Digital Twin and Smart Automation for Bitumen Extraction Process

by

Jansen Fajar Soesanto

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Process Control

Department of Chemical and Materials Engineering
University of Alberta

Abstract

The advent of Industry 4.0 integrates advanced digital technologies and Artificial Intelligence (AI) into system engineering. This research explores the potential of AI in smart automation for industries, bridging it with physics-informed approaches, particularly through Explainable Artificial Intelligence (XAI) and transfer learning from physics to AI. The study unfolds across three interconnected phases, each targeting a specific aspect of industrial automation, with a focus on the bitumen extraction process from oil sands. The solid form of oil sands presents a complex challenge in producing Synthetic Crude Oil (SCO), a process characterized by natural disturbances from ore quality and plant scheduling capacity in upstream mining. The Primary Separation Vessel (PSV), central to the extraction process, is interconnected with the secondary separation unit, with both impacting each other's optimization and control. Our focus is on the digital twin development and autonomous operation of the PSV, including autonomous Real-Time Optimization (RTO) and advanced control. The multi-input multi-output, nonlinear, high-dimensional state-action spaces, and constrained processes present additional challenges that we aim to tackle.

The first phase develops a high-fidelity digital twin for an industrial-scale bitumen extraction facility, incorporating multiparticle settling under non-ideal environments. Modifying the PSV model and integrating it with adjacent units, this plant-wide model accurately captures process dynamics, bitumen quality, and potential losses. High-dimensional parameters in the first-principle model are addressed using systematic parameterization techniques, Bayesian optimization, and sensitivity analysis to fully utilize industrial data. High-fidelity modeling proves crucial for automa-

tion validation and significantly contributes to the field of Explainable Reinforcement Learning (XRL).

The second work focuses on developing autonomous control strategies, introducing a Model Predictive Control (MPC) for multimode operation with disturbances. Plant-model mismatches causing fluctuations in MPC motivate the integration of Reinforcement Learning (RL) for model scheduling and multitasking in the third work. Alongside MPC, this work showcases the capabilities of Reinforcement Learning-based Controller (RLC), achieving performance comparable to MPC with less controller effort. Notably, RLC speeds up computation 10 times faster than MPC. This work extensively tests the continual learning of RLC in multi-mode operations, ensuring their adaptability to changing environments.

To enhance the feasibility of RL in real-world training, this study employs transfer learning approaches such as imitation learning and Simulation-to-Real (Sim2Real) pretraining. This strategy significantly reduces process trips during online training. Generative Adversarial Imitation Learning (GAIL) and Sim2Real pretraining decrease trip count by factors of 8 and 27, respectively, compared to direct agent training. GAIL opens new training pathways for agents in startup and shutdown tasks. The proposed "MPC Safeguarded Exploration" approach strategically uses the alarm system and existing MPC controllers to further decrease trips during online training while maintaining agent explorability and adaptability.

The third phase shifts to supervisory RTO in bitumen extraction, tackling the complexities of multivariable decision making and the interconnected extraction process under disturbances. This phase pioneers a novel framework that uses RL for setpoints optimization and multi-MPC scheduling. It combines the robustness of MPC with the adaptive optimization capabilities of RL to outperform existing operational strategies. First principle analysis elucidates and verifies the RL ability to manage trade-offs in microscale particle settling and balancing workload distribution across each unit to optimize the overall recovery rate. A key finding is that the RL

agent anticipate MPC control policy and optimize its strategies accordingly. This ability to foresee and integrate decisions across control layers enhances collaboration among decision-making layers and optimizes operations in the context of plant-wide connectivity. Furthermore, the agent manages a second objective in control performance by scheduling MPC models based on operational changes. The RL policy reveals that operational modes depend on factors beyond ore grades, such as tailings density. These insights underscore the significance of Explainable Reinforcement Learning (XRL) in enhancing the acceptability of RL in complex industrial applications. The exploratory power and explainability of RL policies open new avenues for real-world implementation, transitioning RL from a learning agent to a teaching agent approach in industrial automation.

"Great things are done by a series of small things brought together."

-Vincent Willem van Gogh

To my beloved parents and all my teachers.

Acknowledgements

I extend my sincere thanks for the opportunity to be part of the esteemed research group under my supervisor, Dr. Biao Huang. His expert guidance and encouragement have been instrumental in my growth and development over the past two years. Engaging in research under his tutelage has been a rewarding and life-turning experience.

Special thanks are due to our industrial partner from Imperial Oil. Their strong interest and commitment to innovative solutions in the smart automation project have significantly contributed to the success of this research. The practical and innovative approaches I learned from their company have been enlightening. Presenting our work at the 2023 INFORMS Annual Meeting was a significant opportunity, broadening the reach and impact of our research.

I would like to express my gratitude to the Alberta Machine Intelligence Institute (AMII) for their weekly artificial intelligence seminars, which keep us updated with cutting-edge research in Reinforcement Learning. Attending Upper Bound 2023 was an incredible experience, filled with inspiring talks and leading research. I am thankful for the opportunity to participate in such impactful sessions.

I owe a debt of gratitude to every member of the PDASA research group. More than just colleagues, they have been true friends, providing mentorship and support in both professional and personal spheres. Our interactions, ranging from productive discussions to social outings, have greatly enriched my life.

The University of Alberta deserves special mention for its commitment to high-quality research and academics. The university has provided numerous opportunities

for professional and personal growth, along with all the necessary resources to facilitate my studies in Canada.

I also wish to recognize the faculty and staff at the University of Alberta for their exceptional support, particularly during my initial transition period amid the pandemic.

I am grateful to the Natural Sciences and Engineering Research Council of Canada and our industrial partners for their support of this research through grants and funding opportunities.

Reflecting on my journey, I acknowledge that my past experiences, including my undergraduate studies and previous work surrounded by amazing and kind people, have endowed me with valuable knowledge and resilience, and taught me important life lessons.

Last but not least, I express my deepest gratitude to my family and friends for their constant support and encouragement. Special thanks to my parents, who, despite being on different continents for the past two years, have provided unwavering support and have played a crucial role in raising me and shaping who I am today.

Table of Contents

List of Tables

List of Figures

Abbreviations

A2C Advantage Actor-Critic.

A3C Asynchronous Advantage Actor-Critic.

AI Artificial Intelligence.

AIC Akaike Information Criterion.

APC Advanced Process Control.

ARMAX AutoRegressive Moving Average with eXogenous inputs.

ARX AutoRegressive with eXogenous inputs.

BC Behavioral Cloning.

BJ Box-Jenkins.

BPCS Basic Process Control System.

CMDP Constrained Markov Decision Process.

CPO Constrained Policy Optimization.

CPU Central Processing Unit.

CRL Constrained Reinforcement Learning.

CV Controlled Variable.

DDPG Deep Deterministic Policy Gradient.

DNN Deep Neural Network.

DP Dynamic Programming.

DQN Deep Q-Network.

DRL Deep Reinforcement Learning.

EMPC Explicit Model Predictive Control.

FM-MPC Fused Multi-Model Predictive Control.

FOMDP Fully Observable MDP.

FOPD First Order Plus Dead Time.

FT Flotation.

GAE Generalized Advantage Estimation.

GAIL Generative Adversarial Imitation Learning.

GAN Generative Adversarial Network.

GP Gaussian Process.

HVAC Heating, Ventilation, and Air Conditioning.

IAU Integral Absolute Controller Effort.

ISE Integral Square Error.

ITSE Integral Time Square Error.

KL Kullback–Leibler.

LIME Local Interpretable Model-Agnostic Explanations.

LQG Linear Quadratic Gaussian.

LQR Linear Quadratic Regulator.

LSTM Long Short-Term Memory.

LTI Linear Time Invariant.

MDP Markov Decision Process.

MIMO Multiple Input Multiple Output.

ML Machine Learning.

MV Manipulated Variable.

NARMAX Nonlinear ARMAX.

NN Neural Network.

OAT One At a Time.

OE Output Error.

OPP Ore Preparation Plant.

PFT Paraffinated Froth Treatment.

PID Proportional-Integral-Derivative.

POMDP Partially Observable MDP.

PPO Proximal Policy Optimization.

PRBS Pseudo Random Binary Signal.

PSV Primary Separation Vessel.

RBS Random Binary Signal.

RELU Rectified Linear Unit.

RGS Random Gaussian Signal.

RL Reinforcement Learning.

RLC RL-based Controller.

RNN Recurrent Neural Network.

RR Recovery Rate.

RTO Real-Time Optimization.

SAC Soft Actor-Critic.

SCO Synthetic Crude Oil.

SIL Safety Integrity Level.

Sim2ReaL Simulation to Real.

SIS Safety Instrumented System.

SISO Single Input Single Output.

SNR Signal to Noise Ratio.

TD Temporal Difference.

TD3 Twin Delayed DDPG.

TPE Tree-structured Parzen Estimator.

TRPO Trust Region Policy Optimization.

TSNE T-distributed Stochastic Neighbor Embedding.

XAI Explainable Artificial Intelligence.

XRL Explainable Reinforcement Learning.

Chapter 1

Introduction

This research covers the concepts of digital twin modeling and smart automation in Industry 4.0, exploring the integration of advanced digital technologies and Artificial Intelligence (AI) into process engineering. AI has made substantial strides in tackling complex tasks across diverse domains such as computer vision [1–6], video gaming [7–10], autonomous driving [11–17], robotics [18–20], and real-world autonomous control [21, 22]. In this transformative phase, the oil and gas industry, known for its complex and large-scale operations, stands to benefit significantly from these technological advancements.

At the core of this revolution lies the advent of autonomous systems — engineered for adaptive operation and minimal human intervention. It heralds a pivotal shift in how operations are conducted, positioning digital transformation as the primary driver of innovation. Thus, this digital evolution represents more than just a technical upgrade, but rather a strategic shift, setting the stage for the future landscape of industrial automation.

This research contributes to this transformative era by exploring AI potential and bridging them with physics-informed approaches, powered by data science and Machine Learning (ML). Our focus extends beyond exploring innovations to devising feasible solutions for smart automation in industrial applications. Incorporating physics knowledge is pivotal, achieved through the implementation of Explainable Artificial

Intelligence (XAI) and the transfer of learning from physics to AI systems.

Our approach leverages various ML methodologies, including supervised, unsupervised, and reinforcement learning (RL). Supervised learning is employed for data-driven system identification using closed-loop industrial data and for behavioral cloning as a precursor for RL pretraining. An unsupervised generative adversarial framework is utilized for RL agent pretraining. While both supervised and unsupervised learning have laid foundational ground in AI, RL stands out for its unique human-like learning capabilities. RL is characterized by temporal decision-making and adaptive development through environmental interaction. A recent breakthrough in RL has showcased its potential in exhibiting metacognitive abilities [23]. RL is primarily applied for autonomous control and optimization in this research.

1.1 Motivation

The surge of AI and ML in Industry 4.0 has brought forth new opportunities and challenges in the process engineering domain. These technologies, particularly ML, hold the potential to analyze vast data sets, identify patterns, and make predictions, thereby driving innovation and operational efficiency. However, integrating advanced computational tools with existing industrial processes, especially in sectors like oil and gas, necessitates novel approaches that consider the realism and safety concerns for industrial applications.

1.2 Thesis Objectives

This research aims to demonstrate feasible and safe automation strategies in real-world settings, pushing the boundaries of process intensification and automation in the era of Industry 4.0. The objective is multi-faceted, with the initial focus being on developing a high-fidelity digital twin to accurately represent the dynamics of industrial processes, enabling rigorous testing and validation of autonomous strategies.

Second, to design and implement advanced control strategies like MPC and RL-based controllers that handle dynamic changes and disturbances within industrial processes. Third, to explore autonomous decision-making at the supervisory level, optimizing economic and operational objectives using Explainable RL (XRL).

1.3 Thesis Outline

This thesis is structured in a three-paper format. It begins with an introduction and progresses through key areas as follows:

Chapter 2 delves into the mathematical foundation crucial for the main works of this research: Modeling, System Identification, Machine Learning (ML), Process Control, and Optimization. System identification covers various methods in first-principle and data-driven approaches modeling, including both open-loop and closed-loop systems, and discusses different types of surrogate models with their strengths and weaknesses. Additionally, it encompasses ML and conventional methods used in this research. Advanced control theories, starting from the fundamentals of optimal control and progressing to MPC, are also examined. The chapter discusses the principles of RL, including fundamental learning techniques based on dynamic programming, Monte Carlo, and TD learning, as well as a literature review of various RL algorithms, highlighting their characteristics, advantages, and limitations.

Chapter 3 introduces a modification in a primary separation model based on first-principle modeling techniques. This model realistically simulates the separation mechanism, capturing both interface level, density dynamics, and recovery rate. This model is expanded to a plant-wide process reflecting actual process configurations, serving as a digital twin of the bitumen extraction facility. Challenges in identifying a vast set of parameters are addressed through systematic optimization and sensitivity analysis. Digital twin enables the realistic replication of the process's steady-state and dynamic behavior. The model is then utilized to understand the process and to develop an autonomous control and optimization framework based on advanced tech-

niques like MPC and RL. This paper was presented at the 2023 INFORMS Annual Meeting and has been submitted to the Computers & Chemical Engineering journal (Manuscript ID: CACE-D-24-00011).

Chapter 4 explores autonomous control supporting the transition of oil sands operations to the Industry 4.0 standard. It details the design of an MPC with disturbance augmentation into model prediction, along with bias correction. The chapter assesses the safety and feasibility of RL-based controllers in both the training and the testing phases. Addressing feasibility issues with imitation learning and Simulation-to-Real (Sim2Real) pretraining approach. The proposed MPC safeguarded exploration ensures the safety integrity of RL deployment. This work suggests that RL is potentially safe for real-world implementation, given its self-learning and adaptive abilities, making it suitable for smart industry applications. This work was presented at the 2023 INFORMS annual meeting.

Chapter 5 focuses on autonomous operation in industrial processes with Explainable Reinforcement Learning (XRL). The RL agent demonstrates multitasking capabilities by orchestrating optimization and control tasks. This framework employs RL for MPC model scheduling and setpoint optimization. The agent, harnessing the power of RL, outperforms existing operational strategies in various scenarios and simultaneously discovers optimal policies. These policies, though complex, are explainable through physical knowledge, showcasing RL model-free learning capability and its potential in process intensification. Explainable policies open up new paradigms in RL application, not just for learning from the environment, but also for teaching optimal strategies. This work was presented at the 2023 INFORMS annual meeting. The final section concludes the work, paving the way for Industry 4.0 and recommending future work to ensure the implementation of these findings in smart industrial automation.

Chapter 2

Background

2.1 System Identification

System identification uses first principles, data-driven approaches, or a combination of both to capture the dynamics of physical systems. Modeling is the critical initial step in process design, optimization, and control. Developing an accurate yet tractable for its application requires balancing model fidelity and complexity. Reference [24] classifies the modeling spectrum based on the degree of knowledge of the underlying physics (fig. 2.1). First principles modeling necessitates an understanding of the fundamental physics governing a process. This scientific knowledge can be limited by the complexity of many chemical processes, governed by conservation laws, reaction kinetics, and thermodynamics [25]. Developing mechanistic models is resource-intensive. However, they reveal the underlying physics required to understand the actual phenomena. While physics defines the model structure, the physical parameters often need estimation from experiments or data. This combined approach is called grey-box or first principles data-driven modeling [24]. On the other hand, data-driven modeling, known as the black-box approach, identifies the system purely from experimental data. Data-driven methods offer a relatively quick and inexpensive model development. However, they require substantial data to avoid overfitting or wrong models which output unreasonable physics values. Black-box models cannot be tuned and do not provide scientific insights. In practice, data-driven models often complement the element of the first principles model that lacks physical knowledge. This modeling technique is called a hybrid modeling approach [26].

Models can be parametric and non-parametric. Parametric models have a defined structure with a relatively small number of parameters to describe the true process dynamics. Nonparametric models use less rigid structures, requiring potentially infinite parameters for exact representation [27]. Both conventional and machine learning algorithms estimate parameters of the predefined structure, which can be linear or non-linear. Examples of linear model structures include autoregressive with exoge-

nous inputs (ARX), autoregressive moving average with exogenous inputs (ARMAX), box-jenkins (BJ), and output error (OE). They are considered as a *dynamic model* that captures the time-dependent behavior. Nonlinear model families include Gaussian processes (GPs), fuzzy models, and nonlinear ARMAX (NARMAX). GPs and fuzzy models are static, while NARMAX is a dynamic model. Neural Networks (NNs) can be linear or nonlinear based on the activation functions, such as sigmoid, tanh, and ReLU. Based on neuron connections, NNs can also be static or dynamic.

Model selection depends on data availability, process complexity, and application requirements. Deep neural networks (DNNs) have revolutionized data-driven modeling through deep, convoluted structures that serve as universal function approximators. DNNs capture complex mechanisms of decision-making policies, process dynamics, generative AI, image processing, and other intricate phenomena [1, 28–32]. The prevalence of DNNs can be attributed to backpropagation, a simple yet powerful technique that recursively adjusts the strength of neuron connections to progressively reduce the defined cost function [33].

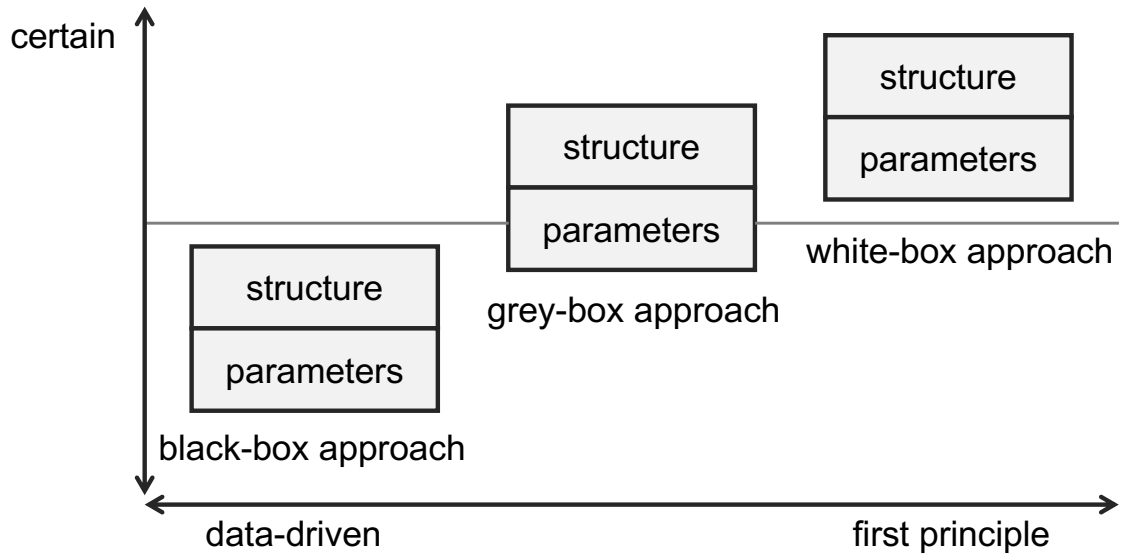


Figure 2.1: Modeling approaches from black-box to first principle

2.1.1 First Principle Model Parameter Identification

We used the constitutive relationship and mass conservation to develop the digital twin of the bitumen extraction process. Section 3.2 goes into detail on modeling. The first principle-based model was developed to study the particle separation mechanism and maintain high fidelity. The model includes empirical parameters that should be identified by fitting the model output to the real process data. However, a large number of parameters must be identified due to plant-wide modeling and high-fidelity requirements. These parameters are not available in the literature as we modified, scaled up, and expanded to the adjacent units that are tailored to the true physical system. We offer a systematic parameterization that makes use of three rounds of optimization in place of trial-and-error methods.

Steady-State Model Parameters

The digital twin model involves parameters that affect both steady-state and dynamic behavior. A steady state is the special case of a dynamic state that reaches the equilibrium point. While we aim to develop the dynamic model, fitting the steady-state output to the actual process roughly estimates some of the parameters that work for the dynamic condition. The steady-state behavior was validated against nine different scenarios to ensure a universally applicable model across operating conditions rather than overfitting to particular cases. The process design specification provides the actual steady state of the process. The model steady-state output is determined by solving a system of equations obtained by setting the time derivative term in the dynamic model to zero. Concurrently, the Tree-structured Parzen Estimator (TPE) optimization algorithm identified parameters that minimized the error between model outputs and industrial steady-state data. TPE evaluates multiple parameter sets, with each parameter lying within specified feasible ranges. Additionally, the tree structure inherently models conditional dependencies between parameters, making TPE well-suited for complex models with coupled parameters that must be

simultaneously solved. Hard constraints were imposed on feasible parameter ranges informed by physics and prior literature. Output constraints were incorporated as soft penalties on physically infeasible values as shown in eq. (2.1).

$$\begin{aligned} \theta^* = \arg\min_{\theta} \left[(\mathbf{y} - \hat{\mathbf{y}}(\theta))^T \mathbf{W} (\mathbf{y} - \hat{\mathbf{y}}(\theta)) + \mathbf{p}^T \max(\mathbf{0}, \mathbf{g}(\theta)) \right] \\ \text{s.t.} \quad \theta_i^L \leq \theta_i \leq \theta_i^U \end{aligned} \tag{2.1}$$

where θ^* is the set of optimal parameters, and θ_i^L and θ_i^U are the lower and upper bounds for parameters θ_i, respectively. $\hat{\mathbf{y}}(\theta)$ is the model output vector, $\mathbf{y}$ is the corresponding industrial data vector, $\mathbf{W}$ is the weighting matrix, $\mathbf{p}$ is the penalty coefficient vector, and $\mathbf{g}(\theta)$ is the inequality constraints on model outputs.

Dynamic Model Parameters

Initial parameters from the previous section approximate the optimal parameter values. Nonetheless, not all parameters affect steady-state behaviors; some are exclusively related to dynamic behaviors. These specific parameters, such as the volume of each modeled layer, can only be determined by fitting them to dynamic data. We fix the initial parameter value from the previous section and use TPE to identify the volume parameters. The objective is to fit the dynamic behavior of the model with the actual process.

The dynamic behavior is typically represented by process gain, time constant, and time delay. They are calculated using the method in section 2.1.2. In the true process, some variables behave like integrating processes, where the variables will not reach a steady state due to their slow dynamics and limited step testing time. Consequently, our focus shifted to comparing the rates of change. Specifically, the linear rate of change derived from the model step response, calculated as $\frac{K_p}{\tau}$, was compared to the integrating process gain obtained from the actual process step tests.

Sensitivity Analysis

To fine-tune the model parameters, local sensitivity analysis was performed by varying parameters and calculating percent changes in outputs. Derivative-based methods explore how model outputs are affected by perturbations in a single input around a nominal value. These methods are local using one-at-a-time (OAT) sampling [34]. Each parameter θ_i was perturbed $\pm 2\%$ and $\pm 5\%$ from its baseline $\bar{\theta}_i$, which is set to be θ_i^* as identified in the section 2.1.1.

The sensitivity index S_i of output Y to input θ_i can be represented by the partial derivative evaluated at the nominal baseline $\bar{\theta}$ (eq. (2.2)).

$$S_i(\bar{\theta}) = \left.\frac{\partial Y}{\partial \theta_i}\right|_{\bar{\theta}} C_i \tag{2.2}$$

where C_i is a scaling factor. Here, $C_i = \bar{\theta}_i / Y(\bar{\theta})$ scales the sensitivity to be the percent change in output over the percent change in input.

Since the analytical form of $\partial Y / \partial \theta_i$ is unknown, a finite difference approximation is used:

$$S_{Y,\theta_i} \approx \frac{Y(\theta_1, ..., \theta_i + \Delta\theta_i, ..., \theta_n) - Y(\theta_1, ..., \theta_i, ...\theta_n)}{\Delta\theta_i} \cdot \frac{\bar{\theta}_i}{Y(\theta_1, ..., \theta_i, ...\theta_n)} \tag{2.3}$$

where $\Delta\theta_i$ is the perturbation of θ_i.

2.1.2 First Order Plus Dead Time (FOPD) Model Identification

The step test is a typical procedure for identifying the First Order Plus Dead Time (FOPD) model of a physical system. A step input change is introduced, typically amounting to 1% of the input range of the actual process. Despite inherent non-linearities and the higher-order nature of the system, the FOPD model captures the step response dynamics of the system. FOPD model is a simplified representation of a dynamic system (eq. (2.4)).

$$\tau \frac{d\tilde{y}(t)}{dt} + \tilde{y}(t) = K_p u(t - T_d) \tag{2.4}$$

where K_p is the process gain, τ is the time constant, and T_d is the dead time. Process gain determines the magnitude of the steady-state output change in response to a step input. The time constant represents the duration for the process to reach approximately 63% of its process gain, while the dead time signifies the delay between the input and the onset of the output response (fig. 2.2). Practical applications of FOPD parameters include Proportional-Integral-Derivative (PID) controller tuning, setting control interval, safety design in delayed systems, stability analysis, and model validation (section 2.1.1). The K_p, τ, and T_d parameters are identified by fitting the system step response $\hat{y}(t)$ to the FOPD model output $\tilde{y}(t)$, formulated as follows:

$$\begin{aligned} \min_{K_p,\tau,T_d} \quad & \sum_{t=1}^{T} \left(\tilde{y}(t) - \hat{y}(t)\right)^2 \\ \text{s.t.} \quad & \tau > 0, \\ & T_d \geq 0 \end{aligned} \tag{2.5}$$

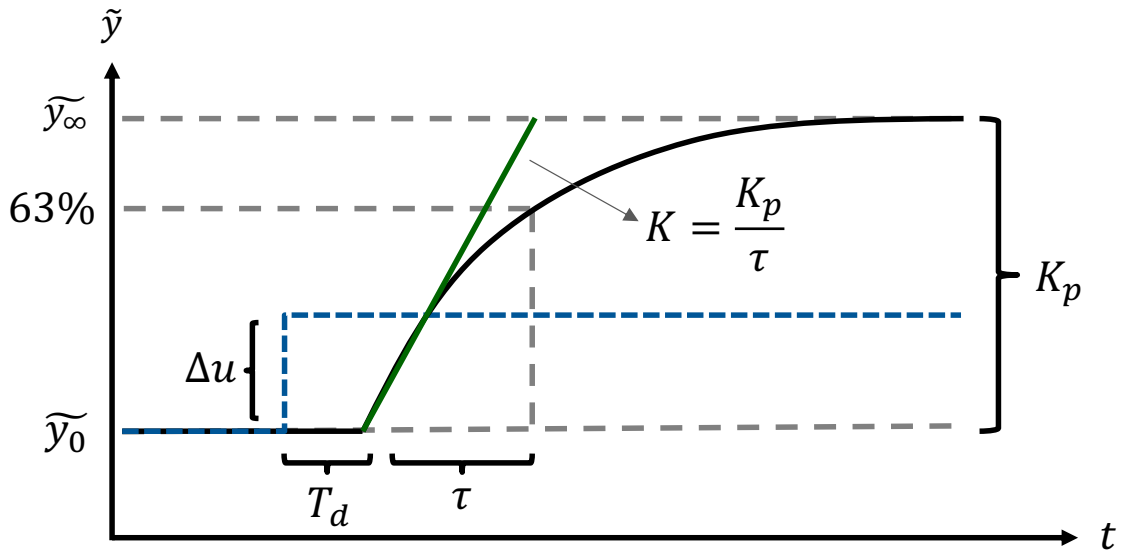


Figure 2.2: FOPD model step responses

2.1.3 Linear Time Invariant (LTI) Discrete State-Space Model Identification

Identifying a system from step tests is common when there is a sufficiently high signal-to-noise ratio (SNR). In significant noise environments, alternative input signals such as Random Binary Signal (RBS), Pseudo Random Binary Signal (PRBS), and Random Gaussian Signal (RGS) are more suitable. RBS and PRBS are essentially sequences of multiple step inputs, each designed to capture the system time constant, delay, and gain effects. It is important to sequence these inputs in a manner that does not prioritize one parameter over another to avoid biased identification [35]. This randomization approach helps in reducing systematic errors and noise effects. Characterized by its normal distribution, RGS provides extensive frequency coverage. This continuous signal is particularly valuable in identifying nonlinear dynamics, which might be missed by discrete bi-level signals like RBS and PRBS [36, 37]. Input design follows certain rules of thumb. the sampling time is set between 0.1 and 0.2 times the smallest time constant, and the frequency bandwidth is chosen to encompass slow dynamics at lower frequencies as well as the faster dynamics of the

process. The frequency range extends from zero to three times the process Nyquist frequency. The amplitude of input changes is adjusted to be both below and above the unit change used in step tests. In our specific control application, we identified the model with four input variables including three manipulated variables Q_m, Q_{dil}, Q_t, and one disturbance variable Q_{fd} (fig. 2.3).

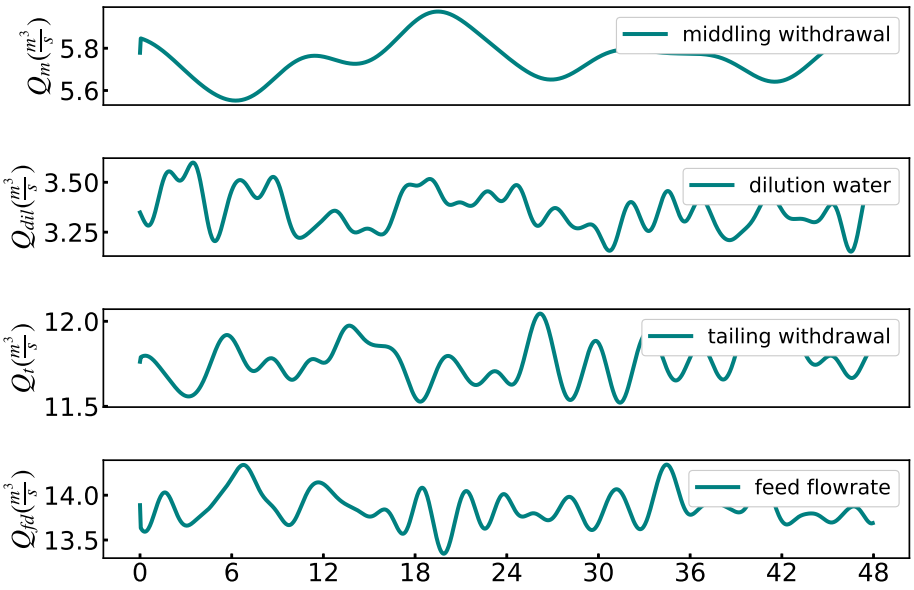


Figure 2.3: Random Gaussian signal inputs

We introduced the RGS inputs to the system to obtain open-loop dynamic responses, collecting 1,440 data points. We allocated 70% of this dataset to train a state-space model, with the remaining data split evenly for model validation and testing. Using Akaike Information Criterion (AIC) to balance model accuracy and simplicity, we identified the optimal model order with minimum AIC within the validation data. This state-space model enables the prediction of system trajectories within a model predictive control (MPC) framework for the multi-input multi-output (MIMO) control problem. Capturing the MIMO dynamics in state-space form allows effective prediction while keeping the model structure simple enough for online optimization. Additionally, compatibility with Kalman filtering facilitates real-time noise mitigation in the future. The state-space model takes the following mathematical

form:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.6}$$

$$y(t) = Cx(t) + Du(t) \tag{2.7}$$

where the vectors $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$, and $y \in \mathbb{R}^{n_y}$ denote the states, inputs, and outputs of the system, respectively. The matrices $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$, and $C \in \mathbb{R}^{n_y \times n_x}$ are the identified model parameters. The dimension n_x of the hidden states is adjusted based on the Akaike Information Criterion (AIC). A is the state matrix, B is the input matrix, C is the observation or output matrix, and D is the feedforward matrix which is set to zero due to zero-order hold in the sampling time. Discrete time models are often convenient if the system of interest is sampled at discrete time. If the sampling rate is chosen appropriately, the behavior between the samples can be safely ignored.

2.1.4 Long Short-Term Memory Closed-Loop Identification

An LSTM model is employed as a simulator for the offline training or pretraining of reinforcement learning (RL) agents. It is essential for the model to accurately capture the dynamics of the variables used in the RL agent states, rewards, and inputs. The intricate connections arising from a large number of input variables, the prediction output, and the system delay are key reasons for utilizing dynamic Deep Neural Networks (DNNs) as a surrogate model. Long Short-Term Memory Networks (LSTMs), a specialized class of Recurrent Neural Networks (RNNs), are adept at capturing temporal dynamic behavior. LSTMs effectively address the vanishing and exploding gradient problems, common in traditional RNNs, by incorporating input and forget gates. These gates facilitate better control over gradient flow and enhance the preservation of long-range dependencies [38]. An LSTM architecture employs specific equations to manage information flow and update cell states.

To accurately represent the true process, the simulator must account for actual

process disturbances such as ore grade and feed flow rate. These factors introduce delayed effects and significantly impact the dynamics of the system. The combination of these disturbances, along with variations in flow rate, can cause rapid divergence in the open-loop system. In an open-loop system, the interplay of these disturbances and flow rate changes can lead to rapid dynamic divergence. Consequently, conducting open-loop experiments to cover a wide range of operational modes, based on different grades and plant capacities over extended periods (months or even years), poses substantial challenges. To address this, we utilize a direct closed-loop identification method. This approach applies open-loop identification techniques but leverages data from a closed-loop system, which operates under feedback control. The following equations describe the closed-loop system:

$$y_t = G_p u_t + G_l e_t \tag{2.8}$$

$$u_t = -G_c y_t + G_c r_t \tag{2.9}$$

This equation can be rewritten as:

$$y_t = G_p u_t + G_l e_t \tag{2.10}$$

$$y_t = -G_c u_t + r_t \tag{2.11}$$

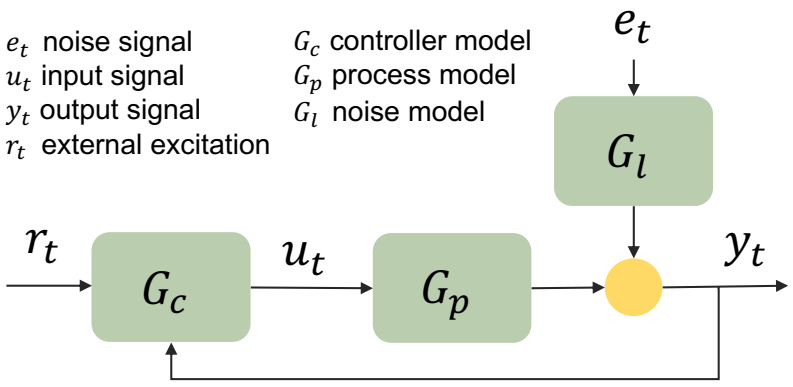


Figure 2.4: Closed-loop system

The correlation between unmeasurable noise e_t and the input u_t can lead to biased

estimates in closed-loop system identification (eq. (2.9)). When analyzing closed-loop data, the identified model might fit between both equations in section 2.1.4, whereas the objective is to identify the dynamic model of the process G_p only. To improve the fit with eq. (2.10), it is crucial to introduce sufficient disturbance. In the absence of external excitation r_t and noise e_t, the process model and controller should have different structures. In our case, we employ Deep Neural Networks (DNNs) for the process model. The DNNs are assumed to be close to the real process due to their distinct structure from the controller. However, the presence of external excitations remains the most reliable condition for closed-loop model identification.

In our experiments, the closed-loop system is randomly perturbed with grade and feed flowrate changes every hour, mimicking real operational conditions. Simultaneously, the setpoints of both controlled variables are randomly changed within their normal operation range. Furthermore, the surrogate model is designed to include at least a one-step delay, meaning the prediction y_t uses inputs from u_{t-1-n} to u_{t-1}, where the index n is the lookback period. These external excitations, coupled with the delay, ensure that the process model is identifiable and acts as a consistent estimator [35]. The consistency of the identified model is equivalent to simultaneous consistency in both the process and disturbance models. For this purpose, DNNs, as universal function approximators, are highly suitable.

The LSTM unit is based on a series of gates that control the information flow (eq. (2.12)).

$$
\begin{aligned}
f_t &= \sigma_g \left(W_f x_t + U_f h_{t-1} + b_f \right) \\
i_t &= \sigma_g \left(W_i x_t + U_i h_{t-1} + b_i \right) \\
o_t &= \sigma_g \left(W_o x_t + U_o h_{t-1} + b_o \right) \\
\tilde{c}_t &= \sigma_c \left(W_c x_t + U_c h_{t-1} + b_c \right) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \sigma_h \left(c_t \right)
\end{aligned} \tag{2.12}
$$

The LSTM equations assign a specific role to each variable:

- $x_t \in \mathbb{R}^n$: The input vector for the LSTM unit at time step t.
- $f_t \in (0,1)^h$: Activation vector for the forget gate.
- $i_t \in (0,1)^h$: Activation vector for the input/update gate.
- $o_t \in (0,1)^h$: Activation vector for the output gate.
- $h_t \in (-1,1)^h$: The hidden state vector, also known as the output vector of the LSTM unit.
- $\tilde{c}_t \in (-1,1)^h$: Cell input activation vector, representing potential updates to the cell state.
- $c_t \in \mathbb{R}^h$: Cell state vector, constituting the unit memory.

Activation functions utilized within the LSTM unit are as follows:

- σ_g: Sigmoid function, yielding values between 0 and 1.
- σ_c: Hyperbolic tangent function, with an output range between -1 and 1.
- σ_h: Typically a hyperbolic tangent function; however, variations such as the identity function $\sigma_h(x) = x$ are also considered, based on specific LSTM adaptations [39].

The initial values for the cell state c_0 and the hidden state h_0 are set to zero. $\odot$ denotes the Hadamard product, an element-wise multiplication operation. The subscript t indexes the time step, reflecting the sequential nature of LSTM operations. During the training process, learned parameters include weight matrices for the input connections $W_k \in \mathbb{R}^{h \times n}$, the recurrent connections $U_k \in \mathbb{R}^{h \times h}$, and the bias vector $b_k \in \mathbb{R}^h$. Here, n and h denote the number of input features and the number of hidden units, respectively. The subscript k specifies the LSTM gate component, which can be the input gate i, output gate o, forget gate f, or memory cell c. We employ vector notation to represent the multiple hidden units within a single LSTM cell. The number of hidden units is a tunable hyperparameter, allowing for adjustments in the trade-off between model complexity and accuracy.

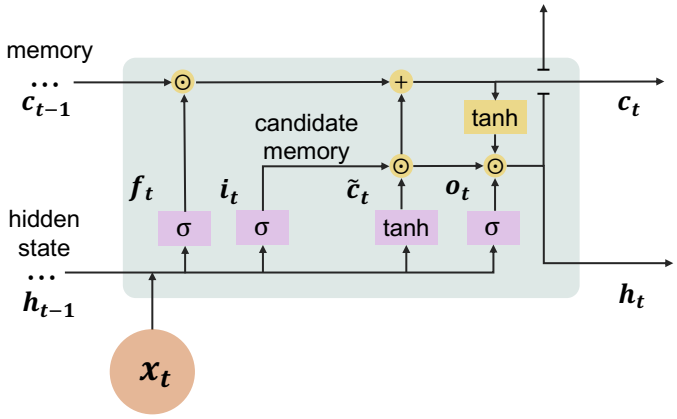


Figure 2.5: LSTM unit

2.2 Optimal Control

Classical control, typified by the Proportional-Integral-Derivative (PID) approach, predominantly employs frequency response techniques, focusing on achieving system stability, robustness, and consistent performance under disturbances and uncertainties. The widespread application of classical control is attributed to its simplicity and proven effectiveness, particularly in univariate control or where linear approaches are applicable. Subsequently, modern control theory emerged in the 1960s by formulat-

ing dynamic and control problems in state space, introducing concepts of controllability and observability, and designing state feedback controllers. By transitioning to the time domain and vector-matrix representations, modern control provided a solid mathematical foundation for classical techniques and broadened the range of controllable systems.

Optimal control formulates the control task as an optimization problem. It seeks to determine control signals that optimize certain performance metrics while adhering to system dynamics and physical constraints. Optimal control is especially valuable in scenarios with clearly defined and quantitatively modeled performance criteria. It is ideal for complex, multivariable, or constraint-laden environments. The Linear Quadratic Regulator (LQR) is a fundamental method in optimal control for linear systems, aiming to minimize a quadratic cost function that typically represents a balance between control effort and deviations from the desired zero state. Globally optimal state feedback laws for infinite horizons can be derived by analytically solving the Algebraic Riccati Equation. The Linear Quadratic Gaussian (LQG) control is another advanced control strategy that combines two major aspects of control theory: Linear Quadratic Regulator (LQR) and Kalman Filter. The Kalman Filter first provides an estimation of the current state. This estimation, despite the presence of noise, is then used by the LQR to compute the optimal control action. LQG control is used in scenarios where the system model is linear, and both the process and measurement noises are assumed to be Gaussian. Dynamic programming provides a broader framework for addressing optimal control problems by decomposing complex decisions into simpler, recursive stages. It entails deriving and solving Bellman's optimality equation to obtain an explicit feedback law offline. However, the curse of dimensionality has limited its applications. Model Predictive Control (MPC) achieves an optimal feedback control law by typically using numerical optimization to address open-loop, finite-horizon control problems in real time. MPC functions as a feedback control law by recalculating optimal control inputs at each sampling time based on

state updates from new measurements. Consequently, MPC is often synonymous with receding (or moving) horizon control, in which an explicit prediction horizon is optimized based on a dynamic process model. MPC adeptly handles complex systems with constraints, a challenging feat for purely analytical optimal control methods. It typically aims for local rather than global optimality, considering computational practicality.

Reinforcement learning, in contrast, seeks to discover the optimal control policy rather than a sequence of optimal control actions. The overarching goal is to learn a policy that maximizes quantitatively modeled rewards from environments but unknown system dynamics. While model-based RL might utilize estimated system dynamics for more efficient learning, policy learning fundamentally relies on interactions with the real environment. The optimal policy is "learned" through continual and "reinforced" interactions with the environment. RL algorithms are discussed in section 2.3.

2.2.1 Performance Measure and Constraints

The objective of an optimal control problem is to identify control signals that guide the system along trajectories minimizing a specified performance metric, while also adhering to certain constraints. Optimal control problems are thus described by their objective function and constraints. The table below summarizes common objective function types that can define an optimal control problem. **Terminal Control Problem** minimizes the deviation of the final state from its desired value at t_f. **Minimum Time Problem** seeks to transition a system from an initial state $x(t_0)$ to a specified target in the shortest possible time. **Minimum-Control-Effort Problem** moves the system from an initial state $x(t_0)$ to a target set S with a minimum control effort. **Tracking Problem** maintains the system state $x(t)$ as close as possible to a desired state $r(t)$ over the interval $[t_0, t_f]$. **Regulator Problem** is a special case of the tracking problem where the target state values are zero or constant. It is

Table 2.1: Performance Measure in Optimal Control

Performance Measure	**Description**
Terminal Control Problem	$J = [\mathbf{x}(t_f) - \mathbf{r}(t_f)]^T \mathbf{H} [\mathbf{x}(t_f) - \mathbf{r}(t_f)]$
Minimum Time Problem	$J = \int_0^T dt$
Minimum-Control-Effort Problems	$J = \int_{t_0}^{t_f} \left[\mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t)\right] dt$
Tracking Problem	$J = \int_{t_0}^{t_f} (\mathbf{x}(t) - \mathbf{r}(t))^T \mathbf{Q}(t)(\mathbf{x}(t) - \mathbf{r}(t))dt$

common to combine different performance measures in the formalization of optimal control problems.

Constraints are another component of optimal control formulation. Manipulated inputs (such as valve positions, throttle, current, voltages, and torques) in most physical systems have limits. If these input constraints are not respected by the controller, they are enforced by the physical system. Rate of change constraints avoid abrupt change that might cause reaction runaway, circuit short, water hammer, and other undesirable operations. Constraints on states or outputs are often imposed for reasons like safety, operability, or product quality. An important function of a controller is to determine in real-time whether output or state constraints are achievable and to relax them satisfactorily if not. Thus, the optimization problem is typically set up with hard constraints for input constraints and soft constraints for output or state constraints. This approach ensures that state constraints do not cause infeasible control problems, as they can be relaxed by choosing large values for the relaxation factor. However, large values may be undesirable as measured by the stage-cost function.

Dynamic Programming

The Dynamic Programming (DP) technique is based on the *principle of optimality*. This principle states that in an optimal trajectory, the remaining path from any state is an optimal subtrajectory. Consider an optimal policy $\pi^* = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ where state X_i has nonzero probability of occurrence at time i . In formulating con-

trol from X_i at time i onwards, the segment $\{\mu_i, \mu_{i+1}, \ldots, \mu_{N-1}\}$ constitutes optimal control for this subproblem. A key idea of DP is that optimal solutions are incrementally constructed from optimal solutions to smaller subproblems. The process involves defining a series of value functions $V_1, V_2, \ldots, V_n$, where each $V_i(x)$ corresponds to the value of state x at time i. The values are computed in reverse, using the Bellman equation which defines a recursive relationship (eq. (2.13)). Backward calculation proceeds until V_1 optimal solution is obtained.

$$V_{i-1}(y) = \max\left[\text{gain}_{i-1} + V_i(\text{new state})\right] \tag{2.13}$$

While DP was originally developed for discrete problems, Bellman extended this concept to continuous control problems. Control theory aims to solve an admissible control policy $\mathbf{u}^*$ that minimizes a performance measure along a desired system trajectory $\dot{\mathbf{x}}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t)$. One way is by solving the Hamilton-Jacobi-Bellman equation (eq. (2.14)).

$$-J_t^* = \min_{\mathbf{u}}\{f(\mathbf{x}(t), \mathbf{u}(t), t) + J_x^* \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t)\} \tag{2.14}$$

where $J_x^* = \frac{\partial J^*}{\partial \mathbf{x}}$ and $J_t^* = \frac{\partial J^*}{\partial t}$. For discrete systems, the Bellman equation is reformulated as eq. (2.15).

$$J_k^*(\mathbf{x}_{n-k}) = \min_{\mathbf{u}_{n-k}}\{\hat{f}(\mathbf{x}_{n-k}, \mathbf{u}_{n-k}) + J_{k-1}^*(\hat{\mathbf{g}}(\mathbf{x}_{n-k}, \mathbf{u}_{n-k}))\} \tag{2.15}$$

applied at the k-th stage of n discrete time intervals.

2.2.2 Model Predictive Control

Model Predictive Control (MPC) emerged in the late 1970s and 1980s to handle constrained multivariable control problems, commonly encountered in process industries. MPC circumvents the challenges of solving Bellman's optimality equation, which is often intractable for nonlinear systems and in the presence of inequality constraints. The fundamental concept of MPC involves using a dynamic model to predict system

behavior over a finite prediction horizon. It optimizes this forecast by producing a series of optimal control decisions. Within a control horizon, only the first control move u_0 is implemented at the current time. At each time step, MPC uses current measurements to estimate the initial state of the system. The process repeats at $t+1$ with the new state measurement x_{t+1}. This defines an implicit control law $u_t = U(x_t)$, where U represents the solution operator for the minimization problem.

Explicit MPC (EMPC), in contrast, pre-computes the control law for every possible state, reducing online computational demands. A traditional (implicit) MPC controller is first designed to generate an explicit MPC controller. EMPC requires less computational effort and provides faster solutions in real time. However, its heavier offline computational load and larger memory footprint may restrict its applications to scenarios with relatively low dimensions, short prediction horizons, and few output constraints. A general MPC problem is formulated in eq. (2.16).

$$\min_{\mathbf{u}} J = \frac{1}{2}\sum_{k=0}^{N-1}\left(\mathbf{x}_k^T Q \mathbf{x}_k + \mathbf{u}_k^T R \mathbf{u}_k + \Delta \mathbf{u}_k^T S \Delta \mathbf{u}_k\right) + V_f(\mathbf{x}_N) \tag{2.16}$$

$$\begin{aligned} \text{s.t.} \quad & \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \\ & \mathbf{x}_0 = x(t), \\ & \mathbf{u} \in \mathcal{U}, \\ & \mathbf{x} \in \mathcal{X}, \\ & \mathbf{x}_N \in \mathcal{X}_f, \\ & \Delta \mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_{k-1}. \end{aligned}$$

where $\mathbf{x}_k$ and $\mathbf{u}_k$ denote the state vector and control input vector at step k, respectively. The matrices Q, R, and S are weighting matrices that define the cost associated with the state, control effort, and control input changes, respectively. $V_f(\mathbf{x}_N)$ is the terminal cost function, a function of the final state $\mathbf{x}_N$ at the end of the horizon. $f(\mathbf{x}_k, \mathbf{u}_k)$ is the system dynamics function, with $\mathcal{U}$ and $\mathcal{X}$ being the admissible sets

for the control inputs and states, respectively. $\mathcal{X}_f$ is the terminal constraint set for the final state, and $\Delta \mathbf{u}_k$ is the change in control input from the previous time step.

Output feedback, often implemented as bias correction, aligns model predictions with actual system behavior. This correction accounts for plant-model mismatches by adjusting model predictions using recent measurements. The model output is added with a bias term, which is the difference between current measurements and predicted values. Under certain disturbances, this feedback approach might lead to suboptimal responses, necessitating the exploration of alternative feedback and disturbance estimation techniques.

2.3 Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning that focuses on training algorithms to make a series of decisions that maximize cumulative reward over time. RL agents learn by interacting with an environment, where they learn from trial and error to achieve a goal or maximize a cumulative reward signal. This learning process is driven by the experiences and environmental feedback. RL is particularly effective at solving complex control and optimization problems that challenge traditional methods. In control systems, RL can adaptively manage the autonomous control of dynamic systems like autonomous vehicles, robots, or heating, ventilation, and air conditioning (HVAC) systems in buildings. RL introduces a paradigm for defining optimal control policies differently from model predictive control (MPC). Rather than solving optimal control actions sequentially, RL employs machine learning to learn policies directly from experience. This facilitates developing offline and model-free solutions. Techniques like Q-learning and policy gradients are model-free, enabling effective policies for complex sequential decisions without relying on accurate system models. Incorporating models addresses sample efficiency limitations in model-free methods. Exploration allows avoiding local optima and ensuring adaptability. For optimization, RL provides a framework to find robust solutions in uncertain, changing

environments for multitasking problems like supply chains, power grid managements, and chemical processes [40–43]. The adaptability and autonomous learning of RL make it well-suited to continually improve efficiency and outcomes across applications.

Learning methods in RL are mainly based on dynamic programming, Monte Carlo methods, and temporal difference learning. These provide mathematical frameworks to solve sequential decision-making problems in Markov decision processes. We will discuss each of these key components for formulating an RL problem in the subsequent sections.

2.3.1 Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) provide a framework for managing dynamic decision-making processes in stochastic environments. In this research we focus on discrete-time models, observing the system at times $t = 1, 2, ..., n$, where n is the horizon, which can be finite or infinite. The former is called finite horizon MDPs, while the latter is called infinite horizon MDPs. Finite MDPs represent episodic tasks that terminate when a certain terminal state is reached. Each episode has a clear beginning and ending. Infinite horizon MDPs are typically applied to continuing tasks. In these scenarios, the policy-maker interacts with the environment indefinitely, without a predetermined endpoint. The policy evaluation and improvement processes consider the long-term consequences of actions, potentially extending to an infinite time horizon. However, continuing tasks can be formulated as finite MDPs by setting an artificial termination point that can be time-based, resource-based limits, performance benchmarks, and other criteria. The term "continuing task" in reinforcement learning, is often confused with "continuous task" and "continual learning." "continuous task" denotes tasks with continuous action spaces, while "continual learning" describes an agent ongoing learning process through continuous interaction with the environment.

MDPs follow Markovian dynamics where the next state depends only on the cur-

rent state and action. The system transitions probabilistically between states as a function of current states and actions. Actions also influence immediate rewards and future rewards. Many applications can be cast as MDPs and solved by dynamic programming, Monte Carlo, Q-network, and other algorithms including RL.

In Fully Observable MDPs (FOMDPs), the agent has complete visibility of the system state at every decision point. This full observability simplifies decision-making processes. A FOMDP is defined as a tuple (S, A, P, R, γ) where S is the set of states, A is the set of actions, P is the state transition probability matrix with $P(s'|s, a)$, representing the probability of transitioning to state s' from state s after taking action a, R is the reward function, and γ is the discount factor.

Partially Observable MDPs (POMDPs) extend MDPs to scenarios where the agent does not have full visibility of the system state, introducing uncertainty and incomplete information into the decision-making process. A POMDP is defined as a tuple $(S, A, P, R, \gamma, O, \Omega)$ where O is the set of observations, and Ω is the observation probability function, $\Omega(o|s', a)$. The primary objective is to determine an optimal decision-making policy which effectively navigates the inherent uncertainties in state information. This model is more representative of real-world processes, where state information is often not fully available or discernible. Estimating or inferring states from observable data is necessary. Other techniques such as as belief state updates and information state methods are commonly used to address this challenge.

Constrained MDPs (CMDPs) are a special form of MDPs where decision aims to maximize expected returns while adhering to safety constraints. A CMDP is defined as a tuple $(S, A, P, R, \gamma, C, D)$, where C represents a set of constraints, and D represents a set of safety constraints. CMDPs are particularly useful in scenarios where safety, resource limitations, or regulatory requirements are as important as (or more important than) the primary objective. They are crucial in applications where standard RL methods may not be safe.

We formulate RL as an MDP problem where an agent acts as the decision-maker

and the environment includes all external factors beyond the agent's control but influenced by its policy. States provide the context or information relevant to decision making process. The environment evolves in response to actions, emitting reward signals. The agent aims to maximize the return, the cumulative sequence of rewards, typically using discounted rewards over an infinite horizon. The expected return is expressed as the sum of discounted rewards:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.17}$$

where γ, with $0 \leq \gamma \leq 1$, is the discount rate. A higher discount rate indicates a more farsighted agent. The interaction may continue indefinitely, but the total reward remains finite as long as each reward is finite or $\gamma < 1$.

The agent learns the policy which maps states to action probabilities. Policies can be deterministic or stochastic. The deterministic policy π maps a state s to an action a, while stochastic policy $\pi(a|s)$ is the probability of taking action a given state s. Through interaction with the environment, the agent learns an optimal policy that maximizes expected return. Value functions represent the expected return under a specific policy, measuring the value of states $V(s)$ or state-action pairs $Q(s, a)$. The state value function of state s under policy π, $V_\pi(s)$, is the expected return starting in s and following π thereafter. The state-action value function for state s and action a under policy π, $Q_\pi(s, a)$, is the expected return starting from s, taking action a, and then following π. Due to the high dimensionality of states and actions, the functions $V(s)$ and $Q(s, a)$ are typically defined as parameterized functions. Reinforcement learning algorithms commonly involve computing or approximating these value functions.

2.3.2 Dynamic Programming in Reinforcement Learning

The link between Dynamic Programming (DP) and Reinforcement Learning (RL) lies in their shared use of value functions to systematically direct the search for effective

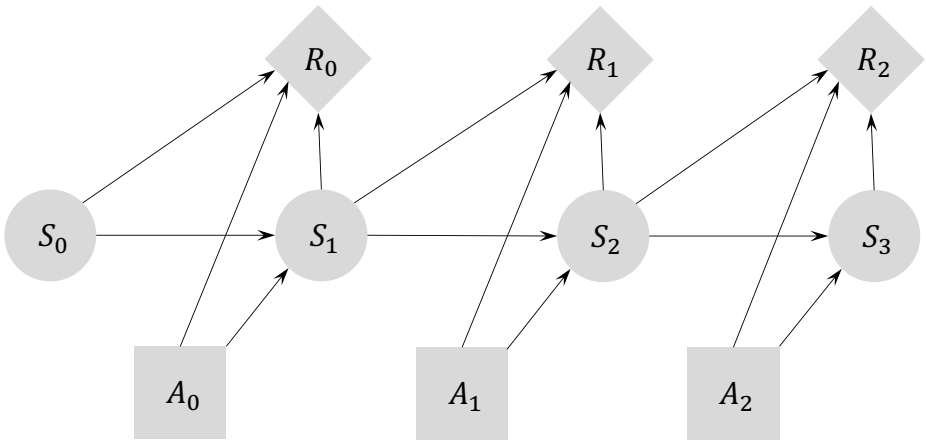


Figure 2.6: Markov decision process

policies. DP provides the theoretical foundation for both policy iteration and value iteration, involving the Bellman optimality principle to determine the best action at each state. Werbos (1977) proposed "heuristic dynamic programming," an approach to approximating DP that emphasizes gradient-descent methods for continuous-state problems, closely related to RL. Watkins (1989) explicitly connected RL to DP, characterizing a class of RL methods as "incremental dynamic programming." Both DP and RL generally use value functions to organize and structure the search for effective policies. Optimal policies can be easily identified after computing the optimal value functions, v^* or q^*, which satisfy the Bellman optimality equations:

$$v^*(s) = \max_a \mathbb{E}\left[R_{t+1} + \gamma v^*(S_{t+1}) \mid S_t = s, A_t = a\right] \tag{2.18}$$

$$= \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma v^*(s')\right] \tag{2.19}$$

or

$$q^*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q^*(S_{t+1}, a') \mid S_t = s, A_t = a\right] \tag{2.20}$$

$$= \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \max_{a'} q^*(s', a')\right] \tag{2.21}$$

The Bellman optimality equations can be intractable, especially for high-dimensional problems. DP iteratively evaluates value functions (*policy evaluation*) and improves policies (*policy improvement*). Policy Evaluation computes the state value for a given

policy:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \tag{2.22}$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \tag{2.23}$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] \tag{2.24}$$

$$= \sum_{a\in\mathcal{A}(s)} \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{2.25}$$

Value functions are commonly estimated by iterative policy evaluation. This process recursively learns the value function by using the Bellman equation as a recurring update mechanism:

$$V_{k+1}(s) = \mathbb{E}_\pi[R_{t+1} + \gamma V_k(S_{t+1})|S_t = s] \tag{2.26}$$

$$= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V_k(s')] \tag{2.27}$$

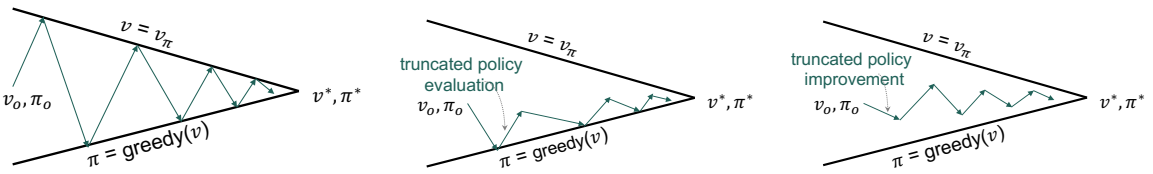
where V_k is the value function estimate, γ is the discount factor, p is the state transition probability, r is the immediate reward, and s' is the next state. This iterative process converges to v_π as k approaches infinity, under conditions that guarantee the existence of v_π. Action-value function $q_\pi(s, a)$ evaluates the advantage of choosing an action a in state s and then following the current policy π thereafter.

$$q_\pi(s, a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s, A_t = a\right] \tag{2.28}$$

$$= \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{2.29}$$

If a particular action consistently yields better returns in a given state (s), adopting the policy that favors that state-action pair improves overall return. Formally, policy improvement is the process of creating a new policy that improves upon an initial policy by making it greedy with respect to the state-action value function $q_\pi(s, a)$ of the original policy.

Optimal policy derivation hinges on the alternating processes of policy evaluation



(a) (b) (c)

Figure 2.7: Optimal policy learning: (a) Policy iteration; (b) Value iteration; (c) Generalized policy iteration

and policy improvement. This interplay forms the basis of distinct approaches: *policy iteration, value iteration,* and *generalized policy iteration* (fig. 2.7).

Policy iteration consists of a cyclic process of policy evaluation and policy improvement. These two processes alternate, each completing before the other begins. The process begins with an evaluation of initial policy π to determine its value function v_π. Based on the evaluated value function, the policy is updated into π'. This iterative procedure results in a succession of policies and corresponding value functions.

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \cdots \xrightarrow{E} \pi^* \xrightarrow{E} v^*$$

where E denotes a policy evaluation and I denotes a policy improvement.

Policy iteration is known for its relatively slow convergence. Each iteration involves a full evaluation of the current policy to compute the converged value function prior to policy improvement. Nonetheless, convergence to v_π occurs asymptotically. Just the initial few iterations are adequate for approximating the value function. Additional evaluations do not influence policy improvement. Therefore, the policy evaluation can be truncated through various methods without losing the convergence.

Value iteration merges policy improvement with a truncated policy evaluation process, where only a single iteration of policy evaluation occurs between each policy improvement:

$$v_{k+1}(s) = \max_a \mathbb{E}[R_{t+1} + \gamma \cdot v_k(s')] \tag{2.30}$$

The value iteration update is similar to the policy evaluation update, but it requires selecting the maximum value across all actions. Although value iteration technically

needs infinite iterations to exactly converge to v^*, in practice, it stops when the improvement falls within a tolerance.

Asynchronous Dynamic Programming methods intertwine evaluation and improvement without systematic sweeps of the state. It updates the state value function in a non-sequential manner, utilizing the values of other states as they become available. This approach allows for a more dynamic and adaptable updating process, rather than following a strict, predetermined order. As long as both processes keep updating all states, they converge to the optimal value function and policy.

The concept of Generalized Policy Iteration (GPI) refers to the broader strategy of allowing policy evaluation and policy improvement processes to interact, regardless of the granularity of these processes. All these iterative methods are fundamentally based on dynamic programming techniques.

2.3.3 Monte Carlo

Unlike dynamic programming, Monte Carlo methods do not assume the knowledge of the environment dynamics. They require only sequences of states, actions, and rewards sampled from interaction with an environment. These methods estimate the value of a state by averaging the returns observed after visits to that state. A distinct feature of Monte Carlo methods is the independence of each state estimate, in contrast to DP methods, which rely on bootstrapping.

The computational cost of estimating the value of a single state in Monte Carlo methods is independent of the total number of states. This makes them particularly advantageous for situations where only specific states are of interest. Monte Carlo methods encompass both every-visit and first-visit approaches, which differ in their method of averaging returns. The every-visit method calculates the average return for all visits to a specific state-action pair, whereas the first-visit method computes the average return only for the first occurrence of that state-action pair in each episode.

A significant challenge arises when certain state-action pairs remain unvisited,

particularly when using a deterministic policy π. In such cases, returns are observed for only one of the available actions within each state, making it impossible to improve the Monte Carlo estimates for the unvisited actions. This necessitates exploration techniques, such as Epsilon-Greedy, Boltzmann exploration, and Upper Confidence Bound (UCB).

Monte Carlo methods offer a sampling-based approach to approximate action value function $q_\pi(s, a)$ and can be viewed as a form of Generalized Policy Iteration (GPI). The idea is to estimate the value function by following a policy, averaging returns over multiple episodes, and then updating the value function at every visited state.

2.3.4 TD Learning

Temporal difference (TD) learning combines concepts from Monte Carlo methods and dynamic programming in reinforcement learning. Like Monte Carlo, TD methods use estimates as a learning target. However, TD methods update estimates without awaiting final outcomes like bootstrapping in DP. The value function for a state is updated as follows:

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right] \tag{2.31}$$

where α represents the learning rate, γ is the discount factor, and R_{t+1} is the immediate reward received after transitioning to state S_{t+1}.

The TD error δ_t computes the difference between current estimate of $V_\pi(s)$ and the updated estimate:

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{2.32}$$

Mathematically, TD learning methods converge towards the true state-value or action-value functions as experience grows, under the same conditions as Monte Carlo methods. By incorporating additional information from estimated successor states, TD often converges faster while requiring less experience.

There are variations of TD learning, such as N-step TD, which updates the value function based on information from the next n steps. This approach offers a middle ground between one-step TD learning and Monte Carlo methods, which wait until the end of the episode. TD(λ) generalizes both N-step TD and one-step TD by introducing a parameter λ to control the balance between short-term and long-term predictions. When λ is set to 0, it resembles one-step TD, and as λ approaches 1, it behaves more like Monte Carlo methods.

2.3.5 RL Algorithm taxonomy

Reinforcement Learning (RL) algorithms can be categorized based on whether they incorporate a model of the environment. Model-based RL attempts to learn a model or utilize the available model to predict state transitions and rewards. It leverages model prediction to plan and choose optimal actions. This can greatly improve sample efficiency compared to model-free methods that learn policies directly through trial-and-error interaction. However, learning accurate models is challenging, and model bias can significantly impact performance. By contrast, model-free methods forgo potential improvement in sample efficiency by planning models but tend to be more straightforward to implement and tune.

RL algorithms can be off-policy or on-policy. Off-policy RL like Q-learning separates the *behavior policy* used for exploration from the *target policy* that we want to optimize. Off-policy algorithms promote wider exploration, without interfering convergence of the target policy. By allowing reuse of any past experience, off-policy methods can make better use of accumulated data. Off-policy methods result in improved sample efficiency and enhanced learning stability. However, performance instability can occur when outdated samples imply contradictory state values relative to the current policy. This instability risk is exacerbated by the *deadly triad* - combining TD learning (or bootstrapping) value updates, nonlinear function approximation like neural networks, and off-policy can substantially increase the chances of

divergence [44]. Techniques like constraint enforcement, clipped updates, or larger replay pools avoid instabilities. On-policy methods, on the other hand, ensure a close alignment between the behavior policy and the target policy. This alignment helps maintain learning coherence and stability, as the state-action values reflect the current policy without the interference of outdated experiences. On-policy methods typically face fewer issues related to the deadly triad compared to off-policy methods. On-policy algorithms can be described as "learning on the job," learning about policy π from experiences sampled under π, while off-policy algorithms, which learn from experiences sampled under a different policy μ, can be likened to "learning by observation."

RL algorithms either optimize policies directly or learn value functions and derive corresponding policies. *Value-based* methods use Bellman equations to evaluate state-action pairs $Q_\theta(s, a)$, indirectly inducing policies by selecting actions that maximize Q-values. However, searching for optimum actions grows expensive with large or continuous action spaces. *Policy-based* methods directly adjust policy parameters towards better performance based on policy gradients, facilitated by on-policy data. These algorithms can learn both deterministic and stochastic policies, whereas value-based methods are limited to deterministic policies. Stochastic policies, which introduce randomness in action selection, aid in exploring a broader range of state-action pairs and help avoid local optima. *Actor-critic* frameworks do both, learning a value function $V_\phi(s)$ which is then used to guide and inform the updates of policy parameters.

2.4 Reinforcement Learning Algorithms

2.4.1 Value-Based Method

Q-Learning

Q-learning is an off-policy Temporal Difference (TD) control algorithm used to estimate the optimal policy π^* by learning the optimal action-value function $Q^*(s, a)$. It learns the optimal policy (or target policy) independently of the current policy (behavior policy). Q-learning can only handle discrete, finite MDPs and might converge slowly in large state spaces. Using function approximation in Q-learning can sometimes lead to a failure of convergence. Double Q-learning is a variation of the Q-learning algorithm aimed at reducing overestimation bias, thereby improving the stability and convergence of Q-learning [45]. The maximization step in the update rule tends to select overestimated values since the same Q-value function selects and evaluates actions. Double Q-learning decouples the selection from evaluation by maintaining two independent Q-value functions, referred to as Q1 and Q2. Q1 selects the best action $a = argmax_{a'} Q1(s, a')$, while Q2 evaluates that action to update the target Q-value. By breaking the max dependency between action selection and value update, this difference in targets helps address the positive bias. Deep Q-Network (DQN) is a deep reinforcement learning algorithm that combines Q-learning with a deep neural network serving as the function approximator [7]. Rainbow DQN integrates multiple improvements to the Deep Q-Network (DQN) algorithm, creating a more robust and efficient learning framework. These enhancements include double Q-learning, prioritized experience replay, dueling networks, multi-step Learning, distributional RL, and noisy nets [46].

Algorithm 1 Q-learning (off-policy TD control)

Input: Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal,\cdot) = 0$
Parameter: step size $\alpha \in (0,1]$, small $\epsilon > 0$
for each episode **do**
  Initialize s
  for each step of episode **do**
    Choose a using policy derived from Q (e.g., ϵ-greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma \max_a Q(s',a) - Q(s,a)]$
    $s \leftarrow s'$
  end for
end for until s is terminal

SARSA

SARSA is an on-policy TD algorithm. SARSA may converge to suboptimal policies and require more time to learn as it must simultaneously explore and exploit. Expected SARSA reduces variability in target values by taking the expectation over all possible next actions instead of the single sampled next action. This expected value provides more stable learning. Double SARSA is another modification that keeps two separate estimators to reduce overestimation bias, similar to Double Q-Learning. Eligibility traces are incorporated in SARSA(λ) to dynamically balance TD and Monte Carlo targets for faster learning and lower bias. Another version like Variable Lambda SARSA adapts the traces themselves, tuning the decay factor based on state visitation frequency. Delayed SARSA introduces delays in weight updates to stabilize the next action, reducing bias from asynchronous updates.

Algorithm 2 SARSA (on-policy TD control)

Input: Initialize $Q(s, a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$, arbitrarily except that $Q(terminal, \cdot) = 0$
Parameter: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
for each episode **do**
  Initialize s
  Choose a using policy derived from Q (e.g., ϵ-greedy)
  for each step of episode **do**
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ϵ-greedy)
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$, $a \leftarrow a'$
  end for until s is terminal
end for

2.4.2 Policy-Based Method

Policy Gradient Methods

Policy Gradient is an on-policy algorithm that optimizes the policy directly by computing gradients of the expected reward with respect to the policy parameters. Algorithms like REINFORCE and Actor-Critic are fundamental policy gradient techniques. The gradient is computed as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) \cdot A_{\pi_\theta} \right] \tag{2.33}$$

where $\nabla_\theta J(\theta)$ is the gradient of the objective with respect to the policy parameters, $\pi_\theta(a|s)$ is the policy (probability of taking action a in state s). A_{π_θ} represents the advantage of being in state s, taking action a, and subsequently following policy π_θ.

Advantage function provides a normalized measure of how advantageous a particular action is relative to the current policy. It reduces the variance of the update signal, which can accelerate learning. The method to calculate the advantage depends on the specific needs of the algorithm, with each method affecting the bias-variance balance in distinct ways:

- The basic advantage estimation is calculated as $A(s, a) = Q(s, a) - V(s)$, which represents the difference between the action-value and the state-value function.

- The Temporal Difference (TD) Error is expressed as $A(s,a) = r+\gamma V(s')-V(s)$, and it uses the value function estimate at the next state to improve the current state value estimate.

- Generalized Advantage Estimation (GAE) is given by $A^{GAE}(\gamma,\lambda) = \sum_{t=0}^{\infty}(\gamma\lambda)^t\delta_t^V$, with the TD error $\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$, aiming to reduce variance by averaging over a longer trajectory.

- The n-step Advantage takes into account a sequence of rewards and is computed as $A(s_t,a_t) = \sum_{k=0}^{n-1}\gamma^k r_{t+k}+\gamma^n V(s_{t+n})-V(s_t)$, balancing immediate and future rewards.

Algorithm 3 Policy Gradient Method

1: **Input**: Policy parameters θ_0, Value function parameters ϕ_0
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect trajectories $\mathcal{D}_k = \tau_i$ by executing policy $\pi_k = \pi(\theta_k)$
4: Calculate rewards-to-go $\hat{R}_t$ for each timestep t
5: Compute advantage estimates $\hat{A}_t$ using any standard method with V_{ϕ_k}
6: Compute the policy gradient $\hat{g}_k$ as follows:

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|}\sum_{\tau\in\mathcal{D}_k}\sum_{t=0}^{T}\nabla_\theta \log\pi(a_t|s_t,\theta_k)\cdot\hat{A}_t$$

7: Update policy parameters via gradient ascent

$$\theta_{k+1} = \theta_k + \alpha_k\hat{g}_k$$

8: Update value parameters by regression on mean-squared error:

$$\phi_{k+1} = \text{argmin}_\phi \frac{1}{|\mathcal{D}_k|T}\sum_{\tau\in\mathcal{D}_k}\sum_{t=0}^{T}(V_\phi(s_t) - G_t)^2$$

where T represents the time horizon of each trajectory
9: **end for**

REINFORCE

REINFORCE, or Monte Carlo Policy Gradient, directly adjusts the parameters of the policy based on the gradient of the expected reward, using entire episodes for

updates [47]. It often suffers from high variance in its estimates, and its performance can be significantly influenced by the choice of learning rate α and discount factor γ.

Algorithm 4 REINFORCE

Input: a policy $\pi_\theta(a|s)$ parameterized by θ
Initialize policy parameters θ arbitrarily
for each episode **do**
    Generate an episode trajectory $\{s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$
    for $t = 0$ to $T - 1$ **do**

$$G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k,$$

$$\theta \leftarrow \theta + \alpha \gamma^t G \nabla_\theta \ln \pi(a_t|s_t, \theta)$$

    end for
end for

Trust Region Policy Optimization (TRPO)

TRPO is an on-policy, policy gradient algorithm that improves vanilla policy gradient methods by constraining the policy update. It updates policies by taking the largest step possible to improve performance while satisfying a distance constraint between updated and current policies. The constraint is expressed in terms of Kullback-Leibler (KL)-Divergence, a measure of distance between probability distributions. TRPO avoids collapsing policy performance and tends to monotonically improve performance.

TRPO modifies the standard policy gradient objective The modified objective with a trust region constraint is given by:

$$\begin{aligned} \max_{\theta} \quad & \mathbb{E}_{\pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A^{\pi_{\theta_{\text{old}}}}(s,a) \right] \\ \text{s.t.} \quad & \mathbb{E}_{\pi_{\theta_{\text{old}}}} \left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s) \parallel \pi_\theta(\cdot|s)) \right] \leq \delta \end{aligned}$$

where D_{KL} denotes the KL-divergence and δ is a small positive number that limits the size of the policy update, ensuring the new policy is within a 'trust region' of the current policy.

2.4.3 Actor-Critic Method

DDPG (Deep Deterministic Policy Gradient)

DDPG is an off-policy algorithm that integrates deterministic policy gradients and Q-learning. DDPG can only be used for environments with continuous action and is considered an extension of Q-learning for continuous action spaces. It uses off-policy data from a replay buffer and the Bellman equation to learn a Q value function [48]. DDPG employs a deterministic policy that is optimized using gradient ascent methods. This makes it possible to handle the optimization challenges that arise in continuous spaces, where finding the maximum Q-value is non-trivial. To facilitate exploration in its deterministic policy framework, DDPG adds noise to the actions during training, such as time-correlated Ornstein–Uhlenbeck (OU) noise or mean-zero Gaussian noise.

Algorithm 5 Deep Deterministic Policy Gradient (DDPG)

Input:
Initialize actor network $\mu_\phi(s)$ and critic network $Q_\theta(s,a)$ with weights ϕ and θ
Initialize target networks μ' and Q' with weights $\phi' \leftarrow \phi$, $\theta' \leftarrow \theta$
Initialize replay buffer $\mathcal{D}$
for each episode **do**
  Observe initial state s
  for $t = 0$ to T **do**
    Select action $a = \text{clip}(\mu_\phi(s) + \mathcal{N}, a_{low}, a_{high})$ where $\mathcal{N}$ is noise for exploration
    Execute action a and observe reward r and new state s'
    Store transition (s, a, r, s') in $\mathcal{D}$
    Sample random mini-batch $\mathcal{B}$ of N transitions (s_i, a_i, r_i, s'_i) from $\mathcal{D}$
    Computer the target

$$y = r + \gamma Q'_{\theta'}(s', \mu'_{\phi'}(s'))$$

    Update critic by one step gradient descent:

$$\nabla_\theta \frac{1}{|\mathcal{B}|} \sum_{(s,a,r,s',d) \in \mathcal{B}} (Q_\theta(s,a) - y)^2$$

    Update the actor policy using the one step gradient ascent:

$$\nabla_\phi \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} Q_\theta(s, \mu_\phi(s))$$

    Update the target networks:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta'$$

$$\phi' \leftarrow \tau\phi + (1-\tau)\phi'$$

  end for
end for

Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 is an actor-critic, off-policy algorithm that improves upon the DDPG algorithm by addressing the overestimation bias. TD3 learns clipped double Q-function and regresses toward the minimum of the two Q-values to reduce overestimation. TD3 updates policy similar to DDPG. However, TD3 adds delayed policy updates and target policy smoothing to further stabilize training. The policy is updated less

frequently than the Q-functions to reduce volatility. TD3 also uses "target policy smoothing" by adding clipped noise to the target action coming from the target policy.

Algorithm 6 TD3 Algorithm

Input:
Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters
Initialize target networks $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{D}$
for each episode **do**
  Observe initial state s_0
  for $t = 0$ to T **do**
    Select action with exploration noise $a_t = \pi_\phi(s_t) + \epsilon$, where

$$\epsilon \sim \mathcal{N}(0, \sigma)$$

    Execute action a_t and observe reward r_t and new state s_{t+1}
    Store transition tuple (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
    Sample mini-batch of N transitions $\mathcal{B} = s, a, r, s'$ from $\mathcal{D}$
    compute targets $y = r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \pi_{\phi'}(s'))$
    Update critics $\theta_i \leftarrow \arg\min_{\theta_i} \frac{1}{|\mathcal{B}|} \sum (y - Q_{\theta_i}(s, a))^2$
    if $t \mod d$ **then**
      Update actor by the deterministic policy gradient:

$$\phi_i \leftarrow \nabla_\phi \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} Q_\theta(s, \mu_\phi(s))$$

      Update target networks:

$$\theta_i' \leftarrow \tau\theta_i + (1 - \tau)\theta_i'$$

$$\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$$

    end if
  end for
end for

Advantage Actor-Critic (A2C)

A2C is an extension of the basic Actor-Critic method, introducing the advantage function to reduce variance in policy updates. The advantage function $A(s, a)$ guides the policy gradient more effectively by considering the relative value of each action compared to the average [49].

Algorithm 7 Advantage Actor-Critic (A2C)

Input:
Initialize actor network π_ϕ and critic network V_θ with random parameters
for each episode **do**
  Observe initial state s
  for each step of episode **do**
    Select action $a \sim \pi_\phi(s)$
    Execute action a and observe reward r and new state s'
    Estimate advantage

$$A(s, a) = r + \gamma V_\theta(s') - V_\theta(s)$$

    Update critic by minimizing loss:

$$L(\theta) = A(s, a)^2$$

    Update actor using policy gradient:

$$\nabla_\phi J(\phi) = \nabla_\phi \log \pi_\phi(a|s) A(s, a)$$

    $s \leftarrow s'$
  end for until s is terminal
end for

Asynchronous Advantage Actor-Critic (A3C)

A3C improves upon standard actor-critic methods by training multiple instances of the agent in parallel in multiple environments. This parallelism not only speeds up training but also diversifies the experience, which stabilizes the update gradients [49].

Algorithm 8 Asynchronous Advantage Actor-Critic (A3C)

Input:
A set of continuous MDPs $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ for each actor-learner
Initialize global actor network $\pi_{\phi_{\text{global}}}$ and critic network $V_{\theta_{\text{global}}}$ with random weights
Initialize multiple actor-learner threads each with its own set of network parameters ϕ and θ
for each actor-learner thread **do**
  for each episode **do**
    Observe initial state s
    Tare local gradients: $d\theta \leftarrow 0$ and $d\phi \leftarrow 0$
    for each step of episode **do**
      Select action $a \sim \pi_\phi(s)$
      Execute action a and observe reward r and new state s'
      Accumulate gradient wrt. θ: $d\theta \leftarrow d\theta + \nabla_\theta \log \pi_\phi(a|s)(G - V_\theta(s))$
      Update global critic by $d\theta$
      Accumulate gradient wrt. ϕ: $d\phi \leftarrow d\phi + \nabla_\phi \log \pi_\phi(a|s)A(s,a)$
      Update global actor by $d\phi$
      Synchronize local networks with global networks
      $s \leftarrow s'$
    end for until s is terminal
  end for
end for

Proximal Policy Optimization (PPO)

PPO aims to balance exploration and exploitation by limiting the size of policy updates. PPO which evolved from policy gradient methods like A2C, introduces a novel objective function that penalizes large changes to the policy, thus avoiding the performance collapse. This is achieved by using a clipped surrogate objective function. This clipping mechanism is what primarily differentiates PPO from A2C [50].

Algorithm 9 PPO-Clip

Input: Initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, \ldots$ **do**
Collect a set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by executing a policy $\pi_k = \pi(\theta_k)$
Calculate rewards-to-go $\hat{r}_t$
Calculate advantage estimates $\hat{A}_t$ based on the current value function V_{ϕ_k}
Update the policy:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \hat{A}_t, \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right)$$

Update value function by mean-squared error regression:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{r}_t \right)^2$$

end for

Soft Actor-Critic (SAC)

SAC bridges the gap between stochastic policy optimization and approaches like DDPG. Soft Actor-Critic is an off-policy actor-critic algorithm that incorporates entropy into the reward signal to encourage the policy to explore more by maximizing a trade-off between expected return and entropy. Entropy regularization enhances exploration and prevents premature convergence to suboptimal policies [51]. SAC is designed for continuous action spaces and employs two Q-functions to mitigate overestimation bias. In SAC, the policy network outputs a distribution over actions. Instead of sampling an action directly from this distribution (which would be a non-differentiable operation), the reparameterization trick rewrites the action as a deterministic transformation of some independent noise. An action a is produced by transforming a noise variable ϵ (usually sampled from a standard normal distribution) using the mean and standard deviation output by the policy network. This can be mathematically represented as

$$a = \mu(s) + \sigma(s) \cdot \epsilon,$$

where $\mu(s)$ and $\sigma(s)$ are the mean and standard deviation for the state s, and ϵ is the random noise.

Algorithm 10 Soft Actor-Critic (SAC)

Input: Initialize policy parameters θ, Q-function parameters ϕ_1, ϕ_2, initialize replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\phi_{\text{targ},1} \leftarrow \phi_1, \phi_{\text{targ},2} \leftarrow \phi_2$
repeat
  Observe state s and select action $a \sim \pi_\theta(s)$
  Execute a in the environment
  Observe new state s', reward r, and done signal d
  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
  if update then **then**
    for each update step **do**
      Randomly sample a batch of transitions $\mathcal{B} = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute targets for the Q functions:

$$y = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}), \tilde{a} \sim \pi_\theta(s')$$

      Update Q-functions by one step of gradient descent using:

$$\nabla_{\phi_i} \frac{1}{|\mathcal{B}|} \sum (Q_{\phi_i}(s, a) - y^2$$

      Update policy by one step of gradient ascent using:

$$\nabla_\theta \frac{1}{|\mathcal{B}|} \sum \left(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}(s)) - \alpha \log \pi_\theta(\tilde{a}(s)|s) \right)$$

      Update target networks with:

$$\phi_{\text{targ},i} \leftarrow \tau \phi_{\text{targ},i} + (1 - \tau)\phi_i$$

    end for
  end if
until convergence

2.5 Imitation Learning

2.5.1 Behavioral Cloning

Behavioral Cloning (BC) is the simplest method in imitation learning where a model learns to mimic expert behavior directly and learns a mapping from states to actions [52]. BC is a supervised learning method where policy π is trained to map states to the action of an expert policy π_E. The objective of BC is to minimize the difference between the actions taken by the learned policy and the expert actions, given the same states. The loss function, often a mean squared error, is given by:

$$L(\pi) = \mathbb{E}_{(s,a)\sim\pi_E}\left[(\pi(s) - a)^2\right] \tag{2.34}$$

where (s, a) are state-action pairs sampled from the expert's policy π_E.

Algorithm 11 Behavioral Cloning (BC)

1: **input:** Expert demonstrations $\mathcal{D}$, policy network π_θ
2: **for** each epoch or batch **do**
3: Sample a batch $\{(s_i, a_i)\}$ from expert demonstrations $\mathcal{D}$
4: Select the policy's action distribution parameters $\mu_\theta(s_i), \sigma_\theta(s_i)$ for each state s_i
5: Calculate the loss L for the policy:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} \|a_i - \mu_\theta(s_i)\|^2 + \lambda_1 H(\pi_\theta) + \lambda_2\|\theta\|^2$$

where $H(\pi_\theta)$ is the entropy of the Gaussian distribution defined by $\mu_\theta(s_i)$ and $\sigma_\theta(s_i)$, λ_1 and λ_2 are regularization coefficients
6: Update the policy parameters θ using gradient descent:

$$\theta \leftarrow \theta - \alpha\nabla_\theta L(\theta)$$

where α is the learning rate
7: **end for**

2.5.2 Generative Adversarial Imitation Learning

Generative Adversarial Imitation Learning (GAIL) combines ideas from Generative Adversarial Networks (GANs) and imitation learning. GANs operate through two

neural networks: generator G and a discriminator D, engaged in a zero-sum game framework [53]. The generator fabricates data resembling real instances, while the discriminator evaluates these instances, discerning real from generated data. The objective function of a GAN is formalized as:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))] \tag{2.35}$$

In GAIL, the generator is the RL policy π, and the discriminator D differentiates between state-action trajectories from the expert and those generated by the policy. Unlike traditional supervised learning, GAIL does not rely on labeled datasets but learns by emulating expert behaviors in an adversarial setting. The goal of GAIL is to train π so that the discriminator cannot distinguish between its trajectories and those of the expert.

One of the key challenges in imitation learning is the issue of distributional shifts. Distributional shift occurs when the policy encounters states that are not represented in the training data, often leading to accumulating errors. GAIL addresses this by learning entire trajectories rather than individual actions. This approach allows the policy to learn a comprehensive strategy that includes recovery from states that are not part of the expert demonstrations. The policy is trained to minimize the log probability that the discriminator correctly identifies its trajectories, while the discriminator is trained to maximize its accuracy in classifying state-action pairs as originating from either the expert or the policy. The GAIL objective is given by:

$$\min_\pi \max_D \mathbb{E}_\pi[\log(1 - D(s, a))] + \mathbb{E}_{\pi_E}[\log D(s, a)] \tag{2.36}$$

Algorithm 12 Generative Adversarial Imitation Learning (GAIL)

1: **input** Expert demonstrations $\mathcal{D}$, initial policy parameters θ, initial discriminator parameters w
2: **while** not converged **do**
3: Sample trajectories τ using policy π_θ
4: Update discriminator parameters w by ascending its stochastic gradient:

$$\nabla_w \left(E_{\tau \sim \pi_\theta} \left[\log \left(1 - D_w \left(s, a \right) \right) \right] + E_{(s,a) \sim \mathbb{D}} \left[\log D_w \left(s, a \right) \right] \right)$$

5: Update policy parameters θ by descending its stochastic gradient:

$$\nabla_\theta \left(\mathbb{E}_{\tau \sim \pi_\theta} [\log \pi_\theta(a|s) Q_w(s,a)] \right)$$

6: **end while**

Chapter 3

Digital Twin of an Industrial-Scale Bitumen Extraction Process

This work was presented at the 2023 INFORMS Annual Meeting and has been submitted to Computers & Chemical Engineering journal as: J.F. Soesanto, B. Maciszewski, L. Mirmontazeri, S. Romero, M. Michonski, A. Milne, and B. Huang "Digital Twin and Control of an Industrial-Scale Bitumen Extraction Process" (Manuscript ID: CACE-D-24-00011)

3.1 Introduction

In recent years, the "digital twin" has achieved a significant advancement in the process industry [54–57]. Defined as a digital replica of a physical system [58–60], the digital twin heralds a new era of digital transformation in the Oil & Gas sector[61–63]. It facilitates innovative simulation-driven strategies, particularly harnessing the power of artificial intelligence applications [64–66]. The digital models not only enable bidirectional data flow between the virtual and real worlds but also ensure a safe and economical exchange of information [67–69]. Such interconnectivity allows hypotheses formulated in the virtual domain to be tested and refined in the physical realm, and vice versa [70–72].

This potential of digital twin becomes particularly relevant in industries facing complex challenges, such as the oil sands sector. With oil sands reserves estimated to be 2.7 times larger than conventional crude deposits and spread across vast regions globally, they constitute an enormous potential resource [73–75]. Unlike conventional crude oil, which undergoes direct refining, oil sands require multiple processes to produce synthetic crude oil (SCO) [76, 77]. A critical unit in surface mining bitumen extraction is the Primary Separation Vessel (PSV) [78]. It is the first process unit absorbing uncertainty from upstream mining ore grade (the main source of process uncertainty) and plant capacity, ensuring over 90% bitumen recovery. Prior research explored separation mechanisms in pilot-scale PSV both theoretically and experimentally [79]. Our work refines these models and scales them up to better reflect real-world conditions. We divide the middlings layer of the PSV into two zones: one encountering turbulence from entrance effects, and a more stable upper middlings. This distinction enables modeling the dynamic interface, expected to improve interface velocity estimation accuracy and enhance the dynamic model stability. In contrast, prevalent models assume a static interface for stability [79], a simplification not reflective of actual operations. The detailed modeling of the interface is essential

due to its critical impact on separation efficiency. Moreover, the model incorporates often-overlooked aspects like bulk flow dynamics, froth quality, and bitumen losses, providing a more holistic understanding of extraction performance. Finally, we scale up the model to actual industrial dimensions, tailored to real-world processes.

Despite the crucial role of PSV, it is vital to consider plant-wide bitumen extraction processes collectively to grasp their interdependence and assess holistic operational strategies [80–82]. Our research expands this perspective by constructing a model that integrates both upstream and downstream facets of PSV. Notably, it acknowledges that not all bitumen in the middlings phase is wasted, with a portion being recoverable in the FT cells [83]. By replicating real-world configurations and scales, the model is adept at utilizing genuine datasets for parameterization. This synchronization makes the model a potential digital twin for real-world processes, catalyzing dynamic studies of bitumen extraction. Such studies can shed light on refining operations, controls, and strategic deployments.

Key features of the model:

- PSV model refinement: The model introduces an upper middlings layer, to achieve a steady-state without static interface assumptions, offering a closer representation of real-world processes.

- Extended plant modeling: The model expands to upstream process and downstream flotation units, enabling deeper understanding of recovery performance and process interactions.

- Digital twin implementation: The model is meticulously calibrated, and mirrors real processes. By validations across various scenarios, it reflects both real steady state and dynamics, capturing gains and key dynamic parameters. The digital twin enables studying extraction performance and designing advanced control systems.

- Inclusion of fines and bitumen loss: The model accounts for often overlooked factors such as froth quality and tailings bitumen loss, key parameters aiming to assess and optimize operations.

By navigating these innovative terrains, we aim to advance research in the bitumen extraction field, enabling the adoption of the dynamic model as a platform for simulation, control, and optimization studies. Its structure, characterized by multi-modal, nonlinear, and multi-input multi-output with constraints, offers a secure and sturdy testing ground. This environment is not only ideal for experimenting with advanced control and optimization strategies, such as Model Predictive Control (MPC) and Reinforcement Learning (RL) [84, 85], but also facilitates the advanced soft sensor design [86–88]. Rigorous validation is crucial before these strategies are introduced in real-world settings to address and minimize inherent risks [86, 88].

3.2 Modeling

In this section, we discuss each unit and its physics in the plant-wide setting. The model is scaled up based on the actual process dimension and working capacity and validated using actual process data. The model improvement is conducted on PSV which is the heart of the extraction process. The interconnectivity to the upstream and downstream processes is investigated, including Ore Preparation Plant (OPP), PSV, and Flotation (FT) cells.

3.2.1 Process Description

Currently, 20% of oil sands reserves are accessible via mining techniques. The mined oil sands undergo extraction processes to recover bitumen, which then proceeds to the upgrading stage. The extraction process encompasses ore preparation, extraction in PSV, and secondary separation in FT cells.

In the OPP, oil sands containing approximately 11% bitumen, 84% solids, and 5% water are crushed to break up clumps into loose, uniform particles. The addition

of caustic reduces bitumen surface tension and promotes aeration. The crushed oil sands slurry is then transported via high-velocity hydrotransport lines to PSV. The turbulent pipeline flow promotes oil sands ablation, liberates bitumen, and provides additional time for bitumen attachment to air bubbles, improving the separation of bitumen [89].

In the PSV, the slurry is fed to the middlings layer or source zone in the middle of the vessel. Here, aerated bitumen rises to the froth overflow containing around 60% bitumen, 10% solids, and 30% water. Then, the froth undergoes deaeration and is further processed in the Paraffinated Froth Treatment (PFT) before moving on to the upgrading phase [77]. Heavy particles settle to the bottom tailings layer containing about 0.4% bitumen, 65% solids, and 34% water. The tailings discharge is subsequently thickened prior to treatment.

The middlings layer contains around 4% bitumen, 20% solids, and 76% water. It is discharged to the FT cells for secondary separation. Air injection aerates bitumen particles, recovered in the flotation froth overflow containing approximately 17% bitumen, 12% solids, and 71% water. This recycles back to the PSV feedwell. Heavy non-aerated particles settle to the flotation tailings withdrawal containing about 0.4% bitumen, 25% solids, and 74% water. The interconnectivity between PSV and FT cells defines the main oil sands separation process to produce synthetic crude oil.

3.2.2 Overall Process

Mass Balance

A schematic representation of the overall mass balance, detailing the process flow, is shown in fig. 3.2. The ore feed Q_{ore} mixes with process water Q_{wpw} to form a slurry Q_{sl}. The superbox serves as a mixing chamber for feeds from two separate lines and can be approximated as a fixed-volume mixer. Therefore, the PSV feed flow rate Q_{fd} from the superbox to the PSV is given by:

$$Q_{fd} = Q_{sl} \tag{3.1}$$

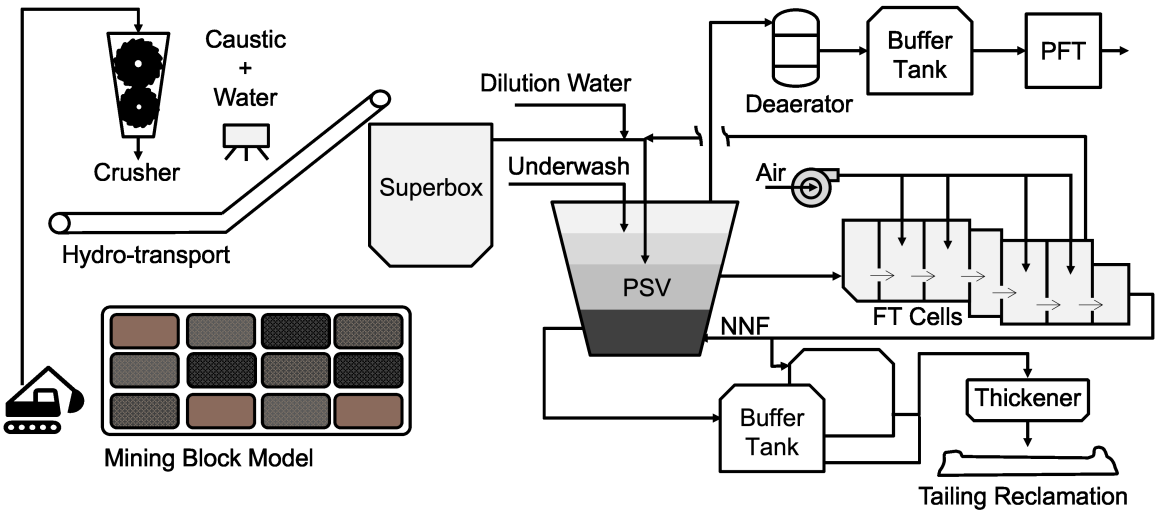


Figure 3.1: Oil sands extraction process schematic

Dilution water Q_{dil} is added to Q_{fd} before entering the PSV. Underwash water Q_{uw} affects separation performance in the PSV by "cleaning" the bitumen droplets as they rise to the top of the vessel and encourages the release of any trapped fines. However, this effect has been omitted from this model to simplify energy balance and heat transfer calculations, as it does not significantly affect overall process behavior. Therefore, Q_{uw} will be considered an addition to the dilution water input to PSV.

Within the PSV, light particles discharge through the froth overflow Q_f, while heavy particles discharge through the tailings withdrawal Q_t. The middlings withdrawal Q_m, which primarily consists of water, remaining bitumen, and residual solid particles, are transferred to FT cells for further separation.

In the FT cells, aerated particles recycle back to the PSV through the flotation froth recycle Q_{ff}, while heavy particles discharge through the flotation tailings withdrawal Q_{ft}. The overall volume balance determines the PSV froth overflow Q_f:

$$Q_f = Q_{fd} + Q_{dil} + Q_{uw} + Q_{ff} - Q_m - Q_t \tag{3.2}$$

where Q_{ff} satisfies:

$$Q_{ff} = Q_m - Q_{ft} \tag{3.3}$$

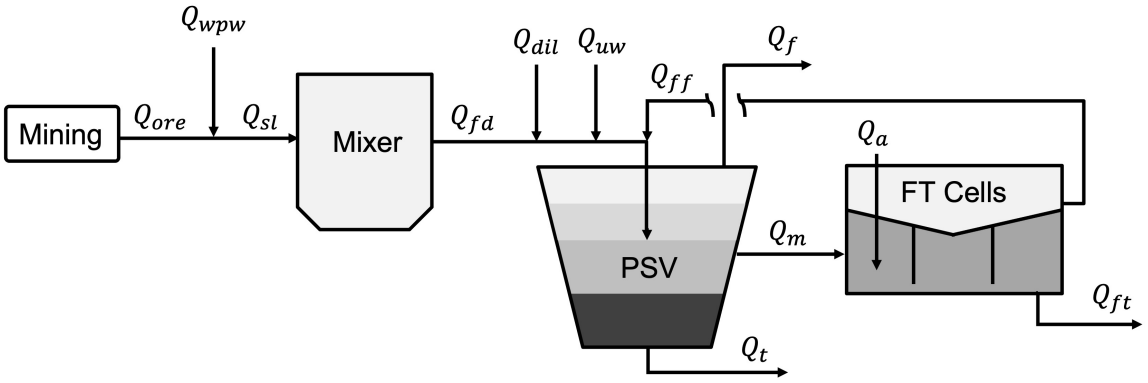


Figure 3.2: Overall mass balance of oil sands extraction process

Dynamic Recovery Rate

To accurately evaluate the separation performance during dynamic operations, we introduce a time-dependent recovery rate calculation. This approach is grounded in the overall mass balances of bitumen that enters and is lost through the tailings. The key aspect is that the bitumen accumulating within the loop over the time period is considered part of the recovered portion. This is because it has separated from the tailings even though it has not yet overflowed. By including accumulation in the recovered bitumen, we avoid ambiguities from changing interface levels and unsteady operation. It provides a meaningful performance metric under dynamic operating conditions.

The recovery rate over a given period is quantified through the following equations:

$$RR_{tot} = \frac{\sum Q_{fd}\alpha_b^{fd} - \lambda_b^t Q_t \alpha_b^t - Q_{ft}\alpha_b^{ft}}{\sum Q_{fd}\alpha_b^{fd}} \tag{3.4}$$

$$RR_{psv} = \frac{\sum Q_{fd}\alpha_b^{fd} + Q_{ff}\alpha_b^{ff} - \lambda_b^t Q_t \alpha_b^t}{\sum Q_{fd}\alpha_b^{fd} + Q_{ff}\alpha_b^{ff}} \tag{3.5}$$

$$RR_{ft} = \frac{\sum \lambda_b^m Q_m \alpha_b^m - Q_{ft}\alpha_b^{ft}}{\sum \lambda_b^m Q_m \alpha_b^m} \tag{3.6}$$

where RR_{tot}, RR_{psv}, and RR_{ft} represent the overall, PSV, and FT recoveries, respectively.

3.2.3 Upstream (Mining/OPP/Hydrotransport)

The variability in ore grade from mining directly influences the slurry mixture produced in the ore preparation plant. The slurry composition can be calculated as:

$$\alpha_j^{sl} = \frac{\alpha_j^{ore}}{1+r} \tag{3.7}$$

where r is the water-to-ore ratio, being set at 0.45 in this work.

The model considers three distinct ore grades to simulate performance under various conditions. The ore recipes are adapted from [79] but tailored to real-world mining statistics. Further classification based on different unaerated bitumen sizes is unnecessary, as the actual distribution in the real process is not well understood and it adds to model complexity [90]. We select one size for unaerated bitumen based on typical ranges. Unaerated bitumen is important to model as it causes bitumen losses in tailings.

Bitumen liberation and aeration are complex phenomena influenced by a multitude of factors including ore grade, fines concentration, temperature, pH, and agitation. While there is a need for further research to model these intricate variables, our study takes a different approach to capture their effects. We vary the composition of unaerated bitumen as a proxy for these influences. It is worth noting that factors like pH, temperature, and agitation are typically controlled within a certain range during the process; however, the presence of fines, particularly in the form of clay, is an unavoidable disturbance. Hydrophilic fines cause slime-coating on the bitumen surface, which increases its hydrophilicity, preventing it from attaching to air bubbles. Meanwhile, hydrophobic fines attach to the bitumen, reducing the froth quality [91–93]. The effect of fines on bitumen aeration is illustrated visually in fig. 3.3.

Therefore, the model is not arbitrary in its treatment of unaerated bitumen variation. Instead, we employ a simple correlation to estimate the fraction of unaerated bitumen in various bitumen grades, linearly increasing with fines concentration

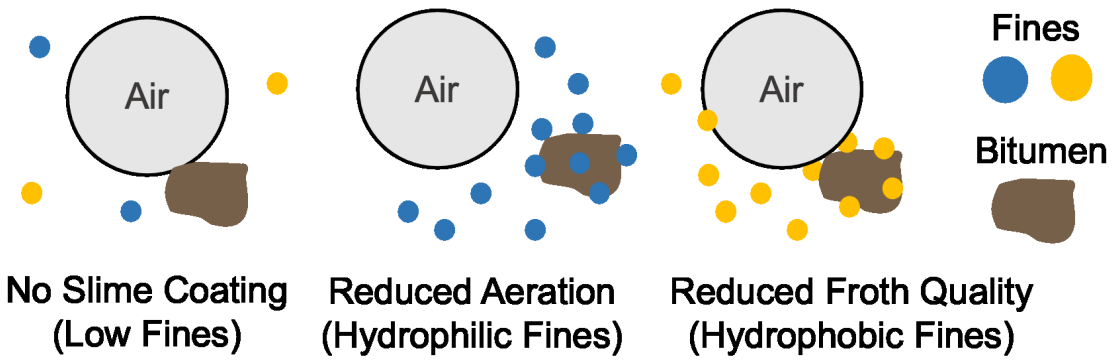


Figure 3.3: Effect of fines on bitumen aeration

$$\frac{\alpha_{b1}^{ore}}{\alpha_{b}^{ore}} = C_1 \alpha_{s1}^{ore} \tag{3.8}$$

The multistep processes from the OPP to the PSV represent the delay associated with the ore grade transition effect on the PSV. The volume of the modeled mixer has been set to replicate the delay observed in the actual process, as reflected in its impact on residence time. This particularly concerns the delayed responses of both the interface and the cone density following a step change in the fines fraction from the OPP. The volumetric fraction of species j in the PSV feed stream α_j^{fd} evolves according to:

$$\frac{d\alpha_j^{fd}}{dt} = \frac{Q_{sl}}{V_{\text{mix}}} \left(\alpha_j^{sl} - \alpha_j^{fd} \right) \tag{3.9}$$

3.2.4 Primary Separation Vessel

Constitutive Relationships

We refer to the modeling principles from [79, 94] and references wherein. The vessel dimensions follow the actual size of cylindrical top and conical bottom. The main novelty is considering an upper middlings above the middlings layer. [95, 96] studied the turbulent flow around the feedwell in the middlings. Consequently, we represent the middlings as a mixing region supplying particles through upper and lower inter-

Table 3.1: Ore recipe

Grade	Low	Average	High
Total Bitumen	10.4%	11.8%	12.9%
Aerated	9.1%	10.9%	12.3%
Unaerated	1.3%	0.9%	0.6%
Total Solids	83.6%	82.5%	81.3%
small	20.9%	15.0%	10.0%
medium	2.7%	3.0%	4.8%
large	60.0%	64.6%	66.5%
Water	5.8%	6.0%	5.8%

faces, with the withdrawn middlings being directed to FT cells. In contrast, the upper middlings exhibit a developed flow profile. In this region, particle settling aligns with Stokes' law. Thus, we model four vertical layers: froth, upper middlings, middlings, and tailings (fig. 3.4).

Throughout the entire vessel, one-dimensional flow is assumed. Particle flow is directionally constrained, moving downward across the middlings/tailings interface and upward across the middlings/upper middlings interface. Bidirectional flow occurs at the froth/upper middlings interface, to account for dynamic interface.

While the interface between the upper middlings and froth layer is dynamic, other interfaces remain static. These static interfaces serve mainly for modeling convenience rather than acting as actual physical boundaries [94]. Their position does not impact steady-state behavior but can affect model dynamics. As such, the volume of each layer is included as empirical parameters subject to adjustment. Details regarding this will be discussed in section 3.3.5.

The dynamic interface between upper middlings and froth layer has positions ranging from 0 at the bottom to 100 at the top of the cylindrical part, with downward velocity deemed positive. The froth/upper middlings interface velocity v_I follows the

Wallis shockwave equations [97]:

$$\text{Wave Velocity} = \frac{\text{Flux Received} - \text{Flux Removed}}{\text{Volume Fraction Difference}} \tag{3.10}$$

$$v_I = \frac{\sum \alpha_{bk}^{m} v_{bk}^{mu} - \sum \alpha_{bk}^{mu} v_{bk}^{f}}{\sum \alpha_{bk}^{mu} - \sum \alpha_{bk}^{f}} \tag{3.11}$$

Here, $k \in \{1, 2, 3\}$ is used as an index for particle size. For solids, k denotes fines, medium, and coarse particles. This index differentiates between unaerated and aerated particles. Specifically, $k = 1$ is used for unaerated particles, as they are generally smaller, while $k = 2$ denotes aerated particles. v_{bk}^{mu} and v_{bk}^{f} denote the bitumen settling velocity in the upper middlings and froth layer, respectively, while α_{bk}^{mu} and α_{bk}^{f} represent the bitumen volume fractions in these layers.

Accurate interface level estimation depends on precisely defining concentration and velocity near the interface. Introducing the upper middlings enhances these predictions. Unlike the highly turbulent middlings, the upper middlings layer has a developed flow profile suited for description by Stokes' equation. This enhances model prediction and provides flexibility, as parameters related to the upper middlings substantially influence interface dynamics, as discussed in section 3.3.5. This diverges from prior models that applied Stokes' equation to the entire turbulent region. Furthermore, the numerator of the shockwave equation aligns with the mass balance of the upper middlings. This alignment allows the interface velocity to naturally reach a steady state as the mass balance achieves equilibrium.

The terminal settling velocities within the multi-particle system are determined by combining Stokes' law with the Richardson-Zaki hindered settling function [98–100]:

$$v_j^i = \left(\frac{g d_j^2 (\rho_j - \rho_i)}{18 \mu_f}\right) \left(\frac{1 - \sum K_j \alpha_j^i}{{\alpha_w^i}^2}\right)^{n_j^i} + v_w \tag{3.12}$$

where v_j^i denotes the settling velocity of species j within layer i. The sign convention is such that the downward direction is positive. Due to the highly turbulent conditions,

particles within the middlings layer are assigned zero vertical velocity. d_j, ρ_j, and K_j represent the diameter, density, and hydrodynamic volume factor of species j, respectively, which are treated as constants. K_j takes on a value of unity for large particles and is greater than 1 for fine particles exhibiting a higher hindered effect. Physical parameters include the gravitational constant g and the dynamic viscosity of water μ. The effective density ρ_i for layer i is calculated as the weighted summation of the species densities:

$$\rho_i = \sum \rho_j \alpha_j^i \tag{3.13}$$

The bulk flow velocity v_w^i in layer i is iteratively solved to ensure overall volumetric continuity [94]. Bulk flows allow the model to accommodate high-density fines in the froth layer, as the upward bulk flow counters the inherent downward settling. Likewise, unaerated bitumen can be modeled in the tailings layer due to the effect of downward bulk flow. Fines particle in froth layer is considered for evaluating the froth quality, while unaerated bitumen is considered for calculating bitumen loss from tailings. The model assumes no aerated bitumen in the tailings and negligible sand particles in the froth layer, as their settling velocities dominate over bulk flow effects. Bulk flow velocity is calculated as follows:

$$v_w^f = \frac{-1}{\alpha_w^f A_{cyl}} \left(Q_f + A_{cyl} \sum \alpha_j^{mu} v_j^f \right) \tag{3.14}$$

$$v_w^{mu} = \frac{-1}{\alpha_w^{mu} A_{cyl}} \left(Q_f + A_{cyl} \sum \alpha_j^{m} v_j^{mu} \right) \tag{3.15}$$

$$v_w^t = \frac{1}{\alpha_w^t A_{con}} \left(Q_t - A_{con} \sum \alpha_j^{mu} v_j \right) \tag{3.16}$$

Then material balances for each layer are formulated considering various components: inlet and outlet flow rates, particle exchanges between layers, and the dynamics at the interfaces. The following equation captures the essence of the accumulation term:

$$\text{accumulation} = \text{inflow} - \lambda_j^i \text{outflow} \tag{3.17}$$

Particle discharge correction factors λ_j^i account for differences between particle

concentrations within the layers and their corresponding outlet streams. The water volume fractions α_w^i are expressed algebraically in terms of the species fractions, eliminating redundant water balance equations and leaving only the independent particle balances. To maintain volume continuity at the outlets, an additional equation is employed:

$$\alpha_w^i + \sum(\lambda_{bk}^i \alpha_{bk}^i + \lambda_{sk}^i \alpha_{sk}^i) = 1 \tag{3.18}$$

The mass fraction of the outlet streams is validated against actual process data, as the sampling locations correspond to these exit points. The specific mass balances for each layer will be elaborated upon in the subsequent subsections.

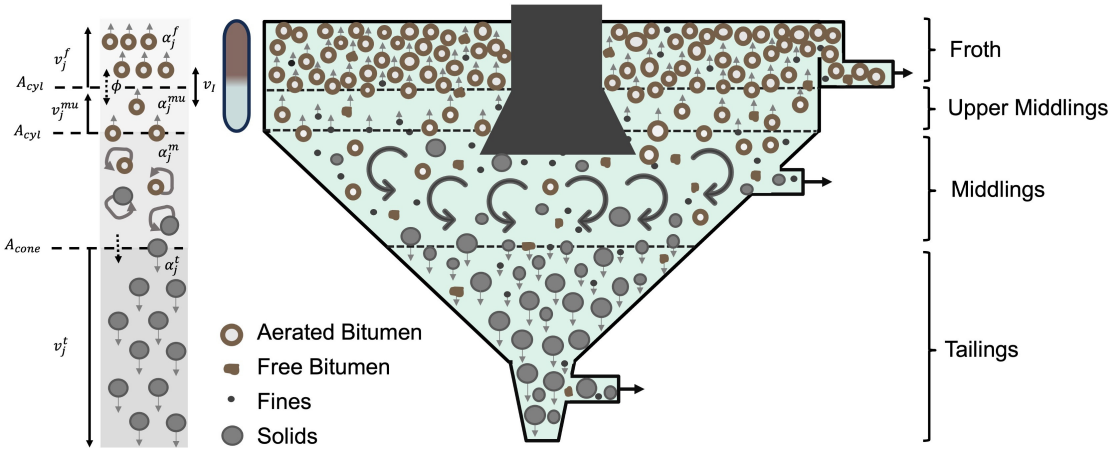


Figure 3.4: PSV schematic

Froth Layer

The interface between the froth and upper middlings is in the cylindrical part of the PSV. We use A_{cyl} to model the froth volume change as follows:

$$\frac{dV_f}{dt} = v_I A_{cyl} \tag{3.19}$$

.

The froth layer has two main flows: the outflow at the top and the inflow from the upper middlings. The mass balance for particle j in the froth layer is given by:

$$\frac{d\alpha_j^f}{dt} = \frac{1}{V_f}\left(\phi_j - \lambda_j^f Q_f \alpha_j^f - v_I A_{cyl} \alpha_j^f\right) \tag{3.20}$$

where ϕ_j is defined as:

$$\phi_j = \begin{cases} \alpha_j^{mu} A_{cyl}(v_I - v_j^{mu}) & \text{if } v_I > v_j^{mu} \\ \alpha_j^{f} A_{cyl}(v_I - v_j^{mu}) & \text{if } v_I \leq v_j^{mu} \end{cases}$$

Here, ϕ represents the particle flux from the upper middlings to the froth layer. If $v_I > v_j^{mu}$, the flux goes from upper middlings to froth; otherwise, it goes in the opposite direction.

Middlings Layer

The middlings layer is subdivided into an upper middlings layer, characterized by well-developed vertical flow, and a middlings which serves as the source zone in the PSV and exhibits high turbulence due to entrance effects. The interface between these layers is considered static and solely for modeling convenience. The area of this interface is denoted by A_{cyl} as it occurs at the tangent line of the PSV. Similar to the froth layer, the volume of the upper middlings is dependent solely on the movement of the froth/middlings interface, but in the opposite direction.

$$\frac{dV_{mu}}{dt} = -v_I A_{cyl} \tag{3.21}$$

The volume balance for the upper middlings is influenced by the particle fluxes at its interfaces with the froth and middlings.

$$\frac{d\alpha_j^{mu}}{dt} = \frac{1}{V_{mu}} \left(-\phi_j - \alpha_j^{m} A_{cyl} v_j^{mu} + \alpha_j^{mu} A_{cyl} v_I \right) \tag{3.22}$$

The middlings layer serves as a mixing region, exhibiting no vertical particle settling owing to turbulent flow conditions. It receives material from both the feed and flotation recycles, and directs it to the FT cells through middlings withdrawal. Additionally, there is an outflux to both the upper middlings and tailings layers. The interface area with the tailings layer, denoted as A_{cone}, is a function of the middlings volume due to its conical shape.

$$\frac{d\alpha_j^{m}}{dt} = \frac{1}{V_m} \left(Q_{fd} \alpha_j^{fd} + Q_{fd} \alpha_j^{ff} - \lambda_j^{m} Q_m \alpha_j^{m} + \alpha_j^{m} A_{cyl} v_j^{mu} - \alpha_j^{m} A_{cone} v_j^{t} \right) \tag{3.23}$$

Tailings Layer

In actual operations, middlings displacement can be recycled from the flotation tailings to the PSV tailings, minimizing dilution water usage. However, in the real-world process, this is typically a no flow (NNF) condition. Given the lack of comprehensive data and its limited impact on capturing the overarching process behavior, this flow has been excluded from the model. The material balance for the tailings layer is governed by the following equation:

$$\frac{d\alpha_j^t}{dt} = \frac{1}{V_t}\left(\alpha_j^m A_{cone} v_j^t - \lambda_j^t Q_t \alpha_j^t\right) \tag{3.24}$$

3.2.5 Downstream (Flotation Cells)

The FT cells modeling is based on the approach developed by [101]. The model consists of two phases – flotation froth and tailings. Assumptions include perfect mixing in each cell and constant attachment efficiency. The FT cells model uses a lumped parameter approach, with a single cell representing the key dynamics of the multi-cell flotation process.

While the actual process consists of 4 cells in series, modeling each cell individually would add complexity without significantly improving model accuracy for the intended application. Therefore, the model equations and parameters are tuned to match overall flotation performance across the multiple cells. Equivalent cell dimensions (volume, area) and overall flowrate, which affect the residence time, are used in the model to represent the combined characteristics of the full circuit. Volumetric balance equations are written for each phase, tracking the volume fractions of air and

mineral components, respectively.

$$\frac{d\alpha_a^{ft}}{dt} = \frac{1}{V_{ft}}(Q_a - J_{TFa}) \tag{3.25}$$

$$\frac{d\alpha_a^{ff}}{dt} = \frac{1}{V_{ff}}(J_{TFa} - Q_{ff}\alpha_a^{ff} - Q_a^l) \tag{3.26}$$

$$\frac{d\alpha_j^{ft}}{dt} = \frac{1}{V_{ft}}(\lambda_j^m Q_m \alpha_j^m - Q_{ft}\alpha_j^{ft} - J_{TFj}) \tag{3.27}$$

$$\frac{d\alpha_j^{ff}}{dt} = \frac{1}{V_{ff}}(J_{TFj} - Q_{ff}\alpha_j^{ff}) \tag{3.28}$$

where α_a^{ff} and α_a^{ft} are air volume fractions in flotation froth and tailings, respectively. α_j^{ff} and α_j^{ft} are mineral j volume fractions in flotation froth and tailings, respectively. V_{ff} and V_{ft} represent equivalent flotation froth and tailings volumes, respectively. Q_a represents the total air injection rate, whereas Q_a^ldenotes the total air loss. J_{TFa} is air volume interphase flux, while J_{TFj} is mineral j volume interphase flux from flotation tailings to froth.

The key modeling aspect is the interphase transfers, which represent the flotation process of mineral particles attaching to air bubbles and transferring from the flotation tailings to the froth. A visual representation of the flotation mechanism is provided in fig. 3.5. The mineral interphase flux J_{TFj} is determined by modeling the bubble dynamics and calculating the mineral volume that attaches during the bubble residence time in the flotation tailings. The attached mineral volume per bubble depends on the mineral j diameter d_j, mineral j volume fraction in the flotation tailings α_j^{ft}, and an attachment efficiency parameter a_j that accounts for surface chemistry. The total number of bubbles transferring mineral is calculated from the tailings air content α_a^{ft}, flotation tailings volume V_{ft}, and air bubble size d_a. Combining these factors yields the modeled mineral interphase flux:

$$J_{TFj} = a_j \frac{6 d_j \alpha_a^{ft} \alpha_j^{ft} V_{ft}}{d_a} \tag{3.29}$$

The air interphase flux J_{TFa} is modeled similarly to the mineral interphase fluxes. The air flux is calculated by considering the local air content α_a^{ft} and the rise velocity

of bubbles v_b through the tailings based on the bubble size. This air flux between phases represents the fundamental flotation transport process. The final equation for air interphase flux is:

$$J_{TFa} = v_b A_{ft} \alpha_a^{ft} \tag{3.30}$$

where A_{ft} is an equivalent FT cells cross-sectional area and v_b is the bubble terminal velocity that is calculated by assuming a perfectly spherical bubble $c_d = 0.47$.

$$v_b = \sqrt{\frac{g d_a}{3 c_d}} \tag{3.31}$$

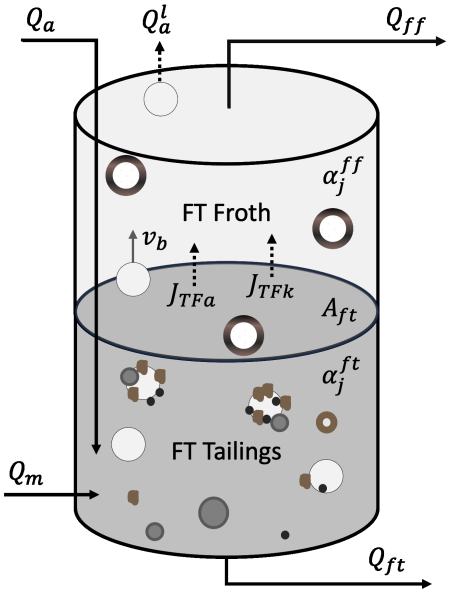


Figure 3.5: FT cells schematic

3.3 Simulation and Validation

3.3.1 Parameterization Based on Steady-State Simulation

The industry standard for steady-state output is outlined in table 3.2. We employed optimization techniques to identify the optimal parameter set, denoted as θ^*. Specifically, we aimed to find the parameters that minimize the deviation between the model steady-state outputs and the criteria outlined in table 3.2. Table 3.3 presents the

baseline parameter set for the model, which is further adjusted based on sensitivity analysis to improve modeling accuracy.

Table 3.2: Typical density and mass fraction across different layers

Layer	Mass Fraction (%)			Density (kg/m^3)
	Bitumen	Water	Solid	
		PSV		
Froth	61	27	12	800-950
Middlings	4	76	21	1050-1120
Tailings	0.4	39	60	1550-1650
		FT		
Froth	18	76	6	924-928
Tailings	0.4	74	25	1170-1243

Table 3.3: Summary of baseline parameters

d_{b1}	d_{b2}	d_{s1}	d_{s2}	d_{s3}	n_b^f	n_s^f	n_b^{mu}	n_s^{mu}
15	240	7	130	360	6.2	7.1	4.5	6.4
n_b^t	n_s^t	$\lambda_{b1/2}^f$	λ_{s1}^f	$\lambda_{b1/2}^m$	λ_{s1}^m	$\lambda_{s2/3}^m$	λ_{b1}^t	λ_{s1}^t
6.5	5	1.06	1.1	0.92	1.05	0.8	1.09	0.9
$\lambda_{s2/3}^t$	k_f	ρ_{b2}	a_{b1}	a_{b2}	a_{s1}	a_{s2}		
1.07	1.31	700	0.1	0.1	0.03	0.3		

3.3.2 Steady-State Sensitivity Analysis and Parameter Fine-Tuning

To understand the model sensitivity to parameter changes, we perturbed each parameter by $\pm 2\%$ and $\pm 5\%$ from its baseline value $\bar{\theta}$. The sensitivity analysis revealed a consistent monotonic trend across different perturbation levels. Therefore, we only present the heatmap for the 5% perturbation (fig. 3.6) to represent parameter sensitivity. It provides insight into the separation mechanisms and guides fine-tuning of

the model parameters. We narrow the discussion to particle distributions, as changes in density are closely tied to the content of solid and bitumen particles.

Increasing d_{b2} reduces bitumen in the middlings and flotation froth as more bitumen is channeled towards the froth layer, decreasing its density since bitumen is a light particle. In contrast, higher hindrance factors, n_b^f and n_b^{mu}, reduce bitumen to the overflow. Consequently, more bitumen is retained within the middlings and directed to the FT cells. It contributes to elevated bitumen fraction in the flotation froth and tailings, ω_b^{ff} and ω_b^{ft}.

Additionally, a reduction in the upward flux of bitumen correlates with a rise in froth solids fraction ω_s^f. This effect is attributed to a compensatory increase in the water flux, thereby escalating the bulk velocity.

Notably, the settling velocities of fine solid particles are substantially influenced by this elevated bulk velocity, overshadowing the impact of their own relative velocities. Consequently, the model output is relatively insensitive to d_{b1}, d_{s1}, n_s^f, n_s^{mu}, and n_b^t, parameters that are intrinsically related to relative velocities.

Higher solids diameters, d_{s2} and d_{s3}, increase the settling velocity of solids towards the tailings. This results in a concomitant decrease of solids in the middlings ω_s^m, flotation froth ω_s^{ff}, and tailings ω_s^{ft}, along with respective reductions in density. Due to the already high concentration of solids and the reduction in the middlings solid fraction, the solids fraction in the tailings ω_s^t is relatively less sensitive. In contrast, increasing n_s^t has the opposite effect.

Generally, an increase in the particle discharge correction factor λ_j^i, which accounts for imperfect mixing within the layer, leads to a proportional rise in the fraction j within the i discharged flows. Elevating λ_b^f increases the bitumen fraction in the froth discharge ω_b^f, accompanied by a corresponding decline in the solids fraction ω_s^f. This also reduces bitumen in the middlings; as more bitumen is discharged from the froth, the concentration and hindrance effects in the froth layer decrease, allowing for greater bitumen flux into this layer. Furthermore, an increment in λ_b^m increased the

bitumen content in the middlings discharge that flows to the FT cell.

Increasing the fines hydrodynamic volume factor K_f intensifies the fines hindrance effects on particles across all layers, leading to elevated concentrations in the source middlings region. The effects are more pronounced for larger particles, whose relative velocities are dominant.

The empirical aerated bitumen density ρ_{b2} has a major influence on the density gradient, the driving force of bitumen settling. An increase in ρ_{b2} reduces the density gradient, resulting in greater retention of aerated bitumen in the middlings ω_b^m. This in turn leads to a decrease in froth density ρ_f and an increase in the concentration of solids within the froth ω_s^f.

Based on this sensitivity analysis, we have fine-tuned the model parameters (table 3.4). Note that when tuning the parameters, we took into account their impact on both steady-state and dynamic behaviors. The influence on dynamic behavior is further discussed in section 3.3.5.

The outcomes at steady-state are tabulated in table 3.5 and table 3.6. Mass fraction calculations are based on the discharge from each layer, after accounting for the imperfect mixing correction factor, since actual process samples are taken from the discharge pipeline. Density, however, is calculated based on the layer, as the sensor is attached to the vessel rather than the discharge line, providing a more accurate comparison. Comparing the fine-tuned model output in table 3.5 and table 3.6 with the industry benchmark data in table 3.2, the model steady-state output aligns well with the real-world process.

Ore grade influences both bitumen distribution in froth and recovery rates. Low ore grade impairs recovery due to the less bitumen and high fines content. Fines particles impair separation in two ways. First, they tend to surround bitumen particles, increasing the unaerated bitumen content, leading to greater bitumen losses to tailings. Second, fines have a higher hindrance effect, impeding the movement of both bitumen and solids, increasing the particle lingering in middlings. This increase

affects FT cell particle distribution. More bitumen goes to FT cells, and FT recovery is reduced as more bitumen loss from flotation tailings.

Plant capacity has a minor impact on recovery. A slight reduction in recovery occurs with increased capacity due to shorter particle residence time. This manifests through increased particle distribution in middlings at higher capacities, subsequently affecting distribution in flotation froth. More bitumen and solids report to flotation froth and tailings with higher plant capacity.

In conclusion, the presented model proficiently replicates the referenced steady-state extraction data across multiple scenarios. Furthermore, it elucidates the variations in process state and recovery observed under these different conditions.

Table 3.4: Summary of fine-tuned parameters

d_{b1}	d_{b2}	d_{s1}	d_{s2}	d_{s3}	n_b^f	n_s^f	n_b^{mu}	n_s^{mu}
21	230	7.5	130	380	5.8	7.3	4	6.4
n_b^t	n_s^t	$\lambda_{b1/2}^f$	λ_{s1}^f	$\lambda_{b1/2}^m$	λ_{s1}^m	$\lambda_{s2/3}^m$	λ_{b1}^t	λ_{s1}^t
6.5	5	1.06	1.09	0.91	1.06	0.75	1.08	0.89
$\lambda_{s2/3}^t$	k_f	ρ_{b2}	a_{b1}	a_{b2}	a_{s1}	a_{s2}		
1.06	1.51	700	0.13	0.13	0.04	0.3		

3.3.3 Dynamic Stability Simulation

The dynamic model stability and robustness are evaluated by introducing step changes in ore grade, transitioning between two extreme cases (low to high-grade ore) while operating at normal plant capacity.

Transient responses to these step changes, shown in fig. 3.7, demonstrate the model intrinsic stability. It asymptotically settles to a steady-state solution, consistent with independent steady-state computations in the previous section. Reverting the inputs back to their initial values further validated the model stability, with the system recovering its original steady-state condition after exhibiting smooth transient behavior.

Table 3.5: Steady-state PSV simulation results across various scenarios

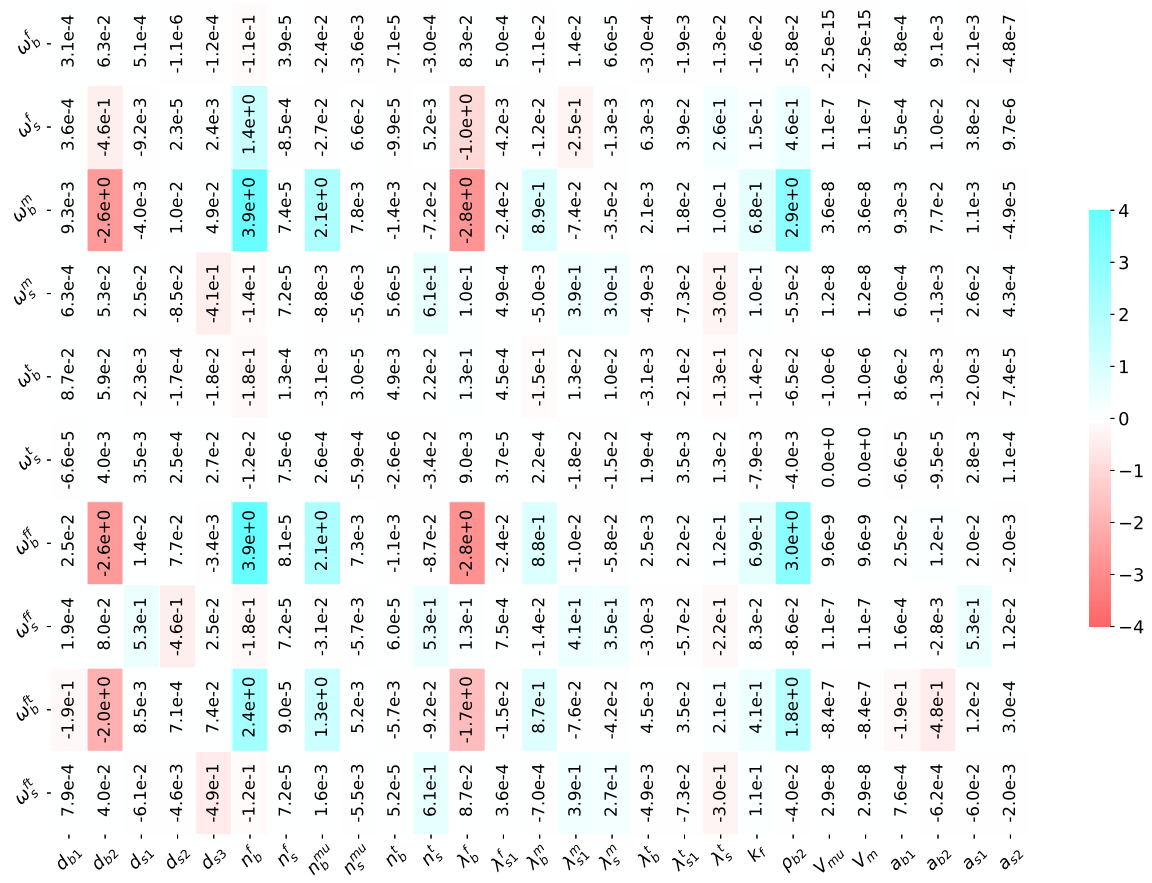
Grade	Capacity	RR_{tot} %	RR_{psv} %	w_{bf} %	w_{sf} %	ρ_f $\frac{kg}{m^3}$	w_{bm} %	w_{sm} %	ρ_m $\frac{kg}{m^3}$	w_{bt} %	w_{st} %	ρ_t $\frac{kg}{m^3}$
High	Turndown	96.4	97.7	60	5	850	5	13	1053	0.3	63	1575
	Normal	96.0	97.9	60	5	853	8	15	1055	0.3	63	1584
	Rated	95.8	97.9	58	5	858	9	15	1057	0.3	63	1574
Avg.	Turndown	94.9	96.4	57	7	869	4	15	1078	0.4	61	1546
	Normal	94.6	96.7	57	7	871	6	16	1076	0.4	61	1545
	Rated	94.2	96.9	58	8	873	8	17	1075	0.4	61	1544
Low	Turndown	93.0	94.7	55	9	886	3	18	1107	0.5	59	1514
	Normal	92.7	95.2	56	9	886	5	18	1102	0.5	58	1501
	Rated	92.3	95.5	56	10	891	6	19	1103	0.5	58	1503

Table 3.6: Steady-state FT cells simulation results across scenarios

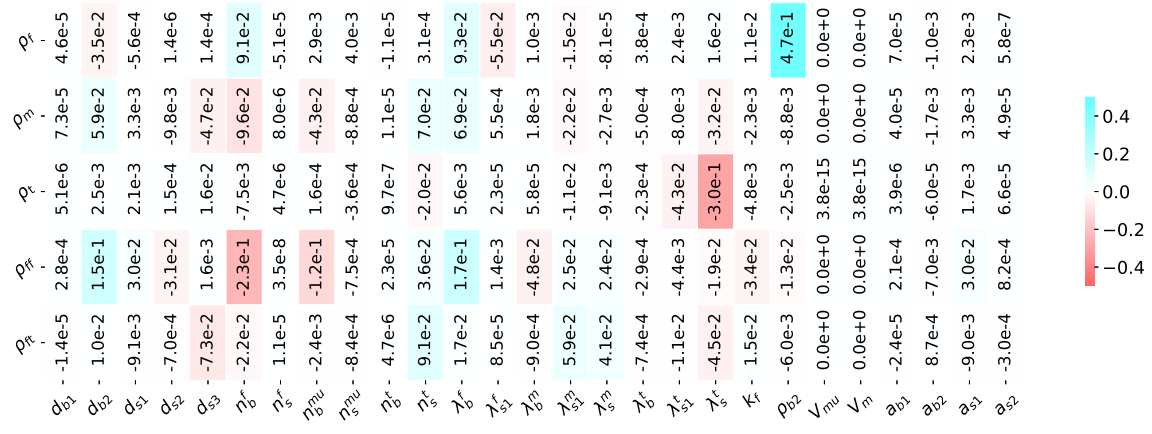
Grade	Capacity	RR_{ft} %	w_b^{ff} %	w_s^{ff} %	ρ_{ff} kg/m^3	w_b^{ft} %	w_s^{ft} %	ρ_{ft} kg/m^3
High	Turndown	95.2	17	11	1007	0.4	14	1069
	Normal	94.5	29	14	990	0.6	15	1075
	Rated	94.2	28	12	980	0.7	17	1090
Avg.	Turndown	94.1	12	11	1019	0.4	18	1097
	Normal	94.1	18	10	996	0.6	19	1111
	Rated	93.8	23	9	976	0.8	21	1122
Low	Turndown	92.8	9	13	1040	0.4	21	1128
	Normal	93.5	11	9	1010	0.6	25	1156
	Rated	93.1	17	10	997	0.7	25	1156

Critically, the interface velocity decayes to zero at steady state, overcoming limitations in prior models that could only achieve "pseudo" steady states with non-zero interface velocities, requiring static interface assumptions. The current model can capture interface dynamics while still reaching real steady-state solutions.

While interface velocity depends solely on inputs, interface level itself does not.



(a) Sensitivity analysis of particle distribution in each layer



(b) Sensitivity analysis of density in each layer

Figure 3.6: Heatmap of sensitivity analysis at 5% perturbation: (a) Particle distribution in each layer; (b) Density in each layer. A red hue signifies negative sensitivity, blue indicates positive sensitivity, and the intensity of the shade corresponds to the magnitude of sensitivity

Initial interface levels may not be recovered when transitioning between steady states, implying interface level is not a state variable determined solely by system inputs.

3.3.4 Dynamic Analysis via FOPD Model Identification

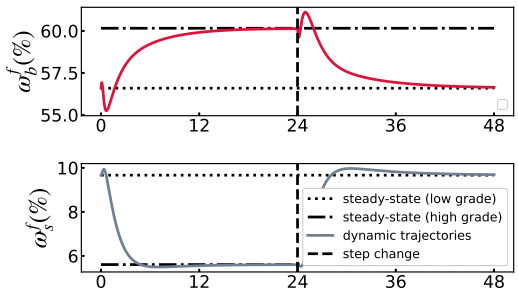
While the model predicts the particle distributions within each layer, it is worth noting that these measurements are not available in real-time operations. In practice, the interface level and tailings density serve as key controlled variables in the primary separation vessel, making the understanding of their dynamics crucial for effective model validation.

An excessively high interface level risks entraining fines and solids from the middlings into the froth, thereby reducing its quality. On the other hand, a low interface level may channel high-bitumen content from the froth layer into the secondary separation cell, thereby increasing its workload and the risk of bitumen loss in the flotation tailings. Maintaining the interface level within a target range of 70-80% optimizes both product quality and recovery, depending on the specific operational conditions.

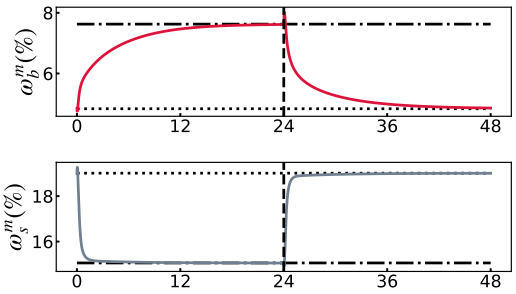
Similarly, the tailings density is an indicator of settling performance, with optimal values ranging between 1580-1650 $\frac{kg}{m^3}$ for improved bitumen recovery. However, exceeding a tailings density of approximately 1750 $\frac{kg}{m^3}$ could pose operational risks, such as sanding in the pipelines and the activation of pressure relief valves.

Given the significance of these control variables, we conducted a comparative analysis of both model-generated and actual step responses for the interface level and tailing density. This analysis took into account the step changes in manipulated variables, Q_m,Q_{dil},Q_t, as well as disturbance from Q_{sl}. The objective is to accurately replicate key variable dynamics.

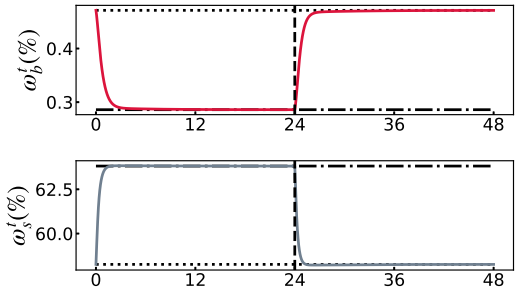
Figure 3.8 presents the model dynamics with respect to middlings withdrawal step change, assuming constant flotation recycling to isolate the effects. Increased middlings withdrawal shifts the interface downward. This shift occurs as higher middlings withdrawal leads to less overflow, resulting in higher bitumen retention in both the



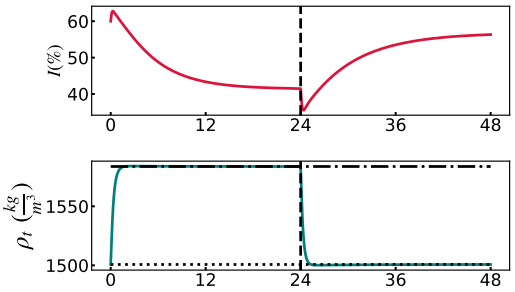
(a) Froth



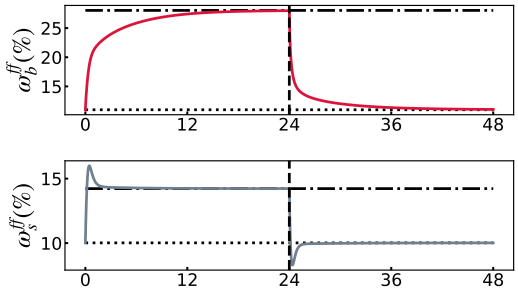
(b) Middlings



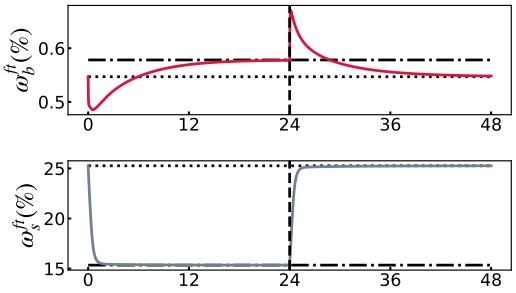
(c) Tailings



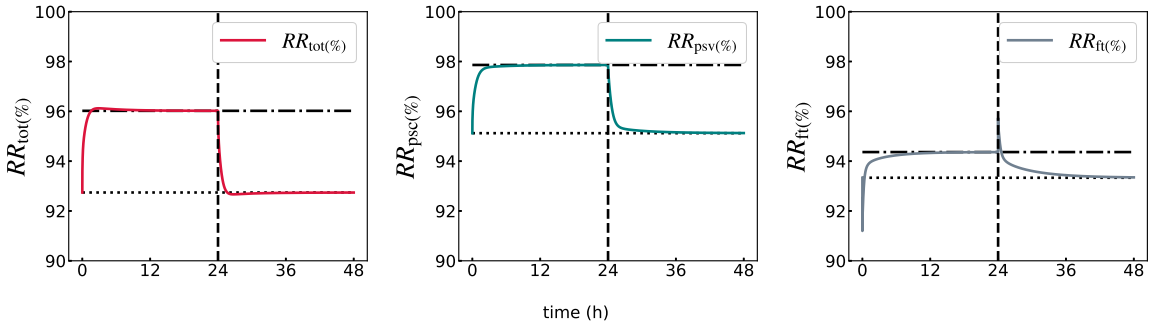
(d) Controlled variables



(e) Flotation froth



(f) Flotation tailings



(g) Recovery rates

Figure 3.7: Dynamic trajectories with step changes in ore grade

froth and middlings. Consequently, the greater concentration of bitumen in the middlings causes the influx of bitumen surpasses its outflux across the interface. The interface then continues its downward movement until the fluxes equalize. The rise in flux can be attributed to the increased bitumen concentration in the middlings. On the other hand, the density shows a minor change, a finding consistent with the actual process where an increase in middlings withdrawal step does not significantly impact the tailing density.

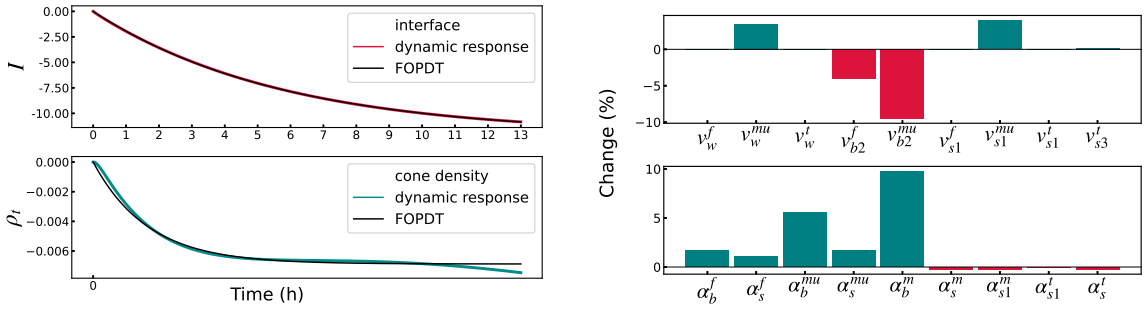
As illustrated in fig. 3.10, tailings withdrawal alters the interface similarly to middlings withdrawal. However, the effect of tailings withdrawal on density reduction is more pronounced, as increasing tailings withdrawal directly reduces the solids in the tailings.

On the other hand, fig. 3.9 shows that a step increase in dilution water leads to an upward shift of the interface. This is a direct result of the reduced bitumen concentration in the middlings, which in turn reduces the influx of bitumen through the interface. Consequently, the inflow of bitumen through the interface becomes less than its outflow, prompting the interface to rise until equilibrium is reached. Additionally, introducing more dilution water reduces the density since the elevated overflow carries more fine particles to the froth, slightly reducing fines in middlings and the tailings.

Figure 3.11 illustrates the effects of increasing the feed flow rate. The interface level rises, similar to the effects of increased dilution water. However, as the feed contains both bitumen and solids, the extra bitumen fed to the middlings counterbalances the greater bitumen overflow. This causes a smaller interface rise compared to the one caused by dilution water. The feed flow increase also adds more solids to the middlings and tailings, raising cone density.

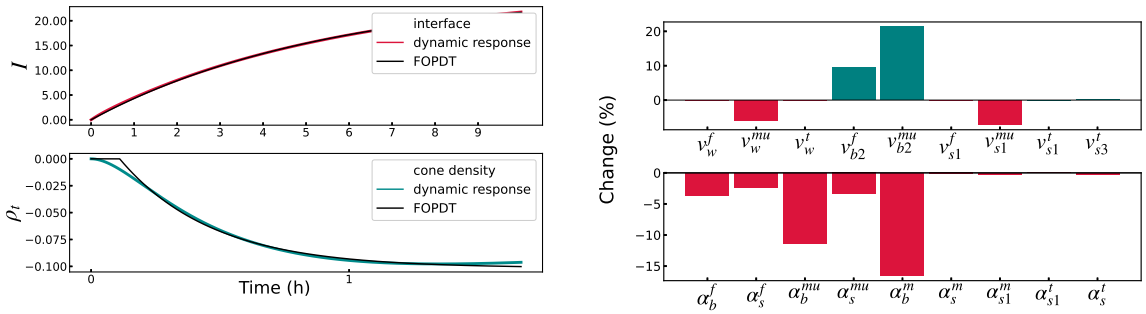
Based on the observed step responses, we identify the FOPD parameters to approximate the model dynamics. To ensure the model fidelity, we compared the derived FOPD parameters from the model step responses to the actual step responses from

the process. While using baseline parameters from section 2.1.1 gives correct dynamics in the sense of direction of change and order of magnitude, achieving higher accuracy necessitates fine-tuning these parameters. We performed a sensitivity analysis on the model parameters to find the optimal fine tuning. The outcomes of the model dynamics from fine-tuned parameters will be discussed in section 3.3.5.



(a) (b)

Figure 3.8: FOPD model identification from step response to middlings withdrawal: (a) Model fit; (b) State changes

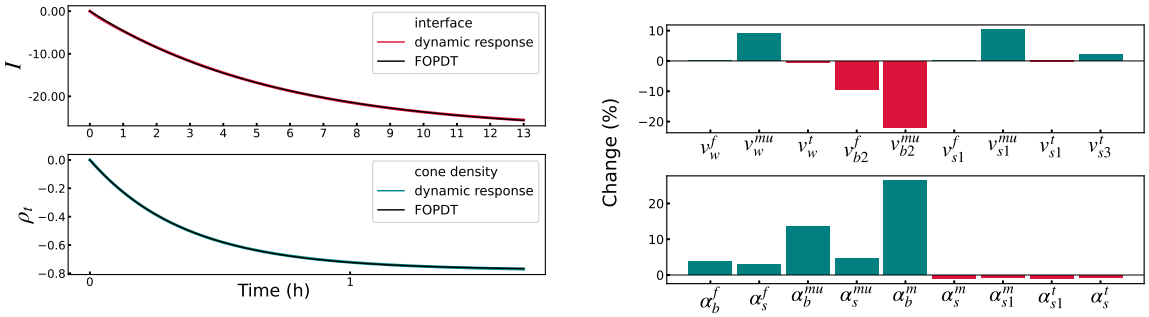


(a)

Figure 3.9: FOPD model identification from step response to dilution water: (a) Model fit; (b) State changes

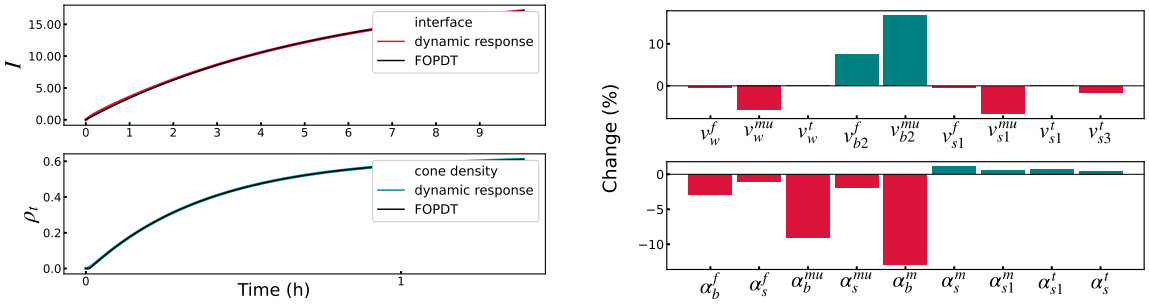
3.3.5 Dynamic Sensitivity Analysis and Parameter Fine Tuning

As depicted in fig. 3.12, the model exhibits greater sensitivity in its dynamics compared to its steady-state behavior. This distinction provides us with more flexibility to adjust parameters for accurate dynamic representation. Given that the model



(a)

Figure 3.10: FOPD model identification from step response to tailings withdrawal: (a) Model fit; (b) State changes



(a) (b)

Figure 3.11: FOPD model identification from step response to feed flow rate: (a) Model fit; (b) State changes

spends most of its operational time in transient states, prioritizing model dynamics accuracy during parameter tuning is paramount. We plotted the sensitivity of density and interface level separately due to their distinct magnitudes of sensitivity.

To begin with, the interface velocity is governed by the net flux within the control volume and the concentration differences across the two distinct layers. Intuitively, as the net flux rises and concentrations between the layers converge, the rate of interface change accelerates. We discovered that the parameters which influence the steady-state distribution of bitumen also significantly impact the dynamics of the interface. Notably, these parameters include d_{b2}, n_b^f, n_b^{mu}, λ_b^f, and ρ_{b2}. As these parameters lead to an increase in bitumen concentration in the upper middlings and a corresponding decrease in the froth, the interface experiences more pronounced shifts, largely due to the diminishing concentration differences between the upper middlings and froth

layers.

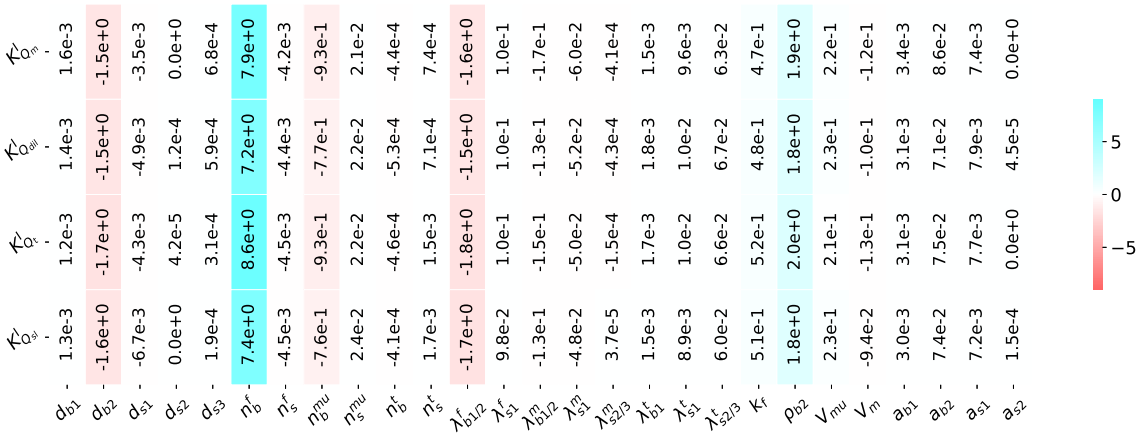
Tailings density dynamics are less susceptible to parameter modifications than interface level dynamics, with their sensitivities varying based on specific inputs. For instance, tailing density dynamics relative to slurry change $K_{Q_{sl}}^{\rho_t}$ are affected by the fines size d_1 and the Richardson-Zaki index parameters n_b^{mu}, n_s^{mu}, and n_b^t. However, when considering changes due to dilution water $K_{Q_{dil}}^{\rho_t}$, the dynamics are influenced by the Richardson-Zaki index n_b^f, n_b^{mu}, and the particle discharge correction factor λ_{s1}^t. Notably, volume changes consistently impact tailing density dynamics. Adjusting the boundaries between the upper middlings & middlings and middlings & tailings empirically impacts the volume of each layer. Given a fixed system volume, an increase in V_{mu} reduces the volume of either the middlings or tailings. This results in shorter particle residence time in the cone, leading to faster density dynamics. Similarly, modeling the conical portion with more middlings than tailings amplifies the tailing density dynamics owing to the shorter residence time in tailings.

In the parameter tuning process, both dynamic and steady-state sensitivities are jointly considered. This comprehensive approach lead to the determination of the final parameters, as presented in table 3.4. Using these parameters, we summarized the process gain for different scenarios in table 3.7. The rate of change is notably distinct across the scenarios.

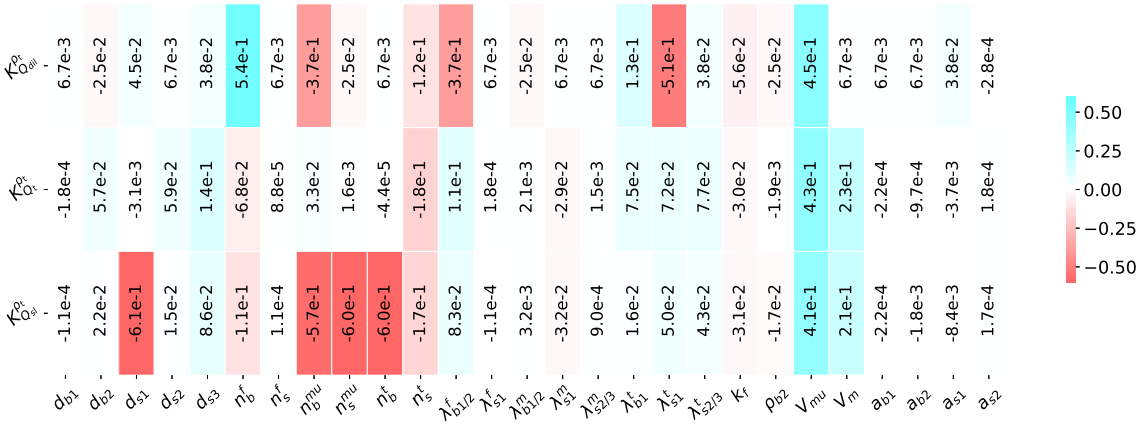
Generally speaking, higher plant capacities lead to faster response rates, with the exception of the tailing density response to the step change in slurry flow rate. The rapid shift in interface level at elevated plant capacities can be attributed to its sensitivity to disruptions in the balance between inflow and outflow across the interface. Given that both values increase with plant capacity, even slight imbalances can induce substantial shifts in the interface level.

The ore grade further complicates this dynamic interplay. Notably, the interface level changes more quickly under high-grade conditions. In contrast, the tailings density exhibits a faster reaction to dilution water under low-grade conditions, and

to the slurry flow rate and tailings withdrawal under high-grade conditions.



(a) Sensitivity analysis of interface level dynamics



(b) Sensitivity analysis of tailings density dynamics

Figure 3.12: Heatmap of sensitivity analysis at 5% perturbation: (a) Interface level dynamics; (b) Tailing density dynamics. A red hue signifies negative sensitivity, blue indicates positive sensitivity, and the intensity of the shade corresponds to the magnitude of sensitivity

Table 3.7: Summary of process gains after fine tuning

Scenario	Grade	Capacity	Gain ($\frac{1}{s}$)						
			$K^I_{Q_m}$	$K^I_{Q_{\text{dil}}}$	$K^I_{Q_t}$	$K^I_{Q_{\text{sl}}}$	$K^{\rho_t}_{Q_{\text{dil}}}$	$K^{\rho_t}_{Q_t}$	$K^{\rho_t}_{Q_{\text{sl}}}$
1	High	Turndown	-0.95	2.07	-2.24	1.64	-0.09	-1.14	0.84
2	High	Normal	-1.19	2.59	-2.93	2.07	-0.09	-1.14	0.81
3	High	Rated	-1.22	2.76	-2.93	2.07	-0.09	-1.10	0.81
4	Average	Turndown	-0.76	1.71	-1.72	1.33	-0.13	-1.02	0.81
5	Average	Normal	-1.02	2.24	-2.41	1.72	-0.14	-1.00	0.79
6	Average	Rated	-1.26	2.76	-3.10	2.24	-0.14	-0.98	0.78
7	Low	Turndown	-0.64	1.43	-1.47	1.14	-0.17	-0.88	0.76
8	Low	Normal	-0.88	1.90	-2.07	1.59	-0.19	-0.86	0.74
9	Low	Rated	-1.02	2.41	-2.59	1.90	-0.19	-0.84	0.72
	Min.		-0.64	1.43	-1.47	1.14	-0.09	-0.84	0.72
	Avg.		-1.00	2.24	-2.41	1.72	-0.14	-1.00	0.78
	Max.		-1.26	2.76	-3.10	2.24	-0.19	-1.21	0.84
	Process		-1.29	2.07	-1.41	1.03	-0.21	-1.24	1.00

Note: Integrating process gains scaled such that the lowest gain from the actual process is 1.

3.4 Conclusions

The digital twin paradigm, applied to bitumen extraction from oil sands, has illuminated advancements in understanding and orchestrating the dynamic complexities of large-scale operations. This research shows a four-layer PSV model aptly mirrors the intrinsic behaviors of real processes. By analyzing the influences of inputs and disturbances on outputs, our methodology provides profound insights into separation mechanisms and the dynamics of key variables - interface and tailings density.

A significant challenge is the intricate nature of the plant-wide model containing a multitude of parameters. To effectively mimic the real process, these parameters demanded adjustments. Leveraging the Tree Parzen Estimator alongside sensitivity analysis proved instrumental in fine-tuning these parameters.

This work introduces a digital twin of the bitumen extraction process, offering significant contributions to the oil sands industry. This technology fosters innovative research into process dynamics while aiding operator training and controller designs or tuning. This tool can predict process behavior, serving as a representative of typical industrial processes laden with disturbances, non-linearities, multi-modal behavior, and multi-input multi-output scenarios. It provides a virtual testing ground for designing and evaluating operational strategies, even for extreme scenarios, thereby minimizing risk and cost before actual fine tuning and implementation. Such advancements champion the digitalization wave in the oil and gas sector.

Future directions involve investigating the effects of interface and density dynamics on bitumen recovery, to determine optimal setpoints. Moreover, by modeling bitumen aeration and integrating aspects like ore psychochemistry and varying operational conditions, we can achieve a more holistic representation of feed characteristics. We believe this will further solidify the role of digital twins in innovating the operational strategies of bitumen extraction.

Chapter 4

Safe Autonomous Control of Primary Separation Vessel using Reinforcement Learning

This work was presented at the 2023 INFORMS Annual Meeting and in preparation to submit as: J.F. Soesanto, B. Maciszewski, L. Mirmontazeri, S. Romero, M. Michonski, A. Milne, and B. Huang "Safe Autonomous Control of Primary Separation Vessel using Reinforcement Learning"

4.1 Introduction

Reinforcement learning-based controller (RLC) has demonstrated significant success in control applications, spanning autonomous vehicle control [12–17, 21, 102], Heating, Ventilation, and Air Conditioning (HVAC) systems [103, 104], industrial process control [22, 84, 105–107], and other control problems [108–111]. The explorative and adaptive power of reinforcement learning (RL) aligns with the Industry 4.0 transformation toward autonomous operations, driven by big data and machine learning. Smart automation is achieved as the agent continues to learn with more data gathered over time.

This research aims to bridge the theoretical advancements in RL with autonomous control applications across various industries, using the bitumen extraction process from the oil sands sector as a primary case study. This digital twin-assisted study ensures the realistic dynamics of industrial-scale processes while representing typical challenges in industrial process control such as Multi-Input Multi-Output (MIMO) systems, multi-mode operations, high-dimensional state-action spaces, nonlinear processes, safety constraints, and random disturbances. Additionally, the controller is benchmarked against MPC to establish a standard for comparison and performance evaluation. Furthermore, to facilitate future integration with Real-Time Optimization (RTO), the problem is simulated with random setpoint changes within its normal range to anticipate dynamic setpoint optimization from the RTO layer.

However, the reliability of AI-based technology remains a practical challenge. Machine learning requires a substantial amount of data. RL samples data through interaction with the environment. During the early stages of training, a "naive" RL agent typically explores its environment randomly. In control settings, this unpredictable behavior can cause disruptions or even shutdowns in the process.

One approach is integrating RL with conventional controllers to preserve the safety integrity of systems while enhancing controller performance. References [112–114] use

RL for dynamic parameters tuning of PID and MPC. While this method improves the controller performance, it limits the exploratory and adaptive strengths inherent to RL. Instead, there is a growing focus on addressing the safety and feasibility of direct RLC deployments in real-world systems.

In safe reinforcement learning, two predominant directions are: (1) reward shaping for risk aversion and (2) modifying the exploration process. CMDPs have been instrumental in shaping Constrained RL (CRL) by incorporating safety constraints into decision-making [115, 116]. However, relying on a reward-based mechanism involves a trade-off between safety and performance. Constrained Policy Optimization (CPO) offers theoretical assurances for near constraint satisfaction [117]. While they guarantee safety constraints during testing [118, 119], ensuring safety constraints during the learning phase remains a challenge. Reference [120] proposes a safe learning approach given a safety model availability.

We study a safe and feasible deployment of RLC using transfer learning and constrained exploration approaches. Transfer learning is a strategy to equip agents with expert knowledge prior to real-world deployment [121]. To this end, we employ two different approaches: imitation learning and offline training. Imitation learning trains RL agents to mimic expert behavior, thereby accelerating the learning process during real-world deployment. Imitation learning technique is particularly useful in tasks like starting and shutdown processes, whose objectives are not easily described, but can be demonstrated by an operator or a working control system.

We explore two imitation learning approaches: Behavioral Cloning (BC) and Generative Adversarial Imitation Learning (GAIL). This study uses closed-loop process data with MPC feedback. MPC serves as the expert controller, which the agent aims to emulate. BC uses a supervised learning approach, training the agent to match its actions to those of the expert based on given state inputs. While BC benefits from learning purely from data, it may encounter issues due to distributional shifts [122, 123]. In contrast, GAIL employs the principles of unsupervised Generative

Adversarial Networks (GANs) to train the agent to produce trajectories that are indistinguishable from the expert trajectories. GAIL is robust against covariate shifts [53, 124].

Offline training in RL is critical due to the high costs and safety concerns during online (stochastic) exploration [125–128]. We use an LSTM-based simulator designed to capture the time-series dynamics of the process. RL agents are pretrained within this simulator before undergoing online training in the digital twin. This simulation-to-reality (Sim2Real) approach, despite potential dynamic mismatches with the actual environment, enables the training of agents with the real rewards, penalties, and constraints of the actual environment. As a result, it is expected to cultivate a risk-averse agent.

While transfer learning avoids erratic behavior, we recognize the imperative need for a safety net to uphold safety integrity. We propose the implementation of MPC-safeguarded exploration to fortify the safety of RL deployment. This method confines the exploration space within predefined safety alarms, assuming optimal states lie within the normal operational range. It balances the need for exploration with process safety, allowing for efficient learning while mitigating potential hazards.

The key contributions of this study are as follows:

- **Digital Twin-Assisted Training:** A digital twin accelerates RLC research and verification prior to real-world deployment. It captures the realistic dynamics of industrial-scale processes and verifies the potential of RLC against common challenges in industrial process control, such as Multi-Input Multi-Output (MIMO) systems, multi-mode operations, high-dimensional state-action spaces, nonlinear processes, safety constraints, and random disturbances.

- **Transfer Learning in RL:** Transfer learning, including imitation learning and Sim2Real, minimizes potential hazards in the training phase. Imitation learning is particularly potential as pretraining for process start-ups and shutdowns.

- **MPC Safeguarded Exploration:** The safety integrity of industrial processes is preserved through the proposed MPC safeguarded exploration. This strategy mitigates potential hazards by restricting the exploration space within safe operational boundaries. Versatile and integrable with various safe RL approaches, this method enhances overall process safety.

This study investigates the potential of RLC for industrial deployment, capitalizing on their superior performance and continual learning abilities suitable for autonomous control applications. Transfer learning and safeguarded exploration are explored to further facilitate the practical and safe implementation of RLC in real-world settings.

4.2 Problem Formulation

Primary Separation Vessel (PSV) extracts over 90% of bitumen from oil sands by separating it from the sand particles. Bitumen particles float to the froth layer while sand particles sediment to the tailing layer. Froth withdrawal extracts bitumen and tailing withdrawal removes sand particles. In the middlings layer, part of the bitumen lingers and withdraws to the flotation (FT) cells to be separated from the remaining sand particles. Flotation froth rich in bitumen is recycled back into the PSV. Sands are removed from flotation tailings. Particle settling is a damped process with no distinct boundaries, allowing sand and bitumen particles to mix. Gravity separation vessels should be maintained in quiescent conditions to maintain laminar particle sedimentation through the viscous medium. Abrupt changes in PSV operation create a chaotic flow that disturbs particle settling. This condition can be observed when there is no clear separation between the froth and middlings interface as they are not strictly immiscible - the medium may mix together resulting in indistinct boundaries between layers. This lack is an indicator of poor separation performance.

Controlling the interface level between froth and middling layers is critical for optimal PSV operation. An excessively high interface facilitates the entrainment of

fines from the middlings into the froth overflow, reducing bitumen quality. Conversely, an overly low interface carries bitumen-rich froth to the FT cell. While the FT cell aerates and refluxes bitumen back to the PSV, its capacity is limited. Excessive bitumen flow pushes the FT cell to its limit, resulting in unaerated bitumen losses through flotation tailings. We constrain the interface level within a normal operating range. The optimal setpoint within this range dynamically adjusts based on the current operating mode.

Tailings density is another controlled variable (CV). Higher densities increase the density gradient between particles and medium, which favors particle separation. However, higher density indicates more particles and solids present, resulting in greater particle hindrance that impedes settling. Optimal density setpoint balances the effects of particle hindrance and density driving force.

The tailings density is constrained as well. Excessively high densities increase pipeline pressure, potentially activating the relief valves. Activation of relief valves requires recalibration and also abruptly disturbs vessel hydrodynamics. In the worst case, high densities increase the risk of pipeline sedimentation and clogging. Therefore, upper limits are imposed on tailings density constraints.

Interface level and tailing density are regulated by Q_m, Q_{dil}, and Q_t, each affecting them in a different direction and magnitude (table 3.7). This poses a Multi-Input Multi-Output (MIMO) control problem. Moreover, PSV is the first equipment in the extraction process, absorbing uncertainties from upstream operations. The innate variability of the ore grade and feed flow rate (dependent on production scheduling) are major disturbances driving changes in the controlled variables, necessitating controllers robust to these fluctuations. Table 3.7 shows how various operating modes, which are characterized by these disturbances, affect the dynamics of each controlled and manipulated variable. This requires a multimodal control strategy on top of the MIMO nature. The combination of MIMO and multimodal dynamics poses limitations on conventional PID control, often needing manual operation to avoid upsets

and requiring frequent operator interventions to maintain performance. As a result, current operations mostly maintain fixed setpoints regardless of the scenario. However, optimal setpoints differ across modes. Advanced control techniques robust to disturbances are thus needed to: (1) stabilize the system under different modes and disturbances and (2) accommodate setpoint optimization in both controlled variables to maximize performance.

The complex MIMO and multimodal dynamics call for advanced process control solutions that can handle changing operating modes while optimizing PSV performance through setpoint adjustments. Recognizing these challenges, we turn to the capabilities of the digital twin to investigate the potential of MPC and RL as part of the autonomous control of the PSV. Figure 4.1 illustrates a control problem of the PSV unit.

In contrast, FT cells, which operate with a mostly fixed air injection, allow for satisfactory control under the PID framework, primarily through manipulating the flotation tailings. Consequently, FT cells control is not integrated into the current autonomous framework. An operational assumption we adopt is that any increment in middlings withdrawal is counterbalanced by a similar increase in flotation tailings, maintaining a steady froth recycle rate.

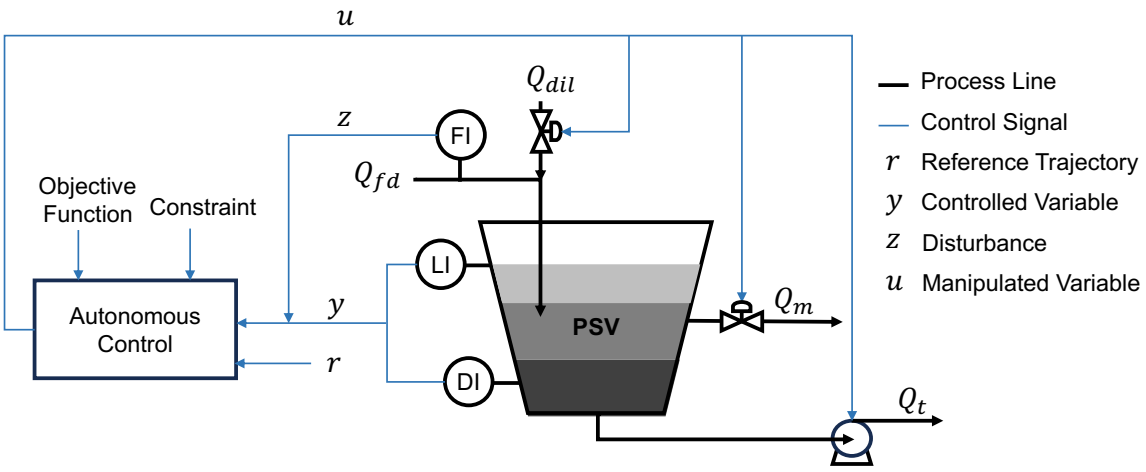


Figure 4.1: PSV control framework

4.3 Model Predictive Control Design

We identify a linear Time-Invariant (LTI) discrete state-space model as a proxy model within MPC. Akaike Information Criterion (AIC) determine model order by striking a balance between complexity and accuracy. MPC uses the model prediction to compute the optimal control actions. Its performance highly depends on the model accuracy. However, plant-model mismatch always happens in dynamics modeling. Process dynamics might change due to aging, changing operating modes, disturbances, and non-stationary processes. Our objective is to engineer an offset-free controller that maintains robust performance despite disturbances encountered in multimode operation. We integrate a disturbance vector $d \in \mathbb{R}^{n_d}$ into the model to capture feed flow rate effects on the system dynamics. Including online feed flow rate measurement anticipates and counteracts real-time feed flow disruption. Ore grade variation is another environmental disturbance that cannot be perturbed manually during system identification experiments. Ore grade is sampled every hour and is not suitable for the MPC as it has a much lower sampling time rate than MPC. Instead, three MPCs are developed using the same objectives and constraints but differ in their underlying models. MPC-1, MPC-2, and MPC-3 use the model identified from low, average, and high ore grade operations, respectively. Automated model scheduling is the objective of our third work (chapter 5). Bias correction estimates the prediction error to correct the model output. These strategies reduce the plant-model mismatch resulting from disturbances and changes in operating modes.

The proxy model is formulated as follows:

$$\begin{bmatrix} x_{t+1} \\ d_{t+1} \end{bmatrix} = \begin{bmatrix} A & B_d \\ 0 & I_d \end{bmatrix} \begin{bmatrix} x_t \\ d_t \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_t, \tag{4.1}$$

$$\hat{y}_t = \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} x_t \\ d_t \end{bmatrix},$$

where I_d is the identity matrix with dimension of 1x1. The augmented matrices B_d and C_d are identified based on the response to disturbances.

Bias correction adjusts the model prediction with the estimated prediction bias:

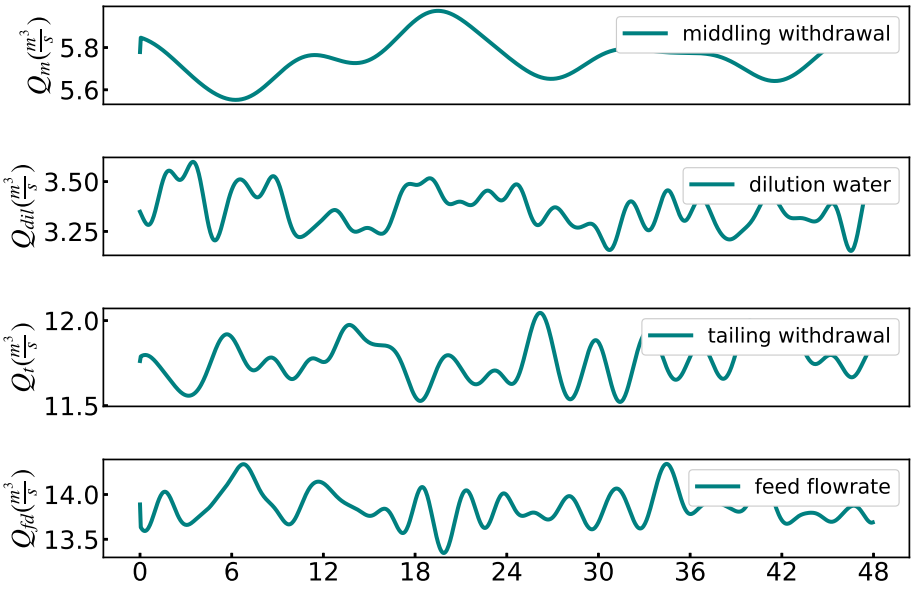
$$\hat{y}_{t+1}^{\text{corrected}} = \hat{y}_{t+1} + b_t, \tag{4.2}$$

where $\hat{y}_t^{\text{corrected}}$ represents the bias-corrected prediction and $\hat{y}_t$ is the model prediction. MPC employs this adjusted prediction to compute optimal trajectories $\hat{y}_{\text{corrected}}$.

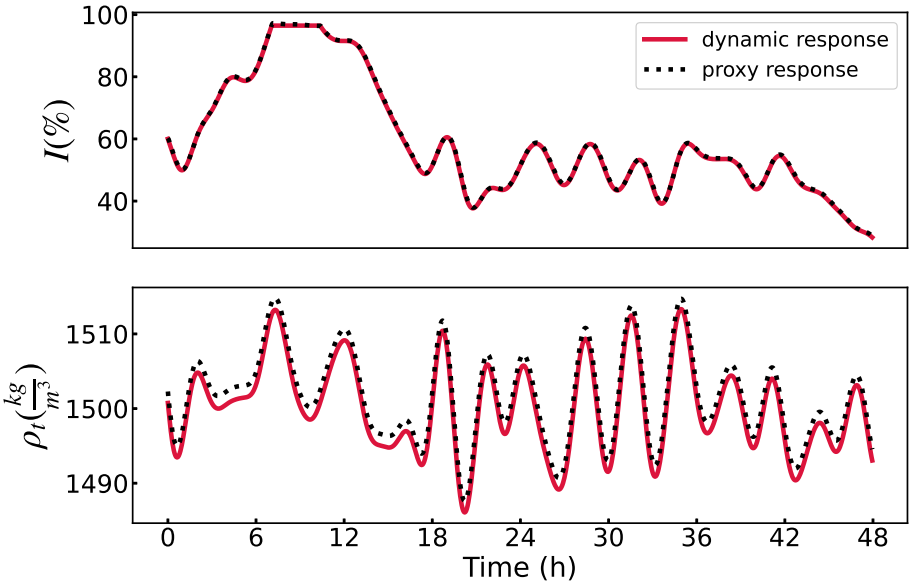
The bias term b_t is calculated using an exponential filter, which smoothens the bias update by accounting for the long-term bias trends from previous steps. The updating rule for the bias is given by:

$$b_t = w_1 \cdot b_{t-1} + w_2 \cdot (y_t - \hat{y}_t), \tag{4.3}$$

where w_1 and w_2 are the weighting coefficients that balance the influence of the previous bias and the current prediction error, respectively. It stabilizes the bias estimation by mitigating short-term variations. The corrected proxy model fitting performance without disturbance is shown in fig. 4.2.



(a)



(b)

Figure 4.2: Proxy model fitting performance: (a) Inputs (scaled); (b) Model outputs comparison

MPC aims to track the setpoint y_{sp} with minimum control effort Δu while satisfying process constraints. Manipulated variables u are constrained within their upper and lower limits. Δu constraint avoids abrupt change in the vessel and overshoots. Mass

balance constraints maintain positive froth overflow. MPC optimization problem is formulated in eq. (4.4).

$$\begin{aligned}
\min_{u_t,\ldots,u_{t+P}} & \sum_{k=1}^{P} \|\hat{y}_{t+k}^{corrected} - y_{t+k,sp}\|_Q^2 + \|u_{t+k} - u_{t+k-1}\|_R^2 \\
\text{s.t.} \quad & x_{t+1} = Ax_t + B_d d_t + Bu_t, \\
& d_{t+1} = d_t, \\
& \hat{y}_t^{\text{corrected}} = Cx_t + C_d d_t + b_{t-1}, \\
& u_{\min} \leq u \leq u_{\max}, \\
& Q_{fd} + Q_{dil} - Q_t - Q_{ft} > 0, \\
& u_{t+k+1} - u_{t+k} - h < 0,
\end{aligned} \tag{4.4}$$

where $y_{t+j,sp}$ and y_{t+j} represent the CV set-point and the predicted output at future time steps. The prediction horizon is denoted by P. Q and R are the weighting matrices that balance the cost of tracking error and controller effort. h represents the limit of Δu.

4.4 Reinforcement Learning-Based Control Design

A control problem can be formulated as an MDP that is described by a tuple of state S, action A, reward R, and transition model. MDP is one special task in reinforcement learning. RL agent (controller) learns the optimal control policy through interactions with the environment (process). States provide the contextual information for making control decisions. Rewarding mechanisms, including penalties, shape the behavior of the learned policy. The action tuple is the changes in manipulated variables:

$$A = [\Delta Q_{dil}, \Delta Q_m, \Delta Q_t]$$

Each action is clipped between -1 to 1, and subsequently scaled to lie within the upper and lower limits of the change in manipulated variables Δu:

$$\Delta u = \frac{(a+1)(\Delta u_{max} - \Delta u_{min})}{2} + \Delta u_{min}$$

Final controller output u is also clipped by the $u_{\max}$ and $u_{\min}$, penalties applied every time an action exceeds this limit to encourage constraint satisfaction.

States are limited to measurable variables in the real process. These states include controlled variables: the interface level and tailing density, along with their respective setpoints and deviations. To capture the current trend of these variables, we also incorporate their Δy to inform their change direction and magnitude. Δy serves as *auxiliary inputs*, summarizing the historical trend of controlled variables [129]. Ore grade and plant capacity determine the operating modes. While plant capacity is measurable, the absence of online ore grade measurement limits the adaptation to various grades. We use the density profile, which is related to the particle composition, to infer the current ore grade. Thus, states include the middling density, froth density, and flow rates to indicate the current operating mode and overall process dynamics:

$$S = [I, I_{sp}, \Delta I, \rho_t, \rho_{t,sp}, \Delta\rho_t, \rho_f, \rho_m, Q_{fd}, Q_{dil}, Q_m, Q_t]$$

The reward consists of controller performance measures and penalties for constraint violations. The controller performance R_{ctrl} includes tracking errors and controller efforts, similar to the MPC cost function. Penalties act as soft constraints.

$$R = R_{ctrl} + p_u + p_{cv}$$

where p_u and p_{cv} denote the manipulated variable and output constraint penalties, respectively. In the MPC safeguarding framework additional penalties apply for every MPC intervention (section 4.8).

Proximal Policy Optimization (PPO) algorithm ensures controller stability during fine-tuning by preventing excessively large updates. The method maintains consistent performance, reducing the risk of sudden or unpredictable behavioral shifts in the controller. We train and validate RLC in a high-fidelity digital twin simulator that captures the real process dynamics and replicates its control challenges. The simulator introduces random setpoint changes every 1 hour on both interface level

and density variables. Continuous Gaussian noise on the feed flow rate is also imposed. Ore grade and plant capacity disturbance are introduced every hour. To ensure the generalizability and reliability of RLC in real-world scenarios, characterized by unpredictable disturbances, we incorporate randomized setpoint changes and disturbances throughout the training and testing phases. RLC is exposed to a wide array of disturbances, closely simulating the uncertainties encountered in actual processes.

4.4.1 Imitation Learning from MPC Demonstrations

Imitation learning focuses on learning tasks by mimicking expert behavior. The learner receives only expert trajectories, cannot request additional data during training, and does not receive any form of reinforcement (reward) signal [130]. Behavioral Cloning (BC) pretrains the policy as a regression task, mapping expert states to their corresponding actions. On the other hand, Generative Adversarial Imitation Learning (GAIL) uses unsupervised learning within the Generative Adversarial Networks (GANs) framework. Agent (acting as a generator) creates trajectories that a discriminator attempts to differentiate from expert trajectories. The generator is trained to deceive the discriminator by producing trajectories that are indistinguishable from those of the expert. BC has the advantage of learning without any direct interaction with the environment, making it efficient for situations where such interaction is impractical or costly. GAIL requires sampling trajectories from the environment. To circumvent the need for real-time environment interaction, Offline GAIL can be employed, utilizing dynamic approximators to simulate trajectory sampling. This approach enables learning without direct environmental interaction.

A critical advantage of GAIL over BC is its robustness to distributional shifts. GAIL is designed to learn complete trajectories that closely emulate the expert behavior, rather than just replicating specific state-action pairs, as is the case with BC. Thus, GAIL generalizes more effectively across environments with varying distributions. In contrast, BC can struggle in such scenarios, often failing to generalize well

to environments with different distributions from those found in the training data.

Imitation learning prior to deployment avoids unsafe trials and errors in the real process. We use MPC as the controller expert. Dynamic simulation in the digital twin generates a dataset that mirrors one year of operational data. The process is controlled by MPC and subject to random disturbances and setpoint variations. Closed-loop data encapsulates states and MPC actions, serving as expert demonstrations. Following the pretraining, we evaluate the RLC performance in the digital twin environment. We continue its training therein to assess its adaptive capabilities during the online tuning.

4.4.2 Simulation-to-Real Pretraining

In addition to expert demonstrations, closed-loop data also captures the dynamics of the process. We use this data to identify a surrogate model with a closed-loop identification approach. We integrate long short-term memory (LSTM) networks into simulators, capturing time-series dependencies in dynamics (fig. 4.3). We investigate the agent ability to generalize its policy under simulator-environment mismatches. Learning from the simulator offers additional benefits compared to solely on expert demonstrations, as the agent learns the actual process constraints and rewarding mechanisms despite mismatches in model dynamics.

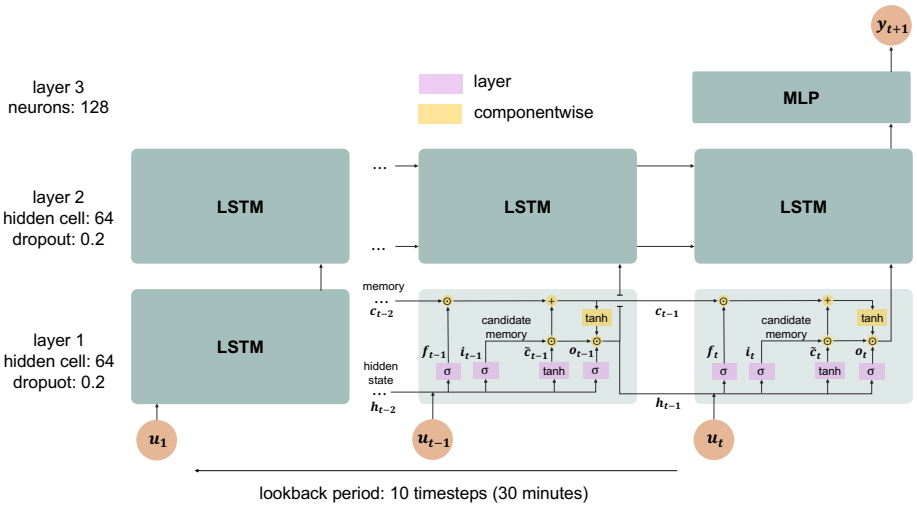


Figure 4.3: Architecture diagram of LSTM-based Simulator

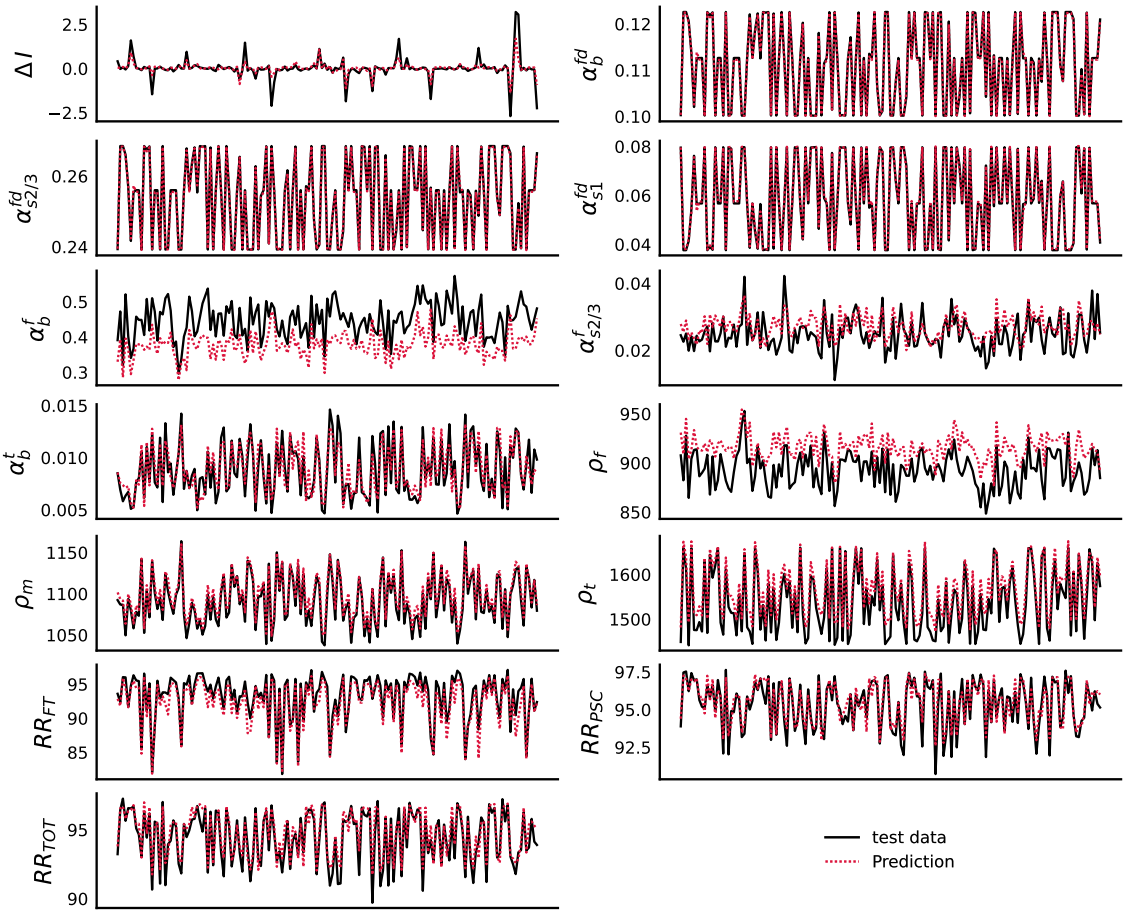


Figure 4.4: Simulator fitting performance

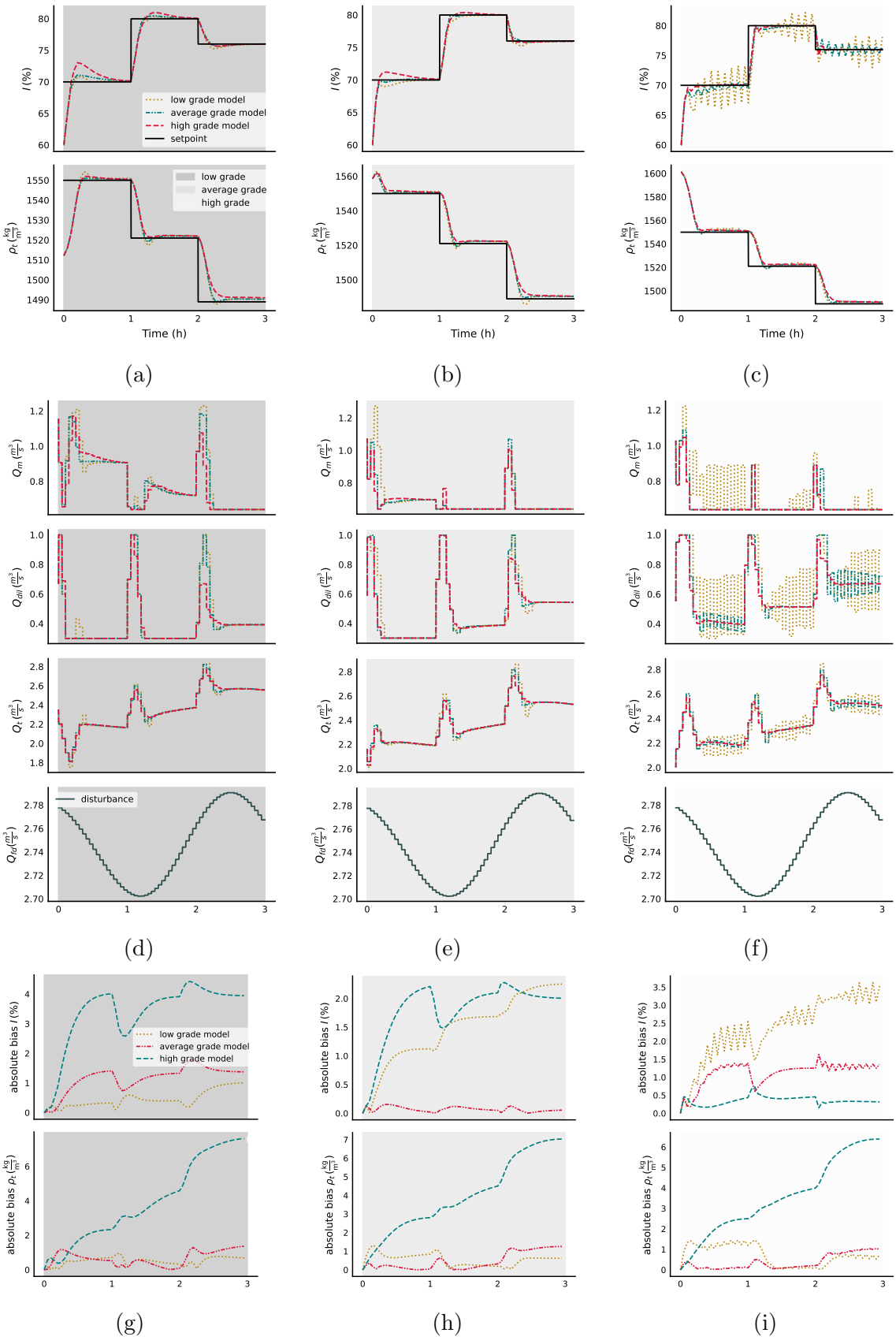
4.5 Model Predictive Control

We test three MPCs under three different ore grade operations, respectively. They are robust to flow disturbances, attributed to the incorporation of feed flow rate disturbance into the MPC model (table 4.1). Bias correction further eliminates offset in both interface level and tailing density control. Figure 4.5g, fig. 4.5h, and fig. 4.5i show the models prediction error over time. The interface prediction bias depends on the ore grade; models identified using data from a specific grade have the lowest prediction bias when operating under that grade. Conversely, the prediction bias of the tailing density is not only influenced by ore grade but also by its current value. Tailing densities are typically lower under low-grade operations because a high content of fines lingers in the middlings layer and settles more slowly into the tailings layer. Here, the tailing density setpoint is progressively decreased to the value of the low-grade operation. Therefore, even under high-grade scenarios, the low-grade model exhibits lower density prediction bias than the high-grade model. Control error measures like Integral Squared Error (ISE) and Integral Time-weighted Squared Error (ITSE) reflect a similar trend (table 4.1). The interface tracking error is lower when the appropriate model is selected based on the ore grade. On the other hand, tailing density control depends on both grade and its current value. The MPC with low and average-grade models exhibits the same tracking errors as the MPC with the high-grade model, even during operations with high ore grades.

Overall, despite plant-model mismatches, the MPCs effectively eliminate steady-state offset (Figure 4.5). However, MPC-1 and MPC-2 cause fluctuation under high-grade operation. Interestingly, no fluctuations are observed when using MPC-3 in either low or average-grade operations. The variations of the process dynamics under different grades explain this fluctuation (table 3.7). Interface level has slower dynamics under low and average grade operation. Models within MPC-1 and MPC-2 capture slower interface dynamics. Thus, MPC-1 and MPC-2 models underestimate

the dynamics under high-grade operations, calculating a more aggressive control action. Table 4.1 shows the controller effort of MPC-3 is always lower than MPC-1 and MPC-2 in various scenarios.

Additionally, bias correction amplifies fluctuations in operating mode with faster dynamics. The current bias is estimated from the previous bias. In environments where dynamics change rapidly, the bias also shifts quickly. Trajectory predictions are continually adjusted with different biases over time. Consequently, the controller, which relies on these predictions, continuously adjusts its flow rate in response to the shifting bias. This frequent adaptation to the latest bias estimations can increase fluctuations, leading to a less stable control. Thus combination of bias correction and MPC model scheduling will eliminate both controller offset and fluctuation.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 4.5: MPC performance

Table 4.1: MPCs performance under different operation modes

Ore Grade	MPC	ISE $I\,10^4$	ITSE $I\,10^4$	ISE $\rho_t\,10^6$	ITSE $\rho_t\,10^6$	IAU Q_m	IAU $\$Q_{dil}$	IAU Q_t
Low	1	5.3	770.0	1.7	424.5	5.1	6.9	5.0
	2	5.3	827.7	1.7	415.8	4.4	6.3	4.1
	3	6.3	1753.4	2.0	808.6	3.7	4.6	3.4
Average	1	4.1	550.4	1.2	319.5	3.1	6.5	4.3
	2	3.8	388.7	1.2	315.7	2.9	6.2	3.6
	3	4.0	603.4	1.4	567.5	3.0	3.9	3.0
High	1	8.3	6165.9	2.4	738.0	8.1	24.8	12.4
	2	3.7	760.2	2.4	684.1	3.3	10.4	5.7
	3	3.1	324.5	2.4	789.8	2.9	4.8	3.7

4.6 Reinforcement Learning-Based Controller Pretraining

We evaluate the performance of RLC right after pretraining with 3 different pretraining methods: BC, GAIL, and Sim2Real. BC pretrains the agent to imitate expert actions given the same states, essentially turning the task into a state-to-action regression. Pretraining solely uses historical data without any interaction with the environment. BC approach failed to pretrain the agent under random setpoint and disturbance changes. The BC pretrained agent showed poor control performance even without disturbances. We suspect it is because the control problem is overly complex. Reference [84] uses a similar approach for single input and single output (SISO) control of PSV. In the study on chemical reactor control using RLC [105], a proxy model along with the BC approach was employed during offline training. However, they do not specify whether the setpoints used during training and testing are identical. We try to simplify the control problem by extending the setpoint change interval to 4 hours and using the sequence of setpoints in the pretrained data for online testing. Figure 4.6 suggests the pretrained agent performs satisfactorily for interface level control and is still acceptable for tailing density control as it is controlled within the constraints. This shows that BC is vulnerable to distributional shift as it only performs well under conditions that are similar to the expert dataset. BC lacks consideration for the sequential nature of decision-making, and suffers from compounding errors in dynamic environments [131, 132]. In dynamic environments, a small error can lead the agent to a state that is significantly different from any state seen during training [133]. Since BC does not learn how to recover from these errors, the agent can quickly drift into unfamiliar state spaces, leading to a cascade of incorrect actions. BC performs poorly in states not well-represented in the training data.

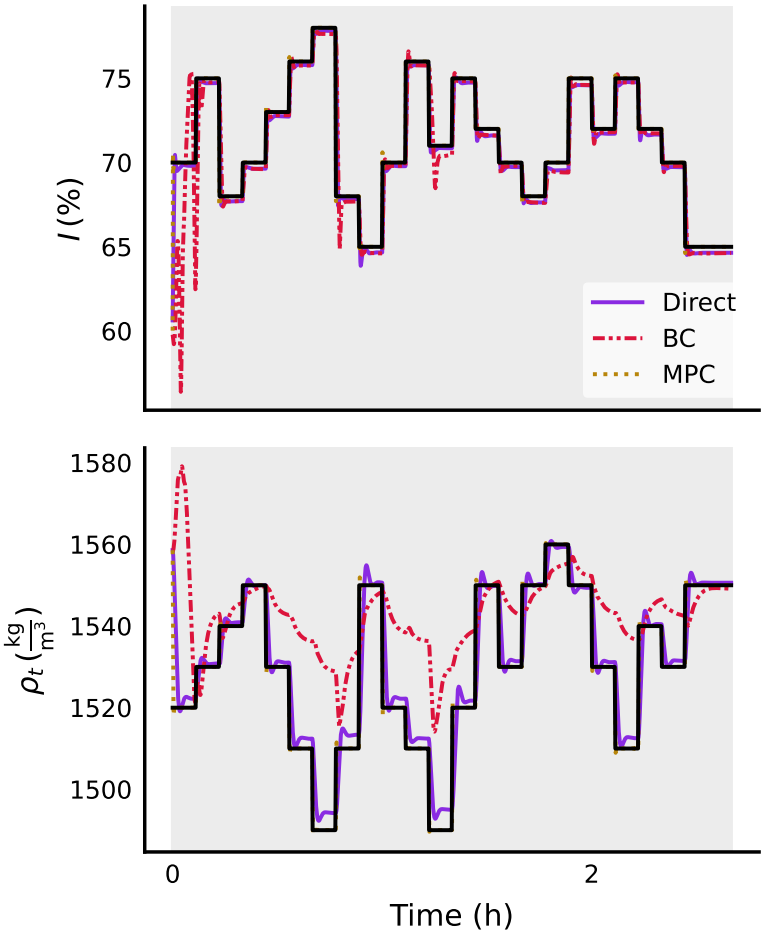


Figure 4.6: BC pretrained agent performance

After BC pretraining, we train the agent in the real environment using the same set of arbitrary setpoints and disturbances. Figure 4.7 shows the agent initially starts with a higher reward value but experiences a drop over time. More critically, it failed to recover from this decline to the convergence level observed in direct agent training. This failure can be attributed to the *loss of plasticity* of Deep Neural Networks (DNNs). Reference [134] demonstrates that deep RL (DRL) agents lose learning ability when they cycle through a series of different tasks. Neural networks tend to lose their ability to learn when they encounter shifts in distributions [135]. In BC pretraining, agents learn to mimic expert actions. This task differs from online training, where the agent aims to maximize the reward function.

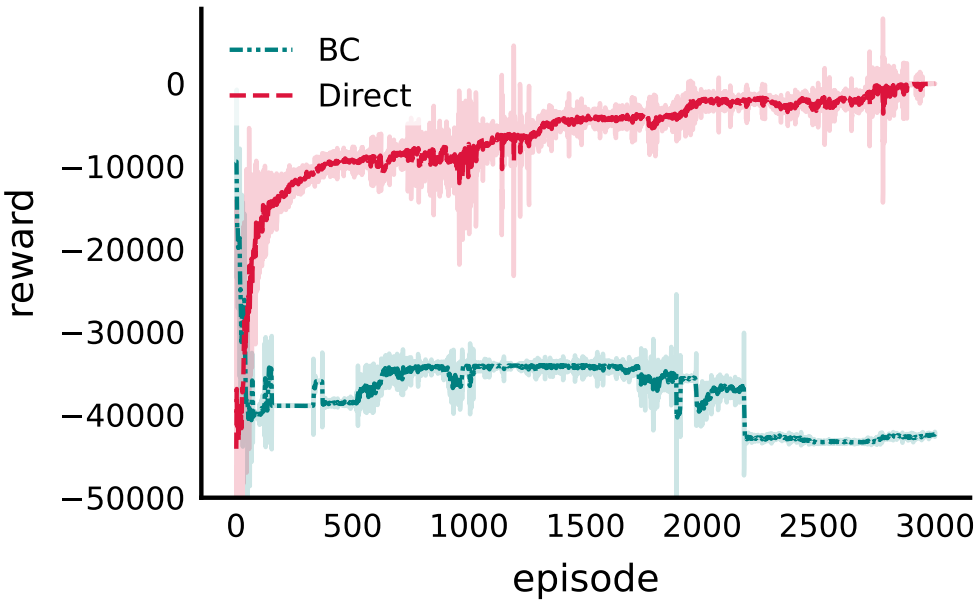


Figure 4.7: BC pretrained agent learning curve during online training

BC pretraining might not be effective in MIMO control problems. Here, controllers regulate setpoint tracking by manipulating multiple variables. The solutions or policies of each controller are not unique; they can vary significantly. Given that we have 3 manipulated variables and 2 controlled variables, the controller has an additional degree of freedom to manipulate the environment. Thus, the RL agent is unlikely to converge to the MPC behavior, which represents just one specific control policy. RL policies are shaped by reward mechanisms and other settings such as state definitions and hyperparameters. Figure 4.8 demonstrates that the MPC and directly trained RLC actions are very different. Therefore, while the expert initially learns from the MPC, it must relearn a new policy. Unfortunately, the loss of plasticity impairs the agent's learning ability. This lack of transferability suggests that BC pretraining might not work for MIMO control tasks. This challenge is exacerbated by the environment with random disturbances such as chemical process [136–138]. BC learning is effective if the expert policy closely resembles the policy that the agent is expected to converge to.

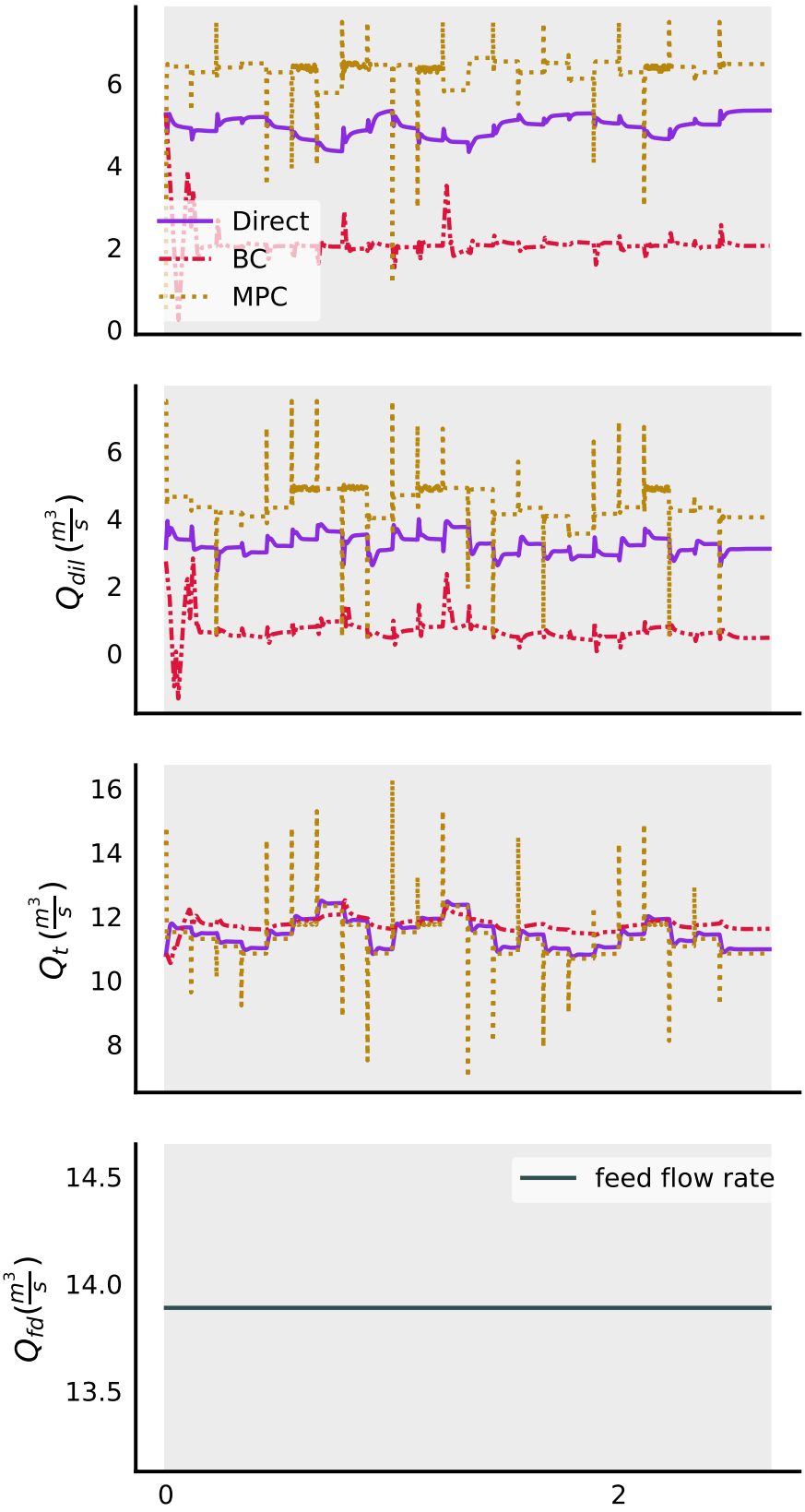


Figure 4.8: RL input trajectories

GAIL-pretrained agent demonstrates satisfactory control even under random setpoint changes and disturbances (fig. 4.9). It regulates the controlled variables to their setpoints and maintains steady states. GAIL focuses on learning entire trajectories through an adversarial process, rather than focusing solely on isolated state-action pairs. This method effectively captures the temporal aspects of expert behavior, crucial in dynamic environments which is driven by the sequence of actions. GAIL counters covariate shifts by training the policy against a discriminator, continuously refining the policy to align with expert trajectories and actions. It pretrains the agent to manage changing environment dynamics and uncertainties. Offline GAIL will further exclude the need for online environment interaction, making this method more practical for industrial settings [139, 140].

However, the GAIL pretrained agent reacts slowly to ore grade disturbances, leading to spikes in the trajectories. This issue arises from the discriminator's difficulty in distinguishing whether sudden jumps result from a setpoint change or an ore grade disturbance. Setpoint changes prompt rapid trajectory adjustments by the MPC to achieve tracking objectives. Consequently, a trajectory spike due to an ore grade change appears normal to the discriminator, as it resembles the expert response to setpoint changes. Despite this, the agent still manages to regulate the controlled variable back to the setpoint. This performance is similar to expert trajectories, which are mostly offset-free.

Agents pretrained in the surrogate model simulator demonstrate satisfactory control performance. It handles changes in grade without overshoot. Unlike GAIL pretraining which imitates expert trajectories, the Sim2Real pretraining objective resembles training in the real environment. The simulator predicts the effects of grade disturbances. The agent learns to minimize tracking errors caused by the grade disturbances. Moreover, the simulator applies identical constraint and penalty mechanisms as the real environment. The penalty deters spikes in trajectories as it could lead to a process trip penalty.

Mismatches between the simulator and the real environment cause controller errors. The policy is a function that maps states to the actions. Bias in state predictions shifts the policy. Thus, simulator fidelity affects the pretraining performance. Despite this, the overall performance of Sim2Real pretrained agents remains robust, even with random setpoint, ore grade changes, and without direct interaction with the real environment.

In summary, BC fails to pretrain the agent under distribution shift in setpoint and ore grade. It works when the training data and testing environment have the same setting and distribution. However, It still fails to adapt to the real environment due to the loss of plasticity. In contrast, GAIL is robust to the distributional shift and able to adapt to the real environment. The GAIL pretrained agent imitates expert behavior without reward feedback from the environment. It is also promising to pretrain tasks such as start-up and shutdown, where defining the objective is not straightforward. However, GAIL agents show less responsiveness to infrequent, sudden grade changes because the adversarial training may not differentiate these changes effectively, given their similarity to the steady-state, expert trajectories. Sim2Real pretraining reproduces the reward and penalty mechanisms of the actual process. It offers a more adaptive approach, capable of handling disturbances and minimizing error across various operating modes.

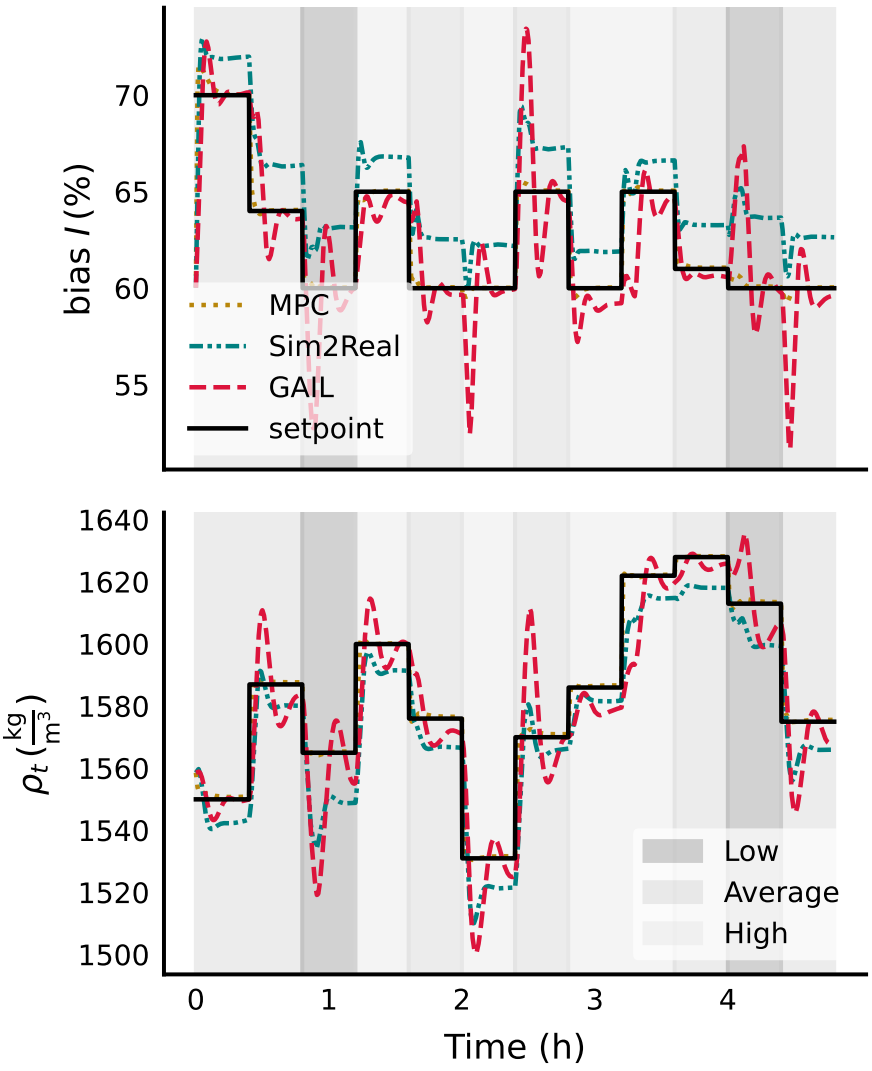


Figure 4.9: RLC performance right after pretraining

4.7 Post Online Training Performance

Adaptive and continual learning abilities are the crucial element in autonomous control. We evaluate RLC performance after online training. The pretrained agents are supposed to adapt and continue their learning in the digital twin environment. To facilitate exploration, we keep stochastic actions during online training. Figure 4.10 shows the pretrained agent avoids overly negative rewards, unlike an agent trained from scratch via RL. Without pretraining, vanilla agents tend to behave erratically at

first, taking a risky trial-and-error interaction with the environment. Highly negative rewards indicate constraint violations that trigger a process trip. Pretraining equips agents with fundamental behavior aligned with the control objectives and constraint satisfaction, reducing unsafe actions during online training. GAIL and Sim2Real pretraining reduces trip count during online training by factors of, 8 and 27 times, respectively, compared to the vanilla agent (table 4.3). Moreover, fig. 4.10 suggests the pretrained agent reward continues to improve during the online training. This highlights RL ability to refine its policy through real-world interactions in dynamic and unpredictable settings.

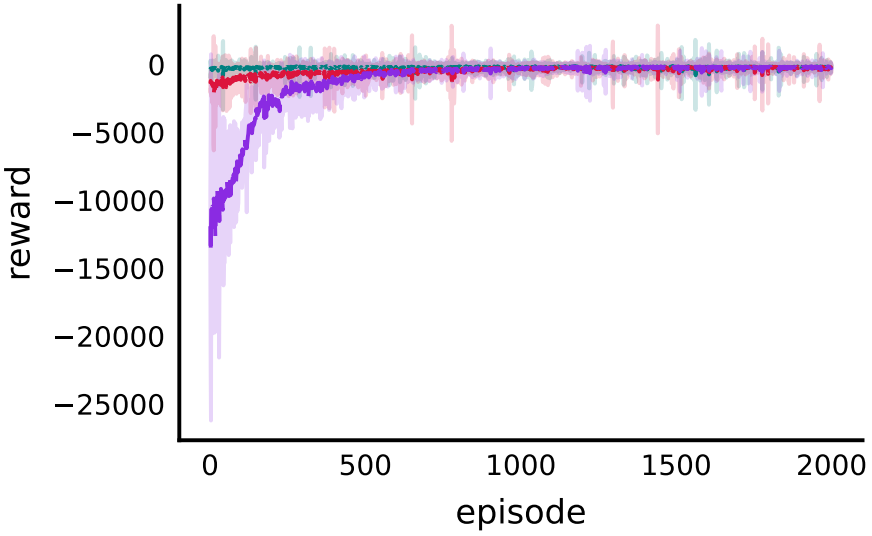
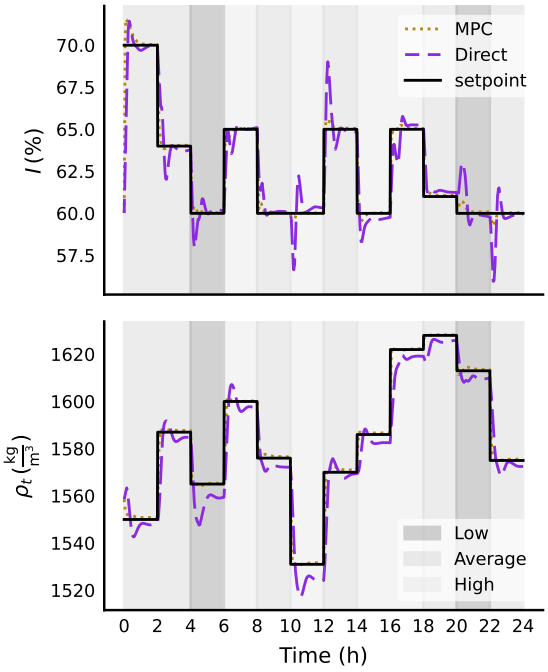


Figure 4.10: Learning curve

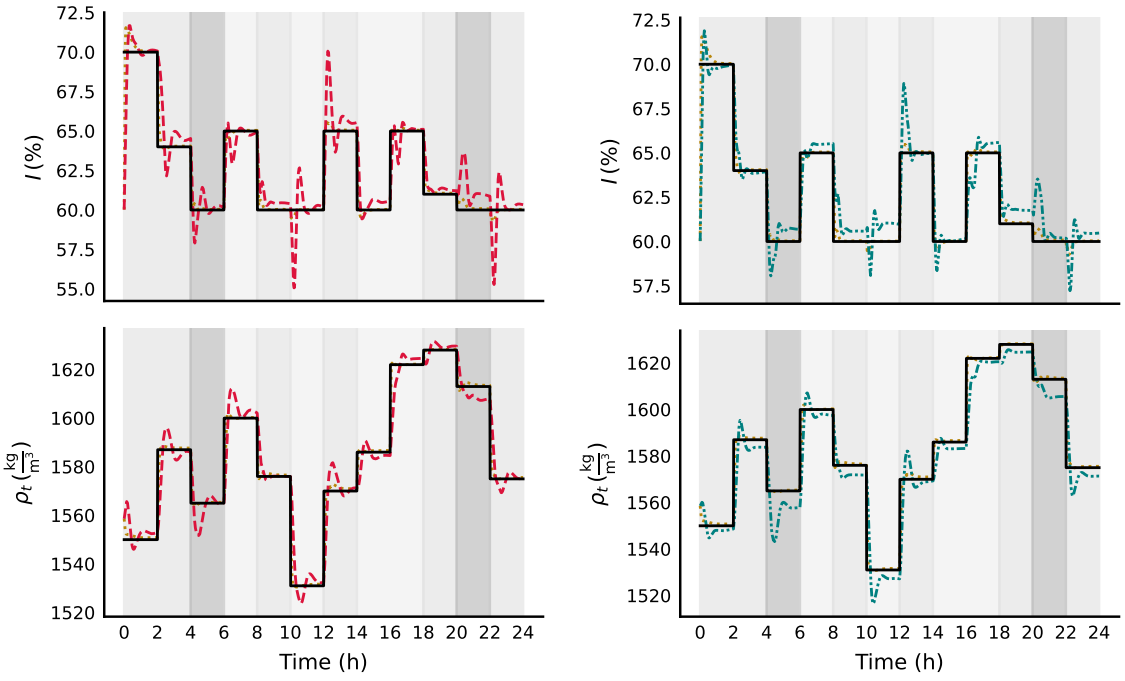
Figure 4.11 shows the performance of RLC after online training. The RLC is offset-free, stable, and robust to the random setpoint changes and the disturbances in ore grade and plant capacity. This underscores the potential of RLC for complex control tasks. It is ready to accommodate setpoint changes from real-time optimization and production capacity planning from the upstream mining. A key strength of RLC over MPC is the ability to operate under a unified policy that adapts to different operating modes. Section 4.5 shows that MPC requires scheduling under varying operating

modes, whereas RLC can integrate multimode control into a single policy. This is advantageous in dynamic environments with rapid changes. Another challenge that RLC tackles is the partial observability in ore grade information. To overcome this, the agent receives density parameters to infer the grade. This requires interpreting and processing relevant information for decision-making. The density profile across PSV depends not only on grade but also on flow rates. The agent must learn relationships between grade, flow rate, and densities altogether in inferring the current mode and dynamics. RL ability to explore is a key to learning complex mechanisms and discovering optimal policy. Given sufficient contexts, RL works under multimode operation.

Table 4.2 summarizes the controller performance in terms of its tracking performance and controller effort. Sim2Real pretrained agent has lower overall tracking error compared to GAIL. Furthermore, online training for GAIL and Sim2Real pretrained agent further reduces the ISE by 88% and 78%, respectively, showing RL continual learning ability. While the control performance does not yet outperform MPC, we observe lower controller effort with RL controllers. RLC controller action is smoother due to the far-sighted optimization objective instead of immediate reward as observed in MPC (fig. 4.12). This benefits the lifespan of control elements and maintains quiescent conditions in the PSV. Moreover, the model-free RL avoids the potential fluctuation due to the plant-model mismatches in MPC (fig. 4.5). It also adapts to new or non-stationary environments through continual learning. Additionally, RLC achieves computational speeds up to 10 times faster than MPC, which is a significant reduction in power consumption. This demonstrates RL potential in industrial control, which is characterized by multimodal, multi-input multi-output, complex dynamics, high-dimensional states, and elements of randomness and non-stationarity.



(a)



(b) (c)

Figure 4.11: RLC performance after online learning: a. Direct training; b. GAIL pretrained; c. Sim2Real Pretrained

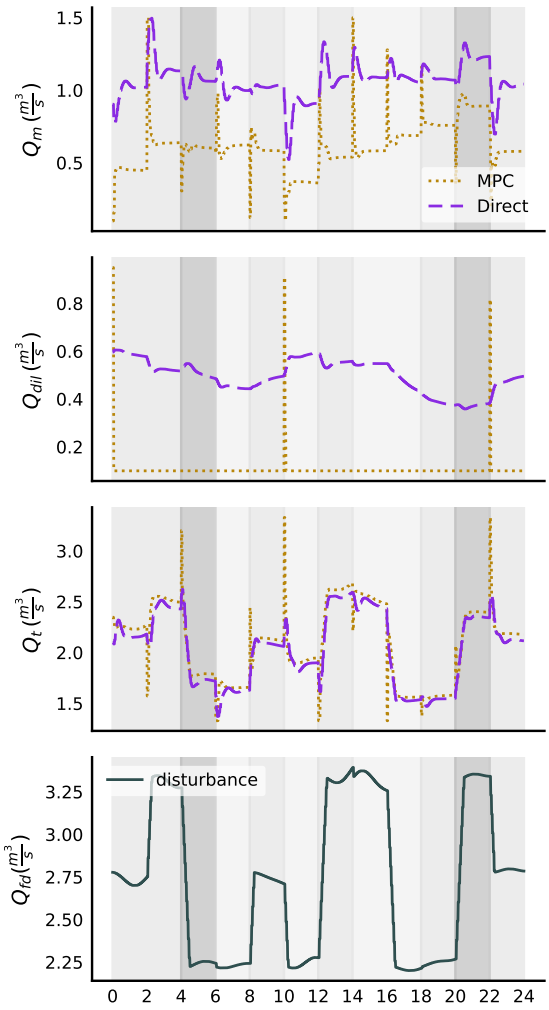


Figure 4.12: RLC vs MPC controller actions

Table 4.2: Controller Performance Metrics

Controller	ISE I 10^4	ITSE I 10^4	ISE ρ_t 10^6	ITSE ρ_t 10^6	IAU Q_m	IAU Q_{dil}	IAU Q_t	CPU Time
MPC	4	310	1	43	73	17	103	180
Direct	21	4546	7	778	40	5	47	18
GAIL Just After Pretraining	89	24549	19	4992	26	38	55	17
GAIL Post Online Learning	26	4923	7	943	32	13	49	19
Sim2Real Just After Pretraining	18	4839	9	1714	33	30	46	17
Sim2Real Post Online Learning	14	2741	6	676	25	25	49	18

4.8 Safety and Feasibility Analysis of Reinforcement Learning-Based Controller

While RLC facilitates autonomous control, their deployment warrants careful analysis of their compliance with safety protocols in the process. Industrial processes typically comply with the safety standards from the International Society of Automation (ISA) and the International Electrotechnical Commission (IEC). Based on these standards, a Safety Instrumented System (SIS) is added to the system with potential hazards that can not be mitigated by the existing protection. Each hazard that is protected by SIS must be assigned with the Safety Integrity Level (SIL). It measures the level of risk reduction provided by a safety function or a target level of risk reduction by SIS.

Figure 4.13 illustrates the layers of protection. The prevention layer begins with safety design and extends through basic process control systems (BPCS), alarm systems, operator interventions, SIS, and relief devices. The mitigation layer comprises passive protection and emergency response. Each layer acts as a barrier against abnormal events escalating. Any additional systems to the process require rigorous validation to ensure they do not undermine or circumvent existing safety measures. We evaluate the risk of implementing autonomous control in the BPCS layer using a Reinforcement learning-based controller (RLC). The severity and frequency of potential hazards must be evaluated, specifically those related to the instrumentation failures. RLC serves as the supervisory control in the cascade control setup. Thus, final control elements are regulated by conventional PID controllers. Control valves are designed with a limit that is suitable for a range of operating modes. Moreover, agent actions can be scaled or clipped within the constraint. Either way ensures RLC satisfies input constraints. Output constraint poses another challenge that is not specific to RLC. Even model-based controllers like MPC cannot guarantee output constraint satisfaction in practice due to model mismatches and disturbances.

Alarm management, operator intervention, and SIS serve as the protection layer to the potential violation of output constraint. Operator intervention de-escalates the abnormal operation that compromises the product quality or production disruption due to process trips. SIS provides penultimate safeguards through alarms and automated emergency shutdowns that protect equipments and prevent accidents. Relief devices provide the ultimate safeguard to release excess pressure or materials to a safe location when critical thresholds are surpassed. Functioning as a critical fail-safe mechanism, these devices are activated in situations where primary safety controls are inadequate or malfunctioning, thus providing the last line of defense against potentially disastrous events. The layered protection ensures RLC deployment preserves the overall safety integrity of the process.

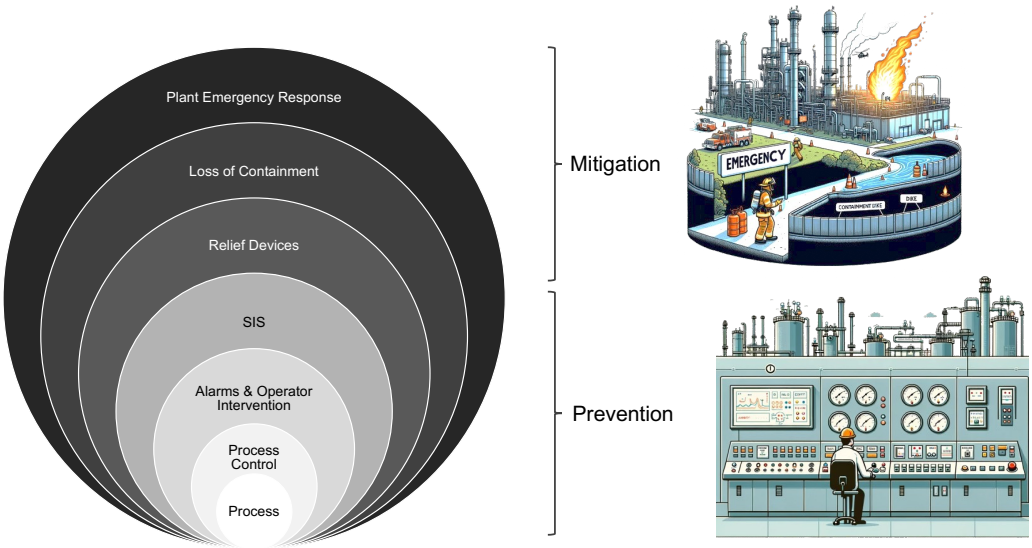


Figure 4.13: Hierarchical layers of process safety

Process trips indicate potential accidents. We define trip conditions when the interface level is too high, causing poor bitumen quality, or too low, risking excessive bitumen loss from the flotation tailings. Tailing density trips pose even greater severity – overly high density could activate relief devices and cause pipeline clogging, while too low density causes poor separation. In terms of the frequency of potential

hazards, an agent with online training results in zero process trips during a one-month equivalent operation testing in the digital twin. However, our research emphasizes ensuring safe deployment not only during the testing phase but also during the training phase. Table 4.3 summarizes trip count during simulation equivalent to 5 years of equivalent online training. Pretraining effectively reduces trip frequency during online training. GAIL and Sim2Real pretraining reduce trip count by factors of, 8 and 27 times, respectively, compared to the direct agent training in the environment (table 4.3). This is a significant reduction, as we maintain the default exploration variance during online training section 4.4. Further tuning of the exploration variance can reduce the trip frequency. The acceptable value of trip count depends on the criticality of the equipment. Frequent trips can severely disrupt processes, affect production quality and targets, increase mechanical wear and tear, and add to manpower costs for manual restarts. For equipment like Primary Separation Vessel (PSV), which requires quiescent conditions, maintaining a minimum trip frequency is essential. We propose an architecture that integrates an RLC with an MPC acting as a safeguard, termed "MPC safeguarded exploration." This framework leverages the MPC strengths in constraint satisfaction and predictive planning to counterbalance the exploratory risk of reinforcement learning.

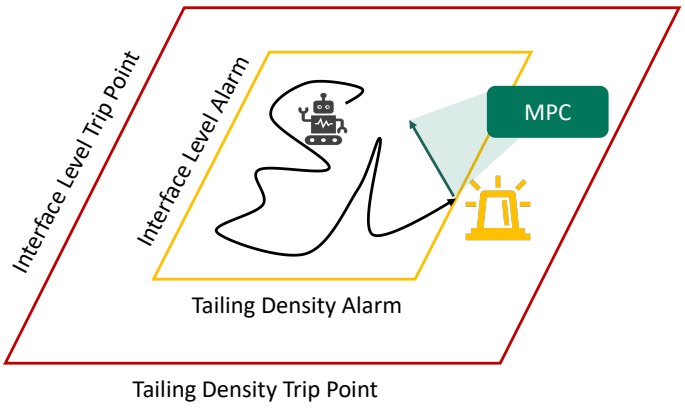


Figure 4.14: MPC safeguarded exploration

Alarms management is strategically used within this framework to delineate safe operational and exploration boundaries (fig. 4.14). High-high (HH) and low-low (LL) alarms indicate trip conditions to ensure operation safety. High (H) and low (L) alarms signal deviations that, while not immediately critical, require corrective action. During the design phase, high and low alarm thresholds are determined to allow sufficient time for corrective responses prior to the activation of trip conditions. These alarms delineate safe boundaries to constrain the exploration space for RL agents. When exploratory actions move process conditions toward these alarm thresholds, MPC promptly overrides agent control authority. It shifts the system back within the normal operating ranges aligned with the current setpoint.

Figure 4.15 indicates that a constrained exploration space does not hinder the convergence to the optimal policy. While the alarms limit the exploration space, the RL agent can still freely explore within the safe boundaries. The policy optimality remains unaffected provided the optimal trajectory resides within the normal operation. Extraneous exploration in unsafe states does not contribute to the convergence to the optimal policy and introduces unnecessary risks. Agents converge at similar rates to those without guided exploration, while attaining higher rewards. Negative rewards at the beginning stem from the penalty on MPC intervention to discourage over-reliance on MPC.

MPC safeguarding eliminates trips during online training of the Sim2Real pre-trained agent (table 4.3). Direct RL training, when coupled with MPC safeguarding, reduces the trip count by 236 times, suggesting the feasibility of direct RL deployment on the non-critical units where trips have lower impact. This analysis suggests safe real-world training is possible. In practice, further tuning on degree of exploration and implementing more stringent pre-trip alarms can minimize trips. Algorithm selection also helps, as PPO avoids large updates by capping policy updates each iteration. To further mitigate risk, agent actions can be deterministic or reduced in stochasticity. We maintain, however, default stochasticity to rigorously evaluate robustness

of MPC safeguarding. The resulting "safeguarded exploration" embodies a prudent approach for industrial RL by ensuring safety in both training and testing phases without limiting the innovative potential of RLC. Various adjustments as described could improve reliability and enable wider adoption.

Table 4.3: Trip count and MPC intervention count during online training

RL Agent	Stand Alone RL	MPC Safeguarded RL	
	Trip Count	Trip Count	MPC Interventions Count
Direct	2591±129	11±2	12085±4254
GAIL Pretrained	318±210	2±2	3644±951
Sim2Real Pretrained	95±20	0±1	1556±180

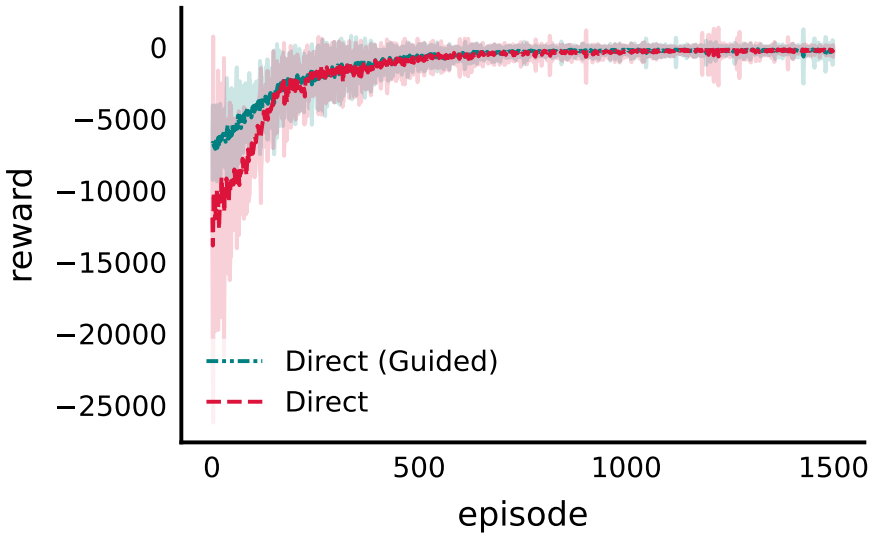


Figure 4.15: MPC safeguarded learning curve

4.9 Conclusions

Model-free Reinforcement Learning is capable of handling complex, multi-input multi-output, multimode, partially observable, random disturbance, and constrained industrial-scale control tasks. The testing environment in the digital twin not only replicates actual process dynamics but also provides more severe conditions, including random setpoint changes in both interface and density, compared to typical operations with mostly fixed setpoints. The RLC is ready to accommodate setpoint changes from real-time optimization and production capacity adjustments from upstream scheduling. Moreover, its adaptive and continual learning ability is suitable for industrial processes with nonstationarity and time-varying uncertainties. Additionally, RLC computation is up to 10 times faster than MPC, significantly reducing power consumption.

Safe RL deployment is ensured through the layered protection inherent in the industrial process. Transfer learning addresses the feasibility consideration of trip frequency through imitation learning and the Sim2Real pretraining framework. Imitation learning has an additional benefit as it can learn without an objective function, making it suitable for autonomous restart and shutdown sequences where objectives are not easily described; demonstration data in this stage can be used to train the agent. Behavioral cloning has limitations in pretraining agents when there is a distributional shift in the environment. However, Generative Adversarial Imitation Learning and Sim2Real pretrained agents demonstrate satisfactory control performance immediately after pretraining and reduce trip conditions during online training by 8 and 27 times, respectively. The MPC safeguarded approach strategically leverages the constraint satisfaction, predictive planning of MPC, and alarm management to eliminate trip frequency of Sim2Real pretrained agents within the simulation equivalent to 5 years of online training in the digital twin. It also makes direct agent training feasible.

In summary, this research explores the power of RL as autonomous in its ability

to make complex decisions without the need for a model. While its control performance may not yet outperform MPC, it addresses potential fluctuations encountered in our study. It effectively learns to make complex decisions based on partially observable information, establishing a unified policy adaptable to varying conditions. RL decisions are farsighted and able to maintain consistent and smooth controller effort, benefiting instrument lifetime. RL demonstrates a continual learning ability, a vital attribute for autonomous systems that require self-corrective capabilities. This feature is especially promising for deploying agents trained in digital twins into real-world processes, accommodating smooth transitions between environments. Transfer learning strategies have shown significant potential in bridging the gap between simulated training and actual industrial application. Moreover, a safeguarding mechanism has proven effective in ensuring the safe and reliable deployment of RL in complex industrial settings. Further tuning of exploration degree, more restrictive exploration space, and algorithm selection potentially improve the feasibility of RLC deployment, thus enabling the innovative solution in autonomous control.

Chapter 5

Autonomous Real-Time Optimization and Adaptive Fused Multi-Model Predictive Contol Using Explainable Reinforcement Learning

This work was presented at the 2023 INFORMS Annual Meeting

5.1 Introduction

The advent of Industry 4.0 has ushered in a transformative era, where advanced digital technologies and Artificial Intelligence (AI) are reshaping system engineering [141, 142]. A key focus in this revolution is autonomous Real-Time Optimization (RTO) and Advanced Process Control (APC) [143–146]. Among various applications, a particularly pertinent application is the bitumen extraction from oil sands, a sector crucial for its substantial energy resources [84, 85]. However, realizing RTO in the bitumen extraction process presents several challenges, including random disturbances, multi-mode operations, non-linearities, with high-dimensional state-action spaces, and intricate separation mechanisms constraints.

A key contributor in addressing these challenges could be Reinforcement Learning (RL), which has consistently outperformed expert performance in various settings. This ranges from its groundbreaking success in AlphaGo [147], video gaming [7–10], and extends to real-world applications in stratospheric balloon control [21] and nuclear fusion systems [22]. A recent breakthrough has showcased RL metacognitive abilities [23]. By interacting with the world, RL algorithms discover things humans could not even imagine at first [40–43]. RL ability to explore and discover solutions to intricate, intractable problems solidifies its role in advancing industrial optimization. Moreover, adaptive and self-learning capabilities allow RL to autonomously optimize processes, adapting to changing environments and the inherent uncertainty in industrial settings [148]. This adaptability makes RL a powerful tool for navigating intricate and dynamic operational landscapes.

On the other hand, Model Predictive Control (MPC) is renowned for its robustness, especially in maintaining safety constraints in industrial applications [149]. The earlier chapter in autonomous model predictive control (MPC) effectively managed process disturbances and multi-variable scenarios. However, plant-model mismatches can lead to either sluggish or overly aggressive control actions by MPC, depending on

whether the model overestimates or underestimates the process dynamics [150, 151]. Fine-tuning MPC parameters does not fully resolve this issue due to the inherent model mismatches.

One solution is Multi-MPC scheduling, incorporating a switching algorithm that combines the predictive models or MPCs output in response to changing operating modes [152, 153]. Model fusion captures dynamics more accurately, but designing reliable switching mechanisms remains challenging. [154] combines two linear MPC outputs using a fixed linear weighting scheme. [152] utilizes a parametric switching mechanism based on a first principle model. However, industrial operating modes often transition continuously without clear boundaries. Linear weighting algorithms may not effectively represent the intricate and continuous transitions of industrial operating modes that rely on various state variables.

This study is the first to apply RL for Fused Multi-MPC (FM-MPC). Instead of merging various MPC outputs, this approach combines different predictive models while employing a single MPC for control, thereby circumventing the complexity of multiple MPCs. Within this framework, three models corresponding to three representative operational modes are identified. The RL agent then dynamically assigns weights to these models in response to shifting operational modes. The advantages of RL here are twofold: it leverages all relevant state variables for decision-making. Second, the policy structure with deep neural networks is capable of handling the intricacies involved in defining operating modes and the weighting mechanisms. This ensures a continuous and precise depiction of the current operating mode. RL adaptability is also crucial for capturing the complex dynamics of industrial systems through continual learning in their interactions.

In addition to MPC scheduling, this work introduces a multitasking RL to determine optimal setpoints simultaneously to enhance recovery rates. It synergistically combines RL exploratory strengths with the robust safety assurances of MPC. The RL agent determines the optimum setpoint and models fusion for MPC, while

MPC manipulates the process. This integration is designed to facilitate harmonious decision-making, addressing the intertwined nature of RTO and control in industrial operations. Direct RL deployment in the real process, without direct interference in the actual environment, ensures safety and reliability.

The integration of Explainable RL (XRL) into our framework is a critical advancement. To transition RL systems into real-world applications where operators cannot inspect the policy representation, the systems must be capable of offering explanations for their behaviors [155, 156]. XRL enhances transparency and understanding in AI-driven decision-making processes, enabling validation and learning from explainable policies [157, 158]. T-distributed Stochastic Neighbour Embedding (t-SNE) on neural activations visualizes state perception [159, 160]. Model-based approach explains a series of actions using models such as MDPs and Bayesian Networks [161, 162]. The intentions behind decision-making policies can be explained through reward decomposition [163], Local Interpretable Model-Agnostic Explanations (LIME) [164], and multi-objective RL [165]. The value-based approach uses historical interaction to extract interestingness elements through introspective analysis [166]. This study employs sensitivity analysis and physics information to derive first-order explanations of the policy, focusing on the agent *perception* on the environment, the resultant action (*introspective*), and the underlying intent of such behavior (*influenced*). The goal is to enhance the acceptability of advanced AI solutions and the possibility of their indirect deployment in industrial settings. The underlying concept is that RL can uncover optimal solutions that might otherwise go unnoticed, bringing them to light for human operators to realize and implement. This combination of advanced machine learning techniques and human oversight ensures a balance between innovation and safety, enabling the responsible and effective use of RL in industrial settings.

Key Contributions of This Research:

- **Fused Multi-MPC:** This work is the first to apply RL for an adaptive weight fusion method in a fused Multi-Model Predictive Control (FM-MPC). The agent seamlessly discerns the current operating mode given the states and fuses multiple models with different weights depending on the operating mode. This approach enhances adaptability and precision in predicting dynamic environments.

- **Multitask RL Framework:** Innovating the integration of RTO and multi-MPC scheduling within a multitask RL framework, merging the explorative power of RL with the constraint satisfaction of existing MPC infrastructure.

- **Explainable RL (XRL):** Advancing XRL through the use of first principle sensitivity analysis to ensure the transparency and reliability of policy implementation.

- **Digital Twin-Assisted Training:** Utilizing a digital twin environment to accelerate RL training from years to days, thereby propelling research and real-world application of RL. This digital twin maintains high fidelity, capturing the dynamics, constraints, and uncertainties of real processes, offering a low-risk and dependable testing ground for RL design.

Our methodology tackles the multifaceted challenges of real-time optimization within the Industry 4.0 paradigm, blending the adaptive strengths of RL with the robust safety measures of MPC to facilitate more intelligent and safe industrial automation. Moreover, XRL illuminates the decision-making process of RL agents, offering validation for policy reasoning and bolstering the implementation of RL, both directly and indirectly.

5.2 Optimization and Control problem

The Primary Separation Vessel (PSV) recovers 90% of the bitumen in the extraction process. Operational decisions involve setpoint optimization and controlling Multi-Input Multi-Output (MIMO) variables subject to various disturbances. The natural variability in ore grade and feed flow rate changes due to production scheduling determine the operating modes, each with unique dynamics. High ore grades and plant capacity lead to faster system dynamics. The system faces constraints on both its inputs and outputs, and its interconnectivity with upstream and downstream units affects the overall dynamics and bitumen recovery rate. This complexity necessitates a tractable solution for Real-Time Optimization (RTO) under uncertainty.

The controller's limited adaptability to setpoint changes from RTO and multi-mode operation expose a significant limitation. The current operational practice is to maintain the system within a fixed target operation setpoint. This strategy prioritizes the system stability over recovery rate performance. Chapter 4 explores Reinforcement Learning (RL) capabilities and its feasibility in real-world applications. However, MPC remains a widely accepted method for autonomous control, credited for its mature technology and ability to satisfy constraints.

Section 4.5 highlights the fluctuation issue in MPC due to plant-model mismatch, especially in changing operating modes. To address this challenge, different MPC models are designed for each operation mode: low, average, and high ore grade scenarios (section 4.3). In a fused multi-MPC framework, RL is the model scheduling algorithm, each model is assigned a weight, and the final prediction is calculated as the weighted average of each model prediction. Thus, the RL agent not only decides the optimum setpoints but also schedules MPC models by fusing the model predictions with different weights. This dual-task framework is feasible for direct real-world deployment by synergizing MPC robustness with RL explorative and complex optimization capabilities.

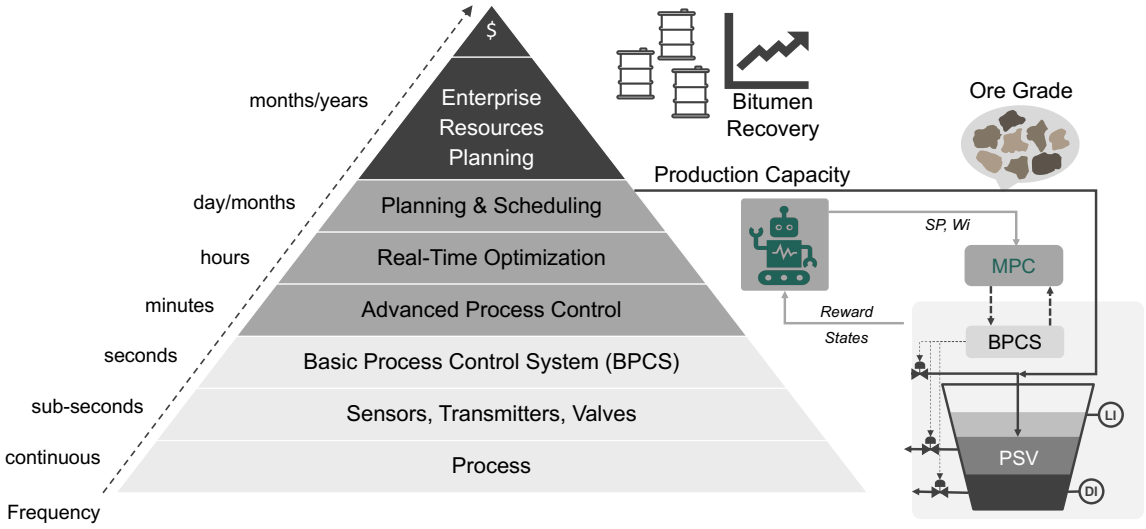


Figure 5.1: Hierarchy of decision-making in industrial processes

The RL agent optimizes the controlled variables (CVs) setpoints, such as interface level and tailings density, to improve the recovery rate. MPC adjusts three manipulated variables to regulate these controlled variables to their setpoints. Thus, the controller has one more degree of freedom. The combination of manipulated variables will affect particle settling dynamics and thus the recovery rate. Therefore, RL agents should anticipate these MPC control policies, optimizing their strategies accordingly. This research aims to automate the Real-Time Optimization (RTO) and advanced control layer as depicted in fig. 5.1.

5.3 Reinforcement Learning-based Optimization and Fused Multi-MPC Design

In this study, we implement Proximal Policy Optimization (PPO), Asynchronous Actor-Critic (A2C), and Deep Deterministic Policy Gradient (DDPG) algorithms. PPO ensures stable learning with incremental updates, A2C is known for its fast convergence and exploratory strength, and DDPG offers a blend of stability and continuous action space proficiency. We aim to select an algorithm that combines stable learning with rapid convergence, minimizing sample inefficiency and ensuring

optimal performance by achieving the highest possible reward during convergence.

The optimization and MPC model scheduling are formulated as MDP. The agent action is to optimize the setpoint and assign weights to MPC models. In a multi-model MPC framework employing RL for model scheduling, each model is assigned a weight, and the final prediction is calculated as the weighted average of each model prediction, denoted as:

$$A = [\Delta I_{SP}, \Delta \rho_{t,SP}, w_1, w_2, w_3]$$

Action intervals are set to be one hour. The agent state includes process states such as density and flow rate, which provide indirect information on the current operating mode. Historical ore grade data are also included to account for the delay from mining to the process. We use a similar approach that incorporates recent history into the agent state to capture the delay effects from upstream mining to PSV [167]. RL state includes setpoints to avoid the setpoint change beyond its normal range. States also include current bitumen recovery rate and loss. All states must be measurable through sensor or lab analysis in the actual process to ensure its applicability. The states are summarized as follows:

$$\begin{aligned} S = [&I, I_{SP}, \rho_f, \rho_t, \rho_{t,SP}, \rho_m, Q_{fd}, Q_{fd,target}, Q_{ff}, \\ &\alpha_b^{fd}(t-1), \alpha_{b1}^{fd}(t-1), \alpha_s^{fd}(t-1), \alpha_b^{fd}(t), \alpha_{b1}^{fd}(t), \\ &\alpha_s^{fd}(t), \alpha_b^t, \alpha_b^{ft}, RR] \end{aligned}$$

The reward mechanism considers the recovery rate, control cost, and penalties for constraint violations. Recovery rates are calculated using cumulative recovery within a one-hour interval. Costs associated with setpoint changes serve to discourage frequent setpoint adjustment. Controller costs arise from the tracking error of the controlled variables, changes in the control effort (Δu), and the number of overshoots. The reward function penalizes model prediction bias, scaled according to the weights assigned to each model. The bias term in the reward function acts as a *reward*

shaping, incentivizing the RL agent to prioritize models with lower prediction errors by assigning them higher weights. The practice of reward shaping in RL adds informative signals to guide the learning process beyond the actual MDP problem [168]. This helps the agent learn model scheduling based not only on control performance but also on bias, which is more direct. Penalties are imposed when the setpoint is changed beyond a normal operating point. Another penalty is imposed when the environment approaches the trip condition to ensure that the agent considers the controller performance and limitations, i.e., avoiding the optimal point that might not be smoothly achievable by the current controller.

During training, arbitrary disturbances are introduced every two hours, including variations in grade and plant capacity. Continuous fluctuations are also introduced in the feed flow rate to mimic consistent noise in the system. The agent performance is evaluated in each individual operating mode to capture the behavior of its policy. It is further evaluated under varying operating modes to validate its robustness against random disturbances in ore grade and plant capacity.

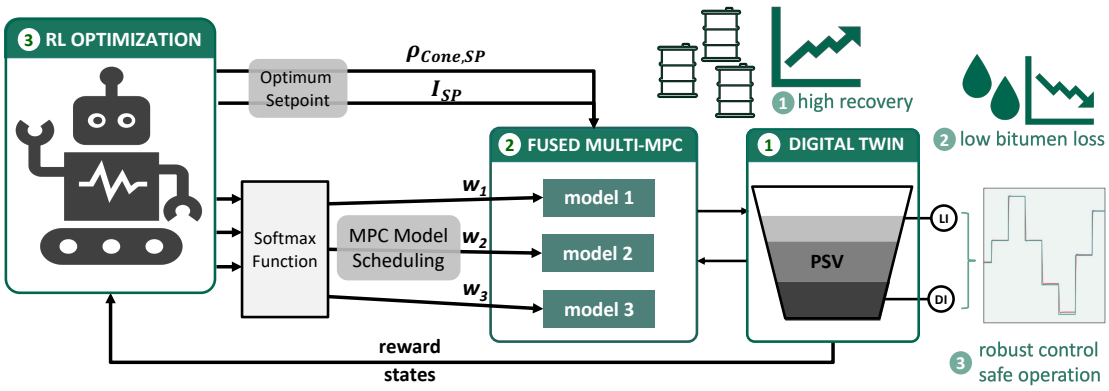


Figure 5.2: Optimization and MPC model scheduling framework

5.4 Optimization and Control Performance Under Individual Grade

Figure 5.3 illustrates the moving average rewards of A2C, PPO, and DDPG agents.The A2C agent not only achieves a higher reward but also converges 2 and 3 times faster than DDPG and PPO, respectively. The final reward value is also the highest. Therefore, performance evaluation in subsequent sections focuses on the A2C agent. Although A2C does not avoid large updates like PPO, employing A2C in the RTO layer is safe as it only adjusts the setpoint and assigns different weights to the MPC model. Setpoints are clipped within the normal range. The MPC model will still ensure process constraints are met regardless of the model used, with the worst case being suboptimal control performance.

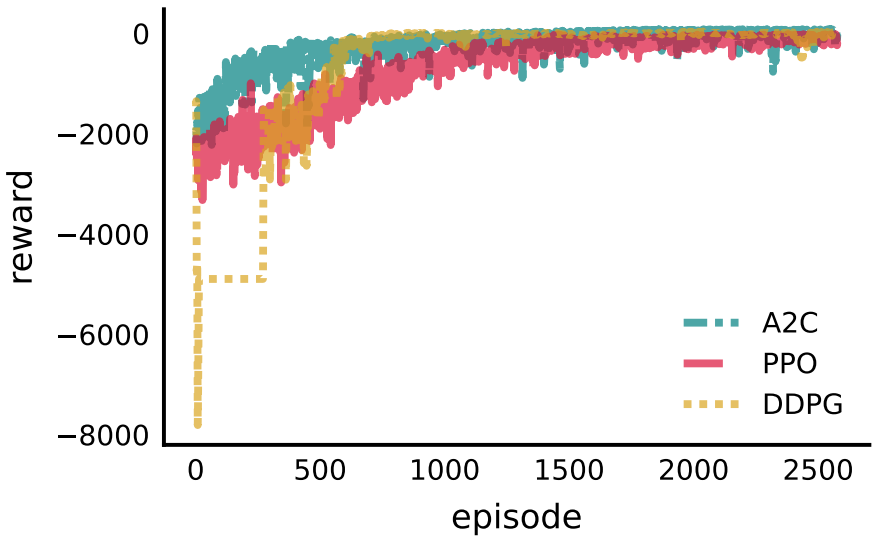
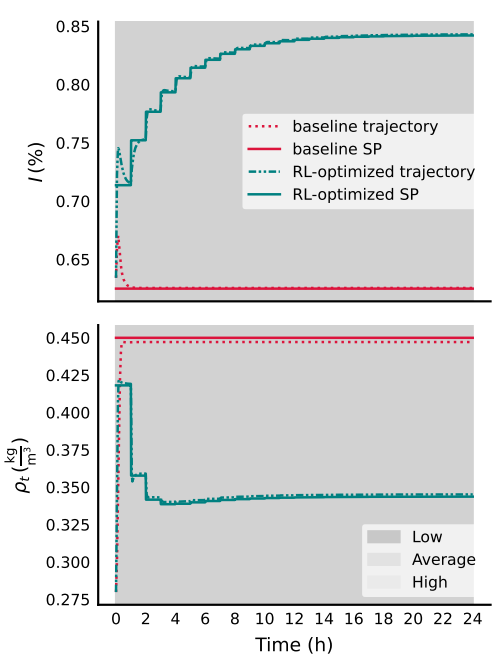


Figure 5.3: Learning curve

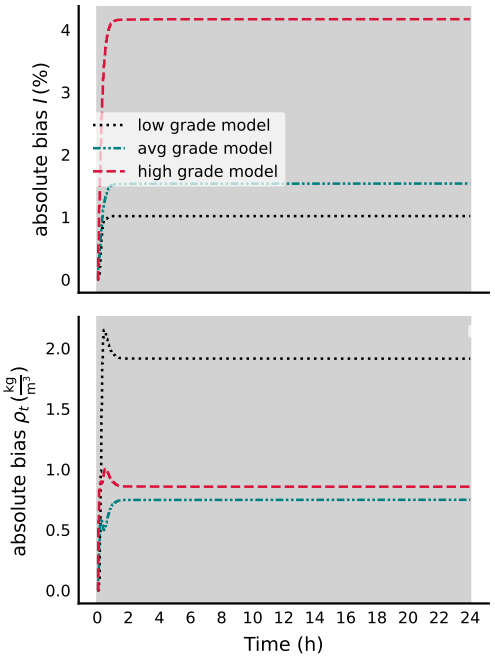
Figure 5.4 presents the trajectories of controlled variables, the recovery rate, and the agent decisions under low-grade operation. The existing operational strategy with fixed interface and tailings density setpoints, serves as a benchmark for comparison. This benchmark is managed by MPC with the high-grade model. As shown in Section 4.5, a high-grade model delivers the best overall performance across various

operating modes, if only one MPC is to be used. Although the training incorporates random disturbances in feed flow rate and ore grade, the evaluation of the agent performance is conducted under each specific ore grade operating mode to capture locally generalizable patterns.

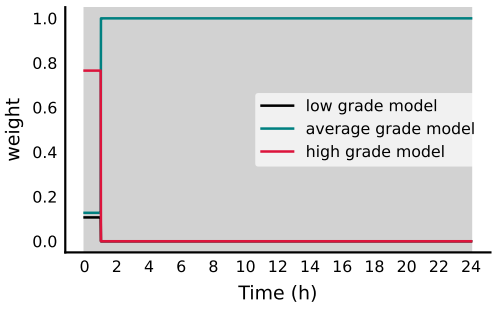
To maintain the confidentiality of the benchmark operation, the y-axis is scaled. The recovery rate objective and controller scheduling performance objective are discussed separately. The RTO agent increases both the interface level setpoint and tailings density from their steady-state values, resulting in a higher recovery rate compared to the benchmark operation. In terms of MPC model scheduling, the RTO agent fully weights the average model, which is optimal based on its satisfactory control performance. Upon closer examination, the average-grade model shows significantly less bias in predicting tailings density than the low-grade model. Both models show similar biases in the interface level predictions. Therefore, the agent decision to select the average grade model appears reasonable. This analysis underscores the agent capability to schedule MPC models based on variables beyond ore grade, as all states can influence the weight decision.



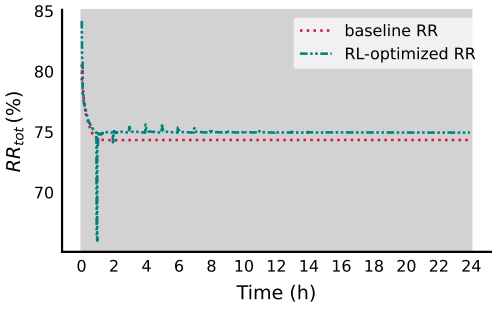
(a) Control performance



(b) Bias



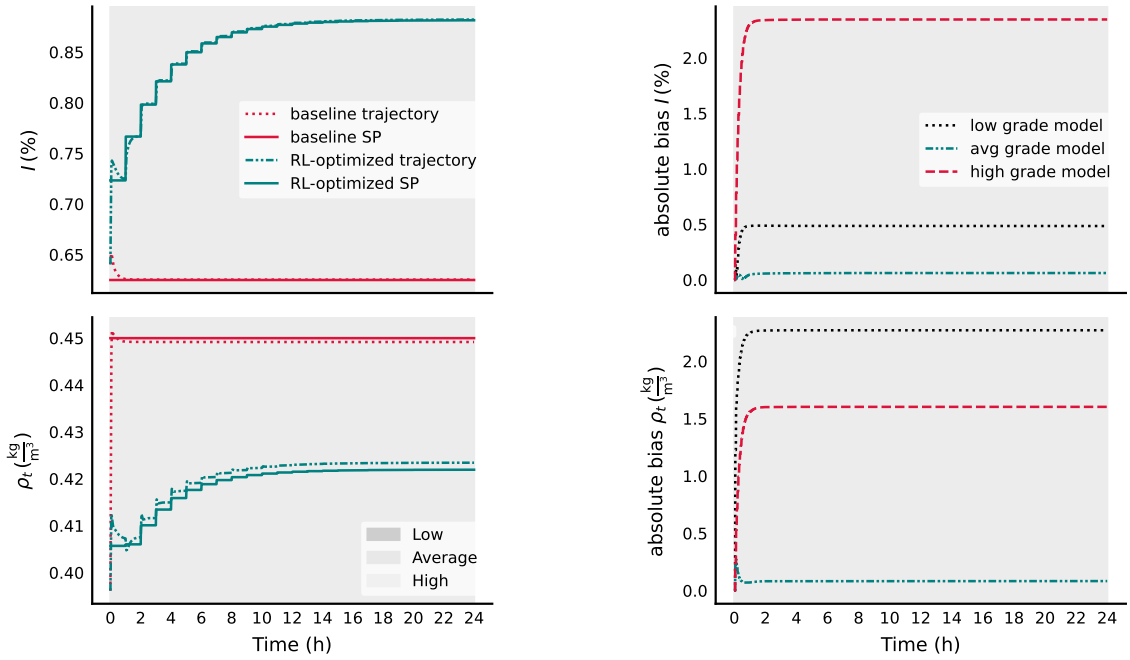
(c) Weight



(d) Recovery Rate

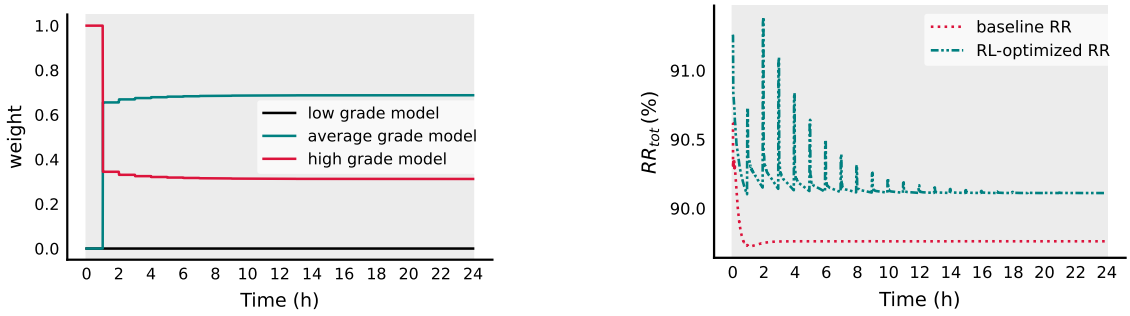
Figure 5.4: Recovery rate optimization and MPC model-scheduling under low-grade operation

During the average ore grade operation, the agent also raises the interface level and tailings density setpoints from their steady-state values. Notably, the tailings density is set higher compared to the low-grade operation. This adjustment results in a recovery rate that surpasses the benchmark operation. In terms of MPC model scheduling, the agent predominantly assigns weight to the average grade model, acknowledging its lower biases in predicting interface level and density. This weighting strategy demonstrates satisfactory control performance (fig. 5.5).



(a) Control performance

(b) Bias

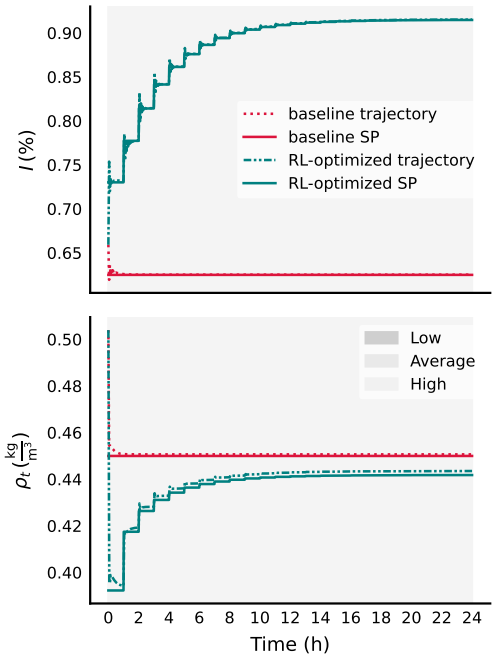


(c) Weight

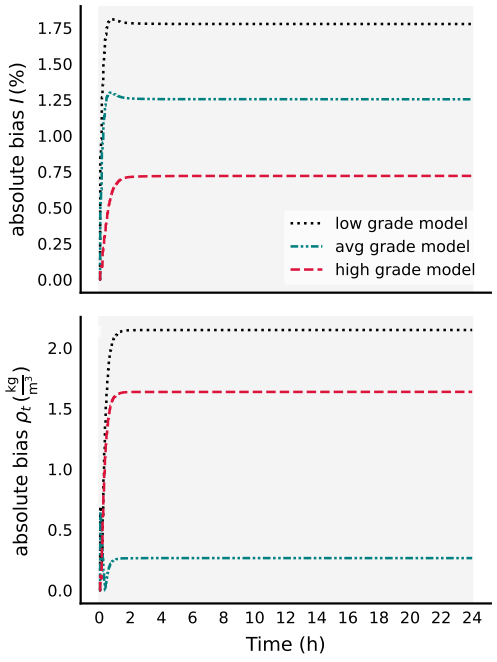
(d) Recovery Rate

Figure 5.5: Recovery rate optimization and MPC model scheduling under average-grade operation

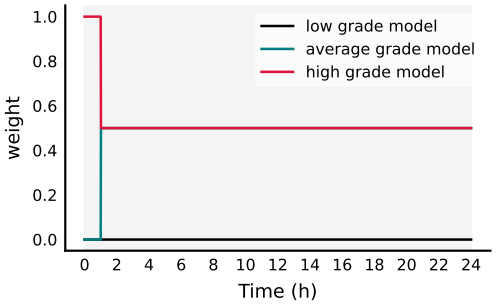
Under high-grade operation mode, the RTO agent raises the interface level setpoint, consistent with the previous two cases. However, it lowers the tailings density setpoint from the initial steady-state, although the final density remains above the average grade scenario. This suggests the agent prefers higher interface levels while tailoring density setpoints to each grade mode, adhering to an increasing trend from low to high grade. Regarding MPC model scheduling, the RTO agent equally weights the average and high-grade models. This decision aligns with the prediction biases. The high-grade model shows the lowest interface level prediction bias, while the average-grade model is more accurate for tailings density prediction. As the tailings density setpoint is set in the average operation range, the average-grade model predicts the tailings density more accurately. This observed pattern indicates that while the interface level dynamics is mainly influenced by the ore grade, tailings density also depends on its operating region (fig. 5.6).



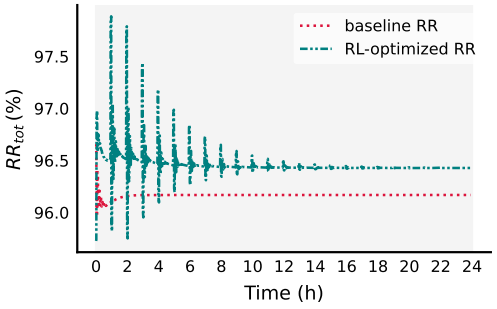
(a) Control performance



(b) Bias



(c) Weight



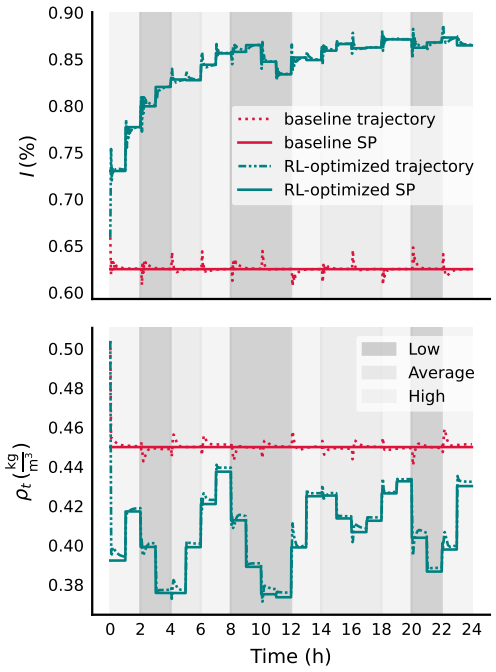
(d) Recovery Rate

Figure 5.6: Recovery rate optimization and MPC model scheduling under high-grade operation

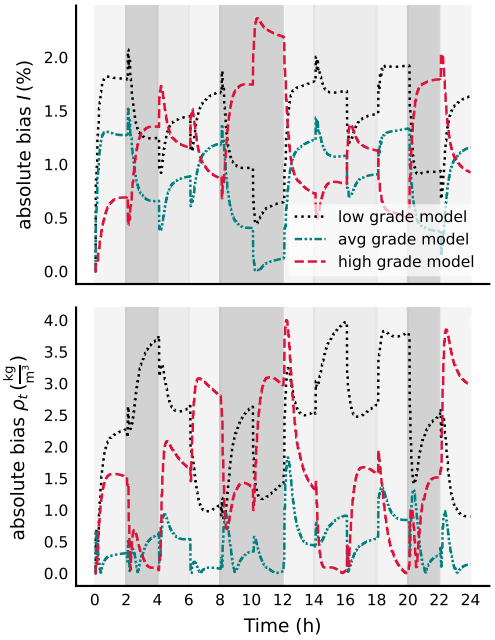
5.5 Optimization and Control Performance Under Random Disturbances

This section tests the RTO agent performance under random fluctuations in ore grade and plant capacity. Figure 5.7 shows the agent optimizes recovery rate in multiple modes without any upset in the controller performance. The recovery rates are higher than the benchmark operation. Our calculations suggest that the RTO-optimized setpoints could reduce bitumen loss daily by 190 bpd under random disturbances (table 5.1). Notably, the agent proved capable of scheduling fused multi-MPC and eliminating inaccurate models.

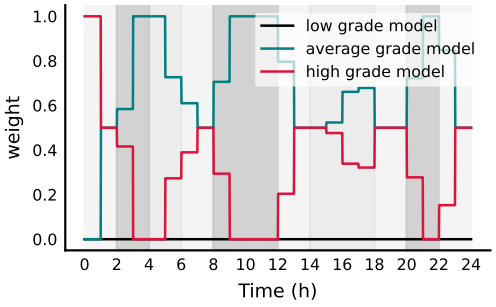
The policy reveals strategies applicable in real-world scenarios. First, we observe that the interface level dynamics are primarily dependent on ore grade, while tailings density is influenced by both ore grade and the operating region. This insight lead to the identification of tailings density grade as another scheduling variable for MPC. Second, the agent favors a higher interface level, preventing bitumen-rich froth from reaching the flotation cell and potential loss through flotation tailing. Third, the agent tends to set higher density setpoints for higher grades, a strategy justified through a first-principles analysis as shown in section 5.6. This approach appears reasonable, as it avoids excessive fines under low-grade conditions. Higher tailings density can lead to an accumulation of fines, impeding separation efficiency. The RL policy not only consistently improves recovery and maintains satisfactory control performance, but it also identifies strategies that are explainable. Consequently, it outlines a potential strategy for indirect policy application in real-world settings.



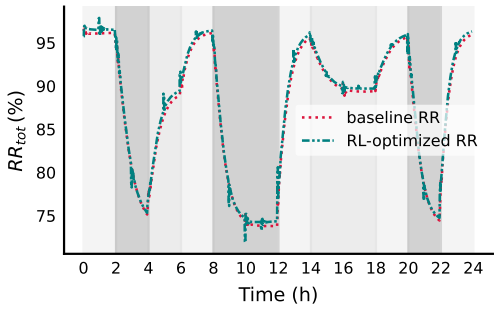
(a) Control performance



(b) Bias



(c) Weight



(d) Recovery Rate

Figure 5.7: Recovery rate optimization and MPC model scheduling under random grade and plant capacity disturbances

Table 5.1: Recovery rate optimization and bitumen loss reduction

Ore Grade	RL Optimized RR	Baseline RR	bitumen loss reduction		
			m3/day	bpd	tpd
Low	75.0%	74.4%	42	266	44
Avg	90.2%	89.8%	29	184	30
High	96.5%	96.2%	24	152	25
Random	88.4%	87.9%	30	190	32

5.6 First Principles Analysis of RL Policy

RL agents interact with an environment by executing actions, leading to new states. Unlike supervised models which focus solely on prediction, RL optimizes control policies by learning state-action values. This learning process inherently involves a temporal relationship that links previous states, actions taken, and subsequent states and rewards, creating a dependency chain through the sequence of decisions [155]. Building on the observed policy patterns from previous sections, this section extends our focus to a first-principle analysis of agent policy. The first-order analysis includes the causal link between perception and action, as well as intentionality, the underlying intention that motivates the agent actions within its environment.

Figure 5.8 illustrates the effects of tailings density and ore grade on multiparticle settling. Terminal particle settling in the Primary Separation Vessel (PSV) is influenced by both the Stokes' velocity, with the density gradient as a primary driving force, and the hindrance effect, which is related to particle concentration. A higher density results in greater particle accumulation within a layer, increasing both the Stokes' velocity and hindrance effects. Balancing these factors is crucial to maintenance of a high terminal velocity for effective separation. Additionally, as a flowing system, bulk flow also impacts the net settling velocity. Therefore, our analysis considers both particle settling velocity and flow rate of each withdrawal stream to elucidate the agent policy.

The overall recovery rate is calculated as the percentage of total fed bitumen minus the losses through PSV and Flotation (FT) tailings, relative to the total bitumen feed. Consequently, the recovery rate is inversely proportional to losses from both PSV and FT tailings. Particle separation in FT cells differs from that in PSV; bitumen is aerated in FT cells, floating to the FT cell froth. Excessive middling withdrawal to FT cells can overload their capacity, leading to a higher percentage of unaerated bitumen lost through FT tailings. Therefore, optimizing recovery involves considering

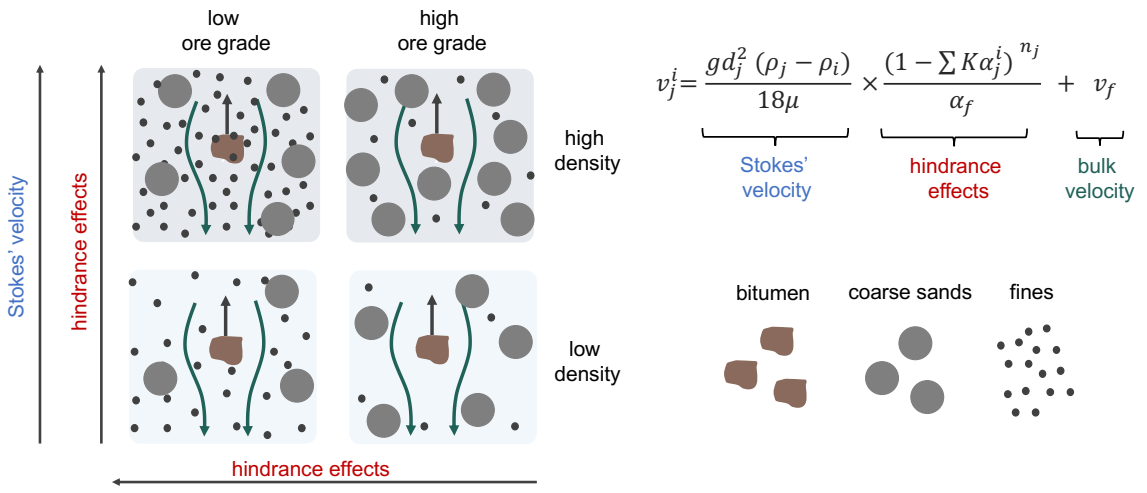


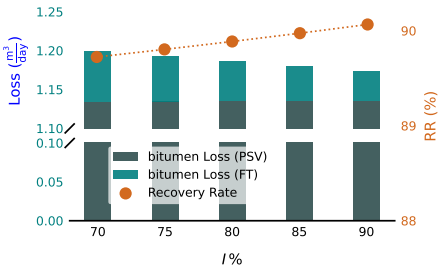
Figure 5.8: Schematic diagram of particle settling: effects of ore grade and tailings density

not only the PSV interface level, tailings density, and withdrawal flowrates but also the load distribution between PSV and FT cells.

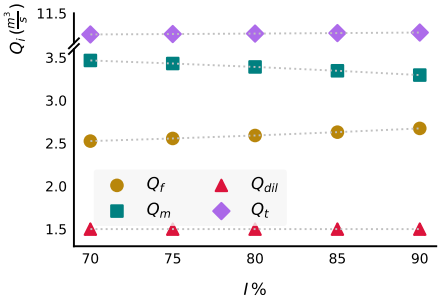
Regarding the recovery optimization, RTO agent consistently increases the interface level in any scenario. For the density setpoints, it assigns progressively higher density setpoints across different grades, meaning the lowest density setpoint is allocated for the low grade, followed by the average grade, and finally, the highest setpoint for the high ore grade operation. We conduct separate sensitivity analyses to evaluate the effects of interface level and tailings density on the recovery rate. Given the consistent sensitivity across all operating modes, we have chosen to illustrate using the average-grade scenario.

The recovery rate increases with the interface level, while overall bitumen loss decreases as the interface level rises (fig. 5.9a). Notably, when the interface level is raised, MPC responds by reducing middling withdrawal while maintaining the dilution water and tailings withdrawal rates. This reduction in middling withdrawal is compensated by an increase in froth overflow. Consequently, the lower middling withdrawal eases the workload on the secondary FT cells, thereby reducing bitumen loss through flotation tailings. Furthermore, an increase in froth overflow elevates

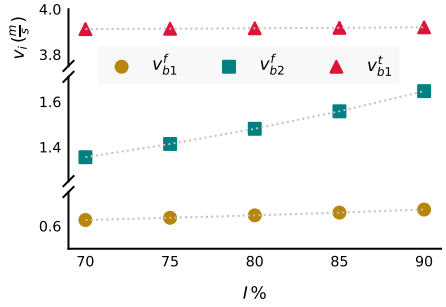
the velocity of unaerated bitumen to the overflow, which in turn enhances bitumen recovery (fig. 5.9c). The RTO agent anticipates MPC policy and optimizes setpoints accordingly. This underscores the benefits of employing a multitasking reinforcement learning framework, which effectively coordinates decision-making in both the optimization and control layers.



(a)



(b)



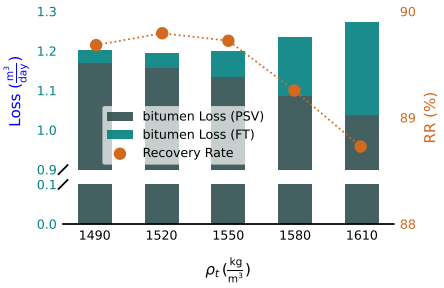
(c)

Figure 5.9: Sensitivity analysis for Explainable RL: effects of interface level on a) overall recovery rate and bitumen loss; b) withdrawal flow rates; c) particle settling velocities

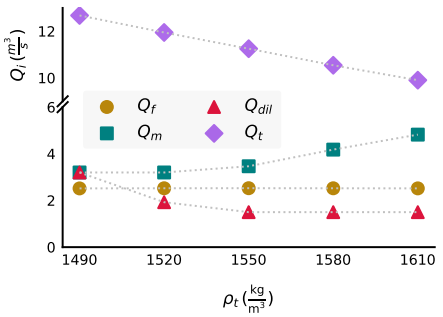
For tailings density, there appears to be an optimal "sweet spot" that maximizes the recovery rate. In increasing the tailings density, MPC decreases both tailings withdrawal and dilution water while increasing middling withdrawal. This aligns with the actual operating procedures. Reducing tailings withdrawal decreases the downward bulk flow effect on bitumen particles, thereby minimizing their loss. However, an excessively high middling withdrawal can lead to increased bitumen loss through the FT tailings due to heightened operational load. Therefore, while increasing tailings density can reduce bitumen loss in PSV tailings, the agent adeptly avoids excessive losses from FT tailings. It maintains a balance between the workloads of the PSV and FT cells, as well as the bulk flow effects. This requires complex reasoning, but the agent effectively realizes and implements this strategy.

For each ore grade scenario, the optimal recovery occurs at different tailings densities. We conducted a targeted sensitivity analysis by adjusting the tailings density $\pm 15\ \frac{\text{kg}}{\text{m}^3}$ around the maximum recovery rate to determine the optimal density more precisely. The results showed that the optimal recovery for low, average, and high ore grade scenarios occurs at tailings densities of 1505, 1535, and 1550 $\frac{\text{kg}}{\text{m}^3}$, respectively. This finding aligns with the RL policy of increasing tailings density as the ore grade increases.

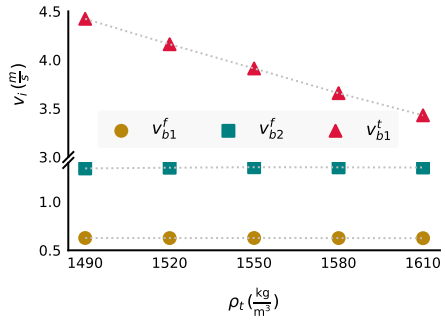
The rationale behind this policy can be explained by the trade-off between Stokes' velocity and hindrance effects (fig. 5.8). Higher tailings densities increase Stokes' velocity but also exacerbate hindrance effects due to increased particle interactions in denser layers. This is particularly relevant in low-grade scenarios with a higher fines content, as fines particles have a higher hydrodynamic volume factor $K > 1$. Therefore, the agent choice of a lower density in low-grade scenarios compared to high-grade ones is justified, striking a balance between maximizing density gradients and managing increased hindrance effects. This strategy provides insights on how setpoints should be adjusted based on ore grade.



(a)



(b)



(c)

Figure 5.10: Sensitivity analysis for Explainable RL: effects of tailing density on a) recovery rate and bitumen loss; b) withdrawal flow rates; c) particle settling velocities

This observation further elucidates the agent strategies in MPC model scheduling. Initially, we presumed ore grade as the primary scheduling variable. However, the agent scheduling strategy takes tailings density into account. This effectively prioritizes models with lower prediction biases. Our analysis confirms that while the interface level is indeed dependent on ore grade, tailings density does not show a direct correlation. We have come to understand that tailings density is influenced

by both the ore grade and its current operating point (fig. 5.8). The tailings density dynamics sensitivity to ore grade changes primarily stems from variances in fines. High fines content in low-grade operations significantly slows down dynamics due to increased hindrance in particle settling. This effect is substantiated by the gain analysis presented in table 3.7, which demonstrates that both the interface level and tailings density exhibit slower dynamics under low-grade conditions. Additionally, in all scenarios, increased tailings density amplifies hindrance, mainly due to sand particle accumulation. Although these particles have a lower hydrodynamic volume factor K, a substantial coarse sand fraction can amplify the hindrance effect, leading to a slower particle settling velocity in tailings. Therefore, both the ore grade, particularly fines content, and the tailings densities are key determinants of the dynamics of tailings density.

5.7 Conclusions

Multitasking Reinforcement Learning (RL) for Real-Time Optimization (RTO) and Fused Multi-Model Predictive Control (FM-MPC) is proposed for the first time. This approach merges RL explorative ability to find optimal policies for complex decisions with MPC's safety advantages. The unified policy under the multitasking RL framework seamlessly orchestrates process optimization and control decisions. RL agent concurrently defines optimal setpoints to maximize bitumen recovery while scheduling different models in the lower MPC layer, handling process constraints with MPC industrial precedence to ensure safety.

High-fidelity digital twin testing showed consistent optimization performance, reducing losses across various operating modes and disturbances from ore grades and production capacities, while maintaining satisfactory control performance. Optimized recovery rates exceeded benchmark operation, demonstrating efficacy in discovering optimal policies for multifaceted problems.

The explainable policy further facilitates the reliability of RL deployment. First-

principle analysis of the setpoint optimization shows that the agent adeptly manages the trade-off between microscale multiparticle settling and workload distribution across interconnected units. The agent learns to optimize recovery rates by maintaining high interface levels within predefined constraints and identifies the optimum tailings density point. It adeptly navigates optimum tailings density across multimode operations to avoid excessive hindering effects. As optimization is also affected by manipulated variables, the agent learns the control policy from MPC and integrates it into its optimization decisions.

RTO agent proficiency extends to MPC scheduling. Extracting information from relevant states allows for a nuanced determination of operating modes and the corresponding scheduling of MPC models. This method surpasses traditional scheduling approaches that predominantly rely on grades for model switching. Its analytical power ensures seamless integration with the existing MPC infrastructure.

The model-free learning is capable of uncovering optimal policies through interaction, equipping a self-learning and adaptive nature. These capabilities facilitate agent pretraining in a digital twin environment, followed by a transition to real-world applications for online tuning. The agent adeptly overcomes mismatches encountered during training and actual deployment.

Additionally, Explainable RL (XRL) opens a new paradigm for indirect RL application. Instead of direct deployment in real-world settings, the policies discovered by RL agents are used to guide the study of optimal decisions. These policies are explainable by first principles, but challenging to identify using conventional methods or human intuition. In this paradigm, the RL agent not only learns but also teaches, unveiling optimal strategies for human operators. This method opens up innovative ways to utilize RL in industrial processes, enhancing decision-making while upholding safety and efficiency.

Chapter 6

Conclusions, Recommendations, & Future Work

6.1 Conclusions

This research merges process engineering with artificial intelligence techniques to achieve autonomous operation, a hallmark of Industry 4.0. First principle modeling creates a high-fidelity digital twin replicating the dynamics and complexity of industrial-scale systems. The proposed 4-layer primary separation vessel model captures key operating variables in the bitumen extraction process like froth-middling interface level and tailings density dynamics. Plant-wide modeling enables the use of actual process data, which allows for the study of separation mechanisms and holistic testing of autonomous strategies. The first principles model explains the phenomena governing process dynamics, such as how interface and density change with different inputs, as well as explaining variations in dynamics under different operating modes. This model serves as a benchmark and testbed for work on optimization and control strategies for an industrial-scale process with constrained MIMO dynamics, multiple operating modes, and interconnected process units. It enables investigation into techniques for control and optimization under such complex real-world conditions.

Control and optimization are the foundation in industrial operations. We aim to automate them to operate the process with reliable, consistent production, minimize environmental footprint, and ensure safe operation. MPC leverages bias correction and disturbance rejection for robust tracking. Reinforcement Learning (RL) based MPC model scheduling further improves MPC adaptability across operating modes. Beyond scheduling, we demonstrate directly using RL as controllers matching MPC performance with additional model-free, self-adaptive, and generalizable advantages critical for automation.

A key contribution is the safe and feasible design of RL-based controllers for industrial-scale processes with constraints, delays, and disturbances. Imitation learning and surrogate-based offline pretraining transfer knowledge to RL agents for safe real-world deployment. RL robustness facilitates seamless mode transitions. Further,

a multitasking RL agent concurrently optimizes control and economic/environmental objectives like recovery rate in an explainable manner based on process physics. Success on previously intractable coordinated optimization and control underscores RL potential. Overall, this work progresses systems engineering through advanced computational techniques to unlock autonomous capabilities in Industry 4.0.

6.2 Future Work

Integrating the digital twin model with online process measurements enhances prediction, early fault detection, and control decision verification for the true process. Detailed upstream models incorporating crusher mechanics and caustic-aerated hydrotransport will characterize feed properties entering the PSV, improving the tailings loss prediction accuracy. The current RL methodology leveraging surrogate modeling, imitation learning, MPC safeguarding, and the digital twin demonstrates the feasibility and safety of Reinforcement Learning (RL) control. Further hyperparameter tuning, algorithm selection, and model-based RL can improve learning efficiency and acceleration. Additional incorporation of safe RL techniques also promotes real-world deployment confidence and process automation adoption. The first principles model reveals flowrate as another factor influencing separation and settling beyond interface and tailings density. Future work will expand the optimization framework for agents to directly manipulate process variables targeting economic objectives. This will demonstrate RL versus economic model predictive control (EMPC). Expanding autonomous operation into downstream flotation units, also tied to recovery rate, introduces potential for further gains. Overall, integrating digital twins, refining RL controllers, and expanding the scope of process optimization collectively enhance the adoption of autonomous decision-making in real-world scenarios. This progresses the bitumen extraction performance, cost-efficiency, safety, and sustainability. The methodologies proven effective on the complex primary separation unit can expand to tackle challenges across other key processes. Advancing journal publications to rigorously document performance could garner interest for real trials in analogous industrial facilities.

Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[2] D. Killock, "Ai outperforms radiologists in mammographic screening," *Nature Reviews Clinical Oncology*, vol. 17, no. 3, pp. 134–134, 2020.

[3] A. Ramesh *et al.*, "Zero-shot text-to-image generation," in *International Conference on Machine Learning*, PMLR, 2021, pp. 8821–8831.

[4] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[5] M. Niemeyer and A. Geiger, "Giraffe: Representing scenes as compositional generative neural feature fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 453–11 464.

[6] M. Havaei *et al.*, "Brain tumor segmentation with deep neural networks," *Medical image analysis*, vol. 35, pp. 18–31, 2017.

[7] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] F. Agostinelli, S. McAleer, A. Shmakov, and P. Baldi, "Solving the rubik's cube with deep reinforcement learning and search," *Nature Machine Intelligence*, vol. 1, no. 8, pp. 356–363, 2019.

[9] C. Berner *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[10] O. Vinyals *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[11] J. A. Bagnell and J. G. Schneider, "Autonomous helicopter control using reinforcement learning policy search methods," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, IEEE, vol. 2, 2001, pp. 1615–1620.

[12] R. Cui, C. Yang, Y. Li, and S. Sharma, "Adaptive neural network control of auvs with control input nonlinearities using reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 6, pp. 1019–1029, 2017.

[13] B. Huval *et al.*, "An empirical evaluation of deep learning on highway driving," *arXiv preprint arXiv:1504.01716*, 2015.

[14] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, eadg1462, 2023.

[15] X. Xiao *et al.*, "Autonomous ground navigation in highly constrained spaces: Lessons learned from the second barn challenge at icra 2023 [competitions]," *IEEE Robotics & Automation Magazine*, vol. 30, no. 4, pp. 91–97, 2023.

[16] Y. Zhang, W. Macke, J. Cui, S. Hornstein, D. Urieli, and P. Stone, "Learning a robust multiagent driving policy for traffic congestion reduction," *Neural Computing and Applications*, pp. 1–14, 2023.

[17] P. R. Wurman *et al.*, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, 2022.

[18] P. Stone, L. Iocchi, F. Tonidandel, and C. Zhou, "Robocup 2021 worldwide: A successful robotics competition during a pandemic [competitions]," *IEEE Robotics & Automation Magazine*, vol. 28, no. 4, pp. 114–119, 2021.

[19] P. Leinen, M. Esders, K. T. Schütt, C. Wagner, K.-R. Müller, and F. S. Tautz, "Autonomous robotic nanofabrication with reinforcement learning," *Science advances*, vol. 6, no. 36, eabb6987, 2020.

[20] I. Masmitja *et al.*, "Dynamic robotic tracking of underwater targets using reinforcement learning," *Science robotics*, vol. 8, no. 80, eade7811, 2023.

[21] M. G. Bellemare *et al.*, "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, vol. 588, no. 7836, pp. 77–82, 2020.

[22] J. Degrave *et al.*, "Magnetic control of tokamak plasmas through deep reinforcement learning," *Nature*, vol. 602, no. 7897, pp. 414–419, 2022.

[23] T. Sugiyama, N. Schweighofer, and J. Izawa, "Reinforcement learning establishes a minimal metacognitive process to monitor and control motor learning performance," *Nature Communications*, vol. 14, no. 1, p. 3988, 2023.

[24] P Czop, G Kost, D Sławik, and G Wszołek, "Formulation and identification of first-principle data-driven models," *Journal of Achievements in materials and manufacturing Engineering*, vol. 44, no. 2, pp. 179–186, 2011.

[25] J. O. Maloney, *Perry Chemical Engineers Handbook*. The McGraw-Hill Companies, Inc, 2008.

[26] N. Sharma and Y. Liu, "A hybrid science-guided machine learning approach for modeling chemical processes: A review," *AIChE Journal*, vol. 68, no. 5, e17609, 2022.

[27] O. NELLES, *NONLINEAR SYSTEM IDENTIFICATION: From Classical Approaches to Neural Networks, Fuzzy Systems,... and Gaussian Processes*. SPRINGER NATURE, 2020.

[28] T. Brown *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[30] N. C. Frey *et al.*, "Neural scaling of deep chemical models," *Nature Machine Intelligence*, pp. 1–9, 2023.

[31] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[32] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[34] F. Pianosi *et al.*, "Sensitivity analysis of environmental models: A systematic review with practical workflow," *Environmental Modelling & Software*, vol. 79, pp. 214–232, 2016.

[35] B. Huang, Y. Qi, and A. M. Murshed, *Dynamic modeling and predictive control in solid oxide fuel cells: first principle and data-based approaches*. John Wiley & Sons, 2013.

[36] K. Godfrey, *Perturbation signals for system identification*. Prentice Hall International (UK) Ltd., 1993.

[37] H. M. Shariff, M. H. Fazalul Rahiman, and M. Tajjudin, "Nonlinear system identification: Comparison between prbs and random gaussian perturbation on steam distillation pilot plant," in *2013 IEEE 3rd International Conference on System Engineering and Technology*, 2013, pp. 269–274. DOI: 10.1109/ICSEngT.2013.6650183.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[39] F. A. Gers and E Schmidhuber, "Lstm recurrent networks learn simple context-free and context-sensitive languages," *IEEE transactions on neural networks*, vol. 12, no. 6, pp. 1333–1340, 2001.

[40] K. M. Powell, D. Machalek, and T. Quah, "Real-time optimization using reinforcement learning," *Computers & Chemical Engineering*, vol. 143, p. 107 077, 2020.

[41] M. Konishi, M. Inubushi, and S. Goto, "Fluid mixing optimization with reinforcement learning," *Scientific reports*, vol. 12, no. 1, p. 14 268, 2022.

[42] S. Nikita, A. Tiwari, D. Sonawat, H. Kodamana, and A. S. Rathore, "Reinforcement learning based optimization of process chromatography for continuous processing of biopharmaceuticals," *Chemical Engineering Science*, vol. 230, p. 116 171, 2021.

[43] H.-E. Byun, B. Kim, and J. H. Lee, "Embedding active learning in batch-to-batch optimization using reinforcement learning," *Automatica*, vol. 157, p. 111 260, 2023.

[44] H. Van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, and J. Modayil, "Deep reinforcement learning and the deadly triad," *arXiv preprint arXiv:1812.02648*, 2018.

[45] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.

[46] M. Hessel *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[47] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[48] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[49] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, PMLR, 2016, pp. 1928–1937.

[50] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[51] T. Haarnoja *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[52] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[53] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[54] A. Khan, F. Shahid, C. Maple, A. Ahmad, and G. Jeon, "Toward smart manufacturing using spiral digital twin framework and twinchain," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1359–1366, 2020.

[55] M. Grieves, "Digital twin: Manufacturing excellence through virtual factory replication. white paper, 2014," *Online:-03-01*,

[56] S. Yang, P. Navarathna, S. Ghosh, and B. W. Bequette, "Hybrid modeling in the era of smart manufacturing," *Computers & Chemical Engineering*, vol. 140, p. 106 874, 2020.

[57] R. Kender *et al.*, "Improving the load flexibility of industrial air separation units using a pressure-driven digital twin," *AIChE Journal*, vol. 68, no. 7, e17692, 2022.

[58] E. Glaessgen and D. Stargel, "The digital twin paradigm for future nasa and us air force vehicles," in *53rd AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference 20th AIAA/ASME/AHS adaptive structures conference 14th AIAA*, 2012, p. 1818.

[59] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the digital twin: A systematic literature review," *CIRP journal of manufacturing science and technology*, vol. 29, pp. 36–52, 2020.

[60] Q. Le, J. Feingold, W. Glandorf, J. Kent, R. Sherman, and J. K. Ferri, "Model-based systems engineering approaches to chemicals and materials manufacturing," *AIChE Journal*, e18114, 2023.

[61] M. Elsholkami and A. Elkamel, "General optimization model for the energy planning of industries including renewable energy: A case study on oil sands," *AIChE Journal*, vol. 63, no. 2, pp. 610–638, 2017.

[62] R. Kender *et al.*, "Reduced order modeling of a pressure column of an air separation unit using the dynamic edmister method," *Computers & Chemical Engineering*, vol. 174, p. 108 250, 2023.

[63] F. P. Knebel, R. Trevisan, G. S. do Nascimento, M. Abel, and J. A. Wickboldt, "A study on cloud and edge computing for the implementation of digital twins in the oil & gas industries," *Computers & Industrial Engineering*, vol. 182, p. 109 363, 2023.

[64] L. Kuang *et al.*, "Application and development trend of artificial intelligence in petroleum exploration and development," *Petroleum Exploration and Development*, vol. 48, no. 1, pp. 1–14, 2021.

[65] W. Xian, K. Yu, F. Han, L. Fang, D. He, and Q.-L. Han, "Advanced manufacturing in industry 5.0: A survey of key enabling technologies and future trends," *IEEE Transactions on Industrial Informatics*, 2023.

[66] T. H.-J. Uhlemann, C. Lehmann, and R. Steinhilper, "The digital twin: Realizing the cyber-physical production system for industry 4.0," *Procedia Cirp*, vol. 61, pp. 335–340, 2017.

[67] C. Chen, Z. Zhou, and G. M. Bollas, "Dynamic modeling, simulation and optimization of a subcritical steam power plant. part i: Plant model and regulatory control," *Energy conversion and management*, vol. 145, pp. 324–334, 2017.

[68] Y. Qin, X. Wu, and J. Luo, "Data-model combined driven digital twin of lifecycle rolling bearing," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 1530–1540, 2021.

[69] J. Yu, P. Liu, and Z. Li, "Data reconciliation-based simulation of thermal power plants for performance estimation and digital twin development," *Computers & Chemical Engineering*, vol. 168, p. 108 063, 2022.

[70] R. Sacks, I. Brilakis, E. Pikas, H. S. Xie, and M. Girolami, "Construction with digital twin information systems," *Data-Centric Engineering*, vol. 1, e14, 2020.

[71] H. D. Perez, J. M. Wassick, and I. E. Grossmann, "A digital twin framework for online optimization of supply chain business processes," *Computers & Chemical Engineering*, vol. 166, p. 107 972, 2022.

[72] Y. Yang, M. Yang, N. Anwer, B. Eynard, L. Shu, and J. Xiao, "A novel digital twin-assisted prediction approach for optimum rescheduling in high-efficient flexible production workshops," *Computers & Industrial Engineering*, p. 109 398, 2023.

[73] J. Gibson and D. Peters, "Water and environmental management in oil sands regions," *Journal of Hydrology: Regional Studies*, vol. 44, p. 101 274, 2022.

[74] R. Meyer and E. Attanasi, *4. natural bitumen and extra-heavy oil: 2010 survey of energy resources, world energy council*, 2010.

[75] J. Wang, L. Feng, M. Steve, X. Tang, T. E. Gail, and H. Mikael, "China's unconventional oil: A review of its resources and outlook for long-term production," *Energy*, vol. 82, pp. 31–42, 2015.

[76] A. de Klerk, "Unconventional oil: Oilsands," in *Future energy*, Elsevier, 2020, pp. 49–65.

[77] M. R. Gray, *Upgrading oilsands bitumen and heavy oil*. University of Alberta, 2015.

[78] J. H. Masliyah, T. K. Kwong, and F. A. Seyer, "Theoretical and experimental studies of a gravity separation vessel," *Industrial & Engineering Chemistry Process Design and Development*, vol. 20, no. 1, pp. 154–160, 1981.

[79] W. A. Gilbert, "Dynamic simulation and optimal trajectory planning for an oilsand primary separation vessel," 2004.

[80] A. Betancourt-Torcat, G. Gutierrez, A. Elkamel, and L. Ricardez-Sandoval, "Integrated energy optimization model for oil sands operations," *Industrial & engineering chemistry research*, vol. 50, no. 22, pp. 12 641–12 663, 2011.

[81] M. Elsholkami, A. Elkamel, and F. Vargas, "Optimized integration of renewable energy technologies into alberta's oil sands industry," *Computers & Chemical Engineering*, vol. 90, pp. 1–22, 2016.

[82] D. Voskov, R. Zaydullin, and A. Lucia, "Heavy oil recovery efficiency using sagd, sagd with propane co-injection and strip-sagd," *Computers & Chemical Engineering*, vol. 88, pp. 115–125, 2016.

[83] O. S. Magazine, "Oil sands 101: Process overview," *Dostupné z http://www. oilsandsmagazine. com/technical/oilsands-101 (staženo 30. 4. 2017)*, 2020.

[84] H. Shafi, K. Velswamy, F. Ibrahim, and B. Huang, "A hierarchical constrained reinforcement learning for optimization of bitumen recovery rate in a primary separation vessel," *Computers & Chemical Engineering*, vol. 140, p. 106 939, 2020.

[85] S. Liu, J. Zhang, and J. Liu, "Economic mpc with terminal cost and application to an oilsand primary separation vessel," *Chemical Engineering Science*, vol. 136, pp. 27–37, 2015.

[86] A. Vicente *et al.*, "Computer vision system for froth-middlings interface level detection in the primary separation vessels," *Computers & Chemical Engineering*, vol. 123, pp. 357–370, 2019.

[87] J. Xie, O. Dogru, B. Huang, C. Godwaldt, and B. Willms, "Reinforcement learning for soft sensor design through autonomous cross-domain data selection," *Computers & Chemical Engineering*, vol. 173, p. 108 209, 2023.

[88] W. de Paula Ferreira, F. Armellini, and L. A. De Santa-Eulalia, "Simulation in industry 4.0: A state-of-the-art review," *Computers & Industrial Engineering*, vol. 149, p. 106 868, 2020.

[89] R. S. Sanders, A. L. Ferre, W. B. Maciejewski, R. G. Gillies, and C. A. Shook, "Bitumen effects on pipeline hydraulics during oil sand hydrotransport," *The Canadian Journal of Chemical Engineering*, vol. 78, no. 4, pp. 731–742, 2000.

[90] J. Z. Zhou, H. Li, R. S. Chow, Q. Liu, Z. Xu, and J. Masliyah, "Role of mineral flotation technology in improving bitumen extraction from mined athabasca oil sands—ii. flotation hydrodynamics of water-based oil sand extraction," *The Canadian Journal of Chemical Engineering*, vol. 98, no. 1, pp. 330–352, 2020.

[91] Q. Chen and Q. Liu, "Bitumen coating on oil sands clay minerals: A review," *Energy & Fuels*, vol. 33, no. 7, pp. 5933–5943, 2019.

[92] A. Li *et al.*, "The effect of clay type and solid wettability on bitumen extraction from canadian oil sands," *Fuel*, vol. 337, p. 126 887, 2023.

[93] Y. Yu, L. Ma, M. Cao, and Q. Liu, "Slime coatings in froth flotation: A review," *Minerals Engineering*, vol. 114, pp. 26–36, 2017.

[94] J. H. Masliyah, J. Czarnecki, and Z. Xu, "Handbook on theory and practice on bitumen recovery from athabasca oil sands," 2011.

[95] M. Wan *et al.*, "Hydrodynamics in a gravity settling vessel: Cfd modelling with lda validation," *The Canadian Journal of Chemical Engineering*, vol. 78, no. 6, pp. 1046–1055, 2000.

[96] J Tyler, J Spence, D Kiel, J Schaan, and G Larson, "The use of physical modeling in the optimisation of a primary separation vessel feedwell," *The Canadian Journal of Chemical Engineering*, vol. 87, no. 6, pp. 821–831, 2009.

[97] G. B. Wallis, *One-dimensional two-phase flow*. Courier Dover Publications, 2020.

[98] J. H. Masliyah, "Hindered settling in a multi-species particle system," *Chemical Engineering Science*, vol. 34, no. 9, pp. 1166–1168, 1979.

[99] F Concha and E. Almendra, "Settling velocities of particulate systems, 2. settling velocities of suspensions of spherical particles," *International Journal of Mineral Processing*, vol. 6, no. 1, pp. 31–41, 1979.

[100] J. Richardson and W. Zaki, "Sedimentation and fluidisation: Part i," *Chemical Engineering Research and Design*, vol. 75, S82–S100, 1997.

[101] J. M. Ortiz and R. T. Olmedo, "A flotation cell model for dynamic simulation," *IFAC Proceedings Volumes*, vol. 46, no. 16, pp. 161–165, 2013.

[102] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[103] T. Zhang, M. Afshari, P. Musilek, M. E. Taylor, and O. Ardakanian, "Diversity for transfer in learning-based control of buildings," in *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, 2022, pp. 556–564.

[104] Z. Zou, X. Yu, and S. Ergan, "Towards optimal control of air handling units using deep reinforcement learning and recurrent neural network," *Building and Environment*, vol. 168, p. 106 535, 2020.

[105] H. Hassanpour, P. Mhaskar, and B. Corbett, "A practically implementable reinforcement learning control approach by leveraging offset-free model predictive control," *Computers & Chemical Engineering*, p. 108 511, 2023.

[106] T. A. Mendiola-Rodriguez and L. A. Ricardez-Sandoval, "Robust control for anaerobic digestion systems of tequila vinasses under uncertainty: A deep deterministic policy gradient algorithm," *Digital Chemical Engineering*, vol. 3, p. 100 023, 2022.

[107] P. Zhao *et al.*, "Intelligent injection molding on sensing, optimization, and control," *Advances in Polymer Technology*, vol. 2020, pp. 1–22, 2020.

[108] R. Crites and A. Barto, "Improving elevator performance using reinforcement learning," *Advances in neural information processing systems*, vol. 8, 1995.

[109] P. Balaji, X German, and D. Srinivasan, "Urban traffic signal control using reinforcement learning agents," *IET Intelligent Transport Systems*, vol. 4, no. 3, pp. 177–188, 2010.

[110] X. Deng, G. Hu, and W. Chen, "Intelligent active flow control of long-span bridge deck using deep reinforcement learning integrated transfer learning," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 244, p. 105 632, 2024.

[111] R. Iwami, T. Mihana, K. Kanno, S. Sunada, M. Naruse, and A. Uchida, "Controlling chaotic itinerancy in laser dynamics for reinforcement learning," *Science Advances*, vol. 8, no. 49, eabn8325, 2022.

[112] O. Dogru *et al.*, "Reinforcement learning approach to autonomous pid tuning," *Computers & Chemical Engineering*, vol. 161, p. 107 760, 2022.

[113] S. Gros and M. Zanon, "Data-driven economic nmpc using reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 636–648, 2019.

[114] E. Hedrick, K. Hedrick, D. Bhattacharyya, S. E. Zitney, and B. Omell, "Reinforcement learning for online adaptation of model predictive controllers: Application to a selective catalytic reduction unit," *Computers & Chemical Engineering*, vol. 160, p. 107 727, 2022.

[115] E. Altman, *Constrained Markov decision processes*. Routledge, 2021.

[116] M. Bowling, J. D. Martin, D. Abel, and W. Dabney, "Settling the reward hypothesis," in *International Conference on Machine Learning*, PMLR, 2023, pp. 3003–3020.

[117] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International conference on machine learning*, PMLR, 2017, pp. 22–31.

[118] Y. Efroni, S. Mannor, and M. Pirotta, "Exploration-exploitation in constrained mdps," *arXiv preprint arXiv:2003.02189*, 2020.

[119] L. Zheng and L. Ratliff, "Constrained upper confidence reinforcement learning," in *Learning for Dynamics and Control*, PMLR, 2020, pp. 620–629.

[120] T. D. Simão, N. Jansen, and M. T. Spaan, "Alwayssafe: Reinforcement learning without safety constraint violations during training," 2021.

[121] Q. Gao, H. Yang, S. M. Shanbhag, and A. M. Schweidtmann, "Transfer learning for process design with reinforcement learning," *arXiv preprint arXiv:2302.03375*, 2023.

[122] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.

[123] D. Silver, J. Bagnell, and A. Stentz, "High performance outdoor navigation from overhead data using imitation learning," *Robotics: Science and Systems IV, Zurich, Switzerland*, vol. 1, 2008.

[124] P. Chaysri, C. Spatharis, K. Blekas, and K. Vlachos, "Unmanned surface vehicle navigation through generative adversarial imitation learning," *Ocean Engineering*, vol. 282, p. 114 989, 2023.

[125] A. K. GS, T. Zhang, O. Ardakanian, and M. E. Taylor, "Mitigating an adoption barrier of reinforcement learning-based control strategies in buildings," *Energy and Buildings*, vol. 285, p. 112 878, 2023.

[126] S. Höfer *et al.*, "Sim2real in robotics and automation: Applications and challenges," *IEEE transactions on automation science and engineering*, vol. 18, no. 2, pp. 398–400, 2021.

[127] S. Brandi, M. S. Piscitelli, M. Martellacci, and A. Capozzoli, "Deep reinforcement learning to optimise indoor temperature control and heating energy consumption in buildings," *Energy and Buildings*, vol. 224, p. 110 225, 2020.

[128] Y. Shi *et al.*, "Dual-mode fast dmc algorithm for the control of orc based waste heat recovery system," *Energy*, vol. 244, p. 122 664, 2022.

[129] R. Y. Tao, A. White, and M. C. Machado, "Agent-state construction with auxiliary inputs," *arXiv preprint arXiv:2211.07805*, 2022.

[130] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[131] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat, "State representation learning for control: An overview," *Neural Networks*, vol. 108, pp. 379–392, 2018.

[132] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[133] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[134] Z. Abbas, R. Zhao, J. Modayil, A. White, and M. C. Machado, "Loss of plasticity in continual deep reinforcement learning," *arXiv preprint arXiv:2303.07507*, 2023.

[135] S. Dohare, R. S. Sutton, and A. R. Mahmood, "Continual backprop: Stochastic gradient descent with persistent randomness," *arXiv preprint arXiv:2108.06325*, 2021.

[136] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 4693–4700.

[137] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[138] A. Rajeswaran *et al.*, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[139] B. Piot, M. Geist, and O. Pietquin, "Bridging the gap between imitation learning and inverse reinforcement learning," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 8, pp. 1814–1826, 2016.

[140] D. Brown, W. Goo, P. Nagarajan, and S. Niekum, "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations," in *International conference on machine learning*, PMLR, 2019, pp. 783–792.

[141] E. Oztemel and S. Gursev, "Literature review of industry 4.0 and related technologies," *Journal of intelligent manufacturing*, vol. 31, pp. 127–182, 2020.

[142] L. Belli, L. Davoli, A. Medioli, P. L. Marchini, and G. Ferrari, "Toward industry 4.0 with iot: Optimizing business processes in an evolving manufacturing factory," *Frontiers in ICT*, vol. 6, p. 17, 2019.

[143] G. D. Patrón and L. Ricardez-Sandoval, "An integrated real-time optimization, control, and estimation scheme for post-combustion co2 capture," *Applied Energy*, vol. 308, p. 118 302, 2022.

[144] Q. Min, Y. Lu, Z. Liu, C. Su, and B. Wang, "Machine learning based digital twin framework for production optimization in petrochemical industry," *International Journal of Information Management*, vol. 49, pp. 502–519, 2019.

[145] H. Zhang, Q. Liu, X. Chen, D. Zhang, and J. Leng, "A digital twin-based approach for designing and multi-objective optimization of hollow glass production line," *Ieee Access*, vol. 5, pp. 26 901–26 911, 2017.

[146] Y. Cohen and G. Singer, "A smart process controller framework for industry 4.0 settings," *Journal of Intelligent Manufacturing*, vol. 32, no. 7, pp. 1975–1995, 2021.

[147] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[148] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[149] 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.

[150] A. S. Badwe, R. D. Gudi, R. S. Patwardhan, S. L. Shah, and S. C. Patwardhan, "Detection of model-plant mismatch in mpc applications," *Journal of Process Control*, vol. 19, no. 8, pp. 1305–1313, 2009.

[151] L. D. Tufa and C. Z. Ka, "Effect of model plant mismatch on mpc performance and mismatch threshold determination," *Procedia engineering*, vol. 148, pp. 1008–1014, 2016.

[152] P. P. Bonissone, F. Xue, and R. Subbu, "Fast meta-models for local fusion of multiple predictive models," *Applied Soft Computing*, vol. 11, no. 2, pp. 1529–1539, 2011.

[153] C. J. Tomlin, J. Lygeros, and S. S. Sastry, "A game theoretic approach to controller design for hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 949–970, 2000.

[154] H. Hu, K. Ou, and W.-W. Yuan, "Fused multi-model predictive control with adaptive compensation for proton exchange membrane fuel cell air supply system," *Energy*, vol. 284, p. 128 459, 2023.

[155] R. Dazeley, P. Vamplew, and F. Cruz, "Explainable reinforcement learning for broad-xai: A conceptual framework and survey," *Neural Computing and Applications*, pp. 1–24, 2023.

[156] R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman, "Metrics for explainable ai: Challenges and prospects," *arXiv preprint arXiv:1812.04608*, 2018.

[157] L. Wells and T. Bednarz, "Explainable ai and reinforcement learning—a systematic review of current approaches and trends," *Frontiers in artificial intelligence*, vol. 4, p. 550 030, 2021.

[158] S. Milani, N. Topin, M. Veloso, and F. Fang, "A survey of explainable reinforcement learning," *arXiv preprint arXiv:2202.08434*, 2022.

[159] J. Hoffmann and D. Magazzeni, "Explainable ai planning (xaip): Overview and the case of contrastive explanation," *Reasoning Web. Explainable Artificial Intelligence: 15th International Summer School 2019, Bolzano, Italy, September 20–24, 2019, Tutorial Lectures*, pp. 277–282, 2019.

[160] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (xai)," *IEEE access*, vol. 6, pp. 52 138–52 160, 2018.

[161] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, "Explainable reinforcement learning through a causal lens," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 2493–2500.

[162] M. Mardani, H. Monajemi, V. Papyan, S. Vasanawala, D. Donoho, and J. Pauly, "Recurrent generative adversarial networks for proximal learning and automated compressive image recovery," *arXiv preprint arXiv:1711.10046*, 2017.

[163] H. Ritschel, "Socially-aware reinforcement learning for personalized human-robot interaction," 2018.

[164] M. Pieters and M. A. Wiering, "Q-learning with experience replay in a dynamic environment," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 2016, pp. 1–8.

[165] S. Rathi, "Generating counterfactual and contrastive explanations using shap," *arXiv preprint arXiv:1906.09293*, 2019.

[166] S. Milani, N. Topin, M. Veloso, and F. Fang, "A survey of explainable reinforcement learning," *arXiv preprint arXiv:2202.08434*, 2022.

[167] T. A. Hester, E. J. Fisher, and P. Khandelwal, *Predictively controlling an environmental control system*, US Patent 9,869,484, 2018.

[168] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, Citeseer, vol. 99, 1999, pp. 278–287.