A Generalized $\mathcal{H}^\infty$ Mixed Sensitivity Convex Approach to Multivariable Control Design Subject to Simultaneous Output and Input Loop-Breaking Specifications

by

Karan Puttannaiah

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved July 2018 by the
Graduate Supervisory Committee:

Armando A. Rodriguez, Chair
Spring M. Berman
Hans D. Mittelmann
Konstantinos Tsakalis

ARIZONA STATE UNIVERSITY

August 2018

ABSTRACT

In this dissertation, we present a H-infinity based multivariable control design methodology that can be used to systematically address design specifications at distinct feedback loop-breaking points. It is well understood that for multivariable systems, obtaining good/acceptable closed loop properties at one loop-breaking point does not mean the same at another. This is especially true for multivariable systems that are ill-conditioned (having high condition number and/or relative gain array and/or scaled condition number). We analyze the tradeoffs involved in shaping closed loop properties at these distinct loop-breaking points and illustrate through examples the existence of pareto optimal points associated with them. Further, we study the limitations and tradeoffs associated with shaping the properties in the presence of right half plane poles/zeros, limited available bandwidth and peak time-domain constraints. To address the above tradeoffs, we present a methodology for designing multiobjective constrained H-infinity based controllers, called Generalized Mixed Sensitivity (GMS), to effectively and efficiently shape properties at distinct loop-breaking points. The methodology accommodates a broad class of convex frequency- and time-domain design specifications. This is accomplished by exploiting the Youla-Jabr-Bongiorno-Kucera parameterization that transforms the nonlinear problem in the controller to an affine one in the Youla et al. parameter. Basis parameters that result in efficient approximation (using lesser number of basis terms) of the infinite-dimensional parameter are studied. Three state-of-the-art subgradient-based non-differentiable constrained convex optimization solvers, namely Analytic Center Cutting Plane Method (ACCPM), Kelley's CPM and SolvOpt are implemented and compared.

The above approach is used to design controllers for and tradeoff between several control properties of longitudinal dynamics of 3-DOF Hypersonic vehicle model - one that is unstable, non-minimum phase and possesses significant coupling between chan-

nels. A hierarchical inner-outer loop control architecture is used to exploit additional feedback information in order to significantly help in making reasonable tradeoffs between properties at distinct loop-breaking points. The methodology is shown to generate very good designs - designs that would be difficult to obtain without our presented methodology. Critical control tradeoffs associated are studied and compared with other design methods (e.g., classically motivated, standard mixed sensitivity) to further illustrate its power and transparency.

To my parents Puttannaiah N. & Radha D.,

and

my uncle Sreenivas R.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Armando A. Rodriguez who has guided and supported me immensely throughout my journey in Graduate School. I have acquired a wealth of knowledge from him in both technical and non-technical aspects. He has been a great positive influence on me and has changed the way I think. I am truly indebted to him.

I am grateful to the members of my thesis committee, Dr. Spring M. Berman, Dr. Hans D. Mittelmann and Dr. Konstantinos Tsakalis for their continuous guidance and support. I would also like to thank Dr. Jennie Si who has helped me a great deal in my research and classes. They have always been very understanding and accomodating, and often went out their ways to help me. Further, I would like to acknowledge the support of Dr. Joseph C. Palais and the School of Electrical, Computer and Energy Engineering at Arizona State University (ASU), which made my PhD possible through assistantships and financial support.

I would like to thank my manager at Intel Corporation, Dr. Niveditha Sundaram who helped gain valuable industry experience during my internship. She has put generous trust and been patient during my internship and final stages of PhD.

I am thankful for the support from colleagues at ASU - Shiba Biswal, Nirangkush Das, Justin A. Echols, Dr. Rakesh Joshi, Shi Lu, Kaustav Mondal, Pragyan Pradhan, Nikhilesh Ravishankar, Venkatraman Renganathan, Aratrik Sarkar, Dr. Ashfaque B. Shafique, Aniket Shirsat, Shubham Sonawani, Dr. Srikanth Sridharan, and at Intel - Kaushik Balasubramanian, Daniel G. Cartagena and Ryan D. Saffores.

Finally, I am grateful to my family and friends for their love and encouragement, esp. Adithya, Akhilesh, Alex, Amith, Arindam, Darshan, Deepak, Ganesh, Ginni, Harshith, Hemanth, Hitha, Manvir, Navaneeth, Pradeep, Rajesh, Ravi Sagar, Sagar, Sandeep, Shwetha, Skanda, Sonali, Thanuchith, Vimarsh, Vinayaka and many more.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Motivation

It is well understood that, in general, a good multivariable control system design must possess acceptable properties at distinct loop-breaking points [1–15]; e.g. the plant output and the plant input. Achieving acceptable properties includes satisfying and trading off nominal performance specifications as well as robustness specifications at these loop-breaking points. It is also well known that tradeoffs can be particularly taxing when the plant is ill-conditioned (having high condition number [3–9] and/or relative gain array (RGA) entries [16–21] and/or scaled condition number [15, 22]). While much insight has been obtained, there still remains a need for a methodology that can be used to systematically address the associated tradeoffs. It is natural, for example, to ask for a methodology that can be used to "equilibrate" (to the extant possible) properties at distinct loop breaking points as well as to manage the associated (possibly very difficult) tradeoffs. Moreover, we also want the methodology to handle a large set of control objectives and constraints (e.g. peak weighted frequency response, output overshoot/peak controls to reference (time-domain) commands, etc.) without undue computational hardship - leveraging off (for example) the state-of-the-art fast interior point convex optimization solvers [23–32] which can handle non-differentiable objective/constraint functions. Further, hierarchical control architecture [20, 33, 34] can exploit the additional feedback information to significantly help in making reasonable tradeoffs between properties at the outputs/errors and at the inputs/controls. Given the above, it is natural to seek a methodology that

can be used to systematically design hierarchical inner-outer loop control systems to simultaneously meet design specifications at distinct loop breaking points (e.g. outputs/errors, inputs/controls). Such a methodology, in principle, can then be used to reveal fundamental tradeoffs and systematically address difficult tradeoffs. Our work is motivated by the above fundamental control-relevant design issues.

1.2 State-of-the-Field, Related Work and Challenges

Much has been done in the field of multiobjective optimization for control design in the past three decades for addressing problems involving multiple performance/robustness specifications. Mixed $\mathcal{H}^2/\mathcal{H}^\infty$ optimization [35–37] has been used to address frequency- and time-domain specifications that can be conflicting. Many algorithms/approaches based on convex optimization and Linear Matrix Inequalities (LMIs) have also been investigated [38–54] for multiobjective control design. Within [55–58], the authors have exploited these ideas to develop methods that directly accommodate specifications at distinct loop breaking points (e.g. outputs/errors, inputs/controls).

Within [58], the author proposed what is called a "generalized" weighted $\mathcal{H}^\infty$ mixed-sensitivity optimization subject to convex constraints. This was achieved by exploiting convex optimization ideas from [40, 59–61] to formulate an optimization that directly addresses closed loop maps associated with distinct loop breaking points (e.g. outputs/errors, inputs/controls). Such a methodology is particularly beneficial for multivariable plants that are ill-conditioned (having high condition number and/or RGA entries and/or scaled condition number) [3–10, 16–21]. More specifically, the generalized mixed-sensitivity methodology presented within [55–58], provides a tool for directly addressing tradeoffs and "equilibration" issues associated with distinct loop breaking points. Here, the term "equilibration" refers to achieving similar properties (e.g. peak sensitivities) at distinct loop breaking points. Within [56],

the authors demonstrated the utility of this approach for directly shaping properties at the inputs/controls and at the outputs/errors. Near-optimal finite-dimensional "equilibrated" controllers were designed for stable infinite-dimensional plants in [56]. Within [57], the authors extended the ideas within [56] - presenting and examining three different approaches to formulating objective functions within the generalized mixed-sensitivity control design methodology.

Within [62, 63], the authors presented a weighted mixed-sensitivity $\mathcal{H}^\infty$ approach that weighs responses from plant input disturbances d_i as well as the (traditionally weighted responses from) reference commands r. This approach directly penalizes d_i-induced responses in order to try to directly influence properties at the plant input.

Within [55], the authors presented a methodology for doing hierarchical inner-outer control design using generalized $\mathcal{H}^\infty$ mixed sensitivity. The objective function in [62, 63] uses closed-loop maps that are different from the closed loop maps that one would want to address. This is a consequence of introducing a fictitious plant output and reference input in order to generate an inner-outer loop control system. In [55], closed loop maps of interest were used in the objective function using "selection matrix". For the hierarchical inner-outer loop control structure, it is observed that three sets of closed loop properties are obtained by breaking the loop at the error/output (e), the control/input (c), and at inner-loop sensor noise (n_i) respectively. In [55], closed loop properties at e and c were shaped directly, whereas the properties at n_i were adjusted after the optimization process using roll-offs in inner-loop controller. The amount of roll-off was varied to tradeoff between retaining acceptable properties at e and c, while achieving required bandwidth specifications at n_i.

Since the scramjet-powered Mach 7/10 flights of X-43A in 2004 [64–66], the research on hypersonic vehicles has seen a resurgence [62, 63, 67–85]. With the recent successful X-51A flight test (May, 2013), the hypersonic application considered here

is timely. Air-breathing hypersonic propulsion is viewed as the next critical step toward achieving (1) reliable, affordable, routine access to space, as well as (2) global reach vehicles. There are commercial and as military implications to both objectives. Rocket-based (combined cycle) propulsion systems are much more expensive to operate because they must carry oxygen. This is particularly costly when traveling at lower altitudes through the troposphere (i.e. below 36,152 ft). They do not exhibit the desired levels of reliability and flexibility (e.g. airplane like takeoff and landing options) either. As a result, much emphasis has been placed on two-stage-to-orbit (TSTO) designs that involve a turbo-ram-scramjet combined cycle first stage and a rocket second stage. This research focuses on control challenges associated with scramjet-powered hypersonic vehicles. Such vehicles are characterized by significant aero-thermo-elastic-propulsion interactions and uncertainty. Hierarchical inner-outer loop controllers have been used controlling such vehicles. In this work, we consider the control design for the longitudinal dynamics of a scramjet-powered hypersonic vehicle flying at Mach 8, 85 kft. The control system will seek good properties at input/control, output/error as well as at the inner-loop sensor noise.

The methodology presented in [55–58] was used to design hierarchical inner-outer controllers for a LCL-filter [86–88] based grid-connected inverter [89]. Three-phase (3-ϕ) voltage source inverters are widely used as an interface to inject power from various distributed power generation systems (DPGSs) like wind and solar [90, 91] into the utility grid. Inner-outer control structure has been widely used for such inverters [92]. Within [89], the authors used the Generalized Mixed Sensitivity methodolgy to address closed loop properties at distinct loop-breaking point associated with this inner-outer control structure. It was show how an inner-outer can be systematically designed using this methodology. Further extensions to the methodology can be made to address relevant issues and tradeoffs in control of multiple ground vehicle robots

toward the ambitious long-term goal of a fleet of cooperating Flexible Autonomous Machines operating in an uncertain Environment (FAME [93]) as discussed in the some recent papers [94, 95] by the author and the related theses [96–103]. Several ongoing projects related to these topics were presented in [104–106]. Connectivity between subsystems can be supported by advanced approaches as described in [107–112]. Further, the control challenges associated with a Hawkmoth Flapping Wing Micro Air Vehicle (MAV) [113–115] can be systematically addressed using Generalized Mixed Sensitivity. Ongoing work in this direction was presented in [116]. Finally, control designs for two challenging application areas, namely dynamic thermal management controllers for multi-core processors [117–119] and convex formulations for modeling and predictive control (MPC) of nonlinear hybrid systems [120–124], that were addressed in papers [125, 126] were contributed by the author.

1.3 Contributions

This work addresses and provides answers to the following fundamental control design questions/problems:

1. How can we develop a multivariable control system design methodology that simultaneously accommodates typical control objectives and constraints involving several loop breaking points? Such a methodology can be particularly useful for plants that are ill-conditioned (having high condition number and/or RGA entries and/or scaled condition number) [3–10, 16–21].

2. How the closed loop properties associated with several loop breaking points in the hierarchical inner-outer loop control architecture can be systematically shaped using the generalized $\mathcal{H}^\infty$ mixed sensitivity (GMS) hierarchical methodology?

3. How can we use such methodology to design controllers for and tradeoff between several control properties of longitudinal dynamics of 3-DOF Hypersonic vehicle model - one that is unstable, non-minimum phase and possesses significant coupling between channels?

In this work, we address the above fundamental control design questions/problems. The main contributions of this work are listed below.

1. We present a $\mathcal{H}^\infty$-based generalized mixed-sensitivity control design methodology (GMS) for multivariable Linear Time Invariant (LTI) systems to simultaneously address specifications at distinct loop breaking points [55–58] e.g. outputs/errors, inputs/controls. The term "generalized" is intended to capture this.

2. The convex optimization based methodology presented can be used to generate what we call *equilibrated* designs. Such designs possess similar properties (e.g. peak sensitivities) at the inputs/controls and at the outputs/errors. As such, equilibrated designs and the presented methodology can be used to directly address fundamental input-output tradeoffs [55–58, 89]. As such, this powerful convex optimization based tool can be used to understand control design limitations and trade-offs.

3. We show how GMS can be used to address broad class of convex frequency-domain (e.g. $\mathcal{H}^\infty$) and time-domain (e.g. $\mathcal{L}^\infty$) design specifications.

4. We analyze the tradeoffs involved in shaping closed loop properties at distinct multivariable loop-breaking points and demonstrate the existence of pareto optimal points associated with them. Further, we study the limitations and tradeoffs associated with shaping the properties in the presence of right half plane

poles/zeros, limited available bandwidth and peak time-domain constraints.

5. Three state-of-the-art subgradient-based non-differentiable constrained (frequency- and time-domain) convex optimization solvers, namely Analytic Center Cutting Plane Method (ACCPM), Kelley's CPM and SolvOpt are utilized to solve the convex probelm and comparisons are made.

6. Basis parameters that result in efficient approximation (using lesser number of basis terms) of the infinite-dimensional Youla et al. parameter are studied.

7. The utility of the methodology is illustrated by designing controllers for longitudinal dynamics of a 3-DOF scramjet-powered hypersonic vehicle model - one that is unstable, non-minimum phase and has high coupling between channels [62, 63, 67–82]. For such a challenging model, the methodology is shown to generate very good designs - designs that would be very difficult to obtain without the methodology presented. Comparisons to other methods are made to further illustrate the power and transparency of the methodology. The methodology presented is shown to generate good designs much faster than that obtained by other methods; i.e. it is much more transparent than other methods. Since the scramjet-powered Mach 7/10 flights of X-43A in 2004 [64–66], the research on hypersonic vehicles has seen a resurgence [62, 63, 67–85]. With the recent successful X-51A flight test (May, 2013), the hypersonic application considered here is timely.

8. While the focus of the dissertation is on finite-dimensional LTI multivariable plants, the work presented here can also be applied to infinite-dimensional plants subject to mixed time- and frequency-domain specifications. Ongoing work in this direction was presented by the author in [56, 57].

In short, this work provides a systematic approach to a large class of important control system design problems.

1.4 Organization of Dissertation

The remainder of the dissertation is organized as follows. In Chapter 2, we provide the mathematical preliminaries and establish notations that are be used throughout the dissertation. In Chapter 3, we analyze the relevant control design challenges and tradeoffs associated with shaping of closed loop properties for Single-Input Single-Output (SISO) as well as Multiple-Input Multiple-Output (MIMO) systems in the presence of inherent control difficulties instability, non-minimum phase, algebraic and analytic constraints, etc. In Chapter 4, we describe the proposed generalized mixed-sensitivity methodology used to systematically design hierarchical inner-outer loop controllers that addresses specifications at several loop-breaking points and discuss several extensions that can be made to address a broad class of design requirements/specifications. In Chapter 5, we discuss

1. the technical approach taken to convexify the problem
2. the technical approach taken to convert the (strictly speaking) infinite-dimensional problem to an approximated finite-dimensional one
3. the implementation and comparisons of the algorithms used to solve them

Chapter 6 contains illustrative control design examples problems that are solved using GMS, and demonstrate

1. the existence of pareto optimality in obtaining properties at error/output & control/input
2. how we can pose $\mathcal{L}^\infty$ time-domain specifications as convex constraints

3. solving some SISO problems that discuss some specific control challenges and tradeoffs

Chapter 7 presents multivariable control designs for the longitudinal dynamics of a scramjet-powered hypersonic vehicle using generalized mixed sensitivity, along with comparision several other control designs/methods. Finally, Chapter 8 summarizes the work presented in the dissertation and provides directions for future research.

Chapter 2

MATHEMATICAL PRELIMINARIES

2.1 Overview

In this chapter, we present some well known mathematical definitions and results that are relevant to our work and establish notations that are used throughout the dissertation.

2.2 Notations and Definitions

Throughout this dissertation, we use the following standard notations:

- $\mathcal{C}$, $\mathcal{R}$, and $\mathcal{Z}$: Set of complex, real, and integer numbers respectively;
- $\mathcal{R}_+ \stackrel{\text{def}}{=} \{\, t \in \mathcal{R} \mid t \geq 0 \,\}$: Non-negative real numbers;
- $\mathcal{R}_- \stackrel{\text{def}}{=} \{\, t \in \mathcal{R} \mid t < 0 \,\}$: Negative real numbers;
- $\mathcal{C}_+ \stackrel{\text{def}}{=} \{\, s \in \mathcal{C} \mid Re\ s > 0 \,\}$: Open Right Half complex Plane (ORHP);
- $\mathcal{F}$: General field designation used to represent $\mathcal{R}$ and $\mathcal{C}$ when both are applicable;
- $j\mathcal{R} \stackrel{\text{def}}{=} \{\, s \in \mathcal{C} \mid Re\ s = 0 \,\}$: Imaginary axis;
- $\mathcal{H}^\infty \stackrel{\text{def}}{=} \mathcal{H}^\infty(\mathcal{C}_+)$: Hardy space of scalar or matrix valued functions which are analytic and essentially bounded in $\mathcal{C}_+$;
- $\mathcal{H}^2 \stackrel{\text{def}}{=} \mathcal{H}^2(\mathcal{C}_+)$: Hardy space of scalar or matrix valued functions which are Laplace transforms of functions in $\mathcal{L}^2(\mathcal{R}_+)$;

- $\mathcal{L}^\infty \stackrel{\text{def}}{=} \mathcal{L}^\infty(j\mathcal{R})$: Lebesgue space of measurable essentially bounded scalar or matrix valued functions with support on $j\mathcal{R}$;
- $\mathcal{RH}^\infty$: Subspace of $\mathcal{H}^\infty$ consisting of real-rational scalar or matrix valued functions;
- $|(.)|$: Magnitude of the complex quantity $(.)$;
- $\angle(.)$: Phase angle of the complex quantity $(.)$;
- $\|g\|_{(.)}$: Norm of a function g belonging to the space $(.)$;

2.3 Signal and System Norms

The signal and system norms relevant to the work in the dissertation are now defined.

2.3.1 Signal Norms

$\mathcal{L}^p$ Norms - Functions on the Real Line. Let $g : \mathcal{R} \to \mathcal{F}$ denote a scalar signal defined on the real axis. :

$$\|g\|_{\mathcal{L}^1} \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} |g(t)|\, dt \tag{2.1}$$

$$\|g\|_{\mathcal{L}^2} \stackrel{\text{def}}{=} \sqrt{\int_{-\infty}^{\infty} |g(t)|^2\, dt} \tag{2.2}$$

$$\|g\|_{\mathcal{L}^p} \stackrel{\text{def}}{=} \left[\int_{-\infty}^{\infty} |g(t)|^p\, dt\right]^{\frac{1}{p}} \quad p = 1, 2, \ldots \tag{2.3}$$

$$\|g\|_{\mathcal{L}^\infty} \stackrel{\text{def}}{=} ess \sup_{t\in\mathcal{R}} |g(t)| \tag{2.4}$$

Here, $\int$ denotes the *Lebesgue integral* [127–129]. Each of the above norms are readily extended to vectors of signals by replacing $|\cdot|$ with a vector norm. The vector norm selected is typically the associated vector norm. Let $g : \mathcal{R} \to \mathcal{F}^n$ denote a vector

valued signal. Given this, we have the following norm extensions:

$$\|g\|_{\mathcal{L}^1} \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} \|g(t)\|_1 \; dt \tag{2.5}$$

$$\|g\|_{\mathcal{L}^2} \stackrel{\text{def}}{=} \sqrt{\int_{-\infty}^{\infty} \|g(t)\|_2^2 \; dt} \tag{2.6}$$

$$\|g\|_{\mathcal{L}^p} \stackrel{\text{def}}{=} \left[\int_{-\infty}^{\infty} \|g(t)\|_p^p \; dt\right]^{\frac{1}{p}} \tag{2.7}$$

$$\|g\|_{\mathcal{L}^\infty} \stackrel{\text{def}}{=} ess\sup_{t\in\mathcal{R}} \|g(t)\|_\infty \tag{2.8}$$

where $\|x\|_p \stackrel{\text{def}}{=} (\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}}$ $(p = 1, 2, \dots)$ and $\|x\|_\infty \stackrel{\text{def}}{=} max_i \; |x_i|$ for an n-dimensional vector $x = [\, x_1 \; \cdots \; x_n \,]^T$.

2.3.2 System Norms

$\mathcal{L}^p$ Norms - Functions on the Imaginary Axis. Let $G : \mathcal{C} \longrightarrow \mathcal{C}$ denote the transfer function of a single-input single-output (SISO) LTI system. The following are typical system norms that arise in engineering applications.

$$\|G\|_{\mathcal{L}^2} \stackrel{\text{def}}{=} \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty} |G(j\omega)|^2 \; d\omega} \tag{2.9}$$

$$\|G\|_{\mathcal{L}^\infty} \stackrel{\text{def}}{=} ess\sup_{\omega\in\mathcal{R}} |G(j\omega)| \tag{2.10}$$

Each of the above system norms can be extended to a multiple-input multiple-output (MIMO) transfer function matrix $G : \mathcal{C} \longrightarrow \mathcal{C}^{q\times n}$. Doing so yields:

$$\|G\|_{\mathcal{L}^2} \stackrel{\text{def}}{=} \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty} trace\,[\, G^H(j\omega)G(j\omega)\,] \; d\omega} \tag{2.11}$$

$$\|G\|_{\mathcal{L}^\infty} \stackrel{\text{def}}{=} ess\sup_{\omega\in\mathcal{R}} \bar{\sigma}\,[\, G(j\omega)]\,. \tag{2.12}$$

$\bar{\sigma}(M)$ denotes the maximum singular value of the matrix M. See Section 2.4 for a brief discussion on singular values of a matrix.

$\mathcal{H}^p$ Norms - Functions Over the Open Right Half Plane.

$$\|G\|_{\mathcal{H}^2} \stackrel{\text{def}}{=} \sup_{\sigma>0} \sqrt{\frac{1}{2\pi}\int_{\sigma-j\infty}^{\sigma+j\infty} |G(\sigma+j\omega)|^2\, d\omega} \tag{2.13}$$

$$\|G\|_{\mathcal{H}^p} \stackrel{\text{def}}{=} \sup_{\sigma>0} \left[\frac{1}{2\pi}\int_{\sigma-j\infty}^{\sigma+j\infty} |G(\sigma+j\omega)|^p\, d\omega\right]^{\frac{1}{p}} \tag{2.14}$$

$$\|G\|_{\mathcal{H}^\infty} \stackrel{\text{def}}{=} \sup_{Res>0} |G(s)|. \tag{2.15}$$

Here, $\sigma = Re(s)$ and $\omega = Im(s)$. MIMO transfer function matrix $G : \mathcal{C}_+ \longrightarrow \mathcal{C}^{q\times m}$.

$$\|G\|_{\mathcal{H}^2} \stackrel{\text{def}}{=} \sup_{\sigma>0} \sqrt{\frac{1}{2\pi}\int_{\sigma-j\infty}^{\sigma+j\infty} trace\,[\, G^H(\sigma+j\omega)G(\sigma+j\omega)\,]\; d\omega} \tag{2.16}$$

$$\|G\|_{\mathcal{H}^p} \stackrel{\text{def}}{=} \sup_{\sigma>0} \left[\frac{1}{2\pi}\int_{\sigma-j\infty}^{\sigma+j\infty} \left[\, trace\{\, G^H(\sigma+j\omega)G(\sigma+j\omega)\,\}\,\right]^p\; d\omega\right]^{\frac{1}{p}} \tag{2.17}$$

$$\|G\|_{\mathcal{H}^\infty} \stackrel{\text{def}}{=} \sup_{Res>0} \bar{\sigma}\,[\, G(s)\,] \tag{2.18}$$

where $trace(M)$ denotes the trace of the matrix M.

2.4 Singular Values of a Matrix

2.4.1 Singular Value Decomposition (SVD)

Theorem 2.4.1 (Singular Value Decomposition of a Matrix)

Let $M \in \mathcal{C}^{m\times n}$. There exists unitary matrices $U \in \mathcal{C}^{m\times m}$, $V \in \mathcal{C}^{n\times n}$, and a block diagonal matrix $\Sigma \in \mathcal{R}^{m\times n}$ such that

$$M = U\Sigma V^H \tag{2.19}$$

where

$$\Sigma = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} \tag{2.20}$$

and $S = diag(\sigma_1, \sigma_2, \ldots, \sigma_r)$ *where* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ *are the singular values of* M *and* r *is the rank of* M *[130, 131]. The above decomposition is referred to as the singular value decomposition for the matrix* M*. The vectors* v_i *are called the right singular vectors, and the vectors* u_i *are called the left singular vectors.*

Proof: See [130, 131]. ∎

2.4.2 Condition Number (κ)

Definition 2.4.1 (Condition Number (κ)**)** *The condition number of a matrix* M *is defined by the ratio*

$$\kappa(M) \stackrel{\text{def}}{=} \frac{\bar{\sigma}(M)}{\underline{\sigma}(M)} \tag{2.21}$$

where $\bar{\sigma}(M)$ *is the maximum singular value, and* $\underline{\sigma}(M)$ *is the minimum singular value of the matrix* M*.* ∎

Definition 2.4.1 is used in Chapter 3 to establish some relations between properties at distinct loop-breaking points that show the difficulties in control of ill-conditioned plants.

2.5 Elements of Convex Optimization

In this section, we provide the definitions of convex set and convex function, and present some properties of convex functions that are relevant to our work within the dissertation.

2.5.1 Convex Set and Convex Function

Definition 2.5.1 (Convex Set) *A set S in a vector space (over real numbers) X is said to be convex if for every $x_1, x_2 \in S$, and for every $t \in [0,1]$,*

$$tx_1 + (1-t)x_2 \in S \tag{2.22}$$

■

Definition 2.5.2 (Convex function) *A real valued function $f : S \to \mathcal{R}$ (where S is a convex set in a vector space X) is said to be a convex function if, for every $x_1, x_2 \in S$, and every $t \in [0,1]$,*

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2) \tag{2.23}$$

■

A comprehensive treatment of convex sets and functions can be found in the book [60] by Boyd et al. Next, we present some very relevant properties of convex functions that can be used to leverage off for establishing convexity results on multiobjective convex optimization problems presented in Chapter 4.

2.5.2 Convexity Preserving Operations.

A convexity preserving operation produces a new convex function out of a set of "atom" functions that are already known to be convex [60, 132]. Here, we present two such functions that are relevant to our work. They are,

1. Pointwise Weighted Maximum of two "atom" functions (f_1 and f_2).

2. Weighted Sum of two "atom" functions (f_1 and f_2).

A larger set of operations that preserve convexity can be found in the book [60] by Boyd et al.

Lemma 2.5.1 (Pointwise Weighted Maximum) *If f_1 and f_2 are convex functions, and if $\mu_1 \in \mathcal{R}_+$, $\mu_2 \in \mathcal{R}_+$, then the pointwise weighted maximum f, defined by*

$$f(x) = \max\{\mu_1 f_1(x), \mu_2 f_2(x)\} \tag{2.24}$$

with $domf = domf_1 \cap domf_2$ is also convex. ∎

This property can be extended to an arbitrary number of convex functions. Further, it can be extended to the pointwise weighted supremum over an arbitrary number of convex functions.

Lemma 2.5.2 (Weighted Sum) *If f_1 and f_2 are convex functions, and if $\mu_1 \in \mathcal{R}_+$, $\mu_2 \in \mathcal{R}_+$, then the weighted sum f, defined by*

$$f(x) = \mu_1 f_1(x) + \mu_2 f_2(x) \tag{2.25}$$

with $domf = domf_1 \cap domf_2$ is also convex. ∎

This property can be extended to an arbitrary number of convex functions. These two properties of convex functions presented in Lemmas 2.5.1 and 2.5.2 can be used to show the convexity properties of multiobjective functions presented in Section 4.5.

2.6 Summary and Conclusions

In this chapter, we presented some well known mathematical results that are relevant to our work, and established notations that are used throughout the dissertation. We also provided definitions involving matrices (e.g. Definition 2.4.1) that are used in Chapter 3 to establish some relations between properties at distinct loop-breaking points that show the difficulties in control of ill-conditioned plants. Further, we discussed relevant convex optimization results that are leveraged off in our work. Specifically, the multiobjective (multipe convex objectives) optimization problems considered in Chapter 4 can be shown to be convex using these results.

Chapter 3

RELEVANT DESIGN CHALLENGES AND TRADEOFFS IN SISO AND MIMO CONTROL SYSTEMS

3.1 Overview

In this chapter, we analyze the design challenges and tradeoffs associated with control of SISO and MIMO systems relevant to the work in the dissertation. We define open and closed loop transfer function matrices (TFMs) for a generic MIMO system using standard P-K feedback structure. These closed loop maps defined here are shaped using our proposed GMS methodology in Chapters 6 and 7. Several design difficulties (e.g. dependence of sensitivities at output versus input on plant condition number) and limitations (e.g., Bode's sensitivity integral constraint, sensitivity limatations due to Right Half Plant (RHP) poles/zeros, etc.) are analyzed in detail to obtain insight into achievable control properties. These help in posing the control problem (e.g., selecting weighting functions) using GMS.

3.2 Open and Closed Loop Transfer Function Matrices (TFMs)

Consider a standard feedback control system as shown in Figure 3.1. It is assumed that the plant (P) and controller (K) are MIMO LTI systems (i.e. P and K are transfer function matrices).

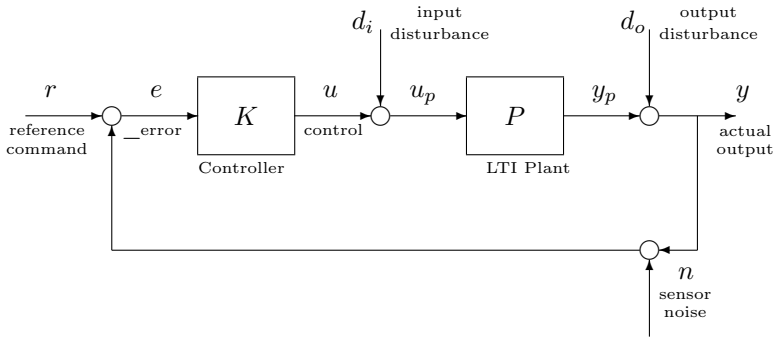


Figure 3.1: Visualization of Standard P-K Negative Feedback Structure

The closed and open loop transfer function matrices [133] when the feedback loop is broken at (1) plant output/error defined as in Equations (3.1) - (3.4) and represented by subscript e, and (2) plant input/control defined as in Equations (3.5) - (3.8) and represented by subscript c are shown below:

- Open loop TFM at Plant Output/Error

$$L_e \stackrel{\text{def}}{=} PK \tag{3.1}$$

- *Sensitivity at Plant Output/Error*

$$S_e \stackrel{\text{def}}{=} [I + PK]^{-1} = T_{re}(\text{unfiltered}) \tag{3.2}$$

- *Control Sensitivity at Plant Output/Error*

$$KS_e \stackrel{\text{def}}{=} K[I + PK]^{-1} = [I + KP]^{-1}K = S_cK = T_{ru}(\text{unfiltered}) \tag{3.3}$$

- *Complementary Sensitivity at Plant Output/Error*

$$T_e \stackrel{\text{def}}{=} I - S_e = PK[I + PK]^{-1} = T_{ry}(\text{unfiltered}) \tag{3.4}$$

- Open loop TFM at Plant Input/Control

$$L_c \stackrel{\text{def}}{=} KP \tag{3.5}$$

- *Sensitivity at Plant Input/Control*

$$S_c \stackrel{\text{def}}{=} [I + KP]^{-1} = T_{d_i u_p} \tag{3.6}$$

- *Disturbance Sensitivity at Plant Input/Control*

$$PS_c \stackrel{\text{def}}{=} P[I + KP]^{-1} = [I + PK]^{-1}P = S_e P = T_{d_i y} \tag{3.7}$$

- *Complementary Sensitivity at Plant Input/Control*

$$T_c \stackrel{\text{def}}{=} I - S_c = [I + KP]^{-1}KP = T_{d_i u} \tag{3.8}$$

3.3 Plant Condition Number Dependent Relations Between Sensitivities at Distinct Loop-Breaking Points

It is well understood that in general, a good multivariable control system design must possess acceptable properties at distinct loop-breaking points. The tradeoffs that exist can be particularly taxing when the plant is ill-conditioned. Freudenberg et al. [3, 10] obtained bounds on singular values of sensitivity function at one loop-breaking point in terms of those at the other. It is assumed that the plant is square and that both plant and controller transfer function matrices are non-singular at frequencies of interest.

Theorem 3.3.1 *Let $det(P) \neq 0$ and $det(K) \neq 0$ at frequencies of interest. Then at those frequencies each singular value of the sensitivity functions (S_e and S_c) satisfy the bounds*

$$\frac{1}{\kappa(P)}\sigma_i(S_c) \leq \sigma_i(S_e) \leq \kappa(P)\sigma_i(S_c) \tag{3.9}$$

$$\frac{1}{\kappa(P)}\sigma_i(S_e) \leq \sigma_i(S_c) \leq \kappa(P)\sigma_i(S_e) \tag{3.10}$$

Proof: The proof follows from the similarity transformation identities

$$S_c = P^{-1} S_e P \tag{3.11}$$

$$S_e = P S_c P^{-1} \tag{3.12}$$

and the minimax property of singular values [131] ∎

We see that, as expected, the sensitivities at distinct loop-breaking point can be very different if $\kappa(P) >> 1$, i.e. obtaining good properties at output/error does not a priori imply the same at input/control. Similar inequalities that depend on condition number of the controller K can be obtained as listed below.

$$\frac{1}{\kappa(K)}\sigma_i(S_c) \leq \sigma_i(S_e) \leq \kappa(K)\sigma_i(S_c) \tag{3.13}$$

$$\frac{1}{\kappa(K)}\sigma_i(S_e) \leq \sigma_i(S_c) \leq \kappa(K)\sigma_i(S_e) \tag{3.14}$$

Note that if $\kappa(K) = 1$ or $K = P^{-1}g(s)$ where $g(s)$ is a scalar transfer function, then by definition $L_e = L_c$, and from the above relations, $S_e = S_c$. Both these strategies are deficient in that they unnecessarily restrict the class of achievable feedback properties. This is especially true if we have different specifications (e.g. different closed loop bandwidths) at distinct loop-breaking points or the plant is non-minimum phase. Further, it is important to note that analogous relations to the above inequalities can be obtained on complementary sensitivity functions (T_e and T_c) based on similar ideas. These ideas are exploited in Chapter 6 to show how multiobjective optimization can be used to address simultaneously addressing sensitivities at distinct loop-breaking points.

3.4 Relations between Open and Closed Loop TFMs

In this section we list some important relations between open and closed loop TFMs. These can be readily obtain from definitions and by using triangle enequality property of norms.

$$|1 - \underline{\sigma}(L)| \leq \frac{1}{\bar{\sigma}(S)} \leq \underline{\sigma}(L) + 1 \tag{3.15}$$

$$\frac{1}{\underline{\sigma}(L) + 1} \leq \bar{\sigma}(S) \leq \frac{1}{|\underline{\sigma}(L) - 1|} \tag{3.16}$$

$$\frac{1}{|\bar{\sigma}(L) - 1|} \leq \underline{\sigma}(S) \leq \frac{1}{\bar{\sigma}(L) + 1} \tag{3.17}$$

$$\frac{1}{\underline{\sigma}(L) + 1} - 1 \leq \bar{\sigma}(T) \leq \frac{1}{|\underline{\sigma}(L) - 1|} + 1 \tag{3.18}$$

where $\underline{\sigma}(M) > 0$ is the minimum singular value of matrix M.

Equality of Geometric Means of Singular Values.

Theorem 3.4.1 *Let P & K both be $n \times n$ square matrices and let* $\det(L_e) \neq 0$ *&* $\det(L_c) \neq 0$ *at frequencies of interest. Then at all those frequencies*

$$\prod_{i=1}^{n} \sigma_i(L_e) = \prod_{i=1}^{n} \sigma_i(L_c) \tag{3.19}$$

Proof: Using the matrix product property of determinants [134] we have $\det(L_e) = \det(PK) = \det(P)\det(K) = \det(KP) = \det(L_c)$. *Further if SVD of L_e is such that $L_e = U\Sigma V^H$, then* $\det(L_e) = \det(U\Sigma V^H) = \det(U)\ \det(\Sigma)\ \det(V)$. *But since U and V are unitary and Σ is diagnoal, we have* $|\det(L_e)| = |\det(\Sigma)| = \prod_{i=1}^{n} \sigma_i(L_e)$. *The required relation follows from the above two identities and extending them to L_c.* ∎

Analogous results can be derived on closed loop sensitivity and complementary sensitivity TFMs. From the above relations it follows that

$$\bar{\sigma}\left(L_e\right) \quad \geq \quad \underline{\sigma}\left(L_c\right) \tag{3.20}$$

$$\bar{\sigma}\left(L_c\right) \quad \geq \quad \underline{\sigma}\left(L_e\right) \tag{3.21}$$

Again, analogous results can be derived on closed loop sensitivity and complementary sensitivity TFMs.

3.5 Measures of Interactions in MIMO Systems

3.5.1 Condition Number (κ)

Definition 3.5.1 (Condition Number (κ)) *The condition number of a matrix A is defined by the ratio*

$$\kappa(A) \stackrel{\text{def}}{=} \frac{\bar{\sigma}(A)}{\underline{\sigma}(A)} \tag{3.22}$$

where $\bar{\sigma}(A)$ is the maximum singular value, and $\underline{\sigma}(A)$ is the minimum singular value of A. ∎

Comment 3.5.1 *For a nonsingular square matrix A, $\underline{\sigma}(A) = \frac{1}{\bar{\sigma}(A^{-1})}$, so*

$$\kappa(A) = \frac{\bar{\sigma}(A)}{\bar{\sigma}(A^{-1})} \tag{3.23}$$

∎

3.5.2 Relative Gain Array (RGA)

Definition 3.5.2 (Relative Gain Array (RGA)) *The Relative Gain Array (RGA) of a nonsingular square matrix A is a square matrix defined as [20]*

$$RGA(A) = \Lambda(A) \stackrel{\text{def}}{=} A \circ \left(A^{-1}\right)^T \tag{3.24}$$

where " $\circ$ " denotes element-wise matrix multiplication (Hadamard or Schur product of matrices) ∎

3.5.3 Scaled Condition Number ($\mathcal{K}^*$)

Definition 3.5.3 (Scaled Condition Number (κ^*)) *The Scaled Condition Number of a nonsingular square matrix A is defined as*

$$\kappa^*(A) \stackrel{\text{def}}{=} \min_{D_1, D_2} \kappa\left(D_1 A D_2\right) \tag{3.25}$$

where D_1 and D_2 are diagonal scaling matrices. ∎

Theorem 3.5.1 *For a 2×2 matrix A, the scaled condition number is given by,*

$$\kappa^*(A) = \|\Lambda\|_{i1} + \sqrt{\|\Lambda\|_{i1} - 1} \tag{3.26}$$

where $\|\Lambda\|_{i1}$ is the induced 1-norm (maximum column sum) of $\Lambda(A)$

Proof: See [135]. ∎

Definition 3.5.4 (Structured Singular Value of a Matrix)

Consider the uncertainty set consisting of diagonally organized scalar and matrix blocks:

$$\mathbf{\Delta} \stackrel{\text{def}}{=} \{\ diag\left[\ \delta_i I_{r_1}, \ldots, \delta_s I_{r_S}, \Delta_1, \ldots, \Delta_F\ \right] \in \mathcal{C}^{n \times n} \ \mid\ \delta_1 \in \mathcal{C}, \quad \Delta_j \in \mathcal{C}^{m_j \times m_j}\ \} \tag{3.27}$$

where we have

$$\sum_{i=1}^{S} r_i + \sum_{j=1}^{F} m_j = n \tag{3.28}$$

for dimensional consistency. We say that $\mathbf{\Delta}$ *is a structured uncertainty set that defines the set of* admissible perturbations.

The structured singular value *of the matrix* $M \in \mathcal{C}^{n\times n}$ *with respect to the uncertainty set* $\mathbf{\Delta}$ *is defined as follows [15, 136]:*

$$\mu_{\mathbf{\Delta}}(M) \stackrel{\text{def}}{=} \frac{1}{\min\{\, \bar{\sigma}(\Delta) \;\mid\; \Delta \in \mathbf{\Delta},\ \det(I - M\Delta) = 0 \,\}} \tag{3.29}$$

where $\mu_{\mathbf{\Delta}}(M) \stackrel{\text{def}}{=} 0$ *if no admissible perturbation makes* $I - M\Delta$ *singular.* ∎

We note that $\mu_{\mathbf{\Delta}}$ is simply the reciprocal of the size of the smallest admissible perturbation that makes $I - M\Delta$ singular. For $S = 0$ and $F = 1$, we have $\mu_{\mathbf{\Delta}}(M) = \bar{\sigma}(M)$.

3.6 Gain and Phase of MIMO Systems

Consider a transfer function matrix $L(s)$. At a fixed point $s \in \mathcal{C}$, let its singular value decomposition (SVD) be given by

$$L(s) = \sum_{1}^{n} \sigma_i(L(s)) v_i(L(s)) u_i^H(L(s)) \tag{3.30}$$

where the singular values are ordered so that $\sigma_1(L(s)) \geq \sigma_2(L(s)) \geq \cdots \geq \sigma_n(L(s))$.

Gain: The gains of $L(s)$ are defined to be its singular values $\sigma_i(L(s))$.

Phase: The phase of $L(s)$ corresponding to $\sigma_i(L(s))$ is defined [137] to be the phase difference between the corresponding left and right singular values in reference to some basis vectors w_i

$$\theta_i(L(s)) \stackrel{\text{def}}{=} \angle w_i^H v_i(L(s)) - \angle w_i^H u_i(L(s)) \tag{3.31}$$

where w_i $(i = 1, 2, \ldots, n)$ are the basis vectors of a given orthonormal basis of $\mathcal{C}$. Selecting $w_i = u_i(L(s))$, we get

$$\theta_i(L(s)) = \angle u_i^H(L(s)) v_i(L(s)) \tag{3.32}$$

Example 3.6.1 *Consider a* 2×2 *TFM as shown below*

$$L(s) = \begin{bmatrix} \frac{1}{s+1} & 0 \\ 0 & \frac{1}{s+2} \end{bmatrix} \begin{bmatrix} 9 & -10 \\ -8 & 9 \end{bmatrix} \tag{3.33}$$

The phase of $L(s)$ *along* $s = j\omega$ *for reference vector* $w_i = u_i$ *is shown by Fig. 3.2.*

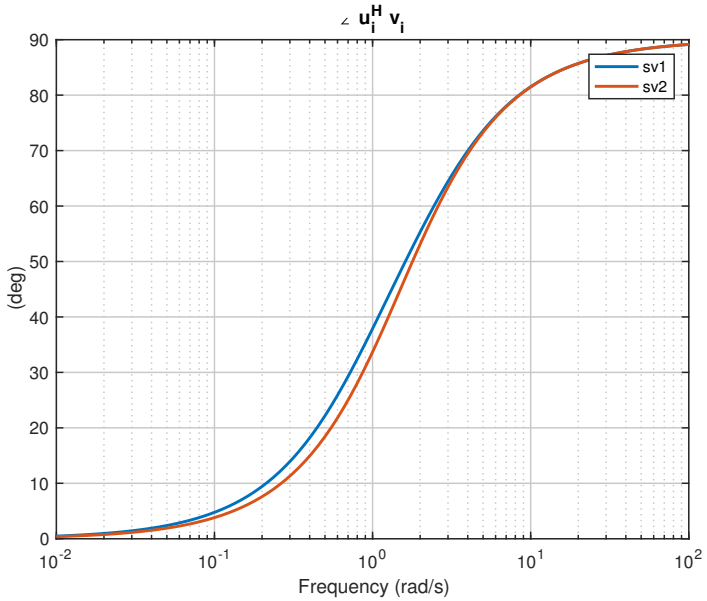


Figure 3.2: MIMO Phase Plot

This plot is obtained by using the MATLAB code given in Appendix A.10.

Example 3.6.2 *Consider a* 2×2 *TFM as shown below*

$$L(s) = \frac{1}{s+1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.34}$$

The phase of $L(s)$ *along* $s = j\omega$ *for reference vector* $w_i = u_i$ *is shown by Fig. 3.3.*

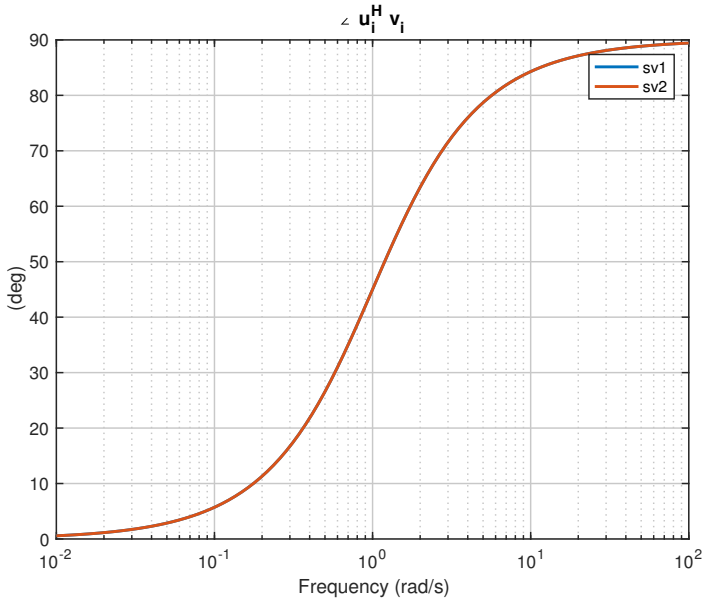


Figure 3.3: MIMO Phase Plot

Compare the above with Fig. 3.4, which shows the phase of $L(s) = \frac{1}{s+1}$.

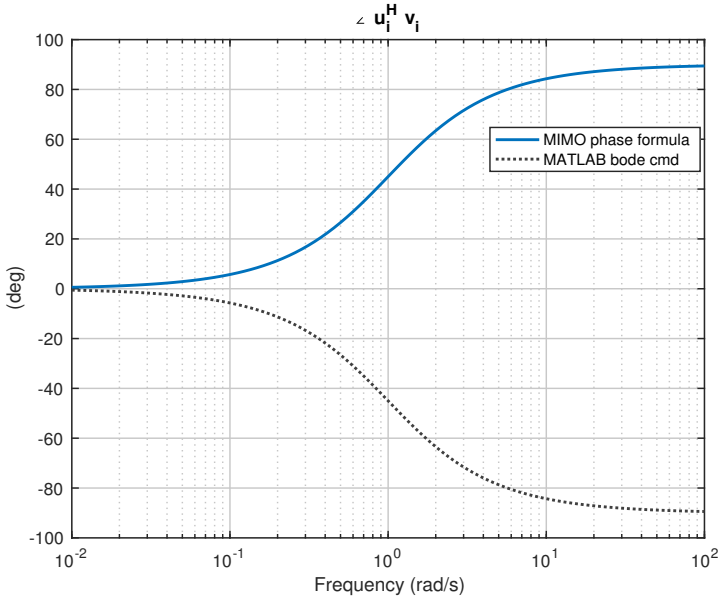


Figure 3.4: MIMO versus SISO Phase Plot

Figures 3.4 and 3.3 are obtained using the MATLAB code given in Appendix A.10

3.7 Bode Sensitivity Integral Constraint

In this section we study performance limitations imposed on sensitivity function given by Bode's sensitivity integral relations. The fundamental problem under investigation is how increasing the performance bandwidth of the system results in undesirable increase of peak sensitivity.

3.7.1 Bode's Generalized Sensitivity Integral Relation

Theorem 3.7.1 (Bode Sensitivity Integral) *[20, 138–144] Suppose that the open loop transfer function matrix $L(s)$ is rational and has at least double pole roll-off. Then for closed-loop stability the sensitivity transfer function matrix must satisfy*

$$\sum_j \int_0^\infty \ln \sigma_j \left(S(j\omega)\right) d\omega \quad = \pi \sum_{i=1}^N Re(p_i) \tag{3.35}$$

where $Re(p_i)$ represents real part of i-th RHP (unstable) pole and N represents number of RHP poles of $L(s)$. If the $L(s)$ is stable, then right hand side of the equaiton is 0. ■

Note that S (or T) here can be S_e (or T_e) or S_c (or T_c). For SISO systems with real RHP poles (p_i), the sensitivity function satisfies the following integral:

$$\int_0^\infty \ln \mid S(j\omega) \mid d\omega \quad = \pi \sum_{i=1}^N p_i \tag{3.36}$$

Similar relations can be derived for complementary sensitivity transfer function matrix can be seen in works by Chen and Åström et al. [141, 142, 145]. In those relations, the RHP transmission zeros of $L(s)$ determine the behavior of T, as opposed to RHP poles of $L(s)$.

3.7.2 Sensitivity Peaking Analysis Using Generic Bounds

A Lower Bound on Sensitivity Peak

Consider a SISO unity feedback loop with an open loop unstable pole located at $p \in \mathcal{R}$ ($p = 0$ if open loop stable).

Assumption 3.7.1 *Let the closed loop sensitivity function satisfy the following upper bound in the frequency range $\omega \in [0, \omega_s] \cup [f_2\omega_p, \infty)$ with $\omega_s < f_2\omega_p$ and $0 < f_2 \leq 1$:*

$$\ln|S(j\omega)| \leq \begin{cases} \ln \epsilon, & 0 \leq \omega < \omega_s \epsilon^{\frac{1}{m_1}} f_1^{\frac{m_1+m_2}{m_1}} \\ m_1 \ln\left(\frac{\omega}{f_1\omega_s}\right) - m_2 \ln f_1, & \omega_s \epsilon^{\frac{1}{m_1}} f_1^{\frac{m_1+m_2}{m_1}} \leq \omega < f_1\omega_s \\ m_2 \ln\left(\frac{\omega}{\omega_s}\right), & f_1\omega_s \leq \omega < \omega_s \\ -m_3 \ln\left(\frac{\omega}{\omega_p}\right), & f_2\omega_p \leq \omega < \omega_p \\ 0, & \omega_p \leq \omega < \infty \end{cases} \tag{3.37}$$

where $0 < \epsilon < 1$, $0 < f_1 \leq 1$, $\epsilon^{\frac{1}{m_1}} f_1^{\frac{m_2}{m_1}} \leq 1$, $m_1, m_2, m_3 > 0$ with the parameter ω_s corresponding to performance bandwidth and ω_p to maximum available bandwidth. ∎

Corollary 3.7.1 (Corollary to Bode Sensitivity Integral) *If Assumption 3.7.1 is satisfied, Then a lower bound on peak of sensitivity function magnitude ($\|S\|_{\mathcal{H}^\infty} > 1$) is as follows*

$$\|S\|_{\mathcal{H}^\infty} \geq EXP\left(\frac{\begin{array}{r} \pi p + m_1 f_1 \omega_s \left(1 - \epsilon^{\frac{1}{m_1}} f^{\frac{m_2}{m_1}}\right) + 2m_2\omega_s f_1 \ln f_1 \left(1 + \epsilon^{\frac{1}{m_1}} f^{\frac{m_2}{m_1}}\right) \\ + m_2\omega_2(1 - f_1) - m_3\omega_p(1 - f_2 - f_2 \ln f_2) \end{array}}{f_2(\omega_p - \omega_s)} \right) = M^* \tag{3.38}$$

∎

Let $m_1 = m_2 = m$ and $m_3 \to \infty$. Then a lower bound on peak and an upper bound on unity gain crossover of sensitivity magnitude are as follows

$$\|S\|_{\mathcal{H}^\infty} \geq EXP\left(\frac{\pi p + \omega_s m\left(1 - \sqrt[m]{\epsilon}\right)}{\omega_p - \omega_s}\right) = M^* \tag{3.39}$$

$$\omega_g \leq \frac{\omega_p \ln M^* - \pi p}{\ln M^* + m\left(1 - \sqrt[m]{\epsilon}\right)} \tag{3.40}$$

Figure 3.5 shows the bounds on the sensitivity function magnitude. The solid-black-thick lines in the frequency range $\omega \in [0, \omega_s] \cup [\omega_p, \infty)$ indicate upper limit on sesnsitivity magnitude. Using the inequalities (3.39) and (3.40), it can be seen that reducing the sensitivity in one frequency range (e.g., $\omega \in [0, \omega_s]$) necessarily increases the sensitivity in some other frequency range (e.g., $\omega \in (\omega_s, \omega_p)$). M^* in the figure represents the right hand side of the inequality in (3.39). That is,

$$M^* = EXP\left(\frac{\pi p + \omega_s m\left(1 - \sqrt[m]{\epsilon}\right)}{\omega_p - \omega_s}\right) \tag{3.41}$$

M^* represents the lower bound on peak sensitivity magnitude.

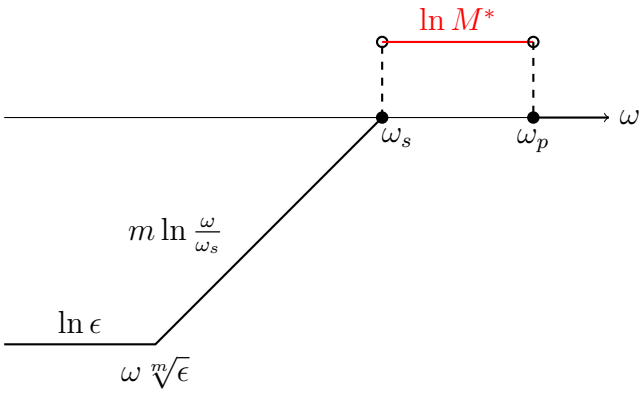


Figure 3.5: Visualization of Sensitivity Bounds

Comment 3.7.1 (Tradeoff between M^* and ω_s) *From the inequality in (3.39), it can be seen that for fixed ω_p, ϵ, m and p, the lower bound on peak sensitivity magnitude (M^*) increases as ω_s approaches ω_p.*

Comment 3.7.2 (M^* **versus** m) *From the inequality in (3.39), it can be seen that for fixed* ω_p, ω_s, p, *and* $\epsilon = 0$, *as the order of the sensitivity function* (m) *is increased,* M^* *increases. Hence, increasing the order of sensitivity below the performance bandwidth worsens the sensitivity magnitude peak. When* $0 < \epsilon < 1$, *it can be shown that* M^* *still increases with* m, *but the increase is slower compared to the case when* $\epsilon = 0$.

Comment 3.7.3 (M^* **versus** ϵ) *From the inequality in (3.39), it can be seen that for fixed* ω_p, ω_s, m *and* p, M^* *increases as* ϵ *is lowered and decreases as* ϵ *is increased.*

Comment 3.7.4 (M^* **versus** ω_p) *From the inequality in (3.39), it can be seen that for fixed* ω_s, ϵ, m *and* p, M^* *increases as* ω_p *decreses and decreases as* ω_p *increases.*

Comment 3.7.5 (M^* **versus** p) *From the inequality in (3.39), it can be seen that for fixed* ω_p, ω_s, ϵ *and* m, M^* *increases as* p *increases and decreases as* p *decreases.*

Minimum Upper Bound on Sensitivity Peak [146]

Assumption 3.7.2 *Let the closed loop sensitivity function satisfy the following upper bound:*

$$\ln|S(j\omega)| = \begin{cases} \ln\epsilon, & 0 \le \omega < \omega_s \sqrt[m_1]{\epsilon} \\ m_1 \ln\left(\frac{\omega}{\omega_s}\right), & \omega_s \sqrt[m_1]{\epsilon} \le \omega < \omega_s \\ m_2 \ln\left(\frac{\omega}{\omega_s}\right), & \omega_s \le \omega < \omega_s \sqrt[m_2]{M} \\ \ln M, & \omega_s \sqrt[m_2]{M} \le \omega < \frac{\omega_p}{\sqrt[m_3]{M}} \\ -m_3 \ln\left(\frac{\omega}{\omega_p}\right), & \frac{\omega_p}{\sqrt[m_3]{M}} \le \omega < \omega_p \\ 0, & \omega_p \le \omega < \infty \end{cases} \tag{3.42}$$

where $M > 1$ *is an upper bound for the peak sensitivity,* ω_s *is performance bandwidth parameter and* ω_p *is maximum available bandwidth parameter. Further,*

$$\omega_s \sqrt[m_2]{M} \le \frac{\omega_p}{\sqrt[m_3]{M}} \tag{3.43}$$

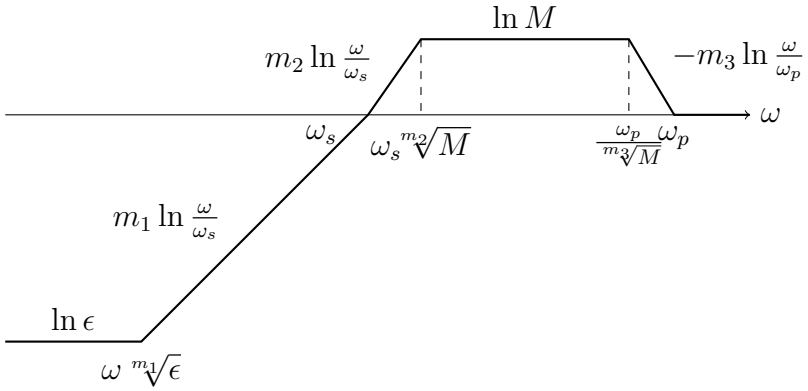


Figure 3.6: Visualization of Sensitivity Bound

Corollary 3.7.2 *If Assumption 3.7.2 is satisfied, then using Bode's sensitivity integral formula in Eqn. 3.36, it follows that,*

$$-\sqrt[m_3]{M}\left(-\pi p + \frac{1}{m_1}\epsilon \ln(\epsilon)\omega_s - m_1\omega_s - \sqrt[m_1]{\epsilon}\ln(\epsilon)\omega_s + m_1 \sqrt[m_1]{\epsilon}\omega_s + m_2\omega_s + m_3\omega_p\right) \tag{3.44}$$
$$+ \sqrt[m_2]{M}\sqrt[m_3]{M} m_2\omega_s + m_3\omega_p \geq 0$$

$$\omega_s \leq \frac{\pi p + \frac{m_3}{\sqrt[m_3]{M}}\omega_p - m_3\omega_p}{\frac{1}{m_1}\epsilon\ln(\epsilon) - m_1 - \sqrt[m_1]{\epsilon}\ln(\epsilon) + m_1\sqrt[m_1]{\epsilon} - m_2\sqrt[m_2]{M} + m_2} \tag{3.45}$$

Figures 3.7 - 3.20 show the behavior of M versus ω_s and vice versa. It is assumed that $\omega_p = 10$, $p = 0$ and $\epsilon = 0^+$. The red curves indicate the inequality (3.43). The blue curves are valid in the region below the red curves. In the figures corresponding to behavior of M versus ω_s, the blue curve does not exit for some high values of ω_s. These values correspond to those where the solution to (3.44) are imaginary (not real). These figures were generated using the MATLAB code given in Appendix A.2.

Comment 3.7.6 *From Figures 3.7 - 3.20, it can be seen that if we fix m_3 and set $m_2 = m_1$, then as we increase m_1, M attains higher at all values of ω_s. This indicates that increasing the slope of sensitivity at low frequencies (both m_1 and m_2 together) adversely affects the sensitivity peak. It can also be observed that in these cases, the maximum value of ω_s that is under the red curve remains constant even as m_1 is increased. This indicates that increasing the slope of sensitivity (both m_1 and m_2 together) does not help the inequalities 3.44 - 3.45 hold good on a wider frequency range (i.e., higher value of ω_s).*

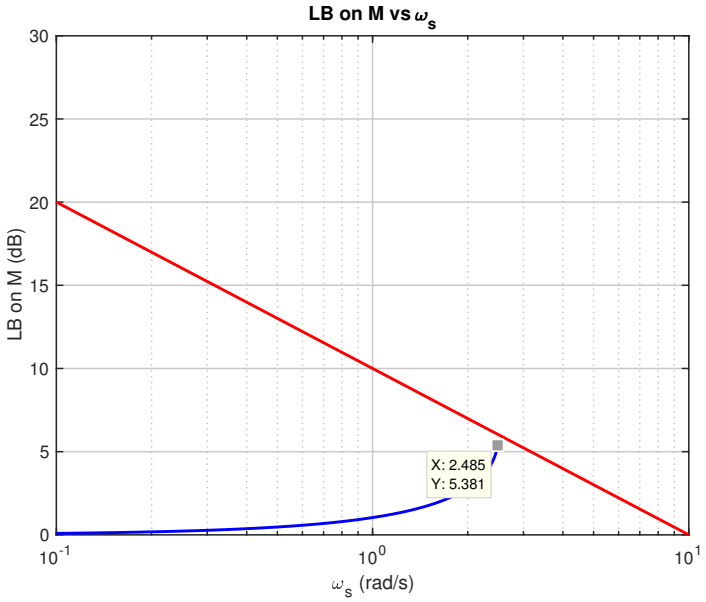


Figure 3.7: Bode's Sensitivity Integral: M versus ω_s for $m_1 = m_2 = 1$ and $m_3 = 1$

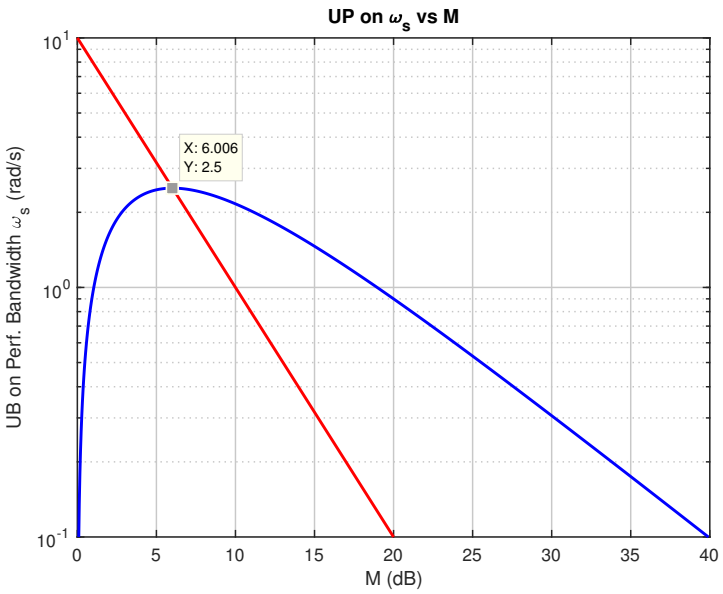


Figure 3.8: Bode's Sensitivity Integral: ω_s versus M for $m_1 = m_2 = 1$ and $m_3 = 1$

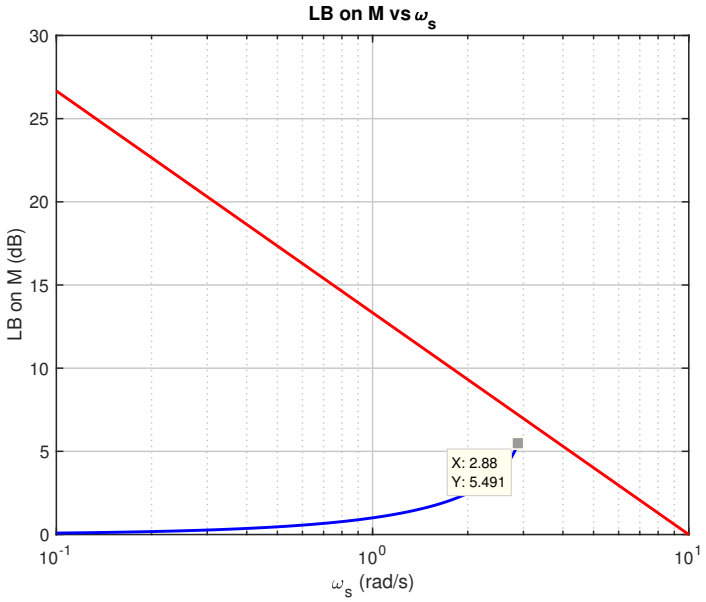


Figure 3.9: Bode's Sensitivity Integral: M versus ω_s for $m_1 = m_2 = 1$ and $m_3 = 2$

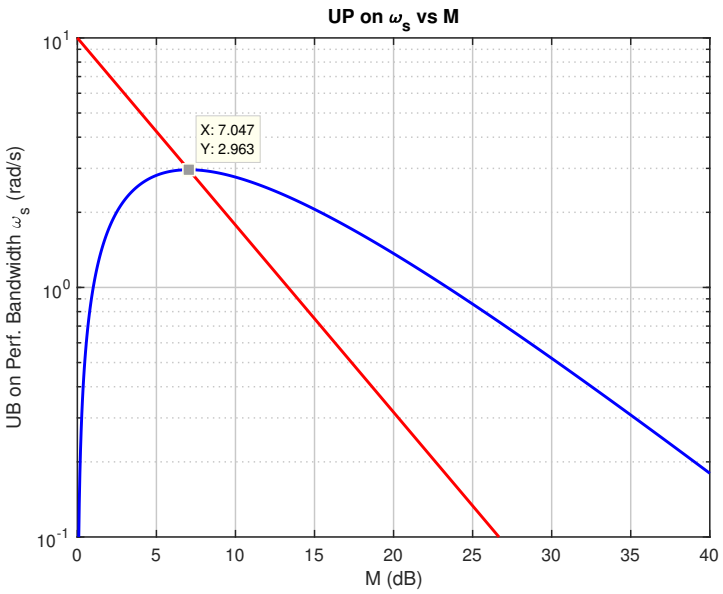


Figure 3.10: Bode's Sensitivity Integral: ω_s versus M for $m_1 = m_2 = 1$ and $m_3 = 2$

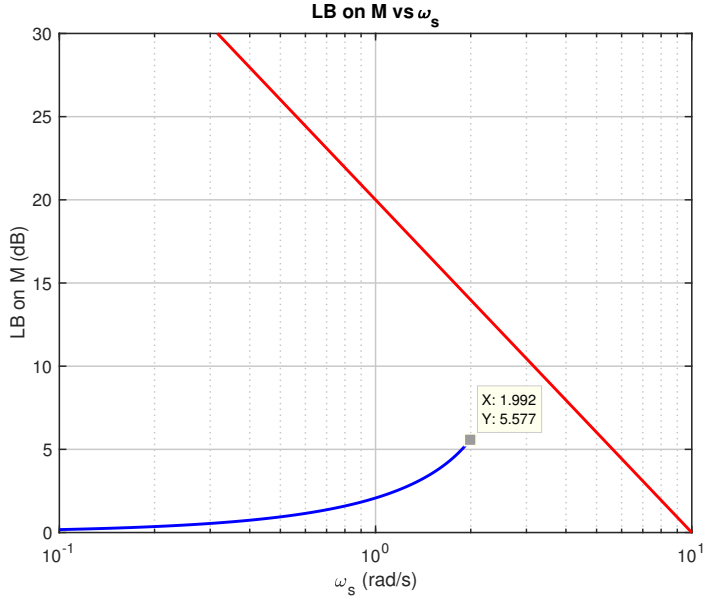


Figure 3.11: Bode's Sensitivity Integral: M versus ω_s for $m_1 = m_2 = 2$ and $m_3 = 2$

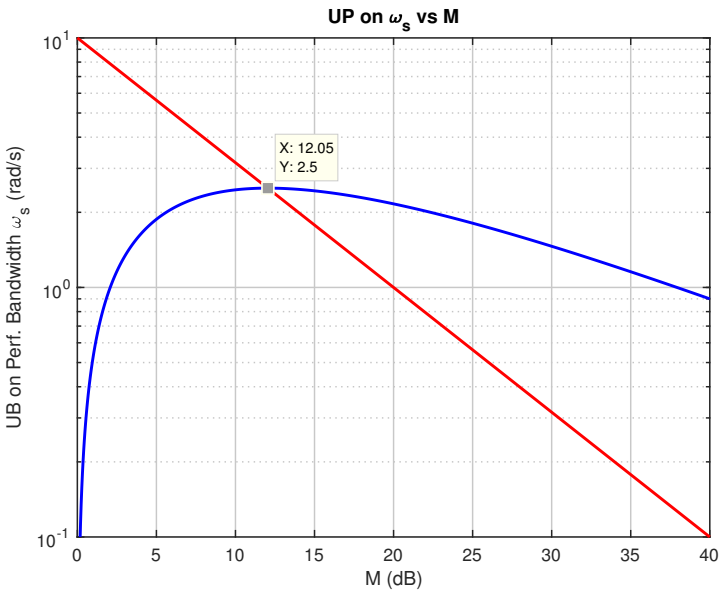


Figure 3.12: Bode's Sensitivity Integral: ω_s versus M for $m_1 = m_2 = 2$ and $m_3 = 2$

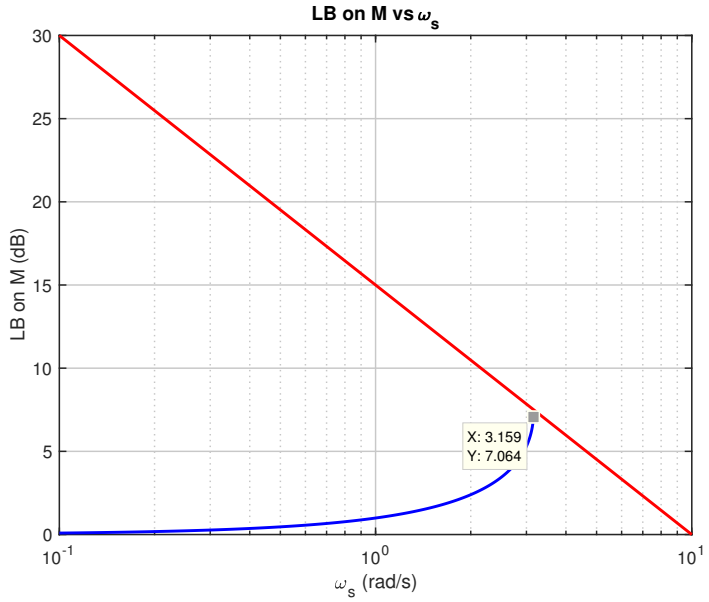


Figure 3.13: Bode's Sensitivity Integral: M versus ω_s for $m_1 = m_2 = 1$ and $m_3 = 3$

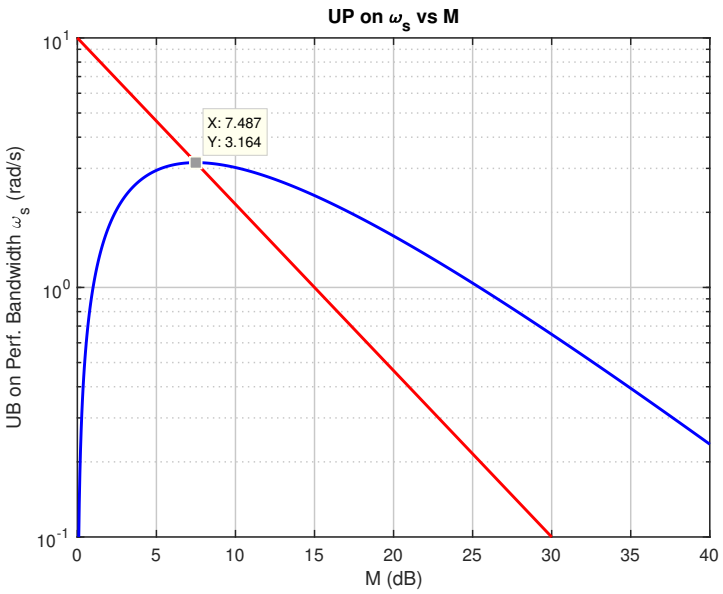


Figure 3.14: Bode's Sensitivity Integral: ω_s versus M for $m_1 = m_2 = 1$ and $m_3 = 3$

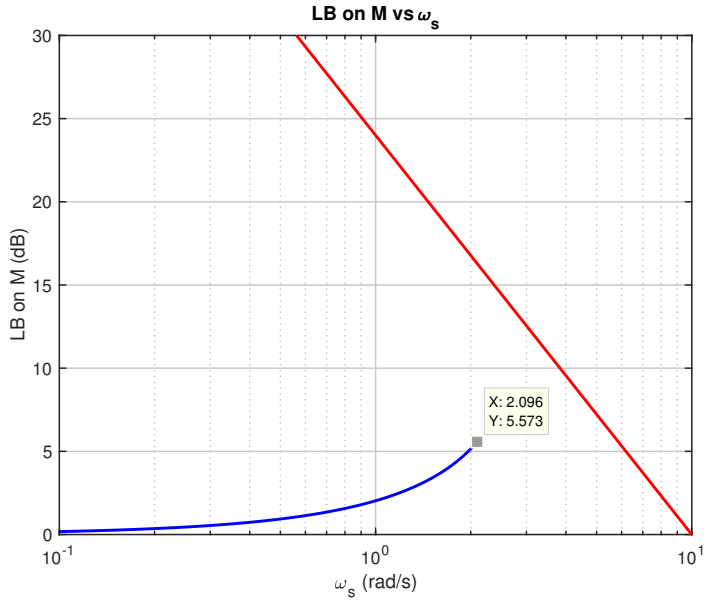


Figure 3.15: Bode's Sensitivity Integral: M versus ω_s for $m_1 = m_2 = 2$ and $m_3 = 3$

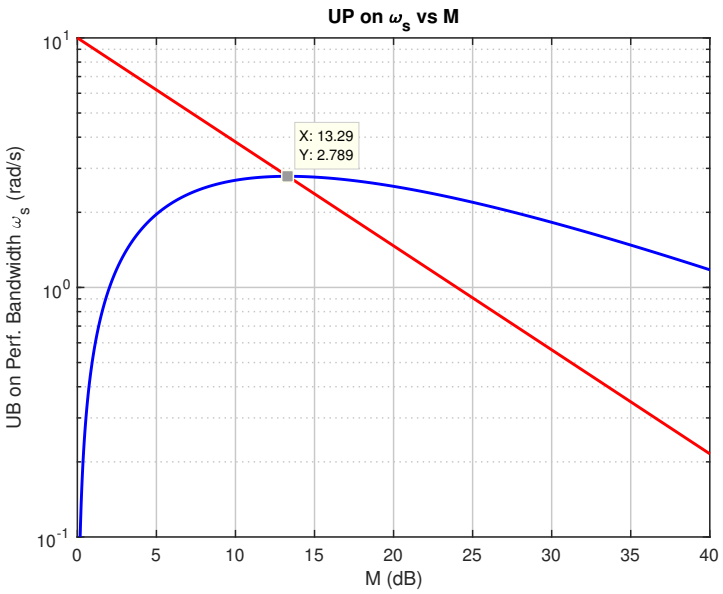


Figure 3.16: Bode's Sensitivity Integral: ω_s versus M for $m_1 = m_2 = 2$ and $m_3 = 3$

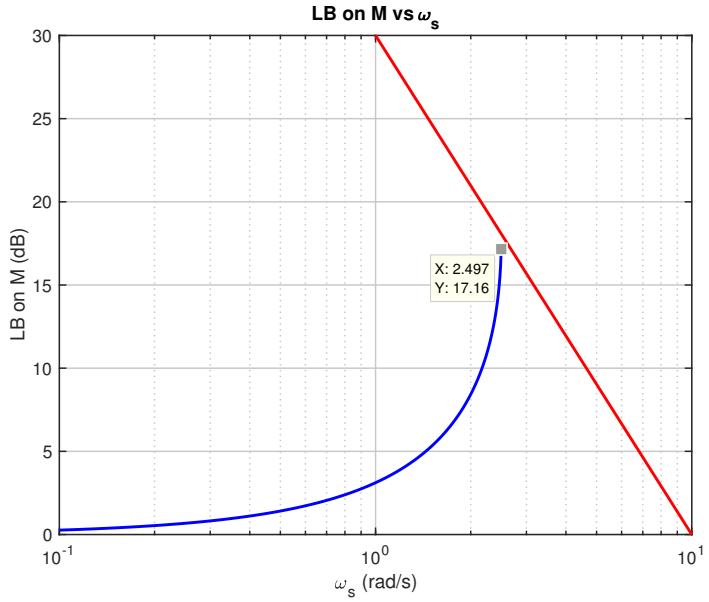


Figure 3.17: Bode's Sensitivity Integral: M versus ω_s for $m_1 = m_2 = 3$ and $m_3 = 3$

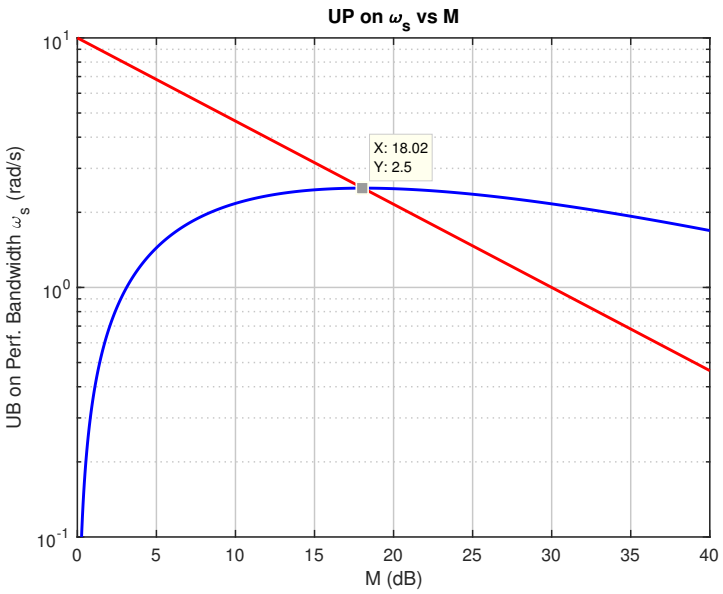


Figure 3.18: Bode's Sensitivity Integral: ω_s versus M for $m_1 = m_2 = 3$ and $m_3 = 3$

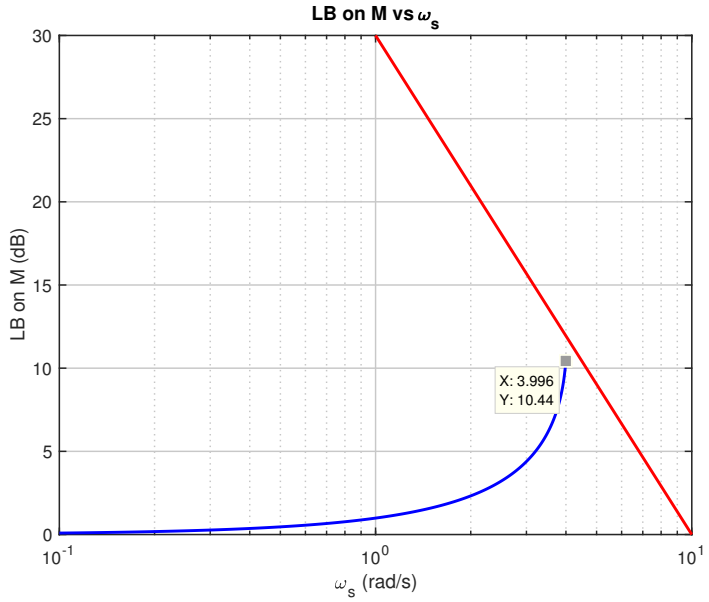


Figure 3.19: Bode's Sensitivity Integral: M versus ω_s for $m_1 = 1$, $m_2 = 3$ and $m_3 = 3$

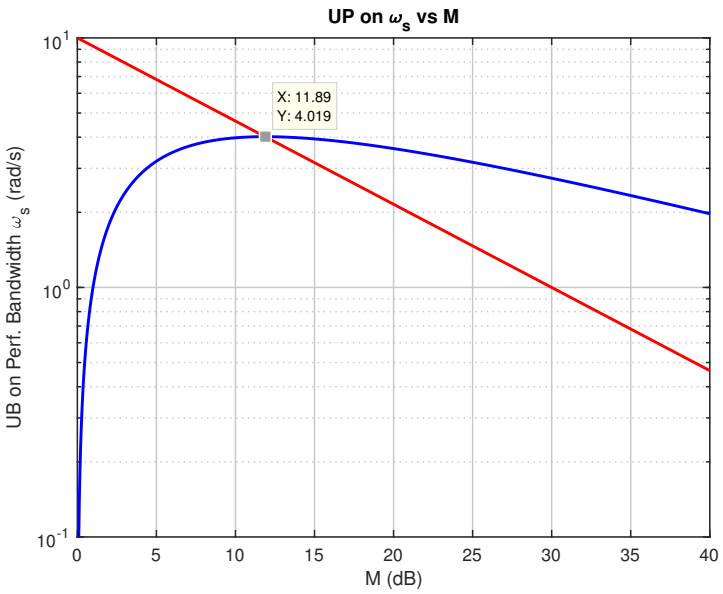


Figure 3.20: Bode's Sensitivity Integral: ω_s versus M for $m_1 = 1$, $m_2 = 3$ and $m_3 = 3$

3.8 Peak Sensitivity Bounds Imposed by RHP Zeros

3.8.1 Weighted Sensitivity Peak Relation

Theorem 3.8.1 (Weighted Sensitivity Peak) *[20] Let a SISO plant $P(s)$ have a RHP zero z and let $W_s(s)$ be a stable weighting function. Then for closed loop stability the weighted sensitivity function must satisfy*

$$\|W_s S\|_{\mathcal{H}^\infty} \geq |W_s(z)| \tag{3.46}$$

■

Let a SISO weighting function $W(s)$ be an upper bound on SISO sensitivity. Then we have [20]

$$|S(j\omega)| < \frac{1}{|W_p(j\omega)|} \quad \forall\omega \tag{3.47}$$

$$|W_p(z)| < 1 \tag{3.48}$$

3.8.2 *Sensitivity Peaking Analysis Using Generic Weighting Functions*

We now consider several weighting functions (upper bounds on sensitivity) to derive bounds on peak sensitivity that can be achieved. Tables 3.1 and 3.2 show the bounds on sensitivity based on several generic weighting functions. Here, the parameters assocated with $W(s)$ can be thought of as follows: $M > 1$ is an upper bound for the peak sensitivity, ω_s is performance bandwidth parameter and ω_p is maximum available bandwidth parameter

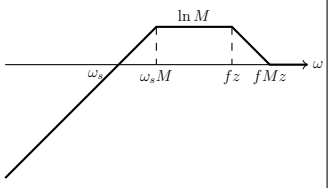
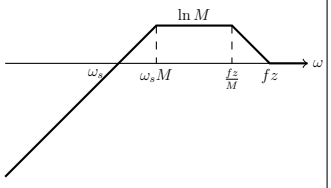
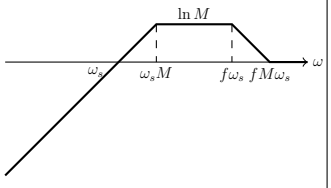
Weight (W)	Figure (Approx. W^{-1})	Bounds on M and ω_s
$W(s) = \frac{s+M\omega_s}{s}\frac{s+fz}{s+fMz}$		$M > \dfrac{z}{z-\left(\frac{f+1}{f}\right)\omega_s}$ $\omega_s < \dfrac{z(M-1)f}{M(f+1)}$ $fz > M\omega_s$
$W(s) = \frac{s+M\omega_s}{s}\frac{s+fz/M}{s+fz}$		$M^2\omega_s + Mf\left(\omega_s - Mz\right) + zf < 0$ $\omega_s < \dfrac{z(M-1)f}{M(M+f)}$ $fz > M^2\omega_s$
$W(s) = \frac{s+M\omega_s}{s}\frac{s+f\omega_s}{s+fM\omega_s}$		$M > \dfrac{z}{\left(\frac{f-1}{f}\right)z-\omega_s}$ $\omega_s < \dfrac{z(Mf-M-f)}{Mf}$ $f > M$

Table 3.1: Sensitivity Bounds Due to RHP Zeros

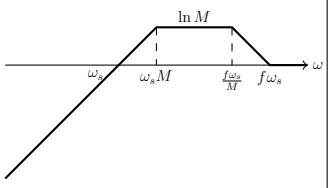
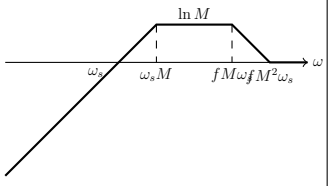
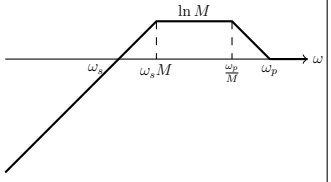
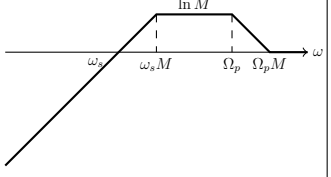
Weight (W)	Figure (Approx. W^{-1})	Bounds on M and ω_s
$W(s) = \frac{s+M\omega_s}{s}\frac{s+f\omega_s/M}{s+f\omega_s}$		$zM^2 - f(z-\omega_s)M + zf < 0$ $\omega_s < \frac{z(Mf - M^2 - f)}{Mf}$ $f > M^2$
$W(s) = \frac{s+M\omega_s}{s}\frac{s+fM\omega_s}{s+fM^2\omega_s}$		$M > \frac{z\,(1+1/f)}{z-\omega_s}$ $\omega_s < \frac{z(Mf - f - 1)}{Mf}$ $f > 1$
$W(s) = \frac{s+M\omega_s}{s}\frac{s+\omega_p/M}{s+\omega_p}$		$z\omega_s M^2 - (z-\omega_s)\omega_p M$ $-(M-1)z\omega_p < 0$ $\omega_s < \frac{z\omega_p(M-1)}{(\omega_p + Mz)M}$ $\omega_p > M^2\omega_s$
$W(s) = \frac{s+\omega_s}{s}\frac{s+\Omega_p}{s+\Omega_p M}$		$M > \frac{z}{z - \omega_s - \frac{z\omega_s}{\Omega_p}}$ $= \frac{\Omega_p}{\Omega_p - \omega_s - \frac{\Omega_p\omega_s}{z}}$ If $\Omega_p >> z, M > \frac{z}{z-\omega_s}$ If $z >> \Omega_p, M > \frac{\Omega_p}{\Omega_p - \omega_s}$ $\omega_s < \frac{z\Omega_p(M-1)}{(z+\Omega_p)M}$ $\Omega_p > M\omega_s$

Table 3.2: Sensitivity Bounds Due to RHP Zeros

Consider the asymtotic approximation of magnitude of a generic weighting function

$$W(s) = \frac{\left(s + \omega_s \sqrt[k_1]{\epsilon}\right)^{k_1} \left(s + \omega_s\right)^{k_2 - k_1} \left(s + \omega_p\right)^{k_3}}{\left(s + \omega_s \sqrt[k_2]{M}\right)^{k_2} \left(s + \frac{\omega_p}{\sqrt[k_3]{M}}\right)^{k_3}} \tag{3.49}$$

This weighting function can be visualized as in Figure 3.21.

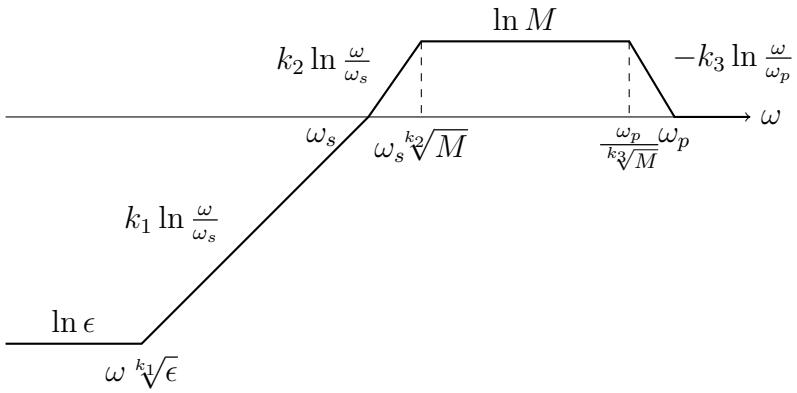


Figure 3.21: Bode Asymptotic Approximation Magnitude of $W(s)$

Using the bound on weighting function $|W(z)| < 1$ and assuming $\epsilon = 0^+$, we obtain

$$\frac{z^{k_1} \left(z + \omega_s\right)^{k_2 - k_1} \left(z + \omega_p\right)^{k_3}}{\left(z + \omega_s \sqrt[k_2]{M}\right)^{k_2} \left(z + \frac{\omega_p}{\sqrt[k_3]{M}}\right)^{k_3}} < 1 \tag{3.50}$$

If $\omega_p >> z$, then the bound (3.50) becomes

$$\frac{z^{k_1} \left(z + \omega_s\right)^{k_2 - k_1} \omega_p^{k_3}}{\left(z + \omega_s \sqrt[k_2]{M}\right)^{k_2} \left(z + \frac{\omega_p}{\sqrt[k_3]{M}}\right)^{k_3}} < 1 \tag{3.51}$$

If $z >> \omega_p$ and hence $z >> \omega_s$, then the bound (3.50) becomes

$$\frac{z^{k_2 k_3}}{\left(z + \omega_s \sqrt[k_2]{M}\right)^{k_2} \left(z + \frac{\omega_p}{\sqrt[k_3]{M}}\right)^{k_3}} < 1 \tag{3.52}$$

Effect of RHP Zero on Sensitivity Peak ($z << \omega_p$).

Figures 3.22 - 3.27 show the behavior of M versus ω_s and vice versa. It is assumed that $z = 10$, $\epsilon = 0^+$ and $\omega_p = 10^5$. It can be seen that as the performance bandwidth approaches closer the RHP zero value, the bound on sensitivity peak increases, and goes to infinity when they become equal. Note that this captures the effect of RHP zero (z) on sensitivity peak, rather than the available bandwidth (ω_p). This is due to the frequency of RHP zero being much below available bandwidth. These figures are generated using the MATLAB code given in Appendix A.3.

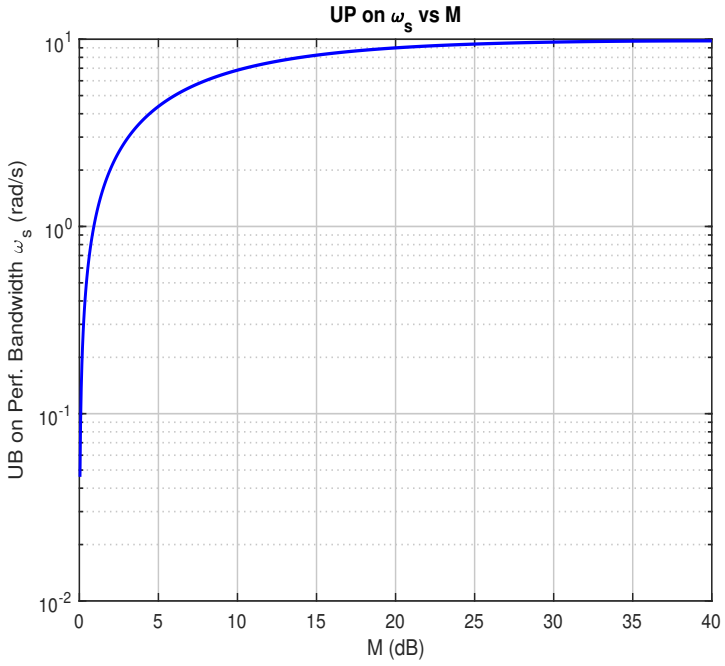


Figure 3.22: Limitation Due to RHP Zero ($z << \omega_p$): ω_s versus M for $k_1 = k2 = 1$ and $k_3 = 1$

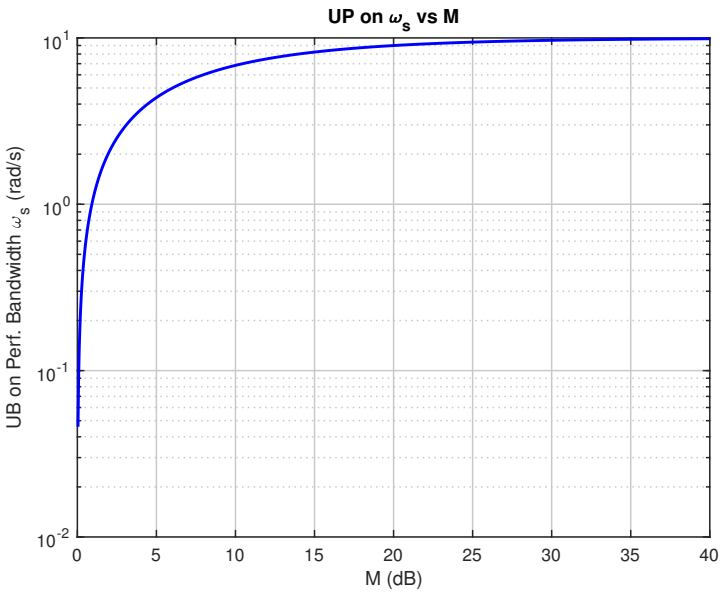


Figure 3.23: Limitation Due to RHP Zero ($z << \omega_p$): ω_s versus M for $k_1 = k2 = 1$ and $k_3 = 2$

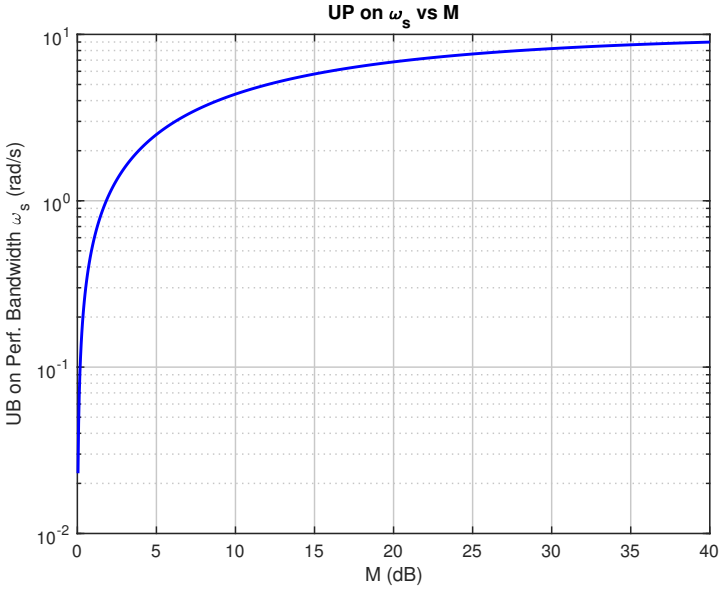


Figure 3.24: Limitation Due to RHP Zero ($z << \omega_p$): ω_s versus M for $k_1 = k2 = 2$ and $k_3 = 2$

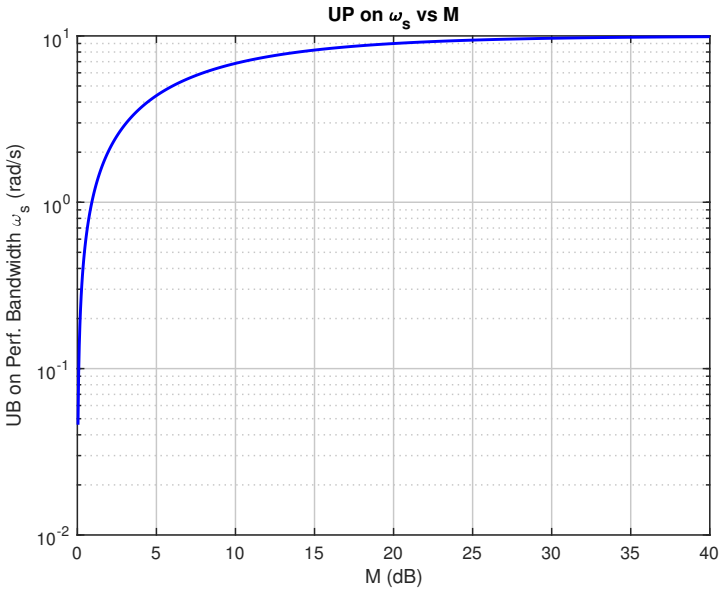


Figure 3.25: Limitation Due to RHP Zero ($z << \omega_p$): ω_s versus M for $k_1 = k2 = 1$ and $k_3 = 3$

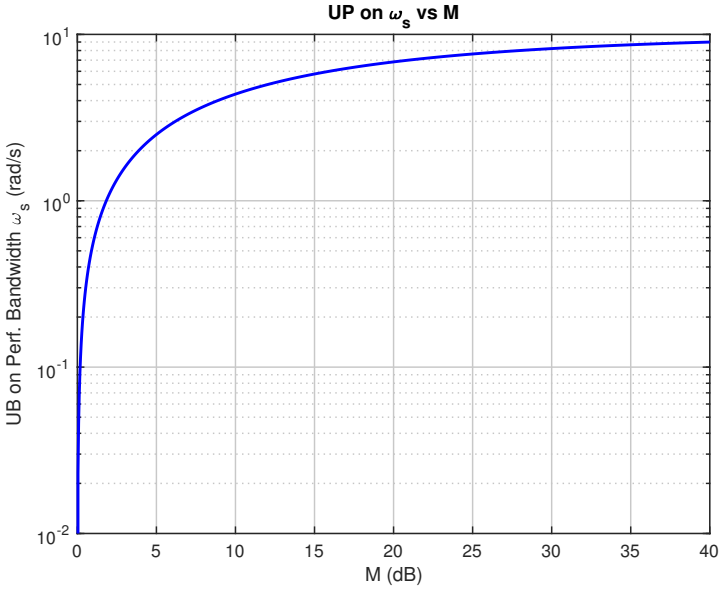


Figure 3.26: Limitation Due to RHP Zero ($z << \omega_p$): ω_s versus M for $k_1 = k2 = 2$ and $k_3 = 3$

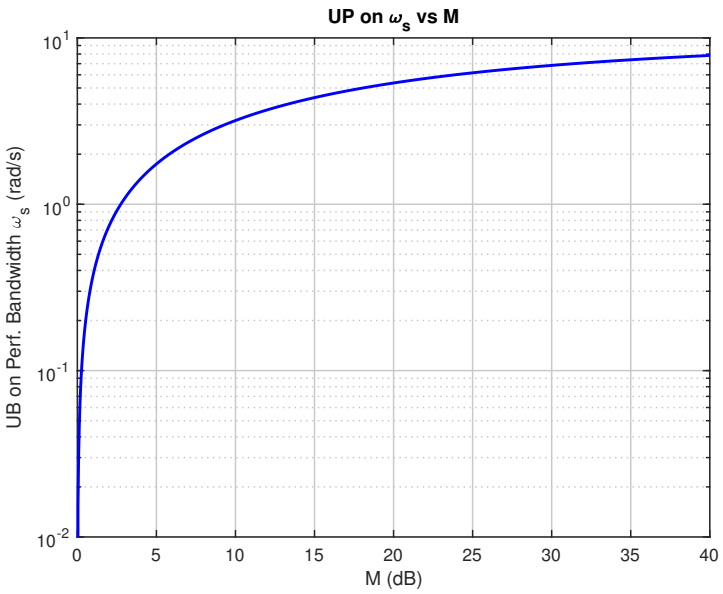


Figure 3.27: Limitation Due to RHP Zero ($z << \omega_p$): ω_s versus M for $k_1 = k2 = 3$ and $k_3 = 3$

Effect of Available Bandwidth on Sensitivity Peak ($\omega_p << z$.

Figures 3.28 - 3.33 show the behavior of M versus ω_s and vice versa. It is assumed that $z = 10^5$, $\epsilon = 0^+$ and $\omega_p = 10$. The red curves indicate the inequality (3.43). The blue curves are valid in the region below the red curves. It can be seen that as the performance bandwidth ω_s increases, the bound on sensitivity peak increases. This behaviour is only seen when the blue curves lie below the red curves. Beyond this region, the inequality (3.43) is violated. This is a limitation inherently present due to the finite slopes of the weighting functions.

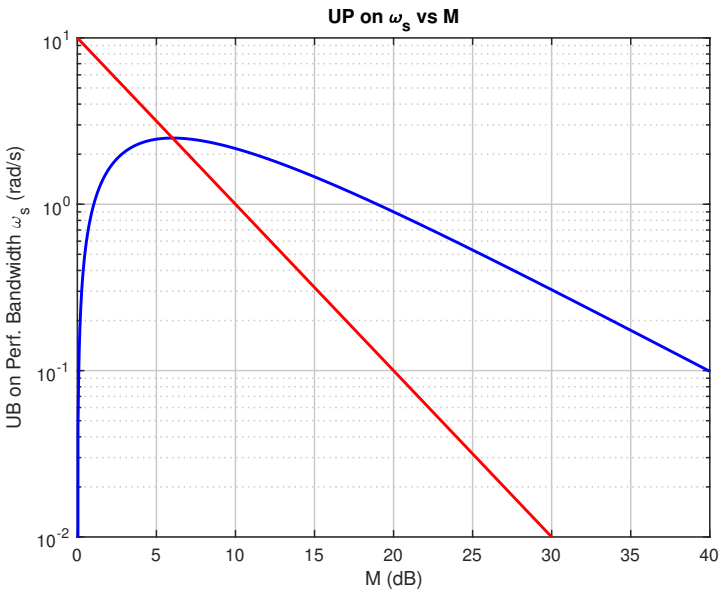


Figure 3.28: Limitation Due to Available Bandwidth ($\omega_p << z$): ω_s versus M for $k_1 = k2 = 1$ and $k_3 = 1$

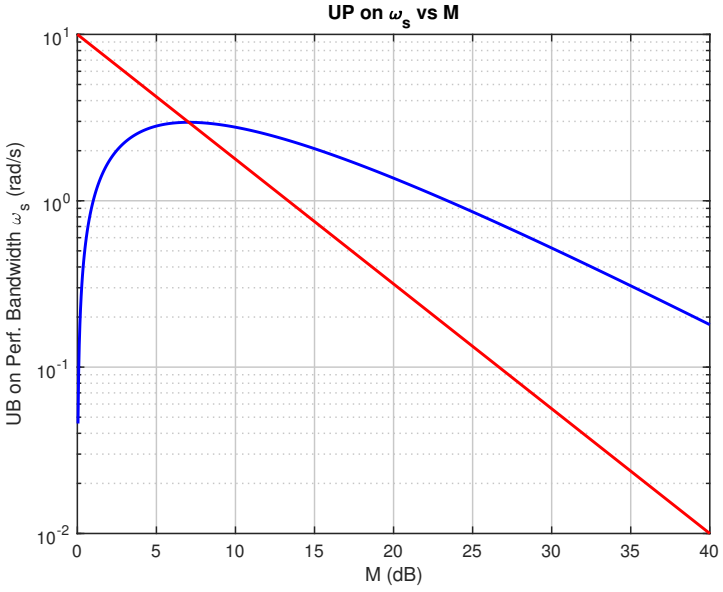


Figure 3.29: Limitation Due to Available Bandwidth ($\omega_p << z$): ω_s versus M for $k_1 = k2 = 1$ and $k_3 = 2$

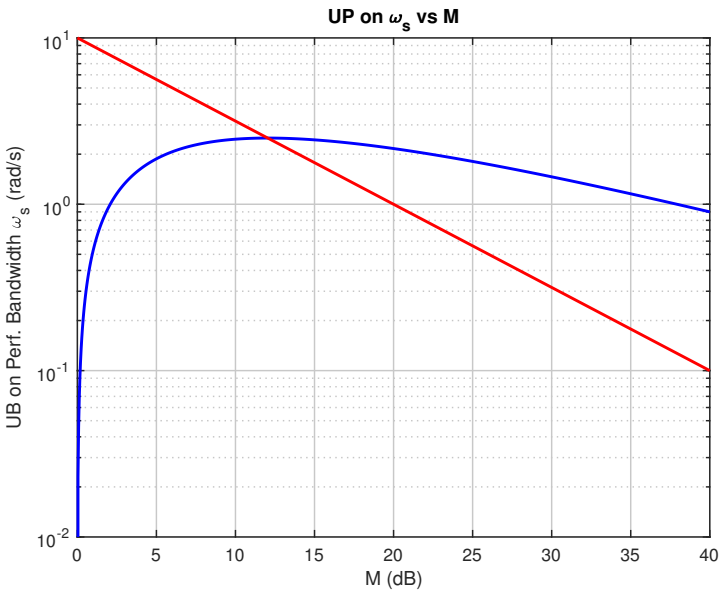


Figure 3.30: Limitation Due to Available Bandwidth ($\omega_p << z$): ω_s versus M for $k_1 = k2 = 2$ and $k_3 = 2$

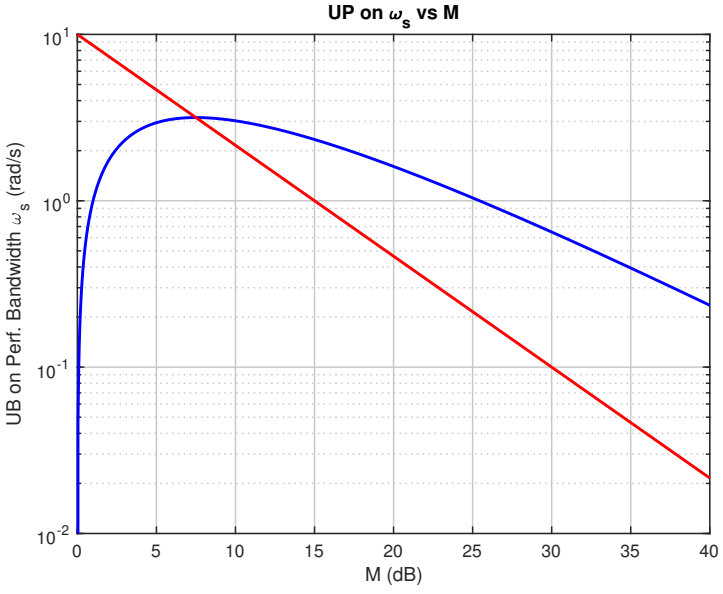


Figure 3.31: Limitation Due to Available Bandwidth ($\omega_p << z$): ω_s versus M for $k_1 = k2 = 1$ and $k_3 = 3$

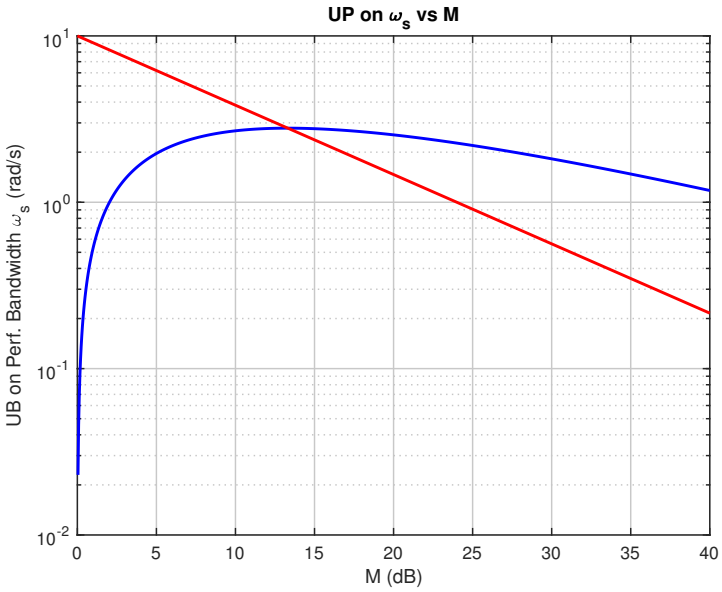


Figure 3.32: Limitation Due to Available Bandwidth ($\omega_p << z$): ω_s versus M for $k_1 = k2 = 2$ and $k_3 = 3$

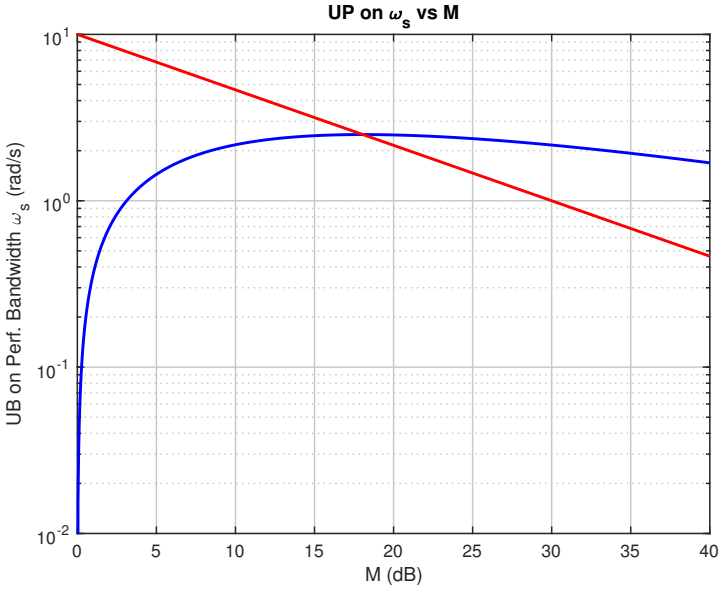


Figure 3.33: Limitation Due to Available Bandwidth ($\omega_p << z$): ω_s versus M for $k_1 = k2 = 3$ and $k_3 = 3$

3.9 Stability Margin Bounds from Closed Loop Frequency Response Bounds

Consider a SISO feedback system which is closed loop stable. Suppose that the sensitivity transfer function S satisfies the following bound

$$|S(j\omega)| \stackrel{\text{def}}{=} \frac{1}{|1+L(j\omega)|} \leq \alpha \tag{3.53}$$

for all $\omega \geq 0$ where $\alpha \geq 1$, and suppose that complementary transfer function T satisfies the following bound

$$|T(j\omega)| \stackrel{\text{def}}{=} \frac{L(j\omega)}{|1+L(j\omega)|} \leq \beta \tag{3.54}$$

for all $\omega \geq 0$ where $\beta \geq 1$. We can obtain the following bounds on stability margins [146]

$$\uparrow GM \geq \max\left\{\frac{\alpha}{\alpha-1}, 1+\frac{1}{\beta}\right\} > 1 \tag{3.55}$$

$$\downarrow GM \geq \max\left\{\frac{\alpha}{\alpha+1}, 1-\frac{1}{\beta}\right\} < 1 \tag{3.56}$$

$$|PM| \geq \max\left\{2\sin^{-1}\left(\frac{1}{2\alpha}\right), 2\sin^{-1}\left(\frac{1}{2\beta}\right)\right\} \tag{3.57}$$

$$\alpha > \max\left\{\frac{\downarrow GM}{1-\downarrow GM}, \frac{1}{2\sin\left(\frac{PM}{2}\right)}, \frac{\uparrow GM}{\uparrow GM-1}\right\} \tag{3.58}$$

$$\beta > \max\left\{\frac{1}{1-\downarrow GM}, \frac{1}{2\sin\left(\frac{PM}{2}\right)}, \frac{1}{\uparrow GM-1}\right\} \tag{3.59}$$

3.10 Sensitivity Bounds Imposed by RHP Poles and RHP Zeros

We have the following relation for peak sensitivity and complementary sensitivity at the plant output: $\|S\|_\infty \geq c$, $\|T\|_\infty \geq c$, where

$$\phi = \arccos|y_z^H y_p| \tag{3.60}$$

$$c = \sqrt{\sin^2\phi + \frac{|z+p|^2}{|z-p|^2}\cos^2\phi} \tag{3.61}$$

where y_z is output directionality of the NMP zero, y_p is the output directionality of the RHP pole, and ϕ is the angle between them.

For a SISO plant with a single RHP zero (z) and a single RHP pole (p), the following holds [20]:

$$\int_0^\infty \ln|S(j\omega)| \quad \frac{2z}{z^2+\omega^2} \quad d\omega \quad = \quad \pi \ \ln\left|\frac{p+z}{p-z}\right| \tag{3.62}$$

3.11 Impact of Uncertainty on Sensitivity

We have the following bound on the impact of multiplicative output uncertainty E_O and input uncertainty E_I on the resulting sensitivity S' [9, 20]:

$$\bar{\sigma}(S') \leq \bar{\sigma}(S)\bar{\sigma}((I+E_O T)^{-1}) \tag{3.63}$$

$$\bar{\sigma}(S') \leq \gamma(P)\bar{\sigma}(S)\bar{\sigma}((I+E_I T_c)^{-1}) \tag{3.64}$$

where T_c is complementary sensitivity at plant input/control.

3.12 Summary and Conclusions

In this chapter, we analyzed the design challenges and tradeoffs associated with control of SISO and MIMO systems. We defined the open and closed loop transfer function matrices (TFMs) for a generic MIMO system using standard P-K feedback structure. Difficulty in control design due to high plant condition number, Bode's sensitivity integral constraint, and Right Half Plant (RHP) poles/zeros, were studied. These help in posing control problems (e.g., selecting weighting functions) using GMS studied in Chapters 6 and 7.

Chapter 4

GENERALIZED $\mathcal{H}^\infty$ MIXED SENSITIVITY OPTIMIZATION CONTROL DESIGN METHODOLOGY

4.1 Overview

In this chapter, we present the Generalized $\mathcal{H}^\infty$ Mixed Sensitivity methodology (GMS) problem posed as a constrained multiobjective optimization problem. Typical control relevant specifications and standard mixed sensitivity methods are first discussed. The GMS problem is then defined along with discussion on different variations based on desired objectives and feedback architectures.

4.2 Typical Closed Loop Frequency-Domain Design Objectives

General closed loop objectives associated with feedback design may be stated as follows:

- the closed loop system should be stable
- $\bar{\sigma}\left[S_e(j\omega)\right]$ and $\bar{\sigma}\left[S_c(j\omega)\right]$ should be small at low fequencies for good low frequency command following and disturbance attenuation
- $\bar{\sigma}\left[K(j\omega)S_e(j\omega)\right]$ should not be too large to prevent the controls from getting too large for anticipated exogneous signals
- $\bar{\sigma}\left[P(j\omega)S_c(j\omega)\right]$ should be small at high frequencies for good high frequency input disturbance attenuation
- $\bar{\sigma}\left[P(j\omega)S_c(j\omega)\right]$ should be small at low frequencies for good low frequency input disturbance attenuation

- $\bar{\sigma}\left[T_e(j\omega)\right]$ and $\bar{\sigma}\left[T_c(j\omega)\right]$ should be small at high frequencies for good high frequency noise attenuation

- $\bar{\sigma}\left[T_e(j\omega)\right]$ and $\bar{\sigma}\left[T_c(j\omega)\right]$ should not be too large in order for the closed loop system to be robust with respect to multiplicative modeling errors at the plant output.

Here, $\bar{\sigma}\left[M\right]$ denotes the maximum singular value of a matrix M.

4.3 Standard $\mathcal{H}^\infty$ Mixed-Sensitivity Optimization Problem

The Standard Weighted $\mathcal{H}^\infty$ mixed sensitivity optimization problem that addresses closed loop maps at plant output is as follows [147–156]:

$$K = \arg\left\{ \min_{K \ stabilizing} \gamma \ \left| \ \left\| \begin{bmatrix} W_1 S_e \\ W_2 K S_e \\ W_3 T_e \end{bmatrix} \right\|_{\mathcal{H}^\infty} < \gamma \right. \right\} \tag{4.1}$$

where W_1, W_2, W_3 are frequency-dependent weighting matrices that are used to trade-off the properties of S_e, KS_e, and T_e. The above Problem 4.1 can be solved using the two-Riccati formulae [157, 158] with loopshifting [159], or using LMI method [45, 160, 161]. The closed loop maps that are shaped (using corresponding weights) in the above Problem (4.1) are with respect to the feedback loop broken at plant output/error. Here, we note that when the plant to be controlled is ill-conditioned, one of the main difficulties is that obtaining good properties at one feedback loop-breaking point, does not guarantee acceptable properties at a different loop-breaking point [9, 10]. For a plant that is square (i.e., number of input/s equal to number of output/s) and assuming $P(j\omega)$ is invertible at frequencies of interest, the following

inequality gives upper bound on $\|S_c\|_{\mathcal{H}^\infty}$ using $\|S_e\|_{\mathcal{H}^\infty}$ and $\kappa[P]$.

$$\frac{1}{\kappa[P]}\sigma_i\left[S_e\right] \leq \sigma_i\left[S_c\right] \leq \kappa[P]\sigma_i\left[S_e\right] \tag{4.2}$$

If the condition number of the plant is high, achieving good feedback properties at plant output might still result in bad properties at plant input. Analogous results can be obtained as upper bound on $\|S_e\|_{\mathcal{H}^\infty}$. Hence, both S_e and S_c must be addressed during the design process based on the specifications. This issue is also discussed in Section 3.3.

Consider the following approach, where one uses d_i as an exogenous signal to a generalized plant to influence properties at the plant input.

$$K = \arg\{\min_{K \ stabilizing} \gamma \mid \|T_{wz}\|_{\mathcal{H}^\infty} < \gamma\} \tag{4.3}$$

$$T_{wz} = \begin{bmatrix} W_1\hat{S}_e & W_1PS_c \\ W_2K\hat{S}_e & W_2T_c \\ W_3\hat{T}_e & W_3PS_c \end{bmatrix} \tag{4.4}$$

While this approach is somewhat effective in impacting properties at the input, directly shaping the actual closed loop maps at distinct loop-breaking points is not straight forward. In what follows, we show how the proposed Generalized Mixed Sensitivity methodology can be used to achieve this. Further, we compare these control design approaches in Chapter 7.

4.4 Proposed Generalized $\mathcal{H}^\infty$ Mixed Sensitivity Optimization Problem

To address the problem of simultaneously addressing design requirements at multiple loop-breaking points, we formulate the following $\mathcal{H}^\infty/\mathcal{L}^\infty$-constrained multiobjective weighted $\mathcal{H}^\infty$ mixed sensitivity minimization problem:

$$K = \arg\left\{ \min_{\substack{K \\ stabilizing}} \gamma \;\middle|\; \begin{array}{c} \max\left(\|\mu_i T_{w_i z_i}\|_{\mathcal{H}^\infty}\right) < \gamma\ , \\ \mathcal{C}_j\left(T_{w_j z_j}\right) \leq c_j \end{array} \right\} \tag{4.5}$$

where $i, j = 1, 2, 3, ...,\ \mu_i \in [0, 1]$, $\mathcal{C}_j$ denotes j^{th} constraint funcional with $c_j \in \mathcal{R}$, $T_{w_i z_i}$ are weighted closed loop transfer function matrices with loop broken at distinct points (E.g. e, c/u and n_i for hierarchical inner-outer control architecture).

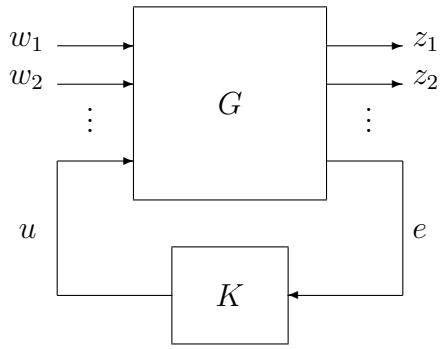


Figure 4.1: Visualization of Generalized Plant Setup for GMS Problem

4.4.1 GMS at Two Loop-Breaking Points for Standard P-K Feedback Structure

When we consider loop-breaking points at error (e) and controls (c), the Optimization Problem in Equation 4.5 reduces the following two-objective problem.

$$K = \arg \left\{ \min_{\substack{K \\ stabilizing}} \gamma \middle| \max \left(\rho \left\| \begin{bmatrix} W_1 S_e \\ W_2 K S_e \\ W_3 T_e \end{bmatrix} \right\|_{\mathcal{H}^\infty}, (1-\rho) \left\| \begin{bmatrix} W_4 S_c \\ W_5 P S_c \\ W_6 T_c \end{bmatrix} \right\|_{\mathcal{H}^\infty} \right) < \gamma \right\} \quad (4.6)$$

s.t. closed loop convex frequency/time-domain constraints.

$W_i \in R\mathcal{H}^\infty$ and $\rho \in [0, 1]$.

The above methodology captures the traditional mixed-sensitivity at the output problem that has been widely addressed within the controls literature [147–156] as well the not so broadly addressed mixed-sensitivity at the input problem [9, 10]. The former can be used to systematically achieve desirable properties at the output, while the

latter can be used to achieve desirable properties at the input. By combining the two as above, a designer can, by using the weighting functions, systematically shape and tradeoff properties simultaneously at both loop breaking points.

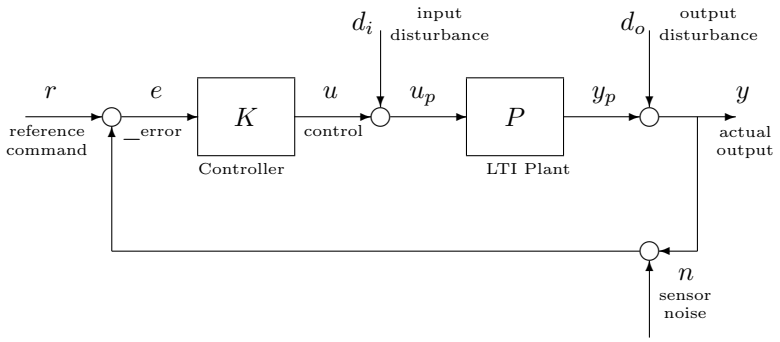


Figure 4.2: Standard P-K Feedback Structure

4.4.2 GMS at Three Loop-Breaking Points for Hierarchical Inner-Outer Loop Feedback Structure

Weighted closed loop maps $T_{w_i z_i}$ corresponding to the three loop breaking points for hierarchical inner-outer architecture (see Figure 4.3) can be formulated as,

$$T_{w_1 z_1} = \left[W_1 S_e{}^T \quad W_2 K S_e{}^T \quad W_3 T_e{}^T \right]^T \tag{4.7}$$

$$T_{w_2 z_2} = \left[W_4 S_c{}^T \quad W_5 P S_c{}^T \quad W_6 T_c{}^T \right]^T \tag{4.8}$$

$$T_{w_3 z_3} = \left[W_7 S_{n_i}{}^T \quad W_8 K S_{n_i}{}^T \quad W_9 T_{n_i}{}^T \right]^T \tag{4.9}$$

The first loop-breaking point corresponds to that at plant output/error ($w_1 = r$). For the case when unfiltered reference command is used, the closed loop maps of interest are $T_{re} = S_e$, $T_{ru} = KS_e$ and $T_{ry} = T_e$. The second loop-breaking point corresponds to that at plant input/control ($w_2 = d_i$). The closed loop maps of interest are $T_{d_i u_p} = S_c$, $T_{d_i y} = PS_c$ and $T_{d_i u} = T_c$. It is observed that when hierarchical inner-outer loop feedback structure is considered, using only the closed loop associated with

the above two loop-breaking points does not capture some other important closed loop maps (namely T_{n_iu} which is observed in many cases to be heavily dependend on frequency characteristics of inner-loop controller K_i). This is discussed in more detail in Chapter 7 in which the control design for the longitudinal dynamics 3-DOF hypersonics model is considered. It is important to note that the closed loop maps corresponding to the third loop-breaking point at inner-loop sensor noise ($w_3 = n_i$) are $T_{n_ie_i} = S_{n_i}$, $T_{n_iy} = KS_{n_iu}$ and $T_{n_iy} = T_{n_i}$. Note that, depending on the design requirements, the closed loop maps T_{n_ie} and $T_{n_iu_i}$ can also be shaped using our GMS methodology.

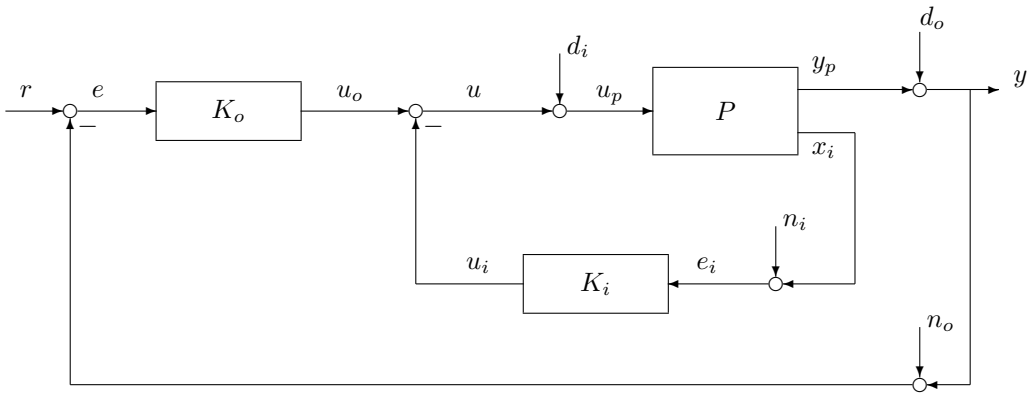


Figure 4.3: Hierarchical Inner-Outer Loop Feedback Structure

In Chapter 7, we use this hierarchical inner-outer loop feedback structure to design controllers for longitudinal dynamics of a 3-DOF hypersonics model. The $\mathcal{H}^\infty/\mathcal{L}^\infty$-constrained multiobjective weighted $\mathcal{H}^\infty$ mixed sensitivity minimization problem (see Equation 4.5)) presented is based on a weighted maximum formulation. This can be modified to use other formulations such as weighted sum and weighted stacking. These formulations are presented in Section 4.5. In many cases, the actual output of the plant (y: see Figure 4.3) depends directly on the extra measurement (x_i:

see Figure 4.3). In such case, the feedback structure may be redrawn as shown in Figure 4.4. Similar to the earlier case, three points loop-breaking become necessary to address all the closed-loop maps associated with the hierarchical inner-outer loop feedback structure. .

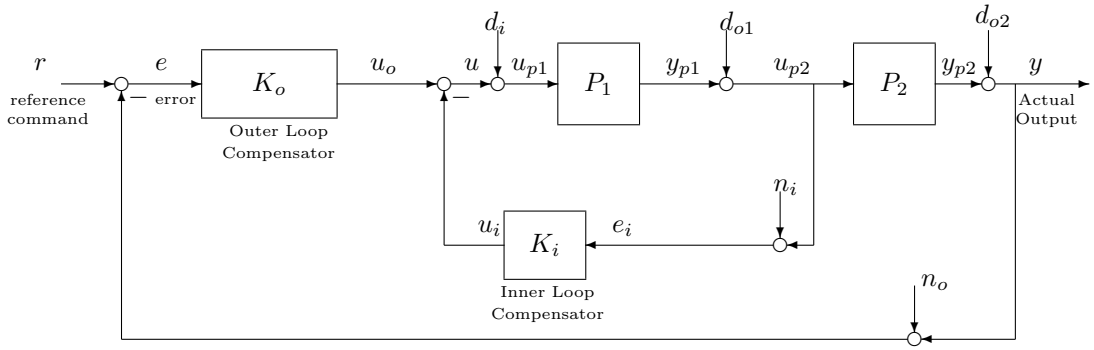


Figure 4.4: Hierarchical Inner-Outer Loop Feedback Structure

4.5 Different $\mathcal{H}^\infty$ based Multiobjective Function Formulations

Multiobjective optimization has been studied as a tool to tradeoff between specifications/requirements that are often conflicting [38–54]. In this section, we present three multiobjective function formulations that combine multiple functionals in different ways. To accurately specify the design objectives, especially in order to address specifications at distinct feedback loop-breaking points (e.g. output and input), the Generalized Mixed Sensitivity methodology accomodates three formulations [162–166]. The objective function can be specified in terms of:

1. Weighted Max
2. Weighted Stacking
3. Weighted Sum

It is important to note that in each of the above formulations, if the individual objective functions that make up the multiobjective function are convex, then the multiobjective function will be convex. This is proved by using the properties of convex "atom" functions (see Subsection 2.5.2 [59, 60] that talks about convexity preserving operations).

4.5.1 Weighted Max Formulation

The Weighted Max formulation, also called the *minimax design*, minimizes the weighted pointwise maximum of two or more objective functionals. Due to the simplicity of the formulation, selection of weighting functions is easy. The weighting functions are selected so as to shape the individual closed loop maps (each objective functional). Then the optimizer minimizes the worst of the objective functional at each iteration of the solution algorithm. The weighted max formulation for generalized mixed sensitivity when two loop-beaking points are considered is shown in Equation (4.10) [46, 47].

$$K = \arg\left\{ \min_{\substack{K \\ stabilizing}} \gamma \left| \max\left(\rho \left\| \begin{bmatrix} W_1 S_e \\ W_2 K S_e \\ W_3 T_e \end{bmatrix} \right\|_{\mathcal{H}^\infty}, (1-\rho) \left\| \begin{bmatrix} W_4 S_c \\ W_5 P S_c \\ W_6 T_c \end{bmatrix} \right\|_{\mathcal{H}^\infty} \right) < \gamma \right. \right\} \quad (4.10)$$

s.t. closed loop convex frequency/time-domain constraints.

$W_i \in R\mathcal{H}^\infty$ and $\rho \in [0, 1]$.

4.5.2 Stacking Formulation

In case of Weighted Stacking, since the $\mathcal{H}^\infty$-norm is taken after the two objectives are stacked, the convexity preservation results are not needed to see why the stacked objective function is convex. It is important to note that due to matrix dimensionality

issues stacking can only be done when the number of external inputs associated with each channel (w_i) are equal.

$$K = \arg \left\{ \min_{\substack{K \\ stabilizing}} \gamma \left\| \begin{bmatrix} \rho \begin{bmatrix} W_1 S_e \\ W_2 K S_e \\ W_3 T_e \end{bmatrix} \\ (1-\rho) \begin{bmatrix} W_4 S_c \\ W_5 P S_c \\ W_6 T_c \end{bmatrix} \end{bmatrix} \right\|_{\mathcal{H}^\infty} < \gamma \right\} \tag{4.11}$$

s.t. closed loop convex frequency/time-domain constraints.

$W_i \in R\mathcal{H}^\infty$ and $\rho \in [0, 1]$.

4.5.3 Sum Formulation

The Weighted Sum formulation, minimizes the weighted pointwise sum of two or more objective functionals. The optimizer minimizes the sum of all objective functionals at each iteration of the solution algorithm. The weighted sum formulation for generalized mixed sensitivity when two loop-beaking points are considered is shown in Equation (4.12).

$$K = \arg \left\{ \min_{\substack{K \\ stabilizing}} \gamma \left| \rho \left\| \begin{bmatrix} W_1 S_e \\ W_2 K S_e \\ W_3 T_e \end{bmatrix} \right\|_{\mathcal{H}^\infty} + (1-\rho) \left\| \begin{bmatrix} W_4 S_c \\ W_5 P S_c \\ W_6 T_c \end{bmatrix} \right\|_{\mathcal{H}^\infty} < \gamma \right\} \tag{4.12}$$

s.t. closed loop convex frequency/time-domain constraints.

$W_i \in R\mathcal{H}^\infty$ and $\rho \in [0, 1]$.

4.6 Summary and Conclusions

In this chapter, we presented the Generalized $\mathcal{H}^\infty$ Mixed Sensitivity methodology problem posed as a constrained multiobjective optimization problem. We also visited the three types of multiobjective functions - weighted max, weighted stacking and weighted sum - formulations that can be handled by our Generalized Mixed Sensitivity methodology. The problems that were formulated have the closed loop maps that depend nonlinearly (non-convex) on controller K. Using Youla parameterization (or Q-parameterization), the relationship can be transformed into affine dependence in the parameter Q. By doing this, we can transform the objective function and constraints ($\mathcal{H}^\infty$-based) into convex functions. This is discussed in detail in this Chapter 5.

Chapter 5

SOLUTION METHOD FOR THE GENERALIZED MIXED SENSITIVITY OPTIMIZATION PROBLEM

5.1 Overview

In the $\mathcal{H}^\infty/\mathcal{L}^\infty$-constrained multiobjective weighted $\mathcal{H}^\infty$ mixed sensitivity minimization problems that are formulated in Chapter 4, the closed loop maps depend nonlinearly on controller K (see closed loop map definitions in Section 3.2). Using Youla et al. parameterization (or Q-parameterization), the relationship can be transformed into affine dependence in the parameter Q. By doing this, we can transform the objective function and constraints ($\mathcal{H}^\infty/\mathcal{L}^\infty$-based) into convex functions. This is discussed in detail in this chapter. Further, this chapter discusses several subgradient-based constrained convex optimization algorithms and solvers, along with comparisons between them. Finally, the dependence of convergence rate on basis parameters used for Youla et al. parameterization are studied through examples.

5.2 Youla et al. (or Q) Parameterization of All Stabilizing Controllers

Youla et al.)(or Q) Parameterization [167–172] is used to parameterize the set of all stabilizing controllers for a given LTI plant. We show how this transforms the closed loop transfer function matrix $T_{wz}(K)$ that depends nonlinearly on controller K to an affine closed loop transfer function matrix $T_{wz}(Q)$ in the parameter Q. This transforms our ($\mathcal{H}^\infty/\mathcal{L}^\infty$-based) optimization problem (see Chapter 4) to a convex optimization problem - albeit infinite-dimensional. This infinite-dimensional problem will be approximated by a finite-dimensional one in Section 5.3. Within the current

section we focus on the Youla et al. (or Q) Parameterization for the set of all LTI compensators that stabilize the LTI plant.

Theorem 5.2.1 (Parameterizing the Set of All Stabilizing Controllers) *Given an LTI plant $P = [A, B, C, D]$, the set of all proper LTI controllers $S(P)$ that internally stabilize P may be parameterized as*

$$S(P) = \{K(Q) \mid Q \in \mathcal{H}^\infty\} \tag{5.1}$$

More specifically, if K_o internally stabilizes P, then there exists $Q_o \in \mathcal{H}^\infty$ such that $K_o = K(Q_o)$. Moreover $K(Q)$ internally stabilizes P for any given $Q \in \mathcal{H}^\infty$. ∎

5.2.1 Observer Based Youla et al. Parameterization

Consider a general feedback configuration of a finite-dimensional plant P, shown in Figure 5.1.

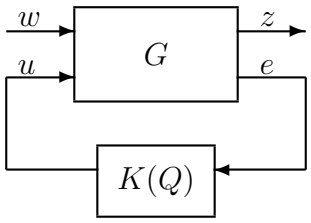


Figure 5.1: General System Interconnection

The transfer function from w to z is given by:

$$\left.\begin{array}{rcl} u &=& K(Q)e \\ z &=& G_{11}w + G_{12}u \\ &=& G_{11}w + G_{12}K(Q)e \\ e &=& G_{21}w + G_{22}u \\ &=& G_{21}w + G_{22}K(Q)e \\ &=& [I - K(Q)G_{22}]^{-1}G_{21}w \end{array}\right\} \Longrightarrow \; T_{wz} \;=\; G_{11} + G_{12}K(Q)[I - K(Q)G_{22}]^{-1}G_{21} \tag{5.2}$$

Let the generalized plant G shown in Figure 5.1 be given by

$$\left[\begin{array}{c} \dot{x} \\ \hline z \\ e \end{array}\right] = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right] \left[\begin{array}{c} x \\ \hline w \\ u \end{array}\right] \tag{5.3}$$

Note that the generalized plant G can include weighting functions augmented to the original plant to be controlled. The original plant P to be controlled can then be given by

$$P = \left[\begin{array}{c|c} A & B_2 \\ \hline C_2 & D_{22} \end{array}\right] \tag{5.4}$$

The parameterization $K(Q)$ may be constructed in terms of a model based compensator $K_o = [A - B_2F - L(C_2 - D_{22}F), L, -F]$ that stabilizes LTI plant P and a stable transfer function matrix Q ($Q \in \mathcal{H}^\infty$) with control gain matrix F, and the filter gain matrix L as in Figure 5.2 and Figure 5.3. Note that Figure 5.2 is a representation of Figure 5.1 to show the interconnection between K_o and Q.

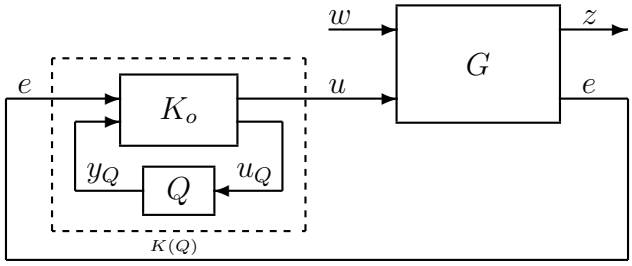


Figure 5.2: General System Interconnection with Q-Parameterization

where K_o is given by

$$\left[\begin{array}{c} \dot{\hat{x}} \\ u \\ u_Q \end{array}\right] = \left[\begin{array}{c|cc} A - B_2F - LC_2 + LD_{22}F & L & B_2 - LD_{22} \\ \hline -F & 0 & I \\ -(C_2 - D_{22}F) & I & -D_{22} \end{array}\right] \left[\begin{array}{c} \hat{x} \\ e \\ y_Q \end{array}\right] \tag{5.5}$$

$$y_Q = Qu_Q \tag{5.6}$$

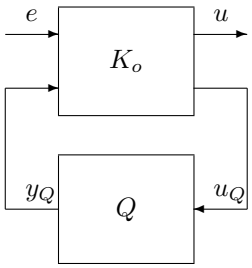


Figure 5.3: Visualization of Q Connected to an Observer-Based Controller

K_o has the following form

$$K_o = \left[\begin{array}{c|c} K_{11} & K_{12} \\ \hline K_{21} & K_{22} \end{array}\right] \tag{5.7}$$

where,

$$K_{11} = -F(sI - A + B_2F + LC_2 - LD_{22}F)^{-1}L \tag{5.8}$$

$$K_{12} = -F(sI - A + B_2F + LC_2 - LD_{22}F)^{-1}(B_2 - LD_{22}) + I \tag{5.9}$$

$$K_{21} = -(C_2 - D_{22}F)(sI - A + B_2F + LC_2 - LD_{22}F)^{-1}L + I \tag{5.10}$$

$$K_{22} = -(C_2 - D_{22}F)(sI - A + B_2F + LC_2 - LD_{22}F)^{-1}(B_2 - LD_{22}) - D_{22} \tag{5.11}$$

Hence, $K(Q)$ can be represented as

$$K(Q) = F_l(K_o, Q) = K_{11} + K_{12}Q[I - QK_{22}]^{-1}K_{21} \tag{5.12}$$

The observer-based structure of controller $K(Q)$ is shown in Figure 5.4

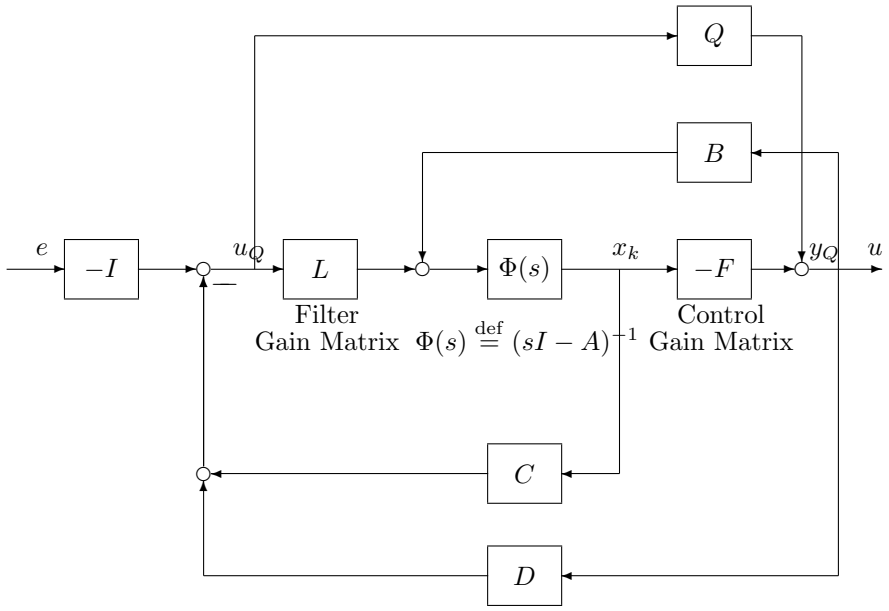


Figure 5.4: Observer Based Q-Parameterization for the Set of All Stabilizing LTI Controllers $K(Q)$

Achieving Convexity: Q-Parameterization for T_{wz} .

The closed loop system T_{wz} can be represented as shown in Figure 5.5 the system T is to be determined below. Note that the general transfer function matrices can be visualized as both T_{rz} and T_{d_iz}. Hence it is sufficient to show the affine relation between T and Q. The state space representation for T. With x denoting the states of F and x_k the states of K_{mbc}, we obtain the following

$$\dot{x} = Ax + BFx - BF(x - x_k) + Bw + B\hat{v} \tag{5.13}$$

$$\frac{d}{dt}(x - x_k) = (A + LC)(x - x_k) + (B + LD)w \tag{5.14}$$

$$z = (C + DF)x - DF(x - x_k) + Dw + D\hat{v} \tag{5.15}$$

$$v = C(x - x_k) + Dw \tag{5.16}$$

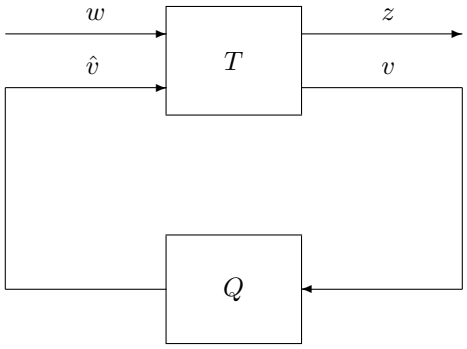


Figure 5.5: Visualization of the Closed Loop System T_{wz} in terms of T and Q

Given this, it follows that the system T can be expressed as follows:

$$T = \begin{bmatrix} T_1 & T_2 \\ T_3 & T_4 \end{bmatrix} = \left[\begin{array}{cc|cc} A+BF & -BF & B & B \\ 0 & A+LC & B+LD & 0 \\ \hline C+DF & -DF & D & D \\ 0 & C & D & 0 \end{array}\right]. \tag{5.17}$$

From Equation 5.17 it can be seen that

$$T_1 = \left[\begin{array}{cc|c} A-B_2F & B_2F & B_1 \\ 0 & A-LC_2 & B_1-LD_{21} \\ \hline C_1-D_{12}F & D_{12}F & D_{11} \end{array}\right], \qquad T_2 = \left[\begin{array}{c|c} A-B_2F & B_2 \\ \hline C_1-D_{12}F & D_{12} \end{array}\right],$$

$$T_3 = \left[\begin{array}{c|c} A-LC_2 & B_1-LD_{21} \\ \hline C_2 & D_{21} \end{array}\right], \qquad T_4 = 0.$$

Given the above, it follows that the closed loop transfer function matrix T is given by

$$T(Q) = F_l(T,Q) \tag{5.18}$$

$$= T_1 + T_2QT_3. \tag{5.19}$$

This shows that

- the closed loop transfer function matrix T_{wz} depends affinely on Q.
- our general control problem is convex in Q.

Given the above, the multiobjective optimization is formulated by selecting the input and output channels of interest. The closed loop TFMs ($T_{w_i z_i},\ i = 1, 2, 3, \dots$) to be shaped by the GMS optimizer (see GMS problem in Equation (4.5)).

5.2.2 Coprime Factorization Based Youla et al. Parameterization

Theorem 5.2.2 *Given an LTI plant $P = [A, B, C, D]$, the set of all proper LTI controllers that internally stabilize P may be parameterized as [149, 170–172]:*

$$K(Q) = (N_k - D_p Q)(D_k - N_p Q)^{-1} \tag{5.20}$$

where

$$N_p = \left[\begin{array}{c|c} A - BF & B \\ \hline C - DF & D \end{array}\right] \qquad D_p = \left[\begin{array}{c|c} A - BF & B \\ \hline -F & I \end{array}\right] \tag{5.21}$$

$$N_k = -\left[\begin{array}{c|c} A - BF & L \\ \hline -F & 0 \end{array}\right] \qquad D_k = \left[\begin{array}{c|c} A - BF & L \\ \hline C - DF & I \end{array}\right] \tag{5.22}$$

∎

where F is control gain matrix and L is filter gain matrix that can be obtained by using the model based compensator configuration. It should be noted that $K_o = N_k D_k^{-1}$ represents one strictly proper LTI compensator that internally stabilizes P.
$K(Q)$ may also be parameterized as,

$$K(Q) = (\tilde{D}_k - Q\tilde{N}_p)^{-1}(\tilde{N}_k - Q\tilde{D}_p) \tag{5.23}$$

where

$$\tilde{N}_p = \left[\begin{array}{c|c} A-LC & B-LD \\ \hline C & D \end{array}\right] \qquad \tilde{D}_p = \left[\begin{array}{c|c} A-LC & -L \\ \hline C & I \end{array}\right] \tag{5.24}$$

$$\tilde{N}_k = -\left[\begin{array}{c|c} A-LC & L \\ \hline -F & 0 \end{array}\right] \qquad \tilde{D}_k = \left[\begin{array}{c|c} A-LC & -(B-LD) \\ \hline F & I \end{array}\right] \tag{5.25}$$

$\tilde{K}_o = \tilde{D}_k{}^{-1}\tilde{N}_k$ represents one strictly proper LTI compensator that internally stabilizes P.

Achieving affiness:

Consider a unity (positive) feedback loop with compensator $K(Q)$ in series with P. The closed-loop transfer function matrices can be parameterized as follows:

$$\begin{aligned}
S_e(Q) &= [I + PK(Q)]^{-1} && (5.26)\\
&= [D_k - N_p Q]\,\tilde{D}_p && (5.27)\\
S_c(Q) &= [I + K(Q)P]^{-1} && (5.28)\\
&= D_p\left[\tilde{D}_k - Q\tilde{N}_p\right] && (5.29)\\
K(Q)S_e(Q) &= S_c(Q)K(Q) && (5.30)\\
&= D_p[\tilde{N}_k + Q\tilde{D}_p] && (5.31)\\
PS_c(Q) &= N_p[\tilde{D}_k - Q\tilde{N}_p] && (5.32)\\
&= [D_k - N_p Q]\tilde{N}_p && (5.33)\\
T_e(Q) &= I - S_e(Q) && (5.34)\\
&= N_p[\tilde{N}_k + Q\tilde{D}_p] && (5.35)\\
T_c(Q) &= I - S_c(Q) && (5.36)\\
&= [I - D_p\tilde{D}_k] + D_p Q\tilde{N}_k && (5.37)
\end{aligned}$$

Note that all the closed loop transfer function matrices can be represented as

$$T(Q) = F_l(T, Q) \tag{5.38}$$

$$= T_1 + T_2 Q T_3 \tag{5.39}$$

Given the above, the multiobjective optimization is formulated by selecting the input and output channels of interest. The closed loop TFMs ($T_{w_i z_i},\ i = 1, 2, 3, \dots$) to be shaped by the GMS optimizer (see GMS problem in Equation (4.5)).

5.2.3 Controller State Space Representation

Once we obtain the Q-parameter, it is easy to obtain $K(Q)$. If the state space representation of Q is as below,

$$Q \stackrel{\text{def}}{=} \left[\begin{array}{c|c} A_Q & B_Q \\ \hline C_Q & D_Q \end{array}\right] \tag{5.40}$$

then, this yields the following state space representation for the controller $K(Q)$:

$$K(Q) = \left[\begin{array}{c|c} A_K & B_K \\ \hline C_K & D_K \end{array}\right]. \tag{5.41}$$

where,

$$A_K = \begin{bmatrix} (A - LC) - (B - LD)\Delta^{-1}(D_Q C + F) & (B - LD)\Delta^{-1} C_Q \\ -B_Q C + B_Q D \Delta^{-1}(D_Q C + F) & A_Q - B_Q D \Delta^{-1} C_Q \end{bmatrix}, \tag{5.42}$$

$$B_K = \begin{bmatrix} L + (B - LD)\Delta^{-1} D_Q \\ B_Q - B_Q D \Delta^{-1} D_Q \end{bmatrix}, \tag{5.43}$$

$$C_K = \begin{bmatrix} -\Delta^{-1}(D_Q C + F) & \Delta^{-1} C_Q \end{bmatrix}, \tag{5.44}$$

$$D_K = \Delta^{-1} D_Q \qquad \text{for} \qquad \Delta = (I + D_Q D). \tag{5.45}$$

If $D = 0$

$$\left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right] = K(Q) = \left[\begin{array}{cc|c} A - BF - LC - BD_QC & BC_Q & L + BD_Q \\ -B_QC & A_Q & B_Q \\ \hline -(D_QC + F) & C_Q & D_Q \end{array}\right] \quad (5.46)$$

If $D = D_Q = 0$

$$K(Q) = \left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right] = \left[\begin{array}{cc|c} A - BF - LC & BC_Q & L \\ -B_QC & A_Q & B_Q \\ \hline -F & C_Q & 0 \end{array}\right]. \quad (5.47)$$

It should be noted that the Youla parameterization is constructed from a nominal controller defined by $Q = 0$. This nominal controller - and hence the Youla parametrization - is defined by the control gain matrix F, and the filter gain matrix L. F and L are not unique.

From the above discussion, it is clear that the optimization problem (nonlinear) over stabilizing controllers K can instead be solved by optimizing over all stable transfer function matrices Q ($Q \in \mathcal{RH}^\infty$). As such, we still have an infinite-dimensional problem. This problem can be transformed to a finite-dimensional problem if Q is appropriately approximated. How this is done is now shown.

5.3 Achieving Finite Dimensionality

Approximation ideas are used to approximate the parameter Q and transform the inifinite-dimensional problem to a finite-dimensional one for which efficient algorithms exist. To do this, express the Q-parameter is expressed as a finite linear combination of *a priori* selected stable transfer functions q_k; i.e.

$$Q_N = \sum_{k=1}^{N} X_k q_k \quad (5.48)$$

where

$$X_k = \begin{bmatrix} x_k^{11} & \cdots & x_k^{1n_e} \\ \vdots & & \vdots \\ x_k^{n_u 1} & \cdots & x_k^{n_u n_e} \end{bmatrix} \in \mathcal{R}^{n_u \times n_e} \tag{5.49}$$

Here, n_u and n_e are the number of inputs and outputs of the plant respectively. Substituting Q_N into 5.19, then yields the following structure for T_{wz}:

$$T_{wz} = T_1 + T_2 \left(\sum_{k=1}^{N} X_k \, q_k \right) T_3 \tag{5.50}$$

$$= T_1 + \sum_{k=1}^{N} T_2 X_k T_3 \, q_k \tag{5.51}$$

Next, we note that X_k may be written as follows:

$$X_k = \sum_{j=1}^{n_e} \sum_{i=1}^{n_u} B^{ij} x_k^{ij} \tag{5.52}$$

where $B^{ij} \in \mathcal{R}^{n_u \times n_e}$ is a matrix with its ij^{th} entry equal to 1 and all other elements zero. Note that the above sum is carried out over rows first and then columns. By so doing, we "vectorize" the problem. Substituting the above expression for X_k into T_{wz} the yields

$$T = T_1 + \sum_{k=1}^{N} T_2 \left(\sum_{j=1}^{n_e} \sum_{i=1}^{n_u} B^{ij} x_k^{ij} \right) T_3 \, q_k \tag{5.53}$$

$$= T_1 + \sum_{k=1}^{N} \sum_{j=1}^{n_e} \sum_{i=1}^{n_u} T_2 \, B^{ij} \, x_k^{ij} \, T_3 \, q_k \tag{5.54}$$

$$= T_1 + \sum_{k=1}^{N} \sum_{j=1}^{n_e} \sum_{i=1}^{n_u} T_2 \, B^{ij} \, T_3 \, q_k \, x_k^{ij}. \tag{5.55}$$

This expression may be written as

$$T = M_o + \sum_{k=1}^{N} \sum_{j=1}^{n_e} \sum_{i=1}^{n_u} M_k^{ij} \, x_k^{ij} \tag{5.56}$$

where

$$M_o = T_1 \tag{5.57}$$

$$M_k^{ij} = T_2\, B^{ij}\, T_3\, q_k. \tag{5.58}$$

Finally, to complete the vectorization of the problem we define a new indexing variable $l = (k-1)n_e + j - 1 + i$ where we sequence over i, and then j, and then k. Defining the scalar x_l and the matrix M_l, we have the following bijective mapping:

$$x_l = x_k^{ij} \tag{5.59}$$

$$M_l = M_k^{ij}. \tag{5.60}$$

With this definition, our expresssion for T_{wz} becomes

$$T = M_o + \sum_{l=1}^{n_u \times n_e \times N} M_l\, x_l. \tag{5.61}$$

From this expression, it follows that T depends affinely on the elements $x_l = x_k^{ij}$. Our general control system design problem has thus been transformed to a finite-dimensional convex optimization in the scalar elements $x_l = x_k^{ij}$.

What makes the approach taken in this chapter very appealing is the fact that many control system design specifications may be posed as convex constraints on the closed loop transfer function matrix [59]. Because the closed loop transfer function matrix is convex in the Youla Q-Parameter, it follows that convex constraint may be incorporated into a convex optimization problem involving Q.

Order of the Controller $K(Q_N)$. When coprime-factorization approach is used to parameterize the set of all stabilizing controllers, the order of the controller (n_k) is given by

$$n_k = n + n_p \tag{5.62}$$

where n_p is the number of states of the generalized plant, and nq is the number of states of QN

5.4 Basis Options

In this work, the following basis options [173–177] have been used:

1. Fixed pole low-pass

$$q_k = \left(\frac{p}{s+p}\right)^{k-1} \tag{5.63}$$

2. Fixed pole all-pass

$$q_k = \left(\frac{p-s}{s+p}\right)^{k-1} \tag{5.64}$$

3. Variable pole low-pass (first order)

$$q_1 = 1, \;\; q_{k+1} = \frac{kp}{s+kp} \tag{5.65}$$

4. Variable pole all-pass

$$q_k = \frac{(k-1)p-s}{s+(k-1)p} \tag{5.66}$$

5. Fixed pole fixed zero

$$q_k = \left(\frac{z-s}{s+p}\right)^{k-1} \tag{5.67}$$

6. Laguerre

$$q_k = \frac{\sqrt{2\alpha}}{s+\alpha}\left(\frac{\alpha-s}{s+\alpha}\right)^{k-1} \tag{5.68}$$

where $k = 1, 2, ..., N$ and $p, z \in \mathcal{R}^+$.

5.5 Computation of Subgradients

5.5.1 *Subgradient for $\mathcal{H}^\infty$ Norm at a Transfer Function Matrix*

In what follows, we will be interested in minimizing the $\mathcal{H}^\infty$ norm of a transfer function matrix. This is useful for control system system design. Given this, we consider the function

$$\phi \;:\; \mathcal{H}^\infty \longrightarrow \mathcal{R}_+ \tag{5.69}$$

$$:\; M \longrightarrow \phi(M) \stackrel{\text{def}}{=} \|M\|_{\mathcal{H}^\infty} \tag{5.70}$$

which maps transfer function matrices $M \in \mathcal{H}^\infty$ (i.e. stable systems) to non-negative real numbers.

The function ϕ is a non-differentiable function. Finding a derviative for ϕ at a transfer function matrix is therefore not possible. While this is undesirable, it should be noted that subgradients for the $\mathcal{H}^\infty$ norm function at a transfer function matrix are easy to find. This is shown in [59, 178, 179].

Subgradient for ϕ at M. Suppose that we have a transfer function matrix given by

$$M = \sum_{k=1}^{n} M_k x_k \tag{5.71}$$

where $\{M_k\}_{k=1}^n$ is a sequence of $\mathcal{H}^\infty$ transfer function matrices (i.e. stable systems). We wish to determine a subgradient for ϕ at M. In [59, 173], the authors provide an answer to this. A subgradient for ϕ at M, denoted $g_\phi(M) \in \partial_\phi(M)$, is given by the

n-dimensional vector

$$g_\phi(M) = \begin{bmatrix} \phi^{sg}(M_1) \\ \phi^{sg}(M_2) \\ \vdots \\ \phi^{sg}(M_n) \end{bmatrix} \tag{5.72}$$

$$\phi^{sg}(M_k) = Re\left(u_o^H M_k(j\omega_o) v_o\right) \tag{5.73}$$

where u_o and v_o are the left and right singular vectors of $M(j\omega_o)$ corresponding to the maximum singular value σ_{max} at the peak frequency ω_o at which the $\mathcal{H}^\infty$ norm of M is achieved; i.e. $\|M\|_{\mathcal{H}^\infty} = \max_{\omega \geq 0} \sigma_{max}[\, M(j\omega)\,] = \sigma_{max}[\, M(j\omega_o)\,]$.

Interpretation of Subgradient. For SISO systems, we have

$$\phi^{sg}(M_k) = Re\left(u_o^H M_k(j\omega_o) v_o\right) = |M_k(j\omega_o)|. \tag{5.74}$$

That is, for SISO systems, the k^{th} component of $g_\phi(M)$ is the value of $|M_k(j\omega)|$ at the peak frequency ω_o of M.

Significance. Just as gradient information for a differentiable function permits one to construct an affine tangent approximation, the subgradient $g_\phi(M)$ provides us with the following affine lower bound approximation for the convex function ϕ:

$$\phi(M_o) + g_\phi(M - M_o) \leq \phi(M) \tag{5.75}$$

for any $\mathcal{H}^\infty$ (i.e. stable) transfer function matrices M and M_o.

Special Case. In what follows, we will apply the above ideas to

$$\phi(T_{wz}^*) = \|T_{wz}^*\|_{\mathcal{H}^\infty} \tag{5.76}$$

where T_{wz}^* takes the form

$$T_{wz}^* = M^* = \sum_{l=1}^{n_u \times n_e \times N} M_l^* x_l. \tag{5.77}$$

Given this, it follows that

$$g = \begin{bmatrix} \phi^{sg}(M_1^*) \\ \phi^{sg}(M_2^*) \\ \vdots \\ \phi^{sg}(M_{n_u \times n_e \times N}^*) \end{bmatrix} \in \partial_\phi(M^*) \tag{5.78}$$

where M_l^*, n_u, n_e, and N are defined in Section 5.3 and

$$\phi^{sg}(M_k) = Re\left(u_o^H M_k^*(j\omega_o) v_o\right) \tag{5.79}$$

where u_o and v_o are the left and right singular vectors of $M^*(j\omega_o)$ corresponding to the maximum singular value σ_{max} at the peak frequency ω_o at which the $\mathcal{H}^\infty$ norm of $T_{wz}^* = M^*$ is achieved; i.e. $\|T_{wz}^*\|_{\mathcal{H}^\infty} = \|M^*\|_{\mathcal{H}^\infty} = \max_{\omega \geq 0} \sigma_{max}[\, M^*(j\omega)\,] = \sigma_{max}[\, M^*(j\omega_o)\,]$.

5.5.2 Subgradient for Time-Domain $\mathcal{L}^\infty$ Norm at a TFM

In what follows, we will be concerned with meeting peak time response specifications (to step input). Given this, we consider the function

$$\phi \;:\; \mathcal{H}^\infty \longrightarrow \mathcal{R}_+ \tag{5.80}$$

$$:\; M \longrightarrow \phi(M) \stackrel{\text{def}}{=} \sup_{t \geq 0} s^{step,t}(M) - 1 \tag{5.81}$$

which maps transfer function matrices $M \in \mathcal{H}^\infty$ (i.e. stable systems) to non-negative real numbers. Here, $s^{step,t}(M)$ denotes the unit step response of a stable system.

The function ϕ is a non-differentiable function. Finding a derviative for ϕ at a transfer function matrix is therefore not possible. While this is undesirable, it should be noted that subgradients for the step response overshoot function at a transfer function matrix are easy to find. This is shown in [59, 180, 181]. In this example, we show how to compute such subgradients.

Subgradient for ϕ at M. Suppose that we have a transfer function matrix given by

$$M = \sum_{k=1}^{n} M_k x_k \tag{5.82}$$

where $\{M_k\}_{k=1}^{n}$ is a sequence of $\mathcal{H}^\infty$ transfer function matrices (i.e. stable systems). We wish to determine a subgradient for ϕ at M. In [59], the authors provide an answer to this. A subgradient for ϕ at M, denoted $g_\phi(M) \in \partial_\phi(M)$, is given by the n-dimensional vector

$$g_\phi(M) = \begin{bmatrix} \phi^{sg}(M_1) \\ \phi^{sg}(M_2) \\ \vdots \\ \phi^{sg}(M_n) \end{bmatrix} \tag{5.83}$$

$$\phi^{sg}(M_k) = \phi^{step,t_o}(M_k) \tag{5.84}$$

$$= \text{value of step response of} M_k \text{ at } t_o. \tag{5.85}$$

where t_o denotes the time that the step response of M reaches its peak. The above shows that the k^{th} component of $g_\phi(M)$ is the value of the step response of M_k at t_o - the time at which the step response of M achieves its peak overshoot.

5.5.3 Subgradient of Multiobjective Functions

This subsection discusses important properties of subgradients that are made use of for solving our non-differentiable multiobjective functions. Thus far in this Section 5.5, subgradient computation for single objective function is presented. Here, we show how subgradients can be computed for multiobjective functions. These make use of the detailed discussions on the theory of subgradients in the works by Nesterov and Boyd et al. in [182, 183].

Nonnegative scaling: Suppose f is a non-differentiable convex function and $\mu \in \mathcal{R}_+$.

Then the subgradient of the scaled function (μf) is given by

$$g_\phi(\mu f) = \mu \; g_\phi(f) \tag{5.86}$$

Sum: Suppose f_1 and f_2 are non-differentiable convex functions. Then the subgradient of the sum function $(f = f_1 + f_2)$ is given by

$$g_\phi(f) = \; g_\phi(f_1) + \; g_\phi(f_2) \tag{5.87}$$

Pointwise maximum: Suppose f_1 and f_2 are non-differentiable convex functions. Let $k \in \{1, 2\}$ be the index for which $f_k = f$ at a given point. Then $g_\phi(f_k)$ is a subgradient of f at that point. Note that the subgradient-based solvers (see Section 5.6) we use require any one of the subgradient of the objective function in each iteration. Given this, the above subgradient computation method is sufficient to provide subgradient information required by the solvers. More general way to compute (obtain all) subgradients at a given point can be found in [182, 183].

The above discussion on sum and pointwise maximum cases was done using two objective functions. But the results can be extended to arbitrary number of simultaneous objective functions.

5.6 Convex Optimization Methods

In this section discusses three convex optimization solvers used within the Generalized Mixed Sensitivity (GMS) framework.

1. Analytic Center Cutting Plane Method (ACCPM) solver
2. Kelley's Cutting Plane Method (Kelley's CPM) solver
3. SOLver for local OPTimization (SolvOpt)

The multiobjective optimization problem to be solved involves non-differentiable, but continuous convex objectives and functions [173, 184–187]. Nondifferentiability means that the gradient information is not available at all points of the function. To tackle these problems, many subgradient-based methods were developed [188, 189]. These are iterative techniques where each iterate is updated using a current subgradient and a chosen step size. It is shown that a constant step size in each iteration does not always converge [29, 188]. A convergent step-size satisfies $\sum_{k=0}^{\infty} t_k = \infty,\ t_k \to 0$, where t_k is step size involved in a iterate of the form $x_{k+1} = x_k + t_k g_k$, where x_k is current point, and g_k is subgradient (at x_k) of the function to be minimized. These ideas are incorporated within the above solvers.

5.6.1 Overview of Interior Point and Cutting Plane Methods

We first focus on two general classes of optimization methods that can use subgradient information [59] of the objective/constraint functions to solve the optimization problem.

1. Interior Point Methods (IPMs), and
2. Cutting Plane Methods (CPMs).

Terminology. To help describe each method, it is useful to consider the following convex optimization problem. We assume that $f_o, f_1, \ldots, f_m$ are convex functions mapping $\mathcal{R}^n \to \mathcal{R}$.

$$\begin{array}{ll} \min\limits_{x \in \mathcal{R}^n} & f_o(x) \\ \text{subject to} & f_i(x) \leq 0, \ \ i = 1, \ldots, m \end{array} \tag{5.88}$$

The function f_o is called the *objective function*. The functions $f_1, \ldots, f_m$ are *constraint functions*. The *feasible set* or *constraint set* defined by the convex constraint functions

$f_1, \ldots, f_m$ is denoted as follows:

$$\mathcal{F} \stackrel{\text{def}}{=} \{\ x \in \mathcal{R}^n \ \mid \ f_i(x) \leq 0, \ \ i = 1, \ldots, m\ \} \tag{5.89}$$

The interior of $\mathcal{F}$ is denoted $int(\mathcal{F})$ and is defined as follows:

$$\text{int}\,(\mathcal{F}) \stackrel{\text{def}}{=} \{\ x \in \mathcal{R}^n \ \mid \ f_i(x) < 0, \ \ i = 1, \ldots, m\ \}. \tag{5.90}$$

The boundary of $\mathcal{F}$ is denoted $\partial\mathcal{F}$ and defined as follows:

$$\partial\mathcal{F} \stackrel{\text{def}}{=} \{\ x \in \mathcal{R}^n \ \mid \ f_i(x) = 0, \ \ i = 1, \ldots, m\ \}. \tag{5.91}$$

It should be noted that $\mathcal{F}$, $int(\mathcal{F})$, and $\partial\mathcal{F}$ are convex sets. For the optimization problem in (5.88), when a minimizer exits, we denote the minimum by f_o^* denotes the minimum, and the minimizer by x^*.

Interior Point Methods (IPMs)

nterior Point Methods (IPMs) are characterized by the property that at each iteration the method generates a point that lies within (interior to) the feasible set (i.e., $x_k \in \text{int}\,(\mathcal{F})$). A popular IPM nown as Analytic Center Method (or Method of Centers) is now described. The analytic center of a set of inequalities $f_i(x) < 0$ $(i = 1, \ldots, m)$, is defined as

$$x_{ac} \stackrel{\text{def}}{=} \arg\left(\max_{x \in int(\mathcal{F})} \prod_{i=1}^{m} -f_i(x)\right) \tag{5.92}$$

The analytic center defines a "center" $x_{ac} \in int(\mathcal{F})$ for the feasible set $\mathcal{F}$. To solve the constrained convex problem (5.88), the constrained problem is replaced by a sequence of unconstrained convex optimization problems. Each unconstrained problem is solved by finding $x_{ac}^*(t)$, which denotes the analytic center of the inequalities

$$f_o(x) < t, \ \ f_i(x) < 0, \quad for\ all \quad i = 1, \ldots, m \ \ and \ \ t \in \mathcal{R}. \tag{5.93}$$

From (5.92), $x^*_{ac}(t)$ is given by

$$x^*_{ac}(t) = \arg\left(\max_{x\in int(\mathcal{F}), f_o(x)<t}(t - f_o(x))\prod_{i=1}^{m} -f_i(x)\right) \quad (5.94)$$

$$= \arg\left(\min_x F_t(x)\right) \quad (5.95)$$

If $t > f^*_o$ and the set $\{\ x \mid f_o(x) < t, \quad f_i(x) \leq 0 \quad i = 1, 2, \ldots, m\ \}$ is bounded, then $x_{ac}(t)$ exists and $0 \leq f_o(x^*_{ac}(t)) - f^*_o \leq m\ [\, t - f_o(x^*_{ac}(t))\,]$. Moreover, $\lim_{t\to f^*_o} f(x^*_{ac}(t)) = f^*_o$.

Analytic Center Method (ACM) Algorithm. The ACM algorithm can be implemented using Sequential Unconstrained Minimization Technique (SUMT) algorithm as described belwo.

Input: strictly feasible $x^{(0)}$*, tolerance* $\epsilon > 0$*,*

initial upper bound $t^{(0)} > f_o(x^{(0)})$*, and* $0 < \theta < 1$.

Output: a feasible x *such that* $\mid f^*_o - f_o(x) \mid < \epsilon$.

Assumption: The feasible set intersected with $\{\ x \mid f_o(x) < t\ \}$ *is bounded.*

Begin

$x := x^{(0)}$, $t := t^{(0)}$ *(Initialization)*

repeat{ *(Minimizer Update Loop)*

repeat{ *(Newton-Line Search Loop)*

$v := -\left[\nabla^2 F_t(x)\right]^{-1}\ \nabla F_t(x)$; *(Newton Direction)*

$\delta^* := \arg(\ \min_\delta \nabla F(x + \delta v)\ )$; *(Line Search)*

$x := x + \delta^* v$; *(Minimizer Update)*

}*until* $\|v\|$ *very small* *(Newton Stopping Criteria)*

return if $m(t - f_o(x)) < \epsilon$ *(Stopping Criteria)*

$t := (1 - \theta) f_o(x) + \theta t$ *(Minimum Update)*

}

end.

A draw back of this algorithm is that the initial upper bound $t^{(0)}$ must always be chosen larger than the value of $f_o(x^{(0)})$. This value may be very conservative - resulting in higher number of iterations to arrive at the solution.

Pros and Cons of Interior Point Methods (IPMs)

- *Pros:*

    1. *Speed.* IPMs are fast, especially for large problems. They have been proven to be polynomial-time for linear programming (LP) problems [190]. This can be extended to nonlinear programming problems [191].
    2. *Size of Problem/Complexity.* The iterates in IPMs do not grow fast in complexity compared to CPMs.

- *Cons:*

    1. *Information Required.* Most IPMs rely on the use of descent methods. This requires additional effort for non-differentiable functions, which is the case in our GMS optimization problems.
    2. *Ease of Coding.* IPM algorithms may be difficult to code. This may be particularly true for nonlinear problems requiring polynomial time performance [191]. This, in part, can be attributed to increased theoretical overhead (e.g. constructing a barrier function that is self-concordant [191]).
    3. *Ease of Use.* Performance of IPM algorithms can be very sensitive to associated algorithm parameter(s). Fortunately, they are not difficult to tune.

Cutting Plane Methods (CPMs)

Cutting Plane Methods (CPMs) were proposed independently by Kelley [192] and Cheney and Goldstein [193] as a solution technique for the constrained convex optimization problem in (5.88). These are one of the first and fundamental solution methods that were developed to solve non-differentiable convex optimization problems. CPMs rely on polyhedral approximations of convex functions to perform the optimization. The CPMs in most cases are observed to be much slower compared to IPMs.

Pros and Cons of Cutting Plane Methods (CPMs)

- *Pros*

    1. *Information Required.* CPMs only requires one subgradient at each iteration. No additional effort is needed for nondifferentiable functions [29].
    2. *Ease of Use.* CPM's are easy to use - since they typically involve less parameters.
    3. *Ease of Coding.* CPMs are easy to code and understand [29].

- *Cons*

    1. *Speed.* Most CPMs are slow, but there are new advanced methods that overcome this problem [194].
    2. *Information Required.* Must specify an initial box which contains a minimizer (however this box can be as large as you want).
    3. *Complexity.* The associated linear program grows linearly in size with iterations. As such, the complexity is non-polynomial.

5.6.2 Analytic Center Cutting Plane Method (ACCPM)

In this section, we describe the *Analytic Center Cutting Plane Method (ACCPM).* ACCPM was developed by Goffin et al. [195] and analyzed in [196–198]. We choose the ACCPM as our main method for solving convex control system design problems because ACCPM combines the simplicity of a cutting plane method (CPM) with the efficiency of an interior point method (IPM) [196–198]. Atkinson and Vaidya [197] showed that ACCPM exhibits polynomial time behavior under certain adaptation. Kelley's cutting plane method computes the minimizer and the query point by solving a linear programming (LP) problem. ACCPM uses the analytic center of the localization set as its query point. The method is described below [23–32]:

ACCPM Algorithm is as follows:

Input: initial point x_1*, initial box* $(x_{min}, x_{max}) \leftarrow$ *any initial box that contains a minimizer*

Output: an approximate minimizer x *such that* x *is feasible and* $f_o(x) - L_k \leq \epsilon_{obj}$

Assumption: x_1 *is feasible*

Begin

$L_0 := -\infty$

$U_0 := f_o(x_1)$

$k := 0$

while $f_1(x_{k+1}) > 0$ *or* $\ldots$ *or* $f_m(x_{k+1}) > 0$ *or* $U_k - L_k > \epsilon_{obj}$

{

$k := k + 1$

compute the analytic center (x_k^a, L_k^a) *of the localization set* $\mathcal{F}_{U_k}$ *as in (5.102)*

If x_k^a *is feasible*{

compute optimality cut and the upper bound $U_k := \min(f_o(x_k^a), U_{k-1})$

}*else*{

compute feasibility cut

}

Add cut to the localization set defined in (5.100)

$x_{k+1} := x_k^a$ *(Update Minimizer)*

$L_k := \max(L_k^a, L_{k-1})$ *(Update Lower Bound)*

}

end.

ACCPM is different from Kelley's CPM in that ACCPM uses the notion of localization set and index sets to describe the relaxed problem. Further, ACCPM uses a cutting plane algorithm that generates optimality cuts for feasible query points and feasibility cuts for infeasible query points, which allow us to consider only feasible points for the stopping criteria.

Consider the constrained convex problem (5.88), where $f_o(x), f_1(x), \ldots, f_m(x)$ are non-smooth convex functions. Since the objective function f_o is convex, subgradients may be used to generate piecewise affine approximations as follows:

$$f_o(x) \simeq \max_{i \in I} \{\ f_o(x_i) + g_{o_i}(x - x_i)\ \} \tag{5.96}$$

where I is an index set and g_{o_i} is a subgradient of f_o at x_i. Larger sets I, in principle, will yield better piecewise affine approximations. Since the constraint functions $\{f_i\}_{i=1}^m$ are convex, subgradients may be used to generate piecewise affine approximations as follows:

$$f_k(x) \simeq \max_{j \in J} \{\ f_k(x_j) + g_{k_j}(x - x_j)\ \} \quad k = 1, \ldots, m \tag{5.97}$$

where J is an index set and g_{k_j} is a subgradient of $f_k(x)$ at x_j.

Given the above, the optimization problem in (5.88) can be approximated as follows:

$$\begin{array}{cc} \min & f_o(x) + g_{o_i}(x - x_i). \\ \max_{j \in J} f_1(x_j) + g_{1_j}(x - x_j) \leq 0 & \\ \vdots & \\ \max_{j \in J} f_m(x_j) + g_{m_j}(x - x_j) \leq 0 & \\ x_{min} - x \leq 0 & \\ x - x_{max} \leq 0 & \end{array} \tag{5.98}$$

The above affine approximation can be formulated as an LP. This is done as follows:

$$\begin{array}{cc} \min & L \\ f_o(x_i) + g_{o_i}(x - x_i) \leq L, \ \forall \ i \in I & \\ f_1(x_j) + g_{1_j}(x - x_j) \leq 0 & \\ \vdots & , \ \forall \ j \in J \\ f_m(x_j) + g_{m_j}(x - x_j) \leq 0 & \\ x_{min} - x \leq 0 & \\ x - x_{max} \leq 0 & \end{array} \tag{5.99}$$

We refer to this as the relaxed problem (5.99) with index sets I_k and J_k. This relaxation gets tighter as more points are included in the index sets I and J.

At each iteration of the ACCPM, a bounded polyhedral set (the localization set) is updated. Since an upper bound U_k to to the relaxed problem can be achieved through the evaluation of the objective function at feasible query points (as in Kelley's CPM),

the localization set is given by

$$\mathcal{F}_{U_k} \stackrel{\text{def}}{=} \left\{ (x, L) \in \mathcal{R}^n \times \mathcal{R} \;\middle|\; \begin{array}{l} L \leq U_k \\ \hline f_o(x_i) + g_{o_i}(x - x_i) \leq L \;\; i \in I_k \\ \hline f_1(x_j) + g_{1_j}(x - x_j) \leq 0 \\ \vdots \qquad\qquad\qquad\qquad j \in J_k \\ f_m(x_j) + g_{m_j}(x - x_j) \leq 0 \\ \hline x_{min} - x \leq 0 \\ x - x_{max} \leq 0 \end{array} \right\} \tag{5.100}$$

where L defines a lower bound on the objective function f_o. The localization set $\mathcal{F}_{U_k}$ is a bounded polyhedral subset of the feasible region for the relaxed problem that contains any optimal solution (x^*, f_o^*).

The k^{th} query point for ACCPM is the analytic center of the localization set $\mathcal{F}_{U_k}$. This new notion was first introduced by Sonnevend [199–201] as the unique pair

$$(x_k^a, L_k^a) = arg\{\min_{(x,L)\in\mathcal{F}_{U_k}} F_{U_k}(x, L)\} \tag{5.101}$$

that minimizes

$$\begin{aligned} F_{U_k}(x, L) \stackrel{\text{def}}{=} \; & - \log(U_k - L) - \sum_{i\in I_k} \log[L - f_o(x_i) - g_{i_0}(x - x_i)] \\ & - \sum_{j\in J_k} \log[-f_1(x_j) - g_{j_1}(x - x_j)] \\ & \vdots \\ & - \sum_{j\in J_k} \log[-f_m(x_j) - g_{j_m}(x - x_j)] \\ & - \log(x - x_{min}) - \log(x_{max} - x) \quad \textit{for all} \quad x \in \mathcal{F}_{U_k}. \end{aligned} \tag{5.102}$$

This function is a potential function similar to the one used by Karmarkar [190] when presenting the first interior point algorithm for LPs. For the stopping criteria, the computation stops if the gap between upper and lower bounds falls below a certain threshold $\epsilon > 0$;

$$U_k - L_k \leq \epsilon_{obj} \tag{5.103}$$

and if the query point x_k is feasible. Further details about the ACCPM algorithm can be found in works by Goffin et al. and Boyd et al. [23–32]. The oracle-based ACCPM package with MATLAB support was obtained [202] to integrate with our MATLAB-based control design framework.

5.6.3 Kelley's Cutting Plane Method (Kelley's CPM)

In this section, we describe the Kelley's CPM that was proposed independently by Kelley [192] and Cheney and Goldstein [193] Kelley's CPM can be used to solve the constrained convex non-differentiable problem in (5.88) by generating piecewise affine (linear) functions that provide affine lower bounds to the objective and constraint functions and then solving the linear program formed by these bounds. The affine lower bounds are generated using only function values and subgradient information. Thus the method can be applied to non-differentiable optimization problems. As the method progresses, the upper and lower bounds converge toward the desired minimum f_o^*. These bounds permits one to compute a solution to a desired a priori accuracy. While the associated linear program to be solved grows linearly with each iteration, an adequate solution is usually found in practice before the computational requirements become excessive. Kelley's cutting-plane algorithm for the constrained problem is given below:

Input: starting point x_1*, initial box* $(x_{min}, x_{max}) \leftarrow$ *any initial box that contains a*

minimizer

Output: an approximate minimizer x such that

$$f_1(x) \leq \epsilon_{feas}, \ldots, f_m(x) \leq \epsilon_{feas} \text{ and } f_o(x) - L_k \leq \epsilon_{obj}$$

Begin

$k := 0$

repeat{

$k := k + 1$

compute function values $f_o(x_k), f_1(x_k), \ldots, f_m(x_k)$

compute subgradients $g_{o_k} \in \partial f_o(x_k), g_{1_k} \in \partial f_1(x_k), \ldots, g_{m_k} \in \partial f_m(x_k)$

solve the associated LP to find approximate minimizer x_k *and lower bound* L_k

compute upper bound U_k

$x_{k+1} := x_k$ *(Update Minimzer)*

} *until* $f_1(x_k) \leq \epsilon_{feas}, \ldots, f_m(x_k) \leq \epsilon_{feas}$ *and* $U_k - L_k \leq \epsilon_{obj}$ *(Stopping Crierion)*

end.

Further details about the Kelley's CPM algorithm can be found in [59, 153, 192, 193]. Within our MATLAB-based GMS control design framework, the Kelley's CPM algorithm is fully implemented using MATLAB code.

5.6.4 Solver for Local Nonlinear Optimization Problems (SolvOpt)

SolvOpt is an (open-source) optimization package developed by Kuntsevich et al. [203–205] for solving nonlinear problems which can handle nonsmooth objective and constraint functions. It is a subgradient-based method that uses a modified version of Shor's r–algorithm [206–208] to find a local minimum of the problems. SolvOpt is compatible with MATLAB, Python, C and Fortran. In this work, we use the MATLAB compatible package. The main idea of the algorithm is to make steps in the direction opposite to a subgradient at the current point. However, the steps are to

be made in the transformed space. Toward this end, at each iteration the difference between a subgradient at the current point and that calculated at the previous step is calculated. This is used to perform dilation of the space with a priori given coefficient ($\gamma \in [0, 1]$). The r-algorithm also uses space dilation, but in the direction of the difference of two successive subgradients. For the boundary cases $\gamma = 0$ and $\gamma = 1$, the method reduces to steepest descent and to a variant of the conjugate gradient method respectively [206–208]. It must be noted that SolvOpt has been used to solve (local minimum) nonlinear and constrained optimization problems in general that are nonsmooth. Within our work, we use the solver for solving our convex optimization problems. Similar solvers are available that are used in popular control design toolboxes. HANSO is a MATLAB package based on BroydenFletcher-GoldfarbShanno (BFGS) and gradient sampling methods for solving unconstrained nonlinear problems that can handle nonsmooth functions. This has been used in HIFOO - A MATLAB Package for Fixed-Order Controller Design and $\mathcal{H}^\infty$ Optimization [209, 210]. GRadient-based Algorithm for Non-Smooth Optimization (GRANSO) is a MATLAB optimization package based on BFGS and Semidefinite Programming (SDP) that can solve constrained nonsmooth optimization problems, involving objective and constraint functions that can be nonsmooth [211, 212].

5.6.5 Comparison of Convex Optimization Solvers Using Control Problem within GMS

In this subsection, we compare the efficiency of the three subgradient-based constrained convex optimization solvers using simple control example problem. The three methods to be used are

1. Analytic Center Cutting Plane Method (ACCPM)

2. Kelleys Cutting Plane Method (CPM)

3. Solver For Local Nonlinear Optimization Problems (SolvOpt)

All the results in this subsection are obtained using

- Processor: `Intel® Core™ i7-3635QM CPU @ 2.40GHz` (Number of Processors: 1, Total Number of Cores: 4, L2 Cache (per Core): 256 KB, L3 Cache: 6 MB, Memory: 8 GB),
- Operating System: `Mac OS X Version:10.13.5 Build:17F77`,
- MATLAB: Version `9.1.0.441655 (R2016b)`, and
- Java: `Java 1.7.0_75-b13` with Oracle Corporation Java HotSpot™ 64-Bit Server VM mixed mode.

Example 5.6.1 *Consider a* 2×2 *plant [15]:*

$$P = \frac{1}{s}\begin{bmatrix} 10 & 9 \\ 9 & 8 \end{bmatrix} \tag{5.104}$$

(5.105)

Consider the following generalized $\mathcal{H}^\infty$ *mixed sensitivity minimization problem*

$$K = \arg\left\{ \min_{\substack{K \\ stabilizing}} \gamma \middle| \max\left(\left\| \begin{bmatrix} W_1 S_e \\ W_2 K S_e \\ W_3 T_e \end{bmatrix} \right\|_\infty, \left\| \begin{bmatrix} W_4 S_c \\ W_5 P S_c \\ W_6 T_c \end{bmatrix} \right\|_\infty \right) < \gamma \right\} \tag{5.106}$$

The closed loop maps $S_e, KS_e, T_e, S_c, PS_c$ *and* T_c *are closed loop maps as defined in*

Section 3.2.

$$W_1 = \frac{1}{M_s}\left[\frac{s+M_s\omega_b}{s+\epsilon\omega_b}\right]\times I_{2\times 2} = W_4 \tag{5.107}$$

$$W_2 = \frac{1}{\epsilon}\left[\frac{s+\omega_{b_u}\sqrt[k_2]{M_u}}{s+\frac{\omega_{b_u}}{\sqrt[k_2]{\epsilon}}}\right]^{k_2}\times I_{2\times 2} \tag{5.108}$$

$$W_3 = \left[\frac{s+\frac{\omega_{bc}}{M_y}}{\epsilon s+\omega_{bc}}\right]\times I_{2\times 2} = W_6 \tag{5.109}$$

$$W_5 = \left[\frac{M_{dc}\omega_{d1}}{s+\omega_{d1}}\right]\left[\frac{\omega_{d2}}{s+\omega_{d2}}\right]\left[\frac{M_{high}\omega_{d3}}{s+\omega_{d3}}\right]\times I_{2\times 2} \tag{5.110}$$

The weighting function parameters selected are as follows: $M_s = 3, \omega_b = 0.5, \epsilon = 0.01, M_u = 0.01, \omega_{b_u} = 50, M_y = 3, \omega_{bc} = 10, k_2 = 2, M_{dc} = 1, M_{high} = 0.1, w_{d1} = 10^{-2}, w_{d2} = 1, w_{d3} = 10^2$. *The infinite dimensional Youla parameter (Q) is approximated by a finite dimensional paraameter* (Q_N) *with* $N = 6$. *Given that the plant P is* 2×2, *this makes the number of optimization variables* 24 ($N \times n_e \times n_u = 6\times 2\times 2$). *Here, all-pass basis with pole location at* 22 *are chosen. Coprime-factorization based Q-parameterization is used along with an inital controller found based on LQG method (controller weights* $Q_f = I, R_f = 15I$ *and observer weights* $Q_l = I, R_l = 15I$*).*

Initial Data. *For the three methods considered, objective value tolerances of* 10^{-4} *for the stopping criterion, and initial points of ones(N,1) were used. For CPM and ACCPM, the initial box is defined by:* $x_{min} = 100 \times ones(N,1)$ *and* $x_{max} = -100 \times ones(N,1)$.

The Table 5.1 shows optimization relevant results for the above problem from all three optimization solvers. The computation time (CPU time) taken by each solver is denoted by "Total time" in the table. It can be seen that ACCPM outperforms Kelley's CPM (denoted simply as CPM in the table) and SolvOpt. The number of total iterations is denoted by "# Iter", number of objective function evaluations by "# f Evals", number of subgradient computations by "# sg Evals" and average time taken per iteration by "Avg Time/Iter". For this two-objective optimization problem,

number of function evaluations is double that of subgradients. At each iteration, the subgradient is computed only for the objective function which is "active" (see Subsection 5.5.3 for details). The updates of objective function value with respect to iteration count and elapsed computation time are shown in Figures 5.6 and 5.7 respectively.

	Total time (s)	# Iter	# f Evals	# sg Evals	Avg Time/Iter
ACCPM	87.50	164	328	164	0.533
CPM	205.83	573	1146	573	0.359
SolvOpt	553.34	457	915	457	1.211

Table 5.1: Optimization Example 5.6.1: Computation Times and Number of Iterations

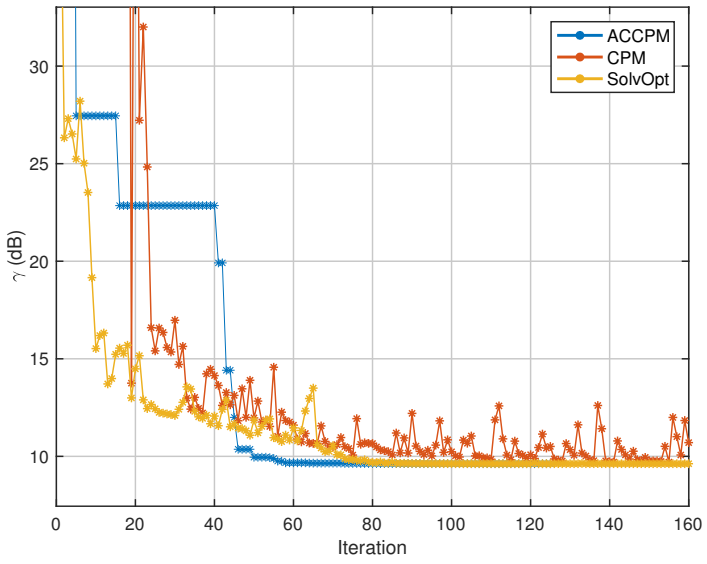


Figure 5.6: Optimization Example 5.6.1: γ versus Iteration Count

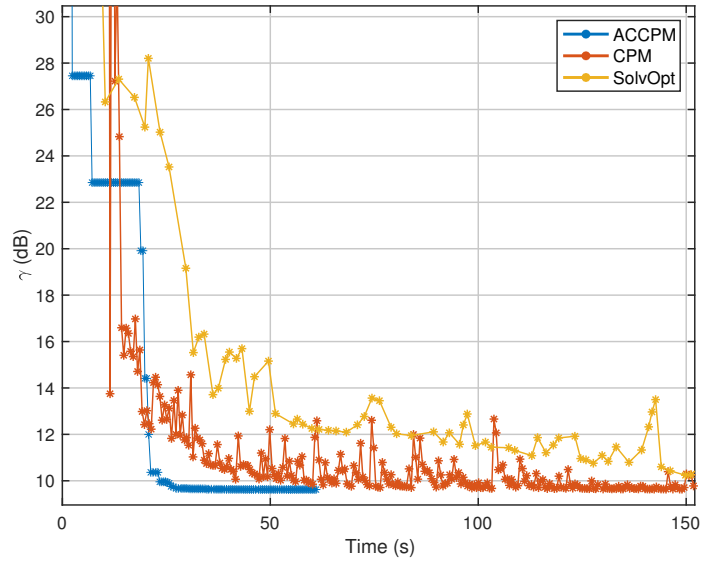


Figure 5.7: Optimization Example 5.6.1: γ versus Computation Time (s)

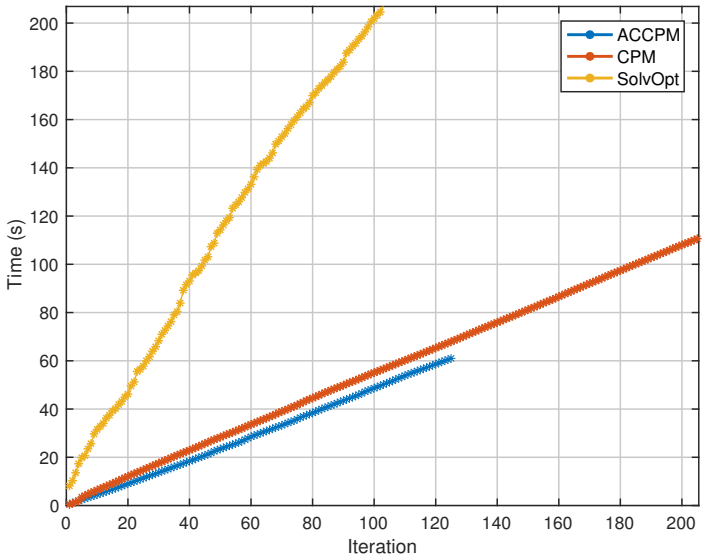


Figure 5.8: Optimization Example 5.6.1: Time versus Iteration Count

Tables 5.2 and 5.3 show the total computation times and number of iterations for varying number of optimization variables. Here, the number of basis terms considered

are $N = 2, 3, 6, 9,$ *and* 10*. Corresponding to these - given that the plant is* 2×2*, the number of optimization variables are* $8, 12, 24, 36,$ *and* 40 *respectively.*

# Variables	8	12	24	36	40
ACCPM	11.45	19.22	87.50	216.49	295.4
CPM	28.41	97.3	205.83	843.65	1475.9
SolvOpt	58.74	116.80	553.34	3436.90	4463.7

Table 5.2: Optimization Example 5.6.1: Computation Times (s) for Increasing Number of Variables

# Variables	8	12	24	36	40
ACCPM	46	58	164	187	197
CPM	92	269	573	686	891
SolvOpt	64	109	457	869	847

Table 5.3: Optimization Example 5.6.1: Number of Iterations for Increasing Number of Variables

∎

5.7 Basis Selection

In this section, we discuss some rules of thumb for selecting basis parameters for finite-dimensional approximation of the infinite-dimensional Youla et al. (Q)-parameter [173–177] using examples. Consider a simple SISO stable system.

$$P = \frac{1}{s+1} \tag{5.111}$$

The set of all stabilizing controllers for this stable SISO system can parameterized as follows:

$$K_{stabilizing} = \frac{Q}{1 - PQ} \tag{5.112}$$

Consider a standard $\mathcal{H}^\infty$ mixed sensitivity problem:

$$K_{mso} = \arg \left\{ \min_{K \ stabilizing} \gamma \ \left| \ \left\| \begin{bmatrix} W_1 S_e \\ W_2 K S_e \end{bmatrix} \right\|_{\mathcal{H}^\infty} < \gamma \right. \right\} \tag{5.113}$$

with the weights

$$W_1 = \frac{1}{M_s} \left[\frac{s + M_s \omega_b}{s + \epsilon \omega_b} \right] \tag{5.114}$$

$$W_2 = \frac{1}{\epsilon M_u} \left[\frac{s + \omega_{b_u}}{s + \omega_{b_u}/\epsilon} \right] \tag{5.115}$$

where $M_s = 1.5$, $\omega_b = 10$, $\epsilon = 0.01$, $M_u = 1/30$, and $\omega_{b_u} = 750$.

Using MATLAB's *hinfsyn* command that uses the two-Riccati formulae [157, 158], we obtain a performance $\gamma = 0.97908820$. The parameter $TOLGAM$ within the command [154] that represents relative error tolerance for γ was set to a very low value ($\approx 10^{-10}$). Given this, we call the performance achieved as our "optimal" performance (i.e., $\gamma_{opt} = 0.97908820$). The corresponding controller obtained is,

$$K_{opt} = \frac{7.9839 \times 10^8 \ (s+1)^2 \ (s+0.1) \ (s + 7.5 \times 10^4)}{(s + 1.113 \times 10^{11}) \ (s + 53.14) \ (s+1) \ (s + 0.1135) \ (s + 0.08666)} \tag{5.116}$$

Figures 5.9 - 5.12 show the open and closed loop maps for this this design.

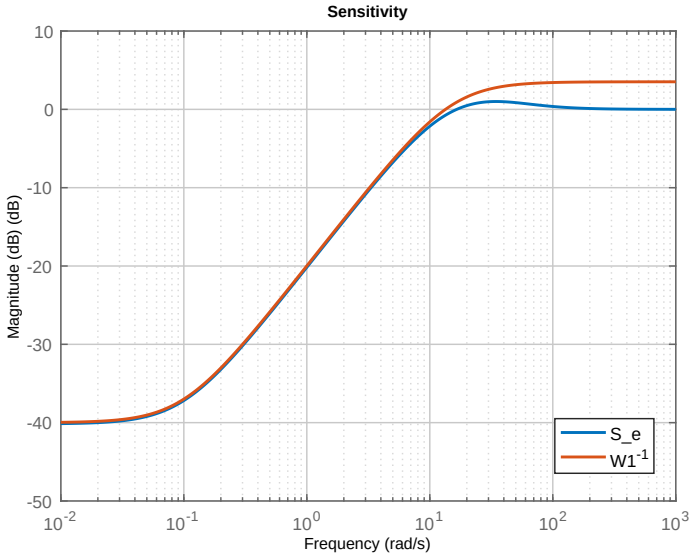


Figure 5.9: Basis Selection: Sensitivity

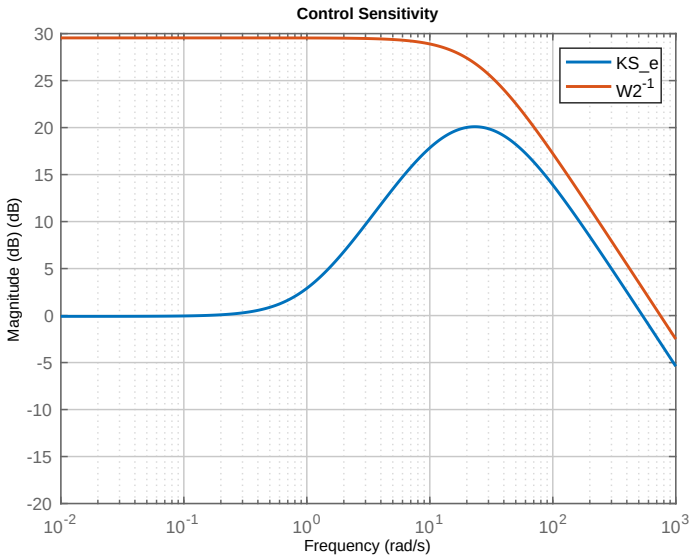


Figure 5.10: Basis Selection: Control Sensitivity

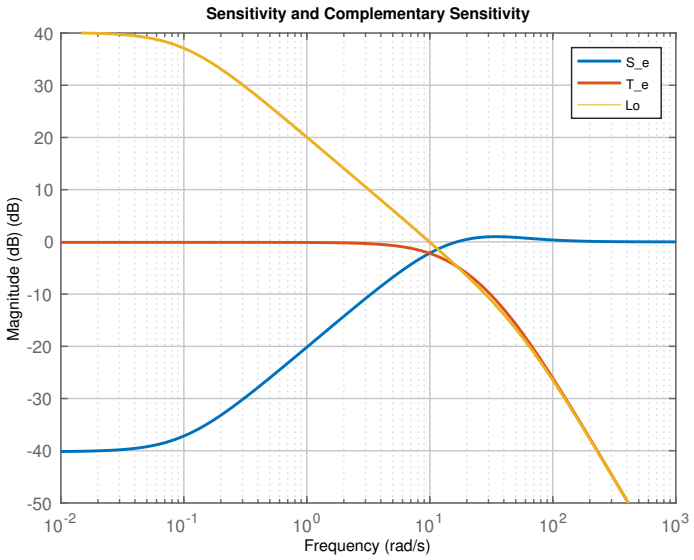


Figure 5.11: Basis Selection: Open and Closed Loop Maps

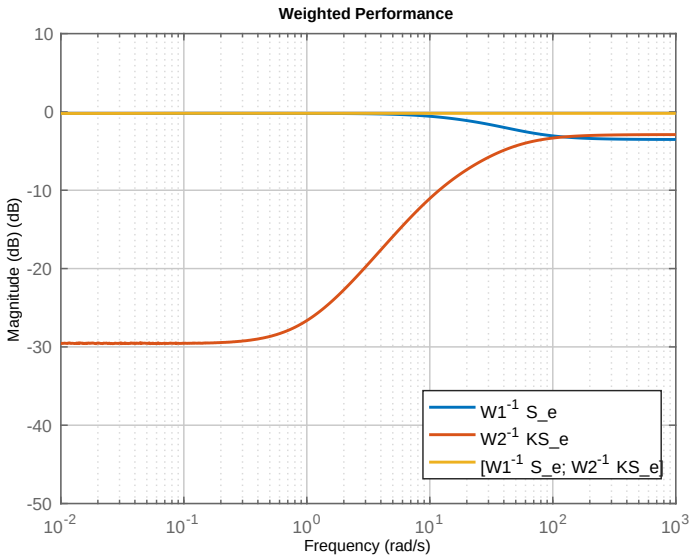


Figure 5.12: Basis Selection: Performance Measure

Now, we solve the Problem (5.113) using our Generalized Mixed Sensitivity framework. Here, we solve the same problem using different values of number of basis terms

(N) and basis pole (α), for the fixed pole all-pass basis

$$q_k = \left(\frac{\alpha - s}{s + \alpha}\right)^{k-1} \quad k = 1, 2, ..., N, \quad \alpha \in \mathcal{R}^+ \tag{5.117}$$

For various values of number of basis terms (N) and basis pole (α) the GMS problem is solved to visualize the behavior of performance γ with respect to N and α. Doing this, we obtain the performance γ as shown in Figure 5.13. It can clearly be seen that the performance improves as number of basis terms (N) increases. But, there is a value of basis pole (α) above or below which performance deteriorates. Here, the parameterization is done using a zero (0) initial controller (Zames parameterization) for simplicity. This is a valid choice, as the plant considered is stable. For an unstable plant, we must use a stabilizing initial controller which is non-zero.

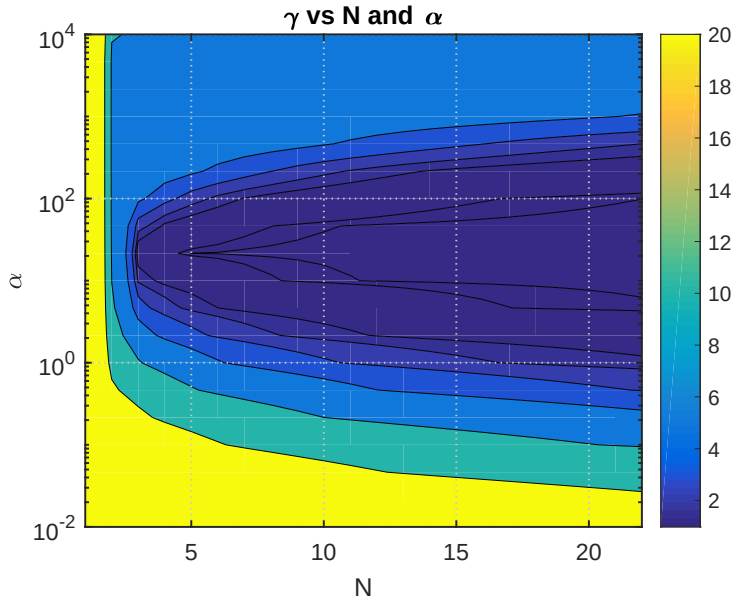


Figure 5.13: Basis Selection: γ versus N and α

Figure 5.14 shows the performance γ versus basis pole (α) for fixed number of basis terms (N).

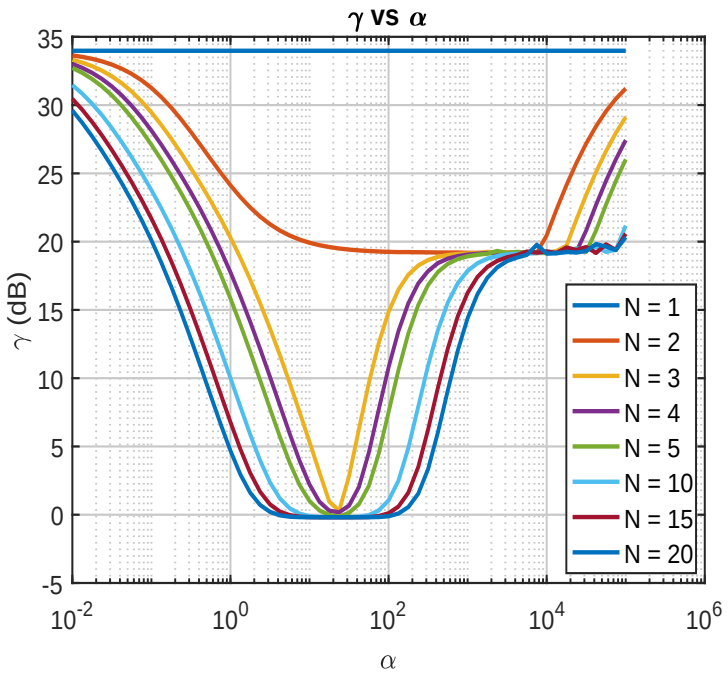


Figure 5.14: Basis Selection: γ versus α for Fixed N

Figure 5.14 shows the minimum value of N required to achieve at least some percentage of optimal Performance for different values of α.

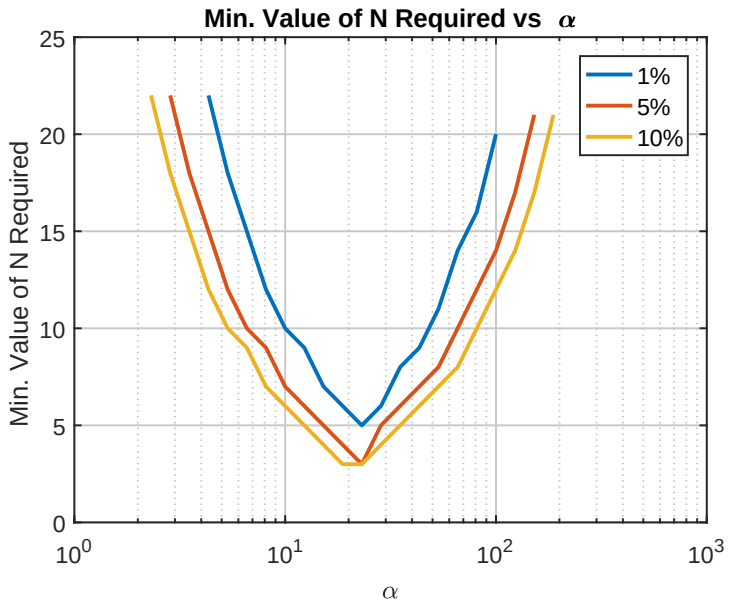


Figure 5.15: Basis Selection: Minimum Value of N Required versus α for Some Percentage of Optimal Performance

Using the basis parameter to be $\alpha \approx 22.5$, lesser number of basis terms N may be sufficient to obtain a desired perormance γ. Note that in this case the open loop bandwidth (unity gain crossover frequency of L) is ≈ 10. The "optimal" basis parameter in this case is nearly double of the open loop bandwidth.

Approximation of $\mathbf{Q_{opt}}$ using a Real-Rational Basis.

Consider the approximation of a function f by the following real-rational expansion [133]:

$$f(x) \approx c_o + c_1 \left(\frac{x - x_o}{x + x_o}\right) + c_2 \left(\frac{x - x_o}{x + x_o}\right)^2 + \cdots . \tag{5.118}$$

The coefficients can be shown to be

$$c_1 = \lim_{x \to x_o} f_1(x) \quad for \quad f_1(x) \stackrel{\text{def}}{=} \frac{(x + x_o)^2}{2x_o} f_o{}'(x) \tag{5.119}$$

$$c_2 = \frac{1}{2!} \lim_{x \to x_o} f_2(x) \quad for \quad f_2(x) \stackrel{\text{def}}{=} \frac{(x + x_o)^2}{2x_o} f_1{}'(x) \tag{5.120}$$

$$c_3 \;=\; \frac{1}{3!}\lim_{x\to x_o} f_3(x) \quad for \quad f_3(x) \stackrel{\text{def}}{=} \frac{(x+x_o)^2}{2x_o} f_2{}'(x) \tag{5.121}$$

Now Q_{opt} corresponding to the K_{opt} obtained in Equation 5.116 is

$$Q_{opt} \;=\; \frac{5.3341e06\ (s+7.5e05)\ (s+1)^3\ (s+0.1)}{(s+1.134e09)\ (s+277.4)\ (s+12.82)\ (s+1)^2\ (s+0.1)} \tag{5.122}$$

Consider the approximation of Q_{opt} using the real-rational basis terms

$$q_k = \left(\frac{\alpha - s}{s+\alpha}\right)^{k-1} \quad k = 1, 2, ..., N, \quad \alpha \in \mathcal{R}^+ \tag{5.123}$$

Figure 5.16 shows the approximation error in Eqn 5.124 as the number of terms N and basis pole α are varied.

$$\epsilon \;=\; \|W\,(Q_{opt} - Q_{N,\alpha})\|_{\mathcal{H}^\infty} \tag{5.124}$$

Here, the weight W is a band-pass filter around the open-loop system crossover frequency. The magnitude plot of W is shown in Figure 5.19

$$W \;=\; \frac{(s+0.001)}{(s+0.1)} \times \frac{(0.01s+1000)}{(s+1000)} \tag{5.125}$$

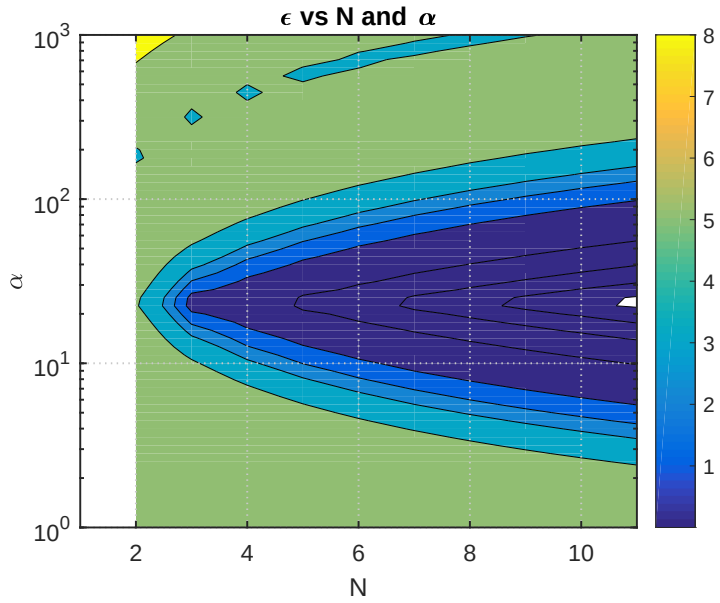


Figure 5.16: Basis Selection: ϵ versus N and α

Figure 5.17 shows the ϵ versus basis pole (α) for fixed number of basis terms (N).

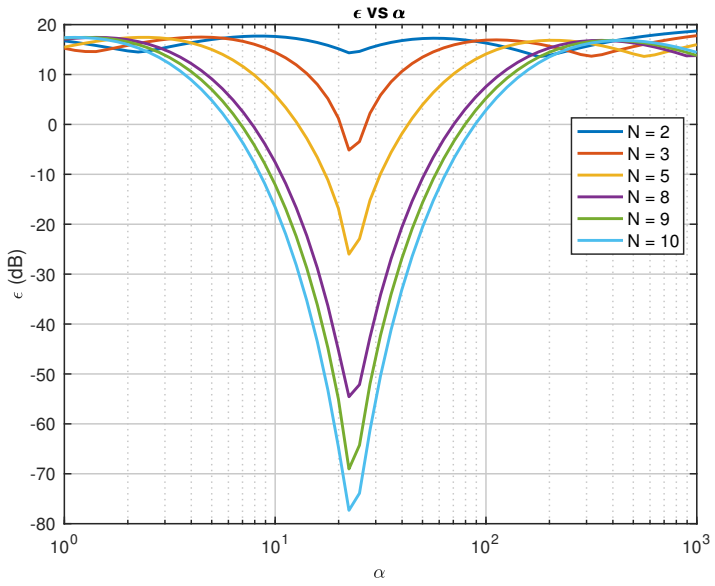


Figure 5.17: Basis Selection: ϵ versus α for Fixed N

Figure 5.18 shows the minimum value of N required to achieve a desired value of ϵ.

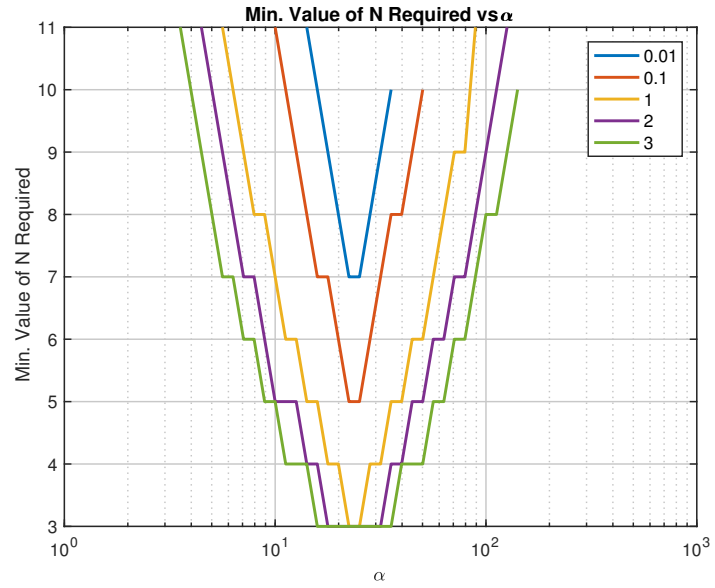


Figure 5.18: Basis Selection: Minimum Value of N Required to Achieve Desired ϵ

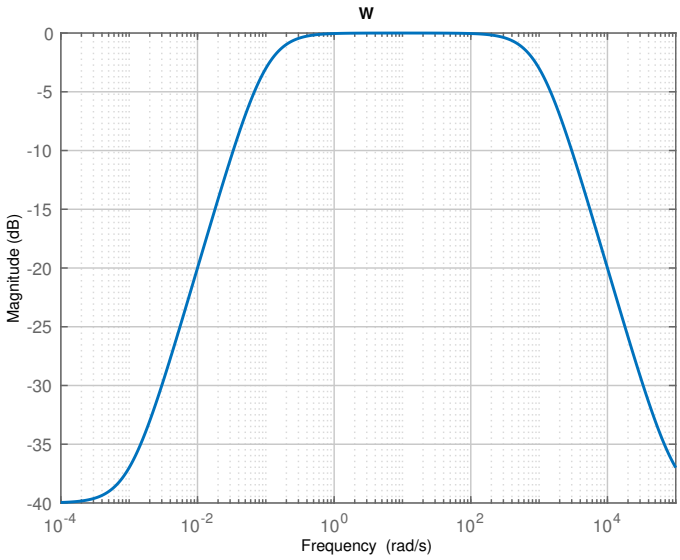


Figure 5.19: Basis Selection: Weighting Function W

The value of basis parameter α that results in acceptable value of performace γ with least number of basis terms N is found to be ≈ 22.5. This is in agreement with the "optimal" value obtained by running GMS for different values of α.

Laguerre Basis. We sweep over the number of basis terms (N) and basis pole (α), for the same problem as above, but using Laguerre basis

$$q_k = \frac{\sqrt{2\alpha}}{s+\alpha}\left(\frac{s-\alpha}{s+\alpha}\right)^{k-1} \quad k = 1, 2, ..., N, \quad \alpha \in \mathcal{R}^+ \tag{5.126}$$

where α is a positive real number. For various values of number of basis terms (N) and basis pole (α), we obtain the performance γ as shown in Figure 5.13. It is clear that the performance improves as number of basis terms (N) is increased. But, there is a value of basis pole (α) above or below which performance deteriorates.

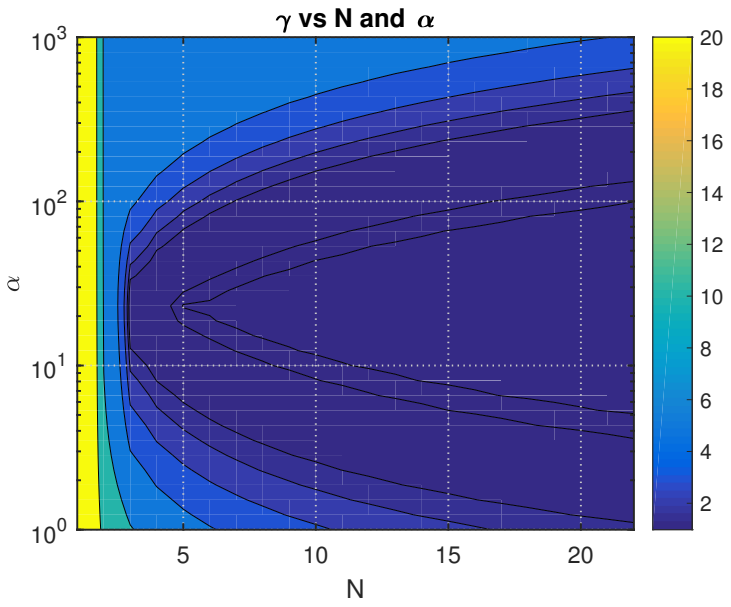


Figure 5.20: Basis Selection: γ vs N and α

Figure 5.14 shows the performance γ versus basis pole (α) for fixed number of basis terms (N).

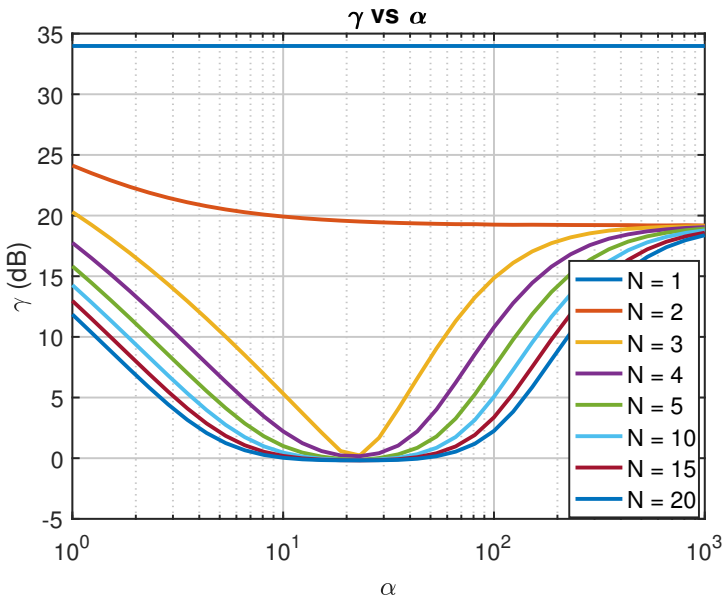


Figure 5.21: Basis Selection: γ vs α for Fixed N

5.8 Summary and Conclusions

In this chapter, a general control system design problem was formulated. The problem was infinite-dimensional and nonlinear in the controller K. It was shown how the Youla et al. parameterization and approximation ideas may be used to transform the problem to a finite-dimensional convex optimization problem for which efficient numerical algorithms exist. This chapter discussed several subgradient-based constrained convex optimization algorithms and solvers, along with comparisons between them. Finally, the dependence of convergence rate on basis parameters used for Youla et al. parameterization were studied through examples. The approach taken - at best - provides us with a methodology for computing finite-dimensional LTI controllers that satisfy important design specifications for which no direct approach exists. At the very least, the approach provides us with a methodology to assess fundamental performance limitations associated with a very wide class of design specifications. These ideas are be applied to several control system design problems in what follows.

Chapter 6

CONTROL-RELEVANT TRADEOFFS USING GENERALIZED MIXED SENSITIVITY METHODOLOGY

6.1 Overview

In this chapter, we present several control problems with different objective and constraint functions to illustrate critical control-relevant tradeoffs associated with them. Firstly, tradeoffs associated with MIMO ill-conditioned plants that are not typically seen in SISO systems are illustrated. This is done using GMS by considering the problem of simultaneously addressing output and input properties. An example that illustrates handling $\mathcal{L}^\infty$ time-domain constraint is also presented. Further, some SISO problems are considered that discuss specific control challenges and tradeoffs.

6.2 Multiobjective Weighted Sensitivity Minimization of an Ill-Conditioned Plant

When the plant to be controlled is ill-conditioned, one of the main difficulties is that obtaining good properties at one feedback loop-breaking point, does not guarantee acceptable properties at a different loop-breaking point [9, 10]. For a plant that is square (i.e., number of input/s equal to number of output/s) and assuming $P(j\omega)$ is invertible at frequencies of interest, the following inequality gives upper bound on $\|S_c\|_{\mathcal{H}^\infty}$ using $\|S_e\|_{\mathcal{H}^\infty}$ and $\kappa[P]$.

$$\frac{1}{\kappa[P]}\sigma_i\left[S_e\right] \leq \sigma_i\left[S_c\right] \leq \kappa[P]\sigma_i\left[S_e\right] \tag{6.1}$$

If the condition number of the plant is high, achieving good feedback properties at plant output might still result in bad properties at plant input. Analogous results can

be obtained as upper bound on $\|S_e\|_{\mathcal{H}^\infty}$. Hence, both S_e and S_c must be addressed during the design process based on the specifications. This issue is also discussed in Section 3.3.

Consider a 2-input, 2-output ill-conditioned plant. The transfer function matrix of the plant is shown below [15]:

$$P(s) = \frac{1}{s}\begin{bmatrix} 10 & 9 \\ 9 & 8 \end{bmatrix} \tag{6.2}$$

The condition number is $\kappa[P(j\omega)] \approx 50dB$ for all $\omega \in \mathcal{R}_+$. RGA-sum norm and scaled condition number are also near $50dB$ for all $\omega \in \mathcal{R}_+$. Note that $\kappa[P(j\omega)] \approx \kappa^*[P(j\omega)]$ means that the plant is (almost) "optimally" scalled in the sense that condition number is minimum with respect to scaling of inputs and outputs.

Consider the following constrained multiobjective weighted sensitivity minimization problem:

$$K = \arg\left\{ \min_{\substack{K \\ stabilizing}} \gamma \;\middle|\; \begin{array}{c} \max\left(\rho \,\|\, W_1 S_e\|_{\mathcal{H}^\infty}, (1-\rho) \,\|\, W_4 S_c\|_{\mathcal{H}^\infty}\right) < \gamma, \\ \|\, W_{2c} K S_e\|_{\mathcal{H}^\infty} < c_2 \end{array} \right\} \tag{6.3}$$

The weighting functions selected are as follows:

$$W_1 = \frac{1}{M_{se}}\left[\frac{s + M_{se}\omega_{be}}{s + \epsilon\omega_{be}}\right] \times I_{2\times 2} \tag{6.4}$$

$$W_4 = \frac{1}{M_{sc}}\left[\frac{s + M_{sc}\omega_{bc}}{s + \epsilon\omega_{bc}}\right] \times I_{2\times 2} \tag{6.5}$$

$$W_{2c} = \begin{bmatrix} c_{11} & 0 \\ 0 & c_{22} \end{bmatrix} \tag{6.6}$$

with $M_{se} = M_{sc} = 2$, $\omega_{be} = 1$, $\omega_{bc} = 2$, $\epsilon = 0.01$, $c_{11} = 0.5$ and $c_{22} = 1$.

Multiobjective Weighted Sensitivity Minimization of an Ill-Conditioned Plant Using GMS.

The optimization problem is solved for several different values of the tradeoff parameter $\rho \in [0, 1]$ using GMS. It can be seen from the tradeoff curves in Figures 6.1 and 6.2 that improving one of the two objectives of the multiobjective function (e.g. $\|W_1 S_e\|_{\mathcal{H}\infty}$) results in degrading the other objective (e.g. $\|W_4 S_c\|_{\mathcal{H}\infty}$). It can also be seen from the figures that at near the point $\rho = 0.5$, the two objective function values become equal. This example illustrates the pareto optimality [213] associated with this weighted sensitivity minimization problem simultaneously at output/error and input/controls.

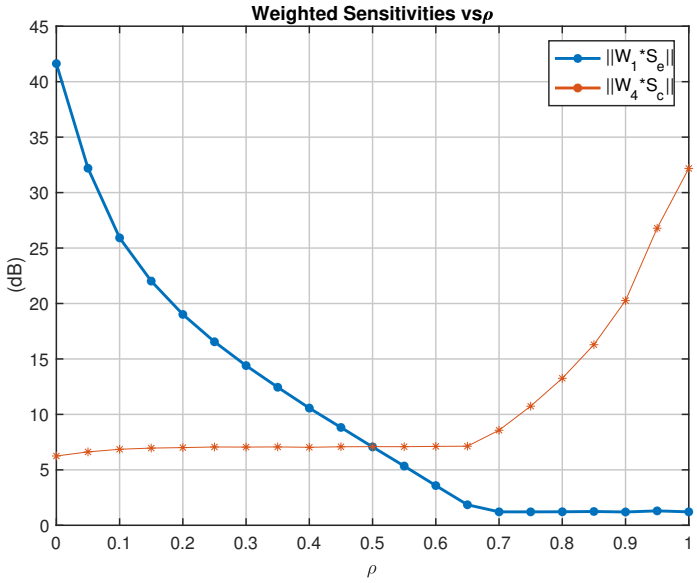


Figure 6.1: Pareto Optimality in Weighted Sensitivity Minimization Using GMS: $\|W_1 S_e\|_{\mathcal{H}\infty}$ and $\|W_4 S_c\|_{\mathcal{H}\infty}$ versus ρ

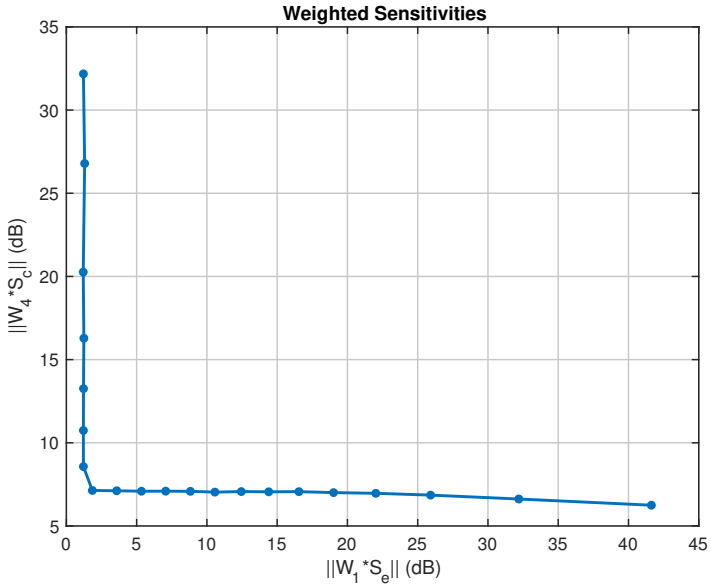


Figure 6.2: Pareto Optimality in Weighted Sensitivity Minimization Using GMS: $\|W_1 S_e\|_{\mathcal{H}^\infty}$ versus $\|W_4 S_c\|_{\mathcal{H}^\infty}$

The setup for the above optimization problem used within GMS framework is now discussed briefly. Initial controller and observer gains used for Youla et al, parameterization are obtained using LQG ideas with the weighting matrices all equal to identity. The basis (fixed pole all-pass) parameters chosen are $\alpha = 2.5$, $N = 7$. The upper and lower bounds on optimization variables are picked to be ± 100 resectively, with an initial value of 1. The MATLAB code used to generate the results can be found in Appendix A.1.

The sensitivities of the "equilibrated" design corresponding to the point where both the objective functions are nearly equal is shown in Figures 6.3. Figure 6.4 shows the constraint weighted KS_e along with the constraint value c_2 (see Equation (6.3)).

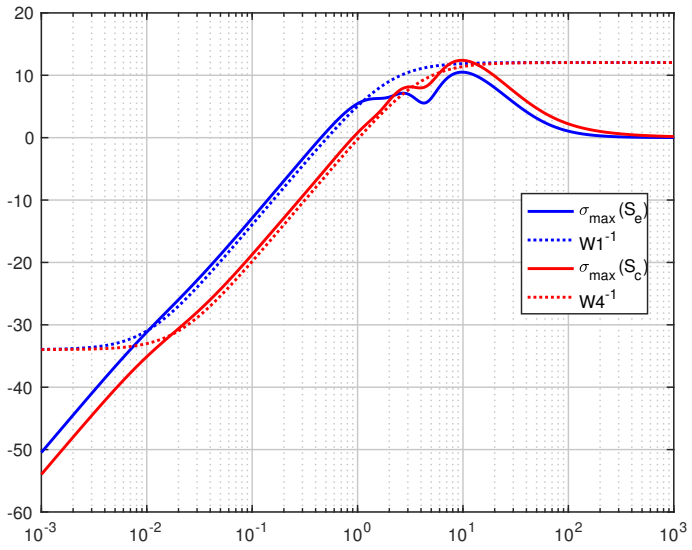


Figure 6.3: Pareto Optimality in Weighted Sensitivity Minimization Using GMS: Sensitivities of Equilibrated Design

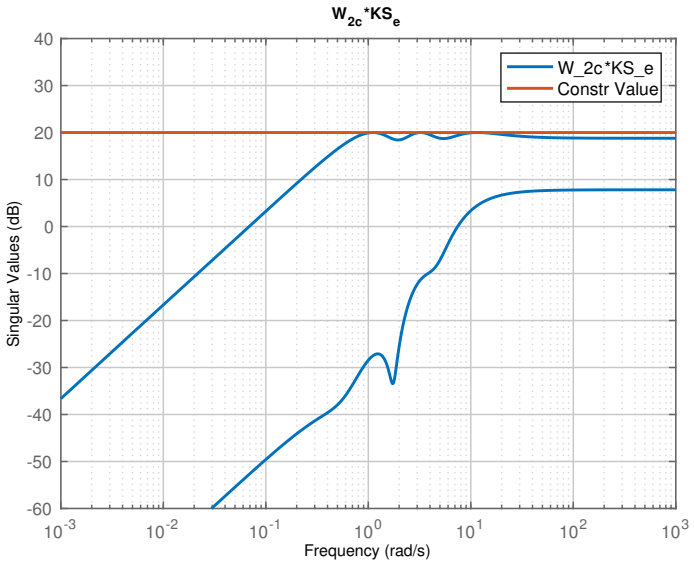


Figure 6.4: Pareto Optimality in Weighted Sensitivity Minimization Using GMS: Constraint Weighted KS_e

Figure 6.5 shows the pareto optimality curve corresponding the problem when three different constraint values on weighted KS_e are considered. As one would expect, as the constraint becomes stricter (c_2 decreases), the optimum objective function value (weighted sensitivities in this case) get worse overall (for $\rho \in [0, 1]$).

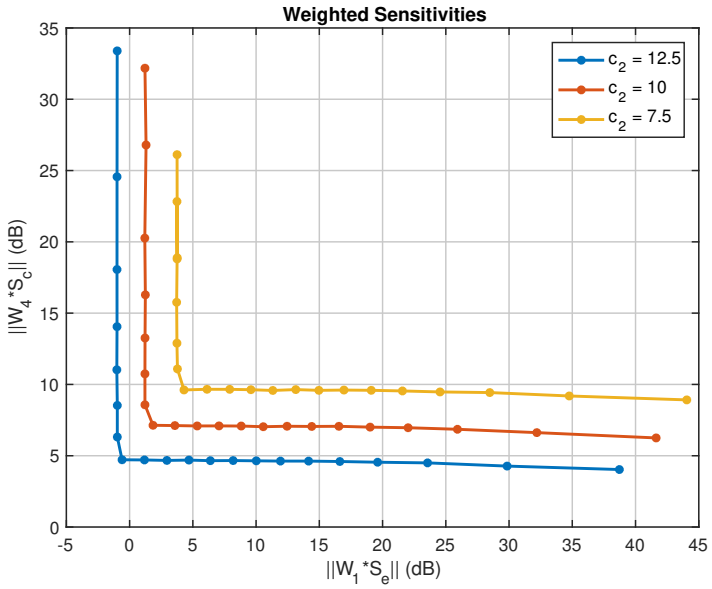


Figure 6.5: Pareto Optimality in Weighted Sensitivity Minimization Using GMS: Dependence on Constraint Value

Multiobjective Weighted Sensitivity Minimization of an Ill-Conditioned Plant using FMINCON.

For simplicity, we assume that the controller is static. That is, the controller K is of the form

$$K = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \tag{6.7}$$

where $k_{ij} \in \mathcal{R}; i = 1, 2;$ and $j = 1, 2$. With these settings we solve the optimization problem (6.3) using MATLAB's nonlinear optimization solver *fmincon* for several different values of the tradeoff parameter $\rho \in [0, 1]$. It can be seen from the tradeoff

curves in the Figures 6.6 and 6.7 that improving one of the two objectives of the multiobjective function (e.g. $\|W_1 S_e\|_{\mathcal{H}^\infty}$) will result in degrading the other objective (e.g. $\|W_4 S_c\|_{\mathcal{H}^\infty}$). It can also be seen from the figures that at near the point $\rho = 0.5$, the two objective function values become equal. This example illustrates the pareto optimality [213] associated with this weighted sensitivity minimizaton problem simultaneously at output/error and input/controls.

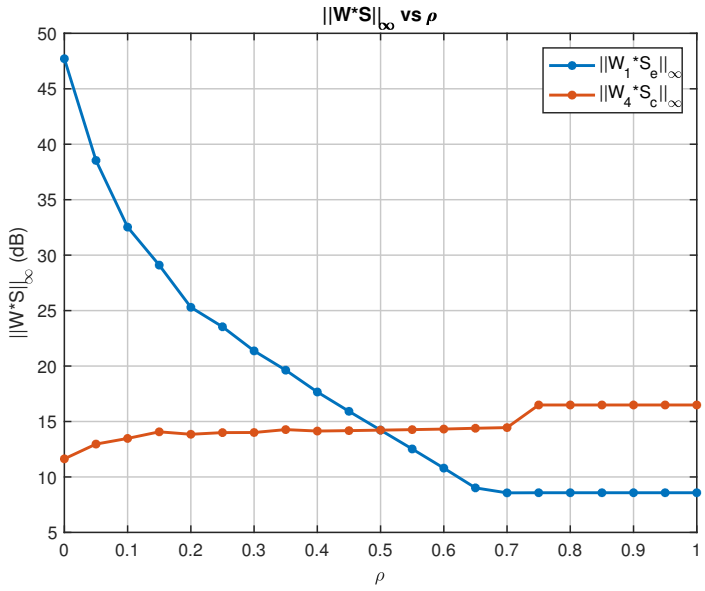


Figure 6.6: Pareto Optimality in Weighted Sensitivity Minimization Using FMINCON: $\|W_1 S_e\|_{\mathcal{H}^\infty}$ and $\|W_4 S_c\|_{\mathcal{H}^\infty}$ versus ρ

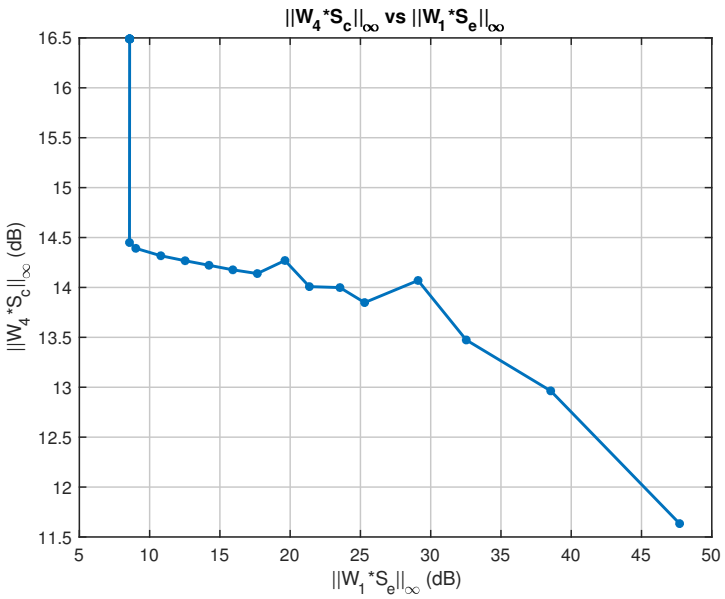


Figure 6.7: Pareto Optimality in Weighted Sensitivity Minimization Using FMINCON: $\|W_1 S_e\|_{\mathcal{H}\infty}$ versus $\|W_4 S_c\|_{\mathcal{H}\infty}$

Note that the controller associated with each value of tradeoff parameter ρ is obtained using MATLAB's nonlinear solver where we assumed K is static (involving only 4 optimization variables). The optimization problem is nonconvex due to objective and constraint functions are nonconvex The MATLAB script used for solving the above problem is as in Appendix A.5. The above solution method to the nonlinear optimization problem faces the issue of being stuck in one of the several possible local minima. One way to tackle this issue is by solving the same problem several times with different random initial points/guesses. A more efficient way is to use global optimization solvers that guarantee obtaining global minimum under some specific settings. A list of such solvers can be seen in the works by Mittelmann [214]. Though it is well known that numerous such global optimization solvers exist, those that directly support MATLAB-based problems are limited in number. To make use of larger set of solvers, our control problems of interest which are constrained

non-smooth and multiobjective can be transformed using modeling languages such as A Mathematical Programming Language (AMPL) [215–217] and General Algebraic Modeling System (GAMS) [218, 219]. These standard widely used platforms enable us to use large set of solvers. As part of future work, the MATLAB-based GMS framework can be extended to include support for using solvers do not have MATLAB support.

6.3 Multiobjective Weighted Mixed Sensitivity Minimization of an Ill-Conditioned Plant

Consider again the 2-input, 2-output ill-conditioned plant considered in Section 6.2

$$P(s) = \frac{1}{s}\begin{bmatrix} 10 & 9 \\ 9 & 8 \end{bmatrix} \tag{6.8}$$

Consider the following multiobjective weighted mixed sensitivity minimization problem:

$$K = \arg\left\{ \min_{\substack{K \\ stabilizing}} \gamma \middle| \max\left(\rho \left\| \begin{bmatrix} W_1 S_e \\ W_2 K S_e \end{bmatrix} \right\|_{\mathcal{H}^\infty}, (1-\rho) \left\| \begin{bmatrix} W_4 S_c \\ W_5 P S_c \end{bmatrix} \right\|_{\mathcal{H}^\infty} \right) < \gamma \right\} \tag{6.9}$$

The weighting functions selected are as follows:

$$W_1 = W_4 = \frac{1}{M_s}\left[\frac{s + M_s\omega_b}{s + \epsilon\omega_b}\right] \times I_{2\times 2} \tag{6.10}$$

$$W_2 = W_5 = I_{2\times 2} \tag{6.11}$$

with $M_s = 2$, $\omega_b = 1$, $\epsilon = 0.01$.

The optimization problem is solved for several different values of the tradeoff parameter $\rho \in [0, 1]$. Figures 6.8 and 6.9 show the relevant properties. It can be seen from the tradeoff curves in the Figures 6.8 and 6.9 that improving one of the two objectives of the multiobjective function $\left(\text{e.g.} \left\| \begin{bmatrix} W_1 S_e \\ W_2 K S_e \end{bmatrix} \right\|_{\mathcal{H}^\infty} \right)$ will result in degrading the other

objective $\left(\text{e.g.} \left\| \begin{bmatrix} W_4 S_c \\ W_5 P S_c \end{bmatrix} \right\|_{\mathcal{H}^\infty}\right)$. It can also be seen from the figures that at near the point $\rho = 0.5$, the two objective function values become equal. This example illustrates the pareto optimality [213] associated with this weighted mixed sensitivity problem simultaneously at output/error and input/controls.

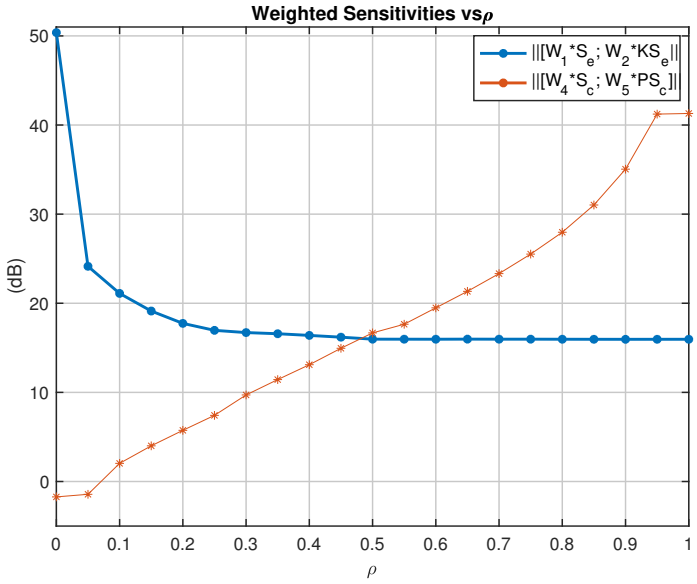


Figure 6.8: Pareto Optimality in Weighted Mixed Sensitivity Minimization Using GMS: First and Second Objectives versus ρ

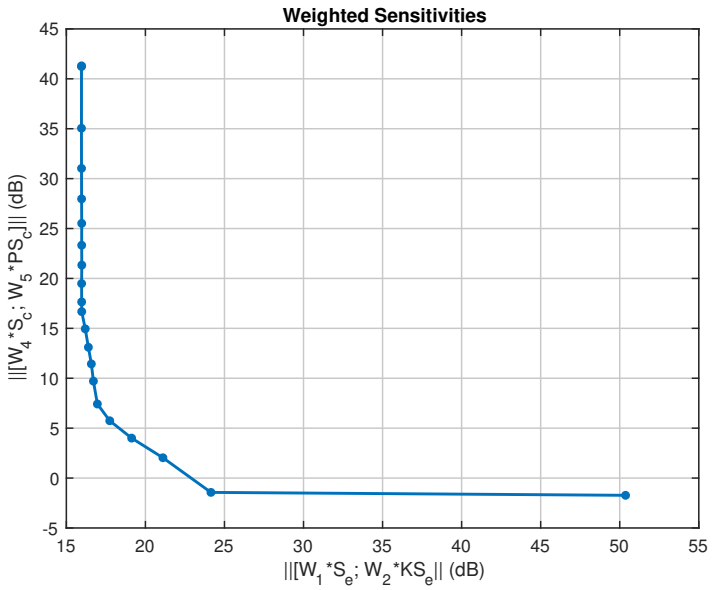


Figure 6.9: Pareto Optimality in Weighted Mixed Sensitivity Minimization Using GMS: First Objective versus Second Objective

The setup for the above optimization problem used within GMS framework is now discussed briefly. Initial controller and observer gains used for Youla et al, parameterization are obtained using LQG ideas with the weighting matrices all equal to identity. The basis (fixed pole all-pass) parameters chosen are $\alpha = 2.5$, $N = 7$. The upper and lower bounds on optimization variables are picked to be ± 100 resectively, with an initial value of 1. The MATLAB code used to generate the results can be found in Appendix A.1.

6.4 Weighted Mixed Sensitivity Minimization Subject to $\mathcal{L}^\infty$ Time-Domain Constraint

In this section, we consider the problem of imposing time-domain $\mathcal{L}^\infty$ constraint on closed loop map. We show how GMS can be used to impose such constraints, and compare the resulting design with the unconstrained case. We consider the same

plant $\left(P = \frac{1}{s+1}\right)$ and objective function considered in Problem 5.113. Along with this objective, we impose $\mathcal{L}^\infty$ constraint on KS_e i.e., peak control response to step reference comamand (unfiltered). Thus the constrained problem becomes

$$K_{mso} = \arg \left\{ \min_{K \ stabilizing} \gamma \ \middle| \ \begin{array}{c} \left\| \begin{bmatrix} W_1 S_e \\ W_2 K S_e \end{bmatrix} \right\|_{\mathcal{H}^\infty} < \gamma \\ \|W_{2c} K S_e\|_{\mathcal{L}^\infty} < c \end{array} \right\} \tag{6.12}$$

with the weighting functions

$$W_1 = \frac{1}{M_s} \left[\frac{s + M_s \omega_b}{s + \epsilon \omega_b} \right] \tag{6.13}$$

$$W_2 = \frac{1}{\epsilon M_u} \left[\frac{s + \omega_{b_u}}{s + \omega_{b_u}/\epsilon} \right] \tag{6.14}$$

$$W_{2c} = 1 \tag{6.15}$$

where $M_s = 1.5$, $\omega_b = 10$, $\epsilon = 0.01$, $M_u = 1/30$, and $\omega_{b_u} = 750$. For the unconstrained case the peak control signal to step reference command (unfiltered) is found to be ≈ 8. This can be seen in Figure 6.10 shown by the curve labelled "Uncon unfilt". Given this, if the constraint parameter $c > 8$ then the constrained problem is equivalent to the unconstrained one. As c is decreased to values below 8, the constrained and unconstrained problems start to differ. Figure 6.10 shows the control response to unit step reference command for both these cases. The curve labelled "Con unfilt" refers to the constrained problem. It can be seen that the control response in this case just reaches a maximum value of 6 (as desired). This is an illustration of how time-domain constraint can be imposed on closed loop responses (Here, KS_e). This can be extended to any other closed loop map (weighted or unweighted) [180, 181, 220, 221]. In all these cases, the subgradient information used in the required by subgradient-based optimization algorithms can be obtained as discussed in Section 5.5.2. In Figure 6.10, the response corresponding to filtered step reference is also presented. In this case, we

use the controller obtained using the unconstrained problem. A prefilter (first order) is iteratively (manual process) designed to limit the peak control signal to a value of 6. The resulting prefilter is

$$W_{filter} = \frac{22}{s + 22} \tag{6.16}$$

The response is shown in Figure 6.10 by the curve labelled "Uncon filt'. It is seen that the response is limited to a maximum value of just below 6. Figures 6.10 - 6.15 shows all the relevant closed loop responses corresponding to the above designs. For comparison purpose, the problem (unconstrained case) is also solved using MATLAB's standard mixed sensitivity tool *hinfsyn*. The corresponding responses (unfiltered) are given by the curves labelled "Uncon hinfsyn". It can be seen that as expected, these responses almost overlap with the unconstrained design obtained by GMS (labelled "Uncon unfilt'). The problem setup when GMS is now discussed briefly. As the plant is stable Zames parameterization is used (along with 0 initial controller). The basis (fixed pole all-pass) parameters chosen are $\alpha = 22$, $N = 10$. The upper and lower bounds on optimization variables are picked to be ± 100 resectively, with an initial value of 1. The MATLAB code used to generate the figures can be found in Appendix A.1.

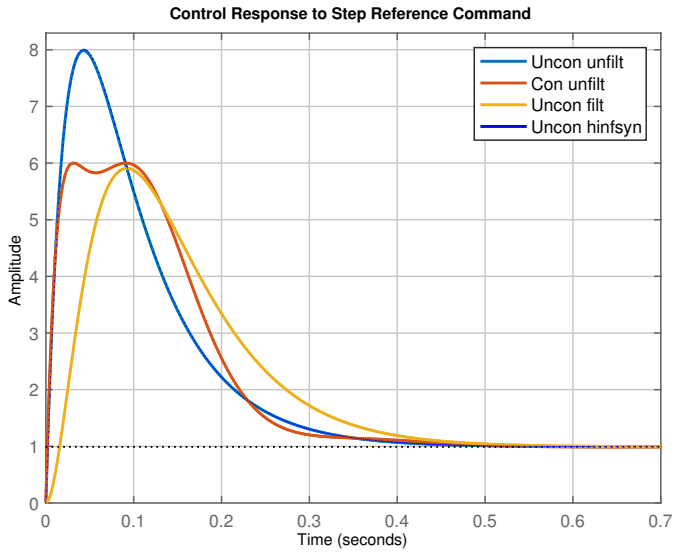


Figure 6.10: Imposing Time-Domain Constraint: Control Response to Step Reference Command

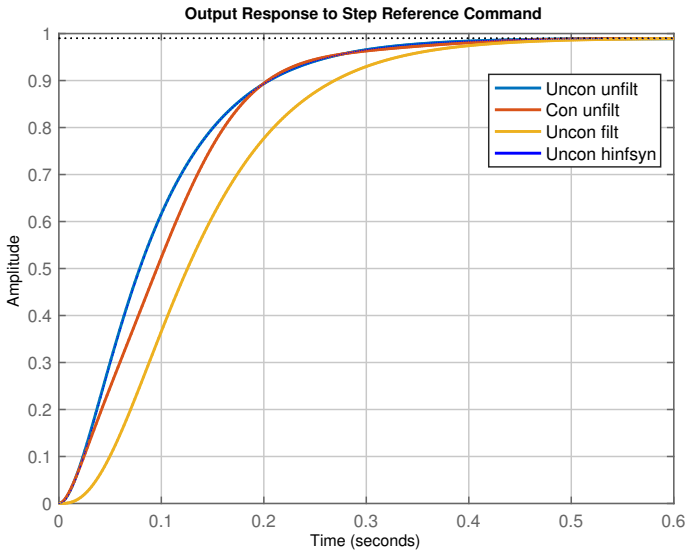


Figure 6.11: Imposing Time-Domain Constraint: Output Response to Step Reference Command

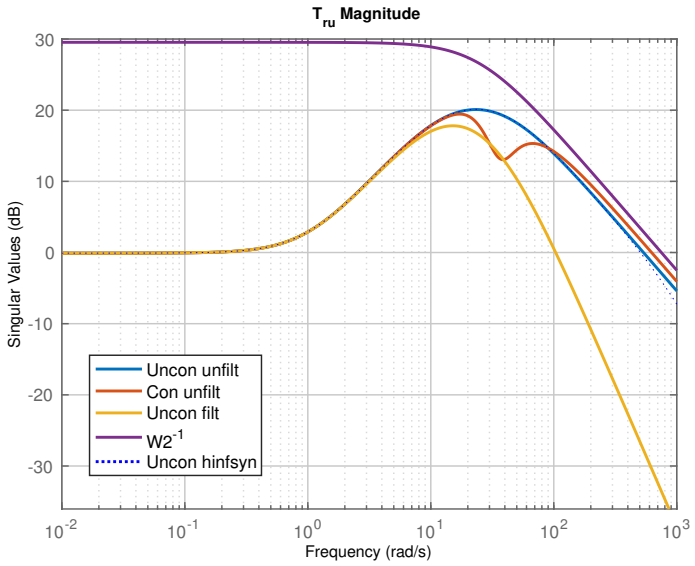


Figure 6.12: Imposing Time-Domain Constraint: Magnitude of T_{ru}

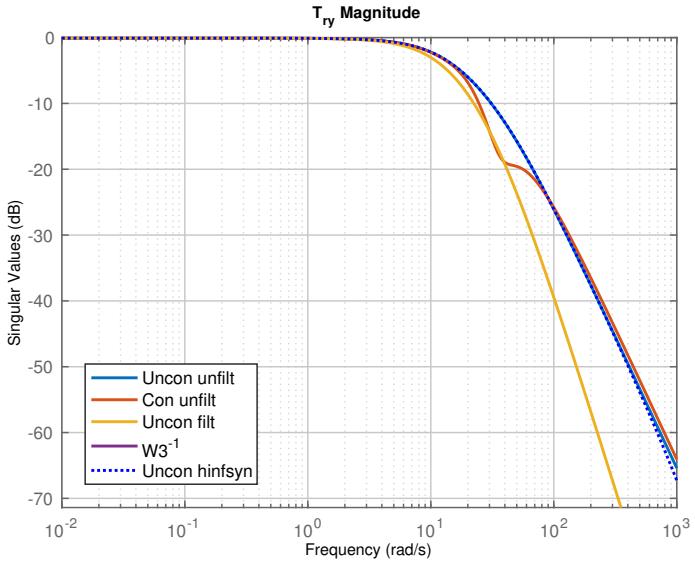


Figure 6.13: Imposing Time-Domain Constraint: Magnitude of T_{ry}

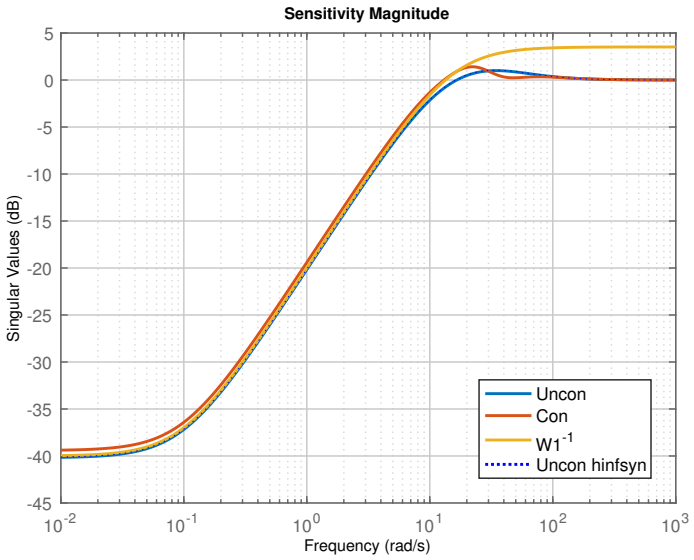


Figure 6.14: Imposing Time-Domain Constraint: Magnitude of Sensitivity

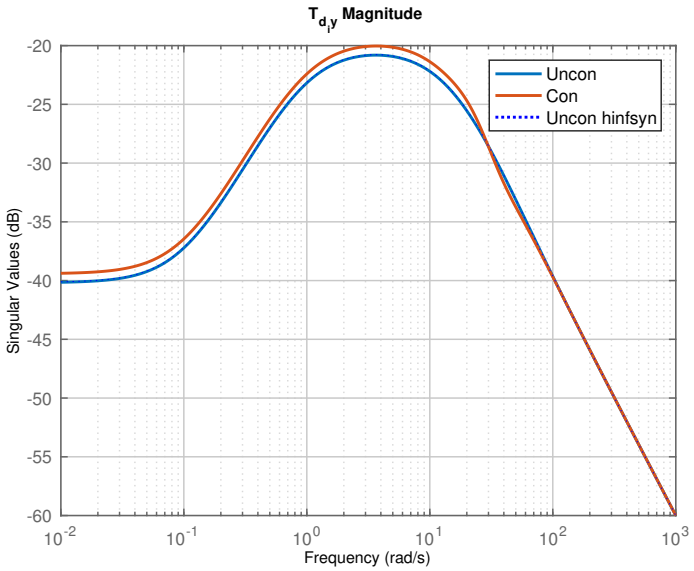


Figure 6.15: Imposing Time-Domain Constraint: Magnitude of $T_{d_i y}$

6.5 Simple Nominal Open Loop $\left(L_o = \frac{1}{s}\right)$ with Challenging Specifications

In this section, we design controllers for and study the corresponding feedback properties of a simple plant ($P = 1$), that try to satisfy certain challenging design specifications. We see that even for a simple system (e.g., $P = 1$), depending on the specifications, the control problem may become very hard. Consider a nominal design specification on open loop transfer function in which $|L(j0.1)| \geq 20dB$ and $|L(j10)| \leq -20dB$. For this design, a trivial controller $K = \frac{1}{s}$ satisfies the specification. But as this specification is made severe, e.g., $|L(j0.1)| \geq MdB$ and $|L(j10)| \leq -MdB$ where $M = 21, 22, 23$, etc., the problem gets more complicated and results in trading off robustness properties (e.g. phase margin, peak sensitivity). Using GMS methodology, we shape the closed loop properties (S and T) to achieve the above specification on L indirectly using weighting functions as shown in Equations (6.17) and (6.18). Note that S (or T) here can be S_e (or T_e) or S_c (or T_c) as the plant is SISO and we have considered standard P-K feedback structure.

$$W_1(s) = \frac{1}{M_s}\left[\frac{\left(s + \sqrt[m_1]{M_s}\omega_b\right)^{m_1}}{\left(s + \frac{\omega_b\epsilon}{(0.1)^{m_1-1}}\right)(s + 0.1\omega_b)^{m_1-1}}\right] \tag{6.17}$$

$$W_3(s) = \frac{1}{\epsilon}\left[\frac{\left(s + \frac{\omega_{bc}}{\sqrt[m_3]{M_y}}\right)^{m_3}}{(s + 10\omega_{bc})^{m_3-1}\left(s + \frac{\omega_{bc}}{10^{m_3-1}\epsilon}\right)}\right] \tag{6.18}$$

The weighting function parameters selected are $\epsilon = 10^{-5}(-100dB), M_s = 2, \omega_b = 1/2$, $M_y = 2, \omega_{bc} = 2$. The slopes associated with the weighting function m_1 and m_2 are varied based on the specification. We solve the following multiobjective minimization problem using our GMS methodology.

$$K = \arg\left\{\min_{\substack{K \\ stabilizing}} \gamma \middle| \max\left(\|W_1 S\|_{\mathcal{H}^\infty}, \|W_3 T\|_{\mathcal{H}^\infty}\right) < \gamma\right\} \tag{6.19}$$

Depending on the slopes (m_1 and m_2 in Equations (6.17) and (6.18)) of magnitudes of weighting functions, the specification is relaxed or made strict. For the case when $m_1 = m_2 = 1$, a trivial controller of $K = \frac{1}{s}$ achieves the specification and has good robustness properties (peak sensitivities and phase margin (PM)) as shown in Table 6.1. But as these values are increased, the robustness properties become worse. This can be seen in the cases when k_1 and k_2 are higher, for e.g., equal to 4 and 5. Figures 6.16 - 6.25 show the corresponding closed loop sensitivity and open loop magnitudes. Also, the weighting functions can be visualized using the figures provided. The above problem 6.19 is solved using our GMS methodology. The basis (fixed pole all-pass) parameters chosen are $\alpha = 2$, $N = 10$. Initial controller and observer gains used for Youla et al, parameterization are obtained using LQG ideas with the weighting matrices all equal to identity. The upper and lower bounds on optimization variables are picked to be ± 100 resectively, with an initial value of 1. Further, during the optimization problem setup the design plant is made to be $P_d = \frac{1}{s}$. The resulting controller is then augmented with this integrator $\frac{1}{s}$ and then used along with the original plant P to form the feedback loop. The MATLAB code used to generate the values in the table and in Figures 6.16 - 6.25 can be found in Appendix A.1.

k_1 & k_2	$\lvert L(j0.1)\rvert$ (dB)	$\lvert L(j10)\rvert$ (dB)	$\lVert S\rVert_{\mathcal{H}^\infty}$ (dB)	$\lVert T\rVert_{\mathcal{H}^\infty}$ (dB)	PM (deg)
1	20.00	-20.00	0.00	0.00	90.00
2	31.34	-31.44	2.83	2.77	46.89
3	40.74	-41.11	7.22	6.94	26.82
4	50.00	-48.67	10.27	9.87	18.62
5	57.22	-52.78	11.33	10.95	16.41

Table 6.1: $\frac{1}{s}$ Example: OL and CL Properties for Various Specifications

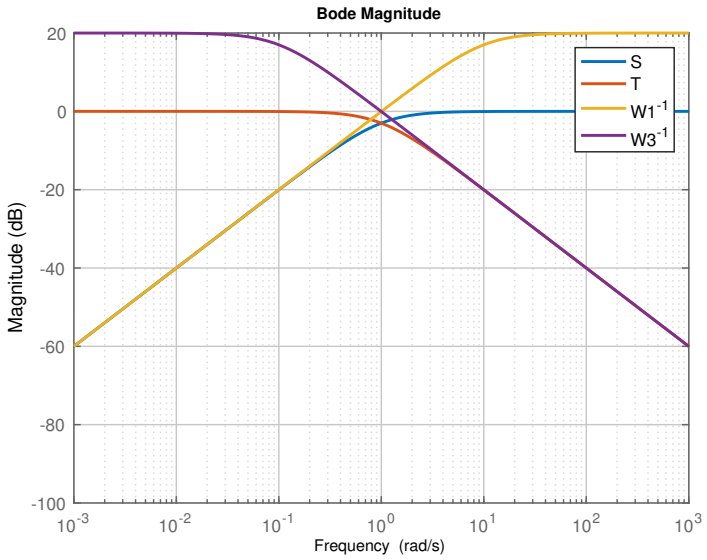


Figure 6.16: $\frac{1}{s}$ Example: Sensitivity Magnitudes for Weighting Function Slopes $m_1 = m_3 = 1$

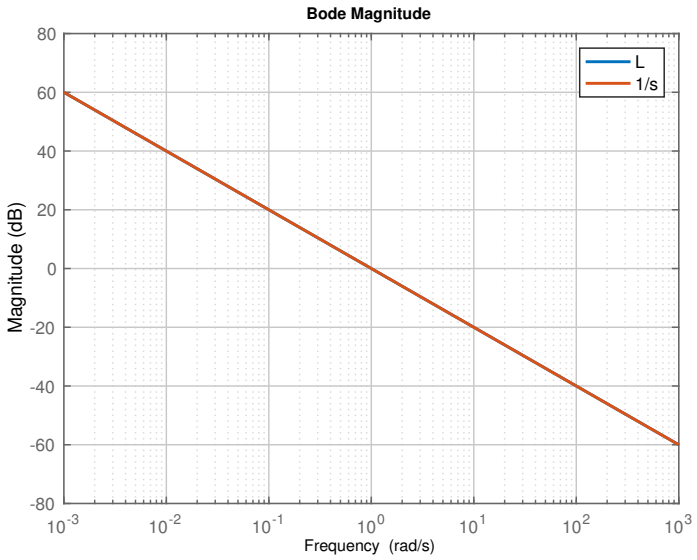


Figure 6.17: $\frac{1}{s}$ Example: Open Loop and Integrator Magnitudes for Weighting Function Slopes $m_1 = m_3 = 1$

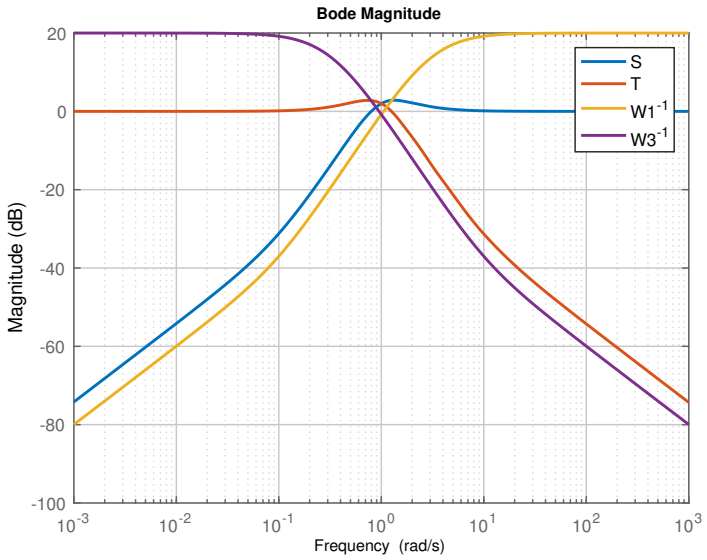


Figure 6.18: $\frac{1}{s}$ Example: Sensitivity Magnitudes for Weighting Function Slopes $m_1 = m_3 = 2$

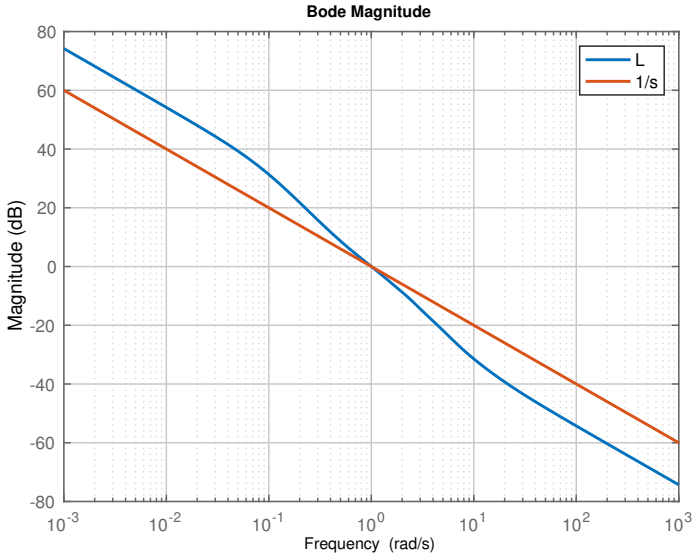


Figure 6.19: $\frac{1}{s}$ Example: Open Loop and Integrator Magnitudes for Weighting Function Slopes $m_1 = m_3 = 2$

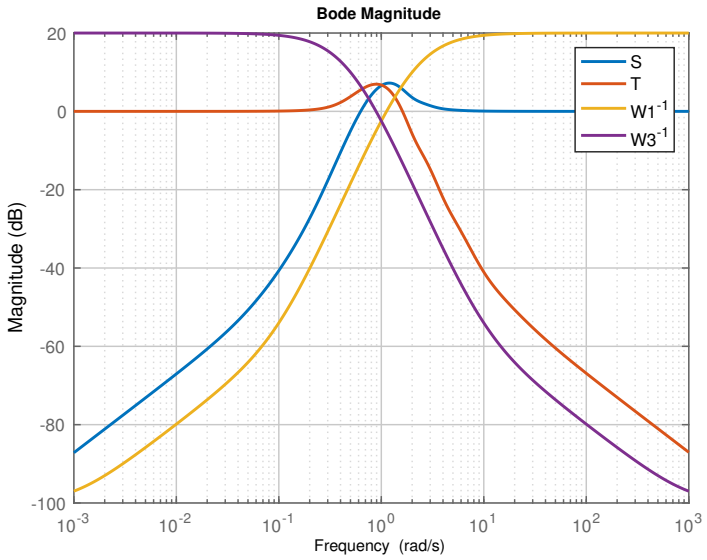


Figure 6.20: $\frac{1}{s}$ Example: Sensitivity Magnitudes for Weighting Function Slopes $m_1 = m_3 = 3$

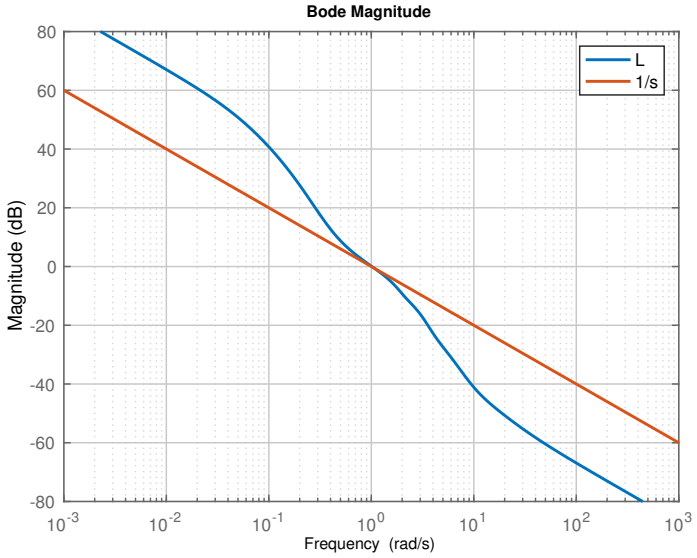


Figure 6.21: $\frac{1}{s}$ Example: Open Loop and Integrator Magnitudes for Weighting Function Slopes $m_1 = m_3 = 3$

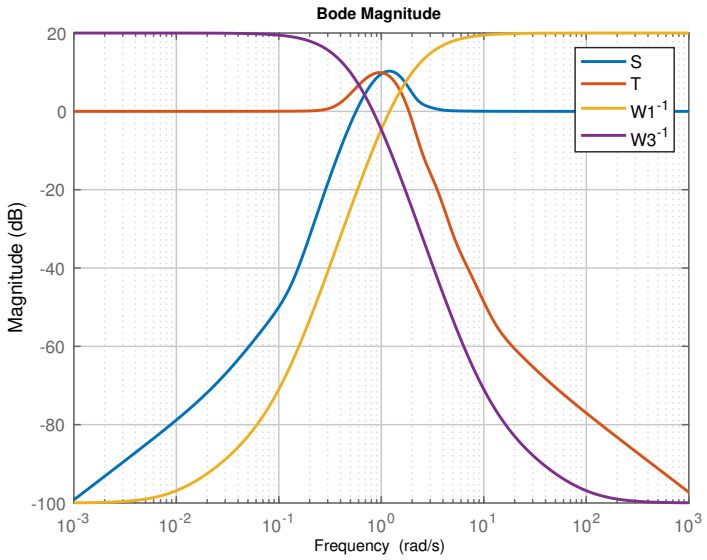


Figure 6.22: $\frac{1}{s}$ Example: Sensitivity Magnitudes for Weighting Function Slopes $m_1 = m_3 = 4$

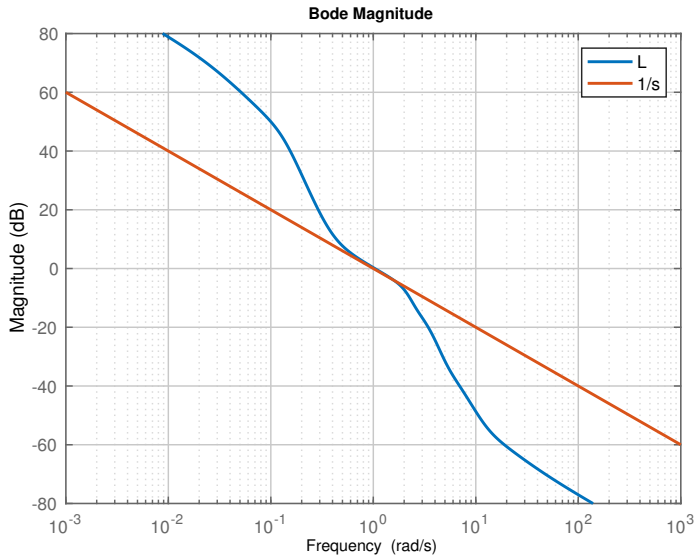


Figure 6.23: $\frac{1}{s}$ Example: Open Loop and Integrator Magnitudes for Weighting Function Slopes $m_1 = m_3 = 4$

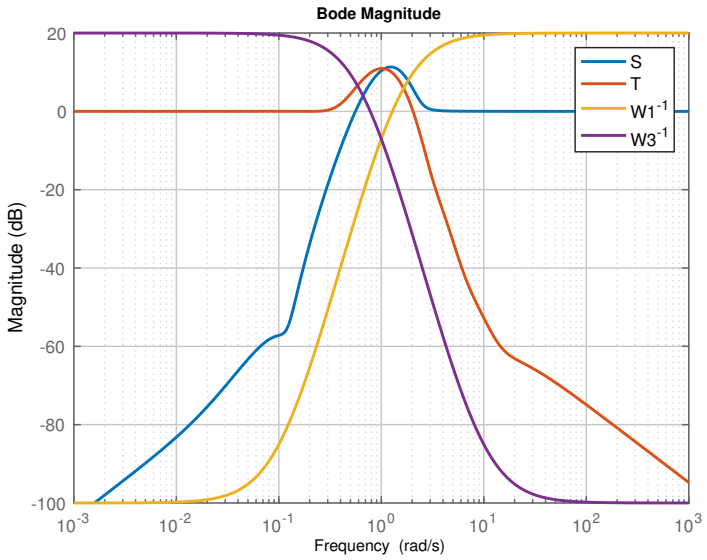


Figure 6.24: $\frac{1}{s}$ Example: Sensitivity Magnitudes for Weighting Function Slopes $m_1 = m_3 = 5$

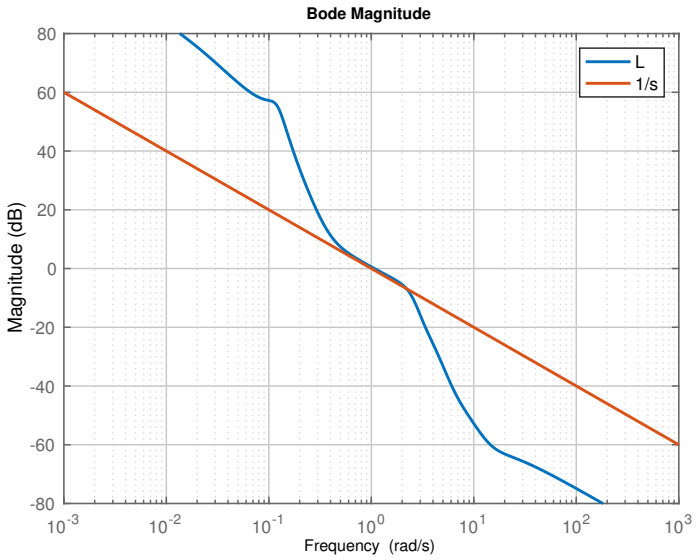


Figure 6.25: $\frac{1}{s}$ Example: Open Loop and Integrator Magnitudes for Weighting Function Slopes $m_1 = m_3 = 5$

6.6 SISO Unstable and Non-minimum Phase Plant with Standard P-K versus Inner-Outer Loop Feedback

In this section, we study the limitations on peak sensitivity and complementary sensitivity imposed by RHP pole and RHP zero present in the plant. It is shown how inner loop may be used to stabilize the system without the bandwidth limitation imposed by this RHP zero. Design limitations due to the unstable pole remain, but are now less severe. The sensitivities must still satisfy the design limitations imposed bode integrals, but these design limitations are less severe with the additional measurement (inner-loop). This was also studied in the paper [222] by Freudenberg et al. Consider a SISO plant (P) having a RHP pole and a RHP zero as in Equation 6.20 below.

$$P = \frac{p\ (z - s)}{z\ (s - p)} \tag{6.20}$$

Let the pole (p) be located at 1. We consider the location of zero (z) to be 10, 5 and 2 in the same order. Here, we intend to find the following:

1. minimum achievable peak sensitivity ($\|S\|_{\mathcal{H}^\infty}$).
2. minimum achievable peak complementary sensitivity ($\|T\|_{\mathcal{H}^\infty}$).
3. minimum achievable "equilibrated" peak sensitivity & peak complementary sensitivity ($\max\{\|S\|_{\mathcal{H}^\infty}, \|T\|_{\mathcal{H}^\infty}\}$).

Note that S and T here refer to S_e and T_e respectively. We use both the standard P-K structure and hierarchical inner-outer structure as shown in Figures 6.26.

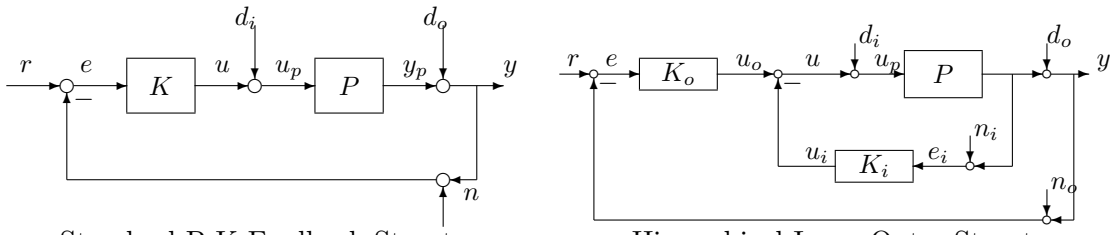


Figure 6.26: Visualization of Feedback Structures

Standard P-K Feedback Structure: For simplicity, we consider a proportional controller ($K = k_p,$ where $k_p \in \mathcal{R}$). The optima (minimum peaks) achieved for different values of RHP zero (z) are shown in Table 6.2.

z	$\left\|\frac{z+p}{z-p}\right\|$	To Minimize	Min Val (dB)	k_p
10	1.7430	$\|S\|_{\mathcal{H}^\infty}$	1.7430	1.8182
		$\|T\|_{\mathcal{H}^\infty}$	1.7430	5.5
		$\max\{\|S\|_{\mathcal{H}^\infty}, \|T\|_{\mathcal{H}^\infty}\}$	3.3018	3.1623
5	3.5218	$\|S\|_{\mathcal{H}^\infty}$	3.5218	1.6667
		$\|T\|_{\mathcal{H}^\infty}$	3.5218	3
		$\max\{\|S\|_{\mathcal{H}^\infty}, \|T\|_{\mathcal{H}^\infty}\}$	5.1489	2.2361
2	9.5424	$\|S\|_{\mathcal{H}^\infty}$	9.5424	1.3333
		$\|T\|_{\mathcal{H}^\infty}$	9.5424	1.5
		$\max\{\|S\|_{\mathcal{H}^\infty}, \|T\|_{\mathcal{H}^\infty}\}$	10.6658	1.4142

Table 6.2: SISO Unstable and Non-Minimum Phase Plant Example: Critical Control Properties & Corresponding Controller Parameters Using Standard P-K Structure

The values in Table 6.2 were obtained by solving the $\mathcal{H}^\infty$ based problems using MATLAB's *hinfstruct* [154, 223–226]. The MATLAB code used to generate the above

results is given in Appendix A.4. Within the code, the designs can be selected to be standard P-K or hierarchical inner-outer structure by assigning a value of 1 or 2 respectively to the variable *flag* as explained in the code.

Hierarchical Inner-Outer Structure: We consider proportional controllers in inner ($K_i = k_{i_p},$ where $k_{i_p} \in \mathcal{R}$) and outer ($K_o = k_{o_p},$ where $k_{o_p} \in \mathcal{R}$) loops respectively. The optima (minimum peaks) achieved for different values of RHP zero (z) are shown in Table 6.3. P_{mod} in the table corresponds to the feedback loop formed by the inner-loop controller (i.e., $P_{mod} = feedback(P, K_i)$).

z	$\left\lvert\frac{z+p}{z-p}\right\rvert$	To Minimize	Min Val (dB)	k_{i_p}	k_{o_p}	P_{mod}
10	1.7430	$\|S\|_{\mathcal{H}^\infty}$	≈ 0	2.25	≈ 0	$\frac{-0.12904(s-10)}{(s+1.614)}$
		$\|T\|_{\mathcal{H}^\infty}$	$\approx -\infty$	4.891	≈ 0	$\frac{-0.21136(s-10)}{(s+9.023)}$
		$\max\{\|S\|_{\mathcal{H}^\infty}, \|T\|_{\mathcal{H}^\infty}\}$	0	2.25	≈ 0	$\frac{-0.12904(s-10)}{(s+1.614)}$
5	3.5218	$\|S\|_{\mathcal{H}^\infty}$	≈ 0	1.223	≈ 0	$\frac{-0.26473(s-5)}{(s+0.2946)}$
		$\|T\|_{\mathcal{H}^\infty}$	$\approx -\infty$	5	≈ 0	$\frac{-0.44399(s-5)}{(s+3.88)}$
		$\max\{\|S\|_{\mathcal{H}^\infty}, \|T\|_{\mathcal{H}^\infty}\}$	0	1.311	≈ 0	$\frac{-0.27107(s-5)}{(s+0.4213)}$
2	9.5424	$\|S\|_{\mathcal{H}^\infty}$	≈ 0	1.732	≈ 0	$\frac{-1.755(s-2)}{(s+1.51)}$
		$\|T\|_{\mathcal{H}^\infty}$	$\approx -\infty$	1.618	≈ 0	$\frac{-2.3109(s-2)}{(s+2.622)}$
		$\max\{\|S\|_{\mathcal{H}^\infty}, \|T\|_{\mathcal{H}^\infty}\}$	0	1.942	≈ 0	$\frac{-1.2445(s-2)}{(s+0.4889)}$

Table 6.3: SISO Unstable and Non-Minimum Phase Plant Example: Critical Control Properties & Corresponding Controller Parameters Using Inner-Outer Structure

The values in Table 6.3 were obtained by solving the $\mathcal{H}^\infty$ based problems using MATLAB's *hinfstruct* [154, 223–226]. The MATLAB code used to generate the above results is given in Appendix A.4. Within the code, the designs can be selected to be standard P-K or hierarchical inner-outer structure by assigning a value of 1 or 2

respectively to the variable *flag* as explained in the code. It can be seen from the table that the limitation on peak sensitivities $\left(\left|\frac{z+p}{z-p}\right|\right)$ can be avoided by using inner-loop controller. This is shown by the low peak sensitivity values in each of the cases in Table 6.3.

A More Realistic Design Using Hierarchical Inner-Outer Structure: Here, we minimize the weighted sensitivities to instead of unweighted ones. The weighting functions (W_1 and W_3) used to shape S and T respectively are as follows:

$$W_1 = \frac{1}{M_s}\left[\frac{s+M_s\omega_b}{s+\omega_b\epsilon}\right] \tag{6.21}$$

$$W_3 = \frac{s+\frac{\omega_{bc}}{M_y}}{\epsilon s+\omega_{bc}} \tag{6.22}$$

where $M_s = M_y = 2$, $\omega_b = \omega_{bc} = 0.02$ and $\epsilon = 0.01$.

We again consider proportional controllers in inner ($K_i = k_{i_p},$ where $k_{i_p} \in \mathcal{R}$) and outer ($K_o = k_{o_p},$ where $k_{o_p} \in \mathcal{R}$) loops respectively. The optima (minimum peaks) achieved for different values of RHP zero (z) are shown in Table 6.4. P_{mod} in the table corresponds to the feedback loop formed by the inner-loop controller (i.e., $P_{mod} = feedback(P, K_i)$).

z	To Minimize	Min Val (dB)	k_{i_p}	k_{o_p}	P_{mod}
10	$\max\{\|W_1S\|_{\mathcal{H}^\infty}, \|W_3T\|_{\mathcal{H}^\infty}\}$	-0.0853	1	0.01794	$\frac{-0.11111(s-10)}{(s+0.0001518)}$
5	$\max\{\|W_1S\|_{\mathcal{H}^\infty}, \|W_3T\|_{\mathcal{H}^\infty}\}$	-0.0743	1	0.01593	$\frac{-0.0039976(s-5)}{(s+0.02014)}$
2	$\max\{\|W_1S\|_{\mathcal{H}^\infty}, \|W_3T\|_{\mathcal{H}^\infty}\}$	-0.0142	1	0.009884	$\frac{-0.0099837(s-2)}{(s+0.02012)}$

Table 6.4: SISO Unstable and Non-Minimum Phase Plant Example: Properties & Parameters for Realistic Design Using Inner-Outer Structure

Figures 6.27 - 6.29 show magnitudes of senstivity and complementary sensitivity functions for the three designs in Table 6.4. The values in Table 6.3 and the Figures 6.27

- 6.29 were obtained using the MATLAB code given in Appendix A.4.

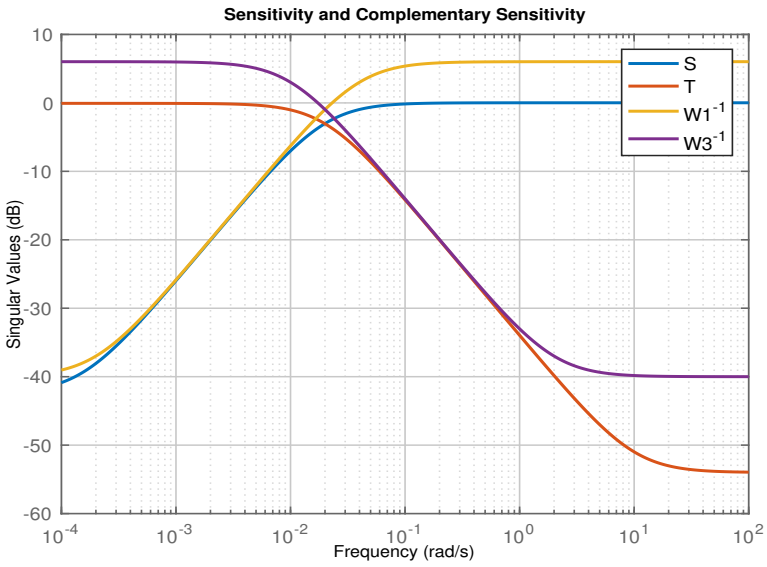


Figure 6.27: Weighted Sensitivity Minimization Example for SISO Plant with RHPP=1 and RHPZ=10: Sensitivity and Complementary Sensitivity

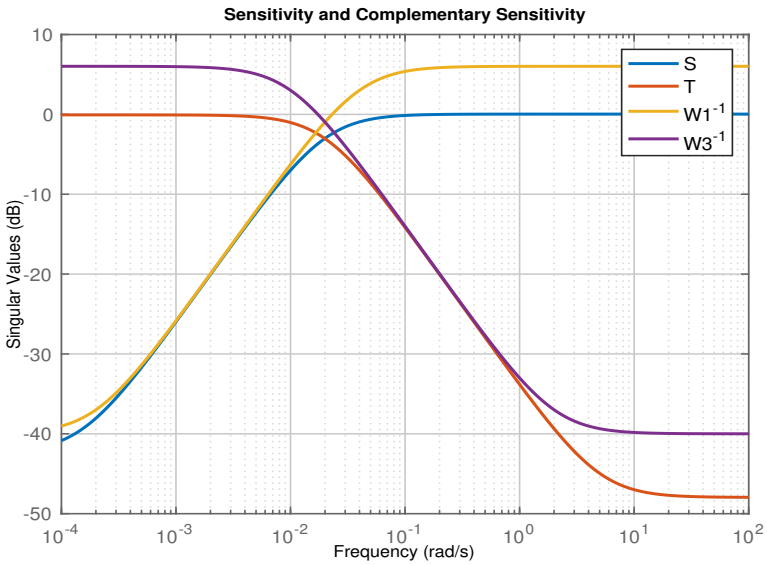


Figure 6.28: Weighted Sensitivity Minimization Example for SISO Plant with RHPP=1 and RHPZ=5: Sensitivity and Complementary Sensitivity

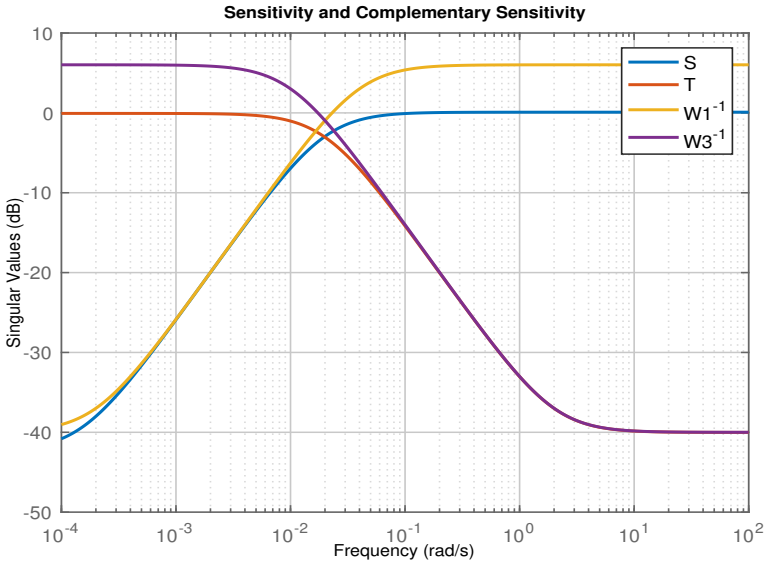


Figure 6.29: Weighted Sensitivity Minimization Example for SISO Plant with RHPP=1 and RHPZ=2: Sensitivity and Complementary Sensitivity

6.7 μ-Synthesis Using GMS: Toward D-Q Iteration

The $\mathcal{H}^\infty$ control methodology provides a powerful framework to analyze and design controllers for a very large performance characteristics and robustness to uncertainty requirements. Open and closed loop responses are shaped based on these specifications using $\mathcal{H}^\infty$ methods. Though it is a powerful tool when unstructured uncertainties are considered [147, 148], $\mathcal{H}^\infty$ methodology is unable to handle structured uncertainties [227], i.e. the designs may become conservative when the plant uncertainties are structured [228–230]. The structured singular value (μ) was developed by Doyle et al. [22] (given in Definition 3.5.4) as a measure for robustness to these structured uncertainties. To design controllers that (approximately) minimize the μ (μ-synthesis problem), DK-iteration technique has been widely used [9, 226, 227, 231–233]. It is important to note here that the $\mathcal{H}^\infty$ design methods are still used within the μ-synthesis design methodology [227]. It essentially integrates the $\mathcal{H}^\infty$ techniques for synthesis and μ techniques for analysis. The μ-synthesis problem can be posed as follows

$$\min_K \left(\min_D \quad \left\| D\mathcal{F}_l(N,K)D^{-1} \right\|_{\mathcal{H}^\infty} \right) \tag{6.23}$$

where D is diagonal scaling matrix, $\mathcal{F}_l$ denotes lower Linear Fractional Transformation (LFT) [221] and N is as shown in Figure 6.30. Here, w & z are exogeneous signals, and u_Δ & y_Δ correspond to signals from and to block diagonal uncertainty Δ (problem dependent) respectively, which is described in Equation 3.27.

In what follows, we consider a 2×2 plant along with divisive uncertainties simultaneously at plant input and output such that $\Delta = \text{diag}\,(\Delta_i, \Delta_o)$. All uncertainties present anywhere in the feedback system (e.g. Δ_i, Δ_o) can be grouped in block-diagonal form by associated matrix $M = \mathcal{F}_l(N,K)$. Hence μ allows us to nonconservatively analyze simultaneous occurances of them [15]. As a performance measure we use mixed

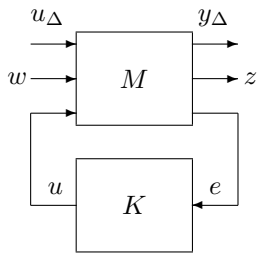


Figure 6.30: Visualization of Control Configuration for μ-Synthesis

weighted sensitivities $W_1 S_e$ and $W_2 K S_e$. As robust stability measure we use weighting functions on these uncertainties. The TFMs of the plant and the weighting functions are as follows.

$$P = \frac{1}{s}\begin{bmatrix} 10 & 9 \\ 9 & 8 \end{bmatrix} \tag{6.24}$$

$$W_1 = \frac{1}{M_s}\left(\frac{s + M_s\omega_b}{s + \epsilon\omega_b}\right) I_{2\times 2} \tag{6.25}$$

$$W_2 = 0.5\ I_{2\times 2} \tag{6.26}$$

$$W_i = I_{2\times 2} \tag{6.27}$$

$$W_o = I_{2\times 2} \tag{6.28}$$

where $M_s = 2$, $\omega_b = 0.1$ and $\epsilon = 0.01$. Δ_i and Δ_o are complex uncertainty blocks. This problem can readily be solved by using DK-iteration technique in MATLAB (using *dksyn* command) [226]. Figures 6.31 - 6.33 show the resulting properties. The MATLAB code used is given in Appendix A.6.

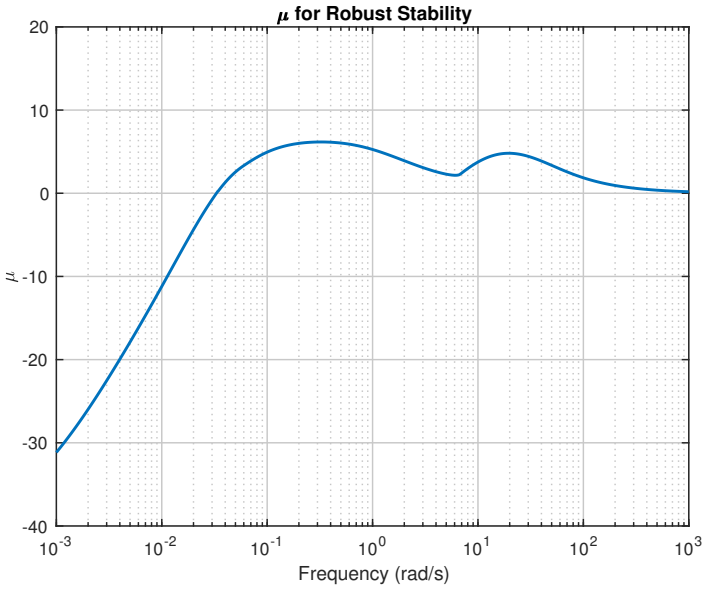


Figure 6.31: DK-Iteration: μ for Robust Stability

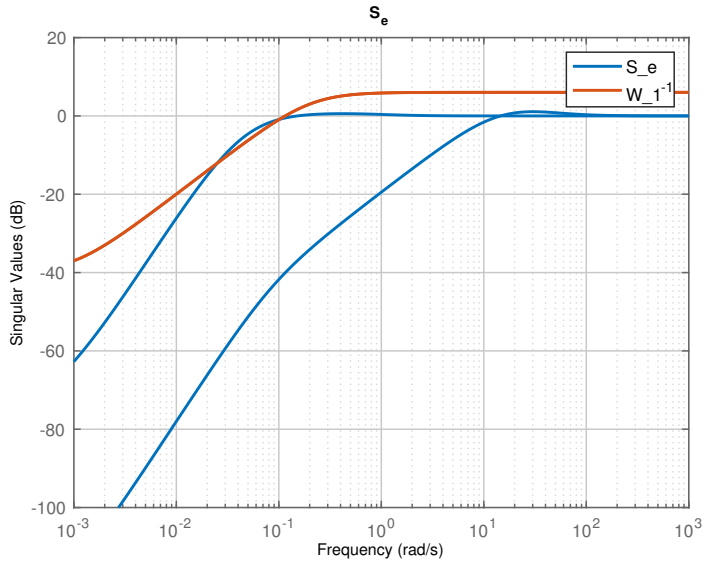


Figure 6.32: DK-Iteration: S_e

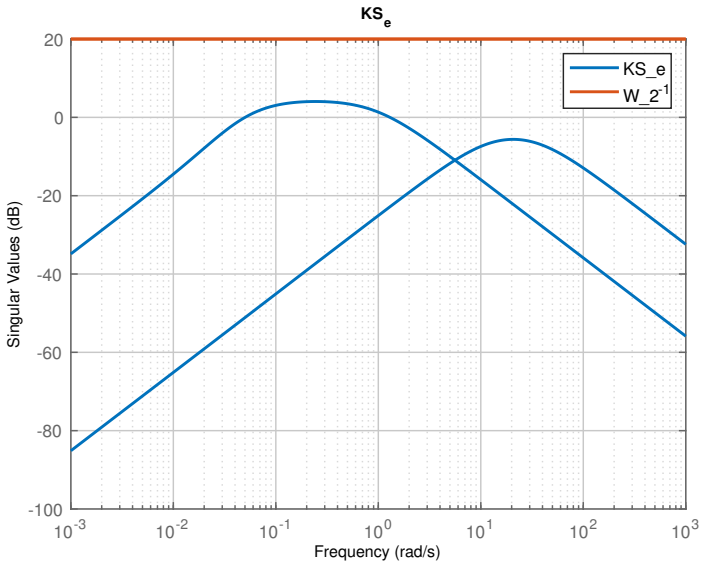


Figure 6.33: DK-Iteration: KS_e

Tsai et al. [228–230] proposed a D-Q iteration procedure that uses Youla et al. parameterization [167–169]. It includes a D-step and a Q-step. In the D-step, the D-scale at some predetermined frequencies is optimized. This step is similar to that in D-K iteration. But D-scale fitting is not required. In the Q-step, using D-scale obtained in the previous step, the problem is approximated by a finite-dimensional Q-parameter optimization problem at same frequencies as before. As part of future work, the GMS methodology can be extended to include the above using some ideas presented in [234–237].

6.8 Forming the Design Plant

6.8.1 Design Plants with Integrator Augmentation

In order to guarantee zero steady state to step reference commands, in many cases, we augment the plant's actual outputs with integrators. The set of stabilizing controllers

is obtained for this augmented plant. At this point the integrator is in P - not in K. To shape the true reference to controls map, we use $T_{ru}(\text{unfiltered}) = (KS_e)_{design} \times \frac{I}{s+\epsilon}$ where ϵ is a small number. Similarly, we use $T_{d_{iy}}(\text{unfiltered}) = (PS_c)_{design} \times (s+\epsilon)$.

6.8.2 Bilinear Transformation

A caveat of of the $\mathcal{H}^\infty$ design methodology is the inversion of invert plant dynamics. For example, $\mathcal{H}^\infty$ controller - if permitted - may try to invert plant poles that are near the imaginary axis. This is typically highly undesirable. When plant poles near the imaginary axis are present, one can use a bilinear transformation method to prevent undesirable inversions by the resulting compensator. Bilinear transformation may be used in such cases to define a shifted (towards left-side along σ axis). stability region [133, 159]. This is made use of in Chapter 7. Mention why and how bilinear transformation is used in $\mathcal{H}^\infty$ Mixed Sensitivity Problems.

6.9 Summary and Conclusions

In this chapter, we presented several control problems with different objective and constraint functions to illustrate critical control-relevant tradeoffs associated with them. Firstly, tradeoffs associated with MIMO ill-conditioned plants that are not typically seen in SISO systems were illustrated using examples. SISO example was used to illustrates handling $\mathcal{L}^\infty$ time-domain constraint using GMS. Further, some SISO problems were presented that discuss specific control challenges and tradeoffs.

Chapter 7

CONTROL OF LONGITUDINAL DYNAMICS OF HYPERSONIC VEHICLE

7.1 Overview

We design an inner-outer loop controller for the longitudinal dynamics of a scramjet-powered hypersonic vehicle flying at Mach 8, 85 kft. The control system will seek good properties at input/control, output/error as well as at the inner-loop sensor noise. Since the scramjet-powered Mach 7/10 flights of X-43A in 2004 [64–66], the research on hypersonic vehicles has seen a resurgence [62, 63, 67–85]. With the recent successful X-51A flight test (May, 2013), the hypersonic application considered here is timely. Air-breathing hypersonic propulsion is viewed as the next critical step toward achieving (1) reliable, affordable, routine access to space, as well as (2) global reach vehicles. There are commercial and as military implications to both objectives. Rocket-based (combined cycle) propulsion systems are much more expensive to operate because they must carry oxygen. This is particularly costly when traveling at lower altitudes through the troposphere (i.e. below 36,152 ft). They do not exhibit the desired levels of reliability and flexibility (e.g. airplane like takeoff and landing options) either. As a result, much emphasis has been placed on two-stage-to-orbit (TSTO) designs that involve a turbo-ram-scramjet combined cycle first stage and a rocket second stage. This research focuses on control challenges associated with scramjet-powered hypersonic vehicles. Such vehicles are characterized by significant aero-thermo-elastic-propulsion interactions and uncertainty.

7.2 Longitudinal Dynamics Model

The longitudinal equations of motion for our 3-DOF (including flexible modes) hypersonic vehicle are given as follows [82–85]:

$$\dot{v} = \left[\frac{T\cos\alpha - D}{m}\right] - g\sin\gamma \tag{7.1}$$

$$\dot{\gamma} = \left[\frac{L + T\sin\alpha}{mv}\right] - \left[\frac{g}{v} - \frac{v}{R_E + h}\right]\cos\gamma \tag{7.2}$$

$$\dot{q} = \frac{\mathcal{M}}{I_{yy}},\ \dot{h} = v\sin\gamma,\ \dot{\theta} = q,\ \alpha \stackrel{\text{def}}{=} \theta - \gamma \tag{7.3}$$

$$\ddot{\eta}_i = -2\zeta\omega_i\dot{\eta}_i - \omega_i^2\eta_i + N_i,\ g = g_0\left[\frac{R_E}{R_E + h}\right]^2 \tag{7.4}$$

where h denotes altitude, L denotes lift, T denotes engine thrust, D denotes drag, $\mathcal{M}$ is the pitching moment, N_i denotes generalized forces, ζ and ω_i denote flexible mode dampic factor and undamped natural frequency respectively, $i = 1, 2, 3$, m denotes the vehicle's total mass, I_{yy} is the pitch axis moment of inertia, g_0 is the acceleration due to gravity at sea level, and R_E is the radius of the Earth. For our 100 ft vehicle, the weight per unit width is 6138 lb/ft and $I_{yy} = 86,698$ slugs ft^2/ft.

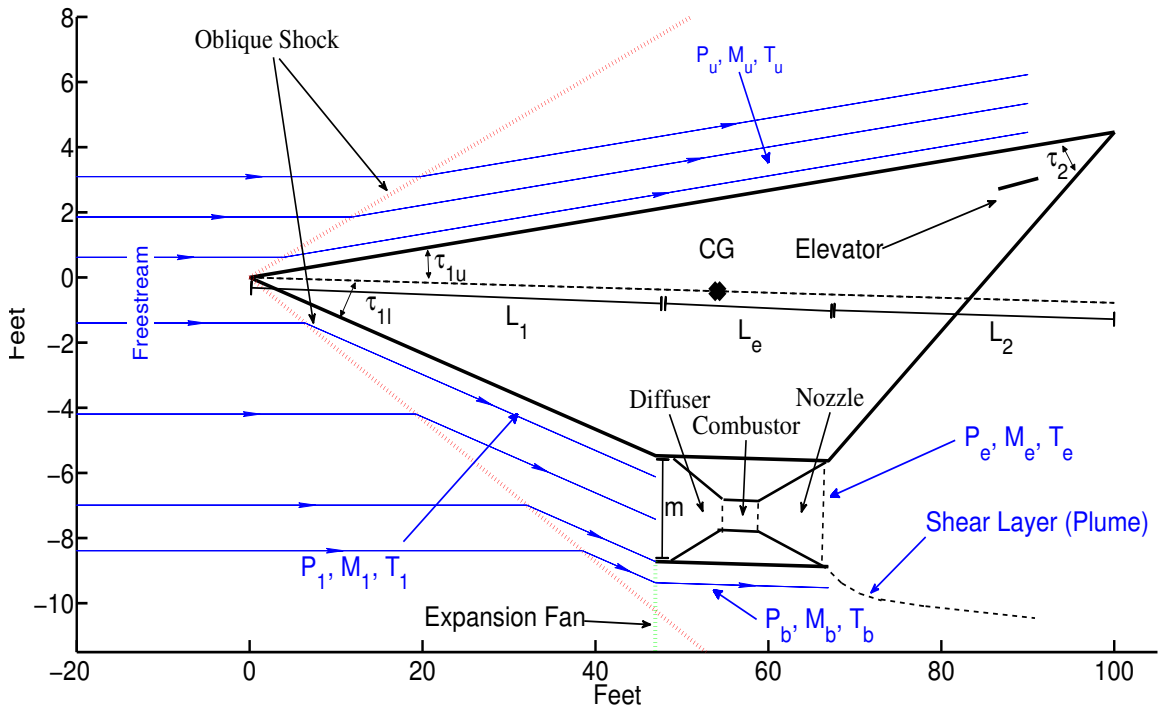


Figure 7.1: Schematic of Hypersonic Scramjet Vehicle

States and Controls. The vehicle possesses ten (10) states. They are: velocity v, flight path angle (FPA) γ, pitch rate q, pitch angle θ, and the 6 flexible body states $\eta_1, \dot{\eta}_1, \eta_2, \dot{\eta}_2, \eta_3, \dot{\eta}_3$. The vehicle has two (2) control inputs: a rearward situated elevator δ_e and stoichiometrically normalized fuel equivalence ratio (FER). The "new engine model" introduced within [79] is used in this work. Within [62], two plume models were considered: (1) an "old plume" model and (2) a "new plume" model. Here, the designation "old plume" refers to a plume whose shape is independent of the flight condition [82–85, 238]. The "new plume" model addresses this by employing Newtonian impact theory to compute aft body pressures [73]. Both nonlinear models were linearized about a Mach 8, 85kft level flight condition [62]. The "new plume" model has high coupling - especially at low frequencies. The condition number varies between 32dB and 65dB, with the peak occuring at 0.01rad/s. For this reason, we

focus exclusively on the new plume model. The associated poles and transmission zeros [133, 146, 239] for the flexible new plume model are given in Table 7.1. We observe that the model is unstable and non-minimum phase (NMP) - unstable because of a forward situated center of pressure associated with the vehicle's shovel-nosed forebody compression ramp; NMP because of the inverse flight-path response to elevator deflection. For simplicity, we consider the vehicle body to be rigid. This makes the plant a 4-state system, which neglects the flexible modes. It must be noted that, despite this assumption, the system still imposes an upper limit on the closed loop bandwidth because of the presence of NMP zero. The presense of right half plane pole imposes a lower limit on the bandwidth. These critical aspects of the plant make it a challenging control problem. Further, due to the strong interactions (coupling) between channels, the use of multivariable (not decentralized) control is justified [63]. In [63], several controller structures were analyzed. It was shown that though decentralized control designs were stable, the performance and robustness were uacceptable [12, 240]. From frequency-domain analysis it was shown using classical ideas, it was found that the closed loop performance strongly depends on the (1,2) element of the inner loop controller. This further justifies using a multivariable controller. Note that the transfer function corresponding to the (2,1)-element (i.e., $T : FER \rightarrow \gamma$) is strong. Table 7.2 shows the poles and transmission zeros of the model when flexible modes are neglected. We call this the "rigid model". We analyze this plant and design controllers using different methodologies for the model at Mach 8, 85kft.

Pole	Damping	Frequency (rad/sec)
$(-0.527 \pm j9.57) \times 10^{-3}$	0.0549	0.00958
2.3 (Unstable)	-1	2.3
-2.4	1	2.4
-0.397 ± j22	0.0180	22
-0.957 ± j48.2	0.0198	48.3
-1.9 ± j94.8	0.02	94.8
Transmission Zero	Damping	Frequency (rad/sec)
7.71 (NMP)	-1	7.71
-7.72	1	7.72
-0.552 ± j19.5	0.0283	19.5
-0.959 ± j48.8	0.0197	48.8
-1.9 ± j95	0.02	95

Table 7.1: Poles and Transmission Zeros of Flexible Model at Mach 8, 85kft

Pole	Damping	Frequency (rad/sec)
$(-0.526 \pm j9.57) \times 10^{-3}$	0.0549	0.00958
2.30 (Unstable)	-1	2.30
-2.39	1	2.39
Transmission Zero	Damping	Frequency (rad/sec)
7.74 (NMP)	-1	7.74
-7.74	1	7.74

Table 7.2: Poles and Transmission Zeros of Rigid Model at Mach 8, 85kft

The singular value and Bode magnitude plots of the rigid body model are shown in Figures 7.2 and 7.3 respectively.

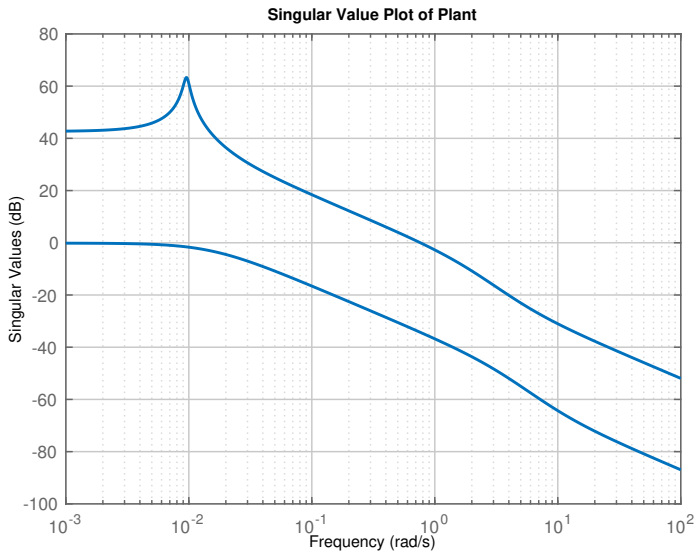


Figure 7.2: Hypersonic Vehicle Model: Singular Value Plot

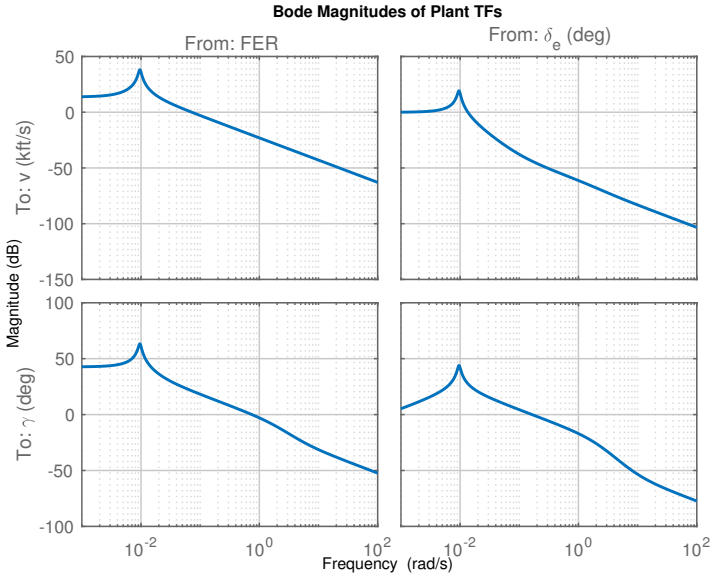


Figure 7.3: Hypersonic Vehicle Model: Bode Magnitude Plot

Figure 7.4 shows the condition number of the plant model at different frequencies. It is observed that it varies between 30dB and 60dB. Figures 7.5 - 7.7 show the RGA and scaled condition number properties of the plant. These figures are obtained using the MATLAB code given in Appendix A.9.

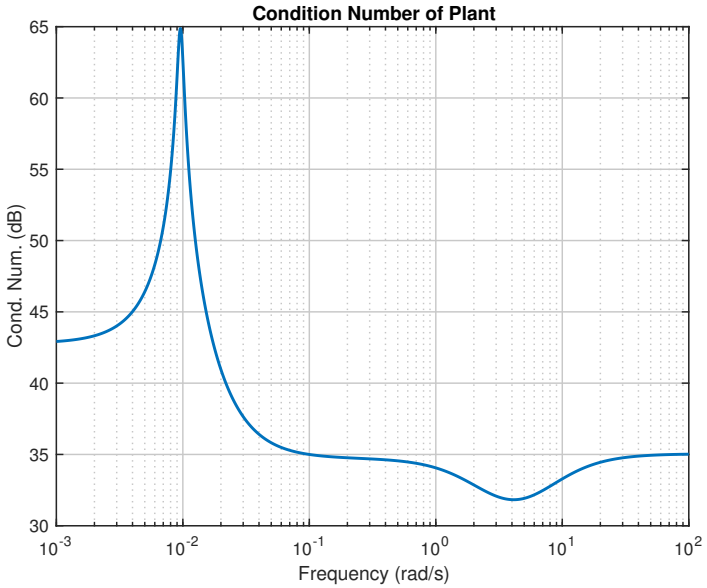


Figure 7.4: Hypersonic Vehicle Model: Condition Number

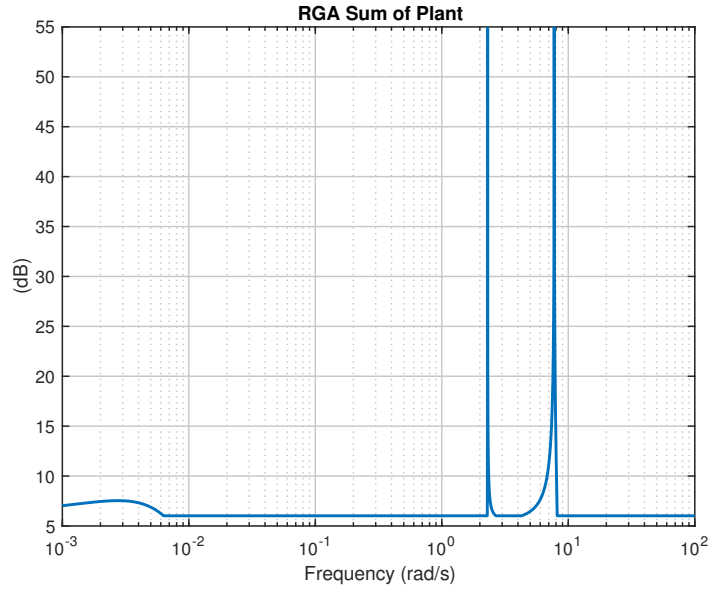


Figure 7.5: Hypersonic Vehicle Model: RGA Sum

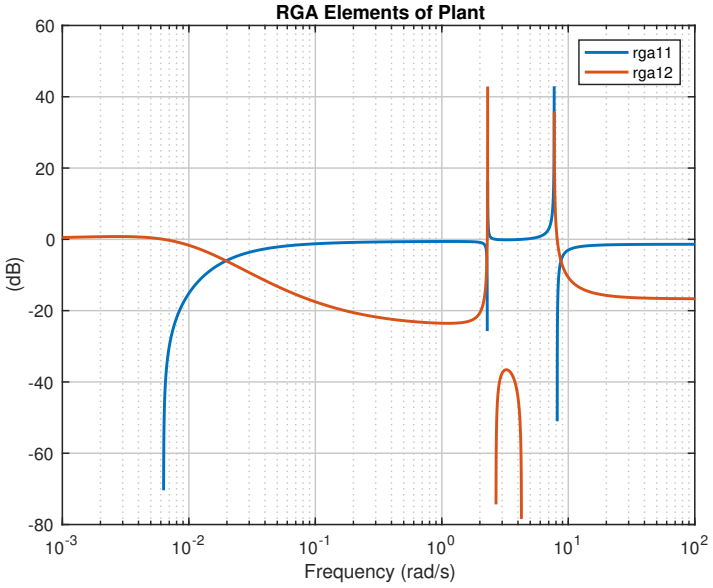


Figure 7.6: Hypersonic Vehicle Model: RGA Elements

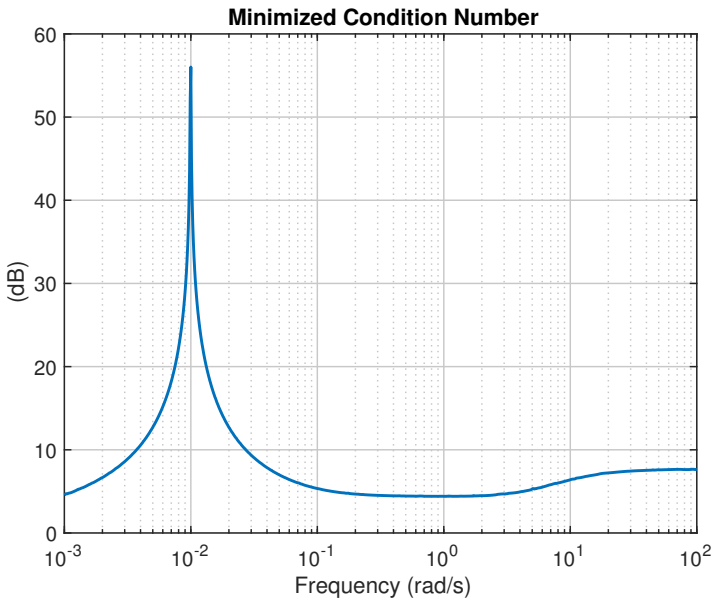


Figure 7.7: Hypersonic Vehicle Model: Minimized Condition Number

7.3 Control Designs

We consider three inner-outer loop control designs for the rigid model of the 3-DOF hypersonic vehicle. These are as D-1, D-2 and D-3. Here, the pitch attitude θ (one of the plant states) is fed back within the inner-loop. The outputs - velocity v and FPA γ - are fed back in the outer loop.

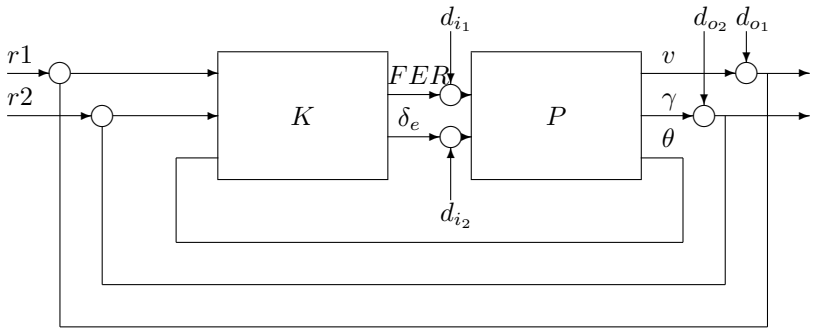


Figure 7.8: Hierarchical Inner-Outer Loop Control Structure for Hypersonics Model

Figure 7.8 shows the hierarchical inner-outer loop control structure used. The sensor noise in the inner-loop (n_i) is associated with the measurement of the angle θ which can be seen as being fed back in the inner loop. T_{n_iu} is shaped so as to attenuate high-frequency sensor noise associated with inner-loop sensor noise. For a fair comparison of designs, based on the challenging plant in consideration, we consider the following specifications: $\|S_e\|_{\mathcal{H}^\infty}, \|T_e\|_{\mathcal{H}^\infty} < 3dB$, $\|S_c\|_{\mathcal{H}^\infty}, \|T_c\|_{\mathcal{H}^\infty} < 7dB$, settling times $v_{ts} < 80s$, $\gamma_{ts} < 15s$, and peak control signals |FER| < 1.5, $|\delta_e| < 20$, with comparable bandwidths in closed loop maps. It is important to note that the closed loop maps corresponding to inner-loop sensor noise (n_i) are not captured when we break the loop at error or controls. To address this, we shape T_{n_iu} by making use of the proposed methodology. Specifically, we make use of the weighting function W_8 in design D-1. This is dicussed in more detail below.

7.3.1 Generalized Mixed Sensitivity Design (D-1)

Design D-1 uses our generalized mixed-sensitivity inner-outer loop methodology. The closed loop maps corresponding to the following loop breaking points: (1) error (e), corresponding to the difference between desired velocity v & FPA γ, and the actual values, (2) controls (c), which are fuel equivalence ratio (FER) & elevator deflection δ_e, and (3) inner-loop sensor noise (n_i) corresponding to the pitch attitude θ, along with suitable weights are used in the optimization methodology captured by Eqn. 4.5. Accomodating these three loop breaking points that are critical for hierarchical inner-outer architecture is a major improvement to the methodology presented in [55]. In the Design 3 (D-3) that is discussed below, we use the ideas presented in [55] for comparing to our current work (captured by D-1). The weighting functions used are

as follows:

$$w_{1i} = \frac{s/M_{e_i} + \omega_{e_i}}{s + \omega_{e_i}\epsilon_{e_i}}, w_{2i} = \frac{s + \omega_{u_i}/M_{u_i}}{\epsilon_{u_i}s + \omega_{u_i}} \tag{7.5}$$

$$w_{3i} = \frac{s + \omega_{y_i}/M_{y_i}}{\epsilon_{y_i}s + \omega_{y_i}}, w_{4i} = \frac{s/M_{e_i} + \omega_{e_i}}{s + \omega_{e_i}\epsilon_{e_i}} \tag{7.6}$$

$$w_{5i} = 1 w_{6i} = \frac{s + \omega_{y_i}/M_{y_i}}{\epsilon_{y_i}s + \omega_{y_i}}, w_{7i} = 0 \tag{7.7}$$

$$w_{8i} = \frac{s + \omega_{n_i}/M_{n_i}}{\epsilon_{n_i}s + \omega_{n_i}}, w_{9i} = 0 \tag{7.8}$$

$W_j = diag(w_{ji}, w_{ji})$ where $j = 1 - 9$ and $i = 1, 2$. Table 7.3 shows the weighting function parameter values (see Eqn. 4.5). The weighting function W_8 is used to shape

	W_1	W_2	W_3	W_4	W_6	W_8
M_i	1.08	0.29	1.3	1.08	1.3	0.1
ω_i	0.05	20	3	0.1	20	25
ϵ_i	0.01	0.3	0.01	0.01	0.01	0.01

Table 7.3: Weights Used for Design-1

T_{n_iu}, so as to attenuate high-frequency sensor noise associated with inner-loop sensor noise. For a fair comparison between designs, we have tried to maintain bandwidth of T_{n_iu} at 30 rad/s. This was achieved in designs D-1 and D-2. But using D-3, this was not possible as the closed loop system became unstable when roll-off was severe. Hence D-3 corresponds the bandwidth of T_{n_iu} set at 200rad/s.

7.3.2 Classically Motivated Design (D-2)

Design D-2 is a classically motivated (multivariable) control design that possesses a fully populated PD inner-loop structure (with lag) and a PI outer-loop structure.

The precise controller structures used are as follows:

$$K_i = \begin{bmatrix} (K_{p_1} + K_{d_1} s)\, \alpha_2 \left(\frac{s + \frac{\omega_{gi_2}}{\sqrt{\alpha_2}}}{s + \omega_{gi_2}\sqrt{\alpha_2}} \right) \\ (K_{p_2} + K_{d_2} s) \end{bmatrix} \tag{7.9}$$

$$K_o = \begin{bmatrix} \left(k_{p_{11}} + \frac{k_{i_{11}}}{s} \right) & \left(k_{p_{12}} + \frac{k_{i_{12}}}{s} \right) \\ \left(k_{p_{21}} + \frac{k_{i_{21}}}{s} \right) & \left(k_{p_{22}} + \frac{k_{i_{22}}}{s} \right) \end{bmatrix} \tag{7.10}$$

where the K_o parameters are $k_{p_{11}}$=2, $k_{i_{11}}$=−0.02, $k_{p_{21}}$=−5, $k_{i_{21}}$=−0.4, $k_{p_{12}}$=−0.7, $k_{i_{12}}$=−0.312, $k_{p_{22}}$=−0.2, $k_{i_{22}}$=−0.074. The K_i parameter values are K_{p_1}=−0.3, K_{d_1}=−0.465, α_2=0.5, $\omega_{g_{i_2}}$=1.1313, K_{p_2}=−0.05, K_{i_2}=−0.065. Along with the above, roll-off terms are used in both K_o (i.e., $(\frac{10}{s+10})^2$ on all four elements in the transfer function matrix and K_i (i.e., $(\frac{20}{s+20})^3$ on the first element and $(\frac{35}{s+35})^3$ on the second element) are used. The roll-off terms, in addition to making the system robust to high frequency modeling uncertainties, are also used to obtain good sensor noise attenuation. We have observed that adjusting roll-off in K_i helps adjust the bandwidth of inner-loop control sensitivity $T_{n_i u}$. The above parameter values were found manually by running loops over a large set of parameter values. Despite its decent properties (worse than D-1, discussed below), computing D-2 required considerable time - thus providing significant motivation for the method used to obtain the superior design D-1. Further, due to the high number of parameters to be optimized (manually), it is not guaranteed that these are optimal values. This is another major disadvantage of D-2.

7.3.3 Standard Mixed Sensitivity Design (with $r - d_i$ Generalized Plant) (D-3)

Design D-3 uses the generalized mixed sensitivity formulation presented in [55]. The closed loop maps corresponding to the following loop breaking points: (1) error (e), corresponding to the difference between desired velocity v & FPA γ, and the actual

values, and (2) controls (c), which are fuel equivalence ratio (FER) & elevator deflection δ_e, along with suitable weights are used in the optimization methodology. Note that, in this formulation, the properties at inner-loop sensor noise (n_i) are adjusted external to the optimization methodology. For e.g., the bandwidth of inner-loop control sensitivity ($T_{n_i u}$) is corresponding to the pitch attitude θ is set by adding roll-off term to inner-loop controller K_i (See [55]). This is necessary for comparing the designs obtained by the methodology to classical design techniques.

NOTE: Given the above, it is important to reiterate that D-3 does not incorporate properties at inner-loop sensor noise (n_i) within the objective function of the optimization methodology. We therefore do not expect to see results from D-3 as good as those from D-1 or D-2. Table 7.4 demonstrates this by summarizing the closed loop properties for each design.

Type	S_e	S_c	T_e	T_c	T_{rc}	$T_{d_i y}$
D-1	3.42	4.90	1.96	6.97	20.64	1.97
D-2	2.40	7.84	2.39	7.21	15.90	1.41
D-3	2.57	7.68	1.87	8.28	15.33	-3.80
	$v_{ts}(s)$	$\gamma_{ts}(s)$	BW S_e	BW S_c	BW T_{rc}	|FER|
D-1	78.65	14.06	0.017	0.018	22.61	1.45
D-2	66.27	11.90	0.020	0.021	20.90	1.48
D-3	80.16	3.10	0.016	0.014	23.35	0.92

Table 7.4: Critical Control-Relevant Properties: Peak Singular Values Are in dB, Bandwidth of $S_{e/c}$ & T_{rc} Measured at $-20dB$ & $0dB$ Respectively

In designs D-1 and D-3, the infinite dimensional Youla parameter (Q) is approximated by a finite dimensional paraameter (Q_N) with $N = 6$. Here, all-pass basis with

pole location at 10 are chosen. Coprime-factorization based Q-parameterization is used along with an inital controller found based on LQG method (controller weights $Q_f = I, R_f = 10^5 \times I$ and observer weights $Q_l = I, R_l = 10^5 \times I$).

7.3.4 Observations

Critical observations for the designs D-1, D-2 and D-3 are now made:

- D-1 outperforms D-2 and D-3 in terms of sensitivity at controls S_c. In terms of sensitivity at error S_e, D-1 is slightly worse, as can be seen in Table 7.4 and Figs.7.9-7.10. In the figures, "GMS" corresponds designs obtained using Generalized Mixed Sensitivity (D-1), and "Classical" corresponds to the designs obtained using classical ideas (D-2). In other words, using D-1, we are able to tradeoff S_e somewhat to obtain better S_c. This can be thought of as approaching our goal of "equilibration". This was done by selecting similar weighting function values corresponding to these closed-loop maps. It must be noted that D-1 was much much faster to obtain than D-2. The time-to-design for D-1 was comparable to D-3. D-1, and hence our method, is therefore viewed as superior because it offers speed while directly shaping the desired closed loop maps.

- It is important to emphasize that in D-1 inner-loop control sensitivity $T_{n_i u}$ is shaped within the optimization methodology via the weighting function W_8, whereas in D-2, a roll-off is used on inner-loop controller K_i to adjust the bandwidth of $T_{n_i u}$. In D-2, the parameters were tuned by brute force search, along with a roll-off added in D-2, where as in D-1, it is more systematic. In D-3, the roll-off is added post-optimization, and hence D-1 and D-2 outperform D-3. D-2 is outperformed by D-1, which is expected as D-1 is a full-order controller and D-2 is much simpler.

- A critical point to be made here is, D-1 (and its desirable sensitivity properties) was far easier to obtain (faster, more transparent) using our generalized mixed-sensitivity inner-outer loop approach than D-2.

- Fig. 7.14 is an illustration of time-domain constraint being handled by the generalized mixed-sensitivity methodology. A maximum overshoot of 4% in velocity channel was allowed, as opposed to 16% in the unconstrained case. This constraint was satisfied, but after trading off settling time. Note that settling time constraint, being quasi-convex, cannot be accomodated. Yet, overshoot constraint is considered to be a useful tool for control design. This also extends to control step response, if one wants to limit the peak value of control signal.

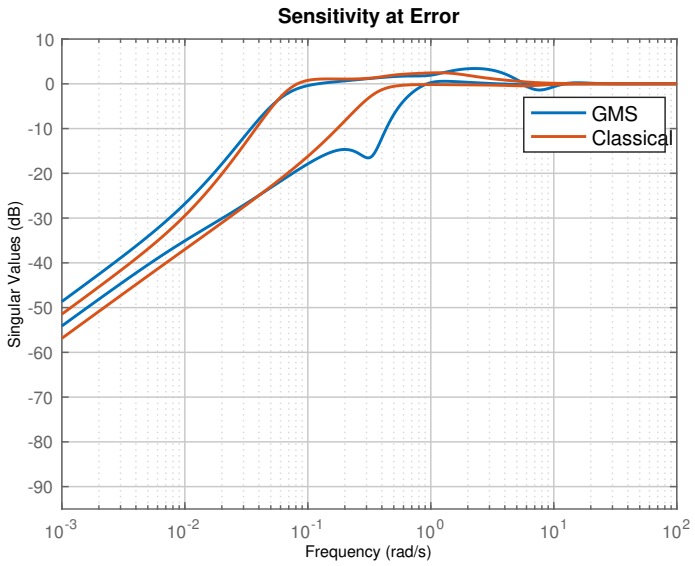


Figure 7.9: Sensitivities at Output S_e: D-1 and D-2

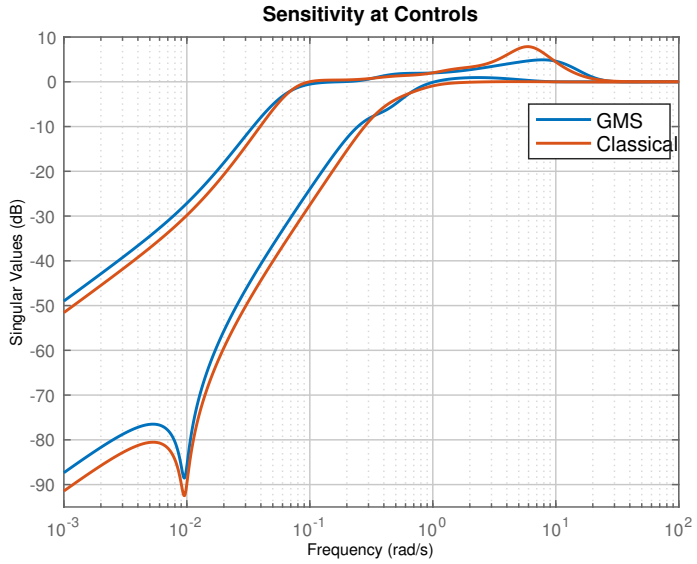


Figure 7.10: Sensitivities at Input S_c: D-1 and D-2

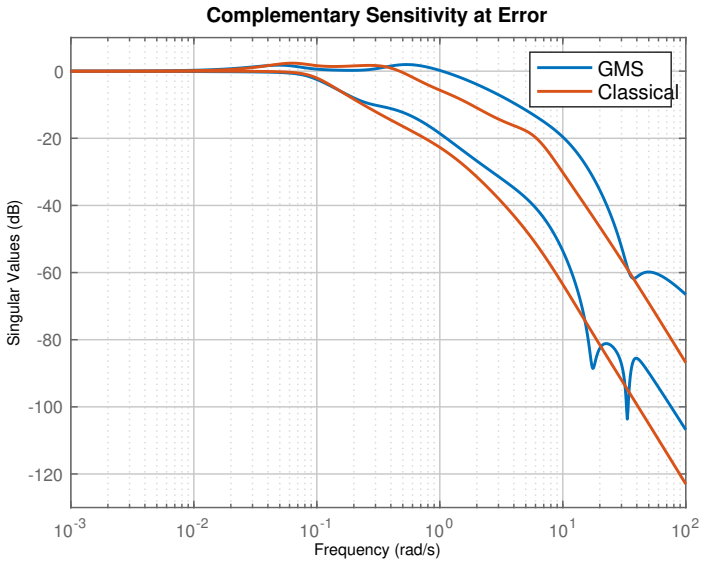


Figure 7.11: Complementary Sensitivities at Output T_e: D-1 and D-2

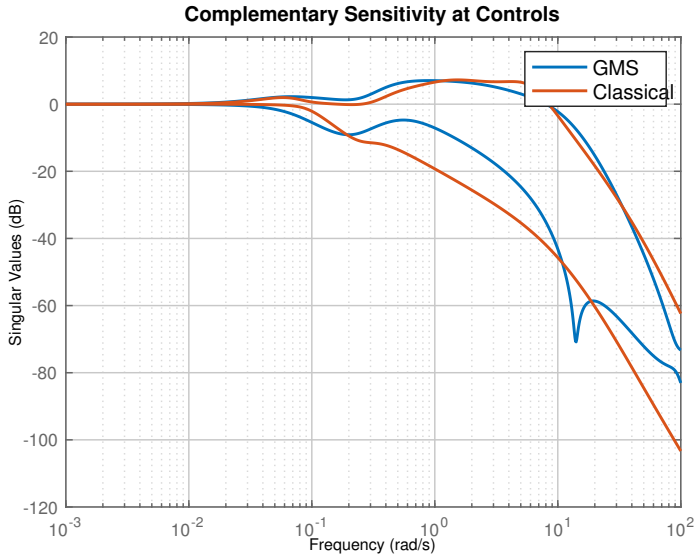


Figure 7.12: Complementary Sensitivities at Input T_c: D-1 and D-2

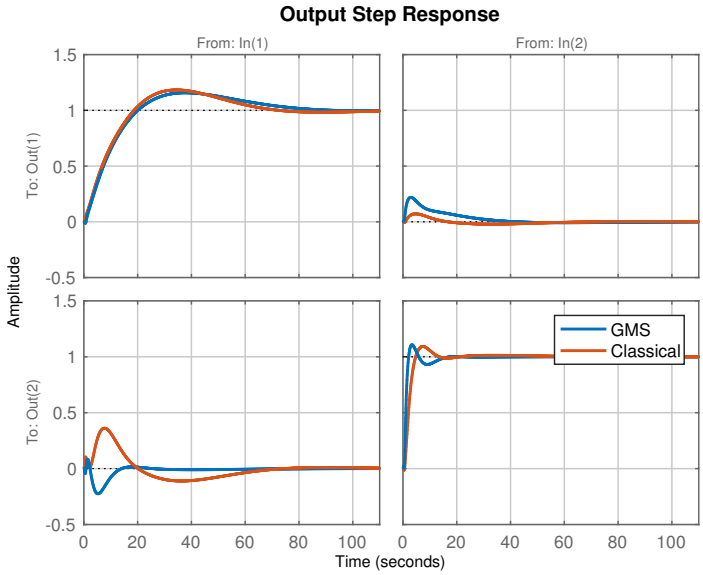


Figure 7.13: Output Respones to Step Input References: D-1 and D-2

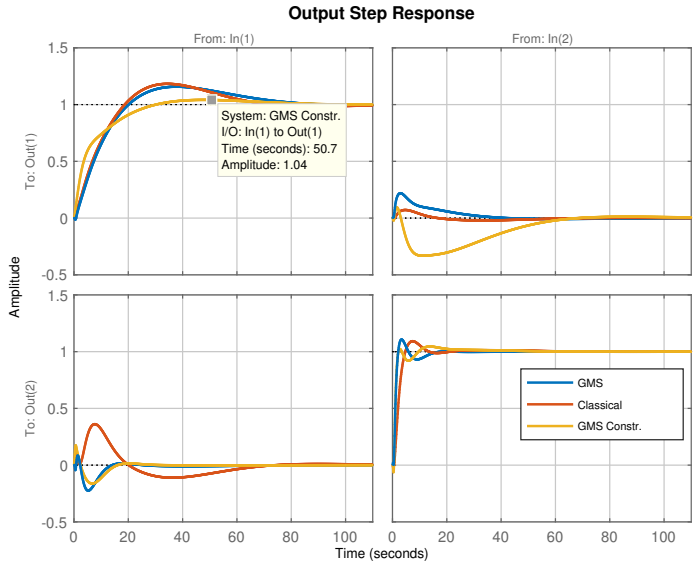


Figure 7.14: Output Respones to Step Input References: D-1, D-2 and D-1 with Overshoot Constrant

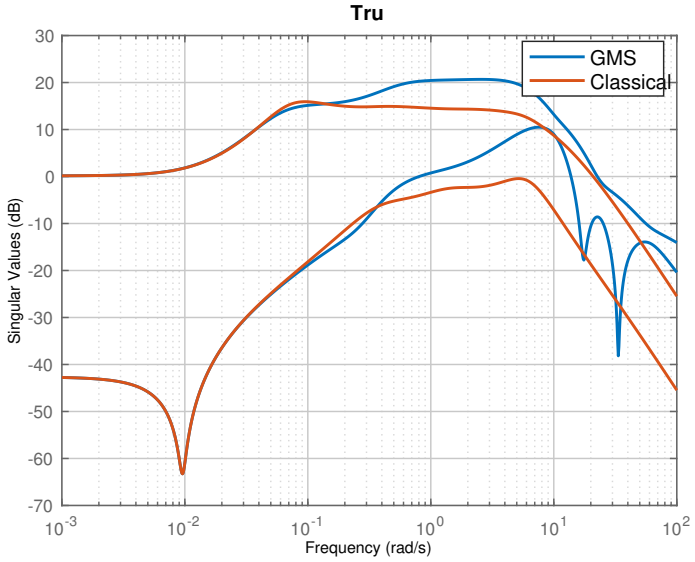


Figure 7.15: Control Sensitivities with Respect to Reference T_{ru}: D-1 and D-2

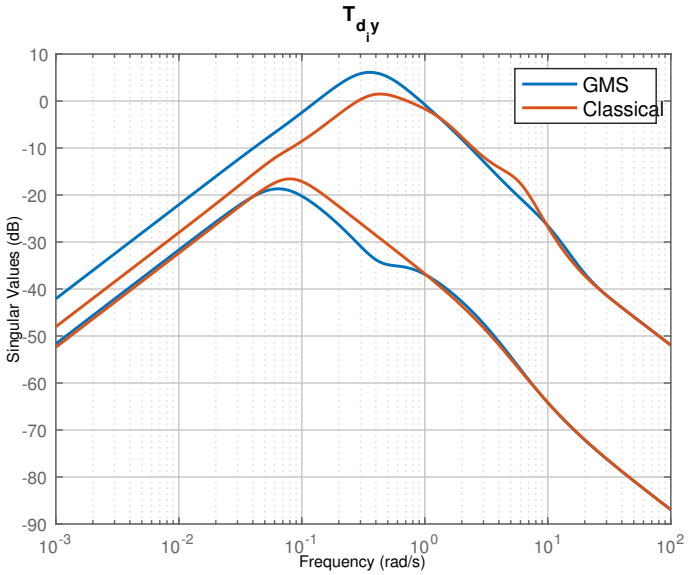


Figure 7.16: Disturbance Sensitivities at Input $T_{d_i y}$: D-1 and D-2

7.4 Summary and Conclusions

In this chapter, we considered a scramjet-powered hypersonic vehicle model in the operating condition Mach 8, 85 kft. We designed inner-outer loop controllers for the longitudinal dynamics using three approaches. The control system seeked to satisfy the specifications at several loop-breaking points. These approaches were studied and compared. It was shown how the proposed Generalized Mixed Sensitivity methodology is able to systematically shape closed loop properties at distinct loop-breaking points, thus directly addressing the control design specifications.

Chapter 8

SUMMARY AND DIRECTIONS FOR FUTURE RESEARCH

8.1 Summary

In this dissertation, we presented a multivariable control design methodology, based on multiobjective $\mathcal{H}^\infty$ mixed sensitivity to effectively and efficiently tradeoff closed loop properties at distinct loop breaking points. We studied the tradeoffs involved in obtaining acceptable properties at these distinct loop breaking points. We analyzed the closed loop shaping challenges and limitations at these distinct loop-breaking points and illustrated through examples the existence of some pareto optimal points associated with them. The methodology accommodates a broad class of frequency- and time-domain control design specifications. The above approach was used to design hierarchical inner-outer loop controllers for trading off control properties for longitudinal model of 3-DOF Hypersonic vehicle model - one that is unstable and non-minimum phase. The methodology was shown to directly address shaping of closed loop properties at these points using convex optimization. This was accomplished by exploiting the Youla-Jabr-Bongiorno-Kucera (YJBK) parameterization. Basis parameters that result in efficient approximation (using lesser number of basis terms) of the infinite-dimensional parameter were studied. Three state-of-the-art subgradient-based non-differentiable constrained convex optimization solvers, namely Analytic Center Cutting Plane Method (ACCPM), Kelley's CPM and SolvOpt were implemented and compared within the GMS framework.

8.2 Directions for Future Research

In this dissertation, we addressed trading off closed loop properties at distinct loop breaking points by using $\mathcal{H}^\infty$ Generalized Mixed Sensitivity methodology. Below, we list some extensions to the work presented in the dissertation.

- It was shown how $\mathcal{H}^\infty$ based objectives and $\mathcal{H}^\infty/\mathcal{L}^\infty$ based constraints can be handled by GMS methodology in Chapter 5. Additional convex objectives/constraints based on different norms such as $\mathcal{H}^2$ [59] on closed loop maps can be incorporated to address wider class of design specifications.

- Additional constraints such as sector constraint on closed loop pole locations can be incorporated [241–243].

- Model reduction techniques can be used to reduce the order of resulting controller obtained from the methodology [133].

- Additional convex optimization algorithms that can handle constrained non-differentiable optimization problems can be investigated and compared to those in Section 5.6. The following solvers are compatible with MATLAB.

  - Optimal SubGradient Algorithm (OSGA) [244–246]
  - Gradient sampling solver (GradSamp) [247]

  A list of similar solvers (both MATLAB based and otherwise) can be found in [214, 248–251]

- "Optimal" basis selection strategies to approximate the infinite dimensional Youla parameter Q using finite dimensional Q_n [173–177, 252–254] can be developed. The types of bases selected can include using "mixed" types where different combinations of basis options shown in Section 5.4 are used.

- Finally, future work can address applying our ideas to stable and unstable multivariable infinite-dimensional systems. Work in this direction was started in [56, 57] where simple stable infinite dimensional systems were considered.

REFERENCES

[1] J. S. Freudenberg, "Analysis and design for ill-conditioned plants part i. lower bounds on the structured singular value," *International Journal of Control*, vol. 49, no. 3, pp. 851–871, 1989. [Online]. Available: http://dx.doi.org/10.1080/00207178908559672

[2] ——, "Analysis and design for ill-conditioned plants part 2. directionally uniform weightings and an example," *International Journal of Control*, vol. 49, no. 3, pp. 873–903, 1989. [Online]. Available: http://dx.doi.org/10.1080/00207178908559673

[3] J. S. Freudenberg and D. P. Loose, "Relations between properties of multivariable feedback systems at different loop-breaking points: Part i," in *Proceedings of the 24th IEEE Conference on Decision and Control, 1985*, Dec 1985, pp. 250–256.

[4] J. S. Freudenberg and D. P. Looze, "Relations between properties of multivariable feedback systems at different loop-breaking points: Part ii," in *IEEE American Control Conference, 1986*, June 1986, pp. 771–777.

[5] J. S. Freudenberg, "Analysis and design for ill-conditioned plants," in *IEEE American Control Conference, 1988*, June 1988, pp. 372–377.

[6] ——, "Directionality, coupling, and multivariable loop-shaping," in *Proceedings of the 27th IEEE Conference on Decision and Control, 1988*, vol. 1, Dec 1988, pp. 399–340.

[7] J. S. Freudenberg and D. P. Looze, *Frequency Domain Properties of Scalar and Multivariable Feedback Systems*, 1st ed., ser. Lecture Notes in Control and Information Sciences. Springer-Verlag Berlin Heidelberg, 1988, vol. 104.

[8] J. S. Freudenberg, C. V. Hollot, and R. H. Middleton, "A tradeoff between disturbance attenuation and stability robustness," in *IEEE American Control Conference, 2003*, vol. 6, June 2003, pp. 4816–4821.

[9] G. Stein and J. C. Doyle, "Beyond singular values and loop shapes," *Journal of Guidance, Control, and Dynamics*, vol. 14, no. 1, pp. 5–16, 1991.

[10] J. S. Freudenberg, "Plant directionality, coupling and multivariable loop-shaping," *International Journal of Control*, vol. 51, no. 2, pp. 365–390, 1990. [Online]. Available: https://doi.org/10.1080/00207179008934071

[11] J. S. Freudenberg and K. Saglik, "Scaling down the plant condition number scales up the size of uncertainty," in *29th IEEE Conference on Decision and Control*, Dec 1990, pp. 1195–1196 vol.3.

[12] J. Freudenberg and R. Middleton, "Design rules for multivariable feedback systems," in *Proceedings of 35th IEEE Conference on Decision and Control*, vol. 2, Dec 1996, pp. 1980–1985 vol.2.

[13] J. S. Freudenberg, C. V. Hollot, R. H. Middleton, and V. Toochinda, "Fundamental design limitations of the general control configuration," *IEEE Transactions on Automatic Control*, vol. 48, no. 8, pp. 1355–1370, Aug 2003.

[14] S. Hara and M. Kanno, *When Is a Linear Continuous-time System Easy or Hard to Control in Practice?* London: Springer London, 2008, pp. 111–124. [Online]. Available: http://dx.doi.org/10.1007/978-1-84800-155-8_8

[15] J. C. Doyle, J. E. Wall, and G. Stein, "Performance and robustness analysis for structured uncertainty," in *21st IEEE Conference on Decision and Control*, Dec 1982, pp. 629–636.

[16] J. Chen, J. S. Freudenberg, and C. N. Nett, "The role of the condition number and the relative gain array in robustness analysis," *Automatica*, vol. 30, no. 6, pp. 1029–1035, 1994.

[17] S. Skogestad and K. Havre, "The use of RGA and condition number as robustness measures," *Computers & Chemical Engineering*, vol. 20, pp. S1005 – S1010, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0098135496001755

[18] S. Skogestad, M. Morari, and J. C. Doyle, "Robust control of ill-conditioned plants: High-purity distillation," *IEEE Transactions on Automatic Control*, vol. 33, no. 12, pp. 1092–1105, Dec 1988.

[19] K. Havre and S. Skogestad, "Achievable performance of multivariable systems with unstable zeros and poles," *International Journal of Control*, vol. 74, no. 11, pp. 1131–1139, 2001. [Online]. Available: https://doi.org/10.1080/00207170110053346

[20] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*, 2nd ed. New York: Wiley, 2007.

[21] A. Khaki-Sedigh and B. Moaveni, *Control Configuration Selection for Multivariable Plants*, ser. Lecture Notes in Control and Information Sciences. Springer-Verlag Berlin Heidelberg, 2009, vol. 391.

[22] J. Doyle, "Analysis of feedback systems with structured uncertainties," *IEE Proceedings D - Control Theory and Applications*, vol. 129, no. 6, pp. 242–250, November 1982.

[23] J.-L. Goffin and J.-P. Vial, "Convex nondifferentiable optimization: A survey focused on the analytic center cutting plane method," *Optimization Methods and Software*, vol. 17, no. 5, pp. 805–867, 2002.

[24] F. Babonneau, C. Beltran, A. Haurie, C. Tadonki, and J.-P. Vial, *Proximal-ACCPM: A Versatile Oracle Based Optimisation Method*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 67–89.

[25] J.-L. Goffin and J.-P. Vial, "Multiple cuts in the analytic center cutting plane method," *SIAM Journal on Optimization*, vol. 11, no. 1, pp. 266–288, 2000. [Online]. Available: https://doi.org/10.1137/S1052623498340266

[26] J. Gondzio, O. du Merle, R. Sarkissian, and J.-P. Vial, "ACCPM - A library for convex optimization based on an analytic center cutting plane method," *European Journal of Operational Research*, vol. 94, no. 1, pp. 206 – 211, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0377221796001695

[27] Y. Nesterov and J. P. Vial, "Homogeneous analytic center cutting plane methods for convex problems and variational inequalities," *SIAM Journal on Optimization*, vol. 9, no. 3, pp. 707–728, 1999. [Online]. Available: https://doi.org/10.1137/S1052623497324813

[28] J. E. Mitchell, *Cutting Plane Methods and Subgradient Methods*. Maryland, USA: Institute for Operations Research and the Management Sciences (INFORMS), 2009, ch. Chapter 2, pp. 34–61. [Online]. Available: https://pubsonline.informs.org/doi/abs/10.1287/educ.1090.0064

[29] S. Elhedhli, J.-L. Goffin, and J.-P. Vial, "Nondifferentiable optimization: Introduction, applications and algorithms," Kluwer Academic Publishers, Dordrecht, pp. 1705–1710, 2000.

[30] O. Péton and J.-P. Vial, "A brief tutorial on ACCPM," University of Geneva, Tech. Rep., 2001.

[31] J.-L. Goffin and J.-P. Vial, "Convex nondifferentiable optimization: A survey focused on the analytic center cutting plane method," *Optimization Methods and Software*, vol. 17, no. 5, pp. 805–867, 2002. [Online]. Available: https://doi.org/10.1080/1055678021000060829a

[32] S. P. Boyd, L. Vandenberghe, and J. Skaf, "Analytic center cutting-plane method," 2018.

[33] G. C. Goodwin, S. F. Graebe, and M. E. Salgado, *Control System Design*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

[34] F. G. Shinskey, *Process Control Systems: Application, Design and Tuning*, 4th ed. New York, NY, USA: McGraw-Hill, Inc., 1996.

[35] K. Zhou, K. Glover, B. Bodenheimer, and J. Doyle, "Mixed $\mathcal{H}^2$ and $\mathcal{H}^\infty$ performance objectives. I: Robust performance analysis," *IEEE Transactions on Automatic Control*, vol. 39, no. 8, pp. 1564–1574, Aug 1994.

[36] J. Doyle, K. Zhou, and B. Bodenheimer, "Optimal control with mixed $\mathcal{H}^2$ and $\mathcal{H}^\infty$ performance objectives," in *IEEE American Control Conference, 1989*, June 1989, pp. 2065–2070.

[37] P. P. Khargonekar and M. A. Rotea, "Mixed $\mathcal{H}^2/\mathcal{H}^\infty$ control: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 36, no. 7, pp. 824–837, Jul 1991.

[38] C. W. Scherer, "Multiobjective $\mathcal{H}^2/\mathcal{H}^\infty$ control," *IEEE Transactions on Automatic Control*, vol. 40, no. 6, pp. 1054–1062, Jun 1995.

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

[40] H. A. Hindi, B. Hassibi, and S. P. Boyd, "Multiobjective $\mathcal{H}^2/\mathcal{H}^\infty$-optimal control via finite dimensional Q-parametrization and linear matrix inequalities," in *IEEE American Control Conference, 1998*, vol. 5, Jun 1998, pp. 3244–3249.

[41] X. Qi, M. H. Khammash, and M. V. Salapaka, "Optimal controller synthesis with multiple objectives," in *IEEE American Control Conference, 2001*, vol. 4, 2001, pp. 2730–2735.

[42] P. Apkarian, D. Noll, and A. Rondepierre, "Mixed $\mathcal{H}^2/\mathcal{H}^\infty$ control via nonsmooth optimization," *SIAM Journal on Control and Optimization*, vol. 47, no. 3, pp. 1516–1546, 2008. [Online]. Available: http://dx.doi.org/10.1137/070685026

[43] T. Iwasaki and S. Hara, "Feedback control synthesis of multiple frequency domain specifications via generalized kyp lemma," *International Journal of Robust and Nonlinear Control*, vol. 17, no. 5-6, pp. 415–434, 2007. [Online]. Available: http://dx.doi.org/10.1002/rnc.1123

[44] ——, "Dynamic output feedback synthesis with general frequency domain specifications," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 345 – 350, 2005, 16th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016370148

[45] P. Gahinet and P. Apkarian, "A linear matrix inequality approach to $\mathcal{H}^\infty$ control," *International Journal of Robust and Nonlinear Control*, vol. 4, no. 4, pp. 421–448, 1994. [Online]. Available: http://dx.doi.org/10.1002/rnc.4590040403

[46] Y. S. Hung and B. Pokrud, "An $\mathcal{H}^\infty$ approach to feedback design with two objective functions," *IEEE Transactions on Automatic Control*, vol. 37, no. 6, pp. 820–824, Jun 1992.

[47] Z. Hu, S. E. Salcudean, and P. D. Loewen, "Numerical solution of the multiple objective control system design problem for siso systems," in *American Control Conference, Proceedings of the 1995*, vol. 2, Jun 1995, pp. 1458–1462.

[48] L. Hosseini-Ravanbod, D. Noll, and P. Apkarian, "Robustness via structured $\mathcal{H}^\infty/\mathcal{H}^\infty$ synthesis," *International Journal of Control*, vol. 84, no. 5, pp. 851–866, 2011.

[49] M. Gabarrou, D. Alazard, and D. Noll, "Structured flight control law design using non-smooth optimization," *IFAC Proceedings Volumes*, vol. 43, no. 15, pp. 536 – 541, 2010, 18th IFAC Symposium on Automatic Control in Aerospace. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667015318966

[50] J. M. Rieber and F. Allgöwer, "From $\mathcal{H}^\infty$ control to multiobjective control: An overview," *at Automatisierungstechnik*, vol. 54, no. 9, pp. 437–449, 2006.

[51] M. V. Salapaka and M. Dahleh, *Multiple objective control synthesis*, ser. Lecture Notes in Control and Information Sciences. Springer, London: Springer-Verlag, 2000.

[52] W. M. Haddad, V. Chellaboina, and R. Kumar, "Multiobjective $\mathcal{L}_1/\mathcal{H}^\infty$ controller design for systems with frequency and time domain constraints," *European Journal of Control*, vol. 6, no. 2, pp. 170 – 183, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0947358000709253

[53] J. R. Corrado and W. M. Haddad, "Robust fixed-structure controller synthesis," Ph.D. Dissertation, Georgia Institute of Technology, 2000.

[54] T. Basar and P. Bernhard, *$\mathcal{H}^\infty$ Optimal Control and Related Minimax Design Problems: A Dynamic Game Approach*, 2nd ed. Boston, MA: Birkhauser, 2008, vol. 235.

[55] K. Puttannaiah, A. A. Rodriguez, K. Mondal, J. A. Echols, and D. G. Cartagena, "A generalized mixed-sensitivity convex approach to hierarchical multivariable inner-outer loop control design subject to simultaneous input and output loop breaking specifications," in *IEEE American Control Conference, 2016*, July 2016, pp. 5632–5637.

[56] K. Puttannaiah, J. A. Echols, and A. A. Rodriguez, "A generalized $\mathcal{H}^\infty$ control design framework for stable multivariable plants subject to simultaneous output and input loop breaking specifications," in *IEEE American Control Conference, 2015*, July 2015, pp. 3310–3315.

[57] K. Puttannaiah, J. A. Echols, K. Mondal, and A. A. Rodriguez, "Analysis and use of several generalized $\mathcal{H}^\infty$ mixed sensitivity framework for stable multivariable plants subject to simultaneous output and input loop breaking specifications," in *Proceedings of the 54th IEEE Conference on Decision and Control, 2015*, Dec 2015, pp. 6617–6622.

[58] K. Puttannaiah, "$\mathcal{H}^\infty$ control design via convex optimization: Toward a comprehensive design environment," M.S. Thesis, Arizona State University, Tempe, AZ, 2013.

[59] S. P. Boyd and C. H. Barratt, *Linear Controller Design: Limits of Performance*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991.

[60] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[61] S. P. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*. SIAM, 1994, vol. 15.

[62] J. A. Echols, K. Puttannaiah, K. Mondal, and A. A. Rodriguez, "Fundamental control system design issues for scramjet-powered hypersonic vehicles," in *AIAA Guidance, Navigation & Control Conference*. American Institute of Aeronautics and Astronautics, 2015. [Online]. Available: http://dx.doi.org/10.2514/6.2015-1760

[63] K. Mondal, "Multivariable control of fixed wing aircrafts," M.S. Thesis, Arizona State University, Tempe, AZ, 2015.

[64] C. Peebles, *Road to Mach 10: Lessons Learned from the X-43A Flight Research Program*. American Institute of Aeronautics and Astronautics, 2008.

[65] C. McClinton, "X-43-scramjet power breaks the hypersonic barrier: Dryden lectureship in research for 2006," in *44th AIAA Aerospace Sciences Meeting and Exhibit, Aerospace Sciences Meetings*, 2006.

[66] M. C. Davis and J. T. White, "X-43a flight-test-determined aerodynamic force and moment characteristics at mach 7.0," *Journal of spacecraft and rockets*, vol. 45, no. 3, pp. 472–484, 2008.

[67] S. Sridharan, J. A. Echols, A. A. Rodriguez, and K. Mondal, "Integrated design and control of hypersonic vehicles," in *IEEE American Control Conference, 2014*, 2014, pp. 1371–1376.

[68] S. Sridharan, "Multidisciplinary optimization for the design and control of uncertain dynamical systems," Ph.D. Dissertation, Arizona State University, Tempe, AZ, 2014.

[69] S. Sridharan and A. A. Rodriguez, "Impact of control specifications on vehicle design for scramjet-powered hypersonic vehicles," in *AIAA Guidance, Navigation and Control Conference*, 2013, pp. 1–18.

[70] S. Sridharan and A. Rodriguez, "Performance based control-relevant design for scramjet-powered hypersonic vehicles," in *AIAA Guidance, Navigation and Control Conference*, 2012, pp. 1–17. [Online]. Available: http://arc.aiaa.org/doi/pdf/10.2514/6.2012-4469

[71] J. J. Dickeson, "Control relevant modeling and design of scramjet-powered hypersonic vehicles," Ph.D. dissertation, Arizona State University, 2012.

[72] S. Sridharan, A. A. Rodriguez, J. J. Dickeson, and D. Soloway, "Constraint enforcement and robust tube-based control for scramjet-powered hypersonic vehicles with significant uncertainties," in *IEEE American Control Conference*, 2012, pp. 4619–4624. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6315659

[73] S. Sridharan, J. J. Dickeson, and A. A. Rodriguez, "Impact of plume modeling on the design and control for a class of air-breathing hypersonic vehicles," in *AIAA Guidance, Navigation and Control Conference*, 2011, pp. 509–531.

[74] J. Khatri, "Modeling, analysis, and control of a hypersonic vehicle with significant aero-thermo-elastic-propulsion interactions: Elastic, thermal and mass uncertainty," M.S. Thesis, Arizona State University, Tempe, AZ, 2011.

[75] J. J. Dickeson, A. A. Rodriguez, S. Sridharan, and A. Korad, "Elevator sizing, placement, and control relevant tradeoffs for hypersonic vehicles," in *AIAA Guidance, Navigation and Control Conference*, 2010, pp. pp. 1–23.

[76] A. S. Korad, "Modeling, analysis, and control of a hypersonic vehicle with significant aero-thermo-elastic-propulsion interactions," M.S. Thesis, Arizona State University, Tempe, AZ, 2010.

[77] S. Sridharan, "Control relevant design of scramjet-powered hypersonic vehicles with aero-thermo-elastic-propulsive effects and uncertainty," M.S. Thesis, Arizona State University, Tempe, AZ, 2010.

[78] J. Dickeson, A. Rodriguez, S. Sridharan, J. Benavides, and D. Soloway, "Decentralized control of an airbreathing scramjet-powered hypersonic vehicle," in *AIAA Guidance, Navigation, and Control Conference*, 2009.

[79] A. A. Rodriguez, J. J. Dickeson, S. Sridharan, A. Korad, J. Khatri, J. Benavides, D. Soloway, A. Kelkar, and J. M. Vogel, "Control-relevant modeling, analysis, and design for scramjet-powered hypersonic vehicles," *AIAA/DLR/DGLR International Space Planes and Hypersonic Systems and Technologies Conference*, pp. 1–45, 2009.

[80] D. Soloway, A. A. Rodriguez, J. J. Dickeson, O. Cifdaloz, J. Benavides, S. Sridharan, A. Kelkar, and J. M. Vogel, "Constraint enforcement for scramjet-powered hypersonic vehicles with significant aero-elastic-propulsion interactions," in *2009 American Control Conference*, June 2009, pp. 3154–3159.

[81] J. M. Vogel, A. G. Kelkar, G. Inger, C. Whitmer, A. Sidlinger, and A. Rodriguez, "Control-relevant modeling of hypersonic vehicles," in *2009 American Control Conference*, June 2009, pp. 2519–2524.

[82] A. A. Rodriguez, J. J. Dickeson, O. Cifdaloz, R. McCullen, J. Benavides, S. Sridharan, A. Kelkar, J. M. Vogel, and D. Soloway, "Modeling and control of scramjet-powered hypersonic vehicles: Challenges, trends, & tradeoffs," *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, no. August, pp. 1–40, 2008.

[83] M. Bolender and D. Doman, *A Non-Linear Model for the Longitudinal Dynamics of a Hypersonic Air-breathing Vehicle.* American Institute of Aeronautics and Astronautics, 2018/04/11 2005. [Online]. Available: https://doi.org/10.2514/6.2005-6255

[84] M. A. Bolender and D. B. Doman, "Nonlinear longitudinal dynamical model of an air-breathing hypersonic vehicle," *Journal of Spacecraft and Rockets*, vol. 44, no. 2, pp. 374–387, 2018/04/11 2007. [Online]. Available: https://doi.org/10.2514/1.23370

[85] M. W. Oppenheimer and D. B. Doman, "Control of an unstable, nonminimum phase hypersonic vehicle model," in *2006 IEEE Aerospace Conference*, 2006, pp. 7 pp.–.

[86] M. Liserre, F. Blaabjerg, and S. Hansen, "Design and control of an LCL-filter-based three-phase active rectifier," *IEEE Transactions on Industry Applications*, vol. 41, no. 5, pp. 1281–1291, Sept 2005.

[87] R. W. Erickson and D. Maksimovic, *Fundamentals of Power Electronics*, 2nd ed. New York, NY, USA: Springer Science & Business Media, 2001.

[88] A. Sarkar, "Modeling and control of a three phase voltage source inverter with an LCL filter," M.S. Thesis, Arizona State University, 2015.

[89] A. Sarkar, K. Puttannaiah, and A. A. Rodriguez, "Inner-outer loop based robust active damping for lcl resonance in grid-connected inverters using grid current feedback," in *IEEE American Control Conference*, 2018, to be published.

[90] D. Pan, X. Ruan, C. Bao, W. Li, and X. Wang, "Optimized controller design for LCL-type grid-connected inverter to achieve high robustness against grid-impedance variation," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 3, pp. 1537–1547, March 2015.

[91] F. Blaabjerg, R. Teodorescu, M. Liserre, and A. V. Timbus, "Overview of control and grid synchronization for distributed power generation systems," *IEEE Transactions on Industrial Electronics*, vol. 53, no. 5, pp. 1398–1409, Oct 2006.

[92] M. Hanif, V. Khadkikar, W. Xiao, and J. L. Kirtley, "Two degrees of freedom active damping technique for LCL filter-based grid connected pv systems," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 6, pp. 2795–2803, June 2014.

[93] "Modeling, Simulation, Animation, and Real-Time Control (MoSART) of Flexible Autonomous Machines operating in an uncertain Environment (FAME)," http://aar.faculty.asu.edu/research/mosart/mosart-fame.html, accessed: 2018-07-20.

[94] A. A. Rodriguez, K. Puttannaiah, Z. Lin, J. Aldaco, Z. Li, X. Lu, K. Mondal, S. D. Sonawani, N. Ravishankar, N. Das, and P. A. Pradhan, "Modeling, design and control of low-cost differential-drive robotic ground vehicles: Part i - single vehicle study," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, Aug 2017, pp. 155–160.

[95] ——, "Modeling, design and control of low-cost differential-drive robotic ground vehicles: Part ii - multiple vehicle study," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, Aug 2017, pp. 161–166.

[96] I. Anvari, "Non-holonomic differential drive mobile robot control and design: Critical dynamics and coupling constraints," M.S. Thesis, Arizona State University, 2013.

[97] D. Chopra, "Feedback control and obstacle avoidance for non-holonomic differential drive robots," M.S. Thesis, Arizona State University, 2013.

[98] Z. Lin, "Modeling, design and control of multiple low-cost robotic ground vehicles," M.S. Thesis, Arizona State University, 2015.

[99] J. A. Lopez, "Image processing based control of mobile robotics," M.S. Thesis, Arizona State University, 2016.

[100] Z. Li, "Modeling and control of a longitudinal platoon of ground robotic vehicles," M.S. Thesis, Arizona State University, 2016.

[101] X. Lu, "Modeling and control for vision based rear wheel drive robot and solving indoor SLAM problem using LIDAR," M.S. Thesis, Arizona State University, 2016.

[102] N. Ravishankar, "Autonomous quadrotor navigation by detecting vanishing points in indoor environments," M.S. Thesis, Arizona State University, 2018.

[103] V. Renganathan, "Kill zone analysis for a bank-to-turn missile-target engagement," M.S. Thesis, Arizona State University, 2016.

[104] A. A. Rodriguez, P. A. Pradhan, K. Puttannaiah, N. Das, K. Mondal, and A. Sarkar, "A comprehensive academic success and professional development (ASAP) framework that uses career-steering/shaping projects to train engineering students and develop critical life/professional skills: Part i - impact on key groups," in *2018 IEEE Frontiers in Education Conference Proceedings (FIE)*, 2018, to be published.

[105] A. A. Rodriguez, P. Pradhan, K. Puttannaiah, N. Das, K. Mondal, A. Sarkar, S. Sonawani, S. Lu, K. Bui, C. Cederstrom, C. Christie, Z. Giacometti, C. Kurowski, N. Lopez, B. Pedroza, T. Rosenthal, M. Sabet, B. Soni, and T. Waggoner, "A comprehensive ASAP framework that uses career-steering/shaping projects to train engineering students and develop critical life/professional skills: Part ii - case studies from students working on funded projects," in *2018 IEEE Frontiers in Education Conference Proceedings (FIE)*, 2018, to be published.

[106] A. A. Rodriguez, P. A. Pradhan, K. Puttannaiah, N. Das, K. Mondal, and A. Sarkar, "Topical concerns and critical questions engineering students want/need answers to: Dependence on key groups," in *2018 IEEE Frontiers in Education Conference Proceedings (FIE)*, 2018, to be published.

[107] A. S. Thyagaturu, A. Mercian, M. P. McGarry, M. Reisslein, and W. Kellerer, "Software defined optical networks (SDONs): A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2738–2786, 2016.

[108] A. S. Thyagaturu, Y. Dashti, and M. Reisslein, "SDN-based smart gateways (Sm-GWs) for multi-operator small cell network management," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 740–753, 2016.

[109] Z. Alharbi, A. Thyagaturu, M. Reisslein, H. ElBakoury, and R. Zheng, "Performance comparison of R-PHY and R-MACPHY modular cable access network architectures," *IEEE Transactions on Broadcasting*, 2017, to be published.

[110] A. S. Thyagaturu, Z. Alharbi, and M. Reisslein, "R-FFT: Function split at IFFT/FFT in unified lte cran and cable access network," *IEEE Transactions on Broadcasting*, pp. 1–18, 2018, to be published.

[111] P. Shantharama, A. Thyagaturu, N. Karakoc, L. Ferrari, M. Reisslein, and A. Scaglione, "LayBack: SDN management of multi-access edge computing (MEC) for network access services and radio resource sharing," *IEEE Access*, 2018, to be published.

[112] M. Reisslein and A. Thyagaturu, "Systems and methods for a smart gateway sdn-based backhaul architecture for small cells," Oct. 5 2017, uS Patent App. 15/476,611.

[113] C. Ellington, "The novel aerodynamics of insect flight: applications to micro-air vehicles," *Journal of Experimental Biology*, vol. 202, no. 23, pp. 3439–3448, 1999. [Online]. Available: http://jeb.biologists.org/content/202/23/3439

[114] C. T. Orlowski and A. R. Girard, "Dynamics, stability, and control analyses of flapping wing micro-air vehicles," *Progress in Aerospace Sciences*, vol. 51, pp. 18 – 30, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0376042112000103

[115] S. Biswal, "Modeling and control of flapping wing micro aerial vehicles," M.S. Thesis, Arizona State University, 2015.

[116] P. A. Pradhan, K. Puttannaiah, K. Mondal, A. A. Rodriguez, and S. Biswal, "Modeling and control of a flappingwing hawkmoth micro air vehicle using generalized mixed sensitivity hierarchical design approach," in *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, 2018, submitted for publication.

[117] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Archit. Code Optim.*, vol. 1, no. 1, pp. 94–125, Mar. 2004. [Online]. Available: http://doi.acm.org/10.1145/980152.980157

[118] "Intel® Xeon® Processor E7- 2800/4800/8800 product family datasheet," Intel Corporation, Tech. Rep., 2014.

[119] E. Intel, "SpeedStep® technology for the Intel® Pentium® M processor," Intel Corporation, Tech. Rep., 2004.

[120] N. N. Nandola and S. Bhartiya, "A multiple model approach for predictive control of nonlinear hybrid systems," *Journal of Process Control*, vol. 18, no. 2, pp. 131 – 148, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0959152407001175

[121] ——, "A computationally efficient scheme for model predictive control of nonlinear hybrid systems using generalized outer approximation," *Industrial & Engineering Chemistry Research*, vol. 48, no. 12, pp. 5767–5778, 2009. [Online]. Available: https://doi.org/10.1021/ie801384y

[122] ——, "Hybrid system identification using a structural approach and its model based control: An experimental validation," *Nonlinear Analysis: Hybrid Systems*, vol. 3, no. 2, pp. 87 – 100, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1751570X08000708

[123] M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control: model and optimal control theory," *IEEE Transactions on Automatic Control*, vol. 43, no. 1, pp. 31–45, Jan 1998.

[124] R. Pörn, I. Harjunkoski, and T. Westerlund, "Convexification of different classes of non-convex MINLP problems," *Computers and Chemical Engineering*, vol. 23, no. 3, pp. 439 – 448, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0098135498003056

[125] D. G. Cartagena, K. Puttannaiah, and A. A. Rodriguez, "Modeling of a multi-core processor thermal dynamics for development of dynamic thermal management controllers," in *IEEE American Control Conference, 2016*, July 2016, pp. 6917–6922.

[126] N. N. Nandola and K. Puttannaiah, "Modeling and predictive control of nonlinear hybrid systems using disaggregation of variables - a convex formulation," in *IEEE European Control Conference, 2013*, July 2013, pp. 2681–2686.

[127] T. M. Apostol, *Mathematical Analysis*. Addison-Wesley Reading, 1974.

[128] W. Rudin *et al.*, *Principles of Mathematical Analysis*. McGraw-hill New York, 1964, vol. 3.

[129] H. Royden and P. Fitzpatrick, *Real Analysis*, 4th ed. Pearson, 2010.

[130] V. Klema and A. Laub, "The singular value decomposition: Its computation and some applications," *IEEE Transactions on Automatic Control*, vol. 25, no. 2, pp. 164–176, Apr 1980.

[131] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed. Baltimore, Maryland: The Johns Hopkins University Press, 2013.

[132] M. C. Grant and S. P. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*, V. D. Blondel, S. P. Boyd, and H. Kimura, Eds. London: Springer London, 2008, pp. 95–110.

[133] A. A. Rodriguez, *Analysis and Design of Multivariable Feedback Control Systems*. Tempe, AZ: CONTROL3D, L.L.C., 2002.

[134] J. F. Humphreys, *A course in group theory*. New York: Oxford University Press, 1996.

[135] P. Grosdidier, M. Morari, and B. R. Holt, "Closed-loop properties from steady-state gain information," *Industrial & Engineering Chemistry Fundamentals*, vol. 24, no. 2, pp. 221–235, 05 1985. [Online]. Available: https://doi.org/10.1021/i100018a015

[136] A. Packard and J. Doyle, "The complex structured singular value," *Automatica*, vol. 29, no. 1, pp. 71 – 109, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/000510989390175S

[137] J. Chen, "Multivariable gain-phase and sensitivity integral relations and design trade-offs," *IEEE Transactions on Automatic Control*, vol. 43, no. 3, pp. 373–385, Mar 1998.

[138] ——, "Logarithmic integrals, interpolation bounds, and performance limitations in mimo feedback systems," *IEEE Transactions on Automatic Control*, vol. 45, no. 6, pp. 1098–1115, Jun 2000.

[139] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, *Feedback Control Theory*. Macmillan Publishing Co., 1990.

[140] K. J. Åström, "Model uncertainty and robust control," in *Lecture Notes on Iterative Identification and Control Design*, 2000, pp. 63–100.

[141] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*, 2nd ed. Princeton University Press, November 2016.

[142] ——, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[143] K. J. Åström, "Limitations on control system performance," *European Journal of Control*, vol. 6, no. 1, pp. 2 – 20, 2000.

[144] G. Stein, "Respect the unstable," *IEEE Control Systems*, vol. 23, no. 4, pp. 12–25, Aug 2003.

[145] J. Chen, "On logarithmic complementary sensitivity integrals for mimo systems," in *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*, vol. 6, Jun 1998, pp. 3529–3530 vol.6.

[146] A. A. Rodriguez, *Analysis and Design of Feedback Control Systems*. Tempe, AZ: CONTROL3D, L.L.C., 2003.

[147] K. Zhou and J. C. Doyle, *Essentials of Robust Control*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1998, vol. 104.

[148] K. Zhou, J. C. Doyle, K. Glover *et al.*, *Robust and Optimal Control*. Prentice hall New Jersey, 1996, vol. 40.

[149] B. A. Francis, *A Course in $\mathcal{H}^\infty$ Control Theory*, ser. Lecture Notes in Control and Information Sciences. Springer-Verlag Berlin Heidelberg, 1987, vol. 88.

[150] B. A. Francis and J. C. Doyle, "Linear control theory with an $\mathcal{H}^\infty$ optimality criterion," *SIAM Journal on Control and Optimization*, vol. 25, no. 4, pp. 815–844, 1987. [Online]. Available: https://doi.org/10.1137/0325046

[151] J. Maciejowski, *Multivariable Feedback Design*. Wokingham, Berkshire, UK: Addison-Wesley, 1989.

[152] O. Cifdaloz, "$\mathcal{H}^\infty$ mixed-sensitivity optimization for infinite dimensional plants subject to convex constraints," Ph.D. Dissertation, Arizona State University, Tempe, AZ, 2007.

[153] M. Shayeb, "Multivariable control system design via convex optimization," M.S. Thesis, Arizona State University, Tempe, AZ, 2002.

[154] G. Balas, R. Chiang, A. Packard, and M. Safonov, "Robust control toolbox users guide," *The Math Works, Inc*, 2008.

[155] B. M. Chen, *$\mathcal{H}^\infty$ Control and its Applications*, ser. Lecture Notes in Control and Information Sciences. London: Springer-Verlag, 1998, vol. 235.

[156] K. Glover, *H-Infinity Control*. London: Springer London, 2013, pp. 1–9. [Online]. Available: https://doi.org/10.1007/978-1-4471-5102-9_166-1

[157] J. C. Doyle, K. Glover, P. P. Khargonekar, and B. A. Francis, "State-space solutions to standard $\mathcal{H}^2$ and $\mathcal{H}^\infty$ control problems," *IEEE Transactions on Automatic Control*, vol. 34, no. 8, pp. 831–847, Aug 1989.

[158] K. Glover and J. C. Doyle, "State-space formulae for all stabilizing controllers that satisfy an $\mathcal{H}^\infty$-norm bound and relations to relations to risk sensitivity," *Systems and Control Letters*, vol. 11, no. 3, pp. 167 – 172, 1988. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0167691188900552

[159] M. G. Safonov, D. J. N. Limebeer, and R. Y. Chiang, "Simplifying the $\mathcal{H}^\infty$ theory via loop-shifting, matrix-pencil and descriptor concepts," *International Journal of Control*, vol. 50, no. 6, pp. 2467–2488, 1989. [Online]. Available: https://doi.org/10.1080/00207178908953510

[160] A. Packard, K. Zhou, P. Pandey, J. Leonhardson, and G. Balas, "Optimal, constant I/O similarity scaling for full-information and state-feedback control problems," *Systems and Control Letters*, vol. 19, no. 4, pp. 271 – 280, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/016769119290065Z

[161] T. Iwasaki and R. Skelton, "All controllers for the general $\mathcal{H}^\infty$ control problem: LMI existence conditions and state space formulas," *Automatica*, vol. 30, no. 8, pp. 1307 – 1317, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0005109894901104

[162] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, Apr 2004. [Online]. Available: https://doi.org/10.1007/s00158-003-0368-6

[163] Y. H. Sardahi, "Multi-objective optimal design of control systems," Ph.D. Dissertation, UC Merced, Merced, CA, 2016.

[164] A. Gambier and M. Jipp, "Multi-objective optimal control: An introduction," in *2011 8th Asian Control Conference (ASCC)*, May 2011, pp. 1084–1089.

[165] A. Gambier and E. Badreddin, "Multi-objective optimal control: An overview," in *2007 IEEE International Conference on Control Applications*, Oct 2007, pp. 170–175.

[166] W.-Y. Ng, *Interactive Multi-Objective Programming as a Framework for Computer-Aided Control System Design*, 1st ed., ser. Lecture Notes in Control and Information Sciences. Springer-Verlag Berlin Heidelberg, 1989, vol. 132.

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

[168] G. Zames, "Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses," *IEEE Transactions on Automatic Control*, vol. 26, no. 2, pp. 301–320, Apr 1981.

[169] V. Kučera, "Algebraic theory of discrete optimal control for multivariable systems [i.]," *Kybernetika*, vol. 10, no. Suppl, pp. (1), 3–56, 1974. [Online]. Available: http://eudml.org/doc/28093

[170] M. Vidyasagar, *Control System Synthesis: A Factorization Approach, Part I.* Morgan & Claypool Publishers, 2011, vol. 2, no. 1.

[171] ——, *Control System Synthesis: A Factorization Approach, Part II.* Morgan & Claypool Publishers, 2011, vol. 2, no. 1.

[172] M.G.Safonov, E. Jonckheere, M.Vermaj, and D.J.N.Limebeer, "Synthesis of positive real multivariable feedback systems," *International Journal of Control*, vol. 45, no. 3, pp. 817–842, 1987. [Online]. Available: https://doi.org/10.1080/00207178708933772

[173] E. Polak and S. E. Salcudean, "On the design of linear multivariable feedback systems via constrained nondifferentiable optimization in $\mathcal{H}^\infty$ spaces," *IEEE Transactions on Automatic Control*, vol. 34, no. 3, pp. 268–276, March 1989.

[174] P. S. Heuberger, P. M. Van den Hof, and B. Wahlberg, *Modelling and Identification with Rational Orthogonal Basis Functions*. Springer-Verlag London, 2005.

[175] P. S. C. Heuberger, P. M. J. V. den Hof, and O. H. Bosgra, "A generalized orthonormal basis for linear dynamical systems," *IEEE Transactions on Automatic Control*, vol. 40, no. 3, pp. 451–465, Mar 1995.

[176] H. Akçay and B. Ninness, "Orthonormal basis functions for continuous-time systems and L_p convergence," *Mathematics of Control, Signals and Systems*, vol. 12, no. 3, pp. 295–305, Aug 1999.

[177] M. B. Jamoom, "Constrained optimization for hierarchical control system design," M.S. Thesis, Massachusetts Institute of Technology, Boston, MA, 1999.

[178] P. Benner and T. Mitchell, "Faster and more accurate computation of the $\mathcal{H}^\infty$ norm via optimization," *arXiv preprint arXiv:1707.02497*, 2017.

[179] A. Varga and P. Parrilo, "Fast algorithms for solving $\mathcal{H}^\infty$-norm minimization problems," in *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, vol. 1, 2001, pp. 261–266 vol.1.

[180] P. Apkarian, D. Noll, and A. M. Simoes, "Time-domain control design: A nonsmooth approach," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 6, pp. 1439–1445, Nov 2009.

[181] V. Bompart, P. Apkarian, and D. Noll, "Control design in the time and frequency domain using nonsmooth techniques," *Systems & Control Letters*, vol. 57, no. 3, pp. 271–282, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167691107001247

[182] S. P. Boyd and L. Vandenberghe, "Subgradients," 2018.

[183] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, 1st ed., ser. Applied Optimization. New York: Springer, 2004, vol. 87.

[184] A. Lewis, "Nonsmooth optimization and robust control," *Annual Reviews in Control*, vol. 31, no. 2, pp. 167 – 177, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1367578807000351

[185] A. S. Lewis and H. S. Sendov, "Nonsmooth analysis of singular values. part i: Theory," *Set-Valued Analysis*, vol. 13, no. 3, pp. 213–241, Sep 2005. [Online]. Available: https://doi.org/10.1007/s11228-004-7197-7

[186] ——, "Nonsmooth analysis of singular values. part ii: Applications," *Set-Valued Analysis*, vol. 13, no. 3, pp. 243–264, Sep 2005. [Online]. Available: https://doi.org/10.1007/s11228-004-7198-6

[187] J. Borwein and A. S. Lewis, *Convex Analysis and Nonlinear Optimization: Theory and Examples*, 2nd ed. New York: Springer-Verlag London, 2006.

[188] N. Z. Shor, K. C. Kiwiel, and A. Ruszcayǹski, *Minimization Methods for Non-differentiable Functions*. Berlin, Heidelberg: Springer-Verlag, 1985.

[189] L. Lukšan and J. Vlcek, "Test problems for nonsmooth unconstrained and linearly constrained optimization," *Technická zpráva*, vol. 798, 2000.

[190] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. ACM, 1984, pp. 302–311.

[191] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1994, vol. 13.

[192] J. J. E. Kelley, "The cutting-plane method for solving convex programs," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 4, pp. 703–712, 1960.

[193] E. W. Cheney and A. A. Goldstein, "Newton's method for convex programming and tchebycheff approximation," *Numer. Math.*, vol. 1, no. 1, pp. 253–268, Dec. 1959. [Online]. Available: http://dx.doi.org/10.1007/BF01386389

[194] A. Nemirovsky and D. Yudin, "Problem complexity and method efficiency in optimization," *New York*, 1983.

[195] J.-L. Goffin and J.-P. Vial, "On the computation of weighted analytic centers and dual ellipsoids with the projective algorithm," *Mathematical Programming*, vol. 60, no. 1-3, pp. 81–92, 1993.

[196] Y. Nesterov, "Complexity estimates of some cutting plane methods based on the analytic barrier," *Mathematical Programming*, vol. 69, no. 1-3, pp. 149–176, 1995.

[197] D. S. Atkinson and P. M. Vaidya, "A cutting plane algorithm for convex programming that uses analytic centers," *Mathematical Programming*, vol. 69, no. 1-3, pp. 1–43, 1995.

[198] Y. Ye, *Interior Point Algorithms: Theory and Analysis*, ser. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2011.

[199] G. Sonnevend, "An "analytical centre" for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming," in *System Modelling and Optimization*, ser. Lecture Notes in Control and Information Sciences, A. Prékopa, J. Szelezsáan, and B. Strazicky, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 866–875.

[200] ——, *New Algorithms in Convex Programming Based on a Notion of "Centre" (for Systems of Analytic Inequalities) and on Rational Extrapolation*. Basel: Birkhäuser Basel, 1988, pp. 311–326. [Online]. Available: https://doi.org/10.1007/978-3-0348-9297-1_20

[201] ——, "Applications of the notion of analytic center in approximation (estimation) problems," *Journal of Computational and Applied Mathematics*, vol. 28, pp. 349 – 358, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0377042789903464

[202] "Analytic Center Cutting Plane Method (ACCPM) webpage," https://www.maths.ed.ac.uk/~gondzio/software/accpm.html, accessed: 2018-07-20.

[203] A. Kuntsevich and F. Kappel, "Solvopt: The solver for local nonlinear optimization problems," *Institute for Mathematics, Karl-Franzens University of Graz*, 1997.

[204] F. Kappel and A. V. Kuntsevich, "An implementation of shor's r-algorithm," *Computational Optimization and Applications*, vol. 15, no. 2, pp. 193–205, Feb 2000. [Online]. Available: https://doi.org/10.1023/A:1008739111712

[205] H. Fendl and H. Schichl, "A feasible second order bundle algorithm for nonsmooth nonconvex optimization problems with inequality constraints: II. implementation and numerical results," *arXiv preprint arXiv:1506.08021*, 2015.

[206] N. Z. Shor, *Minimization Methods for Non-Differentiable Functions*, ser. Springer Series in Computational Mathematics. Berlin Heidelberg: Springer-Verlag, 1985, vol. 3.

[207] P. I. Stetsyuk, *Shor's r-Algorithms: Theory and Practice*. Cham: Springer International Publishing, 2017, pp. 495–520. [Online]. Available: https://doi.org/10.1007/978-3-319-68640-0_24

[208] J. V. Burke, A. S. Lewis, and M. L. Overton, "The speed of shor's r-algorithm," *IMA Journal of Numerical Analysis*, vol. 28, no. 4, pp. 711–720, Oct 2008.

[209] S. Gumussoy, D. Henrion, M. Millstone, and M. L. Overton, "Multiobjective robust control with HIFOO 2.0," *IFAC Proceedings Volumes*, vol. 42, no. 6, pp. 144 – 149, 2009, 6th IFAC Symposium on Robust Control Design.

[210] J. Burke, D. Henrion, A. Lewis, and M. Overton, "HIFOO - a matlab package for fixed-order controller design and $\mathcal{H}^\infty$ optimization," *IFAC Proceedings Volumes*, vol. 39, no. 9, pp. 339 – 344, 2006, 5th IFAC Symposium on Robust Control Design. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667015335229

[211] F. E. Curtis, T. Mitchell, and M. L. Overton, "A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles," *Optimization Methods and Software*, vol. 32, no. 1, pp. 148–181, 2017. [Online]. Available: https://doi.org/10.1080/10556788.2016.1208749

[212] T. Mitchell, "Robust and efficient methods for approximation and optimization of stability measures," Ph.D. Dissertation, New York University, 2014.

[213] Y. Censor, "Pareto optimality in multiobjective problems," *Applied Mathematics and Optimization*, vol. 4, no. 1, pp. 41–59, Mar 1977. [Online]. Available: https://doi.org/10.1007/BF01442131

[214] "Decision tree for optimization software," http://plato.asu.edu/sub/benchm.html, accessed: 2018-07-20.

[215] D. M. Gay, "Hooking your solver to AMPL," Citeseer, Tech. Rep., 1997.

[216] R. Fourer, D. M. Gay, and B. Kernighan, *AMPL*. Boyd & Fraser Danvers, MA, 1993, vol. 117.

[217] R. Fourer, D. M. Gay, and B. W. Kernighan, "A modeling language for mathematical programming," *Management Science*, vol. 36, no. 5, pp. 519–554, 1990. [Online]. Available: https://doi.org/10.1287/mnsc.36.5.519

[218] "General Algebraic Modeling System (GAMS) website," https://www.gams.com/, accessed: 2018-07-20.

[219] A. Brooke, D. Kendrick, A. Meeraus, R. Raman, and U. America, "The general algebraic modeling system," *GAMS Development Corporation*, vol. 1050, 1998.

[220] A. Forrai, "Robust controller design," in *Embedded Control System Design*, 2013.

[221] M. G. Safonov and R. Y. Chiang, "CACSD using the state-space L^∞ theory – a design example," *IEEE Transactions on Automatic Control*, vol. 33, no. 5, pp. 477–479, May 1988.

[222] J. S. Freudenberg, R. Middleton, and A. Stefanpoulou, "A survey of inherent design limitations," in *Proceedings of the 2000 American Control Conference*, vol. 5, 2000, pp. 2987–3001.

[223] P. Apkarian and D. Noll, "The $\mathcal{H}^\infty$ control problem is solved," *AerospaceLab Journal: Design and Validation of Aerospace Control Systems: New Methods & Tools with Illustrations*, no. 13, pp. 1–11, November 2017.

[224] P. Gahinet and P. Apkarian, "Decentralized and fixed-structure $\mathcal{H}^\infty$ control in matlab," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, Dec 2011, pp. 8205–8210.

[225] P. Apkarian and D. Noll, "Nonsmooth $\mathcal{H}^\infty$ synthesis," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 71–86, Jan 2006.

[226] D. W. Gu, P. Petkov, and M. M. Konstantinov, *Robust control design with MATLAB®*, 2nd ed. Springer-Verlag London, 2013.

[227] G. J. Balas, "Robust control of flexible structures: Theory and experiments," Ph.D. Dissertation, California Institute of Technology, Pasadena, CA, 1990.

[228] K.-Y. Tsai and H. A. Hindi, "DQIT: μ-synthesis without D-scale fitting," *IEEE Transactions on Automatic Control*, vol. 49, no. 11, pp. 2028–2032, Nov 2004.

[229] ——, "DQIT: μ-synthesis without D-scale fitting," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, vol. 1, May 2002, pp. 493–498 vol.1.

[230] K.-Y. Tsai and H. Hindi, "Method for design of multi-objective robust controllers," November 2004, uS Patent App. 10/709,458.

[231] J. Doyle, K. Lenz, and A. Packard, "Design examples using μ-synthesis: Space shuttle lateral axis FCS during reentry," in *1986 25th IEEE Conference on Decision and Control*, Dec 1986, pp. 2218–2223.

[232] E. Prempain and I. Postlethwaite, "A constant D-scale μ-synthesis approach based on nonsmooth optimization," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 15 209 – 15 213, 2008, 17th IFAC World Congress.

[233] A. Packard, J. Doyle, and G. Balas, "Linear, multivariable robust control with a μ perspective," *Journal of Dynamic Systems, Measurement, and Control*, vol. 115, no. 2B, pp. 426–438, 1993.

[234] R. S. da Silva de Aguiar, P. Apkarian, and D. Noll, "Structured robust control against mixed uncertainty," *IEEE Transactions on Control Systems Technology*, pp. 1–11, 2017.

[235] E. F. M. Menezes, R. S. S. Aguiar, A. M. Simoes, and P. Apkarian, "Structured robust controller design via non-smooth mixed μ synthesis," *IET Control Theory Applications*, vol. 10, no. 17, pp. 2186–2193, 2016.

[236] P. Apkarian, M. N. Dao, and D. Noll, "Parametric robust structured control design," *IEEE Transactions on Automatic Control*, vol. 60, no. 7, pp. 1857–1869, July 2015.

[237] P. Apkarian and H. D. Tuan, "Nonsmooth μ synthesis," in *2010 11th International Conference on Control Automation Robotics Vision*, Dec 2010, pp. 917–922.

[238] F. R. Chavez and D. K. Schmidt, "Analytical aeropropulsive-aeroelastic hypersonic-vehicle model with dynamic analysis," *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 6, pp. 1308–1319, 2018/04/11 1994. [Online]. Available: https://doi.org/10.2514/3.21349

[239] A. A. Rodriguez, *Linear Systems Analysis and Design*. Tempe, AZ: CONTROL3D, L.L.C., 2004.

[240] G. C. Goodwin, M. E. Salgado, and E. I. Silva, "Time-domain performance limitations arising from decentralized architectures and their relationship to the RGA," *International Journal of Control*, vol. 78, no. 13, pp. 1045–1062, 2005. [Online]. Available: https://doi.org/10.1080/00207170500226016

[241] F. Yang, Z. Wang, Y. S. Hung, and H. Shu, "Mixed $\mathcal{H}^2/\mathcal{H}^\infty$ filtering for uncertain systems with regional pole assignment," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 2, pp. 438–448, April 2005.

[242] M. Chilali, P. Gahinet, and P. Apkarian, "Robust pole placement in LMI regions," *IEEE Transactions on Automatic Control*, vol. 44, no. 12, pp. 2257–2270, Dec 1999.

[243] ——, "Robust pole placement in lmi regions," in *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 2, Dec 1997, pp. 1291–1296.

[244] M. Ahookhosh, "High-dimensional nonsmooth convex optimization via optimal subgradient methods," Ph.D. Dissertation, University of Vienna, 2015.

[245] ——, "User's manual for OSGA (optimal subgradient algorithm)," Vienna, Austria, January 2015.

[246] ——, "Optimal subgradient methods: computational properties for large-scale linear inverse problems," *Optimization and Engineering*, Mar 2018.

[247] J. Burke, A. Lewis, and M. Overton, "A robust gradient sampling algorithm for nonsmooth, nonconvex optimization," *SIAM Journal on Optimization*, vol. 15, no. 3, pp. 751–779, 2005.

[248] "Nonsmooth optimization (NSO) software," http://napsu.karmitsa.fi/nsosoftware/, accessed: 2018-07-20.

[249] M. M. Mäkelä, N. Karmitsa, and A. Bagirov, "Subgradient and bundle methods for nonsmooth optimization," in *Numerical Methods for Differential Equations, Optimization, and Technological Problems: Dedicated to Professor P. Neittaanmäki on His 60th Birthday*, S. Repin, T. Tiihonen, and T. Tuovinen, Eds. Dordrecht: Springer Netherlands, 2013, pp. 275–304.

[250] N. Karmitsa, A. Bagirov, and M. M. Mäkelä, "Comparing different nonsmooth minimization methods and software," *Optimization Methods and Software*, vol. 27, no. 1, pp. 131–153, 2012.

[251] ——, "Empirical and theoretical comparisons of several nonsmooth minimization methods and software," TUCS Technical Report 959, Turku Centre for Computer Science, Tech. Rep., 2009.

[252] K. Glover, J. Lam, and J. R. Partington, "Rational approximation of a class of infinite-dimensional systems i: Singular values of hankel operators," *Mathematics of Control, Signals and Systems*, vol. 3, no. 4, pp. 325–344, Dec 1990. [Online]. Available: https://doi.org/10.1007/BF02551374

[253] ——, "Rational approximation of a class of infinite-dimensional systems ii: Optimal convergence rates of $\mathcal{L}^\infty$ approximants," *Mathematics of Control, Signals and Systems*, vol. 4, no. 3, pp. 233–246, Sep 1991. [Online]. Available: https://doi.org/10.1007/BF02551279

[254] P. Bodin, L. Villemoes, and B. Wahlberg, "Selection of best orthonormal rational basis," *SIAM Journal on Control and Optimization*, vol. 38, no. 4, pp. 995–1032, 2000. [Online]. Available: https://doi.org/10.1137/S036301299732818X

APPENDIX A

MATLAB CODE

All the MATLAB codes presented below are written in the following MATLAB and Java versions respectively:

- MATLAB Version: `9.1.0.441655 (R2016b)`.
- Java Version: `Java 1.7.0_75-b13` with Oracle Corporation Java HotSpot™ 64-Bit Server VM mixed mode.

A.1 Design Using Generalized Mixed Sensitivity

A.1.1 GMS Main Code (gms_main.m)

```
% Main File for Control Design using Generalized Mixed Sensitivity(GMS)
%
% Computes a H-infinity based Feedback Controller based on
% multiobjective constrained convex optimization.
%
% Outline of steps for GMS problem setup:
%   - Form the design plant:
%       - Define the original plant
%       - Integrator augmentation if needed
%       - Bilinear transformation values if needed
%
%   - Select weighting functions:
%       - Tradeoff param rho
%       - W for obj
%       - W for constraint
%
%   - Select optimization params:
%       - LB and UB
%       - Init point
%       - Maximum number of iterations
%
%   - Select Youla/Zames parametrization:
%       - Select Youla or Zames
%       - Initial controller
%
%   - Finite Dimensionality
%       - Basis params
%
%   - Objective function:
%           - sum/max/stacking
%
%   - Find initial controller (Ko, F, L)
%
%   - Youla parameterization
%
%   - Find Initial Q parameter using initial controller (Ko, F, L)
%
%   - Extract required data from problem setup
%
```

```
%    - Vectorize the optimization problem
%
%    - Optimization process
%       - define how subgradient is picked based on sum/max/stacking
%
%    - form Q using the optimized variables and bases
%
%    - form Controller K using the obtained Q
%
%    - Inverse bilinear transformation if needed
%
%    - Inverse of integrator augmentation if needed
%
%    - Compute OL and CL maps

%% Initial Code Setup
clear;
close all;
% clc;
% warning off;

% Transfer function variable
s = tf('s');

%% Design Plant

% % ------- Select one of the following PlntLabel ------- %
% PlntLabel='SISO_Stable'; Bilinear=0; AugInteg=0;
PlntLabel='acad_2by2'; Bilinear=0; AugInteg=0;
% PlntLabel='hsv_io'; Bilinear=1; AugInteg=0; AugTwoChannel=1;
% PlntLabel='1bys'; Bilinear=0; AugInteg=0;

% %-------------- Plant --------------%
switch PlntLabel
    case 'SISO_Stable'
        P_tf = tf([1],[1 1]);
        P_ss = ss(P_tf);
        [Ap, Bp, Cp, Dp] = ssdata(P_ss);

    case 'acad_2by2'
        s=tf('s');
        P_tf = 1/s * [10 9; 9 8]; % Doyle Example
        P_orig = P_tf;
        %         rolloff = 100/(s+100);
        %         P_tf = series(rolloff,P_tf);
        P_design = P_tf;
        P_ss=ss(P_tf);
        [Ap, Bp, Cp, Dp] = ssdata(P_ss);

    case 'hsv_io'
        %         % %------------ Flexible model ------------%
        %         A = [
        %             -0.0008 -0.0006  0.0000  0.0001 -0.0005 ...
           -0.0000 -0.0012...
        %             0  -0.0009  0
        %             0.0398  -0.1068 -0.0000  0.1068 -0.0565  ...
           0.0002 -0.0502...
```

```
%             0  -0.1122  0
%             -8.1371 -6.4875 -0.0018  6.4875 -2.6934 ...
    -0.0500 -13.9072...
%             0  -3.5889  0
%             0       0       1       0       0       0 ...
          0       ...
%             0   0       0
%             0       0       0       0       0       1 ...
          0       ...
%             0   0       0
%             97.7307 -175.4193  0  175.4193 -486.4654 ...
    -0.7903 -62.4896...
%             0 -194.3275  0
%             0       0       0       0       0       0 ...
          0       ...
%             1   0       0
%             -29.74  -8.29    0       8.29    6.187  0 ...
          1.578   ...
%             0  -8995 -3.796
%             ];
%         B = [
%             0.07128  -0.0006543
%             0.242     0.01766
%             -34.65    -9.551
%             0          0
%             0          0
%             -20.3    39.96
%             0          0
%             176.1   -25.53
%             0          0
%             -94.96   -4.327
%             ];
%         C = [
%             1    0    0    0    0    0    0    0    0    0
%             0    1    0    0    0    0    0    0    0    0
%             ];
%         D = [
%             0    0
%             0    0
%             ];
%
%         P_ss=ss(Ap,Bp,Cp,Dp);
%         P_ss_TwoChannel=P_ss; % backup the original plant ...
    (2-output)
%
%         % Parameters for bilinear transformation
%         p2 = -1e20; p1 = -0.001;
%
%         % Augment integrator at output, in first two channels
%         AugTwoChannel=1;
%         if AugTwoChannel==1
%             s=tf('s');
%             [Ap,Bp,Cp,Dp]=ssdata(P_ss);
%
%             % Make it 3-channel.
%             Cps=[P_ss_TwoChannel.c; 0 0 0 1 zeros(1,6)];
%             % First two states are of integrator
```

```
%             P_ss_ThreeChannel=ss(Ap,Bp,Cps,[]); % backup
%             M=[0 0 0 1 zeros(1,6)];
%
%             % Augment integrator in first two output ...
   channels of plant
%             Ap=blkdiag(Ap,zeros(size(Cp,1)));
%             Ap(end-size(Cp,1)+1:end,1:size(Cp,2))=Cp;
%             Bp=[Bp; zeros(size(Cp,1),size(Bp,2))];
%             Cp=[zeros(2,10), eye(2)];
%             Cp=[Cp; zeros(1,3) 1 zeros(1,8)];
%             P_ss=ss(Ap,Bp,Cp,[]);
%             P_ss_ThreeChannel_AugInteg=P_ss; % Backup
%         else
%             % Make it 3-channel
%             Cps=[P_ss_TwoChannel.c; 0 0 0 1 zeros(1,6)];
%             P_ss=ss(Ap,Bp,Cps,[]); [Ap,Bp,Cp,Dp]=ssdata(P_ss);
%             P_ss_ThreeChannel=P_ss; % Backup
%         end

% %------------ Rigid model ------------%
Ap=[-0.0008659  -0.0004395   9.981e-09  -0.0001174
    0.02865    -0.08627  -2.467e-06     0.08627
    -8.706      -5.512   -0.001827       5.512
    0            0           1           0];
Bp =[  0.07122  -0.0006823
    0.242     0.01353
    -35.54      -9.621
    0           0];
Cp =[1   0   0   0
    0   1   0   0
    0   0   0   1];
Dp =[0   0
    0   0
    0   0];
P_ss=ss(Ap,Bp,Cp,Dp);
P_ss_TwoChannel=P_ss(1:2,:); % backup original plant (2-output)

% Parameters for bilinear transformation
p2 = -1e20; p1 = -0.001;

% Augment integrator at output, in first two channels
if AugTwoChannel==1
    s=tf('s');
    [Ap,Bp,Cp,Dp]=ssdata(P_ss_TwoChannel);

    % Make it 3-channel.
    P_ss_ThreeChannel=P_ss; % backup
    M=[0 0 0 1];

    % Augment integrator in first two output channels of plant
    Ap=blkdiag(Ap,zeros(size(Cp,1)));
    Ap(end-size(Cp,1)+1:end,1:size(Cp,2))=Cp;
    Bp=[Bp; zeros(size(Cp,1),size(Bp,2))];
    Cp=[zeros(2,4), eye(2)];
    Cp=[Cp; zeros(1,3) 1 zeros(1,2)];
    P_ss=ss(Ap,Bp,Cp,[]);
    P_ss_ThreeChannel_AugInteg=P_ss; % Backup
```

```
        else
            % Make it 3-channel
            Cps=[P_ss_TwoChannel.c; 0 0 0 1];
            P_ss=ss(Ap,Bp,Cps,[]); [Ap,Bp,Cp,Dp]=ssdata(P_ss);
            P_ss_ThreeChannel=P_ss; % Backup 3-output
        end

        P_ss=ss(Ap,Bp,Cp,[]);
        P_tf=tf(P_ss);

    case '1bys'
        P_tf = tf([1],[1 0]);%*[1 0.1; 0.1 1];
        P_ss = ss(P_tf);
        % s=tf('s'); P_tf=P_tf/s; P_ss=series(ss(0,1,1,0),P_ss);
        [Ap, Bp, Cp, Dp] = ssdata(P_ss);

end

% %-------- Integrator augmentation at output if needed -------%
if AugInteg==1
    Integ=ss(0,1,1,0);
    P0=P_ss;
    P_ss=series(Integ,P_ss);
    [Ap, Bp, Cp, Dp] = ssdata(P_ss);
end

% %------------ Size of design plant ------------%
[n_e, n_u] = size(P_ss);

%% Bilinear Transformation
if  Bilinear==1
    P_ss_BeforeBilin=P_ss; % Backup plant before bilin transform
    [Ap,Bp,Cp,Dp]=bilin(P_ss.a,P_ss.b,P_ss.c,P_ss.d,1,'Sft_jw',[p2 p1]);
    P_ss=ss(Ap,Bp,Cp,Dp);
    P_ss_BilinPlnt=P_ss; % backup bilin transformed plant
end

%% Objective Weighting Functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Design parameters: Weights for multiobjective function
% mu corresponds to weight on properties at plant output,
% rho corresponds to weight on properties at plant input
% If eta is defined, it corresponds to weight on properties at ...
   sensor noise
% for the inner-outer loop case

mu1=1; mu2=1; mu3=1; rho1=1; rho2=1; rho3=1;
% eta1=1e-1; eta3=0;

switch PlntLabel
    case {'SISO_Stable'}
        Eps=0.01;
        Ms=1.5; wb=10;
        W1 = tf([1/Ms wb], [1 wb*Eps]);
        % % W1 = (1/Ms)*tf([1 nthroot(Ms,k1)*wb],[1 ...
           nthroot(Eps,k1)*wb])^k1;
```

```
    Mu=1/30; wbu=750;
    W2 = [tf([1 wbu*Mu],[Eps wbu])];
    % % W2 = (1/Eps)*tf([1 wbu*nthroot(Mu,k2)],[1 ...
       wbu/nthroot(Eps,k2)])^k2;
    My=30; wbc=1000;
    % W3 = tf([1 wbc/My], [Eps wbc]);
    % % W3 = (1/Eps)*tf([1 wbc/nthroot(My,k3)],[1 ...
       wbc/nthroot(Eps,k3)])^k3;
    W3 = ss(1);
    Wd1=W1(1,1);
    wd21=0.1; wd22=1; wd23=10; s=tf('s');
    Wd2=((wd21/(s+wd21))*((s+wd22)/wd22)^2*(wd23/(s+wd23)));
    Wd3=W3(1,1);
    W1=mu1*W1; W2=mu2*W2; W3=mu3*W3;
    W1=ss(W1); W2=ss(W2); W3=ss(W3);
    Wd1=rho1*Wd1; Wd2=rho2*Wd2; Wd3=rho3*Wd3;
    Wd1=ss(Wd1); Wd2=ss(Wd2); Wd3=ss(Wd3);

case {'acad_2by2'}
    Eps=0.01; Ms=2; wb1=1; wb2=1;
    W1 = [tf([1/Ms wb1], [1 wb1*Eps]) 0; 0 tf([1/Ms wb2], [1 ...
       wb2*Eps])];
    Eps=0.01; Ms=2; wb1=2; wb2=2;
    Wd1 = [tf([1/Ms wb1], [1 wb1*Eps]) 0; 0 tf([1/Ms wb2], [1 ...
       wb2*Eps])];
    W2 = ss(eye(2));
    W3 = ss(eye(2));
    Wd2 = ss(eye(2));
    Wd3 = ss(eye(2));
    W1=mu1*W1; W2=mu2*W2; W3=mu3*W3;
    W1=ss(W1); W2=ss(W2); W3=ss(W3);
    Wd1=rho1*Wd1; Wd2=rho2*Wd2; Wd3=rho3*Wd3;
    Wd1=ss(Wd1); Wd2=ss(Wd2); Wd3=ss(Wd3);

case 'hsv_io'
    % Weights for case: Two loop-breaking points
    Eps1=0.01; Ms1=1.08; wb1=0.01; Ms2=1.08; wb2=0.01;
    W1 = [tf([1/Ms1 wb1], [1 wb1*Eps1]) 0; 0 tf([1/Ms2 wb2],...
        [1 wb2*Eps1])];
    Eps2=1; Mu1=0.1; wbu1=1000; Mu2=0.1; wbu2=1000;
    W2 = [tf([1 wbu1*Mu1],[Eps2 wbu1]) 0; 0 tf([1 wbu2*Mu2],...
        [Eps2 wbu2])];
    Eps3=0.01; My=1.3; wbc=100;
    W3 = tf([1 wbc/My], [Eps3 wbc])*eye(2);
    Epsd1=0.01; Msd1=1.05; wbd1=0.18;
    Wd1=tf([1/Msd1 wbd1], [1 wbd1*Epsd1])*eye(n_u);
    wd21=1; wd22=1; wd23=1;
    Wd2=((wd21/(s+wd21))*((s+wd22)/wd22)^2*(wd23/(s+wd23)))*eye(2);
    Epsd3=0.01; Myd=1.3; wbcd=1000;
    Wd3=tf([1 wbcd/Myd], [Epsd3 wbcd])*eye(n_u);
    W1=mu1*W1; W2=mu2*W2; W3=mu3*W3;
    W1=ss(W1); W2=ss(W2); W3=ss(W3);
    Wd1=rho1*Wd1; Wd2=rho2*Wd2; Wd3=rho3*Wd3;
    Wd1=ss(Wd1); Wd2=ss(Wd2); Wd3=ss(Wd3);

    %         % Weights for case: Three loop-breaking points
    %         Eps1=0.01; Ms1=1.08; wb1=0.01; Ms2=1.08; wb2=0.01;
```

```
        %         W1 = [tf([1/Ms1 wb1], [1 wb1*Eps1]) 0; 0 tf([1/Ms2 ...
           wb2],...
        %              [1 wb2*Eps1])];
        %         Eps2=1; Mu1=0.1; wbu1=2000; Mu2=0.1; wbu2=2000;
        %         W2 = [tf([1 wbu1*Mu1],[Eps2 wbu1]) 0; 0 tf([1 ...
           wbu2*Mu2],...
        %              [Eps2 wbu2])];
        %         Eps3=0.01; My=1.3; wbc=100;
        %         W3 = tf([1 wbc/My], [Eps3 wbc])*eye(2);
        %         Epsd1=0.01; Msd1=1.0; wbd1=0.1;
        %         Wd1=tf([1/Msd1 wbd1], [1 wbd1*Epsd1])*eye(n_u);
        %         wd21=1; wd22=1; wd23=1;
        %         ...
           Wd2=((wd21/(s+wd21))*((s+wd22)/wd22)^2*(wd23/(s+wd23)))*eye(2);
        %         Epsd3=0.01; Myd=1.2; wbcd=1000;
        %         Wd3=tf([1 wbcd/Myd], [Epsd3 wbcd])*eye(n_u);
        %         Epsni1=0.1;
        %         Mu1=0.001; wbu1=520; Mu2=0.001; wbu2=520;
        %         Wni1 = [(1/sqrt(Epsni1)*tf([1 wbu1*sqrt(Mu1)],[1 ...
           wbu1...
        %              /sqrt(Epsni1)]))^2 0; 0 (1/sqrt(Epsni1)*tf([1 ...
        %              wbu2*sqrt(Mu2)],[1 wbu2/sqrt(Epsni1)]))^2];
        %         Wni1 = Wni1(1,1)*eye(n_u);
        %         Wni3 = Wd1(1,1);
        %         W1=mu1*W1; W2=mu2*W2; W3=mu3*W3;
        %         W1=ss(W1); W2=ss(W2); W3=ss(W3);
        %         Wd1=rho1*Wd1; Wd2=rho2*Wd2; Wd3=rho3*Wd3;
        %         Wd1=ss(Wd1); Wd2=ss(Wd2); Wd3=ss(Wd3);
        %         Wni1=eta1*Wni1; Wni1=ss(Wni1);
        %         Wni3=eta3*Wni3; Wni3=ss(Wni3);

    case '1bys'
        Eps = 0.00001;
        Ms = 10; wb = 1; k1 = 1;
        W1 = (1/Ms)*(s+nthroot(Ms,k1)*wb)^k1/((s+wb*Eps*0.1/0.1^k1)...
            *(s+0.1*wb)^(k1-1));
        W2 = ss(1);
        My = 10; wbc = 1; k3 = 1;
        W3 = (1/Eps)*(s+wbc/nthroot(My,k3))^k3/((s+10*wbc)^(k3-1)*...
            (s+wbc*10/(Eps*10^k3)));
        Wd1 = W1;
        Wd2 = W2;
        Wd3 = W3;

        W1=mu1*W1; W2=mu2*W2; W3=mu3*W3;
        W1=ss(W1); W2=ss(W2); W3=ss(W3);
        Wd1=rho1*Wd1; Wd2=rho2*Wd2; Wd3=rho3*Wd3;
        Wd1=ss(Wd1); Wd2=ss(Wd2); Wd3=ss(Wd3);
end


%% Constraint Weighting Functions

constr_flag = 1;  % 0 = unconstrained, 1 = constrained.

if constr_flag == 0
    W1c=[];
```

```
    W2c=[];
    W3c=[];
    Wd1c=[];
    Wd2c=[];
    Wd3c=[];

elseif constr_flag == 1
    W1c=[];
    W2c=[];
    W3c=[];
    Wd1c=[];
    Wd2c=[];
    Wd3c=[];

    W2c{1}.tfm = ss(1)*eye(n_u);            % Constraint Weigthing
    W2c{1}.Fun = 'f_Hinf';                  % Constraint Type
    W2c{1}.Val = 20;                        % Constraint Value

    %     W2c{1}.tfm = ss(1)*eye(n_u);      % Constraint Weigthing
    %     W2c{1}.Fun = 'f_Linf';            % Constraint Type
    %     W2c{1}.Val = 6;                   % Constraint Value
    %
    %     W2c{1}.tfm = ss(1)*eye(n_u);      % Constraint Weigthing
    %     W2c{1}.Fun = 'f_Linf';            % Constraint Type
    %     W2c{1}.Val = [Inf Inf; Inf 24];   % Constraint Value

else
    disp('Set constr_flag to indicate unconstrained or constrained')
end

%% Weighting functions data structure
weights.W1 = W1;
weights.W2 = W2;
weights.W3 = W3;
weights.Wd1 = Wd1;
weights.Wd2 = Wd2;
weights.Wd3 = Wd3;

weights.W1c = W1c;
weights.W2c = W2c;
weights.W3c = W3c;
weights.Wd1c = Wd1c;
weights.Wd2c = Wd2c;
weights.Wd3c = Wd3c;

if (exist('Wni1','var'))
    weights.Wni1 = Wni1;
end
if (exist('Wni3','var'))
    weights.Wni3 = Wni3;
end

%% Finite Dimensionality: Basis parameters

Basis.n    = 5;
Basis.type = 2;
Basis.p    = 10;
```

```
Basis.z    = 10;

N = Basis.n;

%% Youla/Zames Parameterization
% Youla=1 or Zames=0; Type of parameterization.
% Zames only for stable plant, zero (0) initial controller
YoulaOrZames=1;

%% Optimization Parametrs: Bounds and initial point
xmax1 = 100; xmin1 = -100;
x01 = 1;  % Initial point for optimization xk'
MaxIter = 100;

% Form the vector of LB, UB and initial point
x0=x01*ones(N*n_u*n_e,1);
xmax=xmax1*ones(N*n_u*n_e,1); xmin=xmin1*ones(N*n_u*n_e,1);

%% Optimization to find Controller
SumOrMax =1;  % 1=Max, 2=Sum.

%% Nominal Controller
[Ko,F,L]=f_KNominal(P_ss);

%% Coprime factorization
if YoulaOrZames==1
    % --------- Classic P*K structure (no inner-outer) ---------%
    [T11rz, T12rz, T21rz,T11dz, T12dz, T21dz]=f_CoprFac(P_ss,...
        F,L, weights);

    %     %--------- Added for Hypersonic inner-outer ---------%
    %     [T11rz, T12rz, T21rz,T11dz, T12dz, ...
       T21dz]=f_CoprFac_hsvio(P_ss,...
    %         F,L, weights);

    %     %--------- Added for Hypersonic inner-outer WITH Tniu ...
       ---------%
    %     [T11rz, T12rz, T21rz, T11dz, T12dz, T21dz,T11niz,T12niz,...
    %         T21niz]=f_CoprFac_hsvio_Tniu(P_ss,F,L, weights);

    %
else
    [T11rz, T12rz, T21rz,T11dz, T12dz, T21dz]=f_CoprFac_ZamesParam...
        (P_ss,F,L, weights);
end

%% Initial Q-parameter
n_x=size(Ap,1); n_e=size(Cp,1); n_u=size(Bp,2);
N = Basis.n;
q = f_Basis(N, Basis.p, Basis.z, Basis.type);
% x0 = x01*ones(N*n_u*n_e,1);
Q = f_FormQN(x0, q, n_u, n_e, N);
xk=x0;

%% Problem Data

% %--------- Classic P*K structure (no inner-outer) ---------%
```

```
[n_e, n_u, ProblemDatarz, ProblemDatadz] = f_GenData(P_ss, weights);

% %--------- Added for Hypersonic inner-outer ---------%
% % P_ss1=P_ss(1:2,1:2); n_u=2; n_e=2;% Added for HSV inner-outer
% [n_e, n_u, ProblemDatarz, ProblemDatadz] = f_GenData_hsvio(P_ss, ...
    weights);

% --------- Added for Hypersonic inner-outer WITH Tniu ---------%
% [n_e, n_u, ProblemDatarz, ProblemDatadz, ProblemDataniz] = ...
    f_GenData_hsvio_Tniu(P_ss, weights);


Datarz=ProblemDatarz; Datadz=ProblemDatadz;
% --------- Added for Hypersonic inner-outer WITH Tniu ---------%
% Datarz=ProblemDatarz; Datadz=ProblemDatadz;  Dataniz=ProblemDataniz;

%% Vectorization
[Mrz, Mobjrz, ...
    Mconrz]=f_Vectorize(T11rz,T12rz,T21rz,q,N,n_u,n_e,ProblemDatarz);
[Mdz, Mobjdz, ...
    Mcondz]=f_Vectorize(T11dz,T12dz,T21dz,q,N,n_u,n_e,ProblemDatadz);

% [Mrzc, Mobjrzc, Mconrzc] = f_Vectorize(T11rz1c, T11rz1c, T11rz1c, ...
    q, N, n_u, n_e, ProblemDatarzc);

% --------- Added for Hypersonic inner-outer WITH Tniu ---------%
% [Mniz, Mobjniz, ...
    Mconniz]=f_Vectorize(T11niz,T12niz,T21niz,q,N,n_u,n_e,ProblemDataniz);


%% Optimization process
NQ=N;
N = length(xk);          % Dimension of problem

algo = 2; % 1=ACCPM, 2=Kelley's CPM, 3=SolvOpt.

if algo == 1
    % -------- ACCPM -------- %
    switch SumOrMax
        case 1
            % % Weighted Minmax

            [xk,fx,iter_cnt,perf_meas]=...
                f_ACCPM_GenMixSens_Optimizer(N,NQ,xk,Mobjrz,Mobjdz,...
                Mconrz,Mcondz,T11rz, T12rz, T21rz,T11dz, T12dz, ...
                T21dz,Datarz,Datadz,Q,q,n_u,n_e,xmax,xmin,MaxIter);

%             %--------- Added for HSV IO WITH Tniu ---------%
%             [xk,fx,iter_cnt,perf_meas]=...
%                 f_ACCPM_GenMixSens_Optimizer_With_Tniu...
%                 (N,NQ,xk,Mobjrz,Mobjdz,Mobjniz,Mconrz,Mcondz,...
%                 Mconniz,T11rz, T12rz, T21rz,T11dz, T12dz, T21dz,...
%                 T11niz,T12niz,T21niz, Datarz, Datadz,Dataniz, Q,q,...
%                 n_u,n_e,xmax,xmin,MaxIter);
%
        case 2
            % % Weighted Sum
```

```
            [xk,fx,iter_cnt,perf_meas]=...
                f_ACCPM_GenMixSens_Optimizer_Sum...
                (N,NQ,xk,Mobjrz,Mobjdz,Mconrz,Mcondz,...
                T11rz, T12rz, T21rz,T11dz, T12dz, T21dz,...
                Datarz, Datadz, Q,q,n_u,n_e,xmax,xmin,MaxIter);
    end

elseif algo == 2

    % -------- KELLEY'S CPM -------- %

    switch SumOrMax
        case 1
            %         Weighted Minmax

            [xk,frz,fdz]=f_KelleyCPM_GenMix_Optimizer...
                (N,NQ,xk,Mobjrz,Mobjdz,Mconrz,Mcondz,T11rz,T12rz,...
                T21rz,T11dz, T12dz, T21dz, Datarz, Datadz, Q,q,n_u,...
                n_e,MaxIter,xmax,xmin);

%             %--------- Added for HSV IO WITH Tniu ---------%
%             [xk,frz,fdz]=f_KelleyCPM_GenMix_Optimizer_With_Tniu...
%                 (N,NQ,xk,Mobjrz,Mobjdz,Mobjniz,Mconrz,Mcondz,T11rz,...
%                 T12rz, T21rz,T11dz, T12dz, T21dz,T11niz,T12niz,...
%                 T21niz, Datarz, Datadz,Dataniz, ...
   Q,q,n_u,n_e,MaxIter,...
%                 xmax,xmin);

        case 2
            %         Weighted sum
            [xk,fo]=f_KelleyCPM_GenMix_Optimizer_Sum...
                (N,NQ,xk,Mobjrz,Mobjdz,Mconrz,Mcondz,T11rz,...
                T12rz, T21rz,T11dz, T12dz, T21dz, Datarz,Datadz,...
                Q,q,n_u,n_e,MaxIter,xmax,xmin);
    end

elseif algo == 3
    % -------- SOLVOPT -------- %

    opts(1) = -1;       % negative => minimization
    opts(2) = 1e-4;
    opts(3) = 1e-4;
    opts(4) = MaxIter;  % default num iter 15000
    opts(5) = 0;        % 1->verbose, 0->silent
    NQ = N/(n_u*n_e);
    [xk_solvopt,fx_solvopt,opts_solvopt] = solvopt(x0,...
        @(x)solvopt_fval(x,NQ,Mobjrz,Mobjdz,Mconrz,Mcondz,T11rz,...
        T12rz, T21rz,T11dz, T12dz, T21dz, Datarz, Datadz, q,n_u,...
    n_e),@(x)solvopt_sg(x,NQ,Mobjrz,Mobjdz,Mconrz,Mcondz,T11rz,...
    T12rz, T21rz,T11dz, T12dz, T21dz, Datarz, Datadz, q,n_u,n_e),opts);

else
    disp('Choose a valid algorithm')
end

%% Form Q
Q = f_FormQN(xk, q, n_u, n_e, NQ);
```

```
disp(' ')

%%
[forz, Gfo] =feval('f_Hinf', Mobjrz, xk, T11rz, T12rz, T21rz, Q, ...
   Datarz.ObjVec);
[fodz, Gfo] =feval('f_Hinf', Mobjdz, xk, T11dz, T12dz, T21dz, Q, ...
   Datadz.ObjVec);
fx=max([forz,fodz]);

%% Form K
if YoulaOrZames==1
    K=f_FormK(P_ss,Q,F,L); % youla parameterization
else
    K=f_FormK_ZamesParam(P_ss,Q,F,L); % Zames parameterization
end

T_copr.T11rz = T11rz;
T_copr.T12rz = T12rz;
T_copr.T21rz = T21rz;

T_copr.T11dz = T11dz;
T_copr.T12dz = T12dz;
T_copr.T21dz = T21dz;

% --------- Added for Hypersonic inner-outer WITH Tniu ---------%
% T_copr.T11niz = T11niz;
% T_copr.T12niz = T12niz;
% T_copr.T21niz = T21niz;

%% *************** Inverse Bilinear Transformations ***************
if  Bilinear==1
    [Acp1,Bcp1,Ccp1,Dcp1] = ssdata(K);
    K_BeforeInvBilin=K; % Backup K before inverse bilin transformation
    [Atk1,Btk1,Ctk1,Dtk1]=bilin(Acp1,Bcp1,Ccp1,Dcp1,-1,'Sft_jw',[p2 ...
       p1]);
    K=ss(Atk1,Btk1,Ctk1,Dtk1);
    P_ss=P_ss_BeforeBilin;
end

%% Analysis of Control Design

% Frequency vector
wvec2=logspace(-3,3,1000); tvec=linspace(0,10,100);

% Form open and closed loop maps
if strcmp(PlntLabel,'hsv_io')
    K_Design=K; % Backup design K (without integ aug)
    K = minreal(K);

    if AugTwoChannel==1
        %         % Augment integrator at input, in all channels
        %         K=series(K,1/s);
        % Augment integrator at output, in first two channels
        Kouter=series(1/s,K(:,1:2));
        Kinner=K(:,3:end);
    else
        Kouter=K(:,1:2);
```

```
        Kinner=K(:,3:end);
    end

    if AugTwoChannel==1
        K=series(blkdiag(1/s, 1/s, 1),K);
    end

    %  Add ROLL-OFF if needed
    K_NoRolloff=K; % backup
    %     K=series(K_NoRolloff,(58/(s+58))^2);

    P_ss=P_ss_TwoChannel; % Plant 2-outputs, without integ

    [Lo,Li,So,Si,To,Ti,KS,PS,Tniy,Tniu]=f_CLMapInnerOuter_BigK...
        (P_ss_TwoChannel,K,M);

    % Modify weights to remove near-zero dummy values
    W1=W1(1:size(So,1),1:size(So,1));
    W3=W3(1:size(To,1),1:size(To,1));
    Wd2=Wd2(1:size(KS,1),1:size(KS,1));
    n_e=2; n_u=2;

else
    %     K = minreal(K);
    % Standard feedback (no inner loop)
    [Lo,Li,So,Si,To,Ti,KS,PS] = f_CLTFM(P_ss,K);
end

K_gms=K;


%% Display max_xk and isstable(To)
max_xk=max(abs(xk))

isstab=isstable(To)


%% CL Performance and Robustness

NormInf = mag2db([hinfnorm(So), hinfnorm(Si), hinfnorm(KS), ...
    hinfnorm(PS), hinfnorm(To), hinfnorm(Ti)])
PerformMeasOutOrigWts=norm([W1*So; W2*KS; W3*To],inf);
PerformMeasInOrigWts=norm([Wd1*Si; Wd2*PS; Wd3*Ti],inf);

% % Bandwidth/crossovers
% BW_20_So = min(getGainCrossover(So,0.1))
% BW_20_Si = min(getGainCrossover(Si,0.1))
% BW_20_To = max(getGainCrossover(To,0.1))
% BW_20_Ti = max(getGainCrossover(Ti,0.1))
% BW_0_KS = max(getGainCrossover(KS,1))
% BW_0_PS = max(getGainCrossover(PS,1))
% % BW_0_Tniu = max(getGainCrossover(Tniu,1))
% BW_0_Lo = max(getGainCrossover(Lo,1))
% BW_0_Li = max(getGainCrossover(Li,1))

% % Time domain properties
% To_stepinfo = stepinfo(To);
```

```
% % v_ts = To_stepinfo(1,1).SettlingTime
% % gamma_ts = To_stepinfo(2,2).SettlingTime
% ts1 = To_stepinfo(1,1).SettlingTime
% ts2 = To_stepinfo(2,2).SettlingTime
% KS_stepinfo = stepinfo(KS);
% % peak_FER = KS_stepinfo(1,1).Peak
% % peak_elev = KS_stepinfo(2,2).Peak
% u_peak1 = KS_stepinfo(1,1).Peak
% u_peak2 = KS_stepinfo(2,2).Peak
```

A.1.2 Nominal Controller Design (f_KNominal.m)

```
function [Ko,F,L]=f_KNominal(P_ss)
% Nominal Controller
[Ap, Bp, Cp, Dp] = ssdata(P_ss);
n_x=size(Ap,1); n_e=size(Cp,1); n_u=size(Bp,2);
F = lqr(Ap, Bp, 1e0*eye(n_x), 1.5e1*eye(n_u));
L = lqr(Ap',Cp',1e0*eye(n_x), 1.5e1*eye(n_e));
L=L';
Ko = ss(Ap-Bp*F-L*Cp+L*Dp*F, -L, -F, 0);
```

A.1.3 Youla et al. Parameterization (f_CoprFac.m)

```
function [T11rz, T12rz, T21rz,T11dz, T12dz, ...
    T21dz]=f_CoprFac(P_ss,F,L,...
     weights)
% Youla Coprime factorization
% Works for general case of stable as well as unstable plants

[Ap, Bp, Cp, Dp] = ssdata(P_ss);

% Right coprime factorization
NumP.a=Ap-Bp*F; NumP.b=Bp; NumP.c=Cp-Dp*F; NumP.d=Dp;
NumP=ss(NumP.a,NumP.b,NumP.c,NumP.d);
DenP.a=Ap-Bp*F; DenP.b=Bp; DenP.c=-F; DenP.d=eye(size(DenP.c,1));
DenP=ss(DenP.a,DenP.b,DenP.c,DenP.d);
% Controller
NumK.a=Ap-Bp*F; NumK.b=-L; NumK.c=-F; ...
    NumK.d=zeros(size(NumK.c,1),size(NumK.b,2));
NumK=ss(NumK.a,NumK.b,NumK.c,NumK.d);
DenK.a=Ap-Bp*F; DenK.b=L; DenK.c=Cp-Dp*F; DenK.d=eye(size(DenK.c,1));
DenK=ss(DenK.a,DenK.b,DenK.c,DenK.d);
% Left coprime factorization
NumPt.a=Ap-L*Cp; NumPt.b=Bp-L*Dp; NumPt.c=Cp; NumPt.d=Dp;
NumPt=ss(NumPt.a,NumPt.b,NumPt.c,NumPt.d);
DenPt.a=Ap-L*Cp; DenPt.b=-L; DenPt.c=Cp; DenPt.d=eye(size(DenPt.c,1));
DenPt=ss(DenPt.a,DenPt.b,DenPt.c,DenPt.d);
% Controller
NumKt.a=Ap-L*Cp; NumKt.b=-L; NumKt.c=-F; ...
    NumKt.d=zeros(size(NumKt.c,1),size(NumKt.b,2));
NumKt=ss(NumKt.a,NumKt.b,NumKt.c,NumKt.d);
DenKt.a=Ap-L*Cp; DenKt.b=-(Bp-L*Dp); DenKt.c=-F; ...
    DenKt.d=eye(size(DenKt.c,1));
DenKt=ss(DenKt.a,DenKt.b,DenKt.c,DenKt.d);
```

```
% Feedback transfer function matrices
SOut11=DenK*DenPt; SOut12=-NumP; SOut21=DenPt;
KSOut11=NumK*DenPt; KSOut12=DenP; KSOut21=DenPt;
TOut11=NumP*NumKt; TOut12=NumP; TOut21=DenPt;
SensIn11=DenP*DenKt; SensIn12=-DenP; SensIn21=NumPt;
SInP11=NumP*DenKt; SInP12=-NumP; SInP21=NumPt;
% TIn11=NumK*NumPt; TIn12=DenP; TIn21=NumPt; % Final!!!
% TIn11=DenP*NumKt*inv(DenPt)*NumPt; TIn12=DenP; TIn21=NumPt;
TIn11=eye(size(DenP,1))-DenP*DenKt; TIn12=DenP; TIn21=NumPt;

W1 = weights.W1;
W2 = weights.W2;
W3 = weights.W3;
Wd1 = weights.Wd1;
Wd2 = weights.Wd2;
Wd3 = weights.Wd3;

W1c = weights.W1c;
W2c = weights.W2c;
W3c = weights.W3c;
Wd1c = weights.Wd1c;
Wd2c = weights.Wd2c;
Wd3c = weights.Wd3c;

if (isfield(weights,'Wni1'))
    Wni1 = weights.Wni1;
end
if (isfield(weights,'Wni3'))
    Wni3 = weights.Wni3;
end

% Parameterization
T11rz1=W1*SOut11; T12rz1=W1*SOut12; T21rz1=SOut21;
if isempty(W2)
    T11rz2=ss([]); T12rz2=ss([]); T21rz2=ss([]);
else
    T11rz2=W2*KSOut11; T12rz2=W2*KSOut12; T21rz2=KSOut21;
end
if isempty(W3)
    T11rz3=ss([]); T12rz3=ss([]); T21rz3=ss([]);
else
    T11rz3=W3*TOut11; T12rz3=W3*TOut12; T21rz3=TOut21;
end
if isempty(Wd1)
    T11dz1=ss([]); T12dz1=ss([]); T21dz1=ss([]);
else
    T11dz1=Wd1*SensIn11; T12dz1=Wd1*SensIn12; T21dz1=SensIn21;
end
if isempty(Wd2)
    T11dz2=ss([]); T12dz2=ss([]); T21dz2=ss([]);
else
    T11dz2=Wd2*SInP11; T12dz2=Wd2*SInP12; T21dz2=SInP21;
end
if isempty(Wd3)
    T11dz3=ss([]); T12dz3=ss([]); T21dz3=ss([]);
else
```

```
    T11dz3=Wd3*TIn11; T12dz3=Wd3*TIn12; T21dz3=TIn21;
end

% Constraint tf parameterization
T11rz1c=[]; T12rz1c=[]; T21rz1c=[];
for ii=1:length(W1c)
    T11rz1c=W1c{ii}.tfm*SOut11; T12rz1c=W1c{ii}.tfm*SOut12;
    T21rz1c=SOut21;
end

% if isempty(W2c)
%     T11rz2c=[]; T12rz2c=[]; T21rz2c=[];
% else
%     T11rz2c=W2c{1}.tfm*KSOut11;
%     T12rz2c=W2c{1}.tfm*KSOut12; T21rz2c=KSOut21;
%     T11rz2c=[T11rz2c; W2c{2}.tfm*KSOut11];
%     T12rz2c=[T12rz2c; W2c{2}.tfm*KSOut12];
% end
T11rz2c=[]; T12rz2c=[]; T21rz2c=[];
for ii=1:length(W2c)
    T11rz2c=[T11rz2c; W2c{ii}.tfm*KSOut11];
    T12rz2c=[T12rz2c; W2c{ii}.tfm*KSOut12];
end

T11rz3c=[]; T12rz3c=[]; T21rz3c=[];
for ii=1:length(W3c)
    T11rz3c=W3c{ii}.tfm*TOut11;
    T12rz3c=W3c{ii}.tfm*TOut12;
    T21rz3c=TOut21;
end

T11dz1c=[]; T12dz1c=[]; T21dz1c=[];
for ii=1:length(Wd1c)
    T11dz1c=Wd1c{ii}.tfm*SensIn11;
    T12dz1c=Wd1c{ii}.tfm*SensIn12;
    T21dz1c=SensIn21;
end

T11dz2c=[]; T12dz2c=[]; T21dz2c=[];
for ii=1:length(Wd2c)
    T11dz2c=Wd2c{ii}.tfm*SInP11;
    T12dz2c=Wd2c{ii}.tfm*SInP12;
    T21dz2c=SInP21;
end

T11dz3c=[]; T12dz3c=[]; T21dz3c=[];
for ii=1:length(Wd3c)
    T11dz3c=Wd3c{ii}.tfm*TIn11;
    T12dz3c=Wd3c{ii}.tfm*TIn12;
    T21dz3c=TIn21;
end

% For Trz1 and Tdiz2
T11rz=[T11rz1; T11rz2; T11rz3; T11rz1c; T11rz2c; T11rz3c];
T12rz=[T12rz1; T12rz2; T12rz3; T12rz1c; T12rz2c; T12rz3c];
T21rz=T21rz1;
T11dz=[T11dz1; T11dz2; T11dz3; T11dz1c; T11dz2c; T11dz3c];
```

```
T12dz=[T12dz1; T12dz2; T12dz3; T12dz1c; T12dz2c; T12dz3c];
T21dz=T21dz1;


function [T11rz, T12rz, T21rz, T11dz, T12dz, ...
    T21dz]=f_CoprFac_hsvio(P_ss,F,L,weights) % W1, W2, W3, Wd1, Wd2, ...
    Wd3, W1c, W2c, W3c, Wd1c, Wd2c, Wd3c)
% Youla Coprime factorization
% For HSV inner-outer with two loop-breaking points

W1 = weights.W1;
W2 = weights.W2;
W3 = weights.W3;
Wd1 = weights.Wd1;
Wd2 = weights.Wd2;
Wd3 = weights.Wd3;

W1c = weights.W1c;
W2c = weights.W2c;
W3c = weights.W3c;
Wd1c = weights.Wd1c;
Wd2c = weights.Wd2c;
Wd3c = weights.Wd3c;

[Ap, Bp, Cp, Dp] = ssdata(P_ss);

% Right coprime factorization
NumP.a=Ap-Bp*F; NumP.b=Bp; NumP.c=Cp-Dp*F; NumP.d=Dp; ...
    NumP=ss(NumP.a,NumP.b,NumP.c,NumP.d);
DenP.a=Ap-Bp*F; DenP.b=Bp; DenP.c=-F; DenP.d=eye(size(DenP.c,1)); ...
    DenP=ss(DenP.a,DenP.b,DenP.c,DenP.d);
% Controller
NumK.a=Ap-Bp*F; NumK.b=-L; NumK.c=-F; ...
    NumK.d=zeros(size(NumK.c,1),size(NumK.b,2)); ...
    NumK=ss(NumK.a,NumK.b,NumK.c,NumK.d);
DenK.a=Ap-Bp*F; DenK.b=L; DenK.c=Cp-Dp*F; ...
    DenK.d=eye(size(DenK.c,1)); DenK=ss(DenK.a,DenK.b,DenK.c,DenK.d);
% Left coprime factorization
NumPt.a=Ap-L*Cp; NumPt.b=Bp-L*Dp; NumPt.c=Cp; NumPt.d=Dp; ...
    NumPt=ss(NumPt.a,NumPt.b,NumPt.c,NumPt.d);
DenPt.a=Ap-L*Cp; DenPt.b=-L; DenPt.c=Cp; ...
    DenPt.d=eye(size(DenPt.c,1)); ...
    DenPt=ss(DenPt.a,DenPt.b,DenPt.c,DenPt.d);
% Controller
NumKt.a=Ap-L*Cp; NumKt.b=-L; NumKt.c=-F; ...
    NumKt.d=zeros(size(NumKt.c,1),size(NumKt.b,2)); ...
    NumKt=ss(NumKt.a,NumKt.b,NumKt.c,NumKt.d);
DenKt.a=Ap-L*Cp; DenKt.b=-(Bp-L*Dp); DenKt.c=-F; ...
    DenKt.d=eye(size(DenKt.c,1)); ...
    DenKt=ss(DenKt.a,DenKt.b,DenKt.c,DenKt.d);

% Feedback transfer function matrices
SOut11=DenK*DenPt; SOut12=-NumP; SOut21=DenPt;
KSOut11=NumK*DenPt; KSOut12=DenP; KSOut21=DenPt;
TOut11=NumP*NumKt; TOut12=NumP; TOut21=DenPt;
SensIn11=DenP*DenKt; SensIn12=-DenP; SensIn21=NumPt;
SInP11=NumP*DenKt; SInP12=-NumP; SInP21=NumPt;
```

```
% TIn11=NumK*NumPt; TIn12=DenP; TIn21=NumPt; % Final!!!
% TIn11=DenP*NumKt*inv(DenPt)*NumPt; TIn12=DenP; TIn21=NumPt;
TIn11=eye(size(DenP,1))-DenP*DenKt; TIn12=DenP; TIn21=NumPt;

%%%%%%%%%%%%%%%%%%%%%%%
% Inner-Outer loop - Select required tf's
SOut11=SOut11(1:2,1:2); SOut12=SOut12(1:2,1:2); SOut21=SOut21(:,1:2);
KSOut11=KSOut11(1:2,1:2); KSOut12=KSOut12(1:2,1:2); ...
    KSOut21=KSOut21(:,1:2);
TOut11=TOut11(1:2,1:2); TOut12=TOut12(1:2,1:2); TOut21=TOut21(:,1:2);
SInP11=SInP11(1:2,1:2); SInP12=SInP12(1:2,1:2); SInP21=SInP21(:,1:2);
% SensIn21=NumPt(1:2,:);
% TIn21=NumPt(1:2,:);
%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Added for Integ Augment fix
s=tf('s');
T11rz2=series(W2*KSOut11,ss(1/s)); ...
    T12rz2=series(W2*KSOut12,ss(1/s)); T21rz2=KSOut21;
T11dz2=series(Wd2*SInP11,ss(s)); T12dz2=series(Wd2*SInP12,ss(s)); ...
    T21dz2=SInP21;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Parameterization

T11rz1=W1*SOut11; T12rz1=W1*SOut12; T21rz1=SOut21;
T11rz2=W2*KSOut11; T12rz2=W2*KSOut12; T21rz2=KSOut21;
T11rz3=W3*TOut11; T12rz3=W3*TOut12; T21rz3=TOut21;
T11dz1=Wd1*SensIn11; T12dz1=Wd1*SensIn12; T21dz1=SensIn21;
T11dz2=Wd2*SInP11; T12dz2=Wd2*SInP12; T21dz2=SInP21;
T11dz3=Wd3*TIn11; T12dz3=Wd3*TIn12; T21dz3=TIn21;

% Constraint tf parameterization
T11rz1c=[]; T12rz1c=[]; T21rz1c=[];
for ii=1:length(W1c)
    T11rz1c=W1c{ii}.tfm*SOut11; T12rz1c=W1c{ii}.tfm*SOut12; ...
        T21rz1c=SOut21;
end
% if isempty(W2c)
%     T11rz2c=[]; T12rz2c=[]; T21rz2c=[];
% else
%     T11rz2c=W2c{1}.tfm*KSOut11; T12rz2c=W2c{1}.tfm*KSOut12; ...
    T21rz2c=KSOut21;
%     T11rz2c=[T11rz2c; W2c{2}.tfm*KSOut11]; T12rz2c=[T12rz2c; ...
    W2c{2}.tfm*KSOut12];
% end
T11rz2c=[]; T12rz2c=[]; T21rz2c=[];
for ii=1:length(W2c)
    T11rz2c=[T11rz2c; W2c{ii}.tfm*KSOut11]; T12rz2c=[T12rz2c; ...
        W2c{ii}.tfm*KSOut12];
end
T11rz3c=[]; T12rz3c=[]; T21rz3c=[];
for ii=1:length(W3c)
    T11rz3c=W3c{ii}.tfm*TOut11; T12rz3c=W3c{ii}.tfm*TOut12; ...
        T21rz3c=TOut21;
end
```

```
T11dz1c=[]; T12dz1c=[]; T21dz1c=[];
for ii=1:length(Wd1c)
    T11dz1c=Wd1c{ii}.tfm*SensIn11; T12dz1c=Wd1c{ii}.tfm*SensIn12; ...
       T21dz1c=SensIn21;
end
T11dz2c=[]; T12dz2c=[]; T21dz2c=[];
for ii=1:length(Wd2c)
    T11dz2c=Wd2c{ii}.tfm*SInP11; T12dz2c=Wd2c{ii}.tfm*SInP12; ...
       T21dz2c=SInP21;
end
T11dz3c=[]; T12dz3c=[]; T21dz3c=[];
for ii=1:length(Wd3c)
    T11dz3c=Wd3c{ii}.tfm*TIn11; T12dz3c=Wd3c{ii}.tfm*TIn12; ...
       T21dz3c=TIn21;
end
% For Trz1 and Tdiz2
T11rz=[T11rz1; T11rz2; T11rz3; T11rz1c; T11rz2c; T11rz3c]; ...
   T12rz=[T12rz1; T12rz2; T12rz3; T12rz1c; T12rz2c; T12rz3c]; ...
   T21rz=T21rz1;
T11dz=[T11dz1; T11dz2; T11dz3; T11dz1c; T11dz2c; T11dz3c]; ...
   T12dz=[T12dz1; T12dz2; T12dz3; T12dz1c; T12dz2c; T12dz3c]; ...
   T21dz=T21dz1;


function [T11rz, T12rz, T21rz, T11dz, T12dz, ...
   T21dz,T11niz,T12niz,T21niz]=f_CoprFac_hsvio_Tniu(P_ss,F,L,weights)
% Youla Coprime factorization
% For HSV inner-outer with three loop-breaking points

W1 = weights.W1;
W2 = weights.W2;
W3 = weights.W3;
Wd1 = weights.Wd1;
Wd2 = weights.Wd2;
Wd3 = weights.Wd3;
Wni1 = weights.Wni1;
Wni3 = weights.Wni3;

W1c = weights.W1c;
W2c = weights.W2c;
W3c = weights.W3c;
Wd1c = weights.Wd1c;
Wd2c = weights.Wd2c;
Wd3c = weights.Wd3c;

[Ap, Bp, Cp, Dp] = ssdata(P_ss);

% Right coprime factorization
NumP.a=Ap-Bp*F; NumP.b=Bp; NumP.c=Cp-Dp*F; NumP.d=Dp; ...
   NumP=ss(NumP.a,NumP.b,NumP.c,NumP.d);
DenP.a=Ap-Bp*F; DenP.b=Bp; DenP.c=-F; DenP.d=eye(size(DenP.c,1)); ...
   DenP=ss(DenP.a,DenP.b,DenP.c,DenP.d);
% Controller
NumK.a=Ap-Bp*F; NumK.b=-L; NumK.c=-F; ...
   NumK.d=zeros(size(NumK.c,1),size(NumK.b,2)); ...
   NumK=ss(NumK.a,NumK.b,NumK.c,NumK.d);
DenK.a=Ap-Bp*F; DenK.b=L; DenK.c=Cp-Dp*F; ...
```

```
    DenK.d=eye(size(DenK.c,1)); DenK=ss(DenK.a,DenK.b,DenK.c,DenK.d);
% Left coprime factorization
NumPt.a=Ap-L*Cp; NumPt.b=Bp-L*Dp; NumPt.c=Cp; NumPt.d=Dp; ...
    NumPt=ss(NumPt.a,NumPt.b,NumPt.c,NumPt.d);
DenPt.a=Ap-L*Cp; DenPt.b=-L; DenPt.c=Cp; ...
    DenPt.d=eye(size(DenPt.c,1)); ...
    DenPt=ss(DenPt.a,DenPt.b,DenPt.c,DenPt.d);
% Controller
NumKt.a=Ap-L*Cp; NumKt.b=-L; NumKt.c=-F; ...
    NumKt.d=zeros(size(NumKt.c,1),size(NumKt.b,2)); ...
    NumKt=ss(NumKt.a,NumKt.b,NumKt.c,NumKt.d);
DenKt.a=Ap-L*Cp; DenKt.b=-(Bp-L*Dp); DenKt.c=-F; ...
    DenKt.d=eye(size(DenKt.c,1)); ...
    DenKt=ss(DenKt.a,DenKt.b,DenKt.c,DenKt.d);

% Feedback transfer function matrices
SOut11=DenK*DenPt; SOut12=-NumP; SOut21=DenPt;
KSOut11=NumK*DenPt; KSOut12=DenP; KSOut21=DenPt;
TOut11=NumP*NumKt; TOut12=NumP; TOut21=DenPt;
SensIn11=DenP*DenKt; SensIn12=-DenP; SensIn21=NumPt;
SInP11=NumP*DenKt; SInP12=-NumP; SInP21=NumPt;
% TIn11=NumK*NumPt; TIn12=DenP; TIn21=NumPt; % Final!!!
% TIn11=DenP*NumKt*inv(DenPt)*NumPt; TIn12=DenP; TIn21=NumPt;
TIn11=eye(size(DenP,1))-DenP*DenKt; TIn12=DenP; TIn21=NumPt;

% Tniu11=NumK*DenPt; Tniu12=DenP; Tniu21=DenPt; % map from sensor ...
    noise in the inner-loop to control signal
% Tniy11=NumK*DenPt; Tniy12=DenP; Tniy21=DenPt; % map from sensor ...
    noise in the inner-loop to output
% Tniei11=NumK*DenPt; Tniei12=DenP; Tniei21=DenPt; % map from sensor ...
    noise in the inner-loop to output

%%%%%%%%%%%%%%%%%%%%%%%
% Inner-Outer loop - Select required tf's
Tniu11=KSOut11(1:2,3); Tniu12=KSOut12(1:2,:); Tniu21=KSOut21(:,3);
% Tniy11=TOut11(1:2,3); Tniy12=TOut12(1:2,3); Tniy21=TOut21(:,3);
% Tnixp11=TOut11(1:2,3); Tnixp12=TOut12(1:2,3); Tnixp21=TOut21(:,3);
Tniei11=SOut11(3,3); Tniei12=SOut12(3,:); Tniei21=SOut21(:,3);

SOut11=SOut11(1:2,1:2); SOut12=SOut12(1:2,1:2); SOut21=SOut21(:,1:2);
KSOut11=KSOut11(1:2,1:2); KSOut12=KSOut12(1:2,1:2); ...
    KSOut21=KSOut21(:,1:2);
TOut11=TOut11(1:2,1:2); TOut12=TOut12(1:2,1:2); TOut21=TOut21(:,1:2);
SInP11=SInP11(1:2,1:2); SInP12=SInP12(1:2,1:2); SInP21=SInP21(:,1:2);
% SensIn21=NumPt(1:2,:);
% TIn21=NumPt(1:2,:);
%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Added for Integ Augment fix
s=tf('s');
% T11rz1=series(W1*SOut11,ss(s+1e-6)); ...
    T12rz1=series(W1*SOut12,ss(s+1e-6)); T21rz1=SOut21;
T11rz2=series(W2*KSOut11,ss(1/(s+1e-1))); ...
    T12rz2=series(W2*KSOut12,ss(1/(s+1e-1))); T21rz2=KSOut21;
% % The eps in the integrator (1/(s+eps)) is picked to relatively ...
    high value
```

```
% % ~0.1 because of a bad effect caused by bilinear transformation. When
% % bilinear transformation is done, the weighted KS or
% % (T11rz2+T12rz2**T21rz2) was going to a high value at low frequencies
% % (near DC), even though the acutal weighted KS (i.e., without ...
   bilin) was
% % low at those frequencies.
% T11dz1=series(Wd1*SensIn11,ss(s+1e-6)); ...
   T12dz1=series(Wd1*SensIn12,ss(s+1e-6)); T21dz1=SensIn21;
T11dz2=series(Wd2*SInP11,ss(s+1e-6)); ...
   T12dz2=series(Wd2*SInP12,ss(s+1e-6)); T21dz2=SInP21;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Parameterization

T11rz1=W1*SOut11; T12rz1=W1*SOut12; T21rz1=SOut21;
% T11rz2=W2*KSOut11; T12rz2=W2*KSOut12; T21rz2=KSOut21;
T11rz3=W3*TOut11; T12rz3=W3*TOut12; T21rz3=TOut21;
T11dz1=Wd1*SensIn11; T12dz1=Wd1*SensIn12; T21dz1=SensIn21;
% T11dz2=Wd2*SInP11; T12dz2=Wd2*SInP12; T21dz2=SInP21;
T11dz3=Wd3*TIn11; T12dz3=Wd3*TIn12; T21dz3=TIn21;

T11niz1=Wni1*Tniu11; T12niz1=Wni1*Tniu12; T21niz1=Tniu21;
% T11niz2=Wni2*Tnixp11; T12niz2=Wni1*Tnixp12; T21niz2=Tnixp21;
T11niz3=Wni3*Tniei11; T12niz3=Wni3*Tniei12; T21niz3=Tniei21;

% Constraint tf parameterization
T11rz1c=[]; T12rz1c=[]; T21rz1c=[];
for ii=1:length(W1c)
    T11rz1c=W1c{ii}.tfm*SOut11; T12rz1c=W1c{ii}.tfm*SOut12; ...
       T21rz1c=SOut21;
end
% if isempty(W2c)
%     T11rz2c=[]; T12rz2c=[]; T21rz2c=[];
% else
%     T11rz2c=W2c{1}.tfm*KSOut11; T12rz2c=W2c{1}.tfm*KSOut12; ...
   T21rz2c=KSOut21;
%     T11rz2c=[T11rz2c; W2c{2}.tfm*KSOut11]; T12rz2c=[T12rz2c; ...
   W2c{2}.tfm*KSOut12];
% end
T11rz2c=[]; T12rz2c=[]; T21rz2c=[];
for ii=1:length(W2c)
    T11rz2c=[T11rz2c; W2c{ii}.tfm*KSOut11]; T12rz2c=[T12rz2c; ...
       W2c{ii}.tfm*KSOut12];
end
T11rz3c=[]; T12rz3c=[]; T21rz3c=[];
for ii=1:length(W3c)
    T11rz3c=W3c{ii}.tfm*TOut11; T12rz3c=W3c{ii}.tfm*TOut12; ...
       T21rz3c=TOut21;
end
T11dz1c=[]; T12dz1c=[]; T21dz1c=[];
for ii=1:length(Wd1c)
    T11dz1c=Wd1c{ii}.tfm*SensIn11; T12dz1c=Wd1c{ii}.tfm*SensIn12; ...
       T21dz1c=SensIn21;
end
T11dz2c=[]; T12dz2c=[]; T21dz2c=[];
for ii=1:length(Wd2c)
    T11dz2c=Wd2c{ii}.tfm*SInP11; T12dz2c=Wd2c{ii}.tfm*SInP12; ...
```

```
        T21dz2c=SInP21;
end
T11dz3c=[]; T12dz3c=[]; T21dz3c=[];
for ii=1:length(Wd3c)
    T11dz3c=Wd3c{ii}.tfm*TIn11; T12dz3c=Wd3c{ii}.tfm*TIn12; ...
        T21dz3c=TIn21;
end

% For Trz1 and Tdiz2
T11rz=[T11rz1; T11rz2; T11rz3; T11rz1c; T11rz2c; T11rz3c]; ...
   T12rz=[T12rz1; T12rz2; T12rz3; T12rz1c; T12rz2c; T12rz3c]; ...
   T21rz=T21rz1;
T11dz=[T11dz1; T11dz2; T11dz3; T11dz1c; T11dz2c; T11dz3c]; ...
   T12dz=[T12dz1; T12dz2; T12dz3; T12dz1c; T12dz2c; T12dz3c]; ...
   T21dz=T21dz1;
T11niz=[T11niz1; T11niz3]; T12niz=[T12niz1; T12niz3]; T21niz=T21niz1;


function [T11rz, T12rz, T21rz,T11dz, T12dz, ...
   T21dz]=f_CoprFac_ZamesParam(P_ss,F,L,weights)
% Zames Coprime Parameterization
% Works stable plants
% Assumes zero initial controller Ko

[Ap, Bp, Cp, Dp] = ssdata(P_ss);

[n_e, n_u] = size(P_ss);
NumP=P_ss;
DenP=ss(eye(n_u));
NumK=ss(zeros(n_u,n_e));
DenK=ss(eye(n_e));

NumPt=P_ss;
DenPt=ss(eye(n_e));
NumKt=ss(zeros(n_u,n_e));
DenKt=ss(eye(n_u));

% Feedback transfer function matrices
SOut11=DenK*DenPt; SOut12=-NumP; SOut21=DenPt;
KSOut11=NumK*DenPt; KSOut12=DenP; KSOut21=DenPt;
TOut11=NumP*NumKt; TOut12=NumP; TOut21=DenPt;
SensIn11=DenP*DenKt; SensIn12=-DenP; SensIn21=NumPt;
SInP11=NumP*DenKt; SInP12=-NumP; SInP21=NumPt;
% TIn11=NumK; TIn12=DenP; TIn21=NumPt;
TIn11=DenP*NumKt*inv(DenPt)*NumPt; TIn12=DenP; TIn21=NumPt;

% Weights
W1 = weights.W1;
W2 = weights.W2;
W3 = weights.W3;
Wd1 = weights.Wd1;
Wd2 = weights.Wd2;
Wd3 = weights.Wd3;

W1c = weights.W1c;
W2c = weights.W2c;
W3c = weights.W3c;
```

```
Wd1c = weights.Wd1c;
Wd2c = weights.Wd2c;
Wd3c = weights.Wd3c;

if (isfield(weights,'Wni1'))
    Wni1 = weights.Wni1;
end
if (isfield(weights,'Wni3'))
    Wni3 = weights.Wni3;
end

% Parameterization
T11rz1=W1*SOut11; T12rz1=W1*SOut12; T21rz1=SOut21;
T11rz2=W2*KSOut11; T12rz2=W2*KSOut12; T21rz2=KSOut21;
T11rz3=W3*TOut11; T12rz3=W3*TOut12; T21rz3=TOut21;
T11dz1=Wd1*SensIn11; T12dz1=Wd1*SensIn12; T21dz1=SensIn21;
T11dz2=Wd2*SInP11; T12dz2=Wd2*SInP12; T21dz2=SInP21;
T11dz3=Wd3*TIn11; T12dz3=Wd3*TIn12; T21dz3=TIn21;

% Constraint tf parameterization
T11rz1c=[]; T12rz1c=[]; T21rz1c=[];
for ii=1:length(W1c)
    T11rz1c=W1c{ii}.tfm*SOut11; T12rz1c=W1c{ii}.tfm*SOut12; ...
       T21rz1c=SOut21;
end
% if isempty(W2c)
%     T11rz2c=[]; T12rz2c=[]; T21rz2c=[];
% else
%     T11rz2c=W2c{1}.tfm*KSOut11; T12rz2c=W2c{1}.tfm*KSOut12; ...
   T21rz2c=KSOut21;
%     T11rz2c=[T11rz2c; W2c{2}.tfm*KSOut11]; T12rz2c=[T12rz2c; ...
   W2c{2}.tfm*KSOut12];
% end
T11rz2c=[]; T12rz2c=[]; T21rz2c=[];
for ii=1:length(W2c)
    T11rz2c=[T11rz2c; W2c{ii}.tfm*KSOut11]; T12rz2c=[T12rz2c; ...
       W2c{ii}.tfm*KSOut12];
end
T11rz3c=[]; T12rz3c=[]; T21rz3c=[];
for ii=1:length(W3c)
    T11rz3c=W3c{ii}.tfm*TOut11; T12rz3c=W3c{ii}.tfm*TOut12; ...
       T21rz3c=TOut21;
end
T11dz1c=[]; T12dz1c=[]; T21dz1c=[];
for ii=1:length(Wd1c)
    T11dz1c=Wd1c{ii}.tfm*SensIn11; T12dz1c=Wd1c{ii}.tfm*SensIn12; ...
       T21dz1c=SensIn21;
end
T11dz2c=[]; T12dz2c=[]; T21dz2c=[];
for ii=1:length(Wd2c)
    T11dz2c=Wd2c{ii}.tfm*SInP11; T12dz2c=Wd2c{ii}.tfm*SInP12; ...
       T21dz2c=SInP21;
end
T11dz3c=[]; T12dz3c=[]; T21dz3c=[];
for ii=1:length(Wd3c)
    T11dz3c=Wd3c{ii}.tfm*TIn11; T12dz3c=Wd3c{ii}.tfm*TIn12; ...
       T21dz3c=TIn21;
```

```
end

% For Trz1 and Tdiz2
T11rz=[T11rz1; T11rz2; T11rz3; T11rz1c; T11rz2c; T11rz3c]; ...
    T12rz=[T12rz1; T12rz2; T12rz3; T12rz1c; T12rz2c; T12rz3c]; ...
    T21rz=T21rz1;
T11dz=[T11dz1; T11dz2; T11dz3; T11dz1c; T11dz2c; T11dz3c]; ...
    T12dz=[T12dz1; T12dz2; T12dz3; T12dz1c; T12dz2c; T12dz3c]; ...
    T21dz=T21dz1;
```

A.1.4 Basis Selection (f_Basis.m)

```
function q = f_Basis(N, p, z, basis_type)
% Form the basis TFs for given basis parameters and type

q{1} = zpk([],[],1);
% q{1} = tf(1,1);
if basis_type == 1  % fixed pole low pass
    for k=2:N
        q{k} = zpk([],-p,p)^(k-1);
    end
elseif basis_type == 2 % fixed pole all pass
    for k=2:N
        q{k} = zpk(p,-p,-1)^(k-1);
    end
elseif basis_type == 3 % variable pole low pass
    for k=2:N
        q{k} = zpk([],-p*(k-1),p*(k-1));
    end
elseif basis_type == 4 % variable pole all pass
    for k=2:N
        q{k} = zpk(p*(k-1),-p*(k-1),-1);
    end
elseif basis_type == 5 % pole and zero
    for k=2:N
        q{k} = zpk(z,-p,-1)^(k-1);
    end
elseif basis_type == 5 % Laguerre
    for k=2:N
        q{k} = zpk([],-p,sqrt(2*p))*zpk(p,-p,1)^(k-1);
    end
end
```

A.1.5 Form Finite-Dimensional Q Parameter (f_FormQN.m)

```
function QN = f_FormQN(x, qk, n_u, n_e, N)
% From Q_N at given point x using the bases qk

xtemp = reshape(x,n_u*n_e,N);
QN = zeros(n_u, n_e);
for i = 1:N
    X{i} = reshape(xtemp(:,i),n_u,n_e);

    % % Find temp = QN + X{i} * qk{i}
```

```
    % % Straight forward way.
    % temp = QN + X{i} * qk{i};
    % Alternative way. minreal in later works better when this is used,
    % i.e., order of QN found will be as expected.
    QNss = ss(QN);
    xqss = ss(series(qk{i},X{i}));
    A = blkdiag(QNss.a,xqss.a);
    B = [QNss.b;xqss.b];
    C = [QNss.c,xqss.c];
    D = QNss.d+xqss.d;
    temp = zpk(ss(A,B,C,D));

    QN = minreal(temp,1e-6);

end
QN = ss(QN);
```

A.1.6 Extract Data From Problem Setup (f_GenData.m)

```
function [n_e, n_u, DATArz,DATAdz] = f_GenData(P, weights)
% Extract data from problem setup for the GMS methodology

W1 = weights.W1;
W2 = weights.W2;
W3 = weights.W3;
Wd1 = weights.Wd1;
Wd2 = weights.Wd2;
Wd3 = weights.Wd3;

W1c = weights.W1c;
W2c = weights.W2c;
W3c = weights.W3c;
Wd1c = weights.Wd1c;
Wd2c = weights.Wd2c;
Wd3c = weights.Wd3c;

if (isfield(weights,'Wni1'))
    Wni1 = weights.Wni1;
end
if (isfield(weights,'Wni3'))
    Wni3 = weights.Wni3;
end

[n_e, n_u, n_s] = size(P);

nObj = 0;
%% Check W1
if ~isempty(W1)
    [noutput, ninput, nstate] = size(W1);
    if noutput ~= ninput
        disp('Error: W1 is not square')
        return
    end
    if noutput ~= n_e
        disp('Error: Dimansion mismatch in W1')
```

```
        return
    end
    nObj = nObj+n_e;
end

%% Check W2
if ~isempty(W2)
    [noutput, ninput, nstate] = size(W2);
    if noutput ~= ninput
        disp('Error: W2 is not square')
        return
    end
    if noutput ~= n_u
        disp('Error: Dimansion mismatch in W2')
        return
    end
    nObj = nObj+n_u;
end

%% Check W3
if ~isempty(W3)
    [noutput, ninput, nstate] = size(W3);
    if noutput ~= ninput
        disp('Error: W3 is not square')
        return
    end
    if noutput ~= n_e
        disp('Error: Dimansion mismatch in W3')
        return
    end
    nObj = nObj+n_e;
end

DATArz.ObjVec = 1:nObj;
TotalRows = nObj;
%% rz
ConstraintCounter = 0;
[nRow nCol]=size(W1c);
for i=1:nCol
    W1 = W1c{i}.tfm;
    if ~isempty(W1)
        [noutput, ninput, nstate] = size(W1);
        if noutput ~= ninput
            disp(['Error: W1c{' num2str(i) '} is not square'])
            return
        end
        if noutput ~= n_e
            disp(['Error: Dimansion mismatch in W1c{' num2str(i) '}'])
             return
        end
        ConstraintCounter = ConstraintCounter + 1;
        DATArz.ConVec{ConstraintCounter} = TotalRows+1:TotalRows+n_e;
        DATArz.ConNam{ConstraintCounter} = W1c{i}.Fun;
        DATArz.ConVal{ConstraintCounter} = W1c{i}.Val;
        TotalRows = TotalRows + n_e;
    end
end
```

```
%%
[nRow nCol]=size(W2c);
for i=1:nCol
    W2 = W2c{i}.tfm;
    if ~isempty(W2)
        [noutput, ninput, nstate] = size(W2);
        if noutput ~= ninput
            disp(['Error: W2c{' num2str(i) '} is not square'])
            return
        end
        if noutput ~= n_u
            disp(['Error: Dimansion mismatch in W2c{' num2str(i) '}'])
             return
        end
        ConstraintCounter = ConstraintCounter + 1;
        DATArz.ConVec{ConstraintCounter} = TotalRows+1:TotalRows+n_u;
        DATArz.ConNam{ConstraintCounter} = W2c{i}.Fun;
        DATArz.ConVal{ConstraintCounter} = W2c{i}.Val;
        TotalRows = TotalRows + n_u;
    end
end

%%
[nRow nCol]=size(W3c);
for i=1:nCol
    W3 = W3c{i}.tfm;
    if ~isempty(W3)
        [noutput, ninput, nstate] = size(W3);
        if noutput ~= ninput
            disp(['Error: W3c{' num2str(i) '} is not square'])
            return
        end
        if noutput ~= n_e
            disp(['Error: Dimansion mismatch in W3c{' num2str(i) '}'])
             return
        end
        ConstraintCounter = ConstraintCounter + 1;
        DATArz.ConVec{ConstraintCounter} = TotalRows+1:TotalRows+n_e;
        DATArz.ConNam{ConstraintCounter} = W3c{i}.Fun;
        DATArz.ConVal{ConstraintCounter} = W3c{i}.Val;
        TotalRows = TotalRows + n_e;
    end
end
DATArz.ConNum = ConstraintCounter;


%% dz
nObj = 0;
%% Check Wd1
if ~isempty(Wd1)
    [noutput, ninput, nstate] = size(Wd1);
    if noutput ~= ninput
        disp('Error: Wd1 is not square')
        return
    end
    if noutput ~= n_u
```

```
        disp('Error: Dimansion mismatch in Wd1')
        return
    end
    nObj = nObj+n_u;
end

%% Check Wd2
if ~isempty(Wd2)
    [noutput, ninput, nstate] = size(Wd2);
    if noutput ~= ninput
        disp('Error: Wd2 is not square')
        return
    end
    if noutput ~= n_e
        disp('Error: Dimansion mismatch in Wd2')
        return
    end
    nObj = nObj+n_e;
end

%% Check Wd3
if ~isempty(Wd3)
    [noutput, ninput, nstate] = size(Wd3);
    if noutput ~= ninput
        disp('Error: Wd3 is not square')
        return
    end
    if noutput ~= n_u
        disp('Error: Dimansion mismatch in Wd3')
        return
    end
    nObj = nObj+n_u;
end

DATAdz.ObjVec = 1:nObj;
TotalRows = nObj;
%%
ConstraintCounter = 0;

[nRow nCol]=size(Wd1c);
for i=1:nCol
    Wd1 = Wd1c{i}.tfm;
    if ~isempty(Wd1)
        [noutput, ninput, nstate] = size(Wd1);
        if noutput ~= ninput
            disp(['Error: Wd1c{' num2str(i) '} is not square'])
            return
        end
        if noutput ~= n_e
            disp(['Error: Dimansion mismatch in Wd1c{' num2str(i) '}'])
             return
        end
        ConstraintCounter = ConstraintCounter + 1;
        DATAdz.ConVec{ConstraintCounter} = TotalRows+1:TotalRows+n_e;
        DATAdz.ConNam{ConstraintCounter} = Wd1c{i}.Fun;
        DATAdz.ConVal{ConstraintCounter} = Wd1c{i}.Val;
        TotalRows = TotalRows + n_e;
```

```
    end
end

%%
[nRow nCol]=size(Wd2c);
for i=1:nCol
    Wd2 = Wd2c{i}.tfm;
    if ~isempty(Wd2)
        [noutput, ninput, nstate] = size(Wd2);
        if noutput ~= ninput
            disp(['Error: Wd2c{' num2str(i) '} is not square'])
            return
        end
        if noutput ~= n_u
            disp(['Error: Dimansion mismatch in Wd2c{' num2str(i) '}'])
             return
        end
        ConstraintCounter = ConstraintCounter + 1;
        DATAdz.ConVec{ConstraintCounter} = TotalRows+1:TotalRows+n_u;
        DATAdz.ConNam{ConstraintCounter} = Wd2c{i}.Fun;
        DATAdz.ConVal{ConstraintCounter} = Wd2c{i}.Val;
        TotalRows = TotalRows + n_u;
    end
end

%%
[nRow nCol]=size(Wd3c);
for i=1:nCol
    Wd3 = Wd3c{i}.tfm;
    if ~isempty(Wd3)
        [noutput, ninput, nstate] = size(Wd3);
        if noutput ~= ninput
            disp(['Error: Wd3c{' num2str(i) '} is not square'])
            return
        end
        if noutput ~= n_e
            disp(['Error: Dimansion mismatch in Wd3c{' num2str(i) '}'])
             return
        end
        ConstraintCounter = ConstraintCounter + 1;
        DATAdz.ConVec{ConstraintCounter} = TotalRows+1:TotalRows+n_e;
        DATAdz.ConNam{ConstraintCounter} = Wd3c{i}.Fun;
        DATAdz.ConVal{ConstraintCounter} = Wd3c{i}.Val;
        TotalRows = TotalRows + n_e;
    end
end


DATAdz.ConNum = ConstraintCounter;
```

A.1.7 Vectorize the Problem (f_Vectorize.m)

```
function [M Mobj Mcon] = f_Vectorize(T11, T12, T21, qk, N, n_u, n_e, ...
   ProblemData)
% Vectorize Problem
```

```
% Forms M_{l} = M_{k}^{ij}
% l = (k-1)*nu*ne+(j-1)*nu+i;
% M_{k}^{ij} = T_{12}*B^{ij}*T_{21}*q_k
% T_wz = M_o + sum_{l=1}^{nu*ne*N} M_l x_l
Mobj = {};
Mcon = {};
Bij = zeros(n_u,n_e);
for k = 1:N
    for j = 1:n_e
        for i = 1:n_u
            l = (k-1)*n_u*n_e+(j-1)*n_u+i;
            Bij = zeros(n_u,n_e);
            Bij(i,j) = 1;
            [size_t21 temp] = size(T21.a);
            [temp size_t12] = size(T12.a);
            if isempty(T12)
                M{l} = ss([]);
            else
                a = [T21.a zeros(size_t21,size_t12);
                    T12.b*Bij*T21.c T12.a];
                b = [T21.b; T12.b*Bij*T21.d];
                c = [T12.d*Bij*T21.c T12.c];
                d = T12.d*Bij*T21.d;
                M{l} = ss(a,b,c,d)*qk{k};
            end
        end
    end
end
for k = 1:N*n_e*n_u
    Mobj{k} = M{k}(ProblemData.ObjVec,:);
end
for i = 1:ProblemData.ConNum
    for k = 1:N*n_e*n_u
        Mcon{i,k} = M{k}(ProblemData.ConVec{i},:);
    end
end
```

A.1.8 $\mathcal{H}^\infty$-Norm Value and Subgradient (f_Hinf.m)

```
function [value sg varargout] = f_Hinf(M, x, T11, T12, T21, Q, vec, ...
   varargin)
% Compute H-infinity norm and subgradients
% of Parameterized TFMs for given Youla et al. parameter Q

if nargin == 8
 conval = varargin{1};
 varargout{1} = conval;
end
n = length(x);

[n_u, n_e, n_s] = size(Q);
if isempty(T11)
    Twz = ss([]);
else
    Twz = parallel(T11,series(series(T21,Q),T12));
```

```
end
%Twz = minreal(Twz);
Twz = Twz(vec,:);

% [ninf, fpeak] = norm(Twz, inf, 1e-8);
[ninf, fpeak] = hinfnorm(Twz, 1e-8);
value = ninf;

if fpeak < 1e-5
    fpeak = 1e-5;
end

if fpeak<1e-5
    wmin=1e-8; wmax=1e-2;
elseif fpeak<1e-2
    wmin=1e-4; wmax=1e0;
elseif fpeak<1e0
    wmin=1e-1; wmax=1e1;
elseif fpeak<10
    wmin=1e-1; wmax=1e2;
elseif fpeak<1e2
    wmin=1e0; wmax=1e4;
elseif fpeak<1e5
    wmin=1e3; wmax=1e7;
elseif fpeak>=1e5
    wmin=1e4; wmax=1e10;
else
    wmin=max([1, fpeak-10]); wmax=fpeak+10;
end
TwzScaled=prescale(Twz,{wmin,wmax});

Hjwo = freqresp(TwzScaled,fpeak);
% Hjwo = evalfr(TwzScaled,fpeak);

[U,S,V] = svd(Hjwo);    % SVD at W0
if isempty(U)
    uo = [];
    vo = [];
else
    uo      = U(:,1);       % Maximum Left Singular Vector
    vo      = V(:,1);       % Maximum Right Singular Vector
end
subgradient = [];
for i = 1:n
    Hjwo = freqresp(M{i},fpeak);
    magHjwo = abs(Hjwo);
    subgradient = [subgradient; real(uo'*Hjwo*vo)];
end
sg = subgradient;
```

A.1.9 $\mathcal{L}^\infty$-Norm Value and Subgradient (f_Linf.m)

```
function [value,sg,ConValVec,varargout] = f_Linf(M,x,T11,T12,...
    T21, Q, vec, varargin)
% Compute peak value and subgradients
```

```
% of Parameterized TFMs for given Youla et al. parameter Q

tvec = 0:0.001:10;
if nargin == 8
    conval = varargin{1};
end
n = length(x);

Twz = parallel(T11,series(series(T21,Q),T12));
Twz = Twz(vec,:);
% [n_output,n_input] = size(Twz); % n_row = n_output

% subgradient = NaN*zeros(n,length(conval));
Counter=0;
for ii = 1:size(conval,1)
    for jj = 1:size(conval,2)
        %         kk=(ii-1)*size(conval,2)+jj;
        if conval(ii,jj)==Inf
            disp('');

        else
            Counter=Counter+1;
            ConValVec(Counter)=conval(ii,jj);

            [y,tvec] = step(Twz(ii,jj), tvec);

            [ypeak,I] = max(y);
            tpeak = tvec(I);
            value(Counter,1) = ypeak;

            for i = 1:n
                %             if n_output == n_input
                %                 [y,tvec] = step(M{i}(ii,jj), tvec);
                %             else
                %                 [y,tvec] = step(M{i}(ii,1), tvec);
                %             end
                [y,tvec] = step(M{i}(ii,jj), tvec);
                subgradient(i,Counter) = y(I);
            end


            if nargin == 8
                varargout{1} = conval(ii);
            end
            %     if value > conval(ii) % See why this is required
            %         return
            %     end
        end
    end
end
sg = subgradient;
```

A.1.10 Form K(Q) (f_FormK.m)

```
function K=f_FormK(P_ss,Q,F,L)
```

```
% Form the controller from Q
% Uses Youla parameterization

Ap=P_ss.a; Bp=P_ss.b; Cp=P_ss.c; Dp=P_ss.d;
Aq = Q.a; Bq = Q.b; Cq = Q.c; Dq = Q.d; % Q - Parameter
n_x=size(Ap,1); n_e=size(Cp,1); n_u=size(Bp,2);

Delta = eye(n_u) - Dq*Dp;
invDelta = inv(Delta);
Ak11 = (Ap-L*Cp)-(Bp-L*Dp)*invDelta*(-Dq*Cp+F);
Ak12 = -(Bp-L*Dp)*invDelta*Cq;
Ak21 = -Bq*Cp+Bq*Dp*invDelta*(-Dq*Cp+F);
Ak22 = Aq+Bq*Dp*invDelta*Cq;
Ak = [Ak11 Ak12; Ak21 Ak22];
Bk = [L-(Bp+L*Dp)*invDelta*Dq;
    Bq+Bq*Dp*invDelta*Dq];
Ck = [invDelta*(-Dq*Cp+F) invDelta*Cq];
Dk = invDelta*Dq;
K = ss(Ak, Bk, Ck, Dk);
```

A.1.11 Kelley's CPM Optimizer (f_KelleyCPM_GenMix_Optimizer.m)

```
function ...
    [xk,frz,fdz]=f_KelleyCPM_GenMix_Optimizer(N,NQ,x0,Mobjrz,Mobjdz,Mconrz,Mcondz,T11rz,
    T12rz, T21rz,T11dz, T12dz, T21dz, Datarz, Datadz, ...
    Q,q,n_u,n_e,MaxIter,xmax,xmin)
% Kelley's CPM

warning off
tol_obj = 1e-4;
tol_feas = 1e-4;


% INITIALIZE
% fx = 0;               % Set output to zero
iter = 0;               % Iteration count
xk  = x0;               % Initial query point
xkStore=NaN*ones(length(xk),MaxIter);
ExitFlagStore=NaN*ones(1,MaxIter);
foStore=NaN*ones(2,MaxIter);

nConrz = Datarz.ConNum; nCondz = Datadz.ConNum; % Number of constraints
% Below matrices are used in solving the LP: min c'x   s.t. Aw<b
Ao = [];                % A matrix associated with objective function
bo = [];                % b vector associated with objective function

c = [zeros(N,1); 1];    % cvector associated with the variable x
UkminLkrz=1000; UkminLkdz=1000; constraint_flagrz=1; ...
    constraint_flagdz=1;
w=zeros(N+1,1);

options = optimset('Display','off');
% options=optimset('MaxIter',500,'TolFun',1e-9,'Display','final')

% START
```

```
while UkminLkrz > tol_obj || UkminLkdz > tol_obj || ...
    (constraint_flagrz>0) || (constraint_flagdz>0)

    Ac=[]; bc=[];
    [forz, Gfo] =feval('f_Hinf', Mobjrz, xk, T11rz, T12rz, T21rz, Q, ...
       Datarz.ObjVec);
    if UkminLkrz > tol_obj
        Ao = [Ao; Gfo' -1];
        bo = [bo; Gfo'*xk-forz];
        %     UkminLkrz3=for3-c'*w;
    end

%         Constraints rz:
    % Compute fi(x), Gfi(x) and Form Ac, bc
    frz{1}=[];
    for ii = 1:nConrz
        Mrz = Mconrz(ii,:);
        [frz{ii}, Gf{ii}, ConValVec] = ...
            feval(Datarz.ConNam{ii}, Mrz, xk, T11rz, T12rz, T21rz, ...
               Q, Datarz.ConVec{ii}, Datarz.ConVal{ii});
        frz{ii} = frz{ii} - ConValVec';
        if constraint_flagrz>0
            Ac = [Ac; Gf{ii}' zeros(size(Gf{ii}',1),1)];
            bc = [bc; Gf{ii}'*xk-frz{ii}];
        end
    end

    [fodz, Gfo] =feval('f_Hinf', Mobjdz, xk, T11dz, T12dz, T21dz, Q, ...
       Datadz.ObjVec);
    if UkminLkdz > tol_obj
        Ao = [Ao; Gfo' -1];
        bo = [bo; Gfo'*xk-fodz];
        %     UkminLkdiz3=fo3-c'*w;
    end

    %     Constraints dz:
    % Compute fi(x), Gfi(x) and Form Ac, bc
    fdz{1}=[];
    for ii = 1:nCondz
        Mdz = Mcondz(ii,:);
        [fdz{ii}, Gf{ii}, ConValVec] = ...
            feval(Datadz.ConNam{ii}, Mdz, xk, T11dz, T12dz, T21dz, ...
               Q, Datadz.ConVec{ii}, Datadz.ConVal{ii});
        fdz{ii} = fdz{ii} - ConValVec';  % In AllStep this is ...
           changed. Originally it was frz{ii} - Datarz.ConVal{ii}
        if constraint_flagdz>0
            Ac = [Ac; Gf{ii}' zeros(size(Gf{ii}',1),1)];
            bc = [bc; Gf{ii}'*xk-fdz{ii}];
        end
    end

    Ao=[Ao;Ac]; bo=[bo;bc];

    % Solve LP (used optimization toolbox function: linprog)
    [w,fval,exitflag] = linprog(c,Ao,bo,[],[],xmin,xmax,xk,options);
    % Check if problem is giving empty w. Try using other algorithms
    optionstemp=options; % temporary option
```

```
    if exitflag == -4
        optionstemp.Algorithm='dual-simplex';
        [w,fval,exitflag] = ...
           linprog(c,Ao,bo,[],[],xmin,xmax,xk,optionstemp);
    end
    if exitflag == -4
        optionstemp.Algorithm='active-set';
        [w,fval,exitflag] = ...
           linprog(c,Ao,bo,[],[],xmin,xmax,xk,optionstemp);
    end

    UkminLkrz=forz-c'*w; fprintf('\n%d %1.6f  %1.6f  ', iter,forz, ...
       UkminLkrz);

    UkminLkdz=fodz-c'*w; fprintf('%1.6f  %1.6f  ', fodz, UkminLkdz);

    foStore(:,iter+1)=[forz; fodz];

    % Update xk
    xk = w(1:N); xkStore(:,iter+1)=xk;ExitFlagStore(1,iter+1)=exitflag;

    % Increment iter
    iter = iter + 1;
    % Check if fi(xk) < epsilon for all i
    constraint_flagrz = 0;
    for ii = 1:nConrz
        if frz{ii} > tol_feas
            constraint_flagrz = 1;
        end
    end
    constraint_flagdz = 0;
    for ii = 1:nCondz
        if fdz{ii} > tol_feas
            constraint_flagdz = 1;
        end
    end

    if iter == MaxIter
        fprintf('\n');
        fprintf('Max Num of Iter exceeded \n')
        break;
    end
    Q = f_FormQN(xk, q, n_u, n_e, NQ);
end
```

A.1.12 ACCPM Optimizer (f_ACCPM_GenMixSens_Optimizer.m)

```
function [x, fx, iter_cnt, perf_meas] = ...
    f_ACCPM_GenMixSens_Optimizer(N,NQ,x0,Mobjrz,Mobjdz,Mconrz,Mcondz,T11rz, ...
    T12rz, T21rz,T11dz, T12dz, T21dz, Datarz, Datadz, ...
    Q,q,n_u,n_e,xmax,xmin,MaxIter)
% ACCPM main file

% Inputs
% xk is initial x
```

```
%% User initialisation of the ACCPM parameters
% x0 = x01*ones(N,1);
[problemS,methodS] = UserInitACCPM(N, x0,xmax,xmin,MaxIter);

%% Call the initialization routine
[accpm, accpm2Oracle] = C_InitProxAccpm(problemS, methodS);
clear problemS methodS;

%% Optimization process
Flag = 1;
% Store the num of iterations and objective func value at each iter
iter_cnt = 0;
perf_meas = NaN*ones(MaxIter,1);
while(Flag)
  % Function evaluate at current point (y) using the Oracle
  [oracleS] = UserOracle(accpm2Oracle, Mobjrz,Mobjdz, Mconrz,Mcondz, ...
     T11rz, T12rz, T21rz,T11dz, T12dz, T21dz, Q, Datarz,Datadz);
  % Call the query point generator to get the next point
  [accpm, accpm2Oracle] = ...
  C_ProxAccpmGen(oracleS, accpm, accpm2Oracle);
  x = get(accpm2Oracle,'y');
  Q = f_FormQN(x, q, n_u, n_e, NQ);
  clear oracleS;
  % Possibly artificial stop
  condition = (get(accpm2Oracle,'ExitCode') ~= 0);
  if (condition)
    Flag = 0;
  end
  iter_cnt = iter_cnt+1;
  perf_meas(iter_cnt) = accpm.ParamS.OptTypeFact * ...
     accpm.ManagerS.D.ObjBounds(2);
end
% Minimizer and Objective
x  = get(accpm2Oracle,'y');
fx = accpm.ParamS.OptTypeFact * accpm.ManagerS.D.ObjBounds(2);
clear accpm2Oracle;
% Display Results
C_Display_ProxAccpmResults(accpm);
clear accpm;
return

function [problemS,methodS] = UserInitACCPM(n, x0,xmax,xmin,MaxIter)
% In this function, the user initializes the ACCPM parameters

% problemS object created
problemS = ProblemS;
problemS = set(problemS,'OptType','min');   % min or max
problemS = set(problemS,'NbVariables',n);   % Number of dual variables
problemS = set(problemS,'NbSubProblems',1); % Number of subproblems
problemS = set(problemS,'StartingPoint',x0);  % starting point
problemS = set(problemS,'VarLowerBounds',xmin); % Lower bounds on ...
   variables
problemS = set(problemS,'VarUpperBounds',xmax); % Upper bounds on ...
   variables

% methodS object created
```

```
methodS = MethodS;
methodS = set(methodS,'Proximal',0); % Use the proximal term (0/1)
methodS = set(methodS,'Rho',1);      % Value of the rho in the ...
    proximal term
methodS = set(methodS,'Verbose',3);  % Display results (0/1/2/3)
methodS = set(methodS,'MaxOuter',MaxIter); % Maximum number of ACCPM ...
    iterations
methodS = set(methodS,'MaxInner',100); % Maximum number of Newton ...
    iterations in the computation of the analytic center
methodS = set(methodS,'Tolerance',1e-4); % Relative optimality gap
methodS = set(methodS,'WeightEpigraphCutInit',10); % Initial weight ...
    on the epigraph cut
methodS = set(methodS,'WeightEpigraphCutInc',0); % Increment on the ...
    epigraph cut

return


function  [oracleS] = UserOracle(accpm2Oracle, Mobjrz,Mobjdz, ...
    Mconrz,Mcondz, T11rz, T12rz, T21rz,T11dz, T12dz, T21dz, Q, ...
    Datarz,Datadz)
% User oracle

% Current point
x = get(accpm2Oracle,'y');

oracleS = OracleS;

% Constraint
% Reference to output
for ii = 1:Datarz.ConNum
  Mrz = Mconrz(ii,:);
  [fc,Gfc,val] = ...
      feval(Datarz.ConNam{ii}, Mrz, x, T11rz, T12rz, T21rz, Q,...
                    Datarz.ConVec{ii}, Datarz.ConVal{ii});
  if (fc > val)
    oracleS = set(oracleS, 'FunctionValues', fc-val); % Value of the ...
       objective
    oracleS = set(oracleS, 'SubGradients', Gfc);  % Subgradient
    oracleS = set(oracleS, 'SubProblemIndex', 0); % Nature of the ...
       cut (Optimality -> 1, Feasibility -> 0)
    return
  end
end
% d_i to output
for ii = 1:Datadz.ConNum
  Mdz = Mcondz(ii,:);
  [fc,Gfc,val] = ...
      feval(Datadz.ConNam{ii}, Mdz, x, T11dz, T12dz, T21dz, Q,...
                    Datadz.ConVec{ii}, Datadz.ConVal{ii});
  if (fc > val)
    oracleS = set(oracleS, 'FunctionValues', fc-val);
    oracleS = set(oracleS, 'SubGradients', Gfc);
    oracleS = set(oracleS, 'SubProblemIndex', 0);
    return
  end
end
```

```
% Optimality
[forz,Gforz] = f_Hinf(Mobjrz, x, T11rz, T12rz, T21rz, Q, Datarz.ObjVec);
[fodz,Gfodz] = f_Hinf(Mobjdz, x, T11dz, T12dz, T21dz, Q, Datadz.ObjVec);
if forz>=fodz
    oracleS = set(oracleS, 'FunctionValues', forz); % Value of the ...
       objective
    oracleS = set(oracleS, 'SubGradients', Gforz);  % Subgradient
    oracleS = set(oracleS, 'SubProblemIndex', 1); % Nature of the ...
       cut (Optimality -> 1, Feasibility -> 0)
else
    oracleS = set(oracleS, 'FunctionValues', fodz);
    oracleS = set(oracleS, 'SubGradients', Gfodz);
    oracleS = set(oracleS, 'SubProblemIndex', 1);
end
return
```

A.2 Bode Sensitivity Integral Constraint

```
% Bode Sensitivity Integral
% SISO LTI plant with P-K classic feedback structure
% Open loop TF is rational and has at least 2-pole roll-off
clear;
close all;
% Sensitivity Integral Relations for
% Generic upper bound on sensitivity

% Available Bandwidth
wp = 10;

% Performance Bandwidth Vector
ws_vec = logspace(-1,1,1000);
% Peak Sensitivity Value Vector
M_vec = logspace(0,2,1000);

% RHP pole
p = 0;  % p=0 => Stable Plant

% Order of transfer function in a given frequency range
% First slope:
k1 = 1;
% Second slope:
k2 = 1;
% Third slope:
k3 = 1;

% Epsilon (magnitude of sensitivity at low freq.)
% Assume that epsilon is 0 (or 0+ to be precise)

% ------------------------------------------------
% % Solve for M for different value of ws
% Expression involving M:
% k2*nthroot(M,k2)*nthroot(M,k3)*ws ...
% - nthroot(M,k3)*(-pi*p - k1*ws + k2*ws + k3*wp) + k3*wp <= 0
```

```
% Preallocate
Msoln_vec = NaN*ones(numel(ws_vec),1);
validM_vec = NaN*ones(numel(ws_vec),1);

for ws_ind = 1:numel(ws_vec)
    ws = ws_vec(ws_ind);

    % Parameterized function:
    M_param_func = @(M,ws,wp,p,k1,k2,k3) ...
        k2*nthroot(M,k2)*nthroot(M,k3)*ws ...
        - nthroot(M,k3)*(-pi*p - k1*ws + k2*ws + k3*wp) + k3*wp;
    % "Single" variable function:
    M_singlevar_fun = @(M) M_param_func(M,ws,wp,p,k1,k2,k3);
    % Initial Point:
    M_init = 1;
    % Solve for M:
    try
        M_soln = fzero(M_singlevar_fun,M_init);
    catch
        M_soln = NaN;
    end
    % Store the solution:
    Msoln_vec(ws_ind) = M_soln;

    % For the assumed upper bound on sensitivity, the relations are ...
       valid
    % under certain assumption. This assumption is based on relation
    % between ws and wp.
    validM_vec(ws_ind) = (wp/ws)^((k2*k3)/(k2+k3));
end
% % Plot
figure;
semilogx(ws_vec,mag2db(Msoln_vec),'-b');
grid on; hold on;
semilogx(ws_vec,mag2db(validM_vec),'-r');
title('LB on M vs \omega_s');
ylabel('LB on M (dB)');
xlabel('\omega_s (rad/s)');
plot_axis;
ylim([0 30])

% ------------------------------------------------
% Solve for ws for different value of M
% Expression involving ws:
% ws <= (pi*p + k3/nthroot(M,k3)*wp - k3*wp)/(-k1 - k2*nthroot(M,k2) ...
   + k2)

% Preallocate
ws_vec = NaN*ones(numel(ws_vec),1);
validws_vec = NaN*ones(numel(ws_vec),1);

for M_ind = 1:numel(M_vec)
    M = M_vec(M_ind);

    % Solve for ws
    ws_soln = (pi*p + k3/nthroot(M,k3)*wp - k3*wp)...
        /(-k1 - k2*nthroot(M,k2) + k2);
```

```
    % Store ws
    ws_vec(M_ind) = ws_soln;

    % For the assumed upper bound on sensitivity, the relations are ...
       valid
    % under certain assumption. This assumption is based on relation
    % between ws and wp.
    validws_vec(M_ind) = wp/(nthroot(M,k2)*nthroot(M,k3));
end
% % Plot
figure;
semilogy(mag2db(M_vec),ws_vec,'-b');
grid on; hold on;
semilogy(mag2db(M_vec),validws_vec,'-r');
title('UP on \omega_s vs M');
xlabel('M (dB)');
ylabel('UB on Perf. Bandwidth \omega_s (rad/s)');
plot_axis;
ylim([1e-1 1e1])
```

A.3 Sensitivity Peak Bounds Due to RHP Zero

```
% Sensitivity Limits Imposed by RHP Zero:
% Sensitivity bounds based on RHPZ and Weighting function
% SISO LTI plant with P-K classic feedback structure
% Stable plant

% ------------------------------------------------------------
clear;
s = tf('s');

% Available Bandwidth
wp = 1e5;

% RHP zero
z = 10;

% % % Generic Weighting FUnction:
% W = ((s+ws*nthroot(M,k2))^k2 * (s+wp/nthroot(M,k3))^k3)/ ...
%     ((s+ws*nthroot(epsilon,k1))^k1 * (s+ws)^(k2-k1) * (s+wp)^k3);

% Order of transfer function in a given frequency range
% First slope:
k1 = 1;
% Second slope:
k2 = 1;
% Third slope:
k3 = 1;

% Performance Bandwidth Vector
ws_vec = logspace(-1,1,1000);
% Peak Sensitivity Value Vector
M_vec = logspace(0,2,1000);

% Epsilon (magnitude of sensitivity at low freq.)
```

```
% Assume that epsilon is 0 (or 0+ to be precise)

% ------------------------------------------------
% % Solve for M for different value of ws
% Expression involving M and ws
% z^k1 * (z+ws)^(k2-k1) * (z+wp)^k3 ...
%     - (z+ws*nthroot(M,k2))^k2 * (z+wp/nthroot(M,k3))^k3 == 0;

% Preallocate
Msoln_vec = NaN*ones(numel(ws_vec),1);
validM_vec = NaN*ones(numel(ws_vec),1);

for ws_ind = 1:numel(ws_vec)
    ws = ws_vec(ws_ind);


    M_param_func = @(M,ws,wp,z,k1,k2,k3) ...
        (z+ws*nthroot(M,k2))^k2 * (z+wp/nthroot(M,k3))^k3 ...
        - z^k1 * (z+ws)^(k2-k1) * (z+wp)^k3;
    % "Single" variable function:
    M_singlevar_fun = @(M) M_param_func(M,ws,wp,z,k1,k2,k3);
    % Initial Point:
    M_init = 1;
    % Solve for M (root of nonlinear expression):
    try
        M_soln = fzero(M_singlevar_fun,M_init);
    catch
        M_soln = NaN; %
    end
    % Store the solution:
    Msoln_vec(ws_ind) = M_soln;

    % For the assumed upper bound on sensitivity, the relations are ...
       valid
    % under certain assumption. This assumption is based on relation
    % between ws and wp.
    validM_vec(ws_ind) = (wp/ws)^((k2*k3)/(k2+k3));
end
% % Plot
figure;
semilogx(ws_vec,mag2db(Msoln_vec),'-b');
grid on; hold on;
semilogx(ws_vec,mag2db(validM_vec),'-r');
title('LB on M vs \omega_s');
ylabel('LB on M (dB)');
xlabel('\omega_s (rad/s)');
plot_axis;
ylim([-10 60])


% ------------------------------------------------------------
% Solve for ws for different value of M
% Expression involving ws:
% ws <= (pi*p + k3/nthroot(M,k3)*wp - k3*wp)/(-k1 - k2*nthroot(M,k2) ...
   + k2)

% Preallocate
```

```
ws_vec = NaN*ones(numel(ws_vec),1);
validws_vec = NaN*ones(numel(ws_vec),1);

for M_ind = 1:numel(M_vec)
    M = M_vec(M_ind);

    ws_param_func = @(ws,M,wp,z,k1,k2,k3) ...
        (z+ws*nthroot(M,k2))^k2 * (z+wp/nthroot(M,k3))^k3 ...
        - z^k1 * (z+ws)^(k2-k1) * (z+wp)^k3;
    % "Single" variable function:
    ws_singlevar_fun = @(ws) ws_param_func(ws,M,wp,z,k1,k2,k3);
    % Initial Point:
    ws_init = 1;
    % Solve for ws (root of nonlinear expression):
    try
        ws_soln = fzero(ws_singlevar_fun,ws_init);
    catch
        ws_soln = NaN; %
    end
    % Store ws
    ws_vec(M_ind) = ws_soln;

    % For the assumed upper bound on sensitivity, the relations are ...
       valid
    % under certain assumption. This assumption is based on relation
    % between ws and wp.
    validws_vec(M_ind) = wp/(nthroot(M,k2)*nthroot(M,k3));
end
% % Plot
figure;
semilogy(mag2db(M_vec),ws_vec,'-b');
grid on; hold on;
semilogy(mag2db(M_vec),validws_vec,'-r');
title('UP on \omega_s vs M');
xlabel('M (dB)');
ylabel('UB on Perf. Bandwidth \omega_s (rad/s)');
plot_axis;
ylim([1e-2 1e1])
```

A.4 SISO Unstable and Non-Minimum Plant Example Using HINFSTRUCT

```
% MATLAB Code to design controller for SISO unstable and non-minimum
% phase plant using hinfstruct
% A classic P-K structure OR a hierarchical inner-outer structure can
% be selected

clear;
close all;

% Classic P-K structure: set flag=1, or
% Hierarchical inner-outer structure: set flag=2.
flag = 1;

% Transfer functions variable
s = tf('s');
```

```
%% Plant
% Plant
p = 1;
z = 10;

P = ((z-s)/(s-p))*((p)/(z));
[n_e,n_u] = size(P);

%% Weighting Functions
Eps=0.01;
Ms=2; wb=0.02;
% W1 = tf([1/Ms wb], [1 wb*Eps]);
W1 = ss(1);
% Mu=0.1/3; wbu=7500;
% W2 = [tf([1 wbu*Mu],[Eps wbu])];
W2 = ss(0);
My=2; wbc=0.02;
% W3 = tf([1 wbc/My], [Eps wbc]);
W3 = ss(1);

% Number of exogenous inputs and outputs
n_w = size([W1; W2; W3],2);
n_z = size([W1; W2; W3],1);

%% Define tunable controller structure

% rolloff at a
a = 100;

if flag == 1
    % % Classic P-K structure
    % ------------------------------------------------
    K_norolloff = ltiblock.pid('K_norolloff','p');
    % K_norolloff = tunablePID('K_norolloff','pi');

    % K = gk/s*(s+zk)*(a/(s+a))^2;
    K = K_norolloff;
    % K = series(K_norolloff,(a/(s+a))^2);

elseif flag == 2
    % % Hierarchical inner-outer structure
    % ------------------------------------------------
    Ko_norolloff = ltiblock.pid('Ko_norolloff','p');
    Ko = Ko_norolloff;
%     Ko = series(Ko_norolloff,(a/(s+a))^2);

    Ki_norolloff = ltiblock.pid('Ki_norolloff','p');
    Ki = Ki_norolloff;
%     Ki = series(Ki_norolloff,(a/(s+a))^2);
end

%% Define closed-loop interconnection

if flag == 1
    % Classic P-K structure
    % ------------------------------------------------
```

```
    % Using feeedback command
    CL0 = blkdiag(W1*feedback(1,P*K),W3*feedback(P*K,1));
    % CL0 = blkdiag(feedback(1,P*K));
    %
    % % % % Generalized Plant - MSO
    % % systemnames='P W1 W2 W3';
    % % inputvar=['[r{' int2str(n_e) '};u{' int2str(n_u) '}]'];
    % % outputvar=['[W1; W2; W3; r-P]'];
    % % input_to_P='[u]';
    % % input_to_W1='[r-P]';
    % % input_to_W2='[u]';
    % % input_to_W3='[P]';
    % % cleanupsysic='yes';
    % % GenP_mso=sysic;
    % %
    % % [Ag,Bg,Cg,Dg]=ssdata(GenP_mso);
    % % % [Ag,Bg,Cg,Dg]=ssdata(GenP_mso_io);
    % %
    % % % % Matrix blocks of Generalized Plants for LMI
    % % % A=Ag;
    % % % B1=Bg(:,1:n_w);
    % % % B2=Bg(:,n_w+1:end);
    % % % C1=Cg(1:n_z,:);
    % % % C2=Cg(n_z+1:end,:);
    % % % D11=Dg(1:n_z,1:n_w);
    % % % D12=Dg(1:n_z,n_w+1:end);
    % % % D21=Dg(n_z+1:end,1:n_w);
    % % % D22=Dg(n_z+1:end,n_w+1:end);
    % %
    % % % Num. of states of GenP
    % % nx_genp = size(Ag,1);
    % % CL0 = lft(GenP_mso,K);

elseif flag == 2
    % Hierarchical inner-outer structure
    % ---------------------------------------------------
    P.InputName = 'u';
    P.OutputName = 'y';
    Ko.InputName = 'e';
    Ko.OutputName = 'uo';
    Ki.InputName = 'y';
    Ki.OutputName = 'ui';
    W1.InputName = 'e';
    W1.OutputName = 'z1';
    W3.InputName = 'y';
    W3.OutputName = 'z3';
    sum_outer=sumblk('e=r-y',1);
    sum_inner=sumblk('u=uo-ui',1);
    WS = connect(P,Ki,Ko,W1,W3,sum_outer,sum_inner,'r','z1');
    WT = connect(P,Ki,Ko,W1,W3,sum_outer,sum_inner,'r','z3');
    CL0 = blkdiag(WS,WT);

end
%% Solve $H_{\infty}$ problem with hinfstruct
opts = hinfstructOptions('Display','final','MaxIter',100,...
    'RandomStart',50);%,'TolGain',1e-7);
[CL,gam1] = hinfstruct(CL0,opts);  % CL is tuned version of CL0
```

```
if flag == 1
    % % Classic P-K structure
    % % ---------------------------------------------------
    % Get proportional and integral gains
    kp = CL.Blocks.K_norolloff.Kp.Value;
    ki = CL.Blocks.K_norolloff.Ki.Value;
    kd = CL.Blocks.K_norolloff.Kd.Value;
    tau = CL.Blocks.K_norolloff.Tf.Value;
    % Form the final controller
    K_norolloff = kp + kd*(s/(tau*s+1)) + ki/s;
    K = K_norolloff;
    % series(K_norolloff,(a/(s+a))^2);

elseif flag == 2
    % Hierarchical inner-outer structure
    % ---------------------------------------------------
    % Ko:
    % Get proportional and integral gains
    kop = CL.Blocks.Ko_norolloff.Kp.Value;
    koi = CL.Blocks.Ko_norolloff.Ki.Value;
    % Form the final controller
    Ko_norolloff = kop + koi/s;
    Ko = Ko_norolloff;
%     Ko = series(Ko_norolloff,(a/(s+a))^2);

    % Ki:
    % Get proportional and differential gains
    kip = CL.Blocks.Ki_norolloff.Kp.Value;
    kid = CL.Blocks.Ki_norolloff.Kd.Value;
    tau = CL.Blocks.Ki_norolloff.Tf.Value;
    kii = CL.Blocks.Ki_norolloff.Ki.Value;
    % Form the final controller
    Ki_norolloff = kip + kid*(s/(tau*s+1)) + kii/s;
    Ki = Ki_norolloff;
%     Ki = series(Ki_norolloff,(a/(s+a))^2);

end

%% Analyze OL and CL maps

if flag == 1
    % % Classic P-K structure
    % % ---------------------------------------------------
    [Lo,Li,So,Si,To,Ti,KS,PS] = f_CLTFM(P,K);
    S = So;
    T = To;
    zpk(T)

    % Plot S and T
    wvec = logspace(-4,3,1000);
    figure; sigma(So,wvec); grid on; hold on; sigma(To,wvec);
    plot_axis;
    [hL,hObj]=legend('S','T');
    plot_legend(hL,hObj);

elseif flag == 2
```

```
    % % Hierarchical inner-outer structure
    % % ------------------------------------------------
    P.InputName = 'u';
    P.OutputName = 'y';
    Ko.InputName = 'e';
    Ko.OutputName = 'uo';
    Ki.InputName = 'y';
    Ki.OutputName = 'ui';
    sum_outer=sumblk('e=r-y',1);
    sum_inner=sumblk('u=uo-ui',1);

    Pmod = connect(P,Ki,sum_inner,'uo','y');
    zpk(Pmod)

    S = connect(P,Ki,Ko,sum_outer,sum_inner,'r','e');
    T = connect(P,Ki,Ko,sum_outer,sum_inner,'r','y');
    zpk(T)

    % Plot S and T
    wvec = logspace(-4,2,1000);
    figure; sigma(S,wvec); grid on; hold on;
    sigma(T,wvec);
    sigma(inv(W1),wvec);
    sigma(inv(W3),wvec);
    title('Sensitivity and Complementary Sensitivity')
    plot_axis;
    [hL,hObj]=legend('S','T','W1^{-1}','W3^{-1}');
    plot_legend(hL,hObj);
end
```

A.5 Pareto Optimality Example Using FMINCON

```
% Illustrative Example: Weighted Sensitivities vs Tradeoff Parameter
% Using nonlinear optimization solver
% % Assumptions:
% P is dynamic
% K is static
% 2X2 system
% K is of the form [k11 k12; k21 k22]

clear all;
close all;

% Transfer function variable
s = tf('s');

% Plant definition
P = 1/s*[10 9; 9 8];

% Weighting functions
% % W1 and Wd1
Eps=0.01; Ms=2; wb1=1; wb2=1;
W1 = [tf([1/Ms wb1], [1 wb1*Eps]) 0; 0 tf([1/Ms wb2], [1 wb2*Eps])];
Eps=0.01; Ms=2; wb1=2; wb2=2;
Wd1 = [tf([1/Ms wb1], [1 wb1*Eps]) 0; 0 tf([1/Ms wb2], [1 wb2*Eps])];
```

```
% W2 and Wd2
W2 = ss([0.5 0; 0 2]);
Wd2 = ss([1 0; 0 1]);

% Constraint values
cvalvec = [10; 1000; 0.0001; 10000];

% rho vector
rho_vec = 0:0.05:1;

% Bounds on optimization variable
k_ub = 20;
k_lb = -20;
kvec_ub = k_ub*ones(4,1);
kvec_lb = k_lb*ones(4,1);

% Initial point for the first iteration
kvec_init = [-8; 9; 9; -10];

% Loop for all the different values of rho
for ind = 1:numel(rho_vec)
    rho = rho_vec(ind);

    % Call the fmincon solver function
    [kvec_soln,fval,flg] = fmincon(@(x) obj_func(x,P,W1,Wd1,rho),...
        kvec_init,[],[],[],[],kvec_lb,kvec_ub,...
        @(c) constr_func(c,P,W2,Wd2,cvalvec));

    % Controller parameters obtained
    k11 = kvec_soln(1);
    k12 = kvec_soln(2);
    k21 = kvec_soln(3);
    k22 = kvec_soln(4);
    K = [k11 k12; k21 k22]

    % Store the relevant control properties
    K_store{ind} = K;
    [Lo,Li,So,Si,To,Ti,KS,PS] = f_CLTFM(ss(P),ss(K));
    NormInf = mag2db([hinfnorm(So), hinfnorm(Si), hinfnorm(KS), ...
        hinfnorm(PS), hinfnorm(To), hinfnorm(Ti)])
    norm1_store(ind) = hinfnorm(W1*So)
    norm2_store(ind) = hinfnorm(Wd1*Si)
    So_BW_20 = max(getGainCrossover(So,0.1));
    To_BW_20 = max(getGainCrossover(To,0.1));
    constr_store(:,ind) = [hinfnorm(W2*KS); hinfnorm(Wd2*PS); ...
        So_BW_20; To_BW_20]

    % Initial point for next iteration
    kvec_init = kvec_soln;

end

figure; plot(rho_vec,mag2db(norm1_store),'-*'); grid on;
hold on;
plot(rho_vec,mag2db(norm2_store),'-*')
plot_axis;
title('||W*S||_{\infty} vs \rho');
```

```
[hL,hObj]=legend('||W_1*S_e||_{\infty}','||W_4*S_c||_{\infty}');
plot_legend(hL,hObj)
ylabel('||W*S||_{\infty} (dB)');
xlabel('\rho');

figure; plot(mag2db(norm1_store),mag2db(norm2_store),'-*'); grid on;
plot_axis;
ylabel('||W_4*S_c||_{\infty} (dB)');
xlabel('||W_1*S_e||_{\infty} (dB)');
title('||W_4*S_c||_{\infty} vs ||W_1*S_e||_{\infty}');

%%
function fval = obj_func(kvec,P,W1,Wd1,rho)
% % Objective Function
% Inputs: kvec P W1 Wd1 rho
% Output: fval

% Form the controller
k11 = kvec(1);
k12 = kvec(2);
k21 = kvec(3);
k22 = kvec(4);
K = [k11 k12; k21 k22];

% Compute the closed loop maps
[~,~,So,Si,~,~,~,~] = f_CLTFM(ss(P),ss(K));

% Objective Function
% fval = rho*hinfnorm(W1*So) + (1-rho)*hinfnorm(Wd1*Si);
fval = max(rho*hinfnorm(W1*So),(1-rho)*hinfnorm(Wd1*Si));
end

function [cineq,ceq] = constr_func(kvec,P,W2,Wd2,cvalvec)
% % Constraint Function
% Inputs: kvec P W2 Wd2
% Output: cineq ceq

% Form the controller
k11 = kvec(1);
k12 = kvec(2);
k21 = kvec(3);
k22 = kvec(4);
K = [k11 k12; k21 k22];

% Compute the closed loop maps
[~,~,So,Si,To,Ti,KS,PS] = f_CLTFM(ss(P),ss(K));

% Check the bandwidths
if ~isempty(getGainCrossover(So,0.1))
    So_BW_20 = max(getGainCrossover(So,0.1));
else
    So_BW_20 = Inf;
end;
if ~isempty(getGainCrossover(To,0.1))
    To_BW_20 = max(getGainCrossover(To,0.1));
else
    To_BW_20 = Inf;
```

```
end;

% Inequality Constraints
cineq = [hinfnorm(W2*KS) - cvalvec(1);
    hinfnorm(Wd2*PS) - cvalvec(2);
    -So_BW_20 + cvalvec(3);
    To_BW_20 - cvalvec(4)
    ];

% No Equality Constraint
ceq = [];
end
```

A.6 μ-Synthesis Using DK-Iteration

```
% MATLAB code for mu-synthesis using DK-iteration technique
% Uses MATLAB's dksyn command in Robust Control Toolbox

clear; close all;

% Transfer function variable
s = tf('s');

%% Plant model
P = 1/s * [10 9; 9 8];
[n_e,n_u] = size(P);

%% Define uncertainty blocks
InputUnc = ultidyn('InputUnc',[2 2],'Bound',1);
OutputUnc = ultidyn('OutputUnc',[2 2],'Bound',1);

%% Weighting Functions
% Weight on Divisive Uncertainty at Input
Wi = 1*eye(n_u);
% Weight on Divisive Uncertainty at Output
Wo = 1*eye(n_e);
% Weight on Se
rho = 1;
Eps=0.01;
Ms=2; wb=0.1;
W1 = tf([1/Ms wb], [1 wb*Eps])*eye(n_e);
% Weight on KSe
W2 = ss(0.1)*eye(n_u);

%% Generalized plant
systemnames = 'P W1 W2 Wi Wo';
inputvar = '[di(2); do(2); w(2); u(2)]';
outputvar = '[Wi; Wo; W1; W2; w-P+do]';
input_to_P = '[u-di]';
input_to_W1 = '[w-P+do]';
input_to_W2 = '[u-di]';
input_to_Wi = '[u-di]';
input_to_Wo = '[P-do]';
sysoutname = 'GenP';
cleanupsysic = 'yes';
```

```
sysic;
% GenP = minreal(ss(GenP));
GenP_unc = lft([InputUnc zeros(2); zeros(2) OutputUnc],GenP);

%% dksyn command
[K_dksyn,CL_dksyn,Bnd_dksyn,Info_dksyn] = dksyn(GenP_unc,2,2);

%% mu-analysis
% del_I, del_O for robust stability (RS)
BlockStructureRS = [2 2; 2 2];
% % del_I, del_O for NP
% BlockStructureNP = [2 2];
% % del_I, del_O, del_P for RP
% BlockStructureRP = [BlockStructureRS; BlockStructureNP];

wvec1=logspace(-3,3,1000);  % freq vec for Mu-analysis
N = lft(GenP,K_dksyn);
Nf = frd(N,wvec1);
% mu for RS:
MuData=mussv(Nf(1:sum(BlockStructureRS(:,1)),1:sum(BlockStructureRS(:,1))),BlockStructu
    % Pick the channels corresponding to del_I and del_O. Reject ...
   the channel corresponding to del_P
muRS(1:length(wvec1))=MuData(1,1).ResponseData(1,1,:);
% % mu for RP:
% MuData=mussv(Nf,BlockStructureRP);
% muRP(1:length(wvec1))=MuData(1,1).ResponseData(1,1,:);
% % mu for NP:
% ...
   MuData=mussv(Nf(end-sum(BlockStructureNP(:,1))+1:end,end-sum(BlockStructureNP(:,1))+
    % Pick the channels corresponding to del_I and del_O. Reject ...
   the channel corresponding to del_P
% muNP(1:length(wvec1))=MuData(1,1).ResponseData(1,1,:);

figure;
semilogx(wvec1,mag2db(muRS)); grid on;
title('\mu for Robust Stability');
ylabel('\mu');
xlabel('Frequency (rad/s)');
plot_axis
% plot_legend(hL,hObj)
ylim([-40 20]);

%% CL maps
[Lo,Li,So,Si,To,Ti,KS,PS] = f_CLTFM(P,K_dksyn);

wvec=logspace(-3,3,10000);  % freq vec for plotting

figure; sigma(So,wvec); grid on; hold on; sigma(inv(W1),wvec);
title('S_e');
[hL,hObj]=legend('S_e','W_1^{-1}');
plot_axis
plot_legend(hL,hObj)
ylim([-100 20]);

figure; sigma(KS,wvec); grid on; hold on; sigma(inv(W2),wvec);
title('KS_e');
[hL,hObj]=legend('KS_e','W_2^{-1}');
```

```
plot_axis
plot_legend(hL,hObj)
ylim([-100 20]);
```

A.7 Forming Closed Loop Maps

Classic P-K structure:

```
function [Lo,Li,So,Si,To,Ti,KS,SP] = f_CLTFM(P,K)

% OL and CL frequency responses (ss based)
% Works for MIMO P and K
% Inputs:
% P: Plant in state space form
% K: Control in state space form
% Outputs:
% Lo, Li: Open loop tfs in ss
% So,Si,To,Ti,KS,SP: Closed loop tfs in ss

[Ap, Bp, Cp, Dp] = ssdata(P);
n_e = size(P,1);
n_u = size(P,2);
n_p = size(P,'order');
[Ak, Bk, Ck, Dk] = ssdata(K);
n_k = size(K,'order');

%% Lo = PK
A_Lo = [Ap Bp*Ck; zeros(n_k,n_p) Ak];
B_Lo = [Bp*Dk; Bk];
C_Lo = [Cp Dp*Ck];
D_Lo = Dp*Dk;
Lo = ss(A_Lo,B_Lo,C_Lo,D_Lo);

%% Li = KP
A_Li = [Ak Bk*Cp; zeros(n_p,n_k) Ap];
B_Li = [Bk*Dp; Bp];
C_Li = [Ck Dk*Cp];
D_Li = Dk*Dp;
Li = ss(A_Li,B_Li,C_Li,D_Li);

%% Mo
Mo = inv(eye(n_e)+Dp*Dk);
%% Mi
Mi = inv(eye(n_u)+Dk*Dp);

%% So = inv(I+PK)
A_So = [Ap-Bp*Dk*Mo*Cp Bp*Ck-Bp*Dk*Mo*Dp*Ck; -Bk*Mo*Cp Ak-Bk*Mo*Dp*Ck];
B_So = [Bp*Dk*Mo; Bk*Mo];
C_So = [-Mo*Cp -Mo*Dp*Ck];
D_So = Mo;
So = ss(A_So,B_So,C_So,D_So);

%% Si = inv(I+KP)
A_Si = [Ak-Bk*Dp*Mi*Ck Bk*Dp*Mi*Dk*Cp-Bk*Cp; Bp*Mi*Ck Ap-Bp*Mi*Dk*Cp];
B_Si = [-Bk*Dp*Mi; Bp*Mi];
C_Si = [Mi*Ck -Mi*Dk*Cp];
```

```
D_Si = Mi;
Si = ss(A_Si,B_Si,C_Si,D_Si);

%% To = PKinv(I+PK)
A_To = [Ap-Bp*Dk*Mo*Cp Bp*Ck-Bp*Dk*Mo*Dp*Ck; -Bk*Mo*Cp Ak-Bk*Mo*Dp*Ck];
B_To = [Bp*Dk*Mo; Bk*Mo];
C_To = [Mo*Cp Mo*Dp*Ck];
D_To = Mo*Dp*Dk;
To = ss(A_To,B_To,C_To,D_To);

%% Ti = inv(I+KP)KP
A_Ti = [Ak-Bk*Dp*Mi*Ck Bk*Dp*Mi*Dk*Cp-Bk*Cp; Bp*Mi*Ck Ap-Bp*Mi*Dk*Cp];
B_Ti = [-Bk*Dp*Mi; Bp*Mi];
C_Ti = [Mi*Ck -Mi*Dk*Cp];
D_Ti = -Dk*Dp*Mi;
Ti = ss(A_Ti,B_Ti,C_Ti,D_Ti);

%% KS
A_ks = [Ap-Bp*Dk*Mo*Cp Bp*Ck-Bp*Dk*Mo*Dp*Ck; -Bk*Mo*Cp Ak-Bk*Mo*Dp*Ck];
B_ks = [Bp*Dk*Mo; Bk*Mo];
C_ks = [-Dk*Mo*Cp Ck-Dk*Mo*Dp*Ck];
D_ks = Dk*Mo;
KS = ss(A_ks,B_ks,C_ks,D_ks);

%% SP
A_sp = [Ak-Bk*Dp*Mi*Ck Bk*Dp*Mi*Dk*Cp-Bk*Cp; Bp*Mi*Ck Ap-Bp*Mi*Dk*Cp];
B_sp = [-Bk*Dp*Mi; Bp*Mi];
C_sp = [Mo*Dp*Ck Mo*Cp];
D_sp = Mo*Dp;
SP = ss(A_sp,B_sp,C_sp,D_sp);
```

Inner-Outer Hierarchical Structure:

```
function ...
    [Lo,Li,So,Si,To,Ti,Tru,PS,Tniy,Tniu]=f_CLMapInnerOuter_generic...
     (P,Ki,Ko,Mi)

% Computes the open and closed loop maps for inner-outer loop
% configuration
% Mi is a matrix with following properties
% num of col = num of states
% num of row = num of states being fed back in inner loop
% Eg: feeding back states 3 and 4:
% Mi = [0 0 1 0 0 0;
%       0 0 0 1 0 0];

[Ap, Bp, Cp, Dp] = ssdata(P);
[Ai, Bi, Ci, Di] = ssdata(Ki);
[Ao, Bo, Co, Do] = ssdata(Ko);

% Open loop
%% Lo
A_Lo=[Ap+Bp*Di*Mi                    Bp*Co                          -Bp*Ci;
      zeros(size(Ao,1),size(Ap,2)) Ao                             ...
      zeros(size(Ao,1),size(Ai,2));
      Bi*Mi                        zeros(size(Ai,1),size(Ao,2)) Ai];
```

```
B_Lo=[Bp*Do;
      Bo;
      zeros(size(Bi,1),size(Bo,2))];
C_Lo = [Cp-Dp*Di*Mi  Dp*Co -Dp*Ci];
D_Lo = Dp*Do;

Lo=ss(A_Lo,B_Lo,C_Lo,D_Lo);

%% Li
A_Li=[ Ap       zeros(size(Ap,1),size(Ao,2)) ...
    zeros(size(Ap,1),size(Ai,2));
       -Bo*Cp    Ao                          ...
          zeros(size(Ao,1),size(Ai,2));
       Bi*Mi     zeros(size(Ai,1),size(Ao,2)) Ai ...
                                  ];

B_Li=[Bp;
      -Bo*Dp;
      zeros(size(Ai,1),size(Bp,2))];

C_Li=[-Do*Cp-Di*Mi Co -Ci];
D_Li= -Do*Dp;
Li=ss(A_Li,B_Li,C_Li,D_Li);

%% Closed loop
Q =inv(eye(size(Do,1)) + Do*Dp);

Amat=[Ap+Bp*Q*(Do*Dp*Di*Mi-Do*Cp)-Bp*Di*Mi          Bp*Q*Co          ...
    -Bp*Ci+Bp*Q*Do*Dp*Ci     ;
      Bo*Dp*Di*Mi-Bo*Cp-Bo*Dp*Q*(Do*Dp*Di*Mi-Do*Cp) Ao-Bo*Dp*Q*Co    ...
      Bo*Dp*Ci-Bo*Dp*Q*Do*Dp*Ci;
      Bi*Mi                                        zeros(size(Ai,1),...
      size(Ao,2)) Ai                         ];

%% To
B_To = [Bp*Q*Do                      ;
        Bo-Bo*Dp*Q*Do                ;
        zeros(size(Ai,1),size(Bo,2))];
C_To=[Cp+Dp*Q*(Do*Dp*Di*Mi-Do*Cp)-Dp*Di*Mi       Dp*Q*Co         ...
    Dp*Q*Do*Dp*Ci];
D_To = Dp*Q*Do;
To=ss(Amat,B_To,C_To,D_To);

%% So
D_So=eye(size(Dp,1),size(Bo,2))-Dp*Q*Do;
So=ss(Amat,B_To,-C_To,D_So);

%% KS (Tru, strictly speaking)
C_KS = [Do*Dp*Di*Mi-Do*Cp-Di*Mi Co Do*Dp*Ci-Ci];
D_KS = Q*Do;
Tru=ss(Amat,B_To,C_KS,D_KS);

%% Si
B_Si=[Bp - Bp*Q*Do*Dp              ;
      -Bo*Dp+Bo*Dp*Q*Do*Dp         ;
      zeros(size(Bi,1),size(Dp,2))];
```

```
D_Si=eye(size(Q,1))- Q*Do*Dp;
Si=ss(Amat,B_Si,C_KS,D_Si);

%% Ti
D_Ti = -Q*Do*Dp ;
Ti=ss(Amat,B_Si,C_KS,D_Ti);

%% PS
D_PS = Dp-Dp*Q*Do*Dp ;
PS=ss(Amat,B_Si,C_To,D_PS);

%% Tniy
B_Tniy=[Bp*Q*Do*Dp*Di-Bp*Di;
    Bo*Dp*Di-Bo*Dp*Q*Do*Dp*Di;
    Bi];
D_Tniy=[Dp*Q*Do*Dp*Di-Dp*Di];
Tniy=ss(Amat,B_Tniy,C_To,D_Tniy);

%% Tniu
D_Tniu=Q*Do*Dp*Di-Di;
Tniu=ss(Amat,B_Tniy,C_KS,D_Tniu);
```

Inner-Outer Hierarchical Structure when Same States are Associated with K_o and K_i:

```
function ...
    [Lo,Li,So,Si,To,Ti,Tru,PS,Tniy,Tniu]=f_CLMapInnerOuter_BigK(P,...
    K,Mi)

% Computes the open and closed loop maps for inner-outer loop ...
   configuration
% This code can be used when both Ko and Ki have same A and C matrices.
% In other words, when big K is formed which has both Ko and Ki in it.
% This code must be used rather than f_CLMapInnerOuter_generic.m in ...
   order
% to avoid nonminimality in the system
% Mi is a matrix with following properties
% num of col = num of states
% num of row = num of states being fed back in inner loop
% Eg: feeding back states 3 and 4:
% Mi = [0 0 1 0 0 0;
%       0 0 0 1 0 0];

[Ap, Bp, Cp, Dp] = ssdata(P);
[Ak, Bk, Ck, Dk] = ssdata(K);

Bo=K.b(:,1:size(Cp,1)); % Number of inputs to Ko is same as num of ...
   plant...
% output
Do=K.d(:,1:size(Cp,1));
Bi=K.b(:,size(Cp,1)+1:end); % Inputs to Ki are the remaining inputs ...
   to K...
% Also equal to num of rows in Mi
Di=K.d(:,size(Cp,1)+1:end);
%
% Bo=K.b(:,1:size(Mi,1)); % Number of inputs same as num of rows in Mi
```

```
% Do=K.d(:,1:size(Mi,1));
% Bi=K.b(:,size(Mi,1)+1:end); % Inputs are the remaining inputs to K...
% Also equal to num of plant output
% Di=K.d(:,size(Mi,1)+1:end);

% Open loop
%% Lo
A_Lo=[Ap-Bp*Di*Mi                  Bp*Ck;
      -Bi*Mi                       Ak  ];
B_Lo=[Bp*Do;
      Bo  ];
C_Lo = [Cp-Dp*Di*Mi Dp*Ck];
D_Lo = Dp*Do;

Lo=ss(A_Lo,B_Lo,C_Lo,D_Lo);

%% Li
A_Li=[ Ap           zeros(size(Ap,1),size(Ak,2));
      -Bo*Cp-Bi*Mi  Ak                          ];
B_Li=[Bp;
      -Bo*Dp];
C_Li=[-Do*Cp-Di*Mi Ck];
D_Li= -Do*Dp;
Li=ss(A_Li,B_Li,C_Li,D_Li);

%% Closed loop
Q =inv(eye(size(Dp,1)) + Dp*Do); %

Amat=[Ap-Bp*Do*Q*Cp+Bp*Do*Q*Dp*Di*Mi-Bp*Di*Mi  Bp*Ck-Bp*Do*Q*Dp*Ck;
      -Bo*Q*Cp+Bo*Q*Dp*Di*Mi-Bi*Mi              Ak-Bo*Q*Dp*Ck     ];

%% To
B_To = [Bp*Do*Q;
        Bo*Q  ];
C_To=[Cp-Dp*Do*Q*Cp+Dp*Do*Q*Dp*Di*Mi-Dp*Di*Mi  Dp*Ck-Dp*Do*Q*Dp*Ck];
D_To = Dp*Do*Q;
To=ss(Amat,B_To,C_To,D_To);

%% So
D_So=eye(size(Dp,1))-Dp*Do*Q;
So=ss(Amat,B_To,-C_To,D_So);

%% KS (Tru, strictly speaking)
C_KS = [-Do*Q*Cp+Do*Q*Dp*Di*Mi-Di*Mi Ck-Do*Q*Dp*Ck];
D_KS = Do*Q;
Tru=ss(Amat,B_To,C_KS,D_KS);

%% Si
B_Si=[Bp-Bp*Do*Q*Dp;
      -Bo*Q*Dp    ];
D_Si=eye(size(Do,1))-Do*Q*Dp;
Si=ss(Amat,B_Si,C_KS,D_Si);

%% Ti
D_Ti =-Do*Q*Dp ;
Ti=ss(Amat,B_Si,C_KS,D_Ti);
```

```
%% PS
D_PS = Dp-Dp*Do*Q*Dp ;
PS=ss(Amat,B_Si,C_To,D_PS);

%% Tniy
B_Tniy=[Bp*Do*Q*Dp*Di-Bp*Di;
    Bo*Q*Dp*Di-Bi          ];
D_Tniy=-Dp*Di+Dp*Do*Q*Dp*Di;
Tniy=ss(Amat,B_Tniy,C_To,D_Tniy);

%% Tniu
D_Tniu=Do*Q*Dp*Di-Di;
Tniu=ss(Amat,B_Tniy,C_KS,D_Tniu);
```

A.8 Modifying the Appearance of Plots

```
function plot_axis
% This function sets the axes and line properties
% Puts grid and adjusts grid transparency
% Adjusts the LineWidth of the graph
% Adjusts the FontSize of axes

grid on;
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
h_axes = findobj(gcf, 'type', 'axes');
set(h_axes,'LineWidth',1,'FontSize',12,'GridAlpha',0.25);


function plot_legend(hL,hObj)
% This function sets the legend properties
% Adjusts the LineWidth and FontSize

set(hL,'FontSize',12);
hTL=findobj(hObj,'type','line'); % get the lines, not text
set(hTL,'LineWidth',2) % set linewidth
hTL=findobj(hObj,'type','Text'); % get the text
set(hTL,'FontSize',12) % set fontsize
% ax = gca; ax.LineWidth = 1; ax.FontSize = 14; ax.GridAlpha = 0.25; ...
    lgd = legend('So','To'); lgd.FontSize = 14;
% hL = gca; hObj = gca;
```

A.9 MIMO Dynamical System Interaction Measures

Condition Number:

```
function cond_num=f_CondNum(G,wvec)
% Code to find condition number of a dynamic system
% Inputs:
%   G: Dynamic system,
%   wvec: frequency sampling points
% Outputs:
```

```
%   Condition number vector
sing_val=sigma(G,wvec);
cond_num=sing_val(1,:)./sing_val(end,:);

% % Plot
% figure; loglog(wvec,cond_num); grid on;
```

Relative Gain Array (RGA):

```
function [RGAMat,RGASumNorm]=f_RGADynSys(G,wvec)
% Code to find rga for a dynamic system
% Inputs:
% G: Dynamic system, wvec: frequency sampling points
% Outputs:
% RGAMat: RGA matrix components for all specified frequencies, ...
    RGASumNorm

% % % Method-1
% RGASumNorm=zeros(length(wvec),1);
% RGAMat=zeros(size(G,1),size(G,2),length(wvec));
% for ii=1:length(wvec)
%     freq=wvec(ii);
%     G_mat=evalfr(G,freq);
%     R=rga(G_mat);
%     RGAMat(:,:,ii)=R;
%     RGASumNorm(ii)=sum(sum(abs(R)));
% end

% % Method-2
% wvec=logspace(1,1,100);
Gss = ss(G);
Gpck = pck(Gss.a,Gss.b,Gss.c,Gss.d);
Gw=frsp(Gpck,wvec);
RGAMat=veval('.*',Gw,vpinv(vtp(Gw)));
RGASumNorm = sum(abs(RGAMat(1:length(wvec),1:2)),2);
```

Optimally Scaled Condition Number:

```
function [condnum_store,L_opt_store,R_opt_store] = f_MinCondNum(...
    ss_Plant,wvec,gamma_vec)

% Function to obtain the minimized condition number of a dynamical ...
    system
% Min cond num at each desired frequency value is obtained
% The value of optimization variable gamma is guessed each time before
% successively solving an LMI. The gamma is looped over several values.
% The minimum condition number obtained over all the gamma values is ...
    chosen
% Set wvec, the frequency points where condnum need to be mimimized
% Set gamma_vec, the values of guess for optimization variable gamma at
% each frequency
%
% Inputs:
% 1) System whose minimized condition number is to be found
% 2) wvec: the frequency points where condnum need to be mimimized
% 3) gamma_vec, the values of guess for optimization variable gamma ...
```

```
   at each
% frequency
% Example:
% wvec=logspace(-3,1,25);
% gamma_vec = logspace(-3,1,50);
% % load FlexNom.mat
% % load Linr-Bolender_NewEng_OldPlm1.mat
% % ss_Plant = HSV_Trim_Data{1,1}.FER_Scaled;
% load('NENP_Rigid');
% ss_Plant=plantRigid(1:2,:);
% % ss_Plant=[1/(s+1) 0; 0 1/(s+2)]*[9 -10; -8 9];
% % ss_Plant=[(10-s)/10 0; 0 1/(s+1)];
%
% Outputs:
% 1) Cell of length same as that of wvec, each with diagonal matrix ...
   matrix
% L of size equal to number of system outputs
% 2) Cell of length same as that of wvec, each with diagonal matrix ...
   matrix
% R of size equal to number of system inputs
% 3) Vector of Minimized Condition Number at every frequency point ...
   given by
% wvec

% addpath(genpath('path to yalmip'))
% addpath(genpath('path to SeDuMi'))

% clear;
% warning off;
% s=tf('s');

[n_y,n_u] = size(ss_Plant);

% wvec=1e-1;
% Preallocate:
condnum_store = NaN*ones(1,length(wvec));
L_opt_store{length(wvec)} = [];
R_opt_store{length(wvec)} = [];

for j=1:length(wvec)

    M = bode(ss_Plant,wvec(j));
    eps = 1e-4;

    for jj = 1:length(gamma_vec)
        g = gamma_vec(jj); % this is the optimization objective
        tic;
        if cond(M) > 1
            P = diag(sdpvar(n_y,1));
            Q = diag(sdpvar(n_u,1));
            MID = M'*P*M;
            C1 = [ Q <= MID, MID <= g^2*Q, Q>eps*eye(2), P>eps*eye(2)];
            % C1 = [Q <= MID, MID <= gamma^2*Q, Q>eps*eye(2)];

            % sdpsettings('bmibnb.roottight',[0|1])
            sdpset = sdpsettings('solver','sedumi','verbose',1); ...
                %,'showprogress',1);
```

```
            % solvesdp(C1,g^2,sdpset);
            diagnostics = optimize(C1,g^2,sdpset);

            P = double(P);
            Q = double(Q);

            PT = isnan(P);
            QT = isnan(Q);
            PT1 = find(PT == 1);
            QT1 = find(QT == 1);
            PT2 = sum(PT1);
            QT2 = sum(QT1);
            if PT2 ==0 && QT2 ==0
                L = sqrtm(P);
                R = Q^(-0.5);
                condnum(jj) = cond(L*M*R);
                L_cur_gam{jj} = L;
                R_cur_gam{jj} = R;
            else
                condnum(jj) = Inf;
                L_cur_gam{jj} = [];
                R_cur_gam{jj} = [];
            end
            clear PT QT PT1 QT1 PT2 QT2
        else
            condnum = 1;
        end

        toc;
    end

    [condnum_store(j),ind] = min(condnum);
    clear condnum;
    L_opt_store{j}=L_cur_gam{ind};
    R_opt_store{j}=R_cur_gam{ind};
end
```

A.10 Phase of MIMO System

```
function [phase_vec] = f_MIMOPhase(Plnt,wval_vec)
% Phase of SISO/MIMO Square Plant
%
% Inputs:
%   Plnt: Dynamical system (usually tf/zpk/ss)
%   wvec: [Optional] Frequency vector for obtaining phase at each
%         of those points
%         [Default] wvec = logspace(-3,3,1e3)
% Outputs:
%   phase_vec: Computed phase of system (in deg)
%
% For SISO systems, for getting the phase same as usual Bode phase
% plot, pick wi=-ui
% See Jie Chen, Multivar Gain-Phase ... , 1998
```

```
% % Also see Freudenberg Book
% Here, it is assumed that the reference vector wi=ui.

% Frequency vector: Assign Default if not provided
if nargin < 2
    wval_vec = logspace(-3,3,1e3);
end

% System
% s = tf('s');
%
% Plnt = [1/(s+1) 0; 0 1/(s+2)] * [9 -10; -8 9];
% % Plnt = [1/(s+1) 0; 0 1/(s+1)];

% sval = 1i*0.5;

for sval_ind = 1:length(wval_vec)

    sval = wval_vec(sval_ind);
    Plnt_jw = evalfr(Plnt,1i*sval);

    [u,sv,v] = svd(Plnt_jw);

    for singval_ind = 1:size(Plnt,1)

        phase_sv(sval_ind,singval_ind) = angle(u(:,singval_ind)'...
            *v(:,singval_ind))*180/pi;
        % fprintf('Angle (deg) corresponding to sv1')
        % angle(u(:,1)'*v(:,1))*180/pi

        % phase_sv2(sval_ind) = angle(u(:,2)'*v(:,2))*180/pi;
        % % fprintf('Angle (deg) corresponding to sv2')
        % % angle(u(:,2)'*v(:,2))*180/pi
    end

end

figure;
for singval_ind = 1:size(Plnt,1)
    semilogx(wval_vec,phase_sv(:,singval_ind));
    hold on;
end
title('\angle u_i^H v_i');
ylabel('(deg)');
xlabel('Frequency (rad/s)');
% legend('sv1','sv2');
plot_axis;
% figure;
% semilogx(wval_vec,phase_sv1,wval_vec,phase_sv2);
% title('\angle u_i^H v_i');
% ylabel('(deg)');
% xlabel('Frequency (rad/s)');
% legend('sv1','sv2');
% plot_axis;
```