Fatigue Structural Testing Enhancement Research (FASTER)

by

Robyn Fortune

Department of Mechanical Engineering
McGill University, Montreal
December 2019

A thesis submitted to McGill University
in partial fulfillment of the requirements of the degree of
Master of Engineering

Supervisor:

Professor James Richard Forbes

Robyn Fortune
robyn.fortune@mail.mcgill.ca

© Robyn Fortune 2019

All Rights Reserved

Acknowledgements

First, I would like to thank my supervisor, Prof. James Forbes, for being an exceptional mentor. I greatly appreciated his constant insight and enthusiasm throughout the project. I also want to thank André Beltempo, Stéphane Brunet, and Cathy Cheung at the National Research Council (NRC) for all of their support and feedback, especially during my time at NRC. I could not have done it without all of you.

I would like to acknowledge the financial support of the NRC, the Natural Sciences and Engineering Research Council of Canada (NSERC), the Fonds de recherche du Québec - Nature et technologies (FRQNT), and the Department of Mechanical Engineering at McGill University. In addition, I want to thank NRC not just for the funding, but for providing access to their large-scale test facilities and the personnel support. I cannot count the number of times my test caused the hydraulics to shut down, and yet Andy Christie, Denis Beaulieu, and Steph Cloutier always fixed it with a smile on their faces.

I want to thank Niels van der Laan for all of his help with reduced-order modelling. In particular, Niels collaborated on the results presented in Section 3.1.2.4 and Appendix A. Furthermore, I want to thank the rest of my research group, especially my fellow master's students Jon Arsenault and Ken Lee, and (now) Prof. Ryan Caverly. Ryan was an invaluable resource to me during our time at McGill, and continued to promptly respond to all of my email queries even after leaving.

I greatly appreciate the support of my family. Even though I'm pretty sure they (and especially Buddy) can't explain what I've been working on for the past two years, they have always been there for me. I also want to thank all of my friends for their support, and especially Adam, Danny, Katie, Marina, Emma, and Rachel for making the move to Ottawa so much easier. I very much appreciate the support I received from Anthea and Sabrina, although they made the move from Montreal that much harder. I would like to thank Jason and especially Sookie for being the perfect roommates while I was in Montreal, and big thanks to Will for housing me during my time in Ottawa. Many thanks to Kevin for the support throughout the thesis writing process, and for the snacks.

Finally, I want to thank the Dominion Arboretum for providing a peaceful and idyllic environment in which I spent a lot of time working on this thesis.

Preface

The contributions of this thesis that are original to the author's knowledge are as follows. In Chapter 5, a novel iterative method for synthesizing $\mathcal{H}_\infty$-optimal PI controllers is presented. In Chapters 6 and 7, the application and implementation of SISO and MIMO 2DOF $\mathcal{H}_\infty$ controllers to a fatigue structural testing rig is presented. Although the inversion-based feedforward controller synthesis method and the 1DOF and 2DOF $\mathcal{H}_\infty$ controller synthesis methods presented are not novel, these types of controllers have not previously been implemented on a load-controlled fatigue structural testing rig to the author's knowledge.

All text, plots, illustrations, and numerical and experimental results in this thesis are produced by Robyn Fortune, although Niels van der Laan assisted with the reduced-order modelling in Section 3.1.2.4 and Appendix A.

Table of Contents

List of Figures

Figure

List of Tables

Table

List of Appendices

Appendix

List of Acronyms

PI	Proportional-integral
PID	Proportional-integral-derivative
1DOF	One degree-of-freedom
2DOF	Two degree-of-freedom
SISO	Single-input-single-output
MIMO	Multi-input-multi-output
LTI	Linear time-invariant
BIBO	Bounded-input-bounded-output
OLHP	Open-left-half-plane
ORHP	Open-right-half-plane
RMS	Root-mean-square
NRMS	Normalized root-mean-square
VAF	Variance accounted for
MP	Minimum phase
NMP	Non-minimum phase
CCC	Cross-coupling compensation
FLF	Forward loop filter
PSO	Profile segment optimization
PCHIP	Piecewise cubic hermite interpolating polynomial

List of Symbols

$\mathbb{R}$	the set of real numbers
$\mathbb{R}^n$	the vector space of real n dimensional vectors
$\mathbb{R}^{m\times n}$	the space of real $m \times n$ dimensional matrices
$\mathrm{tr}(\cdot)$	trace of a matrix
$(\cdot)^\mathsf{T}$	transpose of a matrix
$(\cdot)^{-1}$	inverse of a matrix
$\mathbf{0}$	zero matrix
$\mathbf{1}$	identity matrix
$\star$	symmetric portion of a matrix
$\|\cdot\|_2$	2-norm
$\|\cdot\|_\mathsf{F}$	Frobenius norm
$\|\cdot\|_\infty$	$\mathcal{H}_\infty$ norm
k_p	proportional gain
k_i	integral gain
$\mathbf{M} > 0$	matrix $\mathbf{M}$ is symmetric positive definite
$\mathcal{H}_\infty$	H-infinity
$\mathcal{H}_\infty^I$	H-infinity with integrator

Abstract

This thesis focuses on the modelling and optimal control of a load-controlled fatigue structural testing rig. The modelling phase involves first attempting to analytically model the test system and the test controller, then using these models to estimate test cycle times. Afterwards, a system identification approach is taken to generate a more reliable numerical model using data. Open-loop and closed-loop methods are discussed, although only closed-loop experiments can be performed on fatigue testing rigs to prevent unnecessary damage to the valuable test article. The direct, indirect, and dual-Youla closed-loop system identification methods are applied to measurement data from a fatigue testing rig at the National Research Council of Canada (NRC). The identified models are validated then used in various controller synthesis methods. First, two methods for generating "optimal" proportional-integral (PI) gains are presented. The first uses $\mathcal{H}_\infty$-optimal static output feedback, and the second employs the Bounded Real Lemma, iteration, and bisection method. Next, a single-input-single-output (SISO) approach to designing two degree-of-freedom (2DOF) controllers is presented. The feedback controller can be a PI or $\mathcal{H}_\infty$ controller, for example, and the feedforward controller is designed using an approximate inverse of the plant transfer function. Finally, a multi-input-multi-output (MIMO) 2DOF $\mathcal{H}_\infty$-optimal controller synthesis method is described. The alternative controllers are implemented on the test rig and used to perform tests. Tracking results and their comparison to the standard PI controller are presented.

Résumé

Cette thèse porte sur la modélisation et le contrôle optimal d'un banc d'essai structurel de fatigue à charge contrôlée. La phase de modélisation consiste d'abord à tenter de modéliser analytiquement le système de test et le contrôleur, puis à utiliser ces modèles pour estimer la durée des cycles de test. Ensuite, une approche d'identification du système est adoptée pour générer un modèle numérique plus fiable utilisant des données mesurées. Les méthodes en boucle ouverte et en boucle fermée sont discutées, bien que seulement des expériences en boucle fermée puissent être effectuées sur des bancs d'essais de fatigue, afin d'éviter les dommages inutiles à l'article de test de valeur importante. Les méthodes directe, indirecte et dual-Youla d'identification de systèmes en boucle fermée sont appliquées aux données mesurées d'un banc d'essai de fatigue du Conseil National de Recherches du Canada (CNRC). Les modèles identifiés sont validés puis utilisés dans différentes méthodes de synthèse de contrôleur. Premièrement, deux méthodes pour générer des gains proportionnel et intégral (PI) «optimaux» sont présentées. La première utilise la rétroaction à sortie statique $\mathcal{H}_\infty$-optimal, et la seconde utilise la méthode du lemme réel borné (Bounded Real Lemma), de l'itération et de la bissection. Ensuite, une approche à entrée simple et sortie simple (SISO) pour la conception de contrôleur à deux degrés de liberté (2DOF) est présentée. Le contrôleur de rétroaction peut être un contrôleur PI ou $\mathcal{H}_\infty$, par exemple, et le contrôleur adaptatif par action anticipatrice («feedforward») est conçu en utilisant un inverse approximé de la fonction de transfert du procédé. Enfin, une méthode de synthèse de contrôleurs $\mathcal{H}_\infty$ optimaux à deux degrés de liberté, à entrées multiples et sorties multiples (MIMO) est décrite. Les contrôleurs alternatifs sont implémentés sur le banc d'essai et utilisés pour effectuer des tests. Les erreurs de poursuite et leur comparaison avec le contrôleur PI standard sont présentées.

Part I

Introduction

Chapter 1

Introduction

1.1 Motivation and Objectives

Fatigue testing is a necessary part of aircraft certification, where aircraft load conditions are simulated through the use of hydraulic actuators that apply loads to the airframe [2]. Tests must account for the service lifetime of the aircraft and include relevant operating conditions. Despite its ubiquity in the aerospace industry, fatigue testing remains a time-consuming process, with large-scale fatigue tests taking weeks, months, and even years to complete [3]. Without a reliable numerical model, there is no method to assess the optimality or robustness of the test system, provide an initial estimate of adequate controller gains, or synthesize model-based controllers. The controllers themselves are typically proportional-integral (PI) controllers, which have advantages in terms of familiarity and simplicity, but are not necessarily optimal in terms of performance and robustness. The practice of fatigue testing is consequently governed by "rules of thumb" where tuning of the controllers is performed by hand. Evidently there is a need to improve the fatigue testing process.

Fatigue Structural Testing Enhancement Research (FASTER), a research partnership between McGill University and the National Research Council of Canada (NRC), aims to improve the efficiency of large-scale fatigue testing. The initial goals of the project are to provide a numerical basis for trade-offs in hardware selection and provide improved estimates of system cycle times and initial gain settings. These goals will be accomplished by modelling the system components and using the models in controller synthesis and simulation. In the long term, FASTER intends to use the developed tools to then push the state-of-the-art in terms of test control, in an attempt to maximize cycle times to the limits of the physical systems. The second phase of the project is more exploratory, and involves the design and implementation of alternative controllers. Ultimately, the goal of the project is to allow fatigue tests to run *faster*.

1.2 Prior Work

Previous work was completed at the NRC on the modelling and cycle estimation of a fatigue structural testing rig. Both Hewitt [4, 5] and Cheung [6] worked on modelling the components of a fatigue test system. Cheung also developed a simulation environment using `MATLAB`/*Simulink* in which a single-actuator fatigue test could be simulated using selected hardware models. Much of the fatigue testing equipment at NRC is supplied by the MTS Systems Corporation, including hydraulic components, such as servovalves and actuators, and test controllers. The work of Hewitt and Cheung focused on an earlier version of the MTS controller, but the servovalves and actuators they modelled are the same as those in use today. The models from Hewitt and Cheung thus provide a starting point for the analytical models developed in Chapter 3.

Additionally, Zlotnik [7] completed some preliminary work on the system identification and $\mathcal{H}_2$-optimal control of a helicopter tail boom structural health monitoring test. Zlotnik's work was useful in learning how to implement alternative controller schemes within the MTS system, and provided an initial review of some closed-loop system identification methods. However, Zlotnik's work did not include formal tests of the $\mathcal{H}_2$ controller, or its comparison to PI control. Although the optimal controllers implemented in this thesis are of the $\mathcal{H}_\infty$ variety, $\mathcal{H}_2$ controllers can be implemented in a similar fashion.

1.3 Thesis Overview

This thesis is divided into two parts. The first part focuses on the development of numerical models for fatigue structural testing rigs, both using first-principles modelling and system identification techniques. In addition, the MTS controller is modelled and a cycle estimation tool is developed. The second part includes various controller synthesis methods. Two methods for synthesizing "optimal" PI controllers are presented. Next, methods for synthesizing two degree-of-freedom (2DOF) controllers for single-input-single-output (SISO) and multi-input-multi-output (MIMO) systems are presented.

Although the models and methods included in this thesis can theoretically be applied to any fatigue structural testing rig, the test system used to generate experimental results is the SHM platform at NRC. Pictured in Fig. 1.1, the SHM platform consists of an aluminum-composite beam with four hydraulic actuators, one at each corner. The beam is welded at its center to a metal post. Attached to each actuator is a hydraulic manifold and servovalve, and a load cell that measures the applied force. The test controller applies a voltage to a two-stage servovalve with a hydraulically-piloted spool that amplifies the solenoid voltage

to provide full pump flow to either chamber of the loading actuator [8]. In this way, the actuator can extend or retract as flow is metered through the servovalve. This actuator movement applies a force to the test article that is measured by the load cell.

Figure 1.1: The SHM platform at NRC.

1.3.1 Modelling and System Identification

The test rig itself can be broken down into three main systems. The structure undergoing fatigue testing is known as the test article, which can range from an entire airplane to an aircraft part, such as a wing. The hydraulic system, which consists of actuators, servovalves, load cells, pumps, and accumulators, is used to apply the loads to the article. Lastly, the test structure serves to keep the article and hydraulics in place and distribute loads to the test article [4]. Although modelling each distinct system may be possible, attempting to accurately capture the interactions between them in an analytical model proves to be extremely difficult.

Nevertheless, an attempt was made to model the key components of a fatigue structural testing rig from first-principles. The models, along with a tool developed to estimate cycle times, are detailed in Chapter 3. In Chapter 4, the linear time-domain system identification problem is explained. System identification is a powerful technique for generating numerical models for systems that are difficult to model analytically. Input-output data is used to solve for model parameters using a least squares approach. Open- and closed-loop system identification methods are presented, however, closed-loop methods must be used in the context of fatigue testing to avoid damaging the test article. Identification results are presented for the SHM platform using first SISO then MIMO data.

1.3.2 Controller Synthesis

In Chapter 5, two "optimal" PI controller synthesis methods are presented. The first uses $\mathcal{H}_\infty$-optimal static output feedback, and the second is a novel method for locally minimizing the closed-loop $\mathcal{H}_\infty$ norm using the Bounded Real Lemma, iteration, and bisection method. Because the optimization problem is non-convex, the resulting controllers are only optimal locally and not globally. The algorithms were applied to the analytical model from Chapter 3 and the SISO identified model from Chapter 4; the controllers and corresponding simulation results are presented. Chapter 6 presents a method for designing SISO 2DOF controllers, which are controllers that include both feedback and feedforward. In the SISO case, the feedback and feedforward controllers can be designed independently, where the feedback controller is PI or $\mathcal{H}_\infty$, for example, and the feedforward controller is designed using an approximate inverse of the plant model. Because inverting a MIMO system is not straightforward, Chapter 7 presents a method for synthesizing MIMO 2DOF controllers using $\mathcal{H}_\infty$ controller synthesis methods with embedded feedforward. The SISO and MIMO 2DOF controllers were implemented on the SHM platform and experimental results are presented.

Chapter 2

Preliminaries

In the chapter, standard tools relevant to system identification, optimization, and controller synthesis are reviewed. These tools serve as the foundation for the methods and algorithms described in later chapters.

2.1 Discrete-Time Systems

A discrete-time system can be described by a difference equation, where the current output is written as a linear combination of n past outputs and m past inputs. The MIMO difference equation is given by [9]

$$\mathbf{y}_k = -\mathbf{A}_{n-1}\mathbf{y}_{k-1} - \cdots - \mathbf{A}_1\mathbf{y}_{k-n+1} - \mathbf{A}_0\mathbf{y}_{k-n} + \mathbf{B}_m\mathbf{u}_{k-\tau} + \cdots + \mathbf{B}_1\mathbf{u}_{k-\tau-m+1} + \mathbf{B}_0\mathbf{u}_{k-\tau-m}, \quad (2.1)$$

where $\tau \in \mathbb{Z}_+$ is the time delay as a positive integer number of samples, the number of outputs is n_y, the number of inputs is n_u, $\mathbf{y}_i \in \mathbb{R}^{n_y}$, $\mathbf{u}_i \in \mathbb{R}^{n_u}$, $\mathbf{A}_i \in \mathbb{R}^{n_y \times n_y}$, and $\mathbf{B}_i \in \mathbb{R}^{n_y \times n_u}$. Assuming quiescent initial conditions, this corresponds to a discrete-time transfer matrix given by [10]

$$\mathbf{G}(z^{-1}) = z^{-\tau}\mathbf{P}(z^{-1})^{-1}\mathbf{Q}(z^{-1}), \quad (2.2)$$

where

$$\mathbf{P}(z^{-1}) = \mathbf{1} + \mathbf{A}_{n-1}z^{-1} + \cdots + \mathbf{A}_1 z^{-n+1} + \mathbf{A}_0 z^{-n},$$
$$\mathbf{Q}(z^{-1}) = \mathbf{B}_m + \mathbf{B}_{m-1}z^{-1} + \cdots + \mathbf{B}_1 z^{-m+1} + \mathbf{B}_0 z^{-m},$$

and n and m correspond to the orders of the denominator and numerator, respectively.

In the SISO case, (2.1) simplifies to

$$y_k = -a_{n-1}y_{k-1} - \cdots - a_1 y_{k-n+1} - a_0 y_{k-n} + b_m u_{k-\tau} + \cdots + b_1 u_{k-\tau-m+1} + b_0 u_{k-\tau-m}, \quad (2.3)$$

and (2.2) to a transfer function given by

$$G(z^{-1}) = z^{-\tau}\frac{b_m + \cdots + b_1 z^{-m+1} + b_0 z^{-m}}{1 + a_{n-1}z^{-1} + \cdots + a_1 z^{-n+1} + a_0 z^{-n}} = z^{-\tau}\frac{\sum_{r=0}^{m} b_{m-r} z^{-r}}{1 + \sum_{r=1}^{n} a_{n-r} z^{-r}}. \quad (2.4)$$

2.1.1 Discrete-Time to Continuous-Time

Using data to generate system identified models yields discrete-time transfer functions and matrices. These models can be converted to continuous-time in a few different ways. In this thesis, Tustin's method will be used to convert discrete-time systems to continuous time. Note that converting between discrete- and continuous-time utilizes the state-space realization of a system.

A continuous-time linear time-invariant (LTI) state-space model is given by [11]

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{x}_0 = \mathbf{x}(0) \quad (2.5)$$

$$\dot{\mathbf{y}}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t). \quad (2.6)$$

The discrete-time state-space representation is [11]

$$\mathbf{x}_k = \bar{\mathbf{A}}\mathbf{x}_{k-1} + \bar{\mathbf{B}}\mathbf{u}_{k-1}, \quad \mathbf{x}_0 = \mathbf{x}(0) \quad (2.7)$$

$$\mathbf{y}_k = \bar{\mathbf{C}}\mathbf{x}_k + \bar{\mathbf{D}}\mathbf{u}_k. \quad (2.8)$$

2.1.1.1 Tustin's Method

Tustin's method approximates integrals using the trapezoid rule [11], which is given by

$$\int_{t_{k-1}}^{t_k} \mathbf{x}(\tau)\mathrm{d}\tau = \frac{T}{2}\left(\mathbf{x}_k + \mathbf{x}_{k+1}\right),$$

where T is the sample time.

Then the continuous-time state-space matrices are given by

$$\begin{aligned}
\mathbf{A} &= \frac{2}{T}\left(\bar{\mathbf{A}} - \mathbf{1}\right)\left(\bar{\mathbf{A}} + \mathbf{1}\right)^{-1}, \\
\mathbf{B} &= \frac{2}{T}\left(\mathbf{1} - \frac{T}{2}\mathbf{A}\right)\bar{\mathbf{B}}, \\
\mathbf{C} &= \bar{\mathbf{C}}\left(\bar{\mathbf{A}} + \mathbf{1}\right)^{-1}, \\
\mathbf{D} &= \bar{\mathbf{D}} - \mathbf{C}\bar{\mathbf{B}}.
\end{aligned}$$

2.1.2 Continuous-Time to Discrete-Time

When implementing controllers using C code on a real-time system, the continuous-time controllers must first be discretized. For simplicity, a forward Euler method was used in this thesis. Because the sampling frequency of the system is high (2048 Hz), and much higher than the operating frequency of the system (approximately 1 Hz), this method works well.

2.1.2.1 Forward Euler

The forward Euler method approximates a derivative as [11]

$$\dot{\mathbf{x}}(t) = \frac{1}{T}\left(\mathbf{x}_k - \mathbf{x}_{k-1}\right).$$

This results in discrete-time state-space matrices given by

$$\begin{aligned}
\bar{\mathbf{A}} &= \mathbf{1} + T\mathbf{A}, \\
\bar{\mathbf{B}} &= T\mathbf{B}, \\
\bar{\mathbf{C}} &= \mathbf{C}, \\
\bar{\mathbf{D}} &= \mathbf{D}.
\end{aligned}$$

2.2 Linear Least Squares

2.2.1 SISO Least Squares

The equation $\boldsymbol{\Phi\theta} = \boldsymbol{\psi}$, where $\boldsymbol{\Phi}$ and $\boldsymbol{\psi}$ are known from data and $\boldsymbol{\theta}$ contains unknown parameters, can be solved via a least squares approach as in [12]. The solution involves minimizing the cost function given by

$$J(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{\Phi\theta} - \boldsymbol{\psi})^{\top}(\boldsymbol{\Phi\theta} - \boldsymbol{\psi}) = \frac{1}{2}\left\|\boldsymbol{\Phi\theta} - \boldsymbol{\psi}\right\|_2^2. \tag{2.9}$$

Differentiating $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ and setting the solution equal to zero yields

$$\boldsymbol{\Phi}^\top \boldsymbol{\Phi}\boldsymbol{\theta} = \boldsymbol{\Phi}^\top \boldsymbol{\psi}. \tag{2.10}$$

This equation is known as the normal equation and has solution

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{\psi}.$$

If $\boldsymbol{\Phi}$ is full column rank, $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ is positive definite and $\hat{\boldsymbol{\theta}}$ is a unique minimizing solution.

2.2.1.1 QR Factorization Method

To avoid computing $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ and taking its inverse, write $\boldsymbol{\Phi}$ in terms of its QR factorization,

$$\boldsymbol{\Phi} = \mathbf{Q}\mathbf{R},$$

where $\mathbf{Q}$ is column-orthogonal ($\mathbf{Q}^\top \mathbf{Q} = \mathbf{1}$) and $\mathbf{R}$ is upper triangular. Then $\boldsymbol{\Phi}\boldsymbol{\theta} = \boldsymbol{\psi}$ can be written as

$$\mathbf{Q}\mathbf{R}\boldsymbol{\theta} = \boldsymbol{\psi}.$$

Pre-multiplying by $\mathbf{Q}^\top$ yields

$$\begin{aligned} \underbrace{\mathbf{Q}^\top \mathbf{Q}}_{\mathbf{1}} \mathbf{R}\boldsymbol{\theta} &= \mathbf{Q}^\top \boldsymbol{\psi}, \\ \Leftrightarrow \mathbf{R}\boldsymbol{\theta} &= \mathbf{Q}^\top \boldsymbol{\psi}, \end{aligned}$$

which can easily be solved by backward substitution since $\mathbf{R}$ is upper triangular and invertible.

2.2.2 Recursive Least Squares

The recursive least squares method from [12] will be used in system identification to determine the model order. At each iteration, the order of the system is increased by one, keeping the relative degree constant. This adds columns to the data matrix $\boldsymbol{\Phi}$ while keeping the number of rows constant. The recursive algorithm terminates when a stopping criteria is met.

Suppose the least squares problem has been solved for $\hat{\boldsymbol{\theta}}$ and that $\hat{\boldsymbol{\theta}}$ contains an integer number p of parameters. If the number of desired parameters is increased to q, where $q > p$, then the solution of the p-parameter problem can be used to solve the q-parameter problem.

To solve the q-parameter problem, first partition $\boldsymbol{\theta}$ into two parts,

$$\boldsymbol{\theta} = \left[\begin{array}{cccc|ccc} \theta_1 & \theta_2 & \cdots & \theta_p & \theta_{p+1} & \cdots & \theta_q \end{array}\right]^\top = \left[\begin{array}{c} \boldsymbol{\theta}_1 \\ \hline \boldsymbol{\theta}_2 \end{array}\right].$$

The corresponding $\boldsymbol{\Phi}$ matrix is then

$$\boldsymbol{\Phi} = \left[\begin{array}{ccc|ccc} \phi_{11} & \cdots & \phi_{1p} & \phi_{1,p+1} & \cdots & \phi_{1q} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \phi_{r1} & \cdots & \phi_{rp} & \phi_{r,p+1} & \cdots & \phi_{rq} \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{\Phi}_1 & \boldsymbol{\Phi}_2 \end{array}\right],$$

where $\boldsymbol{\Phi}_1 \in \mathbb{R}^{r\times p}$ and $\boldsymbol{\Phi}_2 \in \mathbb{R}^{r\times(q-p)}$. Note that $\boldsymbol{\psi}$ is unchanged.

Then the normal equation (2.18) becomes

$$\left[\begin{array}{cc} \boldsymbol{\Phi}_1^\top\boldsymbol{\Phi}_1 & \boldsymbol{\Phi}_1^\top\boldsymbol{\Phi}_2 \\ \boldsymbol{\Phi}_2^\top\boldsymbol{\Phi}_1 & \boldsymbol{\Phi}_2^\top\boldsymbol{\Phi}_2 \end{array}\right]\left[\begin{array}{c} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \end{array}\right] = \left[\begin{array}{c} \boldsymbol{\Phi}_1^\top\boldsymbol{\psi} \\ \boldsymbol{\Phi}_2^\top\boldsymbol{\psi} \end{array}\right],$$

or equivalently,

$$\begin{aligned} \boldsymbol{\Phi}_1^\top\boldsymbol{\Phi}_1\boldsymbol{\theta}_1 + \boldsymbol{\Phi}_1^\top\boldsymbol{\Phi}_2\boldsymbol{\theta}_2 &= \boldsymbol{\Phi}_1^\top\boldsymbol{\psi}, \\ \boldsymbol{\Phi}_2^\top\boldsymbol{\Phi}_1\boldsymbol{\theta}_1 + \boldsymbol{\Phi}_2^\top\boldsymbol{\Phi}_2\boldsymbol{\theta}_2 &= \boldsymbol{\Phi}_2^\top\boldsymbol{\psi}. \end{aligned}$$

The solution is then given by

$$\hat{\boldsymbol{\theta}}_1 = \hat{\hat{\boldsymbol{\theta}}}_1 - \mathbf{A}\boldsymbol{\Phi}_2^\top(\boldsymbol{\psi} - \boldsymbol{\Phi}_1\hat{\hat{\boldsymbol{\theta}}}_1), \tag{2.11}$$

$$\hat{\boldsymbol{\theta}}_2 = \mathbf{B}\boldsymbol{\Phi}_2^\top(\boldsymbol{\psi} - \boldsymbol{\Phi}_1\hat{\hat{\boldsymbol{\theta}}}_1), \tag{2.12}$$

where

$$\mathbf{A} = (\boldsymbol{\Phi}_1^\top\boldsymbol{\Phi}_1)^{-1}\boldsymbol{\Phi}_1^\top\boldsymbol{\Phi}_2\mathbf{B}, \tag{2.13}$$

$$\mathbf{B} = [\boldsymbol{\Phi}_2^\top\boldsymbol{\Phi}_2 - \boldsymbol{\Phi}_2^\top\boldsymbol{\Phi}_1(\boldsymbol{\Phi}_1^\top\boldsymbol{\Phi}_1)^{-1}\boldsymbol{\Phi}_1^\top\boldsymbol{\Phi}_2]^{-1}, \tag{2.14}$$

$$\hat{\hat{\boldsymbol{\theta}}}_1 = (\boldsymbol{\Phi}_1^\top\boldsymbol{\Phi}_1)^{-1}\boldsymbol{\Phi}_1^\top\boldsymbol{\psi}.$$

On the first pass, $(\boldsymbol{\Phi}_1^\top\boldsymbol{\Phi}_1)^{-1}$ can be recycled from the solution of the p-parameter problem. Then when computing $\mathbf{B}$, only a $(q-p)\times(q-p)$ matrix inversion is required. The matrix

$(\mathbf{\Phi}_1^\top \mathbf{\Phi}_1)^{-1}$ can be updated recursively via

$$(\mathbf{\Phi}^\top \mathbf{\Phi})^{-1} = \begin{bmatrix} \mathbf{\Phi}_1^\top \mathbf{\Phi}_1 & \mathbf{\Phi}_1^\top \mathbf{\Phi}_2 \\ \mathbf{\Phi}_2^\top \mathbf{\Phi}_1 & \mathbf{\Phi}_2^\top \mathbf{\Phi}_2 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{C} + \mathbf{A}\mathbf{\Phi}_2^\top \mathbf{\Phi}_1 \mathbf{C} & -\mathbf{A} \\ -\mathbf{A}^\top & \mathbf{B} \end{bmatrix}, \tag{2.15}$$

where $\mathbf{C} = (\mathbf{\Phi}_1^\top \mathbf{\Phi}_1)^{-1}$. Equation 2.15 can be derived from the Woodbury matrix identity.

2.2.2.1 Stopping Criteria

The number of parameters should stop increasing when the residual $\hat{\mathbf{r}} = \mathbf{\Phi}\hat{\boldsymbol{\theta}} - \boldsymbol{\psi}$ fails to decrease significantly. Instead of using $\hat{\mathbf{r}}$ directly, evaluate the cost function $J(\boldsymbol{\theta})$, which is given by

$$J(\hat{\boldsymbol{\theta}}) = \frac{1}{2}(\mathbf{\Phi}\hat{\boldsymbol{\theta}} - \boldsymbol{\psi})^\top (\mathbf{\Phi}\hat{\boldsymbol{\theta}} - \boldsymbol{\psi}) = \frac{1}{2}\left\|\mathbf{\Phi}\hat{\boldsymbol{\theta}} - \boldsymbol{\psi}\right\|_2^2 = \frac{1}{2}\left\|\hat{\mathbf{r}}\right\|_2^2.$$

In practice, terminate the algorithm when $J(\boldsymbol{\theta})$ stops decreasing significantly, or when n has been increased to a maximum order M. The selection of M also serves to keep the total number of rows constant.

2.2.3 MIMO Least Squares

In this MIMO least squares problem, the cost function to be minimized is

$$J(\mathbf{\Theta}) = \tfrac{1}{2}\left\|\mathbf{\Phi\Theta} - \mathbf{\Psi}\right\|_{\mathsf{F}}^2 = \tfrac{1}{2}\mathrm{tr}\left((\mathbf{\Phi\Theta} - \mathbf{\Psi})^\top (\mathbf{\Phi\Theta} - \mathbf{\Psi})\right), \tag{2.16}$$

where $\|\cdot\|_{\mathsf{F}}$ denotes the Frobenius matrix norm and $\mathrm{tr}(\cdot)$ is the trace of a matrix, which is the sum of entries along the diagonal. Expanding (2.16), obtain

$$J(\mathbf{\Theta}) = \tfrac{1}{2}\mathrm{tr}\left(\mathbf{\Theta}^\top \mathbf{\Phi}^\top \mathbf{\Phi\Theta} - 2\mathbf{\Psi}^\top \mathbf{\Phi\Theta} + \mathbf{\Psi}^\top \mathbf{\Psi}\right). \tag{2.17}$$

Differentiating (2.17) with respect to $\mathbf{\Theta}$ and setting the solution equal to zero yields

$$\mathbf{\Phi}^\top \mathbf{\Phi\Theta} = \mathbf{\Phi}^\top \mathbf{\Psi}. \tag{2.18}$$

Equation (2.18) is known as the normal equation and has solution

$$\hat{\mathbf{\Theta}} = (\mathbf{\Phi}^\top \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top \mathbf{\Psi}.$$

If $\mathbf{\Phi}$ is full column rank, $\mathbf{\Phi}^\top \mathbf{\Phi}$ is positive definite and $\hat{\mathbf{\Theta}}$ is a unique minimizing solution.

2.2.3.1 QR Factorization Method

To avoid computing $\mathbf{\Phi}^\top\mathbf{\Phi}$ and taking the inverse, write $\mathbf{\Phi}$ in terms of its QR factorization,

$$\mathbf{\Phi} = \mathbf{QR},$$

where $\mathbf{Q}$ is column-orthogonal (i.e., $\mathbf{Q}^\top\mathbf{Q} = \mathbf{1}$) and $\mathbf{R}$ is upper triangular. Then $\mathbf{\Phi\Theta} = \mathbf{\Psi}$ can be written

$$\mathbf{QR\Theta} = \mathbf{\Psi}.$$

Partitioning $\mathbf{\Theta}$ and $\mathbf{\Psi}$ into their n_y columns, obtain

$$\begin{aligned} \mathbf{QR}\begin{bmatrix} \boldsymbol{\theta}_1 & \boldsymbol{\theta}_2 & \cdots & \boldsymbol{\theta}_{n_y} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{\psi}_1 & \boldsymbol{\psi}_2 & \cdots & \boldsymbol{\psi}_{n_y} \end{bmatrix} \\ \begin{bmatrix} \mathbf{QR}\boldsymbol{\theta}_1 & \mathbf{QR}\boldsymbol{\theta}_2 & \cdots & \mathbf{QR}\boldsymbol{\theta}_{n_y} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{\psi}_1 & \boldsymbol{\psi}_2 & \cdots & \boldsymbol{\psi}_{n_y} \end{bmatrix}. \end{aligned} \tag{2.19}$$

Equation (2.19) can be solved column-wise by solving

$$\mathbf{QR}\boldsymbol{\theta}_i = \boldsymbol{\psi}_i \tag{2.20}$$

for $i = 1, \ldots, n_y$. Pre-multiplying (2.20) by $\mathbf{Q}^\top$ yields

$$\begin{aligned} \underbrace{\mathbf{Q}^\top\mathbf{Q}}_{\mathbf{1}}\mathbf{R}\boldsymbol{\theta}_i &= \mathbf{Q}^\top\boldsymbol{\psi}_i, \\ \Leftrightarrow \mathbf{R}\boldsymbol{\theta}_i &= \mathbf{Q}^\top\boldsymbol{\psi}_i, \end{aligned}$$

which can easily be solved by backward substitution since $\mathbf{R}$ is upper triangular and invertible.

2.3 Optimization

Optimization is used in the controller synthesis portion of this thesis, which is contained in Part III. In particular, convex optimization subject to LMI constraints is used to synthesize $\mathcal{H}_\infty$ controllers. In Chapter 5, however, the optimization problem is non-convex and thus iteration and bisection method are used to solve the problem locally, or sub-optimally.

2.3.1 Convex Sets

Definition 2.1 (Convexity [13])**.** A set, $\mathcal{S}$, in a real inner product space is convex if for all $\mathbf{x}, \mathbf{y} \in \mathcal{S}$ and $\alpha \in [0,1]$, $\alpha\mathbf{x} + (1-\alpha)\mathbf{y}$. A function, $f : \mathcal{S} \to \mathbb{R}$, is strictly convex if for all $\mathbf{x}, \mathbf{y} \in \mathcal{S}$, $\alpha \in (0,1)$, and $\mathbf{x} \neq \mathbf{y}$, $f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) < \alpha f(\mathbf{x}) + (1-\alpha)f(\mathbf{y})$.

Proposition 2.2 (Uniqueness of a Convex Function Minimizer [13])**.** *Suppose that $f : \mathcal{S} \to \mathbb{R}$ is strictly convex and continuous. If $\mathcal{S} \subset \mathbb{R}^n$ is closed, bounded, and convex, then a unique minimizer of f exists in $\mathcal{S}$.*

2.3.2 Linear Matrix Inequalities

Definition 2.3 (Matrix Inequality [14])**.** A matrix inequality (MI), $\mathbf{G} : \mathbb{R}^m \to \mathbb{R}^{n\times n}$, in the variable $\mathbf{x} \in \mathbb{R}^m$ is an expression of the form

$$\mathbf{G}(\mathbf{x}) = \mathbf{G}_0 + \sum_{i=1}^{p} f_i(\mathbf{x})\mathbf{G}_i \leq 0,$$

where $\mathbf{x}^\top = [x_i \ \dots \ x_m]$, $\mathbf{G}_i \in \mathbb{R}^{n\times n}$, $i = 0, \dots, p$.

Definition 2.4 (Bilinear Matrix Inequality [14])**.** A bilinear matrix inequality (BMI), $\mathbf{H} : \mathbb{R}^m \to \mathbb{R}^{n\times n}$, in the variable $\mathbf{x} \in \mathbb{R}^m$ is an expression of the form

$$\mathbf{H}(\mathbf{x}) = \mathbf{H}_0 + \sum_{i=1}^{m} x_i\mathbf{H}_i + \sum_{i=1}^{m}\sum_{j=1}^{m} x_i x_j \mathbf{H}_{i,j} \leq 0,$$

where $\mathbf{x}^\top = [x_i \ \dots \ x_m]$, $\mathbf{H}_i, \mathbf{H}_{i,j} \in \mathbb{R}^{n\times n}$, $i = 0, \dots, m$, $j = 0, \dots, m$.

Definition 2.5 (Linear Matrix Inequality [14])**.** A linear matrix inequality (LMI), $\mathbf{F} : \mathbb{R}^m \to \mathbb{R}^{n\times n}$, in the variable $\mathbf{x} \in \mathbb{R}^m$ is an expression of the form

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}_0 + \sum_{i=1}^{m} x_i\mathbf{F}_i \leq 0, \tag{2.21}$$

where $\mathbf{x}^\top = [x_i \ \dots \ x_m]$, $\mathbf{F}_i \in \mathbb{R}^{n\times n}$, $i = 0, \dots, m$.

Example 2.6 (Matrix Form of an LMI)**.** Consider the expression $\mathbf{PA} + \mathbf{A}^\top\mathbf{P} + \mathbf{Q} < 0$, where $\mathbf{A}, \mathbf{P}, \mathbf{Q} \in \mathbb{R}^{n\times n}$, $\mathbf{Q} > 0$, and $\mathbf{P}$ is the design variable. Although this does not appear to have the same form as (2.21), it is indeed an LMI, as shown in [14]. Instead of using the scalar form (2.21), the LMIs in this thesis will be written in matrix form.

2.4 Linear Systems Theory

2.4.1 BIBO Stability

Definition 2.7 (BIBO Stability [15])**.** A system is bounded-input-bounded-output (BIBO) stable if for any bounded input the corresponding output is bounded. A closed-loop system is internally BIBO stable if all internal signals are bounded provided all external signals are bounded.

Theorem 2.8. *A system with transfer function $G(s)$ is BIBO stable if and only if $G(s)$ is proper and all its poles are in the open-left-half-plane (OLHP) [15].*

2.4.2 The $\mathcal{H}_\infty$ Norm

The $\mathcal{H}_\infty$ norm of an LTI system, which is the basis of $\mathcal{H}_\infty$ control, is defined. The $\mathcal{H}_\infty$ norm of the closed-loop system is the objective function when performing $\mathcal{H}_\infty$-optimal controller synthesis.

Definition 2.9 ($\mathcal{H}_\infty$ Norm of an LTI System [14])**.** Consider $\mathbf{y}(t) = (\boldsymbol{\mathcal{G}}\mathbf{u})(t)$ where $\mathbf{y}(s) = \mathbf{G}(s)\mathbf{u}(s)$, $\mathbf{G}(s) = \mathbf{C}(s\mathbf{1} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$, and $\mathbf{A}$ is Hurwitz, meaning all its eigenvalues lie in the open-left-half-plane (OLHP). The $\mathcal{H}_\infty$ norm is

$$\|\boldsymbol{\mathcal{G}}\|_\infty = \sup_{\omega \in \mathbb{R}} \bar{\sigma}(\mathbf{G}(j\omega)), \tag{2.22}$$

where $\bar{\sigma}(\mathbf{G}(j\omega)) = \sqrt{\bar{\lambda}(\mathbf{G}^{\mathsf{H}}(j\omega)\mathbf{G}(j\omega))}$, and $\bar{\lambda}(\cdot)$ denotes the largest eigenvalue of a matrix.

Lemma 2.10 (Bounded Real Lemma [14])**.** *Consider $\mathbf{G}(s) = \mathbf{C}(s\mathbf{1} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$ where $\mathbf{A}$ is Hurwitz, $(\mathbf{A}, \mathbf{B})$ is controllable, and $(\mathbf{A}, \mathbf{C})$ is observable. The following conditions are equivalent.*

1. $\|\boldsymbol{\mathcal{G}}\|_\infty < \gamma$.
2. *There exists $\bar{\mathbf{P}} = \bar{\mathbf{P}}^\top > 0$ such that*

$$\mathbf{F}_1 = \begin{bmatrix} \bar{\mathbf{P}}\mathbf{A} + \mathbf{A}^\top\bar{\mathbf{P}} + \mathbf{C}^\top\mathbf{C} & \bar{\mathbf{P}}\mathbf{B} + \mathbf{C}^\top\mathbf{D} \\ \star & \mathbf{D}^\top\mathbf{D} - \gamma^2\mathbf{1} \end{bmatrix} < 0\,. \tag{2.23}$$

3. There exists $\mathbf{P} = \mathbf{P}^\top > 0$ *such that*

$$\mathbf{F}_2 = \begin{bmatrix} \mathbf{PA} + \mathbf{A}^\top\mathbf{P} & \mathbf{PB} & \mathbf{C}^\top \\ \star & -\gamma\mathbf{1} & \mathbf{D}^\top \\ \star & \star & -\gamma\mathbf{1} \end{bmatrix} < 0\,. \tag{2.24}$$

4. There exists $\mathbf{Q} = \mathbf{Q}^\top > 0$ *such that*

$$\mathbf{F}_3 = \begin{bmatrix} \mathbf{AQ} + \mathbf{QA}^\top & \mathbf{B} & \mathbf{QC}^\top \\ \star & -\gamma\mathbf{1} & \mathbf{D}^\top \\ \star & \star & -\gamma\mathbf{1} \end{bmatrix} < 0\,. \tag{2.25}$$

2.4.3 The Generalized Plant

The generalized LTI plant has a minimal state-space realization given by [14]

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}_1\mathbf{w}(t) + \mathbf{B}_2\mathbf{u}(t)\,, \tag{2.26}$$

$$\mathbf{z}(t) = \mathbf{C}_1\mathbf{x}(t) + \mathbf{D}_{11}\mathbf{w}(t) + \mathbf{D}_{12}\mathbf{u}(t)\,, \tag{2.27}$$

$$\mathbf{y}(t) = \mathbf{C}_2\mathbf{x}(t) + \mathbf{D}_{21}\mathbf{w}(t) + \mathbf{D}_{22}\mathbf{u}(t)\,, \tag{2.28}$$

where $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ is the system state, $\mathbf{z}(t) \in \mathbb{R}^{n_z}$ is the performance signal, $\mathbf{y}(t) \in \mathbb{R}^{n_y}$ is the measurement signal (also the input signal to the controller), $\mathbf{w}(t) \in \mathbb{R}^{n_w}$ is the exogenous signal, $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ is the control signal, and the state-space matrices are of appropriate dimension.

Typically, the exogenous input includes the reference, disturbances, and noise, and is given by

$$\mathbf{w}(t) = \begin{bmatrix} \mathbf{r}(t) \\ \mathbf{d}(t) \\ \mathbf{n}(t) \end{bmatrix}.$$

Using the configuration shown in Fig. 2.1, the performance channels include weighted noise-free tracking error and weighted control effort, where $\mathbf{z}_1(s) = \mathbf{W}_e(s)\mathbf{e}(s)$ and $\mathbf{e}(s) = \mathbf{r}(s) - \mathbf{y}_p(s)$, and $\mathbf{z}_2(s) = \mathbf{W}_u(s)\mathbf{u}(s)$. In addition, there are weighting filters on disturbances and noise. The plant has a minimal state-space realization $(\mathbf{A}_o, \mathbf{B}_o, \mathbf{C}_o, \mathbf{D}_o)$ and the weighting transfer matrices $\mathbf{W}_e(s)$, $\mathbf{W}_u(s)$, $\mathbf{W}_d(s)$, and $\mathbf{W}_n(s)$ have minimal state-space realizations given by $(\mathbf{A}_e, \mathbf{B}_e, \mathbf{C}_e, \mathbf{D}_e)$, $(\mathbf{A}_u, \mathbf{B}_u, \mathbf{C}_u, \mathbf{D}_u)$, $(\mathbf{A}_d, \mathbf{B}_d, \mathbf{C}_d, \mathbf{D}_d)$, and $(\mathbf{A}_n, \mathbf{B}_n, \mathbf{C}_n, \mathbf{D}_n)$. Note that the reference will not be weighted because the MTS controller already applies a smoothing interpolation to the reference signal.

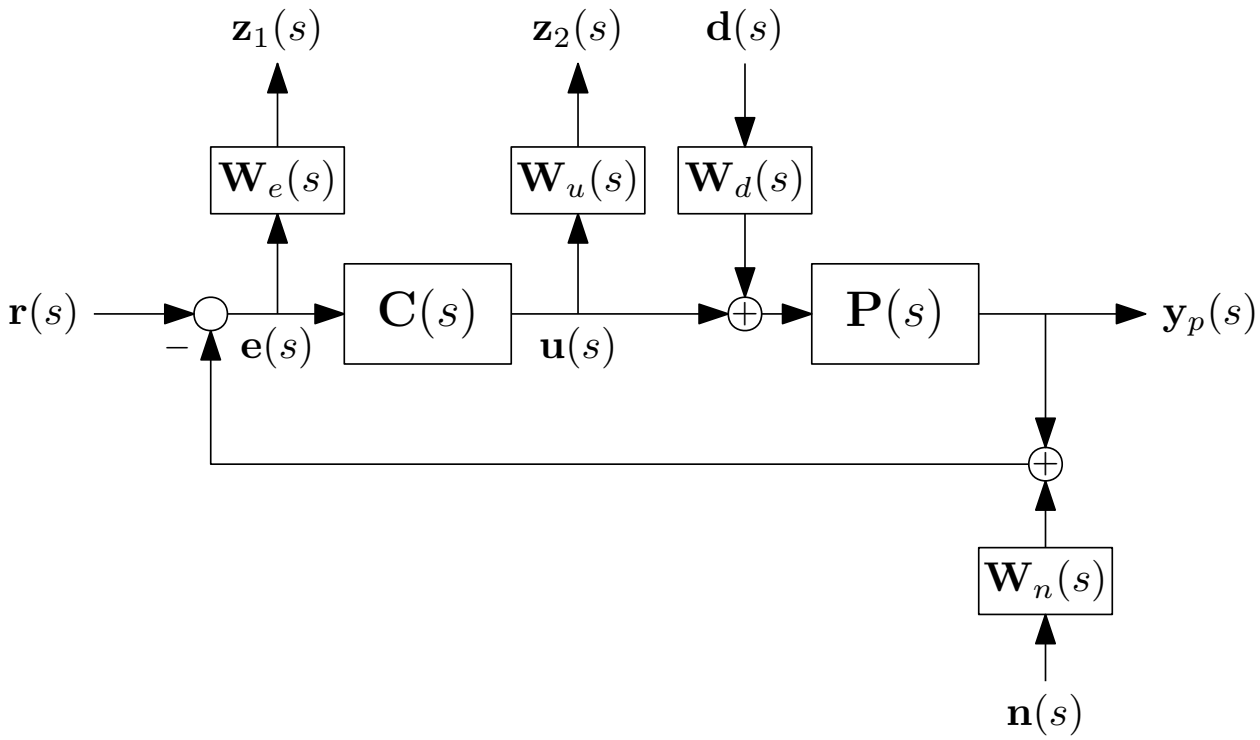


Figure 2.1: Block diagram for controller synthesis.

2.4.3.1 Integral Control

To design a controller that includes an integrator, the input to the controller $\mathbf{y}(t)$ must include the tracking error augmented with the integral of tracking error. Therefore, it is necessary to define an "integral state." The time rate of change of this state is set equal to the tracking error, such that the state itself is the integral of the error [16, 17]. Taking into account noise and disturbance, the state $\mathbf{x}_i(t)$ is defined by

$$\begin{aligned}\dot{\mathbf{x}}_i(t) &= \mathbf{r}(t) - (\mathbf{y}_o(t) + \mathbf{C}_n\mathbf{x}_n(t) + \mathbf{D}_n\mathbf{n}(t)) \\ &= \mathbf{r}(t) - (\mathbf{C}_o\mathbf{x}_o(t) + \mathbf{D}_o\left(\mathbf{u}(t) + \mathbf{C}_d\mathbf{x}_d(t) + \mathbf{D}_d\mathbf{d}(t)\right) + \mathbf{C}_n\mathbf{x}_n(t) + \mathbf{D}_n\mathbf{n}(t)) \\ &= -\mathbf{C}_o\mathbf{x}_o(t) - \mathbf{D}_o\mathbf{C}_d\mathbf{x}_d(t) - \mathbf{C}_n\mathbf{x}_n(t) + \mathbf{r}(t) - \mathbf{D}_o\mathbf{D}_d\mathbf{d}(t) - \mathbf{D}_n\mathbf{n}(t) - \mathbf{D}_o\mathbf{u}(t). \qquad (2.29)\end{aligned}$$

2.4.3.2 1DOF Control

The one degree-of-freedom (1DOF) controller structure is shown in Fig. 2.2. The input to the controller $\mathbf{y}(t)$ is the error, $\mathbf{e}(t)$.

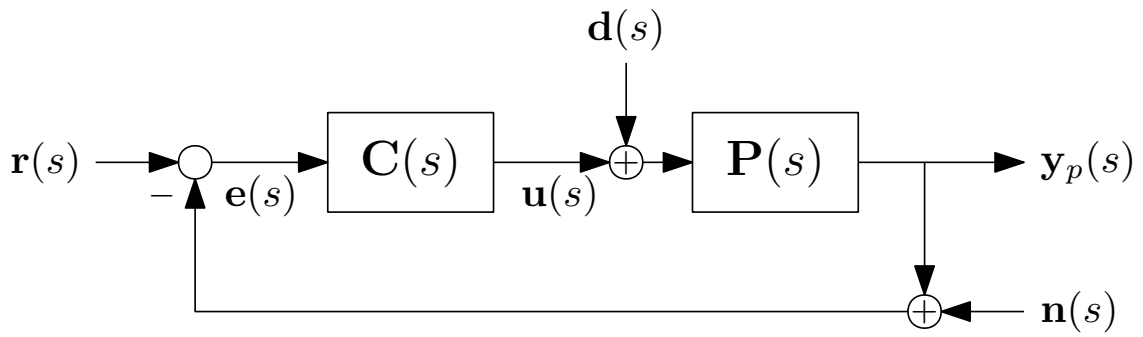


Figure 2.2: 1DOF controller structure.

The generalized plant is then given by

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} \mathbf{A}_o & \mathbf{0} & \mathbf{0} & \mathbf{B}_o\mathbf{C}_d & \mathbf{0} \\ -\mathbf{B}_e\mathbf{C}_o & \mathbf{A}_e & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_u & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_n \end{bmatrix}}_{\mathbf{A}} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{B}_o\mathbf{D}_d & \mathbf{0} \\ \mathbf{B}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_n \end{bmatrix}}_{\mathbf{B}_1} \mathbf{w}(t) + \underbrace{\begin{bmatrix} \mathbf{B}_o \\ -\mathbf{B}_e\mathbf{D}_o \\ \mathbf{B}_u \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{B}_2} \mathbf{u}(t)\,, \tag{2.30}$$

$$\mathbf{z}(t) = \underbrace{\begin{bmatrix} -\mathbf{D}_e\mathbf{C}_o & \mathbf{C}_e & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}_u & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{C}_1} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{D}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{D}_{11}} \mathbf{w}(t) + \underbrace{\begin{bmatrix} -\mathbf{D}_e\mathbf{D}_o \\ \mathbf{D}_u \end{bmatrix}}_{\mathbf{D}_{12}} \mathbf{u}(t)\,, \tag{2.31}$$

$$\mathbf{y}(t) = \underbrace{\begin{bmatrix} -\mathbf{C}_o & \mathbf{0} & \mathbf{0} & -\mathbf{D}_o\mathbf{C}_d & -\mathbf{C}_n \end{bmatrix}}_{\mathbf{C}_2} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{1} & -\mathbf{D}_o\mathbf{D}_d & -\mathbf{D}_n \end{bmatrix}}_{\mathbf{D}_{21}} \mathbf{w}(t) + \underbrace{\begin{bmatrix} -\mathbf{D}_o \end{bmatrix}}_{\mathbf{D}_{22}} \mathbf{u}(t)\,, \tag{2.32}$$

where $\mathbf{x}(t) = \begin{bmatrix} \mathbf{x}_o^\top(t) & \mathbf{x}_e^\top(t) & \mathbf{x}_u^\top(t) & \mathbf{x}_d^\top(t) & \mathbf{x}_n^\top(t) \end{bmatrix}^\top$ includes the plant states and filter states.

2.4.3.3 1DOF Control with Integrator

For a 1DOF controller that includes an integrator, the controller input $\mathbf{y}(t)$ is given by $\mathbf{y}(t) = \begin{bmatrix} \mathbf{e}^\top(t) & \mathbf{x}_i^\top(t) \end{bmatrix}^\top$, where $\mathbf{x}_i(t)$ is defined in (2.29).

The generalized plant used for controller synthesis is then given by

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} \mathbf{A}_o & \mathbf{0} & \mathbf{0} & \mathbf{B}_o\mathbf{C}_d & \mathbf{0} & \mathbf{0} \\ -\mathbf{B}_e\mathbf{C}_o & \mathbf{A}_e & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_u & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_d & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_n & \mathbf{0} \\ -\mathbf{C}_o & \mathbf{0} & \mathbf{0} & -\mathbf{D}_o\mathbf{C}_d & -\mathbf{C}_n & \mathbf{0} \end{bmatrix}}_{\mathbf{A}} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{B}_o\mathbf{D}_d & \mathbf{0} \\ \mathbf{B}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_n \\ \mathbf{1} & -\mathbf{D}_o\mathbf{D}_d & -\mathbf{D}_n \end{bmatrix}}_{\mathbf{B}_1} \mathbf{w}(t) + \underbrace{\begin{bmatrix} \mathbf{B}_o \\ -\mathbf{B}_e\mathbf{D}_o \\ \mathbf{B}_u \\ \mathbf{0} \\ \mathbf{0} \\ -\mathbf{D}_o \end{bmatrix}}_{\mathbf{B}_2} \mathbf{u}(t)\,, \tag{2.33}$$

$$\mathbf{z}(t) = \underbrace{\begin{bmatrix} -\mathbf{D}_e\mathbf{C}_o & \mathbf{C}_e & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}_u & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{C}_1} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{D}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{D}_{11}} \mathbf{w}(t) + \underbrace{\begin{bmatrix} -\mathbf{D}_e\mathbf{D}_o \\ \mathbf{D}_u \end{bmatrix}}_{\mathbf{D}_{12}} \mathbf{u}(t)\,, \tag{2.34}$$

$$\mathbf{y}(t) = \underbrace{\begin{bmatrix} -\mathbf{C}_o & \mathbf{0} & \mathbf{0} & -\mathbf{D}_o\mathbf{C}_d & -\mathbf{C}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}}_{\mathbf{C}_2} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{1} & -\mathbf{D}_o\mathbf{D}_d & -\mathbf{D}_n \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{D}_{21}} \mathbf{w}(t) + \underbrace{\begin{bmatrix} -\mathbf{D}_o \\ \mathbf{0} \end{bmatrix}}_{\mathbf{D}_{22}} \mathbf{u}(t)\,, \tag{2.35}$$

where $\mathbf{x}(t) = \begin{bmatrix} \mathbf{x}_o^\top(t) & \mathbf{x}_e^\top(t) & \mathbf{x}_u^\top(t) & \mathbf{x}_d^\top(t) & \mathbf{x}_n^\top(t) & \mathbf{x}_i^\top(t) \end{bmatrix}^\top$.

2.4.3.4 2DOF Control

In a two degree-of-freedom (2DOF) controller, the controller is based on the feedback as well as the reference signal. The general 2DOF controller structure is shown in Fig. 2.3.

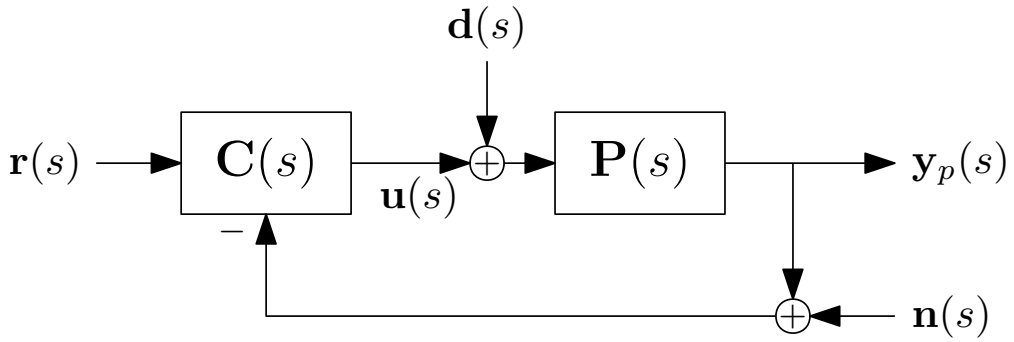


Figure 2.3: General 2DOF controller structure.

Figure 2.4 shows the controller structure in the case where the 2DOF controller includes feedback and feedforward control. For alternative 2DOF control structures, see [15].

If the controller to be designed is a 2DOF controller without integrator, then the input

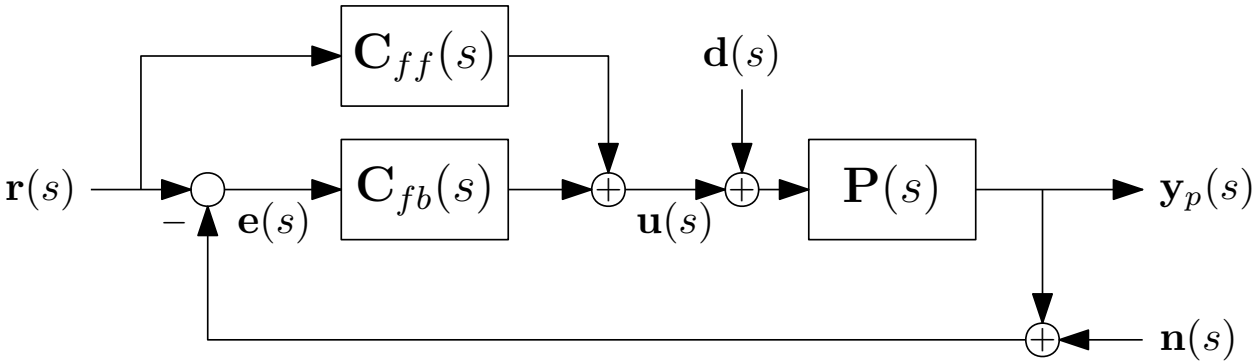


Figure 2.4: Feedback plus feedforward 2DOF controller structure.

to the controller $\mathbf{y}(t)$ is defined as

$$\mathbf{y}(t) = \begin{bmatrix} \mathbf{r}(t) \\ \mathbf{e}(t) \end{bmatrix}.$$

The generalized plant is then given by

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} \mathbf{A}_o & \mathbf{0} & \mathbf{0} & \mathbf{B}_o\mathbf{C}_d & \mathbf{0} \\ -\mathbf{B}_e\mathbf{C}_o & \mathbf{A}_e & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_u & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_n \end{bmatrix}}_{\mathbf{A}} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{B}_o\mathbf{D}_d & \mathbf{0} \\ \mathbf{B}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_n \end{bmatrix}}_{\mathbf{B}_1} \mathbf{w}(t) + \underbrace{\begin{bmatrix} \mathbf{B}_o \\ -\mathbf{B}_e\mathbf{D}_o \\ \mathbf{B}_u \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{B}_2} \mathbf{u}(t),$$

$$\mathbf{z}(t) = \underbrace{\begin{bmatrix} -\mathbf{D}_e\mathbf{C}_o & \mathbf{C}_e & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}_u & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{C}_1} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{D}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{D}_{11}} \mathbf{w}(t) + \underbrace{\begin{bmatrix} -\mathbf{D}_e\mathbf{D}_o \\ \mathbf{D}_u \end{bmatrix}}_{\mathbf{D}_{12}} \mathbf{u}(t),$$

$$\mathbf{y}(t) = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{C}_o & \mathbf{0} & \mathbf{0} & -\mathbf{D}_o\mathbf{C}_d & -\mathbf{C}_n \end{bmatrix}}_{\mathbf{C}_2} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & -\mathbf{D}_o\mathbf{D}_d & -\mathbf{D}_n \end{bmatrix}}_{\mathbf{D}_{21}} \mathbf{w}(t) + \underbrace{\begin{bmatrix} \mathbf{0} \\ -\mathbf{D}_o \end{bmatrix}}_{\mathbf{D}_{22}} \mathbf{u}(t).$$

2.4.3.5 2DOF Control with Integrator

For a 2DOF controller that includes an integrator, the controller input $\mathbf{y}(t)$ is given by

$$\mathbf{y}(t) = \begin{bmatrix} \mathbf{r}(t) \\ \mathbf{e}(t) \\ \mathbf{x}_i(t) \end{bmatrix}.$$

The generalized plant used for controller synthesis is then given by

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} \mathbf{A}_o & \mathbf{0} & \mathbf{0} & \mathbf{B}_o\mathbf{C}_d & \mathbf{0} & \mathbf{0} \\ -\mathbf{B}_e\mathbf{C}_o & \mathbf{A}_e & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_u & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_d & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_n & \mathbf{0} \\ -\mathbf{C}_o & \mathbf{0} & \mathbf{0} & -\mathbf{D}_o\mathbf{C}_d & -\mathbf{C}_n & \mathbf{0} \end{bmatrix}}_{\mathbf{A}} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{B}_o\mathbf{D}_d & \mathbf{0} \\ \mathbf{B}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_n \\ \mathbf{1} & -\mathbf{D}_o\mathbf{D}_d & -\mathbf{D}_n \end{bmatrix}}_{\mathbf{B}_1} \mathbf{w}(t) + \underbrace{\begin{bmatrix} \mathbf{B}_o \\ -\mathbf{B}_e\mathbf{D}_o \\ \mathbf{B}_u \\ \mathbf{0} \\ \mathbf{0} \\ -\mathbf{D}_o \end{bmatrix}}_{\mathbf{B}_2} \mathbf{u}(t)\,,$$

$$\mathbf{z}(t) = \underbrace{\begin{bmatrix} -\mathbf{D}_e\mathbf{C}_o & \mathbf{C}_e & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}_u & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{C}_1} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{D}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{D}_{11}} \mathbf{w}(t) + \underbrace{\begin{bmatrix} -\mathbf{D}_e\mathbf{D}_o \\ \mathbf{D}_u \end{bmatrix}}_{\mathbf{D}_{12}} \mathbf{u}(t)\,,$$

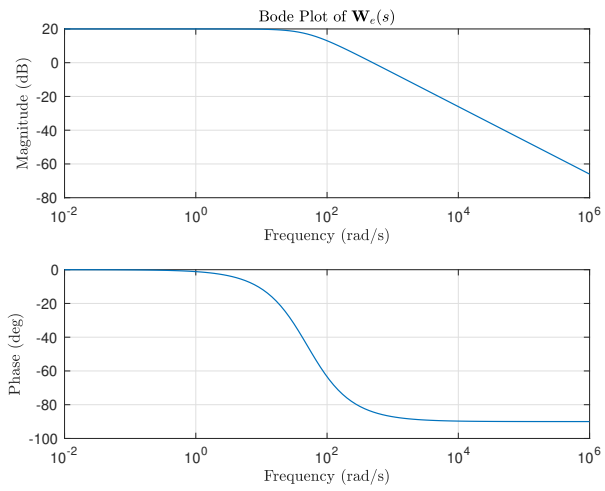
$$\mathbf{y}(t) = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{C}_o & \mathbf{0} & \mathbf{0} & -\mathbf{D}_o\mathbf{C}_d & -\mathbf{C}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}}_{\mathbf{C}_2} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & -\mathbf{D}_o\mathbf{D}_d & -\mathbf{D}_n \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{D}_{21}} \mathbf{w}(t) + \underbrace{\begin{bmatrix} \mathbf{0} \\ -\mathbf{D}_o \\ \mathbf{0} \end{bmatrix}}_{\mathbf{D}_{22}} \mathbf{u}(t)\,.$$

2.4.4 Tuning Weights

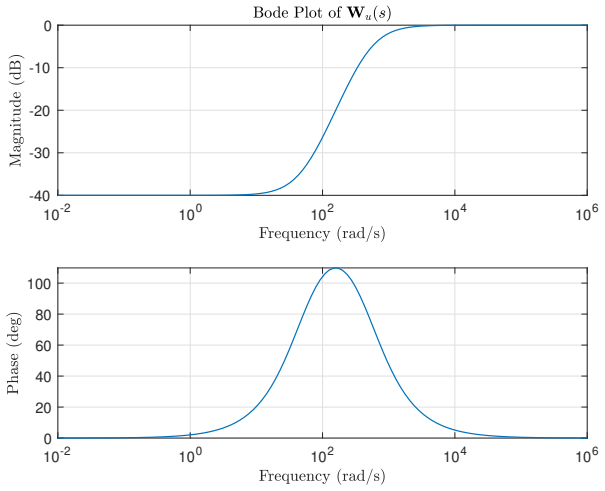
For the performance channels, the weights should be tuned to prioritize minimizing the error in the operating bandwidth and penalizing control effort outside the operating bandwidth. For disturbances and noise, the weights should be tuned to filter broadband white noise into signals that mimic the disturbance and noise properties of the system. Therefore, $\mathbf{W}_e(s)$ and $\mathbf{W}_d(s)$ are designed as low-pass filters, and $\mathbf{W}_u(s)$ and $\mathbf{W}_n(s)$ are chosen to be lead compensators. Figure 2.5 shows a typical set of weights used to synthesize the controllers presented in Chapter 7.

2.5 Implementing Alternative Controllers

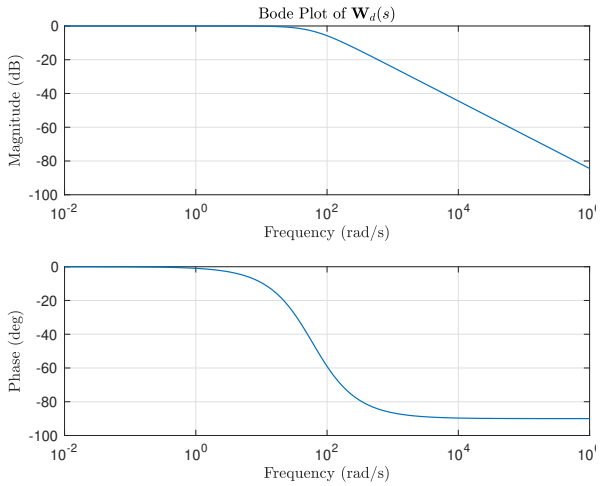
There are a few practical considerations that must be addressed when implementing alternative controller schemes on the MTS system in real-time. Typically, the controller must be reduce-ordered, then used in code generation. Afterwards, the control signal should be simulated in real-time to ensure that it is safe to inject. These practices will be used in Chapters 6 and 7.



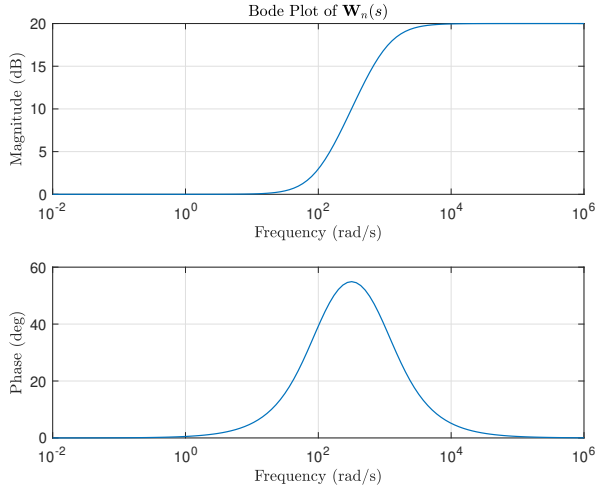
(a) Error.



(b) Control effort.



(c) Disturbances.



(d) Noise.

Figure 2.5: Weighting function frequency responses.

2.5.1 Order Reduction

In general, because the MTS controller is limited computationally, controllers higher than 6^{th}-order cannot be implemented. Order reduction can be performed by manually eliminating higher modes, or using the `balred` command in `MATLAB`. Details on the order reduction process for feedforward and $\mathcal{H}_\infty$ controllers are given in Chapters 6 and 7.

2.5.2 Code Generation

In order to implement alternative controllers on the MTS system, the controller must be implemented via calculated channels. The channels themselves require code to be written in a language similar to C, and the controller state-space matrices must be discretized in

real-time based on the sampling frequency of the MTS system. It was initially thought that the sampling frequency changed dynamically, but upon further investigation it became clear that the sampling frequency of the MTS controller used to perform experiments for this thesis is always 2048 Hz. Code generation templates were created using `MATLAB`, with the discretization method being forward Euler. A code generation template and sample MTS code for a 2DOF $\mathcal{H}_\infty$ controller are included in Appendix B. Because MTS calculated channels can only output scalar values, controller code for each actuator must be implemented in a separate calculated channel.

2.5.3 Signal Injection

Even if a control signal is well-behaved in simulation, it may display undesirable behaviour in practice. To ensure the control signal from an alternative controller does not exhibit saturation, instability, or oscillatory behaviour, the signals should be simulated in real-time on the MTS system before being injected. The presence of an $\mathcal{H}_\infty$ or feedforward controller "gain" that, for all intents and purposes, acts as a switch, allows the signal to be injected not at all, in full, or somewhere in between. Initially, the controller code should be implemented in calculated channels with the controller switch gain set to zero. Depending on how the signal behaves, it may then be injected.

2.5.3.1 Feedforward Control

Because feedforward controllers are based on a smooth, noise-free reference signal, the feedforward control signal should also be smooth. In addition, the injected feedforward signal will be identical to the simulated signal provided the reference is the same. If the signal looks clean and does not saturate, the feedforward control signal can be injected by setting the controller switch gain to one.

2.5.3.2 $\mathcal{H}_\infty$ Control

Unlike inversion-based feedforward controllers, where the input is a smooth, noise-free reference signal, $\mathcal{H}_\infty$ controllers act on the noisy error signal. It is not uncommon for an $\mathcal{H}_\infty$ controller that works well in simulation to exhibit poor behaviour when implemented on the test rig. This is especially true because the controller synthesis method presented in Chapter 7 does not enforce stability of the controller itself—it only requires the closed-loop to be BIBO stable. Because the $\mathcal{H}_\infty$ control signal is based on the error, the signal will change based on the degree to which it is injected. For this reason, it is recommended that the $\mathcal{H}_\infty$ control signal be injected gradually in function generation mode. This is done by increasing

the controller switch gain by increments of 0.1, for example, and observing how the signal responds before injecting further. If the signal has been fully injected and is well-behaved, it can be used to run tests.

2.6 Testing and Post-Processing

The performance of alternative controllers will be evaluated in terms of speed and error. An MTS controller feature called PSO will be employed to determine what speeds can be achieved by a particular controller. Afterwards, the tracking performance will be quantified using RMS error. In this thesis, alternative controller performance will always be compared to the standard controller, which is a tuned PI controller with cross-coupling compensation, as described in Section 3.3.3.

2.6.1 Profile Segment Optimization (PSO)

To assess the potential improvement offered by a particular controller scheme, an MTS controller feature called Profile Segment Optimization (PSO) can be used. A load profile, described in Section 3.3.1, contains a sequence of loads and corresponding transition times. PSO is a feature that automatically shortens transition times by a specified step-size as long as the error in the transition stays within a specified error bound, which is explained in Section 3.3.2. If the error during the transition is outside this limit, the transition time is increased. If a test is run for multiple passes, PSO will shorten the load lines down to a minimum possible transition time, which is specified by the user.

In this thesis, controllers are compared by activating PSO and running the same load profile for multiple passes. The number of passes should be enough that the transition times can, in theory, be reduced to the minimum possible transition time. Typically, the initial transition times are set to 1 second. After the final pass, PSO is turned off and the controller is used to run the load profile once more. The data is recorded and analyzed in terms of runtime and error.

2.6.2 Data Collection and Sampling Frequency

The sampling frequency of the MTS system when recording "continuous" data is 128 Hz. Tests typically run at approximately 1 Hz. Although this sampling frequency is unnecessarily high, "continuous" data can be collected then resampled afterwards. This can be done using the `interp1` command in `MATLAB`, for example. When data is being collected for system identification, a good "rule of thumb" is to use a sampling frequency that is approximately ten times the bandwidth of the system [18].

2.6.3 RMS Error

The improvement in performance offered by a particular controller is quantified by the root-mean-square (RMS) error between the reference and the output. RMS error is given by

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(r_i - y_i)^2}, \tag{2.36}$$

where r_i are the values of the reference signal and y_i are the outputs of the system. RMS error has the same units as the reference signal and system output. This metric can be applied to both simulated and experimental data. For a MIMO system, the RMS error is computed for each column of data.

Part II

Modelling and System Identification

Chapter 3

Modelling and Cycle Estimation

In this chapter, a fatigue test system is broken up into its key components and an analytical model is developed for each distinct part. First, each component of the plant is modelled, then these are combined to give the overall plant model. Next, the built-in MTS controller is modelled. Finally, the plant and controller models are combined in simulation to provide an estimate of cycle time. The closed-loop system is shown in Fig. 3.1.

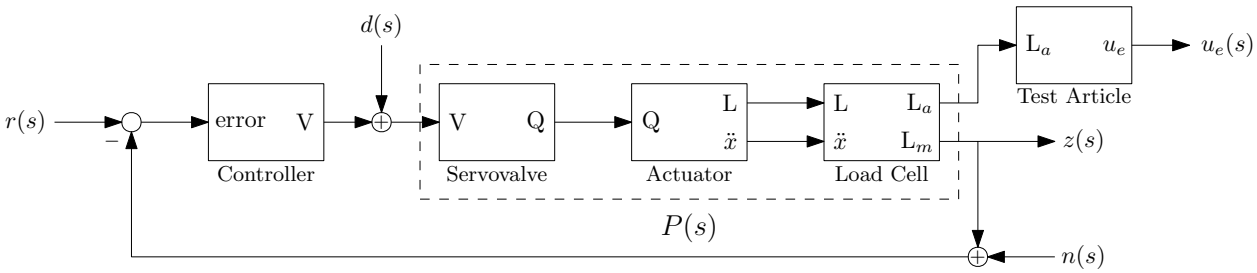


Figure 3.1: Block diagram of closed-loop system.

The key components of the fatigue test system are the controller, servovalve, actuator, load cell, and test article. The system is set up in load control, where the reference signal is a desired force. The controller takes the error between the reference and feedback and outputs a voltage. The voltage is sent to the servovalve and results in a flow rate, which causes movement in the actuator along with a force output. The load cell transfers the load to the test article and provides a force measurement that is fed back to the controller. Depending on which sensors are installed, test article deflection may also be recorded.

The particular fatigue test system modelled in this thesis is the SHM platform at the National Research Council of Canada, pictured in Fig. 1.1. However, servovalve and actuator size, load cell parameters, and article stiffness can be modified to theoretically model any fatigue test rig that uses MTS equipment.

3.1 Plant Model

Despite the fact that real systems are nonlinear, for the purposes of this thesis each system is approximated as linear. This simplifies analysis and simulation and allows the use of linear controller design techniques. Additionally, because fatigue test rigs are configured in load control, it is possible to formulate the problem such that the test article is not "in the loop." Although stiffness information from the test article is incorporated into the actuator model, this approach doesn't require a complete numerical model of the article. Therefore the plant, $P(s)$ in Fig. 3.1, is comprised of the servovalve, actuator, and load cell.

3.1.1 Servovalve

Initially, a high-fidelity nonlinear two-stage servovalve model such as that presented in [19] was investigated. However, such a model required measurements of several internal physical parameters, in addition to continuous measurements of changing pressures and flow rates. This approach was abandoned for two reasons. First, it is not feasible to disassemble every valve to obtain parameter values, and second, the current system does not have sensors in place to record internal pressures or flow rates. Next, a two-stage servovalve model with an internal feedback mechanism like that of [4] and [6] was created. This model was much less complicated than the previous one, but still required unknown parameter values.

Finally, identifying a transfer function from the frequency response given in the MTS catalogue [8] provided a linear model defined in terms of given values. The catalogue gives the frequency response of various servovalve sizes at the rated input current. Depending on the size, the valve transfer function can be approximated as either a first-order or second-order system. The transfer function can be identified based only on maximum flow rate, rated current, cutoff frequency, and, for the second-order systems, damping ratio. The first-order transfer function is given by

$$G_v(s) = \frac{Q(s)}{I(s)} = \frac{K_v \omega_c}{s + \omega_c}, \tag{3.1}$$

where $Q(t)$ is the flow rate with units of litres per second and $I(t)$ is the current with units of Amperes. $K_v = Q_{max}/I_{rated}$ is the DC gain, and ω_c is the cutoff frequency.

The second-order transfer function is given by

$$G_v(s) = \frac{Q(s)}{I(s)} = \frac{K_v \omega_c^2}{s^2 + 2\zeta\omega_c s + \omega_c^2}, \tag{3.2}$$

where ζ was found by comparison with the catalogue frequency response.

3.1.1.1 Units

Although the controller physically applies a current to the servovalve, the MTS interface displays control effort as a voltage. It was confirmed by MTS that the current and voltage are related through a scaling factor that is simply the ratio of the maximum, or fullscale, values, where the fullscale current and voltage are given by I_m and V_m. Because the MTS control law is configured to produce a voltage, the control signal must be multiplied by a factor of I_m/V_m before being passed to $G_v(s)$. The block diagram is shown in Fig. 3.2.

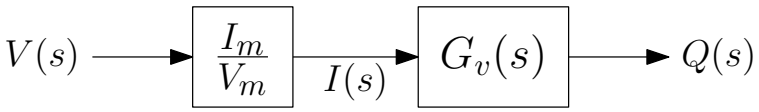


Figure 3.2: Block diagram of servovalve model.

On the SHM platform, the valve used is an MTS 252.21C servovalve. $G_v(s)$ can be approximated by a first order transfer function with $K_v = 1.2618 \frac{\text{L/s}}{\text{A}}$ and $\omega_c = 2\pi \cdot 100$ rad/s. The maximum voltage V_m is 10 V and the maximum current I_m is 25 mA.

3.1.2 Actuator

Several sources were consulted in the development of a functional linear actuator model. Originally, a model similar to that of [4] and [6] was developed. This model incorporated compressibility effects and test article motion at the point of application. The velocity and acceleration of the test article were "fed back" as inputs to the actuator. This formulation assumed that actuator acceleration and velocity were equivalent to that of the test article at the contact point. Because the load output depended on the acceleration, an input, there was a nonzero feedthrough matrix. A slightly different approach was taken when adopting this model. Instead of feeding back the test article motion, expressions were derived for the actuator dynamics. In addition, a leakage term was added because the MTS actuators have deliberate internal leakage to aid lubrication. There was no longer a feedthrough, but due to the presence of bulk modulus β in the state-space matrices—required when considering fluid compressibility—the formulation was very badly scaled. In numerical simulation, this manifested as drift.

An extensive literature review of hydraulic actuator models ensued, without much success. The models found were either nonlinear [20, 21, 22, 23] or did not have the required flow rate input or force output [22, 23, 24, 25]. Others [26, 27] were similar to the initial model and also resulted in drift. Because the system is in force control, it is necessary to have force as an output. As in the original model, issues arose when force was calculated as $F(t) = A\Delta P(t)$,

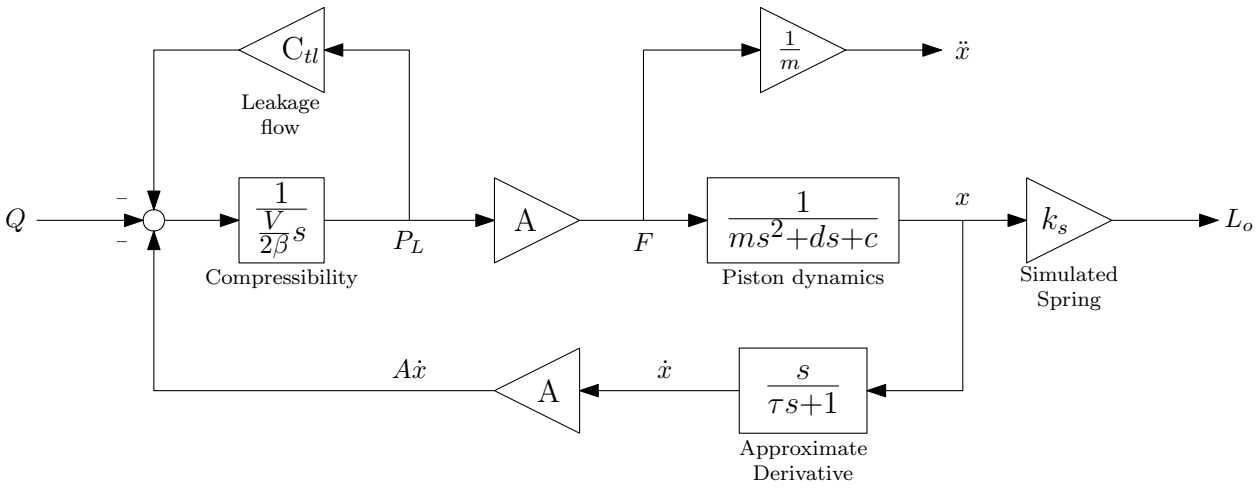


Figure 3.3: Block diagram of actuator model.

where F is force, A is actuator area, and ΔP is the pressure difference across the actuator. However, in some papers, such as Robinson and Pratt [28], a spring is modelled at the end of the actuator and force is given by $F(t) = kx(t)$. In this way, the initial output of the actuator is a displacement that is multiplied by a gain to give a force. This seemed to work better numerically, but would require a spring to be "simulated" on the SHM platform.

3.1.2.1 Simulated Spring

With the simulated spring as an option, a model with displacement as an output could be considered. In particular, a model inspired by Alleyne and Liu [29] was tested. This model accounts for compressibility effects and considers the actuator piston dynamics as a mass-spring-damper system. It conditions the problem with flow rate $Q(t)$ as an input and piston displacement $x(t)$ as an output. For the SHM platform, the mass of the actuator piston m and damping d are known, and the spring force acting on the piston can be taken to be the stiffness of the test article at the contact point. This stiffness c is found by averaging the values of k found in the $F = kx$ data collected on the SHM platform. In this way, some knowledge of the test article is incorporated into the plant without actually bringing the test article into the closed-loop system.

The simulated spring was added to the model to provide an output of force. The stiffness was originally found by taking the maximum force output (given in the actuator catalogue [30]) and dividing it by the maximum displacement (taken as half of the stroke length). However, this value of k_s was much too low to result in a realistic actuator response. Thus k_s was found while testing the numerical simulation at the maximum rated force and ensuring that input current and valve flow rate were also at their maximum values. The resulting

actuator model including the simulated spring is shown in Fig. 3.3.

3.1.2.2 Full Model

The overall actuator transfer functions are given by

$$G_{a_L}(s) = \frac{L(s)}{Q(s)} = \frac{Ak_s(\tau s + 1)(\text{1e-3})}{(\frac{V}{2\beta}s + C_{tl})(ms^2 + fs + c)(\tau s + 1) + A^2 s}, \tag{3.3}$$

$$G_{a_{\ddot{x}}}(s) = \frac{\ddot{x}(s)}{Q(s)} = \frac{F(s)/m}{Q(s)} = \frac{\frac{A}{m}(ms^2 + fs + c)(\tau s + 1)(\text{1e-3})}{(\frac{V}{2\beta}s + C_{tl})(ms^2 + fs + c)(\tau s + 1) + A^2 s}. \tag{3.4}$$

The transfer functions must be multiplied by 1e-3 to account for the units of flow rate $Q(t)$, which is expressed in L/s. Then the units of $G_{a_L}(s)$ and $G_{a_{\ddot{x}}}(s)$ are $\frac{\text{N}}{\text{L/s}}$ and $\frac{\text{m/s}^2}{\text{L/s}}$, respectively.

3.1.2.3 Numerical Issues and Non-Minimum Phase Zeros

When the actuator model given by (3.3) and (3.4) is combined with the servovalve model from Section 3.1.1 and the load cell from [6], the resulting plant transfer function is

$$G(s) = \frac{-2.825\text{e}07s^3 - 2.8275\text{e}10s^2 + 1.952\text{e}15s + 1.952\text{e}18}{s^5 + 8.742\text{e}04s^4 + 1.404\text{e}08s^3 + 5.857\text{e}10s^2 + 5.401\text{e}12s + 1.562\text{e}15}. \tag{3.5}$$

There was extreme difficulty in finding stabilizing proportional-integral (PI) gains to control this model. To guarantee stability and performance, at least an $\mathcal{H}_2$-optimal controller was required. In practice, however, it has been established that PI control can provide stability and tracking on the real rig.

Upon inspection, it was determined that the plant model given by (3.5) is poorly conditioned numerically and therefore difficult to work with for the purposes of simulation and controller design. The main issue is that $G(s)$ is non-minimum phase (NMP), which means it has zeros in the open right-half-plane (ORHP). ORHP zeros are detrimental because they limit controller gains and hinder system performance. Moreover, because ORHP zeros limit controller gains, the range of stabilizing controllers is reduced, making it more difficult to find a PI controller that BIBO stabilizes the closed-loop system.

At the same time, it was determined that the actuator and load cell should be considered as one mass, since the spring stiffness used in the piston dynamics block is that of the test article and the load cell is located between the actuator and the test article. Lumping the actuator and load cell masses together means that the force output of the actuator is directly the load applied to the test article, $L_o(t)$ rather than $L(t)$. The load cell from [6] was

therefore rearranged; the details of this are included in Section 3.1.3. The actuator transfer functions are thus given by

$$G_{a_{L_o}}(s) = \frac{L_o(s)}{Q(s)} = \frac{Ak_s(\tau s+1)(\text{1e-3})}{(\frac{V}{2\beta}s + C_{tl})(ms^2+fs+c)(\tau s+1) + A^2 s}, \tag{3.6}$$

$$G_{a_{\ddot{x}}}(s) = \frac{\ddot{x}(s)}{Q(s)} = \frac{F(s)/m}{Q(s)} = \frac{\frac{A}{m}(ms^2+fs+c)(\tau s+1)(\text{1e-3})}{(\frac{V}{2\beta}s + C_{tl})(ms^2+fs+c)(\tau s+1) + A^2 s}, \tag{3.7}$$

where m is now the combined mass of the actuator piston and the load cell. This actuator-load cell combination resulted in an overall plant transfer function that happens to be minimum phase. However, there remained issues with PI design as this model is still numerically complex.

3.1.2.4 Reduced-Order Modelling

A reduced-order modelling approach was taken in an attempt to eliminate the higher-order dynamics that were suspected to be the cause of numerical scaling issues within the model. An extensive survey of various model simplification methods ensued, with the resulting reduced-order models being used to synthesize PI gains. Closed-loop simulations were performed using each reduced-order model and its associated set of PI gains. The details of the survey are included in Appendix A. Upon analysis of the results, it was determined that the "best" method is to replace the actuator by a set of first-order transfer functions, where the cutoff frequency of the transfer function from flow rate $Q(s)$ to load $L_o(s)$ is lower than the natural frequency by an order of ten. This method was deemed to be the best in that the response is realistic (not too fast or too slow) and the model itself is easy to work with. In addition, it was possible to synthesize stabilizing PI gains for this model using 3 different synthesis methods.

In the model, the actuator transfer function (3.3) from flow rate $Q(s)$ to load $L_o(s)$ was replaced by a first-order transfer function given by

$$G_{a_{L_o}}(s) = \frac{L_o(s)}{Q(s)} = \frac{K_{\ell_{DC}}\omega_\ell}{s+\omega_\ell}, \tag{3.8}$$

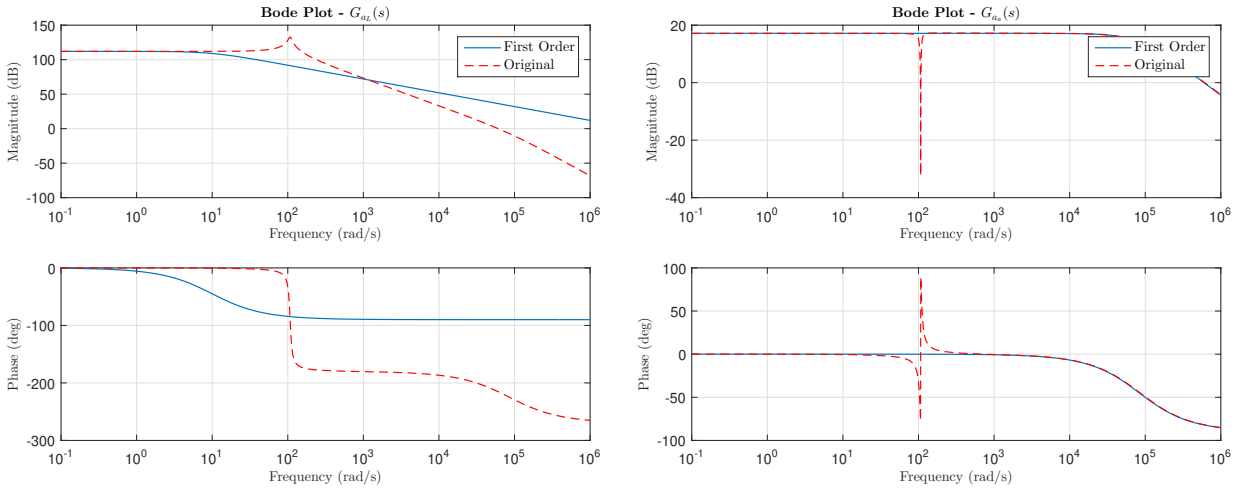
where $K_{\ell_{DC}}$ is the DC gain of the original transfer function and $\omega_\ell = \omega_n/10$ is used as the cutoff frequency. Initially ω_ℓ was set equal to ω_n, but this resulted in a response that was "too perfect," as seen in Fig. A.14b. Furthermore, preliminary system identification results suggest that the natural frequency of the system is lower than 100 rad/s. The values used for the MTS 252.12 actuators on the SHM platform are $K_{\ell_{DC}} = 396410\ \frac{\text{N}}{\text{L/s}}$ and

$\omega_\ell = 10.7190$ rad/s.

The actuator transfer function (3.4) from flow rate $Q(s)$ to acceleration $\ddot{x}(s)$ was replaced by a first-order transfer function given by

$$G_{a_{\ddot{x}}}(s) = \frac{\ddot{x}(s)}{Q(s)} = \frac{K_{a_{DC}}\omega_a}{s + \omega_a}, \tag{3.9}$$

where $K_{a_{DC}}$ is again the DC gain and $\omega_a = \omega_c$, where ω_c is the cutoff frequency. On the SHM platform, $K_{a_{DC}} = 7.2088 \frac{\text{m/s}^2}{\text{L/s}}$ and $\omega_a = 8.5595\text{e}04$ rad/s. The order-reduction method in (3.8) and (3.9) was automated using `MATLAB`. The resulting Bode diagram comparison for the MTS 252.12 actuator model is shown in Fig. 3.4.



(a) Bode plot of $G_{a_{L_o}}(s)$.

(b) Bode plot of $G_{a_{\ddot{x}}}(s)$.

Figure 3.4: Order-reduction for MTS 252.12 actuator model.

3.1.3 Load Cell

The load cell model is a modified version of the one used in [6], which approximates the load cell as two masses, one on the actuator side and one on the test article side (or the "measuring" side). The load cell can therefore be represented by a simple gain block. In [6], the load cell block takes in inputs of actuator load L and acceleration $\ddot{x}$ and provides outputs of measured load L_m and the load applied to the test article L_o. The relationship is given by

$$\begin{bmatrix} L_o(t) \\ L_m(t) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & -m_t \\ 1 & -(m_t - m_m) \end{bmatrix}}_{\mathbf{K}_{lc}} \begin{bmatrix} L(t) \\ \ddot{x}(t) \end{bmatrix}. \tag{3.10}$$

Writing out the equations for $L_o(t)$ and $L_m(t)$ from (3.10) explicitly yields two equations given by

$$L_o(t) = L(t) - m_t\ddot{x}(t), \tag{3.11}$$

$$L_m(t) = L(t) - (m_t - m_m)\ddot{x}(t). \tag{3.12}$$

However, because the actuator model from Section 3.1.2 has outputs of $L_o(t)$ and $\ddot{x}(t)$, (3.10) was rearranged to accept these as inputs while still providing $L_o(t)$ and $L_m(t)$ as outputs. This first involves solving for $L(t)$ in terms of $L_o(t)$ and $\ddot{x}(t)$. Rearranging (3.11) gives

$$L(t) = L_o(t) + m_t\ddot{x}(t). \tag{3.13}$$

Substituting (3.13) into (3.12) yields

$$\begin{aligned} L_m(t) &= (L_o(t) + m_t\ddot{x}(t)) - (m_t - m_m)\ddot{x}(t) \\ &= L_o(t) + m_t\ddot{x}(t) - m_t\ddot{x}(t) + m_m\ddot{x}(t). \end{aligned}$$

Cancelling terms, obtain

$$L_m(t) = L_o(t) + m_m\ddot{x}(t). \tag{3.14}$$

In matrix form, this becomes

$$\begin{bmatrix} L_o(t) \\ L_m(t) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ 1 & m_m \end{bmatrix}}_{\mathbf{K}_{lc}} \begin{bmatrix} L_o(t) \\ \ddot{x}(t) \end{bmatrix}. \tag{3.15}$$

The block diagram of the system is shown in Fig. 3.5. On the SHM platform, $m_t = 12.3955\,\text{kg}$ and $m_m = 3.3055\,\text{kg}$.

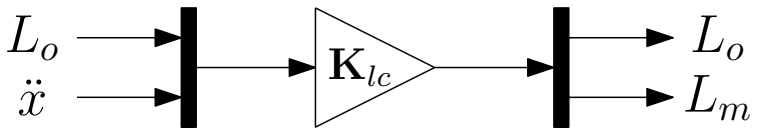


Figure 3.5: Block diagram of load cell.

3.1.4 Overall Plant Model

The plant model to be used for controller design is given by the overall system model from controller output $V(t)$ to feedback term $L_m(t)$. It consists of the servovalve, actuator, and

load cell. The test article is not included because it is outside the feedback loop, however, stiffness information from the test article is incorporated into the actuator model. In the SISO case, after combining the servovalve, reduced-order actuator, and load cell, the minimal plant transfer function is given by

$$G(s) = \frac{L_m(s)}{V(s)} = \frac{1.2465\text{e}07s + 7.21\text{e}11}{s^3 + 8.623\text{e}04s^2 + 5.471\text{e}07s + 5.765\text{e}08}, \tag{3.16}$$

which has poles at -85595.74, -628.32 and -10.72, and a zero at -57837.64. Therefore, $G(s)$ is both asymptotically stable and minimum phase.

If the full-order actuator model is used instead, the resulting plant transfer function is

$$G(s) = \frac{4.0525\text{e}06s^3 + 4.0525\text{e}09s^2 + 7.7025\text{e}14s + 7.7025\text{e}17}{s^5 + 8.742\text{e}04s^4 + 1.403\text{e}08s^3 + 5.574\text{e}10s^2 + 2.129\text{e}12s + 6.158\text{e}14}, \tag{3.17}$$

which is also asymptotically stable and minimum phase, but is numerically difficult to work with. Figure 3.6 shows a comparison of the frequency responses of (3.16) and (3.17).

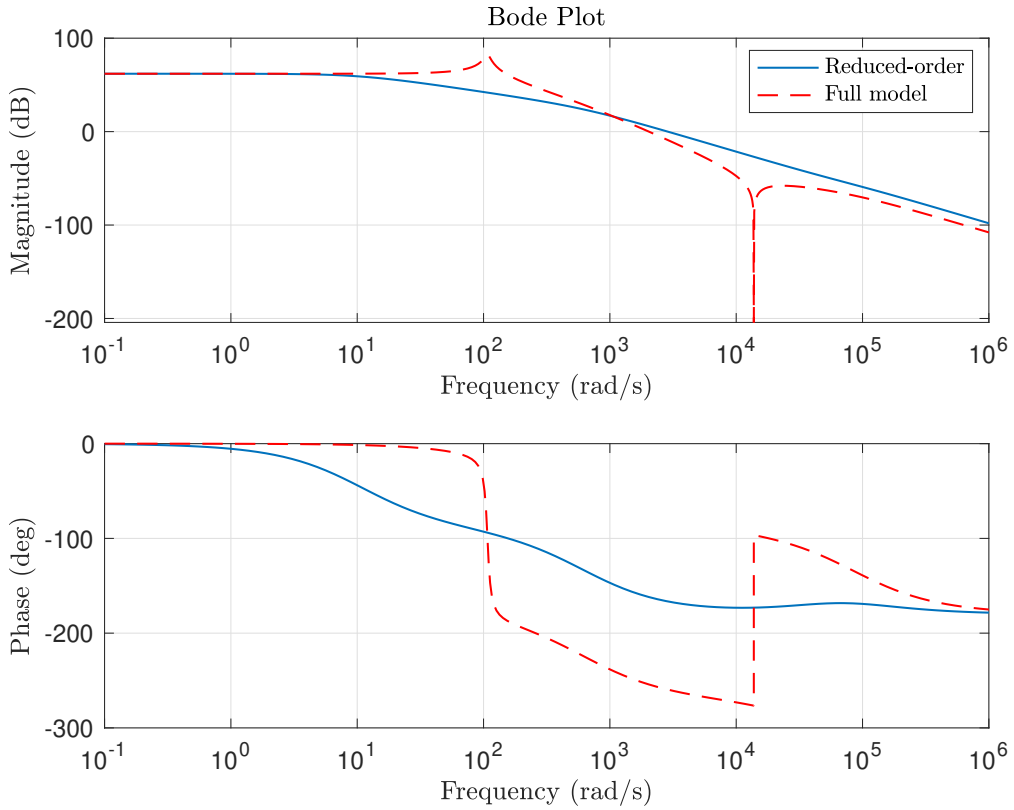


Figure 3.6: Frequency response comparison of reduced-order plant and full-order plant.

3.2 Test Article

The test article is the SHM platform, which is approximated as a beam for the purposes of this thesis. The input to the beam system is a load and the output is a displacement. Although the test article is outside the loop and thus its model is not required for controller synthesis, the model presented in this section can be used to simulate test article deflection.

3.2.1 Rayleigh-Ritz Method

The lateral displacement u_e of the beam was discretized using the Rayleigh-Ritz method [31]. In particular,

$$u_e(x,t) = \sum_{i=1}^{N} \Psi_i(x) q_{ei}(t),$$

where $\Psi_i(x)$ are the basis functions and $q_{ei}(t)$ are the elastic coordinates. The basis functions can be selected as $\psi_i = x^{i+1}$, which satisfy the boundary conditions of an Euler-Bernoulli cantilever beam [31]. Considering lateral deflection only, $\mathbf{\Psi}(x)$ is given by

$$\mathbf{\Psi}(x) = \begin{bmatrix} 0 & 0 & \dots & 0 \\ x^2 & x^3 & \dots & x^{N+1} \\ 0 & 0 & \dots & 0 \end{bmatrix},$$

where N is the number of modes or basis functions. Setting $N = 4$ and thus considering the first four modes proves to be adequate for the purposes of modelling and simulation.

3.2.2 Equations of Motion

For a cantilever beam, the equations of motion are given by

$$\mathbf{M}\ddot{\mathbf{q}}_e(t) + \mathbf{K}\mathbf{q}_e(t) = \mathbf{f}(t),$$

where

$$\mathbf{M} = \sigma \int_0^L \mathbf{\Psi}^\top(x)\mathbf{\Psi}(x)\mathrm{d}x,$$
$$\mathbf{K} = EI \int_0^L \mathbf{\Psi}''^\top(x)\mathbf{\Psi}''(x)\mathrm{d}x,$$

σ is the linear mass density in kg/m, and EI is the flexural rigidity in Nm^2, found by multiplying Young's modulus and the second moment of area.

3.2.3 State-Space Formulation

In matrix form, the equations of motion can be written as

$$\begin{bmatrix} \dot{\mathbf{q}}_e(t) \\ \ddot{\mathbf{q}}_e(t) \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ -\mathbf{M}^{-1}\mathbf{K} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{q}_e(t) \\ \dot{\mathbf{q}}_e(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1} \end{bmatrix} \mathbf{f}(t).$$

By the principle of virtual work for I concentrated loads, work is given by [32]

$$W = \left(\sum_{i=1}^{I} F_i(t) \mathbf{\Psi}(x_i) \right) \mathbf{q}(t).$$

Taking the variation results in

$$\delta W = \delta \mathbf{q}^\top \underbrace{\begin{bmatrix} \mathbf{\Psi}(x_1)^\top & \dots & \mathbf{\Psi}(x_I)^\top \end{bmatrix}}_{\hat{\mathbf{B}}} \underbrace{\begin{bmatrix} F_1(t) \\ \vdots \\ F_I(t) \end{bmatrix}}_{\mathbf{u}},$$

which implies that $\mathbf{f} = \hat{\mathbf{B}}\mathbf{u}$. Defining the states $\mathbf{x}_1(t) = \mathbf{q}_e(t)$ and $\mathbf{x}_2(t) = \dot{\mathbf{q}}_e(t)$, the state matrices of the process model are given by

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ -\mathbf{M}^{-1}\mathbf{K} & \mathbf{0} \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1}\hat{\mathbf{B}} \end{bmatrix} \mathbf{u}(t).$$

Defining $\mathbf{\Psi}_2(x)$ as the second row of $\mathbf{\Psi}(x)$, the displacement $u_{ei}(t)$ and velocity $\dot{u}_{ei}(t)$ at the point x_i are given by

$$\begin{bmatrix} u_{ei}(t) \\ \dot{u}_{ei}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{\Psi}_2(x_i) & \mathbf{0} \\ \mathbf{0} & \mathbf{\Psi}_2(x_i) \end{bmatrix} \begin{bmatrix} \mathbf{q}_e(t) \\ \dot{\mathbf{q}}_e(t) \end{bmatrix}.$$

The output is taken to be the displacement of the beam at $x = L$, which is the point of force application. If only one actuator acts on the beam, the system is SISO and the state-space matrices are given by

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{1} \\ -\mathbf{M}^{-1}\mathbf{K} & \mathbf{0} \end{bmatrix}}_{\mathbf{A}_b} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1}\hat{\mathbf{B}} \end{bmatrix}}_{\mathbf{B}_b} u(t), \qquad y(t) = \underbrace{\begin{bmatrix} \mathbf{\Psi}_2(L) & \mathbf{0} \end{bmatrix}}_{\mathbf{C}_b} \mathbf{x}(t) + \underbrace{0}_{\mathbf{D}_b} u(t).$$

3.3 MTS Controller

More than just a PID controller, the MTS controller model includes many built-in features such as null pacing, integrator limits, and cross-coupling compensation. The block diagram of the control system model is shown in Fig. 3.7, where y is the feedback force measured by the load cell and u is the control effort.

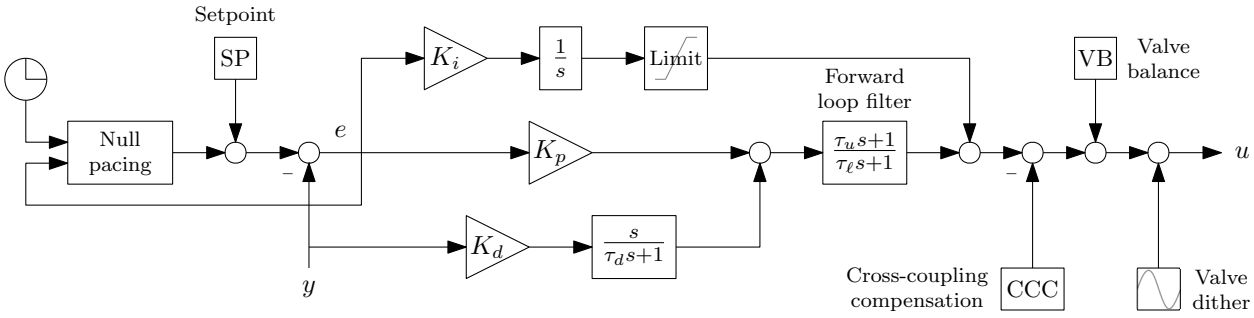


Figure 3.7: Controller block diagram.

3.3.1 Load Profile

The load profile is an array containing load end levels and corresponding transition times. The interpolation between end levels is performed according to the desired wave shape specified by the user in the MTS controller. Typically, a "haversine" wave shape is selected, where the derivative of the command is zero at every end level and there is an inflection point between each end level. The "haversine" wave shape has been approximated using the Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) interpolation method in `MATLAB`. A sample load profile segment is shown in Fig. 3.8. Although the interpolation method does not always exhibit the properties of the "haversine" wave shape, for the purposes of modelling, it was deemed to be good enough.

3.3.2 Null Pacing

Null pacing is a feature built-in to the MTS control system that slows the test depending on the error between feedback and reference signal. There are two types of null pacing, static and dynamic. Static null pacing occurs if the measured force is outside a specified tolerance at the load end level. Then the command is held until the load is achieved. Dynamic null pacing occurs if the error during the transition is outside a tolerance. Then the transition is slowed based on a *Dynamic Null-Pacing Adjustment Rate* until the feedback is within the tolerance. In both cases, the error limit is specified by the user.

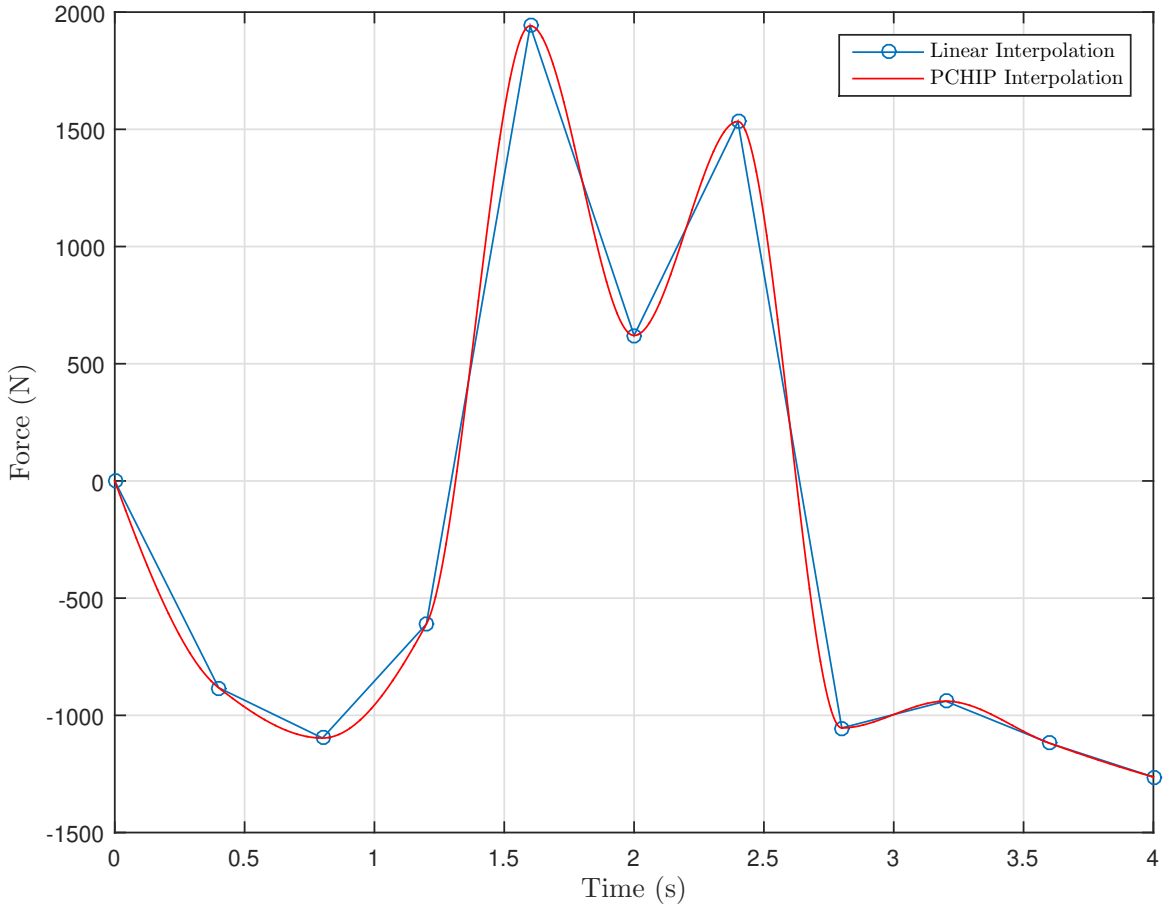


Figure 3.8: Linear vs. PCHIP interpolation.

The null pacing function was first modelled using `MATLAB` and then transitioned into *Simulink* using an embedded `MATLAB` function block. The block takes inputs of time and error, and outputs a command. The function uses *persistent* variables, where the load profile is loaded into the function workspace during the first call to the function. On subsequent runs, the data persists in memory within the function workspace. If static and/or dynamic null pacing occurs, the event is flagged and the load profile variables are modified. "To Workspace" blocks are used to output the final modified load profile in addition to a "flag" array containing occurrences of static and dynamic null pacing. Because a load profile has a variable end time based on the amount of null pacing that occurs, a "stop" flag has been built into the function to halt simulation when the last line of the load profile is reached.

A sample load profile segment with both static and dynamic null pacing is shown in Fig. 3.10. The vertical lines flag the start and end of a region of null pacing, where red lines indicate static null pacing and green lines indicate dynamic null pacing. The black circles show the original load levels and times. The amount of null pacing is the difference between the profile time and the actual runtime.

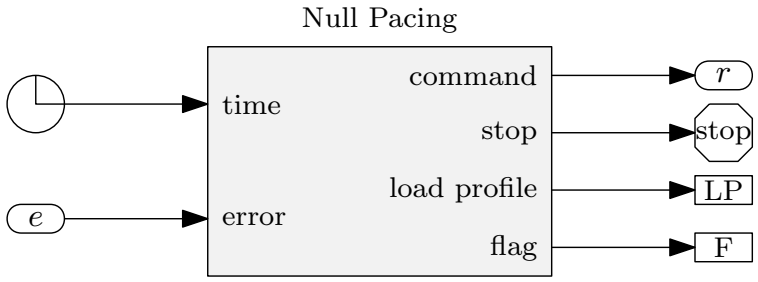


Figure 3.9: Null pacing block.

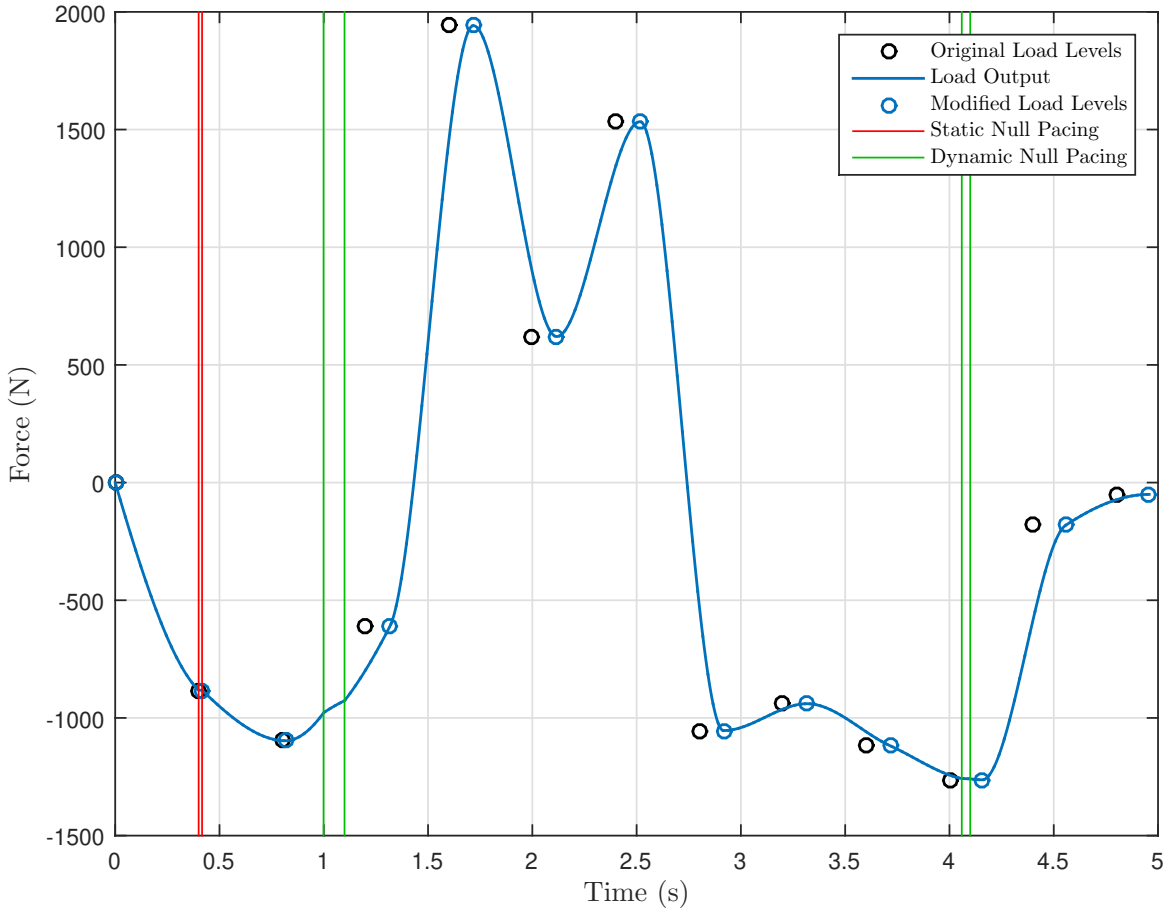


Figure 3.10: Load profile segment with null pacing.

3.3.3 Cross-Coupling Compensation (CCC)

Cross-coupling compensation (CCC) is a feature that is intended for test systems with multiple actuators, or channels. For example, the SHM platform at NRC is set up with 4 channels, corresponding to an actuator at each corner of the platform.

CCC is a control term that accounts for the effects the channels have on each other. These effects are represented by compliance coefficients, which can be obtained using the MTS test system. The compliance coefficients for the SHM platform are shown in Table 3.1.

Table 3.1: Compliance Coefficients

	$\mathbf{C}_1$	$\mathbf{C}_2$	$\mathbf{C}_3$	$\mathbf{C}_4$
$\mathbf{C}_1$	58	51	-32	-31
$\mathbf{C}_2$	93	105	-53	-53
$\mathbf{C}_3$	-52	-51	98	89
$\mathbf{C}_4$	-44	-44	76	87

The effect of channel 2 on channel 1 is given by the coefficient C_{12}, which is the value in the first row, second column of the table.

3.3.4 Integral Control and Integrator Limit

Integral control multiplies the integral of the error between reference signal and feedback by a constant gain, K_i. In the *Simulink* model, a pure integrator, $\frac{1}{s}$, is used.

The integral limit is a feature built into the MTS controller. It limits the amount of valve command used up by the integrator, specified as a percentage of the full-scale output. This is modelled using a saturation block in *Simulink*, where the upper and lower limits are specified as $\pm\, lim \cdot Q_{max}$ and Q_{max} is the full-scale output of the valve.

3.3.5 Proportional Control

In proportional control, the error between the reference signal and feedback is multiplied by a constant gain, K_p. In the MTS controller, the resulting control signal is passed through a forward loop filter (FLF), discussed in Section 3.3.7.

3.3.6 Derivative Control

In derivative control, the derivative of the feedback term is multiplied by a constant gain, K_d. In the MTS controller, a pure derivative is used, represented by $\frac{\mathrm{d}}{\mathrm{d}t}$ in the time domain and s in the frequency domain. In practice, however, pure derivative control is dangerous because it amplifies noise. Therefore, in the *Simulink* model this is replaced by an "approximate" derivative, $\frac{s}{\tau_d s+1}$, which includes a low-pass filter with a bandwidth defined by τ_d. The derivative control signal, along with the proportional control signal is passed through the FLF.

In practice, derivative control is never used in systems with force feedback. Load measurements tend to be very noisy, and this is exacerbated by the MTS controller which operates in pure derivative control.

3.3.7 Forward Loop Filter (FLF)

The forward loop filter is used to filter the control signal before it is injected on the test rig. In particular, it is applied to the sum of the proportional and derivative control signals, and has 5 possible modes of operation. The first, *Disabled*, means that no filtering is applied. The *Low-Pass* setting attenuates signals above a break frequency and limits filter attenuation at frequencies greater than a recover frequency. In *Band-Stop* mode, signals within a bandwidth of a notch frequency are attenuated. *Controller* mode uses a lead-lag compensator in place of the FLF, with up to two poles and two zeros. Finally, the *Optimized* setting uses predefined parameters for the FLF of a specific channel.

Typically, the FLF is used as a low-pass filter, where the transfer function is given by

$$G_{\text{FLF}}(s) = \frac{\tau_u s + 1}{\tau_\ell s + 1}.$$

The break frequency is given by $1/\tau_\ell$ and the recover frequency is given by $1/\tau_u$.

3.3.8 Setpoint

The setpoint is an offset that is applied to the drive signal. It is a constant value specified by the user with the same units as the reference signal, which is a load. On the MTS interface, loads are specified as pounds-force.

3.3.9 Valve Balance

Valve balance is used if the mechanical null of the servovalve is not at the physical centerpoint. It is the voltage that must be applied to the valve to ensure that the command and feedback are equal. If a valve is not centered, a nonzero valve balance is required to ensure the feedback term will settle at zero load. In the MTS system, this value can be found automatically or adjusted manually. In the *Simulink* model, a constant value must be specified before performing simulation.

3.3.10 Valve Dither

Valve dither is a high-frequency, low-amplitude sine signal which keeps the valve moving to counter the effects of friction. Dither is useful for sticky valves. The amplitude in volts

and frequency of the signal can be specified for each valve, but the frequency should be high enough that a force does not have time to develop.

3.4 System Units

After the fatigue test rig and MTS controller were modelled, real-time experiments showed that the MTS control law operates on error not in units of pounds-force (lbf), Newtons (N), or kilo-Newtons (kN), but in units of percentage of fullscale load divided by 10. Although error as a percentage of fullscale is displayed on the MTS system, load has underlying "base units" of kN but is displayed in lbf. Testing of the MTS controller confirmed that for a percent error of, for example, 1%, the control output is 0.1 V if the P gain is 1 and the I gain is zero. The same scaling applies to integral control, except the control signal will be proportional to the integral of error in percentage of fullscale divided by 10.

To incorporate these units into the closed-loop model, a unit conversion block can either be added to the plant model in Section 3.1 or the controller model in Section 3.3. On the SHM platform, the fullscale load is 2500 lbf, or 11.12 kN. To convert from Newtons to % of fullscale/10, the output of the plant model should be multiplied by a factor of

$$k_{\text{unit conv}} = \frac{100\%/10}{11.12\text{e}03\,\text{N}},$$

which is equivalent to 9.0e-04 %/N. Applying this scaling factor to (3.16) yields a transfer function given by

$$G(s) = \frac{1.122\text{e}04s + 6.489\text{e}08}{s^3 + 8.623\text{e}04s^2 + 5.471\text{e}07s + 5.765\text{e}08}. \tag{3.18}$$

This scaling factor should only be applied to the feedback force and not the force that is applied to the test article, due to the fact that the model given in Section 3.2 is based on input units of Newtons.

3.5 Cycle Estimation

Certain features in the MTS controller, in particular the null pacing feature, can have a significant effect on cycle time. Cycle time is estimated by running a given load profile through a *Simulink* model containing the MTS controller and plant model. A block diagram of the closed-loop system is shown in Fig. 3.11, where L_m is the feedback force measured by the load cell, u is the control effort, r is the reference signal, and u_e is the test article deflection. The model has been generalized to include disturbances, d, and noise, n.

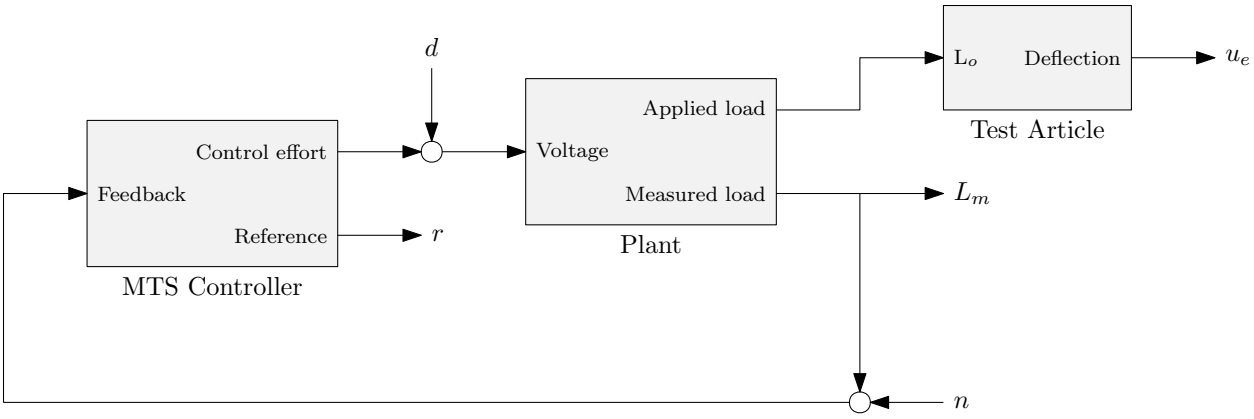


Figure 3.11: Block diagram of closed-loop system.

3.5.1 Simplified Test Article Model

The test article was initially modelled as an Euler-Bernoulli beam using a Rayleigh-Ritz discretization, as in Section 3.2. However, it was observed that using this beam model slowed down simulation and cycle times. Additionally, the stiffness of the test article is already incorporated into the actuator model. Furthermore, for most test articles, stiffness information is typically available even when a complete numerical model is not. For these reasons, it was decided to replace the beam model by the force relation $F = kx$ given by Hooke's law [33]. The resulting simplified transfer function is then

$$G_b(s) = \frac{u_e(s)}{L_o(s)} = \frac{1}{k_b}. \tag{3.19}$$

The effective beam stiffness k_b was found by dividing the maximum force output by the maximum actuator stroke. This resulted in similar deflection values to the original beam model, without slowing down cycle time.

3.5.2 Cycle Time Estimates

Before cycle times can be estimated, the controller and plant parameters must be selected. The controller and plant models are then combined in closed-loop and simulated with a given load profile in `MATLAB`/*Simulink*. Once the load profile is completed, the modified time array is available in the `MATLAB` workspace. These times are used to back out the corresponding transition times and update the load profile. Afterwards, the load profile can be used to perform tests on the real test rig.

3.5.3 Results

The SHM platform model given by (3.18) and the simplified test article from (3.19) were simulated using gains of $k_p = 0.95$ and $k_i = 28.5$ and a sample load profile consisting of 12 lines. The profile time was 4.8 seconds, and the actual time to complete the load profile was 4.96 seconds. The difference in time is due to null pacing, where the extension of the transition times can be seen by the difference between the red and black circles in Fig. 3.12.

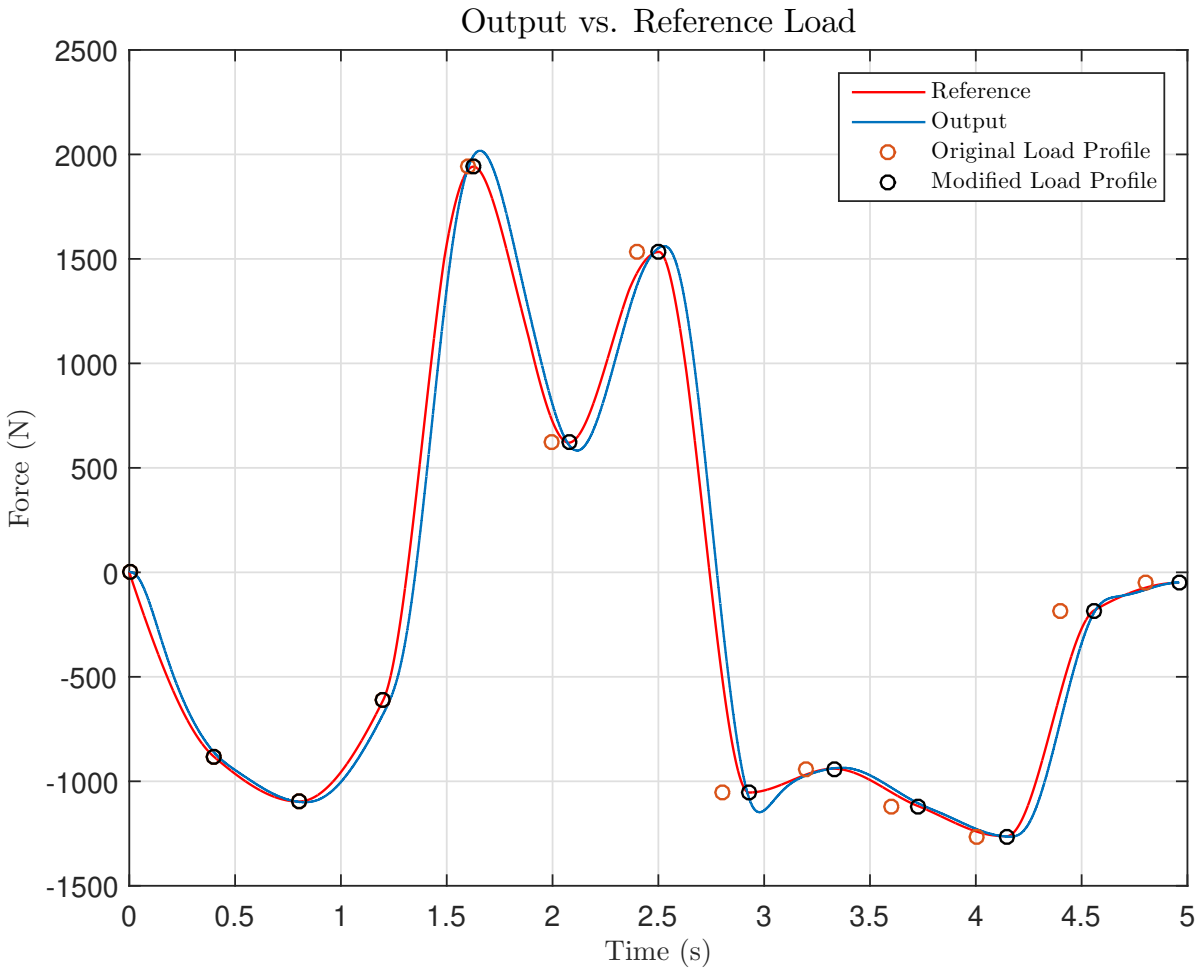


Figure 3.12: Load output.

Chapter 4

System Identification

As evidenced by the previous chapter, attempting to model a fatigue structural testing rig from first-principles is an extremely challenging undertaking. An alternative to analytical modelling, system identification is a technique that allows a numerical model to be "identified" using input-output data. This chapter discusses the linear time-domain system identification problem. Open-loop and closed-loop time domain methods are presented, however, only closed-loop methods can be implemented in the context of fatigue testing in order to protect the test article. Metrics for comparing identified models are discussed, and SISO and MIMO system identification results from the SHM platform at NRC are presented.

4.1 Problem Setup

The discrete-time difference equation (2.1) can be rewritten as

$$\begin{aligned}\mathbf{y}_k^\top &= -\mathbf{y}_{k-1}^\top \mathbf{A}_{n-1}^\top - \cdots - \mathbf{y}_{k-n}^\top \mathbf{A}_0^\top + \mathbf{u}_{k-\tau}^\top \mathbf{B}_m^\top + \cdots + \mathbf{u}_{k-\tau-m}^\top \mathbf{B}_0^\top \\ &= \underbrace{\begin{bmatrix} -\mathbf{y}_{k-1}^\top & \cdots & -\mathbf{y}_{k-n}^\top & \mathbf{u}_{k-\tau}^\top & \cdots & \mathbf{u}_{k-\tau-m}^\top \end{bmatrix}}_{\boldsymbol{\phi}_k} \boldsymbol{\Theta}, \end{aligned} \tag{4.1}$$

where $\boldsymbol{\Theta} = \begin{bmatrix} \mathbf{A}_{n-1} & \cdots & \mathbf{A}_0 & \mathbf{B}_m & \cdots & \mathbf{B}_0 \end{bmatrix}^\top$ is the parameter matrix.

Arranging the data at each timestep in this form yields

$$\underbrace{\begin{bmatrix} \mathbf{y}_N^\top \\ \mathbf{y}_{N-1}^\top \\ \vdots \\ \mathbf{y}_{n+1}^\top \\ \mathbf{y}_n^\top \end{bmatrix}}_{\boldsymbol{\Psi}} = \underbrace{\begin{bmatrix} \boldsymbol{\phi}_N \\ \boldsymbol{\phi}_{N-1} \\ \vdots \\ \boldsymbol{\phi}_{n+1} \\ \boldsymbol{\phi}_n \end{bmatrix}}_{\boldsymbol{\Phi}} \boldsymbol{\Theta} \quad \Leftrightarrow \quad \boldsymbol{\Phi}\boldsymbol{\Theta} = \boldsymbol{\Psi}. \tag{4.2}$$

Note that both $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$ span n_y columns. The matrices $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$ are comprised of data, and all the unknown parameters are contained in $\boldsymbol{\Theta}$. Therefore, (4.2) can be solved by a least-squares approach, as discussed in Section 2.2.

In the SISO case, (4.2) simplifies to

$$y_k = \underbrace{\begin{bmatrix} -y_{k-1} & \cdots & -y_{k-n} & u_{k-\tau} & \cdots & u_{k-\tau-m} \end{bmatrix}}_{\boldsymbol{\phi}_k} \boldsymbol{\theta},$$

where $\boldsymbol{\theta} = \begin{bmatrix} a_{n-1} & \cdots & a_0 & b_m & \cdots & b_0 \end{bmatrix}^\top$ is the parameter column matrix.

4.1.1 Model Order Selection

In the SISO case, the model order can be determined recursively as in Section 2.2.2. In the MIMO case, however, the order of each transfer function in the identified transfer matrix will be n_y times the selected order n. Therefore, for a 4-actuator system, if n is set to 1, the transfer functions will be 4^{th}-order. To prevent the identified model from being extremely high-order, n and m should be selected manually.

4.1.2 Data Generation

For $\boldsymbol{\Phi}$ to be full column rank and $\hat{\boldsymbol{\Theta}}$ to be a unique minimizing solution, the data should be sufficiently varied. This can be accomplished using a randomized input signal. In addition, because system identification involves finding a transfer matrix, the initial conditions of the system should ideally be zero when performing experiments. However, time-domain identification methods can account for nonzero initial conditions, while frequency-domain methods cannot [34]. The experiments must also excite all the relevant modes of the system, as the model will only capture what the data captures. This can be accomplished using an aggressive but realistic input spectrum.

4.2 Open-Loop Identification

Provided open-loop experiments can be performed, the open-loop system from $\mathbf{u}(s)$ to $\mathbf{y}(s)$ can be identified. If $\mathbf{v}(s)$ is zero-mean, Gaussian white noise, its effect should be averaged out when computing the least squares solution, which is the approach used in this research. Alternatively, the signal can be filtered before identification to diminish the effect of noise.

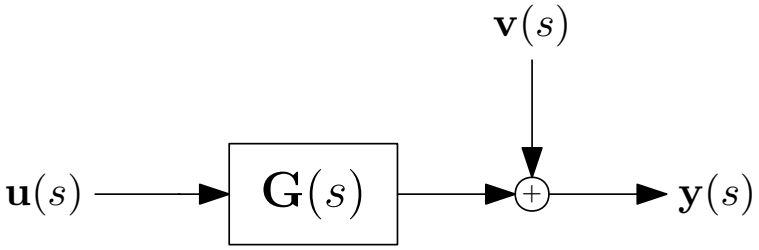


Figure 4.1: Open-loop system.

4.2.1 Open-Loop Identification Algorithm

1. Perform open-loop experiments. Measure and record inputs $\mathbf{u}(t_k)$ and outputs $\mathbf{y}(t_k)$.
2. Choose initial values for orders n and m and time delay τ. Typically start with n and m set to 1. The order of the system will be n_y times n, as discussed in Section 4.1.1.
3. Form data matrices $\mathbf{\Psi}$ and $\mathbf{\Phi}$ using time-domain data.
4. Solve for $\hat{\mathbf{\Theta}}$ using least squares or recursive least squares as in Section 2.2. Note that the recursive least squares algorithm has only been developed for the SISO case.
5. If a non-recursive least squares solver was used, try different values for n, m, and τ to see if the %VAF increases and/or NRMSE decreases significantly, as discussed in Sections 4.4.1 and 4.4.2.
6. Form the discrete-time transfer matrix,

$$\mathbf{G}(z^{-1}) = z^{-\tau}\mathbf{P}(z^{-1})^{-1}\mathbf{Q}(z^{-1}),$$

   where

$$\mathbf{P}(z^{-1}) = \mathbf{1} + \mathbf{A}_{n-1}z^{-1} + \cdots + \mathbf{A}_1 z^{-n+1} + \mathbf{A}_0 z^{-n},$$
$$\mathbf{Q}(z^{-1}) = \mathbf{B}_m + \mathbf{B}_{m-1}z^{-1} + \cdots + \mathbf{B}_1 z^{-m+1} + \mathbf{B}_0 z^{-m}.$$

7. Convert the transfer function from discrete-time to continuous-time.

4.3 Closed-Loop Identification

Closed-loop system identification is necessary for a variety of reasons. In the case where a system is unstable, it cannot be run in open-loop. Alternatively, some systems, such as fatigue testing rigs, can only be run in closed-loop to avoid damaging the test article. Closed-loop experiments can also help capture desired system behaviour. This report discusses three main approaches to the closed-loop system identification problem.

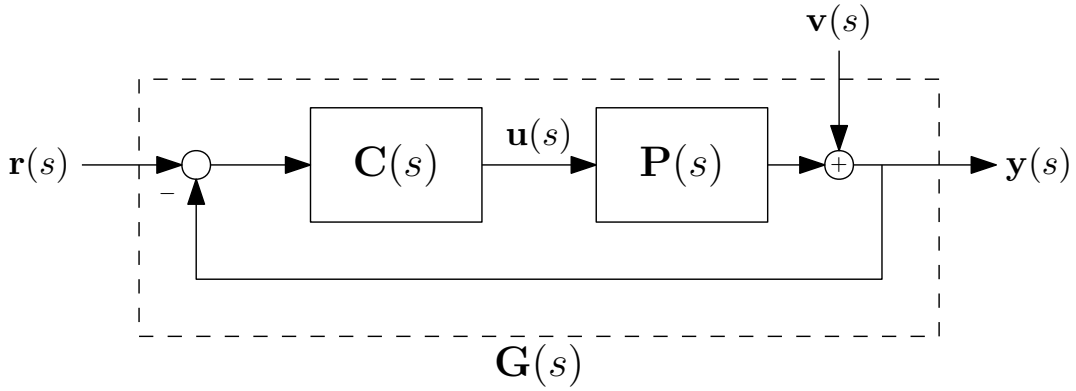


Figure 4.2: Closed-loop system.

4.3.1 Direct Method

In the direct method, inputs $\mathbf{u}(s)$ and outputs $\mathbf{y}(s)$ are used to identify $\mathbf{P}(s)$ directly. This approach ignores the controller and feedback, essentially saying it shouldn't matter that $\mathbf{u}(s)$ comes from a controller [35]. This assumption allows the use of open-loop identification methods.

4.3.1.1 Direct Closed-Loop Identification Algorithm

1. Perform closed-loop experiments.
2. Measure inputs $\mathbf{u}(t_k)$ and outputs $\mathbf{y}(t_k)$.
3. Use open-loop methods to identify the plant $\mathbf{P}(s)$.

4.3.2 Indirect Method

The indirect method uses the reference $\mathbf{r}(s)$ and outputs $\mathbf{y}(s)$ to identify the closed-loop system $\mathbf{G}(s)$. If the controller $\mathbf{C}(s)$ is known, it is possible to solve for the plant transfer matrix $\mathbf{P}(s)$ [35].

The closed-loop transfer matrix from $\mathbf{r}(s)$ to $\mathbf{y}(s)$ is given by

$$\mathbf{G}(s) = (\mathbf{1} + \mathbf{P}(s)\mathbf{C}(s))^{-1}\mathbf{P}(s)\mathbf{C}(s).$$

This can be rearranged to solve for the plant transfer matrix, yielding [36]

$$\mathbf{P}(s) = \mathbf{G}(s)(\mathbf{C}(s) - \mathbf{C}(s)\mathbf{G}(s))^{-1}.$$

Note that the order of $\mathbf{G}(s)$ is the order of the plant plus the order of the controller.

4.3.2.1 Indirect Closed-Loop Identification Algorithm

1. Perform closed-loop experiments using a known controller $\mathbf{C}(s)$.
2. Measure excitation $\mathbf{r}(t_k)$ and output $\mathbf{y}(t_k)$.
3. Use open-loop methods to identify the transfer matrix $\mathbf{G}(s)$.
4. Compute the plant transfer matrix as

$$\mathbf{P}(s) = \mathbf{G}(s)(\mathbf{C}(s) - \mathbf{C}(s)\mathbf{G}(s))^{-1}.$$

4.3.3 Dual-Youla Method

In the dual-Youla method, the dual to the Youla parametrization of all stabilizing controller is used to identify the plant model. A Youla parametrization is used to transform the closed-loop system into an open-loop system [1].

4.3.3.1 Youla Parametrization [15, 37]

Theorem 4.1 (All Stabilizing Controllers [15])**.** *Let $\mathbf{P}_0(s)$ be a plant and $\mathbf{C}_0(s)$ be any controller that BIBO stabilizes $\mathbf{P}_0(s)$. Write $\mathbf{P}_0(s)$ and $\mathbf{C}_0(s)$ in terms of their right coprime factors as*

$$\mathbf{P}_0(s) = \mathbf{N}_0(s)\mathbf{M}_0(s)^{-1}, \qquad \mathbf{C}_0(s) = \mathbf{Y}_0(s)\mathbf{X}_0(s)^{-1}.$$

where $\mathbf{M}_0(s)$, $\mathbf{N}_0(s)$, $\mathbf{X}_0(s)$, $\mathbf{Y}_0(s)$ are all BIBO stable transfer matrices. Two transfer matrices $\mathbf{A}(s)$ and $\mathbf{B}(s)$ are right coprime if and only if there exists $\mathbf{U}(s)$ and $\mathbf{V}(s)$ such that [38]

$$\mathbf{U}(s)\mathbf{A}(s) + \mathbf{V}(s)\mathbf{B}(s) = \mathbf{1}. \tag{4.3}$$

Equation (4.3) *is known as the* Bezout identity. *Then all controllers that BIBO stabilize* $\mathbf{P}_0(s)$ *have the form*

$$\mathbf{C}(s) = (\mathbf{Y}_0(s) + \mathbf{M}_0(s)\mathbf{Q}(s))\,(\mathbf{X}_0(s) - \mathbf{N}_0(s)\mathbf{Q}(s))^{-1},$$

where $\mathbf{Q}(s)$ *is any BIBO stable transfer matrix.*

4.3.3.2 Dual-Youla Parametrization [1]

The dual is useful for system identification, where given a particular controller $\mathbf{C}_0(s)$, $\mathbf{P}(s)$ is selected from the set of all plants stabilized by $\mathbf{C}_0(s)$. The plant can be written as

$$\mathbf{y}(s) = \mathbf{P}(s)\mathbf{u}(s) + \mathbf{H}(s)\mathbf{w}(s). \tag{4.4}$$

Theorem 4.2 (MIMO $(\mathbf{R}(s), \mathbf{S}(s))$ Parametrization [39])**.** *Let* $\mathbf{C}_0(s)$ *be any compensator and* $\mathbf{P}_0(s)$ *be any noise-free nominal plant stabilized by* $\mathbf{C}_0(s)$. *Let* $\mathbf{C}_0(s) = \mathbf{Y}_0(s)^{-1}\mathbf{X}_0(s) = \tilde{\mathbf{X}}_0(s)\tilde{\mathbf{Y}}_0(s)^{-1}$ *and* $\mathbf{P}_0(s) = \mathbf{M}_0(s)^{-1}\mathbf{N}_0(s) = \tilde{\mathbf{N}}_0(s)\tilde{\mathbf{M}}_0(s)^{-1}$ *be left and right coprime factorizations of the compensator and nominal plant that satisfy*

$$\begin{bmatrix} \mathbf{X}_0(s) & \mathbf{Y}_0(s) \\ -\mathbf{N}_0(s) & \mathbf{M}_0(s) \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{M}}_0(s) & -\tilde{\mathbf{Y}}_0(s) \\ \tilde{\mathbf{N}}_0(s) & \tilde{\mathbf{X}}_0(s) \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}. \tag{4.5}$$

Then the set of all plants stabilized by $\mathbf{C}_0(s)$ *can be parametrized by the pair* $(\mathbf{R}(s), \mathbf{S}(s))$, *where* $\mathbf{R}(s)$ *is any BIBO stable transfer matrix and* $\mathbf{S}(s)$ *is any BIBO stable, stably invertible transfer matrix (i.e., it is biproper and minimum phase), in the form given by* (4.4) *with*

$$\begin{aligned} \mathbf{P}(s) &= (\mathbf{M}_0(s) - \mathbf{R}(s)\mathbf{Y}_0(s))^{-1}(\mathbf{N}_0(s) + \mathbf{R}(s)\mathbf{X}_0(s)) \\ &= (\tilde{\mathbf{N}}_0(s) + \tilde{\mathbf{X}}_0(s)\mathbf{R}(s))(\tilde{\mathbf{M}}_0(s) - \tilde{\mathbf{Y}}_0(s)\mathbf{R}(s))^{-1} \end{aligned} \tag{4.6}$$

and

$$\mathbf{H}(s) = (\mathbf{M}_0(s) - \mathbf{R}(s)\mathbf{Y}_0(s))^{-1}\mathbf{S}(s). \tag{4.7}$$

Substituting (4.6) and (4.7) into (4.4) and rearranging yields

$$(\mathbf{M}_0(s) - \mathbf{R}(s)\mathbf{Y}_0(s))\mathbf{y}(s) = (\mathbf{N}_0(s) + \mathbf{R}(s)\mathbf{X}_0(s))\mathbf{u}(s) + \mathbf{S}(s)\mathbf{w}(s),$$

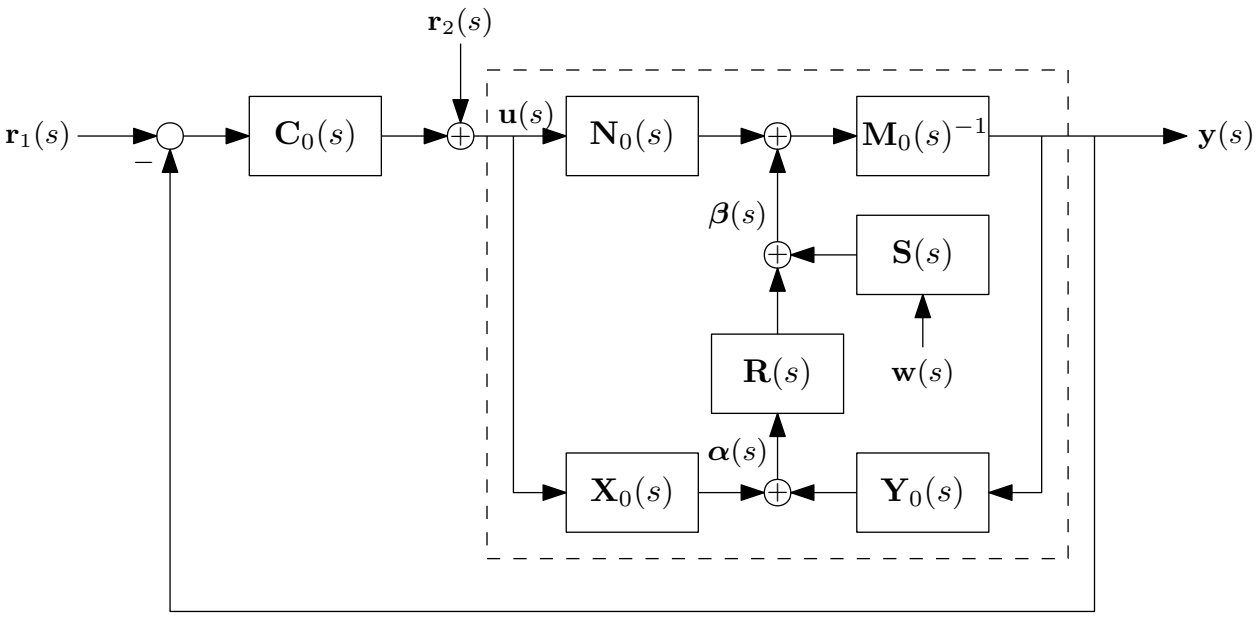


Figure 4.3: Reparametrized plant and noise structure [1].

which can be rearranged to obtain

$$\underbrace{\mathbf{M}_0(s)\mathbf{y}(s) - \mathbf{N}_0(s)\mathbf{u}(s)}_{\boldsymbol{\beta}(s)} = \mathbf{R}(s)\underbrace{(\mathbf{Y}_0(s)\mathbf{y}(s) + \mathbf{X}_0(s)\mathbf{u}(s))}_{\boldsymbol{\alpha}(s)} + \mathbf{S}(s)\mathbf{w}(s).$$

The "open-loop" system is now given by

$$\boldsymbol{\beta}(s) = \mathbf{R}(s)\boldsymbol{\alpha}(s) + \mathbf{S}(s)\mathbf{w}(s). \tag{4.8}$$

The reparametrization of the plant is included in Fig. 4.3, which shows that

$$\boldsymbol{\alpha}(s) = \mathbf{X}_0(s)\mathbf{u}(s) + \mathbf{Y}_0(s)\mathbf{y}(s). \tag{4.9}$$

Equation (4.9) is equivalent to

$$\begin{aligned}
\boldsymbol{\alpha}(s) &= \mathbf{X}_0(s)\left(\mathbf{C}_0(s)\left(\mathbf{r}_1(s) - \mathbf{y}(s)\right) + \mathbf{r}_2(s)\right) + \mathbf{Y}_0(s)\mathbf{y}(s) \\
&= \mathbf{X}_0(s)\left(\mathbf{X}_0(s)^{-1}\mathbf{Y}_0(s)\left(\mathbf{r}_1(s) - \mathbf{y}(s)\right) + \mathbf{r}_2(s)\right) + \mathbf{Y}_0(s)\mathbf{y}(s) \\
&= \mathbf{Y}_0(s)(\mathbf{r}_1(s) - \mathbf{y}(s)) + \mathbf{X}_0(s)\mathbf{r}_2(s) + \mathbf{Y}_0(s)\mathbf{y}(s) \\
&= \mathbf{Y}_0(s)\mathbf{r}_1(s) + \mathbf{X}_0(s)\mathbf{r}_2(s) - \mathbf{Y}_0(s)\mathbf{y}(s) + \mathbf{Y}_0(s)\mathbf{y}(s),
\end{aligned}$$

which, after cancelling the last two terms, yields

$$\boldsymbol{\alpha}(s) = \mathbf{Y}_0(s)\mathbf{r}_1(s) + \mathbf{X}_0(s)\mathbf{r}_2(s). \tag{4.10}$$

Equation (4.10) can be used to compute $\boldsymbol{\alpha}(s)$ given values of $\mathbf{r}_1(s)$ and $\mathbf{r}_2(s)$. Typically $\mathbf{r}_1(s)$ is the command $\mathbf{r}(s)$ and $\mathbf{r}_2(s)$ is a disturbance, $\mathbf{d}(s)$. If disturbances are assumed to be zero, the "open-loop" input is given by

$$\boldsymbol{\alpha}(s) = \mathbf{Y}_0(s)\mathbf{r}(s).$$

The values of $\boldsymbol{\alpha}(s)$ and $\boldsymbol{\beta}(s)$ are used to identify $\mathbf{R}(s)$. Then the plant transfer function is computed as

$$\mathbf{P}(s) = (\mathbf{M}_0(s) - \mathbf{R}(s)\mathbf{Y}_0(s))^{-1}(\mathbf{N}_0(s) + \mathbf{R}(s)\mathbf{X}_0(s)).$$

4.3.3.3 Dual-Youla Closed-Loop Identification Algorithm

1. Select the stabilizing controller $\mathbf{C}_0(s)$ and its left coprime factors $\mathbf{X}_0(s)$, $\mathbf{Y}_0(s)$.
2. Use this controller to perform closed-loop experiments with chosen excitation(s) $\mathbf{r}_1(t_k)$ and/or $\mathbf{r}_2(t_k)$.
3. Measure inputs $\mathbf{u}(t_k)$ and outputs $\mathbf{y}(t_k)$.
4. Choose an initial plant model $\mathbf{P}_0(s)$ that is stabilized by $\mathbf{C}_0(s)$, and express the plant in terms of its left coprime factors $\mathbf{M}_0(s)$, $\mathbf{N}_0(s)$. Select the coprime factors to be identity, $\mathbf{1}$, if possible.
5. Filter the excitation and measurements to obtain $\boldsymbol{\alpha}(t_k)$ and $\boldsymbol{\beta}(t_k)$, where

$$\boldsymbol{\alpha}(s) = \mathbf{Y}_0(s)\mathbf{r}_1(s) + \mathbf{X}_0(s)\mathbf{r}_2(s), \qquad \boldsymbol{\beta}(s) = \mathbf{M}_0(s)\mathbf{y}(s) - \mathbf{N}_0(s)\mathbf{u}(s).$$

6. Use $\boldsymbol{\alpha}(t_k)$ and $\boldsymbol{\beta}(t_k)$ to estimate $\mathbf{R}(s)$ using open-loop techniques.
7. Compute the plant transfer function as

$$\mathbf{P}(s) = (\mathbf{M}_0(s) - \mathbf{R}(s)\mathbf{Y}_0(s))^{-1}(\mathbf{N}_0(s) + \mathbf{R}(s)\mathbf{X}_0(s)).$$

4.4 Model Comparison and Error Metrics

When performing system identification, it is necessary to select a metric that can be used to compare identified models. Ideally, this metric should be non-dimensional or normalized to allow for comparison across different datasets. The metric will be computed for both the training data and validation data. The "best" model is selected to be the one that performs best in validation, as this ensures that the model is robust.

Two metrics will be used to assess the performance and robustness of identified system models. The preferred metric is percent variance accounted for (%VAF), which has a best possible value of 100%. Normalized root-mean-square (NRMS) error will also be discussed. In the case of NRMS error, the best possible value is zero. For MIMO systems, both %VAF and NRMSE will be computed for each column of data.

4.4.1 Variance Accounted For

Percent variance accounted for (%VAF), which provides a measure of the quality of fit for linear models, is given by [40, 41]

$$\%\text{VAF} = 100 \times \left(1 - \frac{\text{var}(\mathbf{y} - \hat{\mathbf{y}})}{\text{var}(\mathbf{y})}\right), \tag{4.11}$$

where $\mathbf{y}$ is measured data, $\hat{\mathbf{y}}$ is data simulated using the identified model, and

$$\text{var}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} x_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N} x_i\right)^2$$

is the variance of variable $\mathbf{x}$ [41]. %VAF should only be used if the residuals are zero-mean, white, and normally distributed. The %VAF attains its highest possible value of 100 in the case that the model is perfectly accurate and there is no noise. In general, the "best" model is taken to be the one that has the highest %VAF in validation.

4.4.2 NRMS Error

Root-mean-square (RMS) error is given by

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - y_i)^2}, \tag{4.12}$$

where $\mathbf{y}$ is measured data and $\hat{\mathbf{y}}$ is data simulated using the identified model. This metric is not a useful comparison across different datasets as they can differ in length, amplitude, etc.

Equation (4.12) can be normalized by dividing it by $\sqrt{\frac{1}{N}\sum_{i=0}^{N} y_i^2}$. This yields

$$\text{NRMSE} = \sqrt{\frac{\frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - y_i)^2}{\frac{1}{N}\sum_{i=0}^{N} y_i^2}}.$$

If the model is able to perfectly recreate the data, the NRMS error would be exactly zero. In general, the "best" model is taken to be the one that has the lowest mean NRMS error in validation.

4.4.3 Simulated Control Effort

To further verify the accuracy of the identified plant models, the identified plant can be recombined with the known controller in order to simulate control effort. In the case of the indirect and dual-Youla closed-loop methods, the measured control effort is not directly used to solve the least squares problem. Regardless, a good model should be able to accurately reproduce this data. Therefore, the %VAF and NRMSE will also be computed for simulated control effort.

4.5 Results

The SHM fatigue testing rig at NRC, pictured in Fig. 1.1 was used to perform closed-loop experiments. Initially, SISO tests were performed by running a randomized load profile on one actuator at a time. Next, MIMO tests were performed on each half of the SHM platform using a randomized 2-actuator load profile. Finally, MIMO tests were performed on the entire platform by simultaneously running a profile on all 4 actuators. In all cases, the first 30 seconds of data was used as training data for the three closed-loop identification algorithms, with the remainder being used for validation. In addition to simulating the output of the identified models, the control effort was also simulated.

4.5.1 SISO System Identification

SISO tests were performed on all 4 actuators, and the results for actuator 3 are discussed here. Figure 4.4 shows the load profile used to perform closed-loop experiments, where the first 30 seconds was used for training and the remainder for validation. The controller used

was a PI controller with $k_p = 20$ and $k_i = 20$, which has a transfer function given by

$$C(s) = k_p + \frac{k_i}{s} = \frac{k_p s + k_i}{s} = \frac{20s + 20}{s}. \tag{4.13}$$

In all cases, a recursive least squares method was used to determine the model order and the data was down-sampled to a sample time of 0.1 second, which is in accordance with the "rule of thumb" mentioned in Section 2.6.2 since the operating frequency of the test is approximately 1 Hz. Table 4.1 includes the results for all of the system identification methods.

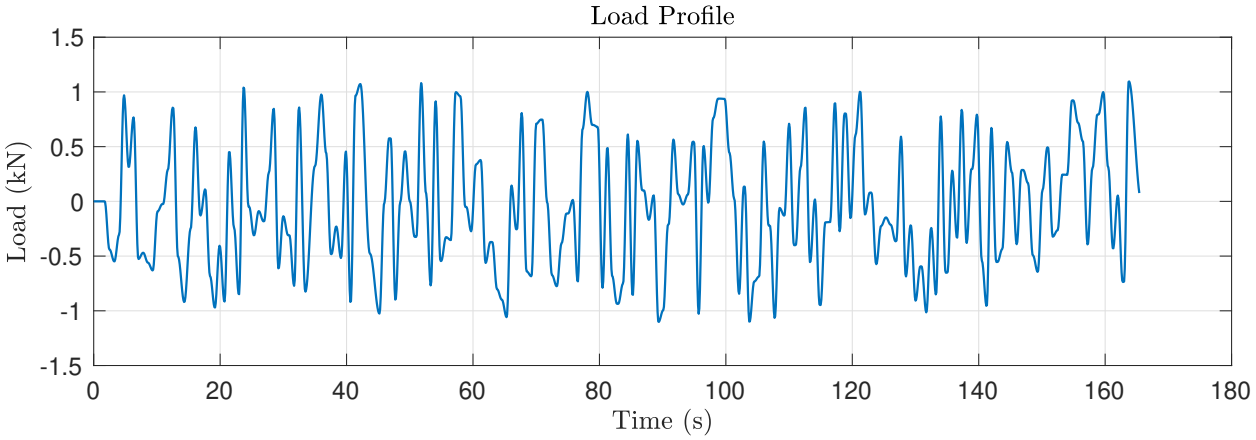


Figure 4.4: Load profile for SISO system identification.

Table 4.1: SISO system ID results for actuator 3.

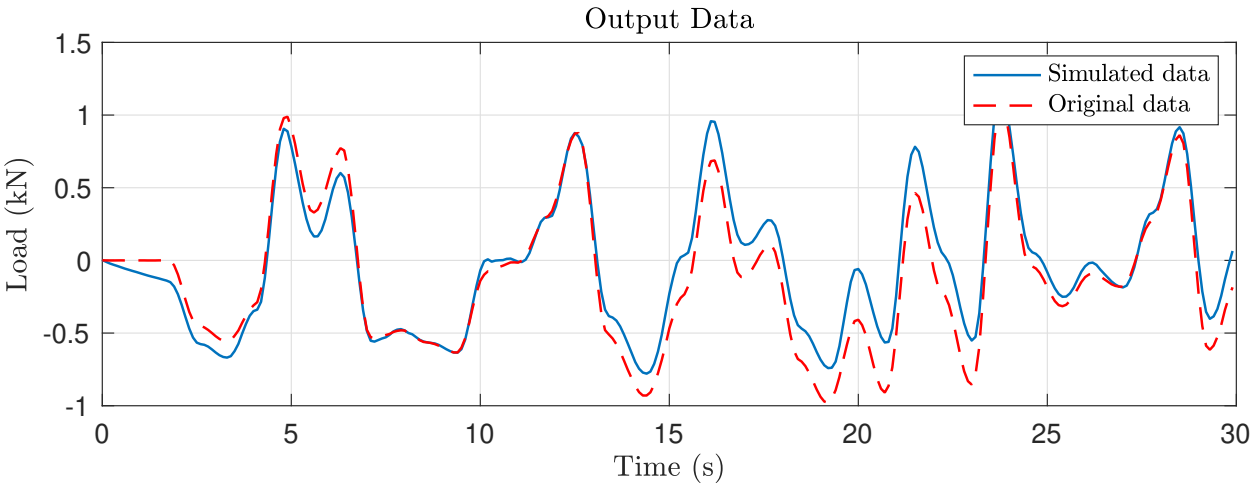
Identification Method	**NRMSE**				**%VAF**			
	Training		**Validation**		**Training**		**Validation**	
	Input	**Output**	**Input**	**Output**	**Input**	**Output**	**Input**	**Output**
Direct	0.1722	0.3373	0.1744	0.3804	97.09	90.77	97.00	90.86
Indirect	0.7442	0.0148	2.0770	0.0154	75.06	99.98	6.68	99.98
Dual-Youla	0.1452	0.0125	0.1473	0.0129	97.89	99.98	97.85	99.98

4.5.1.1 Direct Method

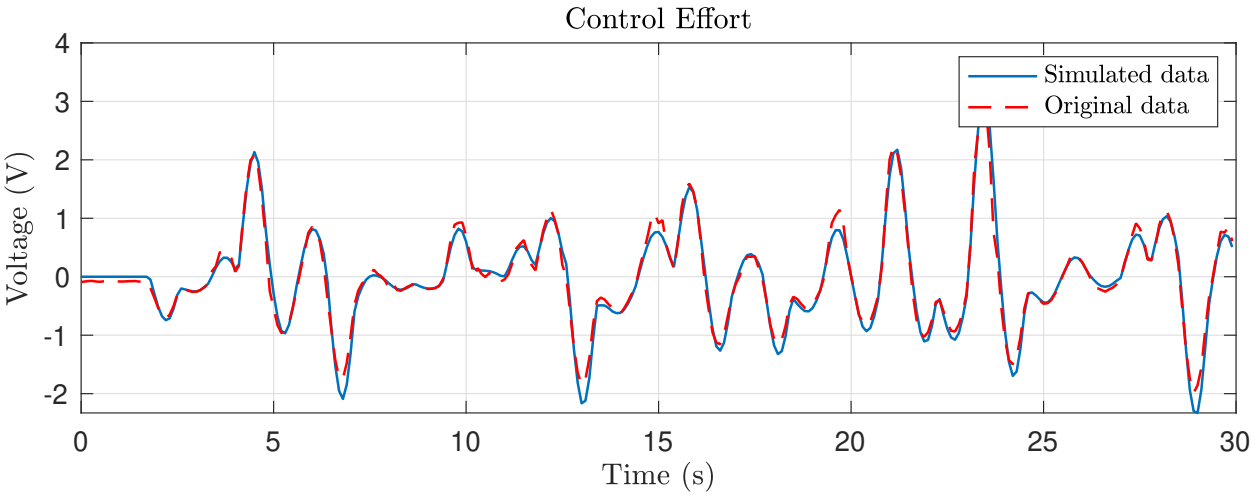
The direct closed-loop method identified a 4^{th}-order BIBO stable non-minimum phase (NMP) transfer function given by

$$P(s) = \frac{-0.02733s^4 + 0.3592s^3 + 28.62s^2 + 286.7s + 1147}{s^4 + 25.7s^3 + 244.4s^2 + 1113s + 202.3}.$$

Figure 4.5 provides a comparison between the measured input and output data and the corresponding simulated data in training.



(a) Load data.



(b) Voltage data.

Figure 4.5: Direct method—training data for actuator 3.

4.5.1.2 Indirect Method

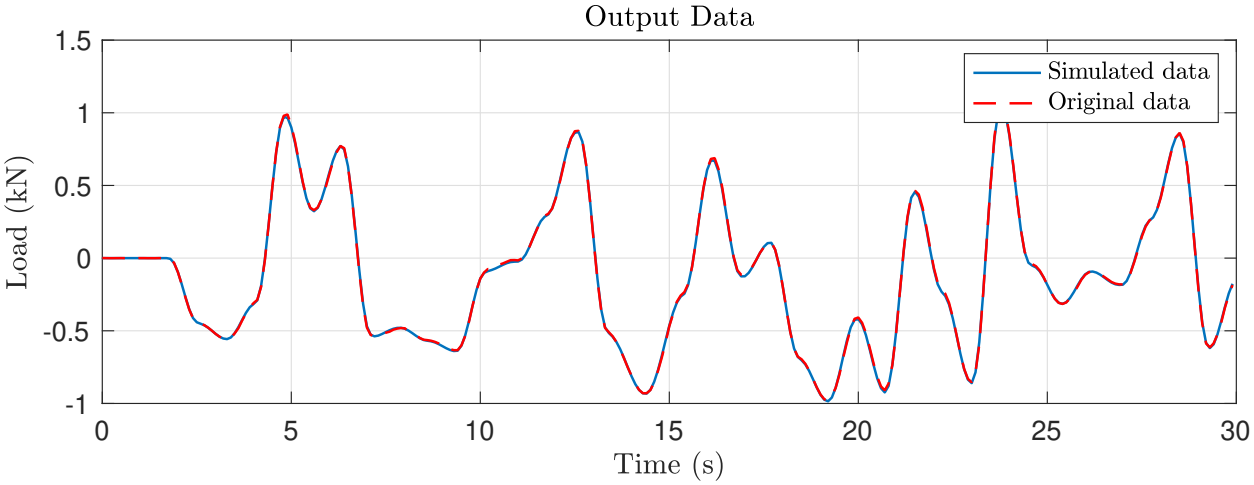
The indirect closed-loop method identified the closed-loop system as a 3$^{\text{rd}}$-order BIBO stable minimum phase (MP) transfer function given by

$$G(s) = \frac{0.3004s^3 + 29.8s^2 + 447.4s + 1163}{s^3 + 46.13s^2 + 478s + 1145},$$

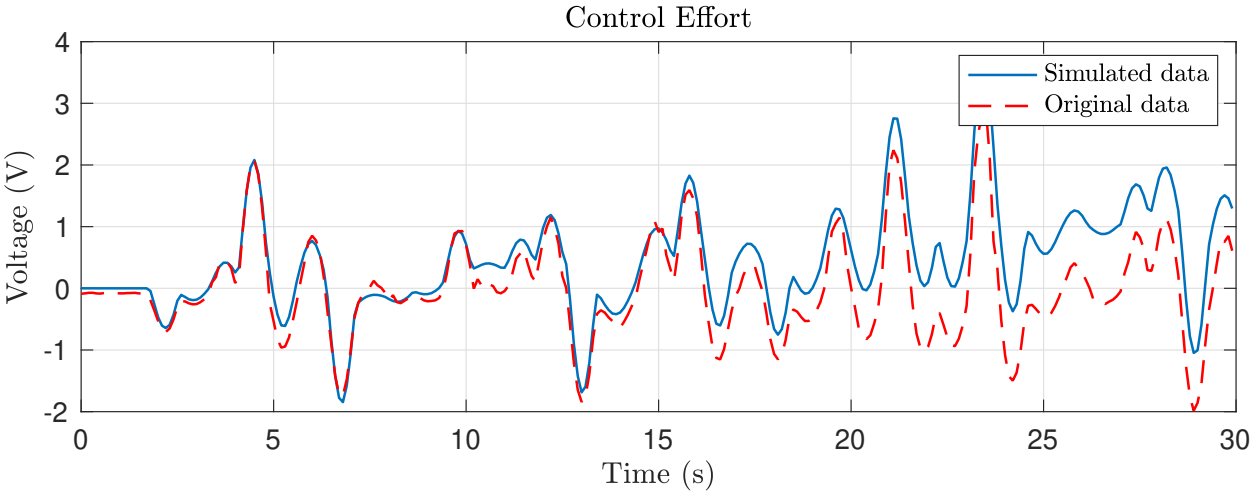
which resulted in a 2nd-order unstable MP plant system given by

$$P(s) = \frac{0.003601s^2 + 1.418s + 6.198\text{e-}16}{s^2 + 0.4618s - 0.4423}.$$

Training data results are shown in Fig. 4.6.



(a) Load data.



(b) Voltage data.

Figure 4.6: Indirect method—training data for actuator 3.

4.5.1.3 Dual-Youla Method

For the dual-Youla closed-loop method, the controller coprime factors were selected as

$$X_0(s) = \frac{s}{s+20}, \qquad Y_0(s) = \frac{20s+20}{s+20},$$

and the initial plant coprime factors were chosen to be

$$M_0(s) = 1, \qquad N_0(s) = 1.$$

The algorithm identified a 2^{nd}-order BIBO stable NMP transfer function for $R(s)$ given by

$$R(s) = \frac{-0.8048s^2 - 16.52s + 19.81}{s^2 + 23.58s + 19.85}.$$

This resulted in a 3^{rd}-order BIBO stable MP plant system given by

$$P(s) = \frac{0.01142s^3 + 1.583s^2 + 29.91s + 23.22}{s^3 + 22.82s^2 + 24.9s + 0.04249}. \tag{4.14}$$

The training results are shown in Fig. 4.7.

4.5.2 MIMO Identification on Actuators 1 & 2

MIMO closed-loop experiments were performed on actuators 1 & 2 using a load profile shown in Fig. 4.8 and a PI controller with $k_p = 10$ and $k_i = 50$ on each actuator. The controller transfer matrix is given by

$$\mathbf{C}(s) = \left(\frac{10s + 50}{s}\right)\mathbf{1}. \tag{4.15}$$

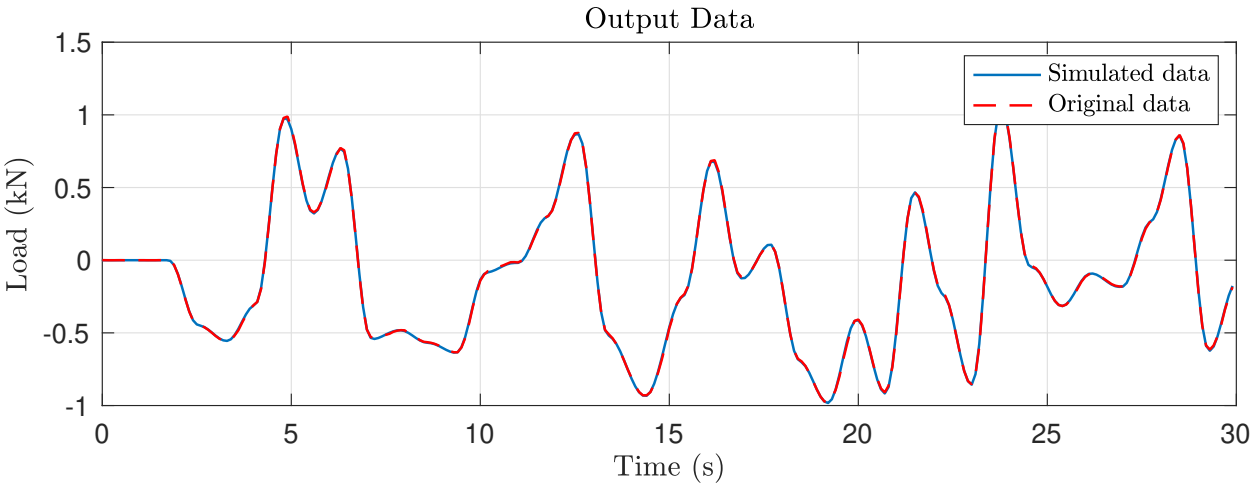
The data was down-sampled to a sample time of 0.1 seconds, which corresponds to a sampling frequency of 10 Hz. Tables 4.2 and 4.3 show the NRMSE and %VAF for all three closed-loop identification methods.

Table 4.2: NRMSE for MIMO identification of actuators 1 & 2.

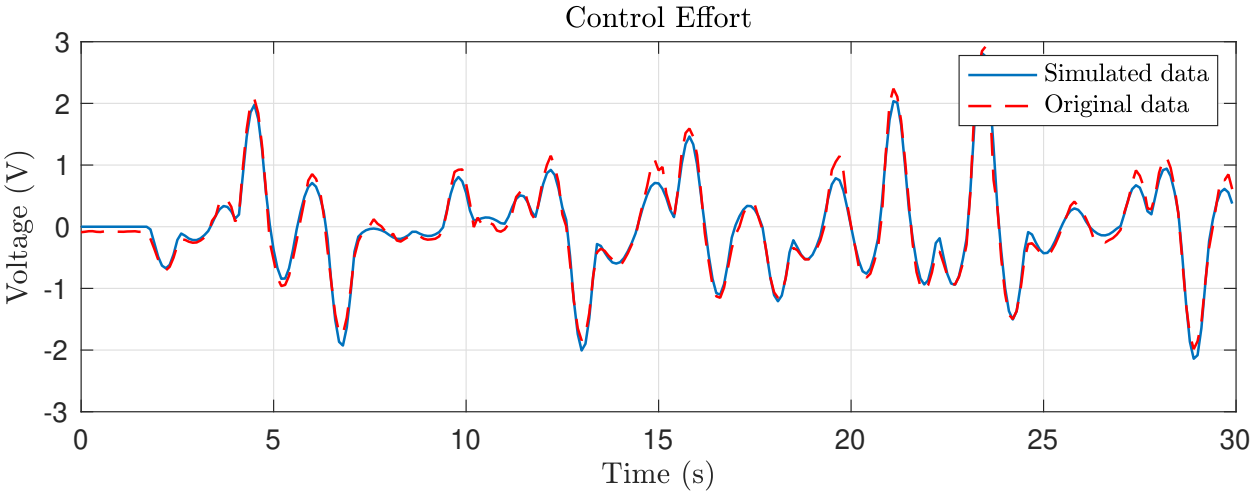
Identification Method	**Training**				**Validation**			
	Input		**Output**		**Input**		**Output**	
	C1	**C2**	**C1**	**C2**	**C1**	**C2**	**C1**	**C2**
Direct	0.7650	0.3563	0.6649	0.6600	0.6713	0.3527	0.7714	0.6993
Indirect	0.4580	0.7525	0.0319	0.0363	0.8801	1.1715	0.0477	0.0387
Dual-Youla	0.2988	0.2088	0.0328	0.0372	0.3057	0.1887	0.0463	0.0385

4.5.2.1 Direct Method

The direct closed-loop method identified a 4^{th}-order BIBO stable transfer matrix for $\mathbf{P}(s)$. Training data results are shown in Fig. 4.9.



(a) Load data.



(b) Voltage data.

Figure 4.7: Dual-Youla method—training data for actuator 3.

Table 4.3: %VAF for MIMO identification of actuators 1 & 2.

Identification Method	**Training**				**Validation**			
	Input		**Output**		**Input**		**Output**	
	C1	**C2**	**C1**	**C2**	**C1**	**C2**	**C1**	**C2**
Direct	42.27	87.34	55.78	53.03	57.37	87.71	39.66	58.46
Indirect	82.62	73.26	99.90	99.84	75.58	72.14	99.77	99.85
Dual-Youla	92.62	95.77	99.89	99.83	92.45	96.49	99.78	99.85

4.5.2.2 Indirect Method

The indirect method identified the closed-loop system $\mathbf{G}(s)$ as a 8^{th}-order BIBO stable transfer matrix. This translated to a 7^{th}-order unstable transfer matrix for $\mathbf{P}(s)$. Figure 4.10

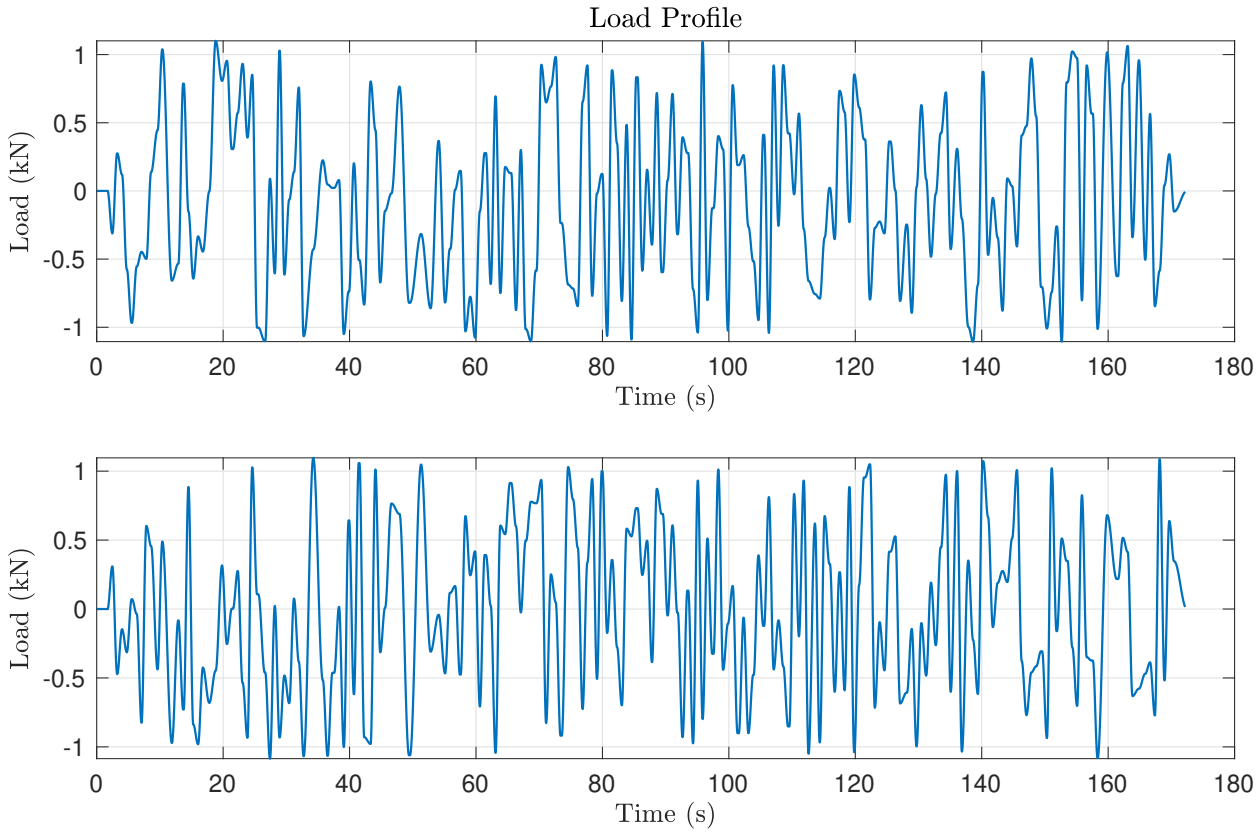


Figure 4.8: Load profile for MIMO system identification of actuators 1 & 2.

shows the results in training.

4.5.2.3 Dual-Youla Method

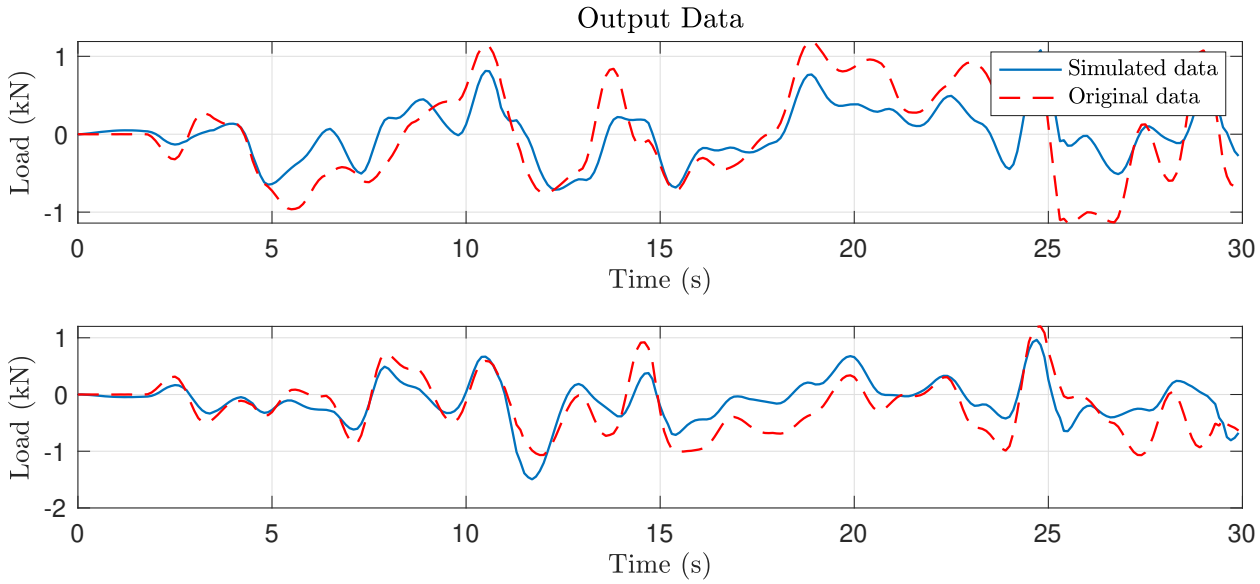
The left coprime factors for the controller and initial plant were selected as

$$\mathbf{X}_0(s) = \left(\frac{s}{s+10}\right)\mathbf{1}, \qquad \mathbf{Y}_0(s) = \left(\frac{10s+50}{s+10}\right)\mathbf{1}, \qquad \mathbf{M}_0(s) = \mathbf{1}, \qquad \mathbf{N}_0(s) = \mathbf{1}.$$

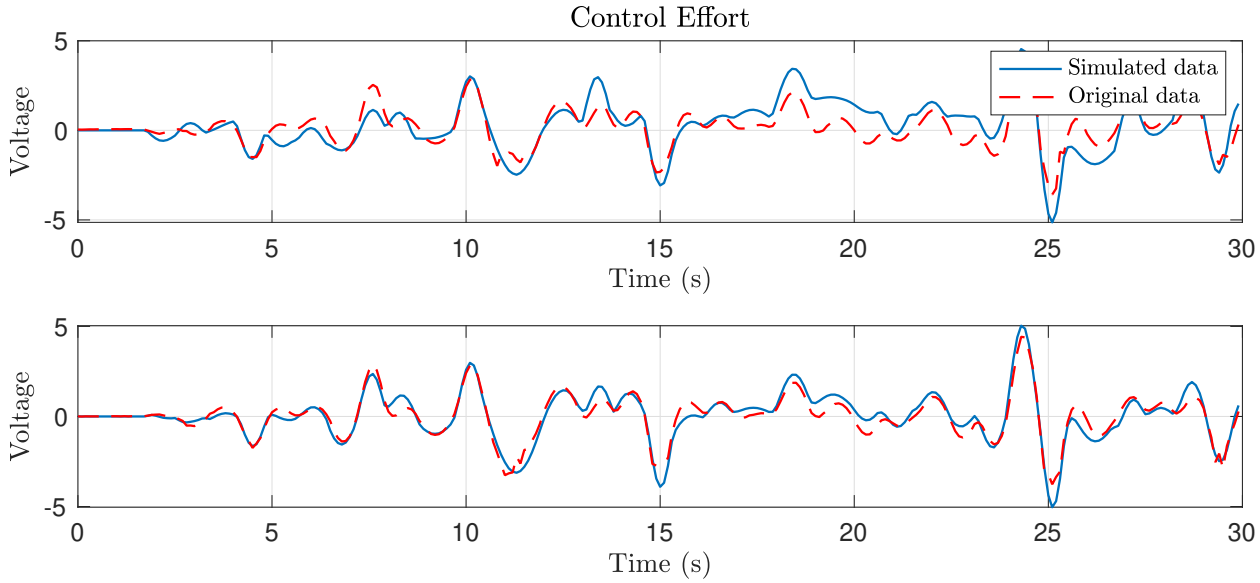
The identified $\mathbf{R}(s)$ was a 4^{th}-order BIBO stable transfer matrix, which corresponded to a 5^{th}-order $\mathbf{P}(s)$ that has one ORHP pole at 0.0592. Although the identified plant is unstable, this transfer matrix resulted in the best feedforward and $\mathcal{H}_\infty$ controllers in Chapters 6 and 7, indicating that it is a good model despite the instability. The training results for the dual-Youla method are shown in Fig. 4.11.

4.5.3 MIMO Identification on Actuators 3 & 4

MIMO closed-loop experiments were also performed on actuators 3 & 4 using the load profile from Fig. 4.8. A PI controller was used, with $k_p = 10$ and $k_i = 40$, yielding the



(a) Load data.



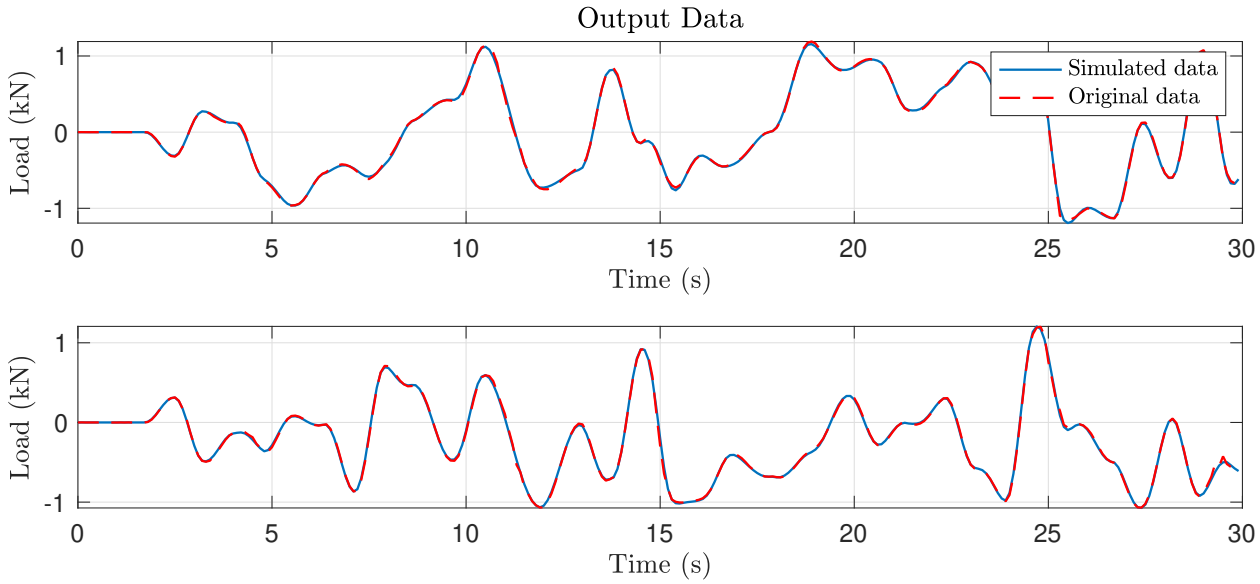
(b) Voltage data.

Figure 4.9: Direct method—training data for actuators 1 & 2.

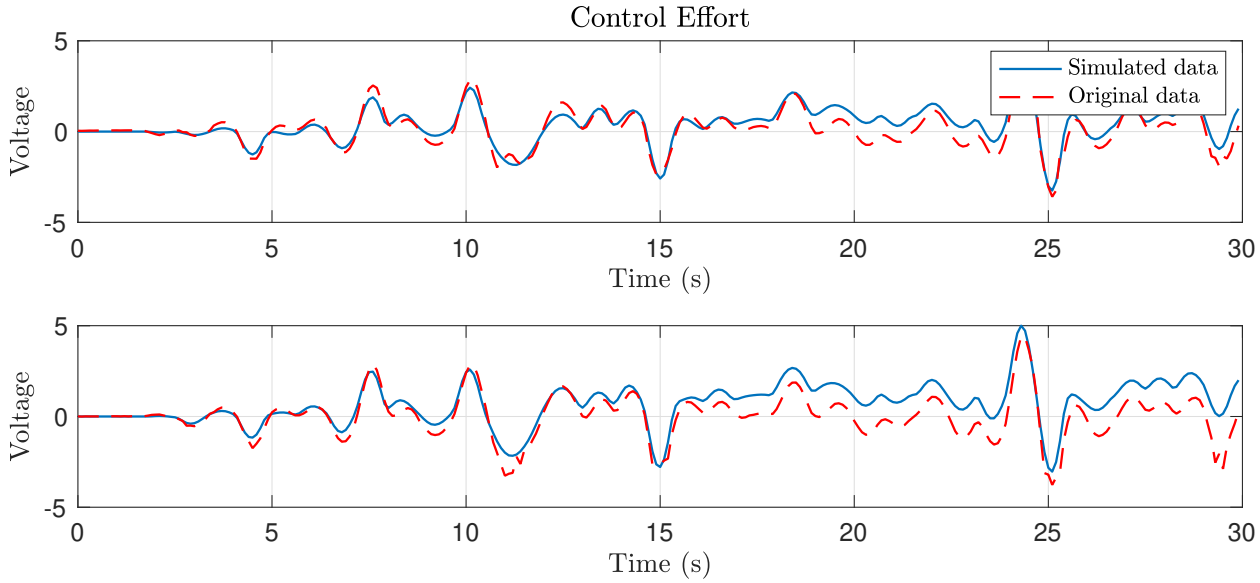
controller transfer matrix

$$\mathbf{C}(s) = \left(\frac{10s + 40}{s}\right)\mathbf{1}. \tag{4.16}$$

Tables 4.4 and 4.5 show the NRMS errors and %VAF values for all three closed-loop identification methods.



(a) Load data.



(b) Voltage data.

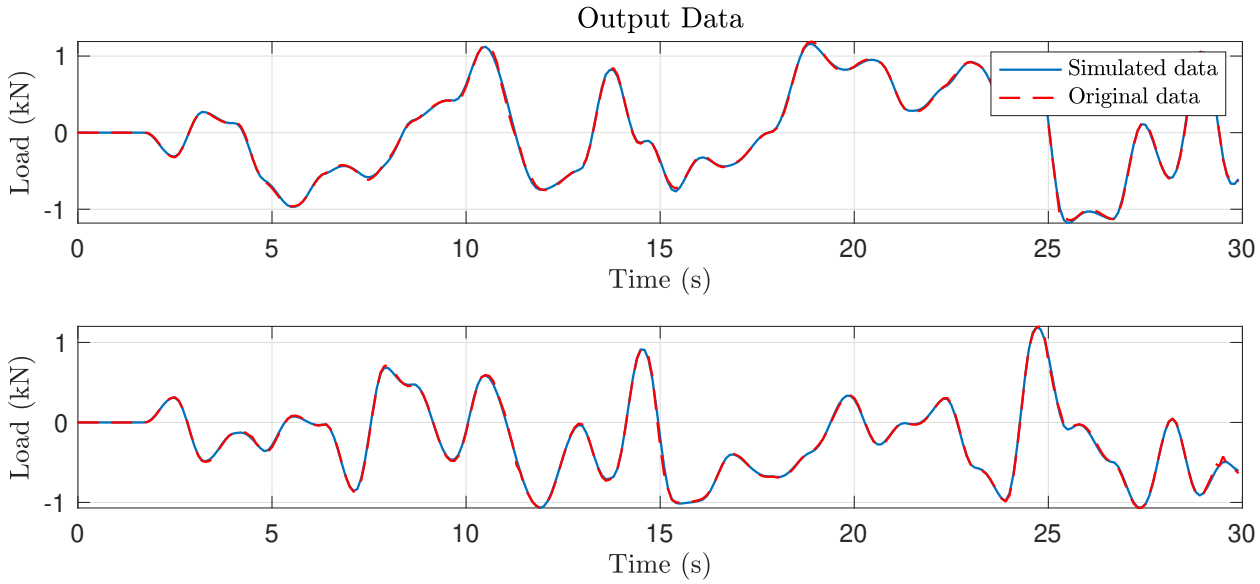
Figure 4.10: Indirect method—training data for actuators 1 & 2.

4.5.3.1 Direct Method

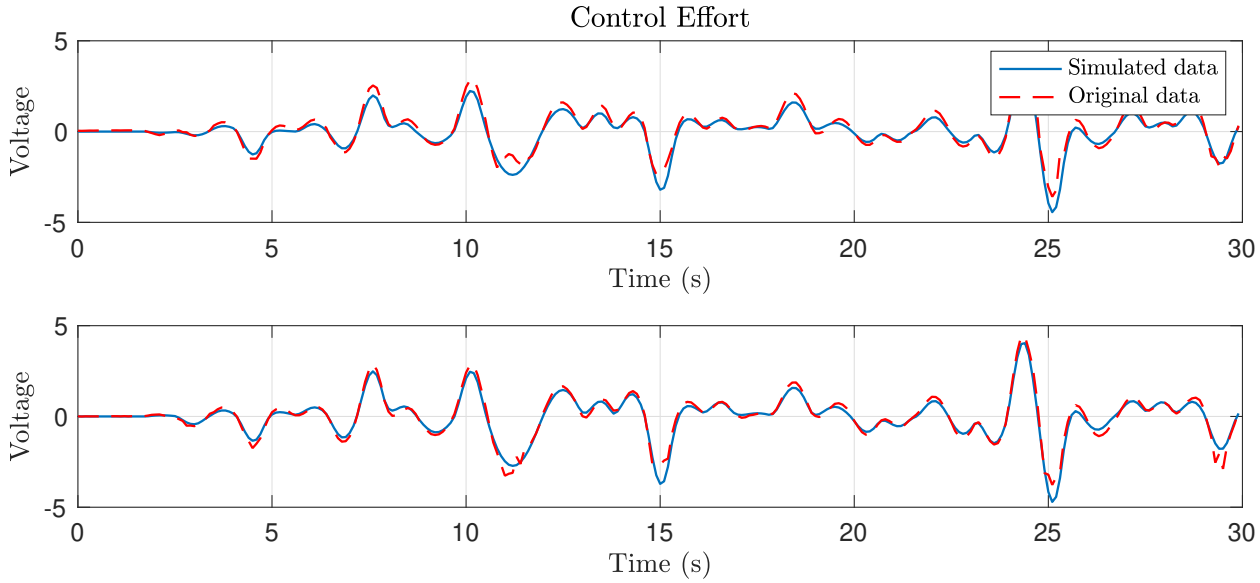
The direct method identified $\mathbf{P}(s)$ as a 4^{th}-order BIBO stable transfer matrix.

4.5.3.2 Indirect Method

Using the indirect method, $\mathbf{G}(s)$ was identified as a 6^{th}-order BIBO stable transfer matrix. This resulted in a 5^{th}-order transfer matrix for $\mathbf{P}(s)$ with two ORHP poles.



(a) Load data.



(b) Voltage data.

Figure 4.11: Dual-Youla method—training data for actuators 1 & 2.

4.5.3.3 Dual-Youla Method

The left coprime factors for the dual-Youla method were chosen to be

$$\mathbf{X}_0(s) = \left(\frac{s}{s+15}\right)\mathbf{1}, \qquad \mathbf{Y}_0(s) = \left(\frac{10s+40}{s+15}\right)\mathbf{1}, \qquad \mathbf{M}_0(s) = \mathbf{1}, \qquad \mathbf{N}_0(s) = \mathbf{1}.$$

The identified $\mathbf{R}(s)$ was a 4^{th}-order BIBO stable transfer matrix, which corresponded to a 5^{th}-order plant system with one ORHP pole at 0.13.

Table 4.4: NRMSE for MIMO identification of actuators 3 & 4.

Identification Method	Training				Validation			
	Input		Output		Input		Output	
	C3	C4	C3	C4	C3	C4	C3	C4
Direct	0.2745	0.8594	0.6652	0.7851	0.2603	0.7637	0.8000	0.8794
Indirect	0.2466	0.3606	0.0202	0.0234	0.2812	0.9272	0.0261	0.0343
Dual-Youla	0.1603	0.2292	0.0217	0.0241	0.1656	0.2857	0.0275	0.0354

Table 4.5: %VAF for MIMO identification of actuators 3 & 4.

Identification Method	Training				Validation			
	Input		Output		Input		Output	
	C3	C4	C3	C4	C3	C4	C3	C4
Direct	93.06	34.42	54.23	53.35	93.25	41.47	36.45	22.90
Indirect	94.61	92.76	99.96	99.94	92.32	78.67	99.93	99.88
Dual-Youla	97.44	94.92	99.95	99.94	97.28	92.70	99.92	99.87

4.5.4 4-Actuator MIMO Identification

A PI controller with forward loop filters (FLF) was used to track a randomized load profile on all 4 actuators. The same controller was used on each actuator, where the P gain was set to 20, the I gain to 50, and the pole and zero frequencies of the FLF were set to 0.75 Hz and 2 Hz, respectively. This resulted in a 4×4 diagonal controller transfer matrix given by

$$\mathbf{C}(s) = \left(\frac{7.5s^2 + 144.2s + 235.6}{s^2 + 4.712s} \right) \mathbf{1}. \tag{4.17}$$

The data was down-sampled to a sample time of 0.1 seconds. The NRMS errors for the training and validation datasets for each method are shown in Tables 4.6 and 4.7, and the corresponding %VAF values are included in Tables 4.8 and 4.9. In the case of the direct and indirect methods, the simulated voltage signal is unstable. These identified models are unable to recover the voltage data. which is indicated by the "NaN" entries in the tables.

Table 4.6: NRMSE in training for 4-actuator MIMO identification.

Identification Method	Input				Output			
	C1	C2	C3	C4	C1	C2	C3	C4
Direct	NaN	NaN	NaN	NaN	0.7884	0.6197	0.8016	0.8002
Indirect	NaN	NaN	NaN	NaN	0.1266	0.1383	0.1578	0.1695
Dual-Youla	0.7112	0.7243	0.9073	0.9271	0.0808	0.0823	0.1054	0.1105

Table 4.7: NRMSE in validation for 4-actuator MIMO identification.

Identification	Input				Output			
Method	C1	C2	C3	C4	C1	C2	C3	C4
Direct	NaN	NaN	NaN	NaN	0.8824	0.7152	0.7481	0.8274
Indirect	NaN	NaN	NaN	NaN	0.1151	0.1334	0.1389	0.1276
Dual-Youla	0.7865	0.7832	0.8938	0.9516	0.0871	0.0958	0.1090	0.0975

Table 4.8: %VAF in training for 4-actuator MIMO identification.

Identification	Input				Output			
Method	C1	C2	C3	C4	C1	C2	C3	C4
Direct	NaN	NaN	NaN	NaN	47.36	60.85	68.17	49.52
Indirect	NaN	NaN	NaN	NaN	98.78	98.40	98.22	97.95
Dual-Youla	48.97	48.00	18.30	14.28	99.34	99.21	98.84	98.75

Table 4.9: %VAF in validation for 4-actuator MIMO identification.

Identification	Input				Output			
Method	C1	C2	C3	C4	C1	C2	C3	C4
Direct	NaN	NaN	NaN	NaN	34.74	52.07	44.39	38.72
Indirect	NaN	NaN	NaN	NaN	98.68	98.22	98.05	98.38
Dual-Youla	38.76	38.67	20.07	9.39	99.24	99.08	98.80	99.05

4.5.4.1 Direct Method

The direct method identified the plant as a 12^{th}-order stable transfer matrix.

4.5.4.2 Indirect Method

The indirect method identified a 4^{th}-order BIBO stable transfer matrix for $\mathbf{G}(s)$, which resulted in an identified $\mathbf{P}(s)$ that was 2^{nd}-order with two ORHP poles.

4.5.4.3 Dual-Youla Method

The initial plant was selected as $\mathbf{P}(s) = \mathbf{1}$, with coprime factors

$$\mathbf{M}_0(s) = \mathbf{1}, \qquad \mathbf{N}_0(s) = \mathbf{1}.$$

The controller, which has diagonal entries given by (4.17), was factorized as

$$\mathbf{X}_0(s) = \left(\frac{s^2 + 4.712s}{(s+30)^2}\right)\mathbf{1}, \qquad \mathbf{Y}_0(s) = \left(\frac{7.5s^2 + 144.2s + 235.6}{(s+30)^2}\right)\mathbf{1}.$$

The identified $\mathbf{R}(s)$ was a 4^{th}-order BIBO stable transfer matrix, which resulted in a 6^{th}-order BIBO stable plant system.

4.5.5 Discussion

In all of the SISO and MIMO cases presented in this chapter, the dual-Youla closed-loop system identification method is consistently able to produce the best results, both in terms of NRMSE and %VAF for both the output data and simulated control effort. The SISO and 2-actuator MIMO identified models are able to recreate the output data and control effort data to a high degree of accuracy. In the 4-actuator case, although the output data in validation is accurate, the simulated control effort is not. However, due to the layout of the SHM platform, the cross-talk between each half of the platform is minimal. Therefore, the SISO and 2-actuator dual-Youla identified models will be used for controller design in Part III.

Part III

Controller Synthesis

Chapter 5

Using the $\mathcal{H}_\infty$ Norm to Synthesize Optimal PI Gains

Although PI controllers are typically tuned by hand without regard to a system model, this chapter explores the synthesis of model-based "optimal" PI controllers. The synthesis methods attempt to minimize the closed-loop $\mathcal{H}_\infty$ norm using convex optimization subject to matrix inequality constraints. However, both methods involve bilinear matrix inequalities (BMIs) and thus the optimization problems are not convex. To deal with this, iteration and bisection method are used, along with initial stabilizing PI gains, to solve the problem locally or sub-optimally. Because the MTS controller includes a SISO PI controller for each actuator, the synthesis methods in this chapter are also SISO. Both the controller synthesis methods presented can either use the closed-loop system shown in Fig. 5.1 or that of Fig. 2.1. If using the configuration in Fig. 5.1, there are no weighting filters on disturbances or noise.

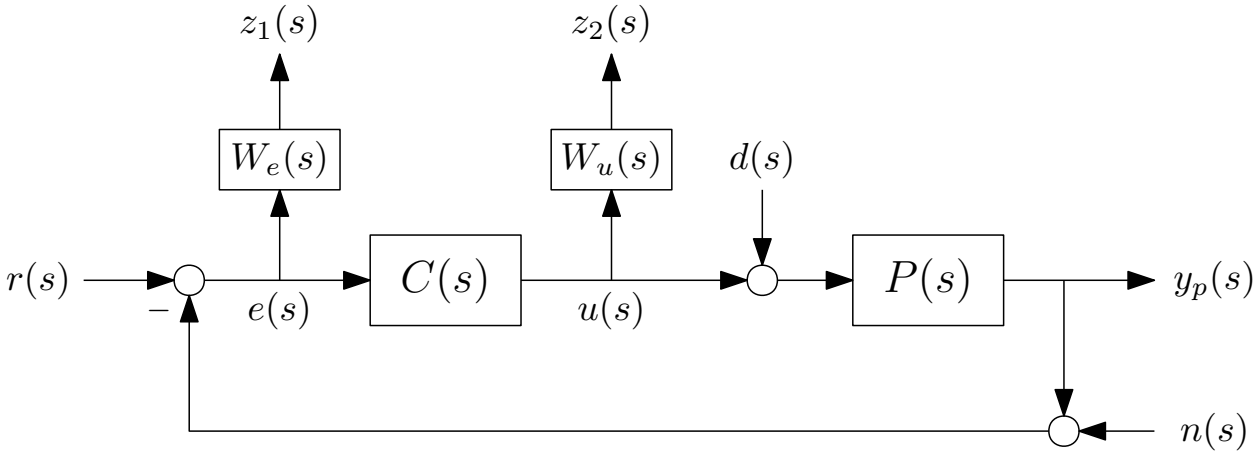


Figure 5.1: Block diagram for controller synthesis.

5.1 Starting Gains

To initialize either of the algorithms mentioned in this section, a set of initial stabilizing gains must be found. One way to find these gains is to select gains that are known to work on the real system. Another approach is to synthesize gains using the `MATLAB` PID tuner.

5.1.1 MATLAB PID Tuner

`MATLAB` includes a built-in PID tuner that generates PID gains based on a model of the system. The user can adjust the desired response manually using the application interface, or synthesize gains automatically from the command line using the function `pidtune`. The tuning objectives are to maintain closed-loop BIBO stability, and provide "adequate" performance and robustness [42]. Because PI control is used in fatigue testing, the `MATLAB` PID tuner is used to synthesize PI gains.

5.2 Static Output Feedback Method

A static output feedback controller is defined as $\mathbf{K} \in \mathbb{R}^{n_u \times n_y}$, where $\mathbf{u}(t) = \mathbf{K}\mathbf{y}(t)$ and the generalized plant is given by (2.26), (2.27), and (2.28) [43]. The static output feedback problem is non-convex due to the presence of a BMI constraint. Therefore, bisection method will be used to solve the problem sub-optimally.

To design $\mathbf{K}$ to be a PI controller, the input to the controller $\mathbf{y}(t)$ must be defined as the error augmented with the integral of the error. Therefore, it is necessary to define an "integral state," as in Section 2.4.3.1 [16, 17].

5.2.1 Integral State

The time rate of change of the integral state is set equal to the error $e(t)$, such that the state itself is the integral of the error. If disturbances and noise are weighted, the definition of the integral state is given by (2.29). If disturbances and noise are not weighted, the state $x_i(t)$ is defined by

$$\begin{aligned}\dot{x}_i(t) &= r(t) - (y_o(t) + n(t)) \\ &= r(t) - (\mathbf{C}_o\mathbf{x}_o(t) + \mathbf{D}_o\left(u(t) + d(t)\right) + n(t)) \\ &= r(t) - \mathbf{C}_o\mathbf{x}_o(t) - \mathbf{D}_o u(t) - \mathbf{D}_o d(t) - n(t).\end{aligned}$$

Then the controller input $\mathbf{y}(t)$ is given by

$$\mathbf{y}(t) = \begin{bmatrix} e(t) \\ x_i(t) \end{bmatrix}.$$

Defining the performance outputs as weighted tracking error and weighted control effort, $\mathbf{z}(s) = [W_e(s)e(s) \quad W_u(s)u(s)]^\top$ and the exogenous input is $\mathbf{w}(t) = [r(t) \quad d(t) \quad n(t)]^\top$. The closed-loop system from exogenous inputs $\mathbf{w}(t)$ to performance outputs $\mathbf{z}(t)$ is then represented by

$$\dot{\mathbf{x}}(t) = \underbrace{(\mathbf{A} + \mathbf{B}_2\bar{\mathbf{K}}\mathbf{C}_2)}_{\mathbf{A}_{\mathrm{CL}}}\mathbf{x}(t) + \underbrace{(\mathbf{B}_1 + \mathbf{B}_2\bar{\mathbf{K}}\mathbf{D}_{21})}_{\mathbf{B}_{\mathrm{CL}}}\mathbf{w}(t), \tag{5.1}$$

$$\mathbf{z}(t) = \underbrace{(\mathbf{C}_1 + \mathbf{D}_{12}\bar{\mathbf{K}}\mathbf{C}_2)}_{\mathbf{C}_{\mathrm{CL}}}\mathbf{x}(t) + \underbrace{(\mathbf{D}_{11} + \mathbf{D}_{12}\bar{\mathbf{K}}\mathbf{D}_{21})}_{\mathbf{D}_{\mathrm{CL}}}\mathbf{w}(t), \tag{5.2}$$

where $\bar{\mathbf{K}} = (\mathbf{1} - \mathbf{K}\mathbf{D}_{22})^{-1}\mathbf{K}$.

5.2.2 Generalized Plant

In the case that disturbances and noise are weighted, in addition to weighting the performance channels, the generalized plant including the integral state is given by (2.33), (2.34), (2.35). If disturbances and noise are not weighted, the generalized plant is described by the state-space realization given by

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} \mathbf{A}_o & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{B}_e\mathbf{C}_o & \mathbf{A}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_u & \mathbf{0} \\ -\mathbf{C}_o & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{A}}\mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{B}_o & \mathbf{0} \\ \mathbf{B}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & -\mathbf{D}_o & -\mathbf{1} \end{bmatrix}}_{\mathbf{B}_1}\mathbf{w}(t) + \underbrace{\begin{bmatrix} \mathbf{B}_o \\ -\mathbf{B}_e\mathbf{D}_o \\ \mathbf{B}_u \\ -\mathbf{D}_o \end{bmatrix}}_{\mathbf{B}_2}u(t)\,, \tag{5.3}$$

$$\mathbf{z}(t) = \underbrace{\begin{bmatrix} -\mathbf{D}_e\mathbf{C}_o & \mathbf{C}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}_u & \mathbf{0} \end{bmatrix}}_{\mathbf{C}_1}\mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{D}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{D}_{11}}\mathbf{w}(t) + \underbrace{\begin{bmatrix} -\mathbf{D}_e\mathbf{D}_o \\ \mathbf{D}_u \end{bmatrix}}_{\mathbf{D}_{12}}u(t)\,, \tag{5.4}$$

$$y(t) = \underbrace{\begin{bmatrix} -\mathbf{C}_o & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}}_{\mathbf{C}_2}\mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{1} & -\mathbf{D}_o & -\mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{D}_{21}}\mathbf{w}(t) + \underbrace{\begin{bmatrix} -\mathbf{D}_o \\ \mathbf{0} \end{bmatrix}}_{\mathbf{D}_{22}}u(t)\,, \tag{5.5}$$

where $\mathbf{x}(t) = \begin{bmatrix}\mathbf{x}_o^\top(t) & \mathbf{x}_e^\top(t) & \mathbf{x}_u^\top(t) & x_i(t)\end{bmatrix}^\top$.

5.2.3 Synthesis Method

The generalized plant equations are substituted into the closed-loop system equations given by (5.1) and (5.2). Because the closed-loop state-space matrices contain the design variable $\bar{\mathbf{K}}$, and $\mathbf{P}$ and $\mathbf{Q}$ are also design variables, (2.24) and (2.25) are actually bilinear matrix inequalities in the variables $\mathbf{P}$ or $\mathbf{Q}$, $\bar{\mathbf{K}}$, and γ. As such, they can only be solved sub-optimally. Since (2.25) is equivalent to (2.24) through a congruence transformation with $\mathbf{Q} = \mathbf{P}^{-1}$, this imposes the additional constraint $\mathbf{PQ} = \mathbf{1}$ [43]. The complete synthesis method can be summarized as follows.

1. Let $\mathbf{P}_0 = \mathbf{Q}_0 = \mathbf{1}$, and set $k = 0$. Given a set of stabilizing PI gains, compute the closed-loop $\mathcal{H}_\infty$ norm. Multiply this value by a safety factor of 1.5, then set this as the upper bound on γ_d, γ_u. Set the lower bound, γ_ℓ, to zero. Then the starting γ_d is equal to $(\gamma_u + \gamma_\ell)/2$.

2. Solve for $\mathbf{P}_{k+1} = \mathbf{P}_{k+1}^\top \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{Q}_{k+1} = \mathbf{Q}_{k+1}^\top \in \mathbb{R}^{n_x \times n_x}$ that minimize $\operatorname{trace}(\mathbf{Q}_k\mathbf{P}_{k+1} + \mathbf{P}_k\mathbf{Q}_{k+1})$ such that $\mathbf{P}_{k+1} > 0$, $\mathbf{Q}_{k+1} > 0$, $\gamma < \gamma_d$,

$$\begin{bmatrix} \mathbf{N}_o & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}^\top \begin{bmatrix} \mathbf{P}_{k+1}\mathbf{A} + \mathbf{A}^\top\mathbf{P}_{k+1} & \mathbf{P}_{k+1}\mathbf{B}_1 & \mathbf{C}_1^\top \\ \star & -\gamma\mathbf{1} & \mathbf{D}_{11}^\top \\ \star & \star & -\gamma\mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{N}_o & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} < 0\,, \tag{5.6}$$

$$\begin{bmatrix} \mathbf{N}_c & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}^\top \begin{bmatrix} \mathbf{A}\mathbf{Q}_{k+1} + \mathbf{Q}_{k+1}\mathbf{A}^\top & \mathbf{Q}_{k+1}\mathbf{C}_1^\top & \mathbf{B}_1 \\ \star & -\gamma\mathbf{1} & \mathbf{D}_{11} \\ \star & \star & -\gamma\mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{N}_c & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} < 0\,, \tag{5.7}$$

$$\begin{bmatrix} \mathbf{P}_{k+1} & \mathbf{1} \\ \star & \mathbf{Q}_{k+1} \end{bmatrix} \geq 0\,, \tag{5.8}$$

   where $\mathcal{R}(\mathbf{N}_o) = \mathcal{N}\left(\begin{bmatrix}\mathbf{C}_2 & \mathbf{D}_{21}\end{bmatrix}\right)$ and $\mathcal{R}(\mathbf{N}_c) = \mathcal{N}\left(\begin{bmatrix}\mathbf{B}_2^\top & \mathbf{D}_{12}^\top\end{bmatrix}\right)$.

3. If $|\operatorname{tr}(\mathbf{Q}_k\mathbf{P}_{k+1} + \mathbf{P}_k\mathbf{Q}_{k+1}) - 2n_x|$ is less than a tolerance, proceed to Step 4. If not, set $k = k + 1$ and return to Step 2.

4. Fix $\mathbf{P} = \mathbf{P}_{k+1}$ and solve for $\bar{\mathbf{K}}$ that minimizes γ such that $\gamma < \gamma_d$ and

$$\begin{bmatrix} \mathbf{P}(\mathbf{A} + \mathbf{B}_2\bar{\mathbf{K}}\mathbf{C}_2) + (\mathbf{A} + \mathbf{B}_2\bar{\mathbf{K}}\mathbf{C}_2)^\top\mathbf{P} & \mathbf{P}(\mathbf{B}_1 + \mathbf{B}_2\bar{\mathbf{K}}\mathbf{D}_{21}) & (\mathbf{C}_1 + \mathbf{D}_{12}\bar{\mathbf{K}}\mathbf{C}_2)^\top \\ \star & -\gamma\mathbf{1} & (\mathbf{D}_{11} + \mathbf{D}_{12}\bar{\mathbf{K}}\mathbf{D}_{21})^\top \\ \star & \star & -\gamma\mathbf{1} \end{bmatrix} < 0\,. \tag{5.9}$$

Perform bisection on γ_d, then return to Step 2.

5. Repeat until γ_u and γ_ℓ are within a specified tolerance of each other.

6. Recover the controller by $\mathbf{K} = \bar{\mathbf{K}}(\mathbf{1} + \mathbf{D}_{22}\bar{\mathbf{K}})^{-1}$.

Equations (5.6) and (5.7) are equivalent forms of (2.24) and (2.25) found using the projection lemma, while (5.8) is an LMI included to enforce the constraint $\mathbf{PQ} = \mathbf{1}$ [43].

5.3 Iterative Method

The "iterative method" aims to find a set of PI gains using (2.24) from the Bounded Real Lemma, which is discussed in Section 2.4.2. Once a PI controller structure has been specified, the optimization problem is no longer convex due to the presence of a BMI constraint. The problem will be solved locally through the use of iteration and bisection method.

5.3.1 PI Controller

A PI controller, which is represented by the transfer function [44]

$$C(s) = k_p + \frac{k_i}{s}\,,$$

has corresponding state-space matrices given by

$$\mathbf{A}_c = 0\,, \qquad \mathbf{B}_c = 1\,, \qquad \mathbf{C}_c = k_i\,, \qquad \mathbf{D}_c = k_p\,. \tag{5.10}$$

Combining this controller with the generalized plant yields the following closed-loop state-space realization,

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} \mathbf{A} + \mathbf{B}_2\mathbf{D}_c\mathbf{C}_2 & \mathbf{B}_2\mathbf{C}_c \\ \mathbf{B}_c\mathbf{C}_2 & \mathbf{A}_c \end{bmatrix}}_{\mathbf{A}_{\mathrm{CL}}} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{B}_1 + \mathbf{B}_2\mathbf{D}_c\mathbf{D}_{21} \\ \mathbf{B}_c\mathbf{D}_{21} \end{bmatrix}}_{\mathbf{B}_{\mathrm{CL}}} \mathbf{w}(t)\,, \tag{5.11}$$

$$\mathbf{z}(t) = \underbrace{\begin{bmatrix} \mathbf{C}_1 + \mathbf{D}_{12}\mathbf{D}_c\mathbf{C}_2 & \mathbf{D}_{12}\mathbf{C}_c \end{bmatrix}}_{\mathbf{C}_{\mathrm{CL}}} \mathbf{x}(t) + \underbrace{(\mathbf{D}_{11} + \mathbf{D}_{12}\mathbf{D}_c\mathbf{D}_{21})}_{\mathbf{D}_{\mathrm{CL}}} \mathbf{w}(t)\,. \tag{5.12}$$

Note that the design variables k_p and k_i are contained exclusively in $\mathbf{C}_c$ and $\mathbf{D}_c$.

5.3.2 Generalized Plant

If disturbances and noise are weighted, the generalized plant is given by (2.30), (2.31), (2.32). Otherwise, the generalized plant is described by the state-space realization given by

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} \mathbf{A}_o & \mathbf{0} & \mathbf{0} \\ -\mathbf{B}_e\mathbf{C}_o & \mathbf{A}_e & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_u \end{bmatrix}}_{\mathbf{A}} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{B}_o & \mathbf{0} \\ \mathbf{B}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{B}_1} \mathbf{w}(t) + \underbrace{\begin{bmatrix} \mathbf{B}_o \\ -\mathbf{B}_e\mathbf{D}_o \\ \mathbf{B}_u \end{bmatrix}}_{\mathbf{B}_2} u(t)\,, \tag{5.13}$$

$$\mathbf{z}(t) = \underbrace{\begin{bmatrix} -\mathbf{D}_e\mathbf{C}_o & \mathbf{C}_e & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}_u \end{bmatrix}}_{\mathbf{C}_1} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{D}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{D}_{11}} \mathbf{w}(t) + \underbrace{\begin{bmatrix} -\mathbf{D}_e\mathbf{D}_o \\ \mathbf{D}_u \end{bmatrix}}_{\mathbf{D}_{12}} u(t)\,, \tag{5.14}$$

$$y(t) = \underbrace{\begin{bmatrix} -\mathbf{C}_o & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{C}_2} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \mathbf{1} & -\mathbf{D}_o & -\mathbf{1} \end{bmatrix}}_{\mathbf{D}_{21}} \mathbf{w}(t) + \underbrace{(-\mathbf{D}_o)}_{\mathbf{D}_{22}} u(t)\,, \tag{5.15}$$

where $\mathbf{y}(t)$ is the error, and $\mathbf{x}(t) = \begin{bmatrix} \mathbf{x}_o^\top(t) & \mathbf{x}_e^\top(t) & \mathbf{x}_u^\top(t) \end{bmatrix}^\top$.

5.3.3 Synthesis Method

The generalized plant is substituted into the closed-loop expression given by (5.11) and (5.12). Next, these closed-loop state-space matrices are used in the matrix inequality given by (2.24). Typically, this formulation could be solved using convex optimization with LMI constraints. However, because the structure of the controller is specified as PI in this case, (2.24) is a BMI. At least one of the design variables k_p and k_i is present in each of the closed-loop state-space matrices, and $\mathbf{P}$ is also a design variable. In the iterative method, the algorithm alternates between fixing the gains k_p and k_i and solving for a feasible $\mathbf{P}$, then fixing $\mathbf{P}$ and optimizing over the gains. The synthesis method can be summarized as follows.

1. Given a set of stabilizing PI gains, fix k_p and k_i and use their values to compute the closed-loop $\mathcal{H}_\infty$ norm. Set $\gamma_u = 1.5\,\|\boldsymbol{\mathcal{G}}_0\|_\infty$. Set $\gamma_\ell = 0$. Then $\gamma_d = (\gamma_u + \gamma_\ell)/2$.

2. Solve for $\mathbf{P} = \mathbf{P}^\top \in \mathbb{R}^{(n_x+n_c)\times(n_x+n_c)}$ such that $\gamma < \gamma_d$, $\mathbf{P} > 0$,

$$\begin{bmatrix} \mathbf{P}\mathbf{A}_{\mathrm{CL}} + \mathbf{A}_{\mathrm{CL}}^\top\mathbf{P} & \mathbf{P}\mathbf{B}_{\mathrm{CL}} & \mathbf{C}_{\mathrm{CL}}^\top \\ \star & -\gamma\mathbf{1} & \mathbf{D}_{\mathrm{CL}}^\top \\ \star & \star & -\gamma\mathbf{1} \end{bmatrix} < 0\,,$$

   where $\mathbf{A}_{\mathrm{CL}}$, $\mathbf{B}_{\mathrm{CL}}$, $\mathbf{C}_{\mathrm{CL}}$, and $\mathbf{D}_{\mathrm{CL}}$ are given in (5.11) and (5.12). Fix $\mathbf{P}$ then proceed to Step 3.

3. Solve for k_p and k_i that minimize γ such that $\gamma < \gamma_d$ and

$$\begin{bmatrix} \mathbf{P}\mathbf{A}_{\mathrm{CL}} + \mathbf{A}_{\mathrm{CL}}^{\top}\mathbf{P} & \mathbf{P}\mathbf{B}_{\mathrm{CL}} & \mathbf{C}_{\mathrm{CL}}^{\top} \\ \star & -\gamma\mathbf{1} & \mathbf{D}_{\mathrm{CL}}^{\top} \\ \star & \star & -\gamma\mathbf{1} \end{bmatrix} < 0\,.$$

   Fix k_p and k_i. Perform bisection on γ_d, then return to Step 2.

4. Repeat until γ_u and γ_ℓ are within a specified tolerance of each other.

5.4 Results

The PI controller synthesis methods were applied to the analytical model of the SHM platform, pictured in Fig. 1.1, from Chapter 3 and the identified model of actuator 3 from Chapter 4. In both cases, the starting gains for the static output feedback and iterative $\mathcal{H}_\infty$ algorithms are those synthesized by the `MATLAB` PID tuner. The controller gains for each method and corresponding closed-loop simulation results are presented.

5.4.1 Analytical Model

The analytical model of the SHM platform has a transfer function given by (3.16),

$$P(s) = \frac{L_m(s)}{V(s)} = \frac{1.2465\text{e}07s + 7.21\text{e}11}{s^3 + 8.623\text{e}04s^2 + 5.471\text{e}07s + 5.765\text{e}08},$$

A unit conversion was applied such that the synthesized gains would be in the proper units of V/(% of fullscale load/10), as explained in Section 3.4.

5.4.1.1 MATLAB PID Tuner Method

For the PID tuner, the resulting gains were $k_p = 0.95$ and $k_i = 28.54$, and the plots are shown in Fig. 5.2.

5.4.1.2 Static Output Feedback Method

Using the generalized plant given by (5.3), (5.4), (5.5), the static output feedback method synthesized gains of $k_p = 488$ and $k_i = 2.367\text{e}04$. The plots are included in Fig. 5.3.

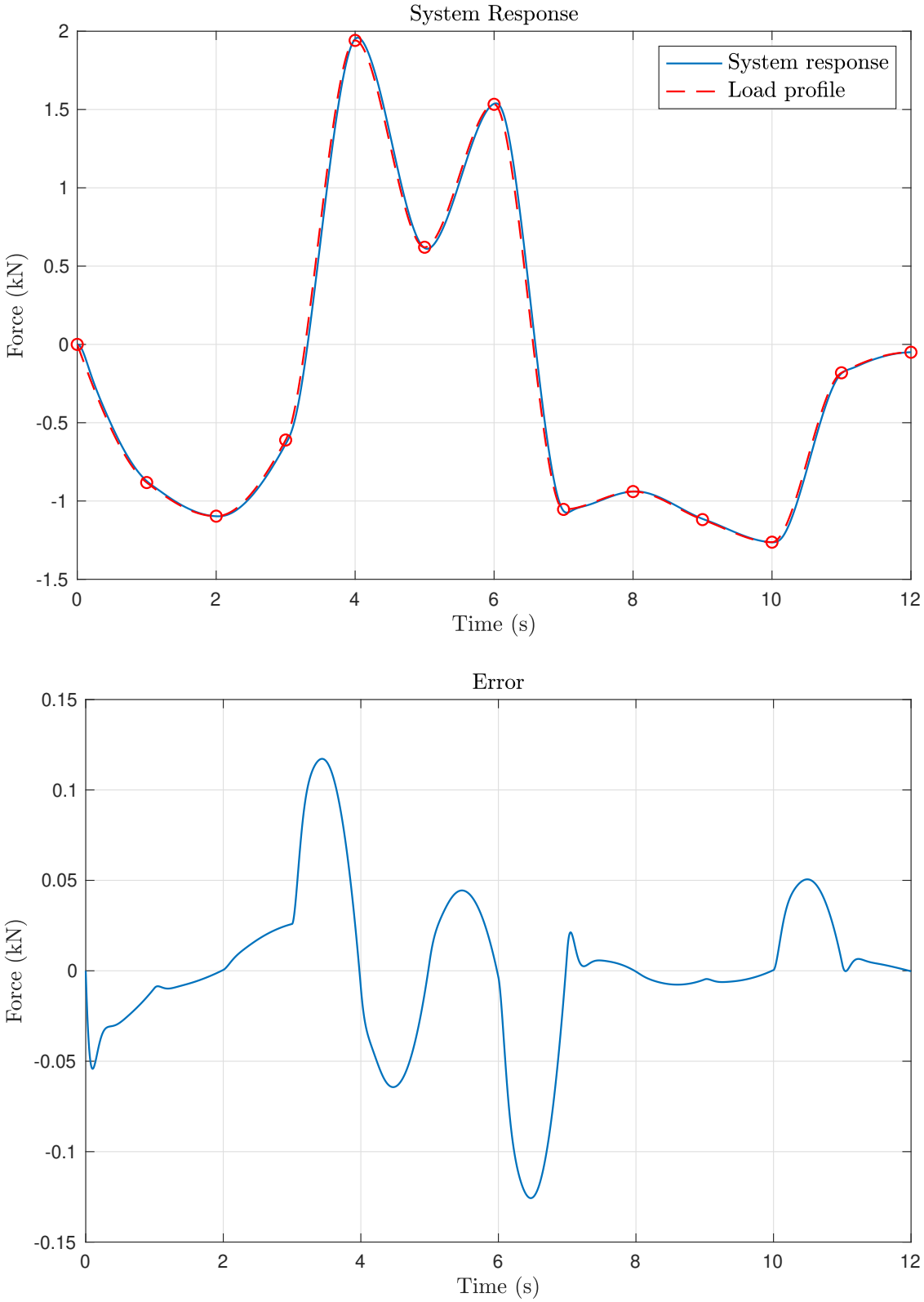


Figure 5.2: Simulation results for PID tuner applied to the analytical model.

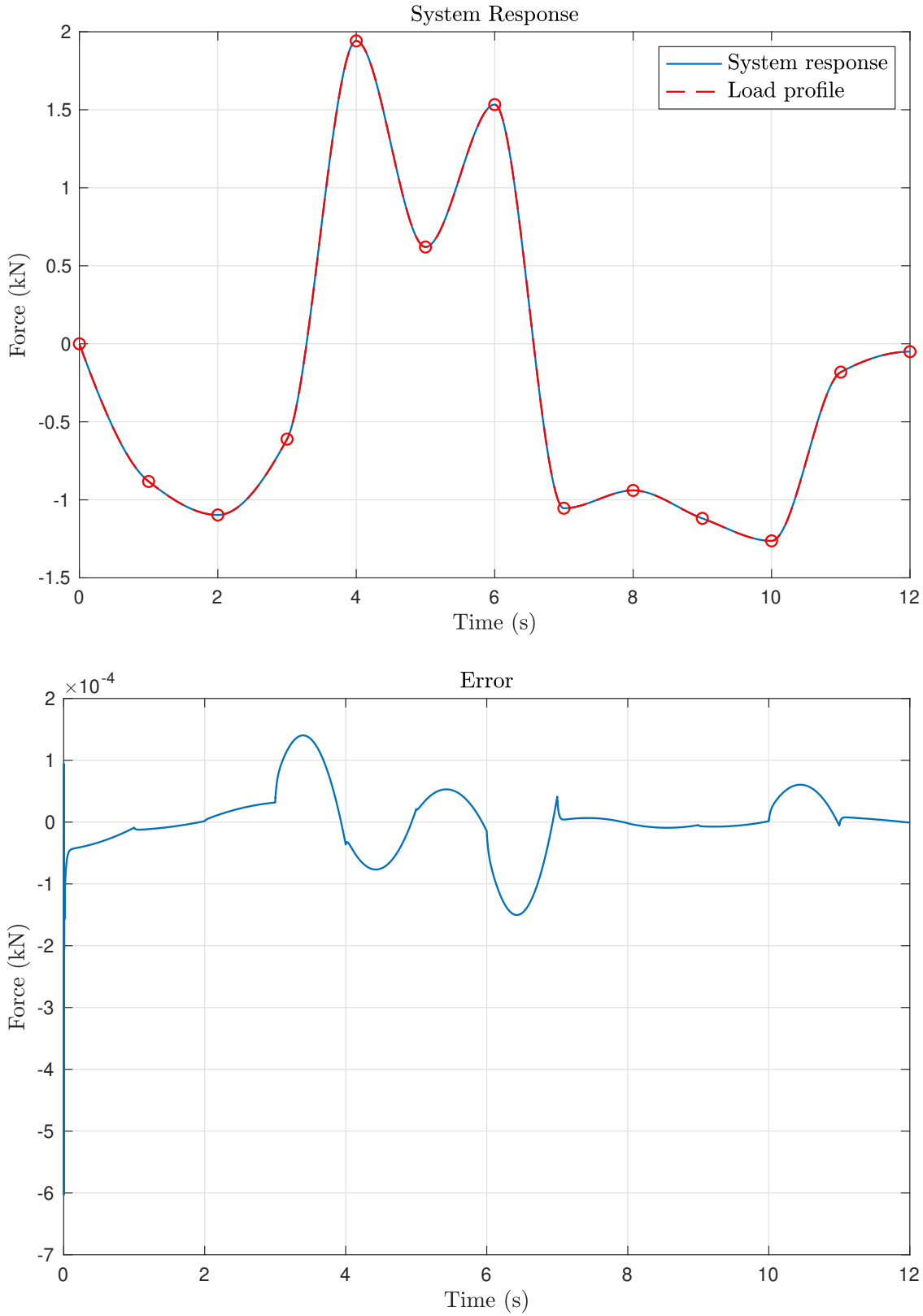


Figure 5.3: Simulation results for static output feedback applied to the analytical model.

5.4.1.3 Iterative Method

The iterative method yielded $k_p = 36.0$ and $k_i = 151.1$, with the generalized plant defined by (5.13), (5.14), (5.15). Figure 5.4 shows the closed-loop simulation results.

5.4.2 Identified Model

Next, the PI controller synthesis algorithms were applied to the identified model of actuator 3 on the SHM platform, given by (4.14),

$$P(s) = \frac{0.01142s^3 + 1.583s^2 + 29.91s + 23.22}{s^3 + 22.82s^2 + 24.9s + 0.04249}.$$

This transfer function is in the proper units, so the gains do not have to be converted.

5.4.2.1 MATLAB PID Tuner Method

The `MATLAB` PID tuner outputted gains of $k_p = 0.0023$ and $k_i = 1.1992\text{e-}05$. This controller is stabilizing but unable to provide tracking, as shown in Fig. 5.5.

5.4.2.2 Static Output Feedback Method

The static output feedback method, using a generalized plant defined by (5.3), (5.4), (5.5), synthesized gains of $k_p = 179.1$ and $k_i = 0.077$. Results are included in Fig. 5.6.

5.4.2.3 Iterative Method

For the identified model, a generalized plant given by (2.30), (2.31), (2.32) was used in the iterative $\mathcal{H}_\infty$ algorithm. This yielded gains of $k_p = 46.3$ and $k_i = 106.3$, with the closed-loop simulation results shown in Fig. 5.7.

5.4.3 Discussion

Both the static output feedback and iterative methods are able to achieve better tracking results than the `MATLAB` PID tuner, using both the analytical and identified models. Although the static output feedback controllers result in lower tracking errors than the PI controllers synthesized using the iterative method, the iterative method produces more realistic controller gains. The gains produced by the iterative method are safe to implement and are similar to gains typically used on the SHM platform. The static output feedback gains, on the other hand, are too high to implement. Therefore, based on the results in this chapter, the iterative method is recommended for synthesizing initial PI gains.

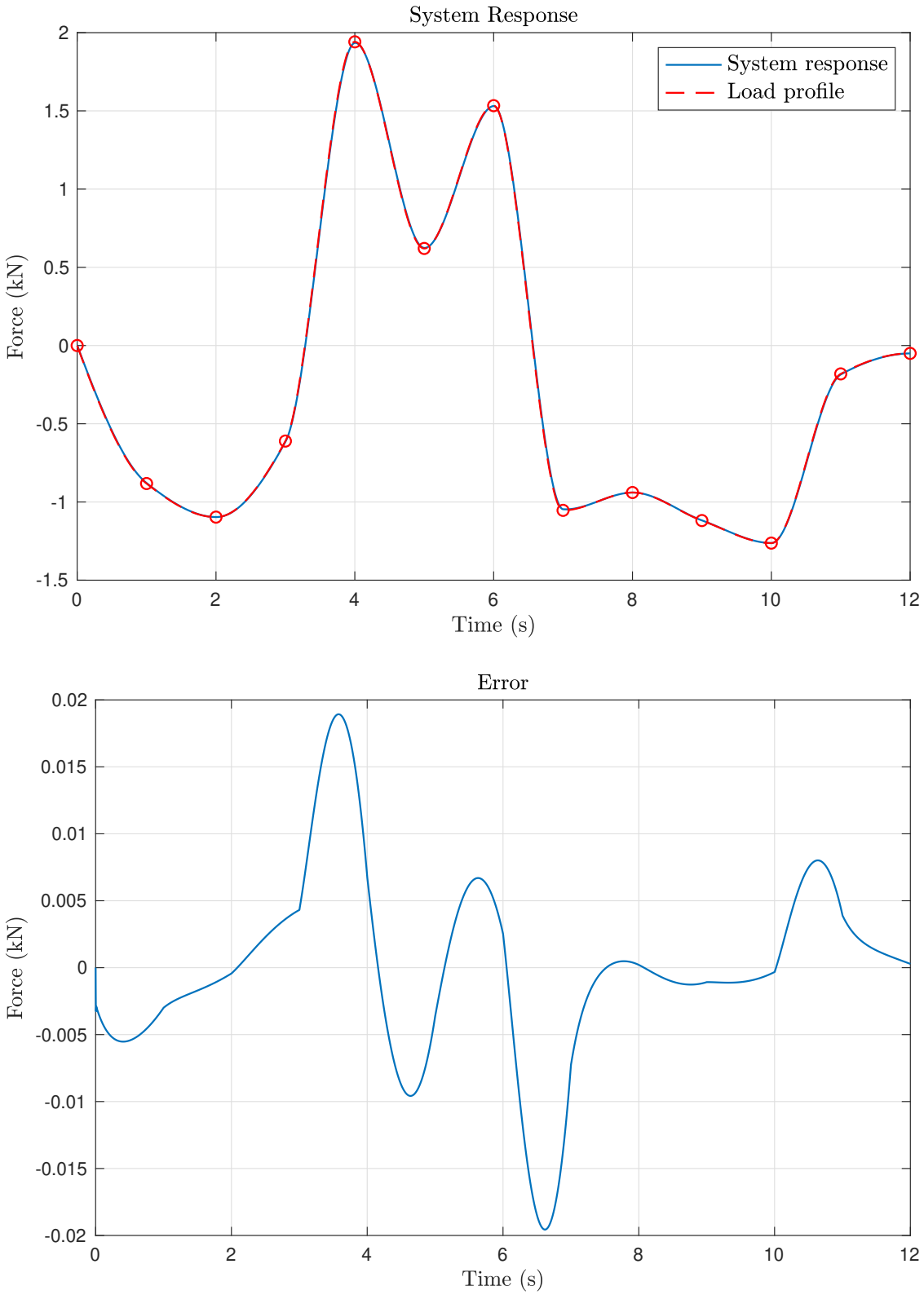


Figure 5.4: Simulation results for iterative method applied to the analytical model.

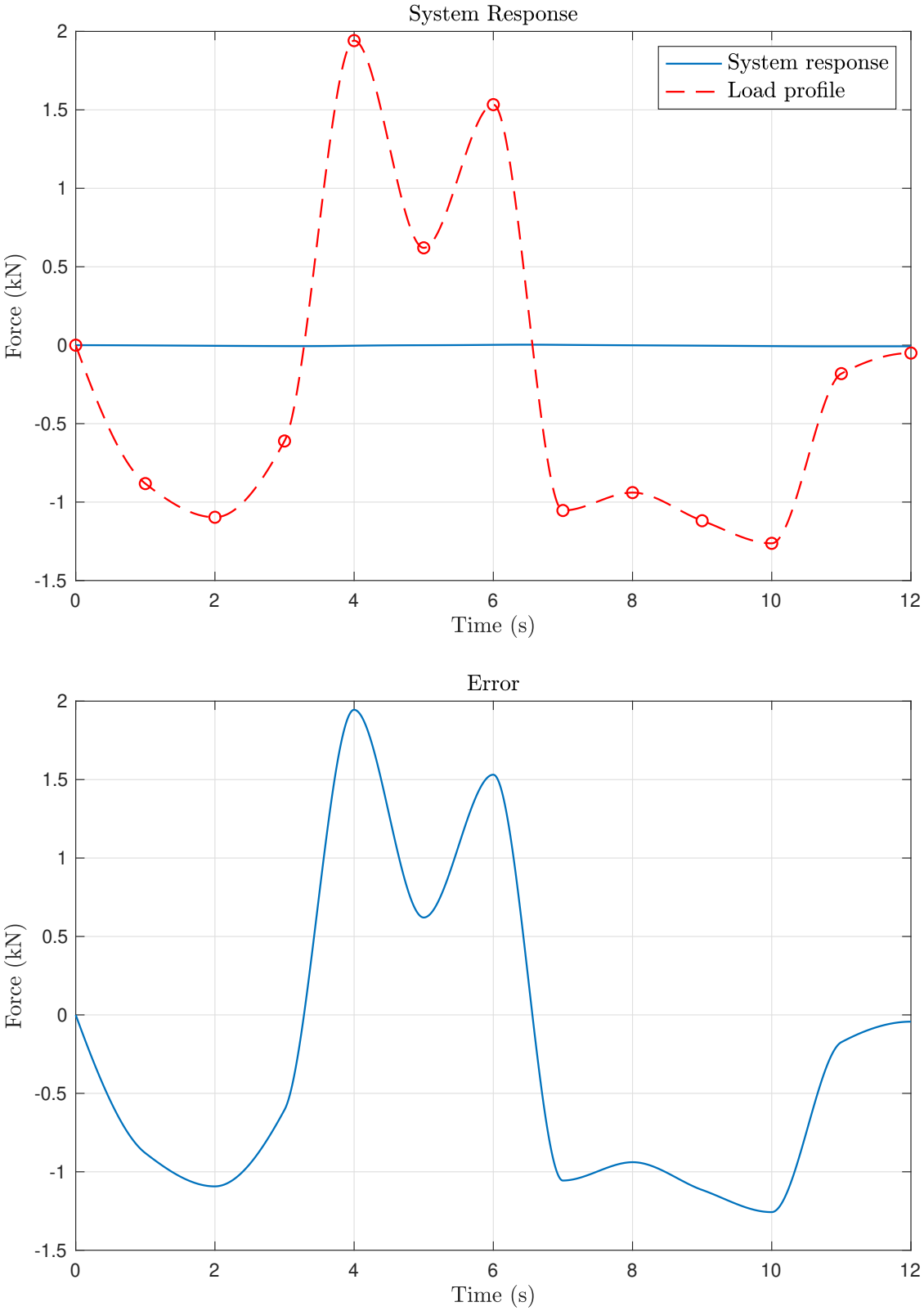


Figure 5.5: Simulation results for PID tuner applied to the identified model.

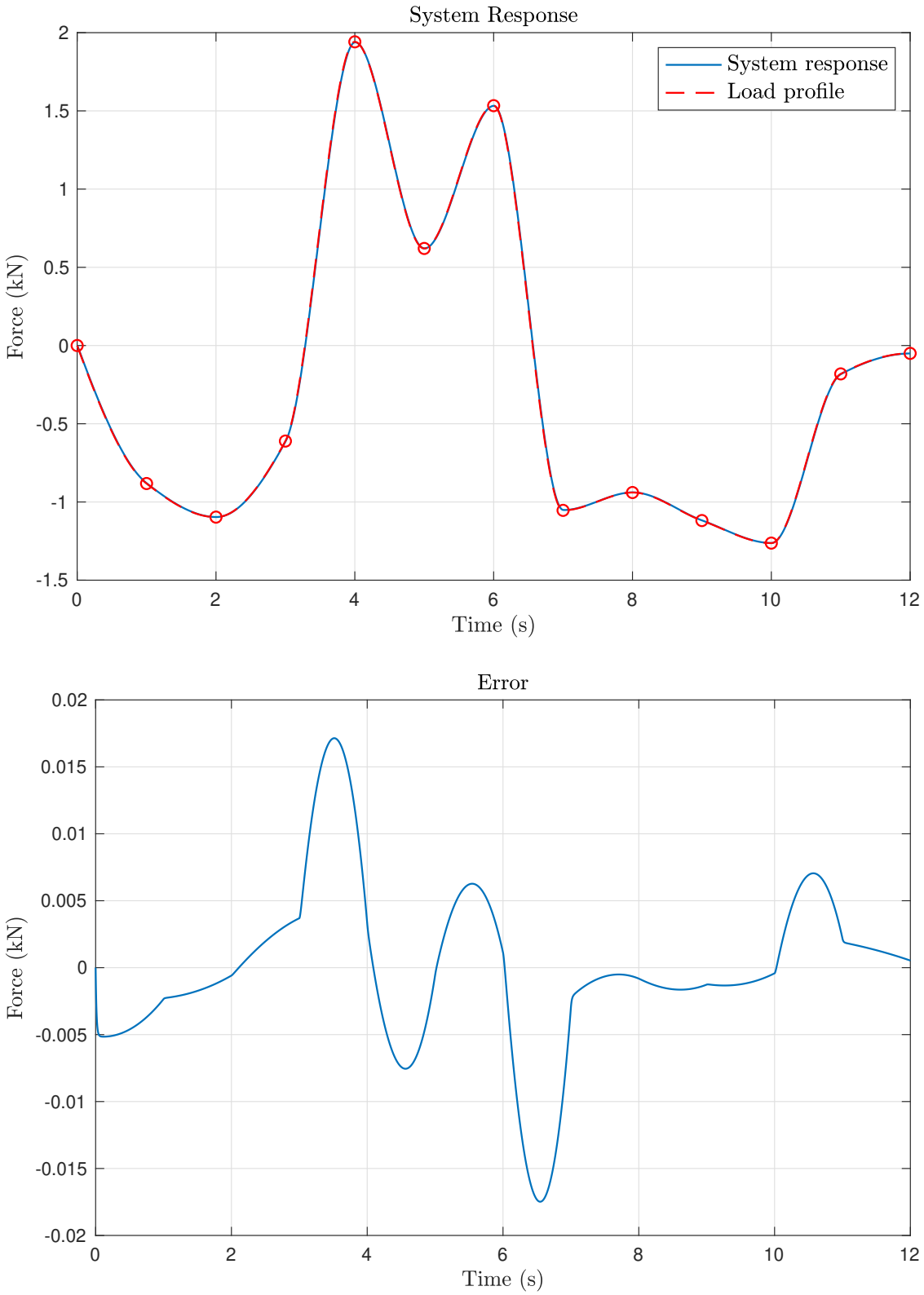


Figure 5.6: Simulation results for static output feedback applied to the identified model.

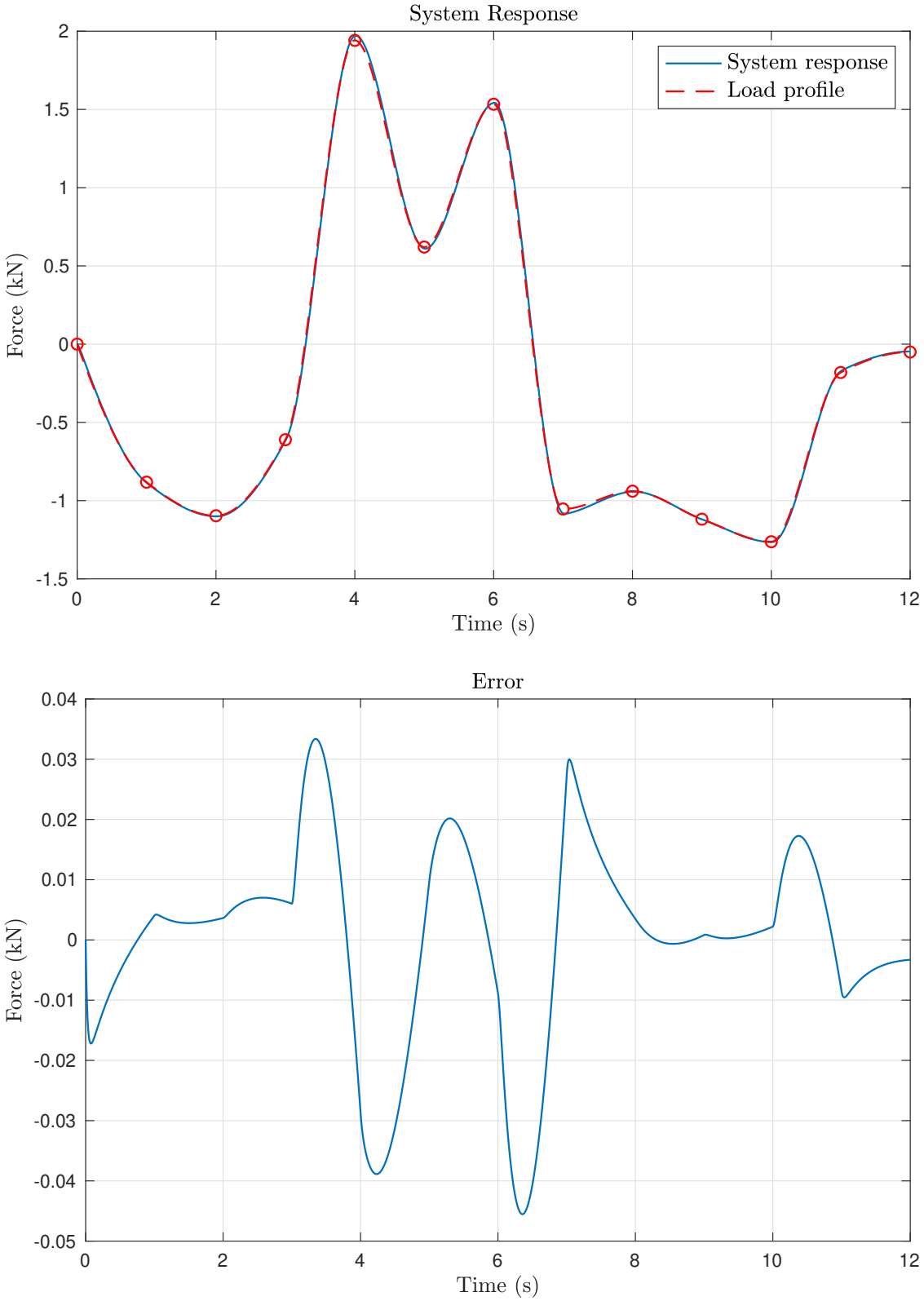


Figure 5.7: Simulation results for iterative method applied to the identified model.

Chapter 6

SISO Two Degree-of-Freedom Control

Two degree-of-freedom (2DOF) controllers are controllers that include feedback and feedforward, meaning that they act on the error as well as the reference signal [15]. In the SISO case, 2DOF controllers can be designed by independently designing the feedback controller and the feedforward controller. The feedback controller can be an $\mathcal{H}_\infty$ controller or a PI controller, for example. The feedforward controller can be designed via an inversion-based feedforward controller synthesis method.

6.1 Feedback Control

Although feedback controllers can be designed in a number of ways, this thesis will focus on PI and $\mathcal{H}_\infty$ controllers. The PI controller may be tuned by hand or designed using one of the methods presented in Chapter 5. If employing an $\mathcal{H}_\infty$ control scheme, controllers can be designed via the synthesis method in Chapter 7 using the 1DOF generalized plants given in Section 2.4.3. The $\mathcal{H}_\infty$ controller can be designed with or without an integrator.

6.2 Feedforward Control

In theory, if a system model is 100% accurate and there are no disturbances or noise, then perfect tracking can be achieved using a feedforward controller that is the inverse of the system model [45]. Mathematically, this results in an exact cancellation of transfer function poles and zeros such that the output equals the reference. In reality, however, this type of feedforward control must be implemented alongside a feedback controller, as shown in Fig. 6.1, to account for model uncertainty. Inversion-based feedforward control has been shown to improve tracking performance when implemented in conjunction with feedback control [45]. Although feedback control, and specifically PI control, is used extensively

in fatigue testing, feedforward control is not. This thesis presents a method for generating inversion-based feedforward controllers that use an approximate inverse of the system model.

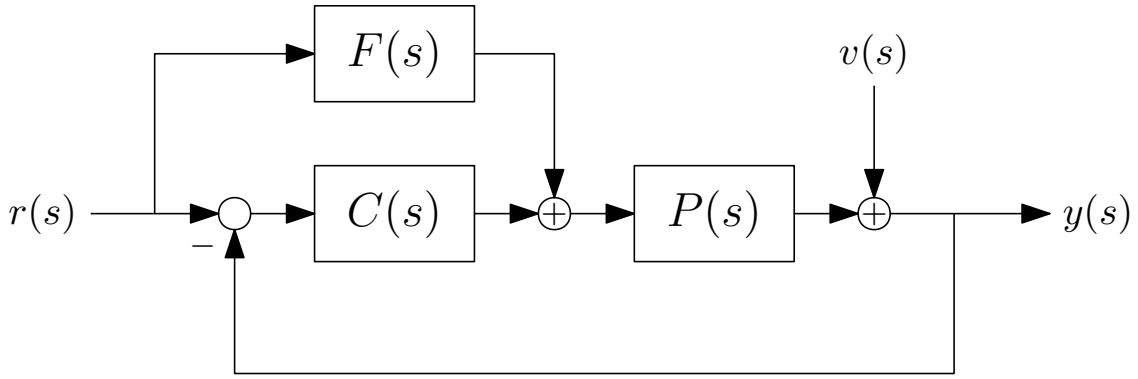


Figure 6.1: Feedforward-feedback control architecture.

Inversion-based feedforward controllers are only recommended for the SISO case. Although it is straightforward to compute an approximate inverse for a transfer function, the analogous process for transfer matrices is not so clear-cut. When attempting to synthesize feedforward controllers for MIMO systems, the recommended approach is to use a 2DOF $\mathcal{H}_\infty$ controller formulation, which is discussed in Chapter 7.

6.2.1 Order Reduction

Because the MTS controller has a limited amount of computational power, it is often necessary to implement reduced-order controllers if the initial models are relatively high order. In the case of feedforward control, rather than reduce-ordering the synthesized controller, the system transfer function should be reduce-ordered to the desired order before performing the inversion. As explained in Section 2.5.1, order reduction can be performed by manually eliminating higher modes, or using the `balred` command in `MATLAB`.

6.2.2 Mirroring Non-Minimum Phase Zeros

Inverting a transfer function is as simple as switching the numerator and the denominator. If, however, the transfer function is non-minimum phase (NMP), its ORHP zeros will yield ORHP poles when performing the inversion, rendering the inverted transfer function unstable. Therefore, any NMP zeros must first be "mirrored" into the OLHP before performing the inversion to ensure that the resulting transfer function is BIBO stable [46]. Although mirroring alters the phase of the transfer function, the gain is unchanged.

As an example, a reduced-order identified model for actuator 2 on the SHM rig at NRC obtained using the direct system identification method is given by

$$G(s) = \frac{-0.2417s^2 + 0.1289s + 103.1}{s^2 + 95.15s + 13.92},$$

which has one NMP zero at 20.92. The inverse of $G(s)$ is given by

$$G^{-1}(s) = \frac{4.138s^2 + 393.7s + 57.61}{s^2 - 0.5333s - 426.7},$$

which is unstable. However, if the NMP zero at 20.92 is first mirrored into the OLHP yielding an MP zero at -20.92, the inverse is then

$$F(s) = \frac{4.138s^2 + 393.7s + 57.61}{s^2 + 41.32s + 426.7}.$$

The transfer function $F(s)$ is BIBO stable and can therefore be used for feedforward control. The Bode diagrams for $G(s)$, $G^{-1}(s)$, and $F(s)$ are shown in Fig. 6.2. Despite the fact that the phase of $G(s)$ and $F(s)$ vary, the gain of $F(s)$ perfectly matches the gain of $G(s)$.

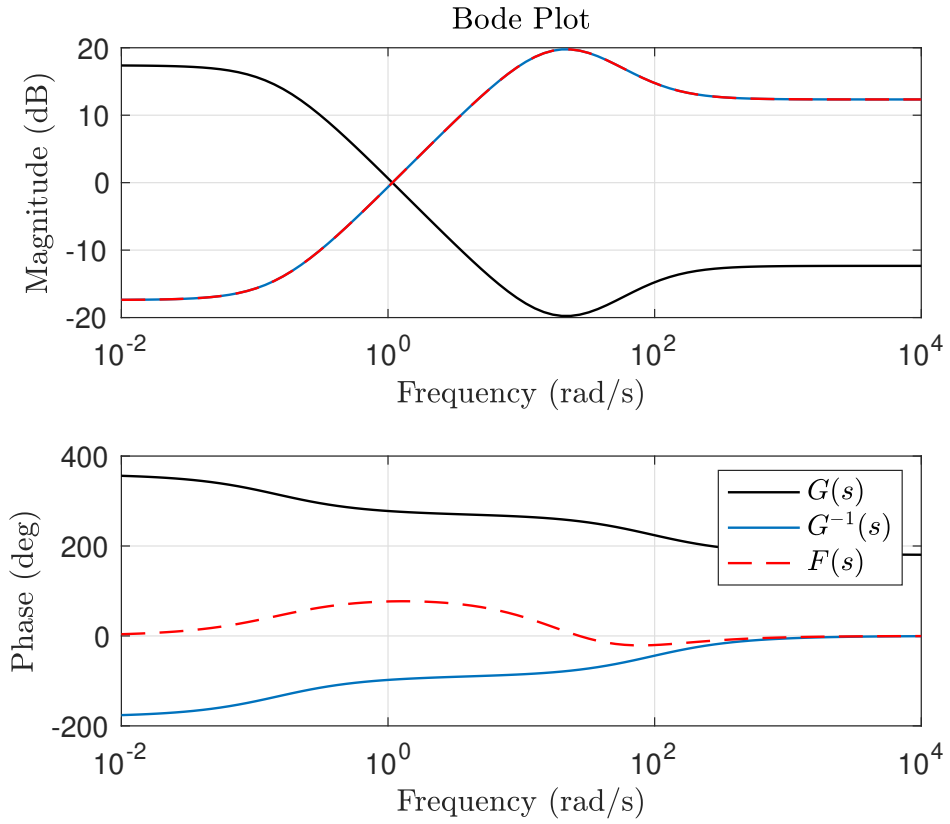


Figure 6.2: Bode diagram comparison of plant and feedforward controllers.

6.2.3 Inverting a State-Space System

Computing inversion-based feedforward controllers involves taking the inverse of a dynamic system. This can be done by inverting a transfer function/matrix using `MATLAB`, or using the state-space formulation in the following lemma.

Lemma 6.1 (Inverse of a Square System [47])**.** *Consider a square causal linear time-invariant (LTI) system $\boldsymbol{\mathcal{G}}$ with minimal state-space realization given by*

$$\begin{aligned}\mathbf{x}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t),\\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t),\end{aligned}$$

where $\mathbf{x}(t) \in \mathbb{R}^{n_x}$, $\mathbf{u}(t) \in \mathbb{R}^{n_u}$, $\mathbf{y}(t) \in \mathbb{R}^{n_y}$, $n_u = n_y$, and it is assumed that $\mathbf{D}$ is invertible. A minimal state-space realization of the inverse of $\boldsymbol{\mathcal{G}}$ is given by

$$\dot{\mathbf{x}}(t) = \underbrace{\left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)}_{\mathbf{A}_i}\mathbf{x}(t) + \underbrace{\mathbf{B}\mathbf{D}^{-1}}_{\mathbf{B}_i}\mathbf{y}(t), \tag{6.1}$$

$$\mathbf{u}(t) = \underbrace{-\mathbf{D}^{-1}\mathbf{C}}_{\mathbf{C}_i}\mathbf{x}(t) + \underbrace{\mathbf{D}^{-1}}_{\mathbf{D}_i}\mathbf{y}(t). \tag{6.2}$$

The matrix $\mathbf{D}$ being invertible indicates that the system has non-zero feedthrough and the transfer function is biproper, meaning that the relative degree is zero. All physical systems are proper, however, only biproper systems can be inverted. If a strictly proper transfer function is inverted, the resulting system will be improper. Improper transfer functions indicate non-causality, which means present outputs depend on future inputs. Fortunately, all of the identified models presented in Chapter 4 are biproper. In the case that a system is strictly proper, the numerator should be multiplied by factors of $(\tau s + 1)$ until the relative degree of the system is zero. The coefficient τ is selected to be small enough that its behaviour is non-dominant.

The above lemma can be used to compute inverses of a transfer function. First, the transfer function (with NMP zeros mirrored into the OLHP) can be converted to state-space using the `ss` function in `MATLAB`. Then equations (6.1) and (6.2) can be used to compute the state-space realization of the inverted system. The inverted state-space system can then be converted back to a transfer function using the `tf` command in `MATLAB`.

6.2.4 Synthesis Method

Synthesizing a SISO inversion-based feedforward controller involves taking an approximate (BIBO stable) inverse of the system transfer function. The algorithm is as follows.

1. Reduce-order the transfer function, if desired.
2. Mirror any NMP from the ORHP into the OLHP.
3. Invert the transfer function by flipping the numerator and denominator, or using the state-space inversion method included in Section 6.2.3.

6.3 Results

Identified models were generated for actuator 3 on the SHM fatigue testing rig at NRC, pictured in Fig. 1.1. The 3$^{\text{rd}}$-order dual-Youla identified model from Section 4.5 was used to generate both 1DOF $\mathcal{H}_\infty$ controllers and inversion-based feedforward controllers. Before synthesizing the $\mathcal{H}_\infty$ controllers, the identified model was prewrapped using a P controller with a gain of 1, as in Section 7.1. Two $\mathcal{H}_\infty$ controllers were successfully implemented, a 1$^{\text{st}}$-order 1DOF controller without integrator, and a 3$^{\text{rd}}$-order 1DOF controller with integrator, where $\mathcal{H}_\infty$ with integrator is denoted by $\mathcal{H}_\infty^I$. 1$^{\text{st}}$-, 2$^{\text{nd}}$-, and 3$^{\text{rd}}$-order inversion-based feedforward controllers were also generated. The 2$^{\text{nd}}$-order controller was shown to yield the lowest RMS error in preliminary testing and was subsequently used to perform timing experiments. This feedforward controller was implemented in conjunction with each of the $\mathcal{H}_\infty$ controllers, and a hand-tuned PI controller. The PI controller was tuned to have a P gain of 40 and an I gain of 80. In addition, the FLF was enabled, with the pole frequency set to 0.75 Hz and the zero frequency set to 3 Hz.

To assess controller performance, each controller combination was used to track the same load profile for the same number of passes with PSO turned on. In addition, the null pacing limits for actuators 1, 2, and 4 were opened up to 100% to ensure that null pacing could only occur due to actuator 3. Afterwards, the RMS errors (described in Section 2.6.3) and runtimes were analyzed and compared. The minimum transition time was initially set to 0.4 seconds, then to 0.1 seconds. First, the tuned PI controller with CCC (described in Section 3.3.3) was implemented and used as the baseline for comparison. PI with CCC is the controller scheme typically used to perform fatigue testing at NRC. Next, CCC was turned off and the inversion-based feedforward controller was implemented in combination with the tuned PI controller, the 1$^{\text{st}}$-order $\mathcal{H}_\infty$ controller, and the 3$^{\text{rd}}$-order $\mathcal{H}_\infty^I$ controller. The results for minimum times of 0.4 and 0.1 seconds are contained in Tables 6.1 and 6.2,

respectively. Plots for all 4 controllers for a minimum time of 0.4 seconds are shown in Figs. 6.3 to 6.6.

Table 6.1: SISO results for actuator 3 with a minimum time of 0.4 s.

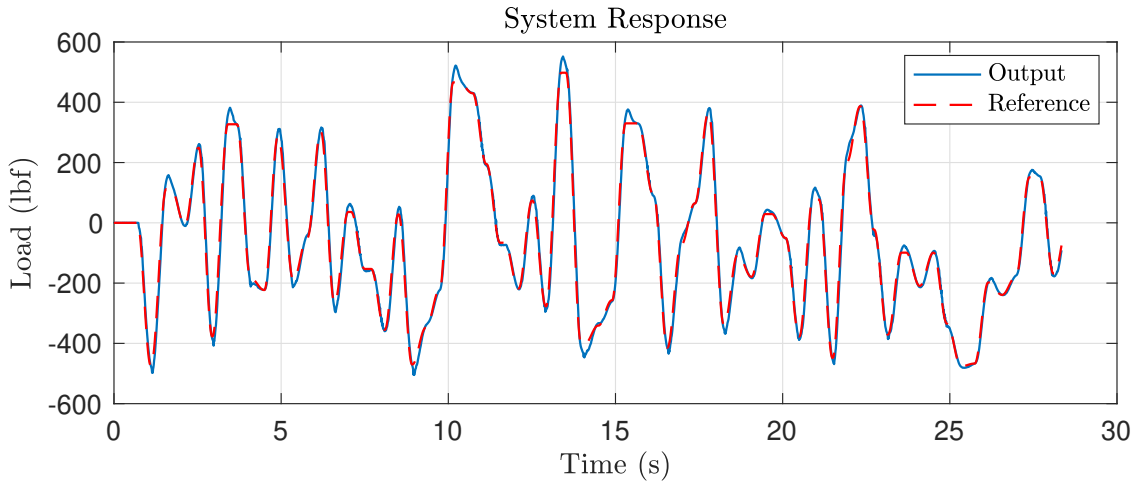
Controller	$\mathcal{H}_\infty$ **Order**	**Profile Time**	**Runtime**	**Null Pace**	**RMS Error**	**% Difference**	
						Time	**RMSE**
PI + CCC	–	25.15 s	28.34 s	3.19 s	22.17 lbf	–	–
PI + FFW	–	24.95 s	27.05 s	2.10 s	14.98 lbf	–4.6%	–32.4%
$\mathcal{H}_\infty$ + FFW	1	24.85 s	27.29 s	2.44 s	9.09 lbf	–3.7%	–59.0%
$\mathcal{H}_\infty^I$ + FFW	3	24.85 s	27.07 s	2.22 s	11.16 lbf	–4.5%	–49.7%

Table 6.2: SISO results for actuator 3 with a minimum time of 0.1 s.

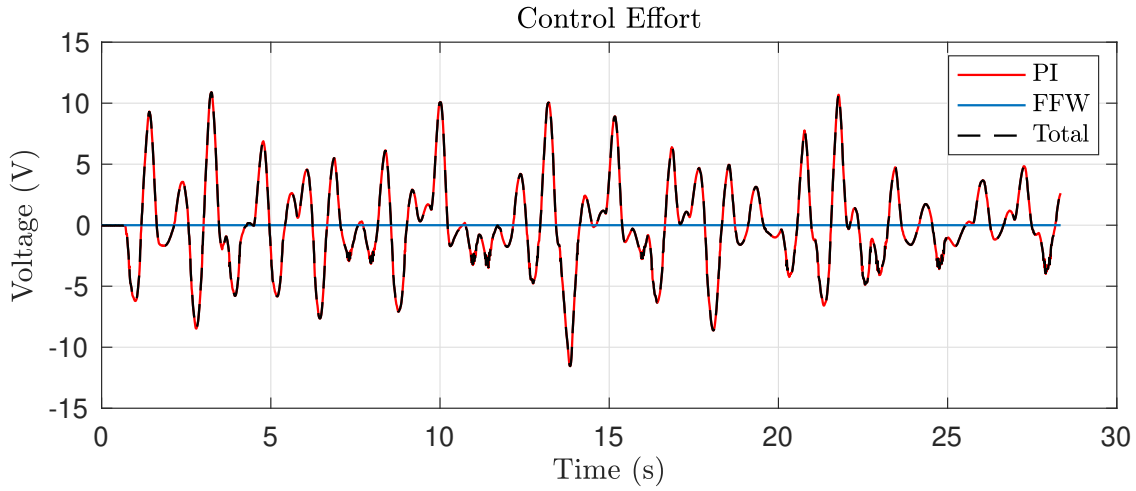
Controller	$\mathcal{H}_\infty$ **Order**	**Profile Time**	**Runtime**	**Null Pace**	**RMS Error**	**% Difference**	
						Time	**RMSE**
PI + CCC	–	12.95 s	16.04 s	3.09 s	37.46 lbf	–	–
PI + FFW	–	11.64 s	16.55 s	4.91 s	37.47 lbf	+3.2%	+0.03%
$\mathcal{H}_\infty$ + FFW	1	9.44 s	12.85 s	3.41 s	31.29 lbf	–19.9%	–16.5%
$\mathcal{H}_\infty^I$ + FFW	3	9.22 s	13.06 s	3.84 s	36.42 lbf	–18.6%	–2.8%

6.3.1 Discussion

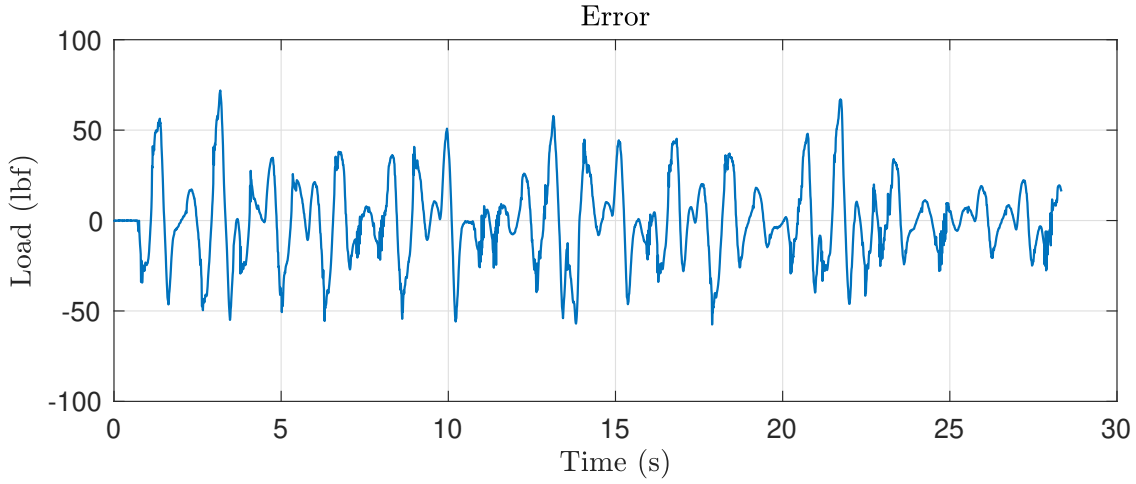
A major advantage of feedforward control is that it is a "predictive" controller, whereas a feedback controller is "reactive." While feedback controllers apply a control input based on the measured error, feedforward applies a control input based on the anticipated response of the plant. If the system model is accurate, an inversion-based feedforward controller should provide the bulk of the control effort. Then the feedback controller is used to "clean up" any inaccuracies that may arise due to model uncertainty, noise, or disturbances. In the case of each of the controllers implemented with feedforward, the feedforward controller is contributing the vast majority of the control effort. However, the $\mathcal{H}_\infty$ and $\mathcal{H}_\infty^I$ controllers are better than PI at cleaning up the control signal. Consequently, these controllers are able to achieve the best performance, reaching faster speeds with less error.



(a) System response data.

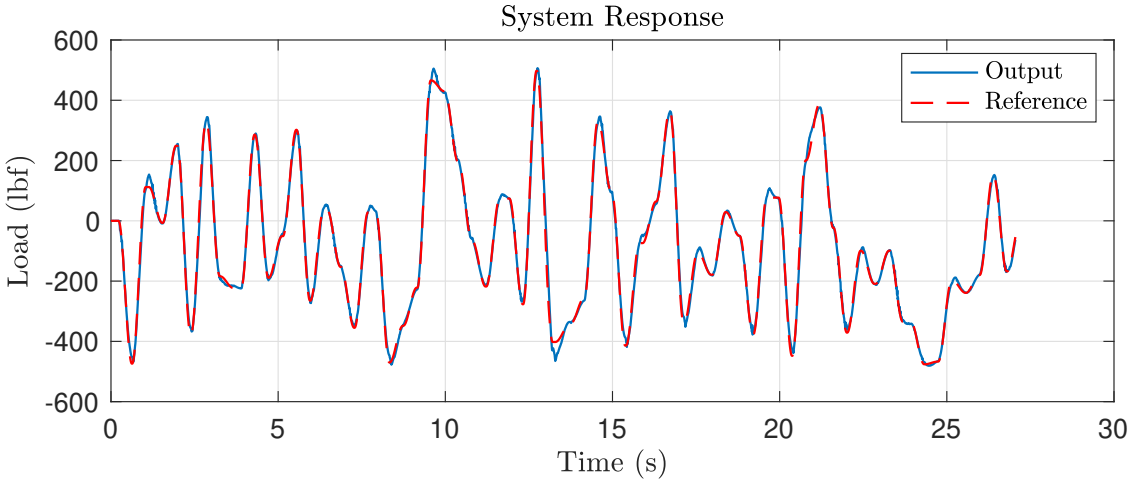


(b) Voltage data.

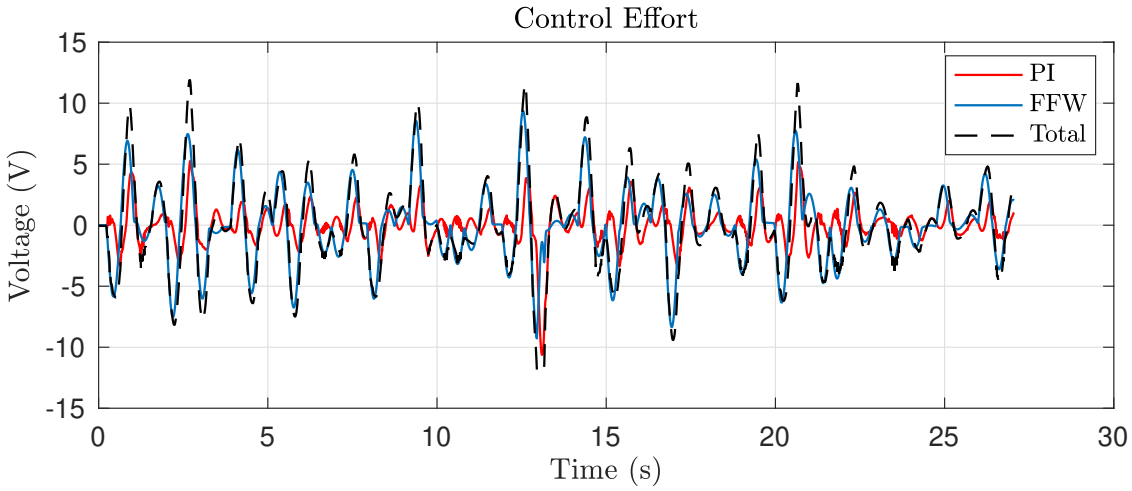


(c) Error data.

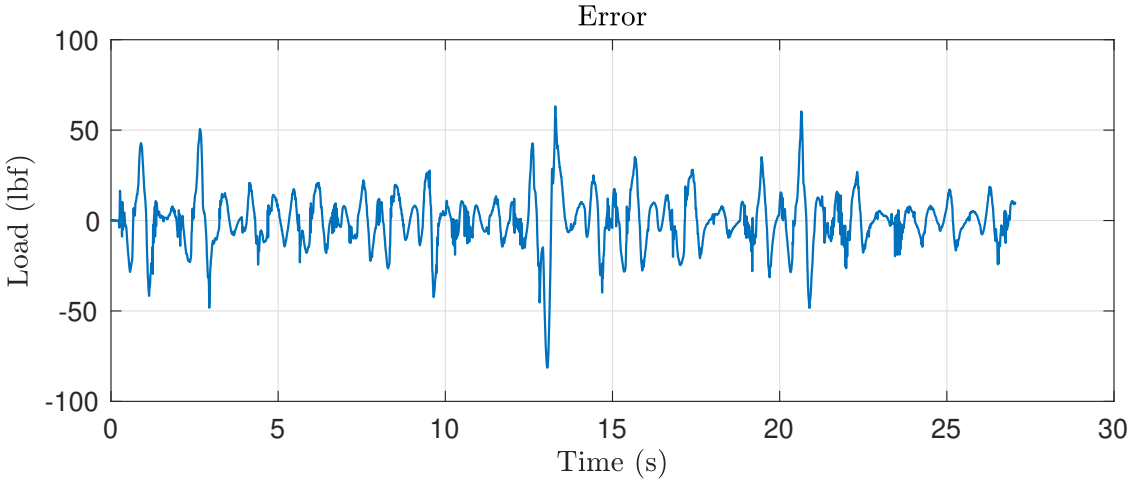
Figure 6.3: PI control + CCC.



(a) System response data.

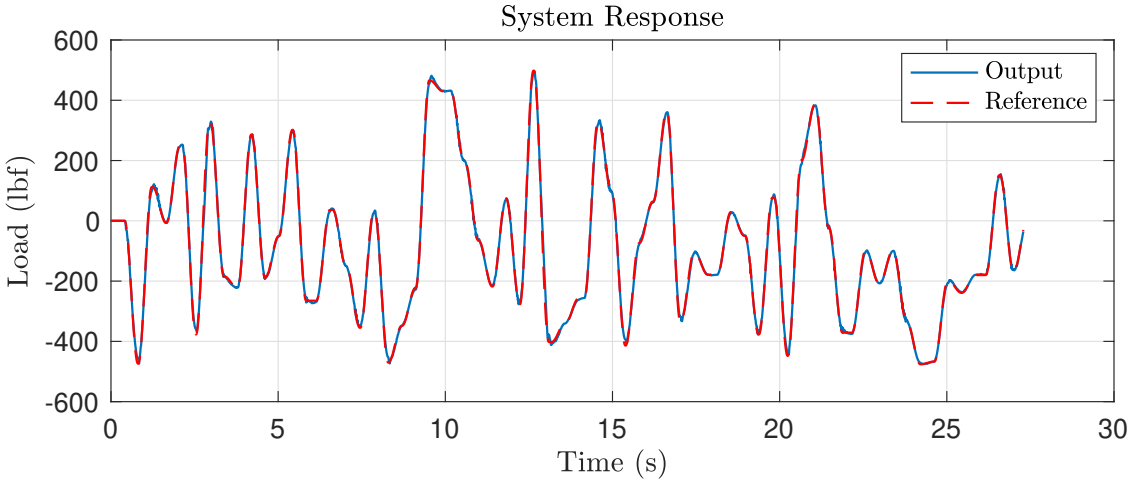


(b) Voltage data.

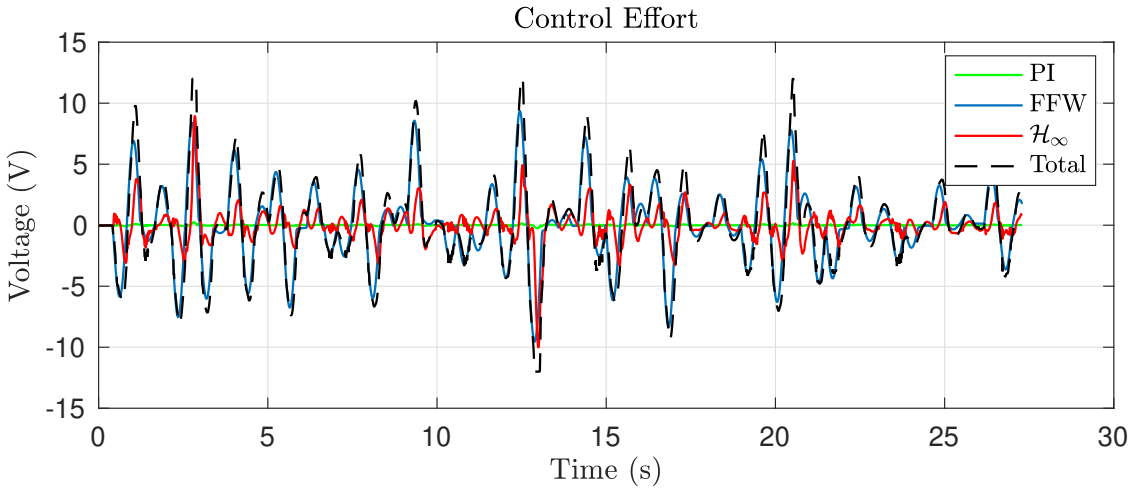


(c) Error data.

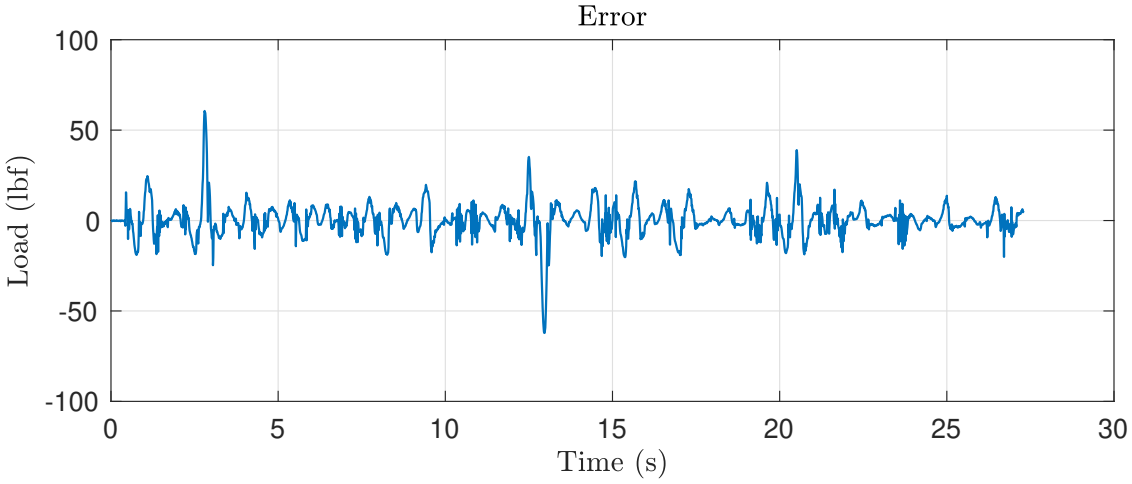
Figure 6.4: PI control + 2nd-order FFW.



(a) System response data.

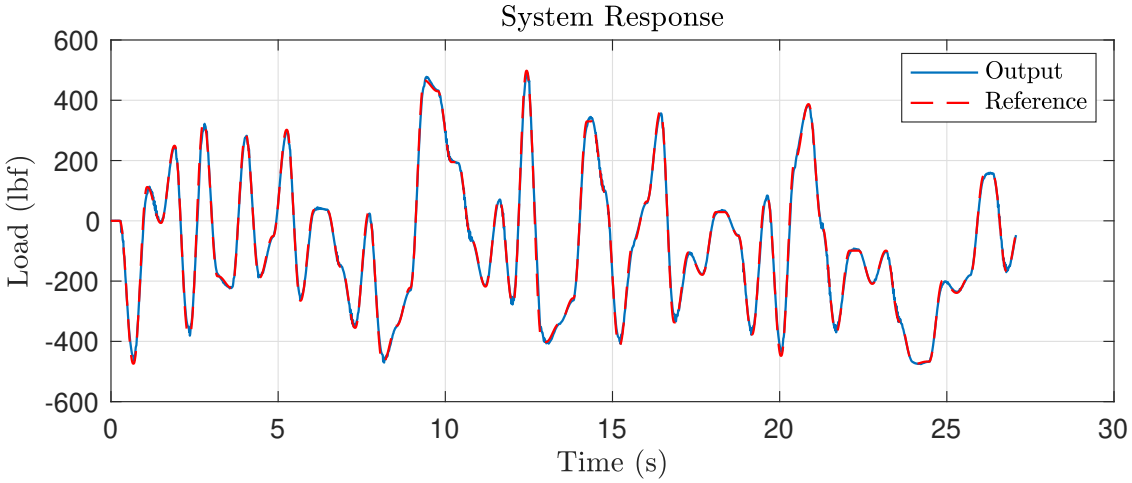


(b) Voltage data.

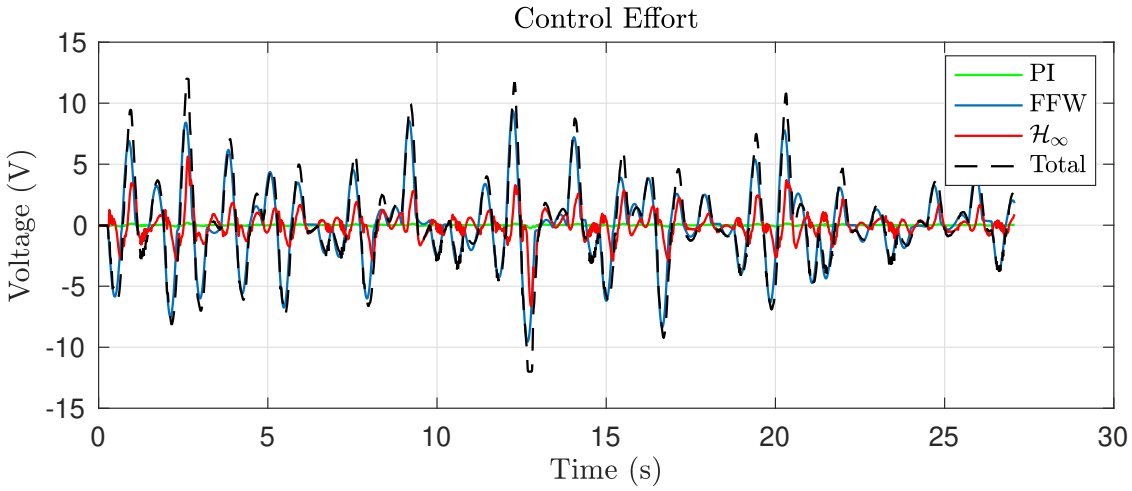


(c) Error data.

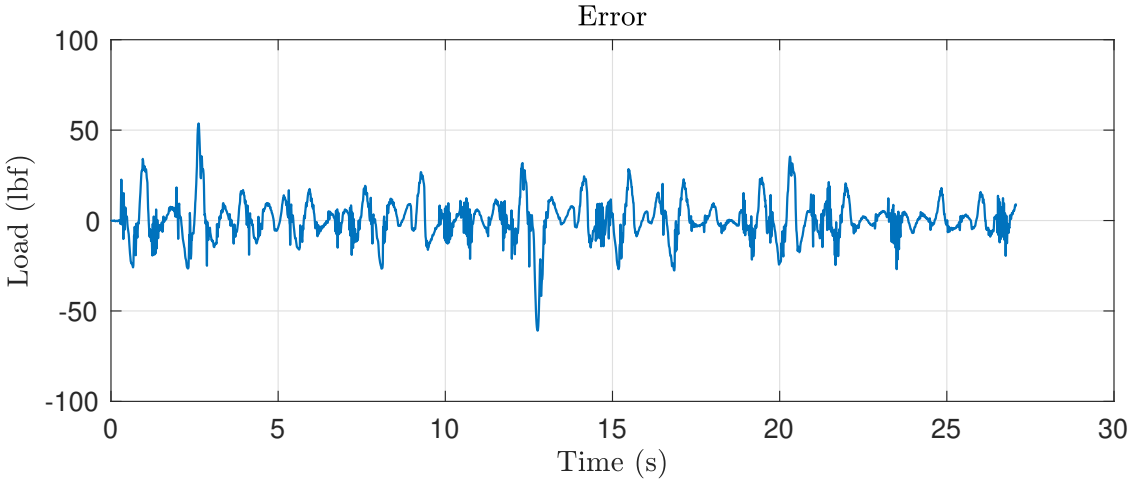
Figure 6.5: 1st-order $\mathcal{H}_\infty$ + 2nd-order FFW.



(a) System response data.



(b) Voltage data.



(c) Error data.

Figure 6.6: 3rd-order $\mathcal{H}_\infty^I$ + 2nd-order FFW.

Chapter 7

MIMO Two Degree-of-Freedom Control via $\mathcal{H}_\infty$

In Chapter 6, 2DOF controllers were designed for SISO systems by independently designing the feedback and feedforward controllers. While generating inversion-based feedforward controllers is straightforward in the SISO case, generating MIMO feedforward controllers via transfer matrix inversion is not recommended. In this chapter, the feedforward and feedback controllers will be designed simultaneously using a 2DOF $\mathcal{H}_\infty$ approach. $\mathcal{H}_\infty$-optimal control synthesis yields a controller that, if successful, minimizes the $\mathcal{H}_\infty$ norm of the closed-loop system [48]. Before performing controller synthesis, the plant model is "prewrapped" with a detuned PI controller. Then the generalized plant is defined, as in Section 2.4.3. Specifically, the 2DOF generalized plant formulation with optional integrator is used in this chapter.

7.1 PI Control "Prewrap"

The MTS controller includes a plethora of safety features and it was preferable to keep these active when implementing alternative controllers. To keep the MTS controller in the loop, while allowing the $\mathcal{H}_\infty$ controller to provide the bulk of the control effort, the built-in PI controller was heavily detuned. The P gain was set to 1 and the I gain to 0, compared to typical P and I gains of 40 and 80, respectively. The detuned PI controller was used to "prewrap" the identified system model by combining them in negative feedback. The joint proportional and $\mathcal{H}_\infty$ controller formulation is shown in Fig. 7.1, and an equivalent block diagram in Fig. 7.2. The plant used for $\mathcal{H}_\infty$ controller design is then the closed-loop system $G(s)$. Technically, this approach assumes that the $-r(s)$ reference input to the P controller in Fig. 7.2 is zero. This assumption was tested in simulation and had a negligible effect on the tracking performance of the closed-loop system.

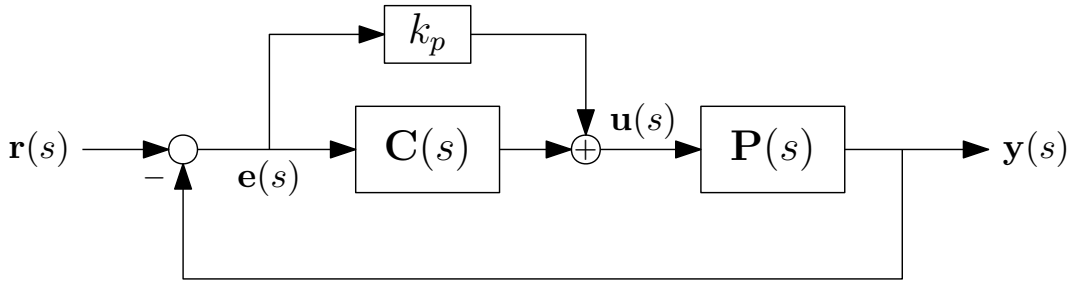


Figure 7.1: Joint proportional and $\mathcal{H}_\infty$ control.

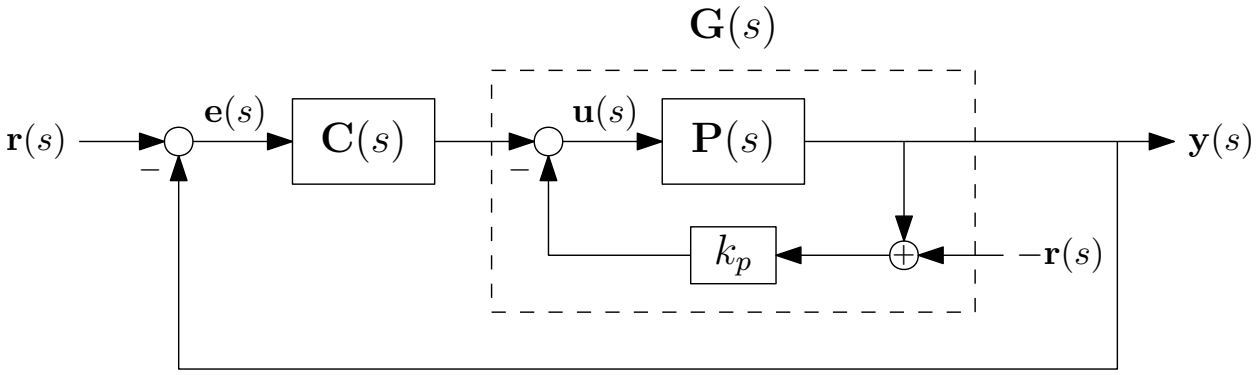


Figure 7.2: P control "prewrap."

7.2 $\mathcal{H}_\infty$ Controller Synthesis Method

The optimal $\mathcal{H}_\infty$ controller, which minimizes γ such that $\|\mathcal{G}\|_\infty < \gamma$, is found by solving a convex optimization problem subject to LMI constraints. Specifically, the problem to be solved thus yielding an $\mathcal{H}_\infty$ controller is [14, 49, 50]

$$\underset{\gamma,\mathbf{X}_1,\mathbf{Y}_1,\mathbf{A}_N,\mathbf{B}_N,\mathbf{C}_N,\mathbf{D}_N}{\text{minimize}} \gamma \qquad \text{such that}$$

$$\begin{bmatrix} \mathbf{X}_1 & \mathbf{1} \\ \mathbf{1} & \mathbf{Y}_1 \end{bmatrix} > 0\,,$$

$$\begin{bmatrix} \mathbf{A}\mathbf{Y}_1 + \mathbf{Y}_1\mathbf{A}^\top & \star & \star & \star \\ \mathbf{A}^\top + \mathbf{A}_N + (\mathbf{B}_2\mathbf{D}_N\mathbf{C}_2)^\top & \mathbf{X}_1\mathbf{A} + \mathbf{A}^\top\mathbf{X}_1 + \mathbf{B}_N\mathbf{C}_2 + \mathbf{C}_2^\top\mathbf{B}_N^\top & \star & \star \\ (\mathbf{B}_1 + \mathbf{B}_2\mathbf{D}_N\mathbf{D}_{21})^\top & (\mathbf{X}_1\mathbf{B}_1 + \mathbf{B}_N\mathbf{D}_{21})^\top & -\gamma\mathbf{1} & \star \\ \mathbf{C}_1\mathbf{Y}_1 + \mathbf{D}_{12}\mathbf{C}_N & \mathbf{C}_1 + \mathbf{D}_{12}\mathbf{D}_N\mathbf{C}_2 & \mathbf{D}_{11} + \mathbf{D}_{12}\mathbf{D}_N\mathbf{D}_{21} & -\gamma\mathbf{1} \end{bmatrix} < 0\,.$$

The controller is recovered by

$$\mathbf{D}_K = (\mathbf{1} + \mathbf{D}_{K2}\mathbf{D}_{22})^{-1}\,\mathbf{D}_{K2}, \qquad \mathbf{C}_K = (\mathbf{1} - \mathbf{D}_K\mathbf{D}_{22})\,\mathbf{C}_{K2},$$
$$\mathbf{B}_K = \mathbf{B}_{K2}\,(\mathbf{1} - \mathbf{D}_{22}\mathbf{D}_K)\,, \qquad \mathbf{A}_K = \mathbf{A}_{K2} - \mathbf{B}_K\,(\mathbf{1} - \mathbf{D}_{22}\mathbf{D}_K)^{-1}\,\mathbf{D}_{22}\mathbf{C}_K,$$

where

$$\left[\begin{array}{c|c} \mathbf{A}_{K2} & \mathbf{B}_{K2} \\ \hline \mathbf{C}_{K2} & \mathbf{D}_{K2} \end{array}\right] = \begin{bmatrix} \mathbf{X}_2 & \mathbf{X}_1\mathbf{B}_2 \\ \mathbf{0} & \mathbf{1} \end{bmatrix}^{-1} \left(\begin{bmatrix} \mathbf{A}_N & \mathbf{B}_N \\ \mathbf{C}_N & \mathbf{D}_N \end{bmatrix} - \begin{bmatrix} \mathbf{X}_1\mathbf{A}\mathbf{Y}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \right) \begin{bmatrix} \mathbf{Y}_2^\top & \mathbf{0} \\ \mathbf{C}_2\mathbf{Y}_1 & \mathbf{1} \end{bmatrix}^{-1}$$

and $\mathbf{X}_2$ and $\mathbf{Y}_2$ are any matrices that satisfy $\mathbf{X}_2\mathbf{Y}_2^\top = \mathbf{1} - \mathbf{X}_1\mathbf{Y}_1$. For example, $\mathbf{Y}_2$ can be set to $\mathbf{1}$, leaving $\mathbf{X}_2 = \mathbf{1} - \mathbf{X}_1\mathbf{Y}_1$. Alternatively, $\mathbf{X}_2$ and $\mathbf{Y}_2$ can be solved using an LU decomposition, which is the method used in this thesis.

7.2.1 Order Reduction

As mentioned in Chapter 6, the MTS controller has a limited availability of computational overhead. Therefore, it is often necessary to implement reduced-order controllers. $\mathcal{H}_\infty$ controllers have the same order as the generalized plant, which means they are typically high-order, especially if multiple weighting filters are used. The $\mathcal{H}_\infty$ controllers should thus be reduce-ordered before being implemented on the MTS system. Order reduction can be performed manually by eliminating higher modes, or using the `balred` command in `MATLAB`, as mentioned in Section 2.5.1.

7.3 Results

Identified models of the SHM fatigue testing rig at NRC, pictured in Fig. 1.1, were used to design $\mathcal{H}_\infty$ controllers. Before performing the controller synthesis, the identified models were prewrapped as in Section 7.1, using a P controller where the P gain was set to 1. Then the weights were tuned as in Section 2.4.4 and used in the generalized plant. The synthesized controllers were first tested in simulation, then reduce-ordered and discretized using forward Euler before being implemented using C code in calculated channels on the MTS system. They were simulated again on the MTS system, then injected gradually in function generation mode before being used to run tests.

Initially, MIMO $\mathcal{H}_\infty$ controllers were synthesized based on identified models for each half of the SHM platform. First, $\mathcal{H}_\infty$ controllers were run on actuators 1 & 2, and compared to results obtained using PI control with CCC, where the PI controllers were tuned as in Chapter 6 to have a P gain of 40 and an I gain of 80. Next, the same tests were performed on actuators 3 & 4. When testing controllers on each half of the platform, the null pacing limits on the other two actuators were opened up to 100%. In this way, null pacing could only occur due to the actuators being tested. Afterwards, both sets of 2-actuator $\mathcal{H}_\infty$ controllers were used to run a load profile on all 4 actuators simultaneously. An attempt was made to

synthesize a MIMO $\mathcal{H}_\infty$ controller for all 4 actuators, and although some controllers looked promising in simulation, they could not successfully control the test rig in practice. However, due to the configuration of the SHM platform, the effect of actuators 1 & 2 on actuators 3 & 4 is limited, and vice versa, which allows the $\mathcal{H}_\infty$ controllers synthesized for each half of the platform to achieve significant improvements over the base case of PI with CCC.

7.3.1 Actuators 1 & 2

Various 2DOF $\mathcal{H}_\infty$ controllers were synthesized using a dual-Youla identified model for actuators 1 & 2, with and without integrator, and using different sets of weighting functions. Various reduced-order versions of these controllers were tested and two controllers were successfully implemented: a 2^{nd}-order $\mathcal{H}_\infty$ controller without integrator and a 6^{th}-order $\mathcal{H}_\infty$ controller with integrator (denoted $\mathcal{H}_\infty^I$). Both sets of controllers were used to PSO a 2-actuator random load profile. The same profile was also run with PSO using the standard PI plus CCC controller. Tables 7.1 and 7.2 include results for minimum transition times of 0.4 and 0.1 seconds, respectively. Figure 7.3 shows the system response and control effort of the 2^{nd}-order $\mathcal{H}_\infty$ controller for a minimum time of 0.4 seconds.

Table 7.1: MIMO results for actuators 1 & 2 with a minimum time of 0.4 s.

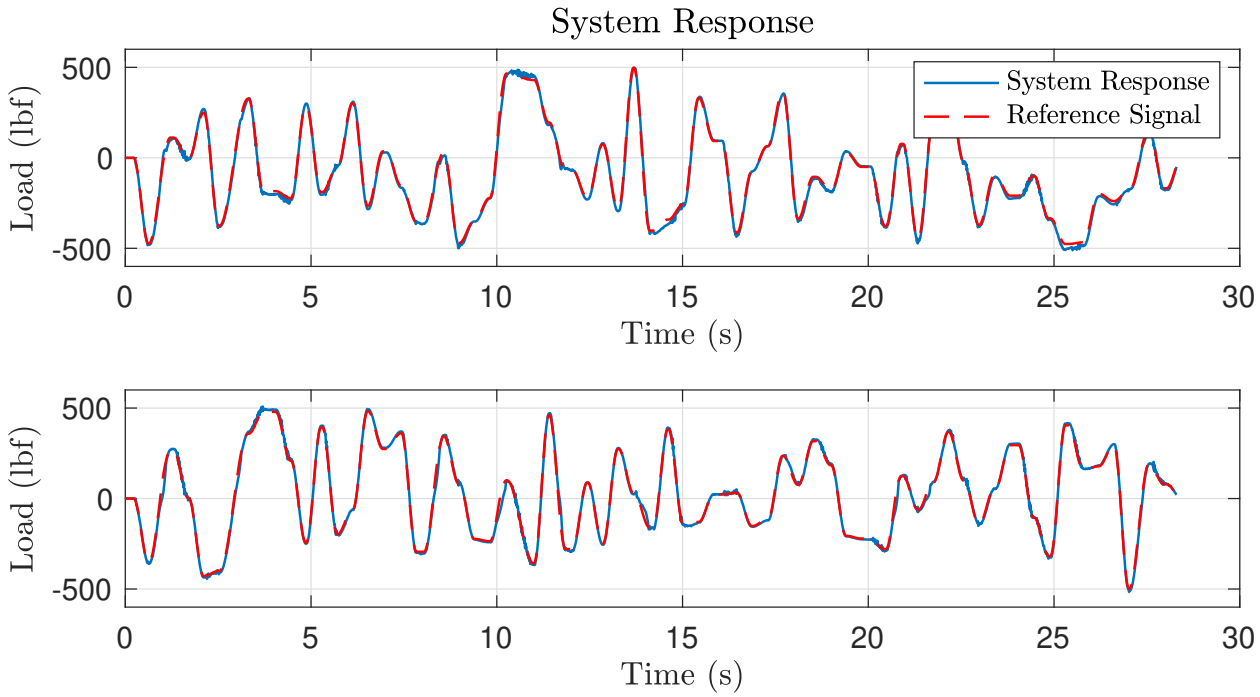
Controller	$\mathcal{H}_\infty$ **Order**	**Profile Time**	**Run Time**	**Null Pace**	**RMS Error**		**% Difference**	
					C1	**C2**	**Time**	**RMSE**
PI + CCC	–	26.46 s	31.32 s	4.86 s	21.36 lbf	24.31 lbf	–	–
2DOF $\mathcal{H}_\infty$	2	25.97 s	28.30 s	2.33 s	15.12 lbf	16.27 lbf	–9.6%	–31.3%
2DOF $\mathcal{H}_\infty^I$	6	25.51 s	27.52 s	2.01 s	19.94 lbf	19.81 lbf	–12.1%	–13.0%

Table 7.2: MIMO results for actuators 1 & 2 with a minimum time of 0.1 s.

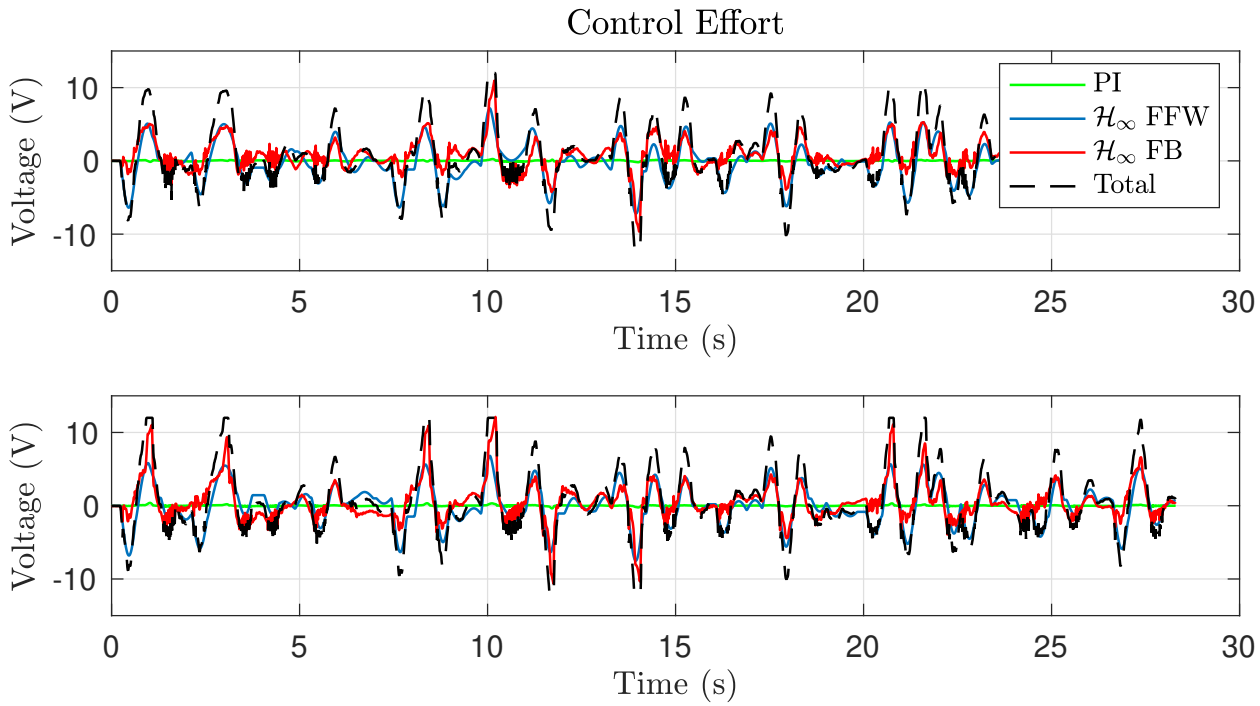
Controller	$\mathcal{H}_\infty$ **Order**	**Profile Time**	**Run Time**	**Null Pace**	**RMS Error**		**% Difference**	
					C1	**C2**	**Time**	**RMSE**
PI + CCC	–	15.05 s	19.12 s	4.07 s	32.92 lbf	38.02 lbf	–	–
2DOF $\mathcal{H}_\infty$	2	12.27 s	14.33 s	2.06 s	22.21 lbf	26.42 lbf	–25.1%	–31.4%
2DOF $\mathcal{H}_\infty^I$	6	11.43 s	14.83 s	3.40 s	29.99 lbf	30.25 lbf	–22.4%	–15.1%

7.3.2 Actuators 3 & 4

A dual-Youla identified model was used to synthesize 2DOF $\mathcal{H}_\infty$ controllers for actuators 3 & 4. One particular set of weighting functions resulted in successful controllers, and the



(a) System response data.



(b) Voltage data.

Figure 7.3: 2nd-order $\mathcal{H}_\infty$ on actuators 1 & 2.

1st-, 2nd-, and 3rd-order versions of this controller were implemented. Each controller was used to PSO a random load profile (the same load profile run on actuators 1 & 2) down to minimum transition times of 0.4 and 0.1 seconds. The results are included in Tables 7.3 to 7.4. The results for the 1st-order controller for a minimum time of 0.4 seconds are shown in Fig. 7.4.

Table 7.3: MIMO results for actuators 3 & 4 with a minimum time of 0.4 s.

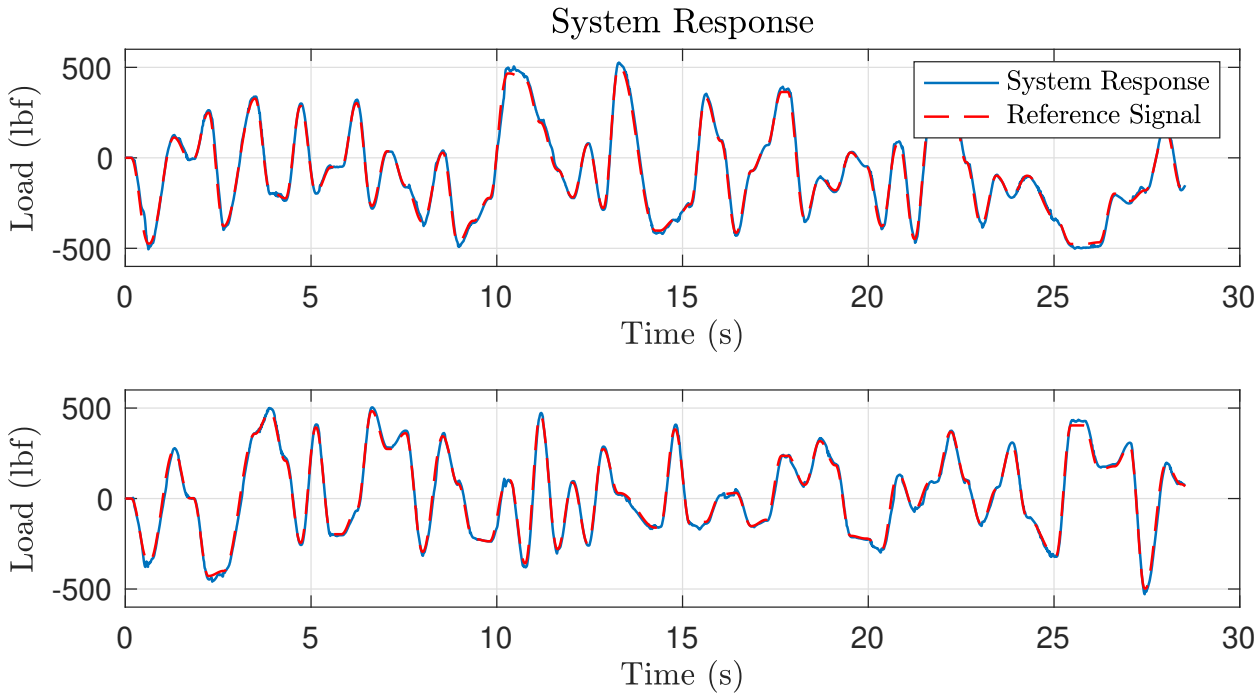
Controller	$\mathcal{H}_\infty$ Order	Profile Time	Run Time	Null Pace	RMS Error C3	RMS Error C4	% Difference Time	% Difference RMSE
PI + CCC	–	26.79 s	30.82 s	4.03 s	24.90 lbf	21.55 lbf	–	–
2DOF $\mathcal{H}_\infty$	1	26.93 s	28.53 s	1.60 s	19.68 lbf	19.80 lbf	–7.4%	–15.0%
	2	25.94 s	28.06 s	2.12 s	20.26 lbf	18.50 lbf	–9.0%	–16.6%
	3	26.14 s	29.06 s	2.92 s	21.53 lbf	19.50 lbf	–5.7%	–11.7%

Table 7.4: MIMO results for actuators 3 & 4 with a minimum time of 0.1 s.

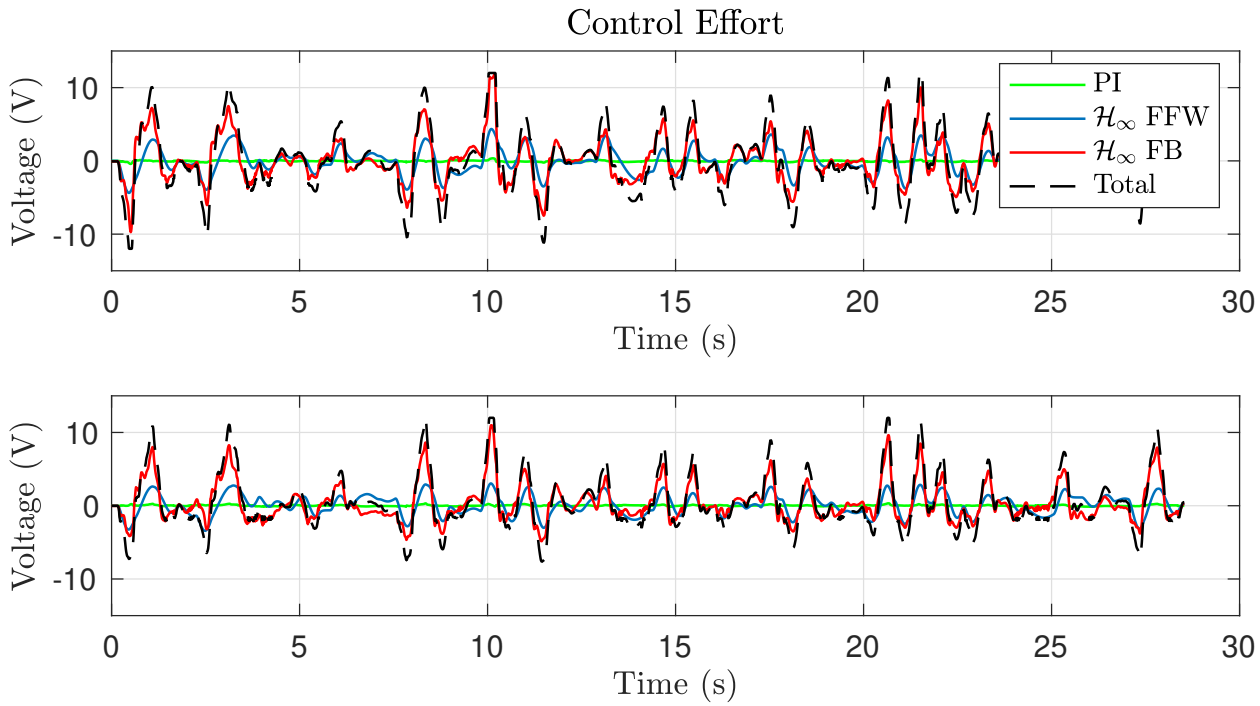
Controller	$\mathcal{H}_\infty$ Order	Profile Time	Run Time	Null Pace	RMS Error C3	RMS Error C4	% Difference Time	% Difference RMSE
PI + CCC	–	17.28 s	21.05 s	3.77 s	35.75 lbf	30.33 lbf	–	–
2DOF $\mathcal{H}_\infty$	1	16.89 s	18.84 s	1.95 s	25.31 lbf	24.10 lbf	–10.5%	–25.2%
	2	15.24 s	17.83 s	2.59 s	29.78 lbf	26.09 lbf	–15.3%	–15.6%
	3	15.53 s	17.60 s	2.07 s	30.27 lbf	27.34 lbf	–16.4%	–12.8%

7.3.3 All 4 Actuators

A random profile was run on all 4 actuators simultaneously with PSO enabled using the 2nd-order 2DOF $\mathcal{H}_\infty$ controller for actuators 1 & 2 and the 1st-, 2nd, and 3rd-order 2DOF $\mathcal{H}_\infty$ controllers for actuators 3 & 4. The 2nd-order controller was selected for actuators 1 & 2 because it is definitively the best controller tested in Section 7.3.1, both in terms of timing and RMS error. However, for actuators 3 & 4, there is no clear "winner." The 1st-order $\mathcal{H}_\infty$ controller results in the lowest error, the 3rd-order controller yields the fastest time, and the 2nd-order controller is second best in both. Therefore, all three controllers were tested. The load profile was run using PSO to minimum times of 0.4 and 0.1 s, and the results were compared to the PI plus CCC case. The results are included in Tables 7.5 and 7.6. Figure 7.5 shows results for the controller combination of 2nd-order $\mathcal{H}_\infty$ on actuators 1 & 2 and 1st-order $\mathcal{H}_\infty$ on actuators 3 & 4 for a minimum time of 0.4 seconds.

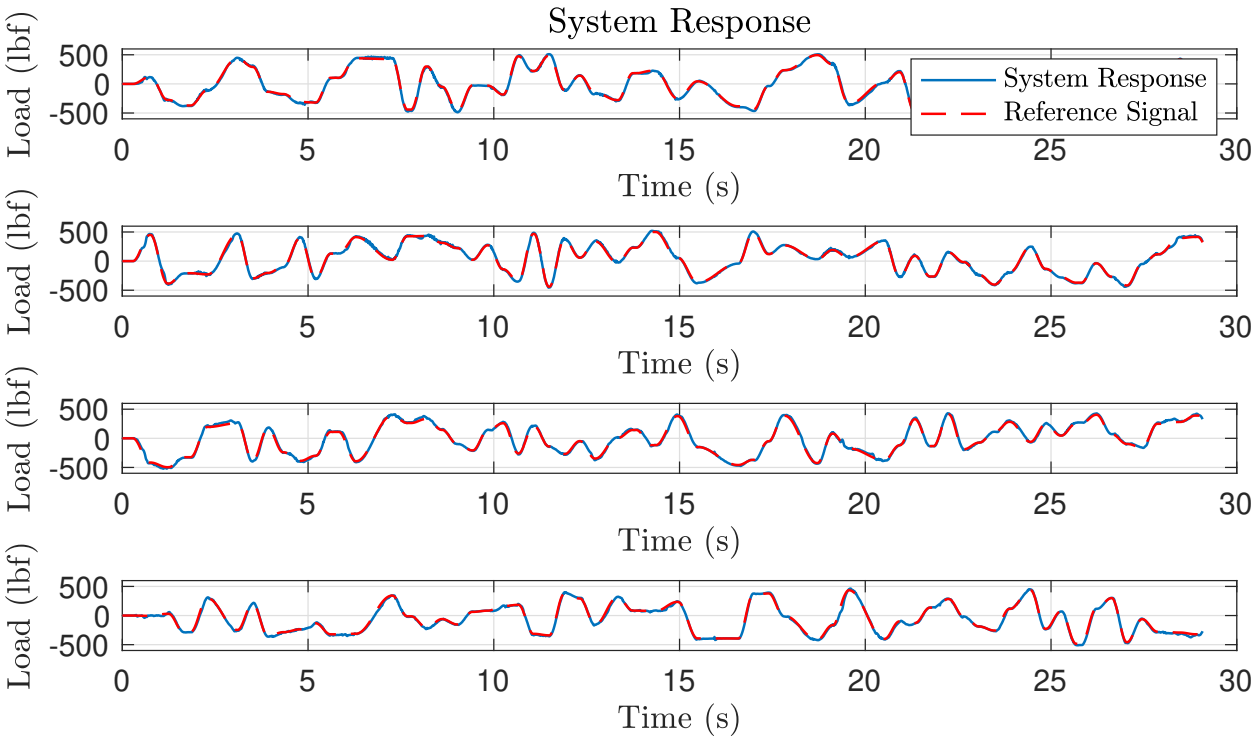


(a) System response data.

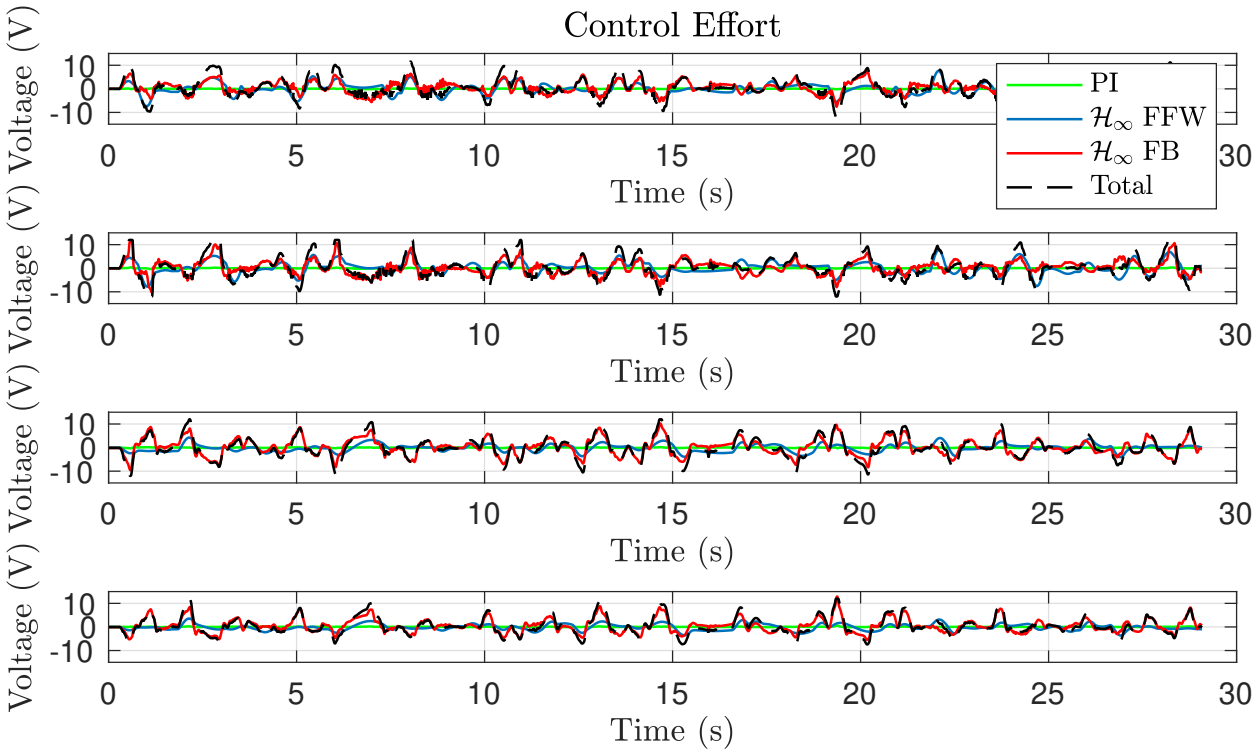


(b) Voltage data.

Figure 7.4: 1st-order $\mathcal{H}_\infty$ on actuators 3 & 4.



(a) System response data.



(b) Voltage data.

Figure 7.5: 2nd-order $\mathcal{H}_\infty$ on actuators 1 & 2 and 1st-order $\mathcal{H}_\infty$ on actuators 3 & 4.

Table 7.5: 4-actuator MIMO results for all actuators with a minimum time of 0.4 s.

Controller	$\mathcal{H}_\infty$ Order	Profile Time	Run Time	Null Pace	RMS Error (lbf)				% Difference	
					C1	C2	C3	C4	Time	RMSE
PI + CCC	–	27.89 s	33.57 s	5.68 s	23.02	24.79	26.21	21.13	–	–
2DOF $\mathcal{H}_\infty$	2 / 1[1]	27.67 s	29.08 s	1.41 s	15.84	16.26	21.48	19.77	–13.4%	–22.9%
	2 / 2	28.08 s	30.31 s	2.23 s	16.53	19.37	22.21	18.48	–9.7%	–19.5%
	2 / 3	28.35 s	31.07 s	2.72 s	15.35	15.71	22.14	20.35	–7.4%	–22.7%

[1] A 2nd-order controller was used on actuators 1 & 2, and a 1st-order controller was used on 3 & 4.

Table 7.6: 4-actuator MIMO results for all actuators with a minimum time of 0.1 s.

Controller	$\mathcal{H}_\infty$ Order	Profile Time	Run Time	Null Pace	RMS Error (lbf)				% Difference	
					C1	C2	C3	C4	Time	RMSE
PI + CCC	–	22.95 s	28.56 s	5.61 s	27.40	29.91	29.77	24.87	–	–
2DOF $\mathcal{H}_\infty$	2 / 1	21.29 s	24.10 s	2.81 s	18.45	20.41	25.88	22.32	–15.6%	–22.2%
	2 / 2	21.76 s	24.31 s	2.55 s	18.37	19.89	26.32	22.97	–14.9 %	–21.8 %
	2 / 3	22.09 s	25.33 s	3.24 s	17.71	19.03	26.03	23.76	–11.3 %	–22.7 %

7.3.4 Discussion

In every test included in this chapter, $\mathcal{H}_\infty$ control is able to achieve test results that are both faster and more accurate than the PI plus CCC standard. Tuning of the controller weights is relatively straightforward, and the $\mathcal{H}_\infty$ controller itself is fairly simple to implement using the code generation template. However, as discussed in Section 2.5, care must be taken to first simulate the $\mathcal{H}_\infty$ control signal on the MTS system and then inject it gradually, as many of the attempted controllers exhibited unstable and/or oscillatory behaviour. Nevertheless, 2DOF $\mathcal{H}_\infty$ control for MIMO systems has great potential for improving timing and error results for large-scale fatigue testing.

Part IV

Conclusion

Chapter 8

Closing Remarks and Future Work

8.1 Conclusions

This thesis considers the modelling and optimal control of a fatigue structural testing rig. Initially, a fatigue test system is modelled analytically and a tool is developed to estimate cycle times. Afterwards, closed-loop linear time-domain system identification techniques are used to identify a numerical model using data. In general, the dual-Youla closed-loop system identification method has proven to consistently produce the best identified models, in terms of both NRMS error and %VAF for both the input and output data. It is worth stating that the analytical model cannot account for physical imperfections in the system; it assumes the valves are perfectly balanced and mechanically-nulled, for example. System identification, on the other hand, can capture this behaviour as best approximated by a linear model. Although the analytical models presented in Chapter 3 are not as accurate as those developed in Chapter 4 using system identification, they provide a useful starting point in the case that the fatigue test rig has not yet been assembled. However, if the test rig has been built, the recommended approach is to perform system identification and use the identified model to estimate cycle times and design controllers.

Next, the various controller synthesis methods in Part III are applied to models of the fatigue test rig. First, a few methods are presented for synthesizing PI controllers. Two of these methods attempt to do so in an optimal fashion, and it is the iterative $\mathcal{H}_\infty$ method that produces the most reasonable PI gains. While formal experiments were not completed using these gains, they are very similar to those typically implemented on the SHM platform. The value of this method becomes evident when it is applied to models of a newly commissioned or unfamiliar test rig. In this case, an initial estimate of starting gains can be extremely valuable, both in terms of time saved in the commissioning phase and improved controller performance. The identified models are also used to synthesize SISO and MIMO 2DOF

controllers. In the SISO case, the feedback control law can be PI or $\mathcal{H}_\infty$, and the feedforward is designed via an inversion-based feedforward controller synthesis method. For MIMO systems, a 2DOF $\mathcal{H}_\infty$ controller synthesis method is used to design $\mathcal{H}_\infty$ controllers with embedded feedforward. In both the SISO and MIMO cases, 2DOF $\mathcal{H}_\infty$ control is able to achieve test results that are considerably better than PI plus CCC.

This thesis presents a novel method for synthesizing locally optimal PI controller gains through the use of the Bounded Real Lemma, iteration, and bisection method. Otherwise, the contributions of this thesis are not in the methods, but in the novel application and implementation of system identification and subsequent inversion-based feedforward and $\mathcal{H}_\infty$ controller synthesis methods to fatigue structural testing rigs. In particular, 2DOF $\mathcal{H}_\infty$-optimal control, in tandem with closed-loop system identification, has been shown to substantially improve the speed and error properties of tests performed on the SHM platform at the NRC. Despite these improvements, there is more progress to be made and recommendations for future work are as follows.

8.2 Recommendations for Future Work

Because system identification has been shown to produce much better models than the analytical modelling approach of Chapter 3, it is recommended that all future work involves system identification methods rather than first-principles modelling. However, the system identification process is not yet seamless. It is recommended that future work involves an investigation into which load profiles and controllers work best for generating identified models to be used in controller synthesis and simulation. In addition, because the choice of dual-Youla coprime factors can have a significant effect on the resulting identified models, it is worth exploring how these should be selected, and perhaps if they can be chosen automatically and, if possible, optimally. The system identification methods appeared to exhibit numerical issues when all 4 actuators were included, and this can only get worse for systems with more actuators. Therefore, future work should be done to ensure that the system identification algorithms can be robustly implemented for MIMO systems with a high number of actuators. Finally, although nonlinear system identification was not explored, the fact that the linear identified models in Chapter 4 are able to accurately simulate the input and output data indicates that linear models are adequate. Nevertheless, potential future work could explore the application of nonlinear system identification algorithms.

Before the controller synthesis methods presented in this thesis can be applied to a "real" test article, it is recommended that they be made more robust to ensure that they are safe to implement. Although most feedforward controllers and some $\mathcal{H}_\infty$ controllers

were "well-behaved," others exhibited saturation, instability, and oscillatory behaviour. The behaviour of the controllers is highly dependent on the identified models used and the choice of weighting filters. It is worth investigating how to robustly select weights such that the resulting controllers are safe to implement. Another worthwhile task would be to characterize the model uncertainty and then perform a robust yet optimal control design in an $\mathcal{H}_\infty$ framework. To complete the work done specifically on the SHM platform, another attempt should be made to synthesize a 2DOF $\mathcal{H}_\infty$ controller for all 4 actuators simultaneously.

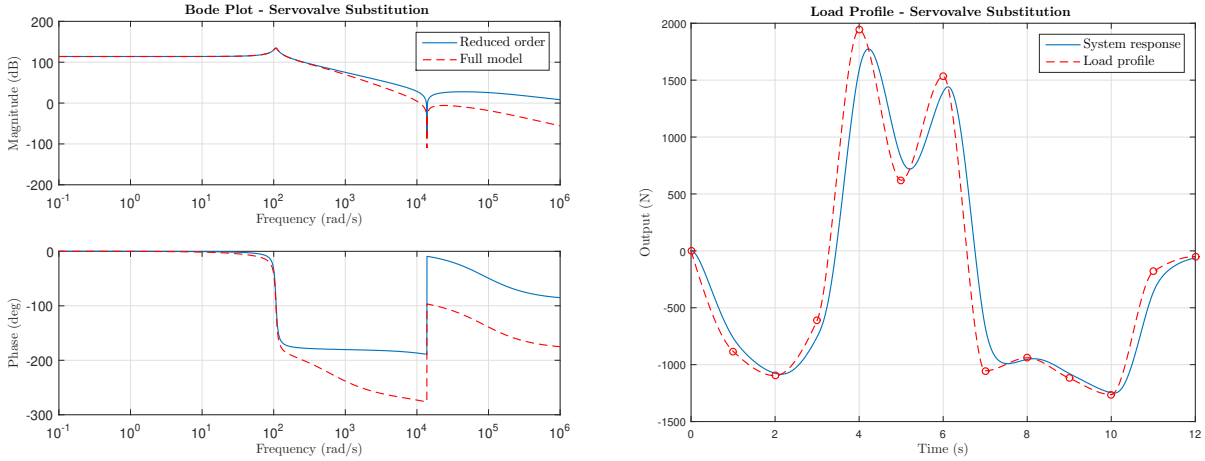
Appendices

Appendix A

Reduced-Order Models

A reduced-order modelling approach was taken in an attempt to eliminate the higher-order dynamics that were suspected to be the cause of numerical scaling issues within the analytical plant model. First, the model was simplified by removing one piece at a time and replacing it with a gain. Next, various parts of the actuator model were reduce-ordered, and the leakage coefficient and integral pole location were experimented with. Afterwards, different combinations of non-dominant poles and zeros were cancelled from the full plant transfer function. Finally, the entire actuator model was replaced by first- and second-order systems. The transfer functions were used in `MATLAB`'s PID tuner to find a stabilizing controller and the system's response to a sample load profile was simulated.

Servovalve gain substitution: The first-order servovalve was replaced with its DC gain. The automatic tuner yielded gains of $K_p = 0$ and $K_i = 4.36$.

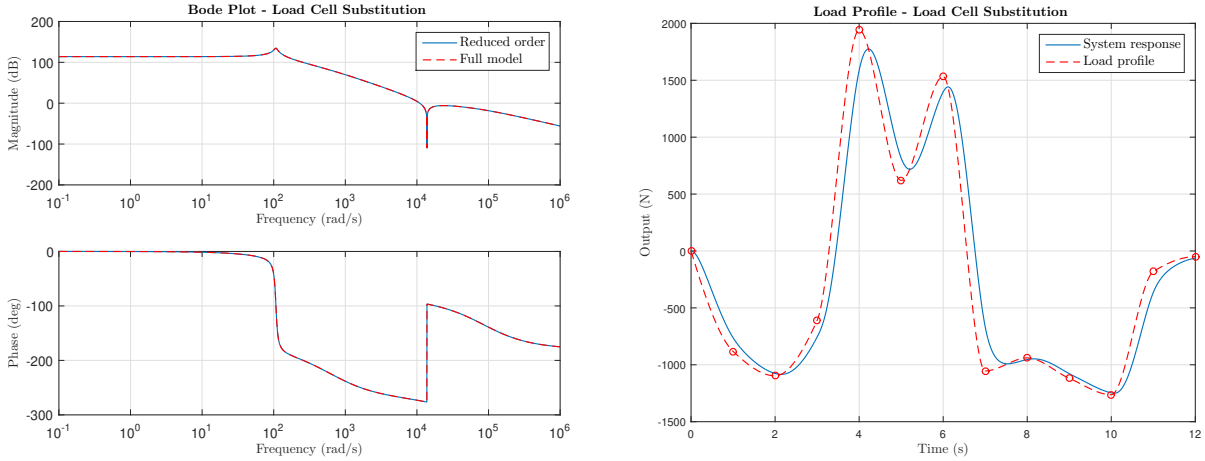


(a) Bode diagram.

(b) Load profile.

Figure A.1: Servovalve gain substitution.

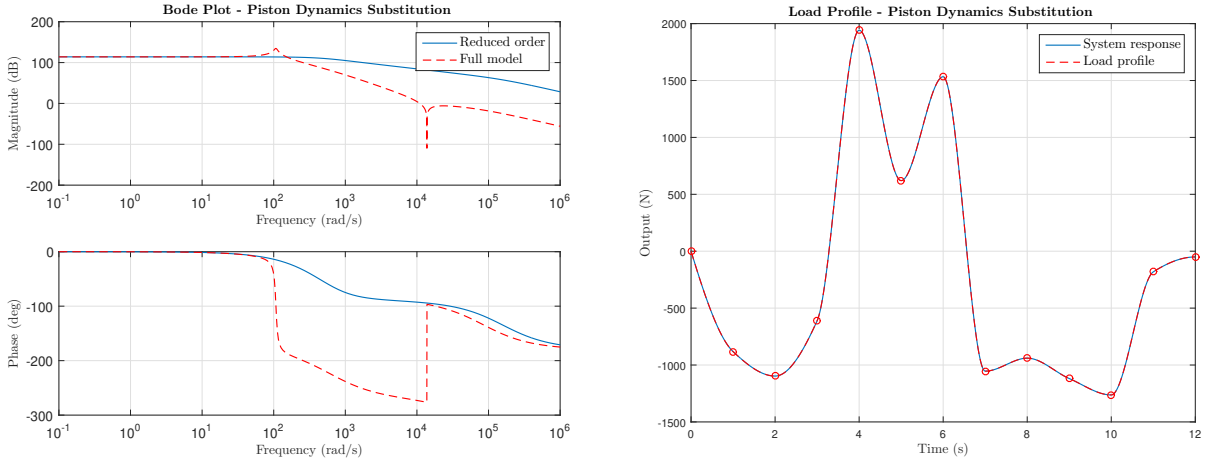
Load cell gain substitution: Note that the load cell was a gain to begin with so the transfer function is exactly the same as the original. The automatic tuner yields gains of $K_p = 0$ and $K_i = 4.34$.



(a) Bode diagram. (b) Load profile.

Figure A.2: Load cell gain substitution.

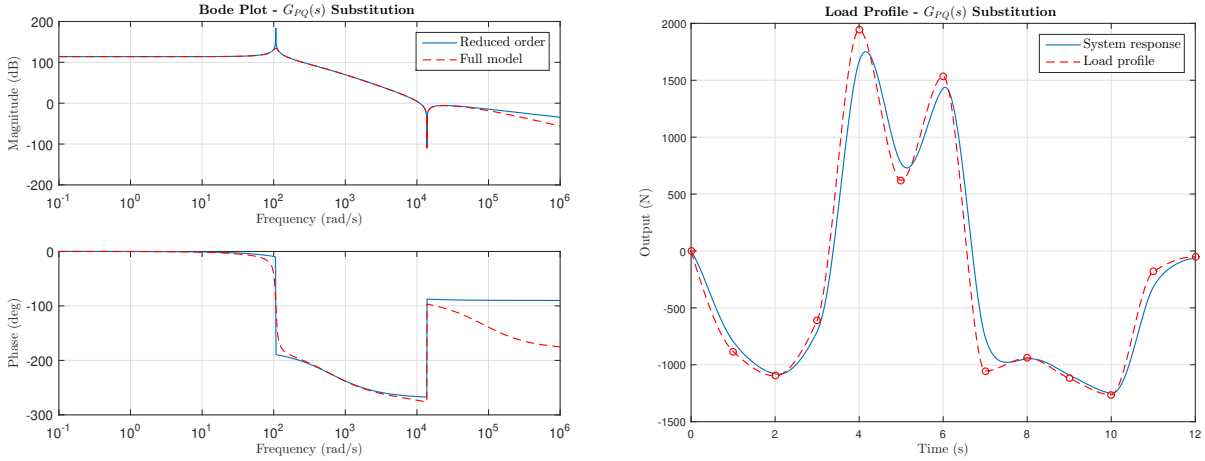
Piston dynamics gain substitution: The second-order expression containing the piston dynamics, $ms^2 + fs + c$, was replaced everywhere by its DC gain. The tuner yields $K_p = 1.27$ and $K_i = 1226$.



(a) Bode diagram. (b) Load profile.

Figure A.3: Piston dynamics gain substitution.

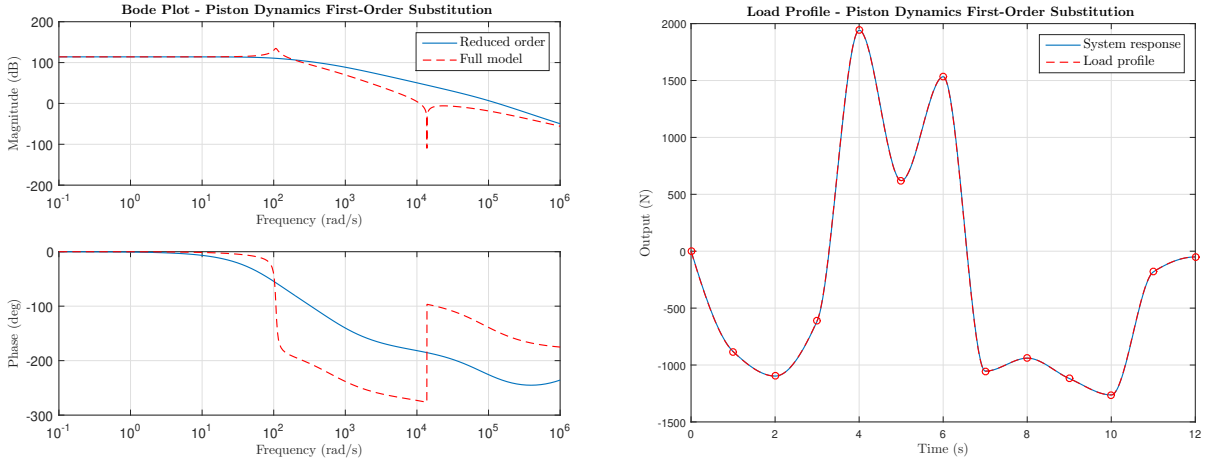
$G_{PQ}(s)$ **gain substitution:** Transfer function $G_{PQ}(s)$ contains the high frequency compressibility effects, as well as the actuator piston dynamics because the piston velocity is fed back. The PI tuner doesn't work, but the PID tuner yields $K_p = 0.63$, $K_i = 6.19$, $K_d = 0.015$.



(a) Bode diagram. (b) Load profile.

Figure A.4: $G_{PQ}(s)$ gain substitution.

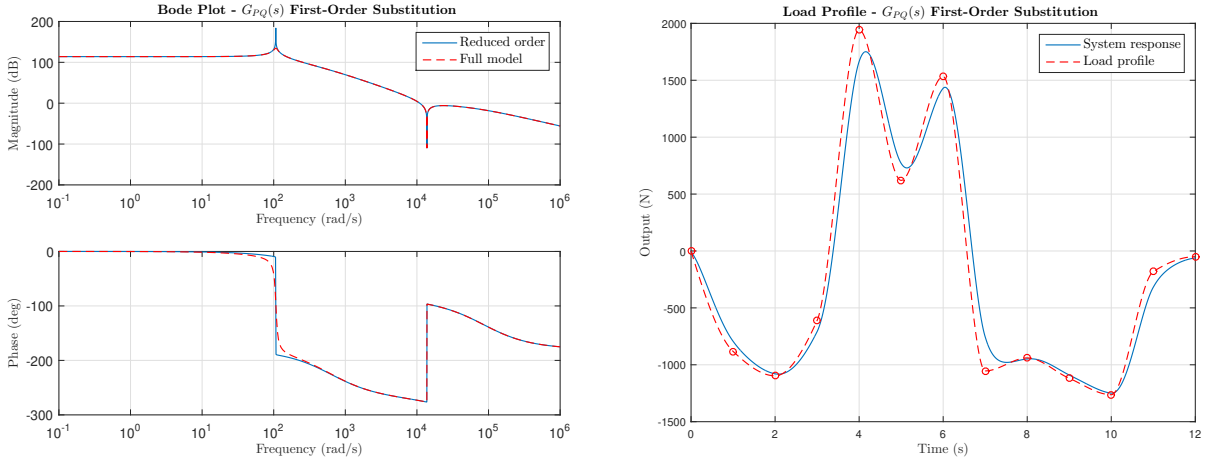
Piston dynamics first-order substitution: The second-order expression containing the piston dynamics, $ms^2 + fs + c$, was replaced everywhere by a first-order system. The tuner yields $K_p = 1.33$ and $K_i = 227.9$.



(a) Bode diagram. (b) Load profile.

Figure A.5: Piston dynamics first-order substitution.

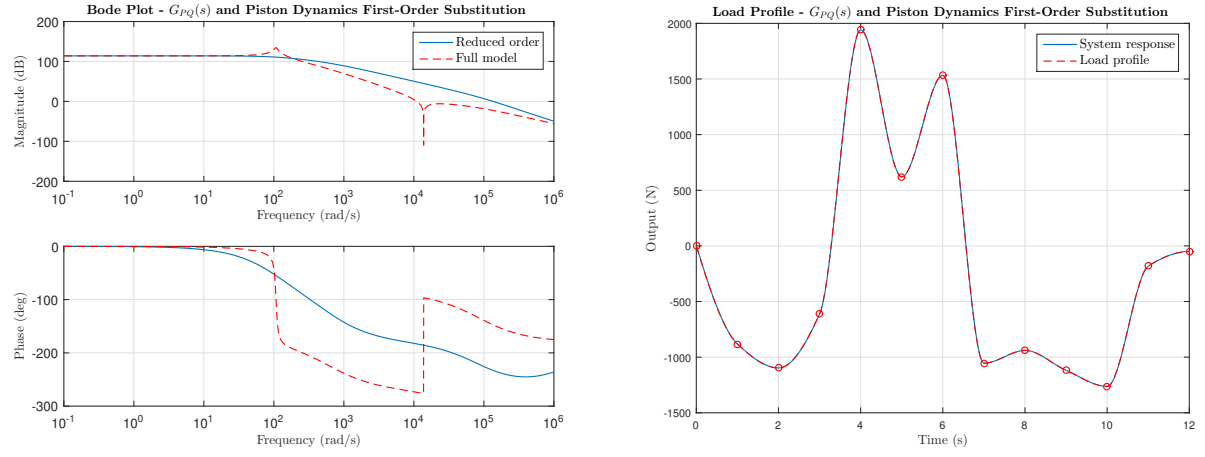
$G_{PQ}(s)$ **first-order substitution:** The transfer function $G_{PQ}(s)$ was replaced by a first-order system. PI control doesn't work. The resulting PID gains are $K_p = 0.63$, $K_i = 6.19$, and $K_d = 0.016$.



(a) Bode diagram. (b) Load profile.

Figure A.6: $G_{PQ}(s)$ first-order substitution.

Piston dynamics and $G_{PQ}(s)$ first-order substitution: The piston dynamics and the transfer function $G_{PQ}(s)$ were both replaced by first-order transfer functions. The resulting PI gains are $K_p = 1.35$ and $K_i = 236.3$.



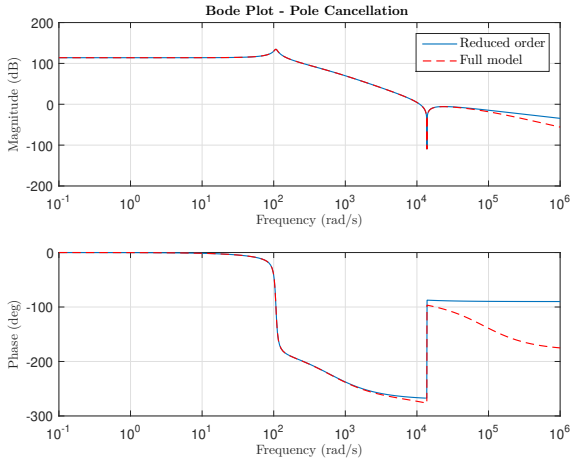
(a) Bode diagram. (b) Load profile.

Figure A.7: Piston dynamics and $G_{PQ}(s)$ first-order substitution.

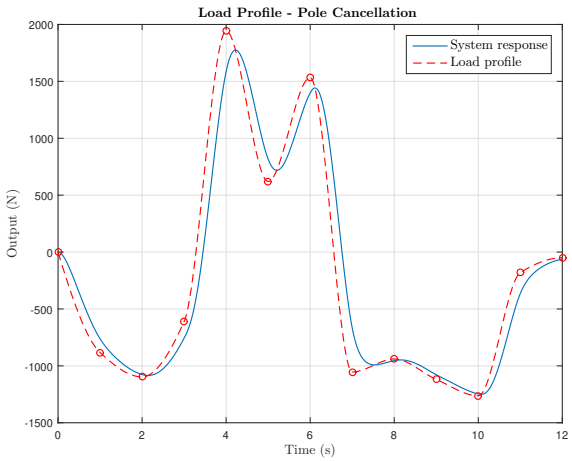
Single pole cancellation: The least dominant pole was cancelled from the overall model by manipulating plant transfer function $G(s)$ as follows

$$G_{ro}(s) = \frac{(s - p_m)}{|p_m|} G(s),$$

where $p_m = -85784.92$, the largest negative pole. This model results in gain of $K_p = 0$ and $K_i = 4.34$.



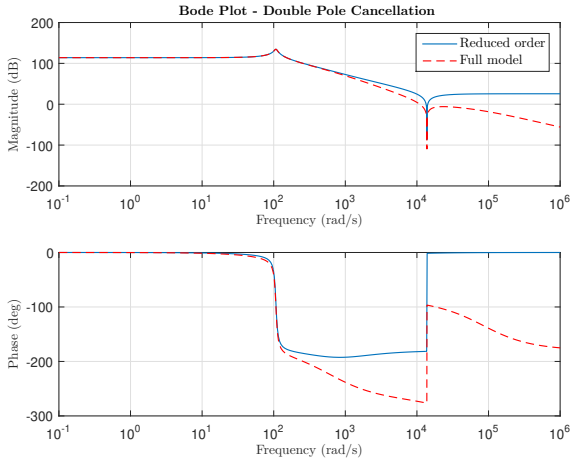
(a) Bode diagram.



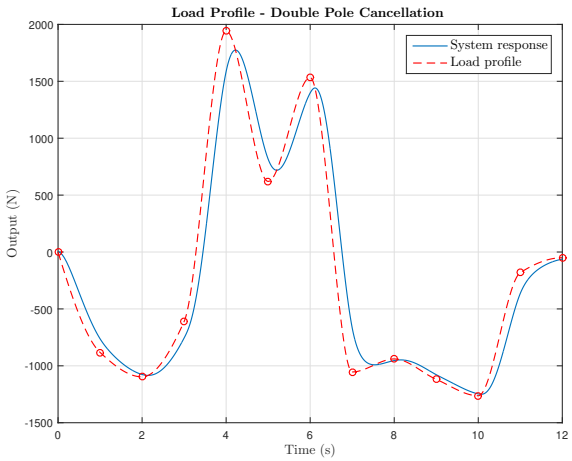
(b) Load profile.

Figure A.8: Single pole cancellation.

Double pole cancellation: The two least dominant poles were cancelled from the overall model, resulting in a biproper 3$^{\text{rd}}$ order transfer function. This yields $K_p = 0$ and $K_i = 4.35$.



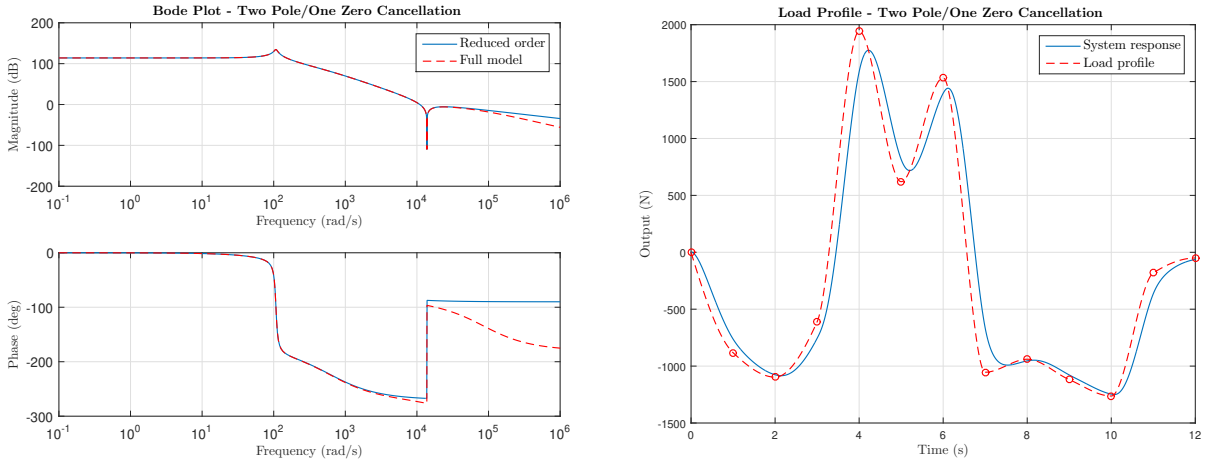
(a) Bode diagram.



(b) Load profile.

Figure A.9: Double pole cancellation.

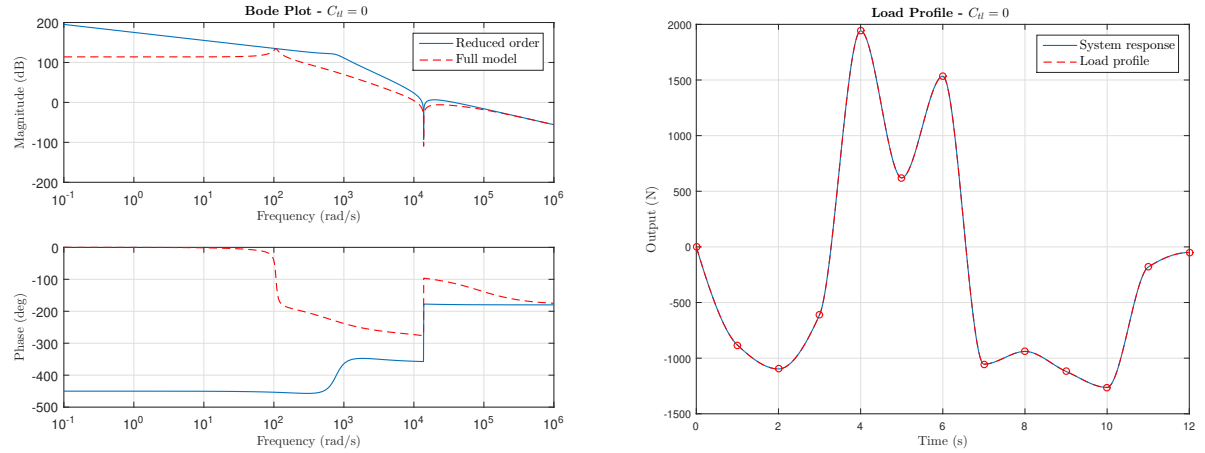
Two pole/one zero cancellation: The two least dominant poles and the least dominant zero were removed from the plant transfer function. This yields $K_p = 0$ and $K_i = 4.34$.



(a) Bode diagram. (b) Load profile.

Figure A.10: Two pole/one zero cancellation.

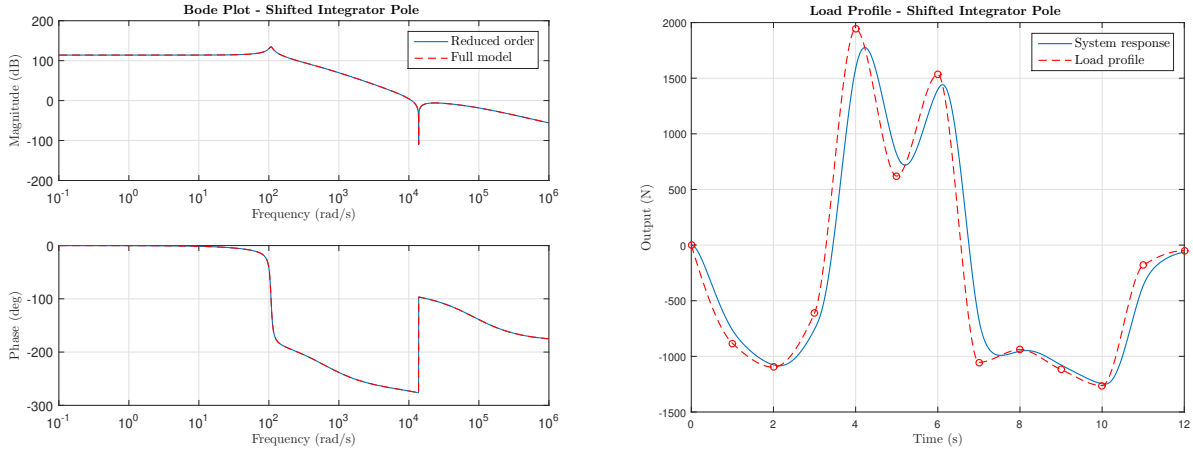
No leakage: The leakage coefficient C_{tl} was set to zero. The PID tuner outputs gains of $K_p = 5.029\text{e}08$, $K_i = 7.841\text{e}10$, and $K_d = 8.062\text{e}05$. Although the system is able to track the reference signal, the resulting closed-loop system has two poles that are slightly in the ORHP.



(a) Bode diagram. (b) Load profile.

Figure A.11: No leakage.

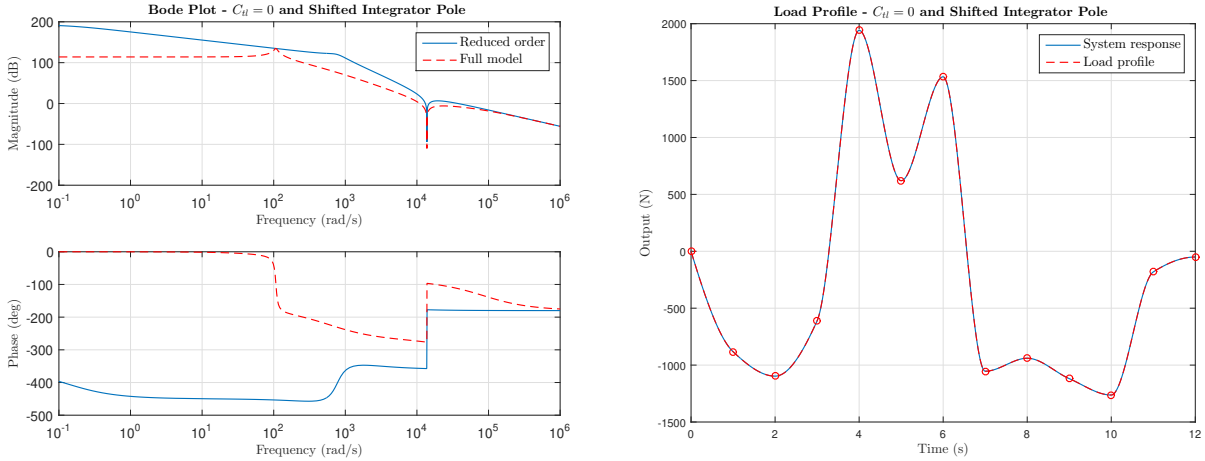
Shifted pole: The integrator pole in the $2\beta/Vs$ block was shifted so as not to be exactly at the origin. To shift the integrator pole, s was replaced by $s+\omega$, where $\omega = 10$. The tuner gains are $K_p = 0$ and $K_i = 4.34$.



(a) Bode diagram. (b) Load profile.

Figure A.12: Shifted integrator pole.

No leakage and shifted pole: The integrator pole in the $2\beta/Vs$ block was shifted so as not to be exactly at the origin, and the leakage coefficient C_{tl} was set to zero. The resulting PID tuner gains were $K_p = 5.109\text{e}05$, $K_i = 8.091\text{e}10$, and $K_d = 8.062\text{e}05$. Although the system is able to track the reference signal, the resulting closed-loop system has two poles that are slightly in the ORHP.



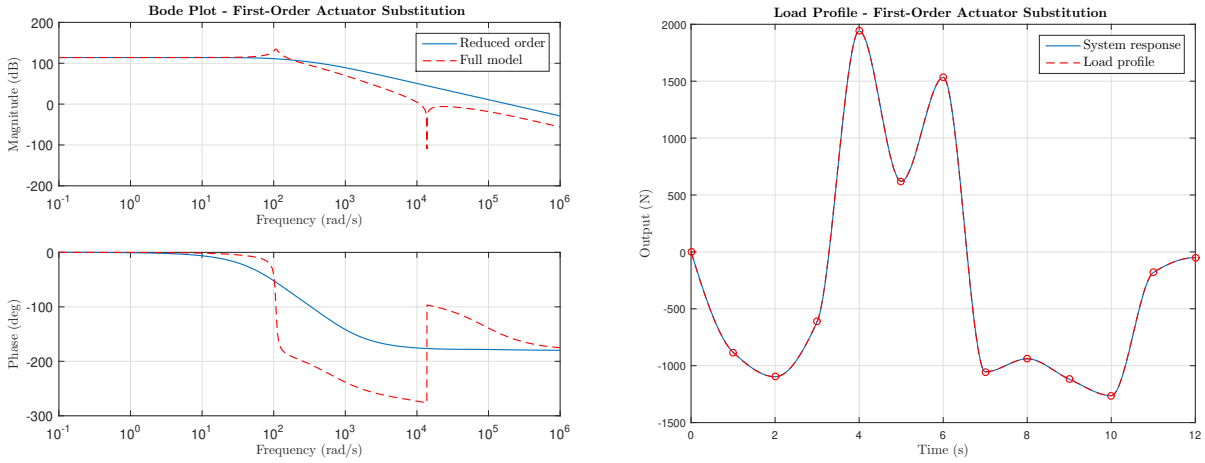
(a) Bode diagram. (b) Load profile.

Figure A.13: No leakage and shifted integrator pole.

First-order actuator substitution: Each of the two actuator transfer functions was substituted for a first-order system. The replacement transfer functions are

$$G_{a_L}(s) = \frac{398000}{0.009328s + 1}, \qquad G_{a_{\ddot{x}}}(s) = \frac{7.211}{1.194\text{e-}05s + 1}.$$

These transfer functions were found automatically using `MATLAB`. The resulting tuner gains are $K_p = 1.35$ and $K_i = 238.0$.



(a) Bode diagram.

(b) Load profile.

Figure A.14: First-order actuator substitution.

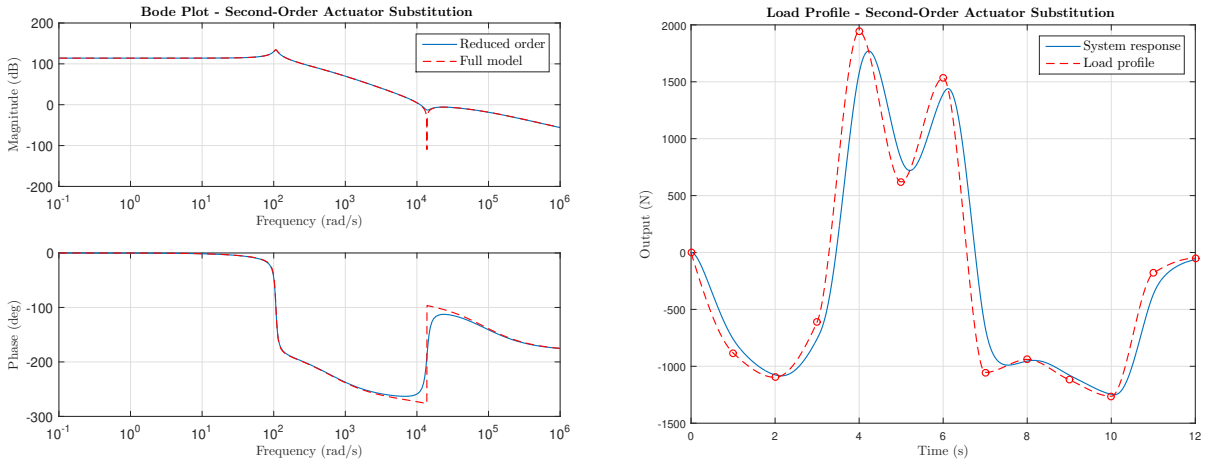
Second-order actuator substitution: The actuator transfer function from flow rate Q to load L was replaced by a second-order transfer function given by

$$G_{a_L}(s) = \frac{4.574\text{e}09}{s^2 + 10.72s + 1.149\text{e}04},$$

while the actuator transfer function from flow rate Q to acceleration $\ddot{x}$ was replaced by a first-order transfer function given by

$$G_{a_{\ddot{x}}}(s) = \frac{7.211}{1.194\text{e-}05s + 1}.$$

The resulting tuner gains are $K_p = 0$ and $K_i = 4.25$.



(a) Bode diagram.

(b) Load profile.

Figure A.15: Second-order actuator substitution.

Lower-frequency first-order actuator substitution: The actuator transfer function from flow rate Q to load L was replaced by a first-order transfer function given by

$$G_{a_L}(s) = \frac{398000}{0.1s + 1},$$

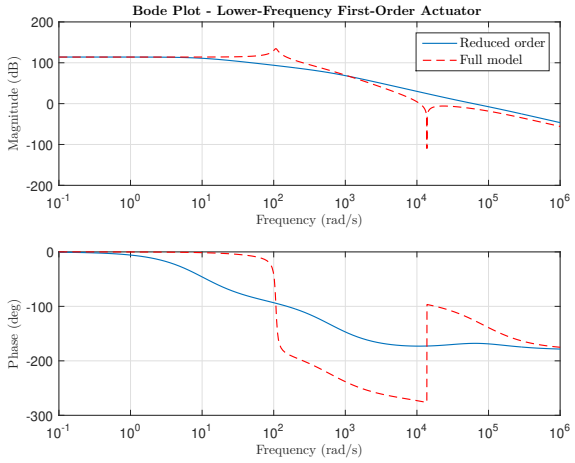
while the actuator transfer function from flow rate Q to acceleration $\ddot{x}$ was replaced by a first-order transfer function given by

$$G_{a_{\ddot{x}}}(s) = \frac{7.211}{1.194\text{e-}05s + 1}.$$

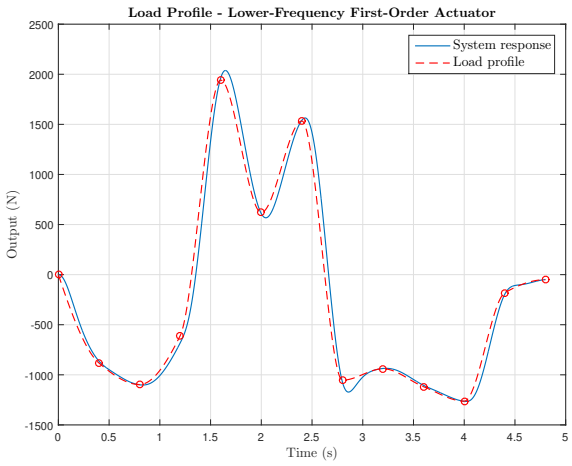
The `MATLAB` PID tuner yields $K_p = 0.96$ and $K_i = 27.1$. The closed-loop response using these gains is shown in Fig. A.16b.

The resulting plant transfer function was also used in the optimal static output feedback (SOF) controller synthesis method. The synthesized gains were $K_p = 724.4$ and $K_i = 6.178\text{e}04$ and the response is shown in Fig. A.16c. The static output feedback gains are much higher than the tuner gains, which translates to a faster response.

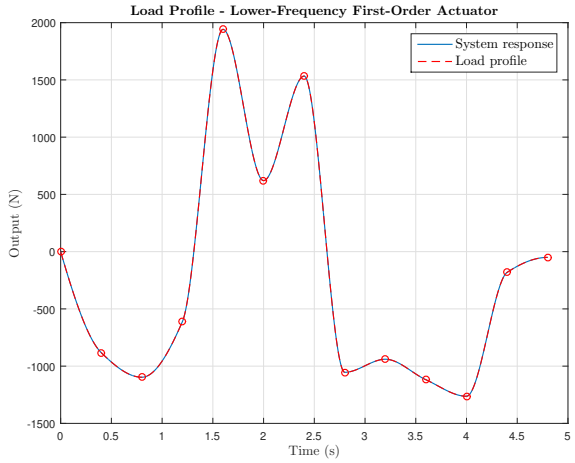
The iterative method also works when initialized with the PID tuner gains. This results in final gains of $K_p = 69.6$ and $K_i = 7.33\text{e}03$. The response is shown in Fig. A.16d. Because the response is realistic, this substitution is chosen as the order reduction method used in Section 3.1.2.3.



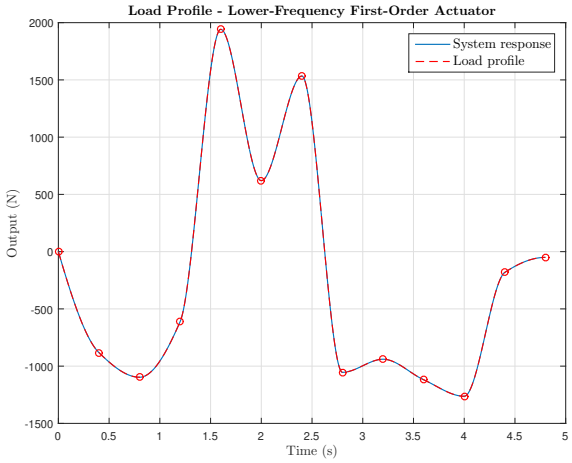
(a) Bode diagram.



(b) Fast load profile with tuner gains.



(c) Fast load profile with SOF gains.



(d) Fast load profile with iterative gains.

Figure A.16: Lower-frequency first-order actuator substitution.

Appendix B

Sample Code

A code generation template and sample MTS controller code for a MIMO 2DOF $\mathcal{H}_\infty$ controller acting on actuator 1 of the SHM platform are given below. The code generation template takes the controller state-space matrices and creates a text file with code that can then be copied into calculated channels on the MTS system.

B.0.1 Code Generation Template

```
function code_gen_fn_m_n(str,m,Cx,xs,A,B,C,D)

n = size(A,1);
ny = size(B,2);

fid = fopen(sprintf('CODE/hinf_%i_%s_%s.txt',m,str,Cx),'wt');
fprintf(fid, 'int init, i, j, k;\n');
fprintf(fid, sprintf('real A[%i], B[%i], C[%i], D[%i];\n',n^2,n*ny,n,ny));
fprintf(fid, sprintf('real  T, x_new[%i], x_old[%i], u[%i], out;\n',n,n,ny));

fprintf(fid, '\nT = 1.0/2048.0; // sample time\n');

fprintf(fid, '\nif (!init)\n{\n');

for i = 1:n
    for j = 1:n
        k = (i-1)*n+(j-1);
        fprintf(fid, sprintf('\tA[%i] = %0.2f;\n',k,A(i,j)));
    end
```

```
end
fprintf(fid, '\n');
for i = 1:n
    for j = 1:ny
        k = (i-1)*ny+(j-1);
        fprintf(fid, sprintf('\tB[%i] = %0.2f;\n',k,B(i,j)));
    end
end
fprintf(fid, '\n');
for j = 1:n
    fprintf(fid, sprintf('\tC[%i] = %0.2f;\n',j-1,C(j)/1.112));
end
fprintf(fid, '\n');
for j = 1:ny
    fprintf(fid, sprintf('\tD[%i] = %0.2f;\n',j-1,D(j)/1.112));
end
fprintf(fid, '\n\tinit = 1;\n}\n');

fprintf(fid, sprintf('\ni = 0;\nwhile (i < %i)\n{\n',n));
fprintf(fid, '\tx_new[i] = x_old[i];\n');
fprintf(fid, sprintf('\tj = 0;\n\twhile (j < %i)\n\t{\n',n));
fprintf(fid, sprintf('\t\tk = %i*i + j;\n',n));
fprintf(fid, '\t\tx_new[i] = x_new[i] + (T*A[k]) * x_old[j];\n');
fprintf(fid, '\t\tj = j + 1;\n\t}\n');
fprintf(fid, sprintf('\tj = 0;\n\twhile (j < %i)\n\t{\n',ny));
fprintf(fid, sprintf('\t\tk = %i*i + j;\n',ny));
fprintf(fid, '\t\tx_new[i] = x_new[i] + (T*B[k]) * u[j];\n');
fprintf(fid, '\t\tj = j + 1;\n\t}\n');
fprintf(fid, '\ti = i + 1;\n}\n');

fprintf(fid, sprintf('\ni = 0;\nwhile (i < %i)\n{\n',n));
fprintf(fid, '\tx_old[i] = x_new[i];\n');
fprintf(fid, '\ti = i + 1;\n}\n');

j = 0;
for i = xs:ny+xs-1
    Cs = sprintf('C%i',i);
    fprintf(fid, sprintf('\nu[%i] = "%s Command" - "%s Active Feedback";',j,Cs,Cs));
    j = j + 1;
end
fprintf(fid, '\n');
fprintf(fid, '\nout = 0;\n');
fprintf(fid, sprintf('i = 0;\nwhile (i < %i)\n{\n',n));
```

```
fprintf(fid, '\tout = out + C[i] * x_new[i];\n');
fprintf(fid, '\ti = i + 1;\n}\n');
fprintf(fid, sprintf('i = 0;\nwhile (i < %i)\n{\n',ny));
fprintf(fid, '\tout = out + D[i] * u[i];\n');
fprintf(fid, '\ti = i + 1;\n}\n');

fprintf(fid, '\n"output0" = out;');

end
```

B.0.2 MTS Controller Code

```
int init, i, j, k;
real A[4], Bff[4], Bfb[4], C[2], Dff[2], Dfb[2];
real  T, x_new[2], x_old[2], r[2], u[2], out;

T = 1.0/2048.0; // sample time

if (!init)
{
A[0] = -96.08;
A[1] = 150.35;
A[2] = -20.75;
A[3] = -74.36;

Bff[0] = -70.37;
Bff[1] = -38.69;
Bff[2] = 47.29;
Bff[3] = 37.88;

Bfb[0] = -4.07;
Bfb[1] = -35.35;
Bfb[2] = -26.05;
Bfb[3] = -28.30;

C[0] = -45.50;
C[1] = 9.20;

Dff[0] = 3.78;
```

```
Dff[1] = 8.40;

Dfb[0] = 8.74;
Dfb[1] = -9.04;

init = 1;
}

i = 0;
while (i < 2)
{
x_new[i] = x_old[i];
j = 0;
while (j < 2)
{
k = 2*i + j;
x_new[i] = x_new[i] + (T*A[k]) * x_old[j];
j = j + 1;
}
j = 0;
while (j < 2)
{
k = 2*i + j;
x_new[i] = x_new[i] + (T*Bff[k]) * r[j];
j = j + 1;
}
j = 0;
while (j < 2)
{
k = 2*i + j;
x_new[i] = x_new[i] + (T*Bfb[k]) * u[j];
j = j + 1;
}
i = i + 1;
}
```

```
i = 0;
while (i < 2)
{
x_old[i] = x_new[i];
i = i + 1;
}

r[0] = "C1 Command";
r[1] = "C2 Command";
u[0] = "C1 Command" - "C1 Active Feedback";
u[1] = "C2 Command" - "C2 Active Feedback";

out = 0;
i = 0;
while (i < 2)
{
out = out + C[i] * x_new[i];
i = i + 1;
}
i = 0;
while (i < 2)
{
out = out + Dff[i] * r[i];
i = i + 1;
}
i = 0;
while (i < 2)
{
out = out + Dfb[i] * u[i];
i = i + 1;
}

"output0" = out;
```

Bibliography

[1] F. Hansen, G. Franklin, and R. Kosut, "Closed-loop identification via the fractional representation: Experiment design," in *American Control Conference, 1989*, pp. 1422–1427, IEEE, 1989.

[2] Federal Aviation Administration, "14 CFR 25.571 Damage-tolerance and fatigue evaluation of structure," *Code of Federal Regulations*, 2017.

[3] K. J. Marsh, *Full-scale fatigue testing of components and structures*. London, UK: Butterworths, 2013.

[4] R. L. Hewitt, "State space modeling and experimental verification of a single channel, moving load cell structural test," *International Journal of Fatigue*, vol. 22, no. 9, pp. 767–780, 2000.

[5] R. L. Hewitt, "Modeling of full-scale aircraft structural tests," *ICAS 2002*, 2002.

[6] C. Cheung, "A *simulink* block library for modelling full-scale structural test systems," tech. rep., National Research Council Canada, March 2004.

[7] D. E. Zlotnik, "Identification and optimal control of a structural health monitoring test for a helicopter tail boom," tech. rep., National Research Council Canada, August 2013.

[8] MTS Systems Corporation, 14000 Technology Drive, Eden Prairie, MN 55344-2290 USA, *100-241-355 Servovalves*, 2014.

[9] J. R. Leigh, *Applied digital control: theory, design and implementation*. Mineola, NY: Dover Publications, 2006.

[10] R. Pintelon and J. Schoukens, *System identification: a frequency domain approach*. New York, NY: IEEE Press, 2012.

[11] K. J. Åström and B. Wittenmark, *Computer-controlled systems: Theory and design*. Mineola, NY: Dover Publications, 2011.

[12] T. C. Hsia, *System identification: least-squares methods*. Lexington, MA: Lexington Books, 1977.

[13] K. Lange, *Optimization*. New York, NY: Springer, 2013.

[14] R. J. Caverly and J. R. Forbes, "LMI properties and applications in systems, stability, and control theory," *arXiv preprint arXiv:1903.08599*, 2019.

[15] L. Qiu and K. Zhou, *Introduction to feedback control*. Upper Saddle River, NJ: Prentice-Hall, 2010.

[16] R. L. Williams and D. A. Lawrence, *Linear state-space control systems*. Hoboken, NJ: John Wiley & Sons, 2007.

[17] J. B. Hoagg, W. M. Haddad, and D. S. Bernstein, "Linear-quadratic control." Unpublished manuscript, 2019.

[18] L. Ljung, *System identification: Theory for the user*. Upper Saddle River, NJ: Prentice-Hall, 1999.

[19] D. Wang, R. Dolid, M. Donath, and J. Albright, "Development and verification of a two-stage flow control servovalve model," in *Proceedings of the 1995 ASME International Mechanical Engineering Congress and Exposition*, IEEE, 1995.

[20] S. G. Baek, H. K. Kim, K. T. Ahn, H. G. Yon, and J. C. Koo, "Study on iterative method of electro-hydraulic actuator in force control," in *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*, pp. 178–183, IEEE, 2012.

[21] A. L. Cologni, M. Mazzoleni, and F. Previdi, "Modeling and identification of an electro-hydraulic actuator," in *Control and Automation (ICCA), 2016 12th IEEE International Conference on*, pp. 335–340, IEEE, 2016.

[22] J. Zhu and S. Chang, "Modeling and simulation for application of electromagnetic linear actuator direct drive electro-hydraulic servo system," in *Power and Energy (PECon), 2012 IEEE International Conference on*, pp. 430–434, IEEE, 2012.

[23] M. Karpenko and N. Sepehri, "Fault-tolerant control of a servohydraulic positioning system with crossport leakage," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 1, pp. 155–161, 2005.

[24] L. Nascutiu, "Feedback linearization of the double-and single-rod hydraulic servo actuators," in *Automation, Quality and Testing, Robotics, 2006 IEEE International Conference on*, vol. 1, pp. 149–154, IEEE, 2006.

[25] D. Williams, M. Williams, and A. Blakeborough, "Numerical modeling of a servohydraulic testing system for structures," *Journal of engineering mechanics*, vol. 127, no. 8, pp. 816–827, 2001.

[26] M. Rahmat, S. M. Rozali, N. A. Wahab, and K. Jusoff, "Modeling and controller design of an electro-hydraulic actuator system," *American Journal of Applied Sciences*, vol. 7, no. 8, p. 1100, 2010.

[27] V. Durbha and P. Li, "A nonlinear spring model of hydraulic actuator for passive controller design in bilateral tele-operation," in *American Control Conference (ACC), 2012*, pp. 3471–3476, IEEE, 2012.

[28] D. W. Robinson and G. A. Pratt, "Force controllable hydro-elastic actuator," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2, pp. 1321–1327, IEEE, 2000.

[29] A. Alleyne and R. Liu, "A simplified approach to force control for electro-hydraulic systems," *Control Engineering Practice*, vol. 8, no. 12, pp. 1347 – 1356, 2000.

[30] MTS Systems Corporation, 14000 Technology Drive, Eden Prairie, MN 55344-2290 USA, *100-361-218 HydraulicActuators 244*, 2017.

[31] S. S. Rao and F. F. Yap, *Mechanical vibrations*, vol. 4. Upper Saddle River, NJ: Prentice-Hall, 2011.

[32] D. H. Hodges and G. A. Pierce, *Introduction to Structural Dynamics and Aeroelasticity*. New York, NY: Prentice-Hall, Inc., 2 ed., 2011.

[33] A. C. Ugural and S. K. Fenster, *Advanced strength and applied elasticity*. Prentice-Hall, 2003.

[34] K. F. Aljanaideh and D. S. Bernstein, "Initial conditions in time-and frequency-domain system identification: Implications of the shift operator versus the Z and discrete Fourier transforms," *IEEE Control Systems*, vol. 38, no. 2, pp. 80–93, 2018.

[35] P. Van den Hof, "System identification - Data-driven modelling of dynamic systems." Lecture notes, Eindhoven University of Technology, March 2018.

[36] U. Forssell and L. Ljung, "Closed-loop identification revisited," *Automatica*, vol. 35, no. 7, pp. 1215–1241, 1999.

[37] D. Youla, H. Jabr, and J. Bongiorno, "Modern wiener-hopf design of optimal controllers - Part II: The multivariable case," *IEEE Transactions on Automatic Control*, vol. 21, no. 3, pp. 319–338, 1976.

[38] C. Desoer, R.-W. Liu, J. Murray, and R. Saeks, "Feedback system design: The fractional representation approach to analysis and synthesis," *IEEE Transactions on Automatic Control*, vol. 25, no. 3, pp. 399–412, 1980.

[39] F. Hansen, "A fractional representation approach to closed-loop system identification and experiment design," *Ph. D. Thesis, Stanford University*, 1989.

[40] D. T. Westwick and R. E. Kearney, *Identification of nonlinear physiological systems*, vol. 7. John Wiley & Sons, 2003.

[41] R. Kearney and I. Hunter, "System identification of human triceps surae stretch reflex dynamics," *Experimental brain research*, vol. 51, no. 1, pp. 117–127, 1983.

[42] K. J. Åström and T. Hägglund, *Advanced PID Control*, vol. 461. Research Triangle Park, NC: Instrumentation, Systems, and Automation Society, 2006.

[43] L. El Ghaoui, F. Oustry, and M. AitRami, "A cone complementarity linearization algorithm for static output-feedback and related problems," in *Proceedings of Joint Conference on Control Applications Intelligent Control and Computer Aided Control System Design*, pp. 246–251, IEEE, 1998.

[44] K. J. Åström and T. Hägglund, *PID Controller: Theory, Design and Tuning*. North Carolina: Instrumentation Society of America, 1995.

[45] L. Silverman, "Inversion of multivariable linear systems," *IEEE Transactions on Automatic Control*, vol. 14, no. 3, pp. 270–276, 1969.

[46] P. Martin, S. Devasia, and B. Paden, "A different look at output tracking: Control of a VTOL aircraft," *Automatica*, vol. 32, no. 1, pp. 101–107, 1996.

[47] P. Misra and R. Patel, "Transmission zero assignment in linear multivariable systems - I: Square systems," in *Proceedings of the 27th IEEE Conference on Decision and Control*, pp. 1310–1311, IEEE, 1988.

[48] S. Skogestad and I. Postlethwaite, *Multivariable feedback control: Analysis and design*, vol. 2. Wiley New York, 2001.

[49] C. Scherer, P. Gahinet, and M. Chilali, "Multiobjective output-feedback control via LMI optimization," *IEEE Transactions on automatic control*, vol. 42, no. 7, pp. 896–911, 1997.

[50] M. Peet, "Modern optimal control." Lecture notes, Arizona State University.