Interpretable Approaches for Condition Monitoring, Control, and Energy Optimization in Electro-Hydrostatic Actuation Systems

A Thesis

Submitted to the Faculty of Graduate and Postdoctoral Studies

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

Soleiman Hosseinpour

Department of Mechanical Engineering

University of Manitoba

Winnipeg, MB, Canada

Copyright © 2026 by Soleiman Hosseinpour

Abstract

Electro-hydrostatic actuator (EHA) systems, as a class of hydraulic actuators, are increasingly used in modern heavy-duty robotics and aircraft applications due to their compactness, efficiency, and high power density. However, their performance is strongly affected by system uncertainties, faults, and changing operating conditions. Based on a Bayesian framework, this thesis develops interpretable approaches for condition monitoring, control, and energy optimization. This thesis builds upon a newly developed EHA system that includes an energy storage and reutilization (ESR) unit.

In the context of condition monitoring, first, a Bayesian classification framework is proposed to distinguish between healthy and faulty operating conditions using a feature extraction method based on fractal complexity measures. To address fault detection under limited data availability, the sparse identification of nonlinear dynamics (SINDy) method is further developed, where Bayesian optimization is used for tuning of its model parameters.

On the control front, the limitations and challenges of the model-based controller are investigated. To circumvent these challenges, a model-free learning strategy is introduced, consisting of offline and online phases. The controller parameters are initially tuned during the offline phase using a method known as augmented random search. The proposed method enables the transfer of offline knowledge to the online phase by incorporating Gaussian process modelling, allowing for adaptive tuning of the controller parameters under changing system dynamics.

With respect to energy optimization, the thesis investigates a multi-objective optimization framework to balance the tracking performance and energy consumption in the EHA system equipped with the ESR unit.

In contrast to black-box learning approaches such as neural networks, the methods developed in this thesis preserve interpretability and provide explainable insights into system behaviour, which are highly desirable in industrial engineering applications.

Contents

Abstract **i**

List of Acronyms **xii**

List of Symbols **xiv**

1 Introduction **1**
1.1 Problem Statement 1
1.2 Research Objectives 2
1.3 Proposed Methodologies 3
1.4 Organization of the Thesis 4

2 Description of the Hydraulic System Under Investigation **5**
2.1 Electro-Hydrostatic Actuator System 5
2.2 Energy Storage and Reutilization Unit 11
2.3 Summary 17

3 Preliminaries **18**
3.1 Bayesian Framework 18
3.1.1 Bayesian Inference 19
3.1.2 Gaussian Processes 20
3.1.3 Bayesian Optimization 21
3.2 Fractals 22
3.2.1 Fractal Dimension and Complexity Measures 23
3.2.2 Self-Affine Time Series 24
3.2.3 Length and Variance Fractal Dimensions 24
3.3 Optimization-Based PID Controller Tuning 28
3.4 Summary 31

4 Condition Monitoring **33**
4.1 Fault Detection Using Bayesian Classification . . . 34
4.1.1 Data Acquisition . . . 36
4.1.2 Noise in Pressure Reading and Denoising Process . . . 39
4.1.3 Feature Extraction . . . 41
4.1.4 Bayesian Classification . . . 45
4.1.5 Posterior Distribution Analysis . . . 47
4.2 Low-Data Fault Detection Using SINDy . . . 54
4.2.1 SINDy Algorithm . . . 55
4.2.2 Bayesian Optimization for SINDy Parameter Tuning . . . 57
4.2.3 Demonstration Results . . . 58
4.2.4 Experimental Validation . . . 60
4.2.5 Application to Fault Detection . . . 63
4.3 Summary . . . 71

5 Controller Tuning **73**
5.1 Offline Model-Based Controller Tuning . . . 73
5.1.1 Design Specifications . . . 74
5.1.2 Optimization Approach . . . 77
5.1.3 Results and Discussions . . . 78
5.2 Model-Free Learning for Online PID Controller Tuning . . . 81
5.2.1 Controller Structure . . . 82
5.2.2 Offline Tuning Phase . . . 83
5.2.3 Online Tuning Phase . . . 92
5.3 Summary . . . 98

6 Energy Optimization **100**
6.1 Objective and Framework . . . 101
6.1.1 Multi-objective Optimization Problem Formulation . . . 104
6.1.2 Gaussian Process–Based Initialization . . . 106
6.1.3 Results . . . 109
6.2 Summary . . . 116

7 Concluding Remarks **118**
7.1 Thesis Conclusion . . . 118
7.2 Contributions Made in the Thesis . . . 119
7.3 Future Work . . . 119

References **121**

Appendix/MATLAB/Simulink Models **131**

A.1 EHA Simulink Model 131

A.2 ESR Simulink Model 137

A.3 Simulink Model for Model-Free Learning Control 144

A.4 Simulink Model for Energy Optimization 144

List of Tables

2.1 Parameters of electro-hydrostatic actuator (EHA) system. 11
2.2 Benedict–Webb–Rubin (BWR) coefficients 14

4.1 Mean and standard deviation of variance fractal dimension (VFD) values for actuator pressures, P_A and P_B, across different sampling intervals. H and F stand for healthy and faulty signals. 45
4.2 Statistics for Bayesian parameter estimation with 200,000 Markov chain Monte Carlo (MCMC) iterations. 47
4.3 Performance metrics of Bayesian classifier with 200,000 MCMC iterations and three-fold cross-validation compared with the support vector machine (SVM) method. 52
4.4 Sparse identification of nonlinear dynamics (SINDy) models based on actuator displacement x_p, actuator velocity v_p, control input signal u, and corresponding optimal Bayesian parameters. 64
4.5 Summary of test datasets used for model validation. 66
4.6 Comparison of the performance metrics for the fault detection using SINDy models with and without using Bayesian optimization. 68
4.7 Integral absolute error (IAE) values for four unseen experimental datasets across three SINDy models. Minimum values are highlighted in bold. 70

5.1 Transient response t_s and overshoot (OS) for different specifications and simulation results. 79
5.2 Comparison of the results for transient response t_s and OS for different specifications and simulation results. 80
5.3 Parameters used in the augmented random search (ARS) and Gaussian process (GP) during the offline tuning phase. 88
5.4 Parameters of the reference signal. 93
5.5 Parameters used in the online tuning phase. 96

6.1 Key parameters used in the GP-based initialization and multi-objective genetic algorithm. 109
6.2 Pareto-optimal solutions for the EHA system with and without an energy storage and reutilization (ESR) unit in the low-error regime ($J_{\text{error}} < 1$). Bold entries indicate the minimum energy achieved by each system within this regime. 111
6.3 Pareto-optimal solutions in the moderate-error regime ($1 < J_{\text{error}} < 3$). Bold entries indicate the minimum energy achieved by each system within this regime. 112
6.4 Controller parameters corresponding to the minimum-energy solutions within each tracking error range. The reported error corresponds to the solution achieving the minimum energy within the specified range. 112

List of Figures

2.1 Typical backhoe loader (adapted from [9]). 6
2.2 Backhoe arm setup [4]. 6
2.3 Schematic of the EHA circuit used in the experimental setup: (1) actuator; (2) mass of piston and rod; (3, 4) check valves; (5, 6) relief valves; (7) bidirectional pump PM_1; (8) servomotor; (9) auxiliary pump-motor; (10) relief valve; (11) tank; (12) directional valve (V1); (13) flow control valve (V2). 7
2.4 Four-quadrant operation of the EHA circuit. 8
2.5 Experimental setup of the energy storage and reutilization (ESR) system applied to the EHA (copied from [10]). 12
2.6 ESR system applied to the EHA system: (12) pump motor 2; (13) flow discharge valve; (14) tank; (15) electro-proportional pressure reducing valve (EPPRV); (16) hydraulic accumulator; (17), (18), (19) check valves; (20) relief valve; (21) bleed-off valve; (V3) float-centre type three-position four-way valve; (V4) reutilization valve. 12
2.7 Energy storage and reutilization flowchart [10]. 13

3.1 Generated Brownian self-affine time series. 26
3.2 Log–log plot used for estimating the length fractal dimension. 27
3.3 Log–log plot used for estimating the variance fractal dimension. 27

4.1 Abbreviations used for the datasets. 37
4.2 Typical test results for dataset OMNJo: (a) input signal u; (b) position x_p; (c) velocity $\dot{x}_p$; (d) leakage level; (e) measured pressure P_A; (f) measured pressure P_B. 37
4.3 Measured pressure signal P_A for various datasets: (a) OMNJo; (b) OMSJo; (c) OMMJo; (d) OMHJo. 38
4.4 Measured pressure signal P_B for various datasets: (a) OMNJo; (b) OMSJo; (c) OMMJo; (d) OMHJo. 38
4.5 Typical fault levels: (a) no leakage dataset OMNJo; (b) small leakage dataset OMSJo; (c) medium leakage dataset OMMJo; (d) high leakage dataset OMHJo. . . 39

4.6 Power spectral density analysis of pressure signal P_A, for datasets OMNJo, CMNJo, and CNNJo. 40
4.7 Denoised pressure signal P_A: (a) OMNJo; (b) OMSJo; (c) OMMJo; (d) OMHJo. . 40
4.8 Denoised pressure signal P_B: (a) OMNJo; (b) OMSJo; (c) OMMJo; (d) OMHJo. . 41
4.9 Scatter plots of variance fractal dimension (VFD) for pressure signals P_A and P_B, and velocity $\dot{x}_p$. The target value of zero shows healthy, and one shows faulty signals. 43
4.10 Log–log plot of $V_{k,\text{ave}}$ versus n_k for pressure signal P_A, in dataset OMNJo. 43
4.11 Log–log plot of $V_{k,\text{ave}}$ versus n_k for pressure signal P_B, in dataset OMNJo. 44
4.12 VFD of the pressure signal P_A, for different levels of leakage. 44
4.13 VFD of the pressure signal P_B for different levels of leakage. 45
4.14 Trace plots of Markov chain Monte Carlo (MCMC) for Bayesian parameter β_0. . . 48
4.15 Trace plots of MCMC for Bayesian parameter β_1. 48
4.16 Trace plots of MCMC for Bayesian parameter β_2. 48
4.17 Density functions for parameter β_0. 49
4.18 Density functions for parameter β_1. 49
4.19 Density functions for parameter β_2. 49
4.20 Auto-correlation function (ACF) plots for parameter β_0. 50
4.21 ACF plots for parameter β_1. 50
4.22 ACF plots for parameter β_2. 51
4.23 Confusion matrix for one of the three-fold cross-validation. 51
4.24 Bayesian (dashed line) and support vector machine (SVM) (solid line) classifiers on testing datasets. 53
4.25 Comparison between simulation and experimental results. 58
4.26 Comparison of the results for training the sparse identification of nonlinear dynamics (SINDy) algorithm using data obtained from the simulation model: (a) control input signal, u; (b) actuator displacement, x_p; (c) actuator velocity, v_p. 59
4.27 Comparison of the results for testing the unseen data using simulation data (trial 1): (a) control input signal, u; (b) actuator displacement, x_p; (c) actuator velocity, v_p. . 60
4.28 Comparison of the results for testing the unseen data using simulation data (trial 2): (a) control input signal, u; (b) actuator displacement, x_p; (c) actuator velocity, v_p. . 60
4.29 Comparison of the results for training the SINDy algorithm using experimental data: (a) control input signal, u; (b) actuator displacement, x_p; (c) actuator velocity, v_p. . 62
4.30 Comparison of the results for testing the unseen data using experimental data (trial 1): (a) control input signal, u; (b) actuator displacement, x_p; (c) actuator velocity, v_p. . 62
4.31 Comparison of the results for testing the unseen data using experimental data (trial 2): (a) control input signal, u; (b) actuator displacement, x_p; (c) actuator velocity, v_p. . 63

4.32 Experimental results for a typical dataset with a high leakage level. 64
4.33 Different leakage levels: (a) no leakage; (b) small leakage; (c) medium leakage; (d) high leakage. 65
4.34 Confusion matrix for all 45 unseen experimental datasets using SINDy models with Bayesian-optimized parameters. Positive indicates faulty, and negative indicates healthy. 67
4.35 (a) Control input signal, u; (b) comparison of the actuator displacement, x_p, for unseen no leakage, medium load dataset, and SINDy models. 68
4.36 (a) Control input signal, u; (b) comparison of the actuator displacement, x_p, for unseen small leakage, medium load dataset and SINDy models. 69
4.37 (a) Control input signal, u; (b) comparison of the actuator displacement, x_p, for unseen medium leakage, medium load dataset and SINDy models. 69
4.38 (a) Control input signal, u; (b) comparison of the actuator displacement, x_p, for unseen high leakage, medium load dataset and SINDy models. 70

5.1 System response for different specifications of the transfer function $G_1(s)$. 79
5.2 System response for different specifications of the transfer function $G_2(s)$. 79
5.3 Comparison of system responses for $G_3(s)$ using the proposed method, Ziegler–Nichols, and Genetic algorithm. 80
5.4 Comparison of system responses for $G_4(s)$ using the proposed method, Ziegler–Nichols, and Genetic algorithm. 81
5.5 Closed-loop tracking response with the best gain vector $K_{\text{best}} = [47.30, 40.30, 14.11]^T$: actuator position response, x_p compared with the reference signal. 86
5.6 Velocity response, $\dot{x}_p$ obtained with the optimal gain vector $K_{\text{best}} = [47.30, 40.30, 14.11]^T$. 86
5.7 Control input signal, u generated by the PID controller using the best gain vector $K_{\text{best}} = [47.30;;40.30;;14.11]^T$. 87
5.8 Tracking error obtained with the best gain vector $K_{\text{best}} = [47.30, 40.30, 14.11]^T$. . . 87
5.9 Evolution of the gains (K_p, K_i, K_d) over augmented random search (ARS) iterations. Iteration 0 corresponds to the best controller gains from the initial random design. . 87
5.10 Return per ARS iteration after the initial random design; iteration 0 corresponds to the best controller obtained from the initial random design before ARS updates begin. 88
5.11 Validation of the Gaussian process (GP) model on unseen gain vectors after the offline phase: scatter plot of GP-predicted return versus the true return obtained from simulations. 89
5.12 One-dimensional slice of the GP model versus the gain K_p. 90
5.13 One-dimensional slice of the GP model versus the gain K_i. 90

5.14 One-dimensional slice of the GP model versus the gain K_d. 90
5.15 Tracking response of the electro-hydrostatic actuator (EHA) system: (a) step response; (b) square-wave response; (c) sinusoidal response. 91
5.16 Desired position trajectory used for the online evaluation over the entire simulation. The vertical dashed lines indicate the boundaries between the 12 non-identical consecutive episodes, each with a duration of 50 s. 92
5.17 Evolution of PID controller gains (K_p, K_i, K_d) across episodes. After the change in β_e, the algorithm explores the gain space for three episodes and terminates exploration once improved gains are identified. 97
5.18 Instantaneous and best online rewards during the online tuning process. The vertical dashed line marks the episode where exploration is terminated. 97
5.19 Tracking response of the EHA system over the entire simulation. The dashed curve represents the reference trajectory, while the solid curve shows the system response. The first vertical dashed line indicates the change in the system parameter β_e. The second vertical dashed line marks the episode at which exploration is terminated, and the controller gains are fixed. 98
5.20 Control signal applied to the EHA system. Vertical dashed lines indicate the change in the system parameter β_e and the episode at which gain exploration is terminated. 98

6.1 The control logic for the proposed energy storage and reutilization (ESR) strategy. 102
6.2 Activation map of storage and reutilization modes as a function of accumulator state of charge (SOC_p). Colored regions indicate storage or reutilization status (on/off), and dashed lines represent switching thresholds. 102
6.3 Actuator position response, x_p and reference signal r. 110
6.4 Control input, u. 110
6.5 Actuator rod velocity, $\dot{x}_p$. 111
6.6 Energy mode indicators: (a) storage-mode activation; (b) regeneration activation. . 113
6.7 The value of the state of charge of accumulator SOC_p. 113
6.8 Accumulator pressure, P_{acc} and electro-proportional pressure reducing valve (EP-PRV) outlet pressure, P_{pprv}. 114
6.9 Load pressure, P_L. 114
6.10 Servomotor response: (a) motor current I; (b) motor angular velocity ω. 115
6.11 Torque responses: (a) servomotor torque, T_e; (b) load torque, T_L; (c) pump–motor torque, T_{PM_2}. 115
6.12 Flow responses: (a) hydraulic chamber flows, including cap-side flow Q_{ac} and rod-side flow, Q_{bc}; (b) accumulator flow rate, Q_{acc}; (c) pump flow rate, Q_{PM_2}. . . . 116

6.13 Instantaneous servomotor power. 116

A.1 Simulink model of the electro-hydrostatic actuator (EHA) system. 131
A.2 Control subsystems: (a) EHA system; (b) magnified view of the torque control mode; (c) magnified view of the velocity control mode. 132
A.3 Flow and pressure subsystem of the EHA Simulink model. 133
A.4 Actuator dynamics subsystem: (a) EHA system; (b) magnified view of the displacement; velocity, and load pressure subsystem. 134
A.5 Quadrant subsystem of the EHA Simulink model. 135
A.6 Control input signal and corresponding actuator response: (a) input signal; (b) actuator position. 136
A.7 System responses: (a) actuator velocity response; (b) chamber pressures P_A and P_B; (c) load torque T_L and motor torque T_e. 136
A.8 System responses: (a) motor current; (b) flow rates of pump PM_1; (c) flow rates Q_{ac} and Q_{bc}. 137
A.9 Simulink model of the ESR attached to the EHA system. 137
A.10 Simulink model of the control subsystem for ESR activation. 138
A.11 ESR system: (a) ESR system; (b) magnified view of the ESR system. 139
A.12 Torque control subsystem: (a) ESR system; (b) magnified view of the torque control subsystem. 140
A.13 Additional subsystems: (a) quadrant block; (b) displacement/velocity/load pressure; (c) flow and pressure dynamic of the EHA. 141
A.14 Control input signal and corresponding actuator response: (a) input signal; (b) actuator position. 142
A.15 System responses: (a) modified control input; (b) actuator velocity response; (c) chamber pressures P_A and P_B. 142
A.16 System responses: (a) load torque T_L, motor torque T_e, and pump torque T_{pm}; (b) motor current; (c) flow rates of pump PM_1, accumulator Q_{acc}, and pump PM_2. . . . 143
A.17 System responses: (a) flow rates Q_{ac} and Q_{bc}; (b) accumulator pressure P_{acc}; (c) state of charge (SOC_p) of the accumulator. 143
A.18 Simulink model for model-free learning control 144
A.19 Simulink model of the energy optimization system. 144
A.20 Magnified view of the optimization subsystem. 145
A.21 Magnified view of the ESR and the PID controller subsystems. 146

List of Acronyms

Acronym	Definition
ACF	Autocorrelation function
ADC	Analog-to-digital converter
ARS	Augmented random search
BO	Bayesian optimization
BWR	Benedict–Webb–Rubin (equation)
DSP	Digital signal processing
EHA	Electro-hydrostatic actuator
EI	Expected improvement
EMF	Electromotive force
EPPRV	Electro-proportional pressure reducing valve
ESR	Energy storage and reutilization
FN	False negative
FP	False positive
GP	Gaussian process
IAE	Integral of absolute error
IRLS	Iteratively reweighted least squares
LASSO	Least absolute shrinkage and selection operator
MCMC	Markov chain Monte Carlo
M–H	Metropolis–Hastings
NRMSE	Normalized root mean square error

OS	Overshoot
PI	Proportional–integral
PID	Proportional–integral–derivative
PM_1	Pump motor 1
PM_2	Pump motor 2
PSD	Power spectral density
RBF	Radial basis function
RL	Reinforcement learning
RMSE	Root mean square error
RTH	Reutilization threshold
SD	Standard deviation
SINDy	Sparse identification of nonlinear dynamics
SOC_p	State of charge based on pressure
SOC_v	State of charge based on volume
SQP	Sequential quadratic programming
STH	Storage threshold
SVM	Support vector machine
TN	True negative
TP	True positive
UCB	Upper confidence bound
VFD	Variance fractal dimension

List of Symbols

Symbol	Definition
A_a	Piston area (m^2)
A_b	Rod-side area (m^2)
B_e	Motor viscous damping coefficient (N m s/rad)
C_{vi}	Valve flow coefficient ($m^3/(s\,Pa^{1/2})$)
D_L	Length fractal dimension
D_σ	Variance fractal dimension
D_{PM_1}	Displacement of bidirectional pump–motor 1 (m^3/rev)
D_{PM_2}	Displacement of bidirectional pump–motor 2 (m^3/rev)
f_c	Cut-off frequency (Hz)
f_h	Highest frequency (Hz)
F_L	Static load force (N)
$G_p(s)$	Transfer function of the plant
g	Gravitational acceleration (m/s^2)
H_r	Hurst exponent
J_e	Total moment of inertia ($kg\,m^2$)
K_p	Proportional gain of the PID controller (V/m)
K_i	Integral gain of the PID controller (V/(m s))
K_d	Derivative gain of the PID controller (V s/m)
k_m	Servomotor gain (rev/(s V))
l_1	Upper arm length (m)

l_2	Lower arm length (m)
$L_{N-1}(\cdot)$	Legendre–Gauss polynomial
m_L	Load mass (kg)
m_{rod}	Piston/rod mass (kg)
M_p	Maximum overshoot
N_F	Size of stationary frame
n_k	Scale number
P_L	Load pressure (Pa)
P_{or}	Cracking pressure (Pa)
P_A	Pressure in chamber A (Pa)
P_B	Pressure in chamber B (Pa)
Q_{ac}	Compensation flow from the charge pump to the cap side (m^3/s)
Q_{bc}	Compensation flow from the charge pump to the rod side (m^3/s)
Q_{cp}	Additional flow supplied by the charge pump (m^3/s)
q_k	Quadrature weights
$\mathbb{N}$	Set of natural numbers
$\mathbb{R}$	Set of real numbers
$s = \sigma + j\omega$	Complex frequency variable
$s_\triangle$	Slope of the line in the log–log plot
sg	Specific gravity of the hydraulic fluid
t_m	Servo time constant (s)
T_f	Time interval (s)
t_p	Peak time (s)
t_r	Rise time (s)
t_s	Settling time (s)
T_e	Motor torque (N m)
T_L	Load torque (N m)

T_{PM_2}	Torque of hydraulic pump–motor 2 (N m)
v_p	Rod velocity (m/s)
x_p	Rod displacement (m)
Δ	Change or difference
$\beta_0, \beta_1, \beta_2$	Regression coefficients
β_e	Nominal effective bulk modulus (Pa)
$\hat{\beta}_e$	Perturbed effective bulk modulus used for online evaluation (Pa)
ψ_k	Shifted Legendre–Gauss nodes
∇	Gradient operator
ϱ	Spectral exponent
σ	Standard deviation
ϕ	Phase angle (rad)
ϕ_m	Phase margin (deg)
ω_d	Damped natural frequency (rad/s)
ω_g	Gain crossover frequency (rad/s)
ω_n	Undamped natural frequency (rad/s)
ω_m	Servomotor angular speed (rad/s)
ω_p	Angular velocity during the storage phase (rad/s)
ω_r	Angular velocity during the reutilization phase (rad/s)
ξ	Piston damping coefficient (N s/m)
ζ	Damping ratio

Chapter 1

Introduction

1.1 Problem Statement

Hydraulic actuation systems offer several key advantages over other types of actuators. The low compressibility of hydraulic fluid enables efficient pressure transmission, allowing hydraulic actuators to achieve fast, smooth, and responsive motion [1]. In addition, hydraulic actuators provide a high power-to-weight ratio, allowing large force generation with relatively low inertia compared to electric motors. Among hydraulic actuation technologies, the electro-hydrostatic actuator (EHA) has gained significant attention in advanced engineering applications, including aircraft systems and robotic platforms. This is due to its ability to combine the benefits of hydraulic power with improved efficiency.

These features make EHA systems particularly attractive for applications requiring high force density, fast dynamic response, and accurate motion control. However, their performance is influenced by several characteristics and challenges:

1. **Uncertainties in system dynamics**
   EHAs exhibit complex dynamics influenced by varying loads, friction, and fluid properties. These uncertainties can lead to deviations in system behavior, making accurate modelling, control, and condition monitoring more challenging.

2. **Noise in sensor measurements**
   Measurements acquired from sensors in EHAs, such as pressure sensors and position encoders, are often contaminated by noise. This can reduce control accuracy and affect system performance. Robust signal processing and control strategies are therefore required to ensure reliable operation.

3. **Nonlinearities and time delays**

EHAs exhibit nonlinear behavior due to valve dynamics, pressure-flow relationships, and friction. Time delays also arise from system dynamics and interfacing components.

4. **Fault detection**
   EHA systems are subject to various faults, including internal and external leakage. Failure to detect these faults can significantly degrade system performance and reliability. Effective fault detection techniques are essential for the timely identification and mitigation of faults in hydraulic systems.

5. **Control strategy design**
   Robust control strategies are required to handle uncertainties, noise, and nonlinearities for stable and accurate performance.

6. **Energy optimization**
   EHA systems are required to operate efficiently under different operating conditions while maintaining high tracking accuracy. However, energy efficiency often involves tradeoffs with other performance requirements, such as controllability and stability. For example, increasing damping can reduce oscillations and improve control performance, but it also increases energy dissipation. Consequently, energy optimization is necessary to minimize energy consumption while maintaining satisfactory tracking performance.

These challenges are further intensified in practical scenarios where data availability is limited, system dynamics change over time, and energy efficiency becomes a critical requirement. Traditional model-based approaches for condition monitoring, controller design, and energy optimization rely on accurate system models and sufficient data, which are often difficult to obtain in real-world hydraulic systems. Although recent artificial intelligence (AI) methods, such as deep neural networks, have demonstrated promising performance, they often suffer from a lack of interpretability, limiting their transparency and trustworthiness in engineering applications. This thesis aims to develop interpretable methods to address these challenges.

1.2 Research Objectives

The main objective of this thesis is to investigate the application and develop new interpretable approaches based on Bayesian principles, machine learning, and optimization-based methods for condition monitoring, adaptive control, and energy optimization in EHA systems. The specific objectives are as follows:

1. To develop fault detection approaches for EHA systems, including an approach based on polyscale complexity measures and Bayesian classification, and a data-driven modelling approach under limited data conditions.

2. To first investigate a model-based control approach in the frequency domain using an optimization method that enables the incorporation of both transient performance and robustness requirements, and next, a model-free adaptive control strategy that enables online tuning of controller parameters in the presence of changing system dynamics.

3. To formulate a multi-objective optimization problem that balances tracking performance and energy consumption in EHA systems.

1.3 Proposed Methodologies

AI-based methods have recently demonstrated strong performance in science and engineering. However, many of these methods are often considered as black-box models that limit their explainability. This lack of explainability can reduce trust and restrict their use in safety-critical applications. As a result, increasing attention has been given to the development of AI methods that improve transparency and interpretability. This thesis proposes interpretable methodologies based on mathematical approaches in machine learning, Bayesian inference, and optimization. These methods can be applied not only to hydraulic systems but also to other engineering problems.

For condition monitoring of the EHA, a fault detection method based on fractal complexity measures and a Bayesian classification framework is first developed to distinguish between healthy and faulty conditions through posterior analysis. To address the challenge of access to low-data, a sparse identification of nonlinear dynamics (SINDy) method is employed to extract governing equations of the system directly from measurement data. Bayesian optimization is used to tune the parameters of the SINDy model. Using the resulting models, a data-driven approach is developed for reliable fault detection under limited data conditions.

For control design, both model-based and model-free control strategies are studied. For model-free controller tuning, a reinforcement learning is used to explore the controller parameter space. The collected data are then used to train a Gaussian process (GP) model, which serves as a surrogate model of system performance. The GP model guides the online selection of controller gains, enabling fast adaptation to changes in system dynamics while limiting unnecessary exploration.

For energy optimization, a multi-objective optimization framework is formulated to minimize tracking error and energy consumption simultaneously. Instead of the random initialization, a GP-based surrogate modelling method is integrated into the initialization stage of the multi-objective

algorithm. All the developments are exemplified with the newly developed EHA system equipped with the ESR unit.

1.4 Organization of the Thesis

This thesis is organized as follows:

- **Chapter 2** presents the description of the EHA system, including its key components, as well as the energy storage and reutilization (ESR) unit integrated within the system.
- **Chapter 3** provides the background required for this work. It includes an overview of Bayesian inference, its connections to GPs and Bayesian optimization, as well as fractals, and a review on optimization based methods for controller tuning.
- **Chapter 4** focuses on condition monitoring of the EHA system under investigation with application to internal leakage fault detection. First, the fault detection using Bayesian classification is presented. Next, the SINDy algorithm is used to model system dynamics under low-data conditions.
- **Chapter 5** presents both model-based and model-free approaches for controller tuning. The model-based method is formulated in the frequency domain, while the model-free approach describes the integration of reinforcement learning and GP models for efficient controller gain tuning under changing system dynamics.
- **Chapter 6** investigates energy optimization of the EHA system equipped with an ESR unit. A multi-objective optimization framework is formulated in which both the controller parameters and the ESR operating thresholds are treated as decision variables. The objective is to minimize tracking error and energy consumption while determining when energy should be stored or reutilized.
- **Chapter 7** summarizes contributions made in this thesis and future research directions.
- **Appendix** provides a review of the MATLAB/Simulink models used in this thesis.

Chapter 2

Description of the Hydraulic System Under Investigation

Hydraulic systems are widely used in robotic manipulators, aircraft actuation systems, mining equipment, and construction machinery [2, 3]. These systems utilize pressurized fluid to transmit power. The fundamental principle of hydraulic systems is to convert mechanical energy into hydraulic energy, and vice versa. The basic components enabling this transformation include pumps, valves, cylinders, and motors. Through the interaction of these elements, pressurized fluid flows within the system, allowing precise control of motion and force.

This chapter introduces the hydraulic systems considered in this thesis. These systems [4, 5] were newly designed and are used as testbeds to investigate fault detection methodologies, control strategies, and energy optimization in the next chapters. First, the electro-hydrostatic actuator (EHA) system is described. Next, the concept of an energy storage and reutilization (ESR) unit integrated with the EHA system is presented. The ESR system aims to capture excess energy, otherwise being wasted, and store it in a hydraulic accumulator. This stored energy can later be released back into the EHA system to assist motion, thereby reducing the overall power demand. By enabling energy recovery and reuse, the ESR system enhances the overall energy efficiency of the EHA system.

The following sections describe the structure of the EHA system and the integration of the ESR system in detail. The mathematical models describing these systems are also presented to aid in understanding their dynamic behavior.

2.1 Electro-Hydrostatic Actuator System

An electro-hydrostatic actuator (EHA) is an emerging actuation technology that replaces centralized hydraulic systems with a self-contained, localized, direct-drive hydraulic actuator. EHAs are

commonly used in aerospace applications and are increasingly being adopted in heavy machinery, including excavators and backhoe loaders [6–8]. In this thesis, the EHA system is investigated as part of the actuator system controlling a backhoe arm. Figure 2.1 illustrates a typical backhoe loader.



Figure 2.1: Typical backhoe loader (adapted from [9]).

An arm of a hydraulic backhoe loader is retrofitted as a research testbed to study fault detection, control, and energy optimization of the EHA system. The setup of the system is illustrated in Fig. 2.2.

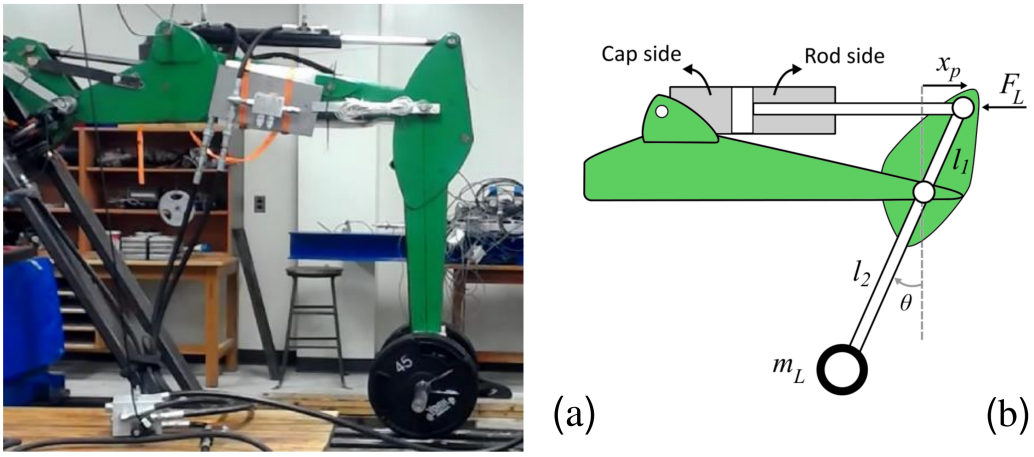


Figure 2.2: Backhoe arm setup [4].

The rig and the mathematical models are based on the hydraulic circuit and modelling framework developed by Costa and Sepehri [4, 5], where a bidirectional pump regulates the motion of an asymmetric cylinder mounted on a backhoe link. A schematic of the electro-hydrostatic circuit is shown in Fig. 2.3.

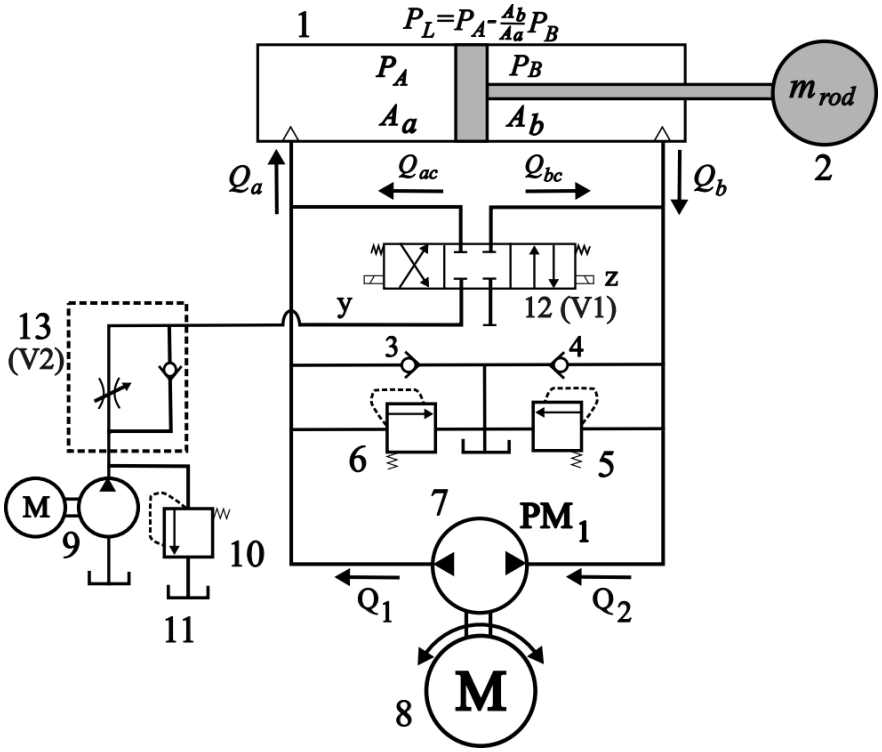


Figure 2.3: Schematic of the EHA circuit used in the experimental setup: (1) actuator; (2) mass of piston and rod; (3, 4) check valves; (5, 6) relief valves; (7) bidirectional pump PM_1; (8) servomotor; (9) auxiliary pump-motor; (10) relief valve; (11) tank; (12) directional valve (V1); (13) flow control valve (V2).

The input to the system is a voltage command signal applied to the servo drive, which controls the servomotor driving the bidirectional pump. This command voltage is proportional to the target motor torque and is distinct from the high-power electrical input supplied to the motor. The servomotor used in the setup is an AC brushless servomotor [10]. In this system, the electromagnetic torque is directly proportional to the motor current, and the servo drive regulates the current to achieve the desired torque. The servomotor speed, ω_m, for normal operation can be calculated using Newton's second law for rotation. This is expressed using the relationship between net torque and angular acceleration as

$$T_e - T_L = J_e \frac{d\omega_m}{dt} + B_e \omega_m \tag{2.1}$$

where T_e is the motor torque, T_L is the load torque, J_e is the total moment of inertia of both the servomotor and the hydraulic pump-motor (PM_1), and B_e is the motor viscous damping coefficient.

The load torque, T_L, generated by pump-motor (PM_1) can be calculated as follows

$$T_L = D_{PM_1}(P_A - P_B) \tag{2.2}$$

where D_{PM1} is the pump displacement, and P_A and P_B are the cap-side and rod-side chamber pressures, respectively.

The operation of the EHA can be classified into four quadrants based on the signs of the piston velocity $\dot{x}_p$ and the load pressure P_L, as illustrated in Fig. 2.4. The load pressure is defined as

$$P_L = P_A - \frac{A_b}{A_a} P_B \tag{2.3}$$

where A_b and A_a denote the rod-side and piston-side areas, respectively. These operating quadrants determine whether the system functions in motoring mode (power returned from the load) or pumping mode (power supplied to the load).

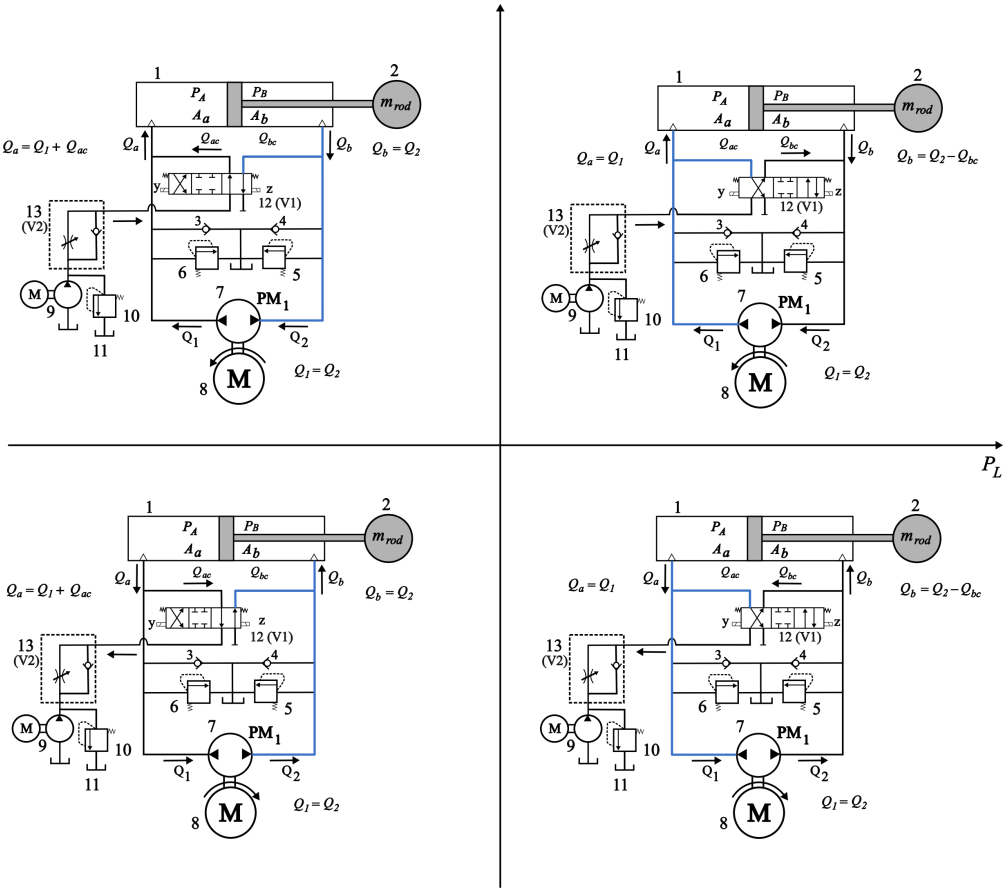


Figure 2.4: Four-quadrant operation of the EHA circuit.

The single-rod hydraulic cylinder is asymmetric. During extension, additional flow is required from a charge pump, whereas during retraction, the excess flow is returned to the tank. Flow compensation is achieved using a three-position four-way directional valve (V1) and a one-directional flow control valve (V2). Valve V2 establishes the corresponding flow path within the circuit, ensuring proper fluid distribution under each operating condition. Check valves and relief valves are employed to regulate the system pressures and ensure safe operation. The directional valve (V1) is actuated based on the load pressure, P_L. Specifically, the switching logic is defined as follows:

(i) Solenoid y is activated when $P_L > 0$,

(ii) Solenoid z is activated when $P_L < 0$.

Depending on the quadrant, the bidirectional pump either supplies additional flow to the actuator chambers (Quadrants I and II) or absorbs excess flow from the circuit (Quadrants III and IV). In the latter case, the surplus hydraulic flow is redirected to the tank.

When $P_L > 0$, and the system operates in Quadrant I, the charge pump supplies additional flow Q_{cp} to the actuator circuit. In this case, $Q_{ac} = 0$, and the ideal flow through valve $V2$ on the rod side is given by

$$Q_{bc} = Q_{cp} \tag{2.4}$$

Conversely, when $P_L < 0$, and the system operates in Quadrant II, the charge pump again supplies flow to the actuator. In this quadrant, $Q_{bc} = 0$, and the ideal flow through valve $V2$ on the cap side becomes

$$Q_{ac} = Q_{cp} \tag{2.5}$$

In contrast, Quadrants III and IV correspond to pumping modes, where excess hydraulic energy is dissipated by routing the flow back to the tank. When $P_L < 0$ and the system operates in Quadrant III, the excess fluid from the cap-side chamber is discharged to the tank through valve $V2$. The ideal flow is expressed as

$$Q_{ac} = -C_{vi}\sqrt{\frac{P_A - P_{or}}{sg}} \tag{2.6}$$

Similarly, when $P_L > 0$ and the system operates in Quadrant IV, the excess fluid from the rod-side chamber is routed to the tank via valve $V2$. The corresponding ideal flow is

$$Q_{bc} = -C_{vi}\sqrt{\frac{P_B - P_{or}}{sg}} \tag{2.7}$$

In these expressions, C_{vi} denotes the valve flow coefficient, sg represents the specific gravity of the hydraulic fluid, and P_{or} is the cracking pressure of the relief valve.

The pump delivers hydraulic flow to the actuator chambers, resulting in the pressures P_A and P_B, whose dynamics are governed by fluid compressibility. For the cap-side chamber, the pressure dynamics are

$$\dot{P}_A = \frac{\beta_e}{V_{0a} + A_a x_p} \left(\omega_m D_{\mathrm{PM_1}} + Q_{ac} - A_a \dot{x}_p \right) \tag{2.8}$$

where β_e is the effective bulk modulus, V_{0a} the cap-side chamber volume at the cylinder mid-stroke, x_p the piston displacement, and Q_{ac} denotes the charge pump flow. A similar expression holds for the rod-side chamber

$$\dot{P}_B = \frac{\beta_e}{V_{0b} - A_b x_p} \left(Q_{bc} - \omega_m D_{\mathrm{PM_1}} + A_b \dot{x}_p \right) \tag{2.9}$$

where V_{0b} is the rod-side chamber volume at the cylinder mid-stroke. The resulting pressure difference across the actuator chambers produces a hydraulic force that drives the piston and causes rod motion within the cylinder. The backhoe arm supports variable payloads, and by attaching disks to the arm, the load mass m_L can be adjusted. The disks provide adjustable inertial loads for controlled testing. However, they do not reproduce soil resistance or impact loads during digging. The static load force acting on the actuator is

$$F_L = \frac{m_L \, g \, l_2 \sin\theta}{l_1 \cos\theta} \tag{2.10}$$

where θ is the arm angle, and l_1 and l_2 are the upper and lower arm lengths, respectively. Rod displacement is measured using an incremental encoder. Experimental data are acquired using a Raspberry Pi Model 4 interfaced with a Texas Instruments ADS1015 analog-to-digital converter (ADC), which provides multiple input channels with 14-bit resolution and a sampling rate ranging from 0.125 to 3300 samples per second.

The piston velocity follows from the force balance

$$\dot{v}_p = \frac{1}{m_{\mathrm{rod}}} \left(A_a P_L - \xi v_p - F_L \right) \tag{2.11}$$

The piston displacement satisfies $\dot{x}_p = v_p$ and ξ is the piston viscous damping coefficient. The system parameters are listed in Table 2.1.

Table 2.1: Parameters of electro-hydrostatic actuator (EHA) system.

Parameter	Symbol	Value	Units
Pump displacement	D_{PM_1}	8×10^{-6}	$\mathrm{m}^3/\mathrm{rev}$
Piston area	A_a	3167×10^{-6}	m^2
Cap-side volume	V_{0a}	2.3×10^{-3}	m^3
Rod-side volume	V_{0b}	2.1×10^{-3}	m^3
Rod-side area	A_b	2380×10^{-6}	m^2
Bulk modulus	β_e	689×10^6	Pa
Piston/rod mass	m_{rod}	10	kg
Load mass	m_L	0–367	kg
Specific gravity	sg	0.86	–
Piston viscous damping	ξ	900	Ns/m
Backhoe effort arm length	l_1	0.3	m
Backhoe load arm length	l_2	1.2	m
Valve coefficient	C_{vi}	1.6×10^{-7}	$\mathrm{m}^3/(\sqrt{\mathrm{Pa}}\,\mathrm{s})$
Cracking pressure	P_{or}	5.5×10^5	Pa
Servo time constant	t_m	3	1/s
Servomotor gain	k_m	5.8	rev/(sV)
Time constant of directional valve, V1	τ_d	9.5×10^{-3}	s

2.2 Energy Storage and Reutilization Unit

Energy storage and reutilization represent key mechanisms for enhancing the efficiency and sustainability of modern hydraulic systems. Energy storage refers to the process of capturing and retaining energy that would otherwise be dissipated or lost, while energy reutilization involves reintroducing the stored energy back into the system when it is beneficial. When these two processes are integrated within a unified framework, the resulting configuration is referred to as an ESR system. Conventional energy storage devices include supercapacitors, flywheels, hydraulic accumulators, and batteries. These components are increasingly employed in industrial machinery to improve energy efficiency, reduce operational costs, and enhance system sustainability.

This section describes the previously developed ESR system integrated with the EHA system [10, 11]. Therefore, only a brief overview is provided, and the reader is referred to [10] for detailed design and implementation. Figure 2.5 shows the experimental setup and final assembly of the ESR system.

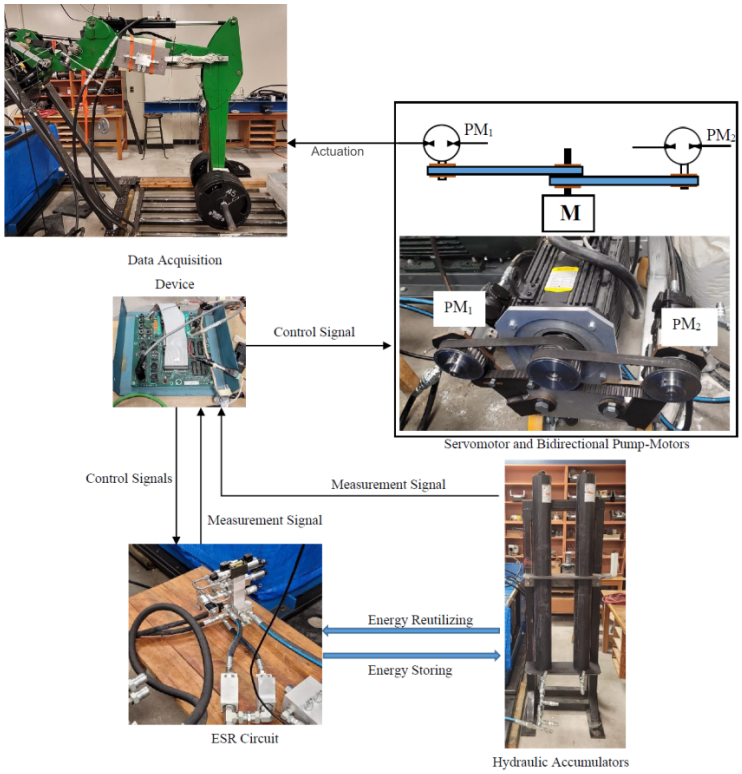


Figure 2.5: Experimental setup of the energy storage and reutilization (ESR) system applied to the EHA (copied from [10]).

Figure 2.6 illustrates the configuration of the ESR system added into the EHA system.

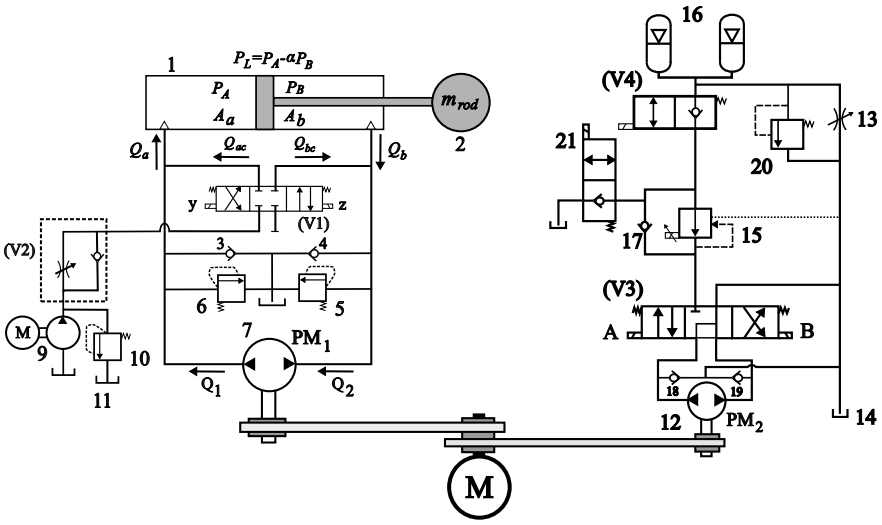


Figure 2.6: ESR system applied to the EHA system: (12) pump motor 2; (13) flow discharge valve; (14) tank; (15) electro-proportional pressure reducing valve (EPPRV); (16) hydraulic accumulator; (17), (18), (19) check valves; (20) relief valve; (21) bleed-off valve; (V3) float-centre type three-position four-way valve; (V4) reutilization valve.

To achieve energy storage and reutilization during different operating conditions of the actuator, a control logic based on the EHA operating quadrants and the corresponding pumping or motoring operation is used [10]. During motoring operation, excess energy can be stored in the accumulator when the specified conditions are satisfied, whereas during pumping operation, the stored energy can be reutilized to assist the system operation. The flowchart 2.7 summarizes the energy storage and reutilization strategy proposed in [10].

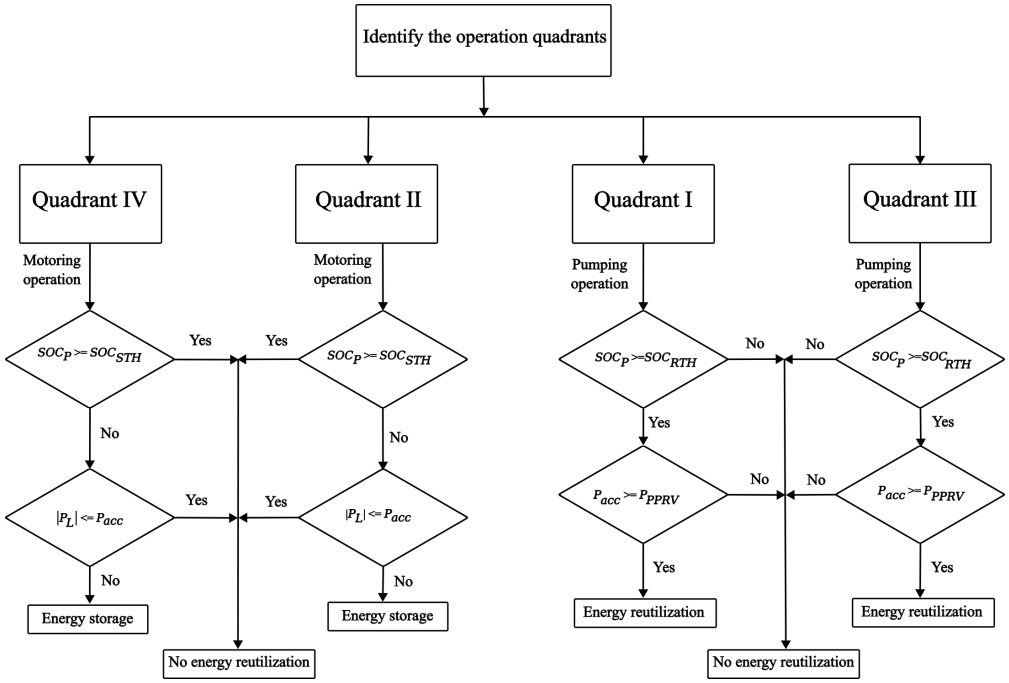


Figure 2.7: Energy storage and reutilization flowchart [10].

The mathematical modelling of the ESR system begins with the modelling of the electro-proportional pressure reducing valve (EPPRV), the hydraulic accumulator, the secondary hydraulic pump-motor (PM_2), and the main control valves. The governing equations of these components are combined to establish the complete dynamic model of the ESR system. As shown in Fig. 2.6, the EPPRV, labeled as component 15, plays a critical role in the energy reutilization process. Its primary function is to regulate the pressure supplied from the hydraulic accumulator, ensuring that the delivered pressure remains at a controlled and reduced level suitable for efficient energy reuse. The regulated hydraulic flow is directed to the secondary pump-motor PM_2 through valve V3, where the hydraulic energy is converted into mechanical rotational energy. In this configuration, PM_2 operates as a hydraulic motor and assists the main servomotor (PM_1) in driving the load, thereby reutilizing the previously stored energy within the system.

The hydraulic accumulator, labeled as component 16 in Fig. 2.6, functions as the primary energy storage element of the ESR system. It stores hydraulic energy by compressing the gas chamber

during regenerative or low-demand operating phases and releases the stored energy when energy reutilization is enabled. Accurate modelling of the accumulator is essential for predicting the available stored energy and its influence on system efficiency. The thermodynamic behavior of the gas within the hydraulic accumulator of the ESR system is modeled using the Benedict–Webb–Rubin (BWR) equation of state. The BWR model provides a more accurate representation of real-gas behavior compared to ideal gas assumptions, particularly under high-pressure conditions. The BWR equation is expressed as

$$P_{acc} = \frac{R_u T_g}{\bar{v}} + \frac{B_0 R_u T_g}{\bar{v}^2} - A_0 - \frac{C_0}{T_g^2} + \frac{b R_u T_g - a}{\bar{v}^3} + \frac{a\alpha}{\bar{v}^6} + \frac{c\left(1 + \frac{\gamma}{\bar{v}^2}\right)}{\bar{v}^3 T_g^2} \exp\left(-\frac{\gamma}{\bar{v}^2}\right) \tag{2.12}$$

where P_{acc} denotes the accumulator gas pressure, $\bar{v}$ represents the specific molar volume (m^3/kmol), T_g is the absolute temperature (K), and R_u is the universal gas constant. The recommended validity range for the BWR equation is $T_g > 250$ K and $P_{acc} < 20{,}000$ kPa. The numerical values of the parameters appearing in (2.12) are listed in Table 2.2. A detailed mathematical formulation of the accumulator temperature evolution and coefficient derivation can be found in [10].

Table 2.2: Benedict–Webb–Rubin (BWR) coefficients

Parameter	Value	Unit
a	2.54	kPa m^9/$kmol^3$
b	0.002328	m^6/$kmol^2$
c	7.379×10^4	kPa m^9 K^2/$kmol^3$
A_0	106.73	kPa m^6/$kmol^2$
B_0	0.04074	m^3/kmol
C_0	8.164×10^5	kPa m^6 K^2/$kmol^2$
α	1.272×10^{-4}	m^9/$kmol^3$
γ	0.0053	m^6/$kmol^2$

To appropriately size the accumulator for the ESR system, a systematic procedure is adopted, consisting of the following steps:

1. Determination of the torque required by the ESR circuit to effectively assist the primary pump-motor unit (PM_1) via the servomotor.
2. Calculation of the accumulator pre-charge gas pressure based on the established relationship between torque and pressure.
3. Estimation of the required oil volume to enable effective energy reutilization.
4. Determination of the accumulator oil capacity based on the above calculations.

Based on this procedure, 9.64 L gas piston accumulators are selected for the experimental setup. A commonly used indicator for accumulator energy level is the state of charge based on volume (SOC_v), defined as the ratio of the instantaneous oil volume stored in the accumulator, V_{oil}, to the maximum oil storage capacity, $V_{\text{total oil stored}}$ [10], given by

$$SOC_v = \frac{V_{oil}}{V_{\text{total oil stored}}} \tag{2.13}$$

Since direct measurement of SOC_v is impractical in most hydraulic systems, the state of charge based on pressure (SOC_p) is introduced. SOC_p represents the ratio of the current energy level to the maximum storable energy in the accumulator and is defined as

$$SOC_p = \frac{P_{acc} - P_{min}}{P_{max} - P_{min}} \tag{2.14}$$

where P_{acc} is the instantaneous accumulator pressure, P_{min} is the pre-charge gas pressure, and P_{max} is the maximum allowable accumulator pressure, which corresponds to the relief valve cracking pressure.

The control of the ESR process can be implemented using either SOC_v or SOC_p, depending on the control strategy. SOC_v characterizes the stored oil volume, whereas SOC_p characterizes the accumulator pressure. In this work, SOC_p is employed to govern the ESR process.

Two threshold values are defined at the beginning of the experiment (see Fig. 2.7): the storage threshold (SOC_{STH}) and the reutilization threshold (SOC_{RTH}). SOC_{STH} limits the maximum energy stored in the accumulator, whereas SOC_{RTH} initiates energy reutilization by discharging the accumulator when SOC_p reaches or exceeds SOC_{RTH}, particularly during pumping phases.

In the experimental setup, the secondary hydraulic pump-motor unit PM_2 is selected to be identical to the primary unit PM_1, ensuring a constant transmission ratio. Both pump-motors are mechanically connected to the servomotor using identical pulleys and belts. Frictional and sliding losses in the pulley-belt system are neglected, and therefore, torque losses are assumed to be negligible. The flow direction of PM_2 is controlled by the float-centre directional valve V3 (refer to Fig. 2.6). Depending on the position of V3, one port of PM_2 is connected either to the storage check valve and EPPRV or to the tank. When V3 is in the centre position, both ports of PM_2 are connected to the tank (refer to Fig. 2.6). The ESR system incorporates valves to facilitate energy storage and reutilization. The reutilization valve, V4 in Fig. 2.6, is a two-position valve exclusively responsible for activating the energy reutilization process. In its right position, the valve behaves as a check valve, permitting unidirectional flow from the pump-motor unit toward the accumulator. In its left position, the valve allows bidirectional flow, enabling downstream discharge from the accumulator during energy reutilization.

A first-order transfer function can be employed to describe the solenoid actuator of valve V4. However, since the mechanical time constant of the solenoid is negligibly small, the transfer function is approximated as unity. The reutilization valve V4 is activated simultaneously with the directional valve V3 during the energy reutilization phase. The activation conditions of the directional valve V3 are described in detail in Chapter 6, for energy optimization.

The float-centre directional valve V3 is actuated by two solenoids, labeled A and B. The activation of these solenoids depends on the load pressure P_L, accumulator pressure P_{acc}, EPPRV outlet pressure P_{pprv}, and the state of charge SOC_p.

Assuming ideal conditions with no volumetric or mechanical losses, the flow rate through PM_2 during the energy storage process is given by

$$Q_{\text{PM}_2} = D_{\text{PM}_2}\,\omega_p \tag{2.15}$$

where D_{PM2} is the displacement of PM_2, identical to that of PM_1, and ω_p is the equivalent angular velocity during the pumping operation (quadrants II and IV).

The equivalent angular velocity ω_p is obtained from the net torque balance

$$T_e - T_L - T_{\text{PM}_2} = J_e\frac{d\omega_p}{dt} + B_e\omega_p \tag{2.16}$$

where T_e is the motor torque, T_L is the load torque, and J_e and B_e denote the equivalent inertia and motor viscous damping of the servomotor, PM_1, and PM_2. The load torque, T_L, generated by the pump-motor (PM_1), can be calculated as follows

$$T_L = D_{\text{PM}_2}(P_A - P_B) \tag{2.17}$$

The torque produced by PM_2 during the pumping stage is expressed as

$$T_{\text{PM}_2} = \begin{cases} D_{\text{PM}_2}P_{\text{acc}}, & \text{Quadrant II} \\ -D_{\text{PM}_2}P_{\text{acc}}, & \text{Quadrant IV} \end{cases} \tag{2.18}$$

During the energy reutilization process, PM_2 operates as a hydraulic motor, and its flow rate is given by

$$Q_{\text{PM}_2} = D_{\text{PM}_2}\,\omega_r \tag{2.19}$$

where ω_r is the equivalent angular velocity during energy reutilization.

The angular velocity ω_r is obtained from

$$T_e - T_L + T_{\text{PM}_2} = J_e\frac{d\omega_r}{dt} + B_e\omega_r \tag{2.20}$$

The torque contribution of PM_2 during the motoring operation (quadrants I and III) is expressed as

$$T_{PM_2} = \begin{cases} D_{PM_2} P_{pprv}, & \text{Quadrant I} \\ -D_{PM_2} P_{pprv}, & \text{Quadrant III} \end{cases} \tag{2.21}$$

where P_{pprv} denotes the outlet pressure of the EPPRV.

To prevent excessive accumulator discharge, the outlet power command is limited to a fixed fraction of the available accumulator power. Specifically, the outlet power is constrained to remain below 95% of the accumulator power.

It should be noted that the governing equations for the pressure dynamics of the cap and rod side chambers, actuator velocity, and piston displacement remain unchanged with the integration of the ESR system. Therefore, Eqs. (2.8)-(2.7) are also valid for the EHA system with the ESR unit.

2.3 Summary

This chapter presented the hydraulic systems considered in this thesis. First, the electro-hydrostatic actuator (EHA) system was introduced, including its structure and governing dynamic equations. The EHA system is considered the basis for the development of condition monitoring and control strategy in Chapters 4 and 5.

Next, the integration of the ESR unit with the EHA system was discussed. The main components of the ESR unit, including the hydraulic accumulator, EPPRV, and secondary pump-motor, were described, and their mathematical models were presented. The integration of the ESR unit with the EHA system was discussed, highlighting its role in improving energy efficiency. The ESR system is further utilized in Chapter 6 for the development of the energy optimization strategy.

Chapter 3

Preliminaries

This chapter provides the fundamental concepts and theoretical background required for the development of the proposed interpretable methodologies in this thesis. The aim is to present key tools and frameworks that are employed throughout the subsequent chapters. First, the principles of Bayesian inference as the basis for probabilistic modelling and representation of uncertainty is described. This perspective is used in multiple parts of the thesis, including condition monitoring, control, and energy optimization. Additionally, fractal theory and fractal complexity measures are presented as powerful feature extraction tools for capturing complex patterns and variability in measurement signals. Furthermore, a review of optimization-based perspectives on PID controller tuning for hydraulic systems is presented to provide context for the strategies introduced in the following chapters. The concepts presented in this chapter are connected to the author's previous publications[1,2,3].

3.1 Bayesian Framework

The Bayesian viewpoint has found many applications across various scientific and engineering disciplines. In machine learning, Bayesian methods are widely used for regression, classification, and modelling [12]. In signal processing and system identification, they enable robust estimation in the presence of uncertainty and measurement noise [13]. In robotics and control, Bayesian filtering, learning of system dynamics, and probabilistic trajectory modelling are employed to enhance

[1]S. Hosseinpour, W. Kinsner, and N. Sepehri, "Precision and Accuracy of Length and Variance Fractal Dimensions Computed from Fractional Self-Affine Signals," *Proceedings of the 8th World Congress on EECSS'22*, 2022.

[2]S. Hosseinpour, W. Kinsner, and N. Sepehri, "Monoscale and Polyscale Analyses of Physical Signals for Compressive Detection of Relevant Complexity Features," *IEEE 21st International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, 2022.

[3]S. Hosseinpour, W. Kinsner, and N. Sepehri, "A Bayesian optimized neural network for fault detection in electro-hydrostatic actuators," *Journal of Machine Intelligence and Data Science (JMIDS)*, vol. 5, no. 1, pp. 144-151, 2024.

system performance under uncertainty [14]. In medical diagnosis and reliability analysis, Bayesian approaches provide a natural mechanism for combining prior expert knowledge with observed data [15]. In optimization and engineering, Bayesian optimization has emerged as a powerful tool for hyperparameter tuning [16, 18].

In this thesis, Bayesian ideas are used in several aspects. First, Bayesian inference provides a framework for classification. Second, Bayesian optimization is employed to tune hyperparameters in a fault detection algorithm. Third, Bayesian inference is utilized to construct Gaussian process surrogate models for online controller tuning. Finally, in the context of multi-objective optimization, these surrogate models are used to generate initial solutions for the optimization process.

3.1.1 Bayesian Inference

There are two general approaches for learning from data: deterministic and probabilistic methods. Deterministic estimation methods typically provide a single best estimate of the unknown quantities. In these estimation, parameters are usually fixed, but unknown, and uncertainty is associated with repeated sampling. In contrast, Bayesian inference assigns a probability distribution directly to the unknown parameters or functions, thereby allowing uncertainty to be propagated naturally through the model. This makes Bayesian methods especially attractive in applications where data are scarce, experiments are costly, or prior engineering knowledge is available and should be incorporated into the learning process [19]. Such situations are common in control, fault detection, and optimization of physical systems, where each experiment may require substantial time or computational cost.

Bayesian inference is based on Bayes' theorem. For two events A and B, Bayes' theorem is given by

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{3.1}$$

In Bayesian inference, the goal is to determine the distribution of an unknown parameter vector θ given the observed data $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$. Using Bayes' theorem, the posterior distribution is given by

$$P(\theta \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \theta)\, P(\theta)}{P(\mathcal{D})} \tag{3.2}$$

where $P(\theta \mid \mathcal{D})$ is the posterior distribution, $P(\theta)$ is the prior distribution, and $P(\mathcal{D} \mid \theta)$ is the likelihood. Assuming independent observations, the likelihood function can be expressed as

$$P(\mathcal{D} \mid \theta) = \prod_{i=1}^{n} f(x_i \mid \theta) \tag{3.3}$$

where $f(x_i \mid \theta)$ is the probability density function.

The denominator, known as the marginal likelihood or evidence, is given by

$$P(\mathcal{D}) = \begin{cases} \sum_\theta P(\mathcal{D} \mid \theta) P(\theta), & \text{for discrete } \theta \\ \int P(\mathcal{D} \mid \theta) P(\theta)\, d\theta, & \text{for continuous } \theta \end{cases} \tag{3.4}$$

and serves as a normalization constant ensuring that the posterior distribution is valid. Therefore, the posterior distribution can be simply expressed in proportional form as

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior} \tag{3.5}$$

The posterior distribution summarizes our belief about the parameter after seeing the data. It takes into account our prior belief (the prior distribution) and the data (likelihood). Any subsequent Bayesian inference, such as a Bayesian estimate or a Bayesian credible interval, is calculated from the posterior distribution.

The prior distribution plays a key role in Bayesian analysis. It may represent physical intuition, historical observations, design constraints, or expert knowledge. A weakly informative prior can be used when little is known in advance, while an informative prior can inject meaningful domain knowledge into the model. As more data are observed, the posterior tends to be increasingly influenced by the likelihood, although the prior remains important when the dataset is small or noisy.

The likelihood function captures the probabilistic relationship between unknown parameters and observations. For instance, if observations are assumed to be corrupted by Gaussian noise, then the likelihood often takes a Gaussian form. When the prior is chosen to be conjugate to the likelihood, the posterior may be obtained in closed form. However, in many practical problems, the posterior is analytically intractable, especially when the model is nonlinear, non-Gaussian, or high-dimensional. In such cases, approximate Bayesian inference techniques are required. One of the widely used methods is Markov chain Monte Carlo (MCMC), which generates samples from the posterior distribution to enable statistical inference [24]. These techniques approximate the posterior when exact evaluation is not feasible. This can extend Bayesian analysis to a broad class of practical problems in science and engineering.

3.1.2 Gaussian Processes

In classical Bayesian inference, models are typically parametric, meaning that a fixed functional form is assumed and the goal is to estimate a finite number of unknown parameters. For example, in a linear model $f(x) = ax + b$, Bayesian inference is used to estimate the parameters a and b, resulting in a finite-dimensional problem.

In contrast, Gaussian processes (GPs) adopt a nonparametric perspective, where no explicit

functional form is assumed in advance [20]. Instead of estimating a limited set of parameters, the objective is to infer the entire function $f(x)$. Since a function has a value at every point in its domain, this corresponds to an infinite-dimensional problem. For this reason, GPs are referred to as nonparametric Bayesian models.

From a Bayesian viewpoint, a GP defines a prior distribution over functions. A GP is fully specified by a mean function $m(x)$ and a covariance function (kernel) $k(x, x')$, written as

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{3.6}$$

The mean function $m(x)$ represents the expected value of the function at each point, while the kernel function $k(x, x')$ defines the correlation between function values at different points x and x'. In practice, the kernel plays a crucial role, as it represents prior assumptions about the function, such as similarity between nearby points.

Given observed data, Bayesian inference is used to update this prior into a posterior distribution over functions. This allows the model to adapt its shape based on the data, without requiring a predefined functional form. As a result, GPs provide both a predictive mean, which represents the estimated function, and a predictive variance, which quantifies the uncertainty of the prediction. This makes GP particularly suitable for modelling complex nonlinear systems and for applications where data are limited or expensive to obtain.

3.1.3 Bayesian Optimization

Bayesian optimization (BO) is a strategy for optimizing expensive black-box functions. It is widely used for hyperparameter tuning in machine learning models [16]. Unlike traditional optimization methods that require many evaluations, BO builds a probabilistic surrogate model of the objective function, often a GP, and uses it to guide the search for optimal parameters.

It then uses an acquisition function to decide where to sample next. The surrogate model is updated using Bayes' rule after each new observation, and the next evaluation point is selected to balance exploration and exploitation. This makes BO particularly effective when only a limited number of function evaluations can be afforded.

Common acquisition functions include probability of improvement, expected improvement (EI), and upper confidence bound (UCB). These functions use the predicted mean and uncertainty of the surrogate model to select the next point to evaluate. For example, EI favors points that are likely either to improve the current best solution or to reduce uncertainty in potentially high-performing regions. The key point is that BO relies fundamentally on Bayesian inference. Indeed, after each observation, the posterior over the objective function is updated, and this posterior guides the next decision. Therefore, BO may be understood as a sequential decision framework built on top of

Bayesian probabilistic modelling. The BO algorithm can be summarized as follows [17]:

1. **Input:** Initial dataset $(x_i, y_i = f(x_i))_{i=1:t}$, acquisition function $u(x)$, and GP model for $f(x)$.
2. **For** $t = 1 : N$ **do**
3. Update the GP model using all available data $(x_i, y_i = f(x_i))_{i=1:t}$.
4. Obtain the GP posterior mean $\mu_{\text{GP}}(x)$ and variance $\sigma_{\text{GP}}(x)^2$, representing the predicted function values and uncertainty, respectively.
5. Define the acquisition function $u(x)$ using the GP posterior.
6. Determine the next evaluation point by maximizing the acquisition function:

$$x_{t+1} = \arg\max_x u(x) \tag{3.7}$$

7. Evaluate the true objective function at x_{t+1}:

$$y_{t+1} = f(x_{t+1}) \tag{3.8}$$

8. Append the new observation (x_{t+1}, y_{t+1}) to the dataset.
9. **End For**
10. Return the optimal point:

$$x^* = \arg\max_{x_i} f(x_i) \tag{3.9}$$

11. **Output:** x^*.

In Chapter 4, Bayesian optimization is used to tune the hyperparameters of the sparse identification of nonlinear dynamics (SINDy) algorithm for fault detection under limited data conditions. The proposed framework combines the interpretability of SINDy models with the efficiency of Bayesian optimization. It enables accurate and reliable fault detection while reducing the need for manual parameter tuning.

3.2 Fractals

Fractals are objects that exhibit self-similarity at different scales, meaning that they display similar patterns or structures regardless of the level of magnification. Fractals have fractional dimensions, such as the famous Mandelbrot set, Triple Cantor set, and Koch curve [21]. Fractional calculus, on the other hand, deals with generalizing differentiation and integration to non-integer orders. Fractional calculus provides a framework to describe and model complex phenomena that exhibit memory, long-range dependence, and anomalous behavior [22]. The connection between fractals and fractional calculus lies in their shared ability to describe and analyse complex systems that

cannot be fully captured by integer-order calculus. Fractal geometry can be used to characterize fractal patterns, while fractional calculus provides a mathematical tool to analyse and model the dynamics of systems with fractal-like behavior. In some cases, fractional calculus is applied to describe the behavior of fractal objects or systems. For instance, fractional differential equations can be used to model phenomena with fractal properties [23].

3.2.1 Fractal Dimension and Complexity Measures

In recent years, fractals have had significant advantages from both theoretical and practical points of view. Digital signal processing (DSP) has recently witnessed the emergence of fractals as a promising domain of interest. Methods for DSP have evolved significantly due to the diverse applications of DSP in areas such as data transmission, speech and image recognition/translation, and biomedical applications [25]. Many of these methods treat the digital signals at a single-scale or mono-scale. Recently, multi-scale and polyscale analyses have been gaining attention because they provide more insight into the internal structure of the data. However, these emerging analyses require much attention to detail to achieve good results. One of the most applicable polyscale methods is based on the concepts of single and multiple power laws in the form of complexity measures, including fractal dimensions, as many physical processes have the characteristics of fractals [26].

Variance and length fractal dimensions are two important polyscale measures used in many data analysis areas [27, 28]. These polyscale methods and related algorithms were introduced and developed by Kinsner in 1994 [29]. Unlike mono-scale analysis, polyscale analysis measures a signal at multiple scales. It considers the measures from each scale cover simultaneously. If the time series is non-stationary, the analysis can be done within a moving stationarity frame to form a complexity measure trajectory. For example, the variance fractal dimension (VFD) has been used in applications such as non-stationary speech segmentation [30]. Multi-scale and polyscale analysis and synthesis using VFD in cognitive systems were introduced in [31]. Feature extraction of DNA sequences based on multi-fractal analysis was performed in [32]. Another important advantage of VFD analysis is that it is not sensitive to noise and can be applied in real-time streaming data [27]. Other applications include the detection of anomalies in (i) traffic analysis, specifically in distributed denial of service attacks [33], and (ii) nonlinear hydraulic actuators used in heavy-duty machinery [34].

This section introduces polyscale analysis for characterizing the complexity of time series using the length and variance fractal dimensions. These measures are used in Chapter 4 as feature extraction tools for fault detection.

3.2.2 Self-Affine Time Series

Fractal behavior in time series is described using self-affine processes, which exhibit scale-invariant properties across different time scales [27]. Two well-known classes of such processes include white Gaussian noise, characterized by a stochastic self-similarity fractal dimension $D_{SS} = 2$, and Brownian noise, with fractal dimension $D_{SS} = 1.5$. These processes illustrate different levels of signal complexity.

Self-affine time series can be characterized from both frequency-domain and time-domain perspectives. In the frequency domain, they exhibit a power-law behavior in the power spectral density (PSD), given by

$$\text{PSD} \sim \frac{1}{f^{\varrho}} \tag{3.10}$$

where ϱ is the spectral exponent. In the time domain, the scaling behavior is described by the Hurst exponent H_r, which characterizes long-term dependence in the signal. For self-affine processes, the variance of increments follows the power-law relation

$$\text{Var}[y(t_r) - y(t_s)] \sim |t_r - t_s|^{2H_r} \tag{3.11}$$

from which the Hurst exponent can be inferred.

The fractal dimension is directly related to the Hurst exponent. For a one-dimensional time series, the VFD is given by

$$D_{\sigma} = 2 - H_r \tag{3.12}$$

indicating that signals with higher irregularity correspond to larger fractal dimensions.

3.2.3 Length and Variance Fractal Dimensions

Fractal dimension is a quantitative measure used to describe the complexity and scaling behavior of signals. For self-affine time series, fractal dimensions provide insight into the irregularity, variability, and structural properties of the signal across multiple scales. Among the commonly used measures, the length fractal dimension (LFD) and the VFD are particularly effective for analyzing stochastic and nonlinear signals.

To compute LFD, the time series $y(j)$ is analyzed using multiple scale factors, referred to as volume elements (vels), denoted by n_k. A typical choice is

$$n_k = \left\lfloor 2^{\frac{k+5}{4}} \right\rfloor, \quad k = 1, 2, \ldots, K \tag{3.13}$$

where $\lfloor \cdot \rfloor$ denotes the floor function.

For each scale n_k, the signal is divided into subsequences, and the total length is computed as

$$L_{k,m} = \sum_{j=1}^{J_k} |y[m + jn_k] - y[m + (j-1)n_k]| \tag{3.14}$$

where the number of segments is defined as

$$J_k = \left\lfloor \frac{N - n_k}{n_k} \right\rfloor \tag{3.15}$$

and N is the total number of samples.

The average length at each scale is then obtained as

$$L_{k,\text{ave}} = \frac{1}{n_k} \sum_{m=1}^{n_k} L_{k,m} \tag{3.16}$$

To estimate the fractal dimension, logarithmic pairs are formed as

$$\big(\log_2(n_k), \log_2(L_{k,\text{ave}})\big) \tag{3.17}$$

and a line is fitted to these points using a regression method, often based on robust techniques such as iteratively reweighted least squares (IRLS). The slope of the fitted line, denoted by s_Δ, is used to compute the LFD as

$$D_L = 1 - s_\Delta \tag{3.18}$$

where D_L denotes LFD. The VFD, on the other hand, is derived from the scaling behavior of the variance of signal increments over different time intervals. For a given time series $y(j)$, subsequences are constructed using different scale factors n_k, and the variance of the increments is computed as

$$V_{k,m} = \frac{1}{J_k - 1} \left[\sum_{j=1}^{J_k} (y[m + jn_k] - y[m + (j-1)n_k])^2 - \frac{1}{J_k} \left(\sum_{j=1}^{J_k} (y[m + jn_k] - y[m + (j-1)n_k]) \right)^2 \right] \tag{3.19}$$

The average variance across segments is then given by

$$V_{k,\text{ave}} = \frac{1}{n_k} \sum_{m=1}^{n_k} V_{k,m} \tag{3.20}$$

To estimate the scaling behavior, log-log pairs are formed as

$$\big(\log_2(n_k), \log_2(V_{k,\text{ave}})\big) \tag{3.21}$$

which follow an approximately linear trend for self-affine signals. The slope of the fitted line, denoted by s_Δ, is related to the Hurst exponent H_r, and the VFD is obtained as

$$D_\sigma = 2 - H_r = 2 - \frac{1}{2} s_\Delta \tag{3.22}$$

where D_σ denotes VFD. In practice, robust regression methods such as IRLS are used to reduce the influence of outliers and improve estimation accuracy. It is also important to ensure that the log-log relationship is approximately linear, as deviations may indicate non-stationary behavior or the presence of multi-fractal characteristics.

Overall, the length and variance fractal dimensions provide complementary perspectives on signal complexity. These measures are particularly useful for feature extraction and analysis of complex signals in engineering systems. To illustrate these complexity measures, a Brownian self-affine time series is considered. This signal has a known theoretical fractal dimension of $D_{SS} = 1.5$, making it a suitable benchmark for evaluating the accuracy and precision of the length and variance fractal dimension measures.

A Brownian noise time series is generated and shown in Fig. 3.1. The LFD is then computed using Eq. (3.18). The algorithm is repeated 100 times to assess its consistency. The results yield an average length fractal dimension of $D_L = 1.5006$ with a standard deviation of 0.0059. Figure 3.2 shows the fitted line obtained from the log–log plot. It demonstrates a strong linear relationship. Similarly, the VFD is evaluated using (3.22). Again, the computation is repeated 100 times. The average value obtained is $D_\sigma = 1.5001$ with a standard deviation of 0.0066. The corresponding fitted line using the IRLS is illustrated in Fig. 3.3, which also confirms the expected linear behavior.

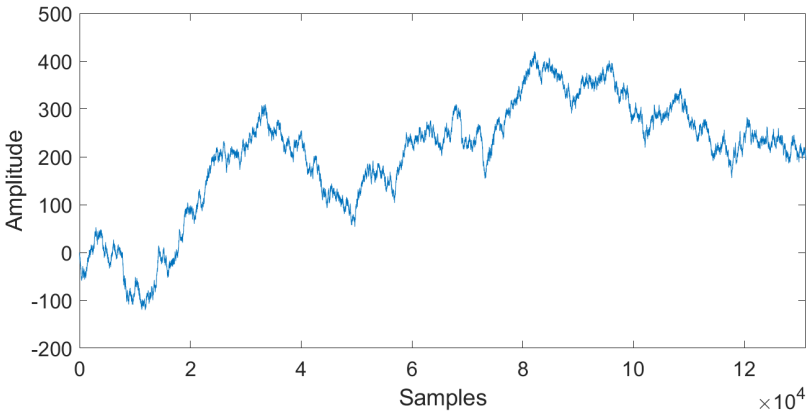


Figure 3.1: Generated Brownian self-affine time series.

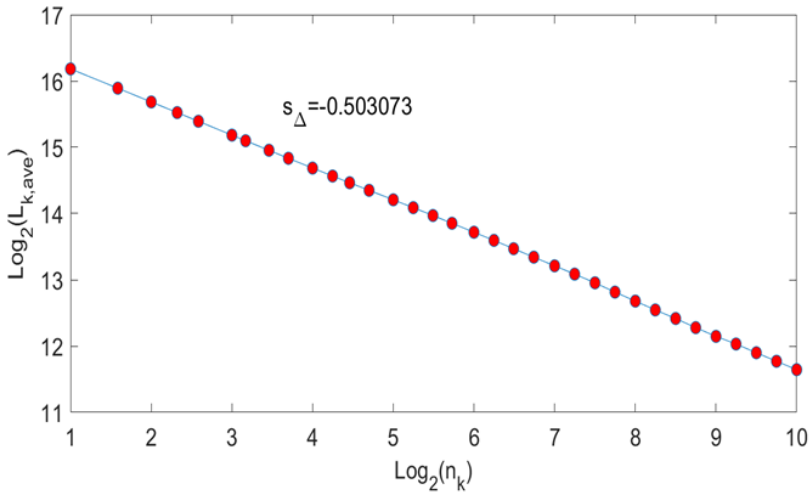


Figure 3.2: Log–log plot used for estimating the length fractal dimension.

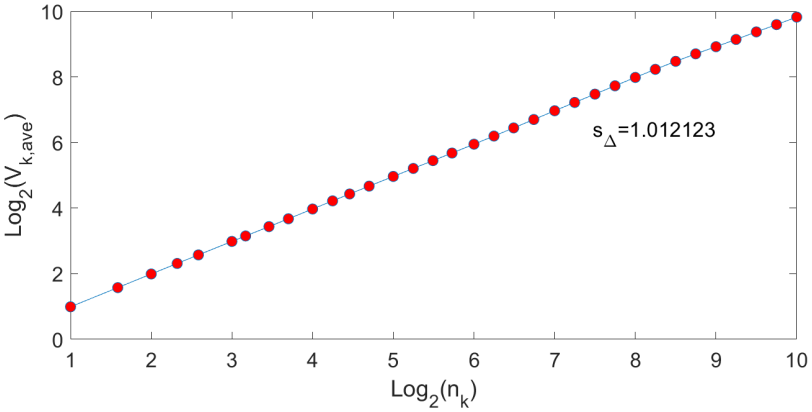


Figure 3.3: Log–log plot used for estimating the variance fractal dimension.

3.3 Optimization-Based PID Controller Tuning

In general, control strategies for EHA systems can be categorized into model-based and model-free approaches. In model-based control, the design relies on the mathematical representation of the system dynamics. Within this category, two main perspectives are commonly adopted: time-domain and frequency-domain approaches. Time-domain methods focus on shaping the transient response and reducing tracking error using error feedback signals.

In contrast, frequency-domain approaches characterize system behavior using the magnitude and phase of the system response, providing direct insight into stability and robustness. This perspective enables the formulation of controller tuning as an optimization problem, where the desired frequency response is explicitly defined and matched with the actual system behavior.

On the other hand, model-free control strategies operate without system models and instead adjust controller parameters based on observed data. These approaches are especially useful in systems with uncertain or time-varying dynamics. In this thesis, both model-based and model-free learning methods are studied in Chapter 5.

In addition, energy efficiency plays a critical role in EHA systems, where control actions directly influence energy consumption [35]. Energy-efficient design can be formulated as an optimization problem that balances tracking performance and energy usage. This aspect is investigated in detail in Chapter 6 for the EHA system integrated with an energy efficiency mechanism.

Proportional–integral–derivative (PID) controllers remain among the most widely used control strategies in industrial applications due to their simple structure, robustness, and ease of implementation [36]. Despite their simplicity, selecting appropriate controller gains is a challenging task, particularly for nonlinear systems, uncertain dynamics, and systems operating under varying conditions. Consequently, numerous PID tuning methods have been developed, ranging from heuristic and empirical techniques to advanced numerical and optimization-based methods.

The standard PID controller can be expressed as

$$u(t) = K_p e(t) + K_i \int_0^t e(\varsigma) d\varsigma + K_d \frac{de(t)}{dt} \tag{3.23}$$

where $u(t)$ is the control input, $e(t)$ is the tracking error, and K_p, K_i, and K_d denote the proportional, integral, and derivative gains, respectively.

The controller tuning problem can therefore be interpreted as an optimization problem in which the objective is to determine suitable values of the controller parameters that satisfy desired performance specifications such as tracking accuracy, robustness, settling time, overshoot, disturbance rejection, and energy consumption.

One of the earliest and most commonly used PID tuning methods is the heuristic or empirical tuning method. These methods rely on practical rules, operator experience, or experimentally

derived formulas rather than solving a formal optimization problem. One of the most widely used heuristic tuning methods is the Ziegler–Nichols method [37].

Heuristic methods are attractive because of their simplicity, low computational cost, and ease of implementation in industrial environments where accurate mathematical models may not be available. However, their performance is often limited for nonlinear or time-varying systems, and the resulting controller gains may produce oscillatory responses or insufficient robustness under changing operating conditions. In addition, heuristic approaches generally provide limited flexibility when multiple design objectives must be considered simultaneously. To overcome some of these limitations, optimization-based PID tuning methods formulate the controller design problem as a mathematical optimization problem in which the controller gains are determined by minimizing an objective function. For PID tuning, the decision vector is commonly defined as

$$\mathbf{x} = [K_p, K_i, K_d]^T \tag{3.24}$$

The objective function may represent various performance criteria such as the integral absolute error (IAE), integral squared error (ISE), or integral time-weighted absolute error (ITAE) [38].

Compared to heuristic methods, optimization-based approaches provide a more systematic tuning framework while allowing performance constraints and multiple design requirements to be incorporated directly into the optimization problem. Among optimization approaches, gradient-based numerical optimization methods are commonly employed when the system model is available and differentiable. These methods iteratively improve the controller parameters using gradient information in order to approach an optimal solution.

One widely used numerical optimization technique is sequential quadratic programming (SQP), which is regarded as an effective method for constrained nonlinear optimization problems [39]. The fundamental idea of SQP is to solve a sequence of quadratic programming subproblems that locally approximate the original nonlinear optimization problem. Given the nonlinear optimization problem

$$\min_{\mathbf{x}} \; J(\mathbf{x}) \tag{3.25}$$

subject to

$$g(\mathbf{x}) \leq 0 \tag{3.26}$$

The SQP method solves the problem iteratively. At the k-th iteration, let $\mathbf{x}_k$ denote the current estimate of the decision variable vector. The objective and constraint functions are locally approximated around $\mathbf{x}_k$ by constructing a quadratic approximation of the Lagrangian function

$$\mathcal{L}(\mathbf{x}, \lambda) = J(\mathbf{x}) + \lambda^T g(\mathbf{x}) \tag{3.27}$$

where λ denotes the vector of Lagrange multipliers. The resulting quadratic subproblem can then be written as

$$\min_{\mathbf{d}} \ \frac{1}{2}\mathbf{d}^T H_k \mathbf{d} + \nabla J(\mathbf{x}_k)^T \mathbf{d} \tag{3.28}$$

subject to

$$g(\mathbf{x}_k) + \nabla g(\mathbf{x}_k)^T \mathbf{d} \leq 0 \tag{3.29}$$

where $\mathbf{d}$ is the search direction and H_k approximates the Hessian matrix of the Lagrangian. The solution of the quadratic subproblem determines the search direction, which is then used to update the current estimate of the decision variables. SQP provides fast convergence and efficient handling of nonlinear constraints, making it attractive for controller design problems involving smooth objective functions. In Chapter 5, the SQP method is used to numerically solve the optimization problem and determine the PID controller gains.

To address some limitations of gradient-based optimization, metaheuristic optimization methods have gained considerable attention in PID tuning applications. These approaches are inspired by natural, biological, or evolutionary processes and include techniques such as genetic algorithm (GA), particle swarm optimization (PSO), differential evolution (DE), and simulated annealing (SA) [40]. Unlike gradient-based methods, metaheuristic approaches do not require derivative information and are generally more suitable for nonlinear and multimodal optimization problems. Their global search capability reduces the likelihood of becoming trapped in local minima, although this advantage is often achieved at the expense of increased computational complexity and slower convergence.

In many practical control applications, multiple conflicting objectives must be considered simultaneously. For example, in EHA systems, improving tracking accuracy may increase energy consumption or control effort, while reducing settling time may result in larger overshoot. Traditional single-objective optimization methods commonly combine multiple objectives into a weighted-sum formulation given by

$$J = w_1 J_1 + w_2 J_2 + \cdots + w_n J_n \tag{3.30}$$

where J_i represents the i-th objective and w_i denotes the corresponding weighting coefficient. Although weighted-sum methods are relatively simple to implement, selecting appropriate weighting coefficients can be difficult and may significantly influence the final solution. In addition, weighted formulations may fail to capture non-convex trade-off solutions and often require repeated optimization runs to investigate different objective priorities.

To overcome these limitations, multi-objective optimization methods explicitly consider multiple objectives simultaneously without reducing them to a single scalar objective [41]. A general

multi-objective optimization problem can be expressed as

$$\min_{\mathbf{x}} \ \mathbf{F}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})]^T \tag{3.31}$$

subject to system constraints.

Since the objectives may conflict with one another, a unique global optimum generally does not exist. Instead, the solution is characterized using the concept of Pareto optimality. Consider two candidate solutions $\mathbf{x}_a$ and $\mathbf{x}_b$. Solution $\mathbf{x}_a$ is said to dominate $\mathbf{x}_b$ if

$$f_i(\mathbf{x}_a) \leq f_i(\mathbf{x}_b) \qquad \forall i \tag{3.32}$$

and

$$f_j(\mathbf{x}_a) < f_j(\mathbf{x}_b) \qquad \text{for at least one } j \tag{3.33}$$

A solution is considered Pareto optimal if no other feasible solution dominates it. The collection of all Pareto optimal solutions forms the Pareto set, while the corresponding representation in the objective space is referred to as the Pareto front.

Compared with weighted-sum approaches, multi-objective optimization provides a more comprehensive representation of the trade-offs between conflicting objectives and generates multiple candidate solutions for engineering decision-making. This characteristic is particularly important in PID controller tuning, where tracking performance, robustness, energy efficiency, overshoot reduction, and control effort may all need to be considered simultaneously. In Chapter 6, a multi-objective optimization framework is proposed to simultaneously improve tracking performance and reduce energy consumption in the EHA system equipped with the energy storage and reutilization unit.

3.4 Summary

This chapter presented the preliminary concepts required for the development of the interpretable methodologies used in this thesis. First, the Bayesian framework was introduced as a probabilistic approach for learning from data under uncertainty. The roles of prior, likelihood, and posterior distributions were discussed, together with the distinction between deterministic and probabilistic inference. Based on this foundation, Gaussian processes were described as nonparametric Bayesian models for representing unknown functions, and Bayesian optimization was introduced as a sequential strategy for optimizing expensive black-box functions through surrogate modelling and acquisition functions.

Next, the chapter introduced fractal concepts and complexity measures for time-series analysis.

The notions of self-affine signals, power-law scaling, and the Hurst exponent were discussed to provide the theoretical basis for fractal complexity measures. In this context, the length and variance fractal dimensions were presented. These concepts establish the basis for feature extraction from complex measurement signals in later chapters. Finally, an overview of optimization-based approaches for PID controller tuning in EHA systems was provided, including heuristic, numerical, and optimization-based methods.

Chapter 4

Condition Monitoring

One of the major issues in hydraulic systems is internal leakage in actuators. Leakage typically occurs through the seals separating the cylinder chambers, resulting in pressure drop, reduced actuator force, and increased energy consumption. Detecting internal leakage is often challenging without disconnecting the actuator from the system or disassembling the cylinder and piston.

In this context, condition monitoring has emerged as an effective approach for assessing the health of hydraulic systems using measurable signals during operation. By continuously analyzing system behavior, condition monitoring enables early detection of faults without interrupting system functionality. Therefore, it is highly desirable to identify internal leakage through condition monitoring techniques based on easily obtainable signals.

Various approaches have been proposed for fault detection in hydraulic systems, ranging from model-based methods to data-driven techniques. Tan and Sepehri [48] employed a Volterra nonlinear model to identify actuator leakage faults, where detection relied on an explicit leakage model. Yan *et al.* [49] investigated fault diagnosis of hydraulic pumps using convolutional neural networks, demonstrating the capability of deep learning methods in capturing complex system dynamics. Shi *et al.* [50] utilized a linearized model combined with an adaptive threshold for internal leakage detection. In addition, a comprehensive review of signal processing-based fault detection methods for hydraulic systems is presented in [51].

Signal processing techniques play a fundamental role in condition monitoring, as they enable the extraction of informative features from raw system signals. These features can then be used as inputs for machine learning classifiers such as artificial neural networks, support vector machines, and decision trees [52]. For instance, internal leakage detection in EHA systems has been investigated in [65] using multiscale analysis of experimental signals. Ghanbari *et al.* [66] utilized actuator velocity and force measurements in conjunction with artificial neural networks for fault detection.

This chapter focuses on condition monitoring of the EHA system, with focusing on the detection of actuator internal leakage faults. First, a data-driven framework is developed in which

informative features are extracted from cylinder chamber pressure signals using complexity measures. These features are then used in a Bayesian classification framework for fault detection. Next, an interpretable modelling method for fault detection is introduced based on sparse identification of nonlinear dynamics (SINDy), in which Bayesian optimization is used to tune model parameters under limited data conditions. The work presented in this chapter is based on the results published in these papers[1,2].

4.1 Fault Detection Using Bayesian Classification

Conventional machine learning classifiers have been widely used for fault detection and probabilistic classifiers such as logistic regression can estimate conditional class probabilities [53]. However, conventional point-estimate formulations use a fitted parameter vector without explicitly representing uncertainty in those parameters. Parameter uncertainty can be particularly relevant in condition monitoring applications where measurements are noisy and available data are limited.

Bayesian inference provides a principled framework for incorporating uncertainty into the classification model. In this approach, unknown model parameters are treated as random variables, and prior knowledge is combined with observed data to obtain posterior distributions [54]. As a result, the model is able to quantify uncertainty and update its predictions as new data becomes available. This capability offers significant advantages in scenarios with limited or highly variable experimental data. In such cases, relying solely on data-driven classification may lead to biased or unreliable results. By contrast, Bayesian methods improve robustness by integrating prior information with observed data, leading to more reliable fault detection in EHA systems.

Bayesian inference may also be affected by thick-tailed distributions and long-term correlations in the system under study. Extreme events or outliers occur in thick-tailed distributions more than in Gaussian distributions. The properties of thick-tailed distributions are important in statistical inference [54]. In Bayesian inference, prior probability distributions are chosen based on our knowledge of the system under study before collecting data. One can choose a prior distribution with thicker tails if the model parameters are known to have long-term correlations. This ensures that the obtained posterior distribution is more likely to be accurate and truly reflects the nature of the system.

Inspired by the inherent capabilities of Bayesian inference, Bayesian classification serves as a versatile tool for classification tasks. It can be used for both binary and multi-class classification

[1] S. Hosseinpour, W. Kinsner, S. Muthukumarana, and N. Sepehri, "Fault Detection in an Electro-Hydrostatic Actuator Using Polyscale Complexity Measures and Bayesian Classification," *IEEE Open Journal of Instrumentation and Measurement*, vol. 3, 2024.

[2] S. Hosseinpour, W. Kinsner, and N. Sepehri, "A Machine Learning Model of Electro-Hydrostatic Actuators With the Low-Data Limit and Its Application to Fault Detection," *IEEE Access*, 2025.

problems and has been successfully applied in various engineering domains [55]. In this context, it has also been explored for fault detection applications. For instance, a fault detection and identification method was done in [56] using a Kullback–Leibler divergence and Bayesian inference for plant-wide process monitoring. A new method for utilizing the prediction uncertainty associated with Bayesian deep learning models was introduced in [57] for an enhanced fault detection method.

While Bayesian classification is advantageous for its probabilistic insights, feature extraction can help improve model performance. Feature extraction can help improve model performance. It involves creating a new set of features that captures useful characteristics of the raw data [58, 59]. Instead of using raw data that often contains redundant or noisy features, extracted features can be fed into the machine learning models. This can enhance the performance of classification algorithms significantly by avoiding overfitting and improving the generalization of new or unseen data [60].

Most of the feature extraction techniques are based on monoscale or multiscale methods for time series data. They include common statistical moments and wavelet parameters. Monoscale methods focus on identifying and extracting features at a single scale [61]. While they may be considered simple, they may not be practical when dealing with time series data [62]. In fact, these methods are not sensitive to the reshuffling of the data, and critical features like correlation and covariance are lost. Multiscale methods, like wavelets, split the data into different scales, but they do not utilize the information about the multiple scales. Polyscale analysis, on the other hand, measures a signal at multiple scales by considering the measurements obtained from each scale simultaneously. Time series data may, therefore, benefit from features extracted using polyscale methods.

Polyscale complexity measures, such as the length and variance fractal dimension (VFD) [62], are vital tools for analyzing signals with long-range dependence. The polyscale measures have been applied to dynamical systems and have been successfully applied in various fields, including machine learning, image processing, natural language processing, and computer vision [63, 64]. To have meaningful results, the considered signal should be stationary. For non-stationary signals, the analysis can be performed within a moving stationarity frame, which generates a complexity measure trajectory. The polyscale feature extraction can also be viewed as a type of compressive detection approach [63]. In order to extract features from signals using VFD, it is necessary to treat the entire signal. This can be accomplished only after denoising the signal. In fact, the power-law may vary over the range considered for a raw signal contaminated by noise, which may imply a multifractal. Using the polyscale measures, denoised signals can be classified by the Bayesian classification approach into normal or faulty categories.

This section investigates a new method for actuator's internal leakage fault detection in EHAs. Several types of signals can be used for fault detection in hydraulic systems. Some factors (such as cost and sensitivity to a fault) should be considered to determine which signals are the most appropriate for the actuator leakage fault detection. In this section, the pressures in both the cap and

rod sides of the hydraulic actuator are utilized for feature extraction. The method is based on the polyscale complexity measures and Bayesian inference. In doing this, a feature extraction method is developed based on VFD to be employed in a Bayesian logistic regression model. An uninformative prior distribution is assigned to the model parameters for Bayesian inference.

4.1.1 Data Acquisition

A collection of 48 datasets was extracted, including healthy and faulty ones, with the faulty datasets categorized into three levels of the actuator's internal leakage: small, medium, and high. The leakage level varies from 0 (L/min) to a maximum of 8 (L/m) depending on operating conditions. Experiments were also done under three different levels of load: no load ($m_L = 0$), medium load ($m_L = 183.5$ kg), and high load ($m_L = 367$ kg). The study involves the utilization of both open-loop and closed-loop velocity controls using step and joystick inputs.

In order to cite each dataset, an abbreviation system is employed using a five-letter code. The first letter, either O or C, represents "open-loop" or "closed-loop," respectively. The second letter corresponds to the load level and is denoted by N, M, or H, representing "no load," "medium load," and "high load," respectively. The third letter represents the level of leakage: N, S, M, and H, indicating "no leakage," "small leakage," "medium leakage," and "high leakage," respectively. The last two letters signify the input, either joystick input (Jo) or step input (St) (see Fig. 4.1). The joystick inputs were generated manually by an operator to explore different actuator motions, rather than to follow a prescribed trajectory. Different input signals were used for the different leakage conditions; no identical recorded signal was replayed across these tests.

For illustration purposes, four typical datasets were chosen, each presenting 50 seconds of experimental results: (1) open-loop control with medium load and no leakage using joystick input (OMNJo), (2) open-loop control with medium load, small leakage using joystick input (OMSJo), (3) open-loop control with medium load, medium leakage using joystick input (OMMJo), and (4) open-loop control with medium load, high leakage using joystick input (OMHJo). Figures 4.2–4.5 relate to four representative datasets. These datasets represent system performance experiencing various levels of actuator leakage. Figure 4.2 shows the input, position, velocity, leakage, and chamber pressure signals, P_A and P_B, for the OMNJo dataset, which represents the system under joystick control.

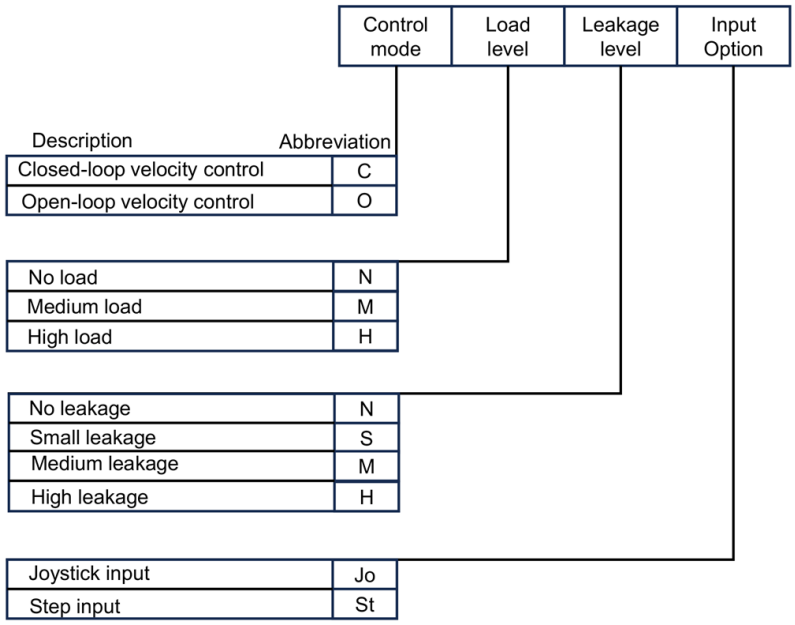


Figure 4.1: Abbreviations used for the datasets.

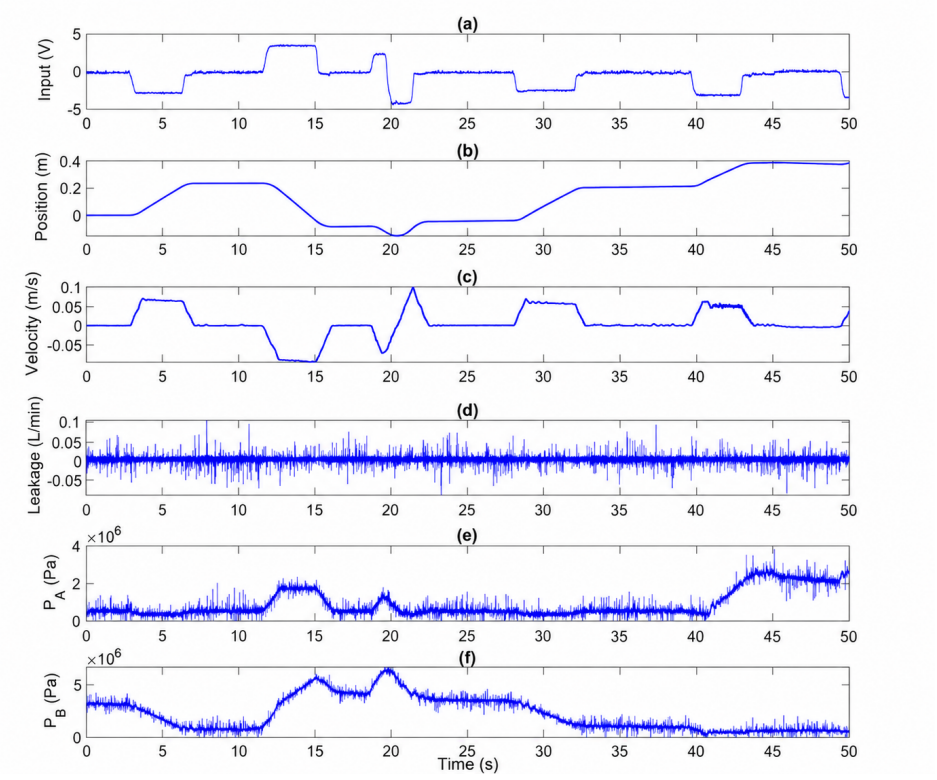


Figure 4.2: Typical test results for dataset OMNJo: (a) input signal u; (b) position x_p; (c) velocity $\dot{x}_p$; (d) leakage level; (e) measured pressure P_A; (f) measured pressure P_B.

Figures 4.3 and 4.4 show the measured pressure signals P_A and P_B for four datasets: OMNJo, OMSJo, OMMJo, and OMHJo.

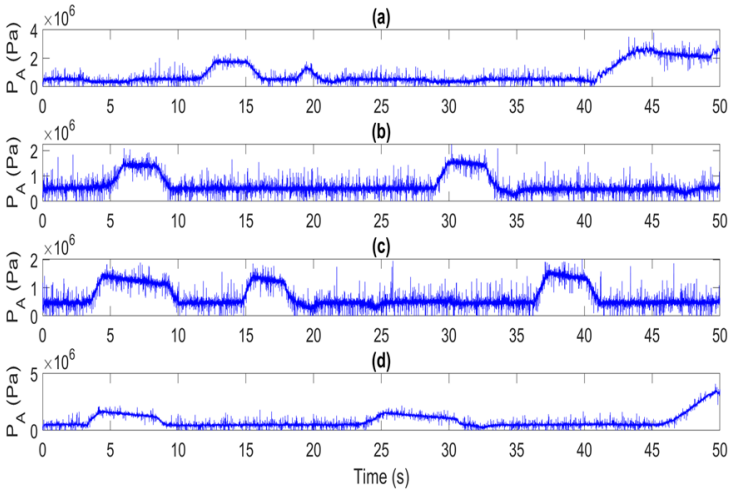


Figure 4.3: Measured pressure signal P_A for various datasets: (a) OMNJo; (b) OMSJo; (c) OMMJo; (d) OMHJo.

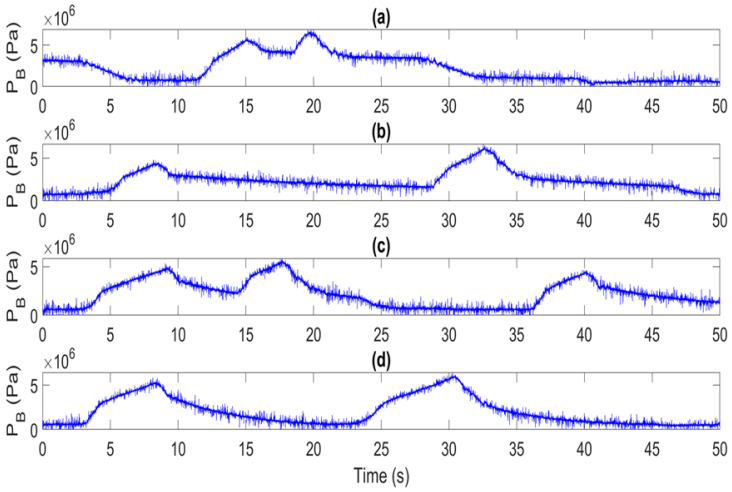


Figure 4.4: Measured pressure signal P_B for various datasets: (a) OMNJo; (b) OMSJo; (c) OMMJo; (d) OMHJo.

Figure 4.5 shows measured leakage values for the same four datasets.

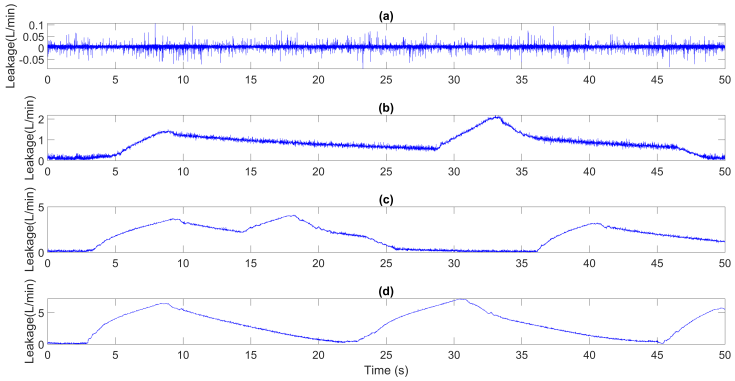


Figure 4.5: Typical fault levels: (a) no leakage dataset OMNJo; (b) small leakage dataset OMSJo; (c) medium leakage dataset OMMJo; (d) high leakage dataset OMHJo.

4.1.2 Noise in Pressure Reading and Denoising Process

The initial step in denoising involves examining the noise frequency components present in the signals and identifying the maximum frequency component of the system. Consider the pressure reading data of the experiments shown in Figs. 4.3 and 4.4. Since the pressure signals are broadband, instead of the -3 dB cut-off frequency, f_c, the highest frequency components are used for the sampling test analysis [68]. The highest frequency component of the pressure signals is approximately $f_h \approx 9.3$ Hz [68]. This is further shown in Fig. 4.6 for the pressure signal P_A from datasets OMNJo, CMNJo, and CNNJo.

Setting f_c equal to f_h might seem ideal to preserve high frequencies, but it causes maximum phase distortion at this point, potentially corrupting the signal. It is important to note that the classification of noise in the pressure signal into broadband, narrowband, and frequency content requires extensive discussion, which is beyond the scope of this thesis. The commonly used first-order low-pass filter (LPF) is used here [67]

$$\mathrm{LPF}(s) = \frac{1}{\left(\frac{1}{2\pi f_c}\right) s + 1} \tag{4.1}$$

With respect to Eq. (4.1), f_c is set to 15 Hz, which is higher than f_h to minimize phase distortion. The sampling frequency, f_s, is set to 1000 Sps, which is above the required Nyquist frequency.

Using the LPF method, the pressure signals P_A and P_B are denoised. Figures 4.7 and 4.8 show the denoised pressure signals P_A and P_B for the datasets OMNJo, OMSJo, OMMJo, and OMHJo.

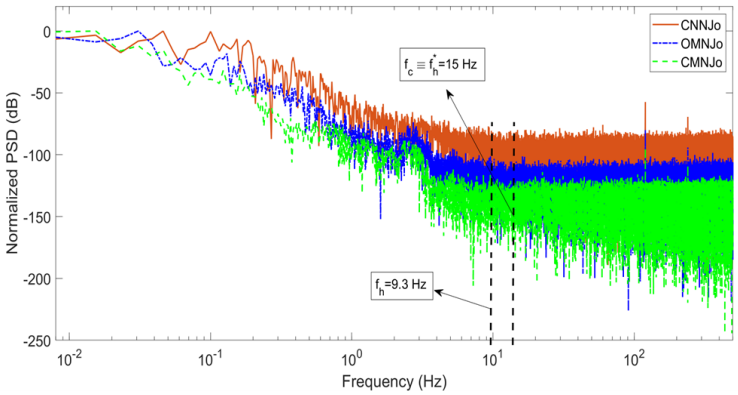


Figure 4.6: Power spectral density analysis of pressure signal P_A, for datasets OMNJo, CMNJo, and CNNJo.

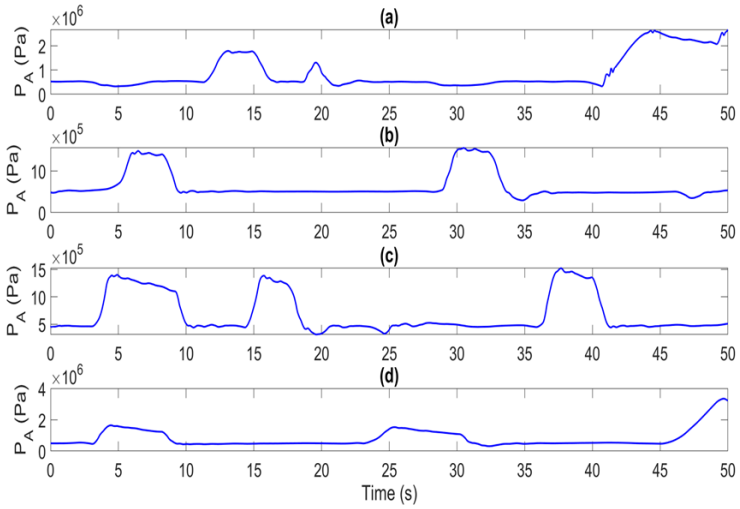


Figure 4.7: Denoised pressure signal P_A: (a) OMNJo; (b) OMSJo; (c) OMMJo; (d) OMHJo.

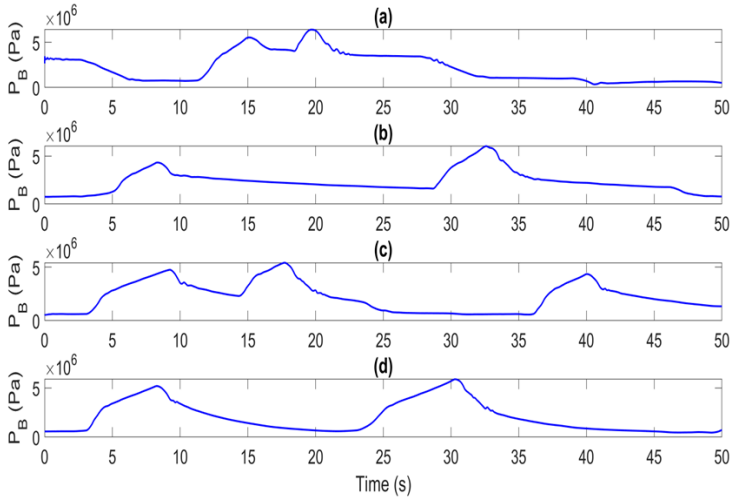


Figure 4.8: Denoised pressure signal P_B: (a) OMNJo; (b) OMSJo; (c) OMMJo; (d) OMHJo.

4.1.3 Feature Extraction

In order to perform Bayesian classification on pressure signals, relevant features are extracted from the collected data. Since the pressure signals exhibit the characteristics of time series data, to analyze and extract useful information from these signals properly, a feature extraction method appropriate for time series data must be employed. Here, the principle of polyscale methods is utilized to extract features that can be used for fault detection. In this thesis, the VFD is calculated for the whole record length after denoising the pressure signals. The corresponding VFD is then obtained for each signal as a feature.

The VFD is employed using the method described in Chapter 3. First, the scale size needs to be defined. The following size selection is used

$$n_k = \left\lfloor 2^{\frac{k+5}{4}} \right\rfloor, \quad k = 1, 2, \ldots, 35 \tag{4.2}$$

where $\lfloor \cdot \rfloor$ is the floor function. The above 35 scale sizes are considered to obtain the VFD for this application. The variance of each subsequence $V_{k,m}$ is computed for the corresponding scale size n_k in Eq. (4.2) as

$$V_{k,m} = \frac{1}{J_k - 1} \left[\sum_{j=1}^{J_k} \left(y[m + jn_k] - y[m + (j-1)n_k] \right)^2 - \frac{1}{J_k} \left(y[m + jn_k] - y[m + (j-1)n_k] \right)^2 \right] \tag{4.3}$$

where

$$J_k = \left\lfloor \frac{N_F - n_k}{n_k} \right\rfloor, \quad m = 1, 2, \ldots, n_k, \quad n_k \in \mathbb{N} \tag{4.4}$$

N_F is considered to be equal to the number of samples in the entire extracted signal in order to obtain the VFD of the signal. The average variance over all subsequences is then given by

$$V_{k,\text{ave}} = \frac{1}{n_k} \sum_{m=1}^{n_k} V_{k,m} \tag{4.5}$$

For each scale n_k, the pairs $(\log_2(n_k), \log_2(V_{k,\text{ave}}))$ are obtained. A line is then fitted using the robust linear regression method known as iteratively reweighted least squares (IRLS) [69]. This is performed using the `robustfit` function in MATLAB software. Finally, the VFD is determined using the experimentally obtained slope s_Δ, which provides an estimate of the scaling behavior. By considering $H = \frac{1}{2} s_\Delta$ and $D_\sigma = E + 1 - H$, the VFD is calculated accordingly.

Figure 4.9 illustrates a scatter plot of all 48 datasets, providing a clearer understanding of how the values for computed VFD values are distributed for pressures P_A, P_B, and velocity v_p. The dataset is categorized into two main groups: healthy and faulty. The faulty group includes data from varying degrees of leakage (small, medium, and high). Circles indicate healthy signals, while squares denote all faulty signals, regardless of the fault level. The plots reveal that the VFD of the pressure signals is consistently lower in faulty cases, suggesting a decrease in signal complexity compared to healthy ones. This pattern can be explained by understanding that leakage can reduce the system's working pressure, which may also affect the variability of the pressure signals. This demonstrates that VFD values for pressures P_A and P_B effectively distinguish between healthy and faulty conditions.

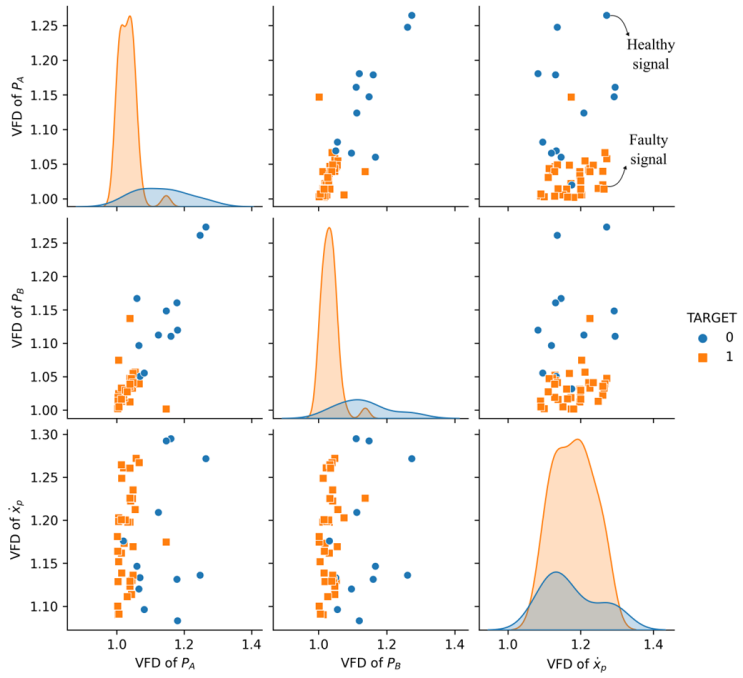


Figure 4.9: Scatter plots of variance fractal dimension (VFD) for pressure signals P_A and P_B, and velocity $\dot{x}_p$. The target value of zero shows healthy, and one shows faulty signals.

Figures 4.10 and 4.11 show the log–log plot of $V_{k,\text{ave}}$ versus n_k for dataset OMNJo, along with the corresponding fitted lines. As can be seen, the pairs form a line, allowing us to calculate the slope of the fitted line. If a line were not formed, these extracted features could not be used for the classification method.

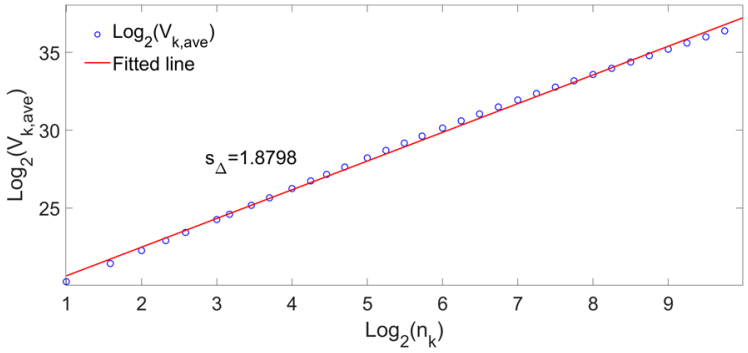


Figure 4.10: Log–log plot of $V_{k,\text{ave}}$ versus n_k for pressure signal P_A, in dataset OMNJo.

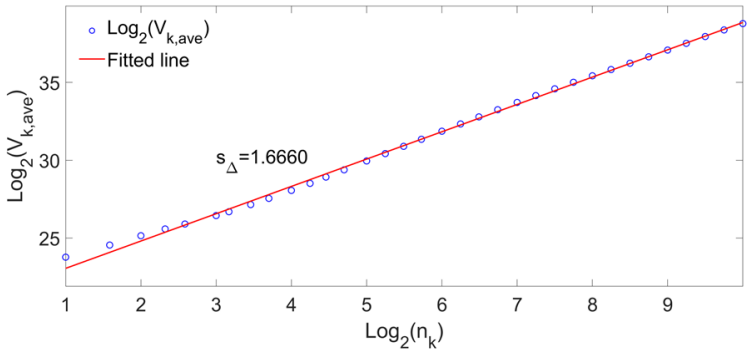


Figure 4.11: Log–log plot of $V_{k,\text{ave}}$ versus n_k for pressure signal P_B, in dataset OMNJo.

Figures 4.12 and 4.13 display the log–log plot for pressure signals P_A and P_B for datasets OMSJo, OMMJo, and OMHJo. These three signals are categorized as faulty signals. It can be observed that the slopes of the lines for these three datasets are similar, which can aid the classification method in distinguishing between healthy and faulty signals. Based on the extracted VFD features, a Bayesian classification method is developed for fault detection. Table 4.1 shows the mean and standard deviation of VFD values for healthy and faulty signals across different sampling intervals. Notably, as the sampling interval increases, the standard deviation for both P_A and P_B tends to decrease. This suggests that longer sampling intervals provide more stable VFD values. Nevertheless, the difference between healthy and faulty signals in terms of VFD mean values remains consistent across intervals. In this study, a sampling interval of 50 seconds is considered for the proposed Bayesian classification.

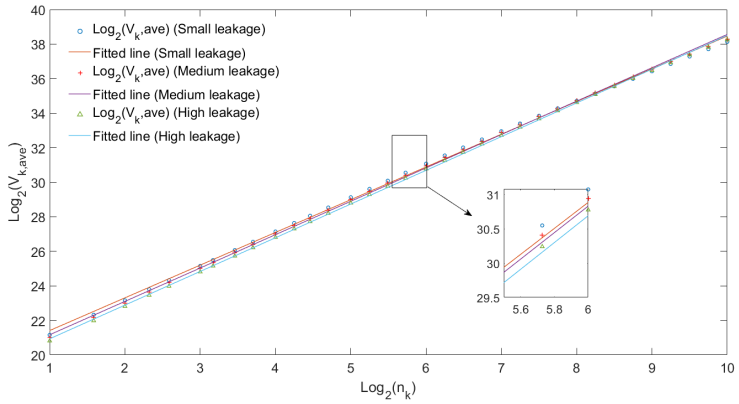


Figure 4.12: VFD of the pressure signal P_A, for different levels of leakage.

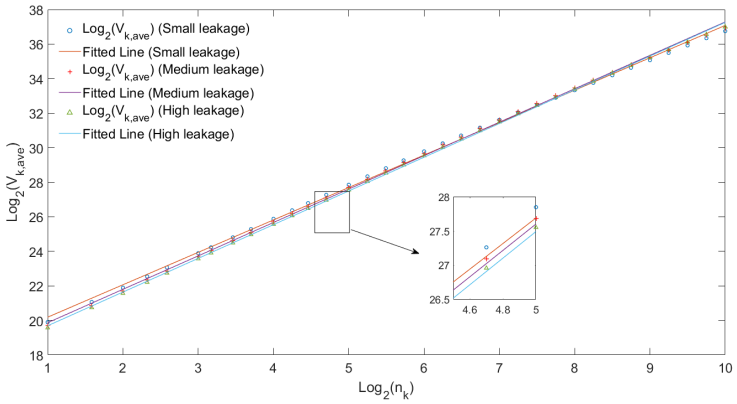


Figure 4.13: VFD of the pressure signal P_B for different levels of leakage.

Table 4.1: Mean and standard deviation of variance fractal dimension (VFD) values for actuator pressures, P_A and P_B, across different sampling intervals. H and F stand for healthy and faulty signals.

Sampling Interval (s)	Mean of VFD				Standard deviation of VFD			
	P_A		P_B		P_A		P_B	
	H	F	H	F	H	F	H	F
10	1.177	1.034	1.205	1.043	0.122	0.050	0.132	0.105
20	1.162	1.025	1.159	1.027	0.106	0.027	0.093	0.040
30	1.131	1.027	1.143	1.029	0.088	0.027	0.082	0.032
40	1.128	1.030	1.134	1.032	0.084	0.027	0.076	0.027
50	1.133	1.031	1.132	1.032	0.077	0.027	0.076	0.025

4.1.4 Bayesian Classification

Bayesian logistic regression is used as a machine learning method for fault detection based on Bayesian inference [71]. The primary goal is to find the posterior distribution of the regression coefficient parameters for the fault detection classification task. To achieve this, the VFD values of the pressure signals P_A and P_B are considered as features.

The study's dataset consists of four groups, each representing different leakage levels: no leakage, small leakage, medium leakage, and high leakage, with each group containing 12 datasets. To ensure a balanced and representative training and testing phase, signals from the "no leakage" and "small leakage" are selected. First, 16 datasets are used for training, while 8 datasets are reserved for testing. The training datasets aimed to train the Bayesian classifier to effectively distinguish between healthy and faulty conditions. The remaining 8 datasets, along with 24 additional datasets representing medium and high leakage levels, were then used for testing. This approach allows us to evaluate the classifier's effectiveness and robustness in detecting faults across a spectrum of scenarios.

Healthy and faulty datasets are separated into two binary categories, with binary values zero and one assigned to healthy and faulty signals, respectively. The binary response y_i is assumed to have a Bernoulli distribution with a probability of success p_i as

$$y_i \sim \text{Bernoulli}(p_i), \quad i = 1, 2, \ldots, M \tag{4.6}$$

where M indicates the number of training datasets considered for classification. The logistic regression model can be written as the linear function of the variables x_{1i} and x_{2i} as follows

$$\log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} \tag{4.7}$$

where β_0, β_1, and β_2 are regression coefficient parameters that need to be estimated using the Bayesian inference approach. x_{1i} and x_{2i}, $i = 1, 2, \ldots, M$, are the values of features extracted for pressure signals P_A and P_B, respectively. By rearranging the logistic regression model, Eq. (4.7) can express the regression as a nonlinear equation for the probability of success p_i as shown below

$$p_i = \frac{\exp(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i})}{1 + \exp(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i})} \tag{4.8}$$

Equation (4.8) guarantees that the probability p_i lies in the interval $[0, 1]$. A threshold criterion is then applied to p_i to classify the condition of the signal. If $p_i \geq 0.5$, the signal is categorized as faulty; otherwise, it is classified as healthy.

A prior distribution is needed for the set of regression coefficient parameters $(\beta_0, \beta_1, \beta_2)$. The prior distribution can be selected based on any prior information about the values of the parameters and by using the Maximum Entropy (Max-Ent) principle. In the absence of any significant prior information, the Max-Ent principle suggests using a non-informative prior, as proposed in [70]. In this study, independent prior distributions are assigned to the model parameters as follows

$$\begin{aligned} \beta_0 &\sim \text{dnorm}(0, 10^{-3}) \\ \beta_1 &\sim \text{dnorm}(0, 10^{-3}) \\ \beta_2 &\sim \text{dnorm}(0, 10^{-3}) \end{aligned} \tag{4.9}$$

where $\text{dnorm}(\mu, \tau)$ indicates the normal distribution, μ is the mean and τ is the precision $1/\sigma^2$. In Bayesian inference, a point estimation for the parameters needs to be considered. This means that a single statistic is used to summarize the posterior distribution of the data. The posterior mean or median is considered a good candidate. The posterior mean will be used as the Bayesian estimate in this thesis.

Since the posterior is not recognizable analytically, numerical techniques should be applied.

A Markov Chain Monte Carlo (MCMC) sampling algorithm is used here to explore the posterior distribution of the unknown parameters, β_0, β_1, and β_2 [72]. In MCMC, the stationary distribution of the Markov chain serves as the target probability distribution. Stationary distributions are probability distributions that remain unchanged over time, even though the Markov chain evolves. This property will be used to show that convergence to the stationary distribution is achieved by the employed MCMC method in the trace plots.

The most commonly used MCMC algorithm is the Metropolis–Hastings (M–H) algorithm, which uses a proposal distribution and an acceptance probability based on the ratio of the probability of the candidate sample to the probability of the current sample. In this thesis, the M–H algorithm is used for sampling [72].

The number of iterations used in the M–H algorithm is N = 200,000 samples, where the first 100,000 samples are not considered in the inference as the burn-in period to reduce transients. To assess the convergence of the algorithm to the stationary distribution, three chains are considered. Each chain in the MCMC algorithm starts from a different point in the parameter space and explores the posterior distribution through a sequence of iterations.

In the next subsection, the Bayesian parameter estimation results are presented, including summary statistics, graphical representations of the density functions, autocorrelations, and trace plots of the MCMC for three different chains. All analyses have been done in R software using the OpenBUGS library [73].

4.1.5 Posterior Distribution Analysis

The posterior distribution summary is a crucial component of Bayesian inference and provides a comprehensive and intuitive representation of the model parameters. Table 4.2 shows the estimates of the posterior mean, standard deviation (SD), median, and 95% credible interval (25th percentile and 97.5th percentile) for each parameter of interest for one of the chains.

Table 4.2: Statistics for Bayesian parameter estimation with 200,000 Markov chain Monte Carlo (MCMC) iterations.

Parameters	Mean	SD	2.5th percentile	Median	97.5th percentile
β_0	36.96	15.67	9.316	35.97	70.22
β_1	−32.34	15.61	−65.18	−31.51	−4.003
β_2	−2.648	13.81	−30.17	−2.573	24.5

Figures 4.14–4.16 show the trace plots for three chains in the MCMC process. The trace plots indicate that the MCMC algorithm has converged to the stationary distribution for all three chains as they are no longer changing significantly over time.

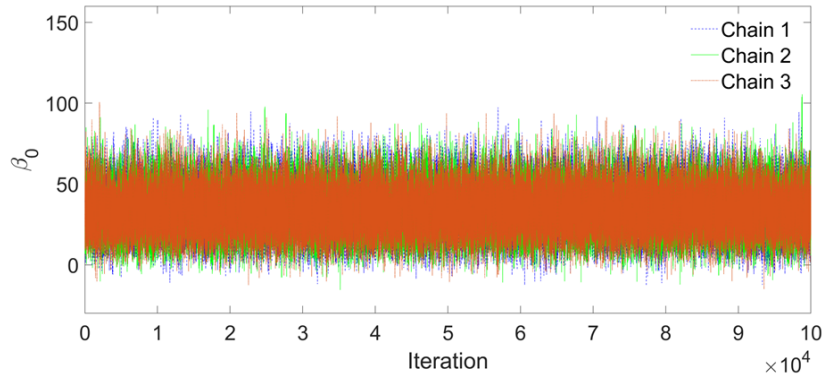


Figure 4.14: Trace plots of Markov chain Monte Carlo (MCMC) for Bayesian parameter β_0.

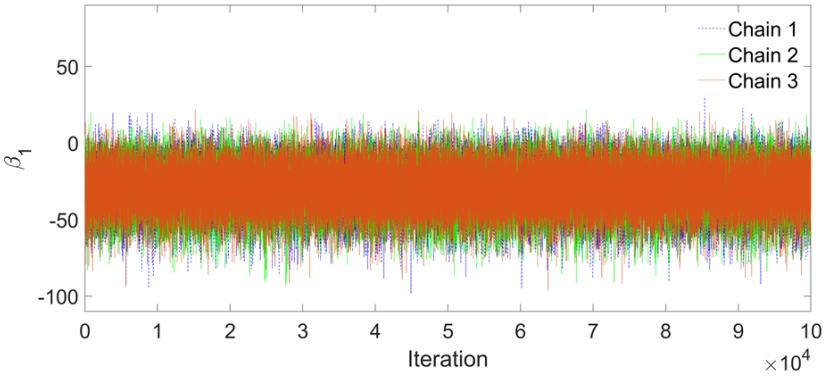


Figure 4.15: Trace plots of MCMC for Bayesian parameter β_1.

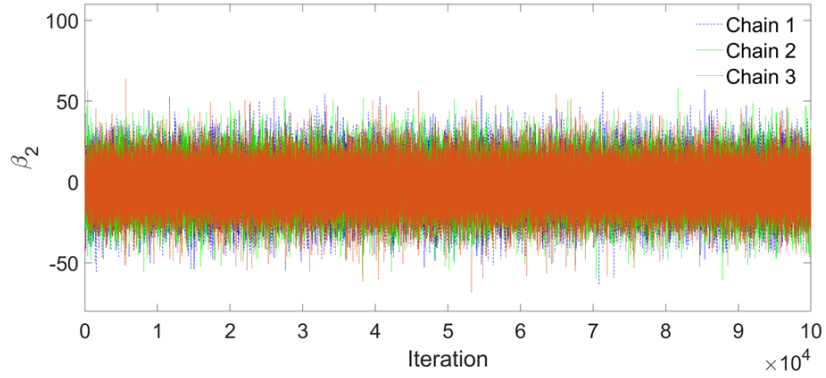


Figure 4.16: Trace plots of MCMC for Bayesian parameter β_2.

Figures 4.17–4.19 show the density functions for different chains of MCMC. The probability density functions of the parameters β_0 and β_1 exhibit left and right skewness, respectively, whereas the density plot of β_2 is symmetric.

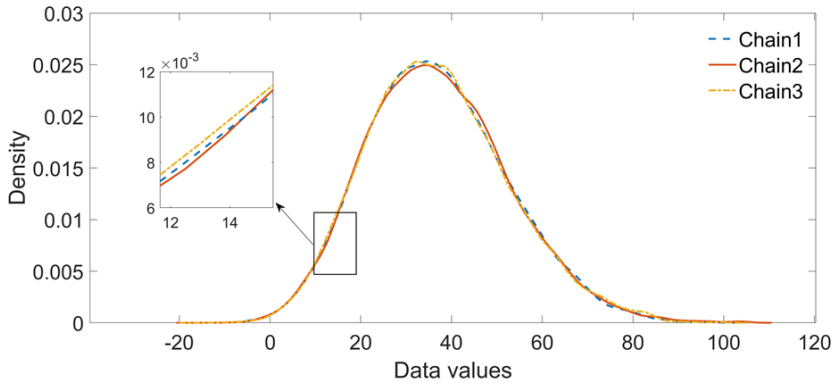


Figure 4.17: Density functions for parameter β_0.

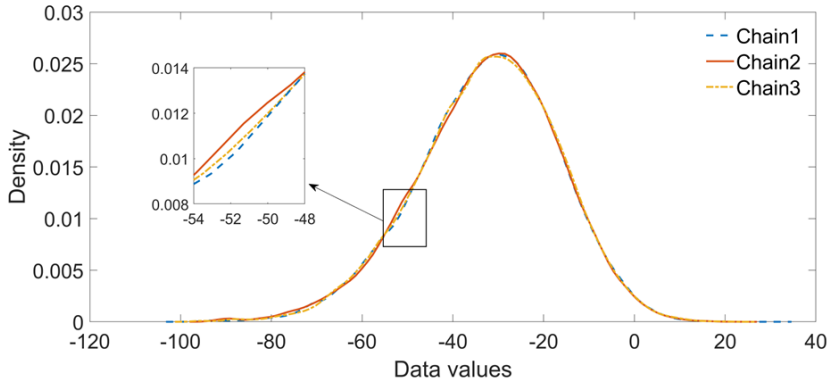


Figure 4.18: Density functions for parameter β_1.

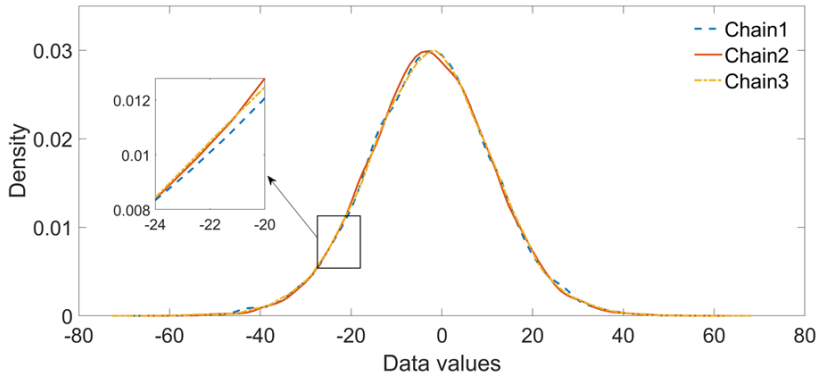


Figure 4.19: Density functions for parameter β_2.

Autocorrelation plots are often used to diagnose the presence of correlation in time series data. They are also used to assess the convergence of MCMC algorithms commonly used in Bayesian inference. For a time series x_n, the auto-correlation function (ACF) is defined as [74]

$$G(\tau) \approx \frac{\sum_{n=1}^{N-\tau}(x_n - \bar{x})(x_{n+\tau} - \bar{x})}{\sum_{n=1}^{N}(x_n - \bar{x})^2} \tag{4.10}$$

where $G(\cdot)$ and τ are the ACF and the time delay (lag), respectively. Figures 4.20– 4.22 show the ACF plots for three chains of MCMC. The rapid decay observed in the ACF plots suggests that successive samples are becoming less correlated, indicating that the chains have reached a stationary state.

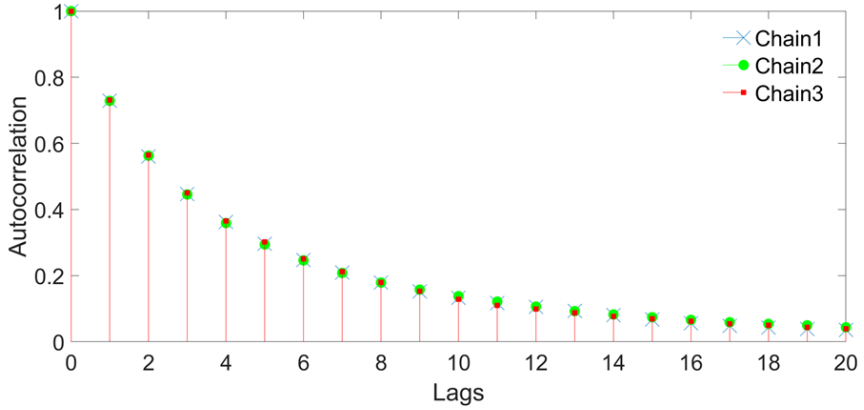


Figure 4.20: Auto-correlation function (ACF) plots for parameter β_0.

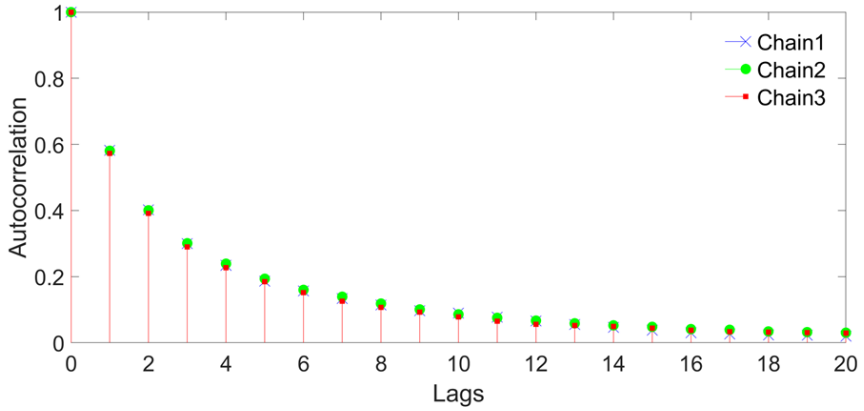


Figure 4.21: ACF plots for parameter β_1.

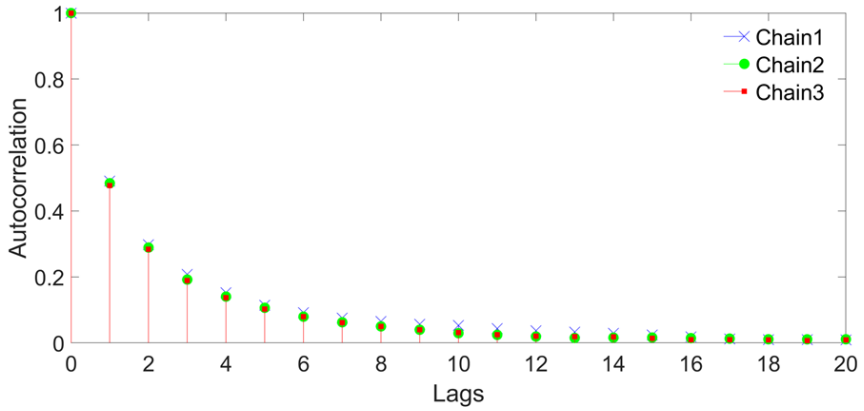


Figure 4.22: ACF plots for parameter β_2.

The confusion matrix can be used to illustrate classifier performance based on four values [75]: true positive (TP) or correct prediction of the fault, false positive (FP) or wrong prediction of the fault, true negative (TN) or true prediction of healthy signals, and false negative (FN) or wrong prediction of healthy signals. Figure 4.23 presents a confusion matrix of the classification results, plotted based on the mean values of the parameters listed in Table 4.2. This visualization helps illustrate the accuracy and misclassification rates between the two categories.

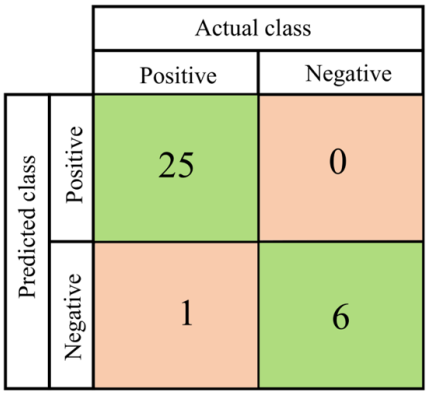


Figure 4.23: Confusion matrix for one of the three-fold cross-validation.

A three-fold cross-validation is performed on the healthy and small-leakage datasets, which are randomly divided into three folds. In each iteration, the classifier is trained on two folds, while the remaining fold is combined with the 24 medium- and high-leakage datasets to form the evaluation set. The main advantage of three-fold cross-validation is that it allows for a more robust evaluation of the model's performance by using multiple train-test splits and averaging the results. This helps to reduce the impact of any particular random sample of the data and to provide a more accurate estimate of the model's performance on unseen data.

Several key evaluation metrics, such as accuracy, precision, recall, and F1-score, were calculated to evaluate the performance of the proposed Bayesian classification. Accuracy measures the proportion of accurate predictions out of all predictions. The three-fold cross-validation results for the Bayesian classification yielded accuracies of $[0.969, 0.875, 0.969]$ for each fold. Precision evaluates the accuracy of positive predictions. The Bayesian classification model achieved precisions of $[1, 0.926, 0.966]$ across the three folds in the cross-validation process. Recall metric measures the classifier's effectiveness in correctly identifying all positive cases within the dataset. The result for three-fold cross-validation yielded recalls of $[0.966, 0.926, 1]$ for each fold. The F1-score, which considers both precision and recall, is commonly defined as the harmonic mean of the two values. The result for three-fold cross-validation is F1-score = 0.9638.

The average values of the three-fold cross-validation for the four performance metrics are presented in Table 4.3. These results are compared against the performance of a support vector machine (SVM), using both a linear kernel and a radial basis function (RBF) kernel. The SVM models are implemented in Python using the `sklearn` library [76]. The results are averaged across the three-fold cross-validation. In both cases, the SVM is trained using a regularization parameter $C = 1$. The linear kernel allows for a straightforward decision boundary, while the RBF kernel offers a more flexible, nonlinear decision surface. However, as demonstrated in Table 4.3, the proposed Bayesian classification method outperforms the SVM model across accuracy, precision, and F1-score metrics, regardless of the kernel type employed.

Table 4.3: Performance metrics of Bayesian classifier with 200,000 MCMC iterations and three-fold cross-validation compared with the support vector machine (SVM) method.

Metric	Presented Method	SVM (RBF)	SVM (Linear)
Accuracy	93.75%	90.7%	92.7%
Precision	96.38%	93.4%	95.3%
Recall	96.38%	96.5%	96.4%
F1-score	96.38%	94.7%	95.8%

Figure 4.24 compares the decision boundary of the Bayesian classifier and SVM with the linear kernel for one of the folds in the three-fold cross-validation. The decision boundary in Bayesian classification is obtained based on the reported parameters in Table 4.2. As is seen, one faulty dataset is misclassified as healthy.

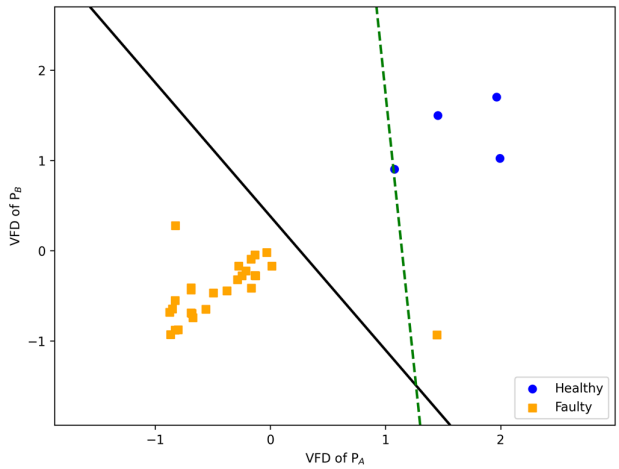


Figure 4.24: Bayesian (dashed line) and support vector machine (SVM) (solid line) classifiers on testing datasets.

The main contributions of this section are summarized as follows:

In the proposed fault detection method, an uninformative prior is used for parameters, which allows the method to effectively learn from the available data and make an accurate classification for unseen data. The uninformative prior avoids imposing any strong assumptions on the data, allowing the method to better capture the underlying distribution and generalize to new data. However, it is important to note that the effectiveness of the uninformative prior for parameter estimation depends on the size and quality of the available data. In situations where the available data are limited or noisy, the use of an informative prior may be necessary to improve the accuracy and robustness of the classification method. Therefore, the choice of prior should be carefully considered in the context of the specific application. One of the helpful approaches is using hierarchical Bayesian inference for more accurate estimates of model parameters and improving the performance of classification.

The use of VFD as a feature for Bayesian classification stands out as a practical feature selection approach for time-series data analysis in EHAs. The VFD proved to be particularly effective in capturing the power-law relationship of the pressure signals. This is evident from the scatter plot and the VFD plots of pressures for different leakage levels, which consistently separate the healthy and faulty conditions. The results show a notable level of accuracy, with an average accuracy rate of 93.75% for fault detection. Moreover, the precision, recall, and F1-score metrics were also assessed, with an average score of 96.38%. Compared with the SVM configurations evaluated in this study, Bayesian classification achieved better performance across the reported metrics.

4.2 Low-Data Fault Detection Using SINDy

Fault detection in EHA systems under limited data is a challenging task. In this section, a fault detection method is proposed based on a data-driven dynamic model of the EHA. Traditional modelling approaches for hydraulic systems often rely on (i) dynamic characteristic analysis in time and frequency domains [77] or (ii) using physics-based principles. For example, in [78] the authors derived both the continuous rolling motion and the discrete actuation dynamics of the electro-hydraulic system using a combination of kinematic and kinetic laws, hybrid dynamics, and nonlinear force modelling of the system.

Given newly introduced machine learning techniques and powerful computing tools, the way complex systems are modelled and understood is changing rapidly [79]. These innovations are transforming traditional modelling and allow building more accurate and efficient models. As a result, data-driven approaches are increasingly replacing classical methods based on physical principles [80].

Machine learning methods, like neural networks, have been used in various fields. However, they come with certain limitations. These models often require large datasets to produce reliable results. This can be considered a limitation in situations where only a limited amount of data is available [81]. This is the case in many practical engineering applications and may lead to reduced performance on unseen data. Additionally, these methods do not easily consider the physical laws to provide the models. Thus, they make them less interpretable and potentially less reliable when compared to methods that are based on physical principles [82].

Despite these challenges in modern machine learning methods, there is an emerging method called sparse identification of nonlinear dynamical systems (SINDy) [83]. The goal of this method is to derive dynamic equations that are simple and interpretable, even when only a small amount of data is available to derive models. This method can accurately capture the dynamics of a system without requiring massive datasets. This capacity makes SINDy more reliable in engineering, especially in the field of control and condition monitoring [84].

The SINDy method has been used as a powerful tool in various fields of engineering in recent years. For instance, in [85], the authors employed SINDy to explore the underlying dynamic equations in fluid mechanics. They demonstrated SINDy's potential to accurately model complex fluid flows with minimal prior knowledge. Moreover, SINDy has been utilized in digital twin identification. In [86], the neural network and SINDy algorithm were combined to improve the accuracy of digital twin modelling for industrial systems.

The SINDy algorithm has not been used for the dynamic modelling of hydraulic systems. This study introduces, for the first time, the use of SINDy for capturing the underlying dynamics of an EHA system. Due to the complexity of EHA systems, selecting the right model parameters is crucial

for SINDy to accurately capture the underlying dynamics of such systems. For instance, SINDy requires the configuration of a feature library, which may include polynomials, the degree of these polynomials, as well as a threshold value that controls the sparsity of the identified model [87]. A common approach for tuning these parameters is based on trial and error. This approach can be time-consuming and may not yield the most optimal results. To address this challenge, this section proposes a Bayesian optimization approach to efficiently tune the parameters.

Bayesian optimization is considered as a powerful tool for automating the tuning of hyper-parameters in machine learning models [34, 89]. Snoek *et al.* [90] showed that this approach can significantly improve model performance. Nguyen [91] summarized recent advancements in Bayesian optimization and emphasized its application in hyperparameter tuning and global optimization of complex black-box functions. Lahmiri [92] demonstrated that Bayesian optimization can enhance the efficiency of neural networks and machine learning algorithms, such as k-nearest neighbors, for diagnosing Alzheimer's disease using MRI data. In [93], a framework using Bayesian optimization to fine-tune the hyperparameters of neural networks was proposed for building energy prediction. They emphasized that Bayesian optimization can improve the accuracy and reliability of forecasting building energy performance.

By utilizing the Bayesian optimization in the SINDy method, the parameter space can be explored more effectively to achieve improved model performance and a more accurate representation of the system dynamics.

This section aims to apply the SINDy to derive the dynamic equations of an EHA system based on data obtained from the actual system's performance. To do this, input signals to the servomotor, as well as actuator displacement and velocity measurements, are utilized to identify the underlying dynamics of the system. Bayesian optimization is used for the first time to tune SINDy's parameters. The optimally tuned parameters are critical for having an accurate model identification in applications such as control and fault detection. The obtained SINDy dynamic models are then used for hydraulic actuator leakage fault detection.

4.2.1 SINDy Algorithm

The SINDy algorithm is employed to identify dynamic equations of the EHA based on the input voltage to the servomotor, u, the displacement of the piston rod, x_p, and the velocity of the piston, v_p. Hydraulic and electrical dynamics, including chamber pressures and motor dynamics, are not represented as separate states. The model therefore assumes that their effects can be approximated through the selected states and control input over the operating conditions considered. Let $\mathcal{X} : [0, T_f] \rightarrow \mathbb{R}^2$ denote the state trajectory of the system, where $\mathcal{X}(t) = [x_p(t) \quad v_p(t)]^T$, and let $u : [0, T_f] \rightarrow \mathbb{R}$ denote the control input. The dynamics are assumed to be governed by a

time-invariant nonlinear system of the form

$$\dot{\mathcal{X}}(t) = \mathcal{F}\big(\mathcal{X}(t), u(t)\big), \quad \mathcal{X}(0) = \mathcal{X}_0 \tag{4.11}$$

where $\mathcal{F} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$ is an unknown nonlinear function that will be identified using the SINDy algorithm.

The SINDy algorithm identifies a sparse nonlinear dynamical system from extracted data, based on the assumption that many systems have relatively few active terms in their dynamics [84]. Thus, $\mathcal{F}$ is assumed to admit a sparse representation within a pre-specified functional space $\mathcal{V} \subset C^\infty(\mathbb{R}^2 \times \mathbb{R}, \mathbb{R}^2)$, constructed from a library of predefined candidate nonlinear functions $\{\psi_i\}_{i=1}^p$. A suitable library of candidate terms is crucial in the SINDy algorithm. One strategy is to start with a basic choice, such as polynomials, and increase the complexity of the library by including other terms, such as trigonometric functions. It is also possible to incorporate partial knowledge of physics to decide on a library.

Let $\mathbf{X} \in \mathbb{R}^{m\times 2}$, $\mathbf{U} \in \mathbb{R}^m$, and $\dot{\mathbf{X}} \in \mathbb{R}^{m\times 2}$ denote the sampled state, input, and time derivative matrices, respectively, evaluated at $\{t_k\}_{k=1}^m \subset [0, T_f]$ with $m = 50001$, corresponding to $T_f = 50$ s of data sampled at frequency $f_s = 1000$ Hz. The candidate nonlinear functions matrix $\mathbf{\Psi} \in \mathbb{R}^{m\times p}$ is constructed as

$$\Psi_{ij} = \psi_j\big(\mathcal{X}(t_i), u(t_i)\big), \quad i = 1, \ldots, m, \ j = 1, \ldots, p \tag{4.12}$$

where $\mathcal{X}(t_i) \in \mathbb{R}^2$ and $u(t_i) \in \mathbb{R}$ denote the i-th sample of the state and control input, respectively. The dynamics can then be expressed as

$$\dot{\mathbf{X}} = \mathbf{\Psi}\mathbf{\Omega}^T \tag{4.13}$$

where $\mathbf{\Omega} \in \mathbb{R}^{2\times p}$ is the sparse coefficient matrix to be identified. Let $\boldsymbol{w}_k \in \mathbb{R}^p$ denote the k-th row of $\mathbf{\Omega}$, which contains the coefficient vector associated with the k-th state derivative. One approach to estimating each coefficient vector $\boldsymbol{w}_k$ is via L_1-regularized regression [84]

$$\boldsymbol{W}_k = \arg\min_{\boldsymbol{c}\in\mathbb{R}^p} \frac{1}{2}\|\dot{\mathbf{x}}_k - \mathbf{\Psi}\boldsymbol{c}\|_2^2 + \rho_{\text{LASSO}}\|\boldsymbol{c}\|_1, \quad k = 1, 2 \tag{4.14}$$

where $\dot{\mathbf{x}}_k \in \mathbb{R}^m$ denotes the k-th column of $\dot{\mathbf{X}}$, and $\rho_{\text{LASSO}} \in \mathbb{R}^+$ is a sparsity-promoting regularization parameter. The L_1 term encourages sparsity in the coefficient vector $\boldsymbol{w}_k$.

In this work, sequentially thresholded least squares (STLS) is used instead of LASSO [94] to obtain the sparse coefficient matrix and construct the SINDy model. The STLS optimizer includes a sparsity threshold parameter, denoted by λ, which controls the trade-off between model complexity and accuracy by removing terms whose coefficient magnitudes are below this threshold.

When λ is small (close to zero), the regularization effect is weak, causing the model to include many terms, which makes it more complex and prone to overfitting by capturing noise. When λ is large, it promotes sparsity, meaning fewer terms are included, resulting in a simpler model that may exclude important features. In the PySINDy package in Python, which implements the SINDy algorithm, λ is referred to as a *threshold*. It is used during sparse regression to select the terms to be included in the model [87].

Choosing the appropriate values for λ and the maximum degree of polynomials using trial and error can be time-consuming and, in many cases, impractical. A Bayesian optimization method is suggested in the following to tackle this challenge of finding the most suitable parameters for the SINDy algorithm.

4.2.2 Bayesian Optimization for SINDy Parameter Tuning

This section aims to enhance the accuracy of models generated by the SINDy algorithm by tuning its hyperparameters through Bayesian optimization. The focus is on selecting the optimal sparsity threshold and polynomial degree that minimize the loss, defined as the difference between predicted and actual states of the EHA. The loss is computed over time using a numerical approximation. The results are validated on both simulation and experimental data to demonstrate the reliability of the proposed method.

In this study, the optimal values for the sparsity threshold $\lambda^* \in \mathbb{R}^+$ in the STLS optimizer and the polynomial degree $d^* \in \mathbb{N}$ in the SINDy framework are obtained. The optimization objective is defined as

$$(\lambda^*, d^*) = \arg\min_{\lambda,d} \mathcal{J}(\lambda, d) \tag{4.15}$$

where the loss function $\mathcal{J} : \mathbb{R}^+ \times \mathbb{N} \to \mathbb{R}$ is defined as

$$\mathcal{J}(\lambda, d) = \int_0^{T_f} \|\mathcal{X}_s(t) - \mathcal{X}_{ref}(t)\|_2^2 \, dt \tag{4.16}$$

Here, T_f is the sample time, and $\mathcal{X}(t) = [x_p(t) \;\; v_p(t)]^T \in \mathbb{R}^2$ denotes the system state, with $x_p(t)$ and $v_p(t)$ representing the position and velocity of the rod, respectively. The subscript s refers to predictions obtained via the SINDy model during training, and ref refers to the extracted data from the EHA.

Given that the data were extracted at discrete time instances $\{t_k\}_{k=1}^m$, the integral in the above equation is approximated numerically using the composite trapezoidal rule

$$\mathcal{J}_m(\lambda, d) \approx \sum_{k=1}^{m-1} \frac{\Delta t_k}{2} \left(\|\mathbf{e}_k\|_2^2 + \|\mathbf{e}_{k+1}\|_2^2 \right) \tag{4.17}$$

where $\Delta t_k = t_{k+1} - t_k$, m is the number of samples, and the error vectors $\mathbf{e}_k \in \mathbb{R}^2$ are defined as

$$\mathbf{e}_k = \mathcal{X}_s(t_k) - \mathcal{X}_{ref}(t_k) = \begin{bmatrix} x_s(t_k) - x_{ref}(t_k) \\ v_s(t_k) - v_{ref}(t_k) \end{bmatrix} \tag{4.18}$$

The optimal values of λ and d are obtained to minimize this error. The Bayesian optimization process is set up to explore thresholds in the interval $[0.01, 0.1]$ and polynomial degrees limited to integers 2 or 3. The optimization begins with two random initial points and proceeds for 10 iterations following the initial exploration phase.

The `gp_minimize` function from the `skopt` library in Python, which is a Bayesian optimization library, is utilized to optimize the hyperparameters [88]. The best hyperparameters obtained from the optimization are then used in the SINDy algorithm to provide an accurate representation of the EHA system. In the next subsections, first, the SINDy model is obtained using simulation data and then evaluated on unseen datasets. Furthermore, experimental data is used to demonstrate the capability of the SINDy algorithm. In both cases, the results confirm the model's effectiveness.

4.2.3 Demonstration Results

The SINDy is trained using the simulation model of the EHA in MATLAB. The accuracy of the simulation model is first verified through comparison with the experimental data. As shown in Fig. 4.25, the results show a good level of accuracy. Three datasets are simulated under a load with no leakage. One simulated dataset, which includes control input signal u, actuator displacement x_p, and actuator velocity v_p, is used to train the SINDy algorithm.

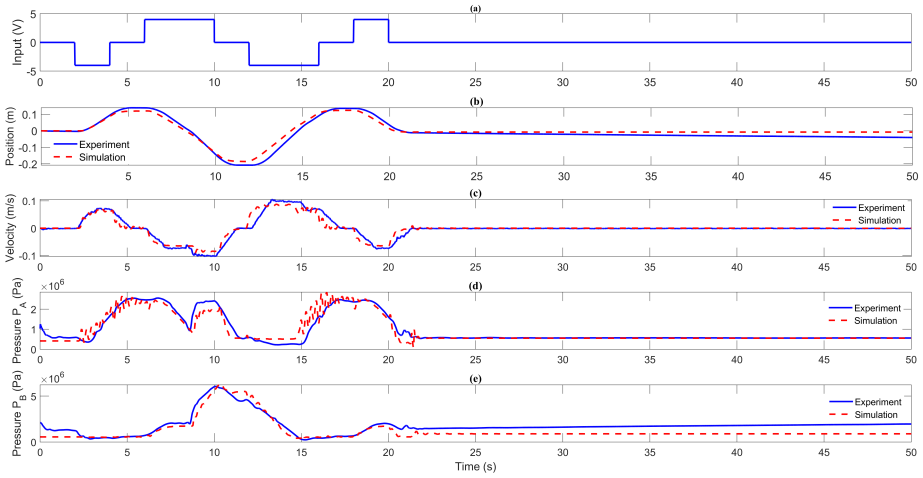


Figure 4.25: Comparison between simulation and experimental results.

The optimal threshold obtained using Bayesian optimization is $\lambda^* = 0.013024$, and the best polynomial degree is $d^* = 2$. The corresponding minimum $\mathcal{J}(\lambda^*, d^*)$ is 0.001420. The Bayesian-optimized SINDy dynamics are obtained as follows

$$\dot{X}(t) = \begin{bmatrix} \dot{x}_p \\ \dot{v}_p \end{bmatrix} = \begin{bmatrix} v_p \\ \phi(x_p, v_p, u) \end{bmatrix} \tag{4.19}$$

where $\phi(x_p, v_p, u)$ is given as

$$\phi(x_p, v_p, u) = -2.032\, v_p - 0.156\, u + 0.009\, x_p^2 + 2.594\, x_p v_p + 0.428\, x_p u + 1.280\, v_p^2 - 1.220\, v_p u - 0.102\, u^2 \tag{4.20}$$

Figure 4.26 shows typical control input signal u, actuator displacement x_p, and actuator velocity v_p for training the SINDy algorithm using the simulation data. To show the performance of the trained model, it is tested on two unseen datasets. Figures 4.27 and 4.28 present the results for two unseen control input signals, which were fed into the derived dynamic model. The results show the accuracy and capability of the SINDy algorithm in capturing the actuator displacement and velocity during training and testing.

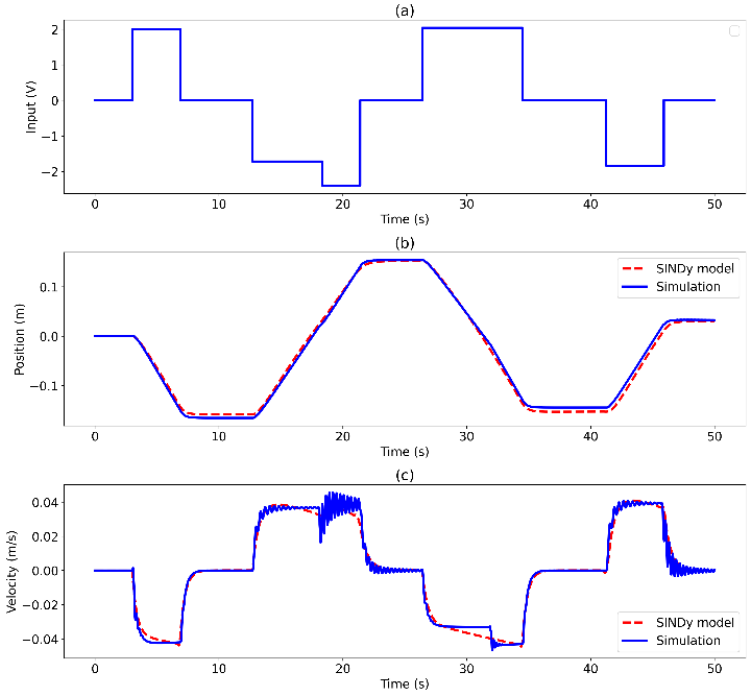


Figure 4.26: Comparison of the results for training the sparse identification of nonlinear dynamics (SINDy) algorithm using data obtained from the simulation model: (a) control input signal, u; (b) actuator displacement, x_p; (c) actuator velocity, v_p.

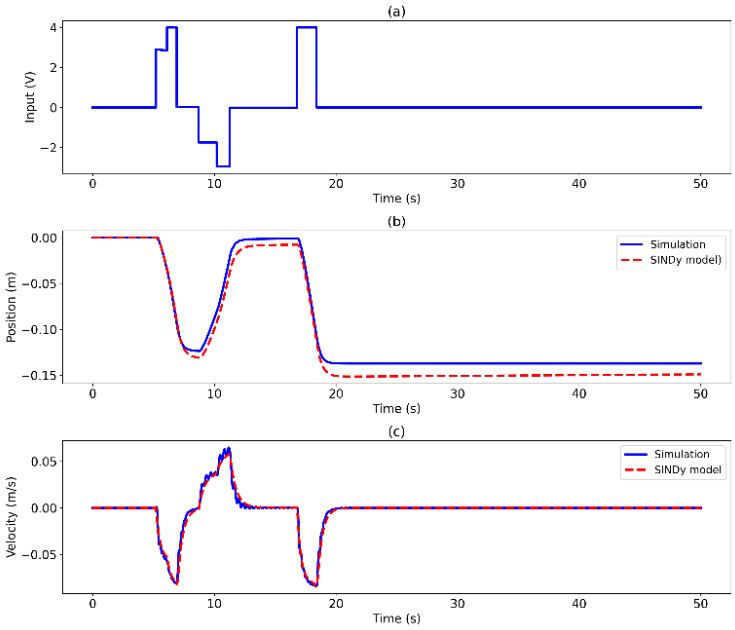


Figure 4.27: Comparison of the results for testing the unseen data using simulation data (trial 1): (a) control input signal, u; (b) actuator displacement, x_p; (c) actuator velocity, v_p.

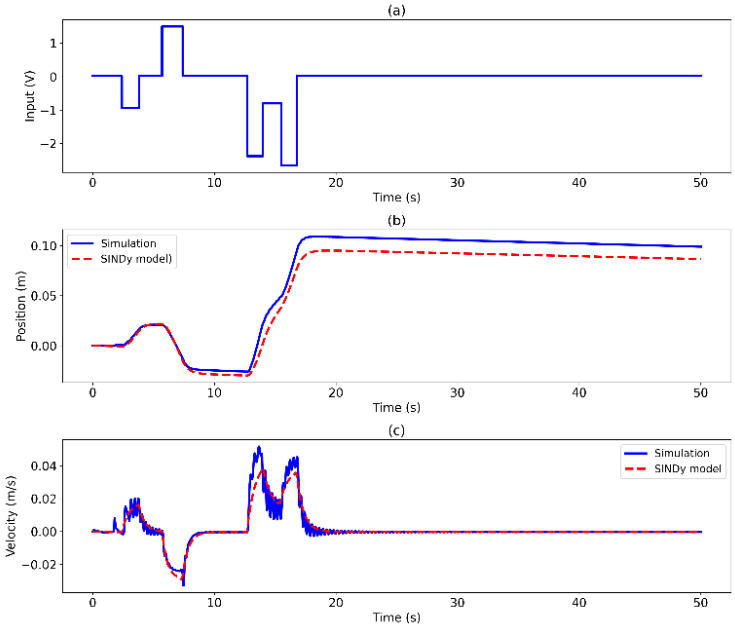


Figure 4.28: Comparison of the results for testing the unseen data using simulation data (trial 2): (a) control input signal, u; (b) actuator displacement, x_p; (c) actuator velocity, v_p.

4.2.4 Experimental Validation

To show the capability of the SINDy algorithm, the dynamic equations of the EHA are derived based on experimental datasets. Three datasets were extracted from the EHA system under a load of 367 kg

with no leakage. One of the datasets was used for training, while the remaining two datasets were used to evaluate the performance of the algorithm. Using Bayesian optimization, the most effective hyperparameters are successfully identified for the SINDy algorithm. The optimal threshold is $\lambda^* = 0.01679$, and the best polynomial degree is $d^* = 2$. The corresponding minimum $\mathcal{J}(\lambda^*, d^*)$ was 0.011732. The Bayesian-optimized SINDy dynamic equations are obtained as follows

$$\dot{\mathcal{X}}(t) = \begin{bmatrix} \dot{x}_p \\ \dot{v}_p \end{bmatrix} = \begin{bmatrix} \phi_1(x_p, v_p, u) \\ \phi_2(x_p, v_p, u) \end{bmatrix} \tag{4.21}$$

where $\phi_1(x_p, v_p, u)$ is given as

$$\phi_1(x_p, v_p, u) = 0.646\, v_p - 0.031\, u + 0.044\, x_p u + 1.067\, v_p^2 + 0.104\, v_p u \tag{4.22}$$

and

$$\begin{aligned} \phi_2(x_p, v_p, u) = {} & -0.044\, x_p - 2.679\, v_p - 0.237\, u - 0.091\, x_p^2 + 2.852\, x_p v_p + 0.569\, x_p u \\ & - 20.301\, v_p^2 - 2.819\, v_p u - 0.098\, u^2 \end{aligned} \tag{4.23}$$

Figure 4.29 shows the control input signal u, actuator displacement x_p, and actuator velocity v_p from the experimental dataset used to train the SINDy algorithm. To evaluate the model's performance, two different datasets are considered. Figures 4.30 and 4.31 illustrate the results for two unseen experimental input datasets, which were fed into the derived dynamic model approximated by (4.21)–(4.23). The results demonstrate the effectiveness of the SINDy algorithm for estimating the position and velocity of the rod with high precision for experimental data.

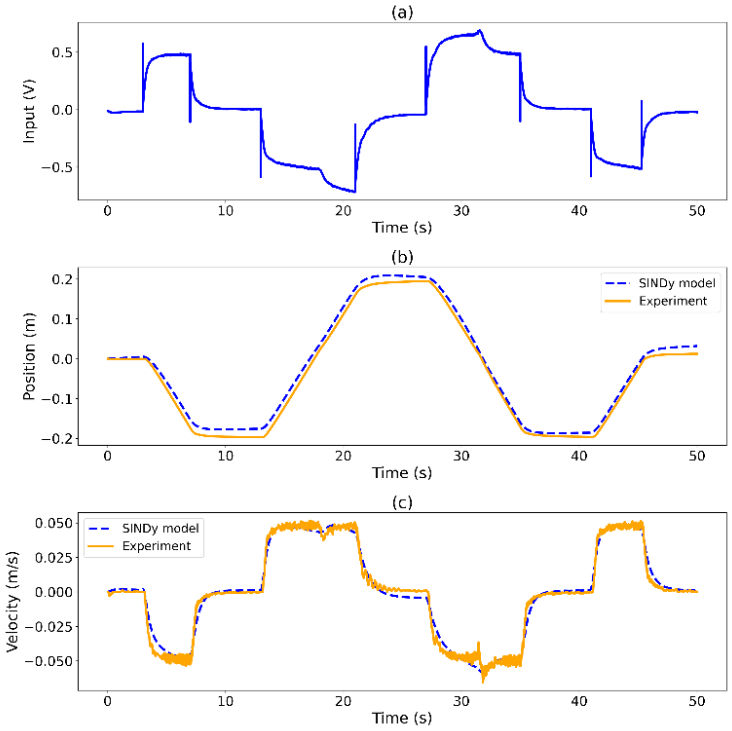


Figure 4.29: Comparison of the results for training the SINDy algorithm using experimental data: (a) control input signal, u; (b) actuator displacement, x_p; (c) actuator velocity, v_p.

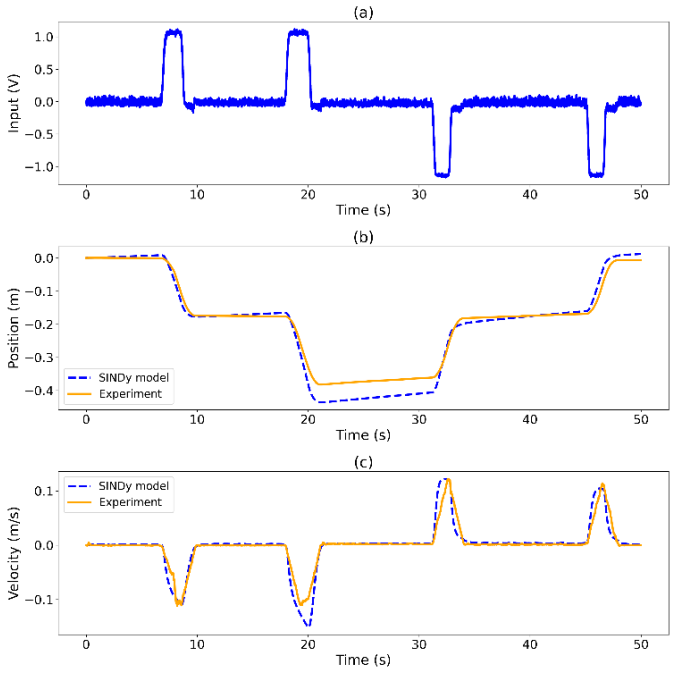


Figure 4.30: Comparison of the results for testing the unseen data using experimental data (trial 1): (a) control input signal, u; (b) actuator displacement, x_p; (c) actuator velocity, v_p.

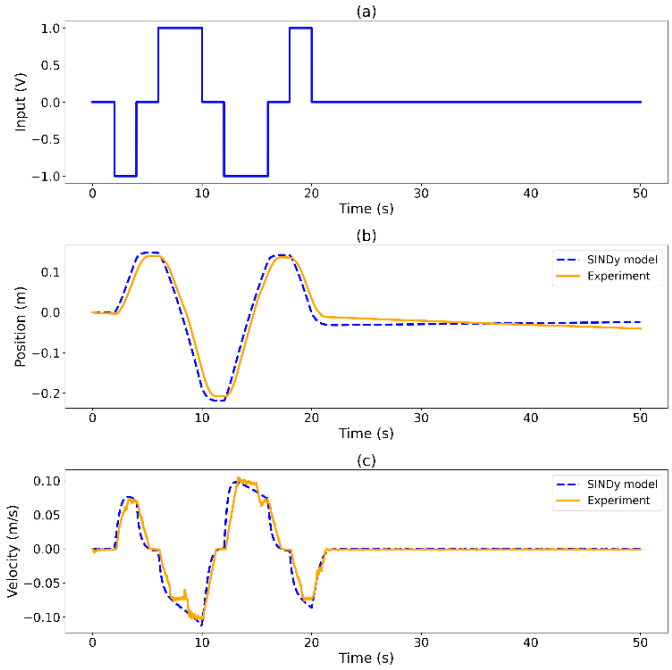


Figure 4.31: Comparison of the results for testing the unseen data using experimental data (trial 2): (a) control input signal, u; (b) actuator displacement, x_p; (c) actuator velocity, v_p.

4.2.5 Application to Fault Detection

In this subsection, a fault detection methodology is implemented to classify actuator internal leakage fault in the EHA based on the SINDy algorithm. A collection of 48 datasets was used, including both healthy and faulty datasets, with the faulty datasets categorized into three distinct levels of the actuator's internal leakage. The leakage level varied from 0 L/m to a maximum of 8 L/m, depending on operating conditions. Experiments were also conducted under three different load levels: no load, medium load, and high load, with the load ranging from 0 to 367 kg. The study involved the utilization of both open-loop and closed-loop controls using step and joystick inputs. Figure 4.32 shows the results of the EHA system for a typical dataset with a high leakage level.

Figure 4.33 illustrates the different leakage levels, including no leakage, small leakage, medium leakage, and high leakage, using the same input signal.

The proposed method considers three distinct dynamic models obtained by the SINDy algorithm and optimized through Bayesian optimization to classify between healthy and faulty conditions. The datasets used in this study include: (i) a no-leakage, no-load dataset for developing the healthy model, and (ii) a small-leakage, no-load dataset, as well as a medium-leakage, high-load dataset to develop faulty models.

Table 4.4 shows the SINDy models obtained for these three datasets, as well as the optimal parameters obtained by Bayesian optimization.

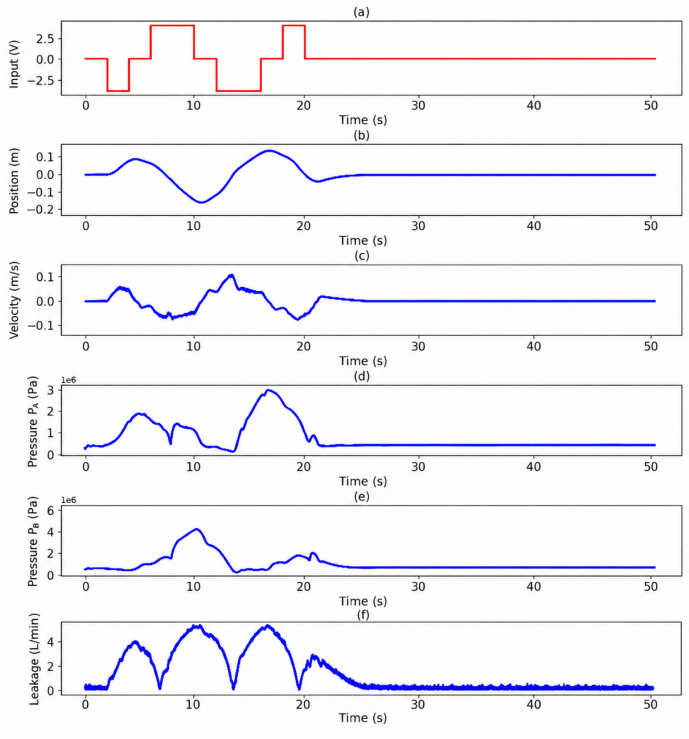


Figure 4.32: Experimental results for a typical dataset with a high leakage level.

Table 4.4: Sparse identification of nonlinear dynamics (SINDy) models based on actuator displacement x_p, actuator velocity v_p, control input signal u, and corresponding optimal Bayesian parameters.

Model	**SINDy model**	**Optimal Bayesian parameters**
Healthy system (No leakage)	$\dot{x}_p = 0.683\,v_p - 0.031\,u + 0.163\,x_p v_p + 0.044\,x_p u + 1.312\,v_p^2 + 0.108\,v_p u - 0.004\,u^2$ $\dot{v}_p = -0.021\,x_p - 2.978\,v_p + 0.299\,u - 0.129\,x_p^2 - 4.869\,x_p v_p + 0.729\,x_p u - 0.197\,v_p u - 0.041\,u^2$	$d^* = 2$, $\lambda^* = 0.010713$, $\mathcal{J}(\lambda^*, d^*) = 0.008447$
Faulty system (Small leakage, $Q_l < 3$ L/min)	$\dot{x}_p = 0.681\,v_p - 0.028\,u + 0.242\,x_p v_p + 0.054\,x_p u + 0.126\,v_p^2$ $\dot{v}_p = -0.074\,x_p - 2.864\,v_p + 0.263\,u + 0.108\,x_p^2 + 6.381\,x_p v_p + 0.857\,x_p u - 50.070\,v_p^2 - 8.618\,v_p u - 0.365\,u^2$	$d^* = 2$, $\lambda^* = 0.010157$, $\mathcal{J}(\lambda^*, d^*) = 0.001476$
Faulty system (Medium leakage, $Q_l < 6$ L/min)	$\dot{x}_p = -0.042\,x_p + 0.762\,v_p - 0.019\,u + 0.029\,x_p v_p + 0.019\,x_p u + 1.251\,x_p^2 + 0.077\,v_p u$ $\dot{v}_p = -0.508\,x_p - 2.696\,v_p + 0.217\,u + 0.232\,x_p^2 + 0.153\,x_p v_p + 0.324\,x_p u + 5.160\,v_p^2 + 0.184\,v_p u$	$d^* = 2$, $\lambda^* = 0.010282$, $\mathcal{J}(\lambda^*, d^*) = 0.001179$

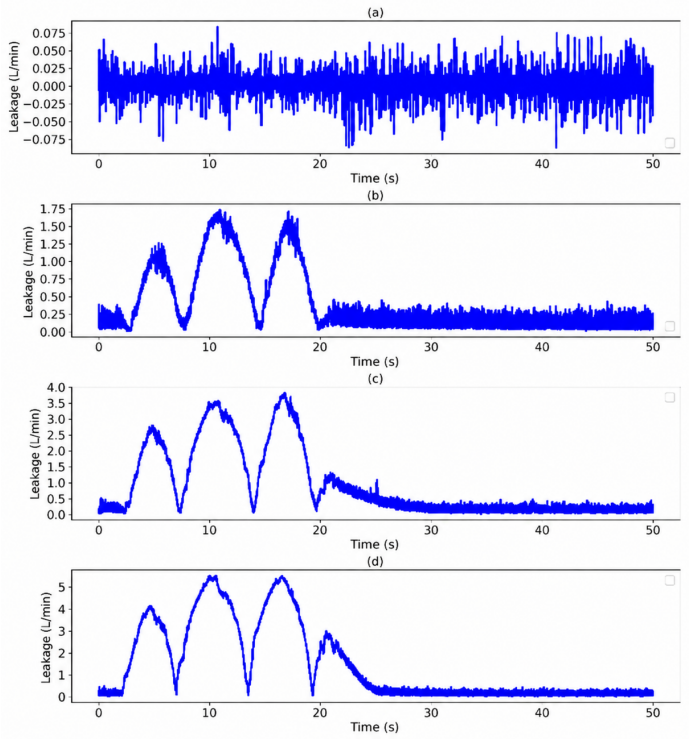


Figure 4.33: Different leakage levels: (a) no leakage; (b) small leakage; (c) medium leakage; (d) high leakage.

The condition of unseen datasets (healthy or faulty) is classified using the three models described in Table 4.4. The "no leakage" model identifies a healthy condition, while the small and medium leakage models detect a faulty condition. The small and medium leakage models are considered to improve the method's accuracy in fault identification compared to relying on a single fault model.

To determine whether an unseen dataset is healthy or faulty, its input data is provided to three models, which generate corresponding actuator displacement, x_p. The actual actuator displacement of the unseen dataset is then compared with the actuator displacements obtained from the three models. If the actuator displacement of the dataset is closer to that predicted by the healthy model, the dataset is classified as healthy. Conversely, if it is closer to the actuator displacements of the faulty models, the dataset is classified as faulty. The proximity criterion is defined based on the integral of the absolute error (IAE) between the actual actuator displacement of the new dataset and the actuator displacements predicted by the three models, and is defined as

$$\mathrm{IAE}_i = \int_0^{T_f} \left| x_{\mathrm{model}_i} - x_{\mathrm{new}} \right| dt, \quad i \in \{H, F_s, F_m\} \tag{4.24}$$

where x_{model_i} shows each of the three SINDy models considered for fault classification. H represents healthy, F_s and F_m correspond to the small and medium faulty models, respectively. $T_f = 50$ s is

defined as the sample time, and x_{new} is considered as the dataset being evaluated. The fault detection algorithm can be summarized as follows:

1. **Simulate actuator displacement:**
   Provide the input from the unseen dataset to the three SINDy models to predict the actuator displacement outputs.
2. **Calculate error:**
   Compute the integrated absolute error (IAE)

$$\text{IAE}_i = \int_0^{T_f} |x_{\text{model}_i} - x_{\text{new}}| \, dt, \quad i \in \{H, F_s, F_m\} \tag{4.25}$$

3. **Identify minimum error:**
   Find the model with the lowest error

$$i^* = \arg \min_{i \in \{H, F_s, F_m\}} \text{IAE}_i \tag{4.26}$$

4. **Classify dataset:**
   - If $i^* = H$, classify the dataset as healthy.
   - If $i^* = F_s$ or $i^* = F_m$, classify the dataset as faulty.
5. **Output:**
   Dataset classification (healthy or faulty).

Forty-five unseen datasets were used to evaluate the performance of the classification method. The diversity of the experimental test datasets is summarized in Table 4.5.

Table 4.5: Summary of test datasets used for model validation.

Factor	Category	Range	Number of datasets
Control mode	Open loop	-5 – +5 V	24
	Closed loop	-4.5 – +6 V	21
External load	No	0 kg	14
	Medium	183.5 kg	16
	High	367 kg	15
Input type	Step input	-5 – +4 V	21
	Joystick	-5 – +6 V	24
Leakage level	No	$Q_l = 0$ L/min	11
	Small	$Q_l < 3$ L/min	11
	Medium	$Q_l < 6$ L/min	11
	High	$Q_l < 8$ L/min	12

As shown in the Table 4.5, the validation data cover a broad range of operating conditions, including variations in load, control mode, and leakage level. Each of the 45 test datasets corresponds to one unique combination of control mode, load condition, input mode, and leakage level. The classification process achieved an accuracy rate of 95.56% across these datasets, which include varying loads and leakage levels with both open-loop and closed-loop controls using step and joystick inputs. Figure 4.34 shows the confusion matrix for the 45 unseen datasets, including 33 faulty and 12 healthy datasets.



Figure 4.34: Confusion matrix for all 45 unseen experimental datasets using SINDy models with Bayesian-optimized parameters. Positive indicates faulty, and negative indicates healthy.

Table 4.6 demonstrates the impact of applying Bayesian optimization across several key metrics, including accuracy, precision, recall, and F1-score. The accuracy of the model improved from 91.11% without Bayesian optimization to 95.56% with it. Precision decreased slightly from 96.88% to 94.44%, suggesting a slight increase in false positives with Bayesian optimization. However, recall improved significantly, increasing from 91.18% without optimization to 100% with Bayesian optimization, which indicates the optimized model effectively captured all true positive cases, reducing false negatives. This improvement in recall contributed to an increase in the F1-score from 93.94% to 97.10%, reflecting a better balance between precision and recall in the optimized model. Overall, the results highlight the effectiveness of Bayesian optimization in enhancing the obtained model's ability to accurately classify data.

Table 4.6: Comparison of the performance metrics for the fault detection using SINDy models with and without using Bayesian optimization.

Metric	With Bayesian optimization	Without Bayesian optimization
Accuracy	95.56%	91.11%
Precision	94.44%	96.88%
Recall	100%	91.18%
F1-score	97.10%	93.94%

Figures 4.35-4.38 illustrate the fault detection results for four unseen datasets: no leakage, small leakage, medium leakage, and high leakage. All datasets were collected under a closed-loop velocity control mode using joystick input signals and were evaluated against the three SINDy models. In Fig. 4.35, the no leakage dataset shows a close match to the no leakage SINDy model, with an integrated absolute error (IAE) of 0.692998, identifying it as healthy. Figure 4.36 reveals that the small leakage dataset aligns closely with the small leakage SINDy model, with an IAE of 1.227809, classifying it as faulty. In Fig. 4.37, the medium leakage dataset most closely matches the medium leakage SINDy model, with an IAE of 3.584143, indicating a faulty condition. Finally, Fig. 4.38 demonstrates that the high leakage dataset also aligns with the medium leakage SINDy model, achieving an IAE of 1.224302, thus recognizing it as faulty. These results confirm the method's ability to distinguish between healthy and faulty conditions across varying leakage levels and under noisy input conditions.

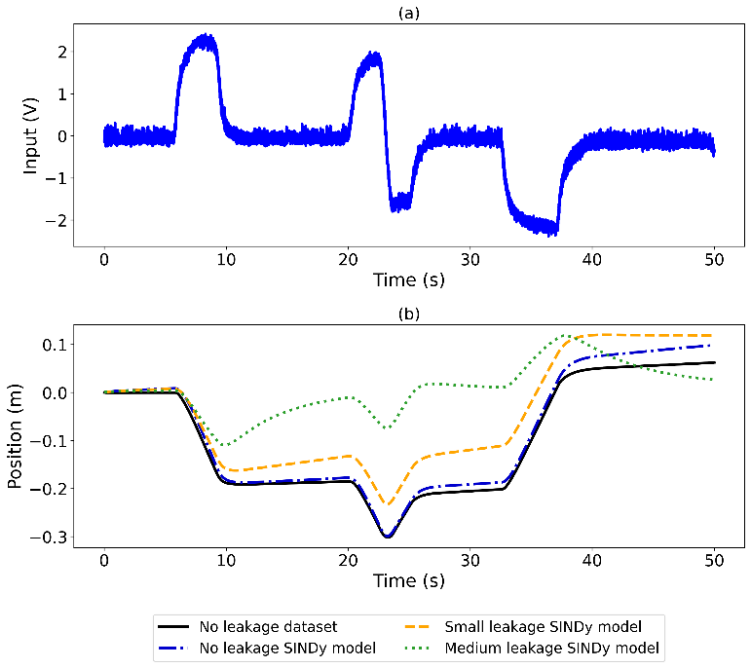


Figure 4.35: (a) Control input signal, u; (b) comparison of the actuator displacement, x_p, for unseen no leakage, medium load dataset, and SINDy models.

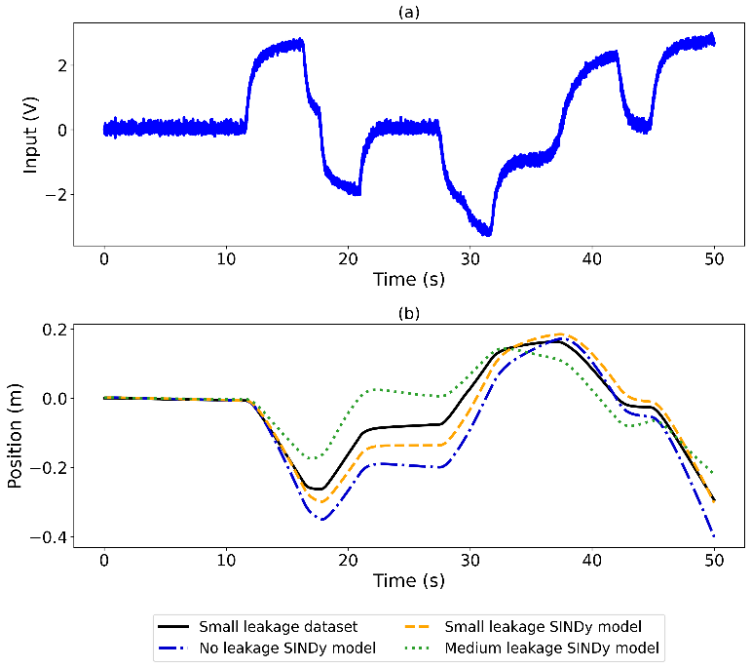


Figure 4.36: (a) Control input signal, u; (b) comparison of the actuator displacement, x_p, for unseen small leakage, medium load dataset and SINDy models.

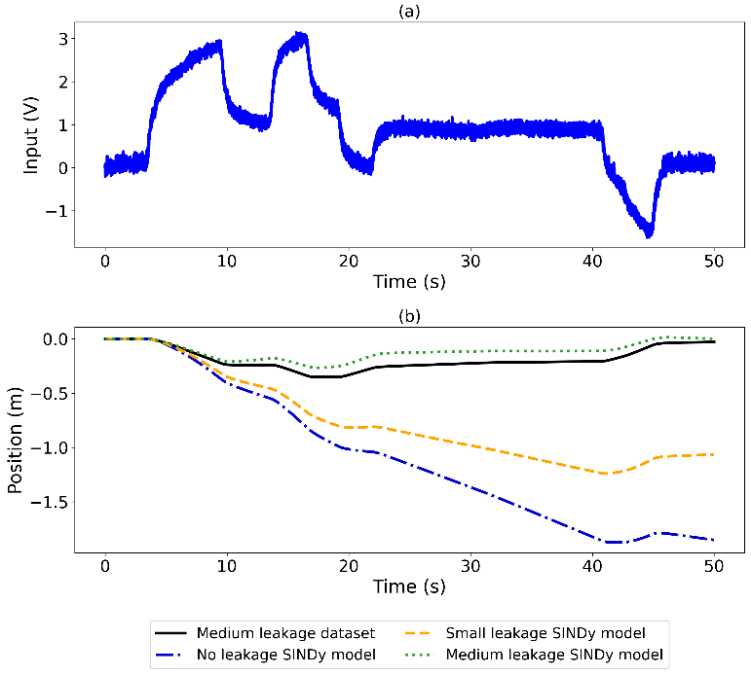


Figure 4.37: (a) Control input signal, u; (b) comparison of the actuator displacement, x_p, for unseen medium leakage, medium load dataset and SINDy models.

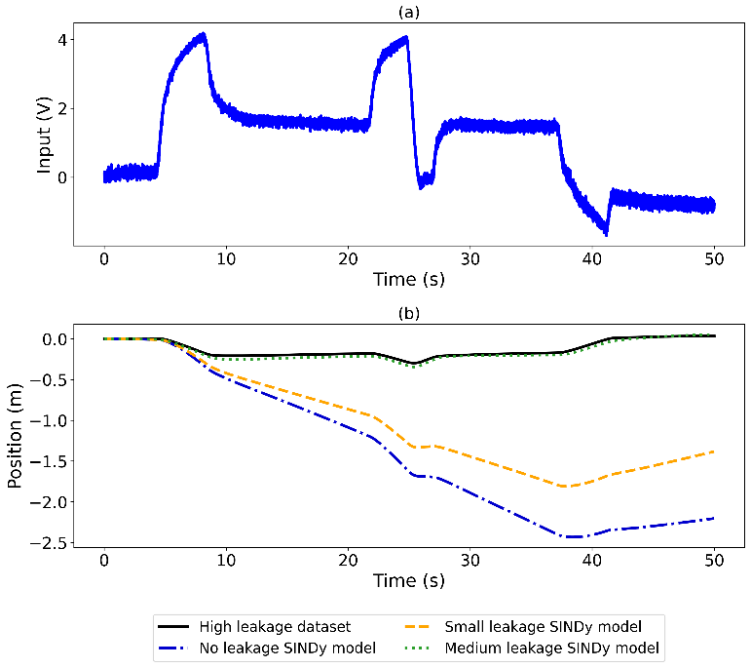


Figure 4.38: (a) Control input signal, u; (b) comparison of the actuator displacement, x_p, for unseen high leakage, medium load dataset and SINDy models.

Table 4.7 reports the IAE results for the four unseen datasets under four different leakage levels. These results demonstrate that all four datasets are correctly classified based on the minimum IAE value.

Table 4.7: Integral absolute error (IAE) values for four unseen experimental datasets across three SINDy models. Minimum values are highlighted in bold.

Test dataset	**IAE (Healthy model)**	**IAE (Small leakage model)**	**IAE (Medium leakage model)**
No leakage	**0.692998**	2.675488	5.491678
Small leakage	2.303017	**1.227809**	2.234562
Medium leakage	44.646680	28.557982	**3.584143**
High leakage	63.315515	45.453870	**1.224302**

The key contributions of the proposed SINDy framework with Bayesian optimization for hyperparameter tuning are highlighted below:

The proposed Bayesian-optimized SINDy model offers significant advantages compared with traditional machine-learning and physics-based approaches. Unlike black-box learning methods, which provide limited insight into system dynamics, the SINDy framework expresses the underlying relationships through the governing equations. In this section, three SINDy models were identified to distinguish healthy from faulty signals. The fault classification relies on computing the integral of absolute error (IAE) between the measured actuator displacement and the model-predicted outputs. It allows the identification of the fault signals through physically meaningful comparisons rather than

only statistical indicators. This characteristic makes the method attractive for industrial applications where explainability and engineering interpretability are essential.

The robustness of the proposed method was evaluated using 45 unseen experimental datasets. They cover a wide range of operational conditions. As summarized in Table 4.5, the validation data consider variations in load (no, medium, and high), control mode (open- and closed-loop), input mode (joystick and step), and leakage level (no, small, medium, and high). Each dataset corresponds to a unique combination of these factors. Then, it provides a comprehensive test environment for fault classification.

It is worth mentioning that including nonlinear terms in the SINDy library does not always improve model accuracy. In some cases, excessive nonlinearities may introduce numerical instability or overfitting. The Bayesian optimization results confirmed that a maximum polynomial degree of two provided the best trade-off between model accuracy and numerical stability. Although the chamber pressures P_A and P_B are not explicitly included as variables, their influence is implicitly represented through the correlations among the piston position x_p, velocity v_p, and control input u. Incorporating additional signals, such as pressure, can enhance model generalizability. However, the proposed framework was designed to remain computationally efficient while relying on a minimal sensor setup. This is an important consideration for embedded or industrial systems, where adding extra sensors increases hardware cost and complexity.

From a computational perspective, the proposed Bayesian-optimized SINDy method is also efficient and suitable for online implementation. The Bayesian optimization and model identification processes are executed offline for experimental data. Depending on the dataset size and initialization, this offline process may require several minutes to a few hours on a standard personal computer (Intel i7, 16 GB RAM). Since these computations are performed offline, they do not affect online performance. During online operation, the algorithm only needs to compute the IAE between the measured actuator displacement and the three pre-trained SINDy models. This involves basic subtraction and numerical integration over sampled data. Because the SINDy models are expressed as compact polynomial equations rather than large neural networks, their evaluation is lightweight and can be executed efficiently on embedded hardware such as a Raspberry Pi or microcontroller.

4.3 Summary

In this chapter, condition monitoring of electro-hydrostatic actuators (EHAs) was investigated through two fault detection methods.

First, a fault detection framework based on Bayesian classification and polyscale complexity measures was developed. By extracting informative features from pressure signals, the proposed method demonstrated high classification accuracy between healthy and faulty conditions, even

under varying operating scenarios. Second, an interpretable data-driven modelling method was introduced using the sparse identification of nonlinear dynamics (SINDy) algorithm. By using Bayesian optimization for hyperparameter tuning, dynamic models of the EHA system were obtained even with limited data. These models were then used to construct a fault detection framework for distinguishing healthy and faulty signals.

These two approaches provide new perspectives on condition monitoring in EHAs, combining feature-based classification and dynamic modelling. While the Bayesian classification method emphasizes direct pattern recognition from signal features, the SINDy-based method derives system dynamics for fault detection. Both methods demonstrate strong performance under limited data conditions.

Chapter 5

Controller Tuning

This chapter investigates two methods for PID controller tuning in the electro-hydrostatic actuator (EHA) system: a model-based framework and a model-free learning method. First, a model-based method is developed in the frequency domain to tune the PID controller gains to meet desired transient-response characteristics. The results of this work have been previously published[1]. Model-based approaches provide a systematic framework for controller design based on the system dynamics. However, when the controller parameters are designed for a nominal operating condition and remain fixed during operation, the controller may not perform well under varying operating conditions and uncertainties, which are common in hydraulic systems. Thus, a model-free method is introduced for online controller tuning under dynamic changes. The proposed model-free framework combines a reinforcement learning method, namely augmented random search (ARS), for offline exploration of the gain space with Gaussian process (GP) surrogate modelling for online tuning. In this way, the controller gains can be updated when the system dynamics change over time due to varying operating conditions or uncertainties. The effectiveness of the proposed framework is demonstrated through simulation studies on the EHA system under both nominal and perturbed operating conditions. A manuscript based on this study is being submitted to the Journal of Dynamic Systems, Measurement and Control[2].

5.1 Offline Model-Based Controller Tuning

Various model-based design techniques have been developed for hydraulic systems, including analytical, frequency-domain, and optimization-based methods. Among these, optimization-based

[1] S. Hosseinpour, W. Kinsner, and N. Sepehri, "A Frequency-Domain Optimization of PID Controllers," in *Proceedings of the IEEE 22nd International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC)*, Stanford University, Palo Alto, CA, USA, Aug. 19–21, 2023.

[2] S. Hosseinpour, and N. Sepehri, "Model-Free Learning for Online PID Controller Tuning in Electro-Hydrostatic Actuators Under Dynamic Changes,", 2026, to be submitted.

approaches determine the controller parameters by optimizing performance objectives such as tracking accuracy, robustness, and transient response characteristics [42, 43].

In this section, a frequency-domain optimization-based tuning method is considered for determining the parameters of a PID controller. The transfer function of the PID controller is defined in the continuous frequency domain as

$$G_c(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s \tag{5.1}$$

where $U(s)$ and $E(s)$ are the control and tracking error signals in the frequency domain, respectively. The transfer function of the PID controller can also be defined as

$$G_c(s) = K_p + \frac{K_i}{s} + K_d \left(\frac{s}{1 + \frac{s}{N}} \right) \tag{5.2}$$

where N is the derivative filter coefficient, which determines the cutoff frequency of the first-order derivative filter. As N approaches infinity, Eq. (5.2) reduces to the ideal PID controller given in Eq. (5.1). The PID parameters can be determined based on the design specifications, which may be defined in terms of gain and phase margins, as well as transient response measures such as overshoot, rise time, and settling time.

5.1.1 Design Specifications

To characterize the performance of PID controller systems, several criteria can be considered, such as transient response and robustness criteria, including phase and gain margins.

(i) **Phase margin ϕ_m and gain crossover frequency ω_g specifications:** Gain and phase margins have always served as important measures of robustness. It is known that the phase margin is related to the damping of the system and therefore can also serve as a performance measure [44]. The equations that define the phase margin and the gain crossover frequency are

$$\left| G_c(j\omega_g) G_p(j\omega_g) \right| = 1 \tag{5.3}$$

$$\arg\left[G_c(j\omega_g) G_p(j\omega_g) \right] = -\pi + \phi_m \tag{5.4}$$

where $G_c(s)$ denotes the transfer function of the PID controller in Eq. (5.2), and $G_p(s)$ represents the transfer function of the plant.

(ii) **Transient response specifications:** Transient response characteristics such as peak time

(t_p), overshoot (OS), and settling time (t_s) can also be used to evaluate the time-domain performance of the system. They serve as key performance indicators in controller design.

The desired closed-loop performance is specified using a reference model. The desired transient response is first defined in terms of performance specifications, such as the maximum overshoot, peak time, rise time, and settling time. Since these specifications are conventionally characterized using the step response, a unit-step input is considered first. For a second-order system, these transient specifications have explicit analytical relationships with the damping ratio and natural frequency. Therefore, the desired transient response can be completely characterized by a standard second-order reference model. Consequently, once the desired transient specifications are selected, the corresponding second-order reference model can be constructed. This model represents the desired closed-loop response that the controller is designed to achieve. The transfer function of the reference model is given by

$$H(s) = \frac{K_{dc}\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{5.5}$$

where K_{dc} is the static gain (assumed to be unity for simplicity), ζ is the damping ratio, and $\omega_n > 0$ is the undamped natural frequency. The corresponding damped natural frequency is [45]

$$\omega_d = \omega_n\sqrt{1-\zeta^2} \tag{5.6}$$

For an underdamped second-order system, the unit-step response is

$$y(t) = 1 - \frac{e^{-\zeta\omega_n t}}{\sqrt{1-\zeta^2}}\sin\left(\omega_d t + \cos^{-1}\zeta\right) \qquad t > 0, \tag{5.7}$$

from which the transient performance measures can be obtained.
The peak time is

$$t_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}} \tag{5.8}$$

and the corresponding maximum overshoot is

$$M_{pt} = e^{-\pi\zeta/\sqrt{1-\zeta^2}} \tag{5.9}$$

Similarly, the rise time is

$$t_r = \frac{\pi - \cos^{-1}\zeta}{\omega_d}, \qquad 0 < \zeta < 1 \tag{5.10}$$

while the settling time is commonly approximated by

$$t_s \simeq \frac{4}{\zeta\omega_n}, \qquad \text{for the 2\% criterion} \tag{5.11}$$

or

$$t_s \simeq \frac{3}{\zeta\omega_n}, \qquad \text{for the 5\% criterion} \tag{5.12}$$

Once the desired transient specifications have been selected, the corresponding desired damping ratio ζ_d and natural frequency ω_{nd} can be determined. The desired damping ratio is obtained from the specified maximum overshoot as

$$\zeta_d = \sqrt{\frac{\ln^2(M_{pt})}{\pi^2 + \ln^2(M_{pt})}} \tag{5.13}$$

while the desired natural frequency is determined from either the desired peak time or settling time,

$$\omega_{nd} = \frac{\pi}{t_p\sqrt{1-\zeta_d^2}} \simeq \frac{4}{\zeta_d t_s} \tag{5.14}$$

The desired output response for a unit-step input is therefore given by

$$Y_d(s) = \frac{\omega_{nd}^2}{s\left(s^2 + 2\zeta_d\omega_{nd}s + \omega_{nd}^2\right)} \tag{5.15}$$

For comparison, the actual tracking error of the closed-loop system is

$$E(s) = \frac{1}{s\left(1 + G_c(s)G_p(s)\right)} \tag{5.16}$$

whereas the desired tracking error associated with the reference model is

$$E_d(t) = 1 - y_d(t) \tag{5.17}$$

or, equivalently, in the Laplace domain,

$$E_d(s) = \frac{1}{s} - Y_d(s) = \frac{s + 2\zeta_d\omega_{nd}}{s^2 + 2\zeta_d\omega_{nd}s + \omega_{nd}^2} \tag{5.18}$$

5.1.2 Optimization Approach

The PID controller parameters are determined by solving a numerical optimization problem. To formulate the frequency-domain optimization problem, Eq. (5.18) is evaluated at $s = j\omega$. The resulting desired error transfer function can be expressed in terms of its real and imaginary components as

$$E_d(j\omega) = \Re(E_d(j\omega)) + j\,\Im(E_d(j\omega)) \tag{5.19}$$

where

$$\Re(E_d(j\omega)) = \frac{2\zeta_d\omega_{nd}(-\omega^2+\omega_{nd}^2)}{(-\omega^2+\omega_{nd}^2)^2+2\zeta_d\omega_{nd}^2\omega^2} + \frac{2\zeta_d\omega_{nd}\omega^2}{(-\omega^2+\omega_{nd}^2)^2+2\zeta_d\omega_{nd}^2\omega^2} \tag{5.20}$$

and

$$\Im(E_d(j\omega)) = \frac{\omega(-\omega^2+\omega_{nd}^2)}{(-\omega^2+\omega_{nd}^2)^2+2\zeta_d\omega_{nd}^2\omega^2} - \frac{2\zeta_d\omega_{nd}^2\omega}{(-\omega^2+\omega_{nd}^2)^2+2\zeta_d\omega_{nd}^2\omega^2} \tag{5.21}$$

In addition, the real and imaginary parts can be separated in $E(s)$ as

$$E(j\omega) = \Re(E(j\omega)) + j\,\Im(E(j\omega)) \tag{5.22}$$

In this case, the optimization problem for satisfying the transient response specifications can be defined as an unconstrained optimization problem

$$\min \mathcal{J} = \int_0^{\omega_M} \left[\Re(E(j\omega)) - \Re(E_d(j\omega))\right]^2 + \left[\Im(E(j\omega)) - \Im(E_d(j\omega))\right]^2 \, d\omega \tag{5.23}$$

where $\omega_M > 0$ can be considered a relatively large number. There can also be constraints such as gain and phase margin specifications. In such cases, the controller design can be formulated as a constrained nonlinear optimization problem. For example, the phase margin specification can be expressed as

$$\arg\left[G_c(j\omega_g)G_p(j\omega_g)\right] = -\pi + \phi_m \tag{5.24}$$

The integral in (5.23) cannot be solved analytically, and therefore numerical methods should be employed. One of the accurate methods for numerical integration is called the Gaussian quadrature rule. In this section, the shifted Legendre–Gauss nodes and their corresponding quadrature weights are defined. The standard Legendre–Gauss nodes on the interval $(-1, 1)$ are defined as the roots of the Legendre polynomial $L_n(x)$ and are denoted by x_k, $k = 1, 2, \ldots, n$. The Legendre polynomials can be expressed on the interval $[-1, 1]$ as

$$L_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n \tag{5.25}$$

The most important property of the Legendre–Gauss points is the following quadrature rule

$$\int_{-1}^{1} f(x)\,dx \approx \sum_{k=1}^{n} q_k f(x_k) \tag{5.26}$$

This integral is exact for polynomials of degree at most $2n-1$, and the quadrature weights is calculated as [47]

$$q_k = \frac{2}{(1-x_k^2)\,[L_n'(x_k)]^2}, \qquad k = 1, \ldots, n \tag{5.27}$$

The shifted Legendre–Gauss nodes on the interval $[0, T_f]$ are denoted by ψ_k, $k = 1, \ldots, n$, and are given by

$$\psi_k = \frac{T_f}{2}(x_k + 1) \tag{5.28}$$

The corresponding quadrature weights are

$$\hat{q}_k = \frac{T_f}{2} q_k \tag{5.29}$$

Using this numerical approximation, the resulting optimization problem can be solved using a variety of optimization algorithms. If constraint (5.24) is imposed, a collocation approach can be employed, leading to a constrained optimization problem. Further details on collocation methods can be found in [47]. Several optimization software packages are available for solving both small- and large-scale optimization problems. To solve the optimization problem in (5.23), sequential quadratic programming (SQP) is employed using Maple software. More details on the SQP algorithm the reader is referred to [46].

5.1.3 Results and Discussions

This section presents the numerical results to evaluate the performance of the proposed method. All computations are performed using Maple 2022, employing the `method = sqp` option in the nonlinear programming solver (`NLPSolve`) [47]. For numerical integration, shifted Legendre–Gauss nodes of degree $n = 50$ are used. The upper frequency bound in the objective function (5.23) is selected as $\omega_M = 10$.

The performance of the method has been tested using transfer functions of different orders. Table 5.1 presents the desired specifications alongside the numerical results obtained using the proposed method. The results demonstrate that the method effectively satisfies the desired settling time and overshoot requirements. Figures 5.1 and 5.2 illustrate the system responses under different specifications for the transfer functions $G_1(s)$ and $G_2(s)$, respectively.

Table 5.1: Transient response t_s and overshoot (OS) for different specifications and simulation results.

Plant	**Specification**		**Result**	
	t_s	OS	t_s	OS
$G_1(s) = \dfrac{1}{s^2+5s+6}$	0.5	20%	0.4770	19.8897%
	1	2%	0.7022	2%
	5	0.2%	5.17	0.1997%
$G_2(s) = \dfrac{1}{s^3+6s^2+11s+6}$	1	2%	1.7780	5.8463%
	2	0.2%	2.9099	0.2907%
	5	0.2%	5.1589	0.2602%

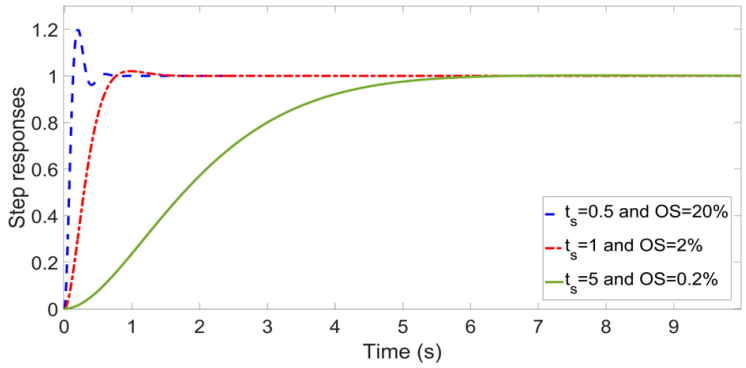


Figure 5.1: System response for different specifications of the transfer function $G_1(s)$.

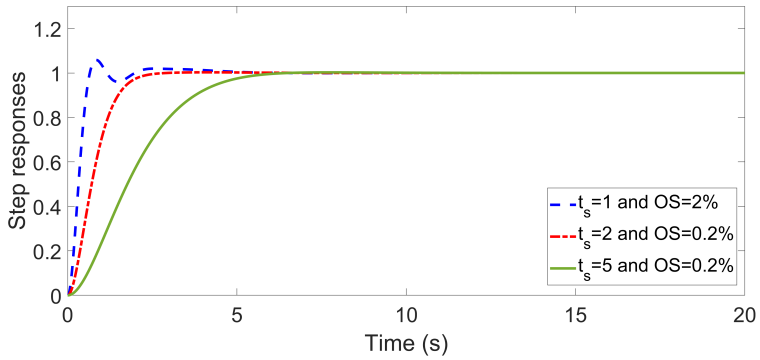


Figure 5.2: System response for different specifications of the transfer function $G_2(s)$.

In the next set of examples, two transfer functions, $G_3(s)$ and $G_4(s)$, are considered—one with a dead time and the other of fourth order. The corresponding results are summarized in Table 5.2. A comparison is conducted between the proposed method, the Ziegler–Nichols method, and the Genetic Algorithm. In all cases, the proposed method demonstrates superior performance in terms of settling time and overshoot. Figures 5.3 and 5.4 provide a visual comparison of the results for these cases.

Table 5.2: Comparison of the results for transient response t_s and OS for different specifications and simulation results.

Plant	**Ziegler–Nichols [20]**	**Genetic algorithm [17]**	**Suggested method**
$G_3(s) = \dfrac{e^{-0.5s}}{(1+s)^2}$	$K_p = 2.808$ $K_i = 1.712$ $K_d = 1.151$ $N = -$ $t_s = 6.1219$ OS = 43.6647% $t_r = 0.7776$	$K_p = 2.391$ $K_i = 1.072$ $K_d = 1.458$ $N = -$ $t_s = 7.0206$ OS = 9.4449% $t_r = 0.9924$	$K_p = 2.1902$ $K_i = 1.0934$ $K_d = 1.4068$ $N = 24842.75$ $t_s = 3.2961$ OS = 2.1285% $t_r = 0.7184$
$G_4(s) = \dfrac{27}{(s+1)(s+3)^3}$	$K_p = 3.072$ $K_i = 2.272$ $K_d = 1.038$ $N = -$ $t_s = 5.4366$ OS = 43.6064% $t_r = 0.7216$	$K_p = 1.772$ $K_i = 1.061$ $K_d = 0.772$ $N = -$ $t_s = 5.8441$ OS = 11.0296% $t_r = 1.1442$	$K_p = 1.135273$ $K_i = 0.7446209$ $K_d = 0.5400974$ $N = 40268.41$ $t_s = 3.8658$ OS = 0.7831% $t_r = 2.1720$

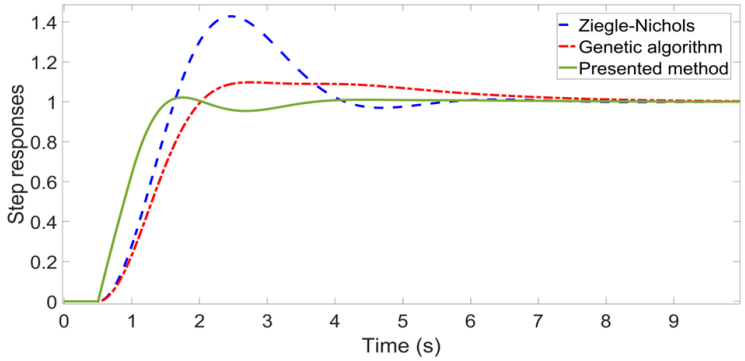


Figure 5.3: Comparison of system responses for $G_3(s)$ using the proposed method, Ziegler–Nichols, and Genetic algorithm.

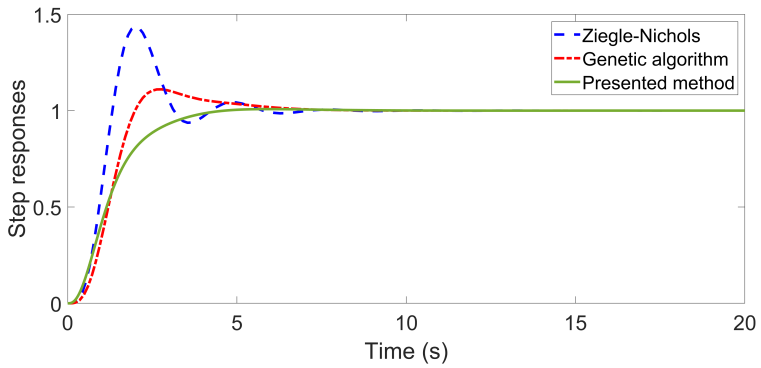


Figure 5.4: Comparison of system responses for $G_4(s)$ using the proposed method, Ziegler–Nichols, and Genetic algorithm.

It should be noted that the second-order reference model considered in Eq. (5.5) is used to represent transient response specifications such as overshoot, rise time, peak time, and settling time. However, the proposed optimization framework is not limited to second-order reference models. Alternative reference models may be adopted to represent different desired reference trajectories. Although the proposed frequency-domain method provides a systematic framework for PID controller design, it relies on a linearized model of the system and is typically developed offline around a specific operating point. Consequently, its performance may deteriorate when the system dynamics change. Since EHA systems operate in nonlinear and dynamic environments, an online adaptive control strategy is desirable to maintain control performance under varying operating conditions.

5.2 Model-Free Learning for Online PID Controller Tuning

To address the limitations of fixed-gain PID control, adaptive tuning methods have been developed to adjust controller gains online based on real-time performance assessment. This category includes classical adaptive control methods as well as recent data-driven and reinforcement learning (RL) approaches [95, 96]. RL has particularly emerged as a powerful tool for solving continuous control problems [97–99]. Actor-critic algorithms such as Deep Deterministic Policy Gradient and Twin Delayed Deep Deterministic policy gradient algorithms learn control policies directly from interaction data and have been applied to continuous-control benchmarks as well as PID tuning problems. While these methods offer improved robustness to time-varying dynamics, ensuring machine safety and interpretability during online tuning remains a major challenge, particularly for safety-critical hydraulic systems [100].

These challenges motivate model-free strategies that preserve the advantages of RL, while avoiding the complexity of deep neural network policies. Augmented random search (ARS) is one

such alternative. ARS is a derivative-free reinforcement-learning method that perturbs controller parameters along random symmetric directions and updates them based on performance differences. Despite its simplicity, ARS has demonstrated competitive performance in continuous-control tasks and is well-suited for offline training where broad exploration is desirable [101]. Importantly, ARS directly optimizes the PID controller gains, rather than learning black-box neural-network policies, making the resulting controller interpretable and compatible with standard industrial practices.

A Gaussian process (GP) is a Bayesian learning method widely used for modelling unknown functions from limited data. GP-based approaches have been successfully applied to robotic control and PID tuning in data-limited settings [102]. GP can provide a complementary capability by learning a smooth surrogate model of the relationship between PID gains and control performance. It can be used as a surrogate model of the relationship between PID gains and closed-loop control performance.

Motivated by these considerations, this chapter proposes a PID controller tuning framework for EHA that combines ARS-based exploration with GP surrogate modelling. The method includes both offline and online tuning phases. In the offline tuning phase, the ARS is used to explore the PID gain space using simulation, generating a diverse set of gain–performance samples and identifying a baseline controller. A GP surrogate model is then trained using this offline dataset and serves as a prior representation of the performance landscape for online tuning. During online operation, where system dynamics change, the GP model is updated using newly observed performance data. This hybrid method enables safe and interpretable online PID controller adaptation.

5.2.1 Controller Structure

Let $x_p(t)$ denote the hydraulic actuator position and $r(t)$ the reference signal. The tracking error is defined as

$$e(t) = r(t) - x_p(t) \tag{5.30}$$

The PID control law is expressed as

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\varsigma)\, d\varsigma + K_d\, \dot{e}(t) \tag{5.31}$$

where K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively, and $u(t)$ denotes the control input.

The controller gains are adjusted through a learning process that interacts directly with the EHA simulation model. The learning variables correspond directly to the PID gain parameters. The observation vector provided to the learning algorithm consists of the tracking error, its time derivative, and the integral of the error.

5.2.2 Offline Tuning Phase

The offline phase is designed to obtain the initial optimized PID gains for the online tuning phase by combining a model-free RL strategy with probabilistic surrogate modelling. In this phase, augmented random search (ARS) is employed to explore the PID gain space using a simulation model of the EHA [101]. It allows a safe exploration without risking unstable behavior on physical system. The data generated during this process are then used to construct a GP model that captures the relationship between controller gains and tracking performance. This GP model provides a compact Bayesian representation of the offline learning outcome, which is subsequently used in the online phase to guide gain tuning when the system dynamics change.

5.2.2.1 ARS-Based Exploration with GP Surrogate Modelling

The hybrid of ARS-based search and GP modelling is used to determine the PID gain vector

$$K = \begin{bmatrix} K_p & K_i & K_d \end{bmatrix}^\top \tag{5.32}$$

such that tracking performance is maximized over a fixed simulation horizon. The return $R(K)$ for the gain vector K is defined as the negative accumulated tracking error over the evaluation horizon

$$R(K) = -\int_0^{T_f} e^2(t)\, dt \tag{5.33}$$

The offline phase seeks

$$K^* = \arg \max_{\ell_b \le K \le u_b} R(K) \tag{5.34}$$

where ℓ_b and u_b denote lower and upper gain bounds.

The method is performed in two stages. An initial random design is first used to explore the admissible gain space. The gain vector yielding the highest return among these random samples is then selected as the initial gains for the subsequent ARS optimization phase. The random initial gains and each newly evaluated sample are added to the GP dataset, and the surrogate model is retrained.

To initialize the surrogate model, N_{init} random gain vectors are sampled uniformly from the admissible domain

$$K^{(i)} = \ell_b + (\, u_b - \ell_b \,) \odot \gamma^{(i)}, \qquad \gamma^{(i)} \sim \mathcal{U}(0, 1) \tag{5.35}$$

where $\odot$ denotes element-wise multiplication, and $\mathcal{U}(0, 1)$ represents the uniform distribution over the interval $[0, 1]$. Each sampled gain vector is evaluated using the Simulink model to obtain its

corresponding return. These initial gain–return pairs form the initial data set

$$\mathcal{S} = \left\{ (K^{(i)},\ R^{(i)}) \right\}_{i=1}^{N_{\text{init}}} \tag{5.36}$$

which is subsequently expanded with additional evaluations during both the offline optimization and the online tuning phases. After initialization with the gain yielding the lowest tracking error (equivalently, the highest return) among randomly sampled gains, ARS iteratively improves the gain vector. At iteration t, N_{dirs} random search directions

$$\delta_j \sim \mathcal{N}(0, I), \qquad j = 1, \ldots, N_{\text{dirs}}$$

are generated, where $\mathcal{N}(0, I)$ denotes a multivariate normal distribution with zero mean and identity covariance matrix. For each search direction, two symmetric perturbations are generated

$$K_j^+ = K_t + \eta\, \delta_j \tag{5.37}$$

$$K_j^- = K_t - \eta\, \delta_j \tag{5.38}$$

where η is the exploration scale. Each candidate is bounded within the admissible domain

$$\bar{K}_j^+ = \min\{\max\{K_j^+,\ \ell_b\},\ u_b\} \tag{5.39}$$

$$\bar{K}_j^- = \min\{\max\{K_j^-,\ \ell_b\},\ u_b\} \tag{5.40}$$

The corresponding returns R_j^+ and R_j^- are computed by evaluating the Simulink model using $\bar{K}_j^+$ and $\bar{K}_j^-$, respectively. The gradient estimate is calculated as

$$g_t = \frac{1}{b} \sum_{j \in \mathcal{I}_b} \left(R_j^+ - R_j^- \right) \delta_j \tag{5.41}$$

where $\mathcal{I}_b$ denotes the indices of the best b directions with the largest scores.

A normalization factor is computed from the standard deviation of all returns

$$\hat{\sigma}_t = \sqrt{\mathrm{Var}\left(\{R_j^+,\ R_j^-\}_{j=1}^{N_{\text{dirs}}} \right)} + \varepsilon \tag{5.42}$$

where ε is a small constant to avoid division by zero. The gain vector is updated as

$$K_{t+1} = K_t + \frac{\alpha}{\hat{\sigma}_t} g_t \tag{5.43}$$

where α is the ARS learning rate. Finally, the best-performing gains are tracked across iterations

$$K_{\text{best}} = \arg \max_{K \in \{K_1, \ldots, K_{t+1}\}} R(K) \tag{5.44}$$

$$R_{\text{best}} = \max_{K \in \{K_1, \ldots, K_{t+1}\}} R(K) \tag{5.45}$$

A GP model $\hat{R}(K)$ is then trained to approximate the relationship between the controller gains and the corresponding return

$$\hat{R}(K) \sim \mathcal{GP}\big(0,\ k(K, K')\big) \tag{5.46}$$

where $\mathcal{GP}\big(0,\ k(K, K')\big)$ denotes a GP prior with zero mean and covariance function $k(\cdot, \cdot)$, chosen here as a squared-exponential kernel with an independent length scale for each input dimension as

$$k(K, K') = \sigma_f^2 \exp\left(-\frac{1}{2}(K - K')^\top \Lambda^{-1}(K - K')\right) \tag{5.47}$$

where σ_f is the signal standard deviation of the GP, and $\Lambda = \text{diag}(r_1^2, r_2^2, r_3^2)$ contains the characteristic length scales [20]. After every ARS iteration:

1. The new evaluations $(\bar{K}_j^+, R_j^+)$ and $(\bar{K}_j^-, R_j^-)$ are appended to the GP dataset (refer to Eq. 5.36);

2. The GP model is retrained using all accumulated data;

3. The best return $R(K)$ is tracked and recorded.

To train the ARS algorithm and obtain the optimal controller gains during the offline phase, the following sinusoidal reference trajectory is used:

$$r(t) = \begin{cases} 0, & t < 6\,\text{s}, \\ 0.2\sin(0.05\pi(t-6)), & t \geq 6\,\text{s}. \end{cases} \tag{5.48}$$

The optimal controller gains are obtained as $K_{\text{best}} = [47.30, 40.30, 14.11]^T$ using the proposed framework. Figure 5.5 shows the closed-loop tracking response, x_p, obtained with the optimal gain vector K_{best}, compared with the reference signal. Figure 5.6 shows the corresponding rod velocity, $\dot{x}_p$.

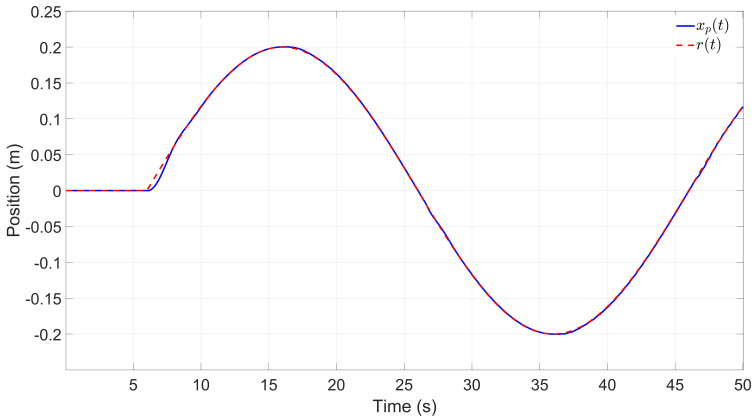


Figure 5.5: Closed-loop tracking response with the best gain vector $K_{\text{best}} = [47.30, 40.30, 14.11]^T$: actuator position response, x_p compared with the reference signal.

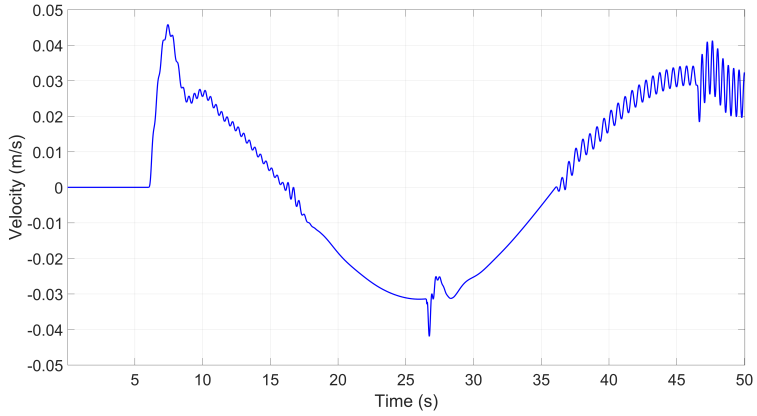


Figure 5.6: Velocity response, $\dot{x}_p$ obtained with the optimal gain vector $K_{\text{best}} = [47.30, 40.30, 14.11]^T$.

The control signal $u(t)$ is depicted in Fig. 5.7, associated with the PID controller using K_{best}. Figure 5.8 shows the resulting tracking error $e(t)$. Figure 5.9 illustrates the evolution of the PID gain vector $K = [K_p, K_i, K_d]^\top$ over successive ARS iterations highlights how the search procedure refines the controller gains. The learning progress of the ARS optimization process is shown in Fig. 5.10, which presents the return achieved at each iteration. The corresponding key parameters are also summarized in Table 5.3.

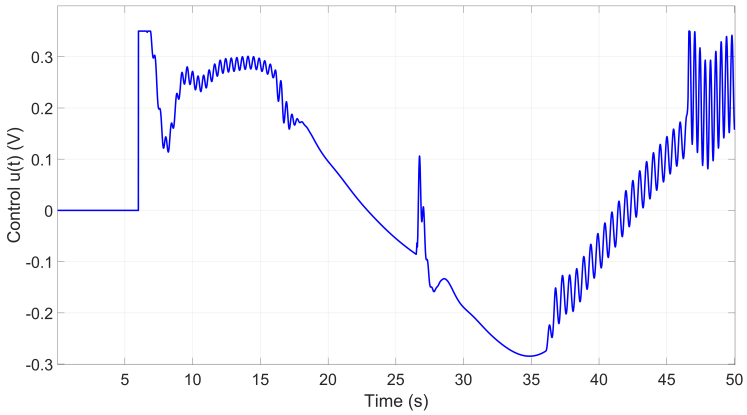


Figure 5.7: Control input signal, u generated by the PID controller using the best gain vector $K_{\text{best}} = [47.30;;40.30;;14.11]^T$.

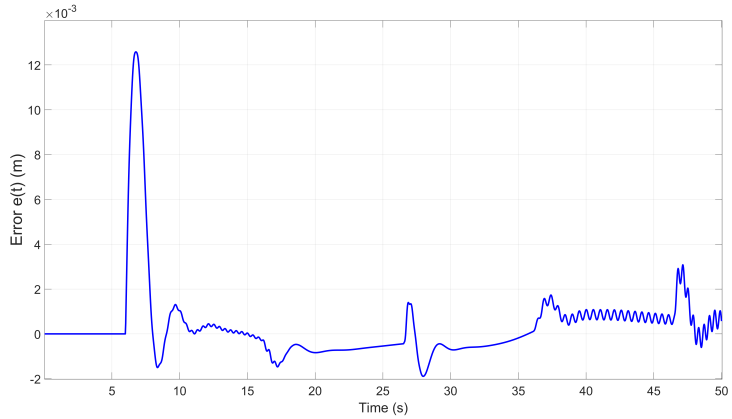


Figure 5.8: Tracking error obtained with the best gain vector $K_{\text{best}} = [47.30, 40.30, 14.11]^T$.

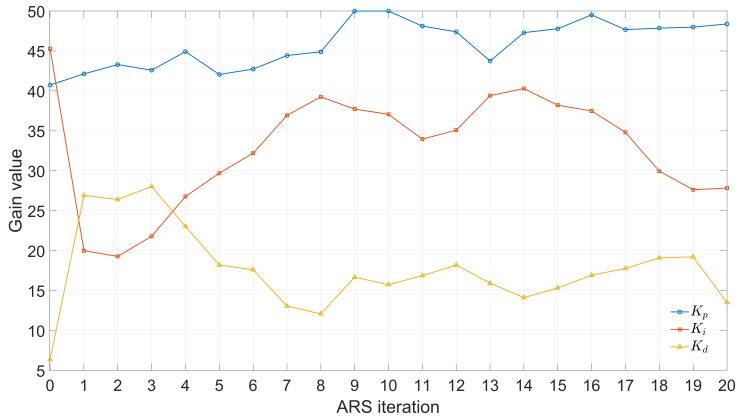


Figure 5.9: Evolution of the gains (K_p, K_i, K_d) over augmented random search (ARS) iterations. Iteration 0 corresponds to the best controller gains from the initial random design.

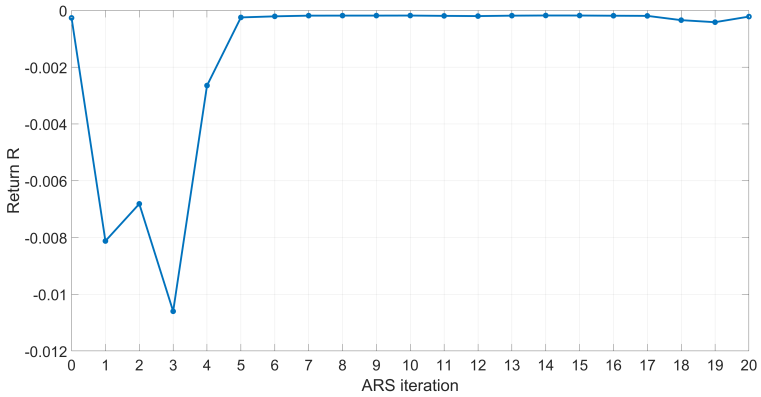


Figure 5.10: Return per ARS iteration after the initial random design; iteration 0 corresponds to the best controller obtained from the initial random design before ARS updates begin.

Table 5.3: Parameters used in the augmented random search (ARS) and Gaussian process (GP) during the offline tuning phase.

Description	**Symbol**	**Value**
Episode length	T_f	50 s
Lower gain bounds	l_b	$[0, 0, 0]^T$
Upper gain bounds	u_b	$[50, 50, 50]^T$
ARS learning rate	α	1.0
Exploration scale	η	1.0
Learned length scale (K_p)	r_1	18.4885
Learned length scale (K_i)	r_2	4.8989
Learned length scale (K_d)	r_3	5.2284
Signal standard deviation	σ_f	0.1759
Directions per iteration	N_{dirs}	8
Selected top directions	b	4
ARS iterations	N_{iters}	20
Initial random samples	N_{init}	10
Small constant	ε	10^{-8}

5.2.2.2 Evaluation of GP Accuracy

To ensure that the GP provides an accurate surrogate model for the online tuning phase, it is evaluated on N_{test} unseen gain vectors $\{K_{\text{test}}^{(i)}\}_{i=1}^{N_{\text{test}}}$. For each test gain vector, the GP predictive mean $\hat{R}\left(K_{\text{test}}^{(i)}\right)$ is compared with the corresponding return $R_{\text{true}}^{(i)}$ obtained from the Simulink simulation. The predictive performance is assessed using the root mean square error (RMSE), the normalized root mean square

error (NRMSE), and the correlation coefficient ρ, defined as

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \left(R_{\text{true}}^{(i)} - \hat{R}\left(K_{\text{test}}^{(i)}\right)\right)^2} \tag{5.49}$$

$$\text{NRMSE} = \frac{\text{RMSE}}{\max_i R_{\text{true}}^{(i)} - \min_i R_{\text{true}}^{(i)}} \tag{5.50}$$

$$\rho = \text{corr}\left(\{R_{\text{true}}^{(i)}\}_{i=1}^{N_{\text{test}}}, \{\hat{R}(K_{\text{test}}^{(i)})\}_{i=1}^{N_{\text{test}}} \right) \tag{5.51}$$

A GP model is considered acceptable if $\rho > 0.8$ and NRMSE < 0.2. If these conditions are not satisfied, the accuracy can typically be improved by adjusting the parameters of the proposed ARS exploration. Based on the final evaluation on $N_{\text{test}} = 20$ unseen gains, the proposed GP model achieves $\rho = 0.9871$ and NRMSE $= 0.0488$. It satisfies both criteria and demonstrates strong predictive accuracy.

Figure 5.11 shows the predictive performance of the final GP surrogate model. Figure 5.12 illustrates a one-dimensional slice of the GP surrogate model along K_p, with K_i and K_d held fixed at their optimized values from K_{best}. The shaded region represents the posterior uncertainty (2σ), which decreases significantly in regions densely explored by the ARS algorithm. This behavior indicates that the GP surrogate becomes increasingly confident near the optimal gain region while maintaining higher uncertainty in sparsely sampled areas, consistent with Bayesian learning principles. Similar trends are observed in Figs. 5.13 and 5.14 for slices along K_i and K_d.

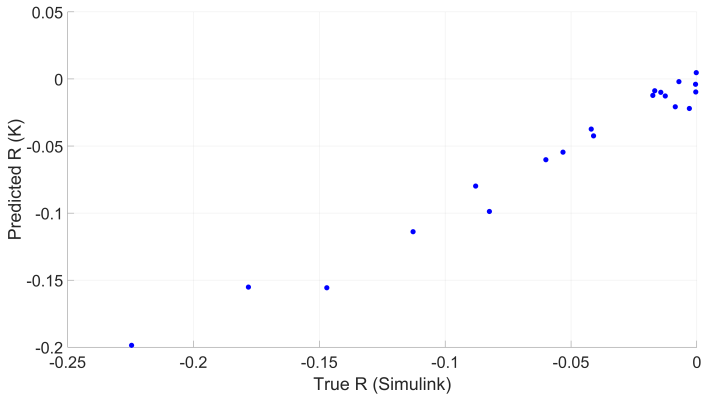


Figure 5.11: Validation of the Gaussian process (GP) model on unseen gain vectors after the offline phase: scatter plot of GP-predicted return versus the true return obtained from simulations.

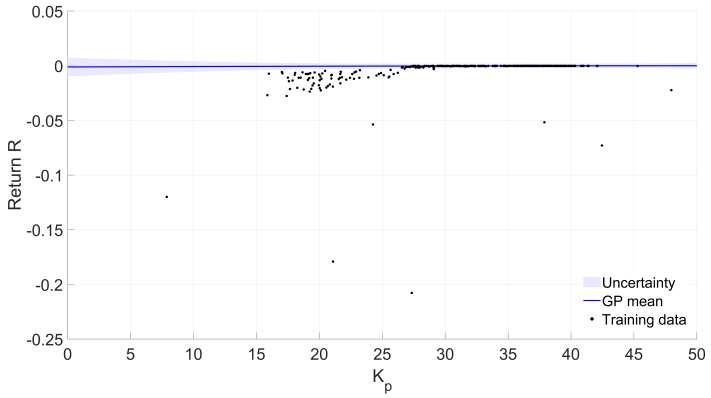


Figure 5.12: One-dimensional slice of the GP model versus the gain K_p.

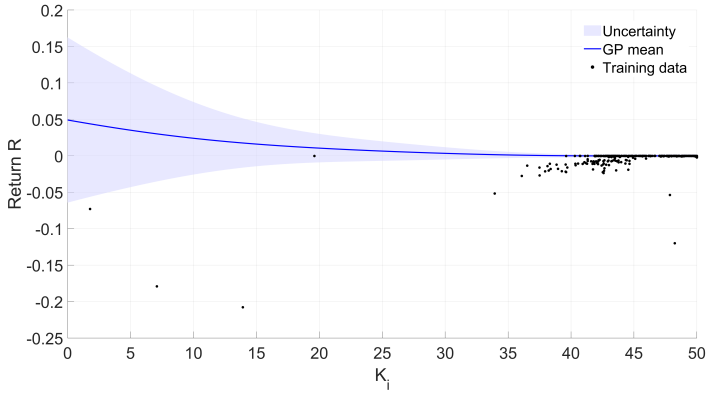


Figure 5.13: One-dimensional slice of the GP model versus the gain K_i.

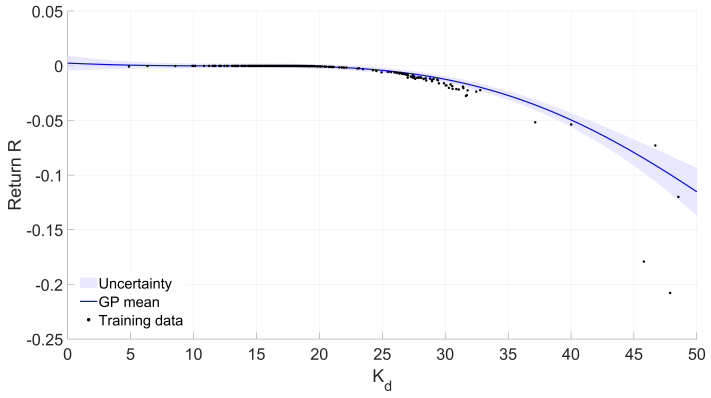


Figure 5.14: One-dimensional slice of the GP model versus the gain K_d.

To evaluate the robustness of the controller gains obtained during the offline phase, three reference trajectories are considered: a step input, a square-wave input, and a sinusoidal input. All reference trajectories are expressed in meters (m) and are defined as follows

(i) Step input:

$$r_{\text{step}}(t) = \begin{cases} 0, & t < 6\,\text{s} \\ 0.2, & t \geq 6\,\text{s} \end{cases} \tag{5.52}$$

(ii) Square-wave input:

$$r_{\text{sq}}(t) = \begin{cases} 0, & t < 6\,\text{s} \\ 0.2\,\text{sign}(\sin(0.06\pi(t-6))), & t \geq 6\,\text{s} \end{cases} \tag{5.53}$$

(iii) Sinusoidal input:

$$r_{\text{sin}}(t) = \begin{cases} 0, & t < 6\,\text{s} \\ \sum_{i=1}^{3} 0.05\sin(2\pi f_i(t-6)), & t \geq 6\,\text{s} \end{cases} \tag{5.54}$$

where $f_1 = 0.01$ Hz, $f_2 = 0.04$ Hz, and $f_3 = 0.06$ Hz.

Figure 5.15 illustrates the corresponding tracking performance under these reference inputs. The results suggest that acceptable tracking performance is achieved when optimized gains K_{best} are applied to different reference inputs.

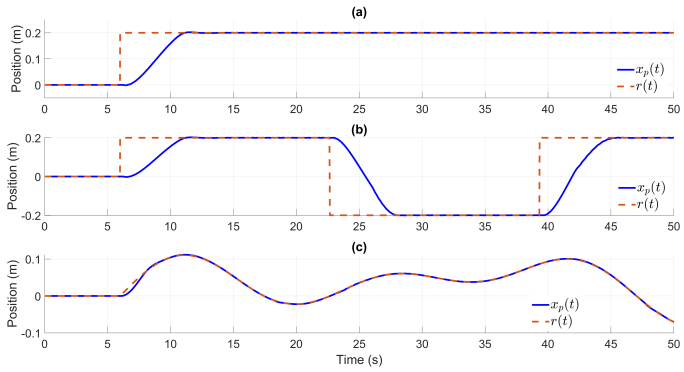


Figure 5.15: Tracking response of the electro-hydrostatic actuator (EHA) system: (a) step response; (b) square-wave response; (c) sinusoidal response.

5.2.3 Online Tuning Phase

Following the offline training phase, an adaptive online tuning strategy is proposed to update the controller gains in response to changes in system dynamics. The strategy is implemented in Simulink using a reference trajectory that enables the controller to operate under realistic conditions. A GP surrogate model trained for the nominal plant is used to guide the online gain updates.

5.2.3.1 Reference Trajectory Design

The reference trajectory considered for the online evaluation is denoted by $r_{\text{ref}}(t)$ and is considered over a 600 s evaluation period. Figure 5.16 illustrates the corresponding reference signal.

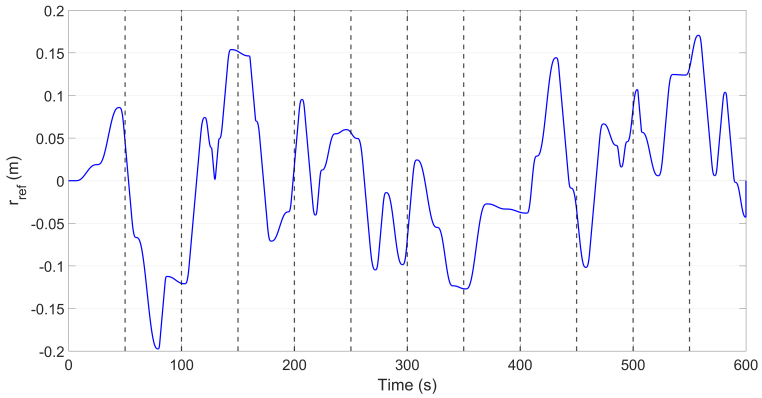


Figure 5.16: Desired position trajectory used for the online evaluation over the entire simulation. The vertical dashed lines indicate the boundaries between the 12 non-identical consecutive episodes, each with a duration of 50 s.

To avoid sudden motion at the beginning of the simulation, the reference signal is held at zero until the motion start time t_0

$$r_{\text{ref}}(t) = 0, \qquad 0 \leq t < t_0 \tag{5.55}$$

For $t \geq t_0$, the reference trajectory is constructed as a sequence of motion segments. The duration of the j-th segment, denoted by $\Delta t^{(j)}$, is randomly selected according to

$$\Delta t^{(j)} \sim \mathcal{U}(\Delta t_{\min}, \Delta t_{\max}) \tag{5.56}$$

where $\Delta t_{\min}$ and $\Delta t_{\max}$ are given in Table 5.4. This randomization introduces variability in the excitation while ensuring that each motion segment is maintained for a minimum duration.

The transition from the current reference value $r_{\text{ref}}^{(j)}$ to the target reference value $r_{\text{ref,goal}}^{(j)}$ is defined

using a smooth quintic polynomial trajectory [103]

$$r_{\text{ref}}(t) = r_{\text{ref}}^{(j)} + \left(r_{\text{ref,goal}}^{(j)} - r_{\text{ref}}^{(j)}\right) \alpha(\varsigma) \tag{5.57}$$

where

$$\alpha(\varsigma) = 10\varsigma^3 - 15\varsigma^4 + 6\varsigma^5, \qquad \varsigma = \frac{t - t_j}{\Delta t^{(j)}}, \qquad 0 \le \varsigma \le 1 \tag{5.58}$$

This quintic trajectory ensures continuity of position, velocity, and acceleration at the segment boundaries, resulting in smooth motion. To enforce realistic actuator dynamics, the discrete-time reference velocity is limited by symmetric saturation

$$|\dot{r}_{\text{ref}}(t)| \le V_{\max} \tag{5.59}$$

where $V_{\max}$ is the nominal velocity limit. Finally, the reference position is limited to satisfy the amplitude constraint

$$|r_{\text{ref}}(t)| \le A_{\max} \tag{5.60}$$

ensuring that the commanded motion always remains within physically admissible bounds. The end value of each segment is used as the initial condition for the next segment, which guarantees continuity over the full time horizon. The parameters for the reference trajectory are listed in Table 5.4.

Table 5.4: Parameters of the reference signal.

Description	**Symbol**	**Value**
Amplitude limit	$A_{\max}$	0.20 m
Velocity limit	$V_{\max}$	0.015 m/s
Minimum segment duration	$\Delta t_{\min}$	3 s
Maximum segment duration	$\Delta t_{\max}$	20 s
Motion start time	t_0	6 s

5.2.3.2 Online Gain Tuning

In the online phase, the controller is evaluated over a sequence of episodes. The total horizon is expressed as

$$T_{\text{tot}} = (N_{\text{base}} + N_{\text{adjust}})T_f \tag{5.61}$$

where T_f denotes the episode length, N_{base} is the number of baseline episodes used for comparison under nominal conditions, and N_{adjust} is the number of episodes used for online tuning after the system dynamics change.

The k-th episode corresponds to the time interval

$$t \in [(k-1)T_f,\ kT_f], \qquad k = 1, 2, \ldots, N_{\text{base}} + N_{\text{adjust}} \tag{5.62}$$

For each episode (refer to Fig. 5.16), the controller is evaluated on a local segment $r_k(t)$ extracted from the global reference trajectory. In the simulation, the final state of episode k is used as the initial state of episode $k+1$, ensuring continuity of both the system state and the reference signal.

During the first N_{base} episodes, the system operates under nominal conditions, as in the offline tuning phase, and the controller gains K_{best} obtained from the offline phase are applied without modification. This segment provides a baseline for comparison.

After this period, the bulk modulus $\beta_e = 400 \times 10^6$ Pa, is reduced by a factor of five to $\hat{\beta}_e = 80 \times 10^6$ Pa, representing a change in system dynamics. This changes the closed-loop dynamics causing a reduced tracking accuracy. The remaining N_{adjust} episodes are then used to evaluate and adjust the controller gains online. The same reward definition $R(K)$ in (5.33) is used to evaluate the controller performance in each episode.

The online tuning process proceeds iteratively until the achieved performance becomes sufficiently close to that obtained during the offline phase. To avoid unnecessary exploration, a stopping criterion is introduced. Let $R_{\text{best,offline}}$ denote the best return obtained during the offline optimization phase. A performance threshold is defined as

$$R_{\text{stop}} = \alpha R_{\text{best,offline}} \tag{5.63}$$

where α is selected experimentally. Online gain adaptation continues only while

$$R_{\text{best,online}} < R_{\text{stop}}$$

Once

$$R_{\text{best,online}} \geq R_{\text{stop}}$$

the exploration phase is terminated, and the controller operates with fixed gains for the remainder of the simulation.

The online gain tuning procedure is summarized as follows.

(i) To quantify the degradation in tracking performance caused by the change in system dynamics, the first online episode evaluates the offline optimal gain vector K_{best} on the perturbed plant. The resulting return is

$$R^{(1)}_{\text{online}} = R\big(K_{\text{best}}; \hat{\beta}_e\big) \tag{5.64}$$

(ii) If

$$R_{\text{best,online}} < R_{\text{stop}}$$

the GP model $\hat{R}(K)$, initialized during the offline phase, is used as a surrogate model to guide the online gain search. Instead of exploring the whole gain space again, at each online tuning iteration, a set of candidate gains $\{K_j\}_{j=1}^{N_{\text{cand}}}$ is generated in the neighborhood of the current best online gain according to

$$K_j = (1-\rho)K_{\text{best,online}} + \rho K_j^{\text{rand}}, \qquad \rho \in (0,1) \tag{5.65}$$

where N_{cand} denotes the number of candidate gain vectors, $K_{\text{best,online}}$ is the current best online gain vector, and K_j^{rand} is sampled uniformly from the admissible gain bounds $[\ell_b, u_b]$.

The GP predicts the performance of each candidate by providing its predictive mean $\mu(K_j)$ and standard deviation $\sigma(K_j)$. An upper confidence bound acquisition function is then computed as

$$a(K_j) = \mu(K_j) + \kappa_{\text{UCB}}\sigma(K_j) \tag{5.66}$$

where $\kappa_{\text{UCB}} > 0$ controls the exploration–exploitation trade-off. The next gain vector is selected according to

$$K_{\text{next}} = \arg\max_{K_j} a(K_j) \tag{5.67}$$

(iii) The selected gain vector K_{next} is evaluated using the full Simulink model, yielding the return

$$R_{\text{next}} = R\big(K_{\text{next}}; \hat{\beta}_e\big)$$

The pair

$$(K_{\text{next}}, R_{\text{next}})$$

is appended to the existing data set $\mathcal{S}$ (refer to Eq. (5.36)), after which the GP hyperparameters are re-estimated. The current best online gain is then updated as

$$K_{\text{best,online}} = \arg\max_{(K,R)\in\mathcal{S}_{\text{online}}} R \tag{5.68}$$

where $\mathcal{S}_{\text{online}}$ contains only the measurements collected during the online phase.

This method provides a systematic way to evaluate how the GP-based adaptation recovers performance after a change in system dynamics. The parameters used in the online framework are summarized in Table 5.5. The online tuning behavior of the proposed framework and the resulting closed-loop performance under changing system dynamics are summarized in Figs. 5.16–5.20.

Table 5.5: Parameters used in the online tuning phase.

Description	**Symbol**	**Value**
Offline best gains	K_{best}	$[47.30 \;\; 40.30 \;\; 14.11]^T$
Stopping threshold parameter	α	1.5
Offline best return	$R_{\text{best,offline}}$	-1.7890×10^{-04}
Episode length	T_f	50 s
Baseline episodes (nominal dynamics)	N_{base}	2
tuning episodes (changed dynamics)	N_{adjust}	10
Offline stiffness parameter	β_e	400×10^6 Pa
Online stiffness parameter	$\hat{\beta}_e$	80×10^6 Pa
UCB exploration parameter	κ_{UCB}	0.3
Candidates per online iter	N_{cand}	200
Candidate mixing factor	ρ	0.3
Reference sample time	Δt	0.001 s

The episode-by-episode evolution of the PID gain vector $K = [K_p, K_i, K_d]^\top$ during the online evaluation is illustrated in Fig. 5.17. The figure shows how the gains evolve from the initial values K_{best} toward the final stabilized gains as exploration is terminated after three episodes.

The progression of the controller performance across episodes is summarized in Fig. 5.18, which reports both the instantaneous reward obtained in each episode after the change in system dynamics and the corresponding best-so-far reward. While the instantaneous reward reflects the tracking performance achieved in each episode, the best-so-far curve highlights the cumulative improvement trend and indicates the episode at which the stopping criterion is satisfied.

Figure 5.19 shows the closed-loop tracking response over the entire simulation. The initial baseline episodes correspond to nominal plant conditions, where the controller operates with the fixed gains K_{best} obtained from the offline tuning stage. After the system parameter β_e is changed, the subsequent episodes allow the controller gains to be adjusted episode by episode. The resulting response illustrates how the controller compensates for the changed dynamics, with tracking performance improving during the tuning period and stabilizing once exploration stops.

Figure 5.20 shows the corresponding control input. It represents the control action applied by the controller, including both the baseline operation under nominal conditions and the subsequent gain tuning following the change in β_e.

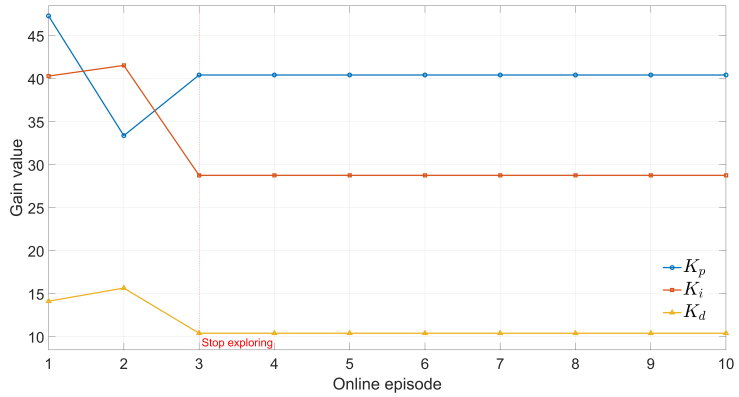


Figure 5.17: Evolution of PID controller gains (K_p, K_i, K_d) across episodes. After the change in β_e, the algorithm explores the gain space for three episodes and terminates exploration once improved gains are identified.

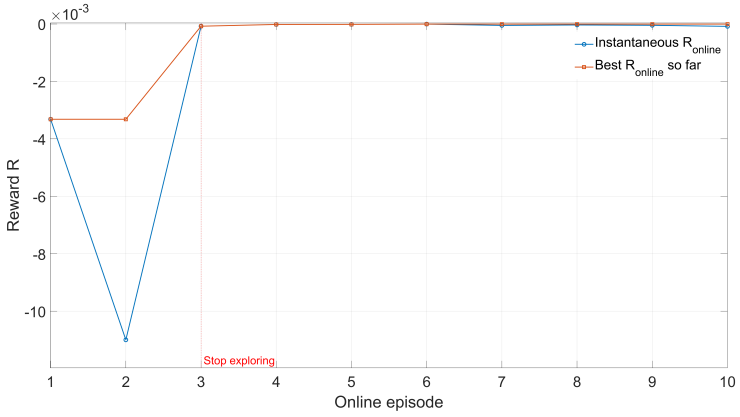


Figure 5.18: Instantaneous and best online rewards during the online tuning process. The vertical dashed line marks the episode where exploration is terminated.

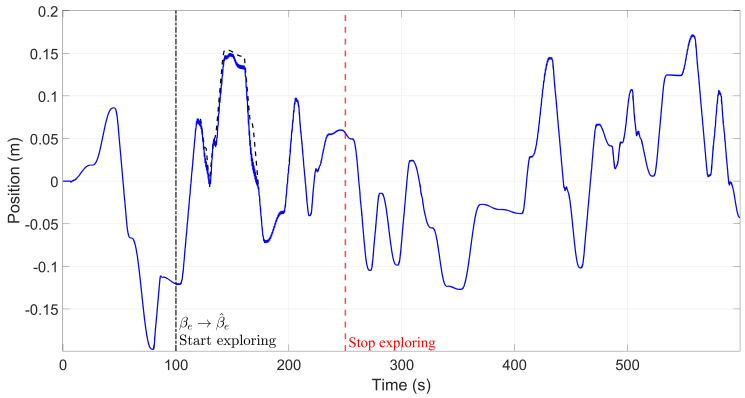


Figure 5.19: Tracking response of the EHA system over the entire simulation. The dashed curve represents the reference trajectory, while the solid curve shows the system response. The first vertical dashed line indicates the change in the system parameter β_e. The second vertical dashed line marks the episode at which exploration is terminated, and the controller gains are fixed.

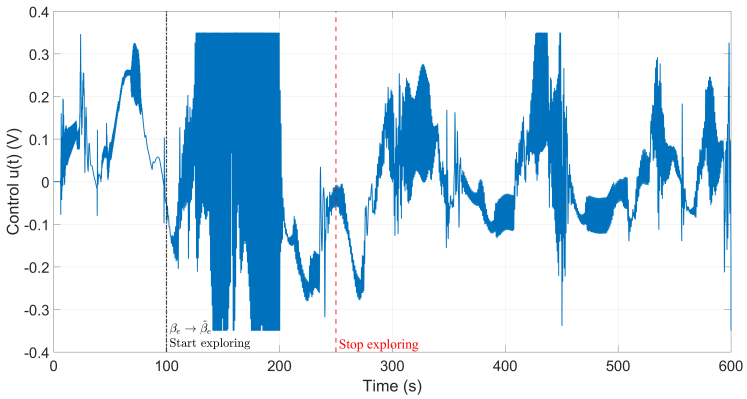


Figure 5.20: Control signal applied to the EHA system. Vertical dashed lines indicate the change in the system parameter β_e and the episode at which gain exploration is terminated.

5.3 Summary

This chapter presented two methods for PID controller tuning in electro-hydrostatic actuator (EHA) systems, namely model-based and model-free methods. The model-based method provides a classical method for tuning the PID controller gains based on frequency-domain optimization.

To overcome the limitations of fixed-gain controllers under varying system dynamics, a model-free learning framework was developed. The proposed method combines augmented random search for efficient exploration of the gain space with a Gaussian process surrogate model that learns the relationship between controller gains and the resulting control performance.

The results demonstrate that while model-based tuning provides a baseline for tuning PID parameters under nominal conditions, it lacks adaptability to changes in system parameters. In contrast, the proposed model-free framework enables efficient and safe online tuning of PID gains, allowing the controller to adapt to changing system dynamics.

Chapter 6

Energy Optimization

Energy efficiency is a key consideration in the design and operation of hydraulic systems, particularly in applications where energy consumption affects operating costs, system performance, and environmental impact. For battery-powered machines, a major motivation for improving energy efficiency is to extend operating time between charges. Reducing the energy consumption of hydraulic actuation can also reduce the battery capacity required for a given duty cycle, potentially lowering battery weight and cost. Compared with conventional valve-controlled hydraulic systems, electro-hydrostatic actuator (EHA) systems generally offer higher energy efficiency by eliminating continuous valve throttling losses. In addition, EHAs provide advantages such as a compact structure and decentralized architecture. However, despite these advantages, further improvement of energy utilization remains an important research challenge [104, 105].

Recent research has explored various strategies to enhance the energy efficiency of EHAs, including model-based control, advanced flow regulation techniques, and integration of energy storage systems [105]. One promising direction is the incorporation of energy storage and reutilization systems using hydraulic accumulators, which allow surplus energy to be stored temporarily and reused when needed [10]. This method can significantly reduce energy consumption and improve overall system efficiency while maintaining performance.

This chapter presents the energy optimization of an EHA system equipped with an ESR unit. The objective is to determine the controller parameters that provide an appropriate trade-off between trajectory tracking accuracy and energy efficiency. Unlike a conventional EHA system, the integration of the ESR unit introduces energy consumption as an explicit design objective alongside tracking performance. As a result, the controller tuning problem is formulated as a multi-objective optimization problem. Such optimization frameworks enable the simultaneous treatment of conflicting objectives, such as minimizing tracking error and reducing energy consumption [106]. These methods generate a set of Pareto-optimal solutions, each representing a different trade-off between objectives [107]. By exploring a diverse set of solutions rather than converging to a single

optimum, multi-objective optimization allows the designer to select a controller that best meets the practical performance and efficiency requirements of the system.

In this chapter, a population-based multi-objective genetic algorithm is used to explore the admissible controller parameter space and to identify a set of Pareto-optimal solutions. This method avoids scalarization biases and provides designers with a spectrum of feasible operating points rather than a single tuned solution [108]. A surrogate modelling technique based on the Gaussian process (GP) is also incorporated into the optimization process to guide the initialization of the evolutionary search. Simulation results demonstrate that the proposed multi-objective optimized PID controller achieves control accuracy while significantly reducing energy consumption. The work presented in this chapter is being submitted to the International Journal of Fluid Power[1].

6.1 Objective and Framework

Let $x_p(t)$ denote the actuator position and $r(t)$ the desired reference trajectory. The primary control task is to regulate $x_p(t)$ such that it follows $r(t)$ with minimal tracking error. The position control of EHAs using a PID controller is given by

$$u(t) = K_p e(t) + K_i \int_0^t e(\varsigma)\, d\varsigma + K_d \frac{de(t)}{dt} \tag{6.1}$$

where $e(t) = r(t) - x_p(t)$ denotes the tracking error.

Energy efficiency can be considered as an additional design requirement due to the integration of the ESR system. The ESR unit enables hydraulic energy to be stored and reused when operating conditions allow. However, its effectiveness depends on internal activation thresholds governing accumulator charging and energy reutilization. Consequently, the closed-loop performance of the EHA system depends not only on the PID gains (K_p, K_i, K_d) but also on the ESR threshold parameters. Therefore, a new control logic for the energy storage and reutilization is proposed based on the operating quadrant and accumulator state of charge, SOC_p (see Eq. 2.14), where thresholds SOC_{p_1}, SOC_{p_2}, and SOC_{p_3} define the state-of-charge thresholds.

Figure 6.1 illustrates the control logic of the proposed ESR strategy. First, the operating quadrant, the accumulator state of charge, load pressure (P_L), accumulator pressure (P_{acc}), and output pressure P_{pprv} are evaluated (see Subsection 2.2 in Chapter 2). Based on these variables, the controller decides whether to store hydraulic energy, reutilize the stored energy, or remain in the neutral mode.

[1] S. Hosseinpour, Badhushan Chithravelpillai, Gustavo Koury Costa, and N. Sepehri, "Energy-Efficient Control of Electro-Hydrostatic Actuators: A Multi-Objective Optimization Approach,", 2026, to be submitted.

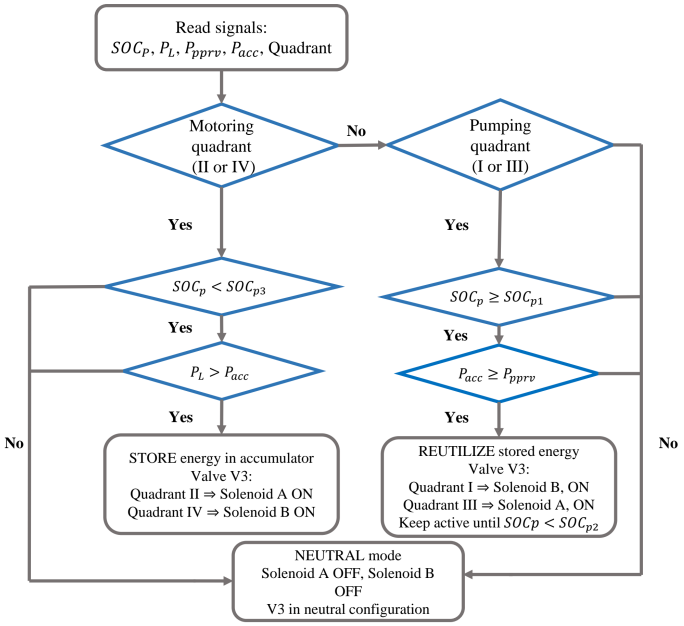


Figure 6.1: The control logic for the proposed energy storage and reutilization (ESR) strategy.

To illustrate how the storage and reutilization modes are activated across different ranges of the accumulator state of charge, Fig. 6.2 provides a compact visualization of the switching behavior of the proposed strategy.

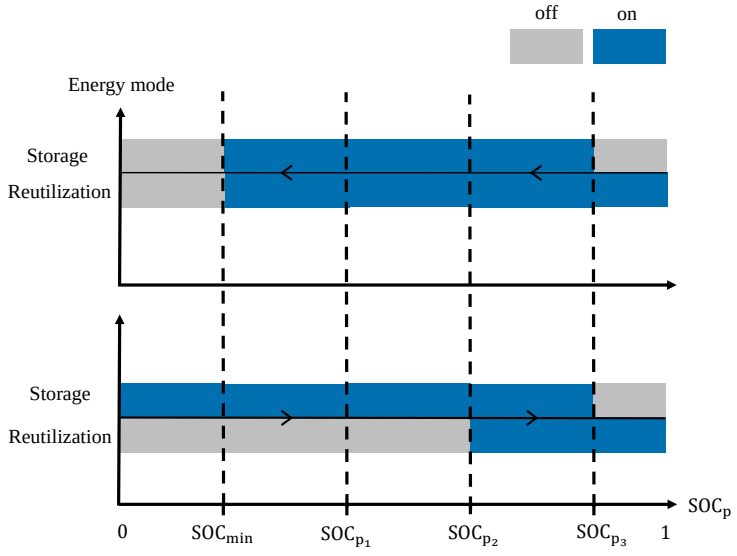


Figure 6.2: Activation map of storage and reutilization modes as a function of accumulator state of charge (SOC_p). Colored regions indicate storage or reutilization status (on/off), and dashed lines represent switching thresholds.

Based on the ESR control logic, the optimization variables consist of the three PID gains together with the ESR thresholds, forming the decision vector

$$\mathbf{x} = \begin{bmatrix} K_p & K_i & K_d & \mathrm{SOC}_{p_1} & \mathrm{SOC}_{p_2} & \mathrm{SOC}_{p_3} \end{bmatrix}^\top \in \mathbb{R}^6 \tag{6.2}$$

where:

(i) (K_p, K_i, K_d) are the proportional, integral, and derivative gains of the PID controller;

(ii) SOC_{p_1} is the *upper reutilization threshold*, which activates energy reutilization when the accumulator state of charge (SOC_p) exceeds this level;

(iii) SOC_{p_2} is the *lower reutilization threshold*, which deactivates energy reutilization, allowing the accumulator to discharge deeply;

(iv) SOC_{p_3} represents the *upper storage threshold*, which stops the energy storage process when SOC_p reaches this level.

The thresholds SOC_{p_1} and SOC_{p_2} implement a two-threshold hysteresis mechanism for energy reutilization. Specifically, energy reutilization is activated when the accumulator state of charge satisfies $\mathrm{SOC}_p \geq \mathrm{SOC}_{p_1}$ and remains active until $\mathrm{SOC}_p \leq \mathrm{SOC}_{p_2}$, after which it is deactivated and the accumulator is allowed to recharge. This hysteresis mechanism mitigates frequent switching and enables effective utilization of the stored energy.

Energy reutilization is controlled by the two-position valve V4, which is activated simultaneously with the float-centre directional valve V3. The directional valve V3 is actuated by two solenoids, A and B. The operating state of the solenoids depends on the load pressure (P_L), accumulator pressure (P_{acc}), output pressure of the electro-proportional pressure reducing valve (P_{pprv}), accumulator state of charge (SOC_p), and the operating quadrant.

Energy storage is permitted only in the motoring quadrants (II and IV), when $\mathrm{SOC}_p < \mathrm{SOC}_{p_3}$ and the load pressure exceeds the accumulator pressure ($P_L > P_{\mathrm{acc}}$). Under these conditions, solenoid A is activated in Quadrant II and solenoid B is activated in Quadrant IV, routing excess hydraulic energy into the accumulator.

Conversely, energy reutilization is enabled in the pumping quadrants (I and III), when $\mathrm{SOC}_p \geq \mathrm{SOC}_{p_1}$ and the accumulator pressure exceeds the output pressure of the EPPRV ($P_{\mathrm{acc}} > P_{\mathrm{pprv}}$). In this mode, solenoid B is activated in Quadrant I and solenoid A is activated in Quadrant III, allowing the stored hydraulic energy to be released and reused by the actuator.

When the state of charge, pressure, or operating quadrant conditions are not satisfied, both solenoids remain deactivated and the directional valve returns to its neutral configuration.

To ensure physically meaningful and logically consistent operation of the ESR unit, the following ordering constraint is enforced

$$\mathrm{SOC}_{\min} \leq \mathrm{SOC}_{p_2} \leq \mathrm{SOC}_{p_1} \leq \mathrm{SOC}_{p_3} \tag{6.3}$$

which guarantees that reutilization is initiated only after sufficient energy has been accumulated and is terminated only after substantial discharge. To ensure a minimum usable pressure level, a lower bound on the accumulator state of charge is imposed. The $\mathrm{SOC}_{\min} = 0.025$ is considered that represents the minimum admissible state of charge.

6.1.1 Multi-objective Optimization Problem Formulation

The optimization is performed over a six-dimensional decision vector $\mathbf{x} \in \mathbb{R}^6$, defined as

$$\mathbf{x} = \begin{bmatrix} K_p & K_i & K_d & \mathrm{SOC}_{p_1} & \mathrm{SOC}_{p_2} & \mathrm{SOC}_{p_3} \end{bmatrix}^\top$$

which includes the PID controller gains and the ESR threshold parameters. The decision variables are subject to bound constraints defining a hyper-rectangular domain

$$\mathcal{X}_b \triangleq \left\{ \mathbf{x} \in \mathbb{R}^6 \,\middle|\, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \right\} \tag{6.4}$$

with elementwise limits

$$\mathbf{l} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top \tag{6.5}$$

$$\mathbf{u} = \begin{bmatrix} 20 & 20 & 20 & 1 & 1 & 1 \end{bmatrix}^\top \tag{6.6}$$

The threshold ordering constraint in Eq. (6.3) can be expressed as a set of linear inequality constraints of the form

$$\mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{6.7}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ -0.025 \end{bmatrix} \tag{6.8}$$

The overall feasible set is therefore defined as

$$\mathcal{X} \triangleq \{ \mathbf{x} \in \mathcal{X}_b \mid \mathbf{A}\mathbf{x} \leq \mathbf{b} \} \tag{6.9}$$

Under these constraints, the multi-objective optimization problem is formulated as

$$\min_{\mathbf{x}\in\mathcal{X}} \mathbf{F}(\mathbf{x}) = \begin{bmatrix} J_{\text{error}}(\mathbf{x}) \\ J_{\text{energy}}(\mathbf{x}) \end{bmatrix} \tag{6.10}$$

where J_{error} denotes the tracking error metric and J_{energy} represents the energy consumption of the system. The tracking-error objective is defined as

$$J_{\text{error}}(\mathbf{x}) = \int_0^{T_f} |r(t) - x_p(t)|\, dt \tag{6.11}$$

where T_f is the total simulation time, $r(t)$ and $x_p(t)$ denote the reference signal and the actuator position, respectively. The energy-consumption objective is given by

$$J_{\text{energy}}(\mathbf{x}) = \int_0^{T_f} |\omega(t) T_e(t)|\, dt \tag{6.12}$$

where $\omega(t)$ and $T_e(t)$ denote the motor angular velocity and torque, respectively.

Because the objectives are inherently competing, no single solution can simultaneously minimize both tracking error and energy consumption. Compared with single-objective formulations based on weighted scalarization, the multi-objective optimization approach avoids dependence on user-defined weighting factors. In weighted scalarization methods, the outcome depends strongly on the choice of weights, which may bias the solution toward a particular performance preference. In contrast, the multi-objective optimization framework considers tracking performance and energy consumption as independent and equally important objectives.

Within this framework, optimality is defined in terms of Pareto dominance. Rather than yielding a single operating point, the optimization process produces a set of Pareto-optimal solutions, each representing a distinct trade-off between tracking accuracy and energy efficiency.

In this work, the optimization problem is solved using MATLAB's `gamultiobj` solver, which implements a controlled elitist genetic algorithm based on the NSGA-II framework [108]. This population-based multi-objective optimization method is well suited to problems involving competing objectives, as it evolves a population of candidate solutions to simultaneously explore multiple regions of the decision space and approximate the trade-off surface within a single optimization run. Accordingly, the outcome of the optimization is a Pareto-optimal set

$$\mathcal{P} = \{\mathbf{x} \in \mathcal{X} \mid \nexists \mathbf{y} \in \mathcal{X} \text{ such that } \mathbf{F}(\mathbf{y}) \prec \mathbf{F}(\mathbf{x})\} \tag{6.13}$$

where $\mathbf{F}(\cdot)$ denotes the vector of objective functions and $\prec$ represents Pareto dominance. A solution

$\mathbf{x}_1$ dominates $\mathbf{x}_2$ if

$$\begin{cases} F_i(\mathbf{x}_1) \leq F_i(\mathbf{x}_2), & \forall i \\ F_j(\mathbf{x}_1) < F_j(\mathbf{x}_2), & \exists j \end{cases} \tag{6.14}$$

The image of this set under the objective mapping, denoted by $\mathbf{F}(\mathcal{P})$, is referred to as the Pareto front. The `gamultiobj` solver returns a finite collection of nondominated solutions that approximates the true Pareto set

$$\left\{\mathbf{x}^{(k)}\right\}_{k=1}^{N_p} \subset \mathcal{P} \tag{6.15}$$

along with the corresponding objective values arranged in the matrix

$$\mathbf{F}_{\text{Pareto}} = \begin{bmatrix} J_{\text{error}}(\mathbf{x}^{(1)}) & J_{\text{energy}}(\mathbf{x}^{(1)}) \\ \vdots & \vdots \\ J_{\text{error}}(\mathbf{x}^{(N_p)}) & J_{\text{energy}}(\mathbf{x}^{(N_p)}) \end{bmatrix} \tag{6.16}$$

where N_p denotes the number of Pareto points. The `gamultiobj` algorithm evolves a population of candidate solutions across successive generations. Each individual represents a candidate decision vector $\mathbf{x} = \begin{bmatrix} K_p & K_i & K_d & \text{SOC}_{p_1} & \text{SOC}_{p_2} & \text{SOC}_{p_3} \end{bmatrix}^\top$, and its performance is evaluated using the objective vector $\mathbf{F}(\mathbf{x}) = [J_{\text{error}}, J_{\text{energy}}]^\top$.

To improve the initialization of the multi-objective genetic algorithm, a surrogate modelling method based on Gaussian processes (GP) is introduced to guide the selection of initial candidate solutions.

6.1.2 Gaussian Process–Based Initialization

The multi-objective evolutionary algorithms rely on repeated evaluations of computationally expensive objective functions. Applying them directly to the nonlinear EHA simulation model may result in slow convergence. To address this limitation, a GP surrogate modelling framework is integrated into the initialization stage of the genetic algorithm. In this method, each decision vector $\mathbf{x}$ is evaluated using the nonlinear EHA simulation to obtain the corresponding objective vector $\mathbf{F}(\mathbf{x})$. An initial set of true objective evaluations is used to train GP surrogate models. The resulting surrogate-informed candidates are then used to seed the initial population of the genetic algorithm prior to evolutionary refinement.

A feasible set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N_\text{F}}$ is generated by uniformly sampling the admissible parameter space while satisfying all bound and linear inequality constraints. In addition to uniformly sampled, a baseline point located at the center of the bounded search domain is included to ensure adequate interior coverage of the feasible region. Each design point is evaluated using the EHA simulation

model to obtain the objective vector

$$\mathbf{y}_i = \begin{bmatrix} J_{\text{error},i} \\ J_{\text{energy},i} \end{bmatrix}, \; i = 1, \ldots, N_F \tag{6.17}$$

Because the objective functions are strictly nonnegative, the logarithmic transformations are applied as

$$\tilde{J}_{\text{error},i} = \log(J_{\text{error},i} + \varepsilon_{\text{err}}) \tag{6.18}$$

$$\tilde{J}_{\text{energy},i} = \log(J_{\text{energy},i} + \varepsilon_{\text{eng}}) \tag{6.19}$$

where ε_{err} and ε_{eng} are small positive constants. Two independent GP models are trained in log-space, such that

$$\tilde{J}_{\text{error}}(\mathbf{x}) \sim \mathcal{GP}_{\text{err}}(\mu_{\text{err}}(\mathbf{x}), k_{\text{err}}(\mathbf{x}, \mathbf{x}')) \tag{6.20}$$

$$\tilde{J}_{\text{energy}}(\mathbf{x}) \sim \mathcal{GP}_{\text{eng}}\big(\mu_{\text{eng}}(\mathbf{x}), k_{\text{eng}}(\mathbf{x}, \mathbf{x}')\big) \tag{6.21}$$

with anisotropic squared exponential kernels $k_{\text{err}}(\cdot, \cdot)$ and $k_{\text{eng}}(\cdot, \cdot)$. Using the trained GP models, the objective functions at any candidate point $\mathbf{x}$ are represented as distributions

$$\tilde{J}_{\text{error}}(\mathbf{x}) \sim \mathcal{N}(\mu_{\text{err}}, \sigma^2_{\text{err}}) \tag{6.22}$$

$$\tilde{J}_{\text{energy}}(\mathbf{x}) \sim \mathcal{N}(\mu_{\text{eng}}, \sigma^2_{\text{eng}}) \tag{6.23}$$

A large feasible candidate set $\mathcal{X}_{\text{cand}} = \{\mathbf{x}_j\}_{j=1}^{N_{\text{cand}}}$ is then sampled from $\mathcal{X}$ without performing additional simulations. Predicted objective values are recovered in the original scale using the log-normal mean approximation

$$\hat{J}_{\text{error},j} = \exp(\mu_{\text{err},j} + \tfrac{1}{2}\sigma^2_{\text{err},j}) - \varepsilon_{\text{err}} \tag{6.24}$$

$$\hat{J}_{\text{energy},j} = \exp(\mu_{\text{eng},j} + \tfrac{1}{2}\sigma^2_{\text{eng},j}) - \varepsilon_{\text{eng}} \tag{6.25}$$

ensuring nonnegative estimates of both objectives. The predicted objective pairs

$$\hat{\mathbf{f}}_j = \begin{bmatrix} \hat{J}_{\text{error},j} \\ \hat{J}_{\text{energy},j} \end{bmatrix} \tag{6.26}$$

are filtered according to Pareto dominance, yielding a surrogate-based approximation of the

nondominated set

$$\mathcal{P}_{\mathrm{GP}} = \left\{ \mathbf{x}_j \in \mathcal{X}_{\mathrm{cand}} \;\middle|\; \nexists\, \mathbf{x}_k : \hat{\mathbf{f}}_k \prec \hat{\mathbf{f}}_j \right\} \tag{6.27}$$

To select a fixed number of seed solutions, a scalar ranking score is defined as

$$s_j = \frac{\hat{J}_{\mathrm{error},j} - \min \hat{J}_{\mathrm{error}}}{\max \hat{J}_{\mathrm{error}} - \min \hat{J}_{\mathrm{error}}} + \frac{\hat{J}_{\mathrm{energy},j} - \min \hat{J}_{\mathrm{energy}}}{\max \hat{J}_{\mathrm{energy}} - \min \hat{J}_{\mathrm{energy}}} \tag{6.28}$$

and the N_{seed} candidates with the smallest scores are chosen. If fewer than N_{seed} candidates are available from the surrogate Pareto set, the remaining positions are filled with the best-ranked candidates from the candidate pool based on the score $s(\mathbf{x})$. This set constitutes a GP-based initial population, which is supplied to the multi-objective genetic algorithm to initiate the final optimization stage. The key parameters used in the GP-based initialization and multi-objective genetic algorithm are summarized in Table 6.1.

The proposed GP-based initialization algorithm for the multi-objective optimization method is summarized as follows:

1. **Input:** Bounds $(\mathbf{l}, \mathbf{u})$, constraints $\mathbf{A}\mathbf{x} \leq \mathbf{b}$, sizes $N_{\mathrm{F}}, N_{\mathrm{cand}}, N_{\mathrm{gen}}, N_{\mathrm{seed}}$, and log offsets $\varepsilon_{\mathrm{err}}, \varepsilon_{\mathrm{eng}}$.
2. Sample feasible $\mathbf{X}_{\mathrm{F}} \subset \mathcal{X}$ and prepend the baseline point.
3. Evaluate $(J_{\mathrm{error},i}, J_{\mathrm{energy},i})$ for each $\mathbf{x}_i \in \mathbf{X}_{\mathrm{F}}$.
4. Apply logarithmic transformations: $\tilde{J}_{\mathrm{error},i} \leftarrow \log(J_{\mathrm{error},i} + \varepsilon_{\mathrm{err}})$, $\tilde{J}_{\mathrm{energy},i} \leftarrow \log(J_{\mathrm{energy},i} + \varepsilon_{\mathrm{eng}})$.
5. Fit two independent GP surrogate models in log-space: $\tilde{J}_{\mathrm{error}}(\mathbf{x}) \sim \mathcal{GP}_{\mathrm{err}}(\mu_{\mathrm{err}}, k_{\mathrm{err}})$, $\tilde{J}_{\mathrm{energy}}(\mathbf{x}) \sim \mathcal{GP}_{\mathrm{eng}}(\mu_{\mathrm{eng}}, k_{\mathrm{eng}})$.
6. Sample feasible candidate pool $\mathbf{X}_{\mathrm{cand}}$ with $|\mathbf{X}_{\mathrm{cand}}| = N_{\mathrm{cand}}$.
7. Predict log-space moments for $\mathbf{X}_{\mathrm{cand}}$: $(\mu_{\mathrm{err}}, \sigma_{\mathrm{err}}) \leftarrow \mathcal{GP}_{\mathrm{err}}(\mathbf{X}_{\mathrm{cand}})$, $(\mu_{\mathrm{eng}}, \sigma_{\mathrm{eng}}) \leftarrow \mathcal{GP}_{\mathrm{eng}}(\mathbf{X}_{\mathrm{cand}})$.
8. Recover predicted objectives in the original scale using the log-normal mean approximation: $\hat{J}_{\mathrm{error}} \leftarrow \exp(\mu_{\mathrm{err}} + \frac{1}{2}\sigma_{\mathrm{err}}^2) - \varepsilon_{\mathrm{err}}$, $\hat{J}_{\mathrm{energy}} \leftarrow \exp(\mu_{\mathrm{eng}} + \frac{1}{2}\sigma_{\mathrm{eng}}^2) - \varepsilon_{\mathrm{eng}}$.
9. Apply surrogate Pareto filtering to $(\hat{J}_{\mathrm{error}}, \hat{J}_{\mathrm{energy}})$ to obtain $\mathcal{P}_{\mathrm{GP}} \subset \mathbf{X}_{\mathrm{cand}}$.
10. Score and select seeds: $s(\mathbf{x}) = \mathrm{norm}(\hat{J}_{\mathrm{error}}) + \mathrm{norm}(\hat{J}_{\mathrm{energy}})$; $\mathbf{X}_{\mathrm{seed}} \leftarrow$ top N_{seed} points from $\mathcal{P}_{\mathrm{GP}}$ in ascending order of s.
11. **If** $|\mathbf{X}_{\mathrm{seed}}| < N_{\mathrm{seed}}$ **then**
    Fill the remaining slots with the best-ranked candidates from $\mathbf{X}_{\mathrm{cand}}$ according to $s(\mathbf{x})$.
12. Run `gamultiobj` initialized with $\mathbf{X}_{\mathrm{seed}}$; return $(\mathbf{X}_{\mathrm{Pareto}}, \mathbf{F}_{\mathrm{Pareto}})$.
13. **Output:** $(\mathbf{X}_{\mathrm{Pareto}}, \mathbf{F}_{\mathrm{Pareto}})$.

Table 6.1: Key parameters used in the GP-based initialization and multi-objective genetic algorithm.

Symbol	Value	Description
N_{F}	50 + baseline	Training samples used for GP fitting.
N_{cand}	20 000	Candidate pool evaluated by the GP models.
N_{seed}	80	GP-selected seed solutions for GA initialization.
N_{pop}	120	Population size of the genetic algorithm.
N_{gen}	120	Maximum number of GA generations.
ε_E	10^{-9}	Logarithmic offset for tracking-error objective.
ε_P	10^{-9}	Logarithmic offset for energy objective.
ρ_{Pareto}	0.7	Fraction of population maintained on the Pareto front.
ρ_{cross}	0.85	Crossover fraction.
Tol_f	10^{-4}	Function tolerance for convergence.
G_{stall}	40	Stall generation limit.
T_{max}	2700 s	Maximum optimization time.

The parameters listed in Table 6.1 play an important role in the performance of the multi-objective optimization algorithm. The population size, N_{pop}, determines the number of candidate solutions maintained in each generation, while the maximum number of generations, N_{gen}, defines the maximum number of the evolutionary search. The number of GP training samples, N_{F}, controls the amount of simulation data used to construct the surrogate models, whereas N_{cand} specifies the size of the candidate pool evaluated by the GP models during the seed-selection stage. The parameter N_{seed} denotes the number of surrogate solutions used to initialize the genetic algorithm. The Pareto fraction, ρ_{Pareto}, specifies the fraction of the population reserved for the best nondominated solutions in each generation, while the crossover fraction, ρ_{cross}, specifies the fraction of offspring generated by combining pairs of parent solutions. The logarithmic offsets, ε_E and ε_P, are introduced to avoid numerical difficulties associated with evaluating the logarithm of zero-valued objective functions. Finally, the function tolerance, Tol_f, the stall generation limit, G_{stall}, and the maximum runtime, T_{max}, serve as practical stopping criteria, terminating the optimization when convergence is achieved or when computational limits are reached. To maintain population diversity and reduce the risk of biasing the search toward regions selected by the GP surrogate model, $N_{\mathrm{seed}} < N_{\mathrm{pop}}$ is considered. The remaining individuals required to complete the initial population are generated automatically by `gamultiobj`.

6.1.3 Results

The effectiveness of the proposed energy optimization framework is evaluated by comparing the performance of the EHA system with and without the ESR unit. The comparison is performed using the obtained Pareto fronts to assess the performance of the ESR system. A sinusoidal reference input is considered to compare the energy performance. Figure 6.3 shows the position response $x_p(t)$ and the sinusoidal reference. The corresponding control input $u(t)$ applied to the servomotor

and velocity of the rod $\dot{x}_p(t)$ are shown in Figs. 6.4 and 6.5, respectively.

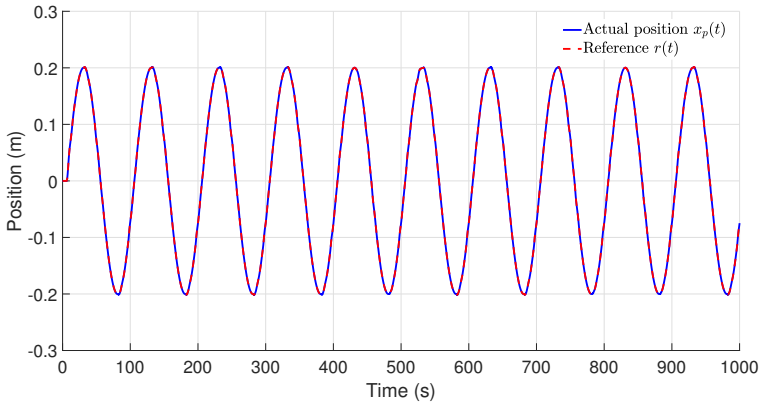


Figure 6.3: Actuator position response, x_p and reference signal r.

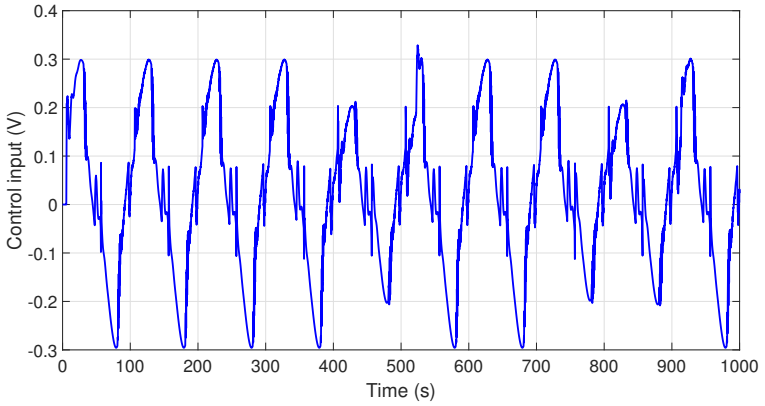


Figure 6.4: Control input, u.

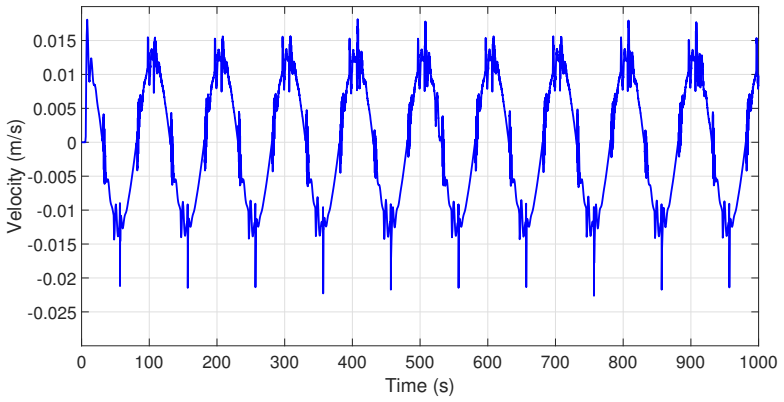
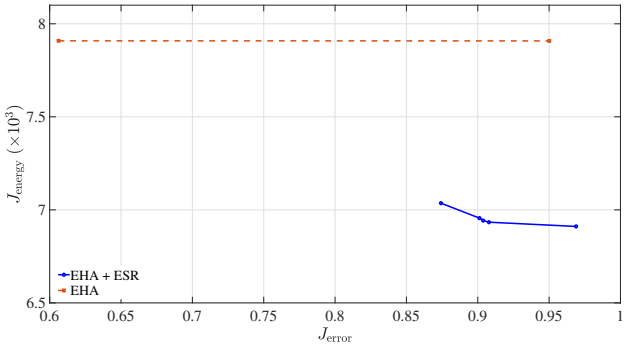


Figure 6.5: Actuator rod velocity, $\dot{x}_p$.

Two tracking error intervals are considered: a low-error interval ($J_{\text{error}} < 1$) and a moderate-error interval ($1 < J_{\text{error}} < 3$). These ranges are selected to examine system performance across different accuracy levels while maintaining a consistent basis for comparison. For the low-error regime ($J_{\text{error}} < 1$), as summarized in Table 6.2, a clear difference is observed in terms of energy efficiency. In this regime, the EHA system equipped with ESR unit achieves a minimum average energy consumption of 6.91 kW at $J_{\text{error}} = 0.9689$, whereas the EHA system requires approximately 7.91 kW at a comparable tracking error. This corresponds to a reduction in average energy consumption of 12.61%.

Table 6.2: Pareto-optimal solutions for the EHA system with and without an energy storage and reutilization (ESR) unit in the low-error regime ($J_{\text{error}} < 1$). Bold entries indicate the minimum energy achieved by each system within this regime.

System	J_{error}	J_{energy}
EHA+ESR	0.8742	7.0360×10^3
	0.9012	6.9555×10^3
	0.9038	6.9426×10^3
	0.9078	6.9337×10^3
	0.9689	$\mathbf{6.9109 \times 10^3}$
EHA	0.6062	7.9084×10^3
	0.9501	$\mathbf{7.9080 \times 10^3}$

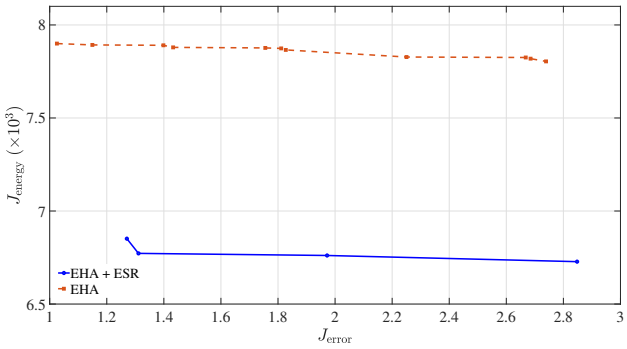


The moderate-error regime ($1 < J_{\text{error}} < 3$) reported in Table 6.3. Within this band, the EHA system equipped with ESR unit again exhibits a superior error–energy trade-off. Specifically, the minimum-energy solution obtained for the EHA equipped with the ESR unit requires 6.72 kW at $J_{\text{error}} = 2.8480$, whereas the corresponding minimum-energy solution for the baseline EHA without

the ESR unit consumes 7.80 kW at $J_{\text{error}} = 2.7392$. Despite operating within a similar error range, the ESR system achieves a energy reduction of 13.79% relative to the EHA baseline.

Table 6.3: Pareto-optimal solutions in the moderate-error regime ($1 < J_{\text{error}} < 3$). Bold entries indicate the minimum energy achieved by each system within this regime.

System	J_{error}	J_{energy}
EHA+ESR	1.2704	6.8513×10^3
	1.3111	6.7723×10^3
	1.9720	6.7610×10^3
	2.8480	$\mathbf{6.7278 \times 10^3}$
EHA	1.0253	7.8999×10^3
	1.1495	7.8926×10^3
	1.3985	7.8906×10^3
	1.4325	7.8794×10^3
	1.7555	7.8765×10^3
	1.8112	7.8739×10^3
	1.8273	7.8658×10^3
	2.2503	7.8276×10^3
	2.6681	7.8251×10^3
	2.6850	7.8188×10^3
	2.7392	$\mathbf{7.8041 \times 10^3}$



The controller parameters corresponding to the minimum-energy solutions within each tracking error range for the EHA systems with and without the ESR unit are summarized in Table 6.4. Overall, the comparison demonstrates that the incorporation of ESR with EHA systematically shifts the Pareto-optimal solutions toward lower energy consumption across both low- and moderate-error operating conditions.

Table 6.4: Controller parameters corresponding to the minimum-energy solutions within each tracking error range. The reported error corresponds to the solution achieving the minimum energy within the specified range.

Case	J_{error}	J_{energy}	$(K_p, K_i, K_d, \text{SOC}_{p_1}, \text{SOC}_{p_2}, \text{SOC}_{p_3})$
EHA+ESR	0.9689	6.91×10^3	(16.84, 17.59, 14.42, 0.37, 0.11, 0.85)
	2.8480	6.73×10^3	(7.79, 4.99, 13.74, 0.42, 0.11, 0.64)
EHA	0.9501	7.91×10^3	(13.65, 12.51, 8.12, –, –, –)
	1.4325	7.88×10^3	(17.22, 8.15, 10.18, –, –, –)

Due to the large number of Pareto-optimal solutions obtained, it is impractical to present the dynamic responses for all cases for the EHA and ESR systems. Therefore, for illustrative purposes, the responses of the EHA equipped with the ESR unit are presented for the Pareto-optimal solution corresponding to $J_{\text{error}} = 0.9689$ and $J_{\text{energy}} = 6.9109 \times 10^3$.

The storage and reutilization signals are depicted in Fig. 6.6, where the storage-mode indicator identifies intervals during which the accumulator is charged, and the regeneration indicator denotes periods of active energy reutilization. Figure 6.7 reports the value of $\text{SOC}_p(t)$, highlighting the charging and discharging dynamics of the ESR unit. It shows that the accumulator does not always need to be fully charged and then discharged to achieve optimal efficiency.

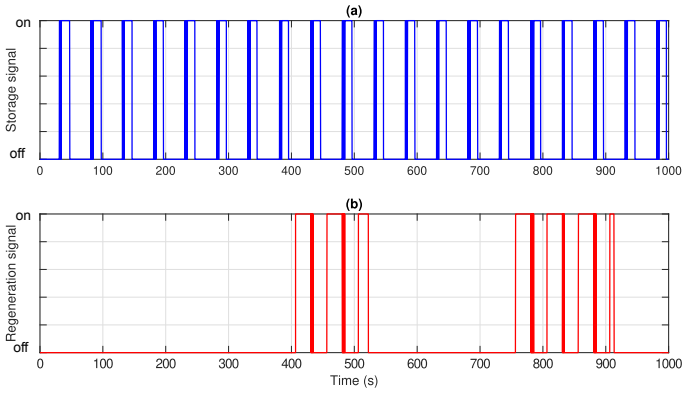


Figure 6.6: Energy mode indicators: (a) storage-mode activation; (b) regeneration activation.

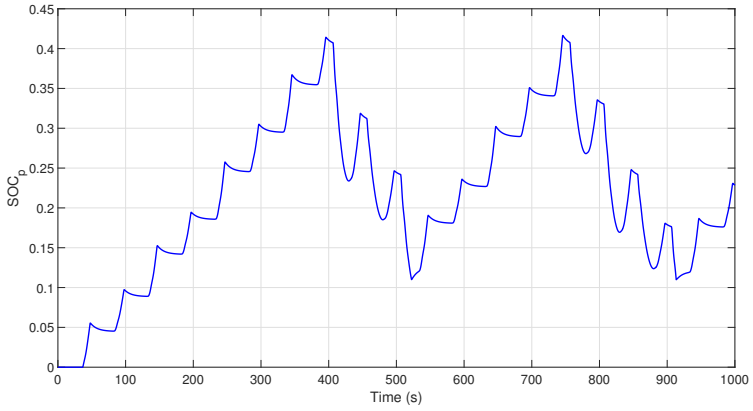


Figure 6.7: The value of the state of charge of accumulator SOC_p.

The pressure evolution within the ESR circuit is shown in Fig. 6.8, which illustrates the accumulator pressure $P_{\text{acc}}(t)$ and the electro-proportional pressure reducing valve (EPPRV) outlet pressure $P_{\text{pprv}}(t)$. The load pressure signal $P_L(t)$ is presented in Fig. 6.9.

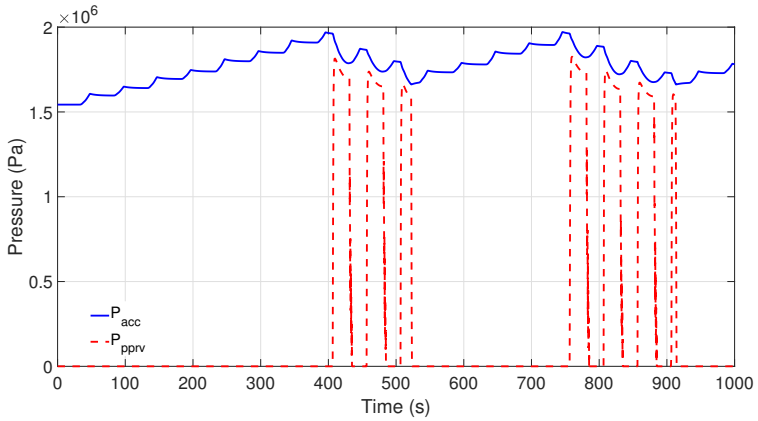


Figure 6.8: Accumulator pressure, P_{acc} and electro-proportional pressure reducing valve (EPPRV) outlet pressure, P_{pprv}.

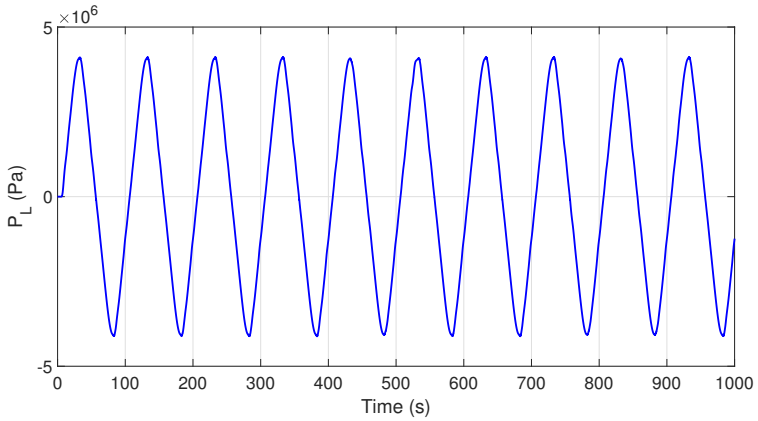


Figure 6.9: Load pressure, P_L.

The motor current $I(t)$ and angular velocity $\omega(t)$ are illustrated in Fig. 6.10. The corresponding torques are shown in Fig. 6.11, including the motor torque $T_e(t)$, the load torque $T_L(t)$, and the pump–motor torque $T_{\mathrm{PM}}(t)$.

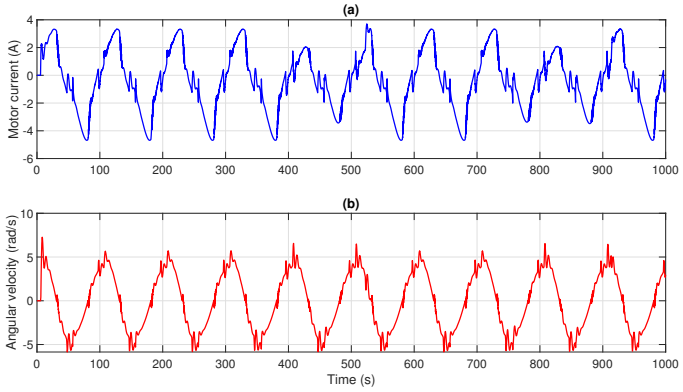


Figure 6.10: Servomotor response: (a) motor current I; (b) motor angular velocity ω.

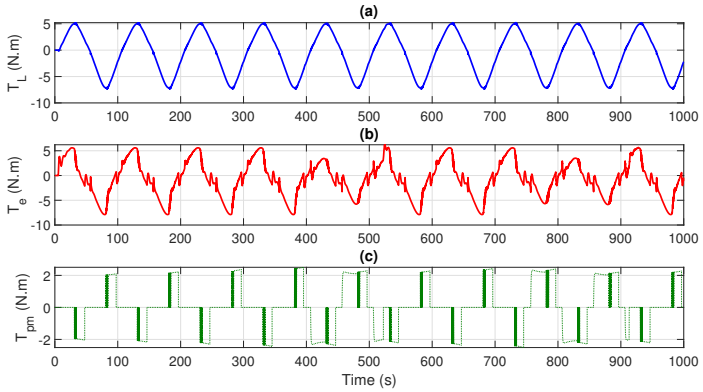


Figure 6.11: Torque responses: (a) servomotor torque, T_e; (b) load torque, T_L; (c) pump–motor torque, T_{PM_2}.

Figure 6.12 presents the corresponding chamber flows $Q_{ac}(t)$ and $Q_{bc}(t)$, the accumulator flow rate $Q_{acc}(t)$, and the pump flow rate $Q_{\mathrm{PM2}}(t)$. Finally, the motor power metric is reported in Fig. 6.13.

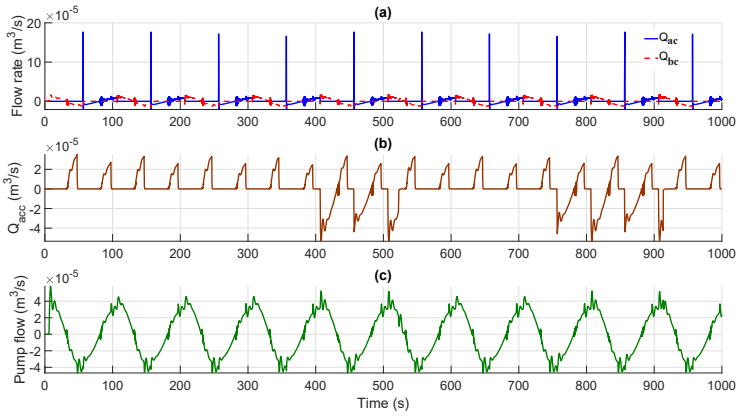


Figure 6.12: Flow responses: (a) hydraulic chamber flows, including cap-side flow Q_{ac} and rod-side flow, Q_{bc}; (b) accumulator flow rate, Q_{acc}; (c) pump flow rate, Q_{PM_2}.

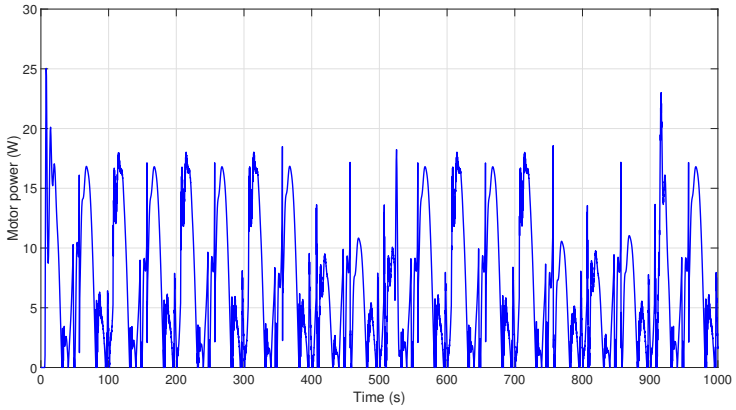


Figure 6.13: Instantaneous servomotor power.

6.2 Summary

This chapter presented a new energy storage and reutilization (ESR) control logic for energy optimization of the electro-hydrostatic actuator (EHA) equipped with an the ESR unit using a multi-objective optimization framework. By jointly tuning the controller gains and ESR decision variables, the proposed method considers the interaction between tracking performance and energy efficiency within an optimization problem. A Gaussian process (GP)-based initialization strategy

was also proposed to identify promising regions of the search space and generate initial solutions for the NSGA-II algorithm implemented through MATLAB's `gamultiobj` solver.

Simulation results based on Pareto-optimal solutions show that the EHA system with the ESR unit achieves reductions in energy consumption relative to the conventional EHA. For solutions with $J_{\text{error}} < 1$, the EHA system equipped with the ESR achieves an energy reduction of approximately 12.61% compared with the conventional EHA system. When moderate tracking errors are considered ($1 < J_{\text{error}} < 3$), the energy reduction increases to approximately 13.79%.

Overall, these results demonstrate that the proposed energy optimization method effectively reduces energy consumption while maintaining the desired tracking performance.

Chapter 7

Concluding Remarks

7.1 Thesis Conclusion

This thesis addressed the challenges of fault detection, control, and energy optimization in electro-hydrostatic actuator (EHA) systems using interpretable Bayesian, machine-learning, and optimization-based methodologies.

For fault detection, the thesis showed that internal leakage can be identified even with limited data. A Bayesian classification method was developed to distinguish between healthy and faulty conditions, achieving a classification accuracy of 93.75%. In addition, a sparse identification of nonlinear dynamics (SINDy) approach was employed to extract interpretable system models directly from measurement data. To improve model performance, Bayesian optimization was used to tune the SINDy hyperparameters. The Bayesian optimized SINDy framework increased the classification accuracy from 91.11% to 95.56%. It demonstrates the effectiveness of Bayesian optimization for enhancing fault-detection performance.

For control design applications, both model-based and model-free learning methods were investigated. The model-based approach provides a framework for tuning the controller parameters based on time and frequency domain specifications. The proposed model-free method enables the controller parameters to adjust online. To achieve this, first, a reinforcement learning method was used in an offline phase to explore the controller parameter space. Next, a Gaussian process (GP) model was trained to represent the relationship between controller gains and system performance. The GP model provided both predictions and uncertainty estimates, enabling efficient selection of controller parameters through a Bayesian strategy. This approach enabled the controller to adapt to changes in system dynamics.

For energy optimization, the thesis proposed a multi-objective optimization method that simultaneously considers tracking performance and energy consumption. A GP-based initialization stage was used in the multi-objective optimization instead of using randomly generated initial points.

The resulting Pareto-optimal solutions provide interpretable trade-offs between tracking error and efficiency, allowing selection based on specific requirements. This is particularly useful for industrial applications, where a set of candidate solutions is often preferred over a single solution.

In summary, the presented methods provide powerful and interpretable frameworks for addressing multiple challenges in EHA systems.

7.2 Contributions Made in the Thesis

The main contributions of this thesis can be summarized as follows:

1. **Bayesian Fault Detection using Polyscale Complexity Measures:** A novel fault detection framework was developed for EHA systems based on the integration of polyscale complexity measures and Bayesian classification. The variance fractal dimension (VFD) was employed as an effective feature extraction tool to capture the underlying complexity of pressure signals under different operating conditions.

2. **Fault Detection under Limited Data:** A novel fault detection approach was developed that combines SINDy-based modeling with Bayesian optimization . This method enables accurate identification of internal leakage using limited and noisy data, while maintaining interpretability of the learned models.

3. **Model-Based Control Design in the Frequency Domain:** A model-based framework is developed for tuning controller parameters based on time- and frequency-domain specifications. Unlike conventional optimization approaches, the proposed method enables a unified design to achieve both desired transient response and robustness characteristics.

4. **Model-Free Adaptive Control using GP Learning:** A model-free learning framework was proposed for online controller tuning. By combining reinforcement learning with a GP model, the method achieves efficient adaptation to changing system dynamics.

5. **Energy-Efficient Multi-Objective Optimization:** A new energy storage and reutilization (ESR) control logic for energy optimization was developed using a multi-objective optimization framework for the EHA system equipped with the ESR system to balance tracking performance and energy consumption.

7.3 Future Work

While the proposed frameworks provided promising results in addressing various challenges in EHA systems, their application is not limited to hydraulic systems. Therefore, several aspects remain

open for further investigation.

1. **Extension to Other Fault Types:** This thesis focused on internal leakage faults. Future research can extend the framework to detect and diagnose other types of faults, such as external faults.

2. **Integration with Deep Learning:** While this work emphasized interpretable and data-efficient models, combining Bayesian methods with deep learning architectures could improve performance in scenarios with larger datasets or more complex dynamics.

3. **Sequential Bayesian Updating for Online Learning:** Future research could extend the Bayesian classifier to update its parameter distributions as new labeled experimental data become available, using the current posterior as the prior for subsequent updates. This could support continued learning under additional operating conditions without retraining the model from scratch.

4. **Reinforcement Learning for Full Control Policy:** The current approach uses reinforcement learning (RL) for controller parameter tuning using the augmented random search. Future research could explore RL methods such as Twin Delayed Deep Deterministic policy gradient and Deep Deterministic Policy Gradient that directly learn control policies.

5. **Online Implementation and Computational Efficiency:** Improving the computational efficiency of Bayesian optimization and GP models is essential for real-time applications. Techniques such as sparse GPs or other optimization methods could be investigated.

6. **Reinforcement Learning for Energy Management and Control:** A potential direction for future research is to replace the current multi-objective optimization framework for the EHA integrated with the ESR unit with an RL-based approach. Instead of identifying controller gains offline, an RL agent could learn control and energy management policies through interaction with the system or its simulation environment. Such an approach would allow the controller to make decisions online.

References

[1] A. Parr, *Hydraulics and Pneumatics: A Technician's and Engineer's Guide*, Butterworth-Heinemann, 1991, ISBN-13: 978-0080966748.

[2] N. Niksefat and N. Sepehri, "Design and experimental evaluation of a robust force controller for an electro-hydraulic actuator via quantitative feedback theory," *Control Engineering Practice*, vol. 8, no. 12, pp. 1335–1345, 2000, doi: 10.1016/S0967-0661(00)00075-7.

[3] B. Heinrichs, N. Sepehri, and A. B. Thornton-Trump, "Position-based impedance control of an industrial hydraulic," *IEEE Control Systems Magazine*, vol. 17, no. 1, pp. 46–52, 1997, doi: 10.1109/37.569715.

[4] G. K. Costa and N. Sepehri, "Four-quadrant analysis and system design for single-rod hydrostatic actuators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 141, no. 2, p. 021011, 2019, doi: 10.1115/1.4041382.

[5] G. K. Costa and N. Sepehri, "Logic-controlled flow compensation circuit for operating single-rod hydrostatic actuators," U.S. Patent 11,060,533, Jul. 13, 2021.

[6] B. Li, Y. Liu, C. Tan, Q. Qin, and Y. Lu, "Review on electro-hydrostatic actuator: system configurations, design methods and control technologies," *International Journal of Mechatronics and Manufacturing Systems*, vol. 13, no. 4, pp. 323–346, 2020, doi: 10.1504/IJMMS.2020.112356.

[7] T. Wiens, "An efficient, compact and low-cost Dual Cylinder Hydrostatic Actuator (DCHA)", International Journal of Fluid Power, vol. 19, no. 2, 80–90, 2018, doi:10.1080/14399776.2018.1437321.

[8] T. Agostini,V. D. Negri, T. Minav, and M. Pietola, "Effect of Energy Recovery on Efficiency in Electro-Hydrostatic Closed System for Differential Actuator", *Actuators*, vol. 9, no.1, 2020, doi:10.3390/act9010012.

[9] Pexels, "Free Stock Photos and Videos." [Online]. Available: https://www.pexels.com. Accessed: Jun. 3, 2026.

[10] B. Chithravelpillai, *Energy Storage and Reutilization for a Single-Rod Electro-Hydrostatic Actuator System*, M.Sc. thesis, Dept. of Mechanical Engineering, University of Manitoba, Winnipeg, MB, Canada, 2022.

[11] B. Chithravelpillai, G. K. Costa, Z. Tomas, and N. Sepehri, "Energy storage and reutilization for a single-rod electro-hydrostatic actuator," U.S. Patent Application 18/412,851, Sep. 19, 2024.

[12] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, 2012, ISBN-13: 978-0262018029.

[13] S. Särkkä, *Bayesian Filtering and Smoothing*, Cambridge University Press, 2013, ISBN-13: 978-1107431997.

[14] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, 2005, ISBN-13: 978-0262201629.

[15] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian Data Analysis*, 3rd ed. Chapman and Hall/CRC, 2013, ISBN-13: 978-1439840955.

[16] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016, doi: 10.1109/JPROC.2015.2494218.

[17] P. I. Frazier, "A Tutorial on Bayesian Optimization," arXiv:1807.02811, 2018.

[18] S. Hosseinpour, W. Kinsner, N. Sepehri "A Machine Learning Model of Electro-Hydrostatic Actuators With the Low-Data Limit and Its Application to Fault Detection, *IEEE Access*," vol. 13, pp. 205685-205696, 2025, doi: 10.1109/ACCESS.2025.3639515.

[19] R. McElreath, *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*, 2nd ed. CRC Press, 2015. ISBN: 978-1482253443.

[20] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006, ISBN: 9780262182539.

[21] W. Chen, H. G. Sun, and X. Li, *Fractional Derivative Modeling in Mechanics and Engineering*. Singapore: Springer, 2022, ISBN: 978-981-16-8802-7.

[22] K. B. Oldham and J. Spanier, *The Fractional Calculus: Theory and Applications of Differentiation and Integration to Arbitrary Order*. Mineola, NY, USA: Dover Publications, 2006, ISBN-13: 978-0486450018.

[23] F. B. Tatom, "The relationship between fractional calculus and fractals," *Fractals*, vol. 3, no. 1, pp. 217–229, 1995.

[24] S. Brooks, A. Gelman, G. Jones, and X. Meng, *Handbook of Markov Chain Monte Carlo*. CRC Press, 2011, ISBN-13: 978-1420079418.

[25] Z. M. Hussain, A. Z. Sadik, and P. O'Shea, *Digital Signal Processing: An Introduction with MATLAB and Applications*. New York, NY, USA: Springer, 2011, ISBN-13: 978-3642155901.

[26] W. Kinsner, "A robust poly-scale length complexity measure for stochastic self-affine processes," in *Proc. IEEE Int. Conf. Cognitive Informatics and Cognitive Computing (ICCI*CC'21)*, Banff, AB, Canada, Oct. 29–30, 2021, doi: 10.1109/ICCICC53683.2021.9811308.

[27] W. Kinsner, "Batch and real-time computation of a fractal dimension based on variance of a time series," Technical Report DEL94-6, Dept. Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB, Canada, June 1994.

[28] W. Kinsner, "A robust variance complexity measure for stochastic self-affine processes," in *Proc. 18th IEEE Int. Conf. Cognitive Informatics and Cognitive Computing (ICCI*CC19)*, Milan, Italy, July 23–25, 2019, pp. 75–82, doi: 10.1109/ICCICC46617.2019.9146065.

[29] W. Kinsner, "Fractal dimensions: Morphological, entropy, spectrum, and variance classes," Technical Report, University of Manitoba, Winnipeg, MB, Canada, 1994.

[30] W. Grieder and W. Kinsner, "Speech segmentation by variance fractal dimension," in *Proc. Canadian Conf. Electrical and Computer Engineering (CCECE)*, Halifax, NS, Canada, Sept. 1994, pp. 481–485, doi: 10.1109/CCECE.1994.405793.

[31] W. Kinsner, "It's time for multiscale analysis and synthesis in cognitive systems," in *Proc. IEEE 10th Int. Conf. Cognitive Informatics and Cognitive Computing (ICCI-CC'11)*, Banff, AB, Canada, Aug. 2011, pp. 7–10, doi: 10.1109/COGINF.2011.6016116.

[32] W. Kinsner and H. Zhang, "Multifractal analysis and feature extraction of DNA sequences," in *Proc. 8th IEEE Int. Conf. Cognitive Informatics*, Hong Kong, China, June 2009, pp. 29–37, doi: 10.1109/COGINF.2009.5250696.

[33] M. Ghanbari and W. Kinsner, "Detecting DDoS attacks using polyscale analysis and deep learning," *International Journal of Cognitive Informatics and Natural Intelligence*, vol. 14, no. 1, pp. 17–34, 2020, doi: 10.4018/IJCINI.2020010102.

[34] S. Hosseinpour, W. Kinsner, and N. Sepehri, "Bayesian Optimized Neural Network for Fault Detection in Electro-Hydrostatic Actuators," *Journal of Machine Intelligence and Data Science (JMIDS)*, vol. 5, no. 1, pp. 144–151, 2024, doi: 10.11159/jmids.2024.016.

[35] S. aus der Wiesche, "Experimental investigation techniques for non-ideal compressible fluid dynamics," *International Journal of Turbomachinery, Propulsion and Power*, vol. 8, no. 2, 2023, doi: 10.3390/ijtpp8020011.

[36] K. H. Ang, G. Chong, and Y. Li, "PID control system analysis, design, and technology," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559–576, 2005, doi: 10.1109/TCST.2005.847331.

[37] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *Transactions of the ASME*, vol. 64, no. 8, pp. 759–765, 1942, doi: 10.1115/1.4019264.

[38] S. Skogestad, "Simple analytic rules for model reduction and PID controller tuning," *Journal of Process Control*, vol. 13, no. 4, pp. 291–309, 2003, doi: 10.1016/S0959-1524(02)00062-8.

[39] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., Springer, 2006, ISBN-13: 978-0387303031.

[40] M. R. H. Mojumder and N. K. Roy, "Review of Meta-Heuristic Optimization Algorithms to Tune the PID Controller Parameters for Automatic Voltage Regulator," arXiv preprint arXiv:2409.00538, 2024.

[41] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley, 2001. ISBN: 978-0-471-87339-6.

[42] R. Borase, D. K. Maghade, S. Y. Sondkar, and S. Pawar, "A review of PID control, tuning methods and applications," *International Journal of Dynamics and Control*, vol. 9, no. 2, pp. 818–827, 2020, doi: 10.1007/s40435-020-00665-4.

[43] S. Hosseinpour, W. Kinsner, and N. Sepehri, "A Frequency-Domain Optimization of PID Controllers," in *Proceedings of the IEEE 22nd International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC)*, Stanford University, Palo Alto, CA, USA, Aug. 19–21, 2023.

[44] C. A. Monje, B. M. Vinagre, V. Feliu, and Y. Q. Chen, "Tuning and auto-tuning of fractional order controllers for industry applications," *Control Engineering Practice*, vol. 16, no. 7, pp. 798–812, 2008, doi: 10.1016/j.conengprac.2007.08.006.

[45] K. J. Åström and T. Hägglund, *PID Controllers: Theory, Design, and Tuning*, ISA: The Instrumentation, Systems, and Automation Society, 1995, ISBN-13: 9781556175169.

[46] X.-W. Liu, "Global convergence on an active set SQP for inequality constrained optimization," *Journal of Computational and Applied Mathematics*, vol. 180, no. 1, pp. 201–211, 2005, doi: 10.1016/j.cam.2004.10.012.

[47] S. Hosseinpour, A. Nazemi, and E. Tohidi, "Müntz–Legendre spectral collocation method for solving delay fractional optimal control problems," *Journal of Computational and Applied Mathematics*, vol. 351, pp. 344–363, 2019, doi: 10.1016/j.cam.2018.10.058.

[48] H.-Z. Tan and N. Sepehri, "Parametric fault diagnosis for electrohydraulic cylinder drive units," *IEEE Transactions on Industrial Electronics*, vol. 49, no. 1, pp. 96–106, 2002, doi: 10.1109/41.982253.

[49] J. Yan, H. Zhu, X. Yang, Y. Cao, and L. Shao, "Research on fault diagnosis of hydraulic pump using convolutional neural network," *Journal of Vibroengineering*, vol. 18, no. 8, pp. 5141–5152, 2016, doi: 10.21595/jve.2016.16956.

[50] J. Z. Shi, F. Gu, B. Lennox, and A. Ball, "The development of an adaptive threshold for model-based fault detection of a nonlinear electrohydraulic system," *Control Engineering Practice*, vol. 13, no. 11, pp. 1357–1367, 2005, doi: 10.1016/j.conengprac.2004.11.014.

[51] L. Zhufeng, Q. Lvjun, W. Xiaodong, J. Wen, and W. Caixia, "Research on Fault Diagnosis Method of Electro-Hydrostatic Actuator," *Shock and Vibration*, pp. 1–9, 2021, doi: 10.1155/2021/6688420.

[52] K. Zhu, J. Ying, H. Fu, and X. Xin, "Metal-Based Additive Manufacturing Condition Monitoring: A Review on Machine Learning Based Approaches," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 1, pp. 509–520, 2017, doi: 10.1109/TMECH.2016.2620987.

[53] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2009, ISBN-13: 978-0387848570.

[54] W. Bolstad and J. Curran, *Introduction to Bayesian Statistics*, 3rd ed. Wiley, 2017, ISBN-13: 978-0470141151.

[55] S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*. Academic Press, 1st ed., 2015, ISBN:978-0128015223.

[56] B. Wang, Q. Jiang, and X. Yan, "Fault detection and identification using a Kullback–Leibler divergence based multi-block principal component analysis and Bayesian inference," *Korean Journal of Chemical Engineering*, vol. 31, no. 6, pp. 930–943, 2014, doi: 10.1007/s11814-013-0295-1.

[57] A. Maged and M. Xie, "Uncertainty utilization in fault detection using Bayesian deep learning," *Journal of Manufacturing Systems*, vol. 64, pp. 316–329, 2022.

[58] A. O. Salau and S. Jain, "Feature Extraction: A Survey of the Types, Techniques, Applications," *International Conference on Signal Processing and Communication (ICSC)*, Noida, India, pp. 158–164, 2019, doi: 10.1109/ICSC45622.2019.8938371.

[59] L. Wang, S. Ma, and Q. Han, "Enhanced Sparse Low-Rank Representation via Nonconvex Regularization for Rotating Machinery Early Fault Feature Extraction," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 5, pp. 3570–3578, 2022, doi: 10.1109/TMECH.2021.3135284.

[60] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2nd ed., 2011, ISBN: 978-0387310732.

[61] M. S. Nixon and A. S. Aguado, *Feature Extraction and Image Processing*. Elsevier, 2nd ed., 2007, ISBN: 978-0750650786.

[62] M. Ghanbari, W. Kinsner, and N. Sepehri, "Adaptive Regularization in ANN for Condition Monitoring and Fault Detection in Heavy-duty Hydraulic Machines," in *Proc. IEEE 21st Int. Conf. on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, Toronto, ON, Canada, Dec. 2022, doi: 10.1109/ICCI*CC57084.2022.10101651.

[63] S. Hosseinpour, W. Kinsner, and N. Sepehri, "Monoscale and Polyscale Analysis of Physical Signals for Compressive Detection of Relevant Complexity Features," *IEEE 21st Int. Conf. on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, pp. 1–7, Dec. 2022, doi: 10.1109/ICCI*CC57084.2022.10101562.

[64] S. Hosseinpour, W. Kinsner, and N. Sepehri, "Precision and accuracy of length and variance fractal dimensions computed from fractional self-affine signals," in *Proc. 8th World Congress on Electrical Engineering and Computer Systems and Science (EECSS)*, Prague, Czech Republic, July 2022, doi: 10.11159/mhmi22.109.

[65] A. Maddahi, W. Kinsner, and N. Sepehri, "Internal Leakage Detection in Electrohydrostatic Actuators Using Multiscale Analysis of Experimental Data," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 12, pp. 1–14, 2016, doi: 10.1109/TIM.2016.2608446.

[66] M. Ghanbari, W. Kinsner, and N. Sepehri, "Detection of Faults in Electro-Hydrostatic Actuators Using Feature Extraction Methods and an Artificial Neural Network," *IEEE World AI IoT Congress (AIIoT)*, pp. 01–07, 2022, doi: 10.1109/AIIoT54504.2022.9817236.

[67] S. V. Vaseghi, *Advanced Signal Processing and Noise Reduction*. John Wiley & Sons, 2nd ed., 2000, ISBN: 978-0471626923.

[68] A. Akhoondi Asadi, S. Bagheri, A. Imam, E. Jalayeri, W. Kinsner, and N. Sepehri, "A data acquisition system based on Raspberry Pi: Design, construction, and evaluation," in *Proc. IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conf. (IEMCON)*, Vancouver, BC, Canada, pp. 1–5, 2016, doi: 10.1109/IEMCON38388.2016.

[69] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," *Communications in Statistics: Theory and Methods*, vol. 6, no. 1, pp. 13-827, 1977, doi: 10.1080/03610927708827533.

[70] E. T. Jaynes, "On the rationale of maximum-entropy methods," *Proceedings of the IEEE*, vol. 70, no. 9, pp. 939–952, 1982, doi: 10.1109/PROC.1982.12425.

[71] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge Univ. Press, 2012, ISBN: 978-0521518147.

[72] D. Gamerman and H. F. Lopes, *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Chapman & Hall/CRC, 2nd ed., 2006, ISBN: 978-1584885870.

[73] M. C. Cowles, *Applied Bayesian Statistics with R and OpenBUGS Examples*. Springer, 2013, ISBN: 978-1461456957.

[74] J. C. Sprott, *Chaos and Time-Series Analysis*. Oxford Univ. Press, 2003, ISBN: 978-0198508403.

[75] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3rd ed., 2011, ISBN: 978-9380931913.

[76] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[77] Y. Zeng, G. Shengtao, F. Lingqing, and B. Boagen, "Modeling and simulation of hydraulic system for the multi-DOF hydraulic arm," in *2015 International Conference on Fluid Power and Mechatronics (FPM)*, Harbin, China, Aug. 2015, doi: 10.1109/FPM.2015.7337271.

[78] L. Khoi, V. Jatin, A. Mayekar, S. Agusavivas, C. Keplinger, M. Rentschler, and N. Correll, "Electro-Hydraulic Rolling Soft Wheel: Design, Hybrid Dynamic Modeling, and Model Predictive Control," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3044–3063, 2022, doi: 10.1109/TRO.2022.3167438.

[79] X. Jingping, S. Hongsheng, L. Wang, and H. Liu, "Adaptive neural network sliding mode control for serially connected hydraulic cylinders of a heavy-duty hydraulic manipulator," *Journal of Mechanical Science and Technology*, vol. 37, pp. 3763–3775, 2023, doi: 10.1007/s12206-023-0640-1.

[80] H. Zhonghai, H. Ma, J. Gao, S. Yu, and G. Geng, "Physics-Informed Machine Learning for Degradation Modeling of an Electro-Hydrostatic Actuator System," *Reliability Engineering & System Safety*, vol. 229, 2023, doi: 10.1016/j.ress.2022.108898.

[81] Y. Bai, L. Xiu, and M. Zhen, "Modeling Neural Networks Training Process with Markov Decision Process," in *2021 2nd International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, Nov. 2021, doi: 10.1109/ICAICE54393.2021.00102.

[82] J. Urban, N. Kutz, W. Brunton, and S. Brunton, "Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control," *Proceedings of the Royal Society A*, vol. 478, 2022, doi: 10.1098/rspa.2021.0904.

[83] S. Brunton, J. Proctor, and J. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016, doi: 10.1073/pnas.1517384113.

[84] E. Kaiser, J. Kutz, and S. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," *Proceedings of the Royal Society A*, 2018, doi: 10.1098/rspa.2018.0335.

[85] J. C. Loiseau, B. Noack, and S. Brunton, "Sparse reduced-order modeling: Sensor-based dynamics to full-state estimation," *Journal of Fluid Mechanics*, vol. 844, pp. 459–490, 2018, doi: 10.1017/jfm.2018.147.

[86] J. Wang, J. Moreira, Y. Cao, and R. B. Gopaluni, "Neural network and sparse identification of nonlinear dynamics integrated algorithm for digital twin identification," in *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 6921–6926, 2023, doi: 10.1016/j.ifacol.2023.10.503.

[87] B. M. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. N. Kutz, and S. L. Brunton, "PySINDy: A Python package for the sparse identification of nonlinear dynamics from data," arXiv:2004.08424, 2020, doi: 10.48550/arXiv.2004.08424.

[88] T. Head, M. Kumar, H. Nahrstaedt, G. Louppe, and I. Shcherbatyi, "scikit-optimize/scikit-optimize," *Zenodo*, 2021, doi: 10.5281/zenodo.5565057.

[89] S. Hosseinpour, W. Kinsner, S. Muthukumarana, and N. Sepehri, "Fault Detection in an Electro-Hydrostatic Actuator Using Polyscale Complexity Measures and Bayesian Classification," *IEEE Open Journal of Instrumentation and Measurement*, vol. 3, 2024, doi: 10.1109/OJIM.2024.3487237.

[90] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, arXiv:1206.2944v2, 2012, doi: 10.48550/arXiv.1206.2944.

[91] V. Nguyen, "Bayesian Optimization for Accelerating Hyper-Parameter Tuning," in *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, June 2019, Sardinia, Italy, doi: 10.1109/AIKE.2019.00060.

[92] S. Lahmiri, "Integrating convolutional neural networks, kNN, and Bayesian optimization for efficient diagnosis of Alzheimer's disease in magnetic resonance images," *Biomedical Signal Processing and Control*, vol. 80, 2023, doi: 10.1016/j.bspc.2022.104375.

[93] M. Abdelkader Bashery Abbass, "Comprehensive framework based on Bayesian optimization and skip connections artificial neural networks to predict buildings energy performance," *Journal of Building Engineering*, vol. 77, 2023, doi: 10.1016/j.jobe.2023.107523.

[94] A. Mirbegi Moghaddam, "Fault Detection and Fault-tolerant Control of Single-rod Electrohydrostatic Actuated System," M.Sc. Thesis, University of Manitoba, Winnipeg, MB, Canada, Department of Mechanical Engineering, 2021.

[95] J.-J. Slotine and W. Li, *Applied Nonlinear Control*, Prentice Hall, 1990, ISBN-13 : {978-0130408907}

[96] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," arXiv:1802.09477, 2018.

[97] G. Bujgoi and D. Sendrescu, "Tuning of PID controllers using reinforcement learning for nonlinear system control," *Processes*, vol. 13, no. 3, 2025, doi: 10.3390/pr13030735.

[98] X. Hao, Z. Xin, W. Huang, S. Wan, G. Qiu, T. Wang, and Z. Wang, "Deep reinforcement learning enhanced PID control for hydraulic servo systems in injection molding machines," *Scientific Reports*, no. 23005, 2025, doi: 10.1038/s41598-025-05904-2.

[99] M. A. Chowdhury, S. S. S. Al-Wahaibi, and Q. Lu, "Entropy-maximizing TD3-based reinforcement learning for adaptive PID control of dynamical systems," *Computers & Chemical Engineering*, vol. 178, no. 108393, 2023, doi: 10.1016/j.compchemeng.2023.108393.

[100] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, pp. 1437–1480, 2015.

[101] H. Mania, A. Guy, and B. Recht, "Simple random search provides a competitive approach to reinforcement learning," *Advances in Neural Information Processing Systems*, arXiv:1803.07055, 2018.

[102] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty," *Annals of Mathematics and Artificial Intelligence*, vol. 76, pp. 5–23, 2016, doi: 10.1007/s10472-015-9463-9.

[103] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *Journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985, doi: 10.1523/JNEUROSCI.05-07-01688.1985.

[104] G. K. Costa and N. Sepehri, "Understanding overall efficiency of hydrostatic pumps and motors," *International Journal of Fluid Power*, vol. 19, no. 2, pp. 106–116, 2018, doi: 10.1080/14399776.2018.1476306.

[105] S. Du, J. Zhou, J. Hong, H. Zhao, and S. Ma, "Application and progress of high-efficiency electro-hydrostatic actuator technology with energy recovery: a comprehensive review," *Energy Conversion and Management*, vol. 321, no. 119041, 2024, doi: 10.1016/j.enconman.2024.119041.

[106] A. Vaiuso, G. Immordino, L. Onofri, G. Coppotelli, and M. Righi, "Methods for Multi-objective Optimization PID Controller for quadrotor UAVs," arXiv:2509.17423, Sept. 2025.

[107] T. Hanne and M. J. Moghaddam, "A review of the evolution of multi-objective evolutionary algorithms," *Computers, Materials & Continua*, vol. 85, no. 3, pp. 4203–4236, Oct. 2025, doi:10.32604/cmc.2025.068087.

[108] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley, 2009, ISBN: 978-0470743614.

Appendix

MATLAB/Simulink Models

This chapter presents the Simulink models used in this thesis to simulate the electro-hydrostatic actuator (EHA) system and its integration with the energy storage and reutilization (ESR) unit. First, the EHA and ESR Simulink models are introduced. Next, the simulation results for EHA and ESR systems are also provided for illustration purposes. Finally, the Simulink models developed for online control and energy optimization are presented.

A.1 EHA Simulink Model

The EHA Simulink model includes the control strategies (torque and velocity control modes), quadrant detection, hydraulic flow and pressure dynamics, and actuator displacement and velocity. Figure A.1 illustrates the overall structure of the Simulink model.

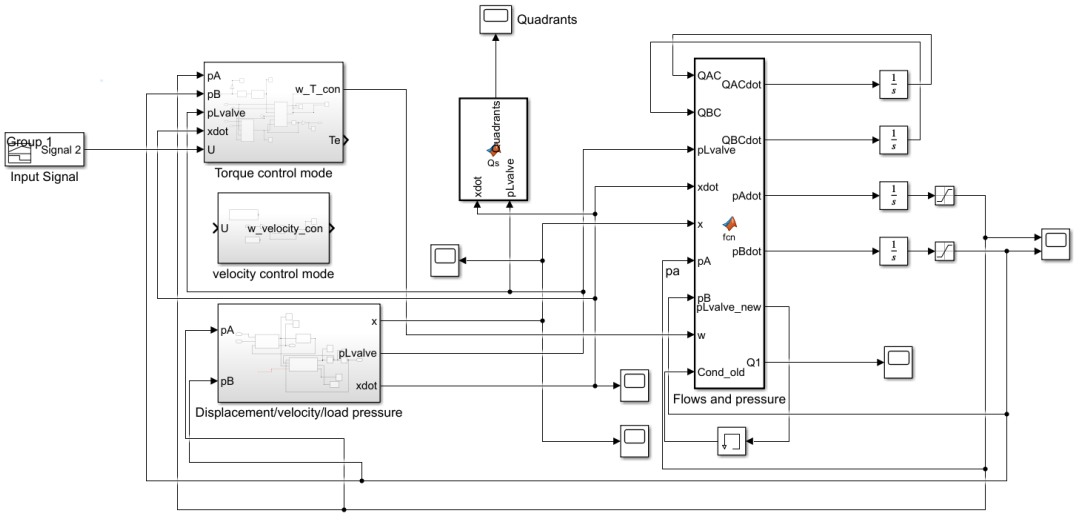


Figure A.1: Simulink model of the electro-hydrostatic actuator (EHA) system.

Figure A.2 illustrates the control subsystem includes torque control, and velocity control modes.

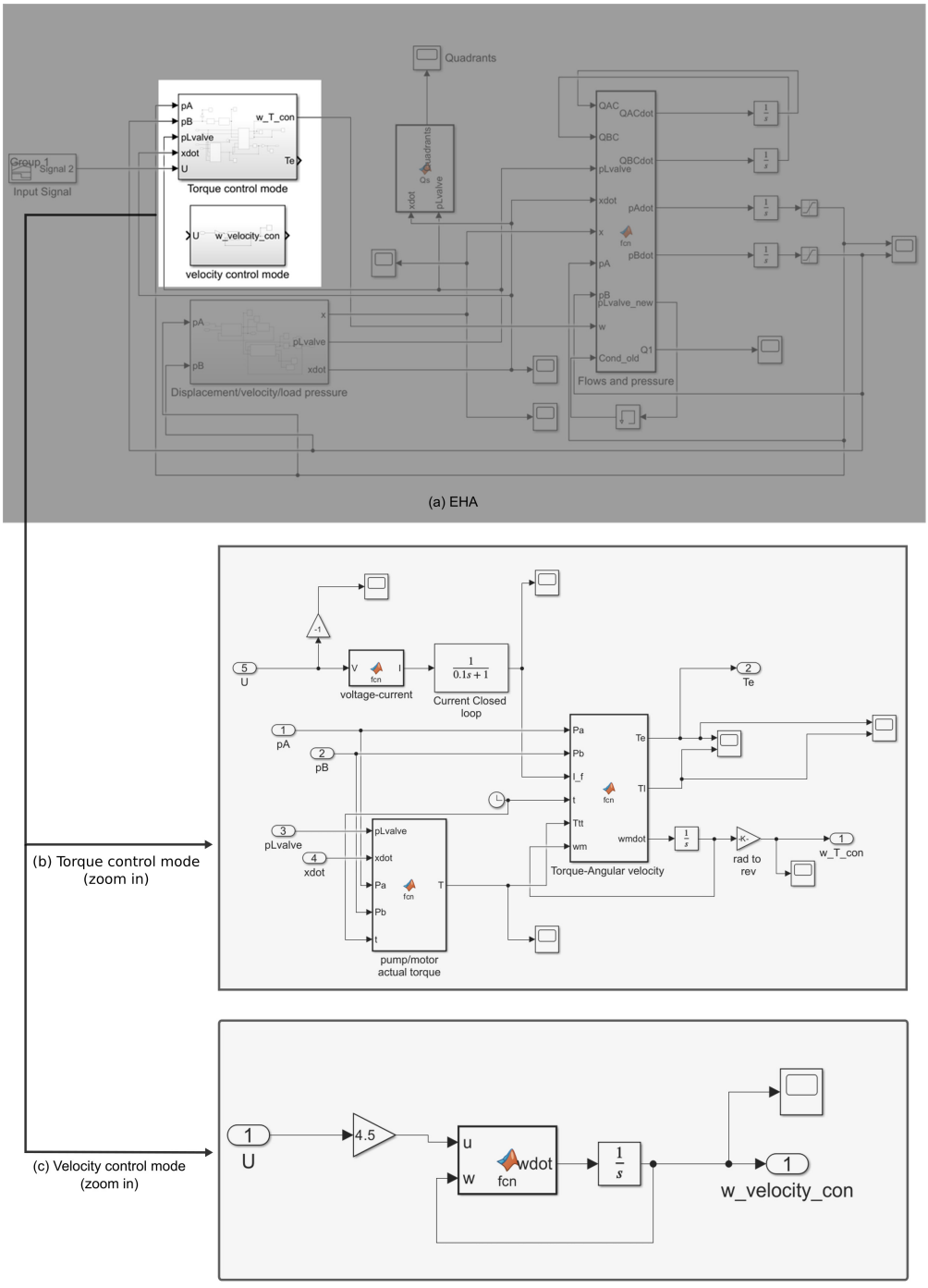


Figure A.2: Control subsystems: (a) EHA system; (b) magnified view of the torque control mode; (c) magnified view of the velocity control mode.

The hydraulic dynamics subsystem is represented by the block labeled *Flows and pressure*. This subsystem computes the dynamic behavior of the hydraulic variables, including chamber pressures and flow-related states. Figure A.3 illustrates the hydraulic dynamics subsystem.

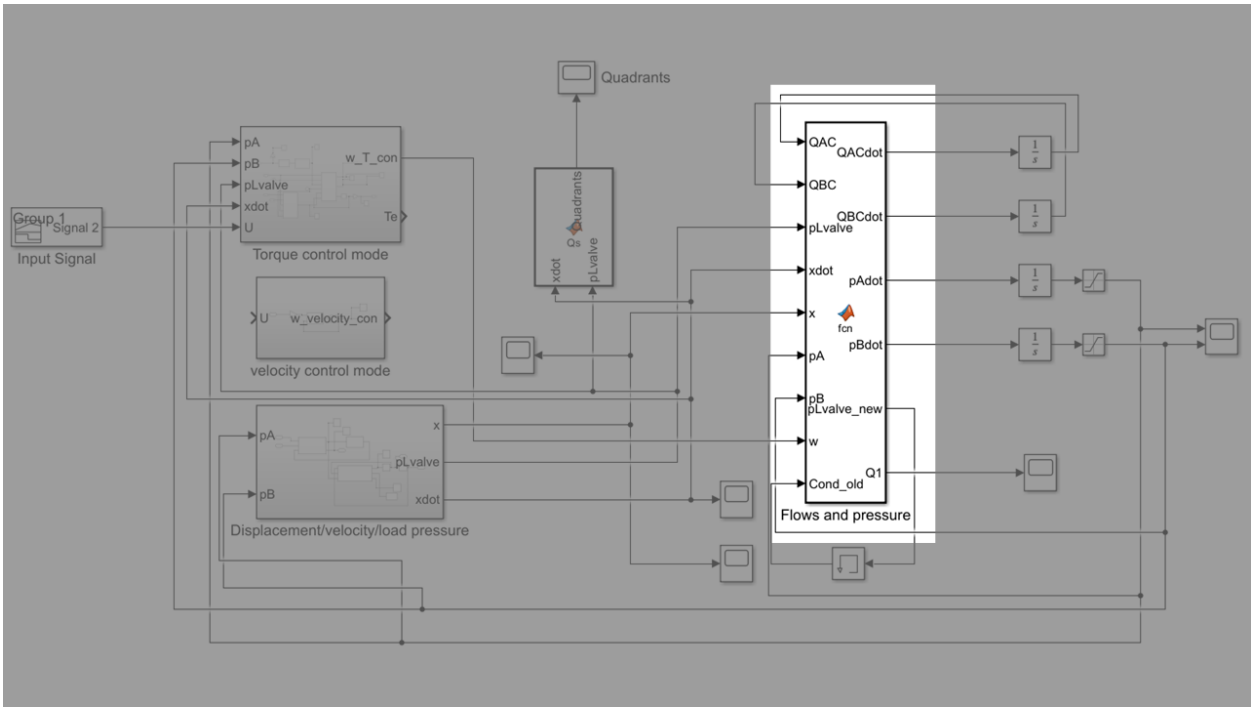


Figure A.3: Flow and pressure subsystem of the EHA Simulink model.

The actuator dynamics are represented by the subsystem labeled *Displacement/velocity/load pressure*. This part of the model calculates the motion of the hydraulic cylinder. In particular, it determines the actuator displacement x_p, velocity $\dot{x}_p$, and load-related pressure variables based on the hydraulic pressures in the cylinder chambers. Figure A.4 illustrates the actuator dynamics subsystem.

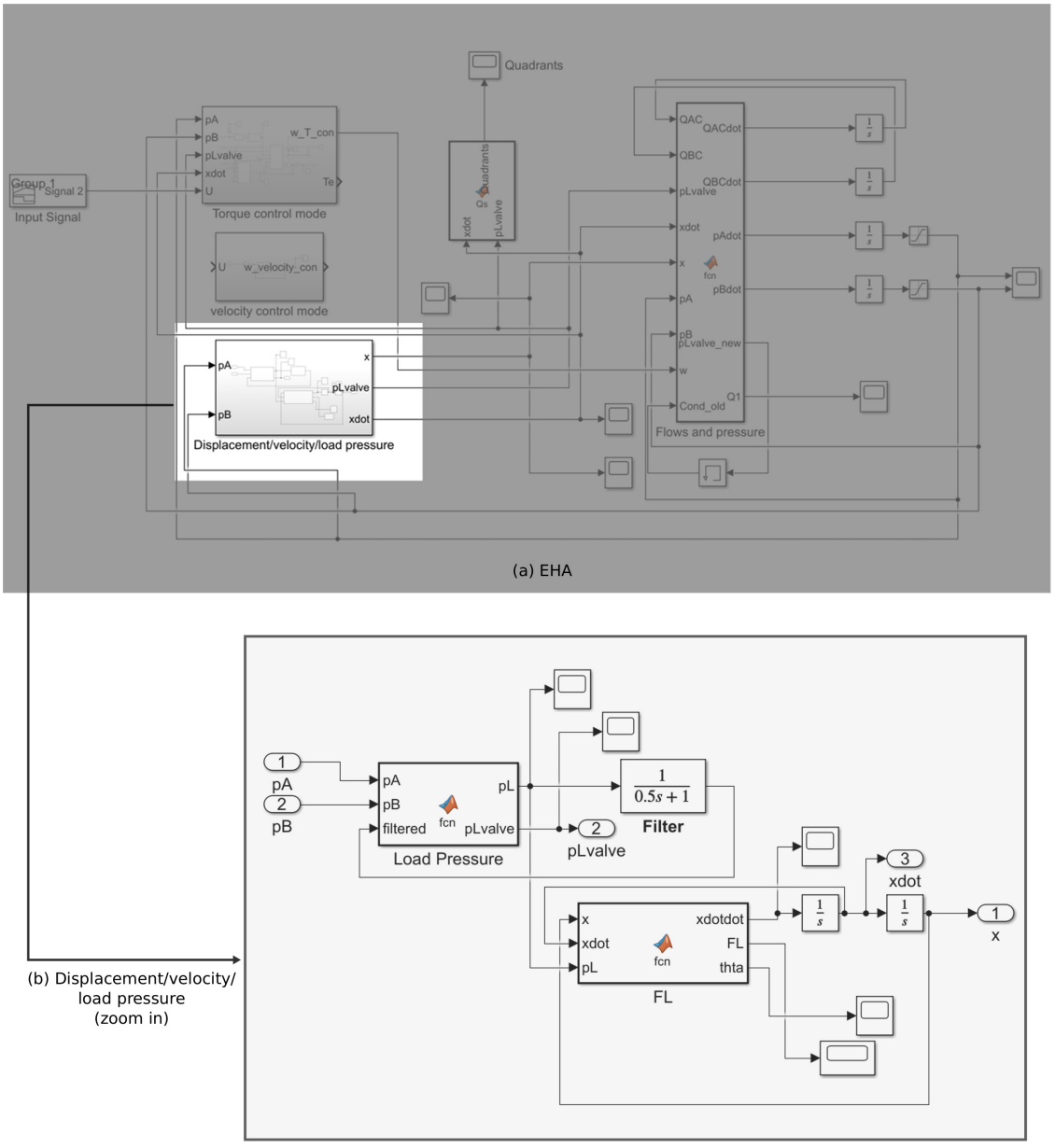


Figure A.4: Actuator dynamics subsystem: (a) EHA system; (b) magnified view of the displacement; velocity, and load pressure subsystem.

The EHA model also includes the *Quadrants* block, which is used to determine whether the system operates in motoring or pumping mode. Figure A.5 illustrates the quadrant block.

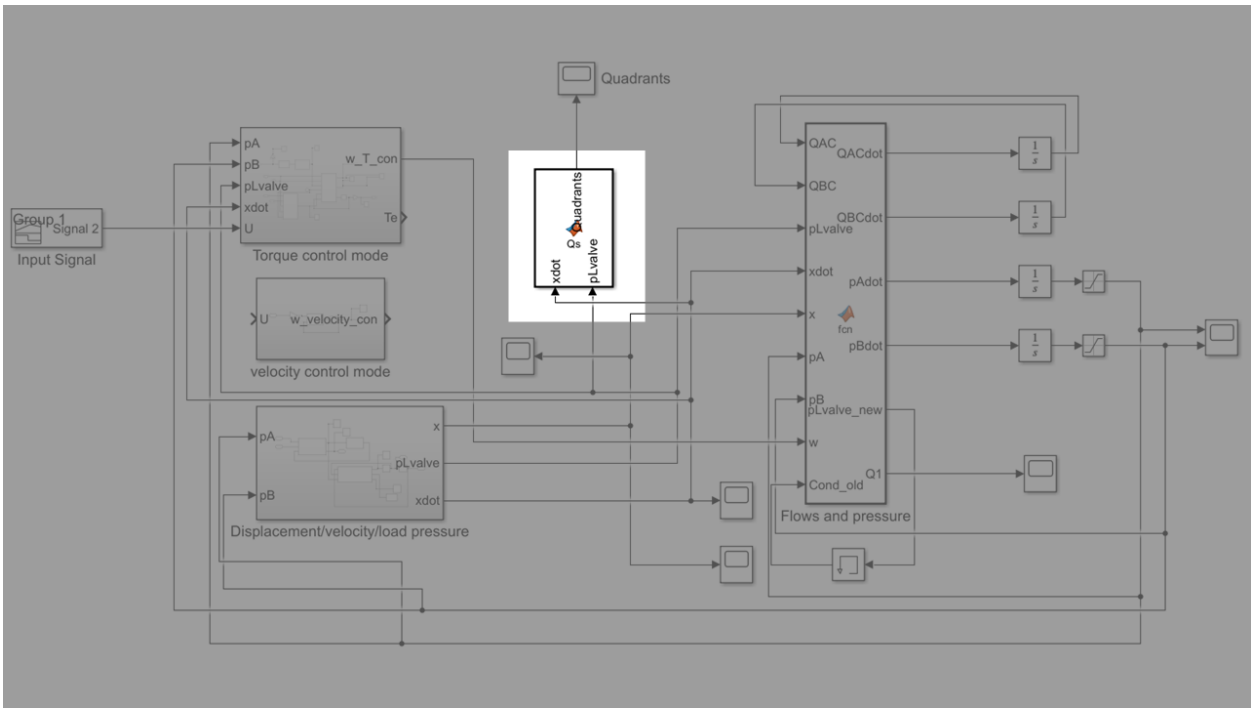


Figure A.5: Quadrant subsystem of the EHA Simulink model.

The simulation results are presented to illustrate the dynamic behavior of the system including input signals, actuator displacement, pressures, torques, and flow rates. These results are shown in Figs. A.6 to A.8. The validation of the simulation and experimental results for the EHA system has been carried out in [10].

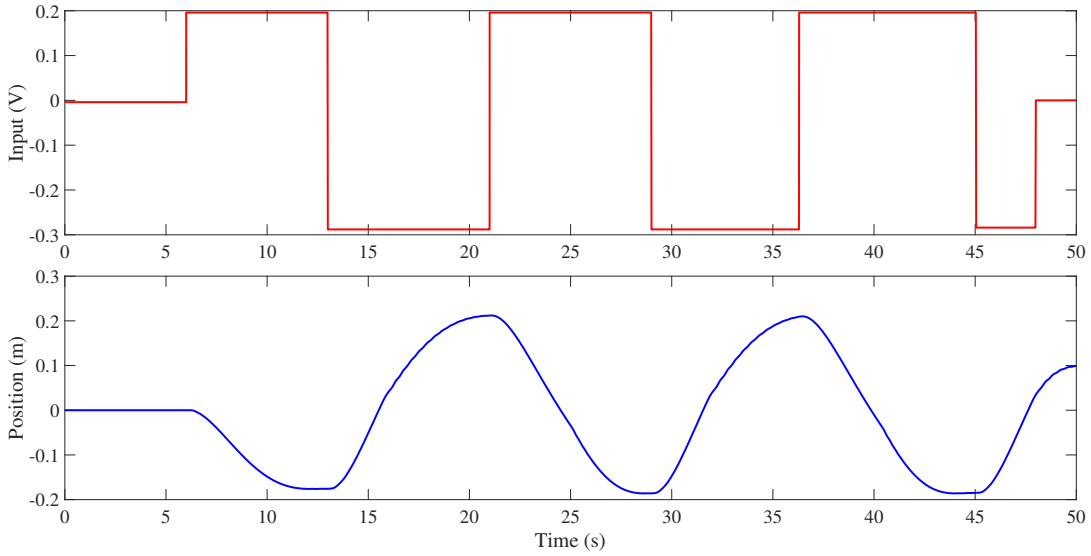


Figure A.6: Control input signal and corresponding actuator response: (a) input signal; (b) actuator position.

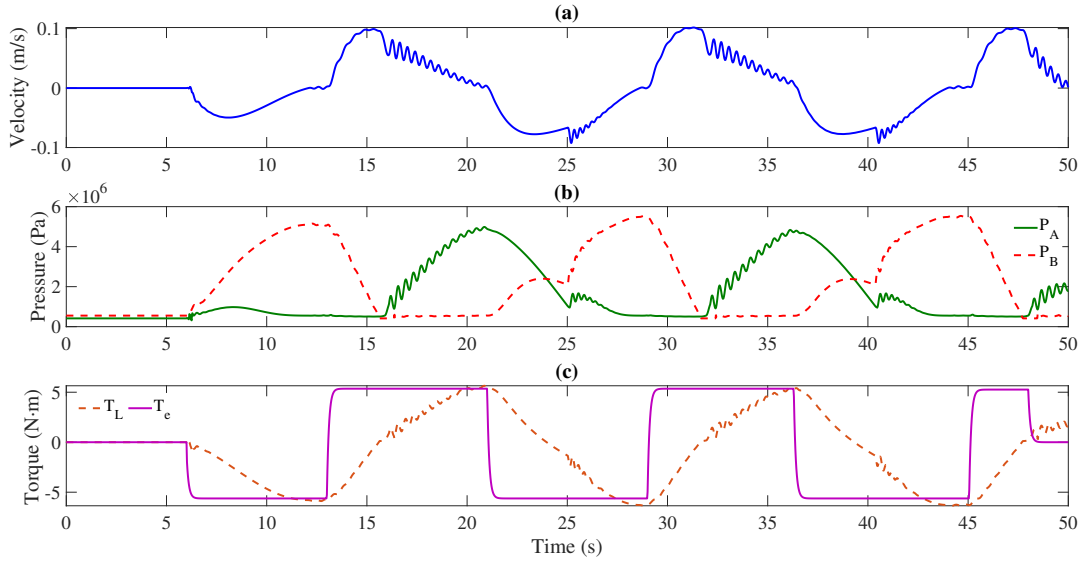


Figure A.7: System responses: (a) actuator velocity response; (b) chamber pressures P_A and P_B; (c) load torque T_L and motor torque T_e.

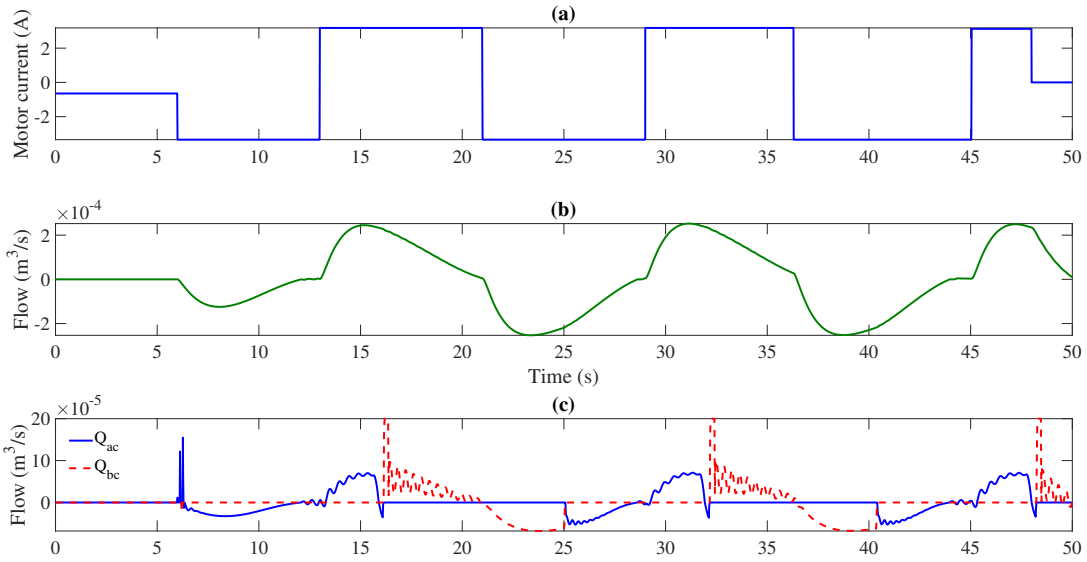


Figure A.8: System responses: (a) motor current; (b) flow rates of pump PM_1; (c) flow rates Q_{ac} and Q_{bc}.

A.2 ESR Simulink Model

This section presents the Simulink model of the energy storage and reutilization (ESR) attached to the EHA system. Figure A.9 illustrates the overall structure of the model. The ESR Simulink model consists of six main subsystems: the control subsystem, the ESR activation logic, the hydraulic flow and pressure dynamics, the actuator dynamics, torque control subsystem, and the quadrant block.

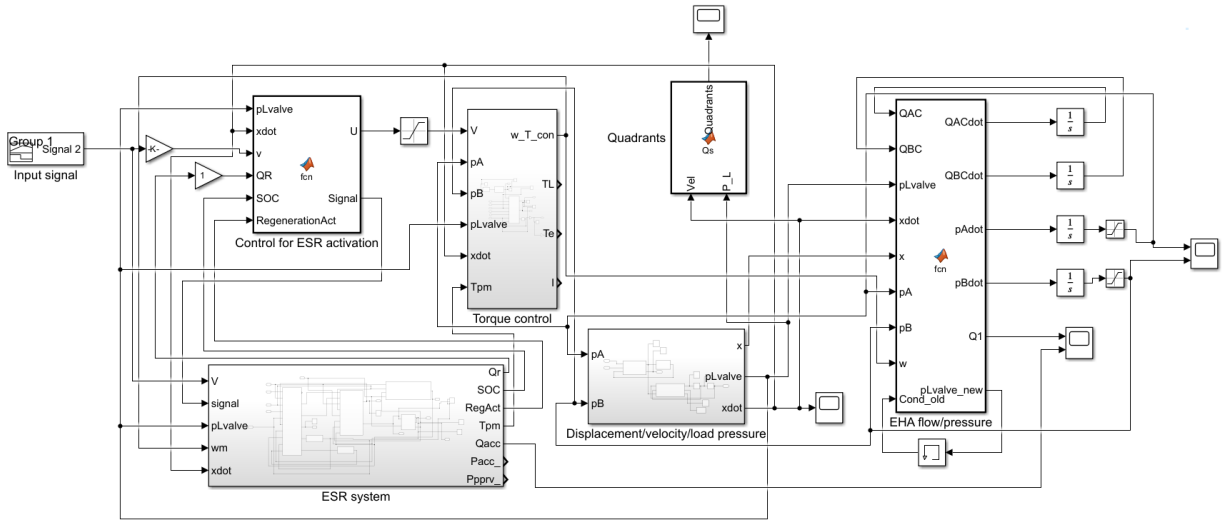


Figure A.9: Simulink model of the ESR attached to the EHA system.

The subsystem labeled *Control for ESR activation* determines the operating mode of the ESR

system. Figure A.10 illustrates the control subsystem for ESR activation.

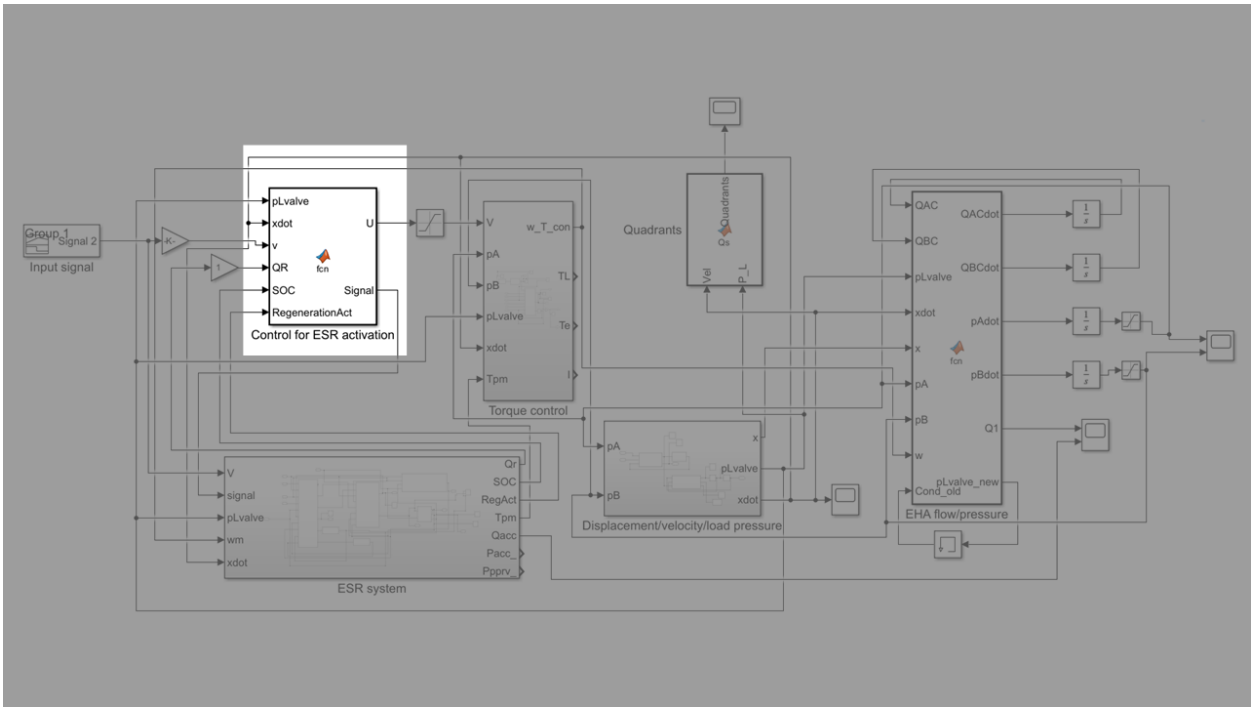


Figure A.10: Simulink model of the control subsystem for ESR activation.

The subsystem labeled *ESR system* represents the dynamic behavior of the ESR unit. This subsystem models the interaction between the hydraulic circuit and the accumulator, including energy storage and reutilization processes. Figure A.11 illustrates the ESR unit.

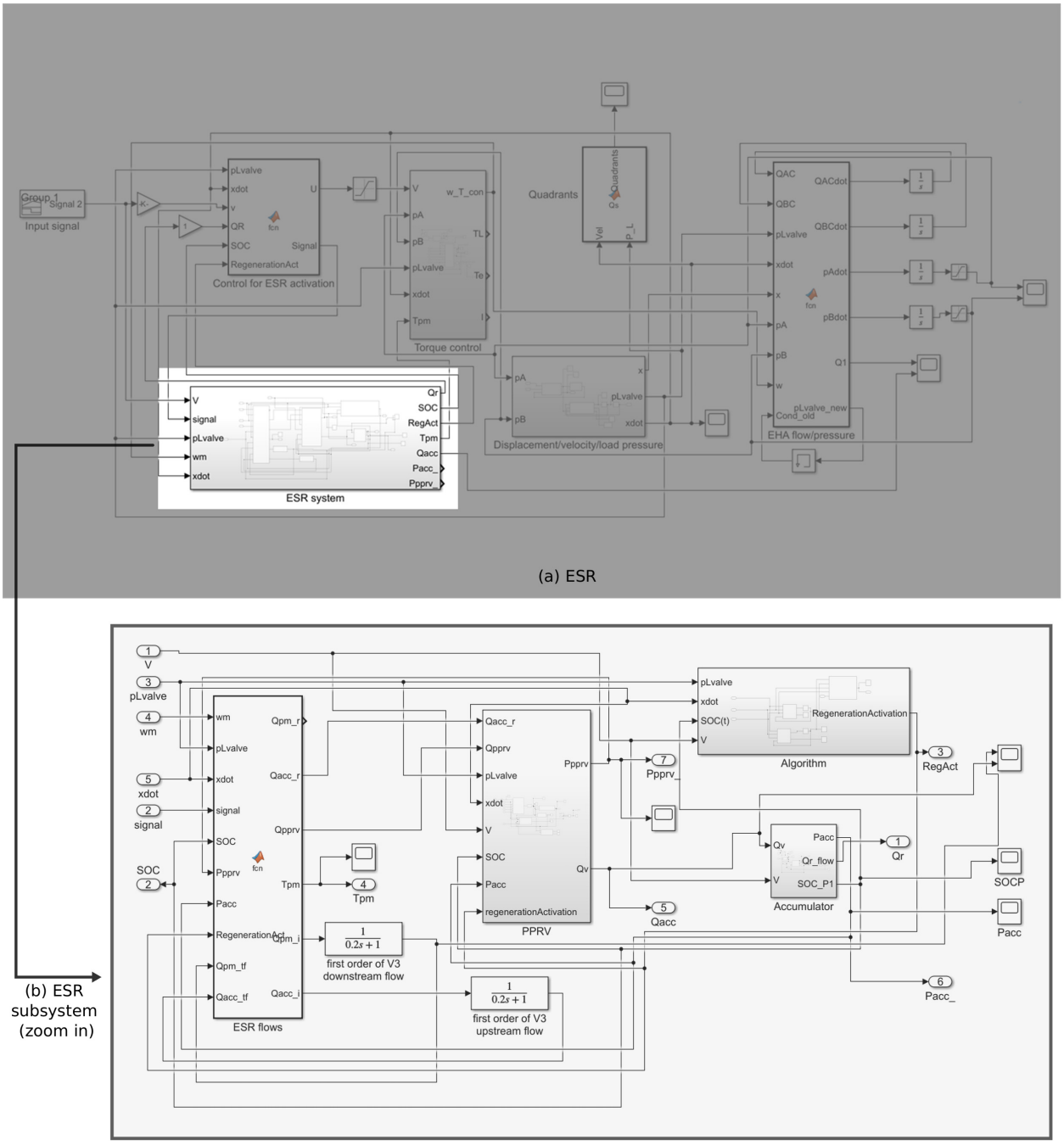


Figure A.11: ESR system: (a) ESR system; (b) magnified view of the ESR system.

Figure A.12 illustrates the torque control subsystem, which generates the required motor torque to achieve the desired system performance.

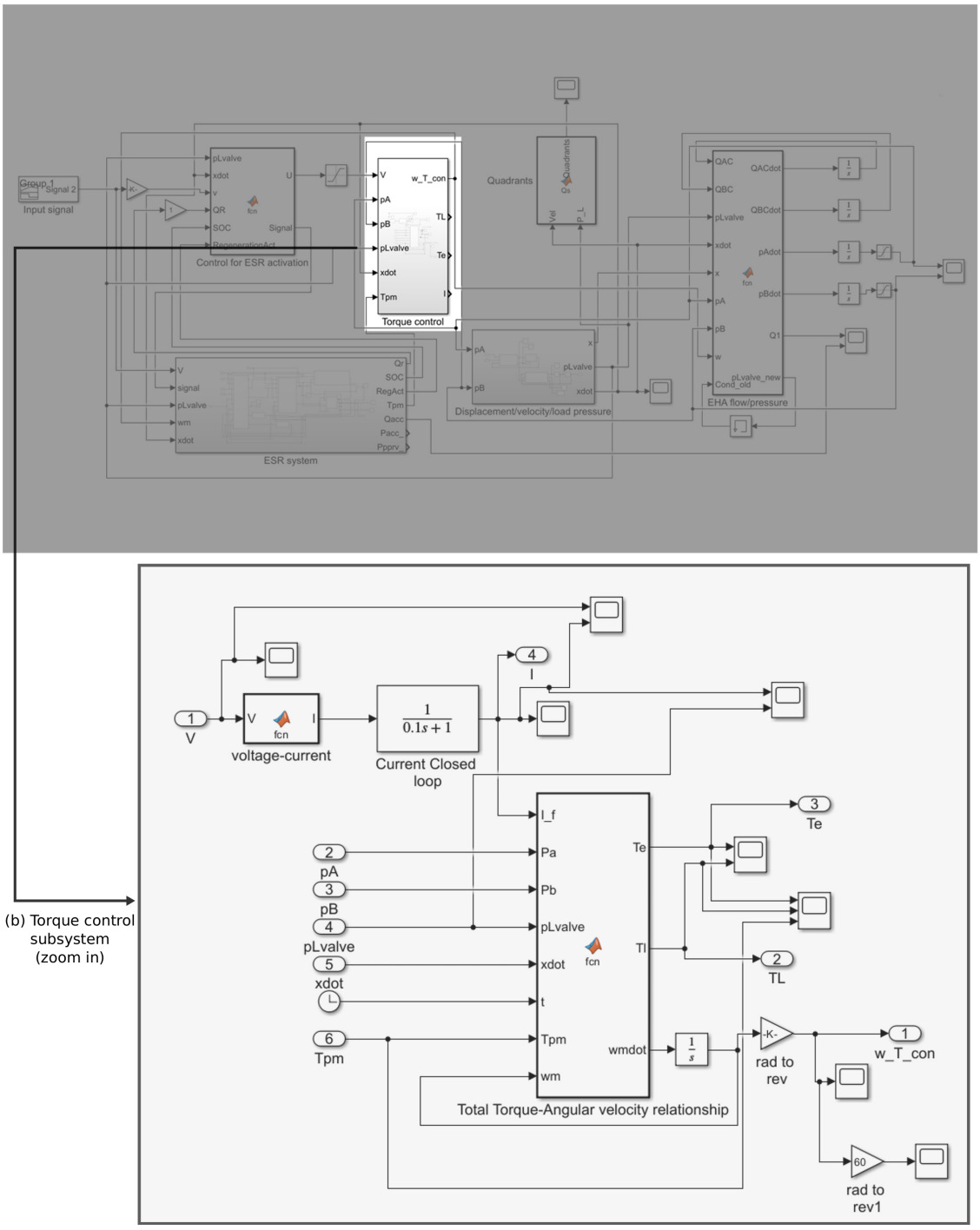


Figure A.12: Torque control subsystem: (a) ESR system; (b) magnified view of the torque control subsystem.

The hydraulic and actuator dynamics, flow and pressure dynamics, as well as the quadrant detection subsystems follow the same modeling framework described for the EHA system and are not repeated here. Figure A.13 illustrates these additional subsystems.

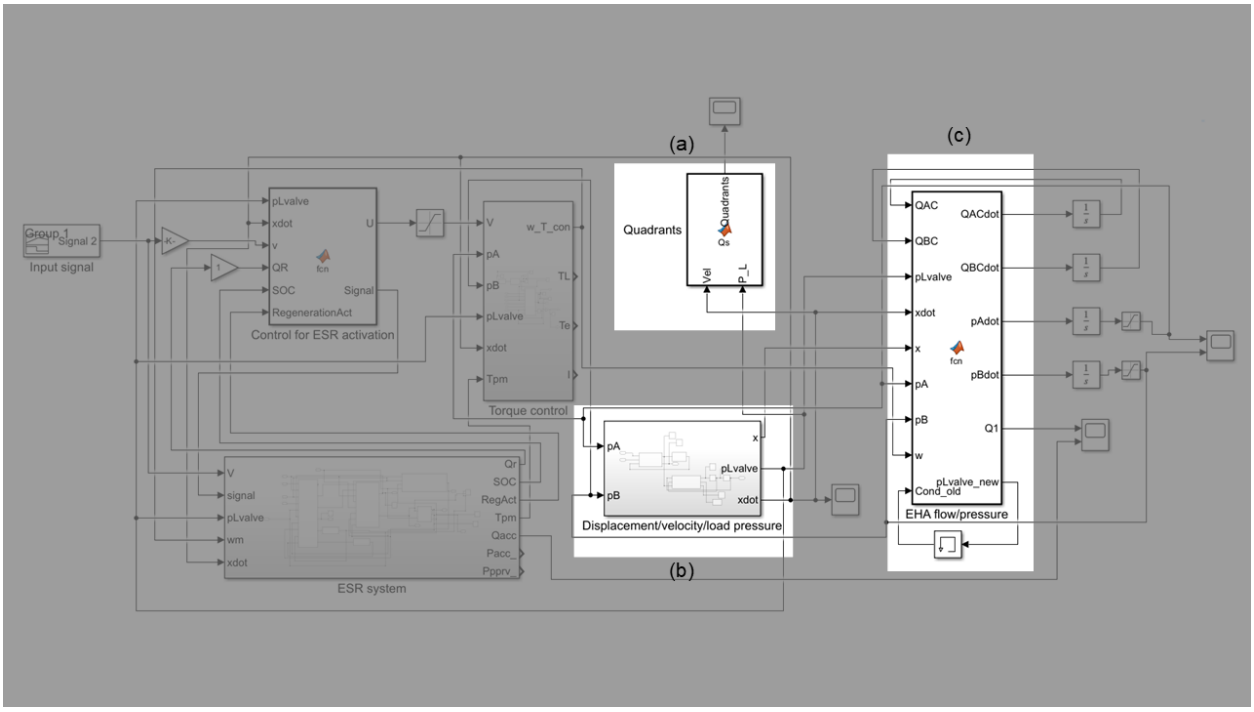


Figure A.13: Additional subsystems: (a) quadrant block; (b) displacement/velocity/load pressure; (c) flow and pressure dynamic of the EHA.

The simulation results are presented to illustrate the dynamic behavior of the ESR system, including the input signals, actuator responses, chamber pressures, motor and load torques, flow rates, and accumulator dynamics. These results are shown in Figures A.14 to A.17. The validation of the simulation against experimental results for the ESR system is reported in [10].

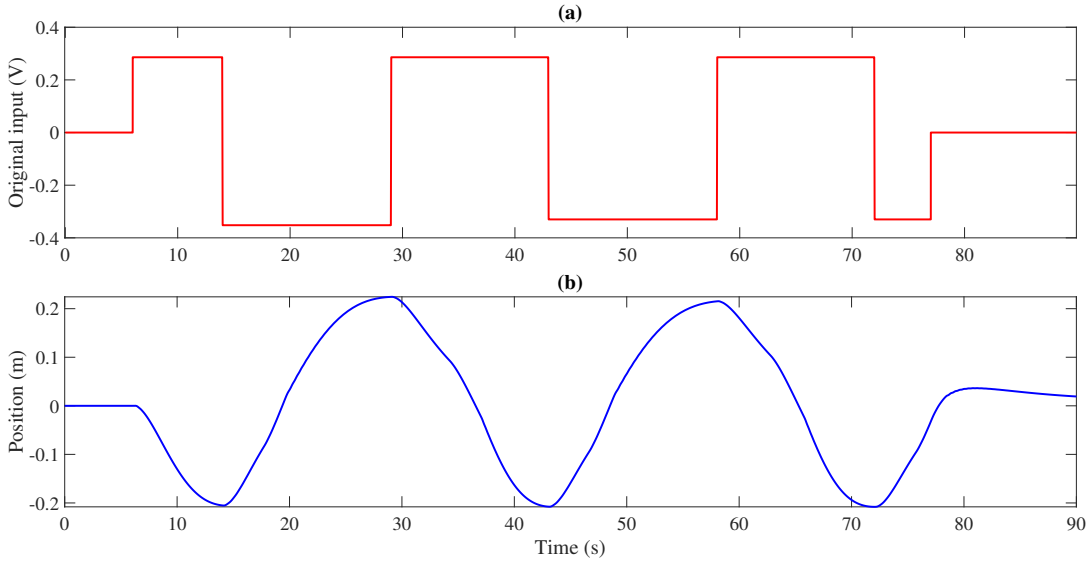


Figure A.14: Control input signal and corresponding actuator response: (a) input signal; (b) actuator position.

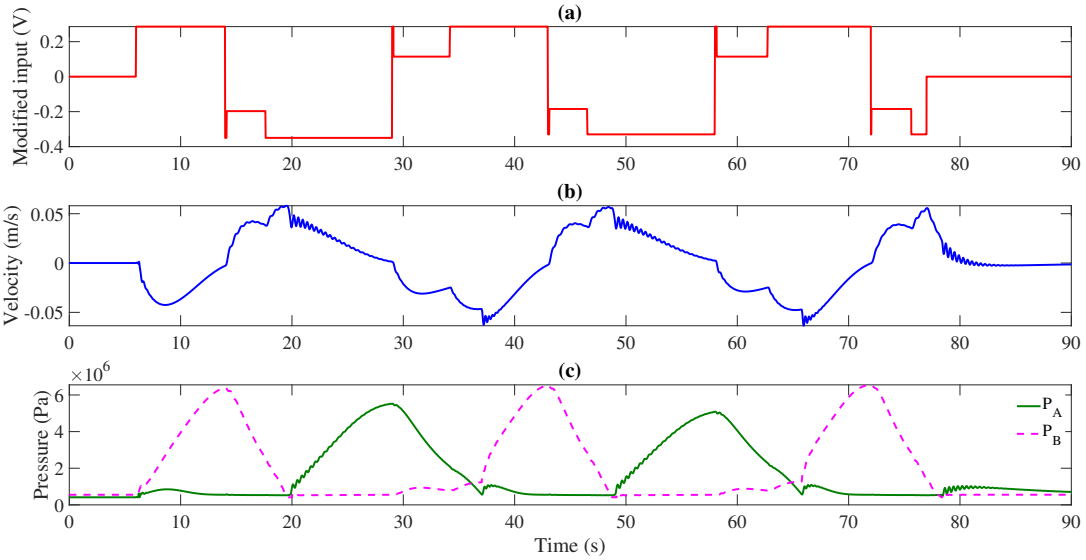


Figure A.15: System responses: (a) modified control input; (b) actuator velocity response; (c) chamber pressures P_A and P_B.

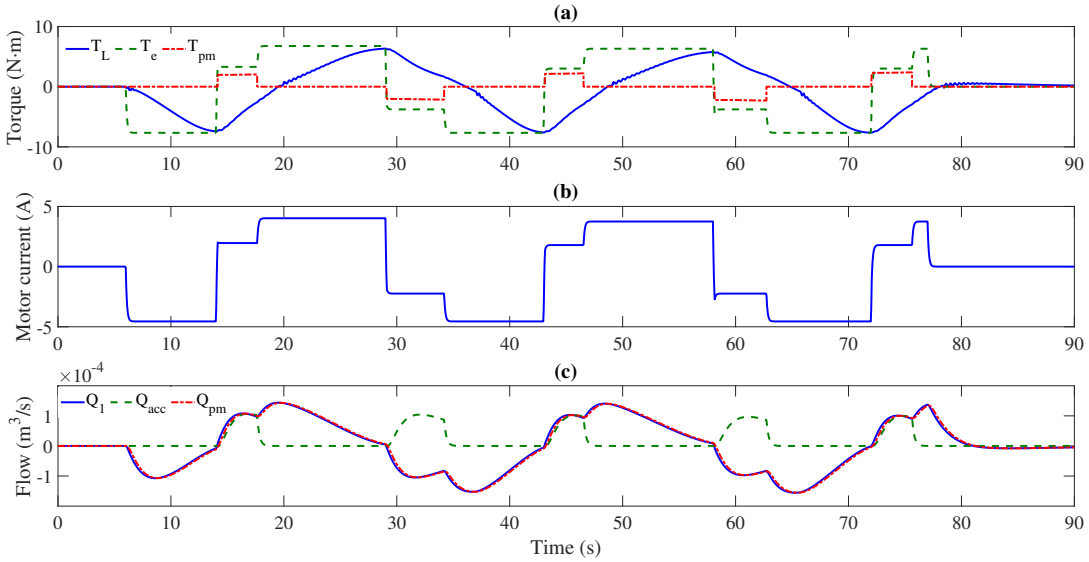


Figure A.16: System responses: (a) load torque T_L, motor torque T_e, and pump torque T_{pm}; (b) motor current; (c) flow rates of pump PM_1, accumulator Q_{acc}, and pump PM_2.

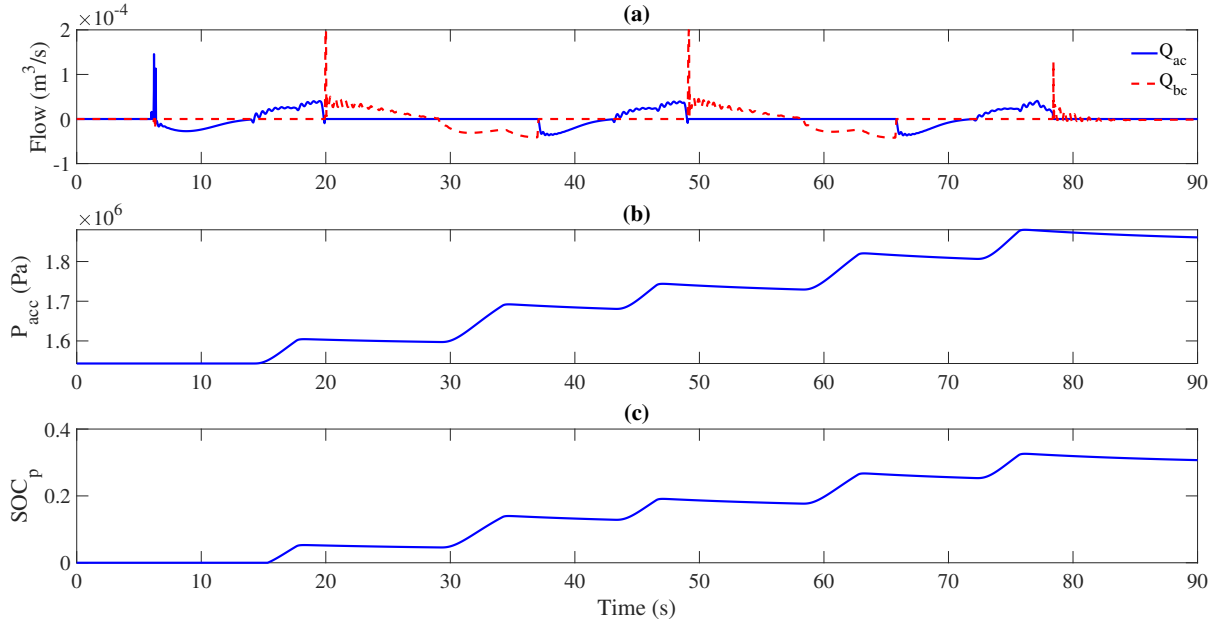


Figure A.17: System responses: (a) flow rates Q_{ac} and Q_{bc}; (b) accumulator pressure P_{acc}; (c) state of charge (SOC_p) of the accumulator.

A.3 Simulink Model for Model-Free Learning Control

Figure A.18 presents the Simulink model of the model-free control framework used for the EHA system. The PID controller includes an anti-windup mechanism to prevent excessive accumulation when actuator saturation occurs. During the online adaptation phase, the bulk modulus parameter β_e is intentionally altered to emulate changing operating conditions.

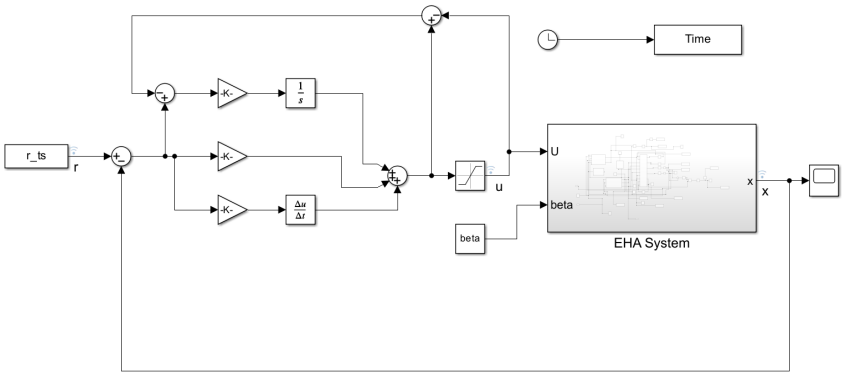


Figure A.18: Simulink model for model-free learning control

A.4 Simulink Model for Energy Optimization

Figure A.19 illustrates the energy optimization. The Simulink model includes three main parts: the optimization part and controller part and ESR subsystem.

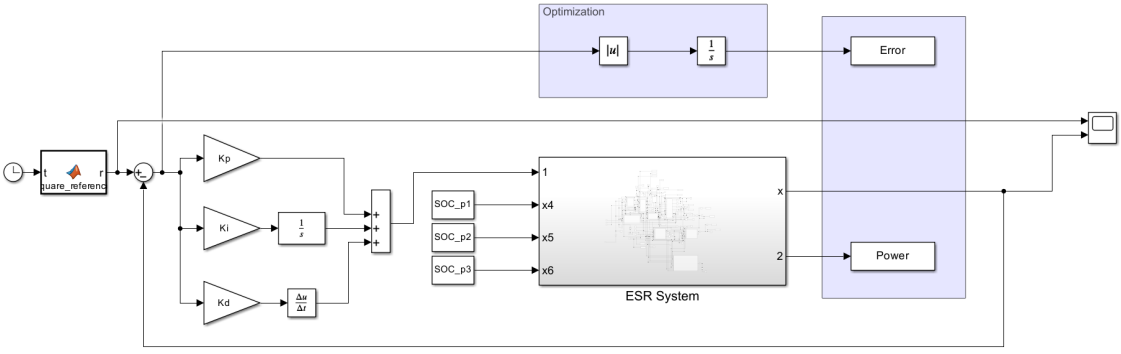


Figure A.19: Simulink model of the energy optimization system.

Figure A.20 shows the blocks for multi-objective optimization framework. The tracking error is first computed as the difference between the reference position and the actual actuator position. The resulting absolute error is subsequently integrated over time to obtain the integral of absolute error, which serves as a measure of the cumulative tracking error throughout the operating period. The

second objective function corresponds to the total energy consumption of the system, enabling a trade-off between tracking accuracy and energy efficiency.

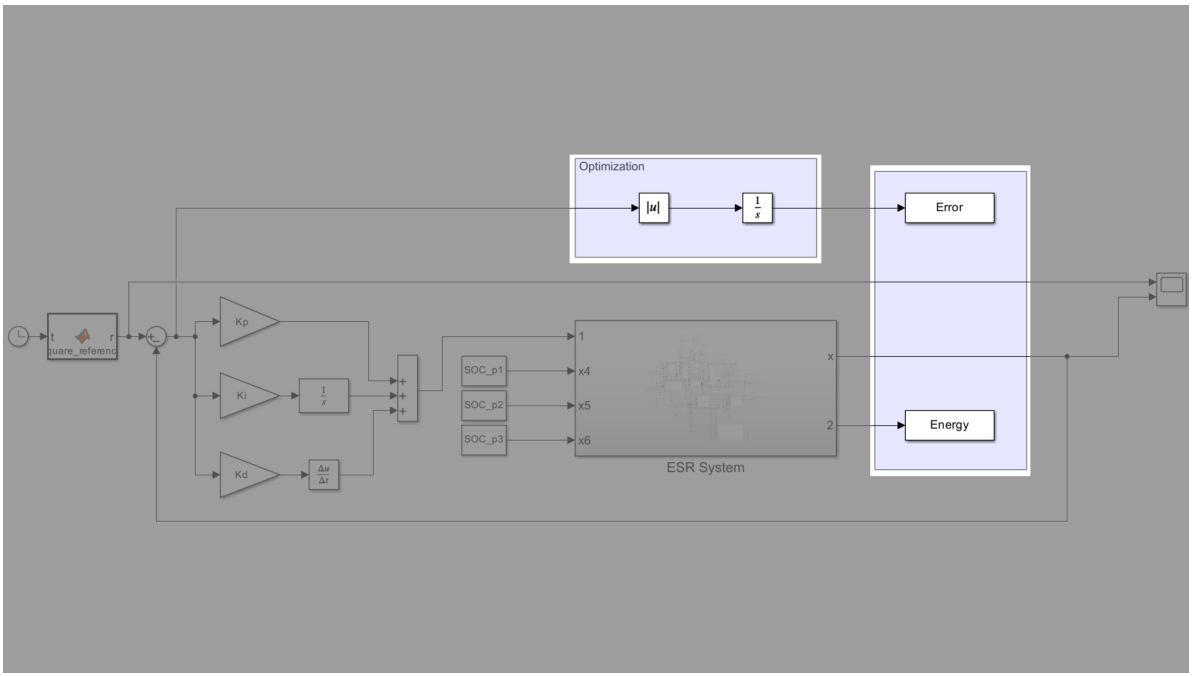


Figure A.20: Magnified view of the optimization subsystem.

Figure A.21 shows the ESR and controller subsystems, including the PID parameters and the ESR optimization variables. The optimization algorithm searches for six decision variables, consisting of the PID gains (K_p, K_i, K_d) and three ESR threshold parameters (SOC_{p1}, SOC_{p2}, SOC_{p3}). The PID gains regulate the actuator position tracking performance, while the ESR thresholds define the storage and reutilization decisions of the ESR system. By optimizing these variables using a multi-objective optimization framework, the proposed approach aims to improve both tracking accuracy and energy efficiency.

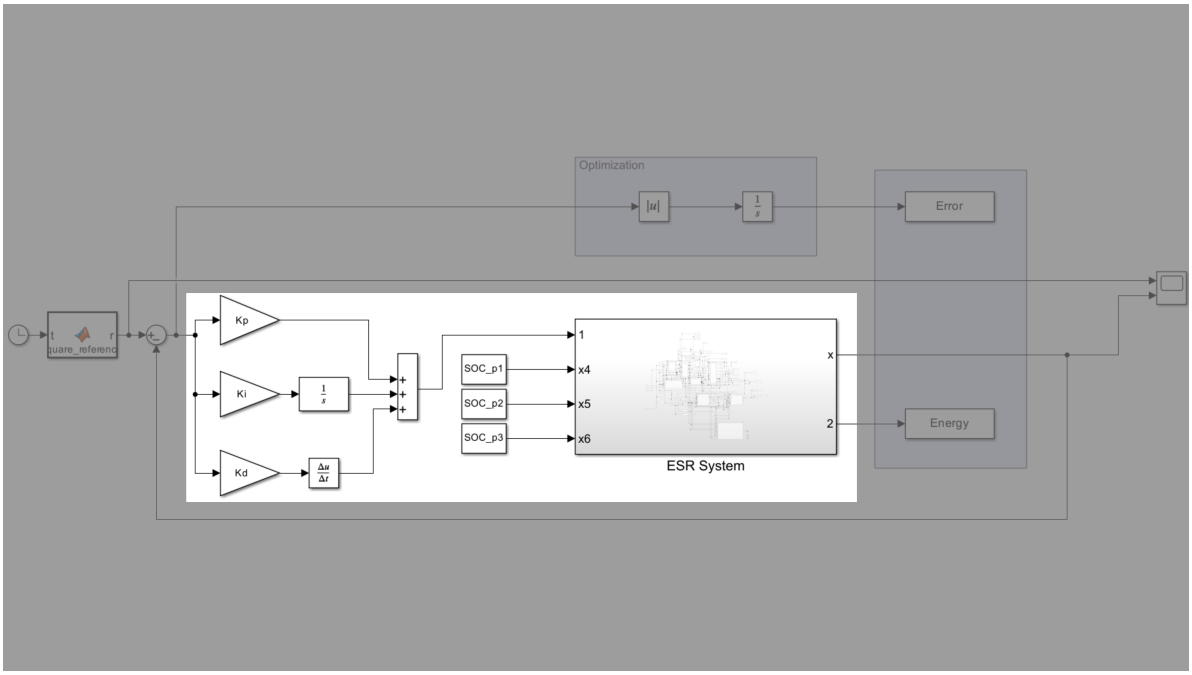


Figure A.21: Magnified view of the ESR and the PID controller subsystems.