# NUMERICAL COMPUTATION AND IMPLEMENTATION OF H-INFINITY CONTROLLERS FOR RETARDED AND NEUTRAL TIME DELAY SYSTEMS

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

By

Mustafa Oğuz Yeğin

Nov 2017

NUMERICAL COMPUTATION AND IMPLEMENTATION OF H-INFINITY CONTROLLERS FOR RETARDED AND NEUTRAL TIME DELAY SYSTEMS

By Mustafa Oğuz Yeğin

Nov 2017

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Hitay Özbay(Advisor)

Arif Bülent Özgüler

Coşku Kasnakoğlu

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

# ABSTRACT

## NUMERICAL COMPUTATION AND IMPLEMENTATION OF H-INFINITY CONTROLLERS FOR RETARDED AND NEUTRAL TIME DELAY SYSTEMS

Mustafa Oğuz Yeğin
M.S. in Electrical and Electronics Engineering
Advisor: Hitay Özbay
Nov 2017

Being the most commonly encountered systems in engineering applications, design of robustly stabilizing $\mathcal{H}_\infty$ controller with maximum performance for time-delay linear time-invariant systems has been an attractive topic in control theory for many decades.

A Matlab package called HINFCON is published in 1996. It applies an early solution to the problem, the so-called Toker-Özbay formula. It assumes that the coprime factorization is given, and computes the optimal performance level and the corresponding optimal controller through a manual iteration. The software introduced in this thesis, combines HINFCON with another Matlab package, YALTA, that is being used to do stability analysis for neutral/retarded systems, to perform coprime and inner-outer factorizations of the given infinite dimensional system. Additionally, an iterative algorithm for computation of optimal performance level automatically is implemented to prevent any manual computation. Finally, the Matlab command `fitfrd` is incorporated into the software that is used for approximation of the optimal controller to obtain a robustly stabilizing suboptimal finite dimensional controller. This thesis also introduces a new structure of the optimal controller which is more reliable and implementable on practical systems. Various examples are solved to compare the suboptimal controllers obtained by using the new structure with direct approximation of plant and optimal controller.

*Keywords:* Robust control, $\mathcal{H}_\infty$ controller, infinite dimensional systems, time-delay systems, neutral and retarded systems.

# ÖZET

# NÖTR VE RÖTARLI ZAMAN GECİKMELİ SİSTEMLER İÇİN H-SONSUZ KONTROLCULARIN HESAPLANMASI VE UYGULANMASI

Mustafa Oğuz Yeğin
Elektrik ve Elektronik Mühendisliği, Yüksek Lisans
Tez Danışmanı: Hitay Özbay
Kasım 2017

Mühendislik uygulamalarında en çok karşılaşılmakta olan zaman gecikmeli doğrusal zamanlı değişmez sistemlere, en yüksek performanslı gürbüz kararlılık sağlayan $\mathcal{H}_\infty$ kontrolcu tasarımı, kontrol teorisinde onlarca yıldır ilgi çekici bir konu olmuştur.

HINFCON ismindeki Matlab pakedi 1996'da yayımlanmıştır. Bu paket bu probleme Toker-Özbay formülü olarak adlandırılan eski bir çözümü uygulamaktadır. HINFCON, aralarında asal çarpanlara ayrılmış terimlerin verilmesine bağlı olarak, manuel yineleme ile optimal performans seviyesi ve buna tekabül eden optimal kontrolcuyu hesaplamaktadır. Bu tezde tanıtılan program, HINFCON ve nötr/gecikmeli sistemlerin kararlılık analizini yapan bir başka Matlab pakedi YALTA'yı, verilen sistemin, aralarında asal çarpanlara ayrılmış terimlerini bulmak için birleştirmektedir. Buna ek olarak, elle herhangi bir hesaplama yapılmasını engellemek için optimal performans seviyesini otomatik olarak hesaplayan, tekrarlayan bir algoritma uygulanmıştır. Son olarak, programa optimale yakın, gürbüz, sınırlı boyutlu kontrolcu elde edilmesi amacıyla Matlab komutu `fitfrd` eklenmiştir. Bu tez ayrıca, daha güvenilir ve işlek sistemlere daha uygulanabilir yeni bir optimal kontrolcu yapısı tanıtmaktadır. Yeni yapı kullanılarak elde edilen optimale yakın kontrolcularla, sistemin veya optimal kontrolcunun yakınsanmasıyla elde edilen kontrolcuları karşılaştırmak amacıyla farklı örnekler çözülmüştür.

*Anahtar sözcükler*: Gürbüz kontrol, $\mathcal{H}_\infty$ kontrolcü, sonsuz boyutlu sistemler, zaman-gecikmeli sistemler, nötr ve gecikmeli sistemler.

# Acknowledgement

I would like to begin with expressing my sincerest gratitudes to my supervisor, Professor Hitay Özbay. I would like to thank him for his understanding, support and guidance throughout my undergraduate thesis, graduate education and studies. He was the advisor just I wished and I am grateful that I could have the chance of being one of his students. I would also like to thank Professor Arif Bülent Özguler and Associate Professor Coşku Kasnakoğlu for serving in my thesis jury.

I would also like to thank Inria DISCO team for making early, and later updated, versions of YALTA available upon special request by Prof. Hitay Özbay.

I want to express my gratitudes to our Department Chair Professor Orhan Arıkan, who had been my advisor throughout my undergraduate education, for guiding, encouraging and supporting me throughout my academic career.

Special thanks to Onur Albayrak and İlim Karaçal of ASELSAN for making their experimental facilities available for implementation and testing of the algorithms developed in this thesis on a real life antenna control system. I would also like to thank Dilan Öztürk, Caner Odabaş and Elvan Kuzucu, the members of our research group.

My special thanks are for my parents, Ayşenur and Haluk Yeğin, and my brother, Mehmet Emre Yeğin, who helped and assisted me at every stage in my life.

I would like to thank İsmail Uyanık, Eftun Orhon, Mustafa Gül, Mansur Arısoy, Bahadır Çatalbaş, Hasan Hamzaçebi and Deniz Kerimoğlu for making valuable contributions on my development in my research area. I would also like to thank Saeed Ahmed, Okan Demir, Ali Nail İnal and Meysam Ghomi, members of control area research group, for helping me along the way.

I want to thank Mürüvet Parlakay and Aslı Tosuner for their helps on administrative works and Ergün Hırlakoğlu, Onur Bostancı and Ufuk Tufan for their technical support.

Outside the laboratory, there are some friends who directly or indirectly contributed to my thesis. I express my special thanks to Hamdi Armağan Gürdal, for his support and encouragement through my graduate studies. I am grateful to my undergraduate friends Orçun Akkoç and Mustafa Orman for assisting me throughout my graduate education.

I want to thank my non-departmental friends Alihan Sağır and Ali Sinan Kalpaklı for spending their time with me.

I would like to acknowledge that this work was supported by a scholarship from TUBITAK project EEEAG-115E820.

# Contents

# List of Figures

# Chapter 1

# Introduction

Model based controller design for a physical system is one of the most common methods used by engineers. However, mathematical model of a system always introduces errors because of the simplifications, assumptions and neglected parameters. Therefore, stability and performance against uncertainties that exist due to these errors has been significant criteria for the controller design, which are concerned within the robust control, an important field in the feedback control theory.

Since $\mathcal{H}_\infty$ norm reflects the worst-case gain of the system, it is one of the most widely used tool in the robust controller design. Two-block problem deals with both robust stability and robust performance problems by using $\mathcal{H}_\infty$ norm of weighted sensitivity and weighted closed loop transfer function. The first mentioned weight represents the pre-specified performance level with respect to frequency and the other represents the uncertainty structure, which in this case is chosen as additive or multiplicative uncertainty bound. By using the solution of the mixed sensitivity minimization problem, an $\mathcal{H}_\infty$ controller can be designed which has a specific structure based on the given plant and weights. One of the major objectives of this thesis is to investigate this structure and propose ways to approximate its infinite dimensional terms.

Due to the fact that many practical systems are infinite dimensional, design of a robustly stable controller of such systems has been an attractive topic for engineers and mathematicians for the last few decades. Fractional systems, systems with time delay and spatially distributed systems are the most widely encountered infinite dimensional systems in the literature. To design an $\mathcal{H}_\infty$ controller that robustly stabilizes the system with high performance, mixed sensitivity minimization problem needs to be solved for infinite dimensional systems. One of the first numerical computation methods for two block problem proposed in the literature is Toker-Özbay formula [7]. This thesis introduces a software that uses Toker-Özbay formula to numerically compute the optimal $\mathcal{H}_\infty$ controller for single input-single output (SISO) retarded/neutral infinite dimensional systems and returns a finite order suboptimal controller that has similar structure with the optimal controller at low frequencies.

The rest of this chapter contains a brief review of literature on optimal $\mathcal{H}_\infty$ controller design by solving weighted sensitivity minimization and two-block problems respectively and the outline of the thesis.

## 1.1 Existing Work on Computation of Optimal $\mathcal{H}_\infty$ Controller for Time-Delay Systems

In early 1980s, there had been various methods for the solution of weighted sensitivity minimization of finite dimensional linear time-invariant (LTI) systems, which only uses a single weight to compute optimal sensitivity and controller [1, 2, 3]. These methods had been extended to solve the one-block problem for time-delay infinite dimensional systems [4, 5, 6]. In the later years, this has been extended to two-block problems and in the 1990s Toker-Özbay formula is introduced to solve two-block problem for SISO time-delay infinite dimensional systems and rational weight functions [7].

The optimal controller structure introduced in Toker-Özbay formula is improved in various works [8, 9] so that the unstable and stable parts of the function are split into different terms. In [8], infinite dimensional part of the optimal controller is represented by a finite impulse response filter (FIR). Both improvements allow easier and safer approximation of infinite dimensional optimal controller and provides the closed loop system with finite dimensional suboptimal controller to be less fragile and more reliable.

Key step in Toker-Özbay is to have the inner-outer factorization which requires finding $\mathbb{C}_+$ poles and zeros of the infinite dimensional transfer functions. Recently, for fractional and time delay system of retarded and neutral type, a software has been released by INRIA: YALTA will be used in this thesis [10]. This Matlab tool is used for stability analysis of fractional retarded/neutral systems and able to return roots at $\mathbb{C}_+$ of a given quasi-polynomial.

The main contribution of the thesis is a Matlab based software for the computation of the optimal $\mathcal{H}_\infty$ controllers for systems handled by YALTA. The program gives the optimal performance level and the optimal controller. Typically, optimal controller is infinite dimensional. The thesis also examines the structure of the optimal controller and proposes a method for its implementation by using finite dimensional subsystems.

## 1.2 Thesis Outline

In chapter 2, the definition of mixed sensitivity minimization problem is described. The explanation of Toker-Özbay formula, the decomposition of optimal controller into a new structure and a bounding method for suboptimal performance level are covered in chapter 3. The implementation of the formula and usage of YALTA are described in chapter 4. This chapter also includes graphical user interface of the final software. In chapter 5, various examples are given. One of them compares the performance levels of suboptimal controllers obtained by using Pade approximation and `fitfrd` with the controllers obtained from the software, by

using a range of orders for approximation. Finally, conclusion of the thesis and future work are given in chapter 6.

# Chapter 2

# Formulation of the Mixed Sensitivity Minimization Problem for Infinite Dimensional Systems

In this chapter, the mixed sensitivity minimization problem is introduced for the feedback system shown in Figure 2.1, where the controller and uncertain plant are represented by $C$ and $P_\Delta$ respectively. The signals shown in the block diagram are described below:

- $r(t)$: Reference signal
- $e(t)$: Tracking error (difference between input and output signals)
- $u(t)$: Output of the controller
- $v(t)$: Disturbance signal
- $y(t)$: Output of the system

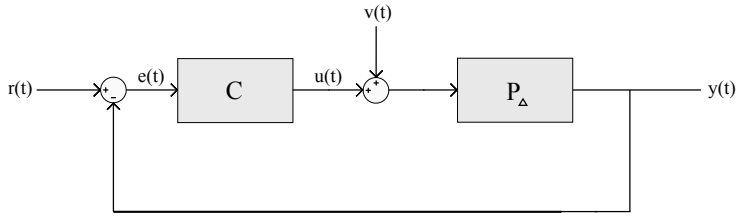


Figure 2.1: Feedback system with controller $C$ and plant $P_\Delta$

We can define a nominal plant ($P$) that covers the significant dynamics of uncertain plant $P_\Delta$. Then, the mixed sensitivity minimization problem is defined as

$$\gamma = \inf_{(C,P) \text{ stable}} \left\| \begin{bmatrix} W_1(1+PC)^{-1} \\ W_2PC(1+PC)^{-1} \end{bmatrix} \right\|_\infty \tag{2.1}$$

where $W_1$ and $W_2$ are the sensitivity and uncertainty weights respectively, and $\gamma_{\text{opt}}$ is called as the optimal performance level. We denote the feedback system formed by the controller $C$ and the plant $P$ as $(C, P)$. The feedback system is stable if all the poles of the closed loop transfer functions from $(r, v)$ to $(u, y)$ are in $\mathbb{C}_-$. Therefore, the optimal $\mathcal{H}_\infty$ controller ($C_{opt}$) both minimizes (2.1) and stabilizes the system.

The sensitivity weight ($W_1$) is determined according to reference signal. For example, if the system needs to track step input, an outer[1] function is defined in such a way that has a similar frequency response with the step input

$$W_1(s) = \frac{1}{s+\epsilon}$$

where $\epsilon << 1$.

The uncertainty weight ($W_2$) is also an outer function that bounds the multiplicative error between uncertain plant and the nominal plant:

$$|W_2(j\omega)| \geq \frac{|P_\Delta(j\omega) - P(j\omega)|}{|P_\Delta(j\omega)|} \quad \forall \omega \in [0\ \infty)$$

[1]An outer function is a minimum phase transfer function, i.e. a function which does not have any poles or zeros in the right half plane

# Chapter 3

# Numerical Computation of $\mathcal{H}_\infty$ Optimal Controller

This chapter briefly explains the constrains and steps of Toker-Özbay formula, which gives a solution to the mixed sensitivity minimization problem defined above. In third section, another form of the optimal controller is introduced to be used in practical systems. The terms in the proposed solution in [7] are decomposed in such a way that finite dimensional unstable and infinite dimensional stable parts are represented by different terms, which provide the implementation of the controller to become more reliable. Furthermore, a bound on the suboptimal performance level is introduced in the final section of this chapter.

## 3.1 Constraints and Assumptions on the Problem Data

Toker-Özbay formula is applicable under following constraints: The plant must admit a coprime factorization of the form (Eq. 3.1), the weights $W_1$ and $W_2$ need to be rational, where $W_1(s)$ is non-constant and $W_1(s)$, $(W_2(s)N_o(s))$, $(W_2(s)N_o(s))^{-1} \in \mathcal{H}_\infty$. Detailed discussion of these constrains can be found

in [11].

Initially, the plant needs to be factorized as

$$P(s) = \frac{M_n(s)N_o(s)}{M_d(s)}, \tag{3.1}$$

where $M_n(s)$ and $M_d(s)$ are inner (all pass) functions and $N_o(s)$ is an outer (minimum phase) function. Specifically, zeros of $M_n(s)$ are the zeros of $P(s)$ in the right half plane (it is infinite dimensional if there is an input/output time delay or if there are infinitely many zeros in the right half plane) and zeros of $M_d(s)$ are the unstable poles of $P(s)$. Therefore, the factorized terms can be represented as follows:

$$P(s) = \frac{Ke^{-hs}(s-z_1)(s-z_2)(s-z_3)\dots}{(s-p_1)(s-p_2)(s-p_3)\dots},$$

where $z$ and $p$ represent the zeros and poles of the plant respectively. Both zeros and poles can be infinitely many, but the number of unstable poles must be finite. Let $\alpha_i$ be $i^{th}$ unstable pole ($\alpha_i \in \mathbb{C}_+$), $n$ be the number of unstable poles, $\eta_i$ be $i^{th}$ unstable zero and $m$ be the number of unstable zeros. Then, the functions introduced for factorized form become

$$\begin{aligned} M_n(s) &= \frac{e^{-hs}(s-\eta_1)(s-\eta_2)\dots(s-\eta_m)}{(s+\eta_1)(s+\eta_2)\dots(s+\eta_m)},\ h \geq 0,\ Re(\eta_i) > 0 \\ M_d(s) &= \frac{(s-\alpha_1)(s-\alpha_2)\dots(s-\alpha_n)}{(s+\alpha_1)(s+\alpha_2)\dots(s+\alpha_n)}. \end{aligned}$$

In this work, we assume that $n$ and $m$ are finite and $\alpha_i$'s are distinct. Since combination of zeros of $M_n(s)$ and $M_d(s)$ involves all the the zeros and poles of $P(s)$ in $\mathbb{C}_+$ and their poles are in $\mathbb{C}_-$, $N_o(s)$ becomes an outer function.

The weights need to satisfy conditions explained in the beginning of this section. Since $W_1(s)$ and $W_2(s)$ are not factorized as plant, they both need to be rational and outer. Furthermore, as described in the following section, spectral factorization is applied in the formula; moreover, in order to have a proper controller we must have, $(W_2(s)N_o(s))$, $(W_2(s)N_o(s))^{-1} \in \mathcal{H}_\infty$.

## 3.2 Computation of the $\mathcal{H}_\infty$ Optimal Performance Level

Steps of Toker-Özbay formula for computing $\gamma_{opt}$ are explained in this part. The method introduces functions that depend on $\gamma$ in Eq. (2.1) and weights $W_1(s)$, $W_2(s)$. These functions are defined according to results obtained by using the so called AAK theory [12] and estimating simplified terms by observation, as similarly done in [13]. More detail regarding the proofs can be found in the original paper introducing Toker-Özbay formula [7].

The steps introduced below are iteratively followed to find neighborhood of $\gamma_{opt}$ according to the given tolerance value. The procedure of the implemented algorithm is explained in the next chapter in detail.

The first function defined in Toker-Özbay formula is

$$E_\gamma(s) = \frac{W_1(s)W_1(-s)}{\gamma^2} - 1 = \frac{nE_\gamma(s)}{\gamma^2 dW_1(s)dW_1(-s)}, \tag{3.2}$$

where $W_1(s) = nW_1(s)/dW_1(s)$ (since $W_1(s)$ is rational). The formula assumes that $W_1(s)$ is a proper function, with an order of $n_1$. Moreover, the zeros of $E_\gamma(s)$, $(\beta_1, \ldots, \beta_{2n_1})$ need to be enumerated as

$$-\beta_{n_1+k} = \beta_k \in \overline{\mathbb{C}}_+ \text{ for } k = 1, \ldots, n_1$$

for further simplifications in the formula. Note that each $\beta_k$ depends on $\gamma$ and the formula assumes that $\beta_k$ are distinct roots at $\gamma = \gamma_{opt}$. In fact, this can be achieved by using a small perturbation on problem data to change $\gamma_{opt}$ value.

The formula continues with the definition of two rational functions, both using $W_1(s)$, $W_2(s)$, and $\gamma > 0$

$$F_\gamma(s) = \gamma \frac{dW_1(-s)}{nW_1(s)} G_\gamma(s), \tag{3.3}$$

where $G(s)$ is obtained by using spectral factorization

$$G_\gamma(-s)G_\gamma(s) = \left(1 + \frac{W_2(-s)W_2(s)}{W_1(-s)W_1(s)} - \frac{W_2(-s)W_2(s)}{\gamma^2}\right)^{-1}. \tag{3.4}$$

By using the terms above, plant factorization and another function to be introduced, the optimal controller is expressed as

$$C_{opt}(s) = E_\gamma(s)M_d(s)\frac{F_\gamma(s)L(s)}{1 + M_n(s)F_\gamma(s)L(s)}N_o^{-1}(s),$$

where $\gamma = \gamma_{opt}$ and $L(s)$ is found by using unstable poles of $P(s)$ ($\alpha_i$ for $i = 1, \ldots, n$) and zeros of $E_\gamma(s)$ ($\beta_i$ for $i = 1, \ldots, n_1$). Since the definition of $L(s)$ in original paper [7] is complicated and the formula is revisited in [14] by using simpler expressions, $L(s)$ is introduced by following the revisited formula.

$$L(s) = \frac{[1 \ s \ldots s^{\eta-1}]\Psi_2}{[1 \ s \ldots s^{\eta-1}]\Psi_1}, \quad \eta := n + n_1, \tag{3.5}$$

where the coefficient vectors,

$$\Psi_1 = [\psi_{11} \ \ldots \psi_{1\eta}] \text{ and } \Psi_2 = [\psi_{21} \ \ldots \psi_{2\eta}],$$

are calculated by the linear equations obtained from interpolation conditions given in [7]. These linear equations can be expressed in matrix form by defining a $k \times k$ diagonal matrix, $\mathfrak{J}_\mathfrak{k}$, whose $i$th diagonal entry is $(-1)^{i+1}$ and Vandermonde matrices. Given a vector $\vec{x} = [x_1, \ldots, x_k]^{\mathrm{T}} \in \mathbb{C}^k$ with distinct elements ($x_i \neq x_j$ for $i \neq j$) and a positive integer $m \geq 1$, the Vandermonde matrix is defined as

$$\mathcal{V}_x^m := \begin{bmatrix} 1 & x_1 & \ldots & x_1^{m-1} \\ 1 & x_2 & \ldots & x_2^{m-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_k & \ldots & x_k^{m-1} \end{bmatrix}_{k \times m}.$$

By using this specific matrix expression, the revisited formula defines $\mathcal{V}_\alpha^n$ and $\mathcal{V}_\beta^n$, where $\vec{\alpha} = [\alpha_1 \ \ldots \ \alpha_n]$ and $\vec{\beta} = [\beta_1 \ \ldots \ \beta_{n_1}]$, and assembles these matrices to form a square matrix

$$\mathcal{V}_\eta := \begin{bmatrix} \mathcal{V}_\alpha^\eta \\ \mathcal{V}_\beta^\eta \end{bmatrix}.$$

Furthermore, the following diagonal matrices are defined for the sake of simplicity of the resulting matrix:

$$\begin{aligned} \mathcal{D}_n &= \text{diag}\{M_n(\alpha_1)F_\gamma(\alpha_1), \ldots, M_n(\alpha_n)F_\gamma(\alpha_n)\} \\ \mathcal{D}_{n_1} &= \text{diag}\{M_n(\beta_1)F_\gamma(\beta_1), \ldots, M_n(\beta_{n_1})F_\gamma(\beta_{n_1})\} \\ \mathcal{D}_\eta &= \text{block diag}\{\mathcal{D}_n, \mathcal{D}_{n_1}\} \end{aligned}$$

The interpolation conditions and the $M_\gamma$ matrix in [7] can be simplified into

$$T(\gamma)\Psi = 0, \tag{3.6}$$

where

$$\mathcal{T}(\gamma) := \begin{bmatrix} \mathcal{V}_\eta & \mathcal{D}_\eta\mathcal{V}_\eta \\ \mathcal{D}_\eta\mathcal{V}_\eta\mathfrak{J}_\eta & \mathcal{V}_\eta\mathfrak{J}_\eta \end{bmatrix}, \quad \Psi := \begin{bmatrix} \Psi_1 \\ \Psi_2 \end{bmatrix}$$

and $\gamma_{\text{opt}}$ is the largest $\gamma$ that makes $\mathcal{T}(\gamma)$ singular. Furthermore, $L(s)$ appearing in the optimal controller is constructed from the corresponding singular vector $\Psi$, as defined in (3.5). To obtain $\gamma_{\text{opt}}$, samples from an interval for $\gamma$ must be used and the resulting $\gamma$ vs $\sigma(\mathcal{T}(\gamma))$ (smallest singular value of $\mathcal{T}(\gamma)$) plot needs to be drawn. By inspection, an approximation of $\gamma_{opt}$ can be obtained, or this operation can be done iteratively by using a smaller interval around the largest gamma that makes $\sigma(\mathcal{T}(\gamma)) \approxeq 0$. This procedure is implemented such a way that $\gamma_{\text{opt}}$ is calculated automatically after a single interval for $\gamma$ is given [9], and the implementation is explained in detail in the next chapter.

## 3.3 Computation of the $\mathcal{H}_\infty$ Optimal Controller

Toker-Özbay formula introduces an equation for optimal controller, which uses weights, factorized terms, $\gamma_{\text{opt}}$, and the terms defined in Section 3.2:

$$C_{opt}(s) = \frac{M_d(s)E_{\gamma_{\text{opt}}}(s)F_{\gamma_{\text{opt}}}(s)L_{\gamma_{\text{opt}}}(s)N_o^{-1}(s)}{1 + M_n(s)F_{\gamma_{\text{opt}}}(s)L_{\gamma_{\text{opt}}}(s)}. \tag{3.7}$$

It should be noted that there are internal unstable pole-zero cancellations in this expression. In order to obtain an alternative expression for the controller, free of such cancellations, we first assume that $W_1$ is bi-proper and introduce the following: $K_{\gamma_{\text{opt}}}(s) = L^{-1}_{\gamma_{\text{opt}}}(s)$ and $M_1(s) = dW_1(-s)/dW_1(s)$, where $dW_1(s)$ notates denominator polynomial of $W_1(s)$. Furthermore, let

$$\hat{G}_{\gamma_{\text{opt}}}(-s)\hat{G}_{\gamma_{\text{opt}}}(s) = \left(W_1(-s)W_1(s) - W_2(-s)W_2(s)E_{\gamma_{\text{opt}}}(s)\right)^{-1},$$

where $\hat{G}_{\gamma_{\text{opt}}}(s)$ is an outer function to be found by using spectral factorization.

Then, the terms defined in (3.4) and (3.3) can be re-written as follows respectively:

$$G_{\gamma_{\text{opt}}}(s) = W_1(s)\hat{G}_{\gamma_{\text{opt}}}(s), \quad F_{\gamma_{\text{opt}}}(s) = \gamma_{\text{opt}} M_1(s)\hat{G}_{\gamma_{\text{opt}}}(s).$$

By using $K_{\gamma_{\text{opt}}}(s)$ and replacing $F_{\gamma_{\text{opt}}}(s)$ with its new expression in (3.7), optimal controller can be expressed as

$$C_{opt}(s) = \frac{\gamma_{\text{opt}} M_d(s) E_{\gamma_{\text{opt}}}(s) M_1(s)\hat{G}_{\gamma_{\text{opt}}}(s) N_o^{-1}}{K_{\gamma_{\text{opt}}}(s) + \gamma_{\text{opt}} M_n(s) M_1(s)\hat{G}_{\gamma_{\text{opt}}}(s)}. \tag{3.8}$$

Furthermore, by replacing $E_{\gamma_{\text{opt}}}(s)$ with 3.2 and $M_1(s)$ in numerator of (3.8) by its definition, the expression of optimal controller becomes

$$C_{opt}(s) = \frac{\gamma_{\text{opt}} M_d(s) n E_{\gamma_{\text{opt}}}(s)\hat{G}_{\gamma_{\text{opt}}}(s) N_o^{-1}}{dW_1(s)^2 \gamma_{\text{opt}}^2 \left(K_{\gamma_{\text{opt}}}(s) + \gamma_{\text{opt}} M_n(s) M_1(s)\hat{G}_{\gamma_{\text{opt}}}(s)\right)}, \tag{3.9}$$

where $nE_{\gamma_{\text{opt}}}$ is the numerator of $E_{\gamma_{\text{opt}}}$. For the sake of simplicity, let us define another function, $R_o(s)$, which is a finite dimensional bi-proper transfer function (recall that $W_1$ is assumed to be bi-proper, so $deg(nW_1) = deg(dW_1)$, that makes $R_o$ bi-proper):

$$R_o(s) = \frac{nW_1(s) dW_1(s)}{nE_{\gamma_{\text{opt}}} M_d(s)}.$$

Then the equation for optimal controller can be re-written as

$$\begin{aligned} C_{opt}(s) &= \left(\frac{W_1(s)}{\gamma_{\text{opt}}^2}\right) \frac{\hat{G}_{\gamma_{\text{opt}}}(s) N_o^{-1}}{\left(\frac{nW_1(s)dW_1(s)}{M_d(s) nE_{\gamma_{\text{opt}}}(s)}\right)\left(\gamma_{\text{opt}}^{-1} K_{\gamma_{\text{opt}}}(s) + M_n(s) M_1(s)\hat{G}_{\gamma_{\text{opt}}}(s)\right)} \\ &= \left(\frac{W_1(s)}{\gamma_{\text{opt}}^2}\right) \frac{\hat{G}_{\gamma_{\text{opt}}}(s) N_o^{-1}}{R_o(s)\left(\gamma_{\text{opt}}^{-1} K_{\gamma_{\text{opt}}}(s) + M_n(s) M_1(s)\hat{G}_{\gamma_{\text{opt}}}(s)\right)}. \end{aligned}$$

By recalling (3.6) and the definition of $L(s)$, the following interpolation conditions can be obtained:

$$1 + M_n(\beta_k) F(\beta_k) L(\beta_k) = 0, \quad k = 1, 2, \ldots, n_1 \tag{3.10}$$

for all zeros of $E_\gamma$. Additionally, by definition of $\psi$ vectors in $L(s)$ in 3.6

$$1 + M_n(\alpha_k) F(\alpha_k) L(\alpha_k) = 0, \quad k = 1, 2, \ldots, n \tag{3.11}$$

interpolation conditions are also satisfied. Therefore, poles of $R_o(s)$ are canceled by the zeros of $(\gamma_{\text{opt}}^{-1} K_{\gamma_{\text{opt}}}(s) + M_n(s) M_1(s)\hat{G}_{\gamma_{\text{opt}}}(s))$ at the same locations.

Now, let us define

$$d_\infty = \gamma^{-1} R_o(\infty) K_{\gamma_{\text{opt}}}(\infty),$$

where both $R_o(s)$ and $K_{\gamma_{\text{opt}}}(s)$ are bi-proper transfer functions, which means $d_\infty$ is a constant and $d_\infty \neq 0$. By multiplying both numerator and denominator with this constant, the terms in the denominator are scaled down from high magnitudes, which will be useful for the approximation of those terms to obtain a suboptimal controller with a similar performance level.

By adding and subtracting 1 to the denominator and using newly defined $d_\infty$ in optimal controller expression, (3.9) becomes

$$C_{opt}(s) = \frac{W_1(s)}{\gamma_{\text{opt}}^2 d_\infty} \left( \frac{\hat{G}_{\gamma_{\text{opt}}}(s) N_o^{-1}(s)}{1 + \left( d_\infty^{-1} R_o(s) \left( \gamma_{\text{opt}}^{-1} K_{\gamma_{\text{opt}}}(s) + M_n(s) M_1(s) \hat{G}_{\gamma_{\text{opt}}}(s) \right) - 1 \right)} \right) \tag{3.12}$$

The terms parenthesized in the denominator of (3.12) can be decomposed as

$$\left( d_\infty^{-1} R_o(s) \left( \gamma_{\text{opt}}^{-1} K_{\gamma_{\text{opt}}}(s) + M_n(s) M_1(s) \hat{G}_{\gamma_{\text{opt}}}(s) \right) - 1 \right) = H_n(s) + H_d(s), \tag{3.13}$$

where $H_n(s)$ and $H_d(s)$ are strictly proper transfer functions. Moreover, the poles of $H_n(s)$ are the poles of $K_{\gamma_{\text{opt}}}$ in $\mathbb{C}_+$, if it has no pole in $\mathbb{C}_+$, then $H_n(s) = 0$. On the other hand $H_d(s)$ is in $\mathcal{H}_\infty$, because terms other than $K_{\gamma_{\text{opt}}}(s)$ have all their poles in $\mathbb{C}_-$.

By using the decomposition above, the final form of the optimal controller can be written as

$$C_{opt}(s) = \frac{W_1(s)}{\gamma_{\text{opt}}^2 d_\infty} \left( \frac{C_o(s) \hat{G}_{\gamma_{\text{opt}}}(s) N_o^{-1}(s)}{1 + C_o(s) H_d(s)} \right), \tag{3.14}$$

where $C_o(s) = (1 + H_n(s))^{-1}$. Note that $W_1(s)$, $\hat{G}_{\gamma_{\text{opt}}}(s)$, and $C_o(s)$ are finite dimensional, $H_d$ and $N_o$ are infinite dimensional transfer functions. Therefore, to implement a finite dimensional suboptimal controller, $N_o(s)$ and $H_d(s)$ need to be approximated.

Since the expression of $C_{opt}$ given in (3.14) has two possibly infinite dimensional functions that are both in $\mathcal{H}_\infty$ and the rest are finite dimensional, approximation

of $C_{opt}$ can be done more efficiently compared to direct approximation of $C_{opt}$. The Matlab command `fitfrd` will be used to obtain approximations of those infinite dimensional terms automatically. The results are checked by using the suboptimal performance level, the error between infinite dimensional terms and corresponding approximations and the bound explained in the next section so to check whether automatically approximated functions are useful enough or not.

## 3.4 Bound on the Suboptimal Performance Level

When the plant has finitely many unstable poles and the weights are finite dimensional rational functions, the final form of the optimal controller shown in (3.14) can be expressed as

$$C_{opt} = \frac{W_1(s)C_0(s)(d_\infty \gamma_{\text{opt}}^2)^{-1} G_{\gamma_{\text{opt}}}(s)}{1 + C_0(s)H_d(s)} N_o(s)^{-1}, \tag{3.15}$$

where $d_\infty$, $\gamma_{\text{opt}}$ are real constants, $C_0(s)$, $G_{\gamma_{\text{opt}}}(s)$ are finite dimensional and $N_o(s)$, $H_d(s)$ are possibly infinite dimensional functions. The suboptimal controller obtained by approximating the optimal controller contains approximation of $H_d(s)$ and $N_o(s)$. Therefore, the approximated controller, $C_a(s)$, can be expressed as

$$C_a = \frac{W_1(s)C_0(s)(d_\infty \gamma_{\text{opt}}^2)^{-1} G_{\gamma_{\text{opt}}}(s)}{1 + C_0(s)H_{da}(s)} N_{oa}(s)^{-1}, \tag{3.16}$$

where $H_{da}(s)$ and $N_{oa}(s)$ are finite dimensional approximations of $H_d(s)$ and $N_o(s)$ respectively. Let us define

$$C_{1a}(s) = W_1(s)(d_\infty \gamma_{\text{opt}}^2)^{-1} G_{\gamma_{\text{opt}}}(s) N_{oa}(s)^{-1}$$

for the sake of simplicity.

When the approximated controller is used to form a negative feedback system, the sensitivity function becomes

$$S_a(s) = \frac{M_d(s)(1 + C_o(s)H_{da}(s))}{M_d(s)(1 + C_0(s)H_{da}(s)) + M_n(s)N_o(s)C_0(s)C_{1a}(s)}. \tag{3.17}$$

Recall that the sensitivity function of the system which uses the optimal controller is

$$S_{opt}(s) = \frac{M_d(s)(1 + C_o(s)H_d(s))}{M_d(s)(1 + C_0(s)H_d(s)) + M_n(s)N_o(s)C_0(s)C_1(s)}. \tag{3.18}$$

By doing proper algebraic manipulations, $S_a(s)$ can be represented as follows using $S_{opt}$:

$$S_a(s) = \frac{S_{opt}(s) + \Delta_1(s)}{1 + \Delta_1(s) + \Delta_2(s)}, \tag{3.19}$$

where

$$\begin{aligned} \Delta_1(s) &= S_{opt}(s)\frac{C_0(s)(H_{da}(s) - H_d(s))}{1 + C_0(s)H_d(s)} \\ \Delta_2(s) &= T_{opt}(s)(N_{oa}^{-1}(s)N_o(s) - 1). \end{aligned}$$

Since $T_a(s) = 1 - S_a(s)$, the closed loop system function with the approximated controller can be expressed as

$$T_a(s) = \frac{T_{opt}(s) + \Delta_2(s)}{1 + \Delta_1(s) + \Delta_2(s)}. \tag{3.20}$$

The feedback system $(C_a, P)$ is stable, i.e., $S_a \in \mathcal{H}_\infty$ if

$$\delta := (\delta_1 + \delta_2) < 1, \tag{3.21}$$

where

$$\delta_1 := \|\Delta_1\|_\infty = \left\| S_{opt}(s)\frac{C_0(s)(H_{da}(s) - H_d(s))}{1 + C_0(s)H_d(s)} \right\|_\infty \tag{3.22}$$

$$\delta_2 := \|\Delta_2\|_\infty = \left\| T_{opt}(s)(N_{oa}^{-1}(s)N_o(s) - 1) \right\|_\infty . \tag{3.23}$$

The equation (3.21) simply means that by using appropriate approximations $N_{oa}(s)$ and $H_{da}(s)$ such that $\delta < 1$, the performance level can be maintained close to optimal performance level. The performance level under the approximated controller $C_a(s)$ can be shown as

$$\begin{bmatrix} W_1(s)S_a(s) \\ W_2(s)T_a(s) \end{bmatrix} = \begin{bmatrix} \frac{C_0(s)H_{da}(s)}{1+C_0(s)H_d(s)} & 0 \\ 0 & N_{oa}^{-1}(s)N_o(s) \end{bmatrix} \begin{bmatrix} W_1(s)S_{opt}(s) \\ W_2(s)T_{opt}(s) \end{bmatrix} \frac{1}{1 + \Delta_1(s) + \Delta_2(s)} \tag{3.24}$$

Thus, the following result is derived:

$$\gamma_a := \left\| \begin{bmatrix} W_1(s)S_a(s) \\ W_2(s)T_a(s) \end{bmatrix} \right\|_\infty \leq \gamma_{\text{opt}} \frac{1+\epsilon}{1-\delta}, \tag{3.25}$$

where

$$1 + \epsilon := \max\{ \left\| \frac{C_0(s)H_{da}(s)}{1 + C_0(s)H_d(s)} \right\|_\infty, \ \left\| N_{oa}^{-1}(s)N_o(s) \right\|_\infty \}.$$

# Chapter 4

# Implementation of Toker-Özbay Formula by Using YALTA

Toker-Özbay formula was implemented by following the steps of [7] as a Matlab program named HINFCON in 1996 [15]. The program allows user to obtain optimal controller by entering $M_n(s)$, $M_d(s)$, and $N_o(s)$ and manually iterating $\gamma$ interval to find $\gamma_{opt}$ in a desired tolerance. Therefore, to find a single optimal controller, the user needs to find coprime factorization in the form of (3.1) and iterate the algorithm of calculation of $\gamma_{\text{opt}}$ manually. Moreover, since the optimal controller is generally infinite dimensional ($M_n(s)$ is infinite dimensional when system has time-delay and $N_o(s)$ may be infinite dimensional depending on $P(s)$), to find a finite dimensional suboptimal controller, manual approximation is needed. By converting these human interactions with the program into an autonomous way, the program can work much faster. In addition, the plant may have quasi-polynomials in its numerator and denominator, which leads to a non-trivial coprime factorization by hand.

This chapter explains the upgraded version of HINFCON, which significantly improves the calculation speed of both optimal controller with infinite and suboptimal controller with finite order. Also, the steps of obtaining controllers are shown by using pictures of the graphical user interface (GUI), which is designed

to make the program more user-friendly.

Firstly, YALTA (Yet Another LTI TDS Algorithm), a Matlab package used for gathering information about the given neutral/retarded plant, such as unstable poles, root locus with respect to delay of the system, effect of delay on number of unstable poles is introduced [10]. Since calculation of location of poles and zeros in the right half plane is enough to find $M_n(s)$ and $M_d(s)$, this package allows program to do the coprime factorization automatically for the plant. Then, the algorithm that iteratively shrinks the interval of $\gamma$ to find $\gamma_{\text{opt}}$ is explained. Finally, calculation of optimal controller according to updated formula and its approximation to a finite order suboptimal controller is described.

## 4.1 Coprime Factorization by Using YALTA

This section provides the background information about YALTA, explains how to use the functions implemented in the package and how the coprime factorization in the form (3.1) is done automatically.

Firsty, the plant function $P(s)$ needs to be in the form

$$P(s) = \frac{t(s^\alpha) + \sum_{\kappa=1}^{N'} t_\kappa(s^\alpha) e^{-\kappa s h_o}}{p(s^\alpha) + \sum_{k=1}^{N} q_k(s^\alpha) e^{-k s h_o}} = \frac{n(s)}{d(s)}$$

to be processed by using YALTA, where $h_o > 0$ is the nominal delay and ($t,\ p,\ q_k$ for all $k \in \mathbb{N}_N$, and $t_\kappa$ for all $\kappa \in \mathbb{N}_{N'}$) are real polynomials in $s^\alpha$ for $0 < \alpha \leq 1$ (fractional system if $\alpha < 1$, integer order system if $\alpha = 1$), which satisfy the conditions $\deg p \geq \deg t,\ \deg p \geq \deg t_\kappa,\ \deg p \geq \deg q_k$, where deg represents degree of $s^\alpha$. The inequality $\deg p \geq \deg q_k$ makes the package be able to process both neutral and retarded systems. Under assumptions that are given in detail in [16], YALTA can provide

- number and location of unstable poles according to given $h_o$,

- stability of the system with respect to $0 < h \leq h_o$,
- loci of the dominant roots of the system for $0 < h \leq h_o$,
- the coprime factors ($N$, $D$) of $P$ and their approximation ($N_n$, $D_n$) in $\mathcal{H}_\infty$-norm

for both retarded and neutral systems with asymptotic axes in $\{\Re(s) < 0\}$. Moreover, if the system is neutral, YALTA is able to find the position of asymptotic axes and return whether the location of asymptotic poles of the chain are on the left or right of the imaginary axis when the imaginary axis is an asymptotic axis.

Since the package is only used to obtain position of unstable poles, it shows stability window and root locus with respect to delay in this work, the explanation of this Matlab package is limited with these processes. Detailed information about the algorithm behind its implementation can be found in [10, 17, 18, 19, 20, 21, 22].

### 4.1.1 Main function of YALTA

The main function of the package, `delayFrequencyAnalysisMin.m`, launches the system analysis and returns information about the given polynomial as a `struct` variable. Furthermore, it returns root locus and stability window according to given delay. Note that the function processes a single quasi-polynomial rather than processing the whole system, therefore numerator and denominator of the plant need to be analyzed separately.

The fractional quasi-polynomials in the form

$$C(s^\alpha, \tau) = \sum_{k=0}^{N} q_k(s^\alpha) e^{-ks\tau}$$

can be input to the function by entering each variable separately.

`T = delayFrequencyAnalysisMin(`$q$`,`$k$`,`$\alpha$`,`$\tau$`,opt,`$\epsilon$`,`$t_{min}$`,`$t_{max}$`)`

First four inputs are to define the polynomial and the others are the optional inputs with defaults that determine how the root locus and stability window are displayed:

- $\boldsymbol{q}$ : coefficients of the polynomial $q_k(s^\alpha)$ are represented in the matrix form

$$q = \begin{bmatrix} q_{0,n} & q_{0,n-1} & \cdots & q_{0,0} \\ \cdots & & & \\ q_{N,n} & q_{N,n-1} & \cdots & q_{N,0} \end{bmatrix}$$

  where $q_k(s) = \sum_{l=0}^{n} q_{k,l}(s^\alpha)^l$. Note that zero polynomials must not be entered, except $q_0$. So, if there is any zero polynomial $q_k(s)$ for $0 \leq k \leq N$, that polynomial must be skipped in the matrix form.

- $\boldsymbol{k}$ : the vector whose elements represent the $k$ values for which the polynomials $q_k(s)$ are nonzero. This vector provides the function to be more user-friendly by removing the necessity of typing each $q$ matrix as $(N+1)\times(n+1)$ even if it has zero rows.

- $\boldsymbol{\alpha}$ : the power of $s$ for fractional polynomials of $s^\alpha$ where $\alpha \in (0\ 1]$.

- $\boldsymbol{\tau}$ : a positive real number that is nominal value of delay shown in (4.1.1).

- **opt** : a boolean option to trace and show root loci if `opt=1` and vice versa. Default value is 1.

- $\boldsymbol{\epsilon}$ : precision of integration procedure for computing root locus. The error of the resulting plot decreases as $\epsilon$ becomes smaller, but the computation time increases. Default value is $10^{-4}$.

- $\boldsymbol{t_{min}}$ : minimum delay to limit x-axis of the stability window ($t_{min} \geq 0$). Default value is 0.

- $\boldsymbol{t_{max}}$ : maximum delay to limit x-axis of the stability window ($t_{max} > t_{min}$). Default value is $\tau$.

The output is a single `struct` variable with 11 fields containing information about the given quasi-polynomial. For example, positions of roots on the right half plane are stored in an array in a field called `T.UnstablePoles`, information about the number of unstable poles is given in `T.AsympStability` etc. Since our software uses `T.UnstablePoles` and `T.AsympStability` fields only, the other fields of output are not described but their explanation can be found in [16].

*Remark:* Although YALTA and HINFCON can handle fractional order systems, definition of "stability" with respect to the root locations requires careful transformations in this case. Since we will be mainly concerned with time delay systems, we assume $\alpha = 1$ from now on. For a fractional order system example on how to implement the ideas used here see [23].

### 4.1.2 Computation of $M_n(s)$, $M_d(s)$ and $N_o(s)$

Since `T.UnstablePoles` field contains the roots of a quasi-polynomial function in the right half plane, only one of the fields is enough to obtain $M_n(s)$ and $M_d(s)$. The roots returned in this field by entering numerator and denominator quasi-polynomials are zeros of $M_n(s)$ and $M_d(s)$ respectively, and their poles are symmetric to zeros with respect to imaginary axis, because they are defined as inner functions. Lastly, $N_o(s)$ can be computed by using (3.1).

Note that this method only works when number of unstable zeros and poles of the system are both finite. Otherwise, the function returns an empty array for the field `UnstablePoles`. Therefore, to expand the range of usable systems in the software, this process also checks the number of roots of given quasi-polynomial in the left half plane when there are infinitely many roots in the right half plane. If the number of roots in the left half plane is finite in this case, the position of them can be obtained by replacing $\hat{s} = -s$ in $P(\hat{s}) = N(\hat{s})/D(\hat{s})$ (location of roots change symmetrically with respect to origin). Since the complex roots have conjugate pairs, the returned roots in `UnstablePoles` are positioned symmetrically with respect to imaginary axis with their original locations. Hence, the original locations of stable roots can be obtained by reconverting $\hat{s} = -s$.

Finally, the expression that is the numerator of $M_n(s)$ or $M_d(s)$ can be found by dividing the quasi-polynomial with the polynomial generated by using stable roots.

If numerator or denominator of plant has infinitely many roots in both left and right half planes, then the software is unable to do coprime factorization since YALTA cannot return expressions.

## 4.2 Iterative Algorithm for Calculation of $\gamma_{\text{opt}}$

Toker-Özbay formula states that $\gamma_{\text{opt}}$ is the largest $\gamma$ that makes $\mathcal{T}(\gamma)$ in (3.6) singular [7]. This statement is implemented by requesting an interval and number of samples to obtain linearly separated $\gamma$ values in [15]. However, it only returns a plot of minimum singular value of $\mathcal{T}(\gamma)$ with respect to $\gamma$. Furthermore, the largest $\gamma$ value which makes $\mathcal{T}(\gamma)$ may not be precise enough, so the user may want to change the interval to find a more precise value again and again, which consumes a great amount of time. The algorithm proposed in [9] solves these problems as explained below and provides faster process of calculation of optimal controller. On the other hand, $\gamma_{\text{opt}}$ may be outside of the initially given interval. In this case, both old and updated versions returns the plot of minimum singular with respect to $\gamma$ to let the user enter a new interval to be searched for.

Instead of requiring human interaction, basically the software iteratively shrinks the interval around $\gamma_{\text{opt}}$ to find a $\gamma$ that satisfies

$$|\gamma_{\text{opt}} - \gamma| < \epsilon,$$

where $\epsilon$ is the precision error entered by the user. Since $\gamma_{\text{opt}}$ is unknown, the software actually shrinks around the local minima of minimum singular value of $T(\gamma)$ that corresponds to largest $\gamma$. This operation is done by defining new boundaries for $\gamma$ depending on number of iterations processed around a single local minimum. However, because the local minimum that corresponds to largest $\gamma$ has a possibility of not converging to 0 as the interval shrinks, the software

needs to skip to previous local minima with respect to $\gamma$ and redo all the steps. This procedure is explained in detail by using an example.

Fig. 4.2 is obtained by using a retarded plant

$$P(s) = \frac{(s+3) + 2(s-1)e^{-h_n s}}{s^3 + s^2 - (s+1)e^{-h_d s}}$$

with parametric uncertainties $h_n \in [0.4,\ 0.44],\ h_d \in [0.19,\ 0.21]$. Weight functions are entered as

$$W_1(s) = \frac{0.001s + 1}{s},\ W_2(s) = \frac{0.13s\,(1 + 1.3(s/1.6) + (s/1.6)^2)\,(1 + s/10)}{1 + s/0.3},$$

where $W_2(s)$ is determined by using the parametric uncertainties given above. Additionally, the interval is specified as $\gamma_{min} = 0.1,\ \gamma_{max} = 5$ by the user and the number of linearly separated samples is chosen as 1000.

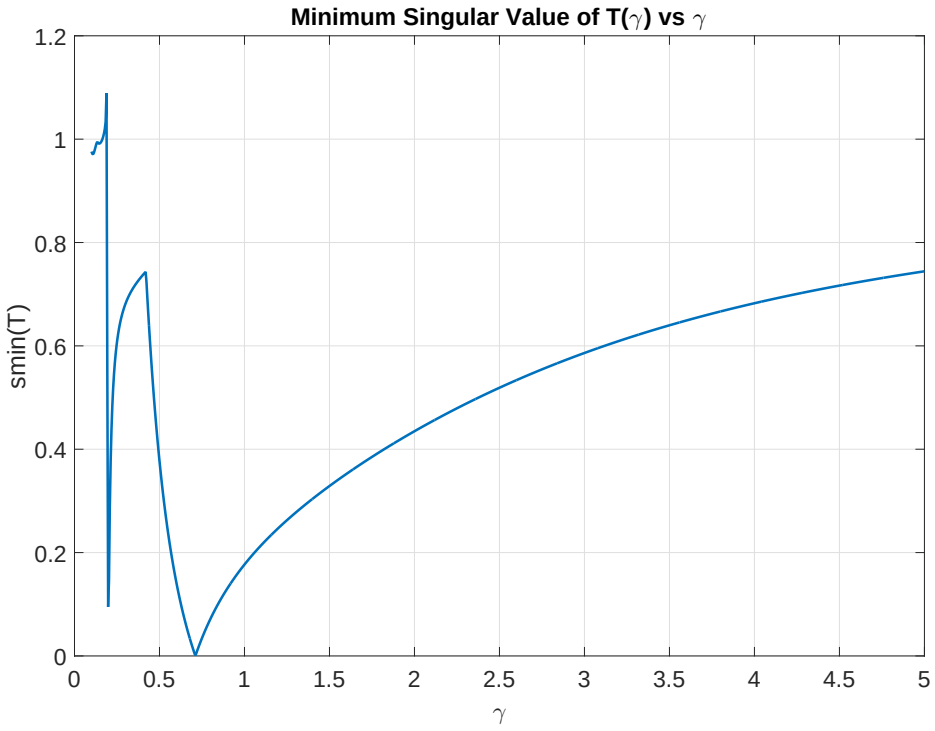


Figure 4.1: Minimum singular values of $T(\gamma)$ with respect to $\gamma$

Minimum singular values of T($\gamma$) with respect to $\gamma$ are shown according to sample number in given interval in Fig 4.2. The resulting plot shows that there are

two local minima in this interval and it is justifiable to claim that $\gamma_{opt} \in [0.5,\ 1]$. However, it is necessary to check whether that local minima converges to 0 or not, by using the shrinking process mentioned above. To do this process automatically, the software replaces x-axis of Fig. 4.2, $\gamma$, to $(\gamma_{max} - \gamma)$ and y-axis, minimum singular value of $T(\gamma)$, to negative of minimum singular value of $T(\gamma)$. Result of this procedure allows usage of `findpeaks` command in Matlab and process local maximums from left to right, and is given in Fig. 4.2.

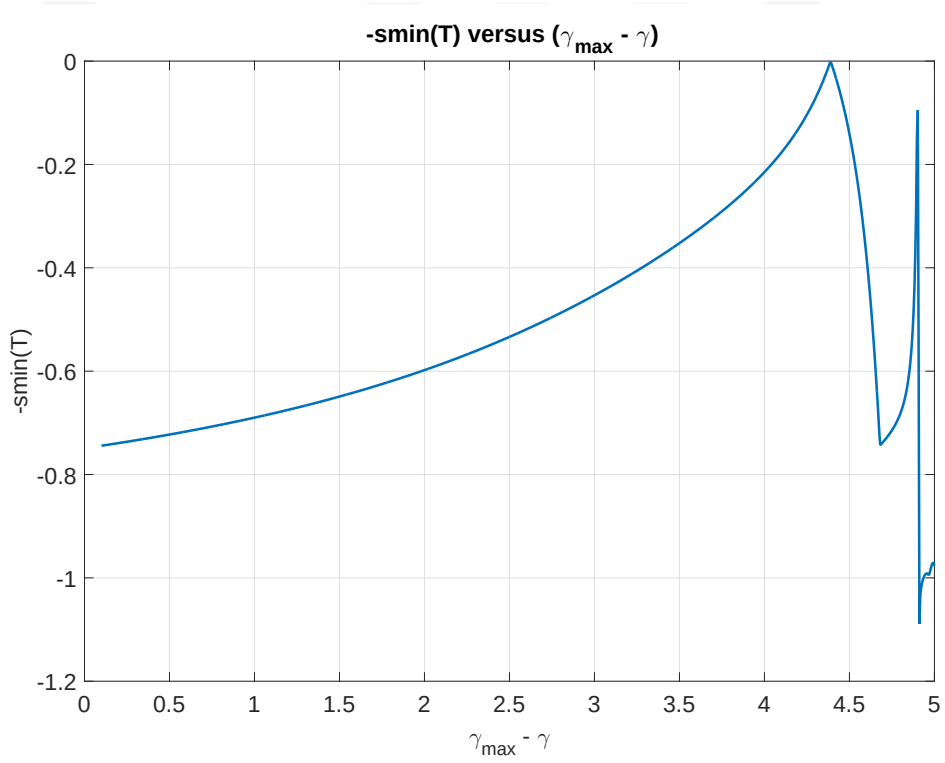


Figure 4.2: Negative of minimum singular values of $T(\gamma)$ with respect to $(\gamma_{max} - \gamma)$

The Matlab function called `findpeaks` is able to return local maxima and corresponding indices of given data. The software stores these indices and begins the shrinking process from first index until $\gamma_{opt}$ is found within the given precision error $\epsilon$. Let us represent the array that stores all $\gamma$ values that correspond to local maxima of `-min(svd(T))` as

$$\texttt{LM} = [\gamma_1\ \gamma_2\ \ldots\ \gamma_u],$$

where $u$ is the number of local maximums in the given interval. The program iteratively follows the steps provided below to find $\gamma_{opt}$. Note that `peak_index` is

the index of $LM$ and initialized as 0. Also, a coefficient is defined to determine interval for the shrinking process, called `interval_coef`.

**Step 1:** Increment `peak_index`. Set iteration number to 1.

**Step 2:** Initialize `interval_coef` to $0.5 \times$ `LM(peak_index)` $-$ `LM(peak_index+1)` or 0.1 if there is no local maxima for smaller $\gamma$. Shrink the interval by repeating steps 3-6.

**Step 3:** To shrink the interval, the coefficient that determines boundaries is recursively decreased: `interval_coef` $= 0.1 \times$ `interval_coef`
$\gamma_{min} =$ `LM(peak_index)` $-$ `interval_coef`
$\gamma_{max} =$ `LM(peak_index)` $+$ `interval_coef`

**Step 4:** Use `max` function to find maximum of the given interval and corresponding $\gamma$, $\gamma_{est}$.

**Step 5:** Check if `min(svd(T))` (minimum singular value of $T(\gamma_{est})$) converges to 0 by comparing current value with the one obtained in previous step. In the implementation, convergence is tested by using

$$\texttt{min(svd(}T(\gamma_{est,k})\texttt{))} < 0.8 \times \texttt{min(svd(}T(\gamma_{est,(k-1)})\texttt{))}$$

inequality, where $k$ represents number of iterations applied for the selected `peak_index`.

**Step 6:** If the iteration fails the test, repeat steps 1-6. If it passes, since $\epsilon$ determines the precision error, return $\gamma_{est}$ as $\gamma_{opt}$ if $\epsilon >$ `min(svd(`$T(\gamma_{est})$`))` and repeat steps 3-6 otherwise.

## 4.3 Optimal Controller and Its Approximation by a Finite Dimensional Controller

The optimal controller is calculated by directly following the revised form of Toker-Özbay formula described in Section 3.3 by using symbolic variables. Using

a built-in function of Matlab called `fitfrd`, infinite dimensional optimal controller is approximated by applying it on both $N_o(s)$ and $H_d(s)$ if necessary, separately.

The function `fitfrd` uses two/four inputs: frequency response data, order of approximation and optionally relative order of approximation and weight of optimization fit criteria. Frequency response data is calculated by using symbolic variables $H_d(s)$ and $N_o(s)$ in the bandwidth entered by the user. A panel is introduced for the user to enter order of approximation for both $H_d(s)$ and $N_o(s)$ with a default of 7 and the relative order of each function optionally as shown in Fig. 4.3. Approximated results are then used to obtain finite dimensional suboptimal controller $C_a(s)$.

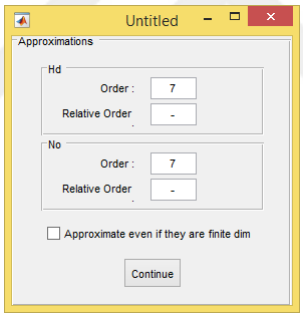


Figure 4.3: Selection of orders for approximation in GUI

In all the examples shown in chapter 5, the optimal controller is approximated by using the new structure by entering orders and relative orders of approximation. The choice of the order and relative degree of the approximation can be critical in the sense that the approximation obtained for $H_d$ and $N_o$ must be stable (to give a small $L_\infty$ error, the software sometimes may result in an unstable approximation, which is not acceptable); also relative degree of the approximation of $N_o$ must be equal or lower than the relative degree of the original $N_o$, plus this function must be outer (sometimes the approximation obtained may have zeros in the right half plane, that is not acceptable).

## 4.4 Graphical User Interface

A GUI is implemented in Matlab to provide a user-friendly environment. After the user enters initial inputs, the GUI performs the steps explained above in the background: coprime factorization, estimation of optimal performance level for the mixed sensitivity problem and optimal controller calculation. The terms defined in steps of Toker-Özbay formula and newly defined terms for the decomposition are shown on the GUI panel. Each step provides related figures such as root loci with respect to delay, plot of $\gamma$ vs minimum singular value of $T(\gamma)$, Bode diagrams of optimal and suboptimal controllers etc. Furthermore, before the final step which calculates the optimal controller, the user is asked to enter orders for approximation of $H_d(s)$ and $N_o(s)$. Demonstration of steps of GUI is given in the following figures by using the same example solved in [7]: Find optimal performance level and controller for the plant $P(s)$, given $W_1(s)$ and $W_2(s)$

$$P(s) = \frac{e^{-0.2s}}{s-1}, \quad W_1(s) = \frac{2(s+1)}{10(s+0.1)}, \quad W_2(s) = 0.2(s+1.1).$$

The numerator and denominator of the plant is entered in $s$ domain. If any of them is quasi-polynomial, then the term with the lowest order exponential delay is written first, and the rest is entered with increasing order, separated by commas. For example, if the denominator is $((s^2-1)+(s+4)e^{-0.25s}+(s+1)e^{-0.5s})$, the user needs to type `s^2-1, s+4, s+1` in the denominator field. Moreover, the field labeled as nominal tau for denominator should be `0.25`. Since this thesis focuses on plants with fractional power of 1, those fields are set to 1 by default, however YALTA is able to do coprime factorization of fractional systems.

The weights $W_1(s)$ and $W_2(s)$ are also typed in s-domain. In this example, the frequency range which will be used for approximations and Bode, Nyquist plots are defined as $\omega \in [10^{-3},\ 10^{3}]$, where this range is logarithmically separated into 1000 samples by default. The values of $\gamma_{min}$ and $\gamma_{max}$ are entered by the user to be used for $\gamma_{opt}$ estimation in second step. This interval is separated into linearly spaced 1000 points by default. Low number of samples may cause to a misleading result for $\gamma_{opt}$, but high number of samples may lead to unnecessary time consumption during the calculation of $\gamma_{opt}$. Finally, a threshold is chosen

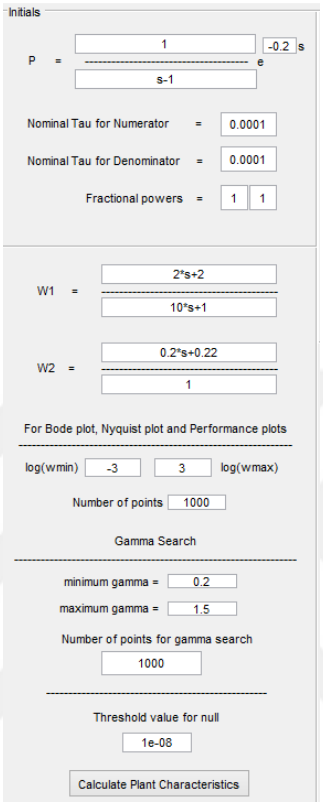


Figure 4.4: Initial parameters and function definitions to be entered by the user

for the estimation of $\gamma_{opt}$, which sets a threshold for the minimum singular value of $T(\gamma)$.

As the user clicks on *Calculate Plant Characteristics*, main function of YALTA is called to do coprime factorization. This package returns unstable roots of given quasi-polynomials, which allows the software to evaluate $M_n(s)$ and $M_d(s)$, hence $N_o(s)$ too (see Fig. 4.5). The function also returns stability and root loci of the system with respect to delay as shown in Figs. 4.6, 4.7.

When the user clicks on *Continue*, the software starts to shrink the interval given by the user by using the iterative algorithm explained in Section 4.2 and finds the optimal performance level according to given threshold. The result of each iteration is displayed in the command window (minimum singular value of best sample in the interval and renewed interval boundaries $\gamma_{min}$ and $\gamma_{max}$). Finally, when $\gamma_{opt}$ is found, the GUI prompts $\gamma_{opt}$ value up to 10 most significant

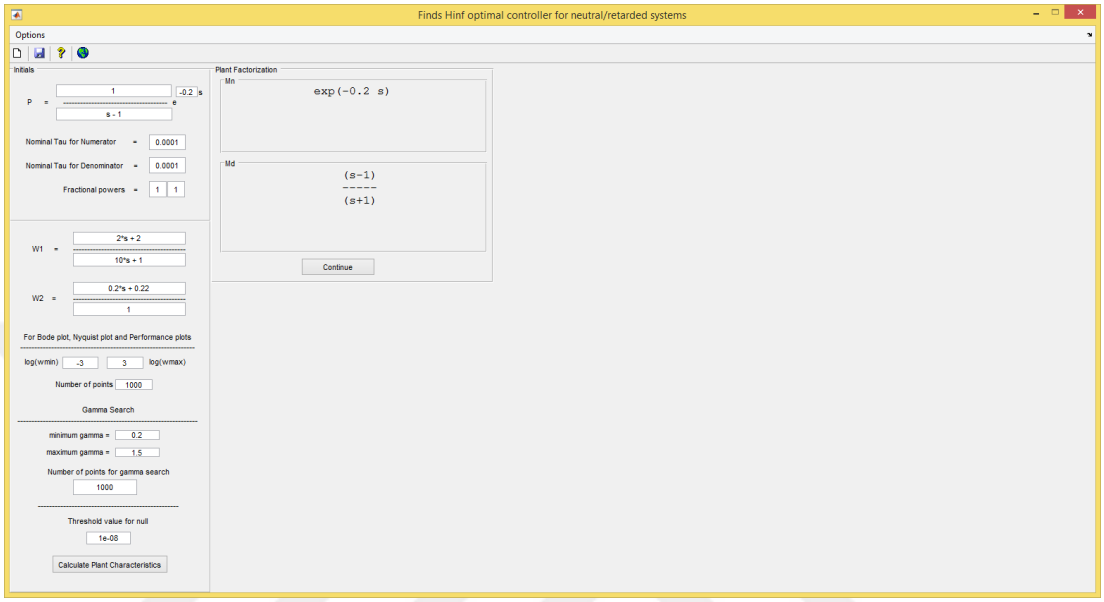


Figure 4.5: Coprime factorization visualization in GUI

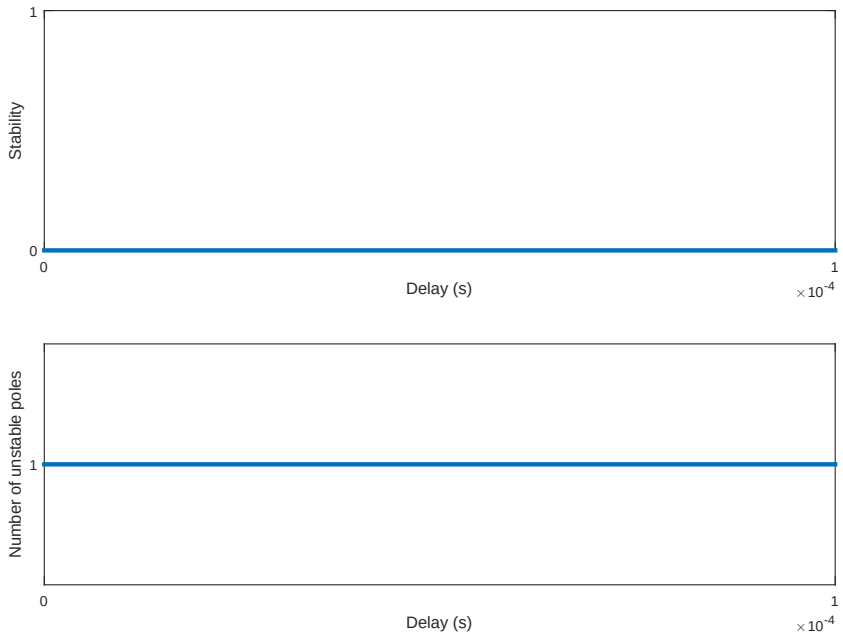


Figure 4.6: Stability analysis with respect to delayed term in denominator

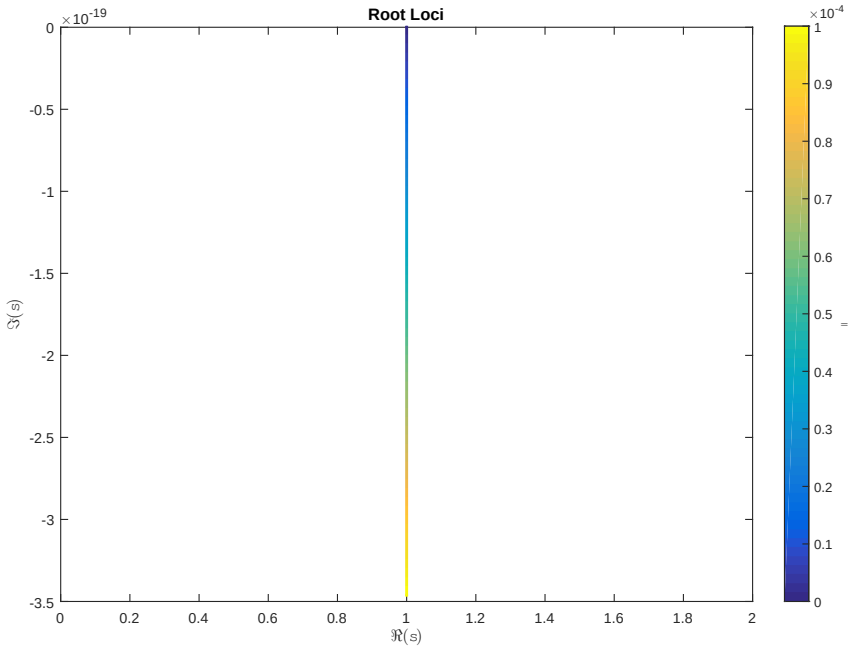


Figure 4.7: Root loci with respect to delayed term in denominator

digits and displays the plot which shows minimum singular value of $T(\gamma)$ with respect to 1000 samples in the interval defined by the user, as shown in Fig. 4.8.

Pressing on *Estimate Optimal Controller* button opens another window, which allows the user to enter orders and relative orders for approximations of possibly infinite dimensional terms, $H_d(s)$ and $N_o(s)$ as shown in Fig. 4.3.

Finally, the software finds the optimal controller by using the estimated optimal performance level, the inputs given entered in the GUI and improved formulation of optimal controller, derived by using Toker-Özbay formula. The terms defined in Toker-Özbay formula are shown in GUI as the optimal controller and the approximation-based estimated suboptimal controller are obtained. Furthermore, Nyquist plot with both controllers, Bode plots of both controllers, Bode diagram of $H_d(s)$ versus its approximation and the performance levels of both systems are displayed in different figures.

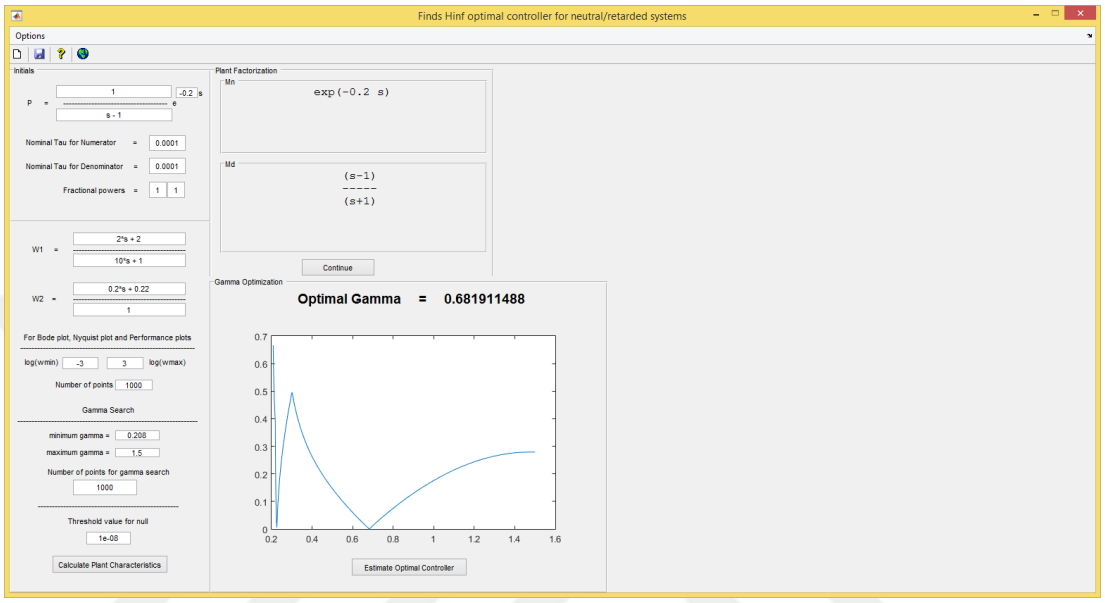


Figure 4.8: Visualization of estimated $\gamma_{opt}$ in GUI

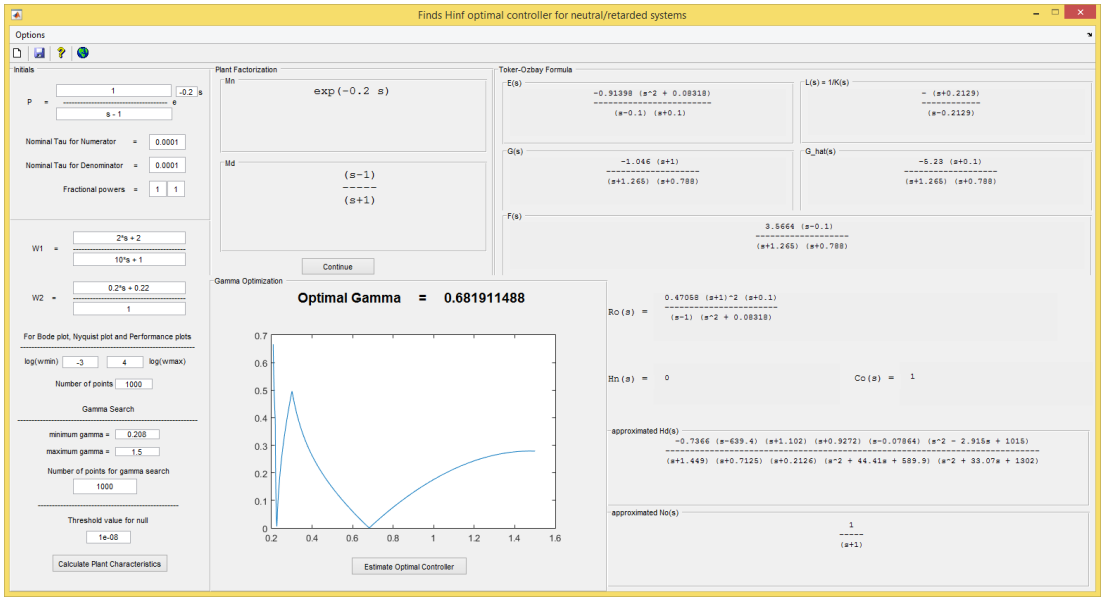


Figure 4.9: Visualization of the terms defined in Toker-Özbay formula

# Chapter 5

# Examples

In this chapter, various time-delay systems are taken into consideration. In the first example, a plant with a varying parameter is defined and two different controllers are designed to be used as switching controller which switches depending on the value of varying parameter. The second example is about computation of a suboptimal controller for a plant which contains time delays in its dynamics, in addition to I/O time delay. Comparison of direct approximation of the optimal controller and approximation by using decomposition in equation (3.14) with respect to order of approximation is given in the third example. In the fourth example, for a plant with time-delay and a single unstable pole, the change of dynamics of the optimal and suboptimal controller is examined with respect to values of nominal time-delay and unstable pole and their parametric uncertainties. Finally, for a time delay system which contains quasi-polynomials in both of its numerator and denominator, the performance levels of controllers, which are designed by using the Matlab command `mixsyn` on Pade approximation of exponential terms in plant and our software explained in Chapter 4, are compared with respect to the orders of controllers.

## 5.1 Switching Suboptimal Controller Design for Retarded Delay System

A time-delay system is given as

$$P(s) = \frac{(s^2 + 6s + 4) + (7s + 1)e^{-0.1s}}{(s^3 + 3s^2 + 9s + 4) - 9e^{-0.25s} + (3s^2 + 2s + 4)e^{-0.5s}} \times e^{-hs}$$

where $h$ is a time-varying parameter and $h \in [0,\ 0.6]$. Also, the sensitivity weight is given as

$$W_1(s) = \frac{(\epsilon s + 1)^2}{(s + \epsilon)^2}$$

where $\epsilon = 10^{-3}$. We need to design a robust controller with high performance which satisfies $\gamma_{opt} < 0.9$ for the mixed sensitivity minimization problem, given in (2.1). Then, the optimal controller needs to be approximated to be implemented and used in simulation with a ramp reference signal. Recall that the characteristics of $W_1$ implies that we need to track ramp-like reference signals, since $W_1$ contains double pole near $s = 0$.

By using YALTA, the coprime factorization of the plant results with

$$M_n(s) = e^{-h_o s}, \quad M_d(s) = \frac{s - 0.0855}{s + 0.0855}, \quad N_o(s) = \frac{P_o(s)M_d(s)}{M_n(s)},$$

where $P_o(s)$ is the nominal plant and $h_o$ is the nominal time delay.

Since the relative order of the plant is 1, to design a strictly proper controller, the multiplicative uncertainty weight must be improper with a relative degree of $-2$, or lower. Therefore, for this example, we will bound multiplicative errors with the least order weight that has relative degree of $-2$, which means that the simplest choice for $W_2$ has two zeros and no poles.

To design of a single controller, for plants with uncertain values of time delays the multiplicative uncertainty weight may be chosen as $W_2(s) = 0.3s(1 + s/20)$ for the nominal time-delay $h_o = 0.3$ seconds. Fig. 5.1 shows that this weight bounds all the multiplicative errors (M. E.) for $h \in [0,\ 0.6]$.

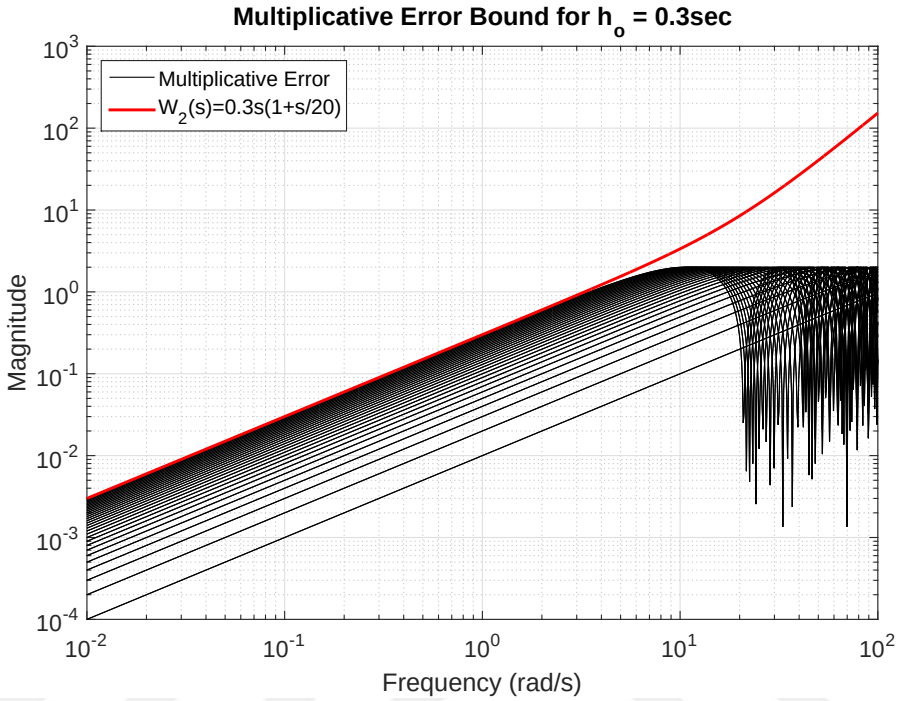


Figure 5.1: Multiplicative error bound for $h_o = 0.3$ and $h \in [0,\ 0.6]$

By using the software for this case which uses a single controller, the optimal performance level is obtained as $\gamma_{\text{opt}} \approx 0.9466$, which does not satisfy the criterion given in the problem ($\gamma_{\text{opt}} < 0.9$). Therefore, we will design two different controllers ($C_1,\ C_2$) to be used as switching controller. The first controller $C_1$ will be designed for $h_o = 0.2$ seconds and $h \in [0,\ 0.4]$, where $C_2$ will be designed for $h_o = 0.4$ seconds and $h \in [0.2,\ 0.6]$. The controller is planned to switch from $C_1$ to $C_2$ when $C_1$ is the used controller and $h \geq 0.4$ seconds, and switch from $C_2$ to $C_1$ when $C_2$ is the used controller and $h \leq 0.3$ seconds. For both cases, the multiplicative uncertainty weight may be chosen as $W_2(s) = 0.2s(1 + s/20)$ as shown in Figs. 5.2, 5.3.

Optimal performance levels for systems controlled by $C_1$ and $C_2$ are computed by using the software, and obtained as $\gamma_{opt_1} \approx 0.6669$ and $\gamma_{opt_2} \approx 0.8531$ respectively. Thus, the criterion given in the problem is satisfied with both optimal controllers.

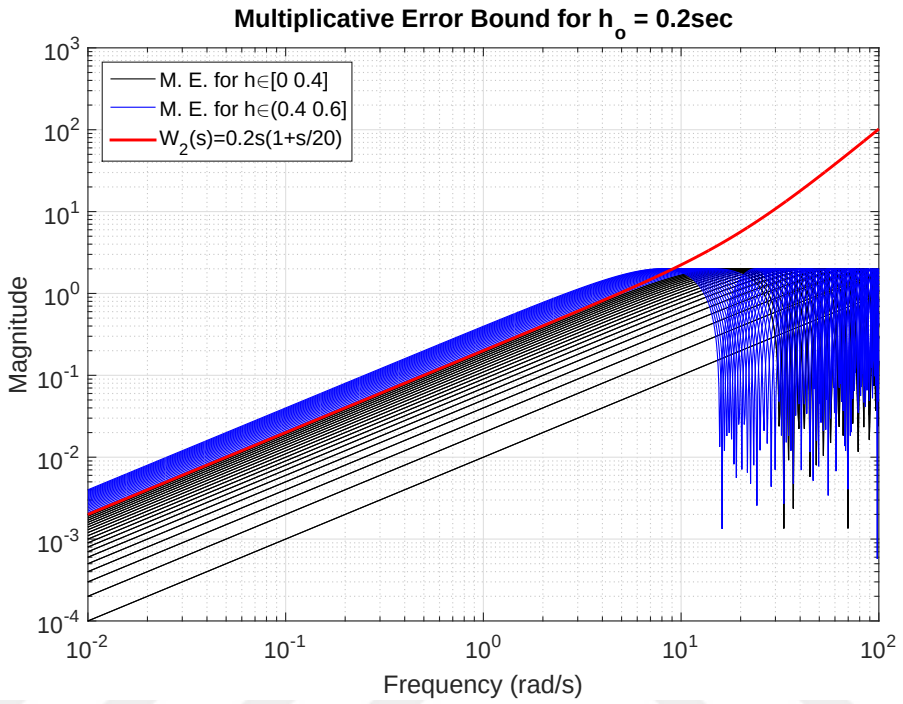


Figure 5.2: Multiplicative error bound for $h_o = 0.2$ and $h \in [0,\ 0.4]$

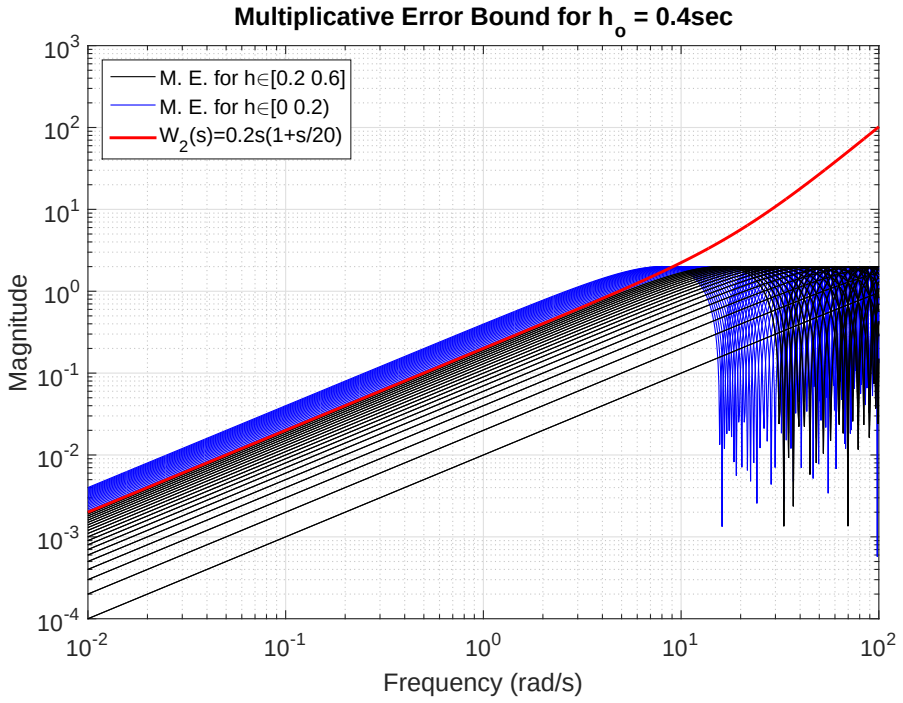


Figure 5.3: Multiplicative error bound for $h_o = 0.4$ and $h \in [0.2,\ 0.6]$

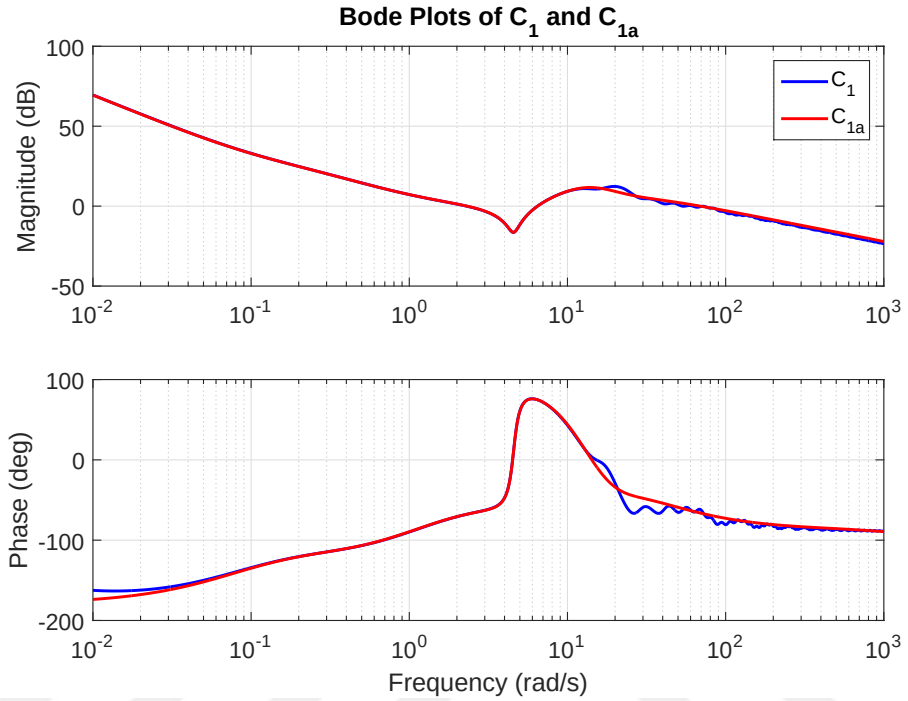


Figure 5.4: Bode diagrams of $C_1$ and $C_{1a}$

By using $\gamma_{opt_1}$, the optimal controller $C_1$ and its approximation $C_{1a}$ are obtained as shown in Fig. 5.4. Note that the phase shift in low frequency is caused by the approximation of poles at $p_{1,2} = -0.001 \pm 3 \times 10^{-5} j$ to $p_{1a} = p_{2a} = 0$. Orders of approximation for both $N_o$ and $H_d$ are chosen as 7, and the relative order of $N_o$ is specified as 1, since the relative order of plant is also 1. Thus, after cancellation of pole/zero, the order of the approximated suboptimal controller $C_{1a}$ becomes 12 as shown below.

$$C_{1a} = \frac{0.295(1+s/1.68)(1+s/0.77)(1+s/0.086)(1+0.14(s/4.57)+(s/4.57)^2)}{s^2(1+s/20)(1+s/0.39)(1+1.909(s/12.12)+(s/12.12)^2)} \times \frac{(1+1.4(s/11.89)+(s/11.89)^2)(1+1.094(s/22.81)+(s/22.81)^2)}{(1+1.192(s/12.53)+(s/12.53)^2)(1+1.14(s/15.21)+(s/15.21)^2)}$$

The performance plots of both controllers are given in Fig. 5.5 (approximation of $p_{1,2}$ to $p_{1a}$ and $p_{2a}$ are neglected on this figure).

By using $\gamma_{opt_2}$, the optimal controller $C_2$ and its approximation $C_{2a}$ are obtained as shown in Fig. 5.6. Note that the phase shift in low frequency is caused

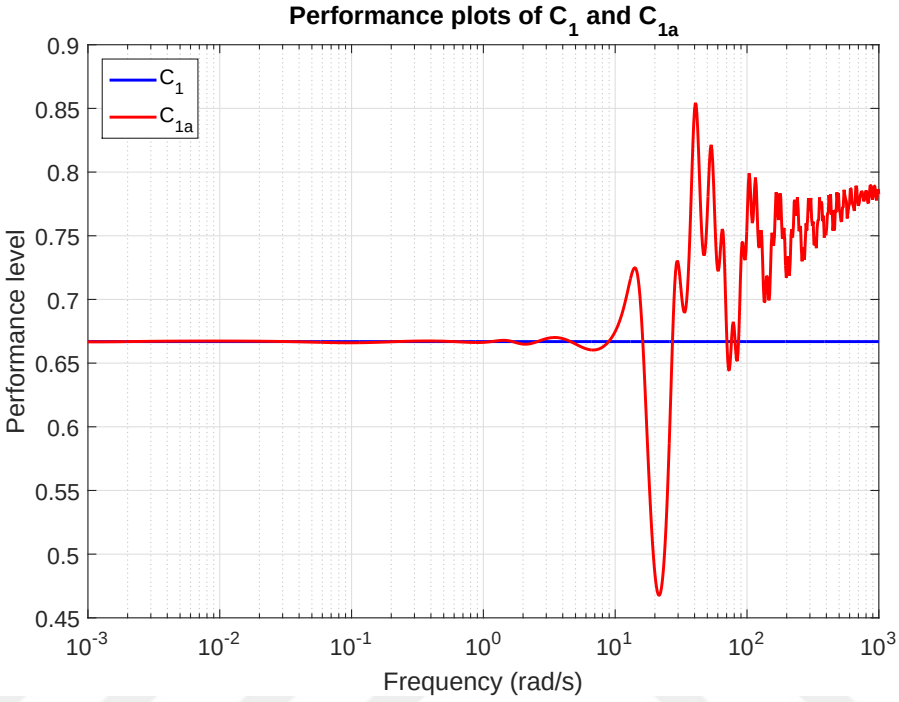


Figure 5.5: Performance plots of $C_1$ and $C_{1a}$

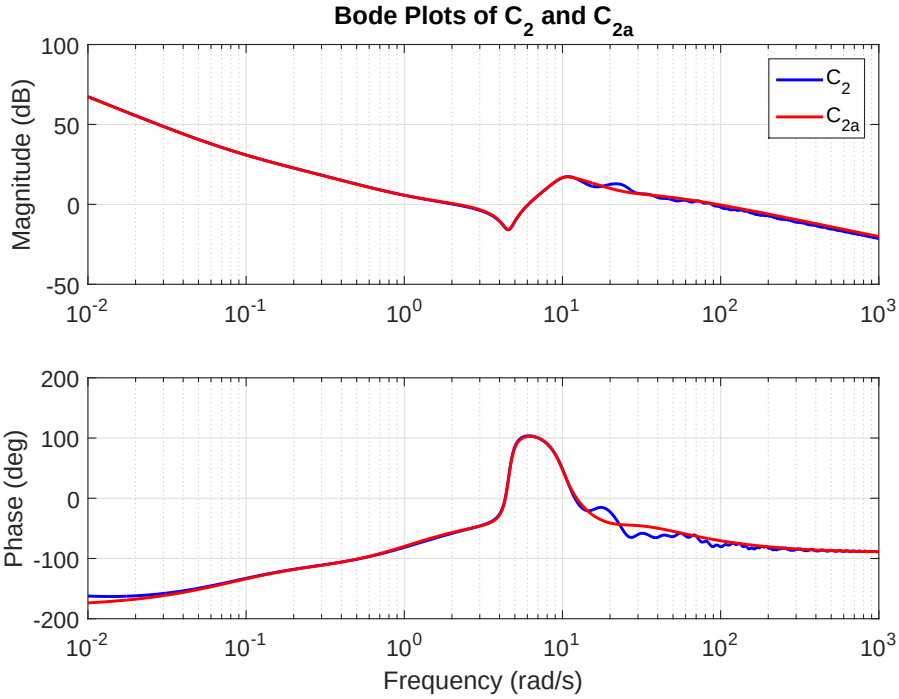


Figure 5.6: Bode diagrams of $C_2$ and $C_{2a}$

by the approximation of poles at $p_{1,2} = -0.001 \pm 3 \times 10^{-5} j$ to $p_{1a} = p_{2a} = 0$. Orders of approximation of $H_d$ and $N_o$ are chosen as the same, 7. Thus, after cancellation of same poles and zeros, the order of the approximated suboptimal controller $C_{2a}$ becomes 13 as shown below.

$$\begin{aligned} C_{2a} &= \frac{0.231(1+s/11)(1+s/1.26)(1+s/0.80)(1+s/0.086)}{s^2(1+s/21.63)(1+s/20)(1+s/0.39)} \\ &\quad \times \frac{(1+0.14(s/4.57)+(s/4.57)^2)(1+1.051(s/9.351)+(s/9.351)^2)}{(1+0.424(s/10.02)+(s/10.02)^2)(1+1.909(s/12.12)+(s/12.12)^2)} \\ &\quad \times \frac{(1+1.094(s/22.81)+(s/22.81)^2)}{(1+1.192(s/12.53)+(s/12.53)^2)} \end{aligned}$$

By comparing the roots of approximated controllers, we observe that the main differences between the controllers are the shift of zeros at low frequencies and the resonance at $\omega = 10.02$ rad/s which occurs in $C_{2a}$. Since the nominal time delay at the second case is two times greater than the nominal delay in first case, the resonance with low damping ratio in $C_1$ at $\omega \approx 20$ rad/s, which is neglected during approximation due to its relatively high frequency, is shifted to $\omega = 10.02$ rad/s which is almost the half of the resonance frequency in $C_1$. Because that the approximation error is weighted by $W_1(s)$ in the software, and $W_1(s)$ is a ramp-like function, dynamics of the system at low frequencies are approximated better compared to those at high frequencies. Therefore, the resonance at the second case is approximated while the one at $C_1$ is neglected during approximation.

The performance plots of both controllers are given in Fig. 5.7 (approximation of $p_{1,2}$ to $p_{1a}$ and $p_{2a}$ are neglected on this figure).

These controllers are then implemented on Simulink by applying the switching algorithm explained above. The block diagram of the system is shown in Fig. 5.8. The time varying parameter $h$ is chosen as a periodic signal

$$h(t) = 0.3 + 0.3\sin(\omega t)$$

where three different frequencies are selected to understand the chattering effect on output: $\omega = \{0.15,\ 1,\ 2\}$ rad/s. To illustrate the responses of the system to different frequencies, different reference signals are used. For the first case, the reference signal ($r(t)$) is chosen as $r(t) = 5\mathtt{r}(t) - 5\mathtt{r}(t-4)$ where $\mathtt{r}(t)$ notates

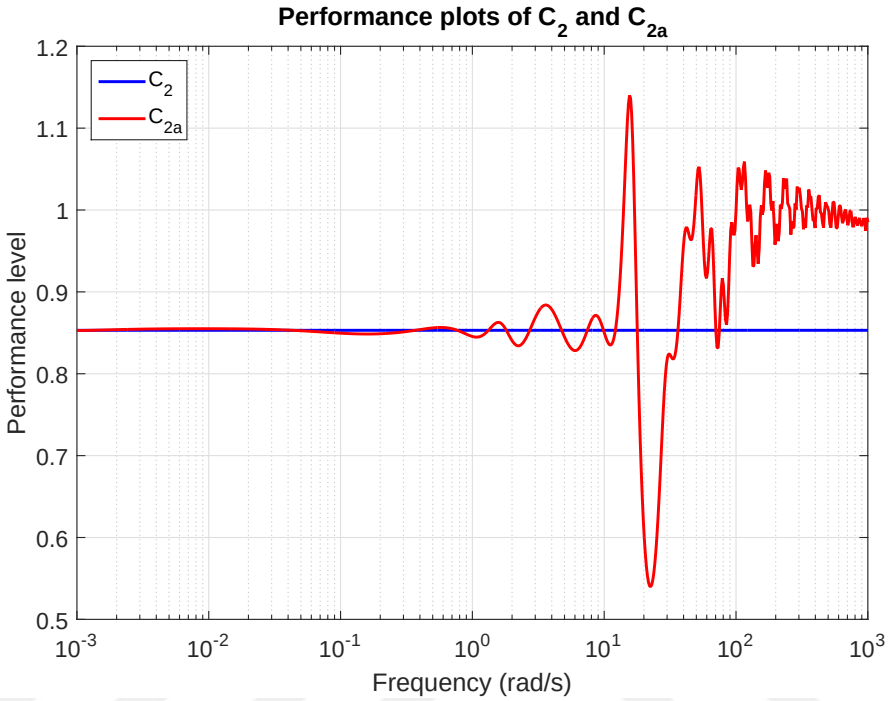


Figure 5.7: Performance plots of $C_2$ and $C_{2a}$

the ramp signal and $\omega$ is selected as 0.15 rad/s. This reference signal is a ramp which saturates at the value $r(4) = 20$. The time varying time delay $h(t)$ and the switching bit $(u_o)$ are shown in Fig. 5.9; $u_o$ specifies whether $C_1$ or $C_2$ is being used: when $u_o = 1$, $C_1$ is used and $C_2$ is the controller of the system when $u_o = 0$. In the second case, $r(t) = 5\mathtt{r}(t) - 5\mathtt{r}(t-7)$ and $\omega = 1$ rad/s. Lastly, for the case where $\omega = 2$ rad/s, the reference signal is specified as $r(t) = 5\mathtt{r}(t) - 5\mathtt{r}(t-10)$.

Fig. 5.10 shows the system responses for all three cases stated above. The output signals for all cases show that the system is robustly stable and settling time for each case is small when the complexity of the system is taken into consideration. The result shows that for low $\omega$, the steady state error converges to zero before each switching. As the controllers switch, a distortion with short duration is generated. As $\omega$ increases, both the duration and magnitude of distortions increase. Therefore, the steady state error increases monotonically as time delay varies more frequently.

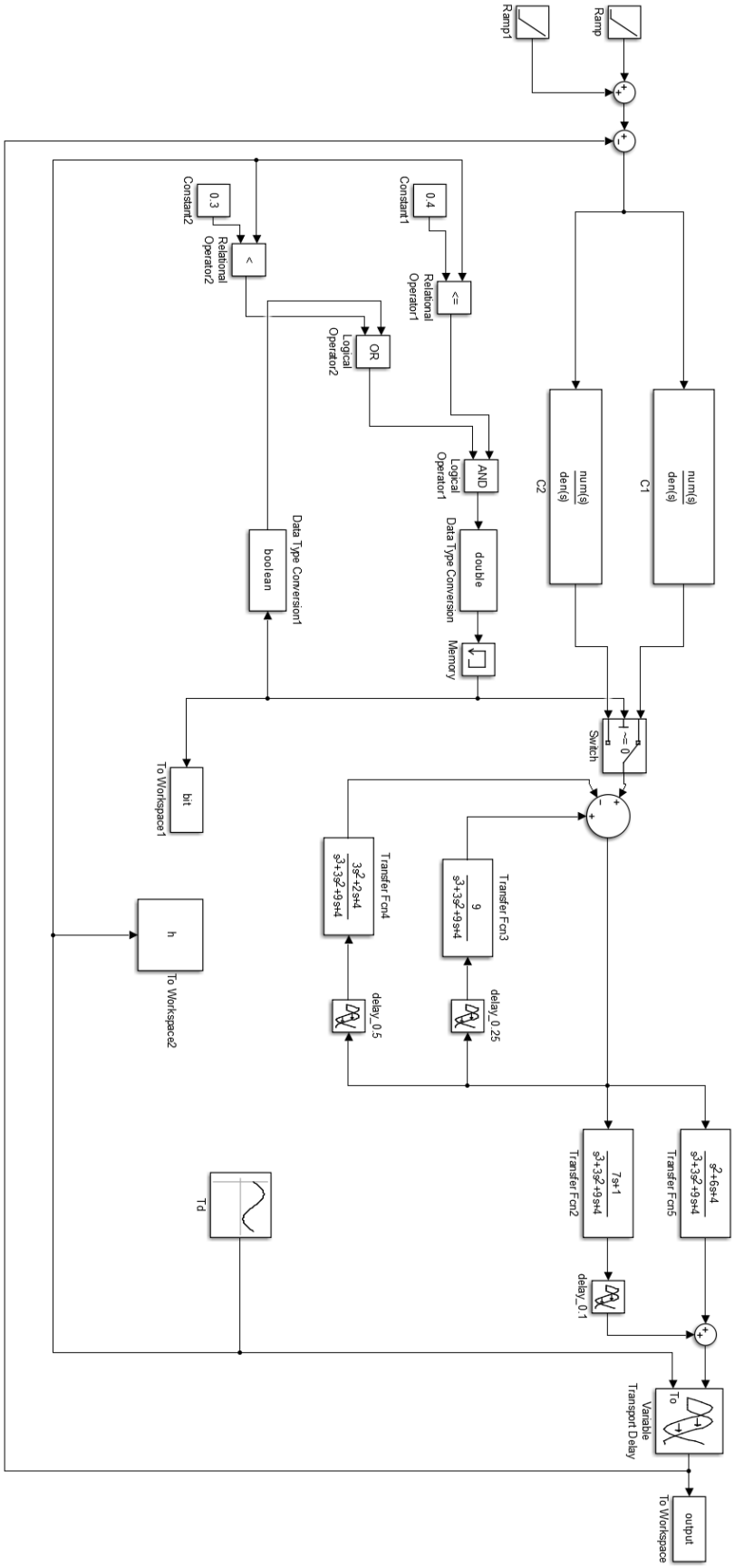


Figure 5.8: Simulink block diagram of the closed loop system

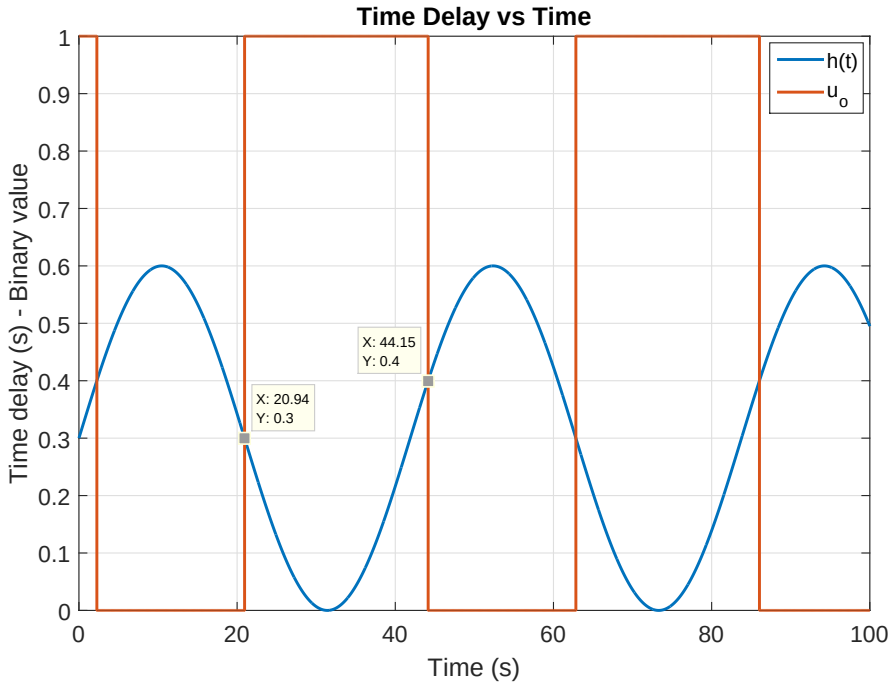


Figure 5.9: Time varying time delay $h(t) = 0.3 + 0.3\sin(0.15t)$ versus time and the switcher bit ($C_1$ is used when $u_o = 1$ and $C_2$ is used otherwise)

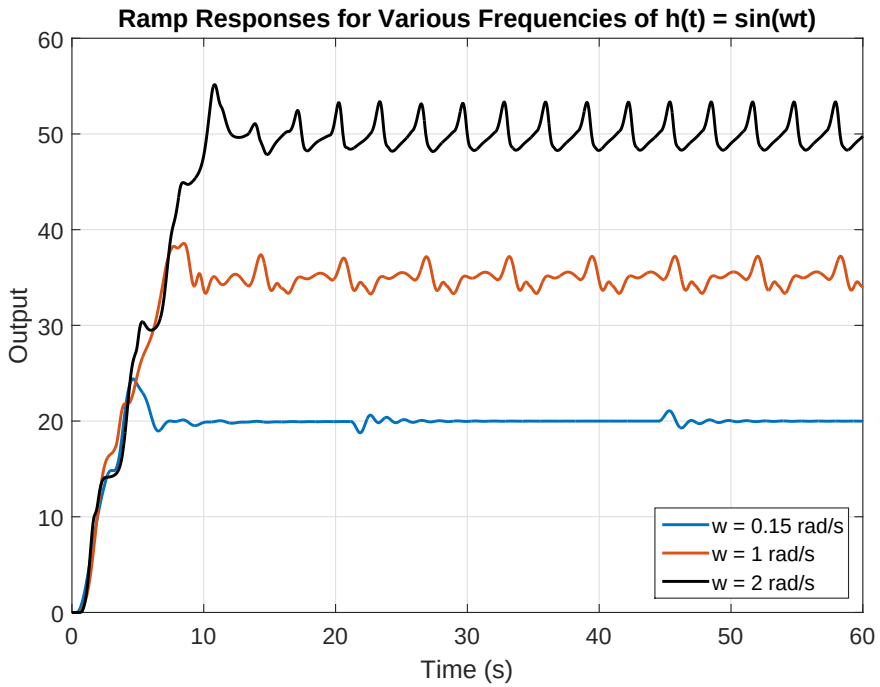


Figure 5.10: Responses of the system for various $\omega$ of $h(t) = \sin(\omega t)$

## 5.2 Suboptimal Controller Design for Neutral Delay System

Let the nominal neutral time delay system and the sensitivity and uncertainty weights are given as

$$P_o(s) = \frac{e^{-0.25s}}{(s+1) - 0.5(s-8)e^{-0.5s}},\ W_1(s) = \frac{1}{s},\ W_2(s) = 2s^2.$$

By running the software, coprime factorization is done with the help of YALTA and it is found that the system has 2 unstable poles:

$$M_d(s) = \frac{1 - 0.2275(s/3.188) + (s/3.188)^2}{1 + 0.2275(s/3.188) + (s/3.188)^2},\ M_n(s) = e^{-0.25s}.$$

To obtain the optimal performance level, $\gamma_{min} = 0.1$ and $\gamma_{max} = 100$ are entered. The iterative algorithm returned $\gamma_{\text{opt}} = 23.7788$, which is the largest $\gamma$ value that makes $T_\gamma(s)$ in (3.6) singular as shown in Fig. 5.11.

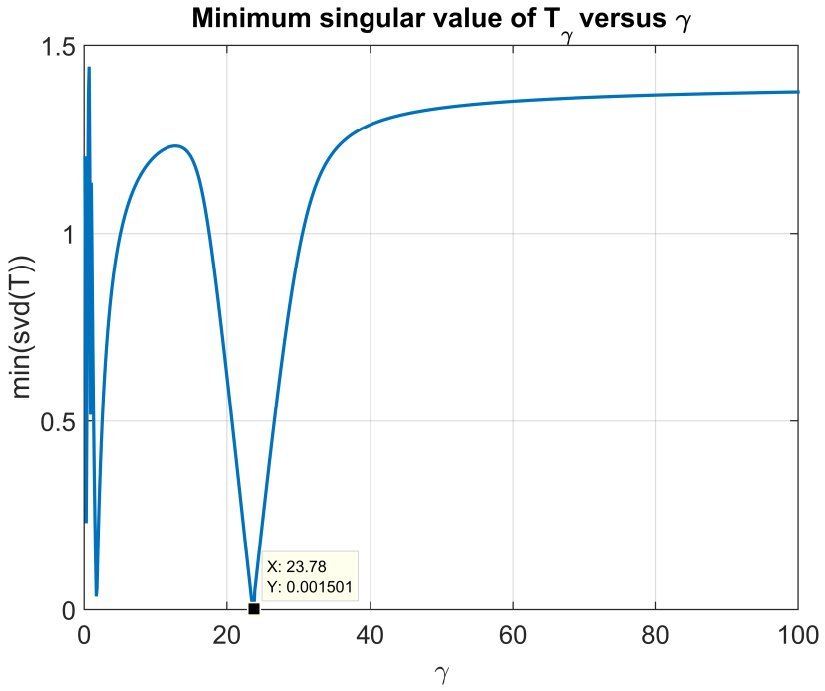


Figure 5.11: Minimum singular value of $T_\gamma$ with respect to $\gamma$ for $\gamma \in [0.1\ 20]$

By using $\gamma_{\text{opt}}$, weights and coprime factorization of the plant, the coefficients of $L(s)$ is computed from singular value decomposition of $T(\gamma_{\text{opt}})$ as

$$L(s) = \frac{(s - 0.1208)(s + 0.01739)}{(s + 0.1208)(s - 0.01739)} \ .$$

The optimal controller and its approximation are obtained by specifying orders of approximation for $H_d(s)$ and $N_o(s)$ as 7. Furthermore, the relative degree of $H_{da}$ is entered as 1, and since the plant is a strictly proper neutral system with a relative degree of 1, the relative degree of $N_{oa}$ is also entered as 1. Bode diagrams of infinite dimensional functions and their approximations are given in Figs. 5.12, 5.13 for $H_d(s)$ and $N_o(s)$ respectively.

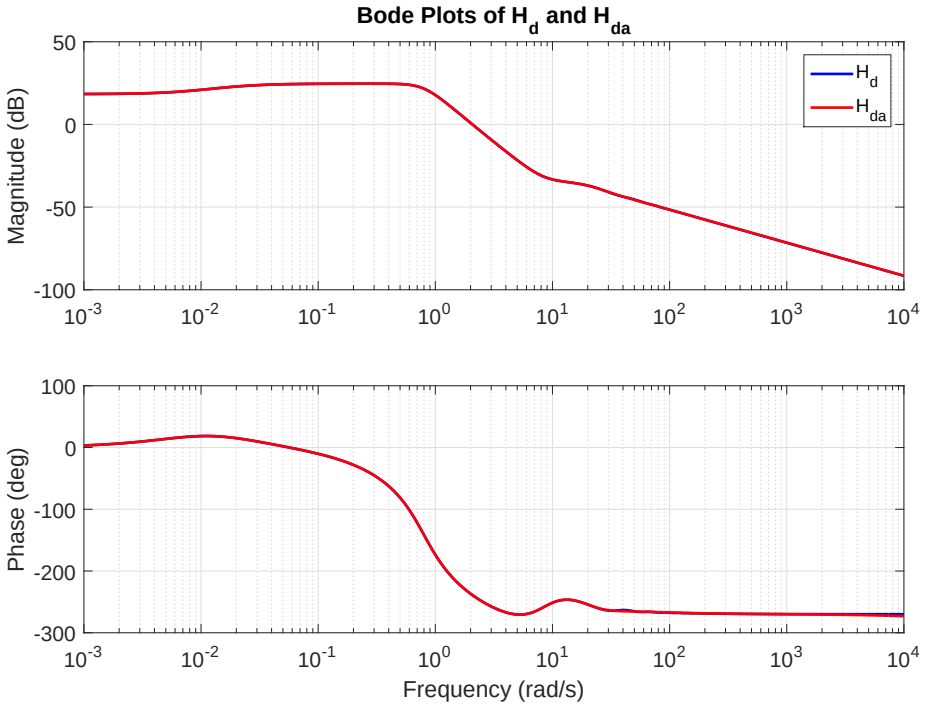


Figure 5.12: Bode plots of $H_d(s)$ and its 7 order approximation obtained by using `fitfrd`

$$\begin{aligned} H_{da}(s) \ &= \ \frac{8.2769(1 - s/5.129)(1 + s/0.00842)(1 + 1.06(s/8.48) + (s/8.48)^2)}{(1 + s/15.76)(1 + s/0.8)(1 + s/0.0174)(1 + (s/0.793) + (s/0.793)^2)} \\ &\quad \times \frac{(1 + 1.302(s/25.19) + (s/25.19)^2)}{(1 + 1.145(s/21.42) + (s/21.42)^2)} \end{aligned}$$

$$N_{o1a}(s) = \frac{0.2(1+1.89(s/12.42)+(s/12.42)^2)(1+1.22(s/15.87)+(s/15.87)^2)}{(1+0.228(s/3.188)+(s/3.188)^2)(1+0.151(s/13.76)+(s/13.76)^2)} \times \frac{(1+0.5101(s/23.49)+(s/23.49)^2)}{(1+s/158.7)(1+0.1198(s/25.85)+(s/25.85)^2)}$$

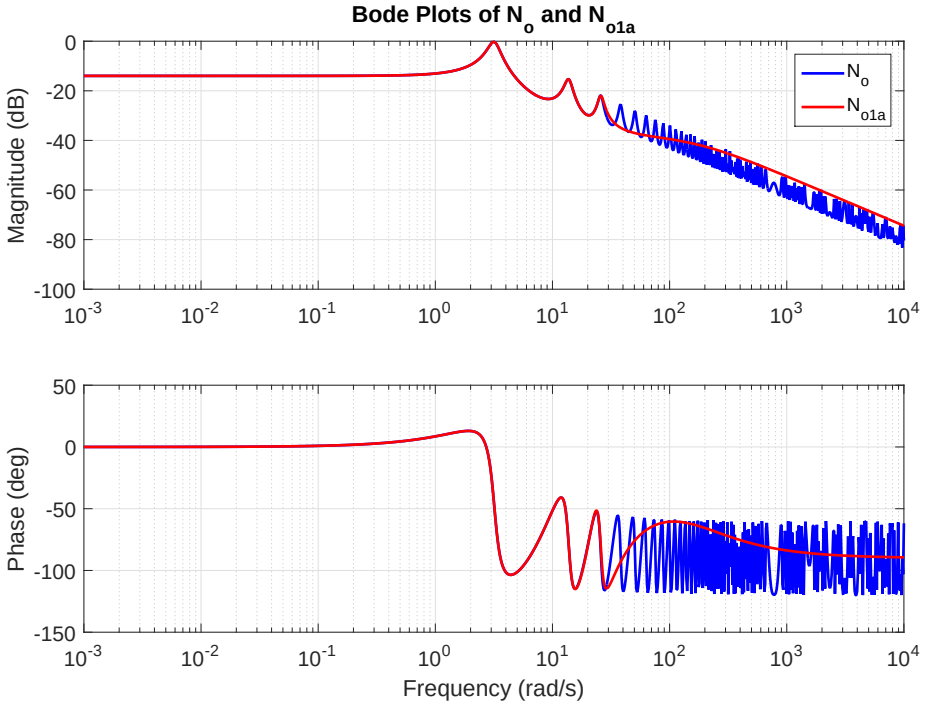


Figure 5.13: Bode plots of $N_o(s)$ and its 7 order approximation obtained by using `fitfrd`

Note that although the difference between $H_d$ and $H_{da}$ is small at low frequencies, $|N_{o1a}(j\omega)|$ is lower than $|N_o(j\omega)|$ at around $\omega \in [25\ 200]$ rad/s. Since the optimal controller is inversely proportional with $N_o$, the magnitude of approximated controller is larger than the optimal controller at these frequencies. Therefore, by using the transfer function $N_{o1a}$ a manual approximation is made and $7^{th}$ order $N_{o2a}$ is obtained by shifting the poles with highest natural frequencies to higher frequencies and increasing its damping ratio.

$$N_{o2a}(s) = \frac{0.2(1+1.9(s/12)+(s/12)^2)(1+1.215(s/15.6)+(s/15.6)^2)}{(1+0.2275(s/3.188)+(s/3.188)^2)(1+0.15(s/13.76)+(s/13.76)^2)} \times \frac{(1+0.6(s/22)+(s/22)^2)}{(1+0.1(s/27.5)+(s/27.5)^2)(1+s/110)}$$

The Bode diagrams of $N_o(s)$ and both of its approximations are shown in Fig. 5.14. By performing the steps introduced in section 3.4, the bounds on the suboptimal performance level by using $H_{da}$, $N_{o1a}$ and $H_{da}$, $N_{o2a}$ for the suboptimal controller are found as $\gamma_{1a} \leq 179.06$ and $\gamma_{2a} \leq 196.19$ respectively.

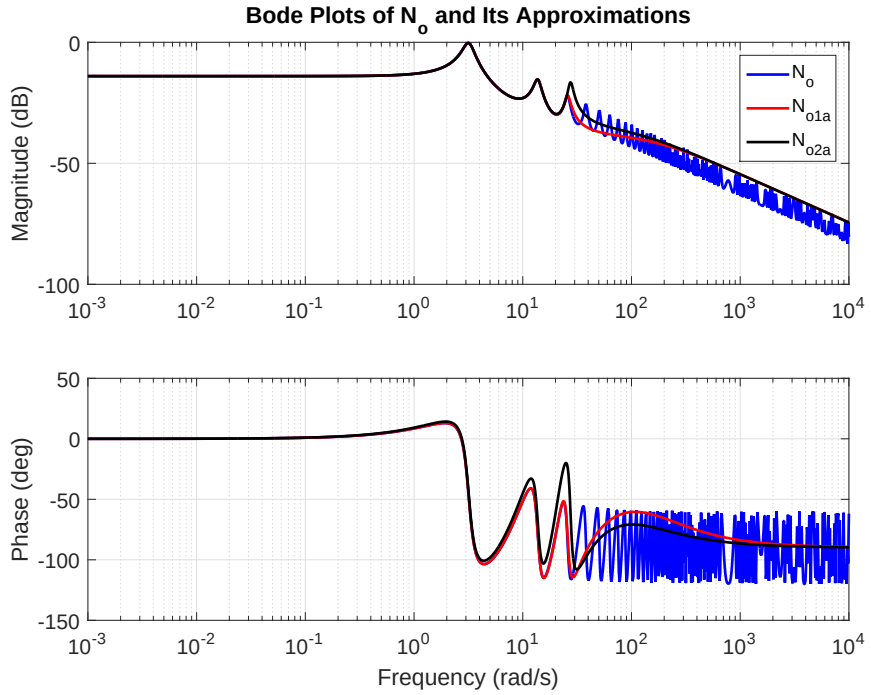


Figure 5.14: The Bode diagrams of $N_o(s)$ and both of its approximations

Let $C_1(s)$ be the suboptimal controller that is found by using $N_{o1a}$ and $C_2(s)$ be the one with $N_{o2a}$. Then, the bode plots of the optimal controller and $C_1$ and $C_2$ become as shown in Fig. 5.15.

The performance levels of optimal and suboptimal controllers with respect to frequency for $\omega \in [0.0110000]$ rad/s are given in Fig. 5.16. The plot shows that $\gamma_{1a} = 72.12$ and $\gamma_{2a} = 44.00$. The results are consistent with $\gamma_{1a} \leq 179.06$, $\gamma_{2a} \leq 196.19$. The upper bound of $\gamma_{2a}$ is larger than the bound of $\gamma_{2a}$, because the error between $N_o$ and $N_{o2a}$ is larger than the error between $N_o$ and $N_{o1a}$. However, since $W_1(s)$ is a step function and $W_2(s)$ is a $2^{nd}$ order function with a relative degree of $-2$, the magnitude of $W_2(j\omega)T(j\omega)$ is almost equal to $\Gamma(j\omega)$

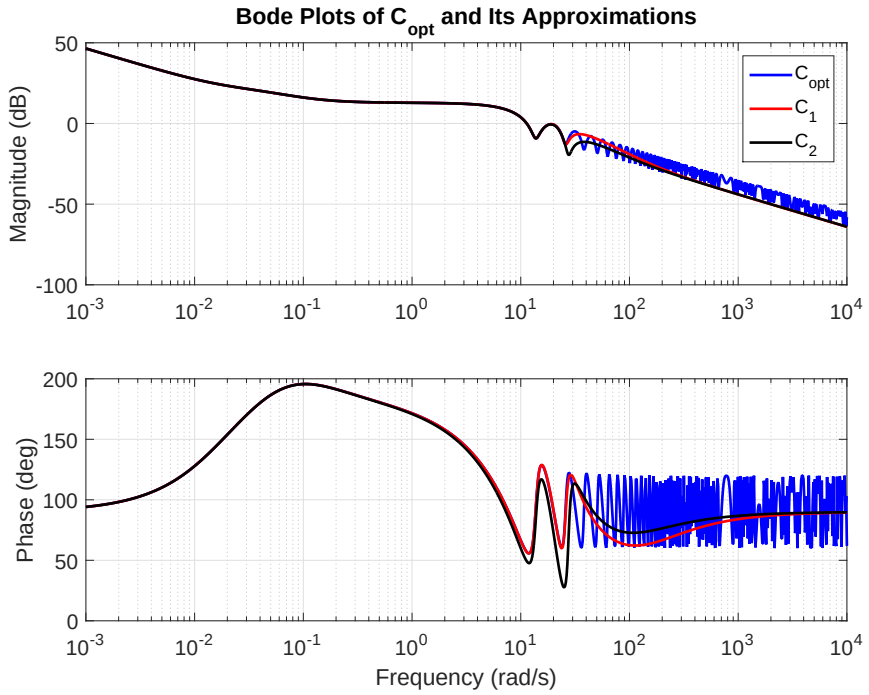


Figure 5.15: The Bode diagrams of $C_{opt}(s)$ and both of its approximations

where

$$\Gamma(j\omega) = \left\| \begin{bmatrix} W_1(j\omega)(1 + P(j\omega)C(j\omega))^{-1} \\ W_2(j\omega)P(j\omega)C(j\omega)(1 + P(j\omega)C(j\omega))^{-1} \end{bmatrix} \right\| \tag{5.1}$$

at high frequencies; the performance level increases as magnitude of the approximated suboptimal controller becomes larger than the magnitude of optimal controller.

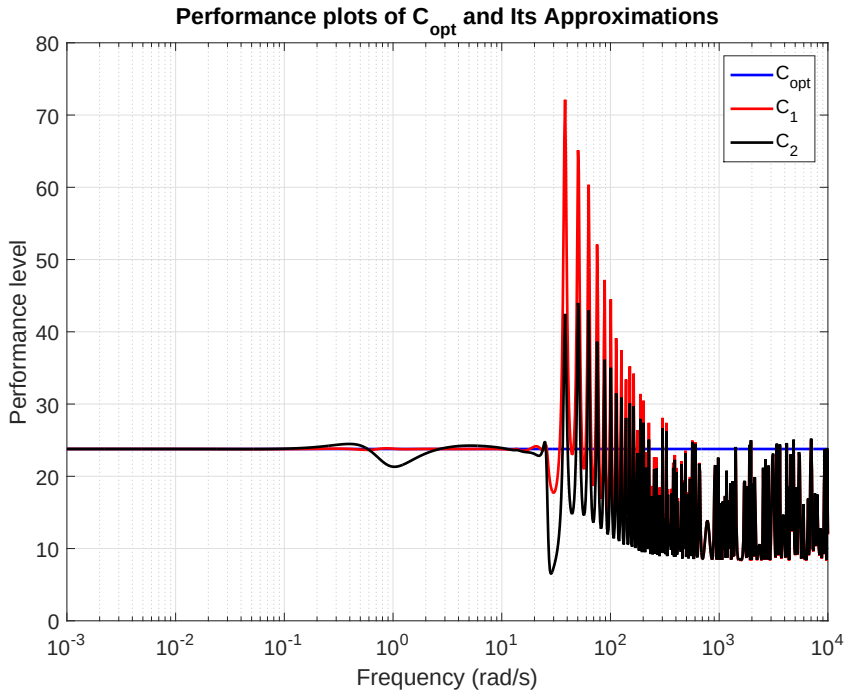


Figure 5.16: Performance levels of optimal and suboptimal controllers

## 5.3 Direct Approximation of Optimal Controller

To understand the utility of the new structure of the optimal controller given in (3.14), the example below compares the performance levels of direct approximation of optimal controller with the controller obtained by using approximations of $H_d$ and $N_o$.

Let the nominal system and weights be

$$P(s) = \frac{s-2}{(s^2-11s+440)+(s+10)e^{-\tau_o s}}\, e^{-s},\ W_1(s) = \frac{1}{s},\ W_2(s) = 0.5s(1+s/10)$$

where $W_1(s)$ is the sensitivity weight, $W_2(s)$ is the uncertainty weight and $\tau_o = 0.5$ seconds.

By using YALTA, coprime factorization of the plant is found and stability analysis of the system with respect to delay in denominator is obtained. In

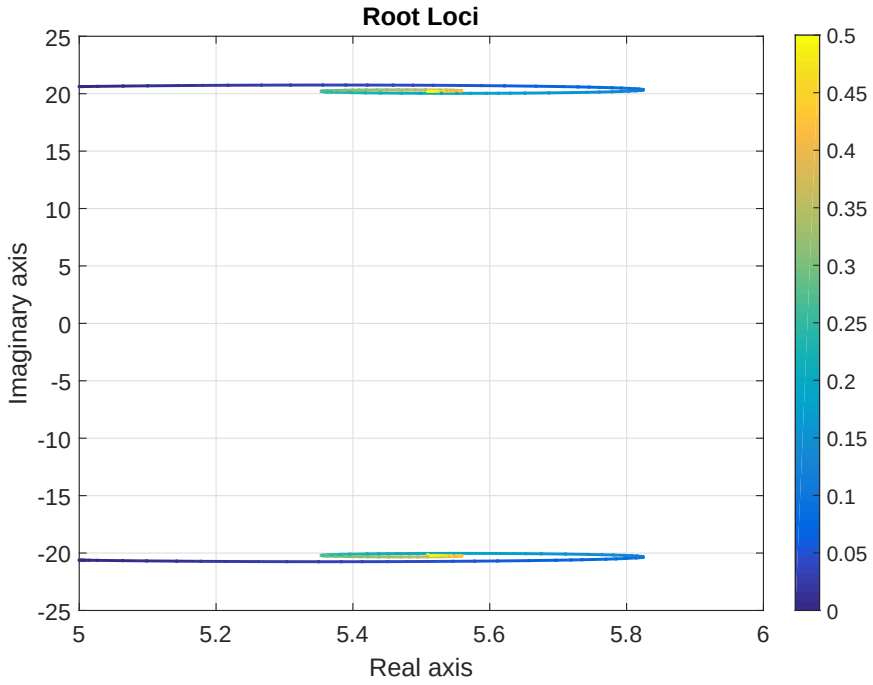


Figure 5.17: Performance levels of optimal and suboptimal controllers

Fig. 5.17, the root loci of unstable poles with respect to $\tau_o$ is plotted, which shows that when $\tau_o = 0.5$, the system has 2 unstable poles. .

As a result, $M_n(s)$ and $M_d(s)$ are found as

$$M_n(s) = \frac{s-2}{s+2}\, e^{-s},\ M_d(s) = \frac{(1 - 0.5262(s/20.9404) + (s/20.9404)^2)}{(1 + 0.5262(s/20.9404) + (s/20.9404)^2)}\ .$$

The optimal performance level $\gamma_{\text{opt}}$ is computed as $\gamma_{\text{opt}} = 1.9323$ as shown in Fig. 5.18 by entering [0.1 10] for the search interval. Then, steps of Toker-Özbay formula are performed by using the software and $K_{\gamma_{\text{opt}}}(s)$ is found as

$$K_{\gamma_{\text{opt}}}(s) = \frac{(1 - 0.5259(s/20.9428) + (s/20.9428)^2)}{(1 + 0.5259(s/20.9428) + (s/20.9428)^2)}\ .$$

*Remark:* Although $K_{\gamma_{\text{opt}}}$ and $M_d$ are very much alike, $K_{\gamma_{\text{opt}}}(s)$ satisfies the interpolation conditions given in (3.10,3.11). Note that the zeros of $E_\gamma$, $\beta_i$, are $\beta = \pm 0.5175j$ and zeros of $M_d$, $\alpha_i$ are $\alpha = 5.5095 \pm 20.2034j$. Then,

$$1 + \frac{M_n(x)F_\gamma(x)}{K_{\gamma_{\text{opt}}}(x)} = \{-4\times10^{-7}j,\ 4\times10^{-7}j,\ 4\times10^{-5} - 4\times10^{-6}j,\ 4\times10^{-5} + 4\times10^{-6}j\}$$

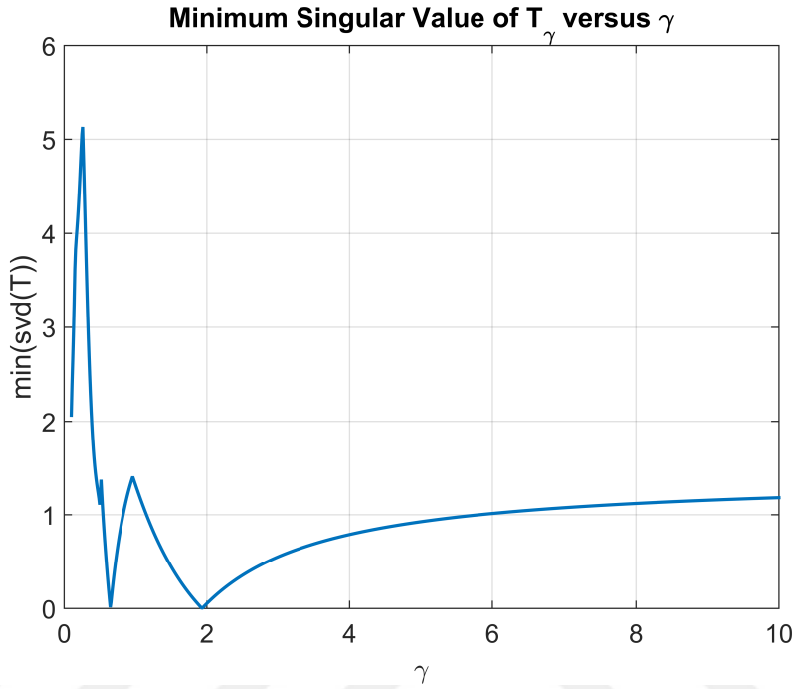


Figure 5.18: Minimum singular value of $T_\gamma$ with respect to $\gamma$ for $\gamma \in [0.1\ 10]$

$$1 + \frac{M_n(x)F_\gamma(x)}{M_d(x)} = \{-10^{-5}j, 10^{-5}j,\ 1.9 \times 10^{11} + 10^{12}j,\ 1.9 \times 10^{11} - 10^{12}j\}$$

where $x = \{\beta_1,\ \beta_2,\ \alpha_1,\ \alpha_2\}$. Therefore, the small difference between locations of poles and zeros of $K_{\gamma_{\text{opt}}}$ and $M_d$ ensures that the interpolation conditions are satisfied.

Since $K_{opt}$ is stable, $H_n(s) = 0$. The optimal controller found by using $\gamma_{\text{opt}}$ is then approximated directly and by using approximations of infinite dimensional transfer functions $H_d(s)$ and $N_o(s)$ with various orders for comparison. Direct approximation of $C_{opt}$ with order $N$ is represented by $C_{d,N}$ and the suboptimal controllers obtained by using the software with order $N$ are denoted by $C_{s,N}$.

For the approximation operations for both controllers, Matlab command `fitfrd` is used. To compute $C_{d,N}$, relative order of approximation is entered as 1 and the approximation is performed by using the frequency response data that corresponds to logarithmically separated 1000 samples for $\omega \in [10^{-3}\ 10^{3}]$. By using the same frequency range, $H_d(s)$ and $N_o(s)$ are approximated for the

estimation of suboptimal controllers $C_{s,N}$. To compare the performance levels obtained by using both controllers, the orders of the controllers are selected within the interval $N \in [9\ 23]$.

To obtain a suboptimal controller $C_{s,N}$ with $N$ order, for the interval $N = [10\ 15]$, order of $H_{da}$ is chosen as 5 and order of $N_{oa}$ is selected as $N-7$. On the other hand, when $N = [16\ 23]$, order of $H_{da}$ becomes $N-10$ and order of $N_{oa}$ is 8. Relative order of both functions are 1.

Fig. 5.19 shows $\Gamma(j\omega)$ in (5.1) with respect to $\omega$, where $C(j\omega)$ is optimal controller $C_{opt}$, $9^{th}$ order controller obtained by using software $C_{s,9}$ and $21^{st}$, $22^{nd}$ and $23^{rd}$ order controllers computed by using direct approximation of $C_{opt}$. The optimal performance level and performance levels obtained by using both approximation methods are shown in Fig. 5.20 with respect to order of controllers. Implementation of the controller designed by approximating $C_{opt}$ directly results with a performance level $\gamma_d$, which converges to approximately 5.3 as $N$ increases. On the other hand, when the controllers computed by running the software leads to a performance level $\gamma_s$, which converges to 1.96, that is only 0.03 greater than $\gamma_{\text{opt}}$, as $N$ increases. Furthermore, even the performance level of the system that uses $C_{s,9}$ is less than the one obtained with $C_{d,23}$.

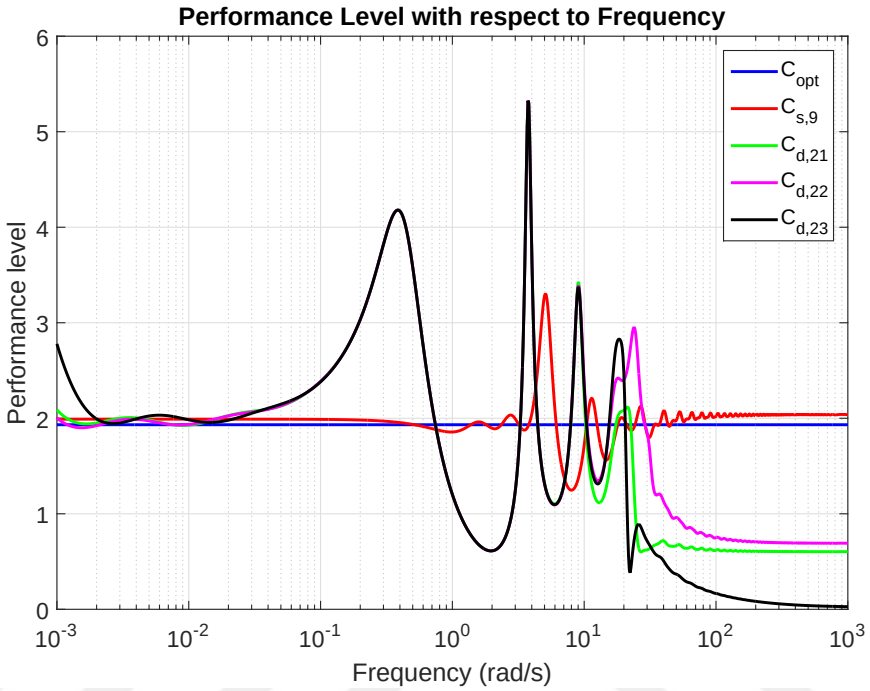


Figure 5.19: Performance levels of various controllers with respect to $\omega$ rad/s

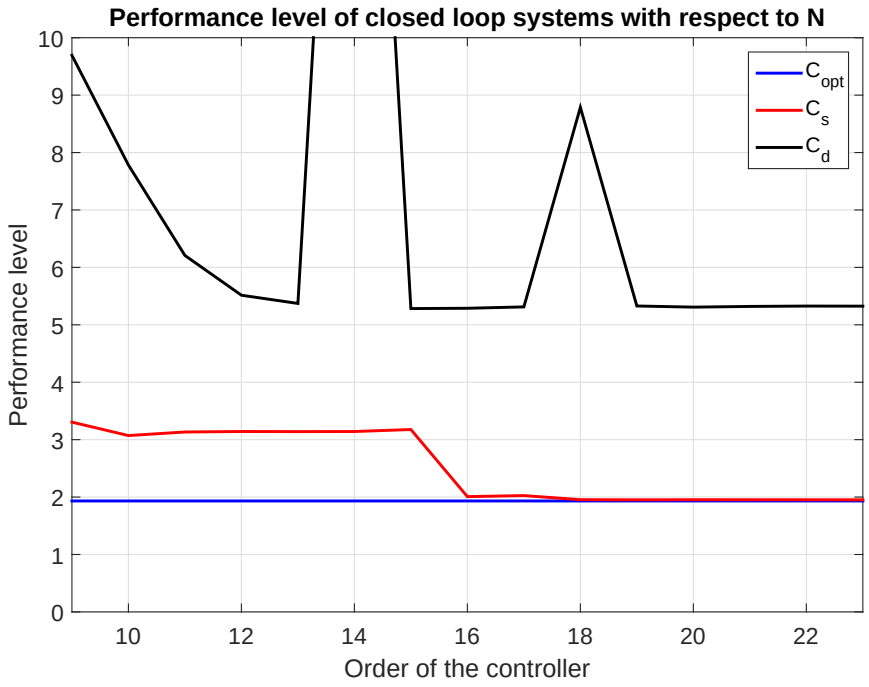


Figure 5.20: Performance levels of the systems with $C_d$ and $C_s$ with respect to order of controllers

## 5.4 Optimal Controller Dynamics for a Time Delay System with a Single Unstable Pole

Let the time delay system has one single pole in $\mathbb{C}_+$,

$$P(s) = \frac{e^{-hs}}{s-a}$$

where $a \in [0.95\ 1.05]$ and interval of $h$ is unknown. Let the nominal value of $a$ be $a_o = 1$ and nominal value of $h$ and its bound of parametric uncertainty be denoted by $h_o$ and $\delta_h$ respectively. $\delta_h = 0.1h$ is given. Therefore, we need to find the suboptimal controller structure that depends on $h$.

The multiplicative uncertainty weight is found as a $2^{nd}$ order improper function with a relative degree of $-2$, to provide a strictly proper optimal controller design, as shown below:

$$W_2(s) = (\delta_h s + 0.05)(\frac{\delta_h}{2}s + 1)$$

By selecting 5 different values of $h$, 5 optimal controllers are computed by using the software. Specifically, $h$ is chosen as $h = \{0.01, 0.1, 0.2, 0.5, 1\}$. To obtain $\gamma_{\text{opt}} \approx 1$ for all cases to observe the effect of $h$ and $\delta_h$ on optimal controller structure, the sensitivity weight $W_1(s)$ is defined as

$$W_1(s) = \frac{c(1 + s/1000)}{200\delta_h s + 0.001}$$

where the coefficient $c$ varies for each case. $c = \{20, 18, 15.5, 9, 2.9\}$ is chosen and $\gamma_{\text{opt}} = \{0.9861, 0.9977, 0.9999, 0.9747, 1.002\}$ is obtained for $h = \{0.01, 0.1, 0.2, 0.5, 1\}$ respectively.

Then, the optimal controllers are found whose Bode diagrams are shown in Fig. 5.21 for 5 different $h$ values. As it can be observed, the frequencies of resonances/anti-resonances and magnitude of the controllers are almost inversely proportional with $h$. To derive a generic structure for suboptimal controllers, infinite dimensional optimal controllers are approximated to $9^{th}$ order suboptimal controllers.

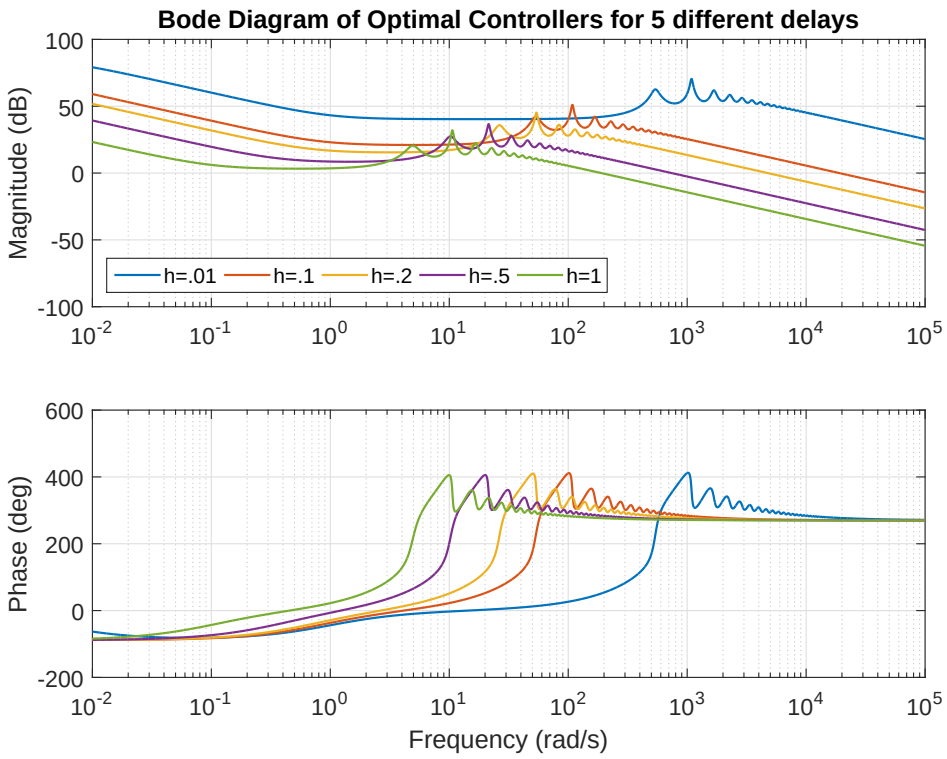


Figure 5.21: Bode diagrams of optimal controllers for $h = \{0.01, 0.1, 0.2, 0.5, 1\}$

All the controllers designed by running the software are strictly proper with a relative degree of 1. They have one conjugate pair of poles in $\mathbb{C}_+$, two conjugate pairs of poles and two real poles in $\mathbb{C}_-$, and one at $s = 0$. On the other hand, all of the 8 zeros are in $\mathbb{C}_-$. By using the transfer functions of the approximated sub-optimal controllers for 5 cases, a generic $9^{th}$ order suboptimal controller structure is derived for $h \leq 1$ second:

$$
\begin{aligned}
C_h(s) \;=\; & \left(\frac{0.36 - 0.2h^2}{h^{1.2}\; s}\right) \\
& \times \quad \frac{(1 + s/(1 - 0.9h^{0.5}))(1 + 1.4(sh/6.5) + (sh/6.5)^2)}{1 - (0.13 + 0.1h)(sh/(5.9 - h^{0.2})) + (sh/(5.9 - h^{0.2}))^2} \\
& \times \quad \frac{(1 + 0.8(sh/9.8) + (sh/9.8)^2)(1 + 0.4(sh/14.7) + (sh/14.7)^2)}{(1 + 0.05(sh/10.75) + (sh/10.75)^2)(1 + 0.15(sh/16.7) + (sh/16.7)^2)} \\
& \times \quad \frac{(1 + sh/5)}{(1 + sh/45)(1 + sh/19)}
\end{aligned}
$$

Fig. 5.22 shows the optimal and suboptimal controller designed by using $C_h(s)$

for $h = \{0.01, 0.2, 1\}$, where optimal controller for $h = 0.01$ is denoted by $C_{opt,0.01}$ and the corresponding suboptimal controller is defined as $C_{a,0.01}$.

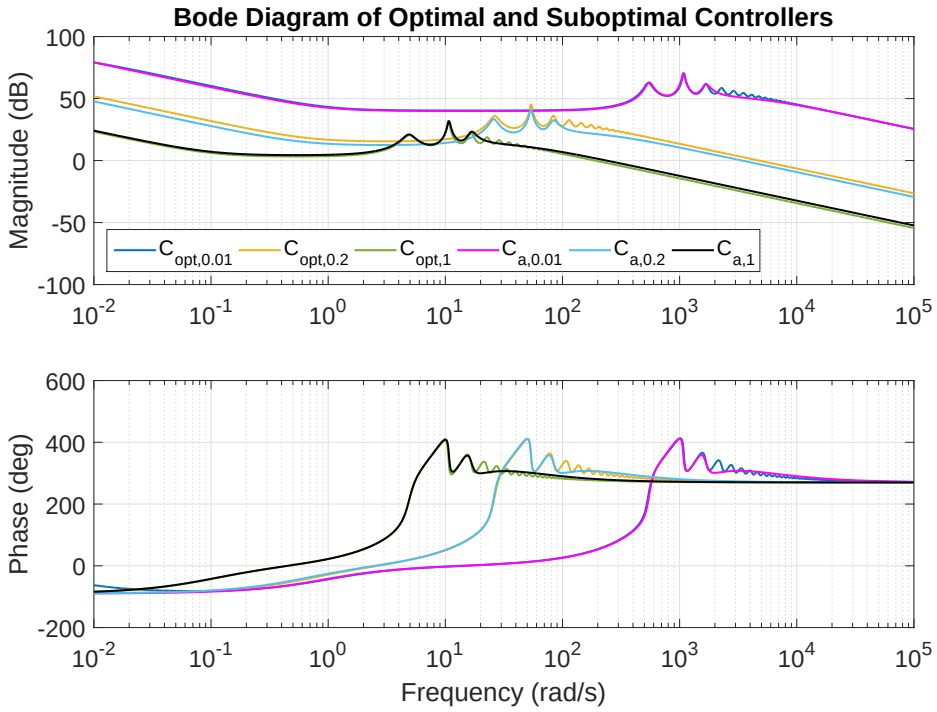


Figure 5.22: Bode diagrams of optimal and suboptimal controllers for $h = \{0.01, 0.2, 1\}$

## 5.5 Direct Approximation of Plant

Let the retarded time delay system be given as

$$P(s) = \frac{(2s+1)e^{-\tau_1 s} + (s^2 + 0.01s + 0.7)}{(s-2)e^{-\tau_2 s} + (2s^2 - 14s - 9)e^{-\tau_3 s} + (s^3 + 0.2s^2 + 4s + 11)} e^{-hs}$$

where $\tau_1 = 0.12$, $\tau_2 = 1$, $\tau_3 = 2$, $h \in [0\ 0.6]$ and the nominal time delay is $h_o = 0.3$ seconds, and the sensitivity weight is given as

$$W_1(s) = \frac{0.1(\epsilon s + 1)^2}{(s + \epsilon)^2}$$

where $\epsilon = 10^{-3}$. To compare the suboptimal controllers found by approximating plant to a finite dimensional function and software, we will use Pade approximation on system delays.

Firstly, let us determine the multiplicative error bound, i.e. the uncertainty weight $W_2$. Since the plant is strictly proper, we can find a bi-proper uncertainty weight to obtain a strictly proper optimal controller. For $h_o = 0.3$ and $h \in [0\ 0.6]$ seconds,

$$W_2(s) = \frac{0.32s}{(1 + s/1000)}$$

bounds all the multiplicative errors as shown in Fig. 5.23.

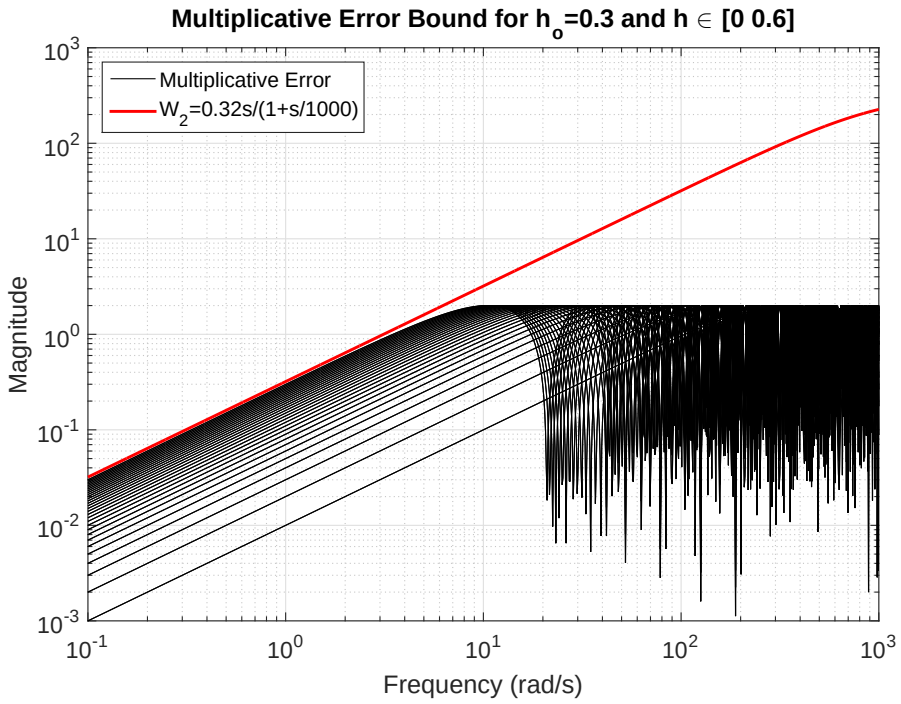


Figure 5.23: Multiplicative errors and the uncertainty weight

By performing the algorithm explained below, the delays in the system are approximated to finite dimensional functions by using Pade approximation:

Let the order of approximation of the plant be shown as $N$. Since the order of the system is 3 when $\tau_1 = \tau_2 = \tau_3 = h = 0$, the order that can be used for Pade approximation becomes $N - 3$. The order of approximations of each delay term are shown by $N_i$ where $i = \{1,\ 2,\ 3\}$ for the delay terms with $\tau_i$ and $i = \{4\}$ for the general system delay $h$. When $N = 7$, $N_i = 1$ for $i = \{1,\ 2,\ 3,\ 4\}$. As $N$ is incremented, $N_i$ are updated iteratively: Let the index of max $\left(\frac{\tau_1}{N_1},\ \frac{\tau_2}{N_2},\ \frac{\tau_3}{N_3},\ \frac{h}{N_4}\right)$ be $m$. Then, for each incrementation of $N$, $N_m$ is increased by 1. For example,

when $N = 14$, $N_i = \{1,\ 2,\ 4,\ 4\}$ for $i = \{1,\ 2,\ 3,\ 4\}$ respectively.

As the nominal plant is approximated to a finite dimensional system, the Matlab command `mixsyn` is used for $N \in [14\ 25]$ for $N \in \mathbb{Z}$ with the weights $W_1$ and $W_2$. The optimal performance levels obtained for each $N$ is saved under the variable called $\gamma_p$.

By running our software, the optimal performance level for the nominal plant is computed as $\gamma_{\text{opt}} = 2.4226$. Afterwards, the orders of approximation of $H_d$ and $N_o$ are equal to 7 and $N - 7$ respectively. Also, relative degrees of each function are defined as 1. Performance levels of suboptimal controllers are saved under the variable called $\gamma_s$. Fig. 5.24 shows the optimal performance level and performance levels of controllers obtained by both approximation. Note that the order of controllers designed by using Pade approximation are greater or equal to the orders of controllers designed by using software.

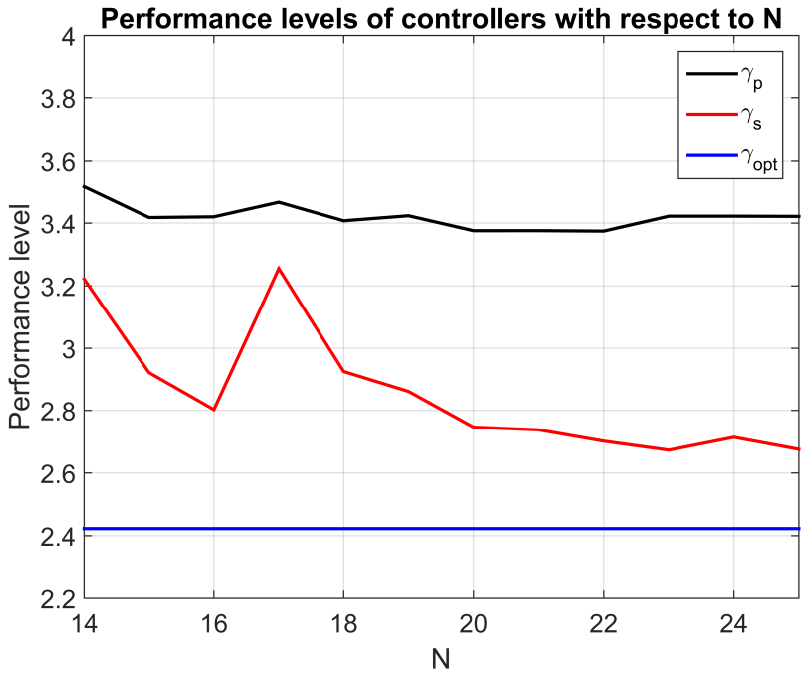


Figure 5.24: Performance levels of controllers with respect to $N$

The results show that the performance levels achieved by using the software are smaller than those obtained by using direct approximation of the plant.

# Chapter 6

# Conclusion

Mathematical model of any practical system in engineering applications uses assumptions and neglects some dynamics in order to derive simplified equations. Therefore, the errors in practical systems with respect to theoretical models leads to uncertainties in the proposed model, regarded as nominal plant. To stabilize time-delay systems with uncertainties, robust controllers are designed by the control engineers. Since performance with respect to a given reference signal also needs attention during robust controller design, the mixed sensitivity problem needs to be solved in many engineering applications. This thesis introduces a software that uses Toker-Özbay formula, which numerically computes the optimal controller by solving mixed sensitivity problem for a given infinite dimensional system, to derive a finite dimensional robustly stabilizing suboptimal controller.

Since the formula is used to numerically compute the optimal controller, and its structure given in [7] has internal pole/zero cancellations, approximation of the terms expressed in the formula may lead to uncanceled unstable pole/zero pairs. This thesis introduces a new structure, which decomposes denominator to separate the optimal controller into finite dimensional terms and two infinite dimensional terms in $\mathcal{H}_\infty$. Therefore, the new structure is more reliable to implement on practical systems.

Although a Matlab package called HINFCON was published in 1996 that uses Toker-Özbay formula to derive optimal controllers, it needs manual computations of coprime factorization and optimal performance levels. The software introduced in this thesis provides a user-friendly GUI, uses YALTA to perform coprime factorization on the given retarded/neutral system and runs an algorithm to compute optimal performance level. Furthermore, given the orders of approximation, the optimal controller can be approximated to a finite dimensional suboptimal controller by using the new structure.

Various examples are solved to illustrate different uses of the software. In the first example, a switching suboptimal controller is designed by using the software to meet the given criteria. The second example shows how the performance level of suboptimal controller varies with respect to the optimal controller given a time-delay system that contains internal delays in its dynamics. Suboptimal controllers designed by using direct approximation of the optimal controller and approximating the infinite dimensional terms in the new structure separately are compared with respect to order of approximation in the third example. The results of that example shows that the performance level of suboptimal controllers designed in the second case converges to optimal performance level as the order of approximation increases, while performance level with other controllers do not. A generic controller design for a simple time-delay system is obtained in the fourth example. Finally, the last example compares the performance levels of finite dimensional controllers by using an approximation of an infinite dimensional plant and the new structure, which shows that again, suboptimal controllers designed by using our software are better in terms of performance levels. In addition, suboptimal controllers designed by using the software are tested on a practical system which fulfilled the desired performance and robustness criteria.

As a future work, a new algorithm can be implemented for the approximation of the new structure such that the performance level of optimal controller with the specified order gets optimized. Additionally, a cost function can be defined to minimize order and performance level at the same time. In addition to current inputs of the software, number of criteria for the design of suboptimal controller can be increased by incorporating gain, phase and delay margins etc.

# Bibliography


[1] G. Zames, "Feedback and optimal sensitivity: model reference transformations, multiplicative seminorms, and approximate inverses," *IEEE Trans. Automatic Control*, vol. 16, pp. 301–320, 1981.

[2] B. Francis, J. Helton, and G. Zames, "$\mathcal{H}_\infty$-optimal feedback controllers for linear multivariable systems," *IEEE Trans. Automatic Control*, vol. 29, pp. 888–900, 1984.

[3] B. Chang and J. Pearson, "Optimal disturbance rejection in linear multivariable systems," *IEEE Trans. Automatic Control*, vol. 29, pp. 880–887, 1984.

[4] C. Foias, A. Tannenbaum, and G. Zames, "Weighted sensitivity minimization for delay systems," *IEEE Trans. Automatic Control*, vol. 31, pp. 763–766, 1986.

[5] K. Zhou and P. Khargonekar, "On the weighted sensitivity minimization problem for delay systems," *Systems & Control Letters*, vol. 8, pp. 307–312, 1987.

[6] T. Lypchuk, M. Smith, and A. Tannenbaum, "Weighted sensitivity minimization: General plants in $\mathcal{H}_\infty$ and rational weights," *Linear Algebra and its Applications*, vol. 109, pp. 71–90, 1988.

[7] O. Toker and H. Özbay, "$\mathcal{H}_\infty$ optimal and suboptimal controllers for infinite dimensional siso plants," *IEEE Trans. Automatic Control*, vol. 40, pp. 751–755, 1995.

[8] S. Gümüşsoy, "Coprime-inner/outer factorization of siso time-delay systems and fir structure of their optimal $\mathcal{H}_\infty$ controllers," *Int. J. Robust Nonlin. Cont.*, vol. 22, pp. 981–998, 2012.

[9] M. O. Yegin and H. Özbay, "Numerical computation of $\mathcal{H}_\infty$ optimal controllers for time delay systems using yalta," *IFAC-PapersOnLine*, vol. 49, no. 10, pp. 182–187, 2016.

[10] D. Avanessoff, A. R. Fioravanti, and C. Bonnet, "Yalta: a matlab toolbox for the $\mathcal{H}_\infty$-stability analysis of classical and fractional systems with commensurate delays," in *IFAC Symposium on System, Structure and Control*, (Grenoble, France), pp. 839–844, 2008.

[11] H. Özbay, "$\mathcal{H}_\infty$ optimal controller design for a class of distributed parameter systems," *Int. J. Cont.*, vol. 58, pp. 739–782, 1993.

[12] V. M. Adamjan, D. Z. Arov, and M. G. Kre?n, "Analytic properties of schmidt pairs for a hankel operator and the generalized schur-takagi problem," *Mathematics of the USSR-Sbornik*, vol. 15, no. 1, p. 31, 1971.

[13] C. Foias, H. Özbay, and A. Tannenbaum, "Robust control of infinite dimensional systems: Frequency domain methods," in *Lecture Notes in Control and Information Sciences*, vol. 209, London: Springer, 1996.

[14] H. Özbay, "Computation of $\mathcal{H}_\infty$ controllers for infinite dimensional plants using numerical linear algebra," *Numerical Linear Algebra with Applications*, vol. 20, no. 2, pp. 327–335, 2013.

[15] O. Toker and H. Özbay, *HINFCON: A matlab based program for $\mathcal{H}_\infty$ Optimal/Suboptimal Controller Design*, 1996. Available at `http://kilyos.ee.bilkent.edu.tr/~ozbay/HINFCON.zip`. Accessed on: 07 Feb 2017.

[16] D. Avanessoff, C. Bonnet, H. Cavalera, A. Fioravanti, and L. Nguyen, *User document: YALTA*, September 2015.

[17] C. Bonnet, A. R. Fioravanti, and J. R. Partington, "Stability of neutral systems with commensurate delays and poles asymptotic to the imaginary

axis," *SIAM Journal on Control and Optimization*, vol. 49, no. 2, pp. 498–516, 2011.

[18] A. R. Fioravanti, C. Bonnet, and H. Özbay, "Stability of fractional neutral systems with multiple delays and poles asymptotic to the imaginary axis," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 31–35, 2010.

[19] A. R. Fioravanti, C. Bonnet, H. Özbay, and S.-I. Niculescu, "A numerical method to find stability windows and unstable poles for linear neutral time-delay systems," *IFAC Proceedings Volumes*, vol. 43, no. 2, pp. 183–188, 2010.

[20] A. R. Fioravanti, C. Bonnet, H. Özbay, and S.-I. Niculescu, "A numerical method for stability windows and unstable root-locus calculation for linear fractional time-delay systems," *Automatica*, vol. 48, no. 11, pp. 2824–2830, 2012.

[21] J. R. Partington, "Approximation of unstable infinite-dimensional systems using coprime factors," *Systems & Control Letters*, vol. 16, no. 2, pp. 89–96, 1991.

[22] P. M. Makila and J. R. Partington, "Laguerre and kautz shift approximations of delay systems," *International Journal of Control*, vol. 72, no. 10, pp. 932–946, 1999.

[23] A. E. Karagül, O. Demir, and H. Özbay, "Computation of Optimal $\mathcal{H}_\infty$ Controllers and Approximations of Fractional Order Systems: a tutorial review", *Applied and Computational Mathematics*, vol. 12 (2013), pp. 261–288.

# Appendix A

# Matlab Codes

```
%% Uses YALTA to perform coprime factorization
syms s;
rowNu=size(qNum,1);
index=0;
for k=2:rowNu
    if ~isempty(find(qNum(k,:),1))
        index=index+1;
        delayVectorNum(index) = k-1;
    end
end
if ~exist('delayVectorNum')
    delayVectorNum=[1];
    qNum = [qNum(1,:);zeros(1,size(qNum,2)-1) 1e-30];
    adderNum = 1e-30*exp(-hNum*s);
else
    qNum = [qNum(1,:);qNum(delayVectorNum+1,:)];
    adderNum = 0;
end
rowNu=size(qDen,1);
index=0;
for k=2:rowNu
    if ~isempty(find(qDen(k,:),1))
        index=index+1;
        delayVectorDen(index) = k-1;
    end
end
if ~exist('delayVectorDen')
    delayVectorDen=[1];
    qDen = [qDen(1,:);zeros(1,size(qDen,2)-1) 1e-30];
    adderDen = 1e-30*exp(-hDen*s);
else
    qDen = [qDen(1,:);qDen(delayVectorDen+1,:)];
    adderDen = 0;
end
if length(qNum(1,:)) == 1
    uNum = [];
else
    num = delayFrequencyAnalysisMin(qNum,delayVectorNum,alphaNum,hNum);
    uNum = num.UnstablePoles;
end
```

```
reversed = 0;
try
    if isnan(uNum)
        qNum2 = qNum(end:-1:1,:);
        for col = 0:(size(qNum2,2)-1)
            qNum2(:,size(qNum2,2)-col) = (-1)^(col)*qNum2(:,size(qNum2,2)-col);
        end
        num = delayFrequencyAnalysisMin(qNum2,delayVectorNum,alphaNum,hNum);
        uNum = num.UnstablePoles;
        reversed = 1;
    else
    end
catch
end
Mn_Num = poly2sym(real(poly(uNum)),s);
Mn_Den = poly2sym(real(poly(-uNum)),s);

if reversed==1
    Mn_Num2 = 0; Mn_Den2 = 0;
    for k=1:size(qNum,1)
        Mn_Num2 = Mn_Num2+(poly2sym(qNum(k,:),s)*exp(-hNum*(k-1)*s));
        Mn_Den2 = Mn_Den2+(poly2sym(qNum2(k,:),s)*exp(-hNum*(k-1)*s));
    end
    Mn_Num = Mn_Num* Mn_Num2;
    Mn_Den = Mn_Den* Mn_Den2;
end
if ~exist('Mn_Num','var')
   Mn_Num = 1;
   Mn_Den = 1;
end
Mn_Num = Mn_Num*exp(-ho*s);
den=delayFrequencyAnalysisMin(qDen,delayVectorDen,alphaDen,hDen,1,1e-3);
uNum = den.UnstablePoles;
if isnan(uNum)
    Md_Num = s/s;
    Md_Den = s/s;
    bull = 1;
    Md = Md_Num/Md_Den
else
    Md_Num = poly2sym(real(poly(uNum)),s);
    Md_Den = poly2sym(real(poly(-uNum)),s);
end

if ~exist('Md_Num')
    Md_Num = s/s;
    Md_Den = s/s;
    bull = 1;
else
    bull = 0;
end
Mn = Mn_Num/Mn_Den;

alpha = uNum;
M = Mn/Md;
for k=1:(length(delayVectorNum)+1)
    if k==1
        Plant_Num = poly2sym(qNum(1,:),s);
    else
        Plant_Num = Plant_Num+poly2sym(qNum(k,:),s)*exp(-delayVectorNum(k-1)*hNum*s);
    end
end
Plant_Num = Plant_Num - adderNum;

for k=1:(length(delayVectorDen)+1)
    if k==1
        Plant_Den = poly2sym(qDen(1,:),s);
    else
        Plant_Den = Plant_Den+poly2sym(qDen(k,:),s)*exp(-delayVectorDen(k-1)*hDen*s);
```

```
    end
end
Plant_Den = Plant_Den - adderDen;
Plant = Plant_Num*exp(-ho*s)/Plant_Den;
No = Plant/M;
```

```
%% Computes optimal performance level by using iterative algorithm
function [gamma_min, xvalue, yvalue, newFunc, beta_fin]= ...
    gammaOpt(nw1,dw1,nw2,dw2,alpha,Npoints,gmin,gmax,EPS,Mn)
rootsEPS = 1e-5;
minValCoeff=0.1;
iterNu = 0;
ind = 1;
while 1
    xval=[];
    yval=[];
    iterNu = iterNu+1;
    minValCoeff=minValCoeff*0.05;
    for gamma=gmin:(gmax-gmin)/Npoints:gmax,
    a1=polyadd(conv(nw1,star(nw1)),-gamma^2*conv(dw1,star(dw1)));
    a1 = a1(find(a1,1):end);

    b1=gamma^2*conv(dw1,star(dw1));
    b1 = b1(find(b1,1):end);

    a2=polyadd(conv(nw2,star(nw2)),-gamma^2*conv(dw2,star(dw2)));
    a2 = a2(find(a2,1):end);
    b2=gamma^2*conv(dw2,star(dw2));
    b2 = b2(find(b2,1):end);
    A1=conv(a1,a2);
    B1=conv(b1,b2);
    A=polyadd(B1,-A1);
    B=B1;
    [dG,nG]=spec(A,B);
    eta=roots(dw1);
    n1=length(eta);
    nF=conv(nG,poly(-eta));
    dF=conv(dG,poly(eta));

    beta=roots(a1);
    beta=beta(find((real(beta)<rootsEPS) | ((abs(real(beta))<=rootsEPS) & (imag(beta)>=0))));
    l=length(alpha);
    if abs(gamma-(gmin+gmax)/2)<1e-6
        beta_fin = beta;
    end
    betas = beta;
      M=[];
       for k=1:l,
           s=alpha(k);
           M=[M; alpha(k).^[0:n1+l-1] eval(Mn)*polyval(nF,alpha(k)) ...
               /polyval(dF,alpha(k))*alpha(k).^[0:n1+l-1]];
       end;
       for k=1:n1,
           s=beta(k);
           M=[M; beta(k).^[0:n1+l-1] eval(Mn)*polyval(nF,beta(k)) ...
               /polyval(dF,beta(k))*beta(k).^[0:n1+l-1]];
       end;
       for k=1:l,
           s=alpha(k);
           M=[M;(-alpha(k)).^[0:n1+l-1]*eval(Mn)*polyval(nF,alpha(k)) ...
               /polyval(dF,alpha(k)) (-alpha(k)).^[0:n1+l-1]];
       end;

       for k=1:n1,
           s=beta(k);
           M=[M; (-beta(k)).^[0:n1+l-1]*eval(Mn)*polyval(nF,beta(k)) ...
               /polyval(dF,beta(k)) (-beta(k)).^[0:n1+l-1]];
       end;
       assignin('base','M',M);
       xval=[xval gamma];
       try
            yval=[yval min(svd(M))];
       catch error
           error
```

```
            break;
        end

    end;

    yval=fliplr(yval);
    xval=fliplr(xval);
    if iterNu==1
        xvalue = xval;
        yvalue=abs(yval);
        plot(xvalue,yvalue)
        newFunc=-yval;
        figure;
        plot(xval,newFunc)
        [~,b]=findpeaks(newFunc);
        gamma_opt=xval(b);
        try
            minValCoeff=0.5*(gamma_opt(ind:ind+1)*[1;-1]);
        catch
            if gamma_opt(ind)>1e-2
                minValCoeff = 1e-2;
            else
                minValCoeff = 3*gamma_pos/4;
            end
        end
        gamma_pos = gamma_opt(ind);
        temp = yval(b);
    else
        [~,b]=min(yval);
        gamma_pos=xval(b);
    end
    if yval(b)<EPS
        Jmatrix = zeros(1,n1);
        for ind=1:n1
            Jmatrix(ind) = (-1)^(ind+1);
        end
        Jmatrix = diag(Jmatrix); s = beta(1:n1);
        Dn1 = diag(eval(Mn).*polyval(nF,beta(1:n1))./polyval(nF,beta(1:n1)));
        break;
    else
        if iterNu>3 && yval(b)>(4*min(temp)/5)
            iterNu = 1;
            ind = ind+1;
            gamma_pos = gamma_opt(ind);
            try
                minValCoeff=0.5*(gamma_opt(ind:ind+1)*[1;-1]);
            catch
                if gamma_opt(ind)>1e-2
                    minValCoeff = 1e-2;
                else
                    minValCoeff = 3*gamma_pos/4;
                end
            end
        end
        temp = yval(b);min(temp)
        gmin=gamma_pos-minValCoeff
        gmax=gamma_pos+minValCoeff
    end
end
gamma_min = gamma_pos;
```

```
%% Computes optimal and suboptimal controller by using new structure
function [warning, E_gamma, G_gamma, F_gamma, L_gamma, Ro, Hn, app_Hd, Hd, hat_G_gamma, ...
    per, Ca, app_No, Copt] = optController(gamma_opt, nw1, dw1, nw2, dw2, alpha, eps, ...
        lw1,lw2,Lpoints,Mn,Md,No,order_inputs)

syms s;
Ro = nan;
Hn = nan;
EPSSS=eps;
Plant = Mn*No/Md;
% Calculation of W1(s)
W1_n = poly2sym(nw1,s);
W1_d = poly2sym(dw1,s);
W1_s = W1_n/W1_d;
% Calculation of W2(s)
W2_n = poly2sym(nw2,s);
W2_d = poly2sym(dw2,s);
W2_s = W2_n/W2_d;

nwm1=zeros(1,length(nw1));
dwm1=zeros(1,length(dw1));
nwm2=zeros(1,length(nw2));
dwm2=zeros(1,length(dw2));
% Calculation of W1(-s)
for k = 1:length(nw1)
    nwm1(k) = nw1(k)*(-1)^(length(nw1)-k);
end
for k = 1:length(dw1)
    dwm1(k) = dw1(k)*(-1)^(length(dw1)-k);
end
mW1_n = poly2sym(nwm1,s);
mW1_d = poly2sym(dwm1,s);
W1_ms = mW1_n/mW1_d;

% Calculation of W2(-s)
for k = 1:length(nw2)
    nwm2(k) = nw2(k)*(-1)^(length(nw2)-k);
end
for k = 1:length(dw2)
    dwm2(k) = dw2(k)*(-1)^(length(dw2)-k);
end
mW2_n = poly2sym(nwm2,s);
mW2_d = poly2sym(dwm2,s);
W2_ms = mW2_n/mW2_d;

den_E_gamma = W1_d*mW1_d*gamma_opt^2;
num_E_gamma = W1_n*mW1_n-den_E_gamma;
E_gamma = minreal(sym2tf(num_E_gamma/den_E_gamma),EPSSS);

a1=polyadd(conv(nw1,star(nw1)),-gamma_opt^2*conv(dw1,star(dw1)));
a1 = a1(find(a1,1):end);
b1=gamma_opt^2*conv(dw1,star(dw1));
b1 = b1(find(b1,1):end);

a2=polyadd(conv(nw2,star(nw2)),-gamma_opt^2*conv(dw2,star(dw2)));
a2 = a2(find(a2,1):end);
b2=gamma_opt^2*conv(dw2,star(dw2));
b2 = b2(find(b2,1):end);

A1=conv(a1,a2);
B1=conv(b1,b2);
A=polyadd(B1,-A1);
B=B1;
[dG,nG]=spec(A,B);
eta=roots(dw1);
n1=length(eta);
nF=conv(nG,poly(-eta));
dF=conv(dG,poly(eta));
```

```
hat_G_gamma = -poly2sym(nG,s)/poly2sym(dG,s)/gamma_opt;%%%!!!!!!
hat_G_gamma = minreal(sym2tf(hat_G_gamma),EPSSS);
hat_G_gamma = tf2sym(hat_G_gamma);
G_gamma = hat_G_gamma*W1_s;
G_gamma = minreal(sym2tf(G_gamma),EPSSS);
G_gamma = tf2sym(G_gamma);
num_F_gamma = poly2sym(nF,s);
den_F_gamma = poly2sym(dF,s);
F_gamma = minreal(sym2tf(num_F_gamma/den_F_gamma),EPSSS);
beta=roots(sym2poly(num_E_gamma));
beta_checker = beta(abs(real(beta))<=1e-5);
indices = [];
for k=1:length(beta_checker)
    if imag(beta_checker(k))==0
        indices = [indices k];
    end
end
beta_checker = beta_checker(indices(1:length(indices)/2));
beta=[beta((real(beta)>1e-5) | ((abs(real(beta))<=1e-5) & (imag(beta)>0))) beta_checker];
len_alpha = length(alpha);
if length(beta)==length(roots(dw1))
    M=[];
    for k=1:len_alpha,
        s = alpha(k);
        M=[M; alpha(k).^(0:n1+len_alpha-1), ...
            eval(Mn)*eval(F_gamma)*alpha(k).^[0:n1+len_alpha-1]];
    end;
    for k=1:n1,
        s = beta(k);
        M=[M; beta(k).^[0:n1+len_alpha-1], ...
            eval(Mn)*eval(F_gamma)*beta(k).^(0:n1+len_alpha-1)];
    end;
    for k=1:len_alpha,
        s = alpha(k);
        M=[M;(-alpha(k)).^[0:n1+len_alpha-1]*eval(Mn)*eval(F_gamma),...
            (-alpha(k)).^[0:n1+len_alpha-1]];
    end;
    for k=1:n1,
        s = beta(k);
        M=[M; (-beta(k)).^[0:n1+len_alpha-1]*eval(Mn)*eval(F_gamma),...
            (-beta(k)).^[0:n1+len_alpha-1]];
    end;

    [~,Suu,Vuu]=svd(M);
    LL=Vuu(:,length(M));
    Suu(length(M),length(M))
    if Suu(length(M),length(M)) < eps
        syms s;
        num_L_gamma=poly2sym(real(LL(2*(n1+len_alpha):-1:n1+len_alpha+1))+...
            imag(LL(2*(n1+len_alpha):-1:n1+len_alpha+1)),s);
        den_L_gamma=poly2sym(real(LL(n1+len_alpha:-1:1))+imag(LL(n1+len_alpha:-1:1)),s);
        L_gamma = num_L_gamma/den_L_gamma;
        L_gamma = minreal(sym2tf(L_gamma),EPSSS);
        L_gamma = tf2sym(L_gamma);
        K_opt = 1/L_gamma;
        K_opt = minreal(sym2tf(K_opt),EPSSS);
        K_opt = tf2sym(K_opt);
        sym2zpk(K_opt)
        Ro = W1_n*W1_d/(num_E_gamma*Md);
        Ro = minreal(sym2tf(Ro),EPSSS);
        Ro = tf2sym(Ro);
        s=1e+45;
        dInf=real(eval(Ro)*eval(K_opt)/gamma_opt);
        syms s;
        M1 = mW1_d/W1_d;
        [Hn,Hd]=findHnHd(gamma_opt,K_opt,Ro,Mn,M1,hat_G_gamma,dInf);
        Co = 1/(1+Hn);
        lw=lw1:(lw2-lw1)/Lpoints:lw2;
```

```
wval=10.^lw;

try sym2tf(Hd)
    if (order_inputs.DO_APP_EVEN_FINITE==1)
        app_Hd = doApprox(Hd,wval,order_inputs.Hd_order,...
            order_inputs.Hd_rel_order,Co/(1+Co*Hd));
        Hd_title = ['Bode Plot of H_d(s) and Its ',...
            num2str(order_inputs.Hd_order),' Order Approximation'];
    else
        app_Hd = Hd;
        Hd_title='Bode Plot of H_d(s) Without Approximation (Finite Dimensional)';
    end
catch
    app_Hd = doApprox(Hd,wval,order_inputs.Hd_order,...
        order_inputs.Hd_rel_order,Co/(1+Co*Hd));
    Hd_title = ['Bode Plot of H_d(s) and Its ',...
        num2str(order_inputs.Hd_order),' Order Approximation'];
end

try sym2tf(No)
    if (order_inputs.DO_APP_EVEN_FINITE==1)
        app_No = doApprox(No,wval,order_inputs.No_order,...
            order_inputs.No_rel_order,No);
        No_bode = ['Bode Plot of N_o(s) and Its ',...
            num2str(order_inputs.No_order),' Order Approximation'];
    else
        app_No = No;
        No_bode='Bode Plot of N_o(s) Without Approximation (Finite Dimensional)';
    end
catch
    app_No = doApprox(No,wval,order_inputs.No_order,...
        order_inputs.No_rel_order,No);
    No_bode = ['Bode Plot of N_o(s) and Its ',...
        num2str(order_inputs.No_order),' Order Approximation'];
end


term = (Co/(1+Co*app_Hd));
Ca = W1_s/(dInf*gamma_opt^2)*term*hat_G_gamma/app_No;
Copt = W1_s/(dInf*gamma_opt^2)*(Co/(1+Co*Hd))*hat_G_gamma/No;

s = 1j*wval;
ca = eval(Ca);
c=(eval(W1_s)/(dInf*gamma_opt^2)).*(eval(Co)./(1+eval(Co).*eval(Hd))).*...
    eval(hat_G_gamma)./eval(No);

psi_inf=sqrt(abs(eval(W1_s)).^2+abs(eval(W2_s).*eval(Plant).*c).^2)./...
    abs(1+eval(Plant).*c);
psi_inf2=sqrt(abs(eval(W1_s)).^2+abs(eval(W2_s).*eval(Plant).*ca).^2)./...
    abs(1+eval(Plant).*ca);
p=eval(Plant);
S=(1+p.*c).^(-1);
T=1-S;
perMatrix=[eval(W1_s).*S; eval(W2_s).*T];
per = sqrt(abs(perMatrix(1,:)).^2+abs(perMatrix(2,:)).^2);
cmag=abs(c);
cang = angle(c);
hd_app_mag=abs(eval(app_Hd));
hd_app_ang=angle(eval(app_Hd));
hdmag=abs(eval(Hd));
hdang=angle(eval(Hd));
camag=abs(ca);
caang = angle(ca);
no_app_mag=abs(eval(app_No));
no_app_ang=angle(eval(app_No));
nomag=abs(eval(No));
noang=angle(eval(No));
pcr=real(p.*c);
```

```
pci=imag(p.*c);
pcr2=real(p.*ca);
pci2=imag(p.*ca);
figure(6);
clf;
subplot(2,1,1);
semilogx(wval,20*log10(cmag));
hold on;
semilogx(wval,20*log10(camag),'r');
grid on;
ylabel('Magnitude (dB)');
legend('C_o_p_t','C_a')
title('Bode Plots of Controllers');
set(gca,'FontSize',16);

subplot(2,1,2);
semilogx(wval,unwrap(cang)*180/pi);
hold on;
semilogx(wval,unwrap(caang)*180/pi,'r');
grid on;
xlabel('Frequency (rad/s)');
ylabel('Phase (deg)');
set(gca,'FontSize',16);
set(gcf,'Name','Bode Plots of Controllers');


figure(7);
clf;
subplot(1,1,1);
plot(pcr,pci,pcr,-pci,'-.');
hold on;
arrow(pcr(1),pci(1),pcr(10),pci(10));
plot(-1,0,'kx','LineWidth',4)
grid on;
title('Nyquist plot with C_{opt}(j\omega)');
set(gca,'FontSize',16);
set(gcf,'Name','Nyquist plot with Copt(jw)');

%%%%%
figure(8);
clf;
subplot(1,1,1);
plot(pcr2,pci2,pcr2,-pci2,'-.');
hold on;
arrow(pcr2(1),pci2(1),pcr2(10),pci2(10));
plot(-1,0,'kx','LineWidth',4)
grid on;
title('Nyquist plot with C_a(j\omega)');
set(gca,'FontSize',16);
set(gcf,'Name','Nyquist plot with Ca(jw)');


figure(9)
clf;
subplot(2,1,1);
semilogx(wval,20*log10(hdmag));
grid on; hold on;
semilogx(wval,20*log10(hd_app_mag));
ylabel('Magnitude (dB)');
title(Hd_title);
legend('H_d','H_d_a');
set(gca,'FontSize',16);

subplot(2,1,2);
semilogx(wval,unwrap(hdang)*180/pi);
grid on;
hold on;
semilogx(wval,unwrap(hd_app_ang)*180/pi);
```

```
        xlabel('\omega (rad/s)');
        ylabel('Phase (deg)');
        set(gca,'FontSize',16);
        set(gcf,'Name','Bode Plot of Hd(s)');


        figure(10)
        clf;
        subplot(2,1,1);
        semilogx(wval,20*log10(nomag));
        grid on;
        hold on;
        semilogx(wval,20*log10(no_app_mag));
        ylabel('Magnitude (dB)');
        title(No_bode);
        legend('N_o','N_{oa}');
        set(gca,'FontSize',16);

        subplot(2,1,2);
        semilogx(wval,unwrap(noang)*180/pi);
        grid on;
        hold on;
        semilogx(wval,unwrap(no_app_ang)*180/pi);
        xlabel('\omega (rad/s)');
        ylabel('Phase (deg)');
        set(gca,'FontSize',16);
        set(gcf,'Name','Bode Plot of No(s)');

        figure(11);
        clf;
        semilogx(wval,(psi_inf),'b');
        hold on
        semilogx(wval,(psi_inf2),'r');
        gamma_app = max(psi_inf2)
        grid;
        xlabel('Frequency (rad/s)');
        ylabel('Performance level');
        set(gca,'fontsize', 22)
        title('Performance plot');
        legend('C_o_p_t','C_a');
        set(gca,'FontSize',16);
        set(gcf,'Name','Performance plot');

        warning = 'Correct GAMMA_OPT value';
    else
        warning = 'Incorrect GAMMA_OPT value'
    end
else
    warning = 'Non-generic case and/or incorrect GAMMA_OPT value'
end
```

```
%% Returns approximation of TF by using given order, relative order and weight
function result = doApprox(mySym,wval,approx_order,rel_order,weight_sym)
s = 1j*wval;
if nargin<5
    WFd = 1;
else
    WFd = frd(eval(weight_sym),wval);
end
data = eval(mySym);
dc_gain = abs(data(1));
data = data/dc_gain;
FFd=frd(data,wval);
if isempty(rel_order)
    Ff=fitfrd(FFd,approx_order,[],WFd);
else
    Ff=fitfrd(FFd,approx_order,rel_order,WFd);
end
[result_n,result_d] = ss2tf(Ff.a,Ff.b,Ff.c,Ff.d);
syms s;
result = dc_gain*poly2sym(result_n,s)/poly2sym(result_d,s);
```