HYDRAULIC SYSTEMS DEGRADATION DETECTION USING SPARSE SENSORS

By

JOSEPH KELLEY

Bachelor of Science in Mechanical Engineering
University of Missouri Kansas City
Kansas City, Missouri, United States
2012

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
Masters of Science
May, 2018

HYDRAULIC SYSTEMS DEGRADATION DETECTION USING SPARSE SENSORS

Thesis Approved:

Martin Hagan

Thesis Advisor

Keith A. Teague

Carl D. Latino

Name: Joseph Kelley

Date of Degree: May, 2018

Title of Study: HYDRAULIC SYSTEMS DEGRADATION DETECTION USING SPARSE SENSORS

Major Field: Electrical Engineering

Abstract: A common issue in hydraulic systems is the degradation of sub-components. This is an important issue because hydraulics are the backbone of manufacturing, construction, and aerospace. When designing a hydraulic system, the sub-component's reliability is the primary requirement in component selection, because they determine the system's overall reliability. But, how the system actually fails is probabilistic, which varies based on operational conditions. To measure failure in a physical system during operation a model must be developed that measures the change in degradation from its initial healthy state. The accuracy of the prediction depends on the dynamics of the system and the system's operational input space. This thesis will present an approach for measuring degradation in a hydraulic system by using a dynamic model's prediction error to classify between sub-component fault states. The purpose of an inline fault detection system is to quickly recognize a failure and to provide information to the maintenance group on which sub-component has failed.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

Hydraulics are the backbone for the aerospace, construction, and manufacturing industry, but a common issue is degradation of sub-components in the system. For hydraulics it is critical to maximize the reliability, which requires a maintenance group with expert knowledge, endurance testing of sub-components, and historical data for failures. Even the most reliable systems will still have a chance of failure, with the types of failure changing based on operational and environmental conditions, so it is important to have an approach for continually checking the health state of system. In industry it is common to use off-line equipment to validate the health of sub-components in a hydraulic system, but this requires a deep understanding of the system and its failure modes. This process can be automated, which has became common practice in industry. This thesis will present an approach to do inline fault detection, based on using dynamic modeling, statistical testing, and classification. If a system can be inline tested for faults, the need to continually take the system off-line for health monitoring would be removed, which would reduce the cost and time for maintenance. This would also allowing for taking preemptive methods when faults are detected in real-time.

There have been a number of papers on fault detection in hydraulic systems using both linear and non-linear observer approaches. A common approach in fault detection is developing an observer from the differential equations for the system [1] [2], these differential models can be both linear and non-linear. The fault is detected based on the prediction error of the observer reaching a set threshold. This type of

fault detection is used for measuring the system's performance degradation inline, but does not provide the operator with an understand of individual sub-component health. Since the differential equations of a system can be difficult to derive, and because of uncertainty in parameter estimation, system identification approaches have been presented using neural networks [3] [4]. Similar to the differential equation approach, the neural network acts as an observer that uses the prediction error to detect a fault. These approaches only determine if a fault is present, but there have been methods that classify types of faults using neural networks and fuzz logic [5] [6]. For this thesis, these methods of system identification and fault classification will be combined for inline measurement of sub-component health states for a physical hydraulic system.

The general approach that will be presented for inline fault detection is dynamic modeling of the system to create a health reference model. When the system is operated in a healthy state, the dynamic model will be able to accurately predict the performance of the system using past inputs and outputs. As the system degrades the dynamics of the system will change, which will result in the healthy reference model being unable to accurately predict the performance of the system. The amount of error in the health reference model will depend on the type and severity of the fault. This means that the error in the health reference model will allow sub-component faults to be classified by the knowledge of what a fault looks like. This can be achieved by simulating faults in the system or from historical data. For this thesis the faults will be simulated in a physical hydraulic system.

The outline of this thesis follows. Chapter 2 will discuss the hydraulic system being used for fault detection, the sub-component differential equations, and the common types of faults in the system. After the hydraulic system is introduced, dynamic modeling approaches will be presented in Chapter 3 and 4, with Chapter 3 demonstrating linear dynamic modeling and Chapter 4 demonstrating non-linear

dynamic modeling. The dynamic models are trained on a healthy system, and then Chapter 5 and 6 will discuss how the errors in the health dynamic models can be used to classify faults. Chapter 5 will introduce the general equations for fault detection, and Chapter 6 will present the full implementation of the fault detection system.

CHAPTER 2

SYSTEM DESCRIPTION

2.1 Introduction

The purpose of hydraulic systems is to transfer mechanical generated power using a working fluid to perform mechanical work. By using a working fluid to transmit power, a hydraulic system is able to achieve high efficiency and handle high torque loads. With modern advancements in controls and sensors, hydraulic systems have become more stable and are the backbone in aerospace, heavy construction, and manufacturing systems. A common application of a hydraulic system is in velocity and torque control of motors coupled to a gearbox. In vehicles these systems are known as hydrostatic transmissions, which allow a vehicle to have a gearless transmission. The hydraulic motor can be flow controlled using two common approaches; pump and servo-valve control. A pump control is the most common in hydrostatic transmission systems due to the low response times required in heavy construction systems, but a servo-valve can be used in applications where a faster response is required than a variable controlled pump can achieve.

In this section a hydrostatic transmission and its individual sub-components will be discussed, to illustrate how a degradation in a sub-component can contribute to a fault. The hydrostatic transmission discussed in this thesis is a servo-valve controlled hydrostatic system that allows continuous velocity and torque control for low flow applications (less then 5 gal/min). The control system description of a hydrostatic transmission, which uses a voltage controlled servo-valve, is shown in Figure 2.1.

A hydrostatic system that is servo-valve controlled has six main components:

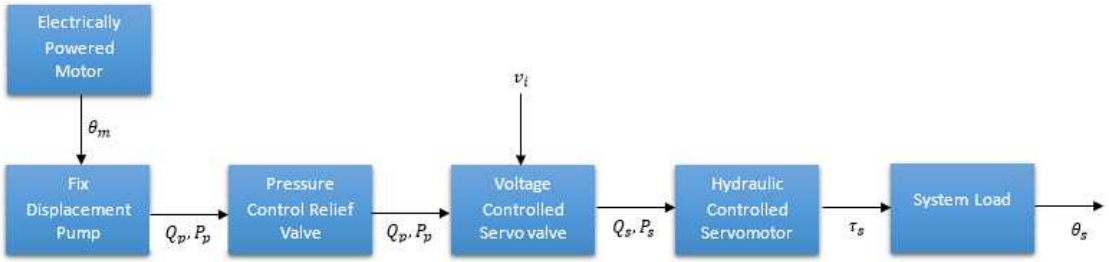

Figure 2.1: Control System Diagram of Hydrostatic Transmission

electrically powered motor, fixed displacement pump, pressure controlled relief valve, voltage controlled servo-valve, and hydraulic controlled motor. Each component will be discussed in-depth in the following sections, but we first provide a system overview.

The purpose of a hydrostatic system is to generate an angular velocity (θ_s) at the system load using a voltage controlled servo-valve and hydraulic motor. The servo-valve is a flow control device that is operated by a voltage difference (v_i) across a solenoid coil that causes a displacement in a spool. Depending on the position of the spool, the size of the opening for the flow channels into the hydraulic motor can be adjusted, which allows the flow rate (Q_s) into the hydraulic motor to be controlled. By controlling the flow into the hydraulic motor, the angular velocity (θ_s) of the motor can be varied.

To insure the system has a fast response, the pressure (P_p) and flow rate (Q_p) at the inlet of the servo-valve are held constant using a fixed displacement pump and pressure control relief valve. The fixed displacement pump in this application is a gear pump, which is coupled to an electrically powered motor that generates a constant flow rate at the inlet of the servo-valve. The pressure at the inlet of the servo-valve is held constant by a proportional controlled relief valve, which is designed to relieve excess pressure in the system. The proportional controlled relief valve can be set at a specified cracking pressure, which will insure that the pressure of the hydraulic system does not exceed the system maximum operational pressure. The fixed displacement

pump is selected based on the its ability supply a continuous pressure greater than the cracking pressure of the proportional controlled relief valve, to insure that there is no mechanical power loss in the system.

A physical model of the system is shown in Figure 2.2.

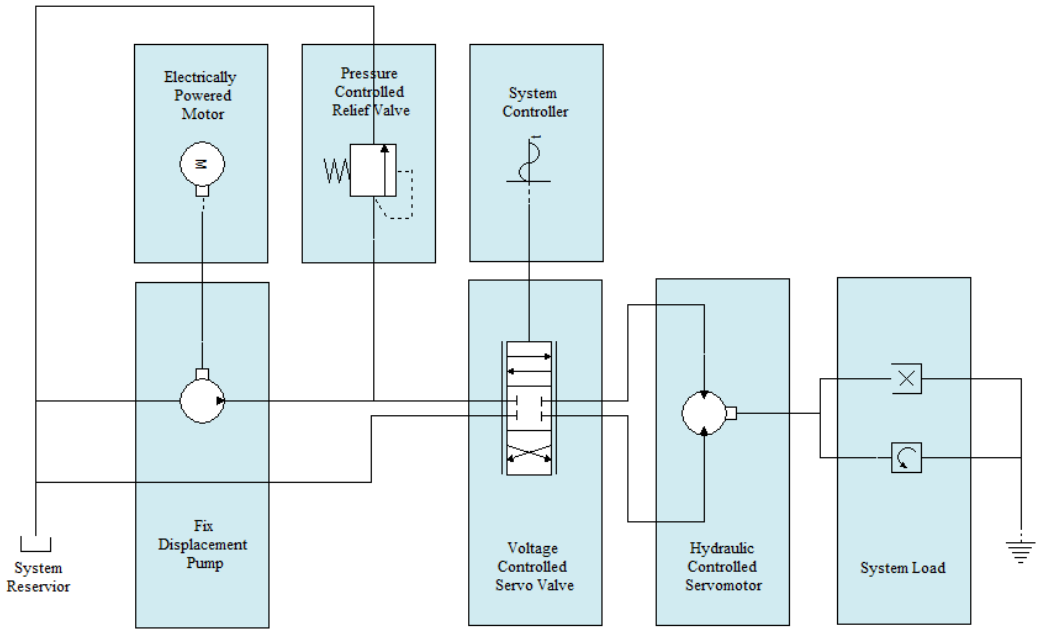

Figure 2.2: Physical Hydrostatic System

Further discussion of the sub-components in a servo-valve controller hydrostatic system continues in the following sections. Each section will describe the sub-component differential models and operation of the sub-component in both healthy and unhealthy states.

2.2 Electrically Powered Motor and Fixed Displacement Pump

2.2.1 Performance

The hydraulic system power is supplied by a gear pump that is driven by an electronic motor. A gear pump is a fixed displacement pump, which means that the flow from the outlet of the gear pump is equal to the rotational speed of the motor times the volumetric displacement of the pump, given that there is no efficiency loss due to internal or external leakage. If there is internal or external leakage, the inefficiency will be a product of the pressure difference across the pump. This is why it's common for hydraulic systems to use a booster pump to supply a fixed displacement pump, since it will insure there is a minimal pressure drop across the pump. The theoretical equation that governs a fixed displacement pump is shown in Equation 2.1 [7].

$$Q_p = D_m \cdot \dot{\theta}_m + \left(C_{im} + \frac{C_{em}}{2} \right) \cdot P_L \tag{2.1}$$

where:

Q_p = Flow Rate at Outlet of Pump (in^3/sec)

D_m = Volumetric Displacement of Motor (in^3/rad)

C_{im} = Internal Leakage Coefficient $(in^3/sec/psi)$

C_{em} = External Leakage Coefficient $(in^3/sec/psi)$

$P_L = P_{si} - P_{so}$ = Load Pressure (psi)

The design of a gear pump is shown in Figure 2.3. The gear pump was designed so that the meshing of the gears to the internal wall of the pump creates fluid pockets. As the gears are rotated, using the electrical motor coupled to the pump, the fluid in the pump is displaced into the outlet of the pump. The displaced fluid at the outlet of the pump generates compressibility flow due to the increased pressure change at the outlet. Gear pumps are designed for low pressure application, due to the contact

between gear's teeth in the pump.

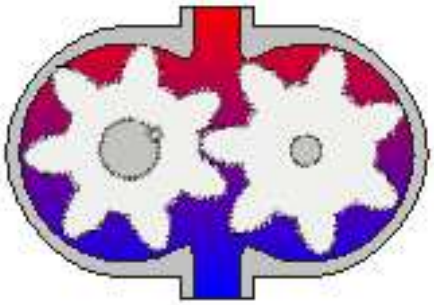

Figure 2.3: Gear Pump

2.2.2 Degradation

From the performance Equation 2.3 for a fixed displacement pump, it can be seen that the degradation in pump performance is due to internal and external leakage, which is a product of the pressure drop across the system. The internal leakage inside the pump is based on the tolerance of the gear teeth to the pump's inner housing, which acts like an orifice that is dependent on the pressure difference, flow area, and discharge coefficient. As a pump begins to wear out, the tolerance of the gears teeth to the pump's inner housing increase, which results in higher internal leakage. The external leakage is dependent on the drain port and pump connection fitting tolerances, which increase as the pump degrades. With the internal and external leakage increasing over the life of the pump, the ability for the system to provide the required pressure and flow to the system will decrease.

2.3 Voltage Controlled Servo-Valve

2.3.1 Performance

The voltage controlled servo-valve is the main component in a hydrostatic system, since it determines the flow and pressure into the hydraulic controlled servomotor. The servo-valve controls the flow into the motor based on an operator supplied input voltage across a solenoid coil that causes a displacement in a spool. Depending on the displacement of the spool and pressure drop across servo-valve, the flow rate into the hydraulic motor can be calculated based on the differential Equation 2.2 [7].

$$Q_{si} = C_d \cdot w \cdot x_v \sqrt{\frac{2}{\rho}(P_s - P_{si})} - (C_{im} + \frac{C_{em}}{2})(P_s - P_r) \tag{2.2}$$

where:

Q_{si} = Flow Rate at Inlet of Motor (in^3/sec)

Q_{so} = Flow Rate at Outlet of Motor (in^3/sec)

P_s = Supply Pressure (psi)

P_r = Return Pressure $psi)$

P_{si} = Pressure at Inlet of Motor (psi)

C_d = Discharge Coefficient

ρ = Density (lb/in^3)

w = Width of Port (in)

The design of the servo-valve used in this application is shown in Figure 2.4. The spool valve is positional controlled using a nozzle flapper that is tilted by the voltage difference between two solenoid coils. The nozzle flapper has a built in feedback system. The spool position is shifted based on the tilt of the nozzle flapper. A back pressure is generated that opposes the tilt of the nozzle flapper. The feedback in the system allows for positional control of the spool and improves the stability of the

system.

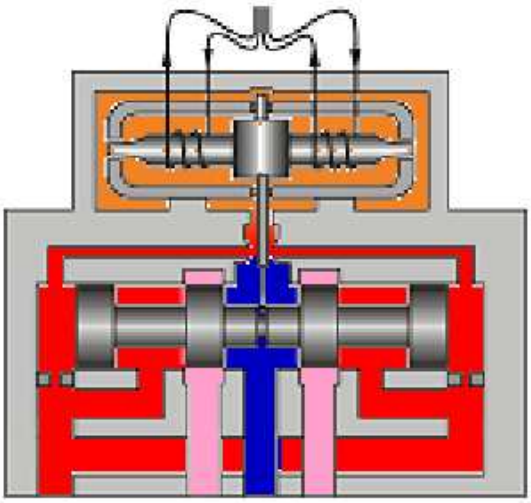

Figure 2.4: Voltage Controlled Servo Valve

2.3.2 Degradation

The common degradation in a servo-valve is internal and external leakage, due to tolerance increases in the spool and decay in the magnitude of the electromagnetic field in the solenoid coils. The internal leakage in the valve is based on tolerances between the supply and return port, which increase with wear in the servo-valve. The external leakage is based on the fittings of the port connection. The decay in the magnitude of the electromagnetic field can be from loss in supply current or wear in the solenoid coils. A degradation in spool tolerances, valve fittings, and electromagnetic field reduces the maximum operational flow rate and pressure at the inlet of the hydraulic controller motor.

2.4 Hydraulic Controlled Motor

2.4.1 Performance

A hydraulic controlled motor provides a torque to a load that generates an angular velocity in the gearbox. The motor is controlled based on both the pressure and flow at the inlet of the motor, which is maintained by the supply pump and servo-valve. A hydraulic controlled motor is designed to have blades that extend to the housing of the motor, which forms fluid pockets that are uniform in size. A hydraulic controlled motor is bidirectional, which can be velocity and torque controlled.

The performance of a hydraulic controlled motor is governed by Equations 2.3 and 2.4 [7]. The leakage in the motor is defined by two coefficients C_{im} and C_{em}, for internal leakage between blades inside the motor and external leakage through a drain port or pump fittings, respectively. Equation 2.3 includes the compressibility flow in the motor, which is proportional to the derivative of the pressure drop. The angular velocity of the motor is governed by Equation 2.4. The torque at the motor is based on the pressure drop across the system times the volumetric displacement of the motor. The differential equations show that the leakage in the motor reduces the pressure difference across the system, which decreases the magnitude of the torque load across the motor.

$$Q_L - D_m \cdot \dot{\theta}_s - (C_{im} + \frac{C_{em}}{2})P_L = \frac{V_0}{2\beta_e} \cdot \dot{P}_L \tag{2.3}$$

where:

$Q_L = \frac{Q_{si}+Q_{so}}{2}$ = Load Flow (in^3/sec)

Q_{si} = Flow Rate at Inlet of Motor (in^3/sec)

Q_{so} = Flow Rate at Outlet of Motor (in^3/sec)

D_m = Volumetric Displacement of Motor (in^3/rad)

C_{im} = Internal Leakage Coefficient $(in^3/sec/psi)$

C_{em} = External Leakage Coefficient $(in^3/sec/psi)$

$P_L = P_{si} - P_{so}$ = Load Pressure (psi)

P_{si} = Pressure at Inlet of Motor (psi)

P_{so} = Pressure at Outlet of Motor (psi)

$V_0 = V_{si} + V_{so}$ = Total Volume for both Inlet and Outlet of Motor (in^3)

β_e = Bulk Modulus of System Fluid (psi)

$$T_m = (P_{si} - P_{so})\dot{D}_m = J_t \cdot \ddot{\theta}_s + B_s \cdot \dot{\theta}_s + G \cdot \theta_s + T_L \tag{2.4}$$

where:

T_m = Torque Generated by Motor $(in \cdot lb)$

J_t = Inertia of Motor and Load $(in \cdot lb \cdot sec^2)$

B_m = Viscous Damping Coefficient $(in \cdot lb \cdot sec)$

G = Torsional Spring Torque on Motor $(in \cdot lb/rad)$

T_L = Torque Load on Motor $(in \cdot lb)$

2.4.2 Degradation

The degradation of the motor is based on the amount of internal and external leakage inside the motor. The internal leakage is based on the tolerance between the blades of the motor and the inside housing of the motor, while the external leakage is based on the drain port and pipe fitting tolerances. When both the internal and external leakage increase, the angular velocity of the motor will decrease based on Equation 2.3 and 2.4. The amount of leakage loss during operation of the motor is dependent on both wear and pressure drop across the motor. As the pressure drop increases across the motor, the level of degradation in the motor will increase, and the ability of the motor to provide the required angular velocity will decrease.

2.5 Proportionally Controlled Relief Valve

2.5.1 Performance

A proportionally controlled relief valve is designed to balance the pressure at the outlet of a fixed displacement pump. A relief valve uses a spool that is spring loaded, that when opened allows access pressure to be discharged. The valve is designed to open at a specific cracking pressure for a preset operating system flow rate. The cracking pressure for the relief valve is based on the spring stiffness and surface area of the spool (Equation 5) [8].

$$P_s \cdot A_s = K_s \cdot (x_0 + x_v) \tag{2.5}$$

where:

P_s = System Pressure (psi)

A_s = Spool Area (in^2)

K_s = Spring Stiffness (lb/in)

x_0 = Precompressed Spring Length (in)

x_v = Spool Displacement (in)

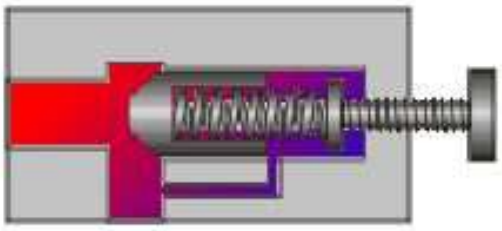

Figure 2.5: Proportionally Controlled Relief Valve

2.5.2 Degradation

The degradation in the relief valve is caused by wear in the spring stiffness and spool tolerance. When wear occurs in the spring stiffness, the cracking pressure for the relief value is reduced, and the maximum pressure at the outlet of the fixed displacement pump is reduced. Similarly, if the tolerances of the spool to the relief valve housing increase, the maximum pressure at the outlet of the fixed displacement will be reduced, due to internal leakage.

2.6 System Load

2.6.1 Performance

The angular velocity of the hydraulic controlled motor is based on torque of the system load and torque generated by the motor, which is governed by Equation 2.4.

2.6.2 Degradation

An increase in system load can generate degradation in the system if the torque load increases past the operational limits of the system. At increased system loads there will be additional wear on the gearbox and motor, which will reduce the angular velocity of the hydraulic controlled motor.

2.7 Common Degradation Types

In the sections above, performance degradation states for each sub-component in the hydrostatic system were discussed. In most cases, the degradation is caused by excessive wear. The common root causes of wear are air in the fluid (cavitation), increased contamination, water in working fluid, and high temperatures [8].

- Cavitation is created in a pump when the inlet pressure of the pump is greater than the outlet pressure, which causes an unsteady flow that cannot completely

fill the pump housing. The unsteady flow causes air pockets in the fluid that implode under high pressure, leading to erosion on the surface of the pump.

- Increased contamination in a hydraulic system can come from the operational environment, erosion of sub-components, clogging of filters, etc. When contamination increases, higher surface wear will occur.

- Water in the fluid can lead to corrosion wear, since metal is positively charged and will give up its excess energy by dissolving into water producing rust. Over time the surface of the metal in the vane pump will corrode, leading to increased tolerances in the system.

- High temperatures in a system will increase oxidation rates and contact wear in hydraulic parts, due to reduction in fluid viscosity.

CHAPTER 3

LINEAR MODELING OF HEALTHY SYSTEM

3.1 Introduction

Chapter 2 showed the differential equations for each sub-component in the hydrostatic transmission and their corresponding unhealthy state. To classify an unhealthy state, the severity of degradation needs to be measured from a known healthy state of the hydrostatic transmission. The healthy state will be based on the transfer function of the system, which is determined based on the dynamics of the hydraulic system. It is difficult to estimate the transfer function from the differential equations of the sub-components, due to the unknown design parameters for the system. A linear model will be estimated instead, using system identification techniques. Some common linear models are autoregressive exogenous (ARX), autoregressive moving average exogenous (ARMAX), and Box-Jenkins. In this report the Box-Jenkins model will be discussed, since it allows the dynamics and noise in the hydraulic system to be individually modeled.

The Box-Jenkins model is a widely used technique for linear modeling and has been implemented in a large range of applications; such as economics, process control, power grids, and etc. The benefit of the linear model is the knowledge that can be gained about the dynamics of the system. Most non-linear models are black box approaches, where very little insight can be gained about the dynamics of the system. The linear models ARX, ARMAX, and Box-Jenkins use past inputs and outputs to estimate the next output of the system. In most applications a linear model is preferred, but due to the complexity of physical systems their responses are

most often non-linear. A system can be operated in regions where the system behaves linearly, by keeping inputs and outputs within a small operating range. This chapter will linearize the servo-motor into a few small regions where linear models can be used.

3.2 Box-Jenkins Model

The steps of system identification and modeling are 1) preliminary identification, 2) parameter estimation, and 3) model validation [9]. Each of the steps will be discussed in the following sections, but the Box-Jenkins model will be presented first. The Box-Jenkins model is a single-input single-output (SISO) model. For the servo-motor, the input into the system is the voltage into the servo-valve that controls flow into the hydraulic motor, causing the hydraulic motor to rotate with an output angular velocity. The output is determined by two components. The first is the response of the system dynamic to the control input. The second is the system response to disturbances and noise, which can be from vibrations and noise in the hydraulic system. For the Box-Jenkins model the two components are combined. The Box-Jenkins model is shown in Equation 3.1 [10, pg.87] with the two components represented by $G(q)u(t)$ and $H(q)e(t)$. In the Box-Jenkins model it is assumed that the disturbance is independent of the input signal. This assumption holds if the system is modeled over small operational regions that are linear.

$$y(t) = G(q) \cdot u(t) + H(q) \cdot e(t) \tag{3.1}$$

where:

$G(q) = \frac{B(q)}{F(q)}$

$H(q) = \frac{C(q)}{D(q)}$

$B(q) = b_1 \cdot q^{-1} + b_2 \cdot q^{-2} + \cdots + b_{n_b} \cdot q^{-n_b}$

$F(q) = 1 + f_1 \cdot q^{-1} + f_2 \cdot q^{-2} + \cdots + f_{n_f} \cdot q^{-n_f}$

$C(q) = 1 + c_1 \cdot q^{-1} + c_2 \cdot q^{-2} + \cdots + c_{n_c} \cdot q^{-n_c}$

$D(q) = 1 + d_1 \cdot q^{-1} + d_2 \cdot q^{-2} + \cdots + d_{n_d} \cdot q^{-n_d}$

and q^{-1} represents a time delay

3.2.1 Preliminary Identification

The preliminary identification is performed by analyzing the impulse response, auto-correlation of the disturbances, and frequency response of the input and output data from the system to identify the orders of the transfer functions $G(q)$ and $H(q)$. The orders (n_b, n_f, n_c, and n_d) of the Box-Jenkins model (Equation 3.1) should match with the physical hydrostatic transmission system. If the orders of the system are chosen to be larger than the actual system, over-fitting can occur when estimating the parameters. However, if the orders of the system are chosen to be smaller than the actual system, then the Box-Jenkins model will be unable to capture the dynamics or disturbances of the actual system.

The impulse response in a linear time invariant (LTI) system defines the output signal's response for any input signal when there is no noise in the system. In this report the impulse response is used to estimate the time delay and orders (n_b and n_f) of the dynamic model ($G(q)$), which are defined by the parameters for $B(q)$ and $F(q)$ from Equation 3.1. The impulse response can be estimated from the cross-correlation of the input and output against the cross-correlation of the input signal (Equation 3.3) [9, pg.67], which measures the correlation of changes in the input signal against changes in the output signal.

$$y(t) = \sum_{k=1}^{T} g(t)u(t-k) + v(t) \tag{3.2}$$

where:

$y(t)$ = Output Signal

$g(t)$ = Impulse Response

$u(t)$ = Input Signal

$v(t) = H(q) \cdot e(t)$ = Random Process

$$g = R_u^{-1} \cdot r_{uy} \tag{3.3}$$

where:

$$R_u = \begin{bmatrix} R_{uu}(0) & R_{uu}(1) & \cdots & R_{uu}(k) \\ R_{uu}(1) & R_{uu}(2) & \cdots & R_{uu}(k-1) \\ \vdots & \vdots & \ddots & \vdots \\ R_{uu}(k) & R_{uu}(k-1) & \cdots & R_{uu}(0) \end{bmatrix}, g = \begin{bmatrix} g(0) \\ g(1) \\ \vdots \\ g(k) \end{bmatrix}, r_{uy} = \begin{bmatrix} R_{uy}(0) \\ R_{uy}(1) \\ \vdots \\ R_{uy}(k) \end{bmatrix} \tag{3.4}$$

$R_{uy}(k) = E\left[u[i]y[i-k]\right]$ = Cross-correlation of Input to Output Signal

$R_{uu} = E\left[u[i]u[i-k]\right]$ = Cross-correlation of Input Signal

The disturbance ($v(t)$) is estimated using Equation 3.5. Any part of the output response that does not correlate with the input will be part of the disturbance model ($H(q)$). The orders (n_c and n_d) of the disturbance model, which are defined by the parameters for $C(q)$ and $D(q)$ from Equation 3.1, can be estimated based on the slope of the auto-correlation of the disturbance signal (Equation 3.6).

$$v(t) = y(t) - \sum_{k=1}^{T} g(t)u(t-k) \tag{3.5}$$

$$R_k = E\left[v[i]v[i-k]\right] \tag{3.6}$$

The frequency response of the system is estimated based on the cross power spectrum between the input and output divided by the power spectrum of the input

(Equation 3.7) [10, pg.41]. An issue with the frequency response is that the dynamics and disturbance are mixed together, which makes estimating the orders (n_b and n_f) of the dynamic model ($G(q)$) for $B(q)$ and $F(q)$ difficult when using only the frequency response.

$$\widehat{H}_f = \frac{\widetilde{\Phi}_{xy}(f)}{\widehat{\Phi}_x(f)} \tag{3.7}$$

where:

$\widetilde{\Phi}_{xy}(f) = \frac{1}{T}X^*(f,T)Y(f,T)$

$\widehat{\Phi}_x(f) = \frac{1}{T}X^*(f,T)X(f,T)$

T = Number of Samples

In order to determine the orders and lags of $G(q)$ and $H(q)$ from $g(t)$ and $R_v(t)$, we will use the Generalized Partial Autocorrelation Function (GPAC). This is discussed by Woodward, Wayne A and Gray, and Henry L in "On the relationship between the S array and the Box-Jenkins method of ARMA model identification" [11]. We will provide examples in a later section.

3.2.2 Parameter Estimation

After the $G(q)$ and $H(q)$ model orders are estimated, the Box-Jenkins model parameters can be estimated using the Levenberg-Marquardt algorithm [12]. The Levenberg-Marquardt algorithm adjusts the Box-Jenkins model's parameters (Equation 3.8) to minimize the sum square prediction error residuals shown in Equation 3.9 [10, pg.219]. A linearized model of the error is shown in Equation 3.10, where the partial derivatives of error residuals with respect to the design parameters (θ) are estimated by a numerical approach as shown in Equation 3.11.

$$\theta = \begin{bmatrix} b_1 & \cdots & b_{n_b} & f_1 & \cdots & f_{n_f} & c_1 & \cdots & c_{n_c} & d_1 & \cdots & d_{n_d} \end{bmatrix}^T \tag{3.8}$$

$$\widehat{\theta}_{ML} = \underset{\theta}{\arg}\ min \sum_{t=1}^{n} \epsilon^2(t,\theta) \tag{3.9}$$

where:

$\epsilon(t,\theta)$ = Residual Prediction Errors

$$\epsilon(t,\theta) = \epsilon(t,\theta_0) + \sum_{i=1}^{n} \left[\frac{\partial \epsilon(t,\theta)}{\partial \theta_i}\right]_{\theta=\theta_0} (\theta_i - \theta_{0i}) \tag{3.10}$$

where:

n = Number of Parameters

$$\frac{\partial \epsilon(t,\theta)}{\partial \theta_i} = \frac{\epsilon(t, \begin{bmatrix} \theta_1 + h \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}) - \epsilon(t, \begin{bmatrix} \theta_1 - h \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix})}{2h} \tag{3.11}$$

The Levenberg-Marquardt process is an iterative approach that converges to a maximum likelihood estimate of the parameters ($\widehat{\theta}_{ML}$) based on the system's training data. The update rule for the Levenberg-Marquardt algorithm is shown by Equation 3.12, which uses the Jacobian matrix (Equation 3.13).

$$\widehat{\theta}(k+1) = \widehat{\theta}(k) + \left[J_k^T J_k + \mu I\right]^{-1} J_k^T \epsilon_k \tag{3.12}$$

where:

k = Iteration Step

$$J = \begin{bmatrix} \frac{\partial \epsilon(1,\theta)}{\partial \theta_1} & \frac{\partial \epsilon(1,\theta)}{\partial \theta_2} & \cdots & \frac{\partial \epsilon(1,\theta)}{\partial \theta_n} \\ \frac{\partial \epsilon(2,\theta)}{\partial \theta_1} & \frac{\partial \epsilon(2,\theta)}{\partial \theta_2} & \cdots & \frac{\partial \epsilon(2,\theta)}{\partial \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \epsilon(N,\theta)}{\partial \theta_1} & \frac{\partial \epsilon(N,\theta)}{\partial \theta_2} & \cdots & \frac{\partial \epsilon(N,\theta)}{\partial \theta_n} \end{bmatrix} \tag{3.13}$$

where:

N = Number of Training Points

3.2.3 Model Validation

Once the Box-Jenkins model parameters are estimated, model validation can be performed to determine if the estimated model is over-fitting or under-fitting the actual system. A few different methods can be implemented for model validation, such as pole zero cancellation, chi-square statistics, and autocorrelation of the predication errors. All methods will be discussed in this report, with the model validation being used to update the orders of the system found by preliminary identification. The process of using model validation to adjust the Box-Jenkins model orders and retraining of the model parameters is key to system identification. It allows the designer to continually adjust parameter orders until the model can achieve the best fit of the physical system.

The model validation technique of pole-zero cancellation is to remove the poles and zeros in the $G(q)$ or $H(q)$ models that are at the same location. If there is a pole-zero cancellation, the orders of the $G(q)$ or $H(q)$ are reduce by one, since the Box-Jenkins model is over-fitting. It is also possible to reduce the model order if the highest order parameter is close to zero. To determine if a parameter is near zero, we need to find confidence intervals for the parameters estimates, using Equation 3.14 [10, pg.217-218] to compute the covariance matrix of the parameters.

$$cov(\widehat{\theta}_{ML}) = \widehat{\sigma}_{\epsilon}^{2}\left[J^{T}(\widehat{\theta}_{ML})J(\widehat{\theta}_{ML})\right]^{-1} \tag{3.14}$$

where:

$\widehat{\sigma}_{\epsilon}^{2} = \frac{1}{N}\epsilon^{T}\epsilon$ = Variance of Residual Errors

Two chi-square test can be used to examine if $G(q)$ and $H(q)$ provide a good fit to the physical system [9, pg.338-343]. The Equations 3.17 and 3.18 for the Q-statistic and S-statistic are shown below. The Q-statistic determines if the residual errors are white noise, which indicates that the $H(q)$ is a good fit to the system. The S-statistic determines if the residual errors are uncorrelated with the input, which indicates that the $G(q)$ is a good fit to the system.

$$\widehat{R}_{\epsilon} = \frac{1}{N-k}\sum_{t=1}^{N-k}\epsilon(t)\epsilon(t+k) \tag{3.15}$$

$$\widehat{R}_{u\epsilon} = \frac{1}{N-k}\sum_{t=1}^{N-k}u(t)\epsilon(t+k) \tag{3.16}$$

$$Q = N\cdot\sum_{k=1}^{K}r_{\epsilon}^{2}(k) \quad \sim \quad \chi^{2}(k-n_{c}-nd) \tag{3.17}$$

where:

$r_{\epsilon}(k) = \frac{\widehat{R}_{\epsilon}(k)}{\widehat{R}_{\epsilon}(0)}$

$$S = N\cdot\sum_{k=1}^{K}r_{u\epsilon}^{2}(k) \quad \sim \quad \chi^{2}(k-n_{b}-nf) \tag{3.18}$$

where:

$r_{\epsilon}(k) = \frac{\widehat{R}_{u\epsilon}(k)}{\sqrt{\sigma_{u}^{2}\cdot\sigma_{\epsilon}^{2}}}$

If the model is correct, Q and S will satisfy chi-square distributions. Otherwise, their values will be inflated.

3.3 Data Collection and Sensor Location

In system identification using linear modeling, the accuracy of the estimated model parameters is dependent on the measured data. For a linear model to properly match the physical system the measured data needs to be obtained with minimal noise and must be rich enough to capture the full dynamics of the system. This section will discuss the sensor location, linear operational ranges, and frequency response.

3.3.1 Sensor Location

The location of the sensors determines if the response of a system can be directly measured and the amount of noise in the sensor. The Box-Jenkins model is a single-input single-output (SISO) model. The input in a hydrostatic transmission is the voltage into the servo-valve, which was discussed in Chapter 2. The voltage change in the servo-valve will create a displacement in the spool, which changes the flow and pressure at the outlet of the servo-valve. The flow and pressure at outlet of the servo-valve and inlet of the hydraulic controlled motor will control the gearbox angular velocity and torque, depending on the load of the system. The sensors are located to measure the key performance characteristics of the system, which are flow at the outlet of the servo-valve, angular velocity of the gearbox, and torque on the gearbox. For this chapter the healthy linear model output will be the angular velocity of the gearbox, but the system identification process was implemented for all three key performance characteristics.

3.3.2 Linear Operational Ranges

When using linear modeling approaches on a non-linear system, we need to constrain the operating range. For the hydrostatic transmission, we will consider two operating ranges, as shown in Table 3.1. Each range has a data sets with 10,000 sampled points. The Box-Jenkins model will trained on one data set.

Table 3.1: Servo-Valve Input Ranges

Operational Range	Minimum Voltage	Maximum Voltage	Sample Size	Data Sets
1	0	2	10000	1
2	2	4	10000	1

Each data set's input and outputs are normalized around zero using Equation 3.19 and 3.20. Normalizing the input and output improves training of the Box-Jenkins model and is required for implementing the techniques discussed above for system identification.

$$u(t) = u(t) - \frac{1}{N}\sum_{i=1}^{N} u(i) \tag{3.19}$$

$$y(t) = y(t) - \frac{1}{N}\sum_{i=1}^{N} y(i) \tag{3.20}$$

3.3.3 Frequency Response

To estimate the Box-Jenkins model the dynamics of the key performance characteristics need to be captured. To see the dynamics of the system there should be no low pass filters on the sensors to prevent attenuation and the sensors' sampling frequency needs to be more than twice the highest frequency in the dynamics of the system to satisfy the Nyquist criterion. The system frequency response can be estimated from Equation 3.7. The data can be collected while running white noise into the system, but pure white noise can lead to extreme changes that can possible harm the physical system. The alternative approach is to send in white noise through a first order low pass filter (Equation 3.21). By passing the white noise through a low pass filter, the extreme changes in the input are removed and the changes are more gradual, but

still random. This input will still be sufficiently exciting to reveal all of the system dynamics.

$$y(t) = \frac{0.9}{1 + 0.9 \cdot q^{-1}} u(t) \tag{3.21}$$

After running filtered white noise signal into the hydrostatic transmission the frequency response was measured and is shown in Figure 3.1 for the first operational range from Table 3.1.

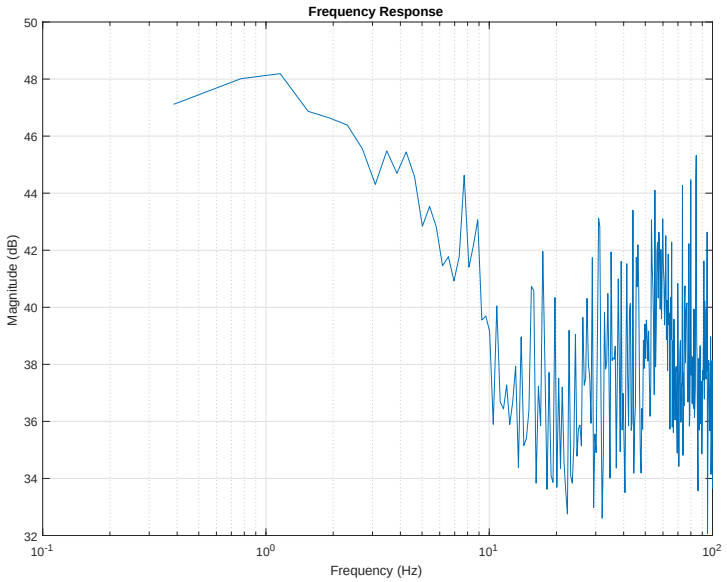

Figure 3.1: Frequency Response

3.4 Model Training and Validation

This section describes the system identification of a healthy hydrostatic transmission with angular velocity as the output. Two models will be estimated for the operational input range from Table 3.1 and their model performance will be discussed in the results section below. The process of system identification (preliminary identification, parameter estimation, and model validation) will be demonstrated for only the first operational input range from Table 3.1. Once the final Box-Jenkins models

are trained, a healthy reference state for the hydrostatic transmission can be used to measure level of degradation in the system.

3.4.1 Preliminary Identification

Preliminary Identification is performed on the servo-valve first operational input range (Table 3.1) for the hydrostatic transmission. The white noise input voltage into the servo-valve and the measured angular velocity in the gearbox is shown in Figure 3.2 below. The input voltage and output angular velocity are normalized around zero using Equations 3.19 and 3.20.

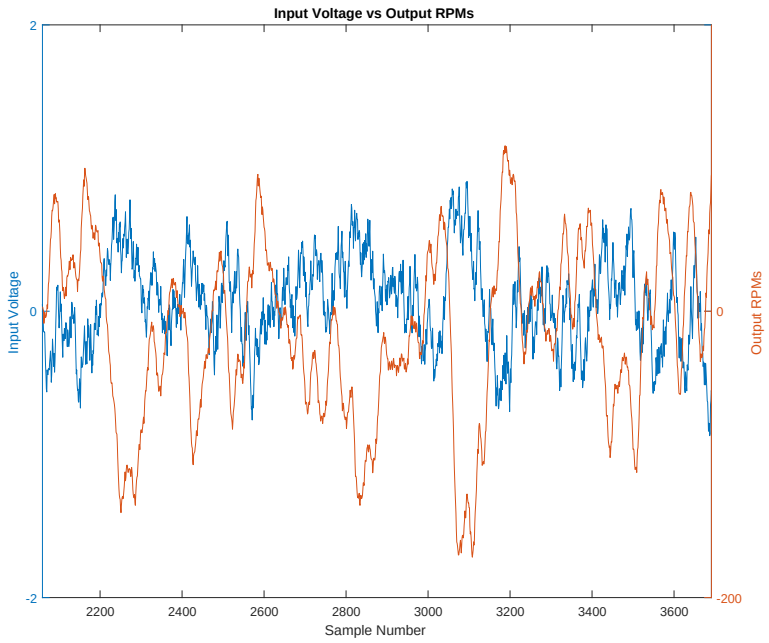

Figure 3.2: Input and Output Training Data

Using the input voltage and output angular velocity the impulse response can be estimated using Equation 3.3. The impulse response is shown in Figure 3.3 below. The impulse response shows a lag of 4 samples at 200Hz, which means the time lag is 0.02 seconds. The lag time in the system is indicated in Figure 3.3 by the number of samples at zero before the response begins. The impulse response also indicates that

the hydrostatic transmission is at least a second order system.

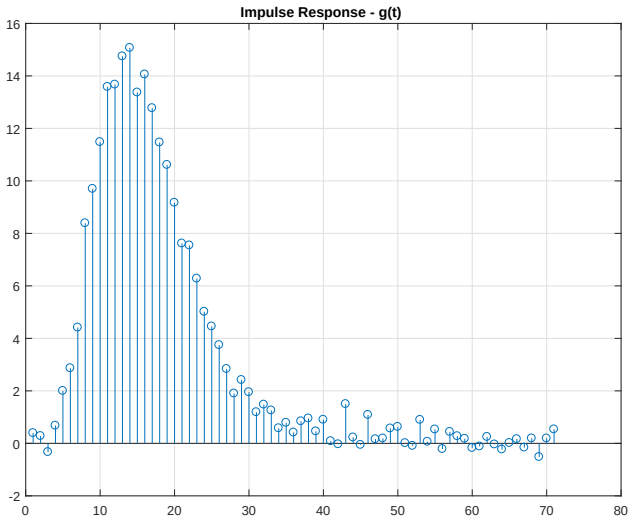

Figure 3.3: Impulse Response

The $G(q)$ orders (n_b and n_f) can be estimated from the G-GPAC, which is shown in the Figure 3.4 below. The B(q) and F(q) order sizes (n_b and n_f) can be estimated from the G-GPAC by looking for a constant column with a row of zeros to the right of the first value of the constant column. There is a noticeable pattern at $n_b = 4$ and $n_f = 4$ (row 8 and column 4).

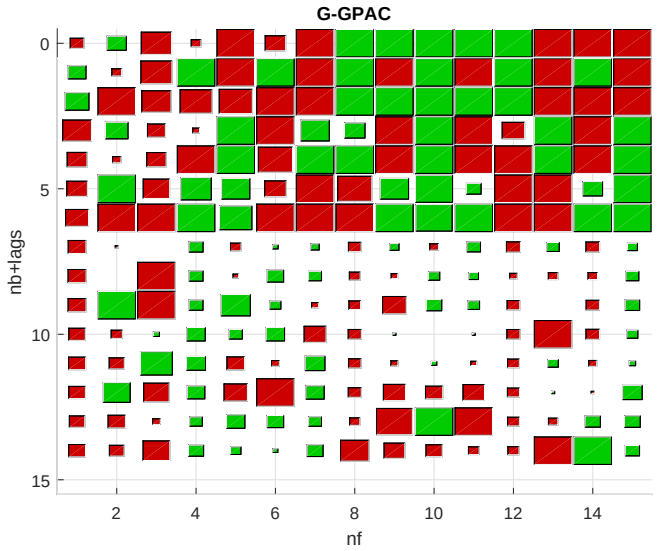

Figure 3.4: Hydrostatic Transmission G-GPAC

The $H(q)$ orders (n_c and n_d) can be estimated from the GPAC of the estimated disturbance's autocorrelation found by using Equations 3.5 and 3.6, which is called the H-GPAC. The autocorrelation of the disturbance is shown in Figure 3.5 and the H-GPAC is shown in Figure 3.6. From Figure 3.5 the autocorrelation of the disturbance does not decay to zero, since there is a periodic noise signal. This will make using the H-GPAC difficult for estimating the orders of n_c and n_d. The order sizes of $C(q)$ and $D(q)$ can be estimated in the H-GPAC based on a constant column of values with a row of zeros right of the first value in the constant column. For $C(q)$ and $D(q)$ there is one noticeable pattern at $n_c = 1$ and $n_d = 3$ (row 1 and column 3).

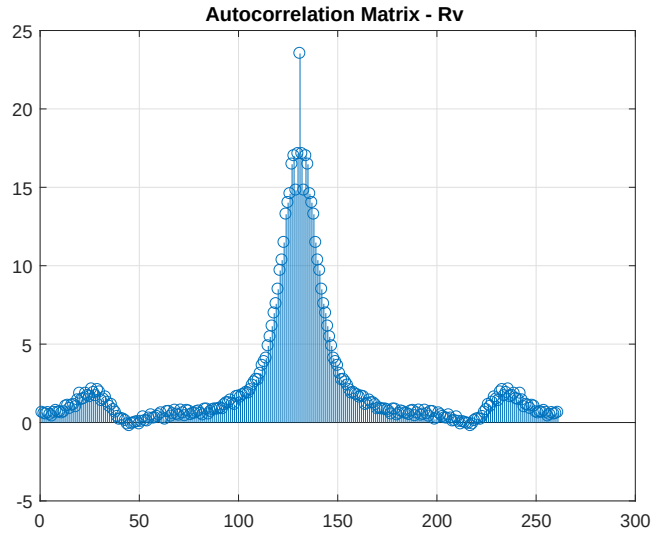

Figure 3.5: Hydrostatic Transmission Autocorrelation of Disturbance

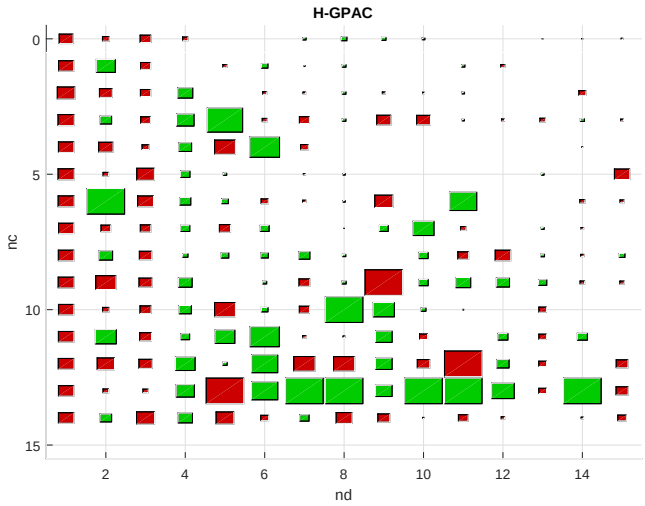

Figure 3.6: Hydrostatic Transmission H-GPAC

3.4.2 Preliminary Parameter Estimation

From preliminary identification, the system orders for the Box-Jenkins model are shown in Table 3.2 and the Box-Jenkins model is shown by Equation 3.22. The Box-Jenkins model parameters are estimated using the Levenberg-Marquardt algorithm discussed in the prior section.

$$y(t) = q^{-4} \cdot \frac{b_1 \cdot q^{-1} + b_2 \cdot q^{-2} + b_3 \cdot q^{-3} + b_4 \cdot q^{-4}}{1 + f_1 \cdot q^{-1} + f_2 \cdot q^{-2} + f_3 \cdot q^{-3} + f_4 \cdot q^{-4}} u(t) + \frac{1 + c_1 \cdot q^{-1}}{1 + d_1 \cdot q^{-1} + d_2 \cdot q^{-2} + d_3 \cdot q^{-3}} e(t) \tag{3.22}$$

Table 3.2: Preliminary Identification System Orders for Box-Jenkins Model

n_b	n_f	n_c	n_d	Delays
4	4	1	3	4

The trained Box-Jenkins model is shown in Equation 3.23 below.

$$y(t) = q^{-4} \cdot \frac{1.5916 \cdot q^{-1} + 0.6997 \cdot q^{-2} - 0.4507 \cdot q^{-3} + 0.9412 \cdot q^{-4}}{1 - 1.6400 \cdot q^{-1} - 0.1340 \cdot q^{-2} + 1.3780 \cdot q^{-3} - 0.5916 \cdot q^{-4}} u(t) + \frac{1 - 0.5600 \cdot q^{-1}}{1 - 0.7518 \cdot q^{-1} + 0.1212 \cdot q^{-2} - 0.2251 \cdot q^{-3}} e(t) \tag{3.23}$$

3.4.3 Model Validation

The estimated Box-Jenkins model, shown by Equation 3.23, can be tested using the Q-statistic and S-statistic for goodness of fit. The Q-statistic and S-statistic are shown in Table 3.3. Based on the Q-statistic of 394.523 with a 31 degrees of freedom, $H(q)$ is under-fitting, since the Chi Square Distribution's 0.05 percentile is 44.9853, which is smaller than the Q-statistic. The $H(q)$ orders (n_c and n_d) need to be increased. The S-statistic is 11.2891 with 27 degrees of freedom, which indicates the $G(q)$ is a good fit, since the S-statistic is below the Chi Square Distribution's 0.05 percentile of 40.1133.

Table 3.3: Model Validation Q-statistic and S-statistic for Original Model

Q-statistic	S-statistic
394.523	11.2891
Degrees of Freedom	Degrees of Freedom
31	27

The order for $H(q)$ can be increased to $n_c = 4$ and $n_d = 5$, which produces a Q-statistic shown in Table 3.4 after retraining the Box-Jenkins model with the new system orders. Although the chi-square test is not quite satisfied, this was the smallest Q value that was obtained over several sets of orders.

Table 3.4: Model Validation Q-statistic and S-statistic for Updated Model

Q-statistic	S-statistic
89.7085	11.2891
Degrees of Freedom	Degrees of Freedom
26	27

With the model order being adequate, the next step is to see if the orders can be reduced to prevent over-fitting by checking to there is pole zero cancellation in $G(q)$ and $H(q)$. From examination of the zeros of $B(q)$ and poles of $F(q)$ in Table 3.5 there is no pole zero cancellation, so the orders of n_b and n_f can stay at 4 and 4, respectively. The same can be seen for the zeros of $C(q)$ and poles of $D(q)$ in Table 3.6, where there is no pole zero cancellation and the orders of n_c and n_d are adequate at 4 and 5, respectively.

Table 3.5: $B(q)$ and $F(q)$ Roots

$B(q)$ Zeros	$F(q)$ Poles
-1.1498 + 0.000i	-0.9117 + 0.0000i
0.3733 + 0.6645i	0.8907 + 0.0000i
0.3733 - 0.6645i	0.8278 + 0.1770i
	0.8278 - 0.1770i

Table 3.6: $C(q)$ and $D(q)$ Roots

$C(q)$ Zeros	$D(q)$ Poles
0.6524 + 0.4132i	-0.1284 + 0.6857i
0.6524 - 0.4132i	-0.1284 - 0.6857i
-0.0445 + 0.6213i	0.8353 + 0.4372i
-0.0445 - 0.6213i	0.4065 + 0.4372i
	0.4065 - 0.4372i

With there being no pole zero cancellation we can check to see if some parameters are near zero. The confidence limits for the parameters are shown in Tables 3.7, 3.9, 3.10, and 3.8 below. Based on the confidence limits, the $B(q)$ and $F(q)$ orders can remain the stay at $n_b = 4$ and $n_f = 4$, since b_4 and f_4 upper and lower limits do not straddle zero. The confidence limits for $C(q)$ and $D(q)$ show that the orders n_c and n_d can also stay the same at 4 and 5, respectively, since none of the confidence intervals include zero.

Table 3.7: $B(q)$ Confidence Limits

	Lower Limit	θ	Upper Limit
b_1	1.2196	1.6051	1.9996
b_2	-0.0492	0.6470	1.3432
b_3	-1.2206	-0.4455	0.3296
b_4	0.5530	1.0721	1.5912

Table 3.8: $F(q)$ Confidence Limits

	Lower Limit	θ	Upper Limit
f_1	-1.9638	-1.6347	-1.3056
f_2	-0.9730	-0.1301	0.7128
f_3	0.6282	1.3594	2.0906
f_4	-0.7971	-0.5819	-0.3667

Table 3.9: $C(q)$ Confidence Limits

	Lower Limit	θ	Upper Limit
C_1	-1.3952	-1.2158	-1.0365
C_2	0.6419	0.8683	1.0946
C_3	-0.5800	-0.4533	-0.3265
C_4	0.1680	0.2314	0.2949

Table 3.10: $D(q)$ Confidence Limits

	Lower Limit	θ	Upper Limit
D_1	-1.5708	-1.3915	-1.2123
D_2	0.8456	1.0989	1.3522
D_3	-0.9864	-0.8340	-0.6815
D_4	0.3581	0.4274	0.4968
D_5	-0.1736	-0.1449	-0.1161

The final Box-Jenkins model is shown in Equation 3.24 below, using the system orders shown in Table 3.11. The autocorrelation of the residual errors is calculated using Equation 3.15 and is shown in Figure 3.7. The autocorrelation of the residual errors is an impulse, which means there is no correlation in the errors, and the Box-Jenkins model in Equation 3.24 is a good fit for the training data.

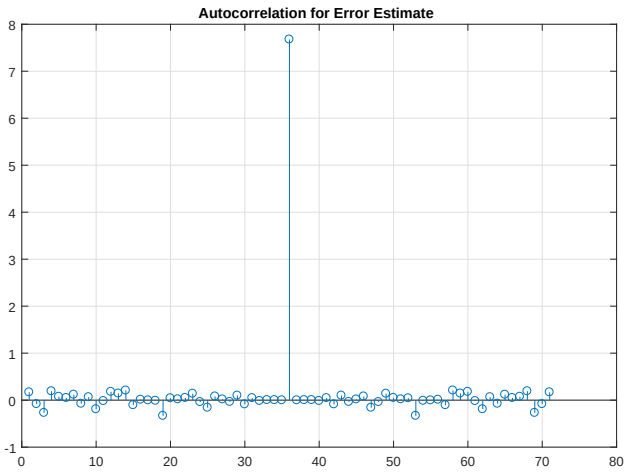

Figure 3.7: Autocorrelation of the Residual Errors for Equation 3.24

The final tuned model parameters tuned shown in Equation 3.24, and they adequately fit the physical model. The stability of the model can be checked by the examining the poles and zeros for $G(q)$ and $H(q)$. For a Box-Jenkins model to be

stable the poles (Table 3.5 and 3.6) for $G(q)$ and $H(q)$ must be inside the unit circle of the z-plane. The z-plane for $G(q)$ and $H(q)$ is shown in Figure 3.8 and 3.9. Based on Figure 3.8 and 3.9 the final Box-Jenkins model is stable, since all ploles are within the unit circle.

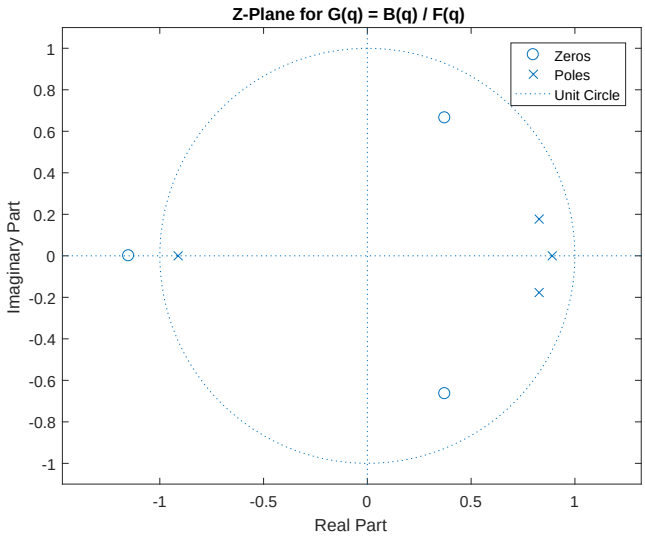

Figure 3.8: Z-Plane for $G(q)$

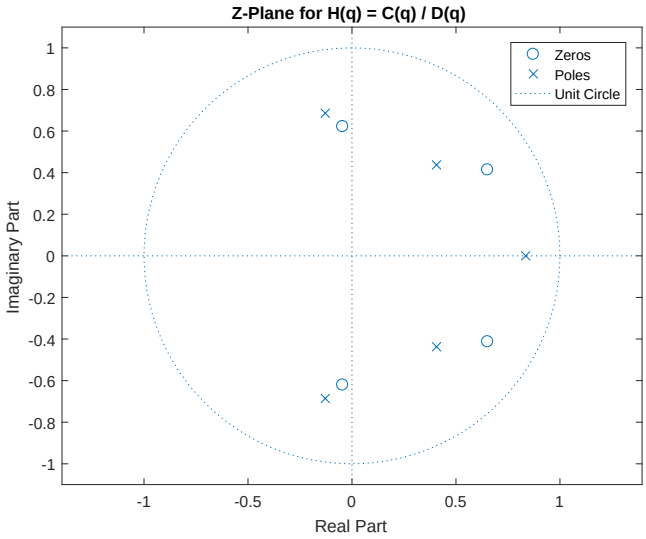

Figure 3.9: Z-Plane for $H(q)$

3.4.4 Final Parameter Estimation

After model validation, the final model orders for the Box-Jenkins model are shown in Table 3.11 below. The final Box-Jenkins model for the first operation range is shown by Equation 3.24 and by 3.25 for the second operational range. Both operational ranges have the same Box-Jenkins model orders found by model validation.

Table 3.11: Final System Orders for Box-Jenkins Model

n_b	n_f	n_c	n_d	Delays
4	4	4	5	4

$$y(t) = q^{-4} \cdot \frac{1.6051 \cdot q^{-1} + 0.6470 \cdot q^{-2} - 0.4455 \cdot q^{-3} + 1.0721 \cdot q^{-4}}{1 - 1.6347 \cdot q^{-1} - 0.1301 \cdot q^{-2} + 1.3594 \cdot q^{-3} - 0.5819 \cdot q^{-4}} u(t) + \frac{1 - 1.2158 \cdot q^{-1} + 0.8683 \cdot q^{-2} - 0.4533 \cdot q^{-3} + 0.2314 \cdot q^{-4}}{1 - 1.3915 \cdot q^{-1} + 1.0989 \cdot q^{-2} - 0.8340 \cdot q^{-3} + 0.4274 \cdot q^{-4} - 0.1449 \cdot q^{-5}} e(t) \tag{3.24}$$

$$y(t) = q^{-4} \cdot \frac{1.0243 \cdot q^{-1} + 0.4649 \cdot q^{-2} - 0.3376 \cdot q^{-3} + 0.4326 \cdot q^{-4}}{1 - 1.7134 \cdot q^{-1} - 0.0310 \cdot q^{-2} + 1.3646 \cdot q^{-3} - 0.6099 \cdot q^{-4}} u(t) + \frac{1 - 0.1709 \cdot q^{-1} + 0.3986 \cdot q^{-2} - 0.8507 \cdot q^{-3} + 0.0504 \cdot q^{-4}}{1 - 0.4255 \cdot q^{-1} + 0.1185 \cdot q^{-2} - 0.9325 \cdot q^{-3} + 0.1360 \cdot q^{-4} + 0.2486 \cdot q^{-5}} e(t) \tag{3.25}$$

Let's examine the performance of the first operational range model (Equation 3.24). The estimated impulse response is shown in the Figure 3.10, where the impulse response of Equation 3.24 is shown in red and the impulse response estimated from the training data using Equation 3.3 is shown in blue. The impulse response of Equation 3.3 is a close fit and is able to capture the dynamics of the physical system.

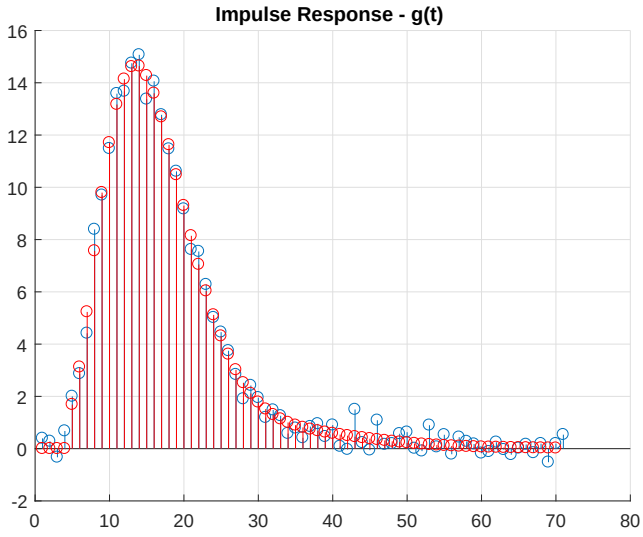

Figure 3.10: Estimated Impulse Response for Equation 3.24

The estimated frequency response for Equation 3.24 is shown in Figure 3.11. Figure 3.11 shows the frequency response for the impulse model ($G(q)$) (blue line in figure), disturbance model ($H(q)$) (orange line in figure), and the Box-Jenkins model (yellow line in figure), which combines the $G(q)$ and $H(q)$ models based on Equation 3.1. The estimated Box-Jenkins frequency response correlates with the estimated frequency response in Figure 3.1 with the high frequencies being modeled by the disturbance model ($H(q)$). This figure indicates how the Box-Jenkins model combines the impulse and disturbance model to estimate the physical system.

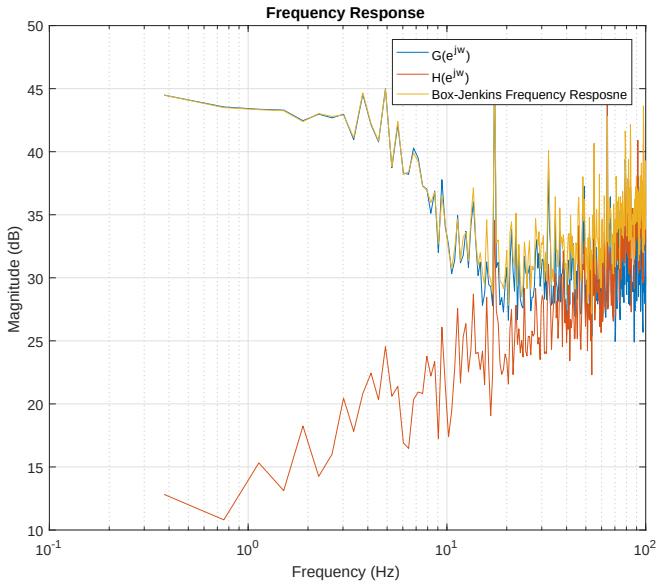

Figure 3.11: Estimated Frequency Response for Equation 3.24

3.5 Implement Box-Jenkins Model

With the Box-Jenkins models trained (Equation 3.24 and 3.25) for the operational ranges from Table 3.1, the models can be used for one step ahead prediction using Equation 3.26 [10, pg.87]. The predication error can be completed with Equation 3.27. The magnitude of the residual errors will indicate the level of degradation in a healthy control system. This concept will be discussed in-depth in the following chapters, with the key performance characteristics of the system (flow at the hydraulic motor, angular velocity at the gearbox, and torque at the gearbox) modeled using system identification techniques introduced in this chapter. Figure 3.12 shows the one step ahead prediction using the Box-Jenkins model from Equation 3.24. The Box-Jenkins model is able to accurately predict the next step in the hydrostatic transmission for a healthy system, which can be implemented as a reference state for a healthy system.

$$\widehat{y}(t|\theta) = \frac{D(q)B(q)}{C(q)F(q)}u(t) + \left[1 - \frac{D(q)}{C(q)}\right]y(t) \tag{3.26}$$

$$\epsilon(t) = y(t) - \widehat{y}(t|\theta) \tag{3.27}$$

where:

$\widehat{y}(t|\theta)$ = Estimated One Step Ahead Prediction

$y(t)$ = Measured Output

$\epsilon(t)$ = Residual Error

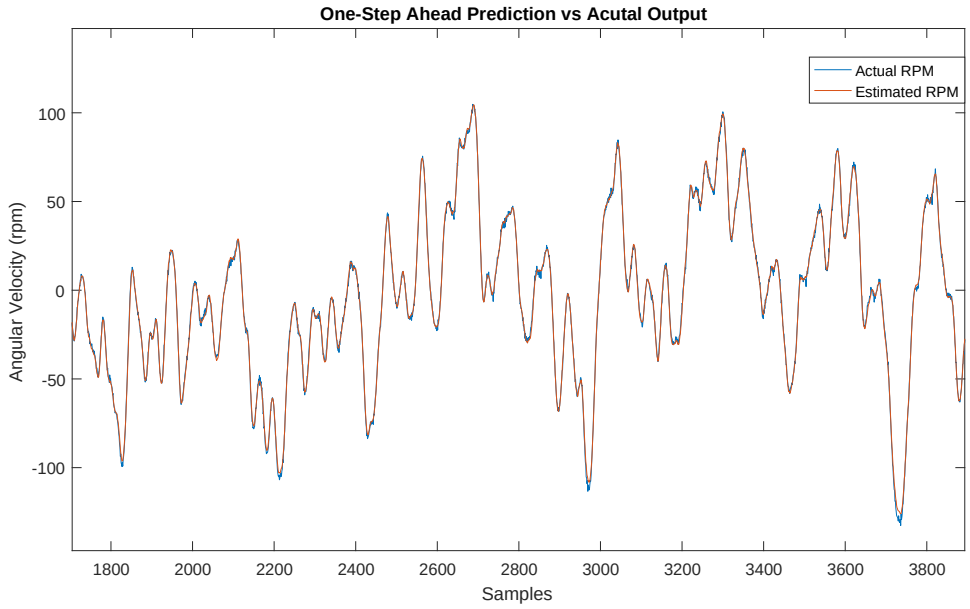

Figure 3.12: Estimated One Step Ahead Prediction for Equation 3.24

CHAPTER 4

NON-LINEAR MODELING OF HEALTHY SYSTEM

4.1 Introduction

Chapter 3 demonstrated linear modeling of the hydrostatic transmission discussed in Chapter 2. The purpose of the linear model was to create a healthy reference state for classifying an unhealthy state based on the degradation in the hydrostatic transmission system. The issue with linear modeling is that the hydrostatic tranmission has some nonlinearities. This required multiple linear models to be trained over smaller operational ranges where the system behaves linearly. Another approach that this chapter will present is to use non-linear modeling techniques; such as the non-linear autoregressive exogenous (NARX) model, which can capture the full dynamics of the system for the entire operational space.

The NARX approach is known as a black box technique, where the model is able to capture the dynamics of the system, but the parameters of the model do not have a physical meaning. The benefit of a non-linear model is that the entire operational range and non-linearity in the system can be captured by a single model. By including the non-linearity of the system in the model the prediction errors of the healthy reference state can be reduced. The weakness of the NARX model is that the weights of the model cannot be used to measure accuracy and do not present information about the model's goodness of fit. Also, there are no preliminary identification techniques, so finding the proper model complexity is reduced to trial and error approaches.

4.2 Neural Network Background

A NARX model is a recursive neural network that uses tapped-delay lines of past inputs and outputs to make a one step ahead prediction of the next output. This is similar to the linear models discussed in Chapter 3. The training of a neural network is more complex than the training of a linear model and requires an understanding of the background notation of a neural network. This section will present the single-input network, multiple-input network, feedforward network, and the backpropagation algorithm for training of a static neural network. The basic notation from this chapter provides the building blocks for implementing and training of a NARX model. The notation and principles presented in this chapter are based on [13].

4.2.1 Single-Input and Multiple-Input Networks

The basic notation for a single-input network is shown in Figure 4.1, with the output (a) being calculated using Equation 4.1. The transfer function (f) can be a linear or non-linear function. A common non-linear function is the log-sigmoid, which is shown in Figure 4.2 and Equation 4.2. The weight (w) and bias (b) are parameters that can can be tuned during training of the network.

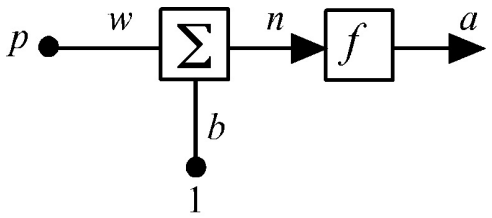

Figure 4.1: Single-Input Network

$$a = f(w \cdot p + b) \tag{4.1}$$

where:

a = Neuron Output

w = Neuron Weight

b = Neuron Bias

p = Neuron Input

f = Transfer Function

The log-sigmoid transfer function, shown in Figure 4.2, converts the transfer function input ($n \in \mathbb{R}$) between 0 and 1. The log-sigmoid is also differentiable, which is an important property that will be discussed in the training of a neural network in the following sections.

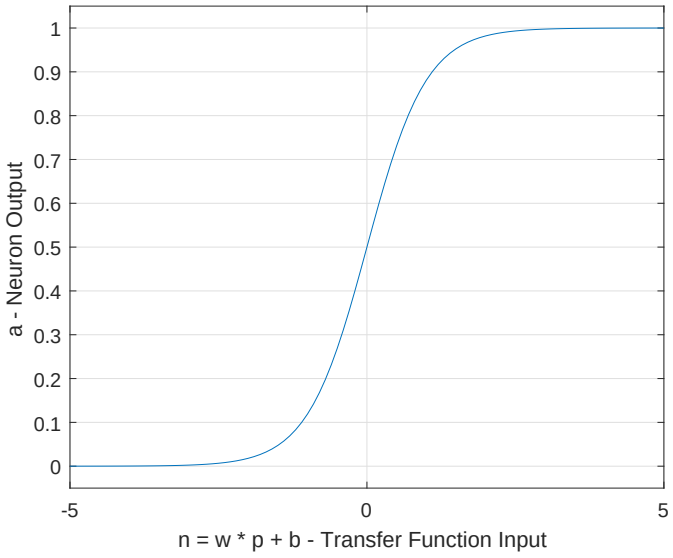

Figure 4.2: Log-Sigmoid Transfer Function

$$a = \frac{1}{1 + e^{-n}} \tag{4.2}$$

where:

$n = w \cdot p + b$ = Transfer Function Input

The single-input network can be expanded to include multiple inputs as shown

in Figure 4.3. For a multiple-input network the output can be solved for by using Equation 4.3, which is an extension of the single-input equation. Equation 4.3 can be written in matrix form based on Equation 4.4.

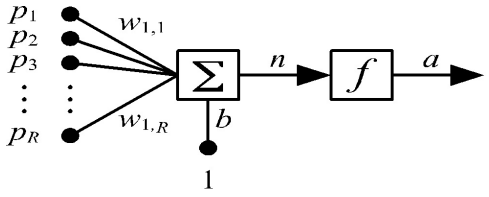

Figure 4.3: Multiple-Input Network

$$a = f\left(\sum_{i=1}^{R} w_{1,i} \cdot p_i + b\right) \tag{4.3}$$

$$a = \mathbf{f}\left(\mathbf{W}\mathbf{p} + b\right) \tag{4.4}$$

where:

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \end{bmatrix} = \text{Weight Array}$$

$$\mathbf{p} = \begin{bmatrix} p_1 & p_2 & \cdots & p_R \end{bmatrix}^T = \text{Input Array}$$

For both the single-input and multiple-input network, the network has only one neuron. The complexity of the network can be increased by adding additional neurons.

4.2.2 Feedforward Network

Based on Figure 4.3, the network complexity can be increased by adding additional neurons, as shown in Figure 4.4. The outputs of a multiple-input multiple-output networks can be calculated using Equation 4.5. In Figure 4.4 the networks has S neurons and R inputs. Every input (p) is connected to each of the neurons in the network, which creates a weight matrix ($\mathbf{W}$) that has the dimensions of R by S.

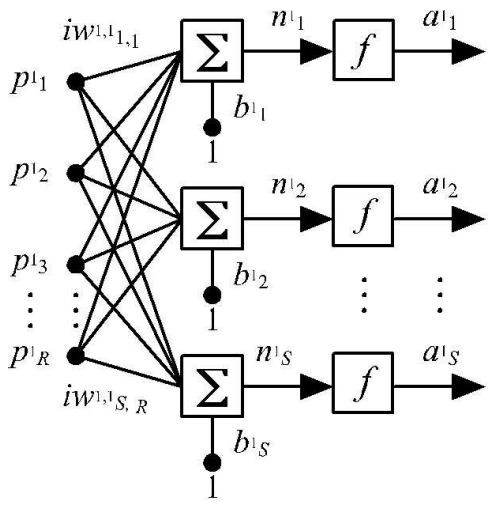

Figure 4.4: Multiple-Input Multiple-Output

$$\mathbf{a}^1 = \mathbf{f}\left(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1\right) \tag{4.5}$$

where:

$$\mathbf{W}^1 = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} = \text{Weight Matrix}$$

$$\mathbf{p} = \begin{bmatrix} p^1_1 & p^1_2 & \cdots & p^1_R \end{bmatrix}^T = \text{Input Array}$$

$$\mathbf{b}^1 = \begin{bmatrix} b^1_1 & b^1_2 & \cdots & b^1_S \end{bmatrix}^T = \text{Bias Array}$$

$$\mathbf{a}^1 = \begin{bmatrix} a^1_1 & a^1_2 & \cdots & a^1_S \end{bmatrix}^T = \text{Output Array}$$

Networks in Figure 4.4 can be stacked to create more layers. This type of network is known as a feedforward network. In a feedforward network, the output ($\mathbf{a}$) becomes the input for the next layer, as shown in the two layer neural network in Figure 4.5

and Equation 4.6. The two layer neural network will be expanded in the future when introducing the neural network non-linear autoregressive exogenous (NARX) model. Additional layers can be added to a neural network, but it has been shown that a two layer network is a universal approximator. The network architecture is defined by the number of layers, neurons in each layer, and the transfer function at each layer. The feedforward network is a static network and a directed graph model where there are no recurrent layers. Since the weight, bias, and inputs are matrices, the network illustrated by Figure 4.5 can be simplified to Figure 4.6. In the future, Figure 4.6 will be used to show the neural network architecture..

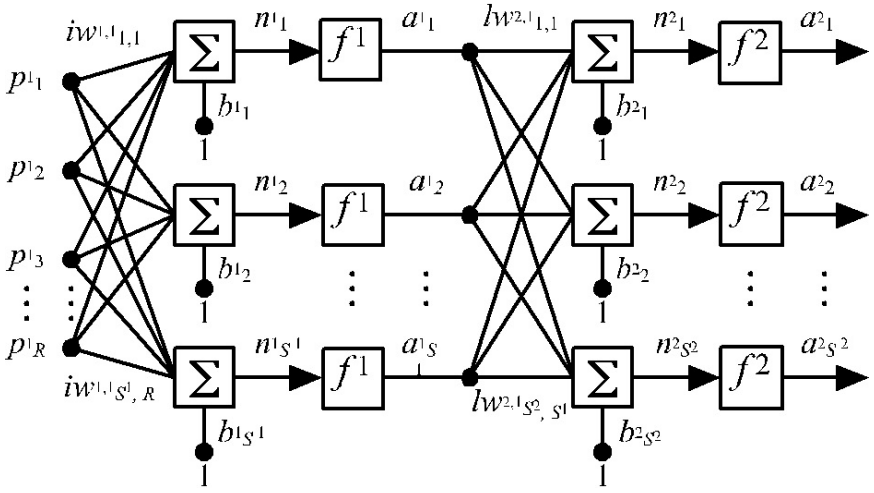

Figure 4.5: Two Layer Network

$$\mathbf{a}^2 = \mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1) + \mathbf{b}^2) \tag{4.6}$$

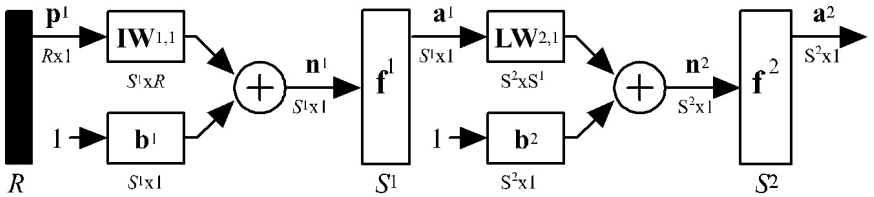

Figure 4.6: Simplified Two Layer Network

Based on Equation 4.1 and 4.6, a generalized algorithm can be used for calculating the output ($\mathbf{a}^M$) for any feedforward network architecture. The generalized algorithm for performing forward propagation through a feedforward network is shown by Equation 4.7 [13, chp.11 pg.26].

$$\mathbf{a}^0 = \mathbf{p} \tag{4.7}$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}) \text{ for m = M - 1,..., 2, 1}$$

$$\mathbf{a} = \mathbf{a}^M$$

4.2.3 Backpropragation Algorithm

In the previous sub-sections the forward propagation algorithm for a static network was presented. The output of the static network is dependent on the inputs, weights, and bias variables, where the weight and bias are tuned by minimizing the mean square error performance index (Equation 4.8) [13, chp.11 pg.25] using stochastic gradient descent shown by Equation 4.9 and 4.10 [13, chp.11 pg.26].

$$\hat{\mathbf{F}}(x) = (\mathbf{t}(k) - \mathbf{a}(k))^T(\mathbf{t}(k) - \mathbf{a}(k)) = \mathbf{e}^T(k)\mathbf{e}(k) \tag{4.8}$$

where:

$\mathbf{a}$ = Output Array

$\mathbf{t}$ = Training Points Array

$$w^m_{i,j}(k+1) = w^m_{i,j}(k) - \alpha\frac{\partial\hat{\mathbf{F}}}{\partial w^m_{i,j}} \tag{4.9}$$

$$b^m_i(k+1) = b^m_i(k) - \alpha\frac{\partial\hat{\mathbf{F}}}{\partial b^m_i} \tag{4.10}$$

where:

α = Learning Rate

The partial derivatives of the transfer function with respect to the weight and bias in Equation 4.9 and 4.10 are calculated using the chain rule shown by Equation 4.11 and 4.12 below [13, chp.11 pg.9]. From the chain rule, the partial derivative of the performance function (F) with respect to the input of the transfer function (n) is known as the sensitivity (s).

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m} \tag{4.11}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m} \tag{4.12}$$

where:

$s_i^m = \frac{\partial \hat{F}}{\partial n_i^m}$ = Sensitivity

From the chain rule, the sensitivity (s) can be calculated for each neural network layer using the backpropagation algorithm (Equation 4.13) [13, chp.11 pg.26]. The sensitivity (s) can then be used to update the weight ($\mathbf{W}$) and bias ($\mathbf{b}$) based on the stochastic gradient descent algorithm (Equation 4.14) [13, chp.11 pg.26]. This approach to training of the weights and biases for the feedforward network is an iterative approach, where the rate of convergence is dependent on the learning rate (α).

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a}) \tag{4.13}$$

$$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1} \text{ for m = M - 1,..., 2, 1}$$

where:

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(n_2^m) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{f}^m(n_{S^m}^m) \end{bmatrix}$$

$\dot{f}^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m}$

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m(\mathbf{a}^{m-1})^T \tag{4.14}$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m$$

where:

α = Learning Rate

4.3 NARX Model

After presenting the basic notation and training algorithm for feedforward neural networks in the previous section, this section will introduce recurrent neural networks for time series modeling. The specific recurrent neural network that will be introduced in this chapter is the non-linear autoregressive exogenous (NARX) model, which is a non-linear variation of the linear autoregressive exogenous (ARX) model. An ARX model uses past inputs and outputs from the dynamic system to predict the next output. The linear ARX model is shown by Equation 4.15 [9].

$$y(t) = \frac{B(q)}{A(q)} \cdot u(t) + \frac{1}{A(q)} \cdot e(t) \tag{4.15}$$

where:

$B(q) = b_1 \cdot q^{-1} + b_2 \cdot q^{-2} + \cdots + b_{n_b} \cdot q^{-n_b}$

$A(q) = 1 + a_1 \cdot q^{-1} + a_2 \cdot q^{-2} + \cdots + a_{n_a} \cdot q^{-n_a}$

The difference between an ARX and a Box-Jenkins model is that the noise in the system can not be separated from the dynamics of the system. This issue can be seen in Equation 4.15 where the disturbance model's $(1/A(q))$ optimization parameters $(A(q))$ are included in the dynamic model $(B(q)/A(q))$. The NARX model has the same issue has the ARX model of begin unable to separate the noise and dynamics, but the NARX model is able to capture the non-linear dynamics in the system that ARX and Box-Jenkins models cannot capture. The NARX model is a black box approach where the weights in the neural network model do not have a direct correlation to the differential equations for the sub-components of the hydrostatic transmission presented in Chapter 2.

This section will first present the neural network architecture for performing function approximation. Then the NARX architecture will be presented, along with the corresponding backpropagation algorithm and several model validation approaches.

4.3.1 Function Approximation

A common application for neural networks is function approximation. This generally uses a two layer neural network (Figure 4.7) [13, chp.11 pg.14] with the first layer being a log-sigmoid and a linear output layer. These types of neural networks have have been implemented in control systems for static modeling of systems. The multiple-layer neural network can be trained using the backproprogation algorithm shown by Equation 4.13 and 4.14.

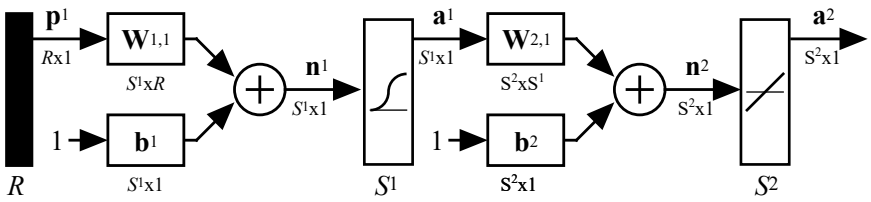

Figure 4.7: Multiple-layer Function Approximation

To implement the backpropagation algorithm for the network shown in Figure 4.7, the derivative of the transfer function (f^m) with respect the transfer function input (n^m) must be calculated for each layer in the neural network. The derivative of the log-sigmoid (Equation 4.2) transfer function for the first layer is shown by Equation 4.16, and the derivative of the linear transfer function is shown by Equation 4.17. Using Equations 4.16 and 4.17, the matrix $\hat{\mathbf{F}}^m$ can be derived, which allows each layer's sensitivities (s^m) to be calculated for performing stochastic gradient descent (Equation 4.14).

$$f_i^1 = sigmoid(n) \cdot (1 - sigmoid(n)) \tag{4.16}$$

where:

$sigmoid(n) = \frac{1}{1+e^{-n}}$

$$f_i^2 = 1 \tag{4.17}$$

An example of a multiple-layer function approximation network that is fitted to a set of training points as shown in Figure 4.8. The function approximation network for this example has a single input and output. Figure 4.8 shows the flexibility for the neural network to fit non-linear functions.

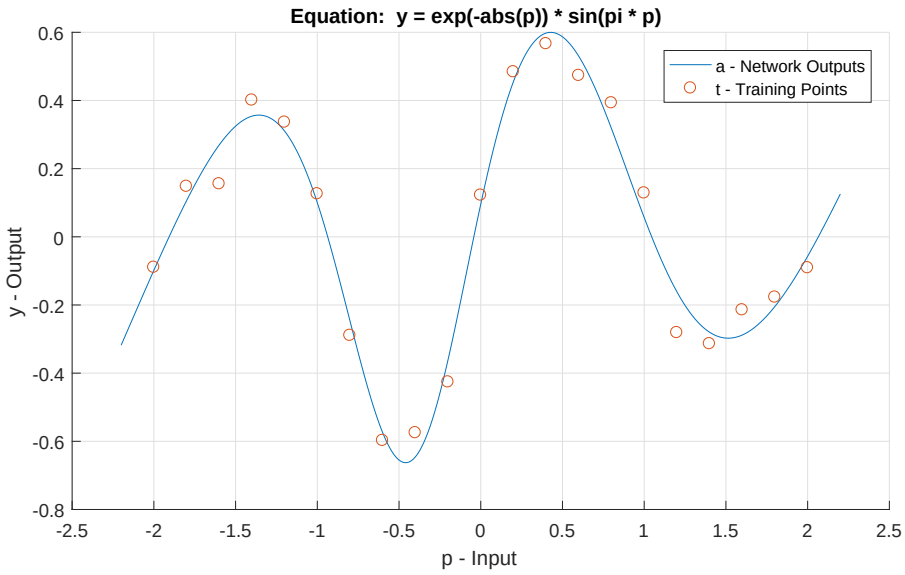

Figure 4.8: Multiple-layer Function Approximation Example

4.3.2 NARX Model Network Architecture

The NARX model has a similar network architecture to the multiple-layer function approximation network presented in the previous section. The input into the NARX model is sequence of past system inputs and outputs, which is represented by a tapped-delay line (TDL) in the network [13, chp.27 pg.5].

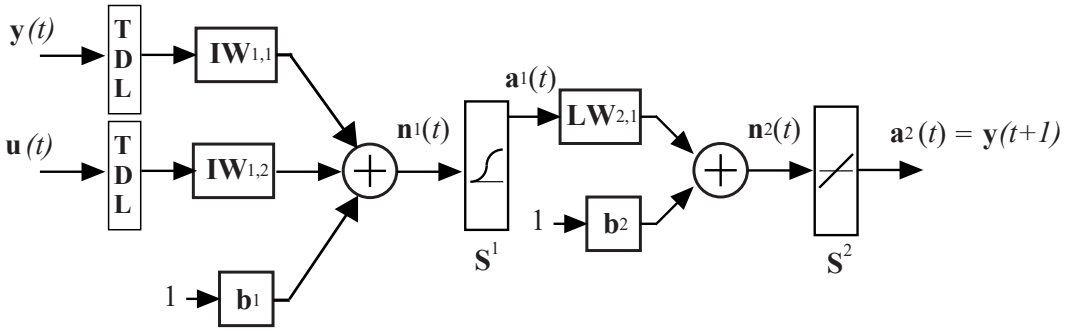

Figure 4.9: Non-Linear Autoregressive Exogenours (NARX) Model

The NARX model in Figure 4.9 is a one-step ahead predictor, which uses the past inputs ($\mathbf{u}(t)$) and outputs ($\mathbf{y}(t)$) to make a prediction of the next output ($\mathbf{y}(t+1)$). For one-step ahead prediction, the NARX model is open-loop with no feedback. To

make multiple-step ahead predictions, the NARX model can be made closed loop by feeding back the output ($\mathbf{y}(t)$) (Figure 4.10). For the application of fault detection, a one-step ahead model is sufficient for creating a healthy reference state, so the network architecture in Figure 4.9 will be implemented. The open-loop NARX model also has the benefit of being trained using the static backpropagaion algorithm (Equation 4.13 and 4.14). For a closed-loop NARX model the network has to be trained using the real-time recurrent learning algorithm, which is more complex than the static backpropagaion algorithm.

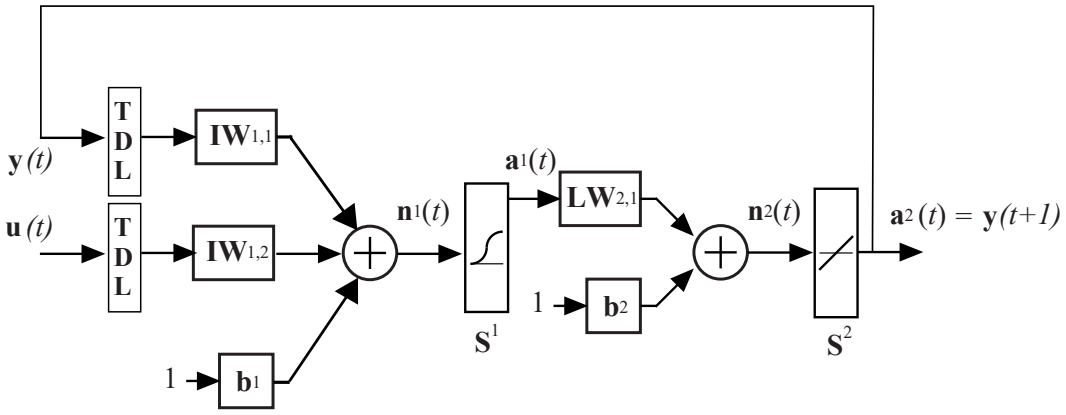

Figure 4.10: Closed Loop Non-Linear Autoregressive Exogenours (NARX) Model

The output ($\mathbf{y}(t+1)$) for the NARX model in Figure 4.9 can be calculated using the forward propagation algorithm in Equation 4.7, where the input ($\mathbf{p}$) into the static neural network is shown by Equation 4.18. The weights ($\mathbf{IW}^{1,1}$ and $\mathbf{IW}^{1,2}$) for the first layer can be combined into a single weight ($\mathbf{W}^1$) matrix shown by Equation 4.19. Using the input ($\mathbf{p}$) and combined weights ($\mathbf{W}^1$) the NARX model in Figure 4.9 can be simplified to the function approximation network in Figure 4.7, where the network can be trained using the backpropagaion and update algorithm shown in Equation 4.13 and 4.14, respectively.

$$\mathbf{p} = \begin{bmatrix} y(t) & y(t-1) & \cdots & y(t-n_y) & u(t) & u(t-1) & \cdots & u(t-n_u) \end{bmatrix}^T \tag{4.18}$$

where:

n_y = Number of Tap-Delays for Output Signal

n_u = Number of Tap-Delays for Input Signal

$$\mathbf{W}^1 = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n_y} & w_{1,1} & w_{1,2} & \cdots & w_{1,n_u} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n_y} & w_{2,1} & w_{2,2} & \cdots & w_{2,n_u} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,n_y} & w_{S,1} & w_{S,2} & \cdots & w_{S,n_u} \end{bmatrix} \tag{4.19}$$

where:

S = Number of Neurons

In this report the NARX model will be trained using the Levenberg-Marquardt algorithm with Bayesian Regularization instead of stochastic gradient descent. The Levenberg-Marquardt algorithm improves the training speed by switching between the Gauss-Newton and Gradient Descent methods depending on whether or not the performance index surface is quadratic. Bayesian Regularization helps to prevent the common issue of over-fitting by using a performance index that combines sum square error with sum squared weights, as shown by Equation 4.20 [13, chp.13 pg.8]. Bayesian Regularization uses β to weight the mean square error, while α weights the sum squared weights and biases ($\mathbf{W}$ and $\mathbf{b}$). By changing the values of α and β the performance index regularization can be changed based on whether or not the network is over-fitting. For Bayesian Regularization, the values of α and β are updated at each iteration of Levenberg-Marquardt. The Levenberg-Marquardt with Bayesian Regularization is discussed in-depth in [13].

$$F(x) = \beta E_D + \alpha E_W = \beta \sum_{q=1}^{Q} (\mathbf{t}_q - \mathbf{a}_q)^T (\mathbf{t}_q - \mathbf{a}_q) + \alpha \sum_{i=1}^{n} x_i^2 \tag{4.20}$$

4.3.3 NARX Model Validation

After training the NARX network, model validation can be performed to determine if the model is over-fitting or under-fitting the actual system. For time series forecasting using NARX models the techniques for model validation are Bayesian Regularization, auto-correlation of the residual errors, and auto-correlation of the inputs against the residual errors. All methods will be demonstrated in this report, with the model validation being used to update the number of neurons in the NARX model. The approach of model validation is iterative with NARX model complexity being adjusted until the best fit of the physical system is achieved.

Bayesian Regularization computes a term called the effective number of parameters - the γ in Equation 4.21 [13, chp.13 pg.16]. If γ is almost equal to the total number of weights and biases in the neural network, then the number of neurons in the network can be increased. After the size of the network is large enough, γ will remain constant as the number of neurons is increased. This allows an iterative approach for measuring the appropriate number of neurons in the network.

$$\gamma = n - 2\alpha^{\mathbf{M}P}\mathbf{tr}(\mathbf{H}^{\mathbf{M}P})^{-1} \tag{4.21}$$

where:

$\mathbf{H}^{\mathbf{M}P}$ = Hessian Matrix

n = Number of Optimization Parameters ($\mathbf{W}$ and $\mathbf{b}$)

Another model validation tool is the auto-correlation function (ACF) of the residual errors, which was used in Chapter 3 for model validation of a linear model. The residual errors should be white noise, which means that the ACF should be an impulse at zero. This indicates that the prediction errors are uncorrelated, meaning the NARX model is able to capture the full dynamics of the system. If there is a correlation in the residual errors then the NARX model complexity needs to be increased by

adding more delays or more neurons. The estimated auto-correlation of the residual errors is shown by Equation 4.22 [13, chp.27 pg.9].

$$R_\epsilon = \frac{1}{N-k} \sum_{t=1}^{N-k} \epsilon(t)\epsilon(t+k) \tag{4.22}$$

where:

ϵ = Residual Errors

The last model validation tool is the cross-correlation of the input against residual errors, which should show no correlation. If there is a correlation between the inputs and residual errors, then the neural network complexity needs to be increased by adding more delays to the inputs or more neurons to the first layer. The cross-correlation of the inputs against the residual errors is shown by Equation 4.23 [13, chp.27 pg.10].

$$R_{u\epsilon} = \frac{1}{N-k} \sum_{t=1}^{N-k} u(t)\epsilon(t+k) \tag{4.23}$$

where:

u = Signal Input

4.4 Data Collection and Sensor Location

The ability to fit a model to a dynamic system is dependent on the measured data. For a NARX model to properly fit a physical system, the measured data need to have minimal noise and capture the full dynamics of the system. This section will discuss the location of the sensors, system operational range, and sampling frequency.

4.4.1 Sensor Location

The sensors are located in the hydrostatic transmission to measure the key performance characteristics of the system, which are flow at the outlet of the servo-valve, angular velocity of the gearbox, and torque on the gearbox. The input into the hydrostatic transmission is the voltage into the servo-valve, which was discussed in Chapter 2. The change in the voltage of the servo-valve will generate a displacement in the spool, which changes the flow and pressure into the hydraulic controlled motor. Depending on the flow and pressure at the inlet of the motor, an angular velocity and torque will be created in the gearbox based on the system load. In this chapter, a healthy reference state for the angular velocity of the gearbox will be modeled using a NARX model to demonstrate the process for fitting a non-linear model to a dynamic system.

4.4.2 Operational Range

For linear modeling, the operational range for the input voltage into the servo-valve was divided into linear ranges as shown by Table 3.1. The advantage of using a NARX model is that the full operational range of the input voltage servo-valve can be captured using a single model, since a NARX model is able to fit the non-linear parts of the dynamic system. The full operational range for the hydrostatic transmission is shown by Table 4.1 below. The NARX model will be trained with a data set of 10,000 points.

Table 4.1: Servo-Valve Input Ranges

Operational Range	Minimum Voltage	Maximum Voltage	Sample Size	Data Sets
1	0	4	10000	1

4.4.3 Sampling Frequency

To properly estimate the NARX model to fit the dynamics of the system the key performance characteristics need to be captured. Based on the Nyquist criterion, to capture the dynamics of the system, the sampling frequency needs to be more then twice the highest frequency in the system dynamics. The system dynamics range can be determined from the frequency response shown in Figure 3.1. The input to the system is generated by running white noise through a low pass filter shown by Equation 3.21, which reduces the extreme changes in the input into the servo-valve, as discussed in Chapter 3. Based on Figure 3.1, a sampling frequency of 200Hz was sufficient to capture the dynamics of the hydrostatic transmission. The data set in Table 4.1 was captured with a sampling rate of 200Hz, with the input being passed through the low pass filter, as discussed above.

4.5 Model Training and Validation

The NARX model will be trained on the operational range shown in Table 4.1 for the input voltage into the servo-valve. The process of fitting the NARX model consists of parameter estimation and model validation. This section will go through each of the processes discussed above by fitting a non-linear model to the healthy reference state for the hydrostatic transmission, with angular velocity as the output.

4.5.1 Parameter Estimation

The NARX model shown in Figure 4.9 will be trained using the architecture shown in Table 4.2. This model architecture is determined based on the linear models discussed in Chapter 3, which had a dynamic system order of three with a time delay of two samples. This indicates that the TDL for the input and output should be at least five. The NARX modeled is trained using backpropagation and the optimization algorithm of Levenberg-Marquardt with Bayesian Regularization using MATLAB's

neural network toolbox [14].

Table 4.2: NARX Model Architecture

Input Tap-Delay Size	Output Tap-Delay Size	Number of Neurons in First Layer
5	5	10

The NARX model from MATLAB for the network architecture in Table 4.2 is shown in Figure 4.11.

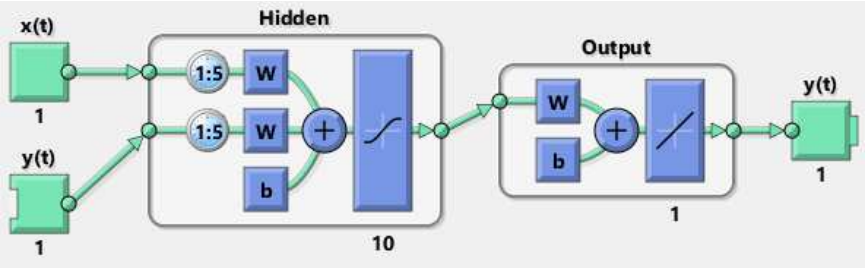

Figure 4.11: NARX Model Architecture

4.5.2 Model Validation

The tools that will be used in this section for model validation are Bayesian Regularization, auto-correlation of the residual errors, and cross-correlation of the inputs against the residual errors. The auto-correlation of the residual errors and cross-correlation of the inputs against residual errors will be used to determine the TDL required for the model to capture the entire dynamics of the system. This will be an iterative approach, where the TDL size will be continually increased until the auto-correlation for the residual errors and the cross-correlation of the inputs against the residual errors are within a confidence limit of zero. After the TDL size is estimated, the number of neurons in the first layer will be estimated using Bayesian Regularization's effective number of parameters.

For the NARX model in Figure 4.11, which has the network architecture shown in Table 4.2, the auto-correlation of the residual errors for the trained model is shown in Figure 4.12 and the cross-correlation of the inputs against the residual errors is shown in Figure 4.13. These figures indicate that the TDL needs to be enlarged, since the magnitude of the correlations are not within the confidence limits around zero.

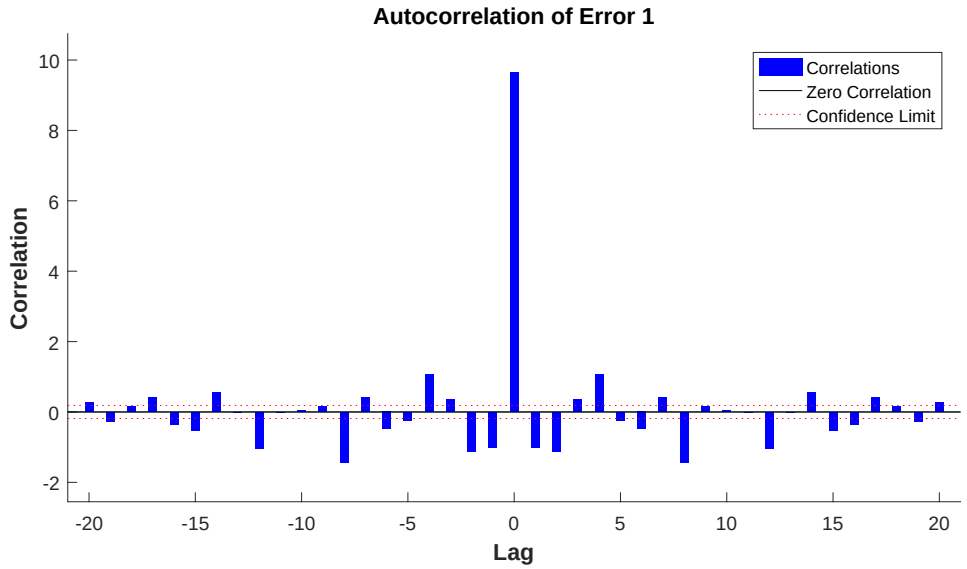

Figure 4.12: Auto-correlation of Residual Errors

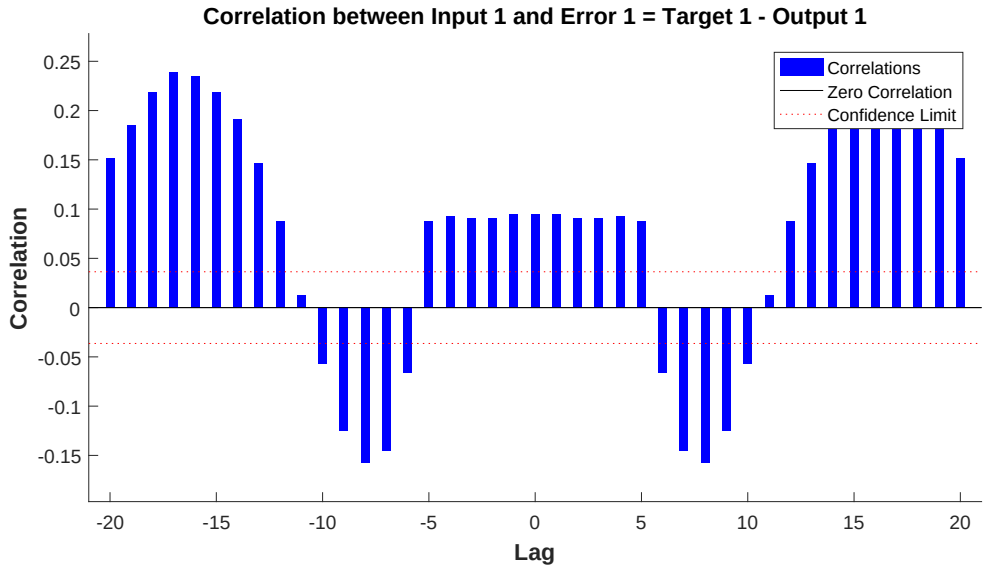

Figure 4.13: Auto-correlation of Inputs Against Residual Errors

After increasing the TDL sizes in the model to 15, as shown by network architecture in Table 4.3, the correlations are within the confidence limits as shown in Figure 4.14 and 4.15.

Table 4.3: NARX Model Architecture

Input TDL Size	Output TDL Size	Number of Neurons in First Layer
15	15	10

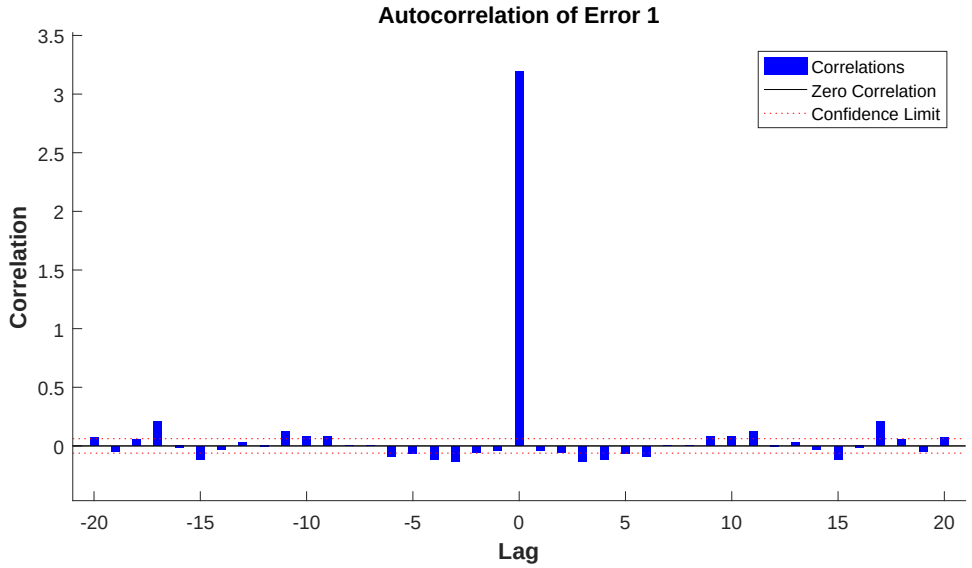

Figure 4.14: Auto-correlation of Residual Errors

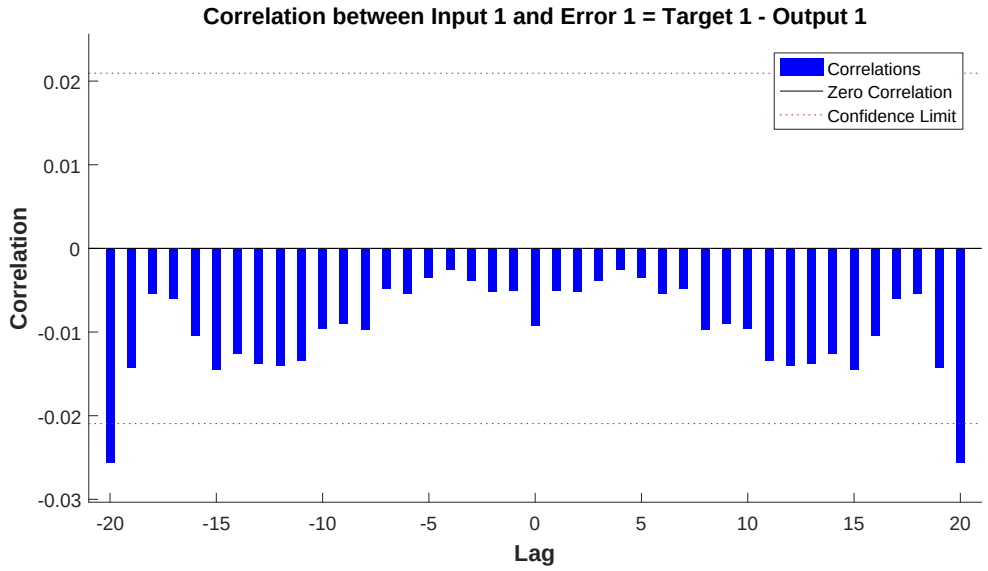

Figure 4.15: Cross-correlation of Inputs Against Residual Errors

With the proper TDL sizes for the input and output signal determined (Table 4.3), Bayesian Regularization can be implemented to determine the effective number of parameters. For the NARX architecture in Figure 4.3 the effective number of parameters using Bayesian Regularization is shown in Table 4.4. The effective number of parameters in the network is close to the total number of parameters, so the number of neurons in the first layer can be increased.

Table 4.4: NARX Model Effective Number of Parameters

Total Number of Parameters	Effective Number of Parameters
321	301

The number of neurons in the first layer of the NARX model can be increased to 35, as shown by the network architecture in Table 4.5. After training the network (Table 4.5) using Levenberg-Marquardt with Bayesian Regularization, the effective number of parameters is shown in Table 4.6.

Table 4.5: NARX Model Architecture

Input TDL Size	Output TDL Size	Number of Neurons in First Layer
15	15	35

Table 4.6: NARX Model Effective Number of Parameters

Total Number of Parameters	Effective Number of Parameters
1,120	934

From Table 4.6 the effective number of parameters is close to the total number of parameters, but based on the performance of the mean square error of the test data set, shown in Figure 4.16, there was little decrease in mean square error after initial training of the network. The network architecture in Table 4.5 with 35 neurons in the first layer is sufficient for capturing the dynamics of the system.

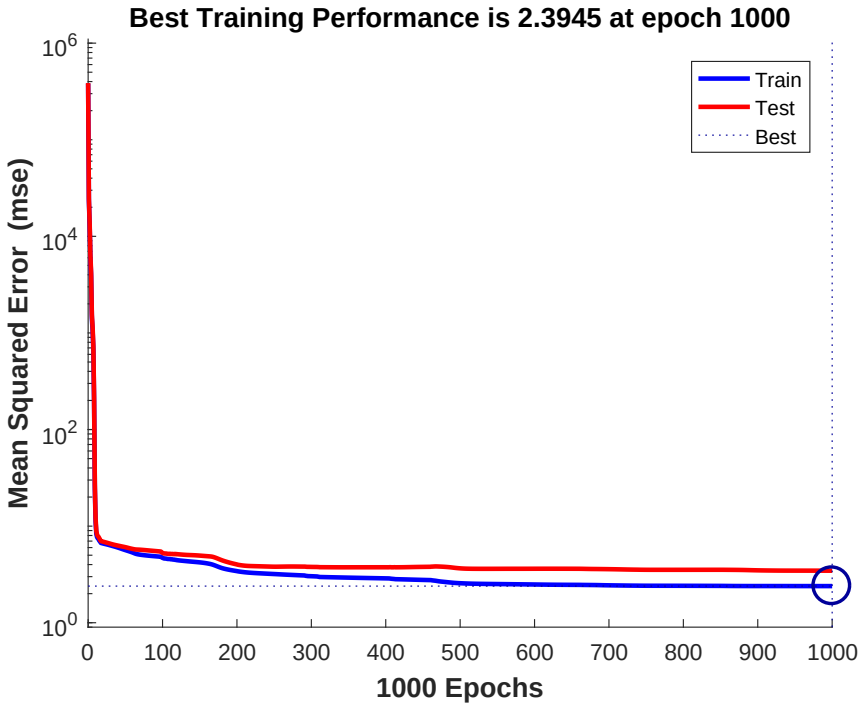

Figure 4.16: Mean Square Error - Network Performance

4.6 Implement NARX Model

After performing parameter estimation and model validation, the optimal NARX model for the hydrostatic transmission's hydraulic motor angular velocity is shown in Table 4.5. The trained NARX model can be used as a one-step ahead estimator, which can be compared with the measured output of angular velocity to find the residual error (Equation 4.24). The magnitude of the residual error indicates the level of degradation in the system. In following sections, each key performance characteristic (flow at the hydraulic motor, angular velocity at the gearbox, and torque at the gearbox) will be modeled using the approaches demonstrated in this chapter. Figure 4.17 shows the one-step ahead prediction based on the NARX model, with the network architecture in Table 4.5. The NARX model is able to accurately predict the next step in the hydrostatic transmission for a healthy system.

$$\epsilon(t) = y(t) - \widehat{y}(t|\theta) \tag{4.24}$$

where:

$\widehat{y}(t|\theta)$ = Estimated One Step Ahead Prediction

$y(t)$ = Measured Output

$\epsilon(t)$ = Residual Error

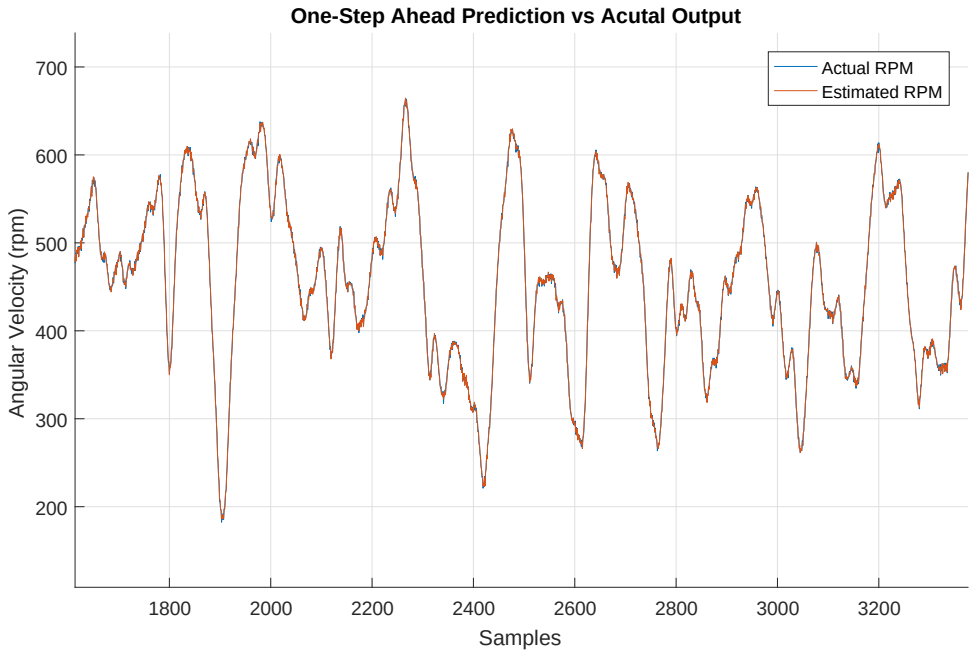

Figure 4.17: One-Step Ahead Prediction

4.7 Comparing Performance of Linear and Nonlinear Models

After showing the implementation of a NARX model, we can compare it against the linear Box-Jenkins model presented in Chapter 3. If we train the Box-Jenkins over the entire operational range shown in Table 4.1 the model's performance can be shown in Figure 4.18. The NARX model's performance is shown in Figure 4.19. It can be seen from the figures, that the NARX model is able to more accurately fit the physical system for the entire operational range. This is due to there being non-linearities in

the system, which the linear Box-Jenkins is unable to capture. For fault detection the inability to accurately fit the system over the entire operational range will result in misclassification of fault states. For the remainder of this thesis the NARX model will be used for fault detection for the reason discussed above.

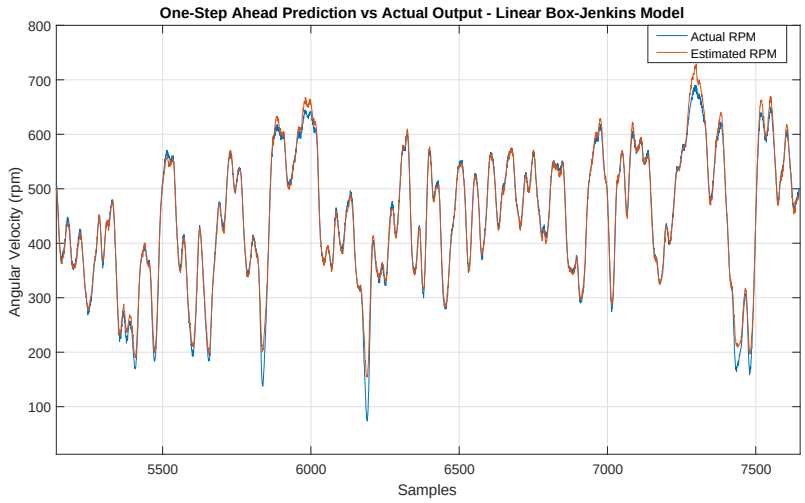

Figure 4.18: One-Step Ahead Prediction for Linear Box-Jenkins Model

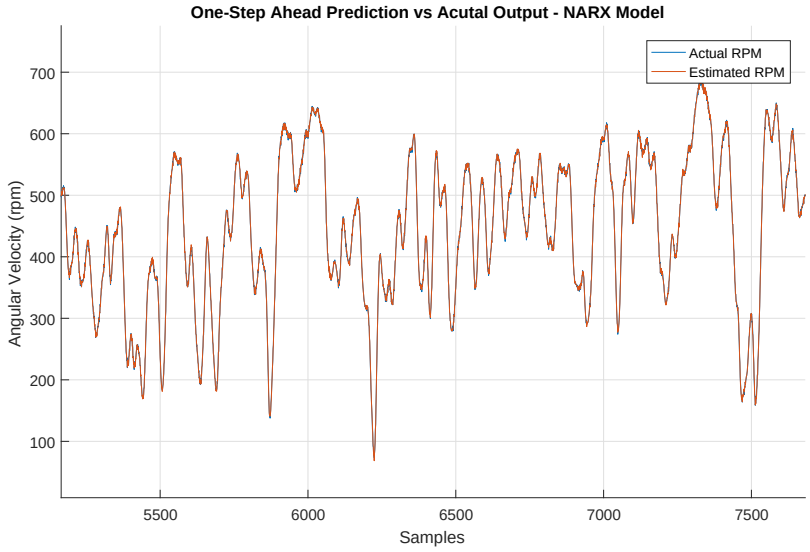

Figure 4.19: One-Step Ahead Prediction for NARX Model

CHAPTER 5

HEALTH MONITORING

5.1 Introduction

In the previous chapters, dynamic modeling for a healthy hydrostatic transmission was demonstrated. After creating a healthy model of the system, we can examine how the prediction errors of the model change due to degradation of a sub-component in the system. A tuned dynamic model will have a small prediction error when the system is operating in a healthy state, but as the system is operated over time, parts will degrade and the dynamics of the physical system will change. This leads to the predication errors changing with the degradation of the system. In this chapter we will examine this idea for predicting a single fault using a dynamic model of a single sensor. Later in the chapter, multiple faults will be predicted using dynamic models of multiple sensors in the hydrostatic transmission.

Before discussing a fault in the hydrostatic transmission, we need to re-introduce the dynamic models presented in Chapter 3 and 4. The dynamic models are the reference states for a healthy system, so remembering the ideas presented will give the needed background to discuss fault detection using these dynamic models. In Chapter 3 and 4 the hydrostatic tranmsission healthy reference state was modeled using a linear Box-Jenkins mdoel and a NARX model. A Box-Jenkins model followed a traditional approach of preliminary identification, parameter estimation, and model validation, while the NARX model is a black box technique where there is no preliminary identification. Instead the model complexity is tuned using parameter estimation and model validation. There are benefits to both approaches, which was

discussed in-depth in the past chapters. After system identification, the tuned dynamic models can be implemented using Equations 3.26 and 4.7 to make a prediction of output $\widehat{y}(t|\theta)$ based on past inputs and output. The prediction $\widehat{y}(t|\theta)$ can be compared against a measured output $y(t)$ from key performance sensors (motor flow rate, gearbox angular velocity, and gearbox torque) in the system to measure the level of degradation from the healthy reference model based on the prediction error $\epsilon(t)$ from Equation 5.1. For the classification of faults in the system, the NARX model will be implemented instead of a linear Box-Jenkins model, due to the benefits discussed in Chapter 4.

$$\epsilon(t) = y(t) - \widehat{y}(t|\theta) \tag{5.1}$$

where:

$\widehat{y}(t|\theta)$ = Estimated One Step Ahead Prediction

$y(t)$ = Measured Output

$\epsilon(t)$ = Prediction Error

The prediction error is the key index for measuring a change in the dynamic system. The level of change in the prediction error will indicate the severity of the degradation. That was why Chapters 3 and 4 had an in-depth discuss of training and validating of dynamic models, since minimizing the predication error on the healthy model will prevent misclassifying a fault. In the next section, the prediction errors using a NARX model will be used to detect a single fault in a system.

5.2 Single Fault Prediction

We will first examine a single fault using a single dynamic model of a sensor and then we expand on the idea for multiple faults. For a single fault, only one dynamic model of a sensor is needed to determine if the system is healthy or in an alert state

(30% loss in system nominal angular velocity). When a system is in an alert state, the hydrostatic transmission should be stopped and examined.

The technique for determining if the system is in a healthy or alert state is to compare an inline measurement of the prediction error distribution with known prediction error distributions for both healthy and alert states. This is shown in Figure 5.1, where the inputs into the system are the voltage into the servo-valve and the gearbox angular velocity. The servo-valve input voltage is fed into the dynamic model that predicts $\widehat{y}(t|\theta)$ for the gearbox angular velocity, which is subtracted from the current measured gearbox angular velocity $y(t)$. This produces a prediction error based on Equation 5.1. This is done over a short time interval, and then the measured prediction error distribution is compared against a known prediction error distribution for both healthy and alert states. The state with the most similar prediction error distribution is chosen as the current state of the system.

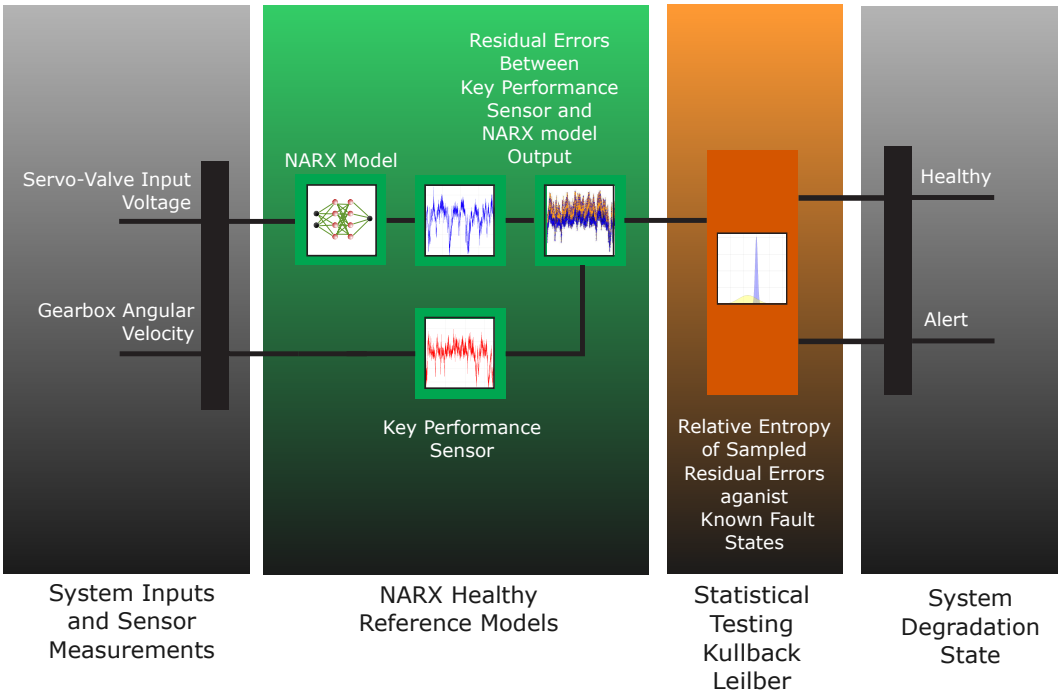

Figure 5.1: Single Load Fault Detection Diagram

The approach taken for statistical testing is to collect inline prediction errors

for an interval of time to form a inline prediction error distribution, which will be statistically compared against the known health state distributions of the system. This section will discuss the statistical approach for comparing distributions, but before discussing statistical testing, the location of the sensors and faults in the system will be presented.

5.2.1 Sensor and Fault Location

For this section the fault being predicted in the hydrostatic transmission is a load increase, which is called a load alert. An increase in the load puts additional wear on the gearbox, due to the increased torque on the gear teeth of the gearbox. The key performance sensor that will be used to measure a load fault in the system is the gearbox angular velocity. The location of a load fault and key performance sensor for gearbox angular velocity is shown in Figure 5.2.

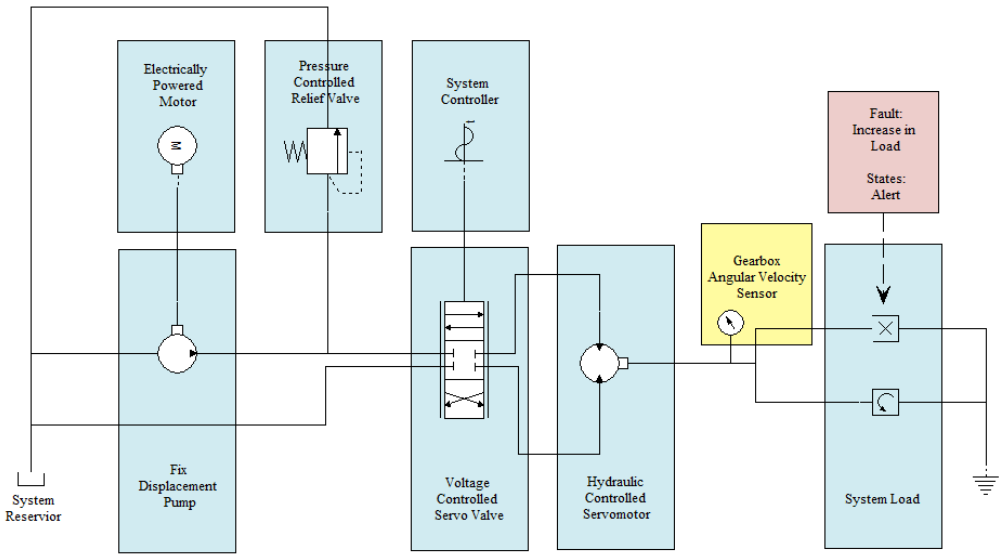

Figure 5.2: Sensor and Fault Location

The dynamic NARX model shown in Chapter 4 used a sampling rate of 200Hz,

which was sufficient to capture the dynamics of the hydrostatic transmission. For fault detection, the prediction errors from the dynamic model are collected over 5 seconds at 200Hz for a total of 1,000 points. A 5 second interval was selected to insure that when a load fault occurred the system can be stopped before there is any long term damage to the system.

5.2.2 Statistical Test

The Kullback-Leibler test measures the relative entropy between two probability distributions using Equation 5.2 [15]. A relative entropy near zero indicates that the probability distributions are very similar, while a higher relative entropy indicates that the distributions are different.

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) log \frac{p(x)}{q(x)} dx \tag{5.2}$$

where:

$p(x)$ = Health State Probability Distribution

$q(x)$ = Sampled Prediction Errors Probability Distribution

For the this thesis, the Kullback-Leibler for two Gaussian probability distributions is used, which is shown by Equation 5.3.

$$D_{KL}(P||Q) = log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \tag{5.3}$$

where:

σ_1 = Health State Probability Distribution Standard Deviation

μ_1 = Health State Probability Distribution Mean

σ_2 = Sampled Prediction Errors Probability Distribution Standard Deviation

μ_2 = Sampled Prediction Errors Probability Distribution Mean

The relative entropy is an important statistical test that measures similarity of two distributions. This will be used in the next section for statistically testing if the inline measured prediction errors is correlated to a known health state distribution.

5.2.3 Implementation of Statistical Test

To predict the health state (healthy or alert) for the dynamic system, the distribution of the inline measured prediction errors will be compared with the known health state distribution. Example prediction error distributions for healthy and alert states are shown in Figure 5.3. The raw distributions are approximated by the Gaussian distribution which is a good fit for the data and allows the continuous Kullback-Leilber to be used, which simplifies the relative entropy calculation and allows for real-time implementation. Note that the healthy state distribution is for errors obtained when using the trained healthy model to predict with data collected from the system in a healthy condition. The alert state distribution is for errors obtained when using the trained healthy model to predict with data obtained when the system had a failure.

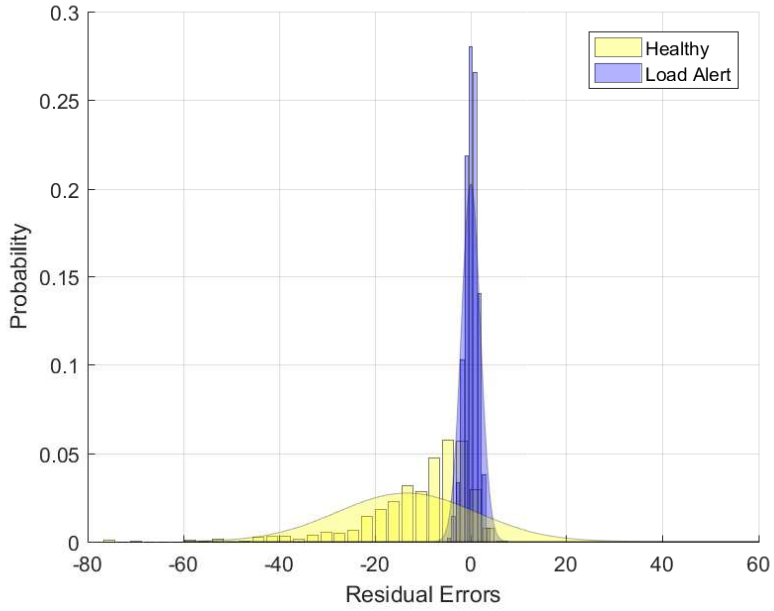

Figure 5.3: Known Health States for Gearbox Angular Velocity

An example inline measured prediction error distribution for the dynamic system operating in a load alert is shown in Figure 5.4. Using the inline prediction error distribution, the relative entropy can be calculated for the inline prediction error distribution against each known health state distribution. The results for the relative entropy calculation are shown in Table 5.1. Based on table, the current health state of the system is load alert, since it has the lowest relative entropy, indicating that inline prediction error distribution is closest to the load alert distribution, which is clear from Figure 5.4. For real-time implementation a threshold for the relative entropy would be decided on based on maintenance cost and risk of not replacing. Selecting a threshold is simple for a single fault, but with multiple faults the threshold becomes more abstract and a rule system needs to be designed for making this choice.

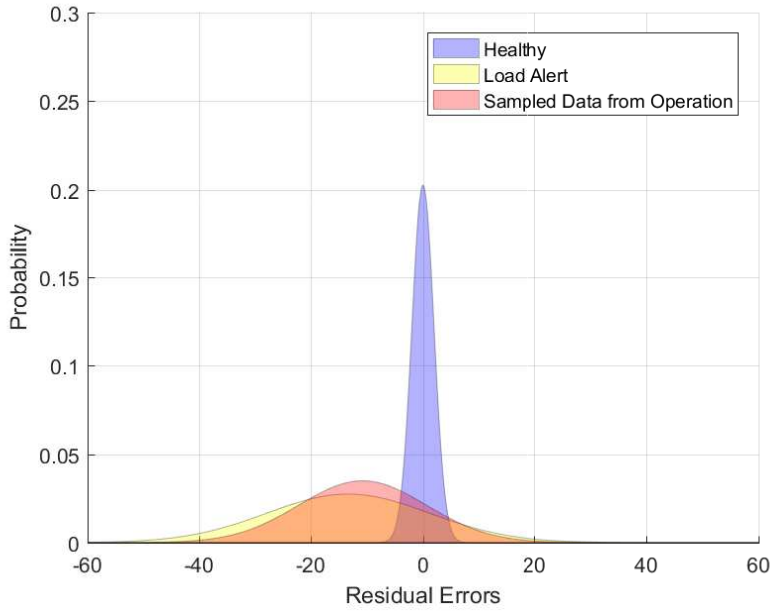

Figure 5.4: Sampled Prediction Errors vs Known Health States for Gearbox Angular Velocity

Table 5.1: Health States Relative Entropy vs Sampled Prediction Errors

Health State	Healthy	Load Alert
Relative Entropy	29.6664	0.0660

5.3 Multiple Fault Prediction

In the previous section the fault detection was presented for a single fault in the hydrostatic transmission. In this section, that will be expanded to multiple health states for a load fault. The multiple health states are healthy, load warning (15% loss in system nominal angular velocity), and load alert (30% loss in system nominal angular velocity). When a system is in a warning state, the system can still be operated, but should be examined by maintenance. If the system is in an alert state, the system should be stopped and examined. The diagram for implementing fault detection for multiple load faults is shown in Figure 5.5. This is similar to the previous section for a single load fault.

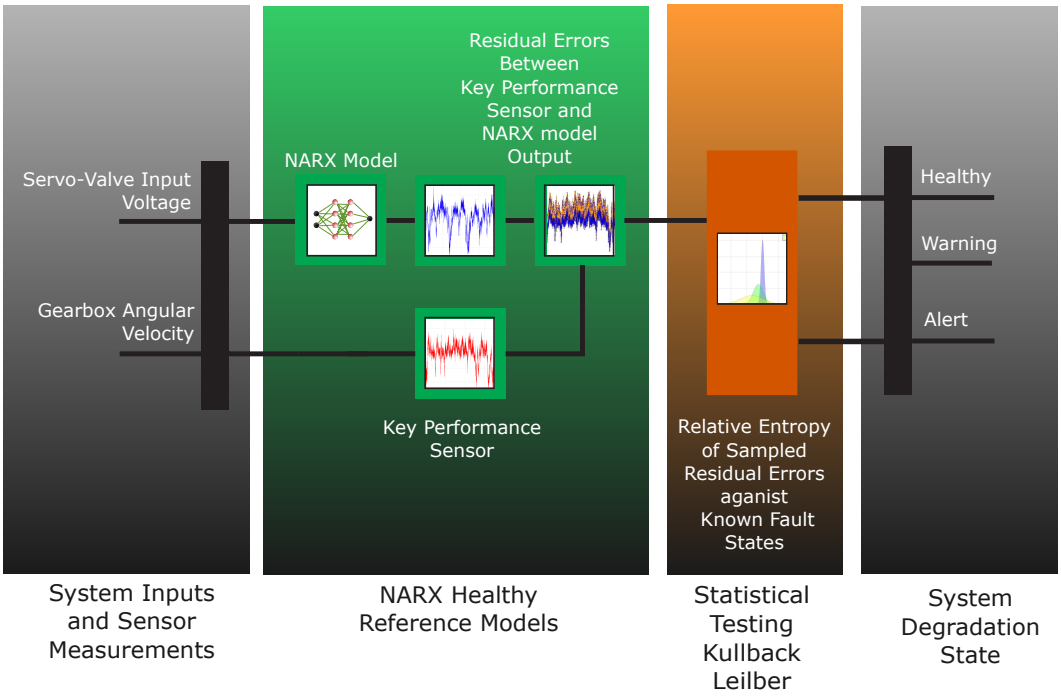

Figure 5.5: Multiple Load Faults Detection Diagram

This section will discuss sensor and fault location, and then will present the implementation of fault detection for multiple faults using a single key performance sensor.

5.3.1 Sensor and Fault Locations

This section will predict an increased load fault for multiple health states (healthy, load warning, and load alert), using the key performance sensor for angular velocity in the gearbox. The locations of the load faults and key performance sensor are shown in Figure 5.6.

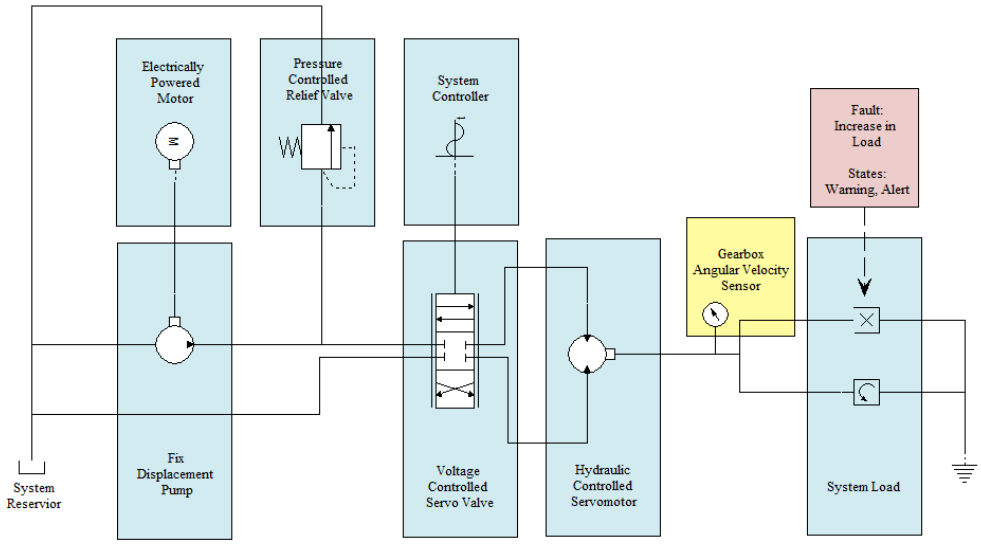

Figure 5.6: Sensor and Fault Location

The dynamic model sampling rate and time interval will be the same as presented in the last section, which is 200Hz and 5 seconds for 1,000 points. These 1,000 points will be used to classify the health state of the hydrostatic transmission in real-time.

5.3.2 Implementing Statistical Test

To predict the health state (healthy, load warning, and load alert) for the dynamic system, the relative entropy of the inline prediction error distribution will be compared with each of the health state distributions. The known health state prediction error distributions are shown in Figure 5.7. The distributions are fitted with a Gaussian as discussed in the previous section.

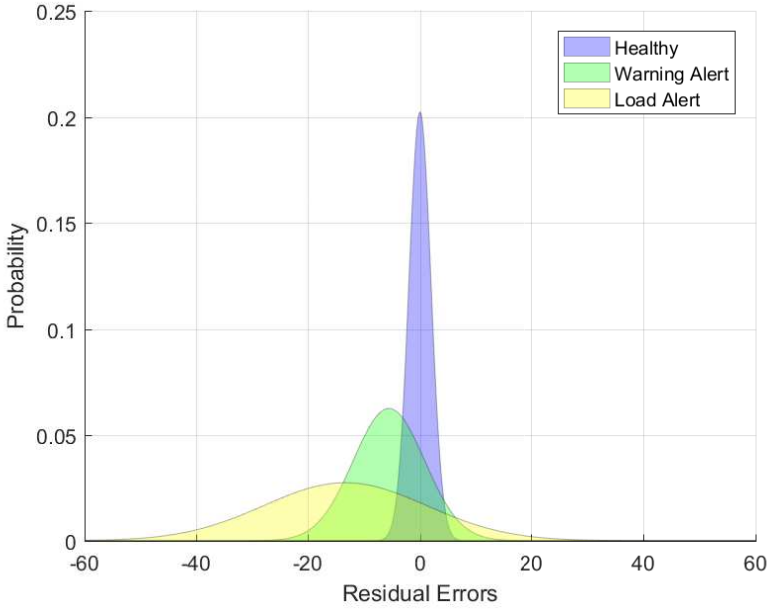

Figure 5.7: Known Health States for Gearbox Angular Velocity

The inline prediction error distribution for the system operating in a load alert is shown in Figure 5.8. The relative entropy for the inline prediction error distribution against the known fault state distributions is shown in Table 5.2. Based on table, the system is currently operating in a load alert and the system should be stopped and examined. It can also be seen from Figure 5.8 that he sampled distribution best matches the load alert distribution.

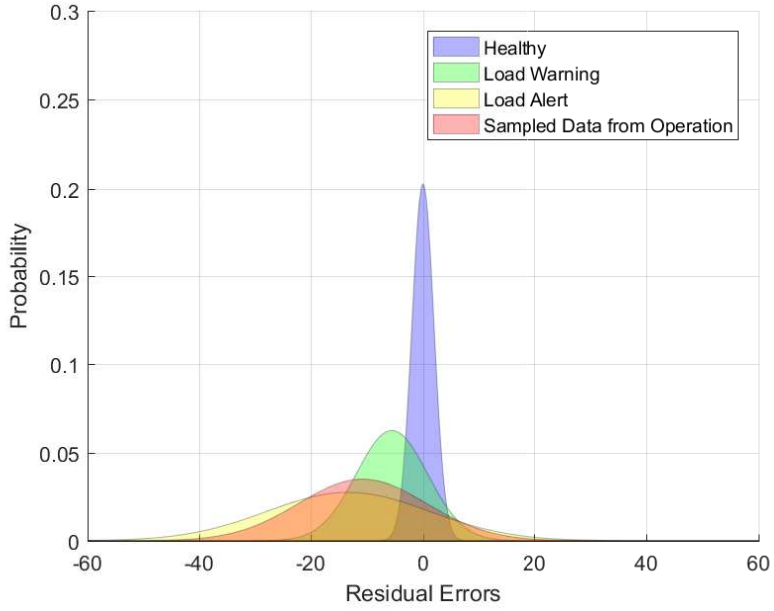

Figure 5.8: Sampled Prediction Errors vs Known Health States for Gearbox Angular Velocity

Table 5.2: Health States Relative Entropy vs Sampled Prediction Errors

Health State	Healthy	Load Warning	Load Alert
Relative Entropy	29.6664	0.8590	0.0660

This section showed that the approach presented for statistical testing, using a single key performance sensor, can be easily expanded from single fault detection to multiple fault detection. In the next section the statistical testing approach will be expanded to multiple fault prediction using multiple key performance sensors. The benefit of using multiple key performance sensors is that the different sensors in the system will pick-up separate dynamic information about the hydrostatic transmission. This will be important when predicting the health state of the system for different kinds of faults.

5.4 Multiple Faults and Multiple Sensors

In the previous section a statistical method was presented for classifying the health state of a hydrostatic transmission using a single key performance sensor. This section will expand on the techniques in the previous section, but will introduce an approach for using multiple key performance sensors to determine the health state of the system. For each key performance sensor there is a prediction error distribution, which can be compared with known health states for each sensor. This is shown in Figure 5.9, where the input into the system is voltage into the servo-valve and the sensor measurements of the gearbox angular velocity, gearbox torque, and motor flow rate. The input into the servo-valve is fed into a dynamic model that predicts $\widehat{y}(t|\theta)$ for each of the sensors, which is subtracted from the measured sensors $y(t)$. Then the prediction errors are sampled for an interval of time to produce a distribution, which is compared by statistical testing with known health state distributions for each sensor. The statistical test is Kullback-Leilber, which was discussed in the previous sections. With there being multiple sensors, there is a relative entropy calculation for each sensor, with the relative entropy being the difference between inline prediction error distributions and known health state distributions for each sensor. A classification neural network will be used to combine the relative entropies among multiple sensors to make a prediction of the system's actual health state.

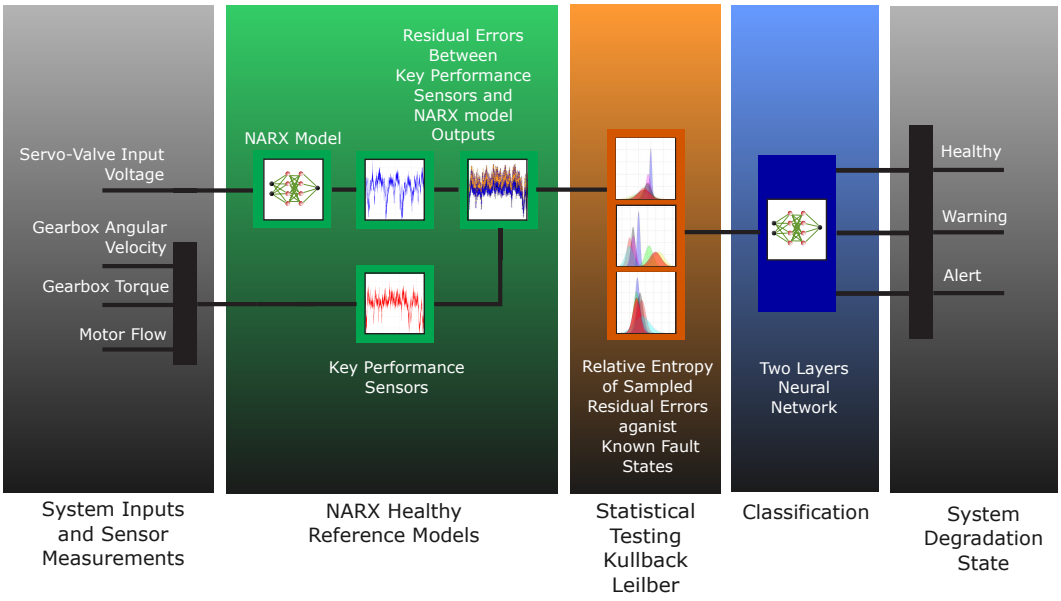

Figure 5.9: Multiple Load Faults Detection using Multiple Sensors Diagram

This section will begin by showing the location of the key performance sensors and load faults. Then statistical testing will be demonstrated for multiple faults and sensors, followed by training and validation of a classification neural network for combining the information from multiple sensors. This section will demonstrate fault prediction for a increased load failure, which has health states of healthy, load warning, and load alert.

5.4.1 Sensors and Faults Location

The key performance sensor locations are shown in Figure 5.10. For this section we are examining a load fault, the location of a load fault in hydrostatic transmission is shown in Figure 5.10 and was further discussed in Chapter 2. A load fault was categorized as a percentage of reduction from the nominal gearbox angular velocity when operating with a 2 volt input into the servo-valve. For a load warning and alert, the reduction in angular velocity is 15% and 30% from nominal operation.

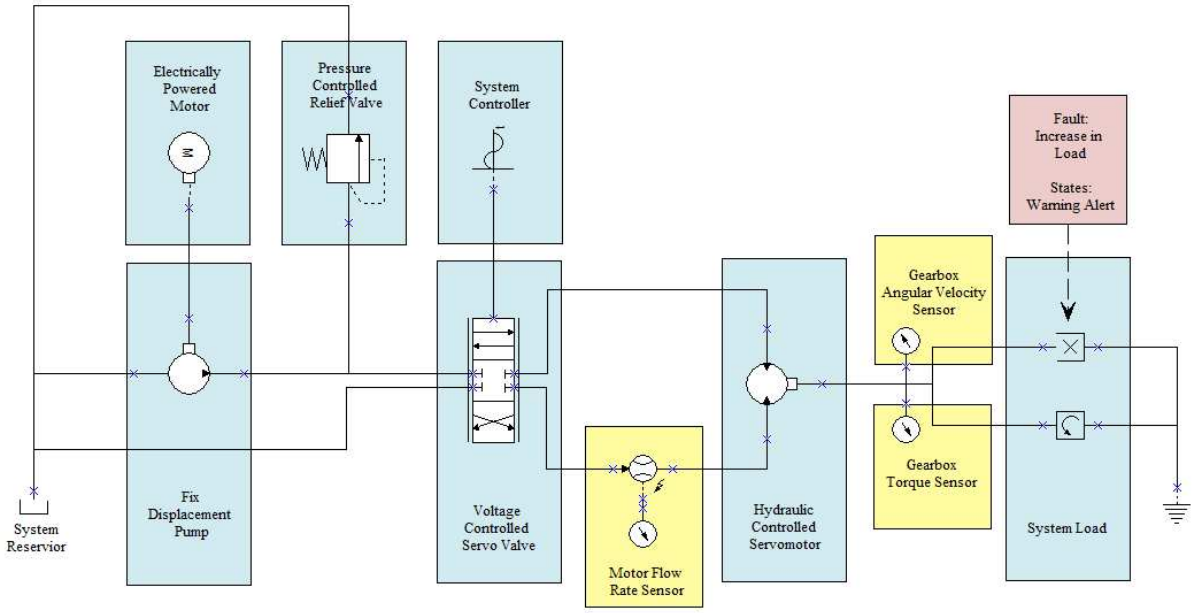

Figure 5.10: Sensors and Faults Location

5.4.2 Implementation of Statistical Test

To demonstrate the Kullback-Leibler for fault detection, distributions for warning and alert for a load failure will be used to predict a load fault from inline measured prediction errors. The health states of healthy, load warning, and load alert for each key performance is shown in Figures 6.1, 6.2, and 6.3. The fault states are based on the prediction errors of a dynamic NARX model for each key performance sensor.

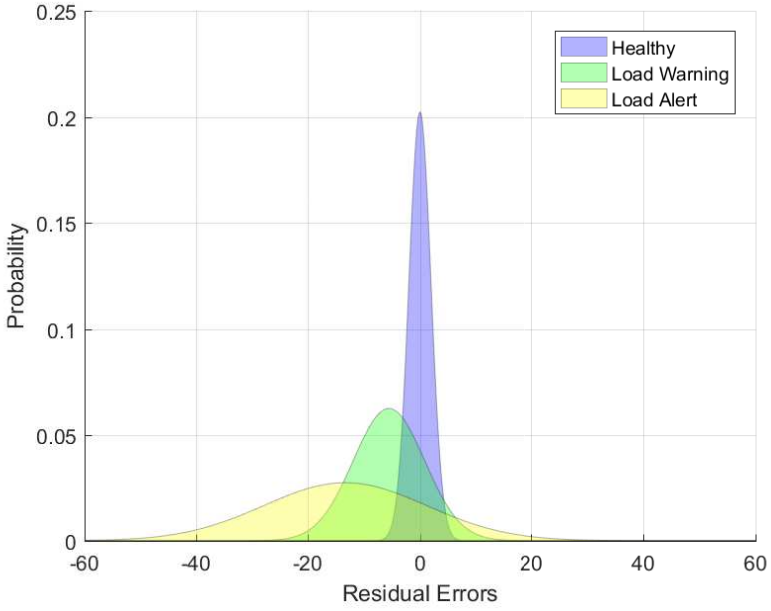

Figure 5.11: System Load Fault PDFs for Gearbox Angular Velocity

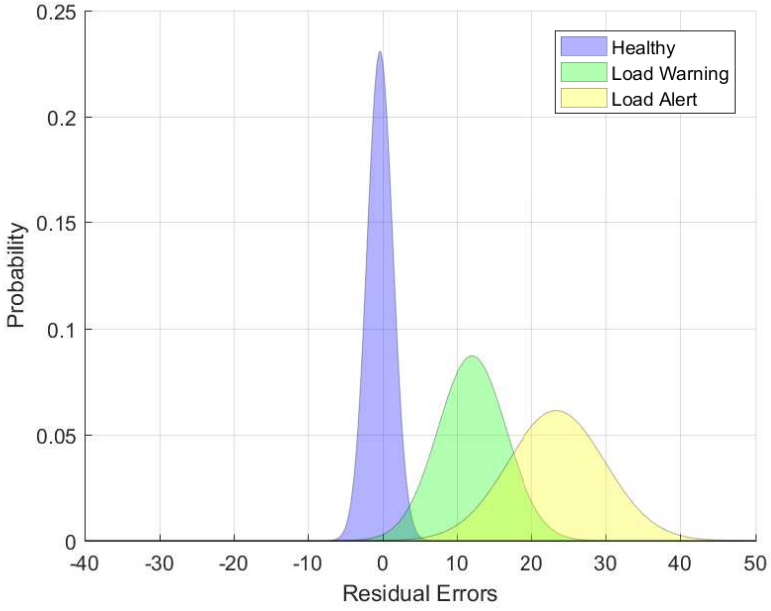

Figure 5.12: System Load Fault PDFs for Gearbox Torque

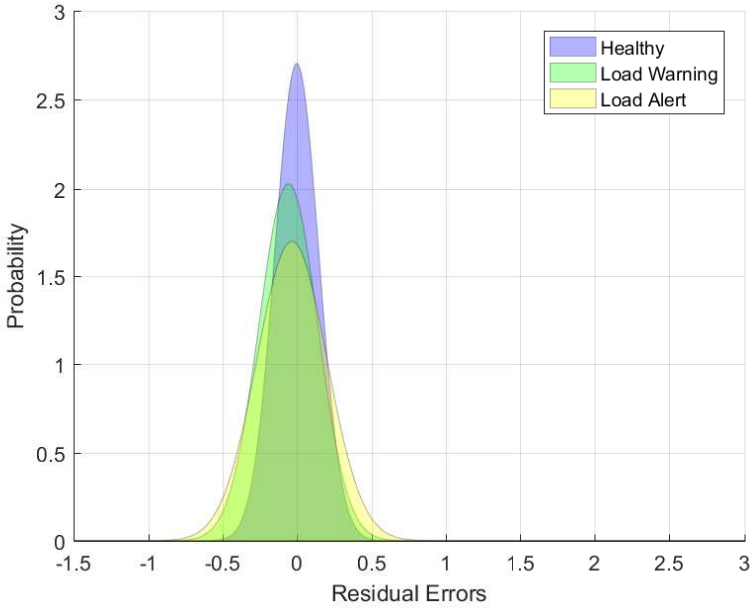

Figure 5.13: System Load Fault PDFs for Motor Flow Rate

To classify a fault state, a set of points are collected over 5 second intervals at a sampling rate of 200 Hz. An example of the prediction error distributions for 1,000 points sampled inline is shown in Figure 5.14, 5.15, and 5.16. The prediction error distributions indicate that the system is in a state of alert for a load failure.

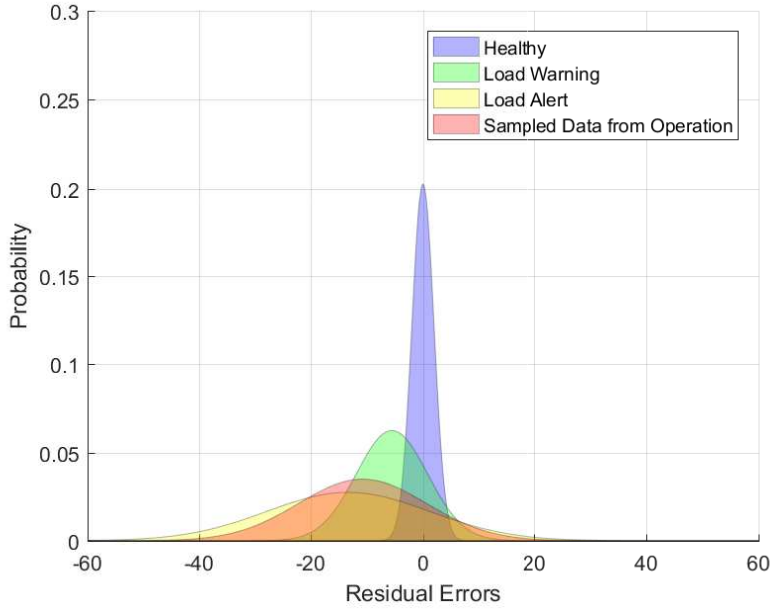

Figure 5.14: System Load Fault PDFs for Gearbox Angular Velocity

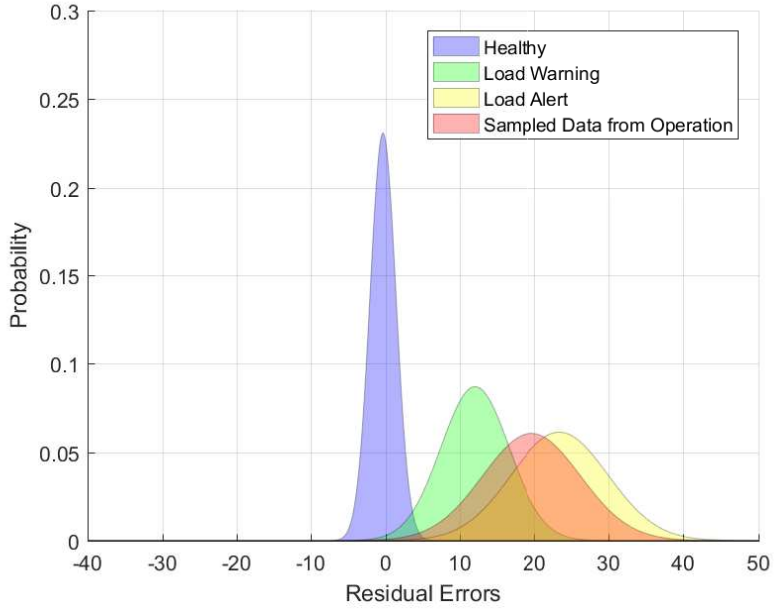

Figure 5.15: System Load Fault PDFs for Gearbox Torque

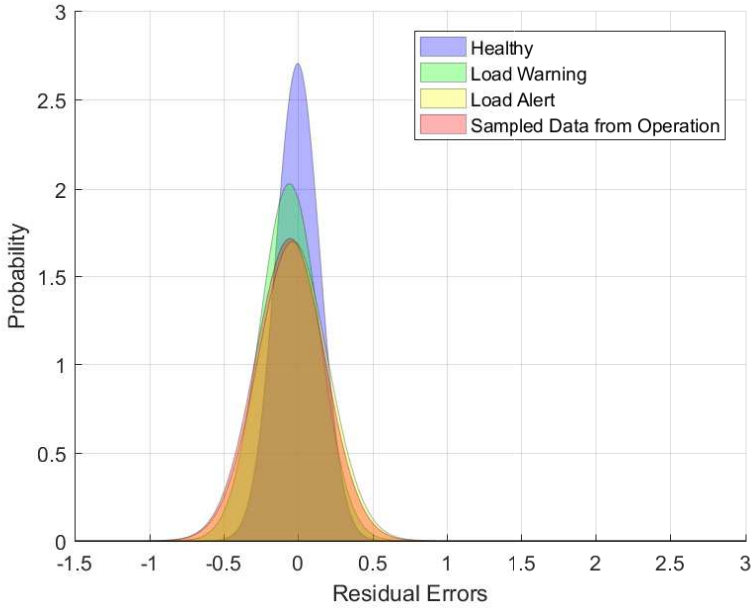

Figure 5.16: System Load Fault PDFs for Motor Flow Rate

The continuous Gaussian Kullback-Leibler Equation 5.3 can be used for the distributions in Figure 5.14, 5.15, and 5.16 to calculate their relative entropy between the inline prediction error distribution and known fault state distributions. The relative entropy of the distributions are shown in Table 5.3. It can be seen from the table, that the system current health state is load alert, since it has the lowest relative entropy for all key performance sensors.

Table 5.3: Load Fault States Relative Entropy vs Sampled Operational Data

Sensors	Healthy	Load Warning	Load Alert
Gearbox Angular Velocity	29.6664	0.8590	0.0660
Gearbox Torque	72.2791	1.5489	0.1679
Motor Flow Rate	0.3507	0.0324	0.0032

From the results in table, each sensor predicts the health state. A rule system needs to be developed for choosing between the relative entropy of multiple sensors

when the sensors do not all agree. A simple approach would be to use a committee where each sensor gets one vote and the health state with the most votes is selected as the most probable health state. For Table 5.3, the selected health state is load alert, since load alert has the smallest relative entropy for each sensor. There is an issue with this approach, since each sensor weights its relative entropy the same, but in reality each sensor should be weighted differently based on ability to observe a fault. By examining Figures 6.1, 6.2, and 6.3, the gearbox angular velocity and gearbox torque health state sensors' distributions are spread out, but the flow rate sensor distributions are on top of each other. This indicates that the flow rate sensor is not as useful in showing a load fault and should be weighted less than the other sensors. To overcome this issue, a classification neural network will be implemented in the next section to develop rules for combining sensors to make a health state prediction.

5.4.3 Classification Neural Network

In the section above, statistical testing was presented for measuring relative entropy between known health states and inline measured prediction errors. With there being multiple key performance sensors, each observing the same known health states, rules need to be developed for weighting the information between the sensors to determine the actual health state of the system. Each sensor measures a different property of the dynamic system, so the rules will factor in the sensors sensitivity to specific sub-component degradation. The rules can be generated by expert knowledge using fuzzy logic or degradation data for weighting a neural network. In this paper the rules are designed by measuring degradation states for training a classification neural network. This section will discuss the architecture of a classification neural network along with the algorithm for training the network. After presenting the training algorithm for the neural network, model validation will be discussed for preventing over-fitting of the classification neural network. The final part of this section will demonstrate the

training of a classification neural network for categorizing a load fault using multiple sensors.

5.4.4 Classification Neural Network Architecture

The classification neural network implemented for fault detection is shown in Figure 5.17. The first layer uses a log-sigmoid transfer function and the second layer (output of the neural network) uses a softmax. The softmax outputs can be viewed as the probability that the system is in a given state. The purpose of a classification neural network is to map inputs to categories.

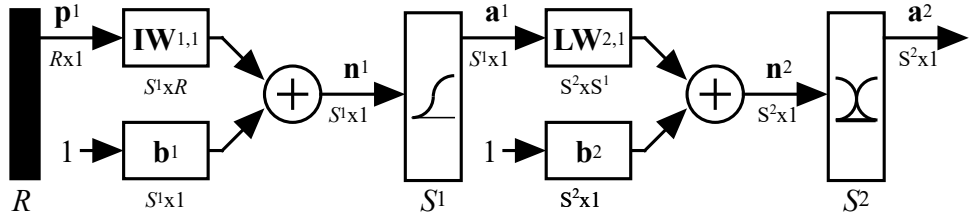

Figure 5.17: Classification Neural Network Architecture

A classification network outputs are calculated using forward propagation based on the generalized algorithm presented in Chapter 4 (Equation 4.7). The transfer function (f^1 and f^2) for the log-sigmoid and softmax are shown by Equations 5.4 and 5.5 [13].

$$sigmoid(n_i) = f^1(n_i) = \frac{1}{1 + e^{-n_i}} \tag{5.4}$$

where:

n_i = transfer function input

$$softmax(n_i) = f^2(n_i) = exp(n_i) + \sum_{j=1}^{S} exp(n_j) \tag{5.5}$$

where:

S = number of output categories

The classification network is trained using the backproprogation algorithm from Chapter 4 (Equation 4.13). To implement the backpropagation algorithm for the classification network shown in Figure 5.17, the derivatives of the transfer functions (f^1 and f^2) with respect to the transfer function input (n^1 and n^2) must be calculated ($\dot{\mathbf{F}}^1(\mathbf{n}^1)$ and $\dot{\mathbf{F}}^2(\mathbf{n}^2)$). The derivative of the log-sigmoid transfer function is shown by Equations 5.6 and 5.7 [13]. The derivative of the softmax transfer function is shown by Equation 5.8 [13]. The derivative of the log-sigmoid and softmax ($\dot{\mathbf{F}}^1(\mathbf{n}^1)$ and $\dot{\mathbf{F}}^2(\mathbf{n}^2)$) allow the layer's sensitivites (s^m) to be calculated for performing stochastic gradient descent (Equation 4.14).

$$\dot{f}^1_{n_i} = sigmoid(n_i) \cdot (1 - sigmoid(n_i)) \tag{5.6}$$

$$\dot{\mathbf{F}}^1(\mathbf{n}^1) = \begin{bmatrix} \dot{f}^1(n^1_1) & 0 & \cdots & 0 \\ 0 & \dot{f}^1(n^1_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{f}^1(n^1_{S^1}) \end{bmatrix} \tag{5.7}$$

$$\dot{\mathbf{F}}^2(\mathbf{n}^2) = \begin{bmatrix} a^2_1\left(\sum_{i=1}^{S^2} a^2_i - a^2_1\right) & -a^2_1 a^2_2 & \cdots & -a^2_1 a^2_{S^2} \\ -a^2_2 a^2_1 & a^2_2\left(\sum_{i=1}^{S^2} a^2_i - a^2_2\right) & \cdots & -a^2_2 a^2_{S^2} \\ \vdots & \vdots & \ddots & \vdots \\ -a^2_{S^2} a^2_1 & -a^2_{S^2} a^2_2 & \cdots & a^2_{S^2}\left(\sum_{i=1}^{S^2} a^2_i - a^2_{S^2}\right) \end{bmatrix} \tag{5.8}$$

5.4.5 Classification Neural Network Validation

For validation of a classification neural network, and to prevent over-fitting, the data set is divided into a training, validation and test set. The validation set is used for early stoppage to prevent over-fitting, while the test set is used for checking the accuracy of the network after training. This section will present early stopping using the validation set and classification accuracy using a confusion matrix.

The data set is typically divided into a training, validation, and test set. With 70% of the data set being used for training, while the remaining 30% of the data set being split between the validation and test set. Early stopping is based on ending the training of the neural network when the validation accuracy begins to decrease over a set number of iterations. The idea is that when the validation set accuracy decreases the classification neural network is over-fitting on the training data set and extrapolating on the validation data set. This is due to the validation data set not being used for tuning of the network weights, but for checking the accuracy of the network during training. After the network has been trained using early stopping, the test data set can be used to measure the final accuracy of the network. The test set is only used after the network has been trained and cannot be used during training.

After the classification network has been trained, a confusion matrix can be created for the test set. The confusion matrix is used to determine misclassification between categories.

To illustrate the use of a confusion matrix, an example is shown in Figure 5.18 [14]. This confusion matrix is for a problem with 3 categories. The output from the network is displayed on the y-axis and the targets for the network on the x-axis. From Figure 5.18 we can see that in 2 cases the network assigned a test input to class 2 when it actually belonged to class 3. This gave the network a classification accuracy of 91.3% on the test set. Based on this result, the misclassified inputs should be checked to determine if there is a similarity in inputs of class 2 and 3. This would

indicate that there are inputs from class 2 that overlap with inputs from class 3.

Figure 5.18: Example Confusion Matrix

5.4.6 Implementation of Classification Neural Network

To illustrate the application of the classification neural network in Figure 5.17 for fault detection, a neural network will be trained for classifying a load fault (warning and alert). The input into the network is the relative entropy for each known fault state (Figure 6.1, 6.2, and 6.3) against inline measured prediction error distribution. In this example, each input has 9 elements, because there are 3 fault states and 3 sensors.

The classification neural network weights are tuned using a data set of 597 inputs and targets. The input has a dimension of 9, as shown by Table 5.3, where there are 3 fault states (healthy, load warning, and load alert) and 3 sensors (Gearbox Angular

Velocity, Gearbox Torque, and Motor Flow Rate). The output from the network is 3 fault states (healthy, load warning, and load alert), the neural network uses the information from multiple sensors to develop rules for deciding the actual fault state. The network architecture is shown in Table 5.4.

Table 5.4: Load Fault Classification Neural Network Architecture

Data Set Size	Number of Inputs	Number of Outputs	Number of Neurons
597	9	3	10

The neural network for classifying a load fault is shown in Figure 5.19. The classification neural network is a small network, which is sufficient for classifying between 3 possible fault states using 9 inputs. The amount of training data limits the complexity of the network.

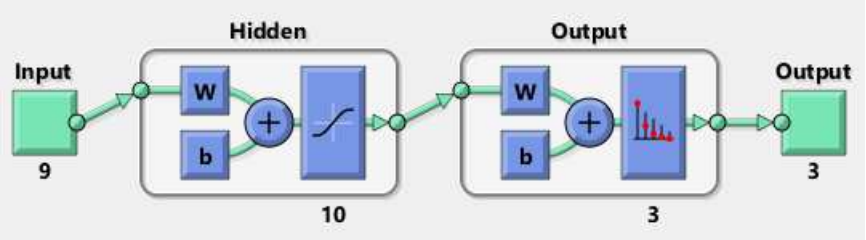

Figure 5.19: Matlab Classification Neural Network for Fault Detection

Using the backproprogation algorithm discussed in the previous section, the classification network shown in Figure 5.19 can be trained using the Matlab Neural Network Toolbox [14]. The confusion matrix is shown in Figure 5.20.

Figure 5.20: Load Fault Classification Network Confusion Matrix

From the confusion matrix, the classification neural network was able to achieve 100% accuracy on the test set. In the next chapter a classification neural network will be used for classifying between fault states for motor leakage, pump leakage, and load increases, which is more complex than classifying just the load fault states.

CHAPTER 6

RESULTS FOR HEALTH MONITORING

6.1 Introduction

In the Chapter 5, fault detection was presented and demonstrated for classifying single and multiple health states. This section will use the techniques presented in the previous chapter, but will expand the number of health states to include leakage in the pump and hydraulic motor. The classification neural network will be used to combine multiple sensors as shown in Chapter 5. The health state prediction error distributions are shown in Figures 6.1, 6.2, and 6.3 for healthy, load warning, load alert, pump leakage warning, pump leakage alert, motor leakage warning, and motor leakage alert. It can be seen from the figures that there is a significant amount of overlap between health state distributions, and a simple statistical test will not be sufficient to classify a fault.

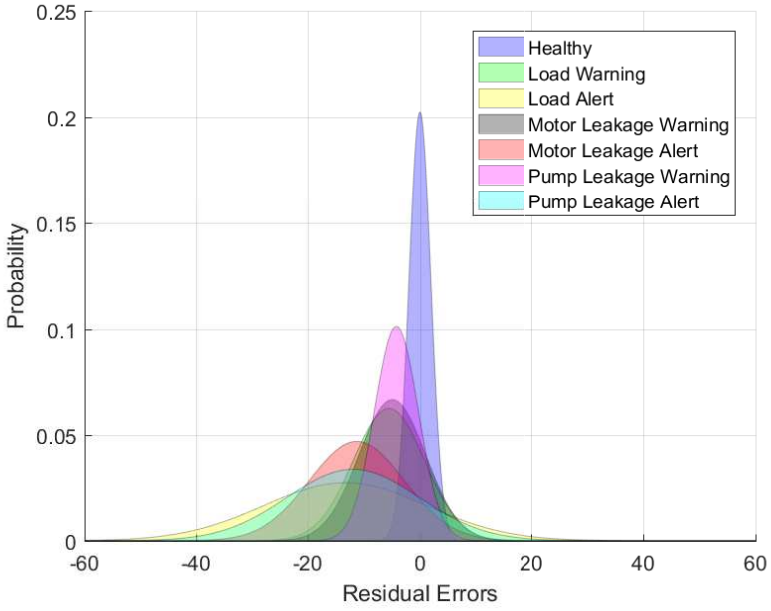

Figure 6.1: Fault PDFs for Gearbox Angular Velocity

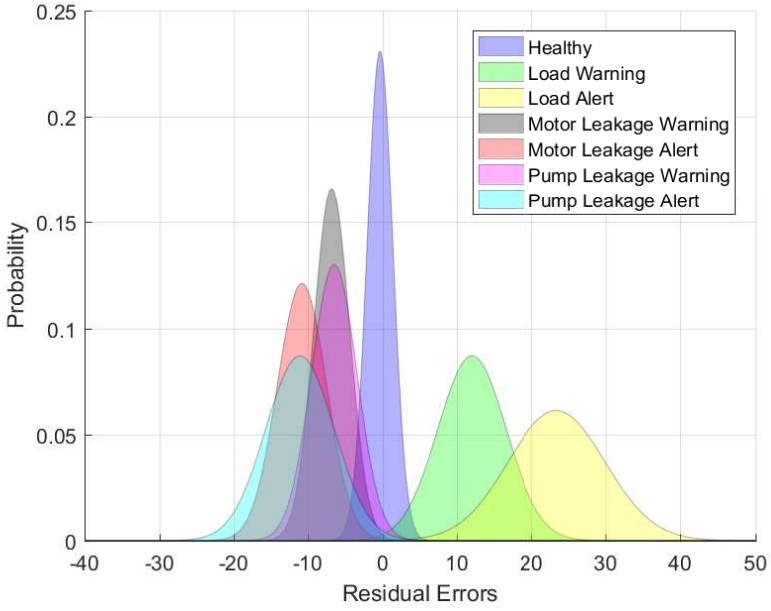

Figure 6.2: Fault PDFs for Gearbox Torque

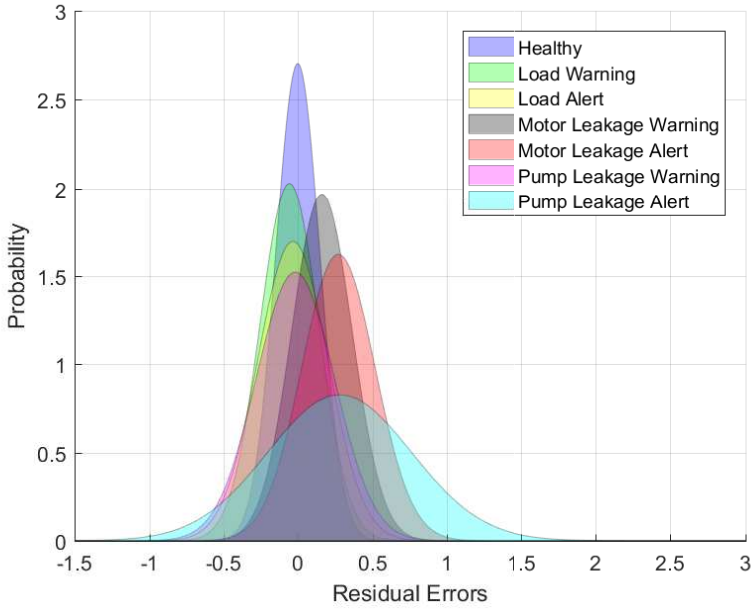

Figure 6.3: Fault PDFs for Motor Flow Rate

Even through there is overlap between fault distributions, there are some noticeable differences between health states. One difference, through inspection of Figure 6.8, is that the gearbox torque health state distribution will shift right when going from warning to alert. This correlates with the first order dynamic equation for a load in Chapter 2. As the load increases, the torque on the system will also increase. Pump and motor leakage health states show the opposite change, since loss in pressure from the pump will decrease the torque. Another observation is that an increase in load will decrease flow rate into the motor. The opposite is true for motor leakage, since when there is an increase in leakage there is less resistance to flow across the motor, which results in a decrease in pressure across the motor and in gearbox angular velocity. This correlates with the first order equations for the motor in Chapter 2. These noticeable differences in health state distributions give us motivation in implementing a classification neural network for distinguishing between health.

The technique for measuring the health state for a dynamic system is shown in Figure 6.4, which was discussed in Chapter 5. We will quickly re-introduce this

approach to insure the technique is understood before preceding. For each sensor there is a prediction error distribution, which is compared with the known health state distributions for each sensor. (The known health state distributions are shown in Figures 6.1, 6.2, and 6.3.) This is shown in Figure 6.4, where the prediction error for each sensor is based on the servo-valve input into a dynamic model that predicts $\widehat{y}(t|\theta)$, which is subtracted from the measured sensors $y(t)$. Then the prediction errors are sampled for an interval of time to produce a distribution, which is compared by statistical testing with known health state prediction error distributions for each sensor. The statistical test is Kullback-Leibler, which measures the relative entropy between two distributions. This produces multiple relative entropies for each sensor, which are combined using a classification neural network to make a prediction of the system's actual health state.

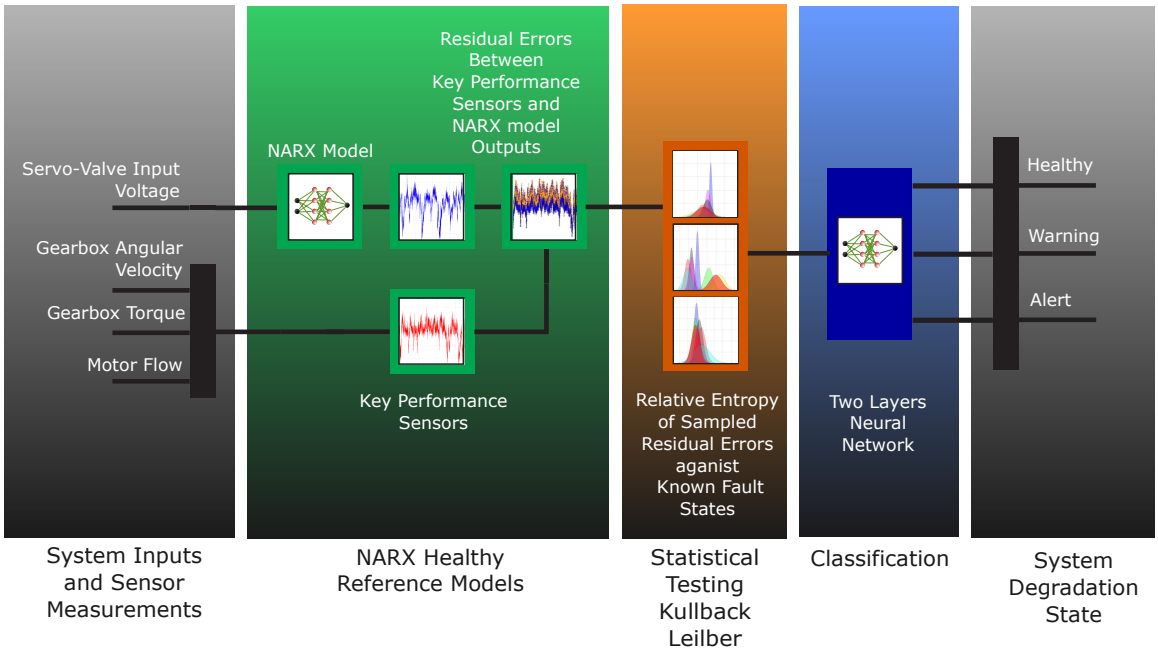

Figure 6.4: Multiple Load Faults Detection using Multiple Sensors Diagram

This chapter will present sensor and fault locations, along with the data set for creating the dynamic model, base heath state prediction error distributions, and classification neural network. Then Kullback-Leibler will be demonstrated for the health

states shown in Figures 6.1, 6.2, and 6.3, followed by training of the classification neural network. After the network is trained, the final results will be presented.

6.2 Sensor and Fault Locations

The key performance sensor locations are shown in Figure 6.5. For this chapter we are examining the health states of healthy, load fault warning, load fault alert, pump leakage warning, pump leakage alert, hydraulic motor leakage warning, and hydraulic motor leakage alert. The health states locations are shown in Figure 6.5. These health states were discussed in-depth in Chapter 2. A fault state of warning and alert for each sub-component is based on a percentage of reduction from the nominal gearbox angular velocity when operating with a 2 volt input into the servo-valve. For a warning and alert for each sub-component, the reduction in angular velocity is 15% and 30% from nominal operation.

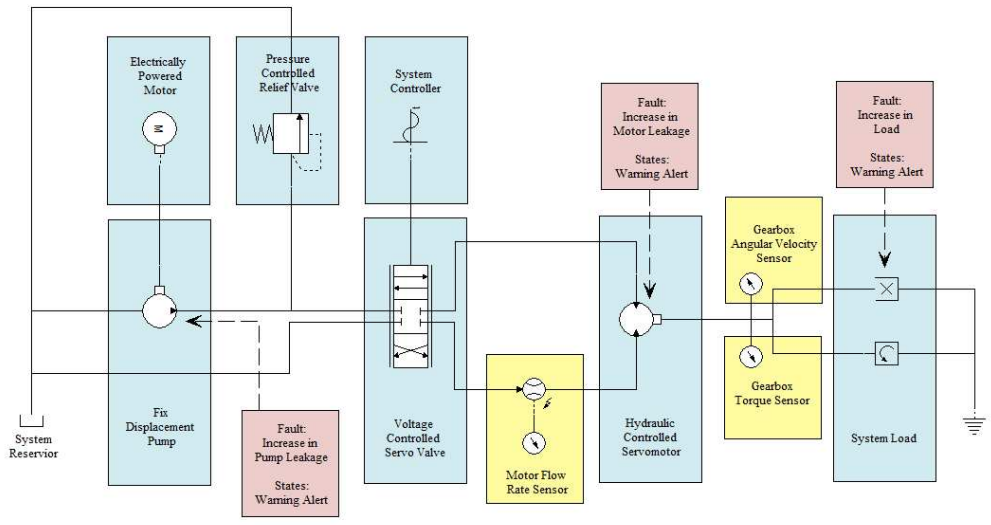

Figure 6.5: Hydrostatic Transmission Key Performance Sensors and Faults Locations

The dynamic model sampling rate and time interval will be the same as presented in Chapter 4, which is 200Hz and 5 seconds for 1,000 points. These 1,000 points will

be used to classify the health state of the hydrostatic transmission in real-time.

6.3 Data Collection

The process for fault detection is shown in Figure 6.4, which was discussed in-depth in Chapter 5. There are three parts to the fault detection process: train the dynamic healthy state NARX model, develop the base health state prediction error distributions, and train the classification neural network. This section will discuss the data used for creating the dynamic NARX model, base health state distributions, and classification neural network. A diagram of the data used is shown in Figure 6.6. There are 250,000 points for each of the health states shown in Figure 6.5, which are split between the following three parts.

1. Dynamic Model (NARX): The dynamic model for the health reference state is trained on 10,000 points, while the system was operated in a healthy state.

2. Base Prediction Error Distributions for Statistical Testing: The base health state prediction error distributions are created using 40,000 points.

3. Sensor Fusion (Classification Neural Network): The classification neural network is trained using 1,400 inputs, composed of 200 inputs from each health state. Each input into the network is the relative entropy of an inline prediction error distribution against the base health state distributions for each sensor. There are 3 sensors and 7 health state distributions, which creates an input with a dimension of 7 x 3 = 21. An inline distribution is created from 1,000 points sampled at 200 Hz for 5 seconds. For each health state 200,000 points are used for generating 200 inline distributions. The network target dimension is 7 for the health states. The color of the data sets in Figure 6.6 correlate with the color of the distributions in Figures 6.1, 6.2, and 6.3.

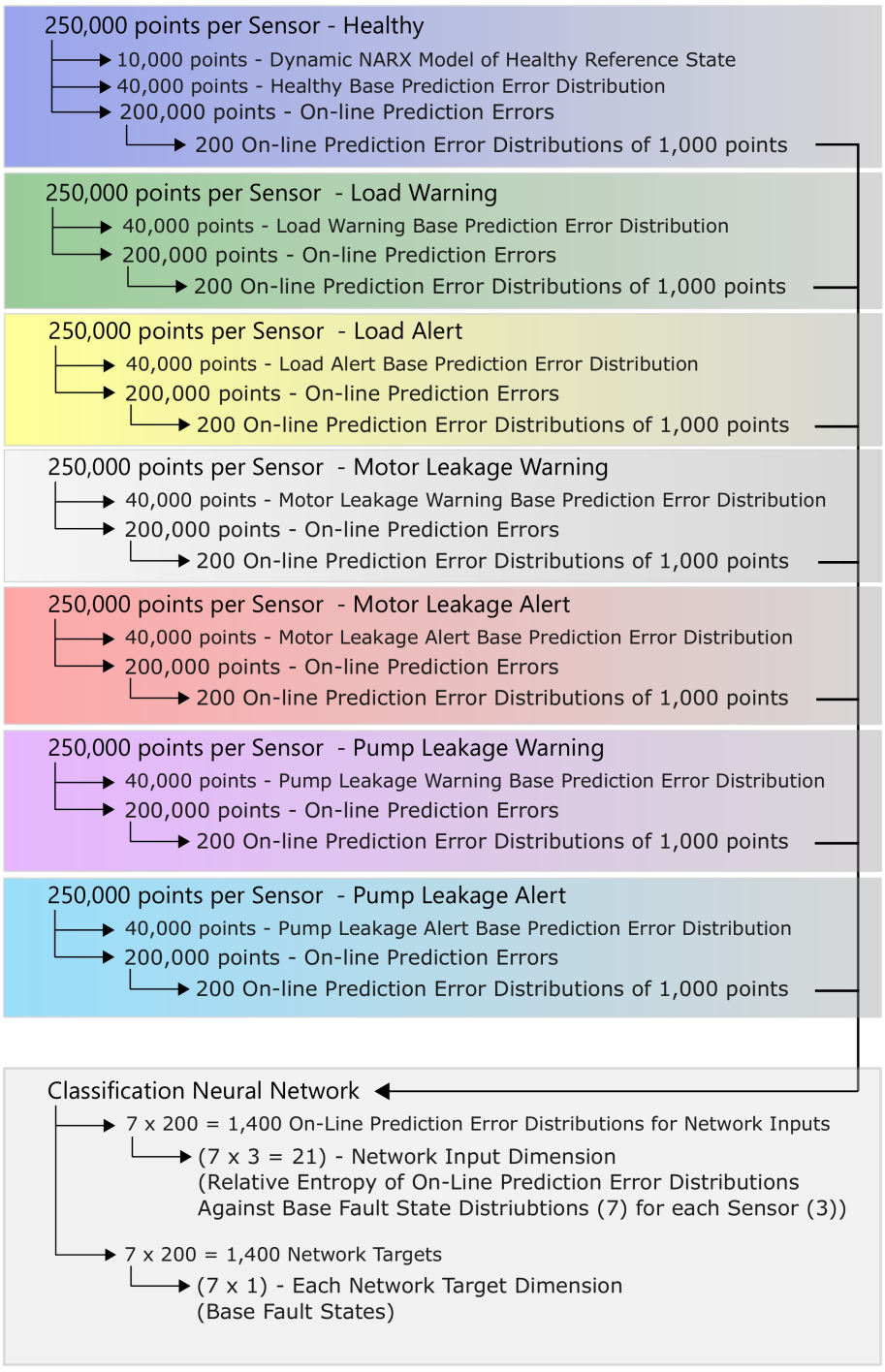

Figure 6.6: Data Diagram

6.4 Implementing the Statistical Test

To predict the health state for the hydrostatic transmission, the relative entropy of the inline prediction error will be compared with each of the health state distributions. An example of a sampled prediction error distribution against the known health state distributions are shown in Figures 6.7, 6.8, and 6.9. (The sampled distribution was obtained from 1,000 points when the system operating under the conditions fro a load alert.)

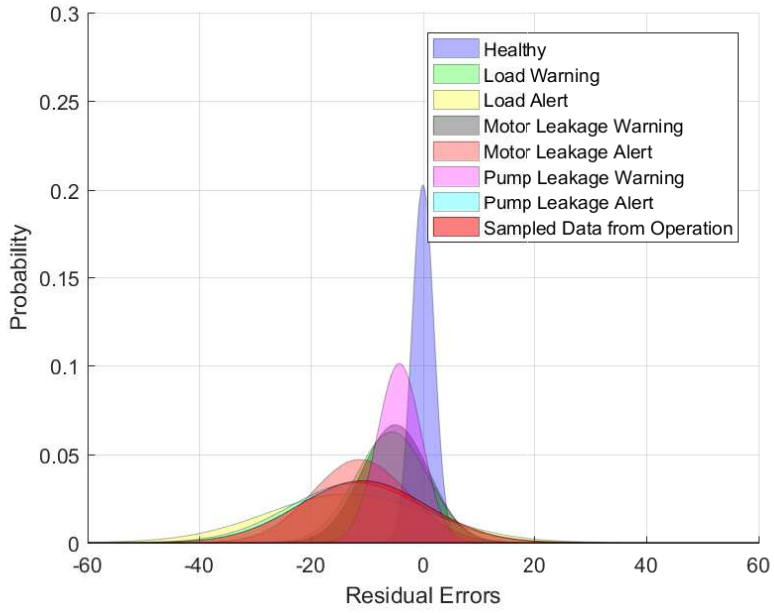

Figure 6.7: Fault PDFs for Gearbox Angular Velocity

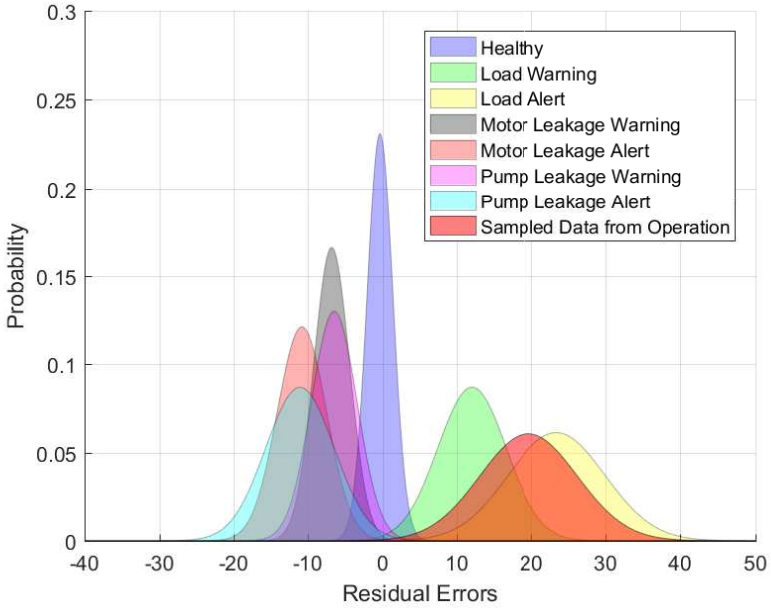

Figure 6.8: Fault PDFs for Gearbox Torque

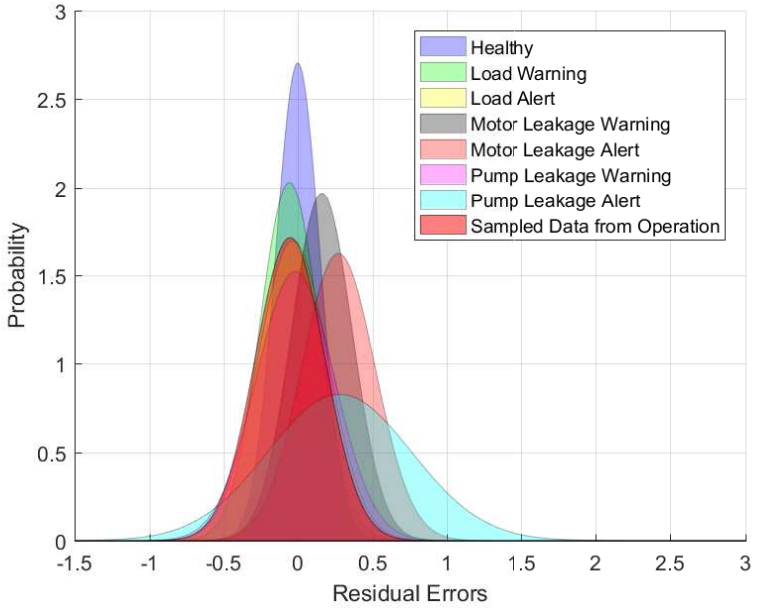

Figure 6.9: Fault PDFs for Motor Flow Rate

The continuous Gaussian Kullback-Leilber test, Equation 5.3, can be used for the distributions in Figure 6.7, 6.8, and 6.9 to calculate the relative entropy between the

inline prediction error distribution and known health state distributions. The relative entropy of the distributions are shown in Table 6.1. It can be seen from the table, that the system's current health is load alert, since load alert has the lowest entropy for gearbox torque and motor flow rate. An issue, though, is that the gearbox angular velocity's entropy indicates a motor leakage alert. In the next section a classification neural network will be trained to generate rules for combining the information from multiple sensors to make a single health state prediction. This will improve prediction accuracy, since each sensor recognizes different dynamic information about the system.

Table 6.1: Load Fault States Relative Entropy vs Sampled Operational Data

Sensors	Healthy	Load Warning	Load Alert	Pump Leakage Warning	Pump Leakage Alert	Motor Leakage Warning	Motor Leakage Alert
Gearbox Angular Velocity	29.6664	0.8590	0.0660	1.1565	0.1091	4.0347	0.0059
Gearbox Torque	72.2791	1.5489	0.1679	63.0359	43.6429	37.2744	22.5967
Motor Flow Rate	0.3507	0.0324	0.0032	05757	0.8717	0.0218	0.5819

6.5 Neural Network for Classification

The classification neural network is trained using the backproprogration algorithm, which was presented in Chapter 5. The inputs and targets for training the network are shown in Figure 6.6. The network architecture for fault detection is shown in Figure 5.17, with the network design parameters (hidden neurons) being shown in

Table 6.2. The network was trained using 15 hidden neurons, since the training data set is relatively small and too many neurons will lead to over fitting issues.

Table 6.2: Fault Detection Classification Neural Network Architecture

Data Set Size	Number of Inputs	Number of Outputs	Number of Neurons
1,400	21	7	15

The classification network is trained using the Matlab Neural Network Toolbox [14]. The input data set is split between training, validation and test percentages shown in Table 6.3.

Table 6.3: Fault Detection Data Set Split Between Training, Validation, and Test

Training	Validation	Test
70%	15%	15%

The confusion matrix after training the classification neural network is shown in Figure 6.10. Based on this figure, the classification neural network is able to perfectly categorize a fault in the system for the original training, test, and validation sets. In the next section a separate test set will be used to validate the fault detection performance. This separate test set was collected within the same operational ranges, but at a different operational time.

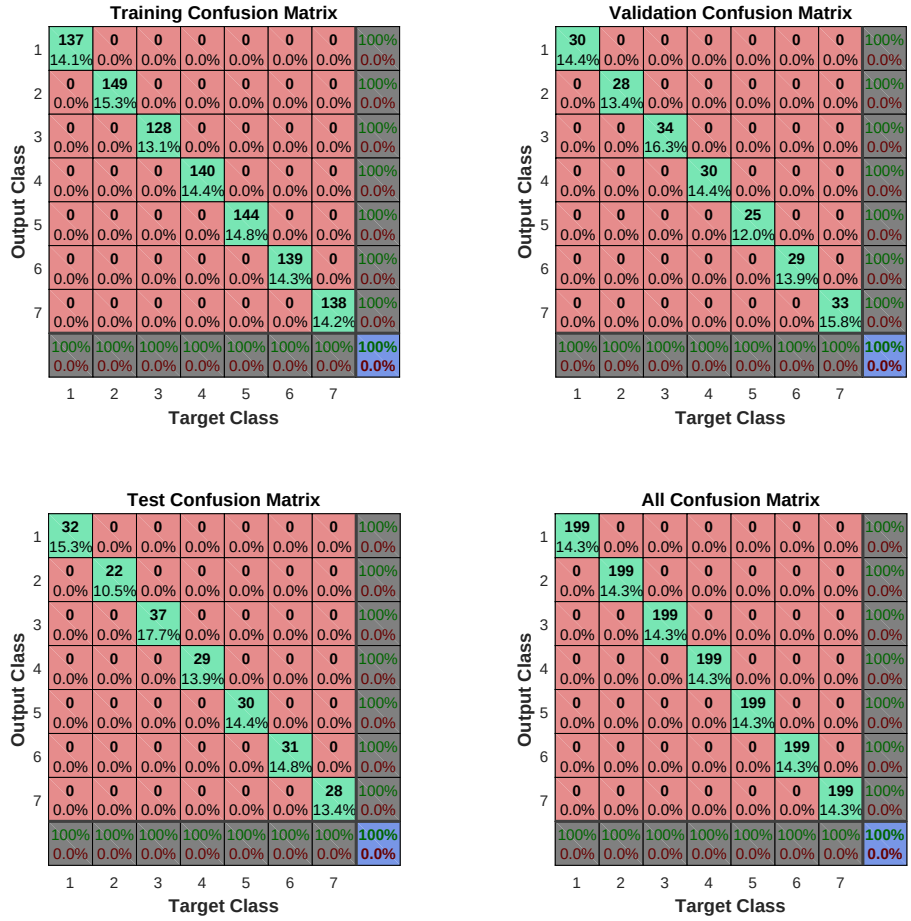

Figure 6.10: Trained Network Confusion Matrix

6.6 Results

To validate the accuracy of the fault detection approach shown in Figure 6.4, a separate test set that was collected at a different operational time will be tested using the trained fault prediction models shown in Figure 6.4. The confusion matrix for the separate test set is shown in Figure 6.11. The classes in the confusion matrix correlate with the health states shown in Table 6.4. Based on the confusion matrix, the fault detection system was able to achieve an accuracy of 96.7%. The misclassifications occur between warning and alert within the categorizes of load, pump, and

motor fault. This indicates that the fault detection system is properly categorizing the degradation, but is misclassifying the severity of the degradation. The number of misclassifications is reasonably small, with misclassification error rate of 14.7% for motor leakage warning and 7% for pump leakage warning. The accuracy can be improved by increasing the data set for training the neural network by collecting more data, but based on the performance of the fault detection model a reasonable prediction of the current health state can be made in real-time.

Table 6.4: Classes Number and Health State

Sensors	Healthy	Load Warning	Load Alert	Pump Leakage Warning	Pump Leakage Alert	Motor Leakage Warning	Motor Leakage Alert
Classes	1	2	3	4	5	6	7

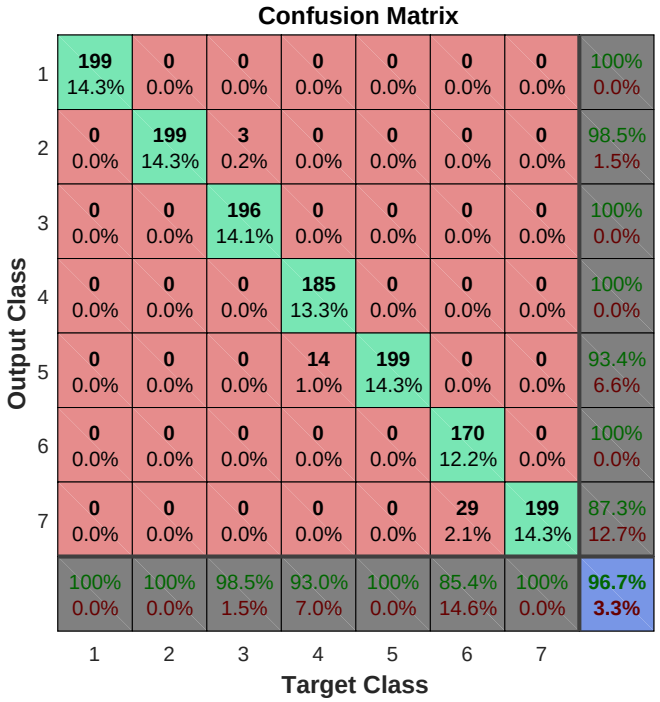

Figure 6.11: Trained Network Confusion Matrix

Using a prediction of the current health state of the hydrostatic transmission, a plan can be developed by the maintenance group to decide whether the parts in system need to be further tested or replaced. In a real application this approach to fault detection will be implemented online, while it is operating with changing inputs. If a warning or alert is indicated, the maintenance group can use the fault prediction model to determine the most probable defective part, which reduces the time of performing maintenance, since finding the location of a fault in dynamic system can be difficult.

CHAPTER 7

CONCLUSIONS

This thesis presented an approach for predicting faults in a hydrostatic transmission. This approach was designed for in-line fault detection, where the inputs into the system are allowed to vary.

The thesis began by discussing the mechanical system and the differential equations for the sub-components, which were used to illustrate how wear in the sub-components effects system performance. To measure the level of wear, we needed a health reference model that accounts for changing inputs into the system. This could be achieved by the differential equations, but the coefficients are difficult to directly measure. It would require special test equipment and the disassembly of the sub-components for measuring part tolerances, which is labor intensive. To overcome this issue, system identification approaches were presented for the modeling the system.

We began the discussion of system identification by presenting the Box-Jenkins linear modeling approach, which has been used in a large number of industrial applications. The Box-Jenkins model strength is that the dynamics and disturbance for the system can be separated by two difference equations. The Box-Jenkins model is trained on a set of healthy inputs and outputs, which was demonstrated for the gearbox angular velocity. The steps for system identification are: preliminary identification, parameter estimation, and model validation. Each of these steps were discussed in detail, with the final trained Box-Jenkins model being a good fit to the actual system. The Box-Jenkins model is linear and cannot be fitted over the entire operational range of the system. To overcome this issue, a non-linear model, known

as the non-linear autoregressive exogenous (NARX) model, was presented for creating a health reference state.

The NARX model is a black box model, where the coefficients of the model have no direct correlation to the system's differential equations and do not provide any information about how well the model fits the physical system. The benefit of the NARX model is that it can be fitted over the entire operational range of the system. With the NARX model being a black box approach, there are no preliminary identification techniques. Instead, a trial and error approach is used for tuning the weights using parameter estimation and model validation. After the NARX model was trained, it was shown that it provided a good fit to the physical system for the entire operational space. This was shown to be beneficial, since it improves accuracy and reduces the need for multiple models.

With a health reference model developed for the system, statistical testing and classification techniques were presented for detecting a fault. This was demonstrated by discussing how the prediction error of the healthy reference model increases with wear of the physical system. The healthy reference model is a good fit to the healthy system, but as sub-components in the system degrade, the dynamics of the system change and the prediction error changes with degradation. It was shown that by statistical testing and classification techniques, a sub-component fault state can be classified by the prediction errors of multiple health reference models, while the system is being operated on-line. Chapter 6 showed that the failures in a hydraulic system can be accurately predicted in real-time.

The ability to predict the current health state of the system allows a maintenance plan to be developed. By being able to predict a health state, the cost and time required for maintenance can be improved, since knowledge of the system's health condition will improve the repair time and prevent unnecessary removal of healthy sub-components. Fault detection is an important issue for hydraulic systems, where

degradation of sub-components is a common problem that is difficult to measure. The method presented in this thesis, in which the distributions of NARX prediction errors are used to automatically determine the health state of the system, shows promise for on-line fault detection.

7.0.1 Future Work

The approach presented for fault detection was shown to be beneficial, but there are some improvements and further testing that can be done to improve this technique. An issue with the approach presented was that it was assumed there were no environmental effects on the system. This can be a safe assumption, when the hydraulic system is being used for manufacturing applications, but for aerospace and construction equipment the environmental effects play a large role on the dynamics of the system. Environmental effects can be treated as an input into the dynamic model, which increases the complexity of the model. In aerospace applications a common environmental effect is temperature changes during take off. This results in a temperature change of 25C to -40C in under 15 minutes. Temperature can be included in the non-linear NARX model presented in Chapter 5, but due to the temperature being a stiff input into the system, the dynamic model will be difficult to train. Instead, multiple NARX models can be trained at different temperature ranges. This would require switching between dynamic models based on environmental conditions.

Another improvement, would be demonstrating how to develop a classification model, presented in Chapters 5 and 6, that can be trained as fault data becomes available. In a real world applications not all data are available for failure states and have to be added during operation of the system. A technique that can be used is adaptive fuzzy logic, where the initial rules are based on expert knowledge from a maintenance group, but as faults occur and are measured, the fuzzy logic model can be updated. This would allow the classification model presented in Chapter 5 and 6

to be continually developed over the entire operational life of the system.

The last improvement is implementing Gaussian mixtures for statistical testing, instead of fitting the prediction errors with a single Gaussian (Chapter 5 and 6). It can be seen from Figure 5.3 that the histogram is skewed, which results in the Gaussian being unable to accurately fit the prediction errors. A Gaussian mixture would better fit the prediction errors and would improve fault state classification accuracy.

BIBLIOGRAPHY

[1] M. Dong, C. Liu, and G. Li, "Robust fault diagnosis based on nonlinear model of hydraulic gauge control system on rolling mill," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 510–515, 2010.

[2] P. Garimella and B. Yao, "Model based fault detection of an electro-hydraulic cylinder," in *American Control Conference, 2005. Proceedings of the 2005*, pp. 484–489, IEEE, 2005.

[3] M. Khoshzaban-Zavarehi, *On-line condition monitoring and fault diagnosis in hydraulic system components using parameter estimation and pattern classification*. PhD thesis, University of British Columbia, 1997.

[4] L. Hongmei, W. Shaoping, and O. Pingchao, "Fault diagnosis based on improved elman neural network for a hydraulic servo system," in *Robotics, Automation and Mechatronics, 2006 IEEE Conference on*, pp. 1–6, IEEE, 2006.

[5] K. Mollazade, H. Ahmadi, M. Omid, and R. Alimardani, "An intelligent combined method based on power spectral density, decision trees and fuzzy logic for hydraulic pumps fault diagnosis," *International Journal of Intelligent Systems and Technologies*, vol. 3, no. 4, pp. 251–263, 2008.

[6] M. Demetgul, I. N. Tansel, and S. Taskin, "Fault diagnosis of pneumatic systems with artificial neural network algorithms," *Expert Systems with Applications*, vol. 36, no. 7, pp. 10512–10519, 2009.

[7] H. E. Merritt, *Hydraulic control systems*. John Wiley & Sons, 1967.

[8] E. Fitch and I. Hong, "Hydraulic component design and selection: Bardyne," 2004.

[9] G. E. Box, G. M. Jenkins, and G. C. Reinsel, *Time series analysis: forecasting and control*. John Wiley & Sons, 2008.

[10] L. Ljung, "System identification: Theory for the user, ptr prentice hall information and system sciences series," *ed: Prentice Hall, New Jersey*, 1999.

[11] W. A. Woodward and H. L. Gray, "On the relationship between the s array and the box-jenkins method of arma model identification," *Journal of the American Statistical Association*, vol. 76, no. 375, pp. 579–587, 1981.

[12] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[13] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesus, *Neural network design*. Martin Hagan, 2014.

[14] H. B. Demuth, M. H. Beale, and M. T. Hagan, "Matlab neural network toolbox," 9.1.0.441655 (R2016b). The MathWorks, Natick, MA, USA.

[15] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

VITA

Joseph Kelley

Candidate for the Degree of

Master of Science

Dissertation: HYDRAULIC SYSTEMS DEGRADATION DETECTION USING SPARSE SENSORS

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Raytown, Missouri, United States on June 21, 1989.

Education:

Received the B.S. degree from University of Missouri Kansas City, Kansas City, Missouri, United States, 2012, Mechanical Engineering
Completed the requirements for the degree of Master of Science with a major in Electrical Engineering from Oklahoma State University in May, 2018.

Experience:

Mechanical Engineer II at Hallmark from February 2013 - February 2014
R&D Engineer at FES, Inc from February 2015 to Current