Photocopy and Use Authorization

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission for extensive copying of my thesis for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature _____

Date _____

A Modeling and Controller Design for Continuous Electric Field Assisted Sintering System

By

Kanan Roy Chowdhury

A thesis

Submitted in partial fulfillment

of the requirements for the degree of

Master of Science

In

Measurement & Control Engineering

Idaho State University

Summer 2022

To the Graduate Faculty:

The members of the committee appointed to examine the thesis of Kanan Roy Chowdhury find it satisfactory and recommend that it be accepted.

_____

Dr. Marco P. Schoen, Major Advisor

_____

Dr. Ken Bosworth, Committee Member

_____

Dr. Paul Bodily, GFR

Dedication

To Avijit Roy,

whose books inspired me to become a freethinker.

Acknowledgements

I would like to express gratitude to my thesis advisor Dr. Marco P. Schoen, who was able to instill a love of control systems in me and allowed me the freedom and latitude to explore more. He consistently supported me during the whole process. I would also like to thank my committee members for letting my defense an enjoyable moment.

I would also like to thank all my fellow graduate students at Idaho State University, who I had the pleasure of getting to know while being able to share the ups and downs of completing a graduate degree.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

FAS	Field Assisted Sintering
SPS	Spark Plasma Sintering
CEFAS	Continuous Electric Field Assisted Sintering
INL	Idaho National Laboratory
RL	Reinforcement Learning
PID	Proportional Integral Derivative
LQR	Linear Quadratic Regulator
LQG	Linear Quadratic Gaussian
SI	System Identification
DDPG	Deep Deterministic Policy Gradient
DC	Direct Current
SISO	Single Input Single Output
PRBS	Pseudo Random Binary Sequence
LA	Linear Actuator

Abstract

Advanced Manufacturing methods such as Continuous Electric Field Assisted Sintering (CEFAS) is a novel approach to create advanced materials. The CEFAS system has the prospect to become an indispensable technique in advanced manufacturing, however, it also presents challenges including the modeling and control of the process with the objective to achieve specific material properties. In this thesis, the problem of modeling and regulating the CEFAS system has been investigated. A first principle method-based model for the CEFAS/hot rolling system is introduced. Furthermore, experiments are outlined to develop data driven models. System identification tools are utilized to extract quality linear representations of the system using simulation data. Proven traditional control schemes are devised to provide optimal solutions to the considered models. Furthermore, the prospects and possibility of implementing deep reinforcement learning as an intelligent control solution to the problem is studied.

Keywords: CEFAS, SPS, Control System, System Identification, Reinforcement Learning, LQG

Chapter 01: Introduction

1.1 Sintering Techniques

The process of sintering, which refers to the solidification of a powdered material by applying the combination of heat and pressure, is used by humans for thousands of years to produce products such as bricks, pottery, porcelain, and jewelry (*CISP History*, n.d.). Sintered materials have reduced porosity and improved thermal and electrical conductivity. Particularly in the USA, the estimated commercial value of sintered engineering materials is about 23 billion dollars in 2003 and is projected to cross 35 billion dollars by the end of 2028 (*CISP History*, n.d.). Traditional sintering methods use heat and pressure as tools to consolidate powder particles, and thus offer limited opportunities to influence the overall process. By introducing electric and magnetic fields as sintering tools, more flexibility and control over the sintering process can be added. This technique, commonly classified as Field Assisted Sintering (FAS), was first introduced by W.L. Voelker in the form of resistance sintering(Olevsky & Dudina, 2018). The progress, advantages and development strategies of FAS techniques are discussed in several publications (Dudina & Mukherjee, 2013; Groza & Zavaliangos, 2000; Mamedov, 2013; Munir et al., 2011a; Olevsky et al., 2013; Omori, 2006; Yurlova et al., 2014) since 1999. In these publications, authors particularly gave attention to a low voltage pulsed current assisted sintering called Spark Plasma Sintering (SPS). In layman's terms, SPS is a modification of the hot-pressing process, where a mold containing the sample material replaces the furnace, and the sample is heated by a direct current flowing through it (Pomeroy, 2021). The presence of plasma during the SPS process is highly arguable(Hulbert et al., 2009). However, statistics of

publications and patents on SPS is showing a rapid growth and is currently considered as the most popular technique among the researchers.

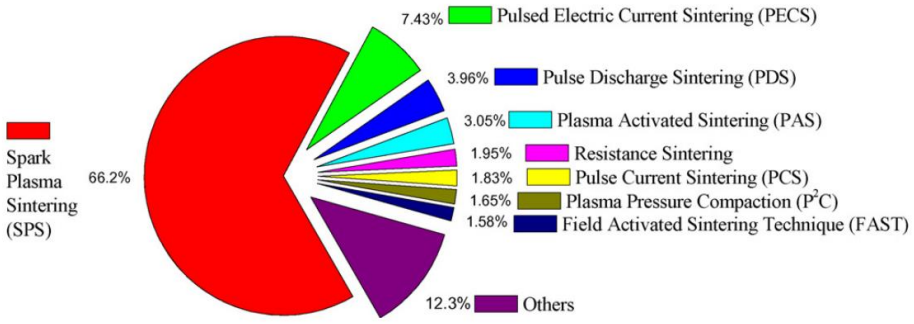


Figure 0.1: Relative distribution of different FAS techniques. (Orrù et al., 2009)

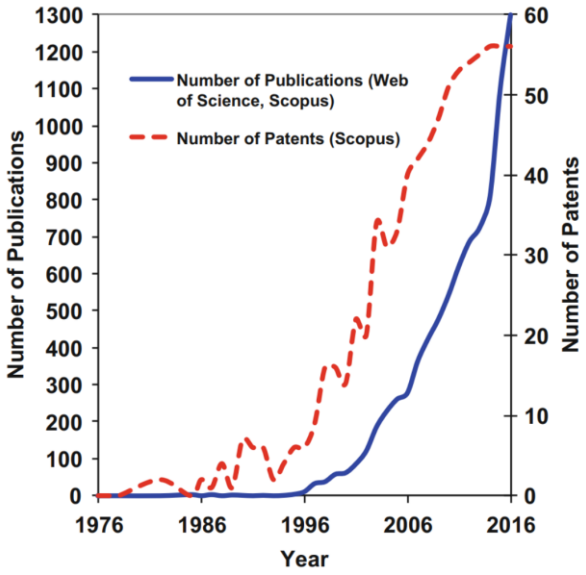


Figure 0.2: Trends of publications and patents on SPS. (Olevsky & Dudina, 2018)

However, all the current SPS technologies are batch processes (Cincotti et al., 2007; Holland & Muhkerjee, 2010; Hulbert et al., 2008; Morin et al., 2016) and experience tooling scalability problems. To encounter the scalability issues in Electric FAS, a new design called Continuous Electric Field Assisted Sintering (CEFAS) is proposed by Colasuonno (*Method for*

Creating Functionally Graded Materials with Spark Plasma Sintering and a Continuous Machine for Future Scalability, 2019.) which comes with the benefits of manufacturing materials faster than the traditional batch SPS techniques. Simulations using Multiphysics computational models showed a 200 times reduction in the processing time along with a 23 times decrement in the energy usage when using the CEFAS technique rather than the traditional batch SPS processes. Currently, Idaho National Laboratory (INL) is investigating the effectiveness of the CEFAS technique for an industrial scale production of fully dense advanced ceramic and metallic materials.

1.2 Problem Statement

The novel CEFAS technique has the potential to be a key technique in advanced manufacturing, however, this also presents challenges to control the continuous process to achieve certain targeted material properties while maintaining homogeneity. The primary goal of the control design is to regulate the stress and the joule heating process within the material of the part. Designing a controller for this purpose requires an adequate model of the system capturing the dynamic characteristics such as the effects of heating/cooling rate, rolling rate and stress relation regarding the densification process. Therefore, developing a reliable model is necessary prior to designing controllers. The modeling group at INL is using COMSOL Multiphysics® to model the system.

To evaluate the structure and dynamics of a model, system identification techniques are useful. This requires exciting the system using special inputs and extracting the corresponding

output signals. Using these input/output datasets and applying parametric system identification algorithms, it is possible to estimate the model parameters. Models extracted using system identification techniques can be used to develop primary controllers for the CEFAS system.

In the presence of model uncertainties and inaccuracies, robust control performs well, however, it still relies on linear dynamics. The CEFAS system is highly non-linear. Therefore, capturing the non-linear dynamics in the model is an essential to design a reliable controller. Currently, researchers are focusing more on developing models and controllers based on data instead of using a priori system models. Among these data-driven control schemes, Reinforcement Learning (RL) is drawing a lot of attention from the researchers. RL will be able to capture the non-linear system dynamics while embedding the controller into its algorithm.

The present work consists of developing conventional and data driven control schemes for the CEFAS system to regulate the pressure applied in the sintered materials. It can be divided into three parts-

- First principle modeling of the roller system and developing traditional controllers for that model.
- Using system identification to linearize the non-linear model of the system developed by INL.
- Proposing a data driven controller on the linearized system.

1.3 Literature Review

As discussed earlier, CEFAS is a novel design, and there is no published article that presents the control mechanism of this system. The CEFAS system developed in (*Method for Creating Functionally Graded Materials with Spark Plasma Sintering and a Continuous Machine for Future Scalability*, 2019.) only studies the scalability of the electric FAS process; thus, it leaves the control section for future work. Nonetheless, batch SPS is a well-known technique, and several researchers have proposed a number of different schemes in literature to control the temperature and the pressure during the process.

In Figure (1.3), a typical batch SPS apparatus is shown with the powder sample inserted into a graphite die and placed between two graphite punches which are not in direct contact with the rams. The rams are connected to an DC source which supplies the power. Most of the literatures (Achenani et al., 2017; Anselmi-Tamburini et al., 2005; Jiang et al., 2021; Locci et al., 2009; Manière et al., 2017; Sakkaki et al., 2020; Schwertz et al., 2016; Wang et al., 2010; Wei & Nolas, 2018) proposed proportional integral derivative (PID) controllers to regulate the entire process. A thorough investigation of how heating and cooling time lags influence the stability of a SPS PID controller is shown by Maniere *et al.*(Manière et al., 2017) where he proposed to place the temperature sensors in highly responsive areas to reduce regulation inaccuracy. Similarly, others (Achenani et al., 2017; Anselmi-Tamburini et al., 2005; Sakkaki et al., 2020; Wei & Nolas, 2018) have successfully implemented PID controllers to predict the current flowing through the sample; thus, evaluating the temperature distribution.

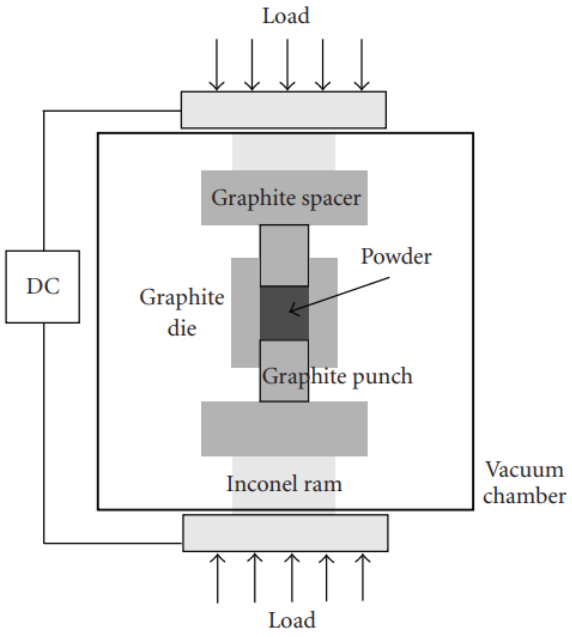


Figure 0.3: A typical batch SPS machine set-up.(Molénat et al., 2010)

However, all of these works only consider the joule heating phenomena and the effect of pressure/stress on the sample is neglected. Several studies (Li et al., 2012; Munir et al., 2006, 2011b) show the uniaxial load generates stresses which play a vital role in the densification process. In 2016, Schwertz et al. (Schwertz et al., 2016) published an article where a PID controller is developed to regulate a strongly coupled thermal-electrical-mechanical model. The authors used COMSOL Multiphysics ® to simulate a sample material from the beginning to the end of the densification process.

The physical phenomena happen on the material during the process of hot rolling is closely related to the CEFAS process. Therefore, implemented control schemes for hot rolling are studied. A novel PID system based on particle swarm optimization is proposed in (*A Novel PSO-PID Controller Application to Bar Rolling Process | IEEE Conference Publication | IEEE Xplore*, n.d.) that shows better performance than the traditional PID controller. Another study

(Calvo-Rolle et al., 2013) combines the conventional PID controller with an artificial neural network to create a hybrid controller to control the rolling process. Several other researchers proposed different control schemes such as Linear Quadratic Gaussian (LQG) controllers (Grimble & Hearns, 1998; Pedersen & Wittenmark, 1998) and H-infinity controller (Grimble & Hearns, 1999) to enhance the performance.

Chapter 02: Applied Theories

2.1 Modeling Theories

This section describes the modeling theories applied in this work. The first principle method is used to model the system primarily, and then, system identification (SI), which is a data driven modeling approach, is applied to build a linear model.

2.1.1 First Principle Method

First principle method of modeling is based on applying fundamental propositions and assumptions on a system. This requires understanding the physics behind a process and applying the appropriate equations and laws to represent the underlying correlations. Models based on first principle method are utilized to design primary controllers of a system. There are four steps in this approach; define modeling objective, acquire system information, develop equations to represent the system, and then validate the model.

2.1.2 System Identification

System identification is widely used technique to infer mathematical models of a dynamic system from the input and output data. This technique requires generating input data for

a system and extracting the output data for that specific input. There are four major steps to follow(Pintelon & Schoukens, 2001; *System Identification Overview - MATLAB & Simulink*, n.d.)-

- Step 1: Generating input and output data in either time or frequency domain.
- Step 2: Choosing an appropriate model structure.
- Step 3: Applying an estimation method to predict the adjustable parameters values in the candidate model structure.
- Step 4: Validating the estimated model.

To estimate a good model for a dynamic system, it is essential to design a good experiment to generates data that captures the dynamic behavior of the system. The input for the experiment should be sufficient to excite all the modes of the system. The next step is to select an appropriate model structure which is a mathematical relationship containing unknown parameters. This step is important as the SI algorithm tries to estimate the parameters based on the selected structure. By minimizing the error between the model output and the actual response, the SI algorithm fits the model parameters. To see whether the estimated model adequately represent the real system, a validation process is necessary.

2.2 Controller Theories

This section discusses the traditional and data-driven control theories that are applied in this work. PID and LQG control schemes are explored as traditional controllers while RL is studied as a data-driven controller.

2.2.1 PID Controller

The most common control technique is the proportional integral derivative controller (Jens Graf, 2016; St Clair, 1999). About 95% of the control loops existing in the process control application industry are either PID or PI controllers (Astrom, 1995). A typical PID controller has three actions: proportional, integral, and derivative. The proportional controller action is responsible for providing a straightforward gain to amplify the error signal. The integral control action enables the ability to eliminate the steady-state error. The derivative control action monitors the rate of change of the error signal and resist any abrupt change.

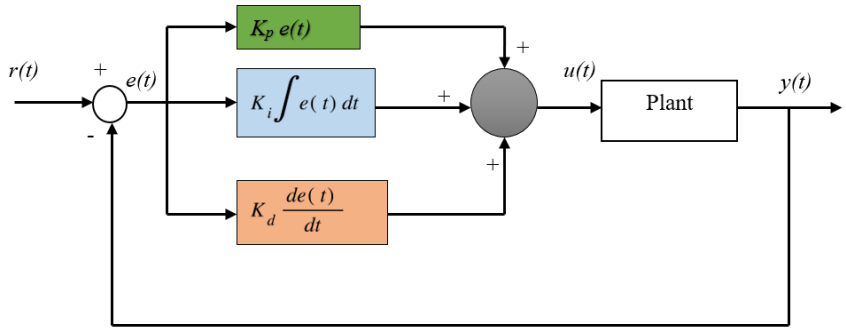


Figure 0.1: An ideal PID controller structure.

Figure (2.1) shows a typical PID controller structure, where *r(t)* is the reference set point, *e(t)* is the error signal, *u(t)* is the actuation command signal or the control signal, and *y(t)* refers to the plant output. The error signal, which is the difference between the reference and the measured output signal, can be represented by Equation (2.1).

$$e(t) = r(t) - y(t) \tag{2.1}$$

The error signal is then fed to the PID controller to convert it into a control signal to send it to the plant. The control signal is the summation of the proportional gain times *e(t)*, the integral gain times of the integral of *e(t)*, and the derivative gain times of the derivative of *e(t)*. This control signal *u(t)* can be described using Equation (2.2)

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{2.2}$$

Here, K_p refers to the proportional gain, K_i is the integral gain, and K_d is the derivative gain. The plant generates an output, *y(t)* which is then fed back to generate a new error signal. This is a recursive process while the PID controller is in effect. By taking the Laplace transform of Equation (2.2), the transfer function of a PID controller can be found.

$$G(s) = K_p + \frac{K_i}{s} + K_d s \tag{2.3}$$

A pure derivative term in the PID structure can amplify the measurement noises. In practice, a lowpass filter with filter coefficient, N is added to the derivative term. The transfer function for the PID with the filter coefficient term is shown in Equation (2.4).

$$G(s) = K_p + \frac{K_i}{s} + K_d \frac{N}{1 + \frac{N}{s}} \tag{2.4}$$

2.2.2 Linear Quadratic Gaussian Regulator

When full-state measurements of a system are available, it is possible to design a full-state feedback controller for a system given that the system is controllable. For example, linear

Quadratic Regulator (LQR) is an optimal full-state feedback controller. However, in practice, the availability of full-state measurements is rare, especially for high-dimensional systems(Brunton & Kutz, n.d.). It is possible to design a full-state estimator to estimate the unmeasurable states if the system is observable. Linear Quadratic Gaussian (LQG) controller is a solution to this problem. LQG performs an optimal full-state estimation by a Kalman filter while it offers a control solution using an LQR. Although LQG provides an optimal control solution, it doesn't have any guarantees on robustness (Doyle, 1978). Figure (2.2) shows a schematic illustration of the LQG controller without considering any system disturbances and measurement noises.

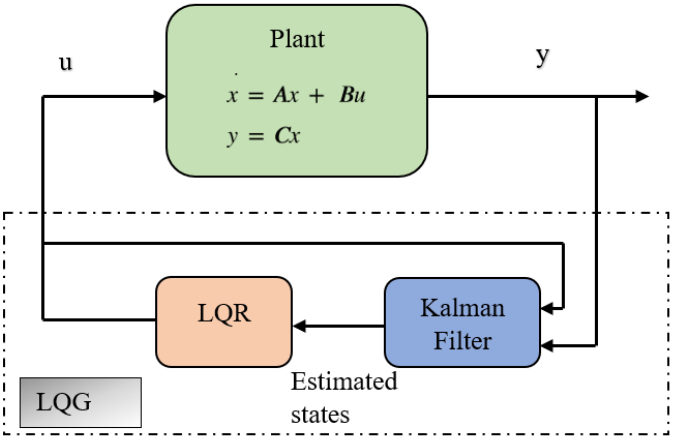


Figure 0.2 : LQG control scheme.

The dynamical system of LQG can be represented by the following state-space equations where the plant output acts as input, *u* as output (Brunton & Kutz, n.d.).

$$\dot{\hat{x}} = \left(A - K_f C - BK_r\right)\hat{x} + K_f y \qquad 2.5$$
$$u = -K_r \hat{x}$$

The estimated states are denoted as $\hat{x}$, K_r and K_f represent the LQR and Kalman filter gains, respectively. The cost function, *J(t)* of the LQG problem is shown in Equation (2.6) (Brunton & Kutz, n.d.).

$$J(t) = \left\langle \int_0^t \left[x(\tau)^* \boldsymbol{Q} x(\tau) + u(\tau)^* \boldsymbol{R} u(\tau) \right] d\tau \right\rangle \tag{2.6}$$

where $\boldsymbol{Q}$ is a semi-definite matrix used for penalizing the states and $\boldsymbol{R}$ is a positive definite matrix used to penalize actuator effort. By adjusting the weights in $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices, the closed-loop behavior of the system can be changed. The state estimation error can be treated as ε and the closed-loop system for LQG can be represented with Equation (2.8) where w_d and w_n represent the system disturbance and measurement noise, respectively.

$$\varepsilon = x - \hat{x} \tag{2.7}$$

$$\begin{bmatrix} \dot{x} \\ \dot{\varepsilon} \end{bmatrix} = \begin{bmatrix} A - BK_r & BK_r \\ 0 & A - K_f C \end{bmatrix} \begin{bmatrix} x \\ \varepsilon \end{bmatrix} + \begin{bmatrix} I & 0 \\ I & -K_f \end{bmatrix} \begin{bmatrix} w_d \\ w_n \end{bmatrix} \tag{2.8}$$

The closed-loop eigenvalues of a LQG regulator are given by the eigenvalues of the two diagonal matrices $(A\text{-}BK_r)$ and $(A\text{-}K_fC)$, which can be optimally chosen by the LQR and the Kalman filter gain matrices. Although LQG is an optimal regulator, it is possible to introduce robustness by different methods such as loop transfer recovery (Athans et al., 2012).

2.2.3 Reinforcement Learning

For a lot of control theories to be applied, it is essential to have a linear-time-invariant (LTI) system. While linearizing a nonlinear system to apply linear control theories, crucial information on nonlinear dynamics can be lost (Bertsekas, n.d.). Instead of linearizing the

system, a machine learning (ML) training algorithm named reinforcement learning (RL) can be implemented to automize a reward/punishment-based method to train itself to regulate the system (Vamvoudakis et al., n.d.). The component that generates the decision of what action to perform is called an RL agent. An RL agent treats the system as an unknown dynamic environment, performs action on the environment based on observations, and modifies itself to perform better. Three different approaches are available to implement an RL algorithm: value-based, policy-based, and model-based learning. While value-based RL method tries to maximize a value function, policy-based method uses Markov decision process to perform actions to maximize the future reward. These two are also known as model-free RL methods. For a model-based method a virtual model of the physical system should be provided. Figure (2.3) shows a schematic of a typical RL controller with a policy-based learning method.

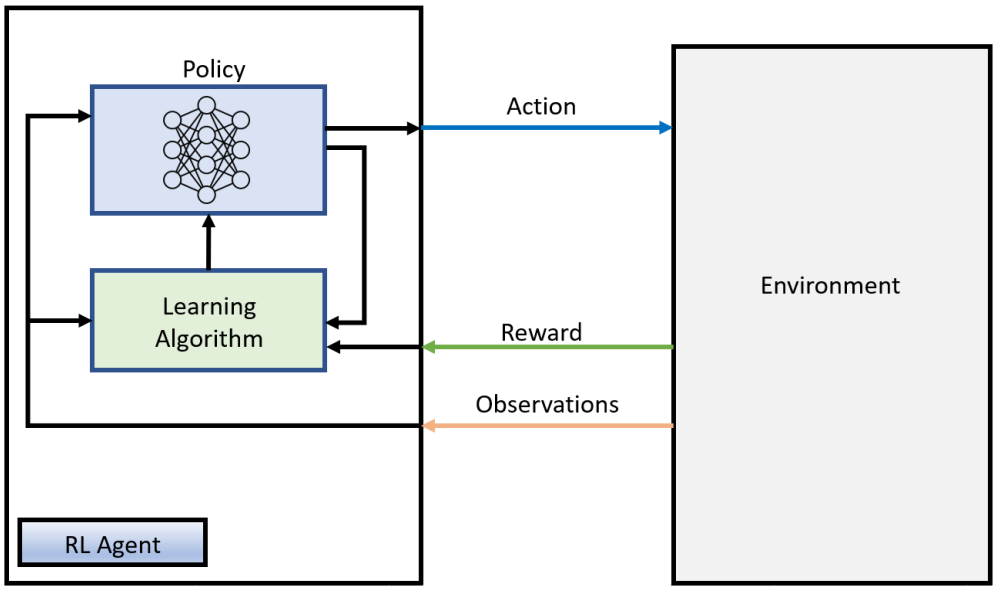


Figure 0.3: A policy based RL method.

The RL workflow can be divided into five steps which is shown in Figure (2.4). Firstly, an environment should be defined where the RL agent can learn. The next step is to create a

reward function to properly incentivize the agent to do the desired task. Then a policy should be constructed to structure the parameters and logic of the decision-making process of the agent. After setting up everything, the agent should be trained to find the optimal policy. The last step is to validate the trained policy to see how well it performs and then deploying it.

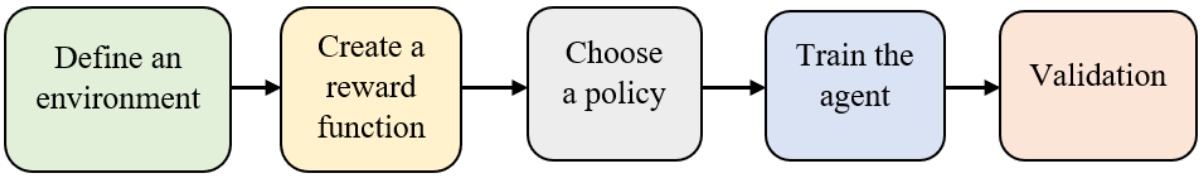


Figure 0.4: Reinforcement learning workflow.

The agent in RL consists of 2 components: a policy and a learning algorithm. Among several existing agents, Deep Deterministic Policy Gradient (DDPG) is an actor-critic type agent for which the action space is continuous. DDPG consists of two neural networks known as actor and critic. The actor network returns an action for a given observation to maximize the reward function and updates its weights by using feedback from the critic network. The critic network tries to estimate the value of the state-action pair and uses the reward to function to determine the accuracy of its prediction. The error is the difference between the new estimated value of the previous state and the old value of the previous state from the critic network. The new estimated value is based on the received reward and the discounted value of the current state. This error is used by the critic network to update itself and the actor network's weights. This shapes the policy to follow the reward slope recommended by the critic and thus tries to learn the optimal behavior. Figure (2.5) shows the schematic of a DDPG policy based RL algorithm.

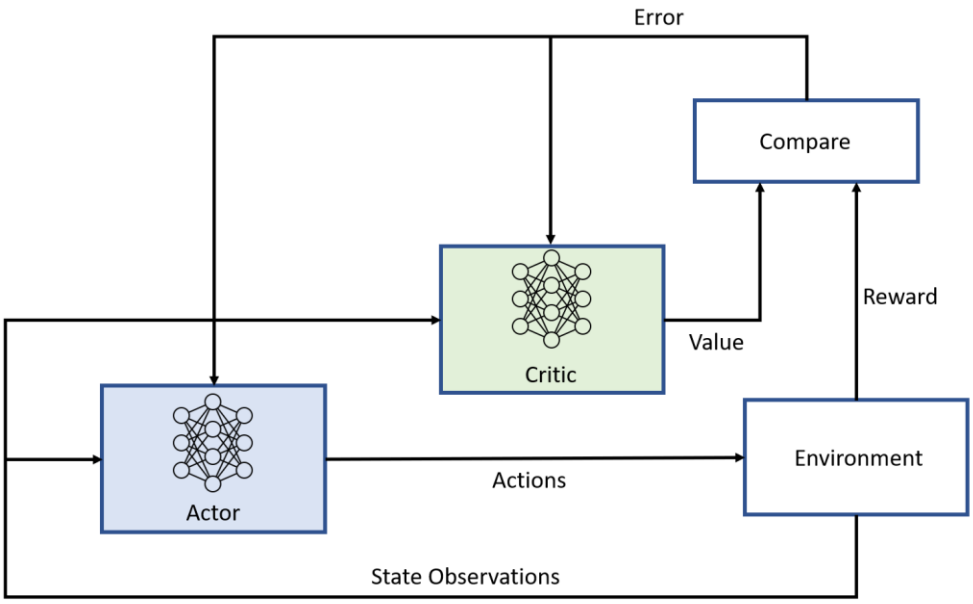


Figure 0.5: Schematic of a DDPG policy based RL algorithm.

Chapter 03: Modeling

3.1 System Development by First Principle Modeling

As previously discussed, the physical phenomena that happens on the material during the CEFAS process can be closely related to the hot rolling process. Therefore, a static model of hot rolling mechanism is developed using first principle method for primary investigation. Furthermore, a linear actuator is utilized for pressing the roller which is included into the model as well.

3.1.1 CEFAS/Hot Rolling Model

A flat plate rolling process along with two rotating rolls is considered for the modeling task. Figure (3.1) shows a schematic of the process considered. The sample material with an entry thickness or height, h_b and an initial velocity, V_0 passes through the rollers to produce a dense material with final height, h_f and exiting the rollers with a final velocity, V_f. The workpiece zone is denoted as L which has a neutral point, N i.e., velocity of the sample is same as the roll velocity.

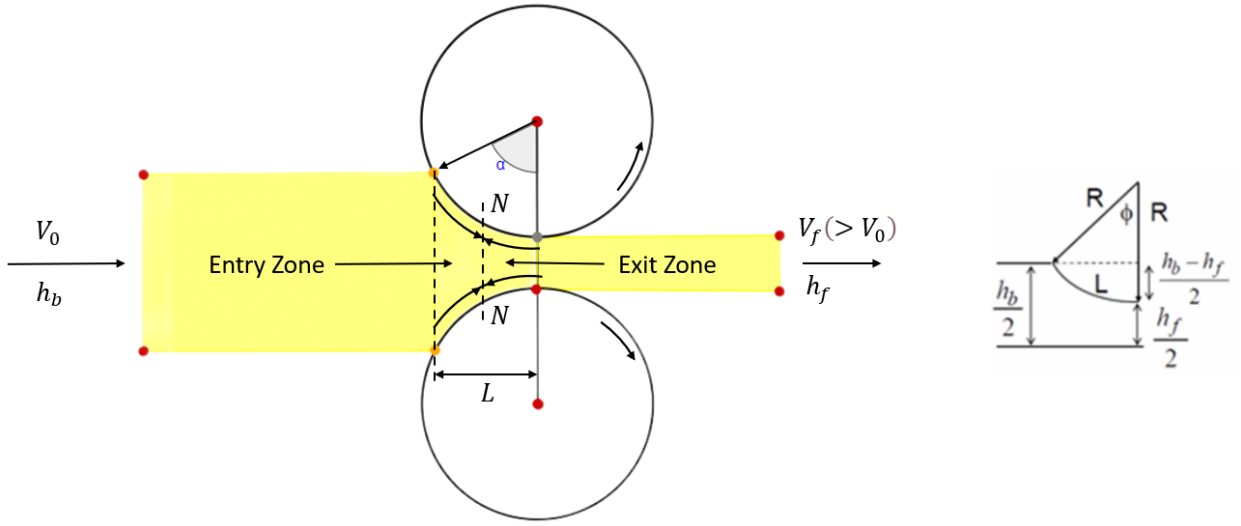


Figure 0.1: Schematic of hot rolling process.

The following Equations (Aghasafari, Abdi, et al., 2014; Aghasafari, Salimi, et al., 2014; Dixit & Narayanan, n.d.; Hirt et al., 2013; Lenard & Pietrzyk, 1992; Nellippallil et al., n.d.) are used to calculate the average flow stress (Y_f), thickness (h), exit pressure (P_f), rolling force (F_{roller}), rolling torque (T_{roller}), and rolling power (P_{roller}). The thickness of the sample, h is a function of the angle, ϕ which designates any point of contact between the sample and the roller surface. The pressure at any location is a function of the thickness.

$$h = h_f + 2R(1 - \cos\phi) \tag{3.1}$$

$$H = 2\sqrt{\frac{R}{h_f}}\tan^{-1}\left(\sqrt{\frac{R}{h_f}}\phi\right) \tag{3.2}$$

$$\varepsilon_f = \ln\left(\frac{h_e}{h_f}\right) \tag{3.3}$$

$$Y_f = k\varepsilon^n = k\frac{\varepsilon_f}{n+1} \quad (3.4)$$

$$p_f = \frac{2}{\sqrt{3}} Y_f \frac{h}{h_f} e^{\mu H} \quad (3.5)$$

$$F_{roller} = 1.15 L w \bar{Y}_f \left(1 + \frac{\mu L}{2h_{ave}}\right) = p_{ave} A \quad (3.6)$$

$$T_{roller} = rF_{roller} = \frac{L}{2} F_{roller} \quad (3.7)$$

$$P_{roller} = T_{roller}\omega = F_{roller}\omega\frac{L}{2} \quad (3.8)$$

Where, R refers to the roller diameter, w is the width of the sample, A is the cross-sectional area, L is the workpiece length, n is the work hardening exponent, k is the strengthening coefficient, omega is the angular frequency depends on the rpm, ε is the average strain and η is the friction coefficient.

Matlab/Simulink is the platform chosen for modeling and simulation of this system. The primary reason behind this choice is that Simulink offers modeling and simulation of nonlinear systems along with the ability to design and analyze control systems. Interested readers are referred to the documentation provided by MathWorks(*Block Libraries - MATLAB & Simulink*, n.d.) to learn more about the blocks used for the design. Figure (3.2) shows the Simulink model of the CEFAS/hot rolling model using all the aforementioned equations. The Y_f block generates average flow stress using which the P_f block calculates the exit pressure. The F_{roller} block takes friction coefficient, average stress flow, average height, width and workpiece length as inputs to generate the roller force. The torque and power are calculated by the T_{roller} and P_{roller} blocks, respectively. The details of the subblocks are shown in Appendix A. The value and unit for different constants used in this model are shown in Table (3.1).

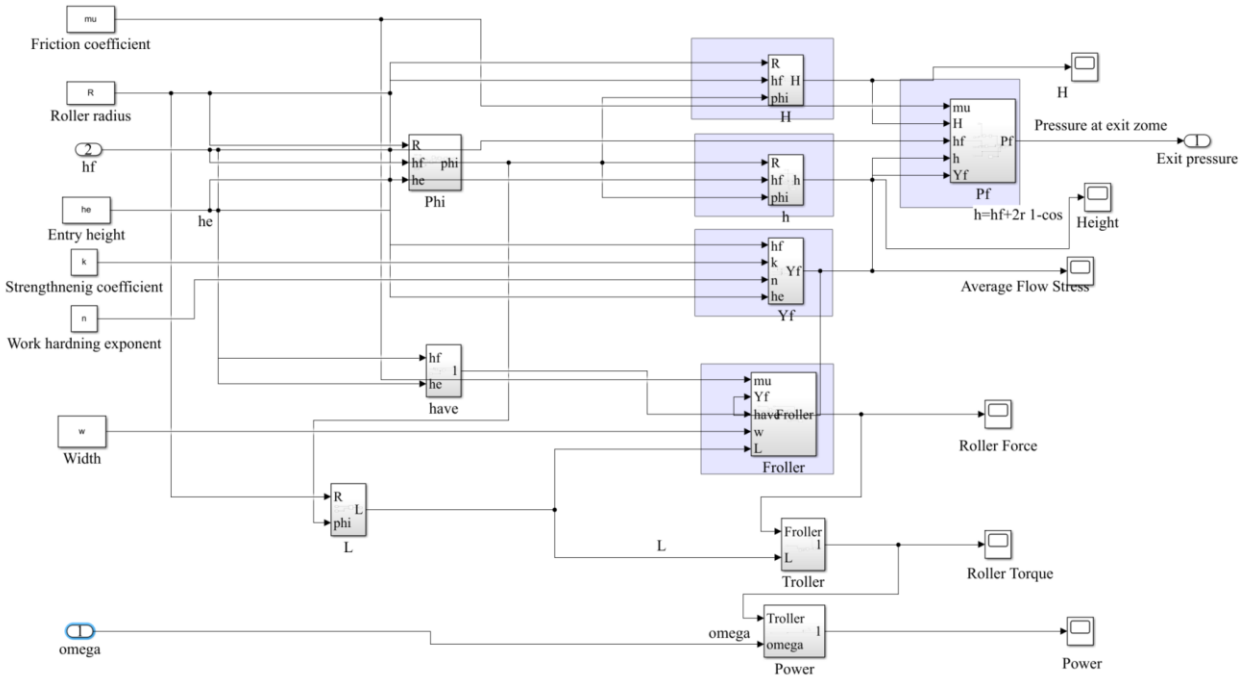


Figure 0.2: Simulink model of the CEFAS/hot rolling system.

Table 0.1 : Parameters for CEFAS/hot rolling model.

Parameter	**Notation**	**Value**	**Unit**
Entry sample height/thickness	h_e	0.05	m
Roller radius	R	0.1778	m
Friction coefficient	η	0.3	-
Initial width	w_0	0.127	m
Final width	w_f	0.18	m
Rolling speed	r	1	rpm

Work hardening exponent	n	0.25	-
Strengthening coefficient	k	500e6	Kg/m^2

3.1.2 Actuator Modeling

The scheme to pressurize the sample when current flowing through it involves an actuator to force the top roller downward. A pneumatic ram was primarily modeled as an actuator. However, CEFAS requires precise pressure control which in some scenarios, pneumatic ram is unable to provide. Therefore, an electric linear actuator i.e., Exlar T2X115 is chosen. To model this, mathematical equations of a typical linear actuator are considered (Antong et al., 2014, 2016; Du et al., 2010). The linear actuator consists of two parts: a DC motor to generate torque and a structure to translate the torque to the load. The motor converts electrical energy into mechanical torque. The motor torque is converted to linear motion using a gearbox and a lead screw. Figure (3.3) shows the simplified equivalent diagram of a linear actuator. The desired force can be achieved by varying the voltage supplied to the motor. R_a denotes the motor armature resistance while L_a is the motor reactance. The term screw lead (l) refers to the linear displacement by the nut after one revolution. The angular motion of the screw is converted to linear motion using the term h which is shown by Equation (3.9). Here, X_a and X_L are the linear displacements of the actuator and the load, respectively. The screw damping, C and the screw stiffness, K are accounted to calculate the working displacement on the load.

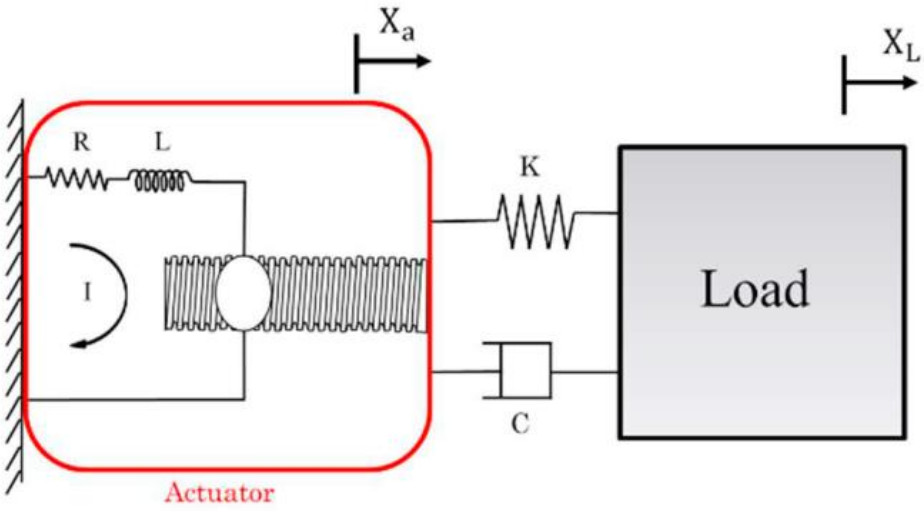


Figure 0.3: Equivalent model of a linear actuator with load (Antong et al., 2014).

The following equations (Du et al., 2010) are used to model the linear actuator system.

$$h = \frac{l}{2\pi} \tag{3.9}$$

$$I' = \frac{1}{L_a}\left[V_s - R_a I - K_e \theta'_m\right] \tag{3.10}$$

$$\theta''_m = \frac{1}{J}\left[K_t I + K_m\left(\theta_m h - X_a\right) h + D\theta'_m\right] \tag{3.11}$$

$$X''_L = \frac{1}{M_L}\left[K_s\left(X_a - X_L\right) - C_s\left(X'_a - X'_L\right)\right] \tag{3.12}$$

$$X''_a = \frac{1}{M_s}\left[-K_m\left(X_a - n\theta_m\right) - K_s\left(X_a - X_L\right) - C_s\left(X'_a - X'_L\right)\right] \tag{3.13}$$

Where, I is the armature current, θ_m represents the angular rotation of the motor shaft, K_t is the torque constant, K_e is the voltage constant and Km is the motor stiffness coefficient, J is the motor inertia, D is the damping coefficient, M_L is the load mass and M_s is the screw mass. To facilitate the control design a state-space form of this system is also extracted. The seven states

considered here are current, angular position, angular velocity, screw position, screw velocity, load position and load velocity. The supply voltage, V_s and the load position are considered as input and output for the single input single output (SISO) system. The meaning, value and unit of the parameters and states are provided in Table (3.1).

$$x = \begin{bmatrix} I & \theta_m & \dot{\theta}_m & X_a & \dot{X}_a & X_L & \dot{X}_L \end{bmatrix}^T \tag{3.14}$$

$$A = \begin{bmatrix} -\frac{R_a}{L_a} & 0 & -\frac{K_e}{L_a} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{K_t}{J} & -\frac{h^2 K_m}{J} & -\frac{D}{J} & \frac{hK_m}{J} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{hK_m}{M_s} & 0 & -\frac{K_s + K_m}{M_s} & -\frac{C_s}{M_s} & \frac{K_s}{M_s} & \frac{C_s}{M_s} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{K_s}{M_L} & \frac{C_s}{M_L} & -\frac{K_s}{M_L} & -\frac{C_s}{M_L} \end{bmatrix}; B = \begin{bmatrix} \frac{1}{L_a} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}; \tag{3.15}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Table 0.2: Parameters and States for linear actuator model

Parameter & States	**Notation**	**Value**	**Unit**
Supply voltage	V_s	-	V
Armature inductance	L_a	0.0064	H
Armature resistance	R_a	2.2	Ohm
Equivalent viscous friction	D	8e-5	$Nm/rads^{-1}$

Motor inertia	J	5.3e-5	Kg/m^2
Motor back emf constant	K_e	0.121	$V/rads^{-1}$
Motor torque constant	K_t	0.121	Nm/Amp
Screw lead	l	2.4e-3	m
Load mass	M_L	2000	Kg
Screw damping	C_s	1.2e3	N/ms^{-1}
Screw stiffness	K_s	1.8e5	N/m
Motor stiffness	K_m	1e7	N/m
Screw mass	M_s	2	Kg
Armature current	I	-	Amp
Screw position	X_a	-	m
Angular rotation	θ	-	rad
End of actuator position	X_L	-	m

The developed Simulink model of the linear actuator is shown in Figure (3.4). The details of the four subsystem blocks are depicted in Appendix A. The Exlar T2X115 model has the option to control either the position, velocity, or torque. Therefore, the first principle model of the linear actuator is also designed to have all those three options as outputs.

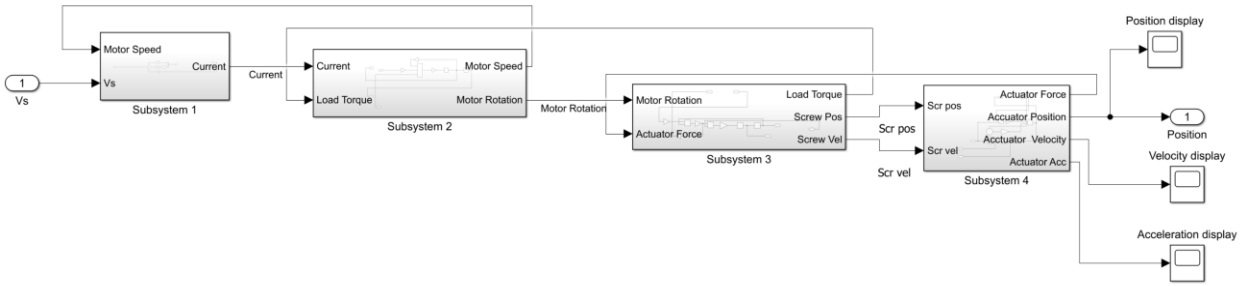


Figure 0.4: Simulink model of the linear actuator.

3.2 Models using System Identification

System identification is a data-driven technique which requires to design good input. To identify INL's CEFAS model, which is designed using COMSOL Multiphysics®, multiple experiments are outlined. This section describes the experiments, and some models identified.

3.2.1 Experiments Design

A total of four experiments are proposed to extract a dynamic model including the estimation of other parameters such as roller speed, joule heating. For each of these experiments, collection of timestamped measurements of pressure data in the material is proposed. At first, five zones are selected for data measurement with each of them having five vertical points. However, some initial experiments show that pressure values in data zones with greater distance from the neutral point of the roller don't change with linear actuator movements. Therefore, a

revised experiment setup is proposed which is depicted in Table (3.3). These three data zones will be equally distributed, and the first and last point of each zone will be on the surface of the sintered material. Considering the entry zone as (0,0), the other two data zones are (-1,0), which is 1 inch before the entry zone, and (1,0), which is 1 inch after the entry zone. Figure (3.5) shows a visual representation of the data measurement zones. The results of the initial experiments are shown in Figure (3.6)

Table 0.3 : Experiments for system identification.

	Roller speed (rpm)	LA movement	Joule heating	Output
Experiment 1	Constant (1 rpm)	Varies	Constant (600K/min)	3 horizontal zones, each with at least 2 vertical points
Experiment 2	Varies	Constant	Constant (600K/min)	3 horizontal zones, each with at least 2 vertical points
Experiment 3	Varies	Varies	Constant (600K/min)	3 horizontal zones, each with at least 2 vertical points
Experiment 4	Varies	Varies	Varies	3 horizontal zones, each with at least 2 vertical points

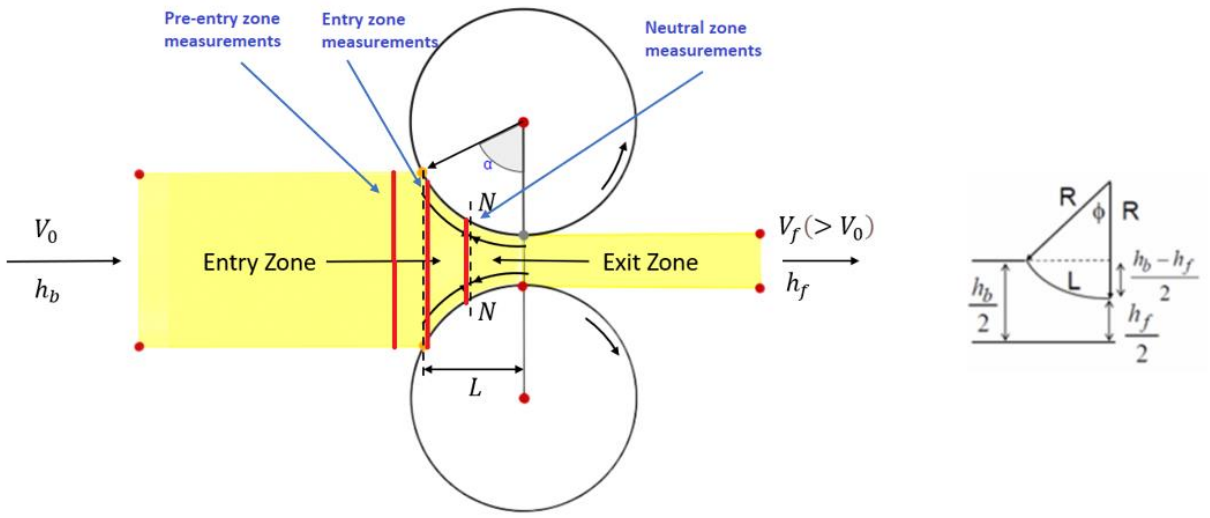


Figure 0.5: Selected zones for data extraction.

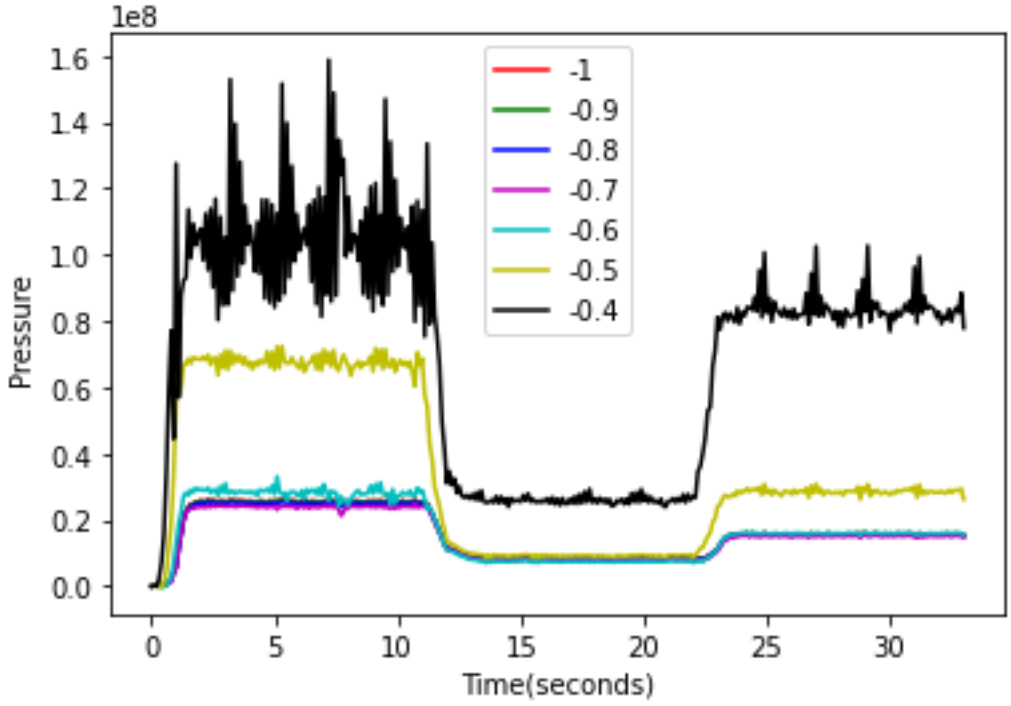


Figure 0.6: Pressure variation with respect to linear actuator movements at different horizontal locations on the sample.

The linear actuator is planned to move during the first experiment from one extreme point to the other (maximum and minimum) as well as randomly selected intermediate points producing a random set of different sample heights. Using the same setup as the first experiment, instead of varying the linear actuator, the roller speed varies between 0 and 4 rpm randomly, while keeping the linear actuator at the nominal sample height settings in the second experiment. The third experiment is a combination of the first two experiments where both the linear actuator and the roller speed varies randomly while keeping the joule heating constant. The fourth experiment is designed to interface the joule heating dynamics with pressure and rolling dynamics.

For the first experiment, a pseudo random binary input (PRBS) is designed. However, the current COMSOL model doesn't support pseudo random movements as an input. To resolve this issue, the entire system is divided into two sequential processes: the first one is the linear actuator, represented by the Simulink model, and the second one is the COMSOL model that captures what happens with the material being sintered. The linear actuator model receives the designed PRBS input and generates an output. This output is no longer a pseudo random sequence but will be altered by the dynamics of the actuator. This output will be collected as a time series data with specific sampling time. This time series data will be provided to be used as the linear actuator movements and hence generates the resulting pressures in the sintered material. A system identification on the entire system will be performed using the PRBS input data and the output data generated from the COMSOL model.

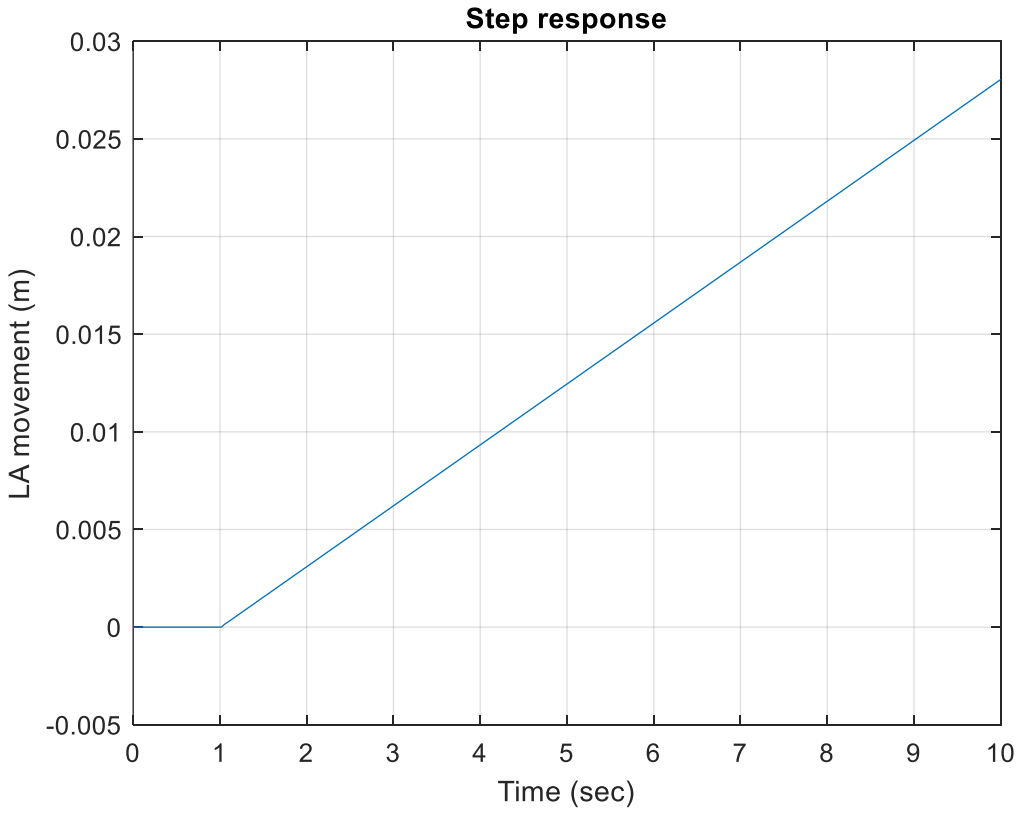


Figure 0.7: Step response of the linear actuator block.

To design a PRBS input it is required to know the system response time. The step response of the linear actuator shown in Figure (3.7) doesn't have a settling time as it shows a linear increment or decrement when positive or negative voltage is applied. Therefore, the entire model is regulated with a PID controller to find out the settling time. The Exlar T2X115 model has a maximum stroke length of 6 inches or 0.1524 m. A movement limiter block is used to keep the linear actuator movement within the 0–6-inches range. A gain is added to help creating an output which covers the whole range. The specifications to generate the PRBS input are listed below.

- Settling time: 0.7 sec
- Band: 20% to 300% of settling time

- Sampling time: 0.1 sec
- Max. stroke length: 6 inches
- Min. Stroke length: 0 inch
- PRBS peaks: +/- 10V
- Runtime: 100 sec

Figure (3.8) shows the schematic of data generation block which uses a PRBS input to generate linear actuator movements. Figure (3.9) shows the generated PRBS input and the linear actuator output.

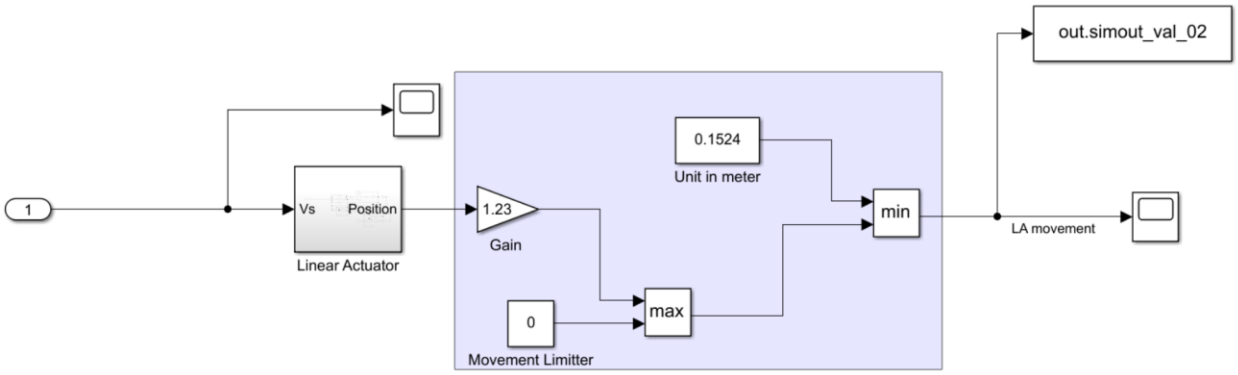


Figure 0.8: Schematic of linear actuator data generation block.

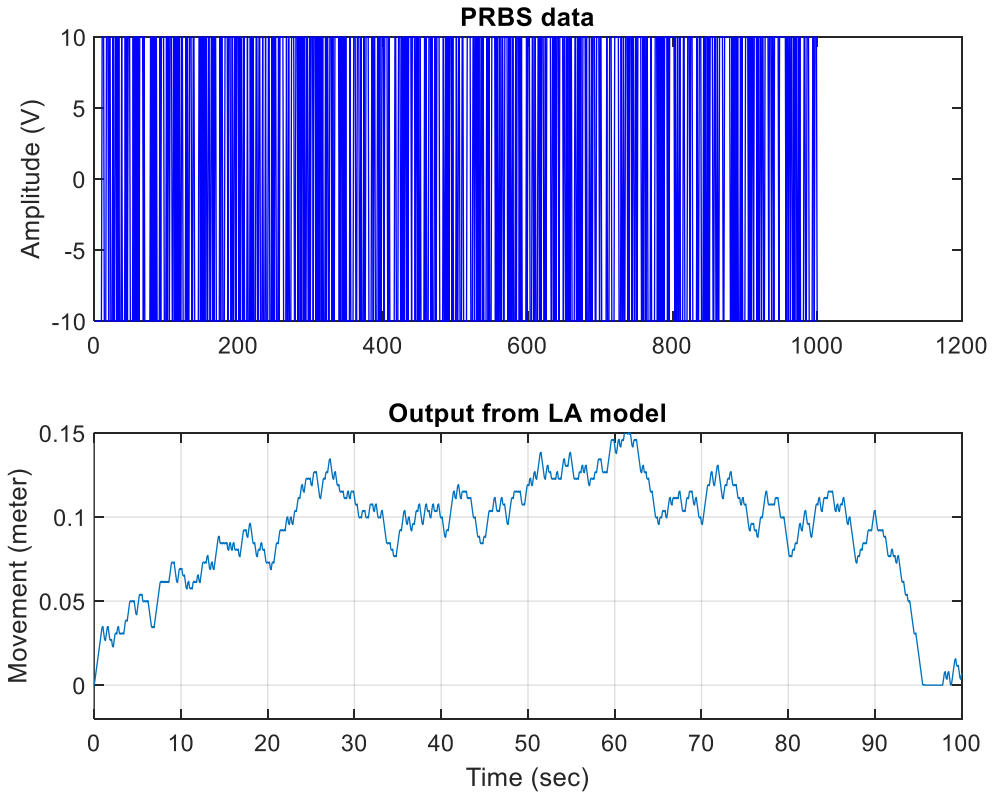


Figure 0.9: Linear actuator output for the generated PRBS input.

3.2.2 Results using System Identification

The linear actuator movements are used to generate pressure data on the sample at the neutral point and at point (-1.25,0) which is just before entering the workpiece. These data points are plotted in Figure (3.10) along with the linear actuator output. The plot shows a surge in pressure at the neutral zone along with a lot of fluctuations.

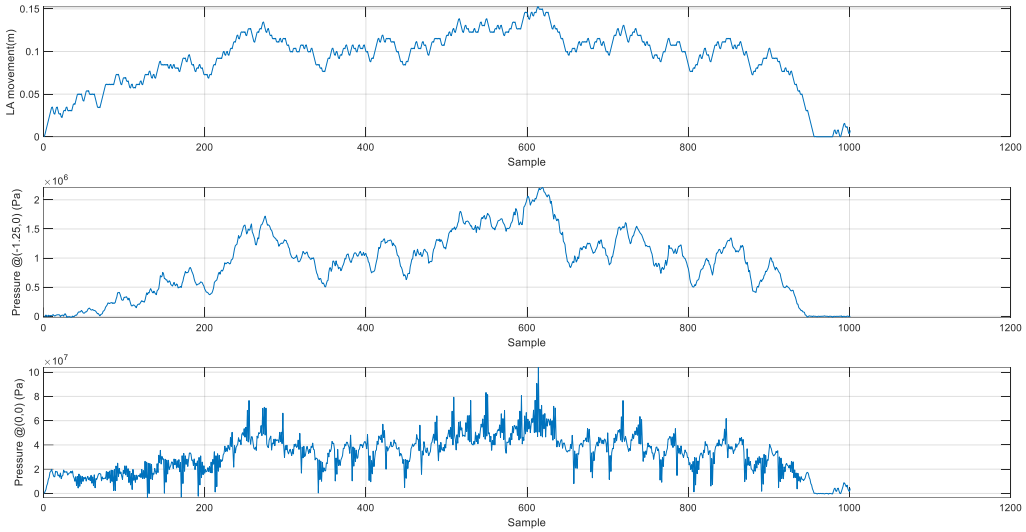


Figure 0.10: Pressure at (0,0) and (-1.25,0) points using the linear actuator movement as input.

The MATLAB system identification toolbox is utilized to investigate several transfer functions to identify models with good fits to the data. Each of the two datasets is divided into an estimation and a validation dataset. The estimation dataset consists of 800 datapoints and the validation dataset has 201 datapoints. A closer look into the datasets reveals that although the actuator moves back to its initial position, there are negative pressure values appearing at some points. This can force the identified system to capture inaccurate dynamics. Therefore, some additional data pre-processing is performed to endorse explainability.

Both (0,0) and (-1.25,0) data zones are used to estimate models. Estimation data for (0,0) data zone is referred as ‘data00_test’ while validation data is denoted as ‘data00_val’. Similarly, estimation and validation datasets for (-1.25,0) data zone are referred as ‘data_test’ and ‘data_val’, respectively. The naming convention for the estimated transfer functions is ‘**tfmpz**’. Here, **tf** means transfer function, **m** refers to data zone (0 for (0,0) and empty for (-1.25,0)), **p** refers to the number of poles, and **z** refers to the number of zeros in the transfer function.

Using the data_test and data00_test several transfer function models are estimated. Some of the good performing models using data00_test and data00_val are tf071, tf043, tf051, and tf011. Good performing models estimated using data_test and data_val are tf71, tf43, tf73, tf11, tf51, and tf41. The plots in Figure (3.11) and (3.12) show the simulated/predicted outputs of the estimated transfer function models where the validation data set is plotted in grey.

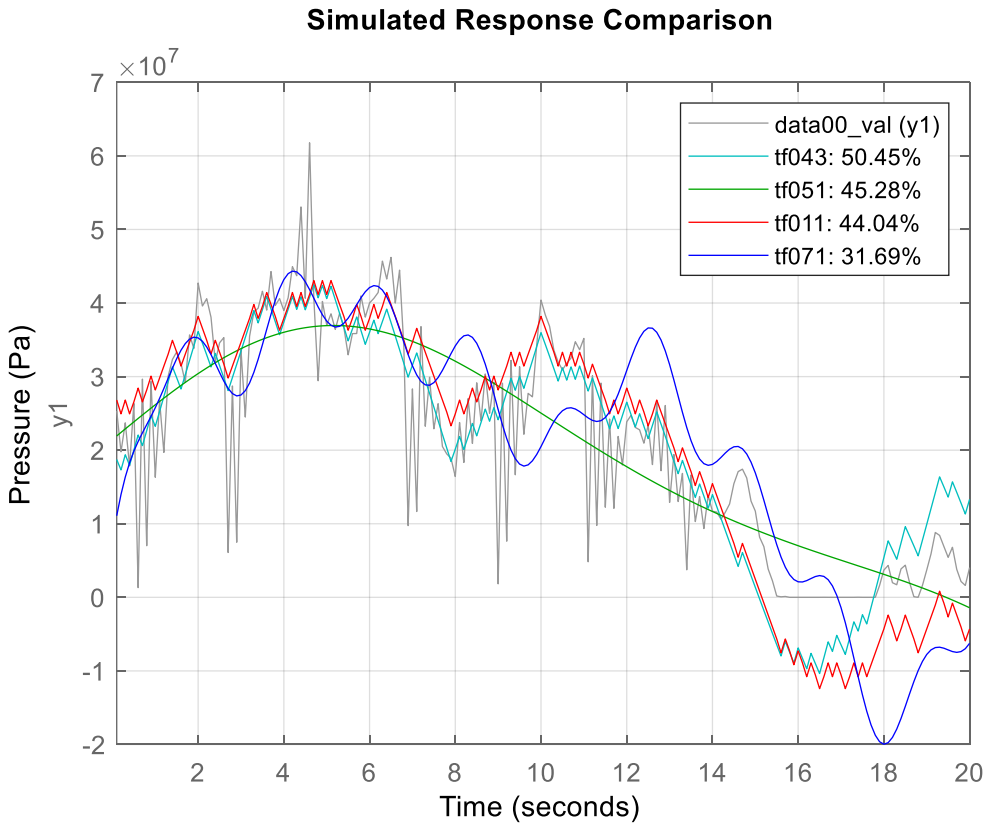


Figure 0.11: Performance comparison for different models estimated using data00_test.

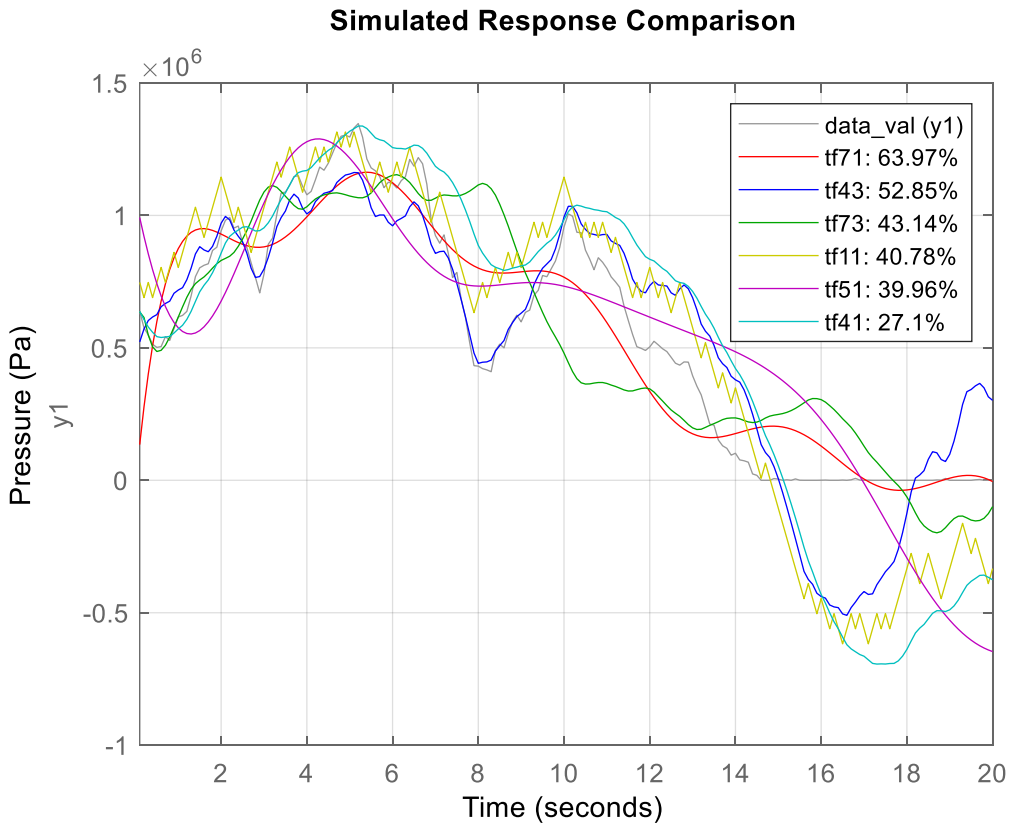


Figure 0.12: Performance comparison for different models estimated using data_test.

Looking into the results, models with higher number of poles tends to perform better. This is an indication the studied data represents a higher order system. Further investigations are performed to see how these models behave when validated against different data zones. Figure (3.13) and (3.14) shows a cross-validation performances of the selected models.

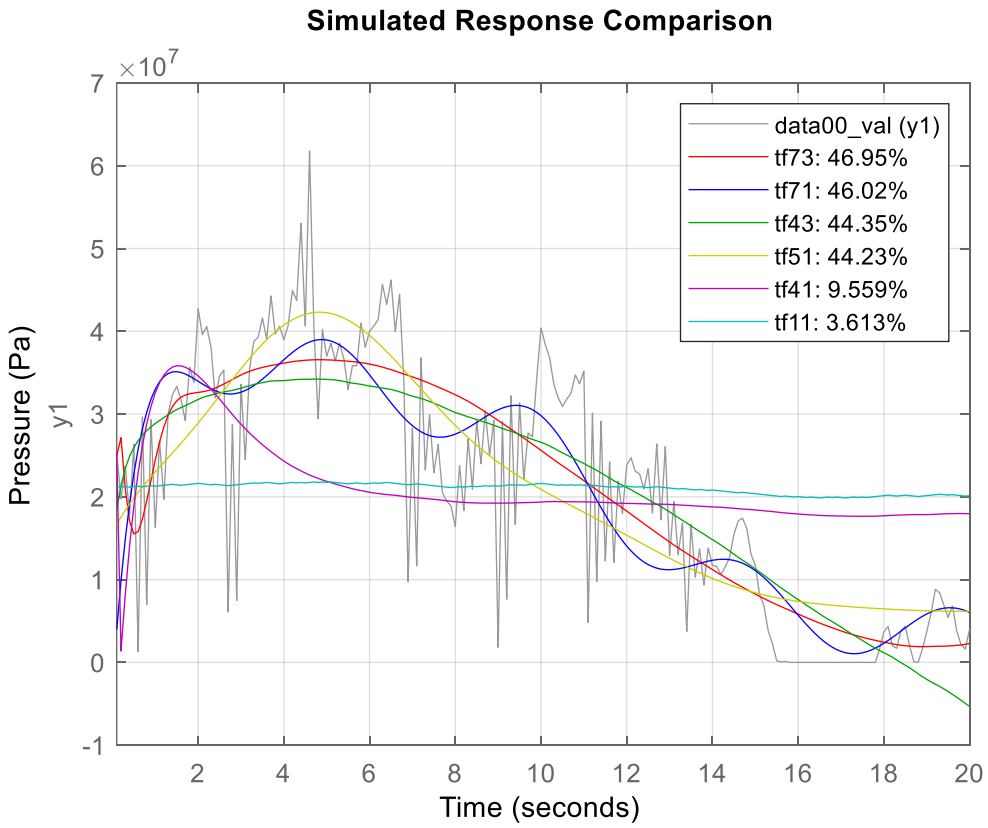


Figure 0.13: Cross-validation for models estimated from data_test.

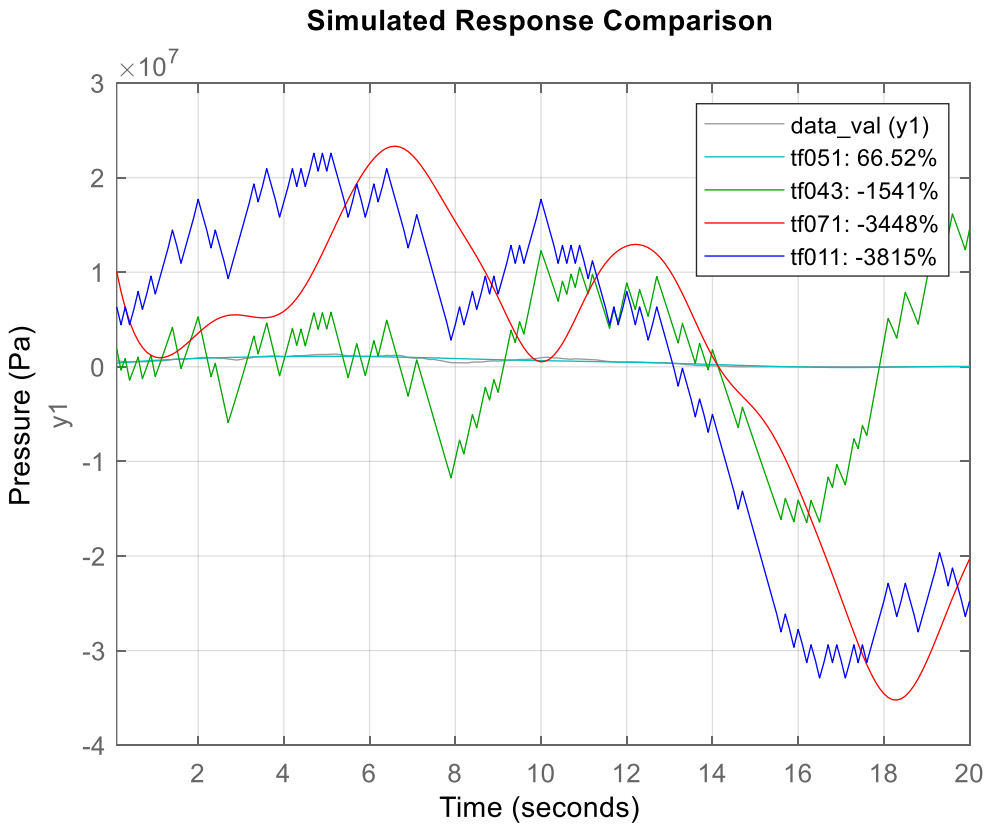


Figure 0.14: Cross-validation for models estimated from data00_test.

The cross validation yields two best candidates from each estimation dataset which are tf051 and tf71. Both these candidates show around 65% fit to the data_val dataset and 45% fit to the data00_val dataset. As the (0,0) data zone shows a lot of fluctuations, a lower fitting percentage using data00_val is expected. Figure (3.15) shows the cross-validation performances for the best candidates. Transfer functions of these two models are shown in Table (3.4) and are considered for the primary data-driven controller investigation.

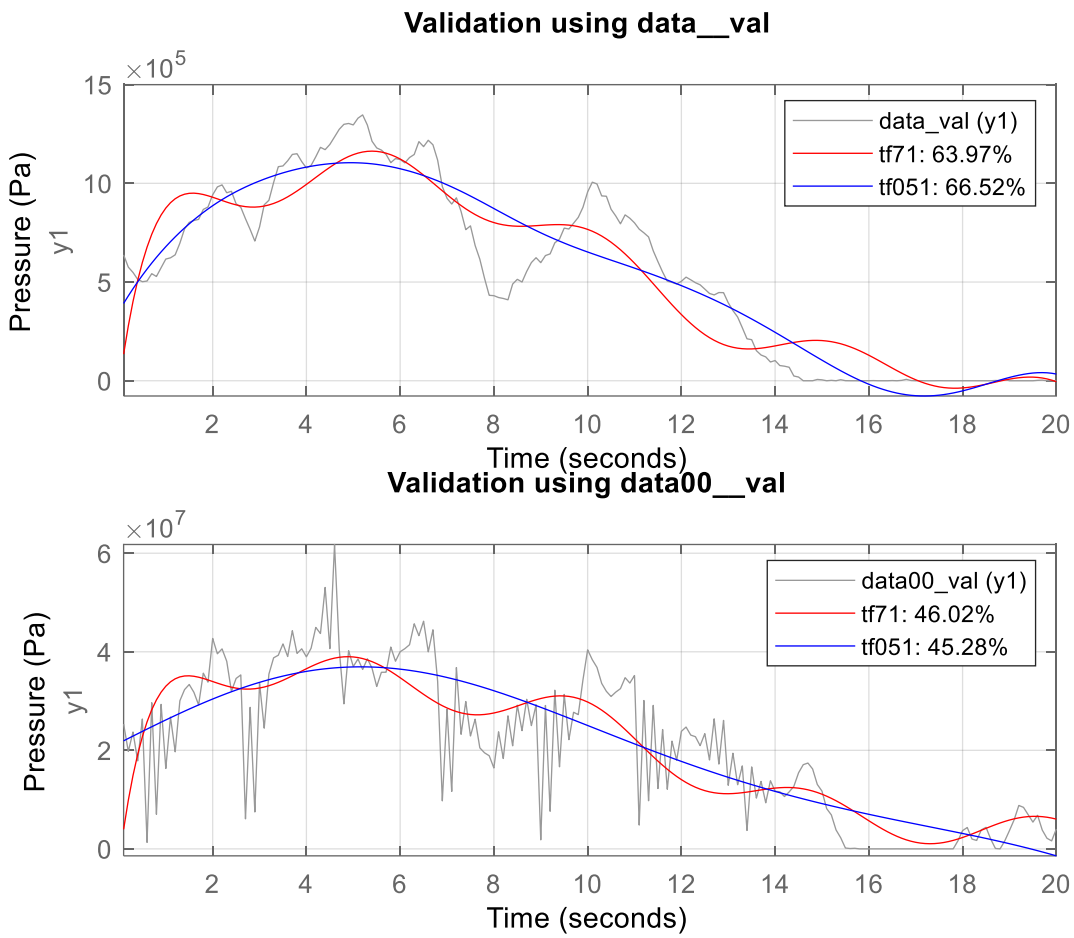


Figure 0.15: Cross-validation performances for the best candidates.

Table 0.4 : Best performed models.

Transfer function name	Fit to estimation	Transfer function	FPE	MSE
tf051	24.51%	$\frac{7670s - 1187}{s^5 + 0.12s^4 + 0.15s^3 + 2 \times 10^{-3}s^2 + 8.2 \times 10^{-4}s + 7.5 \times 10^{-6}}$	1.44e14	1.39e14
tf71	42.51%	$\frac{1625s - 1295}{s^7 + 1.47s^6 + 2.3s^5 + 2.7s^4 + 0.95s^3 + 0.54s^2 + 0.04s + 2.02 \times 10^{-4}}$	9.75e10	9.37e10

Chapter 04: Simulation

4.1 Stability, Observability & Controllability Analysis

A system is said to be stable if it produces a bounded output for a bounded input (Astrom, 1995). This is an important criterion for regulator design as it allows the system to reach and stay at the steady-state value. For a system to be stable, the poles of the transfer function should remain in the left half s-plane. All the control schemes in this work, are designed based on three open-loop systems. The first one is the first principle model which has a transfer function with seven poles and one zero. The other two open-loop systems are the transfer function models identified from the system identification experiment. All these systems have poles with negative real parts, which makes all of them stable. The pole zero plots for these systems are shown in Figure (4.1) and (4.2). In these figures, poles and zeros are referred by crosses and circles, respectively.

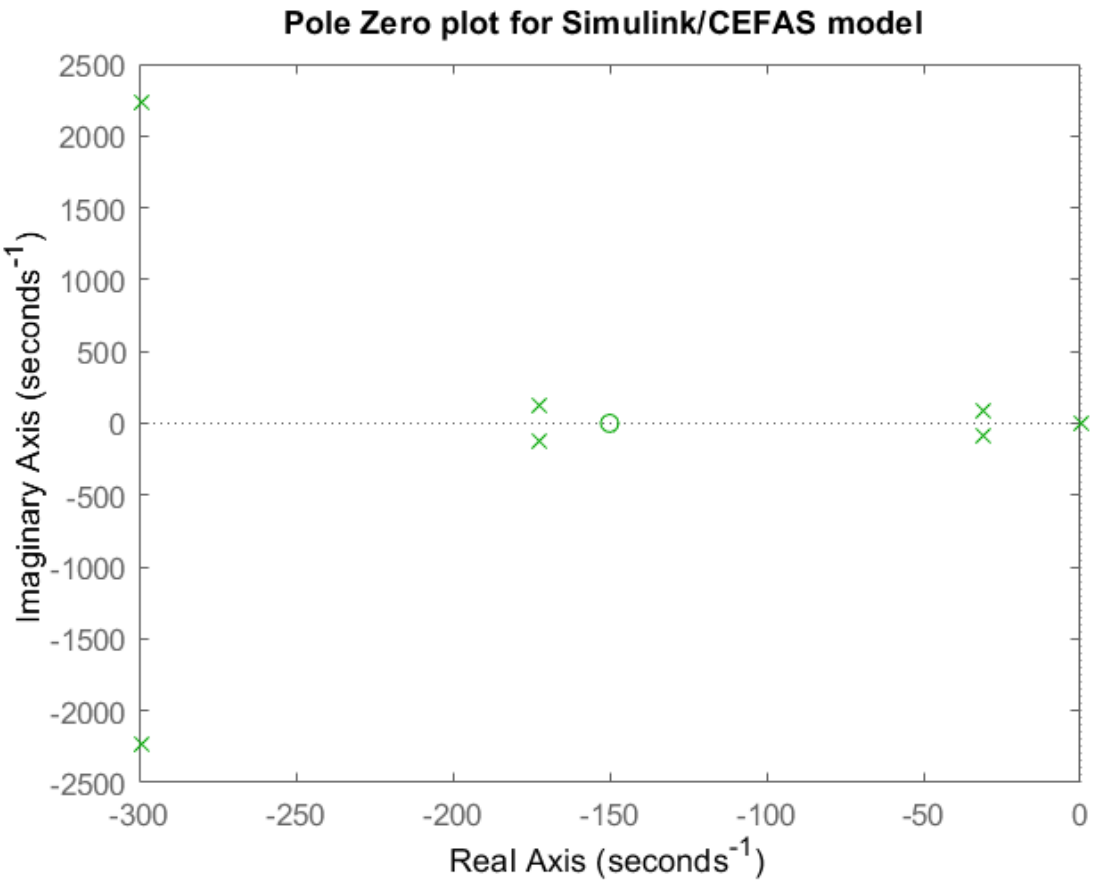


Figure 0.1 : Pole-Zero map for the Simulink/CEFAS model.

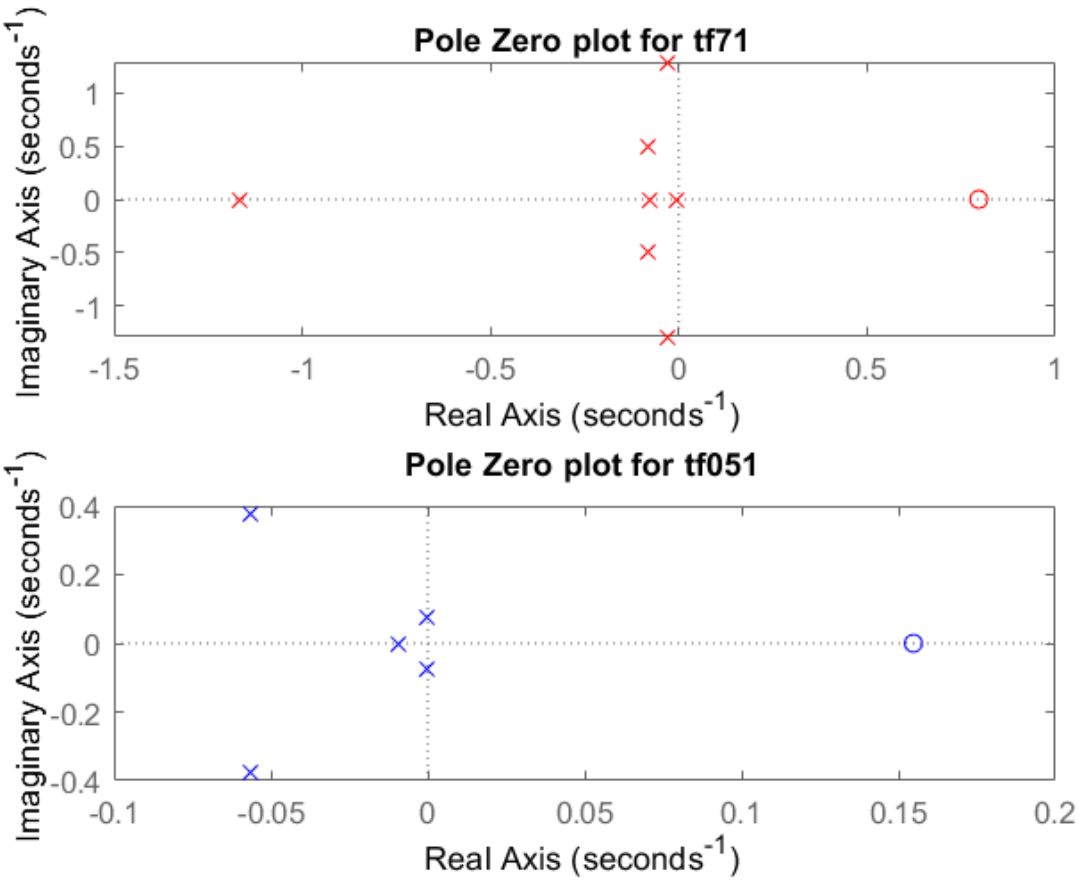


Figure 0.2 : Pole-Zero map for identified CEFAS models.

If a system is controllable, then there exists a control input that can drive the system to any desired state in a finite amount of time (Weiss, 2010). Moreover, the closed-loop poles of the system can be placed arbitrarily anywhere by altering the gain matrix. If a system is not controllable, then it is not possible to design a controller to regulate that system. On the other hand, a system is called to be observable if the initial states of that system can be determined by observing the output of the system in a finite amount of time. If a system is not observable, then designing a full-state feedback controller is impossible.

A quick check to determine if a system is controllable and observable is to see the rank of the controllability and the observability matrices, respectively. If both matrices are full rank, then the system is both controllable and observable. However, this technique fails to identify the uncontrollable or unobservable states. To identify the uncontrollable or unobservable states, the invariant subspace method can be used. This method also provides the information on which states are more controllable or observable than others. All the three considered models are checked for controllability and observability. The models identified from the system identification experiment are both controllable and observable. However, although controllable, the model derived using first principle method lacks observability.

4.2 PID Controller Design

A PID controller is designed to regulate the plant by utilizing the first principle method. The following is a description of developing the corresponding Simulink model and is depicted in Figure (4.3). A voltage limiter block is added before the linear actuator block to keep the PID control signal within the +/- 10V range. To restrict the linear actuator movement within its 0-6 inches range, a position limiter block is added. The movement of the actuator subtracts from the entry sample thickness to generate the final thickness of the sintered material. The LA block represents the linear actuator while the static system block represents the hot rolling model. The step response of the closed-loop system is shown in Figure (4.4).

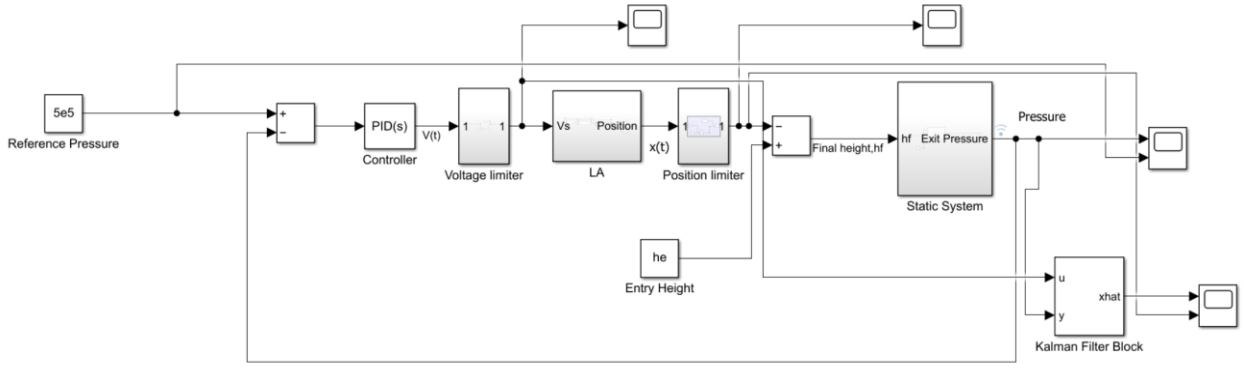


Figure 0.3: Simulink block diagram for PID controller.

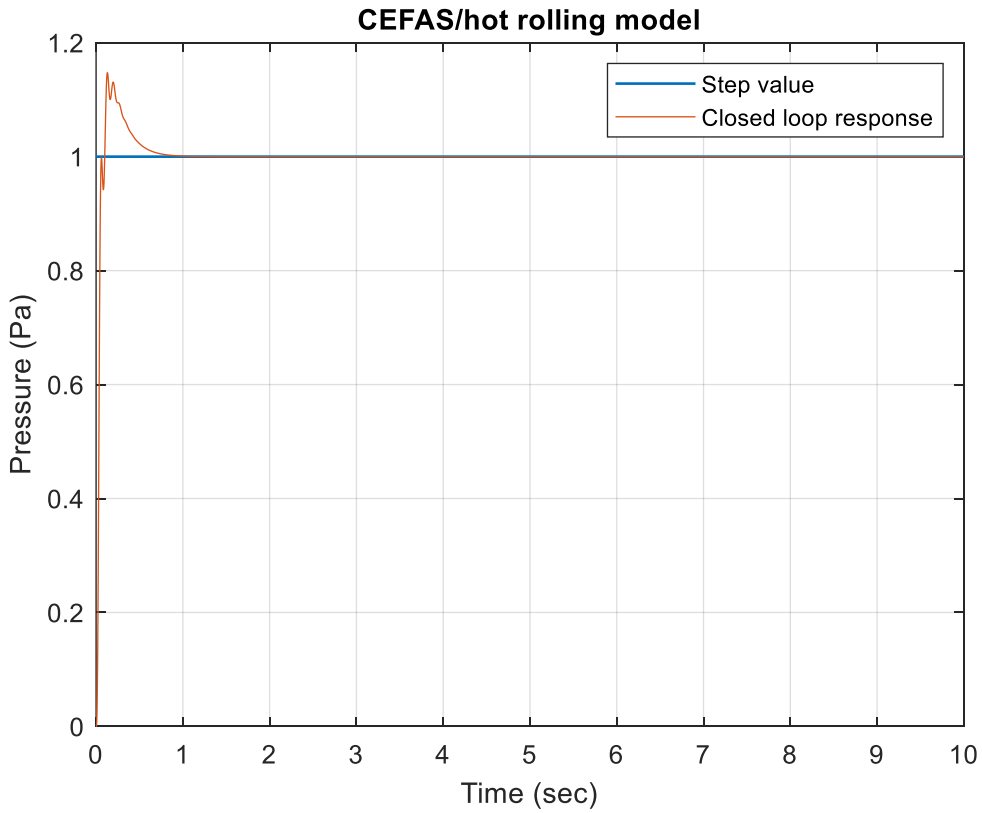


Figure 0.4: Step response of the closed-loop system.

The parameters for the PID controller are tuned using the MATLAB/PID Tuner app. Benefits of using the PID Tuner app is that it always returns a stable controller, even if some

gains are negative. As long as the numerator coefficients of the PID controller transfer function are positive, a negative gain will yield a stable controller. The controller parameters along with the performance and robustness information for the designed PID controller are listed in Appendix A.

A PID control scheme is also developed for the other two transfer function models i.e. Table (3.4). Figure (4.5) shows the closed-loop system for both of the models. For tf051, there is no derivative term, therefore, it is a PI controller. However, the closed-loop step response for tf051 is unstable and shown in Figure (4.6). On the other hand, Figure (4.7) shows the closed-loop step response for tf71 with a tuned PID controller. Although this is a stable response, the settling time implies the sluggishness of the system. The details of the parameters of the controllers can be found in Appendix A.

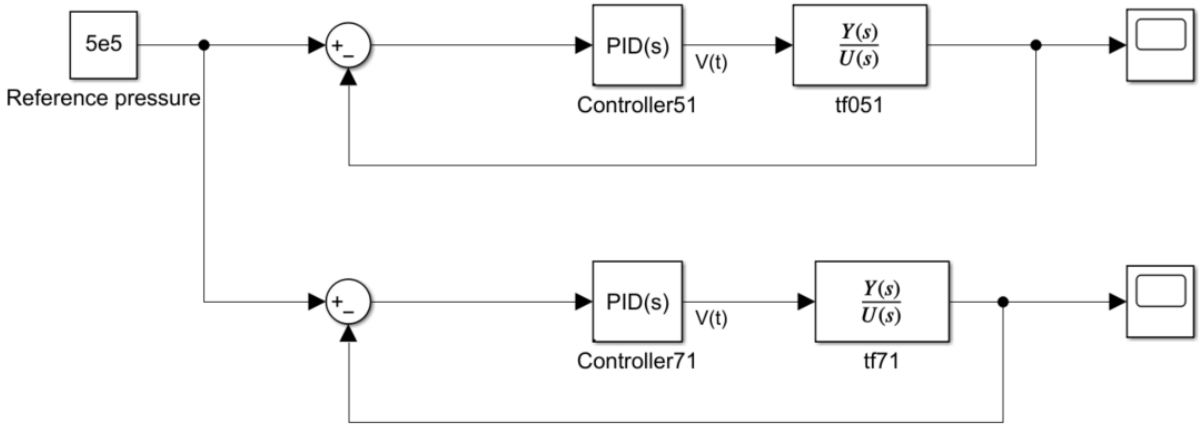


Figure 0.5: PID control scheme for the transfer function models.

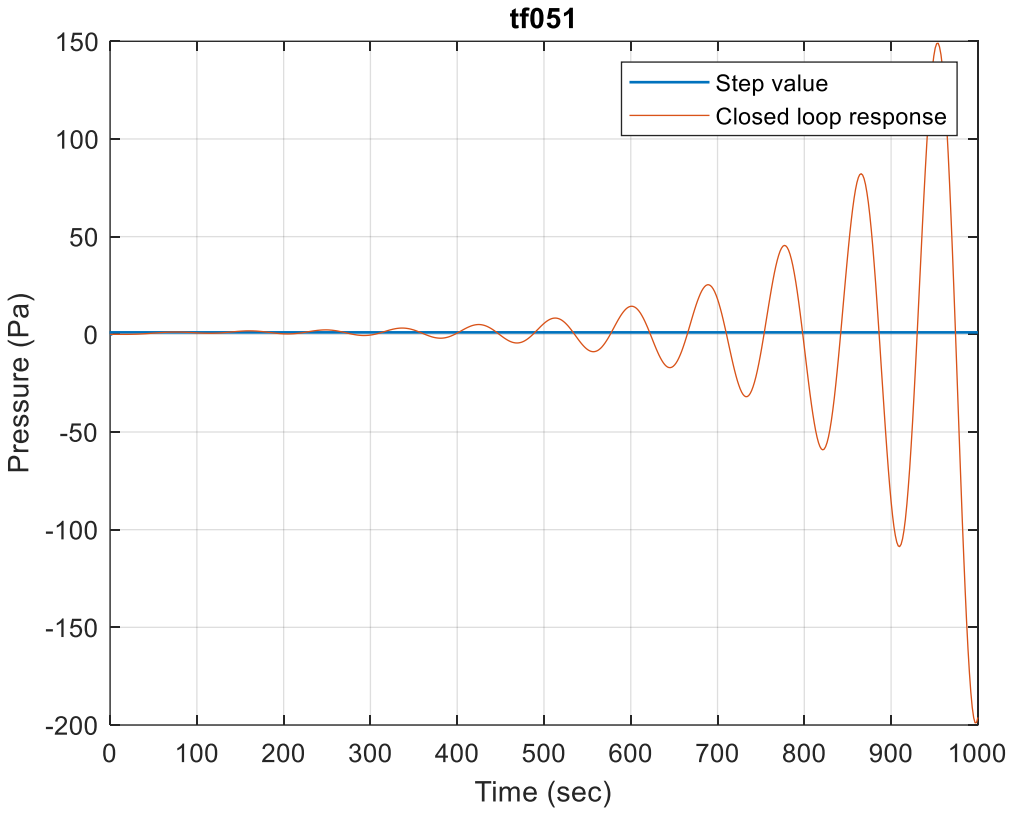


Figure 0.6: Closed-loop step response for tf051 using PI controller.

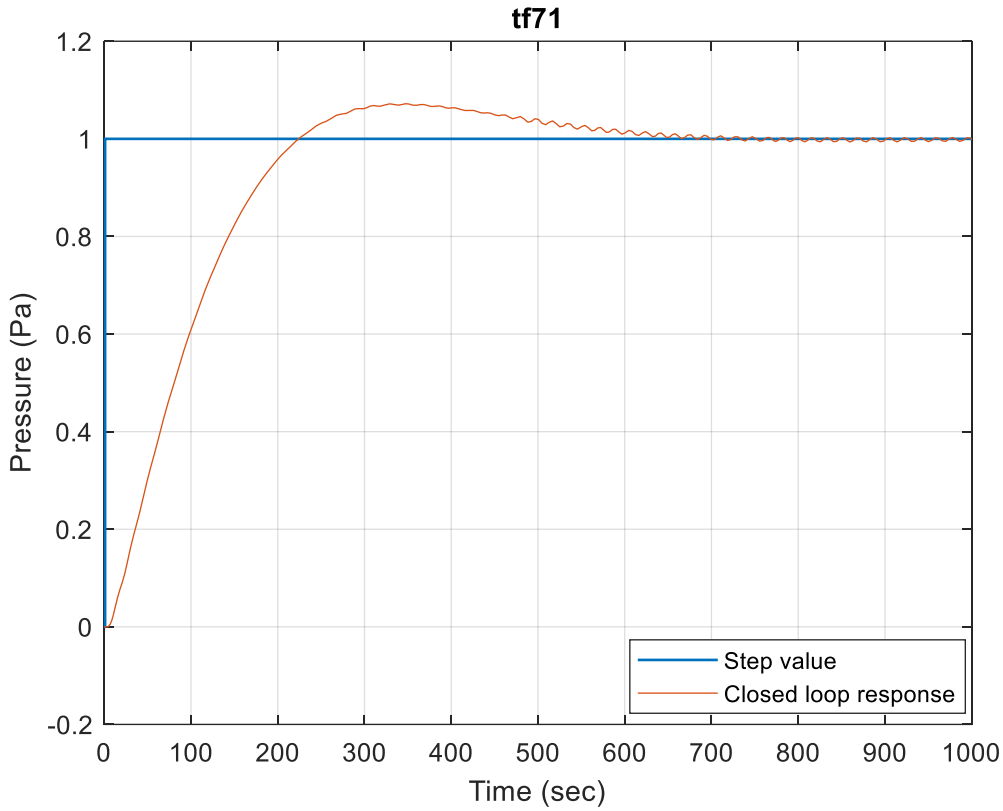


Figure 0.7: Closed-loop step response for tf71 using PID controller.

4.3 LQG Controller Design

For the CEFAS/hot rolling model, all the initial states are not fully observable from the measured output. Therefore, a full-state feedback controller such as LQG is not applicable for this design. However, the other two transfer function models have full observability. Hence, it is possible to design a full-state observer such as a Kalman filter to estimate the unmeasured states and a full-state feedback regulator such as LQR. This is known as the separation principle which states that the state estimator and the state feedback can be designed independently and

combined to build an LQG. The Q and R matrices are designed such that the actuator effort is reduced while critical state errors get penalized heavily. Figure (4.8) shows the general LQG control scheme for both transfer function models where the Kalman filter block estimates all the states and the LQR gain acts as the full-state feedback regulator. Parameters and details of the design can be found in Appendix A.

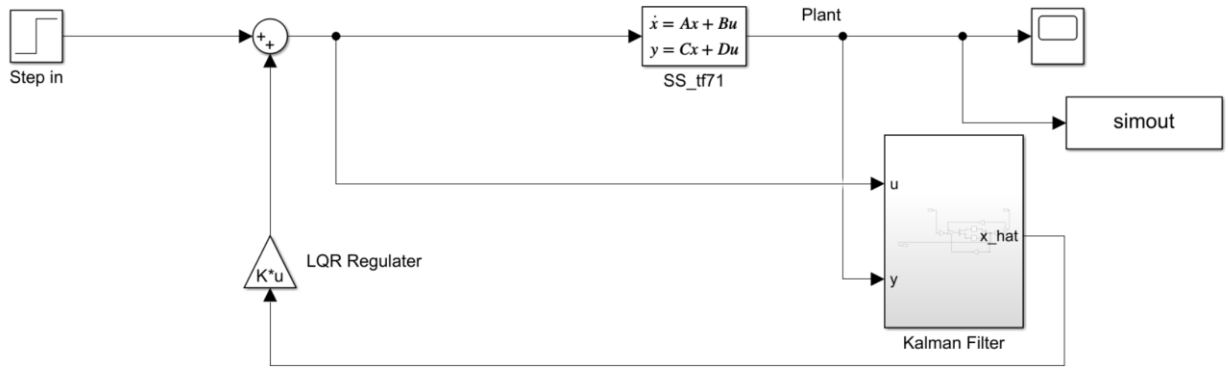


Figure 0.8: LQG control scheme for fully observable systems.

Closed-loop responses for both tf71 and tf51 are shown in Figure (4.9) and (4.10), respectively. Although the PI controller can't produce a stable response for the tf51 model, LQG is able to stabilize it. However, there is a steady-state error, which will be discussed later.

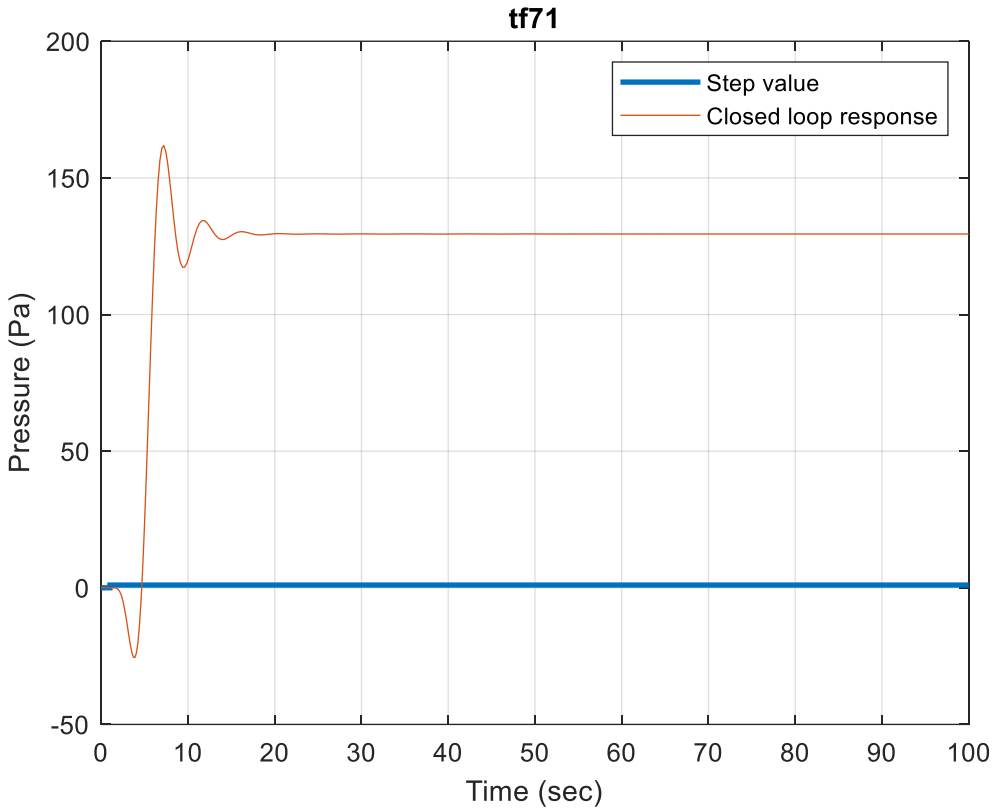


Figure 0.9: Closed-loop step response for tf71 using LQG controller.

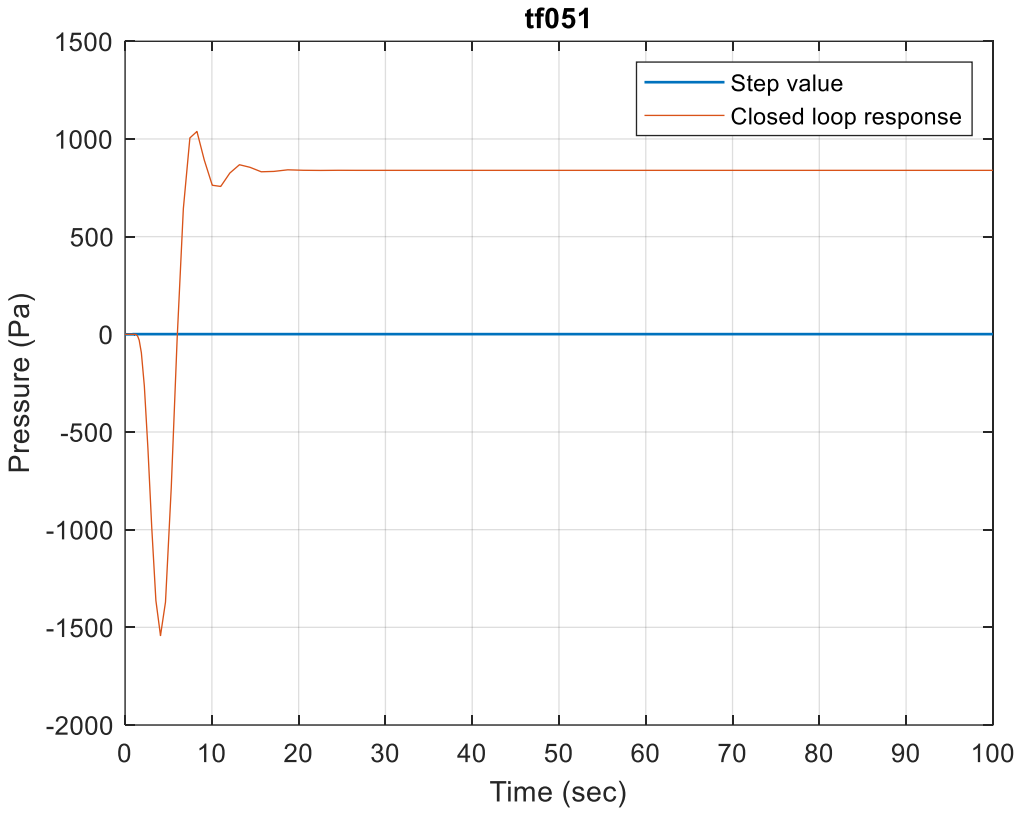


Figure 0.10: Closed-loop step response for tf51 using LQG controller.

4.4 Reinforcement Learning Controller Design

MATLAB/Reinforcement Learning Toolbox® is utilized to develop a RL algorithm with DDPG agent to regulate the considered models. For the environment, CEFAS/hot rolling and the two identified models are used. To train the DDPG agent, it is required to define action and observation objects. The action signal for this RL algorithm is voltage. Three observations are defined for this environment which includes error, pressure, and integrated error. The error signal is the difference between the output pressure and the desired pressure. The bounds for the

observations are also defined. Next, a scalar reward signal is created which is shown in Equation (4.1).

$$Reward = [-a|error| + b] + \left[-c\left(d - \left|\frac{dP}{dt}\right| > 0\right) + e\left(d - \left|\frac{dP}{dt}\right| \leq 0\right)\right] + \left[max\left(-h\int|error|dt\ ,\ -k\right)\right] - 100(P \geq f\ ||\ P < g) \quad 4.1$$

Here, *a, b, c, d, e, f, g, k,* and *h* are positive numbers. These values can be tuned depending on the learning progress of the DDPG agent. The first part of the reward function punishes and rewards the error using *a* and *b*. The second part rewards or punishes the rate of change of pressure using *c* and *e*. The third part punishes the cumulative error by tuning *h*. The final part implements a large reward penalty if the pressure is outside the *f* to *g* range. A termination signal is designed when the output is out of bounds. To randomize the reference signal, a custom reset function is created. Figure (4.11) shows the designed RL control scheme for the CEFAS/hot rolling system. The observation block in the figure feeds the three observations to the DDPG agent. The reward calculation block calculates the reward based on the action taken and informs the DDPG agent. The termination block terminates the episode when the output goes out of bound. A gain block is added after the environment to scale down the output.

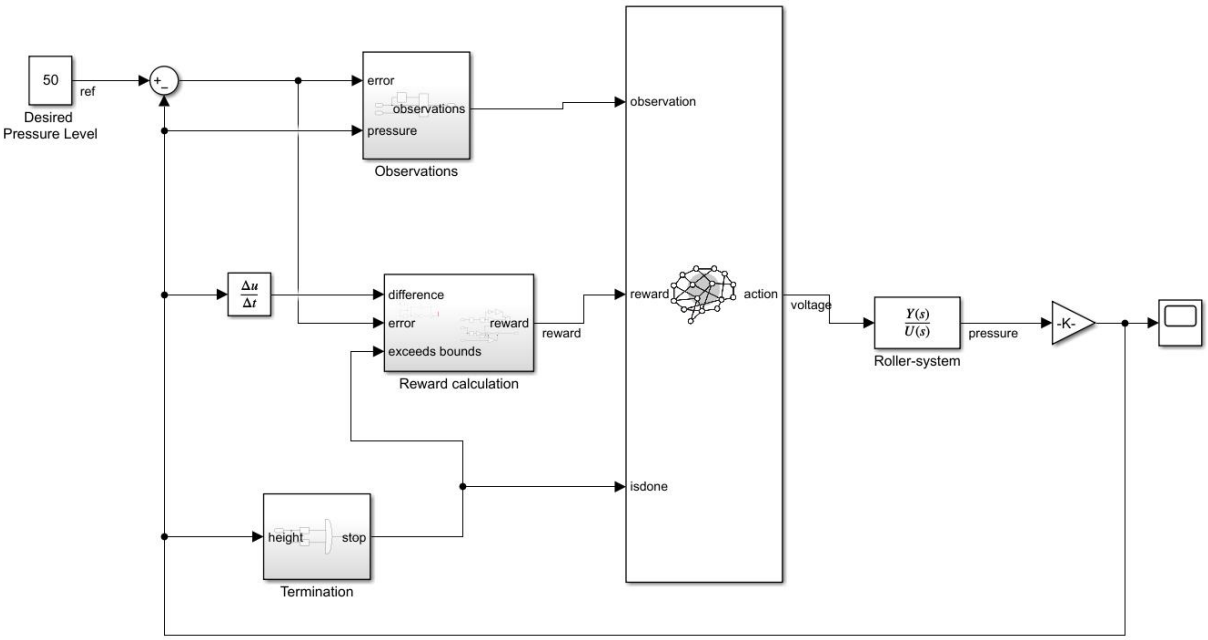


Figure 0.11: RL control scheme for the CEFAS/hot rolling system.

The actor network is a deep neural network with one input, one output and three fully connected layers. The input is the observation, and the output is the action taken. On the other hand, the critic network has is a deep neural network which takes the action and the observation as inputs and one output that updates the actor network. The configuration of the critic network used in the DDPG agent is shown in Figure (4.12).

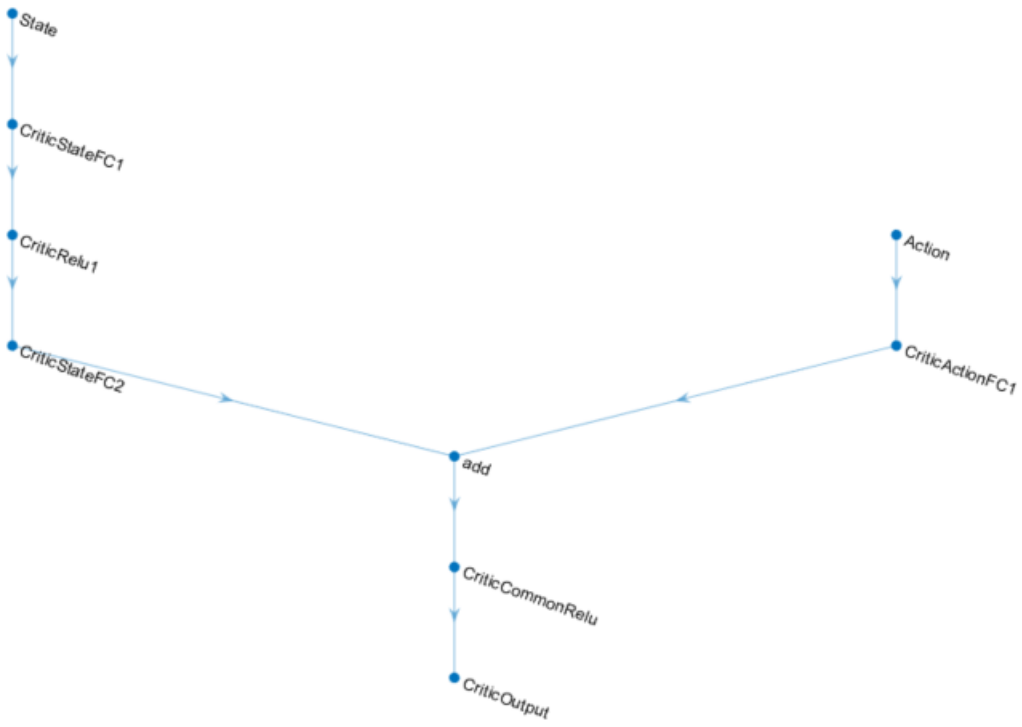


Figure 0.12: Configuration of the critic network.

A complete sequence of states, actions and rewards is known as one episode. An episode terminates either by the termination signal or by reaching the maximum steps. The simulation time is selected as 200 seconds with a sampling time of 0.1 second. Therefore, maximum step size for an episode is 2000. The training process also gets terminated once the average episode reward reaches a value of 15000. The performance and evolution of the RL controller is discussed in the next chapter.

Chapter 05: Discussion & Conclusion

5.1 System Modeling Analysis

The CEFAS/hot rolling system modeled using the first principle method is a stable system with seven poles and one zero. However, the transfer function of the system in Equation (5.1) shows some coefficients with enormous values. Investigating the model implies that the values of the damping coefficient, motor inertia and stiffness, screw damping and stiffness are the primary reason behind this large magnitude. Furthermore, it is possible to obtain a lower order system by factoring out the larger coefficients from the numerator and denominator.

$$G(s) = \frac{4.09 \times 10^{10} s + 6.13 \times 10^{12}}{s^7 + 1005 s^6 + 5.4 \times 10^6 s^5 + 2.1 \times 10^9 s^4 + 3.8 \times 10^{11} s^3 + 3 \times 10^{13} s^2 + 2 \times 10^{15} s + 67.4} \quad 5.1$$

To validate this hypothesis, an experiment using MATLAB/SystemID Toolbox is performed. This experiment involves generating pressure data using a PRBS input through the CEFAS/hot rolling model. For validation data, two different datasets are generated using the following inputs-

- 12 random sinusoids are added together to generate a periodic input.
- A random binary signal with the range of (-1,1) is used to generate a non-periodic input.

The result from this experiment indicates that a lower order system with two poles and one zero (tf21_sys) performs best while a model with seven poles and one zero (tf71_LA) fails to capture the system behavior. The transfer functions of these two models are shown in Table (5.1). The

fitting percentage indicates the similarity between the model response and the measured output. A 100% fitting percentage means a perfect fit, and zero percent indicates a poor fit. A performance comparison between tf21_sys and tf71_LA is shown in Figure (5.1).

Table 0.1: Transfer functions of the selected models.

Model name	Fitting % (Periodic data)	Fitting % (Non-periodic data)	Transfer function
tf21_sys	61.75	39.93	$\frac{7.5\times10^{6}s+4.6\times10^{6}}{s^{2}+0.75s+7.4\times10^{-8}}$
tf71_LA	11.51	19.76	$\frac{7821s-226.5}{s^{7}+0.6s^{6}+0.28s^{5}+0.08s^{4}+0.02s^{3}+0.002s^{2}+9\times10^{-6}s+1.5\times10^{-6}}$

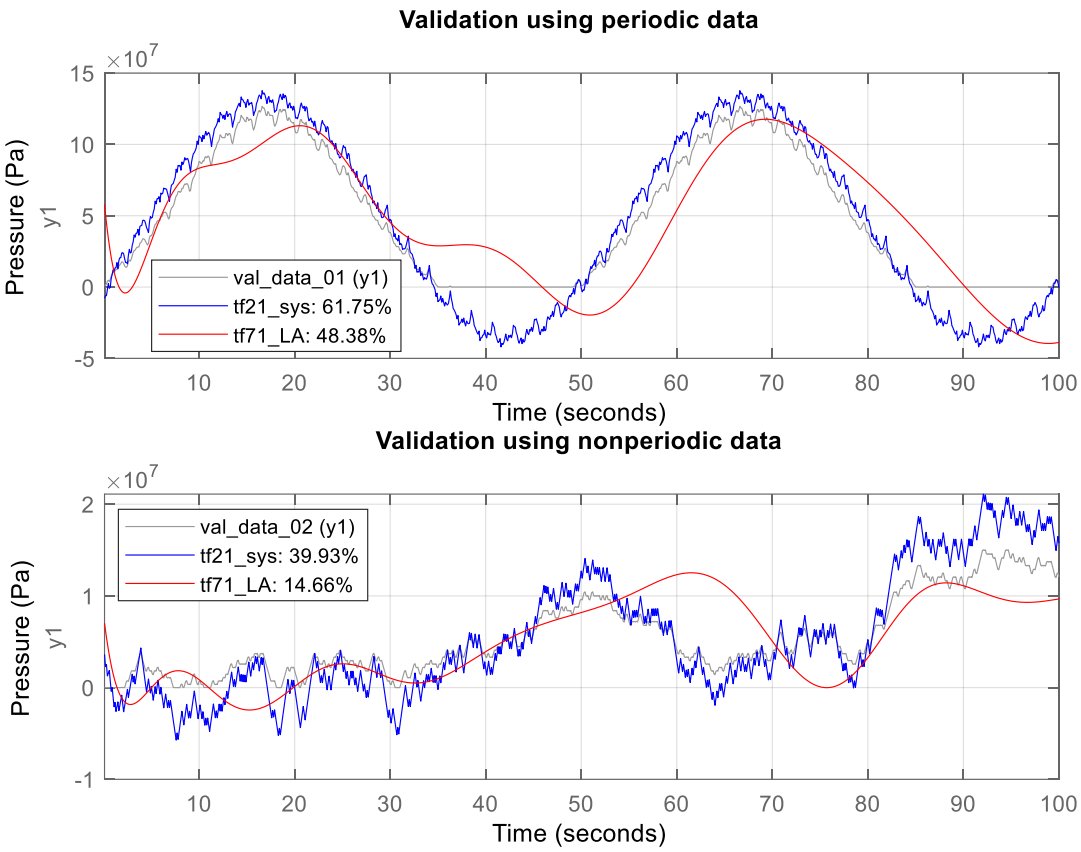


Figure 0.1: Performance analysis between two models.

5.2 Performance Analysis of Traditional Controllers

To investigate the close-loop performance of the considered models, PID controllers have been implemented. Table (5.2) represents the parameters and performance analysis of the PID controllers.

Table 0.2: PID parameters and performance analysis.

Model	K_p	K_i	K_d	N	Rise time(sec)	Settling time(sec)	Overshoot	Closed-loop stability
CEFAS/hot rolling	6.6e-6	3e-5	1e-8	2896	0.03	0.52	14.7%	Stable
tf71	-2.3e-7	-2.2e-9	-1.3e-6	1	153	566	7.06%	Stable
tf051	-8.8e-9	-1.9e-10	0	0	NaN	NaN	NaN	Unstable
tf21_sys	1.6e-7	3.2e-8	-6.1e-8	1	1.45	12.4	13.5%	Stable

Although the PID controller performed well to regulate the CEFAS/hot rolling model, the two identified models are not well suited to control with PID controllers. The PID controller designed for the tf71 model can reach the steady-state. However, the settling time is relatively large and there are oscillations present in the steady-state response. The impulse response of this system is plotted in Figure (5.2) and shows similar type of behavior. The oscillations are happening since five poles of this system are close to the imaginary axis. This can make a system marginally stable. For the tf051 model, although the open-loop system is stable, it doesn't have any closed-loop stability. The designed PI controller fails to control the plant.

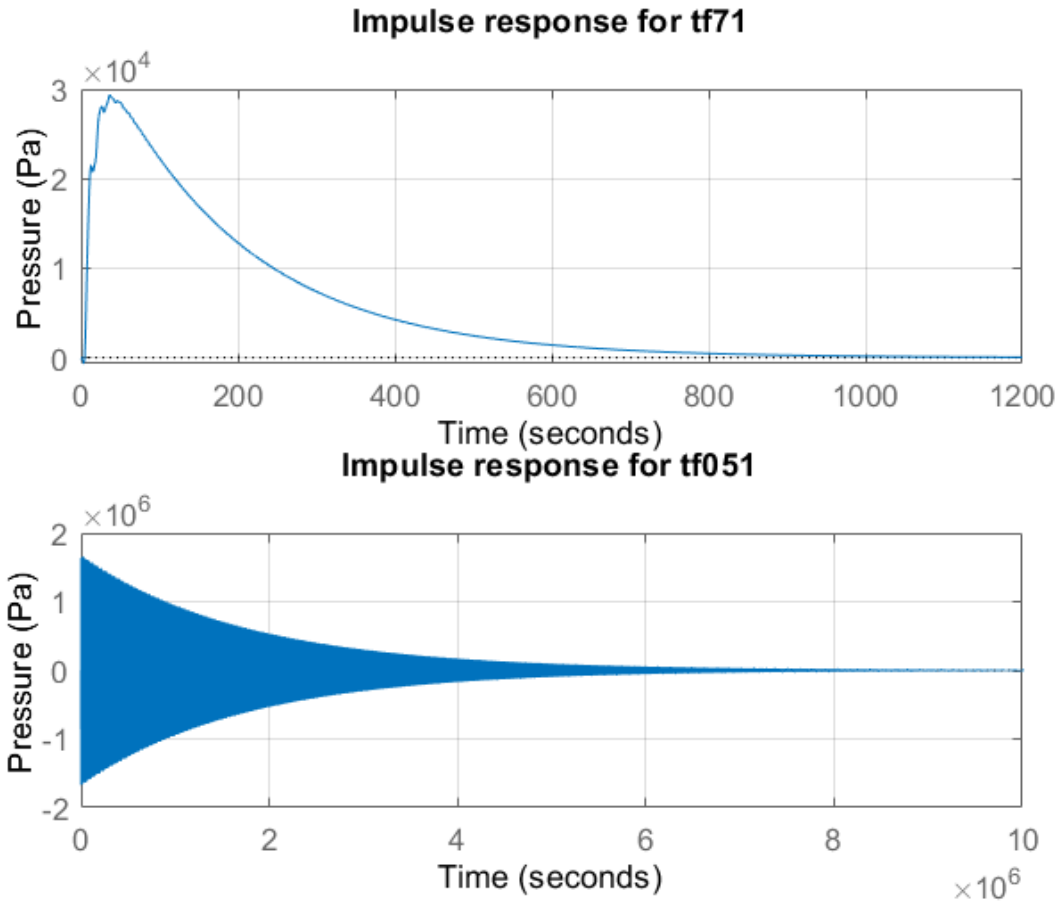


Figure 0.2: Impulse response for identified models.

The closed-loop stability of tf051 is further investigated using the root-locus method and the Nyquist plot. Figure (5.3) shows the root-locus and the Nyquist plot for the tf051 model. This model has two poles which are close to the imaginary axis. From the root locus plot it is noticeable that a small gain change can make the system unstable. Furthermore, this system is non-minimum phase as it has a pole on the right half s-plane. For a non-minimum phase system which is open-loop stable, a Bode plot can be misleading for closed-loop stability analysis. Instead of a Bode plot, a Nyquist plot is preferred (Fan & Miao, 2020) for a non-minimum phase system. A non-minimum phase and open-loop stable system is also closed-loop stable if the

Nyquist curve doesn't cross the critical point in the Nyquist plot. For tf051, the Nyquist plot shows that the Nyquist curve crosses the (0,0) critical point which makes it closed-loop unstable.

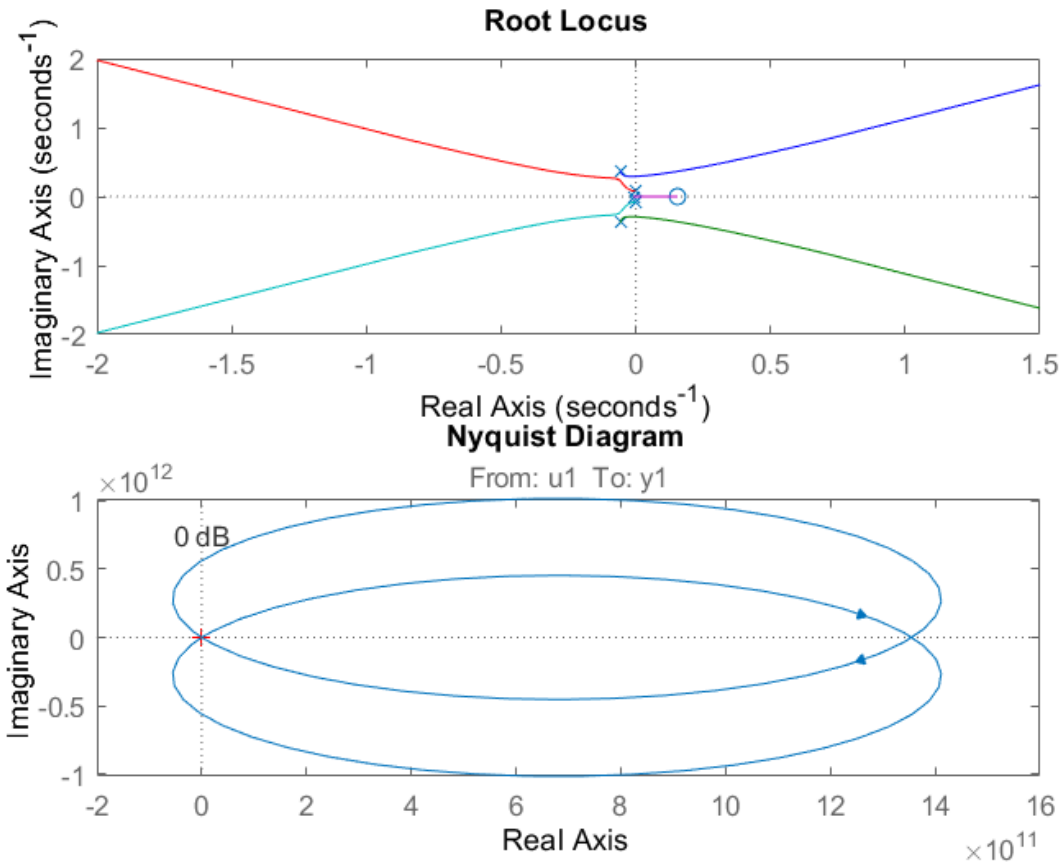


Figure 0.3: Stability investigation for tf051 model.

As the PID control scheme fails to properly regulate the identified models, other control schemes are explored. LQG is an optimal policy that optimizes the cost function of a full-state feedback controller. However, it requires full observability and controllability of the models. Among the considered models, the CEFAS/hot rolling model lacks observability. However, the identified models have full observability and controllability. Therefore, for the identified models, LQG controllers are studied. Figure (4.9) and (4.10) show the step responses for the designed

LQG regulated models. LQG is able to control both these models, however, for both cases there are some additional phase lags. The right half s-plane zero of the nonminimum phase system causes the step response to dip in the wrong direction first before recovering back to the steady state value. Nonminimum phase systems are susceptible to gain increment and experience lower phase margin due to additional phase lags. Therefore, nonminimum phase systems are harder to control. On the other hand, the steady-state error of the step response can be eliminated by tuning the Q and R matrices.

5.3 Evolution of RL controller

The most crucial part of designing an RL controller is to shape the reward function. The reward function should be optimized in such a way that will help the policy to get gradual feedback to get to the steady-state. For the CEFAS/hot rolling model, at first, a reward function that rewards when the error is within 30% is developed which punishes otherwise. Furthermore, a large reward penalty adds whenever an episode terminates due to out of bound output. The termination signal activates when output pressure exceeds the 0 to 100MPa limit. These conditions terminate every episode almost immediately. The cause of this is that the termination condition is not allowing the agent to explore the negative region of the action field. As from the previous analyses, it is evident that the identified models for CEFAS/hot rolling system are nonminimum phase systems which forces the step response to dip in the wrong direction first.

An RL controller trained with a negative bound may cause problems if implemented. However, the goal of this study is to show that an RL algorithm can capture the system response

behavior. Further exploration of RL controller with models that closely represent the real system will yield better results. To allow more exploration area to the agent, the termination limits are changed to -50 to 150MPa. In this case, for the first few episodes the response plummets. However, after some time the agent adopts to the nonminimum phase behavior of the response and tries to go upwards after a dip. Figure (5.4) shows how the RL agent adopts with the nonminimum phase behavior of the system. From the figure, it is apparent that the slope of the response is too steep which restricts the agent from exploring more action area and leads to termination.

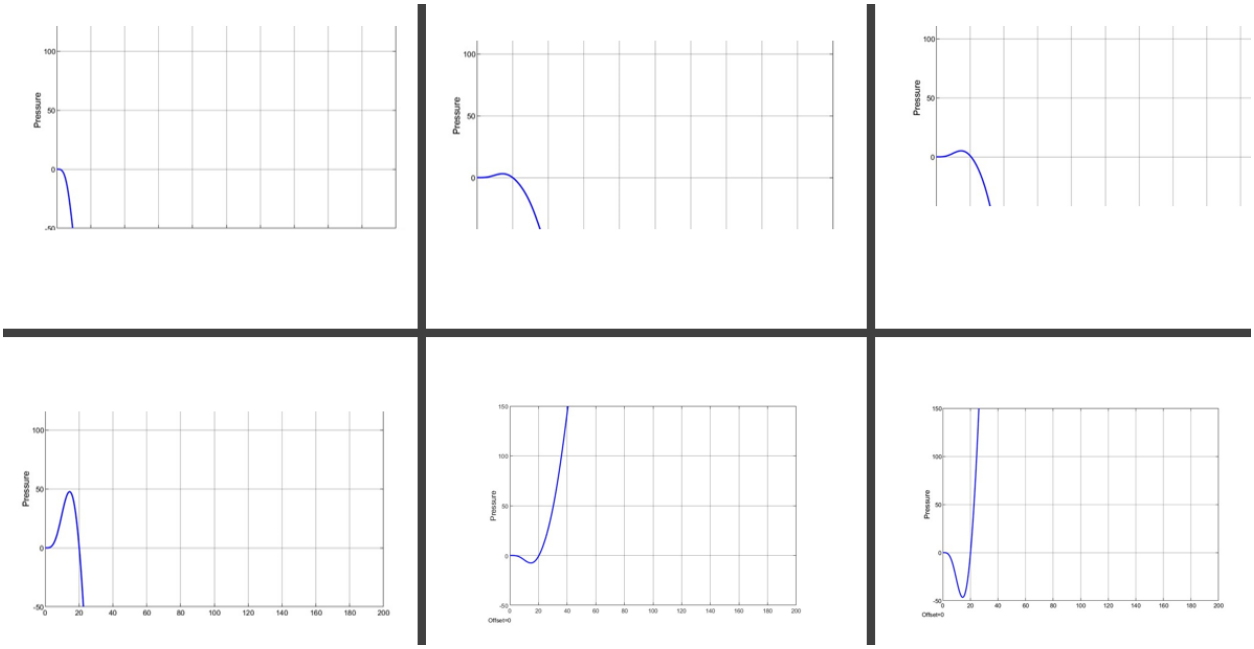


Figure 0.4: RL agent learns nonminimum phase behavior.

One way to solve the steep slope problem is to punish the agent whenever the slope of the output crosses a certain threshold. By tuning this threshold value *d*, the agent can be forced to learn to rise or fall slowly. Whenever the rate of change of pressure is below the threshold value a positive reward is generated. However, if this positive reward is large, then the agent tends to

go slower rather than exploring more action space. Therefore, the value of *e* is kept much lower than the value of *c*. Figure (5.5) shows how the DDPG agent learnt to avoid steep rate of change of pressure.

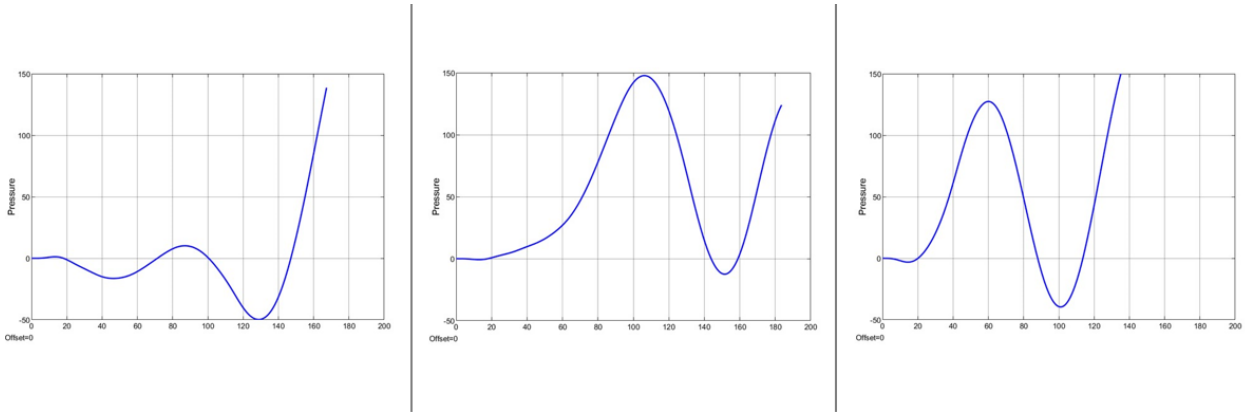


Figure 0.5: RL agent learns to avoid steep slope.

Although the response tends to behave slower, it fluctuates all over the action space. A term to punish the integral of error is added to reduce this fluctuation. The cumulative error will be multiplied by *h* and will generate a negative reward. However, this error term can get really large if an episode runs for several steps. To limit the negative reward, a threshold is added. Figure (5.6) shows the response after implementing this reward function.

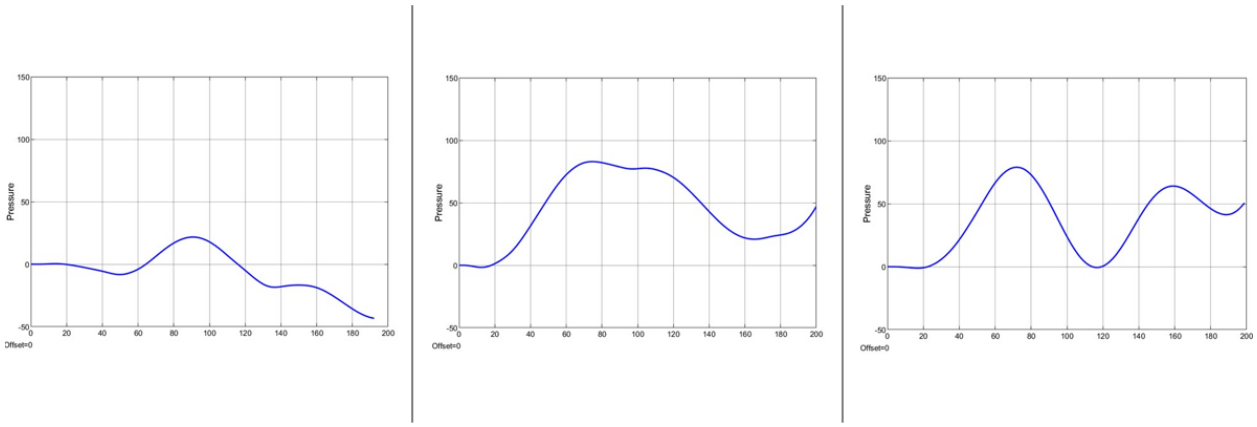


Figure 0.6: RL agent performance using the complete reward function.

5.4 Conclusion

This thesis presents the modeling and control scheme for the CEFAS/hot rolling system. Models are developed for this system using the first principle method and data driven techniques. Traditional control techniques such as PID and LQG are designed for the models while the possibility of introducing RL controllers are studied. Many questions are, however, still unanswered on several issues. For instance, the appearance of negative pressure on the sintered material during the absence of linear actuator movement. A hypothesis to explain this phenomenon is the expansion of the sintered material due to elasticity. Another interesting question is to ask the reason for such large fictional output pressure values. A possible answer could be the absence of friction and joule heating phenomenon in the COMSOL model.

The models identified here are the best linear representation of a highly nonlinear system. The primary reason to introduce RL in this work is to capture the non-linear behavior of the system and the initiation of an intelligent control scheme. Although traditional control schemes are effective and cost-efficient as primary controllers, RL is an online learning algorithm that can modify itself capturing any future disturbances. Benefits of training the RL agent in offline mode is that it doesn't have to start from the scratch when deployed to the real system. Therefore, the current reinforcement learning control scheme is a small step towards an intelligent control scheme.

5.5 Future Research Opportunities

CEFAS is a novel idea for manufacturing highly dense materials. Several opportunities exist for further explorations and improvements of the work presented here. The majority of the models used in this work are identified from the first batch of INL data generated using a COMSOL model. This approach only considers the variation of linear actuator movements and the resulting pressure. It will be interesting to see how this model alters considering rolling speed and joule heating as variable parameters. Besides, nonlinear system identification techniques can also be utilized. Another intriguing area for future research would be implementing a primary control scheme on the real system, allowing the opportunity to collect real time data that will eventually help to design a better controller. Models developed from later data can be used to train the RL controller.

Control schemes designed in this work do not address any robustness. LQG is an optimal controller explored for the nonminimum phase models does not guarantee robustness. A small perturbation can make it unstable. Therefore, robust PID and LQG controllers for primary implementation in the real system can be studied in future.

References

A novel PSO-PID controller application to bar rolling process | IEEE Conference Publication | IEEE Xplore. (n.d.). Retrieved July 2, 2022, from https://ieeexplore.ieee.org/abstract/document/6001520

Achenani, Y., Saâdaoui, M., Cheddadi, A., Bonnefont, G., & Fantozzi, G. (2017). Finite element modeling of spark plasma sintering: Application to the reduction of temperature inhomogeneities, case of alumina. *Materials & Design*, *116*, 504–514. https://doi.org/10.1016/J.MATDES.2016.12.054

Aghasafari, P., Abdi, H., & Salimi, M. (2014). Artificial neural network modeling of flow stress in hot rolling. *ISIJ International*, *54*(4), 872–879. https://doi.org/10.2355/ISIJINTERNATIONAL.54.872

Aghasafari, P., Salimi, M., & Daraei, A. (2014). Flow stress evaluation in hot rolling of steel. *Journal of Materials Engineering and Performance*, *23*(8), 2819–2828. https://doi.org/10.1007/S11665-014-1049-X/TABLES/8

Anselmi-Tamburini, U., Garay, J. E., & Munir, Z. A. (2005). Fundamental investigations on the spark plasma sintering/synthesis process: III. Current effect on reactivity. *Materials Science and Engineering: A*, *407*(1–2), 24–30. https://doi.org/10.1016/J.MSEA.2005.06.066

Antong, H., Dixon, R., & Ward, C. (2014). Modelling and building of experimental rig for high redundancy actuator. *2014 UKACC International Conference on Control, CONTROL 2014 - Proceedings*, 384–388. https://doi.org/10.1109/CONTROL.2014.6915171

Antong, H., Dixon, R., & Ward, C. P. (2016). High Redundancy Actuator with 12 Elements: Open- and Closed-loop Model Validation. *IFAC-PapersOnLine*, *49*(21), 254–259. https://doi.org/10.1016/J.IFACOL.2016.10.563

Astrom, K. (1995). PID Controllers, 2nd Edition. *Instrument Society of America*, 343. http://ci.nii.ac.jp/naid/10013391165/

Athans, M., Athans, M., Kappos, E., & Spang, H. A. (2012). Linear-quadratic Gaussian with loop-transfer recovery methodology for the F-100 engine. *Https://Doi.Org/10.2514/3.20065*, *9*(1), 45–52. https://doi.org/10.2514/3.20065

Bertsekas, D. P. (n.d.). *Reinforcement learning and optimal control*. 373.

Block Libraries - MATLAB & Simulink. (n.d.). Retrieved July 6, 2022, from https://www.mathworks.com/help/simulink/block-libraries.html

Brunton, S. L. (Steven L., & Kutz, J. N. (n.d.). *Data-driven science and engineering : machine learning, dynamical systems, and control*. 472.

Calvo-Rolle, J. L., Casteleiro-Roca, J. L., Quintián, H., & del Carmen Meizoso-Lopez, M. (2013). A hybrid intelligent system for PID controller using in a steel rolling process. *Expert Systems with Applications*, *40*(13), 5188–5196. https://doi.org/10.1016/J.ESWA.2013.03.013

Cincotti, A., Locci, A. M., Orrù, R., & Cao, G. (2007). Modeling of SPS apparatus: Temperature, current and strain distribution with no powders. *AIChE Journal*, *53*(3), 703–719. https://doi.org/10.1002/AIC.11102

CISP History. (n.d.). Retrieved July 1, 2022, from http://www.cisp.psu.edu/history_sintering.html

Dixit, U. S., & Narayanan, R. G. (n.d.). *Metal forming : technology and process modelling*. 568.

Doyle, J. C. (1978). Guaranteed Margins for LQG Regulators. *IEEE Transactions on Automatic Control*, *23*(4), 756–757. https://doi.org/10.1109/TAC.1978.1101812

Du, X., Dixon, R., Goodall, R. M., & Zolotas, A. C. (2010). Modelling and control of a high redundancy actuator. *Mechatronics*, *20*(1), 102–112. https://doi.org/10.1016/J.MECHATRONICS.2009.09.009

Dudina, D. v., & Mukherjee, A. K. (2013). Reactive spark plasma sintering: Successes and challenges of nanomaterial synthesis. *Journal of Nanomaterials*, *2013*. https://doi.org/10.1155/2013/625218

Fan, L., & Miao, Z. (2020). Admittance-Based Stability Analysis: Bode Plots, Nyquist Diagrams or Eigenvalue Analysis? *IEEE Transactions on Power Systems*, *35*(4), 3312–3315. https://doi.org/10.1109/TPWRS.2020.2996014

Grimble, M. J., & Hearns, G. (1998). LQG Controllers for State-Space Systems with Pure Transport Delays: Application to Hot Strip Mills. *Automatica*, *34*(10), 1169–1184. https://doi.org/10.1016/S0005-1098(98)00067-3

Grimble, M. J., & Hearns, G. (1999). Advanced Control for Hot Rolling Mills. *Advances in Control*, 135–169. https://doi.org/10.1007/978-1-4471-0853-5_5

Groza, J. R., & Zavaliangos, A. (2000). Sintering activation by external electrical field. *Materials Science and Engineering: A*, *287*(2), 171–177. https://doi.org/10.1016/S0921-5093(00)00771-1

Hirt, G., Bambach, M., Seuren, S., Henke, T., & Lohmar, J. (2013). Recent developments in modeling of hot rolling processes: Part I - Fundamentals. *AIP Conference Proceedings*, *1532*(1), 222. https://doi.org/10.1063/1.4806828

Holland, T. B., & Muhkerjee, A. (2010). Spark plasma sintering. *Combustion Synthesis: Novel Routes to Novel Materials*, 175–194. https://doi.org/10.2174/978160805155711001010175

Hulbert, D. M., Anders, A., Andersson, J., Lavernia, E. J., & Mukherjee, A. K. (2009). A discussion on the absence of plasma in spark plasma sintering. *Scripta Materialia*, *60*(10), 835–838. https://doi.org/10.1016/J.SCRIPTAMAT.2008.12.059

Hulbert, D. M., Jiang, D., Anselmi-Tamburini, U., Unuvar, C., & Mukherjee, A. K. (2008). Experiments and modeling of spark plasma sintered, functionally graded boron carbide–aluminum composites. *Materials Science and Engineering: A*, *488*(1–2), 333–338. https://doi.org/10.1016/J.MSEA.2007.11.054

Jens Graf. (2016). *PID control fundamentals*.

Jiang, R., Torresani, E., Cui, G., Olevsky, E. A., Bacciocchi, M., Tashiro, S., & Castelli, I. E. (2021). Proportional Integral Derivative Control in Spark Plasma Sintering Simulations. *Materials 2021, Vol. 14, Page 1779*, *14*(7), 1779. https://doi.org/10.3390/MA14071779

Lenard, J. G., & Pietrzyk, M. (1992). Rolling Process Modelling. *Numerical Modelling of Material Deformation Processes*, 274–302. https://doi.org/10.1007/978-1-4471-1745-2_11

Li, W., Olevsky, E. A., McKittrick, J., Maximenko, A. L., & German, R. M. (2012). Densification mechanisms of spark plasma sintering: Multi-step pressure dilatometry. *Journal of Materials Science*, *47*(20), 7036–7046. https://doi.org/10.1007/S10853-012-6515-Y/FIGURES/14

Locci, A. M., Licheri, R., Orrù, R., & Cao, G. (2009). Reactive Spark Plasma Sintering of rhenium diboride. *Ceramics International*, *35*(1), 397–400. https://doi.org/10.1016/J.CERAMINT.2007.11.012

Mamedov, V. (2013). Spark plasma sintering as advanced PM sintering method. *Http://Dx.Doi.Org/10.1179/003258902225007041*, *45*(4), 322–328. https://doi.org/10.1179/003258902225007041

Manière, C., Lee, G., & Olevsky, E. A. (2017). Proportional integral derivative, modeling and ways of stabilization for the spark plasma sintering process. *Results in Physics*, *7*, 1494–1497. https://doi.org/10.1016/J.RINP.2017.04.020

Method for creating functionally graded materials with spark plasma sintering and a continuous machine for future scalability. (n.d.). Retrieved July 2, 2022, from https://mountainscholar.org/handle/10217/185776

Molénat, G., Durand, L., Galy, J., & Couret, A. (2010). Temperature Control in Spark Plasma Sintering: An FEM Approach. *Journal of Metallurgy*, *2010*. https://doi.org/10.1155/2010/145431

Morin, C., le Gallet, S., Ariane, M., & Bernard, F. (2016). Spark Plasma Sintering tool design for preparing alumina-based Functionally Graded Materials. *Ceramics International*, *42*(2), 3056–3063. https://doi.org/10.1016/J.CERAMINT.2015.10.092

Munir, Z. A., Anselmi-Tamburini, U., & Ohyanagi, M. (2006). The effect of electric field and pressure on the synthesis and consolidation of materials: A review of the spark plasma sintering method. *Journal of Materials Science 2006 41:3*, *41*(3), 763–777. https://doi.org/10.1007/S10853-006-6555-2

Munir, Z. A., Quach, D. v., & Ohyanagi, M. (2011a). Electric Current Activation of Sintering: A Review of the Pulsed Electric Current Sintering Process. *Journal of the American Ceramic Society*, *94*(1), 1–19. https://doi.org/10.1111/J.1551-2916.2010.04210.X

Munir, Z. A., Quach, D. v., & Ohyanagi, M. (2011b). Electric Current Activation of Sintering: A Review of the Pulsed Electric Current Sintering Process. *Journal of the American Ceramic Society*, *94*(1), 1–19. https://doi.org/10.1111/J.1551-2916.2010.04210.X

Nellippallil, A. B., De, • P S, Gupta, • A, Goyal, • S, & Singh, • A K. (n.d.). *Hot Rolling of a Non-heat Treatable Aluminum Alloy: Thermo-Mechanical and Microstructure Evolution Model*. https://doi.org/10.1007/s12666-016-0935-3

Olevsky, E. A., Aleksandrova, E. v., Ilyina, A. M., Dudina, D. v., Novoselov, A. N., Pelve, K. Y., & Grigoryev, E. G. (2013). Outside mainstream electronic databases: Review of studies conducted in the USSR and post-soviet countries on electric current-assisted consolidation of powder materials. *Materials*, *6*(10), 4375. https://doi.org/10.3390/ma6104375

Olevsky, E. A., & Dudina, D. v. (2018). Field-assisted sintering: Science and applications. *Field-Assisted Sintering: Science and Applications*, 1–425. https://doi.org/10.1007/978-3-319-76032-2/COVER

Omori, M. (2006). Basic research and industrial production using the spark plasma system (SPS). *Advances in Microwave and Radio Frequency Processing - Report from the 8th*

International Conference on Microwave and High Frequency Heating, 745–754. https://doi.org/10.1007/978-3-540-32944-2_81/COVER/

Orrù, R., Licheri, R., Locci, A. M., Cincotti, A., & Cao, G. (2009). Consolidation/synthesis of materials by electric current activated/assisted sintering. *Materials Science and Engineering: R: Reports*, *63*(4–6), 127–287. https://doi.org/10.1016/J.MSER.2008.09.003

Pedersen, L. M., & Wittenmark, B. (1998). Multivariable controller design for a hot rolling mill. *IEEE Transactions on Control Systems Technology*, *6*(2), 304–312. https://doi.org/10.1109/87.664196

Pintelon, R., & Schoukens, J. (2001). *System Identification*. https://doi.org/10.1002/0471723134

Pomeroy, M. J. (2021). Encyclopedia of materials: Technical ceramics and glasses. *Encyclopedia of Materials: Technical Ceramics and Glasses*, *1–3*, 1–2674. https://doi.org/10.1016/c2017-1-04062-x

Sakkaki, M., Sadegh Moghanlou, F., Vajdi, M., Shahedi Asl, M., Mohammadi, M., & Shokouhimehr, M. (2020). Numerical simulation of heat transfer during spark plasma sintering of zirconium diboride. *Ceramics International*, *46*(4), 4998–5007. https://doi.org/10.1016/J.CERAMINT.2019.10.240

Schwertz, M., Katz, A., Sorrel, E., Lemonnier, S., Barraud, E., Carradò, A., d'Astorg, S., Leriche, A., Nardin, M., Vallat, M. F., & Kosior, F. (2016). Coupled Electro-Thermo-Mechanical Finite Element Modeling of the Spark Plasma Sintering Technique. *Metallurgical and Materials Transactions B: Process Metallurgy and Materials Processing Science*, *47*(2), 1263–1273. https://doi.org/10.1007/S11663-015-0514-8/FIGURES/11

St Clair, D. W. (1999). *Controller tuning and control loop performance : "PID without the math" : a primer*.

System Identification Overview - MATLAB & Simulink. (n.d.). Retrieved July 3, 2022, from https://www.mathworks.com/help/ident/gs/about-system-identification.html

Vamvoudakis, K. G., Wan, Y., Lewis, F. L., & Cansever, D. (n.d.). *Handbook of reinforcement learning and control*.

Wang, C., Cheng, L., & Zhao, Z. (2010). FEM analysis of the temperature and stress distribution in spark plasma sintering: Modelling and experimental validation. *Computational Materials Science*, *49*(2), 351–362. https://doi.org/10.1016/J.COMMATSCI.2010.05.021

Wei, K., & Nolas, G. S. (2018). Enhanced thermoelectric properties of polymer/inorganic bulk composites through EG treatment and spark plasma sintering processing. *Scripta Materialia*, *150*, 70–73. https://doi.org/10.1016/J.SCRIPTAMAT.2018.03.001

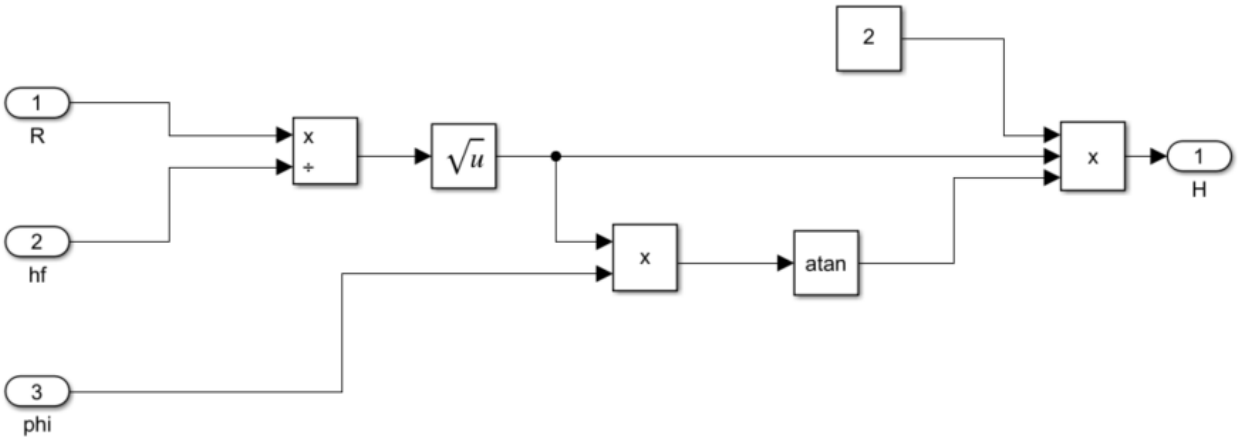
Weiss, L. (2010). Lectures on Controllability and Observability. *Controllability and Observability*, 201–289. https://doi.org/10.1007/978-3-642-11063-4_4

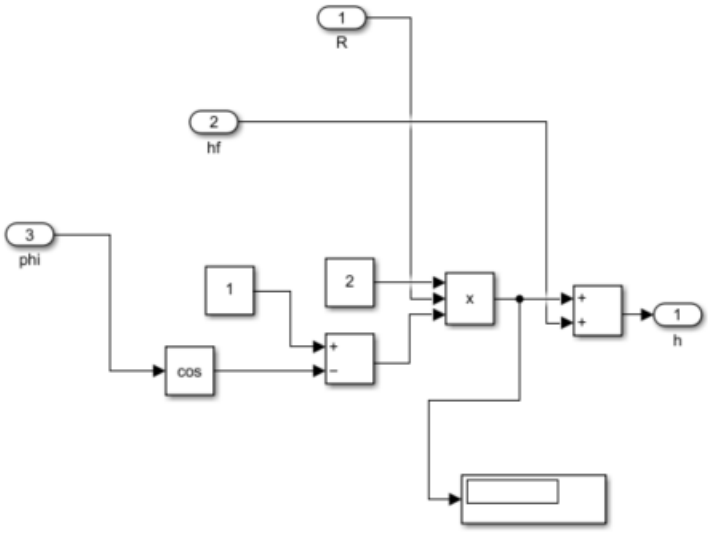
Yurlova, M. S., Demenyuk, V. D., Lebedeva, L. Y., Dudina, D. v., Grigoryev, E. G., & Olevsky, E. A. (2014). Electric pulse consolidation: An alternative to spark plasma sintering. *Journal of Materials Science*, *49*(3), 952–985. https://doi.org/10.1007/S10853-013-7805-8/FIGURES/37

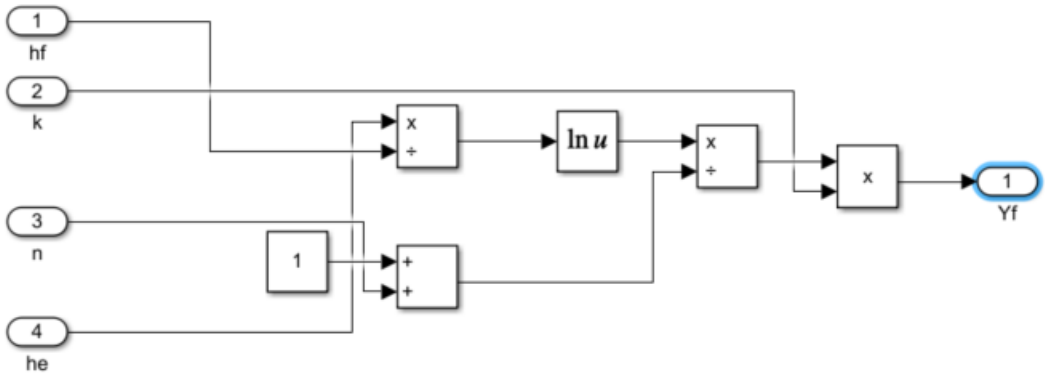
Appendix A: Simulink Design

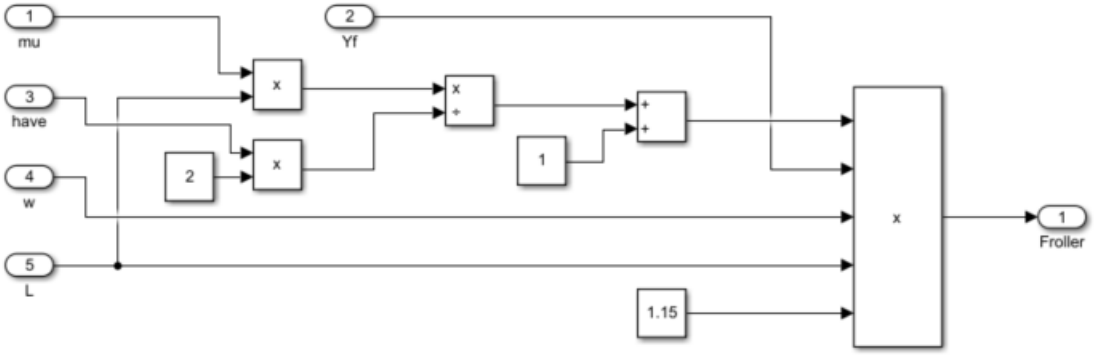
This appendix details the Simulink designs studied in this work.

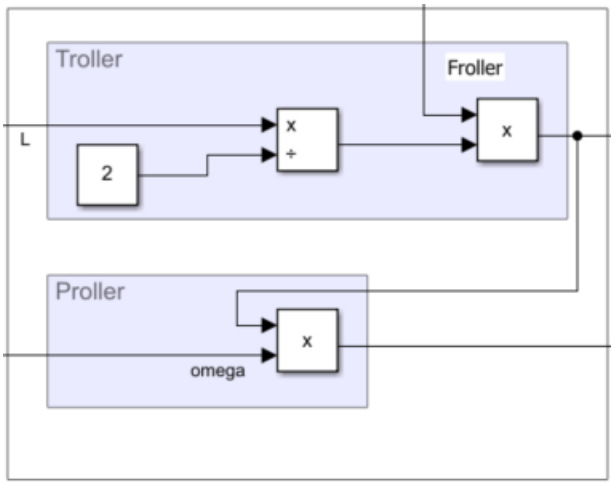
A.1 Hot Rolling System

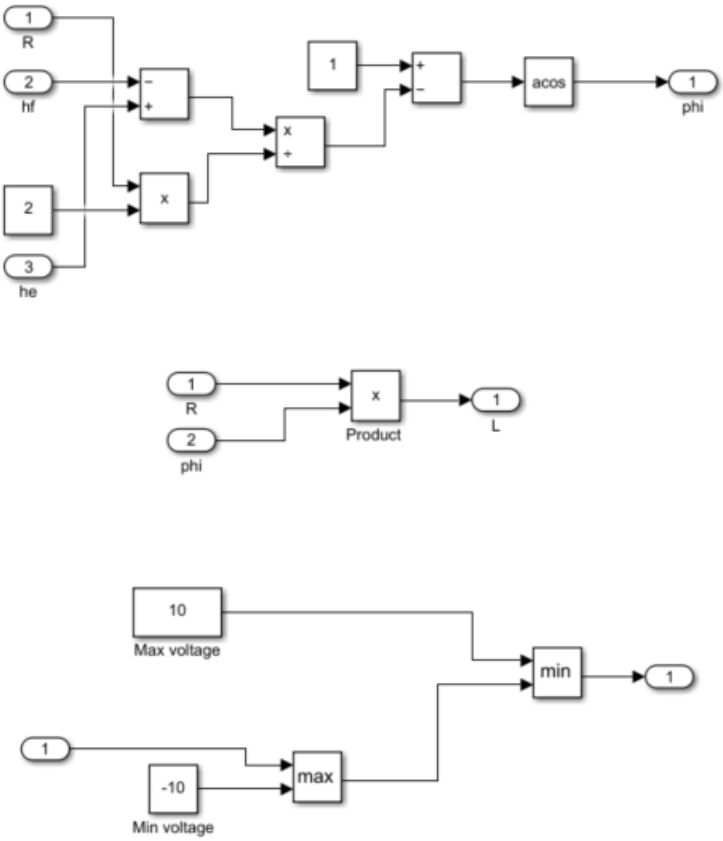




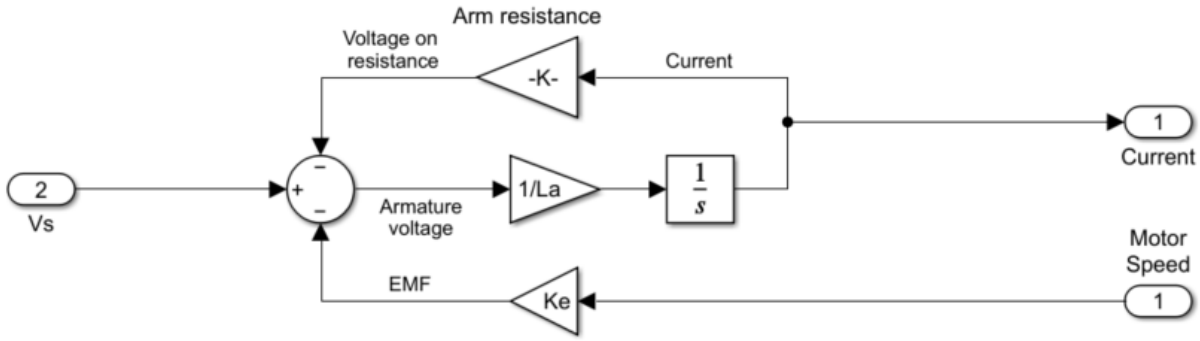

1
hf
2
k
3
n
4
he
x
÷
ln u
x
÷
1
+
+
x
1
Yf

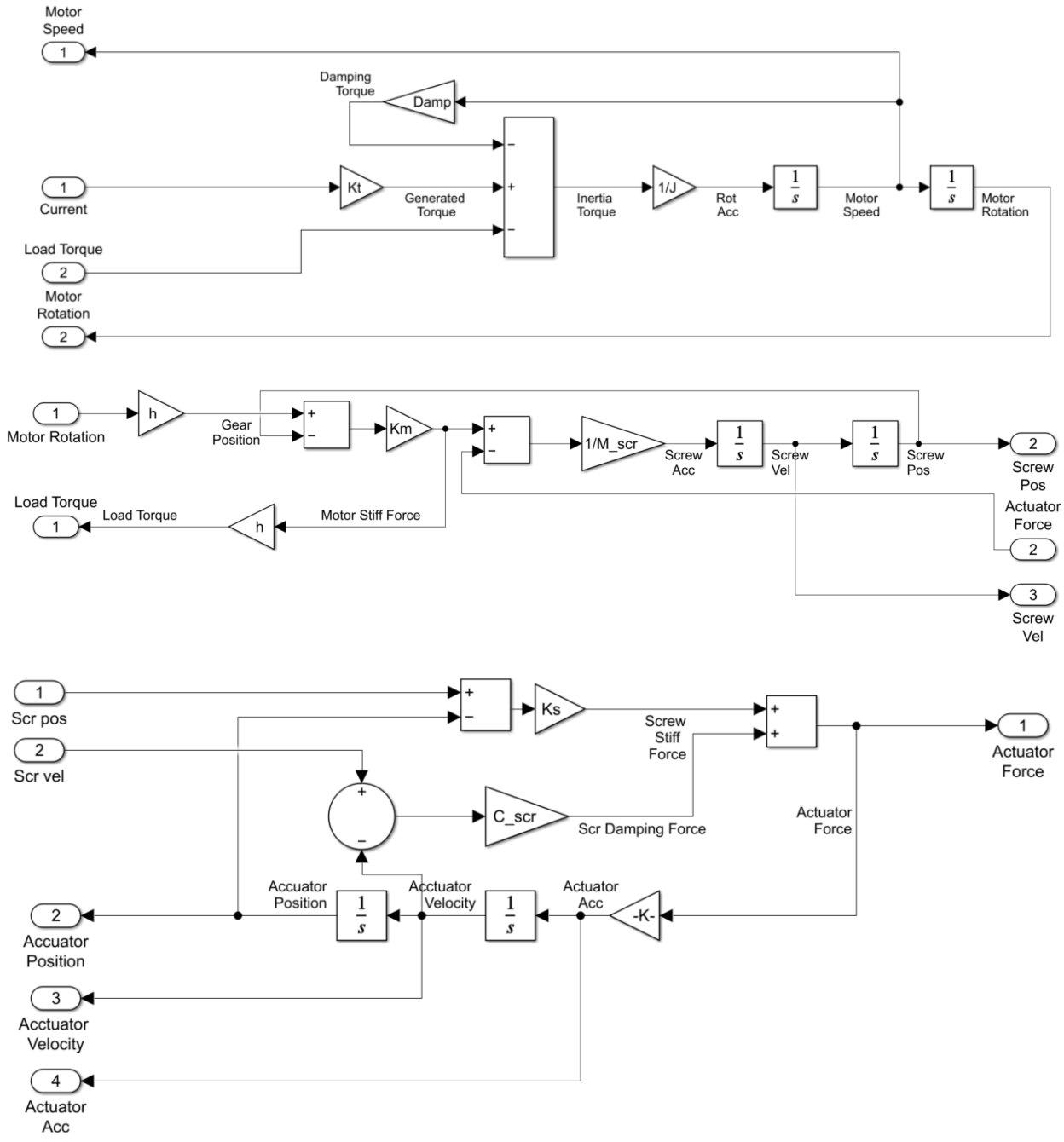

1
mu
2
Yf
3
have
4
w
5
L
x
x
÷
2
x
1
+
+
x
1
Froller
1.15


Troller
Froller
L
2
x
÷
x
Proller
x
omega

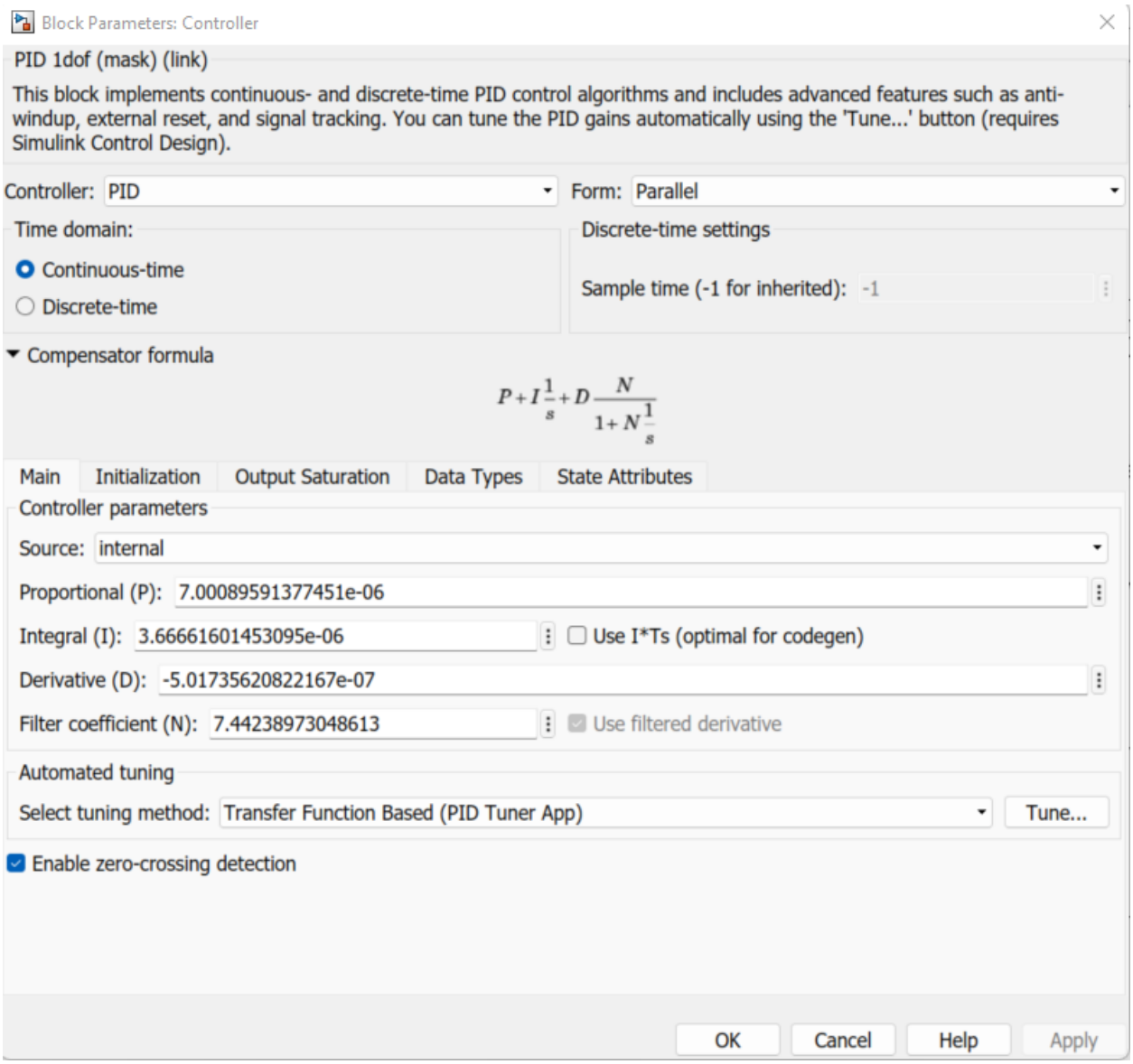


A.2 Linear Actuator Model

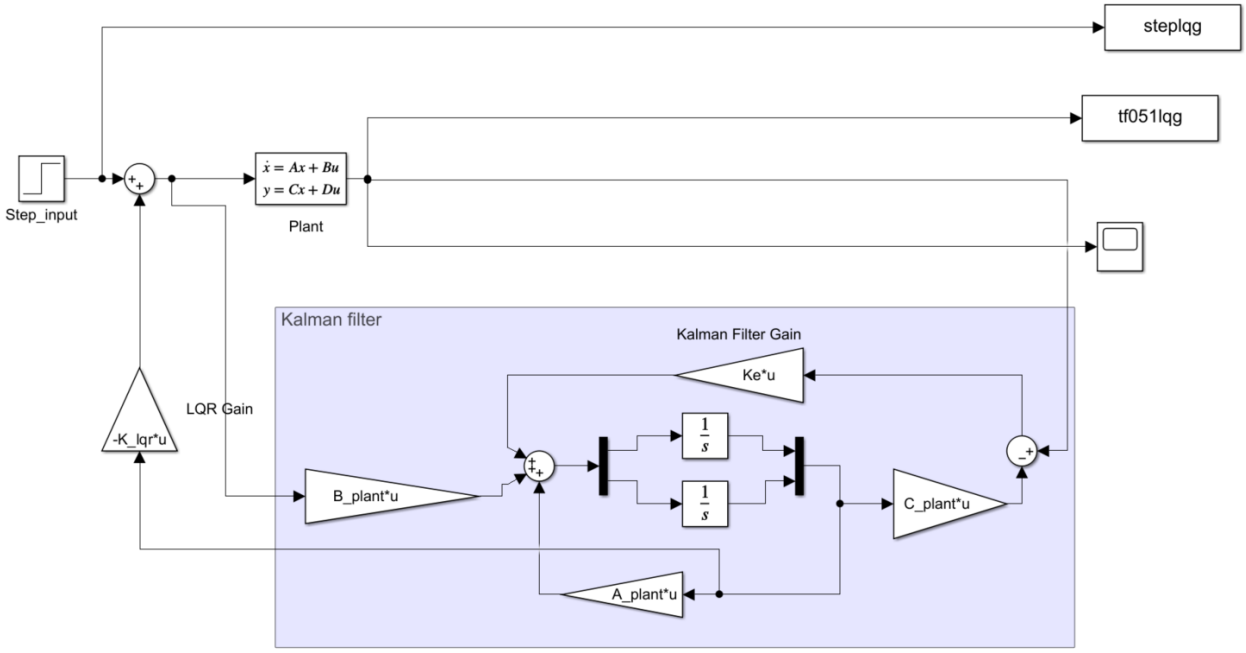




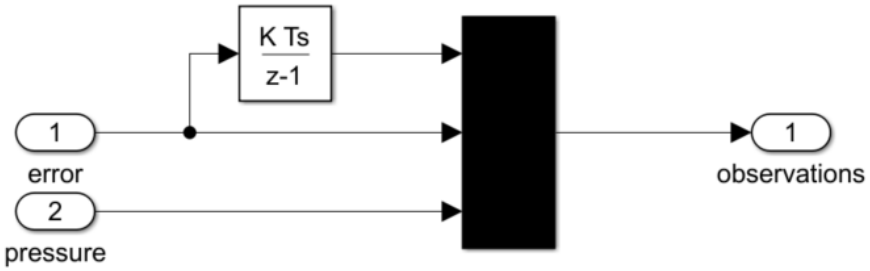
A.3 PID Parameters (General tuning window)

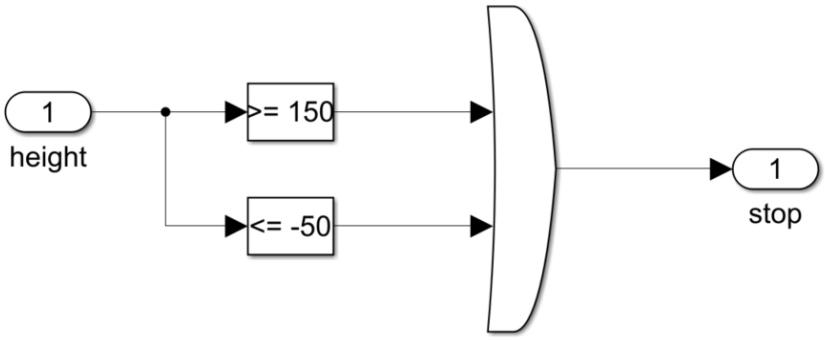


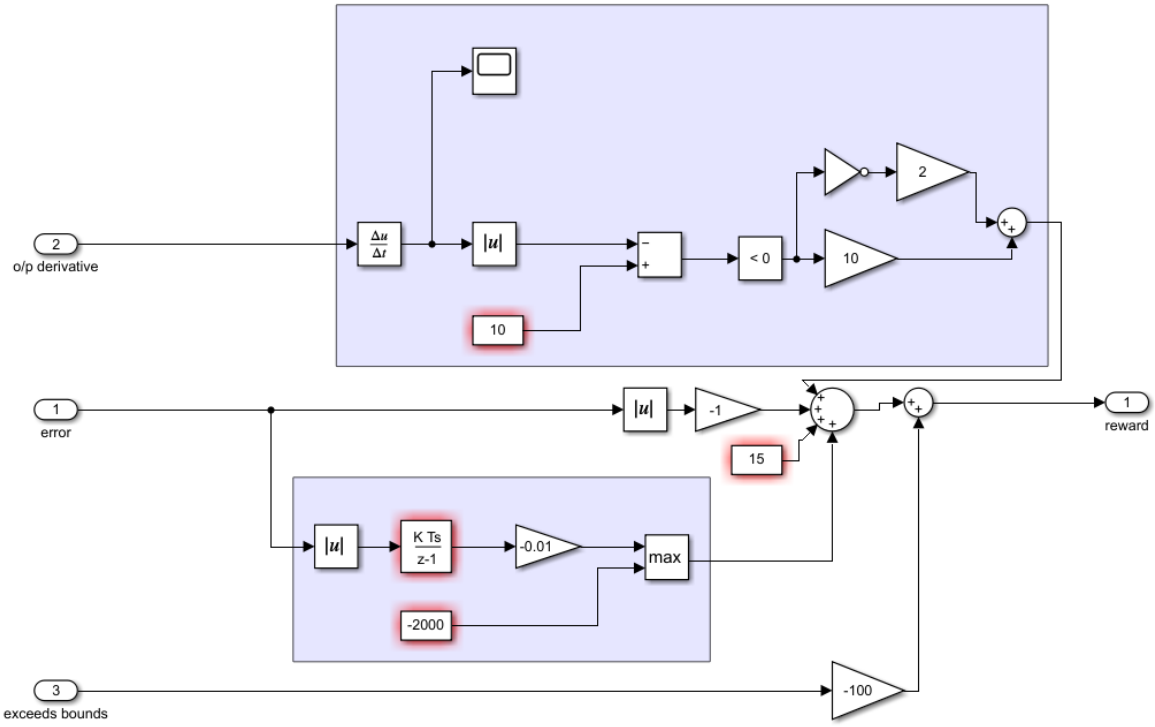
A.4 LQG Regulator Design



A.5 Reinforcement Learning Blocks






2
o/p derivative
Δu/Δt
|u|
10
< 0
2
10
1
error
|u|
-1
15
1
reward
|u|
K Ts
z-1
-0.01
max
-2000
3
exceeds bounds
-100

Appendix B: MATLAB Scripts

B.1 Scripts for Linear actuator & Hot Roll Press control

```
% Run this before running the Simulink model
%% Linear actuator model parameters
La = 6.4e-3; % Windings inductance in Henry
Arm_resistance = 2.2;  % Motor resistance in Ohm
Damp = 8e-5; % Damping coefficient in Nm/rads-1
J = 5.3e-5; % Motor inertia in kgm2
Ke = 0.121; % Voltage constant in V/rads-1
Kt = 0.121; % Torque constant in Nm/A
Km = 1e7; % Motor Stiffness in N/m
Ks = 1.8e5; % Screw Stiffness in N/m
lead_value = 2.4e-3; % in m/rev
M_load = 2000; % Load mass in Kg
M_scr = 2; % Screw mass in Kg
C_scr = 1.2e3; % Screw damping in N/ms-1
h = lead_value/(2*pi); % Calculate h

% State-space model for a 7*7 system
Ala = [(-Arm_resistance/La) (-Ke/La)   0              0          0              0          0;
    (Kt/J)              (-Damp/J)    (-(h^2*Km)/J)     0          ((h*Km)/J)        0          0;
    0                   1         0              0          0              0          0;
    0                   0         (h*Km)/M_scr     (-C_scr/M_scr) -(Ks+Km)/M_scr
C_scr/M_scr   Ks/M_scr;
    0                   0         0              1          0              0          0;
    0                   0         0              C_scr/M_load     Ks/M_load        -(C_scr/M_load)
-(Ks/M_load);
    0                   0         0              0          0              1          0]
Bla = [1/La; 0; 0; 0; 0; 0; 0];
Cla = [0 0 0 0 0 0 1];
Dla = zeros(size(Cla,1),size(Bla,2));

% State space, transfer function and pole zero plot
sys = ss(Ala,Bla,Cla,0);
G_tf = tf(sys)
num = cell2mat(G_tf.numerator);
den = cell2mat(G_tf.denominator);
[z,p,k] = tf2zp(num,den)
pzplot(G_tf,'g')
title('Pole Zero plot for Simulink/CEFAS model ')


% Linear Actuator state space model for 5*5 reduced system
Anew = [(-Arm_resistance/La) 0          (-Ke/La)              0        0;
```

```
        0             0            1                0        0;
       (Kt/J)          (-(h^2*Ks)/J) -(Damp+C_scr*h^2)/J  Ks*h/J     C_scr*h/J;
       0             0            0                0        1;
       0           Ks*h/M_load      C_scr*h/M_load       -Ks/M_load  -C_scr/M_load];
Bnew = [1/La; 0; 0; 0; 0];
Cnew = [0 0 0 1 0];

%% Controbality and Observability

[v_Ala, d_Ala] = eig(Ala);
Controlability_check = inv(v_Ala)*Bla
Observability_check =  Cla*v_Ala

%% Minimum Realization
sysr = minreal(sys);
sys_tf = minreal(G);

% Load lineareized plant from PID app
load('New_plant.mat')

%% Static model (hot rolling) parameters
he = 0.5; %thickness at entry in m
%hf = 0.5; %thickness at exit in mm
R = 0.1778; %Roller outer radius in m
% phi = 50*2*pi/360; % angle in radians
mu = 0.3; % constant
% L = 0.010; % workplace length in m
wo = 0.127;% initial width in m
wf = 0.180;% we need to compute this with the densinfication factor
% the initial height, and the initial width and final height3000; % final width in m
w = (wo + wf)/2; % average width
rpm = 1;
omega = 2*pi*rpm;
k = 500e6; % Strengthning coefficient
n = 0.25; % Work hardning exponent

%% Equations
% phi = (acosd(1-(he-hf)/(2*R))*2*pi)/360;
% L = R*phi
% eps = ln(he/hf);
% H = 2*sqrt(R/hf)* atand(sqrt(R/hf)*phi);
% have = hf + 2*R*(1-cos(phi));
% Yf = k * (eps/(n+1));
```

B.2 Script for LQG control

```
n_state = 5; % no. of states
n_input = 1; % no. of input
n_output = 1; % no. of output
% Define and create state-space system/transfer function
A_plant = A051;
B_plant = B051;
C_plant = C051;
D_plant = D051;
H_lqg = ss(A_plant,B_plant,C_plant,D_plant);

G_lqg = tf(H_lqg);
p = size(C_plant,1);
[n,m] = size(B_plant);

Q_lqr=eye(5);
Q_lqr(5,5) = 100;
Q_lqr(4,4) = 50; % Q penalizes state errors
R_lqr = 50; % R penalizes actuator effort
K_lqr = lqr(A_plant,B_plant,Q_lqr,R_lqr);   %LQR gain
Bnoise = eye(n);
Vd = eye(n); % Disturbance covariance
Vn = 0.01*eye(m); % Noise covariance
sysKF=ss(A_plant,[B_plant Bnoise],C_plant,[0 0 0 0 0 0]);
[Kess,Ke]=kalman(sysKF,Vd,Vn);   % Design kalman filter
```

B.3 Script to generate PRBS input & validation dataset

```
F = 1/0.1;  % Frequency of each element generated by PRBS method.
Timestep = 1/F;  % Interval in sec.
Duration = 100;  % Duration in sec.
Ts = 0.7;  % Settling time for linear actuator
N = ceil(Duration/Timestep)+1;  % Total number of samples.
B = Timestep/(Ts*0.2);  % 1/B is minimum number of samples assigned to per PRBS
element
% 20% of the settling time
dt = 1/F;  % Time stamp for each sample in sec.
Range = [-10 10];
S_N= 3*(Ts/Timestep);  % 3 times the settling sample
Band = [1/S_N  B];  % [0 B]; idinput takes floor(B)
channel=1;

% Generate input
```

```
u_input = ones(N,channel).*idinput(N, 'prbs', Band, Range); % Empty output/Binary
hence [0,1]
data00 = iddata([], u_input, dt);
u_in = data00.u(:,1);
iter = size(u_in);
u_in_re = zeros(iter(1),1);
for n = 1:iter(1) % flipping the original sequence to get valid output from Linear actuator
   if u_in(n,1) == -10
      u_in_re(n,1) = +10;
   else
      u_in_re(n,1) = -10;
   end
end

time = Timestep*(0:Duration/Timestep)'; % timestep data to import in simulink
tt = timetable(seconds(time), u_in_re); % timetable data to import in simulink inport

figure()
subplot(2,1,1)
plot(u_input,'b');
title('PRBS data');
ylabel('Amplitude (V)')
subplot(2,1,2)
plot(time,out.simout)
title('Output from LA model')
axis([0 100 -.02 0.15])
xlabel('Time (sec)')
ylabel('Movement (meter)')
grid on;

figure()
plot(time,out.simout)
title('Output from LA model')
axis([0 100 -.02 0.15])
xlabel('Time (sec)')
ylabel('Movement (meter)')

data_fullSimmodel = iddata(out.simout,u_input, dt);

data_mat = [time,out.simout]

%% Validation Data 01
Period_01 = 500;
NumPeriod_01 = 2;
Range_val = [-10 10];
Band_val = [0 1];
```

```
NumSinusoids = 12;
NumTrials = 15;
GridSkip = 2;
SineData = [NumSinusoids,NumTrials,GridSkip];
u_val_01 = idinput([Period_01 1 NumPeriod_01],'sine',Band_val,Range_val,SineData)
plot(u_val_01)
time_val_01 = Timestep*(1:Duration/Timestep)'; % timestep data to import in simulink
tt_val_01 = timetable(seconds(time_val_01), u_val_01);
out_val_01 = out.simout_val(2:1001);
val_data_01 = iddata(out_val_01,u_val_01, dt)

%% Validation Data 02

NumChannel = 1;
Period_02 = 50;
NumPeriod_02 = 20;
u_val_02 = idinput([Period_02,channel,NumPeriod_02],'rgs');
tt_val_02 = timetable(seconds(time_val_01), u_val_02);
out_val_02 = out.simout_val_02(2:1001);
val_data_02 = iddata(out_val_02,u_val_02, dt)
Num = 1000;
u_val_02 = idinput(Num);
val_data_02 = iddata(out.simout_val_02(2:1001),u_val_02,0.1);
tt_val_02 = timetable(seconds(time_val_01), u_val_02);
plot(u_val_02)
```

B.4 Data preprocessing

```
data_INL = readtable("act_data01.xlsx");
load('u_input.mat');
data_INL01 = table2array(data_INL);
column02 = data_INL01(:,2);
column03_00 = data_INL01(:,3);
press01 = [];
press02 = [];

%% Converting negative pressures to zero pressure
% @(0,0) point
for n = 1:1001
   if column03_00(n,1) >= 0
      press02(n,1) = column03_00(n,1);
   else
      press02(n,1) = 0;
```

```
    end
end

% @(0,-1.25) point
 for n = 1:1001
    if column02(n,1) >= 0
        press01(n,1) = column02(n,1);
    else
        press01(n,1) = 0;
    end
end

%% Plotting actual pressures at experiment points
figure()
subplot(311)
plot((data_INL01(:,4)))
xlabel('Sample')
grid on
ylabel('LA movement(m)')
subplot(312)
plot((data_INL01(:,2)))
xlabel('Sample')
ylabel('Pressure @(-1.25,0) (Pa)')
grid on
subplot(313)
plot((data_INL01(:,3)))
xlabel('Sample')
ylabel('Pressure @(0,0) (Pa)')
grid on

%% Generating testing and training data
p125_test_data = (press01(1:801,1));
p125_val_data = (press01(802:1001,1));
p125_val_data_trim = (press01(802:961,1));

p00_test_data = press02(1:801,1);
p00_val_data = press02(802:1001,1);

time_test = data_INL01(1:801,1);
time_val = data_INL01(802:1001,1);

input_test = u_input(1:801);
input_val = u_input(802:1001);
input_val_trim = u_input(802:961);

data_test = iddata(p125_test_data,input_test,0.1);
```

```
data_val = iddata(p125_val_data,input_val,0.1);
data_val_trim = iddata(p125_val_data_trim,input_val_trim,0.1);

data00_test = iddata(p00_test_data,input_test,0.1);
data00_val = iddata(p00_val_data,input_val,0.1);
```

B.5 Script for SystemID experiments and plottings

```
%% Plots for sysID on INL data
Options = tfestOptions;
Options.EnforceStability = true;

tf71 = tfest(data_test, 7, 1, Options);
tf11 = tfest(data_test, 1, 1, Options);
tf43 = tfest(data_test, 4, 3, Options);
tf73 = tfest(data_test, 7, 3, Options);
tf41 = tfest(data_test, 4, 1, Options);
tf51 = tfest(data_test, 5, 1, Options);

figure()
compare(data_val,tf71,'r',tf43,'b',tf73,'g',tf11,'y',tf51,'m',tf41,'c')
ylabel('Pressure (Pa)')
xlabel('Time')
grid on

tf071 = tfest(data00_test, 7, 1, Options);
tf043 = tfest(data00_test, 4, 3, Options);
tf011 = tfest(data00_test, 1, 1, Options);
tf051 = tfest(data00_test, 5, 1, Options);

figure()
compare(data00_val, tf043,'c',tf051,'g',tf011,'r',tf071, 'b')
ylabel('Pressure (Pa)')
xlabel('Time')
grid on

figure()
subplot(2,1,1)
compare(data_val,tf71,'r',tf051,'b')
ylabel('Pressure (Pa)')
xlabel('Time')
title('Validation using data__val')
grid on
```

```
subplot(2,1,2)
compare(data00_val, tf71,'r',tf051,'b')
ylabel('Pressure (Pa)')
xlabel('Time')
title('Validation using data00__val')
grid on

%% Step response plotting
plot(out.step, 'linewidth',1)
hold on
plot(out.pidla)
xlabel('Time (sec)')
ylabel('Pressure (Pa)')
legend('Step value','Closed-loop response')
title('CEFAS/hot rolling model')
grid on

plot(steplqg,'linewidth',1)
hold on
plot(tf051lqg)
xlabel('Time (sec)')
ylabel('Pressure (Pa)')
legend('Step value','Closed-loop response')
title('tf051')
grid on


%% SystemID TF with pressure modification
load TF_mod_cv.mat
num_71_mod = -tf71.Numerator
den_71_mod = tf71.Denominator

num_51_mod = -tf051.Numerator
den_51_mod = tf051.Denominator

%% Check poles and zeros for tf051 and tf71
[z71,p71,k71] = tf2zp(num_71_mod,den_71_mod)
[z051,p051,k051] = tf2zp(num_51_mod,den_51_mod)

% Convert to state-space
[A71,B71,C71,D71] = tf2ss(num_71_mod,den_71_mod)
sys71 = ss(A71,B71,C71,D71);

o71 = rank(obsv(A71,C71))
r71 = rank(ctrb(A71,B71))
```

```
% % Invariant subspace for ctrb and obsv check
% [v71,d71] = eig(A71);
% inv(v71)*B71
% C71*v71

[A051,B051,C051,D051] = tf2ss(num_51_mod,den_51_mod)
sys051 = ss(A051,B051,C051,D051);
o51 = rank(obsv(A051,C051))
r51 = rank(ctrb(A051,B051))

figure()
subplot(2,1,1)
pzplot(tf71,'r')
title('Pole Zero plot for tf71')
subplot(2,1,2)
pzplot(tf051,'b')
title('Pole Zero plot for tf051')

%% Impulse response for tf051 and tf71
figure()
subplot(211)
impulse(sys71)
ylabel('Pressure (Pa)')
title('Impulse response for tf71')
grid on
subplot(212)
impulse(sys051)
ylabel('Pressure (Pa)')
title('Impulse response for tf051')
grid on

%% CEFAS/hot rolling model reduction
figure()
subplot(2,1,1)
compare(val_data_01,tf21_sys,'b',tf71_LA,'r')
ylabel('Pressure (Pa)')
xlabel('Time')
title('Validation using periodic data')
grid on

subplot(2,1,2)
compare(val_data_02,tf21_sys,'b',tf71_LA,'r')
ylabel('Pressure (Pa)')
xlabel('Time')
title('Validation using nonperiodic data')
```

```
grid on

%% Nyquist and root-locus
figure()
subplot(211)
rlocus(tf051)
subplot(212)
nyquist(tf051)
grid on
```

B.6 Script for Reinforcement Learning

```
% load('TF_mod_cv.mat')
num_71 = tf71.Numerator;
den_71 = tf71.Denominator;
num_051 = tf051.Numerator;
den_051 = tf051.Denominator;

%% Environment interface for CEFAS Roller
% Define the observation specification obsInfo and action specification actInfo

obsInfo = rlNumericSpec([3 1],...
    'LowerLimit',[-inf -inf 0  ]',...
    'UpperLimit',[ inf  inf inf]');
obsInfo.Name = 'observations';
obsInfo.Description = 'integrated error, error, and measured voltage';
numObservations = obsInfo.Dimension(1);
actInfo = rlNumericSpec([1 1]);
actInfo.Name = 'voltage';
numActions = actInfo.Dimension(1);

%% Build the environment interface object
env = rlSimulinkEnv('RL_roller_2020','RL_roller_2020/RL Agent',...
    obsInfo,actInfo);

%% Set a custom reset function that randomizes the reference values for the model
env.ResetFcn = @(in)localResetFcn(in);

%% Specify the simulation time Tf and the agent sample time Ts in seconds.
Ts = 0.1;
Tf = 200;

%% Fix the random generator seed for reproducibility.
rng(0)
```

```
        %% create the critic
        % create a deep neural network with two inputs, the observation and action, and one
output
        statePath = [
           featureInputLayer(numObservations,'Normalization','none','Name','State')
           fullyConnectedLayer(50,'Name','CriticStateFC1')
           reluLayer('Name','CriticRelu1')
           fullyConnectedLayer(25,'Name','CriticStateFC2')];
        actionPath = [
           featureInputLayer(numActions,'Normalization','none','Name','Action')
           fullyConnectedLayer(25,'Name','CriticActionFC1')];
        commonPath = [
           additionLayer(2,'Name','add')
           reluLayer('Name','CriticCommonRelu')
           fullyConnectedLayer(1,'Name','CriticOutput')];
        criticNetwork = layerGraph();
        criticNetwork = addLayers(criticNetwork,statePath);
        criticNetwork = addLayers(criticNetwork,actionPath);
        criticNetwork = addLayers(criticNetwork,commonPath);
        criticNetwork = connectLayers(criticNetwork,'CriticStateFC2','add/in1');
        criticNetwork = connectLayers(criticNetwork,'CriticActionFC1','add/in2');

        %View the critic network configuration.
        % figure
        % plot(criticNetwork)

        %Specify options for the critic representation using rlRepresentationOptions
        criticOpts = rlRepresentationOptions('LearnRate',1e-1,'GradientThreshold',1);

        %Create the critic representation using the specified deep neural network and options.
        critic =
rlQValueRepresentation(criticNetwork,obsInfo,actInfo,'Observation',{'State'},'Action',{'Action'},
criticOpts);

        % Construct the actor in a similar manner to the critic.
        % To create the actor, first create a deep neural network with one input, the observation,
and one output, the action

        actorNetwork = [
           featureInputLayer(numObservations,'Normalization','none','Name','State')
           fullyConnectedLayer(3, 'Name','actorFC')
           tanhLayer('Name','actorTanh')
           fullyConnectedLayer(numActions,'Name','Action')
           ];
        actorOptions = rlRepresentationOptions('LearnRate',1e-1,'GradientThreshold',1);
```

```
actor =
rlDeterministicActorRepresentation(actorNetwork,obsInfo,actInfo,'Observation',{'State'},'Action'
,{'Action'},actorOptions);

%To create the DDPG agent, first specify the DDPG agent options using
rlDDPGAgentOptions.

agentOpts = rlDDPGAgentOptions(...
    'SampleTime',Ts,...
    'TargetSmoothFactor',1e-3,...
    'DiscountFactor',1.0, ...
    'MiniBatchSize',64, ...
    'ExperienceBufferLength',1e6);

%Then, create the DDPG agent using the specified actor representation, critic
representation, and agent options
agent = rlDDPGAgent(actor,critic,agentOpts);

%% Train Agent
maxepisodes = 5000;
maxsteps = ceil(Tf/Ts);
trainOpts = rlTrainingOptions(...
    'MaxEpisodes',maxepisodes, ...
    'MaxStepsPerEpisode',maxsteps, ...
    'ScoreAveragingWindowLength',20, ...
    'Verbose',false, ...
    'Plots','training-progress',...
    'StopTrainingCriteria','AverageReward',...
    'StopTrainingValue',8000);
load('agent.mat','agent')
doTraining = true;

if doTraining
    % Train the agent.
    trainingStats = train(agent,env,trainOpts);
else
    % Load the pretrained agent for the example.
    load('LAagent.mat','agent')
end

simOpts = rlSimulationOptions('MaxSteps',maxsteps,'StopOnError','on');
experiences = sim(env,agent,simOpts);


%% Build Custom reset function

function in = localResetFcn(in)
```

```
% randomize reference pressure
blk = sprintf('RL_roller_2020/Desired \nPressure Level');
h = 20 + 60*rand(1);
while h <= 20 || h >= 80
    h = 20 + 60*rand(1);
end
in = setBlockParameter(in,blk,'Value',num2str(h));
```