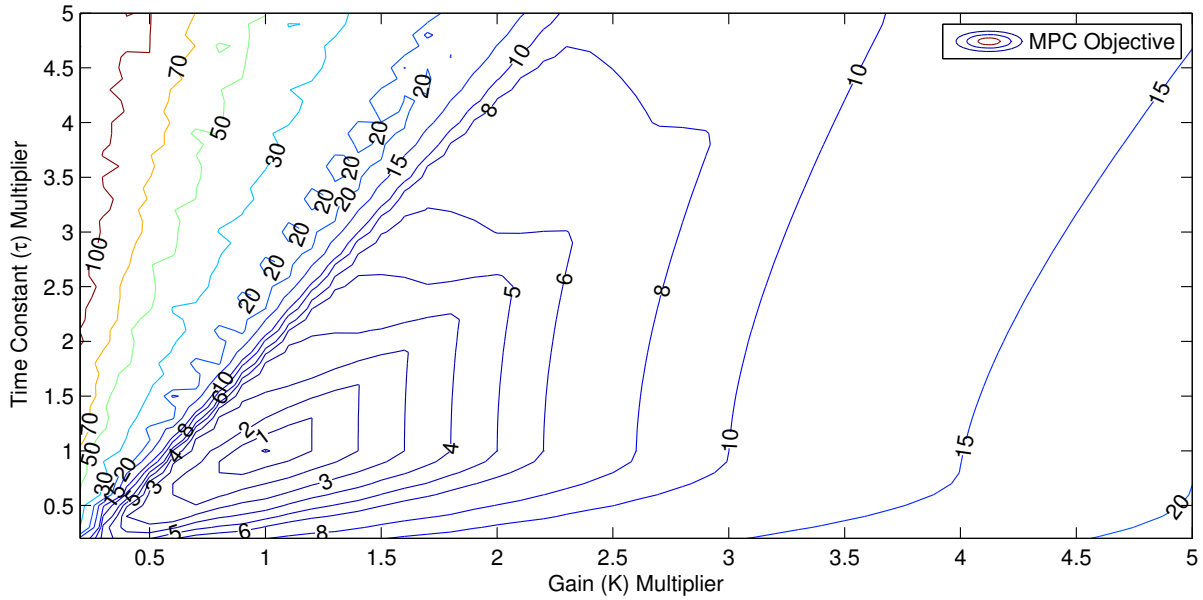


Figure 3.13: Contour plot of the control objective with varying mismatch of the process gain and time constant.

If the model is used in a model predictive controller, a model gain that is lower than the actual process gain may cause controller oscillation and instability. A model gain that is higher than the actual process may cause sluggish control. Likewise, a model response (time constant) that is slower (higher time constant) than the actual process time constant tends to cause controller instability. On the other hand, a model response (time constant) that is faster (lower time constant) than the actual process tends to cause sluggish controller response. The combination of a low gain and high time constant leads to the highest objective function (poor performance and instability). On the other hand, a high gain and low time constant lead to sluggish control, but the controller is generally able to asymptotically drive the process to the correct set point. Put in more quantitative terms, if the time constant is correctly estimated and the gain is overestimated 50%, then the controller error will be 2.5%. Whereas, if the gain is underestimated by 50%, then the controller error is 20%. Similarly, if the gain is correctly estimated and the time constant is over estimated

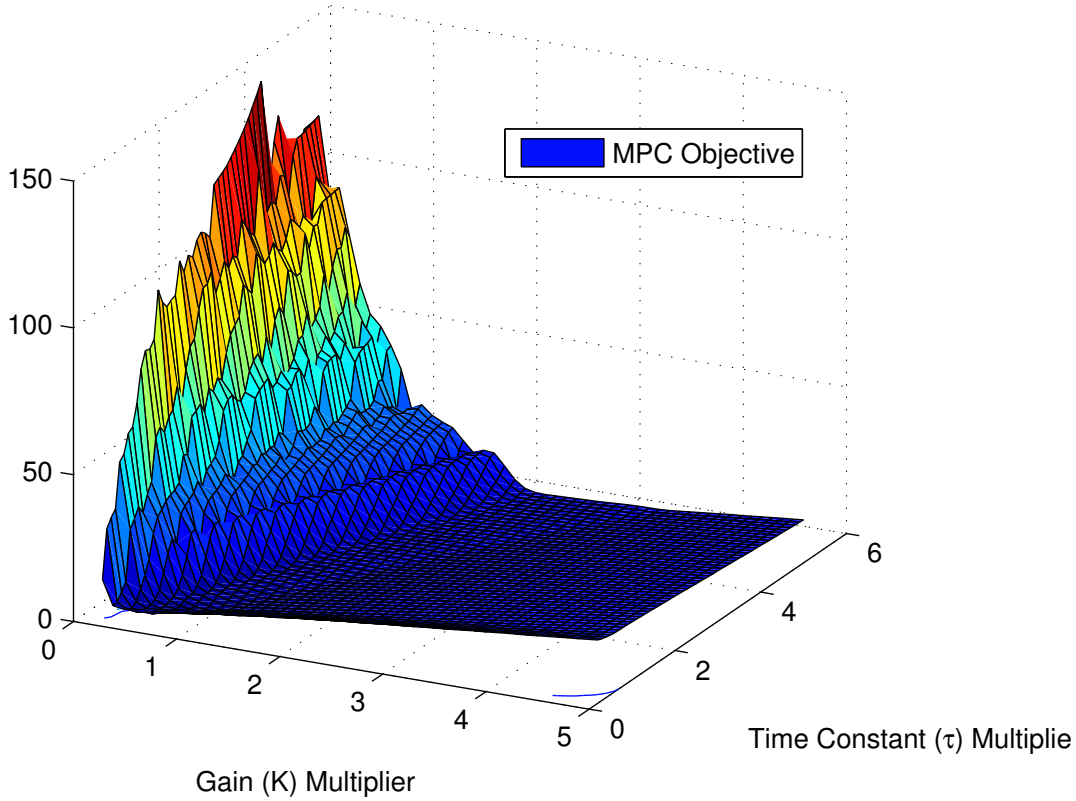


Figure 3.14: Mismatch with too low model gain and too high time constant favor controller instability.

by 100%, then the controller error is 15%, while if overestimated by 100% the error is 8%. These results support the conventional process control wisdom of estimated gain needing to be within 30% of the actual gain , and the time constant being within 50% for sufficiently effective control.

These concepts clarify the effects of control model error on controller performance. Using these guidelines, the controller gain and time constant can be adjusted appropriately until more time intensive model identification techniques can be used. A more rigorous method for increasing confidence in the accuracy of a control model is by using linear or nonlinear confidence regions for estimated parameters [125].

3.5.1 Nonlinear Statistics in Control Model Parameter Estimation

Computing the nonlinear Joint Confidence Region (JCR) for estimated model parameters gives statistical significance to the estimated variable values [126], [127]. The JCR is calculated by solving Equation 3.9.

$$\frac{E(\vartheta) - E(\vartheta^*)}{E(\vartheta^*)} \leq \frac{U}{N-U} F_{N,N-U,1-\alpha_c} \tag{3.9}$$

In Equation 3.9, $E(\vartheta)$ is the error between the process measurements and the control model predictions, $E(\vartheta^*)$ is the error between the process measurements and the model predictions when the best estimates of the model parameters are used. U is the number of model parameters in the regression, N is the number of data points used in the regression, F is the F-statistic at N and $N-U$ degrees of freedom with a confidence level of $1-\alpha_c$.

A simple case study explores the use of nonlinear statistics in online control model parameter estimation. The method is applied to a process similar to Figure 3.8. In drilling an oil or gas well, the drill bit is cooled by the drilling fluid, or mud, which also moves the rock cuttings to the surface and maintains pressure in the well annulus. The well annulus pressure consistently needs to be greater than the geologic reservoir pressure to prevent hydrocarbons from entering the well during the drilling process. If the mud pressure in the well is too high, it can damage the rock formation; if it is too low, hydrocarbons from the subsurface reservoir can come to the surface in an uncontrolled and dangerous manner. The well bore pressure must be maintained within a small range of pressures that balances the reservoir fluid pressure to prevent damaged formations and blowouts. Maintaining the pressure balance in the well is the goal of Managed Pressure Drilling (MPD).

This example uses a MHE estimation scheme with Nonlinear Model Predictive Control (NMPC) to control the wellbore pressure in a simulated oil well using MPD. The controller and the MHE use a reduced order model, developed by Stamnes et. al. [128], which requires an estimation of the annulus friction factor (f_a) and annulus density (ρ_a) at each time step. Figure 3.15 shows the simplified control scheme for this example.

The main mud pump flow rate and the choke valve pressure are manipulated at each time step to drive the pressure at the drill bit to a given set point. Without accurate estimates of f_a and ρ_a the

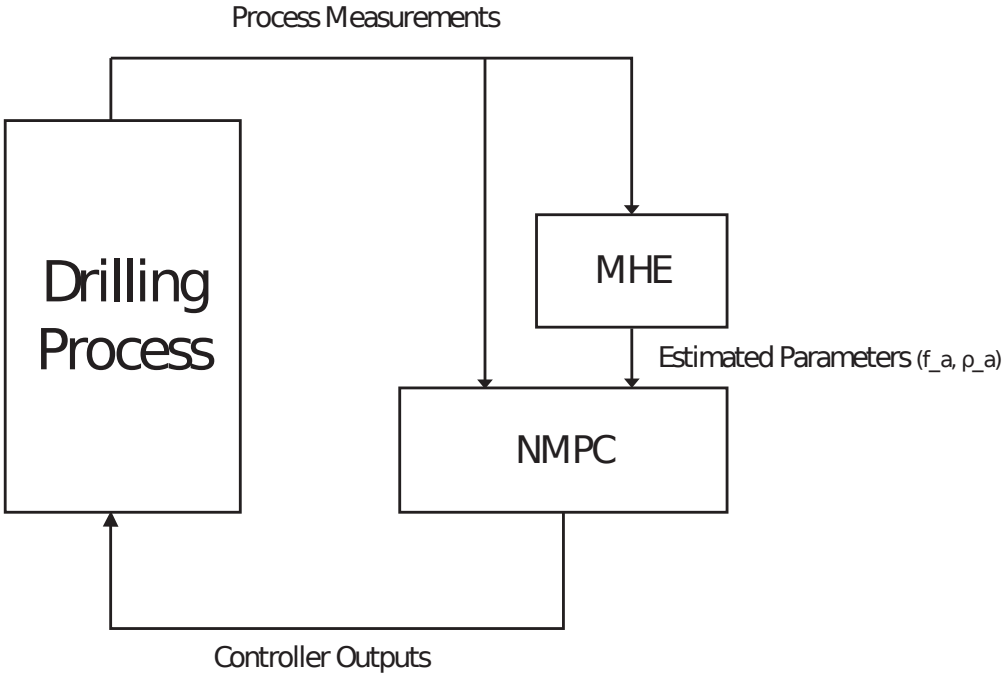
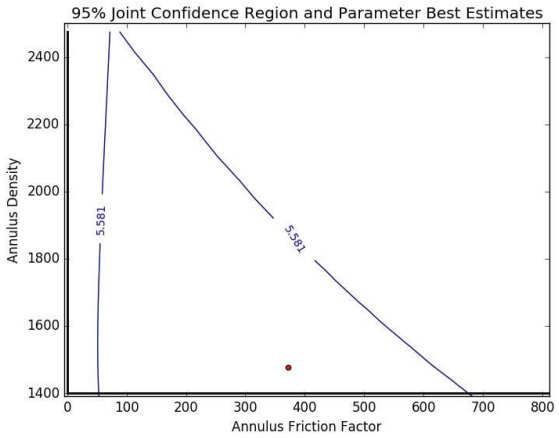


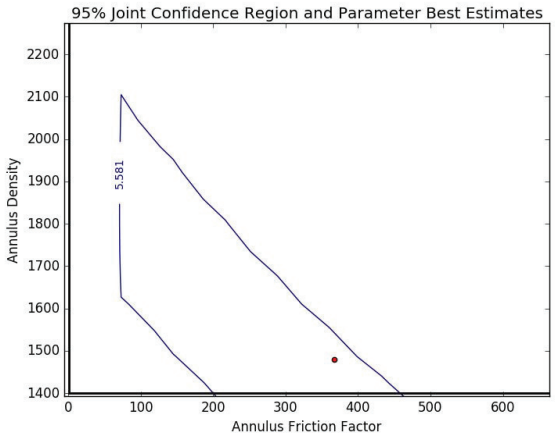
Figure 3.15: Control flow diagram for the MPD controller.

controller can lose control over the process. These two parameters are estimated at each time step using a squared-error MHE scheme. The MHE has an estimation horizon of four steps each eight seconds apart. Over this time horizon, the error between the bit pressure measurement and the model predicted bit pressure is minimized by adjusting f_a and ρ_a in the model. The squared-error is necessary when calculating the JCR because the theory used to develop the nonlinear JCR uses the F-statistic which is formulated using the χ^2 distribution. The χ^2 distribution is founded in the Central Limit Theorem that uses the variance, or square of the standard deviation. A theoretically sound version of the nonlinear JCR based on the ℓ_1-norm has not yet been developed and is left for future work [125].

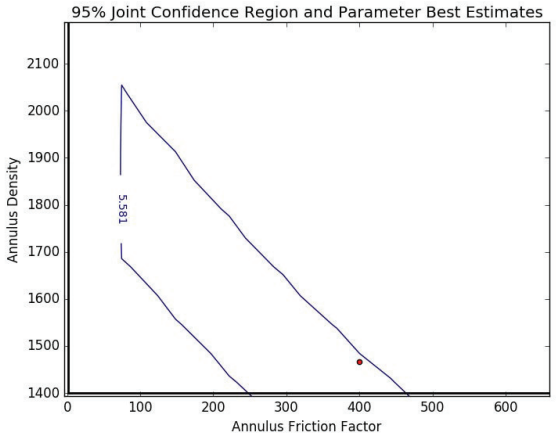
Figure 3.16 shows the nonlinear 95% JCR for the estimated f_a and ρ_a at four time steps in the simulation. Bounds on the estimates were set in the MHE to ensure the estimated variable values were physically realizable. f_a is given a lower bound of 1 m^{-5}, and ρ_a is given a lower bound of the known density entering the drillstring, which in this case is 1400 $\mathrm{kg/m}^3$. In a constrained estimation problem these bounds become the limits of the JCR. For unconstrained estimation, the JCR is defined by all possible solutions for a specified confidence level.



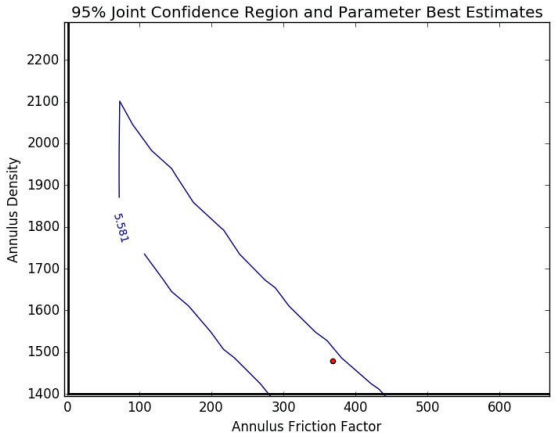
(a) At 2 minutes simulation time.



(b) At 2.3 minutes simulation time.



(c) At 2.5 minutes simulation time.



(d) At 2.8 minutes simulation time.

Figure 3.16: The 95% joint confidence regions for estimated annulus density and friction factor at four different time steps in a MPD simulation. The single contour line denotes the JCR, while the point is the estimated values of the two model parameters. The JCRs have a lower bound of the mud density entering the drillstring. The density units are $\mathrm{kg/m^3}$ and the friction factor units are $\mathrm{m^{-5}}$.

The negative slope of the confidence region suggests the two parameters are strongly correlated. An analysis assuming error in both variables would give further insight into parameter correlation, and would give better insight to the lack of curvature in the JCR. The vertical length of the left bound of the JCR indicates that the simulated measurements from the well contain little information on the annulus density compared to the friction factor. Future work may probe the necessity of the annulus density in the model. However, the first principles-based control model relies on the annulus density for pressure calculations and is theoretically necessary.

It is interesting to note that the subplots in Figure 3.16 show the JCR reducing in area as the simulation progresses. This corresponds to an increase in the confidence of the estimated parameter values. This information can be used to identify problems with the control algorithm, and can possibly be used in controller logic as well. For example, if the estimated parameter value is outside of the JCR, or if the JCR is wide enough that little confidence is given in the estimate, then a model identification procedure could be initiated. The proof of this concept is left to future research. However, the technique has proven to be useful in preliminary evaluations of the control model parameter interactions, and further work is needed to determine the importance of the parameters in the model.

One drawback of the method presented here is the large computation time required to compute the JCR at each time step. This is because at each time step a contour plot is generated to define the confidence region. This adds a large computational burden to an already computationally intensive control scheme. Nevertheless, the computation time can be improved by implementing a parallel processing structure in the controller code.

3.6 Concluding Remarks

There is a recent increase in data availability in the oil and gas industry due to advances in technology, improved networking, and regulatory requirements that require additional monitoring. When measurements are viewed individually they provide insight into the true state of the process, but do not offer a holistic view of the process. When combined with a process model, the data provides an increased understanding of unmeasured disturbances or unmeasured states. This alignment of measurements and model predictions is accomplished with a variety of techniques ranging from a simple bias update to large-scale optimization approaches. Two optimization approaches discussed in this chapter include MHE with ℓ_1-norm and squared errors. Efficient solution of the MHE approach is important for solving large-scale problems of industrial significance. Simultaneous solution of the objective function and model equations is an efficient approach to solving large-scale models for data reconciliation. In many cases, the objective in state and parameter estimation is to obtain a model that is sufficiently accurate for predictive control applications. Model mismatch with a gain that is too low or time constant that is too high may lead to unsatisfactory

control performance. Nonlinear statistical methods can give confidence in parameter estimates and insight to the interactions of control model parameters.

CHAPTER 4. BASIC SWITCHED CONTROL OF MANAGED PRESSURE DRILLING

This chapter is published as: Eaton, A.N., Beal, L.D.R., Thorpe, S.D., Janis, E.H., Hubbell, C., Hedengren, J.D., Nybø, R., Aghito, M., Bjørkevoll, K., Boubsi, R.E., Braaksma, J., van Og, G., *Ensemble Model Predictive Control for Robust Automated Managed Pressure Drilling*, SPE-ATCE, 2015, DOI: 10.2118/174969-MS

4.1 Introduction

To meet the growing demand for energy, more efficient, robust, and reliable technological advances are needed in the petroleum industry. For example, there continue to be an average of 4 uncontrolled well situations in the Gulf of Mexico each year even after the Deepwater Horizon incident and subsequent oil spill [4]. Pressure control is critical to successful drilling and completion to avoid fractured formations and excessive mud loss due to high annulus pressure or gas influx due to an annulus pressure lower than the formation pore pressure. There have been many sensor, communication, and equipment advances in the upstream sector to meet these challenges with technologies such as MPD and Wired Drill Pipe (WDP). With these advances, there is still a lack of robust and stable automation to control pressure during many phases of drilling operation [5]. When automation is applied to more phases of drilling operation, it improves safety and efficiency by responding to process disturbances and by operating within acceptable limits and closer to process economic constraints than possible with traditional control. To attain these benefits, optimizer-based automation and control techniques, such as MPC, require sufficiently accurate models of the process. These models can be obtained from empirical data or from foundational principles such as mass and energy balances. Foundational principles based models that very nearly approximate the actual process are known as high fidelity simulators. High fidelity simulators historically have been implemented in real-time controllers as soft-sensors for PID feedback controllers but the full predictive and multivariate capability has been largely unused because of the excessive computa-

tional requirements of these large-scale models[72]. To address this shortcoming, less accurate first principles models are developed for real-time control purposes. These low-order control models generally express the major process input/output relationships and dynamics, yet the number of equations and variables are significantly reduced, thereby reducing computational requirements. A low-order MPD model was developed by Stamnes et al. [128] and later modified to use bottom hole pressure measurements from wired drill pipe by Asgharzadeh Shishavan et al. [76]. While these low-order models require less computation time than high fidelity models, they can be less likely to converge to a control solution than controllers utilizing empirical based models using MPC. While empirical models can be designed to be more robust against noise than their first-principle counterparts, they require frequent re-tuning as process conditions change. However, it is possible to capture the advantages of each model- high accuracy, reduced computation time, and successful solution convergence by implementing an ensemble control scheme that switches between the various models at the appropriate instance. This work introduces a novel ensemble control structure for MPD that makes use of the strengths of multi-fidelity models− high fidelity, low-order, and empirical. Additionally, the control structure offers the ability to tune and readjust one model while another is used in control without interrupting the drilling process. It also offers the benefit of increased reliability generally associated with redundant hardware and safety systems. The remainder of the chapter is outlined as follows: A description of the MPD automation scenario used in this work, an account of the three controllers, namely the ”high fidelity”, ”low-order” and ”empirical” controllers, a discussion of the mechanism for switching between the controllers, the results of the controller response to both normal drilling and pipe connection procedures, a discussion of the results, and conclusions and further research.

4.2 Managed Pressure Drilling Simulation

MPD is a highly nonlinear process that involves the pressure hydraulics of compressible, non-Newtonian drilling fluid (see Figure 4.1). During MPD the well bore pressure must be maintained within a small range of pressures that will balance the reservoir fluid pressure to prevent both fractured formations and blowouts. In this paper the controllers reach and maintain the desired well pressure at the bit by manipulating the mud pump flow rate and the choke valve opening. Advances in MPD automation have not been ubiquitously implemented; but, it has been demon-

strated that an automated controller can maintain borehole pressure and reject disturbances faster and more accurately than manual control by using NMPC with high quality process data [76]. One of the major challenges with implementing NMPC is the quality of the data sent by the downhole instruments. The most common method of receiving continual pressure measurement data from the BHA is through mud pulsing [129]. Currently, the data transmission rate of mud pulsing is at most 80 bits per second [78]. The small bandwidth and long delay time of mud pulsing technology pose a significant challenge to automating MPD, and improving the technology is an active area of research [130]. In addition to mud pulse telemetry, some researchers and companies are exploring other technologies such as WDP [129] which does not require mud flow to transmit BHA data to the surface. While WDP removes measurement latency, the downhole data quality may be low due to lack of sufficient pressure measurement accuracy and unavailability of data during pipe stand connections. Because poor quality or intermittent data can lead to poor controller performance, data validation and reconciliation are necessary for the predictive models to provide stable feedback control [73]. While many companies and researchers are exploring pressure control using an observer to infer mud flow through the bit or down hole pressure [73], [131]–[134], this work uses a multivariate MPC to control bit pressure. The bit pressure is either measured directly through mud pulse telemetry or by using topside measurements to infer bit pressure when measurements are unavailable or delayed. The advantages of multivariate MPC feedback control over controllers that use PID and bit pressure estimation are reported in [135] and [76]. The predictive power of MPC is demonstrated in this work. However, the techniques used here are most applicable to the variations of MPD described by the SPE Advanced Drilling and Well Technology [136] textbook that include a surface backpressure pump or a closed loop mud system such as Constant Bottom Hole Pressure and Returns Flow Control techniques.

The drilling process is simulated using the SINTEF high fidelity flow model [137] connected to Simulink and MATLAB. The simulated well is offshore in 20 m (65.6 feet) of water with a 0.273 m ($10\frac{3}{4}$ inch) diameter riser and casing. The drill pipe diameter is 0.127 m (5 inch), and a 0.254 m (10 inch) Polycrystalline Diamond Compact (PDC) bit is used. The well has a True Vertical Depth (TVD) of 2150 m and a Measured Depth (MD) of 4300 m. The ROP is 2 m/hr with 100 rad/s for the drillstring rotary speed (RPM). The mud temperature in the drillstring is 35 C, and

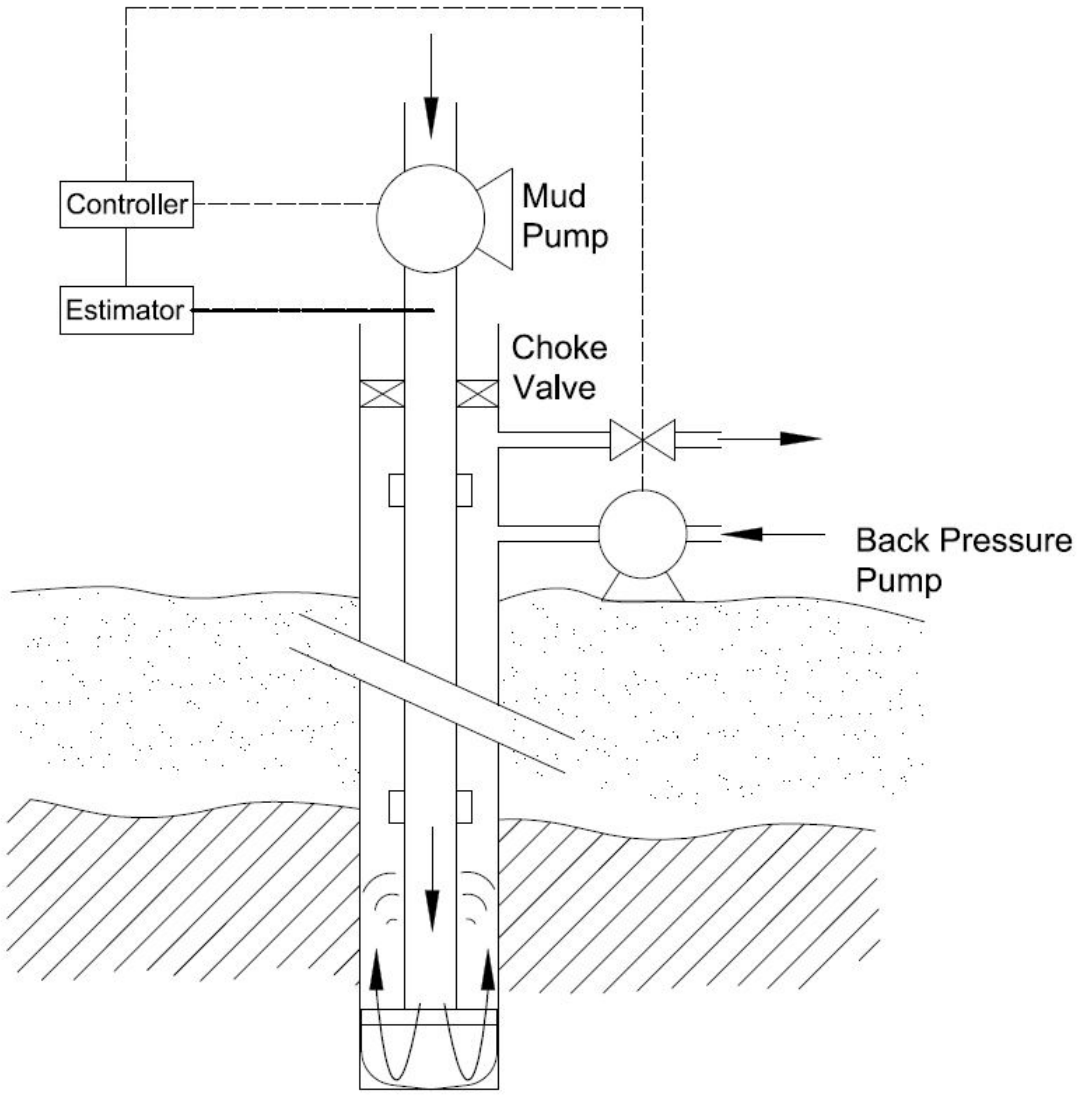


Figure 4.1: Schematic of the MPD process with mud pulsed telemetry.

temperature is assumed constant throughout the well. Table 4.1 gives a summary of the simulated well conditions.

Measurement noise was simulated by adding random noise, outliers and drift to the bit pressure, choke valve pressure, choke valve flow rate, and mud pump flow rate output signals before they entered each controller. The noise added to the original signal was approximately ± 1 bar. An example of the corrupted measurement data signals is shown in Figure 4.2.

Table 4.1: Summary of the simulated well parameters used in this work.

	SI Units	English Units
Casing diameter	0.273 m	$10\frac{3}{4}$ in
Drill pipe	0.127 m	5 in
True Vertical Depth (TVD)	2150 m	7054 ft
Measured Depth (MD)	4300 m	14108 ft
Water depth	20 m	65.6 ft
Rate of Penetration (ROP)	2 m/hr	6.56 ft/hr
Temperature	35° C	95° F

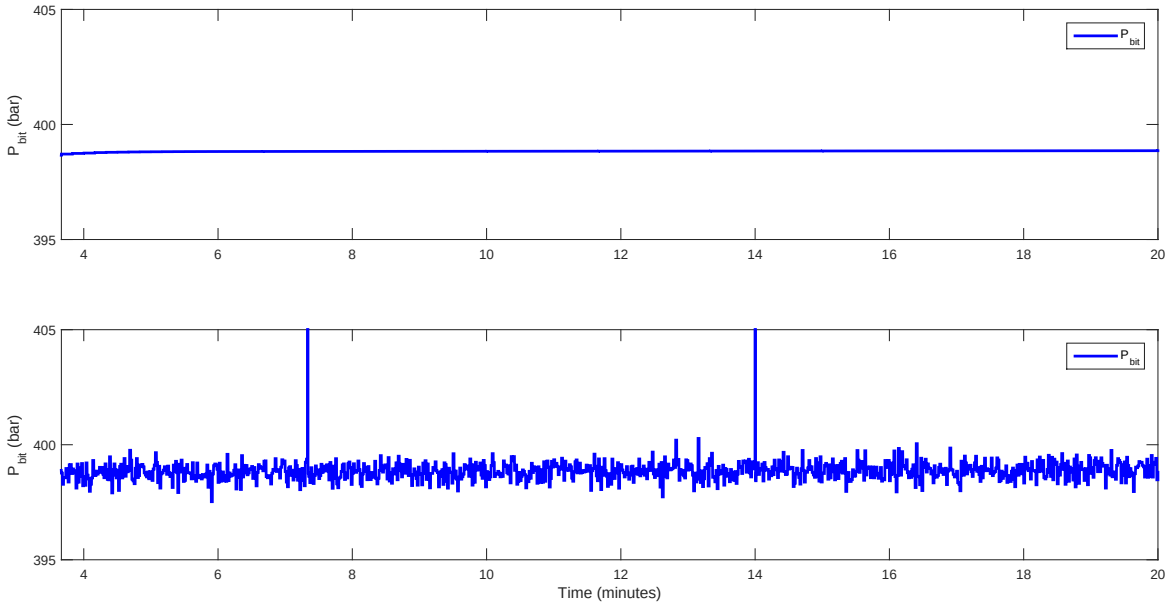


Figure 4.2: Original bit pressure signal (top) and signal corrupted with noise and outliers (bottom). The corrupted signal is sent to the controllers in order to simulate real-world measurement data.

4.2.1 Ensemble Control Structure

The control scheme includes three separate MPC controllers and a logic switch that selects the appropriate controller output to actuate the choke valve position and mud pump flow rate at the well. The three MPC controllers each use different control models, a high fidelity model, low-order model [76], and an empirical model. Each of these models receives measurements from the

well that are corrupted with noise and outliers to simulate real measurement conditions. A simple moving-average was used to filter some of the noise. Figure 4.3 shows a diagram of the ensemble control system and its relationship to the drilling simulation. While ensemble controllers have been researched in many applications, the novel contribution of this work is the implementation of an ensemble controller with bumpless transitions in multivariate MPD control.

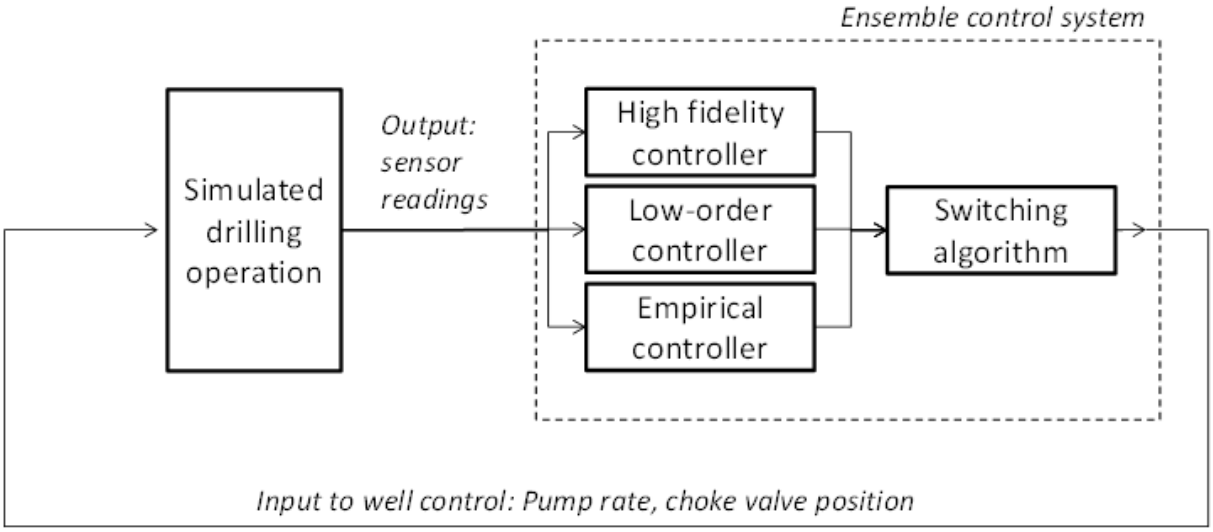


Figure 4.3: A simplified diagram of the simulated well and ensemble controller.

4.2.2 High Fidelity Controller

The high fidelity MPC controller uses the SINTEF flow model and a sequential solution method to find an optimal combination of choke valve position and mud pump flow rate to maintain the target bit pressure. For a detailed discussion of the equations and assumptions of this model see reference [137]. The model is used in an active set solver optimization routine with a 60 second time horizon. The optimization objective function uses the ℓ_1-norm and appropriate variable tuning costs. The ℓ_1-norm calculates the absolute value of the model deviation minus the desired set point. Tuning parameters for this high fidelity controller involve the length of the time horizon, time-based weighting of the bit pressure error, the costs for changing the manipulated variables and increasing the pump rate and the tolerance of the solver. The parameters are adjusted to obtain an optimal solution in a realistic time frame with acceptable use of the choke valve and mud pump. A

simple bias feedback was also built into the controller to ensure control despite some inaccuracies in the well model used by the controller. The bias enables the model to correct for variations as can be seen in Figure 4.4. This figure demonstrates the effect of the bias in diminishing pressure offset.

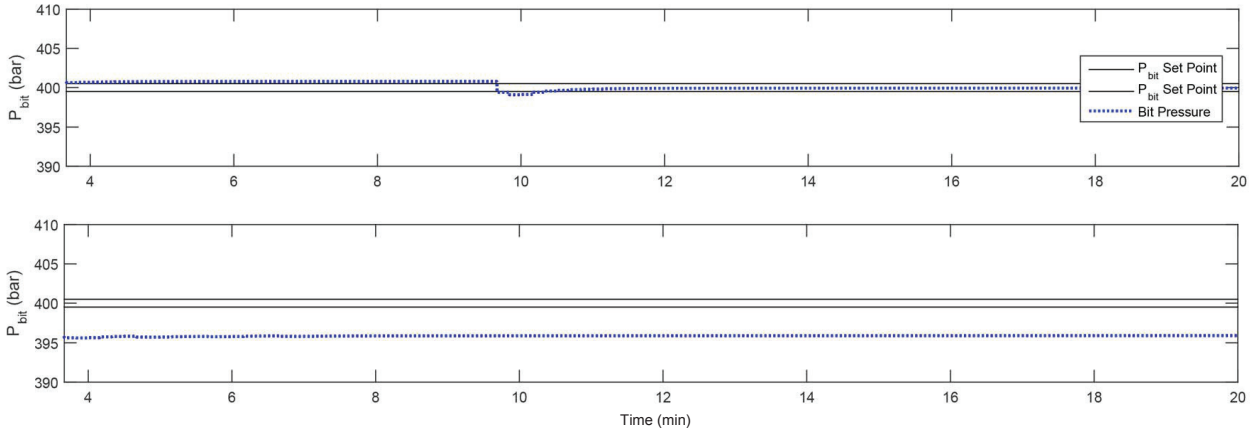


Figure 4.4: High fidelity controller operating with (top) and without (bottom) a bias feedback. The bit pressure set point is a dead-band region (rather than a single value) that is formulated using the ℓ_1-norm in the MPC controller objective function.

Because of the long solution time required by the controller, its control suggestions are only available to the ensemble switch every other control instance.

4.2.3 Low-order Controller

The low-order controller utilizes a reduced order observer model developed by Stamnes et al. [128], adapted for control by Asgharzadeh Shishavan et al. [76], and further modified for mud pulse telemetry in this work as shown in Equations 4.1-4.6. The controller uses this model in a single control algorithm to control the well at each instance. The model has three state variables that are shown in Table 4.2.

$$P_{bit} = P_c + \rho_a f_a h_{bit} (q_{bit})^2 + \rho_a g_c h_{bit} \tag{4.1}$$

$$q_{choke} = K_1 z_c [\rho_a (P_c - P_o)]^{0.5} \tag{4.2}$$

Table 4.2: Description of variables used in the low-order drilling model with initial values.

Variable	Definition	Initial Value
P_p	main pump pressure (State variable -SV)	86 $\text{bar}_{\text{gauge}}$
β_d	bulk modulus of the drillstring	14000 $\text{bar}_{\text{gauge}}$
V_d	volume of the drillstring	19.909 m^3
q_{pump}	flow rate of the main pump	1.8 m^3/min
q_{bit}	flow rate of the fluid through the drill bit (SV)	1.8 m^3/min
P_c	choke valve pressure (SV)	52 $\text{bar}_{\text{gauge}}$
β_a	bulk modulus of the annulus	14000 $\text{bar}_{\text{gauge}}$
V_a	volume of the annulus	13.3515 m^3
q_{back}	back pressure pump flow rate	0 m^3/min
q_{choke}	choke valve flow rate	1.8 m^3/min
q_{res}	reservoir gas influx flow rate	0 m^3/min
M	effective density per unit length	3500 $\text{kg}\,\text{m}^{-4}$
M_a	effective density per unit length of annulus	800 $\text{kg}\,\text{m}^{-4}$
M_d	effective density per unit length of drillstring	2700 $\text{kg}\,\text{m}^{-4}$
f_d	friction coefficient of drillstring	1 $\text{bar}\,\text{s}^2\,\text{m}^{-6}$
f_a	friction coefficient of the annulus	623.87 m^{-5}
ρ_d	actual density in the drillstring	1490 $\text{kg}\,\text{m}^{-3}$
ρ_a	actual density in the annulus	1372.1 $\text{kg}\,\text{m}^{-3}$
g_c	gravitational constant	9.81 $\text{m}\,\text{s}^{-2}$
P_{bit}	pressure at the bit	440 $\text{bar}_{\text{gauge}}$
h_{bit}	well depth	2150 m
K_1	Valve coefficient	0.145
z_c	choke valve position	60% (open)
P_0	pressure at the surface	1 $\text{bar}_{\text{absolute}}$

$$\frac{dP_c}{dt} = \frac{\beta_a}{V_a}(q_{bit} + q_{back} - q_{choke} + q_{res}) \tag{4.3}$$

$$\frac{dq_{bit}}{dt} = \frac{1}{M}(P_p - f_d q_{bit}^2 + \rho_d g_c h_{bit} - P_{bit}) \tag{4.4}$$

$$\frac{dP_p}{dt} = \frac{\beta_d}{V_d}(q_{pump} - q_{bit}) \tag{4.5}$$

$$M = M_a + M_d \tag{4.6}$$

The low-order MPC controller uses a ℓ_1-norm objective function that allows the formulation of a dead-band region for the set point, rather than one specific value. Within this region there is no penalty on the objective function. This dead-band helps reject noise and mitigate unnecessary control moves, helping extend the life of equipment and avoiding actions that vigilant operators would find unnecessary. The absolute value of the error was also implemented in a unique way to avoid the discontinuous first derivative of the objective function (which is inherent to an absolute value function), increasing the probability of solver convergence. Details on the formulation and implementation of this ℓ_1-norm dead-band objective function can be found in [6]. This controller was tuned by adding a cost for changing the valve position and pump flow rate, and limiting the amount the controller could move these variables at each control instance. The density of the mud in the annulus and the annulus friction factor are required inputs to this model and need to be estimated because they cannot be measured. Therefore, the controller is combined with online MHE, which uses a similar ℓ_1-norm objective function as the controller, to estimate the annulus friction factor and density in the annulus at each time step. The estimator uses the same model described above.

4.2.4 Empirical Controller

The empirical controller model is based on measurement data from the simulated well. The data is used to directly quantify the relationship of pump flow rate (q_{pump}) to bit pressure (P_{bit}) and of choke valve position (z_c) to bit pressure. Dynamic data is gathered by stepping each of the MVs up and down across their operating ranges while holding all other inputs constant. The MVs for this process are the choke valve position and the mud pump flow rate. While the MVs are stepped, the dynamic bit pressure response to these steps is recorded. Once the bit pressure response to the step in MVs is recorded, a FOPDT model is used to fit the data by adjusting the gain (K_p) and time constant (τ_p), according to Equation 4.7. Dead time (θ) is assumed negligible for this process. This type of model is used as a good approximation of nonlinear systems and is the most common

Table 4.3: Values of the gains and time constants for the FOPDT model.

	P_{bit}/q_{pump}	P_{bit}/z_c
K_p	8247.4 (bar/m^3s^{-1})	-46.749 (bar/%open)
τ_p	9.0812 (s)	0.96126 (s)

model to generically relate inputs to outputs. In Equation 4.7, x represents p_{bit}, and u represents the inputs (q_{pump}, z_c) into the model.

$$\tau_p \frac{dx}{dt} = -x + K_p u(t-\theta) \tag{4.7}$$

Table 4.3 shows the values of the gain and time constant for the P_{bit}/q_{pump} relationship and the P_{bit}/z_c relationship that were determined from the tests.

In an industrial application the K_p and τ_p values could be fit dynamically from real-time data to achieve a similar result. Once the individual input/output relationships were established, they were concatenated into a single matrix. This matrix was used to generate a state space model for Multiple Input Single Output (MISO) MPD process control. The empirical MPC controller also uses the ℓ_1-norm objective function and set point dead-band used in the low-order controller. Similar to the low-order controller, the empirical controller was tuned by adjusting the allowable rate of change for each MV and giving a penalty in the objective function for deviation from current positions.

4.2.5 Empirical Switch

The ensemble switch receives the suggested pump flow rate and valve opening control moves from each of the individual controllers and outputs a final control move for implementation in the drilling process. The decision of which controller suggestion to implement is based on a hierarchical procedure. The high fidelity controller is used at all times when it is available. However, the long time required for optimal solution convergence often disqualifies these control suggestions. Also, the added complexity of the model can occasionally result in the model predictive controllers not converging to a solution altogether, further disqualifying the recommended actions from the high fidelity controller. When the high fidelity controller is unavailable, the low-order

controller is used. The low-order controller requires the optimal solution convergence of both an MHE estimator and an MPC controller. If this controller fails to converge within the allotted time step, its suggestions are not implemented by the ensemble switch. When this situation is combined with an unavailable high fidelity controller, the empirical controller is used to maintain the pressure at the bit. If all three controllers fail, then the ensemble controller will fail also. This marks one of the shortcomings of this method. However, the probability of all three controllers failing in the same time step is less than the probability of a single controller failing. The control model redundancy increases the chances of stable process control.

One of the challenges associated with ensemble controller switching encountered in this and other works [93] is the tendency of the MVs to jerk when switched between controllers. This occurs because multiple optimal combinations of pump flow rate and valve position are possible due to the slight colinearity of these variables. The values of these optimal combinations for a given time instance are shown in a contour plot that shows all possible combinations of valve position and pump flow rate and the effect on the controller objective function in Figure 4.5.

It should be noted that the valley of optimal solutions is curved. This prevents averaging the individual controller recommendations because an averaged set of moves would fall in a sub-optimal region. Different combinations of these MVs are found by each of the individual controllers. While all may be optimal, the combinations are distinct and can cause an unacceptable jerk in the bit pressure during controller switching, as shown in Figure 4.6. The jerky transition is exemplified by the poor transition from control by the low-order controller to the empirical controller at 20 min. The transition from the high fidelity controller to the low-order controller also causes a ± 5 bar swing in bit pressure before stabilizing.

To overcome this challenge, the current process pump flow rate and valve position are feed into each of the controllers. The individual controllers that are not presently implemented use the current process values as an initial starting point for the optimization routine. They separately converge to the same or similar values for flow rate and valve position. This strategy keeps jerking at a minimum when switching between controllers. When this strategy is applied there is a seamless transition between controller suggestions, as demonstrated in the results section.

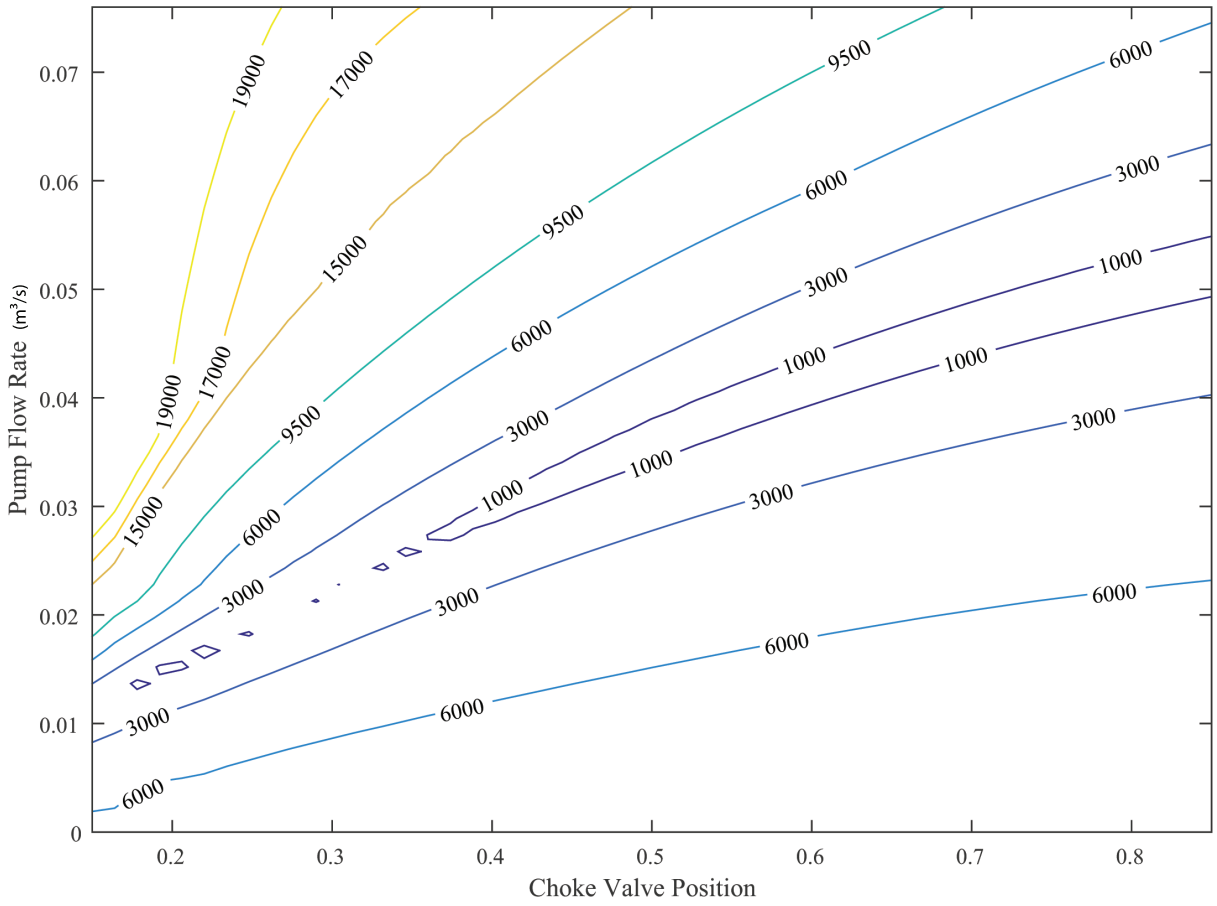


Figure 4.5: Plot of all possible MV control moves and their effect on the high fidelity controller objective function. The contours represent the values of the controller objective function which minimizes the error between the well measurements and the model predictions. The area of minimal error in pump flow rate and valve position combinations denotes the optimal operating region.

4.3 Results and Discussion

4.3.1 Normal Drilling Conditions

Normal drilling conditions are simulated with a bit pressure set point of 400 bar. In this simulation, each controller is demonstrated by simulating a failure in the high fidelity controller at 10 minutes and low-order controller failure at 20 minutes. The results are shown in Figure 4.7.

This figure shows the choke valve position and pump flow rates recommended by each controller, the selected controller moves and the resulting bit pressure, with its allowable band of

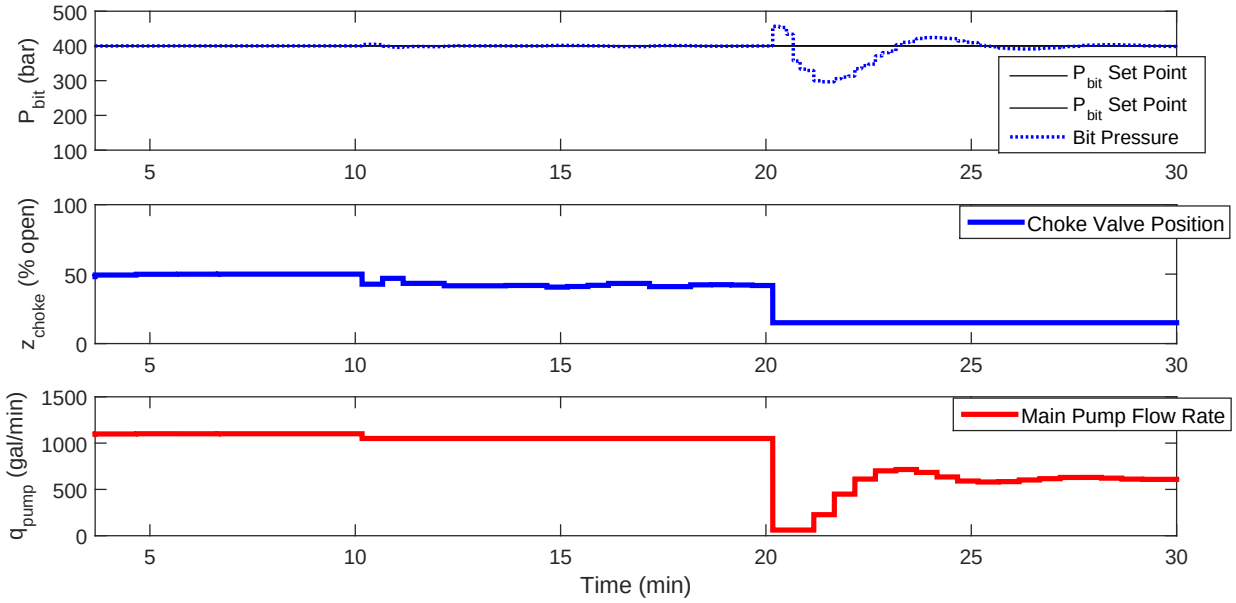


Figure 4.6: Demonstration of poor switching behavior when bit pressure (top), valve position (center), and pump flow rate (bottom) switch between controllers during normal drilling. The high fidelity controller is used until 10 minutes when control is switched to the low-order controller. At 20 minutes the empirical controller is used to control the well. This figure shows the unacceptable jumps in valve position and pump flow rate when switching among controllers, and how it affects bit pressure control.

± 1 bar from the 400 bar set point. The smooth transition between controllers is demonstrated. The low-order controller demonstrates the least stable control in this test, but it is still able to maintain the bit pressure within 1 bar of the desired set point. While the empirical controller performance is much more acceptable than the low-order controller in this instance, the low-order controller is preferred because the tuning is accurate over a wider range of operating conditions. The empirical controller has been tuned to process data at the current well conditions, but will quickly lose tuning as drilling continues and conditions change. Using online data to tune the empirical model when it is not in use is addressed in Chapter 5.

4.3.2 Pipe Connection

To simulate a pipe connection procedure, the main mud pump flow is scaled down to zero as the backpressure pump ramps up to maintain a steady flow of 600 L/min through the choke valve. The bit pressure set point is changed to 340 bar, ± 5 bar, and the ROP and RPM are set to 0

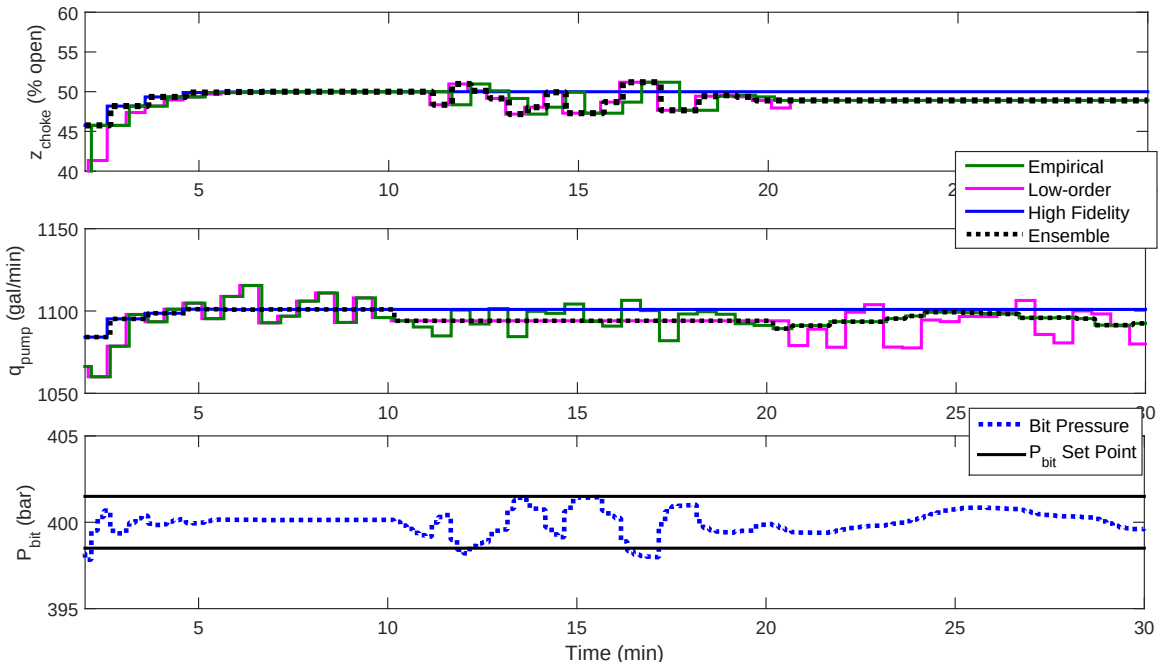


Figure 4.7: Bit pressure (top), valve position (center), and pump flow rate (bottom) when the controller switches between controllers during normal drilling. The high fidelity controller is used until 10 minutes when control is switched to the low-order controller. At 20 minutes the empirical controller is used to control the well.

during the 1 minute connection procedure. Since there is no mud flowing through the drill string, mud pulse telemetry is not available so all measurement feedback systems are turned off. The empirical controller does not account for this procedure in its model so the ensemble switch will not select any recommendations made by the empirical controller. Figure 4.8 shows the results of the pipe connection procedure with moves recommended by the high fidelity controller. The controller demonstrates impressive performance by maintaining the bottom hole pressure within 5 bar of the set point.

The lack of feedback measurements in this procedure forces the controllers to rely solely on the quality of their predictive models. The high fidelity model is the most accurate, and its predictive power is demonstrated here. However, the low-order model is not as detailed, and does not account for many of the more complex dynamics associated with a pipe connection procedure. The same procedure is simulated with control moves solely from the low-order model. The results are shown in Figure 4.9.

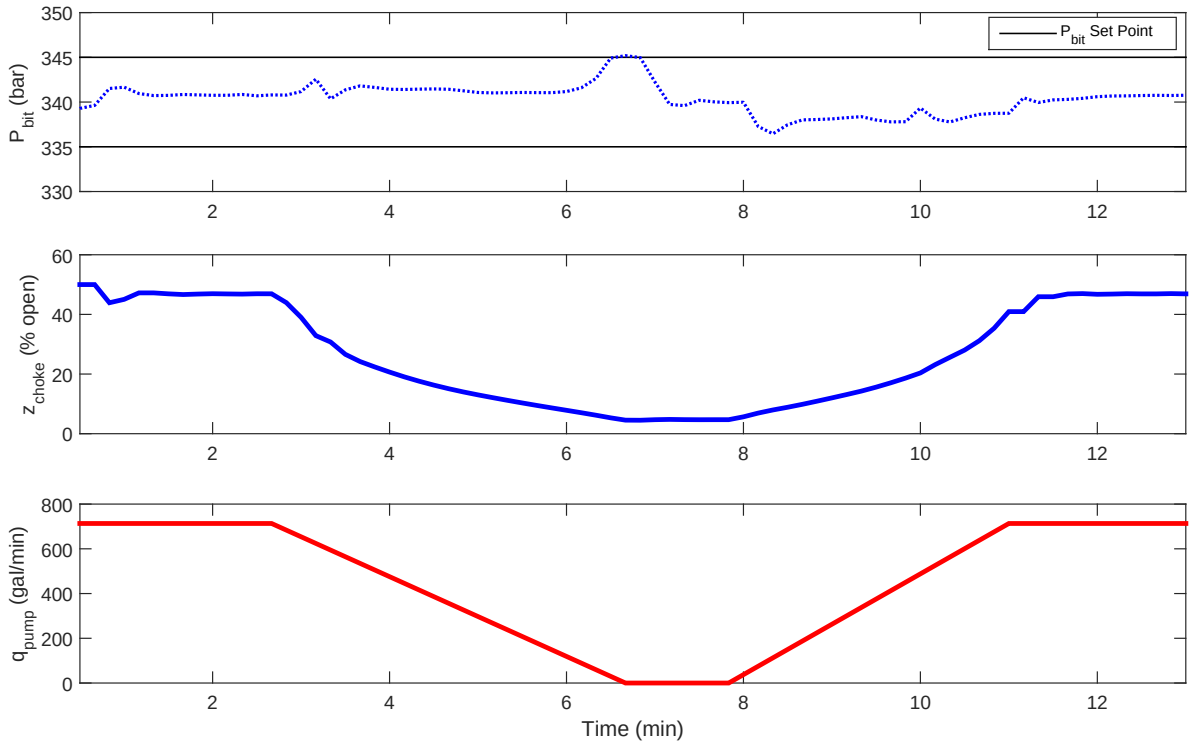


Figure 4.8: Ensemble controller performance during a simulated pipe connection procedure. The ensemble controller switch uses the high fidelity controller during this procedure.

The low-order controller maintains an acceptable bit pressure for almost the first half of the procedure; the pressure stays within the 5 bar dead-band most of the time. However, lack of bit pressure measurements do not allow the model to be updated. Instead, the complex fluid dynamics of stopping and starting the pump and drill string rotation eventually causes the well conditions to stray unacceptably away, and the bit pressure spikes accordingly. However, once sufficient mud is flowing and mud pulse telemetry measurements are available (around 10.5 min), the model very quickly adapts itself and brings the bit pressure to an acceptable level. Clearly the high fidelity controller performance is superior; however, it may not always be available. Figure 4.10 shows the results of simulated high fidelity controller availability during a pipe connection procedure. When the high fidelity controller suggestions are inaccessible, the ensemble switches to the low-order controller to maintain the bit pressure within ± 5 bar of the 340 bar set point. In Figure 4.10, the sharp changes in bit pressure correspond to the loss of high fidelity control moves, this happens at 4, 6.5, and 9 minutes. While the low-order controller is able to regain control over the

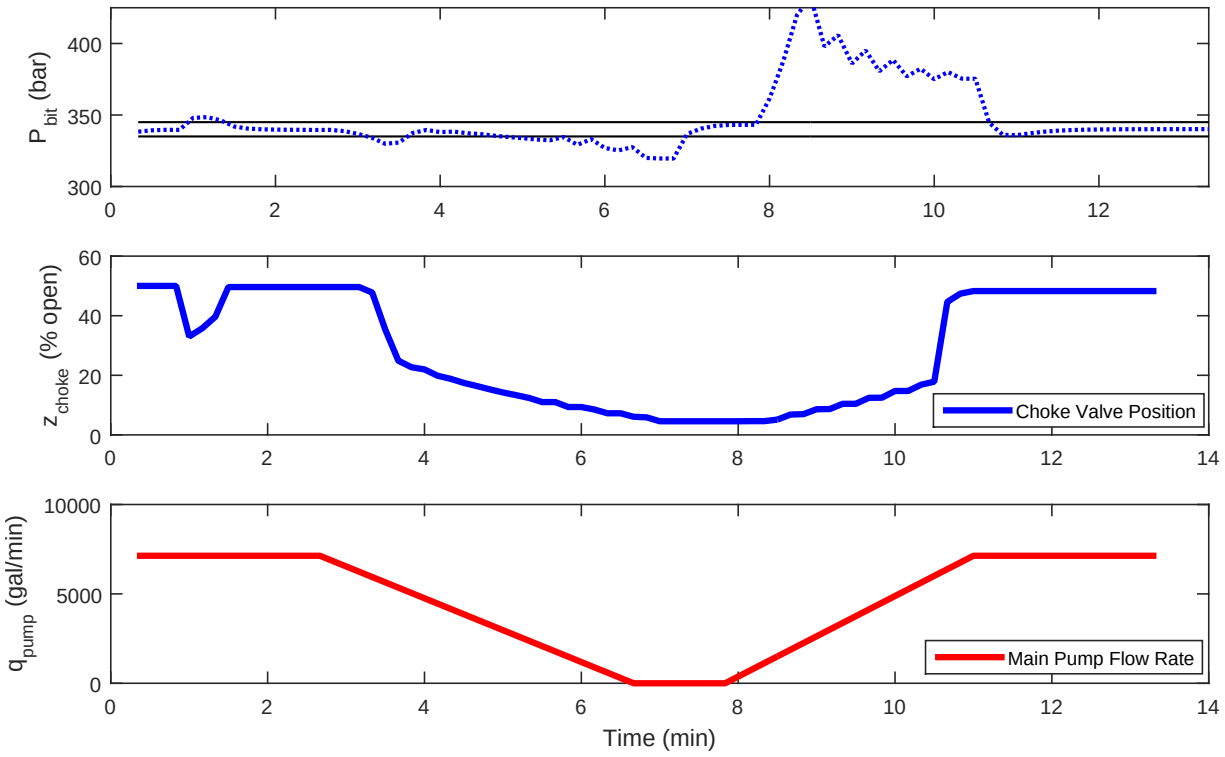


Figure 4.9: Poor ensemble controller performance during a simulated pipe connection procedure. The ensemble controller switch uses the low-order controller during this procedure and the predictive accuracy of the model is not sufficient to maintain desired pressure control.

bit pressure, the overshoot is unacceptable. This behavior is likely due to the lower accuracy and lower predictive capability of the low-order model. However, with further tuning, the controller could control the process in an acceptable manner. This simulation demonstrates that the model hierarchy becomes increasingly important in situations like pipe connection when measurement feedback is lost and the controllers rely solely on their model predictive accuracy.

The power of this work is the functionality and versatility of the ensemble switch. The main purpose of the ensemble switch is to seamlessly implement the most accurate controller available to maintain acceptable bit pressure. Due to the slight co-linearity of the pump flow rate and valve position on bit pressure (see Figure 4.5), each controller is able to find unique, yet optimal solution. This leads to drastic fluctuations in MVs when changing between controllers and the associated unwanted fluctuations in bit pressure. To achieve a seamless transition, the unused controllers are fed the current outputs of the ensemble switch as initial values for their

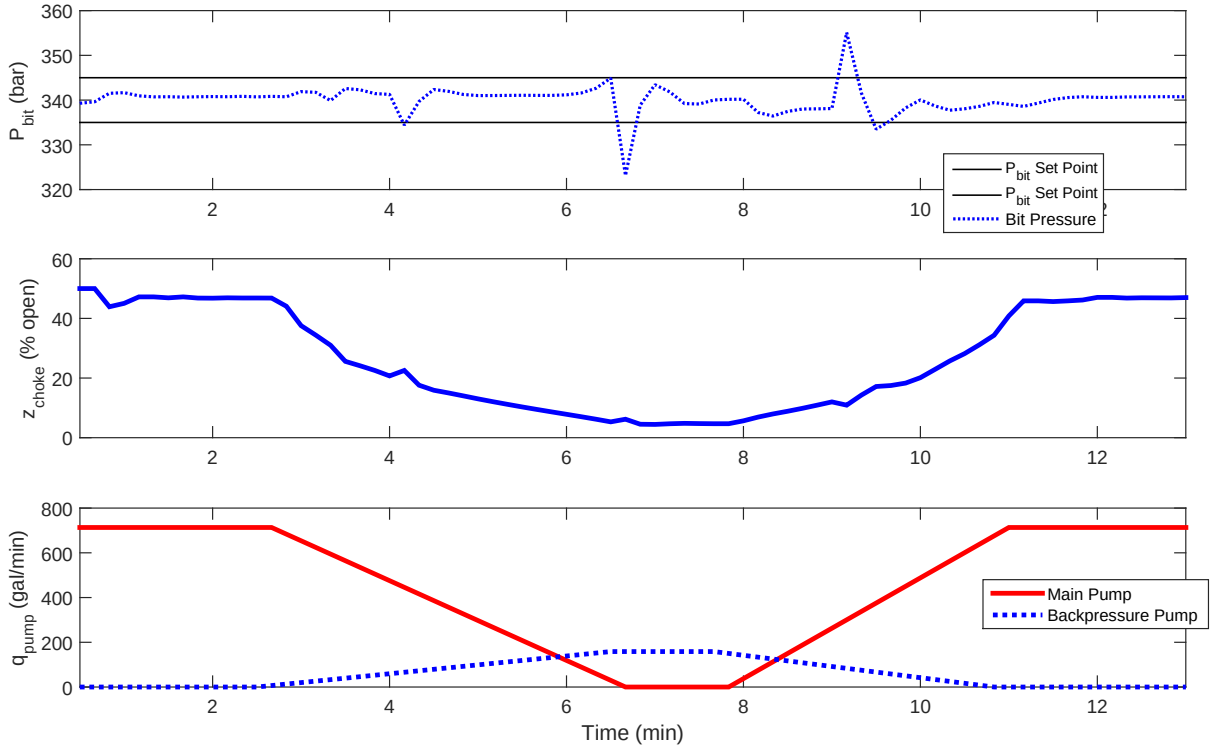


Figure 4.10: Ensemble controller performance during a simulated pipe connection procedure. Loss of high fidelity controller recommendations occurs at 4, 6.5, and 9 minutes. At these times, the low-order controller is used to control the process until the high fidelity controller becomes available. As bit pressure measurements are unavailable, the controllers rely solely on the accuracy of their model predictions to maintain the bit pressure. This figure demonstrates the need for accurate MPC models in MPD.

optimization routine. This strategy allows the optimizers to converge to MV suggestions that are near the current conditions, and facilitates smooth transitions between controllers. Additionally, the multiple control models in the ensemble controller allow for individual controllers to be tuned without interrupting drilling operations. This allows for regular maintenance of all controllers without impacting drilling productivity.

4.4 Conclusion

An ensemble controller consisting of a switch and three multi-fidelity MPC controllers is implemented in a MPD simulation that uses a high fidelity pressure dynamics well model. The switch implements each controller based on the model accuracy and availability. A high fidelity

controller is always used when available, a low-order controller is used when the high fidelity is unavailable, and an empirical controller is used when the other controllers are unavailable due to feedback noise or failed solver convergence. The most significant challenge associated with switching between controllers is a sudden jump in the bit pressure during the switch. This is due to the slight colinearity of the MVs and the unique solutions of each controller. This is overcome by feeding the output of the current choke opening and mud flow rate to the controllers not in use and adding a penalty for deviating from those values. This allows for smooth and successful switching between the various controllers during normal drilling. Additionally, during a simulated pipe connection procedure, when no mud flow causes the loss of bit pressure measurements, the controllers must rely solely on the predictive capabilities of their respective models. In this case, the high fidelity controller is able to maintain acceptable bit pressure, while the empirical and low-order models fail to do so. Once the controller performance is improved, future research will include exploring the ensemble controller performance in other common drilling scenarios such as unwanted reservoir gas influx, stick/slip, and surge and swab effects.

CHAPTER 5. ADVANCED SWITCHED CONTROL OF MANAGED PRESSURE DRILLING

This chapter is published as: Eaton, A.N., Beal, L.D.R., Thorpe, S.D., Janis, E.H., Hubbell, C., Hedengren, J.D., Nybø, R., Aghito, *Real time model identification using multi-fidelity models in managed pressure drilling*, Computers & Chemical Engineering, Volume 97, 2017, Pages 76-84, DOI: 10.1016/j.compchemeng.2016.11.008

5.1 Introduction

High fidelity simulators are first principles based models that closely approximate reality, and are characterized by dynamic nonlinear equations. The most rigorous process models can contain more than 10^6 equations and variables [138]. Such models have been used in Computational Fluid Dynamics (CFD), operator training simulators, and in process systems engineering for over 50 years [11]. Another common use for rigorous first principles models is to derive less rigorous, yet more computationally manageable, control models [139], [140]. The value of first principles models in real time feedback control is most apparent in MPC, which is the most widely used advanced control method in refining, chemical, and petrochemical processes [9].

Ideally, a control model would exactly describe plant dynamics in every operating condition. While this is not possible due to computational and complexity limitations, high fidelity simulators are rigorous models that accurately describe real processes over a wide range of operating conditions, thus requiring significantly less tuning from operational data than empirical models. Empirical model identification can be disruptive to operations and very costly, although this has greatly improved with closed loop identification techniques [141]. Also, closed loop control systems can become unstable even with highly accurate models, and several robust control strategies have been developed to guarantee stability for linear and nonlinear MPC applications [28]–[30], [142]. A control model must have a certain level of accuracy before any guarantees of controller stability and performance can be made. Accurate model predictions can lead to fewer iterations be-

cause the optimizer is able to minimize the error between the process and the model more quickly than inaccurate models. If the error is inherently small, it can quickly be brought within the specified tolerance with linear or quadratic convergence rates that are typical of Sequential Quadratic Programming (SQP) solvers near the solution. Accordingly, it has been demonstrated that improved models provide better control than less rigorous models in MPC [32]. Additionally, many processes, such as polymer grade transitions and oil well drilling, are extremely nonlinear to the extent that linear model approximations are insufficient to control the process.

In addition to performance improvement, highly accurate model predictions allow a controller to maintain control over a process, over the prediction horizon, even when there is no process feedback due to sensor failure, etc. While sensor failure is common in oil and gas well drilling, loss of measurement feedback is more common during normal operations due to pipe connection procedures. Maintaining control during the temporary and intermittent loss of feedback is the major motivation for using high fidelity simulators in real time control.

While high fidelity simulators have many benefits, rigorous models are difficult to implement in NMPC feedback control due to the short (typical range is 1 second to 10 minute) cycle time in which a large NLP problem must be solved. If the optimization is not completed within the cycle time, controller instability will occur [143], [144]. Because of this limitation, rigorous models have not been widely implemented in real time control. However, recent improvements in algorithm design and hardware capabilities have opened up the possibility of using the accuracy of high fidelity simulators in real time feedback loops. For example, multiple control algorithms have been developed that compute the full NLP problem offline and perform only a sensitivity calculation online [145]–[147]. These sensitivity based methods have been demonstrated in controlling reactors [145], distillation columns [147], [148], power plants [148], and adsorption beds [149]. Others have used large scale models in Dynamic Real Time Optimization (DRTO) [32], [150], [151]. However, this work introduces a method for using high fidelity simulators in NMPC by implementing a switched control scheme. The method uses dynamic online model assessment and control model identification from high fidelity simulated data.

5.2 High Fidelity Switched Control

Switched and hybrid systems have been extensively studied [152]–[154], and MPC variants have also been developed [155], [156]. The formulation of stability and performance guarantees have been developed in switched linear and nonlinear MPC by several researchers [142]. However, none of these previous studies have attempted to incorporate high fidelity simulators into the control law to improve model predictions, and subsequently controller performance and stability. The objective of this work is to couple high fidelity model predictions with the speed and stability of linear MPC by dynamically identifying the empirical control model in a switched control scheme.

The switched controller presented in this work uses a high fidelity nonlinear, first principles based model running in parallel with the physical process. Also, a linear, empirical model predictive controller and a nonlinear model predictive controller with a simplified low-order model run parallel with each other. The output suggestions of these controllers are fed into a switching algorithm that implements the linear controller suggestions into the nonlinear process. When the linear model prediction error exceeds a specified tolerance, the algorithm samples the high fidelity model with step inputs to produce a simulated data set of the current operating conditions. Then it uses regression techniques to fit the new gain and time constant in the linear, empirical model to the current simulated data set. While this model identification occurs, the algorithm implements the low-order NMPC controller suggestions to maintain control over the process. Figure 5.1 shows a diagram of the proposed controller scheme.

This strategy allows the slow, yet very accurate predictive capabilities of the high fidelity model to be implemented into a fast, yet locally accurate linear model, all without interrupting the process. It allows control model identification without disrupting operations. As shown in Figure 5.1, this control scheme takes advantage of the accurate predictions of the high fidelity simulator while minimizing online computational costs. Online computation time can be further reduced by computing in parallel on separate resources.

In addition to incorporating high fidelity predictions, this switched control scheme differs from traditional gain scheduling and other switch schemes, such as Multiple Models Predictive Control (MMPC)[155], in that there is no predetermined switching time or scheduling variable. Also, the control models do not need to be identified before implementing the controller. This is

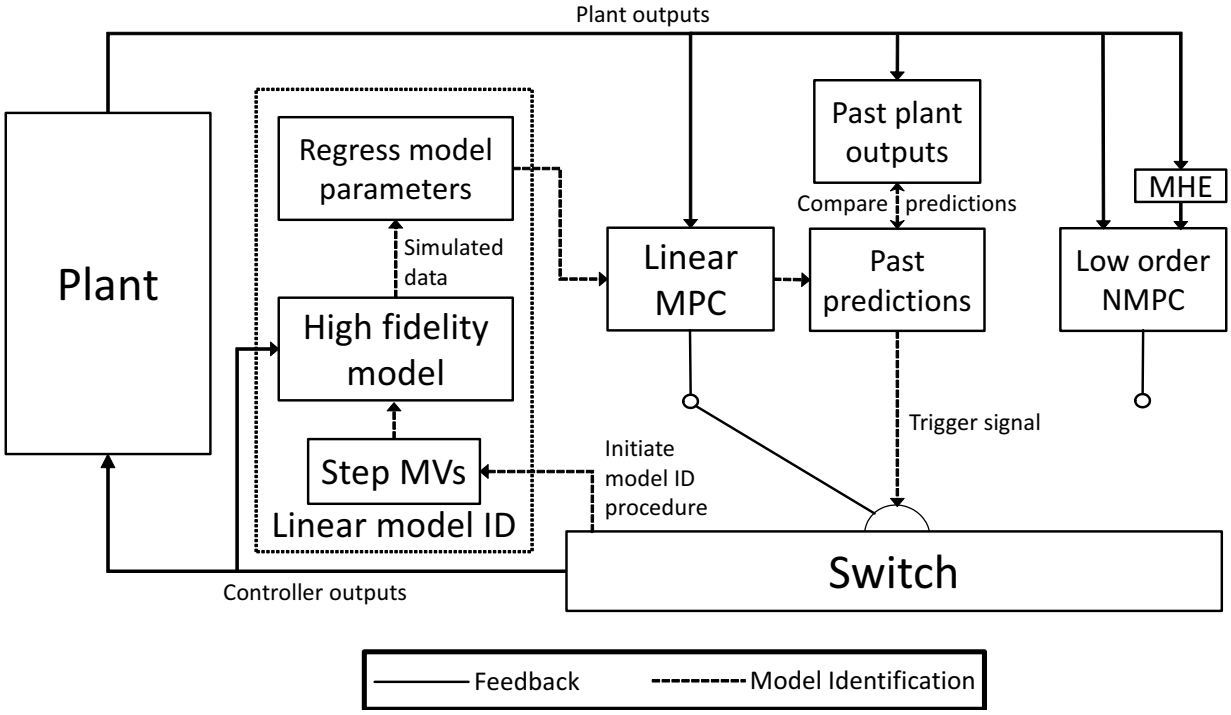


Figure 5.1: Diagram of a switched control structure.

helpful in systems where model identification is disruptive to operations, or when identifying the model before operations begin is not possible such as with automated oil well drilling.

5.3 Controller Stability

Even when each of the individual controllers in a switched system are asymptotically stable, destabilization due to switching among the controllers must be addressed before stability can be guaranteed [157], [158]. Therefore, it is necessary that the switch itself and the individual controllers are stable. For implementation of the present switched control algorithm, the individual MPC controllers can be formulated to take advantage of the recent advances in NMPC stability theory found in [159] and the included citations. In essence, stability can be achieved if a feasible solution is found and the finite optimization horizon is sufficiently long. Methods exist to efficiently calculate the horizon length required for stability to be ensured at each time step [160]. Other methods require the addition of terminal cost constraints or a continuous Lyapunov function. Yet, these stability methods are insufficient for hybrid systems because of the discontinuous

derivatives associated with switching [142]. The stability of this switching algorithm is addressed in a way that is unique to this control scheme.

Stability of switched and hybrid MPC systems has been centered around piecewise affine systems [142], [161]. In other words, switched piecewise affine systems, such as MMPC or Switched MPC (SMPC), switch among controllers with locally affine regions. The switching is from a locally accurate controller to a neighboring locally accurate controller [159]. In contrast, this work presents a scheme that switches among NMPC controllers that are accurate in the same operating regions. This redundant control model structure has several benefits including online model maintenance and some desirable stability properties. For instance, one of the major issues with discontinuous switching of redundant continuous NMPC controllers is the jump in controller outputs that can occur upon switching [162]. This sudden change in process inputs can result in poor or lost control. This jump occurs because the individual NMPC optimizers can find local minima of a nonconvex problem resulting in differing MV suggestions to achieve the same CVs. Factors such as MV move suppression or the multivariate nature of the problem contribute to this discontinuity between MV suggestions to reach the same CV targets. When switching among controller suggestions, the undesirable jump is manifest. One of the contributions of this work is addressing switch stability by synchronizing the NMPC controllers. This is accomplished by initializing each of the NMPC optimization problems, at each time step, with the current process conditions. Because each controller starts with the same initial conditions, and use the same optimization routine, each converges to similar solutions and transitions between solutions are smooth. Also, the switching is induced when control model predictions no longer match process measurements within a specified accuracy. This means that switching between controllers will only occur if it stabilizes the system. After a switch is made, another switch cannot occur for a given ”dwell time”, this prevents instability associated with switch chatter [163]. With the assumption that the individual controllers are stable, and combined with the previously stated synchronization techniques, switch stability is implied.

5.4 Simulated Managed Pressure Drilling

The switched control strategy is demonstrated on a simulated oil well drilling process. An oil well is created by drilling into the earth for several hundred to several thousand feet, stopping

to insert and cement casing pipe to the well bore, then repeating the process until the target depth is reached. As the well deepens, more drill pipe is connected to the drillstring. At the bottom of the drillstring, a BHA, consisting of measurement and steering equipment, is attached to the drill bit. The drill bit is cooled by the drilling fluid, or mud, which also moves the rock cuttings to the surface and maintains pressure in the well annulus (see Figure 5.2). The well annulus fluid pressure consistently needs to be greater than the geologic reservoir fluid pressure to prevent hydrocarbons from entering the well during the drilling process. If the mud pressure in the well is too high it can damage the rock formation; if it is too low hydrocarbons from the subsurface reservoir can come to the surface in an uncontrolled and dangerous manner. When this catastrophe happens it is known as a blowout. The well bore pressure must be maintained within a small range of pressures that will balance the reservoir fluid pressure to prevent fractured formations and blowouts. To help achieve this pressure balance, a variation of traditional drilling, called MPD, was developed. A simplified schematic of MPD is shown in Figure 5.2. MPD uses pressure measurements from the BHA to inform the driller of the need to adjust the main mud pump flow rate and choke valve opening to reach the desired pressure target in the well. A back pressure pump is used to maintain well pressure during pipe connection procedures when the main mud pump is disconnected from the drillstring.

Automation of MPD is advancing in industrial practice with mass balance control of the drilling mud being the most common practice. Another automated MPD method takes advantage of a highly calibrated high fidelity simulator whose predictions are used as feedback for a controller which manipulates the choke valve to control the bit pressure in the real process [137]. It has also been demonstrated that an automated controller can maintain borehole pressure and reject disturbances faster and more accurately than manual control by using NMPC with high quality process data [36]. One of the major challenges with implementing NMPC is the quality of the data sent by the downhole instruments. The most common method of receiving continual pressure measurement data from the BHA is through mud pulsing [129]. In mud pulsing, a pressure transducer sends pressure waves through the annulus fluid to a receiver at the surface. The pulses are then decoded into pressure measurements. Currently, the maximum data transmission rate of mud pulsing is 80 bits per second [164]. One of the major limitations of mud pulsing technology is the need to have the mud flowing for it to work. This makes receiving downhole pressure measurements impossible

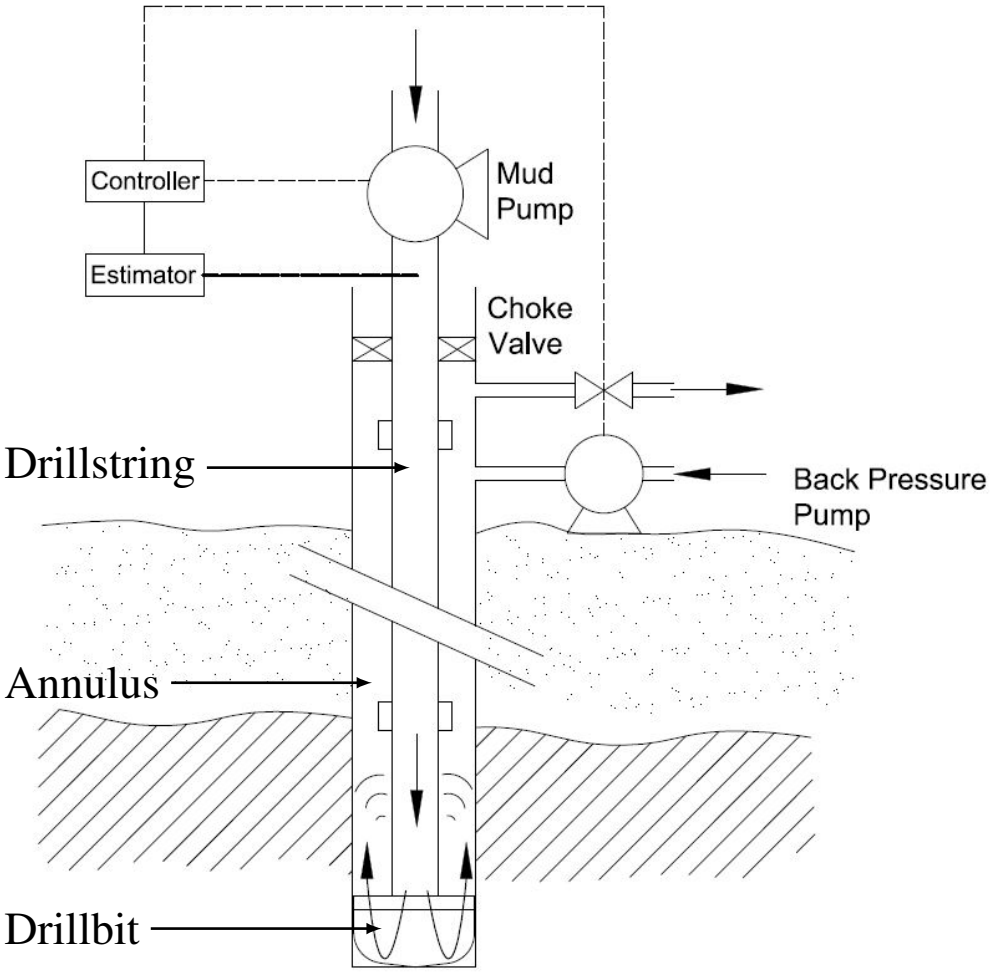


Figure 5.2: Simplified schematic of the automated MPD process.

when the mud pump stops for regular events such as pipe connection procedures. Additionally, the small bandwidth and long delay time of mud pulsing technology pose a significant challenge to automating MPD with direct downhole pressure measurements, and improving the technology is an active area of research [130]. Other researchers are moving away from mud pulsing in favor of other technologies such as WDP [129]. Regardless of the means of transmission, the downhole data quality is substantially low ($\pm$ 20 bar) for traditional sensors and better ($\pm$ 1 bar) for newer sensors [165] that have been developed to meet MPD control requirements. Even with the new pressure sensors, the inherently harsh borehole environment and the discontinuous nature of the drilling process make data collection and reliability a challenge.

Additionally, several abnormal events can occur during drilling operations. For example, drilling into an unexpected high pressure reservoir can offset the well/reservoir pressure balance and allow an unwanted influx of gas into the well bore. This situation is known as unwanted gas influx or kick, and is characterized by an increase in pressure in the annulus as the gas rises to the surface and increased flow rate in the annulus that acts as a disturbance to the process. In practice, the drilling process is stopped and shut in until the gas is circulated out and the well is controlled with a change in mud weight or choke valve position. In automated MPD, the set point for the choke pressure is increased to stop the influx of gas. Then, drilling operations are slowly brought online after the mud density is increased sufficiently to balance the new pressure at the bit. Controlling a kick is known as well control, and is an active area of research in industry and academia. Kicks, pipe connection procedures, delayed bit pressure measurements, and measurement noise are all factors included in the simulation to better simulate issues encountered in industrial practice.

In these simulations, the mud pump flow rate and choke pressure are the MVs and the CV is the pressure at the bottom of the well. As the drill bit is always at the bottom of the well, the bit pressure is used for the CV. An empirical model and a low-order first principles model are used in parallel MPC controllers as depicted in Figure 5.3. These controllers are described in Sections 4.1 and 4.2.

5.4.1 Empirical Controller

The empirical controller model uses a linear FOPDT model, as seen in Equation 5.1, with gain (K_p), time constant (τ_p), and dead time (θ) that are fit to simulated data from the SINTEF Flow Model high fidelity simulator [137] in the procedure described in Section 3. The simulated data are generated by artificially stepping each of the MVs up and down in slow succession across prescribed operating ranges.

$$\tau_p \frac{dx}{dt} = -x + K_p u(t-\theta) \tag{5.1}$$

In Equation 5.1, x represents the bit pressure (P_{bit}) and u is a vector representing the mud pump flow rate (q_{pump}) and the pressure upstream of the choke valve (P_c) which are model inputs.

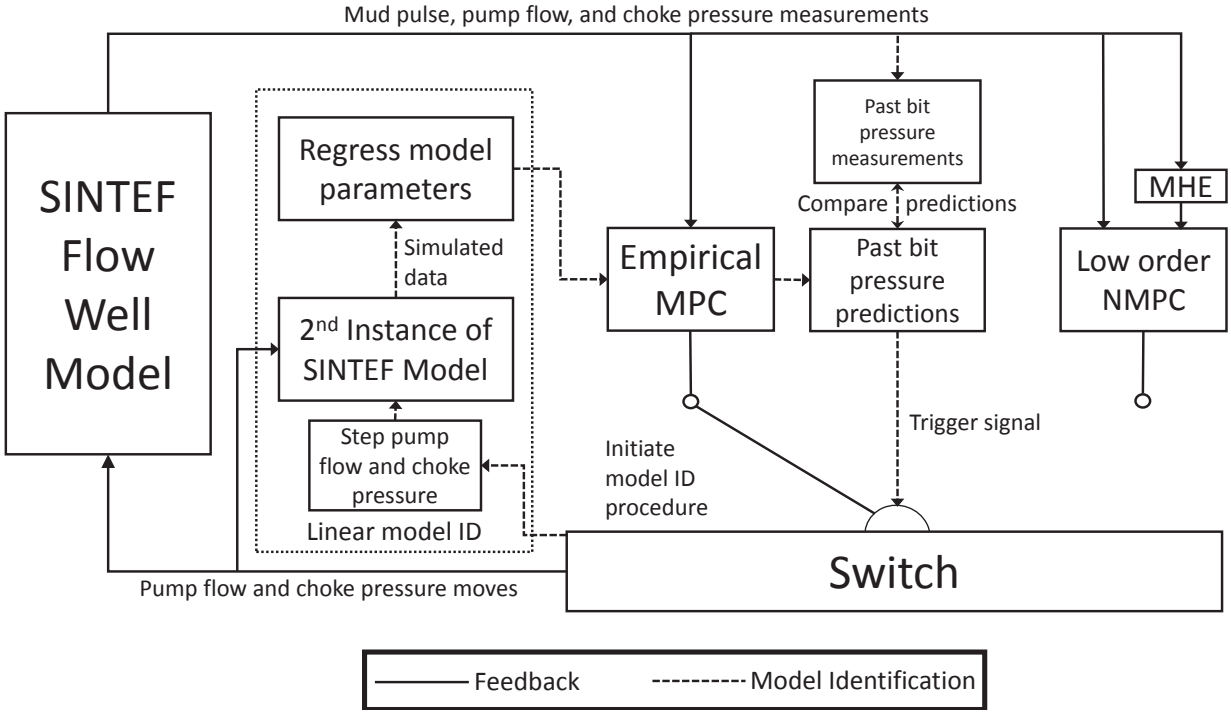


Figure 5.3: Diagram of the switched control structure used in this work.

Once the individual input/output relationships are established, they are concatenated into a single matrix. This matrix is used to generate a state space model for the MISO MPD process controller at each identification routine. The MPC controller uses an ℓ_1-norm objective function that allows the formulation of a dead-band region for the set point, rather than one specific target value. Equation 5.2 shows the generalized ℓ_1-norm control formulation.

$$\begin{aligned}
\min_{x,y,u} \Phi = {} & w_{hi}^T e_U + w_{lo}^T e_L + y^T c_y + u^T c_u + \Delta u^T c_{\Delta u} \\
\text{s.t.} \quad & 0 = f\left(\tfrac{dx}{dt}, x, y, p, u\right) \\
& 0 = h(x, y, p, u) \\
& 0 \leq g(x, y, p, u) \\
& \tau_c \tfrac{dy_{t,hi}}{dt} + y_{t,hi} = sp_{hi} \\
& \tau_c \tfrac{dy_{t,lo}}{dt} + y_{t,lo} = sp_{lo} \\
& e_U \geq y - y_{t,hi} \\
& e_L \geq y_{t,lo} - y
\end{aligned} \tag{5.2}$$

In this formulation, Φ is the objective function, x, y, and u are vectors of the process states, the model predictions, and the model inputs respectively. In the MPD case, u would be a vector of the mud flow rate and choke pressure, y would equal x and be the bit pressure. w_{hi} and w_{lo} are penalty matrices for solutions outside of the dead-band region, while e_U and e_L are slack variables for the dead-band high and low limits. c_y, c_u, and $c_{\Delta u}$ are cost vectors for the model predictions, inputs, and change of inputs respectively. f is a generalized function of the model equations as functions of x, y, u, p and $\frac{dx}{dt}$, where $\frac{dx}{dt}$ is the time derivative of x and p is a vector of the model parameters. Similarly, h is a generalized function of the system equality constraints, and g is a generalized function of the systems inequality constraints. τ_c is the desired CV time constant, and $y_{t,hi}$ and $y_{t,lo}$ are the upper and lower limits of the desired trajectory when changing set points. sp_{hi} and sp_{lo} define the set point dead-band region. The controller is tuned by adjusting the weighting vectors: c_y, c_u, $c_{\Delta u}$, w_{hi}, and w_{lo}, and the CV time constant τ_c. The tuning favors adjusting the choke pressure as much as possible before the pump is adjusted to reach the set point. This is because the pump needs a flow rate sufficient to remove rock cuttings up the annulus. A sudden drop in mud flow can cause the solids in the annulus to precipitate and lead to an expensive and disruptive stuck pipe situation. Unnecessary control moves are further mitigated by the dead-band region set point.

Within the dead-band region (between upper and lower limits) there is no penalty in the objective function. The use of a dead-band helps reject noise and mitigate unnecessary control moves [166], helping extend the life of equipment and avoiding actions that vigilant operators would find unnecessary. The absolute value of the error is also implemented in a unique way to avoid the discontinuous first derivative of the objective function (which is inherent to an absolute value function), improving the effectiveness of gradient based solution techniques such as SQP. Further details on the formulation and implementation of this ℓ_1-norm dead-band objective function are found in [6].

5.4.2 Low-order Controller

The low-order controller uses a reduced order observer model developed by Stamnes et al. [128], adapted for WDP control by Asgharzadeh Shishavan et al. [164], and further modified for mud pulse telemetry in this work as shown in Equations 5.3-5.7. Descriptions of the model

variables and parameters are shown in Table 5.1.

$$P_{bit} = P_c + \rho_a f_a h_{bit} (q_{bit})^2 + \rho_a g_c h_{bit} \tag{5.3}$$

$$\frac{dP_c}{dt} = \frac{\beta_a}{V_a}(q_{bit} + q_{back} - q_{choke} + q_{res}) \tag{5.4}$$

$$\frac{dq_{bit}}{dt} = \frac{1}{M}(P_p - f_d q_{bit}^2 + \rho_d g_c h_{bit} - P_{bit}) \tag{5.5}$$

$$\frac{dP_p}{dt} = \frac{\beta_d}{V_d}(q_{pump} - q_{bit}) \tag{5.6}$$

$$M = M_a + M_d \tag{5.7}$$

The low-order MPC controller also uses the ℓ_1-norm objective function formulation used by the empirical MPC controller. This controller was tuned by adding a cost for changing the valve position and pump flow rate, and limiting the amount the controller moves these variables at each time in the control horizon. Also, the density of the mud in the annulus and the annulus friction factor are changing as rock cuttings are carried away during drilling. These two parameters are required inputs to the model and need to be estimated because they cannot be measured. Therefore, the controller is combined with online MHE, which uses a similar ℓ_1-norm objective function, to estimate the annulus friction factor and density in the annulus at each time step. The estimator

Table 5.1: Description of variables and parameters used in the low-order drilling model with initial values.

Variable	Definition	Initial Value
P_p	main pump pressure (State variable -SV)	86 $\text{bar}_{\text{gauge}}$
β_d	bulk modulus of the drillstring	14000 $\text{bar}_{\text{gauge}}$
V_d	volume of the drillstring	19.909 m^3
q_{pump}	flow rate of the main pump	1.8 m^3/min
q_{bit}	flow rate of the fluid through the drill bit (SV)	1.8 m^3/min
P_c	choke valve pressure (SV)	52 $\text{bar}_{\text{gauge}}$
β_a	bulk modulus of the annulus	14000 $\text{bar}_{\text{gauge}}$
V_a	volume of the annulus	13.3515 m^3
q_{back}	back pressure pump flow rate	0 m^3/min
q_{choke}	choke valve flow rate	1.8 m^3/min
q_{res}	reservoir gas influx flow rate	0 m^3/min
M	effective density per unit length	3500 $\text{kg}\,\text{m}^{-4}$
M_a	effective density per unit length of annulus	800 $\text{kg}\,\text{m}^{-4}$
M_d	effective density per unit length of drillstring	2700 $\text{kg}\,\text{m}^{-4}$
f_d	friction coefficient of drillstring	1 $\text{bar}\,\text{s}^2\,\text{m}^{-6}$
f_a	friction coefficient of the annulus	623.87 m^{-5}
ρ_d	actual density in the drillstring	1490 $\text{kg}\,\text{m}^{-3}$
ρ_a	actual density in the annulus	1372.1 $\text{kg}\,\text{m}^{-3}$
g_c	gravitational constant	9.81 $\text{m}\,\text{s}^{-2}$
P_{bit}	pressure at the bit	440 $\text{bar}_{\text{gauge}}$
h_{bit}	well depth	2150 m
P_0	pressure at the surface	1 $\text{bar}_{\text{absolute}}$

uses the same low-order model described above. Equation 9 shows the objective function, slack variables, and error equations for the MHE used in this work.

$$\begin{aligned}
\min_{x,y,p} \Phi &= w_m^T (e_U + e_L) + w_p^T (c_U + c_L) + \Delta p^T c_{\Delta p} \\
\text{s.t.} \quad & 0 = f\left(\tfrac{dx}{dt}, x, y, p, u\right) \\
& 0 = h(x, y, p, u) \\
& 0 \leq g(x, y, p, u) \\
& e_U \geq y - y_x + \tfrac{db}{2} \\
& e_L \geq y_x - \tfrac{db}{2} - y \\
& c_U \geq y - \hat{y} \\
& c_L \geq \hat{y} - y \\
& e_U, e_L, c_U, c_L \geq 0
\end{aligned} \tag{5.8}$$

Here, p is a vector of the model parameters that are estimated, w_m is a cost given for measurement deviation, w_p is a cost given for deviation from the previous solution, and Δp is the change in model parameters. e_U and e_L are slack variables for the dead-band upper and lower limits, c_U and c_L are slack variables for the upper and lower limits of the prior model solution, and $c_{\Delta p}$ is a cost for changing the previous parameter values. Also, y_x is a vector of process measurements, db is the size of the dead-band, and $\hat{y}$ is the previous model values. A more complete presentation of the estimation form of the ℓ_1-norm objective function is found in [6]. While the ℓ_1-norm objective function has many advantages, one of the shortcomings is the lack of developed theory concerning the estimated parameter nonlinear confidence bands and noise covariance [125]. It is not within the scope of this work to develop this theory, but it should be noted that the nonlinear parameter confidence regions are considered in this work. The measurement variance is addressed by setting the dead-band region at approximately the same size as the bit pressure measurement noise in the system. This prevents the signal noise from significantly influencing the parameter estimates, only when there is a shift outside the predicted dead-band zone. This practical approach assumes that the parameter estimates are sufficiently accurate when the predicted bit pressure remains within the dead-band region.

5.4.3 Controller Switch

The switch in this simulation uses the logic described in Section 4. Specifically, prior linear model predictions are compared to physical process outputs over a horizon of 20 time steps at each instance. When the bit pressure predictions exceed an absolute error of 1.7 bar, the low-order NMPC controller is implemented in the feedback loop, and the linear model tuning procedure is initiated as described in section 4.3.1. The past prediction horizon and the acceptable prediction error are tuning parameters for this controller. Also, if the MPC optimization solution is infeasible or not convergent for both controllers, then the switch defaults to the empirical controller as the probability of convergence is greater for this controller.

Dynamic Model Identification

In the real time model identification procedure, the switch triggers the high fidelity model to simulate the well at the current conditions. The mud pump flow rate is perturbed in a doublet test (up, down, back to nominal conditions) to generate simulated dynamic data from the high fidelity simulator. In this particular application, the simulated step test covers a 12 time step horizon. This is repeated for the choke pressure, and then the simulated data is used to regress the empirical model parameters. Figure 5.4 shows the step in MVs and the response of the CV for the model identification procedure. Figure 5.4 also shows the fit of the model parameters to the simulated data for a typical online model identification procedure. Figure 5.4 is representative of a typical fitting procedure; however, each fitting instance will vary slightly from the results in this figure. Once the new model parameters are identified, they are sent to the controller, and the switch continues to monitor the accuracy of the model predictions for at least 5 time steps before another switch can be made. This dwell time prevents switch chatter and the instabilities associated with it. In this work, it is assumed that the high fidelity model parameters are accurate and do not require updating in the time scales of these simulations. In longer times scales, such as industrial application, high fidelity parameter model parameters should also be updated.

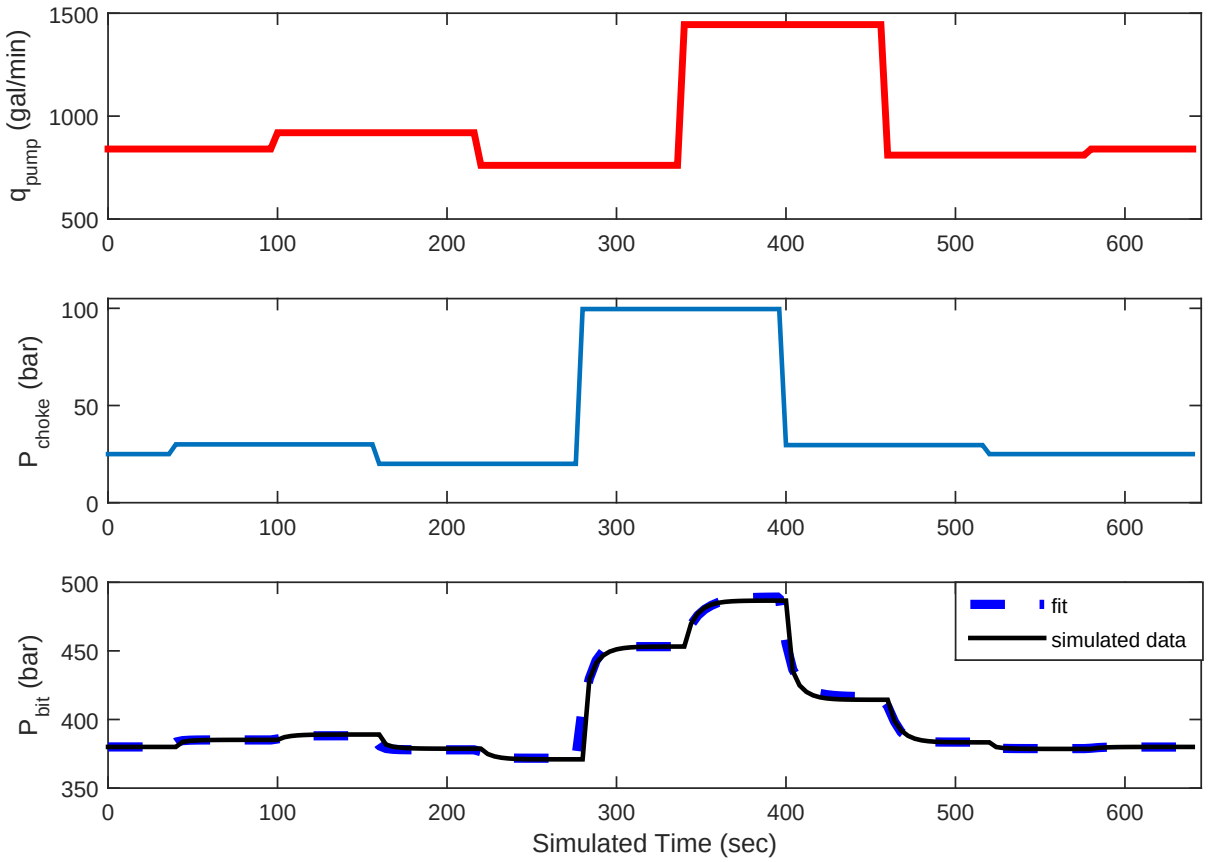


Figure 5.4: Simulated step test, system response, and resulting fit for the empirical model identification.

5.4.4 Oil Well Drilling Process

The drilling process is simulated using the SINTEF Flow Model high fidelity simulator [137] as the plant. A separate instance of the model is used for tuning the linear model. The model parameters that are clearly known *a priori*, such as drill bit diameter and density of the mud in the drillstring, are identical in each instance of the model. However, unknown variables, such as reservoir depth and rock type, and formation pressure are set to different values in the tuning model. This ensures that the model predictions do not contain information that would not be known in practice. Additionally, the low-order and empirical control models are updated with current well information through feedback of the bit pressure. The ROP is held constant at 6.5 ft/hr in the vertical well, and dynamic temperature effects are not included in the simulations.

5.4.5 Simulation Results and Discussion

Two common drilling scenarios are simulated including normal drilling and unwanted gas influx (kick). The results of these simulations are found in Sections 4.5.1 and 4.5.2 respectively.

Normal Drilling Operations

In this simulation, the controller receives measurements from the well every 7 seconds. Figure 5.5 shows the results of the continuous drilling simulation with set point changes. The bit pressure set point for this simulation is 380 bar. At 300 seconds the set point is adjusted to 390 bar and then back to 380 bar at 840 seconds. The top plot of Figure 5.5 shows the bit pressure remains within the dead band set point region even when the empirical control model parameters are being identified and the low-order model is implemented in the feedback loop. The switching between the controllers results in bumpless control. The third and fourth subplots show the movements of the choke pressure and the mud pump flow rate respectively. As seen in the top plot, the controller keeps the bit pressure within the set point range. The second plot shows the individual controller predictions, and at about 370 seconds the empirical controller predictions exceed the acceptable limit. This initiates the tuning procedure. The switching and tuning occurs once again at 966 seconds after the set point change. The switching and tuning does not happen during the set point changes indicating that the loss of control from the empirical model is not due to poor tuning. It is interesting to note that the high fidelity predictions and the empirical model predictions are identical except when the process dynamics change and the empirical model is unable to maintain control. Once the model identification process is complete, the empirical model predictions match the high fidelity model once again. This demonstrates that the control scheme effectively incorporates the high fidelity model predictions into real time control.

Figure 5.6 shows the time of switching between controllers in the top plot, and a comparison of real time and simulation time in the bottom plot. As long as the bottom plot is less than one, as denoted by the black horizontal line, the computation time is faster than real time, and the controller will complete the necessary calculations within the feedback cycle time.

To contrast the switched control scheme, Figure 5.7 shows the results of an NMPC controller using the high fidelity model directly in the optimization routine. Due to the format of the

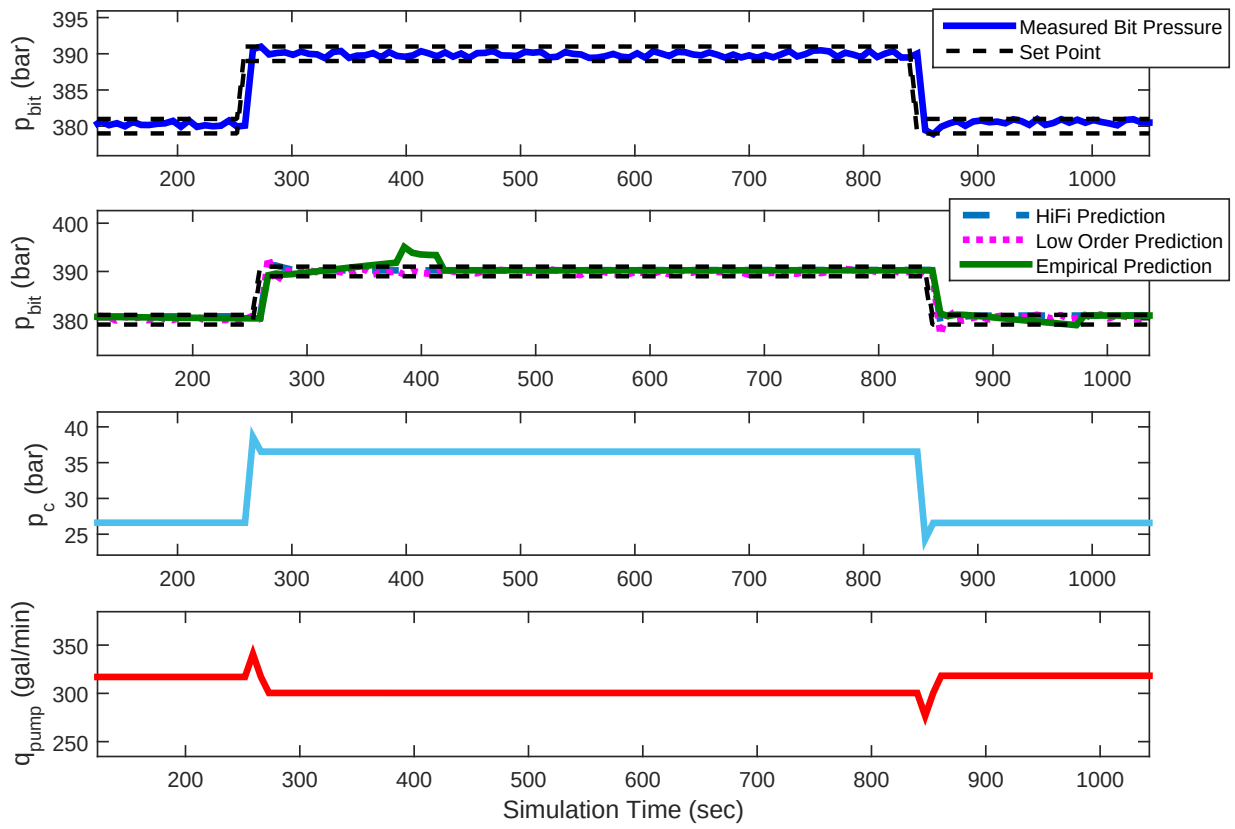


Figure 5.5: Switched controller response to set point changes in bit pressure (P_{bit}).

high fidelity simulator used, this NMPC controller used a shooting method to solve the optimization problem. The control horizon was 4 seconds and was solved using the IPOPT [18] solver option in the *fmincon* function in MATLAB. The controller clearly maintains the bit pressure in the set point region demonstrating very good control.

Figure 5.8 shows a comparison of actual time and simulation time for this simulation, and it is clear that the computation time at each control cycle is too long for real time implementation. This simulation justifies the complexity of this switched control scheme. It implements the accuracy of the high fidelity model for control with significantly reduced computation within the feedback cycle time.

Unexpected Gas Influx

The controller is used in a simulated disturbance rejection scenario. Unexpected gas influx from the reservoir into the well bore is a common occurrence in drilling. In this simulation the

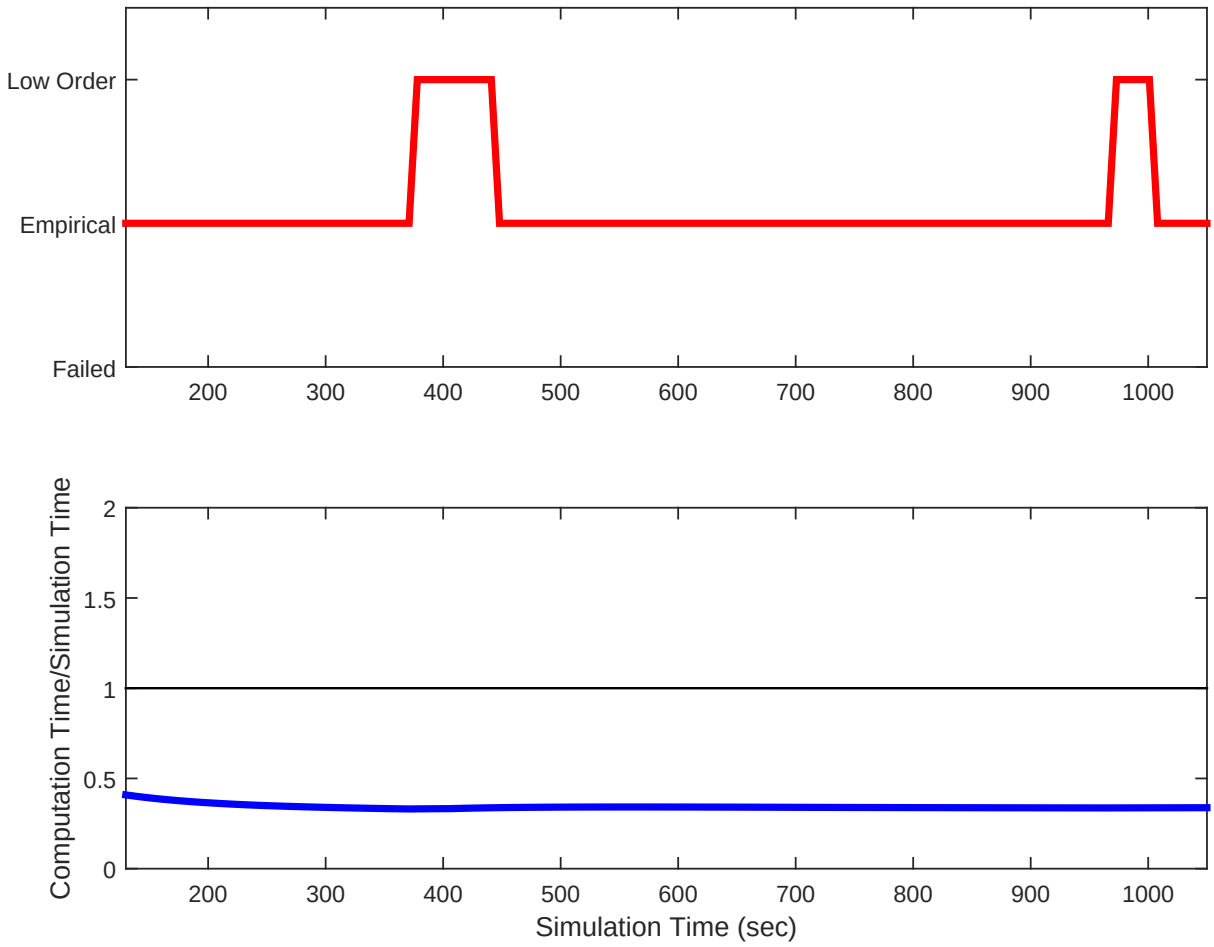


Figure 5.6: Controller switching times and computation time. When the Computation Time / Simulation Time is below the line at 1 on the vertical axis, the computation occurs in real time.

drill bit suddenly hits an unexpected high pressure zone. When this happens, the high pressure reservoir gas disrupts the well pressure balance. Table 2 shows the conditions for this simulation at 130 seconds when the time of interest begins.

Figure 5.9 demonstrates the controller response to this moderate kick. The top subplot of Figure 5.9 shows the controller effectively maintains the bit pressure within the specified set point in the presence of a process disturbance. The kick begins at about 300 seconds and continues throughout the remainder of the simulation (see Figure 5.9). The dashed black lines in Figure 9 denote the set point region. When the bit pressure is within these lines, the controller takes no corrective actions. There is a small ($<$ 2 bar) increase in bit pressure when the kick occurs due to the sudden increase in reservoir pressure encountered by the bit. This pressure increase moves

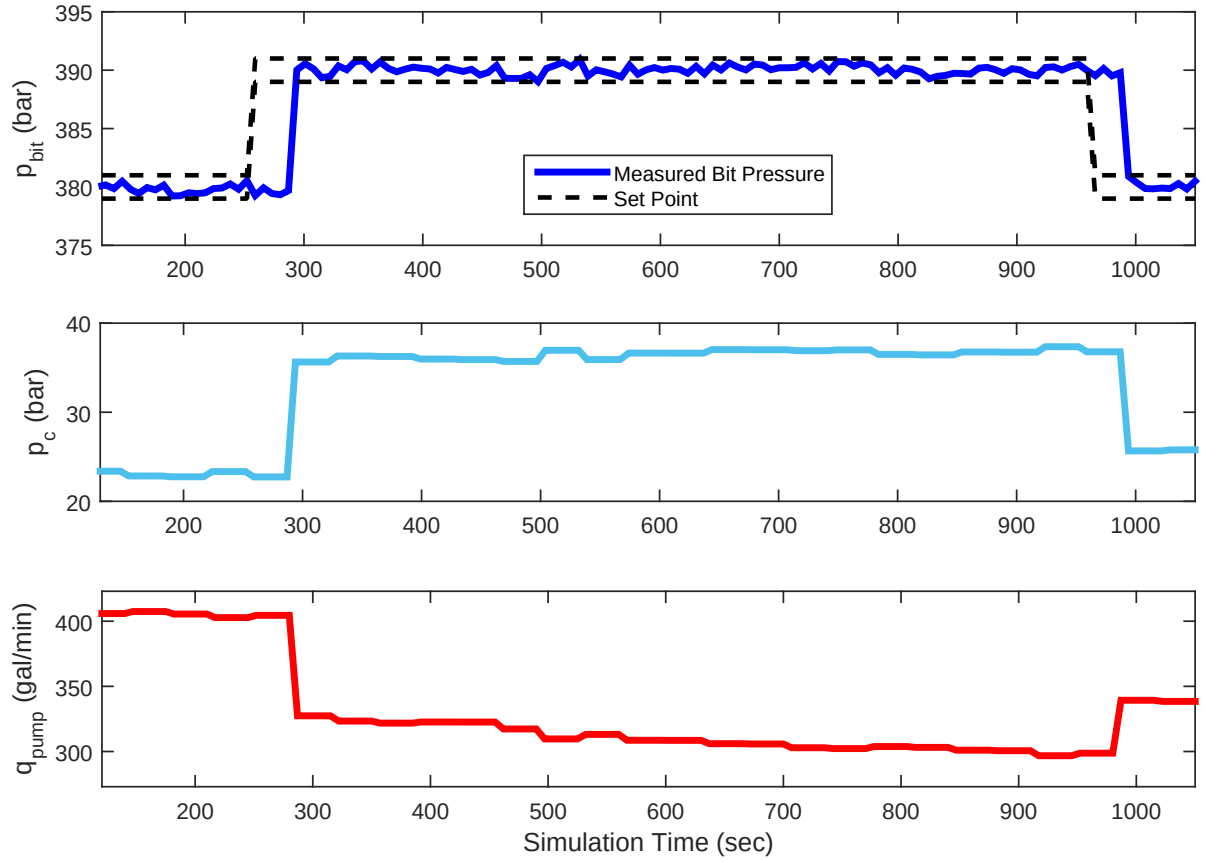


Figure 5.7: MPD control using only a high fidelity control model.

the bit pressure slightly outside of the set point region (at 310 seconds) and provokes a response from the controller as seen in Figure 5.10. Because the mud is oil based and under high pressure, the gas easily dissolves into the fluid which decreases the density of the mud and, consequently, the hydrostatic pressure of the mud. This brings the bit pressure down even though there is little change in the choke pressure or the mud flow. When the bit pressure drops below the set point region at 420 seconds, the controller increases the choke pressure slightly until 450 seconds when a switch is made to the low-order controller for the remainder of the simulation. The controller continues to increase the choke pressure, and eventually the mud flow, because the addition of dissolved gas into the mud column continues to lower the hydrostatic pressure on the bit.

It is interesting to note that the low-order controller is used for disturbance rejection as seen in Figure 5.9. This is because the high fidelity model parameters are not updated by changing process conditions, and it does not account for the disturbance in its predictions. Consequently,

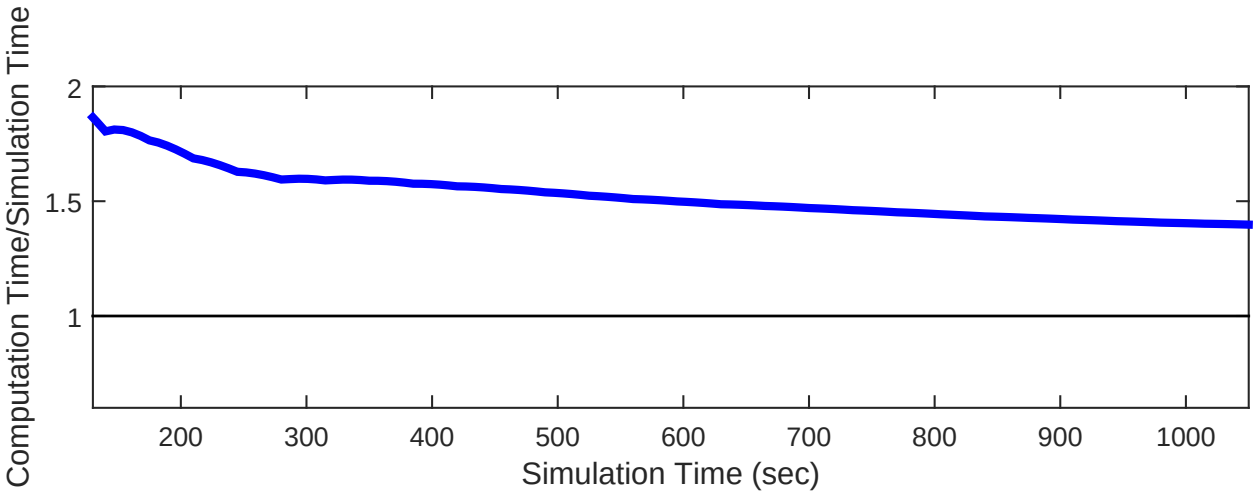


Figure 5.8: High fidelity controller computation time.When the Actual Time / Simulation Time is below the line at 1 on the vertical axis, the computation occurs in real time. This controller could not be implemented in real time.

Table 5.2: Values of key variables used in the gas influx simulation.

Variable	Values at 130 sec. of simulation time
Mud pump flow rate	393.6 gal/min
Choke pressure	22.5 $\text{bar}_{\text{gauge}}$
Reservoir depth	7,874.5 ft
Mud density	11.7 lbs/gal
Bit pressure	378.7 $\text{bar}_{\text{gauge}}$
Well trajectory	Vertical
Well depth (TVD)	7,874.2 ft
ROP	6.5 ft/hr
Kick size	30 bbls
Sample rate	10 sec

the predicted bit pressure is too high and it tunes the empirical model accordingly (see the second subplot in Figure 5.9). At several instances the empirical controller attempts to minimize the error between the predicted bit pressure and the set point, yet as the initial error is greater than the switching tolerance, the controller tunes the empirical model back to the high fidelity model. This continues until the high fidelity model parameters are updated. Meanwhile, the low-order controller is updated by the process, and it is able to keep the bit pressure within the acceptable range. The third subplot in Figure 5.9 shows the choke pressure progressively increases until it

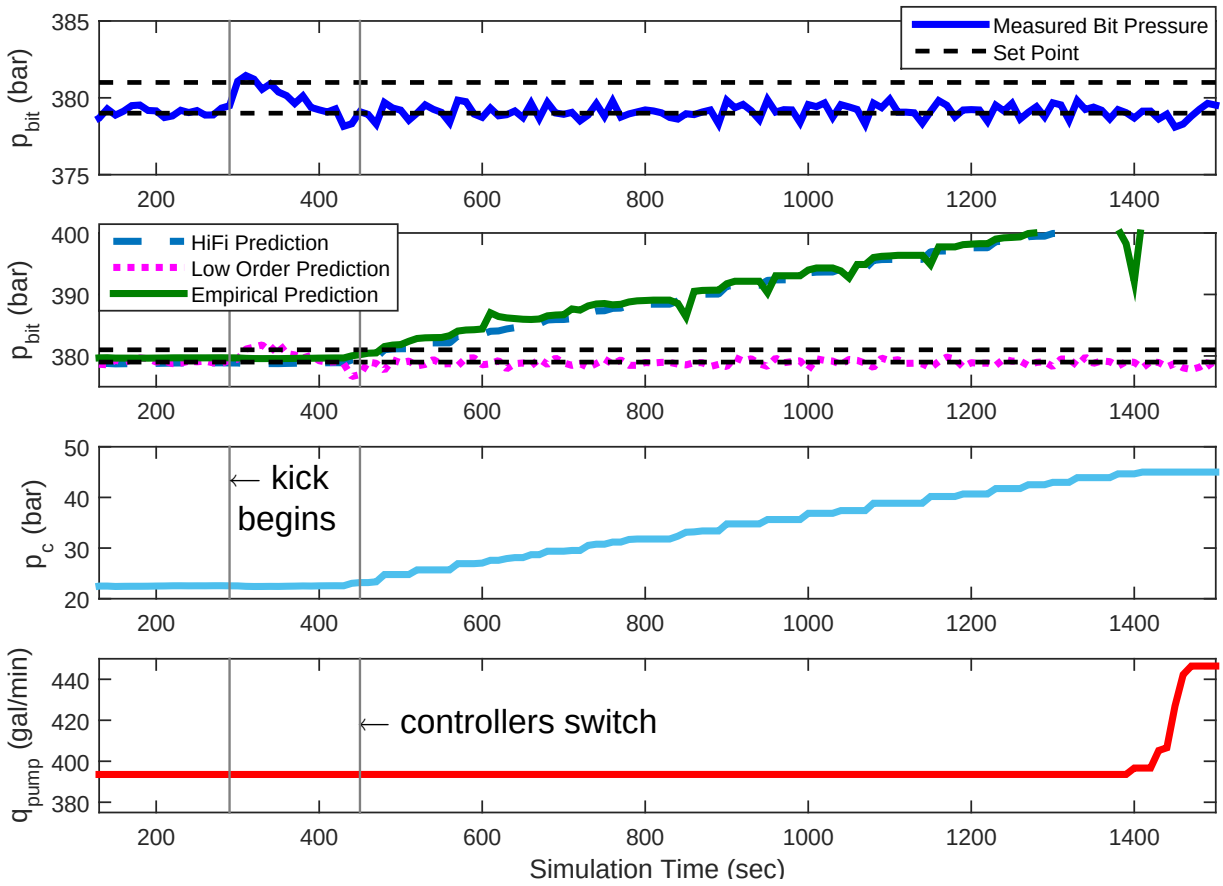


Figure 5.9: Controller response to a process disturbance of unwanted gas influx.

reaches the upper limit of 45 bar. At this point (about 1400 seconds) the choke pressure can no longer be used to maintain the bit pressure and the previously constant mud pump increases flow to compensate.

Figure 5.11 shows the controller switching and also a comparison of simulation time and real time to demonstrate the controller can be used in real time. As long as the bottom plot of Figure 5.11 is less than one, the controller will complete the computations within the required feedback cycle times, and the controller will be stable.

Figure 5.12 shows the drilling mud pit gain, flow through the choke, and the simulated gas influx from the reservoir to the well bore. The sudden increase in pit gain and choke flow are primary signs that a kick is occurring. In a real situation the drilling crew would stop the process and implement the appropriate well control procedures.

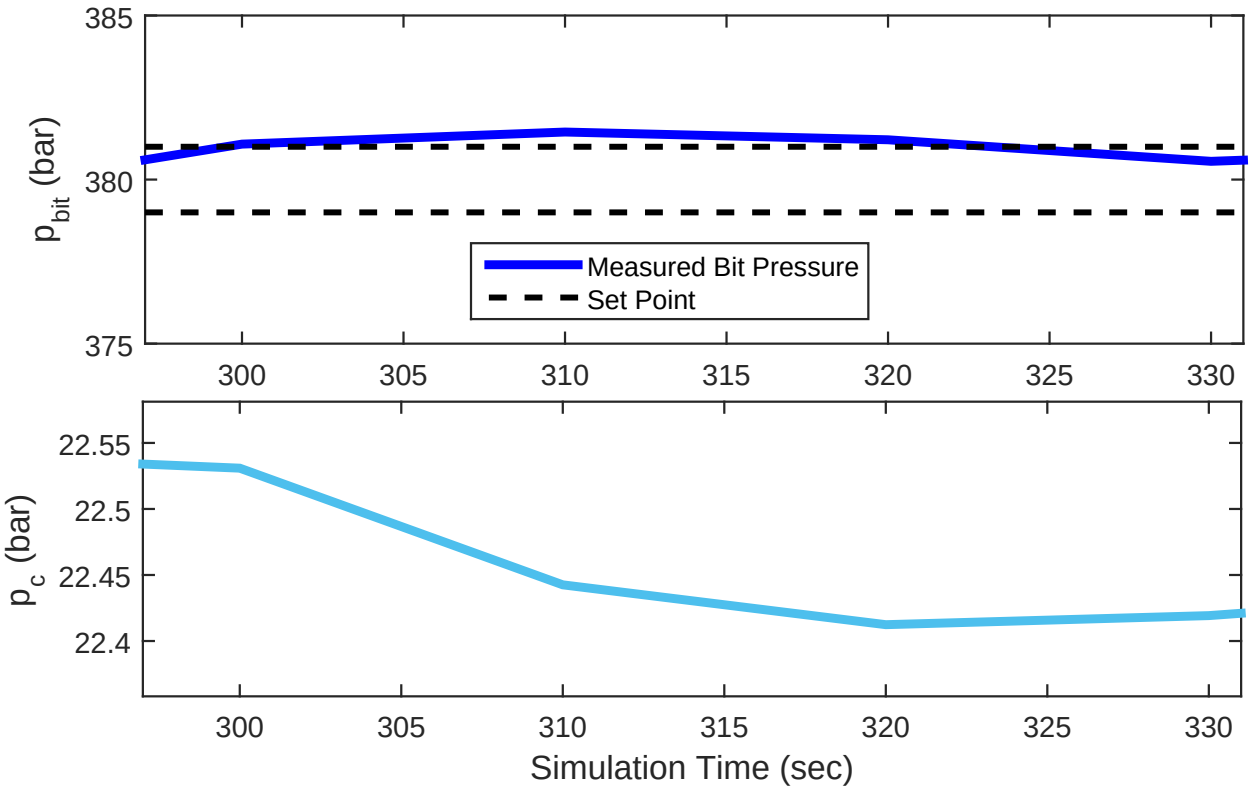


Figure 5.10: Controller response to a set point violation that occurs at the onset of the kick.

5.5 Conclusion

A switched control scheme is presented that makes use of a high fidelity model running in parallel with a process. The high fidelity model is used to generate simulated data to identify the model parameters of a linear empirical control model in MPC. During the model identification procedure, the switch implements a low-order control model to control the process. In this way the model identification procedure does not disrupt the controller. The model identification procedure is triggered when the error in the past predictions of a linear model exceeds a prescribed threshold. The switched control scheme allows the highly accurate predictions of a high fidelity model to be incorporated in real time control without the high online computational cost. The control scheme is applied to a simulated managed pressure drilling process. The controller performance is demonstrated under set point tracking and disturbance rejection scenarios. Future work on the control scheme includes updating the high fidelity model parameters with process data with mov-

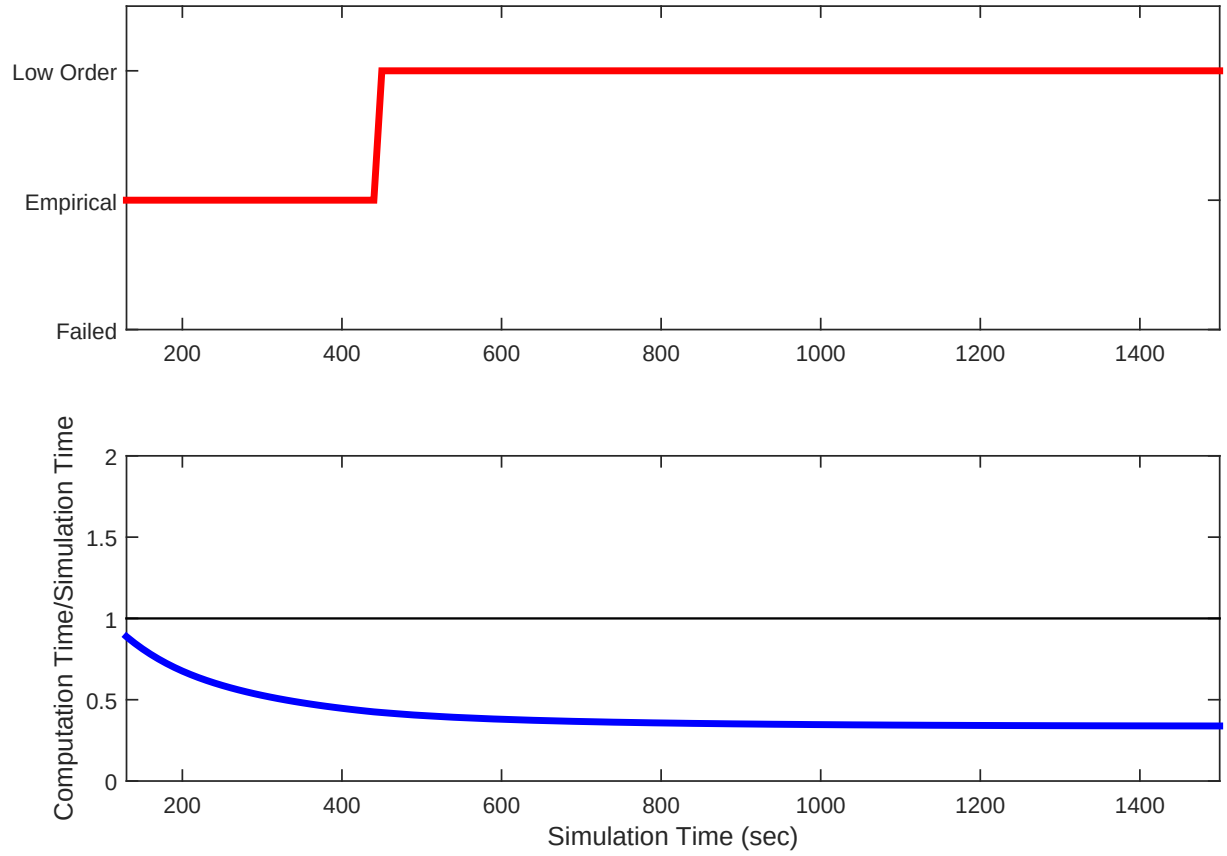


Figure 5.11: Controller switching and simulation time.

ing horizon estimation, and controlling a process when feedback is lost and highly accurate model predictions are necessary.

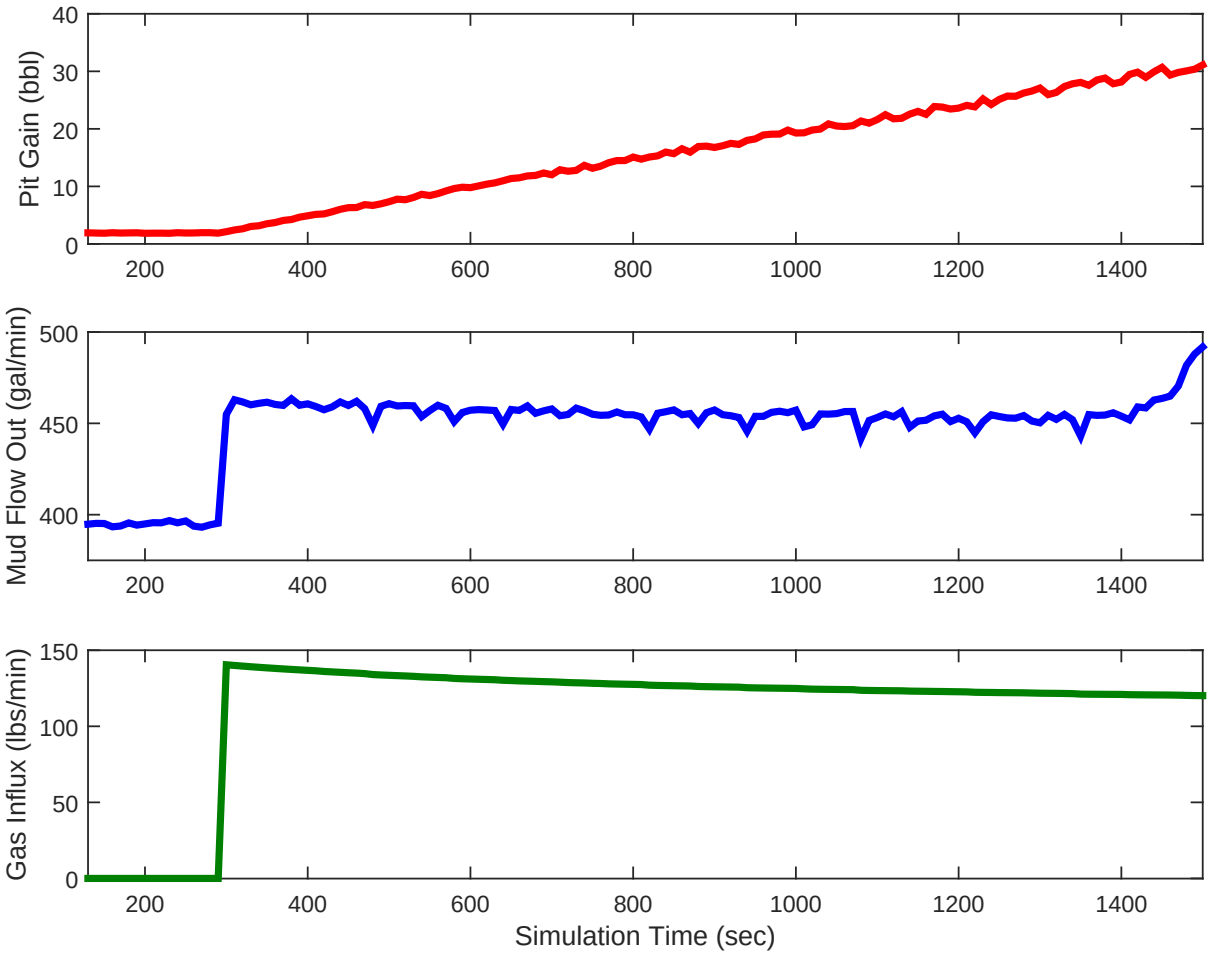


Figure 5.12: Pit gain, choke flow, and gas influx rate for the kick simulation.

CHAPTER 6. CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

The objective of this work is to increase efficiency, safety, and environmental protection through advanced process control of upstream energy production processes. The advanced methods used in this work include multi-fidelity model predictive control, moving horizon estimation, nonlinear gradient based solvers, and predictive switching techniques. Novel algorithms are developed and applied in simulation to subsea riser slugging control and the automated oil well drilling process. While the applications are specific to the upstream oil and gas industry, the control algorithms employed in this work can be used in general applications as well. In addition to novel control algorithms, novel industrially relevant control model parameter estimation techniques are reviewed and heuristics are developed for gain and time constant estimates. In summary, the contributions in this dissertation are:

- The development of a model predictive controller for severe subsea riser slugging control using clamped fiber optic sensors at the riser base. The controller improves performance over a PID controller for the same system.
- A method for identifying the feedback control advantages that come with the location of a pressure sensor on a subsea riser.
- Guidelines for SISO control model identification that encourage a model predictive controller to having acceptable performance and maintain stability under uncertainty.
- A scheme for giving statistical significance to model parameter estimates through the use of nonlinear confidence intervals.

- A basic switched control algorithm that uses multi-fidelity models in real-time feedback control. The controller can maintain control without feedback by utilizing the predictions of a high fidelity simulator.

- A procedure for executing smooth switches between independent model predictive controllers.

- An advanced switched control algorithm that expands the capabilities of the basic switched controller. The predictions from a high fidelity simulator can be incorporated directly into the control law. This controller does not require predetermined switching points as the switching is determined by the accuracy of the current control model predictions.

- A technique for online control model parameter estimation without disrupting the process. A high accuracy model of the process is used to generate simulated data which is then used to identify a control model based off of the current operating region.

A model predictive controller for severe subsea riser slugging mitigation is introduced. The controller uses recent advances in post-installed fiber optic sensor clamp design. These clamps allow the addition of a pressure, temperature, and strain sensor at virtually any location on the riser. Pressure sensing is necessary for acceptable feedback control of the process, and a sensitivity analysis showing the effect of sensor location on controller response is reported. The results for a 4300 meter riser show that when a sensor is placed 33.5 meters above the riser base the controller response begins to deteriorate with persistent minor offset, but is still acceptable. When the sensor is placed 167 meters above the riser base the controller is unable to suppress the slugging or follow step changes in the set point. With a sensor near the riser base, the MPC controller improves the settling time by about 5% and has little persistent offset compared to a PID controller for the same process.

A review of commonly used industrial estimation algorithms and the associated benefits and drawbacks is presented. These methods include filtered bias update, Implicit Dynamic Feedback, Kalman filtering, squared error moving horizon estimation, and ℓ_1-norm moving horizon estimation. The benefits and drawbacks of each technique are outlined, and an example drilling automation application demonstrates each technique's characteristics. The example demonstrates the performance of each estimator in the presence of measurement noise, drift, and outliers. The

example shows that the ℓ_1-norm moving horizon estimator provides a more accurate estimate of the true process state for short periods of bad data than the other methods. Also, the relevance of estimation methods to control model parameter updates is discussed. Guidelines are developed for the effect parameter estimation error will have on controller performance. The analysis reveals that the combination of overestimated gain and underestimated time constant leads to the least amount of error in the controller. When considered independently, underestimated gains lead to increased error, while overestimated time constants immediately lead to significant increased error in single input/single output systems. Put in more quantitative terms, if the time constant is correctly estimated and the gain is overestimated 50%, then the controller error will be 2.5%. Whereas, if the gain is underestimated by 50%, then the controller error is 20%. Similarly, if the gain is correctly estimated and the time constant is over estimated by 100%, then the controller error is 15%, while if overestimated by 100% the error is 8%. These results support the conventional process control heuristics of estimated gain needing to be within 30% of the actual gain, and the time constant being within 50% for sufficiently effective control.

A basic switching control algorithm that makes use of the individual strengths of models with varying fidelity was developed. The algorithm takes advantage of the highly accurate predictions of a high fidelity model, the fast computation time of a nonlinear reduced order model, and the guaranteed convergence of a linear empirical model. Additionally, the control structure offers the ability to tune and readjust one model while another is used for control, without interrupting the process. It also offers the benefit of increased reliability generally associated with redundant hardware and safety systems. The control structure consists of three separate MPC controllers and a switch that selects one of the controllers to implement in the process. Each controller has one of the previously mentioned models, and each receives measurements from the process. For this basic switch, the model with the highest fidelity always has first priority; however, it is not available at every time step due to the required computation time. The reduced order model has second priority, and is implemented unless it does not converge to a solution. Lastly, the linear empirical model is used when the others are not available. A novel method for bumpless switching among controllers is presented. Each MPC optimization problem is started, at each time step, with the current process conditions as the initial conditions. This forces the solvers to find similar solutions

as each begins from the same initial values. This results in a smooth transition from one controller to the next.

The bumpless switching method and controller are applied to a MPD simulation that uses the SINTEF Flow Model high fidelity simulator as the oil well process, and the simulation includes the addition of measurement signal noise. The objective of the controller is to keep the pressure at the drill bit within ± 1 bar of the target pressure. A simulated failure of the high fidelity and low order models demonstrate the usefulness of the controller. The bit pressure is maintained in the acceptable range, and the transition between control models is smooth. During a simulated pipe connection procedure, there is no measurement feedback to the controller. Using the highly accurate predictions of the high fidelity simulator, the controller is able to maintain the bit pressure within ± 5 bar of the set point. This basic switched control algorithm preceded a more complex switched controller.

This work introduces an advanced switched control method that uses multi-fidelity control models: high fidelity, nonlinear reduced order, and linear empirical. The objective in developing this controller is to enhance performance and reliability by incorporating high fidelity models directly into the control law. Accordingly, the algorithm employs the same bumpless switching technique described in Chapter 4. However, instead of the predetermined switching sequence used in the basic controller, the advanced algorithm uses the linear empirical controller when possible. When controller performance becomes unacceptable, the algorithm implements the low order model to control the process while the high fidelity model generates simulated data which is used to estimate the empirical model parameters. Once this online model identification process is complete, the controller reinstates the empirical model to control the process. This control framework allows the widely accurate, yet computationally expensive, predictive capabilities of the high fidelity simulator to be incorporated into the locally accurate linear empirical model while still maintaining solver convergence guarantees. The entire process is done online, and in real time.

The advanced switching algorithm is demonstrated in a MPD application. Two common drilling scenarios are simulated including normal drilling and unwanted gas influx, also known as kick. In these simulations, the objective of the controller is to maintain the drill bit pressure within ± 1 bar of the set point. The normal drilling simulation gives a basic demonstration of the switching procedure, and shows that it can track changing set points in real time. Once the

empirical controller is unable to keep the bit pressure within the acceptable limits, the switching and tuning procedures are triggered. The kick simulation shows the controller ability to reject process disturbances, and also highlights some of the algorithms shortcomings. One drawback is that high fidelity model parameters are not updated with changing process conditions. Updating the model parameters is possible, but will require more computation, and is left for future work.

6.2 Future Work

This research has made significant advances in controller switching, high fidelity simulators in real time control, and model estimation heuristics. Yet, to more fully develop these areas, more work is required. This section discusses areas of future development in this work. It is divided into three subsection: Control Algorithm developments, Application Specific developments, and Theoretical developments.

6.2.1 Control Algorithms

A pipeline wax deposition controller could have a high impact if developed in the future. It is possible to use fiber optic temperature sensors on the exterior of the pipeline to give a feedback measurement for a MPC controller. As the wax deposit inside the pipeline continues to grow it insulates the exterior of the pipe from the warmer oil flowing on the interior. Thus, temperature could be used as a surrogate for wax deposition thickness. When the wax reaches a set tolerance, the controller would trigger remedial action such as sending a pig to clean the pipeline. The inclusion of high fidelity wax deposition models to predict the deposit growth time and thickness would help in the optimization of such a controller.

The advanced switched controller presented in Chapter 5 lacks one key aspect. The high fidelity model parameters are not updated with changing process conditions. This is made clear in the unwanted gas influx simulation. As seen in that simulation, not updating the model parameters leads to incorrect tuning of the empirical controller, and possibly a loss of control. To address this issue, some preliminary work has been done using a Monte Carlo approach to estimate the high fidelity model annulus friction factor and density. This approach requires large computational resources, and parallel programming in the estimation algorithm. For more details on this method

see: Aghito, Manuel, Eaton, Ammon N., Bjørkevoll, Knut S., Nybø, Roar, and Hedengren, John D., *Automatic Model Calibration for Drilling Automation*, SPE-185926-MS, 2017. Another approach is to use a separate instance of the high fidelity model in a MHE scheme that would calculate the necessary parameters, instead of using trial-and-error methods like the Monte Carlo simulation. This more powerful MHE technique would require a versatile high fidelity simulator that can output the parameters that are being estimated in the MHE, yet also use those variables as inputs for the controller. This is difficult to do because many current high fidelity drilling simulators use tabulated data for temperature and pressure dependent fluid properties. While this capability currently does not exist in the SINTEF Flow model, efforts are being made to incorporate this feature into a new model that is being developed for control purposes. Regardless of the method, it is important to update the high fidelity model parameters for stable process control.

Another shortcoming of this work is the lack of validation through experiment. This work has focused on control algorithm development, and has used low and high fidelity simulators as the simulated plant. This approach is acceptable for creating novel control algorithms, but the algorithms can be further refined by implementation in actual processes, including processes not considered in this work. Lab scale and full scale implementations can give further insights into the algorithm behavior, and can lead to significant improvements.

6.2.2 Application Specific

There are several potential areas for future controller development for the riser slugging and drilling applications in this work. One area that would greatly improve the slugging controller is the addition of more complete models in the simulation. The model used in this work is restricted to constant flow and pressure values from the well. This constraint does not allow the controller to maximize production from the well in addition to suppressing severe riser slugs. Including a plant model that can allow maximization of well production would also call for a first principles model in the controller. This area of algorithm development is ripe for contributions.

Another potential area of research in slugging control is the automation and optimization of the offshore platform receiving and separation facilities. Models of the processes need to be developed and fit for control purposes before control algorithm development can begin. This area would combine well with the maximization of well production previously mentioned.

In the drilling application, the use of the back pressure pump as the main mechanism for controlling the bit pressure should be explored. This is because the main mud pump is used for more than just pressure control. It also removes the rock cuttings from the well bore so the bit does not get stuck. Cleaning the well bore is crucial in drilling, and adjusting the flow rate of the main pump can cause cleaning issues. For example, a sudden drop in the main pump flow rate to maintain the bit pressure within the given bounds may result in insufficient flow to keep the rock cuttings suspended in the mud and moving towards the surface. This leads to undesired and expensive disruptions to the drilling process. Setting the main mud pump to a constant and sufficiently high flow rate, and using the choke valve and back pressure pump for pressure control may lead to improved automated drilling. The challenge in this scenario will be maintaining control when the bit pressure needs to drop below the pressure caused by the main mud pump flow.

Another potential direction for future research in switched control of drilling is combining drillstring mechanics and pressure control to take advantage of multivariate effects. The foundations of this concept were formed in previous research [167], and could be joined with the algorithms in this work to maximize ROP and control wellbore pressure in a robust and reliable manner needed for harsh oilfield conditions.

6.2.3 Theory

One future theoretical development of the advanced switched controller should be a more formal stability proof. The stability of the controller is discussed in Chapter 5, but a rigorous stability guarantee is not provided. This can be done by considering the stability of each of the individual MPC controllers and the stability of the switch. One way to do this is to find a common nonlinear Lyapunov function, or proving stability through the inherent gradient decent properties of the nonlinear solvers used in the MPC algorithms.

This work uses the ℓ_1-norm in most of the MPC objective functions. Stability and robustness theory has been developed for linear ℓ_1-norm objective functions [168], but not for nonlinear counterparts. While work has been done on the stability of NMPC, it centers on the ℓ_2-norm objective function. Extending these stability guarantees to include the ℓ_1-norm objective function is a potential area of future research.

The guidelines for linear SISO MPC control model parameter estimation developed in this work could be extended to MIMO systems. Additionally, nonlinear and perhaps even first principles based model estimation bounds could be studied and guidelines established. Further development of these guidelines could include formulas for determining the bounds of model parameters that will ensure control.

Additionally, the development of statistical analysis for online model parameter estimates can give greater confidence in the estimates. Further development of this technique, including using the estimation statistics to determine the need for model tuning, is a potential direction of future work. In this scenario, the parameter estimate confidence regions are used for logic decisions in the controller, not just estimate information. Further developments could include the formulation of parameter joint confidence regions for estimates using the ℓ_1-norm. Currently, the nonlinear joint confidence theory is based on the f statistic. The f statistic relies on the ℓ_2-norm for its significance. This is due to the tenets of the Central Limit Theorem and its foundation in the variance, which is the square of the standard deviation. It would be necessary to reformulate the Central Limit Theorem in terms of the ℓ_1-norm, or else develop the joint confidence regions in a manner that does not use the f statistic.

REFERENCES

[1] U.S. Energy Information Administration, "International Energy Outlook, Chapter 1," 2016. viii, 1

[2] K. Jeffrey and K. Forward, "Improvements with broadband networked drill string," *Digital Energy Journal*, vol. 18, pp. 7–8, April 2009. ix, 45

[3] R. Hutin, R. Tennent, and S. Kashikar, "New mud pulse telemetry techniques for deepwater applications and improved real-time data capabilities," in *SPE/IADC Drilling Conference*, no. 67762-MS. Amsterdam, Netherlands: Society of Petroleum Engineers, February 2001. ix, 45

[4] D. Morris, "Analysis of Well Control Incidents: 2007-2013, U.S. Department of the Interior," in *SPE ATCE Conference and Exhibition*, Amsterdam, Netherlands, 2014. 1, 73

[5] J.-M. Godhavn, A. Pavlov, G.-O. Kaasa, and N. L. Rolland, "Drilling seeking automation control solutions," in *18th IFAC World Congress*, Milano, Italy, 2011. 2, 73

[6] J. D. Hedengren, R. A. Shishavan, K. M. Powell, and T. F. Edgar, "Nonlinear modeling, estimation and predictive control in APMonitor," *Computers & Chemical Engineering*, vol. 70, pp. 133 – 148, 2014, Manfred Morari Special Issue. 3, 31, 57, 58, 59, 81, 100, 103

[7] W. J. Rugh and J. S. Shamma, "Research on gain scheduling," *Automatica*, vol. 36, no. 10, pp. 1401 – 1425, 2000. 3

[8] Y. Zhu, "Estimation of an NLN HammersteinWiener model," *Automatica*, vol. 38, no. 9, pp. 1607 – 1614, 2002. 3

[9] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003. 4, 91

[10] B. W. Bequette, "Nonlinear control of chemical processes: a review," *Industrial & Engineering Chemistry Research*, vol. 30, no. 7, pp. 1391–1413, 1991. 4

[11] C. Pantelides and J. Renfro, "The online use of first-principles models in process operations: Review, current status and future needs," *Computers & Chemical Engineering*, vol. 51, pp. 136–148, 2013. 5, 91

[12] C. Runge, "Ueber die numerische auflösung von differentialgleichungen," *Mathematische Annalen*, vol. 46, no. 2, pp. 167–178, 1895. 5

[13] G.F.Carey and B. A. Finlayson, "Orthogonal collocation on finite elements," *Chemical Engineering Science*, vol. 30, pp. 587–596, 1975. 6

[14] R. Lobatto, "Lessen over de differentiaal- en integraalrekening," *Gravenhage*, vol. 1, 1851. 7

[15] ——, "Lessen over de differentiaal- en integraalrekening," *Gravenhage*, vol. 2, 1852. 7

[16] K. M. Powell, A. N. Eaton, J. D. Hedengren, and T. F. Edgar, "A continuous formulation for logical decisions in differential algebraic systems using mathematical programs with complementarity constraints," *Processes*, vol. 4, no. 7, 2016. 7, 49

[17] J. Martins, A. Ning, and J. Hicken, *Multidisciplinary Design Optimization*. Unpublished, Compiled January 5, 2017. 9

[18] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006. 10, 107

[19] J. Hedengren, "APMonitor modeling language for mixed-integer differential algebraic systems," in *Computing Society Session on Optimization Modeling Software: Design and Applications, INFORMS National Meeting*, Phoenix, AZ, October 2012. 10

[20] S. M. Safdarnejad, J. D. Hedengren, N. R. Lewis, and E. L. Haseltine, "Initialization strategies for optimization of dynamic systems," *Computers & Chemical Engineering*, vol. 78, pp. 39 – 50, 2015. 12

[21] J.-M. Godhavn, M. P. Fard, and P. H. Fuchs, "New slug control strategies, tuning rules and experimental results," *Journal of Process Control*, vol. 15, no. 5, pp. 547–557, 2005. 14, 26

[22] F. P. Donohue, "Classification of flowing wells with respect to velocity," *Petroleum Transactions*, vol. 86, pp. 226–232, 1930. 14

[23] P. Hedne and H. Linga, "Suppression of terrain slugging with automatic and manual riser choking," in *ASME Winter Annual Meeting*, Dallas, Texas, 1990. 14, 26

[24] J. G. Skofteland and J.-M. Godhavn, "Supression of slugs in mulitphase flow lines by active use of topside choke - field experience and experimental results," in *11th BHR Group Multiphase Production International Conference*, San Remo, Italy, 2003. 14, 26

[25] R. Eslamloueyan and E. Hosseinzadeh, "Using neural network predictive control for riser-slugging suppression," *Chemical Product and Process Modeling*, vol. 4, no. 4, 2009. 14, 26

[26] E. Jahanshahi and S. Skogestad, "Comparison between nonlinear model-based controllers and gain-scheduling internal model control based on identified model," in *52nd IEEE Conference on Decision and Control*, Florence, Italy, 2013. 14, 26

[27] F. D. Meglio, N. Petit, V. Alstad, and G.-O. Kassa, "Stabilization of slugging in oil production facilities with or without upstream pressure sensors," *Journal of Process Control*, vol. 22, pp. 809–822, 2012. 14

[28] S. Keerthi and E. Gilbert, "Optimal infinite-horizon feedback laws for a general class of contrained discrete-time systems: stability and moving-horizon approximations," *Journal of Optimization Theory and Applications*, vol. 57, pp. 265–293, 1988. 16, 91

[29] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with garanteed stability," *Automatica*, vol. 34, pp. 1205–1217, 1998. 16, 91

[30] F. Tahir and I. M. Jaimoukha, "Causal state-feedback parameterizations in robust model predictive control," *Automatica*, vol. 49, no. 9, pp. 2675–2682, 2013. 16, 17, 91

[31] P. Scokaert, D. Mayne, and J. Rawlings, "Suboptimal model predictive control (feasibility implies stability)," *IEEE Transactions on Automatic Control*, vol. 44, no. 3, pp. 648–654, 1999. 17

[32] W. S. Yip and T. E. Marlin, "The effect of model fidelity on real-time optimization performance," *Computers & Chemical Engineering*, vol. 28, pp. 267–280, 2004. 17, 92

[33] J. Pannek and K. Worthmann, "Stability and performance guarantees for model predictive control algorithms without terminal constraints," *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift fr Angewandte Mathematik und Mechanik*, vol. 94, no. 4, pp. 317–330, 2014. 17

[34] G. Don M. Hannegan, Wanzer, "Well control considerations - offshore applications of underbalanced drilling technology," in *SPE/IADC Drilling Conference*, Amsterdam, The Netherlands, 2003. 18

[35] Schlumberger, "Drillstring vibrations and vibration modeling, www.slb.com/drillingop," accessed on May 7, 2015. [Online]. Available: www.slb.com/drillingop 19

[36] R. Asgharzadeh Shishavan, C. Hubbell, H. Perez, J. D. Hedengren, and D. Pixton, "Combined rate of penetration and pressure regulation for drilling optimization by use of high-speed telemetry," *SPE Drilling and Completion Journal*, vol. 30, no. 1, pp. 17–26, 2015. 19, 96

[37] F. Allgöwer, T. A. Badgwell, J. S. Qin, J. B. Rawlings, and S. J. Wright, "Nonlinear predictive control and moving horizon estimation: an introductory overview," in *Advances in control*. Springer, 1999, pp. 391–449. 20, 45

[38] R. A. Shishivan, D. Brower, J. Hedengren, and A. Brower, "New advances in post-installed subsea monitoring systems for structural and flow assurance evaluation," in *ASME 33rd International Conference on Ocean, Offshore & Arctic Engineering (OMAE)*, San Francisco, California, June 2014. 27, 36

[39] E. Storkaas, "Stabilizing control and controllability: Control solutions to avoid slug flow in pipeline-riser systems," Ph.D. dissertation, Norwegian University of Science and Technology, 2005. 27, 29, 31

[40] J. D. Hedengren, R. A. Shishavan, K. M. Powell, and T. F. Edgar, "Nonlinear modeling, estimation and predictive control in APMonitor," *Computers & Chemical Engineering*, vol. 70, pp. 133–148, 2014. 31

[41] D. Brower, C. Prescott, J. Zhang, C. Howerter, and D. Rafferty, "Real-time flow assurance monitoring with non-intrusive fiber optic technology," in *Offshore Technology Conference*, no. 10.4043/17376-MS, Houston, Texas, May 2005. 37, 41, 43

[42] L. Grabarski, J. C. Silva, E. Cacao, A. S. Paterno, and H. J. Kalinowski, "Fiber bragg grating temperature sensors used to measure flow in a pipeline," in *Microwave and Optoelectronics Conference, 2007. IMOC 2007. SBMO/IEEE MTT-S International*. IEEE, 2007, pp. 379–383. 37

[43] F. A. Vargas, D. G. Lopes, P. P. Kenedi, J. Clevelario, and F. de Souza Pires, "Experimental comparison of tensile armor wires using strain gages and Fiber Bragg grating techniques," in *ASME 2014 33rd International Conference on Ocean, Offshore and Arctic Engineering*. American Society of Mechanical Engineers, 2014, pp. V06AT04A050–V06AT04A050. 37

[44] M. Optics, "Optical strain gage os3155," 2015, [Accessed 13 Feb 2017]. [Online]. Available: http://www.micronoptics.com/wp-content/uploads/2015/12/os3155_1512.pdf 37

[45] ——, "Non-metallic temperature sensor os4300," 2015, [Accessed 13 Feb 2017]. [Online]. Available: http://www.micronoptics.com/wp-content/uploads/2015/12/os4300_1512.pdf 37

[46] A. F. Bower, *Applied mechanics of solids*. Boca Raton: CRC Press, 2010. [Online]. Available: http://opac.inria.fr/record=b1133709 37

[47] T. Santosham and H. Ramsey, "The dynamic elastic behavior of mild steel, aluminum, and copper as observed in wave propagation tests," *International Journal of Mechanical Sciences*, vol. 11, no. 9, pp. 751–765, 1969. 38

[48] D.-F. Nie, J. Zhao, T. Mo, and W.-X. Chen, "Room temperature creep and its effects on flow stress in x70 pipeline steel," *Material Letters*, vol. 62, no. 1, pp. 51–53, 2008. 38

[49] R. G. Budynas, J. K. Nisbett, and J. E. Shigley, *Stress in Pressurized Cylinders. Shigley's Mechanical Engineering Design. 9th Edition*. New York: McGraw-Hill, 2011. 39

[50] P. Singh, R. Venkatesan, and H. S. Fogler, "Formation and aging of incipient thin film wax-oil gels," *AIChE Journal: Materials, Interfaces, and Electrochemical Phenomena*, vol. 46, no. 5, pp. 1059–1074, 2000. 40

[51] R. Venkatesan and J. Creek, "Wax deposition during production operations: SOTA," in *Offshore Technology Conference*, Houston, Texas, USA, 2007. 40

[52] Z. Huang, H. S. Lee, M. Senra, and H. Scott Fogler, "A fundamental model of wax deposition in subsea oil pipelines," *AIChE Journal*, vol. 57, no. 11, pp. 2955–2964, 2011. 40

[53] K. Ivanhoe and J. Herman, "Paraffin, asphaltene control practices surveyed," July 1999, [Accessed 21 Jan 2017]. [Online]. Available: http://www.ogj.com/articles/print/volume-97/issue-28/in-this-issue/production/paraffin-asphaltene-control-practices-surveyed.html 41

[54] Tracerco, "World's first subsea CT scanner launches at UTC," July 2013, [Accessed 23 Jan 2017]. [Online]. Available: http://www.tracerco.com/news/worlds-first-subsea-ct-scanner-launches-at-utc 41, 43

[55] O. O. Bello, S. O. Fasesan, C. Teodoriu, and K. M. Reinicke, "An evaluation of the performance of selected wax inhibitors on paraffin deposition of Nigerian crude oils," *Petroleum Science and Technology*, vol. 24, no. 2, pp. 195–206, 2006. 43

[56] Halliburton, "SureTherm service: Targeted heat placement in remote locations to remove pipeline deposits," 2012, [Accessed 25 Jan 2017]. [Online]. Available: http://www.halliburton.com/public/bc/contents/Data_Sheets/H07615.pdf 43

[57] M. Wilson, "Radioisotope technology helps ensure pipeline flow," February 2011, [Accessed 25 Jan 2017]. [Online]. Available: http://www.offshore-mag.com/articles/print/volume-71/issue-2/flowlines-__pipelines/radioisotope-technology-helps-ensure-pipeline-flow.html 41, 43

[58] S. Mokhatab and B. Towler, "Wax prevention and remediation in subsea pipelines and flowlines," *Deepwater Technology*, vol. 230, no. 11, 2009. 43

[59] APSensing, "Pipeline monitoring," [Accessed 25 Jan 2017]. [Online]. Available: https://www.apsensing.com/application/pipeline-monitoring/ 43

[60] M. Halstensen, B. K. Arvoh, L. Amundsen, and R. Hoffmann, "Online estimation of wax deposition thickness in single-phase sub-sea pipelines based on acoustic chemometrics: A feasibility study," *Fuel*, vol. 105, pp. 718–727, 2013. 41, 43

[61] A. Aiyejina, D. P. Chakrabarti, A. Pilgrim, and M. Sastry, "Wax formation in oil pipelines: A critical review," *International Journal of Multiphase Flow*, vol. 37, no. 7, pp. 671 – 694, 2011. 43

[62] M. Volk, J. Henshaw, and M. B. Iwata, "Technologies of the future for pipeline monitoring and inspection," Research Partnership to Secure Energy for America, Tech. Rep. 08121-2902-02, 2012. 41, 43

[63] R. C. Sarmento, G. A. S. Ribbe, and L. F. A. Azevedo, "Wax blockage removal by inductive heating of subsea pipelines," *Heat Transfer Engineering*, vol. 25, no. 7, pp. 2–12, 2004. 43

[64] X. D. Chen, D. X. Li, S. X. Lin, and N. zkan, "On-line fouling/cleaning detection by measuring electric resistanceequipment development and application to milk fouling detection and chemical cleaning monitoring," *Journal of Food Engineering*, vol. 61, no. 2, pp. 181 – 189, 2004. 41, 43

[65] R. Hoffmann, L. Amundsen, and R. Schüller, "Online monitoring of wax deposition in subsea pipelines," *Measurement Science and Technology*, vol. 22, no. 7, pp. 671 – 694, 2011. 41, 43

[66] A. Cordoba and C. Schall, "Application of a heat transfer method to determine wax deposition in a hydrocarbon binary mixture," *Fuel*, vol. 80, no. 9, pp. 1285 – 1291, 2001. 41, 43

[67] J. S. Gudmundsson and I. Durgut, "Detection and monitoring of deposits in multiphase flow pipelines using pressure pulse technology," in *12th International Oil Field Chemistry Symposium*, Geilo, Norway, 2001. 41, 43

[68] K. Edalati, N. Rastkhah, A. Kermani, M. Seiedi, and A. Movafeghi, "The use of radiography for thickness measurement and corrosion monitoring in pipes," *International Journal of Pressure Vessels and Piping*, vol. 83, no. 10, pp. 736 – 741, 2006. 41, 43

[69] A. Gleiter and G. Mayr, "Thermal wave interference," *Infrared Physics & Technology*, vol. 53, no. 4, pp. 288 – 291, 2010. 41, 43

[70] M. Zaman, N. Bjorndalen, and M. R. Islam, "Detection of precipitation in pipelines," *Petroleum Science and Technology*, vol. 22, no. 9-10, pp. 1119–1141, 2004. 41, 43

[71] S. E. Guthrie, G. Mazzanti, T. N. Steer, M. R. Stetzer, S. P. Kautsky, H. Merz, S. H. J. Idziak, and E. B. Sirota, "An *in situ* method for observing wax crystallization under pipe flow," *Review of Scientific Instruments*, vol. 75, no. 4, 2004. 41, 43

[72] J. Sugiura, R. Samuel, J. Oppelt, G. P. Ostermeyer, J. D. Hedengren, and P. Pastusek, "Drilling modeling and simulation: Current state and future goals," in *SPE/IADC Drilling Conference and Exhibition*, no. SPE/IADC-173045-MS, London, UK, March 2015. 44, 74

[73] R. Nybø, J. Frøyen, A. D. Lauvsnes, T. Korsvold, and M. Choate, "The overlooked drilling hazard: Decision making from bad data," in *SPE Intelligent Energy International*, no. SPE-150306. Utrecht, The Netherlands: Society of Petroleum Engineers, 2012. 44, 75

[74] R. Long and D. Veeningen, "Networked drill pipe offers along-string pressure evaluation in real time," *World Oil*, pp. 91–94, September 2011. 44

[75] D. S. Pixton and A. Craig, "Drillstring network 2.0: An enhanced drillstring network based on 100 wells of experience," in *IADC/SPE Drilling Conference and Exhibition*, no. SPE-167965-MS. Fort Worth, TX: Society of Petroleum Engineers, 2014. 44

[76] R. A. Shishavan, C. Hubbell, H. Perez, J. D. Hedengren, and D. S. Pixton, "Combined rate of penetration and pressure regulation for drilling optimization using high speed telemetry," *SPE Drilling & Completion Journal*, no. SPE-170275-MS, 2015. 44, 74, 75, 77, 79

[77] R. A. Shishavan, C. Hubbell, H. Perez, J. D. Hedengren, D. S. Pixton, and A. P. Pink, "Multivariate control for managed pressure drilling systems using high speed telemetry," in *SPE Annual Technical Conference and Exhibition*, no. SPE-170962-MS. Amsterdam, The Netherlands: Society of Petroleum Engineers, 2014. 44

[78] D. S. Pixton, R. A. Shishavan, H. D. Perez, J. D. Hedengren, and A. Craig, "Addressing ubo and mpd challenges with wired drill pipe telemetry," in *SPE/IADC Managed Pressure Drilling & Underbalanced Operations Conference & Exhibition*, no. SPE-168953-MS. Society of Petroleum Engineers, 2014. 44, 75

[79] B. Spivey, J. Hedengren, and T. Edgar, "Constrained nonlinear estimation for industrial process fouling," *Industrial & Engineering Chemistry Research*, vol. 49, no. 17, pp. 7824–7831, 2010. 46

[80] J. Hedengren and D. Brower, "Advanced process monitoring of flow assurance with fiber optics," in *AIChE Spring Meeting*, Houston, TX, 2012. 46

[81] K. Jensen and J. Hedengren, "Improved load following of a boiler with advanced process control," in *AIChE Spring Meeting*, Houston, TX, 2012. 46

[82] D. Brower, J. Hedengren, C. Loegering, A. Brower, K. Witherow, and K. Winter, "Fiber optic monitoring of subsea equipment," in *ASME 31st International Conference on Ocean, Offshore & Arctic Engineering (OMAE)*, no. 84143, Rio de Janiero, Brazil, July 2012. 46

[83] A. Eaton, S. Safdarnejad, J. Hedengren, K. Moffat, C. Hubbell, D. Brower, and A. Brower, "Post-installed fiber optic pressure sensors on subsea production risers for severe slugging control," in *ASME 34th International Conference on Ocean, Offshore, and Arctic Engineering (OMAE)*, no. 42196, St. John's, Newfoundland, Canada, June 2015. 46

[84] B. Hallac, K. Kayvanloo, J. Hedengren, W. Hecker, and M. Argyle, "An Optimized Simulation Model for Iron-Based Fischer-Tropsch Catalyst Design: Transfer Limitations as Functions of Operating and Design Conditions," *Chemical Engineering Journal*, vol. 263, pp. 268–279, 2015. 46

[85] S. M. Safdarnejad, J. D. Hedengren, and L. L. Baxter, "Plant-level dynamic optimization of cryogenic carbon capture with conventional and renewable power sources," *Applied Energy*, vol. 149, no. 0, pp. 354 – 366, 2015. 46

[86] K. M. Powell, J. D. Hedengren, and T. F. Edgar, "Dynamic optimization of a hybrid solar thermal and fossil fuel system," *Solar Energy*, vol. 108, pp. 210 – 218, 2014. 46

[87] J. Kelly and J. Hedengren, "A steady-state detection (SSD) algorithm to detect non-stationary drifts in processes," *Journal of Process Control*, vol. 23, no. 3, pp. 326–331, March 2013. 46

[88] M. Darby, M. Nikolaou, J. Jones, and D. Nicholson, "RTO: An overview and assessment of current practice," *Journal of Process Control*, vol. 21, pp. 874–884, 2011. 46, 48

[89] L. Sun, J. D. Hedengren, and R. W. Beard, "Optimal trajectory generation using model predictive control for aerially towed cable systems," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 2, pp. 525–539, 2014. 46

[90] L. Jacobsen, B. Spivey, and J. Hedengren, "Model predictive control with a rigorous model of a solid oxide fuel cell," in *Proceedings of the American Control Conference (ACC)*, Washington, D.C., June 2013, pp. 3747–3752. 46

[91] J. Rawlings, D. Angeli, and C. Bates, "Fundamentals of economic model predictive control," in *2012 IEEE 51st Annual Conference on Decision and Control (CDC)*, Dec 2012, pp. 3851–3861. 46

[92] M. Ellis, H. Durand, and P. D. Christofides, "A tutorial review of economic model predictive control methods," *Journal of Process Control*, vol. 24, no. 8, pp. 1156 – 1178, 2014, economic nonlinear model predictive control. 46

[93] D. Sui, R. Nybø, G. Gola, D. Roverso, and M. Hoffmann, "Ensemble methods for process monitoring in oil and gas industry operations," *Journal of Natural Gas Science and Engineering*, vol. 3, no. 6, pp. 748 – 753, 2011, artificial Intelligence and Data Mining. 46, 83

[94] J. Kelly and D. Zyngier, "Continuously improve the performance of planning and scheduling models with parameter feedback." in *FOCAPO 08 - Foundations of Computer Aided Process Operations*, Boston, MA, 2008. 47

[95] M. Liebman, T. Edgar, and L. Lasdon, "Efficient data reconciliation and estimation for dynamic processes using nonlinear programming techniques," *Computers and Chemical Engineering*, vol. 16, pp. 963–986, 1992. 47, 49

[96] P. Moraal and J. Grizzle, "Observer design for nonlinear systems with discrete-time measurements," *IEEE Transactions on Automatic Control*, vol. 40, no. 3, pp. 395–404, 1995. 47

[97] Z. Abul-el-zeet and P. Roberts, "Enhancing model predictive control using dynamic data reconciliation," *AIChE Journal*, vol. 48, no. 2, pp. 324–333, 2002. 47

[98] J. Taylor and R. del Pilar Moreno, "Nonlinear dynamic data reconciliation: In-depth case study," in *2013 IEEE International Conference on Control Applications (CCA)*, Aug 2013, pp. 746–753. 47

[99] D. M. Prata, E. L. Lima, and J. C. Pinto, "Nonlinear dynamic data reconciliation in real time in actual processes," in *10th International Symposium on Process Systems Engineering: Part A*, ser. Computer Aided Chemical Engineering, R. M. de Brito Alves, C. A. O. do Nascimento, and E. C. Biscaia, Eds. Elsevier, 2009, vol. 27, pp. 47 – 54. 47

[100] T. Soderstrom, T. Edgar, L. Russo, and R. Young, "Industrial application of a large-scale dynamic data reconciliation strategy," *Industrial and Engineering Chemistry Research*, vol. 39, pp. 1683–1693, 2000. 48

[101] C. Rao, J. Rawlings, and J. Lee, "Constrained linear state estimation - a moving horizon approach," *Automatica*, vol. 37, pp. 1619–1628, 2001. 48, 64

[102] J. Renfro, A. Morshedi, and O. Asbjornsen, "Simultaneous optimization and solution of systems described by differential/algebraic equations," *Computers and Chemical Engineering*, vol. 11, no. 5, pp. 503–517, 1987. 48

[103] S. M. Safdarnejad, J. D. Hedengren, N. R. Lewis, and E. Haseltine, "Initialization strategies for optimization of dynamic systems," *Computers & Chemical Engineering*, no. 0, pp. –, 2015. 48

[104] G. Carey and B. Finlayson, "Othogonal collocation on finite elements," *Chemical Engineering Science*, vol. 30, pp. 587–596, 1975. 49

[105] R. Findeisen, F. Allgöwer, and L. Biegler, *Assessment and future directions of nonlinear model predictive control*. Berlin: Springer-Verlag, 2007. 49

[106] J. Hedengren and T. Edgar, "Order reduction of large scale DAE models," in *IFAC 16th World Congress*, Prague, Czechoslovakia, 2005. 49

[107] J. Albuquerque and L. Biegler, "Decomposition algorithms for on-line estimation with nonlinear models," *Computers and Chemical Engineering*, vol. 19, no. 10, pp. 1031–1039, 1995. 49

[108] S. Jang, B. Joseph, and H. Mukai, "Comparison of two approaches to on-line parameter and state estimation of nonlinear systems," *Ind. Eng. Chem. Process Des. Dev.*, vol. 25, pp. 809–814, 1986. 49

[109] Y. Ramamurthi, P. Sistu, and B. Bequette, "Control-relevant dynamic data reconciliation and parameter estimation," *Computers and Chemical Engineering*, vol. 17, no. 1, pp. 41–59, 1993. 49

[110] J. Hedengren and T. Edgar, "Moving horizon estimation - the explicit solution," in *Proceedings of Chemical Process Control (CPC) VII Conference*, Lake Louise, Alberta, Canada, 2006. 49

[111] S. Qin and T. Badgwell, *Nonlinear Model Predictive Control*. Boston, MA: Birkhäuser Verlag, 2000, ch. An overview of nonlinear model predictive control applications, pp. 369–392. 50

[112] V. Zavala and L. Biegler, "Nonlinear programming strategies for state estimation and model predictive control," in *Nonlinear Model Predictive Control*, ser. Lecture Notes in Control and Information Sciences, L. Magni, D. Raimondo, and F. Allgöwer, Eds. Springer Berlin Heidelberg, 2009, vol. 384, pp. 419–432. 50

[113] M. Soroush, "State and parameter estimations and their applications in process control," *Computers and Chemical Engineering*, vol. 23, pp. 229–245, 1998. 50

[114] E. Haseltine and J. Rawlings, "Critical evaluation of extended kalman filtering and moving-horizon estimation," *Ind. Eng. Chem. Res.*, vol. 44, no. 8, pp. 2451–2460, 2005. 54

[115] P. Vachhani, R. Rengaswamy, V. Gangwal, and S. Narasimhan, "Recursive estimation in constrained nonlinear dynamical systems," *AIChE Journal*, vol. 51, no. 3, pp. 946–959, 2005. 54

[116] R. Lambert, I. Nascu, and E. Pistikopoulos, "Simultaneous reduced order multi-parametric moving horizon estimation and model predictive control," *Dynamics and Control of Process Systems*, vol. 10, no. 1, pp. 267–278, 2013. 54

[117] J. Ramlal, V. Naidoo, K. Allsford, and J. Hedengren, "Moving horizon estimation for an industrial gas phase polymerization reactor," in *Proc. IFAC Symposium on Nonlinear Control Systems Design (NOLCOS)*, Pretoria, South Africa, 2007. 55

[118] L. Biegler, X. Yang, and G. Fischer, "Advances in sensitivity-based nonlinear model predictive control and dynamic real-time optimization," *Journal of Process Control*, no. 0, pp. –, 2015. 55

[119] J. Rawlings and D. Mayne, *Model predictive control: theory and design*. Madison, WI: Nob Hill Publishing, LLC, 2009. 55, 56

[120] K. R. Muske and T. A. Badgwell, "Disturbance modeling for offset-free linear model predictive control," *Journal of Process Control*, vol. 12, pp. 617–632, 2002. 55

[121] G. Pannocchia and J. Rawlings, "Disturbance models for offset-free MPC control," *AIChE Journal*, vol. 49, no. 2, pp. 426–437, 2002. 55

[122] G. Pannocchia and E. Kerrigan, "Offset-free control of constrained linear discrete-time systems subject to persistent unmeasured disturbances," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, Maui, Hawaii, December 2003, pp. 3911–3916. 55

[123] B. Odelson, M. Rajamani, and J. Rawlings, "A new autocovariance least-squares method for estimating noise covariances," *Automatica*, vol. 42, no. 2, pp. 303–308, February 2006. 55

[124] T. Binder, L. Blank, H. Bock, R. Burlisch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J. Schlöder, and O. Stryk, *Online Optimization of Large Scale Systems*. Springer-Verlag Berlin Heidelberg, 2001, ch. Introduction to model based optimization of chemical processes on moving horizons, pp. 295–339. 56

[125] S. M. Safdarnejad, J. R. Gallacher, and J. D. Hedengren, "Dynamic parameter estimation and optimization for batch distillation," *Computers & Chemical Engineering*, vol. 86, pp. 18 – 32, 2016. 67, 69, 103

[126] J. Beck and K. J. Arnold, *Parameter Estimation in Engineering and Science*. John Wiley & Sons, 1977. 68

[127] G. A. Seber and C. J. Wild, *Nonlinear Regression*. Hoboken, New Jersey: John Wiley & Sons, 2003. 68

[128] Ø. Stamnes, J. Zhou, G.-O. Kaasa, and O. M. Aamo, "Adaptive observer design for the bottomhole pressure of a managed pressure drilling system," in *IEEE Conference on Decision and Control*, 2008. 68, 74, 79, 100

[129] D. Veeningen, "Along-string pressure evaluation enabled by broadband networked drillstring provide safety, efficiency gains," in *Offshore Technology Conference*, 2011. 75, 96, 97

[130] Y. Zhidan, W. Chunming, G. Yanfeng, S. Jing, H. Xiufeng, and L. Yuan, "Design of a rotary valve orifice for a continuous wave mud pulse generator," *Precision Engineering*, vol. 41, pp. 111–118, 2015. 75, 97

[131] A. Nikoofard, T. A. Johansen, and G. O. Kaasa, "Design and comparison of adaptive estimators for under-balanced drilling," in *American Control Conference (ACC), 2014*, pp. 5681–5687. 75

[132] I. S. Landet, A. Pavlov, and O. M. Aamo, "Modeling and control of heave-induced pressure fluctuations in managed pressure drilling," *Control Systems Technology, IEEE Transactions on*, vol. 21, no. 4, pp. 1340–1351, 2013. 75

[133] A. Nandan, S. Imtiaz, and S. Butt, "Robust control of managed pressure drilling," in *Oceans - St. John's, 2014*, pp. 1–8. 75

[134] Z. Jing, O. N. Stamnes, O. M. Aamo, and G. O. Kaasa, "Switched control for pressure regulation and kick attenuation in a managed pressure drilling system," *Control Systems Technology, IEEE Transactions on*, vol. 19, no. 2, pp. 337–350, 2011. 75

[135] O. Breyholtz, G. Nygaard, J. M. Godhavn, and E. H. Vefring, "Evaluating control designs for co-ordinating pump rates and choke valve during managed pressure drilling operations," in *Control Applications, (CCA) & Intelligent Control, (ISIC), 2009 IEEE*, pp. 731–738. 75

[136] B. Aadnoy, I. Cooper, S. Miska, R. F. Mitchell, and M. L. Payne, *Advanced Drilling and Well Technology*. Society of Petroleum Engineers, 2009. 75

[137] J. Petersen, R. Rommetveit, K. S. Bjørkevoll, and J. Frøyen, "A general dynamic model for single and multi-phase flow operations during drilling, completion, well control and intervention," in *IADC/SPE Asia Pacific Drilling Technology Conference and Exhibition*, 2008. 75, 78, 96, 98, 105

[138] C. Pantelides, M. Nauta, and M. Matzopoulos, "Equation-oriented process modelling technology: Recent advances and current perspectives," in *5th Annual TRC-Idemitsu Workshop*, 2015. 91

[139] W. Marquardt, "Nonlinear model reduction for optimization based control of transient chemical processes," in *Chemical Process Control VI*, J. B. Rawlings, B. A. Ogunnaike, and J. W. Eaton, Eds., Tuscon, Arizona, 2001, pp. 12–42. 91

[140] G. A. Gonçalves, A. R. Secchi, and E. C. B. Jr., "Fast nonlinear predictive control and state estimation of distillation columns using first-principles reduced-order model," in *24th European Symposium on Computer Aided Process Engineering*, ser. Computer Aided Chemical Engineering. Elsevier, 2014, vol. 33, pp. 715–720. 91

[141] J. Yan, E. Harinath, and G. Dumont, "Closed-loop identification for model predictive control: Direct method," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, Dec 2009, pp. 2592–2597. 91

[142] M. Lazar, "Model predictive control of hybrid systems: Stability and robustness," Ph.D. dissertation, Eindhoven University of Technology, 2006. 91, 93, 95

[143] L. O. Santos, P. A. Afonso, J. A. Castro, N. M. Oliveira, and L. T. Biegler, "On-line implementation of nonlinear MPC: an experimental case study," *Control Engineering Practice*, vol. 9, no. 8, pp. 847 – 857, 2001. 92

[144] R. Findeisen and F. Allgöwer, "Computational delay in nonlinear model predictive control," in *Proc. Int. Adv. Control of Chemical Processes, ADCHEM'03*, Hong Kong, 2004. 92

[145] V. M. Zavala and L. T. Biegler, "The advanced-step NMPC controller: Optimality, stability and robustness," *Automatica*, vol. 45, no. 1, pp. 86–93, 2009. 92

[146] X. Yang and L. T. Biegler, "Advanced-multi-step nonlinear model predictive control," *Journal of Process Control*, vol. 23, no. 8, pp. 1116–1128, 2013. 92

[147] H. Pirnay, R. López-Negrete, and L. T. Biegler, "Optimal sensitivity based on IPOPT," *Mathematical Programming Computation*, vol. 4, no. 4, pp. 307–331, 2012. 92

[148] R. López-Negrete, F. J. D'Amato, L. T. Biegler, and A. Kumar, "Fast nonlinear model predictive control: Formulation and industrial process applications," *Computers & Chemical Engineering*, vol. 51, pp. 55–64, 2013. 92

[149] M. Yu and L. Biegler, "Nonlinear model predictive control of a bubbling fluidized bed adsorber for post-combustion carbon capture," in *AIChE Annual Meeting*, Salt Lake City, Utah, 2015. 92

[150] R. Huang, "Nonlinear model predictive control and dynamic real time optimization for large-scale processes," Ph.D. dissertation, Carnegie Mellon University, 12 2010. [Online]. Available: http://repository.cmu.edu/dissertation/29 92

[151] L. Biegler, "Technology advances for dynamic real-time optimization," in *10th International Symposium on Process Systems Engineering: Part A*, ser. Computer Aided Chemical Engineering. Elsevier, 2009, vol. 27, pp. 1–6. 92

[152] K. Narendra and J. Balakrishnan, "Adaptive control using multiple models," *Automatic Control, IEEE Transactions on*, vol. 42, no. 2, pp. 171–187, Feb 1997. 93

[153] L. Giovanini, G. Sanchez, and M. Benosman, "Observer-based adaptive control using multiple-models switching and tuning," *Control Theory Applications, IET*, vol. 8, no. 4, pp. 235–247, March 2014. 93

[154] X. Koutsoukos, P. J. Antsaklis, J. Stiver, and M. Lemmon, "Supervisory control of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1026–1049, 2000. 93

[155] M. Kuure-Kinsey and B. Bequette, "Multiple model predictive control of nonlinear systems," in *Nonlinear Model Predictive Control*, ser. Lecture Notes in Control and Information Sciences, L. Magni, D. Raimondo, and F. Allgöwer, Eds. Springer Berlin Heidelberg, 2009, vol. 384, pp. 153–165. 93

[156] M. Lazar, W. Heemels, S. Weiland, and A. Bemporad, "Stabilizing model predictive control of hybrid systems," *Automatic Control, IEEE Transactions on*, vol. 51, no. 11, pp. 1813–1818, Nov 2006. 93

[157] H. Ye, A. N. Michel, and L. Hou, "Stability theory for hybrid dynamical systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 461–474, 1998. 94

[158] D. Liberzon and A. S. Morse, "Basic problems in stability and design of switched systems," *IEEE Control Systems Magazine*, pp. 59–70, October 1999. 94

[159] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014. 94, 95

[160] J. Pannek and K. Worthmann, "Stability and performance guarantees for model predictive control algorithms without terminal constraints," *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift fr Angewandte Mathematik und Mechanik*, vol. 94, no. 4, pp. 317–330, 2014. 94

[161] L. Magni, R. Scattolini, and M. Tanelli, "Switched model predictive control for performance enhancement," *International Journal of Control*, vol. 81, no. 12, pp. 1859–1869, 2008. 95

[162] A. Eaton, L. Beal, S. Thorpe, E. Janis, C. Hubbell, J. Hedengren, R. Nybø, M. Aghito, K. Bjørkevoll, R. E. Boubsi, J. Braaksma, and G. van Og, "Addressing discontinuous process challenges with multi-fidelity model predictive control," in *AIChE Annual Meeting*, Salt Lake City, Utah, 2015. 95

[163] R. Shorten, F. Wirth, O. Mason, K. Wulff, and C. King, "Stability criteria for switched and hybrid systems," *SIAM Rev.*, vol. 49, no. 4, pp. 545–592, Nov. 2007. 95

[164] R. Asgharzadeh Shishavan, C. Hubbell, H. Perez, J. D. Hedengren, D. Pixton, and A. Craig, "Addressing UBO and MPD challenges with wired drill pipe telemetry and model predictive control," in *SPE/IADC Managed Pressure Drilling and Underbalanced Operations Conference*, 2014. 96, 100

[165] OpenfieldTechnology, "Micro pressure and temperature digital sensor for OEM integration," Brochure, 2015. [Online]. Available: http://openfield-technology.com/wp-content/uploads/2012/04/OpenFieldBrochure_OEM_recorderLight.pdf 97

[166] J. D. Hedengren and A. N. Eaton, "Overview of estimation methods for industrial dynamic systems," *Optimization and Engineering*, pp. 1 – 24, 2015. 100

[167] R. A. Shishavan, "Nonlinear estimation and control with application to upstream processes," Thesis, Brigham Young University, Provo, Utah, USA, 2014. 121

[168] H. Genceli and M. Nikolaou, "Robust stability analysis of constrained l1-norm model predictive control," *AIChE Journal*, vol. 39, no. 12, pp. 1954–1965, 1993. 121

APPENDIX A. RISER SEVERE SLUGGING CONTROLLER

This appendix contains the novel riser slugging controller in the MATLAB interface for APMonitor. The controller code is in A.1, and the initialization code is in A.2.

Listing A.1: Slugging Controller in the APMonitor Modeling Language Interfaced with MATLAB

```
function output = controller(input)
persistent controller_initialize
persistent icount s a
  if (isempty(icount)),
      icount = 0;
  end
icount = icount + 1;
p_meas = input(1);
p_sp = input(2);
p_sphi = p_sp + 0.01;
p_splo = p_sp - 0.01;
% Only execute first cycle
  if (isempty(controller_initialize)),
      addpath('apm');
      % Define Server
      s = 'http://localhost';
      % Define application name
      a = 'slug_mpc';
      % Initialize application
      controller_init(s,a);
  end
apm_meas(s,a,'p',p_meas);
apm_option(s,a,'p.sp',p_sp);
apm_option(s,a,'p.sphi',p_sphi);
apm_option(s,a,'p.splo',p_splo);
```

```
% solve and display output
solver_output = apm(s,a,'solve');
disp(solver_output)
% show status and cpu time
status = apm_tag(s,a,'nlc.appstatus');
cpu_time = apm_tag(s,a,'nlc.solvetime');
disp(['Application Status:' int2str(status) 'CPU Time:' num2str(cpu_time)]);
% retrieve new valve position
z = apm_tag(s,a,'z.NEWVAL');
output(1) = z;
  if (isempty(controller_initialize)),
      % Open a web viewer
      apm_web(s,a);
      % Turn on feedback status after first cycle
      apm_option(s,a,'p.fstatus',1);
      % Request Control Mode (1-3)
      apm_option(s,a,'nlc.reqctrlmode',3);
      controller_initialize = true;
  end
end
```

Listing A.2: Slugging Controller Initialization in the APMonitor Modeling Language Interfaced with MATLAB

```
function [] = controller_init(s,a)
% Clear previous application
apm(s,a,'clear all');
%Load model
apm_load(s,a,'control.apm');
% load data
csv_load(s,a,'control.csv');
%Define Parameters
apm_info(s,a,'FV','K');
apm_info(s,a,'FV','tau');
apm_info(s,a,'MV','z');
apm_info(s,a,'CV','p');
% set additional options
```

```
apm_option(s,a,'nlc.imode',6);
apm_option(s,a,'nlc.reqctrlmode',1);
apm_option(s,a,'nlc.max_iter',100);
apm_option(s,a,'nlc.cv_type',1);
apm_option(s,a,'nlc.mv_type',1);
apm_option(s,a,'nlc.solver',1);
% some additional options
apm_option(s,a,'nlc.mv_step_hor',1);
apm_option(s,a,'nlc.hist_hor',1000);
apm_option(s,a,'nlc.web_plot_freq',5);
%setup CV (p choke valve pressure)
apm_option(s,a,'p.tau',30);
apm_option(s,a,'p.status',1);
apm_option(s,a,'p.tr_init',1);
apm_option(s,a,'p.fstatus',0);
apm_option(s,a,'p.wsphi',10);
apm_option(s,a,'p.wsplo',10);
%setup MV (z valve position)
apm_option(s,a,'z.status',1);
apm_option(s,a,'z.fstatus',0);
apm_option(s,a,'z.dmax',0.2);    % rate of change limits
apm_option(s,a,'z.dcost',10);    % adding cost for change
apm_option(s,a,'z.lower',0.02);  % lower limit
apm_option(s,a,'z.upper',0.33);  % upper limit
% time units (1=sec, 2=min, 3=hr, 4=day, 5=yr)
apm_option(s,a,'nlc.ctrl_units',1);
% 4 points per time interval : 1 second
apm_option(s,a,'nlc.nodes',3);
apm_option(s,a,'nlc.coldstart',1);
% read csv file
apm_option(s,a,'nlc.csv_read',1);
end
```

APPENDIX B. CONTROL MODEL PARAMETER ESTIMATION HEURISTICS

Appendix B contains the code that develops the model parameter estimation heuristic for single input single output systems. This code is written in MATLAB interfaced with APMonitor.

Listing B.1: Control Model Parameter Tolerance in the APMonitor Modeling Language Interfaced with MATLAB

```
clear all; close all; clc
addpath('apm')
s = 'http://127.0.0.1';
a = 'mismatch';
[K_mesh,tau_mesh] = meshgrid(0.2:0.1:5,0.2:0.1:5);
icycle = 0;
total = size(K_mesh,1) * size(K_mesh,2);
for j = 1:size(K_mesh,1),
    for k = 1:size(K_mesh,2),
        icycle = icycle + 1;
        disp(['Cycle ' int2str(icycle) ' of ' int2str(total)])
        apm(s,a,'clear all');
        apm_load(s,a,'model.apm');
        csv_load(s,a,'model.csv');
        apm_option(s,a,'nlc.imode',6);
        apm_info(s,a,'FV','tau2');
        apm_info(s,a,'FV','K2');
        apm_info(s,a,'MV','u');
        apm_info(s,a,'CV','y');
        apm_option(s,a,'u.status',1);
        apm_option(s,a,'u.dcost',01);
        apm_option(s,a,'u.fstatus',0);
        apm_option(s,a,'y.status',1);
        apm_option(s,a,'y.tau',1);
```

```
        apm_option(s,a,'y.sphi',5.0);
        apm_option(s,a,'y.splo',5.0);
        apm_option(s,a,'y.wsphi',100);
        apm_option(s,a,'y.wsplo',100);
        apm_option(s,a,'y.tr_init',0);
        apm_option(s,a,'y.fstatus',1);
        apm_option(s,a,'nlc.web_plot_freq',1);
        apm_meas(s,a,'tau2',tau_mesh(j,k));
        apm_meas(s,a,'K2',K_mesh(j,k));
        for i = 1:20,
            apm(s,a,'solve');
            sol = apm_sol(s,a); z = sol.x;
            x(i) = apm_tag(s,a,'x.model');
            y(i) = apm_tag(s,a,'y.model');
            apm_meas(s,a,'y',z.x(2));
        end
        obj(j,k) = sum(abs(y(2:end)-5));
    end
end
figure(1)
c = [0.001 0.01 0.1 1 2 3 4 5 6 8 10 15 20 30 50 70 100];
[C,h] = contour(K_mesh,tau_mesh,obj,c);
clabel(C,h,'Labelspacing',250);
xlabel('Gain (K) Multiplier');
ylabel('Time Constant (\tau) Multiplier');
legend('MPC Objective')
figure(2)
surfc(K_mesh,tau_mesh,obj);
xlabel('Gain (K) Multiplier');
ylabel('Time Constant (\tau) Multiplier');
legend('MPC Objective')
```

APPENDIX C. BASIC SWITCHED CONTROLLER

The basic switched controller is implemented in Simulink®. The Simulink file ties all of the other MATLAB scripts together. Appendix C has the basic switch controller code in MATLAB and APMonitor. C.1 contains the empirical controller code which calls on the empirical controller initialization file which is found in C.2. The APMonitor/MATLAB code for the low order controller and initialization are in C.3. The code for the high fidelity controller is in C.4. The high fidelity controller does not use APMonitor and is in pure MATLAB, but does require the Optimization Toolbox with the *fmincon* function to run properly. C.5 has the code for the switch logic in MATLAB.

Listing C.1: Empirical Controller Code in the APMonitor Modeling Language Interfaced with MATLAB

```
function [output] = Empirical_controller(input)
persistent controller_initialize
persistent s a1 icount p_c_opt xxx
global Z_choke_past q_p_past
  if (isempty(icount)),
      icount = 0;
  end
icount = icount + 1;
  if icount ==2
       apm_option(s,a1,'nlc.reqctrlmode',3);
  end
p_a_1_meas   = input(1);
p_c_meas     = input(2);
q_p_meas     = input(3);
z_choke_meas = input(4);
sp           = input(5);
% Only execute first cycle
```

```
  if (isempty(controller_initialize)),
      addpath('apm');
      % Define Server
      s = 'http://127.0.0.1';
      % Define application name
      a1 = 'nmpc_emp';
      % Initialize application
      Empirical_controller_init(s,a1);
  end
apm_meas(s,a1,'y[1]',p_a_1_meas);
apm_meas(s,a1,'u[1]',q_p_meas);
apm_meas(s,a1,'u[2]',z_choke_meas);
apm_option(s,a1,'y[1].sp',sp);
apm_option(s,a1,'y[1].sphi',sp*1.00125);
apm_option(s,a1,'y[1].splo',sp*0.99875);
solver_output = apm(s,a1,'solve');
disp(solver_output)
y = apm_sol(s,a1);
m = y.x;
y2hi = m.y1tr_hi(1);
y2lo = m.y1tr_lo(1);
% check solution status
status = apm_tag(s,a1,'nlc.appstatus');
% get cpu time
cpu_time = apm_tag(s,a1,'nlc.solvetime');
q_pump = apm_tag(s,a1,'u[1].NEWVAL');
Z_choke = apm_tag(s,a1,'u[2].NEWVAL');
output(1) = q_pump;
output(2) = Z_choke;
output(3) = y2hi;
output(4) = y2lo;
output(5) = status;
Z_choke_past = Z_choke;
  if (isempty(controller_initialize)),
      % Open a web viewer
      apm_web(s,a1);
```

```
54        apm_option(s,a1,'y[1].fstatus',1);
55        controller_initialize = true;
56    end
57 end
```

Listing C.2: Empirical Controller Initialization Code in the APMonitor Modeling Language Interfaced with MATLAB

```
1  function [] = Empirical_controller_init(s,a1)
2  % Clear previous alication
3  apm(s,a1,'clear all');
4  %Load model
5  apm_load(s,a1,'empirical_model.apm');
6  csv_load(s,a1,'control.csv');
7  %Define MVs
8  apm_info(s,a1,'MV','u[1]');  %q_pump
9  apm_info(s,a1,'MV','u[2]');  %z_choke
10 %Define CV
11 apm_info(s,a1,'CV','y[1]');
12 %Dynamic NMPC Controller mode
13 apm_option(s,a1,'nlc.imode',6);
14 apm_option(s,a1,'nlc.reqctrlmode',3);
15 % set additional options
16 apm_option(s,a1,'nlc.cv_type',1);
17 apm_option(s,a1,'nlc.solver',1);
18 apm_option(s,a1,'nlc.ctrl_units',1);
19 apm_option(s,a1,'nlc.nodes',2);
20 apm_option(s,a1,'nlc.coldstart',0);
21 apm_option(s,a1,'nlc.max_iter',100);
22 % set up CV (p_bit)
23 apm_option(s,a1,'y[1].tau',30);
24 apm_option(s,a1,'y[1].status',1);
25 apm_option(s,a1,'y[1].tr_init',0);
26 apm_option(s,a1,'y[1].tr_open',0);
27 apm_option(s,a1,'y[1].fstatus',0);
28 %setup MV (q_pump)
29 apm_option(s,a1,'u[1].status',1);
```

```
apm_option(s,a1,'u[1].fstatus',0);
apm_option(s,a1,'u[1].lower',0.001);
apm_option(s,a1,'u[1].upper',0.076);
apm_option(s,a1,'u[2].status',1);
apm_option(s,a1,'u[2].fstatus',0);
apm_option(s,a1,'u[2].lower',0.15);
apm_option(s,a1,'u[2].upper',0.85);
end
```

Listing C.3: Low Order Controller Code in the APMonitor Modeling Language Interfaced with MATLAB

```
function output = low_order_controller(input)
persistent controller_initialize
persistent s a icount sp_new q_influx_max stop
global fail
  if (isempty(icount)),
      icount = 0;
  end
   if isempty(sp_new)
       sp_new=0;
   end
icount = icount + 1;
  if icount ==2
       apm_option(s,a,'nlc.reqctrlmode',3);
  end
f_a             = input(1);      % Friction Factor at the bit
ro_a            = input(2);      % Density of the mud in the annulus [kg/m^3]
p_bit_meas      = input(3);    % Bit pressure
p_c_meas        = input(4);    % Choke pressure
h_bit_meas      = input(5);    % Bit position
q_choke_meas    = input(6)*60;   % Flow rate through the choke
q_p_meas        = input(7)*60;   % Mud pump pressure
z_choke_meas    = input(8)*100;  % Choke valve position
sp_new          = input(9);      % New Set Point for p_bit
q_back_meas     = q_choke_meas - q_p_meas; % Back pressure pump
sp              = sp_new;
```

```
  if (isempty(controller_initialize)),
      addpath('apm');
      % Define Server
      s = 'http://127.0.0.1';
            % Define application name
      a = 'nmpc_low';
      % Initialize application
      low_order_controller_init(s,a);
  end
  if isempty(q_influx_max),
      q_influx_max=0;
  end
  if isempty(stop),
      stop=0;
  end
%pass in process parameters
apm_meas(s,a,'f_a',f_a);
apm_meas(s,a,'p_bit',p_bit_meas);
apm_meas(s,a,'p_c',p_c_meas);
apm_meas(s,a,'h_bit',h_bit_meas);
apm_meas(s,a,'q_choke',q_choke_meas);
apm_meas(s,a,'ro_a',ro_a);
apm_meas(s,a,'z_choke',z_choke_meas);
apm_meas(s,a,'q_p',q_p_meas);
apm_meas(s,a,'q_back',q_back_meas);
%Define new set point for p_bit
apm_option(s,a,'p_bit.sp',sp);
apm_option(s,a,'p_bit.sphi',sp+3);
apm_option(s,a,'p_bit.splo',sp-3);
% solve and display output
solver_output = apm(s,a,'solve');
disp(solver_output)
y = apm_sol(s,a);
m = y.x;
y2hi = m.p_bittr_hi(1);
y2lo = m.p_bittr_lo(1);
```

```
% check solution status
status = apm_tag(s,a,'nlc.appstatus');
  if int2str(status) < 1,
      fail = fail + 1;
  end
% get cpu time
cpu_time = apm_tag(s,a,'nlc.solvetime');
q_p = apm_tag(s,a,'q_p.NEWVAL');
Z_choke = apm_tag(s,a,'Z_choke.NEWVAL');
output(1) = q_p/60;
output(2) = Z_choke/100;
output(3) = y2hi;
output(4) = y2lo;
output(5) = status;
  if (isempty(controller_initialize)),
      % Open a web viewer
      apm_web(s,a);
      apm_option(s,a,'p_c.fstatus',1);
      apm_option(s,a,'p_bit.fstatus',0);
      apm_option(s,a,'q_back.fstatus',1);
      apm_option(s,a,'q_p.fstatus',1);
      controller_initialize = true;
  end
end
```

Listing C.4: High Fidelity Controller Code in MATLAB

```
function [rec_changes] = HiFi_Controller2(input)
persistent mod_init x0 model_Path dll_file headerfile mod_inst big_count
   TimeStep;
persistent RTOutP_controller TimeP_controller BitPosition DepthHole t
   FlowInChoke_controller DensityInChoke_controller z;
persistent RTOut_controller MudFlowIn showtimeonce TimeStart;
if isempty(mod_init)
    Mud_init  = 0.04;
    z_init    = 0.5;
```

```
8   MudFlowIn = [ ones(1,15)*.045 linspace(.045,0,30) zeros(1,15) linspace
    (0,.045,30) ones(1,15)*.045];
9   %% HiFi_init
10  tempmode = 0;   % 1 dynamic temperature model, 0 static temperature model
11  TimeStep = 3;   %s
12  TimeStep = TimeStep/(24*60*60);
13  TimeStart = 42160;
14  % Define Flow Model dll files
15  dllfile           = 'FlowModel64r.dll'; % Filename
16  headerfile        = 'FlowModel.h';      % header file with function
    declarations
17  dll_file       = dllfile;
18  config_file ='case2.wel';
19  surveyfile        ='case2.sur';
20  % Defined operation sequences -- initialize variables
21  RotarySpeed(1)            = input(1)      ;% rad/s
22  MudDensityIn(1)           = input(2)      ;% Kg/m3
23  InputMudTemperatureIn(1) = input(3)      ;% Kelvin
24  ROP(1)                    = input(4)      ;% m/hr
25  BitPosition(1)            = 3700          ;% m (MD)
26  DepthHole(1)              = 3700          ;% m (MD)
27  ActualChokePress(1)       = input(7)      ;% Pa
28  SP_p_bit                  = input(8)      ;% Pa
29  z_choke                   = z_init        ;
30  InputDesiredEMW(1)        = 1900          ;% Kg/m3
31  SetPointPosition(1)       = 3700          ;% m (MD)
32  % Define position of pressure sensors
33  Choke                         = 0;
34  CasingShoe                    = 3700;
35  observationPoints_controller =[Choke 1 CasingShoe];
36  atChoke                       = find(observationPoints_controller==Choke);
37  atCasingShoe                  = find(observationPoints_controller==
    CasingShoe);
38  belowChoke                    = find(observationPoints_controller==1);
39  % Initialize model and read configuration data
40  mod_inst = 'FlowModel_controller';
```

```
41    % Clean up in case of previous crashes
42      if (libisloaded(mod_inst))
43          calllib(mod_inst, 'enddll')
44          unloadlibrary(mod_inst)
45          clear ('RTOutP_controller', 'Timep_controller', 'statusP_controller'
    , 'sensorsDataP_controller')
46      end
47    % Create subfolder and delete previous files
48    [s,m,mi] = mkdir(mod_inst);
49    delete(fullfile(mod_inst,'*'))
50    % copy dll and configuration files to subfolder
51    copyfile(config_file,fullfile(cd,mod_inst,config_file),'f');
52      if exist('surveyfile')
53          copyfile(surveyfile,fullfile(cd,mod_inst,surveyfile),'f');
54      end
55    copyfile(dll_file,fullfile(cd,mod_inst,dll_file),'f');
56    copyfile(headerfile,fullfile(cd,mod_inst,headerfile),'f');
57    h = fopen(fullfile(cd,mod_inst,'InputFileName.in'),'w');
58    fwrite(h,config_file);
59    fclose(h);
60    % copy modified configuration file for well, used for simulating
    configuration errors
61      if exist('config_file_controller')
62  copyfile(config_file_controller,fullfile(cd,mod_inst,config_file_cntr),'f');
63          h = fopen(fullfile(cd,mod_inst,'InputFileName.in'),'w');
64          fwrite(h,config_file_controller);
65          fclose(h);
66      end
67    % copy modified survey file for well model, used for simulating
    configuration errors
68      if exist('surveyfile_controller') copyfile(surveyfile_controller,
    fullfile(cd,mod_inst,surveyfile_controller),'f');
69      end
70    cd(mod_inst); % path of InputFileName.in, which points to .wel input file
    with configuration data
71    model_Path = cd;
```

```
72  %load dll
73  [notfound,warnings] = loadlibrary(dll_file(1:end-4),headerfile,'alias',
    mod_inst);
74  % Initialize Flow Model
75  ReturnCode = calllib(mod_inst, 'setmodulestring', 'FilePath', 8,
    model_Path, length(model_Path), 0);
76  cd ..
77  %define data structures and pointers
78  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
79  % RealTime Output data structure
80  RTOut_controller.bhEcdCalc        = -999.99; % Calc Bottom hole ECD
81  RTOut_controller.bhTempCalc       = -999.99; % Calc Bottom hole Temp
82  RTOut_controller.csEcdCalc        = -999.99; % Calc Casing shoe ECD
83  RTOut_controller.csTempCalc       = -999.99; % Calc Casing shoe Temp
84  RTOut_controller.pitGainCalc      = -999.99; % Calc Pit Gain
85  RTOut_controller.sppCalc        = -999.99; % Calc Standpipe pressure
86  RTOut_controller.flowOutCalc      = -999.99; % Calc Flow out
87  RTOut_controller.tempOutCalc      = -999.99; % Calc Temperature Out
88  RTOut_controller.cmpLiftHeight      = -999.99; % Subsea pump lifting height
89  RTOut_controller.cmpPresSet       = -999.99; % CMP inlet pressure setpoint
90  RTOut_controller.cmpSucPCalc      = -999.99; % CMP suction pres
91  RTOut_controller.cmpRpmCalc       = -999.99; % CMP RPM
92  RTOut_controller.cmpPowCalc       = -999.99; % CMP Power
93  RTOut_controller.cmpSurgeVolCalc    = -999.99; % Surge tank blanket volume
94  RTOut_controller.pChoke         = -999.99;
95  RTOut_controller.frictionFactor     = -999.99;
96  RTOut_controller.bhPresCalc       = -999.99; % Calc bottom hole pressure
97  RTOut_controller.csPresCalc       = -999.99; % Calc casing shoe pressure
98  RTOut_controller.ropCalc        = -999.99; % Calc ROP
99  RTOut_controller.diffWellPoreEmwCalc = -999.99;
100 RTOut_controller.diffFracWellEmwCalc = -999.99;
101 RTOut_controller.surgeVolCalc     = -999.99; % Calculated Surge Volume.
102 RTOut_controller.tvdBottomCalc      = -999.99; % Calculated TVD at bottom.
103 RTOut_controller.pwdPresCalc      = -999.99; % Pressure sensor position
104 RTOut_controller.pwdTempCalc      = -999.99; % Temperature sensor position
105 RTOut_controller.volBlanketRiserCalc = -999.99;
```

```
106     RTOut_controller.levelMudRiserCalc    = -999.99;
107     RTOut_controller.volMudRiserCalc    = -999.99;
108     RTOut_controller.volCMPRetLineCalc    = -999.99;
109     RTOut_controller.ecdAtPosCalc     = -999.99; % ecd at userDefined pos
110     % Definition of data pointers
111     RTOutP_controller = libpointer('realtimeDataStructureOut',RTOut_controller
        );
112     TimeP_controller = libpointer('doublePtr',0);
113     sensorsDataP_controller = libpointer('doublePtr',0);
114     statusP_controller = libpointer('int32Ptr',0);
115     gasinfluxrateP_controller = libpointer('doublePtr',0);
116     % Configure well model to use the selected ActualChokePress as boundary
        condition
117     BoundaryPos(1) = 2;
118     ReturnCode = calllib(mod_inst, 'setmoduletable', 'OPTChange', 9,
        BoundaryPos, 1, 0, 0);
119     ReturnCode = calllib(mod_inst, 'setmoduletable', '
        Use_measured_choke_pressure_as_boundary', 39, 1, 1, 0, 0);
120     % Set maximum choke pressure change rate to 10 Bar/s
121     Input_Configur(1:5) = -999; % other values in Input_Configur are not
        changed
122     Input_Configur(6) = 10e5;   % Max choke pressure change rate (Pa/s)
123     ReturnCode = calllib(mod_inst, 'setmoduletable', 'Input_Configur', 14,
        Input_Configur, 0, 5, 0);
124     % Select Dynamic temperature model
125     Input_Configur(1:3) = -999; % other values in Input_Configur are not
        changed
126     Input_Configur(4) = tempmode; % Temperature Mode
127     ReturnCode = calllib(mod_inst, 'setmoduletable', 'Input_Configur', 14,
        Input_Configur, 0, 3, 0);
128     valvecount = 0;
129         for t = 1:4,
130           %% Valve Equations
131           maxchange = 10; %bar per iteration
132           %valve position limits
133             if z_choke > 1
```

```
134                 z_choke = 1;
135             elseif z_choke < 0
136                 z_choke = 0;
137             end
138             if t > 1
139               valvecount = valvecount + 1;
140               if valvecount == 1 % of iterations before choke pressure update
141                   ActualChokePress(t) = ((((FlowInChoke_controller(t-1)+.0125)
    *1000)/(3.5*z_choke))^2)*(DensityInChoke_controller(t-1)) + 101325;
142                   diff = ActualChokePress(t) - ActualChokePress(t-1);
143                   ActualChokePress(t) = ActualChokePress(t-1) + sign(diff)*min
    (maxchange*100000*TimeStep*(24*60*60),abs(diff));
144                   valvecount = 0;
145               else
146                   ActualChokePress(t) = ActualChokePress(t-1);
147               end
148           else
149               ActualChokePress(1)      = 52 * 100000;
150           end
151           if t > 1
152               BitPosition(t)=BitPosition(t-1)+(TimeStep*24)*ROP(1);
153               DepthHole(t)=max([BitPosition(1:t) DepthHole(1)]);
154           end
155         Time(t) = TimeStart + t * TimeStep
156         %% Run Well Model
157         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
158           if ~exist('t')
159               t = 1;
160           end
161         realTable_controller(2:43) = -999;
162         realTable_controller(2)     = Time(t);              %InputTime
163         realTable_controller(3)     = BitPosition(t);       %InputBitPosition
164         realTable_controller(6)     = RotarySpeed(1);       %InputRotarySpeed
165         realTable_controller(9)     = MudFlowIn(t);         %InputMudFlowIn
166         realTable_controller(10)    = MudDensityIn(1);      %InputMudDensityIn
167         realTable_controller(11)    = DepthHole(t);         %InputDepthHole
```

```
168         realTable_controller(12)   = 1900;                    %InputDesiredEMW
169         realTable_controller(18) = ActualChokePress(t);
170         realTable_controller(21) = InputMudTemperatureIn(1);
171         realTable_controller(31) = 3700;        %InputSetPointPosition
172         realTable_controller(44) = 0;           %InSlips
173        % Send updated real time table to FlowModel.dll
174         ReturnCode = calllib(mod_inst, 'setmoduletable', 'Input_RealTime', 14,
     realTable_controller, 43, 0, 0);
175        % Execute FlowModel
176         ReturnCode = calllib(mod_inst, 'runflowmodel', TimeP_controller, 2, 0,
     0);
177         get(TimeP_controller,'Value')
178         ReturnCode_getRealtime = calllib(mod_inst, 'getrealtimeparameters',
    RTOutP_controller,6);
179         data_controller = get(RTOutP_controller,'Value');
180         bhPres_controller          = data_controller.bhPresCalc;
181         getdoubleDataP_controller = libpointer('doublePtr',0);
182         calllib(mod_inst , 'getdouble', 'FlowInChoke', libpointer('uint32Ptr'
    ,11), getdoubleDataP_controller);
183         FlowInChoke_controller(t) = get(getdoubleDataP_controller,'Value');
184         calllib(mod_inst , 'getdouble', 'DensityInChoke', libpointer('
    uint32Ptr',14), getdoubleDataP_controller);
185         DensityInChoke_controller(t) = get(getdoubleDataP_controller,'Value');
186       end
187     ReturnCode = calllib(mod_inst, 'setmoduletable', 'Save_state_memory', 17,
    1, 0, 0, 0);
188     mod_init = 1;
189     big_count = 0;
190     z = t;
191     %% Initial guess
192     x0 = [.5];
193 end
194 RTOutP_controller = libpointer('realtimeDataStructureOut',RTOut_controller);
195 showtimeonce = 1;
196 OPTIONS = optimoptions('fmincon','Algorithm','interior-point','TolFun',.001);
197 [x,FVAL,exitflag] = fmincon(@shootModel,x0,[],[],[],[],[.01],[.90],[],OPTIONS)
```

```
  if exitflag > 0
      status = 1;
  else
      status = 0;
  end
rec_changes = [x(1) .04 status FlowInChoke_controller(z)
    DensityInChoke_controller(z)];
x0 = x;
z = z + 1;
    function err = shootModel(guess)
        ReturnCode = calllib(mod_inst, 'setmoduletable', 'Load_state_memory',
    17, 1, 0, 0, 0);
        RotarySpeed(1)              = input(1)          ;% rad/s
        MudDensityIn(1)             = input(2)          ;% Kg/m3
        InputMudTemperatureIn(1)    = input(3)          ;% Kelvin
        ROP(1)                      = input(4)          ;% m/hr
        DepthHole(1)                = input(6)          ;% m (MD)
        bias                        = input(8)          ; %simple bias to align
    model/measurement
        SP_p_bit                    = input(9)*100000   ;%Pa
        z_choke                     = guess(1)          ;
        big_count = big_count + 1;
        valvecount = 0;
        error = 0;
        for i = 1:1,
            t = z + i;
            %% Valve Equations
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            maxchange = 10; %bar per iteration
            %valve position limits 1-90%
            z_choke = min(z_choke,.9);
            z_choke = max(z_choke,.01);
            %%occasional pressure updates, no deriv, change P limit
            valvecount = valvecount + 1;
            if valvecount == 1 %# of iterations before choke pressure update
```

```
230                 ActualChokePress(t)       = ((((FlowInChoke_controller(t-1)
    +.0125)*1000)/(3.5*z_choke))^2)*(DensityInChoke_controller(t-1)) + 101325;
231                 diff = ActualChokePress(t) - ActualChokePress(t-1);
232                 ActualChokePress(t) = ActualChokePress(t-1) + sign(diff)*min(
    maxchange*100000*TimeStep*(24*60*60),abs(diff));
233                 valvecount = 0;
234             else
235                 ActualChokePress(t) = ActualChokePress(t-1);
236             end
237             if t > 1
238                 BitPosition(t)=BitPosition(t-1)+(TimeStep*24)*ROP(1);
239                 DepthHole(t)=max([BitPosition(1:t) DepthHole(1)]);
240             end
241             Time(t)=TimeStart+t*TimeStep;
242             %% Run Well Model
243             %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
244             % Update values for real time input table
245               if ~exist('t')
246                   t = 1;
247               end
248             realTable_controller(2:43) = -999;
249             realTable_controller(2)    = Time(t);           %InputTime
250             realTable_controller(3)    = BitPosition(t);    %InputBitPosition
251             realTable_controller(6)    = RotarySpeed(1);    %InputRotarySpeed
252             realTable_controller(9)    = MudFlowIn(t);      %InputMudFlowIn
253             realTable_controller(10)   = MudDensityIn(1);   %InputMudDensityIn
254             realTable_controller(11)   = DepthHole(t);      %InputDepthHole
255             realTable_controller(12)   = 1900;              %InputDesiredEMW
256             realTable_controller(18) = ActualChokePress(t);
257             realTable_controller(21) = InputMudTemperatureIn(1);
258             realTable_controller(31)   = 3700;     %InputSetPointPosition
259             realTable_controller(44)   = 0;        %InSlips
260             % Send updated real time table to FlowModel.dll
261             ReturnCode = calllib(mod_inst, 'setmoduletable', 'Input_RealTime',
     14, realTable_controller, 43, 0, 0);
262             % Execute FlowModel
```

```
263         ReturnCode = calllib(mod_inst, 'runflowmodel', TimeP_controller,
    2, 0, 0);
264         ReturnCode_getRealtime = calllib(mod_inst, 'getrealtimeparameters'
    ,RTOutP_controller,6);
265         data_controller = get(RTOutP_controller,'Value');
266         bhPres_controller           = data_controller.bhPresCalc;
      % Bottom Hole Pressure
267         getdoubleDataP_controller  = libpointer('doublePtr',0);
268         calllib(mod_inst , 'getdouble', 'FlowInChoke', libpointer('
    uint32Ptr',11), getdoubleDataP_controller);
269         FlowInChoke_controller(t) = get(getdoubleDataP_controller,'Value')
270         calllib(mod_inst , 'getdouble', 'DensityInChoke', libpointer('
    uint32Ptr',14), getdoubleDataP_controller);
271         DensityInChoke_controller(t) = get(getdoubleDataP_controller,'
    Value');
272        %% build objective function
273         p_off = abs(SP_p_bit - (bhPres_controller-bias))/100000;
274         if i < 2
275             if showtimeonce == 1
276                 disp(['Control move up to ' num2str(get(TimeP_controller,'
    Value')) ' recommended w/ t = ' num2str(t)]);
277                 disp(['Mudflowin = ' num2str(MudFlowIn(t)) ' Flowinchoke =
     ' num2str(FlowInChoke_controller(t-1)) ' Densinchoke = ' num2str(
    DensityInChoke_controller(t-1))]);
278                 showtimeonce = 0;
279             end
280             % deadband
281             if p_off > 2
282                 error = error + p_off*20*i^2;
283             else
284                 error = error + p_off*2*i;
285             end
286         else
287             if p_off > 2
288                 error = error + p_off*10/i^2;
289             else
```

```
                    error = error + p_off*1/i;
                end
            end
            error = error + abs(z_choke-x0(1))/20;
        end
        err = error;
    end
end
```

Listing C.5: Basic switch controller logic in MATLAB

```
function [ output ] = Switch_Test( input )
persistent time control count_down s a_low a_emp
addpath('apm');
s = 'http://byu.apmonitor.com';
a_emp = 'nmpc_emp';
a_low = 'nmpc_low';
Emp_ValvePos = input(1);
Emp_MudFlow = input(2);
Emp_status = input(3);
Low_ValvePos = input(4);
Low_Mudflow = input(5);
Low_status = input(6);
High_ValvePos = input(7);
High_MudFlow = input(8);
High_status = input(9);
if isempty(time);
    time = 0;
    control = 1;
    count_down = 0;
end
count_down = count_down - 1;
if count_down < 0;
    count_down = count_down + 1;
end
if High_status < 1;
    if Low_status < 1;
```

```
        control = 3;
    else
        control = 2;
    end
else
    control = 1;
end
%% outputs based on the previously defined control number
switch control
    case 1  %use hifi
        apm_option(s,a_emp,'nlc.reqctrlmode',2);
        apm_option(s,a_low,'nlc.reqctrlmode',2);
        apm_option(s,a_emp,'u[1].fstatus',1);
        apm_option(s,a_emp,'u[2].fstatus',1);
        apm_option(s,a_low,'z_choke.fstatus',1);
        apm_option(s,a_low,'q_p.fstatus',1);
        output(1) = High_ValvePos;
        output(2) = High_MudFlow;
    case 2  %use low
        apm_option(s,a_emp,'nlc.reqctrlmode',2);
        apm_option(s,a_emp,'u[1].fstatus',1);
        apm_option(s,a_emp,'u[2].fstatus',1);
        apm_option(s,a_low,'nlc.reqctrlmode',3);
        apm_option(s,a_low,'z_choke.fstatus',0);
        apm_option(s,a_low,'q_p.fstatus',0);
        output(1) = Low_ValvePos;
        output(2) = Low_Mudflow;
    case 3  %use emp
        apm_option(s,a_emp,'nlc.reqctrlmode',3);
        apm_option(s,a_emp,'u[1].fstatus',0);
        apm_option(s,a_emp,'u[2].fstatus',0);
        apm_option(s,a_low,'nlc.reqctrlmode',2);
        apm_option(s,a_low,'z_choke.fstatus',1);
        apm_option(s,a_low,'q_p.fstatus',1);
        output(1) = Emp_ValvePos;
        output(2) = Emp_MudFlow;
```

```
63 end
64 time = time + 1;
65 end
```

APPENDIX D. ADVANCED SWITCHED CONTROLLER

Appendix D contains the advanced switched controller code in MATLAB and APMonitor. D.1 contains the empirical controller code and initialization,D.2 contains the low order controller code and initialization file, and D.3 has the moving horizon estimation code. D.4 contains the empirical model identification code. Each of these scripts are written in the MATLAB interface for APMonitor. However, the high fidelity controller is written in MATLAB only, and is found in D.5.

Listing D.1: Empirical Controller Code in the APMonitor Modeling Language Interfaced with MATLAB

```
classdef Empirical_Class<handle
    %% Class definition for everything empirically related
    % Available functions:
    % controller_init()
    % simulation_init()
    % simulation(p_bit,q_p,p_c)
    % control(o,input)
    % control_solve(o,input)
    properties
        server = 'http://127.0.0.1';
        control_app;
        simulate_app;
        %deviation variables
        q_p_dev = 0;%0.04;
        p_c_dev = 0; %20;
        p_bit_dev =0;% 380;
        show_web = false;
    end
    methods
        function controller_init(o)
            Empirical_controller_init;
```

```
22            if o.show_web
23                apm_web(o.server,o.control_app);
24            end
25        end
26        function simulation_init(o)
27            emp_sim_init;
28            if o.show_web
29                apm_web(o.server,o.simulate_app);
30            end
31        end
32        function output = simulation(o,p_bit,q_p,p_c)
33            apm_meas(o.server,o.simulate_app,'y[1]',(p_bit-o.p_bit_dev));
34            apm_meas(o.server,o.simulate_app,'u[1]',(q_p-o.q_p_dev));
35            apm_meas(o.server,o.simulate_app,'u[2]',(p_c-o.p_c_dev));
36
37            emp_sim_solver_output = apm(o.server,o.simulate_app,'solve')
38            y = apm_sol(o.server,o.simulate_app);
39            m = y.x;
40            output = m.y1(end)+o.p_bit_dev;
41        end
42        function output = control(o,input)
43            p_bit = input(1);
44            p_c = input(2);
45            q_p = input(3);
46            sp = input(4);
47            apm_meas(o.server,o.control_app,'y[1]',(p_bit-o.p_bit_dev));
48            apm_meas(o.server,o.control_app,'u[1]',(q_p-o.q_p_dev));
49            apm_meas(o.server,o.control_app,'u[2]',(p_c-o.p_c_dev));
50            apm_option(o.server,o.control_app,'y[1].sp',sp-o.p_bit_dev);
51            apm_option(o.server,o.control_app,'y[1].sphi',sp+1-o.p_bit_dev);
52            apm_option(o.server,o.control_app,'y[1].splo',sp-1-o.p_bit_dev);
53            apm_option(o.server,o.control_app,'nlc.reqctrlmode',2);
54            emp_cont_solver_output = apm(o.server,o.control_app,'solve')
55            output = 1;
56        end
57        function output = control_solve(o,input)
```

```
            p_bit = input(1);
            p_c = input(2);
            q_p = input(3);
            sp = input(4);
            apm_meas(o.server,o.control_app,'y[1]',(p_bit-o.p_bit_dev));
            apm_meas(o.server,o.control_app,'u[1]',(q_p-o.q_p_dev));
            apm_meas(o.server,o.control_app,'u[2]',(p_c-o.p_c_dev));
            apm_option(o.server,o.control_app,'y[1].sp',sp-o.p_bit_dev);
            apm_option(o.server,o.control_app,'y[1].sphi',sp+1-o.p_bit_dev);
            apm_option(o.server,o.control_app,'y[1].splo',sp-1-o.p_bit_dev);
            apm_option(o.server,o.control_app,'nlc.reqctrlmode',3);
            emp_solver_output = apm(o.server,o.control_app,'solve')
            y = apm_sol(o.server,o.control_app);
            m = y.x;
            status = apm_tag(o.server,o.control_app,'nlc.appstatus');
            cpu_time = apm_tag(o.server,o.control_app,'nlc.solvetime');
            q_pump = apm_tag(o.server,o.control_app,'u[1].NEWVAL');
            choke_press = apm_tag(o.server,o.control_app,'u[2].NEWVAL');
            output(1) = q_pump +o.q_p_dev;
            output(2) = choke_press +o.p_c_dev;
            output(3) = status;
        end
    end
end
```

Listing D.2: Low Order Controller Code in the APMonitor Modeling Language Interfaced with MATLAB

```
function output = low_order_controller(well_meas,well_input,Low_MHE_data,
    varargin)
persistent controller_initialize s a
sp = well_input.sp_p_bit;
f_a           = Low_MHE_data.f_a;
ro_a          = Low_MHE_data.ro_a;
p_bit_meas    = well_meas.p_bit;
p_c_meas      = well_input.choke_press; % ensemble controller choice for
    movement
```

```
h_bit_meas    = well_meas.h_bit;
q_choke_meas = well_meas.q_choke * 60;
q_p_meas      = well_input.mudflowin * 60; % controller choice for movement
  if (isempty(controller_initialize)),
      % Initialize application
       low_order_controller_init;
       controller_initialize = true;
  end
apm_meas(s,a,'p_bit.sp',sp);
apm_option(s,a,'p_bit.sphi',sp+1);
apm_option(s,a,'p_bit.splo',sp-1);
apm_meas(s,a,'h_bit',h_bit_meas);
apm_meas(s,a,'q_choke',q_choke_meas);
apm_meas(s,a,'q_p',q_p_meas);
apm_meas(s,a,'p_c',p_c_meas);
apm_meas(s,a,'p_bit',p_bit_meas);
apm_meas(s,a,'f_a',f_a);
apm_meas(s,a,'ro_a',ro_a);
  if nargin > 3
      if varargin{1} == 'solve'
          apm_option(s,a,'nlc.reqctrlmode',3);
          Low_order_solver_output = apm(s,a,'solve')
          disp(Low_order_solver_output)
          y = apm_sol(s,a);
          m = y.x;
          % check solution status
          status = apm_tag(s,a,'nlc.appstatus');
          q_p = apm_tag(s,a,'q_p.NEWVAL');
          p_c = apm_tag(s,a,'p_c.NEWVAL');
          output(1) = q_p/60;
          output(2) = p_c;
          output(3) = status;
      end
  else
      apm_option(s,a,'nlc.reqctrlmode',2);
      Low_order_solver_output = apm(s,a,'solve')
```

```
      p_bit = apm_tag(s,a,'p_bit.PRED[1]');
      output(1) = p_bit;
  end
end
```

Listing D.3: Moving Horizon Estimation Code in the APMonitor Modeling Language Interfaced with MATLAB

```
function [ Low_MHE_data ] = Low_MHE(well_input, well_meas)
persistent controller_initialize s a
p_bit_meas      = well_meas.p_bit;
p_c_meas        = well_input.choke_press;
h_bit_meas      = well_meas.h_bit;
q_choke_meas    = well_meas.q_choke * 60;
q_p_meas        = well_input.mudflowin * 60;
  if (isempty(controller_initialize)),
      Low_MHE_init;                    % Initialize application
      controller_initialize = true;
  end
apm_meas(s,a,'p_bit',p_bit_meas);
apm_meas(s,a,'p_c',p_c_meas);
apm_meas(s,a,'h_bit',h_bit_meas);
apm_meas(s,a,'q_p',q_p_meas);
apm_meas(s,a,'q_choke',q_p_meas);
apm_option(s,a,'f_a.dmax',20);
apm_option(s,a,'ro_a.dmax',40);
apm_option(s,a,'p_bit.meas_gap',2);
solver_output = apm(s,a,'solve');
y = apm_sol(s,a);
display(solver_output);
status = apm_tag(s,a,'nlc.appstatus');
f_a = apm_tag(s,a,'f_a.NEWVAL');
ro_a = apm_tag(s,a,'ro_a.NEWVAL');
Low_MHE_data.f_a = f_a;
Low_MHE_data.ro_a = ro_a;
end
```

Listing D.4: Real Time Model Identification Code in MATLAB

```
classdef HiFi_Controller_Class<handle
    %% Class for creation of high fidelity controller
    %Available functions:
    %initialize(well_input,TimeStep,well_meas)
    %err = predictions(well_input,guess,x0,bias)
    %predictions = advance_instance(well_input,moves)
    %data = step_tests(well_input,empirical)
    properties
        TimeStep;
        model_instance = 'FlowModel_controller';
        RTOutP_controller;
        TimeP_controller;
        getdoubleDataP_controller;
        gasinfluxrateP_controller;
        BitPosition;
        DepthHole;
        RTOut_controller= struct();
        mpc_horizon = 4;
    end
    methods
      % initialize model instance for controller
        function o = initialize(o,well_input,TimeStep,well_meas)
            o.TimeStep = TimeStep;
            initialize_controller_model_script;
        end
       % run model instance to return objective function
        function err = predictions(o,well_input,guess,x0,bias)
            ReturnCode = calllib(o.model_instance, 'setmoduletable', '
   Load_state', 10, 1, 0, 0, 0);
            t                                = well_input.t ;
            Time(t)                          = well_input.time;
            RotarySpeed(t:t+10)              = well_input.rotary_speed;
            MudDensityIn(t:t+10)             = well_input.mud_density_in;
            InputMudTemperatureIn(t:t+10)    = well_input.input_mud_temp;
            ROP(t:t+10)                      = well_input.ROP;
```

```
35            o.BitPosition(t:t+10)              = well_input.bitposition;
36            o.DepthHole(t:t+10)                = well_input.depthhole;
37            InputDesiredEMW(t:t+10)         = well_input.desired_emw;
38            SetPointPosition(t:t+10)        = well_input.set_point_position;
39            bias                          = bias * 100000;
40            SP_p_bit                      = well_input.sp_p_bit * 100000;
41            ActualChokePress(t:t+10)        = guess(1)*100000;
42            MudFlowIn(t:t+10)               = guess(2);
43            z = t;
44            error = 0;
45            for i = 1:o.mpc_horizon, % horizon length
46                t = z + i;
47                if t>1
48                    o.BitPosition(t)=o.BitPosition(t-1)+(o.TimeStep*24)*ROP(t)
    ;
49                    o.DepthHole(t)=max([o.BitPosition(1:t) o.DepthHole(1)]);
50                end
51                % Run Well Model
52                setrealtime_controller;
53                % Send updated real time table to FlowModel.dll
54                ReturnCode = calllib(o.model_instance, 'setmoduletable', '
    Input_RealTime', 14, realTable_controller, 43, 0, 0);
55                ReturnCode = calllib(o.model_instance, 'runflowmodel', o.
    TimeP_controller, 2, 0, 0);
56                ReturnCode_getRealtime = calllib(o.model_instance, '
    getrealtimeparameters',o.RTOutP_controller,6);
57                data_controller = get(o.RTOutP_controller,'Value');
58                bhPres_controller           = data_controller.bhPresCalc;
59                error = error + i*abs(SP_p_bit-(bhPres_controller-bias))
    /10000+abs(ActualChokePress(t)-x0(1)*100000)/1000000+abs(ActualChokePress(
    t)-40*100000)/750000+abs(MudFlowIn(t)-x0(2))*20 ;
60            end
61            err = error;
62        end
63        function predictions = advance_instance(o,well_input,moves)
64            % Read input parameters
```

```
65          t                           = well_input.t ;
66          Time(t)                     = well_input.time    ;
67          RotarySpeed(t)              = well_input.rotary_speed;% rad/s
68          MudDensityIn(t)             = well_input.mud_density_in;% Kg/m3
69          InputMudTemperatureIn(t)    = well_input.input_mud_temp;% Kelvin
70          ROP(t)                      = well_input.ROP;% m/hr
71          o.BitPosition(t)              = well_input.bitposition;% m (MD)
72          o.DepthHole(t)                = well_input.depthhole;% m (MD)
73          InputDesiredEMW(t)          = well_input.desired_emw;
74          SetPointPosition(t)         = well_input.set_point_position;
75          MudFlowIn(t)                = moves(2);
76          ActualChokePress(t)         = moves(1)*100000;
77          % Run Well Model
78          %%%%%%%%%%%%%%%%%%%%%%%%%%%%
79          if t > 1
80              ReturnCode = calllib(o.model_instance, 'setmoduletable', '
    Load_state', 10, 1, 0, 0, 0);
81          end
82          setrealtime_controller
83          % Send updated real time table to FlowModel.dll
84          ReturnCode = calllib(o.model_instance, 'setmoduletable', '
    Input_RealTime', 14, realTable_controller, 43, 0, 0);
85          ReturnCode = calllib(o.model_instance, 'runflowmodel', o.
    TimeP_controller, 2, 0, 0);
86          ReturnCode_getRealtime = calllib(o.model_instance, '
    getrealtimeparameters',o.RTOutP_controller,6);
87          data_controller = get(o.RTOutP_controller,'Value');
88          predictions.p_pred = data_controller.bhPresCalc/100000;
89          predictions.flowout_cont = data_controller.flowOutCalc;
90          predictions.pitGain_cont = data_controller.pitGainCalc;
91          predictions.surgeVolCalc_cont = data_controller.surgeVolCalc;
92          showtimesave = get(o.TimeP_controller,'value')
93          showtimearray = Time - 42200;
94          ReturnCode = calllib(o.model_instance, 'setmoduletable', '
    Save_state', 10, 1, 0, 0, 0);
95      end
```

```
96        function data = step_tests(o,well_input,empirical)
97            ReturnCode = calllib(o.model_instance, 'setmoduletable', '
    Load_state', 10, 1, 0, 0, 0);
98            min = round(60/(o.TimeStep*24*60*60),0);
99            t                                       = well_input.t;
100           Time(t)                                 = well_input.time;
101           RotarySpeed(t:t+12*min)                 = well_input.rotary_speed;
102           MudDensityIn(t:t+12*min)                = well_input.mud_density_in;
103           InputMudTemperatureIn(t:t+12*min)       = well_input.input_mud_temp;
104           ROP(t:t+12*min)                         = well_input.ROP;
105           o.BitPosition(t:t+12*min)               = well_input.bitposition;
106           o.DepthHole(t:t+12*min)                 = well_input.depthhole;
107           InputDesiredEMW(t:t+12*min)             = well_input.desired_emw;
108           SetPointPosition(t:t+12*min) = well_input.set_point_position;
109           ActualChokePress(t:t+12*min) = well_input.choke_press *100000;
110           ActualChokePress(t+2*min:t+4*min) = (well_input.choke_press+5)
    *100000;
111           ActualChokePress(t+4*min:t+6*min) = (well_input.choke_press-5)
    *100000;
112           ActualChokePress(t+6*min:t+8*min) = (80)*100000;
113           ActualChokePress(t+8*min:t+10*min) = (10)*100000;
114           ActualChokePress(t+10*min:t+12*min) = well_input.choke_press
    *100000;
115           MudFlowIn(t:t+12*min) = well_input.mudflowin;
116           MudFlowIn(t+3*min:t+5*min) = well_input.mudflowin +.005;
117           MudFlowIn(t+5*min:t+7*min) = well_input.mudflowin -.005;
118           MudFlowIn(t+7*min:t+9*min) = .055;
119           MudFlowIn(t+9*min:t+11*min) = .015;
120           MudFlowIn(t+11*min:t+12*min) = well_input.mudflowin;
121           z = t;
122          for i = 1:12*min, %horizon length based on time_step
123               t = z + i;
124               if t>1
125                   o.BitPosition(t)=o.BitPosition(t-1)+(o.TimeStep*24)*ROP(t)
    ;
126                   o.DepthHole(t)=max([o.BitPosition(1:t) o.DepthHole(1)]);
```

```
127                 end
128                 % get well time passed in, use as inital point for predictions
129                 Time(t) = Time(z) + i * o.TimeStep;
130                 % Run Well Model
131                 setrealtime_controller
132                 % Send updated real time table to FlowModel.dll
133                 ReturnCode = calllib(o.model_instance, 'setmoduletable', '
    Input_RealTime', 14, realTable_controller, 43, 0, 0);
134                 % Execute FlowModel
135                 ReturnCode = calllib(o.model_instance, 'runflowmodel', o.
    TimeP_controller, 2, 0, 0);
136                 ReturnCode_getRealtime = calllib(o.model_instance, '
    getrealtimeparameters',o.RTOutP_controller,6);
137                 data_controller = get(o.RTOutP_controller,'Value');
138                 bhPres_controller           = data_controller.bhPresCalc;
139                 p_bit(t) = (bhPres_controller)/100000 ;
140             end
141             output = [(Time(z+2*min-10:end)-Time(z+2*min-10))*60*60*24;
    MudFlowIn(z+2*min-10:end); ActualChokePress(z+2*min-10:end)/100000; p_bit(
    z+2*min-10:end)]';
142             dlmwrite('discrete_states.csv',output);
143             data = output;
144             empirical.p_c_dev = output(1,3);
145             empirical.q_p_dev = output(1,2);
146             empirical.p_bit_dev = output(1,4);
147             apm_option(emp_ser,empirical.control_app,'u[1].upper',(.076-output
    (1,2)));
148             apm_option(emp_ser,empirical.control_app,'u[1].lower',(.005-output
    (1,2)));
149             apm_option(emp_ser,empirical.control_app,'u[2].upper',(120-output
    (1,3)));
150             apm_option(emp_ser,empirical.control_app,'u[2].lower',(5-output
    (1,3)));
151         end
152         function o = enddll(o)
153             calllib(o.model_instance, 'enddll')
```

```
154             clear (o.RTOutP_controller, o.TimeP_controller, '
    statusP_controller', 'sensorsDataP_controller')
155             unloadlibrary(o.model_instance)
156         end
157         function output = SS_data_for_model_fit(o,well_input)
158             m = 1;
159             gasinfluxrateP_controller = libpointer('doublePtr',0);
160             fid = fopen('SSdata.csv', 'w');
161             fprintf(fid,'q_p,p_bit,p_c,q_choke,q_res\n');
162             mudflows = linspace(.009,.063,10);
163             for j = 1:10
164                 chokes = linspace(3,(54059*mudflows(j)^2+9E-10*mudflows(j)+1)
    ,10);
165                 for k = 1:10
166                     ReturnCode = calllib(o.model_instance, 'setmoduletable', '
    Load_state', 10, 1, 0, 0, 0);
167                     t                                = well_input.t ;
168                     Time(t)                          = well_input.time;
169                     RotarySpeed(t:t+100)              = well_input.rotary_speed;
170                     MudDensityIn(t:t+100)             = well_input.mud_density_in
171                     InputMudTemperatureIn(t:t+100) = well_input.input_mud_temp
172                     ROP(t:t+100)                      = well_input.ROP;
173                     o.BitPosition(t:t+100)              = well_input.bitposition;
174                     o.DepthHole(t:t+100)                = well_input.depthhole;
175                     InputDesiredEMW(t:t+100)          = well_input.desired_emw;
176                     SetPointPosition(t:t+100)         = well_input.
    set_point_position;
177                     ActualChokePress(t:t+100)         = chokes(k)*100000;
178                     MudFlowIn(t:t+100)                = mudflows(j);
179                     z = t;
180                     for i = 1:20, %horizon length
181                         t = z + i;
182                       Time(t) = Time(z) + i * o.TimeStep;
183                         setrealtime_controller
184                         ReturnCode = calllib(o.model_instance, 'setmoduletable
    ', 'Input_RealTime', 14, realTable_controller, 43, 0, 0);
```

```
185                         ReturnCode = calllib(o.model_instance, 'runflowmodel',
     o.TimeP_controller, 2, 0, 0);
186                         ReturnCode_getRealtime = calllib(o.model_instance, '
    getrealtimeparameters',o.RTOutP_controller,6);
187                         data_controller = get(o.RTOutP_controller,'Value');
188                         bhPres_controller           = data_controller.
    bhPresCalc;
189                         getdoubleDataP_controller = libpointer('doublePtr',0);
190                         calllib(o.model_instance , 'getdouble', 'FlowInChoke',
     libpointer('uint32Ptr',11), o.getdoubleDataP_controller);
191                         FlowInChoke_controller = get(o.
    getdoubleDataP_controller,'Value');
192                     end
193                     SSdata(m,:) = [mudflows(j)*60 bhPres_controller/100000
    chokes(k) FlowInChoke_controller*60 gasinfluxrate_controller*60];
194                     m = m + 1;
195                 end
196             end
197             dlmwrite('SSdata.csv',SSdata,'-append');
198             fclose('all');
199             param_est;
200             output = 1;
201         end
202         function output = save_state(o)
203             ReturnCode = calllib(o.model_instance, 'setmoduletable', '
    Save_state', 10, 1, 0, 0, 0);
204         end
205     end
206 end
```

Listing D.5: High Fidelity Controller Code in MATLAB

```
1 function [rec_changes] = HiFi_Controller(input,hifi_cont,x0,bias)
2 IterateController = @(guess)hifi_cont.predictions(input,guess,x0,bias);
3 OPTIONS = optimoptions('fmincon','Algorithm','interior-point','TolFun',.001,'
    MaxFunEvals',250);
```

```
[x,FVAL,exitflag,showmestuff] = fmincon(IterateController,x0,[],[],[],[],[1
    .009],[45 .063],[],OPTIONS);
  if exitflag > 0
      status = 1;
  else
      status = 0;
  end
rec_changes = [x(1) x(2) status];
end
```

Listing D.6: Advanced switch logic code.

```
classdef Ensemble<handle
    properties
        emp_pred = [];
        low_pred = [];
        hifi_pred = [];
        meas_p_bit = [];
        mud_pump = [];
        choke_press = [];
        time = [];
        chosen_conts = [1 0 0];
        time_horizon = 20;
        SSE_max = 1;
        hifi_cont;
        well_model;
        empirical;
        well_input_store = [];
        TimeStep;
    end
    methods
        %% Receive measurement and predictions
        function o = receive_meas_and_pred(o, emp_pred, low_pred, hifi_pred,
   new_meas, mud,  choke, time)
            num_ep = numel(o.emp_pred);
            extra = num_ep - o.time_horizon +1;
            cutoff = max(extra,0);
```

```
25              o.emp_pred = [emp_pred o.emp_pred(1:end-cutoff)];
26              o.low_pred = [low_pred o.low_pred(1:end-cutoff)];
27              o.hifi_pred = [hifi_pred o.hifi_pred(1:end-cutoff)];
28              o.meas_p_bit = [new_meas o.meas_p_bit(1:end-cutoff)];
29              o.mud_pump = [mud o.mud_pump(1:end-cutoff)];
30              o.choke_press = [choke o.choke_press(1:end-cutoff)];
31              o.time = [time o.time(1:end-cutoff)];
32          end
33          %% Choose controllers
34          function [chosen_conts] = choose_controllers(o,well_input)
35              emp_err = sum((o.emp_pred - o.meas_p_bit(1:numel(o.emp_pred))).^2)
36              low_err = sum((o.low_pred - o.meas_p_bit).^2);
37              hifi_err = sum((o.hifi_pred - o.meas_p_bit).^2);
38              if emp_err > o.SSE_max*numel(o.emp_pred) || numel(o.emp_pred) < 5
39                  o.chosen_conts(1) = 0;
40                  o.chosen_conts(2) = 1;
41                  %give the new parameters some time before recalculating
42                  if numel(o.emp_pred)>=5
43                    data = o.hifi_cont.step_tests(well_input, o.empirical);
44                      %create data for fitting
45                    sysd = apm_id(data,2,2,2);
46                      %clear old and load new apm file
47                    apm(o.emp_ser,o.empirical.control_app,'clear apm');
48                    apm_load(o.emp_ser,o.empirical.control_app,'sysc.apm');
49                    apm(o.emp_ser,o.empirical.simulate_app,'clear apm');
50                    apm_load(o.emp_ser,o.empirical.simulate_app,'sysc.apm');
51                      %reset empirical prediction history
52                    o.emp_pred = [];
53                  end
54              else
55                  o.chosen_conts(1) = 1;
56                  o.chosen_conts(2) = 0;
57              end
58          end
59  end
```